# MERCADO LIBRE

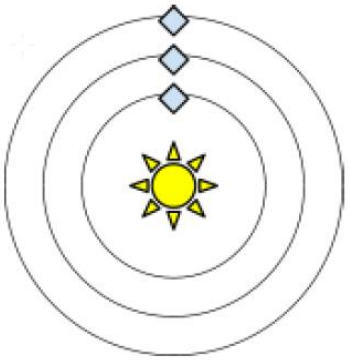
# RESOLUCION AL PROBLEMA DE LOS PLANETAS.

**JUAN MANUEL GARCIA CAÑETE**

# Tabla de contenidos.

# 1 Problema a resolver.

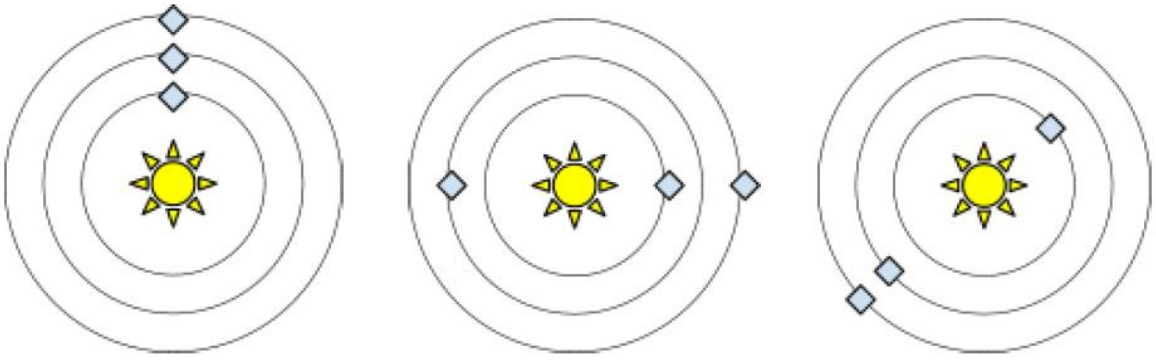
En una galaxia lejana, existen tres civilizaciones. Vulcanos, Ferengis y Betasoides. Cada civilización vive en paz en su respectivo planeta.
Dominan la predicción del clima mediante un complejo sistema informático.



**Premisas:**

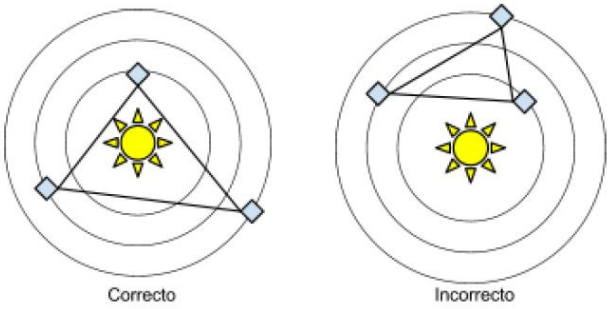
- El planeta Ferengi se desplaza con una velocidad angular de 1 grados/día en sentido horario. Su distancia con respecto al sol es de 500Km.
- El planeta Betasoide se desplaza con una velocidad angular de 3 grados/día en sentido horario. Su distancia con respecto al sol es de 2000Km.
- El planeta Vulcano se desplaza con una velocidad angular de 5 grados/día en sentido antihorario, su distancia con respecto al sol es de 1000Km.
- Todas las órbitas son circulares.

Cuando los tres planetas están alineados entre sí y a su vez alineados con respecto al sol, el sistema solar experimenta un período de sequía.

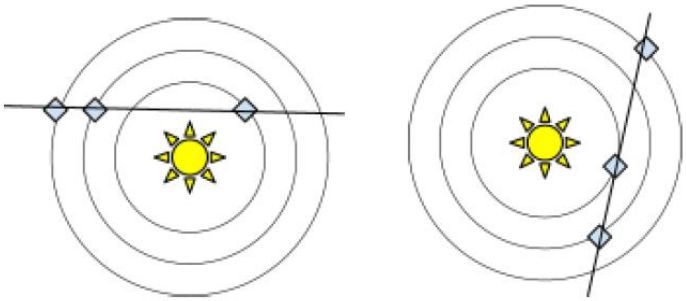


Cuando los tres planetas no están alineados, forman entre sí un triángulo. Es sabido que en el

momento en el que el sol se encuentra dentro del triángulo, el sistema solar experimenta un período de lluvia, teniendo éste, un pico de intensidad cuando el perímetro del triángulo está en su máximo.




Las condiciones óptimas de presión y temperatura se dan cuando los tres planetas están alineados entre sí pero no están alineados con el sol.



Realizar un programa informático para poder predecir en los próximos 10 años:

1. ¿Cuántos períodos de sequía habrá?
2. ¿Cuántos períodos de lluvia habrá y qué día será el pico máximo de lluvia?
3. ¿Cuántos períodos de condiciones óptimas de presión y temperatura habrá?

***Bonus:***

Para poder utilizar el sistema como un servicio a las otras civilizaciones, los Vulcanos requieren tener una base de datos con las condiciones meteorológicas de todos los días y brindar una API REST de consulta sobre las condiciones de un día en particular.

1) Generar un modelo de datos con las condiciones de todos los días hasta 10 años en adelante utilizando un job para calcularlas.

2) Generar una API REST la cual devuelve en formato JSON la condición climática del día consultado.

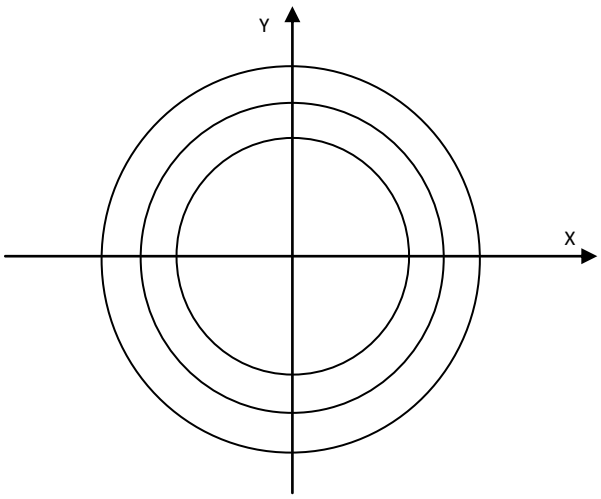
3) Hostear el modelo de datos y la API REST en un cloud computing libre (como APP Engine o Cloudfoudry) y enviar la URL para consulta:

Ej: GET → http://……/clima?dia=566 → Respuesta: {“dia”:566, “clima”:”lluvia”}

# 2 Planeamiento.

## 2.1 Definición.

Lo primero que vamos a observar es que las órbitas donde se encuentran los planetas son circulares y que la velocidad angular de cada uno es constante, por lo cual describe ángulos iguales en tiempos iguales. Este fenómeno en Física se lo denomina como movimiento circular uniforme (MCU).
Este concepto nos permite ubicar a las orbitas sobre los ejes cartesianos para poder resolverlo, utilizando la abscisa “X” y la ordenada “Y”, cuyo origen es el sol.




## 2.2 Abordaje.

Luego de tener definido el tipo de problema que vamos a tratar, podemos analizar la mejor forma de abordarlo e ir realizando una segmentación del mismo para finalmente resolverlo de forma total.

### 2.2.1 Cantidad de dias que tardan en dar una vuelta completa.

En principio sabemos que existen tres planetas que están orbitando el sol y nos proporcionan de cada uno su velocidad angular, por lo cual podemos calcular la cantidad de días que tardan en dar una vuelta completa a

su órbita. Una vez que tengamos esa información conocemos que planeta demora más tiempo y sobre ese mismo calculamos los diez años que solicita el enunciado.

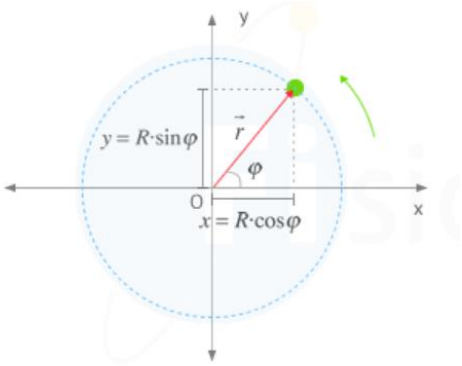
Total de grados de una circunsferencia: 360°
Planeta Ferengi: velocidad angular 1°/día
Planeta Betasoide: velocidad angular 3°/día
Planeta Vulcano: velocidad angular 5°/día
Utilizando la regla de tres simple resolvemos la siguiente incógnita
Velocidad angular.....días
360°.....x

Planeta Ferengi demora 360 días en dar la vuelta al sol
Planeta Betasoide demora 120 días en dar la vuelta al sol
Planeta Vulcano demora 72 días en dar la vuelta al sol

Debemos tomar como referencia al planeta Ferengi, ya que, si elegimos otro planeta, los Ferengi no podrán calcular su clima a diez años y no podriamos resolver la problemática planteada.

### 2.2.2 Direcciónes de los planetas.

El problema no solo esta dando la velocidad angular de cada planeta, sino que también esta diciendo que dos planetas van en la misma dirección, mientras que el otro va en sentido contrario. Es decir, que los planetas que vayan en sentido horario tendrán su velocidad angular de forma negativa, mientras que los que tengan sentido contrario al reloj poseen la velocidad angular positiva




Por esta razón la velocidad angular la tenemos que tomar de la siguiente forma:

Planeta Ferengi: velocidad angular -1°/día

Planeta Betasoide: velocidad angular -3°/día
Planeta Vulcano: velocidad angular 5°/día

### 2.2.3 Conversión de velocidades y de sistema.

La velocidad angular para fines prácticos en el MCU se utiliza en la forma de medicion radianes, por tal motivo debemos convertirla con la siguiente fórmula matemática:

Velocidad angular X pi / 180 = n° radianes

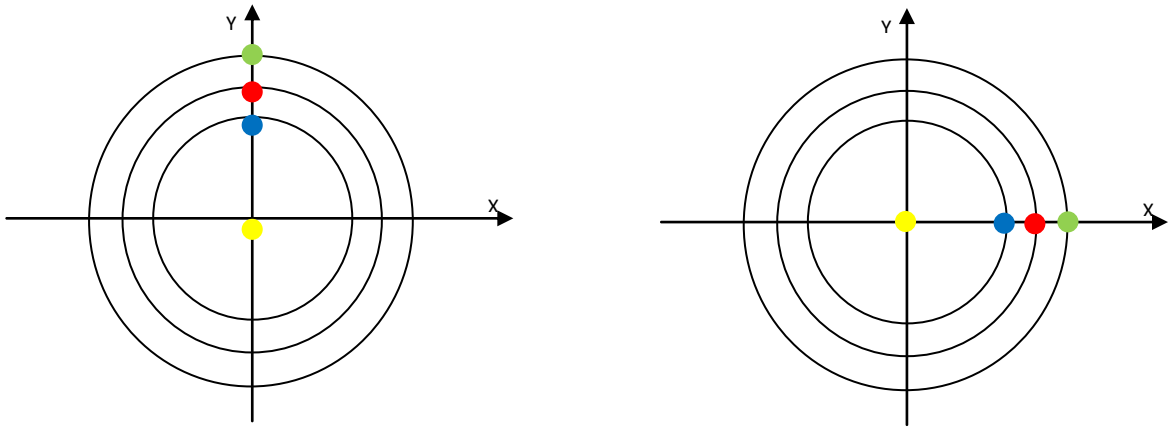
A su vez, de la manera que estan dados los datos debemos pasar de la ecuación polar a la ecuación cartesiana. Esto lo resolvemos con la siguiente fórmula:

EjeX = radio x cos (velocidad angular)

EjeY = radio x sen (velocidad angular)

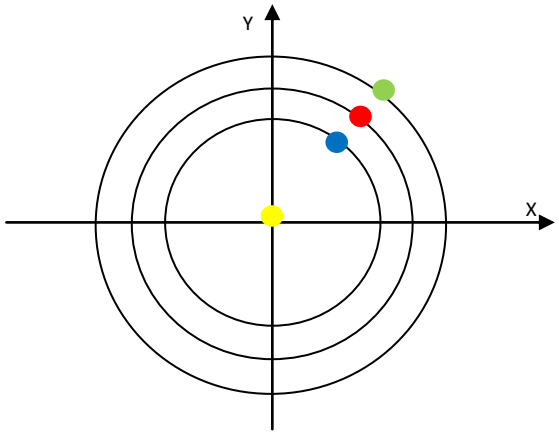
### 2.2.4 Alineación de los planetas.

Para poder descubrir cuando están alineados entre sí vamos a pensar de la siguiente forma:




Cuando la ordenada de todos esten alineados entre sí de manera vertical y cuando su abcsisa seán todas iguales estarán alineados de forma horizontal. A su vez en este caso si necesitamos comprobar si están alineados con el sol debemos pensar al sol como otro planeta cuyas coordenadas serán (0,0).

Para evaluar si los planetas están alineados de forma oblicua debemos analizarlo de la siguiente manera:



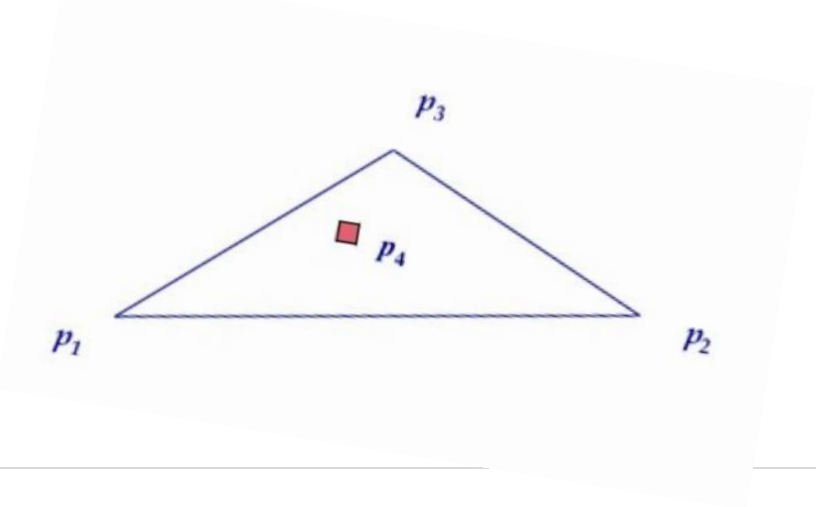

Los puntos A (x1, y1), B(x2, y2) y C(x3, y3) están alineados siempre que los vectores tengan la misma dirección. Esto ocurre cuando sus coordenadas son proporcionales. Nuevamente una vez verificado que los planetas se encuentran alineados, volvemos a validarlo con el sol.

Con la siguiente formula verificamos la direccion del vector:

$$\frac{x_2 - x_1}{x_3 - x_2} = \frac{y_2 - y_1}{y_3 - y_2}$$

### 2.2.5 Forma de triangulo con el sol en el medio

En el enunciado se detalla que cuando los planetas no se encuentran alineados entre sí forman un triangulo. Nos restaría saber cuando el sol se encuentra dentro del triangulo y cuando esta afuera.



Una de las alternativas podría ser calcular el área total del triangulo y luego evaluar si la suma de cada sub área esta dentro, para eso realizaremos el siguiente calculo:

AreaTotal (P1-P2-P3) = Area1 (P1-P3-P4) + Area2 (P3-P2-P4) + Area3 (P2-P1-P4)

Cada área del tringulo lo calculamos con la regla de Sarrus a traves del determinante que se encontrara con el valor absoluto.

**Área de un triángulo por determinantes**

$$A = \frac{1}{2} \cdot \begin{vmatrix} a_1 & a_2 & 1 \\ b_1 & b_2 & 1 \\ c_1 & c_2 & 1 \end{vmatrix}$$

A su vez para calcular el perímetro estamos utilizando la siguiente fórmula:

$$d(P_1, P_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Nos da la distancia entre dos puntos, hay que repetirlo por cada lados del triangulo, es decir unas tres veces para obtener el total del peremitro

# 3 Programa y Bonus

**JOB SQL**

```
Declare @dia int, @ferengiX float, @ferengiY float, @betasoideX float, @betasoideY float,
@vulcanoX float, @vulcanoY float, @solX float, @solY float, @AreaTotal float,
@AreaABD float,@AreaBCD float,@AreaACD float, @PerimTotal float,
@PerimAB float,@PerimBC float,@PerimCA float

set @dia = 0

WHILE @dia < 3600
BEGIN

	set @ferengiX = Round(500 * Cos((-1 * PI() / 180) * @dia),0)
	set @ferengiY = Round(500 * Sin((-1 * PI() / 180) * @dia),0)

	set @betasoideX = Round(2000 * Cos((-3 * PI() / 180) * @dia),0)
	set @betasoideY = Round(2000 * Sin((-3 * PI() / 180) * @dia),0)

	set @vulcanoX = Round(1000 * Cos((5 * PI() / 180) * @dia),0)
	set @vulcanoY = Round(1000 * Sin((5 * PI() / 180) * @dia),0)

	set @solX = Round(0 * Cos((0 * PI() / 180) * @dia),0)
	set @solY = Round(0 * Sin((0 * PI() / 180) * @dia),0)



	/*SABER SI ESTAN ALINEADOS*/
	BEGIN
	If((@ferengiX=@betasoideX) and (@ferengiX=@vulcanoX))
		BEGIN
			If (@ferengiX=@solX)
				insert into Clima values (@dia, 'Sequía',0 )
			Else
				insert into Clima values (@dia, 'óptimo',0 )
			END
	Else If((@ferengiY=@betasoideY) and (@ferengiY=@vulcanoY))
		BEGIN
			If (@ferengiY=@solY)
				insert into Clima values (@dia, 'Sequía',0 )
			Else
				insert into Clima values (@dia, 'óptimo',0 )
		END
	Else If(Round((@betasoideX - @ferengiX) * (@vulcanoY - @betasoideY),0) =
		Round((@vulcanoX - @betasoideX) * (@betasoideY - @ferengiY),0))

		BEGIN
			If ((@betasoideX - @ferengiX) * (@solY - @betasoideY) =
				(@solX - @betasoideX) * (@betasoideY - @ferengiY))
				insert into Clima values (@dia, 'Sequía',0)
			Else
```

```
                        insert into Clima values (@dia, 'óptimo',0)
                END
        Else
                BEGIN
                /*OBTENER TRIANGULO*/
                set @AreaTotal = ABS((@ferengiX * @betasoideY + @ferengiY * @vulcanoX +
                                @betasoideX * @vulcanoY - @betasoideY * @vulcanoX -
                                @ferengiY * @betasoideX - @ferengiX * @vulcanoY) /2)


                set @AreaABD = ABS((@ferengiX * @betasoideY + @ferengiY * @solX +
                                @betasoideX * @solY - @betasoideY * @solX -
                                @ferengiY * @betasoideX - @ferengiX * @solY) /2)


                set @AreaBCD = ABS((@solX * @betasoideY + @solY * @vulcanoX +
                                @betasoideX * @vulcanoY - @betasoideY * @vulcanoX -
                                @solY * @betasoideX - @solX * @vulcanoY) /2)

                set @AreaACD = ABS((@ferengiX * @solY + @ferengiY * @vulcanoX +
                                @solX * @vulcanoY - @solY * @vulcanoX -
                                @ferengiY * @solX - @ferengiX * @vulcanoY) /2)


                set @PerimAB = ABS(SQRT(POWER(@betasoideX - @ferengiX,2) +POWER(@betasoideY
- @ferengiY,2)));
                set @PerimBC = ABS(SQRT(POWER(@vulcanoX - @betasoideX,2) +POWER(@vulcanoY -
@betasoideY,2)));
                set @PerimCA = ABS(SQRT(POWER(@ferengiX - @vulcanoX,2) +POWER(@ferengiY -
@vulcanoY,2)));

                set @PerimTotal = ABS(@PerimAB+@PerimBC+@PerimCA)

                If(@AreaTotal = (@AreaABD + @AreaBCD + @AreaACD))
                        insert into Clima values (@dia, 'Lluvia',@PerimTotal )
                Else
                        insert into Clima values (@dia, 'no definido',0 )

                END
        END

        SET @dia = @dia + 1;
END
```

**API REST**

## RouteController

```
routes.MapRoute(
                name: "Clima",
                url: "Clima/{action}/{id}",
                defaults: new { controller = "Clima", action = "Clima", id =
UrlParameter.Optional }
            );

            routes.MapRoute(
                name: "Default",
                url: "{controller}/{action}/{id}",
                defaults: new { controller = "Home", action = "Index", id =
UrlParameter.Optional }
            );
```

## Modelo

```
public class Clima
    {
        public int dia { get; set; }
        public String clima { get; set; }
    }
```

## Controllers

*HomeController*

```
public class HomeController : Controller
    {
        // GET: Home
        public ActionResult Index()
        {
            return View();
        }
    }
```

*ClimaController*

```
public class ClimaController : Controller
    {
        // GET: Clima
        [HttpGet]
        public ActionResult Clima(int dia)
        {
            String obtenerdia = "";
            ClassSql ex = new ClassSql();
            obtenerdia = ex.diaparticular(dia);
            ViewBag.Tipo = obtenerdia;
            return View();
        }


        // GET: Periodos Sequía
        [HttpGet]
```

```
        public ActionResult Sequia()
        {
            String obtenerdia = "";
            ClassSql ex = new ClassSql();
            obtenerdia = ex.diaSequia();
            ViewBag.Sequia = obtenerdia;
            return View();
        }


        // GET: Periodos Lluvia
        [HttpGet]
        public ActionResult Lluvia()
        {
            String obtenerdia = "";
            String dias = "";
            ClassSql ex = new ClassSql();
            obtenerdia = ex.diaLluvia();
            ViewBag.Lluvia = obtenerdia;
            dias = ex.PicosLluvia();
            ViewBag.Dias = dias;
            return View();
        }

    }
```

**Views**

*Clima.cshtml*

```
@{
    ViewBag.Title = "Clima";
}

<h2>Clima</h2>

<p>@ViewBag.Tipo</p>
```

*Lluvia.cshtml*

```
@{
    ViewBag.Title = "Lluvia";
}

<h2>Lluvia</h2>
<p>@ViewBag.Lluvia</p>
<p>@ViewBag.Dias</p>
```

*Sequia.cshtml*

```
@{
    ViewBag.Title = "Sequía";
}

<h2>Sequia</h2>
<p>@ViewBag.Sequia</p>
```

**Clase**

```
public class ClassSql
    {
        private String con()
        {
            return " Server = tcp:sqlryj.database.windows.net,1433; Initial Catalog =
bdmeli; Persist Security Info = False;" +
                " User ID = {******}; Password ={*******}; MultipleActiveResultSets =
False; Encrypt = True; TrustServerCertificate = False; Connection Timeout = 30;";
        }

        public String diaparticular (int dia)
        {
            String obtenerdia = "";
            Models.Clima cl = new Models.Clima();

            SqlConnection cnn = new SqlConnection(con());
            cnn.Open();
            SqlCommand query = new SqlCommand("Select dia,clima from clima where dia=" +
dia + "", cnn);
            SqlDataReader dr;
            dr = query.ExecuteReader();

            if (dr.HasRows)
            {
                while (dr.Read())
                {
                    cl.dia = Convert.ToInt32(dr[0].ToString());
                    cl.clima = dr[1].ToString();
                }
                obtenerdia = JsonConvert.SerializeObject(cl);
            }
            else
            {
                obtenerdia = "No hay registro para esa consulta";
            }
            return obtenerdia;

        }

        public String diaSequia()
        {
            String diaSequia = "";

            SqlConnection cnn = new SqlConnection(con());
            cnn.Open();
            SqlCommand query = new SqlCommand("Select clima, count(*) from clima group by
clima having clima='Sequía'", cnn);
            SqlDataReader dr;
            dr = query.ExecuteReader();

            if (dr.HasRows)
            {
                while (dr.Read())
                {
```

```
                    diaSequia = "Total de días de sequía: " + dr[1].ToString();
                }
            }
            else
            {
                diaSequia = "No hay días de sequía";
            }
            return diaSequia;

        }


        public String diaLluvia()
        {
            String diaLluvia = "";

            SqlConnection cnn = new SqlConnection(con());
            cnn.Open();
            SqlCommand query = new SqlCommand("Select clima, count(*) from clima group by
clima having clima='Lluvia'", cnn);
            SqlDataReader dr;
            dr = query.ExecuteReader();

            if (dr.HasRows)
            {
                while (dr.Read())
                {
                    diaLluvia = "Total de días de lluvia: " + dr[1].ToString();
                }
            }
            else
            {
                diaLluvia = "No hay días de sequía";
            }
            return diaLluvia;

        }

        public String PicosLluvia()
        {
            String diaLluvia = "";
            String sqlquery = @"declare @val int; set @val = (select top 1 count(*) from
clima group by perimetro order by perimetro desc); select TOP(@val) dia from clima order
by perimetro desc";
            SqlConnection cnn = new SqlConnection(con());
            cnn.Open();

            SqlCommand query = new SqlCommand(sqlquery, cnn);
            SqlDataReader dr;
            dr = query.ExecuteReader();

            if (dr.HasRows)
            {
                diaLluvia = "Los días con picos máximos de lluvia son: ";
                while (dr.Read())
                {
                    diaLluvia+= dr[0].ToString() + ", ";
```

```
                }
            }
            else
            {
                diaLluvia = "No hay días de sequía";
            }
            return diaLluvia;

        }



        public String diaOptimo()
        {
            String diaOptimo = "";

            SqlConnection cnn = new SqlConnection(con());
            cnn.Open();
            SqlCommand query = new SqlCommand("Select clima, count(*) from clima group by
clima having clima='Optimo'", cnn);
            SqlDataReader dr;
            dr = query.ExecuteReader();

            if (dr.HasRows)
                {
                    while (dr.Read())
                    {
                        diaOptimo = "Total de días óptimos: " + dr[1].ToString();
                    }
            }
            else
            {
                diaOptimo = "No hay días de sequía";
            }
            return diaOptimo;

        }
    }
```