# 🌙 Módulo Lunar - Sistema de Análisis de Minerales

Proyecto TypeScript para el análisis y validación de minerales lunares según especificaciones de la agencia espacial.

## 📋 Descripción

Sistema de entrada y análisis de datos de minerales lunares con múltiples interfaces de entrada, criterios de validación configurables y formatos de salida (Europeo/Americano).

## 🏗️ Estructura del Proyecto

```
modulo-lunar/
├── src/
│   ├── types.ts          # Tipos y enumeraciones
│   ├── interfaces.ts     # Interfaces del sistema
│   ├── astronauta.ts     # Clase Astronauta
│   ├── validadores.ts    # Criterios de validación
│   ├── sistemas.ts       # Sistemas de entrada/salida
│   ├── mision.ts         # Clase Misión principal
│   └── main.ts           # Punto de entrada
├── dist/                 # JavaScript compilado (generado)
├── index.html            # Página principal
├── styles.css            # Estilos personalizados
├── tsconfig.json         # Configuración TypeScript
├── package.json          # Configuración npm
└── README.md             # Este archivo
```

## 🚀 Instalación y Uso

### Requisitos Previos

- Node.js (v16 o superior)
- npm o yarn

### Pasos de Instalación

1. **Clonar o descargar el proyecto**
2. **Instalar dependencias**

```
bash
```

```
npm install
```

3. **Compilar TypeScript**


bash

```
npm run build
```

4. **Abrir en el navegador**
   - Abrir `index.html` directamente, o
   - Usar un servidor local:


bash

```
# Con Python
python -m http.server 8000

# Con Node.js (instalar http-server globalmente)
npx http-server

# Con VS Code: Live Server extension
```

### Scripts Disponibles


bash

```
npm run build     # Compila TypeScript a JavaScript
npm run watch     # Modo watch (recompila automáticamente)
npm run clean     # Limpia la carpeta dist
```

# 📖 Características

## 1. Interfaces de Entrada

- **Modo Extendido**: Formulario con etiquetas (labels) explícitas
- **Modo Reducido**: Formulario con placeholders

## 2. Criterios de Validación

### Criterio Ígneas

- Grupo: Ígneas
- Tamaño de grano: Muy grueso (> 30mm)

#### Criterio Metamórficas

- Grupo: Metamórficas
- Tamaño de grano: Medio (2-5mm) o Fino (< 2mm)
- Textura: Vítrea

#### Criterio Sedimentarias

- Grupo: Sedimentarias
- Textura: Fanerítica

### 3. Formatos de Salida

- **Formato Europeo**: Textos en español, temperatura en °Celsius
- **Formato Americano**: Textos en inglés, temperatura en °Fahrenheit

### 4. Validaciones

- **ID**: Formato LLDDDDLL (2 letras, 4 números, 2 letras)
- **Dureza**: Escala de Mohs (1-10)
- **Temperatura**: Rango de -100 a 100 Kelvin
- **Campos obligatorios**: Verificación completa

## 🎯 Uso de la Aplicación

1. **Configurar el sistema**:
   - Seleccionar modo de entrada (Extendido/Reducido)
   - Elegir criterio de validación
   - Seleccionar formato de salida
2. **Introducir datos del mineral**:
   - ID (formato: AB1234CD)
   - Nombre del mineral
   - Características físicas (grupo, dureza, textura, etc.)
3. **Analizar**:
   - Presionar "Analizar Mineral"
   - Ver resultado: 😊 (válido) o 😖 (no válido)
   - Si es válido, se muestra la información formateada

## 🏛️ Arquitectura

### Patrones de Diseño Utilizados

- **Strategy Pattern**: Para validadores y formatos de salida
- **Dependency Injection**: En la clase Misión
- **Interface Segregation**: Interfaces específicas y pequeñas

### Clases Principales

typescript

```
// Astronauta que pilota la misión
class Astronauta implements IPilotable

// Validadores de criterios
class ValidadorIgneas implements IValidable
class ValidadorMetamorficas implements IValidable
class ValidadorSedimentarias implements IValidable

// Sistemas de salida
class FormatoEuropeo implements ISalida
class FormatoAmericano implements ISalida

// Coordinador principal
class Mision implements IMisionable
```

# 🔧 Personalización

## Añadir un Nuevo Criterio de Validación

typescript

```
// En validadores.ts
export class ValidadorNuevo implements IValidable {
    getNombre(): string {
        return 'Criterio Nuevo';
    }

    isValid(mineral: Mineral): boolean {
        // Tu lógica de validación
        return true;
    }
}
```

## Añadir un Nuevo Formato de Salida

typescript

```
// En sistemas.ts
export class FormatoNuevo implements ISalida {
    mostrar(mineral: Mineral): string {
        // Tu formato HTML
        return `<div>...</div>`;
    }
}
```

# 📚 Tecnologías

- **TypeScript 5.3+**: Lenguaje principal
- **Bootstrap 5.3**: Framework CSS
- **ES2020**: Target de compilación
- **Módulos ES6**: Sistema de módulos

# 🧪 Pruebas

## Ejemplos de Minerales Válidos

### Para Criterio Ígneas:

- ID: AB1234CD
- Grupo: Ígneas
- Tamaño de grano: Muy grueso

### Para Criterio Metamórficas:

- ID: XY5678ZW
- Grupo: Metamórficas
- Tamaño de grano: Medio o Fino
- Textura: Vítrea

### Para Criterio Sedimentarias:

- ID: QW9876ER
- Grupo: Sedimentarias
- Textura: Fanerítica

# 🐛 Solución de Problemas

## El código TypeScript no se compila

```
bash
```

```
# Verificar versión de TypeScript
tsc --version

# Reinstalar dependencias
rm -rf node_modules package-lock.json
npm install
```

### Los cambios no se reflejan en el navegador

- Asegúrate de compilar después de cada cambio: `npm run build`
- Limpia la caché del navegador (Ctrl + F5)
- Verifica que estás abriendo el archivo desde un servidor

### Error de módulos

- Asegúrate de que `tsconfig.json` tiene `"module": "ES2020"`
- En `index.html`, el script debe tener `type="module"`

## 📝 Licencia

MIT License - Libre para uso educativo y comercial

## 👥 Autor

Proyecto desarrollado para el curso de programación orientada a objetos

## 🤝 Contribuciones

Las contribuciones son bienvenidas. Por favor:

1. Fork el proyecto
2. Crea una rama para tu feature (`git checkout -b feature/AmazingFeature`)
3. Commit tus cambios (`git commit -m 'Add some AmazingFeature'`)
4. Push a la rama (`git push origin feature/AmazingFeature`)
5. Abre un Pull Request

## 📞 Soporte

Para dudas o problemas, crear un issue en el repositorio.

---

**¡Buena suerte con el análisis de minerales lunares!** 🚀 🌙