CORFO
BecasCapital Humano

Adalid
aspasia
LA FORMACIÓN DE TU FUTURO

MÓDULO 1

ORIENTACIÓN DEL PERFIL Y METODOLOGÍA DEL CURSO

Manual del Módulo 1: Orientación al Perfil y Metodología del Curso

Introducción

El Módulo 1 es fundamental para establecer las bases de todo el curso. En esta sección, los participantes serán introducidos a la industria de TI, comprendiendo sus características, la variedad de perfiles profesionales que existen, y las competencias técnicas y personales más valoradas. Se discutirá en profundidad qué significa ser un desarrollador front-end, cuáles son las expectativas laborales actuales, y cómo se proyecta el crecimiento en esta área. Además, se explorará cómo está estructurado el curso, incluyendo los módulos, herramientas, metodologías de enseñanza-aprendizaje, y la importancia de construir un portafolio profesional que refleje el desarrollo de habilidades adquiridas.

1. La Industria TI

1.1. Características de la Industria

La industria de las Tecnologías de la Información (TI) es una de las más dinámicas y de rápido crecimiento en la economía global. A continuación, se describen sus principales características:

- **Innovación Continua**: La TI está en constante evolución. Cada año, nuevas tecnologías, lenguajes de programación, frameworks y herramientas surgen, obligando a los profesionales a mantenerse actualizados constantemente. Por ejemplo, la transición de modelos monolíticos a arquitecturas basadas en microservicios es una tendencia reciente que ha transformado la manera en que se desarrollan y mantienen las aplicaciones.

- **Globalización**: La TI es una industria global, lo que significa que el trabajo puede realizarse desde cualquier lugar del mundo. Empresas como Google, Amazon y Microsoft operan en múltiples países, y muchos proyectos involucran equipos distribuidos. La capacidad para colaborar a través de fronteras geográficas y culturales es fundamental.

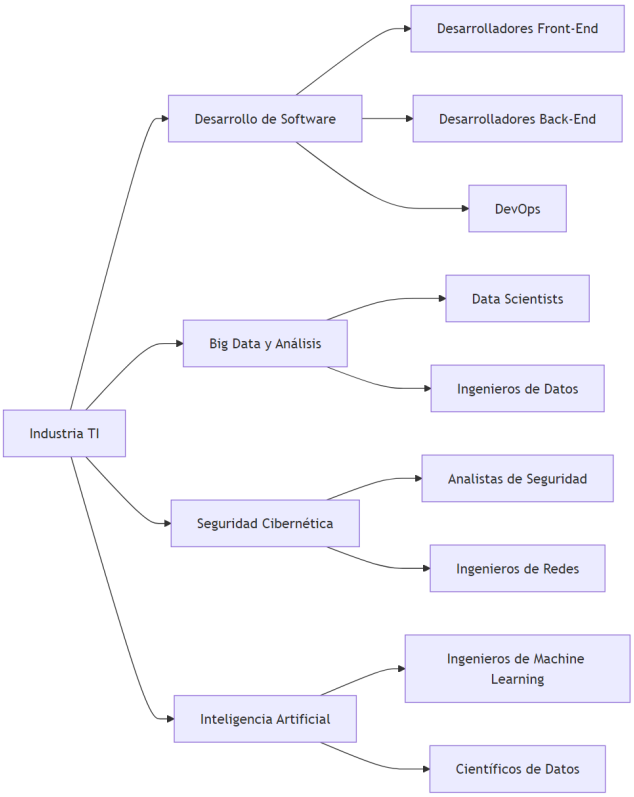
- **Alta Demanda Laboral**: Según datos recientes de LinkedIn y Glassdoor, los desarrolladores de software son uno de los perfiles más buscados en todo el mundo,
- con un déficit significativo de talento en muchas áreas clave. Este fenómeno es especialmente evidente en campos como la inteligencia artificial, la ciberseguridad y el desarrollo de aplicaciones móviles.

- **Agilidad y Flexibilidad**: Las metodologías ágiles, como Scrum y Kanban, dominan el desarrollo de software moderno, reemplazando modelos más rígidos como el Waterfall. La adopción de estas metodologías permite a las empresas responder rápidamente a los cambios en el mercado y a las necesidades de los clientes.

Ejemplo Real:

Un ejemplo claro de la rapidez con la que evoluciona la industria TI es la adopción de **frameworks de JavaScript**. Hace una década, frameworks como jQuery dominaban el desarrollo front-end. Hoy en día, React, Angular y Vue.js han tomado su lugar, ofreciendo soluciones más robustas y escalables.

Diagrama de Ecosistema de la Industria TI:



- **Artículos**: "State of the Developer Nation" de SlashData
- **Libros**: "The Innovator's Dilemma" de Clayton Christensen
- **Videos**: The Rise of DevOps

1.2. Perfiles más Comunes en la Industria TI

La diversidad de perfiles en la industria TI es extensa, reflejando las diferentes áreas de especialización y responsabilidad que existen. Aquí se detallan algunos de los perfiles más destacados:

- **Desarrollador Front-End**: Este perfil se enfoca en la creación de interfaces de usuario. Utiliza tecnologías como HTML, CSS, y JavaScript para construir sitios web y aplicaciones que son visualmente atractivas y fáciles de usar. También deben asegurar que las aplicaciones sean responsivas y accesibles desde cualquier dispositivo.

    - **Ejemplo Práctico**: Un desarrollador front-end puede estar encargado de construir un sitio web para una tienda en línea. Esto incluye diseñar la interfaz, asegurarse de que las imágenes y los textos se visualicen correctamente en dispositivos móviles, y optimizar la velocidad de carga de la página.

- **Desarrollador Back-End**: Responsable del lado del servidor de una aplicación, este perfil se centra en la lógica del negocio, bases de datos y APIs. Utilizan lenguajes como Java, Python, o Node.js para manejar las solicitudes del cliente y devolver respuestas adecuadas.

    - **Ejemplo Práctico**: Un desarrollador back-end podría desarrollar la funcionalidad que permite a los usuarios de una aplicación de comercio electrónico añadir productos a un carrito de compras, calcular los costos de envío y procesar pagos.

- **DevOps Engineer**: Este perfil es crucial en la integración continua y la entrega continua (CI/CD). Los ingenieros DevOps trabajan para automatizar los procesos de desarrollo y despliegue, mejorando la eficiencia y reduciendo el riesgo de errores.

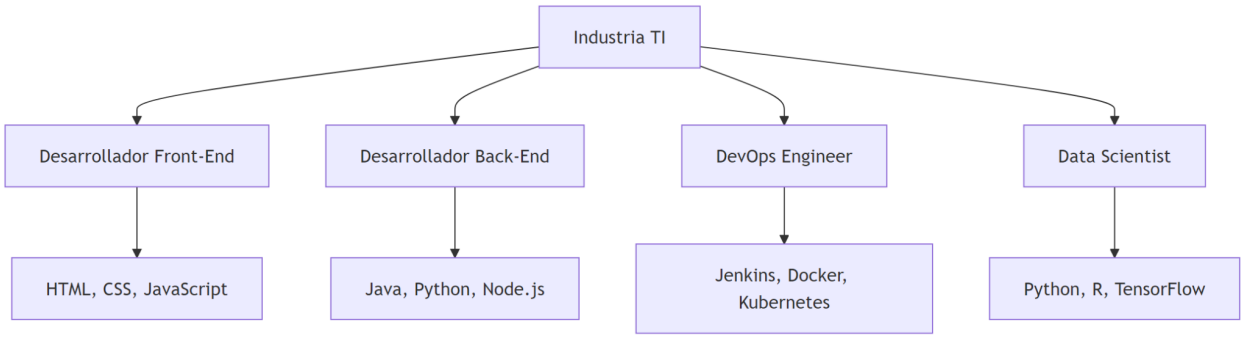
    - **Ejemplo Práctico**: Un ingeniero DevOps puede configurar un pipeline de CI/CD usando herramientas como Jenkins y Docker para automatizar la construcción, prueba y despliegue de una aplicación.

- **Data Scientist**: Analiza grandes volúmenes de datos para descubrir patrones y obtener insights que pueden ser utilizados para tomar decisiones estratégicas. Utilizan

-
- herramientas como Python, R, y SQL, así como bibliotecas de aprendizaje automático como TensorFlow y Scikit-learn.

    - **Ejemplo Práctico**: Un científico de datos podría analizar datos de clientes para identificar comportamientos de compra y recomendar productos específicos a cada usuario basado en su historial de compras.

Diagrama de Perfiles Comunes:



1.3. Competencias Técnicas Valoradas por la Industria TI

Para destacar en la industria TI, no solo es importante poseer un conjunto de habilidades técnicas, sino también saber cómo aplicarlas en contextos prácticos. Las competencias técnicas más valoradas incluyen:

- **Programación**: La habilidad para escribir código en uno o más lenguajes de programación es fundamental. Esto incluye comprender las mejores prácticas de codificación, cómo optimizar el rendimiento del código, y cómo escribir código mantenible y escalable.

- **Lenguajes Clave**:
    - **JavaScript**: Esencial para el desarrollo web front-end y cada vez más utilizado en el back-end con Node.js.
    - **Python**: Popular en ciencia de datos, inteligencia artificial y desarrollo web.

    - **Java**: Predominante en aplicaciones empresariales y Android.

- **Manejo de Frameworks**: Los frameworks son herramientas que permiten a los desarrolladores trabajar de manera más eficiente, ofreciendo estructuras predefinidas para construir aplicaciones. Conocer uno o varios frameworks es crucial.

- **Frameworks Clave**:
    - **React**: Utilizado para construir interfaces de usuario dinámicas.
    - **Angular**: Un framework completo para desarrollar aplicaciones de una sola página (SPA).
    - **Django**: Un framework web de alto nivel en Python que permite un desarrollo rápido y limpio.

- **Control de versiones**: Herramientas como Git son indispensables para gestionar el código de manera colaborativa. Saber cómo realizar commits, crear y manejar ramas, resolver conflictos y realizar merges son habilidades básicas para cualquier desarrollador.

    - **Ejemplo Práctico**: Crear un repositorio en GitHub, realizar commits de código, y manejar ramas para colaborar en un proyecto grupal.

- **Metodologías Ágiles**: Conocer y aplicar metodologías como Scrum o Kanban ayuda a los desarrolladores a trabajar de manera eficiente en equipos, gestionar mejor el tiempo, y entregar valor de manera continua.

- **Scrum**: Implica trabajar en sprints cortos y revisar el progreso en reuniones diarias (daily stand-ups).
- **Kanban**: Enfocado en la gestión visual del trabajo en un tablero, permitiendo a los equipos ver el flujo de trabajo y detectar cuellos de botella.

1.4. Habilidades Personales Valoradas por la Industria TI

Más allá de las habilidades técnicas, la industria TI valora una serie de habilidades personales que son esenciales para el éxito en el entorno laboral:

- **Trabajo en Equipo**: La capacidad para colaborar efectivamente con otros es fundamental. Esto incluye no solo trabajar bien con otros desarrolladores, sino también con diseñadores, gerentes de proyecto, y clientes.

    - **Ejemplo Práctico**: Participar en una sesión de pair programming donde dos desarrolladores trabajan juntos en el mismo código, compartiendo ideas y aprendiendo uno del otro.

- **Adaptabilidad**: Dado que la tecnología cambia rápidamente, los desarrolladores deben ser capaces de aprender nuevas herramientas y tecnologías rápidamente y adaptarse a cambios en los proyectos.

    - **Ejemplo Real**: Un desarrollador que comenzó su carrera trabajando con AngularJS, y luego tuvo que adaptarse a Angular (la versión moderna) cuando el framework fue reescrito desde cero.

- **Resolución de Problemas**: Los desarrolladores se enfrentan a problemas constantemente, desde errores de sintaxis hasta problemas de rendimiento y seguridad. La capacidad para diagnosticar y resolver estos problemas de manera eficiente es esencial.

    - **Ejemplo Práctico**: Resolver un conjunto de bugs en un proyecto de código abierto en GitHub, identificando las causas raíz y proponiendo soluciones.

- **Comunicación**: La habilidad para comunicar ideas técnicas de manera clara y concisa es vital, ya sea en la documentación del código, durante una presentación de progreso (demo), o en la discusión de requisitos con un cliente.

    - **Ejemplo Práctico**: Preparar una presentación para explicar el diseño de una aplicación a un equipo no técnico, utilizando diagramas y terminología accesible.

1.5. Metodologías y Forma de Trabajo del Área

Las metodologías de trabajo en la industria TI han evolucionado significativamente. Aquí se exploran las más relevantes:

- **Scrum**: Una metodología ágil que se centra en entregas incrementales de productos a través de ciclos cortos llamados sprints. Cada sprint suele durar entre 1 y 4 semanas y termina con una revisión y retrospectiva.

- **Roles en Scrum**:
    - **Scrum Master**: Facilita el proceso y se asegura de que el equipo siga las prácticas de Scrum.

- **Product Owner**: Define las prioridades del proyecto y gestiona el backlog del producto.
- **Development Team**: Los miembros que ejecutan el trabajo técnico.

- **Ejemplo Práctico**: Realizar un sprint planning, donde se seleccionan las tareas que el equipo abordará en las próximas dos semanas y se estima el esfuerzo requerido para cada una.

- **Kanban**: Una metodología visual que permite a los equipos gestionar el flujo de trabajo de manera eficiente. Cada tarea es representada como una tarjeta en un tablero, y las tarjetas se mueven a través de columnas que representan diferentes estados de la tarea (por ejemplo, "Por hacer", "En progreso", "Hecho").

    - **Ejemplo Real**: Un equipo de TI que utiliza un tablero Kanban para gestionar las solicitudes de soporte técnico, asegurando que las solicitudes más urgentes se abordan primero.

- **Waterfall**: Un enfoque secuencial donde cada fase del proyecto debe completarse antes de que la siguiente comience. Aunque menos popular en la TI moderna, todavía se usa en proyectos con requisitos bien definidos y poca incertidumbre.

- **Diagrama de Waterfall**:

Ejercicio de Simulación:

Simulación de Proyecto Ágil:

- **Objetivo**: Los estudiantes trabajarán en un proyecto pequeño (por ejemplo, una aplicación de lista de tareas) utilizando Scrum. Se formarán equipos, se asignarán roles, y se realizará una sprint planning, daily stand-ups, sprint review, y una retrospectiva.

- **Materiales Necesarios**: Un tablero Kanban digital (como Trello o Jira), una herramienta de comunicación (Slack), y un repositorio en GitHub.

2. El Perfil Profesional Asociado al Curso

2.1. ¿Qué es un Perfil Profesional?

Un perfil profesional es una descripción detallada de las habilidades, competencias y experiencia que se espera de un individuo en un rol específico dentro de una industria. Para los desarrolladores front-end, este perfil define no solo las habilidades técnicas que deben poseer, sino también las competencias transversales y personales que son esenciales para su éxito.

- **Componentes Clave de un Perfil Profesional**:
  - **Competencias Técnicas**: Conocimiento profundo de HTML5, CSS3, JavaScript, y al menos un framework moderno como React o Angular.
  - **Habilidades Personales**: Comunicación efectiva, trabajo en equipo, y capacidad de adaptación.
  - **Experiencia**: Práctica en el desarrollo de aplicaciones web, desde el diseño hasta el despliegue y mantenimiento.
  - **Educación y Certificaciones**: Título universitario en informática o áreas relacionadas, y certificaciones en tecnologías relevantes (por ejemplo, certificación en React por Meta).

Ejercicio:

Construcción de un Perfil Profesional:

- **Actividad**: Los estudiantes desarrollarán su propio perfil profesional basado en un formato proporcionado, destacando sus competencias técnicas, habilidades personales, y experiencias relevantes. Esto les ayudará a identificar áreas de mejora y a definir metas de desarrollo a lo largo del curso.

2.2. Competencias que Posee el Perfil

El desarrollador front-end que completa este curso deberá haber adquirido competencias en las siguientes áreas:

- **HTML5, CSS3, JavaScript**: Comprensión profunda de la estructura y estilo de las páginas web, junto con habilidades de scripting para añadir interactividad.

- **Ejemplo Práctico**: Desarrollar una página web desde cero que incorpore todas las características clave de HTML5 y CSS3, utilizando JavaScript para añadir funcionalidades como formularios interactivos y manejo de eventos.

- **Responsive Design**: Capacidad para crear interfaces que se adapten a múltiples dispositivos, asegurando una experiencia de usuario consistente en cualquier tamaño de pantalla.

- **Ejemplo Práctico**: Crear un diseño web adaptable utilizando media queries y frameworks como Bootstrap o Foundation.

- **Uso de Frameworks Modernos**: Proficiencia en al menos un framework de JavaScript moderno como React, Angular, o Vue.js.

- **Ejemplo Práctico**: Desarrollar una SPA (Single Page Application) utilizando React, implementando componentes reutilizables y gestionando el estado de la aplicación con Redux.

- **Control de versiones**: Dominio de Git para gestionar proyectos de manera colaborativa, incluyendo el uso de ramas, merges, y manejo de conflictos.

    - **Ejemplo Práctico**: Colaborar en un proyecto grupal, utilizando GitHub para gestionar el código y realizar pull requests, revisiones de código, y merges.

2.3. Habilidades que Posee el Perfil

Además de las competencias técnicas, este perfil incluye habilidades que son esenciales en la industria:

- **Pensamiento Crítico**: Capacidad para analizar problemas complejos y encontrar soluciones eficientes.

    - **Ejemplo Práctico**: Resolver un conjunto de problemas de diseño de interfaz donde se deben considerar múltiples factores como la usabilidad, accesibilidad, y rendimiento.

- **Atención al Detalle**: Precisión en la implementación del diseño y el código, asegurando que cada elemento funcione correctamente y se vea bien.

    - **Ejemplo Práctico**: Revisar un diseño existente y encontrar inconsistencias o errores, proponiendo mejoras y aplicándolas.

- **Creatividad**: Innovación en la creación de interfaces y experiencias de usuario que no solo cumplan con los requisitos, sino que también sorprenden y deleitan a los usuarios.

    - **Ejemplo Práctico**: Diseñar una interfaz existente con un enfoque en mejorar la experiencia del usuario, utilizando herramientas de prototipado como Figma o Sketch.

2.4. Niveles de Experiencia y Seniority del Perfil

Los desarrolladores front-end pueden ser clasificados en diferentes niveles de experiencia, que a su vez determinan sus responsabilidades y expectativas salariales:

- **Junior Developer**:
    - **Experiencia**: 1-2 años.
    - **Competencias**: Dominio básico de HTML, CSS, JavaScript, y uno o dos frameworks.
    - **Responsabilidades**: Tareas técnicas bajo supervisión, colaboración en proyectos pequeños.
    - **Ejemplo Práctico**: Realizar tareas de refactorización de código y mantenimiento en un proyecto existente.

- **Mid-Level Developer**:
    - **Experiencia**: 2-5 años.
    - **Competencias**: Capacidad para trabajar de manera autónoma, contribuir en proyectos complejos, y dominar varias tecnologías front-end.
    - **Responsabilidades**: Desarrollo de nuevas características, optimización de rendimiento, mentoría a desarrolladores junior.
    - **Ejemplo Práctico**: Liderar el desarrollo de una nueva característica en una aplicación existente, desde la planificación hasta la implementación y pruebas.

- **Senior Developer**:
    - **Experiencia**: Más de 5 años.
    - **Competencias**: Profundidad técnica, liderazgo en proyectos, toma de decisiones estratégicas.
    - **Responsabilidades**: Liderar equipos de desarrollo, diseño arquitectónico de aplicaciones, colaboración con otros departamentos (diseño, producto).

- **Ejemplo Práctico**: Diseñar la arquitectura de una aplicación compleja, definiendo la estructura de componentes, la gestión del estado, y la integración con APIs.

2.5. Expectativas Laborales del Mercado Actual para el Perfil

El mercado laboral para los desarrolladores front-end es altamente competitivo y dinámico. Las expectativas incluyen:

- **Dominio de Frameworks Populares**: React, Angular, y Vue.js son altamente demandados, y los empleadores esperan que los desarrolladores no solo conozcan estas herramientas, sino que también puedan utilizarlas para crear aplicaciones escalables y mantenibles.

    - **Ejemplo Práctico**: Desarrollar una pequeña aplicación de gestión de tareas utilizando React, que incluye funcionalidades como creación, edición, y eliminación de tareas, con un enfoque en la escalabilidad y la mantenibilidad del código.

- **Experiencia en Diseño Responsivo**: Las empresas esperan que los desarrolladores puedan crear interfaces que funcionen sin problemas en cualquier dispositivo, desde teléfonos móviles hasta grandes pantallas de escritorio.

    - **Ejemplo Práctico**: Convertir un diseño estático en una página web completamente responsive, optimizada para diferentes tamaños de pantalla y resoluciones.

- **Capacidad de Trabajo en Equipo**: Dado que la mayoría de los desarrolladores trabajan en equipos multidisciplinarios, la habilidad para comunicarse y colaborar efectivamente es crucial.

    - **Ejemplo Práctico**: Participar en un proyecto grupal utilizando herramientas de colaboración como Slack, GitHub, y Trello para gestionar el proyecto y asegurar una comunicación efectiva.

2.6. Proyección Laboral del Perfil

El futuro de un desarrollador front-end es prometedor, con múltiples caminos de especialización y crecimiento:

- **Líder de Equipo**: Con la experiencia, un desarrollador front-end puede avanzar a roles de liderazgo, gestionando pequeños equipos de desarrollo y asegurando la entrega de proyectos dentro del presupuesto y el plazo.
    - **Ejemplo Práctico**: Asumir el rol de líder de equipo en un proyecto simulado, gestionando la asignación de tareas, facilitando reuniones de seguimiento, y asegurando la calidad del código entregado.
- **Arquitecto Front-End**: Especialistas en diseño de arquitecturas de front-end que pueden escalar a grandes aplicaciones, optimizar el rendimiento y garantizar la seguridad.
    - **Ejemplo Práctico**: Diseñar la arquitectura de un sistema complejo, considerando factores como la modularidad, la reutilización de componentes, y la integración con servicios de terceros.
- **Consultor**: Los consultores ofrecen servicios especializados, ayudando a las empresas a mejorar sus interfaces y la experiencia de usuario, optimizar el rendimiento de sus aplicaciones y adoptar las mejores prácticas del sector.
    - **Ejemplo Práctico**: Realizar una auditoría de una aplicación web existente, identificando áreas de mejora y proponiendo soluciones basadas en las mejores prácticas del sector.

3. Currícula del Curso

3.1. Módulos y Competencias a Formar a lo Largo de la Currícula

El curso está estructurado en varios módulos, cada uno enfocado en desarrollar competencias específicas en el ámbito del desarrollo front-end. Cada módulo se ha diseñado para que los estudiantes puedan aplicar inmediatamente los conocimientos adquiridos en ejercicios prácticos y proyectos reales. A continuación, se presenta un resumen detallado de los módulos:

1. **Orientación al perfil y metodología del curso**: Este módulo introduce a los estudiantes al curso, estableciendo expectativas claras sobre el perfil profesional del desarrollador front-end y las competencias que se desarrollarán a lo largo del programa.

2. **Desarrollo de la interfaz de usuario Web**: Los estudiantes aprenderán los fundamentos de HTML5, CSS3, y cómo utilizar estas tecnologías para crear interfaces de usuario visualmente atractivas y funcionales.

3. **Programación avanzada en JavaScript**: Este módulo cubre temas avanzados en JavaScript, incluyendo estructuras de datos, algoritmos, y el paradigma de programación funcional.

4. **Desarrollo de interfaces interactivas con React**: Los estudiantes se especializan en React, aprendiendo a construir aplicaciones interactivas y a gestionar el estado de las aplicaciones de manera eficiente.

5. **Desarrollo de aplicaciones Front-End con React**: Aquí se exploran técnicas avanzadas en React, incluyendo el consumo de APIs, seguridad, y testing.

6. **Desarrollo de aplicaciones Web Progresivas (PWA)**: Los estudiantes aprenderán a crear aplicaciones web que funcionan offline y ofrecen una experiencia de usuario similar a las aplicaciones nativas.
7. **Fundamentos de desarrollo Agile**: Este módulo introduce a los estudiantes a las metodologías ágiles, enseñándoles cómo gestionar proyectos de desarrollo de manera eficiente.

8. **Fundamentos de integración continua**: Los estudiantes aprenderán a automatizar el proceso de desarrollo y despliegue utilizando herramientas de CI/CD como Jenkins y Docker.

9. **Desarrollo de portafolio de un producto digital**: Los estudiantes trabajarán en la creación de un portafolio que refleje las habilidades y competencias adquiridas a lo largo del curso.

Ejercicio de Planeación de Currículo:

Actividad: Los estudiantes revisarán el contenido de cada módulo y desarrollarán un plan personal de estudio que les permita gestionar su tiempo y recursos de manera eficiente. Identificarán las áreas donde necesitan enfocarse más y establecerán metas específicas para cada módulo.

3.2. Herramientas a Utilizar Durante el Curso

Durante el curso, se utilizarán diversas herramientas para facilitar el aprendizaje y el desarrollo técnico. Es crucial que los estudiantes se familiaricen con estas herramientas desde el principio:

- **LMS (Learning Management System)**: Un LMS como Moodle o Canvas se utilizará para la gestión de contenidos y el seguimiento del progreso de los estudiantes. Los estudiantes podrán acceder a materiales de estudio, realizar exámenes y recibir retroalimentación en un solo lugar.
    - **Ejemplo Práctico**: Completar un tutorial de familiarización con el LMS, incluyendo la entrega de una tarea de prueba, participación en un foro de discusión y la revisión de retroalimentación.
- **GitHub**: GitHub será la plataforma principal para la gestión de código, colaboración en proyectos, y la construcción de un portafolio. Los estudiantes aprenderán a utilizar GitHub para gestionar versiones, colaborar con otros desarrolladores, y presentar sus proyectos a potenciales empleadores.
    - **Ejemplo Práctico**: Crear un repositorio en GitHub, realizar commits iniciales de código, colaborar en un proyecto grupal y utilizar GitHub Pages para publicar un proyecto web.
- **Slack/Trello**: Para la comunicación y la gestión de proyectos, se utilizarán herramientas como Slack para la mensajería instantánea y Trello para la gestión de tareas. Estas herramientas facilitarán la colaboración entre estudiantes y la coordinación de proyectos.
- **Ejemplo Práctico**: Configurar un canal de proyecto en Slack, asignar tareas en Trello, y realizar un sprint plan utilizando estas herramientas.

3.3. Características del Trabajo Técnico a Realizar en Cada Módulo

Cada módulo incluirá actividades prácticas diseñadas para simular situaciones reales en el desarrollo front-end. Los estudiantes trabajarán en proyectos incrementales que les permitirán aplicar lo que han aprendido y desarrollar un portafolio sólido. Las características del trabajo técnico incluirán:

- **Desarrollo Iterativo**: Los estudiantes comenzarán con proyectos simples y añadirán complejidad a medida que avancen en los módulos. Esto les permitirá ver el progreso tangible en sus habilidades y en los productos que desarrollan.
    - **Ejemplo Práctico**: Iniciar un proyecto de sitio web simple y expandirlo a medida que se avanzan en los módulos, integrando nuevas tecnologías y técnicas a medida que se introducen.
- **Revisión por Pares**: Se incentiva la revisión por pares como una herramienta para mejorar la calidad del código y aprender de los demás. Los estudiantes revisarán el código de sus compañeros y proporcionarán retroalimentación constructiva.
    - **Ejemplo Práctico**: Realizar una revisión de código de un compañero en GitHub, utilizando pull requests y comentarios para sugerir mejoras.
- **Documentación**: Se enfatizará la importancia de la documentación en cada proyecto. Los estudiantes serán responsables de documentar su código y las decisiones que tomaron durante el desarrollo.
    - **Ejemplo Práctico**: Crear una documentación detallada para un proyecto, utilizando herramientas como JSDoc o Markdown, y publicarla junto con el código en GitHub.

3.4. Productos Obtenidos en Cada Módulo

Al final de cada módulo, los estudiantes habrán desarrollado componentes clave para su portafolio. Estos productos no sólo demuestran las habilidades adquiridas, sino que también sirven como ejemplos tangibles de su capacidad para aplicar lo aprendido en situaciones prácticas:

- **Prototipos de Interfaces Web**: Desde el diseño de wireframes hasta la implementación de prototipos funcionales.

- **Aplicaciones Interactivas Construidas con React**: Aplicaciones que demuestran la habilidad de los estudiantes para manejar el estado, consumir APIs y crear componentes reutilizables.

- **Implementaciones de Aplicaciones Web Progresivas (PWA)**: Aplicaciones que funcionan offline, son responsivas y están optimizadas para el rendimiento.

4. Portafolio de Producto

4.1. ¿Qué es un Portafolio de Producto?

Un portafolio de producto es una recopilación de trabajos y proyectos que demuestran las habilidades y competencias de un profesional en el desarrollo front-end. Es esencial para mostrar la capacidad del desarrollador a potenciales empleadores o clientes.

- **Componentes Clave**:
  - **Proyectos Individuales**: Cada proyecto debe estar acompañado de una descripción detallada, el código fuente, y una demostración funcional.
  - **Documentación**: Explicar las decisiones técnicas, los desafíos enfrentados, y cómo se resolvieron.
  - **Presentación Visual**: El portafolio en sí debe ser un reflejo del ojo para el diseño que tiene el desarrollador.

Ejemplo Práctico:

Desarrollo de un Portafolio Personal:
- **Actividad**: Los estudiantes crearán un portafolio en línea utilizando GitHub Pages o un servicio similar. Incluirán proyectos destacados, una biografía profesional, y enlaces a sus perfiles en redes profesionales como LinkedIn.

4.2. Importancia de un Portafolio de Producto en la Identidad Profesional

El portafolio es una herramienta crucial para:

- **Demostrar Habilidades Técnicas**: Muestra el conocimiento práctico de tecnologías y metodologías.
- **Destacar Proyectos Relevantes**: Ayuda a los empleadores a evaluar la experiencia y el enfoque del desarrollador.
- **Desarrollar la Marca Personal**: Contribuye a la identidad profesional y aumenta la visibilidad en la industria.

- **Ejemplo de un Portafolio Destacado**:
    - Un desarrollador front-end podría incluir un proyecto de una SPA desarrollada con React que muestra un conjunto de datos interactivos, utilizando gráficos dinámicos y filtros avanzados. La documentación explicaría cómo se gestionó el estado con Redux, cómo se optimizó la carga de datos con técnicas de lazy loading, y cómo se aseguró la accesibilidad.

4.3. Metodología de Enseñanza-Aprendizaje

El curso utiliza una metodología activa que incluye:

- **Aprendizaje Basado en Proyectos (PBL)**: Los estudiantes aplican lo aprendido en proyectos prácticos que simulan situaciones reales de trabajo en la industria.

    - **Ejemplo Práctico**: Desarrollar un proyecto completo de una aplicación desde la fase de conceptualización hasta el despliegue, aplicando todas las técnicas y tecnologías aprendidas en los módulos.

- **Revisión por Pares**: Feedback constante entre compañeros para mejorar los resultados y promover un ambiente de aprendizaje colaborativo.

    - **Ejemplo Práctico**: Participar en sesiones de revisión de código, donde cada estudiante presenta su trabajo, recibe retroalimentación y discute mejoras con sus compañeros.

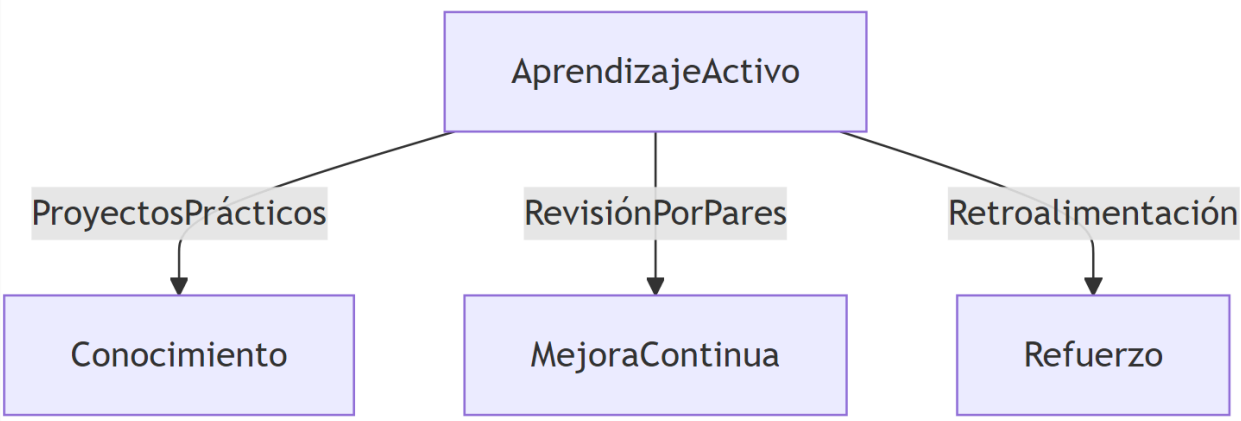
- **Reflexión y Retroalimentación**: Evaluaciones periódicas para reforzar el aprendizaje y asegurar que los estudiantes están progresando hacia sus metas.

- **Ejemplo Práctico**: Al final de cada módulo, los estudiantes escribirán una reflexión sobre lo que aprendieron, los desafíos que enfrentaron y cómo aplicarán esos conocimientos en el futuro.

4.4. ¿Qué es el Aprendizaje Activo?

El aprendizaje activo se centra en la participación directa de los estudiantes en el proceso educativo, fomentando la resolución de problemas, la colaboración y la aplicación práctica de conceptos.

- **Ejemplos de Aprendizaje Activo**:
  - **Hackatones**: Eventos de codificación en los que los estudiantes deben resolver un problema o crear un proyecto en un tiempo limitado.
  - **Talleres Interactivos**: Sesiones en las que los estudiantes trabajan en pequeños grupos para resolver problemas específicos o desarrollar una parte de un proyecto.



4.5. Metodologías Utilizadas a lo Largo del Curso

Las metodologías incluyen:

- **Scrum**: Para gestionar los proyectos de desarrollo, asegurando que los estudiantes aprendan a trabajar en sprints, manejar backlogs y realizar retrospectivas efectivas.

    - **Ejemplo Práctico**: Participar en un ciclo completo de Scrum, desde la planificación del sprint hasta la revisión y retrospectiva, aplicando estas técnicas en un proyecto de desarrollo de software.

- **Code Review**: Revisión constante de código para garantizar la calidad, identificar errores y mejorar las habilidades de codificación de los estudiantes.

    - **Ejemplo Práctico**: Realizar revisiones de código de proyectos grupales, aplicando principios de calidad de código como la limpieza, la modularidad y la eficiencia.

- **Pair Programming**: Programación en pareja para fomentar el trabajo en equipo, compartir conocimientos y mejorar la calidad del código.

    - **Ejemplo Práctico**: Los estudiantes se turnan para trabajar en parejas en tareas de programación, con un estudiante escribiendo el código y el otro revisando en tiempo real, para luego intercambiar roles.

4.6. Rol del Facilitador y del Participante

- **Facilitador**: El facilitador del curso actúa como guía, ofreciendo apoyo, retroalimentación y asegurando que los estudiantes cumplen con los objetivos de aprendizaje.

- **Ejemplo Real**: Durante un sprint, el facilitador podría realizar sesiones de retroalimentación diaria (daily stand-ups) para verificar el progreso de los estudiantes y ofrecer apoyo en las áreas donde se encuentran con dificultades.

- **Participante**: Los estudiantes son responsables de su propio aprendizaje, se espera que sean proactivos, colaboren con sus compañeros y apliquen lo aprendido en proyectos prácticos.

    - **Ejemplo Práctico**: Completar un proyecto de desarrollo de software siguiendo las pautas del facilitador, con una autoevaluación al final del proceso para identificar áreas de mejora.

5. Herramientas a Utilizar a lo Largo del Curso

5.1. Herramientas de Gestión del Proceso de Aprendizaje (LMS)

El LMS será la plataforma central para acceder a los materiales del curso, registrar avances y recibir retroalimentación. Ejemplo: Moodle, Canvas.

- **Características Clave**:
  - **Seguimiento de Progreso**: Los estudiantes podrán ver su progreso a través de un tablero de control, que mostrará las tareas completadas, las calificaciones recibidas, y las áreas donde necesitan mejorar.
  - **Recursos Educativos**: El LMS albergará todos los materiales del curso, incluyendo presentaciones, tutoriales, ejercicios, y enlaces a recursos adicionales.
  - **Evaluaciones y Retroalimentación**: Las evaluaciones serán entregadas a través del LMS, y los estudiantes recibirán retroalimentación detallada de sus instructores.

5.2. Herramientas de Coordinación y Trabajo Colaborativo

Para facilitar la comunicación y el trabajo en equipo, se utilizarán herramientas como:

- **Slack**: Slack será la herramienta principal para la comunicación en tiempo real. Los estudiantes se comunicarán con sus compañeros y facilitadores, organizan reuniones, y compartirán recursos a través de canales específicos.

  - **Ejemplo Práctico**: Configurar un canal de comunicación para un proyecto de equipo, y coordinar el trabajo utilizando integraciones de Slack con herramientas como GitHub y Trello.

- **Trello**: Trello se utilizará para la gestión de tareas, permitiendo a los estudiantes visualizar el flujo de trabajo y asignar responsabilidades dentro de sus equipos.

  - **Ejemplo Práctico**: Crear un tablero de Trello para un proyecto de desarrollo, definir listas y tarjetas para cada tarea, y realizar un seguimiento del progreso durante todo el ciclo del proyecto.

5.3. Herramientas Propias de la Competencia Técnica

Durante el curso, se requerirá el uso de herramientas específicas que son estándar en la industria del desarrollo front-end:

- **Visual Studio Code**: Un editor de código ligero pero potente, que ofrece soporte para múltiples lenguajes de programación y una vasta cantidad de extensiones.
    - **Ejemplo Práctico**: Configurar Visual Studio Code con extensiones útiles para el desarrollo front-end, como ESLint, Prettier, y Live Server, y usarlo para construir un proyecto desde cero.
- **GitHub**: GitHub será esencial para la gestión de código fuente, control de versiones y colaboración en proyectos. Los estudiantes aprenderán a utilizar GitHub para gestionar sus repositorios, colaborar con otros desarrolladores y desplegar sus proyectos.
    - **Ejemplo Práctico**: Crear un repositorio en GitHub, realizar commits de código, colaborar en un proyecto grupal, y desplegar un sitio web utilizando GitHub Pages.
- **Jenkins**: Jenkins se utilizará para la integración continua, permitiendo a los estudiantes automatizar el proceso de construcción, prueba y despliegue de aplicaciones.
    - **Ejemplo Práctico**: Configurar un pipeline de CI/CD en Jenkins para un proyecto de desarrollo, que incluya la ejecución automática de pruebas y el despliegue de la aplicación en un entorno de staging.

6. Habilidades Requeridas a lo Largo del Curso

6.1. Trabajo en Equipo y Autoaprendizaje

El curso enfatiza la importancia del trabajo en equipo y el autoaprendizaje. Los estudiantes deberán ser capaces de colaborar efectivamente con sus compañeros, compartir conocimientos y aprender de manera autónoma.

- **Ejemplo Práctico**: Participar en un proyecto grupal donde cada miembro del equipo tiene una responsabilidad específica, como el desarrollo de una característica, la prueba

de la aplicación, o la documentación del código. El éxito del proyecto dependerá de la colaboración efectiva entre todos los miembros.

6.2. Tolerancia a la Frustración

El desarrollo de software implica superar desafíos y errores. Es crucial que los estudiantes mantengan la resiliencia y la perseverancia.

- **Ejemplo Práctico**: Resolver un conjunto de desafíos de codificación que contienen errores intencionales, obligando a los estudiantes a diagnosticar y corregir los problemas. Este ejercicio simula situaciones reales donde el código no funciona como se esperaba y los desarrolladores deben encontrar soluciones.

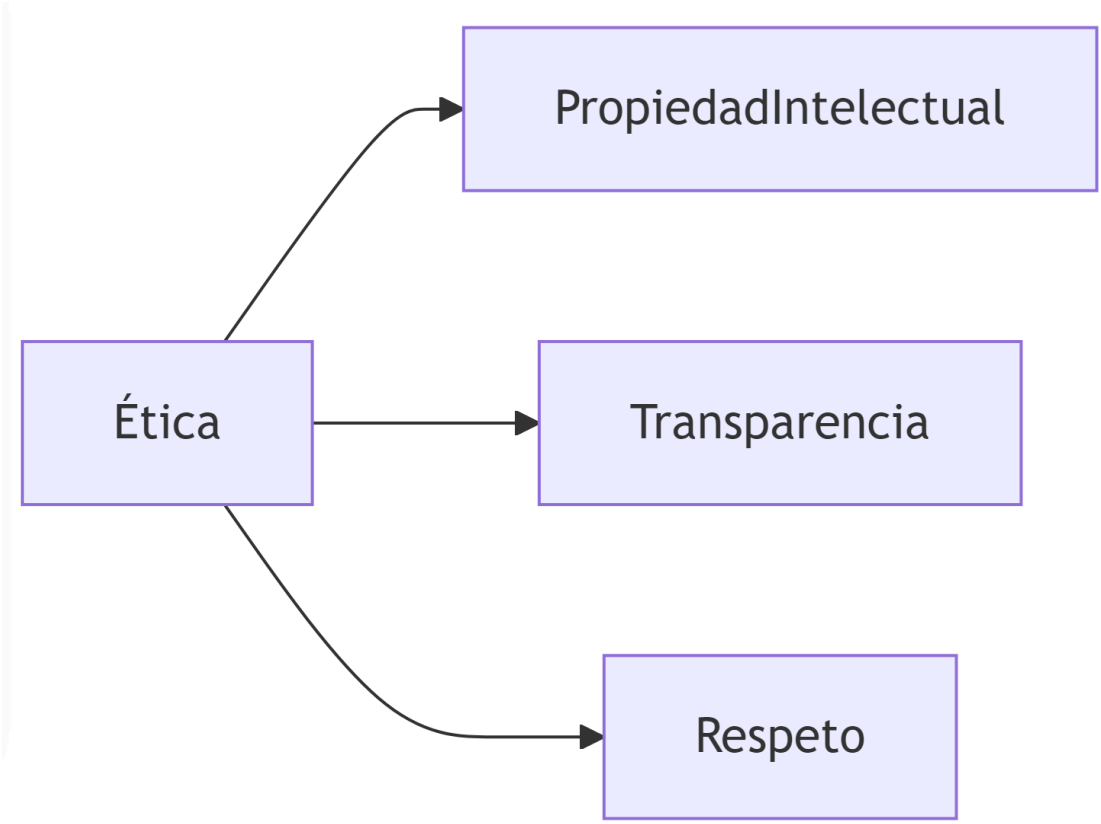
6.3. Comportamiento Ético

El comportamiento ético es fundamental en la industria TI, incluyendo:

- **Respeto por la Propiedad Intelectual**: Uso adecuado de licencias y derechos de autor.

- **Transparencia en el Trabajo**: Honestidad en la comunicación y en la documentación del trabajo.

- **Código de Ética del Curso**:
    - **Integridad**: Los estudiantes deben comprometerse a actuar con integridad en todas las situaciones, evitando el plagio, dando crédito a las fuentes y respetando las contribuciones de otros.
    - **Colaboración Respetuosa**: La colaboración debe ser respetuosa y constructiva, con un enfoque en el crecimiento mutuo y el éxito compartido.



Ejercicio de Reflexión:

Actividad: Los estudiantes escribirán un ensayo sobre un dilema ético que hayan enfrentado o que puedan enfrentar en su carrera como desarrolladores. Discutirán cómo manejaría la situación de acuerdo con los principios de ética profesional.

Material de Referencia

Libros:

- **"Clean Code: A Handbook of Agile Software Craftsmanship"** de Robert C. Martin
  Un libro esencial para cualquier desarrollador que quiera aprender a escribir código limpio, fácil de entender y mantener. **Robert C. Martin** presenta principios clave para mejorar la calidad del código y las buenas prácticas en el desarrollo ágil, cubriendo temas como el manejo de funciones, objetos y clases, entre otros.

- **"The Pragmatic Programmer: Your Journey to Mastery"** de Andrew Hunt y David Thomas
  Este clásico guía a los desarrolladores a través de su evolución profesional, con consejos prácticos sobre cómo pensar, aprender y trabajar de manera eficiente. **Andrew Hunt y David Thomas** cubren una amplia gama de temas, desde técnicas de programación hasta la automatización de tareas y la resolución de problemas.

- **"The Mythical Man-Month: Essays on Software Engineering"** de Frederick P. Brooks Jr.
  Un libro que explora la gestión de proyectos de software, centrándose en los desafíos comunes y cómo abordarlos. **Frederick P. Brooks** analiza el mito del "hombre-mes" y otros problemas clásicos en la ingeniería de software, proporcionando lecciones valiosas para equipos de desarrollo.

Enlaces a Recursos Online:

- **MDN Web Docs**
  Una guía completa y detallada sobre tecnologías web como **HTML**, **CSS**, **JavaScript**, y más. **MDN Web Docs** es una referencia confiable para desarrolladores de todos los niveles que buscan documentación precisa, ejemplos y explicaciones claras de conceptos fundamentales en la programación web.

- **W3Schools**
  Un sitio web de tutoriales interactivos que cubre una amplia gama de tecnologías web. **W3Schools** es ideal para principiantes, ya que ofrece ejemplos claros, pruebas interactivas y una estructura paso a paso para aprender **HTML**, **CSS**, **JavaScript**, y otras herramientas de desarrollo.

- **freeCodeCamp**
  Una plataforma gratuita que ofrece cursos de desarrollo web y programación. **freeCodeCamp** es ideal para aprender mediante la práctica, con desafíos y proyectos que abarcan desde los fundamentos de **HTML** y **CSS** hasta temas avanzados como **APIs** y **diseño responsivo**.

Videos Recomendados:

- **Scrum in 10 Minutes**
  Este video proporciona una introducción rápida y clara a **Scrum**, una de las

metodologías ágiles más populares. Perfecto para entender los conceptos básicos, los roles en Scrum y cómo se estructura un proyecto utilizando sprints.

- **Introduction to Responsive Design**
  Un video que explica los fundamentos del **diseño responsivo**, incluyendo cómo utilizar **media queries**, unidades flexibles, y otros enfoques para crear sitios web que se adapten a diferentes dispositivos y tamaños de pantalla.

- **Git & GitHub Crash Course**
  Un curso rápido y efectivo para aprender los conceptos esenciales de **Git** y **GitHub**, dos herramientas fundamentales en el desarrollo moderno de software. El video cubre desde la instalación de Git hasta la creación y manejo de repositorios en GitHub, perfecto para principiantes.

CORFO
BecasCapital Humano

Adalid
aspasia
LA FORMACIÓN DE TU FUTURO

MÓDULO 1

ORIENTACIÓN DEL PERFIL Y METODOLOGÍA DEL CURSO