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

namespace CupPlaner.Controllers
{
    public class PoolController : Controller
    {
        CupDBContainer db = new CupDBContainer();
        ScheduleManager sm = new ScheduleManager();

        // GET: Pool/Details/5 - The Details function will get the details of the Pool
class, with the pools id as a parameter.
        // The function will get the details of the Teams, FavoriteFields and Matches for
the pool.
        // A JSON object will be returned with the data and send an status message with
it if it either have succeeded or failed.
        public ActionResult Details(int id)
        {
            try
            {
                Pool p = db.PoolSet.Find(id);
                Tournament tourny = db.TournamentSet.Find(p.Division.Tournament.Id);
                Validator validator = new Validator();

                List<object> teams = new List<object>();
                List<object> ffs = new List<object>();
                List<object> matches = new List<object>();

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

                // Get teams in pool
                if (p.Teams != null)
                {
                    foreach (Team t in p.Teams)
                    {
                        teams.Add(new { Id = t.Id, Name = t.Name });
                    }
                }
                // Get favorite fields in pool
                if (p.FavoriteFields != null)
                {
                    foreach (Field f in p.FavoriteFields)
                    {
                        ffs.Add(new { Id = f.Id, Name = f.Name });
                    }
                }
                // Get matches in pool
                if (p.TournamentStage != null && p.TournamentStage.Matches.Count > 0)
                {
                    foreach (Match m in p.TournamentStage.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, Team1 = new { name = team1.Name, Id = team1.Id },
Team2 = new { name = team2.Name, Id = team2.Id } });
                    }
                }
                object obj = new { status = "success", Id = p.Id, Name = p.Name,
FieldSize = p.Division.FieldSize, Teams = teams, FavoriteFields = ffs, Matches = matches,
isValid = FrontendValidation };

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

        // POST: Pool/Create - The Create function will create a new object of a pool
with the name and an id of the division it is about to be created in.
        // The function will add the pool object to the PoolSet in the database and save
it.
        // If the function runs successfully, it will send back an "success" status, and
an "error" status if failed.
        [HttpPost]
        public ActionResult Create(string name, int divisionId)
        {
            try
            {
                Division d = db.DivisionSet.Find(divisionId);

                Pool p = db.PoolSet.Add(new Pool() { Name = name, Division = d });

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

                db.SaveChanges();

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


        // POST: Pool/Edit/5 - The Edit function can edit the values of a specific pool
obejct.
        // The function can change the pools name or the list of fieldIds which will be
the fieldId's for the favoriteFields.
        // The function will either send back an "succes" or an "error" message if the
functions eitehr succeeds or fails.
        [HttpPost]
        public ActionResult Edit(int id, string name, int divisionId)
        {
```

```
            try
            {
                Pool p = db.PoolSet.Find(id);
                p.Name = name;
                p.Division = db.DivisionSet.Find(divisionId);

                db.Entry(p).State = EntityState.Modified;
                db.SaveChanges();

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

        // POST: Pool/Delete/5 - The delete function will delete a pool with the
corresponding id.
        // It will delete all of the matches if any were generated, will remove the teams
TimeIntervals and FavoriteFields, it will also remove all the teams in the pool
        // and ofcourse delte the pool itself.
        // If the function runs successfully, it will send back an "success" status, and
an "error" status if failed.
        [HttpPost]
        public ActionResult Delete(int id)
        {
            try
            {
                Pool p = db.PoolSet.Find(id);

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

                if(p.IsAuto == false)
                {
                    // Remove dependencies
                    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.Remove(p);
                }

                db.SaveChanges();

                return Json(new { status = "success", message = "Pool deleted" });
            }
            catch (Exception ex)
            {
                return Json(new { status = "error", message = "Pool not deleted", details
= ex.Message });
```

```
            }
        }
    }
}
```