Prueba técnica – Keylin Hidalgo Barrios

Tecnologías requeridas: Asp.Net (lenguaje C#), HTML, Javascript (jquery, ajax), git.

Forma de entrega: Repositorio en github

Fecha máxima de entrega: Domingo 31 de marzo de 2013 hasta las 11:59:59pm

Enunciado

El proyecto consiste en demostrar habilidad de investigación y aplicación de conocimientos básicos en las tecnologías relevantes para el proyecto. Debe incluir un README file con instrucciones especiales o requerimientos, cualquier cosa relevante para hacer funcionar el proyecto.

Parte I – RESTful API (valor 50%)

Se implementará un webservice RESTful con ASP.Net que autenticará a un usuario y devolverá datos sobre este (a manera de perfil). La comunicación se hará utlizando JSON, **no** XML (tanto para recibir como para retornar datos). No es necesario que haya conexión con base de datos, los datos pueden estar fijos en el webservice y mantener lo que necesite en memoria.

Se tendrán 3 endpoints/métodos públicos como se especifican a continuación:

URI	**Método**	**Datos que recibe**	**Resultados que devuelve**
api/login	POST	Usuario, contraseña	Resultado (true/false) y token o mensaje de error.
api/perfil/*token*	GET	Solo el token en el URL, no en body.	Un Json con info de perfil de usuario (nombre, apellidos, email, dirección, teléfono).
api/logout/*token*	DELETE	Solo el token en el URL, no en body.	Un Json con un resultado (true o false).

El método login enviará por POST un json con el usuario y contraseña a validar. Si el usuario no existe o el password es incorrecto, deberá devolver false y el mensaje de error adecuado. Si la validación es correcta, debe devolver un token que será un número/cadena aleatoria generada por el servidor y mantenida en memoria. Ejemplo:

{“resultado”: “false”, “mensaje”: “usuario no existe”}

{“resultado” : “true”, “token”: “111000999ABCD”}

El método perfil recibe por GET un token en el URI. El servidor validará que el token sea válido (previamente generado por login, mantenido en memoria para simplificar) y devolverá un json con

datos de perfil (mínimo los mencionados arriba) o bien, un error si el token es inválido.

El método logout, recibe por DELETE un token. El servidor verifica que el token sea válido y lo borra de su memoria, devolviendo true en un json de resultado. Si el token es falso, el servidor devuelve un false en un json de resultado.

En todos los casos, si el endpoint se llama con otro método que no sea el que se especificó, el servidor debe fallar con un código HTTP 405.

Parte II– API consumer / web client (valor 40%)

La segunda parte del proyecto consiste en crear un cliente que consuma el API creado en la parte I. Para esto se usará **solo HTML y javascript**. La comunicación debe ser asincrónica usando Ajax (se recomienda usar Jquery). La interface gráfica no tiene que ser elaborada ni detallada, puede ser HTML básico y las notificaciones de errores pueden ser simples alerts (aunque se prefieren Jquery dialogs).

Con el cliente, el usuario deberá poder hacer login (indicando un usuario y contraseña). Si el login es exitoso, debe llamar al endpoint para obtener datos de perfil, y mostrarselos al usuario en una simple tabla HTML. Estando logeado, el usuario deberá poder hacer log out.

Entrega (valor 10%)

Vamos a manejar el proyecto en un repositorio de Github. Si no tiene cuenta en Github, debe crear una. El repositorio a utilizar es: https://github.com/jeudycecropia/prueba_keilyn.

Debe subir todo su código fuente a ese repositorio. No es necesario tener varios branches, puede utilizar master todo el tiempo. El último commit al proyecto no debe hacerse después de la fecha máxima de entrega.

Puntos extra

Usar una base de datos y aplicar MVC.

Referencias

http://www.asp.net/web-api

http://docs.kendoui.com/tutorials/ASP.NET/asp-net-hello-services