



Proyecto Gamificación

24/05/2023

—

Centro Universitario de Occidente (CUNOC)

Teoría de Sistemas 1

#3 - Oscar Antonio De León Urizar - 201830498

#4 - Carlos Benjamin Pac Flores - 201931012

#13 - Fernando Rodriguez - 202030542

Descripción General.....	4
Arquitectura de la Aplicación.....	4
Diagrama de Entidades.....	4
Diagramas.....	5
Diseños de Vistas.....	5
Diagrama de Secuencia:.....	8
Login.....	8
Registro.....	9
Diagrama de Interacciones:.....	10
Registro de Usuario:.....	10
Login:.....	11
Creación de Juego Personalizado:.....	12
Obtención de los logros generales.....	13
Obtener logros de ahorcado.....	14
Obtener logros de Hanoi.....	15
Obtener logros de sopa de letras.....	16
Juegos a Crear:.....	16
Documentación de Juegos:.....	17
Ahorcado:.....	17
Creación de Juego:.....	17
Jugar una partida:.....	18
Crucigrama:.....	18
Sopa de Letras:.....	18
Torre de Hanoi:.....	18
Diagrama de Clases:.....	19
- Base de Datos:.....	20
Trofeos.....	20
API Gamificación.....	20
Login:.....	20
Registro:.....	20
Modificar:.....	21
Modificar Password:.....	21
Obtener Usuario:.....	22
Agregar un logro:.....	22
Comentarios:.....	23
Registro de Comentarios:.....	23



- Obtener comentarios:.....23
- Manejo de Juegos:.....24
 - Obtener Juego base:.....24
 - Obtener Juegos base:.....25
- Manejo de likes juegos:.....25
 - Registro de like:.....25
 - Obtener Like:.....26
 - Eliminar Like:.....26
- Juegos Personalizados.....27
 - Registro de Juego Personalizado:.....27
 - Obtener Juego Personalizado:.....27
 - Obtener modelos creados por un usuario:.....28
 - Obtener modelo por su codigo.....28
 - Borrar un modelo.....29
- Resultados de Partidas.....29
 - Registro de Partida:.....29
 - Obtener Resultado:.....30
 - Obtener Partidas de un juego:.....30
 - Obtener Estadísticas Generales:.....31
 - Obtener Estadísticas de Ahorcado:.....31
- Ranking de juegos.....32

Descripción General

Arquitectura de la Aplicación

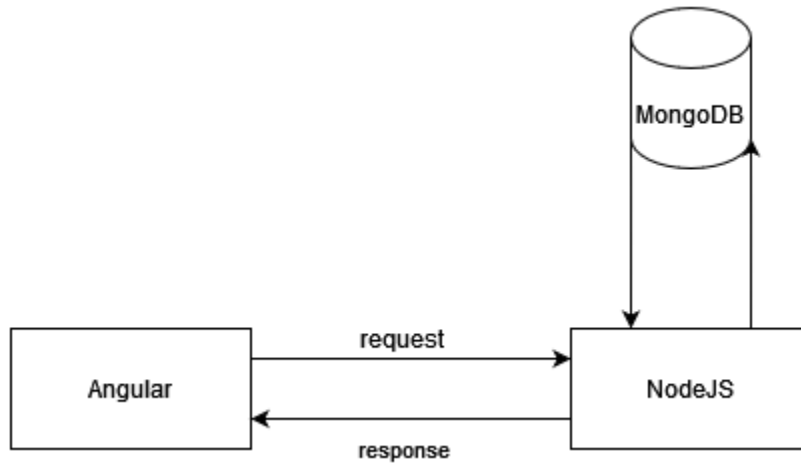


Diagrama de Entidades

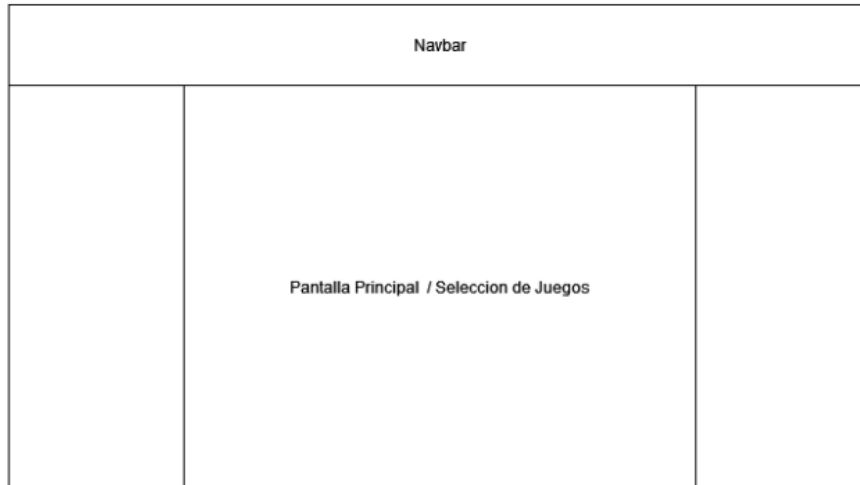
Usuario	
	Username
	Password
	Rol

Juego	
	Nombre
	Autor/es
	Partidas

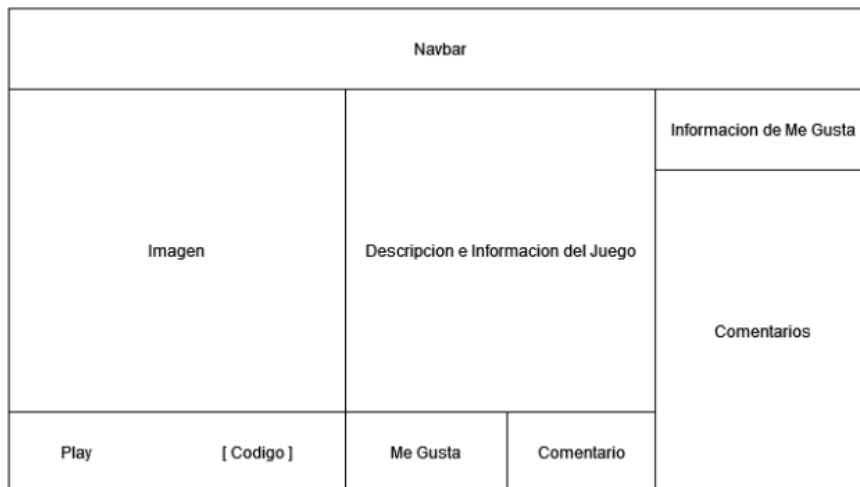
Diagramas

Diseños de Vistas

Pantalla Principal



Pantalla de Informacion de Juego





Pantalla de Juego

Navbar	
Pantalla Principal / Seleccion de Juegos	Info. Jugador Actual
	Estadisticas

Login

Navbar
Username y Password

Registro

Navbar	
Username y Password	Tipo de Usuario

Casos de Uso Generales:

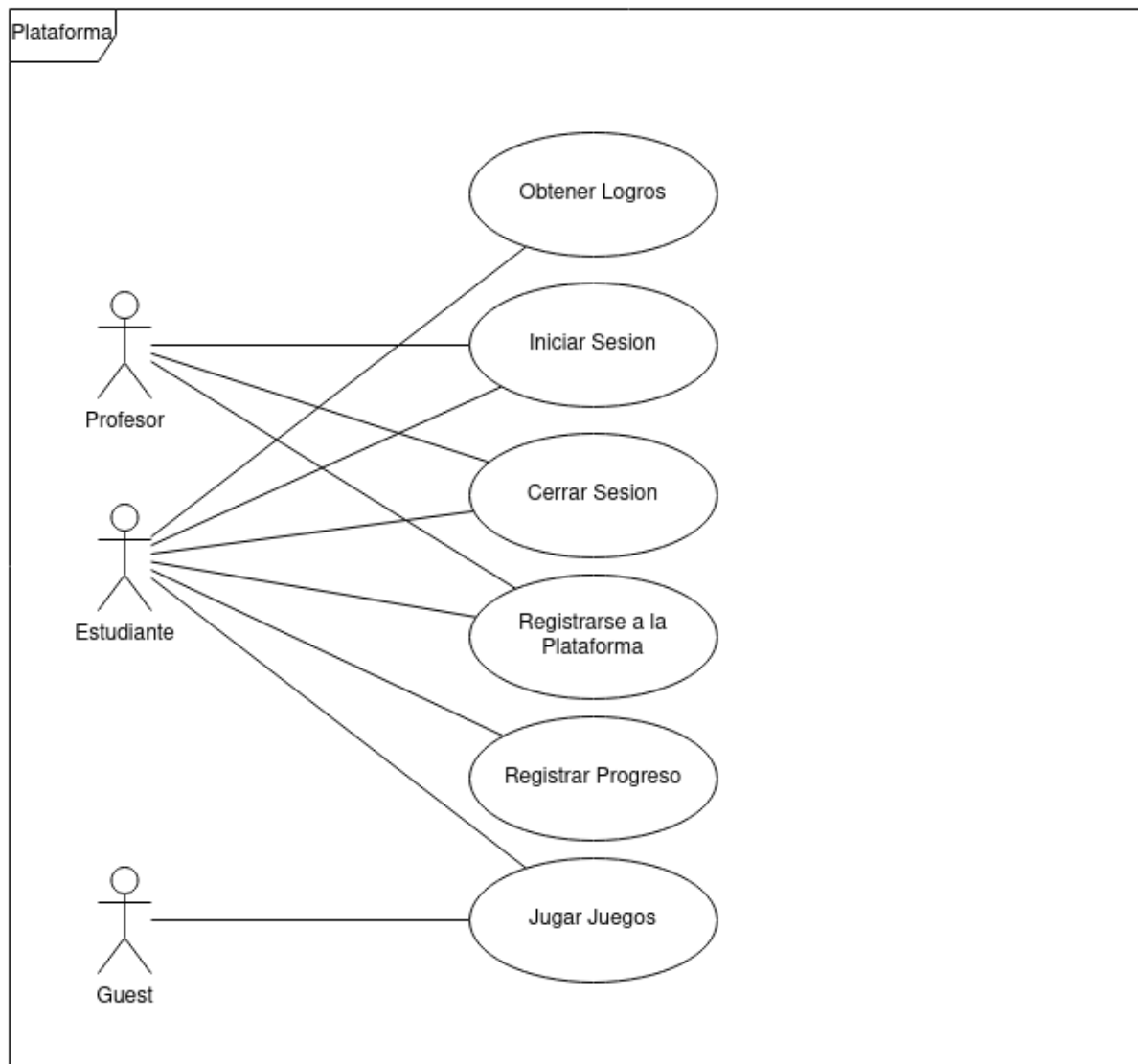
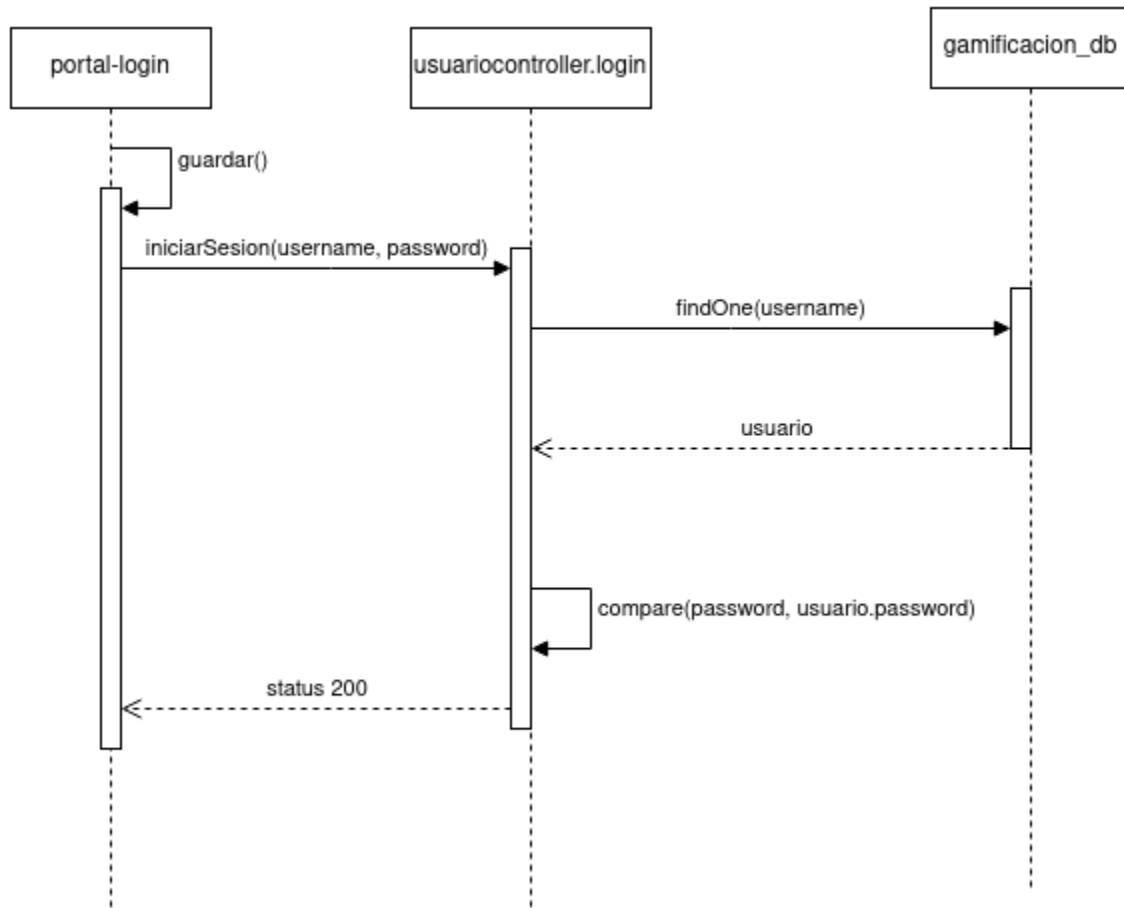


Diagrama de Secuencia:

Login



Registro

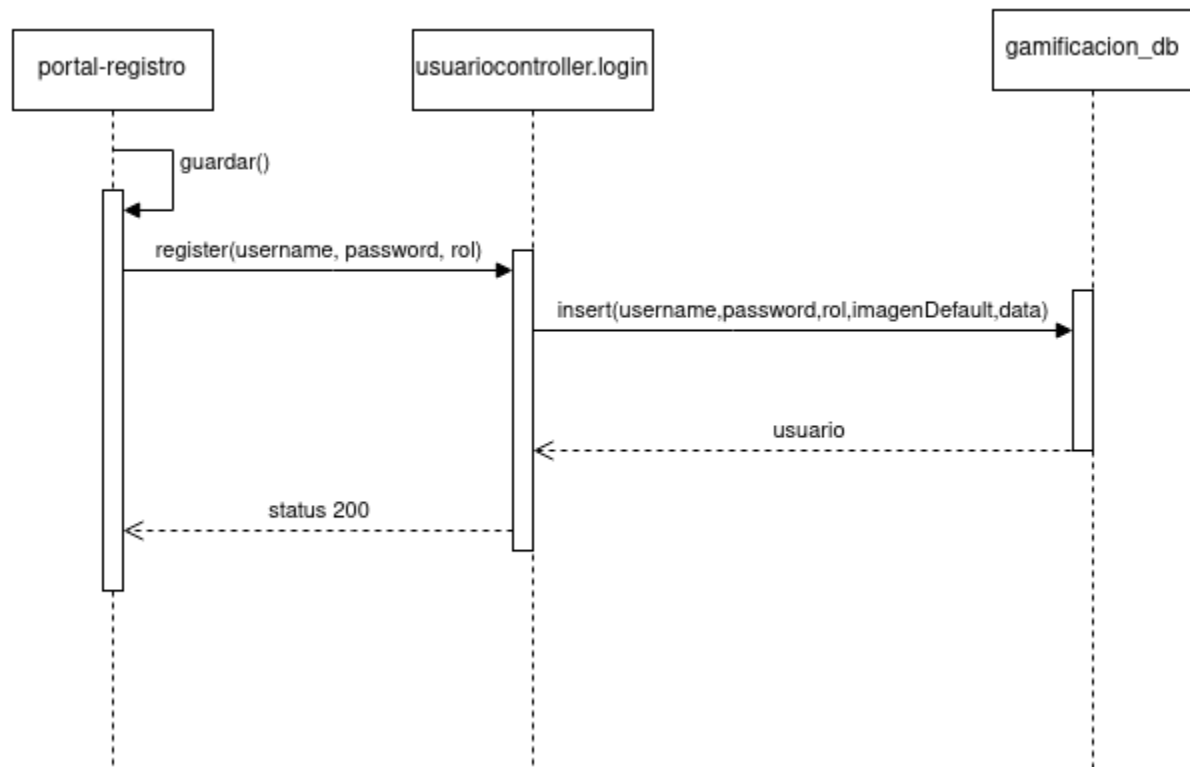
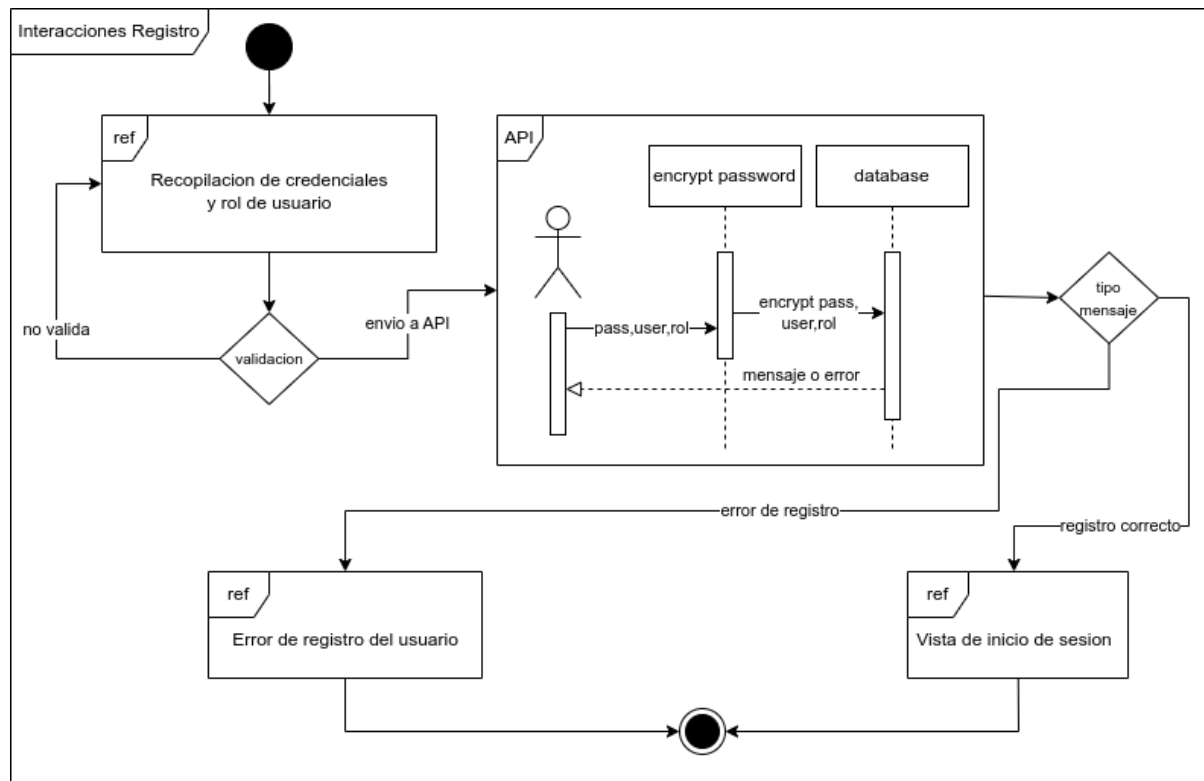
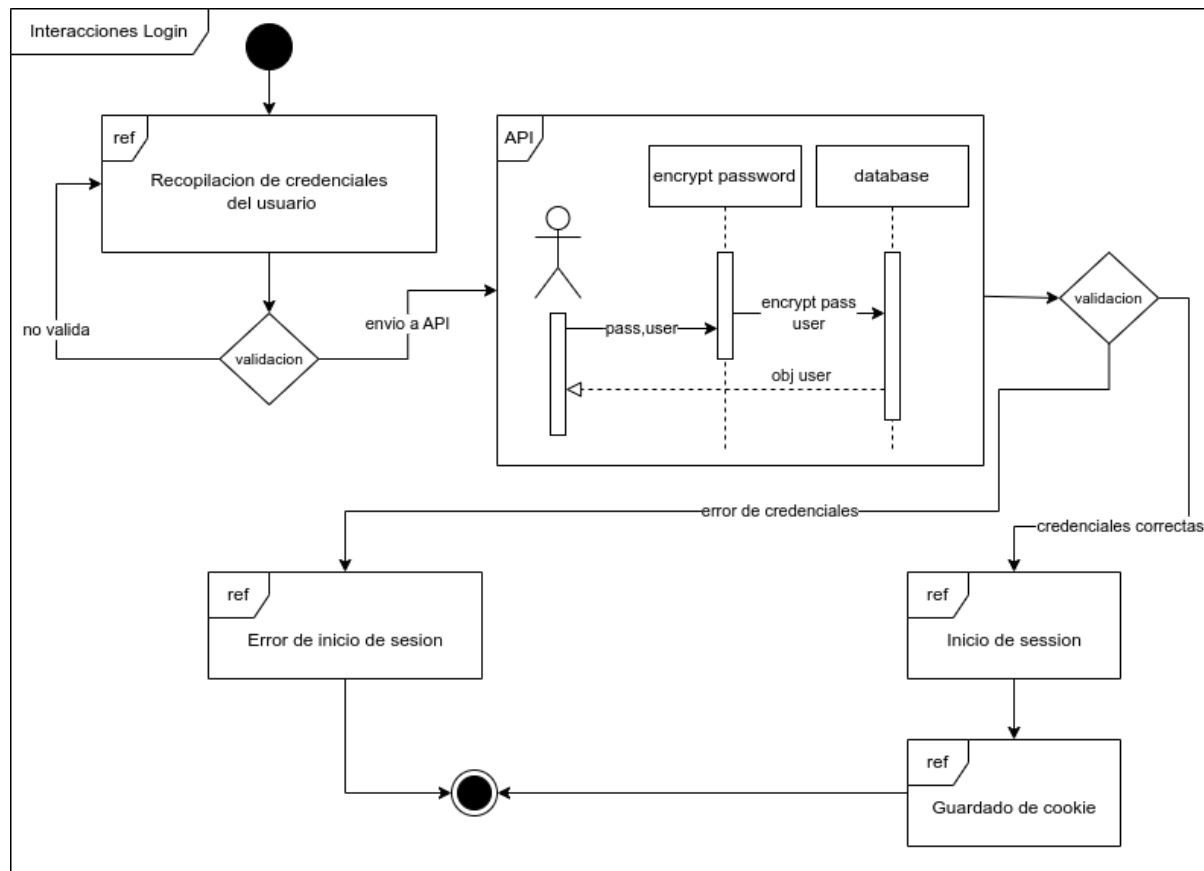


Diagrama de Interacciones:

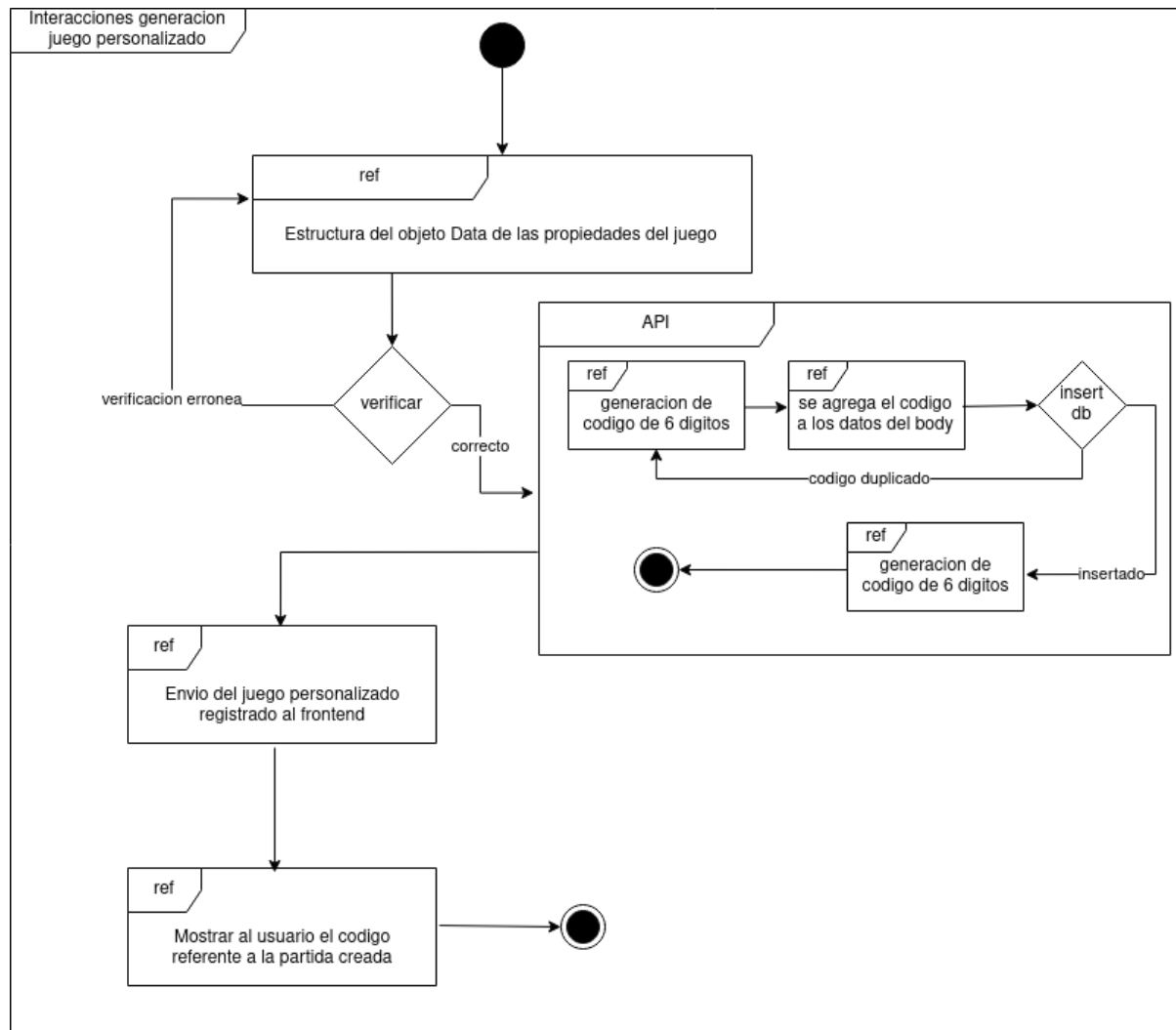
Registro de Usuario:



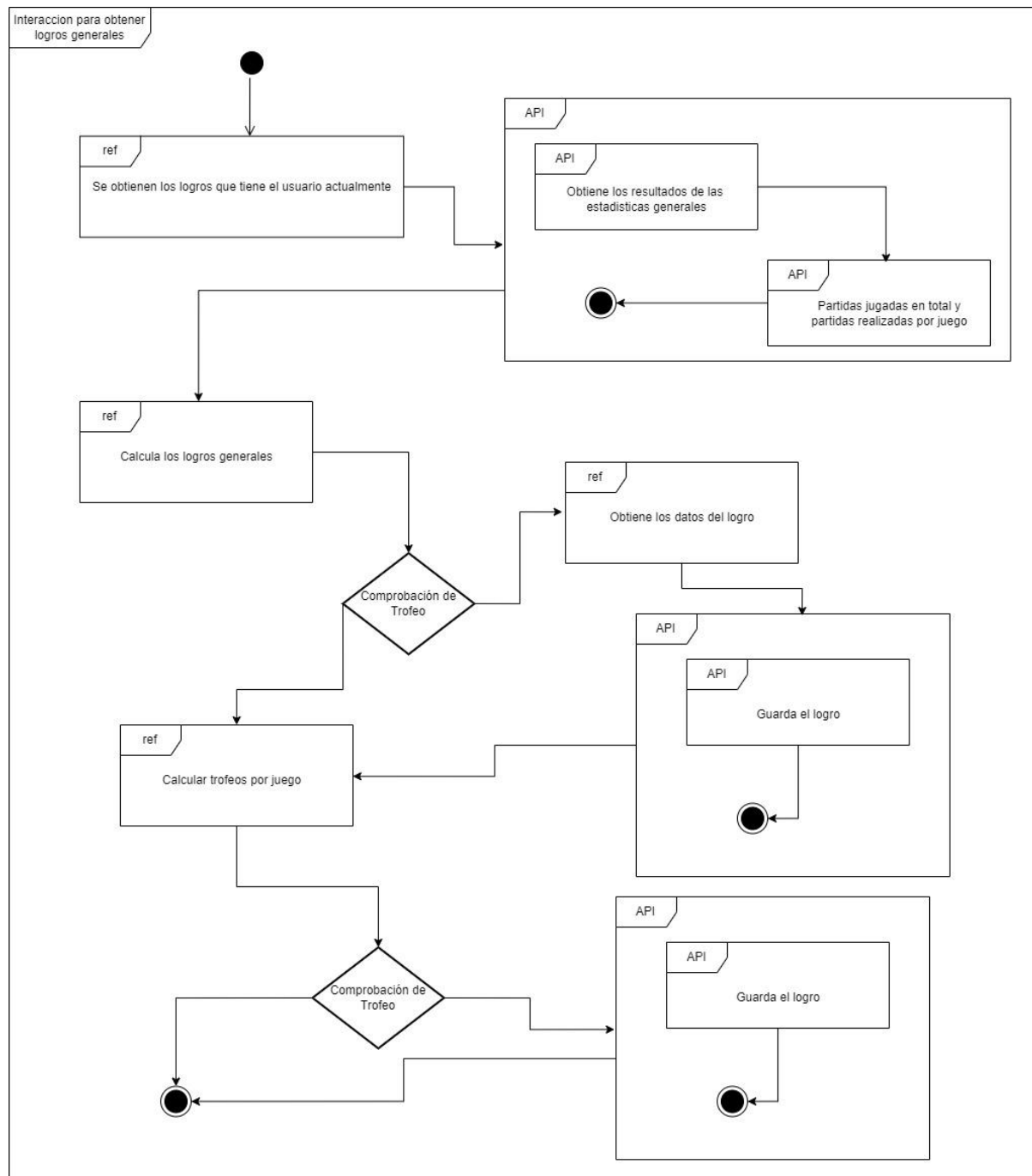
Login:



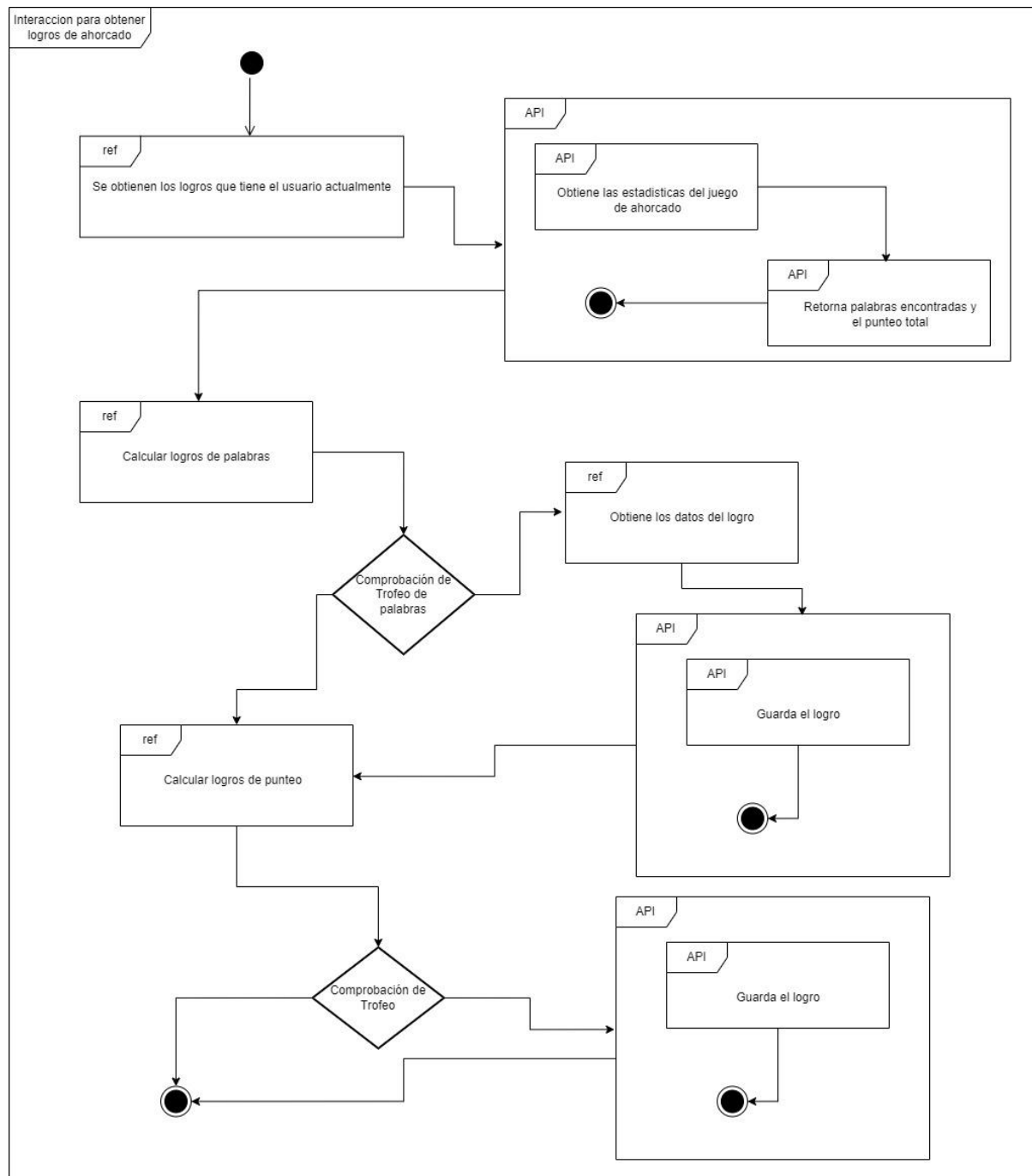
Creación de Juego Personalizado:



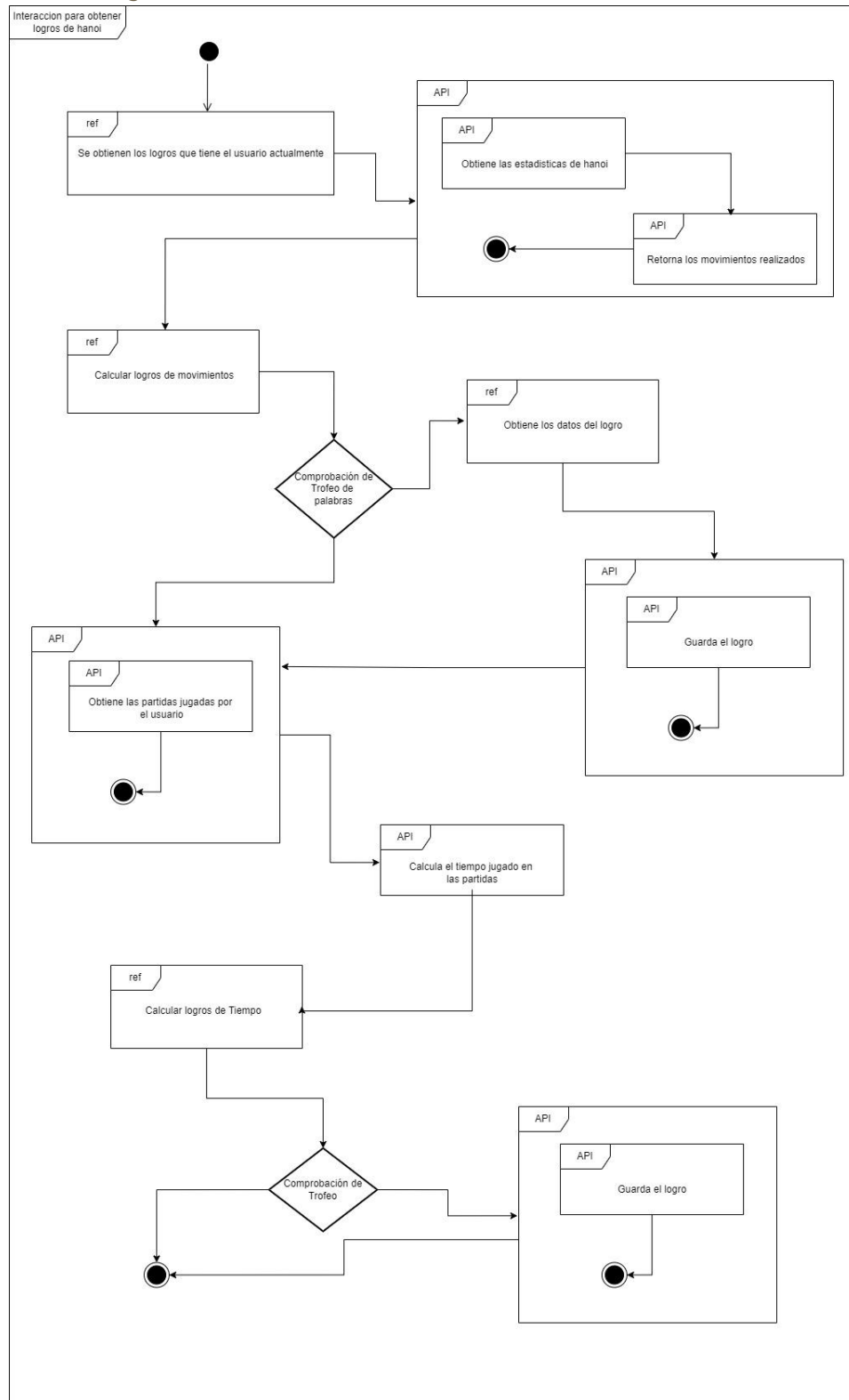
Obtención de los logros generales



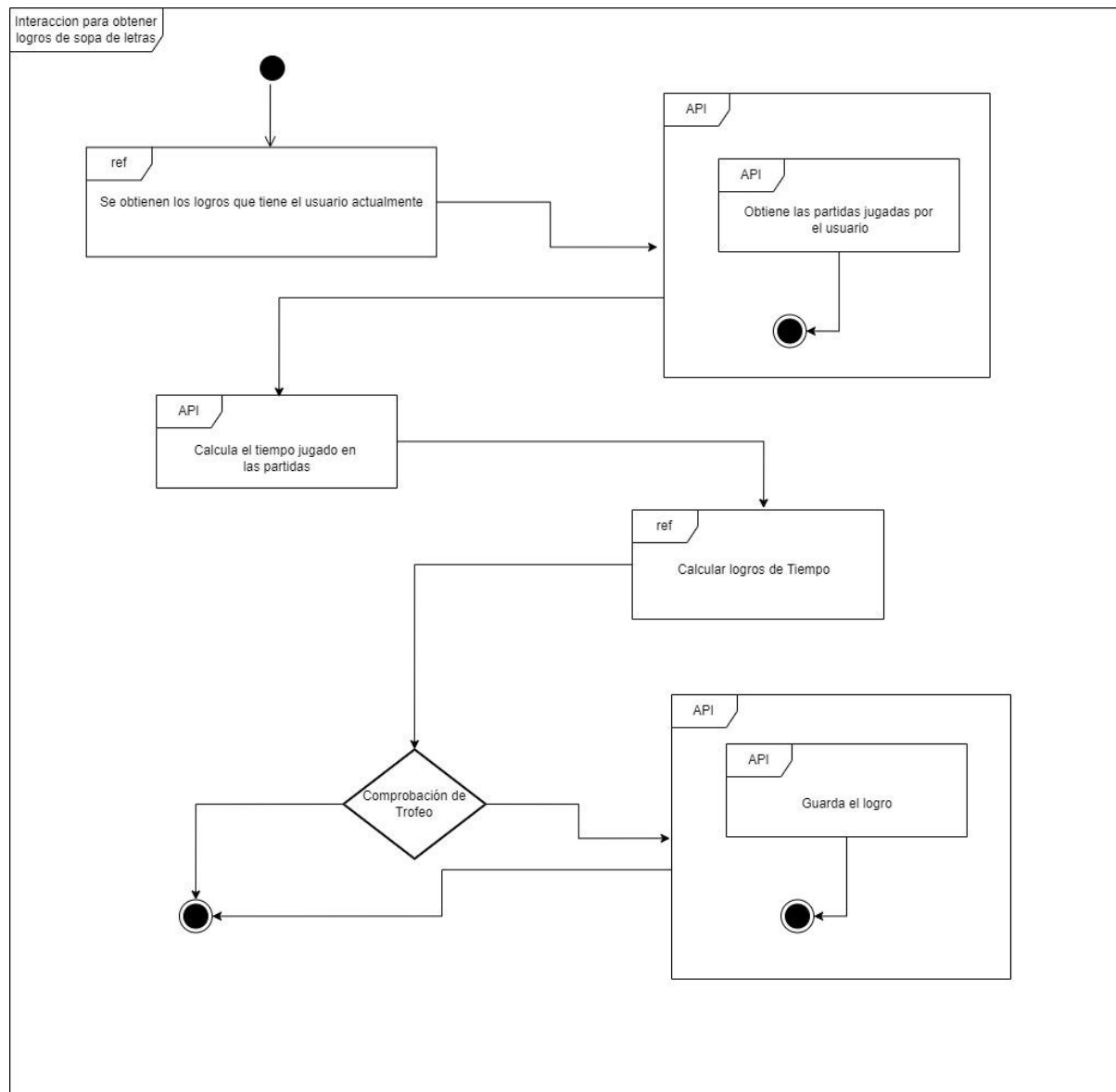
Obtener logros de ahorcado



Obtener logros de Hanoi



Obtener logros de sopa de letras



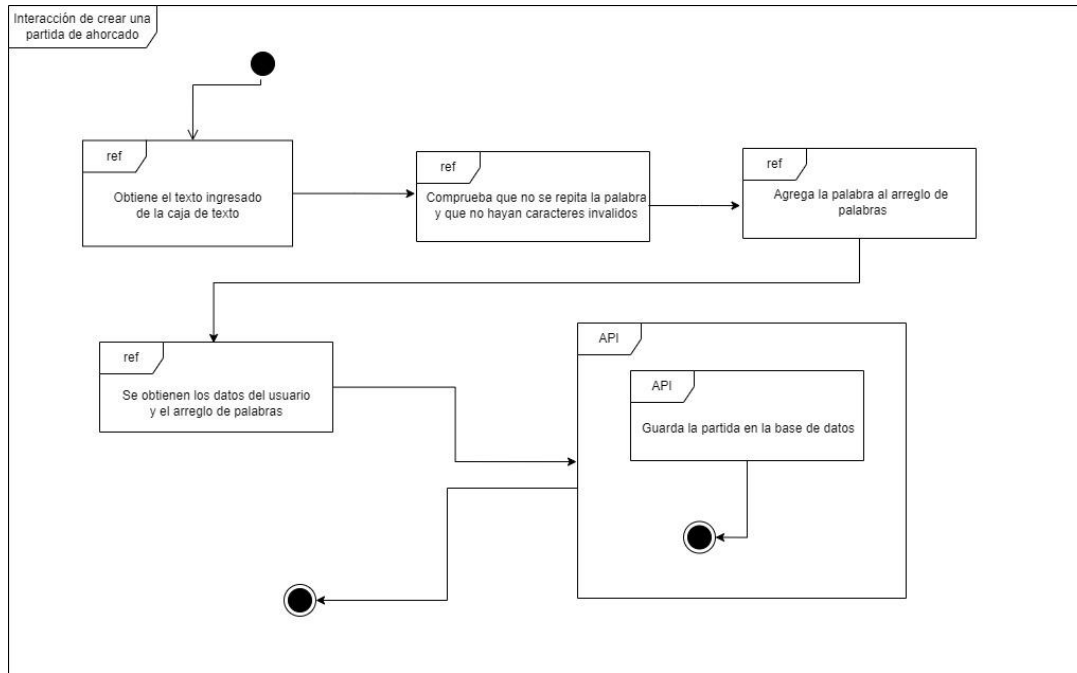
Juegos a Crear:

1. Ahorcado - Oscar
2. Crucigrama - Todos
3. Sopa de Letras - Fernando
4. Torre de Hanoi - Carlos

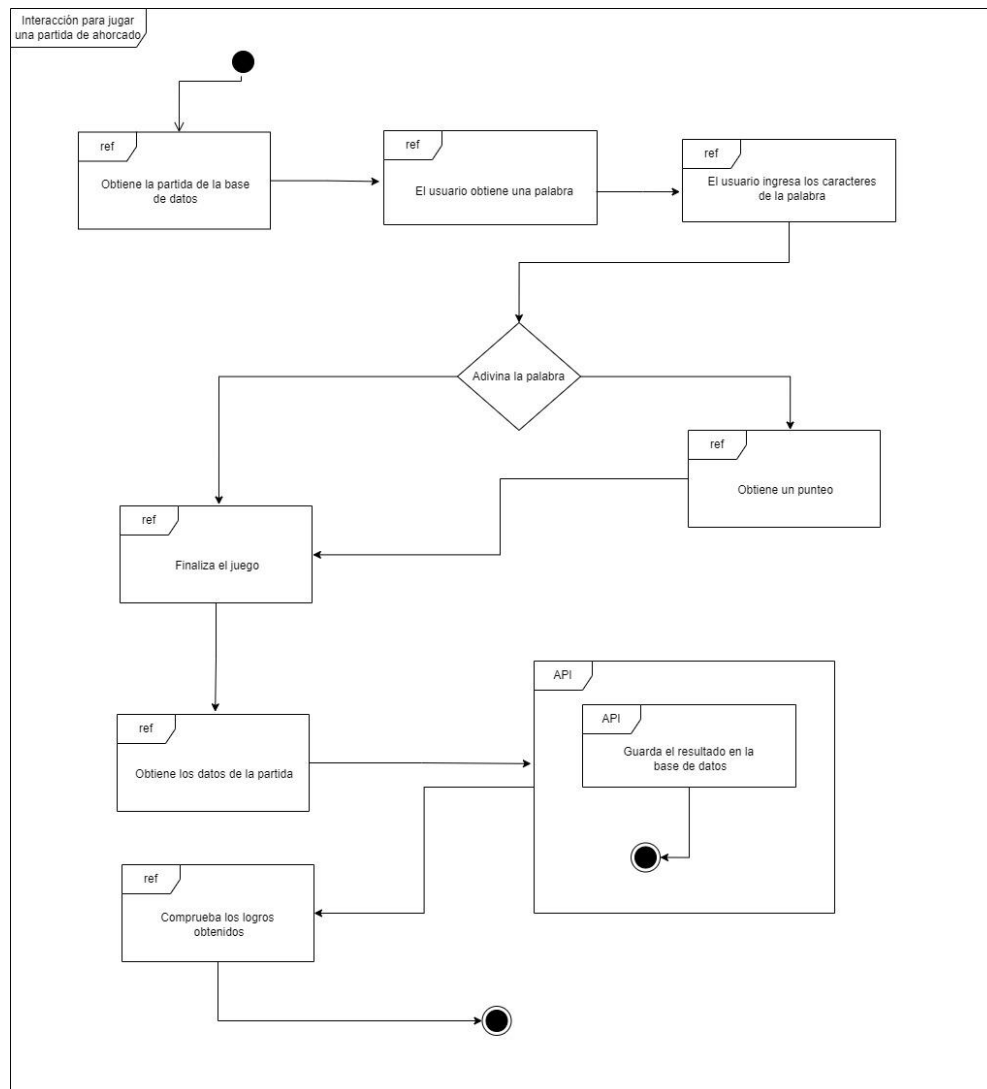
Documentación de Juegos:

Ahorcado:

Creación de Juego:



Jugar una partida:



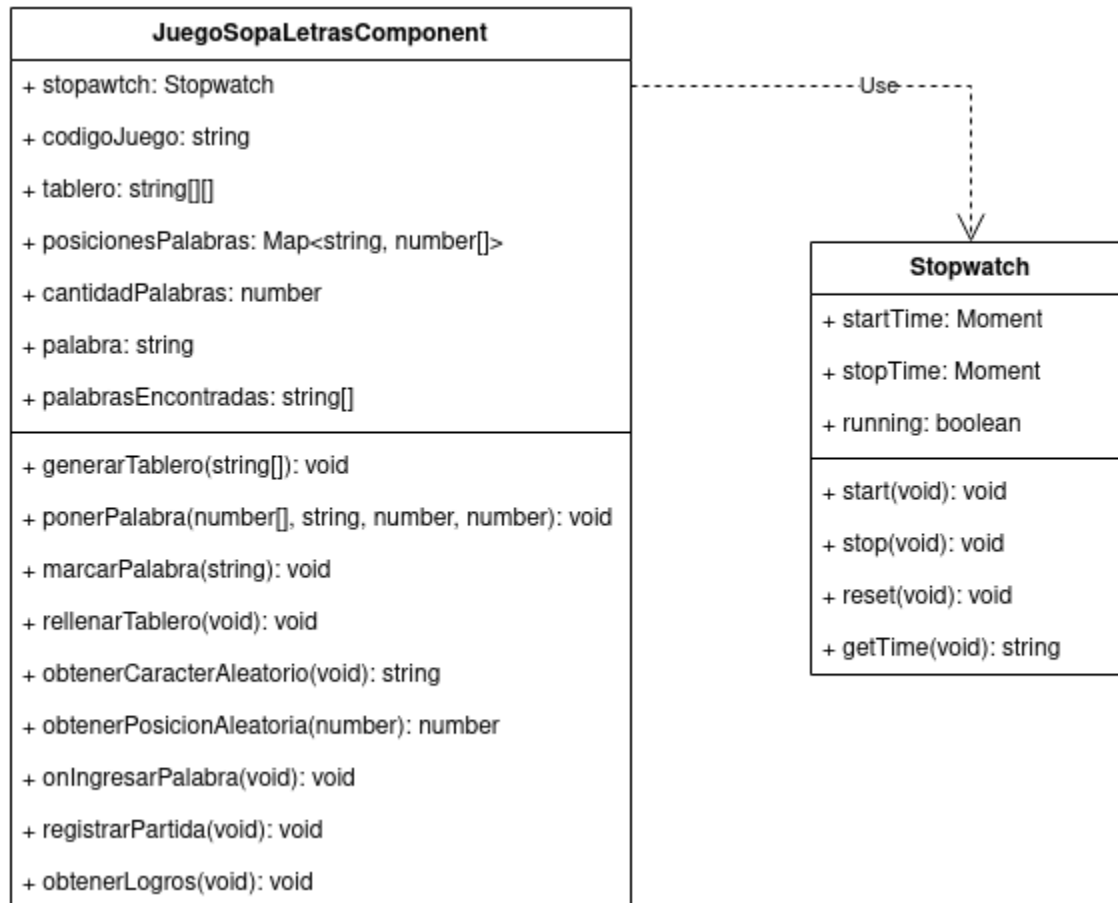
Crucigrama:

Diagrama de Clases:

JuegoAhorcadoComponent
+ numeroPalabra: number + botonDesabilitado: boolean + palabraActual: string + cantidadErrores: number + cantidadAciertos: number + srcImagen: string + punteoTotal: number + codigoJuego: string + palabrasEncontradas: number + palabrasFalladas: number
+ obtenerPalabra(void): void + clickTecla(string, event): void + finalizarJuego(void): void + obtenerLogros(string): void + cambiarEstadoBotones(boolean): void + terminarJuego(void): void

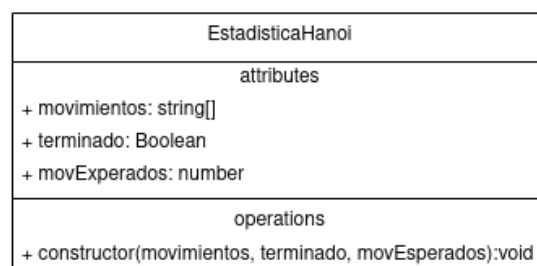
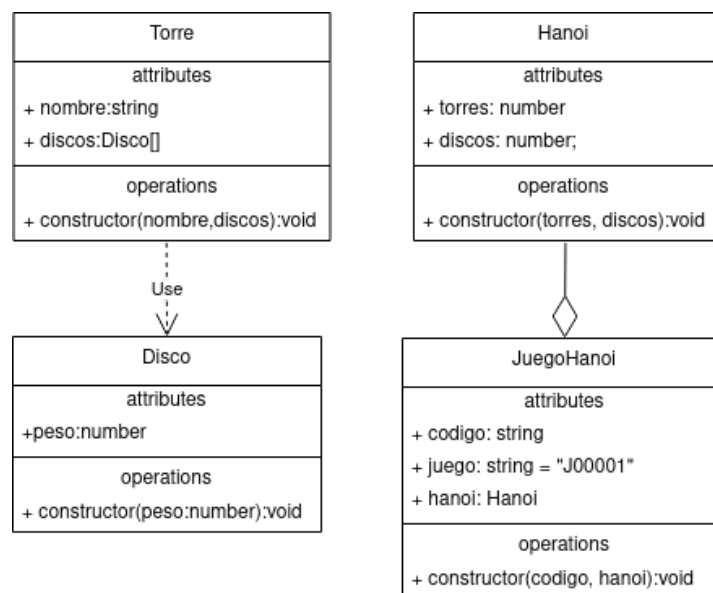
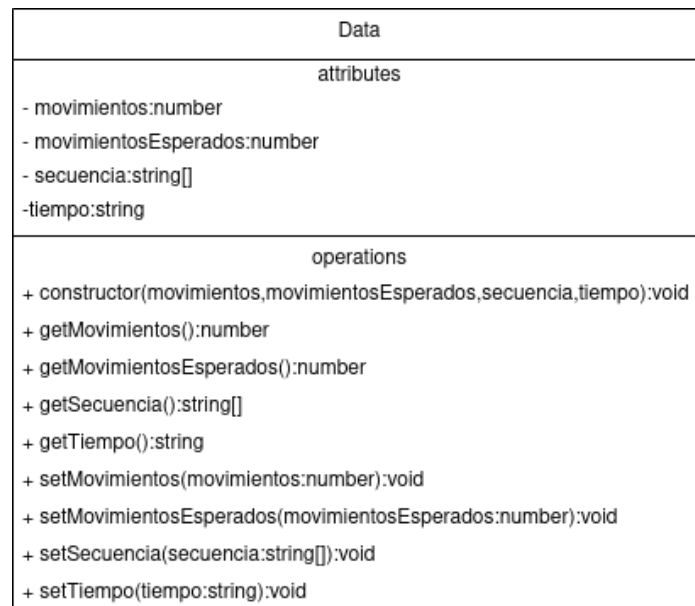
Sopa de Letras:

Diagrama de Clases:



Torre de Hanoi:

Diagrama de Clases:





Trofeos

- 1 partidas jugadas
- 5 partidas jugadas
- 10 partidas jugadas

API Gamificación

La api funciona bajo la dirección <http://localhost:5000>

Login:

Ruta: /api/login

Tipo: POST

Parámetros: username, password

Tanto el username y password son enviados sin encriptar a la ruta de la api.

Respuesta: { mensaje: 'Sesión iniciada, 'usuario encontrado' }

Respuesta de error: { error: 'error encontrado' }

Registro:

Ruta: /api/registro

Tipo: POST

Parámetros: El payload tiene el siguiente formato JSON

```
{
  username: string,
  password: string,
  rol: number
}
```

Dónde rol tipo number tiene los siguientes valores 0 -> Guest, 1 -> Usuario, 2 -> Profesor/Creador



Tanto el username y password son enviados sin encriptar a la ruta de la api.

Respuesta: { mensaje: 'Se registró con éxito el usuario' }

Respuesta de error: { error: 'error encontrado' }

Modificar:

Ruta: /api/modificar

Tipo: PUT

Parámetros: El payload tiene el siguiente formato JSON

```
{
  usernameActual: string,
  nuevoUsername: string,
  perfil: string
}
```

Respuesta: { resultado }

Respuesta de error: { error: 'error encontrado' }

Modificar Password:

Ruta: /api/modificarPassword

Tipo: PUT

Parámetros: El payload tiene el siguiente formato JSON

```
{
  username: string,
  passwordActual: string,
  nuevaPassword: string
}
```

Respuesta: { resultado }

Respuesta de error: { error: 'error encontrado' }



Obtener Usuario:

Ruta: /api/usuario

Tipo: GET

Parámetros: username

Respuesta: {

```
  _id:string,  
  usuario:String,  
  rol:String,  
  perfil:String,  
  data:Object
```

}

Respuesta de error: { error: 'No existe el usuario "username"' }

Agregar un logro:

Ruta: /api/agregarLogro

Tipo: PUT

Parámetros: {

```
  username: String // Username del usuario,  
  logro: Object // Logro que se le agregará al usuario
```

}

Respuesta: Actualizacion

Respuesta de error: { error: error.message }


Comentarios:

Registro de Comentarios:

Ruta: /api/comentario

Tipo: POST

Parámetros: El payload tiene el siguiente formato JSON



```
{
  usuario: string,
  juego: string,
  comentario:string
}
```

Respuesta: el objeto del comentario

```
{
  _id:string,
  usuario:String,
  juego:String,
  comentario:String,
  fecha:String
}
```

Respuesta de error: { error: 'error encontrado' }

Obtener comentarios:

Ruta: /api/comentarios

Tipo: GET

Parámetros: juego -> código del juego "J00001" -> Hanoi, "J00002" -> Ahorcado -> "J00003" -> Crucigrama, "J00004" -> Sopa de Letras

Respuesta: array de comentarios -> [{...},{...},{...}]

formato del comentario:

```
{
  _id:string,
  usuario:String,
  juego:String,
  comentario:String,
  fecha:String
}
```

Respuesta de error: { error: 'error encontrado' }

Manejo de Juegos:

Obtener Juego base:

Ruta: /api/juego

Tipo: GET

Parámetros: juego -> código del juego "J00001" -> Hanoi, "J00002" -> Ahorcado -> "J00003" -> Crucigrama, "J00004" -> Sopa de Letras

Respuesta: JSON juego de la siguiente forma

```
{
  _id:String,
  codigo:String,
  nombre:String,
  autores:String,
  like:Array,
  likes:Number,
  dislikes:Number
}
```

El array de like está compuesto por objetos de la siguiente composición:

```
{
  usuario:String,
  like:bool
}
```

Respuesta de error: { error: 'error encontrado' }

Obtener Juegos base:

Ruta: /api/juegos

Tipo: GET

Parámetros: Ninguno

Respuesta: array de juegos base-> [{...},{...},{...}]



formato:

```
{
  _id:String,
  codigo:String,
  nombre:String,
  autores:String,
  like:Array,
  likes:Number,
  dislikes:Number
}
```

Respuesta de error: { error: 'error encontrado' }

Manejo de likes juegos:

Registro de like:

Ruta: /api/juego/like

Tipo: POST

Parámetros: El payload tiene el siguiente formato JSON

```
{
  juego:String, //Codigo del juego
  usuario:String, //Nombre de usuario
  like:bool, //Estado true or false
}
```

Respuesta: El juego asociado al like con la nueva información

```
{
  _id:String,
  codigo:String,
  nombre:String,
  autores:String,
  like:Array,
  likes:Number,
  dislikes:Number
}
```



```
}
```

Respuesta de error: { error: 'error encontrado' }

Obtener Like:

Ruta: /api/juego/like

Tipo: GET

Parámetros: juego -> código del juego, usuario -> usuario asociado

Respuesta: JSON juego de la siguiente forma

formato:

```
{
  usuario:string,
  like:any //true -> like, false -> dislike, null -> sin asignar
}
```

Respuesta de error: { error: 'error encontrado' }

Eliminar Like:

Ruta: /api/juego/like

Tipo: DELETE

Parámetros: El payload tiene el siguiente formato JSON

```
{
  juego:String, //Codigo del juego
  usuario:String, //Nombre de usuario
}
```

Respuesta: El juego asociado al like y usuario con la nueva información

```
{
  _id:String,
  codigo:String,
  nombre:String,
  autores:String,
  like:Array,
  likes:Number,
```

```
    dislikes:Number
}
```

Juegos Personalizados

Registro de Juego Personalizado:

Ruta: /api/modelo

Tipo: POST

Parámetros: El payload tiene el siguiente formato JSON

```
{
  juego:String, //Codigo del juego
  usuario:String, //Nombre de usuario
  data: Object //Información de la partida personalizada
}
```

Respuesta:

```
{
  codigo:String, //Codigo de partida personalizada
  juego:String, //Codigo del juego
  usuario:String, //Nombre de usuario
  data:Object //Información de la partida personalizada
}
```

Obtener Juego Personalizado:

Ruta: /api/modelo


Tipo: GET

Parámetros: El query debe de contener los parámetros

- codigo:String, //Codigo de partida personalizada
- juego:String, //Codigo del juego

Respuesta:

```
{
  codigo:String, //Codigo de partida personalizada
  juego:String, //Codigo del juego
  usuario:String, //Nombre de usuario
}
```



```
data:Object //Información de la partida personalizada
}
```

Obtener modelos creados por un usuario:

Ruta: /api/modelosUsuario

Tipo: GET

Parámetros: El query debe de contener los parámetros

- usuario:String, //Username del usuario creador

Respuesta: Array de modelos con la siguiente estructura:

```
{
  codigo:String, //Codigo de partida personalizada
  juego:String, //Codigo del juego
  usuario:String, //Nombre de usuario
  data:Object //Información de la partida personalizada
}
```

Obtener modelo por su codigo

Ruta: /api/modeloPorCodigo

Tipo: GET

Parámetros: El query debe de contener los parámetros

- codigo:String, // El código del usuario

Respuesta:

```
{
  codigo:String, //Codigo de partida personalizada
  juego:String, //Codigo del juego
  usuario:String, //Nombre de usuario
  data:Object //Información de la partida personalizada
}
```

Respuesta de Error:

```
{error: 'No existe una partida personalizada con el codigo "codigo"}
```



Borrar un modelo

Ruta: /api/modelo

Tipo: DELETE

Parámetros: El query debe de contener los parámetros

- `codigo:String, //Codigo de el modelo`

Respuesta: Array de modelos con la siguiente estructura:

```
{
  deletedCound = 1
}
```

Respuesta de Error:

```
{error: 'No existe una partida personalizada con el codigo "codigo"')}
```

Resultados de Partidas

Registro de Partida:

Ruta: /api/partida


Tipo: POST

Parámetros: El payload tiene el siguiente formato JSON

```
{
  codigo:String, //Codigo de partida personalizada
  juego:String, //Codigo del juego
  usuario:String, //Nombre de usuario
  fecha: string, // Fecha de realización de la partida
  data: Object //Resultado de la partida
}
```

Respuesta:

```
{
  codigo:String, //Codigo de partida personalizada
  juego:String, //Codigo del juego
  usuario:String, //Nombre de usuario,
```



```
    fecha: string, // Fecha de realización de la partida
    data:Object //resultado de la partida
}
```

Obtener Resultado:

Ruta: /api/partida

Tipo: GET

Parámetros: El query debe de contener los parámetros

- usuario:String, //Nombre de usuario
- codigo:String, //Codigo de partida personalizada
- juego:String, //Codigo del juego

Respuesta:

```
{
    codigo:String, //Codigo de partida personalizada
    juego:String, //Codigo del juego
    usuario:String, //Nombre de usuario
    fecha: String, // Fecha de la partida
    data:Object //resultado de la partida
}
```

Obtener Partidas de un juego:

Ruta: /api/partidasPorJuego


Tipo: GET

Parámetros: El query debe de contener los parámetros

- usuario:String, //Nombre de usuario
- juego:String, //Codigo del juego

Respuesta: Array de partidas [{}], con el formato de:

```
{
    codigo:String, //Codigo de partida personalizada
    juego:String, //Codigo del juego
    usuario:String, //Nombre de usuario
}
```

```
    fecha: String, // Fecha de la partida
    data:Object //resultado de la partida
}
```

Obtener Estadísticas Generales:

Ruta: /api/estadisticasGenerales

Tipo: GET

Parámetros: El query debe de contener los parámetros

- username:String, //Nombre de usuario

Respuesta:

```
{
  generales:number, //Partidas jugadas en total
  juegos: {
    _id: String // Nombre del juego,
    partidas: number // Partidas jugadas
  }
}
```

Obtener Estadísticas de Ahorcado:

Ruta: /api/estadisticasGenerales

Tipo: GET

Parámetros: El query debe de contener los parámetros

- username:String, //Nombre de usuario

Respuesta:

```
{
  _id: null,
  palabrasEncontradas: string, // Palabras encontradas en el juego
  palabrasFalladas: string, // Palabras no encontradas en el juego
  punteo: number // Puntaje total obtenido en el juego
}
```

Ranking de juegos

Ruta: /api/ranking



Tipo: GET

Parámetros: El query debe de contener los parámetros

- juego:String, //Codigo del juego base

Respuesta: Un array de objetos

Para Juego J00001:


```
[
  {
    _id: string, //Nombre de usuario del jugador
    jugado: number, //Cantidad de veces que se jugó el juego
    movimientosDeMas: number, //Cantidad de movimientos extra
    tiempo: number //Tiempo en segundos en el juego
  },
  ...
]
```

Para Juego J00002:

```
[
  {
    _id: string, //Nombre de usuario del jugador
    jugado: number, //Cantidad de veces que se jugó el juego
    palabrasEncontradas: number, //Cantidad de palabras encontradas
    palabrasFalladas: number, //Cantidad de palabras falladas
    punteo: number //Cantidad de puntos del usuario
  },
  ...
]
```

Para Juego J00003:

```
[
  {
    _id: string, //Nombre de usuario del jugador
    jugado: number, //Cantidad de veces que se jugó el juego
```



```
    tiempo: number //Tiempo en segundos en el juego
  },
  ...
]
```

Para Juego J00004:

```
[
  {
    _id: string, //Nombre de usuario del jugador
    jugado: number, //Cantidad de veces que se jugó el juego
    tiempo: number //Tiempo en segundos en el juego
  },
  ...
]
```