Informe de Laboratorio 04
Tema: Ajax

Nota

Estudiante	Escuela	Asignatura
Alexandra Raquel Quispe Paul Andree Cari Jeferson Joao Basurco	Escuela Profesional de Ingeniería de Sistemas	Programación Web II Semestre: I Código: 20231001

Laboratorio	Tema	Duración
04	Ajax	12 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2024 - A	Del 14 Mayo 2024	Al 18 Mayo 2023

1. Tarea

Para la tarea sobre Ajax en w3schools, se debe revisar la sección correspondiente en dicho sitio web y presentar un informe con capturas de pantalla de cada ejemplo de Ajax implementado en su propio servidor web. Además, se requiere realizar una práctica utilizando Ajax y Google Charts. Primero, es necesario instalar Python en la computadora y lanzar un servidor web local utilizando los comandos:

```
python -m SimpleHTTPServer 8000  # Para Python 2
python3 -m http.server 8000       # Para Python 3
```

Alternativamente, se puede utilizar un servidor web como Apache con Xampp. Luego, se debe descargar el archivo `data.json` y colocarlo en el directorio raíz del servidor web.

Para cada problema propuesto, se debe implementar un programa en Ajax y una página que realice las siguientes tareas:

- Listar todas las regiones.
- Mostrar el número total de confirmados por región.
- Encontrar las 10 regiones con mayor número de confirmados.
- Visualizar un gráfico temporal de los valores para Arequipa.
- Crear gráficos comparativos entre regiones usando líneas.
- Visualizar un gráfico del crecimiento en regiones, excluyendo Lima y Callao.
- Realizar gráficos comparativos entre regiones elegidas por el usuario.
- Mostrar un gráfico comparativo del crecimiento diario en regiones, excluyendo Lima y Callao.

2. Equipos, materiales y temas utilizados

- **Sistema Operativo:** Ubuntu GNU/Linux 23.04 Lunar Lobster 64 bits, Kernel 6.2.
- **Editor de texto:** VIM 9.0.
- **Entorno de desarrollo:** OpenJDK 64-Bits 17.0.7.
- **Control de versiones:** Git 2.39.2.
- **Repositorio:** Cuenta en GitHub con el correo institucional.
- **Tecnologías utilizadas:**
  - **Ajax:** Para la implementación de solicitudes asíncronas y la manipulación dinámica de datos.
  - **Google Charts:** Para la visualización gráfica de datos, incluyendo gráficos de líneas y gráficos comparativos.
  - **Python:** Utilizado para lanzar un servidor web local para pruebas.
  - **Servidor web:** SimpleHTTPServer (Python 2) o http.server (Python 3) para el desarrollo y prueba de aplicaciones web.

3. URL de Repositorio GitHub

- URL del Repositorio GitHub para clonar o recuperar.
- `https://github.com/JefersonPWeb2/Pw2-Lab04.git`
- URL de los videos explicativos de cada integrante.
- `https://drive.google.com/drive/folders/1rJNriwChjEpbM9KHMSriJcTZOHE0WXqU?usp=sharing`

4. Actividades con el repositorio GitHub

4.1. Tarea Ajax en w3schools

- Capturas W3schools AJAX.
- `https://docs.google.com/document/d/1mWWgP6RKmzeaMjWebnFMSfqGnqmgGHWAhDCAF-BAftk/edit`

4.2. Tarea Ajax y NodeJS

En esta tarea, utilizamos Node.js para configurar un servidor web con varios módulos esenciales. El código que se muestra en la Figura 1 importa los siguientes módulos:

- **express:** Utilizado para crear y gestionar el servidor web.
- **fs:** Proporciona funciones para interactuar con el sistema de archivos.
- **path:** Facilita la manipulación de rutas de archivos.
- **marked:** Convierte texto en formato Markdown a HTML.
- **body-parser:** Procesa los datos de las solicitudes HTTP.

```
const express = require('express');
const fs = require('fs');
const path = require('path');
const { marked } = require('marked');
const bodyParser = require('body-parser');

const app = express();
const PORT = 3000;
const markdownDir = path.join(__dirname, 'markdown-files');
```

Figura 1: Configuración del servidor web en Node.js

```
const app = express();
const PORT = 3000;
const markdownDir = path.join(__dirname, 'markdown-files');

app.use(express.static(__dirname));
app.use(bodyParser.json());

app.get('/files', (req, res) => {
    fs.readdir(markdownDir, (err, files) => {
        if (err) {
            res.status(500).json({ error: 'No se pudo listar los archivos' });
            return;
        }
        res.json(files.filter(file => file.endsWith('.md')));
    });
});
```

Figura 2: Configuración del servidor y manejo de solicitudes en Node.js

El código crea una instancia de una aplicación Express, define el puerto del servidor como 3000 y especifica el directorio donde se almacenarán los archivos Markdown.

El código en la Figura 2 continúa con la configuración del servidor Node.js. Las siguientes líneas de código realizan estas acciones:

- **Líneas 8-10:** Se inicializa la aplicación Express, se define el puerto del servidor como 3000 y se establece la ruta para almacenar los archivos Markdown.
- **Líneas 12-13:** Se configuran los archivos estáticos para ser servidos desde el directorio actual y se permite que el servidor analice las solicitudes en formato JSON.
- **Líneas 15-23:** Se maneja las solicitudes a la ruta `/files`, leyendo el contenido del directorio `markdownDir` y devolviendo una lista de los archivos que terminan en `.md`.

4.2.1. Manejo de la solicitud para leer un archivo Markdown

El código en la Figura 3 define una ruta que permite leer y convertir archivos Markdown a HTML. Las acciones específicas realizadas son:

- La ruta `/file` toma el nombre del archivo desde la solicitud (`req.query.name`).
- Se construye la ruta completa del archivo utilizando `path.join(markdownDir, fileName)`.

```
25    app.get('/file', (req, res) => {
26        const fileName = req.query.name;
27        const filePath = path.join(markdownDir, fileName);
28
29        fs.readFile(filePath, 'utf8', (err, data) => {
30            if (err) {
31                res.status(500).json({ error: 'No se pudo leer el archivo' });
32                return;
33            }
34            const html = marked(data);
35            res.json({ html });
36        });
37    });
```

Figura 3: Lectura y conversión de archivos Markdown en Node.js

- El archivo se lee en formato UTF-8 utilizando `fs.readFile`.
- Si ocurre un error durante la lectura, se devuelve un mensaje de error con el código de estado 500.
- Si la lectura es exitosa, el contenido del archivo se convierte a HTML usando `marked`.
- El HTML generado se devuelve como una respuesta JSON.

4.2.2. Manejo de la solicitud para crear un archivo Markdown

```
39    app.post('/create', (req, res) => {
40        const { fileName, fileContent } = req.body;
41        const filePath = path.join(markdownDir, `${fileName}.md`);
42
43        fs.writeFile(filePath, fileContent, err => {
44            if (err) {
45                res.status(500).json({ success: false, message: 'No se pudo crear el archivo' });
46                return;
47            }
48            res.json({ success: true });
49        });
50    });
```

Figura 4: Creación de archivos Markdown en Node.js

El código en la Figura 4 define una ruta que permite crear nuevos archivos Markdown. Las acciones específicas realizadas son:

- La ruta `/create` toma el nombre del archivo y su contenido desde el cuerpo de la solicitud (`req.body`).
- Se construye la ruta completa del archivo utilizando `path.join(markdownDir, ‘`$fileName.md$`‘)`.$El archivo se escribe$
- Si ocurre un error durante la escritura, se devuelve un mensaje de error con el código de estado 500 y un indicador de éxito falso.
- Si la escritura es exitosa, se devuelve una respuesta JSON indicando éxito.

4.2.3. Iniciando el servidor Node.js

```
52    app.listen(PORT, () => {
53        console.log(`Servidor corriendo en http://localhost:${PORT}`);
54    });
```

Figura 5: Iniciando el servidor en Node.js

El código en la Figura 5 inicia el servidor Node.js en el puerto 3000. Las acciones específicas realizadas son:

- La función `app.listen(PORT)` inicia el servidor en el puerto especificado (3000 en este caso).
- Se muestra un mensaje en la consola indicando que el servidor está corriendo en `http://localhost:3000`.

4.2.4. Estructura de carpetas del proyecto

Figura 6: Estructura de carpetas del proyecto

En la Figura 6 se muestra la distribución de carpetas del proyecto Node.js. La carpeta `markdown-files` se crea para almacenar archivos con extensión ".md", mientras que la carpeta `node`$_m$*odulesesunacarpetapredeterminadage*
parser.

```
\LAB - PWeb2\PWeb2-Lab04> npm install express body-parser marked
```

Figura 7: Archivos en el proyecto

Figura 8: Archivos en el proyecto

`Además, se crean automáticamente los archivos "package-lock.jsonz "package.jsonçomo parte de la configuración del proyecto.`

4.3. Tarea Ajax y Google Charts

- Liste todas las \regiones".

```
$(document).ready(function() {
    $('#btnMostrarRegiones').click(function() {
        $.getJSON('../data/data.json', function(data) {
            const regionList = $('#regionList');
            regionList.empty();

            data.forEach(regionData => {
                const li = $('<li>').text(regionData.region);
                regionList.append(li);
            });
        }).fail(function() {
            console.error('Error al cargar data.json');
        });
    });
});
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="../css/style.css"> <!-- Ruta actualizada al archivo CSS -->
    <title>Listado de Regiones</title>
</head>
<body>
    <h1>Listado de Regiones</h1>
    <button id="btnMostrarRegiones">Mostrar Regiones</button>
    <ul id="regionList"></ul>

    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script src="../js/list_regions.js"></script> <!-- Ruta actualizada al archivo JavaScript
        -->
</body>
</html>
```

4.4. Actividad

- Muestre el número total de confirmados por región

```
$(document).ready(function() {
    $('#btnMostrarTotales').click(function() {
        $.getJSON('../data/data.json', function(data) {
            const totalList = $('#totalList');
            totalList.empty();

            data.forEach(regionData => {
                const totalConfirmed = regionData.confirmed.reduce((sum, entry) => sum +
                    parseInt(entry.value), 0);
                const li = $('<li>').text(`${regionData.region}: ${totalConfirmed}`);
```

```
            totalList.append(li);
        });
    }).fail(function() {
        console.error('Error al cargar data.json');
    });
});
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="../css/style.css"> <!-- Ruta actualizada al archivo CSS -->
    <title>Total Confirmados por Regin</title>
</head>
<body>
    <h1>Total Confirmados por Regin</h1>
    <button id="btnMostrarTotales">Mostrar Totales</button>
    <ul id="totalList"></ul>

    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script src="../js/total_confirmed.js"></script>
</body>
</html>
```

4.5. Actividad

- Encuentre las 10 regiones cuya suma total sea la mayor

```
$(document).ready(function() {
    $.getJSON('../data/data.json', function(data) {
        const sortedRegions = data.sort((a, b) => {
            const sumA = a.confirmed.reduce((total, entry) => total + parseInt(entry.value),
                0);
            const sumB = b.confirmed.reduce((total, entry) => total + parseInt(entry.value),
                0);
            return sumB - sumA;
        });

        const top10Regions = sortedRegions.slice(0, 10);
        const topRegionsList = $('#topRegionsList');
        top10Regions.forEach((region, index) => {
            const sum = region.confirmed.reduce((total, entry) => total +
                parseInt(entry.value), 0);
            const position = index + 1;
            const li = $('<li>').text(`Top ${position}: ${region.region} - ${sum}`);
            topRegionsList.append(li);
        });
    }).fail(function() {
        console.error('Error al cargar data.json');
    });
});
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="../css/style.css"> <!-- Ruta actualizada al archivo CSS -->
    <title>Top 10 Regiones con mayor nmero de Confirmados</title>
</head>
<body>
    <h1>Top 10 Regiones por Confirmados</h1>
    <ul id="topRegionsList"></ul>

    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script src="../js/top_regions.js"></script>
</body>
</html>
```

4.6. Actividad

- Visualice un gráfico en el tiempo de los valores para la región de Arequipa

```
google.charts.load('current', {'packages':['corechart']});
google.charts.setOnLoadCallback(drawChart);

function drawChart() {
    $.getJSON('../data/data.json', function(data) {
        const arequipaData = data.find(region => region.region === 'Arequipa');
        if (!arequipaData) {
            console.error('No se encontr la regin Arequipa en data.json');
            return;
        }

        const chartData = [['Fecha', 'Confirmados']];
        arequipaData.confirmed.forEach(entry => {
            const dateParts = entry.date.split('-');
            const date = new Date(dateParts[2], dateParts[1] - 1, dateParts[0]);
            chartData.push([date, parseInt(entry.value)]);
        });

        const dataTable = google.visualization.arrayToDataTable(chartData);

        const options = {
            title: 'Casos Confirmados en Arequipa a lo largo del tiempo',
            hAxis: {
                title: 'Fecha',
                format: 'dd/MM/yyyy'
            },
            vAxis: {
                title: 'Confirmados'
            },
            legend: { position: 'bottom' }
        };
```

```
        const chart = new
            google.visualization.LineChart(document.getElementById('chart_div'));
        chart.draw(dataTable, options);
    }).fail(function() {
        console.error('Error al cargar data.json');
    });
}
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="../css/style.css">
    <title>Grfico de Arequipa</title>
    <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script src="../js/aqp_chart.js"></script>
</head>
<body>
    <h1>Grfico de Arequipa</h1>
    <div id="chart_div"></div>
</body>
</html>
```

4.7. Actividad

- Haga gráficos comparativos entre regiones usando líneas

```
google.charts.load('current', {'packages':['corechart']});
google.charts.setOnLoadCallback(drawChart);

function drawChart() {
    $.getJSON('../data/data.json', function(data) {
        const regionsToCompare = ['Amazonas', 'Lima', 'Arequipa', 'Cusco', 'Piura'];
        const chartData = [['Fecha', ...regionsToCompare]];

        const dates = new Set();
        data.forEach(region => {
            if (regionsToCompare.includes(region.region)) {
                region.confirmed.forEach(entry => dates.add(entry.date));
            }
        });

        const sortedDates = Array.from(dates).sort((a, b) => {
            const dateA = new Date(a.split('-').reverse().join('-'));
            const dateB = new Date(b.split('-').reverse().join('-'));
            return dateA - dateB;
        });

        sortedDates.forEach(date => {
            const row = [new Date(date.split('-').reverse().join('-'))];
            regionsToCompare.forEach(regionName => {
                const region = data.find(r => r.region === regionName);
```

```
            const entry = region.confirmed.find(e => e.date === date);
            row.push(entry ? parseInt(entry.value) : 0);
        });
        chartData.push(row);
    });

    const dataTable = google.visualization.arrayToDataTable(chartData);

    const options = {
        title: 'Comparacin de Confirmados entre Regiones',
        hAxis: {
            title: 'Fecha',
            format: 'dd/MM/yyyy'
        },
        vAxis: {
            title: 'Confirmados'
        },
        legend: { position: 'bottom' }
    };

    const chart = new
        google.visualization.LineChart(document.getElementById('chart_div'));
    chart.draw(dataTable, options);
}).fail(function() {
    console.error('Error al cargar data.json');
});
}
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="../css/style.css">
    <title>Comparacin de Regiones</title>
    <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script src="../js/compare_regions.js"></script>
</head>
<body>
    <h1>Comparacin de Regiones</h1>
    <div id="chart_div"></div>
</body>
</html>
```

4.8. Actividad

- Visualice un gráfico comparativo del crecimiento en regiones excepto Lima y Callao

```
google.charts.load('current', {'packages':['corechart']});
google.charts.setOnLoadCallback(drawChart);

function drawChart() {
    $.getJSON('../data/data.json', function(data) {
```

```
        let nombres = [];
        const chartData = [['Fecha', 'Confirmados']];
        data.forEach(regiones => {
            if (regiones.region !== 'Lima' && regiones.region !== 'Callao') {
                nombres.push(regiones.region);
                regiones.confirmed.forEach(entry => {
                    const fecha = entry.date.split('-');
                    const date = new Date(fecha[2], fecha[1] - 1, fecha[0]);
                    chartData.push([date, parseInt(entry.value)]);
                });
            }
        });

        const dataTable = google.visualization.arrayToDataTable(chartData);

        const options = {
            title: 'Grfico Comparativo Excluyendo Lima y Callao',
            hAxis: {
                title: 'Fecha',
                format: 'dd/MM/yyyy'
            },
            vAxis: {
                title: 'Casos'
            },
            legend: { position: 'bottom' }
        };

        const chart = new
            google.visualization.LineChart(document.getElementById('chart_div'));
        chart.draw(dataTable, options);
    }).fail(function() {
        console.error('Error al cargar data.json');
    });
}
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="../css/style.css">
    <title>Grfico sin Lima ni Callao</title>
    <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script src="../js/Lima.js"></script>
</head>
<body>
    <h1>Grfico  sin Lima y Callao</h1>
    <div id="chart_div"></div>
</body>
</html>
```

4.9. Actividad

- Haga gráficos comparativos entre regiones elegidas por el usuario.

```
google.charts.load('current', {'packages':['corechart']});
google.charts.setOnLoadCallback(init);

function init() {
    document.getElementById('myForm').addEventListener('submit', function(event) {
        event.preventDefault();
        const formData = new FormData(this);
        const region1 = formData.get('string1');
        const region2 = formData.get('string2');

        $.getJSON('../data/data.json', function(data) {
            drawChartData(data, region1, region2);
        }).fail(function() {
            console.error('Error al cargar data.json');
        });
    });
}

function drawChartData(data, region1, region2) {
    const dataRegion1 = data.find(region => region.region === region1);
    const dataRegion2 = data.find(region => region.region === region2);

    if (!dataRegion1 || !dataRegion2) {
        console.error('Datos no encontrados para las regiones especificadas');
        return;
    }

    const chartData = [['Fecha', region1, region2]];

    dataRegion1.confirmed.forEach(entry => {
        const dateParts = entry.date.split('-');
        const date = new Date(dateParts[2], dateParts[1] - 1, dateParts[0]);
        const value = parseInt(entry.value);
        const row = [date, value, null];
        chartData.push(row);
    });

    dataRegion2.confirmed.forEach(entry => {
        const dateParts = entry.date.split('-');
        const date = new Date(dateParts[2], dateParts[1] - 1, dateParts[0]);
        const value = parseInt(entry.value);
        const row = [date, null, value];
        chartData.push(row);
    });

    const dataTable = google.visualization.arrayToDataTable(chartData);

    const options = {
        title: 'Casos Confirmados de: ' + region1 + ' y ' + region2,
        hAxis: {
            title: 'Fecha',
            format: 'dd/MM/yyyy'
```

```
        },
        vAxis: {
            title: 'Confirmados'
        },
        legend: { position: 'bottom' }
    };

    const chart = new google.visualization.LineChart(document.getElementById('chart_div'));
    chart.draw(dataTable, options);
}
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="../css/style.css">
    <title>Enviar Strings a JS</title>
    <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
</head>
<body>
    <form id="myForm">
        <input type="text" name="string1" placeholder="Escribe la primera region">
        <input type="text" name="string2" placeholder="Escribe la segunda region">
        <button type="submit">Enviar</button>
    </form>
    <div id="chart_div"></div>

    <script src="../js/comparativo.js"></script>
</body>
</html>
```

4.10. Actividad

- Visualice un gráfico comparativo del crecimiento en regiones excepto Lima y Callao, mostrando el número de confirmados por cada día

```
google.charts.load('current', {'packages':['corechart']});
google.charts.setOnLoadCallback(drawChart);

function drawChart() {
    $.getJSON('../data/data.json', function(data) {
        const regionesFiltradas = data.filter(region => region.region !== 'Lima' &&
            region.region !== 'Callao');
        let fechas = new Set();

        regionesFiltradas.forEach(region => {
            region.confirmed.forEach(entry => fechas.add(entry.date));
        });

        fechas = Array.from(fechas).sort((a, b) => new Date(a) - new Date(b));

        let chartData = [['Fecha', ...regionesFiltradas.map(region => region.region)]];
```

```
        fechas.forEach(fecha => {
            let row = [new Date(fecha.split('-')[2], fecha.split('-')[1] - 1,
                fecha.split('-')[0])];
            regionesFiltradas.forEach(region => {
                let entry = region.confirmed.find(entry => entry.date === fecha);
                row.push(entry ? parseInt(entry.value) : null);
            });
            chartData.push(row);
        });

        const dataTable = google.visualization.arrayToDataTable(chartData);


        const colors = [
            '#FF5733', '#33FF57', '#3357FF', '#FF33A1', '#A133FF',
            '#33FFA1', '#FFBD33', '#3385FF', '#33FFBD', '#FF5733', '#BD33FF',
            '#33FF85', '#FF3380','#80FF33', '#5733FF', '#FF8333', '#33FFDD',
            '#FF33FF', '#33A1FF', '#FF3357', '#85FF33', '#FF5733', '#33FF57',
        ];

        const options = {
            title: 'Grfico Comparativo Excluyendo Lima y Callao',
            hAxis: {
                title: 'Fecha',
                format: 'dd/MM/yyyy'
            },
            vAxis: {
                title: 'Casos'
            },
            legend: { position: 'bottom' },
            colors: colors.slice(0, regionesFiltradas.length)
        };

        const chart = new
            google.visualization.LineChart(document.getElementById('chart_div'));
        chart.draw(dataTable, options);
    }).fail(function() {
        console.error('Error al cargar data.json');
    });
}
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="../css/style.css">
    <title>Grfico sin Lima ni Callao por da</title>
    <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script src="../js/regiones.js"></script>
</head>
<body>
    <h1>Grfico  sin Lima y Callao por da</h1>
```

```
    <div id="chart_div"></div>
</body>
</html>
```

4.10.1. Capturas del funcionamiento

Figura 9

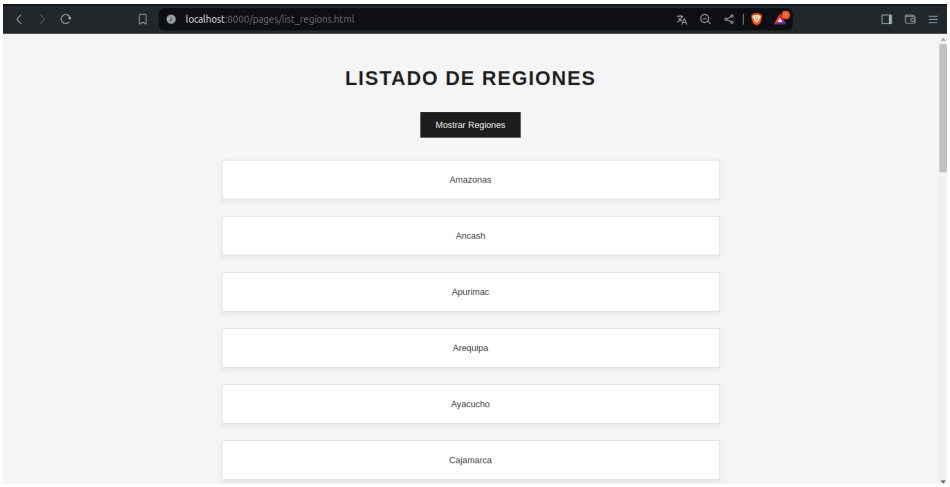

Figura 10

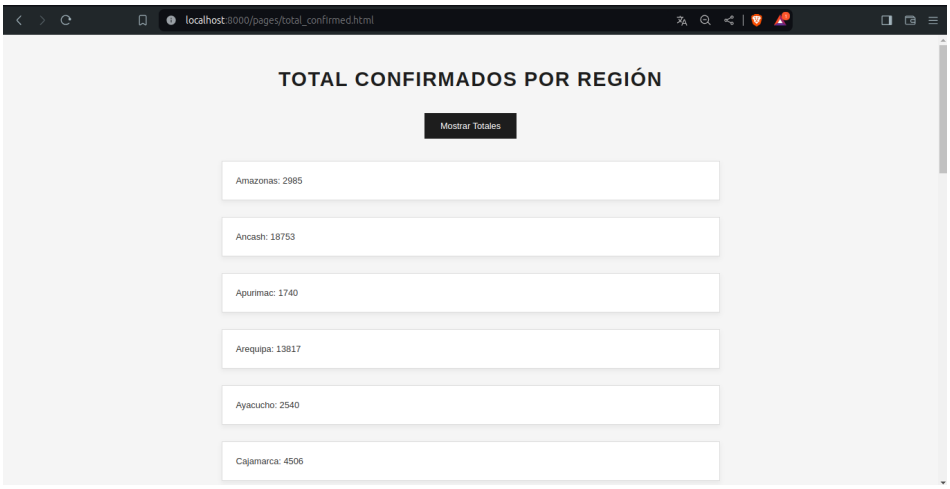

Figura 11

Figura 12

Figura 13

Figura 14

Figura 15

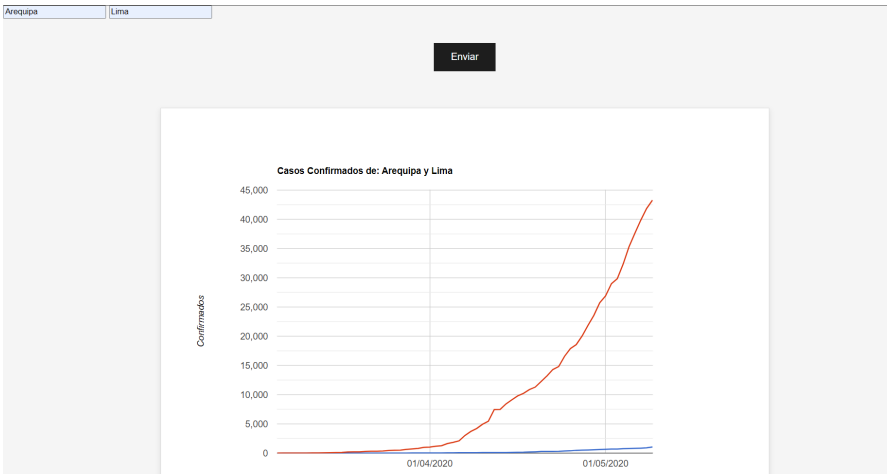

Figura 16

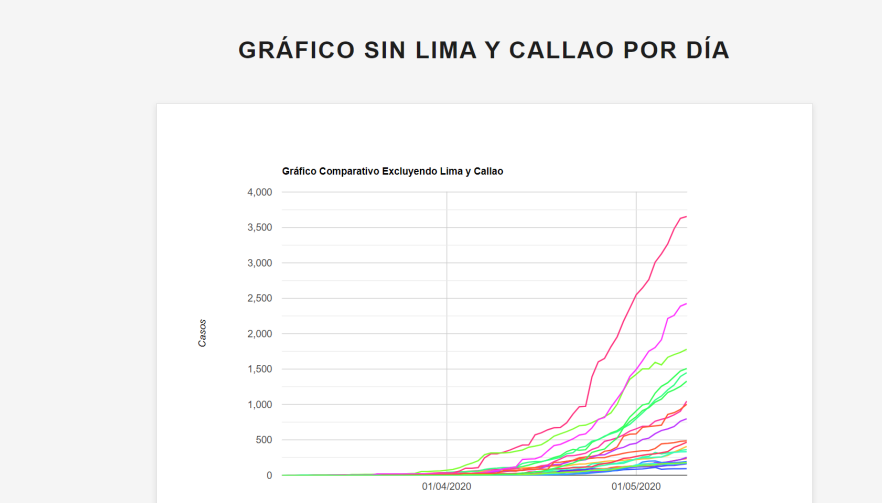
GRÁFICO SIN LIMA Y CALLAO POR DÍA
Gráfico Comparativo Excluyendo Lima y Callao
Casos
4,000
3,500
3,000
2,500
2,000
1,500
1,000
500
0
01/04/2020
01/05/2020

Figura 17