

## UG-InternetApps - P2 - Web Service

# Diseño de una API RESTFUL

**Objetivo: Diseño de un webservice para administrar los trabajadores de una empresa**

| MongoDB | + | Express | + | Angular | + | Node |

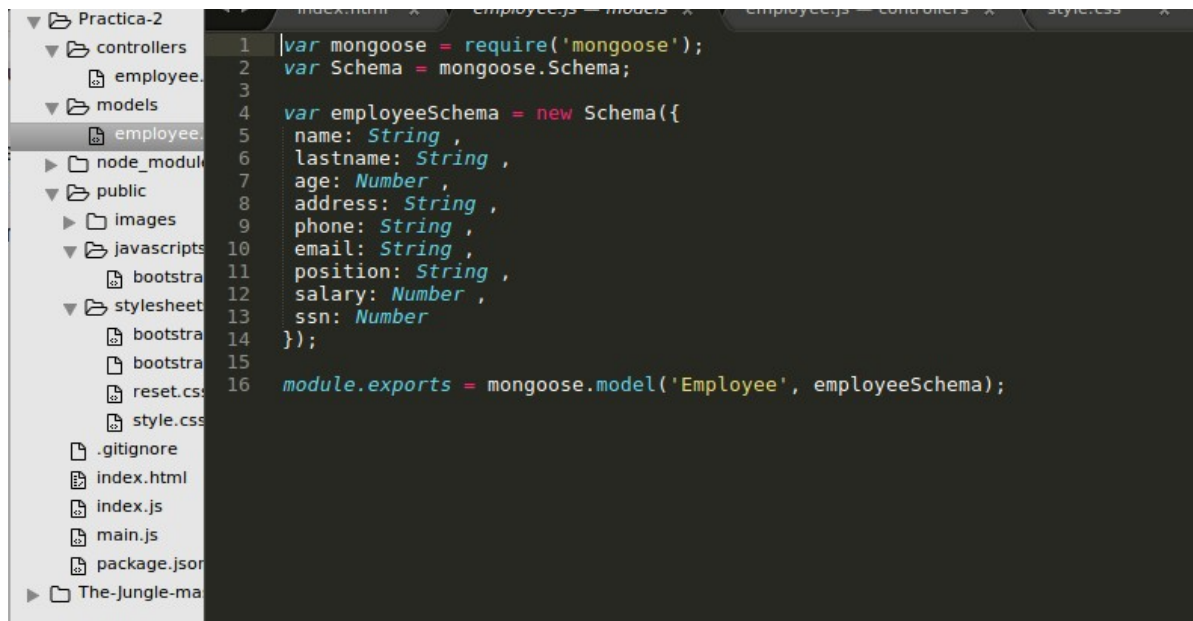
El web service implementado aplica las tecnologías anteriores en una interfaz que permite manipular los empleados de una empresa, con las operaciones características de una base de datos, crear, editar, filtrar y agregar.

Desde ese punto de vista el primer punto del proyecto es implementar mongoDB como nuestro sistema de base de datos, y la herramienta mongoose para gestionar mas facilmente.

Bien, entonces se creo una colección "employees" para almacenar todos los documentos generados, y se vinculo dentro de expressjs

El siguiente paso consiste en crear un modelo, el documento con los atributo de informacion requerida:

- name
- lastname
- email
- phone
- age
- address
- position
- ssn



Este modelo se exporta para ser manipulado desde los controladores, cada controlador de la api, utiliza un metodo http para controlar la información ( objetos json) enviada o recibida

Los controladores implementados son

- FindAll
- Add
- Delete
- FindById
- Update(edit)

Cada uno toma la ruta correspondiente para aplicar el metodo predefinido, en el caso de finall y find id, ambos usan el metodo get, pero en diferentes rutas

```
exports.update = function(req, res) {
  Employee.findById(req.params.id, function(err, employee) {
    employee.name = req.body.name;
    employee.lastname = req.body.lastname;
    employee.age = req.body.age;
    employee.address = req.body.address;
    employee.phone = req.body.phone;
    employee.email = req.body.email;
    employee.position = req.body.position;
    employee.salary = req.body.salary;
    employee.ssn = req.body.ssn;
    employee.save(function(error) {
      if(error) return res.send(500, err.message);
      res.status(200).jsonp(employee);
    });
  });
};

exports.delete = function(req, res) {
  Employee.findById(req.params.id, function(err, employee) {
    employee.remove(function(err) {
      if(err) return res.send(500, err.message);
      res.json({ message: 'Successfully deleted' });
    });
  });
};
```

Cada metodo es declarado en el archivo index.js para poder usarse posteriormente  
-Finalmente la parte front-end de la aplicación consiste el una sencilla interfaz que permite la manipulacion de los metodos por medio de tablas, y formularios

## NJ|E|M|A Employee List !

Firstname	Lastname	Email	Position	ssn
Ignacio Michel	Aguilera Santamarina	ddan.m@gmail.com	developer	5464
Rauslideded	Jlmenez	rauij.fef@hotmail.com	delantero	36503684

### Add more employees

Al integrar AngularJS se permite la gestión e integración de los controladores dentro del código HTML del archivo index.html, se establecen los controladores para cada función de archivo de controladores que permite mediante el uso de algunas directivas manejar el control y el flujo de la información a través de todo el documento.

A screenshot of a code editor with four tabs: 'index.html', 'employee.js — models', 'employee.js — controllers', and 'style.css'. The 'employee.js — controllers' tab is active, displaying the following JavaScript code:

```
$scope.createEmployee = function(){
  $http.post('/api/employees', $scope.formData)
  .success(function(data) {
    $scope.formData = {};
    getEmployees();
  })
  .error(function(data) {
    console.log('Error:' + data);
  });
};

// Delete Employee

$scope.deleteEmployee = function(id) {
  $http.delete('/api/employees/' + id)
  .success(function(data) {
    getEmployees();
  })
  .error(function(data) {
    console.log('Error:' + data);
  });
};

$scope.findEmployee = function(id){
  $http.get('/api/employees/' + id)
  .success(function(data){
    $scope.femployee = data;
    console.log(data)
  })
  .error(function(data){
    console.log('Error: ' + data);
  })
}

$scope.editEmployee = function(id){
  $http.put('api/employees/' + id, $scope.femployee)
  .success(function(data){
    getEmployees();
    $scope.formData = {};
  })
  .error(function(data){
    console.log('Error', + data);
  })
}

function getEmployees(){
```

El objeto formData es usado como un array para la información procedente de los formularios, mientras que el objeto femployee especifica la información devuelta tras hacer uso de la directiva ng-click editEmployee.