COLOMBIA POTENCIA DE LA VIDA

TIC

TALENTO TECH

Bootcamp

Desarrollo Web Full Stack

Nivel Básico

David Ricardo Rivera Arbeláez

UT TALENTOTECH

cymetria | tecnalia colombia

TIC

Módulo 6
Integración Fullstack

- Intercambio de datos frontend y backend.
- Prácticas de desarrollo web full stack (SEO, Rendimiento y Seguridad).
- Despliegue a producción (Netlify, Vercel, Fly.io).

Intercambio de datos frontend y backend

En una aplicación web moderna, el frontend (la parte que el usuario ve e interactúa) y el backend (la parte que maneja la lógica del negocio y los datos) trabajan en conjunto.

La comunicación efectiva entre ambos es fundamental para crear aplicaciones dinámicas y responsivas. En este módulo, exploraremos los diferentes métodos y tecnologías utilizadas para intercambiar datos entre el frontend y el backend.

Intercambio de datos frontend y backend

Conceptos Clave:

- **API (Application Programming Interface):** Un conjunto de reglas y especificaciones que permiten que diferentes software se comuniquen entre sí.
- **Solicitud (Request):** Una petición enviada desde el frontend al backend para obtener o modificar datos.
- **Respuesta (Response):** La respuesta del backend a una solicitud, que puede incluir datos, códigos de estado y otros metadatos.

Intercambio de datos frontend y backend

Conceptos Clave:

- **Métodos HTTP:** Los verbos utilizados en las solicitudes HTTP (GET, POST, PUT, DELETE, etc.) para indicar la acción que se desea realizar.
- **Formato de datos:** El formato en el que se envían y reciben los datos (JSON, XML, etc.).

Intercambio de datos frontend y backend

Métodos comunes: AJAX (Asynchronous JavaScript and XML)

- Permite realizar solicitudes al servidor sin recargar toda la página.
- Utiliza el objeto XMLHttpRequest o bibliotecas como jQuery para enviar solicitudes y manejar respuestas.

Intercambio de datos frontend y backend

Métodos comunes: AJAX (Asynchronous JavaScript and XML), ejemplo

```
const xhr = new XMLHttpRequest();
xhr.open('GET', '/api/data');
xhr.onload = () => {
    if (xhr.status === 200) {
        const data = JSON.parse(xhr.responseText);
        // Procesar los datos recibidos
    }
};
xhr.send();
```

Intercambio de datos frontend y backend

Métodos comunes: Fetch API

- Una interfaz más moderna y prometedora para realizar solicitudes HTTP.
- Ofrece una sintaxis más limpia y promesas para manejar las respuestas asíncronas.

Intercambio de datos frontend y backend

Métodos comunes: Fetch API, ejemplo

```
fetch('/api/data')
    .then(response => response.json())
    .then(data => {
        // Procesar los datos recibidos
    })
    .catch(error => {
        // Manejar errores
    });
```

Intercambio de datos frontend y backend

Bibliotecas y frameworks

- **Axios:** Una biblioteca popular para hacer solicitudes HTTP en JavaScript.
- **Fetch:** Incluida en la mayoría de los navegadores modernos.
- **Bibliotecas de frameworks:** La mayoría de los frameworks frontend (React, Angular, Vue) proporcionan sus propias herramientas para realizar solicitudes.

Intercambio de datos frontend y backend

Formato de datos:

- **JSON (JavaScript Object Notation):** El formato más utilizado para intercambiar datos entre el frontend y el backend debido a su ligereza y facilidad de uso.
- **XML (eXtensible Markup Language):** Menos común que JSON, pero aún utilizado en algunas aplicaciones.

Intercambio de datos frontend y backend

Seguridad en el intercambio de datos:

- CORS (Cross-Origin Resource Sharing): Permite que un navegador realice solicitudes a un servidor de un dominio diferente.
- Token de autenticación: Un token que se envía en cada solicitud para identificar al usuario autenticado.
- Encriptación: Protección de los datos sensibles durante la transmisión.

Intercambio de datos frontend y backend

Ejercicio práctico (Tarea semana 7, opción 1):

Consumir una API pública: Utilizar una API pública para obtener datos de la hora de 5 ciudades diferentes y mostrarlas en una interfaz de usuario.

Prácticas de desarrollo web full stack

El desarrollo web full stack implica dominar tanto el front-end como el back-end de una aplicación web. Para crear sitios web de alta calidad, es esencial considerar tres pilares fundamentales:

- SEO (Search Engine Optimization).
- Rendimiento.
- Seguridad.

Prácticas de desarrollo web full stack

¿Qué es el SEO?

Proceso de optimizar un sitio web para mejorar su visibilidad en los resultados de búsqueda orgánicos.

Prácticas de desarrollo web full stack

Prácticas clave del SEO

Proceso de optimizar un sitio web para mejorar su visibilidad en los resultados de búsqueda orgánicos.

Prácticas de desarrollo web full stack

Prácticas clave del SEO: Optimización on-page

- Títulos descriptivos y meta descripciones.
- Uso estratégico de palabras clave.
- Estructuras de URL amigables.
- Enlaces internos relevantes.
- Contenido de calidad y original.

Prácticas de desarrollo web full stack

Prácticas clave del SEO: Optimización off-page

- Construcción de enlaces de calidad.
- Presencia en redes sociales.
- Citas y menciones en otros sitios web

Prácticas de desarrollo web full stack

Herramientas de SEO:

- Google Search Console.
- Google Analytics.
- SEMrush.
- Ahrefs

SEO – Ejercicio práctico

Análisis de un sitio web:

Evaluar el SEO, rendimiento y seguridad de un sitio web existente. Identificar áreas de mejora.

Rendimiento web

¿Por qué es importante el rendimiento?

- Mayor experiencia de usuario.
- Mejor posicionamiento en buscadores.
- Menor tasa de rebote.

Prácticas para mejorar el rendimiento web

Optimización de imágenes:

- Compresión de imágenes sin pérdida de calidad.
- Uso de formatos adecuados (WebP, AVIF).
- Carga diferida de imágenes.

- **Minificación de código:**
  - Eliminación de espacios en blanco y comentarios innecesarios en HTML, CSS y JavaScript.

Prácticas para mejorar el rendimiento web

Concatenación de archivos:

- Reducción del número de solicitudes HTTP.

Cache:

- Almacenamiento de archivos estáticos en el navegador del usuario.

Reducción del tamaño de las páginas:

- Minimización del HTML, CSS y JavaScript.
- Uso de una CDN (Content Delivery Network).

Seguridad web

Amenazas comunes en la web:

- Inyección SQL
- Cross-Site Scripting (XSS)
- Cross-Site Request Forgery (CSRF)
- Denegación de servicio (DoS)

Seguridad web

Prácticas de seguridad:

- Validación y sanitización de datos.
- Encriptación.
- Autenticación y autorización.
- Gestión de errores.
- Mantener software actualizado.

Seguridad web

Prácticas de seguridad:

- **Validación y sanitización de datos:**
  - Verificación de la entrada de datos del usuario.
  - Eliminación de caracteres especiales peligrosos.
- **Encriptación:**
  - Protección de datos sensibles durante la transmisión y el almacenamiento.

Seguridad web

Prácticas de seguridad:

- **Autenticación y autorización:**
  - Verificación de la identidad de los usuarios.
  - Control de acceso a los recursos.
- **Gestión de errores:**
  - Manejo adecuado de excepciones y errores.
  - Evitar la divulgación de información sensible.
- **Mantener software actualizado:**
  - Aplicación de parches de seguridad.

Seguridad web – Ejercicio práctico

Pruebas de vulnerabilidad

Utilizar herramientas para detectar vulnerabilidades en aplicaciones web.

Taller práctico de despliegue con Vercel

Requisitos previos:

- Una cuenta en Vercel.
- Un proyecto de Astro creado.
- Node.js y npm (o yarn) instalados.

Taller práctico de despliegue con Vercel

1. Inicialización de Vercel.
2. Conectar con un repositorio de Git.
3. Configuración (Opcional).
4. Despliegue.
5. Verificación.

Taller práctico de despliegue con Vercel

Inicialización de Vercel

Instalar la CLI de Vercel

npm install –g vercel

Iniciar el despliegue

Desde la raíz del proyecto Astro, ejecutar:

vercel

Taller práctico de despliegue con Vercel

Conectar con un repositorio de Git

- Vercel solicitará la conexión del proyecto con un repositorio Git (GitHub, GitLab, Bitbucket, etc).
- Seguir las instrucciones en pantalla para autenticarse y seleccionar el repositorio correcto.

Taller práctico de despliegue con Vercel

Configuración (Opcional)

- Personalizar el dominio (Para dominios personalizados).
- Configurar variables de entorno (Si la aplicación lo requiere).

Taller práctico de despliegue con Vercel

Despliegue

- Vercel analizará el proyecto y detectará si es una aplicación de Astro.
- Se iniciará el proceso de construcción y despliegue.
- Vercel proporciona una URL de la aplicación desplegada.

Taller práctico de despliegue con Vercel

Verificación

Acceder a la URL

Integración Fullstack

Taller práctico de despliegue con Vercel

Consideraciones adicionales

- **Construcción:** Vercel se encargará de construir tu aplicación Astro automáticamente cada vez que se detecte un nuevo commit en tu repositorio.
- **Optimizaciones:** Vercel aplica una serie de optimizaciones para mejorar el rendimiento de tu sitio, como la pre-renderización y el almacenamiento en caché.
- **Personalización:** Puedes personalizar aún más tu despliegue utilizando la configuración de Vercel, como la configuración de rutas personalizadas o la integración con otros servicios.

Taller práctico de despliegue con Vercel

Actividad semana 7

- Aplicar las prácticas aprendidas para construir un sitio web optimizado.
- Desplegar un sitio web en Vercel, utilizando Astro.
- Subir el proyecto a la carpeta de trabajo de estudiante, el repositorio de GitHub es opcional.

Recursos de apoyo

[1] Documentación oficial de Vercel para Astro.
[2] Documentación oficial de Astro para despliegue.

COLOMBIA POTENCIA DE LA VIDA

TALENTO TECH

TIC

¡Gracias!

UT TALENTOTECH

cymetria | tecnalia colombia