

## Soccer Stats

# Documentación General

IP: ip:5000

### Microservicios:

- Clientes
- Servicios\_Administrativos
  - Estadio
  - Partido
  - Noticias
- Administrador
  - Persona
  - Reportes
- Usuario
- Predicción

## Country

Guía: <https://gist.github.com/adhipg/1600028>

```
CREATE TABLE IF NOT EXISTS `country` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `iso` char(2) NOT NULL,  
  `name` varchar(80) NOT NULL,  
  `nickname` varchar(80) NOT NULL,  
  `iso3` char(3) DEFAULT NULL,  
  `numcode` smallint(6) DEFAULT NULL,  
  `phonecode` int(5) NOT NULL,  
  PRIMARY KEY (`id`)  
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

Salida:

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

## Encriptación:

### API

See: <https://cryptojs.gitbook.io/docs/>

#### AES Encryption

##### Plain text encryption

```
var CryptoJS = require("crypto-js");

// Encrypt
var ciphertext = CryptoJS.AES.encrypt('my message', 'secret key 123').toString();

// Decrypt
var bytes = CryptoJS.AES.decrypt(ciphertext, 'secret key 123');
var originalText = bytes.toString(CryptoJS.enc.Utf8);

console.log(originalText); // 'my message'
```

## Puertos

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

### Headers:

```
{
  authorization: Bearer <Token>
}
```

### Token JWT

```
{
  id_usuario: 2,
  id_rol: 1
}
```

### Roles:

- 1: admin
- 2: Empleado
- 3: Cliente

## CLIENTES

### Obtener Membresia

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

IP: 5000/esb/client/membership

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

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

Entrada:

```
{
  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

Parámetro: 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\*\***

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/\\*\\*](http://ip:5000/esb/client/reports/person/**)

parámetro: 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\\*\\*](http://ip:5000/esb/client/reports/person/higher**)

parámetro: 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/\\*\\*](http://ip:5000/esb/client/reports/person/lower/**)

parámetro: 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/**`

parámetro: `competicion`

Ejemplo : `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/**`

parámetro: `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 antigüedad

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

parámetro: `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/**`

parámetro: `país`

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\\*\\*](http://ip:5000/esb/client/reports/stadium/capacity**)

parámetros: 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/\\*\\*](http://ip:5000/esb/client/reports/team/game/**)

parámetro: 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/\\*\\*](http://ip:5000/esb/client/reports/team/person/**)

parámetro: 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/\\*\\*](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\\*\\*](http://ip:5000/esb/client/reports/person/competition/incidents**)

parámetro: competicion

parámetro: incidente

parámetro: 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/\\*\\*](http://ip:5000/esb/client/reports/team/competitions/**)

parámetro: 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/\\*\\*](http://ip:5000/esb/client/reports/games/year/**)

parámetro: 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/\\*\\*](http://ip:5000/esb/client/reports/games/teams/**)

parámetro: local

parámetro: 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

### Estadio

Crear estadio -> POST : ip/stadium

Entrada:

```
{
  name : string,
  foundation_date : date,
  capacity: number,
  id_country : number,
  address: string,
  state: string,
  photo : string //base64
}
```

Salida:

```
{
  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:

```
{
  id : number,
  name : string,
  foundation_date : date,
  capacity: number,
  id_country : number,
  address: string,
  state: string,
  photo : string //base64
}
```

Salida:

```
{
  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

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.

Salida:

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,
      foundation_date : date,
      capacity: number,
      id_country : number,
      country : string,
      address: string,
      state: string,
      photo : string //base64
    },
    {
      id : number,
      name : string,
      foundation_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,
    foundation_date : date,
    capacity: number,
    id_country : number,
    address: string,
    state: string,
    photo : string //base64
  }
}
```

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

ip/stadium/:id

Ej : ip/stadium/2

Salida:

```
{
  status : number, // 200 = ok   500 = wrong,
  msj : "Estadio eliminado" / "Error al eliminar el estadio",
  data : []
}
```

## Equipo

Crear equipo -> POST : ip/team

Entrada:

```
{
  name : string,
  foundation_date : date,
  id_country : number
  photo : string //base64
}
```

Salida:

```
{
  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:

```
{
  id : number,
  name : string,
  foundation_date : date,
  id_country : number,
  photo : string //base64
}
```

Salida:

```
{
  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

ip/team?id

Ej : ip/team/?id=2

Parámetros:

- Si viene el query param se manda la información del equipo correspondiente.
- Si no, se mandan la información de todos los equipos.

Salida:

```

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,
      foundation_date : date,
      id_country : number,
      country : string,
      photo : string
    },
    {
      id : number,
      name : string,
      foundation_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,
    foundation_date : date,
    id_country : number,
    country : string,
    photo : string
  }
}

```

## Crear una Liga

Endpoint(post): <http://ip:5000/esb/league>

Entrada:

```

{
  name: string,
  description: string,
  type: number,
  id_country : number
}

```

Salida:

```

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 existir la tabla League

Endpoint(post): <http://ip:5000/esb/competition>

Entrada:

```
{
  league: number,
  year : number,
  champion_team : number,
}
```

Salida:

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

### Eliminar competencia

Endpoint(delete): <http://ip:5000/esb/competition/{id}>

Salida:

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

### Actualizar competencia

Endpoint(put): <http://ip:5000/esb/competition/>

Entrada:

```
{
  league: number,
  year : number,
  champion_team : number,
}
```

Salida:

```
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 qué fecha jugó para ese equipo.

POST : [ip/transfer-player](#)

Entrada:

```
{
  id_player: number,
  id_team_origin : number,
  id_team_destination : number,
}
```

Salida:

```
{
  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:

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:

```
{
  id_coach: number,
  id_team_origin : number,
  id_team_destination : number,
}
```

Salida:

```
{
  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

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:



```
{
  id_game : number,
  id_person : number,
  id_team : number,
  descripcion : string,
  id_incidencia : number,
  minuto : number
}
```

Salida:

```
{
  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

Entrada:

```
{
  id_person: number,
  state : number,
}
```

Salida:

```
{
  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:

```
{
  id_team: number,
  title: string,
  description : string,
  date: date
}
```

Salida:

```
{
  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:

Parámetros:

- id (id 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 las noticias de todos los equipos.
- Si viene id, entonces devolvemos la información de la noticia con el id especificado.

Salida:

```
{
  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\*\***

Salida:

```
{
  status : number, // 200 = ok   500 = wrong
  msj : string,
  data : cualquier objeto [], {}, string, etc
}
```

## Crear Persona

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

Entrada:

```
{
  name : string,
  lastname : string,
  birthday : date,
  nationality : number,
  id_stand : number
  status : string,
  id_team : number,
  photo : string
}
```

Salida:

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

## Actualizar Persona

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

Entrada:

```
{
  id_person : number,
  name : string,
  lastname : string,
  birthday : date,
  nationality : number,
  id_stand : number
  status : string,
  id_team : number,
  photo : string
}
```

Salida:

```
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

Salida:

```
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`

Salida:

```
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\*\***

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/\*\*

Parámetro: equipo

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

Salida:

```
data : [
  {
    equipo : string,
    id_user : number,
    name : string,
    lastname : string
  }
]
```

#### Con o sin membresia

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

Salida:

```
data : [
  {
    id_user : number,
    name : string,
    lastname : string,
    membership : boolean
  }
]
```

#### Top cantidad de membresias

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

Salida:

```
data : [
  {
    id_user : number,
    name : string,
    lastname : string,
    memberships : number
  }
]
```

#### Más dinero gastado:

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

Salida:

```
data : [
  {
    id_user : number,
    name : string,
    lastname : string,
    expenses : number
  }
]
```

#### Usuarios por X país

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

Parámetro: país

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

Salida:

```
data : [
  {
    id_user : number,
    name : string,
    lastname : string,
    country : string
  }
]
```

#### Usuarios por X genero

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

Parámetro: genero

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

Salida:

```
data : [
  {
    id_user : number,
    name : string,
    lastname : string,
    genre : string
  }
]
```

#### Usuarios por X edad

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

Parámetro: edad

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

Salida:

```
data : [
  {
    id_user : number,
    name : string,
    lastname : string,
    edad : number
  }
]
```

#### Empleados con MÁS/MENOS noticias

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

Salida:

```
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/\*\*

Parámetro: equipo

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

Salida:

```
data : [
  {
    id_user : number,
    name : string,
    lastname : string,
    news : number,
    team : string
  }
]
```

#### **Bitácora**

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

Salida:

```
data : [
  {
    id_user : number,
    name : string,
    lastname : string,
    log : string,
    created : date,
    is_error : boolean
  }
]
```

## 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
}
```

### ## Verificación de correo

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

Entrada :

```
{
    id_user : number,
    verify : boolean
}
```

Salida:

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

## 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: ,
}
```

Salida:

```
*200: {"status": true}
*409: {"status": false} //error al guardar el usuario
```

## Delete Usuario

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

Entrada:

```
{
  no_id: 5645
}
```

Salida:

```
*200: {"status": true}
*409: {"status": false} //error al eliminar usuario
```

## Get Usuarios

Endpoint(get): ip/user/all

Entrada: { }

Salida:

```
*200[ ]
*409: {"status": false} //error en sql
```

# PREDICCIÓN

## get Prediccion

Endpoint(post): ip/predict/

Entrada:

Parámetros con código HTTP :

```
{
  team1_id:
  team2_id:
  //time_min: 90 min
}
```

Salida:

```
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

Entrada:

Parámetros con código HTTP :



```
{
  team1_id:
  team2_id:
  goals_teams1:
  goals_teams2:
}
```

Salida:

```
{
  msj : "Unido a quiniela"
}
```

