REFACTORING

Koristeći se radnim okruženjem MS VS2019 i njegovim ekstenzijama stavili smo se u ulogu Martina Fowlera te smo po njegovom citatu izvršavali samo izmjene u kodu bez da “štimamo” da program radi već da zaista bude izvorni refactoring na način da gledamo samo kôd te da očuvamo sistem bez oštećenja.
To nas je dovelo do raznih iskušenja na način da zaista aplikacija nije bila u stanju da se pokrene zbog silnih grešaka iz nepoznatih razloga, mnogo vremena i živaca je bilo utrošeno na kraju samog projekta.

U kontrolerima je bilo jednostavno izvršiti izmjene poput preimenovanja koristeći se ekstenzijom i hotkeys “ctrl+r+r” poput “id” u “idIgraca” ili recimo “idPremiuma” i mnogih slicnih primjera u raznim metodama.

```
// GET: Igraci/Details/5
//[Authorize(Roles = "Admin, Premium")]
0 references
public async Task<IActionResult> Details(int? idIgraca)
{
    if (idIgraca == null)
    {
        return NotFound();
    }

    var igrac = await _context.Igrac
        .FirstOrDefaultAsync(m => m.ID == idIgraca);
    if (igrac == null)
    {
        return NotFound();
    }
    ViewBag.nazivi1 = new List<SelectListItem>();
    List<Igrac> igraci = _context.Igrac.ToList();
    List<Tim> timovi = _context.Tim.ToList();
    foreach (var u in igraci)
    {
        if (u.ID == idIgraca)
        {
            Tim t1 = timovi.Find(t => t.ID == u.TimID);
            ViewBag.nazivi1.Add(new SelectListItem() { Text = t1.ime, Value = t1.ID.ToString() });
        }
    }
    return View(igrac);
}
```

```
0 references
public async Task<IActionResult> Delete(int? idPremiuma)
{
    if (idPremiuma == null)
    {
        return NotFound();
    }

    var premium = await _context.Premium
        .FirstOrDefaultAsync(m => m.ID == idPremiuma);
    if (premium == null)
    {
        return NotFound();
    }

    return View(premium);
}
```

Method extraction je iskorišten na mnogim dijelovima i bio ponuđen “kao na tacni”.
Primjeri su sljedeći:

```
1 reference
private bool IgracExists(int idIgraca)
{
    return _context.Igrac.Any(e => e.ID == idIgraca);
}
```

```
1 reference
private bool IgracExists(int idTima)
{
    return _context.Igrac.Any(e => e.ID == idTima);
}
```

```
1 reference
private bool TimExists(int idTima)
{
    return _context.Tim.Any(e => e.ID == idTima);
}
```

Zatim sužavanje predugih metoda, obzirom da se ponekad duge metode ne mogu tek tako rastaviti na dijelove i izvršiti extraction, bili smo primorani ostaviti jednu dugu metodu ali smo iskoristili prednost smanjivanja (preko minusa sa lijeve strane) pojedinog dijela koda za koji smo bili uvjereni da je ispravno napisan te to izgleda ovako.

```
0 references
public async Task<IActionResult> ZavrsiUtakmicu (int idUtakmice)
{
    var utakmica =
            await _context.Utakmica.Where(m => m.ID == idUtakmice).ToListAsync();

    if (utakmica[0].statusUtakmice != "Zavrsena")
    {

        var tim1 =
            await _context.Tim.Where(m => m.ID == utakmica[0].idTima1).ToListAsync();
        var tim2 =
            await _context.Tim.Where(m => m.ID == utakmica[0].idTima2).ToListAsync();

        await _context.SaveChangesAsync();

        tim1[0].ID = utakmica[0].idTima1;
        tim2[0].ID = utakmica[0].idTima2;

        tim1[0].datiGolovi += utakmica[0].rezTim1;
        tim2[0].datiGolovi += utakmica[0].rezTim2;

        tim1[0].primljeniGolovi += utakmica[0].rezTim2;
        tim2[0].primljeniGolovi += utakmica[0].rezTim1;

        tim1[0].brojOdigranihUtakmica++;
        tim2[0].brojOdigranihUtakmica++;

        if (utakmica[0].rezTim1 > utakmica[0].rezTim2)...
        else if (utakmica[0].rezTim1 < utakmica[0].rezTim2)...
        else...
        utakmica[0].statusUtakmice = "Zavrsena";
        _context.Tim.Update(tim1[0]);
        await _context.SaveChangesAsync();
        _context.Tim.Update(tim2[0]);
        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }
```

Suvišni komentari su obrisani, međutim ostavljeni su oni značajni koji mogu uputiti programere u budućnosti ukoliko budu željeli nadograditi sistem, tako da su dodati slijedeći:

```
// POST: Utakmice/Edit/5
// To protect from overposting attacks, enable the specific properties you want to bind to, for
// more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
```

A obrisani su komentari poput onih koji su bili u komunikaciji programera samim sa sobom kako misli ne bi “odlutale”...

```
// GET: LiveStream
// Potrebno promijeniti Index2 u Index
//[Authorize(Roles = "Admin, Premium")]

// Bespotreban parametar ukoliko ne bude promjene
0 references
public IActionResult Index(int id)
{
    PomocnaLS info = new PomocnaLS();
    Utakmica trenutna = _context.Utakmica.ToList().Find(ls => ls.statusUtakmice == "U toku");
    info.Tim1 = _context.Tim.ToList().Find(t => t.ID == trenutna.idTima1).ime;
    info.Tim2 = _context.Tim.ToList().Find(t => t.ID == trenutna.idTima2).ime;
    info.Rezultat = trenutna.rezTim1 + " : " + trenutna.rezTim2;
    // ukoliko budemo stvarno htjeli mijenjati streamove, klasa LiveStream će imati svoj video
    // sad taj atribut ne postoji pa je zato zakomentarisan
    // info.FileName = _context.LiveStream.ToList().Find(ls => ls.ID == id).Filename;
    info.FileName = "UCsDpnPJARbH4E0JgWjSDaSA";
    return View(info);
}
```

Te je na istom dijelu koda izvršeno brisanje suvišnih parametara…

Obzirom da su bile korištene testne funkcije, sada su one zamijenjene pravim i nisu više u upotrebi (neke poput ovih) :

```
// testna funkcija za predefinisane podatke
0 references
public IActionResult Index2()
{
    PomocnaLS info = new PomocnaLS();
    info.Tim1 = "FK Željezničar";
    info.Tim2 = "FK Sarajevo";
    info.Rezultat = "1 : 0";
    // filename je ID kanala koji vrši livestreamanje na YouTube
    info.FileName = "UCOQMD-IluHFLXBRpAMUE-EA";
    return View(info);
}
```

Naravno radi preglednosti koda i pojednostavljivanja, na raznim mjestima su korišteni oneliner kao i foreach petlja…

```
    }
    ViewBag.Timovi = new List<SelectListItem>();
    List<Tim> timovi = _context.Tim.ToList();
    foreach (var p in timovi)
        ViewBag.Timovi.Add(new SelectListItem() { Text = p.ime, Value = p.ID.ToString() });
    return View(igrac);
}


    return View(await _context.Igrac.OrderByDescending(m => m.brojAsistencija).ToListAsync());


    var statusSort =
        await _context.Utakmica.OrderByDescending(m => m.statusUtakmice == "U toku")
            .ThenBy(m => m.statusUtakmice == "Zavrsena").ToListAsync();
```