

ZÁVĚREČNÁ STUDIJNÍ PRÁCE

dokumentace

Multiplatformní desktopová aplikace v Electronu

Petr Volf



ELECTRON

Obor: 18-20-M/01 INFORMAČNÍ TECHNOLOGIE
se zaměřením na počítačové sítě a programování

Třída: IT4

Školní rok: 2017/2018

Prohlašuji, že jsem závěrečnou práci vypracoval samostatně a uvedl veškeré použité informační zdroje.

Souhlasím, aby tato studijní práce byla použita k výukovým účelům na Střední průmyslové a umělecké škole v Opavě, Praskova 399/8.

V Opavě 31. 12. 2017

podpis autora práce

ANOTACE

Cílem projektu bylo vytvořit multiplatformní desktopovou aplikaci, která bude sloužit k přístupu ke službě Trello s možností manipulace a prohlížení dat za použití REST API k přístupu k datům. Součástí cíle je propojení dat se službami jako Google kalendář a další služby a možnost pracovat s daty bez přístupu k internetu. Aplikace byla naprogramována s použitím frameworku Electron a jazyka Javascript. Uživatelská část aplikace je napsána s použitím knihovny React na tvorbu uživatelských rozhraní. Pro napsání stylů je použit jazyk Sass. Na celou práci jsou použity formátovací pravidla pro zajištění jednoty vzhledu kódu.

OBSAH

OBSAH.....	4
1 VYUŽITÉ TECHNOLOGIE	6
1.1 ELECTRON.....	6
1.2 REACT	6
1.2.1 JSX.....	7
1.3 TYPESCRIPT.....	7
1.4 SASS.....	8
1.5 VISUAL STUDIO CODE	8
1.6 TSLINT / ESLINT	9
1.7 NPM MODULY	9
2 FINÁLNÍ PROVEDENÍ	10
3 HISTORIE VÝVOJE APLIKACE	13
ZÁVĚR	15
SEZNAM POUŽITÝCH INFORMAČNÍCH ZDROJŮ	16

ÚVOD

V dnešní době vzniká software v týmech lidí, spolupracujících na určitých úkolech. Největším slabinou těchto týmů je především komunikace a plánování. Trello je jedna ze služeb, která umožňuje kolaboraci na plánování projektů. Tato služba však nefunguje offline a propojení s více než jednou další službou je zpoplatněno.

Mým cílem bylo tyto omezení odstranit vytvořením vlastní aplikace. Aplikace by následně uměla propojit data a umět je zobrazit více způsoby.

V následující kapitole jsem vypsál použité technologie a důvody pro jejich volbu. V předposlední kapitole je popis stávající verze aplikace. Nakonec v poslední kapitole je rozepsána historie tvorby projektu s některými problémy, se kterými jsem se potýkal.

1 VYUŽITÉ TECHNOLOGIE

1.1 Electron

Electron je framework pro tvorbu desktopových aplikací v Javascriptu, HTML a CSS. Běžící program je tvořen dvěma procesy, hlavním a vykreslovacím. Hlavní proces má přístup k vlastnostem programu jako tvorba menu, nastavení klávesových zkratk a nastavení okna. Vykreslovací proces se stará o stránku a je postavený na Chromiu. Funguje skoro stejně jako běžný JavaScript v prohlížeči s tím rozdílem, že má přístup k Node JS modulům. Procesy mezi sebou komunikují pomocí eventů. Tento framework jsem si vybral proto, že jsem chtěl, aby moje aplikace fungovala na všech hlavních desktopových platformách, a navíc jsem chtěl, aby vypadala dobře.

1.2 React

React je JavaScriptová knihovna na tvorbu uživatelských rozhraní. Rozhraní je tvořeno komponentami, které mohou odpovídat na eventy. Komponenty se tvoří s předem daným vstupem (props). Komponenty, které to vyžadují, mohou ještě obsahovat vnitřní stav (state). Vstup i stav jsou jen pro čtení. Stav se mění pomocí metody setState, která se postará i o potřebné překreslení. React jsem si zvolil pro tvorbu uživatelského rozhraní, protože se mi nelíbila práce se šablonovacími systémy, a navíc jsem byl zvyklý na komponenty díky předchozím projektům v jazyce C#.

1.2.1 JSX

JSX je rozšíření syntaxe pro JavaScript. Kombinuje Javascript a XML syntax. Díky tomu se React komponenty píší daleko snadněji. Nevýhodou je, že se zdrojové soubory musí transpilovat do prohlížečem čitelného souboru. Tento krok mi ale nevadil, jelikož jsem kvůli používání nových JavaScriptových funkcí tento krok už prováděl. Zpočátku jsem na to používal Babel, ale potom už Typescript.

```
class Header extends React.Component<any, any> {  
  public goBack = () => {  
    this.props.changePage('home')  
  }  
  
  public render () {  
    return (  
      <div className='titleHeader'>  
        <button className='buttonHeader' onClick={this.goBack}>  
          <FontAwesomeIcon icon='chevron-left' size='2x' />  
        </button>  
        <h1>Settings</h1>  
      </div>  
    )  
  }  
}
```

Obrázek č.1 – jednoduchá React komponenta

1.3 TypeScript

TypeScript je rozšíření syntaxe JavaScriptu o typy, rozhraní a další vylepšení. Tyto typy jsou kontrolovány během kompilace a tím se negeneruje pomalejší kód. Díky typům dokáže editor nabídnout daleko lepší automatické doplňování kódu. TypeScript je kompatibilní s JSX. Typescript se také transpiluje do standardního JavaScriptu. Mnoho modulů Node JS má veřejně dostupné typové definiční soubory, díky kterým není potřeba přepisovat již stávající kód do

Typescriptu pro využití kontroly typů. Na TypeScript jsem přešel v půlce listopadu. Zaujalo mě vylepšené doplňování kódu oproti standartnímu JavaScriptu.

1.4 SASS

Sass je rozšíření CSS. Umožňuje použití konstant, příkazů if a dalších, vnořování selektorů a mnoho dalších funkcí. Sass jsem si vybral kvůli přidaným funkcím.

```
.star{
  margin: 5px 5px auto auto;
  color: ■black;
  &-dark{
    color: □white;
  }
  &:hover{
    opacity: 0.6;
    color: ■gold;
  }
  &-full{
    color: ■gold;
    &:hover{
      opacity: 0.6;
    }
  }
}
```

Obrázek č.2 – ukázka Sass syntaxe

1.5 Visual Studio Code

Visual Studio Code je rozšířitelný textový editor s některými funkcemi IDE. Je pro něj veliký výběr různých doplňků. Má vestavěný debugger pro Node JS, což mi ulehčilo práci. S tímto editorem jsem už měl zkušenosti a líbí se mi jeho rychlost oproti plným IDE.

1.6 TSLint / ESLint

Ze začátku jsem používal ESLint pro zajištění jednoty vzhledu kódu. ESLint má i integraci do VS Code díky doplňku, takže editor přímo podtrhuje problémový kód. Po přechodu na Typescript jsem ESLint vyměnil za TSLint, kvůli tomu, že ESLint nepodporuje Typescriptovou syntax.

1.7 NPM moduly

Pro ulehčení a zrychlení práce jsem použil několik modulů. K většině z nich jsem po přechodu na Typescript stáhl volně dostupné typové definice. Pro autorizaci k Trello používám modul `oauth`. Pro ikony jsem použil Font Awesome, který jsem ze začátku používal jako přibalené CSS soubory, ale v prosinci vyšla verze 5, která byla dostupná jako NPM modul pro React, což mi více vyhovovalo. Dále využívám modul „sharp“ pro konverzi a úpravu obrázků. Pro seřazování pomocí drag & dropu používám modul „SortableJS“. Pro nastavení výšku textového pole používám modul „autosize“. Pro automatizaci procesu kompilace jsem použil „gulp“. Nakonec jsem použil modul „is-online“ pro zjištění dostupnosti internetového připojení.

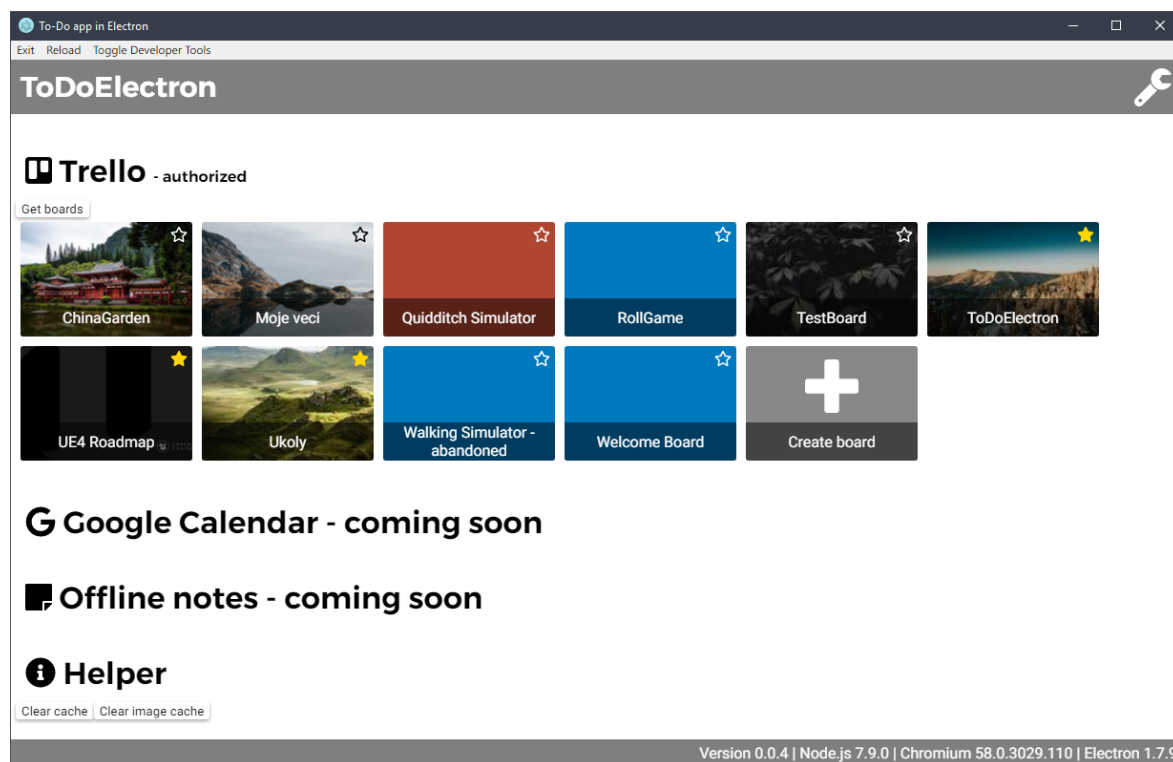
```
"dependencies": {
  "@fortawesome/fontawesome": "^1.0.0",
  "@fortawesome/fontawesome-free-brands": "^5.0.0",
  "@fortawesome/fontawesome-free-regular": "^5.0.0",
  "@fortawesome/fontawesome-free-solid": "^5.0.0",
  "@fortawesome/react-fontawesome": "0.0.14",
  "autosize": "^4.0.0",
  "dotenv": "^4.0.0",
  "is-online": "^7.0.0",
  "oauth": "0.9.15",
  "react": "^16.0.0",
  "react-dom": "^16.0.0",
  "sharp": "^0.18.4",
  "sortablejs": "^1.6.1"
}

"devDependencies": {
  "@types/react": "^16.0.25",
  "@types/react-dom": "^16.0.3",
  "@types/sharp": "^0.17.5",
  "@types/sortablejs": "^1.3.32",
  "@types/dotenv": "^4.0.2",
  "electron": "^1.7.6",
  "electron-rebuild": "^1.6.0",
  "gulp": "^3.9.1",
  "node-sass": "4.5.3",
  "tslint": "^5.8.0",
  "typescript": "^2.6.1"
},
```

Obrázek č. 3 – potřebné moduly v package.json souboru

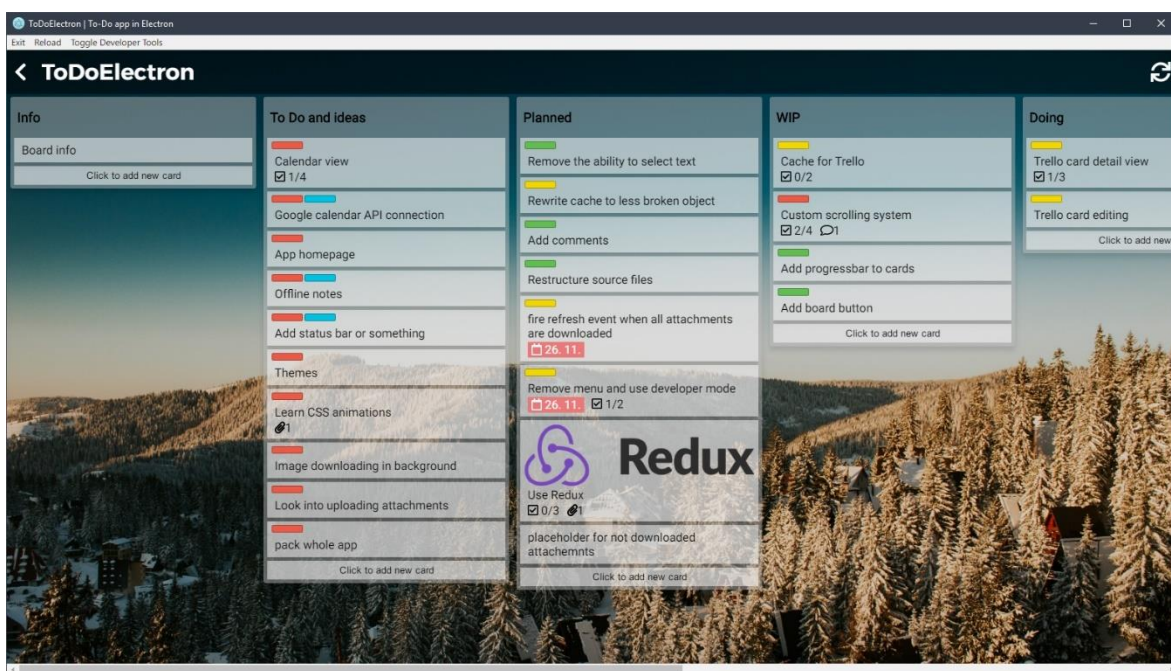
2 FINÁLNÍ PROVEDENÍ

Po spuštění aplikace se zobrazí domovská stránka. Pokud je uživatel přihlášený, tak se zobrazí seznam nástěnek, nebo tlačítko pro autorizaci. Autorizace funguje na OAuth protokolu, takže heslo uživatele se neukládá a veškerá komunikace probíhá přes HTTPS. Program si pamatuje autorizační token, který je uložený na disku. Uživatel má možnost z domovské stránky zajít do nastavení a změnit je, nebo také vyčistit cache. Cache automaticky označí stažená data za neplatná po 24 hodinách od jejich stažení a tím je nutné je aktualizovat.



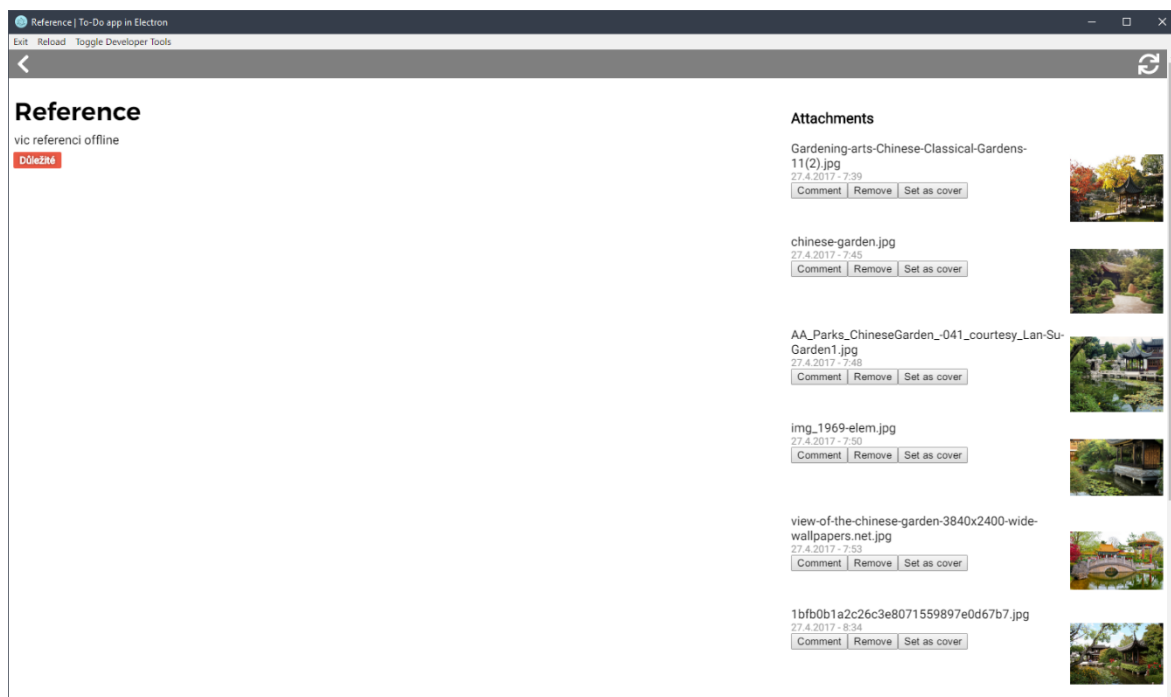
Obrázek č.4 – Domovská stránka

Domovská stránka obsahuje seznam nástěnek uživatele. Po kliknutí na tlačítko nástěnky se uživatel dostane na zobrazení dané nástěnky. Zde je zobrazení srovnatelné s online verzí Trelly. Karty jsou stejně jako v online verzi uspořádány v listech. Pokud program zjistí připojení k internetu, automaticky aktualizuje zobrazené údaje. Možná je také manuální aktualizace. Uživatel může na nástěnce vytvářet listy i karty, přesouvat je, přejmenovat nástěnku nebo listy.



Obrázek č.5 – Pohled nástěnky

Po kliknutí na kartu se zobrazí okno s detaily karty. Zde může uživatel přejmenovat kartu, nebo pozměnit některé údaje. Zde se zobrazují také veškeré přílohy připojené ke kartě. Pokud jde o obrázek a uživatel klikne na miniaturu, otevře se plná verze ve vybraném prohlížeči obrázků. Uživatel zde také nastavuje, která obrázková příloha se zobrazí na vybrané kartě na pohledu nástěnky.



Obrázek č.6 – Pohled karty

3 HISTORIE VÝVOJE APLIKACE

Projekt jsem začal tvořit na začátku června. Na odstartování jsem použil prázdný projekt „electron-quick-start“ volně dostupný na GitHubu. Následovalo připojení Trello API. Ze začátku jsem jen zobrazoval údaje, ale s postupem času jsem začal data i editovat. Stažená data se ukládají do cache na disk. Výsledky editace dat nejsou prozatím ukládány do cache, protože bez přístupu k internetu jsou většinou editace k ničemu. Na této stránce projektu hodlám pokračovat až později.

K integraci Google kalendáře jsem se bohužel nedostal kvůli nedostatku času. Místo jsem upřednostnil intergrací funkcí jako drag & drop seřazování a dalším. Mnoho částí kódu prošlo mnohými přepsáními. Ve chvíli kdy jsem se naučil další funkce jazyka, nebo když jsem přešel na Typescript, jsem měl nutkání přepsat starý kód, aby se svou čitelností a kvalitou rovnal novým částem kódu. Jen cache jsem nějak výrazně nepřepsal.

Díky tomu, že Unreal Engine má veřejně dostupnou Trello nástěnku s plánovacími funkcemi, mě napadlo, že použiji jejich nástěnku jako zátěžový test pro mou aplikaci. Jejich nástěnka ale obsahuje velké množství GIF obrázků. Napadlo mě, že do mého programu přidám knihovnu na konverzi obrázků, pro snížení zátěže na procesor u slabších počítačů.

Okolo začátku prosince jsem přepsal aplikaci, aby používala jen jednu stránku pro všechny své funkce. Hlavní motivací pro toto rozhodnutí bylo především snížení načítacích časů jednotlivých stránek. Díky tomu, že aplikace je napsaná v interpretovaném jazyce, každá stránka při načtení kompiluje přiložený skript. Tento skript díky importům mnohdy daleko zvýšil dobu před prvotním vykreslením uživatelského rozhraní. Díky vestavěným nástrojům v Chromiu jsem

zjistil, že se tato doba pohybovala okolo sekundy. Prvních 400 milisekund se přitom nic nenačítalo, ani nekompilevalo. To se mi nezdálo ale nevěděl jsem, co s tím dělat. Dalších 300 milisekund trvala kompilace všech importovaných skriptů. To byl můj hlavní cíl optimalizace. Kdybych všechny potřebné skripty importoval při startu, tak by načítání stránek netrvalo tak dlouho. Po přepisu všeho na jednu stránku zmizel čas potřebný ke kompilaci i čas, kdy se nedělo nic. Jedinou nevýhodou je nepěkný kód na několika místech, ale to vyřeším v budoucí verzi.

V půlce prosince jsem úplnou náhodou našel obrovskou bezpečnostní chybu. Nenapadlo mě, že může dojít ke zneužití mých autorizačních údajů k Trello, které jsem měl nějakou dobu commitnuté na GitHubu. Chybu v kódu jsem okamžitě opravil, takže stávající kód používá proměnné prostředí pro tyto citlivé údaje. Smazat tyto údaje z Githubu už je naneštěstí možné už jen naprostým smazáním repozitáře jak na serveru, tak i lokálního a vytvoření nového.

ZÁVĚR

Jako hlavní poučení z této práce si беру, že nedokážu správně odhadnout kolik práce jsem schopný za určitý čas stihnout. Cíle jsem si nastavil moc vysoko s ohledem na mé zkušenosti, díky čemuž jsem se ze začátku vývoje zasekával i na triviálních problémech. V současném stavu aplikace nenabízí žádnou výraznou výhodu oproti online verzi Trella. Většinu svých cílů jsem ale splnil. Na vývoji této aplikace budu pokračovat i nadále, aby se dostala do použitelného stavu.

SEZNAM POUŽITÝCH INFORMAČNÍCH ZDROJŮ

[1] *Electron* [online]. [cit. 2017-12-21]. Dostupné z: <https://electronjs.org/>

[2] *React* [online]. [cit. 2017-12-21]. Dostupné z: <https://reactjs.org/>

[3] *Typescript* [online]. [cit. 2017-12-21].

Dostupné z: <https://www.typescriptlang.org/>

[4] *Sass* [online]. [cit. 2017-12-21]. Dostupné z: <http://sass-lang.com/>

[5] *Stack Overflow* [online]. [cit. 2017-12-21].

Dostupné z: <https://stackoverflow.com/>

[6] *Web technology for developers* [online]. [cit. 2017-12-21].

Dostupné z: <https://developer.mozilla.org/en-US/docs/Web>