VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2019

Petr Matěj

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Petr Matěj**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: profiq s.r.o.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Pavel Dohnálek**

Konzultant bakalářské práce: Bc. Petr Večeřa

Datum zadání: 01.09.2018

Datum odevzdání: 30.04.2019

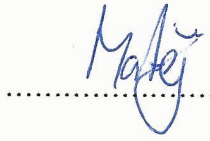
VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA OSTRAVA Fakulta elektrotechniky 5

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 11. dubna 2019



Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.

V Ostravě 11. dubna 2019 .....

Tímto bych chtěl poděkovat vedení firmy profiq s.r.o za možnost vykonávat bakalářskou praxi v rámci mé běžné pracovní náplně. Jmenovitě mé poděkování patří Bc. Petru Večeřovi a Ing. Pavlu Dohnálkovi, kteří mě při práci vedli, podporovali a poskytovali cenné rady při zpracovávání této práce.

Abstrakt

Touto prací popisuji průběh odborné praxe ve společnosti profiq s.r.o. V úvodu se zaměřím na mé pracovní zařazení, stručný popis společnosti a na úkoly, které mi byly zadány. Následně popíši postup řešení těchto zadaných úkolů.

Během práce jsem se seznámil s řadou nových technologií, mimo jiné se streamovací platformou Wowza a platformou Dynepic určenou pro ochranu dat dětí v USA. Hlavním cílem byl vývoj mobilních aplikací pro obě tyto platformy, společně se sepsáním článků v angličtině popisující průběh vývoje. Součástí práce bylo rovněž penetrační testování platformy Dynepic.

V práci popisuji jednotlivé problémy, které při vývoji nastaly, a také způsob jejich řešení. Ke konci práce shrnu znalosti, které jsem získal v průběhu studia, a které mi byly užitečné při zpracování této práce. V závěru uvedu celkové zhodnocení praxe.

Klíčová slova: Odborná praxe, Android, Java, Testování, Zajištění kvality, SDK, Google Firebase, Wowza, Dynepic, GIT, Python, Bash, Penetrační testování, API, Kontinuální integrace, Docker

Abstract

In this thesis, I'm going to describe process of my professional practice in the company profiq s.r.o. In the introduction I'm focusing on my work deployment, assigned tasks and brief description of the company. Further I will describe the solution of my assigned tasks.

During my work, I've got in touch with a lot of new technologies. Among them is streaming platform Wowza and platform designed for security of personal information of children in USA. The main goal was development of mobile applications for both of these platforms together with writing articles in English describing process of development.

In thesis, I'm describing individual problems that appeared during the development and also a way of their solution. At the end of thesis I will describe knowledge I've gained and attained during my studies and which were useful during writing this thesis. I will finish this thesis by evaluation of professional practice.

Key Words: Professional practice, Android, Java, Testing, Quality Assurance, SDK, Google Firebase, Wowza, Dynepic, GIT, Python, Bash, Penetration testing, API, Continuous integration, Docker

Obsah

Seznam použitých zkratek a symbolů

SW	–	Software
API	–	Application Programming Interface
VoD	–	Video on Demand
COPPA	–	Children's Online Privacy Protection Act
CDN	–	Content Delivery Network
IPTV	–	Internet Protocol Television
SDK	–	Software Development Kit
FCM	–	Firebase Cloud Messaging
HTTP	–	Hypertext Transfer Protocol
IP	–	Internet Protocol
CI	–	Continuous Integration
XSS	–	Cross-Site Scripting
DNS	–	Domain Name System

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

V rámci této bakalářské práce popisuji průběh své odborné praxe ve společnosti profiq s.r.o. Byly mi svěřeny dva projekty ve vývoji mobilních aplikací pro zařízení s operačním systémem Android. První aplikace byla vytvořena pro společnost Wowza [1] zabývající se především streamováním živého videa. Druhou aplikaci jsem vyvíjel ve spolupráci se společností Dynepic [2], která se zabývá ochranou osobních dat dětí podle vydaného zákona COPPA v USA.

První věcí, kterou jsem musel projít, bylo naučit se a pochopit principy při vývoji mobilních aplikací. V obou projektech bylo potřeba pochopit, co daná společnost vyvíjí, a dále nastudovat potřebnou dokumentaci k poskytovaným SDK. Obě společnosti umožňují využít pro vývoj nových aplikací své SDK. Díky tomu bylo možné využívat přístupu ke vzdáleným serverům bez nutnosti implementovat své vlastní požadavky. Poté jsem začal pracovat na vývoji aplikací pro obě tyto společnosti. Po dokončení implementace požadovaných aplikací jsem aplikace otestoval a o postupu při vývoji sepsal o obou aplikacích články v angličtině, které jsou dostupné na webu společnosti profiq.

Součástí práce pro společnost Dynepic bylo rovněž penetrační testování jejich platformy. Při tomto úkolu jsem se naučil, jaká možná nebezpečí mohou nastat při vývoji mobilních aplikací. V rámci některých bezpečnostních opatření jsem napsal testy, které se následně spouštěly automaticky při nahrání nových změn zdrojového kódu do hlavního repozitáře projektu.

2 Popis odborného zaměření firmy a popis pracovního zařazení studenta

Tato kapitola popisuje zaměření firmy profiq a pracovní zařazení studenta v rámci běžné pracovní náplně.

2.1 O společnosti profiq s.r.o.

Společnost se zaměřuje na poskytování IT služeb v oblasti zajištění kvality, testování SW (ať už manuálního či automatizovaného) a ve vývoji aplikací pro mobilní zařízení, počítače a webové prohlížeče. Své služby zaměřuje na americký trh v Silicon Valley, který je centrem světového počítačového a technologického průmyslu. Firma zajišťuje služby v různých formách – od poskytování jednotlivých specialistů pro konkrétní úkoly, až po komplexní řešení kompletních softwarových systémů. Mezi přední zákazníky firmy patří některé americké firmy jako ForgeRock, Sencha, Frame, a také firmy působící v ČR – T-Mobile, Avast. [3]

2.2 Pracovní zařazení studenta

Do firmy jsem nastoupil několik měsíců před začátkem odborné praxe na pozici Software Engineer, kterou zde v době psaní této práce vykonávám na částečný úvazek. V rámci této pozice jsem byl zařazen do Technology Research týmu, kde mou hlavní pracovní náplní byl průzkum a analýza nových technologií, které se objevily na trhu. Firma si tímto získává pozornost námi zkoumaných firem a zároveň si rozšiřuje povědomí o technologických novinkách ve světě. Mezi technologie, které jsem měl v rámci své pracovní náplně na starost, patří zmiňované platformy společností Wowza a Dynepic. Hlavním úkolem bylo pro obě společnosti vyvinout aplikaci pro mobilní zařízení s operačním systémem Android. Na toto téma také zaměřuji tuto práci. O vývoji jsem následně sepsal články v angličtině.

3 Seznam úkolů a vyjádření jejich časové náročnosti

V této části popíši jednotlivé úkoly, které mi byly při práci zadány. Jednalo se o dva různé projekty, kde hlavním společným tématem byl vývoj mobilní aplikace pro mobilní telefony s operačním systémem Android.

3.1 Seznámení se streamovací platformou společnosti Wowza

Prvním úkolem byl vývoj části mobilní aplikace, která slouží jako přehrávač vysílání živého videa.

3.1.1 Seznam úkolů

V první části bylo mým úkolem seznámit se s platformou, na které Wowza postavila svůj byznys. V této fázi šlo hlavně o teoretické pochopení toho, jak funguje streamování videí, jaké jsou druhy streamování a na jakých technologiích funguje architektura streamování. Ve druhé části jsem si nastudoval dokumentaci k SDK, kterou Wowza nabízí pro implementaci jejich řešení v mobilních aplikacích. Mým úkolem bylo implementovat přehrávač živého vysílání ve stávající mobilní aplikaci, která slouží k přehrávání videí. Učil jsem se proto od základů programovat mobilní aplikace pro Android v programovacím jazyce Java. To zahrnovalo veškeré zjišťování informací o tom, jak vývoj probíhá, na co si dát pozor a následně prakticky aplikovat tyto poznatky. Při tomto úkolu jsem vycházel z dokumentace dostupné na webových stránkách Android [4] a rovněž z dokumentace k využitému SDK společnosti Wowza. [5] Součástí implementace přehrávače byla také konfigurace webové infrastruktury, jelikož bylo potřeba nastavit parametry připojení aplikace ke streamovacímu serveru. Ve třetí části probíhalo manuální testování aplikace. Dále bylo mým úkolem sepsat článek o mé zkušenosti s touto společností, s jejich SDK a celkově vývojem aplikace.

3.1.2 Vyjádření časové náročnosti úkolů

Pro první část úkolu, kde jsem se seznamoval s technologií, bylo vyhrazeno 5 pracovních dnů. Druhá a třetí část úkolu neměla pěvně daný rozsah, nicméně jsem na tomto úkolu strávil 15 pracovních dnů. Celkově jsem tedy strávil nad tímto projektem 20 dní.

3.2 Seznámení s platformou společnosti Dynepic

Mým druhým úkolem byl kompletní vývoj mobilní aplikace s použitím SDK společnosti Dynepic. Součástí úkolu bylo následné testování aplikace, včetně penetračního testování API rozhraní.

3.2.1 Seznam úkolů

Tento projekt byl rovněž rozdělen do několika částí. V první části jsem se zaměřil na průzkum webu společnosti Dynepic. Cílem bylo vyhledat dostupné nástroje pro vývojáře a pochopit, co může vývojářům Dynepic nabídnout. Po tomto kroku jsem měl za úkol vyvinout mobilní aplikaci, která bude sloužit jako sociální síť a zároveň jako aplikace pro ukládání poznámek. Rovněž zde bylo zapotřebí nakonfigurovat webovou infrastrukturu, pomocí které aplikace komunikovala se svými servery. Jelikož Dynepic neměl své SDK ještě plně hotové, bylo potřeba otestovat jeho funkčnost. Díky tomu jsem se zapojil do vývoje samotného SDK, což zahrnovalo komunikaci s developery z USA. Při vývoji jsem vycházel z dokumentace dostupné na webových stránkách společnosti Dynepic. [6] Po této části jsem o vývoji aplikace napsal článek v angličtině. Následně bylo mým úkolem aplikaci otestovat. Šlo o manuální testování funkčnosti všech funkcí, které jsem v aplikaci implementoval. Dále jsem měl za úkol provést penetrační testy API rozhraní, pomocí kterého aplikace komunikují s databázi. K tomu bylo zapotřebí využít zařízení s operačním systémem Android i iOS, jelikož API pro oba systémy bylo rozdílné. Pro vytváření a odesílání HTTP požadavků jsem využil program Postman. Pro čtení příchozích a odchozích požadavků ze zařízení jsem využil programu Charles.

3.2.2 Vyjádření časové náročnosti úkolů

První části úkolu bylo vyhrazeno 5 dnů. Druhé a třetí části jsem věnoval 15 pracovních dnů. Na čtvrtou část bylo vyhrazeno více než pracovních 10 dnů. Nad tímto projektem jsem celkem pracoval více než 30 pracovních dnů.

4 Zvolený postup řešení zadaných úkolů

V této kapitole se zaměřím na popis řešení jednotlivých částí zadaných úkolů. Nejprve popíši základní teoretické poznatky o programování Android aplikací, jelikož jsou součástí obou projektů a budu se na ně odkazovat. Dále se zaměřím na jednotlivé specifika při vývoji mobilních aplikací pro společnost Wowza a Dynepic. Posledním tématem bude samostatná kapitola o penetračním testování navazující na vývoj aplikace pro společnost Dynepic.

4.1 Programování Android aplikací

Na úvod bych se zmínil o koncepcích, které jsou společné pro oba projekty, které ve své práci popisuji. Při vývoji jsem využil vývojového prostředí Android Studio, které se běžně využívá při vývoji Android aplikací.

4.1.1 Vytvoření nového projektu

Prvním krokem při vytvoření nové aplikace je vytvoření nového projektu. To sebou nese rozhodnutí určit, pro jaké typy zařízení a verze SDK bude aplikace dostupná. Tím, že zajistíme dostupnost pro vyšší verze SDK, snížíme počet zařízení, na kterých půjde aplikaci spustit. Na druhou stranu, vyšší verze SDK nám umožní využívat modernější nástroje při vývoji. Stejný princip platí i obráceně. Tento parametr je možné sice později v Android manifestu snadno změnit, pokud se však začne vytvářet aplikace pro novější verzi a později padne rozhodnutí podporovat i starší verze, bude nutno přidat knihovny kompatibility. Opačně, pokud později padne rozhodnutí ukončit podporu starších verzí, bude užitečné funkce, které byly předtím realizovány pomocí knihoven kompatibility, upravit tak, aby využívaly přímo funkce zabudované v nové verzi. [7] Musíme tedy zvážit, které funkce potřebujeme, a které jsme schopni obětovat pro vyšší dostupnost aplikace.

4.1.2 Základní rozvržení aplikace

Základem aplikace jsou aktivity a rozložení. Aktivity popisují chování v pozadí aplikace, rozložení nám pak udává to, co se uživateli zobrazí na displeji. Tyto dva prvky jsou spolu vzájemně provázány a tím zajišťují celkový chod aplikace. Hlavním prvkem aktivit je metoda onCreate(), která se zavolá pokaždé, když se vytvoří nová aktivita. V zásadě se dá říci, že se jedná o jakýsi konstruktor aktivity.

4.1.3 Manifest aplikace

Při vývoji aplikace je potřeba zajistit, aby aplikace měla dostatečná práva vykonávat určité činnosti – např. připojit se k internetu, využívat fotoaparát apod. Tyto práva určíme v manifestu.

Rovněž zde můžeme nastavit název aplikace, ikonu, nebo například jednotný styl aplikace. Dalším důležitou součástí manifestu jsou aktivity, kterým zde nastavujeme parametry jako název, rodič aktivity nebo odlišný styl od jednotného stylu.

4.1.4 Soubory gradle a použití závislostí

K dalšímu nastavení aplikace máme k dispozici dva build.gradle soubory pro:

- Projekt – zde přidáváme externí uložiště, ze kterých můžeme čerpat nové závislosti. Příkladem uložišť je např. Google, JCenter nebo Maven.
- Modul – slouží jako hlavní soubor, ve kterém definujeme nastavení aplikace spolu se potřebnými závislostmi. Závislosti jsou knihovny, které rozšiřují funkcionalitu aplikace. V tomto souboru je možné nastavit verzi SDK, kterou aplikace využívá, nebo také verzi Javy, ve které se výsledný kód kompiluje.

4.1.5 Pomocné nástroje

Pro usnadnění práce při vývoji aplikace mohou být použity již vytvořené nástroje jako jsou seznamy položek a vysouvací menu. V rámci této práce bylo využito třídy DrawerLayout, pomocí které bylo vytvořeno vysouvací menu. Dále pro vytvoření seznamu poznámek, přátel a videí bylo použito třídy BaseAdapter, která umožňuje jednoduše vytvořit seznam položek v rozložení.

4.2 Specifika při vývoji aplikace pro společnost Wowza

V této části popíši řešení úkolu při vývoji mobilní aplikace za využití SDK společnosti Wowza. Prvním bodem bude nástin toho, čím se Wowza zabývá. Jelikož má streamování videí v pozadí množství teorie, popíši zde i teoretické poznatky, které jsem se při práci na úkolu dověděl. Poté se konkrétně zaměřím na praktické řešení úkolu, tedy na implementaci přehrávače v mobilní aplikaci.

4.2.1 Seznámení s produktem

Wowza je společnost, která vznikla v roce 2005. Je jednou z největších hráčů na poli streamování videa. Nabízí kompletní infrastrukturu pro streamování videí. Hlavním produktem je streamování živého videa, nicméně v nabídce je také streamování videí na vyžádání a streamování hlasu. Tuto technologii využívají společnosti, které nabízí své služby koncovým uživatelům – typicky se jedná o společnosti jako Facebook, Vimeo, Sony, které jsou schopny pomocí této architektury přehrávat svůj obsah uživatelům po celém světě.

Jednou z možností, jak Wowzu zprovoznit, je její nasazení na vlastním serveru. Wowza tento produkt pojmenovala jako Wowza Streaming Engine™. Toto řešení je multiplatformní a můžeme ho nasadit na počítačích s operačními systémy Windows, Linux i Mac OS. Druhou možností je využití služby Wowza Streaming Cloud™. Tato služba nabízí připravený server v

cloudu, na který se stačí připojit a nakonfigurovat ho pro své potřeby. Obě řešení podporují přístup pomocí REST API k nastavování streamů a správu serveru pomocí HTTP požadavků, JavaScript API k rozšíření přehrávače a Java API k rozšíření serveru svými vlastními třídami.

Streamování obsahu můžeme rozdělit do tří částí – samotné zachycení obsahu, příprava a doručení koncovému uživateli. U zachycení obsahu obecně řešíme použité technické prostředky, podporované kodeky a enkodéry. V druhé – přípravné části řešíme protokoly, pomocí kterých budeme obsah doručovat a dále rozlišení a kvalitu obsahu. Třetí část se zaměřuje na podporované přehrávače a zařízení, a také na způsob doručení obsahu – toto řešíme většinou pomocí CDN. Streamovaná videa mohou být přehrávány na několika různých podporovaných přehrávačích, pro příklad uvedu známější přehrávače jako RealPlayer, VLC, Adobe Flash a nechybí ani přehrávače pro Android a iOS. Wowza má rovněž vytvořený svůj vlastní přehrávač Wowza Player. Mezi podporovaná zařízení patří mobilní telefony a tablety, počítače, herní konzole a IPTV.

4.2.2 Teorie streamování

Streamování je metoda doručování videa nebo audia ke koncovým uživatelům na jejich zařízení – mobilní telefony, počítače a televize. Obsah je typicky doručován v reálném čase. Nemusí se však jednat jen o živý přenos obsahu – streamování obecně zahrnuje i přenos videa uloženého na vzdáleném serveru. Typickým příkladem je Youtube, kde jsou videa uložena a uživatel si je může přehrát na vyžádání (VoD). Streamování lze rozdělit do dvou kategorií:

- **Video na vyžádání (Video on Demand, VoD)** - Obsah je uložen na vzdáleném serveru, ze kterého si uživatel může vybrat, které video si přehraje. Toto video se poté postupně začne stahovat do cache prohlížeče a přehraje se uživateli. Po přehrání se cache vymaže. Uživatel si může rovněž video například pozastavit nebo posunout dopředu nebo dozadu. [8]
- **Živý přenos (Live streaming)** - Obsah je současně nahráván a přenášen v reálném čase ke koncovému uživateli. Jsou zde daleko větší nároky na servery, které se přenosu účastní. Obsah v tomto případě nemusí být nutně uložen na serveru. Aby se obsah spolehlivě doručil k uživateli, využívá se CDN. CDN je síť serverů, které jsou geograficky rozmístěny po celém světě. Obsah je ze zdroje rozšířen do všech serverů CDN. Díky tomu je možné obsah stahovat z geograficky nejbližšího CDN, čímž se zaručí spolehlivější a rychlejší přenos dat.

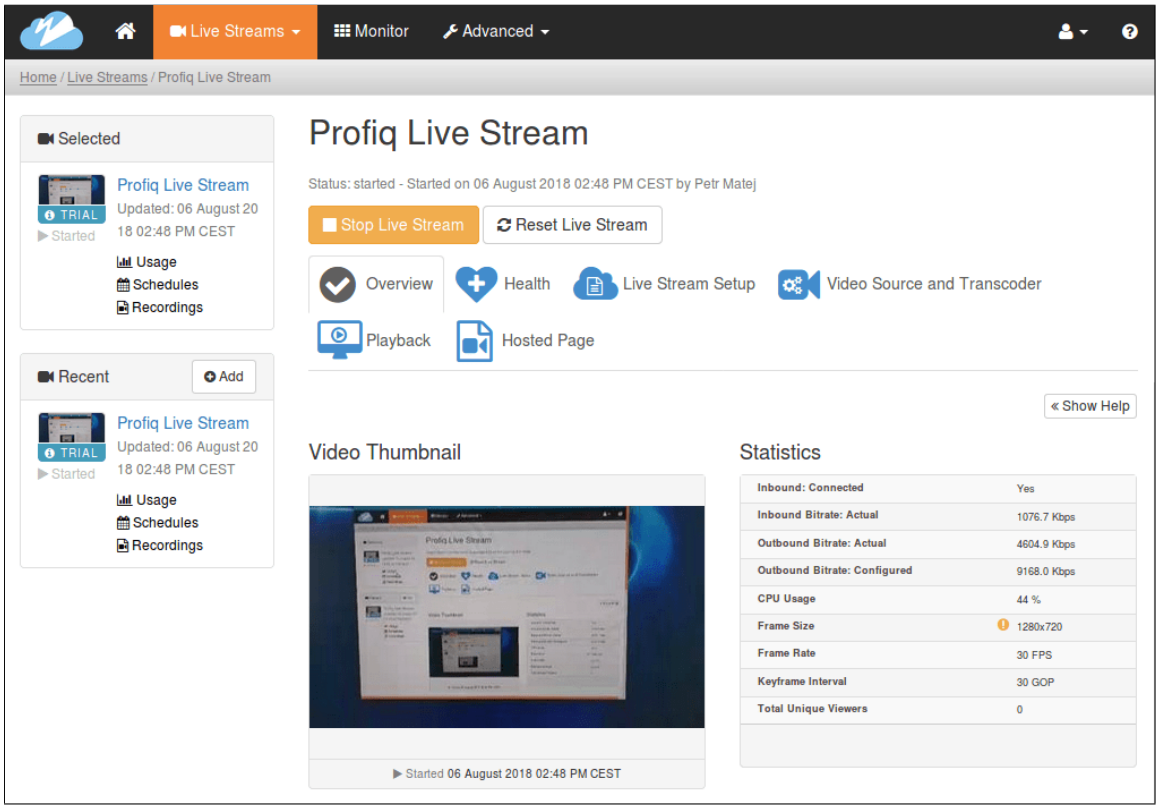
4.2.3 Konfigurace webové infrastruktury

Prvním krokem, který byl potřeba zajistit pro fungování streamování, bylo nastavení přenosu přes webové rozhraní Wowza Cloud. Nastavení mimo jiné zahrnuje:

- Výběr nejbližšího serveru pro co nejnižší latenci přenosu

- Výběr protokolu, přes který se video bude přenášet
- Nastavení rozlišení videa
- Možnost přidat autentizaci uživatele

Po nastavení byly vygenerovány parametry, které byly použity v samotné aplikaci. Na obrázku 1 můžeme vidět webové rozhraní Wowza Cloud s nastaveným a spuštěným přenosem videa.



Obrázek 1: Rozhraní Wowza Cloud

4.2.4 Implementace přehrávače živého videa

Dalším krokem bylo implementovat Wowza přehrávač ve stávající aplikaci. Při práci na tomto úkolu bylo využito dokumentace, kterou Wowza poskytuje ke svému SDK. Nejprve bylo nutné implementovat novou závislost Wowza SDK a přidat uložiště Maven, na kterém bylo SDK uložené. Po úspěšné kompilaci souboru gradle bylo možné začít využívat funkce SDK. Stávající aplikace byla zaměřena na verzi SDK 26, ale podporovala minimální verzi SDK 19, takže byla kompatibilní s podporovanou verzí Wowza SDK. Aplikace také vyžadovala povolení přístupu k internetu, které jsem nastavil v manifestu aplikace.

Byla vytvořena nová aktivita a nové rozložení. V rozložení byl následně implementován přehrávač, který byl obsažen v SDK. Dále stačilo vložit již vytvořené horní menu a přidat

tlačítko pro spuštění přehrávání s textem informujícím o stavu připojení. Logo firmy také nesmělo chybět. V samotné aktivitě pak bylo potřeba nastavit parametry připojení k Wowza Cloud. Ty jsem obdržel v přechozím kroku při nastavení přenosu ve webovém prostředí. Díky tomu se aplikace autorizovala pro připojení k Wowza Cloud, přes který se streamuje video. Ukázku části implementace požadovaného nastavení je možné vidět ve výpisu 1.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    WowzaGoCoder.init(getApplicationContext(),"GOSK-5E45-010F-94EE-E873-1680");
    mySettings = new WOWZPlayerConfig();
    mySettings.setIsPlayback(true);
    mySettings.setHostAddress("f5b03e.entrypoint.cloud.wowza.com");
    mySettings.setApplicationName("app-05ee");
    mySettings.setStreamName("89cddf9b");
    mySettings.setPortNumber(1935);
}
```

Výpis 1: Ukázka nastavení parametrů připojení k Wowza Cloud

Dále bylo potřeba zajistit chování tlačítka pro přehrávání a zastavení videa. Tlačítko se mělo aktivovat při kliknutí kdekoliv do plochy přehrávače. Při úspěšném spojení s Wowza Cloud tlačítko zmizí, při chybě se tlačítko zobrazí spolu s textem informujícím uživatele o problému. SDK obsahuje metody pro zastavení a zahájení přenosu videa, stačilo tedy jen najít související prvek přehrávače v rozložení a na něm zavolat dané metody. Posledním krokem v implementaci bylo zakomponovat aktivitu ve stávající aplikaci. Aplikace obsahuje seznam videí, který je implementován pomocí rozšíření třídy BaseAdapter. Stačilo tedy přidat novou položku do seznamu a odkázat ji na nově vytvořenou aktivitu.

4.2.5 Testování aplikace

Posledním úkolem bylo otestovat funkčnost aplikace. V této fázi šlo pouze o manuální testování a vyzkoušení všech funkcí, zda fungují po implementaci přehrávače správně. Zahrnovalo to také testování přenosu živého videa, kdy bylo použito dvou mobilních telefonů. Jeden sloužil pro nahrávání vysílání, druhý pro příjem tohoto přenosu. Jelikož byla aplikace při vývoji průběžně testována, nenarazilo se při finálním testování na závažnější problémy.

4.3 Specifika při vývoji aplikace pro společnost Dynepic

Tato sekce se zaměří na popis specifik při vývoji mobilní aplikace pro společnost Dynepic s využitím jejich SDK. Je zde opět uveden popis společnosti a to, čím se zabývá. Dále je text

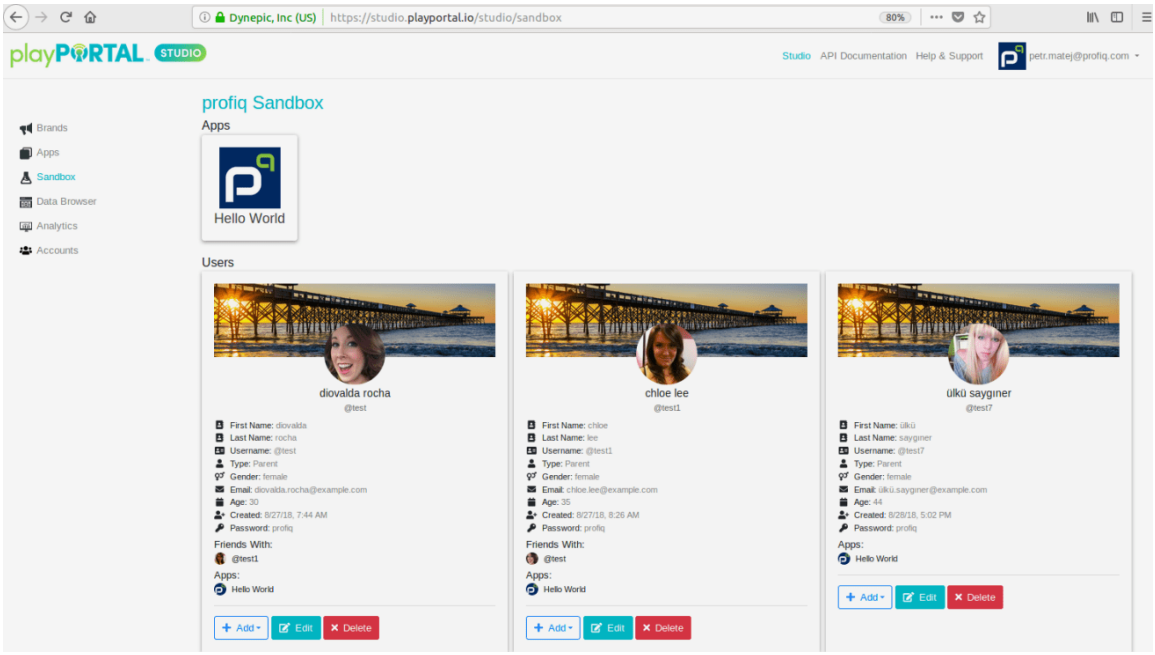
zaměřen na praktickou část vývoje mobilní aplikace. Na tuto kapitolu je navázána následující kapitola o penetračním testování.

4.3.1 Seznámení s produktem

Společnost Dynepic se zabývá problematikou spojenou s ochranou uživatelských dat v USA podle vydaného zákonu COPPA o ochraně osobních dat dětí v online světě. Dynepic pomocí svého řešení umožňuje bezpečné ukládání dat při vývoji mobilních aplikací. Vývojáři stačí implementovat poskytované SDK ve své aplikaci, které mu umožní používat cloudovou databázi k bezpečnému ukládání dat. Tímto se vývojáři usnadní práce - bude automaticky splňovat požadavky zákonu COPPA a nemusí se starat o zabezpečení dat na své straně. Platforma je navržena tak, aby poskytovala rodičům přehled o aktivitách jeho dítěte.

4.3.2 Nastavení webové infrastruktury

Nastavení zahrnuje registraci ve webové aplikaci playPortal [9] a přidání nové mobilní aplikace do seznamu aplikací. V detailech mobilní aplikace pak najdeme potřebné údaje, které jsou použity při následném vývoji. Pro testovací účely je možné v záložce Sandbox přidat testovací uživatele, viz. obrázek 2. Dále zde nalezneme například statistiky o používání aplikace.

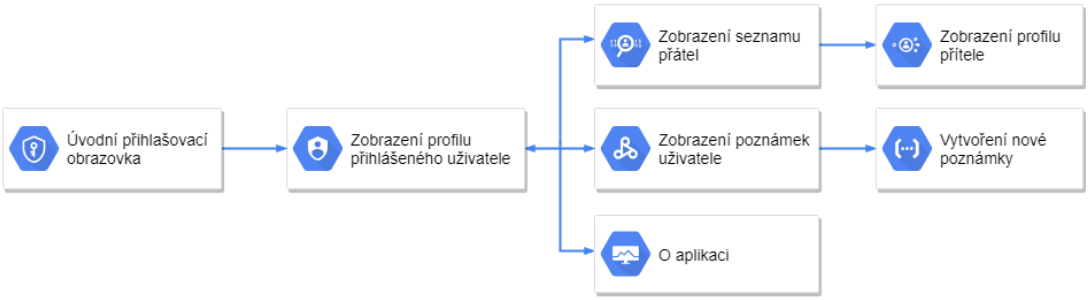


Obrázek 2: Rozhraní Dynepic PlayPORTAL

4.3.3 Návrh aplikace

Návrh a vývoj aplikace byl plně v autorově režii, ale hlavní funkcionalitu byla konzultována s vedoucím. Nejprve bylo zapotřebí navrhnout, jaká bude hlavní funkce aplikace. Jelikož byl

Dynepic ve fázi vývoje SDK pro Android a SDK nebylo plně dokončené, hlavním úkolem bylo otestovat jeho funkčnost v rámci této aplikace. Jako vhodný nápad bylo navrhnutí aplikace, kde si uživatel pod svým profilem bude moci ukládat své poznámky. Zároveň aplikace slouží jako sociální síť a umožňuje odesílat push notifikace. Byl tedy navržen tok aplikace, který je možno vidět na obrázku 3.



Obrázek 3: Tok aplikace

V každé aktivitě bylo kromě přihlášení potřeba implementovat vysouvací menu. To je možné implementovat pomocí DrawerLayout API dostupné v Support Library. Pro ulehčení práce s odkazy na další aktivity byly ostatní aktivity implementovány jako fragmenty. Ty stačilo vyvolat při stisknutí položky ve vysouvacím menu.

4.3.4 Vývoj aplikace

Jelikož poskytnuté SDK bylo kompatibilní pouze s verzí Android SDK pro API 27 a vyšší, musel být projekt navržen také pro tuto verzi. Tím se značně omezil okruh uživatelů, kteří aplikaci budou moci využívat. Podle Android Studia bylo zaměřeno pouze 1% všech Android zařízení.

Hlavním prvkem SDK je třída PPManager, která obsahuje jeho metody a chování. Tato třída se chová jako singleton, a proto lze její instanci uložit zavoláním metody getInstance(). Instance této třídy byla využita ve většině aktivitách, kde bylo zapotřebí komunikovat s databází.

Jako první byla vytvořena aktivita LoginActivity. V té bylo zapotřebí implementovat základní nastavení aplikace jako například parametry, které byly vygenerovány na webu. Díky tomu se aplikace mohla autorizovat a komunikovat se servery a databází. Dále byla v této aktivitě implementována logika přihlášení uživatele. Bylo zde rovněž nutné rozlišit, zdali se už uživatel v minulosti do aplikace přihlásil. Pokud se uživatel v minulosti přihlásil, stačilo ho rovnou přesměrovat k další aktivitě – k aktivitě, ve které bylo implementováno vysouvací menu. V opačném případě byl uživatel vyzván k vyplnění přihlašovacích údajů.

V aplikaci byla dále vytvořena již zmíněná třída pro implementaci vysouvacího menu. Ta se chová jako rodič všech aktivit, které jsou definovány jako fragmenty. Všechny třídy v aplikaci kromě přihlášení byly definovány jako fragmenty. Tím bylo zajištěno, že vysouvací menu bude

dostupné ve všech definovaných aktivitách. Výchozí zobrazovanou aktivitou na pozadí vysouvacího menu je profil uživatele. Ten se zobrazí jako následující aktivita po úspěšném přihlášení.

Profil uživatele obsahuje jeho základní osobní údaje a profilovou a úvodní fotku. Tyto údaje jsou uloženy ve vzdálené databázi na serverech společnosti Dynepic. Ke čtení dat z databáze byla použita metodu read, která jako argumenty funkce přijímá identifikaci uživatele, klíč, pod kterým jsou data uložena a lambda funkci, ve které můžeme pracovat s přečtenými daty. Data, se kterými zde pracujeme, jsou ve formátu JSON. Pak jen stačilo nastavit přečtená data v souvisejících položkách rozložení. Ukázku kódu pro čtení dat z databáze je možné vidět ve výpisu 2.

```
public void readData() {
    manager.data().read(userData.myData(), key, (JsonObject data, String e) ->
        {
        if (e == null) {
            Log.d("Read bucketName:", userData.myData() + " key:" + key + "
                value:" + (data != null ? data.toString() : null));

            int myValue = 0;
            if ((data != null ? data.get("prop") : null) != null) {
                myValue = Integer.parseInt(data.get("prop").toString());
            }
            text.setText(String.valueOf(myValue));
        } else {
            Log.e("Data read error:", e);
        }
    });
```

Výpis 2: Ukázka kódu pro čtení z databáze

Aktivity pro zobrazení seznamu přátel a poznámek byly implementovány pomocí nástroje BaseAdapter. Ten zajišťuje chování jednotlivých položek seznamu, který dostane jako parametr při vytváření jeho instance. Pro jednotlivé položky bylo vytvořeno rozložení, které určovalo jejich vzhled. V případě seznamu přátel položka obsahovala fotku přítele a jeho jméno, v případě seznamu poznámek položka obsahovala ikonu, popis a menu možností, ze kterých si uživatel mohl vybrat například smazání poznámky. Ukázku kódu pro smazání poznámky je možné vidět ve výpisu 3.

```
Button button = convertView.findViewById(R.id.note_menu_button);
button.setOnClickListener(view -> {
    pm = new PopupMenu(mContext, view);
    pm.getMenu().add(1, 1, 1, "Delete");
    pm.setOnMenuItemClickListener(item -> {
```

```
        manager.data().read(userData.myData(), key, (JsonObject data, String e)
             -> {
            if (e == null) {
                JsonArray ja = data != null ? data.getAsJsonArray("notes") :
                    null;
                ArrayList<NoteObject> notes = new Gson().fromJson(ja, new
                    TypeToken<ArrayList<NoteObject>>() {
                }.getType());
                ArrayList<NoteObject> notesFiltered = new ArrayList<>();
                if (notes != null) {
                    for (NoteObject note : notes) {
                        if (note.getItemID() != selectedNote.getItemID()) {
                            notesFiltered.add(note);
                        }
                    }
                }
                JsonObject jo = new JsonObject();
                JsonElement element = new Gson().toJsonTree(notesFiltered, new
                    TypeToken<ArrayList<NoteObject>>() {
                }.getType());
                jo.add("notes", element.getAsJsonArray());
                manager.data().write(userData.myData(), key, jo, (JsonObject
                    data2, String error2) -> {
                    if (error2 == null) {
                        Toast.makeText(mContext, "Note deleted", Toast.
                            LENGTH_SHORT).show();
                        fragmentTransaction.detach(parentFragment);
                        fragmentTransaction.attach(parentFragment);
                        fragmentTransaction.commit();
                    }
                });
            }
        });
        return true;
    });
    pm.show();
});
```

Výpis 3: Odstranění poznámky ze seznamu položek a databáze

Pro vytvoření nové poznámky bylo implementováno vyskakovací okno, které zabírá pouze část obrazovky. V případě, že uživatel zvolí poznámku jako veřejnou, bude odeslána push notifikace jeho přátelům. Metoda pro odesílání push notifikací je součástí Dynepic SDK. Nicméně odesílání push notifikací s sebou nese dodatečné nastavení aplikace. Na stránkách Google Firebase [10] bylo potřeba aplikaci zaregistrovat a vygenerovat FCM klíč. Ten se použil v nastavení aplikace přímo ve webovém prostředí Dynepic. Dále bylo potřeba stáhnout a zahrnout soubor google-services.json v samotné aplikaci. Posledním krokem k aktivaci notifikací bylo přidání závislostí a zavolání metody enablePushNotifications() v hlavní aktivitě.

4.3.5 Testování aplikace

Prvním krokem bylo manuální otestování výsledné aplikace. Šlo opět o to vyzkoušet, zda se všechny aktivity chovají správně a podle požadavků. V případě, že nějaké chování vykazovalo chybové známky, bylo potřeba najít problém a odstranit ho. Při testování většího softwaru se standardně využívá nástrojů pro evidenci chyb jako je například JIRA, ale jelikož šlo v tomto případě o menší projekt, vystačil jsem si s Google Spreadsheet dokumentem.

4.4 Penetrační testování platformy Dynepic pomocí mobilních aplikací

Posledním úkolem bylo testování zabezpečení platformy Dynepic, jejich webových stránek a jimi již vydaných mobilních aplikací. Konkrétně šlo hlavně o testování rozhraní API, pomocí kterého mezi sebou komunikovaly mobilní aplikace s databází. Dále bylo potřeba otestovat webovou stránku na možnosti útoku pomocí SQL injection a XSS. Dynepic vyvinul dvě své mobilní aplikace, které byly při testování použity – aplikace PlayPortal a WonderWorks. Mimo to bylo také využito aplikace, o které je psáno v předešlých kapitolách. Při testování bylo zjištěno, že Dynepic využívá celkem tři API rozhraní – pro Android, iOS a web.

4.4.1 O penetračním testování

Penetrační testování je způsob, jak simulovat metody, které útočník může použít k prolomení bezpečnostních kontrol a tím pádem k získání přístupu do systému oběti nebo organizace. [11] Je to tedy oblast testování, která má za úkol zjistit možné bezpečnostní trhliny v testovaném systému tak, aby bylo možné následně učinit bezpečnostní opatření k ochraně dat a zajištění správné funkcionality. [12] Jde o cílené použití nástrojů pro útok na informační systémy.

Jedním z dostupných nástrojů pro takové testování je distribuce Linuxu jménem Kali [13]. Ta byla využita při testování bezpečnosti, jelikož v základu obsahuje řadu předinstalovaných nástrojů k zjišťování informací o síti, cílových počítačích a serverech a k útokům na ně. Dalším dostupným řešením v rámci operačního systému založeného na distribuci Linux je Parrot Security.

4.4.2 Zachytávání komunikace z mobilních zařízení

Při komunikaci mobilního telefonu se servery se vytvářejí HTTP požadavky, které lze zachytit pomocí programu Charles. [14] Tohoto programu bylo využito při zachytávání komunikace mezi aplikacemi Dynepic a jejich servery. Při testování se využívalo pouze veřejně zjistitelných informací - k testování nebyly společností Dynepic poskytnuty žádné neveřejné informace. Při testování byly použity dva mobilní telefony – jeden s operačním systémem Android, druhý s iOS. Každý z těchto operačních systémů měl pro komunikaci své vlastní API rozhraní.

Nejprve bylo potřeba nastavit program Charles. Jednalo se hlavně o přidání patřičného certifikátu do mobilních telefonů, který umožňoval sledování zabezpečené komunikace. Zde se vyskytl problém se zachytáváním komunikace Android zařízení. Android od verze 7.0 zablokoval možnost zachytávat komunikaci i přes to, že je v telefonu nainstalován potřebný certifikát. Řešením bylo přidat do manifestu vytvořené aplikace několik řádků kódu, které povolily sledování komunikace. Zajímavostí je, že iOS bylo možné sledovat bez jakéhokoliv zásahu do aplikace, narozdíl od vyšší verze Android. Po úspěšné konfiguraci bylo možné zachytávat veškerou zabezpečenou komunikaci mezi vytvořenou aplikací a patřičným API rozhraním.

4.4.3 Testování API rozhraní

Následovalo zkoušení funkcí aplikací a sledování, na které koncové body API rozhraní aplikace vysílá své požadavky. V tomto kroku byl vytvořen seznam všech koncových bodů API, který byl použit k dalšímu testování. Zároveň bylo odhaleno několik bezpečnostních rizik spojených s únikem osobních informací uživatelů. Jelikož toto bylo odhaleno na již vydaných aplikacích, jednalo se o závažný problém. Po nahlášení tohoto problému došlo ihned k jeho opravě.

Další možnou hrozbou bylo nahrávání souborů do databáze pomocí jednoho z koncových bodů API. Pomocí jednoduché úpravy požadavku bylo možné do databáze nahrát jakýkoliv soubor, bez jakékoliv kontroly formátu na straně serveru. Tím se zde naskytla možnost nahrát do databáze škodlivý kód ve formě skriptu. Soubor stačilo pouze převést do formátu Base64, využít požadavek pro vložení profilové fotky, zaměnit zakódovanou fotku ve formátu Base64 za skript a požadavek odeslat. Výsledkem bylo, že při zobrazení profilu uživatele bylo v požadavku možné zachytit celý zdrojový kód vloženého skriptu. Zdrojový kód ale nebylo možné jakýmkoliv způsobem na straně serveru spustit, a tudíž nepředstavoval závažnou hrozbu.

Rovněž zde byly otestovány útoky hrubou silou (Brute-force) a útok odepření služby (Denial of Service). Na servery bylo vysláno velké množství požadavků za použití nástroje Burp Suite. Servery Dynepic byly korektně zabezpečeny, a tudíž se nepodařilo vyřadit servery z provozu ani získat přístup k účtu s jednoduchým heslem. Servery implementovaly omezení počtu požadavků z jedné IP adresy, a proto by útok hrubou silou nebyl efektivní. Jedním ze zjištěných zabezpečení bylo použití webového aplikačního firewallu.

Tabulka 1: Testované chybové kódy API odpovědí a jejich vysvětlení

Testovaná kombinace	Kód chyby	Výsledek
Správné údaje	200	Úspěšné přihlášení
Nesprávný email	4040	Uživatel s daným emailem nenalezen
Správný email, nesprávné heslo	4011	Nesprávné údaje
Email nevyplněn	4040	Uživatel s daným emailem nenalezen
Heslo nevyplněno	4011	Nesprávné údaje
Přihlášení bez parametrů	4040	Uživatel s daným emailem nenalezen

4.4.4 Automatizované testy

Dalším krokem bylo vytvoření automatizovaných testů pro kontrolu bezpečnosti serverů Dynepic. Pomocí nástroje NSlookup byly zjištěny IP adresy všech API rozhraní. Každé API bylo složeno z více serverů pro rozložení zátěže (DNS load balancing), a tedy obsahovalo více IP adres. Nástrojem Nmap byla poté provedena kontrola otevřených portů. Tento proces byl zautomatizován pomocí programovacího jazyka Python. Bylo při něm využito knihovny unittest, která slouží k vytváření testů a testovacích případů, a dále knihovny requests sloužící k odesílání HTTP požadavků. Ukázku kódu jednoho z testů můžete vidět ve výpisu 4. Hlavním účelem tohoto testu bylo zjistit, zda jsou otevřeny pouze povolené porty. Pokud by byl nějaký port otevřen navíc, nebo naopak nějaký chyběl, test by skončil s chybným výsledkem a upozornil tím vývojáře na možnou chybu.

Dále byly zautomatizovány testy pro kontrolu přihlášení přes webovou stránku Dynepic. Jednalo se o provádění různých kombinací platných a neplatných přihlašovacích údajů a kontrolu jejich odpovědí na jednotlivé požadavky. Šlo zde o to, jaké stavové kódy požadavek vrátil. V tabulce 1 je možné vidět jednotlivé testy společně s kódem odpovědi API rozhraní a jejich vysvětlení.

Tyto testy byly dále nahrány na GitLab [15] jako testy kontinuální integrace. Díky tomu se testy spouštěly vždy při nahrání nových změn v hlavním repozitáři projektu. Výhodou takového postupu je, že pokud by vývojář udělal při vývoji chybu a omylem například otevřel nějaký nezabezpečený port, test by skončil chybou. Tím pádem by se ihned vědělo, o jaký problém se jedná a bylo by jej možné ihned opravit. Zároveň by se chybný kód nenahrál do aktuální větve projektu na Git.

Pro nahrání automatizovaných testů do prostředí GitLab CI je nutné vytvořit soubor gitlab-ci.yml. V tomto skriptu definujeme podmínky, ve kterých bude test spuštěn v prostředí Docker. V tomto případě bylo jako obraz použito nejnovější verze Pythonu. Dále následují příkazy jednotlivých testů v jazyce Bash. Testy se následně spouští v samostatném Docker kontejneru. Docker byl již v rámci Gitlab CI pro zadaný úkol nakonfigurován, stačilo tedy pouze nahrát samotný test a spouštěcí skript kontinuální integrace.

```
class PortTest(unittest.TestCase):
    @staticmethod
    def get_nmap_port_list(host_address):
        console_report = os.popen('nmap {0}'.format(host_address))
        report = console_report.readlines()
        console_report.close()

        nmap_port_list = []
        for row in report:
            nmap_port_list.append(''.join(re.findall(r'\d+/tcp', row)))
            nmap_port_list.append(''.join(re.findall(r'\d+/udp', row)))
        return list(filter(None, nmap_port_list))

    @staticmethod
    def get_nslookup_ip_list(host_address):
        console_report = os.popen('nslookup {0}'.format(host_address))
        report = console_report.readlines()
        console_report.close()

        ip_list = []
        for row in range(2, len(report)):
            ip_list.append(''.join(re.findall(r'\d+.\d+.\d+.\d+', report[row])))
        return list(filter(None, ip_list))

    def test_not_opened_ports(self):
        for host in hosts:
            host_address = host['host']
            host_ports = host['open_ports']
            host_ip_addresses_list = self.get_nslookup_ip_list(host_address)

            for ip in host_ip_addresses_list:
                with self.subTest("Check {0}: {1}".format(host_address, ip)):
                    b = set(host_ports).difference(set(self.get_nmap_port_list(ip
                        )))
                    self.assertTrue(len(b) == 0, "Tested IP {0}. Not opened ports
                         are: {1}".format(ip, b))
```

Výpis 4: Unit test pro kontrolu portů

5 Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe

Při práci jsem uplatnil zkušenosti z většiny odborně zaměřených předmětů, které jsem v rámci studia absolvoval. Při psaní této práce jsem však nejvíce využil znalosti z předmětu Programovací jazyky I, ve kterém byly vyučovány základy programování v programovacím jazyce Java. Tento předmět mi byl hlavním základem při psaní obou Android aplikací. Rovněž předměty Skriptovací a programovací jazyky, Správa operačních systémů a Počítačové sítě byly pro mě užitečné. Jelikož bylo mým úkolem penetrační testování, využil jsem při něm znalosti z oblasti Linuxu, s programováním v jazyce Bash a Python a také znalosti počítačových sítí. Konkrétně při práci na automatizaci jsem využil znalosti jazyka Python a při nasazení automatizace do prostředí GitLab CI jsem využil znalost jazyka Bash.

Dále jsem aktivně využíval znalosti z předmětu Angličtina Ab/I-FEI - Ab/IV-FEI. Jelikož firma pracuje ve většině případů pro zahraniční klienty, veškerá komunikace se zákazníkem probíhala v angličtině. To zahrnovalo pravidelné meetingy se zákazníky a také čtení anglicky psané dokumentace. Články, které jsem následně o obou platformách sepsal, byly rovněž napsány anglicky.

6 Znalosti a dovednosti scházející studentovi v průběhu odborné praxe

V průběhu odborné praxe mi nejvíce scházely dovednosti s programováním Android aplikací. Ačkoliv jsem programovací jazyk Java ovládal díky předmětu Programovací jazyky I, psaní Android aplikací bylo pro mě něco nového. Dále mi chyběly jakékoliv základy penetračního testování. Při tom jsem pochopil, jaká bezpečnostní rizika mohou prakticky při vývoji aplikací nastat, a myslím si, že tyto znalosti by mohly být vyučovány již během bakalářského studia. Ačkoliv byly vyučovány teoretické poznatky základních útoků, chyběla jejich praktická aplikace a možnost se s nimi seznámit hlouběji.

Za zmínku také stojí to, že ve škole chyběla jakákoliv práce v prostředí Git, které umožňuje jednoduchou správu a verzování kódu, a kterého jsem denně používal při své práci. Konkrétně jsem využíval GitLab. Ten rovněž umožňuje kontinuální integraci, které jsem využíval při automatizaci testů. Rovněž jsem při studiu nezaznamenal jakoukoliv zmínku o prostředí Docker, kterého jsem využil v konečné fázi automatizovaného testování.

7 Dosažené výsledky v průběhu odborné praxe a jejich celkové zhodnocení

Touto prací jsem popsal svoji zkušenost z reálného pracovního prostředí. Během doby trváno praxe jsem pracoval na dvou různých projektech. Prvním z nich bylo implementování přehrávače živého vysílání pomocí SDK, které vytvořila společnost Wowza. U tohoto projektu jsem se poprvé seznámil s vývojem Android aplikací a také s tím, jak funguje streamování živého obsahu. Druhým projektem byl vývoj mobilní aplikace pomocí SDK společnosti Dynepic. Při tomto projektu jsem si vyzkoušel celý proces vývoje mobilní aplikace, od návrhu, přes implementaci jednotlivých požadovaných funkcí, po konečné testování funkčnosti. Tento projekt byl poté zakončen penetračním testováním, při kterém jsem využil mnou vyvinutou aplikaci. Vyzkoušel jsem si nabourat se do cizího systému a také jsem prakticky vyzkoušel některé typy útoků. Nakonec jsem sepsal testy pro kontrolu bezpečnosti, které byly následně zautomatizovány.

Díky oběma projektům jsem si prohloubil své znalosti s vývojem aplikací pro zařízení s operačním systémem Android. Důležitou zkušeností byla pro mě také práce s verzovacím systémem Git. Pochopil jsem, jaká bezpečnostní rizika můžou při vývoji aplikací nastat a také se setkal s automatizovanými testy v jazyce Python. Automatizované testy se spouštěly v rámci kontinuální integrace GitLabu na serverech, které potřebovaly pro spuštění testů vytvořit skript v jazyce Bash. Při tomto úkolu jsem se také setkal s prací v prostředí Docker, které slouží pro spouštění aplikací v izolovaných kontejnerech.

Rovněž jsem si zdokonalil písemnou i mluvenou formu anglického jazyka, díky pravidelným meetingům v angličtině, vydáním dvou článků a studiem anglicky psané dokumentace. Firma také poskytuje jednou týdně lekce angličtiny s rodilým mluvčím, kterých jsem se pravidelně účastnil. Ve firmě jsem po absolvování praxe nastoupil jako zaměstnanec na poloviční úvazek.

Literatura

[1] *Wowza Live Streaming Software - Best Cloud Platform for Live Media* [online]. Copyright ©2019 [cit. 15.04.2019]. Dostupné z: `https://www.wowza.com/`

[2] *Dynepic* [online]. Copyright ©2019 [cit. 15.04.2019]. Dostupné z: `https://www.dynepic.com/`

[3] About us » profiq. *profiq* [online]. Copyright ©2019 [cit. 15.04.2019]. Dostupné z: `https://www.profiq.com/about-us/`

[4] Developer Guides. *Android Developers* [online]. [cit. 15.04.2019]. Dostupné z: `https://developer.android.com/guide/`

[5] Wowza developer API and SDK docs. *Wowza Live Streaming Software - Best Cloud Platform for Live Media* [online]. Copyright ©2019 [cit. 15.04.2019]. Dostupné z: `https://www.wowza.com/docs/wowza-developer-apis-and-sdks/`

[6] Documentation - playPORTAL. *playPORTAL* [online]. [cit. 15.04.2019]. Dostupné z: `https://docs.playportal.io/`

[7] LACKO, Luboslav *Vývoj aplikací pro Android*. Brno: Computer Press, 2015, ISBN 9788025143476.

[8] How does streaming media work? - Explain that Stuff. *Explain that Stuff* [online]. Copyright ©2019 [cit. 15.04.2019]. Dostupné z: `https://www.explainthatstuff.com/streamingmedia.html`

[9] *playPORTAL Studio* [online]. Copyright ©2018 [cit. 15.04.2019]. Dostupné z: `https://studio.playportal.io/`

[10] *Firebase* [online]. [cit. 15.04.2019]. Dostupné z: `https://firebase.google.com/`

[11] KENNEDY, David, O'GORMAN, Jim, KEARNS, Devon a AHARONI, Mati *Metasploit: The Penetration Tester's Guide*. No Starch Press, 2011, ISBN 9781593272883.

[12] Penetration Testing Tutorial. *Tutorials Point* [online]. Copyright ©2019 [cit. 15.04.2019]. Dostupné z: `https://www.tutorialspoint.com/penetration_testing/`

[13] Kali Linux Official Documentation. *Offensive Security* [online]. Copyright ©2019 [cit. 15.04.2019]. Dostupné z: `https://www.kali.org/kali-linux-documentation/`

[14] Charles Web Debugging Proxy Documentation. *Charles Web Debugging Proxy* [online]. [cit. 15.04.2019]. Dostupné z: `https://www.charlesproxy.com/documentation/`

[15] GitLab Documentation. *GitLab* [online]. [cit. 15.04.2019]. Dostupné z: `https://docs.gitlab.com/`