Guía parte 1: Descubriendo MVC

Producto

En esta guía vas a completar las funcionalidades de `agregarPregunta` y `borrarPregunta`. Como resultado vas a poder agregar un nueva pregunta con sus respuestas asociadas y también borrarlas. Para esto vas a tener que trabajar en el modelo, el controlador y la vista del administrador.

Preparación

Lo primero que vas a hacer es descargar los archivos que vas a usar para este proyecto. A continuación podés encontrar una lista con los principales archivos donde estarás trabajando y una breve descripción de cada uno.

- Archivos `html` y `css`: ya tienen todo lo necesario para que la app se visualice correctamente y **no** es necesario hacer retoques sobre ellos en ningún momento.
- `modelo.js`: Contiene un modelo que representa a las `preguntas`con sus `respuestas`.
- `controlador.js`: se encarga de hacer la conexión entre la `vista`y el `modelo`.
- `vistaAdministrador.js`: vista del administrador desde donde se construyen los elementos `html` para las preguntas y se asocian los eventos a botones
- `vistaUsuario.js`: la vista del usuario donde se muestran las preguntas con sus respuestas y los gráficos de las encuestas.
- `evento.js`: objeto necesario para que el `modelo` pueda notificar sus cambios a las `vistas`. Entender su funcionamiento es de vital importancia, podés volver a ver los videos para recordarlo

en https://www.acamica.com/cursos/354/javascript-patrones/niveles
- `index.js`: archivo donde se inicializan los objetos.
- `libs adicionales`: en esta carpeta se pueden encontrar bibliotecas adicionales que nos brindan funcionalidades que necesitaremos. La más importante de ellas es `validador.js`, ya que tendremos que usarla para el agregado de respuestas.

Paso 1: agregar preguntas

A lo largo de este paso, vas a tener que ir implementando fragmentos en todos los objetos participantes de MVC: el `modelo.js`, el `controlador.js` y la `vistaAdministrador.js`.

Queremos lograr que se puedan agregar preguntas con sus respectivas respuestas y aparezcan visualmente en nuestra app.

Cómo están representadas las preguntas y respuestas

- Una `pregunta` es un diccionario que tiene un `texto`, un `id` y una lista de `cantidad de votos por respuesta`.

Ejemplo de pregunta:

```
{'texto': unTexto, 'id': id, 'cantidadPorRespuesta': respuestas}
```

- Una `respuesta` es un diccionario que tiene un `texto` y una `cantidad de votos`.

Ejemplo de respuesta:

```
{'textoRespuesta': respuesta, 'cantidad': cantVotos}
```

vistaAdministrador: agregado de respuestas

El agregado de respuestas ya viene implementado por defecto. Podés probarlo abriendo `vistaAdministrador.html` y agregar `respuestas` a una pregunta. Esta funcionalidad viene dada por `validacionFormulario()` que se ejecuta en `inicializar` de `vistaAdministrador.js`.

vistaAdministrador: construir el elemento html de una pregunta

Completá el método `construirElementoPregunta` para que construya un elemento html a partir de una pregunta pasada por parámetro (recordá como está formada una pregunta releyendo `Cómo están representadas las preguntas y respuestas`). Este nuevo elemento deberá guardarse en la variable con nombre `nuevoItem` y tener clase `list-group-item`, id `pregunta.id` y texto `pregunta.textoPregunta`.

> Tip: Podés probar si funciona correctamente la construcción del elemento ejecutando desde la consola la siguiente línea: this.modelo.preguntas = [{'textoPregunta': "Mi primer Pregunta", 'id': 0, 'cantidadPorRespuesta': [{'textoRespuesta': "mi unica respuesta", 'cantidad': 2}]}]

vistaAdministrador: completando el inicializador

Agregá los métodos

```
this.reconstruirLista();

this.configuracionDeBotones();
```

al `inicializar` de la vista del administrador. Estos serán necesarios para que funcione correctamente la interacción de los botones y la reconstrucción de la lista de preguntas cuando haya un update.

vistaAdministrador: tomar input de respuestas y llamar al controlador

Para agregar una pregunta se tiene que poder recorrer los elementos del html pertenecientes a las `respuestas`, tomar sus datos y llamar luego al controlador para que cree las preguntas junto a sus respuestas. En la función `agregarPregunta` tendrás que `pushear` al arreglo de `respuestas` cada `respuesta` existente. Recordá como estaba formado el elemento respuesta en `Cómo están representadas las preguntas y respuestas`. Este arreglo de `respuestas` será el que le pases al `controlador`.

> Tip: `respuesta = $(this).val();` contiene el texto de la respuesta. La cantidad de votos deberá ser seteada en 0.

modelo: asignando un id a las nuevas preguntas

Llegó la hora de completar el modelo. Lo único que tendremos que hacer acá es implementar la funcionalidad para obtener un nuevo `id`. con `obtenerUltimoId`. Lo que debe hacer esta función es buscar el id más alto y asignar ese id a la nueva pregunta. Para hacerlo vas a tener que recorrer la lista de preguntas del modelo.

> Tip: cuidado con la primer pregunta que se agrega que no tendrá ningún `id` con el cual compararse. Por lo que deberá tener un valor por defecto.

Si probás hasta acá la funcionalidad debería ejecutarse correctamente el agregado de preguntas con sus respuestas desde la vista del administrador. Si todavía no funciona volvé a chequear que todos los pasos los hayas realizado correctamente. ¡No arrastres un error que se puede volver una bola de nieve!

Paso 2: borrar preguntas

Llegó la hora de implementar el borrado de preguntas. Para eso nuevamente vamos a recorrer vista, controlador y modelo (este va a ser el workflow típico al agregar nuevas funcionalidades. Siempre estarán interconectadas).

vistaAdministrador

Lo unico que tendremos que hacer acá es registrar el botón para borrar la pregunta. Es decir, implementarlo para que cuando se haga click en el botón se llame al método correspondiente del `controlador`. Para eso deberá saber que pregunta debe ser borrada mediante su `id` para poder pasarlo al controlador como parámetro. Esto podrá hacerse mediante la siguiente línea: `var id = parseInt($('.list-group-item.active').attr('id'));` que detecta que elemento está seleccionado.

controlador

Al recibir la señal de la vista para borrar una pregunta éste debe llamar al método correspondiente del `modelo`.

modelo: borrar pregunta y notificar

Por último, el modelo deberá borrar efectivamente la pregunta según el `id`pasado por el controlador. Para eso la tiene que borrar de su arreglo de preguntas y `notificar` que un cambio fue hecho.

> Tip: Para notificar tendrás que previamente haber asignado un `evento` al borrado de preguntas. Algo de esta forma: `this.preguntaEliminada = new Evento(this);` y la vista deberá estas suscripta a este evento para reconstruir la lista. this.modelo.preguntaEliminada.suscribir(function() { contexto.reconstruirLista(); });