České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačové grafiky a interakce

Bakalářská práce

Webová SMS notifikační služba

Jiří Poláček

Vedoucí práce: Ing. Martin Klíma, Ph.D.

Studijní program: Softwarové technologie a management

Obor: Web a multimédia

20. května 2017

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra počítačové grafiky a interakce

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Jiří Poláček

Studijní program: Softwarové technologie a management
Obor: Web a multimedia

Název tématu: Webová SMS notifikační služba

Pokyny pro vypracování:

1) Analyzujte požadavky na webovou aplikaci pro plánování událostí se SMS upozorněními a navrhněte její funkcionalitu a uživatelské rozhraní.
2) Implementujte webovou aplikaci na základě návrhu. Zaměřte se na využití architektury mikroslužeb.
3) Aplikaci otestujte na úrovni navržených REST rozhraní a pomocí jednotkových testů (unit tests).

Seznam odborné literatury:

1) RICHARDSON, Chris. Pattern: Monolithic Architecture. In: Microservice architecture [online]. Chris Richardson, 2014 [cit. 2017-01-15]. Dostupné z: http://microservices.io/patterns/monolithic.html
2) RICHARDS, Mark. Microservices vs. Service-Oriented Architecture [online]. 1005 Gravenstein Highway North,Sebastopol, CA 95472: O'Reilly Media, 2015 [cit. 2017-01-20]. Dostupné z: http://www.oreilly.com/programming/free/microservices-vs-service-oriented-architecture.csp
3) RICHARDSON, Chris a Floyd SMITH. Microservices: From Design to Deployment [online]. NGINX, 2016 [cit. 2017-01-20]. Dostupné z: https://www.nginx.com/resources/library/designing-deploying-microservices/?utm_source=microservices-from-design-to-deployment-ebook-nginx&utm_medium=blog &utm_campaign=Microservices

Vedoucí: Ing. Martin Klíma, Ph.D.

Platnost zadání: do konce zimního semestru 2018/2019

prof. Ing. Jiří Žára, CSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 18.4.2017

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 20. května 2017 …….

Poděkování

Tímto bych chtěl poděkovat panu Ing. Martinu Klímovi, Ph.D. za cenné rady a připomínky během tvorby této práce. Dále bych chtěl poděkovat rodičům a přítelkyni za jejich podporu a motivaci. Děkuji.

Abstract

The goal of this thesis is to analyze, design and implement a web SMS notification service with the use of microservices architecture. The service is conceived as a calendar which allows sending SMS notifications. Calendar data is shared with Google Calendar service. Notification service consists of five server applications, written in Java. The system sends SMS messages via mobile phone with Android operating system, which the user has previously paired with the service using mobile application.

Abstrakt

Cílem této práce je analýza, návrh a implementace webové SMS notifikační služby s využitím architektury mikroslužeb. Služba je koncipována jako kalendář umožňující odesílání SMS upozornění. Kalendářová data jsou sdílena se službou Google Kalendář. Notifikační služba se skládá z pěti serverových aplikací, které jsou napsány v jazyce Java. Odesílání SMS zpráv probíhá prostřednictvím mobilního telefonu s operačním systémem Android, který si uživatel spároval se službou pomocí mobilní aplikace, a která je součástí této práce.

Obsah

Přílohy

Seznam obrázků

Kapitola 1

Úvod

V současné době existuje velké množství aplikací pro plánování událostí. Od jednoduchých kalendářů s poznámkami, až po sofistikované kalendáře, které v sobě integrují synchronizaci s mobilními telefony, odesílání emailových upozornění, seznamy kontaktů a další funkcionality.

Oblíbenou formou těchto aplikací jsou webové aplikace a aplikace pro chytré telefony, nebo tablety. Webové a mobilní aplikace vynikají svou dostupností. Jsou dostupné z jakéhokoliv počítače s připojením k internetu a webovým prohlížečem. Zatímco mobilní aplikace nosíme doslova v kapse, nebo kabelce.

Od plánovacího kalendáře očekáváme, že nám poskytne informace o naplánovaných událostech a v ideálním případě, pokud nás o nich dokáže předem upozornit. Nejčastější formou upozornění jsou emaily, nebo notifikace na mobilním zařízení. Ne vždy však chceme upozornit sebe, ale někoho jiného. Tato osoba nemusí každou chvíli kontrolovat email a nemusí ani vlastnit chytrý telefon. V tomto případě není možnost, jak včas upozornit tuto osobu o nějaké události. Je spousta dalších případů, kde tyto možnosti nejdou aplikovat. Možným východiskem mohou být SMS upozornění. Tím se okruh osob, kterým lze zaslat upozornění výrazně zvětší.

Na českém trhu existuje velmi málo běžně dostupných a uživatelsky přívětivých služeb pro plánování odesílání SMS zpráv. Ještě méně je pak služeb, které podporují integraci SMS upozornění s kalendářem. Tuto službu poskytovala společnost Google, Inc. Od 27. června 2015 tuto službu Google vypnul a ponechal pouze pro uživatele zpoplatněné služby Google Apps.

Dal jsem si za cíl navrhnout službu, která bude mít funkce webového kalendáře s možností odeslání SMS upozornění zvoleným subjektům. Tato služba by našla využití pro osobní použití, nebo pro podnikatelské subjekty. Chtěl bych se soustředit na modularitu, tedy rozdělení do modulů, které budou samostatné aplikace a které budou poskytovat univerzální rozhraní. Tyto moduly se mohou různě implementovat, čímž služba tím získá například nový způsob odesílaní SMS. Díky univerzálnímu rozhraní je možné tyto moduly implementovat v jakémkoli programovacím jazyce podporující toto rozhraní.

Výstupem této práce bude analýza aplikací na trhu, popis požadavků na systém, seznam případů užití, návrh uživatelského rozhraní a popis architektury mikroslužeb. Dále implementace služby a její testování.

Kapitola 2

Analýza

V této kapitole jsou popsány klíčové požadavky na systém, ze kterých se bude odvíjet další postup tvorby. Dále jsou zde zmíněna obdobná řešení, která jsou dostupná na českém trhu. Pro systém SMS kalendáře jsem zvolil architekturu mikroslužeb, která je zde popsána spolu se základními technologiemi, které bude využívat.

2.1 Požadavky

Požadavek lze definovat jako: "Specifikace toho, co má být implementováno". Požadavky jsem v této části rozlišil do dvou skupin:

- Funkční požadavky.
- Kvalitativní požadavky a omezení.

Funkčním požadavkem je formulace toho, jak by se měl systém chovat – popisuje požadovanou funkci aplikace. Kvalitativní požadavky a omezení, často také doslovně překládány z anglického non-functional requirements jako nefunkční požadavky. Jsou omezující podmínky pro danou aplikaci [1].

2.1.1 Funkční požadavky

Následuje výčet funkčních požadavků na systém. Pro zjednodušení zápisu je zde slovo *spravovat* myšleno jako CRUD operace, tedy vytvoření, čtení, upravení a smazání.

- FP01 – Systém umožní uživateli se autentizovat. Uživatel se bude moci přihlásit do systému.
- FP02 – Systém umožní uživateli spravovat své úkoly v kalendáři. Uživatel bude moci spravovat úkoly ve svém kalendáři.
- FP03 – Systém umožní uživateli nastavit SMS notifikace. Uživatel bude moci k budoucím úkolům nastavit SMS notifikace pro jedno, či více mobilních telefonních čísel.
- FP04 – Systém umožní uživateli odeslat SMS zprávu na jedno, či více telefonních čísel.
- FP05 – Systém umožní uživateli prohlížet svůj kalendář. Uživatel bude moci prohlížet svůj kalendář v měsíčním náhledu.
- FP06 – Systém umožní uživateli přidat SMS zařízení.

2.1.2 Kvalitativní požadavky a omezení

Následuje výčet kvalitativní požadavků a omezení.

- KP01 – Aplikace by měla být schopna odeslat alespoň 1000 SMS notifikací denně.
- KP02 – Aplikace musí být dostatečně zabezpečena, aby nedošlo k odcizení citlivých dat, nebo zneužití služby.

- KP03 – Aplikace bude navrhnuta s užitím architektury mikroslužeb. Aplikace bude navrhnuta s jako množina spolupracujících modulů s univerzálním rozhraním. Nebude tedy problém každý modul psát v jiném programovacím jazyce. Jednotlivé moduly mohou být spuštěny na jiných zařízeních.

- KP04 – Aplikace bude dostatečně uživatelsky přívětivá. Aplikace bude poskytovat prostředí, které bude jednoduché na obsluhu.

Naplnění požadavku KP01 se z velké části odvíjí od výkonu zařízení, na kterém bude aplikace nasazena.

2.2 Současná řešení a konkurence

V současné době se na českém trhu vyskytují komerční projekty, které integrují do kalendáře možnost zasílání SMS. Všechny tyto služby jsou placené, a proto uvedené informace reflektují popis služeb z jejich oficiálních webových stránek. Dále jsou pro operační systém Android dostupné aplikace umožňující plánovaní odesílaní SMS, nicméně neintegrují žádný sofistikovaný kalendář a zmíním je pouze okrajově.

2.2.1 Google G Suite

Součástí placené služby Google G Suite pro firemní zákazníky je také služba odesílání SMS upozornění. Text SMS upozornění generuje služba a nelze tento text upravovat. Výhodou je integrace SMS upozornění do služby Kalendář společnosti Google, kterou již využívají miliony lidí na celém světě. Pro zájemce o využití funkce SMS upozornění odpadá nutnost vytvářet nové kalendáře a může použít již stávající. Cena za službu Google G Suite je 4 € za uživatele na měsíc [2].

2.2.2 smsDiary

Služba smsDiary, poskytuje rezervační systém s SMS připomenutím schůzky určený pro lékaře, beauty salóny a kadeřnictví. Služba poskytuje:

- Automatické SMS připomenutí schůzky.
- Kalendáře pro více zaměstnanců.
- Pokladnu s podporou účtování a tisku.
- Agendu klientů.
- Statistiky.

Služba je zpoplatněna měsíčním paušálem 1 198 Kč s DPH a dále poplatkem za každou odeslanou SMS. Cena je od 2,66 Kč/SMS do 3,63 Kč/SMS a odvíjí se podle celkového počtu odeslaných SMS v daném měsíci [3].

2.2.3 Reservio

Služba Reservio je online rezervační systém s SMS připomenutím. Reservio je přímým konkurentem služby smsDiary a láká zákazníky nižší cenou služeb a velmi podobnou funkcionalitou. Služba poskytuje:

- Zaměstnanecké kalendáře.
- SMS notifikace
- Statistiky
- Tisk a export dat

Služba je zpoplatněna měsíčním paušálem dle tarifu za 199 Kč, 399 Kč, nebo 799 Kč. Cena za SMS do České republiky je od 1,20 Kč do 1,50 Kč a odvíjí se podle zakoupeného SMS balíčku [4].

2.2.4 SMS Scheduler: Send Later

Aplikace SMS Scheduler: Send Later slouží pro plánování odesílání SMS. Je určena pro platformu Android. Umožňuje nastavení přesného času odeslání SMS zpráv z mobilního telefonu [5].

2.2.5 Do It Later – Best Scheduler

Aplikace Do It Later – Best Scheduler [6] pro plánování akcí. Je určena pro platformu Android. Aplikace poskytuje:

- Plánování odesílání SMS.
- Plánování odesílání emailu ze služeb Gmail, Hotmail a Yahoo mail.
- Plánování přidávání příspěvků na sociální sítě Facebook a Twitter.
- Plánování úkolů.

2.3 Případy užití

Tato část práce popisuje případy užití funkcí, které systém bude poskytovat. Pro přehlednost jsem použil upravenou šablonu pro případy užití [1] popisující název případu užití, vstupní podmínky, výstupní podmínky, hlavní scénář, alternativní scénáře a při uživatelském vstupu také povinné a nepovinné atributy formulářů.

2.3.1 Uživatelský účet

Zde jsou popsány případy užití týkající se uživatelského účtu.

UC001 – Registrace

Vstupní podmínky:

1. Uživatel musí mít účet Google.
2. Uživatel navštíví webovou stránku systému.

Hlavní scénář:

1. Uživatel zvolí možnost Přihlásit se pomocí účtu Google.
2. Objeví se dialogové okno s přihlášením k účtu Google.
3. Uživatel vyplní údaje a zvolí možnost *Přihlásit se*.
4. Dialogové okno je přesměrováno na upozornění, zda chce uživatel povolit systému přístup k účtu Google.
5. Uživatel potvrdí systému přístup k účtu Google.
6. Zavře se dialogové okno a systém přesměruje uživatele na hlavní stránku SMS kalendáře.

Výstupní podmínky: Uživatel je registrovaný v systému a přihlášen.

Alternativní scénáře:

4. a) Uživatel špatně vyplnil email, nebo heslo.
4. a) 1. Uživatel je informován, že vyplnil špatnou kombinaci jména a hesla. => 3. hlavního scénáře.
5. a) Uživatel nepotvrdil přístup. => Neúspěšný konec.

UC002 – Přihlásit se

Vstupní podmínky:

1. Uživatel musí být registrovaný v systému.
2. Uživatel navštíví webovou stránku systému.

Hlavní scénář:

1. Uživatel zvolí možnost Přihlásit se pomocí účtu Google.
2. Objeví se dialogové okno s přihlášením k účtu Google.
3. Uživatel vyplní údaje a zvolí možnost *Přihlásit se*.
4. Zavře se dialogové okno a systém přesměruje uživatele na hlavní stránku SMS kalendáře.

Výstupní podmínky Uživatel je přihlášen v systému.

Alternativní scénáře:

4. a) Uživatel špatně vyplnil email, nebo heslo.
4. a) 1. Uživatel je informován, že vyplnil špatnou kombinaci jména a hesla. => 3. hlavního scénáře.

UC003 – Odhlásit se

Vstupní podmínky:

1. Uživatel musí být přihlášený v systému.
2. Uživatel musí být na některé ze stránek systému.

Hlavní scénář:

1. Uživatel stiskne tlačítko se svým jménem v pravém horním rohu webové stránky systému.
2. Objeví se seznam možností související s uživatelským účtem.
3. Uživatel zvolí možnost *Odhlásit se*.
4. Uživatel je odhlášen a přeměřován na přihlašovací stránku systému.

Výstupní podmínky: Uživatel je odhlášen ze systému.

UC004 – Zrušit účet

Vstupní podmínky:

1. Uživatel musí být přihlášený v systému.
2. Uživatel musí být na některé ze stránek systému.

Hlavní scénář:

1. Uživatel stiskne tlačítko se svým jménem v pravém horním rohu webové stránky systému.
2. Objeví se seznam možností související s uživatelským účtem.
3. Uživatel zvolí možnost *Nastavení*.
4. Objeví se dialogové okno s nastavením účtu.
5. Uživatel zvolí možnost *Zrušit účet*.
6. Systém se zeptá uživatele, zda si je svou volbou jistý.
7. Uživatel potvrdí volbu tlačítkem *Ano*.

Výstupní podmínky: Uživatelský účet je zrušený.

Alternativní scénáře:

7. a) Uživatel zruší volbu tlačítkem *Ne* => Neúspěšný konec.

2.3.2 Správa úkolu

Zde jsou popsány případy užití týkající se úkolu.

UC005 – Vytvořit úkol

Vstupní podmínky:

1. Uživatel musí být přihlášený v systému.
2. Uživatel musí být na hlavní stránce systému.

Hlavní scénář:

1. Uživatel aktivuje dropdown menu *Přidat +*.
2. Objeví se seznam položek událostí.
3. Uživatel zvolí možnost *Úkol*.
4. Objeví se dialogové okno s formulářem pro nový úkol.
5. Uživatel vyplní povinné atributy formuláře a případně volitelné atributy.
6. Uživatel přidá nový úkol volbou *Přidat*.
7. Systém informuje uživatele, že nový úkol byl vytvořen.
8. Systém uzavře dialogové okno.

Výstupní podmínky: Je vytvořen nový úkol.

Alternativní scénáře:

6. a) Uživatel zanechal povinná pole prázdná.
6. a) 1. Uživatel je informován, že nevyplnil všechna povinná pole. => 5. hlavního scénáře.

Povinné atributy úkolu:

- Název – Název úkolu.
- Termín splnění.
- Informace, zda má systém zaslat SMS upozornění.

V případě zaškrtnutí checkboxu *Zaslat SMS upozornění* musí uživatel vyplnit následující atributy:

- Čas upozornění
- Text zprávy SMS upozornění
- Zařízení, ze kterého má být SMS upozornění odesláno.
- Příjemci zprávy.

Volitelné atributy úkolu:

- Popis

UC006 – Prohlížet úkol

Vstupní podmínky:

1. Uživatel musí být přihlášený v systému.
2. Uživatel musí být na hlavní stránce systému.

Hlavní scénář

1. Uživatel vyhledá v náhledu kalendáře den, na který je úkol naplánován.

2. Uživatel klikne na úkol.
3. Objeví se dialogové okno s atributy zvoleného úkolu.

Výstupní podmínky: Úkol je zobrazen.

UC007 – Upravit úkol

Vstupní podmínky:

1. Uživatel musí být přihlášený v systému.
2. Uživatel musí být na hlavní stránce systému.

Hlavní scénář:

1. Uživatel vyhledá v náhledu kalendáře den, na který je úkol naplánován.
2. Uživatel klikne na úkol.
3. Objeví se dialogové okno s atributy zvoleného úkolu.
4. Uživatel upraví atributy.
5. Uživatel zvolí tlačítko *Uložit*.
6. Dialogové okno se zavře.

Výstupní podmínky Úkol je upraven.

Alternativní scénáře:

5. a) Uživatel zanechá povinná pole prázdná.
5. a) 1. Uživatel je informován, že nevyplnil všechna povinná pole. => 7. hlavního scénáře.
5. b) Uživatel zvolí tlačítko *Zrušit*. => Neúspěšný konec.

UC008 – Smazat úkol

Vstupní podmínky:

1. Uživatel musí být přihlášený v systému.
2. Uživatel musí být na hlavní stránce systému.

Hlavní scénář:

1. Uživatel vyhledá v náhledu kalendáře den, na který je úkol naplánován.
2. Uživatel klikne na úkol.
3. Objeví se dialogové okno s atributy zvoleného úkolu.
4. Uživatel zvolí možnost *Smazat*.
5. Systém zobrazí upozornění, že úkol bude nenávratně smazán.
6. Uživatel potvrdí smazání tlačítkem *Ano*.

Výstupní podmínky: Úkol je smazán.

Alternativní scénáře:

6. a) Uživatel zruší smazání tlačítkem *Ne*. => 4. hlavního scénáře.

2.3.3 SMS zpráva

UC009 – Odeslat SMS zprávu

Vstupní podmínky:

1. Uživatel musí být přihlášený v systému.
2. Uživatel musí být na hlavní stránce systému.

3. Uživatel přidal alespoň jedno SMS zařízení.

Hlavní scénář:

1. Uživatel aktivuje dropdown menu *Přidat* +.
2. Objeví se seznam položek událostí.
3. Uživatel zvolí možnost *SMS zpráva*.
4. Objeví se dialogové okno s formulářem pro novou SMS zprávu.
5. Uživatel vyplní povinné atributy formuláře a případně volitelné atributy.
6. Uživatel odešle zprávu pomocí tlačítka *Odeslat*.
7. Systém informuje uživatele, že SMS zpráva byla odeslána.
8. Systém uzavře dialogové okno.
9. Výstupní podmínky: Zpráva je odeslána.

Alternativní scénáře:

6. a) Uživatel vyplní termín odeslání a zvolí možnost *Odeslat později*.
6. a) 1. SMS zpráva se uloží pro pozdější odeslání.
6. a) 2. Systém ve zvolený čas odešle SMS zprávu.
6. b) Uživatel zanechá povinná pole prázdná.
6. b) 1. Uživatel je informován, že nevyplnil všechna povinná pole. => 5. hlavního scénáře.

Povinné atributy SMS zprávy:

- Text SMS zprávy.
- Neprázdná množina adresátů.
- SMS zařízení, ze kterého se má zpráva/zprávy odeslat.

2.3.4 Správa SMS zařízení

UC010 – Přidat nové SMS zařízení

Vstupní podmínky:

1. Uživatel musí být registrovaný v systému.
2. Uživatel musí vlastnit telefon s OS Android a GSM modulem.

Hlavní scénář:

1. Uživatel na svém zařízení nainstaluje aplikaci pro odesílání SMS ze systému.
2. Uživatel spustí nainstalovanou aplikaci.
3. Uživatel stiskne tlačítko *Přihlásit se*.
4. Aplikace zobrazí přihlašovací formulář k účtu Google.
5. Uživatel vyplní formulář.
6. Uživatel stiskne tlačítko *Přidat zařízení*.
7. Aplikace ověří přihlašovací údaje.
8. Aplikace kontaktuje systém SMS kalendáře a autorizuje SMS zařízení.

Výstupní podmínky: SMS zařízení je přidáno.

Alternativní scénáře:

6. a) Uživatel špatně vyplnil email, nebo heslo.
6. a) 1. Uživatel je informován, že vyplnil špatnou kombinaci jména a hesla. => 4. hlavního scénáře.

UC011 – Prohlížet SMS zařízení

Vstupní podmínky:

1. Uživatel musí být přihlášený v systému.
2. Uživatel musí být na hlavní stránce systému.
3. Uživatel musí mít přidané alespoň jedno SMS zařízení.

Hlavní scénář:

1. Uživatel stiskne tlačítko se svým jménem v pravém horním rohu webové stránky systému.
2. Objeví se seznam možností související s uživatelským účtem.
3. Uživatel zvolí možnost *Nastavení*.
4. Objeví se seznam přidaných SMS zařízení.

Výstupní podmínky: SMS zařízení jsou zobrazena.

UC012 – Odebrat SMS zařízení, webové rozhraní

Vstupní podmínky:

1. Uživatel musí být přihlášený v systému.
2. Uživatel musí být na hlavní stránce systému.
3. Uživatel musí mít přidané alespoň jedno SMS zařízení.

Hlavní scénář:

1. Uživatel stiskne tlačítko se svým jménem v pravém horním rohu webové stránky systému.
2. Objeví se seznam možností související s uživatelským účtem.
3. Uživatel zvolí možnost *Nastavení*.
4. Objeví se seznam přidaných SMS zařízení.
5. Uživatel zvolí možnost *Odebrat*.
6. Systém zobrazí upozornění, že SMS zařízení bude odebráno.
7. Uživatel potvrdí smazání tlačítkem *Ano*.

Výstupní podmínky: SMS zařízení je odebráno.

Alternativní scénáře:

7. a) Uživatel zruší smazání tlačítkem *Ne*. => 4. hlavního scénáře.

UC013 – Odebrat SMS zařízení, aplikace

Vstupní podmínky:

1. Uživatel musí mít přidané alespoň jedno SMS zařízení.

Hlavní scénář:

1. Uživatel otevře aplikaci pro odesílání SMS.
2. Aplikace zobrazí přihlašovací formulář k účtu Google.
3. Uživatel vyplní formulář.
4. Uživatel stiskne tlačítko *Odebrat zařízení*.
5. Aplikace ověří přihlašovací údaje.
6. Aplikace kontaktuje systém SMS kalendáře a odebere SMS zařízení.

Výstupní podmínky: SMS zařízení je odebráno.

Alternativní scénáře:

4. a) Uživatel špatně vyplnil email, nebo heslo.

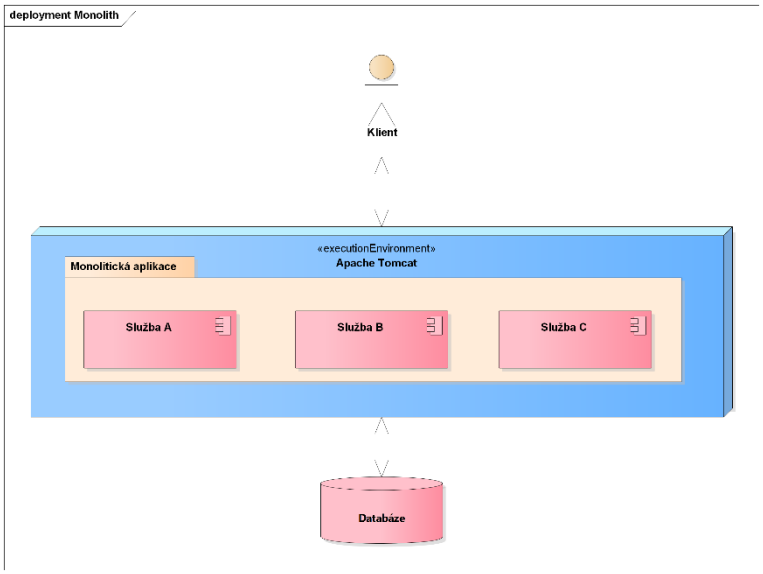
4. a) 1. Uživatel je informován, že vyplnil špatnou kombinaci jména a hesla. => 4. hlavního scénáře.

2.4 Architektura mikroslužeb

Architektura mikroslužeb se zaměřuje na realizaci softwarových systémů jako balíčku malých služeb. Každá služba je samostatně spustitelná aplikace. Služby mezi sebou nejčastěji komunikují pomocí rozhraní REST, nebo jinými RPC technologiemi. Za předpokladu, že služba dodrží dohodnuté komunikační rozhraní, může být napsána v libovolném jazyce a spuštěna na jakékoliv platformě [7] [8] [9].

2.4.1 Monolitické aplikace a mikroslužby

Pojem monolitická aplikace představuje architekturu aplikace, která je vyvíjena jako jeden celek. Obvykle se tyto aplikace skládají ze třech hlavních částí: uživatelského rozhraní na straně klienta, databáze, a serverové aplikace. Zjednodušeně pak serverová aplikace přijme HTTP požadavek, zpracuje ho, získá, nebo aktualizuje data v databázi a odešle HTML soubor zpět klientovi. Tato serverová aplikace je monolitická – jeden spustitelný soubor [8].



Obrázek 1- Typická architektura monolitické aplikace [8]

Časté problémy monolitických aplikací [10]:

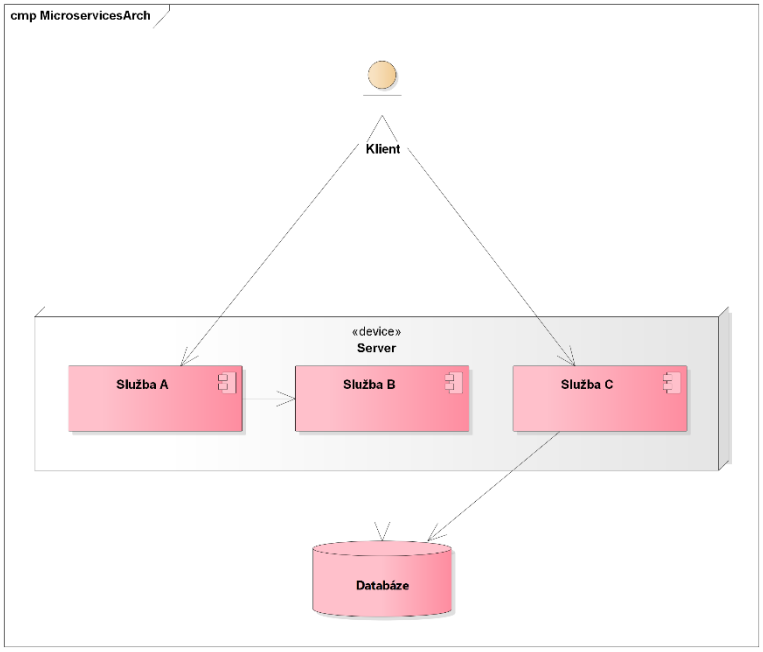
- Součástí jakékoliv změny aplikace je překládání a nasazení nové verze serverové aplikace.
- Škálování aplikace znamená replikaci celé aplikace a nikoliv částí, které jsou více využívány.
- Závislost na jedné platformě.
- Velké projekty mohou zpomalovat vývojové nástroje.
- Velkým aplikacím trvá dlouho, než se na serveru spustí.
- V případě upgradu, nebo opravě chyb se mohou přerušit úkoly běžící na pozadí.

Obrázek 2 - Škálování monolitických aplikací [8]

Výhody monolitických aplikací:

- Nasazení na server znamená nasazení jedné aplikace.
- Podpora ze strany vývojových nástrojů.

Architektura mikroslužeb nemá formální definici, a proto jsou zde popsané obvyklé charakteristiky této architektury. Architektura mikroslužeb oproti monolitické architektuře přináší možnost rozdělení aplikace do více částí. Každá tato část se nazývá mikroslužba (nebo jen služba) a představuje samostatně spustitelnou aplikaci, která komunikuje s ostatními službami. Jednotlivé služby lze nezávisle nahradit. Komunikace nejčastěji probíhá pomocí HTTP protokolu, nicméně lze využít jakýkoliv jiný vhodný protokol. Služby mají nejčastěji právě jednu zodpovědnost. Služby lze vyvíjet v programovacích jazycích, které podporují komunikační protokol. V případě HTTP protokolu lze mít jednu službu naprogramovanou v jazyku Java a druhou například v jazyku Ruby [8].



Obrázek 3 - Typická architektura mikroslužeb [8]

Časté problémy architektury mikroslužeb [11]:

- Komunikace mezi službami je pomalejší než komunikace v rámci monolitické aplikace. Každá zpráva musí být kódována, odeslána, přijata a dekódována.
- Každá služba musí poskytovat rozhraní s definovaným formátem zpráv pro ostatní služby. Služby, které toto rozhraní mají využívat, musí znát tento formát zpráv.
- Návrh komunikačních rozhraní přináší práci navíc.

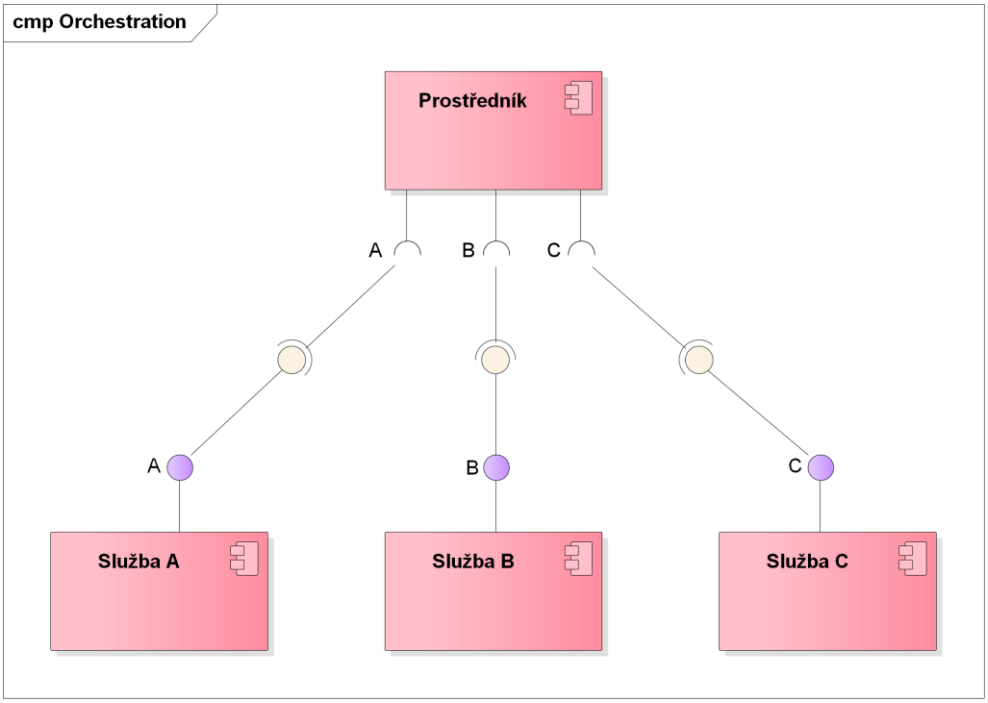
Výhody architektury mikroslužeb:

- Škálovat lze jen nejvíce náročné služby, čímž je škálování efektivnější než v případě monolitických aplikací.
- Nezávislost na platformě. Služby lze programovat v jiných jazycích.
- Při údržbě aplikace není potřeba nasadit celou aplikaci, ale jen její část.
- Možnost automatizace nasazení služeb na server.

Obrázek 4 - Škálování mikroslužeb [8]

2.4.2 Orchestrace a choreografie (Choreography over orchestration)

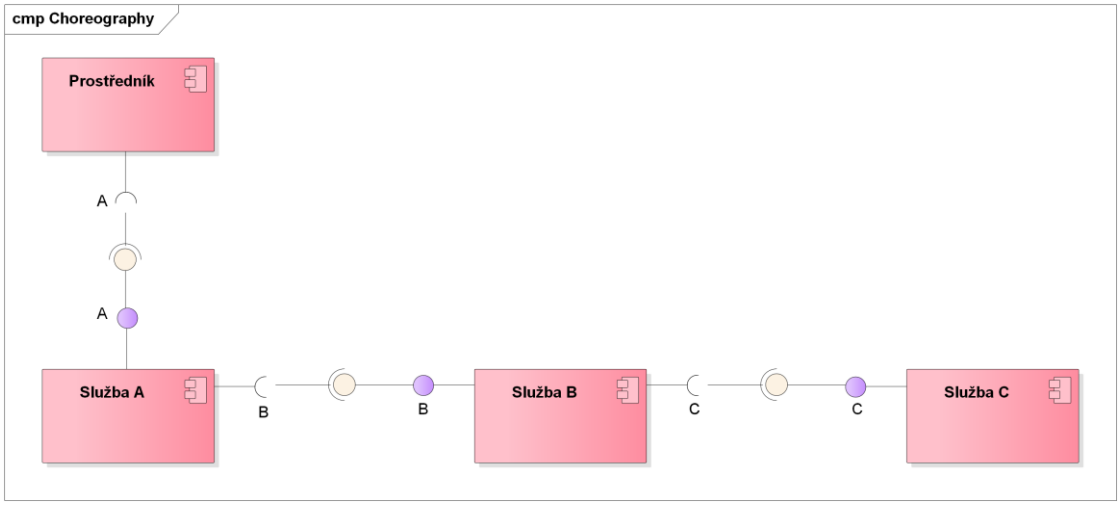
Výraz orchestrace v kontextu mikroslužeb znamená koordinaci více služeb prostřednictvím centrálního prostředníka.



Obrázek 5 - Orchestrace služeb [12]

Orchestrace služeb se dá přirovnat k hudebnímu orchestru. Určitý počet hudebníků hraje na různé hudební nástroje v různé době. Tito hudebníci jsou koordinovaní dirigentem. Stejným způsobem služba prostředníka koordinuje volání všech služeb potřebných pro vykonání business transakce [12].

Choreografie služeb znamená koordinaci více služeb bez centrálního prostředníka. S choreografií služeb služba volá jinou službu, která může volat další službu atd., vytvářejíc takzvané řetězení služeb [12].



Obrázek 6 - Choreografie služeb [12]

Architektura mikroslužeb upřednostňuje choreografii, namísto orchestrace. Primárně protože topologie architektury mikroslužeb postrádá centrální middleware komponentu [12].

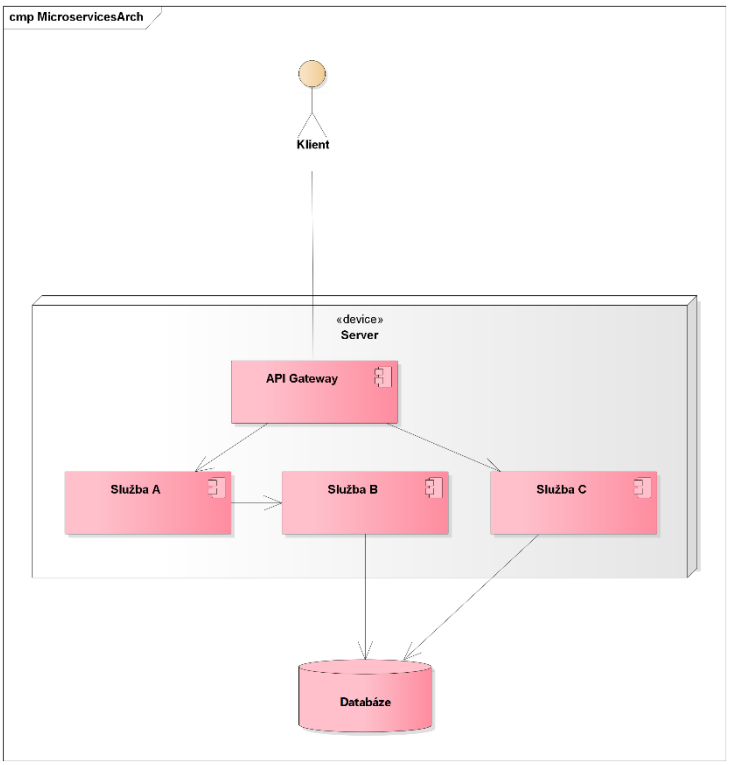
2.4.3 API Gateway

Návrhový vzor architektury mikroslužeb API Gateway, který porušuje pravidlo Choreography over orchestration [2.5.2]. Na druhou stranu přínosy tohoto vzoru převyšují nad jeho nevýhodami.

Problémy:

1. Je potřeba autentizovat/autorizovat uživatele.
2. Při aktualizaci se jedno, nebo více rozhraní změnilo.
3. Uživatel přistupuje k systému více způsoby (webový prohlížeč, nativní aplikace atd.).

Myšlenkou vzoru je vytvoření služby API Gateway, která slouží jako výchozí brána systému, která zpracovává požadavky uživatele a které dále deleguje příslušným službám systému. Tato služba může provádět autorizaci a autentizaci uživatele a tím ušetřit čas zpracování požadavku oproti autorizaci a autentizaci v každé jednotlivé službě. V případě změny některého rozhraní služby v systému odpadá nutnost aktualizace SMS aplikace na Android zařízení uživatele. Při použití API Gateway není vývojář omezený na použití technologie rozhraní jednotlivých služeb podporující webové prohlížeče a nativní aplikace. Komunikace za službou API Gateway je úplně odstíněna od uživatele. Hlavní nevýhoda tohoto vzoru je nutnost návrhu a realizace další služby [13] [14].



Obrázek 7- API Gateway [14]

2.4.4 Persistence

Při použití architektury mikroslužeb je nutností rozhodnout, jakou databázovou architekturu použít. V případě relačních databázových systémů se nabízí několik možností [15] [16]:

- Sdílená databáze – každá služba má přístup k datům jiné služby.
- Sdílená databáze – každá služba má vlastní privátní tabulky.
- Sdílená databáze – každá služba má vlastní privátní schéma.
- Vlastní databáze – každá služba má vlastní databázový systém.

V případě použití sdílené relační databáze, lze využívat ACID transakce. Vlastní databáze pro každou službu se jeví jako vhodný přístup, pokud potřebujeme kombinovat jiné druhy databází [17].

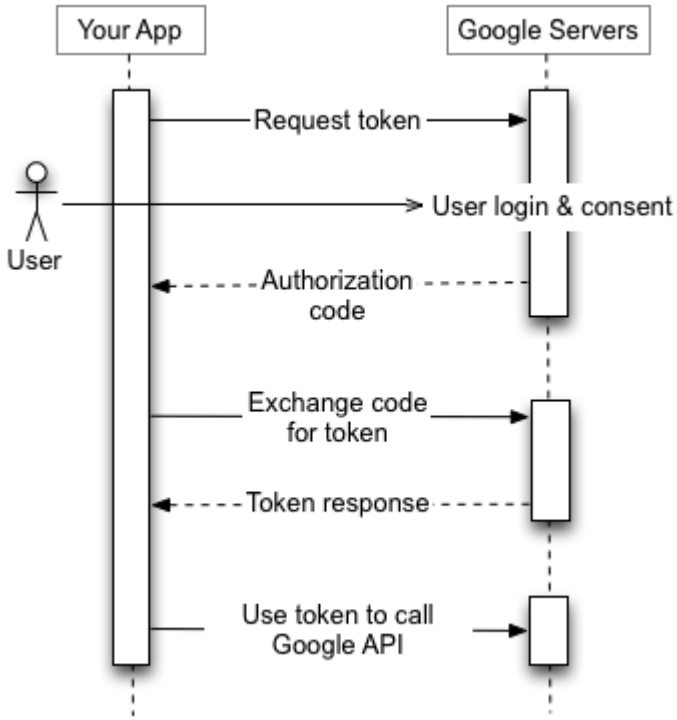
2.5 Google API

Google API jsou programové rozhraní vyvinuté společností Google, které umožňují komunikaci se službami společnosti Google. Aplikace mohou tato API využívat pro jejich rozšíření, nebo naopak pro rozšíření služeb Google. Rozhraní využívají architekturu rozhraní REST a výměna dat probíhá ve formátech JSON a XML. [18].

2.5.1 Google OAuth 2.0

Rozhraní od společnosti Google používají protokol OAuth 2.0 pro autentizaci a autorizaci. Aby systém mohl přistupovat k uživatelským datům prostřednictvím Google API, je nutné získat přístupový token, který opravňuje k přístupu k těmto datům. Jeden přístupový token může umožňovat přístup k různým API s různými přístupovými právy. Ve webových aplikacích lze tento token získat přesměrováním na Google, který zajistí proces přihlašování a potvrzení přístupových práv tokenu. Získaný token se následně odešle zabezpečeným spojením do systému,

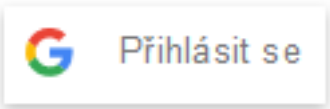
který následně bude tento token využívat k přístupu. Tokeny mají omezenou dobu použití. Pokud doba vyprší, může systém zažádat o takzvaný *refresh token*. Refresh token umožňuje získat nový přístupový token [19].



Obrázek 8 - Proces autorizace [19]

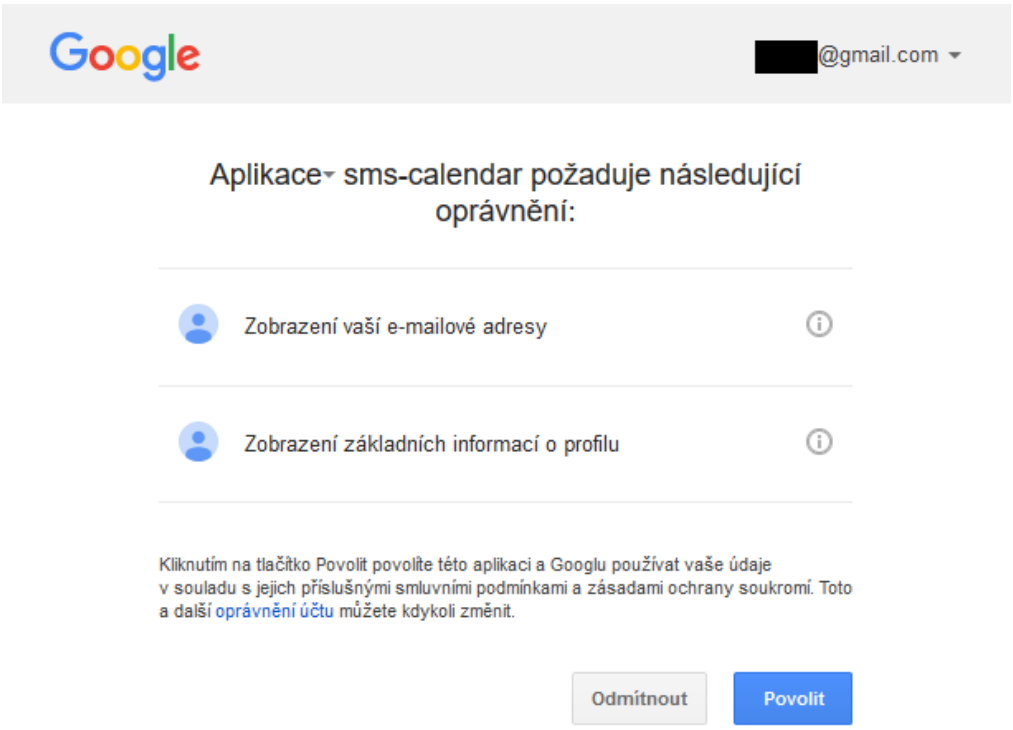
2.5.2 Google Sign-In

Google Sign-In autentizační systém sloužící k přihlašování uživatelů k systémům pomocí účtu Google. Pro webové aplikace existuje knihovna v jazyce Javascript – Google Platform Library, pomocí které lze provádět základní operace s uživatelským účtem Google. Pro použití Google Sign-in je potřeba získat client ID z vývojářské konzole Google, které bude sloužit k identifikaci systému [20]. Pro úspěšnou autentifikaci se klient přihlásí pomocí vloženého přihlašovacího tlačítka.



Obrázek 9 – Screenshot přihlašovacího tlačítka

Po zadání přihlašovacích údajů je při prvním přihlášení uživatel dotázán, zda souhlasí s udělením oprávnění pro aplikaci k informacím o účtu. Kromě základních informací o profilu a emailové adresy lze také vyžadovat oprávnění k jiným službám, jako jsou například Google Kalendář a Google Kontakty.



Obrázek 10 – Screenshot udělení oprávnění přístupu k účtu

Po úspěšném přihlášení lze pomocí Google Platform Library získat uživatelův ID token implementací funkce *onSignIn()* [21]:

```
function onSignIn(googleUser) {
  var id_token = googleUser.getAuthResponse().id_token;
  ...
}
```

Následně zašleme token na server aplikace [21]:

```
var xhr = new XMLHttpRequest();
xhr.open('POST', 'https://yourbackend.example.com/tokensignin');
xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
xhr.onload = function() {
  console.log('Signed in as: ' + xhr.responseText);
};
xhr.send('idtoken=' + id_token);
```

Server aplikace přijme token ID, ověří jeho integritu a získá uživatelské ID ze serveru Google.

2.5.3 Google Calendar API

Calendar API je REST rozhraní umožňující integrace aplikací se službou Google Kalendář. Umožňuje spravovat události.

K rozhraní lze přistupovat pomocí HTTP volání, nebo použití knihoven, které jsou k dispozici pro platformy .NET, Android, Go, Google Apps Script, iOS, Java, Javascript, Node.js, PHP, Python a Ruby. Data, které toho rozhraní odesílá a přijímá, jsou ve formátu JSON [22].

Rozhraní poskytuje 7 zdrojů [23]:

- Acl – Pravidla přístupu k danému kalendáři
- CalendarList – Seznam všech kalendářů.
- Calendars – Metadata kalendáře.
- Colors – Množina barev dostupných pro kalendáře a události.
- Events – Události kalendáře.
- Freebusy – Informace, o dostupnosti v daném čase.
- Settings – Uživatelská nastavení.

Událostem, které přidáváme do kalendáře pomocí Calendar API, je možné přidávat vlastní data pomocí *extended properties*. Jsou to parametry ve tvaru klíč-hodnota. Tyto data mohou být přístupná dalším aplikacím klienta, nebo mohou být skryta a viditelná pouze naší aplikaci. Klíč může mít maximální délku 44 znaků a hodnota maximální délku 1024 znaků. Delší hodnoty jsou bez upozornění zkráceny. Tuto možnost lze například v případě systému SMS kalendáře využít pro ukládání typu události (úkol, schůzka, naplánovaná SMS zpráva).

Calendar API je limitováno na 1 000 000 požadavků denně pro jednu aplikaci [24]. Pro vyšší počet požadavků je nutné zažádat prostřednictvím formuláře na webové stránce [https://support.google.com/code/contact/calendar_api_quota].

2.5.4 Google People API

People API je REST rozhraní umožňující integrace aplikací se službou Google Kontakty. Umožňuje přistupovat ke kontaktům a číst jejich informace.

K rozhraní lze přistupovat podobně jako v případě Google Calendar API, a to pomocí explicitních HTTP volání, nebo pomocí knihoven, které jsou k dispozici pro platformy Java, .NET, PHP a Python. Rozhraní poskytuje a přijímá data ve formátu JSON [25].

Rozhraní poskytuje zdroje [25]:

- Get – Poskytuje informace o kontaktu/osobě.
- List – Poskytuje seznam kontaktů/osob.

2.6 Android SMS Manager

GSM přístroje s operačním systémem Android verze 1.6 a vyšší podporují funkci odesílání SMS zpráv prostřednictvím Android API rozhraní SmsManager.

SmsManager umožňuje rozdělení textu SMS zprávy na části, které délkou nepřesahují maximální délku SMS zprávy. Dále umožňuje odesílání MMS zpráv [26].

2.7 OneSignal

OneSignal je služba pro doručování push notifikací do mobilních zařízení s operačním systémem Android, iOS, Windows Mobile, Amazon Fire. Služba je poskytována zdarma pro neomezený počet zařízení a notifikací [27].

Služba OneSignal se jeví jako vhodná pro informování mobilního zařízení o připravených SMS zprávách, které jsou určené k odeslání.

2.8 Quartz Job Scheduler

Quartz Job Scheudler je open source knihovna pro plánování událostí v Java aplikacích. Je vydávána pod licencí Apache 2.0 Licence. Může být použita v jednoduchých aplikacích i ve velkých podnikových systémech [28].

Základní rozhraní Quartz API [28]:

- Scheduler – Hlavní API pro interakci a plánovaní.
- Job – Rozhraní, které má implementovat komponenta, která bude spuštěna Schedulerem.
- JobDetail – Používané pro definice instancí typu Job.
- Trigger – Komponenta, která definuje Scheduler, pod kterým bude Job spuštěna.
- JobBuilder – Používané pro definici instancí typu JobDetail, které definují instance typu Job.
- TriggerBulider – Používané pro definici instancí typu Trigger.

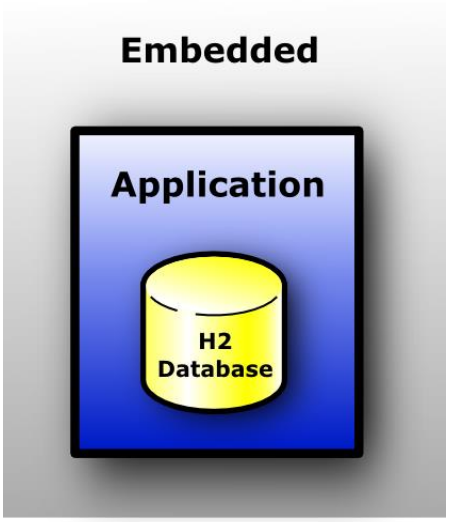
Pro ukládání naplánovaných událostí lze využít RAMJobStore, který ukládá data do paměti RAM. Když se aplikace vypne, ať už cíleně, nebo z důvodu chyby, jsou naplánované události ztraceny. Další možností je JDBCJobStore, který ukládá data do databáze prostřednictvím JDBC API. Pro použití JDBCJobStore je nutné v databázi připravit tabulky pomocí skriptu dodaného s knihovnou [28].

Quartz Job Scheduler se zdá být dobrou knihovnou pro zajištění odesílání SMS notifikací ve správný čas.

2.9 H2 Database

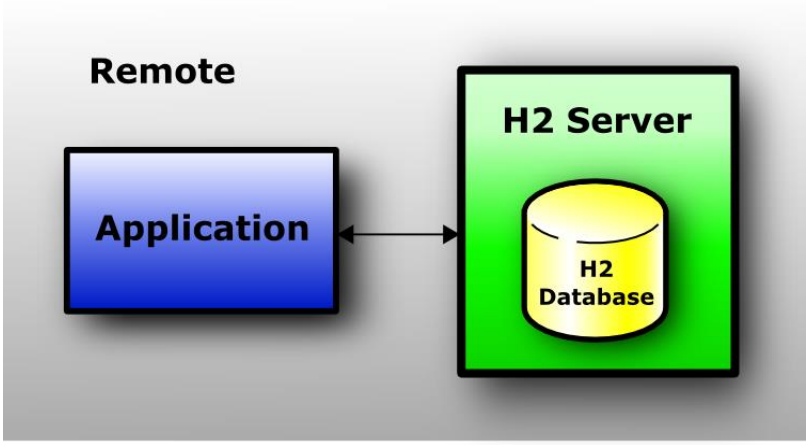
H2 je relační databázový systém napsaný v jazyce Java. Je vydávaný pod licencí Mozilla Public Licence Version 2.0, nebo Eclipse Public Licence. Systém H2 lze spustit ve třech režimech.

V Embedded režimu je H2 systém spuštěn ve stejném JVM, jako aplikace, která ji využívá. V tomto režimu je spojení aplikace a databáze nejrychlejší. [29]



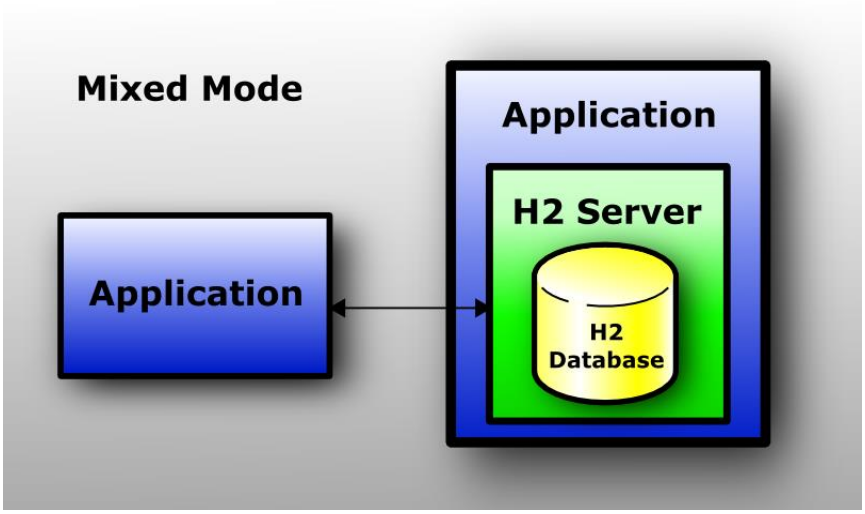
Obrázek 11 - Embedded režim H2 databáze [29]

V server režimu aplikace přistupuje k databázi vzdáleně pomocí JDBC, nebo ODBC API. K databázi tedy může přistupovat přímo více aplikací. V server režimu jsou požadavky na databázi zpracovány pomaleji, protože data jsou přenášena přes protokol TCP/IP [29].



Obrázek 12 - Server režim H2 databáze [29]

Mixed režim je kombinací Embedded a Server režimu. Jedna aplikace spustí uvnitř sebe H2 systém v embedded režimu a také spustí server, ke kterému se mohou připojit další aplikace a přistupovat tak k datům [29].



Obrázek 13 - Mixed režim H2 databáze [29]

Pro více možností režimů spuštění a malé velikosti okolo 1,5 MB, může být tento databázový systém vhodný pro architekturu mikroslužeb.

Kapitola 3

Návrh

V této kapitole je popsán návrh uživatelského rozhraní a architektury systému SMS kalendáře. Návrhy uživatelského rozhraní jsem vytvářel v programu Affinity Designer jako vektorovou kresbu. Pro návrh diagramu architektury jsem využil program Enterprise Architect.

3.1 Uživatelské rozhraní

Uživatelské rozhraní aplikace cílí na uživatele přistupující k systému z osobního počítače. Vytvořil jsem sadu detailních wireframů, které poslouží jako výchozí materiál pro tvorbu finální podoby uživatelského rozhraní.

Protože je aplikace z pohledu klienta koncipována jako single-page aplikace (SPA), je přihlášený uživatel vždy na hlavní stránce aplikace.

V horní části uživatelského rozhraní se nachází hlavní panel obsahující prostor pro logo aplikace a tlačítko se jménem aktuálně přihlášeného uživatele. V levé části hlavního panelu se nachází prostor pro logo aplikace, které zastupuje nápis *LOGO*. V pravé části hlavního panelu je tlačítko, které po kliknutí vyvolá kontextovou nabídku s volbami pro vyvolání dialogového okna pro nastavení, nápovědu a pro odhlášení uživatele.

Pod hlavním panelem vlevo se nachází tlačítko s textem *Přidat*, které po stisknutí vyvolá kontextovou nabídku s volbami pro vytvoření nové události a vytvoření nové SMS zprávy.

Napravo od tlačítka *Přidat* se nachází text s aktuálně zvoleným kalendářním měsícem a rokem. Z levé a pravé strany tohoto textu jsou tlačítka pro sekvenční procházení mezi kalendářními měsíci. Stisknutím tlačítka z levé strany se zvolí jako aktuální předcházející měsíc a stisknutím tlačítka z pravé strany se zvolí jako aktuální následující měsíc. Úplně vpravo pod hlavním panelem se nachází informace, kolik se v aktuálním kalendářním měsíci má odeslat SMS zpráv.

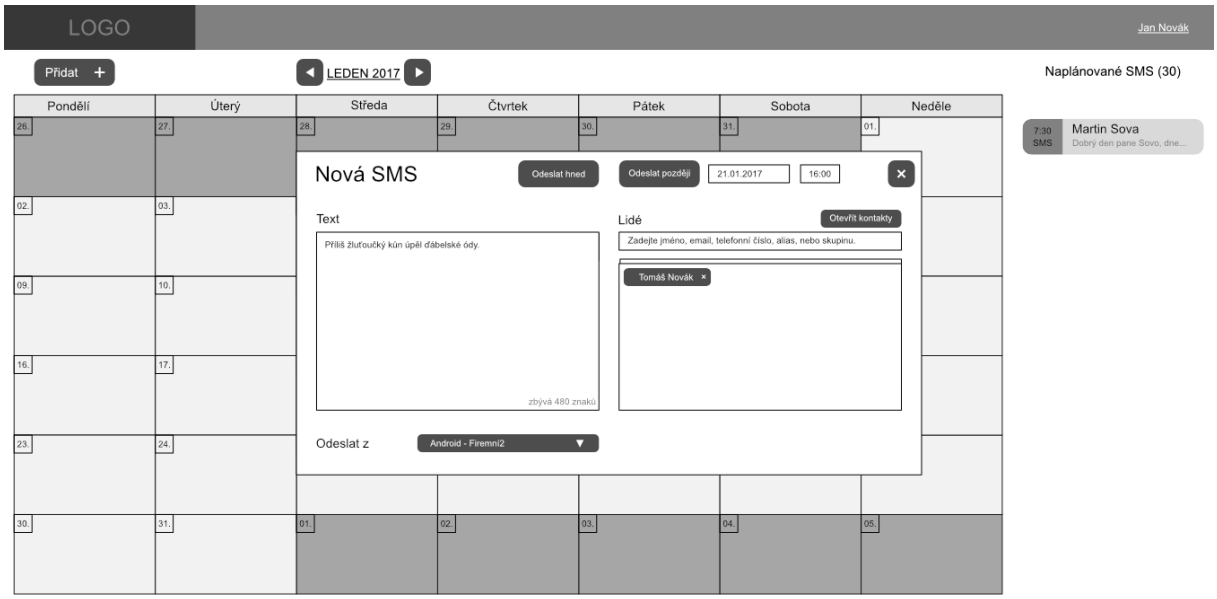
Nejvíce prostoru na hlavní stránce zabírá samotný náhled kalendáře. Ten je rozdělený do tabulky, ve které řádky představují jednotlivé týdny měsíce. Dny, které nejsou součástí měsíce, ale jsou obsaženy v tabulce jsou barevně odlišeny. Událost je v kalendáři vyobrazena jako barevně odlišený pruh obsahující čas a název a je umístěna v příslušné buňce tabulky kalendáře. Každá buňka kalendáře obsahuje informaci, který den v měsíci zastupuje a dále obsahuje události, pokud existují.

Napravo od náhledu kalendáře je prostor, pro naplánované SMS. Tento prostor obsahuje seznam SMS zpráv určených k odeslání v aktuálně zvoleném kalendářním měsíci. Prvek tohoto seznamu obsahuje den odeslání, příjemce a text zprávy.

Nové události kalendáře a SMS zprávy se budou vytvářet v dialogových oknech, jejichž rozložení bude podobné i pro ostatní akce jako výběr kontaktů, nastavení apod. Tato dialogová okna budou jednosloupcová až třísloupcová.

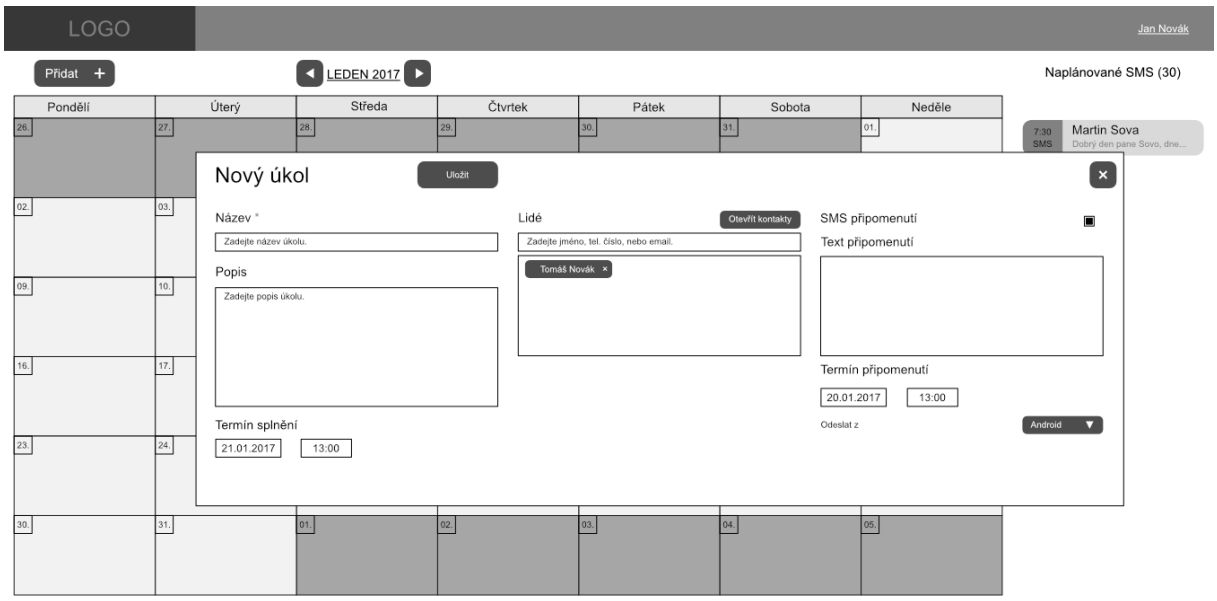
Na obrázku 14 je ukázka dialogového okna pro vytvoření nové SMS zprávy. Toto dialogové okno obsahuje název dialogového okna – *Nová SMS*. Dále obsahuje formulářové prvky pro text zprávy, zadávání příjemců, výběr zařízení, ze kterého se SMS zprávy budou odesílat. Při vytváření nové SMS zprávy má uživatel možnost zprávu odeslat ihned, nebo zvolit v horní části dialogového

okna čas a datum odeslání zprávy. Pod prvkem pro zadávání příjemců se nachází prostor, kde bude seznam již přidaných příjemců.



Obrázek 14 - Příklad dvousloupcového dialogového okna

Na obrázku 15 je dialogové okno pro vytvoření nového úkolu. Dialogové okno opět obsahuje formulářové prvky, kde uživatel zadá název, popis a termín splnění úkolu. U úkolu může uživatel zvolit SMS notifikaci zaškrtnutím políčka napravo od textu *SMS připomenutí*. Po zaškrtnutí tohoto políčka se povolí editace formulářových prvků, které jsou umístěny pod ním. Tj. text SMS zprávy, termín odeslání a zařízení, ze kterého se má zpráva odeslat.



Obrázek 15 - Příklad třísloupcového dialogového okna

Další wireframy jsou umístěny v příloze B.

3.2 Architektura systému

Základní architektura systému se skládá ze sedmi programů – služeb, ze kterých jedna bude pro mobilní telefony s operačním systémem Android a bude zajišťovat odesílání SMS zpráv. Tyto programy budou využívat Google API, které jsem zmínil v kapitole 2. Diagram navrhované architektury je v příloze B.

3.2.1 calendar-gw

Služba calendar-gw slouží jako vstupní brána, prostřednictvím které uživatel komunikuje se systémem. Služba bude se bude starat o obsluhu požadavků uživatele a o autentizaci. Poté, co se uživatel přihlásí do webové aplikace SMS kalednáře, odešle služba veškerá nutná data pro vykreslení aplikace. Tedy kostru HTML stránky, kaskádové styly a potřebné soubory se skripty. Další data jsou načítána dynamicky pomocí AJAX. Služba komunikuje s databází systému a má přístup k entitám USERS a DEVICES viz. Obrázek 16 v kapitole 3.3.

/web/{}* – Handler pro statické soubory zahrnuje všechny HTTP metody s stará se o doručování html souborů, kaskádových stylů, skriptů apod.

/app/{}* – Handler, který ověří, zda je uživatel přihlášen. Pokud je, posílá požadavek dalšímu handleru a pokud není, přesměruje uživatele na přihlašovací stránku.

Veřejné zdroje:

a) **POST** */auth* – Služba získá z těla požadavku přístupový token (accessToken) a requestToken. Dále ověří integritu requestTokenu a accessToken uloží do session. AccessToken bude používat pro volání daších služeb využívající Google API.

b) **GET** */app/logout* – Odhlasí uživatele.

c) **POST** */addDevice* – Přidá nové SMS zařízení, ze kterého byl vyslán požadavek.

d) **POST** */removeDevice* – Odebere SMS zařízení, ze kterého byl vyslán požadavek.

e) **POST** */checkDevice* – Vrátí, zda je zařízení ze kterého byl vyslán požadavek přidaný k účtu ke kterému je uživatel na daném zařízení přihlášen.

f) ***GET*** */home* – Vrátí přihlašovací stránku aplikace

g) ***GET*** */app/calendar* – Vrátí hlavní stránku aplikace

h) **GET** */app/deviceList* – Vrátí seznam SMS zařízení přídaných k účtu.

i) **GET** */app/contactList* – Deleguje požadavek na zdroj b) služby contact-service.

j) **GET** */app/sms/month/{year}/{month}/{page}* – Deleguje požadavek na zdroj f) služby sms-scheduler-service.

k) **GET** */app/eventList/month/{year}/{month}* – Deleguje požadavek na zdroj e) služby calendar-service.

l) **GET** */app/event/{calendarId}/{eventId}* – Deleguje požadavek na zdroj a) služby calendar-service.

m) **POST** */app/event* – Deleguje požadavek na zdroj c) služby calendar-service.

n) **PUT** */app/event/{eventId}* – Deleguje požadavek na zdroj d) služby calendar-service.

o) **DELETE***/app/event/{calendarId}/{eventId}* – Deleguje požadavek na zdroj b) služby calendar-service.

p) **GET** */app/settings* – Vrátí informace o nastavení.

q) **POST** */app/settings* – Uloží upravená nastavení.

3.2.2 contacts-service

Služba contacts-servvice bude delegovat požadavky na službu Google People API. Data získané ze služby Google People API zpracuje a odešle klientovi. Zdroje poskytované službou:

a) **GET** */person/{accessToken}/{id}* – Vrátí detailní informace o kontaktu/osobě dle zadaného id.

b) **GET** */personList/{accessToken}* – Vrátí seznam kontaktů/osob.

3.2.3 calendar-service

Služba calendar-service bude delegovat požadavky na službu Google Calendar API. Data získané ze služby Google Calendar API zpracuje a odešle klientovi. Dále tato služba bude komunikovat se službou sms-scheduler-service. V parametrech zdrojů služby se vyskytuje *{calendarId}*, které je vždy *primary*, protože systém bude pracovat pouze s hlavním kalendářem účtu. Zdroje:

a) **GET** */event/{calendarId}/{eventId}/{accessToken}* – Vrátí informace o události dle zadaného id kalendáře a id události.

b) **DELETE** */event/{calendarId}/{eventId}/{accessToken}* – Odstraní událost a všechny její SMS upozornění.

c) **POST** */event/{calendarId}/{accessToken}* – Vloží událost a upozornění (pokud je zadáno).

d) **PUT** */event/{calendarId}/{eventId}/{accessToken}* – Upraví existující událost.

e) **GET** */eventList/month/{year}/{month}/{accessToken}* – Vrátí seznam všech událostí pro zadaný měsíc.

3.2.4 sms-scheduler-service

Služba sms-scheduler-service bude zodpovědná za odeslání SMS zprávy ve stanovený čas. Služba bude v databázi ukládat SMS k odeslání. Ve stanovený čas bude zpráva odeslána do služby sms-sender-service. Zdroje:

a) **GET** */sms-notification/id/{notificationId}* – Vrátí informace o notifikaci podle identifikačního čísla notifikace.

b) **GET** */sms-notification/event/{eventId}* – Vrátí informace o notifikaci podle identifikačního čísla události.

c) **DELETE** */sms-notification/{notificationId}* – Odstraní notifikaci se zadaným id.

d) **POST** */sms-notificatioin* – Vloží novou notifikaci.

e) **PUT** */sms-notification* – Upraví existující notifikaci.

f) **GET** */sms/month/{userId}/{year}/{month}/{page}* – Vrátí seznam SMS, které mají být odeslány v zadaný rok a měsíc. Výsledky jsou stránkované po dvaceti SMS zpráv, a tedy poslední parametr *page* určuje, kterou stránku vrátit.

Služba bude komunikovat jednak s databází pro notifikace, jejíž konceptuální model je na obrázku 16 v kapitole 3.3. a jednak s databází knihovny Quartz Scheduler, která je její součástí.

3.2.5 sms-sender-service

Služba sms-sender-service bude získávat SMS zprávy k okamžitému odeslání. Tuto službu jsem vyčlenil ze služby sms-scheduler-service, aby bylo možné přidávat více implementací sms-sender-service a tím využít jiné způsoby odesílání SMS. Zdroje:

a) **GET** */notification/{notificationId}/{accessTokenSms}* – Vrátí text SMS zprávy a všechny příjemce.

b) **GET** */notify{deviceId}/{notificationId}* – Odešle do mobilního telefonu upozornění, že SMS zpráva je připravena k odeslání.

První zdroj bude přístupný z internetu a bude ho využívat služba android-sms-sender. Služba bude komunikovat s databází jejíž konceptuální model je na obrázku 16 v kapitole 3.3.

3.2.6 android-sms-sender

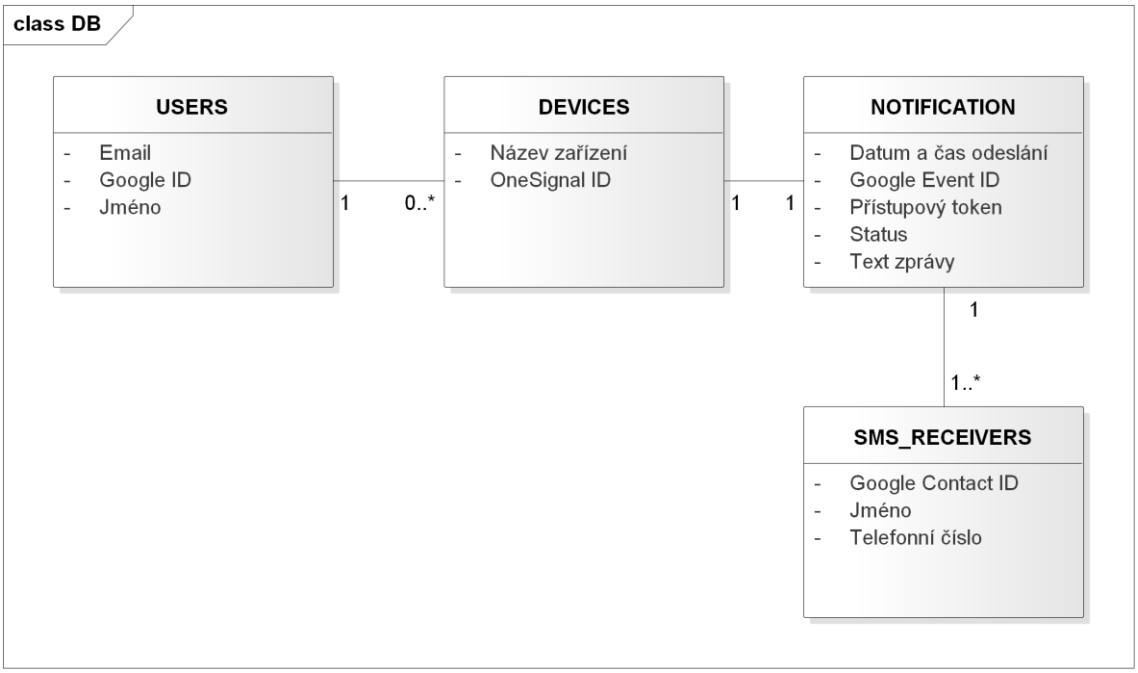
Android-sms-sender je aplikace pro mobilní zařízení se systémem Android. Tato aplikace bude přijímat push notifikace z mikroslužby sms-sender-service prostřednictvím služby OneSignal. Tyto notifikace budou obsahovat informaci, že je SMS zpráva připravena k odeslání. Aplikace po přijmutí této notifikace stáhne zprávu a odešle příjemci.

3.3 Návrh databázového modelu

Konceptuální databázový model zachycuje základní entity SMS notifikací.

Entita *USERS* obsahuje emailovou adresu uživatele, jeho Google ID a jméno. Každá entita *USERS* může mít žádné, nebo více zařízení pro odesílání SMS zpráv. Entita *NOTIFICATION* obsahuje informace o SMS upozornění, text zprávy a jednu, nebo více entit *SMS_RECEIVERS.*

Informace o událostech jsou uloženy na serverech služby Google Kalendář.



Obrázek 16 - Konceptuální model databáze

Kapitola 4

Implementace

4.1 Maven

Pro řízení buildů mikroslužeb jsem zvolil nástroj Maven. V konfiguračním souboru pom.xml jsou umístěné všechny závislosti a informace o službě.

Implicitně Maven kompiluje pomocí výchozí nainstalované verze Javy. V kódu mikroslužeb využívám lambda expressions, diamond operators, a proto vynucuji kompilaci projektů pomocí Javy verze 1.8. K tomu využívám plugin maven-compiler-plugin.

Dále v systému cílím na nezávislost mikroslužeb, a proto jsou všechny závislosti součástí archivu JAR. K tomu využívám plugin maven-shade-plugin.

4.2 Sada nástrojů Vert.x

Základem mikroslužeb se stala sada nástrojů Vert.x, která slouží k vytváření reaktivních systémů. Reaktivním systémem je myšlen takový systém, který odpovídá takzvanému Reaktivnímu manifestu [http://www.reactivemanifesto.org].

V systému SMS Kalendáře jsem využil dvě komponenty: Vert.x Core a Vert.x Web. Core je základní komponenta obsahující podporu pro HTTP, TCP, přístup k souborovému systému a další. Komponenta Web je určená pro tvorbu webových aplikací a HTTP mikroslužeb. Je založená na funkcích komponenty Core. Základním konceptem komponenty Web je Router. Je to objekt, který obsahuje žádné, nebo více Routes. Router převezme HTTP požadavky a předává je příslušným Routes, které požadavek zpracují a buď předají další Route, nebo ho ukončí.

Routes se registrují před spuštěním Router současně s jejich handlerem, který se stará o zpracování požadavku. Parametry jsou metoda požadavku (GET, POST apod.), dále relativní cesta. Volitelně pak content-type a samotný handler. Handler může být asynchronní a využívat vlákna z event loop, nebo blocking handler, který pro zpracování používá vlákna z working pool.

```
router.route(HttpMethod.GET,   "/sms/month/:userId/:year/:month/:page")
        .produces(CONTENT_TYPE_JSON)
        .blockingHandler(monthSMSMessageList::get);
```

V přechozí ukázce kódu je vidět registrace zdroje pro měsíční seznam SMS, které jsou určené k odeslání. Tento zdroj využívá dotazovací metodu GET, na adrese */sms/month/:userId/:year/:month/:page*. Parametry s dvojtečkou jsou dynamické a určují, které SMS zprávy má vrátit jako odpověď. Dále je zde nastaveno, že výsledkem bude odpověď ve formátu JSON.

Implementace tohoto handleru buď po zpracování předá kontext další route, nebo požadavek ukončí a vrátí odpověď.

```
@Override
public void get(RoutingContext routingContext) {
    HttpServerResponse response = routingContext.response();
    HttpServerRequest request = routingContext.request();
    //...
}
```

4.3 Komunikace mezi mikroslužbami

Mikroslužby mezi sebou komunikují pomocí protokolu HTTP. Pro odesílání požadavků využívám knihovnu Okhttp. Metoda, která vytvoří HTTP POST požadavek ze zadané URL adresy a dat.

```
protected Request buildPost(String url, String body){
    RequestBody requestBody =
RequestBody.create(MediaType.parse("application/json"), body);
    Request request = new Request.Builder()
            .post(requestBody)
            .url(url)
            .build();

    return request;
}
```

4.4 Nastavení služeb

U mikroslužeb, které posílají HTTP požadavky jiným mikroslužbám, nebo komunikují s databází využívám Java třídu Properties, která uchovává konfiguraci jako páry klíč/hodnota. Tato konfigurace se serializuje do souboru app.properties. Mikroslužby při startu přečtou obsah souboru a pak periodicky kontrolují změny konfigurace. Díky tomu lze mikroslužby aktualizovat a bez vypínání celého systému.

Příklad postupu aktualizace služby scheduler-service:

1. Nasaďte aktualizovanou verzi scheduler-service.
2. V souboru app.properties služby calendar-service změňte adresu a port na aktualizovanou verzi scheduler-service.
3. V logu calendar-service zkontrolujte, že konfigurace byla aktualizována.
4. Vypněte předchozí verzi scheduler-service.

Pokud je v app.properties nastavena periodická aktualizace s dlouhým intervalem, lze aktualizovat nastavení ručně vysláním HTTP požadavku.

V konfiguračním souboru se také nachází přihlašovací údaje k databázi a další citlivé informace. Je proto nutné zaručit, aby nemohla soubor číst, nebo editovat nepovolaná osoba.

Příklad souboru app.properties služby calendar-gw:

```
debug=true
debug_emails=example@example.com
admin_emails=example@example.com
properties_refresh_seconds=5

#This service preferences
host=localhost
port=1234
#DataSource connection
```

```
data_source_path=jdbc:h2:tcp://localhost/~/database
data_source_name=username
data_source_pass=password
#Google auth clientId
google_client_id=000000000000-
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.apps.googleusercontent.com
#Single paths e.g. http://localhost:1234
calendar-service=http://localhost:1235
scheduler-service=http://localhost:1236
contacts-service=http://localhost:1237
```

Pro aktualizaci nastavení využívám návrhový vzor observer, který pro všechny registrované instance tříd implementující interface IDynamical aktualizuje hodnoty načtené ze souboru app.properties.

```
public void addListeners(IDynamical iDynamical){
    this.iDynamicalListeners.add(iDynamical);
}
private void notifyListeners(){
    for (IDynamical listener : iDynamicalListeners) {
        listener.updateProperties(this.properties);
    }
    notify = false;
}
```

4.5 Komunikace s databází

Pro komunikaci s databázovým systémem jsem zvolil knihovnu JOOQ, která je odlehčenou alternativou pro knihovny pro objektově-relační mapování jako je Hibernate, nebo JPA. JOOQ poskytuje abstrakci nad JDBC a vytváří validní SQL dotazy prostřednictvím doménově specifického jazyka (DSL), který emuluje SQL v jazyku Java. JOOQ poskytuje takzvané fluent API, kde některé metody zavolané nad objektem mají jako návratovou hodnotu instanci objektu, na kterém byla metoda zavolána.

Příklad metody v balíčku DAO služby scheduler-service, která využívá JOOQ a stará se o vymazání informací o SMS notifikaci.

```
/**
 * Deletes information about existing notification
 *
 * @param connection Database connection
 * @param notificationIdPk Internal id of notification
 *
 * @throws DataAccessException Exception thrown while executing
statement
 * */
public static void delete(Connection connection,  Long
notificationIdPk) throws DataAccessException{
    DSL.using(connection)
            .deleteFrom(NOTIFICATION_TABLE)
            .where(ID_PK_FIELD.equal(notificationIdPk))
            .execute();
}
```

4.6 Klientská část aplikace

Klientská část aplikace zahrnuje soubory, které vykresluje a obsluhuje webový prohlížeč uživatele.

4.6.1 Bootstrap a moduly

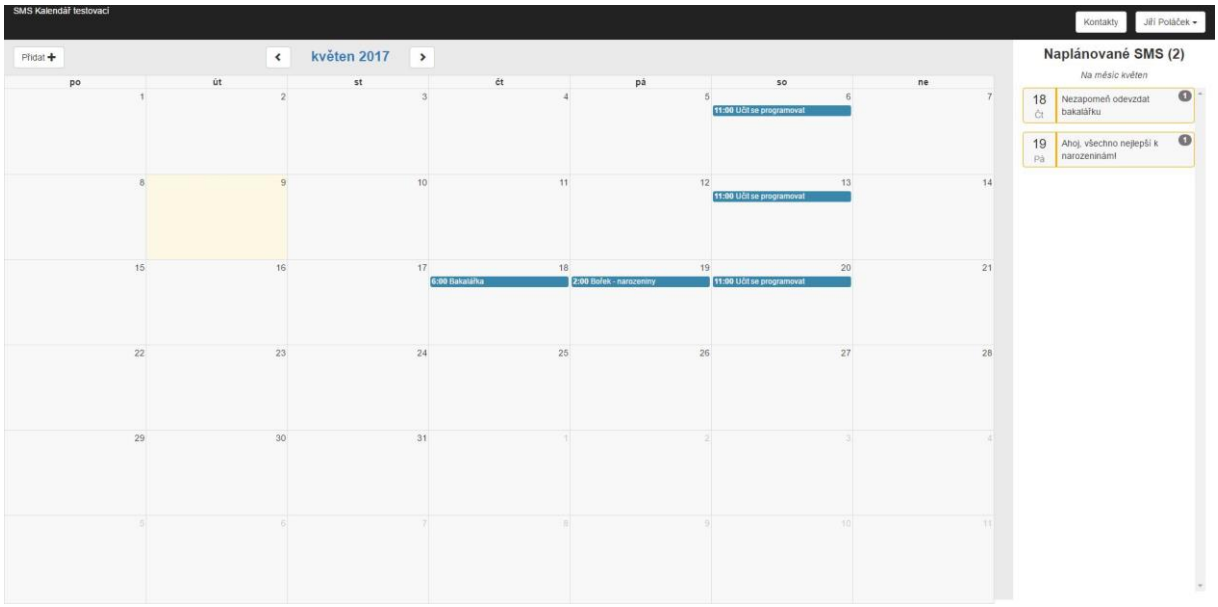
Je volně dostupná knihovna pro tvorbu webových aplikací. Obsahuje soubory kaskádových stylů a Javascriptu. Slouží k úpravě formulářů, tlačítek a dalších HTML elementů. Pro urychlení vývoje jsem využil komunitní widget *Datetime picker*, který slouží jako formulářový prvek pro zadávání data a času. Tento widget je dostupný s MIT licencí [30].

Pro náhled kalendáře a událostí jsem zvolil jQuery plugin *FullCalendar*, který je rovněž dostupný s licencí MIT [30].

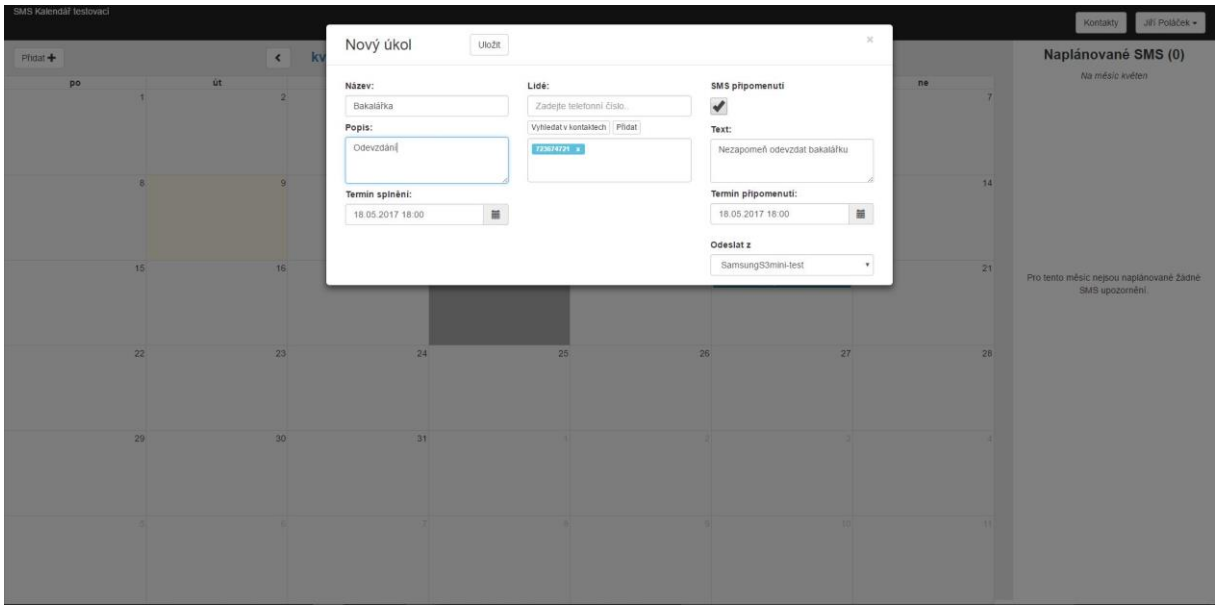
4.6.2 AngularJS

AngularJS je webový Javascriptový framework pro tvorbu single page aplikací. Pomocí speciálních formátovacích značek v HTML dokumentu, se určí, která operace má být provedena a s jakými daty. Pomocí AngularJS jsem zajistil generování dynamických seznamů a ovládání kalendáře.

Na obrázcích 17 a 18 jsou snímky finální podoby hlavní stránky aplikace, která je v rámci testování přístupná na adrese [https://www.calendar.jiripolacek.net].



Obrázek 17 - Finální podoba hlavní stránky



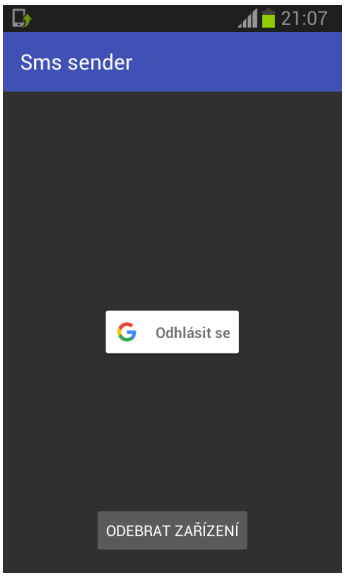
Obrázek 18 - Finální podoba dialogového okna nového úkolu

4.7 android-sms-sender

Mobilní aplikaci android-sms-sender pro operační systém Android verze 4.0, která na pozadí zajišťuje příjem notifikací ze služby OneSignal a následné stažení a odeslání SMS zpráv.

Aplikace obsahuje jednu aktivitu nazvanou *MainActivity*, která po spuštění aplikace zobrazí tlačítko pro přihlášení uživatele. Po přihlášení se toto tlačítko zamění za tlačítko pro odhlášení a aplikace ze serveru stáhne informaci, zda je toto zařízení přidané do systému SMS kalendáře. Pokud není, vykreslí v dolní části obrazovky tlačítko pro přidání zařízení. V opačném případě vykreslí tlačítko pro odebrání zařízení.

Po stisknutí tlačítka Přidat zařízení, nebo Odebrat zařízení, odešle aplikace na server požadavek a jako odpověď přijde potvrzení, zda je zařízení přidané, nebo odebrané.



Obrázek 19 - Uživatelské rozhraní aktivity

Kapitola 5

Testování

5.1 Jednotkové testování

Jednotkové testování je překlad z anglického Unit testing a označuje automatické testování dílčích částí (jednotek) implementace systému. V objektově orientovaném programování je jako dílčí část považována třída, nebo metoda. V ideálním případě by měl být každý testovaný případ nezávislý na ostatních. Při testování se snažíme testovanou část izolovat od ostatních částí programu. Za tím účelem se někdy vytvářejí pomocné objekty, které simulují předpokládaný kontext, ve kterém testovaná část pracuje (mock objekt) [30].

Pro testování jsem zvolil knihovnu *junit 4.11*. Jednotkové testy pokrývají služby sms-scheduler a sms-sender. Při testování tříd DAO, které komunikují s databází jsem před každým spuštěným testem vytvořil H2 databázi v embedded režimu, který je popsán v 2.9 H2 Database. Vytvořenou databázi jsem následně naplnil příslušnými testovacími daty.

Zde je ukázka testu DeviceDAO služby sms-scheduler, který kontroluje správnost výsledku dotazu na identifikační číslo SMS zařízení dle zadaného identifikačního čísla notifikace.

```
@org.junit.Test
public void getDeviceId() throws Exception {
    Connection c = DriverManager.getConnection(dbUrlUseThis);
    String result = DeviceDAO.getDeviceId(c,
"notificationIDsoRandom");

    assertEquals("1b282863-b52b-40e8-99ad-97282d4ce6ea",result);
}
```

Inicializace H2 databáze v embedded řežimu a naplnění testovacími daty načtenými z sql souboru:

```
private void junitDbDefault(String sqlInit) throws Exception{
            this.dbUrlUseThis = "jdbc:h2:mem:junit";
    Connection c = DriverManager.getConnection(dbUrlUseThis);
    c.setAutoCommit(true);
    c.createStatement().execute(sqlInit);
}
```

5.2 Testování rozhraní mikroslužeb

Pro testování rozhraní mikroslužeb jsem zvolil nástroj Postman. Vytvořené testy kontrolují, zda se rozhraní chová, jak se očekává a například nedovolí neplatný vstup. V případě služby sms-scheduler to může být minulé datum odeslání notifikace.

5.2.1 Postman

Nástroj Postman [33] slouží jako pomocný program při vytváření REST API. Ve bezplatné verzi umožňuje vytvářet HTTP požadavky pro účely testování. Požadavky mohou obsahovat testy, pro kontrolu odpovědi serveru. Tyto HTTP požadavky lze ukládat do kolekcí a tím vytvářet sady testů.

Následuje jednoduchý test aktualizace SMS notifikace, který kontroluje odpověď služby sms-scheduler. Nejprve parsuje tělo odpovědi jako JSON. Dále zkontroluje hodnoty odpovědi a zda je kód odpovědi 200, který značí, že požadavek byl úspěšně zpracován.

```
var jsonData = JSON.parse(responseBody);
tests["Status"] = jsonData.status === "success";
tests["Message"] = jsonData.message === "Notification updated";

tests["Content-Type is present"] = postman.getResponseHeader("Content-
Type");
tests["Status code is 200"] = responseCode.code === 200;
```

Testy jsou z programu Postman exportované do formátu JSON a přiložené spolu se zdrojovými kódy.

Kapitola 6

Závěr

Cílem práce bylo analyzovat, implementovat a otestovat webovou SMS notifikační službu se zaměřením na využití architektury mikroslužeb. V průběhu práce jsem se seznámil se samotnou architekturou mikroslužeb a jejími možnostmi využití v této práci. Dále jsem se seznámil s řešeními, které jsou dostupné na českém trhu, a to placené i bezplatné. Vytvořil jsem případy užití aplikace a navrhnul architekturu aplikace a seznámil se s technologiemi, které byli využity při implementaci.

Jako výhodu architektury mikroslužeb vidím vysokou znovupoužitelnost jednotlivých služeb. Například služby sms-scheduler-service a sms-sender-service budou moci sloužit k odesílání SMS zpráv v jiném systému. Jako hlavní nevýhodu architektury mikroslužeb vidím složitý návrh a složitou implementaci, která u malých projektů může být nežádoucí.

Během používání vytvořené služby jsem objevil problém, který se projevil, když mobilní zařízení bylo v době odeslání notifikace ze služby OneSignal bez přístupu k internetu. Po následném obnovení připojení nebyla notifikace doručena, nebo byla doručena s velkým zpožděním. Tento problém se mi již nepodařilo vyřešit před odevzdáním této práce. Možným východiskem je využití jiné služby pro odesílání push notifikací.

6.1 Další rozvoj

V systému jsem z části implementoval další funkčnost, která není součástí této práce. Jde o nový typ události – Schůzka, kde uživatel může nastavit oproti události typu Úkol další parametry. Funkcí systému je kromě měsíčního náhledu kalendáře náhled týdenní, pro kterou jsem již nestihl naimplementovat uživatelské rozhraní.

Další plánovanou funkcí, která by mohla zvýšit uživatelský komfort je fulltextové vyhledávání, které by vyhledávalo ze zadaného řetězce záznamy z kalendáře, kontaktů a SMS upozornění. Zajímavým prvkem se také muže jevit podpora více kalendářů a jejich sdílení.

Příloha A

Seznam použitých zkratek

ACID Atomic, Consistent, Isolatesd, Durable

AJAX Asynchronous JavaScript and XML

API Application Programming Interface

CRUD Create, Read, Update, Delete

DAO Data Access Object

DPH Daň z přidané hodnoty

DSL Domain Specific Language

FP Funkční požadavek

GSM Globální Systém pro Mobilní komunikaci

HTML HyperText Markup Language

HTTP Hypertext Transfer Protocol

ID Identifikátor

IP Internet Protocol

JDBC Java Database Connectivity

JPA Java Persistence API

JSON JavaScript Object Notation

JVM Java Virtual Machine

KP Kvalitativní požadavek

MB Megabyte

MMS Multimedia Messaging Service

ODBC Open Database Connectivity

OS Operační Systém

RAM Random Access Memory

REST Representional State Transfer

RPC Remote Procedure Call

SMS Short Message Service

SPA Single Page Application

SQL Structured Query Language

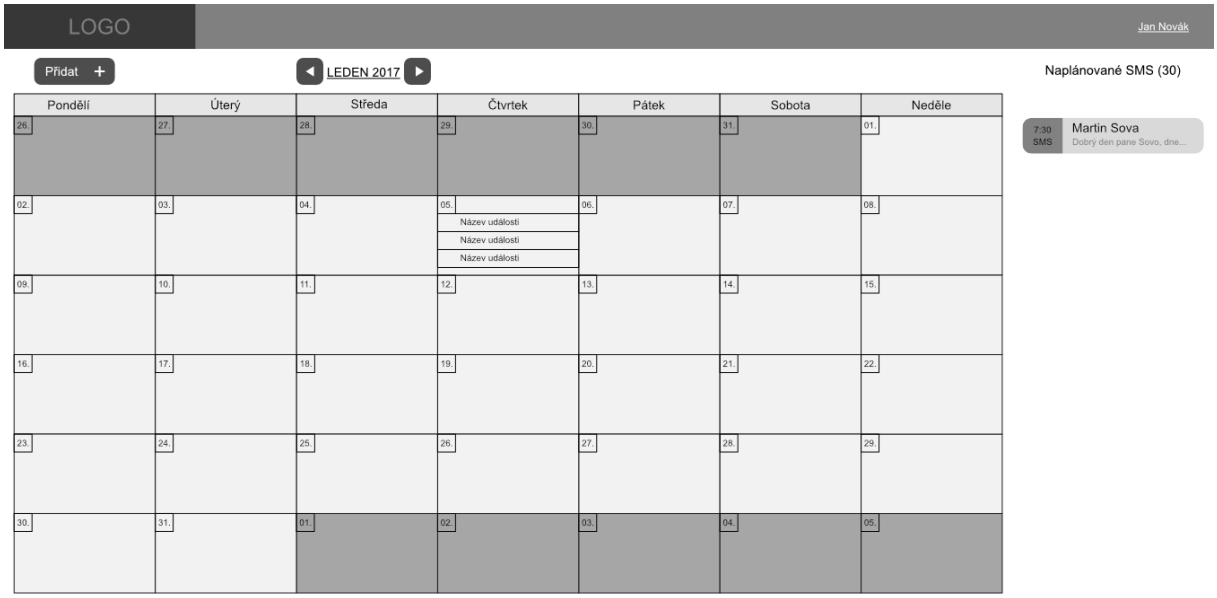
TCP Transmission Control Protocol

URL Uniform Resource Locator

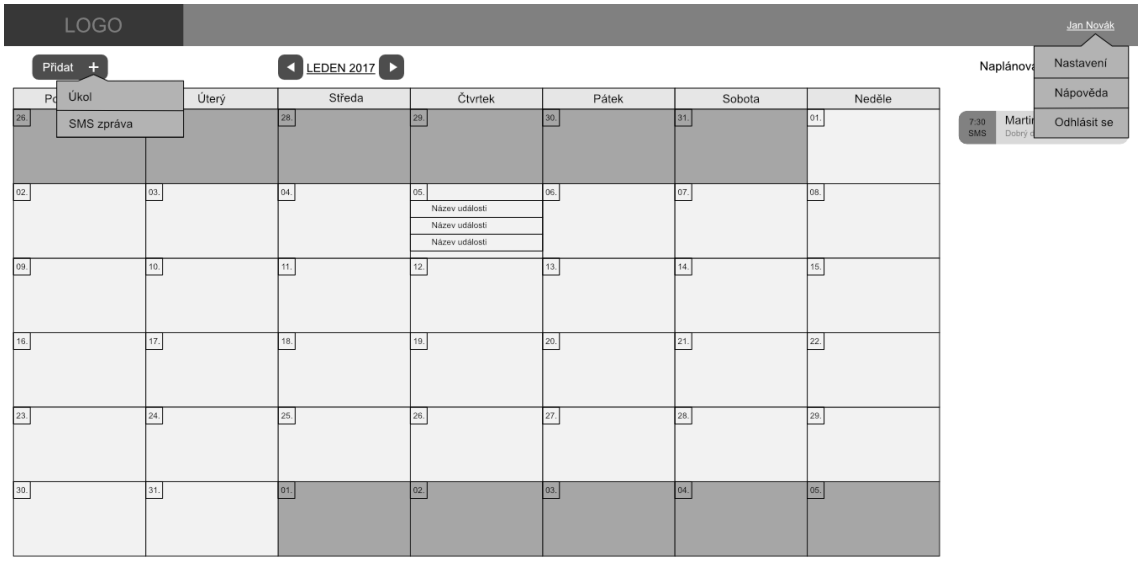
XML Extensible Markup Language

Příloha B

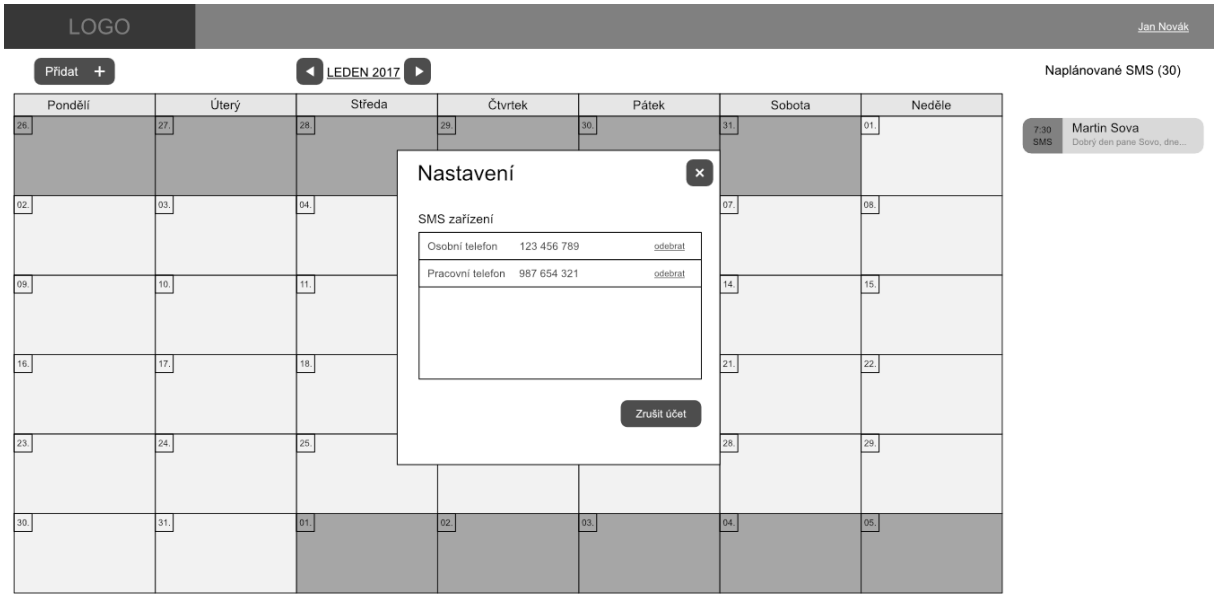
Obrazové přílohy



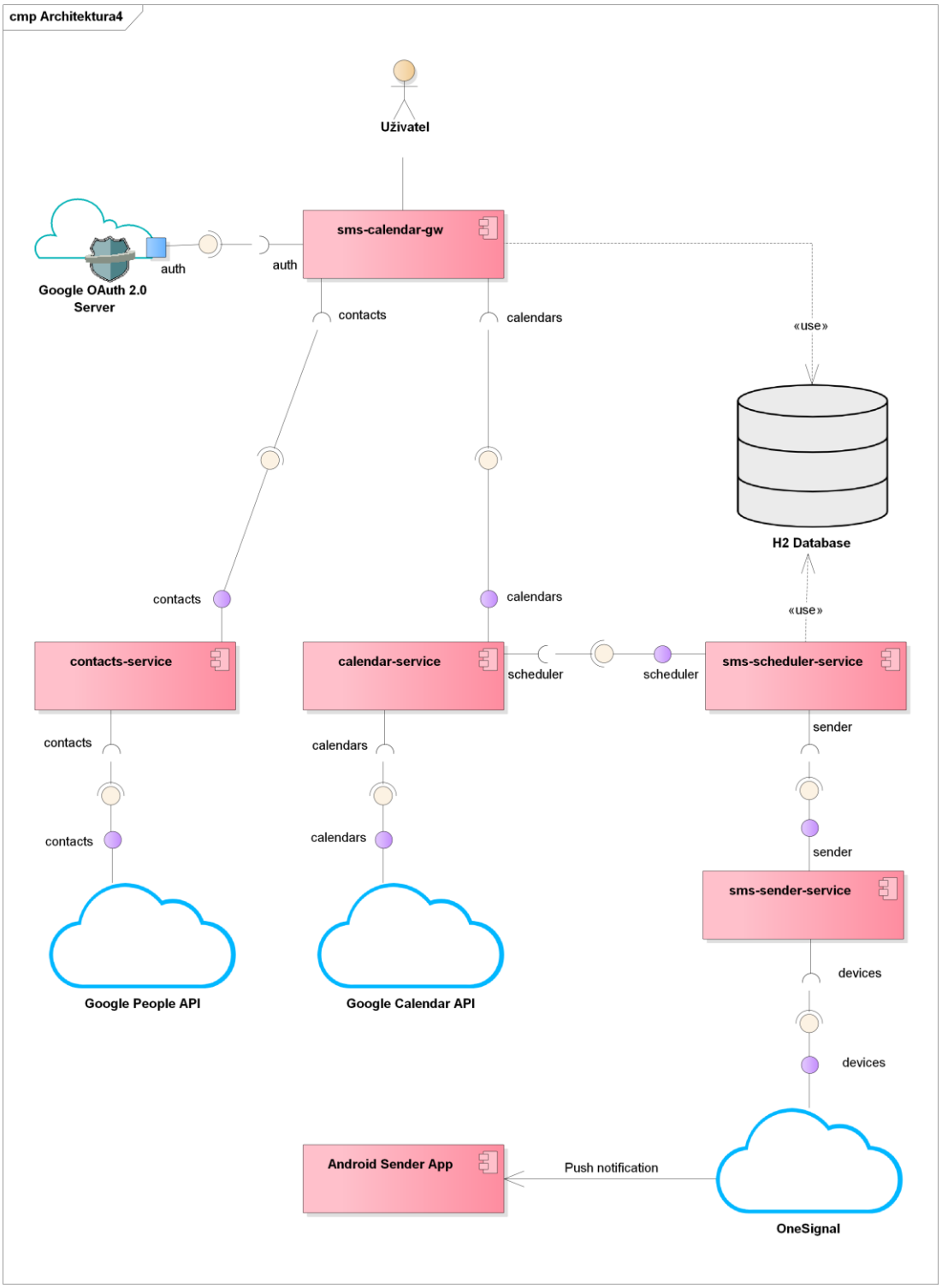
Obrázek B 1 - Hlavní stránka kalendáře (wireframe)



Obrázek B 2 - Kontextové nabídky (wireframe)



Obrázek B 3- Nastavení (wireframe)



Obrázek B 4 – Architektura systému SMS kalendáře

Příloha C

Zdrojové kódy a aplikace

Použité zdroje

[1] ARLOW, Jim a Ila NEUSTADT. *UML 2 a* unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky. 2., aktualiz. a dopl. vyd. Brno: Computer Press, 2007. ISBN 9788025115039.

[2] Kalendář Google: Online kalendář a *plánování pro firmy.* G Suite [online]. Google, 2016 [cit. 2017-01-16]. Dostupné z: https://gsuite.google.cz/products/calendar/

[3] SmsDiary: Rezervační systém s SMS *připomenutím schůzky pro lékaře*, beauty salony, kadeřnictví [online]. Infinity Energy, 2016 [cit. 2017-01-16]. Dostupné z: http://www.smsdiary.cz/

[4] Reservio: Online rezervační systém zdarma *[online]. Brno:* Reservio, 2016 [cit. 2017-01-16]. Dostupné z: https://www.reservio.com/cs/

[5] SMS Scheduler: Sent Later. *Google Play [online]*. Google, 2016 [cit. 2017-01-16]. Dostupné z: https://play.google.com/store/apps/details?id=com.ruh.smsscheduler

[6] Do It Later - Best *Scheduler. Google Play [*online]. Google, 2016 [cit. 2017-01-16]. Dostupné z: https://play.google.com/store/apps/details?id=com.hnib.smslater

[7] BALALAIE, Armin, Abbas HEYDARNOORI *a Pooyan JAMSHIDI. Microservices* Architecture Enables DevOps: Migration to a Cloud-Native Architecture. IEEE Software [online]. 2016, 33(3), 42-52 [cit. 2016-12-21]. DOI: 10.1109/MS.2016.64. ISSN 07407459. Dostupné z: http://ieeexplore.ieee.org/document/7436659/

[8] Microservices. In: Martin Fowler *[online]. 2014 [*cit. 2017-01-15]. Dostupné z: https://martinfowler.com/articles/microservices.html

[9] Representational state transfer. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2016 [cit. 2017-01-15]. Dostupné z: https://en.wikipedia.org/wiki/Representational_state_transfer

[10] RICHARDSON, Chris. Pattern: Monolithic *Architecture. In: Microservice* architecture [online]. Chris Richardson, 2014 [cit. 2017-01-15]. Dostupné z: http://microservices.io/patterns/monolithic.html

[11] RICHARDSON, Chris. Pattern: Microservice *Architecture. In: Microservice* architecture [online]. Chris Richardson, 2014 [cit. 2017-01-15]. Dostupné z: http://microservices.io/patterns/microservices.html

[12] RICHARDS, Mark. Microservices vs. *Service-Oriented Architecture [online]*. 1005 Gravenstein Highway North, Sebastopol, CA 95472: O'Reilly Media, 2015 [cit. 2017-01-20]. Dostupné z: http://www.oreilly.com/programming/free/microservices-vs-service-oriented-architecture.csp

[13] RICHARDSON, Chris. API Gateway. *In: Microservice architecture* [online]. 2014 [cit. 2017-01-16]. Dostupné z: http://microservices.io/patterns/apigateway.html

[14] RICHARDSON, Chris a Floyd SMITH. *Microservices: From Design* to Deployment [online]. NGINX, 2016 [cit. 2017-01-20]. Dostupné z: https://www.nginx.com/resources/library/designing-deploying-microservices/?utm_source=microservices-from-design-to-deployment-ebook-nginx&utm_medium=blog&utm_campaign=Microservices

[15] RICHARDSON, Chris. Pattern: Shared *database. In: Microservice* architecture [online]. 2014 [cit. 2017-01-16]. Dostupné z: http://microservices.io/patterns/data/shared-database.html

[16] RICHARDSON, Chris. Pattern: Database *per service. In:* Microservice architecture [online]. 2014 [cit. 2017-01-16]. Dostupné z: http://microservices.io/patterns/data/database-per-service.html

[17] RICHARDSON, Chris. Event-Driven Data *Management for Microservices. In*: Nginx [online]. 2015 [cit. 2017-01-15]. Dostupné z: https://www.nginx.com/blog/event-driven-data-management-microservices/

[18] Google APIs. In: Wikipedia: *the free encyclopedia [*online]. San Francisco (CA): Wikimedia Foundation, 2001-2016 [cit. 2017-01-18]. Dostupné z: https://en.wikipedia.org/wiki/Google_APIs

[19] Using OAuth 2.0 to *Access Google APIs. Google* Identity Platform [online]. Google, 2016 [cit. 2017-01-20]. Dostupné z: https://developers.google.com/identity/protocols/OAuth2

[20] Google Sign-In for Websites [online*]. Google, b.*r. [cit. 2017-01-16]. Dostupné z: https://developers.google.com/identity/sign-in/web/

[21] Authenticate with a backend server. *Google Sign-In for Websites* [online]. Google, b.r. [cit. 2017-01-16]. Dostupné z: https://developers.google.com/identity/sign-in/web/backend-auth

[22] Google Calendar API [online]. *Google, 2016 [cit*. 2017-01-18]. Dostupné z: https://developers.google.com/google-apps/calendar/

[23] API Reference. Google Calendar API *[online]. Google,* 2015 [cit. 2017-01-18]. Dostupné z: https://developers.google.com/google-apps/calendar/v3/reference/

[24] Google Calendar API Usage Limits. *Google Calendar API [*online]. Google, 2016 [cit. 2017-01-18]. Dostupné z: https://developers.google.com/google-apps/calendar/pricing

[25] People API. Developers Google [*online]. Google Inc.*, 2017 [cit. 2017-05-09]. Dostupné z: https://developers.google.com/people/

[26] SmsManager. Android developers [online]. *Google, 2009 [*cit. 2017-01-18]. Dostupné z: https://developer.android.com/reference/android/telephony/SmsManager.html

[27] OneSignal [online]. Mountain View, *CA: Lilomi,* Inc., 2016 [cit. 2017-01-18]. Dostupné z: https://onesignal.com/

[28] Quartz Job Scheduler [online]. *Terracotta, Inc.*, 2016 [cit. 2017-01-19]. Dostupné z: www.quartz-scheduler.org/

[29] Features. H2 Database Engine [*online]*. *H2 Group,* 2014 [cit. 2017-01-19]. Dostupné z: http://www.h2database.com/html/features.html

[30] Bootstrap 3 Datepicker v4 [online*]*. *GitHub, 2017* [cit. 2017-05-13]. Dostupné z: http://eonasdan.github.io/bootstrap-datetimepicker/

[31] FullCalendar [online]. FullCalendar LLC, *2017 [cit.* 2017-05-13]. Dostupné z: https://fullcalendar.io

[32] Unit testing. In: Wikipedia: *the free encyclopedia [*online]. San Francisco (CA): Wikimedia Foundation, 2001-2017 [cit. 2017-05-13]. Dostupné z: https://cs.wikipedia.org/wiki/Unit_testing

[33] Postman [online]. Postdot Technologies, *2017 [cit.* 2017-05-17]. Dostupné z: https://www.getpostman.com/