CORFO
BecasCapital Humano

dalid
aspasia
LA FORMACIÓN DE TU FUTURO

MÓDULO 9

DESARROLLO DE PORTAFOLIO DE UN PRODUCTO DIGITAL

Manual Módulo 9: Desarrollo de portafolio de un producto digital

Introducción

En este módulo, los estudiantes aprenderán a preparar y presentar un **portafolio de productos digitales**, una herramienta esencial para demostrar las competencias y habilidades adquiridas a lo largo del curso. Un portafolio no solo muestra los proyectos completados, sino que también refleja el proceso de diseño, desarrollo e implementación de soluciones tecnológicas reales.

El objetivo del módulo es guiar a los estudiantes a través de las **buenas prácticas** para la creación de un portafolio profesional que sea atractivo y funcional, utilizando herramientas como **GitHub** y **Behance**, y plataformas de **hosting** para desplegar sus proyectos. Durante el módulo, se proporcionarán técnicas para diseñar y desarrollar productos tecnológicos que resuelvan problemas reales, culminando con la finalización y optimización de un proyecto funcional.

A lo largo del módulo, se revisarán conceptos clave sobre el diseño, la implementación y el ajuste de productos, y se enfatizará la importancia de un portafolio en el desarrollo de una identidad profesional sólida. Al finalizar, los estudiantes estarán capacitados para crear un portafolio que no solo muestre su capacidad técnica, sino que también refleje su crecimiento personal y profesional como desarrolladores.

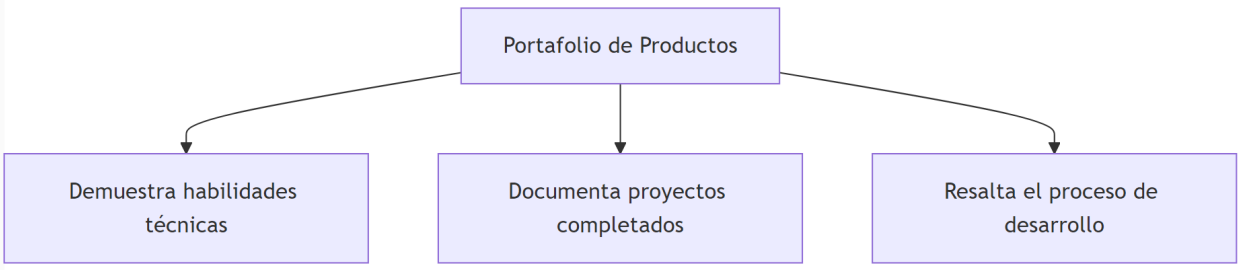
1. El Portafolio de Productos

1.1 Qué es un portafolio de productos

Un **portafolio de productos** es una colección organizada de los proyectos, aplicaciones o productos digitales que una persona o equipo ha desarrollado. Este portafolio permite mostrar de manera visual y estructurada las habilidades técnicas, creativas y de resolución de problemas. Un portafolio no solo documenta los productos finales, sino también el proceso de

desarrollo, incluyendo las decisiones de diseño, las metodologías utilizadas y las herramientas aplicadas.

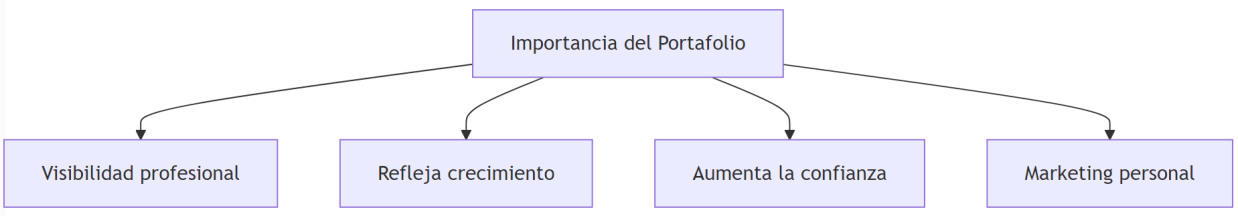
En el contexto de desarrollo de software o diseño digital, un portafolio sirve como una "ventana" a las competencias adquiridas, donde cada proyecto es una prueba tangible del conocimiento en áreas como **front-end**, **back-end**, **UI/UX**, y más.



1.2 Importancia de contar con un portafolio

Tener un **portafolio** es fundamental para cualquier profesional en el ámbito de la tecnología o el diseño digital por varias razones:

1. **Visibilidad profesional**: Permite que posibles empleadores, clientes o colaboradores evalúen tus capacidades de manera directa y tangible.
2. **Refleja crecimiento**: Un portafolio actualizado refleja el progreso y la evolución de tus habilidades.
3. **Aumenta la confianza**: Presentar proyectos exitosos brinda seguridad a clientes o empleadores de que eres capaz de entregar productos de calidad.
4. **Herramienta de marketing personal**: Es una carta de presentación efectiva que habla por ti y que puedes compartir fácilmente en entrevistas, redes profesionales o conferencias.



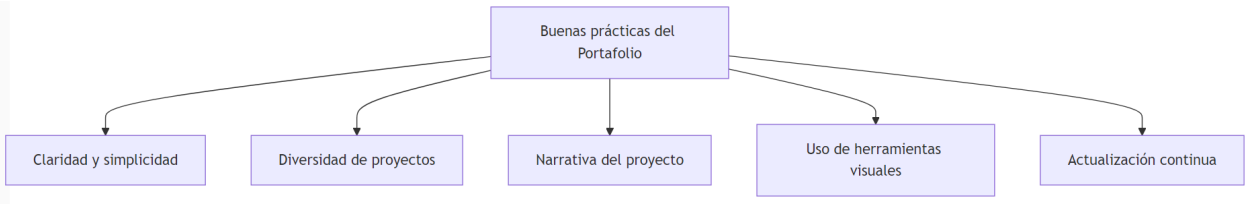
1.3 Buenas prácticas para la creación de un portafolio de productos

Al crear un **portafolio de productos**, es crucial seguir ciertas buenas prácticas para que sea atractivo y profesional:

1. **Claridad y simplicidad**: Muestra tu trabajo de manera clara y organizada. Evita la sobrecarga de información y enfócate en resaltar tus mejores proyectos.
2. **Diversidad de proyectos**: Incluye diferentes tipos de proyectos que muestren una variedad de habilidades (por ejemplo, front-end, back-end, diseño UI/UX).
3. **Narrativa del proyecto**: Acompaña cada proyecto con una breve descripción que explique el objetivo, las herramientas utilizadas y el proceso de desarrollo.
4. **Uso de herramientas visuales**: Utiliza capturas de pantalla, diagramas y videos para mostrar tu trabajo en acción.
5. **Actualización continua**: Mantén tu portafolio actualizado con tus proyectos más recientes para reflejar tus habilidades actuales.

Ejemplo de estructura de un portafolio:

- **Página principal**: Una introducción breve sobre ti y tu carrera.
- **Sección de proyectos**: Cada proyecto con una descripción, tecnologías utilizadas y enlaces a repositorios o demostraciones en vivo.
- **Enlaces a redes profesionales**: Incluye enlaces a tu perfil de **LinkedIn**, **GitHub**, o **Behance**.



1.4 Herramientas que se pueden utilizar para la creación del portafolio de productos

1.4.1 GitHub

GitHub es una de las herramientas más populares para alojar y mostrar proyectos de desarrollo de software. No solo permite compartir el código fuente de los proyectos, sino que también cuenta con funcionalidades para crear una página de portafolio utilizando **GitHub Pages**.

```
# Crear un repositorio en GitHub
git init
git add .
git commit -m "Primer commit"
git push origin main
```

1.4.2 Hosting

Para proyectos que necesitan ser accesibles públicamente, puedes optar por servicios de **hosting** gratuitos o de pago. Servicios como **Netlify**, **Vercel**, o **Heroku** permiten alojar sitios web o aplicaciones de manera gratuita para proyectos pequeños.

```
# Ejemplo de despliegue en Netlify
npm run build
netlify deploy
```

1.4.3 Página web personal

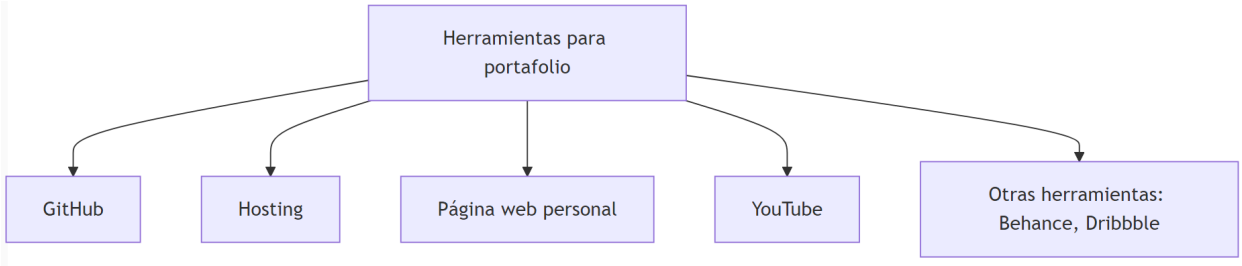
Tener una **página web personal** es una excelente forma de mostrar tu trabajo y tu marca personal. Con plataformas como **WordPress**, **Wix**, o **Squarespace**, puedes crear un sitio atractivo y funcional sin necesidad de conocimientos avanzados en desarrollo web.

1.4.4 YouTube

YouTube puede ser utilizado para mostrar presentaciones de tus proyectos o grabaciones de demostraciones en vivo de tus aplicaciones, lo que ofrece una forma más dinámica de presentar tu trabajo.

1.4.5 Otros

Otras herramientas como **Behance** (para proyectos de diseño), **Dribbble**, y plataformas de blogs pueden complementar tu portafolio, ofreciendo una vista más completa de tus habilidades y logros.



Estas herramientas permiten que los desarrolladores y diseñadores puedan construir portafolios profesionales que sean accesibles y visualmente atractivos, apoyando su crecimiento y exposición en el ámbito digital.

2. Elaborar un Producto Tecnológico

2.1 Distingue conceptos y buenas prácticas para el diseño de un producto digital que resuelve un problema

El diseño de un **producto digital** que resuelve un problema implica la combinación de **investigación**, **planeación**, y **ejecución** de un proyecto de software o diseño. Para asegurar que el producto final sea funcional y útil, es importante seguir ciertas buenas prácticas:

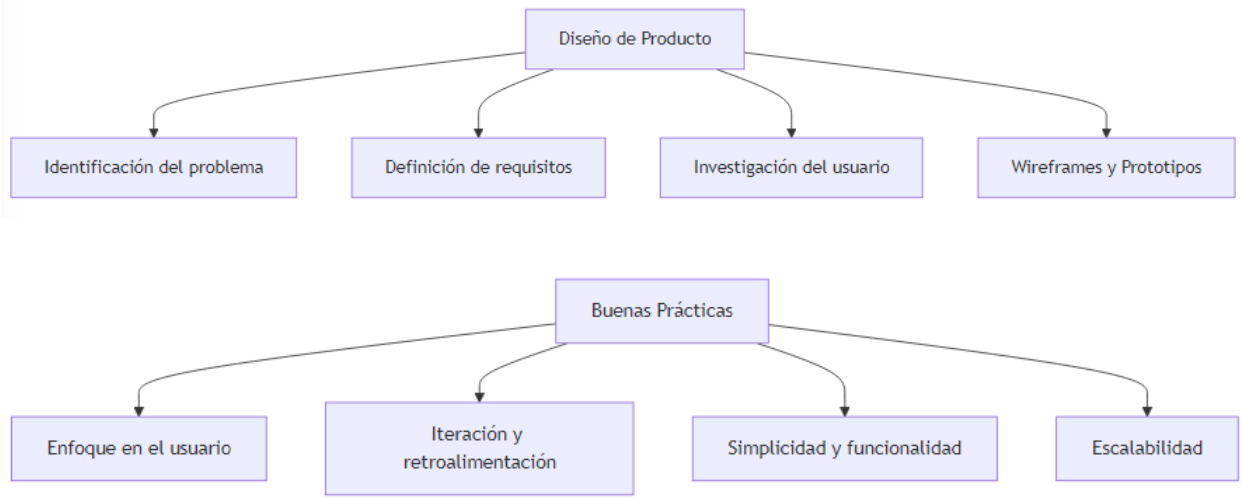
Conceptos clave:

- **Identificación del problema**: Antes de diseñar un producto, es esencial identificar un problema real que los usuarios enfrentan y que el producto puede resolver.
- **Definición de los requisitos**: Establecer las características esenciales que el producto debe tener para resolver el problema de manera eficiente.
- **Investigación del usuario**: Entender las necesidades y comportamientos del usuario final para crear un producto que realmente ofrezca valor.

- **Wireframes y prototipos**: Utilizar herramientas visuales como **wireframes** y **prototipos** para mapear la estructura del producto y validar las ideas antes de empezar el desarrollo.

Buenas prácticas:

1. **Enfoque en el usuario**: Diseñar el producto siempre desde la perspectiva del usuario final, asegurando que sus necesidades y expectativas sean cumplidas.
2. **Iteración y retroalimentación**: Involucrar a los usuarios durante el desarrollo y realizar iteraciones basadas en sus comentarios.
3. **Simplicidad y funcionalidad**: Mantener el diseño limpio y enfocado en resolver el problema sin sobrecargar de características innecesarias.
4. **Escalabilidad**: Asegurarse de que el producto pueda crecer o adaptarse a futuras necesidades sin requerir rediseños completos.



2.2 Utiliza técnicas y herramientas de la disciplina para la implementación de un producto digital

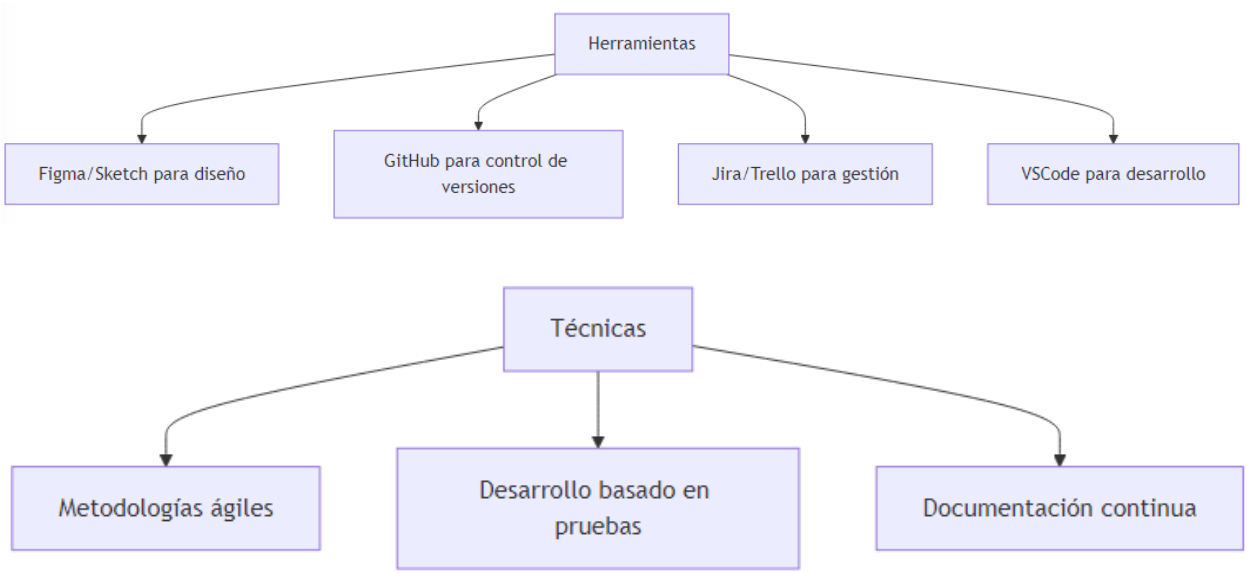
Durante la implementación de un producto digital, es esencial aplicar técnicas y herramientas que optimicen el desarrollo y permitan crear soluciones escalables y mantenibles. Aquí se destacan las principales prácticas y herramientas para un desarrollo exitoso:

Técnicas de desarrollo:

1. **Metodologías ágiles**: Adoptar prácticas como **Scrum** o **Kanban** para gestionar de manera efectiva el proceso de desarrollo, priorizando tareas y entregando valor de forma incremental.
2. **Desarrollo basado en pruebas (TDD)**: Implementar **Test-Driven Development** para asegurar la calidad del código desde el inicio.
3. **Documentación continua**: Mantener la documentación actualizada para facilitar la colaboración y el mantenimiento del proyecto a lo largo del tiempo.

Herramientas recomendadas:

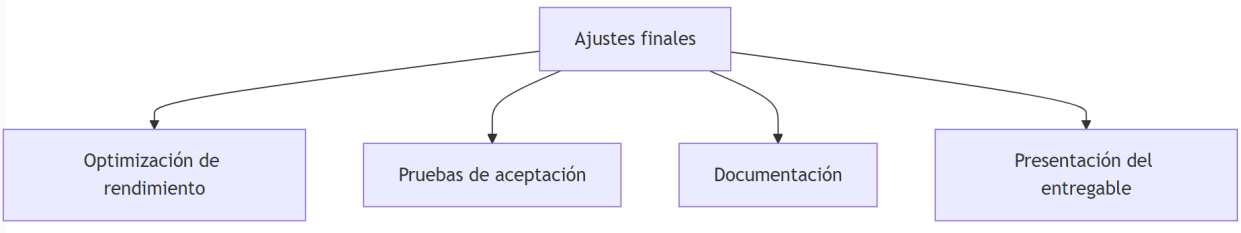
- **Figma o Sketch** para diseño de interfaces y prototipos.
- **GitHub** o **GitLab** para control de versiones y colaboración en el código.
- **Jira** o **Trello** para gestionar tareas y el progreso del proyecto.
- **VSCode** o **WebStorm** como entornos de desarrollo integrados (IDEs) para escribir y organizar el código de manera eficiente.



2.3 Ajustes finales y cierre de entregable

Una vez que el producto digital está funcional, es crucial realizar una serie de **ajustes finales** para asegurar que el resultado sea de alta calidad y cumpla con las expectativas del cliente o del usuario. Estos pasos incluyen:

1. **Optimización**: Revisar el código y la estructura del producto para mejorar su rendimiento y eliminar redundancias.
2. **Pruebas de aceptación**: Validar que el producto cumple con todos los requisitos funcionales y técnicos. Esto puede incluir pruebas con usuarios reales o el uso de herramientas automáticas.
3. **Documentación del producto**: Crear una documentación final que incluya instrucciones de uso, configuración, y mantenimiento.
4. **Presentación del entregable**: Preparar el producto para su despliegue o entrega final al cliente, asegurándose de que se presente de manera profesional y completa.

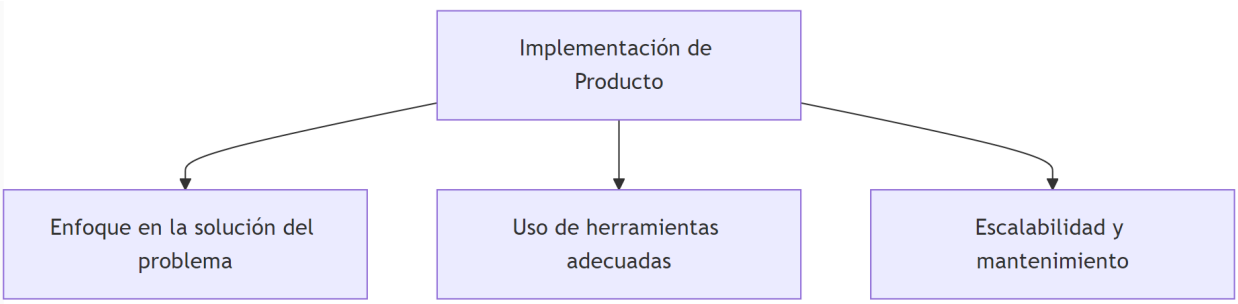


2.4 Implementa un producto funcional que resuelve un problema real utilizando las buenas prácticas de la disciplina

La implementación de un **producto funcional** que resuelva un problema real requiere aplicar todas las fases previamente descritas: desde el diseño hasta la entrega final, garantizando que el producto sea útil y esté alineado con las necesidades del cliente. Las **buenas prácticas** que aseguran el éxito incluyen:

1. **Enfoque en la solución del problema**: Asegurarse de que el producto responde directamente al problema identificado, ofreciendo una solución clara y efectiva.
2. **Uso de herramientas adecuadas**: Utilizar las herramientas más eficientes para la implementación (como frameworks modernos para front-end o back-end).
3. **Escalabilidad y mantenimiento**: Desarrollar el producto pensando en su futuro crecimiento y en la facilidad para corregir errores o agregar nuevas características.

Al seguir estos principios, el producto no solo resolverá el problema inicial, sino que también estará preparado para adaptarse a cambios futuros.



Estos pasos garantizan que el producto final no solo sea funcional y efectivo, sino también robusto y adaptable a largo plazo.

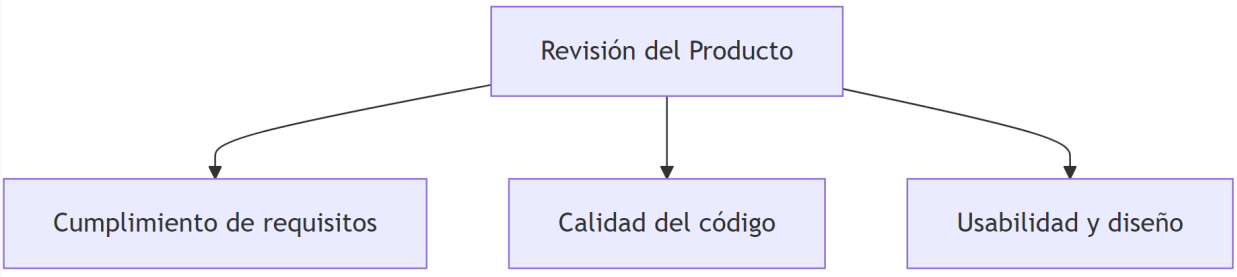
3. Finalización del Proyecto

3.1 Revisión del producto construido a lo largo del curso

Una vez que se ha completado el desarrollo de un producto digital, es crucial realizar una **revisión integral** para asegurarse de que cumple con los objetivos planteados al inicio del proyecto. La revisión debe abarcar tanto los aspectos funcionales como los estéticos, asegurando que el producto sea fácil de usar, eficiente y visualmente atractivo.

Aspectos a revisar:

1. **Cumplimiento de los requisitos**: Verificar que el producto cumple con todos los requisitos funcionales y técnicos definidos en las primeras etapas del proyecto.
2. **Calidad del código**: Revisar la estructura y legibilidad del código, asegurando que siga las buenas prácticas de desarrollo, como modularidad y uso adecuado de variables.
3. **Usabilidad y diseño**: Asegurarse de que la experiencia del usuario sea óptima, considerando la accesibilidad y facilidad de navegación en la interfaz.

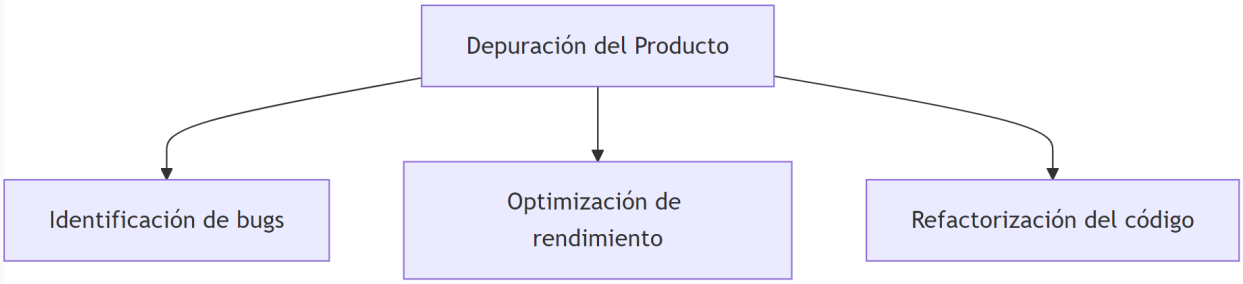


3.2 Depuración y mejora del producto

Después de la revisión inicial, es necesario realizar una **depuración** para corregir errores o inconsistencias que se hayan identificado. Esto incluye la **optimización del rendimiento**, la eliminación de bugs y la implementación de mejoras en la funcionalidad o la interfaz.

Pasos para la depuración:

1. **Identificación de bugs**: Realizar pruebas exhaustivas para encontrar errores que afecten la funcionalidad o la experiencia del usuario.
2. **Optimización de rendimiento**: Mejorar aspectos como tiempos de carga, eficiencia del procesamiento de datos y gestión de recursos.
3. **Refactorización del código**: Revisar el código para hacerlo más legible y mantenible sin alterar su comportamiento.

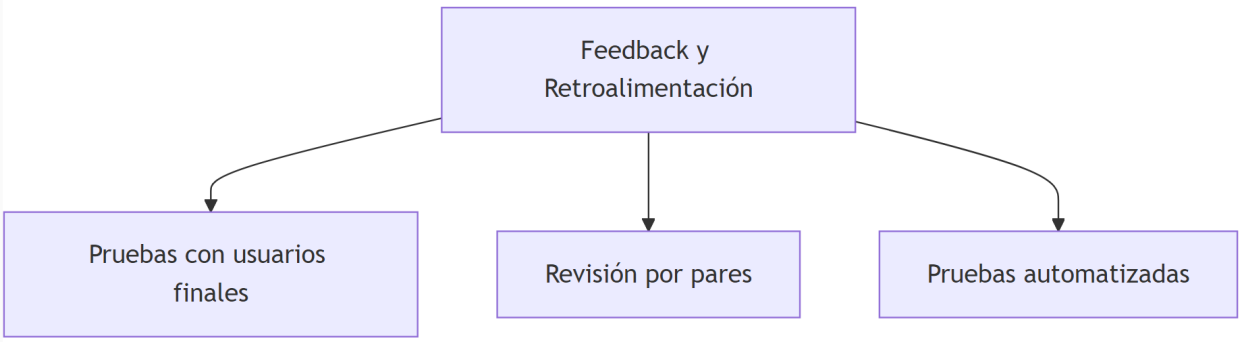


3.3 Feedback y retroalimentación

El **feedback** es una parte esencial en el proceso de finalización de un proyecto. Permite identificar mejoras y optimizaciones basadas en la experiencia de los usuarios o en las observaciones de colegas y expertos. Este proceso es iterativo, lo que significa que el producto debe ajustarse constantemente en función de la retroalimentación recibida.

Fuentes de feedback:

1. **Pruebas con usuarios finales**: Obtener retroalimentación directa de los usuarios que utilizarán el producto para evaluar su usabilidad, funcionalidad y rendimiento.
2. **Revisión por pares**: Pedir a colegas o desarrolladores que revisen el código y el diseño para identificar áreas de mejora.
3. **Pruebas automatizadas**: Usar herramientas de testing para evaluar el comportamiento del producto en diferentes escenarios.

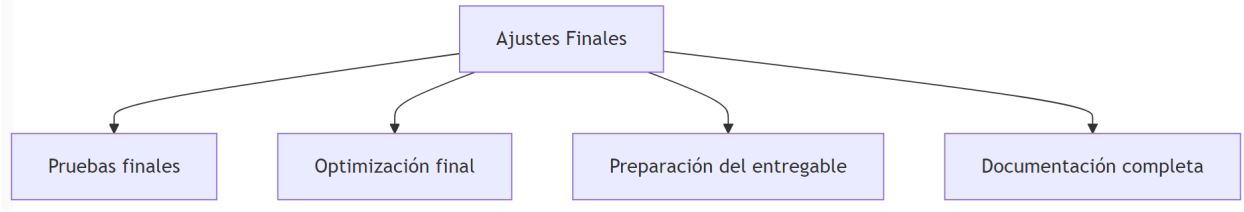


3.4 Ajustes finales y cierre de entregable

Una vez implementadas las mejoras y soluciones basadas en la retroalimentación, el producto debe pasar por una serie de **ajustes finales** antes de su entrega. Estos ajustes incluyen pruebas de última etapa, optimización de detalles y la preparación del producto para su presentación o despliegue final.

Pasos para el cierre del entregable:

1. **Pruebas finales**: Realizar pruebas exhaustivas para asegurarse de que no haya errores críticos o problemas de usabilidad antes del despliegue.
2. **Optimización final**: Asegurar que todos los componentes del producto estén optimizados en términos de rendimiento y recursos.
3. **Preparación del entregable**: Asegurarse de que el producto esté listo para ser entregado, ya sea en formato digital (código o archivos) o en su entorno de producción.
4. **Documentación**: Entregar la documentación completa del producto, que incluya instrucciones para su uso, mantenimiento y futuras actualizaciones.



Este proceso garantiza que el producto esté completamente funcional, optimizado y preparado para ser entregado o presentado, cumpliendo con las expectativas del cliente o de los usuarios finales.

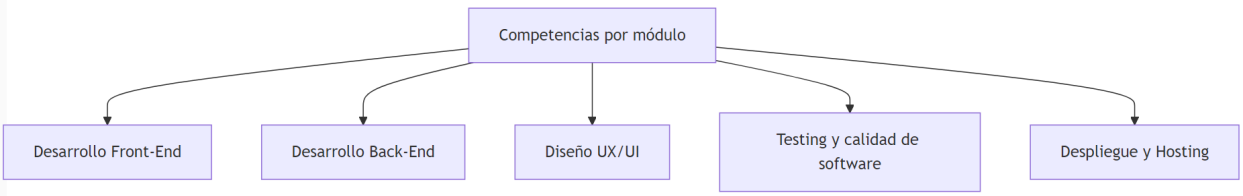
4. Implementación del Portafolio de un Producto Digital

4.1 Distingue las competencias que serán adquiridas a lo largo de cada módulo de la currícula

A lo largo de un curso de desarrollo de productos digitales, los estudiantes adquieren una serie de **competencias técnicas y prácticas** en cada módulo. Estas competencias son las que definen las habilidades profesionales que se plasmarán en el portafolio, y es importante reconocerlas para poder destacar cada proyecto y aprendizaje de manera clara.

Competencias clave por módulo:

- **Desarrollo Front-End**: Habilidad para construir interfaces de usuario interactivas utilizando tecnologías como HTML, CSS y JavaScript.
- **Desarrollo Back-End**: Conocimiento en la creación de API y el manejo de bases de datos para aplicaciones web.
- **Diseño UX/UI**: Capacidad para diseñar interfaces amigables y centradas en la experiencia del usuario.
- **Testing y calidad de software**: Implementación de pruebas unitarias y automatización para garantizar la calidad del producto.
- **Despliegue y Hosting**: Habilidades para poner en producción aplicaciones web, configurando servidores y entornos de hosting.



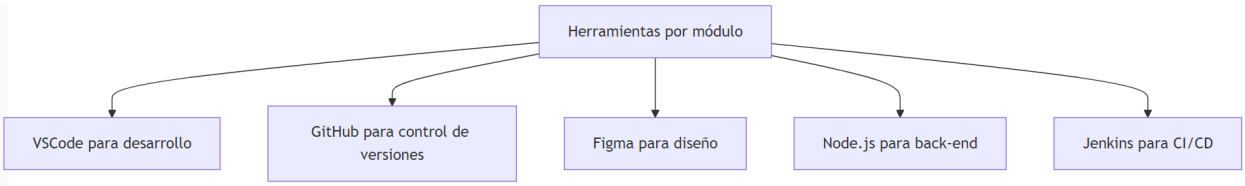
4.2 Reconoce la naturaleza del trabajo que será realizado junto con las herramientas que serán utilizadas a lo largo de cada módulo

Cada módulo del curso está orientado a la adquisición de **habilidades técnicas** específicas y, en conjunto, proporcionan las bases para la construcción de un producto digital completo. Además, cada módulo incluye una serie de **herramientas** que permiten que el trabajo se lleve a cabo de manera eficiente.

Herramientas comunes en los módulos:

- **Visual Studio Code (VSCode)**: Utilizado para escribir y gestionar código.
- **GitHub/GitLab**: Herramientas de control de versiones y colaboración en proyectos.
- **Figma/Sketch**: Software para el diseño de interfaces y creación de prototipos.
- **Node.js/Express**: Frameworks y entornos para el desarrollo de back-end.
- **Jenkins/CircleCI**: Herramientas de integración continua para automatizar pruebas y despliegues.

Cada una de estas herramientas se selecciona para apoyar el flujo de trabajo y garantizar que los estudiantes adquieran experiencia práctica en proyectos reales.

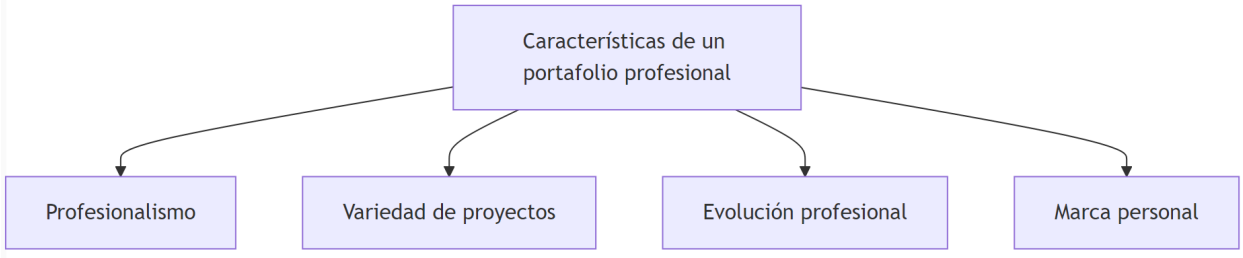


4.3 Reconoce la importancia de un portafolio de producto así como sus características para la formación de una identidad profesional

Un **portafolio de productos** es mucho más que una recopilación de proyectos: es una herramienta clave para formar una **identidad profesional**. En este sentido, un portafolio bien estructurado debe reflejar no solo las habilidades técnicas, sino también la evolución y madurez profesional del desarrollador.

Características clave de un buen portafolio:

1. **Profesionalismo**: Debe estar bien diseñado, ser fácil de navegar y mostrar una atención al detalle que hable sobre el trabajo y la seriedad del desarrollador.
2. **Variedad de proyectos**: Debe mostrar una diversidad de habilidades, desde diseño front-end hasta desarrollo back-end y despliegue de productos completos.
3. **Evolución**: Debe mostrar la evolución del desarrollador a lo largo del tiempo, reflejando una mejora continua en términos de calidad y complejidad de los proyectos.
4. **Marca personal**: El portafolio debe comunicar una identidad visual y profesional que lo distinga.



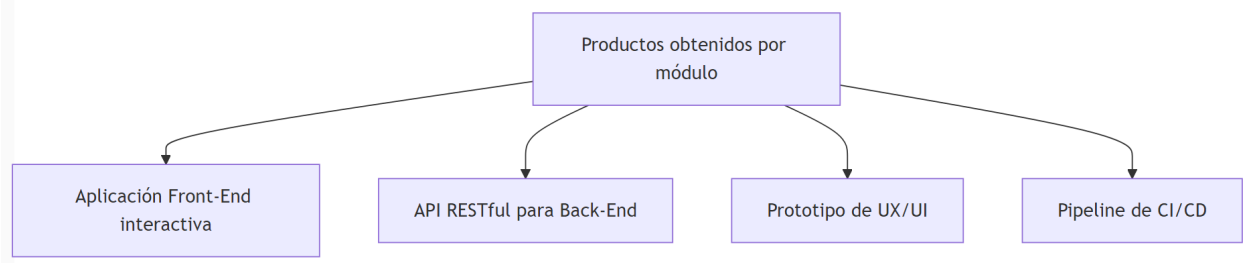
4.4 Identifica los productos que serán obtenidos en cada módulo así como su contribución al portafolio de producto

Cada módulo del curso no solo enseña competencias, sino que también culmina en la creación de **productos tangibles** que pueden ser integrados en el portafolio de un estudiante. Estos productos son fundamentales para demostrar las habilidades adquiridas y se deben integrar en el portafolio de manera estratégica.

Ejemplos de productos por módulo:

- **Módulo de Front-End**: Una aplicación web interactiva con una interfaz bien diseñada y responsive.
- **Módulo de Back-End**: Una API RESTful que conecta una base de datos con la aplicación front-end.
- **Módulo de UX/UI**: Un prototipo funcional de alta fidelidad para una aplicación web o móvil.
- **Módulo de CI/CD**: Un pipeline de integración continua configurado para automatizar pruebas y despliegues.

Cada uno de estos productos debe ser acompañado de una **descripción detallada** en el portafolio que explique el proceso de desarrollo, las tecnologías utilizadas y el impacto del proyecto.



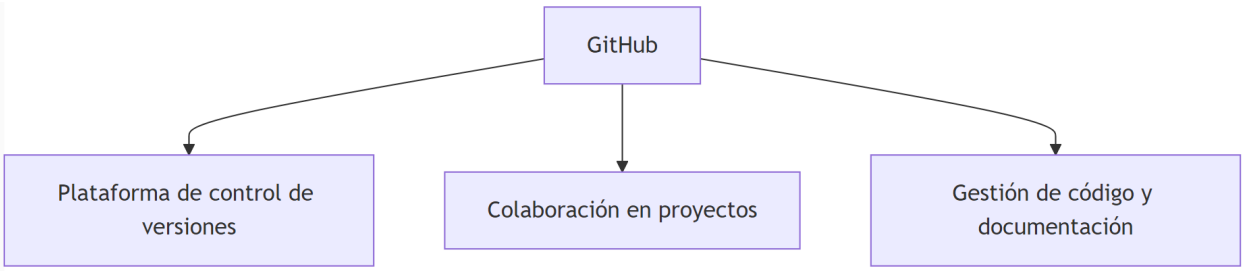
Incorporar estos productos en un portafolio no solo muestra lo que el estudiante ha aprendido, sino también cómo estas habilidades pueden aplicarse para resolver problemas reales, posicionando al desarrollador como un profesional competente y versátil.

5. Utilización de GitHub para Crear un Portafolio

5.1 Qué es GitHub

GitHub es una plataforma basada en la web que permite a los desarrolladores almacenar, compartir y colaborar en proyectos de software utilizando **Git**, un sistema de control de versiones distribuido. GitHub es ampliamente utilizado por desarrolladores y empresas para gestionar el código fuente, realizar revisiones de código, y llevar un historial de cambios de los proyectos.

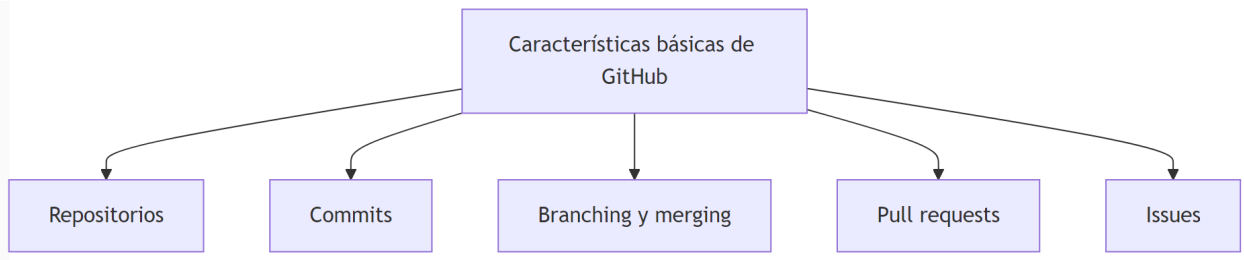
Además de ser una herramienta colaborativa, GitHub es ideal para crear y compartir un **portafolio de productos** ya que permite mostrar el código fuente de los proyectos, incluir descripciones detalladas y proporcionar acceso a demos o aplicaciones desplegadas.



5.2 Características y operaciones básicas de GitHub

GitHub ofrece una serie de características y herramientas que facilitan la colaboración y la gestión de proyectos. Las operaciones más básicas y esenciales que se deben manejar para crear un portafolio son:

1. **Repositorios**: Contenedores donde se almacena el código fuente, la documentación y los recursos relacionados con un proyecto.
2. **Commits**: Cambios o actualizaciones en el código que se guardan en el repositorio.
3. **Branching y merging**: Crear ramas separadas para trabajar en diferentes partes de un proyecto sin afectar el código principal. Una vez que los cambios son aprobados, se pueden fusionar con la rama principal.
4. **Pull requests**: Solicitudes para revisar y fusionar cambios en el proyecto, utilizadas comúnmente en trabajos colaborativos.
5. **Issues**: Herramienta de seguimiento de problemas y mejoras que permite gestionar el flujo de trabajo de un proyecto.



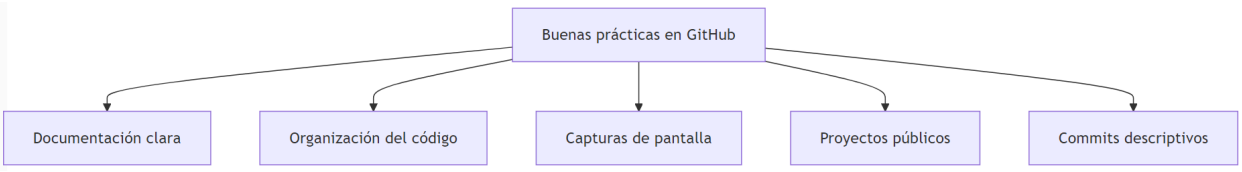
Ejemplos de comandos básicos de Git:

```
# Clonar un repositorio existente
git clone https://github.com/usuario/proyecto.git

# Añadir cambios al área de preparación
git add .

# Crear un commit
git commit -m "Añadir nueva funcionalidad"

# Subir cambios al repositorio remoto
git push origin main
```

5.3 Buenas prácticas para tener un portafolio atractivo en GitHub

Un portafolio en GitHub no solo debe ser funcional, sino también atractivo y fácil de navegar. Algunas buenas prácticas para lograrlo son:

1. **Documentación clara**: Asegúrate de que cada repositorio incluya un archivo `README.md` bien estructurado, donde expliques el propósito del proyecto, las tecnologías utilizadas y cómo ejecutarlo.
2. **Organización del código**: Mantén el código limpio y bien organizado, utilizando nombres de archivos y carpetas que sean significativos y fáciles de entender.
3. **Capturas de pantalla o demos**: Incluye imágenes o enlaces a demostraciones en vivo para que los visitantes puedan ver el proyecto en acción sin necesidad de descargar el código.
4. **Proyectos públicos**: Asegúrate de que los repositorios que quieras mostrar en tu portafolio sean públicos para que cualquier persona pueda acceder a ellos.
5. **Commits descriptivos**: Usa mensajes de commit descriptivos para que sea fácil entender los cambios realizados en cada etapa del proyecto.



5.4 Buenas prácticas para la página principal de un repositorio

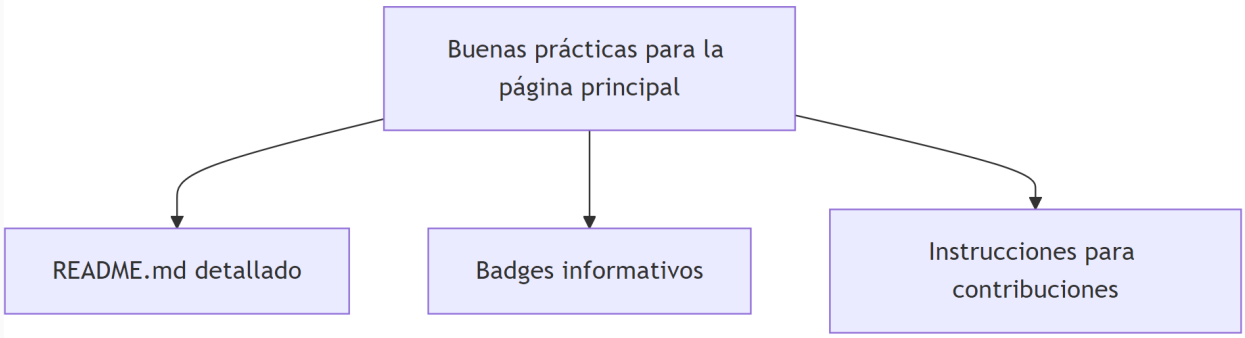
La **página principal de un repositorio** es el primer lugar que verán los visitantes de tu portafolio en GitHub, por lo que debe estar bien estructurada y ser informativa. Algunas recomendaciones para mejorar la página principal son:

1. **README.md detallado**: El archivo `README.md` debe proporcionar una descripción clara del proyecto, incluyendo los siguientes apartados:

    - **Descripción del proyecto**: Un resumen sobre qué hace el proyecto y para qué se utiliza.
    - **Tecnologías utilizadas**: Un listado de las tecnologías y lenguajes de programación empleados.
    - **Instrucciones de instalación y uso**: Pasos claros para que otros usuarios puedan clonar y ejecutar el proyecto en sus entornos.
    - **Licencia**: Incluir una licencia clara sobre los derechos de uso del código.

2. **Badges**: Añadir **badges** (insignias) que muestren el estado del proyecto, como las pruebas que pasan, el lenguaje usado o el estado de la build. Estas se pueden incluir en el README.md para mejorar la presentación visual.

3. **Contribuciones**: Incluir una sección sobre cómo otros desarrolladores pueden contribuir al proyecto, detallando el flujo de trabajo de **branching** y **pull requests**.



Ejemplo de README.md:

```
# Mi Proyecto Increíble
```

```
Este proyecto es una aplicación web que permite gestionar tareas de manera
eficiente.

## Tecnologías utilizadas:

- HTML5, CSS3, JavaScript
- Node.js y Express
- MongoDB

## Instalación:
1. Clonar el repositorio: `git clone
https://github.com/usuario/proyecto.git`
2. Instalar dependencias: `npm install`
3. Ejecutar la aplicación: `npm start`

## Licencia:

Este proyecto está bajo la licencia MIT.
```

5.5 Buenas prácticas para la página de perfil

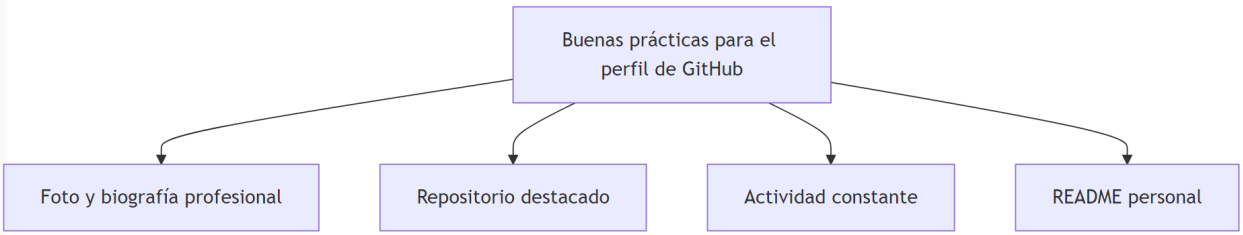
La **página de perfil** de GitHub es tu tarjeta de presentación. Un perfil bien diseñado y completo puede atraer la atención de reclutadores, colaboradores y posibles clientes. Aquí algunas recomendaciones para mejorar tu perfil:

1. **Foto de perfil y biografía**: Usa una foto profesional y escribe una breve biografía que describa tus áreas de especialización y tus intereses. Mantén el tono profesional pero accesible.

2. **Repositorio destacado**: GitHub permite fijar proyectos destacados en la parte superior de tu perfil. Elige tus mejores proyectos para que sean lo primero que vean los visitantes.

3. **Actividad constante**: Mantén tu perfil activo contribuyendo a proyectos, realizando commits y participando en revisiones de código. GitHub muestra un gráfico de contribuciones que ayuda a los visitantes a ver tu nivel de actividad.

4. **README personal**: GitHub permite añadir un `README.md` a tu perfil para personalizarlo. Aquí puedes incluir un resumen de tu carrera, los proyectos en los que trabajas, y enlaces a redes sociales o tu página personal.



Ejemplo de README personal:

```
# ¡Hola, soy [Tu Nombre]! 👋

Soy un desarrollador Full Stack apasionado por crear soluciones
tecnológicas que resuelvan problemas reales. Trabajo con tecnologías como
JavaScript, Node.js y React.

## Proyectos Destacados:

- [Gestor de Tareas](https://github.com/usuario/gestor-tareas): Una
aplicación para gestionar tareas con funcionalidades de tiempo real.

- [Blog Personal](https://github.com/usuario/blog-personal): Un blog
desarrollado con Next
```

6. Utilización de Behance para Crear un Portafolio

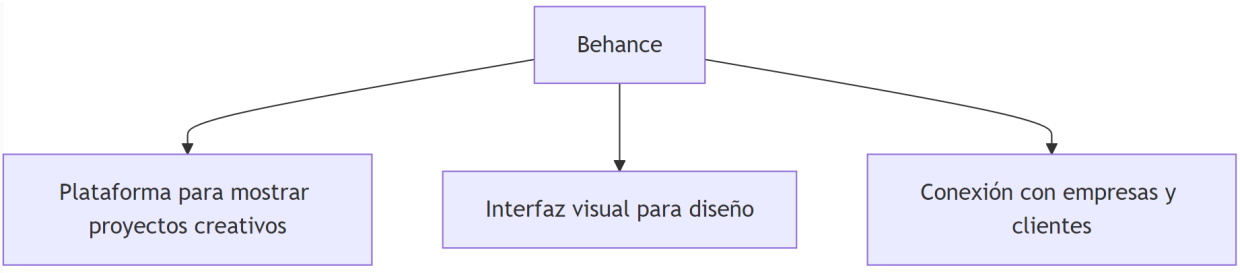
6.1 Qué es Behance

Behance es una plataforma en línea que permite a diseñadores, artistas y creativos compartir y mostrar sus proyectos a una comunidad global. Es especialmente útil para profesionales en

áreas como el diseño gráfico, la fotografía, la ilustración y el diseño de interfaces de usuario (UI) y experiencia de usuario (UX). Behance permite a los usuarios construir un **portafolio visual** atractivo y descubrir nuevas oportunidades de empleo o colaboración con otras personas en la industria creativa.

Ventajas de Behance:

- Exposición a una comunidad global de creativos y empresas.
- Interfaz visual que facilita la navegación por proyectos artísticos y de diseño.
- Ideal para profesionales que buscan destacar su trabajo creativo y visual.



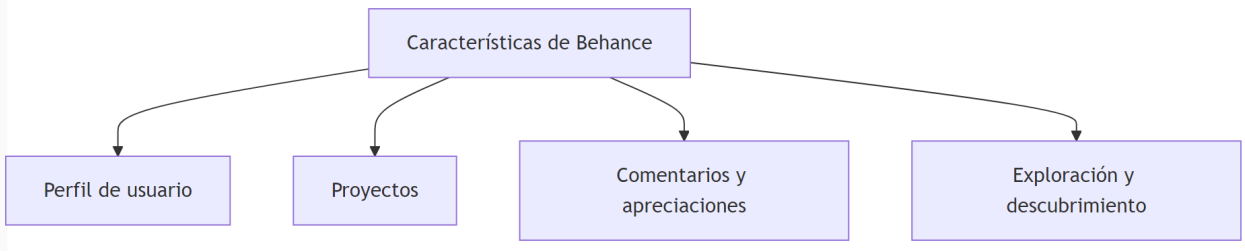
6.2 Características y operaciones básicas de Behance

Behance ofrece una serie de herramientas y funciones que permiten a los usuarios subir sus proyectos, organizarlos y compartirlos con la comunidad. Algunas de las **características básicas** incluyen:

1. **Perfil de usuario**: En Behance, el perfil de cada usuario muestra su portafolio y detalles profesionales, como las habilidades, experiencia y enlaces a otros sitios.
2. **Proyectos**: Los usuarios pueden subir proyectos individuales con imágenes, descripciones, videos y enlaces externos. Estos proyectos se organizan dentro del portafolio y pueden ser presentados como trabajos individuales o colaborativos.
3. **Comentarios y apreciaciones**: Otros usuarios pueden dejar comentarios y dar "apreciaciones" a los proyectos, ayudando a incrementar la visibilidad del trabajo.
4. **Exploración y descubrimiento**: Behance tiene un motor de búsqueda robusto que permite a los usuarios descubrir nuevos proyectos basados en categorías como diseño gráfico, fotografía, UI/UX, y más.

Operaciones básicas:

- **Crear un proyecto**: Sube imágenes, videos, archivos PDF o GIFs para mostrar tu trabajo.
- **Organizar el portafolio**: Clasifica los proyectos por categorías para que sea más fácil para los usuarios navegar.
- **Compartir el proyecto**: Una vez que el proyecto esté listo, puedes compartirlo públicamente o directamente a través de enlaces en redes sociales.

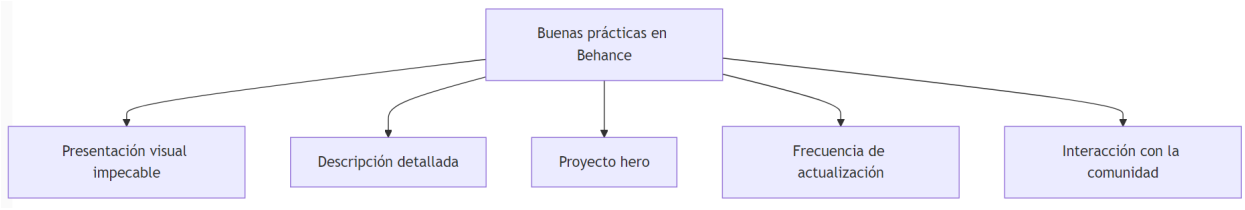


6.3 Buenas prácticas para tener un portafolio atractivo en Behance

Crear un **portafolio atractivo en Behance** implica mucho más que solo subir proyectos. Para captar la atención de posibles empleadores o clientes, es importante seguir ciertas buenas prácticas:

1. **Presentación visual impecable**: Behance es una plataforma altamente visual, por lo que es crucial que cada proyecto esté bien presentado. Utiliza imágenes de alta calidad, asegúrate de que todo esté alineado y que los colores y tipografías sean consistentes.
2. **Descripción detallada**: Acompaña cada proyecto con una **descripción clara** que explique el objetivo, el proceso de desarrollo, las herramientas utilizadas y los resultados obtenidos. Incluye capturas del proceso de trabajo para mostrar cómo llegaste al producto final.
3. **Proyecto "hero"**: Elige un proyecto que represente lo mejor de tu trabajo para destacarlo en tu portafolio. Asegúrate de que sea visualmente impactante y que muestre un conjunto de habilidades variadas.
4. **Frecuencia de actualización**: Mantén tu portafolio actualizado con tus últimos proyectos. La actividad constante ayuda a mejorar tu visibilidad en la plataforma.

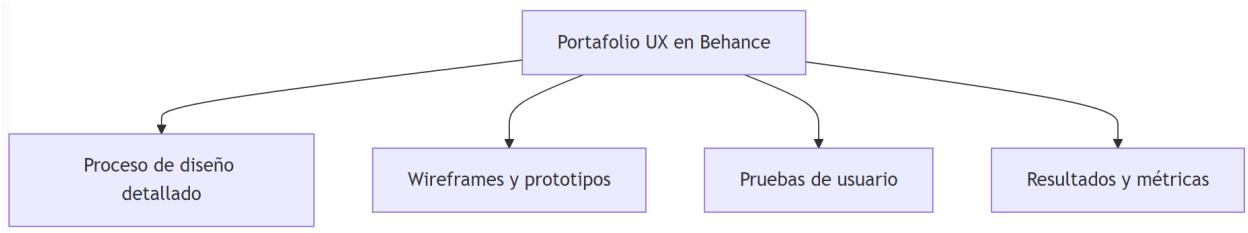
5. **Interacción con la comunidad**: Participa en la comunidad de Behance apreciando y comentando los proyectos de otros usuarios. Esto no solo aumenta tu visibilidad, sino que también te ayuda a aprender y establecer conexiones.



6.4 Behance dedicado y exclusivo a proyectos de UX

Behance es una herramienta excelente para mostrar proyectos de **UX (User Experience)**, dado que permite explicar en detalle el proceso de diseño y el razonamiento detrás de las decisiones tomadas. Para que un portafolio dedicado a UX sea efectivo en Behance, es importante destacar varios aspectos:

1. **Proceso de diseño**: Explica cómo se investigaron las necesidades del usuario, cómo se definieron los problemas y qué soluciones se desarrollaron para resolver esos problemas.
2. **Wireframes y prototipos**: Muestra imágenes de los wireframes y prototipos que se utilizaron en las primeras etapas del diseño para validar las ideas.
3. **Pruebas de usuario**: Incluye resultados y capturas de pantalla de las pruebas realizadas con usuarios. Muestra cómo los comentarios de los usuarios fueron incorporados en el diseño final.
4. **Resultados y métricas**: Si es posible, muestra métricas de éxito, como la mejora en la tasa de conversión o el aumento en la satisfacción del usuario después de la implementación del producto.



Al seguir estas prácticas, Behance se convierte en una plataforma poderosa para mostrar trabajos centrados en **UX/UI** y comunicar el valor detrás de las decisiones de diseño, lo cual es esencial para atraer la atención de potenciales clientes o empleadores que busquen diseñadores especializados en experiencia de usuario.

7. Alojamiento de tu Producto en un Servidor

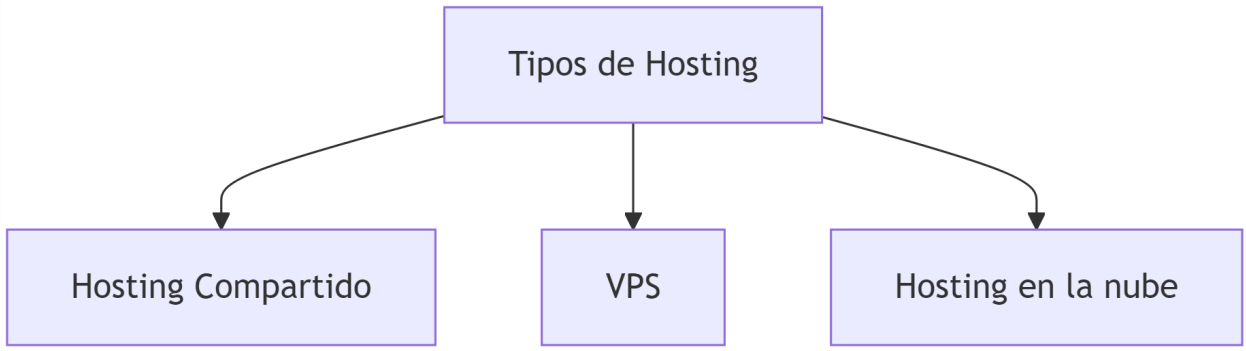
7.1 Qué es un Hosting

El **hosting** (alojamiento web) es un servicio que permite almacenar y publicar un sitio web o aplicación en Internet. Los **servidores de hosting** son equipos conectados a la red que están configurados para ejecutar y mostrar páginas web o aplicaciones a los usuarios cuando acceden a ellas a través de un navegador o un dispositivo móvil.

Existen diferentes tipos de hosting que varían según el nivel de recursos, personalización y costo, como **hosting compartido**, **VPS (servidores privados virtuales)**, y **hosting en la nube**.

Tipos de Hosting:

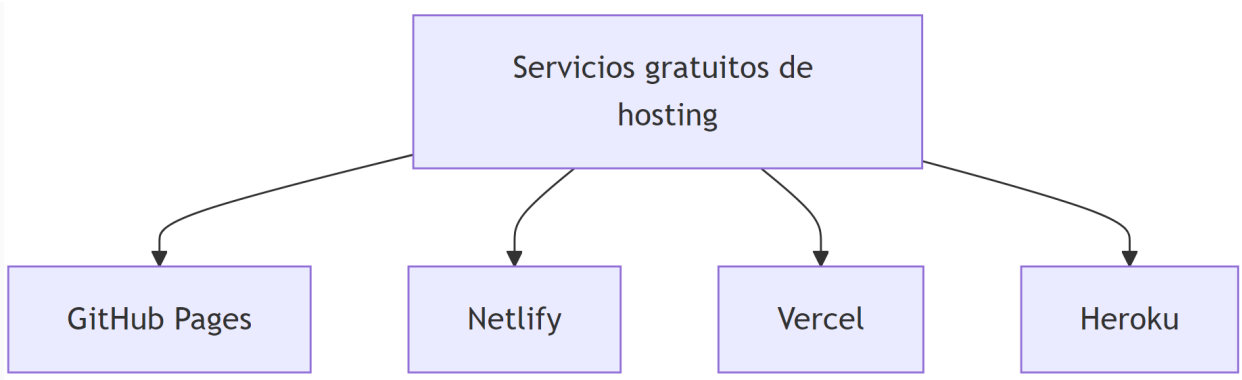
1. **Compartido**: Varios sitios web comparten los mismos recursos del servidor.
2. **VPS**: Ofrece una porción de un servidor dedicado con mayor control y recursos.
3. **Hosting en la nube**: Utiliza una red de servidores para alojar el sitio, lo que permite una mayor escalabilidad y flexibilidad.



7.2 Servicios gratuitos de hosting

Existen varios **servicios de hosting gratuitos** que son ideales para proyectos pequeños, demostraciones o portafolios. A continuación, algunos de los servicios más utilizados:

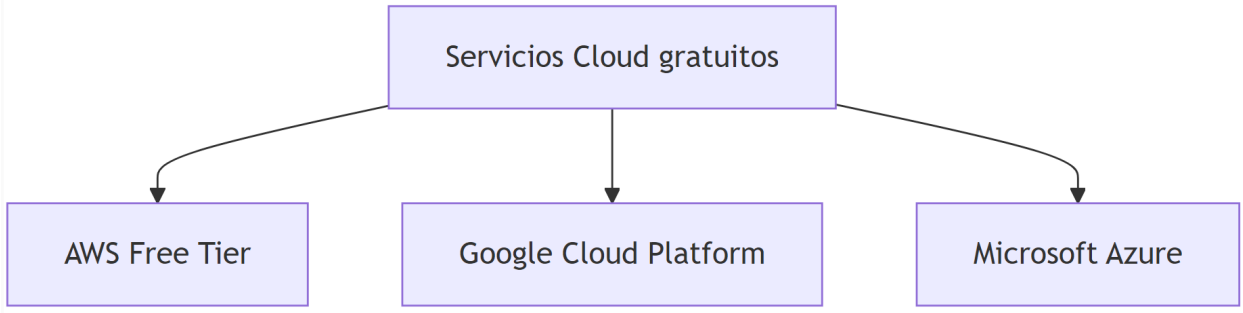
1. **GitHub Pages**: Permite alojar sitios estáticos de manera gratuita, como páginas web personales o portafolios. Ideal para proyectos de Front-End.
2. **Netlify**: Una opción gratuita para alojar proyectos estáticos y JAMstack. Permite integración con Git y despliegues automáticos.
3. **Vercel**: Similar a Netlify, especializado en sitios estáticos y aplicaciones con **Next.js**.
4. **Heroku**: Ofrece hosting gratuito para aplicaciones de back-end y fullstack con lenguajes como Node.js, Python, y Java.



7.3 Servicios cloud gratuitos

Los **servicios cloud gratuitos** ofrecen más flexibilidad y potencia que los servicios de hosting estático, permitiendo alojar aplicaciones más complejas, como aplicaciones fullstack o APIs.

1. **AWS Free Tier**: Ofrece una capa gratuita con acceso a servicios como **EC2** (servidores virtuales) y **S3** (almacenamiento en la nube).
2. **Google Cloud Platform (GCP)**: También tiene una capa gratuita que incluye servicios de almacenamiento, máquinas virtuales y despliegue de contenedores.
3. **Microsoft Azure**: Ofrece servicios gratuitos para alojar aplicaciones, bases de datos y contenedores.



7.4 Cómo alojar un proyecto UX/UI

Los proyectos UX/UI suelen ser **estáticos**, lo que significa que no requieren un servidor de back-end. Algunas opciones recomendadas para alojar estos proyectos incluyen:

1. **GitHub Pages**: Ideal para proyectos que consisten en HTML, CSS, y JavaScript. Simplemente sube tu proyecto a un repositorio y habilita GitHub Pages.
2. **Netlify**: Permite alojar proyectos estáticos de forma rápida y fácil, con la opción de configurar despliegues automáticos al hacer push en el repositorio.
3. **Vercel**: Ofrece una integración excelente con Next.js para proyectos estáticos, permitiendo un despliegue rápido con URLs personalizadas.

```
# Ejemplo: Desplegar en Netlify
npm run build
netlify deploy
```

7.5 Cómo alojar un proyecto Front-End

Para un proyecto front-end que solo utilice HTML, CSS, y JavaScript, el **hosting estático** es la mejor opción. Algunas de las plataformas mencionadas, como **Netlify**, **GitHub Pages** o **Vercel**, son ideales para estos casos.

Pasos básicos para alojar un proyecto front-end en GitHub Pages:

1. Sube tu código al repositorio de GitHub.
2. En la configuración del repositorio, habilita **GitHub Pages**.

3. Elige la rama donde está almacenado el código y selecciona la carpeta `/docs` o la raíz del proyecto.
4. GitHub generará un enlace donde se podrá acceder al sitio.

```
# Ejemplo: Desplegar en GitHub Pages
git add .
git commit -m "Despliegue inicial"
git push origin main
```

7.6 Cómo alojar un proyecto Fullstack Java

Para un proyecto fullstack en **Java**, necesitarás un servicio de hosting que soporte aplicaciones back-end y bases de datos. **Heroku** y **AWS** son opciones populares para aplicaciones Java.

Pasos básicos para desplegar una aplicación Java en Heroku:

1. Configura tu archivo **pom.xml** o **build.gradle** para que Heroku reconozca la aplicación.
2. Crea un archivo **Procfile** que especifique el comando para ejecutar tu aplicación.
3. Sube el proyecto a Heroku con el CLI de Heroku:

```
heroku create
git push heroku main
heroku open
```

7.7 Cómo alojar un proyecto Fullstack Javascript

Para proyectos fullstack en **JavaScript**, especialmente aquellos que utilizan tecnologías como **Node.js** o frameworks como **Express**, **Heroku** y **Vercel** son opciones populares.

Desplegar un proyecto Node.js en Vercel:

1. Instala el CLI de Vercel:

```
npm install -g vercel
```

2. En el directorio de tu proyecto, ejecuta:

```
vercel
```

3. Vercel generará automáticamente la URL donde tu proyecto estará alojado.

7.8 Cómo alojar un proyecto Fullstack Python

Para proyectos desarrollados en **Python** con frameworks como **Flask** o **Django**, se pueden utilizar servicios como **Heroku** o **Google Cloud**. Ambos soportan aplicaciones Python de manera eficiente.

Desplegar un proyecto Django en Heroku:

1. Asegúrate de tener un archivo **requirements.txt** que incluya las dependencias de Python.
2. Crea un archivo **Procfile** que indique a Heroku cómo ejecutar la aplicación.
3. Despliega con:

```
git push heroku main
heroku open
```

7.9 Cómo alojar un proyecto Android

Para alojar un proyecto **Android**, generalmente lo que se publica es el archivo **APK** en una tienda de aplicaciones o un servidor para descargas.

Opciones para compartir un APK:

1. **Google Play Store**: Subir el APK a Google Play requiere una cuenta de desarrollador, pero garantiza mayor alcance.
2. **GitHub Releases**: Puedes alojar el APK como una **release** en un repositorio de GitHub.
3. **Amazon Appstore**: Otra opción de distribución es subir el APK a la tienda de Amazon.

```
# Para alojar un APK en GitHub
git tag -a v1.0 -m "Primera versión"
git push origin v1.0
```

Con estas opciones de hosting y despliegue, puedes hacer que tus proyectos estén disponibles para el público, facilitando el acceso y la demostración de tu trabajo a clientes, empleadores o colaboradores.

Material de Referencia

Libros:

- **"Host Your Web Site in the Cloud"** de Eric A. Marks
  Este libro proporciona una guía clara sobre cómo utilizar los servicios de **cloud hosting** para desplegar aplicaciones y sitios web. Cubre tanto conceptos básicos como avanzados de plataformas cloud como **AWS**, **Azure**, y **Google Cloud**.

- **"Pro Git"** de Scott Chacon y Ben Straub
  Este libro no solo se enfoca en Git, sino que también proporciona una introducción a **GitHub Pages**, que es ideal para alojar portafolios de proyectos front-end o sitios estáticos. Pro Git - Libro Gratis

- **"The Full Stack Developer"** de Chris Northwood
  En este libro, aprenderás cómo alojar proyectos fullstack en diferentes plataformas. Aborda tecnologías como **JavaScript**, **Python**, **Java**, y los diversos servicios de cloud computing para hosting.

- **"Cloud Native Development Patterns and Best Practices"** de John Gilbert
  Este libro es perfecto para desarrolladores que buscan aprender a implementar aplicaciones utilizando patrones y prácticas de **desarrollo nativo en la nube**. Cubre estrategias para desplegar proyectos utilizando servicios como **AWS** y **Google Cloud**.

Enlaces a Recursos Online:

- **GitHub Pages Documentation**
  La documentación oficial para usar **GitHub Pages** para alojar sitios estáticos. Ideal para portafolios o proyectos front-end que no requieren un servidor back-end.

- **Netlify Documentation**
  Una guía completa sobre cómo utilizar **Netlify** para alojar sitios web estáticos o proyectos JAMstack. Ofrece soporte para despliegue continuo y administración de DNS.

- **Heroku - Getting Started with Node.js**
  Documentación oficial de **Heroku** que explica cómo desplegar aplicaciones **Node.js** en su plataforma. Proporciona ejemplos claros y comandos para iniciar un proyecto desde cero.

- **Vercel - Documentation**
  Documentación para aprender a desplegar aplicaciones con **Vercel**, especialmente optimizado para **Next.js** y aplicaciones de React. Explica cómo configurar despliegues automáticos y manejo de dominio.

- **AWS Free Tier Documentation**
  La documentación oficial de **AWS Free Tier**, que detalla cómo aprovechar los servicios gratuitos para desplegar aplicaciones de back-end o fullstack utilizando **EC2**, **Lambda**, y otros servicios.

Videos Recomendados:

- **Deploying Fullstack Applications to Heroku (Traversy Media)**
  Este video tutorial muestra cómo desplegar una aplicación fullstack en **Heroku**, abarcando tanto la configuración del servidor como las bases de datos.

- **Deploy Your React App to Netlify**
  Un tutorial paso a paso sobre cómo desplegar aplicaciones **React** a **Netlify**, una de las plataformas de hosting más fáciles para proyectos front-end.

- **How to Deploy a Fullstack MERN App to Heroku**
  Un tutorial completo que enseña cómo desplegar aplicaciones **MERN** (MongoDB,

Express, React, Node.js) en **Heroku**, desde la configuración inicial hasta el despliegue final.

- **How to Deploy Django Apps on Heroku**
  Un video útil para aquellos que quieren desplegar una aplicación de **Django** en **Heroku**, con ejemplos prácticos de configuración.

Otros Recursos Útiles:

- **Netlify Cheat Sheet**
  Una guía rápida con todos los comandos y configuraciones para utilizar **Netlify** y desplegar sitios estáticos.

- **Heroku Cheat Sheet**
  Una hoja de referencia rápida con los comandos más utilizados para desplegar y gestionar aplicaciones en **Heroku**.

- **Vercel GitHub Integration**
  Documentación sobre cómo integrar **Vercel** con **GitHub** para hacer despliegues automáticos cada vez que se hace un commit en el repositorio.

CORFO
BecasCapital Humano

Adalid
aspasia
LA FORMACIÓN DE TU FUTURO

MÓDULO 9

DESARROLLO DE PORTAFOLIO DE UN PRODUCTO DIGITAL