Nombre: Paula Andrea Taborda Montes
Fecha: 27 de Noviembre de 2023
Email: ptabordam@unal.edu.co

Preguntas conceptuales

1. **¿Qué habilidades técnicas consideras que debe tener un programador front-end?**
   R// Un programador front-end debe ser capaz de plasmar a través de la pantalla las necesidades de los usuarios, de forma accesible y usable. Es importante tener clara la definición del front-end, que se asocia directamente con la parte visual y de interfaces de usuario de una aplicación web, por tanto, un programador en esta área debe dominar las **tecnologías del lado del cliente**, como por ejemplo:

   - **HTML / CSS**: Las cuales son herramientas esenciales para darle **estructura** y **estilo** a las páginas, sitios y aplicaciones web; actualmente, ya existen muchas tecnologías que permiten de forma simple hacer diseños rápidamente sin tener en cuenta muchos detalles, por ejemplo, la creación de un navbar, la creación de una ventana emergente, la creación de un formulario, entre otros, podría darse a través de *Bootstrap* (más adelante explicado), solamente copiando y pegando el código donde sea necesario, pero desde mi punto de vista es necesario conocer la base en que están desarrollados estos elementos, la cual está implementada generalmente con HTML y CSS; existen ocasiones donde para satisfacer necesidades específicas de los proyectos, es necesario adentrarse a los estilos y modificar alguna propiedad de un selector específico, por tanto, reitero la necesidad de conocer las tecnologías desde su origen.

   - **Bootstrap**: Es un framework de código abierto que facilita enormemente el diseño y desarrollo de páginas, sitios y aplicaciones web; podría decirse que es un framework de CSS, porque provee las clases necesarias para los diseños de los diferentes elementos que se requieren ya sea en una página, sitio o aplicación web; no obstante, es necesario decir que bootstrap también utiliza scripts de JavaScript para temas de interactividad con el usuario. Bootstrap entrega elementos responsivos (adaptables a los diferentes tamaños de pantalla), dado que utiliza un sistema de rejilla (conocido como grid system), esta es una de las bondades más llamativas de esta tecnología, tiene una documentación muy completa, la cantidad de componentes que dispone al público son personalizables, entre muchas otras cualidades.

   - **JavaScript**: Es un famoso lenguaje de programación utilizado ampliamente en el desarrollo del frontend de una aplicación web (aunque a veces también es usado para implementar funciones del backend). Con este lenguaje, es posible manejar eventos, usar servicios web, y en general, para manipular el **DOM** (Document Object Model), que se refiere a la estructura del documento HTML de la aplicación; el DOM convierte el documento HTML en un modelo de objetos que puede ser manipulado. Según GitHub, JavaScript en es el lenguaje de programación más usado actualmente en el mundo.

- **Frameworks y librerías de JavaScript**: Es necesario involucrarse en el avance de la tecnología y por tanto conocer las librerías y frameworks que salen al mercado, facilitando la programación, en este caso del frontend. Por ejemplo, para JavaScript existen diferentes librerías y frameworks que definitivamente deben ser estudiados, entre los cuales se encuentran:

    - **React JS**: Es una librería de JavaScript creada por Facebook, que fue desarrollado con la necesidad de que los datos de la aplicación pueden cambiar sin necesidad de recargar la página con cada variación. De esta librería, hay algunos elementos que deben tenerse claros como los que se muestran en la **Figura 1**.

      Figura 1 - Características de React JS

      Fuente: Elaboración propia

    - **Vue JS**: Framework de JavaScript ampliamente utilizado para crear aplicaciones de una sola página, y también se destaca por la posibilidad de crear componentes reutilizables.

    - **Angular**: Framework basado en Typescript, también usado para crear aplicaciones de una sola página; Typescript es una extensión de JavaScript que incorpora funcionalidades de tipado estático al lenguaje, es decir, permite colocar los tipos de datos que se están declarando, lo que disminuye los errores.

- **Herramientas de control de versiones**: Las cuales son fundamentales en la actualidad para mantener un respaldo de los datos y además trabajar en proyectos de diversos tamaños de forma colaborativa. Algunas de las

herramientas de control de versiones que se destacan en la actualidad son **Git** y el repositorio web **GitHub**.

Además, debe contar con habilidades como:

- **Conocimientos de diseño**: Dado que el programador de front-end se va a encargar de las interfaces de usuario; va a mostrarle el cliente a través de la pantalla las capacidades de los sistemas; es realmente un trabajo fundamental. Algunas de las características (generalmente, requerimientos no funcionales) que el programador de front debería saber, son:

    - **Accesibilidad**: Se refiere al uso de buenas prácticas para que los sistemas puedan ser accedidos por todas las personas, sin que haya dificultades al tener condiciones o necesidades especiales. Por ejemplo el atributo alt en las imágenes para describir lo que se ve en una imagen, las paletas de colores para que sea posible ver hasta en situaciones de poca visión, al atributo title de diferentes etiquetas para presentar una descripción de lo que se está clickeando, entre otros.

    - **Usabilidad**: Se refiere a la facilidad con que los usuarios pueden navegar en los sistemas, la cantidad de clics que deben dar para usar ciertas funcionalidades, entre otros.

- **API Rest**: Los programadores front-end deberían conocer las API Rest (Interfaz de Programación de Aplicaciones basada en Transferencia de Estado Representacional) para establecer conexión entre el front-end y el back-end.

Considero además que el programador frontend debe tener comunicación constante con el programador backend para que no haya luego problemas de integración entre los microservicios.

2. **¿Qué son las props y cómo se utilizan en React JS?**
R// Los **props** (que viene de **propiedades**) de React JS, son el mecanismo para pasar datos desde el componente padre a los componentes hijos.

   React cuenta con una **jerarquía de componentes**; el componente padre hace referencia al componente principal o más global; cada componente puede tener uno o varios componentes hijos; recordemos que un componente es una pieza autónoma y reutilizable que encapsula parte de la interfaz de usuario. Un componente puede ser un botón, un formulario o incluso una aplicación web completa. Clara la definición de componente, los **props** entonces son propiedades o información que se mandan desde los componentes padres a los componentes hijos, como una forma de comunicación.

   No es posible que los props sean pasados desde los componentes hijos a los componentes padres, dado que los componentes hijos están diseñados para ser reutilizables en el código y si estos pudieran mandar o devolver props ya no serían tan reutilizables y acabaría con la filosofía de React JS y sus componentes.

Para responder a la parte de la pregunta de ¿Cómo se utilizan en React JS? vemos el siguiente ejemplo (genérico - sacado de Bard, IA de Google), soportado con la Figura 2.

Se tiene un componente padre llamado App (de forma general, siempre hay un nodo Padre con este nombre), el cual tiene dos componentes hijos que son **Header** y **Body**; este componente padre desea enviar una información a su componente hijo Header, esa información tiene como nombre “title”, como puede verse en la línea 5; el componente hijo Header recibe esta información como se muestra en la línea 12 y accede a ella como se muestra en la línea 15.

Figura 2 - Ejemplo de props en React JS

```
1   // Componente padre
2   const App = () => {
3       return (
4           <div>
5               <Header props={{ title: "Mi aplicación" }} />
6               <Body />
7           </div>
8       );
9   };
10
11  // Componente hijo
12  const Header = (props) => {
13      return (
14          <header>
15              <h1>{props.title}</h1>
16          </header>
17      );
18  };
19
20  // Componente hijo
21  const Body = () => {
22      return (
23          <div>
24              <p>Este es el cuerpo de la aplicación.</p>
25          </div>
26      );
27  };
28
```

Fuente: Bard - IA de Google

Otra forma de acceder a los props, mediante la sintaxis propName, se muestra en la **Figura 3**. En este caso, se envía directamente el nombre del prop y se recibe exactamente este mismo nombre.

Figura 3 - Ejemplo 2 de props en React JS

```jsx
// Componente padre
const App = () => {
    return (
        <div>
            <Header title="Mi aplicación" />
            <Body />
        </div>
    );
};

// Componente hijo
const Header = ({ title }) => {
    return (
        <header>
            <h1>{title}</h1>
        </header>
    );
};

// Componente hijo
const Body = () => {
    return (
        <div>
            <p>Este es el cuerpo de la aplicación.</p>
        </div>
    );
};
```

Fuente: Bard - IA de Google

3. **¿Cuál es la importancia del HTML DOCTYPE?**
   R// El HTML DOCTYPE es una declaración que se usa al comienzo de un documento HTML e indica al navegador web la versión de HTML que se está utilizando. Sin esta declaración el navegador web podría interpretar incorrectamente el documento de HTML, lo que claramente puede generar errores de visualización y de comportamiento.
   Algunos ejemplos propuesto por Bard del uso de HTML DOCTYPE, son:

   <!DOCTYPE html> indica la versión 5 de HTML
   <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd"> indica la versión 4.01 de HTML

   Es recomendable usar el HTML DOCTYPE más reciente para que sea compatible con los navegadores de la actualidad.

   El uso del **HTML DOCTYPE** al principio del documento permite también la validación del documento, es decir, luego otros programas o extensiones que validan las

páginas, sitios o aplicaciones web pueden verificar si el documento de HTML cumple con las reglas sintácticas y estructuras establecidas por los estándares.

El HTML DOCTYPE es fundamental para determinar el modo de renderizado en una aplicación; existen dos modos de renderizado:

Modo Estándar: Cuando el navegador encuentra la declaración de HTML DOCTYPE en el principio de un documento HTML, entra en modo estándar, donde se interpreta el código de acuerdo a los estándares definidos por la W3C, por lo que sigue las reglas más recientes.
Modo Quirks: Si por el contrario, no se encuentra la declaración de HTML DOCTYPE, el navegador entra en modo quirks, por lo que interpreta el código de una manera más permisiva y compatible para versiones antiguas, lo que sin duda, puede llevar a problemas de presentación.

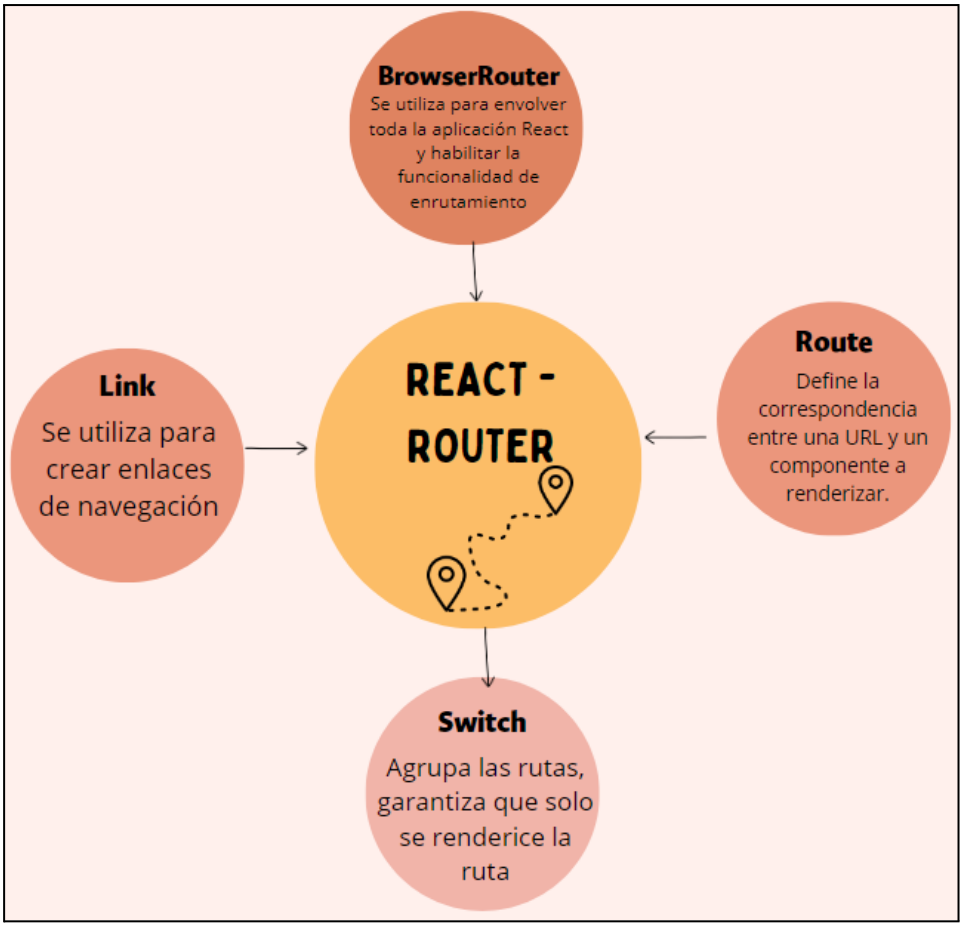
Para tener visualizaciones correctas de las interfaces de usuario y que sigan los estándares de W3C, se deben usar la declaración HTML DOCTYPE al principio del documento HTML.

4. **¿Qué es React Router y cómo se utiliza en React JS?**

Ayudándome de la definición brindada por Bard (IA de Google), puedo decir que el React Router es una **librería** de enrutamiento construida sobre React que permite a los desarrolladores de React crear aplicaciones web de una sola página (SPA) con **navegación dinámica**. Debemos tener en cuenta que en una aplicación **SPA (Single Page Application)**, toda la interfaz de usuario se carga en una sola página. Cuando el usuario navega a una **nueva ruta**, la aplicación no carga una nueva página web, sino que simplemente cambia el contenido de la página actual (renderiza los componentes de las diferentes vistas).

Para entender React Router, se deben entender algunos conceptos como los que se muestran en la **Figura 4**.

Figura 4 - Conceptos alrededor de React Router



Fuente: Elaboración propia

Para responder a la parte de la pregunta ¿Cómo se utiliza en React JS?, se muestra el siguiente ejemplo, basado en ChatGPT.

Se tienen dos componentes para representar la página de Inicio y de Contacto de una empresa, como se muestra en la Figura 5.

Figura 5 - Componentes para página de Inicio y Contacto

```
// src/components/Inicio.js
import React from 'react';

const Inicio = () => {
  return <h2>Página de Inicio</h2>;
};

export default Inicio;
```

```jsx
// src/components/Contacto.js
import React from 'react';

const Contacto = () => {
  return <h2>Página de Contacto</h2>;
};

export default Contacto;
```

Fuente: ChatGPT 3.5

Ahora, se debe configurar el React Router en el componente principal App.js, así como se presenta en la **Figura 6.**

Figura 6 - Configuración del enrutamiento en App.js (componente principal)

```jsx
// src/App.js
import React from 'react';
import { BrowserRouter as Router, Route, Link } from 'react-router-dom';
import Inicio from './components/Inicio';
import Contacto from './components/Contacto';

const App = () => {
  return (
    <Router>
      <div>
        {/* Enlaces de navegación */}
        <nav>
          <ul>
            <li><Link to="/">Inicio</Link></li>
            <li><Link to="/contacto">Contacto</Link></li>
          </ul>
        </nav>
```

```
        {/* Definición de rutas */}
        <Route path="/" exact component={Inicio} />
        <Route path="/contacto" component={Contacto} />
      </div>
    </Router>
  );
};

export default App;
```

Fuente: ChatGPT 3.5

En el ejemplo anterior, **<Router> … </Router>** envuelve toda la aplicación, proporcionando la funcionalidad de enrutamiento; **<Link> … </Link>** se utiliza para crear enlaces de navegación a las páginas de inicio y contacto; y **<Route … />** define las rutas y especifica qué componente se renderizará para cada ruta.

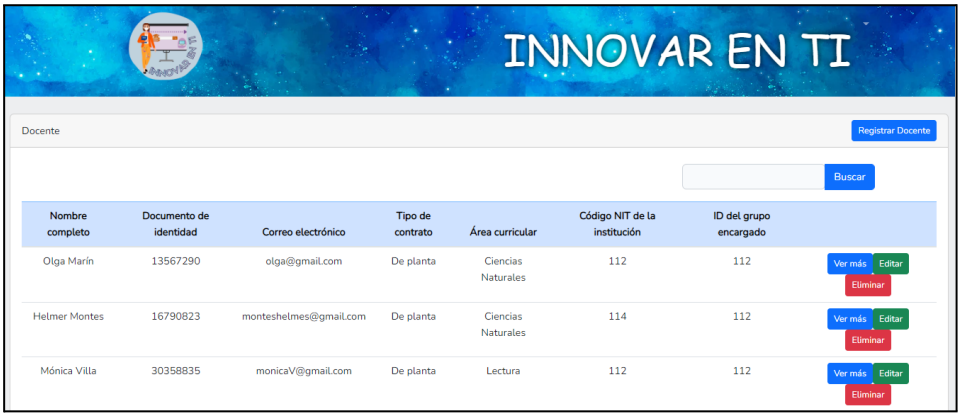
5. **¿Te has enfrentado a retos como Desarrollador front-end? Menciona algunos**

Sí, me he enfrentado a dos retos interesantes como desarrolladora front-end (y también back-end en una de los dos experiencias) en los que conocí acerca de las tecnologías que mencioné en la respuesta a la pregunta 1 de este cuestionario y me pareció muy llamativo el mundo del front. Esos retos, son:

1. **Proyecto final Ingeniería del Software II:** En esta asignatura debimos entregar un proyecto de desarrollo de software como trabajo final, el cual debía tener un front-end y un back-end completo y satisficiera las necesidades de una empresa ficticia creada también en una de las asignaturas de esta carrera (INNOVAR EN TI - Empresa dedicada al desarrollo y venta de recursos educativos digitales de las diferentes materias básicas ofrecidas por las instituciones educativas, y también para la identificación y atención de dificultades específicas de aprendizaje); el software fue desarrollado en Laravel, por lo que para el front se utilizaron principalmentes los **Blade Templates de Laravel**, **Bootstrap** y **CSS**. El sistema luce como se muestra en las Figuras 7 y 8.

   Figura 7 - Vista de la página principal

Figura 8 - Página de administrar docente



2. **Trabajo de grado**: El semestre actual (2023-02) estoy desarrollando mi trabajo de grado (**Esmeralda**), en el cual construí recursos educativos digitales para la detección temprana y atención inicial de la dislexia en estudiantes de grado primero; estos recursos fueron integrados en un entorno informático educativo desarrollado en React JS; con esta experiencia incursioné en el mundo de React JS y en sus infinitas posibilidades. En las Figuras 9 y 10, se muestran algunos pantallazos del entorno. Este se encuentra disponible en: https://gaia.manizales.unal.edu.co/PiedrasPreciosasPrototipo/esmeralda/inicio

Figura 9 - Página inicial de Esmeralda

Fuente: Esmeralda

Figura 10 - Vista inicial del Test de detección temprana de dislexia fonológica

Fuente: Esmeralda