Guia Ágil

TechGuide - Alura

Full-stack

Nivel 1

- [ ] **HTML - Fundamentos**:

- HTML es un lenguaje de marcado que define la estructura de su contenido. HTML consta de una serie de elementos que se utilizan para que se vea o actúe de cierta manera. Las etiquetas de archivos adjuntos pueden vincular una palabra o imagen a otro lugar, pueden poner palabras en cursiva, pueden hacer que la fuente sea más grande o más pequeña, etc.
- Aprender qué etiquetas son necesarias para HTML básico
- Crear de un párrafo de texto
- Mostrar una imagen
- Conocer la diferencia entre 'h1', 'h2', 'h3', etc.
- Crear de un texto con hipervínculo
- Crear un formulario con campos relevantes
- Crear de una lista ordenada o desordenada de elementos
- Crear de una lista de elementos en una lista desplegable
- Vincular a un archivo CSS
- Crear una tabla
- Adicionar ID y clases

- [ ] **Javascript - Fundamentos**:
    - Javascript es el lenguaje de programación más popular del mundo y es una de las tecnologías centrales de la World Wide Web, junto con HTML y CSS. Tiene escritura dinámica, orientación a objetos basada en prototipos y funciones de primera clase. Es un paradigma múltiple que admite estilos de programación imperativos, funcionales e impulsados por eventos.
    - Conocer los tipos primitivos
    - Declarar variables, considerando la diferencia entre 'var', 'let' y 'const
    - Uso de estructuras condicionales ('if', 'else')
    - Conocer los operadores de asignación y comparación ('=', '==', '===')
    - Uso de estructuras de repetición y bucles ('while', 'for')
    - Usar funciones, pasar parámetros y argumentos
    - Manipulación de arreglos y listas
    - Aprender el concepto de Orientación a Objetos
    - Realizar un CRUD
    - Obtener datos de una API
    - Hacer llamadas asincrónicas usando 'Async/Await', 'Promise', etc.
- [ ] **Node.js - Fundamentos**:
    - Node.js es un entorno de ejecución de JavaScript que permite ejecutar aplicaciones desarrolladas en este lenguaje de manera autónoma, sin depender de un navegador.
    - Aprender sobre operaciones bloqueantes y no bloqueantes
    - Comprender el concepto de bucle de eventos (event loop)
    - Aprender a utilizar las bibliotecas de Node.js, como 'net', 'fs', 'http', 'path', entre otras
    - Entender cómo funcionan los temporizadores (Timers)
- [ ] **CSS - Fundamentos**:

- Las hojas de estilo en cascada (CSS) son un lenguaje de hoja de estilo usado para descubrir una presentación de un documento escrito en un lenguaje de marca, como HTML o XML. O CSS puede ser usado para estilizar textos de documentos muy básicos — por ejemplo, para alterar a cor e o tamanho de cabeçalhos e links. Ele pode ser usado para criar um layout — por ejemplo, transformando uma única columna de texto en um layout com uma área de conteúdo principal e uma barra lateral para información relacionada. Pode até ser usado para efectos como animação.
- Aprender a estructura visual de uma página, con 'margem' e 'preenchimento'
- Estabelecer o tamanho com 'larga' e 'altura'
- Aprender sobre la posición de un elemento ('estático', 'relativo' o 'absoluto')
- Aprender sobre la exposición de un elemento ('block', 'inline', 'inline-block')
- Aprender a posicionar imágenes en relación con el texto
- Aprender sobre alinhamento
- Aprender sobre estilo de fuente
- Aprender las diferencias y ventajas de usar las diferentes unidades de medida en CSS (%, relativo, etc.)
- Conectar-se a los elementos (IDs, clases) de un archivo HTML
- Cambiar las características de un elemento cuando se pasa el ratón sobre él
- Aprender a dimensionar cajas
- Aprender Flexbox
- Grado de aprendizaje

☐ **DOM - Fundamentos**:

- El Document Object Model (DOM) es una interfaz de programación para documentos de la web. Representa la página para que los programas puedan cambiar la estructura, el estilo y el contenido del documento. El DOM representa el documento como nosotros y objetos; de esa forma, los lenguajes de programación pueden interactuar con la página.
- Entender cómo funciona el árbol DOM

- Accesando y manipulando elementos HTML y CSS
- Acceder a los padres e hijos de un elemento
- Insertar un nuevo elemento en el árbol
- Quitar un elemento del árbol
- Esperando un evento en un elemento determinado de la página usando

- [ ] **Crear una aplicación React**:

- Create React App es una forma oficialmente admitida de crear aplicaciones React de una sola página. Ofrece una configuración de construcción moderna sin configuración.
- Estructuración de un nuevo proyecto React
- Crear una aplicación funcional desde cero

- [ ] **Conceptos de Orientación a Objetos**:

- La Programación Orientada a Objetos es un paradigma de programación de software basado en la composición e interacción entre diversas unidades llamadas 'objetos' y las clases, que contienen una identidad, propiedades y métodos. Se basa en cuatro componentes de la programación - abstracción digital, encapsulación, herencia y polimorfismo.
- Cómo funcionan los objetos
- Crear y utilizar constructores
- Qué son las clases
- Crear y utilizar métodos
- Cómo funciona la encapsulación
- Qué es la herencia
- Qué es el polimorfismo
- Cómo funcionan las interfaces
- Qué son las abstracciones

- [ ] **Estructura de Datos**:

- En el contexto de los ordenadores, una estructura de datos es una forma específica de almacenar y organizar los datos en la memoria del ordenador para que esos datos puedan ser fácilmente recuperados y utilizados de forma eficiente cuando sea necesario posteriormente.
- Conocer las principales estructuras de datos
- Implementar las principales estructuras de datos

☐ **Java - Fundamentos**:

- Java es un lenguaje de programación ampliamente utilizado para codificar aplicaciones web. Java es un lenguaje multiplataforma, orientado a objetos y centrado en red que se puede utilizar como una plataforma en sí. Es un lenguaje de programación rápido, seguro y confiable para codificar todo, desde aplicaciones móviles y software empresarial hasta aplicaciones de big data y tecnologías de servidor.
- Conocer los tipos primitivos.
- Declarar variables, considerando los diferentes tipos.
- Usar estructuras condicionales ('if', 'Else').
- Conocer a los operadores de comparación.
- Utilizar estructuras de repetición y bucles ('while', 'for').
- Usar funciones, pasar parámetros y argumentos.
- Manipular métodos.
- Manipular arrays y listas.
- Obtener datos de una API.
- Hacer llamadas asíncronas 'Future', etc.
- Crear constructores.

☐ **Python - Fundamentos**:

- Python es un lenguaje de programación de alto nivel de uso general, ampliamente utilizado en aplicaciones web, desarrollo de software, ciencia de datos y aprendizaje automático. Su filosofía de diseño se centra en la

legibilidad del código mediante el uso de sangría significativa. Python es de tipado dinámico y cuenta con un recolector de basura.

- Los tipos de datos primitivos.
- Declarar variables, teniendo en cuenta los diferentes tipos.
- Utilizar estructuras condicionales ('if', 'else').
- Conocer los operadores de asignación y comparación.
- Usar estructuras de repetición y bucles ('while', 'for').
- Utilizar funciones, pasando parámetros y argumentos.
- Manipular métodos.
- Manipular arrays y listas.
- Obtener datos de una API.
- Crear constructores.
- Utilizar funciones anónimas.

☐ **Accesibilidad en Javascript**:

- La Accesibilidad Digital es la eliminación de barreras en la Web. El concepto presupone que los sitios y portales sean diseñados de modo que todas las personas puedan percibir, entender, navegar e interactuar de manera efectiva con las páginas.
- Escribir código teniendo en cuenta la accesibilidad.

☐ **Angular - Fundamentos**:

- Angular es un framework para construir aplicaciones y una plataforma de desarrollo construida en TypeScript para crear aplicaciones eficientes y sofisticadas de página única (SPA).
- Construir interfaces utilizando HTML, CSS y TypeScript
- Crear aplicaciones SPA
- Construir aplicaciones web, mobile o desktop
- Integrar datos con API REST
- Utilizar la composición para crear componentes reutilizables

- Utilizar servicios de tipo Resolver
- Manipular solicitudes creando servicios de tipo Interceptor

Nivel 2

- [ ] **Node.js - Express**:

- Express es un framework flexible de aplicaciones web para Node.js que proporciona un conjunto robusto de recursos para aplicaciones web y móviles.
- Utiliza el framework Express para la creación de APIs REST con Node.js
- Administra peticiones de diferentes verbos HTTP en distintas URLs
- Define el puerto a utilizar para la conexión y la ubicación de los modelos que se utilizan para renderizar la respuesta
- Crea manejadores de rutas utilizando el método 'router'
- Conoce la biblioteca 'Router' y sus verbos HTTP, como 'get', 'post', 'put', etc.
- Define endpoints con route paths

- [ ] **React - Componentes**:

- React te permite definir componentes como clases o funciones. Los componentes definidos como clases proporcionan más capacidades. Aceptan entradas arbitrarias (llamadas "accesorios") y devuelven elementos React que describen lo que debería aparecer en la pantalla.
- Comprender cómo funcionan los componentes
- Conociendo la biblioteca de componentes con estilo
- Comprender la diferencia entre clase y componentes funcionales

- [ ] **Node.js - ORM**:

- Object-Relational Mapping (ORM), en español, mapeo objeto-relacional, es una técnica utilizada para mapear sistemas orientados a objetos y bases de datos relacionales, donde las tablas de la base de datos se representan como clases y los registros de las tablas serían instancias de esas clases.

- Entender qué son los ORMs y para qué se utilizan
- Conocer el SQL y sus sistemas de gestión de bases de datos
- Trabajar con Sequelize, un ORM para usar con Node.js
- Conocer otros ORMs de Node.js, como Prisma

☐ **JavaScript - Pruebas**:

- La prueba de software es el proceso de evaluación y verificación de que un software realmente hace lo que debería hacer. Los beneficios de las pruebas incluyen la prevención de errores, la reducción de los costos de desarrollo y la mejora del rendimiento.
- Usar pruebas unitarias
- Usar pruebas de integración
- Usar pruebas de comportamiento (behavior)
- Usar mocks

☐ **JavaScript- Callbacks y Promises**:

- Una Promesa (Promises) es un proxy de un valor que no necesariamente se conoce cuando se crea la promesa. Esto permite que los métodos asíncronos devuelvan valores como los métodos síncronos- en lugar de devolver inmediatamente el valor final, el método asíncrono devuelve la promesa de proporcionar el valor en algún momento en el futuro.
- Una función de devolución de llamada (Callback) es una función que se pasa a otra función como argumento, que luego se invoca dentro de la función externa para completar algún tipo de rutina o acción.
- Una función asíncrona es una función declarada con la palabra clave asíncrona y la palabra clave espera está permitida dentro de ella. Las palabras clave async y await permiten que el comportamiento asincrónico basado en promesas se escriba en un estilo más claro, lo que evita la necesidad de configurar explícitamente cadenas de promesas.
- Comprender el concepto de programación asíncrona
- Escribir código asíncrono entendiendo el concepto de promesas en JavaScript

- Usar métodos, palabras clave y objetos de JavaScript para manejar promesas como 'Async/Await', '.then()', 'Promise', etc.
- Aprender en qué situaciones necesitas usar la programación asíncrona
- Llamar a las API con 'fetch()'

- [ ] **JavaScript - Manejo de errores**:

- El manejo de errores se refiere a los procedimientos de respuesta y recuperación de las condiciones de error presentes en una aplicación de software. En otras palabras, es el proceso compuesto por la anticipación, detección y resolución de errores de aplicación, programación o comunicación.
- Conocer y manejar las excepciones más comunes
- Conocer los tipos de errores y en qué situaciones se pueden producir
- Comprender cómo Node.js maneja los errores
- Usar 'try' y 'catch' para el manejo de errores
- Aprender en qué ocasiones y cómo usar `throw`
- Creación de excepciones específicas según las necesidades de su aplicación

- [ ] **JavaScript - Modularización**:

- Dividir partes del código en módulos
- Usar import y export

- [ ] **Jest**:

- Jest es un corredor de pruebas de JavaScript que le permite acceder al DOM a través de jsdom. Proporciona una gran velocidad de iteración combinada con funciones potentes como módulos de simulación y temporizadores para que pueda tener más control sobre cómo se ejecuta el código.
- Componentes de prueba

- [ ] **JavaScript - Almacenamiento**:

- Almacenar datos en el front-end con localStorage
- Manipular datos almacenados
- Persistir datos almacenados

- [ ] **NextJS - Fundamentos**:

- Next.js es un marco de desarrollo web de código abierto creado por Vercel que permite aplicaciones web basadas en React con representación del lado del servidor y generación de sitios web estáticos.
- Creación de interfaces web
- Disminución de los tiempos de carga de la página
- Representación de páginas del lado del servidor
- Mejorando el rendimiento en React
- Creación de rutas API con funciones sin servicio

- [ ] **Python - Comunicación con APIs**:

- Una API es una interfaz que los desarrolladores de software utilizan para programar la interacción con componentes o recursos de software fuera de su propio código. Una definición aún más simple es que una API es la parte de un componente de software que es accesible para otros componentes.
- Comprender qué es una API REST
- Conocer los comandos básicos de comunicación HTTP
- Entender qué es una API REST
- Saber cómo hacer solicitudes autenticadas
- Convertir objetos a JSON y viceversa
- Saber cómo utilizar las herramientas del paquete Requests.

- [ ] **Spring Framework**:

- Spring es un framework de código abierto para la plataforma Java. Se trata de un marco no intrusivo, basado en los estándares de diseño (design patterns) de inversión de control (ioc) e inyección de dependencia. En Spring el contenedor se encarga de "instanciar" clases de una aplicación

Java y definir las dependencias entre ellas a través de un archivo de configuración en formato XML, inferencias del framework, lo que es llamado de auto-wiring o incluso anotaciones en las clases, métodos y propiedades. De esta forma, Spring permite el bajo acoplamiento entre clases de una aplicación orientada a objetos.

- Entender el concepto de inyección de dependencias
- Entender el patrón MVC
- Usar Spring Data para manipular datos

- [ ] **Cypress**:

- Cypress es una herramienta de prueba Front-end que permite configurar, escribir, ejecutar y depurar pruebas.

Nivel 3

- [ ] **Arquitectura de Microservicios**:

- Los microservicios son un enfoque de arquitectura en el que el software consiste en pequeños servicios independientes que se comunican entre sí y se organizan de acuerdo con sus dominios de negocio.
- Aprender el concepto de arquitectura diseñada para microservicios
- Realizar la comunicación mediante API
- Mejorar la escalabilidad de un sistema

- [ ] **Node.js - Autenticación y Tokens**:

- JWT (JSON Web Token) es un método creado según el estándar RFC 7519 que representa una comunicación segura entre dos partes. Este token consta de tres partes: encabezado, carga útil y firma.
- Construir un sistema de autenticación utilizando tokens
- Comprender el funcionamiento del JSON Web Token (JWT)
- Crear una lista de permisos para almacenar tokens opacos
- Implementar métodos de actualización de tokens

- [ ] **Contenedores**:
    - Los contenedores son paquetes de software que contienen todos los elementos necesarios para ejecutarse en cualquier entorno. La gestión de contenedores es un área crucial en la computación en nube y DevOps, que implica el uso de tecnologías para automatizar el proceso de creación, implementación, escalado y monitoreo de contenedores. Los contenedores son unidades de software estandarizadas que permiten a los desarrolladores empaquetar todas las dependencias de una aplicación (código, bibliotecas, configuraciones, etc.) en un solo paquete. Esto permite que la aplicación se ejecute de forma consistente en cualquier entorno de infraestructura.
    - La tecnología de contenedores, como ejemplifica Docker, proporciona un entorno coherente y portátil para el desarrollo, las pruebas y la implementación de aplicaciones, lo que es vital para el trabajo eficiente de la ingeniería de datos. Además, Kubernetes, un sistema de organización de contenedores, permite la gestión, automatización y escalabilidad de aplicaciones basadas en contenedores en entornos de producción. Dominar estos conceptos y tecnologías permite a los ingenieros de datos construir y mantener canalizaciones de datos eficientes y confiables.
    - Kubernetes (también conocido como k8s o Kube) es una plataforma de orquestación de contenedores de código abierto que automatiza gran parte de los procesos manuales necesarios para implementar, gestionar y escalar aplicaciones en contenedores.
    - Aislar el software para que funcione independientemente
    - Implementación de software en clústeres
    - Modularizar su sistema en paquetes más pequeños
    - Conocer la plataforma Docker
    - Conocer Kubernetes
- [ ] **TypeScript - Fundamentos**:
    - TypeScript es un lenguaje de programación fuertemente tipado que se basa en JavaScript.

- Comprender en profundidad qué son los tipos y la importancia de la programación tipificada
- Aprender qué es TypeScript, por qué se creó, cómo funciona y su relación con JavaScript
- Conocer las herramientas de TypeScript (integración con el editor de código, verificador estático y compilador)
- Escribir código en TypeScript usando sus herramientas (interfaces, enumeración, decoradores, etc.)

☐ **WebSockets**:

- WebSocket es una tecnología que permite la comunicación bidireccional a través de canales full-duplex sobre un único socket del Protocolo de Control de Transmisión (TCP). Está diseñado para ser ejecutado en navegadores y servidores web que admitan HTML5, pero puede ser utilizado por cualquier cliente o servidor de aplicaciones.
- Conocer el protocolo WebSocket y su uso en la comunicación cliente-servidor
- Aprender sobre los diversos usos de WebSocket en la web
- Crear aplicaciones que utilicen WebSockets con las APIs y bibliotecas de Node.js

☐ **JavaScript - Concurrencia**:

- La programación concurrente es un paradigma de programación para construir programas que utilizan la ejecución simultánea de varias tareas computacionales interactivas, que pueden ser implementadas como programas separados o como un conjunto de hilos creados por un único programa
- Ejecutar tareas en paralelo

☐ **GraphQL**:

- GraphQL es un nuevo estándar API que proporciona una alternativa más eficiente, potente y flexible a REST. Fue desarrollado y de código abierto por

Facebook y ahora lo mantiene una gran comunidad de empresas e individuos de todo el mundo.

- Comprender cómo se utiliza GraphQL en el desarrollo de API
- Creación de API utilizando bibliotecas y marcos GraphQL

- [ ] **Apollo Client**:

- Apollo Client es una biblioteca integral de administración de estado para JavaScript que le permite administrar datos locales y remotos con GraphQL.
- Utilizar Apollo para crear un servidor GraphQL
- Conectar a una API

- [ ] **Nest.js - Fundamentos**:

- NestJS es un framework de Node con soporte total para TypeScript que se ejecuta sobre frameworks HTTP como expressJS o Fastify. Utiliza varios elementos de programación orientada a objetos y una serie de funcionalidades de TypeScript.
- Aprender qué es NestJS y por qué se utiliza
- Utilizar características específicas de NestJS, como proveedores (providers), módulos y controladores (controllers)
- Desarrollar APIs utilizando NestJS
- Utilizar la Interfaz de Línea de Comandos (CLI) de Nest.js

- [ ] **Java - Kafka**:

- Apache Kafka es una plataforma de transmisión de datos distribuida que es capaz de publicar, suscribir, almacenar y procesar flujos de registro en tiempo real. Esta plataforma está diseñada para procesar flujos de datos provenientes de diversas fuentes y entregarlos a varios clientes.
- Utilizar Kafka para la comunicación asíncrona
- Crear microservicios con Kafka
- Crear productores y consumidores
- Entender cómo usar Kafka para paralelismo y ejecución serializada

- Obtener garantías relativas al envío o entrega de los mensajes

Habilidade Auxiliar: Infraestructura y buenas prácticas

- [ ] **Git y GitHub - Fundamentos**:
  - Git es un sistema de control de versiones distribuido gratuito y de código abierto diseñado para manejar todo, desde proyectos pequeños hasta proyectos muy grandes, con rapidez y eficiencia.
  - GitHub es un servicio de hosting para el desarrollo de software y el control de versiones mediante Git.
  - Crear un repositorio
  - Clonar un repositorio
  - Comprometerse, empujar y tirar hacia y desde el repositorio
  - Revertir un commit
  - Crear de ramas y Pull requests
  - Manejar fusiones y conflictos
- [ ] **HTTP - Fundamentos**:
  - HTTP significa Protocolo de transferencia de hipertexto. La comunicación entre las computadoras cliente y los servidores web se realiza mediante el envío de solicitudes HTTP y la recepción de respuestas HTTP.
  - Comprender la diferencia entre los verbos HTTP
  - Probar solicitudes y verificar los códigos de estado en el navegador
  - Aprendiendo a hacer una solicitud HTTP en la línea de comando con WGET
  - Descargar una imagen con WGET
  - Realización de una POST
- [ ] **JSON**:
  - JSON significa Notación de objetos de JavaScript. Es un formato de texto para almacenar y transportar datos.

- Crear un objeto
- Transformar un objeto en una cadena
- Transformar una cadena en un objeto
- Manipular un objeto

- [ ] **Entrega e Integración Continuas (CI/CD)**:

- CI/CD es la abreviación de Continuous Integration/Continuous Delivery, traducido al español como "entrega e integración continuas". Se trata de una práctica de desarrollo de software que tiene como objetivo hacer más eficiente la integración de código a través de compilaciones (builds) y pruebas automatizadas.
- Automatizar la integración de código entre diversas partes del equipo se ha vuelto cada vez más importante, ya que de esta manera es posible acelerar el desarrollo y reducir el tiempo de entrega del software
- Ejecutar pruebas automatizadas de la aplicación para verificar su funcionamiento
- Realizar la entrega de actualizaciones de manera automática y segura
- Realizar pruebas de conexión y pruebas de carga para evitar que la aplicación presente problemas al ser actualizada

- [ ] **SQL - Fundamentos**:

- Conocer los comandos más comunes de SQL
- Usar SELECT para consultar una tabla
- Usar INSERT para insertar datos en una tabla
- Usar UPDATE para actualizar una tabla
- Usar DELETE para eliminar datos de una tabla
- Usar JOIN para conectar los datos de múltiples tablas
- Conocer las cláusulas (FROM, ORDER BY, etc.)

- [ ] **SOLID**:

- Solid tiene cinco principios considerados como buenas prácticas en el desarrollo de software que ayudan a los programadores a escribir los códigos más limpios, dividiendo las responsabilidades, disminuyendo los acoplamientos, facilitando la refactorización y estimulando el reaprovechamiento del código. Propuesto por Robert C. Martin, SOLID propicia el desarrollo de un código limpio, legible y comprobable.

- [ ] **Design Patterns**:

- En ingeniería de software, un "patrón de diseño" (Design Pattern en inglés) es una solución general y reutilizable para un problema que ocurre normalmente dentro de un determinado contexto de diseño de software.
- Conocer y aplicar los principales patrones de diseño.

- [ ] **Clean Architecture**:

- Clean Architecture (Arquitectura Limpia) es una forma de desarrollar software, de tal forma que solo mirando el código fuente de un programa, debes ser capaz de decir lo que el programa hace.

- [ ] **Línea de Comando - Fundamentos**:

- CLI es un programa de línea de comandos que acepta la entrada de texto para ejecutar funciones del sistema operativo.
- Conocer los comandos más importantes

- [ ] **Cloud - Fundamentos**:

- La computación en nube, o cloud computing, es la distribución de servicios informáticos a través de Internet mediante un modelo de tarificación de pago por uso. Una nube se compone de varios recursos informatizados, desde los propios ordenadores (o instancias, en terminología de nube) hasta las redes, el almacenamiento, las bases de datos y todo lo que les rodea. En otras palabras, todo lo que normalmente se necesita para montar el equivalente a una sala de servidores, o incluso un centro de datos completo, estará listo para usar, configurar y ejecutar.
- Conocer la diferencia entre IaaS, PaaS y SaaS
- Conocer los mayores proveedores de nube

- Especializarse en un proveedor específico de su preferencia

- [ ] **Diseño Orientado a Dominio - DDD**:

- El Diseño Orientado a Dominio (DDD) es un enfoque de diseño y desarrollo de software que se informa principalmente por los requisitos de negocio. Los componentes del programa (objetos, clases, matrices, etc.) indican la industria, sector o dominio empresarial en que opera el negocio.
- Modelar dominios de manera efectiva.
- Basar proyectos complejos en modelos de dominio.
- Conocer los bloques de construcción de DDD.

Habilidade Auxiliar: UX y Design

- [ ] **Figma - Fundamentos**:

- Figma es una aplicación web colaborativa para el diseño de interfaces. El conjunto de funciones de Figma se centra en la interfaz de usuario y el diseño de la experiencia del usuario, con énfasis en la colaboración en tiempo real, utilizando una variedad de herramientas de creación de prototipos y editor de gráficos vectoriales.
- Crear diseños de página y componentes

- [ ] **Diseño Responsivo**:

- Ajustar tus páginas al tamaño de pantalla del usuario
- Media queries
- Conocer el concepto de Mobile first

- [ ] **Sistemas de Diseño**:

- Un sistema de diseño es una colección de componentes reutilizables, guiados por estándares claros, que se pueden ensamblar para crear aplicaciones.
- Creación y mantenimiento de bibliotecas que se consumirán y utilizarán como estándar para construir un proyecto.

- [ ] **Sistemas de color**:
  - Definición de una paleta de colores que tenga sentido para una interfaz dada
- [ ] **Cómo usar fuentes**:
  - Elegir la fuente más adecuada para un proyecto determinado

---