```
app.controller('TournamentController', ['$scope', '$rootScope', '$location', '$http',
'$routeParams', '$uibModal', '$window', '$filter', '$location', function ($scope,
$rootScope, $location, $http, $routeParams, $uibModal, $window, $filter, $location) {

  $scope.new = false;
  $scope.newDivName = "";
  $scope.chooseField = "";
  $scope.newMatchDuration = "";
  $scope.tournamentId = $routeParams.tournamentId;

  //Get request - Gets tournament data, creates three field-arrays dependant on
fieldsizes.
  //Loads division data into divisions, and the whole data into the tournament.
  $scope.getDivisions = function(){
    $http.get($rootScope.apiUrl + "/Tournament/Details?id=" +  $routeParams.tournamentId)
      .success(function(data)
      {
        $rootScope.IsScheduled = data.IsScheduled;
        if(data.status === "success"){
          $scope.EmFields = [];
          $scope.OmFields = [];
          $scope.FmFields = [];
          for (var i=0; i < data.Fields.length; i++)
          {
            if(data.Fields[i].fieldSize === 11)
            {
              $scope.EmFields.push(data.Fields[i]);
            }
            else if(data.Fields[i].fieldSize === 8)
            {
              $scope.OmFields.push(data.Fields[i]);
            }
            else
            {
              $scope.FmFields.push(data.Fields[i]);
            }
          }
          $scope.tournament = data;
        } else  {
            $scope.error = "Række kunne ikke læses";
        }
      }).error(function (err) {
        $scope.error = err;
      })
  }

  $scope.getDivisions();

  //Variable used to hide and show a button.
  $scope.createNew = function () {
    $scope.new = !$scope.new;
  }

  /* Modal start */
  $scope.animationsEnabled = true;

  //The modal-open function, used for the "form"-like modal,
  //which is used to submit a division.
```

```
  $scope.open = function (size) {

    var uibModalInstance = $uibModal.open({
      animation: $scope.animationsEnabled,
      templateUrl: 'myModalContent.html',
      controller: 'ModalInstanceCtrl',
      size: size,
      /*resolve: {
        items: function () {
          return $scope.items;
        }
      }*/
    });

    uibModalInstance.result.then(function () {
      $scope.getDivisions();
        $scope.createNew();
    }, function () {
    });
  };

  //Anoter variable used to show and hide the modal.
  $scope.toggleAnimation = function () {
    $scope.animationsEnabled = !$scope.animationsEnabled;
  };
  /* Modal end */

//Redirecting function, redirecting to the specific division.
  $scope.gotoDivison = function (currDiv) {
    $location.url("tournament/" + $routeParams.tournamentId+ "/division/" + currDiv.Id);
  }

  /* Field start */
  $scope.EmField = "";
  $scope.OmField = "";
  $scope.FmField = "";
  $scope.newEm = false;
  $scope.newOm = false;
  $scope.newFm = false;

  //Function to submit a field, adding it to the correct field-size array.
  $scope.submitField = function(fieldName, fieldSize) {
    $scope.buttonDisabled = true;
    $http.post($rootScope.apiUrl + "/Field/Create", { name: fieldName, size: fieldSize,
tournamentId: $routeParams.tournamentId })
    .success(function(data){
        if(data.status === "success"){
          if(fieldSize === 11){
            $scope.createNewEmField();
          }
          else if(fieldSize === 8){
            $scope.createNewOmField();
          }
          else {
            $scope.createNewFmField();
          }
            $scope.getDivisions();
        }
```

```
        else
        {
          $scope.error = "Bane ikke oprettet";
          $scope.buttonDisabled = false;
        }
        $scope.buttonDisabled = false;
    }).error(function(err){
      $scope.createErr = err;
    }).finally(function(hej){
      $scope.getDivisions();
    })
    $scope.getDivisions();
  }
  $scope.buttonDisabled = false;

  //Delete post-request, used to delete a field, and removing it from the database.
  $scope.removeField = function(Field) {
    var deleteField = $window.confirm('Er du sikker på du vil slette banen? \nHvis
kampprogrammet har været planlagt, vil det nu blive slettet og kan tage lidt tid');

    if(deleteField){
      $http.post($rootScope.apiUrl + "/Field/Delete", { id: Field.Id })
      .success(function(data){
        if(data.status === "success")
        {

        }
        else {
          $scope.error = "Bane ikke fjernet";
        }
      }).error(function(err){
        $scope.deleteErr = err;
      }).finally(function(hej) {
        $scope.getDivisions();
      })
    }
  }

  /* 11man */ //Variables used to show and hide the submit field buttons.
  $scope.createNewEmField = function() {
    $scope.newEm = !$scope.newEm;
  }

  /* 8man */
 $scope.createNewOmField = function() {
    $scope.newOm = !$scope.newOm;
  }

  /* 5man */
  $scope.createNewFmField = function() {
    $scope.newFm = !$scope.newFm;
  }

  /* Field end */

  //Schedule funcion, used to schedule a tournaments matches.
  $scope.scheduleTournament = function () {
    $rootScope.scheduler($routeParams.tournamentId);
```

```
  }

  $scope.chosenDay = 0;
  $scope.incDay = function() {
    if ($scope.chosenDay < $scope.tournament.TimeIntervals.length-1)
      $scope.chosenDay++;
  }
  $scope.decDay = function() {
    if ($scope.chosenDay > 0)
      $scope.chosenDay--;
  }
  $scope.getData = function(){
    $http.get($rootScope.apiUrl + "/Field/GetAllTournamentFields?tournamentId=" +
$routeParams.tournamentId).success(function(data) {
      $scope.tournament2 = data;
      $scope.days = [];
      for (var i=0; i < $scope.tournament2.Fields.length; i++) {
        for (var j=0; j < $scope.tournament2.Fields[i].matches.length; j++) {
          $scope.tournament2.Fields[i].matches[j].StartTime =
$filter('jsonOnlyTime')($scope.tournament2.Fields[i].matches[j].StartTime);
          $scope.tournament2.Fields[i].matches[j].EndTime =
$filter('jsonOnlyTime')($scope.tournament2.Fields[i].matches[j].EndTime);
          $scope.tournament2.Fields[i].matches[j].Date =
$filter('jsonOnlyDate')($scope.tournament2.Fields[i].matches[j].Date);
          if (j > 0)
          {
            if ($scope.tournament2.Fields[i].matches[j-1].EndTime !=
$scope.tournament2.Fields[i].matches[j].StartTime)
            {
              var eTime = new Date, time = $scope.tournament2.Fields[i].matches[j-
1].EndTime.split(/\:|\-/g);
              eTime.setHours(time[0]);
              eTime.setMinutes(time[1]);
              var sTime = new Date, time =
$scope.tournament2.Fields[i].matches[j].StartTime.split(/\:|\-/g);
              sTime.setHours(time[0]);
              sTime.setMinutes(time[1]);
              $scope.tournament2.Fields[i].matches[j].Pause = (sTime - eTime) / 1000 / 60
+ 'px';
            }
          }
        }
      }
      for (var i=0; i < $scope.tournament2.TimeIntervals.length; i++) {
        $scope.tournament2.TimeIntervals[i].StartTime =
$filter('jsonOnlyTime')($scope.tournament2.TimeIntervals[i].StartTime);
        $scope.tournament2.TimeIntervals[i].EndTime =
$filter('jsonOnlyTime')($scope.tournament2.TimeIntervals[i].EndTime);
        $scope.tournament2.TimeIntervals[i].Date =
$filter('jsonOnlyDate')($scope.tournament2.TimeIntervals[i].Date);
      }
    })
  }
  $scope.getData();

  $scope.GetMinutes = function(endTime, startTime) {
    var eTime = new Date, time = endTime.split(/\:|\-/g);
    eTime.setHours(time[0]);
```

```
    eTime.setMinutes(time[1]);
    var sTime = new Date, time = startTime.split(/\:|\-/g);
    sTime.setHours(time[0]);
    sTime.setMinutes(time[1]);
    return { height: (sTime - eTime) / 1000 / 60 + 'px' };
  }

}]);

//ModalInstanceController, the functions used to add new divisions,
//added through the modal.
app.controller('ModalInstanceCtrl', ['$scope', '$rootScope', '$uibModalInstance',
'$http', '$routeParams', function ($scope, $rootScope, $uibModalInstance, $http,
$routeParams) {

    $scope.newDivName = "";
    $scope.newMatchDuration = "";
    $scope.chooseField = "";

    //Error message function, used to show the error messages of the post-request
    //in submitNewDiv().
    $scope.errMsg = function () {
      $scope.error = !$scope.error;
    }

    //Post-request to add new divisions to the tournament.
  $scope.submitNewDiv = function(newDivName, newMatchDuration, chooseField) {
    if(newMatchDuration >= 5 && newMatchDuration <= 70 && chooseField != "")
    {
      $scope.buttonDisabled = true;
    $http.post($rootScope.apiUrl + "/Division/Create", { Name: newDivName, MatchDuration:
newMatchDuration, FieldSize: chooseField, tournamentId: $routeParams.tournamentId })
      .success(function(data){
        if(data.status === "success"){
          $uibModalInstance.close();
          $scope.newDivName = "";
          $scope.newMatchDuration = "";
          $scope.chooseField = "";
          //$scope.getDivisions();
        }
        else {
          $scope.error = "Række ikke tilføjet";
          $scope.buttonDisabled = false;
        }
      }).error(function(data){
        $scope.newDivError = data;
      })
    }
    else
    {
      $scope.error = "Kamplængde eller banestørrelse ugyldig";
      $scope.buttonDisabled = false;
    }
  }
  $scope.buttonDisabled = false;

  //Functions used to close the modal.
  $scope.ok = function () {
```

```
    $uibModalInstance.close();
  };

  $scope.cancel = function () {
    $uibModalInstance.dismiss('cancel');
  };


}]);

//Create Tournament Controller, used to control the creation of tournament and tournament
data.
app.controller('CreateTournyController', ['$scope', '$rootScope', '$http', '$location',
'$routeParams', 'FileUploader', 'cfpLoadingBar', function ($scope, $rootScope, $http,
$location, $routeParams, FileUploader, cfpLoadingBar) {

  $scope.tournamentData =  {};

  //File-uploader to create tournaments through excel files.
  var uploader = $scope.uploader = new FileUploader({
    url: $rootScope.apiUrl + '/Tournament/Create'
  });

  //Function to send data when item is selected.
  uploader.onBeforeUploadItem = function(item) {
    item.formData.push($scope.tournamentData);
  };

  //Funcion to send data when an item is successfully uploaded.
  uploader.onSuccessItem = function(fileItem, response, status, headers) {
      cfpLoadingBar.complete();
    $location.path("tournament/" + response.id);
  };

  //Function to send data when an item is not succcessfully uploaded.
  uploader.onErrorItem = function(fileItem, response, status, headers) {
      cfpLoadingBar.complete();
  };

  /* DATE PICKER START */ //The datepicker is used to generate and handle the time-
intervals
  // for a tournament, and it's teams. A default value is generated for each team,
corresponding
  // to the tournaments time-intervals. This can be edited in the team-view.
  $scope.dateRange = 0;
  $scope.today = function () {
    $scope.startDate = new Date();
    $scope.startDate.setHours(0);
    $scope.startDate.setSeconds(0);
    $scope.startDate.setMinutes(0);
    $scope.startDate.setMilliseconds(0);
    $scope.endDate = new Date();
    $scope.endDate.setHours(0);
    $scope.endDate.setSeconds(0);
    $scope.endDate.setMinutes(0);
    $scope.endDate.setMilliseconds(0);
  };
  $scope.today();
```

```
$scope.dateArray = [$scope.startDate];
$scope.startTimes = [$scope.startDate];
$scope.endTimes = [$scope.startDate];

$scope.toggleMin = function () {
  $scope.minDate = $scope.minDate ? null : new Date();
};
$scope.toggleMin();
$scope.maxDate = new Date(2020, 5, 22);
$scope.openEndDate = function ($event) {
  $scope.statusEndDate.opened = true;
};

$scope.isChanged = function () {
  $scope.dateRange = ($scope.endDate - $scope.startDate) / (1000 * 60 * 60 * 24);
  $scope.dateArray = [];
  $scope.startTimes = [];
  $scope.endTimes = [];
  for (var i = 0; i <= $scope.dateRange; i++) {
    var date = new Date($scope.startDate.getTime());
    date.setDate(date.getDate() + i);
    $scope.dateArray.push(date);
    $scope.startTimes.push(date);
    $scope.endTimes.push(date);
  }
}

$scope.openStartDate = function ($event) {
  $scope.statusStartDate.opened = true;
};
$scope.dateOptions = {
  formatYear: 'yy',
  startingDay: 1
};
$scope.formats = ['dd-MMMM-yyyy', 'yyyy/MM/dd', 'dd.MM.yyyy', 'shortDate'];
$scope.format = $scope.formats[0];
$scope.statusStartDate = {
  opened: false
};
$scope.statusEndDate = {
  opened: false
};
/* DATE PICKER END */

//Function to show and hide error messages.
$scope.errMsg = function () {
  $scope.error = !$scope.error;
}

//Upload tournament function, which shows corresponding errors to user input
//and post-requests to create a tournament.
$scope.uploadTournament = function ()
{
  $scope.startTimesString = "";
  $scope.endTimesString = "";
  if (!$scope.tournamentName || !$scope.tournamentPassword){
    $scope.error = "Navn eller kode ikke sat";
```

```
      $scope.errStatus = true;
    }else{
      $scope.startDateTimes = [];
      $scope.endDateTimes = [];
      $scope.error = false;
      for (var index = 0; index <= $scope.dateRange; index++) {
        $scope.startDateTimes[index] = $scope.startTimes[index].toISOString();
        $scope.endDateTimes[index] = $scope.endTimes[index].toISOString();
      }

      if($scope.startDateTimes.length-1 !== $scope.dateRange &&
$scope.endDateTimes.length-1 !== $scope.dateRange){
        $scope.error = "Fejl i start eller slut tidspunkt for en af dagene";
      }else{
        for(var i = 0; i <= $scope.dateRange; i++){
          if($scope.startDateTimes[i] >= $scope.endDateTimes[i]){
            $scope.error = "Alle slut tidspunkter skal være senere end start
tidspunkter";
          }
          $scope.startTimesString += $scope.startDateTimes[i] + (i == $scope.dateRange ?
'': ',');
          $scope.endTimesString += $scope.endDateTimes[i] + (i == $scope.dateRange ? '':
',');
        }
        if(!$scope.error){
          $scope.tournamentData = {
            name: $scope.tournamentName,
            password: $scope.tournamentPassword,
            startTimes: $scope.startTimesString,
            endTimes: $scope.endTimesString
          }
          if (uploader.queue.length > 0)
          {
            cfpLoadingBar.start();
            uploader.queue[0].upload();
          }
          else
          {
            $http.post($rootScope.apiUrl + "/Tournament/Create/",
$scope.tournamentData).success(function(Data)
            {
              if(Data.message == "Password already exists"){
                $scope.error = "Adgangskoden eksisterer allerede";
              } else {
                $location.path("tournament/" + Data.id);
              }
            }).error(function(err)
            {
              $scope.error = "Kunne ikke uploade til serveren";
            });
          }
        }
      }
    }
  }
}]);

//EditTournamentController has the same functionalities as the create controller,
```

```
//but post-requests sends data along with it, to edit a tournament which has already
//been created.
app.controller('EditTournamentController', ['$scope', '$rootScope', '$window', '$http',
'$location', '$routeParams', function ($scope, $rootScope, $window, $http, $location,
$routeParams) {

  $http.get($rootScope.apiUrl + "/Tournament/Details?id=" +
$routeParams.tournamentId).success(function(data){

    if(data.status === "success"){

      $scope.tournamentId = data.Id;
      $scope.tournamentName = data.Name;
      $scope.tournamentPassword = data.Password;
      $scope.dateArray = [];
      $scope.startTimes = [];
      $scope.endTimes = [];

      $scope.startDate = new Date(parseInt(data.TimeIntervals[0].StartTime.substr(6)));
      $scope.startDate.setHours(0);
      $scope.startDate.setSeconds(0);
      $scope.startDate.setMinutes(0);
      $scope.startDate.setMilliseconds(0);
      $scope.endDate = new Date(parseInt(data.TimeIntervals[data.TimeIntervals.length-
1].EndTime.substr(6)));
      $scope.endDate.setHours(0);
      $scope.endDate.setSeconds(0);
      $scope.endDate.setMinutes(0);
      $scope.endDate.setMilliseconds(0);
      $scope.dateRange = ($scope.endDate - $scope.startDate) / (1000 * 60 * 60 * 24);

      for (var index = 0; index < data.TimeIntervals.length; index++) {
        var date = new Date($scope.startDate.getTime());
        date.setDate(date.getDate() + index);
        $scope.dateArray.push(date);
        $scope.startTimes.push(new
Date(parseInt(data.TimeIntervals[index].StartTime.substr(6))));
        $scope.endTimes.push(new
Date(parseInt(data.TimeIntervals[index].EndTime.substr(6))));
      }
      $scope.toggleMin = function () {
        $scope.minDate = $scope.minDate ? null : new Date();
      };
      $scope.toggleMin();
      $scope.maxDate = new Date(2020, 5, 22);
      $scope.openEndDate = function ($event) {
        $scope.statusEndDate.opened = true;
      };

      $scope.isChanged = function () {
        $scope.dateRange = ($scope.endDate - $scope.startDate) / (1000 * 60 * 60 * 24);
        $scope.dateArray = [];
        $scope.startTimes = [];
        $scope.endTimes = [];
        for (var i = 0; i <= $scope.dateRange; i++) {
          var date = new Date($scope.startDate.getTime());
          date.setDate(date.getDate() + i);
          $scope.dateArray.push(date);
```

```
            $scope.startTimes.push(date);
            $scope.endTimes.push(date);
          }
        }

        $scope.openStartDate = function ($event) {
          $scope.statusStartDate.opened = true;
        };
        $scope.dateOptions = {
          formatYear: 'yy',
          startingDay: 1
        };
        $scope.formats = ['dd-MMMM-yyyy', 'yyyy/MM/dd', 'dd.MM.yyyy', 'shortDate'];
        $scope.format = $scope.formats[0];
        $scope.statusStartDate = {
          opened: false
        };
        $scope.statusEndDate = {
          opened: false
        };

        $scope.errMsg = function () {
          $scope.error = !$scope.error;
        }

        $scope.uploadTournament = function () {
          if (!$scope.tournamentName || !$scope.tournamentPassword){
            $scope.error = "Navn eller kode ikke sat";
          }else{
            $scope.startDateTimes = [];
            $scope.endDateTimes = [];
            $scope.error = false;
            for (var index = 0; index <= $scope.dateRange; index++) {
              if($scope.startTimes[index] !== null && $scope.endTimes[index] !== null)
              {
              $scope.startDateTimes[index] = $scope.startTimes[index].toISOString();
              $scope.endDateTimes[index] = $scope.endTimes[index].toISOString();
              }
              else
              {
                $scope.error = "Dette er et udgyldigt tidspunkt";
              }

            }
            if($scope.startDateTimes.length-1 !== $scope.dateRange &&
$scope.endDateTimes.length-1 !== $scope.dateRange){
              $scope.error = "Fejl i start eller slut tidspunkt for en af dagene";
            }else{
              for(var i = 0; i <= $scope.dateRange; i++){
                if($scope.startDateTimes[i] >= $scope.endDateTimes[i]){
                  $scope.error = "Alle slut tidspunkter skal være senere end start
tidspunkter";
                }
              }
              if(!$scope.error){
                var tournamentData = {
                  id: $scope.tournamentId,
                  name: $scope.tournamentName,
```

```
                password: $scope.tournamentPassword,
                startTimes: $scope.startDateTimes,
                endTimes: $scope.endDateTimes
              }
              var save = $window.confirm('Ved at gemme denne data, vil kampprogrammet
slettes.');
              if (save) {
                  $http.post($rootScope.apiUrl + "/Tournament/Edit/",
tournamentData).success(function(Data)
                  {
                    if(Data.message === "Password already exists"){
                    $scope.error = "Adgangskoden eksisterer allerede";
                    }
                    if(Data.status === "success"){
                      $location.path("tournament/" + $routeParams.tournamentId);
                    } else {
                      $scope.error = "Kunne ikke redigere turnering";
                    }
                  }).error(function(err)
                  {
                    $scope.error = "Kunne ikke uploade til serveren";
                  });
              }

            }
          }
        }
      }
    }else{
      $scope.error = "Kunne ikke finde turneringen";
    }
  }).error(function(err){
    $scope.error = "Kunne ikke finde turneringen";
  });
}]);
```