Implementación de un juego de votación en Classpip

1 Presentación

Este tutorial muestra cómo implementar paso a paso un juego de votación. En este juego, cada alumno puede emitir una votación en la que elige a N compañeros, de manera que al primero se le asignan p1 puntos, al segundo p2 puntos, etc. Los valores de N, p1, p2, etc., serán establecidos por el profesor en el momento de la creación del juego. Los resultados de las votaciones de cada alumno se irán mostrando en tiempo real en el panel del Dashboard correspondiente al juego.

A este juego le llamaremos: "Juego de Votación Uno A Todos". De esta manera lo diferenciaremos de otro tipo de juegos de votación posibles, como, por ejemplo, uno en el que todos los alumnos votan a cada uno del resto de compañeros (que podríamos llamar "Juego de Votación Todos a Uno").

Para implementar este juego hay que modificar el proyecto Services (para incorporar los nuevos modelos de datos), el Dashboard (para introducir los mecanismos de creación y seguimiento del juego por parte del profesor), el Mobil-student (para introducir el mecanismo que permitirá a cada alumno emitir su votación) y el Servidor (para implementar el mecanismo de notificación de las votaciones en tiempo real).

La implementación que se describe aquí corresponde a la modalidad de juego individual. De forma similar podría implementarse una modalidad de juego en equipo, aunque en este caso habría que definir primero si los que votas a los equipos participantes son los individuos o los equipos.

El orden en el que se describen los pasos es el que facilita una explicación clara. Ese orden no necesariamente se corresponde al orden en el que se harían las cosas. Por ejemplo, a veces se habla de las clases que hay que implementar después de mostrar las funciones que usan esas clases. Normalmente se implementarían primero las clases y luego las funciones que las usan.

2. Nuevos modelos de datos

Para implementar el juego se necesitan dos nuevos modelos de datos, que se muestran en la figura 1.

El modelo JuegoDeVotacionUnoATodos contiene la información del juego. Además de los atributos habituales de cada juego (id, NombreJuego, Tipo, Modo, JuegoActivo, JuegoTerminado) contiene un vector que indica cuántos puntos se deben asignar a cada uno de los alumnos a los que puede votarse. El número de elementos de ese vector es N (de acuerdo con la descripción del juego en el apartado anterior), el primer elemento del vector es p1, el segundo es p2, etc.

El modelo Grupo tiene una relación hasMany con JuegoDeVotacionUnoATodos. Por esta razón aparece el atributo grupoId. Además, entre Alumno y JuegoDeVotacionUnoATodos hay una relación N:M, que se implementa a través de otro modelo nuevo: AlumnoJuegoDeVotacionUnoATodos. En este modelo, además de los atributos necesarios para

implementar la relación (alumnoId y juegoDeVotacionUnoATodosId) tenemos el atributo puntosTotales que contendrá la suma de puntos recibida por el alumno como resultado del juego de votación y el atributo votos, que es un vector de N enteros que contiene los identificadores de los N alumnos a los que ha votado el alumno (el primero es el alumno al que le concedió los p1 puntos, etc.).

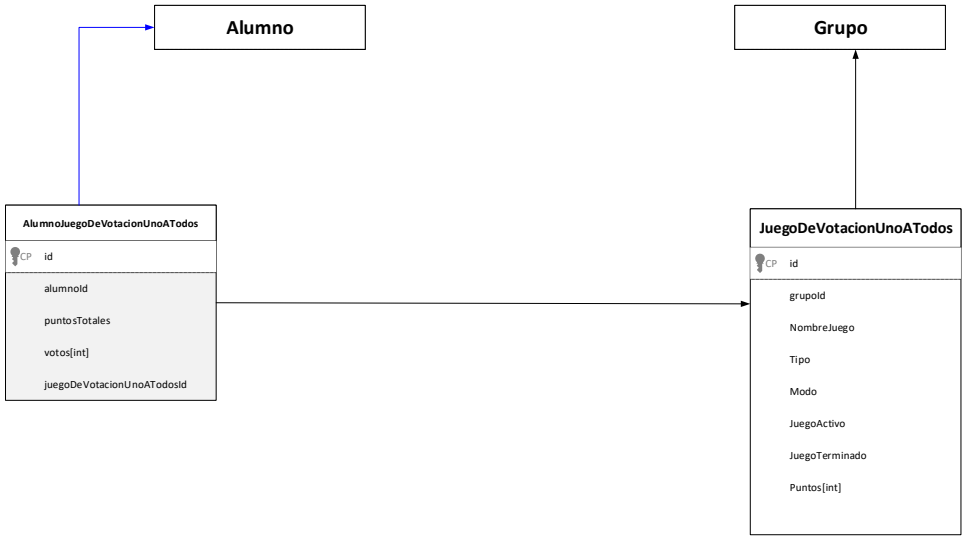

Figura 1: Los dos nuevos modelos de datos que se necesitan

El modelo JuegoDeVotacionUnoATodos se crea con loopback siguiendo los pasos descritos en la lección 13 del tutorial de herramientas de classpip, en el que se muestra cómo crear la relación hasMany entre dos modelos.

Para la creación del modelo AlumnoJuegoDeVotacionUnoATodos y las relaciones necesarias puede seguirse la lección 18. Los pasos son:

1. Crear el modelo AlumnoJuegoDeVotacionUnoATodos con los atributos puntosTotales (number) y votos (array of number).
2. Crear una relación hasMany entre Alumno y JuegoDeVotacionUnoATodos, usando como puente el modelo AlumnoJuegoDeVotacionUnoATodos e introduciendo alumnoId como clave foránea.
3. Crear una relación hasMany entre JuegoDeVotacionUnoATodos y Alumno, usando de nuevo como puente el modelo AlumnoJuegoDeVotacionUnoATodos e introduciendo juegoDeVotacionUnoATodosId como clave foránea.
4. Crear una relación belongsTo entre AlumnoJuegoDeVotacionUnoATodos y Alumno, usando como clave foránea alumnoId.
5. Crear una relación belongsTo entre AlumnoJuegoDeVotacionUnoATodos y JuegoDeVotacionUnoATodos, usando como clave foránea juegoDeVotacionUnoATodosId.

Como resultado de todos estos pasos se habrá creado los dos ficheros siguientes:

juego-de-votacion-uno-a-todos.json

```
{
  "name": "JuegoDeVotacionUnoATodos",
  "plural": "juegosDeVotacionUnoATodos",
  "base": "PersistedModel",
  "idInjection": true,
  "options": {
    "validateUpsert": true
  },
  "properties": {
    "Tipo": {
      "type": "string",
      "required": true
    },
    "NombreJuego": {
      "type": "string",
      "required": true
    },
    "Modo": {
      "type": "string",
      "required": true
    },
    "JuegoActivo": {
      "type": "boolean",
      "required": true,
      "default": false
    },
    "JuegoTerminado": {
      "type": "boolean",
      "required": true,
      "default": false
    },
    "Puntos": {
      "type": [
        "number"
      ],
      "required": true
    }
  },
  "validations": [],
  "relations": {
    "alumnos": {
      "type": "hasMany",
      "model": "Alumno",
      "foreignKey": "juegoDeVotacionUnoATodosId",
      "options": {
        "nestRemoting": true
      },
      "through": "AlumnoJuegoDeVotacionUnoATodos"
    }
  },
  "acls": [],
  "methods": {}
}
```

alumno-juego-de-votacion-uno-a-todos.json

```
{
  "name": "AlumnoJuegoDeVotacionUnoATodos",
  "plural": "alumnosJuegoDeVotacionUnoATodos",
  "base": "PersistedModel",
  "idInjection": true,
  "options": {
    "validateUpsert": true
  },
  "properties": {
    "puntosTotales": {
      "type": "number",
      "required": false,
      "default": 0
    },
    "votos": {
      "type": [
```

```
        "number"
      ]
    }
  },
  "validations": [],
  "relations": {
    "alumno": {
      "type": "belongsTo",
      "model": "Alumno",
      "foreignKey": "alumnoId",
      "options": {
        "nestRemoting": true
      }
    },
    "juegoDeVotacionUnoATodos": {
      "type": "belongsTo",
      "model": "JuegoDeVotacionUnoATodos",
      "foreignKey": "juegoDeVotacionUnoATodosId",
      "options": {
        "nestRemoting": true
      }
    }
  },
  "acls": [],
  "methods": {}
}
```

Y además, se harán incluido la siguiente información en el campo “relations” del fichero alumno.json:

```
"relations": {

      …

   "juegoDeVotacionUnoATodos": {
     "type": "hasMany",
     "model": "JuegoDeVotacionUnoATodos",
     "foreignKey": "alumnoId",
     "options": {
       "nestRemoting": true
     },
     "through": "AlumnoJuegoDeVotacionUnoATodos"
   }
 },
```

3 Modificaciones en Dashboard

3.1 Creación del juego

Lo primero que haremos es añadir al componente juego.component la lógica para la creación del juego de votación. Los pasos son los siguientes:

Añadir un nuevo tipo de juego:

```
seleccionTipoJuego: ChipColor[] = [
   {nombre: 'Juego De Puntos', color: 'primary'},
   {nombre: 'Juego De Colección', color: 'accent'},
   {nombre: 'Juego De Competición', color: 'warn'},
   {nombre: 'Juego De Avatar', color: 'primary'},
   {nombre: 'Juego De Cuestionario', color: 'accent'},
   {nombre: 'Juego De Geocaching', color: 'warn'},
   {nombre: 'Juego De Votación Uno A Todos', color: 'primary'},
 ];
```

Añadir nuevas variables para la creación del juego:

```
// información para crear juego de votación

  tipoDeVotacionSeleccionado: string;
  seleccionTipoDeVotacion: ChipColor[] = [
    {nombre: 'Uno A Todos', color: 'primary'},
    {nombre: 'Todos A Uno', color: 'warn'}
  ];
  tengoTipoDeVotacion = false;
```

Ya se ve que hemos previsto que haya otros tipos de juegos de votación.

Añadir en el stepper de creación del juego los pasos tres (elegir tipo de votación), cuatro (introducir puntuaciones) y último (crear el juego) para la creación del juego de votación:

```
<!-- //// SI ESCOGEMOS EL JUEGO DE TIPO DE VOTACION ////////////////////////////////-->
<div *ngIf="tipoDeJuegoSeleccionado === 'Juego De Votación' && tengoModo">
<!-- TERCER PASO: SELECTION DE TIPO DE VOTACION -->
            <mat-step>
              <ng-template matStepLabel><div>Votación</div></ng-template>
              <div class = "colorsChips">
                  <label class = "LabelStepper">Selecciona el tipo de votación</label>
                  <mat-chip-list class="mat-chip-list-stacked">
                    <mat-chip *
                        *ngFor="let tipo of seleccionTipoDeVotacion; let i = index"
                        selected [color]="tipo.color"
                        selected="{{seleccionTipoDeVotacion[i].selected}}" (
                        (click)="TipoDeVotacionSeleccionado(seleccionTipoDeVotacion[i])"
                        [value]="tipo.nombre"
                        [selectable]="true">
                        {{tipo.nombre}}
                    </mat-chip>
                  </mat-chip-list>
                  <span
                        style ="font-weight: bold;color:green;">
                Has elegido ... {{tipoDeJuegoSeleccionado}} ... {{tipoDeVotacionSeleccionado}}
                  </span>
              </div>
              <div class="btnCambiarStep">
                <button type="button" class="btn Back" matStepperPrevious>Atrás</button>
                <button [disabled]="!tengoTipoDeVotacion" type="button" class="btn Next"
                        matStepperNext>Siguiente</button>
              </div>
            </mat-step>
            <!--CUARTO PASO : INTRODUCIR PUNTUACIONES-->
            <mat-step>
                <ng-template matStepLabel><div>Puntuaciones</div></ng-template>
                <div class="contenedor">
                    <div class="enColumna">
                      <div class = "enFila" style="width: 80%">
                        <div style="margin-top: 15px;">
                          <span>Introduzca el número de puntos y
                                seleccione la posición a la que desea asignarlos: </span>
                          <mat-form-field>
                            <mat-label>Introduzca pos puntos deseados</mat-label>
                            <input matInput formControlName="NuevaPuntuacion"
                                placeholder="Escribe los puntos"
                                requiered (keyup) = "GuardarNuevaPuntuacion()">
                          </mat-form-field>
                        </div>
                        <div style="margin-top: 10px">
                          <button type="button"class="btn Aceptar"
                                [disabled]="!Preparado()"
                                (click) = AnadirPuntos()>
                                Establecer Puntuación
                        </button>
                        </div>
                      </div>
                      <form style="width: 100%">
                          <table style="width: 100%" align="center"
                              mat-table [dataSource]="dataSource" class="mat-elevation-z8">
                            <ng-container matColumnDef="select">
```

```
                        <th mat-header-cell
                                *matHeaderCellDef class="tituloColumnaTabla">
                          <mat-checkbox
                             (change)="$event ? MasterToggle() : null;"
                             [checked]="selection.hasValue() && IsAllSelected()"
                             [indeterminate]="selection.hasValue() && !IsAllSelected()">
                          </mat-checkbox>
                        </th>
                        <td mat-cell *matCellDef="let row">
                          <mat-checkbox
                              (click)="$event.stopPropagation()"
                              (change)="$event ? selection.toggle(row) : null;"
                              [checked]="selection.isSelected(row)">
                          </mat-checkbox>
                        </td>
                      </ng-container>
                      <!-- Posicion Column -->
                      <ng-container matColumnDef="Posicion">
                        <th mat-header-cell
                                *matHeaderCellDef class="tituloColumnaTabla">
                                Posición </th>
                        <td mat-cell style= "text-align: center"
                                *matCellDef="let tablaPuntuacion">
                                {{tablaPuntuacion.Posicion}} </td>
                      </ng-container>

                      <!-- Puntos Column -->
                      <ng-container matColumnDef="Puntos">
                        <th mat-header-cell
                                *matHeaderCellDef
                                class="tituloColumnaTabla">
                                Puntos </th>
                        <td mat-cell
                                *matCellDef="let tablaPuntuacion; ">
                                {{tablaPuntuacion.Puntuacion}} </td>
                      </ng-container>
                      <tr mat-header-row
                                *matHeaderRowDef="displayedColumnsTablaPuntuacion"></tr>
                      <tr mat-row
                                *matRowDef="let row;
                                        columns: displayedColumnsTablaPuntuacion;"></tr>
                    </table>
                </form>
                <div style="margin-top: 10px">
                      <button type="button"class="btn Agregar" (click) = AnadirFila()>
                          Añadir Fila
                          <i class="material-icons">add</i></button>
                      <button type="button"class="btn Eliminar" (click) = EliminarFila()>
                          Eliminar Fila<i class="material-icons">delete</i></button>
                </div>

              </div>
          </div>
          <div class="btnCambiarStep">
                  <button type="button" class="btn Back" matStepperPrevious>Atrás</button>
                  <button type="button" class="btn Next" matStepperNext>Siguiente</button>

          </div>
      </mat-step>
      <!-- ULTIMO PASO: CREAR EL JUEGO DE VOTACION -->
      <mat-step>
        <ng-template matStepLabel><div>Finalizar</div></ng-template>

        <div class="btnCambiarStep">
          <button type="button" class="btn Back" matStepperPrevious>Atrás</button>
          <button style= "width: 60%" type="button" class="btn Aceptar"
                  (click) = "CrearJuegoDeVotacionUnoATodos()">
                  Crear juego de votación Uno A Todos</button>
        </div>
      </mat-step>
      </div>
```

Para le paso cuatro (establecer puntuaciones) usamos exactamente la misma plantilla que para establecer las puntuaciones de las jornadas de una competición de tipo formula uno. Al final hay que hacer lo mismo: indicar cuántos puntos para el primero, cuántos para el segundo, etc. La plantilla genera la vista de la figura 2:

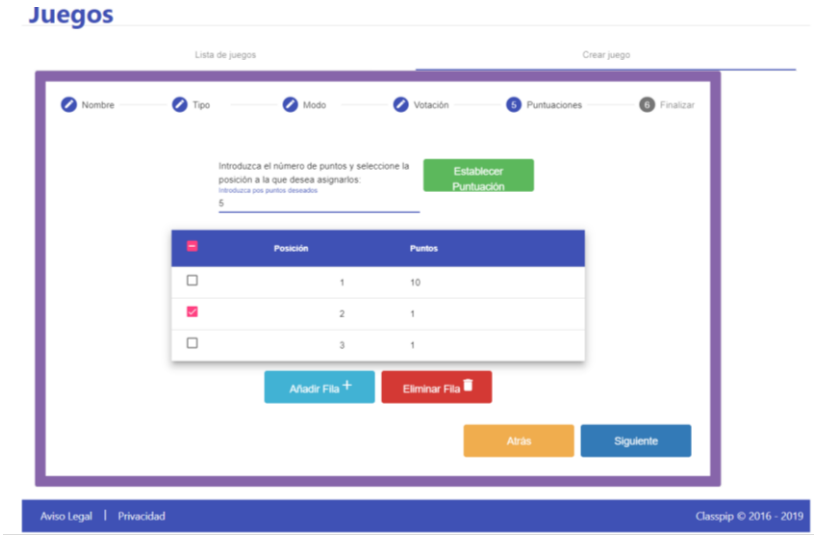

Figura 2: Vista de la página en la que se establecen las puntuaciones para la votación

Pueden añadirse o quitarse filas. En el ejemplo de la figura hemos añadido dos filas (hay una fila por defecto con 10 puntos para el primero). Estamos a punto de establecer que para el segundo se asignarán 5 puntos.

Ahora tenemos que añadir las funciones que se activan desde la plantilla html:

```
 TipoDeVotacionSeleccionado(tipoVotacion: ChipColor) {
  this.tipoDeVotacionSeleccionado = tipoVotacion.nombre;
  this.tengoTipoDeVotacion = true;
}

CrearJuegoDeVotacionUnoATodos() {
  const juegoDeVotacion = new JuegoDeVotacionUnoATodos (
    this.tipoDeJuegoSeleccionado + ' ' + this.tipoDeVotacionSeleccionado ,
    this.modoDeJuegoSeleccionado,
    true,
    this.Puntuacion,
    this.nombreDelJuego,
    false,
    this.grupo.id);
  this.peticionesAPI.CreaJuegoDeVotacionUnoATodos (juegoDeVotacion, this.grupo.id)
  .subscribe (juegoCreado => {
    this.juego = juegoCreado;
    this.sesion.TomaJuego(this.juego);
    this.juegoCreado = true;

    if (this.modoDeJuegoSeleccionado === 'Individual') {

      // tslint:disable-next-line:prefer-for-of
      for (let i = 0; i < this.alumnosGrupo.length; i++) {
        // tslint:disable-next-line:max-line-length
        this.peticionesAPI.InscribeAlumnoJuegoDeVotacionUnoATodos(
              new AlumnoJuegoDeVotacionUnoATodos(this.alumnosGrupo[i].id, this.juego.id))
        .subscribe();
      }
    }
```

```
      Swal.fire('Juego de votación tipo Uno A Todos creado correctamente', ' ', 'success');

      // El juego se ha creado como activo. Lo añadimos a la lista correspondiente
      if (this.juegosActivos === undefined) {
      // Si la lista aun no se ha creado no podre hacer el push
              this.juegosActivos = [];
      }
      this.juegosActivos.push (this.juego);
      // Al darle al botón de finalizar limpiamos el formulario y reseteamos el stepper
      this.Limpiar();
      // Regresamos a la lista de equipos (mat-tab con índice 0)
      this.tabGroup.selectedIndex = 0;

    });
}
```

Todas las variables y funciones a las que se hace referencia en el paso 4 (por ejemplo, NuevaPuntuacion, TablaPuntuacion o EliminarFila()) son las mismas que se usan para el juego de competición tipo formula uno, por lo que no hay que volver a declararlas.

El siguiente paso es crear las clases para representar el juego y las inscripciones de los alumnos (básicamente lo mismo que se indicó al crear los modelos de datos):

```
export class JuegoDeVotacionUnoATodos {
  id: number;
  Tipo: string;
  grupoId: number;
  NombreJuego: string;
  Modo: string;
  JuegoActivo: boolean;
  JuegoTerminado: boolean;
  Puntos: number[];
  constructor(Tipo?: string, Modo?: string, JuegoActivo?: boolean,
              Puntos?: number[], NombreJuego?: string,
              JuegoTerminado?: boolean, grupoId?: number) {

    this.Tipo = Tipo;
    this.Modo = Modo;
    this.JuegoActivo = JuegoActivo;
    this.Puntos = Puntos;
    this.NombreJuego = NombreJuego;
    this.JuegoTerminado = JuegoTerminado;
    this.grupoId = grupoId;
  }
}
```

```
export class AlumnoJuegoDeVotacionUnoATodos {

  puntosTotales: number;
  id: number;
  alumnoId: number;
  juegoDeVotacionUnoATodosId: number;
  Votos: number[];

  constructor(alumnoId?: number, juegoDeVotacionUnoATodosId?: number) {
    this.alumnoId = alumnoId;
    this.juegoDeVotacionUnoATodosId = juegoDeVotacionUnoATodosId;
    this.puntosTotales = 0;
  }
}
```

Ahora incorporamos al servicio peticionesAPI las funciones necesarias para registrar el juego y las inscripciones de los alumnos (que se usan en la función CrearJuegoDeVotacionUnoATodos):

```
public CreaJuegoDeVotacionUnoATodos(juego: JuegoDeVotacionUnoATodos, grupoId: number):
                        Observable<JuegoDeVotacionUnoATodos> {
    return this.http.post<JuegoDeVotacionUnoATodos>(
                        this.APIUrlGrupos + '/' + grupoId + '/juegoDeVotacionUnoATodos', juego);
}
```

```
public InscribeAlumnoJuegoDeVotacionUnoATodos(
                alumnoJuegoDeVotacionUnoATodos: AlumnoJuegoDeVotacionUnoATodos) {
    return this.http.post<AlumnoJuegoDeVotacionUnoATodos>(
                        this.APIUrlAlumnoJuegoDeVotacionUnoATodos,
                        alumnoJuegoDeVotacionUnoATodos);
}
```

En la segunda función se hace referencia a una nueva URL que hay que definir (definimos otra que aún no necesitamos pero que necesitaremos pronto):

```
 private APIUrlJuegoDeVotacionUnoATodos = this.host + ':3000/api/JuegosDeVotacionUnoATodos';
 private APIUrlAlumnoJuegoDeVotacionUnoATodos = this.host + ':3000/api/alumnosJuegoDeVotacionUno
ATodos';
```

Incorporarmos a la función DameListaJuegos del servicio cálculos el código necesario para obtener los juegos de votación del grupo.

```
// antes hemos obtenido todos los otros tipos de juegos
this.peticionesAPI.DameJuegosDeVotacionUnoATodos(grupoID)
.subscribe(juegosVotacionUnoATodos => {
// tslint:disable-next-line:prefer-for-of
for (let i = 0; i < juegosVotacionUnoATodos.length; i++) {
        if (juegosVotacionUnoATodos[i].JuegoActivo === true) {
                        juegosVotacionUnoATodos[i].Tipo = 'Juego De Votación Uno A Todos';
                        juegosActivos.push(juegosVotacionUnoATodos[i]);
        } else {
                        juegosVotacionUnoATodos[i].Tipo = 'Juego De Votación Uno A Todos';
                        juegosInactivos.push(juegosVotacionUnoATodos[i]);
                }
}
// esto que viene ahora ya estaba en la función.
const resultado = {
        activos: juegosActivos,
        inactivos: juegosInactivos,
        preparados: juegosPreparados};
obs.next (resultado);
```

Este bloque de código hay que incorporarlo en la cadena de peticiones para recuperar todos los juegos del grupo. Observar que se reescribe el campo Tipo del juego. Originalmente, ese campo tiene el valor “Juego De Votación”. Lo ampliamos en ese momento a “Juego De Votación Uno A Todos” en previsión de que en el futuro haya otros tipos de juegos de votación.

Añadir a peticionesAPI la nueva función que necesitamos:

```
public DameJuegosDeVotacionUnoATodos(grupoId: number): Observable<JuegoDeVotacionUnoATodos[]> {
    return this.http.get<JuegoDeVotacionUnoATodos[]>(
                this.APIUrlGrupos + '/' + grupoId + '/juegoDeVotacionUnoATodos');
}
```

Este es un buen punto para probar lo que se ha hecho hasta el momento. Si todo ha ido bien, ahora el profesor puede crear un juego de votación y registrarlo. Debería aparecer el juego y las inscripciones de los alumnos en el fichero db.json del proyecto Services y debería aparecer el nuevo juego entre los juegos activos del grupo.

3.2 Supervisión del juego

Para la supervisión del juego necesitamos un componente que nos muestre el listado de alumnos ordenados según los puntos que tengan. Además, este listado se va a ir actualizando en tiempo real a medida que los alumnos vayan emitiendo sus votos.

Primero creamos en la carpeta juego-seleccionado-activo un componente nuevo que se llamará JuegoDeVotacionUnoATodosSeleccionadoActivo. La plantilla html para este componente puede ser muy parecida a la del juego de competición formula uno, en la que se muestra la lista de alumnos ordenada según los puntos que tienen en la competición.

Hay que copiar la plantilla html del juego de competición formula uno y adaptarla. Esa plantilla puede quedar asi:

```
<div class="enColumna">
  <div class="enColumna" style="width: 65%">
    <button type="button" class="btn Eliminar" (click) = "DesactivarJuego()"> Desactivar
                <i class="material-icons">history</i></button>
  </div>
  <br>
  <div  *ngIf="VotacionFinalizada()" class="subtitulo">Clasificación general definitiva</div>
  <div  *ngIf="!VotacionFinalizada()" class="subtitulo">Clasificación general</div>
  <br>

  <!-----CLASIFICACIÓN INDIVIDUAL ////////////////////////////////////////////////////-->
  <form style="width: 100%"  *ngIf="juegoSeleccionado.Modo === 'Individual'">
    <div class="filter"  style="width: 60%">
      <mat-form-field style="width: 95% !Important;">
        <input matInput (keyup)="applyFilter($event.target.value)"
                placeholder="Filtro para buscar alumno...">
        <i class="material-icons" matSuffix>search</i>
      </mat-form-field>
    </div>

    <table style="width: 80%"  align="center"  mat-table
                [dataSource]="datasourceAlumno" class="example-container mat-elevation-z8">
      <ng-container matColumnDef="posicion">
        <th mat-header-cell
              *matHeaderCellDef class="tituloColumnaTabla"
              style="width: 20%"  align="center">
              Posición Global</th>
        <td mat-cell *matCellDef="let alumno; let i = index">
          <div *ngIf = "(i == 0) && (VotacionFinalizada())"
                      style="color: green; font-size: xx-large;">
              {{i+1}} </div>
          <div *ngIf = "(i != 0) || (!VotacionFinalizada())">{{i+1}}</div>
        </td>
      </ng-container>
      <ng-container matColumnDef="nombreAlumno">
        <th mat-header-cell
              *matHeaderCellDef class="tituloColumnaTabla"
              style="width: 20%"  align="center">
              Nombre</th>
        <td mat-cell *matCellDef="let alumno; let i = index">
          <div *ngIf = "(i == 0) && (VotacionFinalizada())"
              style="color: green; font-size: xx-large;">
              {{alumno.nombre}} </div>
          <div *ngIf = "(i != 0) || (!VotacionFinalizada())">{{alumno.nombre}}</div>
        </td>
      </ng-container>
```

```
        <ng-container matColumnDef="primerApellido">
            <th mat-header-cell
                *matHeaderCellDef class="tituloColumnaTabla"
                style="width: 20%"  align="center">
                Primer Apellido</th>
            <td mat-cell *matCellDef="let alumno; let i = index">
              <div *ngIf = "(i == 0) && (VotacionFinalizada())"
                style="color: green; font-size: xx-large;">
                {{alumno.primerApellido}} </div>
              <div *ngIf = "(i != 0) || (!VotacionFinalizada())">{{alumno.primerApellido}}</div>
            </td>
        </ng-container>

        <ng-container matColumnDef="segundoApellido">
            <th mat-header-cell
                *matHeaderCellDef class="tituloColumnaTabla"
                style="width: 20%"  align="center">
                Segundo Apellido</th>
            <td mat-cell *matCellDef="let alumno; let i = index">
              <div *ngIf = "(i == 0) && (VotacionFinalizada())"
                style="color: green; font-size: xx-large;">
                {{alumno.segundoApellido}} </div>
              <div *ngIf = "(i != 0) ||(!VotacionFinalizada())">{{alumno.segundoApellido}}</div>
            </td>
        </ng-container>

        <ng-container matColumnDef="puntos">
            <th mat-header-cell
                *matHeaderCellDef class="tituloColumnaTabla"
                style="width: 20%"  align="center">
                Puntos</th>
            <td mat-cell *matCellDef="let alumno; let i = index">
              <div *ngIf = "(i == 0) && (VotacionFinalizada())"
                style="color: green; font-size: xx-large;">
                {{alumno.puntos}} </div>
              <div *ngIf = "(i != 0) || (!VotacionFinalizada())">{{alumno.puntos}}</div>
            </td>
        </ng-container>

        <ng-container matColumnDef="incremento">
          <th mat-header-cell
                *matHeaderCellDef class="tituloColumnaTabla"
                style="width: 20%"  align="center"></th>
          <td mat-cell *matCellDef="let alumno; ">
            <div *ngIf = "alumno.incremento !==0 " class="blink">+{{alumno.incremento}}</div>
          </td>
        </ng-container>

        <ng-container matColumnDef=" ">
          <th mat-header-cell *matHeaderCellDef class="tituloColumnaTabla"></th>
          <td mat-cell style= "text-align: right" *matCellDef="let alumno">
            <p *ngIf = "alumno.votado" style = "color: green;">
              <i class="material-icons green md-36">done</i> </p>
          </td>
        </ng-container>
        <tr mat-header-row *matHeaderRowDef="displayedColumnsAlumnos; sticky: true"></tr>
        <tr mat-row *matRowDef="let row; columns: displayedColumnsAlumnos;"></tr>
    </table>
  </form>
</div>
```

Que nos presenta la siguiente vista mostrada en la figura 3:

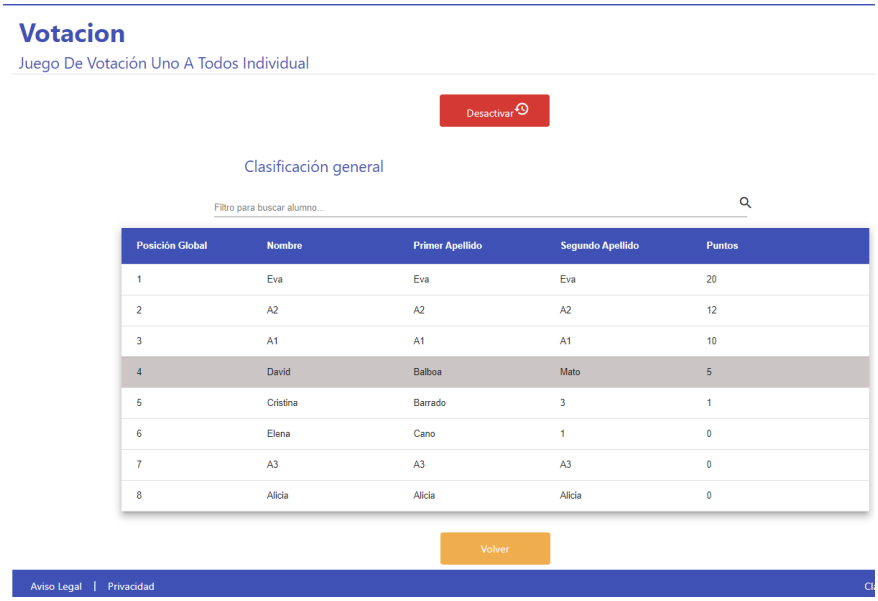

Figura 3: Panel de seguimiento de la votación

Básicamente, es una tabla que muestra los datos de los participantes (nombre y puntos). Hemos previsto dos columnas adicionales (una que se llama “incremento” y otra “ “) que usaremos después para mostrar en tiempo real qué jugador ha recibido los puntos de cada votación y qué jugadores han votado ya.

Las funciones que controlan esta vista son muy parecidas a las del juego de competición formula uno. Son las siguientes:

```
ngOnInit() {
  this.juegoSeleccionado = this.sesion.DameJuego();
  if (this.juegoSeleccionado.Modo === 'Individual') {
    this.AlumnosDelJuego();
  } else {
    console.log ('aun no funciona la modalidad por equipos');
  }
}
  // Recupera los alumnos que pertenecen al juego
AlumnosDelJuego() {
    this.peticionesAPI.DameAlumnosJuegoDeVotacionUnoATodos(this.juegoSeleccionado.id)
    .subscribe(alumnosJuego => {
      console.log ('Ya tengo los alumnos');
      console.log(alumnosJuego);
      this.alumnosDelJuego = alumnosJuego;
      this.RecuperarInscripcionesAlumnoJuego();
    });
}

RecuperarInscripcionesAlumnoJuego() {
  this.peticionesAPI.DameInscripcionesAlumnoJuegoDeVotacionUnoATodos(
              this.juegoSeleccionado.id)
  .subscribe(inscripciones => {
    this.listaAlumnosOrdenadaPorPuntos = inscripciones;
    // ordena la lista por puntos
    // tslint:disable-next-line:only-arrow-functions
    this.listaAlumnosOrdenadaPorPuntos =
              this.listaAlumnosOrdenadaPorPuntos.sort(function(obj1, obj2) {
      return obj2.puntosTotales - obj1.puntosTotales;
    });
    this.TablaClasificacionTotal();
  });
}
```

```
TablaClasificacionTotal() {
  if (this.juegoSeleccionado.Modo === 'Individual') {
    // tslint:disable-next-line:max-line-length
    this.rankingIndividualJuegoDeVotacionUnoATodos =
              this.calculos.PrepararTablaRankingIndividualVotacionUnoATodos (
      this.listaAlumnosOrdenadaPorPuntos,
      this.alumnosDelJuego,
      this.juegoSeleccionado.Puntos);
    // tslint:disable-next-line:only-arrow-functions
    this.rankingIndividualJuegoDeVotacionUnoATodos =
              this.rankingIndividualJuegoDeVotacionUnoATodos.sort(function(obj1, obj2) {
      return obj2.puntos - obj1.puntos;
    });
    this.datasourceAlumno =
              new MatTableDataSource(this.rankingIndividualJuegoDeVotacionUnoATodos);

  } else {
    console.log ('la modalidad en equipo aun no está operativa');

  }
}

VotacionFinalizada() {
  // Miro si todos han votado
  let cont = 0;
  this.rankingIndividualJuegoDeVotacionUnoATodos.forEach (al => {if (al.votado) { cont++; }});
  return (cont === this.rankingIndividualJuegoDeVotacionUnoATodos.length);
}

DesactivarJuego() {
  Swal.fire({
    title: '¿Seguro que quieres desactivar el juego de votación?',
    icon: 'warning',
    showCancelButton: true,
    confirmButtonColor: '#3085d6',
    cancelButtonColor: '#d33',
    confirmButtonText: 'Si, estoy seguro'
  }).then((result) => {
    if (result.value) {
      // Primero registro las puntuaciones definitivas de cada alumno
      this.listaAlumnosOrdenadaPorPuntos.forEach (alumno => {
        alumno.puntosTotales = this.rankingIndividualJuegoDeVotacionUnoATodos.filter (
              al => al.id === alumno.alumnoId)[0].puntos;
        this.peticionesAPI.ModificaInscripcionAlumnoJuegoDeVotacionUnoATodos (alumno)
        .subscribe();
      });

      this.juegoSeleccionado.JuegoActivo = false;
      this.peticionesAPI.CambiaEstadoJuegoDeVotacionUnaATodos (this.juegoSeleccionado)
      .subscribe(res => {
          if (res !== undefined) {
            console.log(res);
            console.log('juego desactivado');
            Swal.fire('El juego se ha desactivado correctamente');
            this.location.back();
          }
      });
    }
  });
}

applyFilter(filterValue: string) {
  this.datasourceAlumno.filter = filterValue.trim().toLowerCase();
}
```

Al cargar el componente, traemos los alumnos del juego y sus inscripciones. Con esa información preparamos la tabla que usaremos para mostrar el ranking. Para ello, necesitamos añadir un par de funciones más a peticionesAPI:

```
public DameAlumnosJuegoDeVotacionUnoATodos(juegoId: number): Observable<Alumno[]> {
    return this.http.get<Alumno[]>(
        this.APIUrlJuegoDeVotacionUnoATodos + '/' + juegoId + '/alumnos');
}

 // tslint:disable-next-line:max-line-length
public DameInscripcionesAlumnoJuegoDeVotacionUnoATodos(juegoId: number):
        Observable<AlumnoJuegoDeVotacionUnoATodos[]> {
    return this.http.get<AlumnoJuegoDeVotacionUnoATodos[]>(
        this.APIUrlAlumnoJuegoDeVotacionUnoATodos +
        '?filter[where][juegoDeVotacionUnoATodosId]=' + juegoId);
  }
```

Para preparar la tabla que necesitamos para mostrar el ranking necesitamos una clase nueva con la información de cada alumno para esa tabla. Además de nombre y puntos recibidos, esa información incluye los campos votado (para registrar si el alumno ha emitido ya su votación) e incremento (que contendrá los puntos recibidos por el alumno en la última votación que le ha afectado).

```
export class TablaAlumnoJuegoDeVotacionUnoATodos {

  posicion: number;
  nombre: string;
  primerApellido: string;
  segundoApellido: string;
  puntos: number;
  id: number;
  votado: boolean;
  incremento: number;

  constructor(posicion?: number, nombre?: string, primerApellido?: string,
              segundoApellido?: string, puntos?: number, id?: number) {

    this.posicion = posicion;
    this.nombre = nombre;
    this.primerApellido = primerApellido;
    this.segundoApellido = segundoApellido;
    this.puntos = puntos;
    this.id = id;
    this.votado = false;
    this.incremento = 0;
  }
}
```

En el servicio de cálculos incluimos la función que se necesita para preparar el ranking:

```
public PrepararTablaRankingIndividualVotacionUnoATodos(
                listaAlumnosOrdenadaPorPuntos: AlumnoJuegoDeVotacionUnoATodos[],
                alumnosDelJuego: Alumno[], puntos: number[]):
                        TablaAlumnoJuegoDeVotacionUnoATodos[] {
    const rankingJuegoDeVotacion: TablaAlumnoJuegoDeVotacionUnoATodos [] = [];
    // tslint:disable-next-line:prefer-for-oF
    for (let i = 0; i < listaAlumnosOrdenadaPorPuntos.length; i++) {
      let alumno: Alumno;
      const alumnoId = listaAlumnosOrdenadaPorPuntos[i].alumnoId;
      alumno = alumnosDelJuego.filter(res => res.id === alumnoId)[0];
      // tslint:disable-next-line:max-line-length

      const elem = new TablaAlumnoJuegoDeVotacionUnoATodos(i + 1, alumno.Nombre,
                        alumno.PrimerApellido, alumno.SegundoApellido,
                        listaAlumnosOrdenadaPorPuntos[i].puntosTotales, alumnoId);
```

```
      rankingJuegoDeVotacion[i] = elem;
    }

    // Ahora voy a ver qué alumnos ya han votado para acumular sus votos y marcarlos
    // como que ya han votado
    // tslint:disable-next-line:prefer-for-of
    for (let i = 0; i < listaAlumnosOrdenadaPorPuntos.length; i++) {
      if (listaAlumnosOrdenadaPorPuntos[i].Votos) {
        // Este alumno ya ha votado
        const alumno = listaAlumnosOrdenadaPorPuntos[i];
        // Asigno los puntos a los destinatorios
        for (let j = 0; j < puntos.length; j++) {
          const votado = rankingJuegoDeVotacion.filter (al => al.id === alumno.Votos[j])[0];
          votado.puntos = votado.puntos + puntos[j];
        }
        // Marque que el alumno ya ha votado
        rankingJuegoDeVotacion.filter (al => al.id === alumno.alumnoId)[0].votado = true;
      }
    }
    return rankingJuegoDeVotacion;
}
```

Primero creamos la lista con la información de cada alumno, uniendo el nombre con los puntos y el identificador del alumno. Luego, por cada alumno que ya ha votado (tiene información en el campo Votos), acumulamos los puntos correspondientes a su votación a los alumnos a los que ha votado.

Finalmente, la función DesactivarJuego registra en la base de datos los puntos totales recibidos por cada alumno y cambia el estado del juego. Para ello, tenemos que añadir a peticionesAPI dos funciones nuevas:

```
// tslint:disable-next-line:max-line-length
 public ModificaInscripcionAlumnoJuegoDeVotacionUnoATodos(
                inscripcion: AlumnoJuegoDeVotacionUnoATodos):
                        Observable<AlumnoJuegoDeVotacionUnoATodos> {
    return this.http.put<AlumnoJuegoDeVotacionUnoATodos>(
                this.APIUrlAlumnoJuegoDeVotacionUnoATodos + '/' + inscripcion.id, inscripcion);
  }
```

```
public CambiaEstadoJuegoDeVotacionUnaATodos( juego: JuegoDeVotacionUnoATodos):
                        Observable<JuegoDeVotacionUnoATodos> {
  // tslint:disable-next-line:max-line-length
    return this.http.put<JuegoDeVotacionUnoATodos>(this.APIUrlGrupos + '/' + juego.grupoId +
                '/juegoDeVotacionUnoATodos/' + juego.id, juego);
  }
```

Ahora hay que Incorporar la lógica para activar el componente JuegoDeVotacionUnoATodosSeleccionadoActivo. Para ello se añaden las líneas siguientes en juego-seleccionado-activo.component.html:

```
<div class="contenedor" *ngIf= "juegoSeleccionado.Tipo === 'Juego De Votación Uno A Todos'">
  <app-juego-de-votacion-uno-atodos-seleccionado-activo></app-juego-de-votacion-uno-atodos-
seleccionado-activo>
</div>
```

Ahora puede ser un buen momento para comprobar el correcto funcionamiento, aunque solo vamos a ver que al clicar sobre el juego recién creado se muestra el ranking con todos los alumnos a 0 puntos y que se clicamos en desactivar el juego pasa a la lista de juegos inactivos.

4 Aplicación móvil del estudiante

El alumno debe poder abrir el juego de votación y emitir su votación si no lo ha hecho ya. Si ya ha votado, debe poder ver cuál fue su votación.

Primero tenemos que copiar las clases JuegoDeVotacionUnoATodos y Alumno JuegoDeVotacionUnoATodos tal y como las tenemos en el Dashboard.

El componente inici carga todos los juegos del alumno llamando a la función DameJuegosAlumno, que está en el servicio CalculosService. Tenemos que modificar esta función para que traiga también los juegos de votación. Hay que añadir a esa función el siguiente trozo de código, justo antes de traer los equipos a los que pertenece el alumno:

```
console.log('ya tengo los juegos de avatar');
console.log('voy a por los juegos de votacion uno a todos');
this.peticionesAPI.DameJuegosDeVotacionUnoATodosAlumno(AlumnoId)
// tslint:disable-next-line:no-shadowed-variable
.subscribe( lista => {
       for (let i = 0; i < (lista.length); i++) {
       if (lista[i].JuegoActivo === true) {
              JuegosActivos.push(lista[i]);
       } else {
              JuegosInactivos.push(lista[i]);
       }
}
console.log('ya tengo los juegos de votacion uno a todos');
```

Hay que añadir una nueva función en peticionesAPI:

```
public DameJuegosDeVotacionUnoATodosAlumno(alumnoId: number):
              Observable<JuegoDeVotacionUnoATodos[]> {
   return this.http.get<JuegoDeVotacionUnoATodos[]>(this.APIUrlAlumnos + '/' + alumnoId +
              '/juegoDeVotacionUnoATodos');
 }
```

Ahora en inici.page.html añadimos la línea siguiente, para que muestre una imagen representativa para el juego de votación.

```
 <img *ngIf="juego.Tipo == 'Juego De Votación Uno A Todos'"
              class = "card-img" src="../../assets/img/votaciones.jpg">
```

En la función JuegoSeleccionado de inici.page.ts hay que añadir la línea siguiente, para que al clicar sobre el juego se abra la página en la que el alumno podrá hacer su votación.

```
} else if (juego.Tipo === 'Juego De Votación Uno A Todos') {
     this.navCtrl.navigateForward('/juego-votacion-uno-atodos');
}
```

Ahora hay que crear la página nueva, llamada JuegoVotacionUnoATodos.

Al cargar esa página se ejecutará esta función:

```
ngOnInit() {
   this.juegoSeleccionado = this.sesion.DameJuego();
   this.alumno = this.sesion.DameAlumno();
   if (this.juegoSeleccionado.Modo === 'Individual') {
     this.peticionesAPI.DameInscripcionAlumnoJuegoDeVotacionUnoATodos(
                        this.juegoSeleccionado.id, this.alumno.id)
     .subscribe (inscripcion => {
       // Traigo la inscripción del alumno
       this.inscripcionAlumnoJuegoDeVotacionUnoATodos = inscripcion[0];
       console.log (this.inscripcionAlumnoJuegoDeVotacionUnoATodos);
       // traigo los alumnos del juego
       this.peticionesAPI.DameAlumnosJuegoDeVotacionUnoATodos (this.juegoSeleccionado.id)
       .subscribe (alumnos => {
         if (!this.inscripcionAlumnoJuegoDeVotacionUnoATodos.Votos) {
           // aun no ha votado
           this.alumnos = alumnos;
         } else {
           // Si ha votado preparlo la lista solo con los alumnos a los que ha votado
           // para mostrar el resultado de su votación
           this.alumnosVotados = [];
           // tslint:disable-next-line:prefer-for-of
           for (let i =0; i<this.inscripcionAlumnoJuegoDeVotacionUnoATodos.Votos.length; i++) {
             const alumno = alumnos.filter
               (al => al.id === this.inscripcionAlumnoJuegoDeVotacionUnoATodos.Votos[i])[0];
             this.alumnosVotados.push (alumno);
           }
         }
         console.log (this.alumnos);
       });
     });
    } else {
      // De momento no hay avatar de equipo
    }
 }
```

Nos traemos los datos del alumno y de su inscripción al juego. También traemos los datos de todos los alumnos del juego. Luego vemos si el alumno ha votado ya o no. Si no ha votado entonces le mostraremos la lista con todos los alumnos. En el caso de que ya haya votado le mostraremos la lista de los alumnos a los que votó.

La plantilla html de esta página es la siguiente:

```
<ion-header class="juegos-header">
  <ion-toolbar class="header-toolbar">
    <ion-buttons slot="start">
      <ion-back-button class="menu-btn"></ion-back-button>
    </ion-buttons>
    <ion-title>Votación Uno A Todos</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <div *ngIf = 'inscripcionAlumnoJuegoDeVotacionUnoATodos &&
                        ¡inscripcionAlumnoJuegoDeVotacionUnoATodos.Votos'>
    <div style="text-align:center; margin-top: 10%;">
      <ion-button class="ranking-btn" (click)="Enviar()"> Envía tu votación </ion-button>
    </div>
    <ion-list *ngIf = 'alumnos'>
      <ion-reorder-group (ionItemReorder)="reorderItems($event)" disabled="false">

        <ion-item *ngFor="let alumno of alumnos; let i=index">
          <ion-item>
          <ion-label>
            {{alumno.Nombre}} {{alumno.PrimerApellido}} {{alumno.SegundoApellido}}
          </ion-label>
          </ion-item>
          <ion-label *ngIf = 'i < juegoSeleccionado.Puntos.length'
                        style="font-size: large; color: red; text-align: right;">
          {{juegoSeleccionado.Puntos[i]}}
```

```
          </ion-label>
          <ion-reorder slot="end"></ion-reorder>
        </ion-item>
      </ion-reorder-group>
    </ion-list>
  </div>
  <div *ngIf = 'inscripcionAlumnoJuegoDeVotacionUnoATodos
                        && inscripcionAlumnoJuegoDeVotacionUnoATodos.Votos'>
    <h2 style="text-align: center;">Esta ha sido tu votación</h2>
    <ion-list *ngIf = 'alumnosVotados'>

        <ion-item *ngFor="let alumno of alumnosVotados; let i=index">
          <ion-item>
          <ion-label>
            {{alumno.Nombre}} {{alumno.PrimerApellido}} {{alumno.SegundoApellido}}
          </ion-label>
          </ion-item>
          <ion-label *ngIf = 'i < juegoSeleccionado.Puntos.length'
                style="font-size: large; color: green; text-align: right;">
          {{juegoSeleccionado.Puntos[i]}}
          </ion-label>
        </ion-item>
    </ion-list>
  </div>

</ion-content>
```

Básicamente tiene dos bloques. En el caso de que no haya votado le presenta la lista de los alumnos y el botón para enviar la votación. La lista se muestra usando un <ion-reorder-groupo> que permite muy cómodamente arrastrar los ítems de la lista para colocarlos en la posición que uno quiera. De esta manera el alumno solo tiene que mover a las primeras posiciones de la lista a los compañeros a los que va a votar. Las primeras posiciones están etiquetadas con los puntos que va a recibir el alumno que ocupa cada una de esas posiciones. La figura 4 muestra cómo se ve en el móvil (justo se está moviendo a Cristina a una de las primeras posiciones):

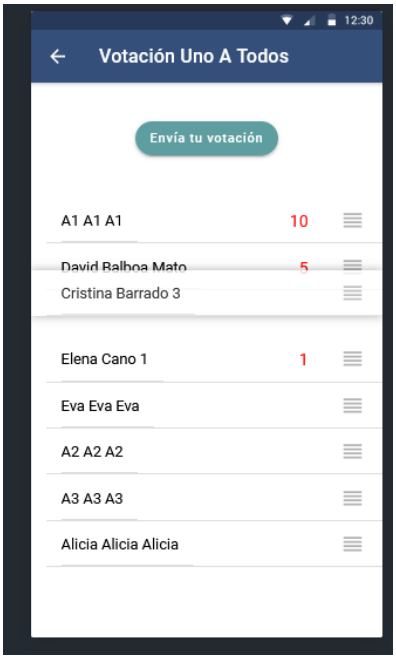

Figura 4: Vista de la página en la que el alumno prepara su votación

Ahora tenemos que añadir dos funciones nuevas al componente. La primera simplemente reordena la lista después de haber movido uno de los ítems.

```
reorderItems(event) {
    const itemMove = this.alumnos.splice(event.detail.from, 1)[0];
    this.alumnos.splice(event.detail.to, 0, itemMove);
    event.detail.complete();
  }
```

La segunda es para registrar la votación:

```
  async Enviar() {

  const confirm = await this.alertCtrl.create({
    header: '¿Seguro que quieres enviar tu votación?',
    buttons: [
      {
        text: 'SI',
        handler: async () => {
          this.inscripcionAlumnoJuegoDeVotacionUnoATodos.Votos = [];
          // tslint:disable-next-line:prefer-for-of
          for (let i = 0; i < this.juegoSeleccionado.Puntos.length; i++) {
            this.inscripcionAlumnoJuegoDeVotacionUnoATodos.Votos[i] = this.alumnos[i].id;
          }
          this.peticionesAPI.RegistraVotacion (this.inscripcionAlumnoJuegoDeVotacionUnoATodos)
          .subscribe (async () => {
            // tslint:disable-next-line:no-shadowed-variable
            const confirm = await this.alertCtrl.create({
              header: 'Votación registrada con éxito',
              buttons: [
                {
                  text: 'OK',
                  handler: async () => {
                    this.alumnosVotados = [];
                    // tslint:disable-next-line:prefer-for-of
                    for (let i = 0;
                        i < this.inscripcionAlumnoJuegoDeVotacionUnoATodos.Votos.length; i++) {
                      const alumno = this.alumnos.filter (al => al.id ===
                              this.inscripcionAlumnoJuegoDeVotacionUnoATodos.Votos[i])[0];
                      this.alumnosVotados.push (alumno);
                    }
                  }
                }

              ]
            });
            await confirm.present();
          });
        }
      }, {
        text: 'NO',
        role: 'cancel',

        handler: () => {
        }
      }
    ]
  });
  await confirm.present();

}
```

Simplemente preparo el vector de votos y registro la votación. Cuando acaba el registro preparo el vector de alumnosVotados para que muestre la lista con los votos emitidos.

Si el alumno ya ha votado, la cosa es más fácil porque solo hay que mostrar la lista de los votos emitidos.

Para acabar tenemos que añadir tres funciones nuevas a peticionesAPI:

```
public DameInscripcionAlumnoJuegoDeVotacionUnoATodos(juegoId: number, alumnoId: number):
                        Observable<AlumnoJuegoDeVotacionUnoATodos[]> {
    return this.http.get<AlumnoJuegoDeVotacionUnoATodos[]>(
        This.APIUrlAlumnoJuegoDeVotacionUnoATodos
                + '?filter[where][juegoDeVotacionUnoATodosId]=' + juegoId
                +  '&filter[where][alumnoId]=' + alumnoId);
}

public DameAlumnosJuegoDeVotacionUnoATodos(juegoId: number): Observable<Alumno[]> {
    return this.http.get<Alumno[]>(this.APIUrlJuegoDeVotacionUnoATodos + '/'
                + juegoId + '/alumnos');
}
// Modifica la inscripcion (la votacion) del alumno
public RegistraVotacion(alumnoJuegoDeVotacionUnoATodos: AlumnoJuegoDeVotacionUnoATodos):
                        Observable<AlumnoJuegoDeVotacionUnoATodos> {
    // tslint:disable-next-line:max-line-length
    return this.http.put<AlumnoJuegoDeVotacionUnoATodos>(
                this.APIUrlAlumnoJuegoDeVotacionUnoATodos + '/'
                + alumnoJuegoDeVotacionUnoATodos.id, alumnoJuegoDeVotacionUnoATodos);
}
```

En estas funciones se usan las declaraciones siguientes:

```
private APIUrlAlumnoJuegoDeVotacionUnoATodos = this.base
                + '3000/api/alumnosJuegoDeVotacionUnoATodos';
private APIUrlJuegoDeVotacionUnoATodos = this.base + '3000/api/juegosDeVotacionUnoATodos';
```

Ahora ya puede hacerce una nueva prueba de funcionamiento. Debe comprobarse que la votación de un alumno se registra bien en la base de datos y que en el componente de seguimiento del Dashboard se muestran correctamente los puntos que tiene cada alumno según se van realizando las votaciones.

5 Seguimiento en tiempo real

Vamos a incorporar ahora la lógica que permite al profesor hacer un seguimiento de las votaciones en tiempo real. Para ello, cada vez que un alumno vota habrá que enviar una notificación al Dashboard para que éste actualice inmediatamente el ranking.

Para gestionar las notificaciones ya tenemos un proyecto Servidor. En el móvil del alumno tenemos un servicio llamado ComServerService que nos permite conectarnos al servidor (cosa que se hace cuando prospera el login) y enviar una notificación. Para enviar la notificación hay que añadir la siguiente línea de código a la función Enviar:

```
this.comServer.Emitir('notificarVotacion',
                { votacion: this.inscripcionAlumnoJuegoDeVotacionUnoATodos});
```

Enviamos el identificador “notificarVotacion” y adjuntamos la inscripción que contiene los datos de la votación.

El servidor debe estar preparado para recibir este tipo de mensajes y reenviarlo al Dashboard. Para ello, hay que añadir las siguientes líneas de código en el fichero app.ts del servidor:

```
 socket.on("notificarVotacion", (res) => {
        console.log("Notifica votacion ");
        dashSocket.emit ("notificarVotacion", res);

 });
```

Cuando el servidor recibe un mensaje de notificación de votación simplemente lo redirige al Dashboard con los datos recibidos.

La parte del Dashboard es algo más complicada. Allí también tenemos un servicio llamado ComServerService con una función para que el Dashboard se conecte al servidor en cuando fructifica el login.

A ese servicio hay que añadir la función siguiente:

```
public EsperoVotacion = () => {
    return Observable.create((observer) => {
        this.socket.on('notificarVotacion', (votacion) => {
            console.log ('llega notificacion');
            observer.next(votacion);
        });
    });
}
```

Esta función permite subscribirse a un observable que espera las notificaciones de votaciones que envía el servidor y las reenvía a los subscriptores.

El Dashboard se subscribirá a este servicio cuando carga el componente JuegoDeVotacionUnoATodosSeleccionadoActivo. Para ello hay que añadir el siguiente código a la función ngOnInit():

```
this.comServer.EsperoVotacion()
   .subscribe((res: any) => {
       for (let i = 0; i < res.votacion.Votos.length; i++) {
         const votado = this.rankingIndividualJuegoDeVotacionUnoATodos.filter (
               al => al.id === res.votacion.Votos[i])[0];
         votado.puntos = votado.puntos + this.juegoSeleccionado.Puntos[i];
         votado.incremento = this.juegoSeleccionado.Puntos[i];
       }
       // Tomo nota de que el alumno ya ha votado
       this.rankingIndividualJuegoDeVotacionUnoATodos.filter (
               al => al.id === res.votacion.alumnoId)[0].votado = true;
       // tslint:disable-next-line:only-arrow-functions
       this.rankingIndividualJuegoDeVotacionUnoATodos =
               this.rankingIndividualJuegoDeVotacionUnoATodos.sort(function(obj1, obj2) {
                       return obj2.puntos - obj1.puntos;
       });
       this.datasourceAlumno = new MatTableDataSource(
               this.rankingIndividualJuegoDeVotacionUnoATodos);

       // Haremos que se muestren los incrementos de esa votaciñón durante 5 segundos
       this.interval = setInterval(() => {
         this.rankingIndividualJuegoDeVotacionUnoATodos.forEach (al => al.incremento = 0);
         clearInterval(this.interval);
       }, 5000);
   });
```

Cuando recibo una votación actualizo los puntos de los alumnos votados y actualizo el campo incremento para que se muestre claramente qué alumnos han recibido votos y cuántos puntos cada uno. Luego marco que el alumno ya ha votado para que se muestre eso también en la tabla y ordeno el ranking. Finalmente, esto lo hago dentro de un temporizador para que los incrementos se muestren solo 5 segundos y luego desaparezcan (porque llegarán nuevas votaciones).

Con todo esto, la tabla de seguimiento de la votación tendrá un aspecto como el de la figura 5, en la que se muestra que ya han votado dos alumnos (ticks en verde) y cuáles son los alumnos beneficiados de la última votación (en rojo, pero solo durante 5 segundos).

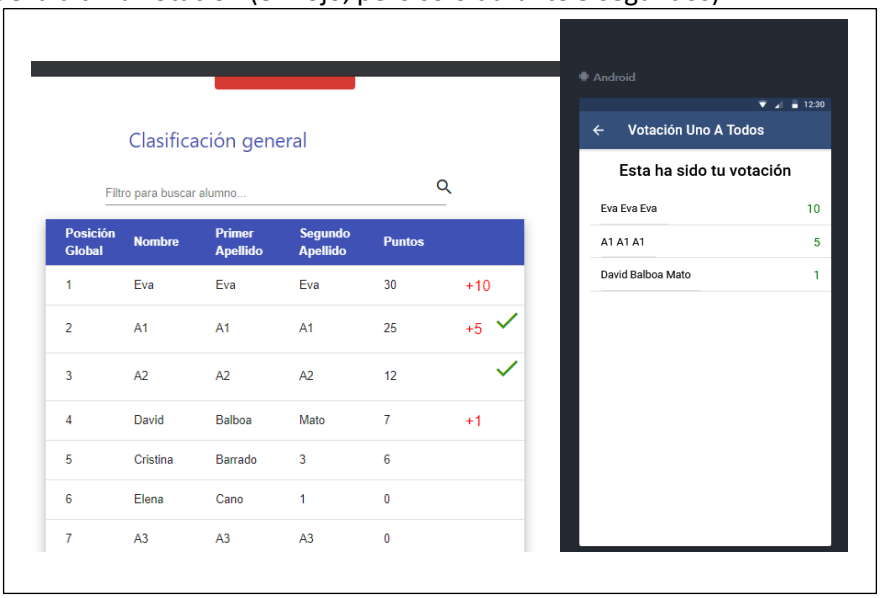

Figura 5: Aspecto del panel de seguimiento en tiempo real (izquierda) en el Dashboad cuando el alumno acaba de hacer una votación (derecha) desde el móvil.

6 Juego inactivo

Ahora vamos a implementar el componente del Dashboard que muestra los datos del juego cuando está inactivo.

Primero añadimos el siguiente código a la plantilla del componente JuegoSeleccionadoInactivo.

```
<div class="contenedor" *ngIf= "juegoSeleccionado.Tipo === 'Juego De Votación Uno A Todos'">
  <app-juego-de-votacion-uno-atodos-seleccionado-inactivo></app-juego-de-votacion-uno-atodos-
seleccionado-inactivo>
</div>
```

De esta manera saltaremos al componente JuegoDeVotacionUnoATodosSeleccionadoInactivo. Ahora tenemos que crear ese componente dentro de la carpeta JuegoSeleccionadoInactivo.

El código de la plantilla HTML de este nuevo componente es este:

```
<div class="contenedor">
  <div class="enColumna">
    <div class="enFila" style="width:40%">
      <button type="button" class="btn Aceptar" (click) = "Reactivar()">Reactivar
                <i class="material-icons">check_circle_outline</i></button>
      <button type="button" class="btn Eliminar" (click) = "Eliminar()"> Eliminar
                <i class="material-icons">delete</i></button>
    </div>

<!-     CLASIFICACIÓN INDIVIDUAL ////////////////////////////////////////////////////-->
  <form style="width: 100%"  *ngIf="juegoSeleccionado.Modo === 'Individual'">
    <div class="filter"  style="width: 60%">
```

```
      <mat-form-field style="width: 95% !Important;">
        <input matInput (keyup)="applyFilter($event.target.value)"
               placeholder="Filtro para buscar alumno...">
        <i class="material-icons" matSuffix>search</i>
      </mat-form-field>
    </div>

    <table style="width: 80%"  align="center"  mat-table
               [dataSource]="datasourceAlumno" class="example-container mat-elevation-z8">
        <ng-container matColumnDef="posicion">
          <th mat-header-cell
                        *matHeaderCellDef class="tituloColumnaTabla"
                        style="width: 20%"  align="center">Posición Global</th>
          <td mat-cell *matCellDef="let alumno; let i = index">
            <div *ngIf = "(i == 0)" style="color: green; font-size: xx-large;">{{i+1}} </div>
            <div *ngIf = "(i != 0)">{{i+1}}</div>
          </td>
        </ng-container>

        <!-- Nombre Alumno Column -->
        <ng-container matColumnDef="nombreAlumno">
          <th mat-header-cell
                        *matHeaderCellDef class="tituloColumnaTabla"
                        style="width: 20%"  align="center">Nombre</th>
          <td mat-cell *matCellDef="let alumno; let i = index">
            <div *ngIf = "(i == 0)" style="color: green; font-size: xx-large;">
                {{alumno.nombre}} </div>
            <div *ngIf = "(i != 0)">{{alumno.nombre}}</div>
          </td>
        </ng-container>

        <!-- Primer Apellido Column -->
        <ng-container matColumnDef="primerApellido">
            <th mat-header-cell
                        *matHeaderCellDef class="tituloColumnaTabla"
                        *style="width: 20%"  align="center">Primer Apellido</th>
            <td mat-cell *matCellDef="let alumno; let i = index">
              <div *ngIf = "(i == 0)" style="color: green; font-size: xx-large;">
                {{alumno.primerApellido}} </div>
              <div *ngIf = "(i != 0)">{{alumno.primerApellido}}</div>
            </td>
        </ng-container>

         <!-- Segundo Apellido Column -->
        <ng-container matColumnDef="segundoApellido">
            <th mat-header-cell
                        *matHeaderCellDef class="tituloColumnaTabla"
                        style="width: 20%"  align="center">Segundo Apellido</th>
            <td mat-cell *matCellDef="let alumno; let i = index">
              <div *ngIf = "(i == 0)" style="color: green; font-size: xx-large;">
                {{alumno.segundoApellido}} </div>
              <div *ngIf = "(i != 0)">{{alumno.segundoApellido}}</div>
            </td>
        </ng-container>

        <!-- Puntos Column -->
        <ng-container matColumnDef="puntos">
            <th mat-header-cell
                        *matHeaderCellDef class="tituloColumnaTabla"
                        style="width: 20%"  align="center">Puntos</th>
            <td mat-cell *matCellDef="let alumno; let i = index">
              <div *ngIf = "(i == 0)" style="color: green; font-size: xx-large;">
                {{alumno.puntos}} </div>
              <div *ngIf = "(i != 0)">{{alumno.puntos}}</div>
            </td>
        </ng-container>
        <tr mat-header-row *matHeaderRowDef="displayedColumnsAlumnos; sticky: true"></tr>
        <tr mat-row *matRowDef="let row; columns: displayedColumnsAlumnos;"></tr>
    </table>
  </form>
  </div>
</div>
```

La vista contiene los botones para reactivar y eliminar el juego, un filtro para seleccionar alumnos y la típica tabla con el ranking, mostrando en verde el alumno que ha resultado ganador en la votación.

El código del componente es similar al caso del juego activo (hay que traer los datos de los alumnos y las inscripciones y construir la tabla con el ranking). Por tanto, se omite aquí.

El código para los botones de reactivar y eliminar es el siguiente:

```
Eliminar() {
  Swal.fire({
    title: '¿Seguro que quieres eliminar el juego de votación?',
    icon: 'warning',
    showCancelButton: true,
    confirmButtonColor: '#3085d6',
    cancelButtonColor: '#d33',
    confirmButtonText: 'Si, estoy seguro'
  }).then((result) => {
    if (result.value) {
      // Primero elimino las inscripciones
      let cont = 0;
      this.listaAlumnosOrdenadaPorPuntos.forEach (inscripcion => {
        this.peticionesAPI.BorraInscripcionAlumnoJuegoDeVotacionUnoATodos (inscripcion.id)
        .subscribe(() => {
          cont++;
          if (cont === this.listaAlumnosOrdenadaPorPuntos.length) {
            // Ya están todas las inscripciones eliminadas
            // ahora elimino el juego
            this.peticionesAPI.BorraJuegoDeVotacionUnoATodos (this.juegoSeleccionado.id)
            .subscribe(() => {
              Swal.fire('El juego se ha eliminado correctamente');
              this.location.back();
            });
          }
        });
      });
    }
  });
}

Reactivar() {
  Swal.fire({
    title: '¿Seguro que quieres activar el juego de votación?',
    icon: 'warning',
    showCancelButton: true,
    confirmButtonColor: '#3085d6',
    cancelButtonColor: '#d33',
    confirmButtonText: 'Si, estoy seguro'
  }).then((result) => {
    if (result.value) {

      this.juegoSeleccionado.JuegoActivo = true;
      this.peticionesAPI.CambiaEstadoJuegoDeVotacionUnaATodos (this.juegoSeleccionado)
      .subscribe(res => {
          if (res !== undefined) {
            Swal.fire('El juego se ha activado correctamente');
            this.location.back();
          }
      });
    }
  });
}
```

Para eliminar, primero eliminamos todas las inscripciones y luego borramos el juego. Hay que añadir dos funciones nuevas a peticionesAPI:

```
public BorraInscripcionAlumnoJuegoDeVotacionUnoATodos(alumnoJuegoDeVotacionUnoATodosId: number)
{
    // tslint:disable-next-line:max-line-length
    return this.http.delete<AlumnoJuegoDeVotacionUnoATodos>(
        this.APIUrlAlumnoJuegoDeVotacionUnoATodos + '/' + alumnoJuegoDeVotacionUnoATodosId);
}

public BorraJuegoDeVotacionUnoATodos(juegoId: number): Observable<JuegoDeVotacionUnoATodos> {
    return this.http.delete<JuegoDeVotacionUnoATodos>(this.APIUrlJuegoDeVotacionUnoATodos
        + '/' + juegoId);
}
```

Finalmente, sería necesario crear una página en el móvil del alumno para mostrar el juego inactivo. La página debería mostrar también el resultado final del juego y quizá también debería permitir recordar cuál fue la votación que hizo el jugador. Esto queda como ejercicio.

7 Borrar grupo

El último paso es añadir a los juegos de votación en los tipos de juegos que hay que borrar cuando se borra un grupo. El botón de borrar grupo en la página del grupo recupera todos los tipos de juegos que tiene el grupo para eliminar las inscripciones de los alumnos en el juego y luego eliminar el propio juego. Eso se hace en la función EliminaJuegos que está en CalculosService. Hay que añadir a esa función el código siguiente:

```
} else if (juego.Tipo === 'Juego De Votación Uno A Todos') {
                if (juego.Modo === 'Individual') {
                      this.peticionesAPI.DameAlumnosJuegoDeVotacionUnoATodos(juego.id)
                      .subscribe( AlumnosDelJuego => {
                        if (AlumnosDelJuego[0] !== undefined) {
                          // Una vez recibo las inscripciones, las voy borrando una a una
                          // tslint:disable-next-line:prefer-for-of
                          for (let i = 0; i < AlumnosDelJuego.length; i++) {
                            this.peticionesAPI.BorraInscripcionAlumnoJuegoDeVotacionUnoATodos(
                                        AlumnosDelJuego[i].id)
                            .subscribe(() => {
                                console.log('Inscripcion al juego borrada correctamente');
                            });
                          }
                        } else {
                          console.log('No hay alumnos en el juego de votacion');
                        }
                      });
                    }
                    this.peticionesAPI.BorraJuegoDeVotacionUnoATodos (juego.id)
                    .subscribe (() => {
                        // Esto es lo que no hace la funcion que borra el juego de liga
                        cont++;
                        if (cont === juegos.length) {
                          obs.next();
                        }
                    });
                  }
```