

¿Qué ofrece Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.
Ese apoyo que siempre quiso tener...

1. Desarrollo de componentes y proyectos a medida



2. Auditoría de código y recomendaciones de mejora

3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



4. Cursos de formación (impartidos por desarrolladores en activo)

Spring MVC, JSF-PrimeFaces /RichFaces,
HTML5, CSS3, JavaScript-jQuery

Gestor portales (Liferay)
Gestor de contenidos (Alfresco)
Aplicaciones híbridas

Tareas programadas (Quartz)
Gestor documental (Alfresco)
Inversión de control (Spring)

Control de autenticación y
acceso (Spring Security)
UDDI
Web Services
Rest Services
Social SSO
SSO (Cas)

JPA-Hibernate, MyBatis
Motor de búsqueda empresarial (Solr)
ETL (Talend)

Dirección de Proyectos Informáticos.
Metodologías ágiles
Patrones de diseño
TDD

BPM (jBPM o Bonita)
Generación de informes (JasperReport)
ESB (Open ESB)

AdictosAlTrabajo

Terrakas 1x04
¡¡Ya está en la web!! :-)
terrakas.com



autentia
Soporte a desarrollo informático
Hosting patrocinado por
enredados

Entra en Adictos a través de  

E-mail

Contraseña

Entrar [Deseo registrarme](#)
[Olvidé mi contraseña](#)

[Inicio](#) [Quiénes somos](#) [Formación](#) [Comparador de salarios](#) [Nuestro libro](#) [Más](#)

» Estás en: [Inicio](#) [Tutoriales](#) [Creación de un Widget JavaScript usando Backbone.js y Require.js](#)



Carlos García Pérez

Técnico especialista en informática de empresa (CEU).

Ingeniero Técnico en Informática de Sistemas (UPM)

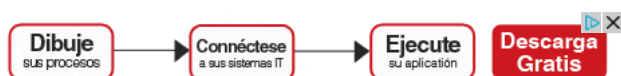
Creador de [MobileTest](#), [Haaala!](#), [Girillo](#), [toi18n](#).

Charla sobre [desarrollo de aplicaciones en Android](#).

Seguir a @cgpoosmad 115 seguidores



[Ver todos los tutoriales del autor](#)



Fecha de publicación del tutorial: 2012-10-31

Tutorial visitado 6 veces [Descargar en PDF](#)

Creación de un Widget JavaScript usando Backbone.js y Require.js

Índice

- [Introducción](#)
- [Aplicación a construir](#)
 - [Modelo de la aplicación](#)
 - [Vistas de la aplicación](#)
 - [Plantillas de las vistas de la aplicación](#)
 - [Internacionalización de los mensajes de la aplicación](#)
- [Referencias](#)
- [Conclusiones](#)

Introducción

Hace un tiempo participé en un proyecto que me hizo darme cuenta que mi forma de programar en JavaScript era más parecido a **programación espagueti**, que a un código que siga patrones de diseño, fácilmente comprensible, modular, mantenible, testeable.

Mi código Javascript realizaba su función correctamente usando JQuery y JSON, pero repito, no era un código de calidad, ¿lo es el tuyo?

En este tutorial vamos a crear una aplicación JavaScript tipo Widget usando las siguientes tecnologías:

- **Backbone**: Para crear la aplicación siguiendo el patrón Modelo-Vista-Controlador.
- **RequireJS**: Para descomponer en módulos (**AMD**) la aplicación (modelos, vistas, controladores) con sus dependencias gestionadas por RequireJS.
- **Plantillas (Templating)**: Usaremos el sistema de plantillas (html) que tiene por defecto Backbone y que se basa en **underscore**. Las plantillas no estará hardcodeada en el código fuente, sino que estarán en un archivo de texto independiente que será cargado por el plugin **text.js** de RequireJS.
- **Internacionalización de cadenas (i18n)** Los mensajes que genere la aplicación estarán internacionalizados usando el plugin **i18n.js** de RequireJS.

NOTA IMPORTANTE:

Este tutorial está dirigido a personas que ya tengan conocimientos sobre JavaScript, JQuery, Underscore, Require.js y Backbone.
Lo publico para que sirva de guía de consulta, pues hay muy pocos ejemplos en donde se puedan ver estas tecnologías trabajando juntas.

Si no es tu caso y te interesa aprender te recomiendo que veas la sección referencias en la parte inferior de este tutorial.

Aplicación a construir. [Ver aplicación en vivo](#)

Captura de pantalla de la aplicación a construir:

Catálogo de servicios Autentia



Síguenos a través de:



Últimas Noticias

» Autentia estuvo en el Duatlón de Torrejón de Ardoz

» Participamos en la Carrera de las Empresas 2012

» ¡¡¡Terrakas 1x04 recién salido del horno!!!

» Estreno Terrakas 1x04: "Terraka por un día"

» Nuevos cursos de gestión de la configuración en IOS y Android

[Histórico de noticias](#)

Últimos Tutoriales

» Roboelectric: aplicando TDD en Android

» WordPress vs. Joomla

» Tu aplicación web ZK enfoque MVVM (5-5)

» Tu aplicación web ZK enfoque MVC (4-5)

» Añadiendo reconocimiento de gestos a nuestra aplicación desde StoryBoard

Últimos Tutoriales del

Autor

» Creación de un API REST protegido por OAuth2

» JQuery Mobile: Desarrollo de aplicaciones y portales web para dispositivos móviles

» Geoposicionamiento Web con HTML5 y Google Maps

» Validación de formularios con el plugin JQuery Validator

» Instalación y uso del plugin de comentarios de Facebook en nuestra Web

Últimas ofertas de empleo

2011-09-08

 Comercial - Ventas - MADRID.

2011-09-03

 Comercial - Ventas - VALENCIA.

2011-08-19

 Comercial - Compras - ALICANTE.

2011-07-12

 Otras Sin catalogar - MADRID.

2011-07-06

 Otras Sin catalogar - LUGO.

Ordenar Palabras

Ordene las siguientes palabras según el enunciado, puede hacer click, arrastrarlas a los huecos o escribir la respuesta directamente.

Título:

Ordene de las siguientes provincias españolas de norte a sur

1. Barcelona
2. Madrid
3. Granada

Solución:

1. X
2. X
3. X

Alias: alias1

Tiempo: 13

Nº Intentos: 2 / 2

Cancelar

Aceptar

Lo sentimos pero no ha respondido correctamente.

Los requisitos funcionales son:

Una actividad tiene:

- Un enunciado y un numero indeterminado de palabras a ser organizadas según las instrucciones del enunciado.
- El usuario podrá responder haciendo click en la palabra (en cuyo caso se colocará en el primer hueco libre), arrastrándola o escribiéndola directamente.
- Un tiempo máximo en segundos para ser contestada, pasado ese tiempo el usuario ya no podrá responder. Además cuando queden 20 segundos el contador cambiará a color rojo.
- Un número máximo de intentos, además el indicador de número de intentos cambiará a rojo al primer fallo que cometa el usuario.
- Al hacer clic en cancelar se borrarán todas las respuestas que el usuario habia dado hasta el momento.
- Al hacer clic en la X se borrarán todas las respuestas asociada.
- La aplicación deberá controlar que el usuario no introduzca palabras que no formen parte del enunciado previniéndose así errores de introducción de datos.

Código fuente de la aplicación

Captura de pantalla de la estructura de archivos y directorios:

```

▼ resources
  ► css
  ▼ js
    ▼ activities
      ▼ sortwords
        ▼ models
          activity.js
          sortWordsActivity.js
          sortWordsResponse.js
          sortWordsResponses.js
          userActivityAnswer.js
        ▼ views
          ▼ nls
            ▼ es
              strings.js
              strings.js
            ▼ templates
              sortwords-response-template.html
              sortwords-result.html
              sortwords-template.html
              sortwords-timeout-result.html
            activityView.js
            sortWordResponseAnswerView.js
            sortWordResponseView.js
            sortWordResultView.js
          sortwords.js
        ▼ helpers
          ► jasmine
            backbone.js
            handlebars-1.0.0.beta.6.js
            i18n.js
            jquery.js
            jquery.validate.min.js
            json2.js
            messages_es.js
            r.js
            require.js
            text.js
            underscore.js
          sortwords.html

```

El HTML que importa el JS principal de la aplicación

El Widget JavaScript se autoenlazará al DOM en la capa con id "activity-container".

sortwords.html

```

view plain print ?
01. <html>
02. <head>
03.   <title>Ejemplo de Widget con Backbone, RequireJS</title>
04.   <script data-
05.     main="resources/js/activities/sortwords/sortwords" src="resources/js/helpers/require.js"></script>
06.   <link rel="stylesheet" href="resources/css/activities.css"/>
07. </head>
08. <body>
09.   <header>
10.     <h1>Creación de un Widget con Backbone y RequireJS</h1>
11.   </header>
12.   <div id="activity-container"></div>
13.   <footer>
14.     <p>Creado por <a href="http://carlos-garcia.es" target="_blank">http://carlos-
15.     garcia.es</a></p>
16.   </footer>
17. </body>
18. </html>
19.

```

El JS principal de la aplicación

resources/js/activities/sortwords/sortwords.js

```

view plain print ?
01. require.config( {
02.   baseUrl: "resources/",
03.   paths: {
04.     "jquery": "js/helpers/jquery",
05.     "underscore": "js/helpers/underscore",
06.     "json2": "js/helpers/json2",
07.     "backbone": "js/helpers/backbone",
08.     "text": "js/helpers/text",
09.     "i18n": "js/helpers/i18n",
10.     "activity": "js/activities/sortwords/models/activity",
11.     "sortWordsActivity": "js/activities/sortwords/models/sortWordsActivity",
12.     "sortWordsResponses": "js/activities/sortwords/models/sortWordsResponses",
13.     "sortWordsResponse": "js/activities/sortwords/models/sortWordsResponse",
14.     "userActivityAnswer": "js/activities/sortwords/models/userActivityAnswer",
15.     "activityView": "js/activities/sortwords/views/activityView",
16.     "sortWordResponseView": "js/activities/sortwords/views/sortWordResponseView",
17.     "sortWordResponseAnswerView": "js/activities/sortwords/views/sortWordResponseAnswerView",
18.     "sortWordResultView": "js/activities/sortwords/views/sortWordResultView",
19.   },
20.   shim: {
21.     underscore: {
22.       exports: "_"
23.     },
24.     backbone: {
25.       deps: ["underscore", "jquery"],
26.       exports: "Backbone",
27.     },
28.   },
29.   waitSeconds: 10,
30. });
31.
32. require(["backbone", "underscore", "activity", "sortWordsActivity", "sortWordsResponse", "activityView",
33.   function(Backbone, _, Activity, SortWordsActivity, SortWordsResponse, ActivityView){
34.     console.log("sortwords module loaded");
35.
36.     Backbone.emulateHTTP = true;
37.
38.     /*
39.     var sortWordsActivity = new SortWordsActivity({id: 1});
40.     sortWordsActivity.fetch({
41.       success: function () {
42.         var v1 = new ActivityView({model: sortWordsActivity});
43.         v1.render();
44.       },
45.       error: function (collection, response) {
46.         alert("Error");
47.       }
48.     });
49.     */
50.
51.     var sortWordsActivity = new SortWordsActivity({id: 1, alias: "alias1", timeLimit: 30, attempts: 10});
52.     sortWordsActivity.addWord(new SortWordsResponse({word: "Barcelona", position: 0}));
53.     sortWordsActivity.addWord(new SortWordsResponse({word: "Madrid", position: 1}));
54.     sortWordsActivity.addWord(new SortWordsResponse({word: "Granada", position: 2}));
55.
56.     var v1 = new ActivityView({model: sortWordsActivity});
57.     v1.render();
58.   });
59.

```

Modelo de la aplicación

resources/js/activities/sortwords/models/activity.js

```

view plain print ?
01. define("activity", ["backbone"], function(Backbone){
02.   console.log("activity module loaded");
03.
04.   var Activity = Backbone.Model.extend({
05.     url: "http://localhost:8080/mobiletest/activities/",
06.     defaults: {
07.       id: "",
08.       alias: "aliasPorDefecto",
09.       category: "",
10.       title: "",
11.       category: ""

```

```

12.         description: "",
13.         definition: "",
14.         timelimit: 0,
15.         attempts: 0,
16.     }
17. });
18. return Activity;
19. });
20.

```

resources/js/activities/sortwords/models/sortWordsActivity.js

```

view plain print ?
01. define("sortWordsActivity", ["activity", "sortWordsResponses", "sortWordsResponse", "userActivityAnsi
02. {
03.     console.log("sortWordsActivity module loaded");
04.
05.     var SortWordsActivity = Activity.extend({
06.         url: function() {
07.             var base = "http://localhost:8080/mobiletest/activities/";
08.             if (this.isNew()) return base;
09.             return base + (base.charAt(base.length - 1) == '/' ? '' : '/') + this.id;
10.         },
11.         defaults: {
12.             words: new SortWordsResponses(),
13.             answers: new SortWordsResponses(),
14.             category: "SortWords",
15.         },
16.         parse: function(obj) {
17.             for (var i = 0; i < obj.words.length; i++){
18.                 var word = obj.words[i];
19.                 this.get("words").add(new SortWordsResponse({position: word.position, word: word.word});
20.                 this.get("answers").add(new SortWordsResponse({position: i, word: ""}), {silent: true});
21.             }
22.             delete obj.words;
23.
24.             // Now backbone init rest properties
25.             return obj;
26.         },
27.         getWordCount: function() {
28.             return this.get("words").length;
29.         },
30.         getAnswers: function() {
31.             return this.get("answers");
32.         },
33.         addWord: function(w) {
34.             this.get("words").add(w, {silent: true});
35.             this._addEmptyAnswerGapForWord(w);
36.         },
37.         _addEmptyAnswerGapForWord: function(word){
38.             var index = this.getWordCount() - 1;
39.             var answer = new SortWordsResponse({position: index, word: ""});
40.             this.get("answers").add(answer, {silent: true});
41.         },
42.         clearAnswers: function(){
43.             var i = 0;
44.             this.get("answers").each(function(answer){
45.                 answer.set({"position": i++});
46.                 answer.clearWord();
47.             });
48.         },
49.         isValid: function(){
50.             var answers = this.get("answers");
51.             var words = this.get("words");
52.             var toReturn = true;
53.
54.             var numWords = this.getWordCount();
55.             for (var i = 0; i < numWords; i++){
56.                 var word = words.at(i); // TODO: Sort by position first?
57.                 var answer = answers.at(i);
58.                 if (!word.equals(answer)){
59.                     toReturn = false;
60.                     break;
61.                 }
62.             }
63.             return toReturn;
64.         },
65.     });
66.
67.     // Enable property inheritance
68.     SortWordsActivity.prototype.defaults = _.extend(Activity.prototype.defaults, SortWordsActivity.p
69.
70.     return SortWordsActivity;
71. });
72.

```

resources/js/activities/sortwords/models/sortWordsReponse.js

```

view plain print ?
01. define("sortWordsResponse", ["backbone"], function(Backbone){
02.     console.log("sortWordsResponse module loaded");
03.
04.     var SortWordsResponse = Backbone.Model.extend({
05.         defaults: {
06.             word: "",
07.         },
08.         equals: function(other){
09.             return ((this.get("word") == other.get("word")) && (this.get("position") == other.get("po
10.         },
11.         clearWord: function(){
12.             this.set({"word": ""});
13.         }
14.     });
15.     return SortWordsResponse;
16. });
17.

```

resources/js/activities/sortwords/models/sortWordsReponses.js

```

view plain print ?
01. define("sortWordsResponses", ["sortWordsResponse", "backbone"], function(SortWordsResponse, Backbone)
02. {
03.     console.log("SortWordsResponses module loaded");
04.     var SortWordsResponses = Backbone.Collection.extend({
05.         model: SortWordsResponse,
06.     });
07.     return SortWordsResponses;
08. });
09.
10.

```

Vistas de la aplicación.

resources/js/activities/sortwords/views/activityView.js

```

view plain print ?
01. define("activityView", ["jquery", "underscore", "backbone", "sortWordsActivity", "userActivityAnswer",
02. "sortWordResultView", "text!js/activities/sortwords/views/templates/sortwords-
03. template.html", "i18n!js/activities/sortwords/views/nls/strings"],
04. function($, _, Backbone, SortWordsActivity, UserActivityAnswer, SortWordResponseView, SortWordRe:
05. {
06.     var ActivityView = Backbone.View.extend({
07.         el: $("#activity-container"),
08.         template: _.template(sortwords_template),
09.         initialize: function() {
10.             this.bind("mobiletest:activity:finished", this.onAttemptsFinished, this);
11.             this.bind("mobiletest:activity:timeEnd", this.onTimeFinished, this);
12.             this.model.bind("change", this.render, this);
13.             this.numFails = 0;
14.             this.timeLimit = this.model.get("timelimit");
15.             this.timerID = setInterval(function(){this.tick();}.bind(this), 1000);
16.         },
17.         events: {
18.             "click #cancel": "clearUserAnswers",
19.             "click #accept": "doAccept",
20.         },
21.         render: function() {
22.             this.model.set({"i18n": str});
23.             $(this.el).html(this.template(this.model.toJSON()));
24.             var model = this.model;
25.             var words = this.model.get("words");
26.             var answers = this.model.get("answers");
27.             var palabras = _.shuffle(_.toArray(words));
28.             for (var i = 0; i < palabras.length; i++){
29.                 var w = palabras[i];
30.                 var view = new SortWordResponseView({model: w, sortWordActivity: model});
31.                 $("#responses").append(view.render().el);
32.                 var answer = answers.at(i);
33.                 var viewAnswer = new SortWordResponseAnswerView({model: answer});
34.                 $("#user-response ol").append(viewAnswer.render().el);
35.             }
36.             return this;
37.         },
38.         doAccept: function(ev){
39.             var isActivityOkAnswered = this.model.isValid();
40.             this._loadAnswersModelFromUI();
41.             // ¿Is set num attempts Limit?
42.             if (this.model.get("attempts") > 0){
43.                 if (isActivityOkAnswered){
44.                     this.trigger("mobiletest:activity:finished");
45.                 } else {
46.                     this.numFails++;
47.                     $("#fails").text(this.numFails);
48.                     $("#fails").addClass("activityWithFailAttempts");
49.                     if (this.numFails >= this.model.get("attempts")){
50.                         this.trigger("mobiletest:activity:finished");
51.                     }
52.                 }
53.             } else {
54.                 if (this._verifyExistAllWords()){
55.                     this.trigger("mobiletest:activity:finished");
56.                 } else {
57.                     $("#alert-incomplete-activity").show();
58.                 }
59.             }
60.         },
61.         onAttemptsFinished: function(){
62.             this._disableUserGUIControls();
63.             var userActivityAnswer = new UserActivityAnswer();
64.             userActivityAnswer.set({"activityId": this.model.get("id")});
65.             userActivityAnswer.set({"attempts": this.numFails});
66.             userActivityAnswer.set({"result": JSON.stringify(this.model.getAnswers())});
67.             userActivityAnswer.save();
68.             var resultView = new SortWordResultView({model: this.model});
69.             resultView.render();
70.         },
71.         tick: function(){
72.             this.timeLimit--;
73.             $("#timelimit").text(this.timeLimit);
74.             if (this.timeLimit <= 0){
75.                 this.trigger("mobiletest:activity:timeEnd");
76.             }
77.         }
78.     });
79. }
80.
81.
82.
83.
84.
85.
86.

```

```

87.         } else if (this.timelimit <= 20){
88.             $("#timelimit").addClass("littleTime");
89.         }
90.     },
91.     onTimeFinished: function(){
92.         this._disableUserGUIControls();
93.         var resultView = new SortWordResultView({model: this.model, playTimeout: true});
94.         resultView.render();
95.     },
96.     resetTimer: function(){
97.         if (this.model.get("timelimit") > 0){
98.             clearInterval(this.timerID);
99.         }
100.    },
101.    clearUserAnsweres : function(ev) {
102.        this.model.clearAnswers();
103.        $("#alert-incomplete-activity").hide();
104.    },
105.    _disableUserGUIControls: function(){
106.        this.resetTimer();
107.        $("#cancel").attr("disabled", true);
108.        $("#accept").attr("disabled", true);
109.    },
110.    _loadAnsweresModelFromUI: function(){
111.        var wordsWritten = $("input[name='response']");
112.        var modelRef = this.model;
113.        this.model.clearAnswers();
114.
115.        $.each(wordsWritten, function(index, value) {
116.            var currentSortWordsResponse = modelRef.get("answeres").at(index);
117.            currentSortWordsResponse.set({"word": $(value).val()});
118.        });
119.    },
120.    _verifyExistAllWords: function(){
121.        var answeres = this.model.get("answeres");
122.        var words = this.model.get("words");
123.        var toReturn = true;
124.
125.        for (var i = 0; i < answeres.length; i++){
126.            var answerText = answeres.at(i).get("word");
127.            var existWord = false;
128.
129.            for (var j = 0; j < words.length; j++){
130.                var wordText = words.at(j).get("word");
131.                if (answerText == wordText){
132.                    existWord = true;
133.                    break;
134.                }
135.            }
136.
137.            if (! existWord){
138.                toReturn = false;
139.                break;
140.            }
141.        }
142.        return toReturn;
143.    },
144.    });
145.
146.    return ActivityView;
147.
148.    });
149.

```

resources/js/activities/sortwords/views/sortWordResponseAnswerView.js

```

view plain print ?
01. define("sortWordResponseAnswerView", ["jquery", "underscore", "backbone", "text!js/activities/sortwo
response-template.html"], function($, _, Backbone, sortwords_response_template){
02.     console.log("SortWordResponseAnswerView module loaded");
03.
04.     var SortWordResponseAnswerView = Backbone.View.extend({
05.         tagName: "li",
06.         template: _.template(sortwords_response_template),
07.         initialize: function() {
08.             this.model.bind("change", this.render, this);
09.         },
10.         render: function() {
11.             $(this.el).html(this.template(this.model.toJSON()));
12.             return this;
13.         },
14.         events: {
15.             "click .delete" : "clearResponseUI",
16.             "change input[name='response']": "updateAnswerModel",
17.         },
18.         clearResponseUI: function(ev){
19.             this.model.clearWord();
20.             return true;
21.         },
22.         updateAnswerModel: function(ev){
23.             var newValue = $(ev.target).val();
24.             this.model.set({"word": newValue});
25.             return true;
26.         },
27.     });
28.
29.     return SortWordResponseAnswerView;
30.
31. });
32.

```

Plantillas de las vistas de la aplicación

resources/js/activities/sortwords/views/templates/sortwords-template.html

view plain print ?

```

01. <div id="activity-view">
02.   <header id="activity-title">
03.     <h2><%= i18n.activityType %></h2>
04.     <p><%= i18n.instructions %></p>
05.   </header>
06.
07.   <div id="activity-content">
08.     <form id="activity-form" name="activity-form" action="" method="post">
09.       <input type="hidden" name="activityID" value="<%= id %>" />
10.       <div id="enunciado" class="fields">
11.         <h2><%= i18n.title %></h2>
12.         <p><%= description %></p>
13.         <ol id="responses" />
14.       </div>
15.
16.       <div id="meta-panel">
17.         <p id="alias"><em><%= i18n.alias %>: </em><%= alias %></p>
18.         <p id="timerPanel"><em><%= i18n.timeLimit %>: </em> <span id="timeLimit"><%= timeLimit %></span>
19.         <p id="attemptsPanel"><em><%= i18n.attempts %>: </em> <span id="fails"><%= fails %></span> /
20.       </div>
21.
22.       <div id="user-response">
23.         <h2><%= i18n.answers %></h2>
24.         <ol class="horizontal"></ol>
25.       </div>
26.
27.       <div class="fields" id="buttons">
28.         <input type="button" class="boton" id="cancel" name="cancel" value="
29. <%= i18n.cancel %>" />
30.         <input type="button" class="boton" id="accept" name="accept" value="
31. <%= i18n.accept %>" />
32.       </div>
33.       <div id="alert-incomplete-activity" class="invisible error"><%= i18n.alertIncompleteActivity %></div>
34.       <div id="activity-result" />
35.     </form>
36.   </div>
37. </div>
38.

```

resources/js/activities/sortwords/views/templates/sortwords-response-template.html

```

view plain print ?
01. <div class="sortwords-response">
02.   <input name="response" type="text" size="25" value="<%= word %>" />
03.   <span class="delete">X</span>
04. </div>
05.

```

resources/js/activities/sortwords/views/templates/sortwords-result.html

```

view plain print ?
01. <div id="activity-result-view">
02.   <div id="alert-activity-complete-ko" class="invisible"><%= i18n.alertActivityCompleteKO %></div>
03.   <div id="alert-activity-complete-ok" class="invisible"><%= i18n.alertActivityCompleteOK %></div>
04. </div>

```

resources/js/activities/sortwords/views/templates/sortwords-timeout-result.html

```

view plain print ?
01. <div id="activity-result-view">
02.   <p><%= i18n.activityResultTimeout %></p>
03. </div>

```

Internacionalización de los mensajes de la aplicación

resources/js/activities/sortwords/views/nls/es/strings.js

```

view plain print ?
01. define({
02.   cancel: "Cancelar",
03.   accept: "Aceptar",
04.   activityType: "Ordenar Palabras",
05.   instructions: "Ordene las siguientes palabras según el enunciado, puede hacer click, arrastrarlas ;",
06.   title: "Título",
07.   alias: "Alias",
08.   enunciado: "Enunciado",
09.   anweres: "Solución",
10.   alertIncompleteActivity: "Actividad incompleta o contiene palabras que no forman parte del enunciado",
11.   alertActivityCompleteKO: "Lo sentimos pero no ha respondido correctamente.",
12.   alertActivityCompleteOK: "Felicitades! La actividad se ha realizado con éxito.",
13.
14.   attempts: "Nº Intentos",
15.   timeLimit: "Tiempo",
16.   activityResultTitle: "Resultado de la actividad",
17.   activityResultTimeout: "Lo sentimos, el tiempo ha terminado.",
18.
19. });
20.

```

Referencias

- Backbone.js
- Tutoriales de Backbone.js
- Charla de una hora sobre Backbone.js
- Why AMD?
- RequireJS
- RequireJS Optimizer

- [El libro "Developing Backbone.js Applications"](#)
- [El libro "Recipes With Backbone.js"](#)

Conclusiones

Yo no soy muy seguidor de modas de programación, me gusta conocerlas para poder opinar con criterio.

En la actualidad hay una tendencia a delegar cada vez más lógica de negocio a los clientes, descargándose los servidores del trabajo de renderización y dejándoles que se ocupen del acceso a los datos a través de un api bien definido.

Yo personalmente no las veo gran futuro para portales transaccionales, principalmente debido la curva de aprendizaje y al SEO (aunque hay estándares para indexación de Javascript). Si lo veo en aplicaciones que forman parte de estos portales (componentes javascript) o aplicaciones independientes.

Claro que existen complejas webs construidas así, por lo que me puedo equivocar :-)

Lo que si puedo afirmar es que no hay motivo para continuar desarrollando código espageti :-)

Bueno, eso es todo, un saludo.

[Carlos García](#).

A continuación puedes evaluarlo:

[Regístrate para evaluarlo](#)

Por favor, vota +1 o compártelo si te pareció interesante

[Share](#) |

[0](#)

Anímate y coméntanos lo que pienses sobre este **TUTORIAL**:

» [Regístrate](#) y accede a esta y otras ventajas «



Esta obra está licenciada bajo licencia [Creative Commons de Reconocimiento-No comercial-Sin obras derivadas 2.5](#)

IMPULSA

Impulsores

Comunidad

[¿Ayuda?](#)

sin clicks

0 personas han traído clicks a esta página

+ + + + + + + +

powered by [karmacracy](#)

Copyright 2003-2012 © All Rights Reserved | [Texto legal y condiciones de uso](#) | [Banners](#) | [Powered by Autentia](#) | [Contacto](#)

