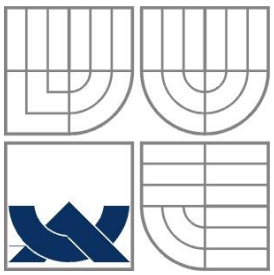
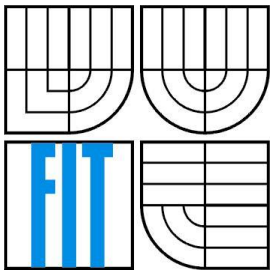


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

# INFORMAČNÍ SYSTÉM ČESKÉ DISCGOLFOVÉ ASOCIACE

INFORMATION SYSTEM OF CZECH DISCGOLF ASSOCIATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR
JAKUB KUDRNA

VEDOUCÍ PRÁCE
SUPERVISOR
Ing. Vladimír Bartík, Ph.D.

BRNO 2017

# Abstrakt

Cílem této práce je vytvoření webového informačního systému využívaného Českou Asociací Discgolfu, komunitou a hráči sportu jménem discgolf. Uživatel se může registrovat do systému a poté přidávat a následně spravovat hřiště s layouty jamek, turnaje, do kterých se registrují hráči, zadávání výsledků, kluby, nebo články stylem blogu. Dále systém počítá výsledky jednotlivých soutěží a lig, součástí systému je i webová mobilní aplikace umožňující uživateli zadávat výsledky turnaje živě. Výkonná rada asociace má vyšší práva, která jim dovolují správu veškerých dat z turnajů, článků i hřišť. Informační systém je implementován v prostředí ASP.NET s využitím databáze Microsoft SQL.



# Abstract

The main objective of my thesis is to create web information system used by the Czech Discgolf Association, it's community and discgolf players. Users can register themselves to the system and then add or edit discgolf courses, hole layouts, make new tournaments and submit scores to them. They can also manage clubs, see league results or add news to blog. There is also mobile web application used for submitting live scores of the tournaments. A specific group of users can have extended rights to editing courses, tournaments or leagues. The information system is implemented in ASP.NET with the use of Microsoft SQL database.



# Klíčová slova

Informační systém, ASP.NET, webová aplikace, discgolf, Microsoft SQL databáze

# Keywords

Information system, ASP.NET, web application, discgolf, Microsoft SQL database


# Citace

KUDRNA, Jakub. Informační systém České discgolfové asociace. Brno, 2017. 34 s. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Bartík Vladimír, Ing., Ph.D.

# Informační systém České discgolfové asociace

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Vladimíra Bartíka, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Jakub Kudrna
10.5.2017

## Poděkování

Za odbornou pomoc a mnoho dobrých rad bych chtěl poděkovat svému vedoucímu práce, Ing. Vladimíru Bartíkovi, Ph.D. Za trpělivost, podporu a věcné názory děkuji rodině, přítelkyni a hráčům discgolfu. Rád bych také poděkoval České asociaci discgolfu za možnost tvorby tohoto systému.

# Obsah

# 1 Úvod

Informační systémy tvoří velkou část webových aplikací a jsou v nějaké podobě využívané většinou uživatelů internetu. Podle kvality jejich zpracování můžeme usoudit, na jak vysoké úrovni daná firma nebo společnost je. To je dáno zpravidla tím, že lepší firma si může dovolit lepší webovou prezentaci, která často na první pohled případného zákazníka nadchne nebo naopak odradí.

Větší problém může nastat u neziskových organizací, jako jsou sportovní asociace nebo kluby. Ty většinou nemají dostatečné prostředky na zaplacení rozsáhlých a specifických informačních systémů, které by využívala celá jejich komunita, místo toho fungují na zastaralých systémech, nezvládající potřebnou funkčnost. Často se však najde nadšenec ochotný tento systém vyvinout za zlomek ceny, navíc se dobře vyzná v problematice dané organizace nebo sportu a s chutí komunitě pomůže. A to je přesně můj případ, jsem nadšený pro rozvíjející se sport jménem discgolf a tento nelehký úkol jsem vzal na sebe.

Přestože podobný projekt dělám poprvé, cílem mojí bakalářské práce je vyvinout informační systém České Asociace Discgolfu (dále ČADG), který bude zvládat všechny potřebné funkce od registrace hráčů, hřišť a turnajů, přes databázi výsledků a soutěží až po přidávání článků stylem blogu, spojování komunity hráčů a poskytování informací novým zájemcům o tento sport. ČADG sice má současný informační systém, ten je však velmi zastaralý a nepodporuje potřebnou funkčnost. Proto jsme si udělali se zástupci asociace dlouhé sezení, kde se probíralo, jak bude systém vypadat a co všechno bude umět. Aby všechno bylo ještě lepší, vymysleli jsme k systému rozšíření - zadávání výsledků turnajů živě z mobilních zařízení.

Z těchto požadavků vznikl návrh, který jsem se rozhodl implementovat v pro mě doposud neznámém prostředí ASP.NET, o kterém jsem si přečetl mnoho dobrých recenzí. V následujících kapitolách si rozebereme principy tvorby aplikací, představíme si framework ASP.NET, popíšeme požadavky na systém, jeho návrh a implementaci. Nakonec si ukážeme, jak probíhalo testování systému a zamyslíme se nad možnými rozšířeními.

# 2 Principy tvorby webových aplikací

Z technického pohledu je web programovatelné prostředí, které dovoluje mnoha uživatelům z celého světa upravovat jeho obsah pomocí velkého počtu různých webových stránek. Dvě důležité části moderního webu jsou webový prohlížeč a webová aplikace. Webový prohlížeč je software, který dovoluje uživateli získávat data a pracovat s nimi přes obsah nacházející se na jednotlivých webových stránkách a aplikacích.

## 2.1 Webová stránka a webová aplikace

Dnešní webová stránka už zdaleka není jen jednoduše graficky upravený holý text, jak tomu bývalo na konci 20. století. Moderní webové aplikace umožňují uživateli zobrazit dynamicky generovaný obsah přizpůsobený jeho potřebám, dovolují mu s ním pracovat, upravovat jej, přidávat nový. Dále webové aplikace umožňují spouštět různé skripty na straně klienta (webový prohlížeč), které ho přetvářejí na prostředí více připomínající program než webovou stránku (například Google Mapy).
Z toho vyplývá, že webová aplikace je počítačový program umožňující uživateli webu přidávat obsah, upravovat ho a pracovat s ním přes databázi pomocí preferovaného webového prohlížeče. Požadovaná data jsou poté dynamicky generována webovým serverem a zobrazována kdekoli po celém světě přes internet a webový prohlížeč [1].

Z technického hlediska webová aplikace posílá dotazy na webový server, který typicky pomocí databáze dynamicky vygeneruje požadovaný obsah stránky. Ten poté odešle klientovi (většinou webový prohlížeč, který ovládá uživatel). Obsah webové stránky je generován ve standardním formátu (jazyce) který webový prohlížeč umí zobrazit. Nejrozšířenější jazyk je HTML.

### 2.1.1 CSS

Aby byl obsah uživatelsky příjemný, musí být graficky zpracován. O to se stará programovací jazyk CSS (Cascading Style Sheets, neboli Kaskádové styly), který popisuje, jak budou HTML elementy zobrazeny na displeji, jakou budou mít například barvu a velikost. [2]

Když si webový prohlížeč stáhne HTML stránku ze serveru, aplikuje na ni její (pokud existují) CSS předpisy. Ty se skládají ze selektoru identifikujícího element, na který předpis použijeme a vlastnosti, jakou element bude mít. Vlastnost může být například červená barva textu, výška menu 50 pixelů a podobně.

Když webový prohlížeč zobrazuje obsah stránky, musí zkombinovat obsah (HTML) se styly (CSS). To se děje ve dvou krocích [3]:

1) Webový prohlížeč konvertuje HTML a CSS data na DOM (Document Object Model). DOM reprezentuje webovou stránku v paměti počítače. Ta poté zkombinuje obsah stránky s CSS předpisy do konečného DOM.
2) Webový prohlížeč zobrazí obsah konečného DOM.

Existují tři hlavní možnosti, jak CSS styly na dokument aplikovat.

1) Externí CSS soubor - předpisy jsou zvlášť sepsané v jednom nebo několika .css souborech. V HTML dokumentu je potom nutné se na tyto soubory odkázat elementem <link> a poté se aplikují pomocí selektorů na celou stránku.
2) Zápis stylů do elementu <style> - tyto předpisy jsou součástí HTML dokumentu a aplikují se na celou stránku pomocí selektorů

3) Přímý “inline” zápis - předpisy jsou aplikovány pouze na daný element, který obsahuje atribut style.

## 2.2 Mobilní aplikace

Mobilní aplikace je software, který uživatel může používat pomocí telefonu nebo jiného mobilního zařízení, například tabletu. Ze začátku se jednalo hlavně o jednoduché aplikace, které uživateli umožňovaly odeslat e-mail, podívat se na kalendář a podobně. V dnešní době však existují aplikace zvládající všemožné úkoly od sledování počasí přes hraní různých her až po GPS navigaci [4].

Mobilní aplikace vyžaduje specifický přístup při vývoji díky omezeným možnostem ovládání a výkonu zařízení. U mobilních zařízení musíme více dbát na optimalizaci výkonu, protože oproti počítači jsou vybaveny slabšími procesory a spotřebovávají baterii. Vývoj mobilní aplikace také vyžaduje specifický přístup k návrhu grafického prostředí a uživatelského rozhraní, protože v dnešní době je většina mobilních zařízení ovládána dotykem. Z toho plyne, že ovládací prvky bývají větší a jinak uspořádány než v klasické webové aplikaci pro stolní PC.

Mobilní aplikace dělíme na dva hlavní typy [5]:

1) Nativní mobilní aplikace - jsou vyvinuty specificky pro jeden operační systém (platformu) a jsou v zařízení nainstalovány podobně jako program na stolním PC. Jejich výhoda je využívání hardwarových dispozic telefonu jako GPS, fotoaparát a podobně, nevýhody jsou potřeba instalace a funkčnost pouze na jedné platformě.
2) Webové mobilní aplikace - jsou přístupné pomocí webového prohlížeče na každém mobilním zařízení, jde v podstatě webové stránky, které jsou uzpůsobeny pro mobilní zařízení a nabízejí funkcionalitu aplikací.

Součástí systému je mobilní aplikace pro zadávání výsledků živě v průběhu turnaje. Pro tuto potřebu jsem si zvolil webovou mobilní aplikaci z několika důvodů. Aplikace by měla být použitelná na co největším počtu různých zařízení bez ohledu na typ, platformu nebo aktuálnost systému. Navíc, jak si popíšeme v kapitole 5.1.3, bude aplikaci využívat velké spektrum uživatelů od juniorů po seniory, takže bude lepší se vyhnout instalaci, která by pro někoho mohla být složitá a pro rychlé použití na turnajích zdlouhavá. Aplikace také nemá potřebu využívat žádné hardwarové speciality zařízení, takže by nativní aplikace nepřinášela žádné výhody.

## 2.3 Možnosti vývoje

Pro vývoj webových stránek a informačních systémů můžeme využít velké množství nástrojů, programovacích jazyků, frameworků a podobně. Každý nástroj má svoje výhody a nevýhody, proto je potřeba zvážit, jaký bude pro naši práci nejvhodnější.

Mezi hlavní kandidáty pro vývoj systému pro mě patřily následující nástroje a programovací jazyky:

1) Skriptovací jazyk PHP spojený s HTML a CSS. Tuto možnost využívá většina dnešních webů. Skripty jsou uloženy spolu s HTML kódem v souborech s koncovkou .php a spouštěny na straně serveru s nainstalovaným PHP [6].
2) Framework ASP.NET vyvíjen Microsoftem [7], který jsem si pro informační systém zvolil a rozepíši se o něm v následující kapitole.

ASP.NET jsem si zvolil z několika důvodů. Splňuje všechny požadavky, které na tvorbu systému jsou, je velmi dobře dokumentovaný a poměrně rychle se s ním dá seznámit. Navíc se mi líbí prostředí, ve kterém se systém vyvíjí a jazyk C#, ve kterém je naprogramovaná funkcionalita je velmi rozšířený v praxi, kde ho mohu později uplatnit.

# 3 Prostředí ASP.NET

ASP.NET je zdarma dostupný framework určený k tvorbě rozsáhlých webových stránek a aplikací používající HTML, CSS, JavaScript a další. Je vyvíjen společností Microsoft a rozsáhle využíván po celém světě. Je odvozen o starší technologie ASP a je součástí frameworku .NET [7].

## 3.1 Základní principy ASP.NET

ASP.NET je založen na CLR (Common Language Runtime) stejně jako ostatní aplikace postaveny na .NET Frameworku, což umožňuje programovat v různých jazycích podporujících CLR. Na rozdíl od skriptovacích jazyků jsou aplikace rychlejší, protože jsou předkompilovány do .dll souborů, takže se při každém spuštění nemusí znovu a znovu kompilovat.

V ASP.NET máme 3 frameworky pro vytváření webových aplikací. Všechny jsou velmi propracované a každý z nich přináší svoje výhody i nevýhody:

1) Web Forms - framework pro tvoření rozsáhlých aplikací obsahující dynamicky generované stránky.
2) MVC (Model-view-controller) - oproti Web Forms přináší větší kontrolu nad HTML, funkcionalita je více separovaná od vzhledu, aplikace se dají snadněji testovat.
3) Web Pages - umožňuje jednoduché a nenáročné prostředí pro méně rozsáhlé webové aplikace, HTML a funkcionalita není rozdělená do více souborů.

## 3.2 ASP.NET Web Forms

Pro moji práci jsem si vybral framework ASP.NET Web Forms, protože se mi zdál pro moje potřeby nejvhodnější. Web Forms používá HTML spojené s ovládacími prvky (ASP.NET Web Controls) jako například tlačítka (Button) nebo pokročilé tabulky (GridView). Oproti HTML mají výhodu v tom, že k nim můžeme přiřazovat nejrůznější vlastnosti, funkce, zachytávat na nich události a podobně. Ovládací prvky produkují na straně serveru HTML kód obsahující potřebné vlastnosti, který poté tvoří jednotlivé části zobrazené stránky.

Každá webová stránka se skládá ze dvou částí:

1) Soubor .aspx obsahující HTML kód obohacený výše zmíněnými ovládacími prvky a případně skripty (například JavaScript), který definuje vzhled a statický obsah stránky.
2) Soubor definující funkcionalitu prvků stránky obsahující třídy, funkce a příkazy, které se spustí při určitých událostech na stránce (například kliknutí na tlačítko, samotné načtení stránky a podobně). V mé práci je použit jazyk C# s koncovkou .cs, další možností je programovací jazyk VB.NET.

## 3.3 Prostředí Visual Studio

Pro vytváření aplikací v ASP.NET se používá volně dostupný program Visual Studio vydaný Microsoftem, který obsahuje mnoho užitečných funkcí a dělá vývoj příjemnějším. Dovoluje také

spouštět vyvíjenou aplikaci a v reálném čase ji upravovat nebo použít debug mód (odstraňování chyb). Zároveň slouží jako skvělé vývojové prostředí s pomocnými prvky, které programátora upozorní na syntaktické chyby v kódu, dovoluje automatické formátování pro tvorbu přehledného kódu a podobně. Dále nám Visual Studio umožňuje vytvářet databáze a pracovat s nimi na vysoké úrovni.

Velkou výhodou používání Visual Studia je funkce *IntelliSense*. Ta ulehčuje psaní C# kódu různými způsoby [8]:

1) Předvídá jaké funkce, metody, nebo proměnné chce programátor využít a nabízí je ve formě automatického doplňování při kliknutí na pomocný text.
2) Detekuje syntaktické chyby v kódu, které označí a nabízí možná správná řešení.
3) Při použití funkce napovídá, jaké parametry očekává a v jakém formátu.
4) Při běhu programu může zobrazit aktuální hodnoty proměnných, obsahy tabulek a dalších tříd.

## 3.4 ASP.NET Web Controls

ASP.NET Web Controls jsou objekty ASP.NET používané na webových stránkách, které se zobrazí při načítání stránky webovým prohlížečem, nebo umožňují interakci s uživatelem. Mnoho Web Controls je velmi podobné klasickým HTML elementům jako tlačítka nebo textová pole, další ovládací prvky nabízí komplexní funkcionalitu jako například kalendář, nebo objekty připojující zdroj dat k tabulce a jejich zobrazení [9].

Visual Studio umožňuje Web Controls přidávat do stránky buď jako psaný kód, nebo má režim zvaný *design*, kde existuje pro přidávání prvků uživatelské rozhraní. To ovládáme myší a můžeme prvky na stránku přidat ze seznamu zvaného *toolbox* a poté si v několika krocích nastavit jeho funkčnost. Oproti psaní kódu je tohle řešení jednodušší pro začátečníky, ale já jsem je nepoužíval, protože automaticky generuje méně přehledný kód a radši jsem funkčnost programoval ručně a všechno měl pod kontrolou.

Teď uvedu několik prvků ASP.NET Web Controls, které jsem často používal pro můj informační systém i s příklady použití, protože se o nich budu ve zbytku práce zmiňovat a je důležité, abychom znali jejich funkcionalitu:

1) Button - funguje podobně jako HTML element *button*, plní funkci tlačítka. Když uživatel na button klikne myší, spustí se část kódu v podobě funkce na serveru. Funkce typicky obsahuje přesměrování na jinou stránku, uložení dat nebo potvrzení přihlášení.
2) TextBox - slouží pro psaní textu do pole na webové stránce. Můžeme zvolit několik možností pro formát vkládaného textu, například *MultiLine* pro rozsáhlejší text, případně rozšíření na textový editor, *Password* pro vkládání hesla a zobrazení teček místo jednotlivých znaků pro bezpečnost. Implicitně je nastaven na *SingleLine*, což je vkládání na jeden řádek.
3) CheckBox - jednoduché zaškrtávací políčko, kterým uživatel stanoví dvě možnosti, většinou ano/ne (true/false). Může se použít např. pro sdělení systému, zda si po přihlášení bude pamatovat uživatele.
4) DropDownList - rozbalovací seznam, který je hojně používán při vyplňování formulářů. Ulehčuje práci uživateli a zároveň se stará o to, že do databáze budou uložena korektní data. Jeden seznam může mít několik položek, každá položka má svoji hodnotu (*value*) a text (*text*).
5) Label - zobrazuje programem nastavený text, který můžeme dynamicky měnit.

6) GridView - zobrazuje tabulku s hlavičkou a hodnotami předanými z databáze, případně vygenerované tabulky podobající se tabulce v SQL databázi. Obsah GridView můžeme dynamicky měnit (přidávat a odebírat řády tabulky). V našem systému bude hojně využíván, protože potřebujeme zobrazit seznamy pro různé účely (hráči, turnaje a další). GridView je podrobněji vysvětlen v kapitole 6.2.1.
7) SqlDataSource - předává data z SQL databáze některým prvkům Web Controls, například GridView. Musíme určit, z jaké databáze se budou data stahovat a zadat patřičné SQL příkazy.
8) Repeater - je to seznam několika po sobě jdoucích útvarů obsahující různá data, které si sami definujeme. Můžeme ho využít například při zobrazení výčtu náhledů článků v blogu (náhled článku obsahuje fotografii, nadpis a část textu článku). Náhled každého existujícího článku se poté zobrazí na jedné stránce ve formě seznamu, nebo tabulky.

# 4 Analýza požadavků

V následující části se budeme věnovat požadavkům na informační systém České Asociace Discgolfu (ČADG). Protože je discgolf poměrně neznámý sport, tak jej krátce představím. Informační systém ČADG existuje, ale je velmi zastaralý a má hodně nedostatků, které je potřeba vylepšit, proto si musíme shrnout všechny požadavky na nový systém a podle nich navrhnout diagram případů užití, ER diagram databáze i celý výsledný systém.

## 4.1 Discgolf a Česká Asociace Discgolfu

Discgolf je nenáročný sport, který má pravidla podobná golfu s tím rozdílem, že místo míčku a hole se hází speciálními disky (létající talíře podobné frisbee). Vznikl v 70. letech v USA a nyní se stává velmi populárním i ve zbytku světa, hlavně v Evropě. Cílem hráče je na co nejmenší počet hodů dostat disk z výhoziště do koše. Tento časově i finančně nenáročný sport může hrát téměř kdokoliv od dětí po důchodce. Discgolf se v posledních letech v České Republice hodně rozmáhá, za posledních 5 let tu vyrostlo přes 60 pevných hřišť včetně dvou v Brně. Hřiště má typicky 9 až 18 jamek, které jsou postaveny v parku, lese, nebo na louce.

Česká asociace discgolfu (dále ČADG) je občanské sdružení které vznikalo v průběhu roku 2011 s cílem vytvořit samostatnou organizaci sdružující hráče discgolfu v České republice, hájit jejich zájmy, organizovat Českou discgolfovou ligu a propagovat discgolf u široké veřejnosti. V současné době má ČADG zhruba 300 aktivních členů a další stovky hráčů hrají rekreačně. Jen v loňském roce proběhlo cca 150 turnajů po celé republice [10].

Hráči soutěží ve dvou hlavních ligách:

1) 1.liga - hrají ji nejlepší hráči, skládá se ze dvanácti dvoudenních turnajů hraných na hřišti, které má minimálně 18 jamek a kapacitu 90 hráčů.
2) 2.liga - je pro slabší hráče, kteří v ní bojují o právo registrace do turnajů 1.ligy na další sezónu.

## 4.2 Nedostatky současného IS

Současný informační systém ČADG fungující na webu [10] je již velmi zastaralý a nevyhovuje současným potřebám ČADG, hráčům ani ředitelům turnaje. Velký nedostatek je také grafický vzhled webu, protože cílem ČADG je také přilákat nové hráče, které by měl web zaujmout nejen obsahem, ale i moderním vzhledem. Za cca 10 let, po které web funguje, se změnily standardy na vzhled i uživatelské rozhraní potřebující aktualizaci.

Další nedostatek současného systému pociťují ředitelé turnajů, kteří zakládají turnaj a potom zadávají výsledky. Uživatelské rozhraní pro správu turnaje je totiž nepromyšlené a zbytečně složité. Jako příklad uvedu nutnost vyplňování informací o turnaji v textové podobě. V současném systému je potřeba při vkládání informací o turnaji použít HTML kód, který většina uživatelů dobře neumí. Jako alternativa by byl vhodný WYSIWYG editor ulehčující zadávání textu. Dále můžu jako příklad uvést vkládání data pomocí seznamu nebo kalendáře, které by nahradilo nynější vkládání jen pomocí textu, kdy pokud datum zadáte špatně, systém vás upozorní, až odešlete celý formulář a datum musíte opravit.

## 4.3 Požadavky na systém

Nový informační systém by měl oproti současnému umožňovat jednodušší ovládání, příjemnější uživatelské rozhraní, modernější vzhled a přinášet nové funkce. Nyní si popíšeme požadavky na jednotlivé části systému, podle kterých poté vznikne návrh.

### 4.3.1 Hlavní strana a informace o ČADG a discgolfu

Při vstupu na hlavní stránku informačního systému by měly být viditelné novinky a články vydané ČADG seřazené podle data vydání. První položky menu by potom měly obsahovat informace o ČADG, dokumenty (soutěžní řád, stanovy asociace), partnery a kontakt na asociaci.

Další položka menu by měla obsahovat informace o discgolfu, co je to discgolf, jak se hraje, jaká jsou pravidla, kde můžeme koupit disky a podobné informace pro začínající hráče.

### 4.3.2 Uživatelé, hráči, členové asociace

Pokud uživatel navštíví informační systém, může se zaregistrovat. Tím se mu vytvoří profil, kde si o sobě může vyplnit informace, přiřadí se mu jednoznačné číslo a může hrát turnaje jako klasický hráč. Pokud hráč zaplatí poplatek asociaci, stává se z něj člen ČADG, který získává možnost přednostní registrace na ligové turnaje, počítají se mu body do první nebo druhé ligy ČADG.

Pokud se člen umístí na předních místech v lize, může se z něj stát TOP hráč, který má ještě dřívější možnost na registraci do prvoligových turnajů. Ztrácí tím ale možnost získávat body ve druhé lize ČADG.

### 4.3.3 Hřiště

Hřiště jsou důležitou součástí discgolfu. Informační systém by měl obsahovat databázi hřišť, které můžou přidávat přihlášení uživatelé. Přidání hřiště vyžaduje vyplnit název hřiště, adresu, počet jamek, délku hřiště, jeho par, typ, charakter a další informace o hřišti. Ke každému hřišti také existují layouty (rozestavění jamek na hřišti), které můžou přidávat ředitelé turnajů pro svoje turnaje. Layout má určitý počet jamek, každá jamka má svoji délku, par a převýšení.

Seznam hřišť by měl mít filtr, podle kterého si uživatel zvolí, jaké hřiště si chce zobrazit. Filtrovat by se mělo podle kraje, délky nebo počtu jamek. Po kliknutí na hřiště se zobrazí jeho detail s informacemi o hřišti a jeho layouty.

### 4.3.4 Turnaje

Správa turnajů je jednou z hlavních částí informačního systému. Ředitel turnaje může přidat turnaj, ke kterému vyplní řadu informací jako název turnaje, datum, kontakt na ředitele, kapacitu hráčů nebo propozice v textové podobě.

Dále musí ředitel vybrat jednu ze tří turnajových kategorií ČADG:

1) Kategorie A je nejvyšší, tyto turnaje se řadí do první discgolfové ligy a jsou za nejvíce bodů a nejkvalitnější. Datum registrace hráčů do turnaje je pevně dáno a vysvětleno níže.
2) Turnaje kategorie B jsou také zařazeny do první ligy, ovšem jsou za méně bodů a méně prestižní než A turnaje. Datum registrace hráčů je také pevně dáno.
3) Turnaje kategorie C jsou jednodenní turnaje, které tvoří druhou discgolfovou ligu. Datum registrace hráčů je opět pevně dáno.

4) Neligové turnaje jsou takové, které nespadají pod ČADG. Ředitel turnaje může zaregistrovat neligový turnaj, který se nepočítá do žádné z lig ČADG. V tomto případě si ředitel sám musí určit datum začátku i konce registrace.

Turnaj také může být registrován pod PDGA (Profesionální discgolfová asociace, která je celosvětová) v kategorií A, B nebo C, které pro informační systém nic zásadního neznamenají. Dále ředitel vyplňuje hřiště a jednotlivá turnajová kola. Každý turnaj se hraje na určitý počet kol, běžně 3 kola + finále. Každé kolo musí mít svůj layout. Pokud potřebný layout neexistuje, je nutné ho přidat.

Každý turnaj musí mít otevřené hráčské kategorie. Ředitel potřebné kategorie může zpřístupnit a ke každé z nich přidá cenu startovného, která se může pro jednotlivé kategorie lišit. Umístění se potom počítá zvlášť pro každou hráčskou kategorii, kterých máme 6:

1) MPO - Open: nejprestižnější kategorie bez limitů
2) FPO - Open women: ženy bez věkového limitu
3) MPM - Master: muži ve věku 40 let a více
4) MPG - Grandmaster: muži ve věku 50 let a více
5) MJ1 - Junior 18-: junioři do maximálního věku 18 let
6) MJ3 - Junior 12-: junioři do maximálního věku 12 let

Na každý turnaj se mohou registrovat sami hráči. Registrace se otevírá v různé časy podle kategorie turnaje. Například u A turnajů se 8 týdnů před začátkem turnaje podle soutěžního řádu ČADG otevírá registrační fáze pro TOP hráče, 6 týdnů před začátkem turnaje pro členy ČADG a 4 týdny před začátkem pro všechny hráče.

Ředitel turnaje může zadávat výsledky hráčů. Každý hráč odehraje kolo turnaje na určitý počet hodů, výsledky se nakonec sečtou a vypočítá se pořadí. Pokud jde o ligový turnaj, dopočítají se hráčům body.

### 4.3.5 Soutěže, ligy a výsledky

Jak je napsáno výše, ČADG má dvě ligy. První liga se skládá ze všech turnajů ČADG kategorie A a B. Za turnaj kategorie A získá vítěz 150 bodů, za turnaj kategorie B 100 bodů. Se zvyšujícím umístěním se body snižují podle tabulky v soutěžním řádu. Do finálního bodování první ligy se počítá součet 5 bodově nejlepších výsledků. Umístění se potom počítá zvlášť pro každou hráčskou kategorii.

Druhá liga ČADG se skládá ze všech turnajů kategorie C. Vítěz turnaje získá 50 bodů a stejně jako v první lize se body snižují podle soutěžního řádu. Do finálního bodování druhé ligy se počítá součet 10 bodově nejlepších výsledků. Umístění se opět počítají pro každou hráčskou kategorii zvlášť.

Kromě dvou hlavních lig existují i další soutěže, které nespadají pro ČADG, ale pořádají je sami hráči. Jedná se většinou o lokální soutěže nebo ligy, kde se počítají body stejným stylem jako u druhé ligy s tím rozdílem, že počet nejlepších započítaných výsledků je proměnný. Každý hráč může vytvořit svoji vlastní soutěž a přidávat do ní turnaje, ze kterých se hráčům budou počítat body. Součet nejlepších výsledků potom udává ve všech otevřených kategoriích výsledek soutěže.

Nové ligy a soutěže se otevírají každý rok k 1. lednu. Proto je potřeba vést archív proběhlých soutěží, kde si uživatel může svolit rok, typ soutěže nebo ligy a podívat se na výsledky.

### 4.3.6 Profil hráče

V informačním systému musí existovat stránka s přehledem všech hráčů. Pokud si uživatel otevře profil hráče, měly by se mu zobrazit informace o daném hráči. Základní informace jako jméno, příjmení, rok narození, PDGA číslo a další si hráč vyplní sám při registraci a některé (přezdívka, textový popis hráče) si může kdykoliv upravit.

Hráči se automaticky na profil přidávají informace o odehraných turnajích a jeho umístěních. Dále by se měly zobrazit informace o tom, na jakých turnajích je hráč zaregistrován, ale ještě se neodehrály. Pokud je hráč členem ČADG, měl by se to uživatel dozvědět zároveň s informací, kdy mu končí členství. Každý hráč může také být členem klubu, jehož název by měl na hráčově profilu být uveden.

### 4.3.7 Kluby

Hráči mohou být členové jednotlivých klubů. Klub je většinou lokální organizace, která sdružuje hráče z jednoho města nebo okolí. Klub může mít neomezený počet členů, avšak hráč může být součástí pouze jednoho klubu.

Klub má jednoho ředitele, který může do klubu přidávat hráče a spravovat informace o klubu. Klub by měl mít název, rok založení, místo, ve kterém sídlí (většinou město) a kontakt na ředitele. Seznam klubů by měl být dostupný z menu a po otevření jednotlivých klubů by každý z nich měl mít svůj profil obsahující potřebné informace stejně jako hráči.

### 4.3.8 Články, blog

Uživatelé by měli mít možnost v informačním systému přidávat vlastní články do několika rubrik (turnaje, návody a tipy, ostatní, oznámení ČADG). Oznámení ČADG mohou přidávat pouze uživatelé s oprávněním (výkonná rada ČADG nebo administrátor, o uživatelských rolích je více v kapitole 4.4), protože obsahují důležité informace a jsou viditelná na rozdíl od ostatních článků na domovské stránce systému.

Pokud uživatel chce přidat článek, musí se prvně přihlásit, potom si otevřít formulář pro přidávání článku, vyplnit název článku, textový obsah, zvolit správnou rubriku a pokud má zájem, přidat ke článku fotografii nebo obrázek. Seznam všech článků, které si budou moci uživatelé zobrazit by měl být dostupný z menu pod položkou “Blog”. U seznamu článků je filtr, kterým si uživatel může vybrat články pouze ze zvolené rubriky.

### 4.3.9 Zadávání výsledků živě z mobilní aplikace

Součástí systému je mobilní aplikace, která umožňuje uživatelům zadávat skóre jednotlivých hráčů v průběhu turnaje (dále live scoring).

Každý turnaj bude možné zpřístupnit pro zadávání online ředitelem turnaje. Ten zadá pro turnaj live scoringu heslo, které poté sdělí všem hráčům na turnaji, kteří pomocí aplikace budou schopni zadávat skóre online.

Uživatel si prvně zvolí turnaj, zadá správné heslo, poté si vybere, pro které hráče bude skóre zadávat, zvolí turnajové kolo, začínající jamku a poté vyplňuje skóre po jednotlivých jamkách.

## 4.4 Uživatelské role

Pro informační systém musíme roztřídit jednotlivé uživatelské role, protože pokud bychom všem uživatelům přidělili stejná práva, systém by byl lehce zneužitelný.

### 4.4.1 Nepřihlášený uživatel

Pokud do informačního systému vstoupí nepřihlášený uživatel, má možnost se buď zaregistrovat, nebo se přihlásit pod svůj účet pokud už ho má vytvořený. Pokud se nepřihlásí, může si zobrazit veškeré informace o ČADG, discgolfu, podívat se na seznam hřišť a otevřít je, uvidí vše ohledně turnajů, výsledků, klubů a také všechny články včetně upozornění ČADG.

Pokud nepřihlášený uživatel zná heslo turnaje, může zadat výsledky i použít live scoring. Je to z důvodu, kdyby kdokoliv chtěl pomoci řediteli turnaje zadat výsledky a ulehčit mu práci, resp. zkrátit čekání hráčů na hotové výsledky. Tento případ popisuje obrázek 4.5.1 pomocí Use case diagramu.

### 4.4.2 Přihlášený uživatel

Přihlášený uživatel může dělat všechno, co nepřihlášený plus mnoho dalších činností. Mezi ně patří:

1) Přidávání a správa hřišť - pokud přihlášený uživatel přidá nové hřiště, zároveň se stává jeho správcem a může je upravovat.
2) Přidávání layoutů - přihlášený uživatel může přidávat ke hřištím nové layouty, protože pokud přidá turnaj a potřebuje zadat kolo s vlastním layoutem, nebylo by to možné.
3) Přidávání a správa turnajů - uživatel může přidat turnaj a spravovat ho. Pokud je to v jeho zájmu, může předat heslo turnaje dalším uživatelům, kteří za něj přidávají výsledky turnaje.
4) Přidávání a správa soutěží - podobně jako u hřišť může přihlášený uživatel přidávat vlastní soutěže a spravovat ty, které přidal.
5) Používání blogu - přihlášený uživatel může přidávat články, upravovat jím přidané články, nebo je mazat.

Tento případ popisuje obrázek 4.5.2 pomocí Use case diagramu.

### 4.4.3 Výkonná rada

ČADG má sedmičlennou výkonnou radu, která se stará o bezproblémový chod asociace. Členové výkonné rady mají možnost dělat všechno co přihlášený uživatel s tím rozdílem, že mají rozšířená práva, která jim dovolují upravovat všechna hřiště včetně těch, které přidal jiný uživatel, stejně jako všechny turnaje, layouty, soutěže a články v blogu. Zároveň mají možnost přidávat články do rubriky "Oznámení ČADG", které se zobrazují na hlavní stránce informačního systému.

Výkonná rada má dále za úkol spravovat členy asociace. To znamená, že musí mít přístup k administrační části systému, která dovoluje následující změny:

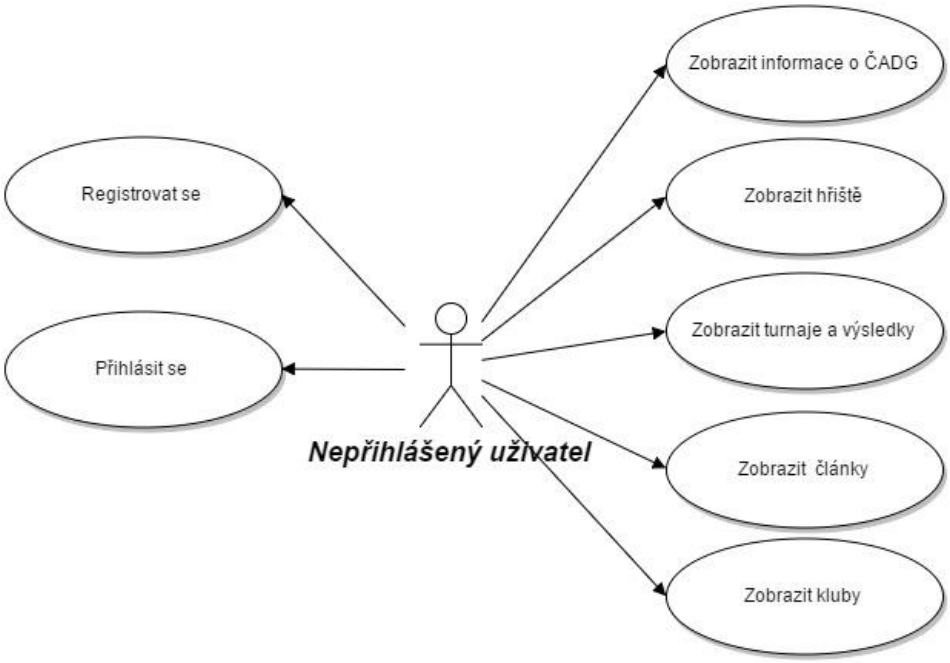
1) Úprava práv uživatelů - členové výkonné rady se mohou postupně měnit, proto je potřeba, aby systém byl schopen měnit práva uživatelů. Pokud se výkonná rada změní, členové mohou přenést svá práva na jiné uživatele.
2) Správa členů ČADG - členové výkonné rady mají za úkol měnit členství jednotlivých uživatelů. Pokud uživatel zaplatí členství, výkonná rada změní datum konce členství uživatele na odpovídající datum.
3) Určování TOP hráčů - jak bylo napsáno výše, TOP hráči jsou výběr nejlepších hráčů uplynulé sezóny. Výkonná rada je musí určovat, informační systém tedy nabízí možnost u každého hráče určit, zda je nebo není TOP hráč.

### 4.4.4 Administrátor

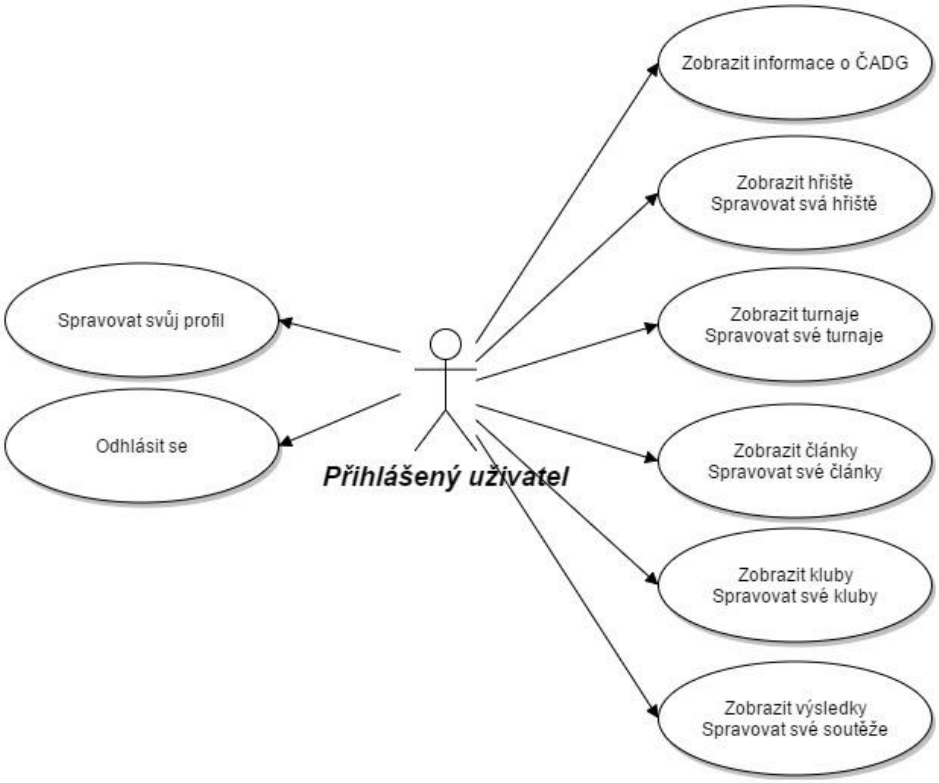
Administrátor má neomezená práva, v podstatě podobná jako člen výkonné rady s tím rozdílem, že má přístup k databázi pro případ, že by se někde stala chyba v zápisu dat a podobně. Musí to být člověk, který se vyzná v databázi a programování, aby byl schopný opravit chyby v systému. Při spuštění systému do reálného provozu to budu já, protože systém znám a v případě potřeby není problém seznámit se systémem někoho jiného. Tento případ společně s výkonnou radou popisuje obrázek 4.5.3 pomocí Use case diagramu.

## 4.5 Diagram případů užití

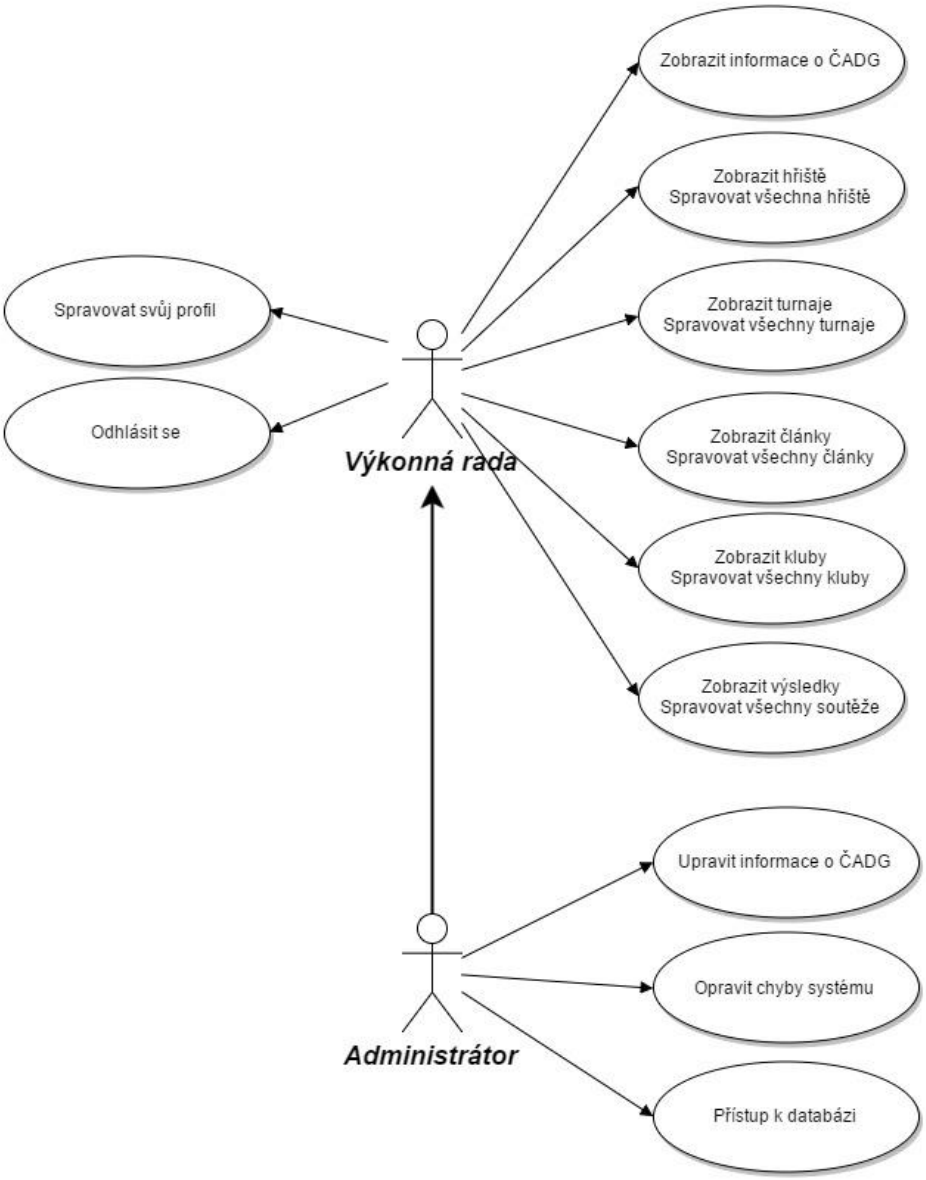
Diagram případů užití (Use case diagram) nám ukazuje všechny možné interakce různých typů uživatelů se systémem, jak je popsáno v kapitole 4.4.




*Obrázek 4.5.1 - Use case diagram nepřihlášeného uživatele*




*Obrázek 4.5.2 - Use case diagram přihlášeného uživatele*




*Obrázek 4.5.3 - Use case diagram výkonné rady a administrátora*

# 5 Návrh informačního systému

V předchozí kapitole jsme si řekli, co všechno by měl informační systém obsahovat. Nyní je na čase všechny požadavky zformovat a upřesnit si, co budou obsahovat jednotlivé webové stránky, navrhnout podle toho databázi a také uživatelské rozhraní.

## 5.1 Návrh jednotlivých částí

Každá stránka informačního systému by měla mít danou strukturu a různé prvky stránek by měly být tvořeny podle stejných konvencí a stylů. Tyto věci si rozebereme v následující kapitole.

### 5.1.1 Struktura stránky

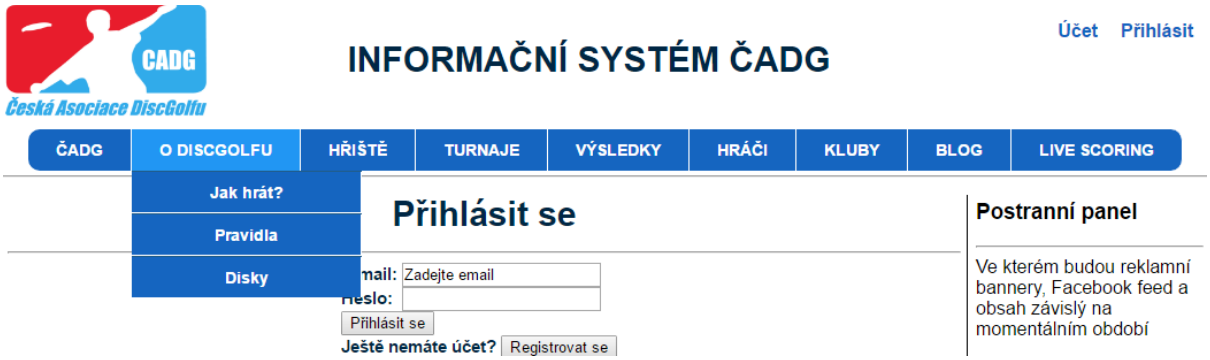
Obsah každé stránky je jiný, ale její struktura je stejná. Začneme hlavičkou (také nazývána header), která je podstatnou částí stránky. Hlavička obsahuje logo ČADG, které zároveň funguje jako odkaz a po kliknutí na ně nás přesměruje na hlavní (domovskou) stránku systému. Poslední dobou se tato funkce stává součástí většiny webů a navíc nám šetří jednu položku menu, které se potom stává přehlednějším.

Dále hlavička obsahuje uživatelské možnosti dovolující uživateli se přihlásit a jít na svůj účet. Pokud je uživatel přihlášen, má možnost se odhlásit. Pokud uživatel přihlášen není, má možnost se buď přihlásit, nebo se registrovat do systému a tím získat svůj účet, na který se bude přihlašovat pomocí e-mailu a hesla, které si zvolí při registraci. Mezi logem a přihlašovací částí je prostor vyplněný názvem systému, který je ale nachystán pro budoucí použití reklamních bannerů. Pod touto sekcí je v celé šířce stránky menu (o menu se rozepíšu níže).

Pod hlavičkou následuje obsahová část, která je rozdělena na dva sloupce. V levém, širším sloupci se nachází obsah stránky, který se pro každou stránku liší, pravý sloupec (RightSideBar) je užší a obsahuje aktuální informace, které spravuje administrátor, a načítají se z databáze. Spodní část každé stránky zakončuje patička (footer), která prozatím zůstává bez obsahu.

### 5.1.2 Menu

Menu neboli navigační lišta je součástí hlavičky každé stránky systému. Skládá se z jednotlivých položek, které odkazují na různé stránky systému. Základní položka menu může mít více dílčích kategorií, které se zobrazí v případě, že na ni umístíme kurzor (obrázek 5.1.1). Každá položka obsahuje jednoduchý odkaz (HTML element *link*) na cílovou stránku, která obsahuje menu ve stejné podobě.




*Obrázek 5.1.1 - hlavička a položky menu*

### 5.1.3 Uživatelské rozhraní

Návrh uživatelského rozhraní je důležitá část práce, nad kterou je potřeba se zamyslet. Musíme se soustředit na to, jaká skupina uživatelů bude systém využívat a jaké mají zkušenosti s používáním internetu a počítačových technologií. Největší počet hráčů je v kategorii Open (18-40 let) a Junior 1 (12-18 let), pro které můžeme předpokládat, že práci s internetem a PC ovládají. Existuje ovšem čím dál více hráčů v kategoriích Master (40-50 let) a Grandmaster (50 a více let), u kterých je potřeba počítat s tím, že s novými technologiemi nejsou dobře seznámeni.

Proto je potřeba uživatelské rozhraní navrhnout tak, aby bylo jednoduché, přehledné a intuitivní. Nyní si popíšeme jednotlivé části návrhu uživatelského rozhraní, které výše zmíněné problémy řeší:

1) Elementy, na které se dá kliknout - například tlačítka, textové odkazy, nebo položky menu mění barvu, pokud na něj umístíme kurzor. Tím si uživatel může všimnout, že s daným elementem může interagovat a kliknutím na něj spustí nějakou akci.
2) Zpětná vazba - pokud uživatel dokončí nějakou akci (např. přidávání hřiště nebo turnaje) a ta úspěšně proběhne, systém informuje uživatele pomocí modálního dialogového okna (alert v JavaScript). Tím docílíme toho, že uživatel je informován o stavu jeho akce a dále nemusí přemýšlet nad tím, jestli všechno co udělal, proběhlo správně. Poté následuje přesměrování na vhodnou stránku, např. pokud uživatel přidá turnaj a potvrdí dialogové okno, bude přesměrován právě na stránku jeho turnaje, protože můžeme předpokládat, že ji bude chtít zkontrolovat.
3) Tabulky - informační systém obsahuje velké množství tabulek (většinou GridView), které je potřeba navrhnout přehledně. Hlavička tabulky by měla být výrazná a jasně oddělená od řádků s daty, které jsou graficky rozdělené na sudé a liché, které se liší barvou pozadí. Tento způsob rozdělení řádků napomáhá orientaci ve čtení a uživatel potom nemá tendenci přeskakovat řádky při čtení. Další použitá metoda pro jednodušší čtení řádků je změna barvy pozadí po přejetí kurzoru.
4) Filtrování informací - tabulky s mnoha řádky mohou být nepřehledné, proto většina z nich dovoluje filtrovat jejich obsah. Pokud například uživatel hledá hřiště, může si filtrovat pouze ta hřiště, která mají 18 jamek, nebo jsou v jihomoravském kraji.

### 5.1.4 Kontrola vstupních dat

Pro správnou funkčnost informačního systému je nutné kontrolovat, zda jsou všechna data vložená uživatelem ve správném formátu. Tím zajistíme správnou funkčnost systému a předejdeme chybám,

které by mohly vzniknout špatným formátem textu uloženého v databázi. Navíc sloupce v tabulce databáze vyžadují určité datové typy, které by jinak nešly uložit a vyvolávaly by chyby.

Proto před každým ukládáním dat do databáze musíme provést kontrolu vstupních dat. Pokud uživatel ukládá text, musíme se zaměřit hlavně na jeho povolenou délku, případně správný tvar. Například název hřiště nebo turnaje musí mít minimálně 3 znaky, aby název vůbec dával smysl, a předejdeme zároveň situaci, kdy uživatel nechtěně daný text nevyplní.

U čísel musíme kontrolovat jeho hodnotu a formát. Například pro kapacitu hráčů na turnaj víme, že to musí být celé číslo mezi 1-200. Proto musíme zkontrolovat, zda je číslo celé (integer) a zároveň v požadovaném intervalu.

Tam, kde je málo možností, jak data zadat používáme rozbalovací seznam (DropDownList). Ten zajišťuje, že zadaná data budou správně vyplněna a zároveň ulehčuje práci uživateli, který vybere jednu z několika možností a tu označí. Tato možnost je využita například u zadávání informace o tom, zda je přidávané hřiště placené. Uživatel na má výběr pouze možnosti “Ano” nebo “Ne”, čímž se omezí problém nevhodně zadaných dat a uživatel má jednodušší práci i jistotu, že je vše v pořádku.

Pokud některá data nejsou zadána správně a algoritmus kontroly vstupních dát detekuje chybu, musí systém informovat uživatele a dát mu možnost chybu napravit bez ztráty dat formuláře. To zajišťuje funkce Alert v JavaScriptu, která spustí modální dialogové okno s krátkou informací, co uživatel zadal špatně. Zároveň se špatně zadaná data zabaví červeně, aby chybu mohl uživatel jednoduše najít. Když jsou poté data zadaná správně, uloží se do databáze.

### 5.1.5 Přihlášení a registrace

Důležitá část většiny informačních systémů je registrace, přihlášení a odhlášení uživatele, a práce s jeho účtem. Pokud již není registrován, má každý nepřihlášený návštěvník systému možnost registrace svého účtu, která probíhá následovně:

1) V hlavičce stránky uživatel klikne na “přihlásit”, nebo “účet”. Pokud již není přihlášený, systém ho přesměruje na přihlašovací stránku, kde je možnost kliknout na “Registrovat se”, která otevře registrační formulář.
2) V registračním formuláři uživatel musí vyplnit potřebné údaje pro registraci jako jméno, příjmení, rok narození, národnost, e-mail a další. Pokud jsou všechna data vyplněna správně a uživatel potvrdí registraci, vytvoří se mu nový účet, pod kterým se bude v budoucnu přihlašovat. Pokud kontrola vstupních dat detekuje chybu, musí ji uživatel opravit. Problematické části jsou neplatný e-mail, již zaregistrovaný e-mail, nebo nesprávně zadané heslo, které se musí zadávat dvakrát pro potvrzení jeho správnosti.

Pokud uživatel již účet má, může kliknout na přihlašovací stránku, potom zadat e-mail a odpovídající heslo. Pokud tyto informace zadal správně, je přihlášen do systému. Přihlášení funguje tak, že je uživateli přiřazena session (vysvětlena v bodě 6.2.2) s hodnotou uživatelova e-mailu. Pokud přihlášený uživatel vstoupí na stránku, kde je vyžadováno přihlášení, proběhne kontrola hodnoty této session a podle toho, zda má uživatel potřebná práva nebo ne se mu zobrazí příslušný obsah.

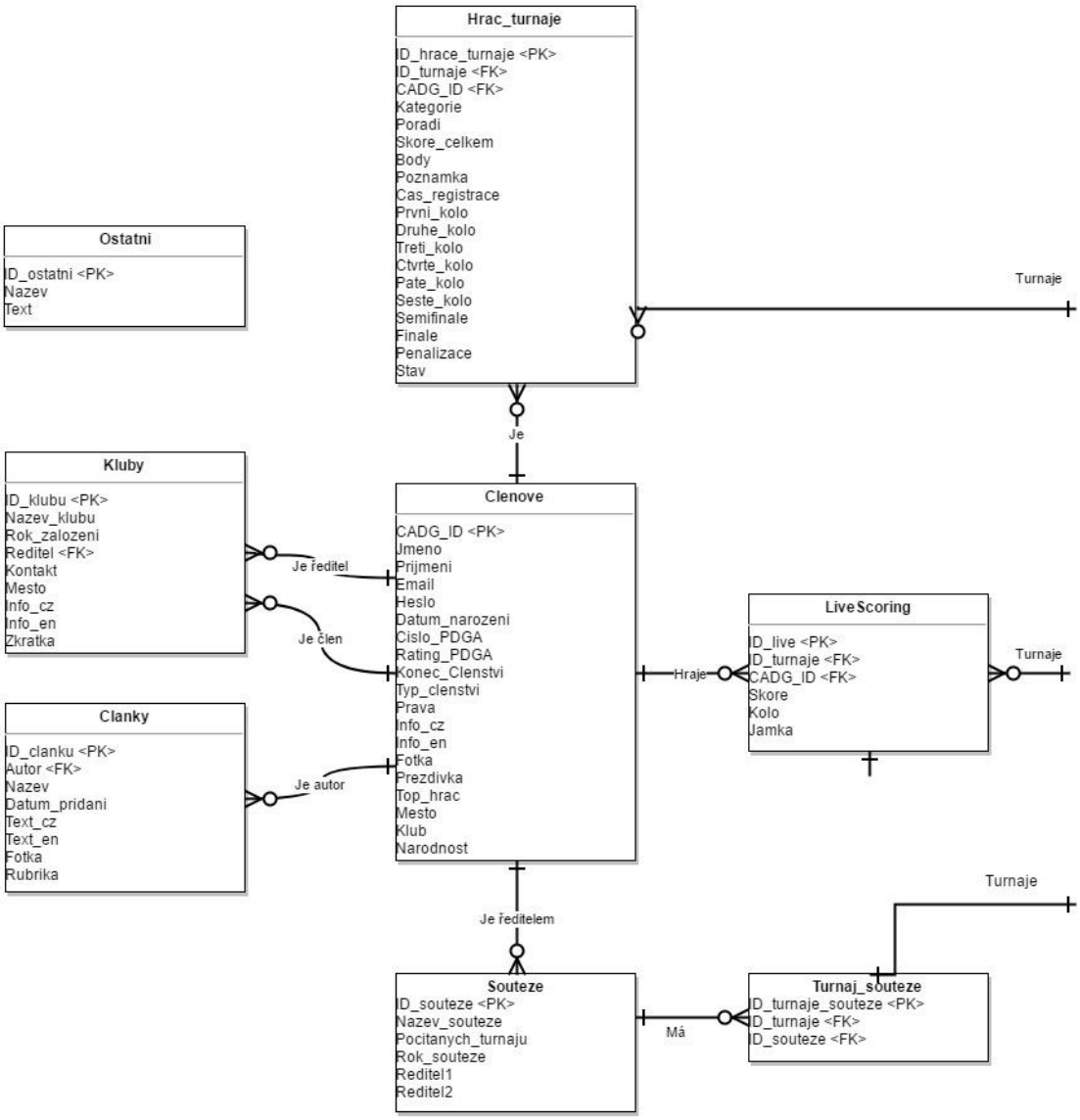
Jako příklad si můžeme uvést ovládací prvky, které se zobrazují jen přihlášeným uživatelům. Pokud si nepřihlášený uživatel prohlíží hřiště, nemá možnost přidat nové hřiště. Pokud se ale přihlásí, hned se mu zobrazí tlačítko “Přidat hřiště”. Pokud se dostane na stránku, kde přidává hřiště, musí proběhnout ještě jedna kontrola přihlášení z důvodu, že by si nepřihlášený uživatel zadal do URL webového prohlížeče odkaz na tuto stránku. V tom případě jej stránka ještě předtím, než se načte, přesměruje na přihlášení. Pokud se uživatel přihlásí, automaticky je přesměrován na původní stránku pro přidání hřiště pro příjemnější práci se systémem.

## 5.2 Návrh databáze

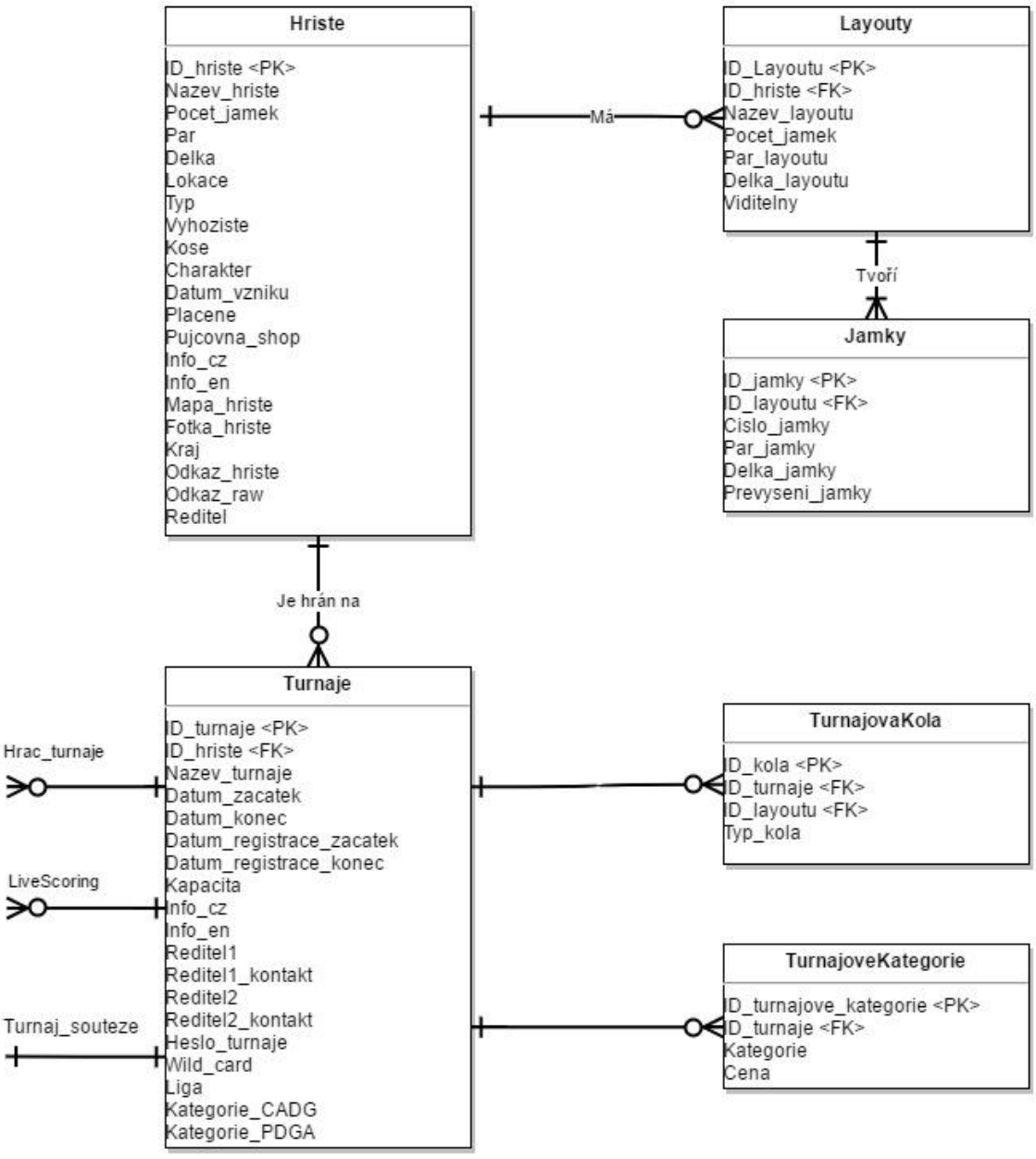
Návrh databáze je při tvorbě systému velmi důležitá část, kterou je dobré udělat co nejdříve a důkladně ji promyslet. Později se může stát, že najdeme v návrhu chybu, která se dá odstranit jednoduše, nebo i složitě, což poté vyžaduje velké a složité zásahy do systému. K popsání návrhu databáze slouží ER diagram.

### 5.2.1 ER diagram

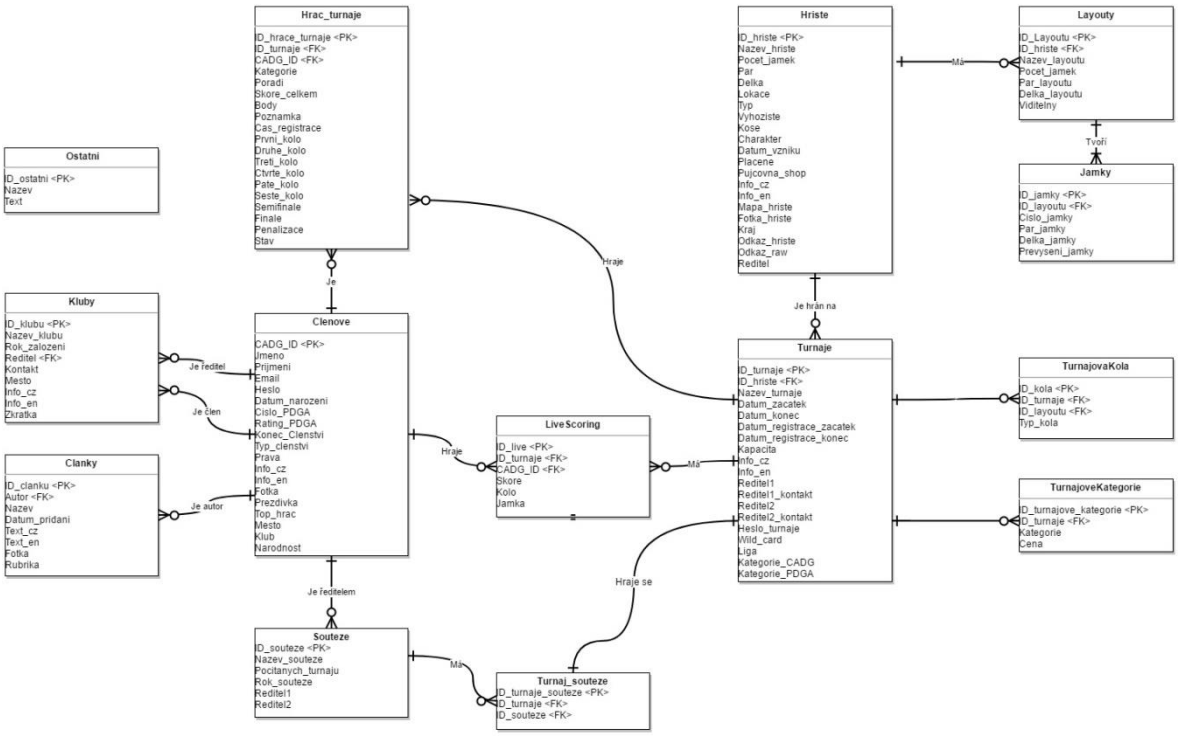
ER diagram popisuje, jak bude vypadat databáze. Jednotlivé obdélníky (entity) představují tabulky databáze, ty mají svůj název a jednotlivé sloupce tabulky (atributy). Tabulky jsou spojené vztahy, které vyjadřují, jak jsou tabulky se sebou propojeny. Pro přehlednost a lepší čitelnost jsem ER diagram rozdělil na dvě části, které jsou propojeny naznačenými vztahy (obrázky 5.2.1 a 5.2.2). Nakonec je zobrazený celý diagram (obrázek 5.2.3), který je hůře čitelný, ale kompletní.




*Obrázek 5.2.1 - levá část ER diagramu*




*Obrázek 5.2.2 - pravá část ER diagramu*




*Obrázek 5.2.3 - celý ER diagram*

# 6 Implementace

Důležitou částí práce je také implementace systému. V této kapitole si představíme několik technologií, které nejsou součástí základního ASP.NET a poté si vysvětlíme, jak jsou naprogramovány některé části systému. Nakonec jsem vybral algoritmy, kterými jsem řešil složitější problémy a vysvětlím je pro jednodušší pochopení funkcionality.

## 6.1 Použité technologie

Kromě prostředků zabudovaných v ASP.NET jsem použil několik dalších technologií, které rozšiřují možnosti systému. Jedná se hlavně o prvky, které ulehčují uživateli práci se systémem a pomáhají lepší přehlednosti webových stránek.

### 6.1.1 TinyMCE

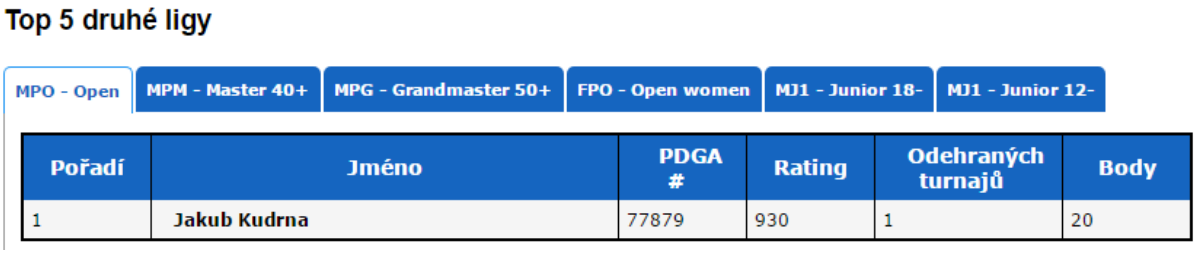
TinyMCE je textový editor typu WYSIWYG (what you see is what you get) [11]. Tato zkratka popisuje způsob editace textu, kdy je výsledná verze textu vzhledově stejná, jako ta při editaci. V systému je použit na stránkách, kde je očekáván od uživatele rozsáhlejší textový vstup. Mezi ně patří vkládání propozic (textových informací) k turnaji, úprava uživatelského profilu nebo psaní článku v blogu.

Tento textový editor značně zjednodušuje formátování textu psaného uživatelem oproti současnému systému, kde je potřeba vkládat tento text čistě pomocí HTML, které většina uživatelů neovládá. Editor stačí stáhnout k ostatním souborům a odkázat se na něj na stránce pomocí tagu *<script>*. Všechny prvky *TextBox* s vlastností *Multiline* se poté automaticky stávají TinyMCE editory.

### 6.1.2 AJAX Control Toolkit

AJAX Control Toolkit je open-source projekt postavený na ASP.NET AJAX frameworku, který přináší nové užitečné prvky přidávané na stránky. Tyto prvky jsou převážně navrženy tak, aby ulehčovaly uživateli práci se systémem, nebo ho dělali hezčí a více přehledný. Instalace probíhá přímo ve Visual Studiu a poté musíme AJAX Control Toolkit přidat zvlášť na každou stránku aby fungoval [12].

V informačním systému je AJAX Control Toolkit využíván pro záložky (TabPanel) na stránce. Využití najdou například při rozdělení výsledků kategorií, kde je vhodné obsah rozdělit a zobrazovat vždy jen jednu část (kategorii). Stránka je potom více přehledná a zbytečně nemusí být vysoká, ukázka na obrázku 6.1.1. Toolkit nabízí k dispozici i další prvky (kalendář, počítadla a podobně), které systém nevyužívá.




*Obrázek 6.1.1 - AJAX control toolkit TabPanel*

# 6.2 Jednotlivé části systému

Nyní si popíšeme způsoby implementace jednotlivých částí systému. Ty jsou rozděleny podle toho, jakým způsobem jsou implementovány na rozdíl od analýzy a návrhu, kde byly rozděleny na stránky a celky, které spolu souvisí z pohledu uživatele. Ještě předtím si zde ale rozebereme některé složitější prvky jako rozsáhlé tabulky, ukládání dat o přihlášeném uživateli, nebo práci s databází.

## 6.2.1 GridView tabulky

Pro většinu tabulek v systému jsou použity prvky ASP.NET Web Controls GridView. Vzhledově se jedná o tabulku, která se ovšem dá pohodlně naprogramovat a tím jí dát mnoho vlastností. Zdroj dat pro GridView je datová struktura *DataTable*, kterou můžeme naplnit daty buď z databáze, nebo dalšími způsoby pomocí C# kódu (pole, proměnné a podobně).

GridView definujeme pomocí elementu *<asp:GridView>*, který může obsahovat množství atributů. Nyní si ukážeme ty, které jsou v systému často používány:

1) ID - každý GridView má na stránce jedinečné ID, podle kterého je identifikovatelný. Pomocí ID v C# kódu určujeme zdroj dat, případně měníme další atributy. Dále slouží pro přiřazení v CSS, obvykle je pro určení stylu vhodné použít jiný atribut, a to *CssClass*, pomocí kterého definujeme třídu prvku pro CSS. Tím pádem můžeme jedním příkazem přiřadit stejný styl všem GridView tabulkám v systému se stejnou třídou.
2) OnRowDataBound - důležitý atribut, pomocí něhož určíme funkci, která proběhne zvlášť pro každý řádek před tím, než je vykreslen ale až po tom, co obsahuje data. Tím můžeme například upravit zobrazená data na řádku (ze sloupců *křestní jméno* a *příjmení* vytvoříme jeden sloupec *jméno* spojením těchto dvou záznamů a přidání mezery mezi ně).
3) Allow Sorting, Allow Paging - udávají, zda je povolené řazení řad, případně rozdělení na více stránek. Zadáváme hodnoty *true/false*.

GridView dále obsahuje elementy, kterými definujeme jeho obsah. Do elementu *<Columns>* umisťujeme potřebné sloupce tabulky, které mohou být přímo data z *DataTable*, nebo další elementy jako *Label, CheckBox* a podobně.

## 6.2.2 Session

Systém podporuje přihlášení uživatelů, takže potřebuje krátkodobě uchovávat data. K tomu slouží *Session*, která si v rámci aplikace ukládá potřebnou hodnotu, kterou poté mohou jednotlivé stránky získávat a podle ní upravovat chování. Když se uživatel přihlásí, do Session systém uloží e-mail, protože jednoznačně identifikuje uživatele. Nová Session s hodnotou např. “e-mail = abcd” se zapíše následovně:

```
Session["e-mail"] = abcd;
```

Pokud na stránce potřebujeme zjistit, jaký uživatel je přihlášený, použijeme následující příkaz:

```
string prihlaseny_uzivatel = Session["e-mail"].ToString();
```

### 6.2.3 Předávání dat mezi stránkami

Důležitá součást systému je také předávání dat mezi jednotlivými stránkami. Když potřebujeme zobrazit například hřiště, máme jednu "šablonu" stránky, která se plní daty právě podle toho, jaké hřiště musí konkrétně zobrazit. Ke zobrazení musí stránka znát jednoznačnou identifikaci, podle které hledá data v databázi, v našem případě ID hřiště (*ID_hriste)*.

První možností, jak stránce tuto informaci předat je *QueryString*, což je text následující v URL stránky za znakem otazníku (*?)*. Tento text obsahuje kolekci dvojic *identifikátor=hodnota* oddělených znakem *&*. Jako příklad uvedeme URL stránky zobrazující detail hřiště s identifikátorem *id_hriste* a hodnotou *15*.

```
/hriste/DetailHriste.aspx?id_hriste=15
```

Na stránce poté hodnotu získáme následujícím příkazem:

```
Request.QueryString["ID_hriste"]
```

Druhá možnost předávání dat mezi stránkami je tzv. Routing [13]. Ten povoluje v rámci systému používání URL adres, které nutně nemusí odkazovat přímo na existující soubor (stránku), ale systém uživatele podle daných pravidel přesměruje na potřebný soubor. Tato pravidla nastavujeme v souboru *Web.config*. Výhodou je, že můžeme vytvořit uživatelsky přívětivé URL adresy, které mohou obsahovat výstižnější popis pro danou stránku. Pokud použijeme příklad z *QueryString* výše, URL adresa odkazující na stejné hřiště může vypadat následovně:

```
/hriste/Detail/15
```

### 6.2.4 Stránky pracující s databází

Systém musí pracovat s databází, odkud si stahuje data, která poté zobrazuje. K práci s databází používáme knihovnu ASP.NET *System.Data.SqlClient*, která pokryje veškeré naše potřeby [14].

Databázi, se kterou systém pracuje, určuje *ConnectionString* obsahující informace o databázi, její umístění a přístupové údaje. Pokud chceme používat stejnou databázi pro celý systém, definujeme jej v souboru *Web.config* a poté k němu na každé stránce přistupujeme pomocí následujícího příkazu:

```
ConfigurationManager.ConnectionStrings["nazev"].ConnectionString;
```

Tímto příkazem stránce řekneme, aby používala databázi s ConnectionString "nazev". Poté potřebujeme vytvořit spojení s databází a otevřít ji pomocí třídy *SqlConnection* [15]. Po spojení můžeme databázi zaslat dotaz, který provedeme pomocí *SqlCommand* [16]. Abychom mohli pracovat

se získanými daty, použijeme metody z další třídy, *SqlDataAdapter* [17]. Ta reprezentuje souhrn připojení k databázi s dotazem, pomocí kterého poté můžeme vložit data do tabulky *DataTable*. S daty v tabulce poté můžeme pracovat jakkoliv bude potřeba.

Pokud potřebujeme do databáze data přidat nebo je upravit, vynecháme třídu *SqlDataAdapter*, zvolíme odpovídající dotaz a provedeme jej příkazem *ExecuteNonQuery()*.

## 6.3 Složitější řešení

V systému se nachází několik složitějších algoritmů, které je vhodné podrobněji popsat, protože při pohledu do kódu by mohlo déle trvat než se programátor, případně budoucí administrátor zorientuje v algoritmu a tím pádem by pro něj bylo složitější dělat úpravy nebo vše správně pochopit.

### 6.3.1 Pořadí hráčů v turnaji a přidělené body

Jeden ze složitějších problémů je zadávání výsledků, počítání skóre k celkovému paru všech odehraných jamek a následné udělení bodů, které dostávají jen hráči, kteří jsou součástí dané ligy.

Tuto funkčnost řeší stránka *turnaje/vysledky.aspx*, ve které uživatel zadá výsledky z turnaje jednotlivých kol pro všechny hráče. Když výsledky chce uložit, spustí první fázi výpočtů - spočítání celkového výsledku, pořadí a bodů.

Prvně se spustí kontrola vstupních dat ve funkci *UlozitClicked()*. Pokud jsou data správná, pro každou hráčskou kategorii se zvlášť zavolá funkce *ziskatVysledky()*, která vytvoří novou tabulku obsahující hráče, body celkem, pořadí a identifikaci členství hráče a informaci zda je TOP hráč nebo ne pro určení, jestli hráč dostane body nebo ne. Potom se pro každého hráče spočítá jeho celkové skóre (součet skóre odehraných kol), skóre jednotlivých kol se uloží do databáze a určí se, jestli hráč odehrál semifinále a finále.

To potřebujeme vědět kvůli řazení výsledků. Pořadí se sice počítá tak, že čím nižší skóre tím lépe, ale zároveň hráči hrající semifinále mají vyšší součet a lepší umístění, stejně jako hráči hrající finále mají vyšší součet než hráči hrající jen semifinále a taktéž jsou umístěni lépe. Tento problém je vyřešen tím, že pokud hráč odehrál semifinále, od součtu je odečteno 1000 a pokud hrál finále, tak je odečteno 10000. Poté se hráč a skóre uloží do tabulky, která je předána funkci *ziskatPoradi()*.

V této funkci pro každý řádek tabulky (hráč) vypočítáme skóre následovně:

1) Seřadíme tabulku hráčů vzestupně podle celkového skóre. S odečtenými hodnotami za semifinále a finále získáme správné pořadí pro všechny hráče. Poté pro každého hráče počítáme pořadí.
2) Pokud má více hráčů stejné skóre, mají stejné pořadí. První hráč, kdo po této situaci má horší skóre má o tolik nižší pořadí, kolik stejných výsledků před ním bylo. Například tři hráči se stejným skóre 50 jsou na pátém místě, další hráč se skóre 51 je osmý. Pokud se skóre liší od předchozího hráče, o 1 inkrementujeme počítadlo pořadí *StejnePoradi* a pokračujeme.
3) Následuje přiřazení bodů. V poli máme definovány body, jaké hráč získá za umístění kde pořadí je roven indexu pole. Tato pole jsou definována pro různé turnajové kategorie zvlášť podle soutěžního řádu ČADG. Pokud se umístí hráč, který nemá nárok získat body (TOP hráč ve druhé lize, nečlen ČADG) nedostane body a inkrementuje počítadlo *neClenove*. Pokud má body získat, uloží se k němu body z odpovídajícího pole s indexem *pořadí-neClenove*.
4) Teď už je pořadí dané, takže můžeme zpět přičíst semifinalistům a finalistům k celkovému skóre hodnoty 1000, resp. 10000 a tím je vrátit na původní hodnotu.

Poté se body a pořadí uloží do tabulky ke hráči a uživateli se zobrazí. Někdy se udělují na turnaji výjimky, proto má uživatel možnost vygenerované pořadí nebo body ručně přepsat. Když už jsou výsledky finální, uloží se do databáze a tím zadávání výsledků turnaje končí.

.

### 6.3.2 Výsledky ligy

Protože by pro databázi bylo zbytečné ukládat po každém turnaji výsledky ligy, rozhodl jsem se je počítat až při vstoupení na stránku výsledků. Pro různé ligy a soutěže se algoritmus skoro neliší, proto si vše ukážeme na příkladu první ligy v souboru */vysledky/PrvniLiga.aspx*.

Funkce *PopulateSite()* naplní daný GridView potřebnou hráčskou kategorií výsledky, které získá následovně:

1) Vytvoří si tabulku *DataTable*, do které bude ukládat informace o hráči, počet získaných bodů, odehraných turnajů a konečné pořadí v lize.
2) Z tabulky *Hrac_turnaje* vybere všechny záznamy odehraných prvoligových turnajů. Pro každého hráče existuje tolik záznamů, kolik odehrál turnajů. Tabulka je seřazená podle hráčů a bodů, takže záznamy jednoho hráče jdou po sobě seřazeny od turnaje s největším počtem získaných bodů po nejmenší.
3) Pro každý záznam tabulky zjistíme, zda se jedná o nového hráče (ještě neregistrovaného v nové tabulce) nebo ne. Pokud ano, vrátíme počítadlo turnajů *Turnaje* na hodnotu 1 a počítadlo bodů *Body* na hodnotu právě počítaného turnaje, uložíme si informace o hráči a pokračujeme dalším záznamem.
4) Pokud se jedná o stejného hráče a zároveň hodnota započítaných turnajů není větší než maximální počet turnajů, ze kterých se počítají body (každá liga má limit, který udává, kolik nejlépe bodovaných turnajů se každému hráči počítá), připočtou se hráči body za daný turnaj k celkovému počtu bodů a inkrementuje se počítadlo započítaných turnajů. Tím docílíme započítání jen nejlepších výsledků.
5) Po spočítání všech hráčů se zavolá funkce *ziskatPoradi()*.

Ta vezme vstupní tabulku z funkce *PopulateSite()*, seřadí ji sestupně podle bodů a vypočítá pořadí stejným algoritmem, jak je popsána funkce *ziskatPoradi()* v kapitole 6.3.1 při počítání pořadí v turnaji. Potom se seřazená tabulka s přidaným pořadím nahraje do odpovídajícího GridView a tím se data zobrazí na stránce k odpovídající kategorii.

# 7 Testování, uvedení do provozu

Součástí vývoje systému by mělo být i testování odhalující nedostatky a chyby, které je potřeba opravit před oficiálním spuštěním a uvedením do provozu. Dále je dobré se nad fungujícím systémem zamyslet a navrhnout další možná rozšíření, která by systém ještě zlepšila.

## 7.1 Testování

Testování softwaru je nezbytná součást vývojového procesu. Existuje množství různých metod testování, každá s nich se hodí pro jiný typ software a také pro jiný testovací tým. Vzhledem k tomu, že discgolfová komunita je poměrně malá a hodně hráčů se zná navzájem, není pro mě problém najít potřebný počet vhodných uživatelů, kteří budou ochotni systém testovat. Tyto uživatele můžeme nazvat testovací tým. Zvolil jsem proto následující metodu, která se mi pro tuto situaci zdála nejvíce efektivní.

1) Testování programátorem (mnou) - po návrhu systému a při programování jeho částí je potřeba se průběžně informovat o tom, zda vše funguje jak má. Po naprogramování např. jedné stránky nebo složité funkce se proto musím ujistit, že systém bude reagovat tak, jak bych očekával. Pokud reaguje nesprávně, problém se snažím opravit. Po naprogramování všech částí systém otestuji jako celek s tím, že zadávám hraniční data, která by mohla působit problémy. Také jako jediný tuším, že existují funkce náchylnější na chyby a ty se snažím otestovat detailněji. Členové testovacího týmu kód neznají, tím pádem na různé problémy nemusí narazit.
2) Až jsem uznal, že je systém funkční a vhodný ke zveřejnění testovacímu týmu, pozval jsem si dobrého kamaráda, který se vyzná v IT a zároveň je hráč discgolfu a požádal ho, ať se na systém podívá. Pokud by našel závažnější chybu nebo problém, na který jsem mohl zapomenout, mohl bych ho opravit. Potom, co jsme se shodli, že je všechno v pořádku, mohu systém zveřejnit pro testovací tým.
3) Testovací tým jsem oslovil přes sociální sítě a kdo měl zájem se zúčastnit, poslal jsem mu odkaz na systém s instrukcemi, jak testovat. Do týmu jsem se snažil v první řadě zařadit zástupce různých uživatelských skupin, například několik juniorů, dospělých uživatelů, nebo i seniory. Zároveň jsem oslovil několik hráčů, kteří se orientují v IT a rádi s testováním pomůžou, stejně jako členy výkonné rady ČADG. Tento testovací tým je proto schopen najít různé chyby a nedostatky jinými způsoby a přístupy k testování. Dále jsem odkaz s testovacím webem poslal komukoliv, kdo o něj měl zájem.
4) Každý člen testovacího týmu má za úkol podívat se na systém, vyzkoušet všechny jeho dostupné funkce (registrace, přihlášení, tvoření hřišť, turnajů, soutěží a podobně) a sdělit jeho dojmy do testovacího formuláře. Ten jsem se snažil udělat vzhledem k široké skupině uživatelů co nejjednodušší, takže obsahuje jen dvě otázky, kde se může uživatel libovolně rozepsat. Formulář se uživatele ptá, co se mu na systému líbí a co by naopak změnil.
5) Výše zmíněné testování probíhá už delší dobu a průběžně testovací tým informuji, jaké jejich návrhy budou zpracovány, případně je rovnou zpracuji. Tato fáze testování bude probíhat do doby, než usoudím, že je všechno v naprostém pořádku a poté systém uvedu do provozu. Původně měl být systém uveden už v dubnu, ale výkonná rada udělala několik změn v soutěžním řádu ČADG a ty ještě bude potřeba otestovat v rámci turnajů, které budou probíhat v červenci a srpnu.

## 7.2 Uvedení do provozu

Po důkladném otestování systému přichází čas na uvedení do provozu. Začneme s prázdnou databází, do které se budou postupně moci zaregistrovat všichni hráči a uživatelé. Měli by na to mít dost času, protože oficiálně se informační systém začne plně používat až 1.1.2018. Do té doby se turnaje a výsledky budou zapisovat primárně na současně fungující systém. Jediné zásahy do databáze proběhnou v případě, že si hráč bude chtít zachovat současné číslo ČADG (primární klíč CADG_ID v databázi členů). V tom případě je administrátor ručně přepíše. Od začátku sezóny 2018 tedy ČADG bude už plně fungovat jen na našem novém systému.

## 7.3 Možná rozšíření

Jako u většiny informačních systémů je i v našem mnoho věcí, kterými by se dal systém do budoucna vylepšit a rozšířit. Některé možnosti si zde uvedeme:

1) Synchronizace turnajů se sdíleným kalendářem - uživatelé používající online kalendář (nejčastěji od Google) by dostali možnost automaticky do něj synchronizovat názvy a data turnajů, do kterých se zaregistrují.
2) Přidávání komentářů - u hřišť, turnajů, výsledků a dalších stránek by mohla být sekce komentářů, kde by přihlášení uživatelé mohli přidávat krátké textové příspěvky a vést diskuse.
3) Podpora dvojic - jeden z formátů turnaje je i hra dvojic (někdy také doubles), kde místo jednotlivců soutěží dvojice hráčů. Ty by potom měli vlastní registraci, turnaje i výsledky soutěží.
4) Více jazyků - systém by mohl být v češtině i v angličtině. Při návrhu databáze jsem na tuto možnost už myslel a velmi pravděpodobně bude později naimplementována.
5) Odesílání e-mailů - když by se hráč registroval na turnaj, přišel by mu e-mail se základními informacemi a potvrzením. Stejně tak by tato možnost mohla být u přidávání turnajů, hřišť a podobně.

# 8 Závěr

Práce byla vytvořena bez zásadních problémů a je plně funkční tak, jak bylo specifikováno v zadání, do reálného provozu však zatím spuštěna nebyla.

Je to z toho důvodu, že stále probíhají testy discgolfovou komunitou a navrhují se další rozšíření. Zadání bylo ze strany asociace v průběhu jara změněno díky novému soutěžnímu řádu, což bude po odevzdání práce vyžadovat několik drobných změn. Oficiální spuštění je naplánováno až na začátek nové sezóny k 1.1.2018. Systém je však připravený na spuštění i v případě, že by se žádné další změny už nedělaly.

Z průběžného testování vyplývá, že mezi největší nedostatky patří neschopnost systému zasílat automatické informativní e-maily uživatelům a možnost změny hesla po registraci. Dále zřejmě bude potřeba navrhnout a implementovat specifický design pro zobrazení na mobilních zařízeních s menším rozlišením displeje.

Co se týče implementace, samotný ASP.NET nestačil, protože jsem se snažil co nejvíce uživatelům zjednodušit práci se systémem. Místo obvyklého JavaScriptu jsem se rozhodl použít v práci popsaný nástroj AJAX Control Toolkit, který se mi zdál pro moje potřeby vhodnější.

Tato práce mi však ukázala, že vývoj informačních systémů je obor, který mě velmi baví a pravděpodobně v něm budu pokračovat v praxi. Poprvé jsem si zkusil, jak zpracovat rozsáhlé požadavky a podle nich navrhnout systém i s databází a také jsem zjistil jaké to je, když se každý měsíc programátorovi mění zadání. Dále jsem se naučil pracovat s ASP.NET, který se mi velmi zalíbil a rád bych s ním v budoucnu pracoval.

# 9 Literatura

[1] *What Are Web Applications?* [online]. [cit. 2017-03-02]. Dostupné z: https://www.acunetix.com/websitesecurity/web-applications/

[2] *CSS Introduction* [online]. [cit. 2017-03-02]. Dostupné z: https://www.w3schools.com/css/css_intro.asp

[3] *How CSS works* [online]. [cit. 2017-03-04]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/CSS/Introduction_to_CSS/How_CSS_works

[4] *Understanding Mobile Apps* [online]. [cit. 2017-03-04]. Dostupné z: https://www.consumer.ftc.gov/articles/0018-understanding-mobile-apps

[5] *Mobilní aplikace* [online]. [cit. 2017-03-04]. Dostupné z: http://wiki.knihovna.cz/index.php/Mobiln%C3%AD_aplikace

[6] *PHP: Hypertext Preprocesor* [online]. [cit. 2017-03-05]. Dostupné z: http://php.net/

[7] *ASP.NET overview* [online]. [cit. 2017-03-05]. Dostupné z: https://docs.microsoft.com/cs-cz/aspnet/overview

[8] *Visual C# IntelliSense* [online]. [cit. 2017-04-12]. Dostupné z: https://msdn.microsoft.com/en-us/library/43f44291.aspx

[9] *ASP.NET Web Server Controls* [online]. [cit. 2017-03-12]. Dostupné z: https://msdn.microsoft.com/en-us/library/bb386416.aspx

[10] *Discgolf.cz: O asociaci* [online]. [cit. 2017-03-12]. Dostupné z: http://discgolf.cz/index.php?id=cadg_asociace

[11] *TinyMCE* [online]. [cit. 2017-04-14]. Dostupné z: https://www.tinymce.com/

[12] *ASP.NET AJAX Control Toolkit* [online]. [cit. 2017-04-14]. Dostupné z: http://www.ajaxtoolkit.net/

[13] *ASP.NET Routing* [online]. [cit. 2017-04-17]. Dostupné z: https://msdn.microsoft.com/en-us/library/cc668201.aspx

[14] *System.Data.SqlClient Namespace* [online]. [cit. 2017-04-20]. Dostupné z: https://msdn.microsoft.com/en-us/library/system.data.sqlclient(v=vs.110).aspx

[15] *Třída SqlConnection* [online]. [cit. 2017-04-20]. Dostupné z: https://msdn.microsoft.com/cs-cz/library/system.data.sqlclient.sqlconnection(v=vs.110).aspx

[16] *Třída SqlCommand* [online]. [cit. 2017-04-20]. Dostupné z: https://msdn.microsoft.com/cs-cz/library/system.data.sqlclient.sqlcommand(v=vs.110).aspx

[17] *Třída SqlDataAdapter* [online]. [cit. 2017-04-20]. Dostupné z: https://msdn.microsoft.com/cs-cz/library/system.data.sqlclient.sqlcommand(v=vs.110).aspx

# Seznam příloh

Příloha 1: CD se zdrojovým kódem.