```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
using System.Data.Entity;
using CupPlaner.Helpers;

namespace CupPlaner.Controllers
{
    // Division controller with CRUD functions.
    public class DivisionController : Controller
    {
        // Database container, has functionalities to connect to the database classes.
        CupDBContainer db = new CupDBContainer();
        ScheduleManager sm = new ScheduleManager();

        // GET: Division/Details/5 - Fetches the details of the class, takes the "id"
parameter to determine the corresponding Divison object.
        // Returns a Json object, which contains a copy of the corresponding Divisions
variables.
        public ActionResult Details(int id)
        {
            try
            {
                Validator validator = new Validator();
                Division d = db.DivisionSet.Find(id);
                Tournament tourny = db.TournamentSet.Find(d.Tournament.Id);
                List<object> pools = new List<object>();
                List<object> teams = new List<object>();
                List<object> matches = new List<object>();
                List<object> finalslinks = new List<object>();

                bool FrontendValidation = validator.IsScheduleReady(tourny.Id);
                bool enoughTeams = false;

                // For finalstage
                string letters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

                // Get all pools in the division and their teams
                if (d.Pools != null)
                {
                    foreach (Pool p in d.Pools)
                    {
                        teams = new List<object>();
                        if (p.Teams.Count >= 2)
                        {
                            enoughTeams = true;
                        }

                        foreach (Team t in p.Teams)
                        {
                            teams.Add(new { Name = t.Name, Id = t.Id });
                        }
                        pools.Add(new { Id = p.Id, Name = p.Name, Teams = teams });
                    }
                }

                // Get matches in division
```

```
                if (d.DivisionTournament != null &&
d.DivisionTournament.TournamentStage.Count > 0)
                {
                    foreach (TournamentStage ts in d.DivisionTournament.TournamentStage)
                    {
                        if (ts.Matches.Count > 0)
                        {
                            foreach (Match m in ts.Matches)
                            {
                                Team team1 = m.Teams.ToList()[0];
                                Team team2 = m.Teams.ToList()[1];

                                matches.Add(new { Id = m.Id, Number = m.Number, StartTime
= m.StartTime, /*FieldName = m.Field.Name,*/ Pool = new { Id = team1.Pool.Id, Name =
team1.Pool.Name }, Team1 = new { name = team1.Name, Id = team1.Id }, Team2 = new { name =
team2.Name, Id = team2.Id } });
                            }
                        }
                    }
                }

                // Get finals links
                if (d.FinalsLinks.Count > 0)
                {
                    foreach (FinalsLink fl in d.FinalsLinks)
                    {
                        finalslinks.Add(new { Id = fl.Id, PoolPlacement =
fl.PoolPlacement, Finalsstage = letters[fl.Finalstage - 1] });
                    }
                }

                object obj = new { status = "success", Id = d.Id, Name = d.Name,
FinalsStage = d.TournamentStructure, Pools = pools, Teams = teams, FieldSize =
d.FieldSize, MatchDuration = d.MatchDuration, Matches = matches, FinalsLinks =
finalslinks, isValid = FrontendValidation, isTeamsValid = enoughTeams };

                return Json(obj, JsonRequestBehavior.AllowGet);
            }
            catch (Exception ex)
            {
                return Json(new
                {
                    status = "error",
                    message = "Could not find division",
                    details = ex.Message
                }, JsonRequestBehavior.AllowGet);
            }

        }

        // POST: Division/Create - Tries to create a Division object, with the parameters
"name" and "tournamentId".
        // Sets the "FieldSize" and "MatchDuration" to the input given in the web input.
        // Adds the Division object to the database DivisionSet, and saves the changes in
the database.
        // Returns a Json object with a state, indicating whether it succeeded creating
the Division object or not.
        [HttpPost]
```

```
        public ActionResult Create(string name, int tournamentId, int MatchDuration,
FieldSize FieldSize)
        {
            try
            {
                Tournament t = db.TournamentSet.Find(tournamentId);

                sm.DeleteSchedule(t.Id, db);

                Division d = db.DivisionSet.Add(new Division() { Name = name, FieldSize =
FieldSize, MatchDuration = MatchDuration, Tournament = t });

                db.SaveChanges();

                return Json(new { status = "success", message = "New division added", id
= d.Id }, JsonRequestBehavior.AllowGet);
            }
            catch (Exception ex)
            {
                return Json(new { status = "error", message = "New division not added",
details = ex.Message }, JsonRequestBehavior.AllowGet);
            }
        }

        // POST: Division/Edit/5 - Tries to edit a Division, determined by the "id"
parameter and "tournamentId".
        // Edits a divisions name, FieldSize and matchDuration. Saves the changes to the
database, if succeeded.
        // Returns a Json object with a state, indicating whether it succeeded editing
the Division object or not.
        [HttpPost]
        public ActionResult Edit(int id, string name, int tournamentId, int fieldSizeInt,
int matchDuration)
        {
            try
            {

                Division d = db.DivisionSet.Find(id);
                d.Tournament = db.TournamentSet.Find(tournamentId);

                d.Name = name;

                // Clear the schedule
                sm.DeleteSchedule(d.Tournament.Id, db);

                if (d.FieldSize != (FieldSize)fieldSizeInt)
                {
                    foreach (Pool p in d.Pools)
                    {
                        p.FavoriteFields.Clear();
                    }
                    d.FieldSize = (FieldSize)fieldSizeInt;
                }

                d.MatchDuration = matchDuration;

                db.Entry(d).State = EntityState.Modified;
                db.SaveChanges();
```

```
                return Json(new { status = "success", message = "Division edited" },
JsonRequestBehavior.AllowGet);
            }
            catch (Exception ex)
            {
                return Json(new { status = "error", message = "Division not edited",
details = ex.Message }, JsonRequestBehavior.AllowGet);
            }
        }

        // POST: Division/Delete/5 - Tries to delete a Division object, determined by the
"id".
        // Deletes the Division object, all Pool objects contained in the Division, all
the FinalsLink's made, and saves to the database, if succeeded.
        // Returns a Json object, indicating whether it succeeded deleting the Division
object and pools, or not.
        [HttpPost]
        public ActionResult Delete(int id)
        {
            try
            {
                Division d = db.DivisionSet.Find(id);

                // Clear the schedule
                sm.DeleteSchedule(d.Tournament.Id, db);

                // Remove dependencies
                foreach (Pool p in d.Pools)
                {
                    foreach (Team team in p.Teams.ToList())
                    {
                        db.MatchSet.RemoveRange(team.Matches);
                        team.TimeIntervals.Clear();
                    }
                    db.TeamSet.RemoveRange(p.Teams);
                    p.FavoriteFields.Clear();
                }
                db.PoolSet.RemoveRange(d.Pools);
                db.FinalsLinkSet.RemoveRange(d.FinalsLinks);
                db.DivisionSet.Remove(d);
                db.SaveChanges();
                return Json(new { status = "success", message = "Division deleted" },
JsonRequestBehavior.AllowGet);
            }
            catch (Exception ex)
            {
                return Json(new { status = "error", message = "Division not deleted",
details = ex.Message }, JsonRequestBehavior.AllowGet);
            }
        }

        [HttpPost]
        public ActionResult ChangeStructure(int divisionId, int typeId)
        {
            Division d = db.DivisionSet.Find(divisionId);
            // Clear the schedule
            sm.DeleteSchedule(d.Tournament.Id, db);
```

```
            d.TournamentStructure = (TournamentStructure)typeId;
            db.Entry(d).State = EntityState.Modified;
            db.SaveChanges();

            return Json(new { State = "Success" });
        }
    }
}
```