actibyti] a netmind social project

METEOR

Autor: Ricardo Ahumada

GOBIERNO DE ESPAÑA
MINISTERIO DE ENERGÍA, TURISMO Y AGENDA DIGITAL
red.es
ESTRATEGIA DE EMPRENDIMIENTO Y EMPLEO JOVEN garantía juvenil
Agenda Digital para España
UNIÓN EUROPEA
Fondo Social Europeo
"El FSE invierte en tu futuro"

Framework Meteor

- Meteor es una plataforma JavaScript completa para desarrollar aplicaciones web y móviles modernas. Meteor incluye un conjunto clave de tecnologías para la construcción de aplicaciones reactivas de clientes conectados, una herramienta de compilación y un conjunto de paquetes comisariados de Node.js y la comunidad general de JavaScript.

- Características de Meteor
  - Web y Móvil - Meteor ofrece una plataforma para desarrollar aplicaciones Web, Android e IOS.
  - Aplicaciones universales: el mismo código para navegadores web y dispositivos móviles.
  - Paquetes - Gran cantidad de paquetes fáciles de instalar y usar.
  - Meteor Galaxy - Servicio de nube para el despliegue de la aplicación Meteor.

Framework Meteor

- Ventajas de Meteor
  - Los desarrolladores sólo necesitan JavaScript para el desarrollo servidor y cliente.
  - La codificación es muy sencilla y agradable para principiantes.
  - Las aplicaciones Meteor son en tiempo real de forma predeterminada.
  - Los paquetes oficiales y comunitarios son un gran ahorro de tiempo.

- Limitaciones de Meteor
  - Meteor no es muy adecuado para aplicaciones grandes y complejas.
  - Hay mucha magia cuando se trabaja con Meteor para que los desarrolladores puedan encontrarse limitados de alguna manera.

Instalación de Meteor

- Requisito previo:
  - AngularJS es la plataforma necesaria para el desarrollo de Meteor. Así que debe instalar Angular en su sistema operativo antes de continuar, la instalación de Angular fue explicada anteriormente.
  - Tener Ionic instalado para trabajar en las aplicaciones Android e los.
- Instalar Meteor
  - Puede descargar el instalador oficial de Meteor desde esta página: https://www.meteor.com/install
  - Si se produce algún error durante la instalación, puede intentar ejecutar el instalador como administrador. Una vez completada la instalación, se le pedirá que cree la cuenta Meteor.
  - Cuando termine de instalar el instalador de Meteor, puede probar si todo está instalado correctamente ejecutando el código siguiente en la ventana del símbolo del sistema:

```
C:\Users\username>meteor
```

```
run: You're not in a Meteor project directory.

To create a new Meteor project:
  meteor create <project name>
For example:
  meteor create myapp

For more help, see 'meteor --help'.
```

Diseño y estructura de Meteor (Primera app)

- Comenzaremos creando la estructura de carpetas del proyecto, Meteor tiene un comportamiento especial para ciertas carpetas:
  - Cliente - Estos archivos estarán disponibles sólo en el lado del cliente.
  - Server - Estos archivos estarán disponibles sólo en el lado del servidor.
  - Public - Estos archivos servirán para el cliente. Como imágenes, fuentes, etc.
  - Lib - Cualquier carpeta llamada lib (en cualquier jerarquía) se cargará primero.
  - Cualquier otro nombre de carpeta se incluirá en el cliente y el servidor y se utilizará para compartir código.
- Así que esta será nuestra estructura de carpetas para el proyecto:
  - Cliente (lado del cliente con AngularJS y código ionic)
  - scripts
  - plantillas
  - Estilos
  - Index.html
  - Servidor (sólo código del servidor)
  - Public (bienes, imágenes)
  - Lib (definir métodos y colecciones con el fin de hacerlos disponibles tanto en el cliente como en el servidor)

Diseño y estructura de Meteor (Primera app)

- Empezaremos por crear nuestro primer archivo, el index.html que se colocará en la carpeta del cliente:

```
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="initial-scale=1, maximum-scale=1, user-scalable=no,
width=device-width">
    <title>Whatsapp Meteor</title>
</head>
<body>
   <!-- Barra de navegación que se actualizará a medida que naveguemos entre las vistas.-->
    <ion-nav-bar class="bar-stable">
    <ion-nav-back-button>
    </ion-nav-back-button>
    </ion-nav-bar>
   <!-- Las opciones se mostrarán en el <ion-nav-view>.-->
    <ion-nav-view>
    </ion-nav-view>
</body>
```

Diseño y estructura de Meteor (Primera app)

- Utilizamos algunas etiquetas de Ionic para lograr estilo móvil:
  - Ion-nav-bar: Crea una barra de navegación en el encabezado de la página.
  - Ion-nav-view: Esto es un marcador de posición para el contenido real. AngularJS y Ionic pondrán tu contenido dentro de esta etiqueta automáticamente.
  - Tenga en cuenta que sólo proporcionamos las etiquetas <head> y <body> porque Meteor se encarga de adjuntar las partes html relevantes en un archivo y cualquier etiqueta que usaremos aquí se agregará al archivo principal index.html de Meteor.

- Nuestro siguiente paso es crear el módulo AngularJS y arrancarlo de acuerdo con nuestra plataforma. Crearemos un nuevo archivo denominado app.js.

- Este archivo debe cargarse primero, ya que cualquier otro código de AngularJS dependerá de este módulo, así que necesitamos poner este archivo dentro de una carpeta llamada lib, crearemos un archivo en esta ruta: client/scripts/lib/app.js

Diseño y estructura de Meteor (Primera app)

- En este fichero inicializaremos todos los módulos que necesitamos y cargaremos nuestros módulos-ayudantes, así que cada vez que creemos un módulo-ayudante se debe cargar aquí inmediatamente después.

```
// Libs
import 'angular-animate';
import 'angular-meteor';
import 'angular-sanitize';
import 'angular-ui-router';
import 'ionic-scripts';
import Angular from 'angular';
import { Meteor } from 'meteor/meteor';
// Modules
const App = 'Whatsapp';
// App
Angular.module(App, [ 'angular-meteor', 'ionic']);
// Startup
if (Meteor.isCordova) {
 Angular.element(document).on('deviceready', onReady);
}else { Angular.element(document).ready(onReady);}
function onReady() { Angular.bootstrap(document, [App]);}
```

Diseño y estructura de Meteor (Primera app)

- Nuestro siguiente paso es crear los estados y rutas para las vistas.
- Nuestra aplicación utiliza Ionic para crear 5 pestañas: favoritos, recientes, contactos, chats y configuraciones.
- Vamos a definir nuestras rutas y estados con angular-ui-router (que está incluido por Ionic), y en el momento vamos a añadir la página principal que es la pestaña chats:

```
//Agregando rutas iniciales

import { Config } from 'angular-ecmascript/module-helpers';
import chatsTemplateUrl from '../templates/chats.html';
import tabsTemplateUrl from '../templates/tabs.html';

export default class RoutesConfig extends Config { configure() {
   this.$stateProvider .state('tab', {
          url: '/tab', abstract: true, templateUrl: tabsTemplateUrl }) .state('tab.chats', { url: '/chats',
views: { 'tab-chats': { templateUrl: chatsTemplateUrl } } });

   this.$urlRouterProvider.otherwise('tab/chats'); }}

RoutesConfig.$inject = ['$stateProvider', '$urlRouterProvider'];
```

Diseño y estructura de Meteor (Primera app)

- Esta página va en la ruta: client/scripts/lib/app.js

```
//Cargando configuración de rutas

import 'angular-ui-router';
import 'ionic-scripts';
import Angular from 'angular';
import Loader from 'angular-ecmascript/module-loader';
import { Meteor } from 'meteor/meteor';
// Modules
import RoutesConfig from '../routes';
const App = 'Whatsapp'; ...some lines skipped... 'ionic']);
new Loader(App) .load(RoutesConfig);
// Startup
if (Meteor.isCordova) { Angular.element(document).on('deviceready', onReady);}
```

Diseño y estructura de Meteor (Primera app)

- Y esta es la plantilla HTML para el pie de página que incluye nuestras pestañas:

```
<ion-tabs class="tabs-stable tabs-icon-top tabs-color-positive" ng-cloak>

  <ion-tab title="Favorites" icon-on="ion-ios-star" icon-off="ion-ios-star-outline"
href="#/tab/favorites"> <ion-nav-view name="tab-favorites"></ion-nav-view> </ion-tab>

  <ion-tab title="Recents" icon-on="ion-ios-clock" icon-off="ion-ios-clock-outline"
href="#/tab/recents"> <ion-nav-view name="tab-recents"></ion-nav-view> </ion-tab>

  <ion-tab title="Contacts" icon-on="ion-ios-person" icon-off="ion-ios-person-outline"
href="#/tab/contacts"> <ion-nav-view name="tab-contacts"></ion-nav-view>

  </ion-tab> <ion-tab title="Chats" icon-on="ion-ios-chatbubble" icon-off="ion-ios-chatbubble-
outline" href="#/tab/chats"> <ion-nav-view name="tab-chats"></ion-nav-view> </ion-tab>

  <ion-tab title="Settings" icon-on="ion-ios-cog" icon-off="ion-ios-cog-outline"
href="#/tab/settings"> <ion-nav-view name="tab-settings"></ion-nav-view> </ion-tab>

</ion-tabs>
```

Diseño y estructura de Meteor (Primera app)

- Vamos a crear el stub para nuestra pestaña predeterminada: la pestaña chats:

```
<ion-view view-title="Chats">
      <ion-content>  </ion-content>
</ion-view>
```

- Nuestro próximo paso pasará por la creación de vistas básicas con algunos datos estáticos utilizando Ionic y sus

```
//Creando controladores de chat
import { Controller } from 'angular-ecmascript/module-helpers';
export default class ChatsCtrl extends Controller { }
ChatsCtrl.$name = 'ChatsCtrl';
```

```
//Cargando controladores de chat
import { Meteor } from 'meteor/meteor';
// Modules
import ChatsCtrl from '../controllers/chats.controller';
import RoutesConfig from '../routes';
const App = 'Whatsapp';...some lines skipped...]);
new Loader(App) .load(ChatsCtrl) .load(RoutesConfig);
//Puesta en marcha
```

Diseño y estructura de Meteor (Primera app)

- Ahora vamos a agregar los datos estáticos a ChatsCtrl. Vamos a crear un esquema de stub para chats y mensajes:

```
import Moment from 'moment';
import { Controller } from 'angular-ecmascript/module-helpers';
export default class ChatsCtrl extends Controller { constructor() {
      super(...arguments);
      this.data = [
          { _id: 0, name: 'Ethan Gonzalez', picture: 'https://randomuser.me/api/portraits/thumb/men/1.jpg',
lastMessage: { text: 'You on your way?', timestamp: Moment().subtract(1, 'hours').toDate() } },
          { _id: 1, name: 'Bryan Wallace', picture: 'https://randomuser.me/api/portraits/thumb/lego/1.jpg',
lastMessage: { text: 'Hey, it\'s me', timestamp: Moment().subtract(2, 'hours').toDate() } },
          { _id: 2, name: 'Avery Stewart', picture: 'https://randomuser.me/api/portraits/thumb/women/1.jpg',
lastMessage: { text: 'I should buy a boat', timestamp: Moment().subtract(1, 'days').toDate() } },
          { _id: 3, name: 'Katie Peterson', picture: 'https://randomuser.me/api/portraits/thumb/women/2.jpg',
lastMessage: { text: 'Look at my mukluks!', timestamp: Moment().subtract(4, 'days').toDate() } },
          { _id: 4, name: 'Ray Edwards', picture: 'https://randomuser.me/api/portraits/thumb/men/2.jpg', lastMessage:
{ text: 'This is wicked good ice cream.', timestamp: Moment().subtract(2, 'weeks').toDate() } } ]; }}

ChatsCtrl.$name = 'ChatsCtrl';
```

Diseño y estructura de Meteor (Primera app)

- Conectamos la vista de chats al CharaCtrl:

```
url: '/chats', views: {
      'tab-chats': {
            templateUrl: chatsTemplateUrl, controller: 'ChatsCtrl as chats' }
} });
```

- Tenga en cuenta que usamos la sintaxis controllerAs con el valor chats. Esto significa que se debe acceder al controlador a través de un modelo de datos llamado chats, que es sólo una referencia al ámbito.
- Ahora haremos que los datos aparezcan en nuestra vista. Utilizaremos las directivas de Ionic para crear un contenedor con una vista de lista (ion-list e ion-item), y añadir ng-repeat para iterar sobre los chats:

```
<ion-view view-title="Chats">     <ion-content>     <ion-list>
<ion-item ng-repeat="chat in chats.data | orderBy:'-lastMessage.timestamp'" class="item-chat item-remove-
animate item-avatar item-icon-right" type="item-text-wrap">
<img ng-src="{{ chat.picture }}">
<h2>{{ chat.name }}</h2>
<p>{{ chat.lastMessage.text }}</p>
<span class="last-message-timestamp">{{ chat.lastMessage.timestamp }}</span>
<i class="icon ion-chevron-right icon-accessory"></i>
</ion-item> </ion-list> </ion-content></ion-view>
```

Uso de plantillas en Meteor

- Las plantillas Meteor utilizan tres etiquetas de nivel superior. Los dos primeros son cabeza y cuerpo. Estas etiquetas están haciendo el mismo trabajo que en HTML normal. La tercera etiqueta es plantilla. Este es el lugar donde conectamos HTML a JavaScript.
- Plantilla Simple: El siguiente ejemplo muestra su funcionamiento, estamos creando una plantilla con el atributo (name="myParagraph“). Nuestra etiqueta de plantilla se crea debajo del elemento <body> pero debemos incluirlo antes de que se muestre en pantalla. Podemos hacerlo usando la sintaxis {{> myParagraph}}. En nuestra plantilla estamos usando llaves dobles ({{text}}). Este lenguaje de plantilla en Meteor es llamado Spacebars.

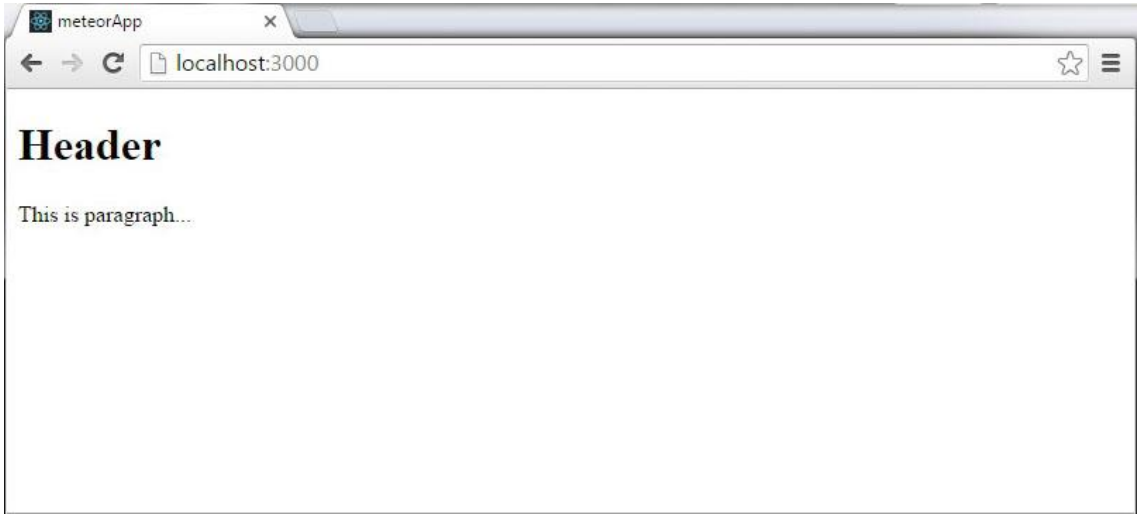
```
<head>
    <title>meteorApp</title>
</head>
<body>
      <h1>Header</h1>
      {{> myParagraph}}
</body>
<template name = "myParagraph">
      <p>{{text}}</p>
</template>
```

Uso de plantillas en Meteor

- En nuestro archivo de JavaScript debemos configurar Template.myParagraph.helpers({}), este método será la conexión a nuestra plantilla.

```
if (Meteor.isClient) {
// Este código sólo se ejecuta en el cliente
Template.myParagraph.helpers({ text: This is paragraph...' });
}
```

- Después de guardar los cambios y ejecutar el programa, obtendremos la siguiente salida:



Uso de plantillas en Meteor

- Plantilla de bloque: En este ejemplo, estamos usando {{#cada-párrafo}} para iterar sobre la matriz de párrafos y devolver la plantilla name="párrafo" para cada valor.

```
<head>
    <title>meteorApp</title>
</head>
<body>
    <div>
        {{#each paragraphs}}
            {{> paragraph}}
        {{/each}}
    </div>
</body>

<template name = "paragraph">
    <p>{{text}}</p>
</template>
```

Uso de plantillas en Meteor

- Necesitamos crear ayudantes de párrafos. Esto será una matriz con cinco valores de texto.

```
if (Meteor.isClient) {
//Este código sólo se ejecuta en el cliente
        Template.body.helpers({
                paragraphs: [
                        { text: "This is paragraph 1..." },
                        { text: "This is paragraph 2..." },
                        { text: "This is paragraph 3..." },
                        { text: "This is paragraph 4..." },
                        { text: "This is paragraph 5..." } ]
}); }
```

- Ahora podemos ver cinco párrafos en la pantalla.

Formularios en Meteor

- Entrada de texto: Primero crearemos el elemento del formulario luego los elementos de campo de texto y Botón de envio.

```
<template name = "myTemplate">
   <form>
      <input type = "text" name = "myForm">
      <input type = "submit" value = "SUBMIT">
   </form>
</template>
```

- RadioButton

```
<template name = "myTemplate">
   <form>
      <input type = "radio" name = "myForm" value = "form-1">FORM 1
      <input type = "radio" name = "myForm" value = "form-2">FORM 2
      <input type = "submit" value = "SUBMIT">
   </form>
</template>
```

Formularios en Meteor

- Checkbox

```
<template name = "myTemplate">
   <form>
      <input type = “checkbox" name = "myForm" value = "form-1">FORM 1
      <input type = “checkbox" name = "myForm" value = "form-2">FORM 2
      <input type = "submit" value = "SUBMIT">
   </form>
</template>
```

- Select Dropdown

```
<template name = "myTemplate">
   <select>
      <option name = "myOption" value = "option-1">OPTION 1</option>
      <option name = "myOption" value = "option-2">OPTION 2</option>
      <option name = "myOption" value = "option-3">OPTION 3</option>
      <option name = "myOption" value = "option-4">OPTION 4</option>
   </select>
</template>
```

BananaTube: Decisión en equipo

- Discute en equipo la conveniencia de usar Meteor para implementar nuevas funcionalidades para BananaTube, por ejemplo un chat
- Tomad nota de las fortalezas y debilidades que hayas observado en el framework, así como, la dificultad de incorporarlo en el proyecto.

Implementando un chat para BananTube

- Implementa un chat para BananaTube que permita que dos usuarios intercambien mensajes.
- Modifica la SPA para hacer esto posible

netmind

WeKnowIT

Barcelona

C. Almogàvers, 123
08018 Barcelona
Tel. 93 304.17.20
Fax. 93 304.17.22

Madrid

Plaza Carlos Trías Bertrán, 7
28020 Madrid
Tel. 91 442.77.03
Fax. 91 442.77.07

www.netmind.es

GOBIERNO DE ESPAÑA
MINISTERIO DE ENERGÍA, TURISMO Y AGENDA DIGITAL
red.es
ESTRATEGIA DE EMPRENDIMIENTO Y EMPLEO JOVEN garantía juvenil
Agenda Digital para España
UNIÓN EUROPEA
Fondo Social Europeo
"El FSE invierte en tu futuro"