I'll help you describe the libraries used in your code and show examples of their usage. Let me organize the information in Polish since you seem to prefer that language.

Let me describe each library and show examples of their usage from your code:

1. Express.js i Express Rate Limit

|  |
| --- |
| **// server.js**  **const express = require('express');**  **const rateLimit = require('express-rate-limit');**  **const app = express();**  **// Konfiguracja rate limitera**  **const limiter = rateLimit({**  **windowMs: 15 \* 60 \* 1000, // 15 minut**  **max: 100 // limit 100 requestów na 15 minut**  **});** |

Express.js jest używany jako główny framework do obsługi serwera HTTP. Rate Limit chroni przed nadmiernym obciążeniem serwera poprzez ograniczenie liczby requestów.

2. UA Parser JS

|  |
| --- |
| **// visitController.js**  **const UAParser = require('ua-parser-js');**  **exports.trackVisit = async (req, res) => {**  **const parser = new UAParser(req.headers['user-agent']);**  **const uaResult = parser.getResult();**    **const visitData = {**  **browser: uaResult.browser.name,**  **os: uaResult.os.name,**  **device: uaResult.device.model || 'Desktop',**  **// ...**  **};**  **}** |

UA Parser JS służy do analizy User Agent przeglądarki, dostarczając informacje o przeglądarce, systemie operacyjnym i urządzeniu użytkownika.

3. Node.js Built-in Modules (crypto, path)

|  |
| --- |
| **// visitController.js**  **const crypto = require('crypto');**  **function generateSecureRandomString() {**  **const array = new Uint32Array(10);**  **crypto.randomFillSync(array);**  **return Array.from(array, dec => ('0' + dec.toString(36)).substr(-2)).join('');**  **}**  **// statsController.js**  **const path = require('path');**  **exports.getStatsPage = (req, res) => {**  **res.sendFile(path.join(\_\_dirname, '../../public/stats.html'));**  **};** |

- crypto jest używany do generowania bezpiecznych losowych identyfikatorów sesji

- path służy do obsługi ścieżek plików w sposób niezależny od systemu operacyjnego

Plotly.js (Frontend)

|  |
| --- |
| **// public/js/stats.js**  **function createBarChart(data, containerId, title, xAxisTitle, yAxisTitle, color = '#3B82F6') {**  **const chartData = [{**  **x: data.map(item => item.page\_url || item.country || item.period),**  **y: data.map(item => item.count || item.visits),**  **type: 'bar',**  **marker: { color: color }**  **}];**  **const layout = {**  **title: title,**  **font: { family: 'system-ui, -apple-system, sans-serif' },**  **xaxis: {**  **title: xAxisTitle,**  **tickangle: -45**  **},**  **yaxis: {**  **title: yAxisTitle**  **},**  **margin: { b: 100 }**  **};**  **Plotly.newPlot(containerId, chartData, layout, commonConfig);**  **}**  **function createPieChart(data, containerId, title, defaultLabel = 'Nieznane') {**  **const chartData = [{**  **labels: data.map(item => item.referrer || item.browser || defaultLabel),**  **values: data.map(item => item.count),**  **type: 'pie'**  **}];**  **Plotly.newPlot(containerId, chartData, layout, commonConfig);**  **}** |

Plotly.js jest wykorzystywany do tworzenia interaktywnych wykresów statystyk, w tym:

- Wykresów słupkowych (np. dla popularności stron)

- Wykresów kołowych (np. dla udziału przeglądarek)

- Wykresów czasowych (dla analizy wizyt w czasie)

5. Baza danych (PostgreSQL)

|  |
| --- |
| // visitModel.js  exports.getVisitTimes = async () => {  const hourlyData = await db.query(`  SELECT  TO\_CHAR(DATE\_TRUNC('hour', timestamp), 'HH24:00') as period,  COUNT(\*) as count  FROM visits  WHERE timestamp >= NOW() - INTERVAL '24 hours'  GROUP BY DATE\_TRUNC('hour', timestamp)  ORDER BY DATE\_TRUNC('hour', timestamp)  `);      // statsModel.js  exports.getTopPages = async () => {  return await db.query(  'SELECT page\_url, COUNT(\*) as count FROM visits GROUP BY page\_url ORDER BY count DESC LIMIT 5'  );  }; |