SEQUELIZE

[Henry Students (soyhenry.com)](https://www.students.soyhenry.com/classes/31?cohortId=9)

const { Sequelize, Op } = require('sequelize');

Para generar la conexion con la base de datos:

const sequelize = new Sequelize(

  "postgres://postgres:Guadalupe10@localhost:5432/demo",

  {

    // logging: (...msg) => console.log(msg)

    logging: false,

  }

);

Sequelize no crea base de datos, solo se puede conectar.

CREACION DE MODELOS (TABLAS) EN LA BASE DE DATOS

const { Player, Team } = sequelize.models;

Todos los modelos los creo en una carpeta aparte (Models) y los creo dentro de un module.exports

module.exports = sequelize => {

  sequelize.define('Player', {

    firstName: {

      type: DataTypes.STRING

    },

    lastName: {

      type: DataTypes.STRING

    },

    username: {

      type: DataTypes.STRING,

      allowNull: false,

      unique: true,

      set(value) {

        this.setDataValue('username', value.toUpperCase());

      }

    },

    birthday: {

      type: DataTypes.DATEONLY,

      defaultValue: DataTypes.NOW

    },

    status: {

      type: DataTypes.ENUM('active', 'injured', 'retired')

    },

    skill: {

      type: DataTypes.INTEGER,

      get() {

        // return this.skill + ' points'; // Wrong!

        return this.getDataValue('skill') + ' points';

      },

      validate: {

        min: 0,

        max: 100,

        isEven(value) {

          if(value % 2 !== 0) {

            throw new Error('Only even values are allowed!')

          }

        }

      }

    },

    password: {

      type: DataTypes.STRING,

      set(value) {

        this.setDataValue('password', (this.firstName + this.lastName + value).split('').sort(() => 0.5 - Math.random()).join(''));

      }

    },

    fullName: {

      type: DataTypes.VIRTUAL,

      get() {

        return `${this.firstName} ${this.lastName}`;

      },

      set() {

        throw new Error('Do not try to set the `fullName` value!');

      }

    }

  }, {

    // timestamps: false

    timestamps: true,

    createdAt: false,

    updatedAt: 'actualizado'

  });

}

Para generar las tablas para mis modelos:

modelPlayer(sequelize);

modelTeam(sequelize);

Ahora debo sincronizar mis modelos para poder enviarlo a la base de datos:

const { Player, Team } = sequelize.models;

Player.sync();

Team.sync();

Sync sin argumentos solo verifica si la table existe. Si existe no hace nada asi tenga diferentes columnas

Sync(alter: true) actualiza a los nuevos atributos (columnas)

Sync(forcé: true) elimina completamente la tabla y la vuelve a crear.

Agrega automáticamente dos columnas (createdAt y updatedAt). Si no los quero les pongo **timestamps:false**

Si queremos sync a muchos modelos seria:

sequelize.sync({force: true})

No ejecuto el sync sino que exporto todos los modelos (…sequelize.models, db: sequelize, Op) porque me voy a generar un servidor de express en el archivo index.js. Cuando levanto el servidor, en ese momento hago el sync.

Arranco con postman Minuto 48.

server.listen(3000, () => {

  console.log('Server running on port 3000');

  db.sync();

});

LOGGING

Para q el logging no describa todas las cosas que hace, en la conexión le puedo pasar **logging:false** cuando me conecto.

QUERYS MINUTO 1:03

findAll 🡪 Trae todos

findByPk 🡪 Busca por Primary Key (uno solo). Esta en GET /players/id/:id

findOne 🡪 Parecido al anterior pero con un atributo diferente a la PK

findOrCreate 🡪 Busca alguna instancia, si lo encuentra lo retorna, si no, lo crea, con los valores q tenga defaults. Devuelve un arreglo con 2 elementos. El primero es la instancia, el segundo es un Bool, true-> lo creo.

ORDENAR 🡪 MINUTO 1:23

AGRUPAR 🡪 MINUTO 1:27 fn puede ser (count, sum, average, etc)

LIMIT, OFFSET (PAGINADO) 🡪 1:30

UTILITY METHODS 1:32 🡪 count, max, min,sum gt(greaterThan)

ACTUALIZAR VARIAS INSTANCIAS 1:36

GETTERS 🡪 1:42 ejemplo 50 points, en lugar de 50. Es un cambio visual, no se guarda en la base de datos

SETTERS 🡪 1:44 Lo modifica antes de guardarlo en la base de datos.

VIRTUAL FIELD 🡪 Campo q no se guarda en la base de datos, se calcula. (FULLNAME)

VALIDATORS 🡪 1:48. Para q no se agreguen datos que no cumplan con ciertas características. (VALIDATE) Ej skill < 100 y skill > 0. Se pueden crear isEven (solo permite valores pares).

ASOCIACIONES 🡪 1:53.

MIXINS 🡪 1:59. Métodos automáticos para hacer las asociaciones (JOINS de sql) crear los enlaces. TRANSFER endpoint (addTeam). Podria hacer varios a la vez (setTeams o addTeams 🡪plural). addTeams no borra y setTeams si borra todo y agrega nuevos.

LAZY LOADING 🡪 me paro en messi y le pido getTeams (no crea el vinculo no JOINEA)

EAGER LOADING 🡪 me paro en messi y le pido getTeams (SI crea el vinculo si JOINEA). Si no quiero que me traiga todos los datos del equipo, uso attributes [].