INFORME DE LABORATORIO

INFORMACIÓN BÁSICA					
ASIGNATURA:	Programación Web 2				
TÍTULO DE LA PRÁCTICA:	*Actividad AJAX*				
NÚMERO DE PRÁCTICA:	*2*	**AÑO LECTIVO:**	*2025*	**NRO. SEMESTRE:**	*III*
FECHA DE PRESENTACIÓN	*03/05/2025*	**HORA DE PRESENTACIÓN**			
INTEGRANTE (s): Auccacusi Conde Brayan Carlos				**NOTA:**	
DOCENTE(s): Corrales Delgado Carlos Jose Luis					

SOLUCIÓN Y RESULTADOS

I. SOLUCIÓN DE EJERCICIOS/PROBLEMAS

Estructura del proyecto

Ejercicios

```
|__problema1.html
|__problema2.js
|__problema2.html
|__problema2.js
|__data.json
```

PROBLEMA1.HTML

```html
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Problema 1 - Comparativa de Regiones</title>
  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
</head>
<body>
  <h2>Selecciona regiones para comparar:</h2>
  <select id="region1">Region A</select>
  <select id="region2">Region B</select>
  <canvas id="grafico"></canvas>
  <script src="problema1.js"></script>
</body>
</html>

```

Conectamos con problema1.js y en el head con https://cdn.jsdelivr.net/npm/chart.js que nos permitirá graficar.

PROBLEMA1.JS

```
let chartInstance;

fetch('data.json')
  .then(res => res.json())
  .then(data => {
    // Obtener nombres de regiones
    const regiones = data.map(d => d.region);
    const select1 = document.getElementById('region1');
    const select2 = document.getElementById('region2');

    // Llenar selects
    regiones.forEach(region => {
      select1.innerHTML += `<option value="${region}">${region}</option>`;
      select2.innerHTML += `<option value="${region}">${region}</option>`;
    });

    const render = () => {
      const r1 = select1.value;
      const r2 = select2.value;

      // Obtener datos de las regiones seleccionadas
      const region1 = data.find(d => d.region === r1);
      const region2 = data.find(d => d.region === r2);

      // Si alguna región no se encuentra, salir
      if (!region1 || !region2) return;

      // Obtener todas las fechas (asumiendo que todas las regiones tienen las mismas fechas)
      const fechas = region1.confirmed.map(d => d.date);

      // Obtener los datos de confirmados (convertir a número)
      const datosR1 = region1.confirmed.map(d => Number(d.value));
      const datosR2 = region2.confirmed.map(d => Number(d.value));

      // Si ya existe un gráfico, destruirlo
      if (chartInstance) {
        chartInstance.destroy();
      }
```

Este fragmento carga datos desde un archivo data.json, obtiene los nombres de las regiones y llena dos listas desplegables (select1 y select2) con esas regiones. Luego define una función render que, al ejecutarse, toma las regiones seleccionadas, busca sus datos correspondientes, extrae las fechas y los valores confirmados (convirtiéndolos a números), y si ya existe un gráfico previo, lo destruye para evitar superposiciones; esta preparación permite comparar visualmente dos regiones distintas mediante un gráfico actualizado dinámicamente según la selección del usuario.

```
    // Crear nuevo gráfico
    chartInstance = new Chart(document.getElementById('grafico'), {
      type: 'line',
      data: {
        labels: fechas,
        datasets: [
          {
            label: r1,
            data: datosR1,
            borderColor: 'blue',
            fill: false
          },
          {
            label: r2,
            data: datosR2,
            borderColor: 'red',
            fill: false
          }
        ]
      },
      options: {
        responsive: true,
        plugins: {
          title: {
            display: true,
            text: `Comparación de casos confirmados entre ${r1} y ${r2}`
          }
        },
        scales: {
          x: {
            title: {
              display: true,
              text: 'Fecha'
            }
          },
          y: {
            title: {
              display: true,
              text: 'Casos confirmados'
            },
            beginAtZero: true
          }
        }
      }
    });
  };

  // Llamar render por defecto para mostrar algún gráfico inicial
  select1.value = regiones[0];
  select2.value = regiones[1] || regiones[0];
  render();

  // Asociar cambios a los select
  select1.addEventListener('change', render);
  select2.addEventListener('change', render);
});
```

Aqui el código crea un nuevo gráfico de líneas utilizando Chart.js, donde se comparan visualmente los casos confirmados de dos regiones seleccionadas (r1 y r2) a lo largo del tiempo. Utiliza como etiquetas del eje X las fechas (fechas) y como datos del eje Y los valores de casos confirmados (datosR1 y datosR2), asignándoles diferentes colores (azul y rojo). El gráfico incluye un título dinámico que refleja las regiones comparadas, y se configura para ser responsivo y con títulos en ambos ejes. Al final, se establece un gráfico inicial por defecto usando las dos primeras regiones del arreglo y se añaden detectores de eventos para que al cambiar las selecciones en los select, se actualice automáticamente el gráfico con los nuevos datos.

PROBLEMA2.HTML

```html
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Problema 2 - Comparativo de Confirmados</title>
  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
</head>
<body>
  <h1>Comparación de Casos Confirmados por Región (excepto Lima y Callao)</h1>
  <canvas id="grafico" width="800" height="400"></canvas>
  <script src="problema2.js"></script>
</body>
</html>

```

Conectamos con problema1.js y en el head con https://cdn.jsdelivr.net/npm/chart.js que nos permitirá graficar.

PROBLEMA2.JS

```javascript
fetch('data.json')
  .then(res => res.json())
  .then(data => {
    // Filtrar regiones excepto Lima y Callao
    const regionesFiltradas = data.filter(d => d.region !== "Lima" && d.region !== "Callao");

    // Obtener fechas desde la primera región filtrada
    const fechas = regionesFiltradas[0].confirmed.map(d => d.date);

    // Preparar datasets para cada región
    const datasets = regionesFiltradas.map((region, index) => {
      const color = `hsl(${(index * 50) % 360}, 70%, 50%)`; // Generar color distinto para cada región

      return {
        label: region.region,
        data: region.confirmed.map(d => Number(d.value)),
        borderColor: color,
        fill: false
      };
    });

```

Esta porción de código obtiene los datos desde un archivo data.json, filtra todas las regiones excluyendo "Lima" y "Callao", y extrae las fechas desde la primera región filtrada, asumiendo que todas las regiones comparten las mismas fechas. Luego, para cada región restante, prepara un conjunto de datos (datasets) que contiene el nombre de la región como etiqueta

(label), sus valores numéricos de casos confirmados por día (data), y un color de línea único generado dinámicamente usando la función hsl() para diferenciarlas visualmente en el gráfico.

```

  // Crear gráfico
  new Chart(document.getElementById('grafico'), {
    type: 'line',
    data: {
      labels: fechas,
      datasets: datasets
    },
    options: {
      responsive: true,
      plugins: {
        title: {
          display: true,
          text: 'Crecimiento Diario de Confirmados por Región (sin Lima y Callao)'
        },
        legend: {
          display: true,
          position: 'bottom'
        }
      },
      scales: {
        x: {
          title: {
            display: true,
            text: 'Fecha'
          }
        },
        y: {
          title: {
            display: true,
            text: 'Casos Confirmados'
          },
          beginAtZero: true
        }
      }
    }
  });
});

```

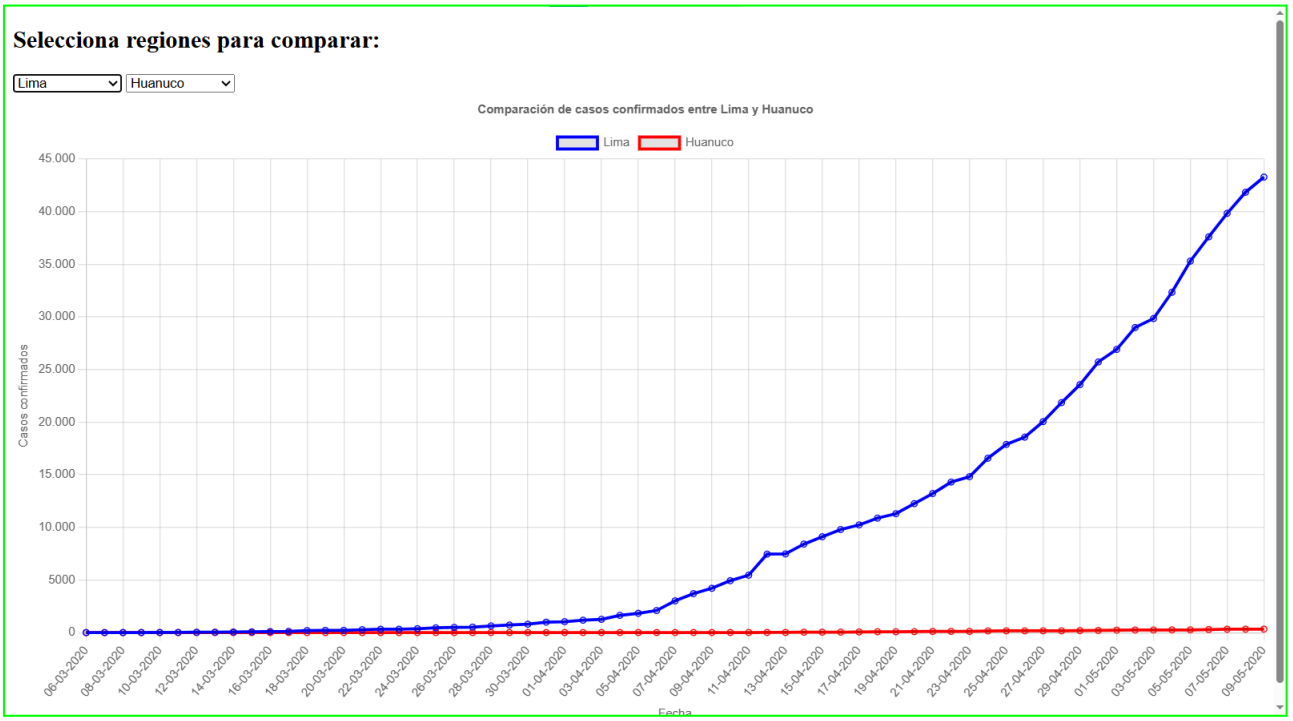
Esta sección del código crea un gráfico de líneas usando Chart.js, en el que se visualiza el crecimiento diario de casos confirmados por región, excluyendo Lima y Callao. Utiliza como etiquetas del eje X las fechas extraídas anteriormente, y como conjuntos de datos (datasets) los correspondientes a cada región filtrada. Cada línea del gráfico representa una región distinta con un color único. Además, se configura el gráfico para que sea responsivo, se incluye un título descriptivo, se muestra una leyenda en la parte inferior y se etiquetan ambos ejes: el eje X con las fechas y el eje Y con el número de casos confirmados, comenzando desde cero.

II. SOLUCIÓN DEL CUESTIONARIO

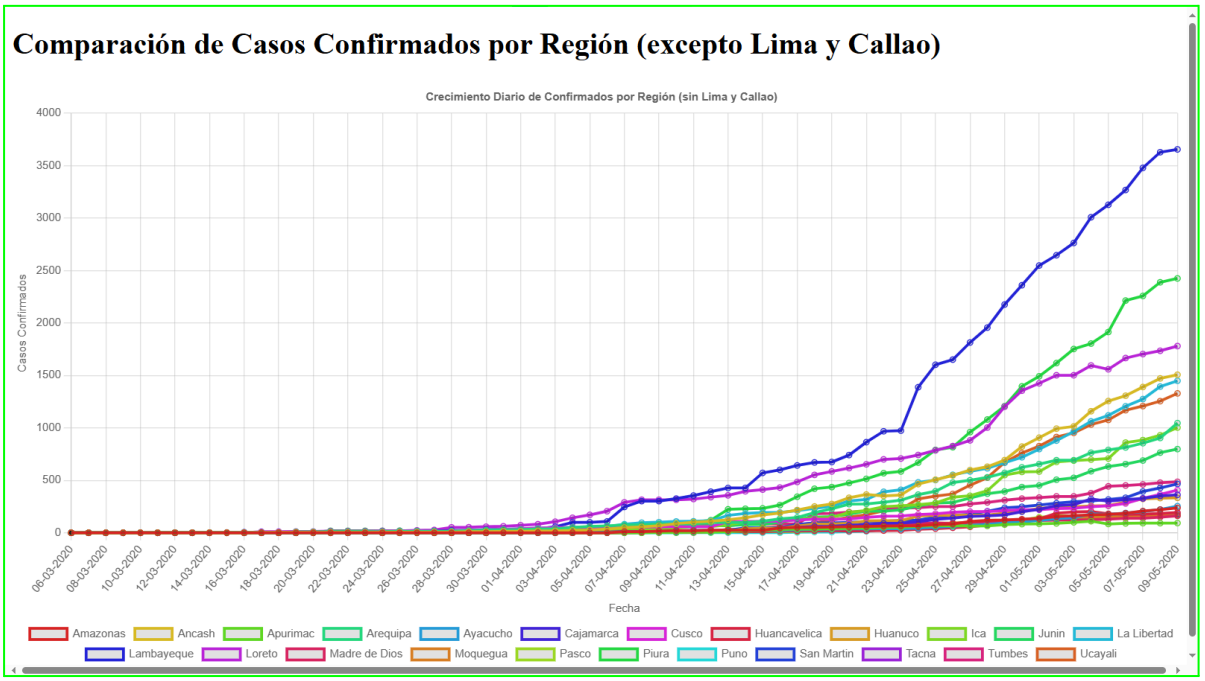
Se usó live server como herramienta para visualizar los cambios.

RESULTADOS:

PROBLEMA 1:



PROBLEMA 2:



III. CONCLUSIONES

- Se logró visualizar comparativamente el crecimiento de casos confirmados por día en diversas regiones del país, excluyendo Lima y Callao, mediante un gráfico dinámico generado con la librería Chart.js.
- Se aprendió a manipular archivos JSON con JavaScript, extrayendo y transformando datos relevantes para su uso en visualizaciones gráficas.
- Se comprendió la importancia del filtrado de datos (filter) y su aplicación para centrarse en regiones específicas, en este caso evitando que Lima y Callao influyeran en la representación de otras zonas.
- Se adquirieron conocimientos sobre cómo estructurar datasets para gráficos en línea, asignando colores únicos de forma dinámica y organizando los datos por fechas comunes.
- Este ejercicio refuerza la utilidad de la programación web para el análisis visual de información epidemiológica, permitiendo detectar patrones de crecimiento o comparaciones entre regiones con facilidad.

REFERENCIAS Y BIBLIOGRAFÍA

- *https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX*
- *https://developer.mozilla.org/en-US/docs/Web/JavaScript*
- *https://www.chartjs.org/docs/latest/*
- *https://developer.mozilla.org/en-US/docs/Learn*