INSTITUTO SUPERIOR TECNOLÓGICO TENA
Tecnología, Innovación y Desarrollo

DESARROLLO DE SOFTWARE

Instrumento para facilitar el proceso de enseñanza-aprendizaje de la asignatura

GUÍA GENERAL DE ESTUDIO DE LA ASIGNATURA 20250022

PROGRAMACIÓN WEB

Período académico
Tercero

Octubre - 2025

ING. AGUSTIN GONZALO GUANIPATÍN RAMÍREZ

GUÍA GENERAL DE ESTUDIO DE LA ASIGNATURA – PROGRAMACIÓN WEB

INSTITUTO SUPERIOR TECNOLÓGICO TENA

Carrera de Tecnología en Desarrollo de Software

ISTT DSW Primer Edición – Tena, octubre 2025
SIN ISBN
Instituto Superior Tecnológico Tena
Km. 1 1/2 Vía Tena - Archidona
Tena, Ecuador

AUTOR - REDACCIÓN Y FORMULACIÓN DE CONTENIDOS

Ing. Agustín Gonzalo Guanipatín Ramírez
Profesor del Instituto Superior Tecnológico Tena

REVISIÓN DE PARES
Mg. Alvaro Santiago Toalombo Díaz
Mg. Henry Fabian Chango Chango
Mg. Martha Janina Duarte Mora
Mg. Danilo Alexander Zamora Núñez
Lcda. María Angélica Campoverde Encalada

Comisión de revisión técnica de guías de estudio del Instituto Superior Tecnológico Tena

APROBACIÓN
Mg. Danilo Alexander Zamora Núñez
Coordinador de Investigación, Desarrollo Tecnológico e Innovación

Impreso y hecho en Ecuador.

TABLA DE CONTENIDO

GUIA GENERAL DE ESTUDIO DE LA ASIGNATURA

DATOS GENERALES DE LA ASIGNATURA					
Carrera	Desarrollo de Software		Nombre asignatura	Programación Web	
Modalidad	Presencial		Campo de Formación	NA	
Jornada	Matutina		Unidad de Organización Curricular	Profesional	
Período académico	Tercero		Código de la asignatura	DSW-303	
Distribución de horas en las actividades de aprendizaje			N° Total de horas de la asignatura	192	
N° de horas Docencia	64	N° de horas Aprendizaje Práctico Experimental		N° de horas Autónomo	40

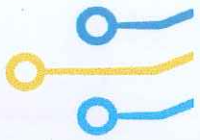
PRERREQUISITOS Y CORREQUISITOS			
Prerrequisitos de la asignatura		Correquisitos de la asignatura	
Asignatura	Código	Asignatura	Código

DESCRIPCIÓN DE LA ASIGNATURA
Las páginas web, es una aplicación en el cual el usuario realiza peticiones a través del navegador a una aplicación remota accediendo a través del internet (o de una intranet), mismo que recibe una respuesta y se muestra en el mismo navegador.

OBJETIVO GENERAL
Desarrollar páginas web, utilizando las nuevas tecnologías de programación web de código abierto, desde su etapa de análisis, diseño, desarrollo, pruebas, implementación, mantenimiento, teniendo en cuenta las necesidades del usuario fin.

CONTRIBUCIÓN DE LOS RESULTADOS DE APRENDIZAJE DE LA ASIGNATURA AL PERFIL DE EGRESO DE LA CARRERA

Resultados de aprendizaje de la asignatura	Resultados de aprendizaje del perfil de egreso de la carrera	Contribución (alta – media – baja)
Aplica conceptos, técnicas, herramientas de programación, que contribuyan con la implementación de soluciones de software. Brinda asistencia técnica en el desarrollo de páginas de software, desde el análisis del pro-	Utiliza herramientas y tecnologías de programación para llevar a cabo tareas específicas en el campo de desarrollo de software. Aplica conceptos, técnicas, herramientas de	Alta



blema y la planificación del proyecto, hasta la implementación, el mantenimiento, la prueba y la documentación.	programación, que contribuyan con la implementación de soluciones de software.	Media
Utiliza herramientas y tecnologías de programación para llevar a cabo tareas específicas en el campo de desarrollo de software.	Aplica técnicas de investigación en la búsqueda de nuevas formas de aplicación del desarrollo de software en los sectores industriales.	Alta

CONTENIDOS DE LA ASIGNATURA (descripción mínima de contenidos de la asignatura)

Unidad 1: Introducción a la web
1.1 Generalidades de la internet y los sistemas web.
1.2 HTML.
1.3 Hojas de estilos en cascada (CSS).
1.4Introduccion al lenguaje de programación JavaScript.

Unidad 2: Gestores de contenidos
2.1 Introducción a los CMS.
2.2 Creando Páginas Web Informativas.
2.3 Configuración de un web hosting y despliegue de una página Web en internet.

Unidad 3: FrontEnd
3.1 JavaScript y TypeScript.
3.2 Frameworks FrontEnd.
3.3 Modo producción y ejecutables del sistema FrontEnd.

ESTRATEGIAS METODOLÓGICAS Y RECURSOS DIDÁCTICOS

ESTRATEGIAS METODOLÓGICAS	HABILIDADES BLANDAS	FINALIDAD
Activas para la enseñanza y aprendizaje	Valores vinculados a la autonomía del sujeto: confianza, crítica y autocrítica, honestidad, integridad	• Generar confianza/ Promover el pensamiento crítico. • Permite a los estudiantes cumplir un rol activo dentro de su formación. • Construye una sociedad participante.

Aprendizaje y trabajo cooperativo	Valores elementales de convivencia y civilidad: crítica y autocrítica, tolerancia, empatía, respeto, justicia, lealtad, paciencia	• Promover un ambiente de colaboración/ trabajo en equipo/ Saber escuchar/Promover el pensamiento crítico/ fomentar el liderazgo/ adaptabilidad. • Mantener una comunicación abierta con el equipo/ tolerancia a los errores, aceptar y aprender de las críticas. • Fomentar el sentido de pertenencia
Aprendizaje individual	Valores vinculados a la autonomía del sujeto: responsabilidad, honestidad, integridad, efectividad, autonomía	• Facilitar la asimilación del contenido por parte del estudiante/ Plantear preguntas para promover la comunicación efectiva /Promover el pensamiento crítico • Lectura comprensiva para fijar contenidos/ Promover el pensamiento crítico

RECURSOS DIDÁCTICOS	
MATERIALES CONVENCIONALES	*Material impreso: libros, folletos, fotocopias, periódicos, etc.*
	Tableros didácticos: pizarra
MATERIALES AUDIOVISUALES	*Imágenes fijas proyectables (fotos): diapositivas y fotografías.*
	Materiales audiovisuales (vídeo): películas y vídeos
NUEVAS TECNOLOGÍAS	*Programas informáticos: procesador de palabras, hojas de cálculo, presentaciones*
	Servicios telemáticos: páginas web, plataforma EVA, correo electrónico, google drive, Visual Studio code

BIBLIOGRAFÍA

Bibliografía Básica de la Asignatura:	Físico	Digital
• Flòrez Fernandez, H., & Herandez Rodriguez, J. (2021). Aplicaciones web con php. (Primera Edición). Colombia: Ediciones de la U. ISBN: 978-958-792-234-9, Número de inventario en biblioteca: ISTT-DS-0143	X	
• Ortega, J. (2020). Desarrollo seguro en ingeníeria del software. Aplicaciones seguras con android, nodejs , phyton y C++. Colombia: colombia Alfaomega Grupo Editor 2020. ISBN: 978-958-778-638-5, Número de inventario en biblioteca: ISTT-DS-0071	X	
• Ramos Varon, A. A., Barbero Muñoz, C. A., Martinez Sanchez, R., Garcia	X	

Moreno, A., & Gonzales Nava, J. M. (2018). Hacking y seguridad de páginas web Seguridad Informática y Hacking. Colombia: Ra-ma. ISBN: 978-958-762-381-9, Número de inventario en biblioteca: ISTT-DS-0153	X	
• Nixon, R. (2019). Aprender, PHP, MySQL y JavaScript Con JQUERY, CSS Y HTML5. (Quinta Edición). España: Marcombo S.A. ISBN: 978-84-267-2715-2, Número de inventario en biblioteca: ISTT-DS-0175		
Bibliografía de consulta de la Asignatura:	Físico	Digital
• Recio García, J. A. (2018). html5, css3 y jquery. Colombia: RA-MA. ISBN: 978-958-762-783-1, Número de inventario en biblioteca: ISTT-DS-0096	X	
• Torres Remón, M. A. (2014). Diseño web con HTML5 y CSS3. Perú: Talleres gráficos de la empresa editora macro EIRL. ISBN: 978-612-304-207-3, Número de inventario en biblioteca: ISTT-DS-0142	X	

DESCRIPTIVA DE LA ASIGNATURA DE PROGRAMACIÓN WEB

La asignatura de Programación Web se centra en el desarrollo de aplicaciones web utilizando tecnologías modernas y de código abierto. Se abordan temas como HTML, CSS, JavaScript, gestores de contenido (CMS), frameworks FrontEnd, y desarrollo de APIs RESTful, así como la integración de tecnologías avanzadas como WebSockets y GraphQL. El objetivo es capacitar a los estudiantes en el análisis, diseño y desarrollo de páginas web efectivas y escalables

Competencias Específicas

UNIDAD 1: Introducción a la web

1.1 Generalidades de la internet y los sistemas web: Explica el origen, funcionamiento y componentes básicos de Internet, así como la estructura de los sistemas web y cómo interactúan el cliente, el servidor y la red.
1.2 HTML: Lenguaje base para crear páginas web. Define la estructura del contenido mediante etiquetas que organizan texto, imágenes, enlaces y otros elementos.
1.3 Hojas de estilos en cascada (CSS): Lenguaje que controla la apariencia visual de las páginas web (colores, tipografías, márgenes, diseño adaptable). Separa el contenido de la presentación.
1.4 Introducción al lenguaje de programación JavaScript: Lenguaje que agrega interactividad y dinamismo a los sitios web, permitiendo validar formularios, crear animaciones o manipular el contenido en tiempo real.

UNIDAD 2: Gestores de contenidos

2.1 Introducción a los CMS: Presenta los sistemas de gestión de contenidos (como WordPress o Joomla) que facilitan la creación y administración de sitios web sin necesidad de programar desde cero.
2.2 Creando Páginas Web Informativas: Guía la elaboración de sitios básicos con contenido estructurado, diseño coherente y navegación funcional, usando un CMS u otros métodos sencillos.
2.3 Configuración de un web hosting y despliegue de una página Web en internet: Explica cómo contratar un servicio de alojamiento, configurar dominios y subir los archivos de un sitio web para hacerlo accesible al público en línea.

UNIDAD 3: FrontEnd

. 3.1 JavaScript y TypeScript: Estudia JavaScript en mayor profundidad y presenta TypeScript como su versión mejorada con tipado estático para desarrollar aplicaciones más seguras y mantenibles.
3.2 Frameworks FrontEnd: Introduce herramientas modernas como React, Angular o Vue.js, que facilitan el desarrollo rápido de interfaces de usuario interactivas y escalables.

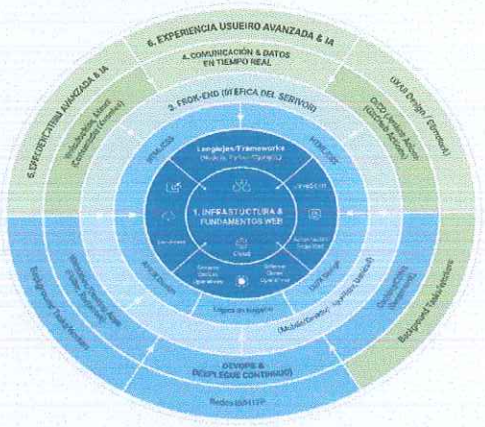
3.3 Modo producción y ejecutables del sistema FrontEnd: Explica cómo optimizar, compilar y desplegar una aplicación web lista para producción, reduciendo tamaño, mejorando rendimiento y asegurando compatibilidad.

1. UNIDAD 1: Introducción a la web

1.1 Generalidades de la internet y los sistemas web.
1.2 HTML.
1.3 Hojas de estilos en cascada (CSS).
1.4 Introducción al lenguaje de programación JavaScript.

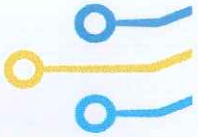
Aplica conceptos, técnicas, herramientas de programación, que contribuyan con la implementación de soluciones de software.

DIAGRAMA DE APRENDIZAJE



SÍNTESIS

La Programación Web se enfoca en el desarrollo de aplicaciones web utilizando tecnologías modernas y de código abierto, abarcando temas como HTML, CSS, JavaScript, frameworks FrontEnd, y APIs RESTful. Se busca capacitar a los estudiantes en el análisis, diseño y desarrollo de páginas web efectivas y escalables, integrando tecnologías avanzadas como WebSockets y GraphQL. Además, se enfatiza la importancia de la programación back-end, arquitecturas de aplicaciones y el uso de bases de datos para crear soluciones de software robustas.



1.1. Generalidades de la internet y los sistemas web

Sirve como la base fundamental para el desarrollo, encargándose de establecer el contexto operacional y estructural de la Web. Esto incluye comprender la arquitectura Cliente-Servidor (cómo interactúan los navegadores y los servidores), el funcionamiento de los protocolos esenciales como HTTP/HTTPS y TCP/IP (las reglas de la comunicación de datos), y el rol de las tecnologías básicas de presentación como HTML, CSS y JavaScript, sentando así las bases teóricas y prácticas antes de abordar temas avanzados de *backend* o *frontend.*

1.1.1. ¿Qué hace un programador Web?

Es el profesional encargado de diseñar, construir, codificar y mantener sitios web y aplicaciones web, asegurando tanto su funcionalidad técnica como su experiencia de usuario, por ejemplo:

- Desarrollador Frontend (Lado del Cliente)
- Desarrollador Backend (Lado del Servidor)
- Desarrollador Full Stack

1.1.2. Servidores web y protocolos de comunicación

Según diversos expertos y fuentes del ámbito tecnológico, cuando se habla de servidores web y comunicación en el back-end web, lo primero que suele venir a la mente son conceptos comunes como Apache, Nginx, HTTP o HTTPS. Sin embargo, el ecosistema real es mucho más amplio y diverso, con múltiples tecnologías que, aunque menos conocidas, son fundamentales para el funcionamiento eficiente, seguro y moderno de las aplicaciones web.

1.1.3. Generalidades de la Internet y los sistemas web

Explica el origen, funcionamiento y componentes básicos de Internet, así como la estructura de los sistemas web y cómo interactúan el cliente, el servidor y la red. Dentro de los sistemas web, el back-end se encarga de procesar la lógica, gestionar bases de datos y atender las solicitudes del cliente a través de diferentes servidores y protocolos de comunicación.

1.2. Servidores y protocolos del back-end

1.2.1. Servidores web.

Los servidores web son programas que reciben solicitudes de los clientes (generalmente navegadores) y responden con datos, páginas web o servicios. Algunos ejemplos destacados son:

- Apache HTTP Server: Uno de los servidores más antiguos y populares, conocido por su modularidad, flexibilidad y gran comunidad. Soporta múltiples lenguajes back-end como PHP o Python y permite configuraciones muy personalizables.
- Nginx: Diseñado para entornos de alto tráfico, es eficiente y consume pocos recursos. Además, funciona como proxy inverso y balanceador de carga.
- Microsoft IIS (Internet Information Services): Servidor web de Microsoft, pensado para entornos Windows y aplicaciones ASP.NET. Es común en corporaciones que usan tecnología Microsoft.
- Servidores embebidos: Frameworks como Flask (Python), Express (Node.js) o Spring Boot (Java) incluyen mini-servidores que permiten ejecutar aplicaciones sin instalar software adicional.
- Lighttpd y Caddy: Servidores ligeros, fáciles de configurar. Caddy, por ejemplo, habilita HTTPS automáticamente con Let's Encrypt, facilitando seguridad rápida y sencilla.

1.2.2 Protocolos de comunicación

Los protocolos definen cómo se transmiten los datos entre clientes y servidores. Entre los más usados:

- HTTP y HTTPS: Base de la comunicación web. HTTP no tiene cifrado; HTTPS usa TLS/SSL para proteger los datos.
- WebSocket: Permite comunicación bidireccional en tiempo real, útil para chats, juegos en línea o aplicaciones colaborativas.
- gRPC: Protocolo de Google que utiliza HTTP/2 y Protobuf, ideal para microservicios por su rapidez y eficiencia.
- GraphQL: Alternativa moderna a REST, permite a los clientes solicitar exactamente los datos que necesitan.
- SOAP: Basado en XML, usado en sectores como banca o gobierno donde la integridad de los datos es crítica.
- MQTT: Ligero y eficiente, diseñado para IoT, permite transmitir pequeños mensajes entre sensores y servidores.
- AMQP y otros protocolos de mensajería: Facilitan la comunicación asincrónica entre servicios mediante colas de mensajes (RabbitMQ, Kafka), favoreciendo escalabilidad y desacoplamiento.

1.3. Bases de datos y modelos de datos

- Definición de base de datos: Colección organizada de datos que permite acceder, manipular y recuperar información de manera eficiente.
- Propósito: Almacenar y organizar información para su uso posterior.
- Ejemplos: Bases de datos relacionales (MySQL, PostgreSQL), NoSQL (MongoDB, Cassandra) y en la nube.
- Estructura: Puede ser relacional (tablas), jerárquica (árbol), de red (nodos conectados), documental o de grafos.

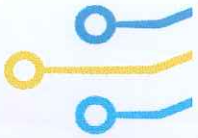
1.4. Modelos de datos

- Definición: Representación abstracta que describe cómo se organizan y relacionan los datos dentro de una base de datos.
- Función: Define la estructura lógica, relaciones entre datos y cómo se realizan operaciones sobre ellos.
- Ejemplos: Modelo relacional, jerárquico, de red, orientado a objetos, conceptual.
- Importancia: Ayuda a desarrolladores y diseñadores a entender la base de datos, comunicar necesidades y construirla correctamente.

1.5. Lenguajes de programación back-end

Los lenguajes back-end permiten crear la lógica de servidor, procesar datos, interactuar con bases de datos, autenticar usuarios y responder al front-end. Algunos populares:

- Python: Lenguaje interpretado y multiparadigma, fácil de leer y con gran cantidad de librerías.
- Java: Orientado a objetos, portable gracias a la JVM, usado en aplicaciones empresariales y móviles.
- PHP: Scripting del lado del servidor, eficiente para sitios web dinámicos y con integración a bases de datos.
- JavaScript (Node.js): Permite usar JavaScript en servidor, ideal para aplicaciones en tiempo real y escalables.
- Ruby: Dinámico y orientado a objetos, conocido por Ruby on Rails, enfocado en productividad y simplicidad.
- C#: Lenguaje de Microsoft, usado en aplicaciones web, móviles, de escritorio y videojuegos.
- Go: Compilado, concurrente y eficiente, ideal para servicios web y sistemas distribuidos de alto rendimiento.
- C++: Compilado, eficiente y de propósito general, usado en software que requiere control de recursos.
- Kotlin: Moderno, seguro y compatible con Java, popular en desarrollo Android y backend.
- Rust: Compilado, seguro y de alto rendimiento, enfocado en concurrencia y fiabilidad.



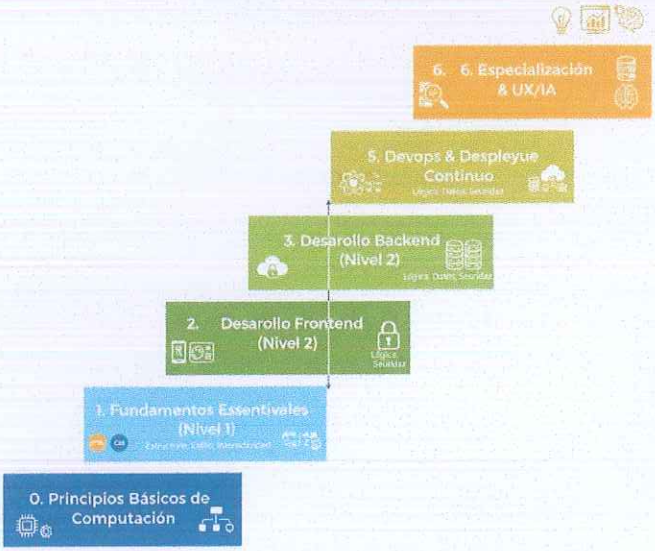
2. UNIDAD 2: Gestores de contenidos

2.1 Introducción a los CMS
2.2 Creando Páginas Web Informativas
2.3 Configuración de un web hosting y despliegue de una página Web en internet

Brinda asistencia técnica en el desarrollo de aplicaciones de software, desde el análisis del problema y la planificación del proyecto, hasta la implementación, el mantenimiento, la prueba y la documentación.

DIAGRAMA DE APRENDIZAJE

PLAN DE APRENDIZAJE



SINTESIS

El back-end es fundamental para el funcionamiento de aplicaciones web modernas. A medida que las aplicaciones se vuelven más complejas y requieren una mayor escalabilidad, es crucial que los desarrolladores de back-end dominen una variedad de tecnologías y enfoques. Este documento se centra en los aspectos avanzados del desarrollo de back-end, incluyendo arquitecturas, lenguajes de programación, bases de datos, y la integración de APIs.

2.1. Introducción a los lenguajes de programación para el back-end

El desarrollo back-end se ocupa de la parte de una aplicación que no es visible para el usuario, pero que es fundamental para su funcionamiento. Incluye la lógica del servidor, el procesamiento de datos, la autenticación de usuarios, el manejo de bases de datos y la seguridad del sistema. A diferencia del front-end, que se centra en la interfaz y experiencia del usuario, el back-end asegura que todo funcione correctamente detrás de escena. Para desarrollar esta parte de la aplicación, se utilizan diferentes lenguajes de programación, cada uno con características específicas que se adaptan a distintos tipos de proyectos.

2.1.1. Node.js

Node.js permite ejecutar JavaScript en el servidor, lo que facilita que los desarrolladores trabajen con un solo lenguaje en toda la aplicación. Destaca por su rapidez, manejo de múltiples conexiones simultáneas y amplio ecosistema de paquetes, siendo ideal para aplicaciones en tiempo real como chats o juegos en línea.

2.1.2. Python

Python es un lenguaje de sintaxis clara y sencilla, muy recomendable para principiantes. Cuenta con frameworks como Flask y Django que permiten construir aplicaciones web robustas de forma rápida y eficiente. Además, se utiliza en inteligencia artificial, análisis de datos y automatización, lo que lo hace versátil y con gran proyección.

2.1.3. Ruby

Ruby se destaca por su elegancia y facilidad de lectura. En back-end, se utiliza principalmente con el framework Ruby on Rails, que ofrece una estructura completa para desarrollar aplicaciones web de manera ágil, promoviendo buenas prácticas y acelerando el lanzamiento de productos, especialmente útil para startups.

2.1.4. Consideraciones para elegir un lenguaje

Al elegir un lenguaje back-end, se deben considerar la curva de aprendizaje, la comunidad de soporte, el tipo de aplicación y la escalabilidad del sistema. Cada lenguaje tiene ventajas según el contexto del proyecto y la experiencia del equipo.

2.2. Arquitectura de aplicaciones web

La arquitectura de una aplicación web define cómo se organizan sus componentes internos para facilitar el desarrollo, mantenimiento y escalabilidad. Entre los modelos más comunes se encuentran MVC (Modelo-Vista-Controlador) y MVVM (Modelo-Vista-VistaModelo).

2.2.1. Patrón MVC

Divide la lógica de la aplicación en tres componentes:

- Modelo: Gestiona los datos y reglas de negocio.
- Vista: Representa la interfaz del usuario.
- Controlador: Intermediario entre Modelo y Vista, gestionando solicitudes y actualizando datos y la interfaz.

Esta separación facilita el trabajo en equipo y el mantenimiento del código.

2.2.2. Patrón MVVM

Se utiliza en aplicaciones con fuerte interacción de interfaz, como las desarrolladas con frameworks modernos. Mantiene el Modelo y la Vista, pero agrega la VistaModelo, que contiene la lógica de presentación y vincula automáticamente los datos entre Modelo y Vista, reduciendo código repetitivo y mejorando la reactividad de la aplicación.

Ambos patrones promueven organización, reutilización de componentes y facilidad de prueba. La elección depende del proyecto, herramientas y preferencias del equipo de desarrollo.

2.3. Bases de datos en el desarrollo web

Las bases de datos permiten almacenar, gestionar y recuperar la información usada por las aplicaciones web. Pueden ser:

2.3.1. Bases de datos relacionales

Cómo MySQL, PostgreSQL y SQL Server, organizan datos en tablas con filas y columnas y utilizan SQL para consultas. Mantienen integridad de datos y relaciones entre diferentes tipos de información, siendo ideales para sistemas bancarios, comercio electrónico y aplicaciones empresariales.

2.3.2. Bases de datos no relacionales (NoSQL)

Como MongoDB, CouchDB o Firebase, son flexibles y escalables, diseñadas para grandes volúmenes de datos que no encajan en tablas. Adecuadas para aplicaciones modernas que requieren alto rendimiento, manejo de datos en tiempo real o estructuras complejas como documentos JSON o grafos.

2.3.3. Elección de la base de datos

Depende del tipo de datos, rendimiento requerido, escalabilidad y experiencia del equipo. Muchos proyectos modernos combinan bases de datos relacionales y no relacionales para aprovechar las ventajas de cada una.

3. UNIDAD 3: FrontEnd

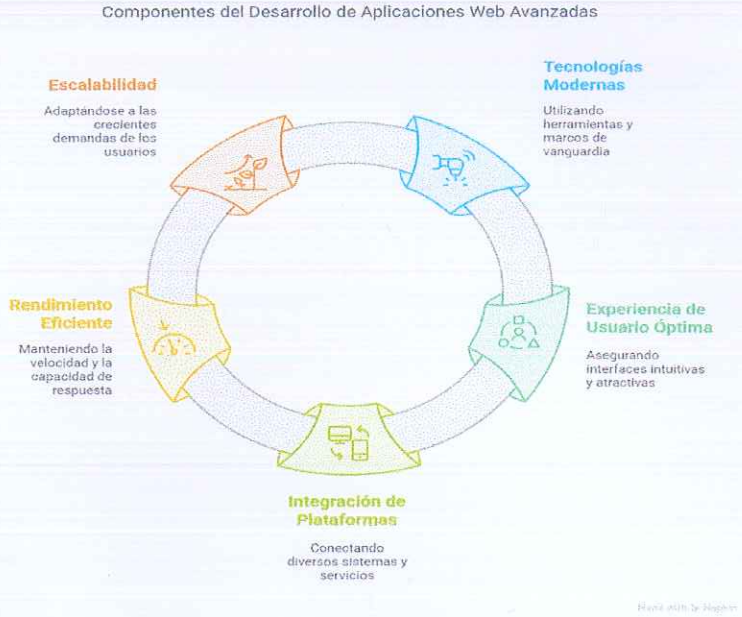
3.1 JavaScript y TypeScript

3.2 Frameworks FrontEnd

3.3 Modo producción y ejecutables del sistema FrontEnd

Utiliza herramientas y tecnologías de programación para llevar a cabo tareas específicas en el campo de desarrollo de software.

DIAGRAMA DE APRENDIZAJE



SINTESIS.

La unidad aborda el desarrollo de aplicaciones web avanzadas, enfocándose en el uso de frameworks de back-end como Ruby on Rails, Django y Spring Boot para estructurar y mantener proyectos complejos. Se exploran tecnologías de comunicación como WebSockets y GraphQL para mejorar la interactividad y eficiencia en tiempo real, así como estrategias para construir aplicaciones escalables y distribuidas. Además, se enfatiza la importancia de las pruebas y la depuración para garantizar la calidad, confiabilidad y rendimiento de las aplicaciones desarrolladas.

3.1. Frameworks de desarrollo back-end

Estudia frameworks que facilitan la creación de aplicaciones web completas y mantenibles, proporcionando estructuras base, funciones predefinidas y buenas prácticas:

- Ruby on Rails (RoR): Framework basado en Ruby, orientado al desarrollo ágil. Promueve la productividad mediante el principio DRY ("Don't Repeat Yourself") y la convención sobre configuración.
- Django: Framework de Python enfocado en seguridad, escalabilidad y rapidez. Utiliza arquitectura Modelo-Template-Vista, ideal para sitios complejos y con gran contenido.
- Spring Boot: Framework de Java modular y potente, diseñado para aplicaciones empresariales, microservicios y sistemas distribuidos.

3.2. Integración de tecnologías modernas

Analiza herramientas que mejoran la comunicación y eficiencia de las aplicaciones:

- WebSockets: Protocolo que mantiene una conexión persistente entre cliente y servidor, permitiendo comunicación bidireccional en tiempo real. Ideal para chats, juegos o paneles de control.
- GraphQL: Tecnología que permite al cliente solicitar únicamente los datos necesarios desde una API, reduciendo tráfico de red y optimizando el rendimiento frente a REST.

La combinación de estas tecnologías permite crear aplicaciones dinámicas, interactivas y centradas en el usuario.

3.3. Desarrollo de aplicaciones escalables y distribuidas

Aborda cómo diseñar sistemas capaces de crecer sin perder rendimiento:

- Escalabilidad: Capacidad de adaptarse a más usuarios o datos. Puede ser vertical (mejorar un servidor) u horizontal (añadir más servidores).
- Aplicaciones distribuidas: Compuestas por múltiples componentes en servidores distintos, aumentando disponibilidad, tolerancia a fallos y flexibilidad. Común en microservicios y la nube.
- Buenas prácticas: Uso de balanceo de carga, caché, bases de datos distribuidas y monitoreo continuo para mantener rendimiento y estabilidad.

3.4. Pruebas y depuración

Destaca la importancia de asegurar la calidad del software antes de su despliegue:

- Pruebas: Garantizan que la aplicación funciona correctamente y cumple requisitos.
  - *Unitarias:* Evalúan funciones o componentes individuales.
  - *Integración:* Verifican interacción entre varios componentes.
  - *Automatización:* Asegura calidad continua y eficiente.
- Depuración: Proceso de identificar y corregir errores usando depuradores, puntos de interrupción y registros. Permite resolver problemas antes de afectar al usuario final.

ELABORACIÓN, REVISIÓN Y APROBACIÓN DE PARES	
Profesor	
Ing. Agustín Gonzalo Guanipatín Ramírez	
Fecha de elaboración: 31/10/2025	
Comisión de revisión de pares de guías de estudio del Instituto Superior Tecnológico Tena	
Lcda. María Angélica Campoverde Encalada	Mg. Alvaro Santiago Toalombo Díaz
Mg. Henry Fabian Chango Chango	Mg. Duarte Mora Martha Janina
Abg. Danilo Alexander Zamora Núñez., Mg.	
Fecha de revisión: 28/11/2025	
Coordinador de Investigación, Desarrollo Tecnológico e Innovación	
Abg. Danilo Alexander Zamora Núñez., Mg.	INSTITUTO SUPERIOR TECNOLÓGICO TENA COORDINACIÓN DE INVESTIGACIÓN
Fecha de aprobación: 09/12/2025	