

Guía de Contribución

¡Gracias por tu interés en contribuir a Simple Smells Detector! Este documento te guiará a través del proceso de contribución y las mejores prácticas del proyecto.

Tabla de Contenidos

- [Código de Conducta](#)
- [Formas de Contribuir](#)
- [Configuración del Entorno de Desarrollo](#)
- [Proceso de Desarrollo](#)
- [Estándares de Código](#)
- [Guía para Agregar Detectores](#)
- [Testing](#)
- [Documentación](#)
- [Proceso de Pull Request](#)

Código de Conducta

Este proyecto adhiere al [Contributor Covenant Code of Conduct](#). Al participar, se espera que mantengas este código. Por favor reporta comportamientos inaceptables a [email del mantenedor].

Formas de Contribuir

Reportar Bugs

- Usa la plantilla de issues para bugs
- Incluye pasos para reproducir el problema
- Especifica la versión de Figma y sistema operativo
- Adjunta capturas de pantalla si es relevante

Sugerir Mejoras

- Usa la plantilla de issues para features
- Explica claramente el problema que resuelve
- Describe la solución propuesta
- Considera alternativas y posibles efectos secundarios

Contribuir Código

- Corregir bugs existentes
- Implementar nuevas características
- Mejorar la documentación
- Agregar o mejorar tests
- Optimizar rendimiento

Mejorar Documentación

- Corregir errores tipográficos
- Clarificar explicaciones existentes
- Agregar ejemplos prácticos
- Traducir documentación (futuro)

Configuración del Entorno de Desarrollo

Requisitos Previos

- **Figma Desktop**: Versión más reciente
- **Editor de Código**: VS Code recomendado con estas extensiones:
 - ESLint
 - Prettier
 - JavaScript (ES6) code snippets
- **Git**: Para control de versiones

Configuración Inicial

1. Fork y clona el repositorio:

```
git clone https://github.com/tu-usuario/simple-smells-detector.git
cd simple-smells-detector
```

2. Configura el plugin en Figma:

- Abre Figma Desktop
- Ve a **Plugins > Development > Import plugin from manifest...**
- Selecciona `manifest.json` del proyecto clonado

3. Configura tu entorno de desarrollo:

```
# Opcional: instalar dependencias de desarrollo si las hay
npm install
```

Estructura del Proyecto

```
|— manifest.json      # Configuración del plugin
|— code.js           # Lógica principal
|— ui.html           # Interfaz de usuario
|— analysis-engine/  # Motor de análisis modular
|   |— core/         # Componentes centrales
|   |— detectors/    # Detectores individuales
|   |— utilities/    # Utilidades compartidas
|— docs/             # Documentación
```

```
|— examples/  
|— tests/
```

```
# Ejemplos y casos de uso  
# Tests (si los hay)
```

Proceso de Desarrollo

Workflow de Git

1. Crea una rama para tu feature/fix:

```
git checkout -b feature/nombre-descriptivo  
# o  
git checkout -b fix/descripcion-del-bug
```

2. Realiza commits atómicos con mensajes descriptivos:

```
git commit -m "feat: agrega detector de espaciado inconsistente ($10)"  
git commit -m "fix: corrige cálculo de proximidad en formularios"  
git commit -m "docs: actualiza documentación de API"
```

3. Mantén tu rama actualizada:

```
git fetch origin  
git rebase origin/main
```

Convención de Nombres de Ramas

- **feature/** - Nuevas características
- **fix/** - Corrección de bugs
- **docs/** - Cambios en documentación
- **test/** - Agregar o modificar tests
- **refactor/** - Refactorización de código

Convención de Commits

Usamos [Conventional Commits](#):

```
<tipo>[scope opcional]: <descripción>  
  
[cuerpo opcional]  
  
[footer opcional]
```

Tipos:

- **feat**: Nueva característica
- **fix**: Corrección de bug
- **docs**: Cambios en documentación
- **style**: Cambios de formato (sin afectar funcionalidad)
- **refactor**: Refactorización de código
- **test**: Agregar o modificar tests
- **chore**: Tareas de mantenimiento

Ejemplos:

```
feat(detector): implementa detector S10 para espaciado vertical
fix(ui): corrige filtrado por severidad en resultados
docs(api): agrega ejemplos de uso para nuevos detectores
```


Estándares de Código

JavaScript

Estilo de Código

- **Indentación**: 2 espacios
- **Comillas**: Simples para strings
- **Punto y coma**: Siempre
- **Nombres de variables**: camelCase
- **Nombres de constantes**: UPPER_SNAKE_CASE
- **Nombres de funciones**: camelCase descriptivo

Ejemplo de Código Bien Formateado

```
//  Bueno
const MAX_FIELDS_THRESHOLD = 8;

function analyzeFormComplexity(formNodes, settings = {}) {
  const threshold = settings.threshold || MAX_FIELDS_THRESHOLD;
  const fieldCount = formNodes.length;

  if (fieldCount > threshold) {
    return {
      type: 'COMPLEXITY',
      severity: 'high',
      message: `Formulario con ${fieldCount} campos excede el límite
recomendado`
    };
  }
}
```

```

    return null;
}

// ✖ Malo
const max_fields=8
function analyze(f,s){
var c=f.length
if(c>max_fields){
return {"type":"COMPLEXITY","severity":"high"}
}
return null
}

```

Documentación de Funciones

```

/**
 * Detecta campos con ancho inadecuado según su tipo de contenido
 * @param {Array<Node>} inputNodes - Nodos de entrada a analizar
 * @param {Object} dataTypes - Configuración de tipos de datos
 * @param {Object} settings - Configuración de umbrales
 * @returns {Array<Finding>} Array de hallazgos detectados
 */
function detectInappropriateFieldSizes(inputNodes, dataTypes, settings) {
    // Implementación...
}

```

HTML/CSS

- **Indentación:** 2 espacios
- **Clases CSS:** kebab-case
- **IDs:** camelCase
- **Responsive design:** Mobile-first
- **Comentarios:** Para secciones complejas

Guía para Agregar Detectores

Estructura de un Detector

Cada detector debe seguir esta estructura estándar:

```

// analysis-engine/detectors/miDetector.js

/**
 * Detector para [descripción del smell]
 * Implementa la detección de [criterio específico]
 */
class MiDetector {

```

```

constructor(settings = {}) {
  this.settings = {
    // Configuración por defecto
    threshold: 10,
    ...settings
  };
}

/**
 * Método principal de análisis
 * @param {Array<Node>} nodes - Nodos a analizar
 * @param {Object} context - Contexto adicional
 * @returns {Array<Finding>} Hallazgos detectados
 */
async analyze(nodes, context = {}) {
  const findings = [];

  // Lógica de detección
  for (const node of nodes) {
    const finding = this.analyzeNode(node, context);
    if (finding) {
      findings.push(finding);
    }
  }

  return findings;
}

/**
 * Analiza un nodo individual
 * @param {Node} node - Nodo a analizar
 * @param {Object} context - Contexto
 * @returns {Finding|null} Hallazgo o null
 */
analyzeNode(node, context) {
  // Implementación específica
  if (this.detectsSmell(node)) {
    return {
      type: 'MI_SMELL_TYPE',
      node,
      severity: this.calculateSeverity(node),
      message: this.generateMessage(node),
      suggestion: this.generateSuggestion(node)
    };
  }
  return null;
}

/**
 * Retorna metadata del detector
 * @returns {Object} Metadata
 */
getMetadata() {

```

```

    return {
      id: 'mi-detector',
      name: 'Mi Detector',
      description: 'Detecta [descripción]',
      category: 'estructura', // o 'semántica', 'consistencia', etc.
      version: '1.0.0'
    };
  }
}

module.exports = MiDetector;

```

Registro del Detector

```

// analysis-engine/core/registry.js
const MiDetector = require('../detectors/miDetector');

const DETECTOR_REGISTRY = new Map([
  // ... detectores existentes
  ['mi-detector', new MiDetector()]
]);

```

Testing del Detector

```

// tests/detectors/miDetector.test.js
describe('MiDetector', () => {
  let detector;

  beforeEach(() => {
    detector = new MiDetector();
  });

  test('detecta el smell correctamente', () => {
    const mockNode = createMockNode({ /* propiedades */ });
    const result = detector.analyzeNode(mockNode);

    expect(result).toBeTruthy();
    expect(result.type).toBe('MI_SMELL_TYPE');
  });

  test('no detecta falsos positivos', () => {
    const mockNode = createValidMockNode();
    const result = detector.analyzeNode(mockNode);

    expect(result).toBeNull();
  });
});

```

Testing

Principios de Testing

- **Cobertura:** Apunta a >80% de cobertura de código
- **Tests unitarios:** Para cada detector individual
- **Tests de integración:** Para flujos completos
- **Tests de regresión:** Para bugs conocidos

Estructura de Tests

```
describe('Detector Name', () => {  
  describe('cuando el nodo tiene el smell', () => {  
    test('lo detecta correctamente', () => {  
      // Arrange  
      // Act  
      // Assert  
    });  
  });  
  
  describe('cuando el nodo es válido', () => {  
    test('no genera falsos positivos', () => {  
      // Test  
    });  
  });  
});
```

Ejecutar Tests

```
# Si configuramos jest u otro framework  
npm test  
  
# Para tests específicos  
npm test -- miDetector.test.js
```

Documentación

Documentar Nuevas Características

1. **README.md:** Actualiza la tabla de detectores
2. **docs/detectors.md:** Agrega especificación técnica
3. **docs/api.md:** Documenta nuevas APIs
4. **Comentarios de código:** Para lógica compleja

Estilo de Documentación

- **Markdown:** Formato estándar
- **Ejemplos:** Código funcional y comentado
- **Diagramas:** ASCII art para flujos simples
- **Enlaces:** Referencias cruzadas entre documentos

Proceso de Pull Request

Antes de Crear el PR

- ☐ Tus cambios están en una rama separada
- ☐ El código sigue los estándares del proyecto
- ☐ Agregaste/actualizaste tests si es necesario
- ☐ Actualizaste la documentación relevante
- ☐ Los tests pasan localmente
- ☐ Los commits siguen la convención establecida

Crear el Pull Request

1. Título descriptivo:

```
feat: implementa detector de espaciado inconsistente (S10)
```

2. Descripción completa:

```
## Descripción
Implementa el detector S10 para identificar espaciado vertical
inconsistente entre campos de formularios.

## Cambios Realizados
- Agrega `spacingDetector.js` con lógica de análisis
- Implementa cálculo de IQR para detectar outliers
- Agrega tests unitarios con 95% de cobertura
- Actualiza documentación de detectores

## Testing
- [x] Tests unitarios pasan
- [x] Probado en prototipos reales
- [x] No introduce regresiones

## Screenshots
[Capturas si es relevante]

Closes #123
```

3. Asigna reviewers según el área de cambio

Durante la Revisión

- **Responde a comentarios** de manera constructiva
- **Realiza cambios** solicitados en commits separados
- **Mantén la conversación** profesional y enfocada

Después de la Aprobación

- El maintainer hará merge usando "Squash and merge"
- Tu rama será eliminada automáticamente
- Los cambios aparecerán en el próximo release

Comunicación

Canales

- **GitHub Issues:** Para bugs y features
- **GitHub Discussions:** Para preguntas generales
- **Pull Requests:** Para revisión de código
- **Email:** Para temas sensibles

Mejores Prácticas

- **Sé específico:** Incluye detalles técnicos relevantes
- **Sé respetuoso:** Mantén un tono profesional
- **Sé paciente:** Los maintainers pueden tardar en responder
- **Busca primero:** Revisa issues existentes antes de crear nuevos

Recursos Adicionales

Documentación Técnica

- [Arquitectura](#)
- [API Documentation](#)
- [Detectores](#)

Herramientas de Desarrollo

- [Figma Plugin API](#)
- [JavaScript Standard Style](#)
- [Conventional Commits](#)

Inspiración

- [Catálogo de Usability Smells](#)
- [Heurísticas de Nielsen](#)

Reconocimientos

¡Gracias a todos los contribuidores que han ayudado a hacer este proyecto mejor! 🍷

Tu nombre aparecerá automáticamente en la lista de contributors cuando hagas tu primer commit.

¿Preguntas? ¡No dudes en abrir un [Discussion](#) o contactarnos directamente!