http.request & http.create server
- Consumiendo imágenes de google.
  - Ejercicio
    - Solucion parcial.

Expressjs
- app.js
- Middleware
  - req.body
    - app2.js
- Ejercicio.

http.request & http.create server

Con el fin de darle un cierre a los temas de http.request y http.server veremos dos ejercicios integradores.

Consumiendo imágenes de google.

El siguiente código fuente genera un servidor web. Por cada request se espera el argumento "query" por GET. El valor del argumento query es pasado a un request hacia google api images, con el el fin de buscar imágenes relacionadas a la query.
Una vez que se obtienen los resultados de google images, los mismo son transformados en json, procesados y utilizados para generar un html para ser enviado al navegador.

```
var http = require('http');

var getImages  = function(search, cb){
    var body = "";
    http.get("http://ajax.googleapis.com/ajax/services/search/images?v=1.0&q="+search,
function(res){
        res.on('data', function(data){
              body +=data;
        });
        res.on('end', function(){
           var json = JSON.parse(body);
           cb(json);
        });
    });
};

http.createServer(function (req, res) {

    (function(url){
        req.query = {};

        if(url.indexOf("?")==-1) return false;

        url = url.replace("?", "");
        url = url.split("/").join("");

        var pars  = url.split("&");

        for(var x=0; x<pars.length; x++){
            var kv = pars[x].split("=");
            req.query[kv[0]] = kv[1];
        }//end for

    })(req.url);
```

```
    var send = function(str){
        console.log("Nuevo pedido: ", req.url);
        res.writeHead(200, {'Content-Type': 'text/html'});
        res.end(str+'\n');

    };

    getImages(req.query.query|| "nodejs", function(json){
        var html = "";
        var images = json.responseData.results;

        for(var x=0; x<images.length; x++){
            var image = images[x];
            html +="<img width='400' height='300' src='"+image.url+"' />";
        }//end for

        send(html);
    });


}).listen(1234);

console.log('Servidor corriendo en el puerto 1234');
```

Ejercicio

- Crear un módulo de usuario llamado "google", el cual en su interior cuenta con un método search, el cual recibe 2 argumentos: keyword y tipo de contenido. Ej: new require("google").search("shakira", "images");
- Dentro del módulo "google.js" implementar el módulo de usuario "images.js", el cual busca imágenes en gogole.
- Crear el módulo de usuario "blogs" el cual busca blogs en google. https://ajax.googleapis.com/ajax/services/search/blogs?v=1.0&q=shakira
- Crear el módulo de usuario "news" el cual busca noticias en google. https://ajax.googleapis.com/ajax/services/search/news?v=1.0&q=shakira

Solucion parcial.

server.js

```
var http = require('http');
var google =  require("./google");

http.createServer(function (req, res) {
```

```
    (function(url){
        req.query = {};

        if(url.indexOf("?")==-1) return false;

        url = url.replace("?", "");
        url = url.split("/").join("");

        var pars  = url.split("&");

        for(var x=0; x<pars.length; x++){
            var kv = pars[x].split("=");
            req.query[kv[0]] = kv[1];
        }//end for

    })(req.url);


    res.send = function(str){
        console.log("Nuevo pedido: ", req.url);
        res.writeHead(200, {'Content-Type': 'text/html'});
        res.end(str+'\n');
    };

       var cbImages = function(json){
           var html = "";
        var images = json.responseData.results;

        for(var x=0; x<images.length; x++){
            var image = images[x];
            html +="<img width='400' height='300' src='"+image.url+"' />";
        }//end for

        res.send(html);
       };


            new google().search(req.query.query|| "nodejs", req.query.type, function(json){
               cache[req.query.query] = json;
                    switch(req.query.type || "images"){
                      case "images": cbImages(json); break;
                    }
            });

}).listen(1234);

console.log('Servidor corriendo en el puerto 1234');
```

google.js

```
module.exports = function(){
   this.search = function(query, type, cb){

   var types = {
        "images": require("./images")
        ,"blogs": require("./blogs")
        ,"news":  require("./news")
   };

   var _ = typeof types[type] == "undefined" ? types.images : types[type];
        new _().search(query, cb);
   };//end search
}//end module
```

images.js

```
var http = require('http');
module.exports = function(){

       this.search  = function(search, cb){
              var body = "";

http.get("http://ajax.googleapis.com/ajax/services/search/images?v=1.0&q="+search,
function(res){
                     res.on('data', function(data){
                            body +=data;
                     });
                     res.on('end', function(){
                        var json = JSON.parse(body);
                        cb(json);
                     });
              });
       };
}
```

Expressjs

Expressjs es un framework para desarrollar aplicaciones web en node.js de forma simple. Es extremadamente potente, rápido y flexible, permitiendo generar soluciones tanto para apps webs como servicios REST.
Una de las grandes ventajas de express es la implementación de middleware.

app.js

app.js es el nombre que suelen recibir por defecto las aplicaciones en express.

```
var express = require('express')
var app = express()

app.get('/', function (req, res) {
    res.send('Hello World!');
    console.log(req.query);
});

app.post("/", function(req, res){
    res.json({data: ["hola, soy el dato"] });
    console.log(req.query);
});

app.get("/news", function(req, res){
    var key = req.query.key || "node.js";
   console.log("Quieren buscar noticias sobre: ", key);
    res.send("buscando...");
});

var server = app.listen(3000, function () {

    var host = server.address().address
    var port = server.address().port

    console.log('Example app listening at http://%s:%s', host, port)

});
```

Middleware

Middleware es un concepto en sistemas que trata básicamente en un intermediario entre dos procesos de una aplicacion (o de diferentes aplicaciones) con el fin de efectuar tareas sobre datos (modificar, extender, etc).
Aplicados en node.js, un middleware es la implementación de funciones en un stack, las cuales se ejecutan una detrás de la otra (cursor) en forma secuencial, alterando (o no) el flujo de ejecución, los objetos transportados, etc.
Por ejemplo, la capacidad de procesar los datos enviados por el método POST en el protocolo HTTP en express se lograr gracias a un middleware.

req.body

En express implementando el módulo "body-parser" podemos obtener un objeto "body" con todas las variables pasadas por POST. El objeto body será una propiedad del objeto "req".

app2.js

```
var app = require('express')();
var bodyParser = require('body-parser');

app.use(bodyParser.urlencoded({ extended: true }));


app.post('/', function (req, res) {
    console.log(req.body);
    res.json(req.body);
});


var server = app.listen(3000, function () {

    var host = server.address().address
    var port = server.address().port

    console.log('Example app listening at http://%s:%s', host, port)

});
```

Ejercicio.

- Pasar el ejercicio anterior a un proyecto express.