UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

BAKALÁŘSKÁ PRÁCE

2017 Tomáš Špidlen

Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Rezervace schůzek pro Kalendář Google

Tomáš Špidlen

Bakalářská práce

2017

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2016/2017**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Tomáš Špidlen**
Osobní číslo:	**I14182**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Rezervace schůzek pro Kalendář Google**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Cílem práce je vytvořit přehled webových aplikací pro rezervaci schůzek, jejich vlastností, ceně a možnostech exportu dat (do kalendáře ve formátu iCAL či tabulky ve formátu CSV).

Student následně navrhne a implementuje webovou aplikaci (skript) na platformě Google Apps Script (programovacím jazykem je JavaScript), která umožní hlavnímu uživateli skriptu integraci s Kalendářem Google a možnost nabídnutí termínů schůzek dalším uživatelům, kteří si následně budou moci rezervovat čas schůzky. Aplikace má víceméně nahradit zrušenou vlastnost Kalendáře Google – bloky schůzek (appointment slots).

Rozsah grafických prací:

Rozsah pracovní zprávy: **cca 40–50 stran**

Forma zpracování bakalářské práce: **tištěná**

Seznam odborné literatury: **viz příloha**

Vedoucí bakalářské práce: **Mgr. Tomáš Hudec**
Katedra informačních technologií

Datum zadání bakalářské práce: **31. října 2016**

Termín odevzdání bakalářské práce: **12. května 2017**

Univerzita Pardubice Česká republika

L.S.

Ing. Zdeněk Němec, Ph.D.
děkan

Mgr. Josef Horálek, Ph.D.
vedoucí katedry

V Pardubicích dne 31. března 2017

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 12. 5. 2017

podpis autora
Tomáš Špidlen

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu mé bakalářské práce panu Mgr. Tomášovi Hudcovi za cenné rady, tipy a konzultace po celou dobu zpracování.

ANOTACE

Tato bakalářská práce vytváří přehled webových aplikací, jejichž účelem je vytvoření schůzek a jejich následné zveřejnění pro jiné uživatele s možností rezervace. Ke každé aplikaci jsou uvedeny informace o základních vlastnostech, přednostech a nedostatcích, ceně a o možnosti exportu dat. Hlavní náplní práce je vytvoření webové aplikace založené na tomto principu jako rozšíření pro kalendář Google.

KLÍČOVÁ SLOVA

Kalendář, schůzka, rezervace, Google, webová aplikace, poskytovatel, klient, událost, Google Apps Script, HTML, CSS, Javascript, aplikační rozhraní kalendáře, funkce, datum a čas

TITLE

Appointment reservation for Google Calendar

ANNOTATION

This bachelor thesis creates an overview of web applications designed for creation of appointment slots and subsequent publication of these slots with the possibility of booking. For each application, information on basic features, strengths and weaknesses, pricing and data export options is provided. The main purpose is to create a web based application based on this principle as a Google Calendar extension.

KEYWORDS

Calendar, appointment, reservation, Google, web application, provider, client, event, Google Apps Script, HTML, CSS, Javascript, calendar API, function, date and time

OBSAH

0 Úvod 13

1 Webové aplikace 14

1.1 Bloky schůzek v Kalendáři Google 14

1.2 Doodle 14

1.3 Setmore 15

2 Návrh aplikace 17

2.1 Hlavní cíl 17

2.2 Poskytovatelský přístup 17

2.2.1 Vytváření termínů schůzek 17

2.3 Klientský přístup 18

2.3.1 Možnost vylepšení 18

2.4 Rozlišení přístupů 18

2.4.1 Omezení autentizace 19

3 Použité technologie 20

3.1 HTML 20

3.2 CSS 21

3.3 JavaScript 22

3.3.1 JavaScriptové knihovny 23

4 Google Apps Script 24

4.1 HTML Service 24

4.1.1 Třída HtmlOutput 24

4.1.2 Skriptlet 25

4.1.3 Třída HtmlTemplate 26

4.1.4 Třída google.script.run 27

4.2 Calendar Service 28

4.2.1 Třída Calendar 28

4.2.2 Třída CalendarEvent 30
4.3 Properties Service 32
4.3.1 Vlastnosti skriptu 32
4.3.2 Vlastnosti uživatele 32
4.3.3 Vlastnosti dokumentu 32
4.4 Session 32
4.5 ScriptApp 33
5 Implementace požadavků 34
5.1 Spuštění aplikace 34
5.1.1 Rozlišení klienta a poskytovatele 34
5.2 Část pro poskytovatele 35
5.2.1 Nastavení kalendáře 35
5.2.2 Vytváření schůzek 37
5.2.3 Přehled schůzek 37
5.2.4 Úprava času schůzky 37
5.2.5 Odstranění schůzky 37
5.3 Část pro klienta 38
5.3.1 Výpis volných termínů 38
5.3.2 Zápis na termíny 38
5.4 Volání funkcí na serveru 38
5.5 Zpřístupnění událostí 39
5.6 Grafické zobrazení kalendáře v aplikaci 39
Závěr 40
Použitá literatura 41
Přílohy 43

SEZNAM ILUSTRACÍ A TABULEK

Obrázek 1 – grafický výstup v prohlížeči 21
Tabulka 1 – Porovnání vybraných vlastností webových aplikací.....16

SEZNAM UKÁZEK KÓDU

Ukázka 1 – Základní struktura HTML dokumentu 20
Ukázka 2 – Příklad elementů a atributů 21
Ukázka 3 – Příklad CSS pravidla 21
Ukázka 4 – Příklad JS funkce 22
Ukázka 5 – Připojení odděleného souboru s JavaScriptem 22
Ukázka 6 – Funkce z ukázky 4 požívající jQuery 23
Ukázka 7 – Použití createHtmlOutput 25
Ukázka 8 – Použití createHtmlOutputFromFile 25
Ukázka 9 – Standard scriptlet 25
Ukázka 10 – Printing scriptlet 26
Ukázka 11 – Použití createTemplateFromFile a evaluate 26
Ukázka 12 – Asynchronní volání funkce na serveru 27
Ukázka 13 – Použití withSuccessHandler 27
Ukázka 14 – Použití getCalendarById a příklad identifikátoru kalendáře 28
Ukázka 15 – Vytvoření události 29
Ukázka 16 – Vytvoření opakované události 29
Ukázka 17 – Zpřístupnění události 30
Ukázka 18 – Smazaní události 31
Ukázka 19 – Použití vlastností skriptu 32
Ukázka 20 – Získání emailové adresy uživatele, pod jehož autoritou je skript spuštěn 32
Ukázka 21 – Získání adresy URL skriptu 33
Ukázka 22 – implementace funkce doGet v ARS 34
Ukázka 23 – ukládání identifikátoru kalendáře jako vlastnost skriptu 36
Ukázka 24 – Načítání identifikátoru kalendáře z vlastnosti skriptu 36
Ukázka 25 – smazání vlastnosti skriptu 36
Ukázka 26 – Serverová funkce removeEvent 37
Ukázka 27 – Použití google.script.run v ARS 38
Ukázka 28 – Zobrazení kalendáře v ARS 39

SEZNAM ZKRATEK A ZNAČEK

KG	Kalendář Google
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
ARS	Aplikace Rezervace Schůzek
JS	JavaScript
GAS	Google Apps Script
URL	Uniform Resource Locator

Typografické konvence

V textu se budou vyskytovat slova či slovní spojení která jsou zvýrazněna kurzívou. Jedná se o odborné názvy v teoretické část nebo názvy proměnných, metod, tříd, souborů či přípony souborů v části, kde je popisována aplikace.

0 ÚVOD

Cílem v teoretické části práce je vytvoření krátkého přehledu aplikací dostupných na internetu, které určitým způsobem umožňují organizaci času. Základním požadavkem na takovou aplikaci je možnost vytvoření událostí, které bude následně možné zveřejnit pro určitou skupinu uživatelů. Tito uživatelé pak mají možnost si tyto události rezervovat. Tento přehled obsahuje bližší informace o těchto aplikacích. Zejména informace o tom, jak daná aplikace funguje, jaké má výhody a nevýhody, jaká je cenová dostupnost a informaci o možnostech exportu dat.

Cílem praktické části je návrh a implementace webové aplikace na platformě Google Apps Script (GAS), která bude splňovat výše uvedené požadavky. Základním předpokladem pro používání této aplikace je uživatelský účet u společnosti Google.

Uživatelský přístup k aplikaci se dělí na dva základní druhy. První přístup je tzv. poskytovatel, který bude mít možnost tvorby událostí a jejich zveřejnění. Druhým přístupem je tzv. klient, který si tyto události může prostřednictvím aplikace rezervovat.

Poskytovateli aplikace umožní integraci s Kalendářem Google (KG). Poté bude mít možnost manipulace s tímto vlastním kalendářem ve smyslu tvorby nových událostí, jejich úpravy a případného smazání. Dále bude mít přehled o vytvořených událostech a o tom, kdo má danou událost zarezervovanou.

Klient bude mít k dispozici seznam volných termínů, které bude mít možnost si rezervovat.

V práci je popsán návrh aplikace, technologie použité k vytvoření aplikace, a nakonec samotná implementace s ukázkami kódu.

1 WEBOVÉ APLIKACE

V této kapitole jsou uvedeny některé webové aplikace podobného charakteru, jaký má za cíl praktická část práce.

1.1 Bloky schůzek v Kalendáři Google

Jedná se o funkci Kalendáře Google, která je zdarma, ale je dostupná pouze prostřednictvím firemních nebo školních účtů Google.

Tuto funkci lze použít při vytváření nové události v KG kliknutím na záložku Úseky schůzek. Uživatel má možnost vytvořit novou událost v časovém rozmezí, které si určil a tuto událost rozdělit na úseky o časové délce, kterou si zvolí. Poté může svůj kalendář s těmito schůzkami poskytnout prostřednictvím adresy URL dalším uživatelům, kteří mají možnost se na tyto termíny zapsat.

Jelikož obě strany musí použít účet Google, je vyřešena autentizace i informace o tom, kdo je daný uživatel, který si rezervoval schůzku.

Toto rozšíření dále podporuje oznámení emailem pro obě strany.

Kalendář Google nabízí export dat v kalendáři do souboru ve formátu ICS (iCalendar).

Funkce byla inspirací pro praktickou část práce, ve které bylo cílem implementovat rozšíření pro KG na obdobném principu, které bude možné použít i prostřednictvím běžného účtu Google. Zdroj [1]

1.2 Doodle

Tato webová aplikace dostupná na adrese www.doodle.com slouží k naplánování události pro skupinu lidí. V aplikaci je možné vytvořit si účet, což umožňuje využít některých funkcí navíc. Zároveň se dá použít i jednorázově bez nutnosti registrace.

Aplikace funguje na principu vybrání termínu hodící se většině. Jeden uživatel vytvoří událost, ke které nabídne několik možných termínů. Tuto událost poté prostřednictvím vygenerovaného odkazu zpřístupní dalším uživatelům. Tito uživatelé si z nabízených termínů vyberou ty, které jim nejvíce vyhovují, a pro ty hlasují.

Tento princip lze modifikovat pro účely rezervace schůzek. Uživatel vytvářející schůzky může ostatní instruovat, aby si každý vybral pouze jeden termín.

Aplikace Doodle umožňuje připojení jiných kalendářových aplikací. Toho lze využít při vybírání z dostupných termínů. Uživatel má totiž při rezervaci náhled také do svého osobního kalendáře.

Doodle je zdarma dostupná aplikace. Nicméně nabízí také prémiové funkce, za které si uživatel musí zaplatit

V základní verzi pro jednotlivce má uživatel možnost:

- vytvářet a plánovat události,
- připojit vlastní kalendář.

Verze Private pro jednotlivce stojí 29 euro na rok a má navíc následující výhody:

- zasílání automatických upomínek,
- přehled o tom, kdo nehlasoval,
- možnost požádat případné účastníky o dodatečné informace,
- šifrování SSL a
- absence reklam.

Verze Business určená pro týmy stojí 49 euro na rok a k výhodám verze Private přidává navíc následující výčet:

- Uživatelé si můžou nadefinovat vlastní vzhled rezervací.
- Plánování navíc usnadňuje možnost vytvoření vlastní subdomény.
- Prémiový uživatel má možnost do jisté míry spravovat účty dalších uživatelů.

Aplikace Doodle neumožňuje export dat. Zdroj [2]

1.3 Setmore

Dalším zástupcem webových aplikací fungujících na podobném principu je aplikace Setmore. Jedná se o velmi kvalitní aplikaci s rozsáhlými možnostmi spravování schůzek. Pro její použití je nutné se registrovat. Jakmile má uživatel vytvořený účet, má přístup k velice účinnému a přehlednému nástroji pro organizaci času, který může najít velmi dobré uplatnění například u provozovatelů kadeřnických či masážních salónů aj.

Prostřednictvím aplikace je možné vytvářet termíny pro nejrůznější účely. Tyto termíny je poté možné pomocí adresy URL poskytnout případným klientům. Provozovatel má také přehled o těchto klientech a má k dispozici správu případných podřízených, které může přiřadit k danému termínu. Dále si také může vést seznam služeb, které nabízí, což ulehčí tvorbu termínů.

Pro samotnou tvorbu termínů slouží jednoduché interaktivní rozhraní kalendáře, ve kterém si uživatel zvolí časové rozmezí pro termín. V podrobnostech o konkrétním termínu může uvést, o jakou službu se jedná, který podřízený ji poskytne, informaci o místě, ceně služby aj. Případným klientům následně poskytne adresu URL, na které si mohou vybrat z dostupných termínů. Drobným nedostatkem při tvorbě schůzek je fakt, že není možné vytvořit více stejných termínů najednou.

Klientské rozhraní nabízí přehled nabízených služeb a volné termíny, které je možné si rezervovat. Klient musí být rovněž registrovaný a přihlášený v aplikaci.

Export dat přímo z aplikace Setmore není podporován. Nicméně je možná integrace s jinou kalendářovou aplikací jako: Kalendář Google, Kalendář Outlook nebo Kalendář Office 365. Do těchto kalendářových aplikací je možné exportovat termíny vytvořené v kalendáři Setmore. Například z Kalendáře Google je možné tyto termíny exportovat ve formátu ICS.

Aplikace je dostupná zdarma. Nicméně je dostupná také prémiová verze, která nabízí určité výhody. Nicméně tyto výhody se týkají zejména pohodlnějšího používání a není tedy nutné je zde zmiňovat.

Vysoká míra komplexnosti a propracovanosti aplikace může být zezačátku překážkou. Nicméně jakmile se uživatel seznámí se všemi vlastnostmi aplikace a osvojí si jejich používání, má k dispozici velice účinný nástroj pro organizaci času. Zdroj [3]

Tabulka 1 – Porovnání vybraných vlastností webových aplikací[1]

Název	Bloky schůzek v KG	Doodle	Setmore
Cenová dostupnost	Zdarma[2]	Zdarma[3]	Zdarma[3]
Export do ICS	Ano	Ne	Částečně[4]
Možnost integrace	Součást KG	KG, Outlook	KG, Outlook
Rychlost osvojení	Velmi rychlá	Rychlá	Střední
Vhodnost použití[5]	Velmi dobrá	Střední	Dobrá

[1] Uvedené údaje jsou založené na autorově vlastní zkušenosti s použitím aplikace
[2] Funkce je dostupná pouze prostřednictvím školních či pracovních účtů Google
[3] Možnost prémiových funkcí
[4] Lze exportovat události do KG a z něj do ICS
[5] Ve smyslu možnosti použití pro účely rezervace schůzek

2 NÁVRH APLIKACE

2.1 Hlavní cíl

Hlavním cílem praktické části práce bylo vytvoření webové aplikace na platformě Google Apps Script. Tato aplikace by měla umožnit hlavnímu uživateli (dále poskytovatel) integraci s Kalendářem Google, díky čemuž bude možné vytvořit termíny schůzek jako události tohoto kalendáře a následně tyto termíny nabídnout dalším uživatelům (dále klienti), kteří si následně budou moci tyto termíny rezervovat a případně vytvořit odpovídající událost ve svém KG.

Jako první bylo tedy nutné definovat dva různé přístupy.

2.2 Poskytovatelský přístup

Jak bylo zmíněno výše, poskytovatel musí mít možnost integrace skriptu s jeho KG, ve kterém bude mít možnost vytvářet termíny schůzek. Pro tento účel bylo potřeba vytvořit jednoduché uživatelské rozhraní, které bude nabízet tyto základní možnosti:

- volba kalendáře, který bude použit pro účely aplikace a případná změna,
- zobrazení tohoto kalendáře,
- vytváření termínů schůzek,
- výpis těchto termínů s informacemi o čase konání a obsazenosti,
- odstranění a editace termínů.

2.2.1 Vytváření termínů schůzek

Hlavní myšlenkou tvorby termínů bylo rozdělení jednoho většího časového úseku na kratší části, jejichž délka bude definovatelná poskytovatelem. Tento způsob je v aplikaci realizován. Poskytovatel tedy nadefinuje datum a časové rozmezí, ve kterém mají být termíny vytvořeny a následně určí délku těchto termínů v minutách.

Kromě tohoto základního způsobu vytváření termínů schůzek aplikace navíc umožňuje:

- možnost každodenního, týdenního a měsíčního opakování až do zvoleného data,
- volitelná upomínka hodinu před začátkem schůzky,
- volitelná upomínka jeden den předem.

2.3 Klientský přístup

Uživatelské rozhraní pro klientský přístup k aplikaci musí umožňovat 2 základní funkce:

1. zobrazení dostupných termínů schůzek od poskytovatele a
2. možnost se na tyto termíny registrovat.

V aplikaci Rezervace schůzek (ARS) je klientovi dostupný výpis volných termínů schůzek a na tyto termíny se může také po vyplnění jména a příjmení zaregistrovat.

2.3.1 Možnost vylepšení

Klient může aplikaci využít pouze k registraci na určitou schůzku, ale už tuto rezervaci nemá možnost zrušit. Pro tento účel je nutné kontaktovat poskytovatele emailem. Poskytovatel má ke schůzkám plný přístup a má možnost také zrušení rezervace. Toto omezení je způsobeno nedostatečnou autentizací uživatele, který chce aplikaci spustit jako klient (viz 2.4.1 Omezení autentizace). Tento nedostatek se při implementaci skriptu nepodařilo odstranit.

Za zvážení by rovněž stálo omezení na počet schůzek, které se klient může rezervovat. To je ale záležitost poskytovatele, takže to nebylo bráno jako požadavek.

2.4 Rozlišení přístupů

Rozlišit, jestli uživatel, který spouští aplikaci, je poskytovatel nebo klient, bylo pro způsob, jakým bude skript implementován, klíčové. Zásadním krokem bylo rozhodnutí, jakým způsobem má být aplikace zprovozněna a pod čí autoritou má být spuštěna. Již od začátku byla práce směrována k tomu, aby byl výsledný skript publikován jako webová aplikace. Ohledně autority, pod kterou má být skript spuštěn, jsou na výběr dvě možnosti:

1. Spustit aplikaci jako autor skriptu nebo
2. Spustit aplikaci jako uživatel, který k ní má přístup.

Pro účely aplikace byl zvolen první způsob, tedy spustit aplikaci jako autor skriptu. Tento způsob vyžaduje, aby každý uživatel, který bude chtít aplikaci používat jako poskytovatel, aplikaci spustil pod vlastní autoritou. Tohoto faktu se dá dobře využít při rozhodování, zda uživateli, který k aplikaci přistupuje, má být zobrazena stránka klienta nebo poskytovatele. Tuto volbu podporuje i fakt, že poskytovat schůzky bude, vždy jen jeden uživatel. Proto není potřeba mezi sebou rozlišovat více poskytovatelů.

V případě druhé možnosti by bylo také zapotřebí nějakým způsobem umožnit klientovi modifikovat kalendář poskytovatele. Tento problém řeší fakt, že aplikace běží pod autoritou poskytovatele, kterému kalendář patří.

2.4.1 Omezení autentizace

Autentizace je v aplikaci řešena na základě emailové adresy (účet Google). Pokud je tato adresa shodná s adresou uživatele, pod jehož autoritou je skript spuštěn (vlastník skriptu), je tento uživatel brán jako poskytovatel. Nicméně pokud skript spustí uživatel, který není vlastníkem skriptu a zároveň nepatří do stejné G Suite domény (např. student.upce.cz) není jeho emailová přístupná. Proto je tento způsob možné použít pouze jako rozhodnutí, zda se jedná o poskytovatele nebo klienta, nikoli k rozlišení jednotlivých klientů.

Zdroj [4]

3 POUŽITÉ TECHNOLOGIE

Aplikace pro rezervaci schůzek spadá do kategorie webových aplikaci. K jejímu vytvoření byly tedy použity technologie, které byly vyvinuty primárně za účelem tvorby webových aplikací. Grafická část, která je zobrazena uživateli, je napsána v jazyce HTML. Její vzhled je dále upraven pomocí Kaskádových stylů (CSS). Dynamika aplikace a zpracování událostí je zajištěno prostřednictvím JavaScriptu (JS). JavaScript dále zajišťuje také propojení s aplikačním rozhraním KG prostřednictvím technologie Google Apps Script, které bude věnována samostatná kapitola.

3.1 HTML

HyperText Markup Language je značkovací jazyk, používaný pro tvorbu webových stránek, které jsou propojeny hypertextovými odkazy.

Jazyk HTML se skládá z množiny značek a jejich atributů, které se řídí určitými syntaktickými pravidly. Mezi tyto značky se uzavírají části textu dokumentu. Tímto způsobem se určuje význam (sémantika) obsaženého textu. Pro správnou prezentaci HTML dokumentu je nutný tzv. interpret, který přeloží dokument napsaný v jazyce HTML a zobrazí jeho výstup. Tuto funkci plní webové prohlížeče.

Zdroj [5]

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
  <meta http-equiv="content-type" content="text/html; charset=utf-8">
  <title></title>
  </head>
  <body>
  </body>
</html>
```

Ukázka 1 – Základní struktura HTML dokumentu

```
<p id="label">Zadej text</p>
<input type="text"/>
<input type="button" value="Klikni"/>
```

Ukázka 2 – Příklad elementů a atributů

Obrázek 1 – grafický výstup v prohlížeči

3.2 CSS

Kaskádové styly (Cascading Style Sheets) jsou jazyk pro popis způsobu zobrazení elementů na stránkách napsaných značkovacím jazykem (např. HTML). Tento způsob zobrazení je definován množinou pravidel. Každé pravidlo se skládá ze selektoru a bloku deklarací. Deklarace jsou od sebe odděleny středníkem. Každá deklarace obsahuje identifikátor vlastnosti a hodnotu vlastnosti oddělené dvojtečkou.

Způsoby aplikace kaskádových stylů v HTML dokumentu

Existují tři základní způsoby, jak aplikovat kaskádové styly do HTML dokumentu:

1) Zápis stylů do elementu style,
2) připojení externího souboru pomocí http hlavičky link,
3) přímý zápis pomocí atributu style.

V ARS jsou styly aplikovány pomocí připojení externího souboru. Tento způsob je zároveň používán nejčastěji.

```
p {
  background-color: blue;
}
```

Ukázka 3 – Příklad CSS pravidla

Kde *p* je selektor, *background-color* je identifikátor vlastnosti a *blue* je hodnota vlastnosti.

Zdroj [6]

3.3 JavaScript

JavaScript (JS) je interpretovaný, multiplatformní programovací jazyk podporující objektově orientovaný přístup, který je prováděn na straně klienta. Univerzální jádro jazyka je obsaženo ve webových prohlížečích. Při provádění JS tedy není třeba kontaktovat server, veškeré provedení skriptu zajišťuje prohlížeč. Hlavním účelem JS je dynamika webových stránek a aplikací. Umožňuje vložení proveditelného obsahu do jinak statických webových stránek. Stránky tak mohou obsahovat různé programy a funkce, které mohou komunikovat s uživatelem, zpracovávat události nebo manipulovat s obsahem HTML.

Zdroj [7]

```
function exampleFunction() {
   var date = new Date();
   document.getElementById("test").innerHTML = date;
}
```

Ukázka 4 – Příklad JS funkce

Podobně jako kaskádové styly lze JavaScript do HTML dokumentu vložit dvěma způsoby. Prvním způsobem je vložení definice funkcí do elementu *script*. Druhý způsob spočívá v nadefinování funkcí v odděleném souboru s příponou *.js*. Tento soubor je poté nutné k HTML dokumentu připojit opět prostřednictvím elementu *script*. K tomuto účelu slouží atribut *src*. Hodnota tohoto atributu musí odpovídat umístění souboru s JS.

```
<script src="/skripty/soubor.js"></script>
```

Ukázka 5 – Připojení odděleného souboru s JavaScriptem

3.3.1 JavaScriptové knihovny

Zápis kódu formou čistého JS je ve většině případů velice zdlouhavý a někdy také nepřehledný. Tento fakt vedl k vytvoření JS knihoven, které takový zápis značně zkracují a zpřehledňují. Tyto knihovny s sebou přináší také nové funkce, jejichž použití je velice snadné a intuitivní.

JQuery knihovna

Jednou z takových knihoven je jQuery. Knihovna je volně dostupná na internetu a aby bylo možné tuto knihovnu použít v rámci webové stránky či aplikaci, je nutné ji připojit do HTML dokumentu výše zmíněným způsobem.

Zdroj [8]

```
function exampleFunction() {
   var date = new Date();
   $("#test").html(date);
}
```

Ukázka 6 – Funkce z ukázky 4 požívající jQuery

V ARS je JS používán v základní podobě i prostřednictvím knihovny jQuery. JS je také základ technologie Google Apps Script, které se věnuje následující kapitola.

4 GOOGLE APPS SCRIPT

Google Apps Script je skriptovací jazyk založený na JS, který uživatelům umožňuje vytvářet nové doplňky a rozšíření pro stávající webové aplikace od Googlu. Pomocí GAS je mimo jiné možné:

- vytvořit a přidat vlastní menu do Dokumentů Google, Tabulek a Formulářů,
- napsat vlastní funkce pro Tabulky Google,
- vytvářet vlastní webové aplikace,
- propojit mezi sebou různé aplikace od Google (Kalendář, Disk, Gmail, Mapy aj.)
- vytvářet rozšiřující doplňky k Dokumentům, Tabulkám a Formulářům Google.

Velká výhoda GAS spočívá v tom, že uživateli pro tvorbu skriptů stačí pouze webový prohlížeč. Google nabízí vlastní editor skriptů a samotné skripty se provádí na serverech Googlu.

V editoru skriptů je potřeba vytvořit nový projekt. Z projektu má uživatel přístup ke kompletnímu aplikačnímu rozhraní GAS nabízející široké spektrum nástrojů a utilit, které může využít pro účely své aplikace. V projektu je automaticky vytvořen soubor Kód.gs. Tento soubor obsahuje hlavní logiku aplikace, která je rozdělena do jednotlivých funkcí. V projektu je možné dále vytvářet soubory .html, které jsou nezbytné pro vytvoření uživatelského rozhraní.

Kompletní aplikační rozhraní GAS je velmi rozsáhlé. Proto se tato práce omezuje pouze na popis nástrojů, tříd a metod, které byly použity v aplikaci, a jejich případných alternativ.

Zdroj [9]

4.1 HTML Service

HTML Service je nástroj GAS, který podporuje tvorbu a práci s HTML dokumenty prostřednictvím skriptu. Součástí tohoto nástroje jsou některá bezpečnostní opatření, která brání předávání HTML výstupu prohlížeči přímo. Před tím, než je HTML výstup předán prohlížeči je nejprve zkontrolován a případně jsou z něj odebrány některé potenciálně škodlivé části. Pro tento účel je používán tzv. Google Caja[6]. Zdroj [10]

4.1.1 Třída *HtmlOutput*

Jedná se o třídu GAS, která zapouzdřuje obsah ve formátu HTML. Objekt odvozený od této třídy může zahrnovat také kaskádové styly a JS a je předáván prohlížeči pro zpracování.

[6] Více informací o Google Caja: https://developers.google.com/caja/

Metoda *createHtmlOutput()*

Tato metoda vytvoří nový prázdný *HtmlOutput* objekt. S takovým objektem je poté možné manipulovat, zejména měnit jeho obsah, prostřednictvím aplikačního rozhraní GAS.

```
var htmlOutput = HtmlService.createHtmlOutput();
```

Ukázka 7 – Použití *createHtmlOutput()*

Metoda *createHtmlOutputFromFile()*

Nástroj HTML Service umožňuje také vytvoření *HtmlOutput* objektu ze souboru. K tomu slouží právě tato metoda. Metoda má jediný vstupní parametr, kterým je textový řetězec obsahující název souboru. Soubor musí být umístěn v projektu a musí splňovat požadavky standardu HTML. Zdroj [11]

```
var htmlOutput = HtmlService.createHtmlOutputFromFile('soubor');
```

Ukázka 8 – Použití *createHtmlOutputFromFile()*

4.1.2 Skriptlet

Skriptlet (v anglickém originále Scriptlet) je jazyková konstrukce, která umožňuje umístění a provedení kódu GAS uvnitř HTML dokumentu. Výsledkem je potom webová stránka, jejíž obsah byl dynamicky vygenerován na základě logiky a dat, která byla vyhodnocena pomocí GAS. Takovýmto způsobem lze například v cyklu vytvořit několik HTML elementů nebo rozhodnout, za jakých podmínek daný element vytvořit či nevytvořit. Stejně tak je možné do těchto elementů vložit data získané pomocí funkcí GAS. Skriptlet je možné použít ve dvou základních variantách.

Standard scriptlet

Tato varianta spustí kód uvnitř svého těla. Tímto způsobem ale nemůže být vložen žádný výstup do HTML dokumentu. Lze do něj ovšem vložit logiku, podle které je vygenerován výstup.

```
<? var a = 2; ?>
<? var b = 2; ?>
<? if (a == b) { ?>
<p>Odstavec, který BUDE vytvořen</p>
<? } else { ?>
<p>Odstavec, který NEBUDE vytvořen</p>
<? } ?>
```

Ukázka 9 – Standard scriptlet

Printing scriptlet

Jedná se o předchozí variantu vylepšenou o možnost vložení návratové hodnoty provedeného kódu do těla HTML dokumentu. Následující ukázka vytvoří v HTML dokumentu odstavec, jehož obsah bude "Kočka".

```
<? var data = ['Pes', 'Kočka', 'Kůň']; ?>
<p><?= data[1] ?></p>
```

Ukázka 10 – Printing scriptlet

Kód skriptletu se provádí před tím, než je vytvořena a vykreslena HTML stránka. Proto je možné tento kód spustit pouze jednou v rámci jednoho načtení stránky. Pro opětovné provedení tohoto kódu je třeba zvonu načíst celou stránku, případně znovu vytvořit část HTML stránky, v níž se kód nachází. Zdroj [12]

Aby bylo možné vykreslit HTML stránku obsahující skriptlet, je třeba z takového dokumentu nejprve vytvořit tzv. šablonu. Tomuto postupu se věnuje následující pododdíl.

4.1.3 Třída *HtmlTemplate*

Tato třída GAS umožňuje dynamickou tvorbu webových stránek na základě HTML dokumentů, které obsahují skriptlet.

Metoda *createTemplateFromFile()*

Tato metoda přijímá jako vstupní parametr textový řetězec s názvem HTML dokumentu. Metoda zpracuje HTML dokument včetně skriptletů v něm obsažených a jako výstup vytvoří objekt typu *HtmlTemplate*. Takový objekt ještě ale nelze předat prohlížeči k vykreslení. Nad takto vytvořenou šablonou je následně nutné provést metodu *evaluate()*. Tato metoda z šablony vytvoří objekt typu *HtmlOutput*, který je poté možné předat prohlížeči k vykreslení.

Zdroj [13]

```
var html = HtmlService.createTemplateFromFile('soubor').evaluate();
```

Ukázka 11 – Použití *createTemplateFromFile()* a *evaluate()*

4.1.4 Třída *google.script.run*

Třída *google.script.run* je nástroj aplikačního rozhraní určený k tomu, aby bylo možné na straně klienta prostřednictvím JS volat GAS funkce na straně serveru. Volání funkcí na serveru probíhá asynchronně, takže se nečeká na jejich dokončení. Pokud je na straně klienta potřebná návratová hodnota volané funkce, je nutné tuto operaci provést synchronně. Pro tento účel třída nabízí metody *withSuccessHandler()* a *withFailureHandler().*

```
function funkceNaStraneKlienta() {
   // kód
   google.script.run.funkceNaServeru();
   // kód
}
```

Ukázka 12 – Asynchronní volání funkce na serveru

Dojde-li k zavolání funkce *funkceNaStraneKlienta(),* standardně se začne provádět tělo funkce. Když dojde k zavolání funkce *funkceNaServeru()* prostřednictvím *google.script.run*, je odeslán požadavek na server. Prohlížeč nečeká na jeho dokončení a ihned pokračuje v provádění dalšího kódu.

Metoda *withSuccessHandler()*

Pokud na straně klienta dojde k zavolání serverové funkce prostřednictvím této metody, provádění dalšího kódu funkce na straně klienta se přeruší a pokračuje až v momentě, kdy dojde k úspěšnému návratu z těla dané funkce na serveru.

Na straně klienta je nutné definovat funkci, která bude provedena ve chvíli, kdy je ze serveru vrácena případná návratová hodnota volané funkce. Tato funkce (tzv. callback) přijímá jako vstupní parametr právě onu návratovou hodnotu serverové funkce. Název této klientské funkce je nutné předat metodě *withSuccessHandler()* jako vstupní parametr.

```
function callback(parametr) {
   alert(parametr);
}

Function funkceNaStraneKlienta() {
   // kód
   google.script.run.withSuccessHandler(callback).funkceNaServeru();
   // kód
}
```

Ukázka 13 – Použití *withSuccessHandler()*

Dojde-li k zavolání funkce *funkceNaStraneKlienta()*, je nejprve vyhodnocena funkce *funkceNaServeru()*. Její případná návratová hodnota je předána funkci *callback()*, která je rovněž provedena. Prohlížeč poté pokračuje v provádění případného kódu, který následuje.

Metoda *withFailureHandler()*

Principiálně se jedná o stejnou metodu jen s tím rozdílem, že signálem pro pokračování provádění funkce na straně klienta není úspěšný návrat z těla funkce na serveru, ale situace, pokud dojde k výskytu výjimky. Zdroj [14]

4.2 Calendar Service

Tento nástroj poskytuje výkonné aplikační rozhraní pro práci s KG. Prostřednictvím tohoto rozhraní má uživatel přístup ke svým kalendářům. Uživatel má mimo jiné možnost:

- vytvořit nový kalendář,
- upravovat vlastnosti a vzhled existujícího kalendáře,
- vytvářet, upravovat a mazat události v těchto kalendářích.

Metoda *getCalendarById()*

Aby bylo možné s konkrétním kalendářem manipulovat, je třeba daný kalendář zpřístupnit. K tomu to účelu slouží metoda *getCalendarById()*, která přijímá jako jediný vstupní parametr textový řetězec s identifikátorem kalendáře. Pokud identifikátor odpovídá existujícímu kalendáři a uživatel k němu může přistupovat, metoda jako svou návratovou hodnotu vrátí tento kalendář v podobě objektu typu *Calendar*. V případě, že jedna z podmínek není splněna, metoda vrátí hodnotu *null*. Zdroj [15]

```
var id = 'ol0jffr19r3p44aqjdrk386n0s@group.calendar.google.com';
var calendar = CalendarApp.getCalendarById(id);
```

Ukázka 14 – Použití *getCalendarById()* a příklad identifikátoru kalendáře

4.2.1 Třída *Calendar*

Jedná se o objektový typ, který zapouzdřuje veškeré vlastnosti kalendáře. Nad objektem tohoto typu lze snadno provádět operace, určené pro manipulaci s jeho obsahem a vlastnostmi. Hlavní účel těchto operací je správa událostí. K tomu slouží následující metody.

Metoda *createEvent()*

Metoda slouží k jednorázovému vytvoření událost v daném kalendáři. Přijímá tři vstupní parametry: název události, datum a čas začátku události, datum a čas konce události. Název události je funkci předán v podobě textového řetězce. Datum a čas začátku i konce události musí být JS objekty typu *Date*. V případě úspěšného vytvoření události metoda vrátí tuto událost jako objekt typu *CalendarEvent*. S tímto objektem je možné dále manipulovat. Této problematice se věnuje samostatný pododdíl.

```
var title = 'Název události';
var start = new Date('July 17 2017 15:00:00 GMT+0200');
var end = new Date('July 17 2017 16:30:00 GMT+0200');
var event = calendar.createEvent(title, start, end);
```

Ukázka 15 – Vytvoření události

V této ukázce kódu dojde k vytvoření proměnných, které jsou potřebné pro vytvoření události. Poté následuje vytvoření této události v kalendáři *calendar*, který byl zpřístupněn stejným způsobem jako v předchozí ukázce (viz Ukázka 14).

Metoda *createEventSeries()*

Tato metoda funguje obdobně jako *createEvent()*. Díky této metodě je možné vytvořit událost, která se bude v kalendáři periodicky opakovat. V případě úspěšného provedení tedy ve výsledku dojde k vytvoření několika událostí podle specifikovaného pravidla o opakování. Jako vstupní parametr přijímá metoda kromě názvu události, jejího začátku a konce také objekt typu *EventRecurrence*, který obsahuje toto pravidlo. Tento objekt specifikuje do kdy a jak často se má událost opakovat.

Jako návratová hodnota metody je vrácena celá série typu *CalendarEventSeries*, která v sobě zapouzdřuje několik událostí typu *CalendarEvent*, se kterými je možné samostatně manipulovat.

```
var until = new Date('September 1 2017 00:00:00 GMT+0200');
var rec = CalendarApp.newRecurrance().addWeeklyRule().until(until);
var eventSerie = calendar.createEventSeries(title, start, end, rec);
```

Ukázka 16 – Vytvoření opakované události[7]

[7] V kódu jsou použity proměnné *calendar title*, *start* a *end* z ukázek 14 a 15

Metoda *getEvents()*

Třída *Calendar* nabízí také metodu pro zpřístupnění svých událostí. Metoda přijímá dva vstupní parametry typu *Date*, které určují začátek a konec události. Návratovou hodnotou metody je *pole* objektů typu *CalendarEvent*, které představuje všechny události, které mají svůj začátek a konec v rozmezí hodnot vstupních parametrů. Pro získání jedné konkrétní události je proto nutné prostřednictvím těchto parametrů předat přesné datum a čas začátku a konce události. Pokud v daném kalendáři neexistují dvě události v totožný čas, je jako návratová hodnota předáno pole objektů *CalendarEvent* o délce 1. Pro získání jedné konkrétní události je proto ještě nutné zpřístupnit položku tohoto pole na indexu 0. Zdroj [16]

```
var event = calendar.getEvents(startTime, endTime)[0];
```

Ukázka 17 – Zpřístupnění události

4.2.2 Třída *CalendarEvent*

Objekty této třídy představují jednotlivé události kalendáře. Třída zapouzdřuje množinu vlastností definujících danou událost, a metod, které mimo jiné umožňují tyto vlastnosti spravovat. Tento pododdíl se bude zabývat popisem některých těchto vlastností a metod použitých v ARS.

Čas události

Základní a zároveň nejdůležitější vlastnosti události jsou její čas začátku a čas konce. Bez těchto vlastností by objekt typu *CalendarEvent* postrádal smysl. Jedná se o dva JS objekty typu *Date*. K jejich nastavení dochází už během vytvoření události. Jejich hodnota lze také později přenastavit pomocí metody *setTime()*. Tato metoda přijímá jako své dva vstupní parametry dva objekty typu *Date*, na jejichž hodnoty přenastaví vlastnosti začátku a konce události. Hodnoty těchto vlastností lze získat pomocí metod *getStartTime()* a *getEndTime()*.

Popis události

Tato vlastnost slouží k nastavení libovolného textového popisu události. K nastavení slouží metoda *setDescription()*, která textový řetězec s popisem jako svůj jediný vstupní parametr. Zpřístupnění má na starost metoda *getDescription()*. Hodnota této vlastnosti je viditelná prostřednictvím uživatelského rozhraní KG a může sloužit např. k poskytnutí bližších informací potenciálním hostům události.

Tag

Jedná se o vlastnost typu klíč-hodnota. Klíč i hodnota jsou textové řetězce. Její primární účel je možnost uložení určité informace o události. Tato informace není přístupná prostřednictvím uživatelského rozhraní. Počet tagů pro jednu událost není omezen. Nastavení tagu umožňuje metoda *setTag()*, která přijímá klíč a hodnotu jako své vstupní parametry. Metoda *getTag()* vrátí hodnotu na základě klíče, který je předán jako vstupní parametr. Pokud daný klíč neexistuje, je vrácena hodnota *null*. Obdobně funguje také metoda *deleteTag()*, která na základě klíče tag odstraní.

Upomínka

Účinný nástroj, jehož název vypovídá za vše. Každé události lze nastavit libovolný počet upozornění. Aplikační rozhraní kalendáře umožňuje tři způsoby upozornění: emailem, vyskakovacím oknem a zprávou SMS. K nastavení upozornění emailem slouží metoda *addEmailReminder()*, která zajistí zaslání upozornění na emailovou adresu stanovený počet minut před začátkem události. Počet minut je metodě předán prostřednictvím vstupního parametru typu Integer (celočíselný datový typ). Obdobným způsobem fungují další dvě zmíněná upozornění prostřednictvím metod *addPopupReminder()* a *addSmsReminder()*. Upomínky je možné smazat pomocí metody *removeAllReminders()* nebo nastavit na výchozí dle nastavení kalendáře pomocí funkce *resetRemindersToDefault()*.

Smazání události

Konkrétní událost je z kalendáře možné smazat použitím metody *deleteEvent()*.

Všechny výše zmíněné metody jsou volány nad konkrétním objektem typu *CalendarEvent* prostřednictvím tečkové konvence. Zdroj [17]

```
var event = calendar.getEvents(startTime, endTime)[0];
event.deleteEvent();
```

Ukázka 18 – Smazaní události

4.3 Properties Service

Tento nástroj umožňuje ukládat jednoduchá data v podobě párů klíč-hodnota. Klíč i hodnota jsou textové řetězce. Existují tři druhy vlastností. Žádný druh vlastností nelze sdílet mezi více skripty.

4.3.1 Vlastnosti skriptu

Tyto vlastnosti jsou přístupné v rámci celého skriptu. Všichni uživatelé, kteří používají daný skript mají k těmto vlastnostem přístup a mohou je měnit.

4.3.2 Vlastnosti uživatele

Tento druh umožňuje ukládat vlastnosti, ke kterým bude mít přístup pouze současný uživatel skriptu. Tento uživatel může tyto vlastnosti prostřednictvím skriptu měnit.

4.3.3 Vlastnosti dokumentu

Tyto vlastnosti jsou dostupné všem uživatelům v rámci otevřeného dokumentu, ve kterém je skript používán.

Následující ukázka demonstruje použití vlastností skriptu. Práce se zbylými dvěma druhy vlastností je obdobná. Liší se pouze v názvu metod. Zdroj [18]

```
var scriptProperties = PropertiesService.getScriptProperties();
scriptProperties.setProperty('klíč','hodnota');
var value = scriptProperties.getProperty('klíč');
```

Ukázka 19 – Použití vlastností skriptu

Pokud vlastnost s klíčem `'klíč'` neexistuje, metoda *getProperty()* vrátí *null*.

4.4 *Session*

Třída *Session* poskytuje přístup k informacím o relaci, jako jsou např. email uživatele, který spustil skript, jeho jazykové preference, email uživatele, pod jehož autoritou skript běží, a další.

Pro možnost získání emailové adresy platí jistá omezení, která jsou blíže popsána v pododdíle 2.4.1. Omezení autentizace. Zdroj [19]

```
var email = Session.getEffectiveUser().getEmail();
```

Ukázka 20 – Získání emailové adresy uživatele, pod jehož autoritou je skript spuštěn

4.5 *ScriptApp*

Třída *ScriptApp* je nástroj GAS, který zprostředkuje přístup a manipulaci se skriptem. Umožňuje definovat tzv. spouště, které provedou nějakou operaci, když dojde k určité události, např. instalaci skriptu, načtení aj. Autor skriptu má dále díky této třídě přístup k informacím o skriptu, jako je např. jeho adresa URL, pokud je skript publikován. Třída také poskytuje autorovi kontrolu nad tím, pro koho a od jaké míry bude skript dostupný. Zdroj [20]

V ARS je tento nástroj použit pouze jako prostředek k získání adresy URL skriptu.

```
var url = ScriptApp.getService().getUrl();
```

Ukázka 21 – Získání adresy URL skriptu

5 IMPLEMENTACE POŽADAVKŮ

Tato kapitola se věnuje konkrétním postupům při implementaci požadavků zmíněných v kapitole 2. V kapitole je řečeno, jakým způsobem je daný požadavek realizován. Je zde také stručně popsána obsluha aplikace.

5.1 Spuštění aplikace

Ve chvíli, kdy je aplikace zprovozněna jako webová aplikace a přístup k ní je nastaven pro kohokoli, může aplikaci použít kdokoli, kdo má příslušnou adresu URL.

Při spuštění aplikace po zadání příslušné adresy URL je jako první provedena funkce skriptu *doGet()*. Primární úkol této funkce je vytvoření uživatelského rozhraní na základě HTML dokumentu umístěného v projektu. Tato funkce je nezbytná pro fungování webové aplikace. Funkce přijímá jeden vstupní parametr. Jedná se o objekt, který obsahuje informace o parametrech URL. Alternativou je funkce *doPost()*, která je provedena v případě, že je ke skriptu přistupováno použitím metody post protokolu HTTP.

V ARS jsou rozlišeny dva základní přístupy (více v kapitole 2) a pro každý je dostupná alespoň jedna HTML stránka. Je tedy nutné, aby bylo ve funkci *doGet()* rozlišeno, co za uživatele spouští skript. Na základě této vnitřní logiky je vytvořena příslušná stránka a ta je předána prohlížeči k vykreslení.

5.1.1 Rozlišení klienta a poskytovatele

```
function doGet(e){

  var emailActive = Session.getActiveUser().getEmail();
  var emailEffective = Session.getEffectiveUser().getEmail();

  if(emailActive == emailEffective){
    if(getCalendarIDProperty() != null) Utilities.sleep(500);
    if(!e.parameter.calID && getCalendarIDProperty() == null){
      return
HtmlService.createTemplateFromFile('webProvider').evaluate();
    }else{
      return
HtmlService.createTemplateFromFile('webProviderCalendarSet').evaluate();
    }
  }else{
    return HtmlService.createTemplateFromFile('webClient').evaluate();
  }
}
```

Ukázka 22 – implementace funkce *doGet()* v ARS

Ve funkci nejprve dojde k ověření, zda se jedná o autora skriptu základě emailové adresy uživatele. Pokud se email uživatele neshoduje s emailem autora, je vykreslena webová stránka pro klienta. V opačném případě následuje rozhodování, která stránka pro poskytovatele má být předána prohlížeči. Pokud není nastavena vlastnost skriptu s identifikátorem kalendáře a zároveň tento identifikátor není předán jako parametr URL, je poskytovateli vykreslena stránka, která slouží pro nastavení této vlastnosti skriptu. Z této stránky se lze po nastavení identifikátoru kalendáře přesměrovat na stránku se správou schůzek. Vytváření HTML stránek je popsáno v oddíle 4.1 HTML Service.

5.2 Část pro poskytovatele

Následující oddíl je věnován způsobu implementace aplikace z pohledu poskytovatele. Je zde ve stručnosti popsána také obsluha aplikace a zmíněny některé úkony, které je nutné provést v aplikaci KG před tím, než je možné ARS začít používat.

5.2.1 Nastavení kalendáře

Aby bylo možné aplikaci používat, je nutná integrace s Kalendářem Google. K tomu je potřebné si prostřednictvím KG vytvořit jeden nebo více kalendářů, které bude později možné využít v rámci ARS. Zvolený kalendář je kdykoli možné zaměnit za jiný, ale současně lze použít právě jeden kalendář.

Prostřednictvím aplikace KG je možné ve vlastnostech konkrétního kalendáře zjistit jeho identifikátor. Tato hodnota je klíčová pro následnou práci s daným kalendářem.

Pro nastavení identifikátoru kalendáře, který má být použit v rámci ARS, slouží jednoduchá stránka, jenž je definována v souboru *webProvider.html*. Tato stránka nabízí velmi jednoduché uživatelské rozhraní. Součástí tohoto rozhraní je textové pole pro vložení identifikátoru kalendáře a dvě tlačítka pro nastavení a následné přesměrování na stránku se správou schůzek v tomto kalendáři.

K uchování hodnoty identifikátoru kalendáře jsou použity vlastnosti skriptu (viz 4.3.1 Vlastnosti skriptu).

Pro uložení identifikátoru kalendáře do vlastnosti skriptu stačí tento identifikátor vyplnit do textového pole na stránce pro nastavení a kliknout na tlačítko *Nastavit*. Toto tlačítko vyvolá serverovou funkci skriptu *saveCalendarIDProperty()*, které je předána jako vstupní parametr právě hodnota identifikátoru. V případě úspěchu je uživatel informován zprávou ve vyskakovacím okně a může být přesměrován na stránku se správou schůzek.

```
function saveCalendarIDProperty(inputForm){
  var calID = inputForm.calendarID;
  if(checkIDProperty(calID)){
    var scriptProperties = PropertiesService.getScriptProperties();
    scriptProperties.setProperty('CALENDAR_ID', inputForm.calendarID);
    return true;
  }else{
    return false;
  }
}
```

Ukázka 23 – ukládání identifikátoru kalendáře jako vlastnost skriptu

Serverová funkce *saveCalendarIDProperty()* přijímá jako vstupní parametr formulář, který obsahuje identifikátor kalendáře. Funkce *checkIDProperty()* zkontroluje, zda hodnota proměnné *calID* je různá od *null,* a zda zadanému identifikátoru odpovídá nějaký poskytovatelův kalendář. Pokud ano, dojde k uložení vlastnosti a je vrácena hodnota *true*, která určuje, jaká zpráva má být uživateli zobrazena.

```
function getCalendarIDProperty(){
  var calID =
PropertiesService.getScriptProperties().getProperty('CALENDAR_ID');
  if(checkIDProperty(calID)){
    return calID;
  }else{
    return null;
  }
}
```

Ukázka 24 – Načítání identifikátoru kalendáře z vlastnosti skriptu

Serverová funkce *getCalendarIDProperty()* se pokusí získat hodnotu vlastnosti skriptu, kterou následně ověří pomocí funkce *checkIDProperty()*. Je-li tato hodnota platný identifikátor, je tato hodnota vrácena jako návratová hodnota. V opačném případě je vrácena hodnota *null*, což lze využít k ošetření chybových stavů.

```
function resetCalendarIDProperty(){
  if(PropertiesService.getScriptProperties().getProperty('CALENDAR_ID')
!= null){

PropertiesService.getScriptProperties().deleteProperty('CALENDAR_ID');
    return 'ID resetováno';
  }else{
    return 'ID nebylo nastaveno';
  }
}
```

Ukázka 25 – smazání vlastnosti skriptu

Aplikace nabízí možnost změny kalendáře. K tomu je nutné nejprve vymazat vlastnost skriptu se stávajícím identifikátorem. K tomu slouží serverová funkce *resetCalendarIDProperty()*.

5.2.2 Vytváření schůzek

Poskytovatel má pro vytváření schůzek k dispozici jednoduché grafické rozhraní s formulářem. Hodnoty z tohoto formuláře jsou předány serverové funkci *createEventsFromInput()* (viz Přílohy). Funkce na základě těchto hodnot vytvoří požadované události do nastaveného kalendáře pomocí funkce *createEvent()* (viz 4.2.1 Třída *Calendar*). Funkce nejprve upraví datum a čas na textové řetězce, ze kterých je poté možné vytvořit potřebné objekty *Date*. Dále rozhodne, zda se má událost periodicky opakovat, a zda k ní má být vytvořené upozornění emailem. Jako návratová hodnota je předán objekt typu *HtmlOutput*, který na straně klienta slouží pro obnovení obsahu stránky.

5.2.3 Přehled schůzek

Po vytvoření schůzek má poskytovatel možnost zobrazit si jejich seznam. V rámci tohoto seznamu má také možnost jednotlivé termíny odstranit, případně upravit čas začátku a konce. Výpis seznamu je v HTML stránce realizován pomocí HTML šablony (viz 4.1.3 Třída *HtmlTemplate*) a skriptletů (viz 4.1.2 Skriptlet). Kód, který tento výpis vytváří lze nalézt na konci práce (viz Přílohy).

5.2.4 Úprava času schůzky

Každá položka seznamu schůzek má také zaškrtávací políčko s názvem *Editovat*. Po označení tohoto políčka se u dané položky zobrazí dvě vstupní pole a tlačítko *Změnit*. Do vstupních polí je nutné vyplnit nový čas začátku a konce události a poté tlačítkem vyvolat serverovou funkci *editEvent()* (viz Přílohy). Tato funkce zpřístupní požadovanou událost na základě jejího začátku a konce a tyto časy následně změní pomocí funkce *setTime()* (viz 4.2.1 Třída *Calendar*).

5.2.5 Odstranění schůzky

Podobně jako lze jednotlivé schůzky editovat, je lze také odstranit. U každé položky v seznamu je vytvořeno tlačítko Odstranit. Kliknutím na toto tlačítko je spuštěna serverová funkce *removeEvent()*, která příslušný termín vymaže z kalendáře.

```
function removeEvent(start, end){
  var startDate = new Date(start);
  var endDate = new Date(end);
  CalendarApp.getCalendarById(getCalendarIDProperty())
                        .getEvents(startDate, endDate)[0]
                        .deleteEvent();
}
```

Ukázka 26 – Serverová funkce *removeEvent()*

5.3 Část pro klienta

Tento oddíl pojednává o způsobu implementace aplikace z pohledu klienta. Možnosti klienta se v aplikaci prakticky omezují jen na přehled nerezervovaných termínů schůzek a na možnost jejich rezervace po vyplnění jména a příjmení prostřednictvím dvou jednoduchých textových vstupů.

5.3.1 Výpis volných termínů

Výpis termínů je v části klienta realizován obdobně jako v části poskytovatele, jen s drobnými omezeními. Klient například nepotřebuje vědět, které termíny jsou obsazeny a kým. Proto se tento výpis omezuje pouze na neobsazené termíny.

5.3.2 Zápis na termíny

Každá položka výpisu disponuje tlačítkem *Zapsat*. Kliknutí na toto tlačítko vyvolá serverovou funkci *setReserved()* (viz Přílohy). Funkce zkontroluje, zda v mezičase nedošlo k obsazení jiným uživatelem. Pokud ano, je jako návratová hodnota vrácen textový řetězec s informací, že nebylo možné daný termín rezervovat. V opačném případě funkce do popisu události zapíše informaci o tom, kým byl termín obsazen a jako návratová hodnota je předán textový řetězec s informací o úspěšné rezervaci.

Funkce dále na danou schůzku vytvoří upomínky pro poskytovatele, pokud byly nadefinovány při vytvoření termínu.

Informaci o tom, zda je termín volný či obsazený, je možné uchovávat také prostřednictvím tagů zmíněných v oddílu 4.2.2 Třída *CalendarEvent*. Nicméně v popisu události informace o obsazenosti musí být také, takže by toto řešení bylo pro účely aplikace nadbytečné.

5.4 Volání funkcí na serveru

Výše zmíněné funkce skriptu jsou prováděny na serveru. Pro spuštění těchto funkcí ze strany klienta slouží třída *google.script.run* (viz 4.1.4 Třída *google.script.run*).

```
google.script.run.withSuccessHandler(refreshApp)
                                .createEventsFromInput(document.forms[0]);
```

Ukázka 27 – Použití *google.script.run* v ARS

5.5 Zpřístupnění událostí

Ke zpřístupňování jednotlivých událostí slouží metoda *getEvents()* třídy *Calendar* (viz 4.2.1 Třída *Calendar*). Aby bylo možné této metodě předat informaci o datumu a čase začátku a konce příslušné události, je nutné tuto informaci získat a uchovat někde na straně klienta. K získání těchto informací slouží metody *getStartTime()* a *getEndTime()* třídy *CalendarEvent* (viz 4.2.2 Třída *CalendarEvent*). Takto získaný formát je velmi špatně čitelný pro uživatele, nicméně je nutné ho dodržet při předání metodě *getEvents()*. K uchování na straně klienta byl využit HTML element *td* interpretující buňku tabulky, na který bylo pomocí atributu *class* aplikováno CSS pravidlo `display: none`. HTML element s touto CSS vlastností není na stránce nijak zobrazen a nijak se při vykreslení neprojeví.

5.6 Grafické zobrazení kalendáře v aplikaci

Do webové aplikace je možné umístit grafickou podobu kalendáře. K tomu slouží HTML element *iframe*. S takto zobrazeným kalendářem není možné manipulovat. Nicméně klient má alespoň možnost si svoji zarezervovanou schůzku zkopírovat do vlastního KG.

```
<div>
      <iframe id="idkalendare"
src="https://calendar.google.com/calendar/embed?src=
<?=getCalendarIDProperty()?>&ctz=Europe/Prague" style="border: 0"
width="800" height="600" frameborder="0" scrolling="no">
      </iframe>
</div>
```

Ukázka 28 – Zobrazení kalendáře v ARS

ZÁVĚR

Vytvoření praktické části předcházelo dlouhé studium a seznamování se s rozsáhlým aplikačním rozhraním Google Apps script, které mi znatelně rozšířilo obzory v problematice tvorby webových aplikací. V praxi jsem si mohl vyzkoušet a použít nejmodernější technologie pro tvorbu webových aplikací od společnosti Google a na jejich základě vytvořit vlastní webovou aplikaci.

Výsledek praktické části určitě není dokonalý. Nicméně do značné míry splňuje stanovené základní funkční požadavky. Prostřednictvím aplikace je možné jednoduše vytvářet, upravovat a odstraňovat události v kalendáři a tyto události také poskytnout k rezervaci. Možností optimalizace se nabízí hned několik.

Část aplikace pro poskytovatele nabízí vcelku přívětivé uživatelské rozhraní pro správu událostí. Jako nedostatek se může jevit fakt, že některé úkony trvají někdy 2 až 4 vteřiny a během tohoto času se může zdát, že se nic neděje. To může uživatele svádět k tomu, zkusit něco znovu, aniž by počkal, zda se to provede. Dobré vylepšení by tedy mohlo být například přidání určitého ukazatele, který by značil, že se v aplikaci provádí nějaká funkce a je potřeba počkat na její dokončení.

Registrace na termíny schůzek na straně klienta funguje bez problému. Značným omezením v této části je nedostatečná autentizace uživatele používajícího aplikaci. I když je nutné, aby byl uživatel přihlášen prostřednictvím svého Google účtu, není možné ho v rámci aplikace podle tohoto účtu identifikovat a tím rozlišit od ostatních. Ačkoliv v GAS existuje nástroj na získání emailové adresy přihlášeného uživatele, existují určitá autorizační opatření, která znemožňují získat adresu uživatele, který není ze stejné G Suite domény, jako vlastník skriptu (např. student.upce.cz). Během návrhu a implementace se nepodařilo tento nedostatek nijak obejít nebo odstranit. Klient tak proto není schopen zrušit svoji rezervaci a je nucen pro tento účel kontaktovat poskytovatele. Do budoucna by proto bylo vhodné se zkusit ještě více zaměřit na tuto problematiku. Pokud by se tento nedostatek podařilo odstranit, tak by také stálo za zvážení určité omezení na počet rezervací, které může jeden klient uskutečnit.

POUŽITÁ LITERATURA

[1] Nápověda Kalendář. *Google* [online]. 2017 [cit. 2017-05-09]. Dostupné z: https://support.google.com/calendar/answer/190998?hl=cs

[2] Premium Doodle. *Doodle* [online]. Berlín [cit. 2017-05-09]. Dostupné z: https://beta.doodle.com/cs/premium

[3] Pricing. *Setmore* [online]. [cit. 2017-05-09]. Dostupné z: https://www.setmore.com/premium

[4] Class User. *Google Apps Script* [online]. 26. duben 2017 [cit. 2017-05-09]. Dostupné z: https://developers.google.com/apps-script/reference/base/user

[5] HTML5. *W3C* [online]. 28. říjen 2014 [cit. 2017-05-09]. Dostupné z: https://www.w3.org/TR/html5/introduction.html#a-quick-introduction-to-html

[6] LIE, Håkon Wium a Bert BOS. *Cascading Style Sheets: Designing for the Web*. 3rd ed. Addison-Wesley Professional, 2005. ISBN 0-321-19312-1. Citována online verze, kapitola 2. Dostupné z: https://www.w3.org/Style/LieBos2e/enter

[7] JavaScript Web APIs. *W3C* [online]. 2016 [cit. 2017-05-10]. Dostupné z: https://www.w3.org/standards/webdesign/script

[8] JQuery API. *JQuery* [online]. 2017 [cit. 2017-05-10]. Dostupné z: http://api.jquery.com/

[9] Overview of Google Apps Script. *Google Apps Script* [online]. 12. srpen 2016 [cit. 2017-05-10]. Dostupné z: https://developers.google.com/apps-script/overview

[10] Class HtmlService. *Google Apps Script* [online]. 23. listopad 2016 [cit. 2017-05-10]. Dostupné z: https://developers.google.com/apps-script/reference/html/html-service

[11] Class HtmlOutput. *Google Apps Script* [online]. 21. únor 2016 [cit. 2017-05-10]. Dostupné z: https://developers.google.com/apps-script/reference/html/html-output

[12] HTML Service: Templated HTML. *Google Apps Script* [online]. 10. listopad 2016 [cit. 2017-05-10]. Dostupné z: https://developers.google.com/apps-script/guides/html/templates

[13] Class HtmlTemplate. *Google Apps Script* [online]. 19. květen 2015 [cit. 2017-05-10]. Dostupné z: https://developers.google.com/apps-script/reference/html/html-template

[14] Class google.script.run. *Google Apps Script* [online]. 13. červenec 2016 [cit. 2017-05-10]. Dostupné z: https://developers.google.com/apps-script/guides/html/reference/run

[15] Class CalendarApp. *Google Apps Script* [online]. 26. duben 2017 [cit. 2017-05-10]. Dostupné z: https://developers.google.com/apps-script/reference/calendar/calendar-app

[16] Class Calendar. *Google Apps Script* [online]. 11. prosinec 2015 [cit. 2017-05-10]. Dostupné z: https://developers.google.com/apps-script/reference/calendar/calendar

[17] Class CalendarEvent. *Google Apps Script* [online]. 26. duben 2017 [cit. 2017-05-10]. Dostupné z: https://developers.google.com/apps-script/reference/calendar/calendar-event

[18] Class PropertiesService. *Google Apps Script* [online]. 26. duben 2017 [cit. 2017-05-10]. Dostupné z: https://developers.google.com/apps-script/reference/properties/properties-service

[19] Class Session. *Google Apps Script* [online]. 26. duben 2017 [cit. 2017-05-10]. Dostupné z: https://developers.google.com/apps-script/reference/base/session

[20] Class ScriptApp. *Google Apps Script* [online]. 13. září 2016 [cit. 2017-05-10]. Dostupné z: https://developers.google.com/apps-script/reference/script/script-app

PŘÍLOHY

Příloha A – Serverová funkce createEventsFromInput()....44
Příloha B – Zdrojový kód pro vytvoření seznamu schůzek....45
Příloha C – Serverová funkce editEvent()....45
Příloha D – Serverová funkce setReserved()....46

Některé nepodstatné části kódu byly z ukázek vypuštěny

Kód v kompletní podobě je dostupný na přiloženém CD.

Příloha A – *Serverová funkce createEventsFromInput()*

```
function createEventsFromInput(inputForm){
 //Formátování textových řetězců s datumem (vypuštěno)
 //Vytváření objektů Date (vypuštěno)
 //Vytvoření dalších proměnných na základě vstupního formuláře (vypuštěno)
  while(endDateTemp <= endDateTimeBlock){
    if(inputForm.opakovat){
      switch(recSelect){
        case 0:
          recurrence =
CalendarApp.newRecurrence().addDailyRule().until(endDateTimeRecurrent);
          break;
        case 1:
          recurrence =
CalendarApp.newRecurrence().addWeeklyRule().until(endDateTimeRecurrent);
          break;
        case 2:
          recurrence =
CalendarApp.newRecurrence().addMonthlyRule().until(endDateTimeRecurrent);
          break;
        default: return null;
      }
      event = CalendarApp.getCalendarById(getCalendarIDProperty())
                          .createEventSeries(inputForm.nazev, startDateTemp,
                  endDateTemp, recurrence).setDescription('Neobsazeno');
    }else{
      event = CalendarApp.getCalendarById(getCalendarIDProperty())
                          .createEvent(inputForm.nazev, startDateTemp,
                  endDateTemp).setDescription('Neobsazeno');
    }
    if(inputForm.rem_hour){
      event.setTag('rem_hour', 'true');
    }

    if(inputForm.rem_day){
      event.setTag('rem_day', 'true');
    }

    startDateTemp.setMinutes(startDateTemp.getMinutes() + slotLength);
    endDateTemp.setMinutes(endDateTemp.getMinutes() + slotLength);
  }
  return getWebProviderCalSetHtml();
}
```

Příloha B – *Zdrojový kód pro vytvoření seznamu schůzek*

```
<div id="event-list">
   <h3>Termíny</h3>
  <?var events = getEventsOfCurrentCalendar();?>
  <?for(var i = 0; i < events.length; i++){?>
   <div class="table-pair">
    <table class="event-control" align="center">
     <tr>
      <td width="30%"><?=getEventTimeString(events[i])?></td>
      <td width="45%"><?=events[i].getDescription()?></td>
      <td width="25%">
       <input class="remove" type="button" value="Odebrat"/>
       <label>Editovat</label>
       <input class="edit-checkbox"type="checkbox"/>
      </td>
      <td class="inv-start"><?=events[i].getStartTime()?></td>
      <td class="inv-end"><?=events[i].getEndTime()?></td>
     </tr>
    </table>
    <table class="event-edit" align="center" hidden>
     <tr>
      <td class="td-right" width="100%">
       <?if(events[i].getDescription().length != 10){?>
         <input class="cancel" type="button" value="Zrušit"/>
       <?}?>
       <input class="edit-start" type="time"/>
       <input class="edit-end" type="time"/>
       <input class="edit" type="button" value="Změnit"/>
      </td>
     </tr>
    </table>
   </div>
  <?}?>
  </div>
```

Příloha C – *Serverová funkce editEvent()*

```
function editEvent(start, end, newStart, newEnd){
  var oldStartDate = new Date(start);
  var oldEndDate = new Date(end);
  var newStartDate = new Date(start);
  var newEndDate = new Date(end);
  newStartDate.setHours(Number(newStart.substring(0, 2)));
  newEndDate.setHours(Number(newEnd.substring(0, 2)));
  newStartDate.setMinutes(Number(newStart.substring(3)));
  newEndDate.setMinutes(Number(newEnd.substring(3)));
  var events = CalendarApp.getCalendarById(getCalendarIDProperty())
                          .getEvents(oldStartDate, oldEndDate);
  if(events.length > 0){
    events[0].setTime(newStartDate, newEndDate);
    return getWebProviderCalSetHtml();
  }else{
    return null;
  }
}
```

Příloha D – *Serverová funkce setReserved()*

```
function setReserved(start, end, name, surname){
  var startTime = new Date(start);
  var endTime = new Date(end);
  var byWho = name + ' ' + surname;
  var desc = 'Obsazeno - ' + byWho;
  var response = '';
  var event = CalendarApp.getCalendarById(getCalendarIDProperty())
                                    .getEvents(startTime, endTime)[0];
  if(event != null){
    if(event.getDescription() == 'Neobsazeno'){
      event.setDescription(desc);

      if(event.getTag('rem_hour') != null){
        event.addEmailReminder(60);
      }

      if(event.getTag('rem_day') != null){
event.addEmailReminder(countMinutesBeforeEventStart(event.getStartTime()));
      }

      response = 'Rezervace v čase ' + getEventTimeString(event) +
' úspěšná!\n' +
        'Kontrolu lze provést zobrazením popisu příslušné schůzky v níže
uvedeném kalendáři';
    }
    else response = 'Termín obsazen';
  }
  else response = 'Chyba – termín nenalezen';
  return response;
}
```