Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Mobilní klient pro monitorování IT zdrojů

Plzeň 2022 Josef Bozděch

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Josef BOZDĚCH**
Osobní číslo:	**A18B0372P**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informační systémy**
Téma práce:	**Mobilní klient pro monitorování IT zdrojů**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Prostudujte dostupné nástroje na monitoring IT zdrojů a služeb.
2. Navrhněte mobilní aplikaci umožňující práci s vybraným monitorovacím systémem, včetně podpory notifikací.
3. Implementujte navrženou mobilní aplikaci.
4. Ověřte funkčnost mobilní aplikace na reálném systému.

Rozsah bakalářské práce: **doporuč. 30 s. původního textu**
Rozsah grafických prací: **dle potřeby**
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

Dodá vedoucí bakalářské práce.

Vedoucí bakalářské práce: **Ing. Luboš Matějka, Ph.D.**
Katedra informatiky a výpočetní techniky

Datum zadání bakalářské práce: **4. října 2021**
Termín odevzdání bakalářské práce: **5. května 2022**

L.S.

Doc. Ing. Miloš Železný, Ph.D.
děkan

Doc. Ing. Přemysl Brada, MSc., Ph.D.
vedoucí katedry

V Plzni dne 14. října 2021

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 3. května 2022

Josef Bozděch

Abstract

The goal of this thesis is the analysis and comparison of existing monitoring systems and mobile applications, which are compatible with these systems, and developing a new application, which will be compatible with a chosen monitoring system. React Native framework will be used to develop this application and will be compatible with iOS and Android devices. The thesis contains description of the development of this application and its backend, which will be developed for the application aswell.

Abstrakt

Cílem této práce je prozkoumání existujících monitorovacích systémů a mobilních aplikací, jež jsou s těmito systémy kompatibilní, jejich porovnání a navržení a implementování nové mobilní aplikace, která bude kompatibilní s jedním z vybraných monitovacích systémů. Aplikace bude implementována pro zařízení s operačními systémy iOS a Android s využitím multiplatformního frameworku React Native. Práce obsahuje popis vývoje této aplikace a backendu, který patří k této aplikaci a bude v této práci také vyvinut.

Obsah

1 Úvod

Počítačové sítě se neustále rozšiřují. Vzrůstá digitalizace společnosti a s ní i potřeba maximalizovat dostupnost síťových prvků. V případě, že přestane síťová služba fungovat, nastává problém, který je potřeba co nejdříve řešit. Aby bylo možné kontrolovat stavy těchto služeb, byly vyvinuty monitorovací nástroje.

Samotná existence monitorovacích nátrojů ale problémy v síti neřeší; je nutná reakce správce. Správce nemůže nicméně neustále kontrolovat stavy pomocí monitorovacího nástroje. Proto byly vynalezeny způsoby notifikování skrz e-mail a SMS. Výhodou je, že téměř každý člověk dnes vlastní mobilní telefon a může být těmito technologiemi notifikován. Tyto technologie jsou dnes nicméně již poměrně starou záležitostí a nedovolují správci na událost přímo reagovat; reakce musí proběhnout např. pomocí webového rozhraní, kterým standardně dnes monitorovací nástroje disponují. Webové rozhraní si je možné otevřít i v mobilním zařízení a správce tak může na notifikaci reagovat okamžitě kdekoliv.

V dnešní době vznikl nový trend, který kombinuje všechny potřeby dohromady: mobilní aplikace. Ty dokáží kombinovat notifikace s možností okamžitě na tyto notifikace reagovat; není již nutné překlikávat mezi více aplikacemi. Výhodou jsou dnes i existující multiplatformní frameworky, které umožňují vývojářům vyvíjet mobilní aplikace pro více platforem najednou, což ještě před několika lety nebylo možné.

Cílem této práce je prostudovat dostupné monitorovací nástroje a mobilní aplikace, které jsou pro tyto nástroje dostupné. Následně budou prostudovány možnosti vývoje mobilních aplikací. Nakonec bude navržena, implementována a otestována nová mobilní aplikace, která bude komunikovat s vybraným monitorovacím nástrojem.

2 Monitoring

IT monitoring se zabývá kontrolou funkčnosti IT infrastruktury. Monitoruje se funkčnost hardwaru, kam řadíme jednotlivé servery, lokální počítačové sítě nebo i celá datová centra, ale i software, například jednotlivé služby běžící na každém stroji. Do IT infrastruktury řadíme jak fyzické, tak i virtuální stroje (např. virtualizované servery). Do IT infrastruktury neřadíme lidské zdroje [86].

U jednotlivých částí IT infrastruktury zpravidla sledujeme několik metrik:

- Hardware - Sledujeme, zdali nedošlo k výpadku jednotlivých částí hardwaru; u datového centra to může být například výpadek disku.
- Operační systémy - Sledujeme využití, zatížení a výpadky.
- Síť - Sledujeme odezvu, využití a chyby v síti.
- Aplikace - Sledujeme výkon a dostupnost aplikací.

[31]

2.1 Vybraný monitorovací software

V následující kapitole budou popsány dostupné monitorovací nástroje, jež jsou doporučovány podle webu [95]. Důležité jsou také uživatelské recenze, neboť dávají hlubší náhled do kvality jednotlivých nástrojů. Recenzující uživatelé mohou zdůraznit výhody a nevýhody, které by na první pohled nemusely být vidět. Z webů, na kterých je možno tyto nástroje recenzovat, byl vybrán web Trustradius, neboť se na něm nachází velké množství recenzí na všechny níže popsané nástroje.

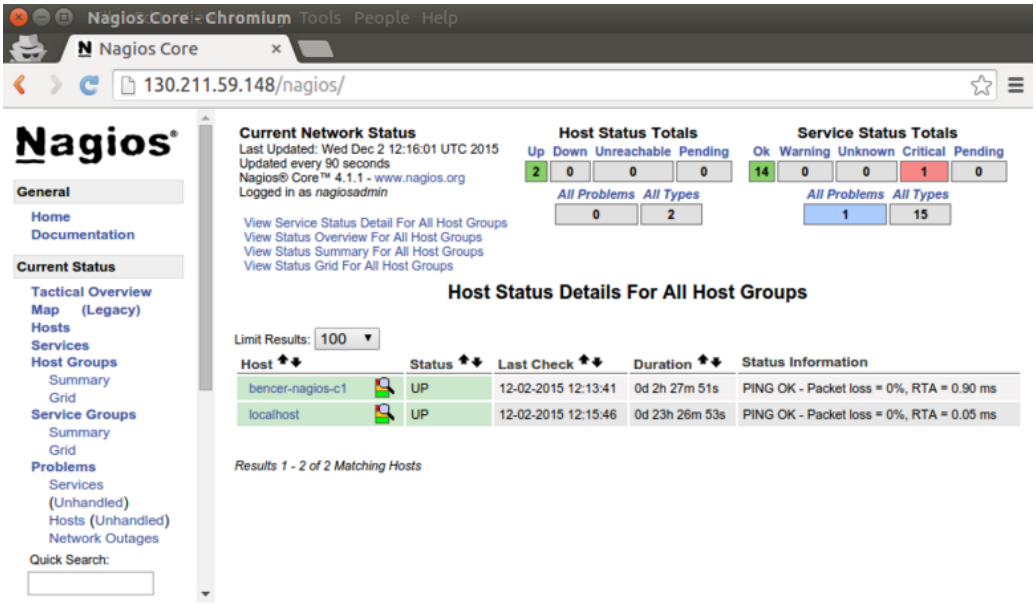
2.1.1 Nagios Core

`Nagios Core` [56] je monitorovací systém zaměřený zejména na jednoduché plánování událostí. Slouží také k upozorněním na chyby u monitorovaných prvků. Důsledkem toho, že je `Nagios` naprogramován v jazyce `C` [66], je vysoký výkon tohoto nástroje. Na obrázku 2.1 je ukázka toho, jak vypadá webové rozhraní tohoto nástroje.

Nagios dokáže monitorovat jednotlivé servery a služby na nich běžící, zobrazit o nich bližší informace, nastavit informace o odstávkách serverů, také má `REST API` [54], pomocí něhož lze s `Nagiosem` komunikovat a získávat z něj data. Za tímto projektem stojí velká komunita, která pro Nagios vytváří nejrůznější `addony` a `pluginy` (doplňkové moduly rozšiřující funkčnost softwaru), které rozšiřují funkčnost `Nagiosu`.

Podobně jako `Icinga2` [35] s sebou `Nagios` přináší webové rozhraní, které je ale zakomponováno v základním balíčku. Nemusí se tedy instalovat zvlášť. Ve webovém rozhraní se hůře orientuje, protože je stylizováno pouze pomocí základního `CSS` [66]. Plní nicméně stejné úkoly jako rozhraní pro `Icinga2`.

Instalace je svou náročností odpovídající nástroji `Icinga2`, nicméně návod na instalaci v dokumentaci je podrobnější, což je pro začátečníky v `Linuxových` [92] systémech velkou výhodou.



Obrázek 2.1: Webové rozhraní Nagios Core
Zdroj: https://blog.serverdensity.com/howto-install-nagios-in-30-minutes-and-jumpstart-your-monitoring/

V `Nagios` existují tzv. `EventHandlery`, které kontrolují stavy a jejich změny u serverů a monitorovaných služeb. Pomocí této funkcionality je možné získávat notifikace o změně stavu jednotlivých objektů [55].

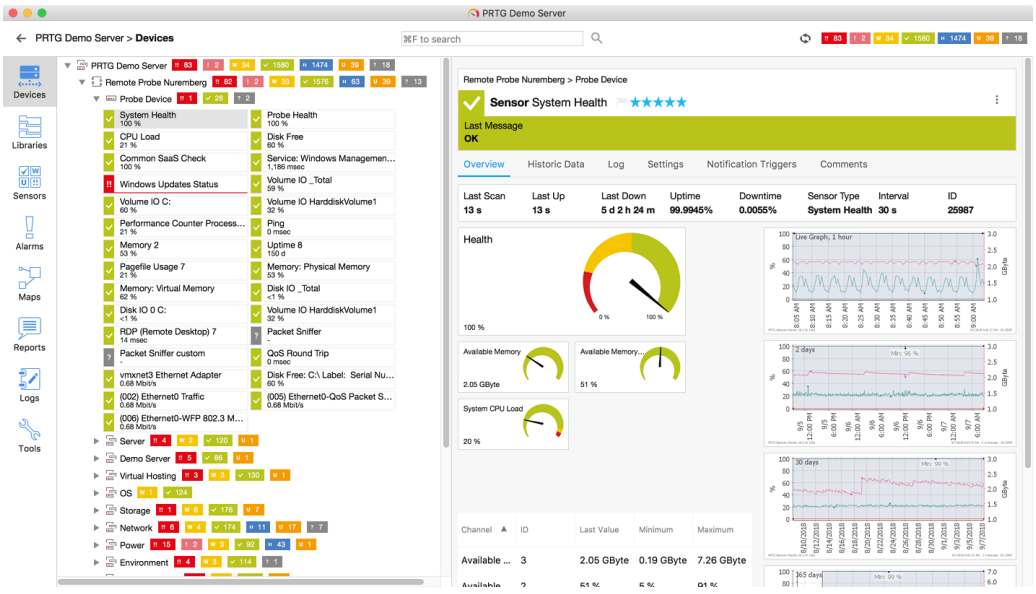
Uživatelé považují `Nagios Core` za kvalitní nástroj pro monitorování služeb, pokud hledáte produkt zdarma. Velkou výhodou je obrovské množství komunitou vytvořených `pluginů`, které zlepšují práci s tímto nástrojem. Za přednosti považují i notifikace. Nedostatečná je podle uživatelů oficiální pod-

pora, což může být kompenzováno přechodem na placenou verzi Nagios XI [58]. I přes podrobnější dokumentaci je počáteční instalace pro některé uživatele obtížná, chybí možnost konfigurace tohoto nástroje skrz uživatelské rozhraní [57].

2.1.2 PRTG Network monitor

`PRTG Network monitor` [67] (dále jen `PRTG`) je monitorovací nástroj, který je dostupný na platformě `Windows` [92]. Jeho používání je zcela zdarma do 100 sledovaných senzorů, pokud je jich potřeba sledovat více, existují placené balíčky, přičemž nejlevnější začíná na hodnotě 500 USD a můžete používat až 500 senzorů. Instalace je velice jednoduchá a rychlá. Stejně jako ostatní monitorovací nástroje, i `PRTG` disponuje webovým rozhraním, které je ale mnohem propracovanější a užitečnější, než u ostatních nástrojů. Splňuje všechny potřeby správce a není potřeba instalovat další rozšíření, která jsou potřeba u předchozích monitorovacích nástrojů. Ukázka programu je na obrázku 2.2

Výhodou `PRTG` je auto-discovery systém, který po nastavení sítě a masky automaticky přidá prvky v dané síti do své databáze a začne je monitorovat [95].



Obrázek 2.2: Webové rozhraní PRTG Desktop
Zdroj: https://blog.paessler.com/prtg-desktop

`PRTG` má svoji vlastní mobilní aplikaci pro `Android` i `iOS` [2]. V `PRTG` lze nastavit několik druhů událostí. V případě, že některá z nich nastane,

může být uživateli zaslána notifikace několika způsoby, mezi něž patří push notifikace (popsané blíže v kapitole 3.3), zaslání e-mailu, SMS, zprávy do aplikace MS Teams [48] nebo do aplikace Slack [82]. Na základě stanovené události může být také nastaveno např. vykonání procedury [69]. Z těchto možností je nejdůležitější zaslání notifikace uživateli pomocí mobilní aplikace PRTG.

Uživatelé PRTG recenzující tento nástroj na serveru Trustradius brali za jeho přednosti možnost monitorovat téměř vše, co je připojené k internetu. Dalším pozitivem podle nich je jednoduchost prvotního nastavení. Za pozitivum také považují skvělou podporu vývojářů tohoto nástroje. Negativem podle uživatelů je cena nástroje PRTG. Výhodnější by podle nich bylo počítání ceny podle množství monitorovaných strojů [68].

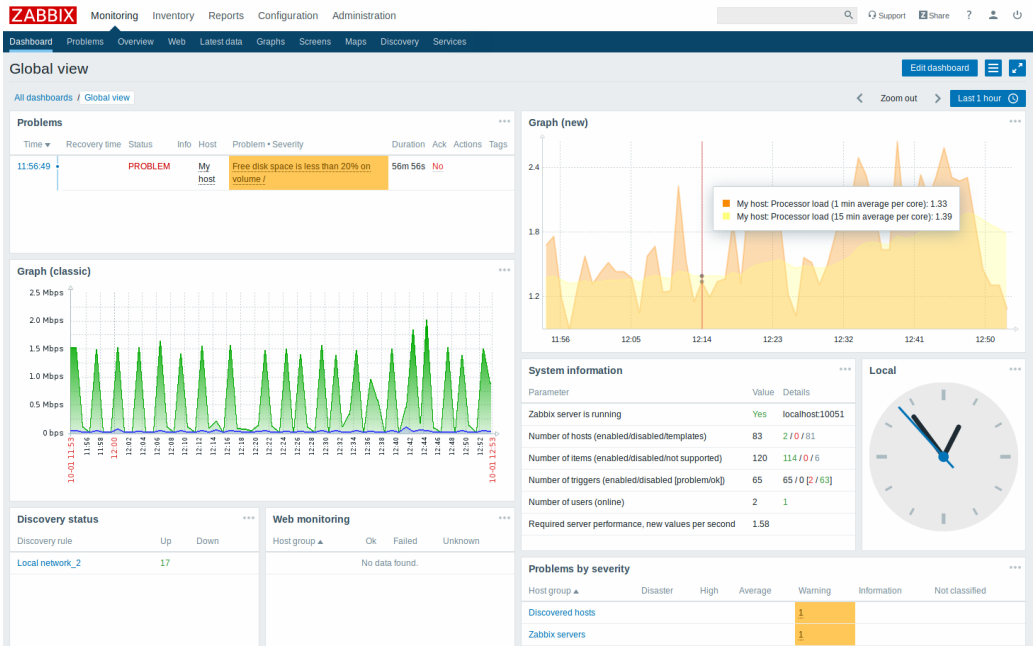
2.1.3 Zabbix

Zabbix [100] je další z open-source monitorovacích nástrojů vedle Icinga2 a Nagios.

Stejně jako PRTG obsahuje Zabbix auto-discovery systém. Pro monitorování sítě používá tzv. agenty, což je software, který beží na monitorovaném prvku a komunikuje se Zabbix serverem. Zabbix nicméně také podporuje klasické monitorování pomocí protokolu SNMP. Stejně jako ostatní monitorovací systémy se Zabbix ovládá přes webové rozhraní. Jeho výhodou je mimo jiné podpora monitorování virtuálních strojů [47]. Na obrázku 2.3 je možné vidět pvodní obrazovku webového rozhraní pro Zabbix.

Na základě vytvořené události může i Zabbix zpracovat a odeslat notifikaci. Existuje několik předem stanovených programů, se kterými je Zabbix kompatibilní, mezi něž patří například Discord [14], Jira [41], Slack nebo MS Teams. Uživatel sám může také do Zabbix integrovat vlastní nástroj, do kterého budou odesílány informace o stavu, například vlastní mobilní aplikace [101].

Zabbix uživatelé považují za kvalitní nástroj, který má velké množství výhod, ale i nevýhod. Za výhodu považují množství grafů ve webovém rozhraní a jeho možnost nastavit podmínky upozornění. Díky tomu, že Zabbix je open-source program, je podle uživatelů jednoduché jej integrovat s ostatními podpůrnými nástroji. Za plus považují někteří uživatelé obsáhlou uživatelskou dokumentaci. Na druhé straně nicméně stojí uživatelé, kterým se dokumentace nelíbí. Důvodem je, že přestože je obsáhlá, nedává uživateli kompletní informace pro instalaci tohoto nástroje. Uživatelé si také stěžují na příliš vysoké množství znalostí napříč několika dispciplínami z oboru databází a operačních systémů, které je pro instalaci Zabbixu potřeba znát.



Obrázek 2.3: Webové rozhraní Zabbix 4.0
Zdroj: https://www.zabbix.com/documentation/4.0
/manual/introduction/whatsnew400

Velkým nedostatkem je také chybějící oficiální podpora. V případě, že je v programu nalezena chyba, může absence oficiální podpory znamenat i několik týdnů do doby, než bude tato chyba opravena, neboť je nástroj spravovaný pouze komunitou [102].

2.1.4 Icinga2

`Icinga2` [35] je open-source monitorovací nástroj dostupný pro `Unixové` [93] operační systémy. Původní monitorovací systém `Icinga` vznikl jako odnož staršího monitorovacího systému `Nagios`, přičemž spousta modulů je mezi `Nagiosem` a `Icingou` kompatibilní. Dovoluje správci monitorovat veškerá zařízení, která si přidá do konfiguračního souboru, tedy i mimo vnitří síť. `Icinga2` s sebou nese několik rozšíření včetně webového rozhraní `Icinga-web2` [37], které je velice intuitivní, jednoduché na pochopení a umožňuje správci vykonávat jednoduché operace jako potvrzení odstávky a její případné naplánování, manuální kontrolu dostupnosti služeb i serverů, odeslání notifikací o službě a další.

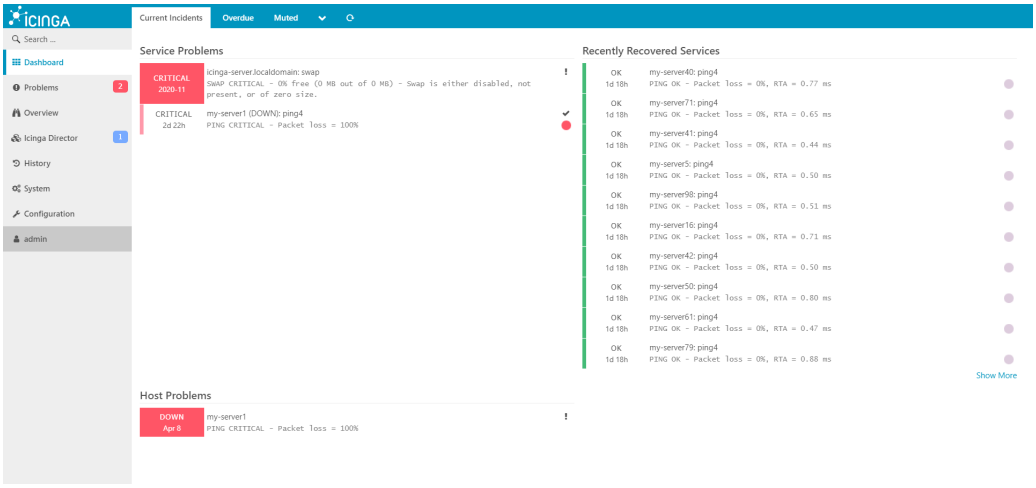
Nastavení monitorovaných objektů lze provádět 2 způsoby: přidáním služby nebo serveru do konfiguračního souboru, nebo přes rozšiřující nástroj pro `Icingaweb2`, `Icinga Director` [34]. Jde o nástroj, který dovoluje

správci upravovat konfiguraci `Icinga2` přes webové rozhraní, čímž mnohonásobně zjednodušuje práci s tímto monitorovacím nástrojem.

Další výhodou tohoto monitorovací nástroje je velice dobře zpracované `Application Programming Interface` [33] (dále jen `API`), se kterým se komunikuje přes tzv. `ApiUsery`. V tomto `API` jsou naimplementovány i tzv. `EventStreamy`, které dovolují uživatelům posílat notifikace pro své aplikace.

Pro uživatele je velice důležitá přehlednost dat, kterou `Icinga2` disponuje. Ve svém webovém rozhraní (vyobrazeném na obr. 2.4) dovoluje uživateli zobrazit služby a servery podle různých skupin, ať už serverů, či služeb. `Icinga2` také uchovává historii, přičemž z některých vytváří metriky, podle kterých může správce vyhodnotit situaci, která se v síti odehrává.

Velkou nevýhodou `Icinga2` je velice složitá instalace pro méně zkušené `Unixové` uživatele. Dokumentace, která je dostupná na webových stránkách se skládá z návodu doplněného o vysvětlivky k jednotlivým krokům. Tato dokumentace je nicméně neúplná, neboť v ní chybí některé kroky, bez kterých nelze v konfiguraci pokračovat, a bez externích návodů a oficiálního fóra `Icinga2` je pro nováčka velice obtížné tento program nainstalovat a nakonfigurovat.



Obrázek 2.4: Webové rozhraní Icingaweb2

Uživatelé považují nástroj `Icinga2` za velice kvalitní. Jeho obrovskou předností je jeho škálovatelnost, díky které je možné monitorovat desetitisíce prvků. Jako velké plus je považováno jednoduché přidávání nových objektů k monitorování. Největší předností je jeho dobře zdokumentované `API`. Pozitivní je i to, že nástroj je zcela zdarma. Přehledné je i webové rozhraní, které nabízí jednoduchou, ale srozumitelnou přehledovou obrazovku. Přestože je přidávání objektů k monitorování jednoduché, uživatelům schází možnost přidávat tyto objekty pomocí uživatelského rozhraní. Obtížná je podle nich

i počáteční instalace a konfigurace pro začínající Unixové uživatele [36].

2.2 Porovnání monitorovacích nástrojů

V předchozích kapitolách byly prozkoumány dostupné monitorovací nástroje, zjištěny jejich vlastnosti a základní práce s nimi. Dále byly prostudovány uživatelské recenzce a zjištěny výhody a nedostatky jednotlivých nástrojů. Na základě těchto poznatků byly vybrány parametry, podle kterých by se potenciální uživatel mohl rozhodovat při výběru monitorovacího systému.

Prvním zvoleným parametrem je cena. Pro začínajícího uživatele monitorovacích nástrojů může být právě cena rozhodujícím parametrem při výběru. Dalšími vysoce důležitými parametry jsou možnosti instalace nástroje na Linux a Windows, neboť v roce 2019 ovládaly Windows servery 72,1 procenta trhu [85]. Komunikace pomocí API je parametr důležitý pro tvorbu mobilních aplikací. Právě díky API může mobilní aplikace a monitorovací nástroj vyměňovat informace. Přestože existují i jiné možnosti komunikace, např. WebSockety [94], je komunikace pomocí API standardem v tomto odvětví. Důležitá je pro uživatele i oficiální podpora vývojářů. Pokud je v nástroji objevena chyba a její odstranění trvá týdny, může uživateli způsobit při monitorování velké problémy. Posledním parametrem, podle kterého by se uživatel mohl řídit, je hodnocení ostatních uživatelů. Tato hodnocení byla brána z webu Trustradius.

Poznatky byly zpracovány do tabulky:

Monitorovací nástroj	Nagios Core	PRTG	Zabbix	Icinga2
Placené	Ne	Ano	Ne	Ne
UNIX/Linux instalace	Ano	Ne	Ano	Ano
Windows instalace	Ne	Ano	Ne	Ne
Komunikace pomocí API	Ano	Ano	Ano	Ano
Oficiální podpora	Ne	Ano	Ne	Ano
Hodnocení uživatelů	7.7/10	8.7/10	8.1/10	9.0/10

Tabulka 2.1: Srovnání monitorovacích nástrojů

Nejjednodušším nástrojem pro práci byl podle uživatelských recenzí nástroj PRTG, což může být vysvětleno tím, že je dostupný pouze na platformě Windows. Napříč všemi neplacenými nástroji byla shoda o velice obtížné počáteční instalaci a konfiguraci těchto nástrojů. Uživatelům také scházela oficiální podpora nebo nedostatečná uživatelská dokumentace.

Nejlépe hodnoceným nástrojem je Icinga2. Je to způsobeno přehledným webovým rozhraním, jeho cenovkou a podporou komunity a vývojářů, kteří jsou velice aktivní na oficiálním fóru. Za velice pozitivní považují i dobře zpracované API, pomocí něhož je možné posílat data i do jiných aplikací.

Z výzkumu podle tabulky 2.1 vyplývají 2 výsledky. Nejlepším nástrojem pro uživatele, kteří nemají znalosti související s Unixovými systémy, je PRTG. Je to dáno tím, že tento nástroj je dostupný na zařízení s OS Windows. Výhodou je i oficiální podpora vývojářů tohoto produktu a také vysoké hodnocení uživatelů. Nevýhodou je však cena produktu.

Pro pokročilejší uživatele je nicméně nejlepším nástrojem pro monitorování Icinga2. Tento výsledek je podpořen nejvyšším uživatelským hodnocením 9.0/10 a oficiální vývojářskou podporou. Velkou výhodou je pro uživatele i to, že je nástroj vydán pod copyleft licencí - uživatel může nástroj používat doma, ale i pro komerční účely. Z těchto důvodů je Icinga2 zcela nejlepší volbou na současném trhu s monitorovacími nástroji.

3 Vývoj mobilních aplikací

Vzhledem k vzrůstající oblibě mobilních telefonů [13] nad stolními zařízeními vzrůstá i významnost mobilních aplikací. Přestože webovou stránku si lze prohlédnout i z mobilu, uživatelé preferují podle výzkumu [42] z roku 2014 využívání mobilních aplikací. Už před 8 lety trávili uživatelé průměrně denně 2 hodiny a 42 minut používáním telefonu, z čehož 86 procent tohoto času uživatelé trávili používáním aplikací. Tento trend je podpořen novějším výzkumem [96], podle kterého se čas strávený používáním telefonu zvýšil na více než 4 hodiny denně, přičemž poměr užívání aplikací a webového prohlížeče se téměř nezměnil.

Níže jsou popsány výhody a nevýhody jednotlivých řešení [46, 50]:

- Mobilní aplikace využívají lépe nativních zdrojů zařízení, jako je úložiště, nebo fotoaparát, než responzivní webové stránky.
- Největší výhodou mobilních aplikací je možnost zasílání push notifikací. Ty uživatele dokážou informovat o událostech, které by jej mohly zajímat.
- Výhodou webových aplikací je jejich větší dosah. Webovou aplikaci může používat uživatel na mobilním zařízení i na stolním počítači, aplikace tak získá větší množství uživatelů.
- Tvorba responzivní webové aplikace je jednodušší, než vytvářet úplně novou mobilní aplikaci, z toho plyne, že jsou i mnohem nižší náklady na vývoj.
- Pokud není webová stránka tvořena s důrazem na responzivitu, může být navigace přes tuto stránku velice obtížná, zatímco při tvorbě mobilní aplikace je na tento fakt kladen důraz.
- Při otevírání webové stránky se musí její obsah stahovat. Mobilní aplikace načte své soubory přímo z telefonu, což může způsobit její rychlejší načtení oproti webové stránce.

Mobilní aplikace jsou v kontextu monitorování velice důležité. Správce sítě musí mít vždy přehled o dění na síti a musí být okamžitě informován. Zároveň je naprosto nezbytné, aby byla data přehledná a operace s nimi jednoduchá. S tímto souvisí právě možnost mobilních aplikací zasílat notifikace,

které správce upozorní na události i přesto, že právě nedrží mobilní telefon v ruce a nekontroluje stále stav sítě.

V současnosti se na trhu s chytrými telefony nachází mnoho firem, na jejichž zařízeních se ale používají pouze 2 operační systémy: `Android`, vyvíjený společností Google, a `iOS` firmy Apple [3]. Aplikace musí být naprogramována v jazyce, který daný operační systém podporuje. Aplikaci naprogramovanou v jiném jazyce na zařízení nelze spustit [91].

Programátor si tedy musel vybrat, pro který operační systém bude aplikaci vyvíjet, nebo vyvíjet aplikaci pro obě platformy ve dvou rozdílných jazycích. To v poslední době ale přestává být problémem díky vzniku frameworků [27] (framework je základní kámen pro vývoj softwaru, aby vývojář nemusel začínat od úplného začátku) dovolujících vyvíjet nativní aplikace pro více operačních systémů najednou. V následujících sekcích budou popsány různé typy vývoje a jejich výhody a nevýhody.

3.1 Vývoj nativních mobilních aplikací

Vývoj nativních aplikací je proces tvorby softwaru, který běží na jednom konkrétním typu zařízení. U zařízení závisí na operačním systému, který na zařízení běží. V dnešní době se bavíme zejména o 2 operačních systémech - `Android` a `iOS`.

Obrovskou výhodou nativních aplikací je jejich přístup k funkcím telefonu, jako je fotoaparát nebo mikrofon, což je u hybridních aplikací popsaných v kapitole 3.2 složitější.

Mezi výhody vývoje nativních aplikací můžeme zařadit vyšší výkon, což je způsobeno tím, že aplikace komunikuje s `API` telefonu bez použití jakéhokoliv middleware, prostředníka, který umožňuje komunikaci se zařízením u hybridních aplikací. To je důležité zejména u her nebo aplikací, které jsou náročné na grafické vykreslování.

Dalším plusem je, že vzhled těchto aplikací je konzistentní, neboť se využívá standardní Software Development Kit [4] operačního systému (`SDK OS`). U hybridních aplikací mohou některé komponenty vypadat jinak, než je u dané platformy zvykem, což může tvořit při vykreslování uživatelského rozhraní problémy.

Narozdíl od hybridních aplikací, nativní aplikace mají přístup k nejnovějším technologiím na zařízení. Vývojáři hybridních aplikací musí čekat, než budou tyto technologie zpřístupněny ve frameworcích.

Funkční aplikace může být nahrána do obchodu `Google Play` [81] nebo `App Store` [81], odkud si ji ostatní uživatelé mohou stáhnout. Další možností

je nainstalovat aplikaci pomocí instalačního balíčku stáhnutého z internetu [59].

3.1.1 Vývoj nativních aplikací pro Android

Aplikace pro Android mohou vývojáři vytvářet pomocí nástrojů, jako je například `Android SDK`, který je dodáván spolu s `Android Studiem` [5], oficiálním `IDE` [39] (Integrated Development Environment) pro platformu `Android`. Druhou možností, kterou může vývojář využít, je platforma `Firebase` [25] spravovaná společností Google.

Aplikace pro Android mohou být vyvíjeny na všech platformách, `Windows`, `UNIX` i `MacOS` [59].

Poté, co je aplikace vytvořena a otestována, je dalším krokem vydání aplikace v obchodě `Google Play`. Z aplikace musí být vytvořen balíček, který musí být podepsán soukromým klíčem. Tento balíček nese název `Android App Bundle` [80] (zkráceně `AAB`). Uživatel si do zařízení stahuje soubory `APK` [80] (Android Application Package), které platforma Google automaticky vytvoří ze souboru `AAB`. Aby mohl vývojář vydávat aplikace v obchodě `Google Play`, musí mít vytvořený `Google Play Developer Console` účet, za který je požadován poplatek 25 USD [84].

3.1.2 Vývoj nativních aplikací pro iOS

Narozdíl od `Android`, nativní aplikace pro systém `iOS` mohou být vyvíjeny pouze na zařízeních společnosti Apple.

Apple nicméně také dodává svým vývojářům nástroje, pomocí nichž mohou být aplikace vyvíjeny. Patří mezi ně:

- iOS SDK
- XCode [98] - oficiální IDE
- TestFlight [90] - platforma pro testování aplikací

Oficiální obchod s aplikacemi pro zařízení Apple, `App Store`, má mnohem přísnější pravidla, kvůli kterým může být aplikace odmítnuta [59].

3.2 Vývoj multiplatformních mobilních aplikací

Multiplatformní vývoj je proces vytváření služby nebo aplikace, která je schopná fungovat na několika operačních systémech [89]. Toho může být

docíleno dvěma způsoby: [12]

1. Hybridní aplikace - je nižší úroveň vytváření multiplatformních aplikací. Aplikace závisí na webovém prohlížeči nainstalovaném v telefonu, který umí vykreslit `HTML5, CSS` (Hypertext Markup Language a Cascade Styles) a `JavaScript` [66]. Tyto aplikace jsou obaleny kontejnerem specifickým pro každý operační systém.

2. Tvorba nativních aplikací pomocí multiplatformního frameworku. Jde o vyšší způsob vytváření multiplatformních aplikací. Ty jsou vytvářeny pomocí nástrojů jako `React Native` [62], nebo `Flutter` [78], které používají nativní `API` zařízení. Výhodou nativních aplikací je jejich lepší výkon oproti hybridním aplikacím. Vývojáři se také mohou naučit pouze jeden programovací jazyk, se kterým mohou vytvářet aplikace pro všechny druhy zařízení. Tyto frameworky mají velikou oblíbenost a jsou podporovány vývojáři i komunitou.

3.2.1 Výhody multiplatformního vývoje

Důvodů pro multiplatformní vývoj je několik [12].

1. Širší dosah - Díky tomu, že je aplikace dostupná na více platformách, než kdyby byla vyvíjena pouze pro jednu, zvýšíme množství uživatelů, kteří aplikaci mohou používat.

2. Náklady - Vývoj multiplatformní aplikace je podstatně levnější, neboť eliminujeme náklady na několikanásobný vývoj aplikace.

3. Znalosti - Multiplatformní aplikace jsou jednodušší na vývoj. Vývojář nemusí znát sadu jazyků pro jednotlivé platformy, stačí mu pouze jeden, který je navíc obecně používaný a oblíbený.

3.2.2 Používané frameworky

V následující kapitole jsou shrnuty podstatné informace o nejoblíbenějších frameworcích mezi vývojáři multiplatformních aplikací podle webu [44].

Flutter

Framework `Flutter` [78] je vyvíjený společností Google. K vývoji je využíván jazyk `Dart`, který byl vytvořen přávě pro tento framework. `Dart` je velice výkonný jazyk, který svojí syntaxí připomíná jazyk `Java` [66], proto je pro

vývojáře poměrně jednoduché se ho naučit. Přechod k tomuto jazyku navíc zlehčuje vysoce kvalitní oficiální dokumentace, která je velice podrobná. Velkou výhodou je i neustále zvětšující se komunita `Flutter` vývojářů, díky které vznikají další návody a knihovny. Aplikace vyvíjená ve `Flutteru` je svojí výkonností velice blízko nativní aplikaci. To je způsobeno tím, že aplikace naprogramovaná ve `Flutteru` je přímo přeložena do nativního kódu. Další výhodou při vývoji aplikace pomocí `Flutteru` je funkce `Hot Reload`, která umožňuje vývojáři vidět změny provedené při programování v reálném čase. Aplikace se nemusí při každé změně kompilovat od začátku, což zlepšuje uživatelský zážitek při programování a zároveň urychluje samotný vývoj.

Tento framework má nicméně i své nevýhody. Vzhledem k tomu, že je na trhu pouze krátkou dobu, existuje pouze malé množství komunitou vytvořených knihoven. Další velkou nevýhodou je velikost balíčků tvořených tímto frameworkem. Ve výzkumu [10], ve kterém byly porovnávány oblíbené frameworky, byly vytvořeny balíčky jedné aplikace pomocí několika metod. Balíček vytvořený pomocí `Flutteru` byl i čtyřnásobně větší, než balíčky vytvořené konkurenčními metodami. Aplikace tak může zabírat zbytečně velké množství místa v telefonu.

`Flutter` v sobě přímo obsahuje správu push notifikací pomocí `Firebase`, neboť jsou obě platformy vyvíjené společností Google [26]. `Flutter` předstihl v roce 2021 `React Native` (popsaný v kapitole 3.2.2) ve svojí oblíbenosti [44].

Xamarin

`Xamarin` [45] je open-source platforma podporovaná společností Microsoft. Podporuje jazyk `C#` [66], a je tak vhodný pro vývojáře s frameworkem `.NET`. Obrovskou výhodou frameworku `Xamarin` je `Xamarin Test Cloud`, nástroj, který dokáže vyvíjenou aplikaci testovat ve stejném okamžiku na mnoha různých zařízeních. Xamarin dovoluje vytvářet mobilní aplikace pro `Android`, `iOS`, ale i `Windows`. Obsahuje objektovou knihovnu, která standardizuje práci s `XML` soubory, databázemi, vstupy a výstupy, řetězci a síťovou komunikací. Komponenty pro projekt lze získat z `Xamarin Component Store`, ve kterém najdeme placené komponenty, ale i komponenty zadarmo.

Obrovskou nevýhodou je nedostatečná podpora komunity, která může scházet při vytváření nových knihoven a komponent. `Xamarin` dovoluje posílání push notifikací. Jedním řešením, které je na webu pospáno pouze pro `iOS`, je posílání notifikací přes platformu `Microsoft Azure` [97], což je platforma pro cloudové služby spravovaná společností Microsoft [49]. Druhou

možností je využití knihovny [24], která dovoluje komunikaci s Firebase Cloud Messaging (FCM) serverem.

Cordova

Cordova [99] je framework společnosti Adobe, který využívá prvky HTML5, CSS a JavaScriptu pro tvorbu multiplatformních aplikací. Výhodou vývoje aplikací v tomto frameworku je jednodušší publikování nových aktualizací oproti vývoji nativních aplikací. Namísto potřeby stahovat pokaždé novou verzi aplikace, Cordova rovnou aktualizuje aplikaci bez potřeby stahovat celou novou verzi. Pro framework Cordova existuje komunitou vytvořená knihovna, která implementuje možnost zasílání notifikací.

Mezi nevýhody vyvíjení aplikací ve frameworku Cordova patří nižší výkon aplikací, což je zvláště u hybridních aplikací standardní problém. Další nevýhodou jsou možné problémy s kompatibilitou některých pluginů napříč různými platformami. Ty nemusí všechny platformy podporovat a aplikace tak může fungovat nesprávně, nebo nemusí fungovat vůbec. V případě, že tento problém nastane, musí být z aplikace vytvořen potomek a ten spravován jako samostatná aplikace, což zvyšuje náklady a časové nároky na vývoj [28].

React Native

React Native [62] byl podle serveru Statista [44] nejpoužívanějším frameworkem za roky 2019 a 2020. V roce 2021 ale jeho oblíbenost klesla na úkor frameworku Flutter. Vychází z frameworku React, který byl vyvinut společností Facebook, a který je používán pro vyvíjení webových stránek. React Native je open-source framework, který využívá mnohé nástroje skriptovacího jazyka JavaScript, jenž je v dnešní době velice oblíbený mezi webovými vývojáři [79], což mnohonásobně zjednodušuje přechod mezi vyvíjením webových stránek a aplikací v React Native. Podle serveru Medium dovoluje vývojáři vytvářet moduly pro React Native nativními jazyky, zejména Javou, C++, Swift, Python a Objective-C [45]. Aplikace je v tomto frameworku tvořena vkládáním komponent do sebe (anglicky nesting). Tyto komponenty jsou následně zkompilovány do nativních komponent každého operačního systému [45].

Pro framework React Native existuje platforma Expo [17], která zjednodušuje vývoj aplikací v tomto frameworku. Obsahuje v sobě spoustu výhod, mezi něž patří automatická aktualizace aplikace nebo implementace notifikací. Platforma také umožňuje testování mobilní aplikace pomocí aplikace Expo Go na Android i iOS, nebo pomocí Apple TestFlight pro iOS. Expo

také umožňuje tvorbu a následnou distribuci balíčků pro `Google Play` a `App Store` [20].

3.2.3 Shrnutí vývoje aplikací

Pro aplikace, u kterých je vyžadován vyšší výkon, standardizace vzhledu a lepší přístup k funkcím telefonu je lepší vyvíjet aplikace v nativních jazycích každé platformy. Tyto výhody mohou být ale vyrovnány v případě, že aplikace se zaměřuje na jednoduché úkony, které nevyžadují maximální výkon a maximální optimalizaci. V případě monitorovacích nástrojů jde o maximální dosah aplikace, uživatel nepotřebuje zobrazovat složité modely a obrázky. Z toho důvodu se zdá multiplatformní vývoj jako preferovaný způsob tvorby tohoto typu aplikací.

Na základě předchozího zkoumání byla vytvořena tabulka 2.1 se základními informacemi o frameworcích, sloupec oblíbenosti je vypracován na základě informací získaných z webu [44].

Byť oblíbenost technologie sama o sobě nesvědčí o její kvalitě, je i tak tento ukazatel dobrou pomůckou při předpovídání trendů toho, jakým směrem se vývoj aplikací bude směrovat v budoucích letech. Díky tomu je možné si vybrat framework, který bude mít dlouhou životnost a podporu.

Framework	Jazyk	Notifikace	Oblíbenost 2020	Oblíbenost 2021
Flutter	Dart	Ano	2.	1.
Cordova	HTML5, CSS, JavaScript	Ano	3.	3.
Xamarin	C#	Ano	5.	5.
React Native	JavaScript	Ano	1.	2.

Tabulka 3.1: Srovnání dostupných frameworků

Všechny z těchto frameworků dovolují vývojáři implementovat notifikace, které jsou v tomto typu aplikace kritickou součástí. Dalším důležitým faktorem při programování je podpora platformy a existence komunity. Přestože `Flutter` se v roce 2021 stal nejoblíbenějším frameworkem pro vývoj multiplatformních aplikací, je stále v současné chvíli nejlepší volbou pro vývoj nové mobilní aplikace framework `React Native`. Je to zejména proto, že v současnosti existuje veliké množství knihoven pro `React Native` a pořád za tímto frameworkem stojí obrovská komunita, která jej neustále rozšiřuje. Tato komunita je rozsáhlá zejména proto, že programovacím jazykem používaným ve frameworku React Native je `JavaScript`, aktuálně nejoblíbenější programovací jazyk na světě. Ani frameworky `Cordova` či `Xamarin` nemají v současné době dostatečnou podporu komunity, aby mohly převážit nad frameworkem `React Native`.

Do frameworku `React Native` jsou stále přidávány nové funkce, jako jedny z posledních tzv. `hooks`, které zjednodušují práci se stavem komponent. Dalším důvodem pro použití `React Native` je platforma `Expo`, která zjednodušuje a standardizuje kompilaci, překládání a vydávání aplikací. V této platformě je také implementována možnost push notifikací, jež jsou důležitým požadavkem v tomto druhu aplikací, přičemž `Expo` zjednodušuje práci s nimi a komunikuje s notifikačním serverem namísto vývojáře.

Fakt, že `React Native` je nejlepší volbou podporuje i empirický výzkum multiplatformních frameworků vydaný v roce 2020. Ve výzkumu bylo porovnáno několik frameworků pro vývoj multiplatformních aplikací, přičemž z výše uvedených jsou zmiňovány `React Native`, `Flutter` a `Cordova` (zastoupená nadstavbou v podobě frameworku `Ionic` [40]). Ve výzkumu je porovnáváno několik aspektů, podle kterých se vývojář může rozhodovat. Z vybraných frameworků získal nejlepší hodnocení React Native, následuje `Flutter` a poslední je `Ionic`, který je zástupcem frameworků pro vývoj hybridních aplikací jako je `Cordova` [10]. Ze všech výše zmíněných důvodů je `React Native` aktuálně nejlepším frameworkem pro tvorbu mobilních aplikací.

3.3 Notifikace

Notifikace jsou nezbytnou součástí dnešních aplikací. Mají významnou roli v marketingu, proto je společnosti využívají k upoutání pozornosti uživatele k jejich mobilní aplikaci. Důležitou roli mají ale i v monitoringu. Díky upozornění mohou správci sítě rychleji zareagovat na výpadek v síti.

Notifikace lze v mobilních aplikacích implementovat 2 způsoby:

- Lokální notifikace
- Push notifikace

Rozdíl mezi těmito druhy je pro tvorbu této mobilní aplikace významný. Zatímco obdržená zpráva má pro uživatele stejný význam, u implementace se liší, kdy mohou být tyto notifikace zasílány.

Zasílání lokálních notifikací naprosto závisí na aplikaci, ve které byla tato funkcionalita implementována. Tento typ notifikací se může vytvářet v aplikaci, která je zapnutá, nebo je na pozadí. Program může také vytvořit notifikaci, které nastaví čas, ve který se uživateli má zobrazit. Problém ale nastává, pokud operační systém tento proces odstraní z paměti, neboť aplikace v tomto případě již není spuštěna a nemůže vytvářet notifikace.

To je pro správce sítě velký problém, neboť se nemusí dozvědět o kolapsu prvků v síti, jež jsou kritické k fungování.

Tento problém řeší push notifikace. Uživatelská aplikace pouze vygeneruje token, který je následně odeslán na backend spojený s mobilní aplikací, jenž se následně stará o zasílání notifikací jednotlivým uživatelům. Pro tyto notifikace jsou používány již standardně existující služby, které se starají o samotné doporučení notifikace [43].

Mezi nejznámější produkty v tomto odvětví patří `OneSignal`, `PushBots` nebo `Google Firebase` [64]. Právě `Firebase` [23] spolu s `Apple Push Notification Service` [6] tvoří dvojici služeb, které jsou standardem pro posílání push notifikací do chytrých telefonů s operačním systémem `Android` a `iOS`. Právě s těmito 2 platformami navíc spolupracuje vývojový nástroj `Expo`, který obstarává komunikaci s těmito službami [21].

Všechny frameworky popsané v přechozích kapitolách využívají velmi podobného způsobu při posílání push i lokálních notifikací. Využívají buď oficiálně vydané knihovny, nebo ty vytvořené komunitou.

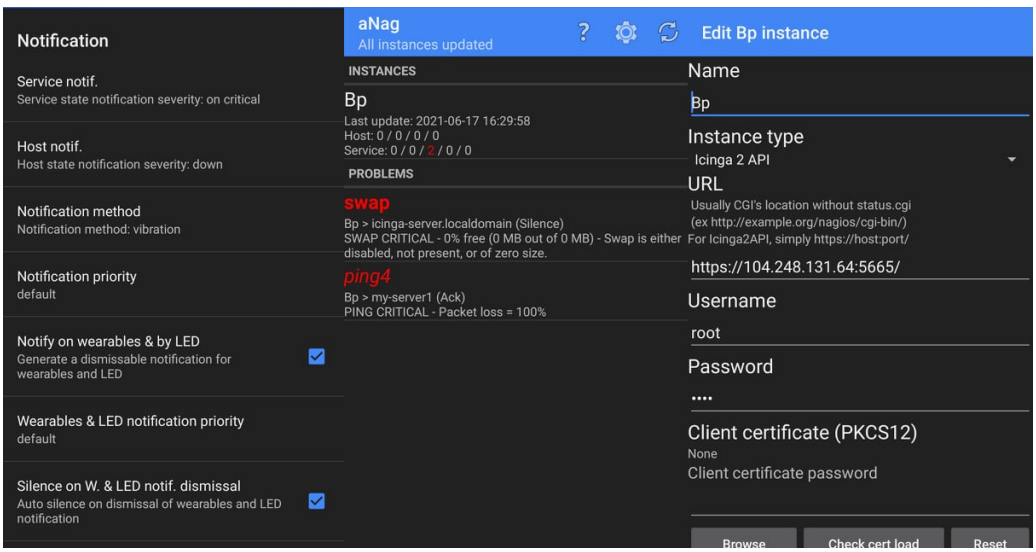
U `Cordova` a `React Native` jsou to pouze komunitně vyvtvořené knihovny, nicméně pro `React Native` byla vytvořena platforma `Expo`, která integraci s push notifikacemi mnohonásobně ulehčuje.

4 Dostupné mobilní aplikace

V následujících kapitolách jsou shrnuty základní informace o mobilních aplikacích pro monitorovací nástroje. Do vyhledávače `Google Play` byla zadána hesla "`Icinga`", "`Nagios`", "`Zabbix`"a "`PRTG`". Tato slova byla vybrána s cílem prozkoumání mobilních aplikací kompatibilních s nástroji zkoumanými v kapitole 2.1. Následným zkoumáním výsledků hledání byly vybrány relevantní aplikace. Ke srovnání byla vybrána i aplikace `IciView` [38], která je dostupná pouze pro mobilní zařízení se systémem `iOS` a `iPadOS`.

4.1 aNag

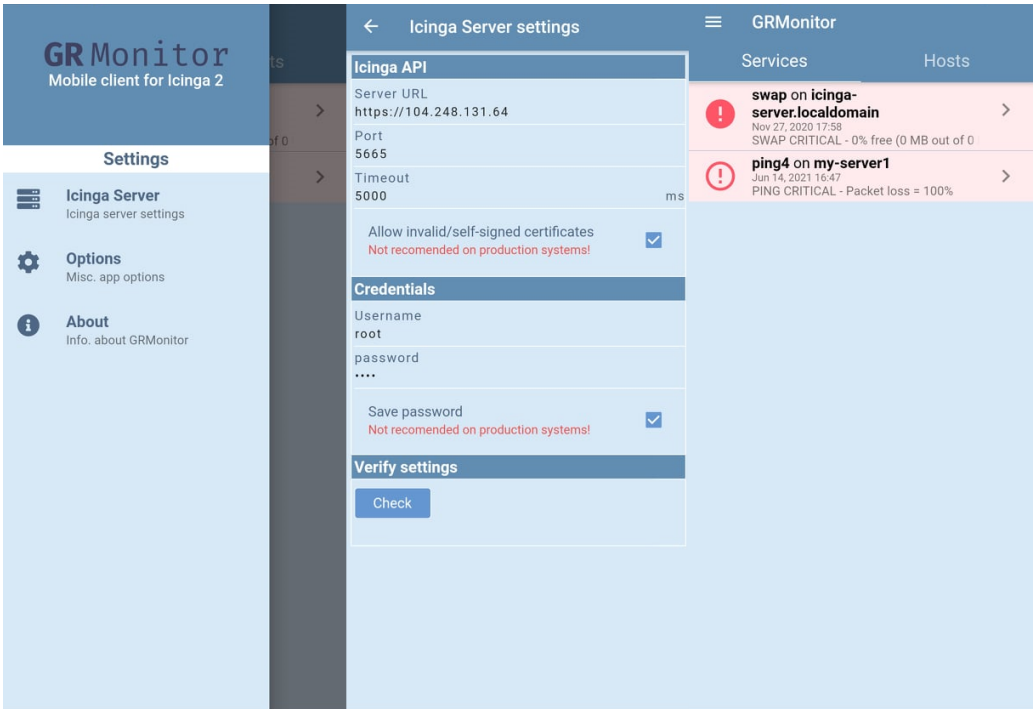
`ANag` (který je možné vidět na obrázku 4.1) je podle počtu stažení na Google Play nejstahovanější mobilní aplikací na Android. Ve svém nastavení nabízí možnost připojit se na několik monitorovacích systémů, z nichž již dříve zmíněné jsou `Nagios` a `Icinga2`. Uživatelské prostředí není pro nového uživatele intuitivní, lehce se v něm dá ztratit. Výhodou je, že v nastavení lze přidat více instancí monitorovacích nástrojů. Aplikace aNag zobrazuje pouze problémy, správně fungující služby a servery již nezobrazuje. Zvládá jednoduchou správu upozornění. V případě, že je aplikace vypnutá (není ani na pozadí), neposílá uživateli notifikace o změně stavu na serveru. Pro iOS tato aplikace dostupná není.



Obrázek 4.1: Mobilní aplikace aNag

4.2 GRMonitor

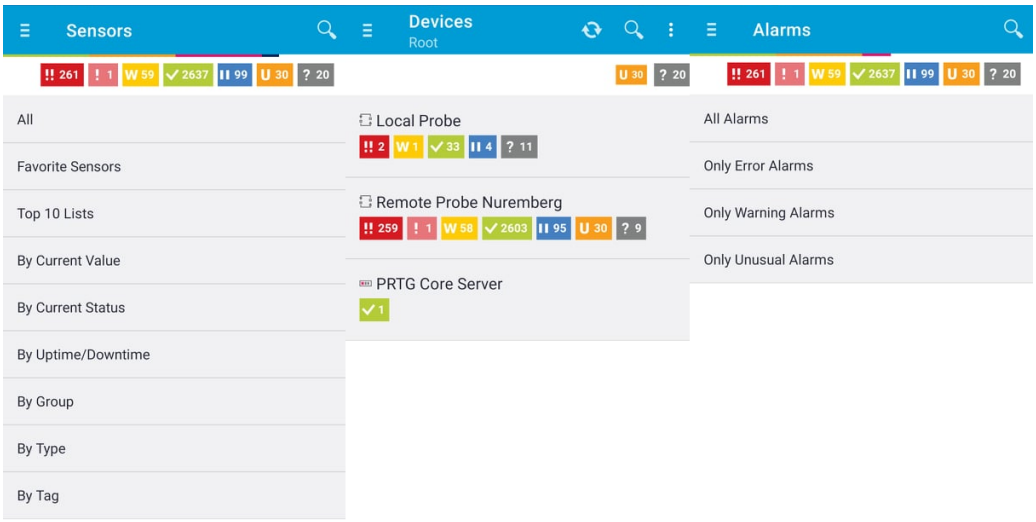
`GRMonitor` je druhá nejstahovanější aplikace pro nástroj `Icinga2` z obchodu `Google Play`, přičemž je dostupný i pro mobilní telefony se systémem `iOS`. S `Icinga2` serverem komunikuje přes jeho `API`, přičemž v nastavení dovoluje uživateli ignorovat nevalidní/self-signed certifikáty. Vzhledem je `GRMonitor` (na obr. 4.2) přívětivější než `aNag`, jednotlivé informace o službách jsou zobrazeny přehledněji. Stejně jako `aNag` zobrazuje tato aplikace pouze služby a servery, u kterých je hlášena chyba. Aplikace dovoluje vyvolat okamžitý recheck a v jednoduchém formuláři lze potvrdit informaci o chybě. V aplikaci se mi nepodařilo zobrazit si podrobnosti o serveru s chybou, obrazovka se nenačetla.



Obrázek 4.2: Mobilní aplikace GRMonitor

4.3 PRTG

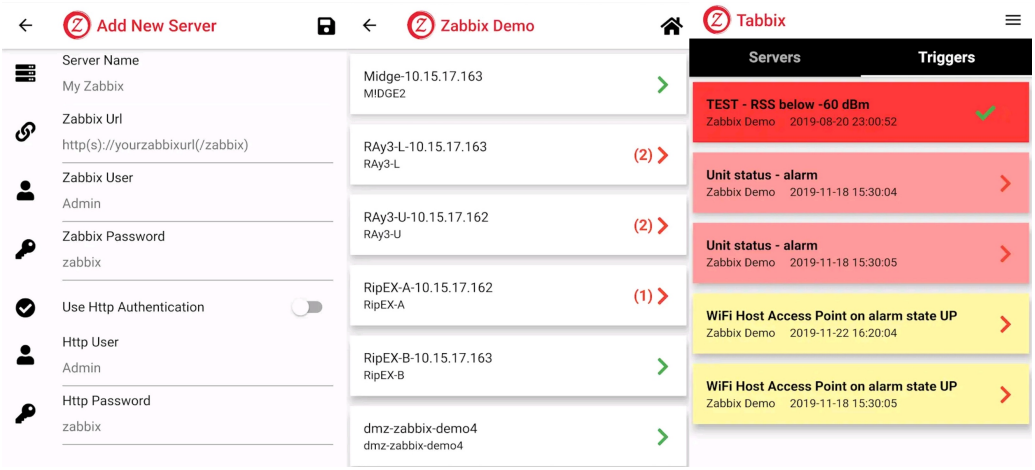
Aplikace pro `PRTG Network Monitor` je podstatně vyspělejší než předchozí aplikace, což může být zdůvodněno tím, že `PRTG Network Monitor` je placená služba a má tedy větší podporu společnosti, pro kterou je aplikace vytvořena. Aplikace kopíruje vzhled webového rozhraní a dovoluje mu stejné možnosti. Po aplikaci, která je vyobrazena na obrázku 4.3, se dá pohybovat pomocí tzv. drawer menu, do podrobností o službách a serverech se dá dostat po kliknutí na samotnou službu. Objekty se dají filtrovat také do knihoven, což zjednodušuje uživateli vyhledávání. Aplikace je dostupná pro `Android` i `iOS` a to zcela zdarma.



Obrázek 4.3: Mobilní aplikace PRTG for Android

4.4 Tabbix

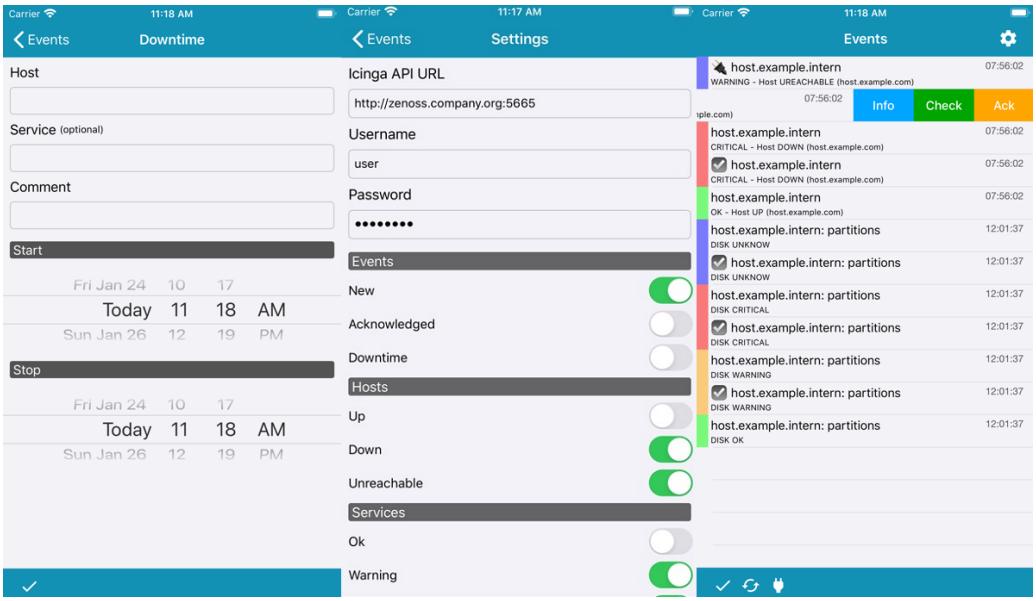
`Tabbix` je aplikace pro `Android`, která je kompatibilní s nástrojem `Zabbix`. Dovoluje monitorovat triggery, servery a jejich detaily. Uživatel může dokonce ke sledování přidat více `Zabbix` serverů, ty si může prohlédnout přehledně na jedné obrazovce. Dále podporuje základní funkce, které může uživatel vykonávat ve webovém prohlížeči. Uživatel si může prohlédnout grafy příslušící jednotlivým objektům. `Tabbix` navíc implementuje push notifikace, uživatel dostává informace tedy i v případě, že není aplikace zapnutá. Vzhled aplikace `Tabbix` je možné vidět na obr. 4.4. Aplikaci `Tabbix` si není možné stáhnout na telefony s `iOS`.



Obrázek 4.4: Mobilní aplikace PRTG for Android
Zdroj: https://play.google.com/store/apps/details?id=com.tirgil.tabbix

4.5 IciView

`IciView` [38] je jedna z nejnovějších aplikací dostupných pro nástroj `Icinga2`. Tato aplikace, která poprvé vyšla v roce 2016, stále dostává aktualizace. Aplikace vzhledově kopíruje styl webového rozhraní `Icingaweb2` (což je možné vidět na obrázku 4.5), nicméně dovoluje pouze podobné akce jako předchodí aplikace `GRMonitor` a `aNag`. Aplikace `IciView` je dostupná pouze pro `iOS` na `App Store` (17. března 2022 je její cena 129 Kč nebo 4,99 USD).



Obrázek 4.5: Mobilní aplikace IciView
Zdroj: https://www.mwallpapers.com/apps/ios-utilities/1141720590

4.6 Hodnocení dostupných mobilních aplikací

V rámci testování těchto mobilních aplikací byly aplikace staženy, následně prostudována jejich funkčnost, vzhled, uživatelská ovladatelnost a podpora důležitých funkcí, které může uživatel při správě nástroje z mobilního zařízení potřebovat. Z těchto kritérií byly vytvořeny body, jež byly zpracovány do tabulky 4.1. Parametry aplikace byly hodnoceny autorem této práce na bodové škále 1-5, kde 5 je nejvyšší hodnocení. Za dostupnost na platformě `Android` a `iOS` bylo k hodnocení přidáno 5 bodů za každou platformu. Některé z aplikací též disponují zasíláním notifikací. V případě, že aplikace umí posílat `lokální notifikace`, získává 5 bodů, za `push notifikace` 10 bodů. Takto vysoké ohodnocení notifikací je zdůvodněno tím, že notifikace jsou při monitorování kritické a uživatel musí být o výpadcích v síti in-

formován. V případě, že je aplikace `placená`, byly od hodnocení odečteny 2 body. Z důvodu, že u některých aplikací nebyla možnost zjistit `uživatelské hodnocení`, je tato položka v tabulce pouze orientační.

Mobilní aplikace	aNag	GRMonitor	PRTG	IciView	Tabbix
Android	Ano	Ano	Ano	Ne	Ano
iOS	Ne	Ano	Ano	Ano	Ne
Uživatelské hodnocení	4.4/5	NaN	4.3/5	NaN	NaN
Vzhled	2	3	5	5	5
Placené	Ne	Ne	Ne	Ano	Ne
Limitace	Icinga2/Nagios	Icinga2	PRTG	Icinga2	Zabbix
Snadnost použití	2	3	4	4	4
Uživatelské rozhraní	3	2	4	4	5
Podpora notifikací	Lokální	Ne	Push	Lokální	Push
Body	17	18	33	21	29
Pořadí	5.	4.	1.	3.	2.

Tabulka 4.1: Srovnání dostupných mobilních aplikací

Porovnáním mobilních aplikací vzešlo několik poznatků. Existuje velké množství aplikací, které podporují širokou škálu monitorovacích systémů. V obou obchodech je možné si stáhnout aplikaci, která uživateli pomůže s monitorováním sítě. Nicméně současně dostupné aplikace pro monitorovací nástroj `Icinga2`, který byl v tomto výzkumu označen v kapitole 2.2 jako nejlepší monitorovací nástroj, jsou v nízké kvalitě. Ani jedna z prozkoumaných aplikací nesplňuje všechny vydefinované parametry. Aplikacím schází možnost stažení na obou platformách, jsou špatně přehledné, nebo placené, nebo schází implementace push notifikací.

Proto bude vyvinuta nová mobilní aplikace pro nástroj `Icinga2`, která bude tvořena jako multiplatformní a její součástí budou push notifikace.

Z průzkumu je vytvořeno několik bodů, které musí nová mobilní aplikace splňovat:

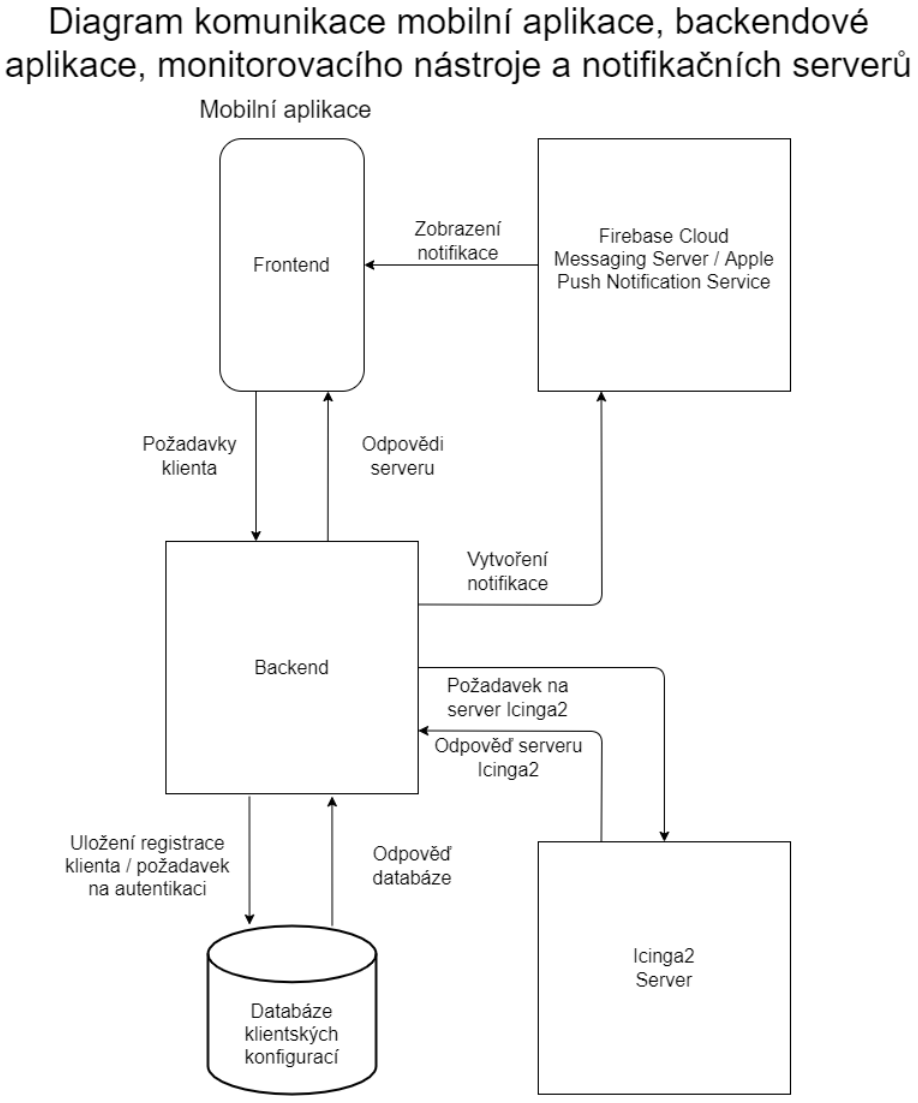
- Aplikace bude zobrazovat monitorované služby a servery, jejich stav a podrobnější informace o nich.
- Aplikace bude podporovat vyhledávání a filtrování služeb a serverů.
- Aplikace bude implementovat push notifikace.
- V aplikaci bude možnost nastavení, jaký typ notifikací bude uživatel dostávat.

- Notifikace budou moci být ztlumeny v časovém rozmezí, nebo je bude uživatel moci úplně vypnout.
- Aplikace bude přehledná a uživatelsky přívětivá.
- Uživatelské rozhraní bude jednoduše orientovatelné a intuitivní.
- Aplikace může být používána na zařízeních s OS Android i iOS.

5 Architektura programu

Práce je rozdělena na dvě části. První je frontend, mobilní aplikace blíže popsaná v kapitole 6, jež zobrazuje a interpretuje data posílaná z monitorovacího systému. Druhou částí práce je backend popsaný v kapitole 7, skrz nějž mobilní aplikace s monitorovacím systémem komunikuje. Backend byl vytvořen zejména pro účel posílání push notifikací. Bez něj by nebylo možné push notifikace odesílat, jak bylo popsáno v kapitole 3.3.

Na obrázku 5.1 je znázorněn obecný model komunikace mezi jednotlivými komponentami projektu.



Obrázek 5.1: Diagram komunikace mezi komponentami

6 Frontend

Pro implementaci frontendu bylo potřeba vybrat framework, ve kterém bude aplikace vytvořena. V důsledku předchozího zkoumání v kapitole 3.2.3 byl vybrán framework `React Native`, který se zdá jako nejlepší volbou pro tento typ aplikace. Aplikace v `React Native` se, jak již bylo popsáno v kapitole 3.2.2, programuje v jazyce `JavaScript`. Pro lepší přehlednost jsem se rozhodl použít jazyk `TypeScript` [66], který je nadstavbou `JavaScriptu`, a který přidává do `JavaScriptu` typování, které pomáhá programátorovi napovídáním a kontrolou typů ve fázi vývoje a debugování.

V aplikaci jsou využity komponenty, které jsou dostupné ve frameworku `React Native`, dále pak knihovní komponenty a komponenty, které vznikly úpravou předchozích dvou druhů. Tyto komponenty jsou opakovaně použity na několika obrazovkách, jež jsou propojeny pomocí navigace.

Uživatelské rozhraní mobilní aplikace je napsáno v anglickém jazyce. Výhodou použití anglického jazyka v mobilní aplikaci je její širší dosah. Webové rozhraní `Icingaweb2` je též v angličtině, uživateli, který již tedy použil `Icingaweb2` by neměl dělat problém přechod na mobilní aplikaci. Při návrhu nebyl brán ohled na multijazyčnost aplikace, není tedy v této aplikaci implementována.

6.1 Použité knihovny

V této kapitole jsou popsány nejdůležitější knihovny použité v mobilní aplikaci. Je zde popsána jejich funkcionalita a proč a v jakých případech je použita.

6.1.1 Axios

`Axios` [9] je klient pro HTTP [32] (Hypertext Transfer Protocol) komunikaci založený na tzv. příslibech [52]. Tento klient se může používat namísto funkce `fetch`, která je obsažena již v základním `React Native` [70]. Přísliby jsou objekty, které představují eventuální dokončení asynchronní operace a její výslednou hodnotu. Přísliby jsou běžně využívaná praxe v síťové komunikaci, neboť existuje nějaká doba mezi odesláním, zpracováním a přijetím požadavku ze vzdáleného zařízení. Zatímco funkce čekající na odpověď je pozastavena, může zbytek aplikace normálně pracovat, přičemž až přijde

odpověď od serveru, data se zpracují a funkce může pokračovat ve vykonávání svého kódu.

V aplikaci je `Axios` použit pro komunikaci s backendem. Mobilní klient se po zadání konfigurace zaregistruje na server pomocí tokenu získaného z knihovny `Expo` popsané v kapitole 6.2.2. Tímto tokenem se při každém volání `API` identifikuje, server klientovi následně odešle odpověď.

6.1.2 NativeBase

Namísto tvorby vlastních designových komponent (tj. tlačítek, seznamů, textových polí, aj.), byla při tvorbě použita knihovna `NativeBase` [60]. V této knihovně existuje sada předdefinovaných komponent, které jsou dostupné pro `Android` i `iOS`, a mění svůj vzhled podle typu `OS`. Použití této knihovny velice zlehčuje proces vývoje aplikace, neboť stylizace jednotlivých komponent od počátku může být velice náročná.

6.1.3 Moment

`Moment` [51] je knihovna, která slouží k manipulaci s časem a daty. `Icinga2` vyžaduje v některých ze svých HTTP požadavků proměnnou ve tvaru `timestamp`, což je údaj v sekundách o čase uplynulém od 00:00:00 1. ledna 1970. Vzhledem k tomu, že tento údaj je ve tvaru celého čísla, není pro člověka dobře čitelný. Knihovna `Moment` proto převádí údaje o datu na `timestamp`, který je pak použit v požadavku pro `Icinga2 Api` [33].

6.1.4 BigList

Jde o knihovnu, která je používána pro zobrazování velkého množství dat. Je použita namísto nativního `FlatListu` z `React Native`, který je několikanásobně pomalejší [76]. Tato komponenta je pro zobrazování dat kritická, neboť `FlatList` nedokáže vysoké množství objektů vykreslovat tak efektivně, jak `BigList`.

6.1.5 React Native Reanimated

Podpůrná knihovna, která přidává některým komponentám přechody pro příjemnější animace [77].

6.1.6 AsyncStorage

Důležitá je i knihovna `AsyncStorage` [75], která byť je knihovnou obsaženou v samotném frameworku `React Native`, je pro funkčnost této aplikace velice důležitá. `AsyncStorage` je perzistentní key-value úložiště (úložiště na disku v telefonu, do kterého se data přiřazují stylem `název: hodnota` podobně jako v JSON (JavaScript Object Notation) objektech). Díky tomuto uložišti je možné ukládat data o uživatelsky nastavené konfiguraci do paměti mobilního zařízení. Tato data pak vydrží v telefonu i po vypnutí aplikace a při přístím spuštění je možné je znovu získat a nahrát do aplikace.

6.2 Použité technologie

Při vytváření mobilních aplikací se mohou využívat i další technologie kromě frameworku samotného. Díky těmto technologiím je možné implementovat nové funkce, jako např. push notifikace v případě `Firebase`, nebo zjednodušují samotný vývoj a následný deployment aplikace. Tyto technologie jsou popsány v následujících kapitolách.

6.2.1 Firebase

`Firebase` [25] je technologie vyvíjená technologickou firmou Google. Jedná se o produkt, který vývojářům umožňuje provádět analytiku mobilních aplikací a jejich případnou monetizaci. Také nabízí nejdůležitější službu pro tuto aplikaci, a to možnost posílat mobilnímu zařízení push notifikace.

Aby mohla aplikace využívat těchto služeb, je potřeba aplikaci na `Firebase` zaregistrovat. Při registraci je třeba přidat doménu a název aplikace. Po vyplnění parametrů je potřeba stáhnout soubor, který vývojář vloží do balíčku aplikace. Následně je potřeba do souboru `app.json` přidat informaci o tomto souboru. `App.json` se nachází v adresáři `frontend`, stažený soubor by se měl nalézat uvnitř kořenového adresáře projektu. Poté už může mobilní aplikace zaregistrovat zařízení pro obdržování notifikací.

Aplikace pro odesílání notifikací využívá `Firebase` a `Apple Push Notification Service` ve spolupráci s platformou `Expo`, která zjednodušuje odesílání notifikací.

`Firebase` dovoluje odesílat notifikace pro `Android` i `iOS`, nicméně odesílání notifikací přes `APNs` je lepší variantou, neboť notifikace využívají přímo servery Apple.

Pro `Android` je nicméně potřeba využít `Firebase`. Soubor, ve kterém se nachází `API` klíč se jmenuje `google_play_api_key.json`. Bez tohoto sou-

boru není možné získávat push notifikace. Informace o existenci tohoto souboru musí být vloženy do projektu staticky, neboť data z něj jsou získávány při kompilaci projektu. Z toho vyplývá, že všechny notifikace budou odesílány přes účet tvůrce projektu, přidání vlastního klíče nahráním souboru uvnitř aplikace není možné. Pokud by si chtěl kdokoliv vytvořit vlastní balíček z tohoto projektu a nebyl by tam tento soubor, notifikace by nebyly možné obdržet (což je až druhotný problém, v první řadě by bylo při překladu uživateli oznámeno, že tento soubor neexistuje, a proto není možné tento projekt přeložit).

Kterýkoliv autorizační a autentizační klíč, který je veřejně dostupný, znamená možné riziko zneužití. Z podstaty bakalářské práce ale vyplývá, že je veřejně dostupná. Proto, aby si kdokoliv mohl tento projekt zkusit přeložit a spustit, bylo rozhodnuto, že tento soubor obsahující klíč zůstane přístupný.

6.2.2 Expo

`Expo` [17] je nástroj, který zjednodušuje a standardizuje vývoj aplikací ve frameworku `React Native`. Slouží k překládání aplikace, přičemž si vývojář může na svém zařízení spustit vývojářský server, který zkompiluje kód naprogramovaný v `React Native` a vytvoří balíček dostupný v mobilní aplikaci `Expo Go`.

V této mobilní aplikaci se zobrazí všechny spuštěné `Expo` servery na současné síti, vývojář si v aplikaci vybere svůj balíček, který se stáhne do mobilního zařízení. Poté může vývojář aplikaci zkoušet.

Expo zároveň umí vytvářet balíčky, které je následně možné instalovat do ostatních mobilních zařízení. Dovoluje vytvářet balíčky pro chytré telefony s `OS Android` i `iOS`, což je pro splnění požadavků této práce nezbytné.

Kromě toho, že `Expo` [17] je nástroj pro vývoj mobilních aplikací, dovoluje vývojářovi také použít `Expo` pro implementaci notifikací. Do projektu se přidávají jednotlivé části `Expo` jako knihovny. Aby mohl chytrý telefon přijímat notifikace, musí mít přiřazený jedinečný token. Ten je získán od `Expo` serveru pomocí funkce `getExpoPushTokenAsync`, která se pokusí získat expoPushToken, pomocí něhož se následně mobilní zařízení identifikuje. Získávání tokenu je automatizováno uvnitř aplikace, uživatel nemusí provádět žádnou akci.

6.2.3 React Native Hooks a Contexts

Do `React` a `React Native` byly přidány nové funkce, které obsluhují stavy nejen jednotlivých komponent, ale i celé aplikace. Dříve bylo uchovávání a

předávání těchto stavů mnohem obtížnější, proto jsou tyto funkce představeny v následujících kapitolách.

React Native Hooks

Jednou z hlavních funkcionalit frameworku `React Native` jsou tzv. `hooks` [71]. `Hooks` jsou skupina funkcí, jež zlehčují práci se stavy komponent. `Hooks` jsou posledním pokusem jazyka `React Native` o ultimátní znovupoužitelnost logické části programu.

K těmto stavům lze pak přistupovat na úrovni jedné komponenty, jiné hooky pak lze importovat do mnoha komponent a používat je ve všech z nich. Na těchto funkcích stojí předávání dat mezi komponentami i v této práci. Na obrázku 6.1 je jednoduchý příklad využití hooks v praxi.

```
import React, { useState, useEffect } from 'react';

function Example() {
  const [count, setCount] = useState(0);

  // Similar to componentDidMount and componentDidUpdate:
  useEffect(() => {
    // Update the document title using the browser API
    document.title = `You clicked ${count} times`;
  });

  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={() => setCount(count + 1)}>
        Click me
      </button>
    </div>
  );
}
```

Obrázek 6.1: Příklad useState a useEffect hooku
Zdroj: https://reactjs.org/docs/hooks-effect.html

V příkladu jsou použity funkce `useState` [74] a `useEffect` [73], přičemž `useState` v příkadu má počáteční hodnotu deklarovanou na 0. Funkce vrací dvojici hodnot; současný stav (`count`) a funkci (`setCount`), jež tento stav mění. Připodobněním těchto hodnot jsou getry a setry, jež se používají například v jazyce `Java`. V obsluze události `onClick` komponenty `Button` v příkladu bylo naprogramováno zvětšení stavu `count` o 1 pokaždé, když na tlačítko uživatel klikne. Toho bylo dosaženo pomocí funkce `setCount`, která zjistí aktuální stav `count` a přičte k němu 1. Pokaždé, když se změní hodnota, aktualizuje se `DOM`, komponenta se znovu vykreslí a funkce `useEffect` provede efekt, který jí byl přidělen. V tomto případě změní titulek dokumentu. Tato funkcionalita je zde zmíněna neboť jde o velice důležitou

součást programu a bez této funkcionality by se tato aplikace programovala mnohonásobně složitěji.

React Native Contexts

Kontexty [72] jsou principem, který pro své fungování využívá `React Native Hooks`.

Motivací pro využití kontextů v aplikaci je udržení a možnost změny důležitých informací o stavu aplikace napříč různými komponentami a obrazovkami.

Při klasickém programování se předávají data shora dolů pomocí parametrů jednotlivých funkcí, při předávání dat komponentám se data předávají skrz parametr props. Tato data pak mohou být využívána v dané komponentě.

Kontexty vytváří k tomuto principu alternativu. Programátor vytvoří objekt poskytovatele kontextu, tím pak obalí základní komponentu. Všechny komponenty obalené kontextem mohou následně využívat data a funkce, jež jsou deklarovány v kontextu.

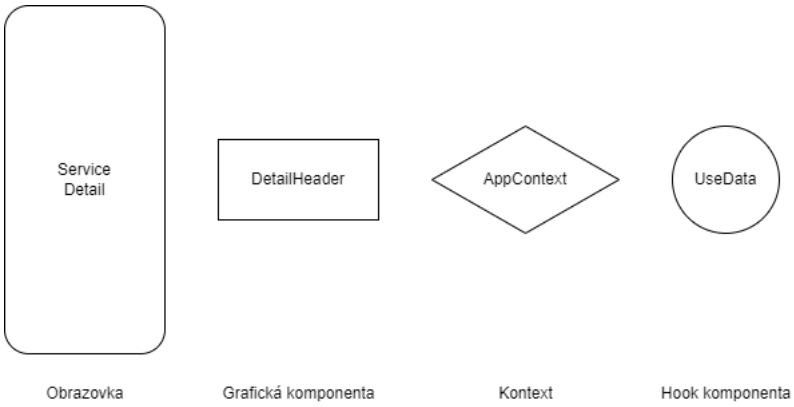
6.3 Obrazovky

Hlavní náplní tvorby aplikace jsou obrazovky. Mezi obrazovkami lze přecházet pomocí navigace, která se nachází na každé obrazovce, nebo prokliknutím služeb a serverů, které tato aplikace v některých z obrazovek vykresluje. Obrazovky využívají komponenty popsané v kapitole 6.6 a funkce popsané v kapitolách 6.4 a 6.5.

Obrazovky byly navrženy tak, aby uživatel mohl provádět základní důležité operace, které by mohl potřebovat provádět okamžitě na místě, aplikace neimplementuje všechny funkcionality webového rozhraní `Icingaweb2`. Pro uživatele je u mobilní aplikace důležitá její přehlednost a rychlost.

Navigace mezi jednotlivými obrazovkami je popsána v kapitole 6.6.5.

Pro lehčí zorientování se mezi jednotlivými obrázky byla vytvořena legenda na obrázku 6.2. Obrázky obsahující tyto objekty popisují využití jednotlivých komponent a funkcí na jednotlivých obrazovkách.



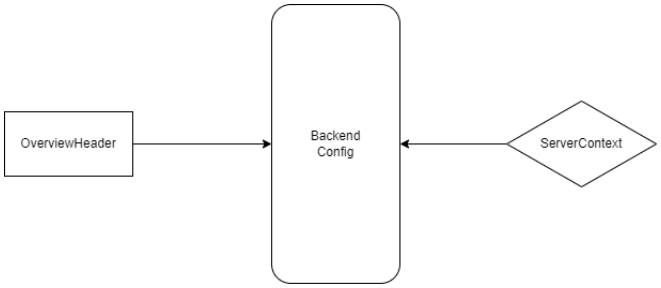
Obrázek 6.2: Legenda ke komponentám a funkcím

6.3.1 BackendConfigScreen

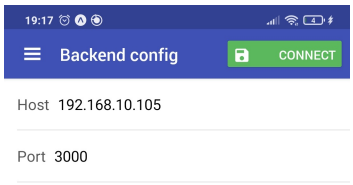
Na obrazovce `BackendConfigScreen` si uživatel vybere backendový server, pomocí něhož chce získávat data z `Icinga2` a z něhož budou odesílány notifikace. Na obrazovce zadá uživatel pouze IP adresu a port, na kterém aplikace běží a následně stiskne tlačítko `Connect`. Aplikace poté provede kontrolu toho, že je konfigurace validní. Pokud validní je, aplikace se sama restartuje, aby se instantně propsaly změny do všech ostatním komponent v aplikaci. Pokud konfigurace validní není, je uživatel o tomto problému informován. Backend přijme požadavky každého uživatele. Zabepečení nebylo v této práci řešeno.

Obrazovka využívá komponentu `OverviewHeader` a kontext `ServerContext` pro ukládání akutální konfigurace.

Využití komponent a vzhled obrazovky je možné vidět na obrázcích 6.3 a 6.4.



Obrázek 6.3: Diagram využití komponent obrazovky BackendConfig



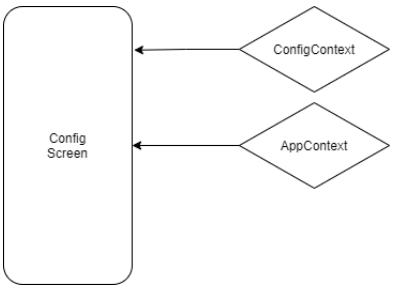
Obrázek 6.4: Obrazovka BackendConfig

6.3.2 ConfigScreen

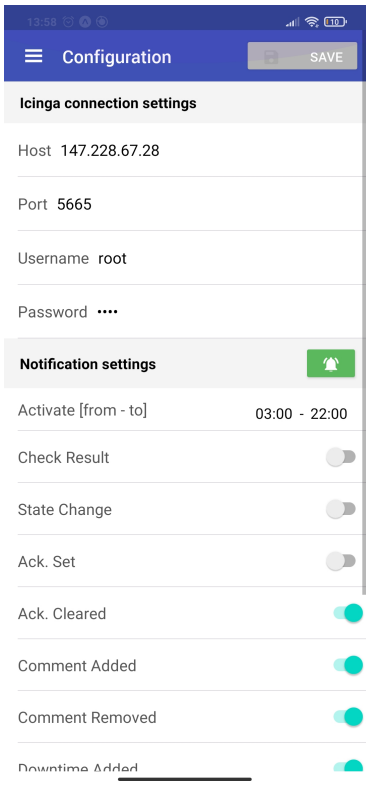
Obrazovku `ConfigScreen` uvidí uživatel poté, co nastaví konfiguraci backendu. Motivací pro rozdělení obrazovek na dvě namísto jedné konfigurační obrazovky je to, že jedna obrazovka by měla vykonávat jednu činnost. Tím, že se jedná o 2 rozdílné konfigurace, by uživatel mohl být zmatený a zadávat údaje tam, kam nepatří. Navíc je na obrazovce `ConfigScreen` velké množství údajů a nemusela by být přehlednou. Na této obrazovce zadává uživatel údaje o uživateli `ApiUser` z monitorovacího nástroje `Icinga2`. Je potřeba znát `IP adresu, port` nástroje Icinga2 a `jméno a heslo ApiUsera`. Dále je na této obrazovce možné nastavit typy notifikací, které chce uživatel získávat. Poslední možností, co může uživatel změnit je, kdy tyto notifikace chce získávat. Tyto informace se následně pošlou backendu, který informace zpracuje a bude se podle nich chovat. V případě, kdy vznikne událost v čase, který nespadá do intervalu vymezeného uživatelem, není backendem notifikace odeslána a uživatel tak není informován.

K nastavení notifikací patří i tlačítko se zvonkem. Pomocí tohoto tlačítka může uživatel okamžitě vypnout všechny notifikace. Po jeho zmáčknutí se notifikace začnou opět posílat podle nastavení uloženém v backendové databázi.

Obrazovka `ConfigScreen`, která se nachází na obrázku 6.6, využívá kontexty `ConfigContext` a `AppContext`, jak je vidět na obrázku 6.5.



Obrázek 6.5: Diagram využití komponent obrazovky ConfigScreen

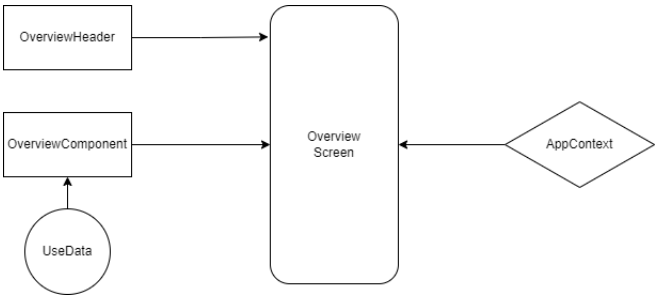


Obrázek 6.6: Obrazovka ConfigScreen

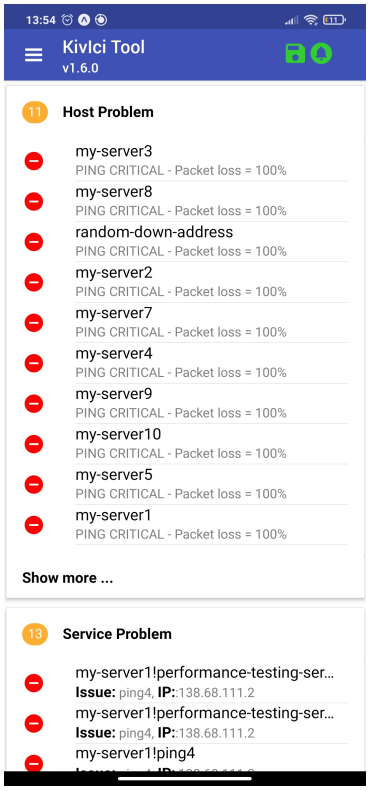
6.3.3 OverviewScreen

Když je aplikace spuštěna a uživatel v předchozím běhu aplikace nastavil platnou konfiguraci, zobrazí se uživateli obrazovka `OverviewScreen`. Na této obrazovce jsou zobrazeny důležité informace o tom, zda je aplikace přihlášena k odebírání informací z backendu, a zdali je uživatel registrován k přijímání push notifikací.

Pro zobrazování informací o stavu objektů je zde využita komponenta `OverviewComponent` popsaná v kapitole 6.6.1. Dále obrazovka využívá `AppContext` a `OverviewHeader`, jak je vyobrazeno na obr. 6.7. Vzhled obrazovky je možné vidět na obr. 6.8.



Obrázek 6.7: Diagram využití komponent obrazovky OverviewScreen

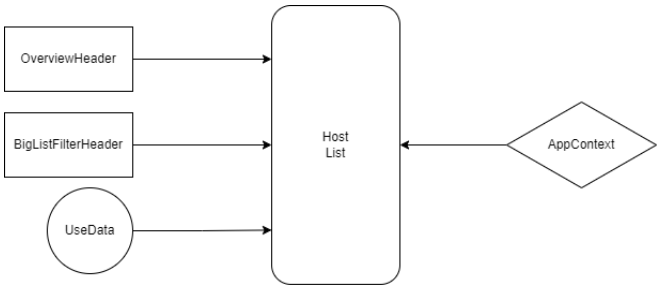


Obrázek 6.8: Obrazovka OverviewScreen

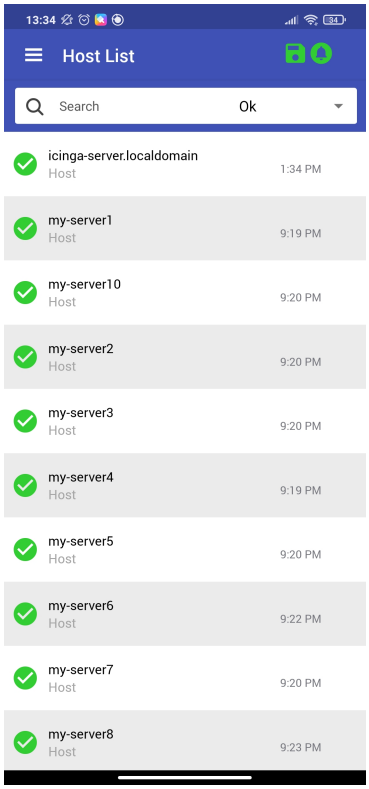
6.3.4 HostList

Tato obrazovka (na obr. 6.10) zobrazuje přehled všech serverů, které uživatel `Icinga2` nastavil pro monitorování. Servery je možné vyhledávat pomocí vyhledávacího pole, zároveň je možné servery filtrovat podle jejich aktuálního stavu. Servery je možné si rozkliknout, což uživatele přesune na obrazovku `HostDetail`.

`HostList` využívá komponenty `OverviewHeader`, `BigListFilterHeader`, kontext `AppContext` a hook `useData` (na obr. 6.9).



Obrázek 6.9: Diagram využití komponent obrazovky HostList



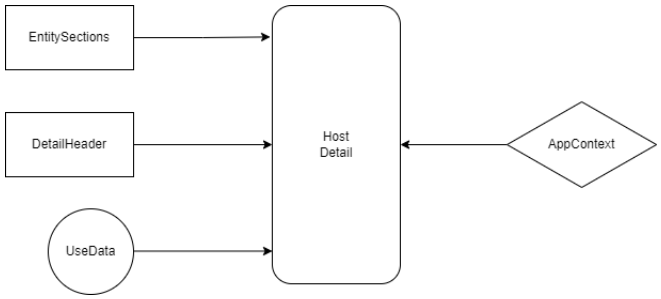
Obrázek 6.10: Obrazovka HostList

6.3.5 HostDetail

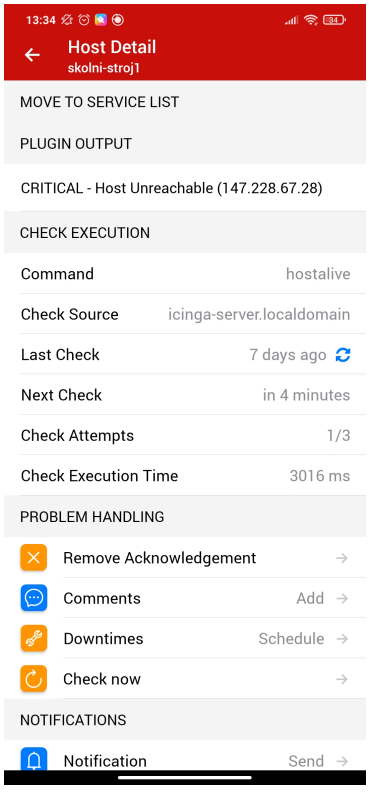
Na této obrazovce jsou zobrazeny podrobné informace o stavu monitorovaného serveru. Toho je docíleno pomocí zobrazování komponent, které byly naprogramovány. Využívá se zde komponenta `useData` ze skupiny funkcí `Hooks`. Této komponentě je předána URL (Uniform Resource Locator) na backend `API`, která za mobilní aplikaci provede požadvek na získání dat z `Icinga2 API`. Předáván je i parametr `refreshInterval`, který zajišťuje to, že data budou aplikaci předávána pravidelně a zůstanou aktualizována bez nutnosti uživatele manuálně aktualizovat data. `RefreshInterval` zde i v

jiných částech programu je nastaven na 30 sekund, uvnitř aplikace si ho uživatel nemůže změnit. Na této obrazovce může uživatel provést několik akcí. Uživatel může provést manuální recheck, nebo vybrat jednu z akcí podporovaných `Icinga2 API`, což uživatele přesměruje podle typu požadavku na obrazovku `ModalForm`, ve které uživatel vyplní údaje požadované k vykonání požadavku.

Obrazovka využívá komponenty `EntitySections`, `DetailHeader`, kontext `AppContext` a hook `useData` (vyobrazeno na obr. 6.11).



Obrázek 6.11: Diagram využití komponent obrazovky ServiceList



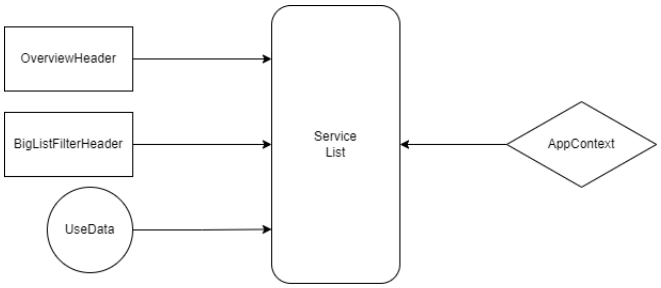
Obrázek 6.12: Obrazovka HostDetail

Z této obrazovky také můžeme přejít přímo na obrazovku `ServiceList`, ve které jsou vyfiltrovány služby přiřazené k tomuto serveru.

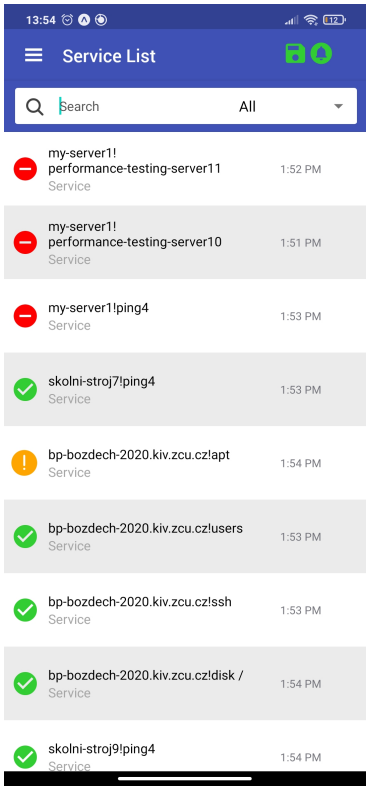
6.3.6 ServiceList

Na každém serveru může být spuštěno více služeb, které mohou být monitorovány, uživatel je může přiřadit k monitorování. Tyto služby jsou následně zobrazeny zde. Služby mohou být stejně jako servery, filtrovány, vyhledávány mohou být pomocí názvu rodičovského serveru.

Obrazovka ServiceList využívá stejné komponenty jako obrazovka HostList (využití je možné vidět na obr. 6.13). I vzhled obrazovky je proto stejný (obr. 6.14).



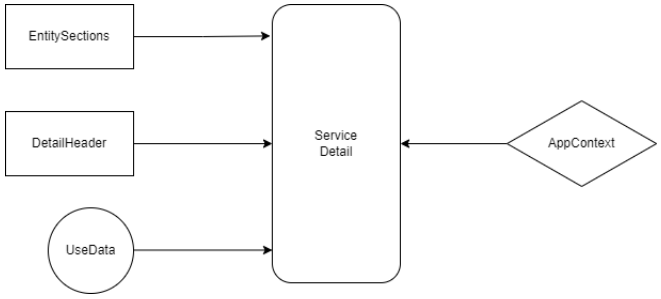
Obrázek 6.13: Diagram využití komponent obrazovky ServiceList



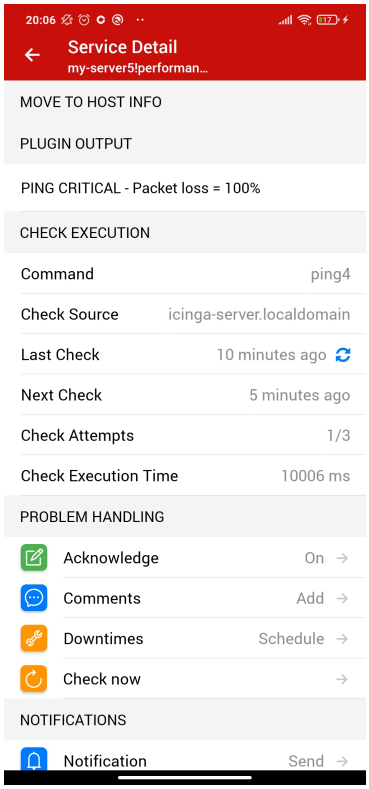
Obrázek 6.14: Obrazovka ServiceList

6.3.7 ServiceDetail

Díky modularitě tohoto programu mohly být použity pro vytvoření této obrazovky stejné komponenty, jako u obrazovky `HostDetail` (využití komponent je na obr. 6.15). Rozdílem je pouze, že funkce `useData` využívá pro získávání dat jiný API endpoint.



Obrázek 6.15: Diagram využití komponent obrazovky ServiceList



Obrázek 6.16: Obrazovka ServiceDetail

Obdobně jako u obrazovky `HostDetail`, i zde se nacházejí podrobnější informace o stavu služby. Můžeme provést stejné akce, a také přejít přímo na informace o stavu serveru, na kterém je služba spuštěná.

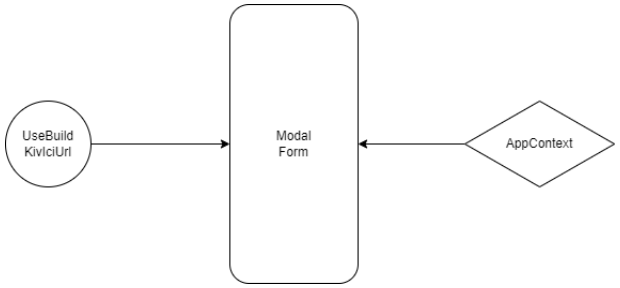
6.3.8 ModalForm

V aplikaci byly implementovány `API` požadavky na obsluhu jednotlivých událostí v monitorovacím nástroji `Icinga2`. Tyto požadavky obsahují požadované i volitelné parametry, které uživatel musí vyplnit, aby se mohl požadavek provést. Vzhledem k tomu, že těchto požadavků je několik, byla naprogramována jedna obrazovka, na které se vykreslují komponenty podle toho, jaký druh požadavku chce uživatel provést. To je provedeno pomocí předávání parametrů z navigace. Podle toho, jakou akci si uživatel vybral, je předáno pole objektů. Každý objekt obsahuje unikátní klíč k vykreslení komponenty, popisek, který se vykreslí u komponenty, její typ a popřípadě další parametry v závislosti na jejím typu.

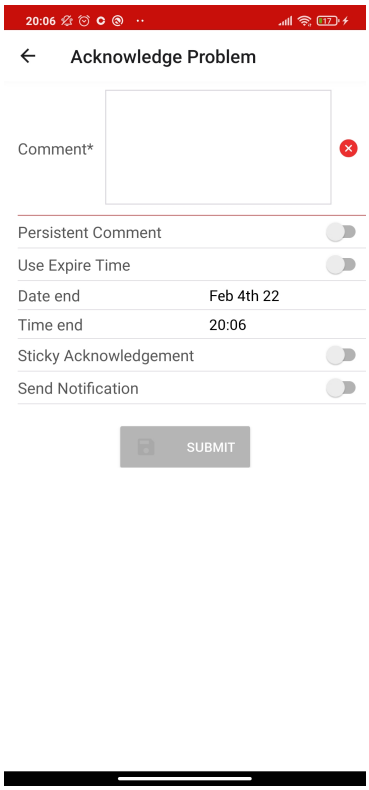
Uživatel následně vyplní požadované parametry pro odeslání požadavku, ten je následně odeslán ke zpracování backendu, který tento požadavek přepošle službě `Icinga2`.

`ModalForm` využívá pouze kontextu `AppContext`, a to z toho důvodu, že veškeré ostatní parametry potřebné pro vykreslení této obrazovky dostává z navigace (obr. 6.17).

Na obrázku 6.18 je možné vidět formulář `Acknowledge Problem`, který byl vykreslený pomocí obrazovky `ModalForm`.



Obrázek 6.17: Diagram využití komponent obrazovky ModalForm



Obrázek 6.18: Obrazovka ModalForm

6.4 Hooks

`Hooks` funkce jsou používány v mnoha částech této aplikace, zejména pak v komponentách obrazovek. Díky tomu, že jsou tyto funkce naprogamovány obecně, se zvýšila jejich znovupoužitelnost, nemusí se opakovaně kopírovat do každé komponenty.

`Hooks` je tato skupina funkcí pojmenována právě z důvodu, že je zde využito více hooks najednou. V některých případech vývojáři nestačí použití jednoho hooku, proto `React Native` nabízí možnost seskupování těchto hooků do funkcí, které lze pouze importovat do libovolné části aplikace. Tato kapitola popisuje hooky, jež byly implementovány v tomto programu.

Pokud nebude řečeno jinak, v kapitolách popisující implementaci jednotlivých komponent bude vždy mluveno o jednotlivých `React Native Hooks`, v opačném případě bude napsáno, že se jedná o funkci spadající do skupiny `Hooks`.

6.4.1 UseBuildKivIciUrl

Tato funkce získává z kontextu `ServerContext` data o aktuálním nastavení backendové aplikace, která se nastavuje na obrazovce `BackendConfigScreen`. Z těchto dat je následně ve funkci `buildKivIciUrl` vytvořena URL, která se následně v ostatních komponentách používá pro získávání dat pomocí funkce `useData`. Funkce vrací jako návratovou hodnotu objekt, ve kterém je dostupná právě funkce `buildKivIciUrl`.

6.4.2 UseData

Funkce `useData` je používána k získání dat o službách, serverech, jejich stavech a dalších podrobných informacích, které jsou důležité při monitorování a obsluze jednotlivých serverů a služeb. Do funkce se vždy musí jako objekt vkládat `url`, endpoint backendu, a objekt `dtoIn`, který v sobě má `expoPushToken`, kterým se zařízení identifikuje. Volitelným parametrem je `refreshInterval` v ms.

Pro funkci `useData` je důležitá funkce `buildKivIciUrl` z předchozího hooku, která vytváří celou `url`, na kterou je pak odesílán požadavek.

V obrazovce, která využívá funkci `useData` je automaticky po načtení obrazovky vyvolán hook `useEffect` naprogramovaný uvnitř této funkce, který zavolá funkci `fetchData`, jež se stará o získávání dat z backendu pomocí klienta `Axios`. V případě, že nemohou být data získána, je na obrazovce zobrazena errorová zpráva s problémem. V opačném případě jsou data přijata a do hooku `useState data` jsou vložena data pomocí `setData`. Zároveň s prvním zavoláním `fetchData` je spuštěn druhý hook `useEffect`, který v případě, že je do funkce `useData` vložen parametr `refreshInterval`, spustí funkci `setTimeout`, která slouží jako časovač. Po uběhnutí časového intervalu je opětovně spustěna funkce `fetchData`. Tímto je docíleno automatické aktualizace dat. Návratovou hodnotou funkce `useData` je objekt, který obsahuje proměnnou `isLoading`, která nabývá hodnoty `true`, nebo `false`, který vyjadřuje to, zdali jsou aktuálně získávána data z backendu. Této hodnoty je v obrazovkách využito pro rozhodování komponenty `Spinner`, která se zobrazuje do doby, než jsou k dispozici data k zobrazení. Návratový objekt dále obsahuje objekt `data`, jenž obsahuje data obdržená z backendu. Dále obsahuje funkci `fetchData`. Tato funkce je potřeba pro obrazovky `HostList` a `ServiceList`. Na těchto obrazovkách se nachází textový filtr, do kterého může uživatel zadávat názvy objektů. Hned po dopsání názvu je zavolána právě funkce `fetchData`. Poslední částí objektu je funkce `onRefresh`, která slouží k okamžitému znovunačtení obrazovky bez nutnosti čekat na automatické znovunačtení.

6.4.3 UseNotifications

`UseNotifications` je funkce, která se stará o to, že zařízení obdrží svoje unikátní ID, pomocí nejž komunikuje s backendem. Využívá se zde `AppContext`, do nejž se získané ID ukládá. Nejčastěji se jedná o `expoPushToken`. V případě, že se nejedná o fyzické zařízení, je vygenerováno jiné unikátní ID, pomocí nějž může virtuální zařízení komunikovat. Funkce využívá hook `useEffect`, kde je zjištěno, zda již aplikace své ID má, v případě že ne, je zavolána funkce `registerForPushNotificationsAsync`. V ní je zjištěno, zdali je aplikace spuštěna na fyzickém zařízení. Je-li aplikace spuštěna na virtuálním zařízení, emulátoru, pak je provedena funkce `uniqueID`, která vygeneruje náhodný řetězec, který následně slouží jako ID, jak je popsáno výše. Pro to, aby mohl být pro zařízení vygerenován expoPushToken, musí uživatel povolit aplikaci používat notifikace. Pokud aplikace zjistí, že nemá k notifikacím přístup, zažádá uživatele o jejich povolení. V případě, že uživatel udělí povolení k užívání, zažádá aplikace `Expo` server o expoPushToken. V případě, že uživatel odmítne povolení udělit, je vygerováno pro zařízení ID pomocí funkce `uniqueID`, uživatel ale ztrácí možnost získávat notifikace.

6.4.4 UseTokenRegistration

Tato funkce se stará o registraci mobilní aplikace k získávání dat z backendu. Jsou zde využity kontexty `ConfigContext`, odkud získává informace o konfiguraci v mobilní aplikaci, a `AppContext`, ve kterém nastavuje stav odběru dat z backendu. Díky hooku `useEffect` je všechno automatizováno na pozadí. Registrace na server je prováděna pomocí klienta `Axios`.

6.5 Contexts

Kontexty jsou v aplikaci využívány pro uchovávání informací, jež se používají napříč více obrazovkami. Díky kontextům se mohou získávat a nastavovat informace o stavu aplikace na více místech v aplikaci.

6.5.1 ServerContext

V tomto kontextu je udržována proměnná `serverConfig`, což je objekt obsahující data o aktuálně nastavené konfiguraci backendu.

6.5.2 AppContext

`AppContext` se stará o udržování dvou proměnných: `isSubscribed` a `expoPushToken`. Je k tomu využitý hook `useState`. V těchto proměnných je uložena informace o tom, zdali je zařízení registrováno na backendu k získávání informací od serveru nakonfigurovaném v nastavení této aplikace. Proměnná `expoPushToken` se stará o udržování tokenu používaného ke komunikaci.

Jedná se o globální kontext celé aplikace. Bez informací držených v tomto kontextu není možné, aby aplikace vykonávala akce týkající se komunikace se serverem.

Příklady informací, které by mohly do budoucna být v tomto kontextu uloženy, jsou například informace o grafickém nastavení aplikace (např. světlý, nebo tmavý režim), nebo informace o aktuální časové zóně.

6.5.3 ConfigContext

Kontext `ConfigContext` obsluhuje udržování a ukládání informací o konfiguraci serveru `Icinga2`. Jsou zde využity 2 hooky `useState`. `IsLoading` je proměnná, která rozhoduje o tom, co bude na obrazovce vykresleno. V případě, že je tato proměnná nastavena na `true`, je na obrazovkách vykreslena komponenta `Spinner` (točící se kolečko, která má uživateli značit, že se něco načítá). Poté, co jsou data získána, je tato hodnota nastavena na false a uživateli jsou místo `Spinneru` vykreslena získaná data. Druhou proměnnou je `config`, ve kterém je uložena informace o konfiguraci po dobu běhu aplikace. Pokaždé, když je aplikace spuštěna, je tento stav aktualizován z asynchronního úložiště, které je součástí knihovny `AsyncStorage` popsané v kapitole 6.1.6. Zároveň je stav změněn v případě, že uživatel zadá do aplikace jiné nastavení. Tento stav je okamžitě uložen do asynchronního úložiště.

6.6 Components

Z důvodu znovupoužitelnosti kódu byly některé komponenty, které jsou využívány na více místech programu, vloženy do samostatných souborů. Jedná se o hlavičky ostatních komponent nebo o navigaci, která je použita na každé obrazovce.

6.6.1 OverviewComponent

Komponenta využitá v obrazovce `OverviewScreen`. Tato komponenta je využita k zobrazování dat o službách a serverech, které hlásí problém, tj. nejsou

ve stavu 0 (stav 0 znamená u objektů monitorovaných nástrojem Icinga2, že jsou v pořádku). Pokud je problémových objektů více než 10, objeví se na konci seznamu tlačítko `Show more...`, díky kterému bude uživatel přesunut na obrazovku `HostList`, nebo `ServiceList`. Jednotlivé objekty je možné rozkliknout, touto akcí bude uživatel přesunut na obrazovku `HostDetail`, potažmo `ServiceDetail`.

Jak již bylo zmíněno výše, na obrazovce se objeví maximálně 10 objektů každého typu. Číslo objektů bylo takto vybráno proto, že při vykreslování více objektů na této obrazovce se rychle snižuje přehlednost. Zároveň na této obrazovce není možnost objekty filtrovat. Větší přehlednost a filtrování poskytují obrazovky `HostList` a `ServiceList`, které byly k těmto úkonům uzpůsobeny.

6.6.2 AppHeaders

V této části textu jsou popsány komponenty, jež jsou používány jako hlavičky.

DetailHeader

Tato hlavička je používána v obrazovkách `ServiceDetail` a `HostDetail`. Stará se o zobrazování názvu služby, nebo serveru, které si uživatel právě prohlíží. Použité komponenty využité v této hlavičce jsou získány z knihovny NativeBase.

OverviewHeader

Hlavička `OverviewHeader` je použita k zobrazení informace o tom, na jaké obrazovce se uživatel právě nachází. Je využívána na každé z obrazovek. Stejně jako `DetailHeader`, i tato hlavička využívá komponenty obsažené v knihovně `NativeBase`.

BigListFilterHeader

Tuto hlavičku používají obrazovky `HostList` a `ServiceList`. V této hlavičce se nachází tzv. `SearchBar`, do kterého může uživatel zadávat vyhledávací řetězec, podle kterého jsou následně objekty filtrovány. Dále se zde nachází komponenta `Picker`, která slouží pro výběr toho, jaké objekty se mají zobrazovat podle jejich stavu.

6.6.3 BigList

Pro zobrazování seznamu objektů zobrazovaných na obrazovkách `HostList` a `ServiceList` byla použita knihovna BigList popsanou v kapitole 6.1.4. Komponenty zde naprogramované jsou následně předávány knihovní komponentě `BigList`, která je následně vykreslí.

6.6.4 EntitySections

Komponenty které jsou obecně pojmenovány `EntitySections` jsou využívány na obrazovkách `HostDetail` a `ServiceDetail`. Tyto obrazovky jsou rozděleny do komponent z důvodu přehlednosti kódu a rozdělení informací podle skupin. Všechny tyto komponenty pro tvorbu `url` navíc využívají `UseBuildKivIciUrl`.

ExistingComments

Tato komponenta slouží z zobrazování komentářů, které jsou k jednotlivým objektům napsány. Využívá k tomu komponenty `useData`, která pravidelně v intervalu 30 vteřin posílá HTTP požadavky na backend. V případě, že se do mobilní aplikace vrátí informace o tom, že je k danému objektu vypsán text komentáře, uživatel, který komentář přidal a doba, po kterou už je komentář u objektu napsán.

Komentář je zároveň možné z objektu odstranit. Potom, co uživatel zmáčkne červené tlačítko, odešle se na backend požadavek, který komentář z nástroje `Icinga2` odstraní.

CheckExecution

V této komponentě jsou vypsány informace o kontrole monitorovaného objektu. Je zde vypsán příkaz, kterým byl objekt zkontrolován, následně pak název serveru, z kterého byla kontrola provedena, jak dlouho zpět byla kontrola provedena a za jak dlouho bude provedena znovu. Dále je uvedeno, kolik pokusů o kontrolu bylo provedeno. Pokud objekt odpoví normálně, je tato hodnota stále na 1/5. Ovšem v případě, kdy objekt neodpovídá, se toto číslo začne zvyšovat. Tato informace je důležitá v případě, že je s objektem něco v nepořádku, uživatel tak může sledovat, kolikrát, respektive jak dlouho už objekt neodpovídá na kontroly. Posledním zobrazovaným parametrem je doba, jak dlouho trvala kontrola.

Notifications

Tato komponenta replikuje funkci `Icingaweb2`, pomocí níž může uživatel poslat notifikaci s libovolným textem o monitorovaném objektu. Při použití této komponenty je uživatel mobilní aplikace přesměrován na obrazovku `ModalForm`, kde může vyplnit text notifikace.

PerformanceData

V této komponentně jsou zobrazována performance data obdržená od serveru `Icinga2`. Tato data jsou poměrně obtížně zpracovatelná, neboť server tato data odesílá pokaždé v jiném formátu. Data přichází jako jeden řetězec oddělený středníkem, data jsou rozdělena a následně zobrazena.

PluginOutput

V této komponentě jsou zobrazeny přesné informace o stavu objektu. V případě, že s monitorovaným objektem není něco v pořádku, právě v této komponentně se nachází text, který problém hlouběji popisuje. Proto se tato komponenta nachází na začátku obrazovek `HostDetail` a `ServiceDetail`.

ScheduledDowntimes

Obdobně jako v komponentě `ExistingComments`, i zde se zobrazují důležité informace o správě objektu. Informace v této komponentně je ale podstatně důležitější, neboť zobrazuje informace o automaticky nastavených nebo uživatelem vytvořených odstávkách serverů a služeb. Uživatel se tak touto informací může řídit například při zálohování důležitých dat.

Stejně jako v komponentně `ExistingComments` může uživatel informace o odstávkách odstranit.

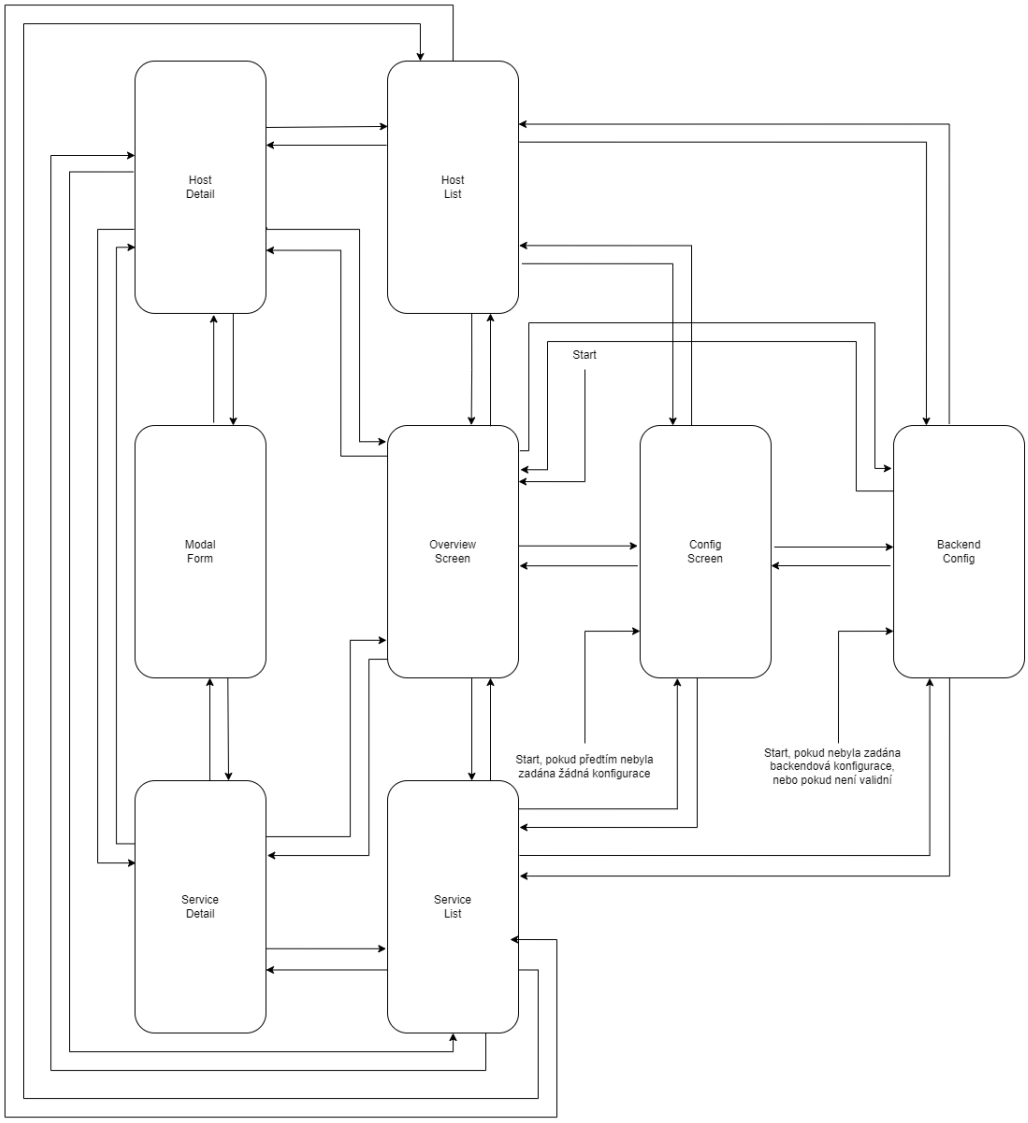
ProblemHandling

Jedná se o velice důležitou komponentu na obrazovkách `HostDetail` a `ServiceDetail`. V této sekci může uživatel provádět akce související se správou objektů. V případě, že s objektem není něco v pořádku, může uživatel přidat potvrzení o tom, že zaregistroval problém na serveru. Další akcí je přidání komentáře k objektu. Třetí volbou, kterou může uživatel udělat je přidat informaci o odstávce serveru, nebo služby. Všechny tyto akce provedené na mobilním zařízení přesměrují uživatele na obrazovku `ModalForm`, ve které vyplní formulář týkající se daného problému.

Uživatel také v této sekci najde tlačítko `Check now`, které odešle příkaz k okamžitému zkontrolování dostupnosti objektu. Poslední možností je odstranění potvrzení o problému, které se uživateli zobrazí pouze v případě, že byl problém již dříve někým potvrzen.

Tlačítka potvrzení a odstranění potvrzení viditelná na obrázcích 6.12 a 6.16 se uživateli nezobrazí v případě, že se jedná o objekt, který je v pořádku.

6.6.5 Router



Obrázek 6.19: Diagram pohybu mezi obrazovkami

Pohyb mezi obrazovkami uvnitř aplikace je realizován pomocí navigace. Komponenta, ve které je navigace realizována, se stará o rozhodování, jaký

typ navigace bude na které stránce zobrazen. Při prvním spuštění aplikace, kdy uživatel ještě nezadal žádnou konfiguraci serveru, může uživatel přistoupit pouze na obrazovku `BackendConfigScreen`. Poté, co uživatel zadá validní backendovou konfiguraci, jsou mu zpřístupněny obrazovky `OverviewScreen` a `ConfigScreen`, přičemž `ConfigScreen` je ta první, co uživatel uvidí. Následně je potřeba zadat konfiguraci pro nástroj `Icinga2`. Bez zadání této konfigurace se uživateli nezpřístupní zbytek aplikace.

V případě, že je aplikace vypnuta s validní backendovou konfigurací a při příštím startu aplikace tato konfigurace již validní není, je zbytek obrazovek znovu znepřístupněn a požaduje po uživateli novou konfiguraci.

Navigace je realizována pomocí tzv. `Draweru`, což je typ navigace, který se vysouvá (v tomto případě) z levé strany obrazovky.

V `Drawer` navigaci se může uživatel pohybovat libovolně mezi obrazovkami `ConfigScreen`, `BackendConfigScreen`, `OverviewScreen`, `HostList` a `ServiceList`.

Pokud si uživatel zobrazí obrazovky `ServiceDetail`, `HostDetail` nebo `ModalForm`, aplikace uživateli dovolí uživateli jít pouze na obrazovku, z které se na aktuální obrazovku dostal. To může provést pomocí gesta, které se v daném `OS` používá pro akci zpět, nebo pomocí šipky zpět, která se nachází v levé horní části obrazovky v případě, že se na obrazovku `ModalForm` uživatel dostane. To je realizováno pomocí `Stack` navigace.

Všechny pohyby mezi obrazovkami je možné si prohlédnout v diagramu na obrázku 6.19.

Z aplikace lze vystoupit na všech obrazovkách tím, že uživatel aplikaci vypne.

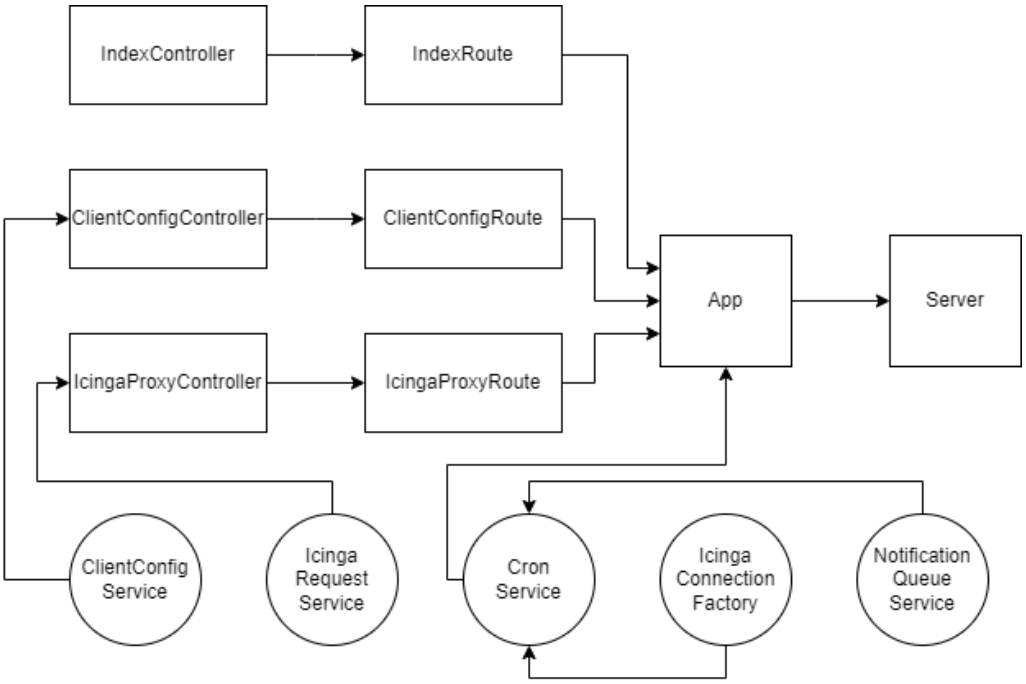
7 Backend

Hlavní motivací pro tvorbu backendu, která je pro tuto funkcionalitu kritická, je potřeba zasílání notifikací. V případě, že uživatel aplikaci úplně vypne, přestává aplikace mít možnost vykonávat úkony potřebné pro zaslání notifikace. Proto tato funkcionalita byla implementována na backendu, který je nezávislý na běhu mobilní aplikace.

Backend se stará o zpracování požadavků všech mobilních klientů. Informace o klientech si ukládá do databáze `MySQL` [53], do které přistupuje pomocí Object Relational Mapping [63] (zkráceně ORM) knihovny `Prisma` popsané v kapitole 7.3.2. Backend slouží jako proxy mezi mobilní aplikací a serverem, na kterém je spuštěn nástroj `Icinga2`. Z důvodů, které zahrnují totožnou syntaxi jazyků a možnost využití stejných knihoven, byl pro naprogramování této služby využit framework `Node.js`. Tento framework používá jazyk `JavaScript`, stejně jako frontend framework `React Native`. Pomocí `Node.js` se vytváří aplikace, jejichž výpočty se provádí na straně serveru a klientovi se posílá pouze odpověď.

`Node.js` byl vybraný jako framework z důvodu velké a nadále vzrůstající popularity jazyka `JavaScript`. `Node.js` je velice stabilní, komunitou podporovaný a vzniká mnoho knihoven, které se dají využít v této aplikaci, např. právě knihovna `Prisma`, jež se dá využít pro integraci s databází. Dalším důvodem pro využití tohoto frameworku před alternativami, jako jsou `Java` nebo `PHP` [66], je schopnost `Node.js` odbavit velké množství dotazů, což se u tohoto typu aplikace očekává.

Pro lepší představu architektury backendu byl vytvořen diagram, který zobrazuje využití jednotlivých částí uvnitř programu (na obr. 7.1).



Obrázek 7.1: Komponentový model backendové aplikace

7.1 Komunikace frontend-backend

Pro komunikaci mezi 2 webovými aplikacemi se využívá API. Tuto komunikaci je možné zprostředkovat několika technologiemi, které mezi sebou nesdílí žádnou standardizaci. Pro komunikaci je možné využít protokol `SOAP` [83], `GraphQL` [30] nebo `REST` [83]. Každá z těchto technoligí má své výhody a nevýhody, pro tuto práci je ale kvůli svým vlastnostem nejlepší `REST`.

Mobilní aplikace komunikuje s backendem pomocí vlastního `REST API`. Dokumentace k API endointům je dostupná v souboru `swagger.yaml`, jenž se nachází v adresáři `ts-backend`. Návod, jak si dokumentaci zobrazit, je popsán v kapitole 7.2.1.

Když uživatel otevře poprvé aplikaci, zadá na obrazovce `ConfigScreen` identifikační údaje `ApiUser Icinga2` a mobilní aplikace zažádá na pozadí službu [19] o `ExpoPushToken`. Informace o uživateli jsou uloženy do databáze `MySQL`, která je připojena k projektu. Do databáze je přistupováno pomocí ORM knihovny `Prisma`, která zprostředkovává pomocí svého klienta komunikaci mezi backendem a databází. Výhodou tohoto řešení je stabilita při pádu aplikace. Data jsou uložena v konzistentním stavu a v případě, že je potřeba restartovat server, se uživatelská nastavení opět načtou z databáze a aplikace může pokračovat ve vykonávání svých funkcí.

Aplikace posílá požadavky na jednotlivé `API` endpointy pomocí protokolu HTTP spolu s daty potřebnými na `Icinga2 API` endpointu. Do požadavku je také přidán `expoPushToken`, pomocí kterého je každý uživatel identifikován. Šifrování komunikace mezi mobilní aplikací a backendem nebylo v této práci řešeno. Backend odešle požadavek na `Icinga2 API` a následně přepošle odpověď koncovému zařízení, které data zpracuje a zobrazí uživateli. Notifikace jsou obsluhovány pomocí cronu [11], který opakovaně odesílá požadavky na `Icinga2` backend. Interval pro odesílání požadavků je implicitně nastaven v aplikaci na 1 minutu. V případě, že dostane odpověď, backend notifikaci transformuje a následně odešle chytrému telefonu notifikaci o události na `Icinga2` backendu.

7.2 Implementace

V této kapitole jsou popsány třídy, které jsou na backendu implementovány. Třídy jsou rozděleny do podkapitol podle toho, jakou funkci každá třída vykonává. Jedná se o třídy, ve kterých jsou definovány `API` endpointy, na než se dotazuje mobilní aplikace, třídy implementující zpracování `API` dotazu, ale i třídy podpůrné, které transformují řetězce nebo odesílají notifikace mobilnímu telefonu.

7.2.1 Routes

Routes třídy se jmenují `ClientConfigRoute`, `ProxyRoute` a `IndexRoute`. Ve třídě `ClientConfigRoute` jsou implementovány 2 `API` endpointy. První z nich se jmenuje `/register` a pomocí něj může uživatel nastavovat instanci `Icinga2`, kterou chce sledovat. Může si také vybírat typy notifikací, které chce získávat a také může nastavovat čas, ve kterém má uživatel notifikace získávat. Druhým endpointem je `/enable`, na který uživatel odesílá požadavek na zastavení nebo obnovení odesílání notifikací.

Třída `ProxyRoute` implementuje endpointy týkající se samotného monitorování. Endpointy v této třídě můžeme rozdělit do 3 skupin:

1. Endpointy získávající data o větším množství objektů (monitorované služby a servery)

2. Endpointy získávající data o 1 objektu v závislosti na parametrech obsažených v požadavku - můžeme získávat data o službách, serverech nebo komentářích a odstávkách na jednotlivých službách a serverech.

3. Endpointy vykonávající akce, které se provádí vždy na 1 objektu. Jedná se o akce potvrzení problému, přidání informace o odstávce, přidání komentáře, odstranění těchto informací nebo okamžité vykonání kontroly objektu.

`IndexRoute` implementuje pouze jeden endpoint, který slouží ke kontrole, toho, zdali se tato aplikace nachází na serveru. Slouží pro ověření toho, zda je správně v mobilní aplikace nastavena konfigurace backendu.

Přesné definice endpointů byly zpracovány do souboru `swagger.yaml` podle specifikace `OpenAPI` [88], který se nachází v kmenovém adresáři `ts-backend`. Tento soubor si je možné prohlédnout v případě, že máte backend spuštěný na svém stroji v libovolném prohlížeči na adrese `http://localhost:3000/api-docs/#`, nebo pomocí editoru `Swagger` [87].

7.2.2 Controllers

Další skoupinou tříd je `Controllers`, kteřá představuje vrstvu mezi `Routes` a `Services`, Stará o zpracování uživatelského požadavku. Data se předávají do `Services` a odesílají se zpět uživateli. Jsou využity kontrolery `ClientConfigController`, `ProxyController` a `IndexController`. `ClientConfigController` se stará o tvorbu a změnu klientských konfigurací, `ProxyController` slouží pouze k přeposílání klientských požadavků na data serveru `Icinga2`. `IndexController` pouze odpovídá na jakýkoliv dotaz, který je přijat stavem `200 OK`.

7.2.3 Services

Jedná se o třídy, ve kterých jsou zpracovávány požadavky přicházející ze tříd `Controllers`. Zatímco `Controllers` pouze zpracovávají požadavky přicházející od uživatelů, třídy `Services` vykonávají funkce, které jsou pro odpověď uživateli potřeba.

IcingaRequestService

Tato třída obsluhuje zasílání HTTP požadavků na server s monitorovacím nástrojem `Icinga2`. Příchozí požadavek z mobilní aplikace je zpracován a je vyvolána jedna z obsluhujích funkcí. Funkce následně podle předaných parametrů rozhodne o typu HTTP požadavku: `GET`, nebo `POST`. Data předaná této funkci jsou přidána k požadavku. Jedná se o asynchronní funkce. Asynchronní funkce fungují tak, že v případě, že je v programu potřeba čekat na odpověď (může jít o náročný výpočet uvnitř programu, nebo v tomto

případě o volání na vzdálený server), zastaví se vykonávání funkce v bodě, kde je třeba čekat. Aplikace vykonává mezitím jiné funkce a ve chvíli, kdy je odpověď získána, pokračuje aplikace ve vykonávání funkce od místa, kde byla tato funkce pozastavena.

IcingaConnectionFactory

Instance této třídy jsou vytvářeny službou cron, která se stará o plánování procedur. Poté, co je instance vytvořena, je spuštěna funkce `makeRequest`, která vytvoří HTTP požadavek na `Icinga2 Api`, konkrétně `EventStreams`. V případě, že je požadavku odeslána odpověď, vytvoří se notifikace s textem, který informuje o události na Icinga2 monitorovacím nástroji.

V případě, že do 60 sekund není obdržena odpověď, je úkol ukončen a znovu spuštěn. Tento interval je dán z toho důvodu, že pro HTTP požadavky existuje maximální doba pro odpověď a v případě, že do této doby nenastane na Icinga2 serveru žádná událost, je požadavek ukončen a uživatel tak žádnou notifikaci nikdy nedostane. Proto se musí pravidelně dotazy obnovovat. Dalším důvodem pro tento postup je to, že uživatel může změnit svoji konfiguraci (změnou serveru, typu notifikací, atd.). V případě, že by se pravidelně tato procedura neobnovovala, uživatel by tuto změnu nepocítil do doby, než by získal notifikaci z původní konfigurace (což může být několik sekund, nebo se notifikace nemusí odeslat vůbec).

NotificationQueueService

Jedná se o službu, která se stará o rozvrhování zasílání notifikací. V okamžiku, kdy je obržena notifikace od `Icinga2`, je vyfiltrována s již existujícími notifikacemi pro případ, že by se notifikace jednomu uživateli odeslala vícekrát. Následně jsou notifikace vytříděny podle toho, zdali má uživatel zapnutou službu notifikací tlačítkem, nebo zdali se nenachází mimo uživatelem vyznačený interval pro odesílání notifikací. Následně jsou tyto notifikace najednou v pravidelném časovém intervalu odesílány jako jedno pole s notifikacemi na `Expo server`. Notifikace by mohly být posílány i v okamžiku, kdy přijdou. Nicméně odesílání notifikací v jednom poli je z důvodu optimalizace výhodnější, neboť posíláním jednoho pole se vytvoří pouze jeden požadavek, zatímco posíláním každé notifikace zvlášť by se mohly vytvářet i tisíce požadavků, což by zvyšovalo zátěž jak na tuto aplikaci, tak i na servery `Expo`.

CronService

`Cron` je služba, která se stará o opakované vykonávání metod. Každou minutu je z databáze získán seznam existujících uživatelských nastavení, které jsou následně transformovány a z nich se vytváří požadavky na tzv. `EventStreamy Icinga2`, jež slouží k získávání notifikací.

ClientConfigService

Zde se zpracovávají požadavky na vytvoření nové klientské konfigurace a změny v nastavení notifikací.

7.3 Použité knihovny

Stejně jako u mobilní aplikace, i zde byly použity knihovny, které zjednodušují vývoj aplikace.

7.3.1 Express

`Express` [22] je open-source framework používaný pro tvorbu webových aplikací. Podporuje přidávání tzv. middleware, což jsou funkce, které jsou vykonány během existence HTTP požadavku. Middleware pak může požadavek upravit, zpracovat a odpovědět či zahodit.

7.3.2 Prisma

`Prisma` [65] je ORM knihovna pro `Node.js`. Používá se pro generaci databázových tabulek a slouží jako prostředník mezi backendovými službami a databází `MySQL`, která je v projektu použita pro ukládání uživatelských dat.

8 Překlad a spuštění

V následujících kapitolách bude popsán postup pro překlad a spuštění obou částí této práce. Celý projekt je možné stáhnout pomocí nástroje `Git` [29]. V příkazové řádce je následně potřeba vybrat adresář, do kterého bude projekt následně stažen. Poté, co je adresář vybrán, spusťte v příkazové řádce příkaz `git clone https://gitlab.com/josef.bozdech2/icinga-monitoring-mobile-application.git`, kterým se projekt stáhne.

8.1 Mobilní aplikace

Aplikace může být spuštěna na telefonu 2 způsoby. Aplikace je na telefonu buď nainstalována, nebo je spuštěna ve vývojářském režimu. V případě, že je aplikace nainstalována standardním způsobem, nepotřebuje žádné jiné technologie, proto se tato kapitola bude zaobírat spouštěním aplikace pomocí technologie `Expo`.

Aplikaci je možno spustit na jakémkoliv zařízení s `OS Android`, verze 6.0+. Telefony s `iOS` mohou aplikaci spustit v případě, že je v nich nainstalována verze `iOS 12.0` a vyšší.

Pro spuštění této aplikace je potřeba stáhnout závislosti. K tomu je potřeba naistalovat framework `Node.js` [61]. Po úspěšné instalaci je potřeba v adresáři `frontend` spustit příkaz `npm install`, který nainstaluje závislosti.

Pokud je aplikace spouštěna ve vývojářském režimu, v telefonu je potřeba mít nainstalovanou aplikaci `Expo Go` [1, 7], skrz níž bude aplikace spouštěna. V tomto případě je potřeba mít i stolní počítač, na kterém je nainstalována `Expo CLI` [18].

Server `Expo` pro testování aplikace (stažený již pomocí příkazu npm install v předchozím kroku) je spuštěn v domovském adresáři projektu příkazem `expo start`. Aplikace se zkompiluje a následně se ve webovém prohlížeči spustí `Metro Bundler`, který slouží k zobrazování informací při debugování aplikace (tyto informace je možné vidět také v příkazové řádce). Poté, co se aplikace zkompiluje, je možné ji spustit v telefonu pomocí aplikace `Expo Go`. Aplikace se následně stáhne do telefonu a je možné ji normálně používat. Aplikace komunikuje s backendovou aplikací, jejíž URL je uložená v proměnné v souboru `env.js`. V případě, že je aplikace testována, neměla by se připojovat na produkční aplikaci, ale na testovací verzi backendové aplikace. Uvnitř aplikace samotné není možné měnit adresu backendu, se kterým aplikace komunikuje. Aplikaci je již pak možné normálně používat.

Z aplikace může být vytvořen balíček pro Android a iOS. K tomu je potřeba mít vytvořený účet Expo. Ten je možný vytvořit na webových stránkách Expo. Tvorba balíčku je provedena příkazem expo build:'typ_os', přičemž typ_os je potřeba nahradit za android, nebo ios. Expo CLI následně umožní si vybrat typ balíčku, který chceme vytvořit. U verze Android jde o APK, nebo AAB. Balíček APK je určený přímo pro instalaci na zařízení, kdežto AAB je balíček, který se přidává do obchodu Google Play. Pro iOS je umožněna tvorba balíčků pro iOS simulátory nebo tvorba archivů používaných k publikaci na App Store a distribuci aplikace pomocí nástroje TestFlight. Podrobný návod k tvorbě balíčků je dostupný v literatuře [20]. Balíčky se kompilují a tvoří na serverech Expo. Balíčky je následně potřeba stáhnout z www.expo.dev.

8.2 Backendová aplikace

Testování instalace proběhlo na operačních systémech Windows 10 a Debian 11. Aplikaci je nicméně možné nasadit na operační systémy Windows, MacOS a všechny typy Unixových OS, kde je možné nainstalovat Docker [15] a Docker Compose [16].

Spustění projektu je provedeno příkazem docker-compose up v adresáři ts-backend. Docker zajistí vytvoření Docker balíčků a jejich spuštění. Proběhne vytvoření 3 balíčků: backendové aplikace, MySQL databáze, která bude ukládat uživatelské konfigurace, a balíčku migračního skriptu, který v databázi vytvoří potřebné tabulky. V příkazové řádce je pak možné sledovat proces tvorby jednotlivých balíčků. Nejdříve proběhne tvorba balíčku mysql, neboť je závislostí pro dva zbylé balíčky. Po vytvoření tohoho balíčku jsou paralelně vytvářeny balíčky server a database-migration-script. Migrační skript se po vykonání ukončí, v příkazové řádce by se měl objevit řádek s textem *database-migration-script exited with code 0*. Balíček server by měl vypsat informace o tom, že se spouští CronService, a že začíná poslouchat na portu 3000. V tuto chvíli je server připravený přijímat požadavky od mobilní aplikace.

9 Testování

Kvůli zjištění stability a funkčnosti aplikace bylo provedeno její testování. Nejdříve bylo provedeno testování výkonu aplikace, které cílilo na maximální uživatelskou spokojenost s užíváním aplikace. Následovalo testování získávání notifikací.

9.1 Testování stability

Testování bylo provedeno v několika fázích, aby došlo ke zjištění kapacity objektů, které aplikace může zobrazovat bez toho, aniž by došlo k havárii, jedná se tedy zejména o testování výkonu aplikace. Do konfigurace `Icinga2` byly postupně přidávány služby a servery, které bylo potřeba monitorovat.

Testování proběhlo na telefonech `Xiaomi Mi 9 Lite, Android` a `iPhone 7, iOS`, backendová aplikace pak byla spuštěna na soukromém serveru s `OS Debian 9`. Na obou zařízeních byly nainstalovány všechny potřebné technologie.

Pro testování aplikace bylo stanoveno několik kritérií, podle nichž byla aplikace následně testována. Nejdůležitějším kritériem je doba načítání jednotlivých obrazovek. Uživatel nemůže čekat na vykreslení všech objektů. Aplikace musí umět rychle zobrazit všechna potřebná data a musí okamžitě reagovat na uživatelem provedené akce.

Proto bylo provedeno testování tak, že byla naměřena doba, během které se jednotlivé obrazovky načítaly. Toto testování bylo provedeno v několika fázích, kdy byly k monitorování na serveru `Icinga2` přidávány jednotlivé objekty. Objekty byly vždy přidávány tak, že ke každému monitorovanému serveru bylo přidáno tisíc monitorovaných služeb. Naměřené časy byly zpracovány do tabulky 9.1. Dále byly do tabulky přidány hodnoty `Odezva aplikace` a `Přehlednost aplikace`. Hodnota `Odezva aplikace` je hodnocení toho, jak rychle aplikace reaguje na uživatelovy akce. Hodnota `Přehlednost aplikace` je pak vyhodnocení, jak dobře se v aplikaci hledají jednotlivé služby a servery. Poslední 2 hodnoty jsou pouze dojmem tvůrce tohoto textu a nemusí se plně shodovat s názory ostatních uživatelů. Aplikace byla testována v nejhorším možném scénáři, tj. všechny služby jsou ve stavu `Critical` a telefon dostává všechna oznámení najednou.

Doba načítání/ostatní	5	50	100	500	1000	2000	5000	10000
OverviewScreen [s]	1	1	1	1	1,1	1,3	1,9	2,2
ServiceList [s]	0,1	0,1	0,2	0,4	0,6	1,2	1,6	2,1
HostList [s]	0,1	0,1	0,1	0,1	0,1	0,2	0,2	0,2
Odezva aplikace	10	10	10	10	10	9	9	8
Přehlednost aplikace	10	9	9	9	9	9	8	8

Tabulka 9.1: Zátěžové testování mobilní aplikace po změně vykreslovací knihovny

Aplikace zvládla testování stability velice dobře. Doba načítání obrazovek `OverviewScreen` a `ServiceList` rostla s množstvím monitorovaných komponent. Hodnoty naměřené u `OverviewScreen` jsou u množství objektů menší než 1000 pravděpodobně ovlivněny samotným načítáním aplikace. V případě, že by se načítala pouze tato obrazovka, dosáhla by nejspíše stejných hodnot, jako ostatní obrazovky.

Obrazova `ServiceList` zvládla testování také velice dobře. Načítání obrazovky bylo rychlé a aplikace reagovala na uživatelské akce také velmi rychle. Délka načítání této obrazovky byla zasažena stahováním dat z `API`. Při větší rychlosti stahování dat by se mohla obrazovka načíst ještě rychleji. Při vyšších tisícovkách objektů se hůře vyhledávaly jednotlivé objekty. Filtrování objektů pomocí názvu ale v tomto případě velice pomáhá.

Obrazovka `HostList` nebyla zvyšujícím se množstvím monitorovaných objektů vůbec zasažena. Reagovala na požadavky okamžitě a vyhledávání na ní také nebylo vůbec zasaženo.

Jako úzké místo bych označil `Icinga2 API`, které nepodporuje stránkování požadavků. Data jsou aplikaci posílána najednou a i přes optimalizaci zobrazování dat v mobilní aplikaci není výkon vyhovující. Při množství objektů kolem 10 000 a více může množství odesílaných dat přesahovat 10 Mb, což je při pravidelné aktualizaci každé stránky poměrně velké množství dat. Tento problém by vyřešilo stránkování, které bohužel `Icinga2 API` nepodporuje.

9.2 Testování notifikací

Pro uživatele je důležité, aby dostával notifikace pouze v případech, kdy je doopravdy chce. Tento případ je menší ze dvou problémů, neboť uživatel je stále informován, byť si to nepřeje. Velký problém nastává v případě, kdy notifikaci dostat má a nedostane ji. Kvůli tomu může v síti nastat problém, který nebude nikým řešen, což může způsobit velké komplikace.

Kvůli těmto případům byly vytvořeny funkční testy, které mají tento problém při vývoji eliminovat. Tyto testy nabírají podobu testovacích scénářů, které dostane tester a podle nichž se řídí při testování aplikace.

V prvním odstavci byly nastíněny problémy, které se mohou v aplikaci objevit. Následně byly definovány možnosti, kdy mohou jednotlivé problémy nastat:

- Aplikace je zapnutá / vypnutá
- Notifikace je zaslána v intervalu / mimo interval, kdy mají být notifikace obdrženy
- Specifický typ notifikace je vypnutý / zapnutý

Pro tyto scénáře byl vytvořen jednoduchý skript, který provede uživatele výběrem scénáře a následně vyvolá na `Icinga2` serveru akci, která má vyvolat vytvoření notifikace. Ten je dostupný v příloze `KivIciTestCaseProvider`, nebo stažením z internetu pomocí příkazu `git clone https://gitlab.com/josef.bozdech2/kivicinotificationtesting.git`.

Pro všechny kombinace scénářů byla provedena 3 měření v různých časech (dohromady 24 měření). Ve všech případech byla notifikace obdržena správně. Správně jsou notifikace získávány i v případě, že je aplikace v telefonu vypnutá.

Výsledkem tohoto testování je plná funkčnost jednotlivých nastavení. Notifikace jsou zasílány v případech, kdy je jejich nastavení platné a jsou zahozeny, kdykoliv alespoň jedna podmínka není platná.

10 Nasazení aplikace do obchodů

Standardem ve světě aplikací je jejich stahování z oficiálních obchodů jednotlivých platforem. Proto byla aplikace přidána do obchodů `Google Play` a `App Store`. Tato kapitola bude popisovat proces nasazení aplikace na jednotlivé platformy.

Google Play

Pro přidávání aplikací na platformu `Google Play` je potřeba mít uživatelský účet na platformě `Google`. Uživatel se musí tímto účtem registrovat pro vývojářský účet na platformě `Google Play Console` a je dostupná na `www.play.google.com/console`. Dále musí uživatel zaplatit jednorázový poplatek, aby mohl vývojářskou platformu využívat (k datu 13. března 2022 byla tato částka 25 USD). Následně uživatel vybere, zdali se jedná o soukromý, nebo firemní účet, a vyplní požadované údaje. Poté už může uživatel přidávat aplikace.

Úplně na začátek musí uživatel zmáčknout tlačítko `Vytvořit aplikaci` a vybere název aplikace. Následně je potřeba přidat aplikační balíček `AAB`, který je získán podle postupu popsaného v kapitole 8.1. Před oficiálním vydáním je potřeba aplikaci přidat do sekce `testování`. To je rozdělené v `Google Play Console` do 3 úrovní: `interní testování`, které je určené především pro samotné vývojáře a nejužší skupinu testerů (do interního testování je možné zapojit maximálně 100 uživatelů), `uzavřené testování` určené pro širší skupinu uživatelů, uživatel nicméně musí být vývojářem stále do testování pozván, a `otevřené testování`, při němž už je aplikace dostupná k předběžnému přístupu na `Google Play` všem uživatelům.

Ve fázi, kdy je aplikace přidána do `Google Play` už je možné aplikaci aktualizovat pomocí nástroje `Expo`. Pro vydání aktualizace je potřeba v příkazové řádce v adresáři, kde se aplikace nachází, zadat příkaz `expo publish`. Jakmile je aktualizace nahrána do `Google Play`, může být aplikace uživateli aktualizována. Aplikace je aktualizována vždy až po druhém spuštění aplikace od vydání aktualizace. Při prvním spuštění je aktualizace stážena a až při druhém spuštění jsou změny promítnuty do aplikace.

Jako další krok by měla být aplikaci přidána ikonka, popis a fotky z aplikace, možné je přidat i video popisující aplikaci. Také je potřeba vybrat

typ aplikace; zdali se jedná o hru, či nikoliv. Poté je možné zvolit kategorii aplikace, která blíže specifikuje, pod jakými hesly bude možné aplikaci najít.

Aplikace následně může být stahována podle toho, v jakém stádiu vývoje se nachází, zdali je aplikace v interním, uzavřeném, otevřeném testování, nebo zdali je aplikace plně vydána.

App Store

Přidávání aplikací do App Store je složitější, než do Google Play. K přidávání aplikací na tuto platformu je potřeba vlastnit MacBook, nebo alespoň jeho emulaci (virtuální stroj, na kterém je MacOS). S pomocí iPhonu, nebo iPadu není možné aplikace přidávat. Dále je potřeba mít Apple ID, což je uživatelský účet, kterým se přihlašují uživatelé do zařízení od společnosti Apple. S pomocí tohoto účtu je nutné se zaregistrovat do Apple Developer Programu [8]. Narozdíl od platformy Google Play zde musí uživatel platit tento poplatek ročně. Výše tohoto poplatku byla k 19. březnu 2022 99 USD.

Na stránkách www.developer.apple.com pak již může uživatel začít přidávat své aplikace. Na záložce Overview je potřeba kliknout na tlačítko Certificates, Identifiers & Profiles. Po přesměrování je potřeba kliknout na záložku Identifiers a v této záložce kliknout na tlačítko plus vedle názvu Identifiers, na další stránce pak vybrat tlačítko App IDs a následně kliknout na tlačítko Continue, v dalším kroku vybrat typ App a znovu Continue. Na další stránce je pak potřeba vyplnit Description, Bundle ID (jednoznačný identifikátor aplikace, díky kterému lze aplikaci poznat, aplikace vytvořená v této práci má Bundle ID com.bozdechj.kivici), a vybrat Capabilites, což jsou parametry, které určují, jaké zdroje bude aplikace potřebovat (pro tuto práci byly vybrány pouze Push Notifications). V dalším kroku už je jen potřeba potvrdit vyplněné hodnoty tlačítkem Register.

Následně je potřeba přejít na stránku www.appstoreconnect.apple.com, na které se přidá aplikační balíček. Na této stránce je potřeba kliknout na tlačítko My Apps. Po přesměrování kliknout na tlatčíko plus vedle nadpisu Apps, což zobrazí malý formulář. Ve formuláři je potřeba vyplnit typ zařízení, na které je aplikace určena (u této aplikace jde pouze o zařízení iOS), následně vyplnit parametr Name (v této aplikaci KivIci). Následuje vybrání primárního jazyka aplikace, přičemž tato aplikace má jako primární jazyk English U.S. Poté je potřeba vybrat Bundle ID, které bylo vytvořeno v předchozím kroku, a nakonec vyplnit parametr SKU, což je další identifikátor (tento identifikátor musí být unikátní mezi aplikacemi jednoho vývojáře). Na úplném konci je potřeba vybrat mezi typem přístupu uživa-

tele k aplikaci, zdali jde o Limited Access, nebo Full Access (zde bylo vybráno Full Access). Následně už je možno potvrdit formulář tlačítkem Create.

Poté je potřeba aktualizovat stránku ve webovém prohlížeči a rozkliknout záložku aplikace, která by se na stránce měla po aktualizaci objevit. Zde je potřeba se rozhodnout, zdali bude aplikace vydána do App Store rovnou, nebo bude nejdříve předána k testování námi vybraným uživatelům pomocí nástroje TestFlight, který je integrovaný s App Store Conect [8]. V této práci bylo řešeno právě testování aplikace, nikoliv plné vydání, proto zde bude následovat postup pro přidání aplikace do TestFlight (do záložky TestFlight je nutné se překliknout i v App Store Connect).

K přidání aplikace do TestFlight je potřeba mít balíček aplikace. Vytváření balíčků pro iOS se v nástroji Expo lehce liší od tvorby balíčků pro Google Play, nicméně do kroku expo build:typ_os je návod nacházející se v kapitole 8.1 stejný. Poté, co je do příkazové řádky zadán příkaz expo build:ios, je potřeba vybrat typ balíčku archive, který je uzpůsobený pro přidávání na App Store. Následně musí uživatel zvolit, zdali nechá Expo obstarat všechny údaje, nebo zdali si chce tyto údaje vyplnit uživatel sám. Poté je potřeba zadat údaje uživatele, který je zaregistrován jako vývojář v Apple Developer Programu. Je potřeba vyplnit Apple ID a heslo. Ihned je uživatel vyzván, zdali chce zadat distribuční a push certifikáty sám, nebo nechá Expo tyto údaje vyplnit. Po tomto kroku je vytvořen balíček, který je potřeba stáhnout ze server www.expo.dev.

Stažený balíček je pak potřeba nahrát na App Store Connect pomocí nástroje Transporter, který je dostupný na MacOS. Aby mohla být aplikace testována, musí být vyplněno několik formulářů. Tyto formuláře se týkají šifrování v aplikaci (pokud nějaké bylo použito; počítá se i volání požadavků pomocí protokolu HTTPS). První z nich se týká compliance, což je formulář o dodržování zákonů stanovených Spojenými státy. Následně je potřeba vyplnit výroční hlášení.

Aplikace nyní může být předána dalším uživatelům k testování. Toho je docíleno tak, že v záložce TestFlight je tlačítkem plus vedle nápisu External Groups vyvolán formulář, do kterého je potřeba napsat název skupiny testovacích uživatelů. Následným kliknutím na tlačítko Create je akce potvrzena. Testovací uživatele lze do skupiny přidat kliknutím na tlačítko Add Testers ve spodní části obrazovky. Poté je potřeba se vrátit na předchozí stránku kliknutím na tlačítko TestFlight v horní části obrazovky. Poté je potřeba vybrat verzi aplikace a následně je potřeba přidat popis toho, co je potřeba v aplikaci testovat ve spodní části obrazovky pod nadpisem Test Details. Pod tímto formulářem je možné vybrat skupinu testerů klik-

nutím na tlačítko plus vedle nadpisu Groups. Vybráním skupiny testerů je otevřeno okno, ve kterém musí vývojář aplikace poskytnout kontaktní údaje na sebe. Může také vybrat, zdali nechá testovací uživatele používat aplikaci bez přihlašovacích údajů, nebo zdali jim poskytne přihlašovací údaje k aplikaci. Po potvrzení nastavení je otevřeno ještě jedno okno, ve kterém musí vývojář popsat, co chce nechat uživatele testovat. Po splnění všech těchto kroků je uživatel pozvaný k testování aplikace notifikován e-mailem o tom, že byl k testování přizván. Aplikaci si pak může pomocí aplikace TestFlight pro iOS stáhnout k testování.

Vydávání aplikace na App Store se pak příliš neliší od postupu u Google Play. To se provádí v záložce App Store namísto záložky TestFlight. I zde je potřeba pod různými záložkami na levé straně obrazovky přidat popis aplikace, fotky z ní a další. Je také potřeba vybrat o jaký typ aplikace jde, podle čehož se pak odvíjí minimální věk pro stažení aplikace z App Store. Také je potřeba přidat odkaz na web, na kterém je popsáno zabezpečení uvnitř aplikace. Je možné také vybrat zpeněžení aplikace.

Ať je aplikace přidána na TestFlight, nebo na App Store, je aplikace manuálně kontrolována zaměstnanci Apple. Před tím, než je aplikace zkontrolována, nemohou s ní být prováděny žádné úkony, tj. dát aplikaci uživatelům k testování nebo vydání aplikace na App Store.

11 Závěr

Cílem práce bylo vytvořit novou multiplatformní mobilní aplikaci pro vybraný monitorovací nástroj. Nejdříve byly prostudovány některé z vybraných monitorovacích nástrojů v kapitole 2.1. Ze zkoumaných nástrojů byl jako nejvhodnější vybrán nástroj Icinga2. Dále byly porovnány v kapitole 3 různé metody vývoje aplikací, jako lepší přístup pro vývoj aplikace pro monitorovací nástroje byl zvolen multiplatformní vývoj. Jako nejlepší framework byl vybrán React Native. Ze studia existujících mobilních aplikací byly vytvořeny parametry, jež by nová aplikace měla splňovat.

V kapitole 5 byla navržena architektura projektu, bylo rozhodnuto o tvorbě backendové aplikace z důvodu implementace push notifikací. Aplikace byla následně implementována a popsána v kapitolách 6 a 7.

Aplikace byla následně testována na výkon postupným přidáváním monitorovaných objektů až na hodnotu 10 000, přičemž nebyly zjištěny žádné problémy. Otestováno bylo také odesílání notifikací, nicméně ani zde žádné problémy nenastaly. Od testujících uživatelů byly obdrženy vylepšující podněty, které mohou být do aplikace implementovány později.

Nakonec byla aplikace nasazena do obchodu Google Play a do testovacího nástroje TestFlight pro iOS. Na Google Play je aplikace dostupná v testovacím režimu. Aplikace zatím nebyla na App Store vydána z důvodu vysokých poplatků pro vývojáře.

Tato nová aplikace disponuje moderním vzhledem, vysokým výkonem a push notifikacemi, které většině současných mobilních aplikací chyběly.

Literatura

[1] *Expo Go* [online]. Expo Project, 2021. [cit. 2022/03/14]. Installation. Dostupné z: play.google.com/store/apps/details?id=host.exp.exponent&hl=cs&gl=US.

[2] *Android vs. iOS* [online]. Diffen, c2022. [cit. 2022/05/03]. Dostupné z: https://www.diffen.com/difference/Android_vs_iOS.

[3] *Mobile Operating System Market Share Worldwide* [online]. StatCounter, c2021. [cit. 2021/02/08]. Dostupné z: https://gs.statcounter.com/os-market-share/mobile/worldwide.

[4] *Android SDK* [online]. Techopedia Inc., c2022. [cit. 2022/05/03]. Dostupné z: https://www.techopedia.com/definition/4220/android-sdk.

[5] *Android Studio* [online]. Google, c2022. [cit. 2022/04/19]. Dostupné z: https://developer.android.com/studio.

[6] *APNs Overview* [online]. Apple Inc., c2018. [cit. 2022/04/19]. Dostupné z: https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html.

[7] *Expo Go* [online]. 650 Industries, Inc., c2020. [cit. 2022/03/14]. Installation. Dostupné z: apps.apple.com/cz/app/expo-client/id982107779?l=cs.

[8] *Your Apps on the App Store* [online]. Apple Inc., c2022. [cit. 2022/05/03]. Dostupné z: https://developer.apple.com/app-store-connect/.

[9] *Axios* [online]. 2021. [cit. 2021/05/28]. Promise based HTTP client for the browser and node.js. Dostupné z: https://github.com/axios/axios.

[10] Biørn-Hansen, A. et al. An empirical investigation of performance overhead in cross-platform mobile development frameworks. *Empirical Software Engineering*. 2020. doi: 10.1007/s10664-020-09827-6.

[11] *Cron* [online]. Sharpened Productions, 2006. [cit. 2022/04/19]. Dostupné z: https://techterms.com/definition/cron.

[12] D'Ambra, S. *What is Cross Platform Development?* [online]. ClearTech Interactive Corp., c2018. [cit. 2021/01/11]. Dostupné z:

https://www.cleart.com/what-is-cross-nativeappdevelplatform-development.html.

[13] Deshdeep, N. *Mobile App or Website? 10 Reasons Why Apps are Better* [online]. Wingify, 2021. [cit. 2021/06/20]. Dostupné z: https://vwo.com/blog/10-reasons-mobile-apps-are-better.

[14] *Discord* [online]. Discord, c2022. [cit. 2022/03/09]. Dostupné z: https://discord.com/.

[15] *Docker overview* [online]. Docker Inc., c2021. [cit. 2022/01/18].

[16] *Overview of Docker Compose* [online]. Docker Inc., c2021. [cit. 2022/01/18].

[17] *Introduction to Expo* [online]. Expo, c2021. [cit. 2021/06/09]. Dostupné z: https://docs.expo.io/.

[18] *Expo* [online]. Expo, c2021. [cit. 2021/06/10]. Installation. Dostupné z: https://docs.expo.io/get-started/installation/.

[19] *Push Notifications Overview* [online]. Expo, c2021. [cit. 2021/06/09]. Dostupné z: https://docs.expo.io/push-notifications/overview/.

[20] *Building Standalone Apps* [online]. Expo, c2022. [cit. 2022/03/14]. Installation. Dostupné z: https://docs.expo.io/distribution/building-standalone-apps/.

[21] *Sending Notifications with Expo's Push API* [online]. Expo, c2022. [cit. 2022/01/18].

[22] *Express* [online]. OpenJS Foundation, c2017. [cit. 2021/06/07]. Dostupné z: https://expressjs.com/.

[23] *About FCM messages* [online]. Google, 2021. [cit. 2021/05/29]. Dostupné z: https://firebase.google.com/docs/cloud-messaging/concept-options/#notifications_and_data_messages.

[24] *Firebase Push Notification Plugin for Xamarin iOS and Android* [online]. GitHub, c2022. [cit. 2022/03/17]. Dostupné z: https://github.com/CrossGeeks/FirebasePushNotificationPlugin.

[25] *Engage Documentation* [online]. Google, 2021. [cit. 2021/06/09]. Dostupné z: https://firebase.google.com/docs/engage.

[26] *Flutter Fire* [online]. Google, c2021. [cit. 2021/06/10]. Notifications. Dostupné z: https://firebase.flutter.dev/docs/messaging/notifications.

[27] *What Is a Framework?* [online]. Code Academy, 2021. [cit. 2022/04/19]. Dostupné z: https://www.codecademy.com/resources/blog/what-is-a-framework/.

[28] *What is Apache Cordova?* [online]. Gangboard, 2019. [cit. 2021/06/10]. Dostupné z: https://www.gangboard.com/blog/what-is-apache-cordova.

[29] *1.5 Getting Started - Installing Git* [online]. Digital Ocean, c2022. [cit. 2022/03/14]. Dostupné z: https://git-scm.com/book/en/v2/Getting-Started-Installing-Git.

[30] *A query language for your API* [online]. The GraphQL Foundation, c2022. [cit. 2022/04/19]. Dostupné z: https://graphql.org/.

[31] *IT Infrastructure Monitoring Essentials* [online]. Heroix, c2021. [cit. 2021/01/10]. Dostupné z: https://go.heroix.com/it-infrastructure-monitoring.

[32] *HTTP* [online]. The Computer Language Co Inc., c2022. [cit. 2022/05/03]. Dostupné z: https://www.pcmag.com/encyclopedia/term/http.

[33] *Api - Icinga 2* [online]. Icinga GmbH, c2021. [cit. 2021/01/11]. Dostupné z: https://icinga.com/docs/icinga-2/latest/doc/12-icinga2-api/#icinga2-api-permissions.

[34] *Introduction - Icinga Director* [online]. Icinga GmbH, c2021. [cit. 2021/01/12]. Dostupné z: https://icinga.com/docs/icinga-director/latest/doc/01-Introduction/.

[35] *Service Monitoring* [online]. Icinga GmbH, 2021. [cit. 2021/01/12]. Dostupné z: https://icinga.com/docs/icinga-2/latest/doc/05-service-monitoring/.

[36] *Icinga Reviews* [online]. TrustRadius, c2021. [cit. 2021/06/20]. Dostupné z: https://www.trustradius.com/products/icinga/reviews.

[37] *About - Icinga Web 2* [online]. Icinga GmbH, c2021. [cit. 2021/01/10]. Dostupné z: https://icinga.com/docs/icinga-web-2/latest/doc/01-About/.

[38] *Iciview* [online]. Apple Inc., c2021. [cit. 2021/01/13]. Dostupné z: https://apps.apple.com/us/app/iciview/id1141720590#?platform=iphone.

[39] *What Is an IDE?* [online]. Codeacademy Inc., c2022. [cit. 2022/05/03]. Dostupné z: https://www.codecademy.com/article/what-is-an-ide.

[40] *What is Ionic: Learn the essentials of what you can do with Ionic and how it works.* [online]. Ionic, c2021. [cit. 2022/03/14]. Dostupné z: https://ionic.io/resources/articles/what-is-ionic.

[41] *Jira Software* [online]. Atlassian, c2022. [cit. 2022/03/09]. Dostupné z: https://www.atlassian.com/software/jira.

[42] Khalaf, S. *Apps Solidify Leadership Six Years into the Mobile Revolution* [online]. Flurry, 2014. [cit. 2021/06/20]. Dostupné z: https://www.flurry.com/blog/apps-solidify-leadership-six-years-into-the-mobile/.

[43] Lagholz, S. *Understanding iOS Remote vs Local Push Notifications* [online]. One Signal, 2022. [cit. 2022/01/18].

[44] Liu, S. *Cross-platform mobile frameworks used by software developers worldwide in 2019 and 2020* [online]. Statista, 2020. [cit. 2021/01/11]. Dostupné z: https://www.statista.com/statistics/869224/worldwide-software-developer-working-hours/.

[45] Martin, S. *7 Popular Cross-Platform App Development Tools That Will Rule in 2021* [online]. Medium, 2020. [cit. 2021/01/11]. Dostupné z: https://medium.com/datadriveninvestor/7-popular-cross-platform-app-development-tools-that-will-rule-in-2020-349c80fb51.

[46] *Mobile App vs Mobile Website — the Pros and Cons for your Business* [online]. Medium, 2019. [cit. 2021/06/20]. Dostupné z: https://medium.com/@codecontrol/mobile-app-vs-mobile-website-the-pros-and-cons-for-your-business-974d4c8f0042.

[47] Melnick, J. *Best Server Monitoring Software Tools* [online]. Netwrix Corporation, 2021. [cit. 2021/02/08]. Dostupné z: https://blog.netwrix.com/2018/10/02/top-10-best-windows-server-monitoring-software-tools/.

[48] *Microsoft Teams* [online]. Microsoft, c2022. [cit. 2022/03/09]. Dostupné z: https://www.microsoft.com/cs-cz/microsoft-teams/group-chat-software.

[49] *What is Cross Platform Development?* [online]. ClearTech Interactive Corp., c2022. [cit. 2021/03/17]. Dostupné z: https://azure.microsoft.com/cs-cz/overview/what-is-azure/.

[50] *People Spent 90% of Their Mobile Time Using Apps in 2021* [online]. Fifty Pixels Ltd., c2022. [cit. 2022/03/13]. Dostupné z: https://www.mobiloud.com/blog/mobile-apps-vs-mobile-websites.

[51] *Moment.js* [online]. Moment.js, 2021. [cit. 2021/06/07]. Parse. Dostupné z: https://momentjs.com/docs/#/use-it/.

[52] *Promise - JavaScript* [online]. Mozilla, c2021. [cit. 2021/05/30]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise.

[53] *MySQL* [online]. Oracle, c2022. [cit. 2022/05/03]. Dostupné z: https://www.mysql.com/.

[54] *API Reference* [online]. Nagios Enterprises, LLC., c2021. [cit. 2021/05/29]. Dostupné z: https://www.nagios.org/ncpa/help/2.0/api.html.

[55] *Event Handlers* [online]. Nagios, c2018. [cit. 2021/06/20]. Dostupné z: https://assets.nagios.com/downloads/nagioscore/docs/nagioscore/4/en/eventhandlers.html.

[56] *Nagios Core* [online]. NagiosEnterprises, 2020. [cit. 2021/01/11]. Nagios Core Github. Dostupné z: https://github.com/NagiosEnterprises/nagioscore.

[57] *Nagios Core Reviews* [online]. TrustRadius, c2021. [cit. 2021/06/20]. Dostupné z: https://www.trustradius.com/products/nagios-core/reviews.

[58] *Nagios XI* [online]. Nagios Enterprises, c2022. [cit. 2022/03/08]. Dostupné z: https://www.nagios.com/products/nagios-xi/.

[59] *Understanding native app development - what you need to know in 2019* [online]. Raygun, 2019. [cit. 2021/06/09]. Dostupné z: https://raygun.com/blog/native-app-development/.

[60] *NativeBase* [online]. NativeBase, c2021. [cit. 2021/06/01]. Dostupné z: https://nativebase.io/.

[61] *Node.js Download* [online]. OpenJS Foundation., c2022. [cit. 2022/03/14]. Dostupné z: https://nodejs.org/en/.

[62] Occhino, T. *React Native: Bringing modern web techniques to mobile* [online]. Facebook, 2015. [cit. 2021/01/11]. Dostupné z: https://engineering.fb.com/2015/03/26/android/react-native-bringing-modern-web-techniques-to-mobile/.

[63] *Object-Relational Mapping (ORM)* [online]. Techopedia Inc., c2022. [cit. 2022/05/03]. Dostupné z: https://www.techopedia.com/definition/24200/object-relational-mapping--orm.

[64] Petrequin, G. *10 Best Push Notification Services And Tools in 2022* [online]. Fifty Pixels Ltd. MobiLoud, c2022. [cit. 2022/01/18].

[65] *Prisma Docs* [online]. Prisma, c2022. [cit. 2022/01/18].

[66] *A List of Programming Languages Every Programmer Should Know (Or at Least Know About)* [online]. Devmountain, c2022. [cit. 2022/05/03]. Dostupné z: https://devmountain.com/blog/a-list-of-programming-languages-every-programmer-should-know-or-at-least-know-about/.

[67] *PRTG Network Monitor* [online]. Paessler AG, c2022. [cit. 2022/03/09]. Dostupné z: https://www.paessler.com/prtg.

[68] *PRTG Reviews* [online]. TrustRadius, c2021. [cit. 2021/06/20]. Dostupné z: https://www.trustradius.com/products/prtg-network-monitor/reviews.

[69] *PRTG Manual: Notifications* [online]. Paessler AG, c2021. [cit. 2021/06/20]. Dostupné z: https://www.paessler.com/manuals/prtg/notifications.

[70] *Networking* [online]. Facebook Inc., c2021. [cit. 2021/05/29]. Dostupné z: https://reactnative.dev/docs/network.

[71] *Introducing Hooks* [online]. Facebook Inc., c2021. [cit. 2021/05/29]. Dostupné z: https://reactjs.org/docs/hooks-intro.html.

[72] *Hooks API Reference* [online]. Facebook Inc., c2021. [cit. 2021/05/29]. Dostupné z: https://reactjs.org/docs/hooks-reference.html#usecontext.

[73] *Using the Effect Hook* [online]. Facebook Inc., c2021. [cit. 2021/05/29]. Dostupné z: https://reactjs.org/docs/hooks-effect.html.

[74] *Using the State Hook* [online]. Facebook Inc., c2021. [cit. 2021/05/29]. Dostupné z: https://reactjs.org/docs/hooks-state.html.

[75] *AsyncStorage* [online]. Facebook Inc., c2022. [cit. 2022/03/11]. Dostupné z: https://reactnative.dev/docs/asyncstorage.

[76] *React Native Big List* [online]. Marco Cesarato, 2022. [cit. 2022/01/18].

[77] *React Native Reanimated* [online]. Software Mansion, 2022. [cit. 2022/01/18].

[78] *Advantages and disadvantages of using Flutter* [online]. Sannacode, c2022. [cit. 2022/03/17]. Dostupné z: https://sannacode.com/blog/advantages-and-disadvantages-using-flutter.

[79] Schlothauer, S. *JavaScript on top, Python ties with Java in RedMonk rankings* [online]. Jaxenter, 2020. [cit. 2021/01/11]. Dostupné z: https://jaxenter.com/redmonk-q12020-javascript-169089.html.

[80] Sha, A. *APK vs AAB (Android App Bundles): Everything You Need to Know!* [online]. Beebom Media Private Limited, 2021. [cit. 2022/05/03]. Dostupné z: https://beebom.com/apk-vs-aab/.

[81] Shatz, E. *The Difference Between the App Store and Google Play Store* [online]. Storemaven, 2020. [cit. 2022/05/03]. Dostupné z: https://www.storemaven.com/academy/apple-app-store-google-play-aso-design/.

[82] *Slack* [online]. Slack Technologies, LLC, c2022. [cit. 2022/03/09]. Dostupné z: https://slack.com/.

[83] *SOAP vs REST. What's the Difference?* [online]. SmartBear, 2020. [cit. 2021/05/29]. Dostupné z: https://smartbear.com/blog/soap-vs-rest-whats-the-difference/.

[84] Smyth, N. *Android Studio 4.0 Development Essentials – Java Edition.* Payload Media, Inc., 2020. ISBN 978-1-951442-21-7.

[85] *Share of the global server market by operating system in 2018 and 2019* [online]. Statista, 2020. [cit. 2022/03/12]. Dostupné z: https://www.statista.com/statistics/915085/global-server-share-by-os/.

[86] *Infrastructure monitoring: What is IT infrastructure monitoring?* [online]. Sumologic, c2021. [cit. 2021/01/10]. Dostupné z: https://www.sumologic.com/glossary/infrastructure-monitoring/.

[87] *OpenAPI Specification* [online]. SmartBear Software, c2021. [cit. 2022/03/14]. Dostupné z: https://swagger.io/tools/swagger-editor/.

[88] *Swagger Editor* [online]. SmartBear Software, 2020. [cit. 2022/03/14]. Dostupné z: https://swagger.io/specification/.

[89] *Cross-Platform Development* [online]. Janalta Interactive, c2021. [cit. 2021/01/11]. Dostupné z: https://www.techopedia.com/definition/30026/cross-platform-development.

[90] *Beta Testing Made Simple with TestFlight* [online]. Apple Inc., c2022. [cit. 2022/05/03]. Dostupné z: https://developer.apple.com/testflight/.

[91] *Types of mobile app* [online]. Unifunds, c2021. [cit. 2021/01/13]. Dostupné z: https://unifunds.com/types-of-mobile-app.

[92] Upasana. *Linux vs Windows: Which One Is The Best Choice For You?* [online]. Brain4ce Education Solutions Pvt. Ltd., 2022. [cit. 2022/05/03]. Dostupné z: https://www.edureka.co/blog/linux-vs-windows/.

[93] Upasana. *Unix vs Linux: Difference and Comparison* [online]. Brain4ce Education Solutions Pvt. Ltd., 2021. [cit. 2022/05/03]. Dostupné z: https://www.edureka.co/blog/unix-vs-linux/.

[94] *The WebSocket API (WebSockets)* [online]. Mozilla, c2022. [cit. 2022/03/12]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API.

[95] Wilson, M. *Best IT Infrastructure Monitoring Tools and Software* [online]. PCWDLD, 2021. [cit. 2021/06/02]. Dostupné z: https://www.pcwdld.com/best-infrastructure-monitoring-tools-and-software.

[96] Wurmser, Y. *The Majority of Americans' Mobile Time Spent Takes Place in Apps* [online]. Insider Intelligence Inc., 2020. [cit. 2022/03/13]. Dostupné z: https://www.emarketer.com/content/the-majority-of-americans-mobile-time-spent-takes-place-in-apps.

[97] *Posílání nabízených oznámení do Xamarin pomocí Azure Notification Hub* [online]. microsoft, 2021. [cit. 2021/06/20]. Dostupné z: https://docs.microsoft.com/cs-cz/azure/notification-hubs/xamarin-notification-hubs-ios-push-notification-apns-get-started.

[98] *Xcode 13* [online]. Apple Inc., c2022. [cit. 2022/05/03]. Dostupné z: https://developer.apple.com/xcode/.

[99] Yaskevich, A. *Apache Cordova: Is there a future for it?* [online]. ScienceSoft USA Corporation., 2017. [cit. 2021/01/11]. Dostupné z: https://www.scnsoft.com/blog/apache-cordova-is-there-a-future-for-it.

[100] *ZABBIX 6.0 LTS* [online]. Zabbix LLC, c2022. [cit. 2022/03/09]. Dostupné z: https://www.zabbix.com/.

[101] *Zabbix Documentation 5.4* [online]. Zabbix SIA., c2021. [cit. 2021/06/20]. 4 Webhook. Dostupné z: https://www.zabbix.com/documentation/current/manual/config/notifications/media/webhook.

[102] *Zabbix Reviews* [online]. TrustRadius, c2021. [cit. 2021/06/20]. Dostupné z: https://www.trustradius.com/products/zabbix/reviews.

Příloha A - Seznam zkratek

- IT - Informační technologie
- SNMP - Simple network monitoring protocol - Protokol používaný k monitorování počítačových sítí.
- API - Application Programming Interface - Rozhraní aplikací pro síťovou komunikaci.
- REST - Representational State Transfer - Druh architektury API.
- URL - Uniform Resource Locator - Řetězec znaků sloužící jako indentifikátor umístění zdrojů informací na Internetu.
- JSON - JavaScript Object Notation - Způsob zápisu dat.
- XML - Extended Markup Language - Způsob zápisu dat.
- HTTP - Hypertext Transfer Protocol - Webový protokol standardu TCP/IP.
- SDK - Software Development Kit - Sada vývojových nástrojů pro vývoj softwaru.
- OS - Operační systém.
- HTML - Hypertext Markup Language - Značkovací jazyk pro tvorbu webových stránek.
- CSS - Cascade Styles - Stylovací jazyk pro HTML.
- FCM - Firebase Cloud Messaging - Nástroj pro posílání push notifikací.
- DOM - Document Object Model - Objektově orientovaná reprezentace HTML objektu.
- ID - Identity Document - Unikátní identifikátor.
- AAB - Android App Bundle - Aplikační balíček mobilní aplikace pro Android, který se nahrává do obchodu Google Play.
- APK - Android application PacKage - Instalační balíček pro zařízení Android.

- IDE - Integrated Development Environment - Vývojové prostředí, ve kterém se vytváří aplikace
- ORM - Object Relational Mapping - Programovací technika pro konverzi dat mezi databází a objektově orientovaným programovacím jazykem.

Příloha B - Uživatelská příručka

Konfigurace

Pro to, aby mohla aplikace fungovat, je potřeba mít nastavenou konfiguraci backendového serveru. Toto nastavení je potřeba provést při prvním spuštění aplikace. Toto nastavení je provedeno na obrazovce BackendConfig.

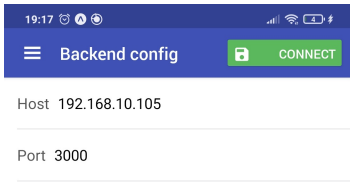
Vyplnit je potřeba:

- IP adresu serveru, na kterém je backendová aplikace spuštěna
- Port, na kterém je tato aplikace dostupná

Pro testování je tato aplikace dostupná na

- 142.93.38.124
- 3000

Poté může uživatel stisknout tlačítko Connect. Aplikace provede kontrolu toho, zdali je na této kombinaci nastavení backendová aplikace dostupná. V případě, že ano, je aplikace restartována, aby se změny okamžitě propsaly do ostatních obrazovek. V případě, že na této adrese aplikace dostupná není, uživatel je informován a musí zadat jinou konfiguraci. Obrazovku je možné vidět na obr. 11.1.



Obrázek 11.1: Obrazovka BackendConfig

OverviewScreen

Aby aplikace mohla zobrazovat informace z Icinga2 monitorovacího nástroje, je potřeba nastavit přihlašovací údaje. Na obrazovce ConfigScreen je potřeba vyplnit tyto údaje:

- IP adresa, na které je spuštěn nástroj Icinga2

- Port, na kterém je nástroj Icinga2 dostupný
- Uživatelské jméno Icinga2 ApiUsera
- Heslo Icinga2 ApiUsera

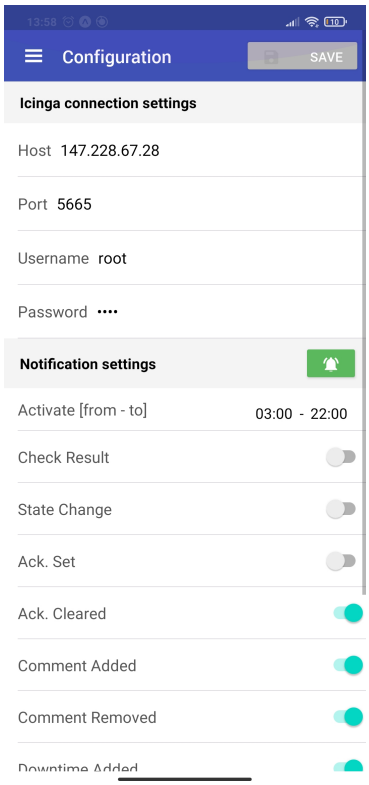
Uživatelské jméno a heslo jsou údaje uživatele Icinga2 ApiUser, které nemusí být totožné s údaji Icingaweb2 uživatele. V případě, že je konfigurace validní, začne mobilní aplikace zobrazovat informace. V případě, že konfigurace validní není, nebude mobilní aplikace zobrazovat nic.

Pro testování byl vytvořen server s Icinga2, pro vyzkoušení aplikace je možné využít tyto údaje:

- 147.228.67.28
- 5665
- root
- 7592

Volitelně může uživatel nastavit jednotlivé možnosti notifikací, které chce uživatel získávat. Notifikace jsou totožné s EventHandlery Icinga2 API. Notifikace budou uživateli přicházet v čase, který si stanoví kliknutím na čas from - to na obrazovce.

Notifikace je možné ale také úplně vypnout. K tomu slouží tlačítko zvonečku. Po vypnutí notifikací tímto tlačítkem nebude uživatel dostávat notifikace až do doby, dokud je opět tímto tlačítkem nezapne.

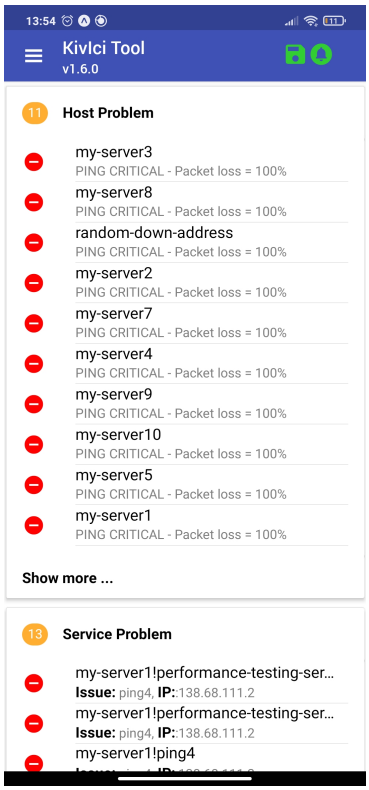


Obrázek 11.2: Obrazovka ConfigScreen

OverviewScreen

Na obrazovce OverviewScreen je uživatel informován o stavu mobilní aplikace: zdali je registrován k backendu, a zdali je přihlášen k odebírání notifikací. Tyto stavy si je možné ověřit pomocí ikonek diskety a zvonku v obrazovce vpravo nahoře. Zelené ikonky značí stav OK. K notifikacím nemůže být přihlášena aplikace, která není spuštěna na fyzickém zařízení, neboť nemůže získat token, jenž je k notifikacím používán.

Na této obrazovce jsou též zobrazeny problémové servery a služby. Na všechny z nich je možné kliknout, uživatel je pak přesunut na obrazovku s detaily objektu. Pokud je objektů s problémem v kategorii více než 10, zobrazí se tlačítko, kterým může uživatel přejít na obrazovky Host List a Service List, kde se mu zobrazí všechny objekty.



Obrázek 11.3: Obrazovka OverviewScreen

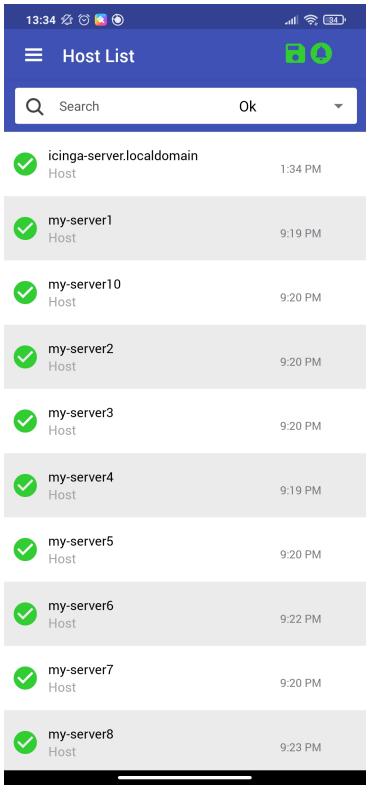
HostList a ServiceList

Na obrazovkách HostList (která je vidět na obrázku 11.4) a ServiceList jsou zobrazeny obecné informace o jednotlivých serverech a službách. Objekty je možné filtrovat, do vyhledávacího řádku je možné zadat libovolné řetězce, zobrazeny budou všechny objekty obsahující daný řetězec. Filtrování je možné pouze podle názvu serveru. Na objekty je možné kliknout, uživatel je pak přesunut na detaily o daném objektu, na obrazovku HostDetail nebo ServiceDetail.

Objekty je také možné zobrazit pouze podle stavu. Ve vyhledávacím řádku je možné napravo kliknout na stav, podle kterého jsou objekty zobrazeny. Mohou být filtrovány takto:

- Problems
- All
- OK
- Warning - Pouze u služeb
- Critical

- Unknown - Pouze u služeb



Obrázek 11.4: Obrazovka HostList

HostDetail a ServiceDetail

Na těchto obrazovkách může uživatel kromě zobrazení podrobných informací o monitorovaných objektech provádět i akce spojené s těmito objekty. Příklad obrazovky HostDetail se nachází na obrázku 11.5.

Uživatel může okamžitě přejít tlačítkem nahoře na HostDetail ze ServiceDetail, a z HostDetail na ServiceList, který je vyfiltrovaný podle jména serveru.

V případě, že není objekt ve stavu OK, může uživatel potvrdit informaci o nastalém problému, nebo potvrzení smazat. Uživatel může také provést okamžitý recheck, a to buď tlačítkem v sekci Problem Handling, nebo dvěma otáčejícími se šipkami v sekci Check Execution. Uživatel může také přidat informaci o odstávce, a to tlačítkem Schedule Downtime. Tlačítka Acknowledge, Schedule Downtime, Comment a Notification přesunou uživatele na formulář, ve kterém vyplní informace k dané akci.

Příklad takového formuláře je možné vidět na obrázku 11.6.

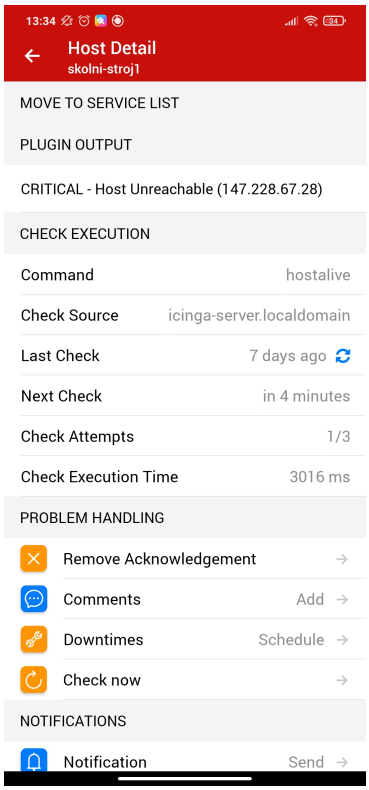
Povinným argumentem je vždy komentář. Ve formuláři k akci Schedule Downtime je potřeba přidat i časový záznam Start Time, Start Date, End

Time, End Date. Bez těchto argumentů není možné formulář odeslat. Pokud uživatel nastaví parametr Type na true, je potřeba vyplnit i položky Hours a Minutes. V případě, že uživatel chce přidat do akce Acknowledge Problem časový záznam expiry, musí zaškrtnout tlačítko Use Expire Time a následně vyplnit informace Time end a Date end.

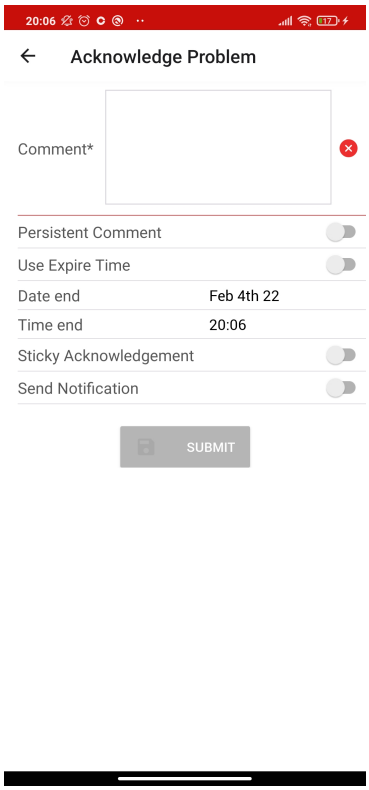
V sekci Notifications je tlačítko, které přesune uživatele do formuláře s komentářem. Tento komentář je pak odeslán jako notifikace do kanálů nastavených pro tato upozornění v nástroji Icinga2.

V případě, že uživatel vyplnil ve formuláři nějakou položku a nechce formulář odeslat, může se vrátit zpět. Před odchodem je ale dotázán, zdali chce doopravdy odejít. V případě, že se vrátí na předchozí obrazovku, jsou vyplněné položky vymazány.

Na obrazovce jsou také zobrazeny existující komentáře a nastavené odstávky v sekcích Existing Comments a Scheduled Downtimes. Ty je možné smazat červeným tlačítkem. Uživatel je nejdříve požádán o potvrzení smazání.



Obrázek 11.5: Obrazovka HostDetail



Obrázek 11.6: Obrazovka AcknowledgeProblem - příklad formuláře