Tresture Hunt

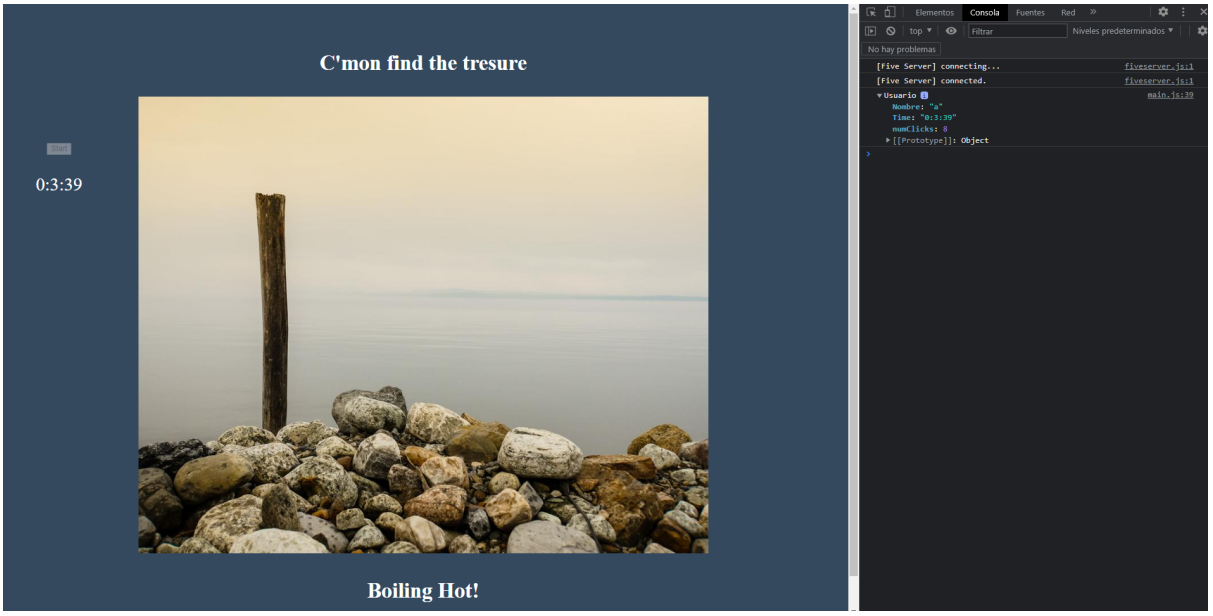
Fernando Fernández Martínez
1 DAW
https://github.com/Sr-fer/TresureHunt

Índice:

Información del juego

Este juego trata de encontrar el tesoro, cada jugador tendrá una imagen, en esta se esconderá un tesoro, este tesoro es aleatorio siempre que se juega, el jugador tendrá que ir “clickando” la pantalla mientras se le dará unas pistas de frío y caliente hasta que se acabe encontrando el tesoro, a su vez, el jugador también dispondrá de un temporizador que contará el tiempo que tarda el usuario en encontrar el tesoro.

Estos datos serán guardados con el nombre del usuario en una base de datos para posteriormente ser comparado con otros jugadores.



Código

El código estará estructurado en tres clases principales.

La clase usuario que servirá para guardar los datos del usuario, en este caso el nombre, el tiempo y el número de clicks para encontrar el tesoro.

```
class User{

    /**
    * @type {Number} clicks del jugador
    * @type {Number} Tiempo del jugador
    * @type {String} Nombre del jugador
    * @param {String} id Id con el que es referenciada cada clase del objeto
    */

    constructor(id){
        this.clicks = 0
        this.time = 0
        this.userName;
        this.id = id
    }
```

La clase de configuración de la imagen, que tratará de declarar el tamaño de alto y ancho de la imagen.

```
class ImageAdjust {

    /**
    * @type {Number} Altura Imagen
    * @type {Number} Ancho Imagen
    * @param {String} id Id con el que es referenciada cada clase del objeto
    */

    constructor(id) {
        this.WIDTH = 0
        this.HEIGH = 0
        this.id = id
    }
```

La clase Chrono que se trata de la clase del temporizador que medirá el tiempo que tarda en encontrar el tesoro el usuario.

```
class Chrono {

    /**
    * @type {Number} Contador del cronometro
    * @type {Number} Minutos del cronometro
    * @type {Number} Segundos del cronometro
    * @type {Number} Milisegundos del cronometro
    * @type {String} Texto que representa el tiempo del cronometro
    * @type {String} Botón para emepezar el cronometro
    * @type {String} Intervalo del cronometro
    * @param {String} id Id con el que es referenciada cada clase del objeto
    */

    constructor(id) {
      this.chronoCounter = 0
      this.chronoMinutes = 0
      this.chronoSeconds = 0
      this.chronoMiliseconds = 0
      this.chronoText;
      this.start;
      this.chronoInterval = null
      this.id = id
    }
```

A su misma vez el proyecto estará estructurado con el modelo MVC.

Empezamos instanciando el Controller, después ya pasamos con el Modelo, donde se define todas las funciones y sus variables que necesita el código para funcionar, excluidas así las que tengan que ver con el apartado gráfico, entre el controller y el modelo tendrás que poder realizar las funciones del programa desde la consola, entonces es dónde se puede empezar con la parte gráfica o view, la tercera de las partes de este MVC.

```
class Controller {

    /**
    * @param {Number} model objeto de la clase Model
    * @param {Number} view objeto de la clase View
    */

    constructor(model, view) {
        this.model = model
        this.view = view
```

Repositorio

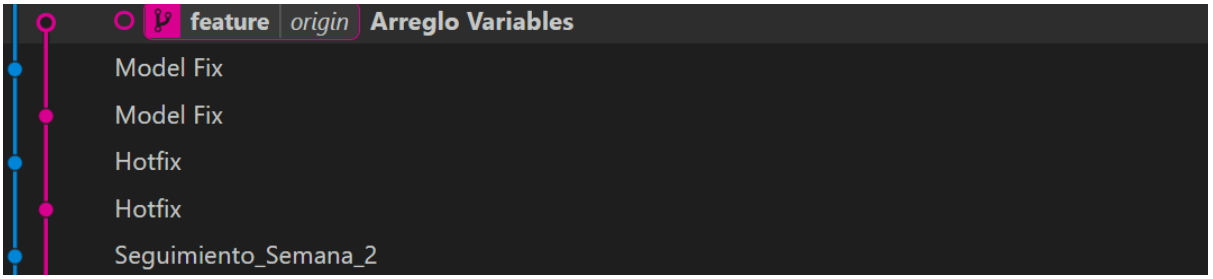
El repositorio tendrá dos proyectos, uno para la parte de documentación y otro para la parte del código.

Cada uno corresponderá con sus respectivos Issues que se irán resolviendo a lo largo del proyecto.

El proyecto consta de tres ramas diferenciadas.

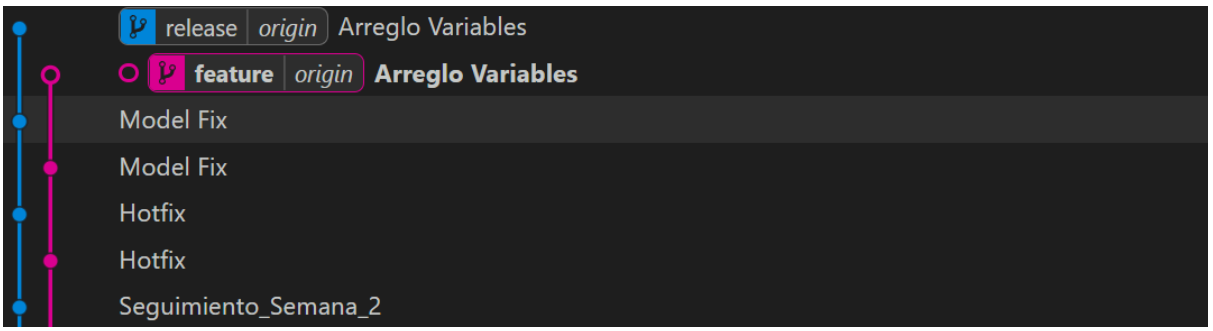
Features:

Rama donde se trabajará normalmente y donde tendremos la mayoría de commits



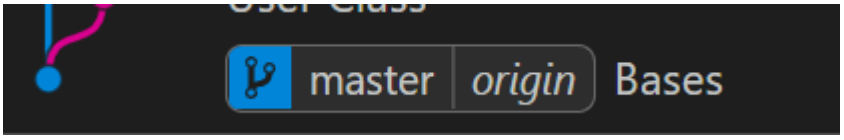
Release:

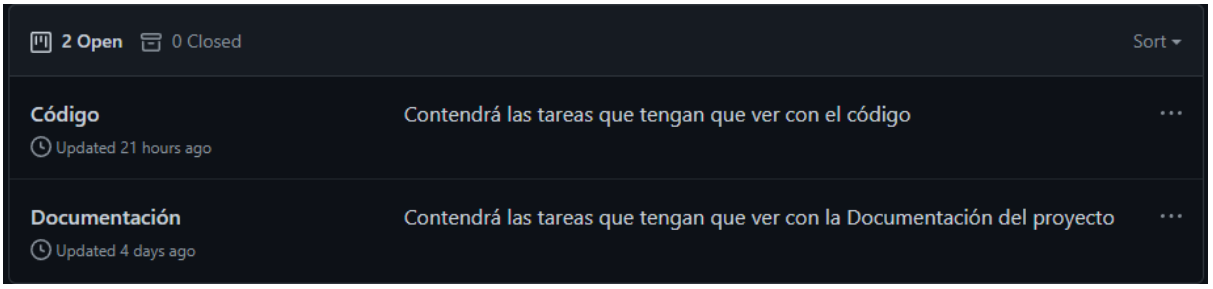
Rama donde se juntara con la features y su respectivo commit cada vez que el proyecto de un paso importante hacia delante.

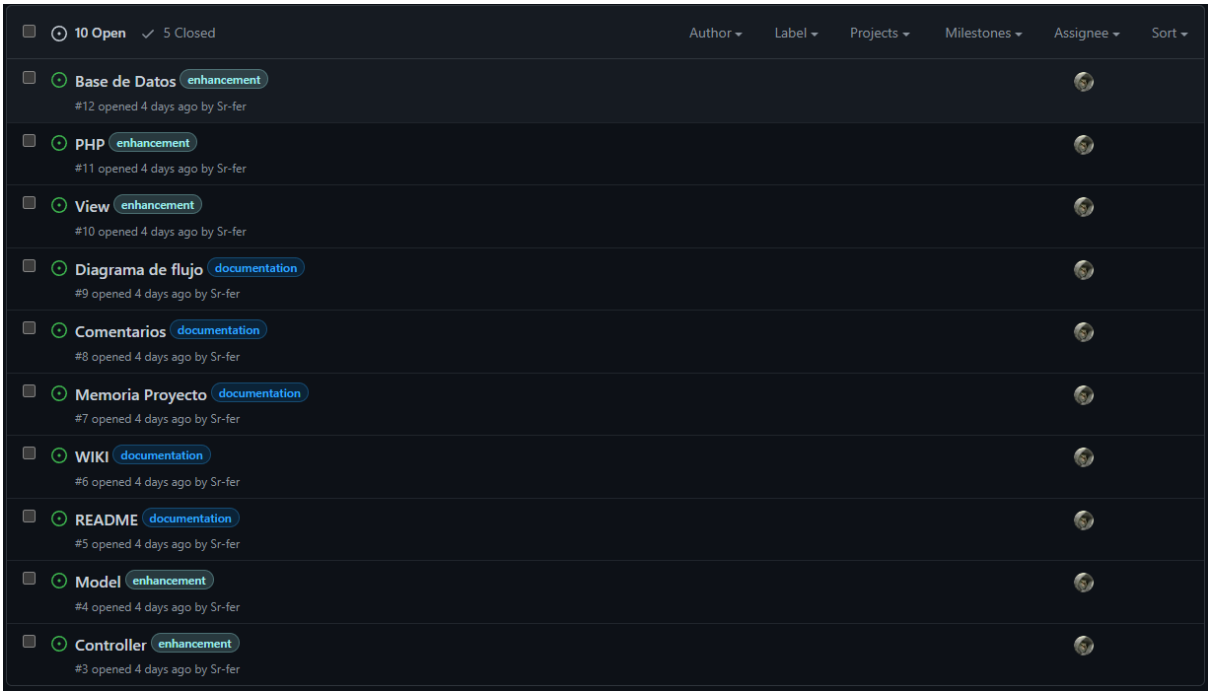


Master:

Esta rama se juntará con el resto cuando diferenciada por versiones donde cada una de estas es el programa de forma funcional y solo añadiendo mejoras o cambios notorios para la salida del mismo.

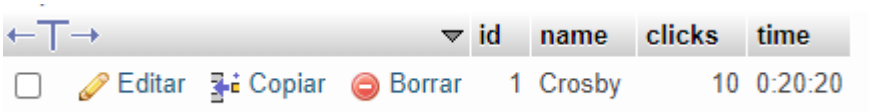






PHP y Bases de Datos

El proyecto tendrá una base de datos donde se guardará los datos del usuario y se comparará con otros haciendo así un ranking, el cual será proporcionado a la base de datos por un archivos php, y devuelto desde el mismo al JS, obteniendo así de forma visible para el usuario el ranking.




```php
<?php
class User{
    public $Name;
    public $Time;
    public $Clicks;
        function __construct($Name,$Time,$Clicks){
            $this->Name=$Name;
            $this->Time=$Time;
            $this->Clicks=$Clicks;
        }
}

    $serverName = "localhost";
    $user = "root";
    $password = "";
    $database = "ranking";
    $conection = new mysqli($serverName,$user,$password,$database);

        if($conection->connect_error){
            die("Failed: ". $conection->connect_error);
        }

    $consult = "SELECT * FROM ranking";
    $result = $conection->query($consult);
    $users = array();
        if($result->num_rows>0){
            while($row = $result->fetch_assoc()){
            $aux_user = new User($row["name"],$row["time"],$row["clicks"]);
            array_push($users,$aux_user);
        }
        }else{
            print"Error: ".$conection->error;
        }
        echo json_encode($users);
?>
```

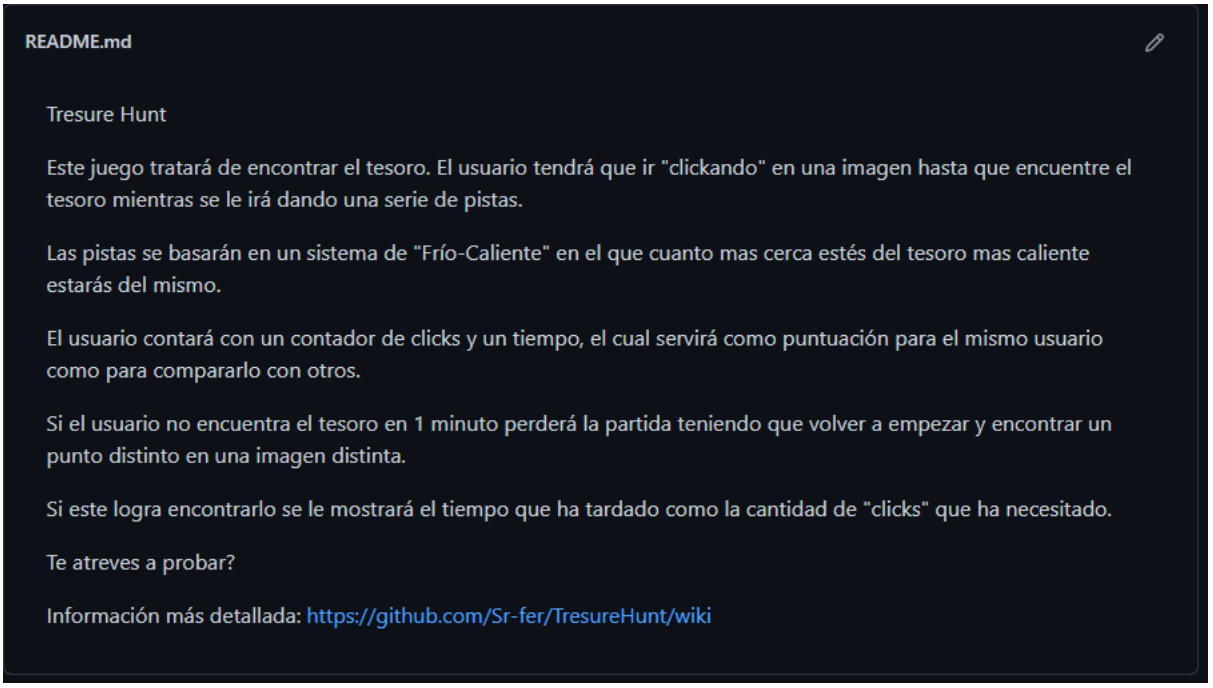
```javascript
sendRequest(auxUser) {
    $.ajax({
        data:{"name": auxUser.userName ,"clicks": auxUser.clicks , "time": auxUser.time},
        url:'php/main.php',
        type:'get',
        success:function(response){
        var conect = JSON.parse(response);
        console.log(conect);
        for(var i=0;i<conect.length;i++){
            console.log("Name: "+ conect[i].name +" Time: "
            + conect[i].time + " Clicks: "+ conect[i].clicks);
        }
        }
    })
```

Documentación

La documentación del proyecto será basada en un un README y una WIKI.

README:

Contendrá de forma resumida aunque completa la funcionalidad del programa.



README.md

Tresure Hunt

Este juego tratará de encontrar el tesoro. El usuario tendrá que ir "clickando" en una imagen hasta que encuentre el tesoro mientras se le irá dando una serie de pistas.

Las pistas se basarán en un sistema de "Frío-Caliente" en el que cuanto mas cerca estés del tesoro mas caliente estarás del mismo.

El usuario contará con un contador de clicks y un tiempo, el cual servirá como puntuación para el mismo usuario como para compararlo con otros.

Si el usuario no encuentra el tesoro en 1 minuto perderá la partida teniendo que volver a empezar y encontrar un punto distinto en una imagen distinta.

Si este logra encontrarlo se le mostrará el tiempo que ha tardado como la cantidad de "clicks" que ha necesitado.

Te atreves a probar?

Información más detallada: https://github.com/Sr-fer/TresureHunt/wiki

WIKI:

Contendrá la organización del proyecto así como las bases explicadas del mismo.

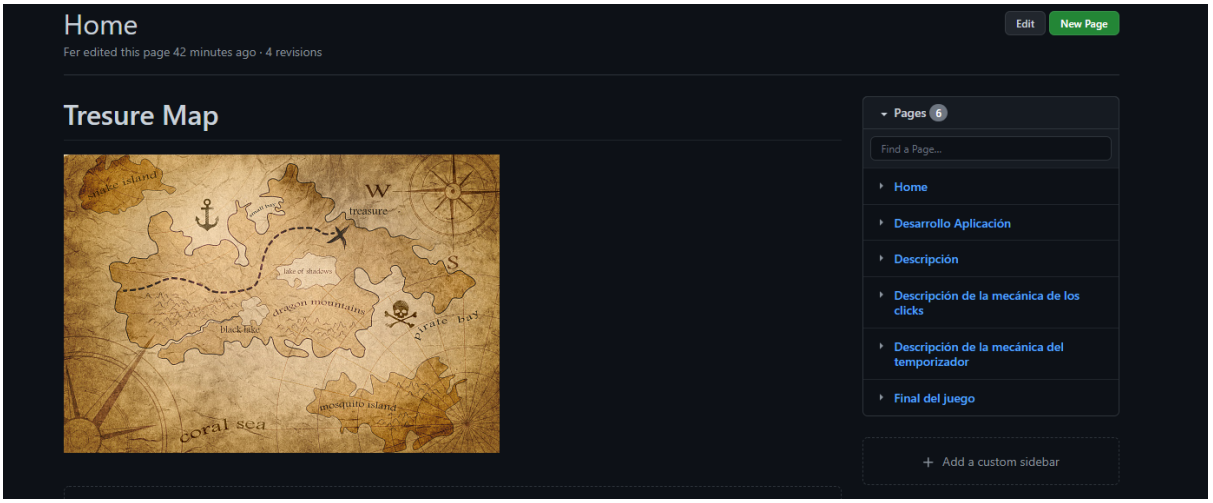
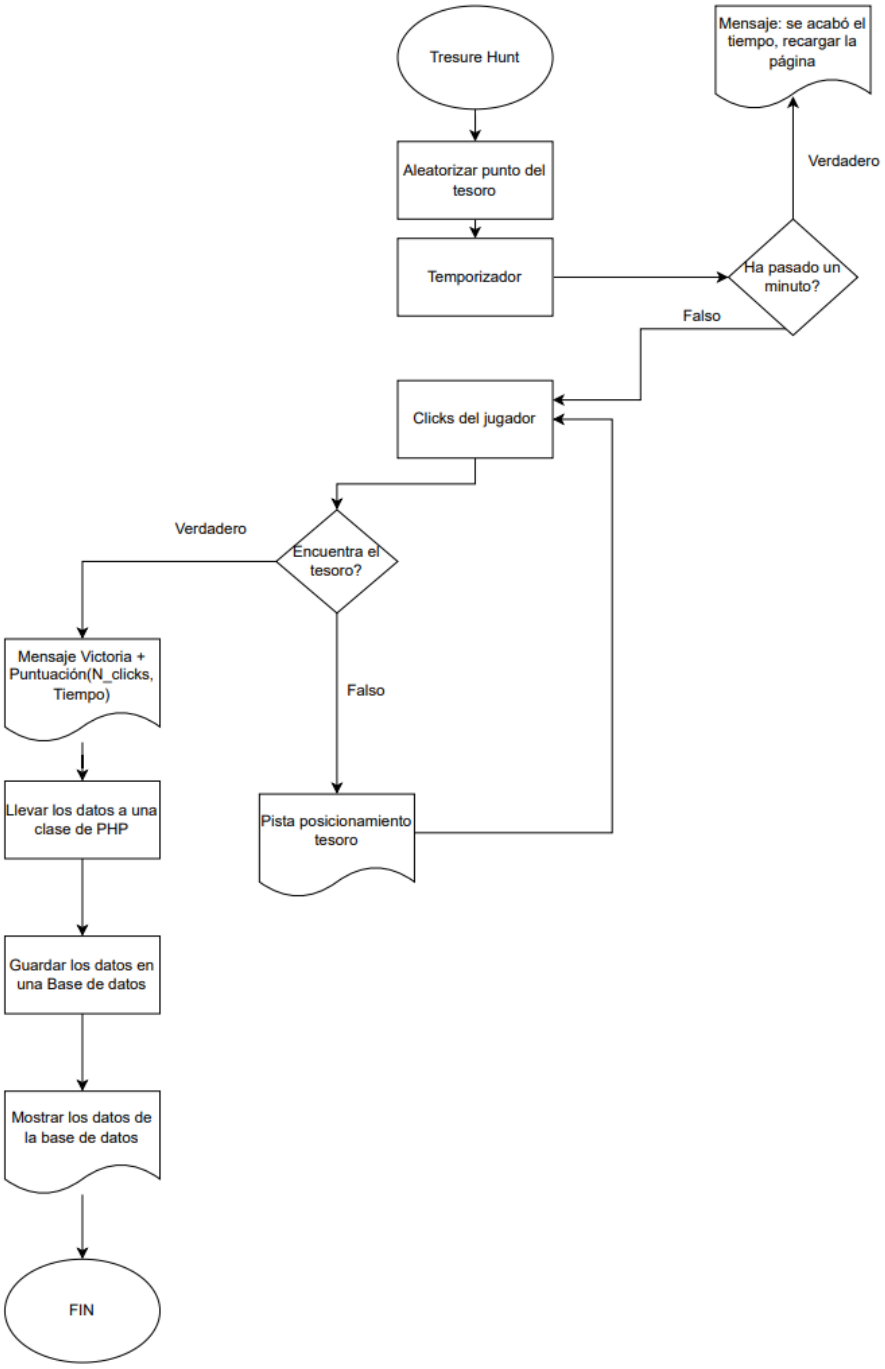


Diagrama:

El proyecto contará con unos esquemas que definen la distribución proceso y orden del mismo, este o estos esquemas representarán un breve resumen y como empezar a trabajar en el proyecto de forma ordenada y precisa para no perdernos durante el desarrollo del mismo.


Tresure Hunt
Aleatorizar punto del tesoro
Temporizador
Ha pasado un minuto?
Verdadero
Mensaje: se acabó el tiempo, recargar la página
Falso
Clicks del jugador
Encuentra el tesoro?
Verdadero
Falso
Mensaje Victoria + Puntuación(N_clicks, Tiempo)
Pista posicionamiento tesoro
Llevar los datos a una clase de PHP
Guardar los datos en una Base de datos
Mostrar los datos de la base de datos
FIN