**SERVICIO NACIONAL DE APRENDIZAJE (SENA)**

**CRISTIAN RODRIGO PRADA CASTRO**

**ANALISIS Y DESARROLLO DE SOFTWARE.**
**CUNDINAMARCA**
**BOGOTA**
**2024**

# ESTÁNDARES DE CODIFICACIÓN PARA EL DESARROLLO DE SOFTWARE

**CRISTIAN RODRIGO PRADA CASTRO**

**Formulación de las actividades de aprendizaje**

**ANALISIS Y DESARROLLO DE SOFTWARE**
**CUNDINAMARCA**
**BOGOTA**
**2024**

## OBJETIVO GENERAL:

- Establecer un conjunto de estándares de codificación para el desarrollo de la aplicación Ware Innobig S.O.S, que garantice una estructura de código coherente, modular y fácil de mantener, cumpliendo con los principios del paradigma orientado a objetos.

## OBJETIVOS ESPECÍFICOS:

- Revisar el material de formación, hacer análisis de la información presentada y de acuerdo con ello realizar las evidencias y trabajos correspondientes.

- Seguir las normas básicas de presentación de un documento escrito.

- Realizar una serie de conclusiones al finalizar

## Introducción

En este documento vamos a detallar las normas de codificación que seguiremos para desarrollar la plataforma Ware Innobig S.O.S, un sistema pensado para facilitar la gestión de empleados, elaboración, traslados, ventas y reportes de resultados. La idea es que cada parte del código esté bien organizada y sea fácil de entender y modificar. Todo esto se basa en los principios de la programación orientada a objetos, lo que asegura que la aplicación sea modular y esté lista para adaptarse en el futuro.

## ESTÁNDARES DE CODIFICACIÓN

1. **Nombramiento de Variables**

- **Reglas de Estilo:**

    - Nombres descriptivos y en inglés (ejemplo: userName, userAge).
    - Notación camelCase para variables (userName, userEmail).
    - Uso de prefijos para identificar el alcance o tipo, si es necesario (strUserName, intUserAge).

- **Ejemplo de Aplicación:**

```
let userName = "John Prada";
let userAge = 30;
```

2. **Declaración de Clases**

- **Reglas de Estilo:**

    - Clases nombradas en PascalCase (ejemplo: User, ProductManager).
    - Cada clase debe representar un objeto del sistema con atributos y métodos claramente definidos.

- **Ejemplo de Aplicación:**

```
class User {
    constructor(userName, userAge) {
        this.userName = userName;
        this.userAge = userAge;
    }

    displayUser() {
        return `${this.userName}, ${this.userAge} years old.`;
    }
}
```

3. **Declaración de Métodos**

- **Reglas de Estilo:**

    - Métodos nombrados en camelCase (createUser, deleteProduct).
    - Describir claramente la funcionalidad en el nombre del método.

- **Ejemplo de Aplicación:**

```
class User {
   constructor(userName, userPassword) {
      this.userName = userName;
      this.userPassword = userPassword;
   }

   validatePassword(inputPassword) {
      return this.userPassword === inputPassword;
   }
}
```

4. **Organización del Código**

- **Reglas de Estilo:**

    - Una clase por archivo para modularidad (User.js, Product.js).
    - Agrupación de métodos relacionados dentro de las clases.
    - Uso de comentarios para describir bloques de código y funciones críticas.

5. **Manejo de Constantes**

- **Reglas de Estilo:**

    - Constantes nombradas en mayúsculas con guiones bajos (MAX_USERS, DEFAULT_COLOR).

- **Ejemplo de Aplicación:**

```
CcMAX_USERS = 1000;
```

**APLICACIÓN EN EL PROYECTO**

**Página de Inicio**

- Código modular, con el logo y botones encapsulados en clases individuales.

**Ejemplo:**

```
class HomePage {
   render() {
      return `
         <div class="home">
            <img src="logo.png" alt="Company Logo">
            <button onclick="navigateToLogin()">Login</button>
```

```
            <button onclick="navigateToSignup()">Sign Up</button>
        </div>
    `;
  }
}
```

**Inicio de Sesión**

- Métodos para validar credenciales y mostrar mensajes de error.
- **Ejemplo:**

```
class Login {
  constructor(usersDB) {
    this.usersDB = usersDB;
  }

  authenticate(userName, password) {
    const user = this.usersDB.find(user => user.userName === userName);
    if (user && user.password === password) {
      return "Welcome!";
    }
    return "Invalid credentials.";
  }
}
```

**Creación de Usuario**

- Validación de entradas y manejo de errores.
- **Ejemplo:**

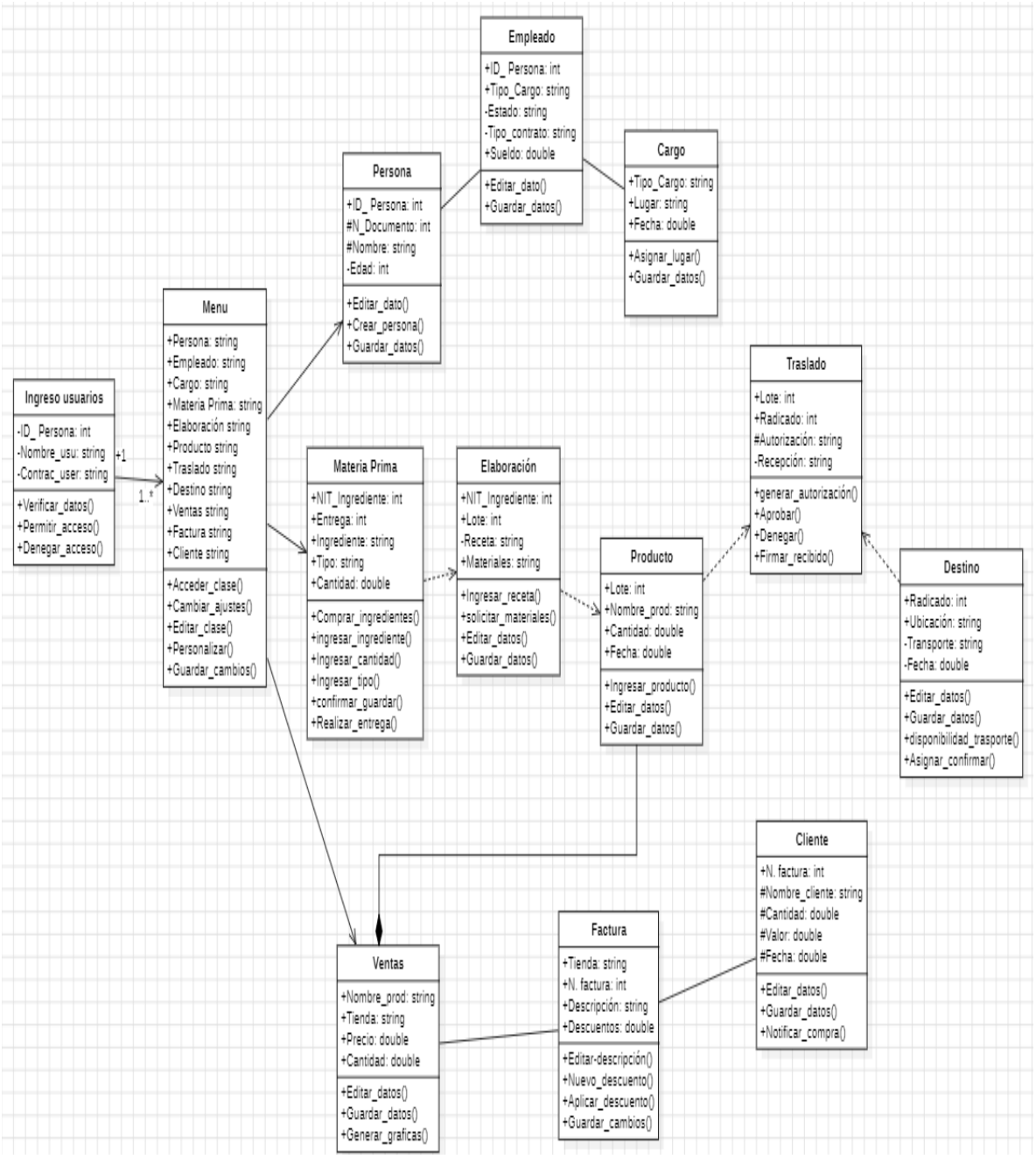
```
class UserRegistration {
  constructor() {
    this.users = [];
  }

  addUser(userDetails) {
    if (this.validateUser(userDetails)) {
      this.users.push(userDetails);
      return "User successfully registered.";
    }
    return "Invalid user data.";
  }

  validateUser({ userName, email }) {
    return userName.length > 0 && email.includes("@");  } }
```

## CONCLUSIÓN

Los estándares de codificación son fundamentales para el desarrollo de software de calidad. En Ware Innobig S.O.S, su aplicación asegura un sistema organizado, funcional y adaptable. Al seguir estos lineamientos, se fomenta la colaboración entre desarrolladores y se facilita el mantenimiento y escalabilidad del software.



## CONCLUSIONES:

- Se establecieron un conjunto de estándares de codificación para el desarrollo de la aplicación Ware Innobig S.O.S, que garantice una estructura de código coherente, modular y fácil de mantener, cumpliendo con los principios del paradigma orientado a objetos.

- Se revisó el material de formación y se hizo análisis de la información presentada y de acuerdo con ello se realizaron las evidencias y trabajos correspondientes.

- Se siguieron las normas básicas de presentación de un documento escrito.

- Se realizaron una serie de conclusiones al finalizar como se puede observar.

## BIBLIOGRAFÍA:


- Guia_aprendizaje_7.pdf