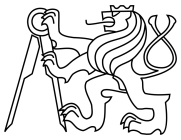
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

| | |
|---|---|
| **Název:** | 10K SAGE2 Dashboard: Sada widgetů pro velkoplošnou obrazovku |
| **Student:** | Radek Meduna |
| **Vedoucí:** | Ing. Radek Richtr |
| **Studijní program:** | Informatika |
| **Studijní obor:** | Web a multimédia |
| **Katedra:** | Katedra softwarového inženýrství |
| **Platnost zadání:** | Do konce letního semestru 2017/18 |

## Pokyny pro vypracování

Úkolem je vytvořit specializovaný dashboard sestávající ze sady SAGE2 (Scalable Amplified Group Environment, velkoplošné zobrazovací virtualizační zařízení tvořené 20 LCD displeji řízené sw. SAGE a SAGE2 postaveným na web. technologiích) aplikací schopných získat a vizualizovat textová (RSS, zprav. kanály, ...) a obrazová (mapy, grafy, ...) data.

1) Proveďte analýzu API SAGE2, stávajících aplikací, možnosti zobrazení různých dat a možnosti modulového systému jedné zastřešující aplikace kontrolující jednotlivé zobrazovací widgety.

2) Stanovte obecná i konkrétní omezení a doporučení pro budoucí vyvíjená ovládací rozhraní (vzhled, ovládání, ...).

3) Navrhněte:
- způsob realizace jednotlivých aplikací včetně jejich organizace, komunikace, konfigurace a ovládání,
- způsob získávání dat z externích zdrojů,
- minimálně 5 konkrétních aplikací (např. zprávy, lokální a globální počasí, hodiny, ...).

4) Implementujte alespoň 3 z navržených aplikací

5) Vytvořené aplikace vhodně otestujte.

## Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 28. ledna 2017

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Bakalářská práce

# 10K SAGE2 Dashboard: Sada widgetů pro velkoplošnou obrazovku

***Radek Meduna***

Vedoucí práce: Ing. Radek Richtr

5. ledna 2018

# Poděkování

V první řadě chci poděkovat vedoucímu mé bakalářské prace Ing. Radku Richterovi za trpělivost a ochotu při našich pravidelných konzultačních schůzkách. Dále posílám tisíceré díky svým rodičům, kteří mě vždy velice podporovali. V poslední řade děkuji své sestře Petře za provedení velice přínosné kontroly pravopisu a překlepů, kterých bylo v práci nespočet.

# Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 5. ledna 2018 .....

### Odkaz na tuto práci

Meduna, Radek. *10K SAGE2 Dashboard: Sada widgetů pro velkoplošnou obrazovku*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

# Abstrakt

Tato bakalářská práce se věnuje analýze, návrhu a následné implementaci widgetů na platformě SAGE2. Cílem je stanovení konkrétních doporučení pro tvorbu aplikací pro SAGE2. Práce se zaměřuje zejména na kvalitní návrh a vysokou konfigurovatelnost widgetů.

Hlavním přínosem práce je stanovení základních doporučení pro návrh a implementaci zastřešující aplikace pro sadu widgetů a jejich vzájemná komunikace mezi sebou.

**Klíčová slova** SAGE2, SAGElab, widget, node.js, Flickr, počasí, zpravodajství

# Abstract

This bachelor thesis deals with analysis, design and subsequent implementation of widgets on the SAGE2 platform. The goal is to set specific recommendations for creating applications for SAGE2. The work focuses on quality design and high widget configurability.

The main contribution of the thesis is to establish basic recommendations for the design and implementation of parent application for widgets and their mutual communication with each other.

**Keywords** SAGE, SAGElab, widget, node,js, Flickr, weather, news

# Obsah

**Úvod** **1**

**1 Cíl práce** **3**

**2 Struktura práce** **5**

**3 Analýza** **7**
3.1 SAGE2 . . . . . 7
3.2 SAGE2 api . . . . . 11
3.3 Analýza existujících aplikací . . . . . 12

**4 Návrh** **19**
4.1 Nefunkční požadavky . . . . . 19
4.2 Funkční požadavky . . . . . 20
4.3 Zastřešující aplikace . . . . . 21
4.4 Mapa měst . . . . . 22
4.5 Počasí pro konkrétní oblast . . . . . 23
4.6 Novinky . . . . . 25
4.7 Projektor obrázků z Flickru . . . . . 29
4.8 Populace města . . . . . 31

**5 Realizace** **35**
5.1 Zastřešující aplikace . . . . . 35
5.2 Počasí pro konkrétní oblast . . . . . 39
5.3 Novinky . . . . . 40
5.4 Projektor obrázků z Flickru . . . . . 42

**6 Testování** **47**
6.1 Testování během vývoje . . . . . 47
6.2 Uživatelské testování použitelnosti . . . . . 47

6.3 Heuristická analýza . . . . . 49

**Závěr** **53**
Možná vylepšení . . . . . 53
Splnění zadání . . . . . 53

**Literatura** **55**

**A Seznam použitých zkratek** **57**

**B Obsah přiloženého CD** **59**

# Seznam obrázků


3.1 Administrační panel . . . . . 8
3.2 SAGE2 architektura . . . . . 8
3.3 Widget bar . . . . . 10
3.4 US Wetaher widget . . . . . 13
3.5 Photo slideshow . . . . . 14
3.6 Digitální hodiny . . . . . 15
3.7 Insta feed . . . . . 17
3.8 Generátor grafů . . . . . 18

4.1 Google maps SAGE UI . . . . . 21
4.2 Počasí pro konkrétní oblast . . . . . 24
4.3 Novinky - seznam článků . . . . . 27
4.4 Novinky - detail . . . . . 28
4.5 Projketor obrázků ze soc. sítě Flickr . . . . . 30
4.6 Object-fit:cover . . . . . 31
4.7 Populace města . . . . . 33

5.1 Sekvenční diagram vykreslení widgetů . . . . . 37
5.2 Widget bar - zastřešující aplikace . . . . . 38
5.3 SAGE UI . . . . . 38
5.4 Chybové hlášky . . . . . 39
5.5 Widget počasí . . . . . 40
5.6 Widget novinky . . . . . 41
5.7 Widget novinky - detail . . . . . 42
5.8 Widget projektor . . . . . 43
5.9 Projektor - 1 dominantní fotografie . . . . . 45

# Úvod

Tématem této bakalářské práce je provedení analýzy stávajících SAGE2 aplikací, prozkoumání možností zobrazení textových i obrazových dat a možnosti modulového systému jedné zastřešující aplikace kontrolující dashboard[1].

Hlavním úkolem dashboardu je integrace jednotlivých různorodých widgetů. Každý jednotlivý widget je samostatný modul, který je delegován zastřešující aplikací. Výhodou takové zastřešující aplikace je zapouzdření logiky jednotlivých widgetů do konkrétního modulu. Aplikace se v důsledku takového přístupu mnohem lépe udržuje a rozšiřuje.

„Widget je vizuální interaktivní prvek, sloužící k otevírání a ovládání různých programů a programových skupin“[1]. Widget je program, který stále běží a podává uživateli stálý přísun specifických informací, pro které byl vytvořen. Dále může být widget koncipován jako zjednodušený ovládací prvek (například operačního systému). Widgety jsou v dnešní době velice hojně rozšířeny na mobilních telefonech (zejména na operačním systému *Android*) a osobních počítačích. Například již dříve zmíněný *Android* podporuje vývoj widgetů od dubna r. 2009[2]. Mezi typické zástupce widgetů (ať už ze světa mobilů, nebo počítačů) lze zařadit widget zobrazující data o aktuálním počasí, widget s poznámkami nebo kalendář.

Při návrhu je nutné zohlednit fakt, že výsledné widgety budou spouštěny a zobrazovány na velkoplošných obrazovkách[2]. Dalším faktorem, který je nutno brát v potaz, je specifické ovládání telestěny. Telestěna je často ovládána pouze myší (ukazovátkem). Je tedy nutno zajistit rozumné ovládání aplikace např. bez klávesnice.

SAGE (Scalable Amplified Group Environment) je velkoplošné zobrazovací virtualizační zařízení tvořené více dispeji, řízené softwarem SAGE2 postaveným na webových technologiích. Je vhodné zejména pro zobrazovaní velkého množství dat na obrazovkách tvořených více displeji a pro kooperaci více uži-

[1] v češtině přístrojová deska, nicméně nadále bude v práci použit výraz dashboard

[2] pro představu, laboratoř sageLab je tvořena dvaceti fullHD monitory

vatelů.

Rešeršní část se věnuje analýze již existujících widgetů. Pozornost je nejprve věnována samotné aplikaci, jež spadá do kategorie widgetů, které jsou dodávány společně s distribucí SAGE2. Následně jsou analýze podrobeny widgety kolegy Aleha Kuchynskeho, který je navrhl a implementoval v r. 2017 jako součást své bakalářské práce.

Navrhnuté a následně implementované aplikace se zabývají zobrazením dat souvisejících s počasím, zpravodajstvím a sociální sítí Flickr. Implementované widgety jsou následně podrobeny vhodnému testování.

Při návrhu a implementaci byl kladen důraz na co největší konfigurovatelnost a co nejsnažší rozšiřitelnost, ať už ze strany autora textu, nebo v budoucnu od jiných kolegů. Naopak bylo upuštěno od snahy o dokonalý vzhled aplikací.

Na závěr je uvedena řadu konkrétních doporučení a zkušeností týkajících se vývoje aplikací pro SAGE2.

Kapitola 1

# Cíl práce

Cílem práce je vytvoření jedné zastřešující aplikace, skládající se ze sady jednotlivých widgetů postavených na webových technologiích pro platformu SAGE2. Widgety budou schopny získávat a vizualizovat textová a obrazová data. Tyto widgety musí být co nejlépe konfigurovatelné a snadno rozšiřitelné. Konkrétně se tato práce bude zaměřovat na následující body:

1. **analýza API SAGE2, analýza stávajících aplikací, možnost modulového systému jedné zastřešující aplikace.**

   Ke splnění výše zmíněného bodu je třeba důkladně prostudovat uživatelskou a vývojářskou dokumentaci SAGE2 a pečlivě analyzovat již existující aplikace pro tuto platformu. Velký důraz bude kladen na důkladnou analýzu aplikací, které jsou svým charakterem podobné widgetům, které budou implementovýny jako součást této práce.

2. **Stanovení obecných i konkrétních omezení a doporučení pro budoucí vyvíjená ovládací rozhraní.**

   Stanovená doporučení a omezení se bude zakládat zejména na analýze SAGE2 API a na analýze již existujících aplikací pro tuto platformu.

3. **Návrh minimálně pěti konkrétních aplikací, jejich způsob realizace (organizace, komunikace, ovládání, konfigurace) a popis způsobu získávání dat z externích zdrojů.**

   Návrh widgetů bude reflektovat poznatky získané při analýze již existujících aplikací a požadavky, které byly stanoveny hned na začátku práce (vysoká konfigurovatelnost a rozšiřitelnost). V návrhu dojde na stanovení funkčních a nefunkčních požadavků na widgety. U každého jednotlivého widgetu bude uveden způsob získání dat nutných pro jeho chod a popis případných podmínek zacházení s těmito daty (určených poskytovatelem). Při návrhu bude zohledněn fakt, že aplikace jsou určeny pro velkoplošnou telestěnu, která má své specifické charakteristiky (ovládání, velké rozlišení atd.).

4. **Implementace alespoň tří z navržených aplikací.**

   Z výše navhrnutých widgetů budou vybrány tři, které budou implementovány. U každého jednotlivého widgetu budou popsány problémy, které se při implementaci objevily a jejich případné řešení.

5. **Testování implementovaných widgetů.**

   Implementované widgety budou vhodně otestovány. Případné nedostatky a návrhy na zlepšení, které vyjdou z testování budou uvedeny a popsány.

Kapitola 2

# Struktura práce

V kapitole Analýza3 jsou popsány hlavní charakteristiky platformy SAGE2 se zaměřením na API, které SAGE2 poskytuje vývojářům. Na konci kapitoly jsou analyzovány některé existující aplikace pro tuto platformu a to včetně jejich výhod, nevýhod a návrhů na jejich zlepšení.

V další kapitole Návrh4 jsou reflektovány výsledky kapitoly Analýza3 a na jejich základě je zde prezentován návrh pěti možných widgetů a to včetně funkčních a nefunčních požadavků, rozboru požadavků na ovládání, API, a základních grafických návrhů. Zvláštní pozornost je pak věnována návrhu zastřešující aplikace, která dané widgety integruje do jednoho funkčního celku.

V sekci Realizace5 je podrobně popsáno zvolené řešení implementace jednotlivých widgetů a jejich zastřešující aplikace. Vybrané postupy jsou zde logicky odůvodněny.

V kapitole Testování6 jsou uvedeny typy testů, které byly provedeny. Je zde uveden důvod výběru konkrétních testů a popsán přínos, který mohou dané testy poskytnout. Na závěr kapitoly dojde k popisu výstupů testů.

V Závěru6.3.2 práce je zhodnocena bakalářská práce. Jsou zde popsány hlavní přínosy práce. V této kapitole jsou popsány cíle práce, které se podařilo splnit.

Kapitola 3

# Analýza

V analytické části bakalářské práce je popsána platforma SAGE2, její základní vlastnosti a omezení vztahující se k tématu práce. Zvláštní pozornost je věnována specifickým částem SAGE2 API. Na závěr jsou analyzovány jak oficiálně distribuované widgety, tak widgety vytvořené v rámci bakalářské práce jiným studentem. Dále bude zmíněno SAGE2 API a na závěr kapitoly dojde k analýze existujících widgetů pro SAGE2.

## 3.1 SAGE2

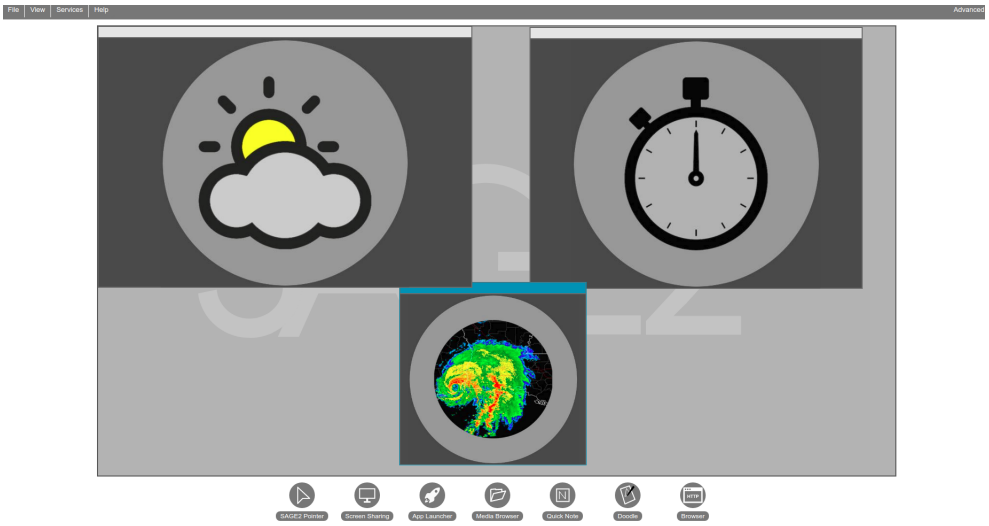
SAGE2 navazuje na původní SAGE[3] software, který byl vyvinut v roce 2004. Původní SAGE byla přijata více než stovkou mezinárodních internetových stránek a byla navržena pro zobrazení obsahu na velkých displejích s vysokým rozlišením. Dalším nesporným benefitem, který SAGE2 platforma nabízí, je usnadnění spolupráce mezi členy týmu[3].

SAGE2 je nástupcem SAGE a přichází s novým návrhem, který je založený na využití cloudu a webového prostředí. Narozdíl od svého předchůdce, je založen na technologii node.js, což je javascriptový softwarový systém používaný, pro javascript netradičně, na straně serveru. K serveru je možné se připojit pomocí webového rozhraní, které obsahuje i administrační panel (obr. 3.1). Schéma SAGE2 architektury je k vidění na obrázku 3.2.

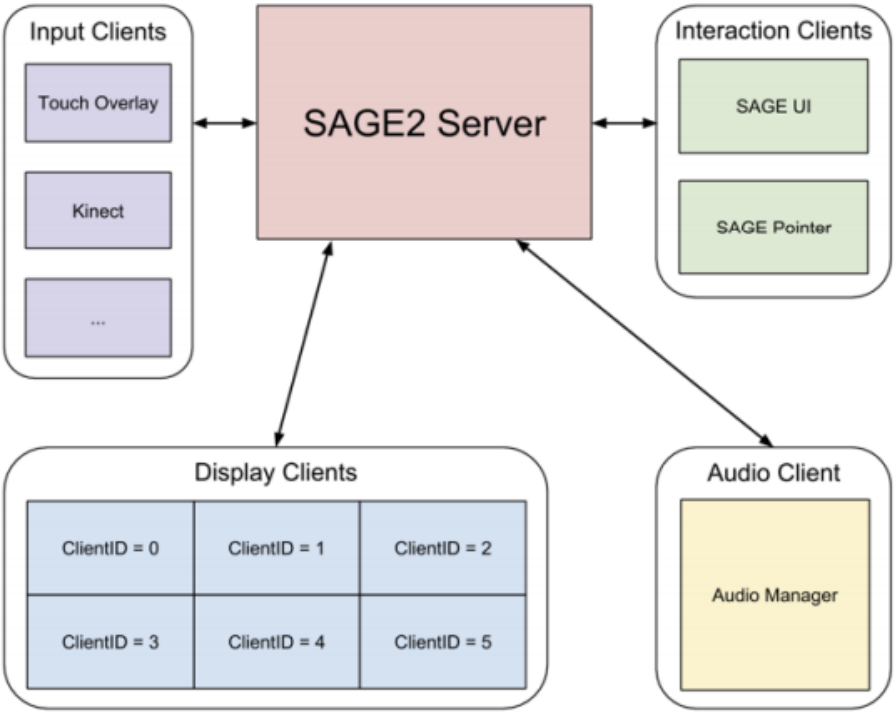
Dle uživatelské dokumentace[4] se SAGE2 skládá ze čtyř hlavních částí:

- server,
- display client,
- audio manager,
- web interface.

[3] https://www.sage.com/




Obrázek 3.1: Administrační panel




Obrázek 3.2: SAGE2 architektura[5]

SAGE2 server je psaný v jazice javascript a jeho hlavním úkolem je delegace zobrazovacích klientů. Reaguje na požadavky od webového rozhraní SAGE2 UI a deleguje příslušné klienty požadovanými akcemi (např. spuštění konkrétní aplikace).

### 3.1.1 Ovládání

K SAGE2 aplikacím lze přistupovat hned několika způsoby, jedinou podmínkou je nainstalovaný webový prohlížeč. Logicky je tedy možné se k SAGE2 připojit pomocí osobního počítače, tabletu nebo mobilního telefonu. Z důvodu široké škály zařízení, které se mohou připojit k SAGE2, je důležité řešit ovládání. Jedním ze zákadních ovládacích prvků je SAGE2 pointer. Pointer má dva módy (window managment a application interaction). Mezi uvedenými módy lze přepínat. Pointer se využívá pro interakci s běžícími aplikacemi v klientech. Pomocí SAGE2 pointeru je možné vyvolat widget bar (obr. 3.3), díky kterému uživatel může s aplikací interagovat.

Se SAGE2 lze interagovat hned několika způsoby. Jednou z možností je ovládací rozhraní, kde lze dané aplikace spouštět a pokud to implementace umožňuje, lze v této části zadávat např. uživatelské vstupy. Další možností je ovládání pomocí klávesnice, nicméně v praxi uživatel často nemá klávesnici k dispozici a právě toto omezení alespoň částečně řeší widget bar.

Další nástrahou pro uživatele je značná nepřesnost bezdotykových ovládacích zařízení. Pří návrhu je nutné tento fakt zohlednit v podobě dostatečně velkých ovládacích prvků.

Jak již bylo zmíněno výše, uživatelé často ovládají SAGE2 bez klávesnice. V případě, že aplikace vyžaduje textový uživatelský vstup, je vhodné např. ve widget baru uživateli nabídnout sadu již předpřipravených, často používaných voleb, které lze vybrat pouhým kliknutím.

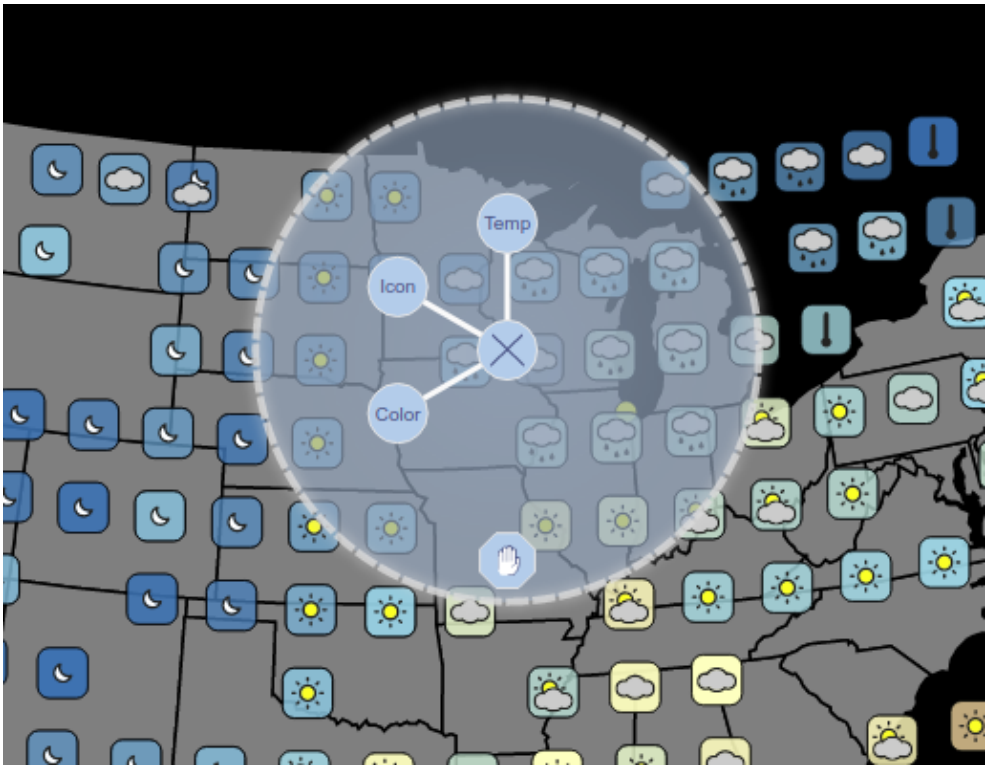
Další doporučení se týká opět aplikací, které vyžadují textový uživatelský vstup. Pole pro zadání textu je vhodné umístit nejen do widget baru, ale také do ovládacího rozhraní. Výhodou umístění vstupu do těchto dvou míst, je možnost zadání uživatelského vstupu bez nutnosti přepnutí do pointer režimu.

### 3.1.2 Elementy uživatelského prostředí

Do aplikace lze přidávat 3 různé elementy, jimiž jsou:

- button (tlačítko),
- slider (posuvník),
- text input (textový vstup).

Tyto elementy lze najít ve widget baru. Pomocí těchto elementů může uživatel interagovat s aplikací.




Obrázek 3.3: Widget bar v aplikaci US Weather

### 3.1.3 Typy aplikací

Dle vývojářské příručky[5], aplikace psané pro SAGE2 můžou být různých typů.

- DOM,
  - pro čistě HTML aplikace
  - vytváří elementy pomocí javascriptového DOM api
- Canvas,
  - pro aplikace využívající 2D grafiku
- SVG,
  - vhodná pro většinu aplikací pro SAGE2
  - škálovatená grafika se přizpůsobí obřímu rozlišení, kterého je možné dosáhnout

- D3,
  - pro aplikace vykreslující rúzne grafické prvky, jakou jsou např. grafy
  - využívá javascriptovou knihovni d3.js
- WebGL,
  - pro aplikace využívající 3D grafiku
- three.js.
  - pro aplikace využívající 3D grafiku
  - využívá WebGL

## 3.2 SAGE2 api

Dle vývojářské dokumentace[5] je každá aplikace JavaScriptovým objektem. Součástí aplikace musí být následující části:

- zdrojové kódy v jazyce JavaScript,
- soubor instruction.json – konfigurační soubor, kde jsou uvedeny základní údaje o aplikaci (výška, šířka, titulek atd.),
- logo aplikace v jednom z formátů: webp, png, jpg.

Jistým hendikepem při vytváření nové SAGE2 aplikace je nepříliš obsáhlá vývojářská dokumentace. Pro dobré pochopení používání SAGE2 API je nejlepší podívat se detailně na jednu z aplikací, které jsou součástí SAGE2 distribuce.

### 3.2.1 SAGE2_App

Každá nová aplikace musí dědit od třídy SAGE2_App a implementovat její vybrané metody. V následujícím výčtu jsou uvedeny všechny metody, jež jsou podstatné pro vývoj nových aplikací. Podrobnější výčet lze nalézt například ve vývojářské dokumentaci[5].

- Init - inicializuje uživatelské nastavení. V této metodě musí programátor definovat všechny ovladače a tlačítka, která budou používa v budoucnosti.
- Draw - tato metoda je volána za účelem překreslení aplikace.
- Resize - metoda je volána v případě změny rozměrů okna widgetu. Sem patří veškerý kód, který souvisí právě se změnou velikosti a tvaru okna.

- Event - metoda je volána při spuštění různých událostí (stisknutí určitého tlačítka na klávesnici, kliknutí myší atd.). Tyto události jsou následně v metodě event zpracovány. Zde programátor rozhodne, jakým způsobem bude aplikace reagovat na události vyvolané uživatelem.

## 3.3 Analýza existujících aplikací

V této kapitole je uvedena analýza již existujících aplikací. Analýze jsou podrobeny aplikace z oficiální distribuce SAGE2 a aplikace kolegy Aleha Kuchynského, který widgety vypracoval jako součást své bakalářské práce.

Widgety kolegy Kuchynského nebudou hodnoceny z hlediska grafického, jelikož to nebylo cílem práce. Hlavní pozornost je věnována analýze funkcionalit jednotlivých widgetů a následnému návrhu drobných úprav, které by vedly k zlepšení uživatelského pocitu z widgetu.

### 3.3.1 US Weather

Aplikace US Weather (náhled aplikace obr. 3.4) je jednou z aplikací, kterou uživatel získá při stáhnutí SAGE2. Aplikace se nachází v oficiálním repozitáři SAGE2 aplikací. Je to tedy jedna ze vzorových aplikací, kde se mohou začínající vývojáři inspirovat a získat přehled o zvyklostech implementace aplikací pro SAGE2. Tato aplikaci bude podrobena celkové analýze a následně budou navhrnuta případná vylepšení.

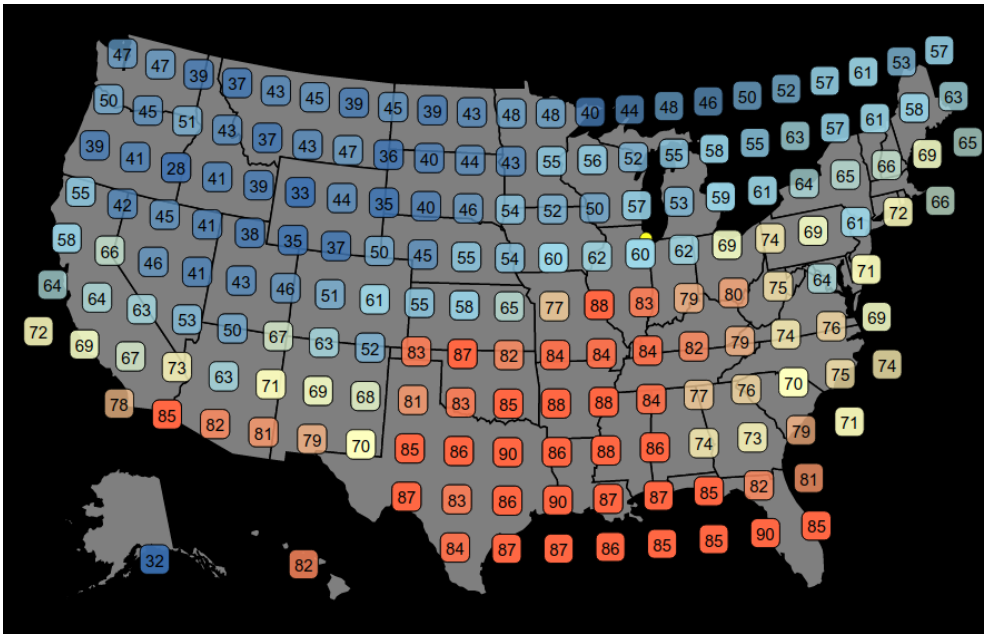
Aplikace US Weather (jak již samotný název napovídá) slouží k zobrazení počasí pro území Spojených státu amerických. Díky aplikaci se dozvíme aktuální teplotu vyjádřenou číselnou hodnotou a charakter počasí díky ikonce. Nicméně, najednou lze zobrazovat pouze jednu z těchto dvou informací. Mezi módy lze přepínat ve widget baru.

**Volání api.** K získávání dat o počasí widget využívá Yahoo Weather API[4]. V aplikaci je definováno pole zeměpisných šířek a délek. U každého elementu z tohoto pole jsou na mapě uvedena příslušná data o počasí. Pro každý element pole je volán endpoint Yahoo Weather API. Příklad voláné url:

```
https://query.yahooapis.com/v1/public/yql?
q=select temp_f, weather, icons from
wunderground.currentobservation where location='39.700,-75.400';
&format=json&env=store://datatables.org/alltableswithkeys&callback=
```

Nicméně toto řešení není ideální. Yahoo Weather API umožňuje poslat dávkový požadavek. Jistě by tedy bylo lepší zvolit algoritmus, který volá API pouze jednou a odoveď API by obsahovala všechna požadovaná data. Taková

[4] https://developer.yahoo.com/weather/



Obrázek 3.4: Náhled aplikace US Weather

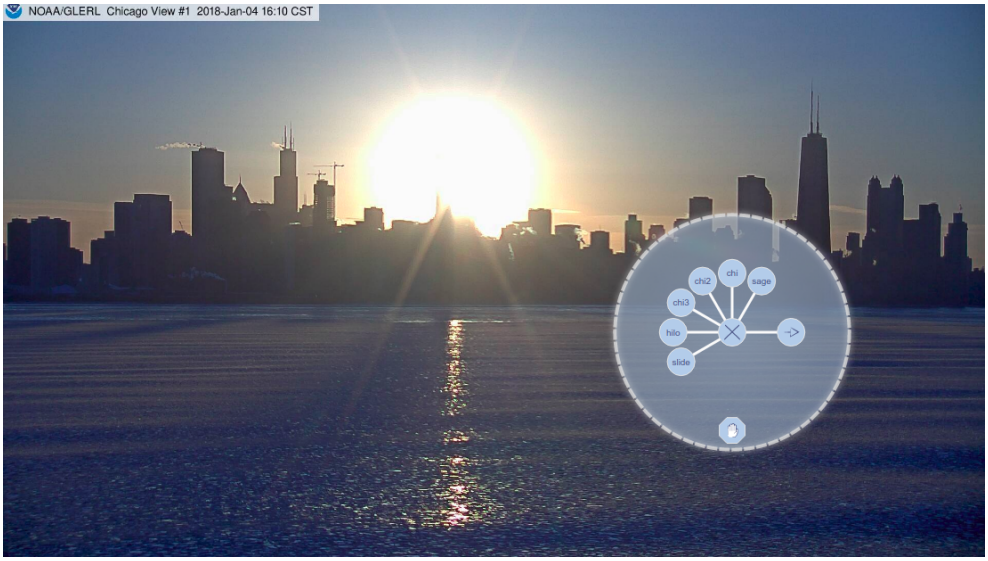
url by vypadala obdobně (podstatná část url je v obou příkladech zvýrazněna podtržením):

```
https://query.yahooapis.com/v1/public/yql?
q=select temp_f, weather, icons from wunderground.currentobservation where
location in ('39.700,-75.400', '42.361,-71.057', ...);
&format=json&env=store://datatables.org/alltableswithkeys&callback=
```

Při současné implementaci se api volá 179x při každé aktualizaci dat. Výše navrhnuté řešení by tedy ušetřilo 178 volání api, značně by zvýšilo rychlost vykreslení dat a snížilo potřebný datový tok.

**Kešování dat.** Data týkající se počasí se získávají až několikrát za sekundu. I přes značnou proměnlivost počasí je toto řešení velice plýtvavé. Jednou z možností by bylo získaná data kešovat a aktuální data získávat v delších intervalech. Rozumným kompromisem se zdá být volání api jednou za 10 minut. Vůbec nejlepším řešením by bylo umožnění nastavení intervalu v konfiguračním souboru.

**Módy.** Tato poslední poznámka k aplikaci je tou nejméně důležitou, jedná se pouze o drobnou výtku. Aplikace obsahuje 3 módy. Lze zobrazovat teplotu, charakter počasí nebo pouze barevné pozadí, které souvisí s teplotou.




Obrázek 3.5: Náhled aplikace Photo slideshow

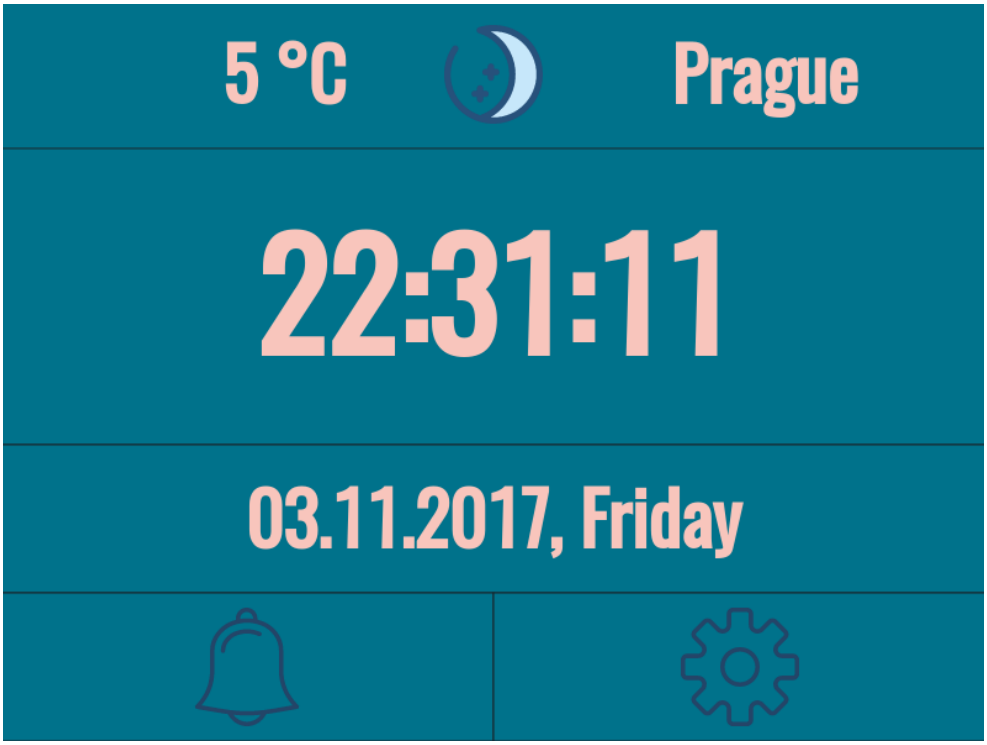
Samostatný mód pro zobrazení teploty barevným pozadím se zdá být naprosto zbytečný, jelikož se barva zobrazuje i v pozadí ostatních dvou módů. Tento mód je tedy redundantní a nepřináší uživateli žádnou přidanou hodnotu oproti dalším dvou módům.

### 3.3.2 Projektor obrázků ze SAGE distribuce

Základní distribuce obsahuje aplikaci *Photo slideshow* (k vidění na obrázku 3.5). Aplikace zobrazuje předpřipravenou sekvenci fotek. Uživatel má dále možnost nahrát sadu fotografií do uživatelské složky na SAGE2 server a aplikace bude zobrazovat tuto konkrétní sekvenci. Aplikace sama o sobě má velice chudé možnosti a nic dalšího už nenabízí.

### 3.3.3 Projektor obrázků od Jonatana Matějky

Další widget, který zobrazuje fotografie vytvořil Jonatan Matějka jako semestrální úlohu pro předmět *Programování grafických aplikací* na ČVUT FIT v roce 2017. Projektor umožňuje přímo v aplikaci procházet složky a zobrazovat jejich obsah v projektoru. Možnost průchodu složkami je velice povedená a určitě je k uživateli přívětivější, než nutnost nahrát svůj obsah do jedné konkrétní předpřipravené složky (jako je tomu u aplikace *Photo slideshow*). Jinak obě výše zmíněné aplikace fungují velice podobně.




Obrázek 3.6: Náhled aplikace Digitální hodiny

### 3.3.4 Digitální hodiny

Kolega Aleh Kuchynski implementoval digitální hodiny s rozšířenou funkcionalitou (k vidění na obrázku 3.6). Hodiny zobrazují teplotu v zadaném městě pomocí číslené hodnoty a stav počasí pomocí příslušné ikonky. Na hodinách lze nastavit budík, který v zadaném čase signalizuje vypršení nastaveného času. Digitální hodiny lze snadno konfigurovat v příslušném konfiguračním souboru. Uživatel smí vybrat město, pro které chce zobrazovat počasí, formát času a časové pásmo.

**Paměť uživatelské volby.** Prvním navrhovaným zlepšením je paměť uživatelské volby v nastavení. Když jednou uživatel v nastavení vybere, že chce zobrazovat počasí např. pro Chrudim a ve dvanácti hodinovém časovém formátu, je velice pravděpodobné, že toto chování bude očekávat a vyžadovat i při dalším spuštění aplikace. Momentálně widget funguje tak, že při každém spuštění načte statická data z konfiguračního souboru a při změně nastavení se tato data nikam nepropíší.

**Nastavení alarmu.** Další drobnost, která by mohla vylepšit použitelnost aplikace je lepší propracování nastavení času budíku. V budíku lze nastavit alarm pouze na probíhající den. Tato implementace je značně omezující. Například uživatel, který si chce nastavit budík 10 minut před půlnocí má značně limitující výběr validních časů pro budík. Takový uživatel má možnost nastavit budík pouze v intervalu současnost - půlnoc. Možným řešením je přidat v nastavení alramu uživatelský vstup pro datum. Další možností, která nevyžaduje zadávání data, je povolení nastavení budíku na následujících 24 hodin.

### 3.3.5 InstaFeed

Widget InstaFeed zobrazuje data z populární sociální sítě Instagram, kde spolu uživatelé sdílí své fotografie (k vidění na obrázku 3.7). Tento konrétní widget umožňuje procházet fotkami posuvníkem, komentovat fotky a v neposlední řadě fotky lajkovat. Další velice pěknou funkcí je možnost zobrazení bodu na mapě, kde byla daná fotografie pořízena. Jedním z omezujících faktorů je fakt, že aplikace běži v sandbox módu[5]. Nicméně tento fakt nemá přímou spojitost s implementací a nebude tedy detailně rozebírán.

**Lajkování.** Na widgetu by se určitě dala vylepšit funkce lajkování. Při standardním klinutí na srdíčko dojde ke zvýšení počtu o jedno, při dalším kliknutí se dané srdíčko odebere. Nicméně při zběsilém klikání se počet srdíček může zvýšit o více než 1, což je zajisté neočekávané chování.

**Posuvník fotek.** Dalším matoucím prvkem v aplikaci jsou posuvníky fotek. Když má uživatel zobrazenou první fotografii, logicky nemůže zobrazit žádnou přechozí a odpovídá tomu i skrytí posuvníku, který odkazuje na předchozí fotografii. Nicméně v případě, že uživatel má zobrazenou poslední možnou fotografii, posuvník na další fotografii je stále viditelný a uživatel je tedy zmaten a neví, zdali se aplikace zasekla, nebo opravdu už žádné další fotografie nelze zobrazit.

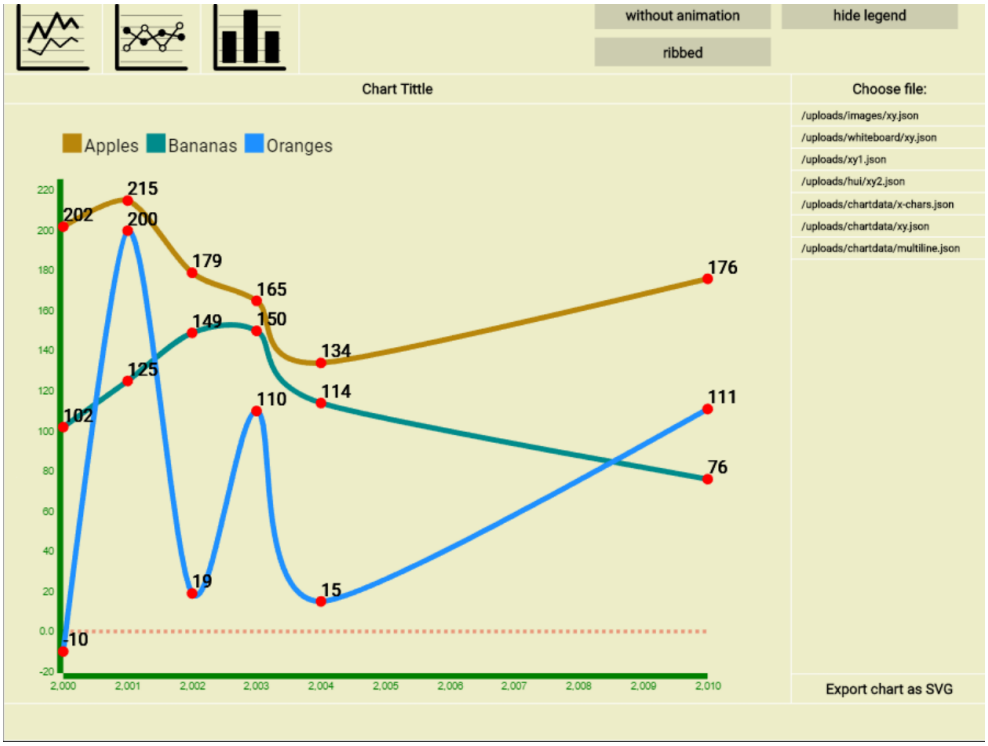
**Kešování.** Autor zvolil výborné řešení s kešováním dat z instagramu, není tedy nutné neustále posílat požadavky a dojde tak k ušetření výkonu a ke zrychlení běhu aplikace.

### 3.3.6 Generátor grafů

Widget generátor grafů vizualizuje data, která jsou do aplikace naimportována soubory typu JSON a XML (k vidění na obrázku 3.8). Widget zobrazuje data ve formátu sloupcového, jednočárového a vícečárového grafu a umožňuje mezi těmito formáty volně přecházet. Aplikace podporuje možnost vizualizace dat

[5]tedy ve zkušebním módu, kde je řada funkcionalit omezena


Obrázek 3.7: Náhled aplikace InstaFeed




Obrázek 3.8: Náhled aplikace Generátor grafů

pomocí animací, vytvoření os grafu, nastavení formátu a rozměru popisků nebo export výsledného grafu.

**Přepínání zobrazovacích módů.** Prvním drobným vylepšením by mohlo být přepínání mezi typy grafů pomocí widget baru (menu vyvolané v režimu SAGE2 pointer kliknutím pravým tlačítkem na aplikaci). Toto je standardní a elegantní způsob přepínání mezi různými režimy v již existujících aplikacích pro SAGE2.

**Import dat.** Dalším možným vylepšením je podpora dalších typů souborů pro import dat. Příkladem může být formát souborů typu YAML. Dále by bylo určitě dobré zpříjemnit proces nahrání souboru na SAGE2 server. V ideálním případě by uživatel mohl nahrát soubor na server přímo v aplikaci pomocí SAGE2 pointeru a uživatelského vstupu, kde by vybral požadovaný soubor.

**Další typy grafů.** V neposlední řadě je možné do aplikace přidat podporu více typů grafů, které vizualizují příslušná data. Mohlo by se jednat např. o výsečový nebo plošný graf.

Kapitola 4

# Návrh

Tato kapitola pokrývá návrh jedné zastřešující aplikace, jejíž součástí budou jednotlivé widgety. Nejprve dojde ke stanovení funkčních a nefukčních požadavků na aplikaci. Podrobně zde bude popsán i návrh jednotlivých widgetů, budou uvedeny způsoby získávání dat z externích zdrojů a stanovena obecná i konkretní doporučení pro vývoj aplikace pro SAGE2 prostředí.

## 4.1 Nefunkční požadavky

Řada nefunkčních požadavků logicky vyplývá již z faktu, že aplikace je psaná pro SAGE2 prostředí. Při návrhu byly zohledněny zejména následující nefunkční požadavky.

- **Aplikace bude psána v javascriptu**. Aplikace bude psaná pro SAGE2 prostředí. Z tohoto faktu nutně vyplývá, že aplikace bude psaná v javascriptu.

- **Optimalizace pro SAGElab**. *„Laboratoř SAGElab je společným pracovištěm sdružení CESNET, Fakulty informačních technologií ČVUT a Fakulty elektrotechnické ČVUT. Účelem laboratoře je podpora výuky předmětů a výchovy odborníků v oblastech síťových technologií, multimédií a virtuální reality a vytvoření prostředí pro výzkum a vývoj nových síťových aplikací. Laboratoř je využívána pro přednášky odborníků z řady oblastí výzkumu a vývoje, kde je potřeba pracovat s vizualizacemi rozsáhlých dat, pro distanční přednášky, pro samostatnou práci diplomantů, doktorandů a zaměstnanců na výzkumu a vývoji v oblasti síťových a multimediálních technologií a pro spolupráci s externími pracovišti v ČR i v rámci mezinárodní komunity laboratoří s obdobným zaměřením.“*[6] Aplikace bude spustitelná a použitelná pro libovolné SAGE2 zařízení, nicméně bude vyvíjena a testována v SAGElabu, bude tedy optimalizovaná přímo pro toto pracoviště, kde se nachází telestěna o rozlišení 10K.

Tato optimalizace bude spočívat ve vlastním konfiguračním souboru, kde budou uvedeny hodnoty ideální pro SAGElab.

- **Dokumentace kódu**. Zdrojový kód bude velmi dobře dokumentovaný. Každá metoda bude obsahovat stručný popis vlastní funkcionality, popis vstupních parametrů a popis případné návratové hodnoty. Dokumentaci ocení zejména ti, kteří se rozhodnou prací inspirovat, nebo dokonce se rozhodnou v této práci pokračovat. Dokumentace bude poskytnuta dle standardů JSDocu[6].

- **Vysoká konfigurovatelnost**. Výsledná aplikace bude co nejlépe konfigurovatelná. V konfiguračním souboru bude možné nastavit hodnoty jako jsou barva textu, barva pozadí, odsazení různých elementů od sebe, velikost textu nebo například počet zobrazovaných elementů (kupř. počet dní, pro které se bude zobrazovat počasí).

- **Snadná rozšiřitelnost**. Aplikace bude snadno rozšiřitelná, v práci bude možné snadno pokračovat např. jinými studenty. Velký prostor pro vylepšení bude zejména v grafické úpravě vzhledu aplikace.

## 4.2 Funkční požadavky

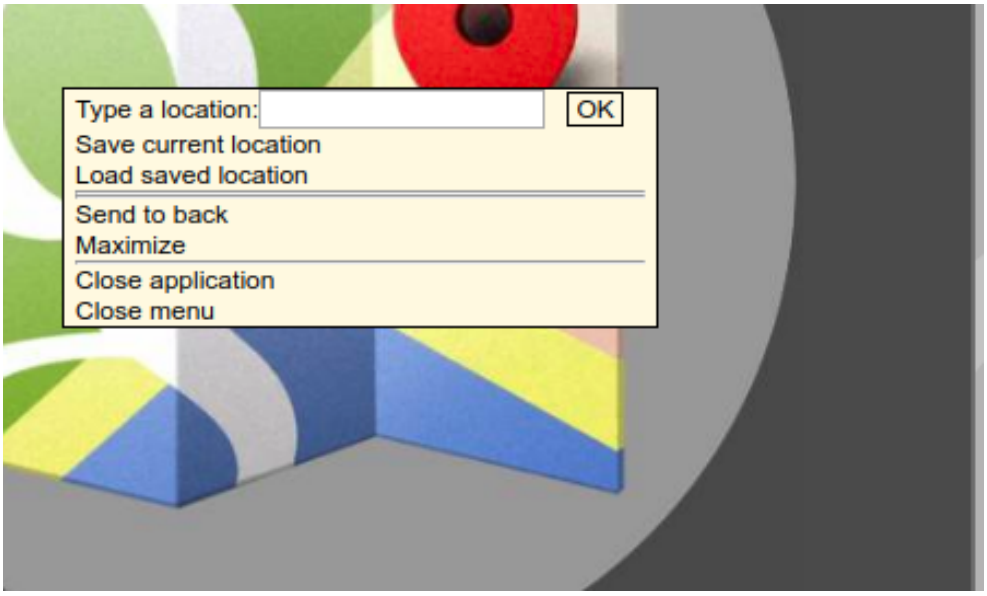
Z analýzy existujících aplikací a SAGE2 platformy vyplynuly následující funkční požadavky.

- **Možnost změny volby za běhu aplikace**. Uživatel bude mít možnost měnit volby za běhu aplikace. Tyto volby půjdou měnit zejména ve widget baru (obr. 3.3). Takové chování je pro platformu SAGE2 standarndní. Zmíněnými volbami je myšleno např. změnu města, pro které se zobrazuje počasí. U voleb, kde to bude mít smysl, bude mít možnost uživatel měnit jejich hodnotu i v SAGE UI (obr. 4.1).

- **Paměť uživatelských voleb**. Důležité uživatelské volby se budou ukládat a při příštím spuštění aplikace dojde k jejich načtení. Například když uživatel za běhu aplikace nastaví Chrudim jako město, pro které se aktuálně bude zobrazovat počasí. Po vypnutí a zapnutí aplikace se bude počasí opět zobrazovat pro Chrudim.

- **Aktualizace dat**. Zobrazovaná data bude možné aktualizovat manuálně stisknutím příslušného tlačítka. Další možností aktualizace dat bude uvedení počtu aktualizací za hodinu v konfiguračním souboru. Tyto aktualizace se budou provádět automaticky.

[6] https://github.com/jsdoc3/jsdoc




Obrázek 4.1: SAGE UI aplikace Google maps

## 4.3 Zastřešující aplikace

Zastřešující aplikace nad konkrétními widgety bude řídit zejména distribuci dat mezi danými widgety. Zastřešující aplikace např. načte globální konfigurační soubor a data poskytne všem widgetům.

**Synchronizace dat.** Zastřešující aplikace bude rovněž zpracovávat samotné uživatelské vstupy a akce. Aplikace v případě uživatelského vstupu upozorní všechny své widgety na nutnost aktualizace dat podle aktuální hodnoty (např. v případě změny města se musí všechny widgety překreslit, aby zobrazovaná data byla relevantní). Jednotlivé widgety, které jsou delegovány zastřešující aplikací tedy nebudou zobrazovat data pro různá města, protože vždy v případě změny dojde k aktualizaci proměnné ve všech widgetech.

Dalším možným řešením, které bylo vzato v potaz a promyšleno, bylo takové, že všechny widgety budou mít svou vlastní konfiguraci (proměnné budou lokální). Nicméně po důkladném zvážení bylo zvoleno výše uvedené řešení s globálními proměnnými, hlavně z důvodu pohodlí uživatele. Je předpokládáno, že v případě, že uživatel změní město v jednom widgetu, v drtivé většině případů si přeje, aby se i ostatní widgety přizpůsobily změně.

V neposlední řadě se zastřešující aplikace bude starat o pravidelné překreslení widgetů v časových intervalech (např. data o počasí po čase ztratí svou informační hodnotu i bez nutnosti změny příslušné lokality).

Jak již bylo zmíněno, zastřešující aplikace bude spouštět všechny ostatní widgety a poskytovat jim uživatelskou konfiguraci. Aby se konfigurace promítla v aplikaci, je nutné upravit konfigurační soubor a poté aplikaci znovu spustit (konfigurovat nelze za běhu).

**Ovládání.** Zastřešující aplikace se bude starat o ovládání widgetů. Při návrhu bylo myšleno na specifické ovládání SAGE2 aplikací, kdy uživatel často nemá k dispozici klávesnici. V případě zadávání města má uživatel možnost město vyplnit v SAGE UI (bez nutnosti přepnutí do pointer módu) nebo po přepnutí do pointer módu ve widget baru. Ve widget baru má uživatel možnost nejen požadované město vyplnit pomocí klávesnice, ale také smí město vybrat kliknutím na jednu z předvybraných možností. Tato varianta je zamýšlena pro uživatele, kteří nemají k dispozici klávesnici.

**Aktualizace dat.** Zastřešující aplikace se bude starat o aktualizaci dat. Při vyvolání aktualizace dat dojde k novému načtení všech zdrojů s konfigurací, která byla nastavena před vyvoláním aktualizace. Aktualizaci dat bude možné vyvolat dvěma způsoby:

- manuálně, kliknutín na příslušné tlačítko ve widget baru,
- automaticky, vyplněním příslušné konstanty v konfiguračním souboru, která udává počet aktualizací dat za hodinu.

**Validace vstupů.** Jak již bylo zmíněno výše, zastřešující aplikace se bude starat o uživatelské vstupy. Sama uživatelské vstupy nebude validovat, ale předá je dál do konkrétních widgetů. Každý widget si sám daný uživatelský vstup zvaliduje a případně zobrazí chybovou hlášku. Může tak nastat situace, kdy např. widget počasí zobrazí data pro dané město, nicméně widget novinky nezíská žádná data pro toto město a zobrazí chybovou hlášku.

## 4.4 Mapa měst

Widget bude zobrazovat mapu, kde budou graficky zvýrazněna města. Na zvýrazněná města bude možné kliknout a tím dané město označit. Zobrazovaná data všech widgetů z kapitoly Návrh4 se budou vztahovat ke konkrétnímu městu (lokalitě). Jednou z možností bude přijímání uživatelského vstupu ve widget baru, kam uživatel zadá požadovanou oblast, pro kterou chce data zobrazit. Další elegantní možnost výběru města bude poskytovat právě mapa měst. Po kliknutí na konkrétní město zastřešující aplikace upozorní všechny ostatní widgety o změně lokality. Tyto widgety se následně překreslí podle vybraného města na mapě.

Samotný koncept widgetu není ničím vyjímečný a dokonce i pro platformu SAGE2 (pro kterou podle [7] existuje zhruba 50 veřejně dostupných javascriptových aplikací v oficiálním repozitáři) již podobné, mnohem propracovanější mapové aplikace existují. Jako zástupce povedené mapové aplikace je možno uvést *Google Maps* widget.

**Struktura.** Struktura widgetu bude velice jednoduchá. Celý widget bude pokrývat interaktivní mapa, kde bude uživatel mít možnost vybrat požadované město. Mapa bude zároveň reagovat i na ostatní způsoby zadání města (ve widget baru a SAGE UI).

**Funkce.** Při návrhu byly stanoveny následující funkční požadavky:

- zobrazovaná mapa bude načtena asynchronně,
- mapa bude interaktivní (možnost přiblížení, oddálení, zobrazení aktuální polohy),
- při kliknutí na město dojde k ovlivnění celého dashboardu.

Pro zobrazení interaktivní mapy s možností označovat města lze využít *Google Maps API*[7]. Hlavními výhodami *Google Maps API* je bohatá vývojářská dokumentace a vysoká rozšířenost API mezi vývojáři (celá řada problémů a řešení je popsána na diskuzních fórech). *Google maps API* je zdarma pro nekomerční účely.

Mapa měst bude alternativou pro zadání města, výsledek tedy bude stejný, jako kdyby byl daný název města ručně zadán do uživatelského vstupu (o validaci a distribuci dat se opět stará zastřešující aplikace).

**Přínos.** Přínosnost takového widgetu spočívá v provázanosti s ostatními navrhnutými widgety (počasí4.5, novinky4.6, projektor4.7, populace4.8). Widget tedy v důsledku poskytne další možnost uživatelského vstupu. Mapa měst přináší uživateli ještě další výhodu. Při zadání města pomocí textového řetězce ve widget baru se mapa aktualizuje podle zadané lokality a uživatel tím získá další informace týkající se města (pozice na mapě, okolní města atd.). V případě měst se stejným názvem se uživatel podle mapy může ujistit, zda aplikace opravdu vybrala město, které zamýšlel.

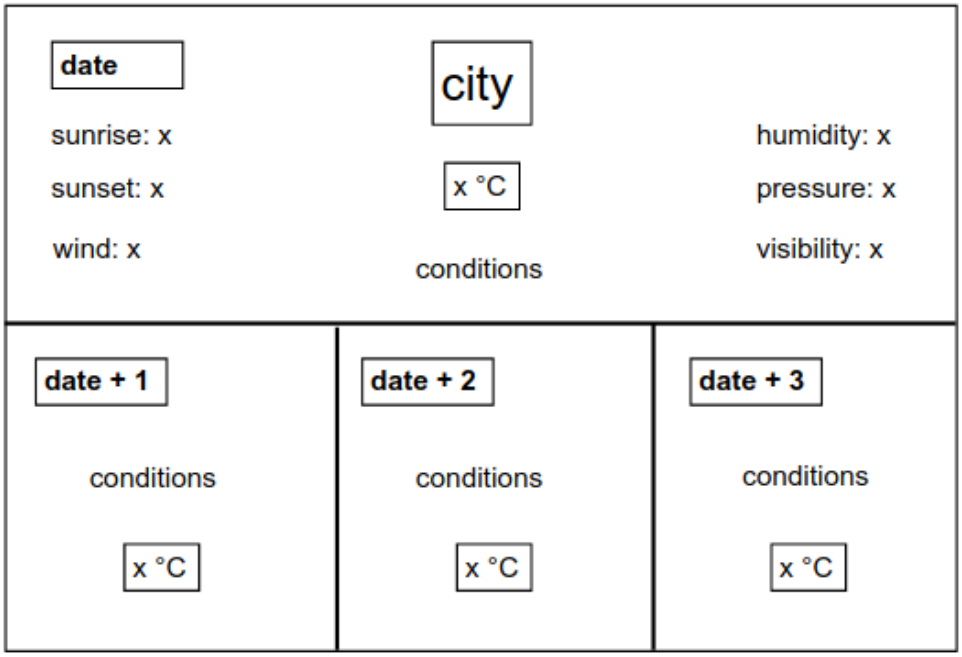
## 4.5 Počasí pro konkrétní oblast

Widget bude zobrazovat počasí pro konkrétní oblast. Danou oblast bude možné za běhu aplikace změnit. Informace pro počasí budou zobrazeny detailně pro aktuální den a dále ve zjednodušené podobě pro dny následující.

[7] https://developers.google.com/maps/

**Struktura.** Widget bude horizontálně rozdělen na dvě poloviny. Horní polovina widgetu bude věnována aktuálnímu dnu, kde se zobrazí detailní informace nejen o teplotě, ale také například o síle větru, západu a východu slunce, vlhkosti nebo viditelnosti.

Spodní polovina widgetu pak bude věnována následujícím dnům. Zde již informace o počasí nebudou tak detailní a budou omezeny pouze na teplotu a slovní popis charakteru počasí. Uživatel bude mít možnost ovlivnit počet následujících dnů, které aplikace bude zobrazovat. Náhled wireframu na obr. 4.2.




Obrázek 4.2: Wireframe widgetu počasí

**Funkce.** Při návrhu byly stanoveny následující funkční požadavky:

- widget nebude fixován na konkrétní město, bude možnost ho za běhu aplikace měnit,
- v případě zadání nevalidního města dojde k zobrazení chybové hlášky,
- počet dnů pro předpověď počasí bude určovat hodnota v konfiguračním souboru,
- widget bude možné lokalizovat do češtiny a angličtiny.

Data o počasí aplikace bude získávat z Yahoo Weather API[8]. Použití API pro nekomerční účely je zcela zdarma, nicméně je limitováno dvěma tisíci požadavky na den.

Aplikace bude lokalizována do češtiny a angličtiny. Přepínat mezi jednotlivými jazykovými mutacemi půjde ve widget baru. V případě, že se uživateli nebude líbit zvolený překlad z angličtiny do češtiny, bude mít možnost dané české slovo (frázi) změnit v konfiguračním souboru aplikace.

Widget bude umožňovat měnit počet zobrazovaných následujících dnů, které jsou viditelné ve spodní polovině aplikace. V případě změny počtu dnů, dojde ke změně šířky boxu každého následujícího dne tak, aby se nový počet boxů vtěsnal do spodní poloviny widgetu. Box aktuálního dne tedy bude na počtu následujících dnů nezávislý a vždy zaujme celou horní polovinu widgetu. V konfiguračním souboru bude možnost dále nastavit například formát data nebo teplotní jednotky.

**Přínos.** Součástí distribuce SAGE2 již je aplikace, která se zabývá počasím. Nicméně aplikace US Weather se omezuje pouze na počasí ve Spojených státech amerických. Navíc zobrazuje pouze teplotu a aktuální podmínky pomocí obrázku.

Tento widget přináší možnost zobrazení počasí pro konkrétní oblast, přičemž danou oblast bude možné za běhu aplikace měnit. Tuto funkcionalitu nenabízí žádná z již existujících aplikací v oficiálním SAGE2 repozitáři.

Další aplikace, která zobrazuje data týkající se počasí je widget kolegy Aleha Kuchynskeho. Tento widget umožňuje zvolit takřka libovolnou oblast, nicméně zobrazovaná data jsou opět velice omezená.

Vylepšením oproti dříve zmíněným aplikacím je rozsah zobrazovaných dat. Pro konkrétní oblast se uživatel dozví např. informace o východu slunce, západu slunce, rychlosti větru, vlhkosti, tlaku atd.

Dalším přínosem widgetu je velký prostor pro vlastní konfiguraci. Konfigurační soubor nabízí řadu možností pro personalizaci aplikace. Další významnou vlastností widgetu je možnost přepínání mezi jazyky. Podobnou funkcionalitu momentálně nenabízí žádný z widgetů v oficiálním repozitáři aplikací.

## 4.6 Novinky

Dalším navrhnutým widgetem bude zobrazovač novinek. Hlavní motivací je absence podobného widgetu v základní sadě aplikací, které jsou distribuovány se SAGE2.

Aplikace bude zobrazovat nadpisy novinek. Každý nadpis bude možné rozkliknout a podívat se do detailu novinky. V detailu se uživatel dozví další údaje o dané zprávě jako např. jméno autora, datum publikace, podrobnější popis

[8] https://developer.yahoo.com/weather/

a nebo související obrázek. Z detailu novinky se uživatel bude moci opět proklinkout zpět na seznam všech posledních zpráv.

**Struktura.** Widget bude poskytovat dvě různé zobrazovací plochy. První z nich bude seznam titulků (obr. 4.3). Titulky budou umístěny v samostatných boxech pod sebou. Po kliknutí na titulek se uživatel dostane na detail článku (obr. 4.4). Detail článku překryje seznam článků, bude tedy roztažen přes celou plochu widgetu. Dominantou detailu článku bude titulek, pod kterým se bude nacházet detailnější popis konkrétního článku. V horní části widgetu bude uveden autor a datum publikace článku. Pod popisem článku se bude nacházet fotografie související s daným tématem. Ne každý článek bude obsahovat výše zmíněné prvky, může se stát, že článku bude chybět příslušný obrázek, nebo např. datum publikace. V takovém případě se zkrátka příslušná informace v detailu článku nezobrazí. Pro návrat zpět na seznam všech článků bude muset uživatel kliknout na libovolné místo na ploše tohoto widgetu.

**Funkce.** Při návrhu byly stanoveny následující funkční požadavky:

- počet novinek bude udávat proměnná v konfiguračním souboru,
- každá novinka bude umožňovat zobrazení detailních informací po kliknutí,
- při pohybu mezi novinkami a detailem nebude docházet k ovlivnění ostatních widgetů (např. nedojde k znovunačtení všech zdrojů).

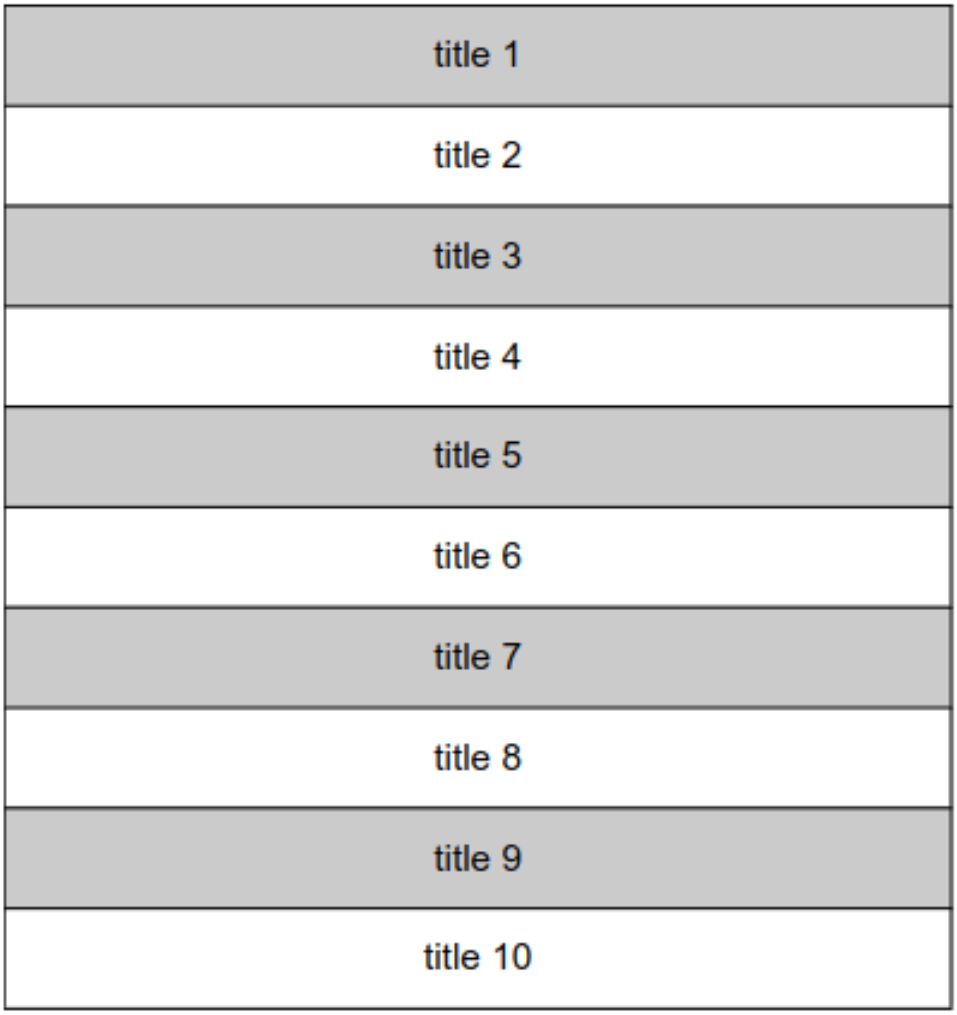
Příslušná data bude widget získávat z News API[9]. News API je dostupné ve třech základních variantách: *Developer*, *Professional* a *Performance*. Pro potřebuje widgetu bohatě dostačuje nekomerční varianta *Developer*, která je kompletně zdarma a poskytuje bohatě postačujících 10 000 požadavků za měsíc.

Počet zobrazovaných novinek bude určovat hodnota v konfiguračním souboru. Při změně hodnoty se vypočte nová výška boxu tak, aby se všechny novinky vešly pod sebe do přiděleného prostoru.

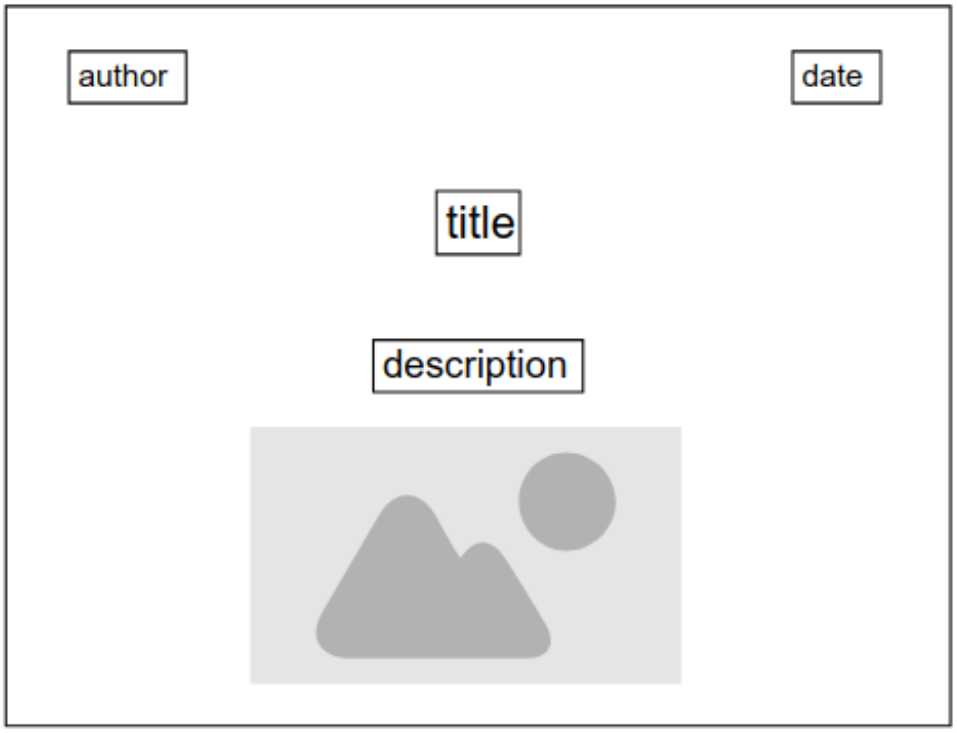
U každé novinky bude možné přepnutí do detailu. Přepnutí do detailu uživatel docílí kliknutím myší na box, ve kterém bude novinka vykreslena. Aplikace poskytne uživateli zpětnou vazbu obarvením aktuálního titulku, nad který uživatel najel myší. Zvýrazňovací barvu bude možno nastavit v konfiguračním souboru.

Jak již bylo zmíněno, widget bude zobrazovat titulky článků. Další variantou, která byla vzata v potaz, bylo zobrazení titulku a příslušného obrázku souvisejícího s článkem. Nicméně tato varianta byla zamítnuta, zejména kvůli

[9] https://newsapi.org




Obrázek 4.3: Wireframe widgetu novinky




Obrázek 4.4: Wireframe widgetu novinky - detail

větší prostorové náročnosti. Je zřejmé, že v případě zobrazení obrázku s titulkem článku by nebyl widget schopný zobrazit srovnatelné množství novinek. V tomto případě tedy dostala přednost kvantita poskytovaných informací.

Aplikace v návrhu počítá s vícejazyčností. Widget bude získávat články z *News API* podle aktuálně zvolené jazykové mutace. Články se tedy budou načítat pro konkrétní město v jazyce, který si uživatel zvolil za svůj výchozí.

**Přínos.** Widget bude agregovat články, které se vztahují k jedné konkrétní oblasti. Samotné *News API*, ze kterého budou články získávány, agreguje více než 5000 informačních zdrojů[8]. Uživatel tedy získá nejnovější informace o konkrétní oblasti a navíc z více informačních zdrojů. Stejnou a ani vzdáleně podobnou funkcionalitu nenabízí žádná z oficiálních aplikací ze SAGE2 repozitáře. Aplikace navíc nebude fixovaná na konkrétní lokalitu a bude podporovat více jazykových mutací.

## 4.7 Projektor obrázků z Flickru

Projektor obrázků, jak již název napovídá, bude zobrazovat různé sekvence obrázků. Zde popisovaný widget *Projektor obrázků* bude vykreslovat tematické obrázky pro konkrétní město. Tyto obrázky aplikace bude získávat za běhu ze sociální sítě pro sdílení fotografií Flickr[10]. Obrázky se tedy budou měnit v případě změny města, pro které se obrázky zobrazují. Obrázky se dále budou měnit v čase, bude docházet k rotaci obrázků přes pozice na widgetu.

**Struktura.** Widget bude podporovat více možných rozložení obrázků na ploše. Jedním z nich bude rozložení s dominantním obrázkem a řada dalších rovnocených obrázků. Dominantní fotografie bude zaujímat větší plochu než ostatní řadové obrázky. Náhled takového rozložení je na obr. 4.5. Další podporované rozložení bude obsahovat sadu stejně velkých obrázků, které jsou rozmístěny na příslušném prostoru widgetu, žádný z nich nebude dominantní.

**Funkce.** Při návrhu byly stanoveny následující funkční požadavky:

- obrázky budou načítány asynchronně, nebudou spomalovat načtení aplikace,
- obrázky budou v čase rotovat na všech pozicích,
- při rotaci se obrázky nebudou znovu načítat z api.

Zobrazované fotografie jsou získávány z Flickr API[11]. Flickr API je zdarma pro nekomerční použití. Pro funkcionality, které využívá *Projektor obrázků z Flickru* dokonce ani není nutná registrace. Nicméně řada dalších metod Flickr API je dostupná pouze pro vlastníky *api_key* řetězce, který uživatel získá registrací projektu na stránkách Flickru. Flickr API poskytuje fotografie v řadě rozlišení, s tím, že nejvyšší možné rozlišení závisí na velikosti původní nahrané fotografie. Nejde tedy dopředu stanovit jedno rozumné rozlišení, ve kterém se budou fotografie vykreslovat. Widget tedy bude vždy získávat fotografii v největším rozlišení, které API poskytuje.

Fotografie, které jsou nahrávány na *Flickr* nejsou striktně limitované rozměry, nebo poměrem stran. Vyvstává zde tedy problém se zobrazením takových fotografií v předem připraveném boxu s napevno určeným poměrem stran. Jedním z naivních řešení by bylo fotografii natvrdo přizpůsobit požadovaným rozměrům. Nicméně takový přístup by danou fotografii deformoval. Dalším z uvažovaných přístupů je zobrazení obrázku s přihlédnutím ke skutečnému poměru stran. Obrázek by tedy byl prezentován nedeformovaný, nicméně nepokrýval by celý box, který mu byl přidělen. V tomto případě bude zvolena

[10] https://www.flickr.com
[11] https://www.flickr.com/services/api/

Obrázek 4.5: Wireframe widgetu Projektor obrázků z Flickru

varianta, kdy si fotografie zachová svůj poměr stran a zároveň vyplní celý prostor. Části fotogtafie, které se do příslušného prostoru nevejdou, budou oříznuty. Takové chování lze spatřit na obr. 4.6. V levo je původní obrázek. Na pravé straně je obrázek, který byl oříznut (v tomto případě nahoře a dole) tak, aby byl zachován původní poměr strana a zároveň obrázek pokryl celý box.

Obrázek 4.6: Oříznutí obrázku za účelem zachování poměru stran a vyplnění celého boxu[9]

Widget bude v tomto případě zobrazovat zejména fotografie týkající se měst, nicméně jeho použití je obecnější. Widget zobrazuje data z *Flickru* na základě textu, který je k fotografii při jeho nahrání na sociální síť připsán. Widget tedy nebude omezen pouze na zobrazení fotografií pro konkrétní města.

Počet načtených fotografií z Flickru by měl být větší než počet aktuálně zobrazovaných fotografií. V opačném případě widget zobrazí všechny fotografie, které byly načteny a na místech, kde se měly nacházet další obrázky bude prázdné místo. Fotografie, které se zprvu nedostanou na plochu widgetu se v důsledku postupné rotace fotografií zobrazí až s určitým časovým odstupem. Fotografie se budou v čase měnit na základě časovače, všechny fotografie tedy v určitém čase budou[12] na dominantní pozici.

**Přínos.** Pro SAGE2 již existuje podobná aplikace *Photo Slideshow*. Tato aplikace umožňuje vykreslování již připravených sad obrázků a vykreslování vlastních obrázků nahraných na SAGE. Nicméně výsledek je vždycky takový, že aplikace vykresluje předem definovanou množinu obrázků. Naproti tomu *Projektor obrázků* je schopen vykreslovat fotografie, které jsou za běhu aplikace získávány z externího zdroje.

## 4.8 Populace města

Populace města bude widget, který, ne příliš překvapivě, vizualituje data týkající se populace konkrétního města a charakterem zobrazovaných informací

[12] v případě, že bude nakonfigurováno rozložení s dominantním obrázkem

se výborně hodí do dashboardu. Pod těmito daty si lze představit např. celkovou populaci města, hustotu osídlení, průměrný věk populace, vývoj počtu obyvatel v čase apod.

Zejména data, která se mění v čase (počet obyvatel v čase, průměrný věk dožití v čase, ...) lze v tomto widgetu zobrazit, pro uživatele přehledným způsobem, pomocí různých typů grafů. Grafové vizualizaci je přímo nakloněna platforma SAGE2, která podporuje atraktivní zobrazení dat pomocí *d3.js* frameworku[13].

**Struktura.** Aplikace bude v levé části zobrazovat data, která jsou jednodušší (bez vývoje v čase). Jsou to např. data jako populace, hustota zalidnění nebo průměrný věk. Pravá část widgetu bude vyhrazena datům, která budou zobrazena s pomocí grafů (vývoj populace v čase, střední délka života v čase). Grafy budou pouze zobrazovat příslušná data, nebudou interaktivní. Wireframe aplikace je k vidění na obr. 4.7.

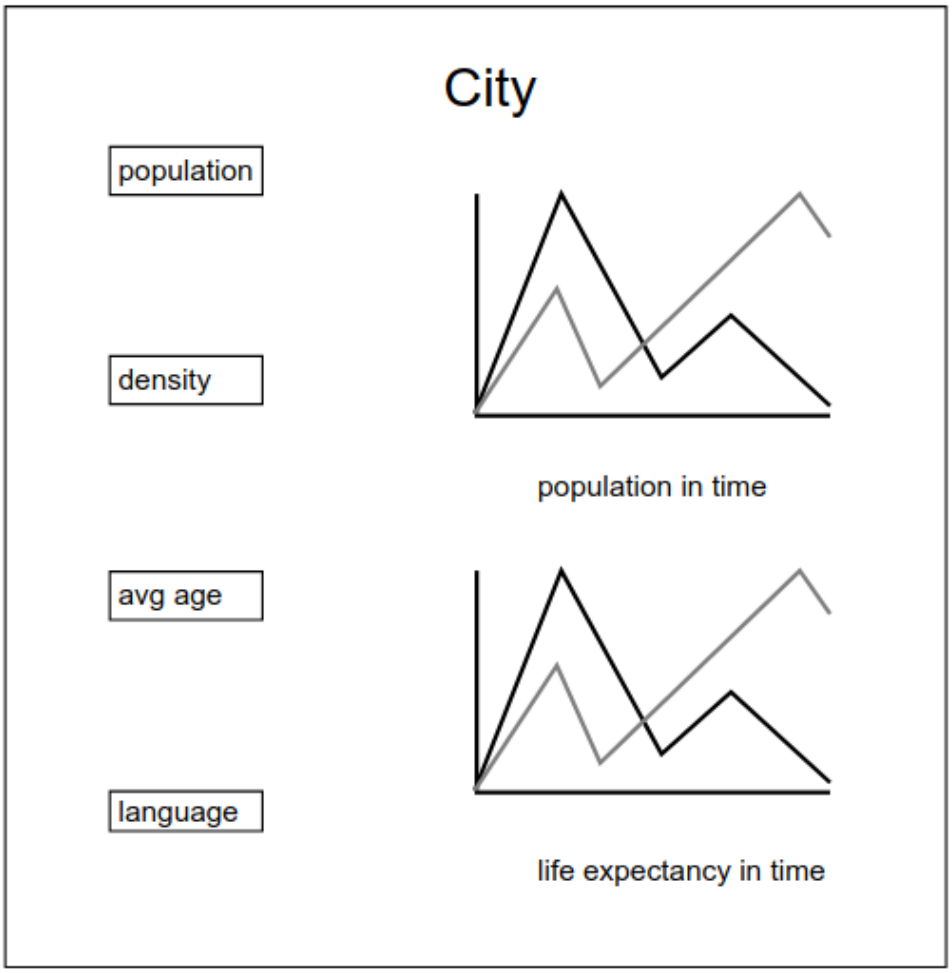
**Funkce.** Při návrhu byly stanoveny následující funkční požadavky:

- zobrazovaná data bude možné ovlivňovat v konfiguračním souboru,
- dlouhodobá data budou zobrazena pomocí grafů.

Příslušná data o populaci pro konkrétní město lze snadno získat pomocí *Teleport public apis*[14]. API poskytuje širokou škálu informací o městech (od dat o kvalitě života ve městě přes informace o bezpečnosti až např. k datům o vzdělání). Podle dokumentace[10] API získává data ze dvou databází. Jsou do databáze *GeoNames*[15] a *Time Zone Database*[16]. API je zcela zdarma pro nekomerční účely.

**Přínos.** Aplikace se výborně hodí do mozaiky výše navržených widgetů, které zobrazují nejrůznější data, která souvisí s konkrétním městem. V oficiálním repozitáři SAGE2 aplikací žádná aplikace neposkytuje podobná data, jako právě tato.

[13] https://d3js.org
[14] https://developers.teleport.org/api/
[15] http://www.geonames.org
[16] https://www.iana.org/time-zones




Obrázek 4.7: Wireframe widgetu populace města

Kapitola 5

# Realizace

V této kapitole budou nejprve uvedeny problémy a zajímavosti, které se týkají zastřešující aplikace a nebo jsou společné pro všechny konkrétní widgety ze sady. Následně bude probrán každý z implementovaných widgetů jednotlivě a budou uvedeny obtíže a jejich řešení, které se vyskytly při jejich realizaci.

## 5.1 Zastřešující aplikace

V této kapitole jsou popsány problémy a jejich řešení, které byly identifikovány při samotné realizaci zastřešující aplikace.

**Třída Dashboard.** Hlavní třída *Dashboard*, která se stará od delegaci všech konrétních widgetů, je třídou, která dědí od *SAGE2_App*. Tuto třídu (dle vývojářské dokumentace[5]) je nutné dědit vždy, při vývoji pro platformu SAGE2.

Hlavním úkolem třídy *Dashboard* je implementace poděděných metod od *SAGE_App*. Těmi nejzajímavějšími metodami jsou:

- Init,
- Draw,
- Resize,
- Event.

Např. metoda Resize se vyvolá vždy, když uživatel změní velikost okna, ve kterém běží aplikace. Tato metoda byla implementována tak, že si metoda nejprve získá aktuální šířku a výšku okna, podle toho upraví velikost hlavního rodičovského SVG kontejneru a následně zavolá inicializaci všech widgetů, které deleguje. Tyto widgety se přizpůsobí aktuální výšce a šířce.

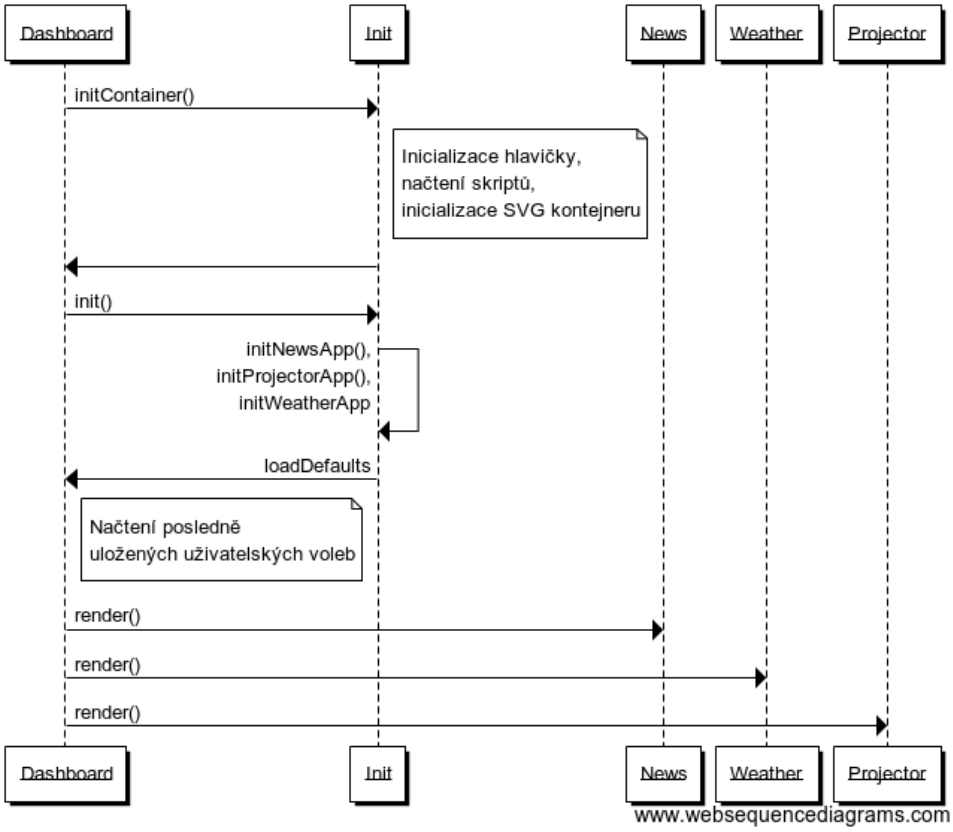
**Konfigurace.** Jedním z hlavních požadavků na aplikaci byla co největší konfigurovatelnost. Dalším úkolem třídy *Dashboard* je načtení konfiguračního souboru. V konfiguračním souboru jsou uvedeny informace týkající se aplikace (verze, výška, šířka, závislosti, licence atd.) a také uživatelská konfigurace. Konkrétně v konfiguračním souboru popisované aplikace jsou uvedeny konstanty, které ovlivňují počet zobrazených článků, formát data, jednotky počasí, barvy použité v aplikaci, velikost písma a v neposlední řadě také překlady. Přáním autora bylo, aby aplikace běžela v českém a anglickém prostředí. Konfigurační soubor tedy obsahuje pole anglických slov s ekvivalentními českými překlady.

Zkrácená ukázka konfiguračního souboru:

```
"dependencies": ["Widget.js", "Init.js"],
  "load": {
    "globalConfig" : {
      "unit": "C",
      "timeFormat24": true,
      "color": "white",
      "articles": "10",
      "hoverColor": "green",
      "imagesTotal": 8,
      "items": 4,
      "textStyle": "uppercase" },
    "translation"{
      "en" : ["Tornado", "Tropical Strom", "Hurricane", "Sunny", "Clear"],
      "cs" : ["Tornádo", "Tropická bouře", "Hurikán", "Slunečno", "Jasno"]
    }
```

**Komunikace tříd.** Třída *Dashboard* dále komunikuje se třídou *Init*. Tato třída se stará zejména o úvodní inicializaci celé aplikace (načtení potřebných skriptů, inicialiazaci položek ve widget baru, vytvoření rodičovského SVG kontejneru pro všechny widgety) a o úvodní inicializaci každého widgetu. Podrobnější popis komunikace mezi třídami lze vidět na sekvenčním diagramu5.1.

**Pameť uživatelské volby.** Jedním z možných vylepšení, která byla zmíněna v analytické části3.3.4, je paměť uživatelské volby. Vztaženo na tuto aplikaci, když uživatel vybere jako oblast jeho zájmu Chrudim a vypne aplikaci. Při opětovném zapnutí aplikace se opět načtou informace týkající se Chrudimi. O tuto funkcionalitu se starají 2 metody: *saveDefaults* a *loadDefaults*. Tyto metody při změně uloží aktuální hodnoty do souboru na SAGE2 serveru a při opětovném spuštění aplikace jsou hodnoty opět načteny do paměti.

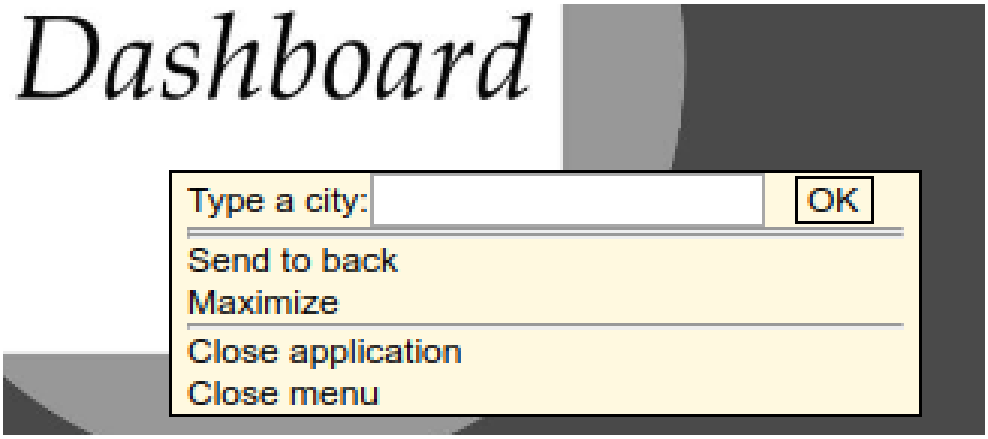



Obrázek 5.1: Sekvenční diagram úvodního vykreslení widgetů

**Třída BaseWidget.** Třída *BaseWidget* je abstraktní třídou, od které dědí konkrétní widgety. Tato třída obsahuje řadu metod, které jsou společné pro všechny konkrétní widgety. Hlavním účelem třídy je poskytnutí metod, které souvisejí s vykreslováním jednotlivých prvků. Třída poskytuje jednoduché rozhraní k vykreslení obrázků, obdélníků, SVG kontejnerů a textů.

**Ovládání.** Aplikaci lze ovládat ve widget baru (obr. 5.2) a v SAGE UI (obr. 5.3). Pro zadání města tedy není potřeba přepnutí do pointer módu a pro jednoduchou ukázku není třeba ani klávesnice. Lokalitu lze zvolit z jednoho ze tří již předvybraných měst.


Obrázek 5.2: Widget bar




Obrázek 5.3: SAGE UI - zastřešující aplikace

**Ošetření vstupů.** Aplikace ošetřuje uživatelské vstupy. V případě, že uživatel zadá neexistující město (nebo pro dané město neexistují žádná relevantní data), aplikace zobrazí chybovou hlášku o daném stavu. Každý widget obsahuje metodu *renderNoData()*, která je volána v případě, že daný widget nezískal příslušná data z api. Ukázka chybového stavu na obr 5.4.


Obrázek 5.4: Zobrazení chybové hlášky

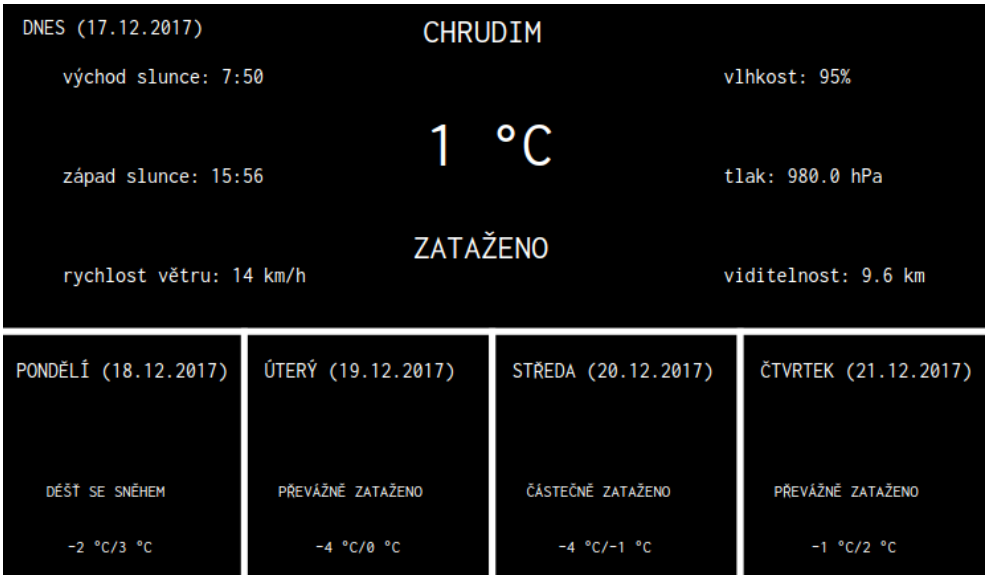
## 5.2 Počasí pro konkrétní oblast

Widget zobrazuje počasí pro konkrétní oblast. Danou oblast (město) lze za běhu aplikace hned několika způsoby měnit. V této kapitole jsou popsány záležitosti týkající se realizace a jejich konkrétní řešení.

**Volání API.** Při získávání dat týkajících se počasí dochází k volání Yahoo Weather API. Takový dotaz může vypadat např. následovně (příklad převzat z oficiální vývojářské dokumentace[11]):

```
select * from weather.forecast where woeid in
(select woeid from geo.places(1) where text="Chrudim")
```

Lze si všimnout, že dotaz zasílaný do API se nápadně podobá tradičním SQL dotazům. API vrátí odpověď ve formátu JSON.




Obrázek 5.5: Vzhled widgetu Počasí pro konkrétní oblast

**Jazykové mutace.** Widget podporuje lokalizaci do angličtiny a češtiny. Uživatel má možnost volbu jazyka provést ve widget baru. Překlad je realizován pomocí dvou polí v konfiguračním souboru. Jedno pole obsahuje anglické výrazy a druhé pole obsahuje příslušné české výrazy. V případě, že se uživateli nelíbí zvolený český překlad, má možnost dané pole upravit a nadefinovat vlastní překladovou frázi.

**Výsledek.** Výsledná aplikace (obr. 5.5) zobrazuje počasí pro konkrétní město a splňuje všechny funkční i nefunkční požadavky, které na ni byly kladeny pří návrhu4.5.

## 5.3 Novinky

Novinky zobrazují sadu článků pro konkrétní město. Widget může být ve dvou stavech, buďto zobrazuje seznam posledních novinek (obr. 5.6), nebo zobrazuje detail konkrétní novinky (obr. 5.7). Se změnou města se změní i konkrétní články, které jsou opět relevantní k zadanému městu.

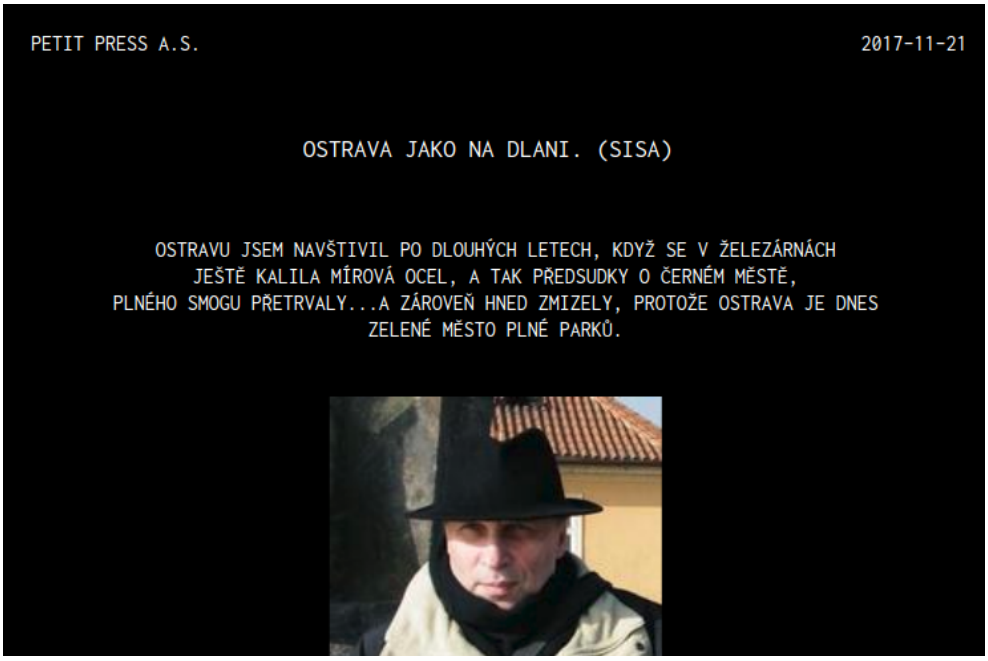
**Přepínání stavu.** Jak již bylo uvedeno, aplikace zobrazuje buď seznam titulků článků nebo detail článku. Do detailu článku se uživatel dostane kliknutím na konkrétní box titulku. Zpět na seznam článků se uživatel dostane kliknutím na libovolné místo widgetu. Při realizaci přepínání stavů je nutná detekce kliknutí myši. Při každém kliknutí myší si aplikace získá souřadnice


Obrázek 5.6: Vzhled widgetu Novinky

kurzoru myši a v případě že je ve stavu, kdy zobrazuje titulky, aplikace určí, zdali dané kliknutí proběhlo v boxu některého z titulků. V případě, že tomu tak je, aplikace smaže elementy, které souvisí se zobrazováním titulků článků a vykreslí ty elementy, které souvisí s detailem článku. V případě přepínání z detailu do seznamu aplikace pouze ověřuje, zda se kurzor myši nachází v prostoru, který byl widgetu přidělen.

**Jazykové mutace.** Widget podporuje dvě jazykové mutace, češtinu a angličtinu. *News API*, ze kterého jsou články získávány umožňuje při posílání požadavku určit jazyk, ve kterém mají články být napsány. Články by měly být tedy v praxi jiné např. pro Prahu a zvolenou češtinu a pro Prahu a angličtinu. Realita je taková, že články opravdu jiné jsou, nicméně na to, že text bude v požadovaném jazyce se spolehnout nedá. Často se stává, že řada článků pro české město, které by měly být v angličtině, jsou ve skutečnosti v češtině. Tento problém je bohužel problémem samotného API a je velice obtížně řešitelný.

**Výsledek** Výsledný widget se skládá ze dvou různých logických částí. První část zobrazuje seznam titulků novinek (obr. 5.6) a druhá zobrazuje detail konkrétního článku (obr. 5.7). Při realizaci byly splněny všechny funkční i nefunkční požadavky, které byly stanoveny při návrhu widgetu4.6.




Obrázek 5.7: Vzhled widgetu Novinky - detail článku

## 5.4 Projektor obrázků z Flickru

Projektor obrázků z Flickru (obr. 5.8) zobrazuje fotografie získané ze sociální sítě Flickr. Fotografie je možné zobrazit v různých konfiguracích. V této kapitole jsou popsány problémy, které se objevily při implementaci a jejich řešení.

**Vhodné zobrazení obrázků.** Jak bylo uvedeno v návrhu4.7, widget zobrazuje obrázky s různorodým poměrem stran v boxech, které mají identické poměry stran. Aby bylo dosaženo zachování původních poměrů stran obrázků a zároveň byl vyplněn celý příslušný box, bylo zvoleno využití vlastnosti *object-fit* HTML tagu *img*. Dle konsorcia W3C[12] vlastnost *object-fit* slouží k určení velikosti obrázku tak, aby odpovídala svému kontejneru. *Object-fit* podporuje řadu hodnot, nicméně pro tento případ je využita hodnota *cover*. Dle W3C[12] hodnota *cover* způsobí, že: „*Nahrazený obsah je dimenzován tak, aby zachoval svůj poměr stran při vyplňování celého obsahu prvku. Objekt bude oříznut tak, aby se vešel do kontejneru.*“

**Rotace obrázků.** V návrhu widgetu4.7 bylo dále uvedeno, že při rotaci obrázků nesmí docházet k dalšímu načítání zdrojů z externího api. Obrázky jsou načteny z api pouze při spuštění aplikace nebo při aktualizaci dat. Při sa-

Obrázek 5.8: Vzhled widgetu Projektor obrázků z Flickru

motné rotaci obrázků dochází pouze k vhodné výměně zdrojových url obrázků u HTML atributů *img*.

**Výběr vhodných obrázků.** Widget zobrazuje obrázky, které souvisí s hodnotou, která reprezentuje město. Nicméně implementace dovoluje zobrazovat obrázky dle libovolného klíčového slova. Obrázky jsou z *Flickru* získávány na základě porovnávání[17] řetězce, který poskytuje widget (hodnota reprezentující město) a textu, který napsal uživatel při nahrávání obrázku na tuto sociální síť. Důsledkem toho tedy není widget omezen pouze na zobrazování fotografií souvisejících s konkrétním městem.

**Výsledek.** Výsledný widget splňuje všechny funkční i nefunkční požadavky, které byly kladeny v návrhu4.7. Widget lze nakonfigurovat do režimu s jednou dominantní fotografií (obr. 5.9) nebo do režimu rovnocených fotografií (obr. 5.8).

[17] o porovnávání se stará samotné api, nikoli autor widgetu


Obrázek 5.9: Vzhled widgetu Projektor obrázků z Flickru

Kapitola 6

# Testování

Dle [13]: „*Testování softwaru je aktivita, která kontroluje, zda skutečné výsledky odpovídají očekávaným výsledkům a zda je softwarový systém bezchybný. Testování softwaru také pomáhá identifikovat chyby, mezery nebo chybějící požadavky v rozporu se skutečnými požadavky. Může být provedeno ručně nebo pomocí automatizovaných nástrojů.*“

## 6.1 Testování během vývoje

Widgety byly testovány v průběhu vývoje. Vývoj widgetů probíhal na lokální SAGE2 na osobním počítači, nicméně aplikace byla v řadě případů testována i v SAGElabu. Testování prováděné během vývoje přinášelo velice podstatné poznatky, které byly následně využívány během následné implementace.

Jedním z podstatných přínosů průběžného testování v SAGElabu bylo např. odhalení špatně implementované synchronizace dat mezi obrazovkami telestěny v SAGElabu. Tento neduh se týkal aplikace *Projektor obrázků z Flickru*, který měl problémy se zobrazováním obrázků na více než jedné fyzické obrazovce. Tento problém byl identifikován právě díky průběžnému testování v SAGElabu a následně byl odstraněn.

## 6.2 Uživatelské testování použitelnosti

Jak uvádí [14], díky uživatelskému testování lze zjistit, jakým problémům čelí skuteční uživatelé. Lze vysledovat, které prvky na jednotlivých stránkách nejsou pro uživatele srozumitelné a zda se uživatelé dokáží v aplikaci dobře orientovat. Průběh uživatelského testování dle [14]:

1. analýza cílových skupin,
2. vytvoření scénáře testování,
3. výběr testerů,

4. samotný průběh testování,

5. analýza výsledků testování.

**Analýza cílových skupin.** „*Na začátku je nutné znát, kdo jsou uživatelé a co by jim měla aplikace nabídnout. Jedině tak je možné testování postavit tak, aby byly jeho výsledky relevantní a užitečné.*“[14]

Všechny implementované widgety jsou relativně obecné, cílovou skupinu tedy tvoří především uživatelé SAGE2 softwaru. Jelikož přístup k laboratoři se SAGE2 softwarem má velice omezená skupina lidí, cílovou skupinou widgetů tak budou především akademičtí pracovníci technických vysokých škol a studentni.

**Vytvoření scénáře testování.** „*Hlavním těžištěm testování jsou úkoly, které jsou jednotlivým testerům uloženy. Jedná se o většinou typické akce cílových skupin.*“[14] Došlo ke stanovení následujících úkolů:

1. změňte město,

2. zobrazte si detail některého článku,

3. vraťte se na seznam článků,

4. přepněte aplikaci do angličtiny,

5. zvolte jedno z předvybraných měst,

6. proveďte manuální aktualizaci všech dat.

**Výběr testerů.** „*Vzorek účastníků testování by měl odpovídat cílovým skupinám. Rovněž je třeba získat zhruba rovnoměrný podíl pokročilých a nezkušených uživatelů – každá skupina totiž narazí na zcela jiný typ problémů.*“[14]

K testování byly přizváni dva studentni Univerzity Hradec Králové, obor informatika a managment a jeden student ČVUT v Praze z fakulty strojní. Vzorek účastníků tedy pokrývá velmi zkušené počítačové uživatele i uživatele pouze s běžnými znalostmi.

**Samotný průběh testování.** „*Pod dohledem zkušeného odborníka na použitelnost plní testeři úkoly dle scénáře testování.*“[14] Samotné testování z organizačních důvodů neprobíhalo v SAGElabu ale na lokální SAGE2 na osobním počítači. Všichni testeři dostali stejnou sadu úloh6.2, které se snažili splnit.

**Analýza výsledků testování.** Všichni testeři úspěšně splnili každý jednotlivý úkol ze seznamu. Nicméně k některým úkolům měli uživatelé přínosné poznámky. Podrobné výsledky uživatelského testování jsou k vidění v následující tabulce.

| | tester 1 | tester 2 | tester 3 |
|---|---|---|---|
| úkol 1 | splněn | splněn | splněn |
| úkol 2 | splněn | splněn | splněn |
| úkol 3 | splněn s výhradami6.2.1 | splněn | splněn s výhradami6.2.1 |
| úkol 4 | splněn | splněn | splněn |
| úkol 5 | splněn | splněn | splněn |
| úkol 6 | splněn s výhradami6.2.2 | splněn | splněn |

### 6.2.1 Neintuitivní návrat z detailu na seznam článků.

Hned dva testeři identifikovali návrat z detailu článku do seznamu článků jako neintuitivní. Tuto akci dokázali vykonat, nicméně provedení jim nebylo na první pohled zřejmé.

### 6.2.2 Zkrácený název tlačítka pro aktualizaci dat.

Jeden z testerů označil text tlačítka pro aktualizaci dat za ne úplně srozumitelný[18]. I přes tento fakt dokázal danou akci vykonat.

## 6.3 Heuristická analýza

Při heuristické analýze odborníci na použitelnost přezkoumají rozhraní aplikace a porovnávají ho s přijatými zásadami použitelnosti. Výsledkem analýzy je seznam možných problémů týkajících se použitelnosti aplikace.[15]

Při popisu výhod a nevýhod vylo čerpáno z [15]. Výhody heuristické analýzy:

- poskytuje rychlou a relativně levnou zpětnou vazbu o použitelnosti aplikace,
- dokáže poskytnout zpětnou vazbu ještě při procesu návrhu aplikace,
- lze použít v kombinaci s dalšími metodami testování použitelnosti.

Heuristická analýza ovšem není všemocná a má i řadu nevýhod. Nevýhoami jsou například:

- vyžaduje znalosti a zkušenosti pro efektivní aplikaci heuristiky,

[18] text tlačítka je *rfrsh*, autor si je vědom, že název není ideální, nicméně SAGE2 klade podmínku maximálně pěti písmenného názvu tlačítka

- vyškolených expertů není mnoho a mohou být drazí,
- mělo by být použito více expertů najednou.

Existuje řada používaných heuristik, necméně jednou z nejznámějších, které se používají při analýze, jsou heuristická pravidla Jakoba Nielsena. Jedná se o deset následujících pravidel.[15]

- **Viditelnost stavu systému.** Systém by měl poskytovat uživateli zpětnou vazbu o tom, kde se nachází,
- **Propojení systému a reálného světa.** Systém by měl používat popisky v souladu s terminologií cílové skupiny.
- **Uživatelská kontrola a svoboda.** Systém by měl poskytovat možnost opustit současný stav aplikace bez dlouhé sekvence kroků.
- **Konzistence a standardy.** Systém by měl dodržovat konvence pro danou platformu.
- **Prevence chyb.** Systém by měl být navržený tak, aby předcházel chybovým zprávám.
- **Rozpoznání namísto vzpomínání.** Systém po uživateli nevyžaduje paměť informací z předcházejících akcí. Takové informace a pokyny by měly být snadno viditelné a dostupné.
- **Flexibilita a účinnost použití.** Systém umožňuje uživatelům přizpusobovat časté akce. Urychluje práci se systémem zkušeným uživatelům, nicméně je snadno použitelný i nováčky.
- **Estetický a minimalistický design.** Dialogy by neměly obsahovat informace, které jsou irelevantní nebo zřídka kdy potřebné.
- **Pomoc uživatelů pochopit, poznat a vzpamatovat se z chyb.** Chybová hlášení by měla být vyjádřena v běžném jazyce (bez kódů), přesně označovat problém a konstruktivně navrhnout řešení.
- **Nápověda a dokumentace.** Nápověda systému by měla být snadno prohledávatelná, zameřená na úkol a měla by uvádět konkrétní kroky, které je potřeba provést. Neměla by být příliš rozsáhlá.

**Analýza výsledků.** V této bakalářské práci byla zvolena výše uvedená Nielsenova heuristika. Do role odborníků byly pasováni stejní testeři, jako v kapitole Uživatelské testování použitelnosti6.2. Jsou tedy celkem tři. Heuristická analýza probíhala po uživatelském testování použitelnosti, experti již tedy byly dobře seznámeni s danou aplikací. Odborníci spolu v průběhu analýzy

nekomunikovali. Ná závěr analýzy odborníci sepsali problémy, které při průchodu aplikací objevily a následně je označily dle jejich závažnosti. Po agregaci výstupů od odborníků byly identifikovány následující problematické části aplikace.

### 6.3.1 Stav aplikace Novinky – vysoká priotita

Uživatel nemá zpětnou vazbu od systému v jakém stavu se nachází. Zejména detail článku může být pro uživatele matoucí. Uživateli vůbec nemusí dojít, že z detailu je možné se vrátit zpět na seznam všech článků.

### 6.3.2 Nabídka posledně zadaných měst – nízká priorita

Aplikace nenabízí seznam posledně zadaných měst. Taková nabídka by umožnila uživatelům zkrátit čas při výkonu častých akcí, jakou je právě změna města.

# Závěr

Tato bakalářská práce se věnuje návrhu a vývoji widgetů pro platformu SAGE2. Jednotlivé widgety jsou ovládány jednou zastřešující aplikací.

Nejprve došlo k samotné analýze platformy SAGE2 a SAGE2 API. V té samé kapitole byly následně analyzovány již existující aplikace pro platformu SAGE2. V další kapitole došlo na návrh pěti různých widgetů a samotné zastřešující aplikace. Samotný návrh je značně ovlivněn analýzou již existujících aplikací. Po návrhu došlo na samotnou realizaci tří vybraných widgetů. Realizované widgety byly následně vhodně otestovány.

## Možná vylepšení

Jak bylo uvedeno v samotném úvodu práce, při návrhu a následném vývoji bylo upuštěno od snahy o dokonalý vzhled aplikace. Zcela logicky je tedy největší prostor pro vylepšení právě v této oblasti. Samotné aplikaci by prospělo, kdyby na tuto práci navázal student s grafickými schopnostmi a posunul tak aplikaci zase o kus dál.

## Splnění zadání

1. **Analýza API SAGE2, analýza stávajících aplikací, možnost modulového systému jedné zastřešující aplikace.** Tomuto bodu se věnuje celá analytická část3 bakalářské práce. Nejprve je analýze podrobena samotná platforma SAGE2 a SAGE2 API. Následně jsou v práci analyzovány již existující aplikace. Zdrojem aplikací je oficiální SAGE2 repozitář pro aplikace a widgety dalších studentů ČVUT FIT v Praze.

2. **Stanovení obecných i konkrétních omezení a doporučení pro budoucí vyvíjená ovládací rozhraní.**

   Obecná i konkrétní omezení a doporučení pro budoucí vyvíjená ovládací rozhraní jsou uvedena v analytické části bakalářské práce3. Tato dopo-

ručení a omezení vycházejí zejména z analýzy SAGE2 API a analýzy existujících aplikací.

3. **Návrh minimálně pěti konkrétních aplikací, jejich způsob realizace (organizace, komunikace, ovládání, konfigurace) a popis způsobu získávání dat z externích zdrojů.** V této4 kapitole je navrhnuto 5 konkrétních widgetů a jejich zastřešující aplikace. U každého widgetu jsou uvedeny funkční, nefunkční požadavky a wireframe widgetu. Konkrétně byly navrženy widgety *Mapa měst*4.4, *Počasí pro konkrétní oblast*4.5, *Novinky*4.6, *Projektor obrázků z Flickru*4.7 a *Populace města*4.8.

4. **Implementace alespoň tří z navržených aplikací.** Z navržených widgetů došlo k implememtaci *Počasí pro konkrétní oblast*5.2, *Novinky*5.3 a *Projektor obrázků z Flickru*5.4. Detaily týkající se implementace jsou uvedeny v kapitole *Realizace*5. Všechny tyto widgety je možné reálně spustit na platformě SAGE2.

5. **Testování implementovaných widgetů.** Implementované widgety byly vhodně otestovány v kapitole *Testování*6. Widgety procházeli průběžným testováním při vývoji na lokální SAGE2 a v SAGElabu. Na závěr došlo na uživatelské testování použitelnosti.

# Literatura


[1] IT-SLOVNIK.cz: *Widget [online]*. [cit. 2017-12-22]. Dostupné z: https://it-slovnik.cz/pojem/widget/

[2] Google: *Android 1.5 [online]*. [cit. 2017-12-22]. Dostupné z: https://developer.android.com/about/versions/android-1.5-highlights.html

[3] EVL: *SAGE2 introduction [online]*. [cit. 2017-10-07]. Dostupné z: http://sage2.sagecommons.org/project/introduction/

[4] EVL: *SAGE2 user documentation [online]*. [cit. 2017-10-07]. Dostupné z: http://sage2.sagecommons.org/download/1312/

[5] EVL: *SAGE2 developer documentation [online]*. [cit. 2017-10-07]. Dostupné z: http://sage2.sagecommons.org/download/1309/

[6] CESNET: *SAGElab [online]*. [cit. 2017-11-18]. Dostupné z: https://sagelab.cesnet.cz/cz/

[7] Bitbucket: *SAGE2 javascript app repository [online]*. [cit. 2017-12-07]. Dostupné z: https://bitbucket.org/sage2/sage2_apps/src/d844a3139bea77c2025b7633bdae6dcfdc4c4e31/javascript/?at=master

[8] News api: *News api [online]*. [cit. 2017-12-25]. Dostupné z: https://newsapi.org

[9] Css tricks: *Object-fit property [online]*. [cit. 2017-12-27]. Dostupné z: https://css-tricks.com/almanac/properties/o/object-fit/

[10] Teleport: *Teleport api developer documentation [online]*. [cit. 2017-12-07]. Dostupné z: https://developers.teleport.org/api/

[11] Yahoo: *Yahoo weather api [online]*. [cit. 2017-12-27]. Dostupné z: https://developer.yahoo.com/weather/

[12] W3C: *CSS The object-fit Property [online]*. [cit. 2017-12-27]. Dostupné z: https://www.w3schools.com/css/css3_object-fit.asp

[13] Guru99: *What is Software Testing? [online]*. [cit. 2017-12-29]. Dostupné z: https://www.guru99.com/software-testing-introduction-importance.html

[14] Dobrý web: *Uživatelské testování použitelnosti [online]*. [cit. 2017-12-29]. Dostupné z: http://www.dobryweb.cz/uzivatelske-testovani

[15] usability.gov: *Heuristic Evaluations and Expert Reviews [online]*. [cit. 2018-01-03]. Dostupné z: https://www.usability.gov/how-to-and-tools/methods/heuristic-evaluation.html

Příloha A

# Seznam použitých zkratek

**SAGE** Scalable Amplified Group Environment

**UI** User Inerface

**HTML** HyperText Markup Language

**URL** Uniform Resource Locator

**W3C** World Wide Web Consortium

**JSON** JavaScript Object Notation

**API** Application Programming Interface

**DOM** Document Object Model

**SVG** Scalable Vector Graphics

**ČVUT FIT** České vysoké učení technické, fakulta informačních technologií

Příloha **B**

# Obsah přiloženého CD

- `src`
  - `impl` ..... zdrojové kódy implementace
    - `Dashboard` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `BP_Radek_Meduna_2018.pdf` ..... text práce ve formátu PDF