DOCUMENTACIÓN PROYECTO HÍPICA

APIREST

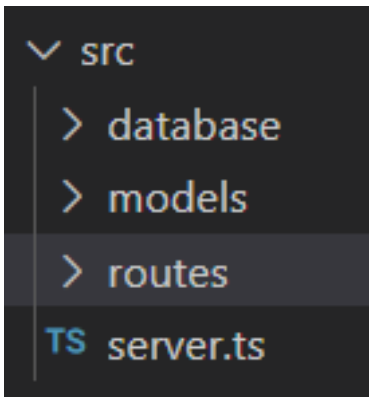
La ApiRest está compuesta por:

-Base de Datos.

-Esquemas.

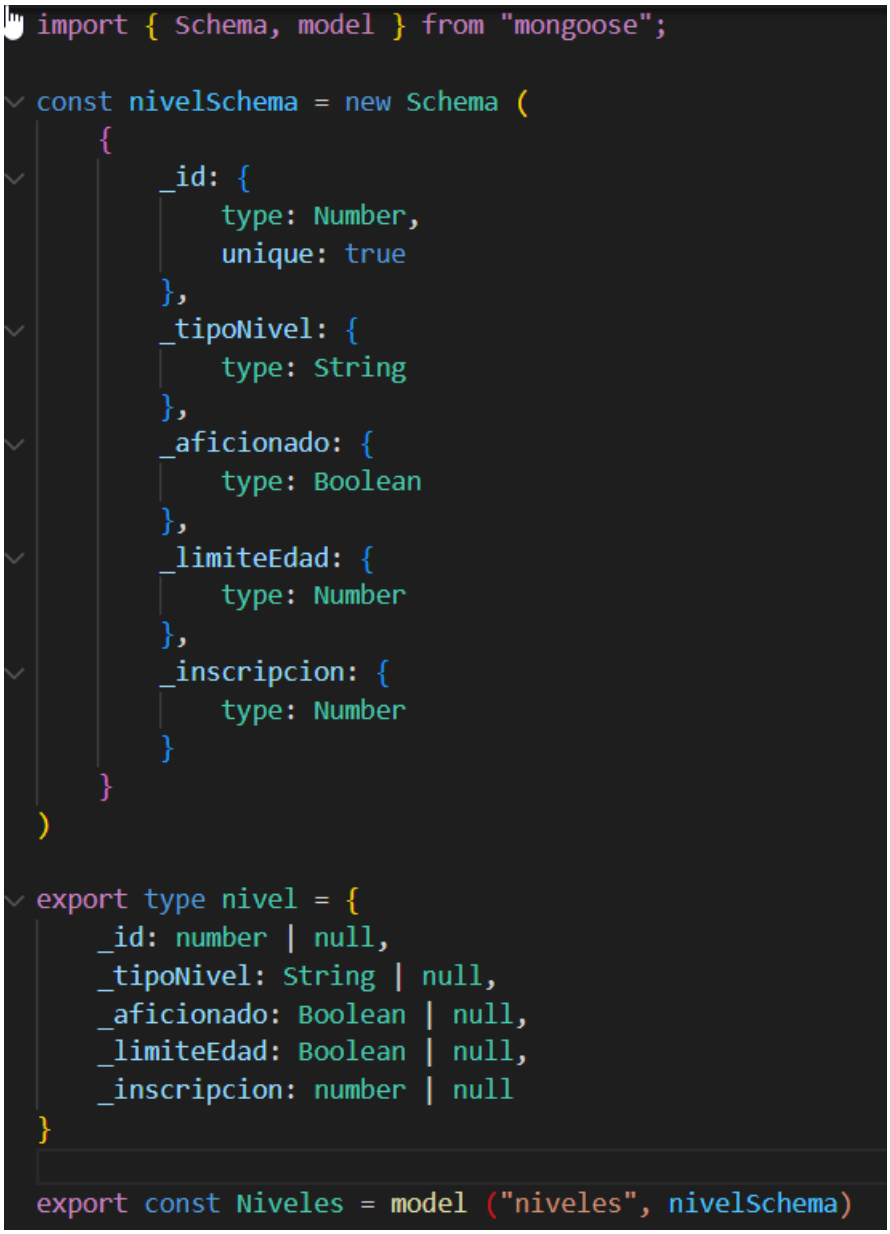
-Rutas

-Servidor.



Los esquemas utilizados son:

-Esquema niveles.



```
import { Schema, model } from "mongoose";

const nivelSchema = new Schema (
    {
        _id: {
            type: Number,
            unique: true
        },
        _tipoNivel: {
            type: String
        },
        _aficionado: {
            type: Boolean
        },
        _limiteEdad: {
            type: Number
        },
        _inscripcion: {
            type: Number
        }
    }
)

export type nivel = {
    _id: number | null,
    _tipoNivel: String | null,
    _aficionado: Boolean | null,
    _limiteEdad: Boolean | null,
    _inscripcion: number | null
}

export const Niveles = model ("niveles", nivelSchema)
```

-Esquema participantes.

```
import { Schema, model } from "mongoose";

const participanteSchema = new Schema (
    {
        _id: {
            type: Number,
            unique: true
        },
        _nombre: {
            type: String
        },
        _edad: {
            type: Number
        },
        _nivel: {
            type: String
        },
        _modalidad: {
            type: String
        },
        _nacionalidad: {
            type: String
        },
        _nomCaballo: {
            type: String
        },
        _raza: {
            type: String
        },
        _edadCaballo: {
            type: Number
        },
        _cabEstabulado: {
            type: Boolean
        },
        _totalSaltos: {
            type: Number
        },
        _maxAltura: {
            type: Number
        },
        _TLimiteS: {
            type: Number
        },
        _derribosS: Number,
        _rehusoS: {
            type: Number,
            max: 3
        },
        _caidaS: {
            type: Number,
            max: 2
        },
        _tiempoS: Number,
        _TLimiteC: {
            type: Number
        },
        _rehusoC: {
            type: Number,
            max: 3
        },
        _caidaC: {
            type: Number,
            max: 2
        },
        _tiempoC: Number,
        _parada: {
            type: Number,
            max: 10
        },
        _paso: {
            type: Number,
            max: 10
        },
        _trote: {
            type: Number,
            max: 10
        },
        _galope: {
            type: Number,
            max: 10
        },
        _pasoAtras: {
            type: Number,
            max: 10
        },
        _transiciones: {
            type: Number,
            max: 10
        },
        _cambioDirec: {
            type: Number,
            max: 10
        },
        _figuras: {
            type: Number,
            max: 10
        },
        _movLateral: {
            type: Number,
            max: 10
        },
        _piruetas: {
            type: Number,
            max: 10
        }
    }
)

export type participante = {
    _id: Number | null,
    _nombre: String | null,
    _edad: Number | null,
    _nacionalidad: String | null,
    _nivel: String | null,
    _modalidad: String | null,
    _nomCaballo: String | null,
    _raza: String | null,
    _edadCaballo: String | null,
    _cabEstabulado: Boolean | null
}

export type pSalto = {
    _id: Number | null,
    _nombre: String | null,
    _edad: Number | null,
    _nacionalidad: String | null,
    _nivel: String | null,
    _modalidad: String | null,
    _nomCaballo: String | null,
    _raza: String | null,
    _edadCaballo: String | null,
    _cabEstabulado: Boolean | null,
    _totalSaltos: Number | null,
    _maxAltura: Number | null,
    _TLimiteS: Number | null,
    _derribosS: Number | null,
    _rehusoS: Number | null,
    _caidaS: Number | null,
    _tiempoS: Number | null
}

export type pCross = {
    _id: Number | null,
    _nombre: String | null,
    _edad: Number | null,
    _nacionalidad: String | null,
    _nivel: String | null,
    _modalidad: String | null,
    _nomCaballo: String | null,
    _raza: String | null,
    _edadCaballo: String | null,
    _cabEstabulado: Boolean | null,
    _TLimiteC: Number | null,
    _rehusoC: Number | null,
    _caidaC: Number | null,
    _tiempoC: Number | null

}

export type pDoma = {
    _id: Number | null,
    _nombre: String | null,
    _edad: Number | null,
```

```
    _nombre: String | null,
    _edad: Number | null,
    _nacionalidad: String | null,
    _nivel: String | null,
    _modalidad: String | null,
    _nomCaballo: String | null,
    _raza: String | null,
    _edadCaballo: String | null,
    _cabEstabulado: Boolean | null,
    _parada: Number | null,
    _paso: Number | null,
    _trote: Number | null,
    _galope: Number | null,
    _pasoAtras: Number | null,
    _transiciones: Number | null,
    _cambioDirec: Number | null,
    _figuras: Number | null,
    _movLateral: Number | null,
    _piruetas: Number | null
}

export const Participantes = model ("participantes", participanteSchema)
```

La conexión con la base de datos es la siguiente:

```
import mongoose from 'mongoose';

class DataBase {

    private _cadenaConexion2: string = 'mongodb://localhost/test'
    private _cadenaConexion:string= `mongodb+srv://usuario2:usuario2@cluster0.q55uu.mongodb.net/test?retryWrites=true&w=majority
    constructor(){

    }
    set cadenaConexion(_cadenaConexion: string){
        this._cadenaConexion = _cadenaConexion
    }

    conectarBD = async () => {
        const promise = new Promise<string>( async (resolve, reject) => {
            await mongoose.connect(this._cadenaConexion, {
            })
            .then( () => resolve(`Conectado a ${this._cadenaConexion}`) )
            .catch( (error) => reject(`Error conectando a ${this._cadenaConexion}: ${error}`) )
        })
        return promise

    }

    desconectarBD = async () => {

        const promise = new Promise<string>( async (resolve, reject) => {
            await mongoose.disconnect()
            .then( () => resolve(`Desconectado de ${this._cadenaConexion}`) )
            .catch( (error) => reject(`Error desconectando de ${this._cadenaConexion}: ${error}`) )
        })
        return promise
    }

}
export const db = new DataBase()
```

Para realizar las rutas se han utilizado diferentes métodos: GET, POST, PUT Y DELETE.

GET:

```
private getNiveles = async (req: Request, res: Response) => {
    await db.conectarBD()
    .then( async (mensaje) => {
        console.log(mensaje)
        const query  = await Niveles.aggregate(
            [
                {
                    $lookup: {
                        from: 'participantes',
                        localField: '_tipoNivel',
                        foreignField: '_nivel',
                        as: 'participantes'
                    }
                }
            ]
        )
        res.json(query)
    })
    .catch((mensaje) => {
        res.send(mensaje)
    })
    await db.desconectarBD()
}
```

POST:

```
private newNivel = async (req: Request, res: Response) => {
    const {id, tipoNivel, aficionado, limiteEdad, inscripcion} = req.body
    await db.conectarBD()
    let dSchema = {
        "_id": id,
        "_tipoNivel": tipoNivel,
        "_aficionado": aficionado,
        "_limiteEdad": limiteEdad,
        "_inscripcion": inscripcion
    }
    const oSchema = new Niveles(dSchema)
    await oSchema.save()
        .then((doc: any) => res.send('Nivel salvado: ' + doc))
        .catch((err: any) => res.send(err))
    await db.desconectarBD()
}
```

PUT:

```
private modiNivel = async (req: Request, res: Response) => {
    const { id } = req.params
    const { aficionado, limiteEdad, inscripcion } = req.body
    await db.conectarBD()
    await Niveles.findOneAndUpdate(
        {
            "_id": id
        },
        {
            "_aficionado": aficionado,
            "_limiteEdad": limiteEdad,
            "_inscripcion": inscripcion
        },
        {
            new: true,
            runValidators: true
        }
    )
    .then((doc: any) => res.send('Nivel modificado: ' + doc))
    .catch((err: any) => res.send('Error: ' + err))
    await db.desconectarBD()
}
```

-DELETE:

```
private elimParticipante = async (req: Request, res: Response) => {
    const {nombre} = req.params
    await db.conectarBD()
    await Participantes.findOneAndDelete(
        {
            '_nombre': nombre
        }
    )
    .then((doc: any) => res.send('Participante eliminado ' + doc))
    .catch((err: any) => res.send('Error: ' + err))
    await db.desconectarBD()
}
```

Todas las rutas que se hay en la ApiRest son:

```
misRutas(){
    this._router.get('/niveles', this.getNiveles)
    this._router.get('/participantes', this.getParticipantes)
    this._router.get('/niveles/:tipoNivel', this.getNivel)
    this._router.get('/participante/:nombre', this.getParticipante)
    this._router.get('/doma', this.getDoma)
    this._router.get('/salto', this.getSalto)
    this._router.get('/cross', this.getCross)
    this._router.post('/nivel', this.newNivel)
    this._router.post('/participante', this.newParticipante)
    this._router.put('/modificarNivel/:id', this.modiNivel)
    this._router.put('/modificarPartici/:nombre', this.modiPartici)
    this._router.delete('/eliminarPartici/:nombre', this.elimParticipante)
    this._router.delete('/eliminarNivel/:tipoNivel', this.elimNivel)
}
```

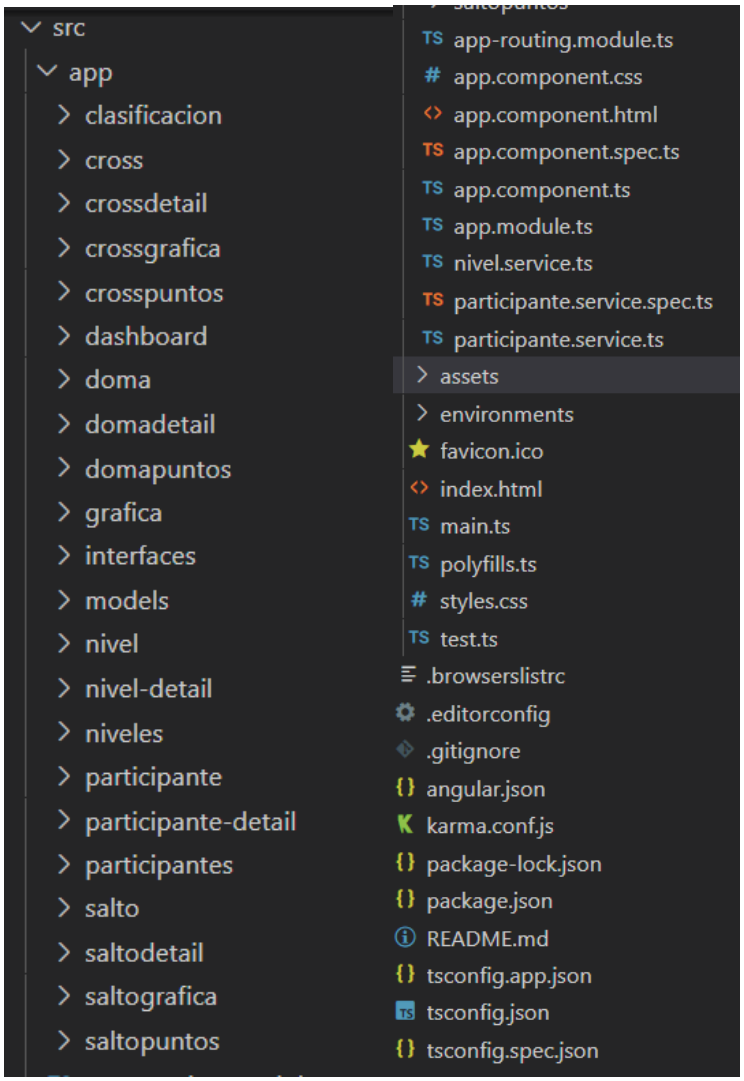
ANGULAR

El proyecto Angular está compuesto por:

-Modelos.

-Componentes.

-Servicios.



A través de los servicios creamos la conexión con las rutas de la ApiRest.

```
export class NivelService {
  private apiURL = 'https://restapi-hipica.herokuapp.com/'
  selectedNivel: any;

  constructor(private http: HttpClient) {}
```

Los modelos son los siguientes:

-La clase Nivel.

```
export class Nivel {
  _id: number;
  _tipoNivel: string;
  _aficionado: boolean;
  _limiteEdad: number;
  _inscripcion: number;
  _participantes: Array<Participante>;
```

-La clase Participante, que es la clase “padre”, de ella extienden otras tres clases.

```
export class Participante {
    _id: number;
    _nombre: string;
    _edad: number;
    _nacionalidad: string;
    _nivel: string;
    _modalidad: string;
    _nomCaballo: string;
    _raza: string;
    _edadCaballo: number;
    _cabEstabulado: boolean;
```

```
puntosT() {
    let puntosT: number
    puntosT = 0
    return puntosT
}
```

-Clase extendidas de Participante.

<u>Doma</u>

```
export class Doma extends Participante {
    _parada: number
    _paso: number
    _trote: number
    _galope: number
    _pasoAtras: number
    _transiciones: number
    _cambioDirec: number
    _figuras: number
    _movLateral: number
    _piruetas: number
```

```
override puntosT() {
    let puntosT: number
    puntosT = this._parada + this._paso + this._trote + this._galope +this._pasoAtras + this._transiciones +
    this._cambioDirec + this._figuras + this._movLateral + this._piruetas
    return (puntosT)
}
```

Salto

```
export class Salto extends Participante {
    public _totalSaltos: number
    public _maxAltura: number
    public _TLimiteS: number
    public _derribosS: number
    public _rehusosS: number
    public _caidaS: number
    public _tiempoS: number
```

```
Stiempo(): any {
    let tiemp: number
    if (this._TLimiteS < this._tiempoS) {
        tiemp = this._tiempoS * 1
        return(tiemp)
    } else {
        tiemp = 0
        return(tiemp)
    }
}

penalizaciones() {
    let penal: number
    penal = (this._derribosS * 4) + (this._rehusosS * 10) + (this._caidaS * 20) + (this.Stiempo())
    return (penal)
}

override puntosT(): any {
    let puntosT: number
    puntosT = 100 - this.penalizaciones()
    return (puntosT)
}
```

Cross

```
export class Cross extends Participante {
    public _TLimiteC: number
    public _rehusoC: number
    public _caidaC: number
    public _tiempoC: number
```

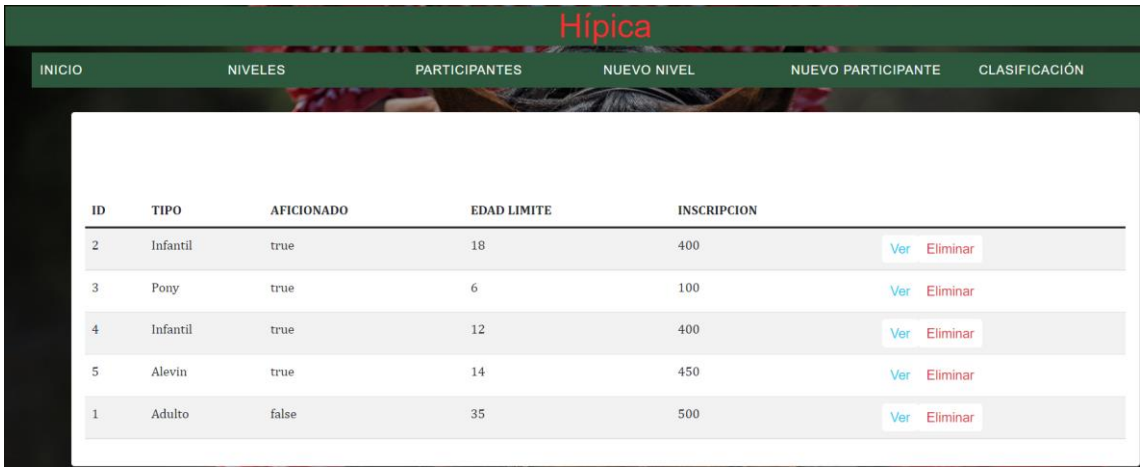
```
tiempo(): any {
    let tiem: number
    if (this._TLimiteC < this._tiempoC) {
        tiem = this._tiempoC * 1
        return(tiem)
    } else {
        tiem = 0
        return(tiem)
    }
}

penalizaciones() {
    let penal: number
    penal = (this._rehusoC * 10) + (this._caidaC * 20) + (this.tiempo())
    return (penal)
}

override puntosT(): any {
    let puntosT: number
    puntosT = 100 - this.penalizaciones()
    return (puntosT)
}
```

VISTA DEL PROYECTO

Muestra de los niveles y participantes.



ID	TIPO	AFICIONADO	EDAD LIMITE	INSCRIPCION	
2	Infantil	true	18	400	Ver Eliminar
3	Pony	true	6	100	Ver Eliminar
4	Infantil	true	12	400	Ver Eliminar
5	Alevín	true	14	450	Ver Eliminar
1	Adulto	false	35	500	Ver Eliminar

ID	NOMBRE	EDAD	NACIONALIDAD	NIVEL	MODALIDAD	NOM.CABALLO	RAZA	ED.CABALLO	CAB.ESTABULADO	
4	Miguel	18	Español	Infantil		Pionero	Árabe	9	false	Ver Eliminar
5	Ana	14	Francesa	Alevín		Lombardo	Anglo-Árabe	6	true	Ver Eliminar
8	David	16	Española	Infantil	Cross	Pirlo	PRE	4	true	Ver Eliminar
6	Marta	22	Española	Adulto	Doma	Dino	Árabe	5	true	Ver Eliminar
1	Pedro	24	Española	Adulto	Doma	Caudor	Árabe	10	true	Ver Eliminar
2	Anthony	16	Británica	Infantil	Salto	Mento	Anglo	5	true	Ver Eliminar
3	Sandra	12	Española	Alevín	Cross	Lucho	PRE	11	true	Ver Eliminar
7	Carmen	15	Española	Infantil		Cento	PRE	5	false	Ver Eliminar

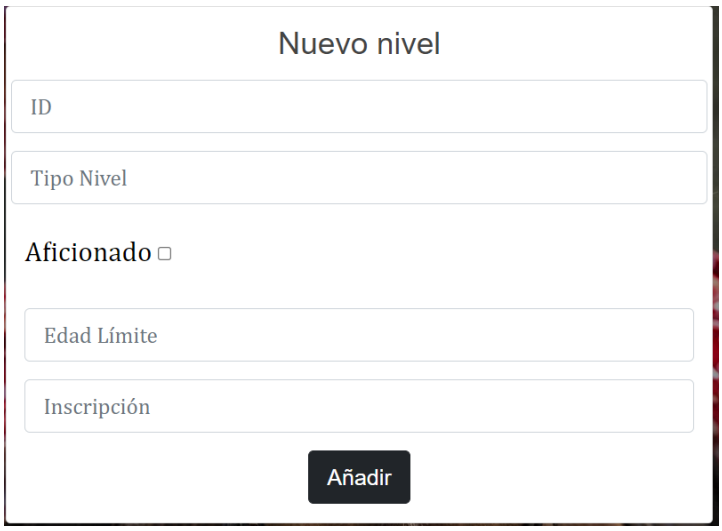
Los participantes según de la modalidad a la que pertenezcan

ID	NOMBRE	EDAD	NACIONALIDAD	NIVEL	MODALIDAD	NOM.CABALLO	RAZA	ED.CABALLO	CAB.ESTABULADO	
6	Marta	22	Española	Adulto	Doma	Dino	Árabe	5	true	Puntuación Eliminar
1	Pedro	24	Española	Adulto	Doma	Caudor	Árabe	10	true	Puntuación Eliminar

ID	NOMBRE	EDAD	NACIONALIDAD	NIVEL	MODALIDAD	NOM.CABALLO	RAZA	ED.CABALLO	CAB.ESTABULADO	SALT.TOTALES	MAX.ALTURA	T.LIMITE	
2	Anthony	16	Británica	Infantil	Salto	Mento	Anglo	5	true	25	1.5	18	Puntuación Eliminar

ID	NOMBRE	EDAD	NACIONALIDAD	NIVEL	MODALIDAD	NOM.CABALLO	RAZA	ED.CABALLO	CAB.ESTABULADO	T.LIMITE	
8	David	16	Española	Infantil	Cross	Pirlo	PRE	4	true	18	Puntuación Eliminar
3	Sandra	12	Española	Alevín	Cross	Lucho	PRE	11	true	30	Puntuación Eliminar

Para agregar un nivel o participante.

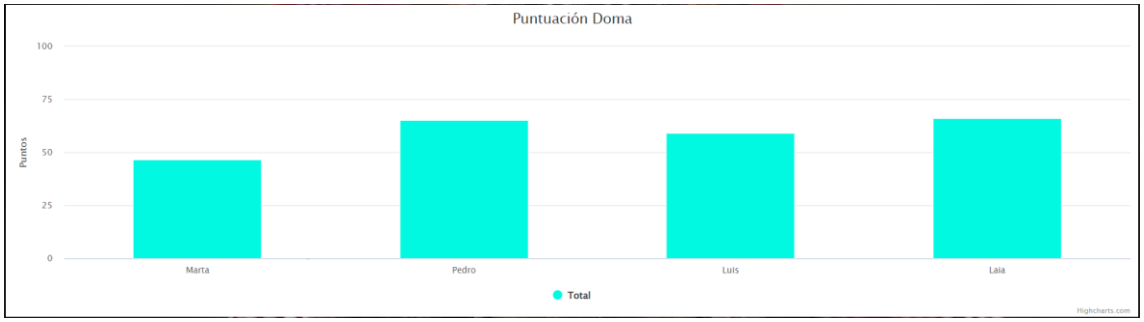


Editar un nivel o participante ya añadido.

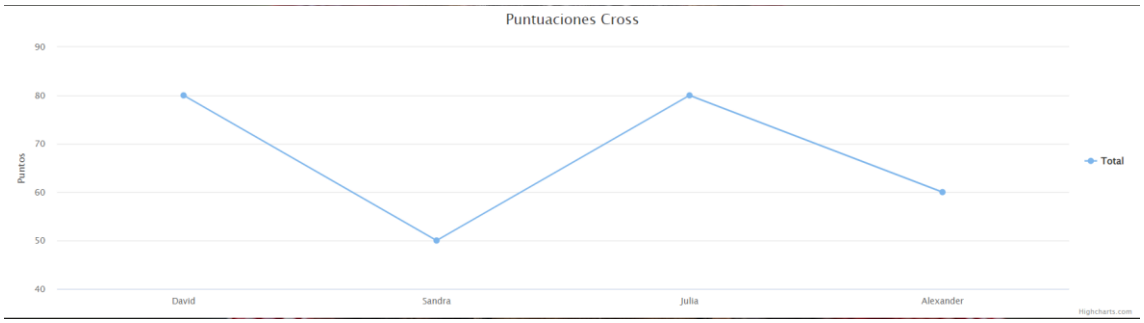
GRÁFICAS

Las gráficas representan el método creado para la suma de los puntos finales de cada participante, dependiendo de la modalidad en la que esté.

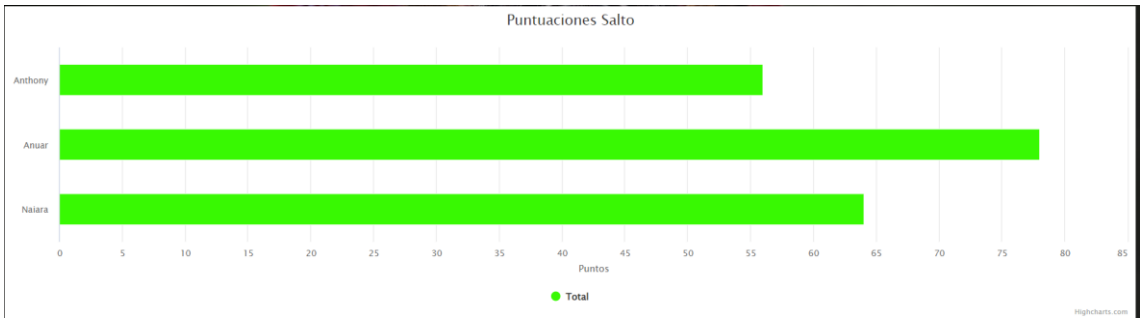
Los de Doma:



Los de Cross:



Los de Salto:


Puntuaciones Salto
Anthony
Anuar
Naiara
0
5
10
15
20
25
30
35
40
45
50
55
60
65
70
75
80
85
Puntos
Total
Highcharts.com