VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

SYSTÉM PRO SPRÁVU KAMEROVÝCH ZÁZNAMŮ

CAMERA RECORDING MANAGEMENT SYSTEM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **ONDŘEJ ŠULC**
AUTHOR

VEDOUCÍ PRÁCE — **Mgr. Ing. PAVEL OČENÁŠEK, Ph.D.**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2020/2021

Zadání bakalářské práce

Student: **Šulc Ondřej**
Program: Informační technologie
Název: **Systém pro správu kamerových záznamů**
Camera Recording Management System
Kategorie: Web

Zadání:

1. Seznamte se s technologiemi nahráváním a streamováním záznamů prostřednictvím webových kamer. Nastudujte webové technologie pro tvorbu bezpečných webových informačních systémů.
2. Analyzujte a specifikujte požadavky na bezpečný systém pro ukládání a správu záznamů z kamerových systémů.
3. Navrhněte aplikaci s webovým rozhraním, která bude podporovat získávání, správu a sdílení záznamů z kamerových systémů.
4. Navrženou aplikaci implementujte dle pokynů vedoucího práce.
5. Aplikaci otestujte na reálných datech.
6. Diskutujte získané výsledky a navrhněte další rozšíření aplikace.

Literatura:

- Bishop, M.: Computer security: Art & Science. Addison-Wesley, Boston, 2003, ISBN 0-201-44099-7.
- Menezes, A. J., Oorschot, P.C. van, Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press, 1996, ISBN 0-8493-8523-7 <http://www.cacr.math.uwaterloo.ca/hac/>.
- Dále dle doporučení vedoucího práce.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3 zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Očenášek Pavel, Mgr. Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 22. října 2020

Abstrakt

Cílem této bakalářské práce je tvorba síťového bezpečnostního systému se zaměřením na kompatibilitu s co nejširší skupinou IP kamer od různých výrobců a podporou vzdáleného přístupu k záznamům i z míst mimo lokální síť. Tato problematika je řešena pomocí dvou úzce spolupracujících aplikací.

První aplikace běží na zařízení v lokální síti (v našem případě miniPC Raspberry Pi). Aplikace vytváří záznamy přijímáním a ukládáním RTSP přenosu z jednotlivých kamer. Následně tyto záznamy upravuje do podoby použitelné k zobrazení na webových stránkách a odesílá druhé aplikaci, ze které se k nim dostane uživatel. Aplikace je v textu označována jako *agent* nebo *agentská aplikace*.

Druhá aplikace disponuje webovým uživatelským rozhraním a slouží koncovému uživateli jako prostředek pro správu agentů, IP kamer a pořízených záznamů. Z principu aplikace je důležité, aby byla přístupná na veřejně dostupné doméně. Aplikace je v textu označována jako *web* nebo *webová aplikace*.

Využitím této architektury není narušen žádný ze standardů síťové komunikace, mezi které patří především inicializace komunikace ze sítě za službou NAT.

Abstract

The aim of this bachelor thesis is to create a network security system focusing on compatibility with the widest possible group of IP cameras from various manufacturers and support for remote access to records even from places outside the local network. This problem is solved using two closely cooperating applications.

The first application runs on a device in the local network (in our case miniPC Raspberry Pi). The application creates the records by receiving and storing RTSP transmissions from individual cameras. It then modifies these records into a form usable for display on websites and sends the second application from which the user can access them. This application is referred to in the text as *agent* or *agent application*.

The second application has a web user interface and serves the end user as a tool to manage agents, IP cameras and captured records. By principle of the application, it is important for it to be accessible on a publicly available domain. This application is referred to in the text as *web* or *web application*.

The use of this architecture does not violate any of the network communication standards, which include, in particular, the initialization of communication from the network behind the NAT service.

Klíčová slova

IP kamera, bezpečnostní systém, vzdálený přístup, .NET Core, Raspberry Pi, NAS, FTP, RTSP, RTP, HTTP, HTTPS

Keywords

IP camera, security system, remote access, .NET Core, Raspberry Pi, NAS, FTP, RTSP, RTP, HTTP, HTTPS

Citace

ŠULC, Ondřej. *Systém pro správu kamerových záznamů*. Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Mgr. Ing. Pavel Očenášek, Ph.D.

Systém pro správu kamerových záznamů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Mgr. Ing. Pavla Očenáška, Ph.D. Další doplňující informace mi poskytl pan Bc. Tomáš Herceg. V seznamu literatury jsem uvedl všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Ondřej Šulc
7. května 2021

Poděkování

Rád bych poděkoval panu Mgr. Ing. Pavlu Očenáškovi, Ph.D., za odborné vedení této bakalářské práce a za rady, které mi poskytl při její realizaci. Dále bych chtěl poděkovat panu Bc. Tomáši Hercegovi za pomoc při práci s platformami .NET Core a Microsoft Azure.

Obsah

Seznam obrázků

Kapitola 1

Úvod

V dnešní době patří kamerové systémy mezi nejčastěji využívanou volbu v oblasti fyzického zabezpečení objektů. Na trhu je možné vybírat mezi dvěma druhy kamer – analogovými a IP kamerami.

Analogová kamera (případně více kamer) tvoří spolu s videorekordérem analogový systém. V rámci tohoto systému musí být kamery fyzicky propojeny s příslušným videorekordérem pomocí datového kabelu.

Mezi nevýhody těchto systémů tedy patří především vysoké nároky na infrastrukturu objektu a neefektivní vynakládání finančních prostředků v případě systému obsahujícího jen jednu kameru. Takový systém by vyžadoval pořízení zbytečně drahého videorekordéru, bez kterého by nebylo možné záznamy zpracovávat. Posledním, avšak poněkud zásadním, nedostatkem těchto kamer oproti jejich IP protějškům je omezenost jakýchkoli funkcí. Analogové kamery umí nahrávat, nicméně pro jakékoli další funkce postrádají potřebnou výbavu.

Tyto složité analogové systémy volí především majitelé rozsáhlejších objektů, u kterých je potřeba pro pokrytí celé hlídané oblasti relativně velké množství kamer a řešení pomocí IP kamer by bylo až příliš finančně náročné.

IP kamerové systémy jsou naopak mnohem jednodušší na instalaci a jejich požadavky na infrastrukturu se ani zdaleka nepřibližují požadavkům analogových systémů. IP kamery lze totiž připojit buď pomocí ethernetového rozhraní, které se v dnešní době nachází téměř v každé místnosti, anebo je lze v krajním případě připojit bezdrátově pomocí WiFi.

Dále jsou IP kamery mnohem chytřejší. Často disponují celou řadou funkcí, jako je například rozpoznávání pohybu nebo přímo obličeje, detekce neočekávaného zvuku, ovládání PTZ a mnoho dalších.

V tom se ovšem odráží značná nevýhoda, kterou je jejich cena. Ta totiž několikanásobně převyšuje cenu analogové kamery. V případě rozsáhlých kamerových systému je tedy nutné zvážit jejich celkovou finanční náročnost.

Pro svoji intuitivnost jsou IP kamery oblíbené i mezi laickou veřejností – s nastavením by neměl mít problém žádný technicky průměrně zdatný uživatel. Co však může u takových systémů způsobit problém, je řešení úložiště pro pořizované záznamy – a to i přes to, že kamery v této oblasti často umožňují hned několik variant:

- SD karta

  Téměř všechny kamery mají možnost jednoduchého ukládání záznamů na SD kartu. Toto řešení je sice pro uživatele nejjednodušší, ovšem znamená to značné komplikace

při práci se záznamy. Nejen, že data lze jen těžko zálohovat, ale také mohou být snadno odcizena i s kamerou v případě nějakého vloupání.

- Vzdálené úložiště poskytovatele

  Oproti SD kartám, které představují řešení skrze lokální úložiště, je možné využít vzdáleného úložiště nabízené výrobcem dané kamery. Data jsou chráněna před odcizením i ztrátou a navíc k nim lze svobodně přistupovat odkudkoli. Kde však může uživatel narazit, je využívání IP kamer od různých výrobců. Pokud se uživatel rozhodne pro tuto variantu ukládání, poněkud si znesnadňuje, ba dokonce v některých případech znemožňuje, koupi produktů od konkurenčních výrobců.

- Místní úložiště v lokální síti

  Poslední variantou, ke které se však uchylují spíše nadšenci, je zřízení vlastního síťového úložiště. To bývá nejčastěji realizováno pomocí FTP serveru, pro které má mnoho kamer i přímou podporu. Při odcizení kamery jsou tedy data stále na záložním disku. Navíc tato forma ukládání umožňuje využívat kamery od různých výrobců. Co však tato varianta často postrádá, je přístup k datům z míst mimo danou lokální síť. Navíc neodborné nastavení může v krajních případech představovat bezpečnostní riziko – především ve veřejně přístupných sítích.

Právě problematikou bezpečného vzdáleného přístupu k pořízeným záznamům z IP kamer v lokální síti a současně možností využití IP kamer od různých výrobců se bude tato bakalářská práce zabývat. [9] [8]

Kapitola 2

Existující systémy

2.1 FreeIP

Free IP je služba, která umožňuje svým uživatelům bezplatné vytvoření uživatelských účtů, ke kterým si následně mohou přidat své IP kamery pomocí sériového čísla. U takto přidávaných kamer slibují vzdálený přístup k živému přenosu i k záznamům. K Free IP lze přistupovat jako k webové službě nebo si lze stáhnout aplikaci z Google Play (Android) nebo App Store (iOS).

Po delším snažení bohužel nefungovala ani jedna z nabízených variant. Služba byla testována na 2 IP kamerách – TP-Link NC250 a BESDER 60R18MB.

Webová aplikace funguje na principu ActiveX (zastaralá služba kdysi využívaná internetovým prohlížečem Internet Explorer od společnosti Microsoft pro přenos médií) a vlastního pluginu do stejného prohlížeče. Využívání webové služby je tedy omezeno čistě na Internet Explorer. Při procesu přidávání kamery je vyžadováno sériové číslo kamery. Po zadání čísla webový prohlížeč však zamrzne a přidávání selže. U mobilní aplikace je bohužel zkušenost stejná. Zkušební počítač i mobil byli po celou dobu připojeni do stejné sítě jako kamera. [4]

2.2 Q-See QC IP HD System

QC IP HD System je IP kamerový systém sestávající od společnosti Q-See. Společnost Q-See v rámci svého bezpečnostního systému nabízí vlastní hardware v podobě IP kamer a nahrávače. IP kamery i nahrávač se napojí na lokální síť uživatele, kterou plně využívají k přenosu dat. Nahrávač disponuje uživatelským rozhraním, ke kterému uživatel získá přístup snadným připojením monitoru a myši.

Q-See v rámci systému také nabízí mobilní aplikaci, skrze kterou se lze na nahrávač i IP kamery připojit ovšem pouze v případě, že je uživatel na stejné lokální síti.

Vzdálený přístup z internetu je však v manuálu popsán velice nešťastně. Manuál automaticky předpokládá otevření portů 83 (NFS) a 6036 na směrovačích. Tento přístup zaprvé diskriminuje technicky neznalé uživatele a zadruhé nepředpokládá měnitelnou veřejnou IP adresu, kterou konkrétnímu routeru dodává příslušný poskytovatel internetu. [19]

2.3 TP-LINK cloud

TP-LINK cloud je webová aplikace společnosti TP-Link zprostředkovávající živý přenos z IP kamer vlastní výroby. V aplikaci kromě nastavení názvu dané kamery nejsou umožněny prakticky žádné další možnosti.

Aplikace sice živý obraz bez problému zprostředkovává odkudkoli, ale neumožňuje kusé ani kontinuální ukládání záznamů.

Podle informací v aplikaci byly všechny původní funkce přesunuty do mobilní aplikace TP-LINK tpCamera.

2.4 TP-LINK tpCamera

TP-LINK tpCamera je mobilní aplikace společnosti TP-Link pro kamery NC210 – NC260 a NC450. Aplikace zprostředkovává snadné připojení ke kameře. Automaticky vyhledává IP kamery v lokální síti (Toto vyhledávání podle zaznamenané komunikace probíhá pomocí broadcastového dotazu na port 1068). Aplikace po připojení nabízí živý přenos a pořizování nahrávek a snímků, ale navíc umožňuje uložení i dalších doplňkových informací usnadňujících používání. Živý přenos je stejně jako u TP-LINK cloudu možný odkudkoli - komunikace totiž probíhá právě přes něj.

Stejně jako u TP-LINK cloud zde není umožněno nastavení průběžného nahrávání a ukládání záznamů.

2.5 mydlink

Mydlink je mobilní aplikace sloužící k ovládání všech IP zařízení dodávaných společností D-Link. Aplikace nabízí intuitivní nastavení IP kamer své vlastní výroby. Automaticky vyhledává IP kamery v lokální síti a přesměrovává do jejich nastavení. Umožňuje pořizování snímku a nahrávání záznamů a jejich ukládání přímo do mobilu. Aplikace umožňuje nastavení nahrávání a přístup na službu D-Link cloud. S využitím služby D-Link cloud je možné přistupovat k záznamům a živému přenosu odkudkoli.

Kapitola 3

Technologie pro komunikaci a nahrávání

3.1 RTP

Real-time Transport Protocol je protokol určený pro přenos dat po síti v reálném čase. V dnešní době je využíván především pro živý přenos audio a video dat.

Tento protokol je možno využívat v kombinaci s oběma transportními protokoly (TCP i UDP) podle potřeby aplikace, která jej využívá. V praxi je však častěji využíván spíše protokol UDP. Při využití RTP se předpokládá, že je potřeba dbát především na rychlost komunikace. Pokud by aplikace využívala TCP a stalo se, že se některý paket ztratí, TCP se jej pokusí odeslat znovu. Z pohledu aplikace však takový znovu odeslaný paket nemusí být vůbec důležitý a využití TCP by tak výrazně a zbytečně zdržovalo přenos.

RTP lze využívat pro přenos peer-to-peer, ale také pomocí vícesměrového vysílání (multicastu) do více destinací. Je totiž primárně určen pro víceuživatelské konference k přenosu videa a audia (VoIP) mezi všemi uživateli.

Mezi služby, které tento protokol nabízí patří sekvenční číslování a časová razítka, avšak u přenášených dat není zajištěno včasné doručení ani doručení ve správném pořadí. Aplikace využívající RTP musí být na tyto situace připravena. Pokud jsou tyto vlastnosti v aplikaci i přesto potřeba, musí je pochopitelně zajistit nižší vrstva.

Samotný protokol nezajišťuje bezpečnou šifrovanou komunikaci, avšak je možné jí docílit využitím SRTP.

S RTP úzce souvisí protokol RTCP. K tomuto protokolu nám však pro účely této práce stačí znalost, že slouží k informování o kvalitě služby a ke sdělování informací o účastnících probíhajícího sezení. [22] [5]

3.2 RTSP

„Put simply, RTSP acts as a „network remote control“ for multimedia servers.“[1]

Real-time Streaming Protocol je protokol aplikační vrstvy, který připravuje přenos a udržuje kontrolu nad daty, která mají být přenášena v reálném čase. Protokol funguje mezi RTSP klientem a RTSP serverem, přičemž protokolu nevadí proxy servery na cestě.

Nejprve klient potřebuje znát informace o datech, která chce získat. To může učinit tak, že se na dané informace zeptá přímo samotného serveru anebo může použít libovolný externí zdroj informací. Následně jsou pomocí přenosových protokolů připraveny datové toky podle poskytnutých informací.

Mezi hlavní příkazy pro zahájení komunikace patří:

- OPTIONS – server vypíše možné požadavky, které je schopen obsloužit
- DESCRIBE – server poskytne popis dat
- SETUP – sdělí serveru informaci, jak má přenášená data posílat a který port klienta použít

Klienti pak mohou posílat požadavky pro práci s daty:

- PLAY – zahájení přenosu
- PAUSE – pozastavení přenosu
- TEARDOWN – úplné zastavení přenosu

Důležité ale je, že RTSP samotná data, která jsou předmětem komunikace neposílá. K tomu využívají výše popsaný protokol RTP.

Samotný RTSP protokol není šifrovaný. Pro šifrování přenosu je možné použít RTSP over SSL (RTSPS). [21]

Protokol RTSP je v IP kamerách běžně využívaný a port je otevřený. Co však bývá občasným problémem je nastavení RTSP. Přihlašovací údaje nejsou vždy stejné jako jsou přihlašovací údaje např. do uživatelského prostředí a je možné, že si výrobce tento protokol v některých případech nechává čistě pro tvorbu vlastních aplikací, které se na něj mohou připojit.

3.3 FTP

File Transfer Protocol je protokol vytvořený za účelem přenosu dat po síti. Je možné jej využívat pomocí terminálu, nicméně byl navržen především pro využití aplikacemi. Základní forma FTP využívá pro komunikaci standardní porty 20 a 21. Port 21 je využíván přenos příkazů a port 20 pro přenos samotných dat. V dnešní době patří FTP mezi hojně využívané protokoly pro přenos dat.

Jeho využití však skýtá velkou nevýhodu v podobě zabezpečení. FTP nebyl vytvořen, aby nabízel bezpečný přenos a je považován za protokol, jehož využívání může v konkrétních případech znamenat nepříjemné bezpečnostní riziko. Posílaná data (přihlašovací údaje i přenášené soubory) nejsou nijak šifrovaná a z toho důvodu se mohou stát snadnou obětí základních útoků jako jsou sniffing a spoofing.

[1] RFC 7826 Schulzrinne, et al.

Pro zašifrování komunikace existuje zabezpečené FTP over TLS (FTPS). Alternativou FTPS může být ještě SFTP, které přenos dat provádí přes SSH, avšak tento protokol nemá s původním FTP nic společného. [17] [7]

IP kamery protokol FTP ve většině případů podporují, avšak zabezpečená verze FTP se objevuje až u dražších produktů. Při využití FTP je tedy nezbytné správně nastavit konfiguraci FTP serveru, aby nemohlo dojít k nežádoucímu vniknutí na úložný server.

3.4 HTTP a HTTPS

Hypertext Transfer Protocol je bezstavový protokol aplikační vrstvy typu dotaz/odpověď. Jak jeho název napovídá, je určen pro přenos textů po síti. Mezi taková textová data patří především webové stránky, nicméně díky standardu MIME je možné přes HTTP posílat i binární data – především video a případně audio.

Dnes je nejvíce využívána nejnovější verze protokolu HTTP 2.0, která významným způsobem zefektivňuje využívání síťových zdrojů – zejména při víceuživatelském přístupu.

Komunikace přes HTTP není šifrovaná. Pro šifrovanou komunikaci je potřeba využít HTTP over TLS (HTTPS). HTTP 2.0 musí podle rfc7540 využívat TLS 1.2 anebo vyšší. [1]

IP kamery často disponují webovým uživatelským rozhraním, na které je možné se připojit pomocí webového prohlížeče. Pomocí tohoto webového prostředí je možné nastavit většinu funkcionalit dané IP kamery a zároveň sledovat živý přenos. Šifrované HTTPS však využívá pouze minimum. Uživatelské jméno bývá posíláno v úplně odkryté formě a heslo bývá bývá skryto jen pomocí Base64.

3.5 NAS

Network Attached Storage je síťové zařízení sloužící jako datové úložiště. To znamená, že k ukládaným souborům lze přistupovat z jakéhokoli zařízení, které má připojení k dané síti a není tedy potřeba mít soubory uloženy lokálně. Přístup k ukládaným datům může být navíc zabezpečen heslem.

Mezi běžné protokoly podporované NAS službou bývají:

- SMB/CIFS

  Server Message Block taktéž známý jako Common Internet File System je protokol sloužící ke sdílenému přístupu k síťovým zdrojům.

- NFS

  Network File System umožňuje ke vzdálenému úložišti přistupovat jako k lokálnímu

- FTP

  File Transfer protocol sloužící k přenosu dat po síti.

NAS přitom nemusí být pouze speciální zařízení, ale může za něj být považován prakticky jakýkoli počítač, který bude služby spojené s NAS poskytovat. [18]

NAS je výborná volba pro ukládání záznamů z IP kamer. Data jsou oddělena od IP kamery a jsou tak chráněna v případě odcizení či poškození kamery.

3.6 NAT

Network Address Translation je způsob změny IP adres v hlavičce paketů během síťového provozu, který prochází přes router mezi vnitřní a vnější sítí. Tento standard byl především zaveden z důvodu omezeného počtu adres v IPv4.

Pokud uživatel ve vnitřní síti chce posílat data do vnější sítě, router před vypuštěním paketu na výstupní port provede tzv. NAT překlad. Ten spočívá v nahrazení IP adresy původního odesílatele IP adresou přiřazenou na výstupním rozhraní a přidělením portu, ze kterého bude daný dotaz přeposlán. Na tomto portu následně router také očekává odpověď, kterou přepošle zpět klientovi.

Při komunikaci přes NAT je především důležité zahajovat komunikaci z vnitřní sítě (tedy té, která je „za“ routerem). Vnitřní síť je z veřejné sítě prakticky neviditelná a plně zastoupena routerem.

Existují způsoby, jak umožnit zahájení komunikace i z vnější sítě – např. Port Forwarding. Tyto postupy však zasahují do běžné funkcionality NAT a při nesprávném nastavení se mohou snadno stát potencionálními bezpečnostními dírami dané sítě.

3.7 ONVIF

ONVIF je otevřené průmyslové fórum poskytující standardizaci síťového rozhraní IP zařízení pro efektivní interoperabilitu síťových bezpečnostních produktů napříč různými výrobci.

Specifikace ONVIF jsou podle typu rozděleny do několika odvětví – tzv. profilů. V těchto jednotlivých profilech je pak dále rozlišováno, zda se jedná o „vyhovující zařízení“ (angl. conformant device) nebo o „vyhovujícího klienta“ (angl. conformant client).[10]

- Profil A

  Profil A je určen pro produkty v elektronických přístupových systémech sloužících pro identifikaci osob.

  Rozhraní zařízení vyhovující profilu A musí umět odesílat informace o vlastním stavu zařízení, upozornění na nečekané události a konfigurovat interní entity jako jsou přístupová práva, přístupové údaje a časové rozvrhy.

  Klient vyhovující profilu A musí umožňovat konfiguraci přístupových práv, přístupových údajů a časových rozvrhů vyhovujících zařízení a zároveň přijímat a odesílat standardizované události. [11]

- Profil C

  Profil C je určen především pro bezpečnostní dveře a obsluhu nečekaných událostí.

  Rozhraní zařízení i klienta profilu C musí umět zpracovávat informace o stavu hlídané plochy a bezpečnostních dveří a musí poskytovat správu událostí a alarmů. [12]

- Profil Q

  Profil Q specifikuje požadavky pro rychlou instalaci zařízení. Je určen především pro síťové videosystémy a snaží se poskytnout snadné vyhledání zařízení v síti a základní konfiguraci zařízení, které tento profil podporují.

  Zařízení vyhovující profilu Q je takové, které může být v dané síti nalezeno, nakonfigurováno a ovládáno klientem podporujícím profil Q.

Klient vyhovující profilu Q je takový, který dovede nalézt, nakonfigurovat a ovládat zařízení podporující profil Q přes lokální síť.

Profil také popisuje standardy pro využití TLS (Transport Layer Security), aby bylo takovým zařízením umožněno vzájemně komunikovat v zabezpečené formě. [14]

- Profil S

  Profil S z roku 2011 je prvním ONVIF profilem zabývajícím se vzájemnou komunikací síťových monitorovacích zařízení. V první řadě se standardizuje přenos samotného videozáznamu tak, aby IP kamera jejíž síťové rozhraní vyhovuje profilu S dovedlo bezproblémově obdržet příkazy z vyhovujícího klienta a následně odpovědět odesláním pořízeného záznamu.

  Profil S kromě této povinné implementace, kterou musí disponovat každé zařízení označené jako „vyhovující“, standardizuje i některé nepovinné vlastnosti kamer jako jsou například:

  – PTZ (Pan-Tilt-Zoom) je schopnost kamery si samostatně nastavit své zorné pole. K tomuto většinou slouží 3 zabudované elektrické motory, kde každý z nich ovládá jednu ze 3 os:
    * Osa „pan“, zajišťuje panoramatické polohování kamery.
    * Osa „tilt“ zajišťuje polohování kamery ve vertikální ose.
    * Osa „zoom“ zajišťuje optické přiblížení snímací čočky.
  – přenos audia společně s videem.
  – multicasting. Tato funkce může v některých případech významně usnadnit práci při nastavení přijímání záznamu na straně klienta, pokud to daná bezpečnostní politika dovoluje.

  [15]

- Profil G

  Profil G z roku 2014 popisuje základní standard pro řešení úložiště IP kamer v síťových videosystémech. Popisuje dvě varianty, jakými může vyhovující IP kamera ukládat své záznamy.

  1. Lokální úložiště

     V tomto případě se předpokládá, že IP kamera disponuje vlastním slotem například pro SD kartu. Specifikace profilu G se v takovém případě zaměřuje na standardizaci síťového rozhraní, přes které je možné pomocí vyhovujícího klienta uložené záznamy vyhledat, přehrát nebo jinak konfigurovat.
  2. Sítový videorekordér

     U této varianty se naopak předpokládá, že IP kamera nemá žádné vlastní úložiště a trvalost záznamů závisí na jiných zařízeních.

     Specifikace profilu G pak udává standard síťového rozhraní vyhovujícího videorekordéru, pomocí kterého pak vyhovující IP kamera může pořízené záznamy ukládat.

  Specifikace také obsahuje standard pro ukládání a přeposílání audia a metadat. [13]

- Profile T

  Profil T z roku 2018 především rozšiřuje původní profil S z roku 2011 tak, aby lépe vyhovoval novodobým požadavkům na bezpečný přenos a poskytoval široké rozhraní pro ovládání všech parametrů vyhovující IP kamery. Profil S však s příchodem profilu T není zcela nahrazen a stále má své oblasti využití. Stejně tak může existovat zařízení, které podporuje pouze profil T.

  Profil specifikuje nastavení/využití následujících vlastností:

  - Nastavení snímaného obrazu (jas, kontrast, natočení obrazu, zaostření)

    Vyhovující IP kamera musí disponovat síťovým rozhraním, které umožní nastavení těchto parametrů z vyhovujícího klienta za předpokladu, že ona samotná má prostředky pro modifikaci snímaného obrazu.
  - Komprese videa (kodeky H.264 a H.265)

    Vyhovující IP kamera musí umět své záznamy komprimovat alespoň jedním z uvedených kodeků. Vyhovující klient však musí umět přijímat záznamy obou variant.
  - Využití HTTPS pro bezpečný přenos dat

    Vyhovující klient i IP kamera musí umět komunikovat zabezpečenou formou přes protokol HTTPS.
  - PTZ (Pan-Tilt-Zoom)

    Vyhovující IP kamera musí disponovat síťovým rozhraním, které umožní nastavení 3 os PTZ za předpokladu, že ona samotná k tomu má potřebné prostředky.
  - Upozornění na nečekané události (detekce pohybu)

    Vyhovující klient musí disponovat síťovým rozhraním, které umožní vyhovujícím IP kamerám zasílat upozornění na nečekané události.
  - Přenos metadat

    Vyhovující IP kamera musí společně se záznamem zasílat další metadata o stavu kamera (například stav PTZ)
  - Obousměrný přenos audia

    Vyhovující IP kamera musí disponovat síťovým rozhraním, které umožní přijímání audia. Tato funkce může být využita, pokud je možné ke kameře připojit reproduktor nebo kamera reproduktor přímo obsahuje.

  [16]

Kapitola 4

Technologie pro tvorbu webových informačních systémů

4.1 React

React je knihovna jazyka JavaScript spravovaná společností Facebook. Zaměřuje se pouze na blok View v modelu MVC. Stará se tedy pouze o zobrazení dat uživateli a nikoli o logiku aplikace. K napojení na aplikaci s logikou je možné využít volání rozhraní REST API, které React nativně podporuje.

Velká výhoda, která spočívá ve využití React je snadno implementovatelná dynamická změna i velkých bloků obsahu stránky. Pro projevení změny v obsahu stránky tedy není potřeba stránku obnovit a přitom programátor není zatížen složitou implementací JavaScriptu.

Projekt v React je skládán z tzv. komponent. Tyto komponenty se dělí na stavové a bezstavové a slouží k vygenerování malé, znovupoužitelné části HTML kódu. Stavové komponenty navíc mají schopnost udržovat různá data, která je možné následně na stránce libovolně využít. Nejčastěji bývají ve formátu JSON, nicméně je možné využít import XML. U bezstavových komponent je především běžné nastavení vstupních parametrů, které jsou těmito komponentami dosazeny do programátorem připraveného HTML kódu. [24] [6]

4.2 Vue.js

Vue.js je framework pro tvorbu uživatelských rozhraní podporovaný společností Google. Stejně jako React se zaměřuje pouze na blok View v modelu MVC a jeho využití vyžaduje alespoň základní znalosti HTML, CSS a JavaScriptu. Nezáleží, v jakém jazyce je programován backend neboť Vue.js bude vždy možné napojit prostřednictvím REST API rozhraní. Jako bonus Vue.js nabízí vlastní IDE zvané Vue DevTools, které vývoj uživatelského rozhraní může ještě více usnadnit.

Strukturálně je Vue.js velice podobný knihovně React. Je složen z komponent generujících HTML kód a přímém vkládání hodnot do HTML kódu. Samotná data však nejsou uložena přímo v komponentě jako je tomu u Reactu, ale jsou uloženy ve vedlejší struktuře, která se na zobrazovací strukturu napojí pomocí tzv. bindingu. [25]

4.3 ASP .NET Core

ASP .NET Core je vývojová platforma v jazyce C# pro tvorbu multiplatformních webových aplikací vyvinutá a spravovaná společností Microsoft. Na rozdíl od Reactu a Vue.js se však nezaměřuje pouze na grafickou stránku projektu, ale na celý projekt jako celek. Nabízí tak stejný programovací a jazyk a vývojové prostředí pro frontend i backend. ASP .NET Core dodržuje model MVC, díky čemuž není komplikováno testování a zároveň je možné využít rozhraní REST API i pro další klienty než jen webové rozhraní, které je nativně součástí projektu.

Projekt v ASP .NET Core je rozdělen do 3 částí podle modelu MVC. Programátor je tak přirozeně veden k dodržování standardů, díky čemuž je docíleno strukturovaného čitelného kódu. View je psáno pomocí běžného HTML, CSS a JavaScriptu, ale je možné do kódu přidat i prvky C# pro generování kódu před odesláním. View je navíc rozděleno na hlavní stránku a obsahy, aby se snadno docílilo konzistentního vzhledu. ASP tedy používá u všech dodávaných stránek stejnou strukturu v podobě hlavní stránky, do které se následně přidá stránka s obsahem.

Části Model a Controller jsou od View odděleny, aby byl zachován princip s voláním přes REST API a jsou psány už čistě v jazyce C#. [20]

4.4 Node.js

Node.js je open-source asynchronní událostmi řízené běhové prostředí jazyka JavaScript. Bylo vytvořeno za účelem využití JavaScriptu i na straně serveru a nikoli jen na straně klienta ve webovém prohlížeči. Je navržen pro tvorbu velkých a rozšiřitelných aplikací.

Projekt v Node.js je složen z jednoho hlavního modulu, který je spouštěn a několika přídavných modulů, které jsou z tohoto hlavního modulu volány. Tento runtime je určen především pro práci s protokolem HTTP, pro který má rozsáhlou podporu.

Při vývoji je možné využít rozsáhlou knihovnu Node package manager (npm), ze které si lze stáhnout předpřipravené moduly, které pak lze snadno zakomponovat do vyvíjené aplikace.

Jedním z nejznámějších takto využívaných modulů je modul ExpressJS, který je využíván pro rychlou a snadnou tvorbu REST API. [3]

4.5 .NET Core

.NET Core je bezplatná open-source vývojová platforma pro vývoj multiplatformních aplikací všeho druhu v jazyce C#. Tato platforma je základem pro již zmiňovanou platformu ASP pro vývoj webových aplikací. Tato platforma však nabízí základ i pro řadu dalších vývojových platforem:

- Console – vývoj konzolových aplikací pomocí .NET CLI
- Xamarin – mobilní aplikace pro OS Android
- Windows WPF – desktop aplikace pro OS Windows
- ML.NET – sada knihoven pro strojové učení
- .NET for Unity – vývoj her [23]

Kapitola 5

Návrh systému

5.1 Analýza a specifikace požadavků

Žádané a očekávané funkce od spolehlivého kamerového bezpečnostního systému:

- Dlouhodobé ukládání záznamů

  Systém musí být schopen dlouhodobě uchovávat pořízené záznamy a poskytovat k nim snadný přístup.

- Ukládání záznamů z většího množství kamer

  Systém musí umět ukládat a správně třídit záznamy z více kamer naráz. Všechny záznamy by měly být snadno přístupné z jednoho místa.

- Bezpečnost záznamů

  Systém musí patřičně chránit pořízené záznamy proti jejich ztrátě.

- Odolnost

  Systém musí být připraven vhodně reagovat na nečekané události – především při výpadku komunikace, který je při využití domácí sítě a internetu očekávaným zdrojem problémů.

- Upozornění

  Systém by měl umět uživatele vhodnou formou upozornit na nečekanou událost (výpadek zařízení, detekce pohybu).

- Ovládací rozhraní

  Systém musí disponovat přívětivým uživatelským rozhraním. Prostředí by mělo podporovat zabezpečení uživatelským heslem či systémem uživatelů.

- Vzdálený přístup

  Systém by měl podporovat zabezpečený vzdálený přístup a ovládání systému i z míst mimo lokální síť.

5.2 Struktura

Pro uspokojení všech požadavků uvedených v předchozí sekci bude systém rozdělen do dvou aplikací:

- Webová aplikace
- Agentská aplikace

Obě aplikace budou naprogramovány v jazyce C# s využitím vývojové platformy .NET Core.

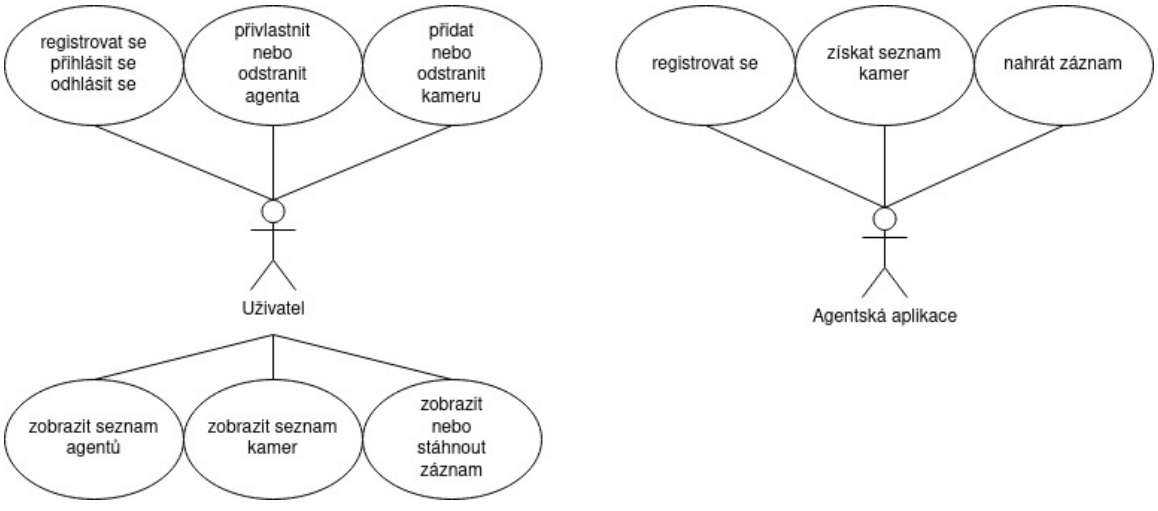
Webová aplikace

Webová aplikace bude sloužit jako uživatelské rozhraní umožňující zmiňovaný vzdálený přístup. Díky typu aplikace je navíc umožněn přístup z téměř jakéhokoli zařízení, které má dostatečné připojení k internetu. Uživatel tak není zatěžován instalací dodatečných aplikací, avšak je při tvorbě je důležité dodržet základní návrhové standardy, aby daná aplikace nespotřebovávala příliš velké množství dat.

Velká spotřeba dat zde však není myšlena jen z pohledu kvantity, kterou by mohli nepříjemně pocítit uživatelé mobilních dat, jejichž datový objem bývá zastropen, ale i šířky datového toku zejména pro uživatele domácího připojení, kde datový objem není sice limitován, ale mohlo by dojít k nepříjemnému zpomalení rychlosti internetu.

Diagram užití webové aplikace

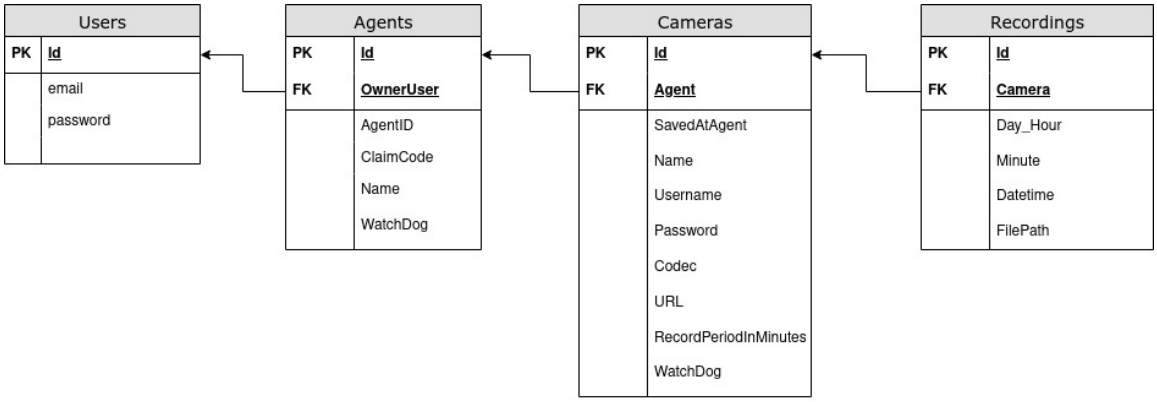
Uživatel bude mít z webové aplikace možnost nastavovat prvky svého bezpečnostního systému a zároveň prohlížet, stahovat a mazat pořízené záznamy.



Obrázek 5.1: Diagram užití webové aplikace

UML diagram databáze webové aplikace

Informační data budou na webové aplikaci ukládána do databáze. Samotné záznamy, jejichž velikost by mohla přesáhnout maximální možnou velikost pro uložení do databáze, budou ukládány přímo do souborového systému webové aplikace a v databázi bude uložena jen cesta k příslušnému souboru s obsahem.



Obrázek 5.2: UML diagram databáze webové aplikace

Agentská aplikace

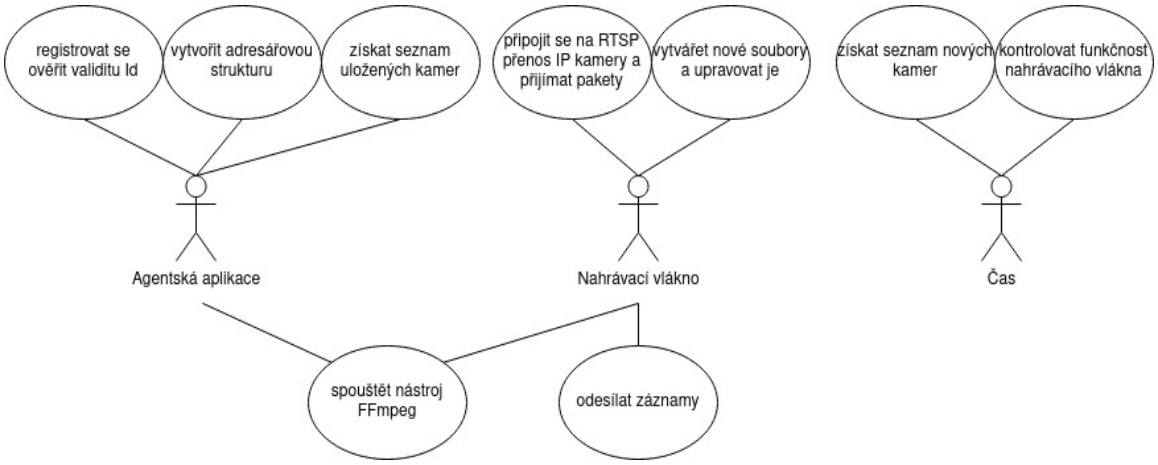
Princip agentské aplikace spočívá v jejím spuštěním na zařízení přímo v lokální síti. Díky tomu nebude od IP kamer oddělena NAT, který by znesnadňoval komunikaci. Uživatel přitom se samotnou aplikací nemusí vyjma spuštění nijak dále pracovat. Ta bude rovnou komunikovat s webovou aplikací, ze které bude přijímat veškeré příkazy k nastavení.

Aplikace musí ideálně běžet 24 hodin denně 7 dní v týdnu. Je tedy nutné aplikaci optimalizovat pro běh i na menších úsporných zařízeních. Příkladem takového zařízení může být malý počítač Raspberry Pi – bližší informace v sekci Potřebný hardware

Po spuštění se připojí na RTSP přenos z kamery a bude jej nahrávat do souboru na lokálním úložišti po specifikovanou dobu. Po skončení této doby příslušný soubor uzavře, překonvertuje na datový typ MP4 a odešle na webovou aplikaci.

Výhodou tohoto přístupu je, že koncovým bodem s RTSP přenosem disponuje většina IP kamer a díky tomu je zajištěna kompatibilita s většinou z nich. Nevýhodou pak může být, že uživatel musí aplikaci (jakožto jemu cizí aplikaci) sdělit přístupové údaje ke kameře. Některé IP kamery disponují systémem účtů a uživatel by tak mohl vytvořit zabezpečený účet – toto však záleží na konkrétní IP kameře a z pohledu aplikace je zcela neovlivnitelné.

Diagram užití agentské aplikace



Obrázek 5.3: Diagram užití agentské aplikace

5.3 Komunikace

REST API

Webová aplikace bude disponovat REST API rozhraním, na které bude uživatel a agentská aplikace posílat dotazy.

Agentská aplikace po spuštění pošle dotaz webové aplikaci, aby získala aktuální informace o kamerách, na které se má připojit. Aplikace si následně bude vytvářet zálohy těchto informací pro případ výpadku spojení.

Dále bude agentská aplikace posílat pořízené záznamy na speciální koncový bod aplikace. Data budou posílána ve formě formuláře, aby bylo možné společně se samotným obsahem záznamu zasílat další indentifikační informace.

Webová aplikace si bude díky pravidelné komunikaci uchovávat informaci, že je daný agent i kamery funkční.

JSON serializace a deserializace s využitím modelů

Komunikace mezi webovou a agentskou aplikací bude probíhat přes protokol HTTP. Předmětem tohoto protokolu je vždy zasílání řetězcových zpráv.

Při přenosu určitého objektu či datové struktury tedy musí odesílatel přenášenou informaci vhodným způsobem převést do řetězcové formy tak, aby ji příjemce uměl po přijetí opět převést zpět do strukturované podoby.

Tento převod strukturovaných dat na řetězec nazýváme jako serializace. V praxi se používají dva typy serializací:

- JSON (JavaScript Object Notation) – jednodušší syntaxe, která však není vhodná pro příliš komplikované struktury
- XML (Extensible Markup Language) – komplexnější syntaxe vhodná pro popis komplikovanějších strukturované

Pro komunikaci bude využita varianta využívající JSON. Samotná serializace pak probíhá s využitím modelů.

Modely

Modelem v .NET Core označujeme speciální třídu, kterou využíváme k popsání struktury dat přenášených ve formátu JSON. Vzhledem k využití .NET Core při programování obou aplikací je možné mít v obou aplikacích totožné sady modelů a předcházet tak chybám spočívajícím ve špatném překládání struktur.

V aplikaci jsou využívány 3 modely:

- CheckAgentModel: přenáší AgentID – slouží k identifikaci agenta při zaslání jakéhokoli požadavku na webovou aplikaci
- RegisterAgentModel – přenáší AgentID a ClaimCode – slouží k odeslání informací při registraci agenta.
- SearchCameraModel – přenáší informace o kameře

5.4 Potřebný hardware

IP kamera

IP kamera je síťové zařízení zpravidla disponující síťovým rozhraním s koncovými body pro konfiguraci a zpřístupnění živého přenosu.

Dále většina kamer obsahuje slot na SD kartu tedy přímo a schopností pořizovat a ukládat audio-vizuální záznamy. Tyto záznamy pak uživateli zprostředkovává prostřednictvím uživatelského rozhraní anebo lokálního či cloudového úložiště.

S rostoucí cenovou kategorií pak IP kamery pak mohou navíc mít i řadu dalších pokročilejších funkcí.

Mezi nejčastější patří:

- možnost bezdrátového připojení k místní síti
- detekce pohybu a hluku
- zasílání upozornění
- noční vidění
- odolnost proti živlům (venkovní IP kamery)

Raspberry Pi 4

Raspberry Pi spadá to elektronických zařízení obecně definovaných jako miniPC. Verze 4 je poslední a nejvýkonnější verzí z celé rodiny Raspberry Pi. Disponuje 4 jádrovým procesorem o výkonu 1.5GHz a existuje ve třech provedeních podle velikosti RAM (2GB, 4GB, 8GB). Dále Raspberry Pi obsahuje Gigabit Ethernet slot s konektorem RJ-45, WiFi a Bluetooth pro snadné zakomponování do sítě IoT. Pevné úložiště je realizováno pomocí SD karty, která nese operační systém a kořenový souborový systém. Dále je deska osazena 4 USB porty (2x USB2.0 a 2x USB3.0) pro snadné připojení dalších periferií.

Raspbian je oficiální a doporučovaný operační systém určený pro mini počítače Raspberry Pi. Je volně stažitelný z webových stránek Raspberry Pi Foundation ve 3 variantách.

1. Lite

   Verze obsahující základní systém ovladatelný z příkazového řádku.

2. Desktop

   Verze obsahující základní systém obohacený o uživatelské rozhraní.

3. Desktop + Recommended software

   Verze obsahující základní systém, uživatelské rozhraní a sadu doporučeného softwaru (především IDE pro python3 a sadu her)

Hostovací platforma

Webová aplikace musí být veřejně přístupná. Nejsnazší způsob jak takové dostupnosti docílit je využití veřejného hostování. Mezi nejznámější poskytovatele cloudových služeb patří například Microsoft Azure a Amazon AWS.

Pro účely této aplikace bude využita služba Azure – převážně díky větším zkušenostem z minulosti. Ve službě bude nejprve vytvořena nová skupina prostředků, ve které budou následně vytvořeny 3 prostředky:

- App Service – pro webovou aplikaci
- SQL Server – nutný pro přidání databáze
- SQL Databáze – databáze doplňující webovou aplikaci

Nasazení webové aplikace na prostředek App Service bude probíhat přes vývojové prostředí Visual Studio. Webová aplikace následně sama provede potřebné nastavení SQL databáze.

Kapitola 6

Webová aplikace

6.1 Struktura projektu

Aplikace byla vytvořena za pomoci vývojové platformy ASP.NET Core, která je blíže popsaná v předchozí kapitole. Pro samotnou strukturu projektu byla vybrána standardní šablona MVC (Model View Controller), která je nejvhodnější pro tvorbu rozsáhlejších webových aplikací.

Šablona systematicky rozděluje celý projekt do 7 modulů:

- soubor Program.cs

  Vstupní soubor s funkcí *Main*. Main obsahuje volání předdefinované metody pro spuštění hlavní části aplikace. V tomto souboru je tedy možné snadno provádět instrukce, které mají být provedeny ještě před spuštěním hlavní části.

- soubor Startup.cs

  V tomto souboru jsou nastavovány globální vlastnosti celé aplikace. Mezi nejdůležitější parametry, které se zde nastavují patří

  - struktura URL cesty
  - URL cesta pro zpracování chybových hlášek
  - HTTPS a přesměrování z HTTP
  - uživatelské účty včetně požadavků na strukturu hesla, dvoufaktorové ověření nebo ověření emailem
  - databázový systém
  - chování na různých prostředích (většinou hlavně nastavení pro zobrazení podrobnější chybové hlášky pro vývojové prostředí, avšak stručné pro produkci)

- adresář wwwroot

  Jedná se o adresář v kořenové složce projektu určený pro soubory, které mají být v aplikaci stále veřejně přístupné všem uživatelům.

  Mezi takové soubory patří typicky ikony, loga, obrázky a externí soubory s kaskádovými styly nebo JavaScriptem.

- adresář Views

  Tento adresář slouží k uschování speciálních .cshtml souborů. Tyto speciální soubory obsahují zdrojový kód běžného jazyka HTML, avšak mohou být navíc obohaceny o dodatečný kód v jazyce C#. Tento dodatečný kód se spouští vždy těsně před odesláním příslušného souboru uživateli stejně, jako je tomu například u jazyka PHP.

  Soubory .cshtml navíc umožňují implementaci tzv. „partials“, které umožňují celý návrh webu rozdělit do několika souborů, které jsou pak skládány dohromady. Programátor tak má možnost jednotlivé stránky rozdělit a tím si nejen zachovat přehlednost, a případně některé části webu snadno recyklovat, aniž by docházelo k duplikaci kódu.

- adresář Models

  Tento adresář obsahuje tzv. „ViewModels“. Jedná se o speciální třídy objektů, které slouží ke snazšímu předávání dat ze stránek od klienta zpět do aplikace. Na každou .cshtml stránku je možné přidat právě jeden ViewModel, jehož veřejné proměnné by měly odpovídat položkám, které je potřeba na klienta odeslat nebo od něj získat.

- adresář Data

  Pokud se u dané ASP.NET Core aplikace předpokládá, že bude využívat komunikaci s databází, pak k tomu slouží speciální třída *ApplicationDbContext*, která se stará nejen o veškerou komunikaci s databází, ale i o její strukturu a verze. Typicky je uložena ve složce *Data* spolu s dalšími soubory potřebnými pro komunikaci s databází. Mezi takové soubory patří zejména třídy reprezentující jednotlivé tabulky v databázi.

- adresář Controllers

  Dostáváme se k samotnému jádru aplikace, kterým je definice jednotlivých koncových bodů (angl. endpoints) aplikace, na které uživatelé odesílají své požadavky. Samotné koncové body jsou shlukovány do tzv. „řadičů“ (angl. controllers). Každý řadič je v aplikaci veden jako speciální třída, která dědí z třídy *Controller*. Koncové body jsou v rámci řadiče implementovány jako jednotlivé veřejné metody třídy.

  V řadičích se pak odehrává veškerá logika celé aplikace. Mají přístup k datům od uživatele ať už prostřednictvím běžného těla dotazu, parametrů v URL nebo využitím dříve zmíněných „ViewModels“. Dále se z nich volají metody objektu *ApplicationDbContext* pro přístup k databázi a v poslední řadě mají přístup k souborům ve složce Views, které mohou odeslat jako odpověď uživateli.

Aplikace byla vyvíjena ve vývojovém prostředí Microsoft Visual Studio 2019 Community Edition, které má pro projekty v ASP.NET Core širokou podporu.

6.2 Uživatelské účty

Pro zajištění podpory uživatelských účtů byla využita připravená knihovna *IdentityUser*, která je standardně součástí vývojového balíčku pro ASP.NET Core.

Tato knihovna poskytuje:

- připravenou strukturu uživatele a uživatelských rolí
- metody pro přihlášení a odhlášení (včetně dvoufaktorové varianty a ověřování údajů emailem)

- nástroje pro zajištění přihlašování pomocí externích platforem (Google, Facebook)
- instrukce pro automatické vytvoření databázové struktury
- připravené webové stránky pro přihlášení

Vzhledem k využití aplikace bylo použito pouze základní přihlašování za účelem usnadnění testování, nicméně dodatečné funkce lze díky využití této knihovny snadno doplnit.

6.3 Webové uživatelské rozhraní

Struktura všech webových stránek má společný základ pro všechny stránky, kterým je soubor *_Layout.cshtml*. Tento soubor obsahuje odkazy na přihlášení a na domovskou stránku. Do této struktury jsou pak dosazovány ostatní části.
Tyto části spadají do 3 kategorií:

- stránky pro správu agentů

  Na těchto stránkách je možné zobrazit seznam vlastněných agentů, přidávat nové a přejít do seznamu kamer každého z nich. Pro přidání nového agenta slouží vyskakovací okno, které se zobrazí při stisknutí tlačítka *Add Agent*. Definice tohoto vyskakovacího okna je ve struktuře v samostatném souboru.

- stránky pro správu kamer

  Na těchto stránkách je možné zobrazit seznam kamer, přidávat nové a přejít k záznamům konkrétní kamery. Zvolený agent je určen pomocí hodnoty *id* v URL. Platnost zadané hodnoty a práva přístupu jsou kontrolovány v samotném řadiči. Stejně jako je to mu u agentů, pro přidání nové kamery slouží vyskakovací okno, které je definované v samostatném souboru.

- stránky pro zobrazení záznamů

  Stránky pro zobrazení záznamů mají 3 úrovně:

  1. stránka pro zvolení data a času
  2. stránka pro zvolení záznamu – uživatel vybírá ze seznamu minut, ve kterých nabízené záznamy začínají
  3. stránka s přehrávačem

  Po zvolení záznamu si může uživatel daný záznam spustit přímo na stránce nebo si jej pomocí tlačítka stáhnout do svého zařízení, ze kterého je právě přihlášen.

Pro stylizaci stránek byla využita otevřená knihovna kaskádových stylů *Bootstrap*. Ikony odznaku a kamery, které jsou ve webové aplikaci použity, byly v souladu s licenčními podmínkami převzaty z portálu www.flaticon.com. Autor těchto ikon je uveden jako uživatel Freepik.

6.4 Tvorba modelu tabulky v databázi

Předpokládejme jednoduchou třídu *Car*:

```
public class Car
{
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int Id { get; set; }

        [ForeignKey("Owner")]
        public int OwnerID { get; private set; }

        public Owner Owner { get; set; }

        public string LP { get; set; }

        public string? AdditionalComponents { get; set; }
}
```

Pak je tato třída modelem reprezentujícím skutečnou tabulku *Car* v příslušné databázi. Každý záznam z tabulky *Car* bude mít **právě 4** parametry:

1. **Id**: primárním klíč daného záznamu (pomocná syntaxe [Key]), jehož hodnota je generována databází a nikoli programátorem (pomocná syntaxe [DatabaseGenerated(DatabaseGeneratedOption.Identity)])
2. **Owner**: cizí klíč z tabulky *Owner* odkazující na záznam vlastníka daného vozidla
3. **LP**: povinná hodnota datového typu varchar[MAXCHAR]
4. **AdditionalComponents**: nepovinná (NULLABLE – udáno znakem „?“ za datovým typem) hodnota datového typu varchar[MAXCHAR]

Parametr *OwnerID* se díky pomocné syntaxi *[ForeignKey("Owner")]* v tabulce nijak neprojeví. Parametr slouží programátorovi k propojení záznamů pouze na straně aplikace. Při získání záznamů z tabulky *Car* je totiž zmíněný parametr *Owner* nastaven na hodnotu NULL a k hodnotám samotného záznamu z tabulky *Owner* se pomocí jednoduchého odkázání tedy nedostaneme.

Pokud se tedy chceme dostat k parametrům vlastníka konkrétního vozidla, musíme nejdříve nalézt konkrétní záznam vozidla v tabulce *Car* a následně nalézt uživatele v tabulce *Owner* pomocí hodnoty v parametru *OwnerID*.

6.5 Databáze

V rámci tvorby aplikace byly vytvořeny 3 modely:

Model Agent

```
public class Agent
{
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int Id { get; set; }

        public string AgentID { get; set; }

        public string? ClaimCode { get; set; }

        public string? OwnerUser { get; set; }

        public string? Name { get; set; }

        public DateTime WatchDog { get; set; }
}
```

- **AgentID**: speciální ID tvořené náhodně vygenerovaným, 40místným řetězcem, skládajícím se z velkých i malých písmeny anglické abecedy a číslic

  Tímto způsobem lze vytvořit až $1,33 \cdot 10^{71}$ různých kombinací (pro srovnání, adresní prostor IPv6 nabízí „pouhých“ $3,4 \cdot 10^{38}$ adres).

  Vzorec pro výpočet možných kombinací pro AgentID je následující: $(25 + 25 + 10)^{40}$.

  Toto ID agent obdrží pří první registraci, uloží si jej a následně pomocí něj prokazuje svoji identitu při zaslání jakéhokoli dalšího dotazu.

- **ClaimCode**: 10místný kód sloužící ke snadnému propojení agenta a uživatelského účtu

  Tento kód je vygenerován při registraci agenta, kterému je následně také zaslán. Agent kód zobrazí na standardní výstup uživateli, který jej pak ve webové aplikaci zadá z přihlášeného účtu. Tato akce „přivlastnění“ způsobí, že je ClaimCode daného agenta vymazán a na místo OwnerUser je uloženo ID účtu, ze kterého k přivlastnění došlo.

- **OwnerUser**: řetězec obsahující ID z tabulky uživatelů

  Pokud je tento parametr nastavený, znamená to, že je příslušný agent vlastněný uživatelským účtem.

- **Name**: řetězec sloužící uživateli k identifikaci agenta

  Vyjma zobrazení uživateli tento parametr nenese žádnou funkci.

- **WatchDog**: parametr, do kterého je pravidelně ukládán čas ohlášení agenta

  Každý agent v pravidelném intervalu 2 minut posílá jednoduchý dotaz, kterým prokazuje aplikaci, že je aktivní a připojený k internetu. Případná informace o neaktivním agentovi je na požádání zobrazena uživateli.

Model Camera

```
public class Camera
{
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int Id { get; private set; }

        [ForeignKey("Agent")]
        public int AgentID { get; private set; }

        public Agent Agent { get; set; }

        public bool SavedAtAgent { get; set; }

        public string Name { get; set; }

        public string Username { get; set; }

        public string Password { get; set; }

        public string Codec { get; set; }

        public string URL { get; set; }

        public int RecordPeriodInMinutes { get; set; }

        public DateTime WatchDog { get; set; }
}
```

- **SavedAtAgent**: parametr značící, zda jsou informace o dané kameře uložené na příslušném agentovi

  Agent se za svého běhu pravidelně na tyto kamery dotazuje, aby si je mohl lokálně uložit a začít s nahráváním.

- **Name**: řetězec sloužící uživateli k identifikaci dané kamery v rámci agenta

  Vyjma zobrazení uživateli tento parametr nenese žádnou funkci.

- **Username**: uživatelské jméno, pomocí kterého se může agent prokázat příslušné fyzické kameře v lokální síti

- **Password**: uživatelské heslo k uvedenému uživatelskému jménu

- **Codec**: informace o využívaném kodeku kamery

  Tato informace slouží agentovi k pozdější konverzi záznamu na spustitelný formát MP4.

- **URL**: adresa fyzické kamery v **lokální síti**

  Na této adrese se agent následně snaží přistoupit k RTSP přenosu.

- **RecordPeriodInMinutes**: informace, jak dlouho má při stabilních podmínkách trvat jeden záznam v minutách
- **WatchDog**: parametr, do kterého je pravidelně ukládán čas nahrání posledního pořízeného záznamu

Model Recording

```
public class Recording
{
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int Id { get; set; }

        public Camera Camera { get; set; }

        public string Day_Hour { get; set; }

        public string Minute { get; set; }

        public DateTime Datetime { get; set; }

        public string FilePath { get; set; }
}
```

- **Day_Hour**: řetězec obsahující datum a hodinu pořízeného záznamu. Tato hodnota je zasílána agentem, aby bylo zachováno časové pásmo.
- **Minute**: řetězec obsahující informaci, ve které minutě dané hodiny začalo nahrávání. Tato hodnota je také zasílána agentem.
- **Datetime**: parametr nesoucí informaci, kdy byl zvolený záznam uložen. Tuto informaci zadává sama aplikace.
- **FilePath**: absolutní cesta k souboru se samotným záznamem v souborovém systému.

Nasazení modelů do databáze

Pro samotné vložení vytvořených modelů do databáze byly do třídy *ApplicationDbContext* přidány položky typu DbSet<K>:

```
public class ApplicationDbContext : IdentityDbContext
{
        public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)
                : base(options){}

        + public DbSet<Agent> Agents { get; set; }
        + public DbSet<Camera> Cameras { get; set; }
        + public DbSet<Recording> Recordings { get; set; }

}
```

Názvy přidaných položek poté odpovídají názvům tabulek v databázi.

6.6 Řadiče a koncové body

V rámci aplikace jsou rozlišovány 2 sady řadičů:

1. Hlavní sada obsluhující webové uživatelské rozhraní. Tyto jsou umístěné ve složce *Controllers*.
2. Sada pro komunikaci s agenty. Tyto jsou umístěné ve složce *Areas/AgentApi/Controllers*.

Sada koncových bodů pro uživatelské rozhraní

Při používání aplikace je zapotřebí, aby byl uživatel stále přihlášen. Jakýkoli dotaz od nepřihlášeného uživatele (vyjma dotazů ze skupiny */Home*) vede k automatickému přesměrování na bod */Home/Index*.

- **Soubor**: *Controllers/HomeController.cs*
  - **GET** */Home/Index*
    Výchozí koncový bod pro komunikaci s aplikací. Tento bod vyhodnotí, zda je uživatel přihlášený. Pokud ne, uživateli je zaslána úvodní stránka vybízející k přihlášení. Pokud ano, uživatel je přesměrován na koncový bod */Agent/Index*.
  - **GET** */Home/Privacy*
    Bod pro zaslání stránky s podmínkami ochrany soukromí dat uživatelů.
  - **GET** */Home/UserError*
    Pomocný bod pro zaslání stránky s chybovou hláškou pro uživatele. Tento bod je využit například, když se uživatel pokusí o přístup na nedovolenou stránku.
  - **GET** */Home/Error*
    Pomocný bod pro zaslání informační stránky v případě nečekané chyby.
- **Soubor**: *Controllers/AgentController.cs*
  - **GET** */Agent/Index*
    Bod pro zaslání stránky se seznamem vlastněných agentů.
  - **POST** */Agent/ClaimAgent*
    Bod pro přidání nového agenta – připojení agenta k účtu. Z těla modelu *AgentsViewModel* je získán uživatelem zadaný *ClaimCode*. Následně je odeslán dotaz na vyhledání záznamů z tabulky agentů s korespondujícím kódem. Pokud je takový záznam nalezen, dochází k přidělení příslušného agenta přihlášenému uživateli. V opačném případě dochází k ohlášení chyby.
- **Soubor**: *Controllers/CameraController.cs*
  - **GET** */Camera/Index/agent_id*
    Bod pro zaslání stránky se seznamem kamer příslušného agenta zvoleného podle databázového ID (nejedná se tedy o AgentID). Toto ID může být předáno buď prostřednictvím hodnoty v URL anebo přes model *CamerasViewModel*. Následně je ověřeno, že je dotazující se uživatel skutečně vlastníkem zvoleného agenta. Nakonec dochází k získání všech kamer a odeslání stránky.

- **POST** */Camera/CreateCamera*

  Bod pro vytvoření kamery na zvoleném agentovi. Veškeré potřebné údaje o nové kameře jsou obsaženy v *CamerasViewModel*. Dochází opět ke kontrole vlastnictví příslušného agenta a následně k vytvoření nové instance třídy *Camera* zakončeném uložením nového záznamu do databáze.

- **Soubor**: *Controllers/RecordingController.cs*

  - **GET** */Recording/Folders/{agent_id}/{camera_id}*

    Bod pro zaslání stránky se seznamem termínů záznamů ze zvolené kamery. Termíny jsou z databáze získány jako parametry *Day_Hour*, přičemž duplicitní záznamy jsou vypsány pouze jednou.

    Uživatel si pak z poskytnutého seznamu vybírá den a hodinu, kdy začalo pořizování jím hledaného záznamu.

  - **GET** */Recording/Records/{agent_id}/{camera_id}/{record_dh}*

    Bod pro zaslání stránky se seznamem termínů záznamů ze zvolené kamery. Termíny jsou z databáze získány jako parametry *Minute*.

    Uživatel si pak z poskytnutého seznamu vybírá konkrétní záznam na základě minuty, kdy začalo pořizování jím hledaného záznamu.

  - **GET** */Recording/Player/{agent_id}/{camera_id}/{record_dh}/{min}*

    Bod pro zaslání stránky s webovým přehrávačem. Stránka uživateli zobrazuje datum, hodinu a minutu zvoleného záznamu, webový přehrávač s pořízeným záznamem a tlačítko pro stažení záznamu.

    Zdroj webového přehrávače je automaticky nastaven na bod */Recording/PlayRecording/* s parametry požadovaného záznamu. Zdroj tlačítka pro stažení záznamu je pak nastaven na bod */Recording/GetRecording/* se stejnými parametry.

  - **GET** */Recording/PlayRecording/{agent_id}/{camera_id}/{record_dh}/{min}*

    Tento bod se stará o správné zprostředkování záznamu pro online přehrání. Před samotným odesláním záznamu dochází k ověření práv k přístupu přihlášeného uživatele.

    Samotné odeslání záznamu pak má nastavený příznak enableRangeProcessing na hodnotu logické 1. Díky tomu je možné především u větších záznamů vždy přenášet pouze tu část záznamu, na kterou se chce uživatel dívat. Navíc nepodmiňuje načtení celého souboru do vyrovnávací paměti zařízení před samotným přehráním, ale pouze zvolené přehrávané části. Dochází tak významnému ušetření výpočetních prostředků uživatele.

    Tyto zmiňované akce za uživatele automaticky provádí webový přehrávač.

  - **GET** */Recording/GetRecording/{agent_id}/{camera_id}/{record_dh}/{min}*

    Tento bod se stará o správné stáhnutí záznamu do zařízení. Stejně jako u */Recording/PlayRecording/* dochází před samotným odesláním záznamu k ověření práv k přístupu přihlášeného uživatele. Následně je v závislosti na nastavení prohlížeče uživatele zvolený záznam stažen na zařízení.

Sada koncových bodů pro komunikaci s agenty

Tato sada bodů slouží pouze ke komunikaci s agentskou aplikací spuštěné v místní síti uživatele. Součástí každého dotazu zaslaného z agenta musí být unikátní AgentID, které agent získal při registraci. Neuvedení AgentID nebo jeho podvrh vede na chybu 400 „Bad request“

Jedinou výjimkou je dotaz zaslaný na bod */AgentApi/Agent/RegisterAgent*, který slouží k registraci nového agenta.

- **Soubor**: *Areas/AgentApi/Controllers/AgentController.cs*
  - **GET** */AgentApi/Agent/PingAgent*
    Testovací bod sloužící k ověření dostupnosti rozhranní pro agenty.
  - **POST** */AgentApi/Agent/CheckAgent*
    Kontrolní bod ověřující pravost zaslaného AgentID a stav, zda je nastaven jako vlastněný.
  - **POST** */AgentApi/Agent/RegisterAgent*
    Koncový bod pro registraci nového agenta. Čerstvě spuštěný agent pomocí tohoto bodu ohlašuje svoji existenci aplikaci.
    Na základě tohoto požadavku jsou vygenerovány dva alfanumerické kódy:
    * AgentID: 40místný
    * ClaimCode: 10místný

    Po vygenerování je zkontrolována jejich unikátnost v databázi – pokud by i při nízké pravděpodobnosti došlo k vygenerování kódu, který už je přiřazen jinému agentovi, dojde k novému generování.
    Následně je vytvořena nová instance třídy *Agent*, která je uložena jako nový záznam do databáze. Nakonec jsou oba kódy odeslány zpět jako odpověď tázajícímu se agentovi.

- **Soubor**: *Areas/AgentApi/Controllers/CameraController.cs*
  - **POST** */AgentApi/Camera/RequestSavedCameras*
    Bod, který agentovi poskytne aktuální seznam kamer, ze kterých má nahrávat.
  - **POST** */AgentApi/Camera/RequestNewCameras*
    Bod který agentovi poskytne seznam kamer, které byly nově vytvořeny, a které si ještě neuložil.
  - **POST** */AgentApi/Camera/ConfirmCamera*
    Bod pomocí kterého agent potvrzuje úspěšné přijetí nové kamery ze seznamu nově vytvořených kamer a zahájení nahrávání záznamu.

- **Soubor**: *Areas/AgentApi/Controllers/RecordController.cs*
  - **POST** */AgentApi/Record/UploadFile*

    Koncový bod pro nahrávání nových záznamů.

    Před samotným uložením souboru a zpřístupněním uživateli je však provedena řada kontrolních akcí:

    1. Požadavek zaslaný na tento bod musí být ve tvaru formuláře.
    2. Maximální počet souborů obsažených ve formuláři je 1
    3. Maximální velikost nahrávaného souboru je 2GB
    4. Požadavek obsahuje všechny identifikační údaje (AgentID, id kamery, datum a hodinu pořízení záznamu, minutu pořízení záznamu)

    Při úspěšné kontrole dochází k vytvoření nové instance třídy *Recording* do které jsou uloženy informace z těla formuláře a následně k samotnému uložení souboru do souborového systému aplikace.

    1. Vytvoření nové složky (pokud neexistuje)
    2. Vytvoření nového souboru s příponou .mp4
    3. Postupné ukládání datového toku do souboru
    4. Uzavření souboru
    5. Uložení instance jako nový záznam do databáze

Kapitola 7

Agentská aplikace

7.1 Struktura

Aplikace byla vytvořena za pomoci vývojové platformy .NET Core, která je blíže popsaná v předchozí kapitole. Jako typ projektu byla zvolen model standardní konzolové aplikace.

Aplikace je rozdělena do 7 modulů:

- soubor Program.cs

  Vstupní soubor s funkcí *Main*

- Setup

  Třída obsahující metody pro kontrolu nastavení hostitelské platformy a uchování informací o nastavení aplikace

- CamerasSetup

  Třída obsahující informace o obsluhovaných kamerách, a sadu metod pro komunikaci s webovou aplikací

- Recorder

  Třída sloužící k zapouzdření a obsluhování nahrávacích procesů z IP kamer

- RtspReceiver

  Třída definující postup samotného nahrávání záznamu z RTSP přenosu IP kamery

- Uploader

  Třída sloužící k zapouzdření a obsluhování odesílání záznamů na webovou aplikaci

- Modely

  Pomocné třídy sloužící jako modely pro deserializaci zpráv z webové aplikace

7.2 Program.cs

Program.cs je vstupním souborem s funkcí *Main*. Ve funkci *Main* je implementováno využití ostatních modulů pro systematický chod celé aplikace.

Po spuštění jsou provedeny tyto kroky:

1. Vytvoření instance třídy *HttpClient* pro komunikaci s webovou aplikací
2. Nastavení adresy URL odkazující na online verzi webové aplikace – uživatel může zadat URL jako 1. parametr při spuštění aplikace (jinak je nastavena předdefinovaná hodnota „https://ibtcloudsecapp.azurewebsites.net“)
3. Vytvoření instance třídy *SystemSetup* a spuštění její metody *Run*
4. Vytvoření instance třídy *CamerasSetup*
5. Vytvoření instance třídy *Uploader*
6. Vytvoření instance třídy *Recorder* a spuštění nahrávání kamer voláním její metody *AddAndRunListOfCameras*
7. Spuštění hlavní nekonečné smyčky programu.

Hlavní smyčka každých 5 sekund provádí ikrementaci kontrolní hodnoty. Pomocí této hodnoty je:

1. každou 1 minutu proveden pokus o znovunahrání záznamů, u kterých došlo k selhání.
2. každých 5 vteřin provedena kontrola běhu všech vláken odpovědných za nahrávání obrazů z kamer a případné odeslání záznamu, který byl před selháním pořízen.
3. každou 1 minutu odeslán požadavek na zaslání seznamu kamer, které byly nově přidány uživatelem a jejich případné potvrzení webové aplikaci.
4. každou 1 minutu odesláno ohlášení webové aplikaci s informací, že daný agent v pořádku běží.

Kontrolní hodnota je po dosažení hodnoty 720 (každou hodinu) resetována zpět na 0 ().

7.3 SystemSetup

Třída *SystemSetup* slouží k prvotnímu ověření dostupnosti zdrojů, které jsou za běhu aplikace zapotřebí. Celé ověřování je spuštěno voláním veřejné funkce *Run* a je rozděleno do 3 kategorií:

- Kontrola nástroje FFmpeg

  Funkce vytvoří nový proces, v rámci kterého provede příkaz „ffmpeg -version“. Úspěšné skončení procesu v intervalu 2 vteřin značí správně nastavený nástroj.

- Kontrola a případné vytvoření adresářové struktury v dedikované složce *agentapp*

  Funkce zkontroluje existenci a případně vytvoří složku *./agentapp* v kořenové složce aplikace. Uvnitř této složky následně vytvoří dva další adresáře – *./agentapp/.config* a *./agentapp/records*

- Kontrola komunikace s webovou aplikací

  Funkce zkontroluje existenci souboru *./agentapp/.config/agentid*, který obsahuje unikátní AgentID konkrétní aplikace.

Pokud tento soubor chybí – aplikace odešle žádost o nový odesláním požadavku na registraci. Získané AgentID uloží do zmíněného souboru pro příští spuštění a ClaimCode do souboru *./agentapp/.config/claimcode* pro případ, že by došlo k ukončení aplikace ještě před tím, než bude agent na straně webové aplikace přivlastněn uživatelským účtem.

Pokud je soubor nalezen – aplikace odešle žádost o ověření pravosti kódu. Při potvrzení kódu je ze zprávy také vyčtena informace o vlastnictví. Pokud agent ještě nebyl zabrán žádným uživatelským účtem, pokusí se o čtení uloženého souboru s ClaimCode a jeho zobrazení na standardní výstup. Následně zahájí čekání dokud nebude zvolený agent přivlastněn.

7.4 CamerasSetup

Třída *CamerasSetup* slouží k získávání a uchování informací o všech kamerách obsluhovaných daným agentem.

Disponuje 3 metodami:

- **GetSavedCameras**: získává seznam kamer, které byly už v minulosti potvrzeny, a ukládá jej do proměnné přístupné zbytku aplikace. Děje se tak odesláním dotazu na webovou aplikaci a následným uložením odpovědi do souboru. Je-li komunikace neúspěšná, pokusí se seznam vyčíst právě z tohoto náhradního souboru – pokud existuje.
- **GetNewCameras**: získává seznam nových kamer z webové aplikace.
- **ConfirmCamera**: potvrzuje webové aplikaci přijetí a úspěšné nastavení nové kamery.

7.5 Uploader

Třída *Uploader* obsluhuje odesílání všech pořízených záznamů na webovou aplikaci.

Odeslání obstarává hlavní metoda *Upload*. Při zavolání vytvoří instanci třídy *MultipartFormDataContent*, který zajišťuje stejnou strukturu zprávy jakou můžeme najít u běžných webových formulářů označovaných speciální značkou *<form>*. Do zprávy jsou následně načteny bajty záznamu a identifikační informace (AgentID, CameraID, Day_Hour, Minute) a zpráva je odeslána.

Pokud dojde k selhání, připravený záznam je uložen do seznamu neúspěšně odeslaných záznamů a později dojde k pokusu jej odeslat znovu.

Toto opětovné odeslání je zahájeno voláním metody *ReuploadFailedRecs*, která začne procházet zmiňované pole a postupně opětovně volá hlavní funkci *Upload*.

7.6 RtspReceiver

Tato třída slouží k samotnému pořizování záznamu z konkrétní kamery. Zprostředkovává získávání jednotlivých rámců, které přichází datovým tokem z kamery a jejich ukládání do dedikovaného souboru. K tomu využívá speciální třídu *RtspClient* z externí knihovny *RtspClientSharp*[2].

Startovací funkce *StartReceiving* připraví novou instanci třídy *RtspClient* s potřebnými parametry, podle požadovaného typu záznamu nastaví obsluhovací funkci, vytvoří nový soubor pro záznam a nakonec vytvoří nové vlákno aplikace, které se bude starat o příjem a zpracování paketů.
Obsluhovací funkce jsou rozděleny na dva typy podle podporovaných formátů:

- Funkce pro záznam formátu H.264
- Funkce pro záznam formátu JPEG

Funkce pro zpracování záznamu ve formátu H.264 při příchodu každého paketu provádí následující:

1. Zkontroluje, zda se jedná o tzv. „IFrame“
2. Pokud ano:
   (a) Zkontroluje, zda skončilo časové okno pro zvolený záznam:
   (b) Pokud ano:
      i. Uzavře právě nahrávaný soubor a zavolá funkci ze třídy *Recorder* pro zpracování nového záznamu.
      ii. Vytvoří nový soubor.
      iii. Uloží příchozí paket do nového souboru.
   (c) Pokud ne:
   Uloží příchozí paket do připraveného souboru.
3. Pokud ne:
   Uloží příchozí paket do připraveného souboru.

Funkce pro zpracování záznamu ve formátu H.264 při příchodu každého paketu provádí následující:

1. Zkontroluje, zda skončilo časové okno pro zvolený záznam:
2. Pokud ano:
   (a) Uzavře právě nahrávaný soubor a zavolá funkci ze třídy *Recorder* pro zpracování nového záznamu.
   (b) Vytvoří nový soubor.
   (c) Uloží příchozí paket do nového souboru.
3. Pokud ne:
   Uloží příchozí paket do připraveného souboru.

7.7 Recorder

Třída *Recorder* zapouzdřuje veškerou práci s obsluhovanými kamerami. Obsahuje seznam instancí třídy *RtspReceiver*, u kterých provádí kontrolu správného běhu voláním metody *CheckReceivers*.

Dále obsahuje definice 2 metod pro zpracování záznamu nahraného některou ze zmiňovaných instancí.
Funkce pro zpracování nahraného záznamu jsou 2, neboť odpovídají možným datovým typům záznamu:

- Funkce pro záznamy ve formátu H.264 nejprve pořízený záznam konvertuje pomocí nástroje *FFmpeg* do formátu MP4 a následně informace o záznamu předá instanci třídy *Uploader*, která záznam odešle webové aplikaci.
- Funkce pro záznam formátu JPEG pořízený záznam pouze předává k odeslání.

Kapitola 8

Testování

Rozhraní aplikace bylo průběžně testováno pomocí aplikace Postman API. Pro ověření správného nahrávání záznamů byl vytvořen speciální soubor *uploadsouboru.html* obsahující formulář, který je identický s formulářem používaným v agentské aplikaci. Dále byl vytvořen seznam testovacích scénářů, které mají za úkol prověřit odolnost systému a splnění požadavků uvedených v předchozí podkapitole Analýza a specifikace požadavků.

Na závěr byl systém podroben 24hodinovému testu, během kterého prováděl nahrávání. Během tohoto nahrávání byla v náhodných okamžicích odpojována a zpátky připojována kamera i internetové připojení.

Testovací scénáře

Všechny scénáře předpokládají stabilní spojení mezi webovou aplikací a agentskou aplikací, a zároveň webovou aplikací a uživatelem – pokud není v daném scénáři uvedeno jinak.

Registrace

Cíl: Ověření funkčnosti registrace a přihlašování
Požadavky: uživatel není přihlášen
Kroky:

1. Otevřete webovou aplikaci.
2. Stiskněte tlačítko „Register“ v pravém horním rohu.
3. Po přesměrování zadejte validní hodnoty do zobrazených polí.
4. Stiskněte tlačítko „Register“ pod vyplněnými poli.
5. Při zobrazení chybové hlášky „User name ’xxx’ is already taken.“ opakujte předchozí kroky s jinou hodnotou v poli Email.
6. Po úspěšném přesměrování klikněte na odkaz „Click here to confirm your account“.
7. Po úspěšném přesměrování na stránku uvádějící nápis „Confirm email“ stiskněte tlačítko „Login“ v pravém horním rohu.
8. Po přesměrování zadejte stejné hodnoty do zobrazených polí, jako jste zadali při registraci.

9. Stiskněte tlačítko „Log in“.

10. Počkejte až budete přesměrováni na úvodní stránku.

Výsledek:
Úspěch: Došlo k úspěšnému přihlášení a je zobrazena webová stránka s textem „You don’t have any agents yet. :) “
Selhání: Testování nebylo možné dokončit z důvodu výskytu neřešitelné chyby.

Přidání agenta

Cíl: Ověření funkčnosti propojení agenta s webovou aplikací
Požadavky: uživatel je přihlášen ve webové aplikaci a je na úvodní stránce pro agenty, uživatel má připravené zařízení s agentskou aplikací a nainstalovaným nástrojem FFmpeg (ffmepg -version)
Kroky:

1. Spusťte agentskou aplikaci na zařízení.

2. Řiďte se případnými pokyny vypsanými do konzole.

3. Po výzvě k propojení s webovou aplikací zkopírujte z konzole zobrazený „ClaimCode“.

4. Přejděte do webové aplikace.

5. Stisknutím tlačítka „Add Agent“ otevřete vyskakovací okno.

6. Do pole „Claim Code“ zkopírujte 10místný kód z konzole aplikace.

7. Pole „Name“ vyplňte libovolným textem. (například: „test case 2 agent“)

8. Stiskněte tlačítko „Add Agent“ ve spodní části vyskakovacího okna.

Výsledek:
Úspěch: Ve webové aplikaci se zeleně zobrazila hláška o přidání nového agenta a nový agent se zobrazuje v seznamu.
Selhání: Testování nebylo možné dokončit z důvodu výskytu neřešitelné chyby.

Přidání kamery

Cíl: Ověření funkčnosti přípojení agenta na IP kameru a odesílání záznamů na webovou aplikaci
Požadavky: uživatel je přihlášen ve webové aplikaci a je na úvodní stránce pro agenty, uživatel vlastní alespoň jednoho agenta, zvolený agent běží, uživatel má přístup ke konzoli běžícího agenta, v lokální síti agenta běží IP kamera s RTSP přenosem záznamu – je možné se na daný RTSP stream připojit například pomocí aplikace VLC.
Kroky:

1. Stiskněte tlačítko „Cameras“ u příslušného agenta.

2. Stiskněte tlačítko „Add Camera“.

3. Vyplňte požadované údaje

- URL pro přenos – jedná se o stejnou adresu jaká se zadává ze zařízení ve stejné lokální síti
- Camera Credentials – jedná se o přihlašovací údaje k přenosu z kamery
- Časování nastavte na 1 minutu pro rychlé provedení testů.

4. Stiskněte tlačítko „Add Camera“ ve spodní části vyskakovacího okna.
5. Zkontrolujte zobrazení zeleného upozornění o úspěšném přidání kamery a její přidání do seznamu kamer zvoleného agenta.
6. Na konzoli agenta čekejte na zprávu informující o tom, že agent získal zprávu o přidání kamery (tato akce může trvat až 3 minuty)
7. Po zobrazení této informace zkontrolujte změnu stavu kamery ve webové aplikaci na „online“.
8. Podle nastaveného počtu minut počkejte danou dobu.
9. Po uplynutí doby stiskněte na tlačítko „Recordings“ na položce příslušné kamery.
10. Zkontroluje, že dochází ke správnému nahrávání záznamů.

Výsledek:
Úspěch: Ve webové aplikaci se zobrazila nová kamera. U nové kamery se dá stisknout na tlačítko „Recordings“ a je možné procházet záznamy.
Selhání: Testování nebylo možné dokončit z důvodu výskytu neřešitelné chyby.

Smazání kamery

Cíl: Ověření správného vymazání kamery a pořízených záznamů ze systému
Požadavky: existuje uživatelský účet s přivlastněným agentem a nahrávající kamerou, příslušná agentská aplikace běží, uživatel je na příslušném účtu přihlášen
Kroky:

1. Stiskněte tlačítko „Delete“ u příslušné kamery.
2. Potvrďte zobrazené vyskakovací okno.
3. Restartuje agentskou aplikaci.

Výsledek:
Úspěch: Agentská aplikace se znovu načetla a v seznamu spouštěných kamer se smazaná kamera už nenachází. Ve webové aplikaci položka zvolené kamery zmizela. V databázi webové aplikaci zmizely záznamy o kameře a všech pořízených nahrávkách. Z úložiště zmizely fyzické soubory s příslušnými nahrávkami.
Selhání: Testování nebylo možné dokončit z důvodu výskytu neřešitelné chyby.

Smazání agenta

Cíl: Ověření správného vymazání agenta, jeho kamer a pořízených záznamů ze systému
Požadavky: existuje uživatelský účet s přivlastněným agentem a nahrávající kamerou, příslušná agentská aplikace běží, uživatel je na příslušném účtu přihlášen
Kroky:

1. Stiskněte tlačítko „Delete“ u příslušného agenta.
2. Potvrďte zobrazené vyskakovací okno.
3. Restartuje agentskou aplikaci.

Výsledek:
Úspěch: Agentská aplikace zaznamenala chybu AgentID, resetovala se a čeká na nové přidělení. Ve webové aplikaci položka zvoleného agenta zmizela. V databázi webové aplikaci zmizely záznamy o agentovi, jeho kamerách a všech pořízených nahrávkách. Z úložiště zmizely fyzické soubory s příslušnými nahrávkami.
Selhání: Testování nebylo možné dokončit z důvodu výskytu neřešitelné chyby.

Zotavení nepřivlastněného agenta z ukončení

Cíl: Ověření správného chování agenta při jeho ukončení ještě před přijetím informace na přivlastnění
Požadavky: agentská aplikace nebyla zatím spuštěna nebo byla resetována
Kroky:

1. Spusťte agentskou aplikaci.
2. Vyčkejte na zobrazení výzvy k přivlastnění.
3. Poznamenejte si zobrazený *Claim Code*.
4. Restartujte agentskou aplikaci.
5. Vyčkejte na zobrazení výzvy k přivlastnění.
6. Porovnejte původní kód s právě zobrazeným.
7. Použijte původní kód ve webové aplikaci k přivlastnění agenta.

Výsledek:
Úspěch: Nový agent byl úspěšně přidán.
Selhání: Kódy se v 6. kroku neshodují nebo testování nebylo možné dokončit z důvodu výskytu neřešitelné chyby.

Zotavení přivlastněného agenta z ukončení

Cíl: Ověření správného chování agenta při jeho ukončení.
Požadavky: agentská aplikace běží a je přivlastněna, daný agent má přidělenou alespoň 1 kameru
Kroky:

1. Restartujte agentskou aplikaci

Výsledek:
Úspěch: Agentská aplikace se úspěšně spustila a začala automaticky s nahráváním ze všech kamer.
Selhání: Testování nebylo možné dokončit z důvodu výskytu neřešitelné chyby.

Zotavení agenta z výpadku spojení s webovou aplikací

Cíl: Ověření správného chování agenta při výpadku spojení s webovou aplikací.
Požadavky: agentská aplikace běží a je přivlastněna, daný agent má přidělenou alespoň 1 kameru
Kroky:

1. Odpojte agentskou aplikaci od spojení s webovou aplikací (např. i vypnutím webové aplikace)
2. Vyčkejte dostatečně dlouhou dobu, aby bylo pořízeno několik záznamů.
3. Agentskou aplikace opět propojte s webovou.

Výsledek:
Úspěch: Po opětovném spojení agentská aplikace automaticky dodatečně nahrála všechny záznamy, které nahrála během doby bez spojení.
Selhání: Testování nebylo možné dokončit z důvodu výskytu neřešitelné chyby.

Zotavení agenta z výpadku kamery

Cíl: věření správného chování agenta při výpadku spojení s kamerou.
Požadavky: agentská aplikace běží a je přivlastněna, daný agent má přidělenou alespoň 1 kameru
Kroky:

1. Odpojte agentskou aplikaci od spojení s IP kamerou
2. Ve webové aplikaci zkontrolujte, že došlo k nahrání záznamu.
3. Kameru opět připojte.
4. Zkontrolujte, že agentská aplikace začala z dané kamery opět nahrávat.

Výsledek:
Úspěch: Ve 2. kroku došlo k uložení záznamu. Ve 4. kroku agentská aplikace úspěšně obnovila nahrávání.
Selhání: Testování nebylo možné dokončit z důvodu výskytu neřešitelné chyby.

Kapitola 9

Shrnutí

9.1 Dosažené výsledky

1. Bezpečnostní systém úspěšně pořizuje záznamy z IP kamer a ukládá je na webovou aplikaci.
2. Uživatel může záznamy zobrazit, přehrát a stáhnout.
3. Záznamy jsou zabezpečeny právy uživatelských účtů – neoprávněná manipulace s URL adresou je detekována a zablokována.
4. Aplikaci se podařilo úspěšně nasadit a spustit na hostujícím prostředí Azure.
5. Webové uživatelské rozhraní se vhodně přizpůsobuje různým rozlišením displejů.
6. Webová i agentská aplikace úspěšně splnily testovací scénáře uvedené v kapitole Testování.

9.2 Návrhy na rozšíření

1. Automatické mazání zastaralých záznamů za účelem uživatelsky přívětivějšího hospodaření s úložným prostorem.
2. Zprostředkování živého přenosu z kamer.
3. Povolení úpravy již uložených položek.
4. Ukládání i audio stopy záznamu.
5. Rozšíření seznamu podporovaných formátů.
6. Zavedení podrobnějšího rozdělení záznamů – oddělení data a hodiny, u minut uvádět nejen úvodní, ale i koncovou minutu.
7. Převedení ukládání záznamů ve webové aplikaci z místního úložiště do tzv. Azure Blob Storage, které je pro takový styl ukládání určené.

Kapitola 10

Závěr

Tato bakalářská práce se zabývala vývojem síťového bezpečnostního systému s využitím IP kamer třetích stran. Zaměřovala se zejména na dvě problematiky.

První problematikou bylo řešení bezpečného a přirozeného vzdáleného přístupu k záznamům pořízeným IP kamerou z míst mimo lokální síť. Bezpečnostní systém byl proto rozdělen na dvě úzce spolupracující aplikace – webovou a agentskou.

Webová aplikace je dostupná veřejně na internetu, a tím je tedy zajištěna i přístupnost z libovolného zařízení a místa. Agentská aplikace pak neustále běží na hostitelském počítači u zákazníka v lokální síti. Díky tomu se agentská aplikace může na IP kamery uživatele připojit přímo a pořízené záznamy odesílat přes REST API rozhraní webové aplikace, odkud jsou zpřístupněny uživateli.

Druhá problematika práce se týkala co nejpřirozenější podpory IP kamer od nejširší množiny výrobců. Tohoto cíle bylo dosaženo využitím RTSP přenosu, který je podporován většinou kamer. Navržený systém se připojí na živý přenos z kamery a ukládá získávané pakety do vytvořeného souboru. Následně pomocí volně dostupného nástroje FFmpeg provede konverzi záznamu na formát MP4.

Systém byl úspěšně dokončen a je připraven ke každodennímu používání.

Literatura

[1] Belshe, M., Peon, R. a Thomson, M. *Hypertext Transfer Protocol Version 2 (HTTP/2)* [online]. Květen 2015 [cit. 2021-01-14]. Dostupné z: https://tools.ietf.org/html/rfc7540.

[2] Bogdanov, K. *RtspClientSharp* [online]. github.com, 2019 [cit. 2021-04-27]. Dostupné z: https://github.com/BogdanovKirill/RtspClientSharp.

[3] Dahl, R. *About Node.js* [online]. Node.js, 2021 [cit. 2021-01-15]. Dostupné z: https://nodejs.org/en/about/.

[4] *Help* [online]. freeip.com, 2021 [cit. 2021-01-16]. Dostupné z: https://www.freeip.com/guide.

[5] Hujka, P. *Real - Time Transport Protocol a aplikační rozhraní RTP Java Media Framework* [online]. elektrorevue.cz, květen 2013 [cit. 2021-01-13]. Dostupné z: http://www.elektrorevue.cz/clanky/03018/index.html.

[6] Khandelwal, A. *React.js (Introduction and Working)* [online]. GeeksforGeeks, leden 2021 [cit. 2021-01-15]. Dostupné z: https://www.geeksforgeeks.org/react-js-introduction-working/.

[7] Lord, N. *What is FTP Security? Securing FTP Usage* [online]. digitalguardian.com, září 2018 [cit. 2021-01-14]. Dostupné z: https://digitalguardian.com/blog/what-ftp-security-securing-ftp-usage.

[8] ltd, D. I. . S. *IP CCTV vs Analogue: Pros & Cons* [online]. disnetwork.co.uk, září 2020 [cit. 2020-12-22]. Dostupné z: https://www.disnetwork.co.uk/ip-cctv-vs-analogue-pros-cons/.

[9] Ludwig, S. *Pro's and Cons for IP vs. Analog Video Surveillance* [online]. securitymagazine.com, duben 2018 [cit. 2020-12-22]. Dostupné z: https://www.securitymagazine.com/articles/88854-pros-and-cons-for-ip-vs-analog-video-surveillance.

[10] *ONVIF Home* [online]. onvif.org, 2021 [cit. 2021-04-15]. Dostupné z: https://www.onvif.org/.

[11] *Profile A* [online]. onvif.org, 2021 [cit. 2021-04-15]. Dostupné z: https://www.onvif.org/profiles/profile-a/.

[12] *Profile C* [online]. onvif.org, 2021 [cit. 2021-04-15]. Dostupné z: https://www.onvif.org/profiles/profile-c/.

[13] *Profile G* [online]. onvif.org, 2021 [cit. 2021-04-15]. Dostupné z: https://www.onvif.org/profiles/profile-g/.

[14] *Profile Q* [online]. onvif.org, 2021 [cit. 2021-04-15]. Dostupné z: https://www.onvif.org/profiles/profile-q/.

[15] *Profile S* [online]. onvif.org, 2021 [cit. 2021-04-15]. Dostupné z: https://www.onvif.org/profiles/profile-s/.

[16] *Profile T* [online]. onvif.org, 2021 [cit. 2021-04-15]. Dostupné z: https://www.onvif.org/profiles/profile-t/.

[17] Postel, J. a Reynolds, J. *FILE TRANSFER PROTOCOL (FTP)* [online]. říjen 1985 [cit. 2021-01-14]. Dostupné z: https://tools.ietf.org/html/rfc959.

[18] Ptáčník, J. *Kam ukládat filmy, hudbu a fotky? Na datové úložiště NAS* [online]. digilidi.cz, prosinec 2016 [cit. 2021-01-14]. Dostupné z: https://www.digilidi.cz/datove-uloziste-nas.

[19] *Quick start manual* [online]. q-see.com.au, 2014 [cit. 2021-01-16]. Dostupné z: http://www.q-see.com.au/wp-content/uploads/2017/03/QTN_IPCamera_Guide_v1.0.pdf.

[20] Roth, D., Anderson, R. a Luttin, S. *Introduction to ASP.NET Core* [online]. Microsoft, duben 2020 [cit. 2021-01-15]. Dostupné z: https://docs.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-5.0.

[21] Schulzrinne, H., A.Rao, R.Lanphier, Westerlund, M. a Stiemerling, M. *Real-Time Streaming Protocol Version 2.0* [online]. Prosinec 2016 [cit. 2021-01-13]. Dostupné z: https://tools.ietf.org/html/rfc7826.

[22] Schulzrinne, H., Casner, S., Frederick, R. a Jacobson, V. *RTP: A Transport Protocol for Real-Time Applications* [online]. červenec 2003 [cit. 2021-01-13]. Dostupné z: https://tools.ietf.org/html/rfc3550.

[23] tdykstra, BillWagner, Youssef1313, mairaw, realcooltrev et al. *Introduction to .NET* [online]. Microsoft, 2020 [cit. 2021-01-15]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/core/introduction.

[24] Walke, J. *React A JavaScript library for building user interfaces* [online]. React, říjen 2020 [cit. 2021-01-15]. Dostupné z: https://reactjs.org/.

[25] You, E. *The Progressive JavaScript Framework* [online]. Vue.js, 2021 [cit. 2021-01-15]. Dostupné z: https://v3.vuejs.org/.

Příloha A

Uživatelský manuál

A.1 Spuštění

Ověřte správné spojení mezi zařízením, na kterém poběží agentská aplikace, a IP kamerou. Také ověřte, že zvolená IP kamera umožňuje RTSP přenos.

Otevřete stránky webové aplikace. Pokud nemáte uživatelský účet, klikněte na odkaz „Register“ v pravém horním rohu a proveďte registraci. Pokud uživatelský účet už máte, proveďte přihlášení (odkaz „Login“ v pravém horním rohu). Po přihlášení budete na úvodní obrazovce.

Online verze webové aplikace je přístupná na adrese:
https://ibtcloudsecapp.azurewebsites.net.

A.2 Přidání agenta

Spusťte agentskou aplikaci. Vyčkejte na zobrazení 10místného alfanumerikého kódu pojmenovaného jako Claim Code. Vypsání kódu je následováno vypsáním hlášky „Waiting for claim...“. V tento okamžik přejděte zpět do webové aplikace a stiskněte modré tlačítko „Add Agent“. Do vyskakovacího okna zadejte zmíněný 10místný kód z konzole agentské aplikace a zadejte název daného agenta (dle svého uvážení). Akci dokončete stisknutím druhého modrého tlačítka „Add Agent“ ve spodní části vyskakovacího okna.

Správné přidání agenta bude oznámeno zeleným informačním oknem a zobrazením nové karty na úvodní stránce.

Pokud chcete daného agenta přesměrovat na jinou webovou aplikaci, než základní https://ibtcloudsecapp.azurewebsites.net, můžete novou adresu specifikovat jejím zadáním jako 1. parametr při spuštění.

A.3 Přidání kamery

Přejděte do seznamu kamer agenta stisknutím modrého tlačítka „Cameras“. Na zobrazené stránce stiskněte na modré tlačítko „Add Camera“. Do vyskakovacího okna zadejte požadované informace. Zvýšenou pozornost věnujte přihlašovacím údajům ke kameře a adrese URL. Adresu URL bude využívat agent ve Vaší lokální síti. Hodnotu *Record period in minutes* je pro testovací účely doporučeno nastavit na nízkou hodnotu. Zadaný počet minut

potrvá, než se objeví první záznam. Akci dokončete stisknutím modrého tlačítka „Save changes“ ve spodní části vyskakovacího okna.

A.4 Správa záznamů

Přejděte do seznamu záznamů stisknutím modrého tlačítka „Cameras“ na kartě příslušného agenta a následně stisknutím modrého tlačítka „Recordings“ na kartě příslušné kamery. V zobrazeném seznamu si vyberte skupinu záznamů (formát zápisu je DDMMYYYY_HH). Pro zobrazní záznamů stiskněte modré tlačítko „Open“.

V zobrazeném seznamu se budou nacházet už konkrétní záznamy. Hodnoty odpovídají minutě ve které začalo nahrávání. Pro zobrazení záznamu stiskněte modré tlačítko „Play“.

Nyní se nacházíte na stránce přehrávače. Můžete si záznam přehrát pomocí webového přehrávače, stáhnout pomocí žlutého tlačítka „Download recording“ nebo jej smazat.

A.5 Mazání

Pro smazání konkrétního záznamu přejděte na stránku přehrávače pro konkrétní záznam. Na stránce stiskněte červené tlačítko „Delete recording“.

Pro smazání kamery přejděte do seznamu kamer. Na kartě dané kamery stiskněte červené tlačítko „Delete“. Nakonec potvrďte vyskakovací okno.

Pro smazání agenta přejděte na úvodní stránku. Na kartě daného agenta stiskněte červené tlačítko „Delete“. Nakonec potvrďte vyskakovací okno.

Při mazání jsou odstraněny informace z databáze a fyzické soubory ze souborového systému. Veškeré záznamy však nadále přetrvávají v lokálním úložišti agenta. Pro jejich odstranění je potřeba takové odstranění provést ručně nebo resetováním agenta.

A.6 Reset agenta

Reset agenta se provádí spuštěním agenta s jediným parametrem „–reset“ nebo „-r“. Při resetování jsou smazány všechny pořízené záznamy a agent je uveden do výchozího stavu.