UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

DIPLOMOVÁ PRÁCE

2017 Bc. Jakub Hyksa

Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Webový vědomostní kvíz

Bc. Jakub Hyksa

Diplomová práce

2017

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2016/2017**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Jakub Hyksa**
Osobní číslo:	**I14257**
Studijní program:	**N2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Webový vědomostní kvíz**
Zadávající katedra:	**Katedra softwarových technologií**

Zásady pro vypracování:

Cílem práce je navrhnout a realizovat webovou aplikaci typu zábavný vědomostní kvíz. Aplikace bude responsivní a tedy dostupná i na mobilních zařízení.
V teoretické části bude provedena rešerše aplikací typu kvíz nebo vědomostní hra. V práci budou popsány .NET technologie potřebné pro vývoj tohoto typu aplikací. Dále budou uvedeny typy a popis webových služeb.
Webová aplikace bude implementována pomoci technologie ASP.NET. Pro komunikaci mezi webovými zařízení budou využity webové služby. Uživatel bude mít možnost vytvořit kvíz, kde definuje jeho parametry, poté může hrát s dalšími uživateli, nebo bude celou hru řídit.

Rozsah grafických prací:

Rozsah pracovní zprávy: **cca 65 stran**

Forma zpracování diplomové práce: **tištěná**

Seznam odborné literatury:

***GALLOWAY, Jon. Professional ASP.NET MVC 5. Wiley, 2014. ISBN 1118794753.**
***LACKO, L'uboslav. Mistrovství v SQL Server 2012: Kompletní průvodce databázového experta. COMPUTER PRESS, 2013. ISBN 978-80-2513-773-4.**
***CIBRARO, Paulo. Professional WCF 4: Windows Communication Foundation with .NET 4. Wrox, 2010. ISBN 9780470563144.**

Vedoucí diplomové práce: **Ing. Zdeněk Šilar, Ph.D.**
Katedra informačních technologií

Datum zadání diplomové práce: **31. října 2016**
Termín odevzdání diplomové práce: **17. května 2017**

Ing. Zdeněk Němec, Ph.D.
děkan

L.S.

prof. Ing. Antonín Kavička, Ph.D.
vedoucí katedry

V Pardubicích dne 15. listopadu 2016

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 17. 05. 2017

Bc. Jakub Hyksa

PODĚKOVÁNÍ

Chtěl bych poděkovat Ing. Zdeňkovi Šilarovi, Ph.D za odborné vedení a konzultace, které mě vedly k úspěšnému dokončení diplomové práce.

Dále bych chtěl poděkovat své rodině a přátelům za jejich podporou po celou dobu studia.

ANOTACE

Cílem práce je navrhnout a realizovat webovou aplikaci typu zábavný vědomostní kvíz. Aplikace bude responzivní a tedy dostupná i na mobilních zařízení.

V teoretické části bude provedena rešerše aplikací typu kvíz nebo vědomostní hra. V práci budou popsány .NET technologie potřebné pro vývoj tohoto typu aplikací. Dále budou uvedeny typy a popis webových služeb.

Webová aplikace bude implementována pomoci technologie ASP.NET. Pro komunikaci mezi webovými zařízení budou využity webové služby. Uživatel bude mít možnost vytvořit kvíz, kde definuje jeho parametry, poté může hrát s dalšími uživateli, nebo bude celou hru řídit.

KLÍČOVÁ SLOVA

kvíz, soutěžící, webová služba, otázka, odpověď, statistika, server, klient

TITLE

Web knowledge quiz

ANNOTATION

The aim of the thesis is to design and implement a web application of type of entertaining knowledge quiz. The application will have responsive web design and available on mobile devices.

The theoretical part recherche apllications of type of quiz or knowledge game. The work will be describe the .NET technologies needed to develop this type of applications. Web services types and descriptions will also be listed.

The web application will be implemented using ASP.NET technology. Web devices will used web services for communicate. The user will be able to create a quiz, where he defines his parameters, then he can play with other users or control the entire game.

KEYWORDS

quiz, competitor, web service, question, answer, statistics, server, client

OBSAH

SEZNAM ILUSTRACÍ, TABULEK A SEZNAM ZDROJOVÝCH KÓDŮ

SEZNAM ZKRATEK A ZNAČEK

HTTP	Hypertext Transfer Protocol
XML	eXtensible Markup Language
JSON	JavaScript Object Notation
SOAP	Simple Object Access Protocol
REST	Representational state transfer
URI	Uniform Resource Identifier
API	Application Programming Interface
MVC	Model-View-Controller
LINQ	Language Integrated Query
SQL	Structured English Query Language
CRUD	Create, Read, Update, Delete
UML	Unified Modeling Language

ÚVOD

Diplomová práce se zabývá tématem zábavný vědomostní kvíz nebo vědomostní hra. Práce je rozdělena do částí teoretické a praktické.

V teoretické části je provedena rešerše vybraných existujících aplikací a jejích vlastnosti jsou porovnávány s diplomovou prací. Mezi vybranými aplikacemi jsou aplikace dostupné pro desktopové i mobilní zařízení. U vybraných aplikací je kladen důraz na vlastnosti, jako je typ a počet otázek, možnost výběru kategorií otázek a možnost výběru více kategorií u jednoho spuštění kvízu. Dále je zkoumán maximální počet soutěžících a jejich nutnost přihlášení při spuštění kvízu. Posledními zkoumanými vlastnostmi je možnost nápovědy a časový limit pro odpověď. Na základě těchto vlastností je vytvořena srovnávací tabulka. V další kapitole teoretické části jsou představeny technologie .NET potřebné pro vývoj aplikací tohoto typu. V kapitole je představena architektura MVC a technologie potřebné pro vývoj jsou rozděleny do třech částí, ze kterých se skládá architektura MVC. V poslední kapitole této části jsou popsány webové služby.

V praktické části je provedena analýza Webového vědomostního kvízu. V kapitole jsou představeny funkční a nefunkční požadavky. Dále jsou zobrazeny UML diagramy, jako například UML Usecase diagram, UML Activity diagram a UML diagram tříd. V diagramu tříd je představen pouze jeho výřez vzhledem ke složitosti diagramu. Poslední část kapitoly je věnována návrhu databáze. V poslední kapitole je představena implementace Webového vědomostního kvízu. Aplikace je rozdělena do dvou částí, a to serverové a klientské. V serverové části je představena webová služba a spojení aplikace s databázovou vrstvou. V klientské části jsou představeny důležité funkcionality aplikace. Aplikace je navržena pomocí responzivního designu, takže umožnuje zobrazení na mobilním zařízení. Webový vědomostní kvíz je možné spustit pro více hráčů, a proto je v serverové i klientské části implementována komunikace v reálném čase.

1 VYBRANÉ EXISTUJICÍ APLIKACE

1.1 Dobyvatel

Dobyvatel[1] je webová vědomostní hra, která obsahuje i strategické prvky. Hra se skládá z více typů her, a to Běžná pravidla, Dlouhá kampaň, Mini turnaje a Přátelská hra. Jsou zde dva typy pravidel Běžná pravidla a pravidla pro dlouhou kampaň. Obě pravidla se skládají ze tří společných částí, kde se trošku liší. Dlouhá kampaň obsahuje jednu část navíc. [1] [2]

V následující části budou stručně popsány pravidla dlouhé kampaně, která se skládá ze čtyř částí Stavba základny, Expanze, Dobývání a Bitva. [2]

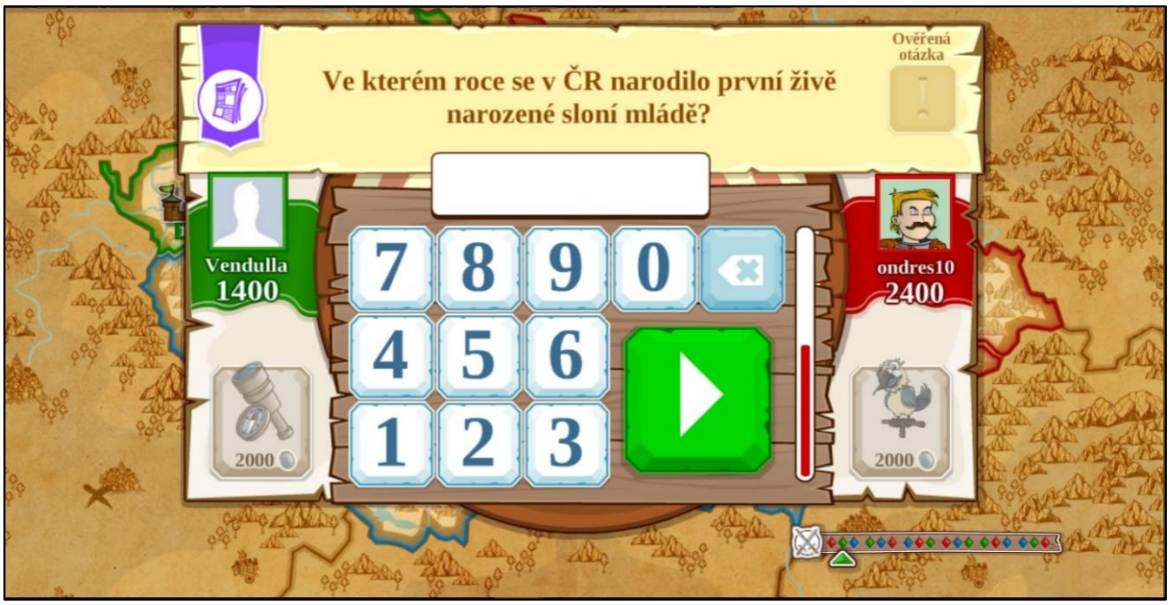
Stavba základny spočívá v zabrání jednoho území na mapě České republiky. Každý hráč si postaví jednu základnu, které se nesmějí vzájemně dotýkat svými hranicemi. Každá základna obsahuje tři věže, které je možné v dalších částí hry dobýt. [2]

V části Expanze bojují hráči o zbývající území na mapě. Tato část se skládá z šesti kol a v každém kole si hráč vybere území. Následně musí hráč odpovědět na otázku, pokud na ní dobře odpoví, získává vybrané území. Hráčům je zobrazena otázka se čtyřmi odpověďmi, kde je pouze jedna odpověď správná. Na obrázku 1 je zobrazena část Expanze, kde si hráči vybírají volná území. [2]

Obrázek 1 - Část Expanze. Zdroj autor

[1] http://dobyvatel.nova.cz/

V části Dobývání se hraje o zbývající území, která nebyla obsazeny v části Expanze. Při výběru území se hráčům zobrazí tipovací otázka, kde hráč zadává celočíselný tip na zobrazenou otázku. Vyhrává hráč, jehož tip je nejbližší správné hodnotě. V případě shody vyhrává hráč, který odpověděl rychleji. Na obrázku 2 je zobrazena tipovací otázka. [2]



Obrázek 2 - Tipovací otázka. Zdroj autor

Po obsazení všech částí mapy se hra přesouvá do části Bitva. Bitva se odehrává v šesti kolech, kde si jednotlivý hráč vybírá protihráčovo území, které chce dobýt. Boj o území probíhá v odpovědění na otázku mezi dvěma hráči. Na obrázku 3 je zobrazená otázka obsahující čtyři odpovědi, kde je pouze jedna správná. [2]

Obrázek 3 - Dobyvatel otázka. Zdroj autor

Pokud oba odpoví správně, je zobrazena tipovací otázka, která určí vítěze. Pokud vyhraje hráč útočící na území, tak území získává. Pokud vyhraje bránící hráč, tak mu území zůstává. Hráč může, také zaútočit na soupeřovu základnu. Základna je dobyta, pokud útočící hráč odpoví na tři správné otázky. Každá správná otázka zničí soupeřovu věž. Za každou správnou odpověď získává hráč body. Na konci hry, tedy po šesti kolech, vyhrává hráč s nejvíce body. Podle získaných bodů získává hráč body zkušeností, které rozšiřují celkovou hru Dobyvatel. [2]

Typ hry přátelská hra je souboj s přáteli a je možné si zvolit běžná pravidla nebo pravidla pro Dlouhou kampaň. Tento typ hry mohou hrát dva nebo tři hráči. Typ hry Mini turnaje je hra s dvaceti sedmi hráči po třech kolech, kde postupují jenom vítězové. V prvním kole proběhne devět her paralelně, kde v každé hře soutěží tři hráči. Ve druhém kole se hrají tři hry paralelně znovu po třech hráčích a v posledním kole hrají tři nejlepší. [3]

Hru dobyvatel hrají vždy tři hráči proti sobě. Hráč má omezený čas na odpověď, a to zhruba 12 sekund. V případě, že hráč není spokojený se zadáním otázky, muže otázku nahlásit. Otázku také může ohodnotit palcem nahoru, nebo palcem dolu.

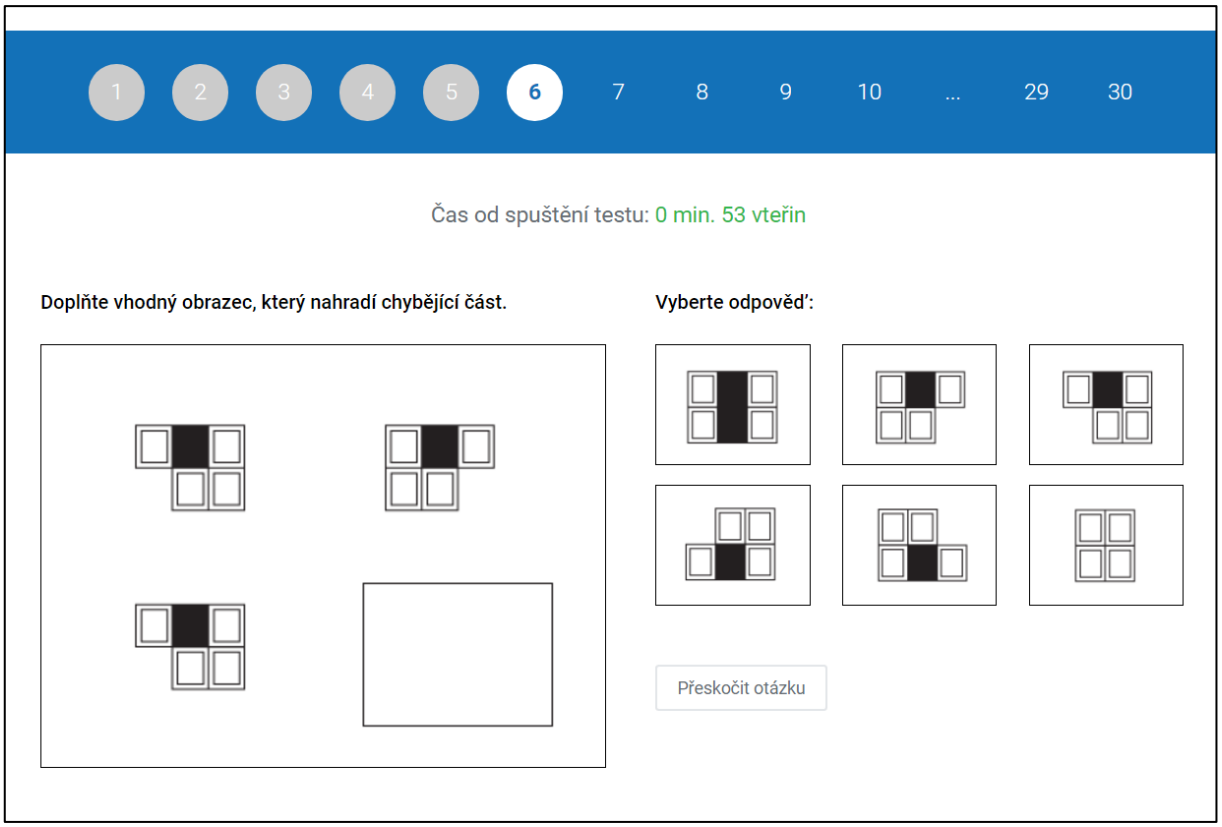
Ve hře je spoustu dalších rozšíření, jako nápovědy, klany, různé žebříčky a spousty dalších vylepšení, které nebudou dále popsány, protože není účelem kapitoly dopodrobna popsat hru Dobyvatel.

1.2 IQ test

IQ test[2] je typ testu, pomocí kterého si uživatel změří úroveň mentálních a rozumových schopností. Lze je vyjádřit číslem, které nazýváme inteligenční kvocient, neboli IQ. „*IQ testy patří mezi tzv. psychologické testy, které zkoumají různé aspekty našeho duševního chování. Snaží se postihnout co nejvíce úloh a problémů. Tím se zjistí všeobecná inteligence a vyloučí se zkreslení dané nějakým zvláštním talentem nebo nadáním. Proto například bývají testy směsicí číselných a obrázkových úloh.“* [4]

IQ test se skládá ze třiceti otázek a na odpověď není omezený čas. Zadaná otázka má většinou obrázkové, nebo číselné zadání a jsou k ní přiřazeny obrázkové, nebo číselné odpovědi. Na každou otázku je vždy jedna správná odpověď. IQ testy jsou předem definované testy, takže uživatel nemá možnost nějaké konfigurace. Na obrázku 4 je zobrazena otázku IQ testu.

[2] https://www.iqtest-certification.com/cs-cz/



Obrázek 4 - IQ Test. Zdroj autor

1.3 Chcete být milionářem

Chcete být milionářem je zábavná vědomostní soutěž. Poprvé se objevila, jako televizní soutěž. V dnešní době je soutěž také dostupná na elektronických zařízení, a to v desktopovém režimu na serverech s online hrami, nebo na mobilních zařízeních. Na mobilních zařízeních je dostupná v Apple store[3], Google play[4] pod názvem Milionář a na Microsoft store pod názvem Chcete být milionářem.

Princip soutěže je odpovědět na co nejvíce otázek a získat co největší finanční odměnu. Každá otázka je ohodnocena různou finanční částkou. V soutěži jsou dva záchytné body. Pokud se soutěžící dostane na záchytný bod, tak v případě špatné odpovědi získá finanční odměnu posledního dosaženého záchytného bodu. Soutěžící také může v libovolném kole ukončit soutěž a získá finanční odměnu poslední dosažené otázky. Soutěž obsahuje patnáct otázek a záchytné body jsou na páté a desáté otázce.

[3] https://itunes.apple.com/cz/app/milion%C3%A1%C5%99/id638221901?mt=8
[4] https://play.google.com/store/apps/details?id=org.trywin.czech.millionaire&hl=cs

Soutěžícímu jsou zobrazeny otázky, kde se s každou další otázkou zvyšuje její obtížnost. Každá otázka má k dispozici výběr ze čtyř odpovědí, kdy je pouze jedna správná. V případě, že soutěžící nezná odpověď na otázku, může využít tři nápovědy:

- 50 na 50 – Nápověda skryje dvě špatné odpovědi.
- Pomoc publika – Nápověda zobrazí graf, kde je u každé odpovědí zobrazeno procentuální počet odpovědí publika na danou otázku.
- Přítel na telefonu – Nápověda simuluje telefonování příteli, kde je zobrazeno náhodné jméno přítele a jeho odpověď.

Na obrázku 5 je zobrazena mobilní verze aplikace Milionář.

Obrázek 5 - Milionář. Zdroj autor

1.4 Mozkovna

Mozkovna[5] je vědomostní hra dostupná na mobilních telefonech. Ke stažení je na Google play a App Store. Po spuštění aplikace je zobrazena úvodní obrazovka, kde je možnost přihlásit se pomocí e-mailu, Facebooku, nebo je možné tento krok přeskočit a nechat si vygenerovat jméno.

[5] http://hramozkovna.cz/

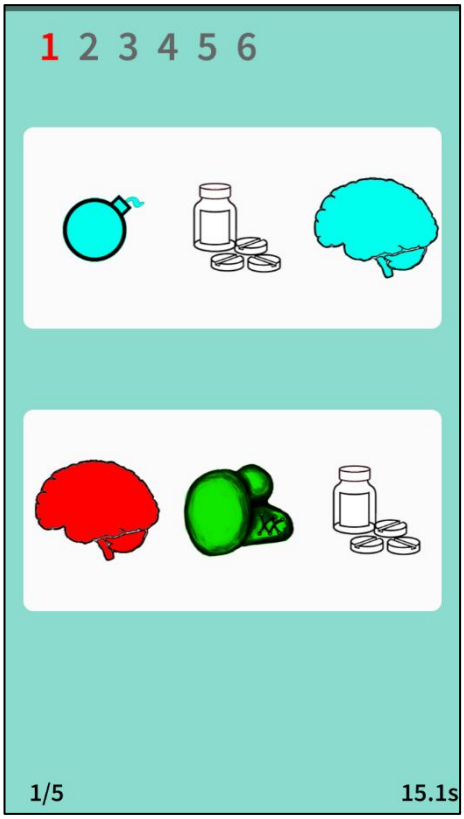
Vygenerované jméno je přiřazené postavičce, která uživatele doprovází celou hru. Vzhled postavičky je možné kdykoliv měnit.

Hra obsahuje dva mody Nová hra a Souboje. V případě vybrání Nové hry, je spuštěna hra, kde hrajete proti vygenerované postavičce. V případě souboje může uživatel vyzvat protihráče, kterého může najít pomocí zadání e-mailu, Facebook, nebo ID hráče. Po vyhledání protihráče, je spuštěna hra. Před spuštěním je vždy vybrán tematický balíček, ze kterého se vygenerují otázky. Spuštěná hra je zobrazena na motivu boxerského utkání. Hra obsahuje šest kol neboli šest otázek. Ke každé otázce jsou zobrazeny čtyři odpovědi, kde je vždy jedna správná. Uživatel má na zodpovězení omezený čas a to 15 sekund. Po zodpovězení je uživateli zobrazena správná otázka, pokud uživatel odpoví dobře, je mu připočten jeden bod. U některých otázek se po odpovědění zobrazí poučný popisek, který vysvětluje danou odpověď. Na obrázku 6 je zobrazena otázka se čtyřma odpověďmi.

Obrázek 6 - Hra mozkovna. Zdroj autor

V případě, že uživatel neví správnou odpověď, má možnost použít bombičku, která uživateli změní otázku za jinou. Počet bombiček je omezený. Po prvním spuštění aplikace jsou k dispozici tři bombičky. Bombičky si může uživatel dokupovat za vyhrané zlaťáky. Další možností je použít nápovědu pomocí minihry. Uživateli je zobrazena minihra, kterou musí

splnit co nejrychleji a s nejlepším možným řešením. Na základě úspěšnosti vyřešení minihry se skryjí špatné odpovědi. V aplikaci jsou dostupné různé typy miniher. Na obrázku 7 je zobrazena minihra, kde musí uživatel v co nejrychlejším čase vybrat dva shodné předměty.



Obrázek 7 - Mozkovna minihra. Zdroj autor

Po zodpovězení šesti otázek je uživateli zobrazeno vyhodnocení hry. V případě, že odpověděl dobře, získává jeden bod. V opačném případě získává bod protihráč. Podle počtu bodů získává zlaťáky. Za zlaťáky je možné pořídit bombičky, balíčky otázek, nebo koupit oblečky pro svoji postavičku. Při prvním spuštění aplikace má uživatel k dispozici 100 zlaťáků a základní balíček otázek. K dispozici jsou různé balíčky např.: Historie starověku, Přírodní vědy, Astronomie, Biologie atd. Každý balíček je ohodnocen jiným počtem zlaťáků a také různým počtem otázek. U každého balíčku je zobrazena úspěšnost správných odpovědí, a také počet odpovědí, na které už uživatel odpověděl. V aplikaci jsou také dostupné in-ap nákupy, pomocí kterých si uživatel kupuje zlaťáky.

1.5 Srovnaní vybraných aplikací

V tabulce 1 jsou porovnány vybrané aplikace.

Tabulka 1 - Porovnání vybraných aplikací. Zdroj autor

	Dobyvatel	IQ test	Chcete být milionářem	Mozkovna	Webový Vědomostní kvíz
Počet otázek	12 a více	30	15	6	libovolný
Počet typů odpovědí	2	1	1	1	4
Možnost vybrat kategorie	NE	NE	NE	ANO	ANO
Různé kategorie v průběhu soutěžení	ANO	NE	ANO	NE	ANO
Omezeny čas na odpověď	ANO	NE	NE	ANO	ANO i NE
Maximální počet soutěžících	3	1	1	2	libovolný
Nápověda	ANO	NE	ANO	ANO	NE
Nutnost přihlášení	ANO	ANO	NE	NE	NE

Aplikace Webový vědomostní kvíz byla porovnána s ostatními existujícími aplikacemi. Z tabulky je zřejmé, že aplikace Webový vědomostní kvíz obsahuje nejlepší vlastnosti mezi těmito vybranými aplikacemi. V aplikaci je neomezený počet otázek, možnost vybírat různý typ kategorií a otázek. Počet možných soutěžících je libovolný. Aplikace neobsahuje oproti ostatním aplikacím možnost nápovědy.

2 TECHNOLOGIE .NET POTŘEBNÉ PRO VÝVOJ

V této kapitole budou představeny technologie, které byly použity pro tvorbu Webového vědomostního kvízu. Aplikace byla napsána pomocí architektury MVC, kde v každé vrstvě byly použity jiné technologie.

2.1 Architektura MVC

Je moderní softwarová architektura, která se skládá ze tří částí Model, View a Controller. Architektura byla popsána v roce 1979 Trygvem Reenkaugem. Odděluje aplikační kód (Model), kód zobrazující data (View) a kód obsluhy (Controller). [5]

Základem celé aplikace je model, který si uchovává svůj vnitřní stav, obsahuje bussiness logiku a komunikuje s databází. Model je nezávislá vrstva v architektuře a neví o částech View a Controller. [5]

View je grafický výsledek aplikace. Přijímá data od controlleru a pomocí front-endových frameworků data zobrazuje. [5]

Controller je prostředník mezi modelem a view. Od view přijímá události, které zpracuje a předá je části modelu. Z opačné části architektury si controller řekne modelu o data, která pošle k zobrazení do view. [5]

2.2 Entity framework

Entity framework je objektově-relačně mapovací framework, který umožnuje konverzi dat mezi relační a objektově orientovaným jazykem. Při konverzi se datová tabulka mapuje na objekt programovacího jazyka. Programátor poté vytváří LINQ dotaz, kde načítá a manipuluje s daty v podobě objektů. LINQ, neboli Language Integrated Query, je jazyk, který se pomocí jednotné syntaxe dotazuje k datům. Zdrojový kód 1 zobrazuje příklad LINQ dotazu, kde je vybrán student s jménem Bill. [7] [8]

```
//Querying with LINQ to Entities
using (var context = new SchoolDBEntities())
{
    var L2EQuery = context.Students.where(s => s.StudentName == "Bill");

    var student = L2EQuery.FirstOrDefault<Student>();

}
```

Zdrojový kód 1 - LINQ dotaz. [8]

V Entity frameworku je také možné použít klasický SQL dotaz. Ve zdrojovém kódu 2 je použit klasický SQL dotaz pro získaní všech studentů. [8]

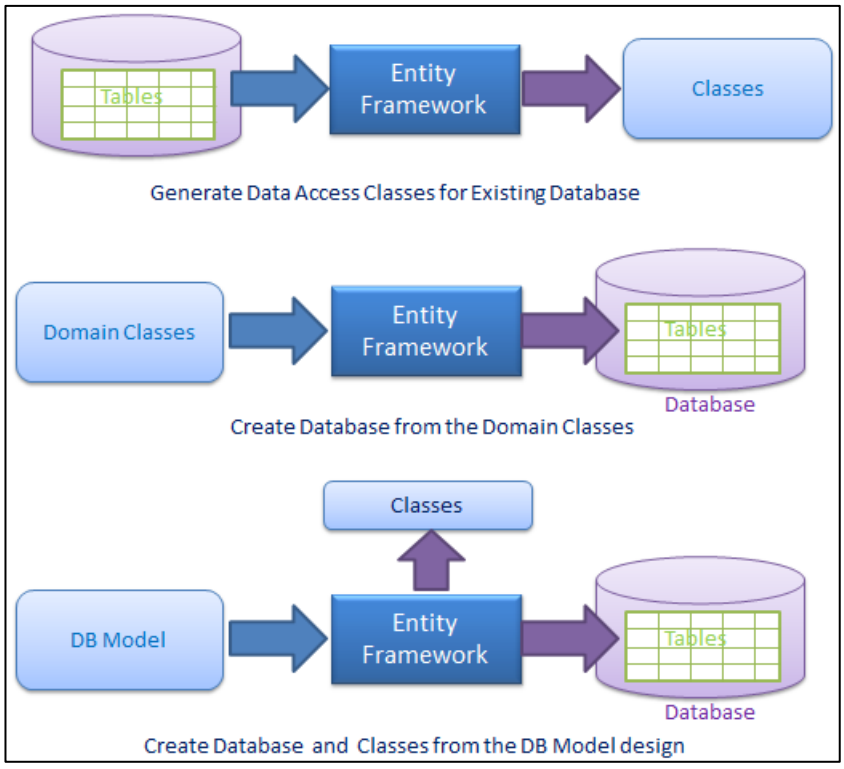
```
using (var ctx = new SchoolDBEntities())
{
    var studentList = ctx.Students.SqlQuery("Select * from Student").ToList<Student>();

}
```

Zdrojový kód 2 - SQL dotaz. [8]

Entity framework je open-source projekt a do projektu ho lze naimportovat pomocí Nuget balíčku. [7]

Entity framework se dá využít ve třech scénářích. Prvním případ využití je v případě, kdy máme hotovou databázi a Entity framework nám z databáze vytvoří objekty. Druhý případ je, kdy máme vytvořené třídy a databázi vygenerujeme na základě tříd. Posledním případem je návrh schématu pomocí vizuálního návrháře, a poté dojde k vygenerování databáze. Popsané scénáře znázorňuje obrázek 8. [6]



Obrázek 8 - Využití Entity frameworku. [6]

Entity framework využívá dva typy tříd POCO a Dynamic Proxy. POCO třída, neboli Plain Old CLR Object, je klasická .NET třída, která není závislá na žádné jiné. Umožňuje klasické chování pro vkládání, upravování a mazaní, jako generované třídy. Dynamic Proxy třída je

runtimová POCO třída, která umožnuje opožděné načítání a automatickou detekci změn. Aby třída mohla být Dynamic Proxy, musí splňovat následující kritéria:

- musí být deklarováná pomocí klíčového slova *public*,
- nesmí být uzavřená,
- nesmí být abstraktní,
- property, neboli stav třídy, musí být deklarována jako *public* a *virtuál,*
- kolekce musí být typu *ICollection<T>*,
- vlastnost *ProxyCreationEnabled* nesmí být *false (*děfaultně *true).*

Ve zdrojovém kódu 3 je znázorněna Dynamic Proxy třída Student, která splňuje všechny výše zmíněná kritéria. [7] [9]

Entity framework podporuje vztahy mezi tabulkami, stejně jako databáze:

- One-to-One.
- One-to-Many.
- Many-to-Many.

Pokud tabulka obsahuje vztah na další tabulku, je po objektovém mapování v dané třídě vytvořen odkaz na konkrétní třídu. Ve zdrojovém kódu 3 je červeně vyznačen vztah One-to-*, kdy třída obsahuje instanci objektu. Modře je vyznačen vztah Many-to*, kde třída obsahuje kolekci objektů dané třídy. [7] [9]

```
public partial class Student
{
    public Student()
    {
        this.Courses = new HashSet<Course>();
    }

    public int StudentID { get; set; }
    public string StudentName { get; set; }
    public Nullable<int> StandardId { get; set; }
    public byte[] RowVersion { get; set; }

    public virtual Standard Standard { get; set; }
    public virtual StudentAddress StudentAddress { get; set; }
    public virtual ICollection<Course> Courses { get; set; }
}
```

Zdrojový kód 3 - Dynamic Proxy třída Student. [9]

2.3 ASP .NET

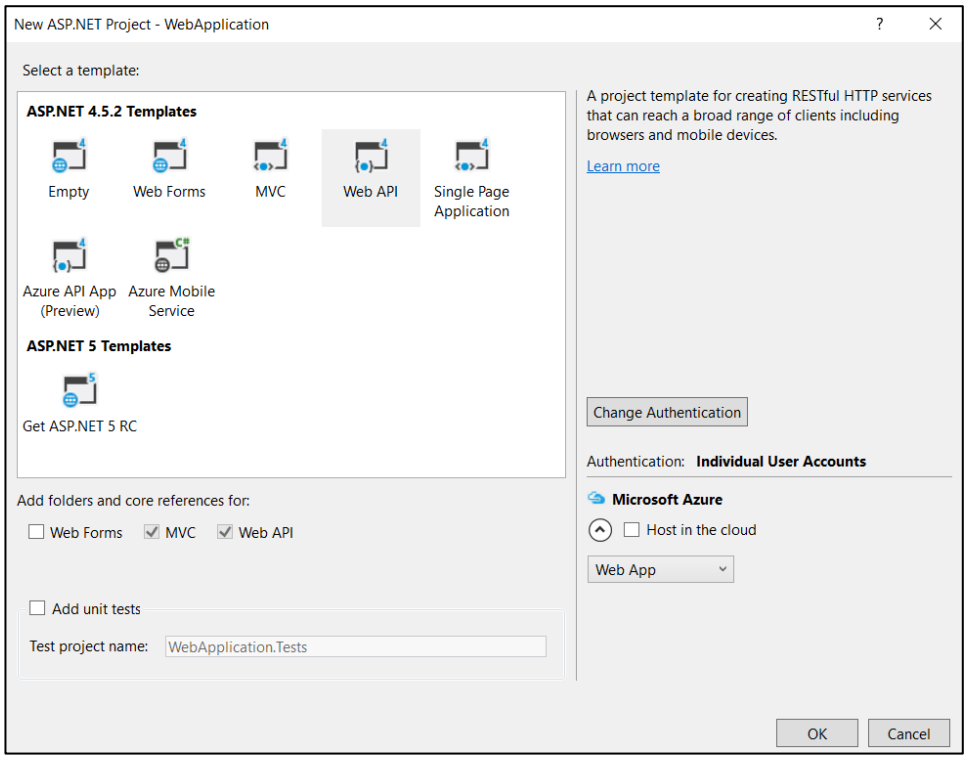
Je framework pro vytváření webových aplikací. Vyvinula ho společnost Microsoft. Je určen k vytváření dynamických webových stránek, webových aplikací a webových služeb. Poprvé

byl vydán v roce 2002 a je nástupcem technologie ASP, neboli Active Server Pages, také vyvinutý společností Microsoft. Technologie ASP .Net je postavena na CLR, neboli Common Language Runtime, který umožnuje vytvářet aplikace pomocí libovolného .Net jazyka. [10]

CLR využívá specifické knihovny a překladače programovacích jazyků. Programy se poté překládají do mezi-jazyka, který se pomocí překladače přeloží do strojového jazyka. Z kompilované části poté využívá funkce cílového procesoru. Je podporována velká množina jazyků, mezi které patří například C#, C++, F# a VB.NET. [10]

Mezi nejpoužívanější jazyky patří C#. C# je programovací jazyk, který běží na rozhraní .NET framework. Je to jednoduchý objektově orientovaný jazyk. Zjednodušuje některé složitosti Jazyka *C++*. Obsahuje například delegáty, lambda výrazy a přístupy do paměti, které nejsou umožněny v jazyku Java . Jazyk samozřejmě obsahuje více odlišností od jazyků Java a C++, ale dále se nimi nebudeme zaobírat, jelikož to není účelem této kapitoly. [10] [11]

ASP .NET obsahuje předdefinované předlohy pro vývoj aplikací. V následujících podkapitolách si představíme nejpoužívanější předlohy. Mezi nejznámější předlohy patří MVC, WEB API a Single Page Application. Programátor nemusí využívat předdefinované předlohy a muže si vytvořit prázdný projekt, kde si adresářovou strukturu navrhne podle svých představ. Na obrázku 9 jsou zobrazeny dostupné předlohy.



Obrázek 9 - ASP. net předlohy. Zdroj autor

Pro vývoj aplikace založené na MVC architektuře je možné v ASP.NET vytvořit aplikaci pomocí dvou způsobů. První způsob je založit aplikaci pomocí předlohy ASP. NET MVC. Aplikace je vytvořena na serverové straně a při změně stavu aplikace se bude celá aplikace znovu načítat. Druhým způsobem je vytvořit MVC aplikaci pomocí dvou dostupných předloh, a to spojení ASP. NET WEB API a ASP. NET Single Page Application. Tento druhý způsob je v dnešní době moderní, protože se aplikace rozdělí na serverovou a klientskou část. Serverovou část bude vytvořena pomocí ASP. NET WEB API, kde v MVC architektuře se bude jednat o Model a Controller. Klientská část bude vytvořena pomocí ASP. NET Single Page Application, kde se v MVC architektuře bude jednat o View. Po změně stavu aplikace se aplikace renderuje pouze na klientské části a serverové části si říká jenom o data.

2.3.1 ASP. NET MVC

ASP. NET MVC je v podstatě framework pro tvorbu webových aplikací pomocí MVC architektury. V roce 2009 byl uvolněn v rámci veřejné licence Microsoftu. Hlavní myšlenka pro vytvoření vzoru MVC je oddělení komponent webové aplikace. Na základě oddělení bude snadnější vývoj a kód půjde lépe upravovat a také testovat. [12] [14]

Některé výhody použití ASP .NET MVC:

- Lepší členění složitých částí aplikace do Model, View a Controller.
- Lepší kontrola nad vykreslenou HTML stránkou
- Lepší kontrola nad HTML zajišťuje lepší vytváření webových standardů.
- Lepší možnost rozšíření stávající aplikace
- Lepší testování aplikace
- Lepší rozdělení práce na aplikaci do většího vývojového týmu.

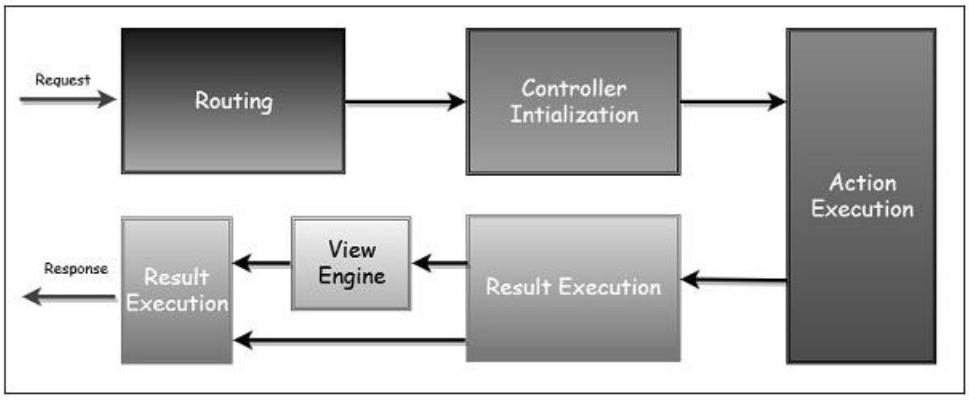
Tato technologie nebyla použita pro vývoj Webového vědomostního kvízu, ale pomocí této technologie se dají vytvářet aplikace založené na architektuře MVC. [12] [14]

2.3.1.1 Životní cyklus

Jedná se o sled událostí, které nastanou při každém požadavku na naší aplikaci.

Prvním bodem při životním cyklu aplikace MVC je příchod požadavku. Na základě požadavku vzniká směrovaní. ASP .NET platforma zjistí, kam by měl být požadavek směrován prostřednictvím směrovacího modulu URL. Po nasměrovaní požadavku se inicializuje Controller, který bude zodpovědný za zpracování požadavku. Po inicializaci následuje vybrání vhodné metody Controlleru. Metoda požadavek zpracuje a dochází k vytvoření výsledku, který

bude předáván zobrazovací enginu, který výsledek zobrazí. Popřípadě je výsledek předán tomu, kdo vyvolal požadavek. Životní cyklus je ilustrován na obrázku 10. [13] [14]



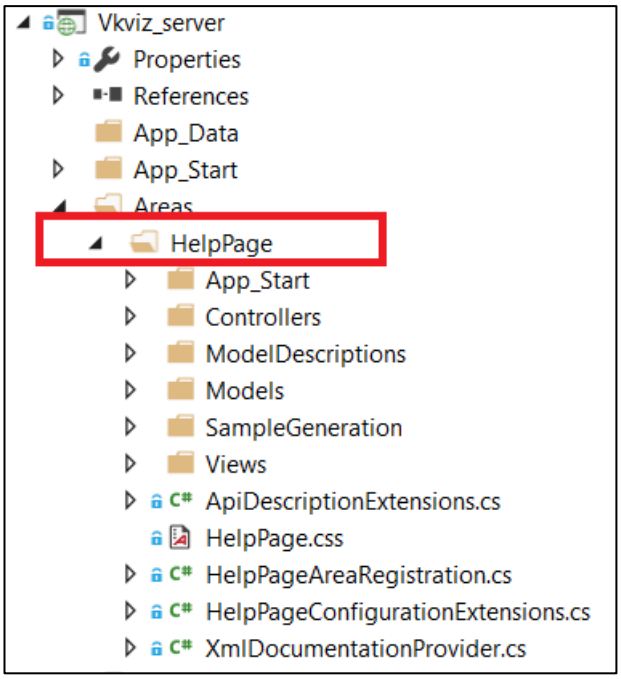
Obrázek 10 - Životní cyklus. [13]

2.3.2 ASP. NET WebAPI

ASP. NET WebAPI je framework, který umožnuje vytvářet webové služby. Pomocí Web API lze vytvářet snadno RESTful aplikace. Slovo API plní svůj význam na sto procent, pomocí vytvořené služby je možné získávat, vytvářet, upravovat a mazat datové objekty, a každá operace bude mít svoji vlastní URI adresu. Při vytvářeni API jsou data přijímána a odesílána ve standartních formátech jako je XML a JSON. [14] [15]

Web API navíc podporuje OData, neboli Open Data Protokol. Je to protokol pro přístup k datům. Poskytuje jednotný způsob komunikace a manipulace s daty. Aplikace přistupuje k datum pomocí CRUD operací. [14] [15]

Web API generuje dokumentaci, kterou je pak možno zobrazit na webu. V adresářové struktuře si vygeneruje složku HelpPage, kde si vytvoří adresářovou strukturu potřebnou k vytvoření dokumentace, viz obrázek 11. Generování API jde samozřejmě potlačit pomocí vlastnosti *[ApiExplorerSettings(IgnoreApi=true)]*. Tuto vlastnost můžeme potlačit bud u celého Controlleru nebo u vybraných akcí. Dokumentaci zobrazíme pod adresou */Help*.



Obrázek 11 - Struktura HelpPage. Zdroj autor

Vytváření REST API je v podstatě jednoduché. Stačí, aby Controller dědil od třídy *ApiController*. Vytvořenému Controlleru přiřadíme anotaci *RoutePrefix*, podle kterého se bude skládat URI adresa. Vytvořeným metodám Controlleru přidáme anotaci typu HTTP požadavku, a také o anotaci *Route*, která také určuje URI adresu. Ve zdrojovém kódu je vytvořen Controller UzivatelController s metodou pro přihlášení uživatele. Výsledná URI adresa pro danou metodu bude vypadat takto: „/api/Uzivatele/Prihlaseni“. Pro úspěšné přihlášení uživatele musí mít v těle HTTP požadavku JSON data, která budou odpovídat struktuře PrihlaseniRequest.

```
[RoutePrefix("api/Uzivatele")]
1 reference | 0 requests
public class UzivateleController : ApiController
{
    Property
    0 references | 0 exceptions
    public UzivateleController(IUzivatelRepository uzivatelRepository, IRoleUzivatelRepository roleUzivatelRepository,
        IRoleRepository roleRepository, IStatistikaRepository statistikaRepository)...

    /// <summary> Přihlášení soutěžícího
    [HttpPost]
    [Route("Prihlaseni")]
    0 references | 0 requests | 0 exceptions
    public string Prihlaseni(PrihlaseniRequest request)
    {
        DataResponse response = new DataResponse();
        var uzivatel = uzivatelRepository.GetSingle(x => x.Email == request.email);
        if (uzivatel != null && request.heslo == uzivatel.Heslo)
        {
            response.Stav = true;
            response.Data = Mapper.Map<UzivatelVM>(uzivatel);
        }
        return JsonConvert.SerializeObject(response);
    }
```

Zdrojový kód 4 - ukázka Controlleru. Zdroj autor

2.4 ASP .NET Single Page Application

SPA je technologie pro vytváření webový aplikací. Aplikace je napsána pomocí HTML, CSS a javascriptového frameworku a se serverem komunikuje pomocí Web API. SPA tedy klade důraz na klientskou část. Při startu aplikace se ze serveru stáhne stránka, a poté se ze serveru stahují pouze data. Navigaci po stránkách si řídí javascript. Na server jsou data posílána ve formátu XML nebo JSON. [16]

Výhody:

- Kombinace nejlepších vlastností z desktopu a webu.
- GUI se nachází na klientské části.
- Okamžitá reakce aplikace.
- Multiplatformní aplikace.
- Mohou fungovat v režimu off-line.

Nevýhody:

- Složitější aplikace kvůli javascriptové vrstvě.
- Javascript je netypový jazyk.

Jak bylo zmíněno, SPA je spojeno s javascriptovým frameworkem. V dnešní době existuje mnoho frameworků mezi nejznámější patří Knockout.js, React, Backbone.js a AngularJS. V aplikaci Webový vědomostní kvíz byla klientská část vytvořena pomocí AngularJS, proto bude v následující podkapitole přiblížen framework AngularJS.

2.4.1 AngularJS

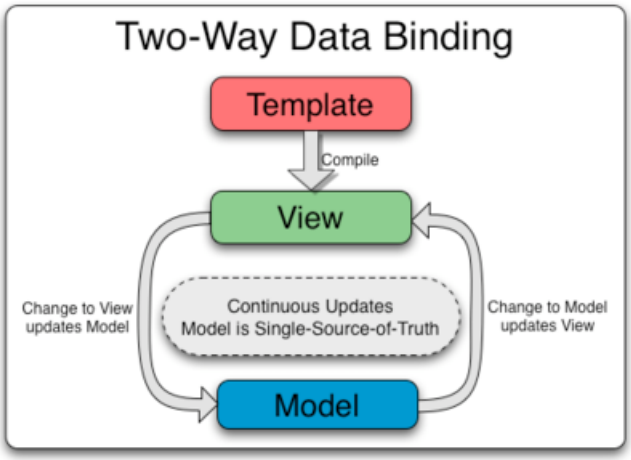
AngularJS je strukturální framework pro tvorbu webových aplikací. AngularJS využívá a rozšiřuje HTML. HTML rozšiřuje o speciální značky, které upravují funkčnost, nebo jsou využity na bindování dat. AngularJS je možné kombinovat s dalšími javascriptovými knihovny, třeba nejznámější jQuery. Mezi nejužitečnější komponenty patří, Two Way Data-binding, implementace Dependency injection, testovatelnost, direktivy a znovu použitelnost komponent. [17] [18]

2.4.1.1 Two Way Data-Binding

Two Way Data-Binding, neboli dvoucestná synchronizace dat, je koncept, který řeší synchronizaci data mezi modelem a view. [17]

V praxi si to můžeme představit tak, že máme HTML element input, kterému přidáme direktivu *ng-model*. Tím mu přiřadíme objekta a když něco zapíšeme do inputu, tak se nám data nahrají do modelu. To je první cesta. Pod inputem budeme mít element *span*, kterému přiřadíme direktivu *ng-bind*, a když už data máme v modelu, tak se přenesou do view. To je druhá cesta synchronizace. [18]

Na obrázku 12 je znázorněná synchronizace. Template, neboli šablona, znázorňuje HTML stránku. Model znázorňuje nějaký objekt a view vznikne při vložení dat do šablony.



Obrázek 12 - Two-way data binding. [18]

2.4.1.2 Dependency Injection

Dependency injection je návrhový vzor řešící závislosti mezi komponentami programu. „*AngularJS obsahuje zabudovaný subsystém, který řeší DI napříč celou vyvíjenou aplikací. Umožňuje vždy přesně specifikovat, které jiné komponenty jsou uvnitř konkrétní komponenty používané.*“ [18]

Ve zdrojovém kódu 5 se Controleru předávají 4 parametry, které jsou předány přes závislosti. Při práci s parametry uvnitř Controlleru se nemusí starat, jak vznikly. Angular si sám zkontroluje závislosti v deklaraci, a poté je předá jako parametr. Díky dependency injection jsou aplikace dobře testovatelné.

```
app.controller('VytvorKategoriiController', function ($scope, $uibModalInstance, $location,
                $rootScope, otazkaService) {
```

Zdrojový kód 5 - Dependency Injection. Zdroj autor

2.4.1.3 Direktivy

Direktivy učí HTML nové elementy a atributy. Angular umožnuje vytvářet nové elementy a atributy, ale také je možné použít direktivy, které jsou součástí Angularu. Nyní si představíme důležité direktivy, které se zapisují jako atribut elementu. [18]

Nejdůležitější direktiva je *ngApp,* kterou přidáme do elementu *html*. Direktiva nám spouští celou aplikaci a jako parametr je zapsán název hlavního modulu aplikace. [18]

Direktivu *ngController* používáme pro předání scope, neboli Contextu, controlleru. Ve vnořených elementech pak je možné přistupovat k objektům daného Controlleru. [18]

Další důležitou direktivou je *ngModel,* kterou využíváme pro Two way data binding. Tuto direktivu přiřazujeme elementům, které chceme navázat na nějaký objekt. [18]

Direktiva *ngRepeat* nám slouží jako cyklus. Direktiva projde pole, které ji přiřadíme, a opakuje element, ke kterému se váže. Uvnitř elementu vytváří speciální proměnné:

- Proměnnou $index určující aktuální index pole.
- Proměnná $first je nastavena na *true*, pokud se jedná o první prvek pole
- Proměnná $middle je nastavena na *true*, pokud se jedná o prvek mezi prvním a posledním prvkem.
- Proměnná $last je nastavena na *true*, pokud se jedná o poslední prvek pole.
- Proměnná $even je nastavena na *true*, pokud se jedná o sudý prvek pole.
- Proměnná $odd je nastavena na *true*, pokud se jedná o lichý prvek pole.

Při vytváření vlastní direktivy definujeme několik konfiguračních vlastních:

- restrict – určuje v jakém kontextu se bude direktiva zobrazovat:
  - E – element,
  - A – atribut,
  - C – třída,
  - M – komentář,
- replace – určuje, zdá se má direktiva zaměnit za HTML kód,
- scope – určuje kontext direktivy,
- template – určuje HTML, které direktiva nahrazuje,

Ve zdrojových kódech 6 a 7 je příklad direktivy „můj zákazník“. Zdrojový kód 6 je napsaný v javascriptu a zdrojová kód 7 je element, který se použije v HTML kódu. [18]

```
angular.module('docsIsolateScopeDirective', [])
.controller('Controller', ['$scope', function($scope) {
  $scope.naomi = { name: 'Naomi', address: '1600 Amphitheatre' };
  $scope.igor = { name: 'Igor', address: '123 Somewhere' };
}])
.directive('myCustomer', function() {
  return {
    restrict: 'E',
    scope: {
      customerInfo: '=info'
    },
    templateUrl: 'my-customer-iso.html'
  };
});
```

Zdrojový kód 6 - vytvoření direktivy. [19]

```
<div ng-controller="Controller">
  <my-customer info="naomi"></my-customer>
  <hr>
  <my-customer info="igor"></my-customer>
</div>
```

Zdrojový kód 7 - Direktiva v HTML kódu. [19]

2.5 ASP .NET SignalR

SignalR je knihovna, která slouží k rozšíření komunikace webových aplikací, a to o komunikaci v reálném čase. V případě, že server přijme data, nečeká na klienta, až si o data řekne, ale hned mu je pošle. Knihovna umožnuje vytvořit jakoukoliv webovou aplikaci, která je napsaná pomocí .NET technologie. Pro klientskou část je k dispozici SignalR Javascript client, který jde stáhnou jako jQuery knihovna. Real-time aplikace jsou aplikace, které je potřeba aktualizovat v reálném čase. Nejčastější příkladem je obyčejný chat, kde server rozesílá připojený klientům zprávy, které přijme od jednoho z připojených klientů. Dalšími aplikacemi mohou být aplikace sloužící k monitorování, hry pro více hráčů a spoustu další aplikací. [20]

SignalR poskytuje jednoduché API, které podporuje vzdálené volání procedur, tzn. že klient může volat metody vytvořené na serveru a naopak. Poté co se klient připojí k serveru, je uložen do skupiny, které jsou od sebe vzájemně rozlišitelné pomocí identifikátorů. Na základě identifikátoru skupiny lze pak odsílat data všem členům skupiny. Zprávu lze také poslat pouze konkrétnímu uživateli. [20]

Spojení mezi serverem a klientem je trvalé, na rozdíl od protokolu HTTP. SignalR je abstrakce přes protokoly, které jsou nutné pro komunikaci v reálném časem. Spojení začíná pomocí HTTP protokolu, poté se využívá protokol, který je k dispozici pro komunikaci v reálném čase.

Vybírá se mezi protokoly, jako jsou WebSocket, Server Sent Events, Forever Frame a Ajax long polling. [20]

WebSocket je jediný typ spojení, který vytváří skutečné trvalé obousměrné spojení. Vyžaduje nejpřísnější požadavky, které jsou pouze v nejnovějších webových prohlížečích. Prohlížeč musí podporovat HTML 5, operační systém vyšší jak Windows 8 a Windows Server 2012. Verze .Net frameworku je vyžadována 4.5 a vyšší. WebSocket je ideální spojení, protože nejefektivněji využívá paměť serveru. Má nejnižší latenci. [20]

Server Sent Events je typ jednosměrného spojení, kde události jdou ze serveru na klienta. Pokud má server data a klient je připojen, jsou mu zaslána data. Příchozí události jsou obsluhovány prohlížečem. Výhodou oproti WebSocket je, že Server Sent Events nepotřebuju podporu na klientské straně. Server Sent Events jsou podporované od HTML 5. [20]

Forever Frame je jednosměrné spojení podporované pouze u prohlížeče Internet Explorer. Forever Frame vytvoří skrytý iframe, který se na serveru nedokončí. Server tak neustále posílá script, který je okamžitě proveden, tím je poskytnuto jednosměrné spojení ze serveru na klienta. Při spojení z klienta na server se vytváří samostatné spojení. [20]

Ajax long polling nevytváří trvale spojení, ale neustále dotazuje server s požadavkem, který zůstává otevřen, dokud server neodpoví, poté se znovu opakuje dotazování na server. To může vytvářet zpoždění, než se spojení obnoví. [20]

Pro výběr spojení je použit následující scénář:

1. Pokud je použit webový prohlížeč Internet Explorer 8 a starší je použito spojení typu **LongPolling**.
2. Pokud je při způsobu komunikace nastaven *jsonp* na hodnotu *true*, je využíváno **LongPolling**.
3. Pokud je spojení mezi různými domény a klient podporuje toto spojení, a také klient i server podporují **WebSocket** je použit typ spojení pomocí **Websocket**. V případě nesplnění je použit **Long Polling**.
4. Pokud není spojení mezi různými domény a není nastaven *jsonp* na *true* je použit typ spojení pomocí **Websocket**, pokud ho samozřejmě klient a server podporují.
5. Pokud není podporováno spojení **Websocket** a je k dispozici **Server Sent Events** je tento typ spojení použit.
6. Pokud **Server Sent Events** není k dispozici, je použit **Forever Frame.**

7. Pokud není k dispozici **Forever Frame**, je použit **Long Polling.**

Velkou výhodou knihovny SignalR je, že se programátor nemusí starat o způsob spojení a je vždy vybráno to nejlepší spojení podle předchozího scénáře. [20]

3 WEBOVÉ SLUŽBY

K modernímu vývoji aplikací bezpochyby patří použití webových služeb. Webová služba je technologie, která spojuje platformově nezávislé informační systémy a umožnuje jim mezi sebou komunikovat. Pro komunikaci mezi sebou používají protokol HTTP. Přenášená data se nejčastěji serializují do značkovacího jazyka XML nebo do javascriptového zápisu objektů JSON. [22]

Příklad uložení zaměstnanců do XML, viz zdrojový kód 8, a do JSON, viz zdrojový kód 9. Z příkladu je patrný menší přenos dat, který probíhá pomocí JSON.

```xml
<employees>
    <employee>
        <firstName>John</firstName> <lastName>Doe</lastName>
    </employee>
    <employee>
        <firstName>Anna</firstName> <lastName>Smith</lastName>
    </employee>
    <employee>
        <firstName>Peter</firstName> <lastName>Jones</lastName>
    </employee>
</employees>
```

Zdrojový kód 8 - Ukázka XML. [21]

```json
{"employees":[
    { "firstName":"John", "lastName":"Doe" },
    { "firstName":"Anna", "lastName":"Smith" },
    { "firstName":"Peter", "lastName":"Jones" }
]}
```

Zdrojový kód 9 - Ukázka JSON. [21]

V dnešní době existují dva typy webových služeb, a to SOAP a REST.

3.1 SOAP

SOAP, neboli Simple Object Access Protocol, je protokol sloužící pro výměnu dat mezi počítači nejčastěji prostřednictvím HTTP protokolu. Díky složení SOAP zprávy se nejčastěji data přenáší pomocí XML. SOAP může být využit k přenášení dat, ale také k volání procedur. Díky komunikaci pomocí XML je platformově a jazykově nezávislý. Pomocí XML můžeme definovat, jak mají data vypadat. [23]

SOAP zpráva se skládá ze čtyř prvků: Envelope, Header, Body a Fault. Ve zdrojovém kódu 10 je vytvořena základní struktura SOAP zprávy. [23]

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2001/12/soap-envelope"
      SOAP-ENV:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

   <SOAP-ENV:Header>
      ...
      ...
   </SOAP-ENV:Header>


   <SOAP-ENV:Body>
      ...
      ...
      <SOAP-ENV:Fault>
         ...
         ...
      </SOAP-ENV:Fault>
      ...
   </SOAP-ENV:Body>


</SOAP_ENV:Envelope>
```

Zdrojový kód 10 - SOAP zprava. [23]

Envelope, neboli obálka, definuje začátek a konec zprávy, díky tomu příjemce pozná, kdy byla zpráva přijata. Každá SOAP zpráva musí obsahovat prvek Envelop. V tomto prvku je nula a více prvků typů Header a vždy jeden prvek typu Body. Pokud se změní SOAP verze, mění se i obálka. Obálka je určena pomocí prefixu ENV. [23]

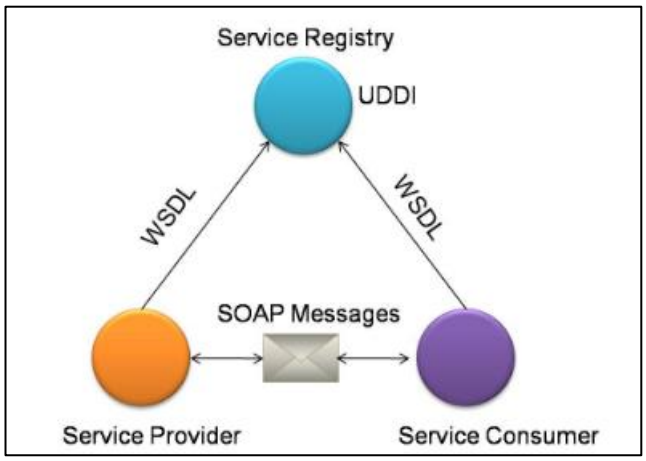
Header, neboli hlavička, je určena pro zadávání dodatečných požadavků. Je to první prvek obálky a je nepovinný, ale může být v obálce definovaný vícekrát. [23]

Body, neboli tělo, je povinný prvek a obsahuje data. Prvek je podřízený prvku obálky a musí dodržet všechny definované hlavičky. Sémantika těla je popsána pomocí SOAP schématu. [23]

Fault, neboli chyba, obsahuje informace o chybách, které vzniknou při zpracování zprávy. Je to volitelný prvek, ale může být obsažen pouze jednou. [23]

SOAP je protokol, který ke své funkci potřebuje další dvě technologie, a těmi jsou WSDL a UDDI. WSDL, neboli Web Services Description Language, je jazyk popisující rozhraní webové služby. Pomocí této služby je možné vygenerovat SOAP clienta, který bude komunikovat s webovou službou. UDDI, neboli Unversal Description, Discovery and Integration, je registr, který obsahuje seznam webových služeb. [23] [24]

Na obrázku 13 je znázorněný vztah mezi technologiemi.



Obrázek 13 - Webové technologie. [24]

3.2 REST

REST, neboli Representational State Transfer, navrhl v roce 2000 Roy Fielding v rámci své disertační práce. Jedná se o architekturu, která umožnuje přistupovat ke zdrojům pomocí protokolu HTTP. Pod pojmem zdroj si můžeme představit data nebo stav aplikace. Jednotlivý přístup ke zdrojům je identifikován pomocí vlastní URI adresy. REST je tedy orientován datově a proto implementuje čtyři základní metody pro přístup k datům. Metody se označují zkratkou CRUD, neboli, create, retrieve, update, delete. [25]

Tabulka 2 - CRUD operace. Zdroj autor

Metoda	HTTP metoda
Create	POST
Retrive	GET
Update	PUT
Delete	DELETE

Pro použití REST musíte splňovat následující čtyři základní principy:

- používat HTTP metody,
- být bezestavový,
- jedinečnost zdrojů identifikovatelnou pomocí URI,
- data převádět do XML, nebo JSON.

Create (POST) je Metoda, která se používá pro vytvoření dat. Pro každého programátora známá minimálně pod uložením dat z nějakého formuláře. Ukládaná data jsou uložena v těle metody http požadavku.

Retrive (GET) je metoda, která se používá pro přístup k datům. Pomocí této metody, můžeme stáhnout třeba celý seznam dat, nebo konkrétní položky. Vše je určeno pomocí URI.

Update (PUT) je metoda, která se používá pro změnu dat. Tato metoda je podobná metodě POST, ale liší se v URI, kde uvádíme konkrétní identifikátor zdroje, který chceme změnit. V těle metody je předáván nová hodnota zdroje.

Delete (DELETE) je metoda, která se používá pro smazání dat. Volání této metody je podobné jako u metody GET. Tato metoda je v praxi často nahrazována metodou POST, kde v těle metody udáváme konkrétní zdroj, který chceme smazat.

4 ANALÝZA WEBOVÉHO VĚDOMOSTNÍHO KVÍZU

4.1 Nefunkční požadavky

Systém by měl běžet na dedikovaném serveru, aby vydržel vetší nápor uživatelů. V jednu chvíli je možné soutěžit v několika kvízech, kde v jednom kvízu může soutěžit více soutěžících, a proto je důležitá stabilita serveru. V případě výpadku mmusí být dostupná záloha.

Na serveru by měl být nainstalovaný operační systém Windows server 2012 a na desktopovém zařízení minimálně Windows 8. Na serveru i desktopu by měl být dostupný framework .NET s minimální verzí 4.5. Webové prohlížeče by měli podporovat HTML 5. Tyto minimální požadavky jsou potřeba pro použiti protokolu WebSocket. Vzhledem k velké dostupnosti webových prohlížečů by měla být zajištěna maximální nezávislost aplikace na použitém webovém prohlížeči.

Jelikož se aplikace skládá z více částí, je nutné předpokládat, že se aplikace může v budoucnu měnit. Úprava jedné části aplikace by neměla zasahovat do dalších částí aplikace, proto by aplikace měla být dobře rozšířitelná.

4.2 Funkční požadavky

Mezi funkční požadavky patří:

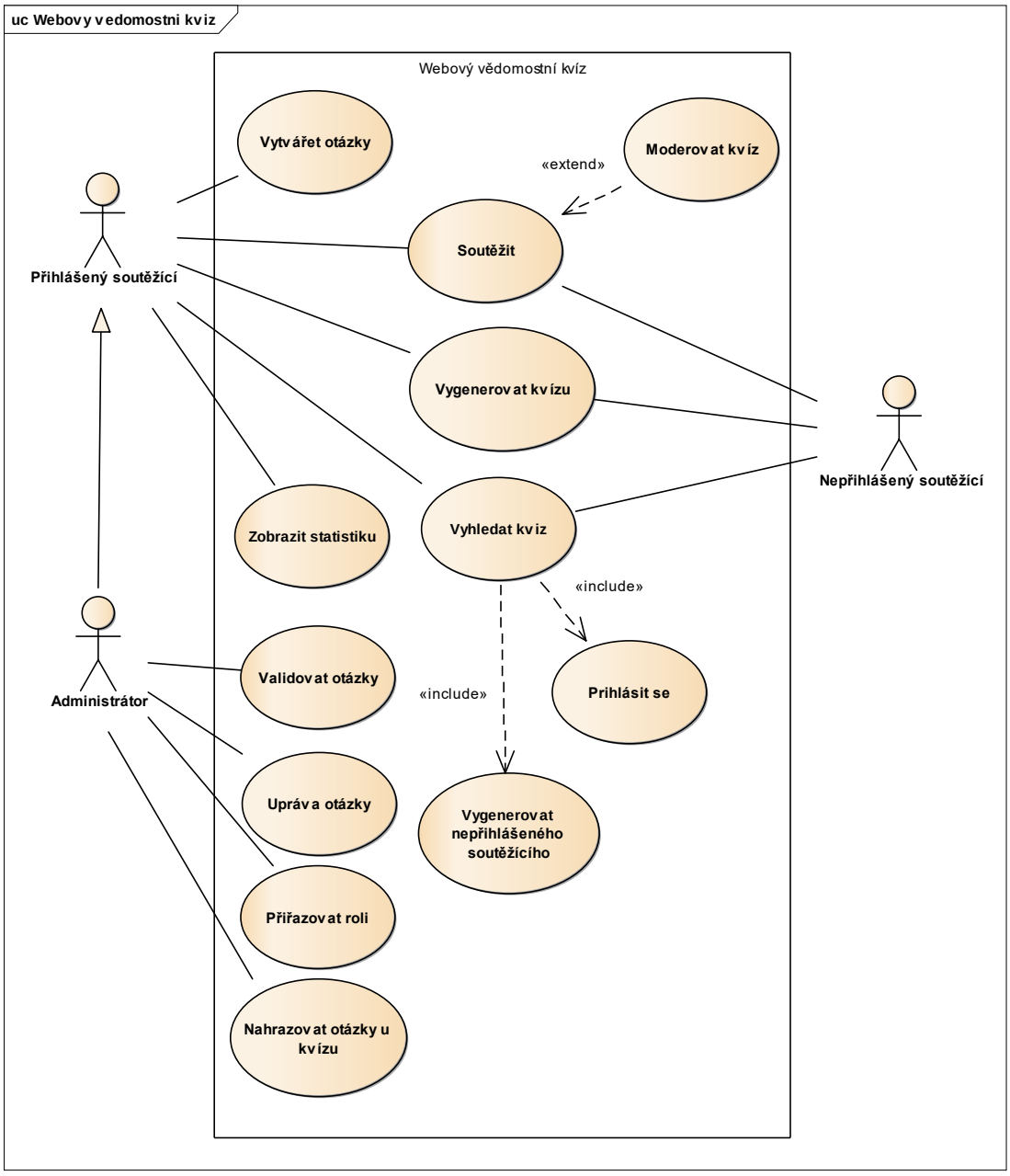
- Možnost nepřihlášeného soutěžícího – aplikace umožní vytvářet nepřihlášeného soutěžícího. Takovému soutěžícímu nebudou vytvořeny žádné statistiky. Také nebude mít možnost znovu se do aplikace přihlásit.
- Vytváření otázek – aplikace umožní vytvářet otázky podle definovaných kategorii a podle vybraných typů otázek. Aplikace umožní vytvořit čtyři typy otázek:
    - Jedna správná - Otázka bude obsahovat 4 odpovědi, kde je jenom jedna správná odpověď.
    - Seřaď odpovědi – Odpovědi budou seřazeny podle pořadí vyplývajícího ze znění otázky.
    - Ano x Ne – Otázka obsahuje odpověď „Ano“ a „Ne“, kde je jedna z možností správná.
    - Zadej hodnotu – Odpověď na otázku bude v podobě zadání číselné hodnoty. Nejbližší hodnota z dostupných odpovědí, které zadají soutěžící, je správná.
- Generování kvízu – aplikace umožní generovat kvíz z konfiguračních parametrů název kvízu, typ kvízu, počet soutěžících, počet otázek, typ kategorie a typ odpovědí. Na

základě těchto parametrů se vygeneruje kvíz, kde otázky přiřazené kvízu budou vybrány náhodně podle typu kategorie a typu otázek.

- Upravování otázek – aplikace nabídne možnost zobrazit schválené a neschválené otázky. Znění otázky a odpovědí bude možné upravovat.
- Validování otázek – aplikace umožní validovat otázky. Přihlášený soutěžící bude mít možnost vytvořit otázku, která nebude schválena do té doby, než ji zkontroluje administrátor. Po kontrole administrátor označí otázku jako schválenou.
- Nahrazení otázek kvízu – aplikace umožní nahrazení otázek u vygenerovaného kvízu. Otázku bude možné vyměnit za otázku podle parametru kvízu.
- Přiřazení role – aplikace umožní přiřadit přihlášenému soutěžícímu roli administrátora. Role bude přiřazovat pouze administrátor.
- Zobrazení statistik – Přihlášený soutěžící bude mít možnost zobrazení statistik. Do statistik se započítává počet výher a proher v jednotlivých kvízech a počet správných a špatných odpovědí.
- Vybrání typu kvízu – aplikace umožní vygenerovat kvíz na základě dvou typů kvízu:
    - Omezený čas na odpověď - Aplikace umožní vygenerovat typ kvízu, který bude mít omezený čas na odpověď. Soutěžící bude muset v daném časovém intervalu odpovědět na zadanou otázku. Po vypršení časového intervalu bude zobrazena další otázka.
    - Neomezený čas na odpověď - Aplikace umožní vygenerovat typ kvízu, který bude mít neomezený čas na odpověď. Soutěžící nebude omezený časovým intervalem na zodpovězení otázku. Moderátor hry bude přepínat otázky. U každé zobrazené otázky bude zobrazeno počet soutěžících, kteří už na otázku odpověděli. Poté co soutěžící odpoví, bude čekat na zobrazení další otázky.
- Vyhodnocení hry – aplikace umožní zobrazit vyhodnocení daného kvízu, kde bude zobrazeno každému soutěžícímu počet správných a špatných odpovědí. Za každou správnou odpověď získá soutěžící jeden bod. Na základě bodů bude určen vítěz kvízu.

4.3 UML UseCaseDiagram

V aplikaci jsou dostupné tři role uživatelů administrátor, přihlášený soutěžící a nepřihlášený soutěžící. Každá role má jiné možnosti využití aplikace. Na obrázku 14 jsou tyto možnosti zobrazeny.



Obrázek 14 - UseCase diagram. Zdroj autor

4.4 UML Activity Diagram

V následujících podkapitolách bude popsáno základní chování aplikace, a to průběh kvízu. V aplikaci je možno soutěžit ve dvou typech kvízu, a to s omezeným a neomezeným časem na odpověď.

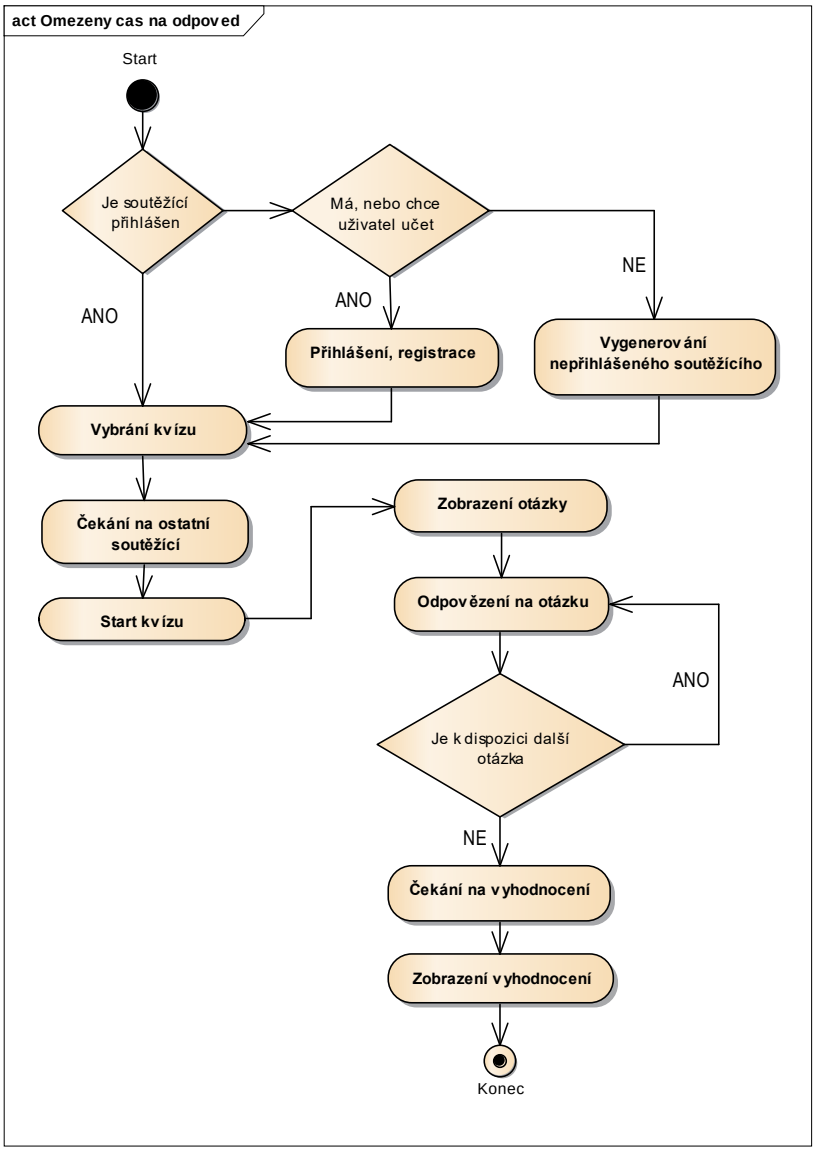
Při pokusu o vyhledání kvízu aplikace zkontroluje, zda je přihlášený nějaký soutěžící. Pokud soutěžící není přihlášen je zobrazeno modální okno. V modálním okně je na výběr ze tří možností přihlásit se, zaregistrovat se, nebo vygenerovat nepřihlášeného soutěžícího. Po

přihlášení do aplikace si soutěžící vybere kvíz podle vytvořených typů otázek a vědomostních kategorií.

Soutěžící čeká, než se ostatní soutěžící připojí. Všichni připojení soutěžící musí potvrdit, že jsou připravení soutěžit. Od tohoto bodu se chování aplikace liší, takže popis každého typu kvízu bude pokračovat v následujících podkapitolách.

4.4.1 Omezený čas na odpověď

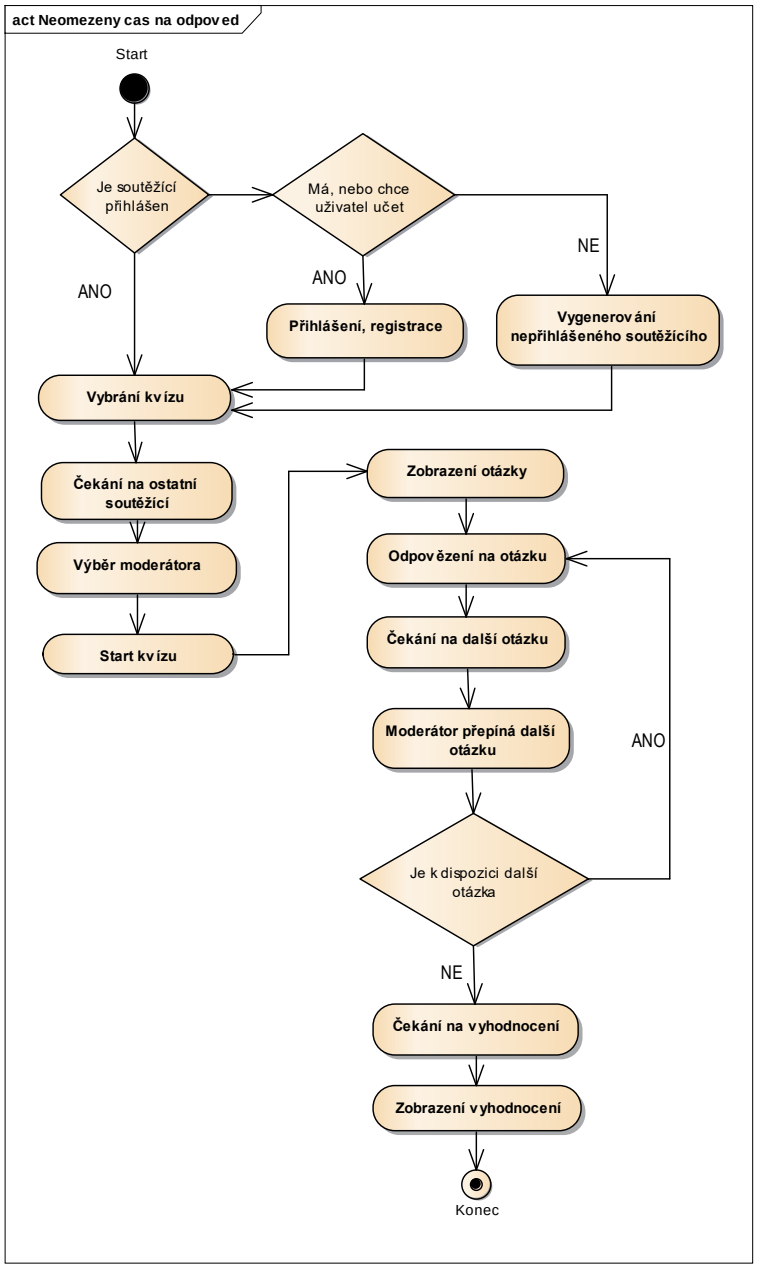
V případě že jsou všichni soutěžící připravení soutěžit, je možné kviz spustit. Spustit kvíz může libovolný soutěžící. Po spuštění je všem soutěžícím zobrazena první otázka. Po zodpovězení otázky jsou zobrazeny další otázky. Přepínání otázek je čistě na soutěžícím a každý si otázky přepíná podle sebe, dokud nevyprší časový limit na odpověď, poté je automaticky otázka změněna. Jakmile jsou zobrazeny všechny otázky, soutěžící čeká, než všichni dokončí kvíz. Po dokončení kvízu posledním soutěžícím je zobrazeno vyhodnocení hry.



Obrázek 15 - Activity diagram. Zdroj autor

4.4.2 Neomezený čas na odpověď

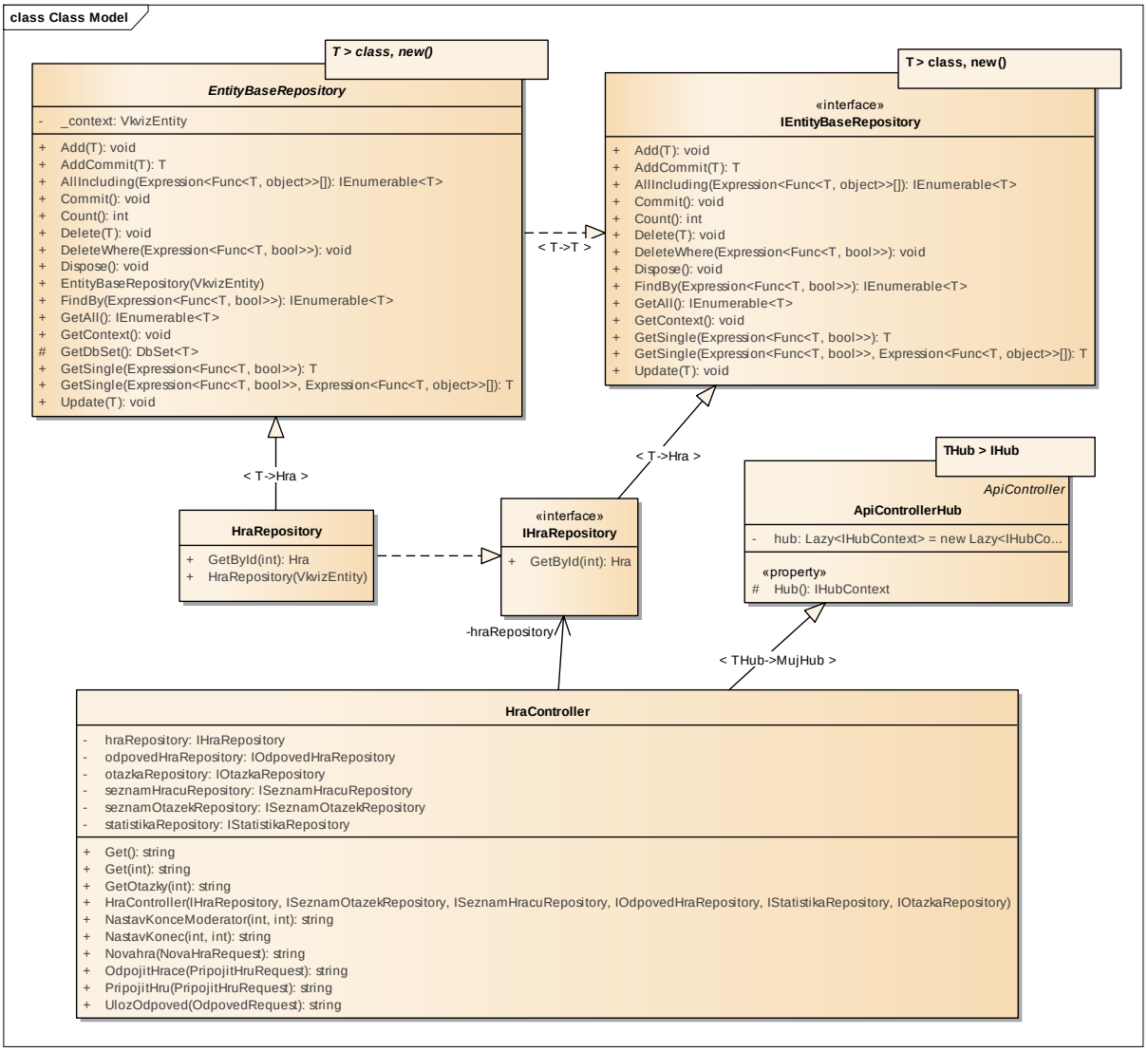
V případě že všichni soutěžící potvrdí svoji připravenost, musí být zvolen ještě jeden moderátor hry. Poté je možné kvíz spustit. Moderátor nesoutěží, ale pouze přepíná zobrazené otázky. Po spuštění je všem soutěžícím zobrazena první otázka. Po zodpovězení otázky soutěžící čeká, než všichni odpoví. Moderátorovi je zobrazeno počet soutěžících, kteří už odpověděli. Je pouze na jeho rozhodnutí, zda počká, než všichni odpoví. Moderátor přepne otázku a všem se zobrazí nová otázka. Jakmile jsou zobrazeny všechny otázky, soutěžící čeká, než moderátor ukončí kvíz a poté je zobrazeno vyhodnocení hry. Moderátor se do výsledného hodnocení samozřejmě nezapočítává, protože na otázky nemůže odpovědět.



Obrázek 16 - Activity diagram. Zdroj autor

4.5 UML Diagram tříd

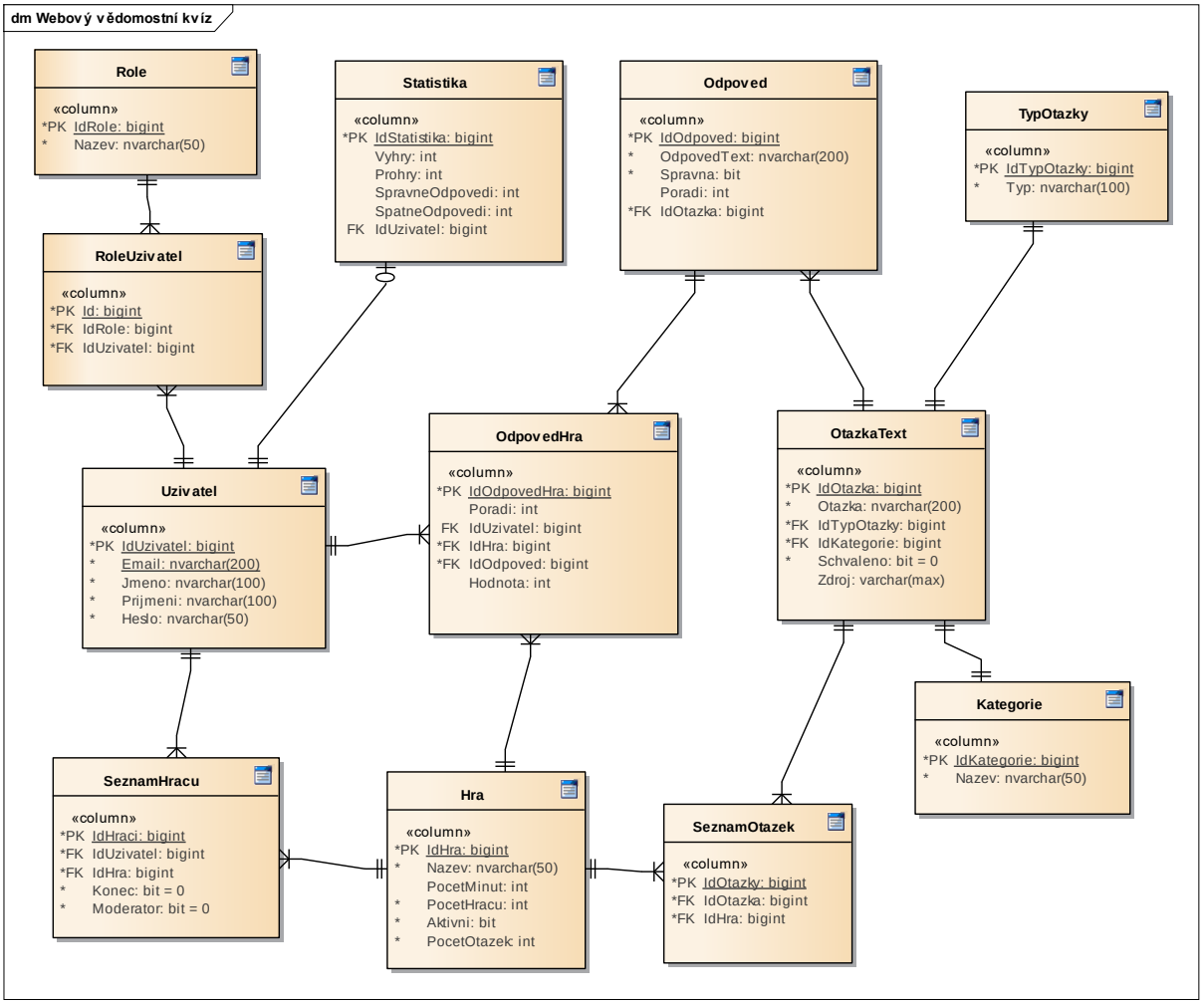
Vzhledem ke složitosti diagramu tříd, je na obrázku 17 zobrazen výřez diagramu. Třída HraController neobsahuje třídy, které jsou ve stejném vztahu jako IHraRepository a HraRepository.



Obrázek 17 - Diagram tříd. Zdroj autor

Celá struktura diagramu tříd je dostupná v přiloženém CD disku.

4.6 Návrh databáze



Obrázek 18 - Návrh databáze. Zdroj autor

Tabulka Uzivatel slouží k uložení uživatelů. Tabulka obsahuje základní údaje jméno, příjmení, email a heslo. Další údaje nejsou důležité. Ukládají se zde administrátoři, přihlášení a nepřihlášení soutěžící. Nepřihlášeným soutěžícím se automaticky vygenerují základní údaje. Pro tabulky z hlediska uživatelských rolí jsou důležité tabulky RoleUzivatel a Role, který slouží pro definování uživatelských rolí.

Tabulka Statistika slouží k ukládání statistik. Statistika je vytvořena pouze pro přihlášeného soutěžícího. Nepřihlášenému uživateli není, žádná statistika vytvořena. Mezi sledované statistiky patří počet výher a proher v jednotlivém kvízu a počet správných a špatných odpovědí.

Tabulka Hra slouží k uložení vygenerovaných kvízů. U každého kvízu lze nastavit jeho název, počet hráčů, kteří se mohou připojit a počet otázek. Počet otázek se generuje náhodně a žádná otázka se nesmí opakovat. Dále lze nastavit počet minut, za který je nutné na otázku odpovědět.

V případě typu kvízu Neomezený čas na odpověď je v této položce uložena hodnota 0. Poslední atribut, se nazývá Aktivní a slouží k identifikaci, zda je možné se k vygenerovanému kvízu připojit. Po spuštění kvízu je hodnota nastavena na 0 a po jeho ukončení je znovu atribut nastaven na hodnotu hodnotou 1.

Tabulka SeznamHracu slouží jako vazební tabulka mezi tabulkami Uzivatel a Hra. Kromě cizích klíčů obsahuje tabulka navíc dva atributy Konec a Moderator. Atribut Konec slouží k identifikaci, zda soutěžící dokončil spuštěný kvíz. Pokud kvíz ukončil, je nastavena hodnota 1, jinak je defaultně nastavena hodnota 0. Atribut Moderator slouží k identifikaci, zda připojený soutěžící bude kvíz moderovat. V případě moderovaní je nastavena hodnota 1, jinak hodnota 0.

Tabulka Otazka slouží k uložení otázek. Atribut Schvaleno slouží k identifikaci, zda je otázka schválena administrátorem. Otázku může vytvořit přihlášený soutěžící, ale u vytvořené otázky je atribut Schvaleno nastaven na 0. Dále je u otázky možné udat zdroj, ze kterého bylo při jejím vytváření čerpáno. Tabulka obsahuje vazbu na tabulku TypOtazky, která slouží k identifikování typu otázky. Poslední vazba je na tabulku Kategorie, která slouží k ukládání kategorií dané otázky.

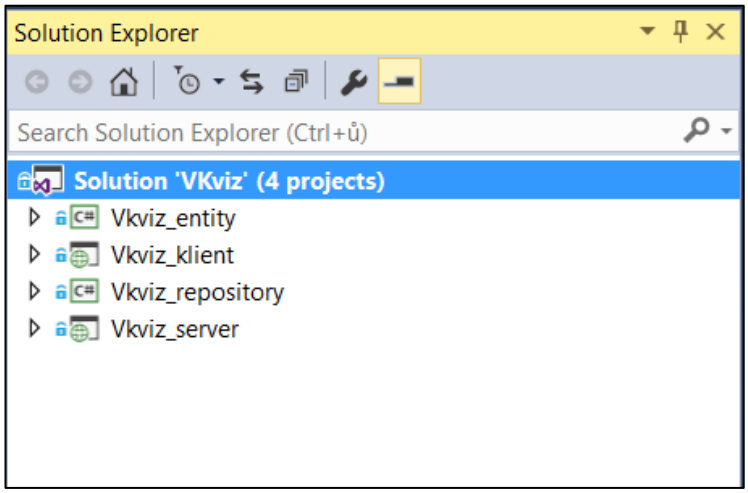
Tabulka Odpoved slouží k uložení odpovědí pro danou otázku. Do této tabulky je možné uložit všechny odpovědi podle typu otázky, proto některé atributy nabývají hodnot NULL. Správnou odpověď určuje atribut Spravna. Atribut Poradi slouží k uložení pořadí v případě, že se jedná o otázku typu „Seřaď Odpovědi“.

Poslední důležitá tabulka OdpovedHra slouží k ukládání odpovědí na daný kvíz uživatelem. Do této tabulky lze ukládat všechny odpovědi daných typů otázek.

5 IMLEMENTACE WEBOVÉHO VĚDOMOSTNÍHO KVÍZU

5.1 Rozdělení práce

Webový vědomostní kvíz je rozdělen do dvou knihoven Vkviz_entity a VKviz_repository, a také do dvou projektů Vkviz_server a Vkviz_klient, viz obrázek 19. Práce je rozdělena z důvodu možných implementačních změn v budoucnosti. V případě nějaké úpravy lze jednoduše danou část upravit bez nutnosti zásahu do ostatních projektů.



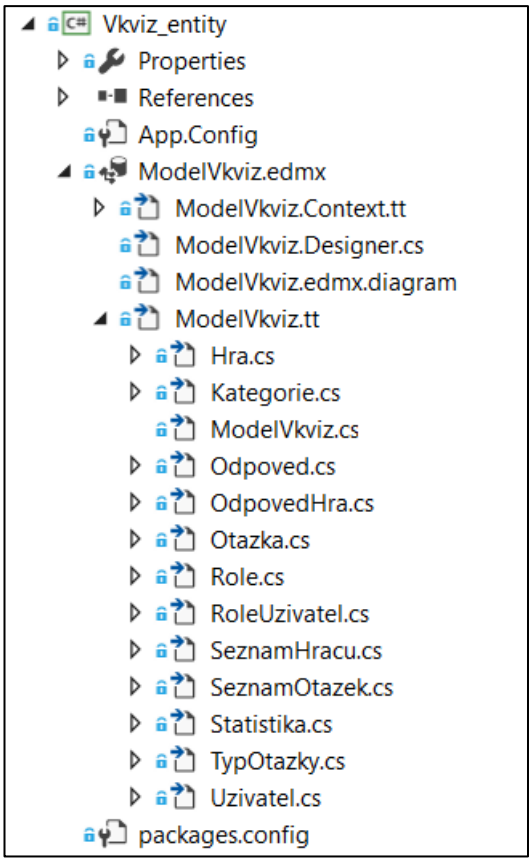
Obrázek 19 - Rozdělení do projektů. Zdroj autor

V příloze jsou verze knihoven, které byly v projektu použity.

5.2 Vkviz_entity

V knihovně VkvizEntity je vytvořeno objektové mapovaní relační databáze. Pro mapování byl využit Entity framework. Pro každou tabulku z databáze je vytvořena odpovídající objektová entita. Datová entita je vytvořena jako Dynamic Proxy třída. Kromě datových entit je v projektu vytvořena třída VkvizEntity, která dědí od DbContexu. Tato třída slouží k vytvoření databázového kontextu, pomocí kterého se připojíme k databázi. Tato třída dále obsahuje DBSety vytvořených entit, které jsou automaticky inicializovány při vytvoření instance třídy. Při vytvoření objektového mapovaní databáze je také vytvořen diagram entit, který je součástí projektu. V tomto projektu nebylo dále nic jiného implementováno. Projekt se soustředí pouze na objektové mapování databáze. Díky použití objektového přístupu je odstraněna možnost vzniku syntaktické chyby, protože SQL dotazy nejsou psány ručně.

Struktura projektu je zobrazena na obrázku 20.



Obrázek 20 - Projekt Vkvíz_entity. Zdroj autor

5.3 Vklient_repository

Knihovna Vklient_repository je vytvořena pro komunikaci s daty mezi databázovou částí, tedy knihovnou Vkviz_entity a bussines logikou serveru, tedy projektem Vkviz_server. Projekt obsahuje rozhraní, které slouží k definovaní vlastností, pomocí kterých můžeme komunikovat s databází. Pro komunikaci s databází bylo vytvořeno generické rozhraní, které obsahuje základní metody jako získaní všech dat, vyhledaní dat podle podmínky, uložení dat, smazání dat, úprava dat a další metody. Ve zdrojovém kódu11 je znázorněná část generického rozhraní.

```
public interface IEntityBaseRepository<T> where T : class, new()
{
    10 references | 0 exceptions
    IEnumerable<T> AllIncluding(params Expression<Func<T, object>>[] includeProperties);
    3 references | 0 exceptions
    IEnumerable<T> GetAll();
    1 reference | 0 exceptions
    int Count();
    9 references | 0 exceptions
    T GetSingle(Expression<Func<T, bool>> predicate);
    1 reference | 0 exceptions
    T GetSingle(Expression<Func<T, bool>> predicate, params Expression<Func<T, object>>[] includeProperties);
    7 references | 0 exceptions
    IEnumerable<T> FindBy(Expression<Func<T, bool>> predicate);
    7 references | 0 exceptions
    void Add(T entity);
    17 references | 0 exceptions
    T AddCommit(T entity);
    6 references | 0 exceptions
    void Update(T entity);
    9 references | 0 exceptions
    void Delete(T entity);
```

Zdrojový kód 11 - Generické rozhraní. Zdroj autor

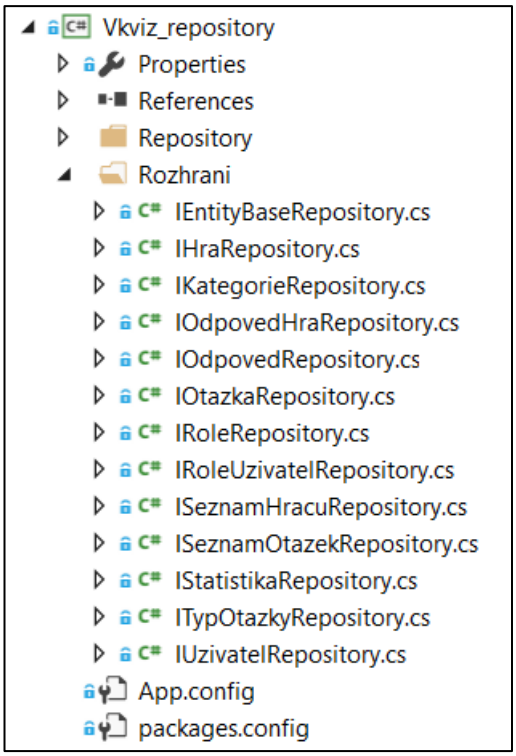
Obecné generické rozhraní implementují jednotlivé rozhraní pro dané databázové entity. V konkrétním rozhraním jsou definovány pouze metody potřebné pro daný objekt. Jednotlivá rozhraní jsou implementována pomocí tříd uložených ve složce Repository, kde rozhraní IHraRepository implementuje třída HraRepository. Třída také dědí od abstraktní generické třídy EntityBaseRepository. která implementuje rozhraní IEntityBaseRepository. Ve zdrojovém kódu 12 je zobrazena třída HraRepository.

```
public class HraRepository : EntityBaseRepository<Hra>, IHraRepository
{
    0 references | 0 exceptions
    public HraRepository(VkvizEntity context)
    : base(context)
    {
    }

    1 reference | 0 exceptions
    public Hra GetById(int id)
    {
        return this.GetDbSet().FirstOrDefault(x => x.IdHra == id);
    }
```

Zdrojový kód 12 - Třída HraRepository. Zdroj autor

Na obrázku 21 je zobrazena struktura projektu. Ve složce rozhraní jsou definovány rozhraní pro jednotlivé databázové entity a ve složce Repository jsou vytvořeny jejich implementace.



Obrázek 21 - Struktura projektu Vkviz_repository. Zdroj autor

5.4 Vkviz_server

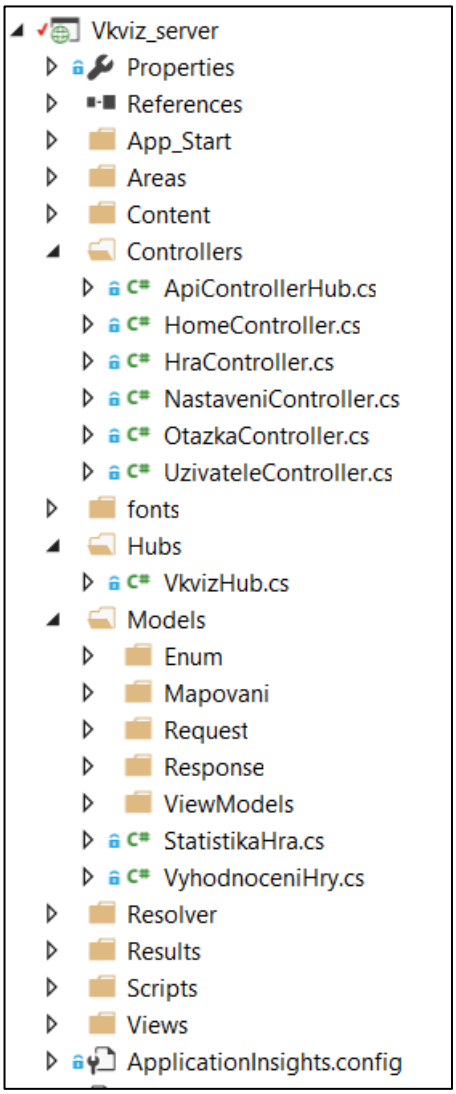
V projektu Vkviz_server je vytvořena webová služba typu REST. Webová služba slouží k odesílání a přijímání dat z klienta. Je vytvořena pomocí technologie ASP.Net Web API. Projekt také obsahuje další funkce, které rozšiřují webovou službu.

Struktura projektu obsahuje tyto důležité adresáře:

- Controllers - Hlavní část webové služby, kde jsou definované http metody pro CRUD operace s daty. Controlery jsou rozděleny do 4 tématických tříd HraController, OtazkaControler, UzivatelController a NastaveniController. ApiControllerHub slouží pro komunikaci v reálném čase pomocí třídy VkvizHub.
- Hubs - Aplikace umožnuje komunikaci v reálném čase pomocí .Net knihovny ASP.net SignalR. Třída VkvizHub obsahuje základní metody pro komunikaci v reálném čase s klientskou částí aplikace.
- Model – Obsahuje základní business logiku. V podsložce Request a Response jsou uloženy třídy pro komunikace s klientskou částí. Třídy jsou pomocí webové služby převáděny do formátu JSON. Dále jsou vytvořeny třídy pro mapovaní Entity na jednoduché C# třídy. Třidy jsou uložené ve složce ViewModels a definice mapovaní je uložena ve složce Mapování.

- Resolver – Obsahuje definici UnityContainer, který slouží pro vytvoření Dependency Injection.

Popsaná struktura projektu je zobrazena na obrázku 22.



Obrázek 22 - Struktura projektu Vkviz_server. Zdroj autor

Dependency Injection je implementováno pomocí Unity container[6]. Díky dependency injection je vyřešeno předávání závislostí tříd knihovny Vkviz_repository. Ve zdrojovém kódu 13 je vytvořen kontejner a registrace jednotlivých typů závislostí. Rozhraní IHraRepository je implementováno pomocí třídy HraRepository. V případě předaní parametru typu IHraRepository, předá kontejner instanci třídy HraRepository.

[6] https://msdn.microsoft.com/en-us/library/ff647202.aspx

```
var container = new UnityContainer();
container.RegisterType<IHraRepository, HraRepository>();
container.RegisterType<IKategorieRepository, KategorieRepository>();
container.RegisterType<IOdpovedHraRepository, OdpovedHraRepository>();
container.RegisterType<IOdpovedRepository, OdpovedRepository>();
container.RegisterType<IOtazkaRepository, OtazkaRepository>();
container.RegisterType<IRoleUzivatelRepository, RoleUzivatelRepository>();
container.RegisterType<IRoleRepository, RoleRepository>();
container.RegisterType<ISeznamHracuRepository, SeznamHracuRepository>();
container.RegisterType<ISeznamOtazekRepository, SeznamOtazekRepository>();
container.RegisterType<IStatistikaRepository, StatistikaRepository>();
container.RegisterType<ITypOtazkyRepository, TypOtazkyRepository>();
container.RegisterType<IUzivatelRepository, UzivatelRepository>();
config.DependencyResolver = new UnityResolver(container);
```

Zdrojový kód 13 - Dependency injection pomocí UnityContainer. Zdroj autor

V knihovně Vkviz_entity jsou vytvořeny entity, pomocí objektově relačního mapovaní databáze. Jelikož vytvořené entity obsahují instance, nebo kolekce instanci na závislé tabulky, je pro jednoduchou reprezentaci dat vytvořeno pomocí knihovny AutoMapper[7] mapovaní na jednoduché c# třídy, tzv. ViewModels třidy. Zdrojový kód 14 vytváří mapování mezi entitou Hra a třídou HraVM. Entita Hra obsahuje kolekci instancí entity Kategorie, které jsou mapovány do kolekce stringu obsahující jednotlivé názvy kategorií, které jsou uloženy ve třídě HraVM. Dále jsou namapovány typy otázek a počet uživatelů připojených k jednotlivému kvízu.

```
CreateMap<Hra, HraVM>()
   .ForMember(vm => vm.Kategorie,
        map => map.MapFrom(s => s.SeznamOtazek.Where(x=> x.IdHra == s.IdHra)
        .Select(x => x.Otazka.Kategorie.Nazev).Distinct()))
    .ForMember(vm => vm.TypOtazky, map =>
        map.MapFrom(s => s.SeznamOtazek.Where(x => x.IdHra == s.IdHra)
        .Select(x => x.Otazka.TypOtazky.Typ).Distinct()))
    .ForMember(vm => vm.PocetPrihlasenych, map =>
        map.MapFrom(s => s.SeznamHracu.Where(x => x.IdHra == s.IdHra)
        .Select(x => x.Uzivatel).Count())).PreserveReferences();
```

Zdrojový kód 14 - Mapováni entity na třídu. Zdroj autor

Další výhodou mezi mapováním entity na ViewModels třídy je možnost ignorování vybraných atributů. Tato vlastnost je dobře využitelná při posílaní informací o uživateli, kde lze pomocí mapovaní ignorovat atribut heslo.

[7] http://automapper.org/

Pro komunikaci v reálném čase je vytvořena třída VkvizHub, která dědí od třídy Hub. Třída Hub je implementována v knihovně ASP .net SignalR[8]. Zdrojový kód 15 zobrazuje třídu VkvizHub, která obsahuje dvě nejdůležitější metody. Metoda Prihlaseni připojí soutěžícího do skupiny klientů, která je definována podle názvu skupiny. Soutěžící je připojen pomocí unikátního identifikátoru, který je určen pomocí připojeného zařízení. Název skupiny je určen pomocí idHry, neboli kvízu, ke kterému se klient připojí. Druhá metoda Odhlaseni slouží k odhlášení připojeného uživatele ze skupiny klientů.

```
[EnableCors(origins: "*", headers: "*", methods: "*")]
[HubName("VkvizHub")]
1 reference
public class VkvizHub : Hub
{

    0 references | 0 exceptions
    public void Prihlaseni(int idHra)
    {
        Groups.Add(Context.ConnectionId, idHra.ToString());
    }


    0 references | 0 exceptions
    public void Odhlaseni(int idHra, int idUzivatel)
    {
        this.Clients.Group(idHra.ToString()).odstranitHrace(idUzivatel);
        Groups.Remove(Context.ConnectionId, idHra.ToString());
    }
```

Zdrojový kód 15 - Hub pro komunikaci v reálném čase. Zdroj autor

Controller ApiCotrollerHub vytváří a vrací instanci VkvizHubu, pomocí kterého komunikuje s klienty v reálném čase. Ve zdrojovém kódu 16 je vytvořen controller HraController, který dědí od controlleru ApiControllerHub.

```
[EnableCors(origins: "*", headers: "*", methods: "*")]
[RoutePrefix("api/Hra")]
1 reference | 0 requests
public class HraController : ApiControllerHub<VkvizHub>
{
    private IHraRepository hraRepository;
    private ISeznamOtazekRepository seznamOtazekRepository;
    private ISeznamHracuRepository seznamHracuRepository;
    private IOdpovedHraRepository odpovedHraRepository;
    private IStatistikaRepository statistikaRepository;
    private IOtazkaRepository otazkaRepository;
```

Zdrojový kód 16 - Třída HraController. Zdroj autor

[8] http://signalr.net/

Zdrojový kód 17 zobrazuje HTTP Post metodu PripojitHru, která připojí soutěžícího do vytvořeného kvízu. Po připojení soutěžícího stáhne údaje o soutěžících již připojených ke kvízu. Poté na řádku č. 123 namapuje entitu na viewModels třídu SeznamHracuVM a informace posílá pomocí řádku č.124 na všechny připojené soutěžící.

```
107   [HttpPost]
108   [Route("PripojitHru")]
      0 references | 0 requests | 0 exceptions
109   public string PripojitHru(PripojitHruRequest request)
110   {
111       var response = new BaseResponse();
112       if (seznamHracuRepository.GetSingle(x => x.IdHra == request.IdHra
113               && x.IdUzivatel == request.IdUzivatel) == null)
114       {
115           SeznamHracu hrac = new SeznamHracu();
116           hrac.IdHra = request.IdHra;
117           hrac.IdUzivatel = request.IdUzivatel;
118           seznamHracuRepository.AddCommit(hrac);
119       }
120       response.Stav = true;
121       var hraci = seznamHracuRepository.AllIncluding(s => s.Hra, s => s.Uzivatel)
122           .Where(x => x.IdHra == request.IdHra).ToList();
123       var hraciVM = Mapper.Map<List<SeznamHracuVM>>(hraci);
124       Hub.Clients.Group(request.IdHra.ToString()).aktualizovatHrace(hraciVM);
125
126       return JsonConvert.SerializeObject(response);
127   }
```

Zdrojový kód 17 - HttpPost metoda pro přijení soutěžící ke kvízu. Zdroj autor

Stručný popis vytvořeného API je v příloze.

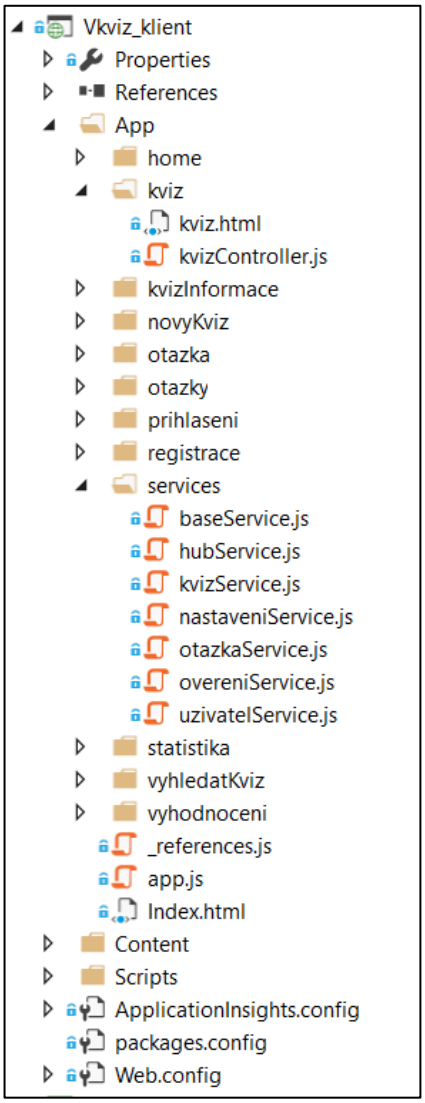
5.5 Vkviz_klient

Projekt Vkviz_klient zobrazuje aplikaci na klientské straně. Je vytvořen pomocí technologie ASP Net Single Page Application, kde spolu s HTML a CSS je použit javascriptový framework AngularJS. Dále je použit pro responzivní design framework Bootstrap, a také knihovna AngularJS.SignalR.Hub, která slouží ke komunikaci v reálném čase se serverem. Ve zdrojovém kódu 18 je zobrazeno připojení k serveru. Metoda spojení bude vybrána automaticky podle typu webového prohlížeče na klientské straně a podle operačního systému na serverové straně.

```
$.connection.hub.start().done(function () {
    console.log("pripojeno");
});
$rootScope.hub.server.prihlaseni($rootScope.hra.IdHra);
```

Zdrojový kód 18 - Připojení k serveru. Zdroj autor

Na obrázku 23 je zobrazena struktura projektu. Projekt má rozdílnou strukturu projektu, než je zvyklé u objektového programovaní. Zde jsou soubory sjednoceny do balíčků podle obrazovky, která je uživateli zobrazena. Obrazovka vždy obsahuje soubor .html a .js. Jediná společná složka je services, která obsahuje služby pro různé obrazovky. Tento styl struktury projektu je obvyklý u projektů využívající framework AngularJS. CSS soubory jsou obsaženy v balíčku Content a javascriptové knihovny v balíčku Scripts.

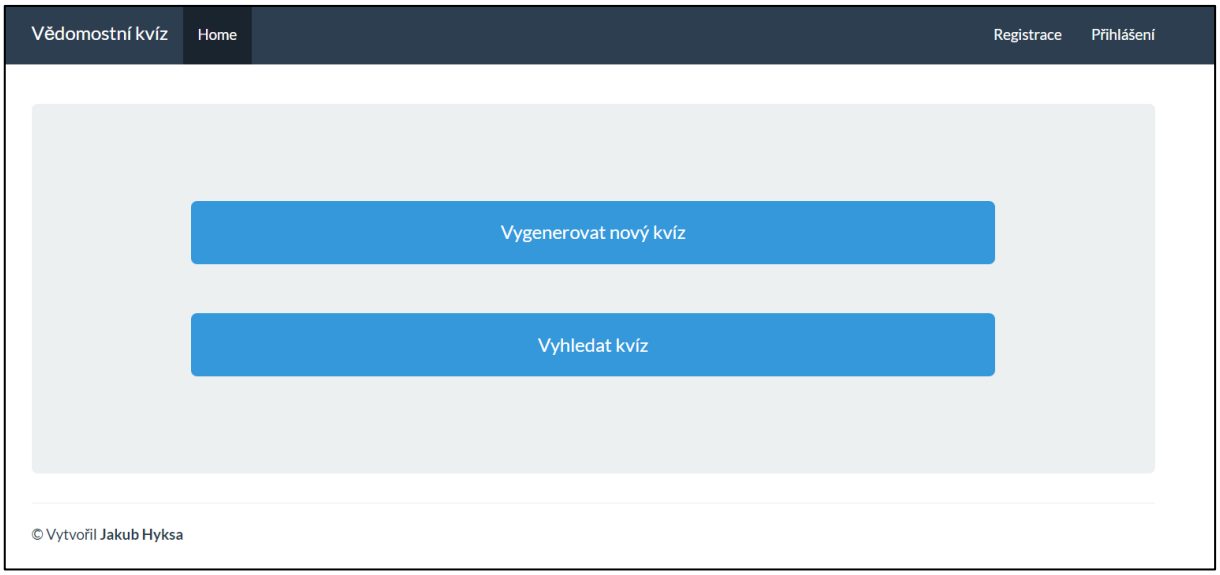


Obrázek 23 - Struktura projektu Vkviz_klient. Zdroj autor

Dále budou představeny nejdůležitější obrazovky Webového vědomostního kvíz.

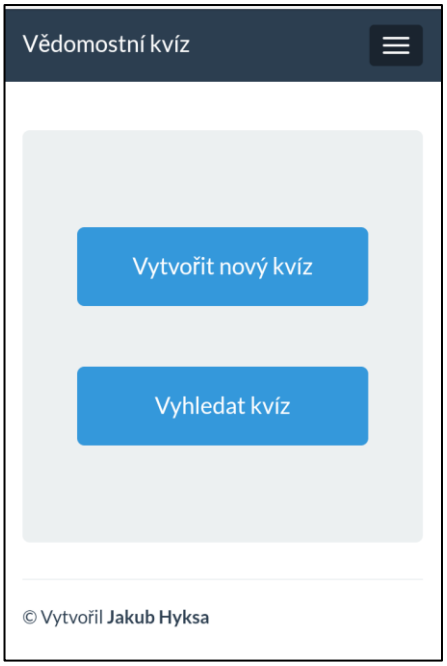
Při spuštění aplikace je zobrazena úvodní obrazovka, která je zobrazena na obrázku 24. Soutěžící má na výběr následující možnosti. Jelikož není přihlášený, tak se může přihlásit, pokud už má vytvořený účet, nebo se registrovat a vytvořit si nový účet. Soutěžící si nemusí nutně pro soutěžení vytvořit nový účet, ale může se přihlásit jako nepřihlášený soutěžící, kde

se mu vygenerují náhodné údaje. Dále si soutěžící může vybrat mezi vygenerováním nového kvízu, nebo vyhledáním kvízu.



Obrázek 24 - Úvodní obrazovka. Zdroj autor

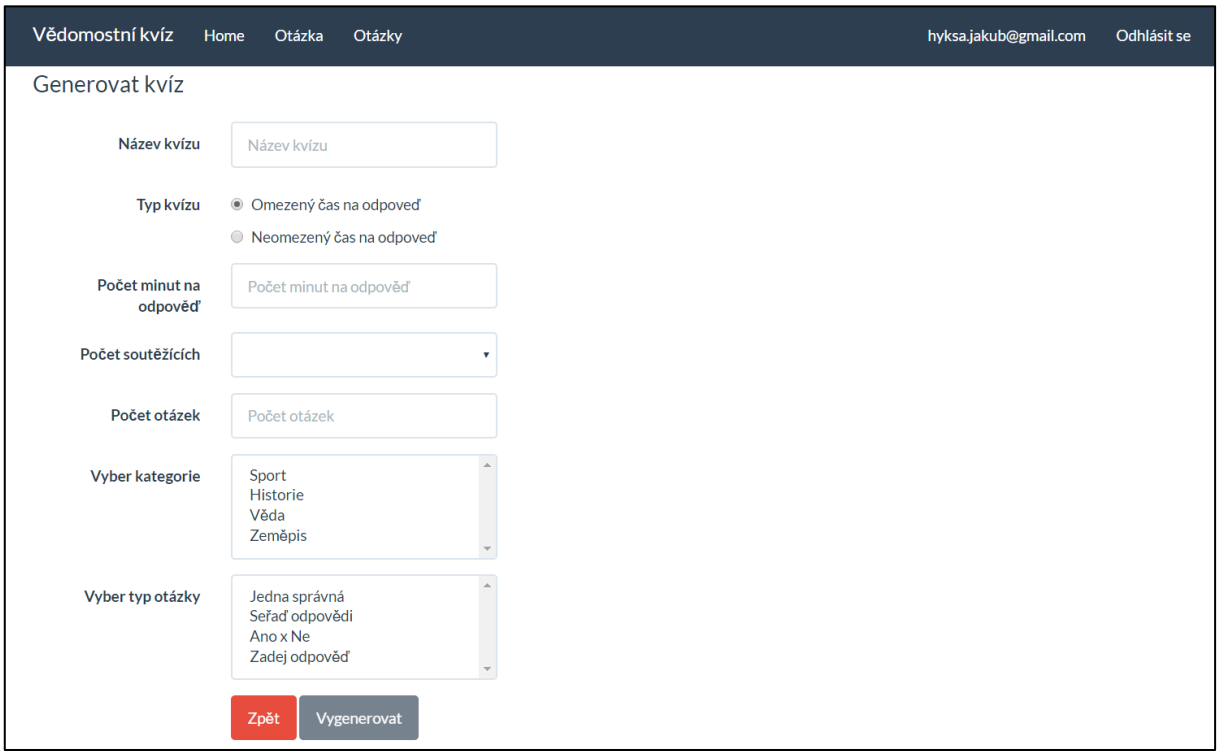
Jelikož aplikace umožnuje responzivní design je možné aplikaci zobrazit i na mobilním zařízení. Na obrázku 25 je zobrazena úvodní obrazovka na mobilním zařízení.



Obrázek 25 - Úvodní obrazovka. Zdroj autor

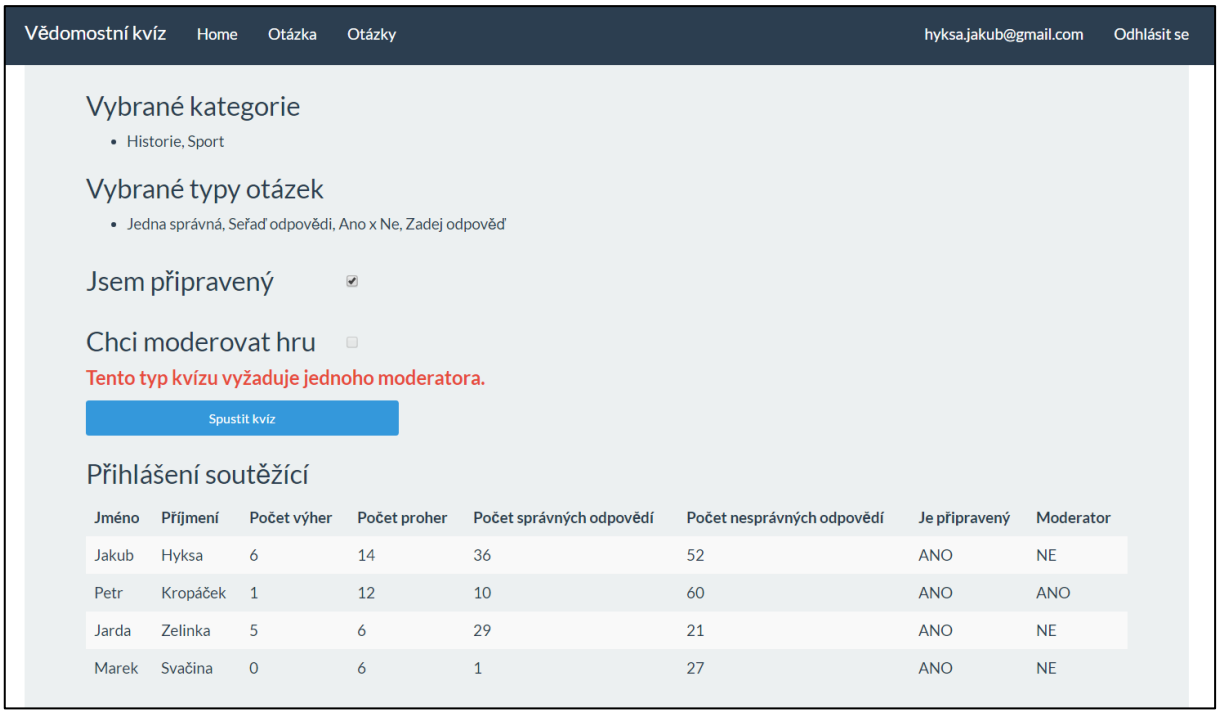
V případě vybrání možnosti „Vygenerovat nový kvíz“ je soutěžícímu zobrazeno nové okno, kde se pomocí formuláře (obrázek 26) vygeneruje kvíz. Soutěžící musí vyplnit všechny položky formuláře. Položka „Počet minut na odpověď“ se vyplňuje podle typu kvízu. V případě vybrání

typu kvízu „Neomezený čas na odpověď“ se tato položka skryje. U položek „Vyber kvíz“ a „Vyber typ otázky“ je možné vybrat jednu a více hodnot. Po vyplnění položek se vygeneruje nový kvíz. Novému kvízu jsou náhodně přiřazeny otázky podle vybraných kategorií a typu otázky. V každém kvízu je náhodně vybraná otázka pouze jednou.



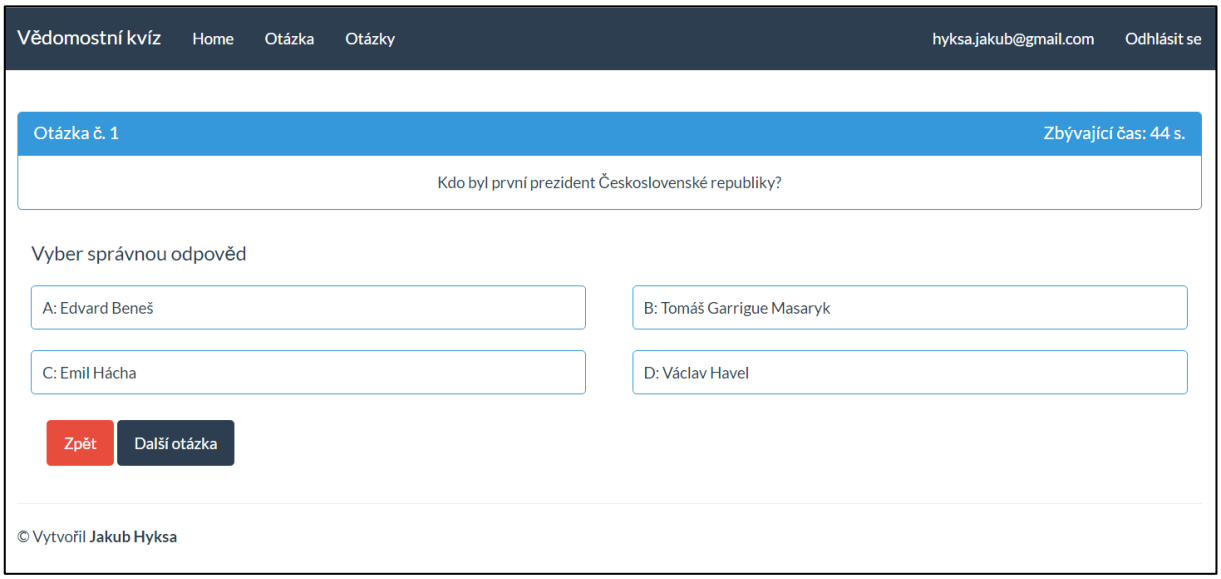
Obrázek 26 - Generování kvízu. Zdroj autor

V případě vybrání možnosti „Vyhledat kvíz“ je soutěžícímu zobrazeno nové okno, kde je možné si vybrat z vygenerovaných kvízů. Soutěžící si vybere na základě parametrů kvízu, které jsou vyplněny při jeho vygenerovaní. Mezi klíčové parametry pro výběr patří typ kategorie a typ otázky. Po vybraní kvízu je soutěžícímu zobrazeno okno s informacemi o kvizu (obrázek 27). Dále jsou zobrazeny přihlášení soutěžící a jejich statistiky. Po spuštění kvízu je nutné, aby všechny soutěžící vybrali možnost „Jsem připravený“. V případě typu kvízu „Neomezený čas na odpověď“ je zobrazena možnost „Chci moderovat“, kde musí jeden ze soutěžících tuto možnost vybrat. Jakmile poslední soutěžící vybere možnost „Jsem připravený“ , je možné kvíz spustit.



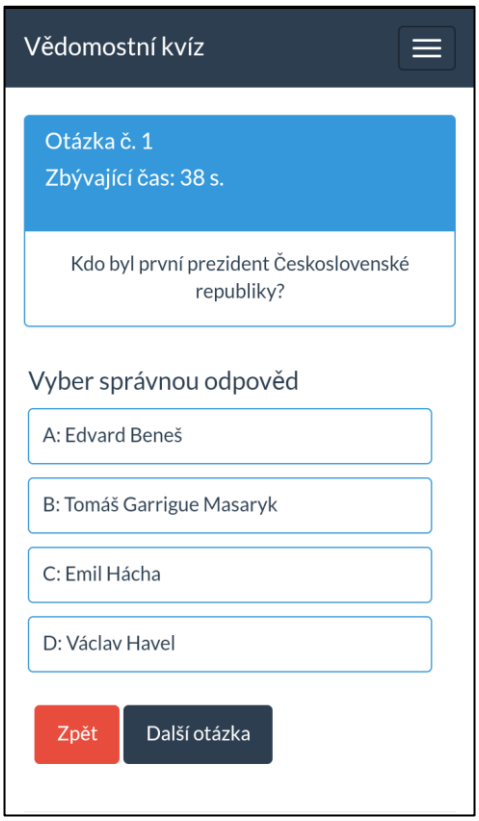
Obrázek 27 - Informace o kvízu. Zdroj autor

Po spuštění kvízu se soutěžícím zobrazí otázka. Pokud je typ kvízu „Omezený čas na odpověď“, musí soutěžící odpovědět v časovém intervalu, který je zobrazený v pravém horním rohu. Pokud časový interval vyprší, tak se otázka automaticky změní a soutěžící automaticky odpověděl špatně. Poté co soutěžící odpoví na otázku, je ostatním soutěžícím zobrazena notifikační zpráva. Zpráva zobrazuje jméno soutěžícího a číslo odpověděné otázky. Po odpovědi na otázku je hned zobrazena další otázka, takže v libovolném čase od spuštění kvízu muže mít každý soutěžící zobrazenou jinou otázku.



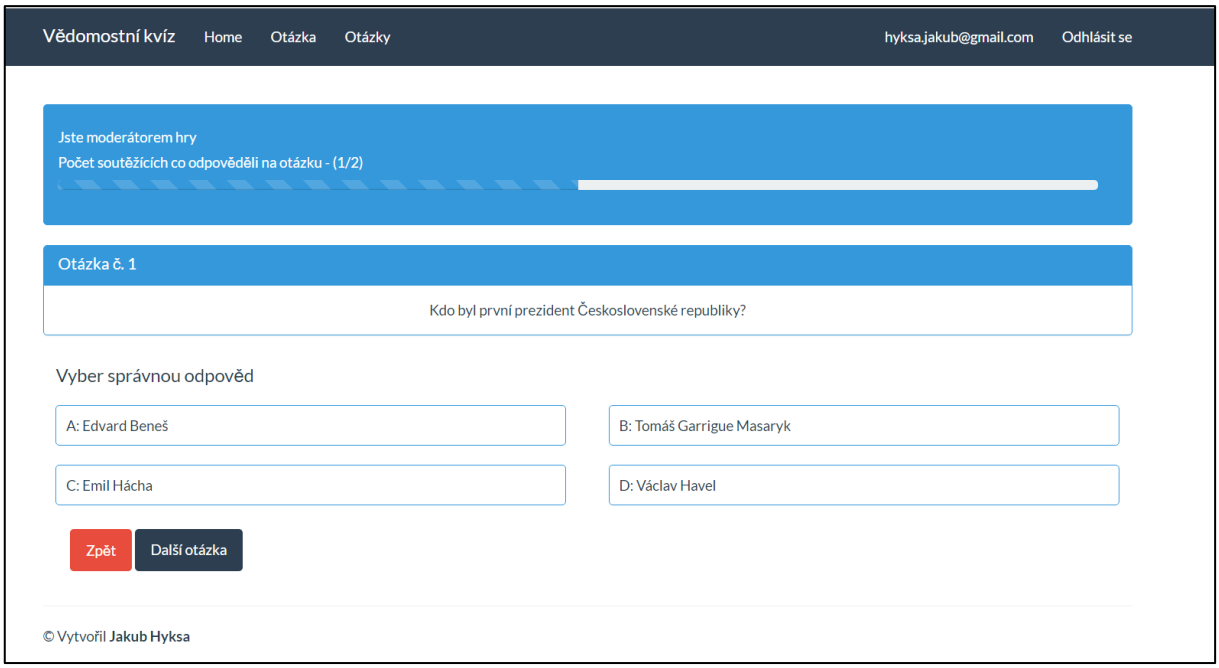
Obrázek 28 - Zobrazení otázky. Zdroj autor

Na mobilních zařízení (obrázek 29) je časový interval zobrazen pod číslem otázky.



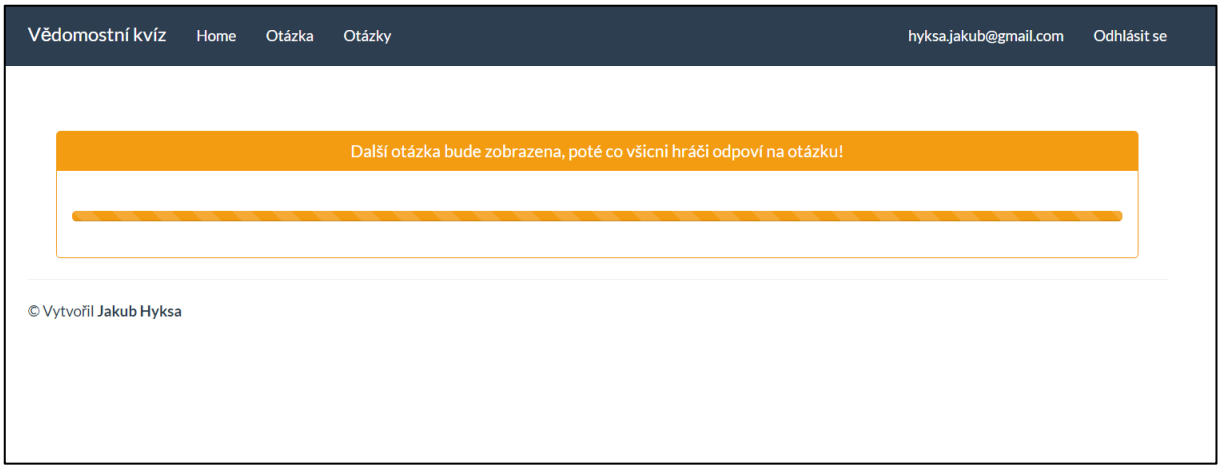
Obrázek 29 - Zobrazení otázky. Zdroj autor

Pokud je typ kvízu „Neomezený čas na odpověď', tak soutěžícímu neběží žádný časový interval, během kterého by musel odpovědět. Časový interval na otázku určuje pouze moderátor hry. Moderátor nemůže na otázku odpovídat, ta mu je pouze zobrazena. Moderátorovi je zobrazena informace o tom, že je moderátor a také počet soutěžících, kteří už na danou otázku odpověděli. Moderátor může v libovolný čas přepnout otázku. Je pouze na jeho rozhodnutí, zda počká na to, než všichni soutěžící odpoví. V případě, že některý soutěžící nestihne odpovědět, je mu otázka započítána, jako špatná. Na obrázku 30 je zobrazena moderátorova obrazovka.



Obrázek 30 - Zobrazeni otázky - moderátor. Zdroj autor

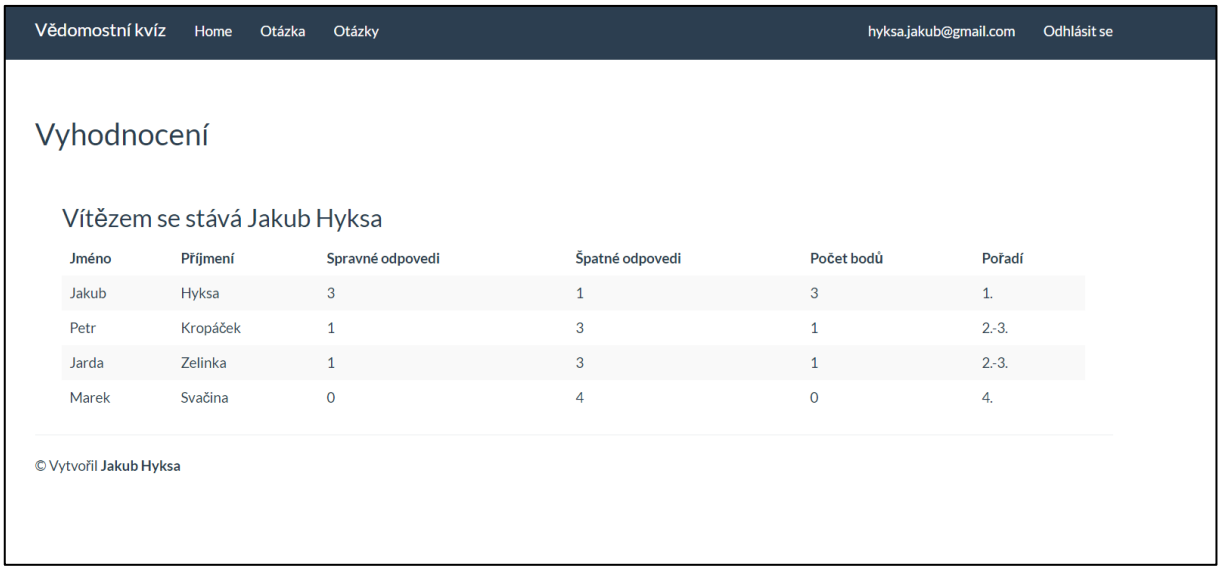
Obrázek 31 ukazuje stav po zodpovězení na otázku, kde soutěžící čeká na to, až moderátor přepne otázku a všem soutěžícím se zobrazí nová otázka. Tímto stavem se zajišťuje, že všichni soutěžící budou mít zobrazenou stejnou otázku. Tento stav je zobrazen pouze v případě typu kvízu „Neomezený čas na odpověď“.



Obrázek 31 - Čekání na další otázku. Zdroj autor

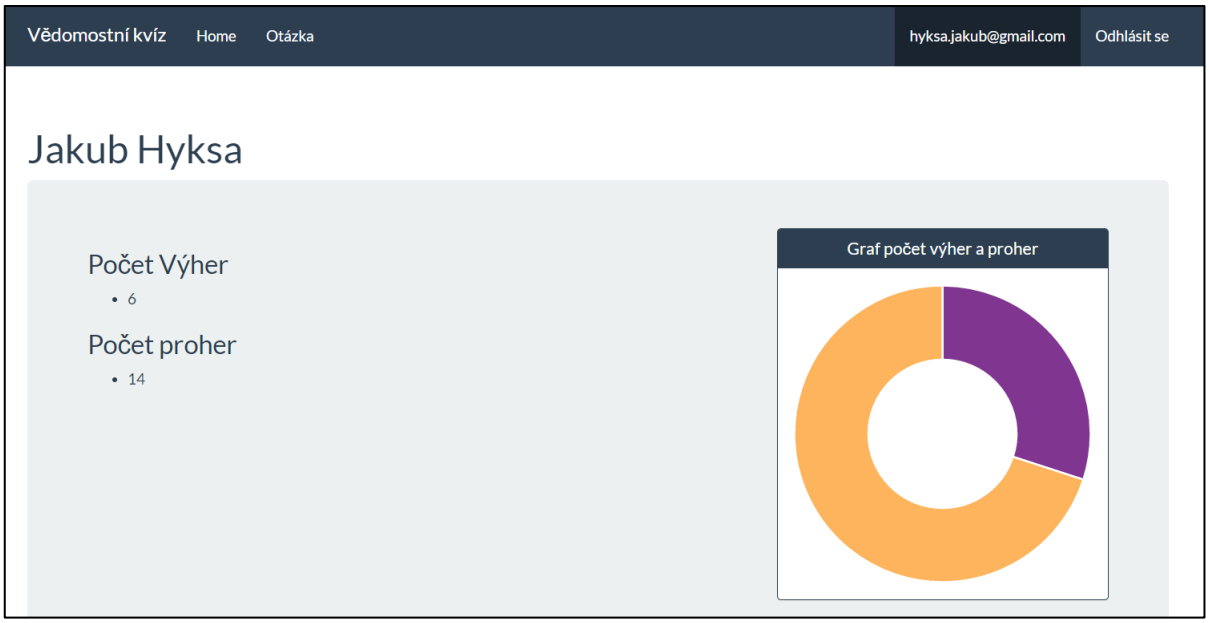
Poté co všichni soutěžící odpoví na všechny otázky, je všem zobrazeno vyhodnocení kvízu (obrázek 32), kde je vypsán vítěz kvízu. Pořadí v kvízu je určeno podle počtu bodů. Za každou správnou odpověď je soutěžící připočítán jeden bod. V případě typu otázky „Zadej hodnotu“, získává bod soutěžící, který zadá nejpřesnější hodnotu. V případě shody nejpřesnější hodnoty je připočítán bod všem soutěžícím, kteří tuto hodnotu zadali. Každý přihlášený soutěžící má

svoje statistiky, kde je zobrazeno počet vítězství a proher v kvízu. Dále jsou započítávány správné a špatné odpovědi.



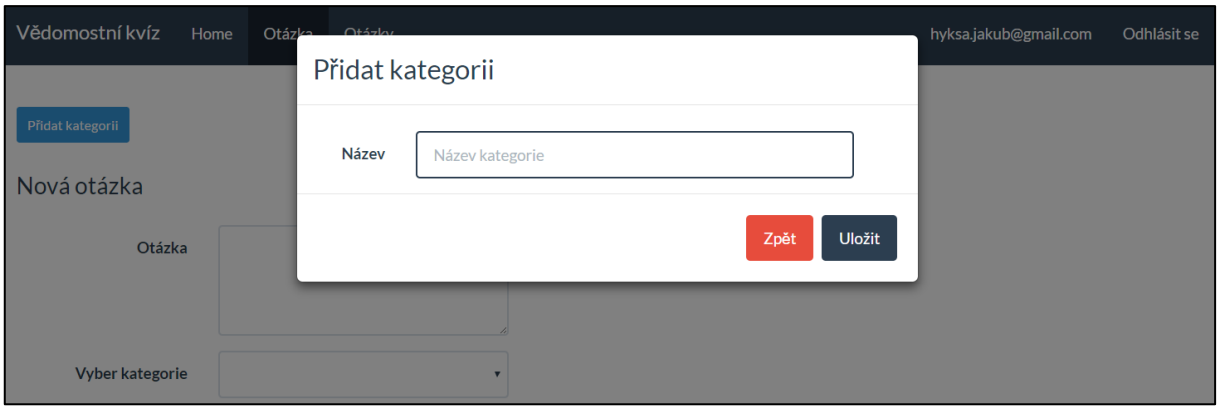
Obrázek 32 - Vyhodnocení hry. Zdroj autor

Přihlášený soutěžící si také může zobrazit stručný přehled svých statistik. Přehled statistik je zobrazen na obrázku 33.



Obrázek 33 - Statistiky soutěžícího. Zdroj autor

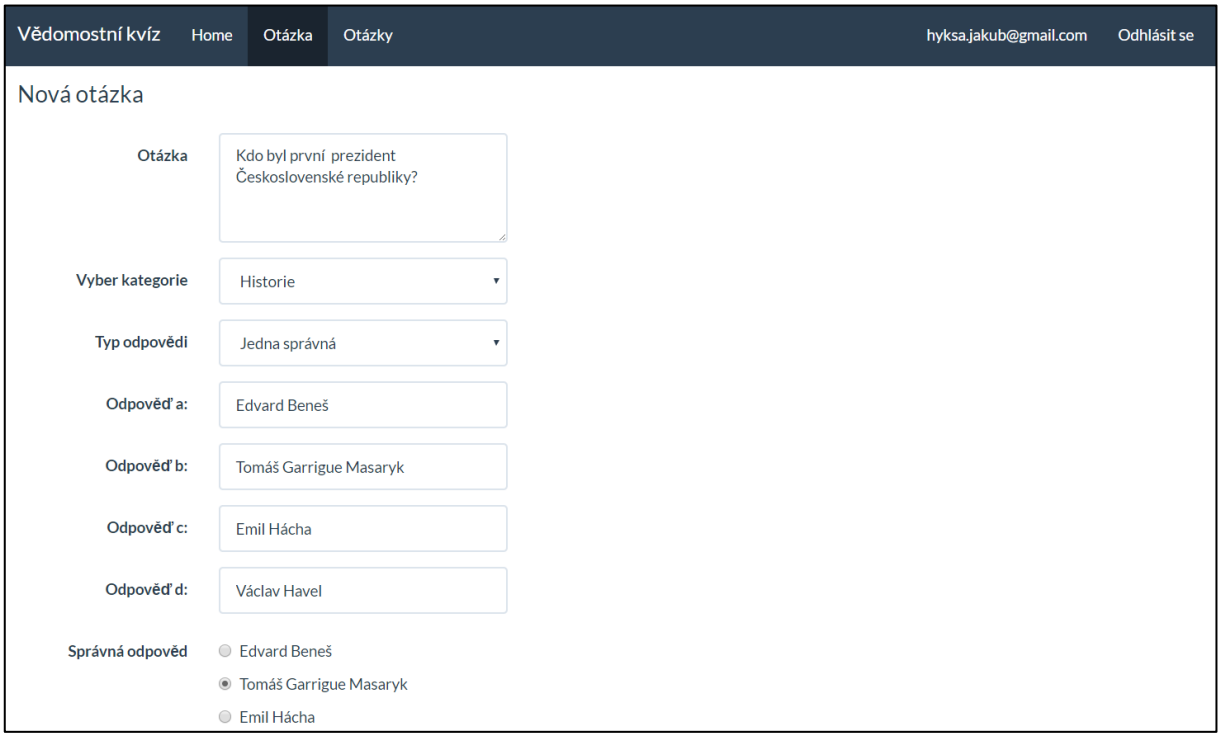
Na obrázku 34 je znázorněno vytváření kategorií. Kategorie může vytvářet pouze administrátor, a to z důvodu zamezení vytváření nevhodného obsahu.



Obrázek 34 - Vytvoření kategorie. Zdroj autor

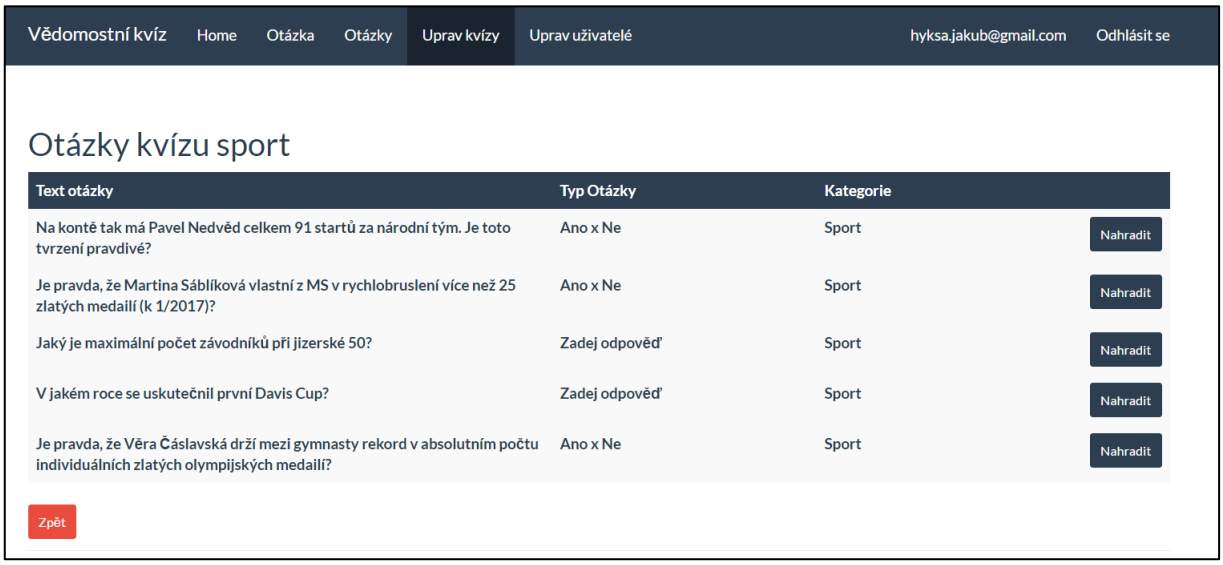
Další možností aplikace je vytváření otázek. Otázky může vytvářet přihlášený soutěžící a administrátor. V případě vytvoření otázky přihlášeným uživatelem je otázka uložena jako neschválená z důvodu zabránění vytvoření nevhodného obsahu. Neschválenou otázku musí schválit odpovědná osoba, a to je administrátor.

Nová otázka se vytvoří pomocí formuláře zobrazeného na obrázku 35. Důležitou položkou formuláře je položka typ kategorie, která určuje kategorii otázky, podle které se potom vybírají otázky pro generovaný kvíz. Další důležitou položkou formuláře je položka typ otázky, pomocí které se dynamicky vygeneruje obsah formuláře. Dynamicky se generuje počet odpovědí podle vybraného typu otázky. U vytvořené otázky lze zadat zdroj inspirace při vytváření otázky.



Obrázek 35 - Vytvoření otázky. Zdroj autor

Záleží pouze na administrátorovi, zda schválí otázku, popřípadě může ještě znění otázky nebo odpovědí změnit. Dále také může měnit vygenerované otázky u daného kvízu. Vybranou otázku může nahradit za jinou, která odpovídá parametrům kvízu typ kategorii a typ otázek. Na obrázku 36 je zobrazena obrazovka s možností nahrazení otázky u vybraného kvízu.



Obrázek 36 - Nahrazení otázek. Zdroj autor

ZÁVĚR

V praktické části byly představeny vybrané existující aplikace typu vědomostní kvíz nebo vědomostní hra. Byly vybrány aplikace Dobyvatel, IQ test, Chcete být milionářem a Mozkovna. Tyto aplikace byly mezi sebou porovnány a také byly porovnány s Webovým vědomostním kvízem. Mezi porovnávanými vlastnostmi byl vybrány počet a typ otázek. Dále byla vybrána možnost výběru kategorii a možnost výběru více kategorií u jednoho spuštěného kvízu. Také byl sledován maximální počet soutěžících a jejich nutnost přihlášení. Za důležité vlastnosti byly také považovány časový limit na odpověď a možnost nápovědy. Ze sledovaných vlastností byly vybrány nejlepší hodnoty vlastností a ty byly implementovány ve Webovém vědomostním kvízu. V této části byly také představeny moderní .NET technologie potřebné pro vývoj aplikací.

V praktické části byly splněny všechny požadované úkoly. Aplikace byla rozdělena do dvou částí, a to serverové a klientské. V serverové části byla implementována webová služba typu REST, která je propojena s databázovou vrstvou. V klientské části byly zobrazeny všechny implementované funkčnosti aplikace. Byla implementována možnost vytvoření dvou typů kvízu. První typ kvízu má omezený časový limit na otázku, po který musí soutěžící odpovědět na zobrazenou otázku. Pokud soutěžící neodpoví je mu automaticky zobrazena jiná otázka. Druhý typ kvízu nemá omezený časový limit na otázku. U tohoto typu kvízu řídí zobrazení otázek třetí osoba, takzvaný moderátor. Moderátor neodpovídá na otázku, pouze řídí jejich zobrazení. Veškerá komunikace mezi soutěžícími probíhá v reálném čase. V aplikaci bylo umožněno vytvářet nové otázky, které musí být schváleny zodpovědnou osobou. Zodpovědná osoba muže také u vygenerovaného kvízu nahradit náhodně přiřazené otázky. Díky responzivnímu designu byla implementována možnost zobrazení kvízu na mobilním zařízení.

Největším problémem diplomové práce byla implementace komunikace v reálném čase, protože protokol WebSocket podporují pouze prohlížeče s podporou HTML 5. Tento problém byl vyřešen použitím knihovny ASP .NET SignalR, která podporuje všechny protokoly potřebné ke komunikaci v reálném čase.

Aplikace byla implementována pomocí moderních technologií. Na serverové části byla vytvořena REST služba pomocí ASP. NET Web API a klientská část byla implementována pomocí javascriptového frameworku AngularJS. Komunikace v reálném čase byla implementovaná pomocí knihovny ASP .NET SignalR, která využívá protokol WebSocket.

Díky rozdělení aplikace do dvou částí je možné v budoucnosti využít serverovou část a aplikaci rozšířit o mobilní aplikaci, která bude komunikovat s webovou službou implementovanou v serverové části. Dále by bylo možné aplikaci rozšířit o podrobnější statistiky.

6 POUŽITÁ LITERATURA

[1] ABC Mill. Dobyvatel nápověda: Úvod [online]. 2017 [cit. 2017-05-10]. Dostupné z: http://dobyvatel.nova.cz/help.php?tl= 102

[2] ABC Mill. Dobyvatel nápověda: Pravidla hry [online]. 2017 [cit. 2017-05-10]. Dostupné z: http://dobyvatel.nova.cz/help.php?tl= 120

[3] ABC Mill. Dobyvatel nápověda: Mini turnaje [online]. 2017 [cit. 2017-05-10]. Dostupné z: http://dobyvatel.nova.cz/help.php?tl= 124

[4] IQ TEST: Články a zajímavosti o IQ [online]. 2017 [cit. 2017-05-10]. Dostupné z: https://www.iqtest-certification.com/cs-cz/o-iq

[5] GRUDL, David. Nette Framework: MVC & MVP. In: *Zdroják.cz* [online]. 2009 [cit. 2017-05-10]. Dostupné z: http://www.zdrojak.cz/clanky/nette-framework-mvc-mvp/

[6] Entity Framework Tutorial: What is Entity Framework? [online]. 2016 [cit. 2017-05-10]. Dostupné z: http://www.entityframeworktutorial.net/what-is-entityframework.aspx

[7] LERMAN, Julia. Programming Entity framework. 2nd ed. Sebastopol: O´Reilly, 2010. ISBN 978-0-596-80726-9.

[8] Entity Framework Tutorial: Execute Native SQL Query [online]. 2016 [cit. 2017-05-10]. Dostupné z: http://www.entityframeworktutorial.net/EntityFramework4.3/raw-sql-query-in-entity-framework.aspx

[9] Entity Framework Tutorial: Entity Relationships: [online]. 2016 [cit. 2017-05-10]. Dostupné z: http://www.entityframeworktutorial.net/entity-relationships.aspx

[10] Developer network: Common Language Runtime (CLR) [online]. [cit. 2017-05-10]. Dostupné z: https://msdn.microsoft.com/en-us/library/8bs2ecf4

[11] Microsoft: Introduction to the C# Language and the .NET Framework [online]. [cit. 2017-05-10]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/articles/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework

[12] TutorialsPoint: ASP.NET MVC - Overview [online]. [cit. 2017-05-10]. Dostupné z: https://www.tutorialspoint.com/asp.net_mvc/asp.net_mvc_overview.htm

[13] TutorialsPoint: ASP.NET MVC - Life Cycle [online]. [cit. 2017-05-10]. Dostupné z: https://www.tutorialspoint.com/asp.net_mvc/asp.net_mvc_life_cycle.htm

[14] GALLOWAY, Jon., Brad WILLSON, K. Scott ALLEN a David MATSON. Professional asp.net mvc 5. Indianapolis: Wrox, 2014. ISBN 1118794753.

[15] TutorialsPoint: ASP.NET MVC - Web API [online]. [cit. 2017-05-10]. Dostupné z: https://www.tutorialspoint.com/asp.net_mvc/asp.net_mvc_web_api.htm

[16] ČÁPKA, David. ASP.NET MVC - Single Page Application: 1. díl - Úvod do Single Page Application v ASP.NET [online]. [cit. 2017-05-10]. Dostupné z: https://www.itnetwork.cz/csharp/asp-net/single-page-application/tutorial-uvod-do-asp-net-single-page-application

[17] GREEN, Brad a Shyam SESHADRI. AngularJS. Sebastopol: O'Reilly Media, 2013. ISBN 978-1-4493-4485-6.

[18] MROZEK, Jakub. Zdrojak.cz: Začínáme s AngularJS [online]. 2012 [cit. 2017-05-10]. Dostupné z: https://www.zdrojak.cz/clanky/zaciname-s-angularjs/

[19] AngularJS: Creating Custom Directives [online]. [cit. 2017-05-10]. Dostupné z: https://docs.angularjs.org/guide/directive

[20] FLETCHER, Patrick. Microsoft: Introduction to SignalR [online]. 2014 [cit. 2017-05-10]. Dostupné z: https://docs.microsoft.com/en-us/aspnet/signalr/overview/getting-started/introduction-to-signalr

[21] W3school.com: JSON vs XML [online]. [cit. 2017-05-10]. Dostupné z: https://www.w3schools.com/js/js_json_xml.asp

[22] Interval.cz: Jak fungují webové služby [online]. 2002 [cit. 2017-05-10]. Dostupné z: https://www.interval.cz/clanky/jak-funguji-webove-sluzby/

[23] TutorialsPoint: What is SOAP? [online]. [cit. 2017-05-10]. Dostupné z: https://www.tutorialspoint.com/soap/soap_quick_guide.htm

[24] Part II. Architectures for the Web: Chapter 6. Web Services [online]. [cit. 2017-05-10]. Dostupné z: https://gyires.inf.unideb.hu/GyBITT/08/ch06.html

[25] RODRIGUEZ, Alex. IBM: RESTful Web services: The basics [online]. 2008 [cit. 2017-05-10]. Dostupné z: https://www.ibm.com/developerworks/webservices/library/ws-restful/

7 PŘÍLOHY

Příloha A - *Dokumentace WEB API*

Hra

API	Description
GET api/Hra/GetHry	Získání všech kvízů
GET api/Hra/GetHru?id={id}	Ziskaní informací o jednom kvízu
GET api/Hra/GetOtazky?idHra={idHra}	Získání všech otázek, konkrétního kvízu
GET api/Hra/GetOtazkyKUprave?idHra={idHra}	Získání otázek kvízu a dostupných otázek
POST api/Hra/VymenOtazky	Nahrazení otázek u vybraného kvízu
POST api/Hra/PripojitHru	Připojení uživatele ke kvízu
POST api/Hra/NovaHra	Vygenerování nového kvízu
POST api/Hra/OdpojitHrace	Odpojení soutěžící z daného kvízu
PUT api/Hra/NastavKonec?idHra={idHra}	Nastavení příznaky konce kvízu pro daného soutěžicího
PUT api/Hra/NastavKonceModerator?idHra={idHra}	nNastaveni konce kvízu pro moderátora
POST api/Hra/UlozOdpoved	Uložení odpovědi
DELETE api/Hra/{id}	Smazaní vybraného kvízu

Nastaveni

API	Description
GET api/Nastaveni/GetNastaveni	Získání nastavení hry - Kategorie a typ otázky

Uzivatele

API	Description
POST api/Uzivatele/Prihlaseni	Přihlášení soutěžícího
GET api/Uzivatele/NahodnyUzivatel	Vygenerování nepřihlášeného soutěžicího
GET api/Uzivatele	Ziskání uživatelů typu přihlášený soutěžicí
PUT api/Uzivatele/{id}	Nataveni admin role vybranému soutěžicímu
POST api/Uzivatele	Vytvoření uživatele

Otazka

API	Description
GET api/Otazka/GetOtazky	Získání všech otázek
GET api/Otazka/GetOtazkyKeSchvaleni	Získení neschválených otázek
POST api/Otazka/ulozJednaSpravna	Uložení otázky typu jedna správná
POST api/Otazka/UlozSeradOdpovedi	Uložení otázky typu seřaď odpovedi
POST api/Otazka/UlozAnoNe	Uložení otázky typu Ano x Ne
POST api/Otazka/UlozZadejOdpoved	Uložení otázky typu zadej odpoved
POST api/Otazka/UpravOtazku	Upravení otázky
POST api/Otazka/VytvorKategorii	Vytvoření kategorie

Příloha B – *Verze použitých knihoven*

- Framework .NET 4.6
- Entity Framework 6.1.3
- AutoMapper 5.2.0.
- ASP .NET SignalR 2.2.1
- Unity 4.0.1
- Newtonsoft.Json 6.0.4
- json-serialize 1.1.2
- AngularJS 1.5.9
- AngularJS.UI.UI-Router 0.31
- AngularJS.SignalR.Hub 1.6.2
- AngularJS.UI.Bootstrap 2.3.0
- AngularJS.Animate 1.5.9
- AngularJS.Core 1.5.9
- Angular-ui.notification 0.3.6
- Angular-chart 1.1.1
- JQuery 3.1.1
- Boostrap 3.0.0