

Guía para el Desarrollo de Aplicaciones Web Responsivas Utilizando el Stack MEAN

Héctor Andrade ¹¹ and Victor Reboloso ²²

¹Departamento de Ciencias y Computación, \LaTeX University

²Estudiante de ingeniería en sistemas computacionales, \LaTeX
University

15 de noviembre de 2016

¹handrade@itver.edu.mx

²vrebo.deg@gmail.com

Resumen

El presente documento pretende ser una guía para el desarrollo de aplicaciones web utilizando el stack MEAN (MongoDB, Express, AngularJS, Node.js). Se incluye una breve introducción a cada una de estas tecnologías. El documento contiene también una guía para la instalación de las herramientas que conforman este stack así como un ejemplo completo de un sistema de administración de bibliotecas, el cual está accesible para su descarga. Este sistema está formado a su vez por dos sub-sistemas el "back-end" (programación en el lado del servidor) y el "front-end" (programación en el lado del cliente).

Índice general

1. Desarrollo de Aplicaciones Web Responsivas	2
1.1. Introducción	2
1.2. Patrón Arquitectónico MVC	2
1.3. Aplicaciones de Página Única	3
1.4. HTML	4
1.5. CSS	4
1.5.1. Bootstrap	5
1.6. JavaScript	5
1.6.1. JQuery	6
1.7. Stack de Desarrollo MEAN	6
1.7.1. MongoDB	7
1.7.2. Express	7
1.7.3. AngularJS	8
1.7.4. Node.js	8
2. Instalación de las Herramientas	9
2.1. Node.js	9
2.2. MongoDB	10
2.3. Express y Mongoose	12
2.4. Bootstrap, JQuery, AngularJS	13
3. Desarrollo de una aplicación MEAN a través de servicios web REST	15
3.1. Descripción de la aplicación	15
3.2. Back-end	15
3.2.1. API REST	15
3.2.2. Esquema de la Base de Datos	17
3.2.3. Estructura de la Aplicación	18
3.3. Front-end	19
3.3.1. Estructura de la Aplicación	19
3.3.2. Conexión con el Back-end	20
3.4. Instalación de la aplicación	20
3.4.1. Descarga	20
3.4.2. Configuración	21
3.4.3. Arranque de los Sistemas	21

Capítulo 1

Desarrollo de Aplicaciones Web Responsivas

1.1. Introducción

Es indudable que el uso de dispositivos móviles para la navegación en la web ha crecido de manera explosiva recientemente, y que esta tendencia continuará en los próximos años. El desarrollo de aplicaciones web que se adapten a este tipo de dispositivos es prácticamente una obligación para las empresas que desean de alguna forma interactuar con clientes o con el público en general a través de la web.

El diseño web responsivo o adaptivo como algunos autores lo llaman, consiste en diseñar y desarrollar aplicaciones para la web que puedan adaptarse a diferentes tipos de dispositivos desde pequeños dispositivos móviles hasta computadoras con monitores de gran tamaño.

El diseño responsivo no solo trata de ajustar la interface del usuario al tamaño de la pantalla. Algunos aspectos a considerar son también el comportamiento del usuario, la plataforma y tipo del dispositivo móvil e incluso la orientación.

La implementación de un diseño responsivo consiste en el uso adecuado de tecnologías tales como HTML, CSS y marcos de trabajo como Bootstrap y Foundation entre otros.

En esta guía mostraremos los conceptos, patrones arquitectónicos y algunas de las tecnologías y herramientas emergentes más utilizadas en el desarrollo de aplicaciones web responsivas. El primer concepto que veremos es el del patrón arquitectónico Modelo Vista Controlador.

1.2. Patrón Arquitectónico MVC

Un patrón arquitectónico es una descripción abstracta de una arquitectura de software que ha sido utilizada exitosamente y que es producto de buenas prácticas de desarrollo de software. En esencia, el patrón de diseño Modelo Vista Controlador (MVC) consiste en dividir conceptualmente la arquitectura de un sistema y, en la medida de lo posible, la implementación de la misma

en componentes que pertenezcan a una de tres categorías, Modelo, Vista y Controlador.

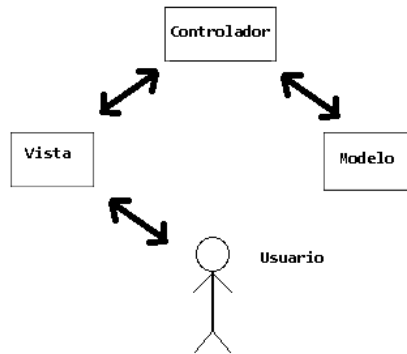


Figura 1.1: Esquema de la relación de MVC.

Dentro de la categoría de Modelo se ubican aquellos componentes encargados de: encapsular los datos utilizados en el sistema, representar el comportamiento de las entidades involucradas en los procesos de negocio y realizar tareas de persistencia de datos.

La categoría de Vista contiene los componentes con los que el usuario interactúa directamente, comúnmente también se les refiere como el front-end del sistema. Es importante notar la independencia entre la vista y el modelo. Modificaciones en los componentes que pertenecen a la vista no necesariamente implican modificaciones en el modelo y viceversa.

Por último, los componentes pertenecientes a la categoría Controlador se encargan de atender las peticiones del usuario solicitadas a través de los componentes de la Vista haciendo uso de los componentes del Modelo. Los controladores son el puente entre el Modelo y la Vistas del sistema.

Algunos de los beneficios de desarrollar un sistema bajo el patrón MVC son:

- Simplifica el desarrollo brindando un desglose su arquitectura.
- Separa las distintas responsabilidades (lógica de negocios, interfaz de usuario) de sus componentes.
- Facilita su mantenibilidad.

1.3. Aplicaciones de Página Única

Las aplicaciones de página única son aplicaciones web que cargan sus recursos (hojas de estilo, scripts, y HTML) desde el servidor una sola vez, y que en respuesta a la interacción del usuario obtienen datos o fragmentos de HTML mediante AJAX o técnicas similares y los presentan al usuario evitando volver a cargar la página entera. Este tipo de aplicaciones web mejora la experiencia de usuario acortando los tiempos de espera para interactuar con el front-end.

Desarrollar una SPA (del inglés Single-Page Application) implica mover hacia el cliente parte de la lógica que usualmente se contenía en el servidor para el renderizado de las páginas. Por lo anterior, es JavaScript el lenguaje encargado de lidiar con las nuevas responsabilidades delegadas.

Existen varios marcos de trabajo de JavaScript para el desarrollo de SPAs.

- AngularJS
- Backbone.js
- Ember.js
- React.js

Esta guía contiene AngularJS, más adelante se incluye un breve explicación de este framework.

1.4. HTML

El lenguaje de marcado de hipertexto, es un lenguaje basado en etiquetas utilizado para definir la estructura de páginas web y describir su contenido. Las estructuras de documentos HTML siguen una secuencia y una jerarquía, esto significa que un elemento puede seguir después de otro y que un elemento puede contener a otro.

Como todo lenguaje, HTML tiene reglas sintácticas y semánticas las cuales varían dependiendo de la versión en cuestión. La versión más reciente es HTML5.

Para profundizar en el tema se recomienda la siguiente lista de referencias:

- Especificación de HTML5 por la World Wide Web Consortium (W3C) ¹.
- Especificación de HTML5 por la Web Hypertext Application Technology Working Group (WHATWG) ².
- Excelente tutorial y sitio de referencia ³.

1.5. CSS

Las hojas de estilo en cascada (de su traducción del inglés Cascading Style Sheets) son un mecanismo que permite añadir estilo a los documentos web. Su uso delega la responsabilidad de definir la presentación de los documentos a los archivos CSS y se evita la práctica poco recomendable de añadir estilos mediante los elementos del documento.

CSS es un lenguaje basado reglas, las cuales describen al navegador como presentar los elementos de HTML. A diferencia de otros lenguajes, CSS no tiene versiones, en su lugar tiene niveles siendo el último el nivel 3.

Enlaces de referencia acerca de CSS:

¹www.w3.org/TR/html5/

²html.spec.whatwg.org/multipage/

³www.w3schools.com/html/

- Documento de la W3C recopilatorio de las especificaciones que definen CSS.⁴.
- Sitio web del grupo de desarrollo de CSS⁵.
- Tutoriales de CSS⁶ ⁷.

1.5.1. Bootstrap

Bootstrap es un framework para el front-end que facilita el desarrollo de páginas web responsivas. Tuvo su origen en un proyecto de Twitter cuyo objetivo era crear una herramienta que redujera el tiempo de desarrollo y diera consistencia al diseño de las interfaces de usuario. La primera versión fue liberada en el 2011 y hasta la fecha son cuatro las versiones del framework siendo la tercera la última estable.

El framework contiene clases de CSS para los elementos básicos de HTML como botones, listas, tablas e imágenes; también tiene clases para crear componentes complejos como dropdowns, ventanas modales, paneles, grupos de botones y barras de navegación. Además se incluyen plugins de JavaScript para manipular sus componentes.

Bootstrap utiliza un sistema de cuadrículas para definir el espacio, la ubicación y como se re-acomodan los componentes del front-end en diferentes tamaños de pantallas.

Referencias sobre Bootstrap.

- Página Web del proyecto⁸.
- Repositorio del proyecto⁹.
- Tutorial del framework¹⁰.

1.6. JavaScript

JavaScript es un lenguaje de programación interpretado y multiparadigma que originalmente sólo se ejecuta en el navegador web (en la actualidad es un lenguaje de programación de uso general que se puede ejecutar independiente al navegador). Sus características de lenguaje funcional, basado en prototipos y orientado a objetos son de gran utilidad para el desarrollo web. Con JavaScript se puede dar comportamiento a las páginas HTML, manipulando su contenido, estructura y estilos. La manipulación es posible debido a la implementación del Document Object Model en el lenguaje.

Originalmente el JavaScript nació como un lenguaje de programación, incluso con diferente nombre, pero con el tiempo se adoptó como un estándar. Actualmente no existe una sola implementación del lenguaje JavaScript sino

⁴www.w3.org/TR/css-2015/

⁵www.w3.org/Style/CSS/

⁶www.w3.org/Style/CSS/learning

⁷www.w3schools.com/css/

⁸<http://getbootstrap.com/getting-started/>

⁹<https://github.com/twbs/bootstrap>

¹⁰<http://www.w3schools.com/bootstrap>

que cada navegador se encarga de implementar su propio dialecto basado en el estándar ECMA-262, pero por convención a todas esas implementaciones se les refiere como JavaScript.

La última versión estable del estándar es la ECMA Script 6 pero la versión con mayor compatibilidad en los navegadores es ECMA Script 5. Actualmente se está trabajando en la versión 7.

A continuación se da una lista de referencias sobre JavaScript.

- Página Web del estándar ¹¹.
- Tabla de compatibilidad del lenguaje según su versión ¹².
- Tutorial del lenguaje. ¹³.

1.6.1. JQuery

Es una librería de JavaScript que facilita tareas comunes de la manipulación del DOM. Su función selectora de elementos es la parte más importante de la librería; permite obtener elementos del árbol del documento por su etiqueta, identificador, clases, atributos e incluso combinar estos criterios de selección en una sola consulta. Además, incluye funciones para el manejo de clases, de modificación del DOM y de peticiones asíncronas (AJAX) a servidores HTTP.

Más sobre JQuery.

- Página Web de la librería ¹⁴.
- Sitio web tutorial oficial de JQuery ¹⁵.
- Tutorial de la W3Schools ¹⁶.

1.7. Stack de Desarrollo MEAN

El stack de desarrollo MEAN es un conjunto de tecnologías de código abierto que permite construir aplicaciones web basadas en el lenguaje JavaScript. Los componentes de este stack son los siguientes:

MongoDB: Base de datos NoSQL orientada a documentos.

Express.js: Permite el desarrollo de la programación de una aplicación web en el lado del servidor (Back-End) a través de Interfaces de aplicación (APIs).

Angular.js: Permite la implementación del patrón arquitectónico modelo vista controlador en la programación de los componentes de la Vista.

Node.js: Provee el ambiente de ejecución para JavaScript en el lado del servidor.

¹¹<http://www.ecma-international.org/publications/standards/Ecma-262.htm>

¹²<http://kangax.github.io/compat-table/es6/>

¹³<http://www.w3schools.com/js>

¹⁴<http://jquery.com>

¹⁵<https://learn.jquery.com>

¹⁶<http://www.w3schools.com/jquery/>

En la figura 1.7 podemos ver la relación entre estas tecnologías. Como puede observarse, el usuario interactúa con las vistas del sistema a través de AngularJS. Las vistas a su vez realizan peticiones al servidor a través de llamadas provistas por AngularJS. El servidor recibe dichas peticiones y realiza las consultas pertinentes al manejador de base de datos de MongoDB el cual responde a las peticiones y el servidor express se encarga de conformar las respuestas a dichas peticiones y enviárselas a la vista provista por AngularJS.



1.7.1. MongoDB

MongoDB ¹⁷ es la capa inferior del stack, la base en la que se apoyan el resto de capas. En MongoDB, lo equivalente a una tabla de un gestor SQL es una colección y en lugar de almacenar datos en registros los almacena como "documentos".^{en} un formato llamado BSON (Binary JSON) por ello se dice que está orientada a documentos.

Las principales ventajas de MongoDB sobre los RDBMS son la flexibilidad de los esquemas de las colecciones y la escalabilidad que dicha flexibilidad implica. Los documentos no están restringidos a poseer una misma estructura, pueden diferir en cantidad y tipo de atributos.

MongoDB da soporte a más de una decena de lenguajes de programación.

Mongoose

Mongoose es una librería ODM (Object Data Modeling) de MongoDB para Node. Con esta herramienta se pueden modelar los esquemas de las colecciones definiendo sus atributos con sus tipos, dominio y valores por defecto. Esta librería facilita las operaciones CRUD en la base de datos pues cada modelo creado a partir de un esquema definido con mongoose posee métodos para crear, actualizar consultar y eliminar documentos de las colecciones en MongoDB.

1.7.2. Express

Express es un marco de trabajo minimalista y robusto para el desarrollo de aplicaciones web en Node. Es la tercera capa del stack y forma parte de las

¹⁷<https://www.mongodb.com/>

tecnologías que se emplean en el lado del servidor.

Express tiene una herramienta para crear proyectos web de forma sencilla. Dicha herramienta genera una estructura de directorios para organizar los archivos del nuevo proyecto.

Con express configurar las rutas por las que dará servicios una aplicación es cosa sencilla y se realiza en pocas líneas de código.

1.7.3. AngularJS

AngularJS es la cuarta capa del stack y su entorno de ejecución es el cliente. Angular es un marco de trabajo de JavaScript utilizado para crear aplicaciones web de una sola página de una manera más natural. Trabaja con el patrón de diseño MVC y extiende el lenguaje HTML agregando atributos, directivas y expresiones.

Angular hace posible ligar los modelos utilizados en los controladores con la vista de maneras uni- y bidireccional, por lo que cambios en el modelo se ven reflejados en la vista automáticamente y viceversa (si se trata del caso bidireccional).

1.7.4. Node.js

Node es el entorno de ejecución de JavaScript en el lado del servidor. Utiliza el motor de JavaScript V8 de Chrome para compilar los scripts a código máquina. Posee un conjunto de módulos con funcionalidades para: el uso de mensajes por consola, resolución de nombres de dominio, manejo del sistema de archivos, comunicación mediante HTTP y HTTPS, obtener información del SO, entre otras.

Capítulo 2

Instalación de las Herramientas

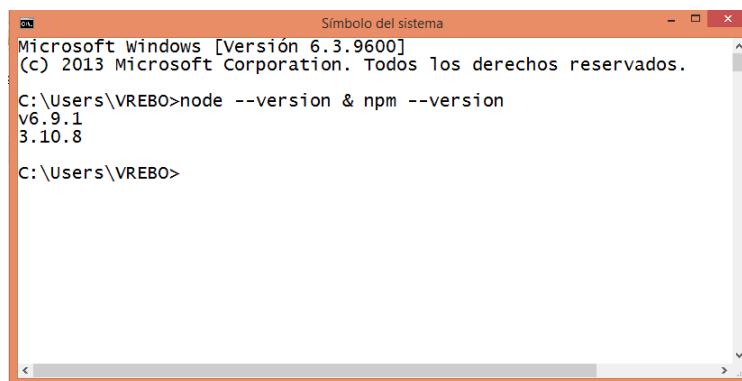
En este capítulo se explica el procedimiento de instalación de las herramientas utilizadas en esta guía y las configuraciones necesarias para su utilización en el sistema operativo Windows.

2.1. Node.js

Para instalar Node basta con ejecutar el instalador adecuado para la arquitectura del equipo en que se instalará y seguir la configuración recomendada. Los paquetes de instalación se pueden descargar desde este el portal de descargas de Node¹.

En la imagen 2.1 se puede notar que en la instalación de Node se incluye npm. npm es un administrador de paquetes al estilo apt y yum de Linux o homebrew de OS X para Node.

Una vez terminada la instalación, compruebe que se hizo correctamente introduciendo el comando **node --version** & **npm --version** en el CMD.

A screenshot of a Windows Command Prompt window titled 'Símbolo del sistema'. The window shows the following text: 'Microsoft Windows [Versión 6.3.9600]', '(c) 2013 Microsoft Corporation. Todos los derechos reservados.', 'C:\Users\VREBO>node --version & npm --version', 'v6.9.1', '3.10.8', and 'C:\Users\VREBO>'. The output indicates that Node.js version 6.9.1 and npm version 3.10.8 are installed and accessible from the command line.

```
Microsoft Windows [Versión 6.3.9600]
(c) 2013 Microsoft Corporation. Todos los derechos reservados.
C:\Users\VREBO>node --version & npm --version
v6.9.1
3.10.8
C:\Users\VREBO>
```

Figura 2.2: Comprobación de la instalación de node.

¹<https://nodejs.org/en/download/>

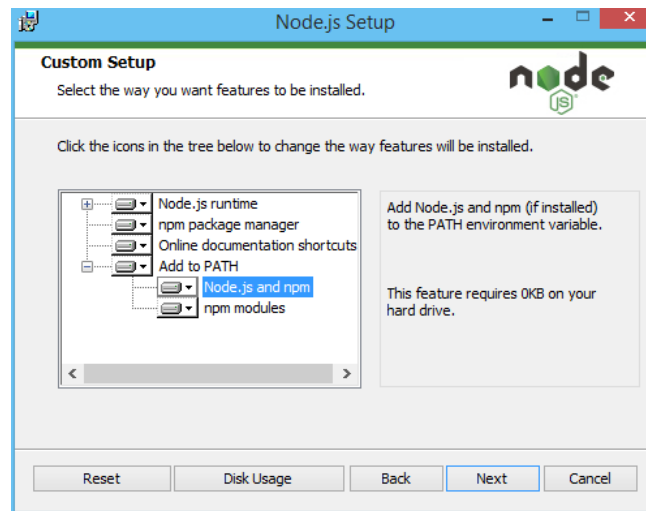


Figura 2.1: Configuración sugerida por el instalador de Node en Windows.

2.2. MongoDB

MongoDB puede obtenerse desde el centro de descargas de su sitio web.

Su instalación en Windows es sencilla, sólo hay que ejecutar el paquete de instalación descargado y seguir la instalación recomendada. La instalación se realizará en el directorio *C:/Program Files/MongoDB/Server/3.2* o similar, dependiendo de la versión instalada.

En la imagen 2.3, se observa el resultado de la instalación. Dentro de la carpeta *bin* se encuentran los componentes principales y utilerías de MongoDB. Se recomienda la lectura del archivo README en el que se explica claramente la función de algunos de los componentes.

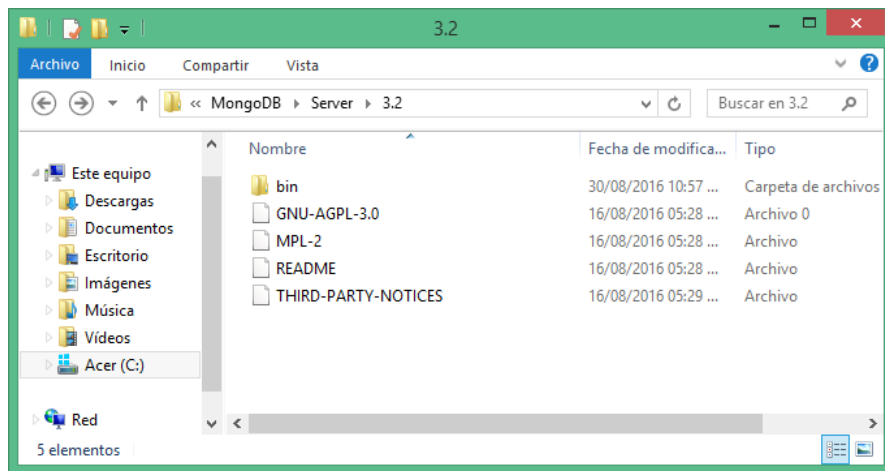


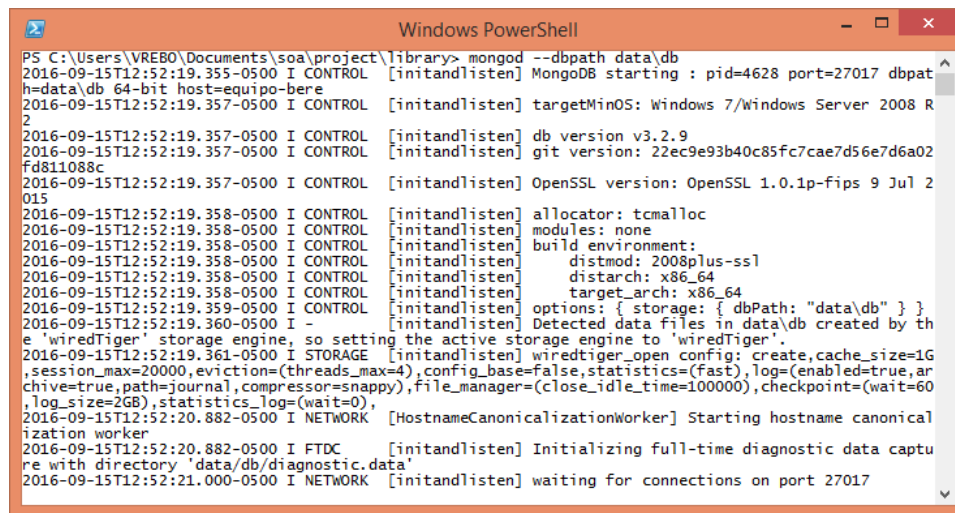
Figura 2.3: Directorio de instalación de MongoDB

Para poder hacer uso de Mongo desde el Command Prompt o Power Shell de Windows es necesario editar la variable de entorno PATH agregando la ruta de la carpeta *bin* del directorio de instalación de Mongo.

Para modificar la variable PATH se pueden seguir estos pasos. En el paso 4 se selecciona la variable PATH, se da click en Editar y se agrega `”;C:/Program Files/MongoDB/Server/3.2/bin”` al final del valor de la variable.

Para poner a andar MongoDB se utiliza el ejecutable **mongod** acompañado de la opción `—dbpath C:/../whatever/data/db`, esta opción indica al servicio de de Mongo cual es el directorio donde estarán o están contenidas las bases de datos. Es necesario que el directorio exista previo al inicio del servicio **mongod**.

En la figura 2.4 se observa el servicio de MongoDB en ejecución esperando por conexiones a través del puerto 27017, que es el puerto por defecto del servicio.



```
PS C:\Users\VREBO\Documents\soa\project\library> mongod --dbpath data\db
2016-09-15T12:52:19.355-0500 I CONTROL [initandlisten] MongoDB starting : pid=4628 port=27017 dbpat
h=data\db 64-bit host=equipo-bere
2016-09-15T12:52:19.357-0500 I CONTROL [initandlisten] targetMinOS: Windows 7/Windows Server 2008 R
2
2016-09-15T12:52:19.357-0500 I CONTROL [initandlisten] db version v3.2.9
2016-09-15T12:52:19.357-0500 I CONTROL [initandlisten] git version: 22ec9e93b40c85fc7cae7d56e7d6a02
fd811088c
2016-09-15T12:52:19.357-0500 I CONTROL [initandlisten] OpenSSL version: OpenSSL 1.0.1p-fips 9 Jul 2
015
2016-09-15T12:52:19.358-0500 I CONTROL [initandlisten] allocator: tcmalloc
2016-09-15T12:52:19.358-0500 I CONTROL [initandlisten] modules: none
2016-09-15T12:52:19.358-0500 I CONTROL [initandlisten] build environment:
2016-09-15T12:52:19.358-0500 I CONTROL [initandlisten]   distmod: 2008plus-ssl
2016-09-15T12:52:19.358-0500 I CONTROL [initandlisten]   distarch: x86_64
2016-09-15T12:52:19.358-0500 I CONTROL [initandlisten]   target_arch: x86_64
2016-09-15T12:52:19.359-0500 I CONTROL [initandlisten] options: { storage: { dbPath: "data\db" } }
2016-09-15T12:52:19.360-0500 I - [initandlisten] Detected data files in data\db created by th
e 'wiredTiger' storage engine, so setting the active storage engine to 'wiredTiger'.
2016-09-15T12:52:19.361-0500 I STORAGE [initandlisten] wiredtiger_open config: create,cache_size=1G
,session_max=20000,eviction=(threads_max=4),config_base=false,statistics=(fast),log=(enabled=true,ar
chive=true,path=journal,compressor=snappy),file_manager=(close_idle_time=100000),checkpoint=(wait=60
,log_size=2GB),statistics_log=(wait=0),
2016-09-15T12:52:20.882-0500 I NETWORK [HostnameCanonicalizationWorker] Starting hostname canonical
ization worker
2016-09-15T12:52:20.882-0500 I FTDC [initandlisten] Initializing full-time diagnostic data captu
re with directory 'data/db/diagnostic.data'
2016-09-15T12:52:21.000-0500 I NETWORK [initandlisten] waiting for connections on port 27017
```

Figura 2.4: Servicio mongod iniciado desde Windows Power Shell.

Ahora se puede ejecutar el cliente de MongoDB desde una línea de comandos para gestionar las colecciones de nuestras bases de datos.

2.3. Express y Mongoose

Ambos son módulos de Node y como tal su instalación se realiza con el gestor de paquetes npm. La instalación puede hacerse de forma global o local para cada proyecto. En esta guía mostramos solo el modo local.

Previo a la instalación es necesario tener un archivo `package.json` con una estructura como la del siguiente cuadro.

```
{
  "name": "my-project",
  "version": "0.0.0",
  "dependencies": {
  }
}
```

Sólo resta ejecutar el comando **npm install express mongoose --save** (con doble guión medio antes de "save") y npm creará un directorio `node_modules` (si no lo hay), descargará y copiará en él los archivos de los módulos de express (figura 2.5) y mongoose (figura 2.6), y agregará a nuestro archivo `package.json` las dependencias de ambos módulos.

```
...
"dependencies": {
  "express": "^4.14.0",
  "mongoose": "^4.6.8"
}
...
```

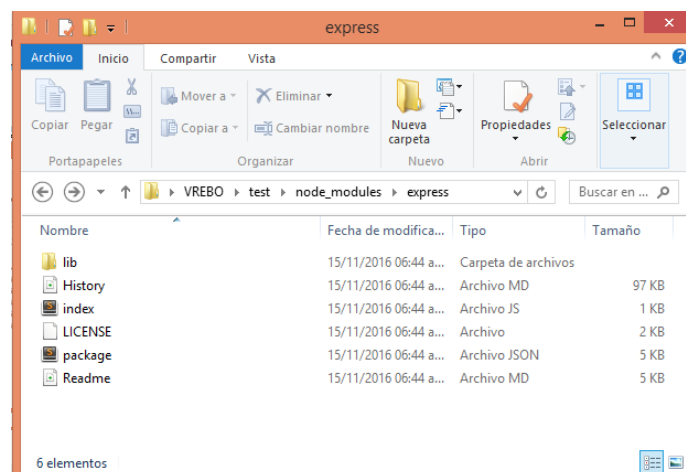


Figura 2.5: Directorio de instalación de express

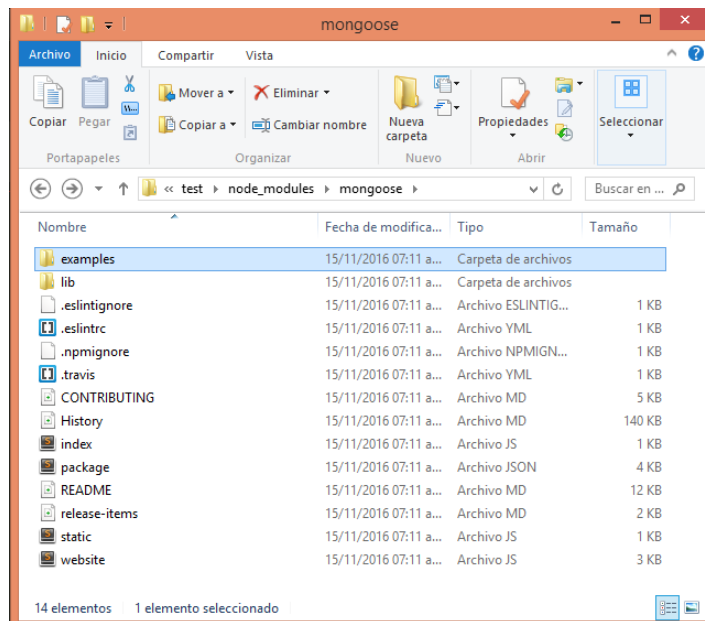


Figura 2.6: Directorio de instalación de mongoose

2.4. Bootstrap, JQuery, AngularJS

En el caso de estas tres dependencias no se realiza una instalación tal cual. Puede optarse por incluirlas en las páginas que lo requieran mediante una CDN (Content Delivery Network).

```
<!-- Hojas de estilo de Bootstrap -->
<link
  rel="stylesheet"
  href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.
min.css">

<!-- Librería de JQuery -->
<script
  src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.
js"></script>

<!-- Marco de trabajo de AngularJS -->
<script
  src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.
min.js"></script>

<!-- Scripts requeridos por Bootstrap -->
<script
  src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.
min.js"></script>
```

Hay que mencionar que para el proyecto de ejemplo de esta guía sí se descargaron estas dependencias y se proveen desde el servidor a la aplicación. Para

lograr esto se hicieron unas configuraciones adicionales en el servidor pero no se entra en detalle para evitar confundir al lector.

Capítulo 3

Desarrollo de una aplicación MEAN a través de servicios web REST

3.1. Descripción de la aplicación

Para ejemplificar el uso del Stack MEAN y las herramientas antes descritas se desarrolló una aplicación web destinada a bibliotecas. La aplicación permitirá por un lado, a la institución bibliotecaria hacer accesible el catálogo de su acervo a sus usuarios a través un sitio web, además los usuarios que hagan uso del servicio de préstamo de ejemplares a domicilio podrán consultar sus fechas de entrega por el mismo portal.

El proyecto está compuesto por dos sub-proyectos, uno para el back-end y otro para el front-end.

3.2. Back-end

El proyecto del back-end contiene la base de datos de MongoDB, los modelos necesarios para hacer las operaciones en la base de datos y un servidor web express que atiende las peticiones a los endpoints REST listados en la tabla ??.

3.2.1. API REST

En la tabla 3.2.1 se listan y describen los endpoints utilizados por el sistema.

Método HTTP	Endpoint	Descripción
GET	/api/v1/books	Obtiene la lista de todos los libros
GET	/api/v1/books/{id}	Obtiene el libro cuyo id coincide con {id}
POST	/api/v1/books	Agrega un nuevo libro a la base de datos
PUT	/api/v1/books/{id}	Actualiza el libro cuyo id coincide con {id}
DELETE	/api/v1/books/{id}	Elimina el libro cuyo id coincide con {id} de la base de datos
GET	/api/v1/copies	Obtiene la lista de todas las copias de libros
GET	/api/v1/copies?bookId=	Obtiene la lista de copias del libro cuyo id coincide con el parámetro bookId
GET	/api/v1/copies/{id}	Obtiene la copia cuyo id coincide con {id}
POST	/api/v1/copies	Agrega una nueva copia a la base de datos
DELETE	/api/v1/copies/{id}	Elimina la copia cuyo id coincide con {id} de la base de datos
GET	/api/v1/categories	Obtiene la lista de todas las categorías
GET	/api/v1/categories/{id}	Obtiene la categoría cuyo id coincide con {id}
POST	/api/v1/categories	Agrega una nueva categoría a la base de datos
PUT	/api/v1/categories/{id}	Actualiza la categoría cuyo id coincide con {id}
DELETE	/api/v1/categories/{id}	Elimina la categoría cuyo id coincide con {id} de la base de datos
GET	/api/v1/users	Obtiene la lista de todos los usuarios
GET	/api/v1/users/{id}	Obtiene el usuario cuyo id coincide con {id}
POST	/api/v1/users	Agrega un nuevo usuario a la base de datos
PUT	/api/v1/users/{id}	Actualiza el usuario cuyo id coincide con {id}
DELETE	/api/v1/users/{id}	Elimina el usuario cuyo id coincide con {id} de la base de datos
GET	/api/v1/lendings	Obtiene la lista de todos los préstamos de libros
GET	/api/v1/lendings?readerId=	Obtiene la lista de todos los préstamos de libros del usuario cuyo id coincide con readerId
POST	/api/v1/lendings	Agrega un nuevo préstamo a la base de datos
DELETE	/api/v1/lendings?copyId=	Elimina el préstamo de la copia cuyo id coincide con copyId de la base de datos

Cuadro 3.1: Endpoints REST del sistema

3.2.2. Esquema de la Dase de Datos

La figura 3.1 muestra el esquema de las colecciones diseñadas para las funciones sistema.

books	users
*isbn String	*name String
*title String	*lastname String
*author String	*password String
*editorial String	*phone String
*description String	*email String
*cover_photo String	*address String
*categories [ObjectId refers categories]	*createdAt Date
	*profilePhoto String
	*updatedAt Date

categories
*name String
*description String

copies	lendings
*pages Number	*copy ObjectId refers copies
*state enum	*reader ObjectId refers users
*availabilities enum	*lentAt Date
*edition String	*returnAt Date
*publishedAt Date	
*language enum	
*book ObjectId refers books	

Figura 3.1: Esquema de la base de datos con atributos.

A continuación se describe el propósito de cada colección de la base de datos.

books: Almacena la información de los libros del acervo de la biblioteca. Posee un atributo para conocer las categorías a las que pertenece cada libro.

copies: Contiene información sobre el estado de los ejemplares de un libro y algunos datos propios de la entidad como el lenguaje y su edición.

categories: En esta colección se guardan aquellas categorías temáticas a las que puede pertenecer un libro.

users: Almacena los datos de los usuarios del servicio bibliotecario.

lendings: Registra los préstamos pendientes de entrega. Los documentos de esta colección contienen atributos para saber que copia se prestó a que usuario.

3.2.3. Estructura de la Aplicación

En el siguiente cuadro se muestra la lista de directorios que componen el proyecto library-backend junto con una descripción de su contenido.

library-backend		
__ bin		
__ www		Archivo de arranque del servidor.
__ data		
__ db/		Archivos de la base de datos.
__ models		
__ database		
__ crud/		Archivos .js con las clases para hacer operaciones CRUD en la base de datos.
__ schema/		Archivos .js con la definición de los de los esquemas de la entidad de la base de datos.
__ node_modules		Dependencias de node.js para el sistema.
__ public		
__ images/		Directorio público del servidor de las imágenes de perfil de usuarios y portadas de libros.
__ routes		
__ api-v1/		Archivo .js donde se declaran los endpoints de los servicios web del sistema.
__ app.js		Archivo de configuración del servidor.
__ package.json		Archivo para declarar dependencias y metadatos del proyecto.

3.3. Front-end

Este proyecto contiene todas las dependencias del front-end (Angular, JQuery y Bootstrap) junto con un servidor express para que los usuarios administradores accedan a la aplicación de página única del sistema.

3.3.1. Estructura de la Aplicación

```
library-frontend
|-- bin
|   |-- www                Archivo de arranque del servidor.
|-- bower_components/      Directorio que contiene las depen-
|                           dencias web del sistema (Bootstrap, JQuery,
|                           AngularJS).
|-- node_modules            Dependencias de node.js para el
|                           sistema.
|-- public
|   |-- css/               Hojas de estilo personalizadas.
|   |-- js/
|       |-- app.js         Archivo de configuración de Angu-
|                           larJS, en
|                           él se definen los fragmentos de la
|                           IU que
|                           se intercambiarán en la SPA.
|       |-- controllers.js Archivo de definición de los con-
|                           troladores
|                           de los diferentes fragmentos de la
|                           SPA.
|       |-- web-service.js  Archivo en donde se encapsulan las peti-
|                           ciones a los endpoints del backend.
|-- routes
|   |-- admin.js           Archivo donde se declaran las URLs principal
|                           y de los fragmentos de la SPA.
|-- views
|   |-- app.html           Esqueleto de la SPA.
|   |-- book/              Contiene fragmentos los de html relativos a
|                           actividades de gestión de libros que serán
|                           insertados en la SPA.
|   |-- category/          Contiene fragmentos los de html relativos a
|                           actividades de gestión de categorías que serán
|                           insertados en la SPA.
|   |-- user/              Contiene fragmentos los de html relativos a
|                           actividades de gestión de usuarios que serán
|                           insertados en la SPA.
|   |-- lending/           Contiene fragmentos los de html relativos a
|                           actividades de gestión de préstamos que serán
|                           insertados en la SPA.
|-- app.js                 Archivo de configuración del servidor.
|-- package.json           Archivo para declarar dependencias y
                           metadatos del proyecto.
```

3.3.2. Conexión con el Back-end

La interfaz de comunicación entre el front-end y back-end del sistema se encuentra en el archivo *web-service.js*. Dentro se definió una función constructora genérica que recibe un objeto *\$http* utilizado para las peticiones AJAX al back-end y una cadena de texto *url* que indica el endpoint al que se hará la petición. Esta función genérica devuelve un objeto JSON con funciones generales para consultar, crear, actualizar y eliminar recursos al endpoint.

En el mismo archivo se definió una función constructora por cada endpoint en el back-end. En estas funciones se hace uso de la función genérica y se especializa el objeto devuelto reemplazando algunas funciones según se requiera.

Por último, se registran las funciones constructoras especializadas como servicios en AngularJS para que sean usadas posteriormente por los controladores.

3.4. Instalación de la aplicación

El proyecto de ejemplo se encuentra a disposición del lector en este enlace¹ a la plataforma GitHub.

3.4.1. Descarga

El primer paso es descargar el proyecto. La descarga puede hacerse clonando el proyecto desde el repositorio con un sistema de control de versiones (git) o mediante la descarga directa de los archivos del proyecto en un archivo comprimido. Para esta instalación descargaremos el archivo .zip como se muestra en la figura 3.2.

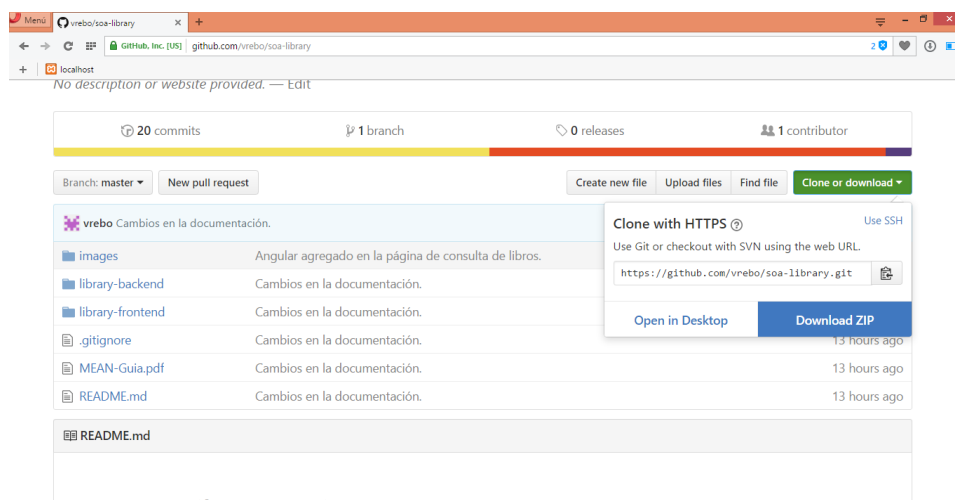


Figura 3.2: Directorio de instalación de mongoose

Terminada la descarga se descomprime el archivo en el directorio que el lector prefiera.

¹<https://github.com/vrebo/soa-library>

3.4.2. Configuración

Ambos proyectos contienen un servidor web express que por defecto están configurados para escuchar peticiones en los puertos 3000 y 3001. En caso de que esto cause conflicto con algún otro software en el equipo se puede hacer el cambio de los puertos la línea 15 de los archivos *C:/directorio_de_instalacion/soa-library-master/library-backend/bin/app.js* y *C:/directorio_de_instalacion/soa-library-master/library-frontend/bin/app.js*.

El código del siguiente cuadro muestra la línea del código donde se debe hacer el cambio.

```
...
//En esta línea se cambia el puerto.
var port = normalizePort(process.env.PORT || '3000');
app.set('port', port);
...
```

3.4.3. Arranque de los Sistemas

Back-end

Antes de iniciar los servidores es necesario arrancar el servicio de MongoDB tal como se mostró en 2.2. Para ello se debe abrir una consola de comandos en el directorio del proyecto del back-end y ejecutar el comando *mongod --dbpath data\db*.

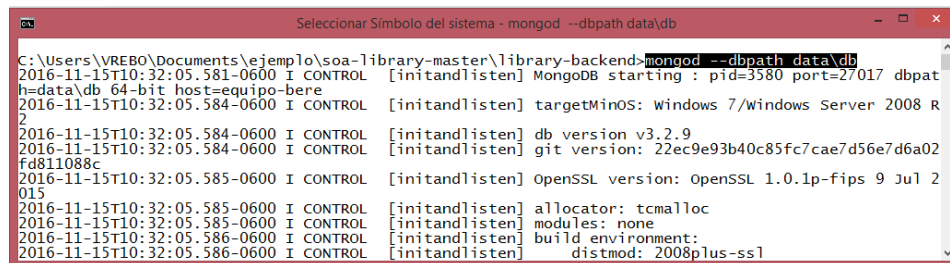


Figura 3.3: Arranque de la base de datos del sistema

Ahora, en otra consola de comandos en el mismo directorio del proyecto del back-end, iniciamos el servidor con el comando *npm start*.

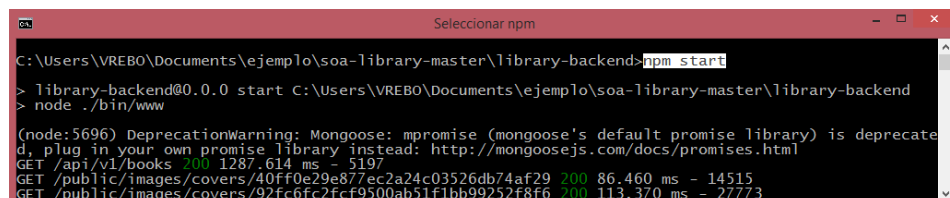


Figura 3.4: Arranque del servidor del back-end

Se puede comprobar que el servidor del back-end funciona correctamente y atiende las peticiones a los endpoints usando herramientas como Postman (extensión de Chrome) que nos permite armar peticiones HTTP a servidores web.

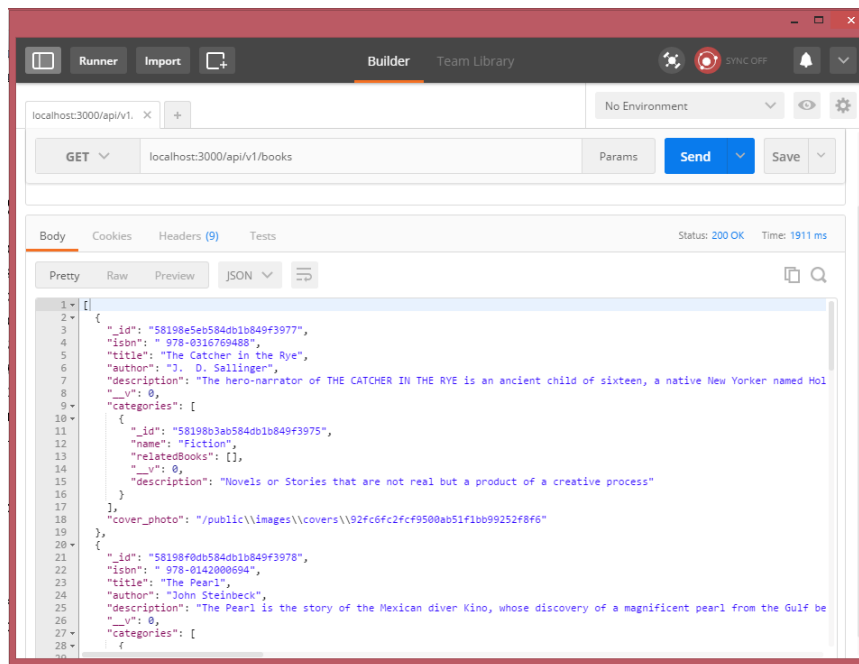
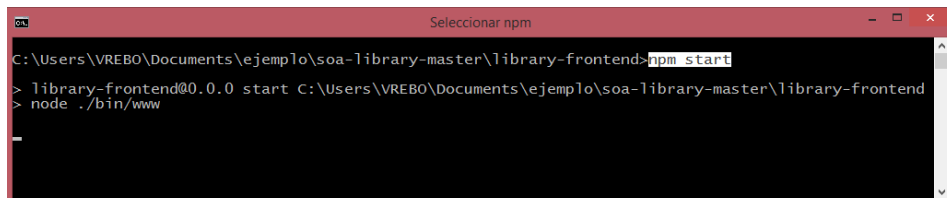


Figura 3.5: Front-end del sistema

Front-end

El mismo comando usado para arrancar el servidor del back-end se utiliza para el front-end en el directorio *C:/directorio_de_instalacion/soa-library-master/library-frontend/*.



```
Seleccionar npm
C:\Users\VREBO\Documents\ejemplo\soa-library-master\library-frontend>npm start
> library-frontend@0.0.0 start C:\Users\VREBO\Documents\ejemplo\soa-library-master\library-frontend
> node ./bin/www
```

Figura 3.6: Arranque del servidor del front-end

Para probar que el front-end junto con el back-end funcionan correctamente, abra la URL *http://localhost:3001/admin/* en un navegador web. La página de la figura 3.7 debería mostrarse en el navegador.

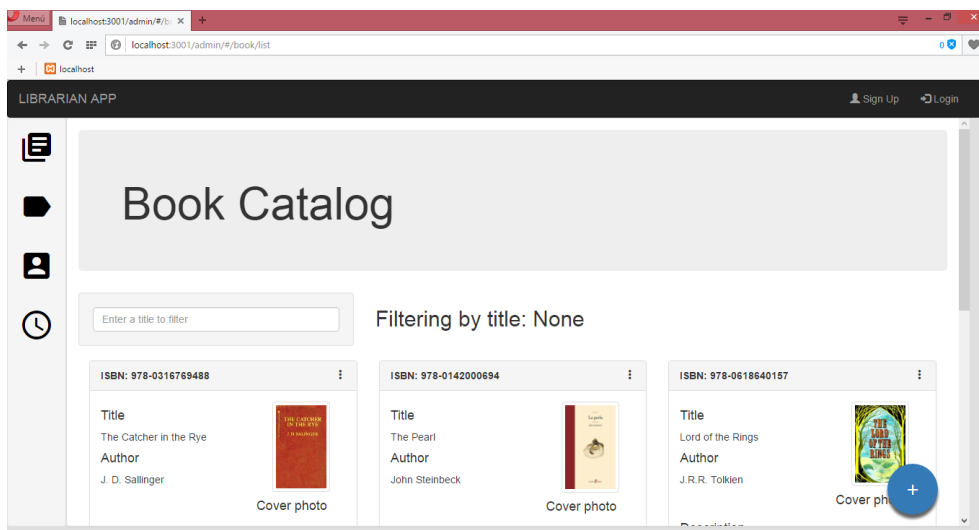


Figura 3.7: Front-end del sistema