Especialización en Desarrollo de Aplicaciones Front-End

El curso "Especialización en Desarrollo de Aplicaciones Front-End" está diseñado para desarrolladores interesados en especializarse en la creación y mantenimiento de interfaces de usuario dinámicas y responsivas utilizando tecnologías modernas del front-end. Dirigido a profesionales con título universitario, técnico o profesional y con al menos dos años de experiencia comprobable en proyectos tecnológicos, este curso capacita a los participantes para trabajar en diversas industrias como retail, banca, salud, minería, manufactura y servicios TI. Los participantes adquirirán habilidades para diseñar y desarrollar interfaces de usuario utilizando HTML5, CSS3 y JavaScript, y para trabajar con frameworks modernos como React, mejorando así su desempeño laboral y aumentando sus posibilidades de empleabilidad y ascenso

Módulos de aprendizaje

Módulo 1: Orientación al perfil y metodología del curso

Objetivo específico

Comprender el perfil del desarrollador front-end, las competencias necesarias y la metodología del curso.

Contenidos

- La Industria TI
    - Características de la Industria
    - Perfiles más Comunes en la Industria TI
    - CompetenciasTécnicas Valoradas por la Industria TI
    - Habilidades Personales Valoradas por la Industria TI
    - Metodologías y Forma deTrabajo del Área
- El Perfil Profesional Asociado al Curso
    - Qué es un Perfil Profesional
    - Competencias que Posee el Perfil
    - Habilidades que Posee el Perfil
    - Niveles de Experiencia y Seniority del Perfil
    - Expectativas Laborales del Mercado Actual para el Perfil
    - Proyección Laboral del Perfil
- Curricula del Curso
    - Módulos y Competencias a Formar alo Largo de la Curricula
    - Herramientas a Utilizar Durante el Curso
    - Características del Trabajo Técnico a Realizar en Cada Módulo
    - ProductosObtenidos en Cada Módulo
- Portafolio de Producto
    - Qué es un Portafolio de Producto
    - Importancia de un Portafolio de Producto en la Identidad Profesional
    - Metodología de Enseñanza-Aprendizaje
    - Qué es el Aprendizaje Activo

    - Metodologías Utilizadas a lo Largo del Curso
    - Rol del Facilitador y del Participante
- Herramientas a Utilizar a lo Largo del Curso
    - Herramientas de Gestión del Proceso de Aprendizaje (LMS)
    - Herramientas de Coordinación y Trabajo Colaborativo
    - Herramientas Propias de la Competencia Técnica
- Habilidades Requeridas a lo Largo del Curso
    - Trabajo en Equipo Autoaprendizaje
    - Tolerancia a la Frustración
    - ComportamientoÉtico
        - Importancia del Comportamiento Ético en la Industria TI
        - Código de Ética del Curso

Módulo 2: Desarrollo de la interfaz de usuario Web

Objetivo específico

Implementar interfaces de usuario responsivas utilizando HTML5, CSS y frameworks de estilos.

Contenidos

- El Proceso de Desarrollo de un Producto Visual
    - Diferencias entre Front-End, Back-End y Full-Stack
    - Aplicaciones SPA vs. monolitos
    - Renderización cliente y servidor
    - El proceso de ideación de un producto digital
    - El proceso de implementación de un producto digital
    - El rol del diseñador UX/UI
    - El rol del desarrollador Front-End
    - Importancia de las guías de estilos y representaciones visuales en la maquetación
- El Lenguaje de marcas
    - Qué es lenguaje de marcas
    - Qué es el DOM
    - Acerca de HTML4 y anteriores
    - HTML5 y sus diferencias sobre HTML tradicional
    - HTML5 en dispositivos móviles
    - Qué es XHTML y cómo se usa
- Metodologías para la organización y modularización de estilos
    - Ventajas de utilizar una metodología
    - Metodologías más utilizadas (BEM, OOCSS, SMACSS)
- Preprocesadores CSS
    - Qué es un procesador CSS
    - Importancia del preprocesador en el desarrollo front-end

    - Preprocesadores más utilizados (SASS, LESS)

- Conceptos avanzados del desarrollo web

    - Protocolo HTTP
        - Qué es el protocolo HTTP
        - Dónde se utiliza el protocolo HTTP
        - Entendiendo los métodos y sus usos
        - Qué se entiende por Endpoint
        - Interpretando los códigos de retorno
    - Qué es un CDN y cómo se usa
    - Implementando la seguridad con HTTPS
    - Entendiendo el Encoding de páginas
    - Acerca de Cookies y su uso en los navegadores
    - Compatibilidad del desarrollo web en los distintos navegadores
    - Entendiendo el LocalStorage y SessionStorage como almacenaje local de los datos en un navegador
    - Aplicaciones de consola: Otra forma de consumir aplicaciones Web (Lynx, Wget, Curl)

- Hojas de Estilo

    - Acerca de CSS y su uso en la interfaz de usuario Web
    - Ventajas de usar CSS
    - Desde CSS1 hasta CSS3
    - Aspectos avanzados de CSS
        - Patrones
        - Bloques de declaración
        - Uso de estilos
        - Fuentes
        - Especificidad
        - Herencia
        - Posicionamiento
    - Limitaciones de CSS
    - Soporte en distintos navegadores web
    - CSS4 y el futuro de las hojas de estilo

- Responsividad

    - Qué es Responsividad
    - Porqué desarrollar aplicaciones web Responsivas
    - Impacto de la responsividad en la experiencia de usuario
    - Cómo funciona la adaptabilidad a distintos anchos de pantalla
    - Diseño Web Responsivo vs. Diseño Adaptativo
    - Cómo crear aplicaciones web responsivas
    - Los bloques de construcción del Diseño WebResponsivo
    - Puntos de ruptura Responsivos comunes
    - Unidades y valores del CSS para el diseño responsivo

- Modelo de Cajas

    - Qué es el modelo de cajas. ¿Existen otros modelos?
    - Propiedades que componen el modelo de cajas
    - Tipos de cajas: Diferencias entre elementos de bloque y elementos de línea
    - Inspeccionando elementos con navegador para identificar las cajas

- Posicionamiento de Elementos y Visualización

    - Conceptos básicos de propiedades de posicionamiento
    - Tipos de posicionamiento (normal, relativo, absoluto, fijo, flotante)
    - Visualización

- Layout

    - Qué es un layout
    - Tipos de layout (fluido, fijo, elástico,absoluto)
    - Ventajas y desventajas

- El Preprocesador SASS

    - Qué es Sass y porqué utilizarlo
    - Conocer Sass y aprovechar las ventajas en el proceso de construcción de un sitio web
    - Instalar Sass y linters para editores de código
    - Conocer patrón 7-1
    - Sintaxis, flujo de trabajo y buenas prácticas usando Sass
        - Utilización de Sass desde línea de comandos
        - Instalación de plugins de Sass en editores de texto para automatización de tareas
        - Uso de variables para reutilización de código
        - Elementos anidados y namespaces
        - Manejo de parciales e imports
        - Manejo de mixins e includes

- Modularización de estilos

    - Porqué modularizar
    - Ventajas demodularizar los estilos
    - Utilización de metodologías de modularización
    - Arquitectura CSS

- Metodologías de modularización

    - Qué es una metodología de modularización
    - Para qué sirve una metodología
    - Ventajas de usar una metodología
    - Identificando algunas metodologías y sus características (BEM, OOCSS, SMACSS, ITCSS)

- Metodología BEM

    - En qué consiste esta metodología
    - Conceptos claves de BEM
    - Convención de nombres
    - Cómo estructurar un proyecto con BEM

- Metodología OOCSS
    - En qué consiste esta metodología
    - Principios básicos de OOCSS
- Metodología SMACSS
    - En qué consiste esta metodología
    - Categorías de elementos SMACSS
- Frameworks CSS
    - Qué es un Framework CSS, porqué y cuándo utilizarlo
    - Alternativas de frameworks basados en CSS (Bootstrap, Foundation, Pure CSS, Bulma)
    - Ventajas de usar un Framework CSS vs.HTMLtradicional
    - Componiendo estilos propios
- Analizando un Framework CSS: Bootstrap
    - Qué es Bootstrap
    - Ventajas
    - Filosofía Bootstrap
        - Responsividad
        - Principio Mobile First
    - Modificar y extender funcionalidad de Bootstrap con Sass
    - Conociendo layouts, contenidos y componentes de Bootstrap
        - Grillas
        - Botones
        - Paneles
        - Tablas
        - Formularios
        - Navegación y menús
    - Conociendo componentes JavaScript
        - Alertas
        - Breadcrumb
        - Card
        - Carrusel
        - Tooltip
        - Modal
        - Popover

Módulo 3: Programación avanzada en JavaScript

Objetivo específico

Codificar piezas de software de mediana complejidad utilizando JavaScript avanzado.

Contenidos

- Generalidades del Lenguaje JavaScript

    - Historia de JavaScript
    - Uso en navegadores web
    - Entornos virtuales de JavaScript
    - Diferencias entre JavaScript y otros lenguajes
    - Paradigmas de programación soportados por JavaScript
    - Fortalezas del lenguaje y forma correcta de uso
    - JavaScript como lenguaje asíncrono
    - Sintaxis e indentación
- Evolución del Lenguaje
    - Lenguaje interpretado vs. compilado
    - El estándar ECMAScript
    - JavaScript vs. ECMAScript
    - Evolución de ECMAScript: desde ES3 a ES9
    - Acerca de TypeScript y sus características
    - ¿Dónde y cómo se aplica TypeScript?
- El Stack JavaScript
    - Qué es unstack de desarrollo
    - Stacks basados en JavaScript (MEAN, MERN)
    - Aplicaciones para Frontend
    - Aplicaciones para Backend
    - Frameworks basados enJavaScript
    - Acceso a datos con JavaScript
    - Lenguajes derivados de JavaScript
    - Utilitarios
    - ElGarbageCollector en JavaScript
- Entorno de Ejecución JavaScript
    - Entorno de ejecución
    - La consola de comandos
    - Setting de variables
    - console.log()
    - Estructura de un programa JavaScript (import, require)
    - El eventloop (stack, heap, queue)
- Desarrollo de Aplicaciones JavaScript
    - Variables
        - Scope de variables
        - Hoisting
        - var vs. let vs. const
    - Tipos de datos primitivos
    - Operadores
    - Diferencia entre null y undefined
    - El operador typeof
    - El comparador == vs. ===
    - El tipado en JavaScript
    - Instrucciones condicionales
    - Ciclos e iteraciones
        - Ciclos y condiciones anidados (break, continue)

    - try/catch y manejo de errores
    - Expresiones regulares
    - Arreglos
    - Debugger
    - Manipulación de archivos
- Notación de Objetos JavaScript (JSON)
    - ObjetosJSON
        - Sintaxis y notación
        - Tipos de datos en objetos JSON
        - Objetos anidados
        - Funciones dentro de objetos
        - Asignación por destructuración
    - Operaciones con objetos JSON
        - Clonación
        - Merge
        - Recorrido, parse y stringify
    - JSON como base para el protocolo API REST
    - Otros formatos de intercambio de información
        - El lenguaje YAML
        - Comparación con XML
- Estructuras de Datos en JavaScript
    - Qué es una estructura de datos
    - Arreglos deDatos
        - Tipos(vectores, matrices)
        - Operaciones sobre arreglos
    - Listas
        - Tipos(simples, dobles, circulares)
        - Operaciones sobre listas
    - Pilas
    - Colas
    - Árboles
        - Tipos(binarios, ordenados, multicamino)
    - Conjuntos
    - Grafos
- Programación de Algoritmos
    - Porqué estudiar algoritmos
    - Eficiencia y Complejidad de un algoritmo (complejidad ciclomática, Big-O)
    - Algoritmos Recursivos
    - Algoritmos de Ordenamiento
    - Algoritmos de Búsqueda
    - Implementación en JavaScript
    - Técnicas deDebugging
    - Librerías que facilitan el trabajo con estructuras de datos (underscore, lodash,moment)
- Programación Funcional en JavaScript
    - El paradigma de programación funcional
    - Enfoque imperativo vs. declarativo

    - Funciones como objetos de primera clase
    - Funciones y procedimientos
        - Invocación de funciones
        - El paso de argumentos y retorno
        - La función flecha () => {}
        - Funciones dentro de objetos
        - Variables funciones
    - Recursión en programación funcional
    - "Currying" en programación funcional
    - Composición de funciones
- Programación Orientada a Eventos y Programación Asíncrona
    - ¿Qué es un evento?
    - Paradigma de la orientación a eventos
    - Captura eventos (listener)
    - Disparar eventos
    - Eventos en los navegadores Web
    - Programación síncrona vs. asíncrona
    - Blocking vs. non-blocking
    - Creación de Callbacks
    - Invocación de Callbacks
    - Creación de funciones ASYNC/AWAIT
    - Utilización de Promise/Resolve/Reject
    - Utilización de Catch/Throw y errores personalizados en llamadas asíncronas
- Programación Orientada a Objetos en JavaScript
    - En qué consiste el paradigma de programación orientada a objetos
    - Principios de la POO
        - Encapsulación
        - Abstracción
        - Herencia
        - Polimorfismo
    - Implementación de la POOenJavaScript
        - Objetos literales
        - MétodoCreate
        - FunciónConstructor
        - Herencia por prototipos
    - Implementación de la POOconECMAScript 2015
        - Definición de clases
        - Definición de subclases
        - Creación de instancias

Módulo 4: Desarrollo de interfaces interactivas con React

Objetivo específico

Construir aplicaciones web orientadas a componentes utilizando React.

Contenidos

- Introducción a ReactJs
    - ¿Qué es ReactJs y cuál es su origen?
    - ¿Porqué usar ReactJs?
    - Acerca de las aplicaciones SPA
    - Renderización de elementos del DOM
    - ReactJsyReactJs Native
    - Descripción y uso de ReactJs en aplicaciones web
    - ¿EsReactJs un framework?
    - ReactJs versus otros Frameworks basados en Javascript
    - ¿Qué NO es ReactJs?
- Características de ReactJs
    - Librería basada en Javascript
    - Lenguaje declarativo
    - Basado en componentes
    - Manejo del DOM
    - Isomorfismo (renderización en cliente y servidor)
    - Sintaxis JSX
    - ReactJs sin JSX
    - Flujo de datos unidireccional
- Entornos de Ejecución
    - Prerrequisitos para la utilización de ReactJS
    - Versiones deJavascript compatibles con ReactJs
    - Ejecución desde CDN
    - Ejecución desde un ambiente local
    - Librerías usadas por ReactJs (Webpack, Babel, Next.js, Redux)
- Elementos Fundamentales de ReactJS
    - Introducción a JSX
    - Pensando en ReactJs
    - Componentes en ReactJs
    - Composición versus herencia
    - Paso de datos con Props
    - Listas y keys
    - Formularios
    - El ciclo de vida ReactJs
    - Desplegando datos en la interfaz (Renderización)
    - Uso de extensión browser “React Developer Tools”
    - Hojas de estilo en ReactJs
    - Introducción a Hooks
    - Introducción a uso de API’s
- JSX, la Simplificación de ReactJS
    - Porqué usarJSX
    - Insertando expresiones en JSX
    - Notación y palabras reservadas
    - JSX también es una expresión
    - Especificando atributos con JSX
    - Especificando hijos con JSX

    - Previniendo ataques de inyección con JSX
    - Representación de objetos en JSX
    - El rol de Babel en la conversión de objetos
    - Comentarios en JSX
- Implementar componentes reutilizables
    - Describir los aspectos fundamentales y beneficios de la componentización en el desarrollo de aplicaciones front-end
    - Reconocerlos elementos y sintaxis requerida para la especificación de un componente
    - Utilizar componentes previamente construidos utilizando sus propiedades para su personalización
- Manejo del DOM
    - Elementos DOM en ReactJs
    - Acerca de React.Component
    - El DOM Virtual en ReactJs
        - Qué es el DOM Virtual
        - Qué es el Shadow DOM
        - Qué es React Fiber
    - Uso de referencias
        - Qué son las referencias
        - Cuándo usar referencias
        - Creación de referencias
        - Accediendo a las referencias
        - Agregando referencias al DOM
        - Referencias mediante Callback
    - El paquete React-DOM
        - React DOM en el cliente
        - React DOM en el servidor
        - Soporte en navegadores
    - Diferencias en atributos HTML que funcionan diferente en ReactJs (checked, className, onChange, selected)
    - Utilidades para pruebas
    - Renderizador de prueba
    - Requerimientos del entorno de JS
- Elementos avanzados de ReactJS
    - Manejo de socket.io
    - División de código
    - Transformar elementos
    - Contexto
    - Fragmentos
    - Transitions
    - Componentes de orden superior
    - Optimizando el rendimiento
    - Portales
    - Profiler
    - Reconciliación
    - Comprobación de tipos estáticos

    - Modo estricto
    - Verificación de tipos con PropTypes
    - Componentes no controlados
    - WebComponents

Módulo 5: Desarrollo de aplicaciones Front-End con React

Objetivo específico

Implementar aplicaciones web completas utilizando librerías avanzadas y prácticas de testing.

Contenidos

- Consumo de API
    - El rol del front en una aplicación Cliente/Servidor
    - Interacción a través de APIs
    - Usando el Hook useEffect
        - Qué hace useEffect
        - Omite efectos para optimizar el rendimiento
        - Cómo realizar peticiones en React con useEffect
        - Cómo realizar peticiones a partir de eventos del usuario
        - Manejando errores
    - Usando el Hookuse Sate
    - Usando Fetch API
        - Qué es Fetch API
        - Cómo utilizar Fetch API
        - Cómo manejar errores con Fetch API
    - UsandoAxios
        - Sintaxis
        - Invocación
        - Manejo de errores
        - UsandoAsync/Await
    - Principales diferencias entre Fetch API y Axios
- Introducción a TypeScript
    - Qué es TypeScript
    - Para qué se utiliza TypeScript
    - TypeScript en React.js
    - TypeScript vs. Javascript
    - TypeScript y Webpack
    - Definiendo tipos
    - Tipos por inferencia
    - Componiendo tipos
    - Sistema de tipo estructural
    - Interfaces y Clases
    - Algunos Frameworks que soportan TypeScript (Next.js, Gatsby.js)
- Seguridad en un Aplicativo Front-End

    - Conceptos básicos de seguridad en aplicaciones Web (Clickjacking, Ataque XSS, SQL Injection, Ataque DoS)
    - Recomendaciones de seguridad en una aplicación Web
    - Recomendaciones de seguridad en una aplicación ReactJs
    - Identificando vulnerabilidades en una aplicación ReactJs
    - Consumiendo servicios REST con Api Key y JWT
    - Cómo proteger rutas con React Router DOM
    - Implementando seguridad por Roles en React
    - La seguridad en la autenticación de usuarios
    - Encriptación de datos en el front
- Hooks
    - Qué son los Hooks
    - Un vistazo en detalle a los Hooks
    - Usando el Hook de estado
    - Usando el Hook de efecto
    - Reglas de los Hooks
    - Construyendo Hooks personalizados
- Manejo y Límite de Errores
    - Describe el concepto de hook y sus utilidades de acuerdo al entorno React
    - Identifica los conceptos de Hook de Estado y de Hook de Efecto en un aplicativo React
    - Reconoce los mecanismos para el manejo de errores en un aplicativo React
    - Utiliza hooks en un aplicativo React para resolver un problema
    - Utiliza mecanismos disponibles en el entorno React para el manejo y control de los errores y excepciones

Módulo 6: Desarrollo de aplicaciones Web Progresivas (PWA)

Objetivo específico

Implementar aplicaciones web progresivas utilizando React.

Contenidos

- Introducción a las Aplicaciones PWA
    - Qué es unaPWA(Progressive Web Application)
    - Características de una PWA (Progresiva,responsiva, adaptable, segura, independiente)
    - Beneficios de una PWA
    - Limitaciones de una PWA
    - Diferencias entre una PWA, una aplicación Web tradicional y una aplicación Nativa
    - Arquitectura y componentes de una PWA (Service Workers, Manifiesto, Shell de la aplicación)
    - Frameworks que soportan el desarrollo de PWA’s
- El Manifiesto
    - Qué es el Manifiesto
    - Para qué se usa el Manifiesto
    - Estructura de un archivo de Manifiesto
- El Service Worker
    - Qué es el ServiceWorker

    - Para qué se usa el Service Worker
    - Ventajas de usar un Service Worker
    - Descripción general de un Service Worker (API asíncrona, API basada en eventos, precaching, aislamiento del hilo principal)
    - Ciclo de vida de unService Worker
    - Cómo interactúa el Service Worker con el caché y el acceso a la red
    - Cómo configurar Service Worker para ReactJs
    - Funcionamiento de una PWAconHTTPS
    - Estrategias de almacenamiento en caché de Service Worker
        - Stale-While-Revalidate
        - Cache-first
        - NetworkFirst
        - CacheOnly
        - NetworkOnly
        - CacheAndNetwork
- Almacenamiento en una PWA
    - Cómo se administra el Almacenamiento Web en una PWA
    - El uso de LocalStorage y SessionStorage
    - Acerca de WebAssembly y el código precompilado
    - Bibliotecas de bases de datos con soporte para PWA(IndexedDB, PouchDB, RxDB, GunDB)
- Creando una PWA
    - Creando un proyecto PWAenReactJs
    - Registrando un Service Worker
    - Personalizando un Service Worker
    - Navegando a través de una PWA
    - Implementando las distintas estrategias de almacenamiento en caché de Service Worker
    - Accediendo a periféricos del Sistema Operativo
    - Despliegue de una PWA
- Explorando el Estado de una PWA con Lighthouse
    - Instalando la extensión lighthouse en el navegador
    - Algunos aspectos relevantes de lighthouse (Rápido, instalable, optimizado para PWA)
    - Ejecutando lighthouse (Análisis de carga de página)
    - Revisando el informe de lighthouse

Módulo 7: Fundamentos de desarrollo Agile

Objetivo específico

Adquirir los conocimientos fundamentales de las metodologías ágiles y el rol del desarrollador en un equipo Scrum.

Contenidos

- Metodologías Ágiles
    - Orígenes de las metodologías ágiles
    - Diferencias con modelos tradicionales de desarrollo (Waterfall, RUP, Modelo PMBOK)
    - Diferencias entre “Agile”, “Agilidad” y “Agilismo”

    - Manifiesto Ágil
    - Valores y Principios del Manifiesto Ágil
    - Abanico de Metodologías Ágiles
        - DSDM-Atern
        - LeanSoftwareDevelopment
        - ExtremeProgramming (XP)
        - Kanban

- Fundamentos de Scrum

    - Qué es Scrum
    - Principios y Valores de Scrum
    - Roles en Scrum
        - ScrumMaster
        - ScrumDevelopers
        - ProductOwner
    - Prácticas en Scrum
        - SprintPlanning
        - DailyScrum
        - SprintReview
        - SprintRetrospective
    - Artefactos en Scrum
        - ProductBacklog
        - SprintBacklog
        - Incremento

- Visión del Producto en Scrum

    - Producto y suVisión
    - CómoEstimar
    - Definición de Alcance
    - ReleaseManagement
    - Scrum como modelo iterativo y evolutivo
    - MínimoProductoViable (MVP)

- Taller de Historias de Usuario

    - Qué es una Historia deUsuario
    - Utilidad de las Historias de Usuario
    - Escribiendo Historias de Usuario
    - ModeloINVEST
    - ModeloSMART

- Taller de Estimación

    - Estimación en Scrum
    - Desechando la estimación en Horas Hombre (HH)
    - Modelos de Estimación Ágiles
        - Puntos de Historia

        - Tallas
        - Matriz de Incertidumbre
    - MovimientoNoEstimates

- Priorización

    - En qué consiste la práctica de priorización
    - Diferenciar lo Urgente de lo Importante
    - Modelos de Priorización
        - Visión de Negocio
        - Triage
        - Moscow
    - Cómo Negociar prioridades

- Refinamiento del Backlog

    - Para qué sirve el Refinamiento del Backlog
    - Importancia de Refinar Historias
    - Cómo llevar sesiones de Refinamiento
    - Malas prácticas en el Refinamiento

Módulo 8: Fundamentos de integración continua

Objetivo específico

Conocer los procesos, técnicas y herramientas para la implementación de un circuito de integración continua.

Contenidos

- Fundamentos de DevOps

    - Qué es DevOps y cuál es su propósito
    - Origen y evolución de DevOps
    - Cultura y principios DevOps
    - Beneficios de DevOps
    - ModeloCAMS(Culture, Automation, Measurement, Sharing)
    - DevOps y DevSecOps
    - Ciclo de vida DevOps

- Integración Continua / Entrega Continua

    - Qué es Integración Continua
    - Despliegue Continuo vs Entrega Continua
    - Flujo del proceso de Integración Continua
        - Control de versiones
        - Compilación
        - Testing
    - Sistemas de integración continua
        - Alternativas (Jenkins, Circle CI, GitLab CI, Bamboo)

- Contenedores de Aplicaciones
    - Qué es un contenedor de aplicaciones
    - Diferencias entre un contenedor y una máquina virtual
    - El contenedor Docker - Conceptos básicos de Docker
        - Images
        - DockerFile
        - Containers
        - Volumes
    - Beneficios de utilizar un contenedor Docker
    - Rol del contenedor Docker dentro del ciclo DevOps
    - Implementación de Docker
        - Instalación y configuración
        - Comandos administrativos básicos
        - Utilización de imágenes
        - Monitoreo de eventos y logs asociados a contenedores
- Las Pruebas en un Entorno de Integración Continua
    - Testing en un entorno ágil
        - Qué es Agile Testing
        - Principios de Agile Testing
        - Prácticas relacionadas con Agile Testing
        - Rol del tester en un marco ágil
        - Qué es TDD
    - Objetivo de las pruebas
    - Tipos de prueba
        - Pruebas unitarias
        - Pruebas de integración
        - Pruebas de sistema
            - Funcionales
            - Rendimiento
        - Pruebas de aceptación
        - Pruebas de humo
    - Automatización de las pruebas
        - Objetivo de la automatización
        - Herramientas para la automatización de las pruebas
        - Incorporación de la automatización de pruebas al pipeline de integración continua
- Implementación de un Pipeline CI
    - Qué es Jenkins y para qué sirve
    - RoldeJenkins dentro de la Integración Continua
    - Instalación y configuración de Jenkins
        - Configuración de plugins, usuarios, roles y permisos
    - Definición de pipelines de integración con código: JenkinsFile
    - Utilización de Jenkins en el ciclo CI/CD
        - Creación y configuración de un job en Jenkins

            - Creación y configuración de pipeline de Jenkins mediante código JenkinsFile
            - Creación y configuración de despliegue sobre el ambiente de pruebas

- Infraestructura y Operaciones

    - Qué se entiende por infraestructura y por plataforma
    - Infraestructura evolutiva
    - Diferencias entre on-premise y cloud
    - Importancia de la infraestructura en el ciclo DevOps
    - Infraestructura como código
    - El modelo de infraestructura cloud
    - Modelos de servicio en la nube (IaaS, PaaS, SaaS)
    - Operaciones y escalamiento
        - Problemas de la operación
        - Orquestadores de contenedores
        - Qué es Kubernetes
    - Monitoreo
        - Qué es el monitoreo continuo
        - Importancia del monitoreo continuo durante el ciclo DevOps
        - Importancia del manejo de logs y eventos
        - Stacks de monitoreo (ELK, GFG)

- Sistemas de Control de Versiones (SCV)

    - Qué es un SCV
    - Problema que resuelve un SCV
    - Principales conceptos de un SCV (Repositorio, Diff, Commit, Branch, Merge, Clone, Fork)
    - Tipos de SCV y alternativas
        - Centralizados (SVN, CVS)
        - Distribuidos (Git, Mercurial)
    - Git como sistema de control de versiones
        - Instalación, configuración y comandos básicos
        - Utilización de Git en repositorios locales
        - Commits y restauración de archivos
        - Ignorando archivos
        - Ramas,uniones, conflictos y tags
        - Stash y rebase
    - Centralización de repositorios
        - Servicios de centralización de repositorios (Gitlab, Github, Bitbucket)
        - Repositorios remotos, push y pull
        - Fetch vs Pull
        - Clone y Fork de un repositorio
        - Flujos de trabajo

Módulo 9: Desarrollo de portafolio de un producto digital

Objetivo específico

Preparar y presentar el portafolio de trabajos desarrollados a lo largo del curso

Contenidos

- El Portafolio de Productos
    - Qué es un portafolio de productos
    - Importancia de contar con un portafolio
    - Buenas prácticas para la creación de un portafolio de productos
    - Herramientas que se pueden utilizar para la creación del portafolio de productos:
        - GitHub
        - Hosting
        - Página web personal
        - YouTube
        - Otros
- Elaborar un Producto Tecnológico
    - Distingue conceptos y buenas prácticas para el diseño de un producto digital que resuelve un problema
    - Utiliza técnicas y herramientas de la disciplina para la implementación de un producto digital
    - Ajustes finales y cierre de entregable
    - Implementa un producto funcional que resuelve un problema real utilizando las buenas prácticas de la disciplina
- Finalización del Proyecto
    - Revisión del producto construido a lo largo del curso
    - Depuración y mejora del producto
    - Feedback y retroalimentación
    - Ajustes finales y cierre de entregable
- Implementación del Portafolio de un Producto Digital
    - Distingue las competencias que serán adquiridas a lo largo de cada módulo de la currícula
    - Reconoce la naturaleza del trabajo que será realizado junto con las herramientas que serán utilizadas a lo largo de cada módulo
    - Reconoce la importancia de un portafolio de producto así como sus características para la formación de una identidad profesional
    - Identifica los productos que serán obtenidos en cada módulo así como su contribución al portafolio de producto
- Utilización de GitHub para Crear un Portafolio
    - Qué es GitHub
    - Características y operaciones básicas de GitHub
    - Buenas prácticas para tener un portafolio atractivo en GitHub
    - Buenas prácticas para la página principal de un repositorio
    - Buenas prácticas para la página de perfil
- Utilización de Behance para Crear un Portafolio
    - Qué es Behance
    - Características y operaciones básicas de Behance
    - Buenas prácticas para tener un portafolio atractivo en Behance
    - Behance dedicado y exclusivo a proyectos de UX
- Alojamiento de tu Producto en un Servidor
    - Qué es un Hosting

- Servicios gratuitos de hosting
- Servicios cloud gratuitos
- Cómo alojar un proyecto UX/UI
- Cómo alojar un proyecto Front-End
- Cómo alojar un proyecto Fullstack Java
- Cómo alojar un proyecto Fullstack Javascript
- Cómo alojar un proyecto Fullstack Python
- Cómo alojar un proyecto Android.