# AngularJS tutorial for beginners with NodeJS ExpressJS and MongoDB (Part I)

1. Agregamos el siguiente código en nuestro Index.html

Index.html

<html ng-app="app">

<head>

<title>ngTodo</title>

</head>

<body>

<ng-view></ng-view>

<!-- Libraries -->

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.25/angular.min.js"></script>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.25/angular-route.min.js"></script>

<!-- Template -->

<script type="text/ng-template" id="/todos.html">

Search: <input type="text" ng-model="search.name">

<ul>

<li ng-repeat="todo in todos | filter: search">

<input type="checkbox" ng-model="todo.completed">

<a href="#/{{$index}}">{{todo.name}}</a>

</li>

</ul>

</script>

<script type="text/ng-template" id="/todoDetails.html">

<h1>{{ todo.name }}</h1>

completed: <input type="checkbox" ng-model="todo.completed">

note: <textarea>{{ todo.note }}</textarea>

</script>

<script>

angular.module('app', ['ngRoute'])

//---------------

// Services

//---------------

.factory('Todos', *function*(){

return [

{ name: 'AngularJS Directives', completed: true, note: 'add notes...' },

{ name: 'Data binding', completed: true, note: 'add notes...' },

{ name: '$scope', completed: true, note: 'add notes...' },

{ name: 'Controllers and Modules', completed: true, note: 'add notes...' },

{ name: 'Templates and routes', completed: true, note: 'add notes...' },

{ name: 'Filters and Services', completed: false, note: 'add notes...' },

{ name: 'Get started with Node/ExpressJS', completed: false, note: 'add notes...' },

{ name: 'Setup MongoDB database', completed: false, note: 'add notes...' },

{ name: 'Be awesome!', completed: false, note: 'add notes...' },

];

})

//---------------

// Controllers

//---------------

.controller('TodoController', ['$scope', 'Todos', *function* (*$scope*, *Todos*) {

$scope.todos = Todos;

}])

.controller('TodoDetailCtrl', ['$scope', '$routeParams', 'Todos', *function* (*$scope*, *$routeParams*, *Todos*) {

$scope.todo = Todos[$routeParams.id];

}])

//---------------

// Routes

//---------------

.config(['$routeProvider', *function* (*$routeProvider*) {

$routeProvider

.when('/', {

templateUrl: '/todos.html',

controller: 'TodoController'

})

.when('/:id', {

templateUrl: '/todoDetails.html',

controller: 'TodoDetailCtrl'

});

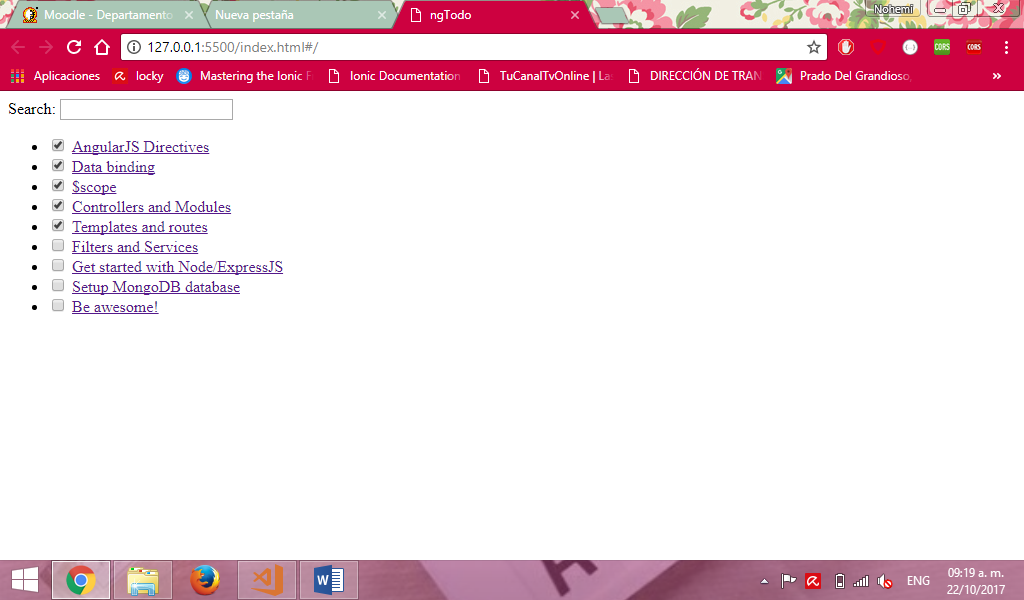
}]);

</script>3

</body>

</html>

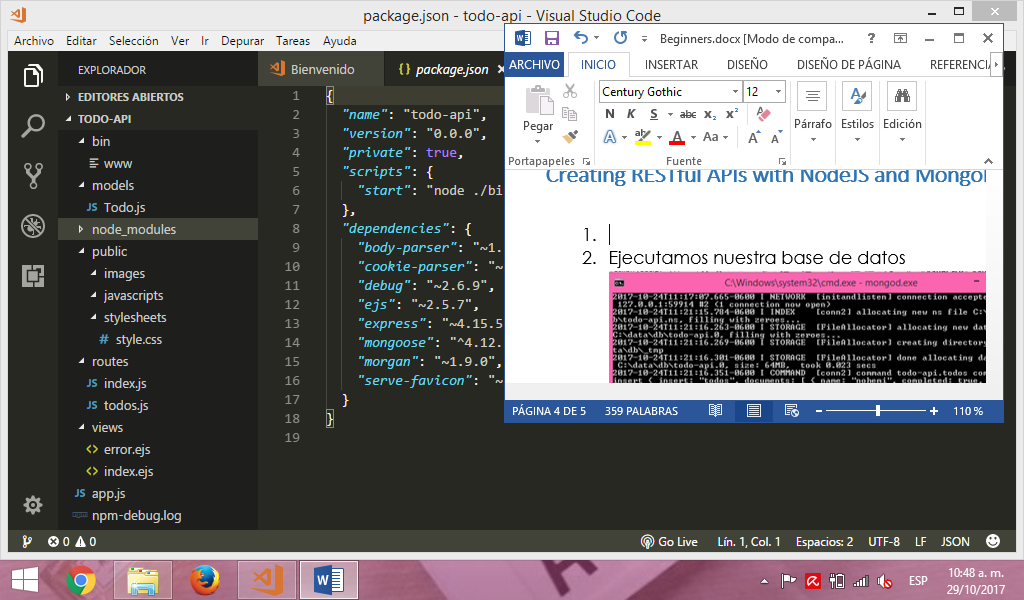
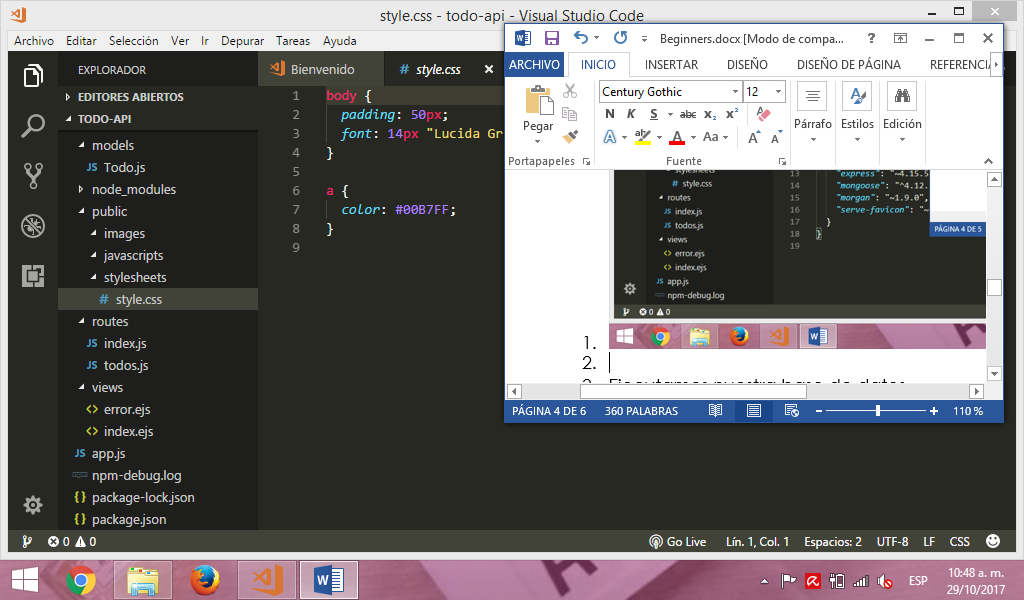
1. Damos clic derecho en el index.html y ejecutamos código. Vamos a nuestro explorador en 127.0.0.1:5500/index.html#/ y podremos ver el siguiente resultado.



Fuente: <http://adrianmejia.com/blog/2014/09/28/angularjs-tutorial-for-beginners-with-nodejs-expressjs-and-mongodb/>

# Creating RESTful APIs with NodeJS and MongoDB Tutorial (Part II)

1. Debemos contar con la siguiente jerarquía de archivos

1. En bin>www agregamos el siguiente código

#!/usr/bin/env node

/\*\*

\* Module dependencies.

\*/

*var* app = require('../app');

*var* debug = require('debug')('todo-api:server');

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

/\*\*

\* Get port from environment and store in Express.

\*/

*var* port = normalizePort(process.env.PORT || '3000');

app.set('port', port);

/\*\*

\* Create HTTP server.

\*/

*var* server = http.createServer(app);

/\*\*

\* Listen on provided port, on all network interfaces.

\*/

server.listen(port);

server.on('error', onError);

server.on('listening', onListening);

/\*\*

\* Normalize a port into a number, string, or false.

\*/

*function* normalizePort(*val*) {

*var* port = parseInt(val, 10);

if (isNaN(port)) {

// named pipe

return val;

}

if (port >= 0) {

// port number

return port;

}

return false;

}

/\*\*

\* Event listener for HTTP server "error" event.

\*/

*function* onError(*error*) {

if (error.syscall !== 'listen') {

throw error;

}

*var* bind = typeof port === 'string'

? 'Pipe ' + port

: 'Port ' + port;

// handle specific listen errors with friendly messages

switch (error.code) {

case 'EACCES':

*console*.error(bind + ' requires elevated privileges');

process.exit(1);

break;

case 'EADDRINUSE':

*console*.error(bind + ' is already in use');

process.exit(1);

break;

default:

throw error;

}

}

/\*\*

\* Event listener for HTTP server "listening" event.

\*/

*function* onListening() {

*var* addr = server.address();

*var* bind = typeof addr === 'string'

? 'pipe ' + addr

: 'port ' + addr.port;

debug('Listening on ' + bind);

}

1. En models>todo.js agregamos el siguiente código

*var* mongoose = require('mongoose');

*var* TodoSchema = new mongoose.Schema({

name: *String*,

completed: *Boolean*,

note: *String*,

updated\_at: { type: *Date*, default: *Date*.now },

});

*module*.*exports* = mongoose.model('Todo', TodoSchema);

1. En public>stylesheets>style.css agregamos el siguiente código

body {

*padding*: 50px;

*font*: 14px "Lucida Grande", Helvetica, Arial, sans-serif;

}

a {

*color*: #00B7FF;

}

1. En routes>index.js agregamos el siguiente código

*var* express = require('express');

*var* router = express.Router();

/\* GET home page. \*/

router.get('/', *function*(*req*, *res*, *next*) {

res.render('index', { title: 'Express' });

});

*module*.*exports* = router;

1. En routes>todos.js agregamos el siguiente código

*var* express = require('express');

*var* router = express.Router();

*var* mongoose = require('mongoose');

*var* Todo = require('../models/Todo.js');

/\* GET /todos listing. \*/

router.get('/', *function*(*req*, *res*, *next*) {

Todo.find(*function* (*err*, *todos*) {

if (err) return next(err);

res.json(todos);

});

});

/\* GET /todos/id \*/

router.get('/:id', *function*(*req*, *res*, *next*) {

Todo.findById(req.params.id, *function* (*err*, *post*) {

if (err) return next(err);

res.json(post);

});

});

/\* PUT /todos/:id \*/

router.put('/:id', *function*(*req*, *res*, *next*) {

Todo.findByIdAndUpdate(req.params.id, req.body, *function* (*err*, *post*) {

if (err) return next(err);

res.json(post);

});

});

/\* DELETE /todos/:id \*/

router.delete('/:id', *function*(*req*, *res*, *next*) {

Todo.findByIdAndRemove(req.params.id, req.body, *function* (*err*, *post*) {

if (err) return next(err);

res.json(post);

});

});

/\* POST /todos \*/

router.post('/', *function*(*req*, *res*, *next*) {

Todo.create(req.body, *function* (*err*, *post*) {

if (err) return next(err);

res.json(post);

});

});

*module*.*exports* = router;

1. En views>error.ejs agregamos el siguiente código

<h1><%= message %></h1>

<h2><%= error.status %></h2>

<pre><%= error.stack %></pre>

1. En views>index.ejs agregamos el siguiente código

<!DOCTYPE html>

<html>

<head>

<title><%= title %></title>

<link rel='stylesheet' href='/stylesheets/style.css' />

</head>

<body>

<h1><%= title %></h1>

<p>Welcome to <%= title %></p>

</body>

</html>

1. En >app.js agregamos el siguiente código

*var* express = require('express');

*var* path = require('path');

*var* favicon = require('serve-favicon');

*var* logger = require('morgan');

*var* cookieParser = require('cookie-parser');

*var* bodyParser = require('body-parser');

*var* index = require('./routes/index');

*var* todos = require('./routes/todos');

*var* app = express();

// load mongoose package

*var* mongoose = require('mongoose');

// Use native Node promises

mongoose.Promise = global.Promise;

// connect to MongoDB

mongoose.connect('mongodb://localhost/todo-api')

.then(() *=>* *console*.log('connection succesful'))

.catch((*err*) *=>* *console*.error(err));

// view engine setup

app.set('views', path.join(\_\_dirname, 'views'));

app.set('view engine', 'ejs');

// uncomment after placing your favicon in /public

//app.use(favicon(path.join(\_\_dirname, 'public', 'favicon.ico')));

app.use(logger('dev'));

app.use(bodyParser.json());

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

app.use(cookieParser());

app.use(express.static(path.join(\_\_dirname, 'public')));

app.use('/', index);

app.use('/todos', todos);

// catch 404 and forward to error handler

app.use(*function*(*req*, *res*, *next*) {

*var* err = new Error('Not Found');

err.status = 404;

next(err);

});

// error handler

app.use(*function*(*err*, *req*, *res*, *next*) {

// set locals, only providing error in development

res.locals.message = err.message;

res.locals.error = req.app.get('env') === 'development' ? err : {};

// render the error page

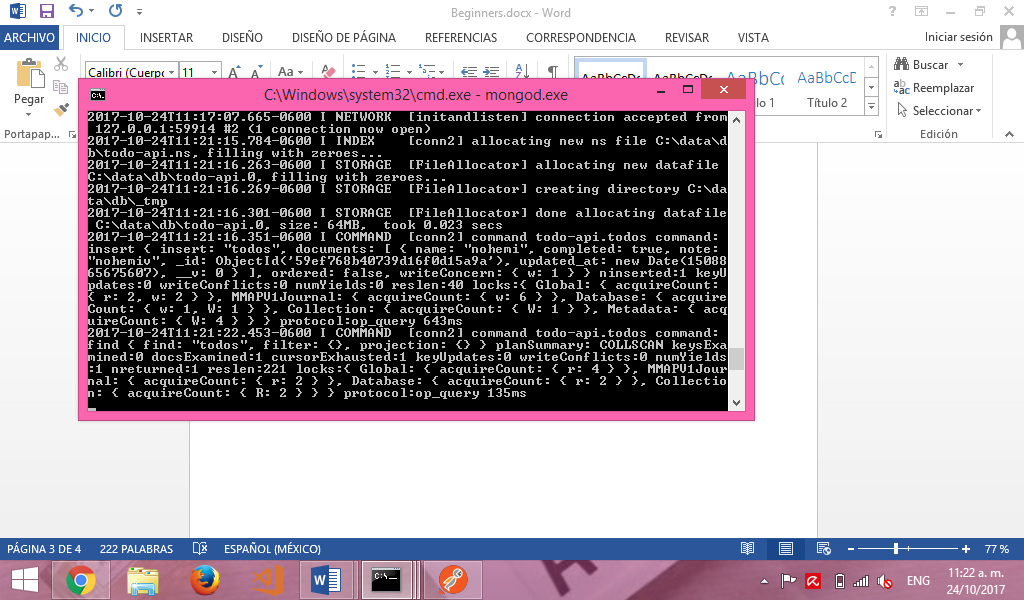
res.status(err.status || 500);

res.render('error');

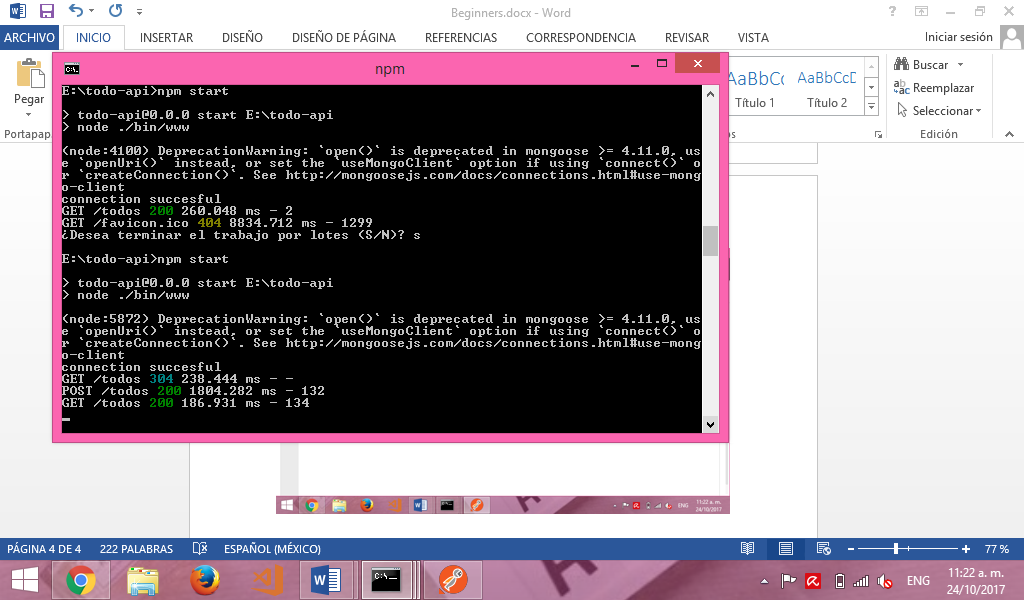
});

*module*.*exports* = app;

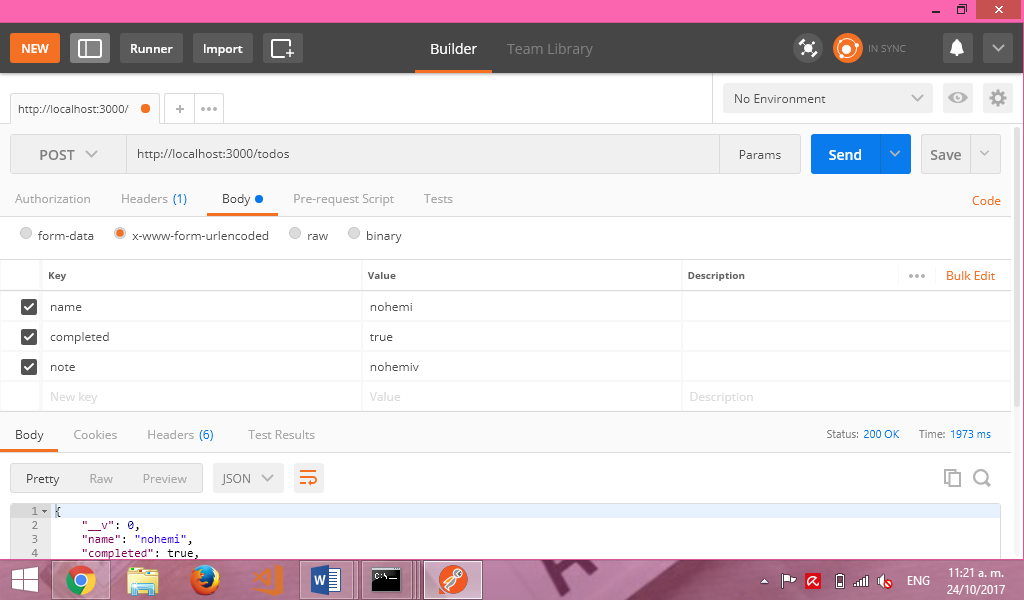
1. Ejecutamos nuestra base de datos



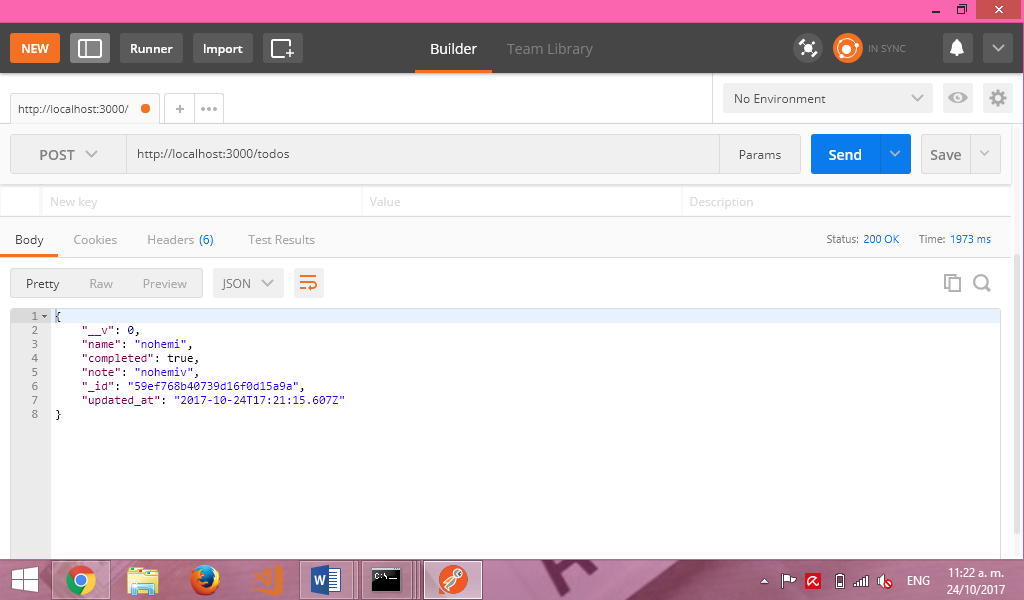
1. En la carpeta que contiene nuestro proyecto, ejecutamos el comando npm start



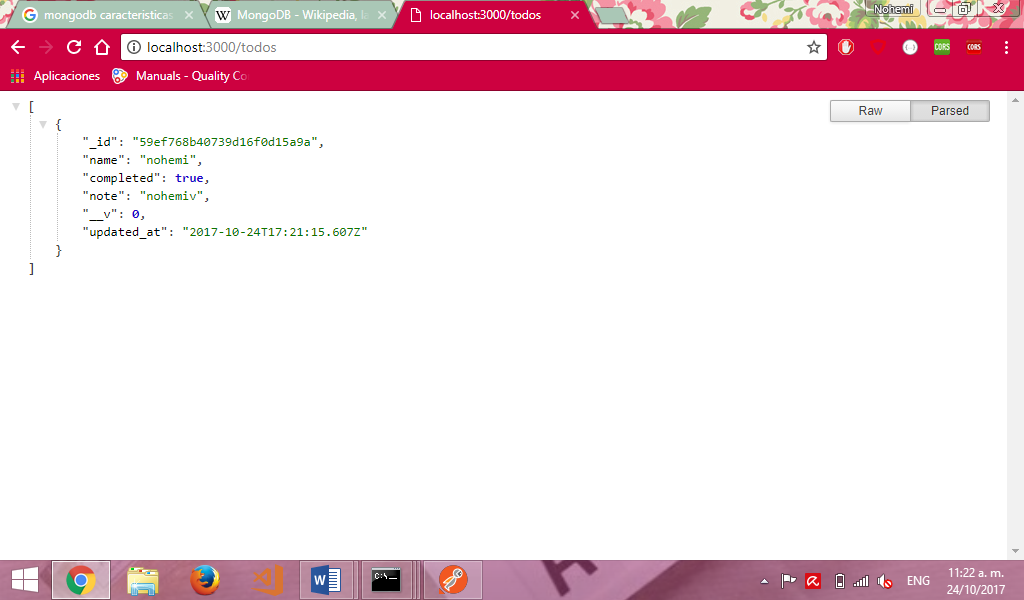
1. En postman, agregamos un nuevo elemento por medio del método POST. Agregamos lo siguiente:



1. Damos clic en ‘Send’ y nos debiera de mostrar el siguiente resultado en la misma página de Postman, en la parte inferior, justo debajo de lo observado en el paso anterior.



1. Vamos a nuestro explorador en localhost:3000/todos y podremos observar el elemento que acabamos de agregar.



Fuente: <http://adrianmejia.com/blog/2014/10/01/creating-a-restful-api-tutorial-with-nodejs-and-mongodb/>