DOCUMENTACIÓN TÉCNICA PARA DESARROLLADORES

DESARROLLADA POR:

SDE. JESSICA MARTÍNEZ

TEMA:

ETIQUETA ISMAP

- ¿CÓMO FUNCIONA?
- ¿SIGUE SIENDO UTILIZADA EN WEBS MORDERNAS?
- LIMITACIONES DE ISMAP
- COMPARACIÓN CON ALTERNATIVAS MODERNAS
- FORMA DE USO
- BREVE INTRODUCCIÓN A PHP

Contenido

ALCANCE DEL DOCUMENTO

Este documento esta dirigido a desarrolladores con conocimientos básicos de HTML interesados en aprender del uso de una etiqueta actualmente obsoleta pero que aporta conocimiento histórico de este lenguaje.

No se profundiza en PHP avanzado, únicamente se toca superficialmente etiquetas de apertura y cierre, etiquetas para condiciones y lógica de excepciones.

El objetivo es comprender ISMAP desde su funcionamiento real dentro del lado del servidor y que no quede como una etiqueta simple ya que esta fue una etiqueta compleja para webs antiguas y por lo cual se encuentra obsoleta.

Se recuerdan conceptos matemáticos básicos nada de complejos como es la obtención de mínimos y máximos de una serie de números, en este caso en coordenadas x e y.

Asimismo, se brinda una pequeña explicación para la ejecución de códigos PHP desde un servidor externo o como un proyecto de GitHub.

INTRODUCCIÓN

En este documento se presenta el proceso de obtención para las coordenadas del mapa usando el atributo **ISMAP.**

Su uso se aborda de forma comprensiva, considerando que actualmente esta técnica se encuentra obsoleta y ya no es utilizada en webs modernas debido a su limitada capacidad de manipulación, baja precisión y escasa accesibilidad para la interacción con el usuario.

ISMAP permite que una imagen funcione como un mapa interactivo, pero únicamente del lado del servidor. Su funcionamiento depende de que la etiqueta <*img*> se encuentre de una etiqueta <*a*>*;* de lo contrario, no se ejecuta correctamente.

Para hacer uso de ISMAP es necesario apoyarse en lenguajes de programación que se encuentren dentro del servidor, ya que esta técnica no es procesada por HTML ni ejecutada directamente por el navegador utilizado por el usuario.

En este caso se emplea **PHP** para recuperar las coordenadas que el navegador envía al servidor, permitiendo evaluar dichas coordenadas mediante condiciones previamente definidas y decidir la acción correspondiente.

PHP es un lenguaje sencillo de comprender, por lo que no se profundizara extensamente en él. Únicamente se abordarán los conceptos necesarios para comprender su función dentro de este proceso.

Cabe recalcar que la técnica utilizada en este documento es de elaboración propia, basada en fundamentos matemáticos y desarrollada tras un arduo trabajo de análisis para determinar la mejor forma de implementación. Se espera que este material resulte de utilidad para desarrolladores que requieran comprender o aplicar este enfoque.

CONTEXTO HISTORICO DE ISMAP

ISMAP fue una etiqueta que permitía indicar que la imagen cargada era un mapa del lado del servidor, es decir una imagen que contenía muchas zonas clicables "invisibles" para el lado del usuario; pero conocidas para el servidor definidas en un lenguaje de ejecución de su mismo lado.

Es un atributo **booleano** es decir no tiene valor o valores asociados a él, es decir su sola presencia indica que es verdadero (true) y su ausencia que es falso (false), cuando aparecía indicaba que la imagen pertenecía a un mapa de imagen.

Este atributo solo era permitido en el caso estricto de que la etiqueta <*img*> que contiene a la imagen sea descendiente de una etiqueta <*a*> y que esta contenga un *href* válido.

La sintaxis para el uso de este atributo es:

```
<a href=" link_valido>
        <img src= "ruta/de/la/imagen"
    alt= "Descripción breve de la imagen"
    ismap />
<a/>
```

En la actualidad se encuentra en desuso por su poca compatibilidad con el usuario para visualizar o identificar las zonas clicables que se encuentran dentro de la imagen, funcionando únicamente como conocimiento histórico de webs antiguas.

LIMTACIONES DE ISMAP

Los mapas de imágenes del lado del servidor no siempre son prácticos ni la mejor opción en todas las situaciones.

- **Son no accesibles por defecto.** Ya que, al estar asociado con un mapa del lado del servidor, el usuario no tiene control sobre él. El problema es que, si un usuario no puede usar un ratón o un panel táctil (porque usa teclado, comandos de voz o lectores de pantalla), no hay forma de que este pueda enviar coordenadas precisas de un clic ya que el teclado solo permite activar el enlace completo, sin posibilidad de seleccionar coordenadas especificas dentro de la imagen, pero no puede simular un clic en algún pixel de la imagen. Para las personas que son ciegas y usan lectores de pantalla, esto se vuelve un agujero negro porque este se vuelve invisible funcionalmente; el lector de pantalla solo le dirá, al usuario que hay una imagen con un enlace, pero el usuario no tiene forma de saber que partes de la imagen son

clicables o a donde lo lleva cada zona ya que no usa títulos o textos alternativos para indicarle si no que esa lógica esta escondida en el servidor.

- **No manejable desde HTML/CSS.** Ni HTML ni CSS tienen control directo sobre ISMAP, todo depende del servidor. HTML únicamente indica que la imagen pertenece a un mapa de imagen de lado del servidor mediante el atributo *ismap,* sin definir las áreas ni su comportamiento y CSS se encarga de darle estilo, tamaño o forma, pero su funcionalidad es ajena a ellos, esta bajo el control del servidor que está sujeto a condiciones ya dadas, recibe una coordenada clicada por el usuario, con un lenguaje que se ejecuta dentro del servidor la recupera, la evalúa y decide qué hacer con ella.

- **Dependencia total del servidor.** Como venimos mencionando anteriormente este atributo está sujeto al servidor, ya que él es quien recibe, evalúa y ordena que hacer con la información recibida, el usuario no tiene visibilidad ni control sobre la lógica que determina el destino de cada coordenada.

- **Difícil mantenimiento.** Para estar manteniéndolo funcional y actualizado se vuelve complicado ya que hay que volver a evaluar las zonas para definir las áreas que contendrán información y las que no, si se cambia el tamaño, se modifica algo, los rangos condicionados dejan de coincidir automáticamente.

- **No escalable.** ISMAP se rompe visual y funcionalmente en el diseño web moderno (responsivo: escritorio, tabletas, celulares, etc.). Ya que hoy en día las imágenes suelen tener un diseño fluido para que se vean bien en cualquier dispositivo, entonces ISMAP envía coordenadas exactas basadas en el archivo de imagen original; por ejemplo un clic en el pixel 445,109 en el tamaño de la imagen mostrada en una computadora, si la imagen ahora se observa desde un celular se encoge para caber en él, y el pixel antes clicado 445,109 ya no esta representando al mismo objeto que en la versión de escritorio, puede que hasta haya desaparecido porque no ocupa 400 pixeles y el servidor sigue esperando coordenadas fijas pero la imagen que ve el usuario ha cambiado de tamaño y así el mapa deja de coincidir con lo que el usuario ve y el servidor quiere mostrar.

Otro contra es para las pantallas táctiles es más difícil calcular la exactitud de la coordenada donde se hará clic, en una computadora un puntero tiene la medida exacta de un pixel, hay más precisión; en un móvil el puntero de este, un dedo humano, viene a ser más grande y por lo tanto cubre un área más grande lo que podría enviar una coordenada errónea al servidor el cual devolvería un error o un enlace erróneo.

Y, por último, otro problema de escalabilidad es que en ISMAP la lógica de que a qué zona lleva qué página, reside en el servidor, si decidimos cambiar el diseño de la imagen, añadirle algún detalle, botón u otra función, tenemos que actualizar el archivo de imagen y redefinir los scripts del servidor, o sea crear áreas nuevas para generar nuevos rangos y nuevas condiciones.

Estas limitaciones explican porque ISMAP fue reemplazado por los mapas de imagen del lado del cliente (*usemap+map*), combinados con HTML semántico, CSS y consideraciones de accesibilidad modernas.

COMPARACIÓN CON ALTERNATIVAS MODERNAS

- **USEMAP + MAP:**

Hoy en día, en webs modernas, la forma típica y más accesible para manejar mapas de imágenes es del lado del cliente (en el navegador) usando el atributo *usemap* y la etiqueta *map* para especificar una colección de mapas de áreas definidas. Donde el atributo *map* usa otros atributos como ***shape*** para indicar la forma del área, ***coords*** donde se colocan absolutamente todas las coordenadas que pertenecen a esa área definida únicamente separadas por comas, un ***href*** que dirigirá al usuario a la pagina designada de acuerdo al área donde haga clic, son los atributos más comunes que se encuentran y otros que también son muy útiles y resultan muy profesionales son ***target, rel, title, alt, etc.***

La sintaxis queda así:

```
<img src="ruta/de/la/imagen” usemap="#mapa_name"/>
    <map name="mapa_name”>
            <area
             shape="poly"
             coords="todas,las,coordenadas,del,área"
            href="link_valido"
            target="_self"
            title="Título que se muestra al pasar el cursor por el área"
            alt="Texto alternativo que describe la imagen">
    </map>
```

- **OTRAS ALTERNATIVAS:**

En el mundo del desarrollo web moderno, soluciones como mapas de imágenes del lado del cliente (usemap), gráficos SVG interactivos, Canvas, JS+eventos, entre otros han reemplazado completamente a ISMAP, debido a que ofrecen escalabilidad, accesibilidad y control desde HTML y CSS.

NOTA: Estas alternativas se mencionan solo como referencia y no se desarrollan en este documento.

FLUJO GENERAL DEL PROCESO (DIAGRAMA TEXTUAL)

1. Usuario hace clic en un punto de la imagen.
2. Navegador envía coordenadas del punto a servidor.
3. Servidor recupera los puntos de la coordenada.
4. PHP evalúa ambos puntos de la coordenada.
5. Verifica a que rango pertenecen.
6. Se toma la decisión.
7. Usuario es redirigido a la página definida para ese rango.

ERRORES COMUNES

A la hora de usar el atributo ISMAP podemos cometer errores muy comunes tales como:

- Olvidar que ISMAP solo funciona si la etiqueta <*img*> es descendiente de una etiqueta <*a*>.
- Creer que el navegador es quien recibe y evalúa las coordenadas.
- Confundir las coordenadas X e Y.
- Olvidar que puede haber zonas que sufren traslapes con otras.
- Creer que HTML es quien maneja la lógica y toma decisiones.
- Tratar de controlar con CSS los rangos.

Para una compresión detallada de cada paso (obtención de coordenadas, rangos, manejo de traslapes, véase el anexo A).

CONCLUSIONES

ISMAP en su momento fue un atributo útil para el manejo de mapa de imágenes del lado servidor, con sus funciones limitadas logro solventar muchas necesidades, pero al igual que

todo, cada vez se necesita ir avanzando y simplificando todo, fue quedando casi obsoleta con la aparición de **usemap** y muchas más alternativas modernas a la fecha de hoy que la han dejado como concepto histórico.

Con esta documentación se logró explicar su contexto histórico, sintaxis, limitaciones, cómo funcionaba, los errores comunes que se podían llegar a cometer al usarlo.

Se aprendió como funcionaba realmente, todo el trabajo lógico que había detrás de un mapa de imagen en la época del uso de ISMAP, una formula personal para solucionar problemas que a medida iban apareciendo representaban un reto más.

A pesar de ser un atributo muy complicado el cual realmente es la última opción para mapas de imágenes en el desarrollo web moderno, hay excepciones donde podría resultar un poco útil como en el caso de navegadores para terminales de comandos (como Lynx) o dispositivos industriales muy viejos que no entienden o soportan HTML moderno, CSS y mucho menos SVG, si estos dispositivos solo pueden mostrar un imagen y enviar un clic, ismap permite que toda la inteligencia resida en el servidor y así no se sobrecarga al navegador y este no tiene que calcular nada, solo reportar donde se hizo el clic.

Y aún así estos casos no justifican el uso de ISMAP porque todo se ha modernizado e incluso hasta los dispositivos móviles son muy potentes que soportan que el navegador realice todo el trabajo en lugar de cargarlo en el servidor y ralentizar su ejecución.

ANEXOS

Los anexos presentados tienen como objetivo ampliar la explicación técnica del funcionamiento de ISMAP sin afectar la claridad del desarrollo teórico y se dividen en varios para una explicación detallada de los desafíos a los cuales nos podemos enfrentar en el avance de su uso con conceptos matemáticos y del uso de un lenguaje ejecutado en el servidor.

ANEXO A: METODOLOGÍA PARA OBTENCIÓN DE COORDENADAS DE ZONAS CON UNA HERRAMIENTA EXTERNA.

Cargamos una imagen en el editor de texto con la sintaxis mostrada en la parte del contexto histórico de ISMAP.

Para obtener las coordenadas de cada región, nos auxiliamos de una herramienta en línea:

https://www.image-map.net/

¿Cómo usar esta herramienta?

- Cargamos la imagen desde nuestro explorador o la web con las mismas medidas de ancho y alto que están especificadas en el editor de texto o en la clase del archivo externo en CSS, estas medidas deben de coincidir para asegurarnos de que los rangos sean los mismos.

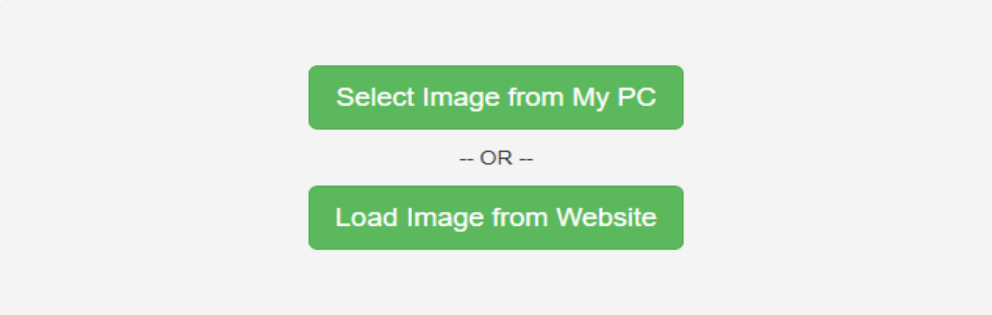

Ilustración 1. Pantalla principal de image-map para subir la imagen.

- Una vez cargamos la imagen nos vamos a la parte de debajo de la imagen donde se ubica la barra de herramientas.

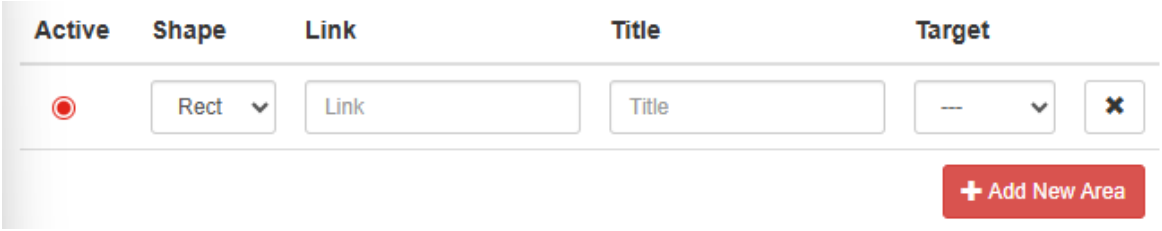

Ilustración 2. Barra de herramientas de image-map.

- Primero, verificamos que este activo.
- Elegimos el **shape** o la forma de la zona, región u figura de la cual buscamos obtener las coordenadas. Dependiendo la forma, tenemos 4 opciones:
    - **Rect:** Para una forma rectangular o similar a un rectángulo, como casas, edificios, cofres de tesoro, etc.
    - **Circle:** Para formas circulares u ovaladas como rostros, llantas, soles, monedas, etc.
    - **Poly:** Para figuras extrañas como caminos, partes de un mapa, animales; en otras palabras, figuras que están constituidas por varios pares de puntos.
    - **----** o **Default:** Este último valor es para marcar las zonas restantes del mapa, que ya no pertenecen a ninguna zona que queremos incluir como relevante.

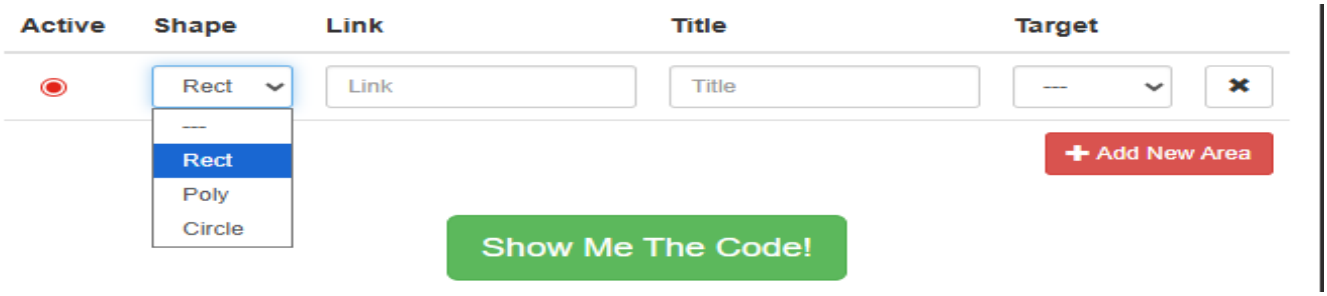

Ilustración 3. Etiqueta para seleccionar shape o forma de la cual se obtendrán las coordenadas.

- Una vez seleccionado el shape, los valores de **link, title** y **target** son opcionales para la obtención general del link, pero en este caso lo más relevante son las coordenadas.
- Ahora dependiendo del shape seleccionado, procedemos a marcar la zona uniendo las coordenadas lo mejor posible para no dejar nada fuera de ella o algo dentro que no pertenezca a esa zona.

Ilustración 4. Ejemplo de cómo se marcan la zona elegida uniendo cada coordenada.

- Luego, el sito nos genera las coordenadas como números sueltos, las ordenamos en pares (X, Y). (**Véase el anexo B)**

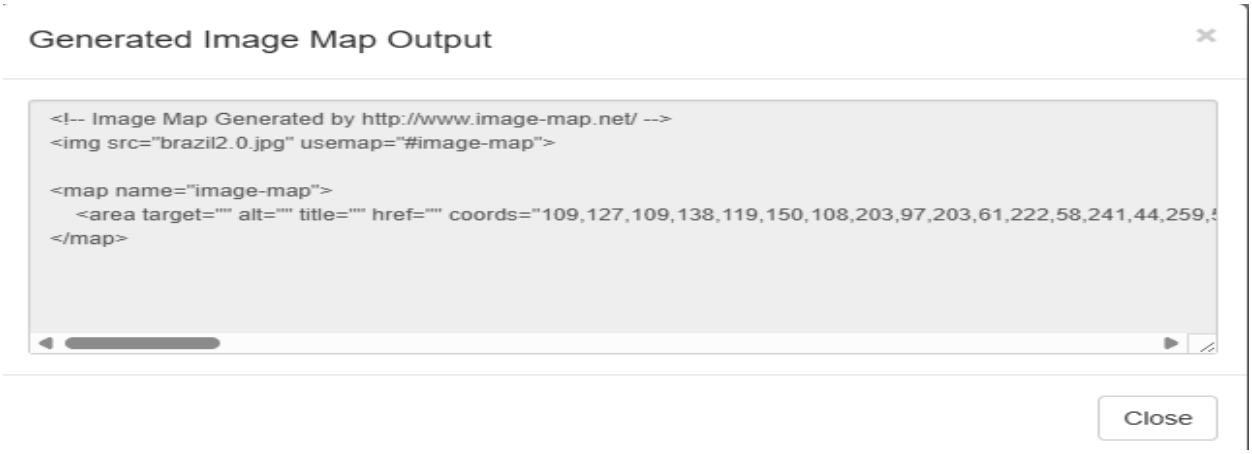

Ilustración 5. Ventana emergente que muestra target, alt, href y coords de la zona definida.

- Luego de haber marcado la primera zona, hacemos lo mismo con las siguientes zonas, regiones o figuras hasta tener todas generadas con sus pares de puntos ya ordenados.
- Procedemos a la obtención de mínimos y máximos para la obtención de los rangos para generar las condiciones. **(Véase el anexo C y D)**
- Ahora, cuidado, ya que a simple vista todo pareciera estar bien, pero hay zonas que se traslapan, quedan unas encima de otras, lo que generaría que al hacer clic en un punto que pertenezca a dos zonas, el servidor evaluaría la primera condición y si esa se cumple, esa se ejecuta cuando podríamos esperar que no fuese ese la evaluada.
- Para evitar esto, nos auxiliamos de graficas para evaluar las zonas que son victimas de esto. **(Véase el anexo E)**
- Todas estas condiciones y excepciones son analizadas y ejecutadas por un lenguaje que se ejecuta en el servidor, en este caso PHP. **(Véase el anexo F)**

Y así es como el atributo ISMAP era usado en web y páginas antiguas.

ANEXO B: ORDENACIÓN DE COORDENADAS EN PARES.

REGION NORTE

(415,38) (393,72) (378,72) (371,76) (356,70) (343,70) (341,78) (321,76) (311,83) (302,84) (286,91) (277,72) (275,51) (281,45) (277,36) (269,33) (275,25) (263,22) (260,33) (251,39) (236,42) (226,48) (220,45) (208,45) (197,40) (207,54) (208,61) (209,68) (219,68) (210,77) (202,86) (187,94) (181,98) (180,91) (172,98) (162,90) (156,80) (154,73) (147,75) (139,80) (114,80) (112,89) (121,91) (123,97) (110,100) (108,112) (120,126) (111,181) (98,178) (81,182) (65,194) (59,208) (59,217) (51,220) (49,229) (51,239) (56,250) (63,256) (59,262) (69,263) (74,271) (89,272) (99,266) (99,281) (104,291) (110,288) (133,290) (147,283)

(157,277) (165,271) (185,267) (186,278) (188,291) (188,297) (196,307) (209,314) (221,317) (236,324) (252,311) (272,297) (266,283) (267,267) (244,269) (240,240) (287,240) (293,216) (312,245) (410,235) (407,247) (407,275) (433,290) (445,278) (462,279) (497,262) (485,241) (493,230) (469,224) (471,195) (461,189) (478,179) (490,161) (497,124) (483,118) (469,124) (463,124) (465,111) (451,114) (444,108) (436,103) (440,92) (440,79) (429,78)

REGION CENTRO-OESTE

(241,326) (254,313) (276,297) (277,288) (270,283) (273,269) (245,265) (241,241) (289,244) (294,223) (309,246) (405,238) (405,274) (418,289) (433,289) (445,280) (462,283) (472,274) (502,267) (517,276) (515,289) (504,298) (510,317) (508,361) (505,369) (498,368) (491,375) (489,385) (479,388) (477,393) (457,392) (447,397) (437,415) (426,429) (425,437) (415,448) (405,469) (386,489) (369,511) (362,509) (353,511) (349,497) (346,485) (338,481) (328,483) (309,477) (312,459) (307,448) (311,442) (305,439) (313,424) (315,412) (312,398) (302,395) (302,382) (304,374) (297,378) (274,378) (268,360) (274,356) (271,348) (270,337) (247,331)

REGION NORDESTE

(499,125) (494,148) (488,167) (474,182) (464,186) (469,191) (472,218) (491,228) (494,238) (487,240) (499,263) (515,271) (519,279) (506,298) (511,317) (510,333) (508,361) (530,351) (533,365) (540,370) (552,369) (567,366) (578,375) (588,366) (597,367) (594,379) (595,391) (610,405) (617,402) (621,370) (617,351) (620,334) (618,327) (628,323) (653,284) (662,278) (668,269) (685,248) (689,225) (685,213) (679,194) (659,192) (647,187) (637,179) (602,153) (578,156) (566,152) (547,147) (538,154) (529,153) (532,144) (522,127) (517,134)

REGION SUDESTE

(608,411) (593,401) (592,384) (592,370) (580,380) (566,371) (560,375) (549,374) (540,374) (530,367) (527,356) (520,360) (513,365) (506,372) (499,372) (492,381) (480,389) (477,396) (457,394) (447,398) (439,413) (429,426) (427,437) (418,447) (412,459) (405,470) (388,487) (409,487) (419,489) (440,497) (442,516) (455,519) (458,529) (467,538) (489,520) (500,513) (513,511) (522,501) (535,498) (570,494) (575,482) (586,478) (585,466) (590,459) (595,455) (607,430)

REGION SUL

(371,513) (386,491) (413,491) (439,499) (439,518) (452,520) (464,537) (462,551) (460,563) (465,574) (458,598) (443,611) (436,625) (425,645) (404,660) (398,675) (385,695) (380,682) (389,678) (380,670) (378,662) (368,658) (363,650) (353,646) (344,639) (327,625) (316,628) (321,621) (332,608) (342,595) (358,581) (377,569) (379,542) (365,542)

ANEXO C: OBTENCIÓN DE MINIMOS Y MAXIMOS PARA PUNTOS X e Y.

Para obtener los mínimos y máximos de cada región, separamos los puntos **x** de los **y.**

El **mínimo** de un conjunto de puntos corresponde al menor valor de todos los números y se representa por $\mathbf{X_{mín}}$ y $\mathbf{Y_{mín.}}$

El **máximo** corresponde al mayor valor de ese conjunto de números y se representa por $\mathbf{X_{máx}}$ y $\mathbf{Y_{máx.}}$

Color **verde** para mínimos.

Color **rojo** para máximos.

REGIÓN NORTE

X= 415, 393, 378, 371, 356, 343, 341, 321, 311, 302, 286, 277, 275, 281, 277, 269, 263, 260, 251, 236, 226, 220, 208, 197, 207, 208, 209, 219, 210, 202, 187, 181, 180, 172, 162, 156, 154, 147, 139, 114, 112, 121, 123, 110, 108, 120, 111, 98, 81, 65, 59, 51, 49, 56, 63, 69, 74, 89, 99, 104, 133, 147, 157, 165, 185, 186, 188, 196, 221, 236, 252, 272, 266, 267, 244, 240, 287, 293, 312, 410, 407, 433, 445, 462, 497, 485, 493, 469, 471, 461, 478, 490, 497, 483, 469, 463, 465, 451, 444, 436, 440, 429

X_{min}= 49 **X_{max}=** 497

Y= 38, 72, 76, 70, 78, 76, 83, 84, 91, 51, 45, 36, 33, 25, 22, 39, 42, 48, 40, 54, 61, 68, 77, 86, 94, 98, 91, 90, 80, 73, 75, 89, 97, 100, 112, 126, 181, 178, 182, 194, 208, 217, 220, 229, 239, 250, 256, 262, 263, 271, 272, 266, 281, 291, 288, 290, 283, 277, 271, 267, 278, 291, 297, 307, 314, 317, 324, 311, 283, 267, 269, 240, 216, 245, 235, 247, 275, 275, 290, 279, 262, 241, 230, 224, 195, 189, 179, 161, 124, 118, 111, 114, 108, 103, 92, 79

Ymin= 22 **Ymax=** 324

X= 241, 254, 276, 277, 270, 273, 245, 241, 289, 294, 309, 405, 418, 433, 445, 462, 472, 502, 517, 515, 504, 510, 508, 505, 498, 491, 489, 479, 477, 477, 457, 437, 426, 425, 415, 405, 386, 369, 362, 353, 349, 346, 338, 328, 309, 312, 307, 311, 305, 313, 315, 312, 302, 304, 297, 274, 268, 271, 270, 247

Xmin= 241 **Xmax=** 517

Y= 326, 313, 297, 288, 283, 269, 265, 241, 244, 223, 246, 238, 274, 289, 280, 283, 274, 267, 276, 289, 298, 317, 361, 369, 368, 375, 385, 388, 393, 392, 397, 415, 429, 437, 448,

469, 489, 511, 509, 497, 485, 481, 483, 477, 459, 442, 439, 424, 412, 398, 395, 382, 374, 378, 360, 356, 348, 337, 331

Ymin= 223 **Ymax= 511**

X= 499, 494, 488, 474, 464, 469, 472, 491, 494, 487, 499, 515, 519, 506, 511, 510, 508, 530, 533, 540, 552, 567, 578, 588, 597, 594, 595, 610, 617, 621, 621, 620, 618, 628, 653, 662, 668, 685, 689, 679, 659, 647, 637, 602, 578, 566, 547, 538, 529, 532, 522, 517

Xmin= 464 **Xmax= 689**

Y= 125, 148, 167, 182, 186, 191, 218, 228, 238, 240, 263, 271, 279, 298, 317, 333, 361, 351, 365, 370, 369, 366, 375, 367, 379, 391, 405, 402, 370, 334, 327, 323, 284, 278, 269, 248, 225, 213, 194, 192, 187, 179, 153, 156, 152, 147, 127, 134

Ymin= 125 **Ymax= 405**

X= 608, 593, 592, 580, 566, 560, 549, 540, 530, 527, 520, 513, 506, 499, 492, 480, 477, 457, 447, 439, 429, 427, 418, 412, 405, 388, 409, 419, 440, 442, 455, 458, 467, 489, 500, 513, 522, 535, 570, 575, 586, 585, 590, 595, 607

Xmin= 388 **Xmax= 608**

Y= 411, 401, 348, 370, 380, 371, 375, 374, 367, 356, 360, 365, 372, 381, 389, 396, 394, 398, 413, 426, 437, 447, 459, 470, 487, 489, 497, 516, 519, 529, 538, 520, 513, 511, 501, 498, 494, 482, 478, 466, 459, 455, 430

Ymin= 348 **Ymax= 538**

X= 371, 386, 413, 439, 452, 464, 462, 460, 465, 458, 443, 436, 425, 404, 398, 385, 380, 389, 380, 378, 368, 363, 353, 344, 327, 316, 321, 332, 342, 358, 377, 379, 365

Xmin= 316 **Xmax= 465**

Y= 513, 491, 499, 518, 520, 537, 551, 563, 574, 598, 611, 625, 645, 660, 675, 695, 682, 678, 670, 662, 658, 650, 646, 639, 625, 628, 621, 608, 595, 581, 569, 542

Ymin= 491 **Ymax= 695**

ANEXO D: FORMULACIÓN DE RANGOS Y CONDICIONES

Una vez obtenidos los valores mínimos y máximos de cada región, procedemos a formular las condiciones para cada rango, de una forma sencilla y simple.

El menor valor de ese rango será el inicio de la condición para la letra x por ejemplo y el máximo valor será el final de ese rango para esa letra, ambos valores van incluidos, es decir se toman con el igual.

Para la formulación de los rangos, lo hacemos con la notación matemática:

$$\boldsymbol{X_{min} \leq x \leq X_{max}}$$

En PHP, y otros lenguajes de programación como CSS, JavaScript no leen notaciones matemáticas como humanos, si no que evalúan una condición a la vez y devuelven true o false, por lo tanto, en lugar de una frase matemática se escribe la lógica implícita de la forma:

$$\boldsymbol{X_{min} \geq x \leq X_{max}}$$

El lenguaje lo interpreta como: $x_{min} \geq x \rightarrow true\ or\ false$

$$x_{max} \geq x \rightarrow true\ or\ false$$

$$\&\&\ solo\ si\ ambos\ son\ \boldsymbol{true}$$

Esto quiere decir que para el servidor devuelva true, ambas evaluaciones deben devolver un true, si una arroja false, el servidor no entra al if y pasa a un elseif hasta encontrar uno que arroje un **true global.**

REGIÓN NORTE:

Para x:

$\mathbf{X_{min}}$= 49 $\mathbf{X_{max}}$= 497

Rango:

$$x \leq 49 \wedge x \leq 497$$

Condición:

$$49 \geq x \leq 497$$

Para y:

$\mathbf{Y_{min}}$= 22 $\mathbf{Y_{max}}$= 324

Rango:

$$y \leq 22 \wedge y \leq 324$$

Condición:

$$22 \geq y \leq 324$$

REGIÓN CENTRO-OESTE:

Para x:

X_{min}= 241 **X_{max}**= 517

Rango:

$$x \leq 241 \wedge x \leq 517$$

Condición:

$$241 \geq x \leq 517$$

Para Y:

Y_{min}= 223 **Y_{max}**= 511

Rango:

$$y \leq 223 \wedge y \leq 511$$

Condición:

$$223 \geq y \leq 511$$

REGIÓN NORDESTE:

Para X:

X_{min}= 464 **X_{max}**= 689

Rango:

$$x \leq 464 \wedge x \leq 689$$

Condición:

$$464 \geq x \leq 689$$

Para Y:

Y_{min}= 125 **Y_{max}**= 405

Rango:

$$y \leq 125 \wedge y \leq 405$$

Condición:

$$125 \geq y \leq 405$$

REGIÓN SUDESTE:

Para X:

X_{min}= 388 **X_{max}**= 608

Rango:

$$x \leq 388 \wedge x \leq 608$$

Condición:

$$388 \geq x \leq 608$$

Para Y:

$\mathbf{Y_{min}}$= 348 $\mathbf{Y_{max}}$= 538

Rango:

$$y \leq 348 \wedge y \leq 538$$

Condición:

$$348 \geq y \leq 538$$

REGIÓN SUR:

$\mathbf{X_{min}}$= 316 $\mathbf{X_{max}}$= 465

Rango:

$$x \leq 316 \wedge x \leq 465$$

Condición:

$$316 \geq x \leq 465$$

Para Y:

$\mathbf{Y_{min}}$= 491 $\mathbf{Y_{max}}$= 695

Rango:

$$y \leq 491 \wedge y \leq 695$$

Condición:

$$491 \geq y \leq 695$$

ANEXO E: MANEJO DE TRASLAPE ENTRE REGIONES

Una vez ya tengamos las condiciones formuladas, antes de ser declaradas en el editor del lenguaje que se ejecutara en el servidor, es necesario verificar que no haya zonas traslapadas o encimadas (una encima de la otra).

¿Qué es un traslape?

Un **traslape (Overlap)** se da cuando un mismo par de puntos (x, y) pertenecen a dos o más zonas o regiones lógicas definidas en el servidor. Esto se da porque principalmente al usar *ismap,* el servidor suele simplificar zonas complejas (en este caso regiones, en otros ejemplos piezas de motor, partes del cuerpo, etc.) mediante rectángulos envolventes y no como la forma real que tienen.

Entonces aquí sucede el principal problema no solo con ismap si no en desarrollo de aplicaciones o videojuegos.

El servidor no tiene una visión sobre ellas o en palabras de desarrollo, no tiene una "visión espacial"; solo lee líneas de código de arriba hacia abajo sin discriminar la forma que el rango podría tener y el manejo se reduce a una jerarquía de evaluación que en palabras simples es: **la primera condición que se cumple, gana,** y esta es la condición que ejecutará sin evaluar más y el flujo de evaluación se detiene y PHP no tiene posibilidad de preguntarle al usuario a cual de las dos zonas que pertenece el pixel quería ir.

En esta ocasión hemos manejado los traslapes con una forma sencilla pero efectiva, ya que nuestro trabajo no consiste en buscar la forma más "moderna" de como resolver estos problemas. El método podría ser denominado **"Recorte de dominios".**

Antes de pasar a ella, veamos de forma grafica los traslapes que sufren las regiones.

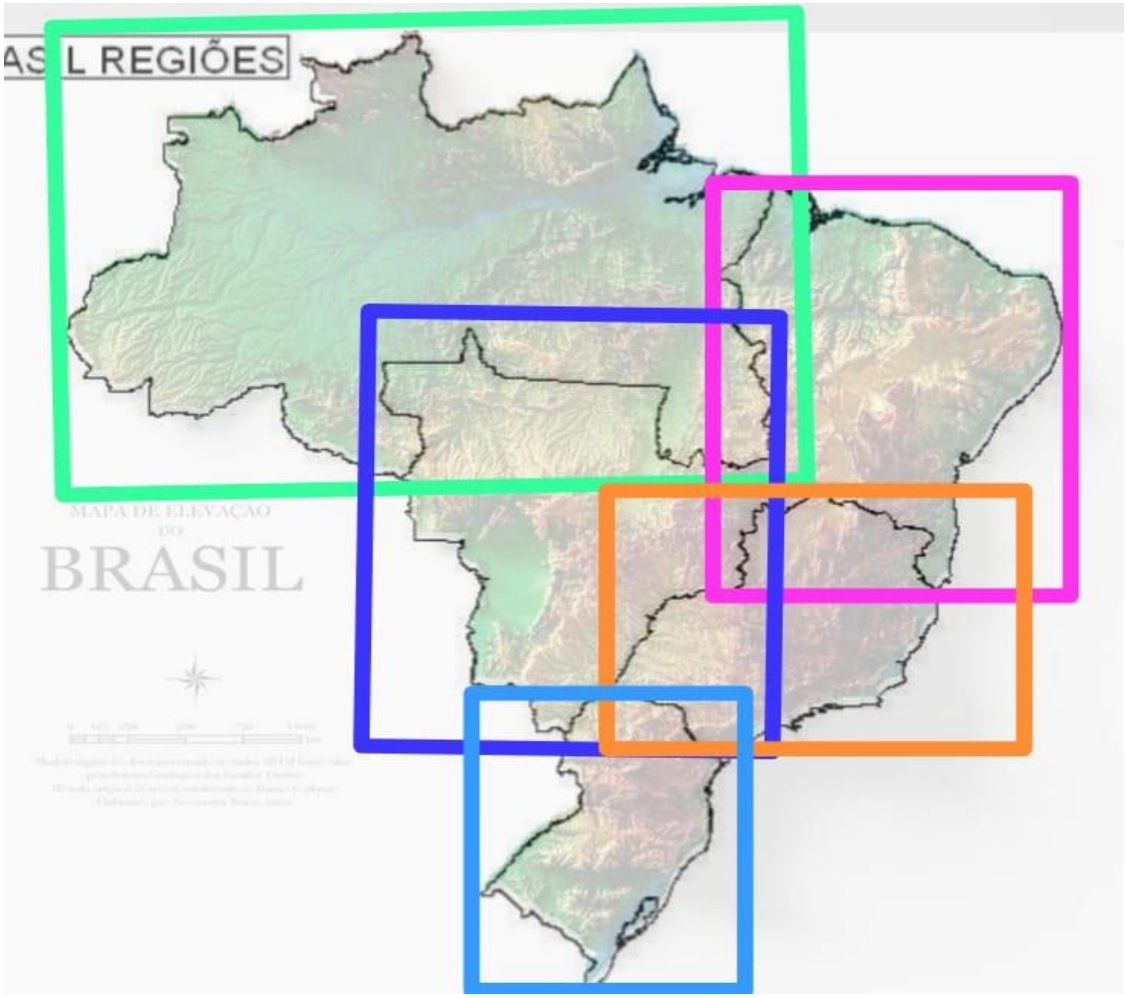

Ilustración 6. Rectángulos con los que la lógica de ismap trabaja para definir regiones, formando rectángulos que se posicionan unos encima de otros.

Esta es la forma en la que ISMAP suele simplificar zonas complejas como se menciono anteriormente, todas las zonas están encimadas una sobre la otra.

Para saber en que puntos inicia y termina exactamente el traslape lo podemos hacer de forma manual mediante una gráfica, como la del ejemplo:

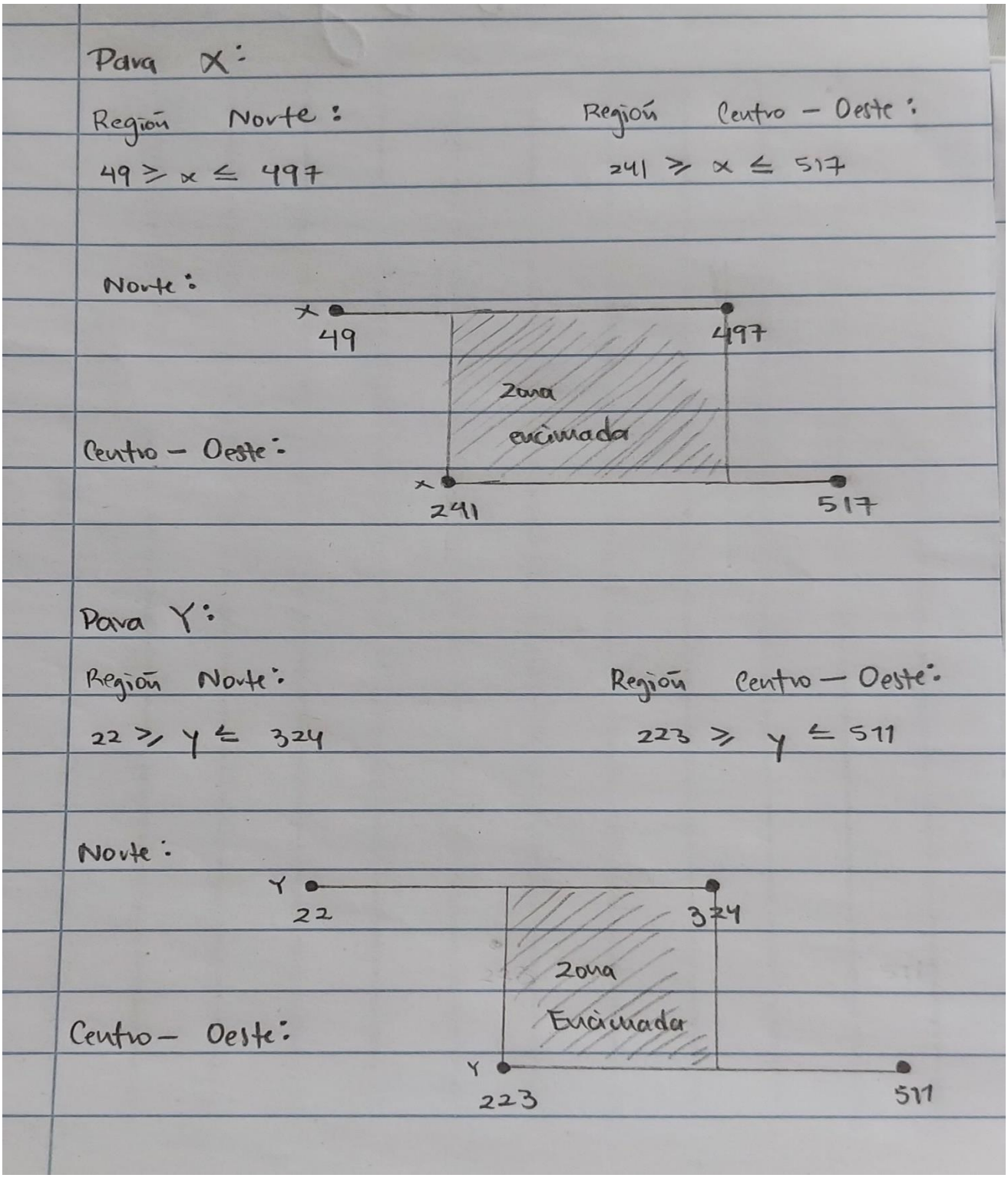

Ilustración 7. Gráfica manual de los rangos donde se dan las zonas traslapadas o encimadas.

Problema: Debido a la naturaleza rectangular de las regiones en el servidor, observamos que en *x* el rango [241, 497] es compartido por ambas regiones, esto genera una zona "encimada" y en *y* en el rango [223, 324].

Análisis visual: Se realizo un mapeo manual de limites (ver imagen adjunta arriba) para poder identificar el área de colisión identificándolo con la zona sombreada en la imagen.

Estrategia de solución: Se optó por una **jerarquía de importancia**, decidiendo que la región Centro-Oeste tuviera prioridad sobre la región Norte en el área de conflicto, lo cual en otras palabras sería decirle al servidor: *"Usuario hizo clic en el pixel (250, 300), la primera condición dice:* ***"Cumple con Norte, PERO también cumple con la zona prohibida (la de Centro-Oeste), así que ignora el Norte y pasa a Centro-Oeste".***

Región en conflicto	Coordenadas compartidas	Región prioritaria
Norte vs. Centro-Oeste	$x\ [241, 517], y\ [223, 324]$	Centro-Oeste
Centro-Oeste vs. Nordeste	$x\ [464, 517], y\ [223, 405]$	Nordeste
Nordeste vs. Sudeste	$x\ [464, 608], y\ [348, 405]$	Sudeste
Sudeste vs. Sur	$x\ [388, 465], y\ [491, 538]$	Sur

Tabla 1. Jerarquía de solución para zonas encimadas, donde se muestra cual región tuvo prioridad sobre otra cediéndole las coordenadas en guerra.

Implementación: Agregamos una nueva sentencia condicional para la región de menor prioridad (Norte) integrando un operador de negación que excluye las coordenadas de la región dominante.

ANEXO F: PROCESAMIENTO DE COORDENADAS EN EL SERVIDOR (PHP)

Para que el servidor pueda no solo recibir si no también interpretar el clic del usuario sobre el mapa de imagen, se implemento un script en PHP encargado de capturar, procesar y segmentar los datos espaciales. Este código siempre escrito en un archivo en el editor de texto, este caso VS CODE con la extensión **.php** para indicarle que no es código HTML.

1. **Etiquetas de inicio y cierre para escribir código PHP:** Para iniciar a escribir código PHP, se usa la sintaxis:

```
<?php
// todo el código php
?>
```

2. **Captura de datos:** $\$\boldsymbol{query} = \$\_\boldsymbol{SERVER}[\text{QUERY\_STRING}]$.
   Cuando se usa *ismap,* el navegador añade las coordenadas al final de la URL en el formato $?x, y$ (por ejemplo: mapa_brazil.php?50, 200).
   Aquí entra en juego la declaración anterior:
   - En este caso, $\$\boldsymbol{query}$ almacena el valor crudo $50, 200$.
   - La variable superglobal $= \$\_\boldsymbol{SERVER}[\text{QUERY\_STRING}]$ se utilizó en este caso para recuperar esa cadena de texto después del signo de interrogación.
3. **Segmentación de datos:** $\boldsymbol{list}(\ )\boldsymbol{y\ explode}(\ )$

Una vez obtenida la cadena, es necesario separara los valores numéricos para que el servidor los pueda evaluar individualmente.

- ***list*** **(**$***x***, $***y***)**:** Se utilizo esta construcción del lenguaje para asignar de forma elegante y directa los valores del array a dos variables independientes.
- ***explode***(**",",** $***query***)**:** Esta función actúa como un "cuchillo", cortando la cadena de texto cada vez que se encuentra una coma. Devuelve un array con dos elementos: el primero es la coordenada x y el segundo la coordenada y.

4. **Estructura de decisión espacial (sentencias de control):**
   Para que el servidor elija el destino correcto, se utilizó una cadena de condiciones que funcionan como un filtro. El orden es vital porque el servidor se detiene en la primera "puerta" que encuentre abierta y "lo deje pasar".
   - **La apertura: *if* (La condición inicial):** Es el punto de entrada. Se usa para la primera región o para la verificación más importante.
     - **Significado:** "Si las coordenadas (x, y) caen dentro de este rango especifico…"
     - **En este proyecto:** Fue el primer intento de evaluar una región antes de saber que habría conflictos.
   - **El cuerpo: *elseif* ("El "Si no, intenta esto…"):** Se utiliza para todas las regiones intermedias (de la segunda hasta la antepenúltima).
     - **Significado:** Es una contracción de ***else if*****.** Significa "Si la condición anterior no se cumplió, **pero** esta nueva condición si se cumple, entonces haz esto".
     - **En este proyecto:** Aquí vive la **lógica de excepciones.** Cada ***elseif*** revisa una región del mapa de Brasil y, gracias al operador **!,** verifica que no estemos en una zona de guerra con otra región.
   - **El cierre: *else* (La red de seguridad):** Es la última sentencia y no lleva ninguna condición entre paréntesis.
     - **Significado:** "Si ninguna de las condiciones anteriores fue verdadera, por descarte, haz esto".
     - **En este proyecto:** Funciona como un "atrapa-todo". Si el usuario hizo clic fuera del mapa, en el espacio en blanco o fuera de las fronteras de Brasil, el ***else*** lo captura y lo envía a una página sin información. Sin esto, el servidor podría intentar procesar un clic vacío y generar un error.

La arquitectura del script sigue un flujo de **descarte sucesivo.** Se inicia con un ***if*****,** se segmentan las regiones mediante múltiples ***elseif*** (donde se resolvieron los traslapes gráficos), y se finaliza con un ***else*** que actúa como un **manejador de excepciones predeterminados** para clics fuera del área de interés.

Ejemplo de flujo de datos:

1. Usuario hace clic en pixel (x=455, y=600).
2. Navegador envía "procesar_region.php?455,600"
3. PHP captura la coordenada con $query y se queda como "455, 600".

4. PHP separa ambos números con list ($x, $y) y convierte a $x en 455 y a $y en 600.
5. El script comienza a evaluar las condiciones verificando a que zona pertenecen aplicando la regla de la soberanía de pixeles.
6. Luego de la evaluación, PHP decide a donde redirigir al usuario.

El código final con lógica de excepciones y jerarquía de condiciones queda como:

```php
<?php
$query=$_SERVER["QUERY_STRING"];
list($x, $y)=explode(",",$query);

//Lógica de regiones donde nosotros decidimos el rango, dividimos el mapa en las regiones requeridas.
//BRASIL
if(
    $x >= 21 && $x <= 201 &&
    $y >= 345 && $y <= 438
){
    header ("Location: brazil.html");
    exit;
}

//REGION NORTE
elseif(
    $x >= 49 && $x <= 497 &&
    $y >= 22 && $y <= 324 &&
    !($x >= 241 && $x <= 497 &&
    $y >= 223 && $y <= 324)
){
header("Location: norte.html");
exit
}
//REGION CENTRO-OESTE
elseif(
    $x >= 241 && $x <= 517 &&
    $y >= 223 && $y <= 511 &&
    !($x >= 464 && $x <= 517 &&
    $y >= 223 && $y <= 405)
){
    header("Location: centro-oeste.html");
    exit;
}
//REGION NORDESTE
elseif(
    $x >= 464 && $x <= 689 &&
    $y >= 125 && $y <= 405 &&
    !($x >= 464 && $x <= 608 &&
$y >= 348 && $y <= 405)
){
```

```php
    header("Location: nordeste.html");
    exit;
}
//REGION SUDESTE
elseif(
    $x >= 388 && $x <= 608 &&
    $y >= 348 && $y <= 538 &&
    !($x>= 388 && $x <= 465 &&
    $y >= 491 && $y <=538)
){
    header("Location: sudeste.html");
    exit;
}

//REGION SUL
elseif(
    $x >= 316 && $x <= 465 &&
    $y >= 491 && $y <= 695
){
    header("Location: sur.html");
    exit;
}
//Si toca fuera del mapa
else{
    header("Location: sin_informacion.html");
exit;
}
?>
```

CONCLUSIÓN DE LOS ANEXOS: "LECCIONES DE UN ATRIBUTO OBSOLETO"

El estudio y la implementación del atributo *ismap* permitieron comprender conceptos fundamentales que trascienden el propio HTML como base de una página web.

1. **Interpretación de datos crudos:** El uso de $\$\_\boldsymbol{SERVER} = [\text{"QUERY\_STRING"}]$ y $\boldsymbol{explode}$ demostró como el servidor debe "traducir" acciones físicas del usuario en variables numéricas manejables (x, y).
2. **Geometría lógica:** La creación de gráficas de límites reveló la discrepancia que puede haber o que realmente hay entre la visualización de sus limites reales y la rigidez matemática con la que el servidor trabajaba antiguamente.
3. **Jerarquía de control:** La estructura $\boldsymbol{if, elseif, else}$ no solo nos ayudo a dirigir el gráfico, si no a establecer un sistema de prioridades y excepciones de forma manual que resolvió un problema muy común.
4. **Evolución hacia la accesibilidad:** Finalmente, porque *ismap* requiere tanto esfuerzo manual deja claro porque las tecnologías modernas lo han dejado en el olvido y ha sido reemplazado por tecnologías modernas que trasladan toda lo que el servidor realizaba al navegador, haciendo que los mapas de imágenes sean escalables, interactivos y sobre todo accesible.

Y como conclusión final en palabras coloquiales de esta servidora: es que lo que inicio como una simple etiqueta obsoleta me llevo a comprender y adquirir una base sólida que me ayudo y seguirá ayudando para seguir aprendiendo mas lenguajes de programación porque la web nunca deja de evolucionar y como desarrolladores y programadores nunca está de más conocer las tecnologías que antecedieron a las que conocemos hoy en día y que seguirán dando paso a más nuevas, ánimos para todos, que el desarrollo es lo más lindo y somos afortunados de conocer lo que hay detrás de lo que a diario vemos!

Con amor, su servidora, SDE JESSICA MARTÍNEZ.