



Programa educativo

INGENIERÍA EN SISTEMAS
COMPUTACIONALES

Grupo

8 "B"

Nombre de la materia

ARQUITECTURA DE SERVICIOS

Nombre del alumno

JAIME GONZÁLEZ VERA

Número y Nombre del trabajo

2.1 AVANCE DE PROYECTO

Unidad # 2

TECNOLOGÍAS PARA EL DESARROLLO DE
SERVICIOS

Nombre del Profesor

M.I.S. ROBERTO SUÁREZ ZINZÚN

Fecha

18 de marzo de 2024

Índice

| | |
|---|----|
| 1. Diseño del modelo de la base de datos..... | 3 |
| 2. Script de la BD..... | 6 |
| 2.1 Esquema de validación de la colección usuario..... | 7 |
| 2.2 Esquema de validación de la colección personaje..... | 8 |
| 2.3 Esquema de validación de la colección partida..... | 9 |
| 2.4 Esquema de validación de la colección compra..... | 10 |
| 3. Documentación de los Servicios..... | 13 |
| 3.1 Servicio REST usuarios..... | 13 |
| 3.1.1 Operación: Autenticar usuario..... | 13 |
| 3.1.2 Operación: Crear usuario..... | 14 |
| 3.1.3 Operación: Actualizar datos..... | 15 |
| 3.1.4 Operación: Eliminar usuario..... | 16 |
| 3.1.5 Operación: Consultar usuarios..... | 16 |
| 3.1.6 Operación: Consultar usuario por identificador..... | 17 |
| 3.2 Servicio REST compra..... | 17 |
| 3.2.1 Operación: Realizar compra..... | 18 |
| 3.2.2 Operación: Consultar compras..... | 19 |
| 3.2.3 Operación: Consultar compra por identificador..... | 20 |
| 3.3 Servicio REST personajes..... | 21 |
| 3.3.1 Operación: consultar personajes..... | 22 |
| 3.3.2 Operación: consultar personaje por identificador..... | 22 |
| 3.3.3 Operación: crear personaje..... | 23 |
| 3.3.4 Operación: actualizar personaje..... | 24 |
| 3.3.5 Operación: eliminar personaje..... | 25 |
| 3.4 Servicio REST partida..... | 25 |
| 3.4.1 Operación: Crear partida..... | 26 |
| 3.4.2 Operación: Finalizar partida..... | 27 |
| 3.4.3 Operación: Consultar partidas..... | 28 |
| 3.4.4 Operación: Consultar partida por identificador..... | 29 |
| 3.4.5 Operación: Agregar participante..... | 30 |

ACTIVIDAD 1. AVANCE DE PROYECTO

1. Diseño del modelo de base de datos

Para el modelado del diseño de la base de datos se propone que sea documental (NoSQL), puesto que para el tipo de entorno se requiere procesar una cantidad de datos muy grandes y tenerlo de manera rápida y sin entorpecer demás acciones, ya que en un modelo relacional para las consultas y las inserciones son muy estrictas y pudiesen ralentizar la jugabilidad y por ende una disminución de usuarios, por ello se decide utilizar NoSQL para maximizar la jugabilidad, la entrada de datos a gran escala.

El modelo consta de 4 entidades principalmente las cuales son: Usuarios, personajes, compra y partida, en los usuarios consta de atributos propios del usuario, así como un almacén donde dispondrá de los personajes que adquiera mediante compras, en segundo los personajes dispondrán únicamente de atributos como precio, imagen y nombre; en las compras los atributos se lleva como claves foráneas el identificador del usuario y el identificador del personaje para poder realizar la compra, tiene un arreglo de los personajes que se adquieran en dicha compra; por último, la partida requiere de identificadores de los personajes y de los usuarios que serán portadores de dichos personajes durante la partida.

A continuación, se detalla de manera documental y en lenguaje JSON de los documentos y su estructura.

```
{
  "_id": "ObjectId",
  "nombre": "string",
  "correo": "string",
  "estatus": "string",
  "contrasena": "string",
  "almacen": [
    {
      "idPersonaje": "string",
      "fecha_agregado": "datetime"
    }
  ]
}
```

Usuario

```
{
  "_id": "ObjectId",
  "nombre": "string",
  "precio": "float",
  "imagen": "string",
  "estatus": "string"
}
```

Personajes

```
{
  "_id": "ObjectId",
  "numTarjeta": "string",
  "cvc": "string",
  "anioExpiracion": "integer",
  "mesExpiracion": "integer",
  "subtotal": "float",
  "total": "float",
  "fechaCompra": "datetime",
  "idUsuario": "string",
  "detalleCompra": [{"detail": "object"}]
}
```

Compra

```
{
  "_id": "ObjectId",
  "duracion": "int",
  "fechaInicio": "str(date)",
  "fechaTerminacion": "str(date)",
  "horaInicio": "str(time)",
  "horaFin": "str(time)",
  "estatus": "char",
  "cupoMinimo": "integer",
  "cupoMaximo": "integer",
  "participantes": [{"participant": "object"}]
}
```

Partida

```
{  
  "estatus": "string",  
  "ganador": "boolean",  
  "usuario":  
    {  
      "idUser": "string",  
      "idPersonaje": "string"  
    }  
}
```

Objeto participant

```
{  
  "idPersonaje": "str",  
  "precio": "float"  
}
```

Objeto detail

2. Script de la BD

Para el Script de la base de datos se creó la base de datos en COMPASS que es de lo más usado para base de datos de manera documental, se crearon las colecciones y reglas de validación para tener cierto tipo de restricciones y evitar problemas con la base de datos.

Se crearon las 4 colecciones previamente mencionadas en el paso anterior (compra, partida, personaje, usuario), en estas colecciones se almacenarán los datos de manera masiva mayormente en la colección de partidas puesto que los usuarios harán uso de esta frecuentemente.

Se decidió usar COMPASS para efectos prácticos de la materia, así como tener conocimiento previo de esta herramienta de base de datos.

Connect Edit View Help

localhost:27017 ...

My Queries

Databases

Search

SurviveGame

- compra
- partida
- personaje
- usuario
- admin
- config
- local
- otraBD
- shopitesz

Collections

+ Create collection Refresh

View Sort by Collection Name

| Collection Name | Storage size | Documents | Avg. document size | Indexes | Total index size |
|-----------------|--------------|-----------|--------------------|---------|------------------|
| compra | 20.48 kB | 11 | 236.00 B | 1 | 24.58 kB |
| partida | 20.48 kB | 5 | 505.00 B | 1 | 20.48 kB |
| personaje | 20.48 kB | 10 | 234.00 B | 1 | 36.86 kB |
| usuario | 20.48 kB | 11 | 170.00 B | 1 | 24.58 kB |

2.1 Esquema de Validación de la colección usuario:

```
{
  $jsonSchema: {
    required: [
      'nombre',
      'correo',
      'estatus',
      'contrasena'
    ],
    properties: {
      nombre: {
        bsonType: 'string',
        description: 'nombre del usuario nombre, apellidos'
      },
      correo: {
        bsonType: 'string',
        pattern: '^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+.[a-zA-Z]{2,}$',
        description: 'correo del usuario'
      },
      estatus: {
        'enum': [
          'A',
          'I',
          'P'
        ],
        description: 'estatus del usuario'
      },
      contrasena: {
        bsonType: 'string',
        description: 'contrasena del usuario'
      }
    }
  }
}
```

2.2 Esquema de validación de la colección personaje:

```
{
  $jsonSchema: {
    required: [
      'nombre',
      'precio',
      'imagen'
    ],
    properties: {
      nombre: {
        bsonType: 'string',
        description: 'nombre del personaje'
      },
      precio: {
        bsonType: 'double',
        description: 'precio del personaje'
      },
      imagen: {
        bsonType: 'string',
        description: 'imagen relacionada al personaje'
      }
    }
  }
}
```

2.3 Esquema de validación de la colección partida:

```
{
  $jsonSchema: {
    required: [
      'duracion',
      'fechaInicio',
      'fechaTerminacion',
      'horaInicio',
      'horaFin',
      'estatus',
      'cupoMinimo',
      'cupoMaximo'
    ],
    properties: {
      duracion: {
        bsonType: 'double',
        description: 'Duracion de la partida'
      },
      fechaInicio: {
        bsonType: 'string',
        pattern: '^([0-9]{2})+([0-9]{2})+([0-9]{4})$',
        description: 'Fecha de inicio de la partida'
      },
      fechaTerminacion: {
        bsonType: 'string',
        pattern: '^([0-9]{2})+([0-9]{2})+([0-9]{4})$',
        description: 'Fecha de cierre de la partida'
      },
      horaInicio: {
        bsonType: 'string',
        description: 'Hora de inicio de la partida'
      },
    },
  },
}
```

```
horaFin: {  
  bsonType: 'string',  
  description: 'Hora de cierre de la partida'  
},  
estatus: {  
  'enum': [  
    'A',  
    'T'  
  ],  
  description: 'Estatus de la partida'  
},  
cupoMinimo: {  
  bsonType: 'int',  
  description: 'cupo minimo de la partida'  
},  
cupoMaximo: {  
  bsonType: 'int',  
  description: 'cupo maximo de la partida'  
}  
}  
}
```

2.4 Esquema de validación de la colección compra:

```
{  
  $jsonSchema: {  
    required: [  
      'numTarjeta',  
      'cvc',  
      'anioExpiracion',  
      'mesExpiracion',  
      'subtotal',  
    ],  
  },  
}
```

```
'total',
'fechaCompra',
'idUsuario'
],
properties: {
  numTarjeta: {
    bsonType: 'string',
    pattern: '^[0-9]{16}$',
    description: 'numero de tarjeta '
  },
  cvc: {
    bsonType: 'string',
    pattern: '^[0-9]{3}$',
    description: 'tres numeros de la parte trasera de la tarjeta'
  },
  anioExpiracion: {
    bsonType: 'int',
    minimum: 2024,
    maximum: 3000,
    description: 'anio de expiracion de la tarjeta'
  },
  mesExpiracion: {
    bsonType: 'int',
    minimum: 1,
    maximum: 12,
    description: 'mes de expiracion de la tarjeta'
  },
  subtotal: {
    bsonType: 'double',
    description: 'subtotal a pagar o pagado'
  },
  total: {
    bsonType: 'double',
```

```
description: 'total a pagar o pagado'
},
fechaCompra: {
  bsonType: 'string',
  description: 'fecha en la que fue hecha la compra'
},
idUserario: {
  bsonType: 'int',
  description: 'usuario por quien fue hecha la compra'
}
}
}
}
```

3. Documentación de los servicios

3.1 Servicio REST usuarios

Descripción:

Este servicio tiene como función principal realizar operaciones propias de un CRUD (Create, Read, Update, Delete), ya que se requieren hacer uso de estas para un buen manejo de la BD; asimismo se busca una autenticación de credenciales para poder acceder a las demás funcionalidades del juego.

Tipo:

Este servicio como tal es de entidad y tarea al poder hacer uso de las operaciones que ya anteriormente se mencionan las cuales son de un CRUD de los documentos de la entidad de usuarios.

Operaciones:

Como principales operaciones que se requieren del servicio son:

- ❖ Autenticar usuario
- ❖ Crear usuario
- ❖ Actualizar datos
- ❖ Eliminar usuario

3.1.1 Operación: Autenticación de usuario

Esta operación es la encargada de verificar que las credenciales que proporcione el usuario sean validadas y acceder a las demás acciones del juego, como lo son unirse a una partida, compra de personajes, consulta de personajes, consulta de almacén.

| Elemento | Valor |
|------------------|---|
| Método de acceso | GET |
| Ruta | /usuarios/autenticar |
| Protocolo | HTTP |
| Actor(es): | Usuario |
| Entrada: | <pre>{ "correo":"string", "contrasena":"string" }</pre> |

| | |
|----------|--|
| Proceso: | Verificar credenciales del usuario |
| Salida: | <pre>{ "Estatus":"string", "Message":"string", "nombre_usuario":"string", "Usuario":{ "nombre":"string", "correo":"string", "estatus":"string" } }</pre> |

3.1.2 Operación: Crear usuario

Esta operación de la creación de un nuevo usuario, para lo cual se requieren datos obligatorios para validar datos y que sea apto para poder ser parte del juego, autenticarse para poder acceder a las demás funciones que ofrece el juego.

| Elemento | Valor |
|------------------|---|
| Método de acceso | POST |
| Ruta | /usuarios/crear |
| Protocolo | HTTP |
| Actor(es): | Usuario |
| Entrada: | <pre>{ "nombre":"string", "correo":"string", "contrasena":"string" }</pre> |
| Proceso: | <p>Acción:</p> <p>Registrar un usuario en la base de datos, para realizar una inserción esto cuando los datos necesarios son llenados, esto a su vez genera datos de manera automática los cuales son:</p> <ul style="list-style-type: none"> ❖ Un identificador de usuario |

| | |
|---------|--|
| | <ul style="list-style-type: none"> ❖ Crear un campo llamado estatus y le asigna un valor por default una "A" de activo. <p>Validaciones:</p> <ul style="list-style-type: none"> ❖ Que el correo exista ❖ Que la contraseña sea texto |
| Salida: | <pre>{ "Estatus":"string", "Message":"string" }</pre> |

3.1.3 Operación: Actualizar datos

Para esta operación el usuario debe proporcionar los datos correspondientes para poder hacer el cambio solicitado por el usuario.

| Elemento | Valor |
|------------------|--|
| Método de acceso | PUSH |
| Ruta | /usuarios/actualizar/{idUserario} |
| Protocolo | HTTP |
| Actor(es): | Usuario |
| Entrada: | <pre>{ "nombre":"string", "contrasena":"string" }</pre> |
| Proceso: | Actualizar los datos del usuario |
| Salida: | <pre>{ "Estatus":"string", "Message":"string", "nombre_usuario":"string", "Usuario":{ "nombre":"string", "correo":"string", "estatus":"string" } }</pre> |

| | |
|--|---|
| | } |
|--|---|

3.1.4 Operación: Eliminar usuario

Para esta operación el administrador debe ser el encargado de hacer la eliminación, la operación se usa solamente cuando sea estrictamente necesario.

| Elemento | Valor |
|------------------|---|
| Método de acceso | DELETE |
| Ruta | /usuarios/eliminar/{idUserio} |
| Protocolo | HTTP |
| Actor(es): | Administrador |
| Entrada: | N/A |
| Proceso: | Actualizar el estatus del usuario a uno inactivo ("I"). |
| Salida: | { "Estatus":"string" "Message":"string" } |

3.1.5 Operación: Consultar usuarios

| Elemento | Valor |
|------------------|---|
| Método de acceso | GET |
| Ruta | /usuarios/consultar |
| Protocolo | HTTP |
| Actor(es): | Usuario |
| Entrada: | N/A |
| Proceso: | Consultar los usuarios |
| Salida: | { "Estatus":"string", "Message":"string", "nombre_usuario":"string", "Usuarios":[{ "nombre":"string", "correo":"string", }] } |

| | |
|--|---|
| | <pre>"estatus":"string", "ID_Usuario":"string" }] }</pre> |
|--|---|

3.1.6 Operación: Consultar usuario por identificador

| Elemento | Valor |
|------------------|---|
| Método de acceso | GET |
| Ruta | /usuarios/consultar/{idUserio} |
| Protocolo | HTTP |
| Actor(es): | Usuario |
| Entrada: | El id del usuario en la cadena de consulta |
| Proceso: | Consultar un usuario en específico. |
| Salida: | <pre>{ "nombre":"string", "correo":"string", "estatus":"string" }</pre> |

3.2 Servicio REST compra

Descripción: Este servicio que al igual que el servicio anterior se tiene operaciones propias de un CRUD para lo cual es fundamental ya que las compras al igual que la autenticación de usuarios son las partes importantes, de igual manera se requiere que las compras tengan como finalidad un registro de los personajes, usuarios y tarjetas con las que fué pagada la compra.

Tipo:

El servicio es de tipo tarea y entidad ya que como anteriormente se menciona se usa para crear actualizar, leer y eliminar de manera adecuada los documentos de la entidad de compra.

Operaciones:

Entre las principales funciones que requiere esta entidad para tener un correcto funcionamiento, destacan:

- ❖ Realizar compra
- ❖ Consultar compras
- ❖ Consultar compra por identificador

3.2.1 Operación: Realizar compra

Esta operación agrega un personaje a la compra antes de ser pagada.

| Elemento | Valor |
|------------------|--|
| Método de acceso | POST |
| Ruta | /compras/realizar_compra |
| Protocolo | HTTP |
| Actor(es): | Usuario |
| Entrada: | <pre>{ "numTarjeta":"string", "cvc":"string", "anioExpiracion":"integer", "mesExpiracion":"integer", "subtotal":"float", "total":"float", "fechaCompra":"datetime", "idUsuario":"str", "detalleCompra":[{ "idPersonaje":"str", "precio":"float" }] }</pre> |
| Proceso: | <p>Acción:</p> <p>Agrega un registro a la base de datos, para generar una compra correcta y exitosa debe crear de manera automática los siguientes datos:</p> <ul style="list-style-type: none"> ❖ Identificador único de la compra |

| | |
|---------|---|
| Salida: | <pre> { "Estatus": "string", "Message": "string", "Compra": { "numTarjeta": "string", "cvc": "string", "anioExpiracion": "integer", "mesExpiracion": "integer", "subtotal": "float", "total": "float", "fechaCompra": "datetime", "idUserario": "string", "detalleCompra": [{ "idPersonaje": "string", "precio": "float" }] } }</pre> |
|---------|---|

3.2.2 Operación: Consultar compras

Esta operación hace una consulta general de las compras que ha realizado el usuario a lo largo de su estadía en el juego.

| Elemento | Valor |
|------------------|--|
| Método de acceso | GET |
| Ruta | /compras/consultar |
| Protocolo | HTTP |
| Actor(es): | Usuario |
| Entrada: | N/A |
| Proceso: | Consultar a la base de datos las compras que haya hecho el usuario y poder mostrarlos. |

| | |
|---------|--|
| Salida: | <pre>{ "Estatus": "string", "Message": "string", "Compras": [{ "_id": "string" "numTarjeta": "string", "cvc": "string", "anioExpiracion": "integer", "mesExpiracion": "integer", "subtotal": "float", "total": "float", "fechaCompra": "datetime", "idUsuario": "string", "detalleCompra": [{ "idPersonaje": "string", "precio": "float", "nombrePersonaje": "string" }], "usuario": "string" }] }</pre> |
|---------|--|

3.2.3 Operación: Consultar compra por identificador

Esta operación busca mediante el identificador de compra los datos de la misma y mostrarlos al usuario.

| Elemento | Valor |
|------------------|-------------------------------|
| Método de acceso | GET |
| Ruta | /compras/consultar/{idCompra} |
| Protocolo | HTTP |
| Actor(es): | Usuario |

| | |
|----------|---|
| Entrada: | idCompra (string) – En la cadena de consulta |
| Proceso: | Consultar a la base de datos la compra que haya hecho el usuario y poder mostrarlos. |
| Salida: | <pre>{ "_id": "string" "numTarjeta": "string", "cvc": "string", "anioExpiracion": "integer", "mesExpiracion": "integer", "subtotal": "float", "total": "float", "fechaCompra": "datetime", "idUsuario": "string", "detalleCompra": [{ "idPersonaje": "string", "precio": "float", "nombrePersonaje": "string" }], "usuario": "string" }</pre> |

3.3 Servicio REST personajes

Descripción:

Este servicio requiere que los personajes se puedan crear, actualizar y buscar el eliminar no se incluye ya que los personajes que tenga el usuario serán de manera permanente; dependiendo de lo que se requiera es la acción que se ejecutará, en este caso todo esto almacenado en la base de datos de SurviveGame.

Tipo:

Este servicio es catalogado como de entidad ya que no requiere de más que de casi todas las funciones un sencillo CRUD para poder ser utilizado.

Operaciones:

De las principales operaciones que tiene como este servicio se encuentran las siguientes:

- ❖ Consultar personajes
- ❖ Consultar personaje por identificador
- ❖ Agregar personaje
- ❖ Actualizar personaje

3.3.1 Operación: consultar personajes

| Elemento | Valor |
|------------------|--|
| Método de acceso | GET |
| Ruta | /personajes/consultar |
| Actor(es): | Usuario, Administrador |
| Entrada: | N/A |
| Proceso: | Consultar a la base de datos los personajes que existen y poder mostrarlos |
| Salida: | <pre>{ "personajes": [{ "_id": "ObjectId", "nombre": "string", "precio": "float", "imagen": "string", "estatus": "string" }] }</pre> |

3.3.2 Operación: consultar personaje por identificador

| Elemento | Valor |
|------------------|-------------------------------------|
| Método de acceso | GET |
| Ruta | /personajes/consultar/{idPersonaje} |

| | |
|------------|---|
| Protocolo | HTTP |
| Actor(es): | Usuario, Administrador |
| Entrada: | idPersonaje (string) – En la cadena de la consulta |
| Proceso: | Consultar a la base de datos los personajes que existen y poder mostrarlos |
| Salida: | <pre>{ "_id":"string", "nombre":"string", "precio":"float", "imagen":" string ", "estatus":"string" }</pre> |

3.3.3 Operación: crear personaje

| Elemento | Valor |
|------------------|--|
| Método de acceso | POST |
| Ruta | /personajes/agregar |
| Protocolo | HTTP |
| Actor(es): | Administrador |
| Entrada: | <pre>{ "nombre":"string", "precio":"float", "imagen":"string", "estatus":"string" }</pre> |
| Proceso: | <p>Acción:</p> <p>Registrar un personaje en la base de datos, para realizar una inserción esto cuando los datos necesarios son llenados, esto a su vez genera datos de manera automática los cuales son:</p> <ul style="list-style-type: none"> ❖ Un identificador de personaje ❖ Un estatus de Activo “A” para denotar que esta activo y pueda se visible para los usuarios. |

| | |
|---------|---|
| Salida: | { "Estatus": "string", "Message": " string " } |
|---------|---|

3.3.4 Operación: actualizar personaje

| Elemento | Valor |
|------------------|--|
| Método de acceso | PUT |
| Ruta | /personajes/actualizar/{idPersonaje} |
| Protocolo | HTTP |
| Actor(es): | Administrador |
| Entrada: | { "nombre": "string", "precio": "float", "imagen": " string " } |
| Proceso: | Acción: Actualizar un personaje en la base de datos, para realizar una modificación esto cuando los datos necesarios son llenados. |
| Salida: | { "Estatus": "string", "Message": "string", "Personaje": { "nombre": "string", "precio": "float", "imagen": "string", "estatus": "string" } } |

3.3.5 Operación: eliminar personaje

| Elemento | Valor |
|------------------|---|
| Método de acceso | DELETE |
| Ruta | /personajes/eliminar/{idPersonaje} |
| Protocolo | HTTP |
| Actor(es): | Administrador |
| Entrada: | El identificador del personaje en la cadena de texto, este identificador debe ser en texto hexadecimal. |
| Proceso: | Acción: Actualiza un personaje con el estatus de Inactivo "I" para que este ya no pueda ser comprado, pero si pueda ser utilizado a los usuarios que lo hayan comprado antes de ser retirado. |
| Salida: | <pre>{ "Estatus":"string", "Message":" string " }</pre> |

3.4 Servicio REST partida

Descripción:

Este servicio tiene como principal propósito la creación, consulta y actualización de las partidas del usuario.

Tipo:

Este servicio es catalogado como uno de entidad ya que proporciona datos de consulta y asimismo se puede crear y actualizar la partida, estos atributos son propios de un CRUD.

Operaciones:

De las operaciones destacables de la partida son:

- ❖ Iniciar partida
- ❖ Finalizar partida
- ❖ Consultar partidas

❖ Consultar partida por identificador

3.4.1 Operación: Crear partida

Esta operación para iniciar partida, los usuarios se pueden unir a la partida y ellos van a iniciar la partida.

| Elemento | Valor |
|------------------|--|
| Método de acceso | POST |
| Ruta | /partida/iniciarPartida |
| Actor(es): | Usuario |
| Entrada: | <pre>{ "duracion":"integer", "fechalnicio":"str(date)", "fechaTerminacion":"str(date)", "estatus":"string", "cupoMinimo":"integer", "cupoMaximo":"integer", "participantes":[{ "estatus":"string", "ganador":"bool", "usuario:{ "idUserario":"string", "idPersonaje":"string" } }] }</pre> |
| Proceso: | <p>Acción:</p> <p>Iniciar la partida, una inserción en la base de datos, con los datos correspondientes, así mismo cuando se crea la partida envía como valor a fechalnicio la fecha actual y horalnicio la hora actual de igual manera fechaTerminacion y horaFin se establecen con la fecha y hora actual, ya cuando se quiere</p> |

| | |
|---------|--|
| | <p>finalizar la partida se cambian los valores de fechaTerminacion y horaFin, de igual manera la duración es 0, de igual manera cada participante que su estatus cambiara a "P" denotando que esta jugando.</p> <p>Validaciones:</p> <ul style="list-style-type: none"> ❖ Que el campo ganador falso para cada participante ❖ Que el campo de estatus al iniciar la partida sea "A" |
| Salida: | <pre>{ "Estatus": "string", "Message": " string " }</pre> |

3.4.2 Operación: Finalizar partida

Esta operación para finalizar la partida, la partida terminará hasta que haya un usuario ganador en la partida.

| Elemento | Valor |
|------------------|---|
| Método de acceso | PUT |
| Ruta | /partida/{idPartida}/finalizarPartida |
| Actor(es): | Usuario |
| Entrada: | <pre>{ "idGanador": "string" }</pre> |
| Proceso: | <p>Acción:</p> <p>Finalizar la partida al momento que un usuario se indique como ganador (que sea true), la hora de finalización (horaFin) se actualice a la hora actual, asimismo la fechaTerminación que sea la actual.</p> <p>El estatus de la partida al momento de finalizar cambiarlo a "T".</p> <p>Validaciones:</p> <ul style="list-style-type: none"> ❖ Que el estatus de los jugadores cambie a uno de activo ❖ Que solo exista un valor de verdadero para que solo haya un solo ganador. |

| | |
|---------|---|
| Salida: | { "Estatus": "string", "Message": " string " } |
|---------|---|

3.4.3 Operación: Consultar partidas

| Elemento | Valor |
|------------------|--|
| Método de acceso | GET |
| Ruta | /partida |
| Protocolo | HTTP |
| Actor(es): | Usuario |
| Entrada: | N/A |
| Proceso: | Consultar a la base de datos las partidas. |
| Salida: | { "Estatus": "string", "Message": "string", "partidas": [{ "_id": "string", "duracion": "float", "fechaInicio": "str(date)", "fechaTerminacion": "str(date)", "horaInicio": "str(time)", "horaFin": "str(time)", "estatus": "str", "cupoMinimo": "integer", "cupoMaximo": "integer", "participantes": [{ "estatus": "string", "ganador": false, "usuario": { |

| | |
|--|---|
| | <pre> "idUsuario": "string", "idPersonaje": "string", "nombreUsuario": "string", "nombrePersonaje": "string" } }] }] } </pre> |
|--|---|

3.4.4 Operación: Consultar partida por identificador

| Elemento | Valor |
|------------------|---|
| Método de acceso | GET |
| Ruta | /partida/{idPartida} |
| Protocolo | HTTP |
| Actor(es): | Usuario |
| Entrada: | idPartida (string) – En la cadena de consulta |
| Proceso: | Consultar a la base de datos la partida específica que ha tenido el usuario a detalle. |
| Salida: | <pre> { "Estatus": "string", "Message": "string", "partidas": { "_id": "string", "duracion": "float", "fechaInicio": "str(date)", "fechaTerminacion": "str(date)", "horaInicio": "str(time)", "horaFin": "str(time)", "estatus": "str", "cupoMinimo": "integer", </pre> |

| | |
|--|--|
| | <pre> "cupoMaximo": "integer", "participantes": [{ "estatus": "string", "ganador": false, "usuario": { "idUserario": "string", "idPersonaje": "string", "nombreUsuario": "string", "nombrePersonaje": "string" } }] } </pre> |
|--|--|

3.4.5 Operación: Agregar participante

| Elemento | Valor |
|------------------|--|
| Método de acceso | PUT |
| Ruta | /partida/agregarParticipante/{idPartida} |
| Actor(es): | Usuario |
| Entrada: | <pre> { "estatus": "string", "ganador": "boolean", "usuario": { "idUserario": "string", "idPersonaje": "string" } } </pre> |
| Proceso: | <p>Acción:</p> <p>Agregar a la partida un participante nuevo para esto se requiere</p> |

| | |
|---------|---|
| | <p>que cuando se actualice el estatus del usuario el cual se pone en "P" (playing) para denotar que está en partida.</p> <p>Validaciones:</p> <ul style="list-style-type: none"> ❖ Que el usuario en el campo ganador sea false ❖ Que el estatus del jugador siga siendo false al momento de determinar un ganador. ❖ Que solo exista un valor de verdadero para que solo haya un solo ganador. ❖ Verificar que el personaje si sea del usuario ❖ Agregar un nuevo objeto a participantes |
| Salida: | <pre>{ "Estatus": "string", "Message": "string" }</pre> |