Trabajo de Arquitectura de Software

Arquitectura de aplicaciones Web

Integrantes:

Mengoa Padilla, Raí Paolo

Lima, 14 de Abril de 2016

I N D I C E

INTRODUCCION

En este trabajo de arquitectura de software de aplicaciones web, usted encontrará la información más oportuna sobre los conocimientos obtenidos en el curso de arquitectura de software, y las bases teóricas necesarias para empezar a poner en marcha una aplicación realista en el mundo personal y laboral.

En las primeras páginas se encontrara con la parte teórica del trabajo, escrito por los aplicados alumnos del curso de arquitectura de software. Luego una serie de estudios exhaustivos del tema para una presentación limpia, comprensible y concisa.

En el fondo el trabajo muestra ejercicios con un desarrollo detallado de los mismo, aplicando las últimas tecnologías y arquitecturas empresariales, dicho de otro modo de las tendencias tecnologías pero principalmente el sentido común y analítico aplicado a las soluciones empresariales de los integrantes del grupo.

Por último pero no menos importante, usted encontrara unos prácticos aportes para aprender de la manera más realista, las aplicaciones en el mundo real de los temas llevados y especialmente el mundo de la ingeniería de software y la gestión.

El presente volumen se presenta así: Sección teórica, fundamentos y formulas necesarias. Luego sigue: Ejercicios propuestos, como parte de los datos recogidos, para su posterior análisis e interpretación.

Por último: como aplicar en el mundo real, con énfasis en la ingeniería de software y gestión a modo de conclusión.

El beneficio que este nos trae es complementario a nuestra formación como profesionales de debido al carácter analítico del mismo.

Secretos de óptima elaboración de soluciones, resolución de problemas de la vida real, aspectos quizás desligados aparentemente de la temática acostumbrada, pero no por ello no importantes para las empresas y para nosotros que podemos ofrecer estas mejoras a través de nuestras habilidades.

El ideal de los escritores del presente trabajo, es compartir sus experiencias para que por medio de él usted logre fácilmente lo que a todos ellos les ha costado tanto: mejorar aspectos tal vez en su propia vida, pasar con éxito el curso y tener éxito.

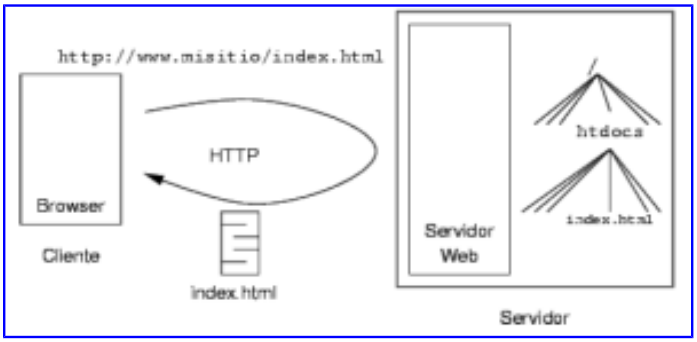
FUNDAMENTO TEÓRICO

Arquitectura de las aplicaciones Web.

Una aplicación Web es proporcionada por un servidor Web y utilizada por usuarios que se Conectan desde cualquier punto vía clientes Web (browsers o navegadores). La arquitectura de un Sitio Web tiene tres componentes principales:

Un servidor Web
Una conexión de red
Uno o más clientes

El servidor Web distribuye páginas de información formateada a los clientes que las solicitan. Los requerimientos son hechos a través de una conexión de red, y para ello se usa el protocolo HTTP. Una vez que se solicita esta petición mediante el protocolo HTTP y la recibe el servidor Web, éste localiza la página Web en su sistema de archivos y la envía de vuelta al navegador que la solicitó.



Las aplicaciones Web están basadas en el modelo Cliente/Servidor que gestionan servidores web, y que utilizan como interfaz páginas web.

Las páginas Web son el componente principal de una aplicación o sitio Web. Los browsers piden páginas (almacenadas o creadas dinámicamente) con información a los servidores Web. En algunos ambientes de desarrollo de aplicaciones Web, las páginas contienen código HTML y scripts dinámicos, que son ejecutados por el servidor antes de entregar la página.

Una vez que se entrega una página, la conexión entre el browser y el servidor Web se rompe, es decir que la lógica del negocio en el servidor solamente se activa por la ejecución de los scripts de las páginas solicitadas por el browser (en el servidor, no en el cliente). Cuando el browser ejecuta un script en el cliente, éste no tiene acceso directo a los recursos del servidor. Hay otros componentes que no son scripts, como los applets (una aplicación especial que se ejecuta dentro de un navegador) o los componentes ActiveX. Los scripts del cliente son por lo general código JavaScript o VBSscript, mezclados con código HTML.

La colección de páginas son en una buena parte dinámicas (ASP, PHP, etc.), y están agrupadas lógicamente para dar un servicio al usuario. El acceso a las páginas está agrupado también en el tiempo (sesión). Los componentes de una aplicación Web son:

1. Lógica de negocio.

Parte más importante de la aplicación.
Define los procesos que involucran a la aplicación.
Conjunto de operaciones requeridas para proveer el servicio.

2. Administración de los datos.

Manipulación de BD y archivos.

3. Interfaz

Los usuarios acceden a través de navegadores, móviles, PDAs, etc.
Funcionalidad accesible a través del navegador.
Limitada y dirigida por la aplicación.

Las aplicaciones web se modelan mediante lo que se conoce como modelo de capas, Una capa representa un elemento que procesa o trata información. Los tipos son:

Modelo de dos capas: La información atraviesa dos capas entre la interfaz y la administración de los datos.
Modelo de n-capas: La información atraviesa varias capas, el más habitual es el modelo de tres capas.

Modelo de dos Capas.

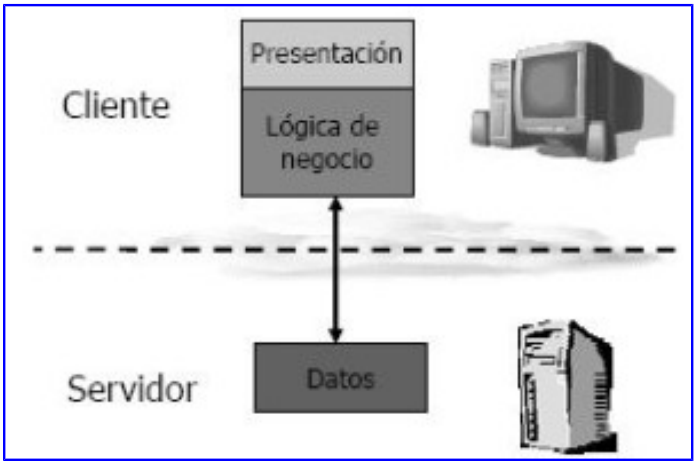
Gran parte de la aplicación corre en el lado del cliente (fat client).

Las capas son:

Cliente (fat client): La lógica de negocio está inmersa dentro de la aplicación que realiza el interfaz de usuario, en el lado del cliente.
Servidor: Administra los datos.

Las limitaciones de este modelo son.

Es difícilmente escalable
Número de conexiones reducida
Alta carga de la red.
La flexibilidad es restringida
La funcionalidad es limitada.



Modelo de tres Capas.

Esta diseñada para superar las limitaciones de las arquitecturas ajustadas al modelo de dos capas, introduce una capa intermedia (la capa de proceso) Entre presentación y los datos, los procesos pueden ser manejados de forma separada a la interfaz de usuari o y a los datos, esta capa intermedia centraliza la lógica de negocio, haciendo la administración más sencil a, los datos se pueden integrar de múltiples fuentes, las aplicaciones web actuales se ajustan a este modelo.

Las capas de este modelo son:

1. Capa de presentación (parte en el cliente y parte en el servidor)

   Recoge la información del usuario y la envía al servidor (cliente)
   Manda información a la capa de proceso para su procesado
   Recibe los resultados de la capa de proceso
   Generan la presentación
   Visualizan la presentación al usuario (cliente)

2. Capa de proceso (servidor web)

   Recibe la entrada de datos de la capa de presentación
   Interactúa con la capa de datos para realizar operaciones
   Manda los resultados procesados a la capa de presentación

3. Capa de datos (servidor de datos)

   Almacena los datos
   Recupera datos
   Mantiene los datos
   segura la integridad de los datos

Uso del patron MVC

El patrón de arquitectura Modelo-Vista-Controlador (MVC) separa una aplicación en tres componentes principales: el modelo, la vista y el controlador. El marco ASP.NET MVC proporciona una alternativa al modelo de formularios Web ASP.NET para crear aplicaciones Web.

MVC es un patrón de diseño estándar que muchos desarrolladores están familiarizados. Algunos tipos de aplicaciones Web se beneficiarán del marco MVC. Otros seguirán utilizando el patrón tradicional de aplicación ASP.NET que se basa en las solicitudes web y las devoluciones de datos. Otros tipos de aplicaciones Web se combinarán los dos enfoques; ni el planteamiento excluye a la otra.
El marco MVC incluye los siguientes componentes:

•**Modelos.** Los objetos del modelo son las partes de la aplicación que implementan la lógica de dominio de datos de la aplicación. A menudo, los objetos del modelo y recuperar el estado tienda de modelo en una base de datos. Por ejemplo, un producto objeto podría recuperar información de una base de datos, operar en él, y luego escribir la información actualizada en la tabla productos en una base de datos SQL Server.
En aplicaciones pequeñas, el modelo es a menudo una separación conceptual en lugar de una física. Por ejemplo, si la aplicación sólo lee un conjunto de datos y lo envía a la vista, la aplicación no tiene una capa del modelo físico y clases asociadas. En ese caso, el conjunto de datos asume el papel de un objeto modelo.

•**Vistas.** Las vistas son los componentes que muestran la interfaz de usuario de la aplicación (interfaz de usuario). Por lo general, esta interfaz de usuario se crea a partir de los datos del modelo. Un ejemplo sería una vista de edición de una tabla que muestra los productos cuadros de texto, listas desplegables y casillas de verificación basadas en el estado actual de un producto objeto.

•**Controladores**. Los controladores son los componentes que manejan la interacción con el usuario, el trabajo con el modelo, y finalmente seleccionar una vista para hacer que muestra la interfaz de usuario. En una aplicación MVC, la vista sólo muestra la información; el controlador maneja y responde a la entrada del usuario y la interacción. Por ejemplo, el controlador trata los valores de cadena de consulta, y pasa estos valores en el modelo, que a su vez podría utilizar estos valores para consultar la base de datos.

•

El patrón MVC le ayuda a crear aplicaciones que separan los diferentes aspectos de la aplicación (lógica de entrada, la lógica de negocio, y la lógica de interfaz de usuario), mientras que proporciona un acoplamiento débil entre estos elementos. El patrón especifica que cada tipo de lógica se debe colocar en la solicitud. La lógica de la interfaz de usuario pertenece en la vista. lógica de entrada pertenece en el controlador. La lógica de negocio pertenece en el modelo. Esta separación ayuda a gestionar la complejidad cuando se genera una aplicación, ya que le permite centrarse en un aspecto de la aplicación a la vez. Por ejemplo, usted puede centrarse en la vista sin depender de la lógica de negocio.
El acoplamiento débil entre los tres componentes principales de una aplicación MVC también promueve el desarrollo paralelo. Por ejemplo, un desarrollador puede trabajar en la vista, un segundo desarrollador puede trabajar en la lógica del controlador, y un tercero desarrollador puede concentrarse en la lógica de negocio en el modelo.

Uso de framework Front-End / Web Services

El uso de libreria en la capa de presentación a evolucionado tanto en hoy en día se pueden encontrar un cantidad considerable de framework de desarrollo principalmente desarrollando en javascript, así como embedidos en el lenguaje o una mescla de ambos.

Principales caracteristicas:

Data Binding (Enlace de datos) – permite el mapeo de datos en base de un modelo de datos, permite tanto la carga, modificación y actualización de datos, de manera muy practica y sencilla

UI widget(controles personalizados) – otra característica de los framework es el manejo de una gama de controles personalizados, generalmente orientados a formularios de datos, reportes y estadisticos.

AJAX(*Asynchronous JavaScript And XML)* – generalmente la comunicación con los componentes back-end se realiza mediante ajax, sea llamando a web services o recursos en json.

Framework más conocidos

AngularJS, o simplemente **Angular**, es un framework de JavaScript de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página. Su objetivo es aumentar las aplicaciones basadas en navegador con

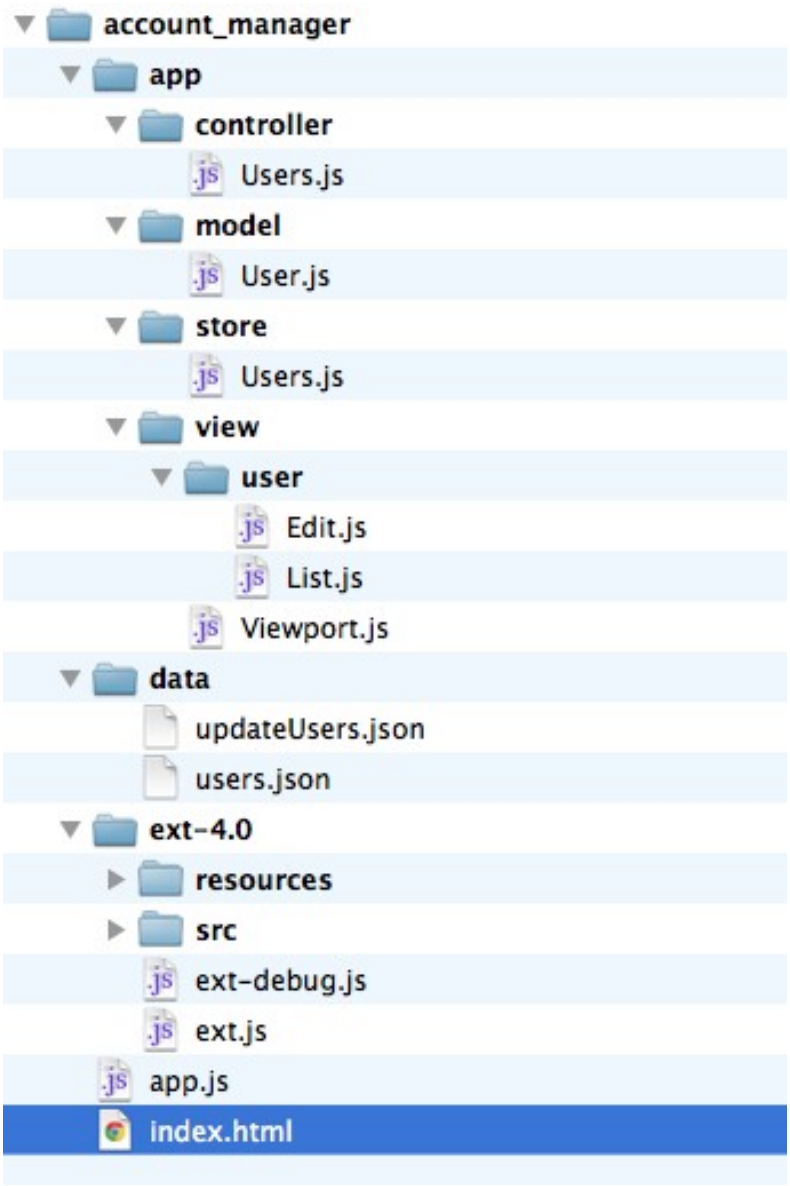
capacidad de Modelo Vista Controlador (MVC), en un esfuerzo para hacer que el desarrollo y las pruebas sean más fáciles.
La biblioteca lee el HTML que contiene atributos de las etiquetas personalizadas adicionales, entonces obedece a las directivas de los atributos personalizados, y une las piezas de entrada o salida de la página a un modelo representado por las variables estándar de JavaScript. Los valores de las variables de JavaScript se pueden configurar manualmente, o recuperados de los recursos JSON estáticos o dinámicos.

Ext JS es una biblioteca de JavaScript para el desarrollo de aplicaciones web interactivas usando tecnologías como AJAX, DHTML y DOM. Fue desarrollada por Sencha.
Originalmente construida como una extensión de la biblioteca YUI por Jack Slocum, en la actualidad puede usarse como extensión para las biblioteca jQuery y Prototype. Desde la versión 1.1 puede ejecutarse como una aplicación independiente.

jQWidgets ofrece una solución completa para la creación de sitios web profesionales y aplicaciones móviles. Está construido en su totalidad en estándares abiertos y tecnologías como HTML5, CSS, Javascript y jQuery. jQWidgets permite el desarrollo web de respuesta y ayuda a crear aplicaciones y sitios web que se ven hermosas en los escritorios, tabletas y teléfonos inteligentes. jQWidgets es un marco completo de características con los widgets de profesionales táctiles jQuery, temas, la validación de entrada, arrastrar y soltar los plug-ins, datos adaptadores, una función de accesibilidad WAI-ARIA, la internacionalización y el apoyo MVVM.

FUNDAMENTO PRÁCTICO / EJEMPLOS

Contexto general basado en MVC con el framework EXT JS.



Clase de inicio (App.js)

Cada aplicación Ext. JS 4 se inicia con una instancia de la clase Application . La instancia de Application contiene la configuración global de su aplicación (por ejemplo, el nombre de la aplicación), así como mantiene referencias a todos los modelos, vistas y controladores utilizados por la aplicación. Una aplicación también contiene una función de lanzamiento, que se ejecuta automáticamente cuando todo está cargado.

```
Ext.application({
    name: 'AM',

    appFolder: 'app',

    launch: function() {
        Ext.create('Ext.container.Viewport', {
            layout: 'fit',
            items: [
                {
                    xtype: 'panel',
                    title: 'Users',
                    html : 'List of users will go here'
                }
            ]
        });
    }
});
```

Definir un controlador

Los controladores son como el pegamento que une todas las partes de la aplicación entre si. Todo lo que realmente hacen es escuchar los eventos (por lo general de vistas) y realizar algunas acciones cuando lleguen. Continuando con nuestra aplicación de Account Manager, vamos a crear un controlador.

```
Ext.define('AM.controller.Users', {
    extend: 'Ext.app.Controller',

    init: function() {
        this.control({
            'viewport &gt; panel': {
                render: this.onPanelRendered
            }
        });
    },

    onPanelRendered: function() {
        console.log('The panel was rendered');
    }
});
```

Definir una vista

Hasta ahora nuestra aplicación sólo ha sido unas pocas líneas y sólo esta contenida en dos archivos – app.js y app/controller/Users.js. Ahora que queremos agregar una grilla que muestre todos los usuarios de nuestro sistema, es el momento de organizar nuestra lógica un poco mejor y empezar a usar vistas.

```
Ext.define('AM.view.user.List' ,{
    extend: 'Ext.grid.Panel',
    alias : 'widget.userlist',

    title : 'All Users',

    initComponent: function() {
        this.store = {
            fields: ['name', 'email'],
            data  : [
                {name: 'Ed',    email: 'ed@sencha.com'},
                {name: 'Tommy', email: 'tommy@sencha.com'}
            ]
        };

        this.columns = [
            {header: 'Name',  dataIndex: 'name',  flex: 1},
            {header: 'Email', dataIndex: 'email', flex: 1}
        ];

        this.callParent(arguments);
    }
});
```

BIBLIOGRAFIA

- Angular JS

  https://es.wikipedia.org/wiki/Angular_JS

- EXT JS

  https://es.wikipedia.org/wiki/Ext_JS

- Ajax

  https://es.wikipedia.org/wiki/AJAX

- Arquitectura web

  https://programacionwebisc.wordpress.com/2-1-arquitectura-de-las-aplicaciones-web/

- JQWidgets

  http://www.jqwidgets.com/

- Ext JS ejemplo

  http://www.marioperez.com.mx/extjs/mvc/