Examen 01

Luis D Palomo, Mario Marin Villalobos, José Redondo

La finalidad de este documento es poder documentar la funcionalidad básica de cada método dentro de cada módulo creado para la solución del examen.

Se explicará a continuación cada uno de ellos para un mejor entendimiento del programa junto con una imagen que representa cada método o función realizada dentro de la solución.

Estilos.css

```
body {
    background-color:lightgrey;
    background-size: cover;
    background-repeat: no-repeat;
    box-sizing: border-box;
    image-resolution: 500dpi;
}
```

Se configura el color para la página en general y el tamaño que deberá tener.

```css
.menu-bar {
    width: 100%;
    float: left;
    margin-top: 10px;
    background-color:black;
}

    .menu-bar ul {
        display: inline-flex;
        list-style: none;
        margin-left: 15px;
    }

        .menu-bar ul li {
            width: 120px;
            margin: 10px;
            padding: 10px;
            left: 0;
        }

            .menu-bar ul li a {
                text-decoration: none;
                left: 0;
            }

            .active, .menu-bar ul li:hover {
                border-radius: 3px;
                background-color: #3F60A5;
                text-transform: capitalize;
                color:white;
            }

                .active, .menu-bar ul li:hover a {
                    border-radius: 3px;
                    text-transform: capitalize;
                    color: white;
                }
```

Se crea la barra de navegación con su color y alineación, así como el estilo que tendrá cada opción dentro de ella, incluyendo los cambios de color y estilo al posicionar el mouse sobre cada opción. Se ajusta de igual manera la posición dentro de la barra de navegación.

```
ul {
    padding: 0;
    margin: 0 0 0 10px;
}

li {
    color:dimgrey;
    font-family: 'Century Gothic';
    font-size: 15px;
    margin-right: 10px;
    margin-top: 10px;
}
```

Se da estilo a cada opción de la barra principal, su espaciado, tipo de letra, márgenes, color, etc.

```
.sub-menu2 {
    display: none;
    background-color: #3F60A5;
    color: dimgrey;
}

.menu-bar ul li:hover .sub-menu2 {
    display: block;
    position: absolute;
    background: rgb(0,0,0);
    margin-top: 10px;
    margin-left: -40px;
}

    .menu-bar ul li:hover .sub-menu2 ul {
        display: block;
        margin: 10px;
    }
```

Aquí se da estilo a las opciones que se despliegan de la barra principal, incluyendo color, posición y efectos cuando el mouse se posicione sobre cada opción.

Layout.html

Aquí s

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>El Padron Nacional</title>
        <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
    </head>
    <body>
        {% include 'includes/_navbar.html' %}
        <div class="container">
        {% block body %}{% endblock %}
        </div>


        <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
    </body>
</html>
```

Esta es nuestra pagina html principal de la cual las demás heredaran estilo y en este caso también el título.

_navbar.html

```
<nav class="navbar navbar-default">
     <title>Padron Electoral</title>
     <link rel="stylesheet" href="{{ url_for('static', filename='css/Estilos.css') }}">

     <div class="menu-bar">
        <ul>
            <li><a href = "/">Principal</a></li>
            <li><a href = "/disxprovincia"> Distritos por Provincia</a></li>
            <li><a href = "/disxcanton"> Distritos por Canton</a></li>
            <li><a href = "/xProvincia"> Votantes por Provincia</a></li>
            <li><a href = "/xBusca"> Busca Votantes por Cedula</a></li>
            <li><a href = "/xSexo"> Votantes por Sexo</a>
                <div class="sub-menu2">
                    <ul>
                        <li><a href = "/xSexo"> Listado de Votantes separado por Sexo</a>
                        <li><a href="/xMasculino">Listado de Votantes Masculinos</a></li>
                        <li><a href="/xFemenino">Listado de Votantes Femeninos</a></li>
                    </ul>
                </div>
            </li>
            <li><a href = "/about"> Acerca de Nosotros</a></li>
        </ul>
      </div>

</nav>
```

Esta es nuestra navbar principal desde donde se llaman las demás páginas del programa. La misma llama al .css para poder obtener el estilo deseado.

disxcanton.html, disxprovincia.html, xBusca.html, xFemenino.html, xMasculino.html, xProvincia.html, xResultado.html, xSexo.html

```
{% extends 'layout.html' %}

{% block body %}
    <div class="jumbotrom text-center">
        <h1>Examen 01</h1>
        <h1>Padron Nacional</h1>
        <h1>Cantidad Distritos por Canton</h1>
    </div>

{% for table in tables %}
            {{titles[loop.index]}}
            {{ table|safe }}
{% endfor %}


{% endblock %}
```

En cada pantalla básicamente es donde se visualizan los resultados de los métodos con el mismo nombre, con un encabezado estándar y que nos permite debajo del mismo ver la data recolectada de acuerdo con la función implementada en forma de tabla.

xBusca.html

```
{% extends 'layout.html' %}

{% block body %}
    <div class="jumbotrom text-center">
        <h1>Examen 01</h1>
        <h1>Padron Nacional</h1>
        <h1>Busca Votantes por Cedula</h1>
    </div>

    <form action="#" method="POST">
        Numero de Cedula:<br>
        <input type="text" name="Cedula"><br>
        <input type="submit" value="Aceptar">
    </form>


{% endblock %}
```

Esta pantalla posee el mismo encabezado estándar de las anteriores con la diferencia de poder desde acá realizar una búsqueda por cedula sobre el padrón que nos desplegara en pantalla la información solicitada.

```python
@app.route('/disxprovincia', methods=("POST", "GET"))
def html_disxprovincia():
    df1 = pd.read_csv('./Data/Distelec.csv',encoding='ISO-8859-1')
    df1 = manipuladatos.agregaColumnaDistrito(df1)
    df1 = manipuladatos.DistritoPorProvincia(df1)
    return render_template('disxprovincia.html',  tables=[df1.to_html(classes='table table-striped')], titles=df1.columns.values)
```

Se encarga de mostrar la tabla de distrito por provincia de acuerdo con el archivo .csv agregando cada una de las columnas en base a lo solicitado en la consulta. Relacionado con otros métodos muestra en el html la información solicitada.

```python
@app.route('/disxcanton', methods=("POST", "GET"))
def html_disxcanton():
    df1 = pd.read_csv('./Data/Distelec.csv',encoding='ISO-8859-1')
    df1 = manipuladatos.agregaColumnaDistrito(df1)
    df1 = manipuladatos.DistritoPorCanton(df1)
    return render_template('disxcanton.html',  tables=[df1.to_html(classes='table table-striped')], titles=df1.columns.values)
```

Se encarga de mostrar la tabla de distrito por cantón de acuerdo con el archivo .csv agregando cada una de las columnas en base a lo solicitado en la consulta. Relacionado con otros métodos muestra en el html la información solicitada

```python
@app.route('/xSexo', methods=("POST", "GET"))
def html_xSexo():
    df1 = pd.read_csv('./Data/padron.csv',encoding='ISO-8859-1')
    df1 = manipuladatos.agregaColumnas(df1)
    df1 = manipuladatos.votantesPorSexo(df1)
    return render_template('xSexo.html',  tables=[df1.to_html(classes='table table-striped')], titles=df1.columns.values)
```

Se encarga de mostrar la tabla de votantes por sexo de acuerdo con el archivo .csv agregando cada una de las columnas en base a lo solicitado en la consulta. Relacionado con otros métodos muestra en el html la información solicitada

```python
@app.route('/xProvincia', methods=("POST", "GET"))
def html_xProvinca():
    df1 = pd.read_csv('./Data/padron.csv',encoding='ISO-8859-1')
    df1 = manipuladatos.agregaColumnas(df1)
    df1 = manipuladatos.votantesPorProvincia(df1)
    return render_template('xProvincia.html',  tables=[df1.to_html(classes='table table-striped')], titles=df1.columns.values)
```

Se encarga de mostrar la tabla de votantes por Provincia de acuerdo con el archivo .csv agregando cada una de las columnas en base a lo solicitado en la consulta. Relacionado con otros métodos muestra en el html la información solicitada

```python
@app.route('/xFemenino', methods=("POST", "GET"))
def html_xFemenino():
    df1 = pd.read_csv('./Data/padron.csv',encoding='ISO-8859-1')
    df1 = manipuladatos.agregaColumnas(df1)
    df1 = manipuladatos.votantesFemenino(df1)
    return render_template('xFemenino.html',  tables=[df1.to_html(classes='table table-striped')], titles=df1.columns.values)
```

Se encarga de mostrar la tabla de votantes Femeninas de acuerdo con el archivo .csv agregando cada una de las columnas en base a lo solicitado en la consulta. Relacionado con otros métodos muestra en el html la información solicitada

```python
@app.route('/xMasculino', methods=("POST", "GET"))
def html_xMasculino():
    df1 = pd.read_csv('./Data/padron.csv',encoding='ISO-8859-1')
    df1 = manipuladatos.agregaColumnas(df1)
    df1 = manipuladatos.votantesMasculino(df1)
    return render_template('xMasculino.html',  tables=[df1.to_html(classes='table table-striped')], titles=df1.columns.values)
```

Se encarga de mostrar la tabla de votantes Masculinos de acuerdo con el archivo .csv agregando cada una de las columnas en base a lo solicitado en la consulta. Relacionado con otros métodos muestra en el html la información solicitada

```python
@app.route('/about', methods=("POST", "GET"))
def html_about():
    return render_template('about.html')
```

Se encarga de redireccionar hacia el html de about donde se muestra la información de los integrantes del grupo que realizaron el examen.

```python
@app.route('/xBusca',  methods=("POST", "GET"))
def html_xBusca():
    try:
        if request.method == 'POST':
            Ced = request.form["Cedula"]
            intCed = int(Ced)
            df1 = pd.read_csv('./Data/padron.csv',encoding='ISO-8859-1')
            df1 = manipuladatos.agregaColumnas(df1)
            df1 = manipuladatos.ListaVotantePorCedula(df1,intCed)
            return render_template('xResultado.html',  tables=[df1.to_html(classes='table table-striped')], titles=df1.columns.values)

        return render_template('xBusca.html')

    except Exception as e:

        return render_template('xBusca.html')
```

Se encarga de redireccionar hacia el html de búsqueda por cedula donde se declara la variable, se solicita el dato, se verifica si la información está dentro del Padrón y se muestra en pantalla.

manipuladatos.py

```python
def agregaColumnas(miPadron):
#agrega columna provincia y usando los numeros de la cedula convierte ese numero en nombre de provincias
    miPadron['Provincia'] = miPadron['Cedula'].astype(str)
    miPadron['Provincia'] = miPadron['Provincia'].str[0:1]
    miPadron['Provincia'] = miPadron['Provincia'].replace({'1':'San Jose', '2':'Alajuela', '3':'Cartago','4':'Heredia','5':'Guanacaste','6':'Puntare
    miPadron['Count'] = 1
    return (miPadron)
```

Se realiza la carga de data y realiza una comparación del inicio de la cedula con la provincia a la cual estaría asociada para poder mostrar los datos.

```python
#Calcula votantes por provincia
def votantesPorProvincia(miPadron):
    resultado = miPadron['Provincia'].value_counts().reset_index()
    resultado.columns = ['Provincia','Cantidad de Votantes']
    return (resultado)
```

Realiza una verificación del padrón, cada una de las provincias y realiza un conteo de los resultados para poder desplegar la cantidad de votantes por provincia.

```python
def ListaVotantePorCedula(miPadron,CedId):
    resultado = miPadron.loc[miPadron['Cedula'] == CedId]
    return (resultado)
```

Realiza una búsqueda de la cedula ingresada en el html para verificar que la persona se encuentre dentro del padrón y así poder mostrar los datos.

```python
def votantesPorSexo(miPadron):
    miPadron['Sexo'] = miPadron['Sexo'].astype(str)
    miPadron['Sexo'] = miPadron['Sexo'].replace({'1':'Masculino','2':'Femenino'})
    resultado = miPadron['Sexo'].value_counts().reset_index()
    resultado.columns = ['Sexo','Cantidad']
    return (resultado)
```

Realiza una búsqueda dentro del padrón para poder desplegar el resultado de cuantas personas tanto masculinas como femeninas se encuentran dentro del archivo

```
#Lista Votantes Mujeres
def votantesFemenino(miPadron):
    miPadron['Sexo'] = miPadron['Sexo'].astype(str)
    miPadron['Sexo'] = miPadron['Sexo'].replace({'1':'Masculino','2':'Femenino'})
    resultado = miPadron.loc[miPadron['Sexo'] =="Femenino"]
    return (resultado.head(50))
```

Realiza una búsqueda dentro del padrón para poder desplegar el resultado de cuantas personas mujeres se encuentran dentro del archivo, arrojando un resultado de máximo 50 registros.

```
#cuenta distritos electorales por provincia
def DistritoPorProvincia(miDistrito):
    resultado = miDistrito['Provincia'].value_counts().reset_index()
    resultado.columns = ['Provincia','Cantidad Distritos']
    return (resultado)
```

Realiza una búsqueda dentro del padrón para poder desplegar el resultado de cuantas personas hombres se encuentran dentro del archivo, arrojando un resultado de máximo 50 registros.

```
#cuenta distritos electorales por Canton
def DistritoPorCanton(miDistrito):
    resultado = miDistrito['Canton'].value_counts().reset_index()
    resultado.columns = ['Canton','Distritos Electorales']
    return (resultado)
```

Realiza una búsqueda dentro del padrón para poder desplegar el resultado de cuantas personas por distrito se encuentran dentro del archivo.