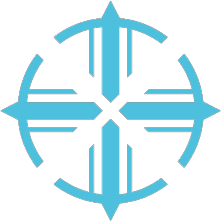
# PONTIFICIA UNIVERSIDAD CATOLICA DEL ECUADOR FACULTAD DE INGENIERIA

ESCUELA DE SISTEMAS



# DISERTACIÓN PREVIA A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS Y COMPUTACIÓN

DESARROLLO DE UNA APLICACIÓN WEB PARA LA GESTIÓN ACADÉMICA, CASO DE ESTUDIO: “ESCUELA FISCOMISIONAL MERCEDARIA SAN JOSÉ DE EL TEJAR”

# AUTOR:

ALEX SEBASTIÁN FLORES CHICAIZA

# DIRECTOR:

NELSON ESTEBAN SALGADO REYES

# QUITO, 2023

## RESUMEN

El presente trabajo presenta una aplicación web factible para la gestión y control académico de la Escuela Fiscomisional Mercedaria San José de El Tejar, en la ciudad de Quito. La propuesta comprende el desarrollo e implementación de un sistema que optimice el tiempo de registro de estudiantes, docentes, cursos, registro de matriculación y calificaciones de cada periodo lectivo que, en dicho establecimiento educativo se lleva de forma manual.

El aplicativo de la aplicación web está desarrolla en un lenguaje de programación orientado a objetos El desarrollo de la aplicación de gestión académica soluciona los problemas manuales de pérdida de registros en papel, confusión en las fichas de los estudiantes y docente y fichas traspapeladas.

## DEDICATORIA

## AGRADECIMIENTOS

**Tabla de contenidos**

[Resumen I](#_bookmark0)

[Dedicatoria II](#_bookmark1)

[Agradecimientos III](#_bookmark2)

1. [Aspectos Introductorios 1](#_bookmark3)
   1. [Datos del Centro Educativo ... 1](#_bookmark4)
      1. [Misión 1](#_bookmark5)
      2. [Visión 1](#_bookmark6)
   2. [Justificación 6](#_bookmark8)
   3. [Planteamiento Del Problema 7](#_bookmark9)
   4. [Alcance 8](#_bookmark10)
   5. [Objetivos 9](#_bookmark11)
      1. [Objetivo General 9](#_bookmark12)
      2. [Objetivos Específicos 9](#_bookmark13)
2. [Fundamento Teórico 10](#_bookmark14)
   1. [Sistemas Web 10](#_bookmark15)
   2. [Arquitectura Web 10](#_bookmark16)
      1. [Arquitectura Cliente – Servidor 12](#_bookmark18)
      2. [Arquitectura En Capas 13](#_bookmark19)
      3. [Arquitectura De Microservicios 16](#_bookmark22)
   3. [Servicios Web 17](#_bookmark23)
      1. [Servicio Web Soap 17](#_bookmark24)
      2. [Servicio Web Restful 18](#_bookmark25)
   4. [Base De Datos 19](#_bookmark27)
   5. [Tecnologías 21](#_bookmark28)
      1. [Back-End 21](#_bookmark29)
      2. [Front-End 22](#_bookmark30)
   6. [Herramientas 22](#_bookmark31)
      1. [Base De Datos 22](#_bookmark32)
         1. [Postgresql 22](#_bookmark33)
      2. [Backend 23](#_bookmark34)
         1. [Node.Js 23](#_bookmark35)
         2. [Express.Js 23](#_bookmark36)
         3. [Sequelize 23](#_bookmark37)
      3. [Frontend 24](#_bookmark38)
         1. [React.Js 24](#_bookmark39)
      4. [Pruebas 25](#_bookmark42)
         1. [Jest 25](#_bookmark43)
      5. [Despliegue 25](#_bookmark44)
         1. [Amazon Web Services (Aws) 25](#_bookmark45)
         2. [Vercel 26](#_bookmark46)
   7. [Metodología 27](#_bookmark47)
   8. [Metodología De Desarrollo 28](#_bookmark48)
      1. [Metodologías Ágiles 28](#_bookmark49)
      2. [Extreme Programming 30](#_bookmark50)
   9. [Justificación De Las Herramientas y Metodologías Seleccionadas 34](#_bookmark51)
3. [Desarrollo Del Sistema Web Para La Automatización De Procesos Asistenciales 36](#_bookmark52)
   1. [Fase De Exploración 36](#_bookmark53)
      1. [Historias De Usuario 36](#_bookmark54)
      2. [Formalización De Requerimientos 41](#_bookmark55)
         1. [Requerimientos Funcionales Y No Funcionales 41](#_bookmark56)
         2. [Casos De Uso 44](#_bookmark57)
   2. [Fase De Mantenimiento 50](#_bookmark58)
      1. [Fase De Planificación 50](#_bookmark59)
         1. [Definición De Entregables 50](#_bookmark60)
      2. [Fase De Iteraciones 51](#_bookmark61)
         1. [Diseño 52](#_bookmark62)
            1. [Arquitectura 52](#_bookmark63)
            2. [Base de datos 53](#_bookmark64)
         2. [Iteración 1 56](#_bookmark65)
            1. [Historia de usuario 1 56](#_bookmark66)
            2. [Historia de usuario 2 58](#_bookmark67)
         3. [Iteración 2 62](#_bookmark68)
            1. [Historia de usuario 3 62](#_bookmark69)
            2. [Historia de usuario 4 67](#_bookmark70)
         4. [Iteración 3 70](#_bookmark71)
            1. [Historia de usuario 5 71](#_bookmark72)
            2. [Historia de usuario 6 76](#_bookmark73)
         5. [Iteración 4 81](#_bookmark74)
            1. [Historia de usuario 7 82](#_bookmark75)
            2. [Historia de usuario 8 85](#_bookmark76)
      3. [Fase De Producción 90](#_bookmark77)
         1. [Despliegue 90](#_bookmark78)
         2. [Aceptación 90](#_bookmark79)
4. [Conclusiones Y Recomendaciones 91](#_bookmark80)
   1. [Conclusiones 91](#_bookmark81)
   2. [Recomendaciones 93](#_bookmark82)

[Referencias 94](#_bookmark83)

[Anexos 98](#_bookmark84)

[***Manual De Usuario 1***](#_bookmark85)

**Índice de figuras**

Figura 1 Clasificación de procesos asistenciales. 2

Figura 2. Diagrama de procesos, Gestión de citas. 3

Figura 3. Diagrama de procesos, Recepción del paciente 4

Figura 4. Diagrama de procesos, Consultas médicas. 5

Figura 5 Tipos de Arquitectura 11

Figura 6 Arquitectura monolítica 12

Figura 7 Arquitectura cliente – servidor 12

Figura 8 Arquitectura en capas. 13

Figura 9 Arquitectura orientada a eventos EDA 14

Figura 10 Arquitectura Orientada a Servicios. 15

Figura 11 Arquitectura de microservicios. 16

Figura 12. Proceso cualitativo. 27

Figura 13 Fases de XP 32

Figura 14. Distribución de las herramientas seleccionadas 36

Figura 15 Formato de historias de usuario. 37

Figura 16 Historia de usuario No. 1 38

Figura 17 Historia de usuario No. 2 38

Figura 18 Historia de usuario No. 3 39

Figura 19 Historia de usuario No. 4 39

Figura 20 Historia de usuario No. 5 40

Figura 21 Historia de usuario No. 6 40

Figura 22 Historia de usuario No. 7 41

Figura 23 Historia de usuario No. 8 41

Figura 24 Nivel general 45

Figura 25 RF1. Ingreso al sistema 46

Figura 26 RF2. Administración de Pacientes. 46

Figura 27 RF3. Agendamiento de Citas. 47

Figura 28 RF4. Administración de Historias Clínicas. 47

Figura 29 RF5. Administración de Evoluciones. 48

Figura 30 RF6. Administración de Usuarios. 48

Figura 31 RF7. Administración de Médicos. 49

Figura 32 RF8. Administración de Consultorios. 49

Figura 33 Planificación inicial del proyecto. 50

Figura 34 Definición de entregables. 51

Figura 35. Arquitectura del sistema 53

Figura 36. Modelo conceptual 54

Figura 37. Modelo físico. 55

Figura 38. Historia de usuario 1, Login. 56

Figura 39. Historia de usuario 1, Restablecer contraseña 57

Figura 40. Historia de usuario 1, Pruebas unitarias. 57

Figura 41. Historia de usuario 2, Crear paciente 58

Figura 42. Historia de usuario 2, Editar paciente 59

Figura *43*. Historia de usuario 2, Eliminar paciente 59

Figura 44. Historia de usuario 2, Listado de pacientes. 60

Figura *45*. Historia de usuario 2, Detalle de paciente por id. 60

Figura 46. Historia de usuario 2, Buscar pacientes. 61

Figura *47*. Historia de usuario 2, Pruebas unitarias. 61

Figura *48*. Historia de usuario 3, Crear cita médica 63

Figura 49. Historia de usuario 3, Editar cita médica 64

Figura 50. Historia de usuario 3, Eliminar cita médica 64

Figura 51. Historia de usuario 3, Detalle de cita por id. 65

Figura 52. Historia de usuario 3, Calendario de citas. 65

Figura 53. Historia de usuario 3, Buscar citas. 65

Figura 54. Historia de usuario 3, Enviar recordatorios de citas. 66

Figura 55. Historia de usuario 3, Pruebas unitarias. 66

Figura 56. Historia de usuario 4, Crear historia clínica 67

Figura 57. Historia de usuario 4, Editar historia clínica 68

Figura 58. Historia de usuario 4, Eliminar historia clínica 68

Figura 59. Historia de usuario 4, Detalle de historia clínica por id. 69

Figura 60. Historia de usuario 4, Buscar historias clínicas. 69

Figura 61. Historia de usuario 4, Listado de historias clínicas. 69

Figura 62. Historia de usuario 4, Pruebas unitarias. 70

Figura 63. Historia de usuario 5, Crear evolución. 72

Figura 64. Historia de usuario 5, Editar evolución. 72

Figura 65. Historia de usuario 5, Eliminar evolución. 73

Figura 66. Historia de usuario 5, Detalle de evolución por id. 73

Figura 67. Historia de usuario 5, Listado de evoluciones. 74

Figura 68. Historia de usuario 5, Buscar evoluciones. 74

Figura 69. Historia de usuario 5, Generar certificado médico. 75

Figura 70. Historia de usuario 5, Generar certificado médico. 75

Figura 71. Historia de usuario 5, Pruebas unitarias. 76

Figura 72. Historia de usuario 6, Crear usuario. 77

Figura 73. Historia de usuario 6, Editar usuario. 78

Figura 74. Historia de usuario 6, Eliminar usuario. 78

Figura 75. Historia de usuario 6, Detalle de usuario por id. 79

Figura 76. Historia de usuario 6, Listado de usuarios. 79

Figura 77. Historia de usuario 6, Buscar usuarios. 79

Figura 78. Historia de usuario 6, Recuperar contraseña 80

Figura 79. Historia de usuario 6, Actualizar datos. 80

Figura 80. Historia de usuario 6, Cambiar contraseña 80

Figura 81. Historia de usuario 6, Pruebas unitarias. 81

Figura 82. Historia de usuario 7, Crear médico. 82

Figura 83. Historia de usuario 7, Editar médico. 83

Figura 84. Historia de usuario 7, Eliminar médico. 83

Figura 85. Historia de usuario 7, Detalle de médico por id. 84

Figura 86. Historia de usuario 7, Listado de médicos. 84

Figura 87. Historia de usuario 7, Buscar médicos. 84

Figura 88. Historia de usuario 7, Pruebas unitarias. 85

Figura 89. Historia de usuario 8, Crear consultorio. 86

Figura 90. Historia de usuario 8, Editar consultorio. 86

Figura 91. Historia de usuario 8, Eliminar consultorio. 87

Figura 92. Historia de usuario 8, Detalle de consultorio por id. 87

Figura 93. Historia de usuario 8, Listado de consultorios. 88

Figura 94. Historia de usuario 8, Buscar consultorios. 88

Figura 95. Historia de usuario 8, Pruebas unitarias. 89

**Índice de tablas**

Tabla 1 Clasificación de las bases de datos. 20

Tabla 2 Requerimientos funcionales. 42

Tabla 3 Requerimientos no funcionales. 43

Tabla 4 Requerimientos mínimos. 43

Tabla 5. Historia de usuario 1, Endpoints. 56

Tabla 6. Historia de usuario 2, Endpoints. 58

Tabla 7. Historia de usuario 3, Endpoints. 62

Tabla 8. Historia de usuario 4, Endpoints. 67

Tabla 9. Historia de usuario 5, Endpoints. 71

Tabla 10. Historia de usuario 6, Endpoints. 76

Tabla 11. Historia de usuario 7, Endpoints. 82

Tabla 12. Historia de usuario 8, Endpoints. 85

## ASPECTOS INTRODUCTORIOS

## DATOS DEL CENTRO EDUCATIVO

La Escuela Fiscomisional Mercedaria San José de El Tejar es un centro educativo católico mercedario con más de 60 años de experiencia en enseñanza integral y de calidad en la ciudad de Quito, Ecuador. En la actualidad posee alrededor de 140 estudiantes de distintos sectores y con niveles de estudio según lo especifica el Ministerio de Educación: niños y niñas con edades comprendidas entre 3 a 12 años. Posee 11 profesionales que desempeñan funciones en el área educativa y forman además parte de la gestión administrativa de la escuela.

## MISIÓN

Formar niños y niñas con una amplia conciencia social, crítica, creativa y participativa para mejorar la calidad de vida de cada uno de estos; mediante sólidos conocimientos científicos, tecnológicos y prácticas de valores cristianos y mercedarios.

## JUSTIFICACIÓN

Los centros educativos fiscomisionados del cantón Quito cuentan con un presupuesto limitado, mayoritariamente, destinado el 70% al pago de salarios a docentes, 20% servicios básicos, 5% infraestructura, 3% materiales de enseñanza y, en menor medida – 2% – a la actualización de sistemas e insumos tecnológicos (Villa, Vargas y Merino 2018). En tal sentido, los procesos administrativos y de gestión académica tales como: registro del personal docente, matriculación de estudiantes, asignación de horarios de clase y registro de calificaciones en la Escuela Fiscomisional Mercedaria San José de El Tejar son realizados manualmente en documentos físicos.

Procesos deficientes que traen como consecuencia traspiés en los indicadores del inicio, proceso y finalización del año lectivo, debido a la pérdida de información de los registros en papel, confusión en las fichas de los estudiantes y docentes y, fichas traspapeladas. Ante esta situación y, en concordancia al escenario actual globalizado que cuenta con herramientas de desarrollo tecnológico para crear aplicaciones que faciliten la gestión de los procesos educativos, la documentación física-manual de la Escuela Fiscomisional Mercedaria San José de El Tejar puede ser sustituida por documentación digitalizada; permitiéndoles optimizar tiempo y recursos en la administración y gestión académica.

De ahí que, la propuesta del presente proyecto se enfoca en desarrollar una aplicación web eficiente e integradora, capaz de brindar soporte a la gestión académica del centro educativo San José de El Tejar. Es decir, un sistema que permita llevar un control automatizado en un menor tiempo, con la menor cantidad de errores y por consecuencia, mínimos recursos y horas de trabajo ya que el sistema web hace uso de un interfaz amigable, fácil de comprender, administrar y acceder.

Todo ello, gracias al uso de una metodología XP basada en distintos valores, principios y técnicas que se centran en la velocidad y la simplicidad con ciclos de desarrollo cortos y con menos documentación. Asimismo, el ciclo de vida con el que cuenta esta metodología: Exploración, Planificación, Iteraciones y Puesta en producción hace que la tasa de errores sea mínima y que las respuestas ante cambios inesperados sean más veloces, manteniendo el bienestar entre los desarrolladores y los clientes y, por consecuente, asegurando el éxito del proyecto académico.

La contribución académica e importancia del presente trabajo de titulación radica en integrar sistemas informáticos con la educación; por lo que, el sistema web impulsará el uso de herramientas tecnológicas en el ámbito de la educación para obtener datos reales y exactos, lo cual serán almacenados en la base de datos del sistema académico y permitirá conocer reportes de la información guardada. Igualmente, sentará un precedente en la educación fiscomisional en las que se pretenda diseñar, desarrollar e implementar un sistema de gestión educativa con características semejantes a la Escuela Fiscomisional Mercedaria San José de El Tejar.

## PLANTEAMIENTO DEL PROBLEMA

Ante el actual avance tecnológico, los sistemas y aplicativos informáticos se han convertido en una pieza clave en distintas áreas de cualquier institución u organización, especialmente en la educación (Santana y Parrales 2016) donde es fundamental contar con accesibilidad, confidencialidad e integridad de datos. Las soluciones informáticas llegan a ser entonces, un factor importante dentro de “las diferentes actividades de gestión educativa y en especial cuando son orientadas hacia servicios que brindan atención a niños y niñas” (Montenegro 2006, 23).

Las aplicaciones web brindan grandes ventajas en las instituciones académicas porque eliminan acciones manuales, resuelven problemas de concurrencia, impiden la pérdida de datos, permiten acceder al sistema desde cualquier lugar de la red disponible dentro del centro educativo y mejoran la calidad del servicio y la satisfacción del académico o administrativo en operaciones como: matriculación, registros de notas, horarios, entre otros.

Bajo este contexto, la Escuela Fiscomisional Mercedaria San José de El Tejar presenta en la actualidad falencias en el manejo funcional de la información y su integración en la matriculación de los estudiantes, registro del alumnado y docentes, asignación de materias-horarios y registro de calificaciones. De esta manera, se vuelve imprescindible el desarrollo e implementación de una aplicación web para la gestión académica de la Escuela San José de El Tejar.

Los principales procesos académicos que manejará son:

* Administración y registro de profesores.
* Administración y registro de alumnos.
* Administración y registro de materias.
* Administración y registro de calificaciones.
* Administración y registro de horarios.

La propuesta de proyecto técnico que se plantea en este trabajo final de grado se orienta a reemplazar el trabajo administrativo y académico manual llevado a cabo en la Escuela Fiscomisional San José de El Tejar por un trabajo ágil, óptimo e innovador capaz de eliminar cualquier malversación de recursos y la probabilidad de pérdida de información. Un trabajo que se traduce en una prestación de servicios con un nivel de calidad apropiado.

En función a lo expuesto, se identifica la siguiente pregunta principal del proyecto de titulación:

* ¿Cómo afecta el desarrollo de un sistema web para la gestión de procesos académicos en la Escuela Fiscomisional Mercedaria San José de El Tejar?

Y las siguientes preguntas secundarias:

* + ¿Es posible reducir la probabilidad de pérdida de información de los estudiantes y docentes? ¿Cómo es el trato de la información de los estudiantes y docentes?
  + ¿Es posible reducir el tiempo de administración y registro de alumnos, docentes, calificaciones, materias y horarios?
  + ¿Es factible la implementación del aplicativo web en la Escuela Fiscomisional Mercedaria San José de El Tejar?

## ALCANCE

El uso de aplicaciones web en la educación es una práctica de desarrollo informático que cada vez va siendo más común debido a su capacidad de crear un entorno administrativo más personalizado, adaptado a las necesidades concretas de cada alumno y docente, fomentando una administración y registro de datos ágil y óptima. De ahí que, la adquisición o desarrollo de aplicativos webs no es un lujo, es una necesidad para brindar prontitud a la gestión educativa, así como, la capacidad de generar a los clientes una imagen institucional fuerte, de prestigio y confiable (Alcocer 2012).

De esta manera, el alcance de este proyecto esta direccionado a entregar una aplicación web para el control y gestión académica en la Escuela Fiscomisional Mercedaria San José de El Tejar. Un sistema que cuenta con un diseño característico, adecuado a las necesidades del personal docente, administrativo y estudiantil y que, contará con los siguientes módulos:

* Matriculación.
* Registro de docentes.
* Registro de calificaciones.
* Registro de materias y horarios.

## OBJETIVOS

## OBJETIVO GENERAL

* + - * Diseñar una aplicación web que gestione los datos académicos de la Escuela Fiscomisional Mercedaria San José de El Tejar.

## OBJETIVOS ESPECÍFICOS

* + - * Analizar los requerimientos de la Escuela Fiscomisional Mercedaria San José de El Tejar para el diseño y desarrollo del sistema.
      * Disponer de una aplicación web para agilizar el proceso de matriculación de los estudiantes.
      * Facilitar la asignación de cursos y horarios a los docentes de manera óptima y eficiente.
      * Visualización de calificaciones de forma rápida y concisa.

## FUNDAMENTO TEÓRICO

## SISTEMAS WEB

Los sistemas web o también conocidos aplicaciones web son herramientas informáticas con un aspecto muy similar a las páginas web que se ve normalmente, pero en realidad, los sistemas web van un paso más allá debido a sus altas y potenciales funcionalidades que brindan respuestas a casos particulares demandados por los usuarios.

En palabras de Tim Berners-Lee (1999), las aplicaciones web son sistemas de distribución de información basados en hipertextos o hipermedios enlazados y accesibles a través del internet. Su facilidad de uso se ve reflejada en los costos – hardware y software –, en la rapidez de obtención de información, en la optimización de tareas y en la gestión (López 2022) debido a que pueden ser usados por varios usuarios al mismo tiempo. Al estar toda la información centralizada no es necesario compartir pantallas o enviar emails con documentos adjuntos; son accesibles desde cualquier lugar.

Paralelamente, Benjamín Aumaille (2002) enfatiza que los sistemas web son un conjunto de recursos web que participan en el funcionamiento de la propia aplicación; es decir, “a diferencia de los sistemas de escritorio, estos no se encuentran adosados o instalados sobre un sistema operativo. Por el contrario, están alojados en un servidor en internet o sobre una intranet” (Berzosa 2022).

Así pues, Molina Rios et al. (2017) argumentan que los sistemas web se pueden utilizar en cualquier navegador web (Chrome, Firefox, internet explorer, etc) sin importar el sistema operativo y, utilizan el formato estándar HTML (HyperText Markup Language o Lenguaje de Hipertextos) para efectuar las peticiones que el usuario desea.

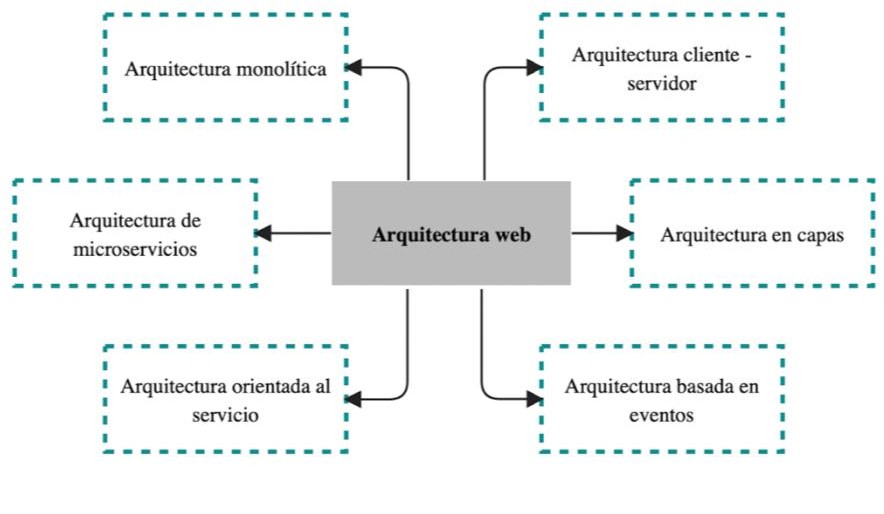
## ARQUITECTURA WEB

La planificación y diseño de la arquitectura web permite la planificación técnica, funcional y visual de un sitio web, antes de que sea diseñado, desarrollado e implementado. De acuerdo con Iván de Souza (2021), es el medio para delinear y desplegar los sistemas, mediante la descripción de patrones y técnicas que aseguran la obtención de una aplicación correctamente estructurada.

Por lo tanto, la arquitectura web “es un medio en contante cambio que permite la comunicación entre diferentes usuarios y la interacción técnica (interoperabilidad) entre distintos sistemas y subsistemas” (De Souza 2021). Una importancia múltiple que acarrea diferentes seis tipos de arquitectura como se puede apreciar en la siguiente figura:

**Figura**

*Tipos de Arquitectura*



**Nota.** Adaptada de (Red Hat Inc., 2021).

## ARQUITECTURA CLIENTE – SERVIDOR

Estilo arquitectónico compuesto por dos componentes, el proveedor y el consumidor. El proveedor es uno o varios servidores que brinda una serie de servicios o recursos al cliente-consumidor a través de una red de cimunicación. En esta arquitectura, el servidor se comunica con el consumidor generalmente por TCP (Protocolo de Control de Transmisión). Una comunicación continua y bidireccional que permite generar peticiones y respuestas en forma de texto HTML, imágenes u otros (Duarte Vega, 2016).

Los beneficios del uso de la arquitectura cliente-servidor son:

1. Centralización: el servidor fungirá como única fuente de la verdad, lo que impide que los clientes conserven información desactualizada (Blancarte 2018).

ii. Instalación: generalmente, la aplicación es simple, libre de dependencias; facilitando la instalación al cliente.

iii. Seguridad: el servidor generalmente está protegido por *firewall* o subredes que impiden ataques externos a la base de datos o los recursos.

iv. Separación de responsabilidades: la arquitectura cliente servidor permite implementar la lógica de negocio de forma separada del cliente (Blancarte 2018).

v. Portabilidad: una de las ventajas de tener dos aplicaciones es que se puede desarrollar cada parte para correr en diferentes plataformas. Por ejemplo, el servidor solo en Linux, mientras que el cliente podrías ser multiplataforma (Blancarte 2018).

Estas son pues, las especificaciones de la arquitectura cliente-servidor y, se puede observar su estructura en la siguiente figura:

Figura 7

*Arquitectura cliente – servidor.*

Diagrama

Descripción generada automáticamente

## ARQUITECTURA EN CAPAS

Este tipo de arquitectura de acuerdo con Juan Peláez (2012) se enfoca en la distribución de roles y responsabilidades de forma jerárquica proveyendo una forma muy efectiva de separación de responsabilidades. En otras palabras, la arquitectura en capas se centra en la separación de la aplicación en capas, como podría ser:

1. **Capa de presentación:** también denominada capa de usuario tiene la capacidad de comunicación únicamente con la capa de negocio y, presenta la interfaz del sistema al usuario.
2. **Capa de negocio:** examina la lógica de servicios, se comunica con la capa de presentación para receptar solicitudes y presentar resultados; asimismo, se comunica con la capa de datos para solicitar el almacenamiento o recuperación datos.
3. **Capa de datos:** formada por uno o más gestores de base de datos, cumple la función de almacenaje y acceso a los datos.

Tres capas capaces de cumplir con el proceso de *layering* o descomposición de un sistema software en sus partes; como lo evidencia la siguiente figura:

**Figura 8**

*Arquitectura en capas.*

Diagrama

Descripción generada automáticamente

Las principales ventajas de la arquitectura basada en capas son:

1. Separación de responsabilidades: permite la separación de preocupaciones (SoC) debido a que cada capa se enfoca en una sola responsabilidad.

ii. Desarrollo: la arquitectura por capas posee un gran renombre debido a su facilidad de implementación y utilización por varias aplicaciones.

iii. Prueba: gracias a la utilización de capas, es posible ir probando de forma individual y por separado cada capa.

iv. Mantenimiento: debido a que cada capa hace una tarea especifica, detectar el origen de un bug es relativamente fácil y sencillo para corregirlo.

v. Seguridad: la utilización de capas permite el aislamiento de los servidores en subredes diferentes; impidiendo posibles ataques.

## ARQUITECTURA DE MICROSERVICIOS

La arquitectura de microservicios es un método de desarrollo de aplicaciones software que funciona como un conjunto de pequeños servicios que se ejecutan de manera independiente y autónoma, proporcionando una funcionalidad de negocio completa (Microsoft learn 2022). La característica principal de esta arquitectura radica en que los servicios pueden ser acoplados de forma flexible debido a que, cada uno de estos, cuenta con un código de base pequeño e independiente.

La Figura presenta la estructura de un sistema distribuido que ejecuta varios servicios en distintos servidores o hosts. Los microservicios se comunican entre sí a través de APIs (Application Programming Interfaces) y, cuentan con sistemas de almacenamiento propios que les ayuda a evitar sobrecargas o caídas de la aplicación (Rodríguez et al., 2019).

**Figura 11**

*Arquitectura de microservicios.*

*Interfaz de usuario gráfica, Texto

Descripción generada automáticamente*

**Nota.** Microsoft learn 2022

Entre las ventajas de la arquitectura de microservicios se encuentra:

1. Agilidad: dado que los microservicios se implementan de forma independiente, resulta más fácil de administrar las correcciones de errores y las versiones de características (Microsoft learn 2022).

ii. Equipos pequeños y centrados: un microservicio es lo suficientemente pequeño; facilitando la comunicación y disminuyendo la sobrecarga.

iii. Base de código pequeña: al no compartir el código ni los almacenes de datos, la arquitectura de microservicios minimiza las dependencias y resulta más fácil agregar nuevas características (Microsoft learn 2022).

iv. Escalabilidad: Los servicios se pueden escalar de forma independiente, lo que permite escalar horizontalmente los subsistemas que requieren más recursos, sin tener que escalar horizontalmente toda la aplicación. Mediante un orquestador como *Kubernetes* o *Service Fabric* se puede empaquetar una mayor densidad de servicios en un solo host, lo que aumenta la eficacia en el uso de los recursos (Microsoft learn 2022).

## SERVICIOS WEB

De acuerdo con IBM (2022), los servicios web son “aplicaciones autónomas modulares que se pueden describir, publicar, localizar e invocar a través de una red” (párr. 1). Tienen como función principal la comunicación; es decir, permite a las computadoras conectadas a internet comunicarse entre sí y con sistemas heterogéneos dentro y fuera de una organización (Severance, 2020).

Dicha comunicación, se realiza a través de mensajes XML que se envían a través de protocolos estándares de internet tales como el protocolo HTTP (Hypertext Transfer Protocol). De esta forma, en lugar de retornar paginas web como respuesta, lo que se hace es devolver un mensaje de respuesta que también se encuentra formateado en XML (Guerrero 2022).

Bajo este contexto, Lamarca (2013) comenta que los servicios web son como un “conjunto de aplicaciones o tecnologías con capacidad para interoperar en la web y capaces de intercambiar datos entre ellas con el fin de ofrecer unos servicios” (párr. 1). Asimismo, reconoce que existe diferentes tipos de servicios web y que, se detallan a continuación:

## SERVICIO WEB SOAP

SOAP (Simple Object Access Protocol) “es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML (Extensible Markup Language)” (López 2013). El SOAP fue creado por Microsoft, IBM y otros y está actualmente bajo el auspicio de la W3C.

El servicio web SOAP es uno de los protocolos más utilizados en los servicios Web, ofreciendo un *framework* de mensajería básica en la cual los *webs services* se puedan construir. Siguiendo a los ingenieros Denis Morejón, Anay Carrillo y Darian Martínez de la Universidad Cienfuegos de Cuba (2016), el SOAP consiste en tres partes: una sobre (*envelope*), contenido y procesamiento del mensaje; un conjunto de reglas de codificación para expresar instancias de tipos de datos; y una convención para representar llamadas a procedimientos y respuestas (párr. 8).

De la misma forma, el protocolo SOAP posee tres características principales:

1. Extensibilidad (seguridad y WS-routing son extensiones aplicadas en el desarrollo).

ii. Neutralidad (SOAP puede ser manejado sobre cualquier protocolo de transporte como: HTTP, SMTP, TCP o JMS).

iii. Independencia (SOAP consiente cualquier modelo de programación).

## SERVICIO WEB RESTFUL

Siguiendo a Sayago et al. (2019), el servicio web RESTful se identifica por ser un diseño basado en la arquitectura REST (*Representational State Transfer*) y, utiliza estándares web tales como: URIs, HTTP, XML y JSON.

El término REST proviene de la tesis doctoral de Roy Fielding publicada en el año 2000. REST es un conjunto de restricciones que, cuando “son aplicadas al diseño de un sistema, crean un estilo arquitectónico de software” (Alonso 2008). Dicho estilo se caracteriza por seguir los siguientes fundamentos:

* Deber ser un sistema cliente-servidor.
* Tiene que ser sin estado, es decir, cada petición al servicio tiene que ser independiente de las demás (Alonso 2008).
* Debe soportar un sistema de cachés.
* Debe poseer un interfaz uniforme para simplificar y desacoplar la arquitectura, permitiendo que cada una de sus partes pueda evolucionar de forma independiente (Alonso 2008).

A partir de dichos principios, el servicio web RESTful es capaz de entablar una comunicación sin mantener estados; esto es, una optimización de almacenamiento en memoria caché. También, exponer los métodos estandarizados GET, POST, PUT y DELETE, así como la reutilización de mecanismos de autenticación, cifrado y autorización por HTTP.

## BASE DE DATOS

De acuerdo con el *European Knowledge Center for information Technology* (2022), la base datos es una herramienta que recopila datos, los organiza y los relaciona para que se pueda hacer una rápida búsqueda y recuperar con ayuda de un computador. Actualmente, las bases de datos también sirven para desarrollar análisis y, las más modernas, tienen motores específicos para generar informes con datos complejos.

De acuerdo con Pulido et al. (2019) y FARGUSZ de la Universidad Central del Ecuador (2020), las características de las bases de datos son:

* Independencia lógica y física de los datos.
* Redundancia mínima.
* Consultas complejas optimizadas.
* Acceso concurrente por parte de múltiples usuarios.
* Integridad de los datos
* Seguridad a los datos frente a usuarios no autorizados.
* Respaldo y recuperación de datos.
* Acceso a través de lenguajes de programación estándar.

En torno a dichas características y a su variedad de forma de trabajo, las bases de datos se pueden clasificar según lo detalla la siguiente Tabla 1:

**Tabla 1**

*Clasificación de las bases de datos.*

**Nota.** Adaptada de Pulido et al., 2019.

## TECNOLOGÍAS

## BACK-END

Dentro del contexto de las aplicaciones web, el back-end se identifica como la parte lógica de la aplicación, es decir, es el encargado de ejecutar aquellos procesos necesarios para que la página o sistema web funcionen de manera adecuada, abarca todas las actividades realizadas del lado del servidor, incluyendo las tareas de base de datos y los servidores de aplicaciones.

Los lenguajes de programación comúnmente utilizados son PHP, Java, Ruby, .NET, Python, entre otros. (Pinto et al., 2016; Jovaldiv, 2016).

## FRONT-END

Se define como la interfaz gráfica de usuario (GUI), se caracteriza por ser la parte de la aplicación web que interactúa con el usuario, razón por la cual debe cumplir con estándares de usabilidad y contar con un diseño adecuado. Generalmente estas tecnologías son desarrolladas en lenguajes como HTML, CSS, JavaScript.

En el ámbito del desarrollo de aplicaciones web, es posible implementar el front-end de una aplicación sin contar con la parte del back-end, sin embargo, comúnmente es el responsable de enviar peticiones y consumir los recursos provistos por el back-end (Jovaldiv, 2016).

## HERRAMIENTAS

Para la selección de las tecnologías a emplear en el proyecto de disertación, se utilizará un cuadro comparativo en el cual se evaluarán distintos lenguajes de programación como candidatos potenciales.

|  |  |  |
| --- | --- | --- |
| **Lenguaje de Programación Framework** | **Ventajas** | **Desventajas** |
| **FrontEnd** | | |
| JavaScript – ReactJS | * React se basa en la idea de componentes, lo que facilita la creación y el mantenimiento de interfaces de usuario modularizadas. * React utiliza un Virtual DOM para mejorar el rendimiento de las actualizaciones de la interfaz de usuario. Esto significa que solo se actualizan las partes de la interfaz que han cambiado, en lugar de toda la página. * Hay una gran cantidad de desarrolladores que facilita bibliotecas y herramientas de terceros | * React puede tener una curva de aprendizaje empinada, especialmente si no están familiarizados con JavaScript moderno y conceptos como JSX. * Es necesario agregar otras bibliotecas y herramientas, lo que puede aumentar el tamaño total de la aplicación. * React no prescribe una arquitectur lo que puede llevar a inconsistencias si no se maneja adecuadamente. a específica, |
| JavaScript – VueJS | * Vue.js es conocido por su curva de aprendizaje suave. Es relativamente fácil de entender y usar. * Vue.js se basa en un enfoque de componentes reutilizables, lo que facilita la creación de interfaces modulares y mantenibles. * Vue.js utiliza directivas en su sintaxis, lo que facilita la vinculación de datos y la manipulación del DOM de manera declarativa. | * Vue.js es menos popular y puede tener menos oportunidades de trabajo y recursos de capacitación disponibles. * Vue.js proporciona una estructura clara y flexible, algunas aplicaciones extremadamente grandes y complejas pueden requerir más personalización y estructuras de arquitectura específicas. |
| Thymeleaf | * Integración que se tiene con el Spring Boot, facilita el desarrollo de aplicación web dinámicas * Al ser un motor de platillas se puede usar documentos HTML generados por aplicación de diseño web y utilizarlos para el desarrollo. | * Su curva de aprendizaje si no se tiene conocimientos base de Spring Boot es complejo * El rendimimiento al tiempo de la ejecución a suele ser mas bajo que frameworks |
| **BackEnd** | | |
| JavaScript – VueJS | * Node.js permite a los desarrolladores utilizar JavaScript tanto en el lado del cliente como en el lado del servidor, lo que facilita la reutilización de código * Node.js se basa en un modelo de E/S sin bloqueo y orientado a eventos, lo que lo hace altamente eficiente y adecuado para aplicaciones de alto rendimiento que manejan múltiples conexiones simultáneas. * Node.js cuenta con una serie de frameworks y bibliotecas populares, como Express.js, que facilitan el desarrollo de aplicaciones web y API de manera rápida y estructurada. | * Node.js es monohilo, lo que significa que ejecuta una sola operación a la vez. Aunque esto es eficiente para operaciones de entrada y salida (E/S), puede ser un problema en aplicaciones intensivas en CPU, * La gestión de errores en Node.js puede ser complicada debido al estilo de programación asincrónica y orientado a eventos. * A medida que las aplicaciones crecen en tamaño y complejidad, el código puede volverse difícil de mantener si no se siguen buenas prácticas de diseño y organización. |
| JAVA - SpringBoot | * Spring Boot simplifica el desarrollo de aplicaciones Java al proporcionar una configuración predefinida y una estructura de proyecto lista para usar. * Spring Boot se basa en el ecosistema de Spring, que ofrece una amplia gama de módulos y herramientas para abordar diversos aspectos de desarrollo empresarial, como la gestión de transacciones, la seguridad, la gestión de datos y más. * Spring Boot facilita la integración con otras tecnologías y bibliotecas mediante la configuración basada en anotaciones y la detección automática de componentes. * Spring Boot utiliza Apache Maven o Gradle para la gestión de dependencias, lo que simplifica la administración de las bibliotecas y facilita la actualización de las mismas. | * Spring Boot simplifica muchas tareas de desarrollo, todavía puede tener una curva de aprendizaje empinada para los desarrolladores nuevos en el ecosistema de Spring * Las aplicaciones Spring Boot pueden ser más grandes en tamaño debido a la inclusión de bibliotecas y características específicas de Spring. * Spring Boot es eficiente, su sobrecarga de abstracción puede afectar ligeramente el rendimiento en comparación con frameworks más ligeros y específicos. |

## BASE DE DATOS

## 2.6.1.1 POSTGRESQL

Citando a Blanch et al., (2018), PostgreSQL es el sistema gestor de bases de datos *Open Source* de más alto nivel, multiplataforma, con capacidad para trabajar con proyectos grandes sin aumentar su complejidad. Posee un *software* libre, con una licencia BSD, compatible con el uso personal o comercial.

PostgreSQL es considerado un sistema empresarial más avanzado de código abierto. Posee dos ventajas fundamentales, primero, su funcionalidad y capacidad permite trabajar con mayores cantidades de datos. Y, segundo, su licencia es totalmente abierta para cualquier uso. De esta manera, PostgreSQL es capaz de funcionar de manera estable en el servidor.

Entre las características más importantes de PostgreSQL se puede detallar las siguientes:

* Emplea un modelo cliente/servidor que le hace un sistema estable, de alto rendimiento, gran flexibilidad y compatibilidad (Aliaga & Miani, 2008; Denzer, 2002).
* Es consciente y tolerante a fallos.
* Es compatible con el modelo relacional, ya que aegura siempre su integridad referencial (Blanch et al., 2018).
* Es capaz de atender a múltiples clientes al mismo tiempo y entregar la misma información de sus tablas, sin bloqueos.
* Soporte a *triggers*, permite definir eventos y generar acciones cuando estos se disparan.
* Soporte para gran cantidad de lenguajes trabajo con vistas.
* Ofrece los tipos de datos habituales en los sistemas gestores, pero además muchos otros que no están disponibles en otros competidores como: direcciones IP, direcciones MAC, Arrays, figuras geométricas, números decimales con presión configurable, entre otros (Blanch et al., 2018).

## BACKEND

## GRADLE

Es una herramienta de gestión y automatización de proyectos de software que se utiliza especialmente para proyectos Java. Destaca por su flexibilidad, rendimiento y capacidad para manejar grandes proyectos. Se configura mediante un DSL basado en Groovy o Kotlin, lo que le permite administrar fácilmente dependencias y crear múltiples proyectos simultáneamente. Además, es extensible con scripts y complementos personalizados y se integra bien con varios IDE y herramientas CI/CD. Gradle es popular por su velocidad y su gran comunidad de usuarios.

## SPRING

Es un marco de desarrollo de aplicaciones Java ampliamente utilizado conocido por su enfoque en la "inversión de control" para facilitar la configuración y administración de aplicaciones. Es versátil y modular, lo que permite a los desarrolladores elegir los componentes que necesitan. Spring ayuda a simplificar la programación empresarial al brindar soporte para inyección de dependencias, transacciones de bases de datos, gestión de aspectos y servicios web. Es popular en el desarrollo de aplicaciones web y empresariales debido a su solidez.

## SPRING BOOT

Spring Boot es una extensión del marco Spring diseñada para simplificar el desarrollo y la implementación de aplicaciones Spring. Proporciona configuración automática, le permite crear aplicaciones independientes y es ideal para el desarrollo de microservicios. Se integra fácilmente con otros componentes de Spring y tiene una amplia compatibilidad con complementos. Es popular por su facilidad de uso y eficiencia y es ampliamente utilizado en la industria.

## FRONTEND

## THYMELEAF

Es un motor de plantillas de aplicaciones web desarrollado en Java. Se usa ampliamente en proyectos Spring MVC, que es un marco popular para aplicaciones web Java. Thymeleaf destaca por su capacidad para manejar archivos HTML del lado del servidor y su sintaxis clara y expresiva. Permite a los desarrolladores crear vistas dinámicas donde los datos generados por el servidor se pueden insertar fácilmente en plantillas HTML. Una característica importante de Thymeleaf es que las plantillas que crea son válidas y pueden mostrarse como páginas HTML estáticas incluso antes de ser procesadas en el servidor. Esto es útil para el diseño y la depuración. Además, Thymeleaf ofrece una amplia integración con Spring, lo que facilita la creación de formularios, la validación de datos y la internacionalización.

## BOOTSTRAP

Es un marco de desarrollo web de código abierto que facilita la creación de sitios web y aplicaciones atractivos y receptivos. Proporciona un sistema de cuadrícula para un diseño responsivo, elementos de interfaz de usuario reutilizables y es compatible con la mayoría de los navegadores. Además, es personalizable y cuenta con varios complementos de JavaScript para agregar funciones adicionales. Es muy utilizado por su facilidad de uso y eficiencia a la hora de crear una interfaz coherente y elegante.

## PRUEBAS

## SPRING TEST

Spring Test es un módulo integrado en el ecosistema Spring que proporciona una amplia gama de herramientas para probar aplicaciones Spring. Este módulo facilita la ejecución de pruebas unitarias y de integración, permitiendo a los desarrolladores probar el comportamiento de sus aplicaciones en un entorno que simula un contexto Spring real. Con Spring Test.

Una característica notable de Spring Test es su conjunto de anotaciones especiales que simplifican la configuración de diferentes tipos de pruebas. Por ejemplo, @SpringBootTest se usa para pruebas de integración que requieren el contexto completo de la aplicación, @DataJpaTest se usa para pruebas orientadas a JPA, @WebMvcTest se usa para pruebas orientadas al controlador MVC, etc. Estas anotaciones ayudan a reducir la configuración y las pruebas. más legible y fácil de mantener. En general, Spring Test es un módulo esencial para cualquier desarrollador que trabaje en el ecosistema Spring, ya que proporciona una forma sólida y flexible de garantizar la calidad y estabilidad de las aplicaciones Spring.

## DESPLIEGUE

## AMAZON WEB SERVICES (AWS)

Plataforma en la nube que surgió en 2006, ofrece más de 200 servicios integrales entre los cuales se encuentran potencia de computación, opciones de almacenamiento, redes y bases de datos que se suministran como una utilidad.

Proporciona una infraestructura altamente fiable, disponible, escalable y de bajo costo (Amazon Web Services Inc, 2018).

Los servicios de AWS que se utilizarán en el presente proyecto son:

* + - * + *RDS:* Amazon Relational Database Service se establece como un servicio de base de datos relacional, sencillo de configurar,

utilizar y escalar, incluye los siguiente motores de bases de datos: Amazon Aurora, PostgreSQL, MySQL, MariaDB, Oracle Database y SQL Server.

Se caracteriza por su alto rendimiento, disponibilidad, seguridad y compatibilidad (Amazon Web Services Inc, 2021b).

* + - * + *EC2:* Amazon Elastic Compute Cloud es un “servicio web que proporciona capacidad informática en la nube segura y de tamaño modificable” (Amazon Web Services Inc, 2021ª, párr. 1), está diseñado para simplificar a los desarrolladores el uso de la informática en la nube a escala web, además, proporciona un control completo sobre los recursos informáticos a nivel de procesador, almacenamiento, red y sistema operativo (Amazon Web Services Inc, 2021a).

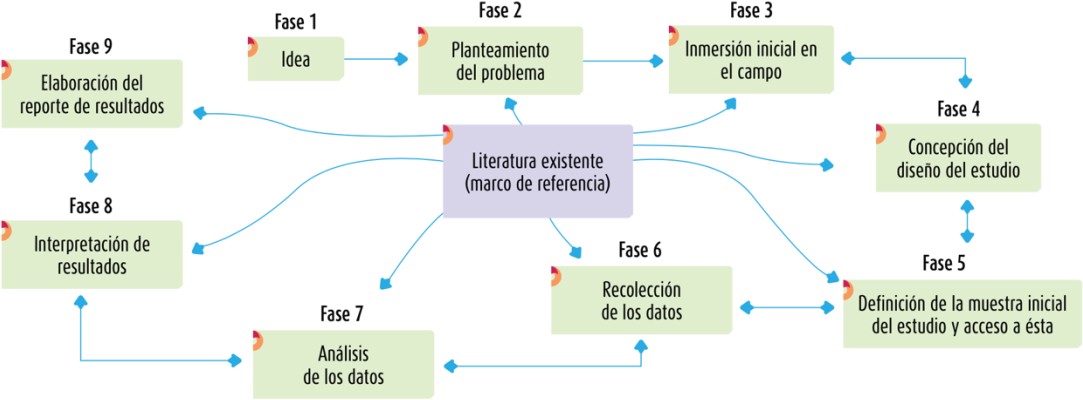
## METODOLOGÍA

De acuerdo con Hernández Sampieri (2014), la metodología cualitativa constituye un proceso circular, que carece de un patrón predecible y estructurado; esto permite retroceder, eludir o ejecutar varias fases de manera simultánea. Es decir, la metodología cualitativa hace que la muestra, la recolección y el análisis sean fases que se realicen prácticamente de manera sincrónica. Así pues, el investigador comienza examinando el mundo social y en este proceso desarrolla “una teoría coherente con los datos, de acuerdo con lo que observa, frecuentemente denominada teoría fundamentada, con la cual observa que ocurre” (Esteberg 2002).

El enfoque cualitativo puede definirse como “un conjunto de prácticas interpretativas que hacen al mundo visible, lo transforman y convierten en una serie de representaciones en forma de observaciones, anotaciones, grabaciones y documentos” (Hernández Sampieri 2014). Un proceso complejo y flexible que la siguiente figura 12 explica el devenir de la metodología cualitativa en 9 fases:

**Figura 12.**

*Proceso cualitativo.*



**Nota.** Tomada de Hernández et al., 2014.

Bajo este contexto, la metodología cualitativa es capaz de extraer el significado de los datos sin la necesidad de reducirlos a números o de interrumpir, alterar o imponer un punto de vista externo, sino tal como los aprecian los actores del sistema social. De acuerdo con Creswell (1997) y Neuman (1994), la metodología cualitativa se caracteriza por:

* Abierta, libre, muy flexible, construido durante la realización del estudio.
* La teoría no se fundamenta en estudios anteriores, sino que se genera a partir de datos obtenidos y analizados.
* Casos individuales, representativos.
* La recolección de los datos está orientada a proveer de un mayor entendimiento.
* Basado en casos o personas y sus manifestaciones.
* Brinda credibilidad, confirmación, valoración y transferencia.

## METODOLOGÍA DE DESARROLLO

* + - 1. **METODOLOGÍA TRADICIONAL**

El enfoque tradicional para el desarrollo de software, a menudo llamado modelo en cascada es un enfoque lineal y secuencial para construir sistemas de software. Se caracteriza por un avance ordenado en diferentes etapas: análisis de requisitos, desarrollo, implementación, pruebas, implementación y mantenimiento. Cada fase debe completarse antes de pasar a la siguiente, lo que implica una planificación detallada y una extensa documentación desde el inicio del proyecto.

En la fase de análisis de requisitos, se identifican y documentan las necesidades y expectativas del sistema. Esta fase sienta las bases para un mayor diseño y desarrollo. En el diseño, se detalla la arquitectura del sistema, incluyendo tanto el diseño de alto nivel (estructura general) como el diseño detallado (operaciones específicas e implementación de cada componente). La implementación implica convertir el diseño en código funcional, seguido de una fase de prueba para verificar la funcionalidad del software y eliminar errores. Una vez que el software ha pasado las pruebas, se implementa en el entorno del usuario final. Posteriormente, durante la fase de mantenimiento, se realizan actualizaciones y correcciones para garantizar la funcionalidad continua y el cumplimiento del software.

Este enfoque funciona mejor para proyectos con requisitos estables y bien definidos. Sin embargo, su rigidez lo hace menos flexible a los cambios durante el desarrollo, lo cual es una desventaja en proyectos dinámicos o proyectos donde los requisitos son ambiguos o pueden cambiar con el tiempo. Por estas razones, si bien el modelo en cascada ha sido fundamental a lo largo de la historia del desarrollo de software, los métodos ágiles han ganado popularidad como una alternativa más adaptativa y orientada a la iteración.

## METODOLOGÍA AGILE

La metodología agile surge en Estados Unidos durante la reunión de expertos desarrolladores de software en 2001 como una forma de gestionar sus proyectos con rapidez y flexibilidad. Así, la metodología agile se convierte en una estrategia integral de gestión de servicios. “Una innovadora forma de trabajar y organizar flujos, que divide los proyectos en partes, permite adaptarse sobre la marcha, complementa y resuelve etapas en poco tiempo” (Salesforce 2022).

De acuerdo con Progressa Lean (2022) y Bahit (2012), la metodología agile consta de principios y valores establecidos en el manifiesto agile. Entre los primordiales se encuentran:

1. Las interacciones de las personas sobre los procesos y las herramientas.
2. Un software en funcionamiento frente a documentación exhaustiva.
3. La participación activa del cliente durante todo el proceso de desarrollo.
4. La capacidad de respuesta ante los cambios e imprevistos.
5. Consentir que los requisitos cambien, incluso en etapas imprevistas del desarrollo.
6. Trabajar de forma coordinada y cercana entre desarrolladores y clientes durante todo el desarrollo del proyecto.
7. Desarrollar proyectos en torno a individuos motivados.
8. Verificar continuamente la excelencia técnica y al buen diseño.
9. Maximizar la cantidad de trabajo no realizado en cada ciclo.
10. Maximizar la calidad del trabajo mediante una retroalimentación continua.

Bajo dichos principios, la metodología agile es capaz de generar respuestas rápidas a las valoraciones que se realizan del propio proyecto, y eso es precisamente lo que define su naturaleza: su carácter ágil (De Tena 2019).

Por su parte, Bahit (2012) expresa que la metodología agile puede clasificarse de la siguiente forma:

* + - * SCRUM
      * Extreme Programming (XP)
      * KANBAN

Una clasificación enfocada a la gestión de proyectos adaptativos para el desarrollo de software; empero, para el desarrollo de la aplicación web para la gestión académica de la “Escuela Fiscomisional Mercedaria San José de El Tejar, se ha seleccionado Extremme Programming, misma que es analizada en el siguiente apartado.

## CUADRO COMPARATIVO DE METODOLOGIAS AGILES

|  |  |  |  |
| --- | --- | --- | --- |
| **METODOLOGIA** | **SCRUM** | **EXTREME PROGRAMMING** | **KANBAN** |
| **VENTEJAS** | * Fomenta la colaboración y la comunicación * Mejora a la adaptabilidad a cambios * Productos de alta calidad * Identifica y se resuelve errores rápidamente en el desarrollo | * Mejorar la calidad del software y la satisfacción del cliente. * Menores costos a largo plazo debido a menos errores. * Fomente la retroalimentación constante. * Facilitar una rápida adaptación al cambio. | * Capacidad para ser flexible y cambiar prioridades rápidamente. * Fácil de implementar y entender. Reduzca el tiempo del ciclo y aumente la eficiencia. * Fomentar la transparencia y visualización del trabajo. |
| **DESVENTAJAS** | * La adaptación de esta metodología para los equipos de desarrollo es compleja * Requiere el compromiso de todos los miembros del equipo * Si los requerimientos son muy dinámicos se vuelve complejo el desarrollo | * Esto requiere una cultura de desarrollo abierta y colaborativa, lo que puede resultar un desafío. * Esto puede requerir más recursos, especialmente en el lado de las pruebas. * No es ideal para equipos grandes o geográficamente dispersos, ya que se requiere una colaboración estrecha. | * Menos estructurado que Scrum, lo que puede resultar un desafío para algunos equipos. * Esto puede llevar a centrarse excesivamente en la finalización de la tarea en lugar de en la calidad del producto. * Se requiere disciplina para mantener el flujo de trabajo y evitar compromisos excesivos. |

## EXTREME PROGRAMMING

La metodología Extreme Programming o mejor conocida como XP, pertenece a la clasificación de la metodología agile y, tiene como objetivo el desarrollo y gestión de proyectos con eficacia, flexibilidad y control (Bello 2021). Una metodología basada en la comunicación, la reutilización del código desarrollado y la realimentación; surgida de las ideas de Kent Beck y Ward Cunningham en 1996 para resolver los problemas de entrega de software de calidad en un periodo de tiempo corto para sus consumidores.

Extreme Programming destaca por su desarrollo de software ágil, una comunicación constante con todos los involucrados en el proyecto y sobre todo la retroalimentación permanente en cada fase que, ayuda a los desarrolladores a ajustarse a los requerimientos cambiantes de los clientes.

En este sentido, siguiendo a Laínez Fuentes (2015) y Montero Molina et al., (2018), la metodología XP se caracteriza por:

* Planificación flexible y abierta.
* Desarrollo iterativo e incremental aportando pequeñas mejoras.
* Utilización en proyectos con equipos pequeños o medianos y en los cuales los requerimientos cambien con un nivel de frecuencia alto.
* El equipo de proyecto se considera como el factor principal de éxito del proyecto.
* El software funciona por encima de la documentación.
* Potencia el trabajo en equipo y minimiza el riesgo en cuanto a costos, tiempo, calidad y alcance.
* Respuesta rápida y eficaz ante posibles imprevistos.

Por su parte, Bahit (2012) y Bello (2021) argumentan que la metodología XP cuenta con cinco valores fundamentales. El objetivo de estos valores es que el equipo de desarrolladores forje una esencia colaborativa para crear un producto de alto nivel. Los cinco valores son:

* + - 1. Comunicación:busca una plena interacción interna entre el equipo desarrollador y el cliente. El objetivo es romper las barreras entre negocio y desarrollo; para ello, se promueve que todos los requisitos sean comunicados y trabajados con el equipo y no mediante documentación (Bello 2021).
      2. Simplicidad:se enfoca en el desarrollo de únicamente lo necesario. Es decir, la metodología XP pone el foco en codificar las necesidades de hoy, no las de un futuro. Conservar la simplicidad involucra mantener la refactorización del código.
      3. Retroalimentación: gracias a los ciclos muy cortos de presentación de resultados por parte de los desarrolladores hacia el cliente, se minimiza el riesgo de tener que hacer partes que no cumplen con las expectativas del cliente.
      4. Respeto:implica el acatamiento responsable del equipo de desarrolladores y el cliente de forma mutua. Esto lleva consigo un mayor ritmo de producción.
      5. Valentía:los desarrolladores deben mostrar la realidad sobre el avance del proyecto. Reconocer los errores tan pronto como se detecten.

A partir de estos valores, inicia el ciclo de vida de la metodología XP. Ambler (2002) divide al ciclo de vida de XP en dos grandes fases. Por un lado, la fase de exploración que incluye el desarrollo inicial; esto es, la definición del alcance del proyecto a partir de los requerimientos del cliente. Por otra parte, se encuentra la fase de mantenimiento que abarca tanto las fases de planificación, iteraciones y producción como las actividades enfocadas a la operación y soporte del sistema:

 Fase de planificación: tiene como objetivo precisar el tiempo y las fechas de entrega para la implementación de las historias de usuario.

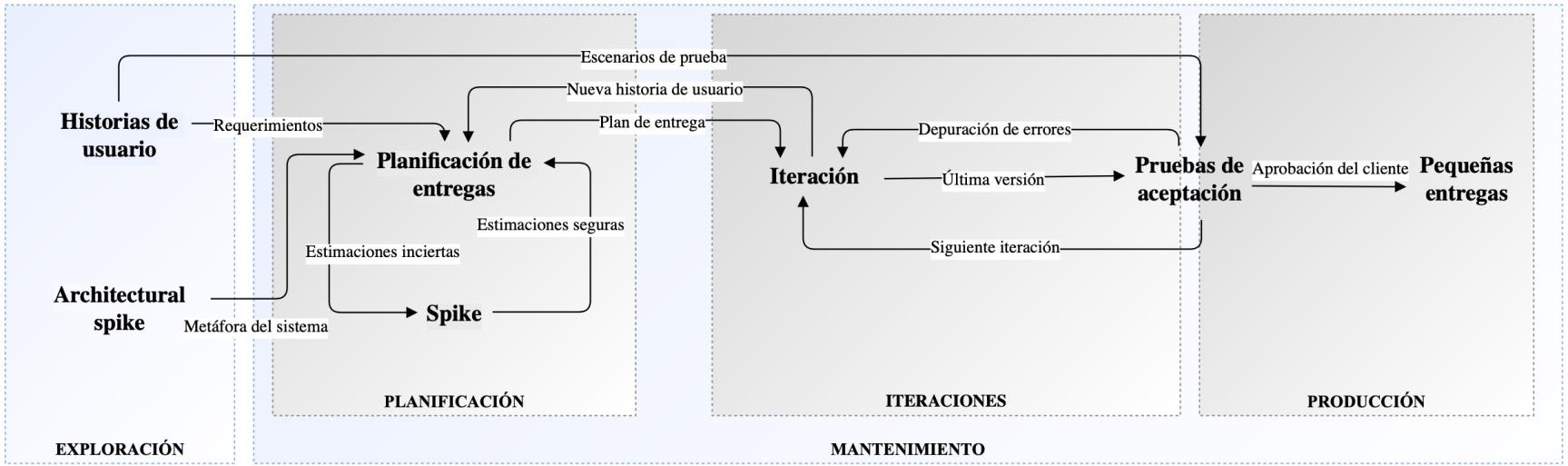
 Fase de iteraciones: fase con mayor necesidad de ahínco debido a su nivel de dificultad y detalle en el modelamiento, programación, testeo e integración de cada una de las historias de los usuarios.

 Fase de puesta en producción: se enfoca en certificar que el sistema esté operativo para el escenario de producción. En esta fase la documentación del sistema, la documentación de las operaciones, la documentación de soporte y la documentación para el usuario es ejecutada.

Fases que se pueden evidenciar de manera más detalla en la siguiente Figura 13:

Desarrollo de una aplicación web para la gestión académica, caso de estudio: “Escuela Fiscomisional Mercedaria San José de El Tejar”.

**Figura 13**

*Ciclo de vida de XP.*

**Nota.** Adaptada de Ambler, 2002.

32

Pontificia Universidad Católica del Ecuador, Facultad de Ingeniería, Carrera de Sistemas y Computación.

## JUSTIFICACIÓN DE LAS HERRAMIENTAS Y METODOLOGÍAS SELECCIONADAS

Para el desarrollo del presente trabajo de disertación se seleccionó la metodología cualitativa debido a que ésta es utilizada en investigaciones enfocadas a evaluar ciertas características de una población o situaciones en específico. Su principal objetivo es la observación sistémica del objeto de estudio a fin de catalogar la información que se observa para que posteriormente pueda ser utilizada y replicada por terceros (Yanez, 2019).

Por su parte para el desarrollo del sistema web se optó por la metodología XP; esta, se caracteriza por su simplicidad, comunicación constante con todos los involucrados en el proyecto y sobre todo por la retroalimentación permanente en cada fase. La metodología XP tiene como objetivo satisfacer al cliente, potenciar el trabajo en equipo y minimizar el riesgo en cuanto a costos, tiempo, calidad y alcance.

Asimismo, en cuanto a las herramientas para la implementación del sistema web decidió utilizar las que se detalla a continuación:

El motor de bases de datos relacionales PostgreSQL debido a que es Open Source y, ofrece un gran rendimiento y entorno de alta disponibilidad, soporta alta concurrencia, permite mantener la integridad referencial y la seguridad de los datos.

Las razones para elegir Spring Boot como tecnología back-end son su sencillez y rapidez en el desarrollo de aplicaciones independientes, así como aspectos críticos para proyectos con plazos ajustados. Su configuración automática y gestión eficiente de dependencias facilitan una implementación flexible y reducen los conflictos comunes en el desarrollo. Además, su escalabilidad y mantenibilidad lo hacen ideal para aplicaciones comerciales como sistemas de gestión académica. La extensa comunidad y el soporte de Spring Boot también son factores clave para proporcionar una base de recursos sólida y ayudar con los problemas durante el proceso de desarrollo.

La integración de Thymeleaf como solución front-end se basa en su compatibilidad óptima con Spring, lo que garantiza una integración perfecta entre el back-end y el front-end. Su enfoque en plantillas nativas facilita la visualización de páginas en el navegador y en su forma estática, lo cual es fundamental para el desarrollo y la depuración. Thymeleaf sobresale en la eficiencia de representación de páginas, un factor clave para aplicaciones con múltiples solicitudes simultáneas. Además, su sintaxis clara y su facilidad de uso reducen la curva de aprendizaje, lo que permite un desarrollo más rápido y eficiente.

El uso de Spring Test para pruebas automatizadas garantiza la calidad y estabilidad del sistema de gestión académica. Está estrechamente integrado con Spring Boot y se puede probar en un entorno similar a la producción, lo que aumenta la precisión y relevancia de los resultados obtenidos. Spring Test proporciona las herramientas para realizar pruebas unitarias y de integración que son esenciales para validar cada componente por separado y todo el sistema. Esta herramienta no solo mejora la confiabilidad del sistema, sino que también facilita el mantenimiento efectivo de las pruebas, lo cual es un beneficio significativo para el desarrollo continuo de aplicaciones.

En relación con el despliegue del sistema web, Amazon Web Services (AWS) servirá alojar la aplicación de gestión académica en función de su flexibilidad y escalabilidad para que los recursos puedan adaptarse a las necesidades cambiantes de sus aplicaciones. AWS ofrece una serie de servicios integrados que facilitan la personalización de su entorno de alojamiento, lo cual es esencial para la gestión de datos académicos y la alta disponibilidad.

A continuación, se presentará la estructura general del sistema web con las herramientas elegidas previamente:

**Figura 14.**

*Estructura de herramientas Seleccionadas.*

A screenshot of a computer

Description automatically generated

## DESARROLLO DE UNA APLICACIÓN WEB PARA GESTION ACADEMICA

## FASE DE LANZAMIENTO INICIAL

Es la fase de planificación y preparación, que sienta las bases del proyecto. Esto incluye reunirse con los clientes para comprender sus necesidades, crear historias de usuarios, planificación de proyectos de alto nivel, configurar el entorno de trabajo, definir prácticas de trabajo (como programación y pruebas en pares (desarrollo guiado)) y una comunicación efectiva. plan.

## HISTORIAS DE USUARIO

Se trata de descripciones breves orientadas al usuario que indican la funcionalidad o características requeridas por el software. Están escritos desde la perspectiva del usuario final y siguen un formato simple que resalta las necesidades y motivos del usuario. Las historias de usuarios son herramientas conversacionales y conversacionales diseñadas para facilitar la discusión y la colaboración entre los equipos de desarrollo y los clientes. Deben ser lo suficientemente claros para ser evaluados, lo suficientemente pequeños como para implementarlos en iteraciones de desarrollo y permitir criterios de aceptación y pruebas.

Las historias de usuario deben contar con las propiedades:

* Independiente
* Negociable
* Valiosa
* Estimable
* Pequeña
* Comprobable

Para el desarrollo de la aplicación web para la gestión académica se establecieron las siguientes historias de usuario:

**Figura 15.**

*Historia de usuario No. 1*



**Figura 16.**

*Historia de usuario No. 2*



**Figura 17.**

*Historia de usuario No. 3*



**Figura 18.**

*Historia de usuario No. 4*



**Figura 19.**

*Historia de usuario No. 5*



**Figura 20.**

*Historia de usuario No. 6*



**Figura 20.**

*Historia de usuario No. 7*



## FORMALIZACION DE REQUERIMIENTOS

## REQUERIMIENTOS FUNCIONALES Y NO FUNCIONALES

Los requisitos representan las necesidades o expectativas de los usuarios para el sistema que se está desarrollando, incluidas la funcionalidad, las limitaciones y las características de calidad.

Los requerimientos se clasifican en:

* **Requerimientos funcionales**: Los requisitos funcionales son especificaciones que describen lo que debe hacer el sistema, es decir, una función, tarea o servicio específico que el sistema debe poder realizar.
* **Requerimientos no funcionales**: Por otro lado, los requisitos no funcionales se centran en cómo el sistema realiza sus funciones, definiendo estándares de desempeño y calidad.

**Tabla 2**

*Requerimientos Funcionales.*

|  |  |
| --- | --- |
| *REQUERIMIENTOS FUNCIONALES* | *REQUERIMIENTOS DE DISEÑO* |
| *Módulo de autenticación* | *Ingreso al sistema* |
| *Módulo de gestión de profesores* | *Administración de profesores* |
| *Módulo de gestión de alumnos* | *Administración de alumnos* |
| *Módulo de gestión de materias* | *Administración de materias* |
| *Módulo de gestión de calificaciones* | *Administración de calificaciones* |
| *Módulo de gestión de horarios* | *Administración de horarios* |
| *Módulo de gestión de matriculación* | *Matriculación de alumnos* |

**Tabla 3**

*Requerimientos No Funcionales.*

|  |  |
| --- | --- |
| *REQUERIMIENTOS NO FUNCIONALES* | *DETALLE* |
| *Rendimiento* | *La aplicación debe poder manejar una gran cantidad de datos sin degradación del rendimiento.* |
| *Disponibilidad y Fiabilidad* | *La aplicación debe estar disponible el 99.9% del tiempo, asegurando un acceso constante para los usuarios y administradores.* |
| *Seguridad* | *La aplicación debe asegurar la protección de datos sensibles* |
| *Usabilidad* | *La aplicación debe ser compatible con diversos navegadores (como Chrome, Firefox, Safari)* |
| *Compatibilidad* | *Administración de calificaciones* |
| *Escalabilidad* | *El sistema debe ser escalable para poder adaptarse a un aumento en el número de usuarios o en la carga de trabajo sin que ello afecte su rendimiento.* |
| *Mantenimiento y Soporte* | *El software debe ser fácil de mantener y actualizar. Debería haber un plan claro para el soporte técnico y la resolución de problemas para los usuarios.* |

**Tabla 4**

*Requerimientos Mínimos.*

|  |  |
| --- | --- |
| *REQUERIMIENTOS NO FUNCIONALES* | *DETALLE* |
| *Hardware* | *No aplica.* |
| *Software* | *Navegador web*  *Acceso Internet* |

## CASOS DE USO

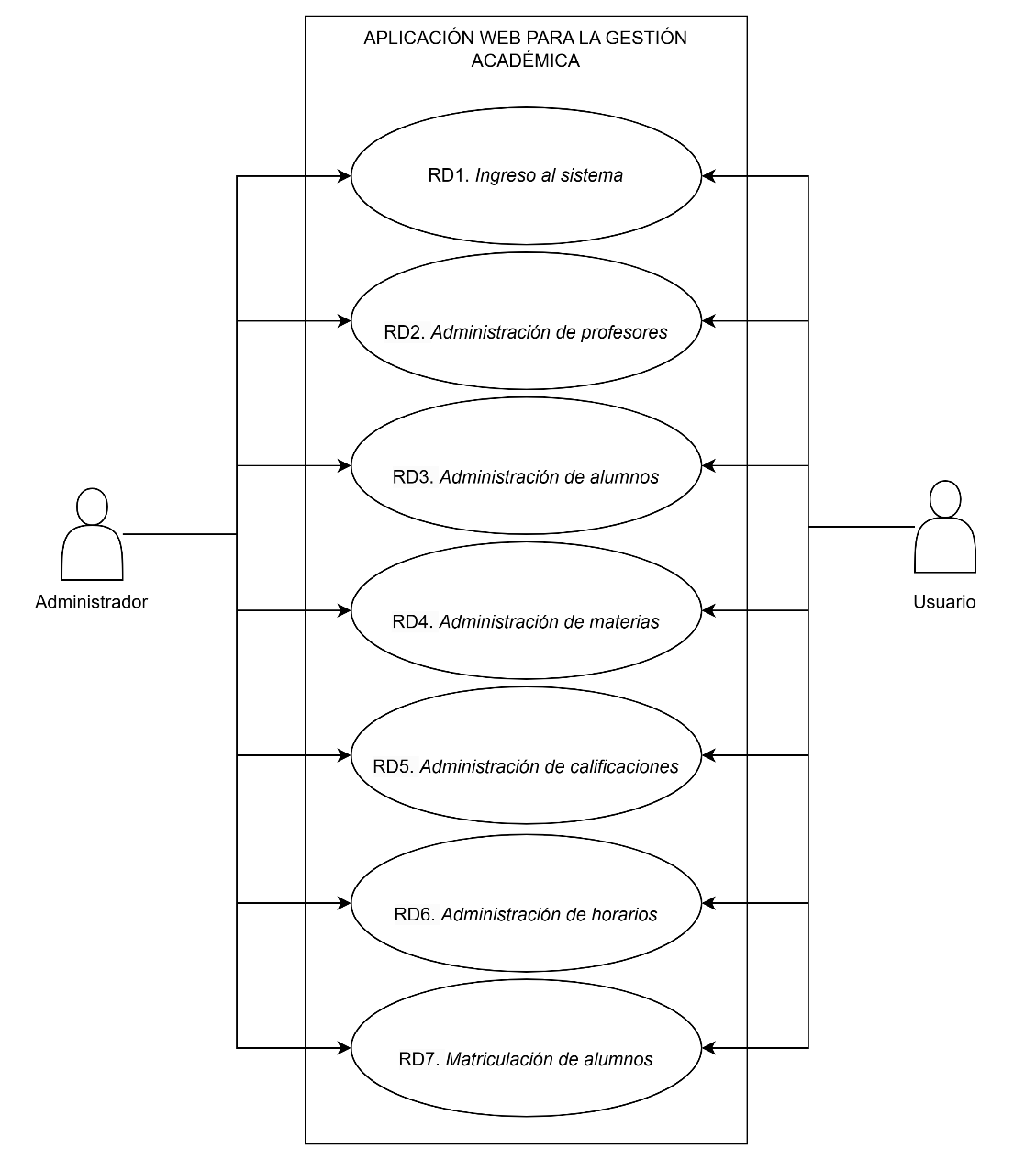
Es una técnica utilizada en el desarrollo de sistemas y software para capturar los requisitos funcionales de un sistema. Un caso de uso describe una secuencia de acciones realizadas por un sistema para proporcionar resultados observables y valiosos a un actor específico.

1. **NIVEL GENERAL**

Caso de uso a nivel general describe todas las funcionalidades del sistema y presenta los actores que intervendrán en él.

**Figura 21.**

*Nivel General.*



1. **SIGUIENTE NIVEL**

Presenta a detalle cada una de las funcionalidades en el nivel general y especificaciones de interacción de los usuarios en función del rol

**Figura 22.**

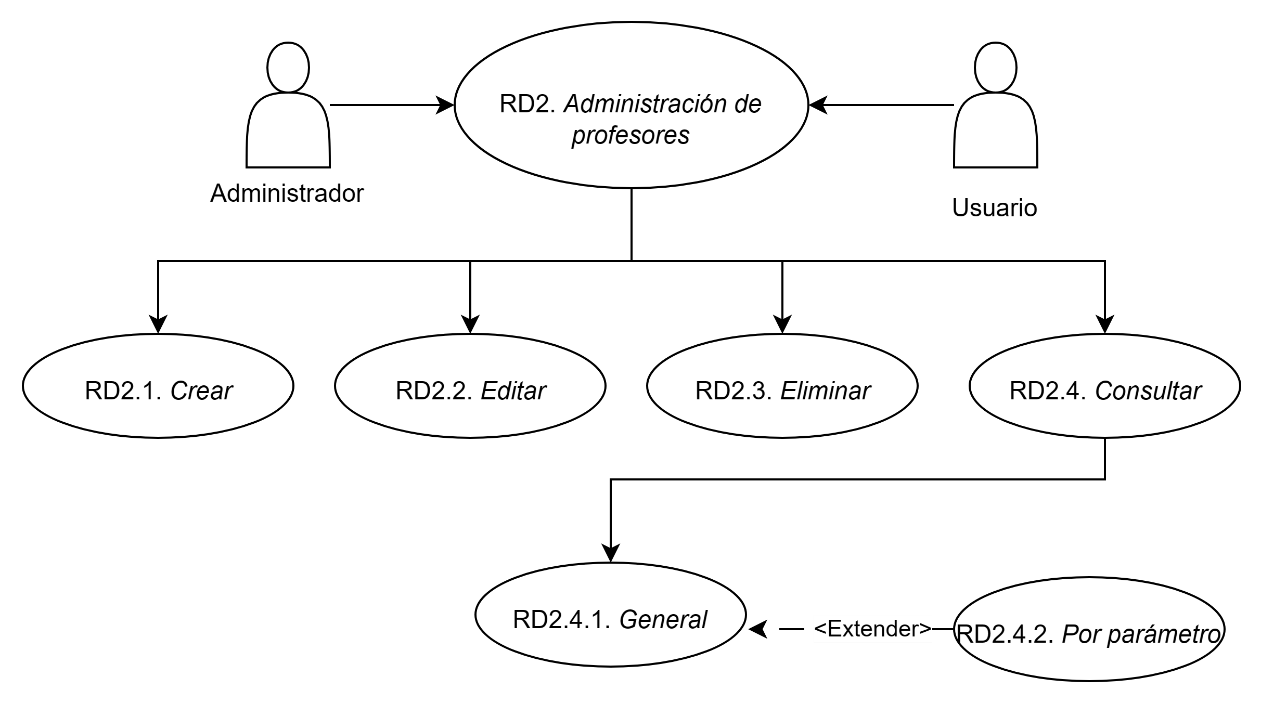
*RD1. Ingreso al sistema*

A white oval with black text

Description automatically generated

**Figura 23.**

*RD2. Administración de profesores*



**Figura 23.**

*RD3. Administración de alumnos*

A group of white ovals with black text

Description automatically generated

**Figura 23.**

*RD4. Administración de materias*

A group of white ovals with black text

Description automatically generated

**Figura 24.**

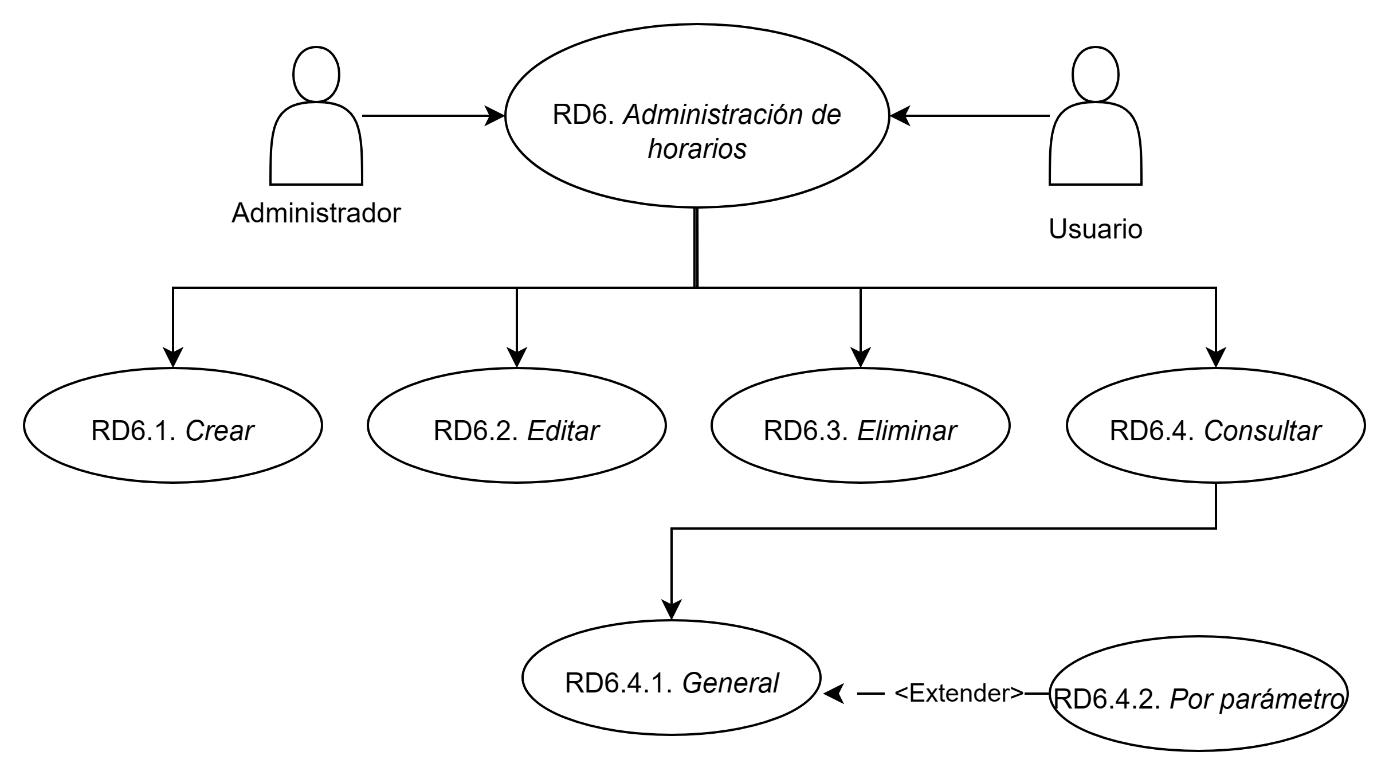
*RD5. Administración de calificaciones*

A group of white ovals with black text

Description automatically generated

**Figura 25.**

*RD6. Administración de horarios*



**Figura 26.**

*RD7. Matriculación de alumnos*

A group of white ovals with black text

Description automatically generated

## DISEÑO

El método XP enfatiza que el desarrollo es simple y limpios, porque los diseños simples son mucho más rápidos de implementar.

Además, sugiere utilizar la metáfora como explicar el propósito del proyecto de una manera sencilla y alentar a refactorizar el código para que sea lo más mantenible simple y fácil de entender. (Collazo Garcia & Labrador, 2017)

* 1. ***ARQUITECTURA***

La arquitectura que se propuso para el desarrollo de la aplicación del sistema web es por capas, esto quiere decir que esta compuesta por la capa de presentación o interfaz gráfica, capa de negocio o lógica de negocio y la capa de datos o administración de datos.

Por la clasificación dicha, se manejará de una manera clasifica las diferentes capas y estas se podrán comunicar por un cliente HTTP que se encarga de transformar la información de peticiones y respuestas del servidor

En la figura 47 se presenta la arquitectura de la aplicación junto con sus herramientas utilizadas para su desarrollo.

**Figura 27.**

*RD7. Arquitectura del sistema.*

A screenshot of a computer

Description automatically generated

* 1. ***BASE DE DATOS***

En base a los requerimientos descritos en las historias de usuario y analizando los requerimientos funcionales, el modelo de la base de datos es el siguiente:

A diagram of a computer program

Description automatically generated with medium confidence

* + 1. **ITERACIÓN 1**

En base a lo planificado, para la primera iteración se implementaron las historias de usuario 1 y 2 es decir:

* Módulo de autenticación
* Módulo de gestión de profesores

1. **Historia de usuario 1**
   1. **BackEnd**
   2. **FrontEnd**
2. **Historia de usuario 2**
   1. **BackEnd**
   2. **FrontEnd**
3. **Historia de usuario 3**
   1. **BackEnd**
   2. **FrontEnd**
4. **Historia de usuario 4**
   1. **BackEnd**
   2. **FrontEnd**
5. **Historia de usuario 5**
   1. **BackEnd**
   2. **FrontEnd**
6. **Historia de usuario 6**
   1. **BackEnd**
   2. **FrontEnd**
7. **Historia de usuario 7**
   1. **BackEnd**
   2. **FrontEnd**

## CONCLUSIONES Y RECOMENDACIONES

## RECOMENDACIONES

## REFERENCIAS

Aliaga, A., & Miani, M. (2008). *PostgreSQL* (Vol. 1). Amaya, V. (2021). *Requerimientos & historias de usuario*.

Amazon Web Services Inc. (2018). *Información general sobre Amazon Web Services*.

Amazon Web Services Inc. (2021a). *Amazon EC2*. https://aws.amazon.com/es/ec2/?ec2- whats-new.sort-by=item.additionalFields.postDateTime&ec2-whats-new.sort- order=desc

Amazon Web Services Inc. (2021b). *Amazon Relational Database Service (RDS)*. https://aws.amazon.com/es/rds/

Ambler, S. (2002). Agile Modeling: Effective Practices for eXtreme Programming and the Unified Process. In *John WileySons*. <http://www.amazon.fr/exec/obidos/ASIN/0471202827/citeulike04-> [21%5Cnht](http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle%3AAgile%2BM)tp://s[cholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Agile+M](http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle%3AAgile%2BM) odeling+:#0

Bahit, E. (2012). *Scrum & Extreme Programming Para Programadores*. 162. <http://www.cursosdeprogramacionadistancia.com/static/pdf/material-sin-> personalizar-agile.pdf

Casas, M. (2014). *Servicios Web*. https://[www.ibm.com/support/knowledgecenter/es/SSAW57\_9.0.5/com.ibm.websph](http://www.ibm.com/support/knowledgecenter/es/SSAW57_9.0.5/com.ibm.websph) ere.nd.multiplatform.doc/ae/cwbs\_wbs2.html

Collazo Garcia, A., & Labrador, D. (2017). *La Programación Extrema Autores : Marycarmen Díaz Labrador*. *November 2013*. https://doi.org/10.13140/RG.2.2.29359.43687

De las Pueblas Encinas, G. M. (2019). *Definición de Requisitos Funcionales bajo Especificación IEEE para un*. 1–133.

Denzer, P. (2002). 51. PostgreSQL. *Profesores.Elo.Utfsm.Cl*, 21. <http://profesores.elo.utfsm.cl/~agv/elo330/2s02/projects/denzer/informe.pdf>

Dept. Ciencia de la Computación e IA, U. de A. (2014). Introducción a los Servicios Web. Invocación de servicios web SOAP. *Universidad de Alicante*, 1–46. <http://www.jtech.ua.es/j2ee/publico/servc-web-2012-13/sesion01-apuntes.html>

Duarte Vega, G. E. (2016). *Arquitectura Propuesta para un Servicio Web Completo: Metodología de Desarrollo e Implementación*. 1–154.

Encalada, C. (2018). *ANALISIS, DESARROLLO, IMPLEMENTACION E IMPLANTACION DE UN SISTEMA PARA LA AUTOMATIZACION DE PROCESOS EN UN CENTRO MEDICO. CASO DE ESTUDIO: E.M. CENTRO DE ESPECIALIDADES MEDICAS, UTILIZANDO LA METODOLOGIA EXTREME PROGRAMMING (XP).*

Express.js. (2021). *Introducción a Express.js*. https://expressjs.com/ Facebook Inc. (2017). *Jest*. https://doc.ebichu.cc/jest/es-ES/

Hernández, R., Fernández, C., & Baptista, P. (2014). *Metodología de la investigación*. https://[www.uca.ac.cr/wp-content/uploads/2017/10/Investigacion.pdf](http://www.uca.ac.cr/wp-content/uploads/2017/10/Investigacion.pdf)

HYPERS. (2016). *Rsuite*. https://github.com/rsuite/rsuite

IBM. (2019). *¿Qué es SOAP?*

https://[www.ibm.com/support/knowledgecenter/es/SSMKHH\_10.0.0/com.ibm.etool](http://www.ibm.com/support/knowledgecenter/es/SSMKHH_10.0.0/com.ibm.etool) s.mft.doc/ac55770\_.htm

Joskowicz, J. (2008). Reglas y prácticas en eXtreme Programming. *Universidad de Vigo.*

*España*, 1–22. <http://iie.fing.edu.uy/~josej/docs/XP>- Jose Joskowicz.pdf Jovaldiv, V. C. (2016). *Modelo De Procesos Para El Desarrollo Del*. 187–208.

Laínez Fuentes, J. R. (2015). *Desarrollo de Software Ágil: Extremme Programming y Scrum.* (IT Campus Academy (ed.); Segunda ed).

Lamarca M. (2013). *Tesis Doctoral Hipertexto: el nuevo concepto de documento en la cultu- ra de la imagen. Universidad Complutense de Madrid.* <http://www.hipertexto.info/documentos/serv_web.htm>

Lezama López, J. A., & Machuca Abanto, E. (2017). Universidad Privada Antonio

Guillermo Urrelo. *Universidad Privada Antonio Guillermo Urrelo*. <http://repositorio.upagu.edu.pe/handle/UPAGU/269>

López Menéndez de Jiménez, R. (2015). *Metodologías ágiles de desarrollo de Software aplicadas a la gestión de proyectos empresariales*. 6–11.

Molina Rios, J. R., Zea Ordonez, M. P., Contento Segarra, M. J., & Garcia Zerda, F. G. (2017). State of Art: Development Methodologies in Web Applications. *3C Tecnologia*, *6*(3), 54–71. https://[www.3ciencias.com/articulos/articulo/estado-del-](http://www.3ciencias.com/articulos/articulo/estado-del-) arte-metodologias-desarrollo-aplicaciones-web/

Montero Molina, B., Cevallos Vite, H., & Dávila Cuesta, J. (2018). Metodologías ágiles frente a las tradicionales en el proceso de desarrollo de software. *Espirales Revista Multidisciplinaria de Investigación ISSN: 2550-6862*, *2*(17), 114–121. https://[www.researchgate.net/publication/327537074\_Metodologias\_agiles\_frente\_a](http://www.researchgate.net/publication/327537074_Metodologias_agiles_frente_a)

\_las\_tradicionales\_en\_el\_proceso\_de\_desarrollo\_de\_software

Moreno Saiz, S. (2015). *Estudio de Arquitecturas Software para Servicios de Internet de las Cosas*. <http://oa.upm.es/37339/>

Muñoz, A. (2013). *Introducción a NodeJS a través de Koans*. 242. <http://nodejskoans.com/>

Node.js. (2021). *Introducción a Node.js*. https://nodejs.org/es/

Pinto, S., Avila, M., & Mejia, M. (2016). *Diseño y desarrollo de una prueba de concepto de un lector electrónico (e-reader) de textos en inglés*. *1*, 1917–1921.

Pulido, E., Escobar, Ó., & Nuñez, J. Á. (2019). *Base de datos* (Primera ed). Patria Educación.

Red Hat Inc. (2021). *¿Qué es una arquitectura de aplicaciones?*

Rick, L. (2016). *Express. js Guide Book on Web framework for Node. js*.

Rodríguez, Á. I., Padilla, J. I., Parra, H. A., & Rodríguez, P. (2019). *Arquitectura basada en micro-servicios para aplicaciones web Microservices based architecture for web applications*.

Salud, D. E. (2006). *Guía de diseño y mejora continua de procesos asistenciales*.

Sataloff, R. T., Johns, M. M., & Kost, K. M. (2018). *COMPARACIÓN DE TENDENCIAS TECNOLÓGICAS EN APLICACIONES WEB*.

Sayago, J., Flores, E., & Recalde, A. (2019). *Análisis Comparativo entre los Estándares Orientados a Servicios Web SOAP , REST y GRAPHQL Comparative Analysis between Standards Oriented to Web Services SOAP , REST and GRAPHQL*. *9*, 10– 22.

Semantic-Org. (2018). *Semantic UI*. https://github.com/Semantic-Org/Semantic-UI Sequelize. (2021). *Introducción a Sequelize*. https://sequelize.org/

Severance, C. (2020). Uso de Servicios Web. In *Python para todos: explorando la información con Python 3*.

Veps, J. (2017). *SurviveJS - React De aprendiz a maestro Juho Vepsäläinen y Raúl Expósito*.

Vercel Inc. (2021). *Vercel*. https://vercel.com/docs

Yanez, D. (2019). Método descriptivo: características, etapas y ejemplos. *1*, 1. https://[www.lifeder.com/metodo-descriptivo/](http://www.lifeder.com/metodo-descriptivo/)