Inverspot API

Version 0.1.0

Inverspot API es una interfaz ligera para la gestión de Inversiones, Usuarios y Propiedades a traves de sus diversas aplicaciones.

Iniciando.

Inverspot está configurado para un inicio rápido y sección de configuraciones generales, además está diseñado para implementarse en un contenedor para un deploy más fiable con Docker.

Requisitos previos

- git
- nodejs
- npm
- mongodb

Es recomendable tener las siguientes versiones, o versiones superiores.

```
node -v
#v6.9.5
```

> *El API se desarrollada en esta versión.*

Nos aseguramos que mongo este iniciado

```
sudo service mongod start
```

Instalación

Clonamos el repositorio en nuestra zona de trabajo.

```
git clone **.git
```

accedemos a la carpeta para instalar los módulos e iniciar el servidor.

```
cd api_node_docker/Inverspot_API
npm start
#Server running on port 8080
#--> Database coneccted
```

Ahora vamos con un navegador hacemos una petición a *http://localhost:8080/*, o en el puerto que nos indique tras iniciar el servidor. Deberá mostrar un mensaje *"Bienvenido a Inverspot API"*

> *http://localhost:8080/apidoc/ muestra la documentación de los endpoints de la API.*

Documentación

Estructura de Archivos

```
|-Inverspot_API
|  |-apidoc
|  |-config
|  |-src
|  |  |-api
|  |  |  |-auth
|  |  |  |-builder
|  |  |  |-investment
|  |  |  |-property
|  |  |  |-user
|  |  |  |-work-progress
|  |  |-filemanager
|  |  |-mailing
|  |  |  |-templates
|  |-upload
```

Configuraciones

El servidor está diseñado para tener la mayoría de configuraciones agrupadas en la carpeta **Inverspot_API/config**

```
|-config
|  |-auth.json
|  |-db.json
|  |-facebook.json
|  |-index.js
|  |-mailing.json
|  |-server.json
```

El archivo index.js sirve para exportar las configuraciones que están en .JSON al servidor.

Auth

En el JSON de auth podemos modificar la palabra secreta para encriptar el Json Web Token (JWT).

```
{
  "secret": "inverspot"
}
```

Base de Datos

En las configuraciones de la base de datos tenemos los datos necesarios para conectarnos con mongoose a la base de datos mongoDB por medio de URI.

```
{
  "dbCloud":{
    "port"      : 53699,
    "database"  : "inverspot",
    "address"   : "ds153699.mlab.com",
    "user"    : "is_developer",
    "password": "inverspot"
  },
  "dbLocal":{
    "port"      : 27017,
    "database"  : "inverspot",
    "address"   : "127.0.0.1"
  },
  "debug":  true
}
```

> *Tenemos las configuraciones de una BD de prueba en mongoLab, dejamos los datos como ejemplo para conexión con una BD remota.*

para poder conectarse ya sea a una base local o remota basta con descomentar alguna de las dos URI después de haber configurado los datos anteriores

```
//config/index.js

//URI Database local or cloud
// ---- Remota
//db.dbUri =
`mongodb://${db.dbCloud.user}:${db.dbCloud.password}@${db.dbCloud.address}:$
{db.dbCloud.port}/${db.dbCloud.database}`

// ---- Local
db.dbUri =
`mongodb://${db.dbLocal.address}:${db.dbLocal.port}/${db.dbLocal.database}`
```

Facebook

En las configuraciones de facebook podemos ajustar los parámetros de los permisos de la aplicación dada de alta para poder autenticar a los usuarios.

```
{
  "clientID" : "1430245840616178",
  "clientSecret": "4d5c7b426ee7429379c52ecf2e2ed142",
  "callbackURL": "http://192.169.174.96:8080/api/auth/facebook/callback"
}
```

Mailing

Engloba las configuraciones para y variables involucradas en el envío de emails, sobretodo podemos editar los datos de cuenta sobre el mail donde serán enviadas.

```
{
  "connection" : {
    "host"   : "smtp.gmail.com",
    "port"   : 465,
    "secure" : true,
    "auth"   : {
      "user" : "axel.ramiro.92@gmail.com",
      "pass" : "fwdajvasjzpvboqu"
    }
  },
  "from" : "\"Inverspot\" <hola@inverspot.mx>",
  "cc": "\"Contacto\" <villak@startuphidalgo.com>",
  "domain": "http://192.169.174.96/",
  "imagesUrl": "http://192.169.174.96/is-img/",
  "logoImg": "http://192.169.174.96/is-img/9ff1961d-0e06-4dba-928b-
58e1c1771bfd.png",
  "adminUserUrl": "http://192.169.174.96/",
  "recoveryURL": "http://192.169.174.96/recovery/",
  "recoveryUrlAdmin": "http://192.169.174.96/recoveryAdmin/",
  "loginUser": "http://192.169.174.96/user",
  "loginAdmin": "http://192.169.174.96/admin"
}
```

Todas las URL descritas son usadas en los templates para los correos electrónicos, si cambiamos de dominio o direccionamiento basta con ajustarlas para que las urls de todos los correo funcionen correctamente.

Server

En estas configuraciones podemos definir el puerto en el que se ejecuta la API, e que parte del server se guardan las imágenes y archivos que se suben, así como la URL en la que se verá la documentación de la API *dominio:8080/apidoc*

```
{
  "port" : 8080,
  "uploadPath": "/home/jorge/upload/",
  "apidocUrl": "/apidoc"
}
```

Base de Datos

Se ha configurado la API para el uso de una base de datos con MongoDB, debido a su escalabilidad y rapidez. Se utiliza el modulo MongooseJs para lograr un modelado y control correcto de los datos antes de consultar la base de datos.

Se definieron los siguientes modelos de datos:

Users

En la colección de usuarios englobamos a los diversos actores de las aplicaciones de Inverspot clasificados por el campo *"level"* con los siguientes valores:

valor	Descripción
user	Usuario registrado en el sitio web prospecto a invertir.
investor	Usuario que ya ha invertido alguna vez, con datos de inversión capturados.
asesor	Usuario asesor de investor tiene derechos de administrar a los user e investor, crear inversiones, avances de obra y propiedades
admin	Usuario de máximo rango, capaz de modificar y administrar todo.

> *Como usuario y como investor se requiere de una relación con un asesor.*

Para ver el listado de campos, los obligatorios, tipos y valores por defecto lo podemos ver en el modelo de user ubicado en **/Inverspot_API/src/api/user/model.js**

Builder

También llamado desarrolladora o constuctora, estas colecciones tienen un tipo de relación 1:N con las propiedades, y tiene datos muy básicos, su finalidad es ser un catálogo de desarrolladoras.

Para ver el listado de campos y tipo de cada uno se encuentran descritos en el modelo de builder ubicado en **/Inverspot_API/src/api/builder/model.js**

Property

La colección de datos de propiedades no es solo un catálogo de propiedades debido a que tiene campos en constante actualización, con relaciones 1:N con las inversiones y N:1 con las desarrolladoras.

Campos clave:

Campo	tipo	Descripción
builder	ObjectId	Id de la desarrolladora
status	String	Estado de la propiedad iniciando con un estado *"available"* y pasando a *"fund"* cuando el total de participaciones es igual a las participaciones

	vendidas.
dataSheet.sharesSold Number	Las participaciones vendidas son actualizadas constantemente, cuando se ingresa una nueva inversión, se modifica o se elimina

Para ver el listado de campos y tipo de cada uno se encuentran descritos en el modelo de property ubicado en **/Inverspot_API/src/api/property/model.js**

investment

La colección de inversiones mantiene una relación de N:1 con las propiedades y de 1:1 con los usuario (inversores), la finalidad es lograr ese tipo de relación mediante un ticket de inversión, por lo que contiene muy pocos datos.

Para ver el listado de campos y tipo de cada uno se encuentran descritos en el modelo de investment ubicado en **/Inverspot_API/src/api/investment/model.js**

Work progress

También llamado avance de obra es una colección con una relación de N:1 con las propiedades, pretende ser un catálogo de los avances que ha tenido cierta propiedad, por lo que sus campos no modifican o alteran el de otras colecciones.

Para ver el listado de campos y tipo de cada uno se encuentran descritos en el modelo de work-progress ubicado en **/Inverspot_API/src/api/work-progress/model.js**

> *Todos los archivos de imagen manejados en propiedades y avances de obra no son guardados en las base de datos, en los campos de dicha base sólo se almacena una referencia de la imagen, estas son guardadas en el servidor en una carpeta pública.*

Mailing

La función mailing es la encargada de enviar los diversos correos, esta función está echa con el módulo **nodemailer**, quien se encarga de enviar el correo, lo que hace Mailing es simplemente agregar las opciones de envío y el facilitar el contenido del correo. Esto lo logra a través de templates html encapsulados en un string para inyectar variables aprovechando la funcionalidad de *Template Strings* de ECMAScript 2015 (ES6).

sendMail(options, template, params, cb)

Campo	Tipo	Descripción
options	Object	Objeto js, que debe contener correo de destino y asunto (to, subject).

template	String	Nombre del template a ocupar para el contenido del correo.
params	Object	Objeto js, con los parámetros necesarios para el template.
cb	Function	Función callback

Ejemplo

```
// Importa todas las configuraciones de mailing
const config = require('./config')
// Importa la función de mailing
const sendMail = require('./src/mailing')(config)

let options = {
  to: resUser.email,
  subject: `¡Jhon, Te damos la bienvenida a Inverspot!`
}

let template = 'welcome-admin'

let params = {
  email: 'a@a.com',
  password: '123456',
  level: 'asesor'
}

sendMail( options, template, params, console.log)
```

Mailing templates

Los templates actuales se encuentran en */Inverspot_API/src/mailing/templates*

```
|____mailing
| |____templates
| | |____admin_investment.js
| | |____investment.js
| | |____recovery.js
| | |____welcome-admin.js
| | |____welcome.js
```

Estructura:

```
// Importa parámetros y funciones externas, además de las configuraciones
generales de mailing.
module.exports = (params, config, currency) => {
  return `
  <!DOCTYPE html>
  <html>
    <head>
      <meta charset="utf-8">
      <title>${params.title}</title>
    </head>
    <body>
      <h1>Costo Total</h1>
      <b>${currency(params.amount)}</b>
      <a href="${config.mailing.domain}">Paga Ahora!</a>
    </body>
  </html>
  `
}
```

> *currency es una función para dar formato a una variable tipo número como formato moneda.*

Referencias

Official Node.js Docker Image (https://hub.docker.com/_/node/)

Dockerizing a Node.js web app (https://nodejs.org/en/docs/guides/nodejs-docker-webapp/)

Docker Repo (https://github.com/nodejs/docker-node/tree/90d5e3df903b830d039d3fe8f30e3a62395db37e)

Create Docker Image

```
docker build -t [name]  .
docker build -t isapi .
```

> *NOTA: Si vamos a usar el volumen, ejecutar "npm install" localmente antes de todo ya que el volumen reemplaza por completo el directorio de la app*

Docker started

```
docker run -p [Puerto Externo:Puerto Interno] -d --name [Nombre Docker] --link [Nombre Docker db : Nombre interno] -v [Directorio Local:Directorio interno] -t [nombre de la imagen]
docker run -p 8080:8080 -d --name is_api --link is-mongodb:is-mongodb -v ~/Documents/Inverspot/api_node_docker/Inverspot_API:/usr/src/app -t isapi
```

> *NOTA: Usando ALPINE no se puede ejecutar "npm start", por lo que el iniciar el docker se ejecuta la aplicación como "node index.js", esto implica que tenemos que instalar localmente y los módulos necesarios, y el volumen de Docker incluirá (no es lo recomendable) node_modules.*

apidoc

Compilar documentación de la API

```
apidoc -i Inverspot_API/ -o Inverspot_API/apidoc/ -e Inverspot_API/node_modules
```