

Análisis Estático de Código Fuente con...



Indice

- Análisis estático de código fuente.
- SonarQube.
- Demo.
- Cierre.

Análisis Estático de Código Fuente

Es un tipo de análisis de software que se realiza sin ejecutar el programa.

Análisis Estático de Código Fuente

Beneficios:

- Ayuda a la detección temprana de bugs, vulnerabilidades, code smells.
- Posibilidad de identificar errores antes de desplegar aplicación.
- Ayuda a reducir la deuda técnica.
- Ayuda a que el código mas mantenible.
- Mejora la calidad de código.

Análisis Estático de Código Fuente

- El análisis estático de código no reemplaza a los tests.
- Lo ideal sería hacer ambos.

- Hay que tener en cuenta que los tests hay que programarlos.
- Significa escribir mas código.
- Significa tickets para crear tests.

FIXING UNIT TESTS



Los desarrolladores
buscan la calidad
interna (mejor
código fuente) y los
gestores la calidad
externa (agregar
funcionalidad).



Steve McConnell, 2007

http://www.construx.com/10x_Software_Development/Technical_Debt/

“No hay tiempo / presupuesto”



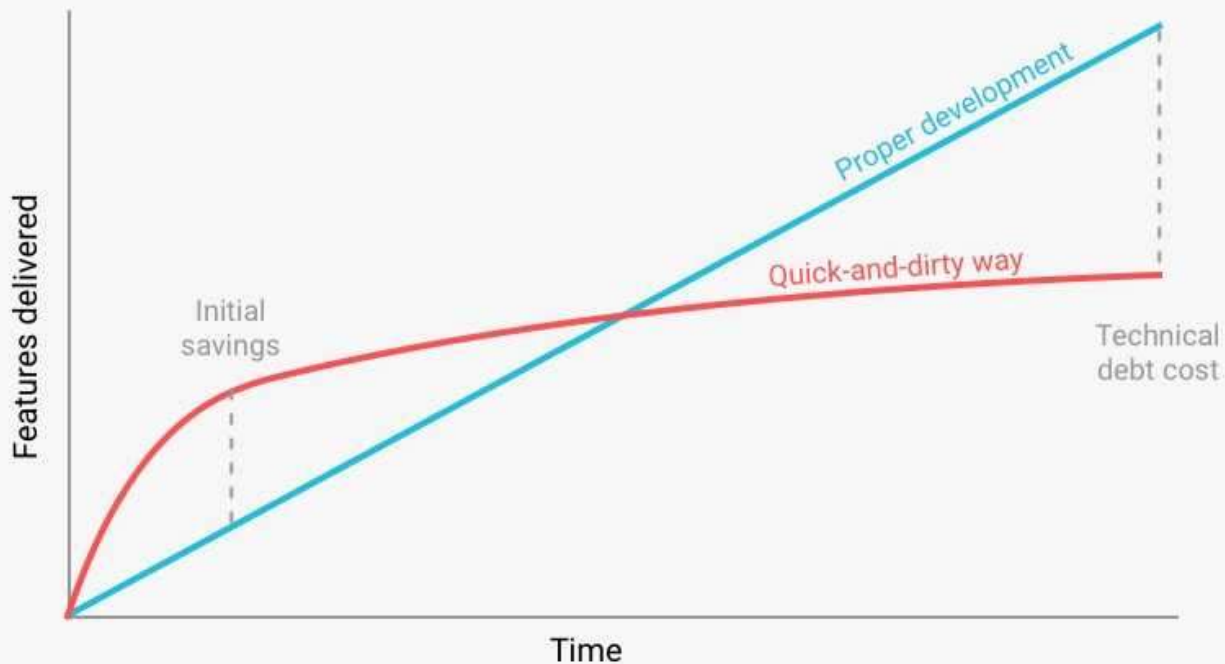
Y tiene sentido...

El software
evoluciona con el
tiempo
haciéndose más
complejo, y su
calidad declina.



Lehman, M. M., Ramil, J. F., Wernick, P. D., Perry, D. E., & Turski, W. M. Metrics and laws of software evolution-the nineties view. In Software Metrics Symposium, 1997.

La deuda técnica impacta a la hora de crear nuevas features.



Conclusión

- Nosotros somos los responsables de reducir la deuda técnica.
- El análisis estático de software nos puede ayudar.

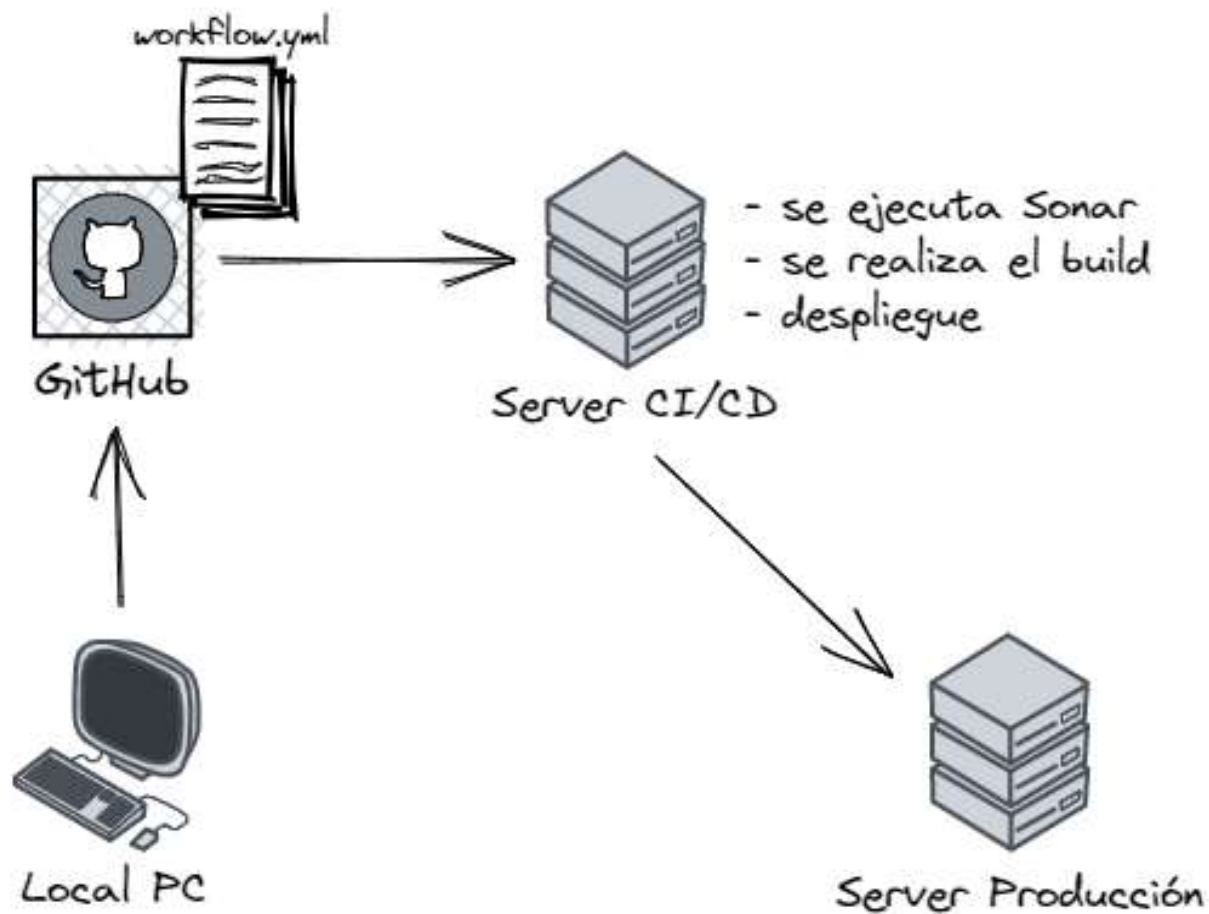
Ok. ¿Y Sonar?



- Herramienta para realizar análisis estático de código fuente.
- Nos sirve para obtener métricas que pueden ayudar a mejorar la calidad del código.
- Permite a los equipos de desarrollo hacer seguimiento y detectar errores y vulnerabilidades de seguridad para mantener el código limpio.
- Incluye soporte para más de 20 lenguajes de programación.
- Detecta principalmente tres tipos de problemas:
 - Bugs: error de codificación que rompe el código.
 - Vulnerabilidad: puntos de código abiertos a ataques o hacking.
 - CodeSmells: fragmentos de código confuso que lo hacen difícil de mantener.



Integración con entornos de CI/CD (Jenkins, TravisCI, AWS Pipelines, Github Actions, etc).
Por ejemplo, si estamos usando un pipeline en Jenkins tendríamos una etapa de inspección, sirve de filtro para, si pasa el Quality Gate (o Umbrales de Calidad), continúa con los siguientes pasos, o, por el contrario, si no lo pasa, notificar del error y no se continua hasta que se corrija.



Quality Gate predeterminado de SonarQube

Coverage: Indica el porcentaje de código que está cubierto por pruebas unitarias. SonarQube establece en el Quality Gate predeterminado que la cobertura debe ser superior al 80%

Duplications: indica la cantidad de líneas que se repiten, en comparación con el total de líneas del proyecto. En este caso, se establece que este valor sea inferior al 3%.

Bugs: esta calificación se obtiene en relación a la severidad más alta de los bugs detectados. Para esta métrica, se determina que la calificación debe ser A, lo que significa que no haya ningún bug.

Vulnerabilities: se refiere también a la severidad más alta, pero de las vulnerabilidades encontradas. También debe ser A, lo que indica que no existiría ninguna vulnerabilidad en el código nuevo.

Hotspots Reviewed: son los fragmentos de código sensible a la seguridad que el desarrollador debe revisar. La principal diferencia entre un hotspot y una vulnerabilidad es la necesidad de una revisión antes de decidir si aplicar una corrección.

Code Smells: se mide el porcentaje de deuda técnica que existe en el código. La calificación debe ser A, qué vendría a ser que el porcentaje de deuda técnica es menor o igual al 5%.

| | | | | | | |
|---|---|---|---|---|---|---|
|  Bugs |  Vulnerabilities |  Hotspots Reviewed |  Code Smells | Coverage | Duplications | Lines |
| 0  | 0  | 0.0%  | 2  | 0.0%  | 0.0%  | 497  TypeScript |

Ejemplos de utilización de SonarQube

Dos formas para utilizar SonarQube son:

- Desde SonarCloud (en este caso el código analizado es compartido en su plataforma a menos que se pague la versión premium).
- Instalar SonarQube en un servidor propio.

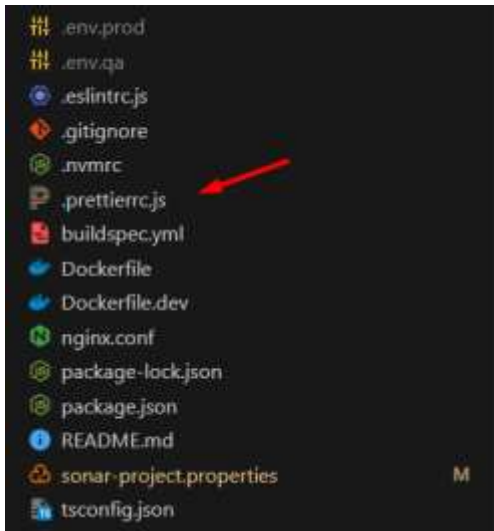
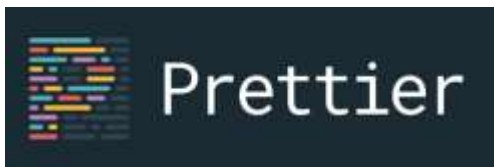


Claramente no estamos por instalar SonarQube

Que **SÍ** podemos hacer entonces...

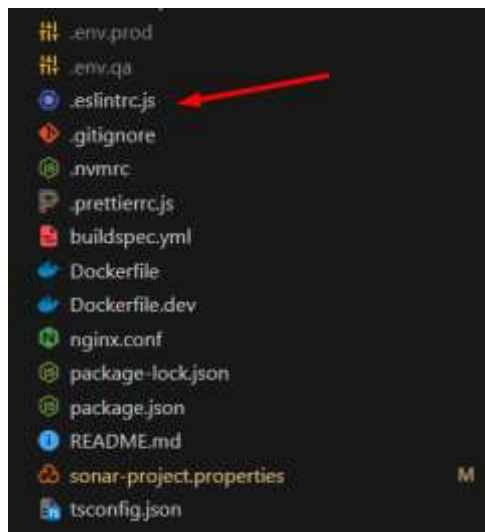


Usar formateadores de Código.



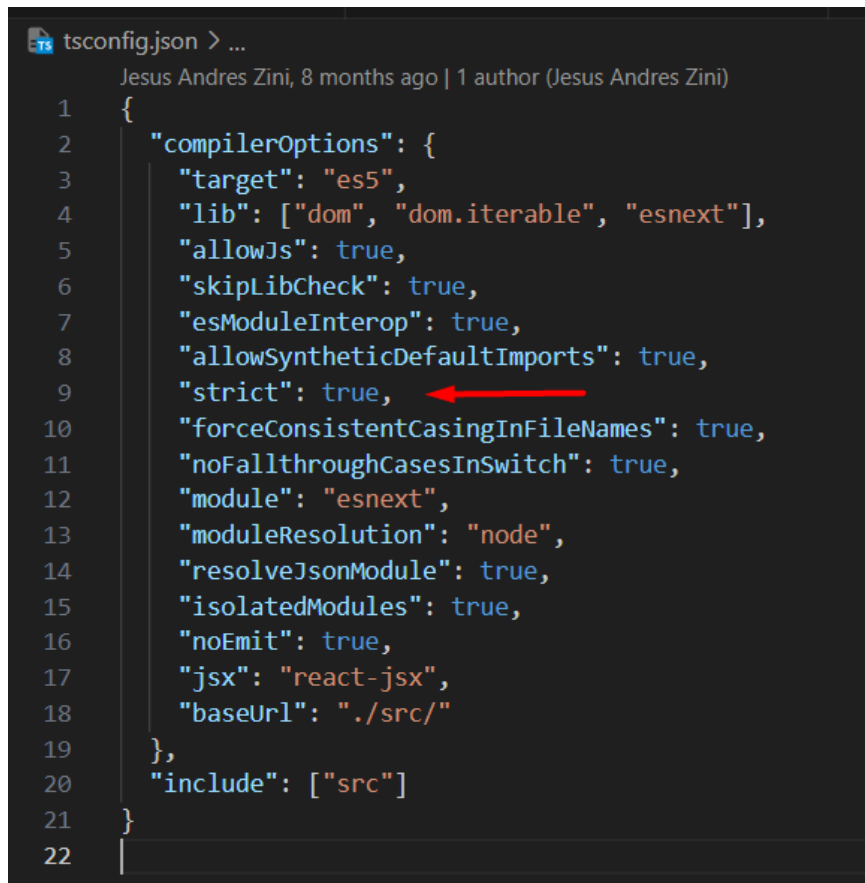
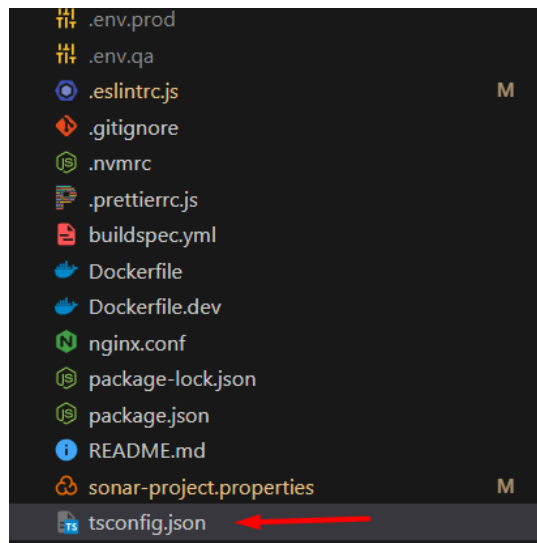
```
.prettierrc.js > ...  
Jesus Andres Zini, 3 months ago | 1 author  
1  module.exports = {  
2    bracketSpacing: true,  
3    jsxBracketSameLine: true,  
4    singleQuote: true,  
5    trailingComma: 'all',  
6    arrowParens: 'avoid',  
7    tabWidth: 2,  
8    semi: true,  
9    // printWidth: 120,  
10   endOfLine: 'auto',  
11 };  
12
```

Usar linters

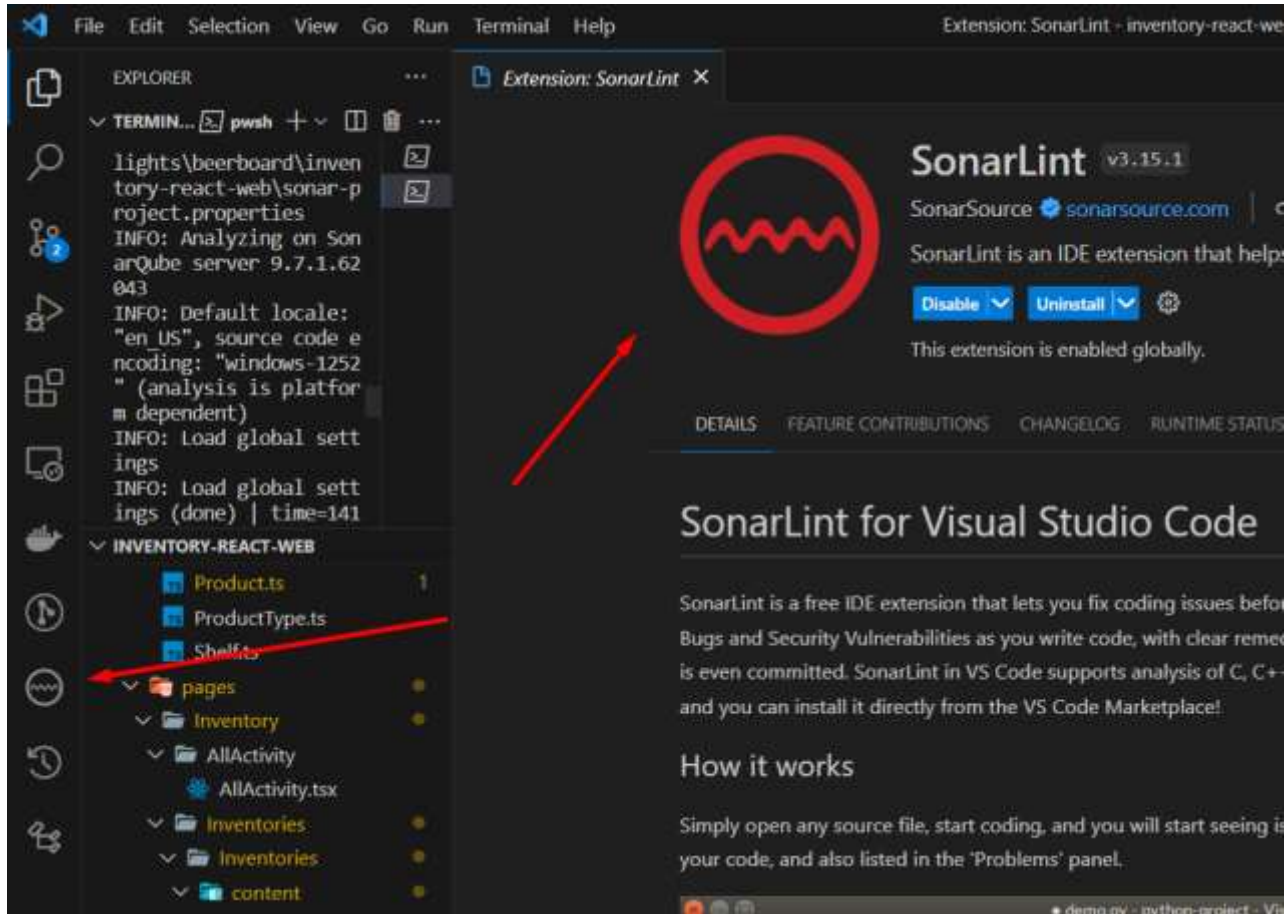


```
.eslintrc.js > ...  
You, 1 minute ago | 2 authors (Jesus Andres Zini and others)  
1 module.exports = {  
2   parser: '@typescript-eslint/parser',  
3   plugins: ['@typescript-eslint/eslint-plugin', 'prettier'],  
4   extends: ['react-app', 'plugin:@typescript-eslint/recommended', 'prettier'],  
5   rules: {  
6     'prettier/prettier': [  
7       'off',  
8       {  
9         endOfLine: 'auto',  
10      }],  
11    ],  
12    eqeqeq: 'error',  
13    'no-empty-pattern': 'off',  
14    'jsx-ally/alt-text': 'error',  
15    'array-callback-return': 'error',  
16    'jsx-ally/anchor-is-valid': 'error',  
17    'jsx-ally/iframe-has-title': 'error',  
18    'react/jsx-no-target-blank': 'error',  
19    'jsx-ally/img-redundant-alt': 'error',  
20    'react-hooks/exhaustive-deps': 'off',  
21    '@typescript-eslint/ban-types': 'off',  
22    '@typescript-eslint/ban-ts-comment': 'off',  
23    '@typescript-eslint/no-explicit-any': 'off',  
24    '@typescript-eslint/no-unused-vars': 'error',  
25    '@typescript-eslint/no-empty-function': 'off',  
26    '@typescript-eslint/no-use-before-define': 'off', // Not Recommended on Type  
27    '@typescript-eslint/no-useless-constructor': 'error',  
28    '@typescript-eslint/no-non-null-assertion': 'error',  
29    '@typescript-eslint/consistent-type-assertions': 'error',  
30    '@typescript-eslint/explicit-module-boundary-types': 'off',  
31    'no-console': 1,  
32  },  
33  };
```

Usar TypeScript



Usar sonarlint



The screenshot displays the Visual Studio Code interface with the SonarLint extension installed. The left sidebar shows the Explorer view with a file tree for 'INVENTORY-REACT-WEB'. A red arrow points to the SonarLint icon in the sidebar. The main editor area shows the 'Extension: SonarLint' page, which includes the SonarLint logo, version information (v3.15.1), and buttons to 'Disable' or 'Uninstall' the extension. The terminal window on the left shows the output of a command, including information about the SonarQube server and the analysis process.

File Edit Selection View Go Run Terminal Help

Extension: SonarLint - inventory-react-we

EXPLORER

TERMIN... pwsh

lights\beerboard\inventory-react-web\sonar-project.properties

INFO: Analyzing on SonarQube server 9.7.1.62 043

INFO: Default locale: "en_US", source code encoding: "windows-1252" (analysis is platform dependent)

INFO: Load global settings

INFO: Load global settings (done) | time=141

INVENTORY-REACT-WEB

- Product.ts
- ProductType.ts
- Shelf.ts
- pages
 - Inventory
 - AllActivity
 - AllActivity.tsx
 - Inventories
 - Inventories
 - content

SonarLint v3.15.1

SonarSource sonarsource.com

SonarLint is an IDE extension that helps you fix coding issues before they are even committed.

[Disable](#) [Uninstall](#) [Settings](#)

This extension is enabled globally.

[DETAILS](#) [FEATURE CONTRIBUTIONS](#) [CHANGELOG](#) [RUNTIME STATUS](#)

SonarLint for Visual Studio Code

SonarLint is a free IDE extension that lets you fix coding issues before Bugs and Security Vulnerabilities as you write code, with clear remediation is even committed. SonarLint in VS Code supports analysis of C, C++ and you can install it directly from the VS Code Marketplace!

How it works

Simply open any source file, start coding, and you will start seeing issues in your code, and also listed in the 'Problems' panel.

```
</Box>
) : errorMore || errorEffect ? (
  <Box py={3}>
    <ErrorMsg top={false} />
  </Box>
) : filteredData && filteredData.items.length === 0 ? (
  <Box py={3}>
    <NoItemsMsg
      message="There are no current inventories"
      top={false}
    />
  </Box>
) : filteredData ? (
  filteredData?.items.map((inventory, i) => (
    <React.Fragment key={`_${inventory.id}-${i}`}>
      <InventoryItem
        inventory={inventory}
        handleView={handleView}
        handleOpenContextMenu={handleOpenContextMenu}
      />
    {filteredData?.items.length - 10 === i && (
      <Waypoint
        onEnter={() => handleFetchMore()}
        scrollableAncestor="window"
      />
    )}
  ))
)
```

No array index for keys in JSX list components (typescript:S6479)

Code Smell Major

React expects a unique identifier for performance optimizations. An array index is an identifier most of the time. This results in unnecessary renders when the array item index following some mutation. When components have state, this might also produce bugs that are hard to diagnose.

We recommend using an explicit identifier to avoid misuse and accidental re-rendering. If no unique attribute is available, consider concatenating existing properties - hashing is necessary - or creating a dedicated unique identifier.

Noncompliant Code Example

```
function generateButtons(props) {
  return props.buttons.map((button, index) => {
    <Button key={index}>{button.number}</Button>
  });
}
```

Compliant Solution

```
function generateButtons(props) {
  return props.buttons.map((button, index) => {
    <Button key={button.number}>{button.number}</Button>
  });
}
```


Fin! Muchas Gracias!! Felicidades!

