UTN.BA

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

# Programador Web Avanzado

## Clase N° 7: Mongoose Validators, paginator, dotenv y nodemailer

Profesor: Ing. Leandro Rodolfo Gil Carrano
Email: leangilutn@gmail.com

# Validators

# Mongoose – Validators

Mediante la definición del esquema se pueden validar los campos a insertar:

```
var UsuariosSchemna = mongoose.Schema({


    name:String,
```

En este caso validamos que el tipo de dato insertado sea un string

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

# Mongoose – Validators

```
var UsuariosSchemna = mongoose.Schema({
    name:String,
    usuario:{
        type: String,
        required: [true,"El campo usuario es obligatorio"],
        unique: true
        },
```

En el campo usuario vemos aplicado:

- Unique: no permite insertar otro documento con ese campo repetido
- Required: Es un campo obligatorio. El parámetro puede ser required: true (valida que es obligatorio, es caso de error se muestra mensaje por defecto)
  required: [true, msj] (permite personalizar el mensaje)

# Mongoose – Validators

```
password:{
    type: String,
    trim: true,
    required: [true,"El password es obligatorio"],
    minlength: [6,"El password debe tener al menos 6 caracteres"],
    maxlength: [8,"El password debe tener como máximo 8 caracteres"]
    },
```

En el campo usuario vemos aplicado:

- Minlength: N (Valida que el campo tenga al menos n caracteres, en caso de error muestra mensaje por defecto)
- Minlength: [n,msj] (personaliza el mensaje en caso de error

Para maxlength sigue la misma regla

# Mongoose – Validators

```
phone: {
    type: String,
    validate: {
        validator: function(v) {
        return /\d{2}-\d{4}-\d{4}/.test(v);
        },
        message: '{VALUE} no es un teléfono válido!'
    },
    required: [true, 'El campo teléfono es obligatorio']
}
```

También se puede aplicar una validación personalizada
El método validate tiene 2 índices:

- Validator: Método que se aplicara para validar el campo (reglas de validación)
- Message: Mensaje renderizado en caso de error

# Mongoose – Validators

Para ver mas acerca de métodos de validación:
https://mongoosejs.com/docs/4.x/docs/validation.html

# Queries

# Mongoose – Queries

Sobre los métodos de búsqueda, por ejemplo find se pueden aplicar queries para filtrar información, ordenar o bien seleccionar elementos del resultado a visualizar.
Ejemplo:

```
var producto = await productModel.find({})
.sort({ price: 1 })
```

*Se aplica el orden por precio de forma ascendente sobre los resultados devueltos por el método find.*
*En caso de aplicar Price: -1 el orden será descendente*

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

# Mongoose – Queries

```
var producto = await productModel.find({})
.select({ price: 1, name: 1})
```

*Aplica el select de los atributos especificados (en el ejemplo Price y name), de esta forma solo se retornaran dichos datos.*

# Mongoose – Queries

```
var producto = await productModel.find({})
.where('name').equals('Heladera');
```

*En este caso se aplica un where para filtrar por igualdad por campo name*

Para ver mas acerca de queries:
https://mongoosejs.com/docs/4.x/docs/queries.html

# Mongoose Paginate

# Mongoose – Mongoose Paginate

Mongoose paginator es un modulo que nos permite paginar nuestras consultas.

Para su utilización debemos instalando ejecutando:

**npm install mongoose-paginate-v2**

# Mongoose – Mongoose Paginate

En el archivo bin/mongodb.js aplicar lo siguiente:

```
var mongoose = require('mongoose');
var mongoosePaginate = require('mongoose-paginate-v2');
mongoose.connect('mongodb://localhost/startup', { useNewUrlParser: true }, function(error){
    if(error){
        throw error;
    }else{
        console.log('Conectado a MongoDB');
    }
});
mongoosePaginate.paginate.options = {
    lean:  true,
    limit: 1
};
mongoose.mongoosePaginate = mongoosePaginate
module.exports = mongoose;
```

*El campo limit se aplica por defecto a todas las consultas.*

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

# Mongoose – Mongoose Paginate

Luego en el modelo en el cual queramos aplicar el paginado, configurar lo siguiente (ejemplo ProductsModel)

```
const mongoose = require('../bin/mongodb');
const Schema = mongoose.Schema;
var childSchema = new Schema({ name: 'string' });
//Define a schema
const ProductSchema = new Schema({
 name: {
  type: String,
  trim: true,
  required: true,
 },
 sku: {
  type: String,
  trim: true,
  required: true
 },
 price: {
  type: Number,
  trim: true,
  required: true
 },
 categoria: {type:Schema.ObjectId, ref:"categorias"},
 relacionados:[childSchema]
});
ProductSchema.plugin(mongoose.mongoosePaginate);
module.exports = mongoose.model('products', ProductSchema);
```

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

# Mongoose – Mongoose Paginate

Por ultimo al realizar las consultas debemos aplicar el método paginate (ejemplo controller/products.js)

```
getAll: async function(req, res, next) {
    try{
        var producto = await productModel.paginate({

        },{
            populate: 'categoria',page:req.query.page
        })
        res.status(200).json({status: "success", message: "ok", data: producto});
    }catch(err){
        next(err);
    }


},
```

1er parámetro ({}): Permite filtrar documentos, al igual que find
2 do parámetro ({populate…}): son las opciones aplicadas al paginado

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

# Mongoose – Mongoose Paginate

```
var query   = {};
var options = {
    select:   'title date author',
    sort:     { date: -1 },
    populate: 'author',
    lean:     true,
    offset:   20,
    limit:    10
};

Book.paginate(query, options).then(function(result) {
    // ...
});
```

Opciones

- Select: Selección de ciertos atributos
- Sort: Orden en los resultados
- Populate: Aplica el populate en base al modelo especificado
- Offset
- Page
- Limit

# Mongoose – Mongoose Paginate

Se puede encontrar mas información en:
https://www.npmjs.com/package/mongoose-paginate

# DOTENV

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

# Mongoose – DotEnv

Modulo que nos permite manejar con mayor facilidad la configuración de nuestras variables de entorno

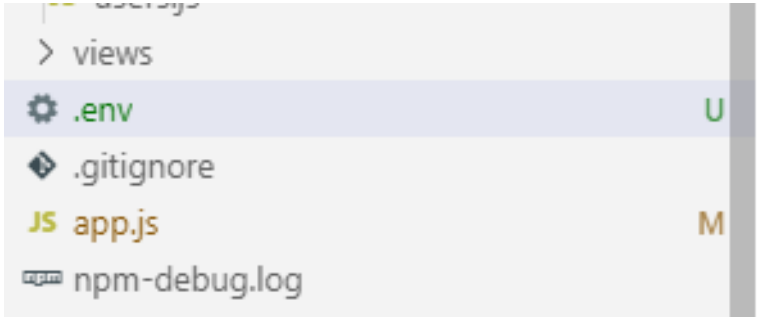
Instalar el modulo ejecutando:

**npm install dotenv**

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

# Mongoose – DotEnv

Crear un archivo llamado .env en la raíz de nuestro proyecto:



# Mongoose – DotEnv

En dicho archivo declarar las variables de entorno:

```
SECRET_KEY=nodeRestApi
MONGO_DB_HOST=localhost
MONGO_DB_DB=startup
MONGO_DB_LIMIT=1
MYSQL_HOST=localhost
MYSQL_DB=pwa
MYSQL_USER=root
MYSQL_PASSWORD=
```

# Mongoose – DotEnv

Hacer uso de dichas variables, por ejemplo para la definición del secretKey en el app.js

```
//Definicion de secretKey
app.set('secretKey', process.env.SECRET_KEY); // jwt secret token
```

*A la variable definida se le antepone **process.env.***

UTN.BA

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

# Nodemailer

# Mongoose – Nodemailer

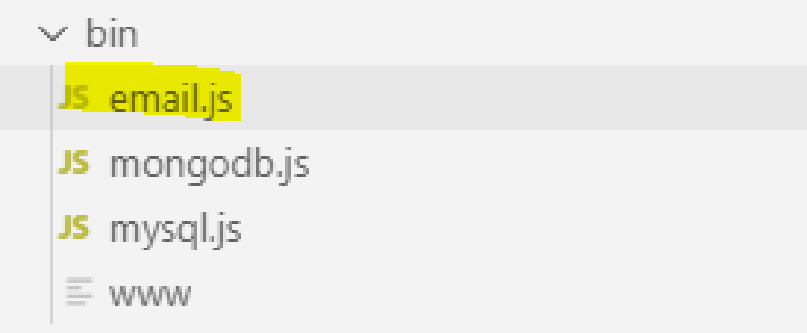
Nodemailer es un modulo utilizado para node que nos permite realizar el envío de emails.

Para instalar ejecutamos:

**Npm install nodemailer**

# Mongoose – Nodemailer

Luego creamos el archivo email.js dentro del directorio bin



UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

# Mongoose – Nodemailer

Bin/email.js

```
const nodemailer = require('nodemailer');

let transporter = nodemailer.createTransport({
    service: 'gmail',
    auth: {
        user: process.env.EMAIL_USER, // generated ethereal user
        pass: process.env.EMAIL_PASS // generated ethereal password
    },
    tls: {
        rejectUnauthorized: false
    }
});

module.exports = transporter;
```

*En caso de configurar una cuenta de otro cliente distinto a Gmail, se debe hacer de la siguiente forma*

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

# Mongoose – Nodemailer

Bin/email.js

```
let transporter = nodemailer.createTransport({
    host: 'smtp.ethereal.email',
    port: 587,
    secure: false, // true for 465, false for other ports
    auth: {
        user: testAccount.user, // generated ethereal user
        pass: testAccount.pass // generated ethereal password
    }
});
```

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

# Mongoose – Nodemailer

## controller/autentication.js

```
var transporter = require("../bin/email")
const jwt = require('jsonwebtoken');

module.exports = {
  save: async function(req, res, next) {
    try{
      var data = await autenticationModel.create({ name: req.body.name, usuario: req.body
      let info = await transporter.sendMail({
          from: process.env.EMAIL_USER, // sender address
          to: req.body.email, // list of receivers
          subject: 'Bienvenido '+req.body.name, // Subject line
          text: 'Bienvenido a este sitio', // plain text body
          html: '<b>Bienvenido a este sitio</b>' // html body
      });
      res.json({status: "success", message: "User added successfully!!!", data: data});
    }catch(err){
      console.log(err)
      next(err);
    }

  },
```

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

# Mongoose – Nodemailer

Para ver mas acerca de nodemailer:
https://nodemailer.com/

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

# PDF

# Mongoose – PDF

El modulo que vamos a utilizar es Express-pdf. El mismo permite generar un pdf en base a un html.

Instalar el modulo:
**npm install --save express-pdf**

# Mongoose – PDF

En el app.js incluir el modulo y definir un middleware con el mismo

```
var pdf = require('express-pdf');
require('dotenv').config()

var usersRouter = require('./routes/users');
var productosRouter = require('./routes/productos');
var autenticacionRouter = require('./routes/autentication');

var app = express();

app.use(pdf);
```

# Mongoose – PDF

En el app.js incluir el modulo y definir un middleware con el mismo

```
var pdf = require('express-pdf');
require('dotenv').config()

var usersRouter = require('./routes/users');
var productosRouter = require('./routes/productos');
var autenticacionRouter = require('./routes/autentication');

var app = express();

app.use(pdf);
```

# Mongoose – PDF

Para generar el pdf utilizaremos el método **pdfFromHTML**
En el siguiente ejemplo vemos como generar el pdf utilizando una vista en ejs

```
pdf: async function(req, res, next) {
 try{

        var id = req.params.productId;
        var data = await productModel.findById(id);

        res.render('index',{title:data["name"]},function(err,html){
            res.pdfFromHTML({
                filename: 'generated.pdf',
                htmlContent: html
            });
        })
    }catch(err){
        next(err)
    }
}
```

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

# Mongoose – PDF

Para ver mas acerca de este modulo, consultar:
https://www.npmjs.com/package/express-pdf