UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO
FACULTAD DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA, INFORMÁTICA Y MECÁNICA
ESCUELA PROFESIONAL DE INGENIERÍA INFORMÁTICA Y DE SISTEMAS

Code Smells

Asignatura: Ing. de Software I

Docente: Roxana Lisette Quintanilla Portugal

Integrantes:

- 170429 CONDORI LOPEZ, Juan Carlos
- 174442 ESCOBEDO MESCCO, Angie
- 171258 ESPEJO FRANCO, Melissa
- 170432 GUTIERREZ DAZA, Gonzalo
- 150394 HUAMAN GUEVARA, Alexander Javier
- 171915 NINANTAY DIAZ, Mileydy
- 171570 RAMOS ALVAREZ, Edgar
- 171805 ROJAS SOTO, Claudia Luz

Cusco - Perú
2021

Code Smells

¿Qué son los Code Smells?

Síntomas en el código que ciertos procesos no se hacen de manera correcta y que pueden crear problemas a futuro. Por lo general no son problemas de programación, no son técnicamente incorrectos y quizás el programa funciona correctamente, sin embargo si indican deficiencia en el diseño con un desarrollo más lento, aumentando el riesgo de errores y fallos en el futuro.

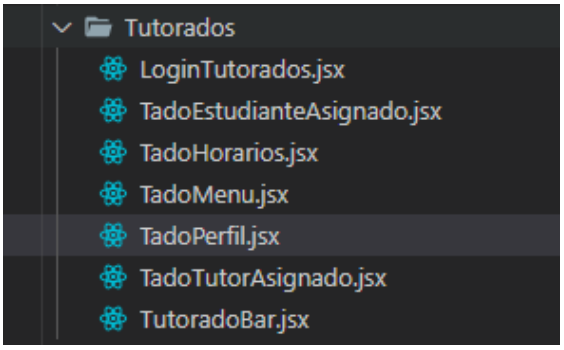
JavaScript no es un lenguaje orientado a objetos del todo, no fue diseñado para serlo, la noción de clases no le es aplicable en absoluto. Si bien todo en JS es de hecho un objeto, estos objetos son diferentes de los de Java o C #.

Programación reactiva, o Reactive Programming, es un paradigma enfocado en el trabajo con flujos de datos finitos o infinitos de manera asíncrona, permitiendo que estos datos se propaguen generando cambios en la aplicación, es decir, "reaccionan" a los datos ejecutando una serie de eventos.

Es así que no consideramos los siguientes code smells, al no utilizar un paradigma orientado a objetos :

- ***God Class* .-** Clase larga y compleja que centraliza la inteligencia del sistema.
- ***Data class* .-** Clase que contiene datos pero no comportamiento relacionado con los datos
- ***Feature envy* .-** Método que llama a más métodos de una sola clase externa que los métodos internos de su propia clase interna.
- ***Refused bequest* .-** Subclase que no usa los métodos protegidos de su superclase.
- Tradition breaker .- Subclase que proporciona un gran conjunto de servicios que no están relacionados con los servicios proporcionados por la superclase.

- **Nombres inapropiados.-** Ciertas archivos o variables podrían estar mal nombradas, no haciendo referencia a su propósito o no brindando la ayuda para entender el código.



- **Comentarios.-** Para el entendimiento del código es necesario tener comentado que hace cada parte del código, así si en algún momento es necesario que alguien externo al manejo del código necesita realizar alguna modificación o mejora se le haga fácil.

```
import express from 'express'
import config from './config'
import estudiantesRoutes from './routes/Estudiantes.routes'
import docentesRoutes from './routes/Docentes.routes'
const cors=require('cors');
//usamos el framework express para la creacion del servidor
const app=express();
//Cors para la comunicacion entre front y back
app.use(cors());
//settings
//definir el puerto dentro de app
app.set('port',config.port);

//middlewares
app.use(express.json());//para poder recibir json desde el cliente
app.use(express.urlencoded({extended:false}));//para poder recibir

//port
//usamos todas las rutas de la api para estudiantes,docente
app.use(estudiantesRoutes);
app.use(docentesRoutes);
```

- **Código muerto.-** Cuando los requisitos del software han cambiado o se han realizado correcciones, nadie ha tenido tiempo de limpiar el código antiguo.

Código no usado

```
src > Administracion > AdminDocentes.jsx > AdminDocentes > useEffect() callback
133             setExcel(d);
134         })
135     }
136     const abrirCerrarModalInsertar=()=>{
137         setModalInsertar(!modalInsertar);
138       }
139       const abrirCerrarModalWarning=()=>{
140         setWarningview(!warningView);
141       }
142     //const guardarExcel=()=>{
143      //   console.log(excel)
144        // document.getElementById('inputGroupFile04').value ='';
145     //}
146     useEffect(()=>{
147         peticionGet();
148       })
```

Componente: Hacer Tutor

```
import React from 'react'
import AdminBar from '../Administracion/AdminBar'
import {Col,Row} from 'react-bootstrap'
import '../styles/AdminHacerTutor.css'
const AdminHacerTutor = () => {
    return (
        <div>
            <AdminBar/>
            <div className="contenido">
                <div className="Principal2">
                    <div className="cont">
                        <h5>Lista de docentes :</h5>
                        <div className="TablaHacerTutor">
                            <div className="col tableScrollHacerTutor scrollHacerTutor">
                                    <table className="table table-bordered bg-light ">
                                        <thead>
                                            <tr>
                                                <th>Nro</th>
                                                <th>Curso</th>
                                                <th>Estudiante</th>
                                                <th>Detalles</th>

                                            </tr>
                                        </thead>
                                        <tbody>
                                            <tr>
```

- **Código duplicado.-** El código duplicado es un código que se repite en diferentes lugares haciendo lo mismo. Los fragmentos de código que son muy similares también pueden considerarse duplicados. En general las consultas que se realizan son muy similares para los CRUD de docente y estudiante, haciendo un code smell.

El método agregar estudiante

src > controllers > Estudiantes.controllers.js > getEstudianteById

```
48  export const addEstudiantes=async (req,res)=>{
49      try{
50          const Lista=req.body;
51          let queriestemp='';
52          for (let i = 0; i < Lista.length; i++) {
53              let CodEstudiante=Lista[i].CodEstudiante,Nombres=Lista[i].Nombres,
54              Email=Lista[i].Email,Direccion=Lista[i].Direccion,Celular=Lista[i]
55              queriestemp+="Insert into TEstudiante Values ('"+CodEstudiante+"'
56          }
57          console.log(queriestemp);
58          const pool=await getConnection();
59          const result=await pool.request().query(queriestemp);
60          console.log('addEstudiantes executed')
61          res.json(result.recordset);
62      }catch(error){
63          res.status(500);
64          res.send(error.message);}
65  };
```

El método agregar docentes

src > controllers > Docentes.crontrollers.js > addDocentes > CodDocente

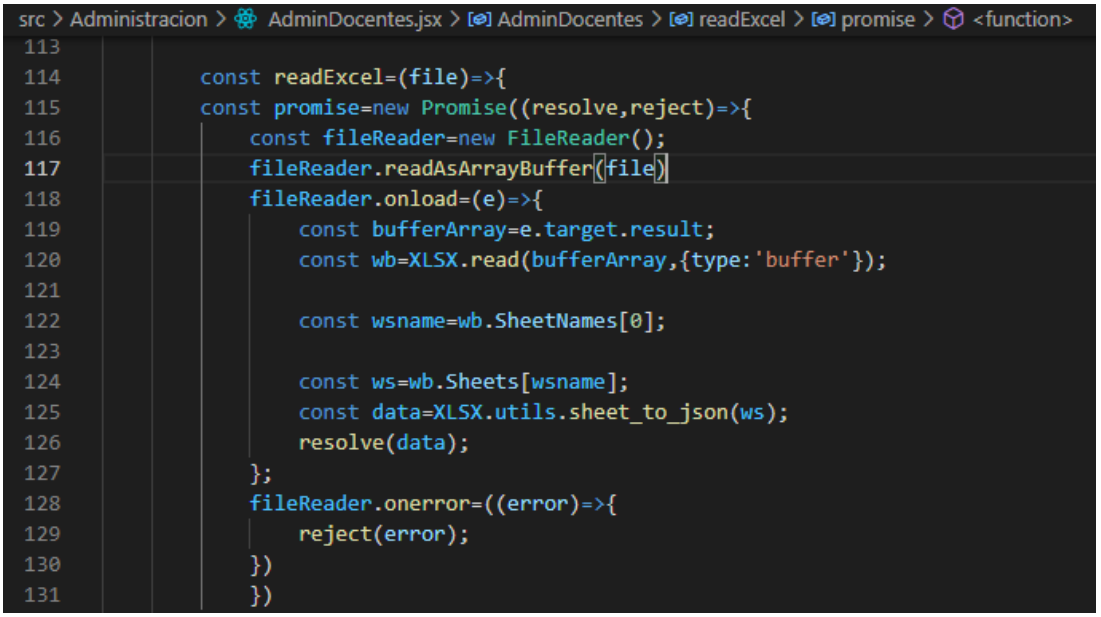
```
62  export const addDocentes=async (req,res)=>{
63      try{
64          const Lista=req.body;
65          let queriestemp='';
66          for (let i = 0; i < Lista.length; i++) {
67              let CodDocente=Lista[i].CodDocente,Nombres=Lista[i].Nombres,ApPater
68              DNI=Lista[i].DNI,Categoria=Lista[i].Categoria,Celular=Lista[i].Celu
69              EsTutor=Lista[i].EsTutor;
70              queriestemp+="Insert into TDocente Values ('"+CodDocente+"','"+Nomb
71          }
72          console.log(queriestemp);
73          const pool=await getConnection();
74          const result=await pool.request().query(queriestemp);
75          console.log('addDocentes executed')
76          res.json(result.recordset);
77      }catch(error){
78          res.status(500);
79          res.send(error.message);}
80  };
```

- **Código largo.-** Los programadores generalmente encuentran mentalmente menos agotador colocar una nueva característica en una clase existente que crear una nueva clase para la característica.

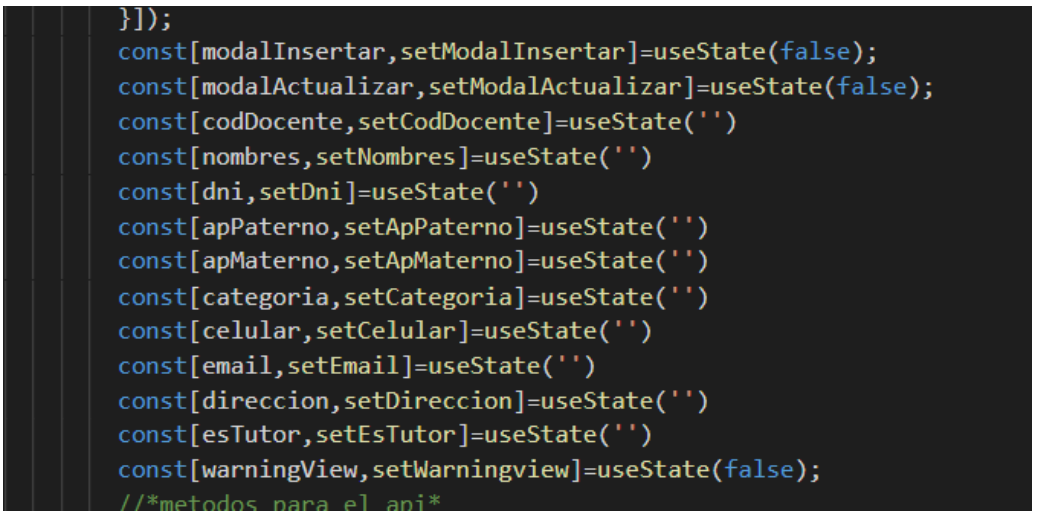
src > Administracion > AdminDocentes.jsx > AdminDocentes

```
64      const peticionGet=async()=>{
65          await axios.get(baseUrl)
66          .then(response=>{
67            setData(response.data);
68
69          }).catch(error=>{
70            console.log(error);
71          })
72        }
73        const peticionPostExcel=async()=>{
74          await axios.post(baseUrlExcel,excel)
75          .then(response=>{
76          setData(data.concat(response.data));
77          limpiar();
78          document.getElementById('inputGroupFile04').value ='';
79          }).catch(error=>{
80          console.log(error);
81          })
82        }
83        const peticionPost=async()=>{
84          if(!codDocente.trim()||!nombres.trim()||!dni.trim()||!apPaterno.trim()||!ap
85              setWarningview(true)
86              return
87          }
```

- **Deep Indentation.-** Se evita la legibilidad del código.



- **Uso de muchos useState():**
  Un componente con muchos useState() hooks probablemente esté haciendo demasiadas cosas y probablemente sea un buen candidato para dividirse en múltiples componentes, pero también hay algunos casos complejos en los que necesitamos administrar algún estado complejo en un solo componente.



```
}]);
const[modalInsertar,setModalInsertar]=useState(false);
const[modalActualizar,setModalActualizar]=useState(false);
const[codDocente,setCodDocente]=useState('')
const[nombres,setNombres]=useState('')
const[dni,setDni]=useState('')
const[apPaterno,setApPaterno]=useState('')
const[apMaterno,setApMaterno]=useState('')
const[categoria,setCategoria]=useState('')
const[celular,setCelular]=useState('')
const[email,setEmail]=useState('')
const[direccion,setDireccion]=useState('')
const[esTutor,setEsTutor]=useState('')
const[warningView,setWarningview]=useState(false);
//*metodos para el api*
```

Deuda Técnica: Trabajo que se adquiere al producir código pobre, incumpliendo prácticas aconsejadas para el desarrollo de software

- **<u>Documentación escasa, incompleta o inservible</u>**
  En el proceso de la formación del nuevo grupo, se junto la documentación planteada por ambos grupos y se realizó un estudio evaluando qué criterios tomar para continuar el proyecto, esto nos llevó a tener documentación de más que al final se tuvo que desechar, así mismo se hizo la adecuación de requisitos para su cumplimiento y la actualización de ciertos archivos.

- **<u>Arquitectura no escalable</u>**
  El diseño del sistema que estamos desarrollando no cuenta con una arquitectura preestablecida para el desarrollo en general, es así que no consideramos que permita hacer el mantenimiento adecuado a los componentes de manera adecuada, ralentizando la inclusión futura de nuevas funcionalidades al software.

- **Ausencia o deficiente control de versiones.**

- **Rigidez para actualizar a nuevas tecnologías o plataformas.**

<u>¿Por qué hemos caído en estos Code smells?</u>

➔ Falta de experiencia en la utilización de frameworks de desarrollo web.
➔ Malos hábitos de programación
➔ Inexistencia de una arquitectura de software específica

Referencias

- https://medium.com/oceanize-geeks/code-smells-and-refactoring-c2c0e0642582
- https://refactoring.guru/es/refactoring/smells
- https://antongunnarsson.com/react-component-code-smells/
- https://hackernoon.com/lessons-learned-common-react-code-smells-and-how-to-avoid-them-f253eb9696a4
- https://betterprogramming.pub/looking-for-code-smells-in-javascript-677f1a312f29