

Estándar de Codificación

*Desarrollo de una Aplicación Web con Machine Learning para
el Análisis de Estados Financieros y la Detección de Fraude en
la Empresa Ciclo Contable*

Fecha: 11/07/2025

1. Introducción

Los estándares de codificación son una guía para asegurar que el código fuente desarrollado sea mantenible, legible, eficiente y seguro. En el contexto del presente proyecto, estos estándares garantizan que los módulos de Machine Learning, la aplicación web y el manejo de estados financieros estén alineados con las mejores prácticas de ingeniería de software.

2. Objetivos

- Garantizar la consistencia del código entre todos los módulos del sistema.
- Facilitar el mantenimiento y la escalabilidad del sistema a futuro.
- Prevenir errores mediante la implementación de buenas prácticas de programación.
- Asegurar la seguridad del manejo de datos contables y financieros.

3. Convenciones de Nombres

3.1. Variables y funciones

Usar `snake_case` en Python:

```
def predecir_fraude(indicadores):  
    valores = [  
        indicadores['Liquidez Corriente'],  
        indicadores['Prueba Ácida'],  
        indicadores['Nivel de Endeudamiento'],  
        indicadores['Margen Neto']  
    ]  
  
    prediccion = modelo_fraude.predict([valores])[0]  
    probabilidad = modelo_fraude.predict_proba([valores])[0][1]  
  
    resultado = "FRAUDE" if prediccion == 1 else "NO FRAUDE"  
    return resultado, prediccion == 1, float(probabilidad)
```

Usar `camelCase` en JavaScript:

```
const validarEmail = (email) => {  
  const pattern = /^[ \w.-]+@[ \w.-]+\.\w+$/;  
  return pattern.test(email);  
};
```


3.2. Clases

Python: PascalCase:

```
class Notificacion(db.Model):  
    __tablename__ = 'notificaciones'  
  
    notificacion_id = db.Column(db.Integer, primary_key=True)  
    usuario_id = db.Column(db.Integer, db.ForeignKey('usuarios_app.usuario_id'), nullable=False)  
    mensaje = db.Column(db.String(255), nullable=False)  
    leído = db.Column(db.Boolean, default=False)  
    creado_en = db.Column(db.DateTime, default=datetime.utcnow)  
  
    usuario = db.relationship('UsuarioApp')
```

3.2. Archivos

Descriptivos y en minúsculas:

 datos_fraude_simple.csv

4. Manejo de Errores

Validar entradas del usuario en frontend y backend.

Usar try/except en Python:

```
try:  
    rubro = float(str(er.get('rubro_economico', 0)).replace(',', '').strip())  
except (ValueError, TypeError):  
    rubro = 0.0
```

Usar try/catch en JavaScript:

```
try {  
  const res = await api.post('/auth/register', form);  
  setMsg(res.data.msg);  
} catch (err) {  
  setMsg(err.response?.data?.msg || 'Error al registrar');  
}  
};
```