Masarykova univerzita
Fakulta informatiky



Aplikácia na prezentovanie multimediálneho obsahu na TV

Diplomová práca

Bc. Marián Varaga

Brno, jar 2017

Namiesto tejto stránky vložte kópiu oficiálneho podpísaného zadania práce a prehlásenie autora školského diela.

Prehlásenie

Prehlasujem, že táto diplomová práca je mojím pôvodným autorským dielom, ktoré som vypracoval samostatne. Všetky zdroje, pramene a literatúru, ktoré som pri vypracovaní používal alebo z nich čerpal, v práci riadne citujem s uvedením úplného odkazu na príslušný zdroj.

Bc. Marián Varaga

Vedúci práce: RNDr. Barbora Kozlíková, Ph.D

Poďakovanie

Rád by som poďakoval RNDr. Barbore Kozlíkovej, Ph.D za vedenie mojej práce a cenné rady a taktiež by som chcel poďakovať konzultantovi Ing. Jánovi Vernerovi za trpezlivosť a ústretový prístup pri konzultáciách.

Zhrnutie

Cieľom diplomovej práce je návrh a vytvorenie aplikácie na zdieľanie multimediálneho obsahu a jeho prezentáciu na televíznej obrazovke. Pre vytvorenie práce bola potrebná analýza požiadaviek a návrh architektúry. Požadované bolo použitie technológie Microsoft .NET z dôvodu možného neskoršieho rozšírenia aplikácie vo firme Siemens Corporate Technology Brno. Práca je zameraná na naprogramovanie jednoducho použiteľnej aplikácie, ktorá umožnuje zdieľanie a prezentovanie multimediálneho obsahu na televízii. V práci opisujem použité technológie, knižnice a výsledky implementácie.

Kľúčové slová

webové služby, zdieľanie obsahu, REST, ASP.NET, MVC, .NET, WPF, MVVM, televízna obrazovka

Obsah

1 **Úvod** 1
2 **Analýza existujúcich aplikácií a možných riešení** 3
2.1 *Sociálne siete* 3
2.2 *Programy na prezentáciu obsahu* 3
2.3 *Systémy digital signage* 4
2.4 *Vlastná aplikácia na Smart TV* 5
2.5 *Zhrnutie* 5
3 **Analýza požiadaviek** 6
3.1 *Užívateľské požiadavky na systém* 6
3.2 *Prípady použitia aplikácií* 7
3.2.1 Registrácia a prihlásenie 10
3.2.2 Vytváranie a úprava príspevku 11
3.2.3 Zobrazovanie príspevkov 11
3.2.4 Ovládanie TV 12
3.3 *Nefunkčné požiadavky* 14
3.4 *Prehľad technológií* 15
4 **Aplikácia zobrazujúca obsah na TV** 19
4.1 *Prenos dát* 19
4.2 *Architektúra aplikácie zobrazujúcej obsah na TV* 22
4.3 *Ukladanie dát* 23
4.4 *Windows Presentation Foundation* 24
4.5 *Vzor Model-View-ViewModel* 27
4.6 *AutoMapper* 28
4.7 *Ďiaľkové ovládanie* 28
4.8 *Užívateľské rozhranie aplikácie na zobrazovanie obsahu na televízii* 30
5 **Webová aplikácia** 31
5.1 *Architektúra aplikácie* 31
5.2 *Ukladanie obsahu* 32
5.3 *Knižnica Hangfire* 32
5.4 *IoC – Simple Injector* 32
5.5 *Užívateľské rozhranie webovej aplikácie* 34
6 **Vyhodnotenie a nasadenie** 37
7 **Záver** 38
Literatúra 39

1 Úvod

V dnešnom svete sú technológie neodmysliteľnou súčasťou nášho každodenného života. Technológie, ktoré aktívne používame, sa neustále menia, zlepšujú a ovplyvňujú naše životy. Dnešní ľudia sa čoraz viac zaoberajú sociálnymi sieťami, najnovšími aplikáciami či mobilnými zariadeniami. Žijeme v dobe, keď sú informácie veľmi cenné a ľahko dostupné. Ľudia majú radi nové technológie a radi zdieľajú zaujímavé informácie medzi sebou. V pracovnom prostredí chcú informovať svojich kolegov o nových aktualitách týkajúcich sa práce, pracovných udalostiach, ale aj o novinkách zo súkromného života.

V práci trávime väčšinu nášho času, sme súčasťou sociálnej skupiny, v ktorej komunikujeme rôznymi spôsobmi. Najčastejším spôsobom je verbálna komunikácia, ktorá je vhodnejšia na použitie v menšej skupine ľudí, napríklad v rámci tímu. Naopak emailová komunikácia umožnuje zasiahnuť veľké množstvo ľudí. Nevýhodou emailovej komunikácie môže byť, že nie vždy dokážeme určiť ľudí, pre ktorých bude zasielaná informácia zaujímavá. Medzi ďalšie spôsoby komunikácie, ktoré sa využívajú v pracovnom prostredí, patria okamžité správy, telefonická komunikácia, sociálne siete a mnohé ďalšie.

V prípade, že chceme zasiahnuť širšie publikum, väčšina spôsobov komunikácie spôsobí informovanie aj ľudí, pre ktorých nemusí byť daná informácia relevantná a ani zaujímavá. Prílišné informovanie nerelevantnými informáciami môže byť rušivé a takisto može spôsobovať zahltenie informáciami, čím sa môže sťať, že unikne dôležitá informácia.

Vo firemnom prostredí sa často nachádzajú obrazovky či televízie, ktoré sú väčšinu času nevyužité, a preto sa ponúka možnosť využiť ich práve na zdieľanie obsahu medzi zamestnancami. Tieto zariadenia sa zväčša nachádzajú v priestoroch, ktoré neslúžia priamo na prácu, ale skôr na relax a doplnenie energie. Týmto spôsobom sa umožní zdieľanie informácií naprieč celou firmou v rámci jednej geolokácie a zároveň sa obmedzí rušivý vplyv zdieľania informácií počas práce.

Využívanie televíznej obrazovky za účelom zdieľania obsahu popisujem v mojej diplomovej práci. Témou práce je preskúmať možnosti zdieľania multimediálneho obsahu vo firemnom prostredí s prezentáciou na televíznej obrazovke.

Práca je vypracovaná v spolupráci s firmou Siemens Corporate Technology Brno a jej cieľom je prezentácia multimediálneho obsahu na televízii, ktorá sa nachádza v zdieľanom priestore. Do projektu zdieľaného obsahu boli zapojení zamestnanci pobočky, ktorí mi umožnili prieskum, akou formou a aký obsah by mali záujem zdieľať.

Výsledkom práce sú dve aplikácie – webová aplikácia, ktorá umožňuje zadávanie a spracovanie multimediálneho obsahu a aplikácia zobrazujúca zadaný obsah na televíznej obrazovke. Prostredníctvom webovej aplikácie je zadávaný obsah, ktorý sa prezentuje na televíznej obrazovke. Prezentovanie obsahu na televíznej obrazovke má na starosti klientská aplikácia, ktorá preberá obsah z webovej aplikácie. Obe aplikácie sú neoddeliteľnou súčasťou jedného celku.

Diplomová práca má nasledujúcu štruktúru. V prvej časti práce opisujem existujúce riešenia s ich výhodami a nevýhodami. V druhej kapitole práce vysvetlujem požiadavky na aplikácie, ktoré vyplynuli z dotazníka vyplneného zamestnancami brnenskej pobočky firmy Siemens. Ďalej sú v tejto kapitole popísané užívateľské scenáre. V nasledujúcich dvoch kapitolách práce popisujem použité technológie. Nasleduje popis návrhu a implementácie dvoch aplikácií, ktoré boli vytvorené ako súčasť tejto práce. V posledných kapitolách vyhodnocujem implementované aplikácie. Súčasťou zhodnotenia je aj spätná väzba od zadávateľa.

2 Analýza existujúcich aplikácií a možných riešení

Cieľom tejto časti je spraviť prehľad existujúcich riešení, ktoré môžu byť použité na zdieľanie obsahu na firemnú televíznu obrazovku. Základnými požiadavkami je, aby mohli užívatelia zdieľat obsah, ktorý bude zobrazovaný na televíznej obrazovke. Obsah by mal byť zobrazovaný v užívateľsky prívetivej forme a jednoducho ovládateľný.

2.1 Sociálne siete

Ako prvé riešenie sa ponúkajú sociálne sieťe, ktoré sú bežne rozšírené. Či už ide o Facebook, Twitter alebo iné, ich použitie je veľmi podobné. Spĺňajú jednoduché zadávanie obsahu, ktoré je užívateľsky príjemné a otestované na miliónoch užívatelov. Napríklad sociálna stránka Facebook má dobrú podporu multimediálneho obsahu, textových príspevkov, ako aj linkov. Problémom týchto riešení je, že zdieľaný obsah by mohol byť prístupný aj mimo organizáciu, čo nie je vždy želané. Obsah prezentovaný na televíznej obrazovke môže byť zobrazovaný priamo v prehliadači, ale tento prístup očakáva bežný spôsob ovládania pomocou myši a klávesnice. Možnosťou je zobrazovať obsah vo vlastnej aplikácii, ktorá by bola vytvorená podľa požiadavkov. Problémom zostáva závislosť na API externej služby, obsah uložený mimo firemný intranet. Neoddeliteľným problémom tohto riešenia zostáva použitie sociálnych sietí, ktoré nemusia byť povolené alebo užívateľa rozptyľujú od práce.

2.2 Programy na prezentáciu obsahu

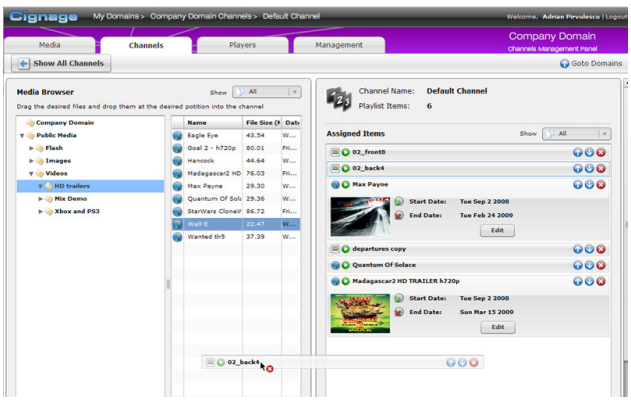
Využitie programov na prezentácie, ako sú PowerPoint z balíka Microsoft Office, Keynote z prostredia operačných systémov spoločnosti Apple, alebo služieb ako Prezi, čo je online nástroj na vytváranie prezentácií, umožňuje vytváranie komplexných prezentácií s rôznymi prvkami. Umožňujú veľkú flexibilitu pri prezentácii obsahu, zvládajú rôzne animácie a časovanie medzi jednotlivými slidami. Tieto riešenia neumožňujú menenie obsahu prezentácií v čase, ich obsah nie je

možné dynamicky meniť. Riešenie neposkytuje dostatočnú flexibilitu pri vytváraní obsahu užívateľmi.

2.3 Systémy digital signage

Digital signage je forma zobrazovania dynamického obsahu prostredníctvom rozličných displejov za účelom inzerovania, marketingu a predaja. Najčastejšie sa s nimi možno stretnúť v nákupných strediskách, múzeách, športových štadiónoch, kde zobrazujú reklamné informácie, informácie o produktoch a službách, navigáciu. Ako zobrazovacie médium je možné použit akúkoľvek zobrazovaciu technológiu: LCD, plazmová obrazovka, LED steny, projektor. Systémy digital signage zároveň zahŕňajú vzdialené vytváranie a upravovanie zobrazovaného obsahu [1].

Na trhu sa nachádza veľa dostupných riešení, niektoré sú zahrnuté priamo v televíznych jednotkách, ide však o špeciálne modely od rôznych spoločností, ako *LG SuperSign TV*, *Philips SmartCMS*, *Samsung MagicInfo* alebo *Sharp SDSS*. Existujú aj voľne dostupné programy s dostupnými zdrojovými kódmi, ktoré nie sú naviazané na konkrétneho výrobcu TV ako spomenuté možnosti. Ide napríklad o programy Xibo [2] alebo Screenly [3] a mnoho ďalších. Do systémov digitial signage patria najmä komplexné komerčné riešenia od spoločností ako *Cignage*, *Digital Recall* alebo *DisplayOp*. Na obrázku 2.1 môžeme vidieť zadávanie obsahu do programu od spoločnosti *Cignage*.



Obr. 2.1: Systém Cignage [4]

Spomenuté systémy ponúkajú komplexnú správu zobrazovaného obsahu, možnosti týchto systémov sú prevažne orientované na zobrazovanie obsahu za účelom inzercie a marketingu. Neumožňujú jednoduché zadávanie obsahu bez komplexného nastavovania a očakávajú správcu obsahu, ktorý určuje, aký obsah sa má zobrazovať, takže nie je možné, aby obsah tvorili nezainteresovaní užívatelia.

2.4 Vlastná aplikácia na Smart TV

Počet predaných inteligentných televizií na trhu stále rastie, cenovo sú stále dostupnejšie a je možné vyvíjať aplikácie bežiace priamo na televízii. Na trhu sú dostupné telévizie s rôznymi operačnými systémami [5]. Aplikácie je potrebné vyvíjať pre každý operačný systém televízie samostatne, takže nie je možné vytvoriť jednu aplikáciu, ktorá bude bežať na všetkých typoch televízie. V prípade výmeny televízie by aplikácia nemusela byť kompatibilná. Je možné použiť rôzne softvérové vývojové sady, ktoré vyriešia problém s kompatibilitou na rôznych platformách, ale tie sú zväčša finančne náročné. Z firemnej politiky vyplýva, že zariadenie nemôže byť pripojené do internej siete firmy.

2.5 Zhrnutie

Žiadne z možných riešení neposkytuje dostatočnú flexibilitu, či už pri tvorbe obsahu, alebo pri následnom prezentovaní na televíznej obrazovke. Sociálne siete poskytujú vhodný spôsob nahrávania obsahu, ale neumožňujú ukladať obsah v rámci firmy. Programy na prezentáciu neumožňujú flexibilné pridávanie obsahu v priebehu prezentovania. Najlepšou možnosťou je vytvoriť aplikáciu, ktorá bude zodpovedná za prezentáciu obsahu na televízii a bude umožňovať aj zadávanie a správu obsahu. Zabezpečí sa tým plná kontrola pri vytváraní obsahu, ako aj pri prezentovaní. Ďalšou výhodou je nezávislosť na službách tretích strán a možnosť ukladať obsah na lokálnych serveroch.

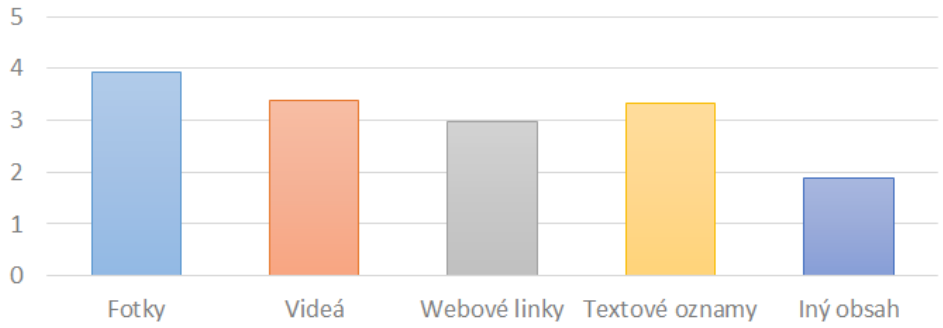
3 Analýza požiadaviek

V tejto kapitole vysvetlíme základné funkčné požiadavky webovej aplikácie na zadávanie obsahu a aplikácie na zobrazovanie obsahu na televíznej obrazovke. Najskôr je popísaný dotazník, pomocou ktorého boli zozbierané údaje od budúcich užívateľov systému, ďalej sú rozoberané prípady použitia aplikácií.

3.1 Užívateľské požiadavky na systém

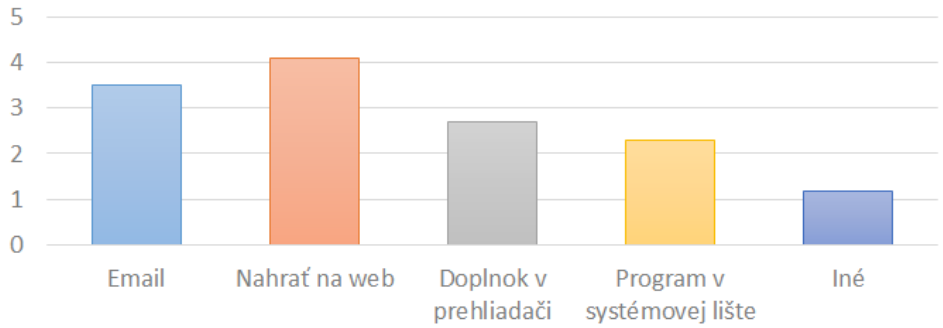
Požiadavky na používanie boli zistené formou dotazníka priamo od zamestnancov, ktorí budú systém neskôr využívať. Otázky boli smerované najmä na spôsob zdieľania a obsah, ktorý by chceli zamestanci zdieľať na TV obrazovke.

Celkovo sa do získavanie požiadaviek zapojilo 24 zamestnancov, ktorí vyplnili dotazník. V dotazníku sa nachádzali otázky s výberom možnosti, s bodovým ohodnotením ponúknutých možností, ale otázky s voľnou odpoveďou. Otázky, v ktorých sa priraďovali body jednotlivým možnostiam, mali rozsah 1 až 5 bodov, kde 5 bodov bola najpreferovanejšia možnosť. V nasledujúcich grafoch sú použité priremerné hodnoty získaných bodov. Z dotazníka vyplýva, že zamestnanci majú záujem o zdieľanie obsahu rôzneho typu. Najviac by chceli zdieľať obrázky, videá, textové príspevky a odkazy, ako vidieť na grafe 3.1.



Obr. 3.1: Výsledky dotazníka na otázku preferovaného typu zdieľaného obsahu

Ďalej sme sa dozvedeli, že najprijateľnejšie pre zamestancov je zdieľanie cez webovú stránku, na ktorú by nahrávali obsah, ktorý by bol prezentovaný na TV obrazovke, ako vidieť na grafe 3.2. Kompletné zozbierané dáta z dotazníka sa nachádzajú v prílohe.



Obr. 3.2: Výsledky dotazníka na otázku týkajúcu sa preferovanej formy zdieľania príspevkov

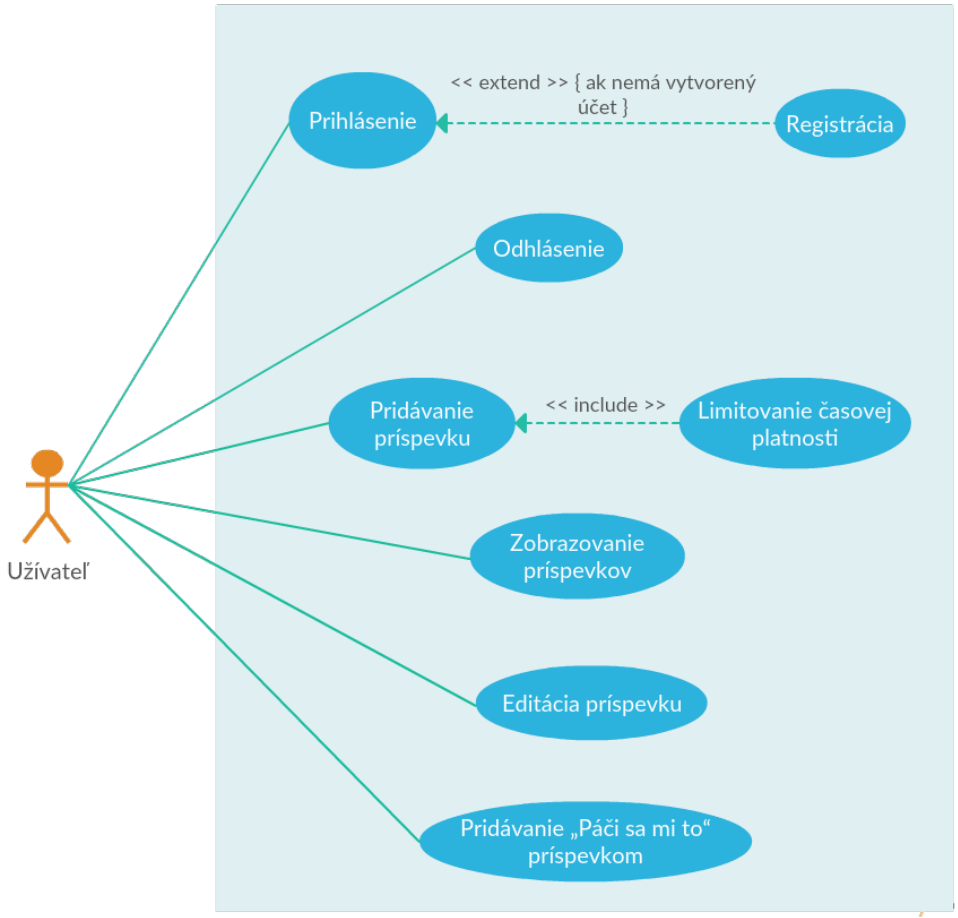
Z výsledkov dotazníka ďalej vyplýva, že je potrebné zabezpečiť jednoduché ovládanie prezentovaného obsahu, kvôli požadovanej možnosti preskočiť prehrávanie dlhého videa. Na základe získaných dáť sme rozhodli umožniť v aplikácii zdieľať obrázky, videá a textové príspevky. Zdieľanie prostredníctvom webovej stránky malo majoritnú podporu zo strany užívateľov, takže nebolo potrebné zabezpečovať iné formy nahrávania obsahu okrem spomenutej webovej stránky.

3.2 Prípady použitia aplikácií

Na základe požiadaviek boli navrhnuté a vytvorené dve aplikácie. Webová aplikícia, ktorá umožnuje zdieľanie a správu obsahu prostredníctvom webového prehliadača a aplikácia zobrazujúca takto vytvorený obsah na TV obrazovke.

Aby mohol užívateľ používať aplikáciu na zadávanie obsahu, je potrebné, aby vlastnil účet. Do systému sa môže užívateľ registrovať sám pomocou emailovej adresy. Užívateľ s vytvoreným užívateľským účtom môže vytvárať nové príspevky, editovať a mazať už existujúce príspevky. Medzi jednu z najhlavnejších možností patrí zobrazovanie

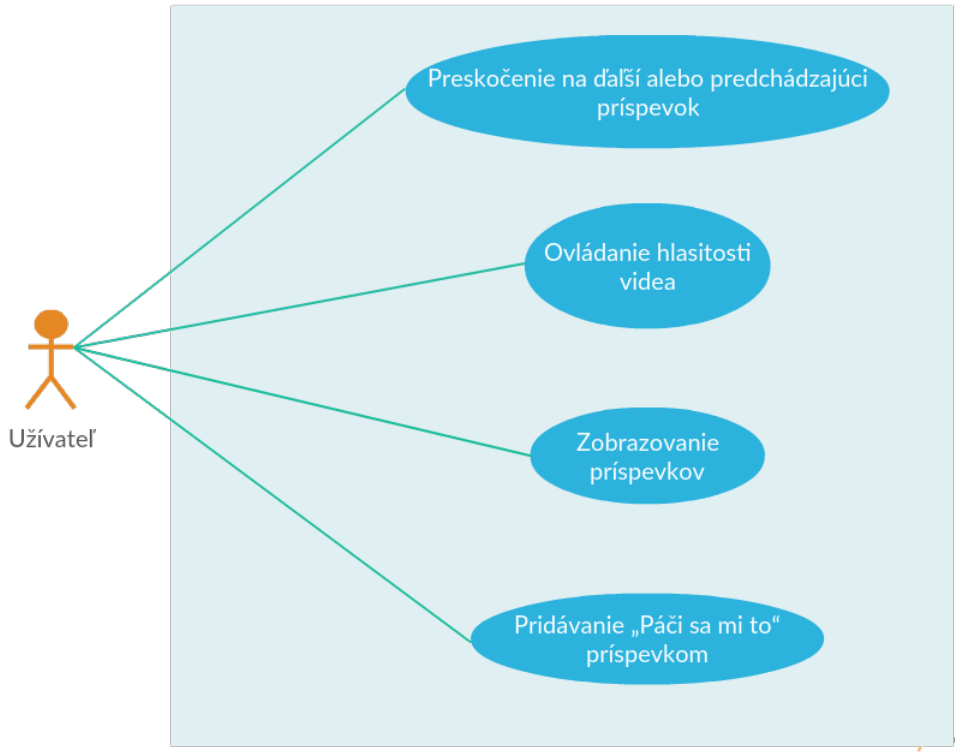
príspevkov od ostatných užívateľov. Spomenutú funkcionalitu je vidieť v nasledovnom diagrame prípadov použitia 3.3.



Obr. 3.3: Diagram prípadov použitia webovej aplikácie

Druhá aplikácia, vytvorená v rámci tejto diplomovej práce slúži na prezentáciu obsahu na TV obrazovke. Primárnym cieľom tejto aplikácie je neustále zobrazovanie textov, obrázkov a videí, ktoré užívatelia zadali prostredníctvom webovej aplikácie. Aplikácie pracuje v bežnom režime bez zásahu užívateľov a v prípade potreby umožnuje jednoduché ovládanie pomocou bezdrôtového ovládača. Užívateľ sa môže navigovať medzi zobrazovanými príspevkami dopredu a dozadu, môže meniť hlasitosť prehrávaného videa, pozastaviť prehrávané vi-

deo a jednotlivým príspevkom môže pridať „Páči sa mi to“. Popísané možnosti užívateľa sú znázornené na diagrame prípadov použitia 3.4.



Obr. 3.4: Diagram prípadov použitia aplikácie na prezentáciu obsahu

Funkčné požiadavky sú reprezentované pomocou diagramov a zachytené vo forme tzv. user stories. Každá user story zobrazuje jeden scenár prípadu použitia z užívateľského hľadiska a po zadefinovaní je možné ju implementovať. V spomenutých diagramoch použitia a user stories využívam nasledovné role:

- **prihlásený užívateľ** – užívateľ, ktorý pristupuje k webovej aplikácii, má vytvorený bežný užívateľský účet bez špeciálnych oprávnení a je prihlásený. Užívateľ má pridelenú jednu z troch rolí, ktoré sa využívajú v aplikácii.
  - **bežný užívateľ** – užívateľ, ktorý pristupuje k webovej aplikácii a má možnosť editovať a mazať iba svoje príspevky,

  - **správca obsahu** – užívateľ, ktorý pristupuje k webovej aplikácii a má možnosť editovať a mazať všetky príspevky,
  - **administrátor** – užívateľ, ktorý pristupuje k webovej aplikácii, má vytvorený špeciálny užívateľský účet, ktorý má okrem možností správcu obsahu aj možnosť meniť ostatným užívateľom role.

- **anonymný užívateľ** – užívateľ, ktorý ovláda televíznu obrazovku bez prihlásenia alebo užívateľ, ktorý používa webovú aplikáciu tiež bez prihlásenia.

3.2.1 Registrácia a prihlásenie

Užívateľ sa može registrovať pomocou platného emailu a zadaním hesla. Užívateľ sa následne prihlasuje pomocou prihlasovacích údajov. Autentizácia pomocou emailovej adresy a hesla umožňuje užívateľom používať rôzne zariadenia na používanie systému. Automatické prihlasovanie zariadení v doméne by ulahčilo zadávanie obsahu, ale nebolo by možné použiť iné zariadenie, ktoré nie je v doméne pripojené.

1. Anonymný užívateľ sa môže registrovať do aplikácie pomocou validnej emailovej adresy a prihlasovacieho hesla.

   1.a. Užívateľ sa môže registrovať len s emailovou adresou, ktorá nie je použitá v systéme.

2. Registrovaný užívateľ sa prihlasuje pomocou emailovej adresy a hesla.

   2.a. Po zadaní správnych údajov je užívateľovi zobrazená hlavná stránka s príspevkami.

3. Prihlásený užívateľ môže meniť svoje užívateľské heslo v nastaveniach profilu.

4. Užívateľ môže pri prihlásení použiť možnosť zapamätania prihlásenia a užívateľ nebude dotazovaný na prihlasovacie údaje pri ďalšom pripojení do aplikácie.

3.2.2 Vytváranie a úprava príspevku

Prihlásený užívateľ môže vytvárať nové príspevky a upravovať svoje príspevky. Príspevok je jeden samostatný objekt, ktorý môže obsahovať text, obrázky, videá a iné súbory. Príspevok môže mať limitovaný časový termín zobrazovania.

1. Prihlásený užívateľ môže vytvoriť nový príspevok.
   1.a. Užívateľ môže zadať text.
   1.b. Užívateľ môže zadať súbory - obrázky, videá a iné súbory.
   1.c. Užívateľ môže zadať čas a dátum, odkedy a dokedy sa má príspevok zobrazovať.
2. Prihlásený užívateľ môže mazať príspevky, ktoré vytvoril.
3. Ak má užívateľ priradenú rolu správcu obsahu alebo administrátora, tak môže okrem svojich príspevkov mazať a upravovať aj príspevky ostatných užívateľov.

3.2.3 Zobrazovanie príspevkov

Prihlásený užívateľ môže zobrazovať príspevky zadané v systéme. Na hlavnej obrazovke sa zobrazujú príspevky zoradené chronologicky. Účelom tejto stránky je poskytnúť užívateľom prehľad existujúcich príspevkov. Užívatelia môžu prehliadať tieto príspevky, alebo sa môžu dostať k zobrazenému príspevku, ktorý predtým videli na TV obrazovke. Užívateľ si môže zobraziť detailný príspevok na samostatnej obrazovke. Obe obrazovky umožňujú užívateľovi vyjadriť, že sa mu páči konkrétny príspevok a dať mu „Páči sa mi to“. Príspevky, ktoré sa páčia ľuďom, sa budú preferovanejšie zobrazovať na televíznej obrazovke.

1. Prihlásený užívateľ môže prehliadať príspevky na hlavnej obrazovke.
   1.a. Predvolene sa zobrazujú príspevky všetkých užívateľov.
   1.b. Užívateľ si môže zobraziť iba svoje príspevky.

2. Prihlásený užívateľ si môže zobraziť príspevok na samostatnej obrazovke s kompletnými informáciami.

3. Prihlásený užívateľ može pridať „Páči sa mi to“.

3.2.4 Ovládanie TV

Neoddelitelnou súčasťou práce je aplikácia na ovládanie obsahu na televíznej obrazovke. Užívateľom umožňuje zobrazovať multimediálny obsah, ktorý bol zadaný do systému. Užívateľ môže ovládať program pomocou infračerveného diaľkového ovládača, čo je preferovaný spôsob, prípadne za použitia myši a klávesnice.

1. Anonymný užívateľ ovláda prezentáciu obsahu.

   1.a. Užívateľ môže preskočiť na ďalší príspevok.

   1.b. Užívateľ môže ísť na prechádzajúci príspevok.

   1.c. Užívateľ môže zastaviť prezentáciu a znovu ju spustiť.

2. Anonymný užívateľ ovláda prehrávané video.

   2.a. Užívateľ može meniť hlasitosť.

   2.b. Užívateľ može vypnúť zvuk.

3. Anonymný užívateľ môže pridať príspevku anonymné „Páči sa mi to“. Nebolo by vhodné, aby sa užívatelia prihlasovali kvôli tejto funkcionalite. Preto je možné zadávať „Páči sa mi to“ opakovane na rovnaký príspevok. Predpokladá sa, že užívatelia nebudú túto možnosť zneužívať.

Spomenutý diaľkový ovládač je zobrazený na obrázku 3.5, k ovládaniu aplikácie sa používajú vyznačené tlačidlá. Šípky doľava a doprava ovládajú navigáciu medzi príspevkami, umožňujú prejsť na predchádzajúci a nasledujúci príspevok. Šípky hore a dole ovládajú hlasitosť prehrávaného videa. Červené tlačidlo „OK“ slúži na pozastavenie a opätovné spustenie prehrávania príspevkov. Posledné vyznačené tlačidlo, symbol „*“, umožnuje pridať príspevkom „Páči sa mi to“. Okrem diaľkového ovládača je možné použiť štandardnú klávesnicu. Ovládanie na klávesnici je podobné ako na diaľkovom

ovládači, šípky majú rovnakú funkciu, klávesa *medzerník* nahrádza tlačidlo „OK“ a umožňuje pozastaviť a spustiť prehrávanie obsahu. Namiesto tlačidla „*“ sa používa klávesa „L“, ktorá umožňuje pridať príspevkom „Páči sa mi to“.



Obr. 3.5: Diaľkové ovládanie používané na ovládanie aplikácie prezentujúcej obsah na TV obrazovke

3.3 Nefunkčné požiadavky

Ako bolo spomenuté v predchádzajúcej časti, výsledné riešenie zadávanie a prezentovania sa skladá z dvoch aplikácií. Aplikácia na zobrazovanie a ovládanie obsahu počas prezentácie na televíznej obrazovke, ktorý bol vytvorený pomocou aplikácie na zadávanie obsahu. Aplikácia na zadávanie obsahu je webová aplikácia, ktorá beží na serveri v sieti. Zodpovednosťou tejto aplikácie je umožnovať zadávanie obsahu do systému, jednoduchú prezentáciu a sprístupňovanie obsahu prostredníctvom REST API. Pre správnu funkciu zadávania obsahu je potrebné, aby bol server neustále dostupný. Aplikácia na prezentovanie obsahu na televízii obsah získaný zo servera spracováva a zobrazuje vo forme vhodnej na prezentáciu. Stiahnuté údaje si ukladá na disk, aby nedochádzalo k opakovanému prenášaniu dát a zbytočnému zahlteniu siete. V prípade, ak sa pridajú nové príspevky, klient túto informáciu získa prostredníctvom API a dáta spracuje. Klient musí byť dostupný len v prípade prezentácie obsahu na televízii.

3.4 Prehľad technológií

Do internej siete zadávateľa práce je možné pripojiť len zariadenie, na ktorých beží operačný systém Microsoft Windows 7. Zo zadania ďalej vyplýva, že sú použité technológie postavené na platforme Microsoft .NET. Najčastejšie používaný programovací jazyk v tomto prostredí je jazyk C#. Na platforme .Net sa používajú dve majoritné technológie, pomocou ktorých sa tvorí užívateľské rozhranie desktopových aplikácií. Staršia technológia, ale stále široko rozšírená, je WinForms – Window Forms a novšia technológia, ktorá podporuje lepšie oddelenie logiky aplikácie od užívateľského rozhrania, je WPF – Windows Presentation Foundation.

- **WinForms** je prvá technológia, ktorá bola používaná na vytváranie grafického rozhrania na platforme .Net. Stále je plne funkčná na jednoduchšie aplikácie, ale v nových moderných aplikáciách sa od nej opúšťa.
- **WPF** je aktuálne odporúčaná technológia na vytváranie moderných aplikácií. Využíva jazyk XAML na návrh užívatelského rozhrania a umožňuje lepšie oddelenie prezentačnej vrstvy od logiky aplikácie a biznis vrstvy.

Podobná situácia v rámci prezentačnej vrstvy je aj vo webových technológiach .Net. Staršia technológia, už nevyvíjaná, je ASP.NET WebForms, v ktorej sa aplikácie vytvárajú veľmi podobne ako v desktopovej technológii WinForms. Novšia a aktívne vyvíjaná technológia je ASP.NET MVC, ktorá umožnuje väčšiu kontrolu nad výsledným správaním aplikácie a podporuje oddelenie prezentačnej logiky od biznis logiky. Spomenuté webové technológie sú vysvetlené v nasledujúcich bodoch:

- **Webforms** – využitím technológie WebForms je možné vytvárať webové stránky veľmi podobne, ako sú tvorené desktopové aplikácie prostredníctvom WinForms. Obsahujú podobné ovládacie komponenty, ako sú napríklad tlačítka a popisy. Tímto komponentom je možné priradiť rôzne vlastnosti a reagovať na udalosti, ktoré vyvolávajú. Aplikácie sa správajú veľmi podobne ako desktopové, ale na pozadí sa nachádza zložitejšia logika,

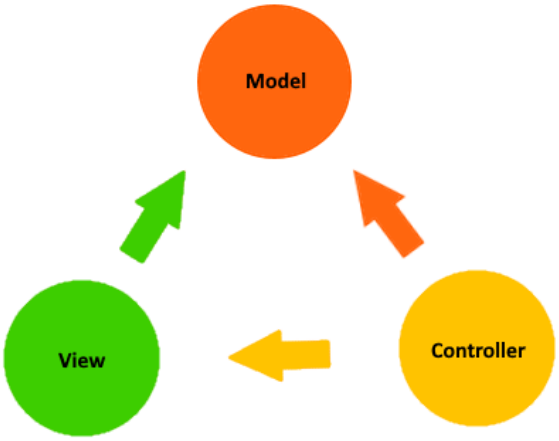
ktorá udržiava stav komponentov a aplikácie nad bezstavovým protokolom HTTP. Stav môže reprezentovať vyplnené hodnoty vo formulári alebo uložené kroky v sprievodcovi. Server nemá uložené informácie o týchto hodnotách. Stav sa ukladá pomocou ViewState (stav zobrazenia), čo je uloženie všetkých potrebných hodnôt do skrytého poľa vo formulári. Skryté poľe sa prenáša spolu s ostatnými údajmi pri odoslaní formuláru na server. Server obnoví stav aplikácie, vykoná požadovanú úlohu a stav opäť prevedie do ViewState. Týmto spôsobom je možné udržiavať stav aplikácie.

Hlavné výhody WebForms:

- Podporuje model udalostí, ktorý udržiava stav nad protokolom HTTP. Poskytuje veľké množstvo prvkov server controls, elementy vykonávajúce kód na strane servera.
- Používa view state, umožňuje jednoduchšie udržiavanie stavu komponentov view.
- Technológia vhodná pre malé tímy, ktorým umožňuje využiť velké množstvo vstavaných komponentov a urýchlit tak vývoj.
- Menej komplexná technológia na vývoj, komponenty sú úzko integrované a zvyčajne vyžadujú menej kódu ako model MVC.

- MVC – je novšia technológia, ktorá slúži na vytváranie webových aplikácií. Využíva koncept Model – View – Controller, ktorý je hneď na prvý pohľad odlišný od WebForms. Hlavným znakom je rozdelenie prezentačnej vrstvy od logiky aplikácie. Koncept MVC rozdeluje aplikáciu na tri hlavné časti, ako vidieť na obrázku 3.6.

  - Model – reprezentuje dáta.
  - View – prezentuje dáta získané z modelu.
  - Controller – spracováva požiadavky a sprístupňuje dáta do prezentačnej vrstvy.



Obr. 3.6: Vzťahy komponentov MVC

Model v aplikácii MVC má za úlohu reprezentovanie stavu aplikácie a biznis logiku vrátane operácií, ktoré sú nad ňou vykonávané. Biznis logika zapúzdruje model s implementačnou logikou na perzistovanie stavu. Model typicky zahŕňa dáta, ktoré sa zobrazujú v časti View.

View zobrazuje dáta z modelu v užívateľskom prostredí. V časti View by mala byť len minimálna logika, ktorá súvisí len zo zobrazovaním. Všetká ostatná logika sa má nachádzať v modeli.

Controller je komponent, ktorý spracováva užívateľský vstup, pracuje s modelom a rozhoduje, ktorý view sa má zobraziť. Podobne je jeho úlohou rozhodovať, ktorý model sa vyberie a ako sa použije vzhľadom na interakciu užívateľa [6].

Rozdelenie na jednotlivé komponenty umožnuje súčasnú prácu viacerých vývojárov, tým môže nezávisle pracovať na front-end časti a back-end časti. Rozdelenie takisto prispieva k znovupoužiteľnosti komponentov, napríklad môže byť vytvorený jeden model, ktorý bude využívaný vo viacerých views. Komponenty umožňujú využívať vývoj riadený testovaním, takzvaný test-driven development (TDD) [7]. Je to agilná programovacia technika, pri ktorej sa píšu testy pred samotnou funkcionalitou. Naj-

skôr sa definuje čiastočná funkcionalita, pre ktorú sa napíše test, ktorý ju overuje. Implementuje sa funkcionalita požadovaná v teste a nasleduje refaktoring. Keď prechádzajú všetky testy, opätovne sa pridajú testy definujúce novú funkcionalitu a pokračuje sa v cykle, pokiaľ nemáme kompletnú požadovanú funkcionalitu definovanú prechádzajúcimi testami.

Hlavné výhody MVC [8]:

- rozdelenie funkcionality medzi model, view a controller,
- nepoužíva view state a server-base forms, čo umožňuje plnú kontrolu nad aplikáciou,
- lepšia podpora vývoja riadeného testovaním,
- vhodné pre veľke tímy a web designérov, ktorí potrebujú veľkú kontrolu nad správaním aplikácie.

Medzi aplikáciami je potrebné prenášať dáta, a preto je potrebné, aby navzájom komunikovali. Medzi najčastejší spôsob komunikácie s webovými službami na internete sa používa architektonický štýl REST. Protokol SOAP, ktorý využíva prenos správ v XML, sa používa často v systémoch, ktoré sú zamerané viac na vykonávanie programu ako na prenos dát. Protokol SOAP využíva vzdialené volanie procedúr (RPC).

4 Aplikácia zobrazujúca obsah na TV

Implementačná časť práce je rozdelená na dve samostatné kapitoly, v ktorých sa zaoberám dvomi vytvorenými aplikáciami, ktoré sú popísané v kapitole 3. Ako prvá je popísaná aplikácia zobrazujúca obsah na TV obrazovke. V úvode tejto kapitoly popíšem architektúru komunikácie medzi aplikáciami a ďalej sa budem venovať samotnej aplikácii zobrazujúcej obsah na TV vrátane použitých technológií a implementácie.

4.1 Prenos dát

Komunikácia medzi aplikáciami je zabezpečená pomocou RESTful webovej služby. Webové služby sú technológie, ktoré povolujú vytváranie spojení medzi koncovými uzlami, typicky sa používajú v internete. REST (REpresentational State Transfer) je architektonický štýl pre distribuované aplikácie. Webové služby REST sú jedna z možností, ako zabezpečiť interoperabilitu medzi zariadeniami v sieti alebo v internete. Služby REST využívajú textovú reprezentáciu dát, s ktorými je možné ďalej manipulovať. Základná vlastnosť webových služieb REST je bezstavovosť, služby neuchovávajú žiadne informácie o stave.

Architektúra REST bola definovaná v roku 2000 v dizertačnej práci Roya T. Fieldinga „Architectural Styles and the Design of Network-based Software Architectures“[9]. Fielding vyvinul architektonický štýl REST paralelne s protokolom HTTP 1.1, založený na existujúcom návrhu protokulu HTTP 1.0.

Základné obmedzenia architektúry REST:

1. Klient-server model – v modeli je jeden uzol, server, ktorý má za úlohu obsluhovať niekoľko klientov, tento model je známy z internetu.

2. Bezstavovost (stateless) – server si neuchováva žiadne informácie o spojení, tieto informácie su predávané v rámci požiadaviek a udržiavanie spojenia má na starosti klient. Hlavnými výhodami je škálovatelnosť, server nemusí udržiavať spojenia, spracovávanie požiadaviek je jednoduchšie a server ich zvládne spracovávať rýchlejšie.

3. Vyrovnávacia pamäť – toto obmedzenie zvyšuje efektívne využívanie siete. Odpovede na požiadavky môžu byť označené, či sa majú ukladať vo vyrovnávacej pamäti alebo nie. Keď je odpoveď uložená vo vyrovnávacej pamäti, klient má možnosť využiť uložené dáta neskôr pre rovnaké požiadavky.

4. Uniformné rozhranie – hlavná vlastnosť, ktorá oddeluje REST od iných sieťovo založených štýlov, je dôraz na uniformné rozhranie medzi komponentmi. Implementácie sú oddelené od služieb, ktoré poskytujú, čo podporuje nezávislý vývoj.

5. Vrstvený systém – klient nedokáže rozlíšiť, či je pripojený priamo ku koncovému uzlu alebo je pripojený cez prostredníka. Prostredníkom môže byť napríklad pamäť cache alebo load balancer. Prostrednícke uzly zvyšujú škálovateľnosť, ale zároveň zanášajú oneskorenie. Load balancer je zariadenie alebo program, ktorého úlohou je rozdelovanie požiadaviek na viaceré uzly podľa definovaných pravidiel za účelom zvýšenia dostupnosti a spolahlivosti aplikácie.

6. Kód na vyžiadanie – je voliteľné obmedzenie architektúry. Server môže zasielať klientovi vykonateľný kód vo forme apletov alebo skriptov, ktoré môžu rozšíriť funkcionalitu klienta. Obmedzenie zjednodušuje klientov redukovaním funkcionalít, ktoré je potrebné implementovať, umožnuje klientom stiahnuť novú funkcionalitu aj po ich nasadení. Zároveň toto obmedzenie redukuje viditeľnosť a preto je len nepovinné.

Webové služby, ktoré rešpektujú spomenuté obmedzenia, sa nazývajú RESTful webové služby. RESTful webové služby sa v dnešnom svete riadia nasledovnými návrhovými princípmi:

- používajú HTTP metódy,
- sú bezstavové,
- vystavujú URI adresy, ktoré zachovávajú adresárovú štruktúru,
- prenášajú dáta vo formáte XML, JSON alebo oboch.

RESTful webové služby využívajú metódy HTTP spôsobom, aký bol zamýšlaný protokolom definovaným v RFC 2616 [1]. Napríklad metóda HTTP GET je definovaná ako metóda, ktorá je používaná klientom za účelom získania zdrojov a prenosom dát zo serveru. Služby REST využívajú mapovanie medzi operáciami CRUD a metódami HTTP. Operácie CRUD sú 4 základné operácie: vytvoriť, čítať, aktualizovať a zmazať definované nad nejakým úložiskom.

- POST - vytváranie zdrojov na serveri,
- GET - získavanie zdrojov,
- PUT - menenie stavu zdrojov,
- DELETE - mazanie zdrojov.

Niektoré webové služby nevyužívajú metódy HTTP tak, ako je definované v RFC a napríklad menia stav zdrojov alebo vytvárajú nové zdroje na serveri pri použití metódy GET, napríklad: *GET /add?text=HelloWorld*. Webové služby, ktoré používajú metódy HTTP nesprávne, nie sú RESTful webové služby.

Návrh API

Aplikácie navrhnuté v rámci tejto diplomovej práce medzi sebou komunikujú prostredníctvom RESTful webovej služby. Základná entita, ktorá je medzi aplikáciami prenášaná, je objekt reprezentujúci príspevok – *Post*. Objekt JSON Post obsahuje všetky údaje, okrem binárnych dát, ktoré sú reprezentované pomocou URI odkazov uložených v poli Attachments. V prípade potreby binárnych dát je možné ich získať prostredníctvom odkazov URI.

1. `https://www.ietf.org/rfc/rfc2616.txt`

4.2 Architektúra aplikácie zobrazujúcej obsah na TV

Aplikácia je rozdelená do jednotlivých modulov, aby umožnila jednoduché udržiavanie, flexibilitu a oddelenie zodpovedností. Rozdelenie do modulov je nasledovné:

- Doménové modely
  - modul obsahuje entity, ktoré reprezentujú dáta uložené v databázových tabuľkách.
- ViewModels
  - v module sa nachádzajú komponenty ViewModel, ktoré používajú príslušné komponenty View,
  - komponenty obsahujú dáta z doménového modelu vo forme vhodnej na prezentáciu,
  - zodpovednosťou modulu je spracovávať vstupy od užívateľov a vykonať príslušné akcie.
- Views
  - v module sa nachádzajú komponenty View, ktoré slúžia na zobrazovanie užívateľského rozhrania,
  - komponenty zobrazujú dáta z príslušných komponentov ViewModel.
- Services – služby
  - modul služieb implementuje biznis funkcionalitu aplikácie,
  - služby pracujú s doménovými modelmi.
- UserControls – ovládacie prvky
  - modul obsahuje vlastné ovládacie prvky,
  - ovládacie prvky je možné opakovateľne používať v užívateľskom prostredí.

4.3 Ukladanie dát

Aplikácia si ukladá dáta získané z webovej služby lokálne na disk. Dáta získané prostredníctvom REST API sú ukladané do databázy SQLite bez binárnych dát. Binárne dáta sú pre jednoduchšie a rýchlejšie spracovanie ukladané vo forme súborov v adresároch na disku. Cesta k súborom na disku je uložená v databáze.

Pri ukladaní dát do databáze sa využíva Entity Framework, ktorý využíva ORM – mapovanie objektov z programovacieho jazyka na relačné dáta uložené v relačnej databáze [10]. Konverziu medzi databázou a objektami zabezpečuje Entity Framework, takže nie je potrebné implementovať logiku pracujúcu s databázou. Pri získavaní dát sa v programe používa LINQ – dotazovací jazyk implementovaný v programovacom jazyku, ktorý vnútorne využíva dotazy SQL, ale vracia silne typované objekty [11].

Entity Framework využíva tzv. code-first prístup, pri ktorom sa najskôr vytvorí model v kóde a následne sa vytvorý databázová schéma až za behu. Tento prístup odstraňuje zodpovednosť udržiavať databázovú schému synchronizovanú s modelom. Vývojári sa môžu sústrediť na vytváranie doménových modelov bez toho, aby museli neustále vytvárať SQL príkazy na aktualizáciu databázy. Jednotlivé databázové zmeny sú ukladané do migrácií. Pri vytváraní migrácie sa generuje kód, ktorý aktualizuje databázu na základe existujúcej schémy a nových zmien, ktoré boli vytvorené v kóde a ešte nie sú reprezentované v databáze.

4.4 Windows Presentation Foundation

Implementácia užívateľského rozhrania využíva technológiu „Windows Presentation Foundation“, ktorá je odporúčaná spoločnosťou Microsoft na budovanie desktopových aplikácií v prostredí operačného systému Microsoft Windows.

Pri vytváraní užívateľského rozhrania pomocou WPF sa používa jazyk XAML. Jazyk XAML je deklaratívny značkovací jazyk, súbory XAML sú vlastne súbory XML, ktoré majú koncovku .xaml. Zjednodušuje vytváranie užívateľského rozhrania, ktoré nie je vytvárané v C# kóde, ale je tvorené XAML elementami, podobne ako je to v prípade jazyka HTML. Na rozdiel od iných značkovacích jazykov, XAML nie je interpretovaný jazyk, ale jeho elementy reprezentujú instancie objektov z kódu.

Štandardne XAML element definuje vytváraný objekt, ktorý môže mať rôzne vlastnosti a môže obsahovať ďalšie elementy. Nasledujúca ukážka XAML kódu 4.1 reprezentuje tri spôsoby zápisu vlastností XAML elementov. Prvé tlačidlo na riadku 2 ukladá hodnoty do atribútov. Pri ukladaní textových hodnôt dochádza k automatickej konverzii na požadovaný typ.

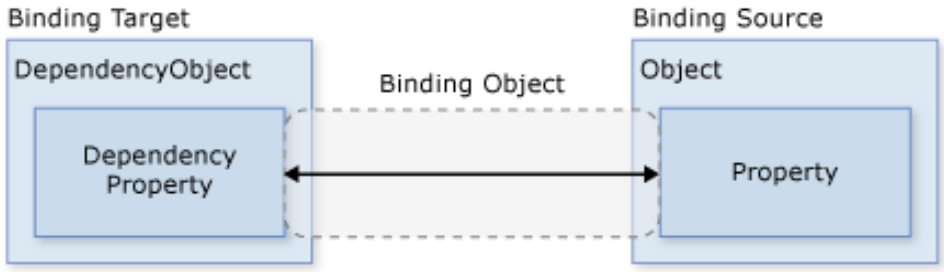
Listing 4.1: Explicitný zápis vlastností elementu XAML

```
1  <StackPanel>
2    <Button
3      Background="Blue"
4      Foreground="Red"
5      Content="First Button" />
6    <Button>Second Button</Button>
7    <Button Content="Third Button">
8      <Button.Foreground>
9        <SolidColorBrush Color="Blue"/>
10         </Button.Foreground>
11     <Button.Foreground>
12       <SolidColorBrush Color="Red"/>
13     </Button.Foreground>
14   </Button>
15 </StackPanel>
```

Druhé tlačidlo na riadku 6 využíva uloženie hodnoty do dcérskeho elementu. Pre využitie dcérskeho elementu je potrebné definovať obsahový element na rodičovskom type pomocou triedy *ContentPropertyAttribute*. Podobne je využitý tento spôsob aj pri elemente StackPanel, kde je obsahovou vlastnosťou definovaná vlastnosť Children. Tretí spôsob zápisu na riadku 7 je explicitný, používa sa pri používaní zložitejších typov a typov, pre ktoré nie je definovaná typová konverzia. Prvé a tretie tlačidlo používa rovnaké vlastnosti pre farbu popredia a pozadia, ale v prvom prípade je použitá automatická konverzia.

Dátové naviazanie

WPF poskytuje jednoduchý a konzistentný spôsob na prezentáciu a interakciu s dátami z modulu v užívateľskom prostredí. Elementy môžu byť naviazané na rôzne dátové zdroje, môžu to byť objekty z CLR alebo iné elementy XML. Naviazanie dát medzi elementami a zdrojom sa nazýva „Data Binding“. Data Binding umožnuje spojenie vrstvy užívateľského rozhrania s biznis vrstvou. Data Binding pre svoju podporu vyžaduje implementovanie rozhrania *INotifyPropertyChanged* na zdrojových triedach. Naviazanie dát môže byť jednosmerné, umožnuje prenos hodnôt zo zdroja do elementu, alebo obojsmerné, kde sa pri zmene hodnoty v elemente zmena propaguje do zdroja. Jednosmerné naviazanie sa využíva pre zobrazovanie dát, napríklad pri zobrazovaní tabuľky alebo popisku, obojsmerné naviazanie sa využíva napríklad pri textových poliach, kde sa hodnoty prenášajú z užívateľského rozhrania do modelu. Pri použití naviazania je potrebné v súbore XAML nastaviť dátový kontext a zdroj naviazania. Obojsmerné naviazanie je schematicky znázornené na obrázku 4.1.



Obr. 4.1: Data Binding [12]

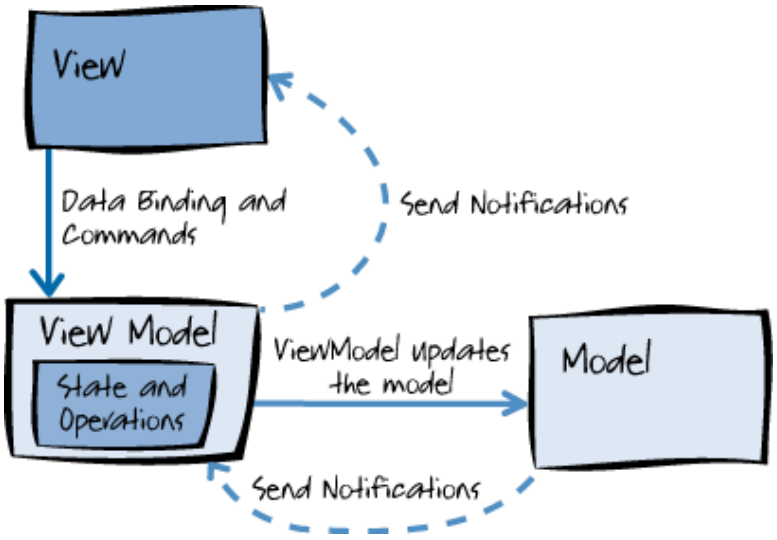
Na nasledovnej ukážke kódu 4.2 je ukázané nastavovanie dátového kontextu pre rodičovský element a všetky dcérske elementy. *StackPanel* obsahuje nastavanie dátového kontextu na riadku 3, ktorý zdedia všetky dcérske elementy a následne na riadku 5 tlačidlo naviaže vlastnosť *Content* na premennú *ButtonText* instancie *viewModel* typu *MainWindowViewModel*.

Listing 4.2: Naviazanie kontextu

```
1 <StackPanel>
2   <StackPanel.DataContext>
3     <vm:MainWindowViewModel x:Name="viewModel" />
4   </StackPanel.DataContext>
5   <Button Content={Binding ButtonText} />
6 </StackPanel>
```

4.5 Vzor Model-View-ViewModel

Pri vytváraní aplikácií je snaha oddeľovať vrstvu užívateľského rozhrania od biznis vrstvy. Vo vrstve užívateľského rozhrania by sa nemal nachádzať žiaden kód, ktorý rieši logiku aplikácie. Pri používaní WPF sa využíva vzor MVVM, Model-View-ViewModel. Vzor MVVM je vychádza z návrhového vzoru MVC, ktorý sa používa vo webových aplikáciách. Komponenty View a Model majú rovnakú funkciu, ako pri použití vzoru MVC. Na rozdiel od MVC sa používa ViewModel, ktorý je jediný komponent, s ktorým komunikuje View. Zvyčajne je každý komponent View mapovaný na vlastný komponent ViewModel. Komponent ViewModel obsahuje dáta, ktoré sa zobrazujú v komponente View. ViewModel obsahuje logiku, získava a spracováva dáta. Na spojenie View a ViewModel sa používa Data Binding. Pri použití vzoru MVVM je možné okrem naviazania dát naviazať aj príkazy. V aplikácii sa potom pri akciách nevyužíva spracovanie udalostí, ako je to pri štandardných aplikáciach v technológii WebForms, ale naviazanie na príkaz v komponente ViewModel. Na obrázku 4.2 je vzor MVVM, na ktorom sú znázornené komponenty a komunikácia, ktorá medzi nimi prebieha.



Obr. 4.2: Vzor MVVM [13]

MVVM Light Toolkit

Pre implementáciu vzoru MVVM bola použitá knižnica MVVM Light Toolkit. Ide o jednoduchú knižnicu, ktorá umožnuje jednoduchšiu a rýchlejšiu implementáciu za použítia vzoru MVVM. Z knižnice sú použité nasledovné moduly:

- **RelayCommand** – umožnuje jednoduchšie vytváranie príkazov v komponente ViewModel,
- **Messenger** – slúži ako zbernica na zasielanie správ medzi jednotlivými komponentmi,
- **ViewModelBase** – umožnuje triedam, ktoré dedia z ViewModelBase, jednodušiu implementáciu rozhrania INotifyPropertyChanged.

4.6 AutoMapper

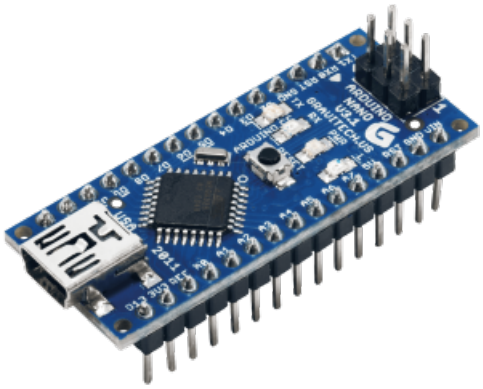
V aplikácii je potrebné prevádzať jeden typ objektu na iný, ide najmä o objekty reprezentujúce prenášané objekty JSON, objekty používané v kompenentoch ViewModel a doménové objekty, s ktorými pracuje Entity Framework. Na transformáciu objektov je použitá knižnica AutoMapper, ktorá zabezpečuje mapovanie medzi objektami. Knižnica je jednoduchá a zverejnená pod licenciou MIT.

4.7 Ďiaľkové ovládanie

Aplikácia zobrazuje obsah, ktorý automaticky cyklí bez potreby zásahu užívateľa. Užívateľ môže vykonať jednoduché akcie, ktoré nepožadujú komplexné ovládacie prvky, ako sú myš a klávesnica, ktoré je komplikované vhodne umiestniť a používať. Aplikácia využíva ovládanie pomocou infračerveného ovládača, ktorý je znázornený na obrázku 3.5 v časti 3.2.4, a prijímača. Je potrebné splniť podmienku priamej viditeľnosti medzi ovládačom a prijímačom, inak je možné umiestniť ovládač na akékoľvek vhodné miesto. Nespornou výhodou infračerveného ovládania v porovnaní s inými formami ovládanie je bezdrôtová technológia a zároveň dlhá životnosť na batériu.

Hodnoty infračerveného signálu sú prijímané pomocou infračerveného senzoru, ktorý je pripojený k zariadeniu Arduino Nano [14]. Arduino je elektronická platforma s otvoreným zdrojovým kódom založená na jednoducho použiteľnom hardvéri a softvéri. Dosky Arduino sú schopné prijímať rôzne vstupy pomocou senzorov, ako sú svetelné senzory, tepelné čidlá, pohybové senzory, a signál z nich transformovať pomocou výstupných zariadení na výstup, ako zapnutie LED diód, otáčanie krokových motorov, zaslanie SMS. Riadiacou jednotkou je mikrokontrolér, ktorý ovláda celé zariadenie. Arduino sa programuje v programovacom jazyku C a C++ vo vývojovom prostredí Arduino Software. V aplikácii je použité zariadenie Arduino Nano, ktoré je menšou verziou štandardného Arduino.

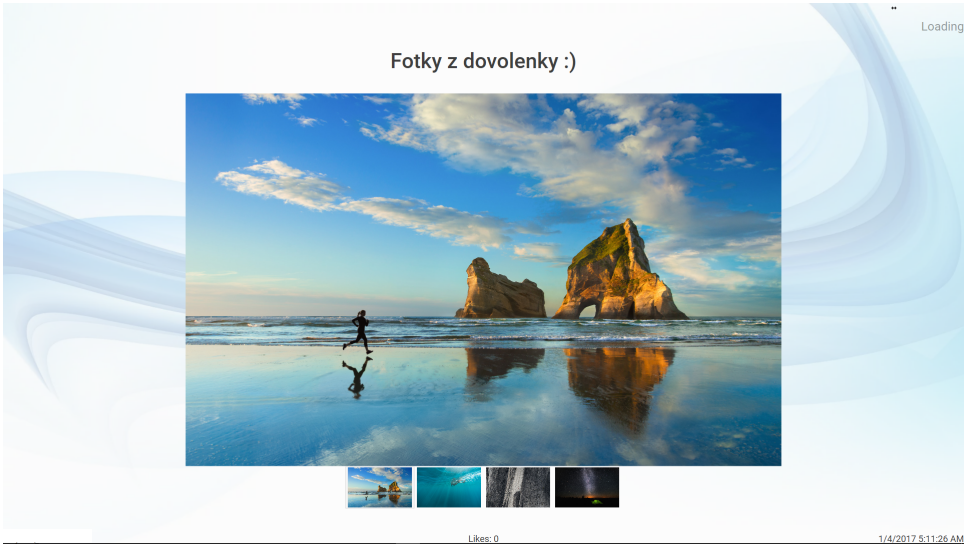
Použité Arduino Nano znázornené na obrázku 4.3 je pripojené k počítaču pomocou kábla USB. Arduino obsahuje program, ktorý spracováva signál z infračerveného prijímača a prenáša ho do počítača cez virtuálny sériový port. Aplikácia v počítači na základe dát prijatých zo sériového portu vykoná požadovanú akciu.



Obr. 4.3: Vývojová doska Arduino Nano [15]

4.8 Užívateľské rozhranie aplikácie na zobrazovanie obsahu na televízii

Aplikácia dokáže plniť svoju úlohu aj bez akéhokoľvek zásahu užívateľa a podporuje základné operácie spomenuté v časti 3. Snaží sa zachovať jednoduché a intuitívne ovládanie, aby s ňou dokázal pracovať každý užívateľ bez potrebného školenia a aby ovládacie prvky nepôsobili rušivo. Pri zobrazovaní príspevku s obrázkami sa jeden z obrázkov zobrazuje zväčšený a ostatné sa zobrazujú ako miniatúry pod ním. Na nasledujúcom obrázku 4.4 je vidieť, ako vyzerá aplikácia pri zobrazovaní obrázkov.



Obr. 4.4: Televízna obrazovka s príspevkom obsahujúcim obrázky

V prípade príspevku obsahujúceho video sa zobrazí veľmi podobná obrazovka s prehrávaným videom, ktoré je možné ovládať.

5 Webová aplikácia

V predchádzajucej kapitole bola popísaná aplikácia zobrazujúca obsah na TV obrazovke. V tejto kapitole popíšem architektúru aplikácie zabezpečujúcej zadávanie, ukladanie a spravovanie obsahu, ďalej sú spomenuté použité technológie a postupy implementácie.

5.1 Architektúra aplikácie

Podobne ako je rozdelená aplikácia zobrazujúca obsah na televízii, je rozdelená aj webová aplikácia na správu obsahu. Aplikácia obsahuje jednotlivé moduly, čo umožnuje jednoduché udržiavanie, flexibilitu a oddelenie zodpovedností. Rozdelenie do modulov je nasledovné:

- Doménové modely
  - modul obsahuje entity, ktoré reprezentujú dáta uložené v databázových tabuľkách.
- Controllers
  - modul spracováva užívateľove požiadavky a následne zobrazuje príslušné komponenty View,
  - obsahuje aj komponenty zodpovedné za sprístupnenie REST API.
- Views
  - v module sa nachádzajú komponenty View, ktoré slúžia na zobrazovanie užívateľského rozhrania,
  - komponenty vytvárajú užívateľské prostredie a generujú príslušný HTML kód.
- Services – služby
  - modul služieb implementuje biznis funkcionalitu aplikácie,
  - služby pracujú s doménovými modelmi.
- Content

  - modul obsahuje kaskádové štýly, obrázky a ikony,
  - obsahuje vlastné súbory CSS a aj súbory CSS knižníc.

- Scripts
  - modul obsahuje použité skripty jazyka JavaScript.

5.2 Ukladanie obsahu

V práci je použitá databáza MS SQL, čo vyplýva z použitých technológií spoločnosti Microsoft. Databáza umožnuje jednoduché spojenie s ostatnými technológiami spoločnosti Microsoft, ktoré sú použité v práci. Obsah príspevkov sa ukladá do databáze okrem binárnych dát súborov. Tie sú ukladané na disk a do databázy sa ukladá len adresa s umiestnením súboru. Takéto riešenie je vhodnejšie ako ukladať súbory v databázi, nakoľko sa počíta s častým používaním multimediálnych súborov a odpadá tak záťaž databazy.

5.3 Knižnica Hangfire

Aplikácia umožnuje postupné nahrávanie súborov k ešte nevytvorenému príspevku, tieto súbory sú ukladané ako dočasné a pri vytvorení príspevku sa z nich stanú súbory, ktoré sa zmažú až pri mazaní príspevku. O mazanie dočasných súborov, ku ktorým nebol vytvorený príspevok, sa stará knižnica Hangfire, ktorá zaobstaráva spúštanie odložených a naplánovaných úloh. Raz za 24 hodín sa spustí úloha, ktorá premaže dočasné súbory, ktoré sú staršie ako 1 deň. Týmto riešením je umožnené pohodlne nahrávať obrázky a videá k príspevkom a zároveň sa dočasné dáta pravidelne mažú, keď už nie sú potrebné.

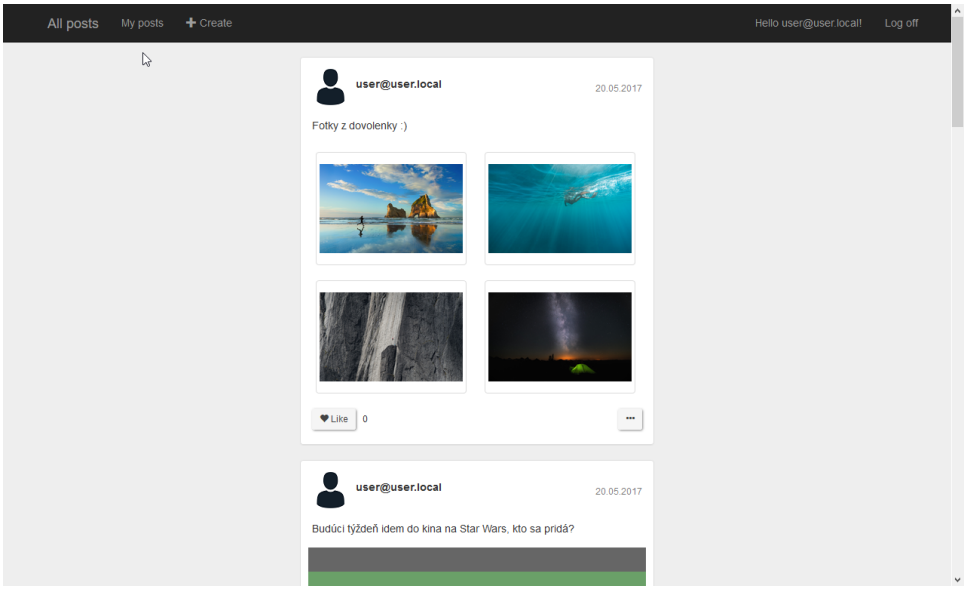
5.4 IoC – Simple Injector

Inversion of Control, čiže obrátené riadenie znamená, že tok programu je opačný oproti klasickému procedurálnemu programovaniu. Komponent nie je zodpovený za získavanie zdrojov, očakáva, že užívateľ komponent dodá všetky potrebné zdroje. Návrhový princíp IoC podporuje dobrý návrh a zachováva princíp jednej zodpovednosti.

Komponent by nemal mať zodpovednosť za vytváranie instancií závislostí, túto zodpovednosť predáva inému objektu a zároveň redukuje závislosti medzi komponentmi [16]. Vkladanie závislostí (DI – Dependency Injection) je variant obráteného riadenia, komponenty prostredníctvom svojho API informujú o tom, aké zdroje potrebujú. O zapezpečenie týchto zdrojov sa postará iný komponent, nazývaný IoC kontajner. Kontajner je zodpovedný za priraďovanie zdrojov do príslušných komponentov a takisto aj životný cyklus týchto zdrojov. Pre vkladanie závislosí v aplikácii je použitý projekt Simple Injector, ktorý spĺňa požiadavky aplikácie. Je aktívne vyvíjaný, má dobrú dokumentáciu a je jednoduchý na použitie. Vkladanie závislostí je použité pri priraďovaní služieb do komponentov ViewModel.

5.5 Užívateľské rozhranie webovej aplikácie

Webová aplikácia slúži primárne na zadávanie obsahu pre televíziu a prípadné zobrazenie obsahu, ktorý ľudí zaujal na televízii a chcú si ho pozrieť detailnejšie. Po prihlásení sa užívateľovi zobrazí hlavná obrazovka, ktorá obsahuje všetky príspevky zoradené chronologicky, ako vidieť na obrázku 5.1.

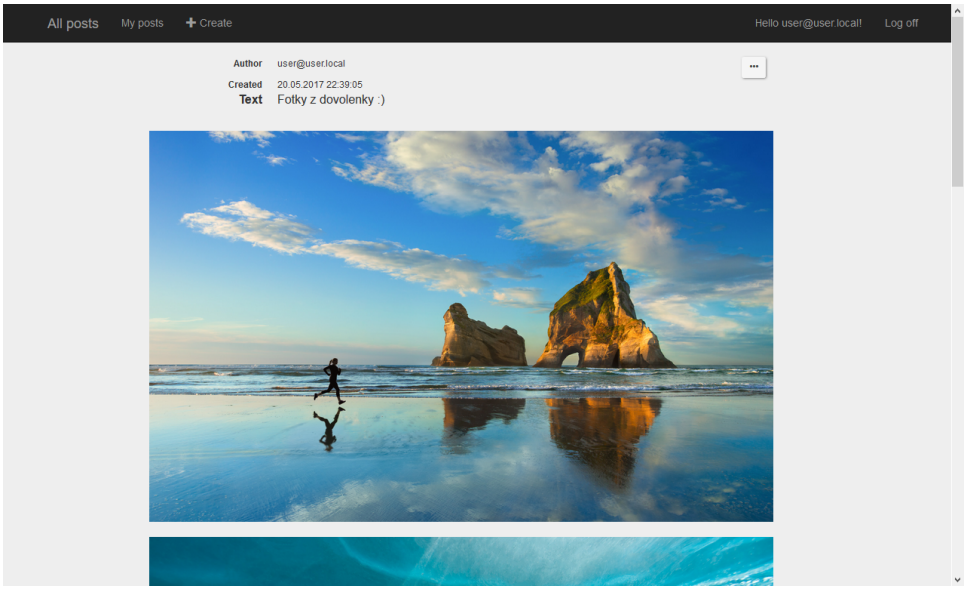


Obr. 5.1: Hlavná obrazovka s príspevkami

Táto obrazovka sa zobrazuje užívateľom aj po vytvorení nového príspevku. Grafika je podobná niektorým užívateľsky známym stránkam, ako je Facebook či Twitter. Dizajn aplikácie je preto rýchly na orientáciu aj pre bežného užívateľa. V ľavej časti vrchného panela sa nachádza odkaz na zobrazenie hlavnej obrazovky so všetkými príspevkami, ďalej má užívateľ možnosť zobraziť si iba svoje príspevky. Panel obsahuje aj tlačidlo na pridávanie príspevkov, po prekliknutí sa zobrazí stránka, ktorá je zobrazená na obrázku 5.3. V pravej časti panela sa nachádza zobrazený email prihláseného užívateľa, po prekliknutí má užívateľ možnosť zmeniť svoje heslo. Vedľa užívateľského emailu sa nachádza odkaz s možnosťou odhlásiť aktuálne prihláseného užívateľa. Ako som uviedol v časti 3, užívatelia môžu príspevok ohodnotiť

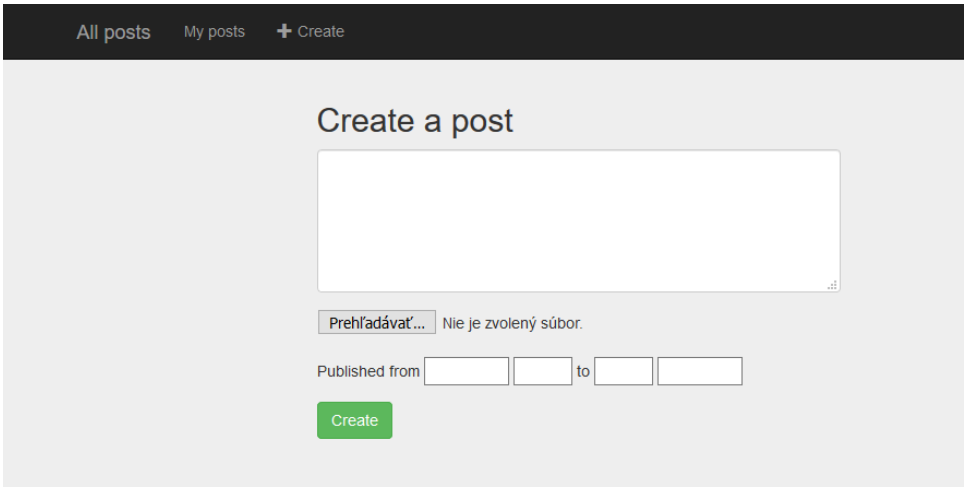
tlačidlom „Páči sa mi to“, čím vyjadria, že sa im tento príspevok páči. Ako je na obrázku 5.1 vidieť, grafika stránky je veľmi jednoduchá a intuitívna. Výhodou aplikácie je, že na bokoch panelu, kam sa vkladajú príspevky, nie sú rušivé príspevky, ako majú známe webové aplikácie a kam sa bežne umiestňujú reklamy.

Užívateľ môže priamo na tejto obrazovke prehliadať obsah, prípadne si môže zobraziť detail jednotlivých príspevkov, ktoré ponúkajú výhodu vo väčšom zobrazení multimediálneho obsahu. Príspevok obsahujúci fotografie je zobrazený na obrázku 5.2.



Obr. 5.2: Zobrazenie príspevku s obrázkami

Aplikácia umožňuje užívateľom vytvárať nové multimediálne príspevky, kde je okrem samotného obsahu možné nastaviť, v akom časovom rozsahu sa má daný príspevok zobrazovať. Vďaka tejto možnosti môže užívateľ nastaviť, aby sa príspevky prestali zobrazovať v čase, keď už nie sú relevantné, alebo naopak zobrazovali sa, až keď je to potrebné. Zadávanie nového príspevku je znázornené na obrázku 5.3.



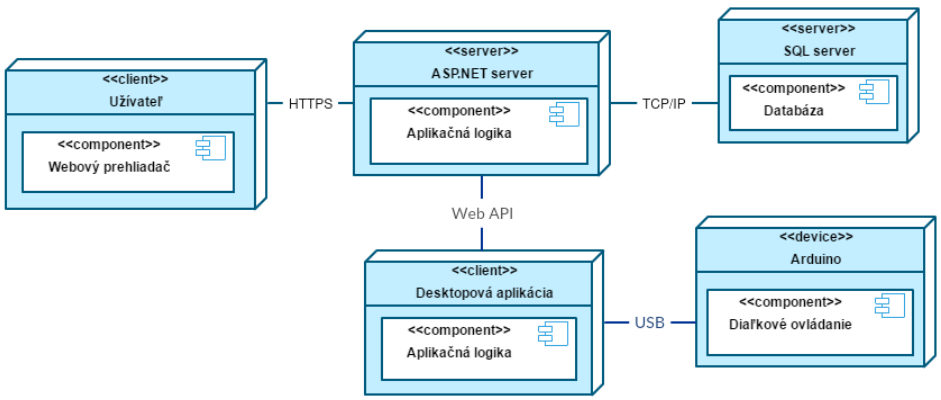
Obr. 5.3: Vytváranie príspevku

6 Vyhodnotenie a nasadenie

Výsledné aplikácie spĺňajú požiadavky definované v zadaní práce a získané od zamestnancov spoločnosti Siemens Corporate Technology Brno. Počas vývoja aplikácií boli priebežné výsledky diskutované s konzultantom práce so spoločnosti Siemens. Konzultant kladne hodnotí jednoduchosť a použiteľnosť vytvorených aplikácií, použité technológie, spoluprácu hardvéru so softvérom a komunikáciu medzi aplikáciami. Pozítivne je hodnotené aj ukladanie súborov na strane klienta, aby nedochádzalo k zbytočnému zaťažovaniu siete pri opakovanom zobrazovaní príspevkov.

Pri výsledom hodnotení riešenia sa našli drobné problémy s ovládaním zvuku v klientskej aplikácii, prostredie aplikácie umožnovalo nastaviť záporné hodnoty hlasitosti. Ďalšia pripomienka bola k chýbajúcej možnosti zobrazenia príspevkov prihláseného užívateľa. Bola opravená chyba s ovládaním hlasitosti a užívateľ si môže zobraziť všetky príspevky, alebo len svoje. Obe pripomienky sú zapracované do výsledného riešenia.

Výsledné aplikácie sú pripravené bežať na rôznych uzloch. Webová aplikácia komunikuje s desktopovou aplikáciou zobrazujúcou obsah prostredníctvom zabezpečeného REST API. Na diagrame nasadenia môžme vidieť rozloženie programových komponentov systému na harvérové uzly 6.1.



Obr. 6.1: Diagram nasadenia

7 Záver

Cieľom tejto práce bolo navrhnúť a implementovať riešenie prezentovania multimediálneho obsahu na televíznej obrazovke umiestnenej v zdieľanom priestore pracovného prostredia. Textová časť práca popisuje analýzu požiadaviek, použité technológie, implementačné detaily a vyhodnotenie aplikácií na zadávanie a prezentovanie obsahu na televíznej obrazovke. Výsledné aplikácie boli úspešne implementované na základe požiadaviek definovaných v analytickej časti tejto práce. Základné požiadavky boli definované zadávateľom práce a detailnejšie informácie boli zistené od zamestnancov firmy, užívateľov vyvíjanej aplikácie. V práci sú popísané využité technológie, knižnice a návrhové vzory. Výsledkom implementačnej časti sú aplikácie umožňujúce zadávanie príspevkov a zobrazovanie príspevkov na televíznej obrazovke. Zadávané a zobrazované príspevky obsahujú multimediálny obsah ako text, obrázky a videá. Webová aplikácia beží na serveri a aplikácia zobrazujúca obsah na televíznej obrazovke beží na počítači pripojenom k televízii. Aplikácie boli navrhované s úmyslom jednoduchého použitia. Výsledné aplikácie sú pripravené na nasadenie v reálnom prostredí. Aplikácie boli úspešne odprezentované kozultantovi práce vo vývojárskom prostredí a jeho pripomienky boli zapracované do aplikácií. Aplikácie využívajú technológie požadované zadávateľom práce, aby bolo možné budúce rozšírenie vytvorených aplikácií.

Literatúra

[1] Jimmy Schaeffler. „Digital Signage: Software, Networks, Advertising, and Displays: A Primer for Understanding the Business“. In: Focal Press, 2008. Kap. 1.

[2] *Xibo*. URL: http://xibo.org.uk (cit. 05. 02. 2016).

[3] *Screenly*. URL: https://www.screenly.io (cit. 05. 02. 2016).

[4] *Cignage media solutions*. URL: http://81.27.40.158/images/gallery/3.jpg (cit. 06. 02. 2016).

[5] *Smart TVs Are Better Than Ever, But Face One Glaring Problem*. URL: https://arc.applause.com/2015/01/06/smart-tvs-better-ever-face-one-glaring-problem/ (cit. 27. 01. 2016).

[6] *Overview of ASP.NET Core MVC*. URL: https://docs.microsoft.com/en-us/aspnet/core/mvc/overview (cit. 22. 07. 2016).

[7] *Introduction to Test Driven Development (TDD)*. URL: http://agiledata.org/essays/tdd.html (cit. 18. 01. 2017).

[8] *ASP.NET MVC Overview*. URL: https://msdn.microsoft.com/en-us/library/dd381412(v=vs.108).aspx (cit. 22. 07. 2016).

[9] Roy Thomas Fielding. „Architectural Styles and the Design of Network-based Software Architectures“. PhD Dissertation. University of California, Irvine, 2000.

[10] *Object Role Modeling: An Overview*. URL: https://msdn.microsoft.com/en-us/library/aa290383(v=vs.71).aspx (cit. 09. 04. 2017).

[11] *LINQ: .NET Language-Integrated Query*. URL: https://msdn.microsoft.com/en-us/library/bb308959.aspx (cit. 09. 04. 2017).

[12] *Data Binding Overview*. URL: https://msdn.microsoft.com/en-us/library/ms752347(v=vs.110).aspx (cit. 10. 04. 2017).

[13] *The MVVM Pattern*. URL: https://msdn.microsoft.com/en-us/library/hh848246.aspx (cit. 10. 04. 2017).

[14] *Arduino Nano*. URL: https://www.arduino.cc/en/Main/arduinoBoardNano (cit. 07. 10. 2016).

[15] *Make Your Own Arduino Nano In The Simplest Way (DIY – Arduino Nano)*. URL: http://www.electronics-lab.com/make-your-own-arduino-nano-diy-arduino-nano/ (cit. 09. 04. 2017).

[16] Robert C. Martin. *Čistý kód*. Computer Press, a.s., 2009. Kap. 11.