Listado de tecnologías utilizadas:

- Tecnologías WEB: HTML, CSS, Javascript.
- Framework Bootstrap
- Framework Angular

Buenas Prácticas Desarrollo Web:

Fuente: Mejores prácticas de desarrollo web (diego.com.es)

1.Comentarios y documentación

Los comentarios y la documentación son fundamentales en cualquier proyecto, ya que además de servir para otros desarrolladores que tengan que continuarlo, te sirve a tí mismo para recordar las funcionalidades y retornos de las cosas. Los IDE's (Integrated Development Environment) han permitido que los comentarios en el código sean todavía más útiles. Siguiendo ciertos estándares en los comentarios los IDE's y otras herramientas pueden proporcionar información de forma directa, ahorrando mucho tiempo.

2.Indentación consistente

Se ha de emplear una indentación consistente, que no cambie durante el proyecto. Generalmente en la actualidad se suele seguir el modelo marcado por PSR-2, pero no hay ninguna obligación en hacerlo. Un ejemplo resumen de la forma de indentar según el PSR-2

3.Evitar comentarios obvios

Comentar el código es fundamental, pero hacerlo demasiado tampoco es bueno. Hay que llegar a un nivel medio en el que no se comente cada línea del código sino cada vez que sea realmente necesario para entender el proceso.

4.Agrupar el código

Muchas tareas requieren más de una línea de código. Es bueno mantener estas tareas en bloques separados de código, separados por una línea. Incluir un comentario al comienzo de cada bloque también enfatiza esta separación.

5.Esquema de nombrado consistente

PHP no es el ejemplo ideal del nombrado consistente de funciones (por ejemplo, strpos(), _strsplit()). Una normal fundamental es que los nombres deben tener separación entre las palabras, ya sea con un nombrado camelCase o con barras bajas _getall, y que mantengan el formato en el proyecto.

6.Principio DRY

DRY significa Don't Repeat Yourself, también llamado DIE, Duplication Is Evil. El principio dice que "cada funcionalidad debe tener una representación singular, sin ambigüedades y autoritaria".
El objetivo de la mayoría de aplicaciones (y de la computación en general) es automatizar tareas repetitivas. La misma pieza de código no debe repetirse continuamente.
Por ejemplo, si una página web consiste en varias páginas, es muy posible que estas tengan elementos comunes (header, navigation, footer, etc). Lo ideal es que sólo existan en un archivo concreto y se incluyan de unos a otros por herencia, como ocurre en Twig.

7.Evita los niveles excesivos

Demasiados niveles con estructuras de control (como utilizar demasiados if / else) empeoran la legibilidad del código.

8.Limita la longitud de línea

Nuestros ojos leen mejor columnas de texto estrechas que texto demasiado amplio (por eso los periódicos tienden hacia ese formato). Es una buena práctica evitar escribir horizontalmente grandes líneas de código.

9.Organiza bien archivos y directorios

Técnicamente es posible escribir una aplicación entera en un simple archivo, pero eso sería muy difícil de leer y de mantener.
Una buena organización en diferentes archivos y directorios mejora mucho la calidad del código. Lo ideal es emplear frameworks como Symfony que estructuran el código con estándares.

10.Nombres temporales consistentes

Normalmente las variables han de ser descriptivas y contener una o más palabras. Pero eso no se aplica necesariamente en variables temporales, ya que pueden ser tan cortas como de un sólo carácter.

11.Pon en mayúsculas las palabras especiales en SQL

Las interacciones con la base de datos es una parte importante en las aplicaciones web. Si escribes SQL, es importante que también sea legible. A pesar de que las palabras especiales y funciones de SQL no son sensibles a mayúsculas y minúsculas, es una práctica común poner en mayúscula las palabras especiales (SELECT, FROM, UPDATE, SET, WHERE, LEFT JOIN, GROUP, ORDER BY, LIMIT...).

12.Separa el código de los datos

En el caso del desarrollo web los datos suelen ser el output o salida HTML. Cuando PHP comenzó hace algunos años, era visto principalmente como un motor de templates. Era usual tener archivos HTML grandes con algunas líneas de PHP. Con los años esto ha cambiado y los sitios han evolucionado a construcciones más dinámicas y funcionales. El código es ahora una parte enorme de las aplicaciones web, y no es una buena práctica mezclarlo con el lenguaje de marcado HTML.
Esta práctica actualmente es sencilla gracias a los frameworks como Symfony y a los motores de templates como Twig.

13.Programación orientada a objetos

La Programación Orientada a Objetos permite crear código bien estructurado. La mayoría de los frameworks y aplicaciones web ya aplican esta forma de construcción en prácticamente la totalidad de su código.

14.Lee código open source

El código open source de frameworks y proyectos ya construidos son ejemplos de cómo se ha de construir proyectos. Los desarrolladores respetan una serie de reglas comunes que hacen que el código sea legible, consistente y bien estructurado. Actualmente la mayoría respetan las normas establecidas en el FIG.

15.Haz refactoring

Cuando se refactoriza el código se hacen cambios pero no se cambia su funcionamiento. Se hace "limpieza" para mejorar la legibilidad y la calidad del código. Lo ideal es hacerlo lo más pronto y frecuente posible.

16.Debugging

Los desarrolladores tienden a escribir el código completo y después empezar a depurar (debug) y comprobar errores. Aunque esta forma de hacerlo puede ahorrar tiempo en proyectos pequeños, en proyectos de mayor envergadura tienden a tener demasiadas variables y funciones que requieren de atención específica. Por esta razón es mejor depurar cada módulo cuando se termina y no el proyecto entero. Esto ahorra tiempo a la larga, ya que no hay que buscar dónde concretamente se está produciendo el error. Para evitar que en un futuro aparezcan bugs que sean difíciles de encontrar, se emplea el testing, y más concretamente, el Test Driven Development.

17.Testing

El testing es una parte integral del desarrollo web que ha de ser planificada. Consiste en la comprobación de la funcionalidad de los módulos de un proyecto o del proyecto en sí. Las comprobaciones se pueden centrar en la seguridad de la aplicación, la funcionalidad, la accesibilidad en función del rango de los usuarios, en el análisis del rendimiento (load testing), etc.

Prácticas responsive

Fuente: 7 Buenas prácticas para un Diseño Web Responsive (somoscafeina.com)

1. Menú de navegación oculto.

Las pantallas más pequeñas que ocultan el menú de navegación principal es una buena manera de mantener los diseños simples. Un icono, texto o combinación de ambos indica al usuario dónde está el menú.

Tus opciones incluyen un simple menú desplegable, donde el menú se desliza hacia abajo y cubre el contenido principal o el método de superposición donde el menú se expande y cubre toda la pantalla.

2. Menú de desplazamiento horizontal.

Otra manera es dejar el menú visible en pantallas pequeñas pero dejar el contenido del menú fuera del borde de la pantalla donde se encuentra la información. Dejar parte del texto del menú cortado indica que se puede desplazar hacia un lado, así el ejemplo del menú de Google en pantallas móviles al buscar un tema es especifico.

3. Ubicar botones y links largos, áreas “clickeables”.

En lugar de hacer botones más pequeños en los móvil, hazlos más grandes, por lo que son más fáciles de aprovechar. De hecho, esto no sólo se aplica a las pequeñas pantallas, es bueno para cualquier medio donde se vean los sitios web.

Los botones grandes mejoran la usabilidad. A la vez que tienes botones más grandes, también debes tener enlaces de texto más grandes. Si, por ejemplo, tiene una rejilla de titulares de noticias, con un enlace de texto que dice "Leer más" dentro de cada uno de ellos; en lugar de hacer esto, puedes convertir en link todo el resumen del contenido, así todo pasa a ser clickeable.

4. Balance entre el peso de la fuente y el tamaño.

La relación entre el tamaño de los encabezados y el texto del párrafo debe estar bien equilibrados. Los encabezados grandes no se ven bien en el móvil, sobre todo si se extienden sobre unas pocas líneas. Todo debe cambiar de tamaño adecuadamente.

Los dispositivos móviles nuevos tienen pantallas de alta resolución, lo que hace que un texto sea más legible y más fácil de leer. Puedes permitirte el lujo de ir un poco más pequeño en las pantallas móviles y aumentar los tamaños de fuente cuando se llega a una pantalla más grande.

5. Anchuras de lectura óptima.

Al hacer un diseño más amplio en las pantallas más grandes, es importante tener en cuenta las longitudes de las líneas de tu contenido.

Si una línea de texto es demasiado larga, es más difícil de leer porque es difícil seguir la línea a línea. Del mismo modo, si se tienen líneas que son demasiado cortas rompe el ritmo de la lectura ya que los ojos tienen que moverse hacia atrás y hacia adelante con demasiada frecuencia.

La práctica común es mantener las longitudes de línea en alrededor de 60-75 caracteres. Esto se puede lograr mediante el establecimiento de sus áreas de texto para tener un máximo de anchura de aproximadamente 500/700 píxeles de ancho.

6. Poner la información importante en la parte superior.

Muestra números de teléfono, información de contacto, los CTAs, etc. en la parte superior en el móvil. Los usuarios de móviles quieren la información de forma rápida, pero esto también funciona bien en cualquier dispositivo. Por ejemplo, en una página de detalles de productos de comercio electrónico es bueno tener el botón "Añadir a la cesta" visible para la mayoría de los usuarios, sin que tengan que desplazarse.

7. Cambiar el orden de los bloques de contenido cuando se colapsan en pantallas pequeñas.

Decide lo que es más importante para mostrar por primera vez en una pequeña pantalla, a continuación, cambia el orden de contenido.

Si en una pantalla de escritorio hay un bloque de contenidos de texto y un bloque de imagen situado uno junto al otro, decide lo que es más relevante y eso se mostrará a primera vista.

Buenas Prácticas de Angular

Fuente: https://www.ideas2it.com/blogs/angular-development-best-practices/

1. Utilice Angular CLI

Angular CLI es una herramienta de interfaz de línea de comandos que se utiliza para inicializar, desarrollar, andamiaje, mantener e incluso probar y depurar aplicaciones Angular. Es por eso que en lugar de crear los archivos y carpetas manualmente, use Angular CLI para generar nuevos componentes, directivas, módulos, servicios y tuberías.

2.Estructura de carpetas

Crear una estructura de carpetas es un punto muy importante que debe ser considerado antes de iniciar su proyecto. Nuestra estructura de carpetas debe adaptarse fácilmente a los nuevos cambios que surgen durante el desarrollo.

3. Seguir estilos de codificación Angular consistentes

- Limite los archivos a 400 líneas de código.
- Definir funciones pequeñas y limitarlas a no más de 75 líneas
- Tener nombres consistentes para todos los símbolos. El patrón recomendado es feature.type.ts.
- Si los valores de las variables están intactos, declararlo con 'const'.

- Usar guiones para separar palabras en el nombre descriptivo y use puntos para separar el nombre descriptivo del tipo.
- Los nombres de propiedades y métodos siempre deben estar en minúsculas camel case.
- Dejar siempre una línea vacía entre importaciones y módulos, como importaciones de aplicaciones y de terceros y módulos de terceros y módulos personalizados.

4. Uso de TypeScript

Los beneficios de usar TypeScript son:

- Soporte para clases y módulos.
- Verificación de tipos.
- Acceso a las funciones de ES6 y ES7, antes de que sean compatibles con los principales navegadores.
- Soporte para empaquetado de Javascript.
- Excelente soporte de herramientas con IntelliSense

6. Utilice trackBy junto con ngFor

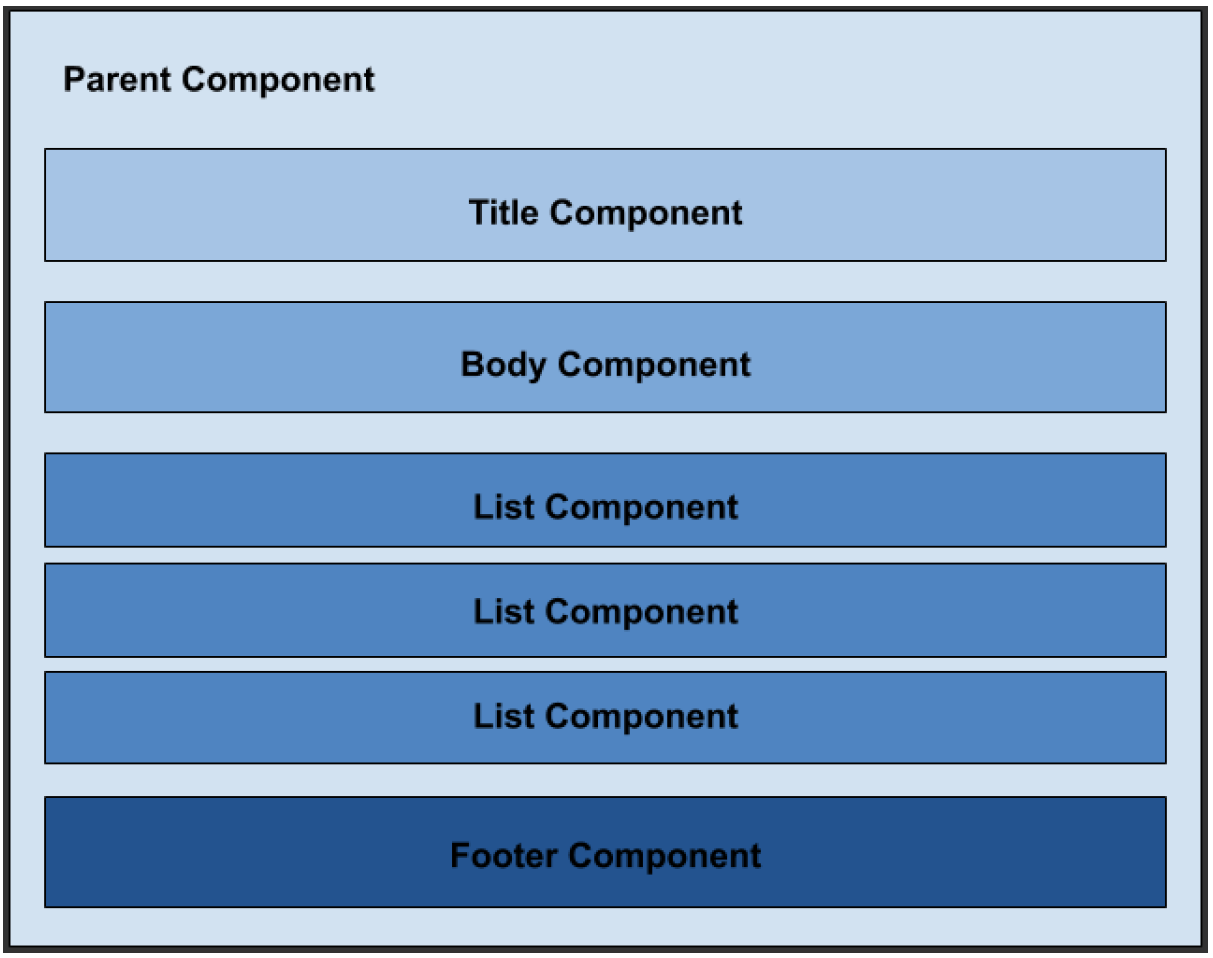
Cuando use ngFor para recorrer una matriz , usela con una función trackBy que devolverá un identificador único para cada elemento DOM. Cuando una matriz cambia, Angular vuelve a renderizar todo el árbol DOM. Pero cuando usa trackBy, Angular sabrá qué elemento ha cambiado y solo hará cambios DOM solo para ese elemento

5. Utilizar las funciones de ES6

- Funciones de flecha.
- Interpolación de cadenas.
- Literales de objeto.
- Let y Const.
- Desestructuración.
- Predeterminado.

7. Dividir en pequeños componentes reutilizables

Si el componente se vuelve grande, dividirlo en componentes más pequeños reutilizables para reducir la duplicación del código, de modo que podamos administrarlo, mantenerlo y repararlo fácilmente con menos esfuerzo.



8-Utilizar la carga diferida

Cargar los módulos en una aplicación angular siempre que sea posible.Esto reducirá el tamaño del tiempo inicial de carga de la aplicación y mejorará el tiempo de inicio de la aplicación al no cargar los módulos no utilizados

9.Utilice Index.ts

index.ts nos ayuda a mantener todas las cosas relacionadas juntas para que no tengamos que preocuparnos por el nombre del archivo fuente. Esto ayuda a reducir el tamaño de la declaración de importación.

10. Evitar la lógica en las plantillas

Toda la lógica de la plantilla se extraerá en un componente. Lo que ayuda a cubrir ese caso en una prueba unitaria y reduce los errores cuando hay un cambio de plantilla

11.Caché de llamadas a la API

Agregar un mecanismo de almacenamiento en caché y almacenar el valor de una API. Cuando se realiza otra solicitud a la misma API, obtenemos una respuesta de la

verificación, si no hay ningún valor disponible en la caché, hacemos una llamada a la API y almacenamos el resultado

13. Declarar cadenas seguras

La variable de tipo cadena tiene solo un conjunto de valores y podemos declarar la lista de valores posibles como el tipo. Entonces, la variable aceptará solo los valores posibles. Podemos evitar errores al escribir el código durante el tiempo de compilación.

14. Evite cualquier tipo

Declare variables o constantes con tipos distintos.Esto reducirá los problemas no deseados. Otra ventaja de tener buenos mecanografiados en nuestra aplicación es que facilita la refactorización

15. Gestión del estado

La administración del estado en Angular ayuda a administrar las transiciones de estado al almacenar el estado de cualquier forma de datos. En el mercado existen varias librerías de administración de estado para Angular como NGRX , NGXS , Akita , etc. y todas tienen diferentes usos y propósitos.

Algunos beneficios son:

- Permite compartir datos entre diferentes componentes
- Proporciona control centralizado para la transición de estado
- El código será limpio y más legible
- Facilita la depuración cuando algo sale mal
- Hay herramientas de desarrollo disponibles para rastrear y depurar el estado bibliotecas de gestión

16. Utilice CDK Virtual Scroll

La carga de cientos de elementos puede resultar lenta en cualquier navegador. Pero el soporte de desplazamiento virtual de CDK muestra grandes listas de elementos de manera más eficiente. El desplazamiento virtual permite una forma eficaz de simular todos los elementos que se renderizan haciendo que la altura del elemento contenedor sea la misma que la altura del número total de elementos que se van a renderizar, y luego solo renderizando los elementos a la vista.

17. Utilice variables de entorno

Angular proporciona configuraciones de entorno para declarar variables únicas para cada entorno. Los entornos predeterminados son desarrollo y producción. Incluso podemos agregar más entornos o agregar nuevas variables en archivos de entorno existentes.

18. Use reglas para Typecript y SCSS

TSLint y Stylelint tienen varias opciones integradas, lo que obliga al programa a ser más limpio y consistente. Es ampliamente compatible con todos los editores modernos y se puede personalizar con sus propias reglas y configuraciones de pelusa. Esto garantizará la coherencia y la legibilidad del código

19.Documentar siempre

Documente siempre el código tanto como sea posible. Ayudará al nuevo desarrollador involucrado en un proyecto a comprender su lógica y legibilidad.Es una buena práctica documentar cada variable y método. Para los métodos, necesitamos definirlo usando comentarios de varias líneas sobre la tarea que realmente realiza el método y se deben explicar todos los parámetros.

Buenas Prácticas Bootstrap 5

Fuente: https://getbootstrap.com/docs/4.0/getting-started/best-practices/

- Trabajar con CSS
- Trabajar con archivos Sass
- Construyendo nuevos componentes CSS
- Trabajar con flexbox
- Preguntar en Slack (mensajería)