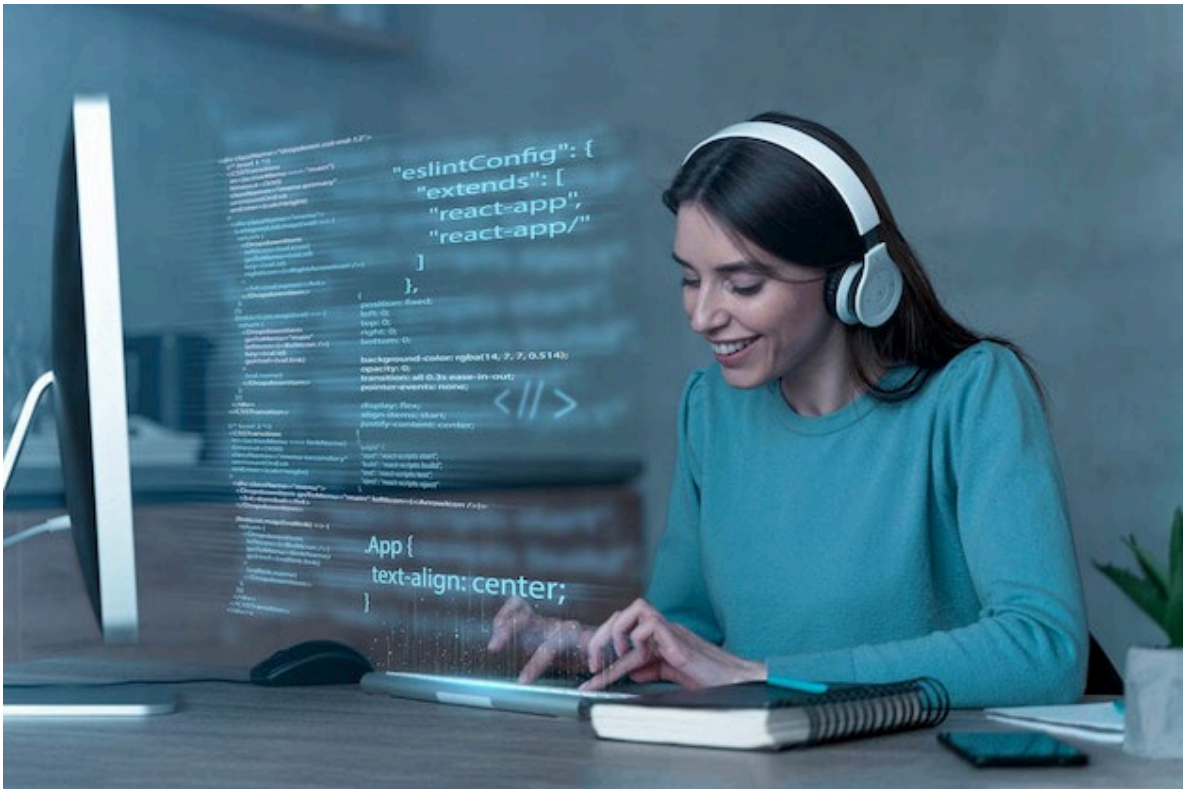

CARRERA TECNOLOGICA DE DESARROLLO DE APLICACIONES WEB FULL STACK (600 HORAS)

Esta carrera está diseñada para formar desde cero a personas interesadas en convertirse en desarrolladores web Full Stack, capaces de diseñar, construir e implementar aplicaciones web modernas tanto en el lado del cliente (*frontend*) como en el lado del servidor (*backend*). La formación incluye programación, diseño de interfaces, bases de datos, APIs y despliegue de aplicaciones, utilizando herramientas y tecnologías demandadas en el mercado laboral.

No se requieren conocimientos previos de programación. Al finalizar la carrera, el estudiante contará con las competencias necesarias para acceder a empleos como desarrollador web junior, participar en proyectos freelance o continuar su formación hacia certificaciones internacionales.

Público a quien va dirigido

Esta carrera está dirigida a personas con **interés en aprender a programar desde cero**, sin necesidad de tener conocimientos previos en informática o desarrollo de software. Está especialmente diseñada para:

- Bachilleres recién graduados.
- Estudiantes universitarios de cualquier carrera.
- Personas que desean reconvertirse profesionalmente hacia el mundo de la tecnología.

- Emprendedores que deseen crear soluciones digitales propias.
- Trabajadores que deseen adquirir habilidades digitales para mejorar su perfil laboral.

Requisitos previos

- No se requieren conocimientos previos de programación o tecnología.
- Solo necesitas: Tener habilidades básicas en el uso del computador (navegar por Internet, crear documentos, utilizar carpetas y archivos).
- Contar con un computador con conexión a Internet estable.
- Tener la disposición de dedicar entre 8 y 12 horas semanales para clases, prácticas y proyectos.

¿Qué lograrás al culminar esta carrera?

Al finalizar la carrera, el estudiante será capaz de:

- Diseñar y desarrollar aplicaciones web dinámicas, interactivas y seguras.
- Implementar soluciones del lado del cliente utilizando HTML, CSS, JavaScript y frameworks modernos.
- Crear aplicaciones del lado del servidor con Node.js, Express o tecnologías equivalentes.
- Diseñar y administrar bases de datos relacionales utilizando SQL y PostgreSQL.
- Crear y consumir APIs RESTful para la comunicación entre el cliente y el servidor.
- Desplegar proyectos web en servidores de desarrollo o plataformas cloud como Vercel, Render o Netlify.
- Trabajar en proyectos colaborativos utilizando Git y GitHub.
- Continuar su formación hacia certificaciones técnicas internacionales en desarrollo web.

CONTENIDO PROGRAMATICO

UNIDAD I - INTRODUCCIÓN AL DESARROLLO DE SOFTWARE (128 H)

MODULO1: FUNDAMENTOS DE LÓGICA, COMPUTACIÓN Y PENSAMIENTO COMPUTACIONAL (40 Horas)

Objetivo: Brindar una base sólida en lógica, resolución de problemas, algoritmos y fundamentos de programación, con enfoque práctico y visual.

Contenido:

1. **Pensamiento Computacional y Resolución de Problemas**
    - ¿Qué es programar?

- Lógica cotidiana vs lógica computacional.
- Introducción a algoritmos y pseudocódigo.
- Diagramas de flujo: símbolos, reglas y ejercicios.
- Actividades prácticas con problemas cotidianos.

2. **Introducción a la Lógica de Programación**
    - Variables, tipos de datos y operadores lógicos y aritméticos.
    - Estructuras de control: condicionales y bucles.
    - Arreglos y estructuras básicas.
    - Ejercicios con pseudocódigo y diagramas.
3. **Primer Lenguaje de Programación - Python Inicial**
    - Instalar Python y usar IDLE o Visual Studio Code.
    - Sintaxis básica, input/output, condiciones y ciclos.
    - Desarrollo de pequeños scripts y ejercicios prácticos.
    - Proyecto final: Juego o simulador simple (ej: adivinanzas, calculadora, menú interactivo).

MODULO 2: MANEJO DE CONTROL DE VERSIONES CON GIT (24 HORAS)

El objetivo de este curso es proporcionar a los participantes una comprensión sólida de Git, un sistema de control de versiones distribuido ampliamente utilizado en el desarrollo de software. Los participantes aprenderán los conceptos fundamentales de Git y cómo utilizarlo de manera efectiva en su trabajo diario como desarrolladores de software.

Contenido:

4. **Introducción a Git**:
    - ¿Qué es Git? Conceptos básicos y beneficios.
    - Instalación y configuración de Git en diferentes plataformas (Windows, macOS, Linux).
    - Configuración inicial de un repositorio Git.
5. **Conceptos fundamentales de Git**:
    - Control de versiones: seguimiento de cambios, ramificación, fusión.
    - Ciclo de vida de un archivo en Git: áreas de trabajo (working directory), área de preparación (staging area) y repositorio (repository).
    - Comandos básicos de Git: init, add, commit, status, log.

6. **Trabajo con ramas (branches) en Git**:
    - Creación y gestión de ramas.
    - Fusión de ramas: resolución de conflictos.
    - Estrategias de ramificación: ramas de características, ramas de lanzamiento, ramas de corrección de errores.
7. **Colaboración con Git**:
    - Clonación de repositorios remotos.
    - Sincronización con repositorios remotos: pull, push.
    - Colaboración en proyectos: forks, pull requests.
8. **Flujo de trabajo con Git**:
    - Trabajo con commits: mensajes descriptivos, commits atómicos.
    - Uso de etiquetas (tags) para marcar versiones importantes.
    - Revertir cambios y deshacer commits.

MODULO 3: SCRUM DEVELOPER (24 HORAS)

Este módulo tiene como objetivo proporcionar a los participantes una comprensión sólida de los principios y prácticas de desarrollo de software ágil utilizando el marco de trabajo Scrum. Los participantes aprenderán las habilidades y técnicas necesarias para trabajar de manera efectiva en equipos ágiles y contribuir al éxito de los proyectos de desarrollo de software.

1. **Introducción a Scrum y metodologías ágiles**:
    - Principios y valores del desarrollo ágil.
    - Conceptos básicos de Scrum: roles (Product Owner, Scrum Master, Scrum Team), artefactos (Product Backlog, Sprint Backlog, Incremento) y eventos (Sprint, Sprint Planning, Daily Scrum, Sprint Review, Sprint Retrospective).
2. **Desarrollo de software en Scrum**:
    - Planificación y estimación ágil: técnicas de estimación (Story Points, Planning Poker), elaboración de Product Backlog, creación de Sprint Backlog.
    - Desarrollo iterativo e incremental: realización de entregas parciales de funcionalidad al final de cada sprint.
    - Técnicas de desarrollo ágil: integración continua, desarrollo guiado por pruebas (TDD), refactorización de código, diseño emergente.
3. **El rol del Scrum Developer y trabajo en equipo**:
    - Responsabilidades del Scrum Developer.

- Comunicación efectiva en equipos ágiles: reuniones diarias (Daily Scrum), retrospectivas de sprint, revisión de sprint.
- Colaboración con el Scrum Master y el Product Owner.
- Colaboración con el Product Owner y los stakeholders: definición de requisitos, priorización del backlog, revisión del producto.
- Planificación y ejecución de tareas dentro de un Sprint.

4. **Evaluación del Desempeño y Mejora Continua**
    - Medición de productividad y KPIs en equipos Scrum.
    - Identificación de obstáculos y mejora continua.
    - Estrategias para mantener la velocidad del equipo.
5. **Uso de la herramienta Kanban**:
    - Tablero Kanban: Utiliza un tablero físico o digital dividido en columnas que representan etapas del proceso, como "Por hacer", "En progreso" y "Terminado".
    - Tarjetas de trabajo: Cada tarea se representa con una tarjeta que se mueve a través de las columnas a medida que avanza en el proceso. Las tarjetas contienen información relevante sobre la tarea, como descripción, responsable y fecha límite
    - Limitación del trabajo en progreso (WIP): Kanban limita la cantidad de tareas que pueden estar en progreso simultáneamente en cada columna, lo que ayuda a evitar la sobrecarga y a mantener un flujo de trabajo constante.
    - Visualización del flujo de trabajo: Kanban proporciona una visualización clara del estado actual de cada tarea y del flujo general de trabajo, lo que facilita la identificación de cuellos de botella y la optimización del proceso.
    - Iteración y mejora continua: Kanban fomenta la colaboración del equipo y la mejora continua del proceso mediante la revisión periódica del tablero, la identificación de áreas de mejora y la implementación de cambios.
6. Preparación para los Examenes de Certificación
    - Revisión de temas clave del examen Scrum Developer Certified (SDC).
    - Simulacro de examen y análisis de preguntas.
    - Consejos y estrategias para aprobar el examen de CertiProf.

MODULO 4: PROGRAMACIÓN ORIENTADA A OBJETO. (40 HORAS)

Este módulo tiene como objetivo introducir a los estudiantes en los conceptos fundamentales de la programación orientada a objetos (POO) y proporcionarles las

habilidades necesarias para diseñar y desarrollar aplicaciones utilizando este paradigma de programación.

1. **Introducción a la programación orientada a objetos (POO)**:
    - Conceptos básicos de POO: objetos, clases, atributos, métodos, encapsulamiento, herencia, polimorfismo.
    - Principios SOLID: SRP (Principio de responsabilidad única), OCP (Principio de abierto/cerrado), LSP (Principio de sustitución de Liskov), ISP (Principio de segregación de interfaces), DIP (Principio de inversión de dependencia).
2. **Lenguajes de programación orientada a objetos**:
    - Introducción a un lenguaje de programación orientado a objetos, como Java, C++, Python o C#.
    - Sintaxis básica y características del lenguaje: declaración de clases, creación de objetos, definición de métodos, encapsulamiento, herencia, polimorfismo.
3. **Diseño de clases y objetos**:
    - Identificación de clases y objetos en un problema.
    - Definición de atributos y métodos de clase.
    - Relaciones entre clases: asociación, agregación, composición.
4. **Herencia y polimorfismo**:
    - Definición y uso de la herencia para reutilizar y extender funcionalidades.
    - Implementación del polimorfismo para manejar objetos de diferentes tipos a través de una interfaz común.
5. **Aplicación de la programación orientada a objetos en proyectos reales**:
    - Desarrollo de aplicaciones prácticas utilizando los conceptos aprendidos.
    - Resolución de problemas mediante el diseño de soluciones orientadas a objetos.

UNIDAD II - HERRAMIENTAS EN EL DESARROLLO DE SOFTWARE (120 H)

MODULO 5: PROGRAMACIÓN DE BASES DE DATOS RELACIONALES (40 HORAS)

Este curso tiene como objetivo proporcionar a los estudiantes una comprensión básica de la programación de bases de datos, incluyendo conceptos fundamentales, diseño de bases de datos y consulta de datos utilizando SQL.

Contenido:

1. **Introducción a las bases de datos**

- ¿Qué es una base de datos?
- Importancia de las bases de datos en el desarrollo de software.
- Tipos de bases de datos: relacionales, no relacionales.

2. **Modelado de datos**
    - Conceptos básicos de modelado de datos: entidad, atributo, relación.
    - Diseño de bases de datos relacionales: creación de tablas, definición de claves primarias y foráneas.
    - Conceptos de normalización: formas normales (1NF, 2NF, 3NF).
    - Importancia de la normalización en el diseño de bases de datos.
    - Aplicación de técnicas de normalización en bases de datos existentes.
3. **Lenguaje de Consulta Estructurado (dependiendo la BDD)**
    - Introducción a SQL: historia y características.
    - Consultas básicas con SELECT: proyección, filtrado, ordenamiento.
    - Consultas avanzadas con JOIN: INNER JOIN, LEFT JOIN, RIGHT JOIN.
    - Operaciones de manipulación de datos: INSERT, UPDATE, DELETE.
    - Funciones y operadores en SQL: SUM, AVG, COUNT, LIKE, BETWEEN, etc.
4. **Práctica y ejercicios**
    - Sesiones prácticas de consulta y manipulación de datos utilizando SQL.
    - Resolución de problemas y ejercicios prácticos para reforzar los conceptos aprendidos.
5. **Evaluación y retroalimentación**
    - Revisión de conocimientos mediante pruebas o ejercicios de evaluación.
    - Retroalimentación sobre el desempeño y áreas de mejora para cada estudiante.

MODULO 6: ACCESO A DATOS (40 HORAS)

Este curso tiene como objetivo proporcionar a los estudiantes una comprensión sólida de los fundamentos del acceso a datos en el desarrollo de software, incluyendo técnicas de almacenamiento, recuperación y manipulación de datos.

Contenido:

1. **Introducción a la gestión de datos**
    - ¿Qué es el acceso a datos?

- Importancia del acceso a datos en el desarrollo de software.
- Tipos de datos: estructurados, no estructurados, semiestructurados.

2. **Acceso a datos en aplicaciones** (Depende de la tecnología)
    - Conexión a bases de datos desde aplicaciones: JDBC en Java, SQLAlchemy en Python, ADO.NET en C#.
    - Uso de ORM (Mapeo Objeto-Relacional) para simplificar el acceso a datos: Hibernate en Java, Entity Framework en C#.
    - Implementación de operaciones CRUD (Crear, Leer, Actualizar, Eliminar) en aplicaciones.
3. **Práctica y ejercicios**
    - Sesiones prácticas de acceso y manipulación de datos utilizando SQL y herramientas de acceso a datos.
    - Desarrollo de aplicaciones de ejemplo que utilizan acceso a datos para almacenar, recuperar y manipular información.
4. **Evaluación y retroalimentación**
    - Revisión de conocimientos mediante pruebas o ejercicios de evaluación.
    - Retroalimentación sobre el desempeño y áreas de mejora para cada estudiante.

MODULO 7: FUNDAMENTOS E IMPLEMENTACIÓN DE API REST (40 HORAS)

Este curso tiene como objetivo proporcionar a los estudiantes una comprensión sólida de los fundamentos de las API REST (Interfaz de Programación de Aplicaciones Representacional del Estado Transferido), incluyendo su diseño, implementación y consumo en aplicaciones web y móviles.

Contenido:

1. **Introducción a las API REST**
    - ¿Qué es una API REST?
    - Principios y características de las API REST.
    - Beneficios de las API REST en el desarrollo de software.
2. **Principios de diseño de API REST**
    - Conceptos básicos de diseño de API RESTful: recursos, URIs, métodos HTTP.
    - Convenciones de nomenclatura y estilo: CamelCase, snake_case, kebab-case.

- Versionado de API y control de cambios.

3. **Métodos HTTP y sus operaciones en API REST**
    - Métodos HTTP: GET, POST, PUT, PATCH, DELETE.
    - Operaciones CRUD (Crear, Leer, Actualizar, Eliminar) en API REST.
    - Manejo de parámetros de consulta y encabezados HTTP.
4. **Formatos de intercambio de datos**
    - JSON (JavaScript Object Notation): sintaxis, estructura de datos.
    - XML (eXtensible Markup Language): sintaxis, estructura de datos.
5. **Seguridad en API REST**
    - Autenticación y autorización: tokens JWT (JSON Web Tokens), OAuth.
    - Protección contra ataques comunes: CSRF (Cross-Site Request Forgery), XSS (Cross-Site Scripting).
6. **Documentación de API REST**
    - Importancia de la documentación de API.
    - Herramientas de documentación de API: Swagger, OpenAPI Specification.
    - Mejores prácticas para escribir documentación clara y completa.
7. **Práctica y ejercicios**
    - Sesiones prácticas de diseño, implementación y consumo de API REST.
    - Desarrollo de aplicaciones de ejemplo que utilizan una API REST para intercambiar datos.
8. **Evaluación y retroalimentación**
    - Revisión de conocimientos mediante pruebas o ejercicios de evaluación.
    - Retroalimentación sobre el desempeño y áreas de mejora para cada estudiante.

UNIDAD III – DISEÑO Y PROGRAMACIÓN EN FRONTEND (120 HORAS)

MÓDULO 8: FUNDAMENTOS DE DESARROLLO WEB (40 HORAS)

Objetivo: Brindar a los estudiantes los conocimientos básicos para crear páginas web estáticas con HTML, CSS y JavaScript.

Contenido:

1. **Introducción al desarrollo web**

- Funcionamiento de la web: cliente-servidor, navegador, HTTP.
- Estructura básica de un proyecto frontend.
- Herramientas de desarrollo (editores, extensiones, navegadores).

2. **HTML5**
   - Estructura del documento HTML.
   - Etiquetas semánticas: <header>, <nav>, <section>, <article>, <footer>.
   - Formularios, tablas y elementos multimedia.
3. **CSS3**
   - Selectores, propiedades y cascada.
   - Modelo de cajas (box model), posicionamiento, flexbox y grid.
   - Transiciones, animaciones básicas, diseño responsive con media queries.
4. **JavaScript básico**
   - Variables, tipos de datos, operadores.
   - Condicionales, bucles y funciones.
   - Manipulación del DOM (Document Object Model).
   - Eventos y validaciones básicas.
5. **Proyecto práctico:** Crear una página web personal o un portafolio interactivo.

MÓDULO 9: DESARROLLO WEB INTERACTIVO CON FRAMEWORK (40 HORAS)

Objetivo: Introducir a los estudiantes en el uso de frameworks modernos de JavaScript para crear aplicaciones web dinámicas e interactivas.

Contenido:

1. **JavaScript moderno (ES6+)**
   - Let, const, arrow functions, template literals.
   - Objetos, arrays, desestructuración, promesas y async/await**.**
2. **Introducción a React.js (o Vue.js, según preferencia)**
   - Componentes, props y estado.
   - Eventos, formularios controlados.
   - Ciclo de vida del componente.

- Hooks básicos: useState, useEffect.

3. **Manejo de rutas**
    - SPA (Single Page Application) y React Router (o Vue Router).
4. **Consumo de APIs REST**
    - Fetch API / Axios.
    - Mostrar datos en componentes.
5. **Proyecto práctico:** Crear una pequeña aplicación web dinámica (ej. lista de tareas, buscador de películas, galería de imágenes).

MÓDULO 10: INTERFACES MODERNAS Y BUENAS PRÁCTICAS (40 HORAS)

Objetivo: Enseñar a los estudiantes a aplicar principios de diseño y buenas prácticas para mejorar la experiencia del usuario (UX) y el diseño visual (UI).

Contenido:

1. **Diseño centrado en el usuario (UX/UI)**
    - Principios básicos de diseño de interfaces.
    - Accesibilidad web (WCAG).
    - Usabilidad y patrones de diseño comunes.
2. **Frameworks de diseño y estilos**
    - Uso de Bootstrap o Tailwind CSS.
    - Sistemas de diseño y componentes reutilizables.
3. **Control de versiones aplicado a frontend**
    - Uso de Git y GitHub para colaboración en frontend.
    - Buenas prácticas: estructura de carpetas, commits claros, ramas.
4. **Optimización y despliegue**
    - Minimización de recursos.
    - Herramientas de construcción: Webpack, Vite, etc.
    - Despliegue en servicios como GitHub Pages, Vercel o Netlify.

5. **Proyecto final integrador:** Aplicación web con frontend moderno y consumo de API REST desarrollada en las unidades anteriores.

UNIDAD IV - APLICACIONES, PRUEBAS Y VERSIONES (80H)

Objetivo: Consolidar conocimientos mediante proyectos reales, pruebas automatizadas y despliegue básico.

MÓDULO 11: TESTING Y CONTROL DE CALIDAD (40H)

- Tipos de pruebas: unitarias, de integración, funcionales.
- Herramientas de pruebas: JUnit, PyTest, Mocha (según lenguaje).
- Escribir pruebas, mocks y cobertura de código.
- Automatización de pruebas en CI/CD.

MÓDULO 12: PROYECTO FINAL Y CONSUMO DE API (40H)

- Elección de un caso de uso: Sistema de inventario, sistema de reservas, blog, etc.
- Fases: análisis, diseño, desarrollo, pruebas y despliegue.
- Presentación final con documentación técnica.

UNIDAD V – DESPLIEGUE Y ORQUESTACIÓN EN EL DESARROLLO DE SOFTWARE (88 HORAS)

MODULO 13: DESARROLLO Y DESPLIEGUE DE APLICACIONES CON DOCKER (24 HORAS)

En este módulo te enseñaremos como comprender los conceptos fundamentales de Docker y la contenerización. Aprenderás a crear, gestionar y desplegar contenedores Docker y las herramientas y técnicas para trabajar eficientemente con Docker. Exploraras casos de uso comunes y prácticas recomendadas para el desarrollo y despliegue de aplicaciones con Docker.

Contenido:

1. **Introducción a Docker**
    - ¿Qué es Docker?
    - Ventajas de utilizar contenedores.
    - Instalación de Docker en diferentes sistemas operativos.
    - Conceptos básicos: imágenes, contenedores, volúmenes, redes.
2. **Creación y gestión de contenedores**

- Creación de imágenes Docker.
- Ejecución de contenedores.
- Gestión de contenedores en Docker CLI.
- Trabajo con Docker Hub.

3. **Configuración de redes y volúmenes**
    - Configuración de redes en Docker.
    - Uso de volúmenes para persistencia de datos.
    - Conexión de contenedores en redes personalizadas.
4. **Docker Compose**
    - Introducción a Docker Compose.
    - Creación y gestión de aplicaciones multicontenedor.
    - Definición de servicios y dependencias en Docker Compose.
5. **Orquestación con Docker Swarm**
    - Introducción a Docker Swarm.
    - Creación de un clúster de Docker Swarm.
    - Despliegue y gestión de servicios en Swarm.
6. **Monitorización y Logging**
    - Uso de herramientas para monitorización de contenedores.
    - Configuración de registros (logs) en Docker
    - Análisis de registros para depuración y seguimiento.
7. **Seguridad en Docker**
    - Buenas prácticas de seguridad en Docker.
    - Configuración de políticas de acceso y control de recursos.
    - Gestión de secretos y credenciales sensibles.
8. **Prácticas recomendadas y casos de uso**
    - Casos de uso comunes de Docker en diferentes entornos.
    - Estrategias para optimizar el uso de Docker en el desarrollo y despliegue de aplicaciones.
    - Recomendaciones para seguir aprendiendo y profundizando en Docker.

MODULO 14: ORQUESTACIÓN DE APLICACIONES EN KUBERNETES (24 HORAS)

En este módulo comprenderás los conceptos fundamentales de Kubernetes y la orquestación de contenedores. Aprenderás a implementar, gestionar y escalar aplicaciones utilizando Kubernetes. Utilizaras las herramientas y técnicas para trabajar eficientemente con Kubernetes. Exploraras casos de uso comunes y prácticas recomendadas para el despliegue de aplicaciones con Kubernetes.

Contenido:

1. **Introducción a Kubernetes**
    - ¿Qué es Kubernetes?
    - Principios de orquestación de contenedores.
    - Arquitectura de Kubernetes.
    - Instalación de Kubernetes en local (minikube).
2. **Pods y Servicios**
    - Concepto de pods en Kubernetes.
    - Creación y gestión de pods.
    - Uso de servicios para acceder a los pods.
    - Exposición de servicios dentro y fuera del clúster.
3. **Despliegue de Aplicaciones**
    - Despliegue de aplicaciones en Kubernetes.
    - Estrategias de despliegue (Rolling Update, Blue-Green Deployment, Canary Deployment).
    - Actualización y reversión de aplicaciones.
4. **Escalado y Balanceo de Carga**
    - Escalado automático y manual de aplicaciones.
    - Configuración de equilibrio de carga en Kubernetes.
    - Gestión de tráfico con Ingress Controllers.
5. **Almacenamiento Persistente**
    - Configuración de almacenamiento persistente en Kubernetes.
    - Uso de volúmenes persistentes.
    - Gestión de almacenamiento dinámica.
6. **Configuración y Gestión de Redes**

- Configuración de redes en Kubernetes.
- Uso de Políticas de Red para controlar el tráfico de red.
- Exposición de servicios mediante servicios NodePort y LoadBalancer.

7. **Secretos y ConfigMaps**
    - Gestión de secretos sensibles en Kubernetes.
    - Uso de ConfigMaps para la configuración de aplicaciones.
    - Seguridad en la gestión de Secrets y ConfigMaps.
8. **Monitoreo y Logging**
    - Configuración de herramientas de monitoreo en Kubernetes.
    - Captura y análisis de registros de contenedores.
    - Uso de métricas para el monitoreo de la salud de la aplicación.
9. **Seguridad en Kubernetes**
    - Buenas prácticas de seguridad en Kubernetes.
    - Configuración de políticas de acceso y control de recursos.
    - Auditoría y registro de eventos en Kubernetes.
10. **Prácticas Recomendadas y Casos de Uso**
    - Casos de uso comunes de Kubernetes en diferentes entornos.
    - Estrategias para optimizar el uso de Kubernetes en el despliegue de aplicaciones.
    - Recomendaciones para seguir aprendiendo y profundizando en Kubernetes.

MODULO 15: APLICACIONES EN LA NUBE PARA DESARROLLADORES DE SOFTWARE (40 HORAS)

Los objetivos de este curso son familiarizar a los desarrolladores de software con las aplicaciones y servicios en la nube más utilizados en el desarrollo moderno. Enseñar las mejores prácticas para aprovechar las herramientas en la nube en diferentes etapas del ciclo de desarrollo de software. Capacitar a los participantes para utilizar eficazmente las plataformas en la nube y aumentar su productividad en el desarrollo de aplicaciones.

Contenido:

1. **Introducción a las Aplicaciones en la Nube**
    - ¿Qué son las aplicaciones en la nube?
    - Ventajas de utilizar servicios en la nube en el desarrollo de software.

- Principales proveedores de servicios en la nube (AWS, Azure, Google Cloud).

2. **Desarrollo Ágil con Plataformas en la Nube**
    - Metodologías ágiles en el desarrollo de software.
    - Herramientas de gestión de proyectos en la nube (Jira, Trello, Asana).
    - Colaboración remota con herramientas de comunicación en la nube (Slack, Microsoft Teams, Zoom)
    - Realización de ejercicios prácticos utilizando las herramientas presentadas para planificar y gestionar un proyecto ágil ficticio.
    - Simulación de reuniones diarias y retrospectivas utilizando herramientas de comunicación en la nube.
3. **Almacenamiento y Gestión de Datos en la Nube**
    - Servicios de almacenamiento en la nube (Amazon S3, Google Cloud Storage, Azure Blob Storage).
    - Bases de datos en la nube (Amazon RDS, Google Cloud SQL, Azure SQL Database).
    - Estrategias de respaldo y recuperación de datos en la nube.
4. **Desarrollo de Aplicaciones Serverless**
    - Conceptos de computación sin servidor (serverless).
    - Funciones como servicio (FaaS) en la nube (AWS Lambda, Google Cloud Functions, Azure Functions).
    - Implementación de aplicaciones sin servidor en la nube.
5. **Implementación Continua y Despliegue Continuo (CI/CD) en la Nube**
    - Automatización de pruebas e implementación de código en la nube.
    - Herramientas de integración continua y entrega continua (Jenkins, CircleCI, GitLab CI).
    - Estrategias para implementar CI/CD de manera efectiva en el desarrollo de software.
6. **Monitoreo y Análisis de Aplicaciones en la Nube**
    - Herramientas de monitoreo de aplicaciones en la nube (AWS CloudWatch, Google Cloud Monitoring, Azure Monitor).
    - Métricas y alertas para garantizar el rendimiento y la disponibilidad de las aplicaciones.

- Análisis de registros (logs) para identificar y solucionar problemas de rendimiento.

7. **Seguridad en las Aplicaciones en la Nube**
    - Principios de seguridad en la nube.
    - Herramientas y servicios de seguridad en la nube (AWS Identity and Access Management, Google Cloud Identity-Aware Proxy, Azure Active Directory).
    - Mejores prácticas para proteger aplicaciones y datos en la nube.
8. **Escalabilidad y Tolerancia a Fallos en la Nube**
    - Estrategias de escalabilidad horizontal y vertical en la nube.
    - Configuración de equilibrio de carga y escalada automática.
    - Diseño de arquitecturas resilientes para aplicaciones en la nube.
9. **Casos de Estudio y Prácticas Recomendadas**
    - Análisis de casos reales de aplicación de servicios en la nube en el desarrollo de software.
    - Recomendaciones y mejores prácticas para maximizar el rendimiento y la eficiencia en la nube.
    - Sesión interactiva de preguntas y respuestas.

UNIDAD VI - COMPETENCIAS PARA EL DESARROLLO PROFESIONAL (64 Horas)

Objetivo: Preparar al estudiante para el mundo laboral y certificaciones básicas.

MÓDULO 16: DESARROLLO PROFESIONAL Y EMPLEABILIDAD (24H)

- Elaboración de portafolio en GitHub.
- Currículum para programadores junior.
- Simulación de entrevistas técnicas.
- Soft Skills en TI: trabajo en equipo, comunicación y liderazgo.

MÓDULO 17: PREPARACIÓN PARA CERTIFICACIONES (40H)

- Revisión y simulacro de exámenes:
    - **Scrum Developer Certified**
    - **CertiProf – Fundamentos de Programación**
    - **Certificación C#, Java Foundations o Python Essentials (según ruta)**