DOCUMENTO DE BUENAS PRÁCTICAS

Este documento tiene como objetivo establecer un conjunto de buenas prácticas para el desarrollo de la aplicación **TANGO APP**, utilizando **Ionic Angular** en el frontend y **Express.js** en el backend. Con estas pautas, buscamos mejorar el rendimiento, la seguridad y la facilidad de mantenimiento de la aplicación, así como hacer que el proceso de desarrollo sea más eficiente y ordenado.

1. Organización modular y componentes reutilizables

- **Dividir el proyecto en partes pequeñas y funcionales**: Estructura la aplicación en módulos y componentes separados según su función, para que cada parte haga solo una cosa, y lo haga bien. Esto hace que el proyecto sea más fácil de entender y mantener a largo plazo.
- **Crear componentes reutilizables**: En lugar de repetir código, crea componentes que puedas reutilizar en diferentes partes de la aplicación. Esto no solo mejora la eficiencia, sino que también facilita la actualización de funcionalidades en el futuro.

2. Estructura clara para estados y funciones

- **Asegurar de que los estados, variables y funciones estén colocados en los lugares más lógicos dentro del código:** Ten en cuenta qué parte del código se encarga de cada función y cómo se relaciona con el resto. Es importante que el código sea fácil de leer y comprender, de modo que cualquiera pueda seguir la lógica y encontrar rápidamente dónde se maneja cada aspecto. Esta claridad en la estructura ayuda a mantener el proyecto ordenado y facilitar futuras modificaciones.

3. Mejora del rendimiento de la interfaz de usuario

- **Optimizar el desplazamiento en listas grandes**: Si la aplicación muestra grandes volúmenes de datos, como listas largas, asegúrate de usar técnicas que hagan más fluido el desplazamiento. Esto mejorará la experiencia del usuario, especialmente en dispositivos móviles.

4. Estructura clara del proyecto

- **Organizar las carpetas de forma lógica**: Coloca archivos relacionados en una misma carpeta y usa nombres claros que describan su función, para que

cualquier desarrollador pueda entender cómo está organizada la aplicación sin tener que preguntar.
- **Nombres consistentes**: Asegúrate de que los nombres de los archivos y carpetas sigan un patrón que refleje su función en el código. Esto hará que el proyecto sea más fácil de navegar y entender.

5. Usar solo lo necesario

- **Elegir bibliotecas livianas**: No agregues más herramientas de las que realmente necesitas. Evalúa cada biblioteca que uses para que no sobrecargue el proyecto con código innecesario. Además, busca versiones más ligeras de herramientas comunes.
- **Evita redundancias en el código**: Siempre que notes que estás repitiendo el mismo código en varias partes de la aplicación, trata de simplificarlo o reutilizarlo. Menos código repetido significa menos errores y una aplicación más fácil de mantener.

6. Carga de imágenes y recursos de manera eficiente

- **Imágenes optimizadas**: Asegúrate de que las imágenes se carguen en formatos modernos para mejorar la velocidad de la aplicación, sobre todo en dispositivos móviles, y que solo se carguen cuando realmente sean necesarias.

7. Almacenamiento seguro

- **Proteger la información sensible**: Evita almacenar datos sensibles como contraseñas o tokens de autenticación en lugares inseguros. Usa opciones de almacenamiento más seguras que protejan la información de posibles ataques.

8. Usar compresión para mejorar la velocidad

- **Comprimir las respuestas**: Al comprimir los datos antes de enviarlos al usuario, puedes reducir el tiempo de carga y mejorar el rendimiento, especialmente en redes lentas.
- Ejemplo: Typescript

```typescript
import express from 'express';
import compression from 'compression';

const app = express();
app.use(compression());
```

9. Evitar bloquear el proceso

- **No usar funciones que bloqueen el servidor**: Siempre que sea posible, usa funciones que trabajen de manera asíncrona para evitar que el servidor se detenga o ralentice al manejar varias solicitudes al mismo tiempo.

10. Registrar la actividad del servidor de manera eficiente

- **Llevar un registro organizado de lo que pasa en la aplicación**: En lugar de usar herramientas básicas como console.log() para registrar eventos, usa bibliotecas especializadas que hagan el trabajo de forma más eficiente y ordenada, lo que facilitará la depuración y el monitoreo.

11. Manejo de errores controlado

- **Controlar los errores adecuadamente**: Asegúrate de que, si ocurre un error, tu aplicación no se detenga por completo. Utiliza técnicas para manejar las excepciones y evitar que los errores inesperados afecten la experiencia del usuario.
- Ejemplo: Typescript

```
import express, { Request, Response, NextFunction } from 'express';

const app = express();

app.use((err: Error, req: Request, res: Response, next: NextFunction)
=> {
    console.error(err.stack);
    res.status(500).send('Something broke!');
});
```

12. Usar un Proxy Inverso

- **Servidor intermedio** que se sitúa entre los clientes y los servidores de backend.
- Esta configuración mejora el rendimiento, la seguridad y la escalabilidad de tu aplicación al distribuir el tráfico de manera más efectiva.

FUENTES/ BIBLIOGRAFÍA:

https://expressjs.com/en/advanced/best-practice-performance.html
https://ionic.io/enterprise-guide/angular