```
% Algoritmo de visualización sonora
% Autora: Ana Lucía Rodríguez García

classdef AudioVisualizerApp < matlab.apps.AppBase

    % Componentes de la App
    properties (Access = public)
        UIFigure            matlab.ui.Figure
        SoundTab            matlab.ui.container.Tab
        WaveformTab         matlab.ui.container.Tab
        TabGroup            matlab.ui.container.TabGroup
        LoadButton          matlab.ui.control.Button
        PlayButton          matlab.ui.control.Button
        StopButton          matlab.ui.control.Button
        UIAxes1             matlab.ui.control.UIAxes % Forma de onda
        UIAxes2             matlab.ui.control.UIAxes % Visualización sonora
(en ambas pestañas)
        UIAxes3             matlab.ui.control.UIAxes % Visualización sonora
(segunda pestaña)
    end

    % Propiedades adicionales
    properties (Access = private)
        audioFile           % Ruta del archivo de audio
        audioData           % Datos de audio
        sampleRate          % Frecuencia de muestreo
        player              % Objeto audioplayer
        coloresArcoiris     % Colores del arco iris para los acordes
        notas               % Notas musicales
        nSamples            % Número de muestras por bloque
        nPoints             % Puntos para visualización
        theta               % Ángulo para visualización en capas
        timerObj            % Temporizador para la visualización
    end

    % Métodos públicos (Constructor)
    methods (Access = public)
        function app = AudioVisualizerApp
            % Crear interfaz de usuario
            app.UIFigure = uifigure('Name', 'Visualizador de Audio',
'Position', [100, 100, 1000, 600]);
            app.LoadButton = uibutton(app.UIFigure, 'push', 'Text', 'Cargar
Archivo', ...
                'Position', [20, 530, 120, 40], 'ButtonPushedFcn', @(~, ~)
app.LoadButtonPushed());
            app.PlayButton = uibutton(app.UIFigure, 'push', 'Text',
'Reproducir', ...
                'Position', [160, 530, 120, 40], 'ButtonPushedFcn', @(~, ~)
app.PlayButtonPushed());
            app.StopButton = uibutton(app.UIFigure, 'push', 'Text', 'Parar',
...
                'Position', [300, 530, 120, 40], 'ButtonPushedFcn', @(~, ~)
```

```
app.StopButtonPushed());

            % Crear pestañas
            app.TabGroup = uitabgroup(app.UIFigure, 'Position', [0, 0, 1000,
500]);

            % Pestaña de forma de onda y visualización sonora
            app.WaveformTab = uitab(app.TabGroup, 'Title', 'Forma de Onda y
Visualización Sonora');

            % Altura y posición común para el centrado vertical
            figuraAltura = 300; % Altura de las figuras
            figuraAnchura = 450; % Anchura de las figuras
            tabAltura = 500;    % Altura total de la pestaña
            tabAnchura = 1000;  % Anchura total de la pestaña
            espaciado = 20;     % Espaciado entre las figuras
            centroVertical = (tabAltura - figuraAltura) / 2;
            centroHorizontal = (tabAnchura - (2 * figuraAnchura +
espaciado)) / 2;

            % Figura de forma de onda (izquierda)
            app.UIAxes1 = uiaxes(app.WaveformTab, 'Position',
[centroHorizontal, centroVertical, figuraAnchura, figuraAltura]);
            title(app.UIAxes1, 'Forma de Onda');
            xlabel(app.UIAxes1, 'Tiempo (s)');
            ylabel(app.UIAxes1, 'Amplitud');

            % Figura de visualización sonora (derecha)
            app.UIAxes2 = uiaxes(app.WaveformTab, 'Position',
[centroHorizontal + figuraAnchura + espaciado, centroVertical, figuraAnchura,
figuraAltura]);
            title(app.UIAxes2, 'Visualización Sonora');
            xlabel(app.UIAxes2, 'X');
            ylabel(app.UIAxes2, 'Y');

            % Pestaña de visualización sonora completa
            app.SoundTab = uitab(app.TabGroup, 'Title', 'Visualización Sonora
Completa');
            app.UIAxes3 = uiaxes(app.SoundTab, 'Position', [20, 20, 960,
460]);
            title(app.UIAxes3, 'Visualización Sonora');
            xlabel(app.UIAxes3, 'X');
            ylabel(app.UIAxes3, 'Y');

            % Inicializar los colores del arco iris y acordes
            app.coloresArcoiris = [linspace(1, 0, 7); linspace(0, 1, 7);
linspace(0, 0, 7)]'; % Rojo, naranja, amarillo, verde, azul, índigo, violeta
            app.notas = {'C', 'C#', 'D', 'D#', 'E', 'F', 'F#', 'G', 'G#',
'A', 'A#', 'B'};
        end
    end

    % Métodos privados (Callbacks y funcionalidad interna)
    methods (Access = private)
```

```
        % Callback para cargar archivo de audio
        function LoadButtonPushed(app, ~)
            [file, path] = uigetfile({'*.mp3;*.wav', 'Audio Files (*.mp3,
*.wav)'});
            if isequal(file, 0)
                uialert(app.UIFigure, 'No se seleccionó ningún archivo.',
'Error');
                return;
            end

            % Cargar archivo
            app.audioFile = fullfile(path, file);
            [app.audioData, app.sampleRate] = audioread(app.audioFile);
            app.audioData = app.audioData(:, 1); % Usar un canal

            % Crear reproductor
            app.player = audioplayer(app.audioData, app.sampleRate);
            uialert(app.UIFigure, 'Archivo cargado exitosamente.', 'Éxito');
        end

        % Callback para reproducir audio
        function PlayButtonPushed(app, ~)
            if isempty(app.player)
                uialert(app.UIFigure, 'Cargue primero un archivo de audio.',
'Error');
                return;
            end
            play(app.player);
            app.startVisualization();
        end

        % Callback para parar audio
        function StopButtonPushed(app, ~)
            if ~isempty(app.player)
                stop(app.player);
            end
            if ~isempty(app.timerObj)
                stop(app.timerObj);
            end
        end

        % Función para iniciar la visualización
        function startVisualization(app)
            blockDuration = 0.1; % Duración del bloque en segundos
            app.nSamples = round(blockDuration * app.sampleRate); % Número de
muestras por bloque
            app.nPoints = 300; % Puntos para visualización
            app.theta = linspace(0, 2 * pi, app.nPoints); % Para la elipse

            % Configurar temporizador
            app.timerObj = timer('ExecutionMode', 'fixedRate', 'Period',
blockDuration, ...
                'TimerFcn', @(~, ~) app.updateVisualization());
            start(app.timerObj);
```

```
        end

        % Función para actualizar la visualización
        function updateVisualization(app)
            currentSample = get(app.player, 'CurrentSample');
            if currentSample + app.nSamples > length(app.audioData)
                stop(app.timerObj);
                return;
            end

            % Obtener bloque de audio
            block = app.audioData(currentSample:(currentSample + app.nSamples
- 1));
            tiempo = linspace(0, app.nSamples / app.sampleRate, app.nSamples);

            % Actualizar forma de onda
            plot(app.UIAxes1, tiempo, block, 'Color', [0.8 0.8 0.8]);
            app.UIAxes1.XLabel.String = 'Tiempo (s)';
            app.UIAxes1.YLabel.String = 'Amplitud';
            app.UIAxes1.Title.String = 'Forma de Onda';
            app.UIAxes1.Color = 'k';
            app.UIAxes1.YLim = [-0.8, 0.8]; % Eje Y fijo

            % Detectar los picos de la forma de onda
            [pks, locs] = findpeaks(abs(block)); % Picos de la forma de onda
            if ~isempty(pks)
                maxPeak = max(pks); % Obtener el pico máximo
                escalaIntensidad = 0.5 + 2 * maxPeak; % Escala de
visualización según el pico de la forma de onda
            else
                escalaIntensidad = 1;
            end

            % Calcular el espectro del bloque
            spectrum = abs(fft(block));
            spectrum = spectrum(1:round(app.nSamples / 2)); % Usar solo la
mitad positiva
            spectrum = spectrum / max(spectrum); % Normalizar
            spectrum = interp1(linspace(0, 1, length(spectrum)), spectrum,
linspace(0, 1, app.nPoints), 'linear');
            spectrum(isnan(spectrum)) = 0; % Evitar NaN

            % Detectar la nota principal (simplificación como la frecuencia
principal)
            [~, idxMax] = max(spectrum);
            frecuenciaPrincipal = (idxMax - 1) * app.sampleRate /
app.nSamples;
            notaDetectada = app.notas{mod(round(frecuenciaPrincipal / 10),
length(app.notas)) + 1};

             % Asignar un color basado en la nota detectada
            acordeIdx = mod(find(strcmp(app.notas, notaDetectada)),
size(app.coloresArcoiris, 1)) + 1;
            currentColor = app.coloresArcoiris(acordeIdx, :); % Obtener el
```

```
color correspondiente

            % Añadir ruido aleatorio
            ruido = (rand(1, app.nPoints) - 0.5) * 0.2; % Rango de ruido
entre [-0.1, 0.1]

            % Visualización sonora
            cla(app.UIAxes2);
            cla(app.UIAxes3);
            hold(app.UIAxes2, 'on');
            hold(app.UIAxes3, 'on');
            for layer = 1:5
                layerScale = (0.5 + (layer - 1) * 0.2) * escalaIntensidad;
                layerShape = (0.6  + ruido) * layerScale;

                % Coordenadas de las capas
                x = layerShape .* cos(app.theta);
                y = layerShape .* sin(app.theta);

                % Dibujar las capas en ambas pestañas
                fill(app.UIAxes2, x, y, currentColor, 'FaceAlpha', 0.3);
                plot(app.UIAxes2, x, y, 'Color', currentColor, 'LineWidth',
1.2);
                fill(app.UIAxes3, x, y, currentColor, 'FaceAlpha', 0.3);
                plot(app.UIAxes3, x, y, 'Color', currentColor, 'LineWidth',
1.2);
            end

            % Ajustar los límites de los ejes de la visualización sonora
            axis(app.UIAxes2, 'equal');
            app.UIAxes2.XLim = [-2.5 2.5];
            app.UIAxes2.YLim = [-2.5 2.5];
            app.UIAxes2.Color = 'k';
            app.UIAxes2.Title.String = 'Visualización Sonora';

            %axis(app.UIAxes3, 'equal');
            app.UIAxes3.XLim = [-6.5 6.5];
            app.UIAxes3.YLim = [-4.5 4.5];
            app.UIAxes3.DataAspectRatio = [1, 1, 1]; % Relación de aspecto
cuadrada
            app.UIAxes3.Color = 'k';
            app.UIAxes3.Title.String = 'Visualización Sonora';
            drawnow;
        end
    end
end


ans =

  AudioVisualizerApp with properties:

      UIFigure: [1x1 Figure]
      SoundTab: [1x1 Tab]
```

```
    WaveformTab: [1x1 Tab]
       TabGroup: [1x1 TabGroup]
     LoadButton: [1x1 Button]
     PlayButton: [1x1 Button]
     StopButton: [1x1 Button]
        UIAxes1: [1x1 UIAxes]
        UIAxes2: [1x1 UIAxes]
        UIAxes3: [1x1 UIAxes]
```

Published with MATLAB® R2024b