PROYECTO - DOCUMENTACIÓN GENERAL

# MICROSERVICIOS

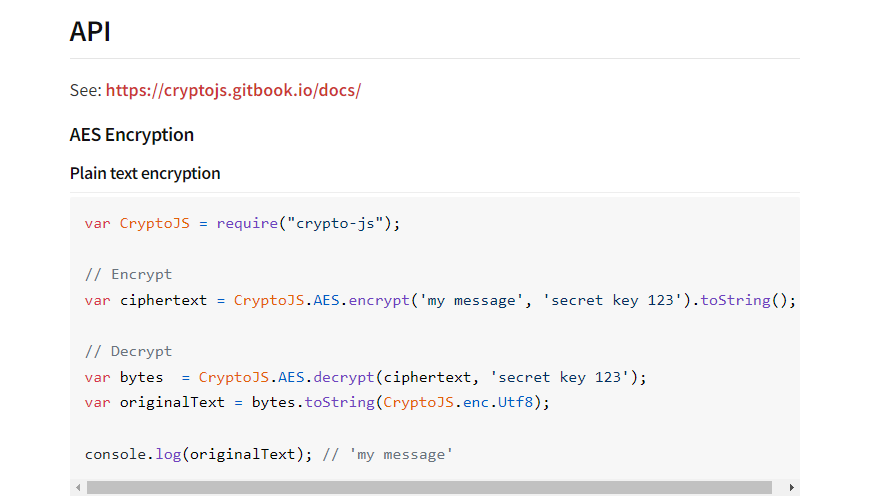
* Clientes: Jesús y Fredy
* Servicios\_Administrativos(Antes llamado Empleados): Alison - Cristian (Estadio y Partido van dentro de este microservicio)
  + Noticias : Elba y Mariana
* Administrador: Christopher - Pablo (Persona, Reportes )
* Usuario: Erick, Gonzalo
* Predicción:

Consideraciones:

* Para la tabla **Country**, manejar esta estructura y utilizar estos inserts -> <https://gist.github.com/adhipg/1600028>
* Para la respuesta de la api considerar esta estructura (Esto es para facilitar el trabajo y tener entendido que en todos los endpoints recibiremos esta estructura):

| {  status : number, // 200 = ok 400 = wrong  msj : string,  data : cualquier objeto [], {}, string, etc  } |
| --- |

Encriptacion:



-

# ENDPOINTS

* Endpoints en Inglés

# PUERTOS

* Clientes:5000
* Servicios\_Administrativos: 5001
* Administrador: 5002
* Usuario: 5003
* Predicción: 5004

**Headers:**

{

authorization: Bearer <Token>

}

**Guardan en Token JWT**

{

id\_usuario: 2,

id\_rol: 1

}

Roles:

1: admin

2: Empleado

3: Cliente

## CLIENTES

**ObtenerMembresia**

**Endpoint (patch) http://ip:5000/esb/client/membership**

**IP:5000/esb/client/membership**

**Enviar**

**Entrada**:

{

id\_client:number

}

**Salida**:

{

status : number, // 200 = ok 400 = wrong

msj : "Ahora cuenta con una membresia" / "Error al obtener membresia ",

data : cualquier objeto [], {}, string, etc

}

**Restablecer contraseña**

**Endpoint (GET)**  http://ip:5000/esb/client/reset

**IP:5000/esb/client/reset**

**Enviar**

**Entrada**:

{

id\_c: number

new\_pass : string

}

**Salida**:

{

status : number, // 200 = ok 400 = wrong

msj : "Se ha enviado un correo para reestablecer la clave" / "Error al reestablecer la clave",

data : cualquier objeto [], {}, string, etc

}

**Seguir equipo**

**Endpoint (POST)**  http://ip:5000/esb/client/follow/

**IP:**5000/esb/client/follow

ENVIAR

{

id\_client: number \*\* cambio de id\_cliente -> id\_client

id\_team: number

}

SALIDA

{

status : number, // 200 = ok 400 = wrong

msj : "Error al seguir a un equipo",

data : cualquier objeto [], {}, string, etc

}

**Ver notificaciones**

**Endpoint (GET)**  http://ip:5000/esb/client/notifications

**IP:**5000/esb/client/notifications

QUERY PARAM \*\*id\*\*

Ejemplo : http://ip:5000/esb/team/?id=2

SALIDA

{

status : number, // 200 = ok 400 = wrong

msj : "Se envia las notificaciones",

data : [

{

id\_news :number,

id\_team : number,

name\_team : string

}

]

}

**Unirse a quiniela**

**Endpoint (POST)**  http://ip:5000//esb/client/quiniela

**IP:**5000//esb/client/quiniela

**Entrada**:

{

id\_client : number,

id\_game : number,

result\_local : number,

result\_visiting: number

}

**Salida**:

{

status : number, // 200 = ok 400 = wrong

msj : "Estado de la quiniela actualizado" / "Error al actualizar el estado de la persona",

data : cualquier objeto [], {}, string, etc

}

**# CONSULTAS Y ESTADISTICAS POR CLIENTE**

\*\*URL : http://ip:5000/esb/client/reports\*\*

En los endpoints solo se especificará lo enviado en \*\*data\*\* y/o en \*\*msj\*\*

Formato de Salida :

~~~

{

status : number, // 200 = ok 500 = wrong

msj : string,

data : cualquier objeto [], {}, string, etc

}

~~~

---

**## Jugadores o Técnico de X equipo**

\*\*GET : http://ip:5000/esb/client/reports/person/\*\*

QUERY PARAM \*\*equipo\*\*

##### Ej : http://ip:5000/esb/client/reports/person/?equipo=2

Salida :

~~~

data : [

{

id\_person : number,

name : string,

lastname : string,

photo : string,

id\_team : number,

name\_team : string

}

]

~~~

**## Jugadores o Técnico mayores a X años**

\*\*GET : http://ip:5000/esb/client/reports/person/higher\*\*

QUERY PARAM \*\*edad\*\*

##### Ej : http://ip:5000/esb/client/reports/person/higher/?edad=2

Salida :

~~~

data : [

{

id\_person : number,

name : string,

lastname : string,

photo : string

age : number

}

]

~~~

**## Jugadores o Técnico menores a X años**

\*\*GET : http://ip:5000/esb/client/reports/person/lower\*\*

QUERY PARAM \*\*edad\*\*

##### Ej : http://ip:5000/esb/client/reports/person/lower/?edad=2

Salida :

~~~

data : [

{

id\_person : number,

name : string,

lastname : string,

photo : string

age : number

}

]

~~~

**## Equipos que participaron en X competición**

\*\*GET : http://ip:5000/esb/client/reports/competition/team/\*\*

QUERY PARAM \*\*competicion\*\*

##### Ej : http://ip:5000/esb/client/reports/competition/team/?competicion=2

Salida :

~~~

data : [

{

id\_competition : number,

name\_competition : string,

id\_team : number,

name\_team : string

}

]

~~~

**## Equipos de X país**

\*\*GET : http://ip:5000/esb/client/reports/country/team/\*\*

QUERY PARAM \*\*pais\*\*

##### Ej : http://ip:5000/esb/client/reports/country/team/?pais=2

Salida :

~~~

data : [

{

id\_country : number,

name\_country : string,

id\_team : number,

name\_team : string

}

]

~~~

**## Equipos X años de antiguadad**

\*\*GET : http://ip:5000/esb/client/reports/team/age\*\*

QUERY PARAM \*\*edad\*\*

##### Ej : http://ip:5000/esb/client/reports/country/team/age/?edad=2

Salida :

~~~

data : [

{

id\_team : number,

name\_team : string,

age : number

}

]

~~~

**## Estadios en X país**

\*\*GET : http://ip:5000/esb/client/reports/country/stadium/\*\*

QUERY PARAM \*\*pais\*\*

##### Ej : http://ip:5000/esb/client/reports/country/stadium/?pais=2

Salida :

~~~

data : [

{

id\_stadium : number,

name\_stadium : string,

id\_country : number,

name\_country : string

}

]

~~~

**## Estadios con capacidad menor o igual a X**

\*\*GET : http://ip:5000/esb/client/reports/stadium/capacity\*\*

QUERY PARAM \*\*capacidad\*\*

##### Ej : http://ip:5000/esb/client/reports/stadium/capacity/?capacidad=2

Salida :

~~~

data : [

{

id\_stadium : number,

name\_stadium : string,

capacidad : number

}

]

~~~

**## Histórico de partidos de X equipo**

\*\*GET : http://ip:5000/esb/client/reports/team/game/\*\*

QUERY PARAM \*\*equipo\*\*

##### Ej : http://ip:5000/esb/client/reports/team/game/?equipo=2

Salida :

~~~

data : [

{

id\_team : number,

name\_team : string,

id\_game : number

date\_game : date,

id\_stadium : number,

name\_stadium : string

viewers : number,

winner : string,

result : string

}

]

~~~

**## Equipos en los que ha estado o dirigido X técnico o jugador**

\*\*GET : http://ip:5000/esb/client/reports/team/person/\*\*

QUERY PARAM \*\*persona\*\*

##### Ej : http://ip:5000/esb/client/reports/team/person/?persona=2

Salida :

~~~

data : [

{

id\_team : number,

name\_team : string,

id\_person : number,

name\_person : string

}

]

~~~

**## Partidos donde hubo al menos X goles**

\*\*GET : http://ip:5000/esb/client/reports/game/goal/\*\*

Salida :

~~~

data : [

{

id\_game : number,

name\_game : string,

goals : number

}

]

~~~

**## Jugadores con más X incidencias en Y competición, (de Z año)**

\*\*GET : http://ip:5000/esb/client/reports/person/competition/incidents\*\*

QUERY PARAM \*\*competicion\*\*

QUERY PARAM \*\*incidente\*\*

QUERY PARAM \*\*anio\*\*

##### Ej : http://ip:5000/esb/client/reports/person/competition/incidents/?competicion=1&?incidente=2&anio=2011

Salida :

~~~

data : [

{

id\_game : number,

name\_game : string,

total\_incidents : number,

name\_incidents :string

id\_competition : number,

name\_competition : string,

id\_person : number,

name\_person : string,

year: string

}

]

~~~

**## Cantidad de X competiciones que ha ganado Y equipo**

\*\*GET : http://ip:5000/esb/client/reports/team/competitions/\*\*

QUERY PARAM \*\*equipo\*\*

##### Ej : http://ip:5000/esb/client/reports/team/competitions?equipo=2

Salida :

~~~

data : [

{

id\_team : number,

name\_team : string,

total\_competitions : number

}

]

~~~

**## Listado de partidos en X año**

\*\*GET : http://ip:5000/esb/client/reports/games/year/\*\*

QUERY PARAM \*\*anio\*\*

##### Ej : http://ip:5000/esb/client/reports/games/year?anio=2

Salida :

~~~

data : [

{

id\_game : number,

name\_game : string,

year : number

}

]

~~~

## Listado de partidos entre X equipo contra Y equipo

\*\*GET : http://ip:5000/esb/client/reports/games/teams/\*\*

QUERY PARAM \*\*local\*\*

QUERY PARAM \*\*visitante\*\*

##### Ej : http://ip:5000/esb/client/reports/games/teams?local=2&?=visitante=3

Salida :

~~~

data : [

{

id\_game : number,

name\_game : string,

id\_team\_local : number,

team\_local : string,

id\_team\_visit : number,

team\_visit : string,

result\_local : number,

result\_visiting:number,

date\_game : date

}

]

## EMPLEADOS

## CRUD Estadio

Crear estadio -> POST : ip/stadium

| Entrada | Salida |
| --- | --- |
| {  name : string,  fundation\_date : date,  capacity: number,  id\_country : number,  address: string,  state: string,  photo : string //base64  } | {  status : number, // 200 = ok 500 = wrong,  msj : "Estadio creado con éxito" / "Error al crear el estadio",  data : [] // Se podría devolver la información del estadio creado o no devolver nada **(?)**  } |

Actualizar estadio -> PUT : ip/stadium

| Entrada | Salida |
| --- | --- |
| {  id : number,  name : string,  fundation\_date : date,  capacity: number,  id\_country : number,  address: string,  state: string,  photo : string //base64  } | {  status : number, // 200 = ok 500 = wrong,  msj : "Estadio actualizado" / "Error al actualizar estadio",  data : [] // Se podría devolver la información del estadio actualizado o no devolver nada **(?)**  } |

Ver estadio -> GET : ip/stadium?id

| QUERY PARAM | Salida |
| --- | --- |
| ip/stadium?id  Ej : ip/stadium/?id=2   * Si viene el query param se manda la información del estadio correspondiente. * Sino, se mandan la información de todos los estadios. | **Información de todos los estadios**  {  status : number, // 200 = ok 500 = wrong,  msj : "Información de los estadios" / "Error al obtener los estadios",  data : [  {  id : number,  name : string,  fundation\_date : date,  capacity: number,  id\_country : number,  country : string,  address: string,  state: string,  photo : string //base64  },  {  id : number,  name : string,  fundation\_date : date,  capacity: number,  id\_country : number,  address: string,  state: string,  photo : string //base64  },  …  ]  }  **Un estadio**  {  status : number, // 200 = ok 500 = wrong,  msj : "Información del estadio" / "Error al obtener el estadio",  data : {  id : number,  name : string,  fundation\_date : date,  capacity: number,  id\_country : number,  address: string,  state: string,  photo : string //base64  }  } |
|  |  |

Eliminar estadio -> DELETE : ip/stadium/?id

| PATH PARAM | Salida |
| --- | --- |
| ip/stadium/:id  Ej : ip/stadium/2 | {  status : number, // 200 = ok 500 = wrong,  msj : "Estadio eliminado" / "Error al eliminar el estadio",  data : []  } |

## CRUD Equipo

Crear equipo -> POST : ip/team

| Entrada | Salida |
| --- | --- |
| {  name : string,  fundation\_date : date,  id\_country : number  photo : string //base64  } | {  status : number, // 200 = ok 500 = wrong,  msj : "Equipo creado con éxito" / "Error al crear equipo",  data : [] // Se podría devolver la información del equipo creado o no devolver nada **(?)**  } |

Actualizar equipo -> PUT : ip/team

| Entrada | Salida |
| --- | --- |
| {  id : number,  name : string,  fundation\_date : date,  id\_country : number,  photo : string //base64  } | {  status : number, // 200 = ok 500 = wrong,  msj : "Equipo actualizado" / "Error al actualizar equipo",  data : [] // Se podría devolver la información del equipo actualizado o no devolver nada **(?)**  } |

Ver equipo -> GET : ip/team?id

| QUERY PARAM | Salida |
| --- | --- |
| ip/team?id  Ej : ip/team/?id=2   * Si viene el query param se manda la información del equipo correspondiente. * Sino, se mandan la información de todos los equipos. | **Información de todos los equipos**  {  status : number, // 200 = ok 500 = wrong,  msj : "Información de los equipos" / "Error al obtener los equipos",  data : [  {  id : number,  name : string,  fundation\_date : date,  id\_country : number,  country :string,  photo : string  },  {  id : number,  name : string,  fundation\_date : date,  id\_country : number,  country :string,  photo : string  },  …  ]  }  **Un equipo**  {  status : number, // 200 = ok 500 = wrong,  msj : "Información de los equipos" / "Error al obtener los equipos",  data : {  id : number,  name : string,  fundation\_date : date,  id\_country : number,  country :string,  photo : string  }  } |

Eliminar equipo -> DELETE : ip/team/?id

| PATH PARAM | Salida |
| --- | --- |
| ip/team/:id  Ej : ip/team/2 | {  status : number, // 200 = ok 500 = wrong,  msj : "Equipo eliminado" / "Error al eliminar el equipo",  data : []  } |

## CRUD PARTIDO

Crear partido -> POST : ip/soccer-game

| Entrada | Salida |
| --- | --- |
| {  game\_date : date,  attendees : number,  result\_local: number,  result\_visiting: number,  state: string,  id\_stadium : number,  id\_team\_local : number,  id\_team\_visiting : number,  } | {  status : number, // 200 = ok 500 = wrong,  msj : "Partido guardado con éxito" / "Error al guardar el partido",  data : [] // Se podría devolver la información de la partida creada o no devolver nada **(?)**  } |

Actualizar partido -> PUT : ip/soccer-game

| Entrada | Salida |
| --- | --- |
| {  game\_date : date,  attendees : number,  result\_local: number,  result\_visiting: number,  id\_stadium : number,  id\_team\_local : number,  id\_team\_visiting : number,  } | {  status : number, // 200 = ok 500 = wrong,  msj : "Partido actualizado" / "Error al actualizar partido",  data : [] // Se podría devolver la información del partido actualizado o no devolver nada **(?)**  } |

Ver partido -> GET : ip/soccer-game?id

| Entrada | Salida |
| --- | --- |
| ip/soccer-game?id  Ej : ip/soccer-game/?id=2   * Si viene el query param se manda la información del partido correspondiente. * Sino, se mandan la información de todos los partidos. | **Información de todos los partidos**  {  status : number, // 200 = ok 500 = wrong,  msj : "Información de los partidos" / "Error al obtener los partidos",  data : [  {  game\_date : date,  attendees : number,  result\_local: number,  result\_visiting: number,  id\_stadium : number,  id\_team\_local : number,  team\_local : string  id\_team\_visiting : number,  team\_visiting : string  },  {  game\_date : date,  attendees : number,  result\_local: number,  result\_visiting: number,  id\_stadium : number,  id\_team\_local : number,  team\_local : string  id\_team\_visiting : number,  team\_visiting : string  },  …  ]  }  **Un partido**  {  status : number, // 200 = ok 500 = wrong,  msj : "Información de del partido" / "Error al obtener el partido",  data : {  game\_date : date,  attendees : number,  result\_local: number,  result\_visiting: number,  id\_stadium : number,  id\_team\_local : number,  team\_local : string  id\_team\_visiting : number,  team\_visiting : string  }  } |

Eliminar partido -> DELETE : ip/soccer-game/:id

| PATH PARAM | Salida |
| --- | --- |
| ip/soccer-game/:id  Ej : ip/soccer-game/2 | {  status : number, // 200 = ok 500 = wrong,  msj : "Partido eliminado" / "Error al eliminar el partido",  data : []  } |

## CRUD Competencia

### Obtener los datos

#### GET -> [http://ip:5000/esb/](http://ip:5000/esb/competencia)competition?championship=id

| Entrada | Salida |
| --- | --- |
| * Si viene championship en query param se manda la información de la competencia correspondiente. * Sino, se mandan la información de todas las competencias. | **Información de todas las competencias**  {  status : number, // 200 = ok 500 = wrong,  msj : "Información de las competencias" / "Error al obtener las competencias",  data :[  { year : number, type : string, champion\_team : string, country : name },  { year : number, type : string, champion\_team : string, country : name },  …  ]  }  **Una competencia en específico**  {  status : number, // 200 = ok 500 = wrong,  msj : "Información de la competencia" / "Error al obtener la competencia",  data :[  { year : number, type : string, champion\_team : string, country : name },  { year : number, type : string, champion\_team : string, country : name }  ]  } |

### Crear una Liga

#### **Endpoint**(**post**): [http://ip:5000/esb/](http://ip:5000/esb/competencia)league

**Enviar**

{

name: string,

description: string,

type: number,

id\_country : number

}

**Respuesta**:

msj : { msj:”league was created successfully”, idLeague: number } / "Error al crear partido"

### 

### Crear una competencia

Se define el año de la competencia en el que se llevará a cabo la competencia y el campeón de ese torneo. Tomar en cuenta que debe de existir la tabla League

#### **Endpoint**(**post**): [http://ip:5000/esb/](http://ip:5000/esb/competencia)competition

**Enviar**

{

league: number,

year : number,

champion\_team : number,

}

**Respuesta**:

msj : { msj:”competition created successfully”, idChampionship: number } / "Error al crear competencia"

### Eliminar competencia

### **Endpoint**(**delete**): http://ip:5000/esb/competition/{id}

**Respuesta**:

msj : { msj:”Operation completed successfully”, idChampionship: number } / "Error al eliminar competencia"

### Actualizar competencia

### **Endpoint**(**put**): http://ip:5000/esb/competition/

**Enviar**

{

league: number,

year : number,

champion\_team : number,

}

**Respuesta**:

msj : { msj:”competition updated successfully”, idChampionship: number } / "Error al actualizar competencia"

## Acciones Empleado

1. Transferir un jugador de un equipo a otro, los jugadores deberán tener su propia bitácora de los equipos en los que han militado y de qué fecha a que fecha jugó para ese equipo.

POST : ip/transfer-player

| Entrada | Salida |
| --- | --- |
| {  id\_player: number,  id\_team\_origin : number,  id\_team\_destination : number,  } | {  status : number, // 200 = ok 500 = wrong,  msj : "Jugador actualizado" / "Error al transferir el jugador",  data : []  }   * Considerar que en este endpoint se debe guardar internamente la información en la **bitácora de transferencias**   {  id:number,  id\_player: number,  id\_team\_origin : number,  id\_team\_destination : number,  transfer\_date: date,  team\_origin\_date: date,  } |

De lo solicitado anteriormente surge el endpoint para obtener los datos de la bitácora de transferencias.

GET : ip/transfer-log

| Entrada | Salida |
| --- | --- |
|  | {  status : number, // 200 = ok 500 = wrong,  msj : "Información de la bitácora de transferencias" / "Error al obtener la información de la bitácora de transferencias",  data : [  {  id:number,  id\_player: number,  id\_team\_origin : number,  id\_team\_destination : number,  transfer\_date: date,  team\_origin\_date: date,  }  ]  } |

2) Transferir un técnico de un equipo a otro, los jugadores deberán tener su propia bitácora de los equipos en los que han militado y de qué fecha a que fecha jugó para ese equipo.

POST : ip/transfer-coach

| Entrada | Salida |
| --- | --- |
| {  id\_coach: number,  id\_team\_origin : number,  id\_team\_destination : number,  } | {  status : number, // 200 = ok 500 = wrong,  msj : "Jugador actualizado" / "Error al transferir el jugador",  data : []  }   * Considerar que en este endpoint se debe guardar internamente la información en la **bitácora de transferencias**   {  id:number,  id\_coach: number,  id\_team\_origin : number,  id\_team\_destination : number,  transfer\_date: date,  team\_origin\_date: date,  } |

De lo solicitado anteriormente surge el endpoint para obtener los datos de la bitácora de transferencias.

GET : ip/transfer-log-coach

| Entrada | Salida |
| --- | --- |
|  | {  status : number, // 200 = ok 500 = wrong,  msj : "Información de la bitácora de transferencias" / "Error al obtener la información de la bitácora de transferencias",  data : [  {  id:number,  name: string,  id\_team\_origin : number,  team\_origin: string,  id\_team\_destination : number,  team\_destinantion: string,  transfer\_date: date,  team\_origin\_date: date,  }  ]  } |

3) Podrán cambiar el estado de un partido (Sin iniciar, en curso, finalizado, suspendido)

* Se puede hacer uso del mismo endpoint de actualizar un partido (PUT -> ip/soccer-game), sabiendo que el atributo state solo puede tener los valores de : “unstarted”, “in-progress”, “completed” , “suspended”

4) Podrán agregar incidencias a un partido media vez se encuentre en curso

POST : ip/add-incidence

| Entrada | Salida |
| --- | --- |
| {  id\_game : number,  id\_person : number,  id\_team : number,  descripcion : string,  id\_incidencia : number,  minuto : number  } | {  status : number, // 200 = ok 500 = wrong,  msj : "Incidencia agregada con éxito" / "Error al intentar agregar incidencia",  } |

5) Cambiar el estado de un jugador o técnico

ENDPOINT ??????????????????????????????????????????????????????????????

| Entrada | Salida |
| --- | --- |
| {  id\_person: number,  state : number,  } | {  status : number, // 200 = ok 500 = wrong,  msj : "Estado de persona actualizado" / "Error al intentar actualizar el estado de la persona",  } |

6) Publicar una noticia acerca de un equipo.

## Noticias

Crear noticia -> POST : ip/notice

| Entrada | Salida |
| --- | --- |
| {  id\_team: number,  title: string,  description : string,  date: date } | {  status : number, // 200 = ok 500 = wrong,  msj : "Ha insertado una noticia con éxito" / "Error al guardar una noticia",  data : [] // Se podría devolver la información de la noticia o no devolver nada **(?)**  } |

Obtener la noticia -> GET : ip/notice

| Entrada | Salida |
| --- | --- |
| Query params:   * **id (idi de la noticia)** * **team (id del team)**   **Ejemplo:**  ip/notice?team=1  ip/notice?id=10  Explicación:   * Si en query params viene **team** entonces devolvemos las noticias correspondientes a ese equipo. * Si no vienen query params devolvemos todas la noticias de todos los equipos. * Si viene **id**, entonces devolvemos la información de la noticia con el id especificado. | {  status : number, // 200 = ok 500 = wrong,  msj : "Información de la noticia con éxito" / "Error al  obtener una noticia",  data : [  {  id : number,  id\_team: number,  team :string  title: string,  description : string,  date: date  }  ]  } |

## ADMINISTRADOR

**PERSONA (UTILIZADOS POR ADMINISTRADORES O EMPLEADOS)**

**URL : http://ip:5000/esb/person/\*\***

**En los endpoints solo se especificará lo enviado en \*\*data\*\* y/o en \*\*msj\*\***

**Formato de Salida :**

~~~

{

status : number, // 200 = ok 500 = wrong

msj : string,

data : cualquier objeto [], {}, string, etc

}

**Crear Persona**

**Endpoint**(**post**): http://ip:5000/esb/person/create\*\*

**Enviar**

{

name : string,

lastname : string,

birthday : date,

nationality : number,

id\_stand : number

status : string,

id\_team : number,

photo : string

}

**Respuesta**:

msj : "Jugador o DT creado con exito" / "Error al crear jugador o dt"

**Actualizar Persona**

**Endpoint**(**put**): http://ip:5000/esb/person/update\*\*

**Enviar**

{

id\_person : number,

name : string,

lastname : string,

birthday : date,

nationality : number,

id\_stand : number

status : string,

id\_team : number,

photo : string

}

**Respuesta**:

msj : "Jugador o DT actualizado" / "Error al actualizar jugador o dt"

**Ver Persona**

**Endpoint**(**get**):http://ip:5000/esb/person/read/\*\*

Ej : http://ip:5000/esb/person/read/?id=2

**Respuesta**:

data : [

{

id\_person : number

name : string,

lastname : string,

birthday : date,

nationality : string,

id\_stand :number,

stand : string

status : string,

id\_team : number,

name\_team : string

photo : string

}

]

**Eliminar Persona**

**Endpoint**(**delete**): http://ip:5000/esb/person/delete/:id\*\*

Ej : http://ip:5000/esb/person/delete/2

**Respuesta**:

msj : "Jugador o DT eliminado" / "Error al eliminar jugador o dt"

**Actualizar estado de persona:**

**Endpoint**(**post**) http://ip:5000/esb/user/admin/update/status

**Entrada :**

{

id\_user : number,

status : string // “congelada” o “activa”

}

**Salida :**

msj : "Estado de usuario actualizado" / "Error al actualizar el estado del usuario"

Este endpoint es de usuario, persona es para jugador o tecnico

**REPORTES**

**URL : http://ip:5000/esb/admin/report/\*\***

**En los endpoints solo se especificará lo enviado en \*\*data\*\* y/o en \*\*msj\*\***

**Formato de Salida :**

~~~

{

status : number, // 200 = ok 500 = wrong

msj : string,

data : cualquier objeto [], {}, string, etc

}

**Suscritos a X equipo**

**Endpoint**(**get**):http://ip:5000/esb/admin/report/subscribe/\*\*

QUERY PARAM \*\*equipo\*\*

Ej : http://ip:5000/esb/admin/report/subscribe/1

**Respuesta**:

data : [

{

equipo : string,

id\_user : number,

name : string,

lastname :string

}

]

**Con o sin membresia**

**Endpoint**(**get**):http://ip:5000/esb/admin/report/membership\*\*

**Respuesta**:

data : [

{

id\_user : number,

name : string,

lastname :string,

membership : boolean

}

]

**Top cantidad de membresias**

**Endpoint**(**get**):http://ip:5000/esb/admin/report/memberships\*\*

**Respuesta**:

data : [

{

id\_user : number,

name : string,

lastname :string,

memberships : number

}

]

**Mas dinero gastado**

**Endpoint**(**get**):http://ip:5000/esb/admin/report/expenses\*\*

**Respuesta**:

data : [

{

id\_user : number,

name : string,

lastname :string,

expenses : number

}

]

**Usuarios por X pais**

**Endpoint**(**get**): http://ip:5000/esb/admin/report/country/\*\*

QUERY PARAM \*\*pais\*\*

Ej: http://ip:5000/esb/admin/report/country/?pais=1

**Respuesta**:

data : [

{

id\_user : number,

name : string,

lastname :string,

country : string

}

]

**Usuarios por X genero**

**Endpoint**(**get**): http://ip:5000/esb/admin/report/genre/\*\*

QUERY PARAM \*\*genero\*\*

Ej: http://ip:5000/esb/admin/report/genre/?genero=1

**Respuesta**:

data : [

{

id\_user : number,

name : string,

lastname :string,

genre : string

}

]

**Usuarios por X edad**

**Endpoint**(**get**): http://ip:5000/esb/admin/report/age/\*\*

QUERY PARAM \*\*edad\*\*

Ej: http://ip:5000/esb/admin/report/age/?edad=19

**Respuesta**:

data : [

{

id\_user : number,

name : string,

lastname :string,

edad : number

}

]

**Empleados con MAS/MENOS noticias**

**Endpoint**(**get**): http://ip:5000/esb/admin/report/news\*\*

**Respuesta**:

data : [

{

id\_user : number,

name : string,

lastname :string,

news : number

}

]

**Empleados con MAS/MENOS noticias por X equipo**

**Endpoint**(**get**): http://ip:5000/esb/admin/report/news/team/\*\*

QUERY PARAM \*\*equipo\*\*

ej: http://ip:5000/esb/admin/report/news/team/?equipo=2

**Respuesta**:

data : [

{

id\_user : number,

name : string,

lastname :string,

news : number,

team : string

}

]

**Bitácora**

**Endpoint**(**get**): http://ip:5000/esb/admin/report/logs\*\*

**Respuesta**:

data : [

{

id\_user : number,

name : string,

lastname :string,

log : string,

created : date,

is\_error : boolean

}

]

## USUARIO

**Crear**

**Endpoint**(**post**): ip/user/add

**Request**

Parámetros con código HTTP :

{  
 name: “”,

lastname: “”,

password: “pass”,

email: “@email.com” ,

phone: ,

photo : “”,

gender: “”,

birth\_date : “”,

signup\_date: “”,

address: “”,

id\_country: number,

id\_rol: ,  
}

**Response**:

\*200: {"status": true}

\*409: {"status": false} //error al guardar el usuario

**Delete Usuario**

**Endpoint**(**delete**): ip:7500/user/delete

**Enviar**:

{  
 no\_id: 5645  
}

**Respuesta**:

\*200: {"status": true}

\*409: {"status": false} //error al eliminar usuario

**Get Usuarios**

**Endpoint**(**get**): ip/user/all

**Enviar**:

{  
}

**Respuesta**:

\*200: [

{

},

{

},

{

},

{

},

{

}

]

\*409: {"status": false} //error en sql

**LOGIN**

## Login

Para todos los usuarios

\*\*POST : http://ip:5000/esb/user/login\*\*

Entrada :

~~~

{

email : string,

password : string

}

~~~

Salida :

~~~

msj : "" //enviar msj de error si no se loguea

data : {

token : string,

statusAccount: // ‘congelada’, ‘activa“

}

~~~

Nota  
Password: contraseña encriptada con cryptoJs, contraseña para encriptar: ‘SiSaleSA\_’

# Cliente

## Nuevo usuario

\*\*POST : http://ip:5000/esb/user/client/create\*\*

Entrada :

~~~

{

name : string,

lastname : string,

password : string,

email : string,

telephone : string,

photo : string, // ejemplo.jpg

genre : string, // F, M, U

birthday : date,

created : date,

address : string,

id\_country : number,

id\_status : number,

id\_rol : number,

age : number,

membership : boolean

}

~~~

Salida :

~~~

msj : "Usuario creado con exito" / "Error al crear el usuario"

~~~

## Actualizar usuario

\*\*PUT : http://ip:5000/esb/user/client/update\*\*

Entrada :

~~~

{

id: number,

name : string,

lastname : string,

password : string,

email : string,

telephone : string,

photo : string,

genre : string,

birthday : date,

address : string,

id\_country : number,

age : number

}

~~~

Salida :

~~~

msj : "Datos actualizado" / "Error al actualizar los datos"

\*200: {"status": “Datos actualizado”}

\*409: {"status": "Error al actualizar los datos"} //error al guardar el usuario

~~~

## Ver Perfil de usuario

\*\*GET : http://ip:5000/esb/user/client/:id\*\*

##### Ej : http://ip:5000/esb/user/client/1

Salida :

~~~

data : {

name : string,

lastname : string,

email : string,

telephone : string,

photo : string,

genre : string,

birthday : date,

address : string,

id\_country : number,

country : string,

age : number

}

~~~

## Verificacion de correo

\*\*PATCH : http://ip:5000/esb/user/client/:id\*\*

Entrada :

~~~

{

id\_user : number,

verify : boolean

}

~~~

Salida :

~~~

msj : "Correo verificado" / "Error al verificar correo"

~~~

## 

## PREDICCIÓN

**get Prediccion**

**Endpoint**(**post**): ip/predict/

**Enviar**

Parámetros con código HTTP :

{

team1\_id:

team2\_id:

//time\_min: 90 min

}

Respuesta :

data : {

team1: 5 //integer de los goles

team2: 5 //integer de los goles

graphic: base64 // “” enviar imagen o cadena vacia

}

# Quiniela

**Crear Quiniela**

**Endpoint**(**post**): ip/quiniela/create

**Enviar**

Parámetros con código HTTP :

{

team1\_id:

team2\_id:

goals\_teams1:

goals\_teams2:

}

Respuesta :

{

msj : “Unido a quiniela”

}