

MASARYKOVA UNIVERZITA
FAKULTA INFORMATIKY



Robomise v mobilnom prostredí

BAKALÁRSKA PRÁCA

Miriama Zemaníková

Brno, jeseň 2019

MASARYKOVA UNIVERZITA
FAKULTA INFORMATIKY



Robomise v mobilnom prostredí

BAKALÁRSKA PRÁCA

Miriama Zemaníková

Brno, jeseň 2019

Na tomto mieste sa v tlačenej práci nachádza oficiálne podpísané zadanie práce a vyhlásenie autora školského diela.

Vyhlásenie

Vyhlasujem, že táto bakalárska práca je mojím pôvodným autorským dielom, ktoré som vypracovala samostatne. Všetky zdroje, pramene a literatúru, ktoré som pri vypracovaní používala alebo z nich čerpala, v práci riadne citujem s uvedením úplného odkazu na príslušný zdroj.

Miriama Zemaníková

Vedúci práce: Mgr. Marek Grác, Ph.D.

Podakovanie

These are the acknowledgements for my thesis, which can span multiple paragraphs.

Zhrnutie

This is the abstract of my thesis, which can span multiple paragraphs.

Klíčové slová

android , mobilná aplikácia, react native

Obsah

1	Úvod	1
1.1	<i>Motivácia</i>	1
1.2	<i>Ciele práce</i>	2
1.3	<i>Štruktúra práce</i>	2
2	Používateľské testovanie webovej aplikácie	3
2.1	<i>Webová aplikácia Robomise</i>	3
2.1.1	Princíp hry	3
2.1.2	Hracia plocha	4
2.1.3	Úlohy a program	4
2.2	<i>Príprava testovania použiteľnosti</i>	4
2.2.1	Testovací scenár	5
2.2.2	Testovací účastníci	6
2.3	<i>Priebeh testovania</i>	6
2.4	<i>Vyhodnotenie testovania</i>	7
2.4.1	Vyhodnotenie dát	7
2.4.2	Návrhy riešení	9
3	Mobilná aplikácia	13
3.1	<i>React a React Native</i>	13
3.2	<i>Základná kostra projektu</i>	14
3.3	<i>Navigácia</i>	15
3.4	<i>Správa stavov</i>	16
3.5	<i>Api</i>	17
3.6	<i>Obrázky a Štýly</i>	19
	Bibliografia	21

Zoznam tabuliek

Zoznam obrázkov

- 2.1 Príklad riešenia úlohy hry Robomise 5
- 2.2 Úloha Obklíčený diamant 8
- 2.3 Vylepšená úloha Obklúčený diamant 10
- 3.1 Schéma architektúry React Native a porovnanie s ReactJS. 14

1 Úvod

Od roku 2000 narástol počet užívateľov mobilných zariadení až päťnásobne. Podľa štúdie Cisco Visual Networking Index[1] sa má toto množstvo do roku 2020 zvýšiť až na 5,5 miliardy, čo je 70 % svetovej populácie. Viac ľudí tak bude mať mobilný telefón ako pripojenie k elektrine.

Podľa štatistických dát analytickej služby StatCounter[2] sa od roku 2016 viac pristupuje na webové stránky z mobilných zariadení ako z klasických počítačov. Aktuálny percentuálny pomer prístupu z mobilných zariadení je 53,8 %.

Z prognózy Cisca a podľa štatistík StatCounter vyplýva, že je nutné vhodným spôsobom optimalizovať webové stránky, aby ich zobrazenie na mobilnom zariadení nebolo ovplyvnené kvalitou pripojenia, ako je to u webových stránok, a aby neboli ovládané iba počítačovou myšou, ktorú si k telefónu nepripojíme. Naopak, je vhodné využiť výhody mobilných zariadení, ako napríklad možnosť pracovať v offline režime alebo použitie dotykov na ovládanie celej aplikácie.

Preto je moja práca zameraná na vývoj novej natívnej mobilnej aplikácie Robomise, ktorej predlohou je webová aplikácia rovnakého názvu. Natívna aplikácia ponúka lepšie využitie cieľového zariadenia.

1.1 Motivácia

Mobilná aplikácia pre zariadenie pracujúce na mobilnej platforme Google Android, vytvorená v rámci tejto bakalárskej práce, je určená na rozvoj algoritmického myslenia. Cieľovou skupinou môžu byť deti, ktoré sa chcú naučiť programovať, ale aj samotní programátori, ktorých touto formou skúša ich algoritmické znalosti na pracovnom pohovore. Kvôli prvej koncovej skupine má aplikácia formu hry. Sú v nej využívané adaptabilné algoritmy. Pre deti to je zábavnejšie a učia sa rýchlejšie ako pri hrách, kde sú ďalšie levely generované automaticky.

Aplikáciu začal vyvíjať tím Adaptabilného učenia na Fakulte informatiky Masarykovej univerzity, a to prostredníctvom webového používateľského rozhrania, ktoré je primárne ovládané myšou. Hlavným účelom mobilnej aplikácie Robomise je vytvorenie nového pou-

1. Úvod

živatelského rozhrania, ktoré je prispôsobené pre ovládanie dotykmi. Sekundárnym účelom mobilnej aplikácie je testovanie použiteľnosti webovej a mobilnej aplikácie Robomise.

1.2 Ciele práce

Cieľom tejto práce je používateľským testovaním zistiť slabé a silné stránky predlohovej webovej aplikácie Robomise, na základe ktorých sa navrhne nová natívna mobilná aplikácia. Tento cieľ práce je rozdelený do niekoľkých menších cieľov.

- a) Prvým cieľom je analýza a testovanie používateľskej štruktúry celého systému súčasnej webovej aplikácie.
- b) Druhým cieľom je pomocou všetkých získaných poznatkov naprogramovať novú mobilnú aplikáciu na platforme Google Android. Táto mobilná aplikácia bude čiastočne zastupovať súčasnú webovú aplikáciu. Aplikáciu budú používať hlavne používatelia, ktorí si chcú otestovať znalosti programovania alebo sa naučiť základné algoritmy. Prispôsobená bude hlavne na tablety s dostatočne veľkým displejom.
- c) Tretím cieľom je používateľsky otestovať natívnu aplikáciu.

1.3 Štruktúra práce

Text práce je rozdelený do niekoľkých kapitol, ktoré odpovedajú menším cieľom. V druhej kapitole je popísaná súčasná webová aplikácia a používateľské testovanie na nej. Kľúčovou súčasťou druhej kapitoly je zoznam požiadaviek na novú aplikáciu. V tretej kapitole je popísaný návrh a implementácia novej mobilnej aplikácie. V záverečnej kapitole je zaznamenané používateľské testovanie na natívnej aplikácii a uvedený zoznam možných budúcich rozšírení.

2 Používateľské testovanie webovej aplikácie

Je veľmi dôležité odstrániť problémy, ktoré by mohli používateľa frustrovať alebo donútiť ho opustiť aplikáciu. Používateľské testovanie je metóda, ktorá pomáha odhaliť, ako používatelia používajú produkt. Vďaka tomu je možné odhaliť nedostatky a problémy aplikácie. K týmto nedostatkom dochádza kvôli tomu, že vývojari, dizajnéri a ostatní ľudia, ktorí sú spoluautormi produktu, nie sú zároveň aj jeho používateľmi.

Princípy používateľského testovania, jeho prípravu a realizáciu, ktoré boli použité v tejto bakalárskej práci, opisuje diplomová práca Mgr. Evy Vaškovej [3].

Steve Krug prirovnáva používateľské testovanie k návšteve cudzincov vo vašom meste. Mnohokrát sa totiž pri prevádzaní používateľa aplikáciou začnú ukazovať veci, ktoré si predtým tvorcovia neuvedomili. Taktiež to pomáha uvedomiť si, že niektoré veci nemusia byť novému používateľovi zrejmé, napriek tomu, že nám pripadajú úplne prirodzené.[4]

2.1 Webová aplikácia Robomise

Pred vytvorením testovacieho scenára 2.2.1 sa s produktom musí osoba, ktorá ho zostavuje, vopred zoznámiť a pochopiť jeho funkcionálnu. Táto podkapitola predstavuje rozhranie testovaného softvéru a predstavuje základné pojmy, s ktorými aplikácia pracuje. Webovú aplikáciu Robomise¹ naprogramoval tím Adaptabilného učenia na Fakulte informatiky Masarykovej univerzity. Webová aplikácia Robomise je naprogramovaná v reacte. Implementačné detaily si môžete naštudovať v diplomovej práci Mgr. Tomáša Effenbergera[5].

2.1.1 Princíp hry

Webová aplikácia Robomise je hra, ktorej cieľom je hravým spôsobom naučiť používateľa algoritmicky rozmýšľať. Študent sa učí programovať riešením úloh. Preto sa pri používateľskom testovaní tester zameria hlavne na časť aplikácie, v ktorej sa úloha rieši.

1. <https://robomise.cz>

Hra má tému vesmíru a úlohou hráča je dostať vesmírnu loď do cieľovej rovinky. Počas letu nemôže vesmírna raketa naraziť do asteroidu a musí pozbierať všetky diamanty v hracom poli. Aby to nebolo úplne jednoduché a aby sa hráči naučili aj zložitejšie príkazy, ako sú napr. cykly a podmienky, sú niektoré úlohy obmedzené počtom príkazov, ktoré môžu použiť.

Používateľ sa tak vďaka hre naučí všetky základné programovacie koncepty: postupnosť príkazov, podmienky, cykly. Pri spustení programu tlačítkom Spustiť hráč vidí, ako sa raketka pohybuje a ktorý príkaz to spôsobuje, a tak pri neúspešnom doletení do cieľa používateľ vie, na ktorý príkaz sa má sústrediť pri oprave.

2.1.2 Hracia plocha

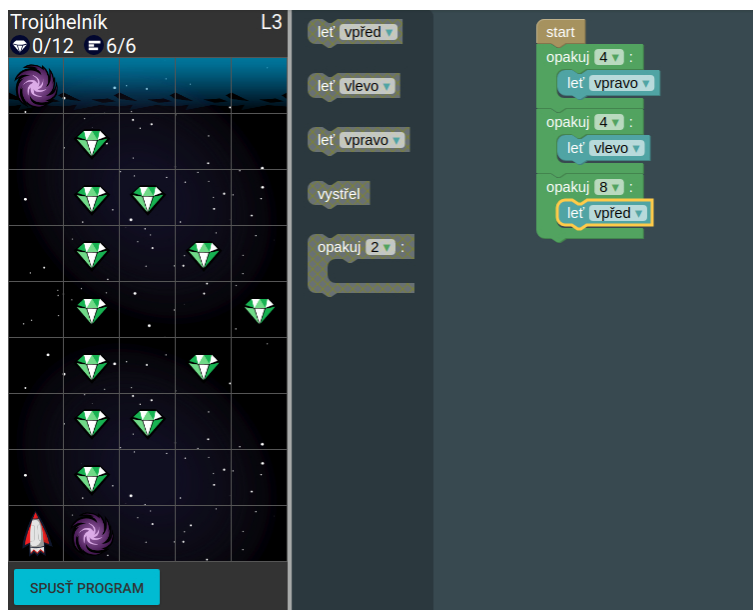
Hracia plocha, v ktorej sa vesmírna loď pohybuje, je mriežka, kde každé pole má svoju farbu, na základe ktorej sa môže raketka rozhodnúť zmeniť smer. Každé pole môže mať okrem farby aj hrací objekt. Ako napríklad diamant, ktorý musí raketka pozbierať, asteroid, ktorému sa musí vyhnúť, meteorid, ktorý môže zničiť výstrelom, a červiu dieru, ktorá slúži ako teleport. Raketka začína v dolnom riadku mriežky a každým príkazom sa posunie o jeden riadok hore.

2.1.3 Úlohy a program

Každá úloha je riešená skladaním blokov príkazov, ako môžeme vidieť na obrázku 2.1. Tieto bloky presúva používateľ pomocou myši z kontajnera dostupných blokov a pripája ich za blok Štart. Náročnosť úloh sa postupne zvyšuje a s narastajúcou úrovňou pribúdajú aj nové bloky, reprezentujúce príkazy programu. Hra obsahuje viac ako 80 úloh, rozdelených do 9 levelov.

2.2 Príprava testovania použiteľnosti

Príprava a plánovanie testovania použiteľnosti je jednou z najdôležitejších častí celého procesu testovania použiteľnosti, pretože bez prípravy môže dôjsť k úplne zbytočným výsledkom. Účastníci testovania musia vykonávať reálne úlohy. Musia to byť tie úlohy, ktoré by skutoční užívatelia s produktom naozaj robili.



Obr. 2.1: Příklad řešení úlohy hry Robomise

2.2.1 Testovací scénár

Pre používateľské testovanie bol vytvorený jeden scenár so 6 úlohami. Tie boli zamerané na hraciu časť aplikácie. Teda či je hráč schopný sa zorientovať v hre, manipulovať s príkazovými blokmi, prejsť úlohou alebo si v zozname úloh vybrať inú úlohu ako mu ponúka systém.

Testovací scénár sa skladá z týchto častí:

- Prvou časťou scenára bolo predstavenie osoby, ktorú testovací účastník bude reprezentovať. Nakoniec nebol potrebný, pretože testovanie prebiehalo s potenciálnymi používateľmi, ktorí sa nepotrebovali vcítiť do cudzej role.
- Druhou časťou boli otázky týkajúce sa zázemia účastníkov, aby sme výsledky testovania mohli posúdiť na základe pozície a predošlých skúseností používateľa.
- Treťou časťou bolo samotné testovanie.

2. POUŽÍVATEĽSKÉ TESTOVANIE WEBOVEJ APLIKÁCIE

- Poslednou úlohou testovaného účastníka bolo ohodnotiť hracie prostredie a prípadne navrhnúť zmeny, pri ktorých by sa mu hrou prechádzalo lepšie.

Testovacie úlohy boli pre potreby testovania napísané v slovenčine a ich presné znenie je uvedené v prílohe.

2.2.2 Testovací účastníci

Pri testovaní bol vždy prítomný iba jeden účastník a moderátor, aby bolo dosiahnuté tých najlepších výsledkov. Viacerí účastníci naraz totiž podliehajú dynamike skupiny a niektorí účastníci tak nemusia vysloviť svoj názor alebo postupne konvergujú do spoločného názoru, čo v konečnom dôsledku vedie k nepresným dátam [6]. Testovanie len s jedným účastníkom v tom istom čase umožní účastníkovi viac sa uvoľniť, prezentovať svoje skutočné názory a moderátorovi umožní určiť, či používateľ dokáže so systémom pracovať samostatne.

Ideálnym počtom testovaných účastníkov pre jedno testovanie použiteľnosti je päť.[7] Preto sa užívateľského testovania Robomise zúčastnil práve tento počet. Pri výbere účastníkov som sa zamerala na jednu skupinu potencionálnych používateľov, a tou sú programátori. Táto skupina už základné príkazy, ktoré sú používané v aplikácii, pozná. Tým pádom sa účastníci viac sústredia na hrácké prostredie, manipuláciu s príkazmi a ďalšie rušivé elementy pri hre. Je nepravdepodobné, že by nedokončili úlohu z dôvodu, že nevedia správne poskladať príkazy alebo použiť cyklus. Programátorov som vybrala z troch rôznych firiem: SledovániTV, SychrovNET a ModernTV.

2.3 Priebeh testovania

Po príprave testovania sa plynule prešlo na samotné vykonávanie testovania. To prebiehalo v účastníkovom prirodzenom prostredí, na jeho pracovisku. Každý účastník bol vopred oboznámený s nahrávaním testovania. Testovanie prebehlo bez závažnejších problémov. Po skončení každého testovania bol spracovaný záznam do textovej podoby, ktorá bola podkladom pre vyhodnotenie testovania.

2.4 Vyhodnotenie testovania

Po testovaní sa prešlo do etapy spracovania dát. Dáta boli zaznamenané hlavne počas testovania, po testovaní sa doplnilo iba pár drobností z nahrávok. Tieto výstupy sú dostupné v textovej prílohe tejto práce.

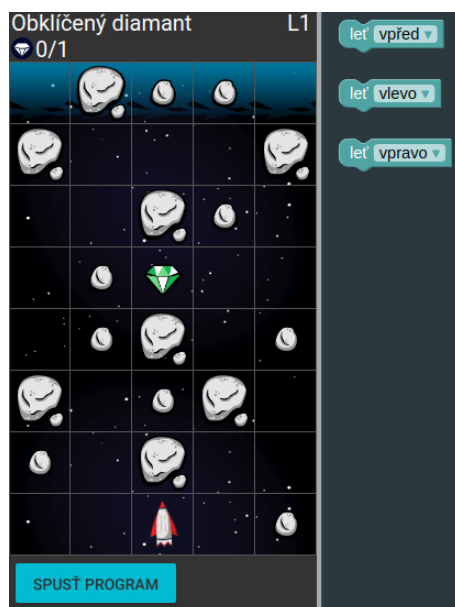
2.4.1 Vyhodnotenie dát

Zo spracovaných dát vyplynulo, že účastníci pochádzajú z rôznych pracovných pozícií. Najväčšie zastúpenie mali frontend programátori, ďalej sa testovania zúčastnil backend developer a UX designér. Ani jeden z nich sa predtým nestretol s podobným typom hry, kde by mohli pomocou blokov reprezentujúcich príkazy ovládať panáčka. Vo väčšine prípadov boli milo prekvapení z tohto grafického programovacieho jazyka.

Tým, že sa pred testovaním účastníci nestretli so žiadnou podobnou hrou na rozvíjanie algoritmického myslenia, mali odlišné očakávania:

- Prvý účastník očakával tlačítka áno a nie, na základe ktorých bude riešiť algoritmické problémy.
- Druhý tester predpokladal, že uvidí veľa obrázkov a grafov, pri ktorých bude musieť logicky myslieť.
- Tretí účastník očakával veľký problém, v ktorom bude musieť nájsť menšie podproblémy a tie ďalej vyriešiť.
- Štvrtý čakal pythonovskú korytnačku, ktorá bude podľa príkazov kresliť obrázky.
- Poslednému testerovi sa podarilo odhadnúť hru spomedzi účastníkov najviac. Ten si hru predstavoval ako kocky, ktoré sa budú do seba spájať a na základe ich správneho spojenia bude v hre postupovať do ďalších levelov.

Nasledujúca časť sa zaoberá detailným rozborom najväčších problémov, ktoré boli na základe pozorovania odhalené. Každému účastníkovi sa podarilo prejsť všetkých šesť úloh. Počas testovania nenarazili



Obr. 2.2: Úloha Obklíčený diamant

na nič, čo by im bránilo v dokončení úlohy. Nemali problém nájsť plný zoznam úloh, problém im nerobilo ani nájsť začiatok programu, do ktorého mali myšou pretahovať bloky. Niektoré úlohy boli pre testerov náročnejšie a zabralo im viac času ich vyriešiť, ale vo výsledku ich vyriešili bez toho, aby som musela zasiahnuť do testovania.

Na druhej strane som u účastníkov vypožadovala problém s pochopením príkazov: vľavo, vpravo. Napríklad pri úlohe s názvom Obklíčený diamant (viď Obr. 2.2) každý testovací účastník začal skladať príkazy v poradí: štart, vpravo, vpred, vpravo, vpred. Pri posledom príkaze sa zastavil a oznámil mi, že sa úloha nedá vyriešiť. Potom stlačil tlačidlo Spustiť program, aby zistil, ako sa bude vesmírna loď správať. Vďaka tomu, že hra zvyrazňuje príkaz, kvôli ktorému sa raketka pohla, účastníci pochopili, že príkazom Vľavo sa vesmírna loď pohla o jedno pole vyššie a o jedno pole doľava. Potom dokázali správne navoliť príkazy tak, aby nenarazili do asteroidu a získali diamant.

Druhým častým problémom, s ktorým sa potýkali 4 účastníci testovania, bolo určenie pozície, v ktorej má raketka zastaviť. Tester

predpokladali, že vesmírna loď má preletieť celým hracím poľom. Takže pri už spomínanej úlohe s názvom Obklúčený diamant zadali 9 príkazov vrátane bloku Štart, pričom príkazov je potreba práve 8. Teda o jeden menej, aby raketka ostala na cieľovej čiare, ktorú reprezentuje modrá farba.

Ďalším problémom, kvôli ktorému nevedelo ľahko vyriešiť úlohu niekoľko testerov, bolo neporozumenie končiacim podmienkam. Vo vyšších leveloch môže raketka zmeniť smer na základe farby, ktorá je pod ňou. Takže napríklad raketka lieta v cykle stále rovno, pokiaľ pod ňou nie je modrá farba. K tomuto problému dochádzalo, pretože som testovacích účastníkov prostredníctvom testovacieho scenára prinútila otvárať úlohy z vyšších levelov, kvôli čomu preskočili vysvetlivky, ktoré túto funkcionálnosť popisujú.

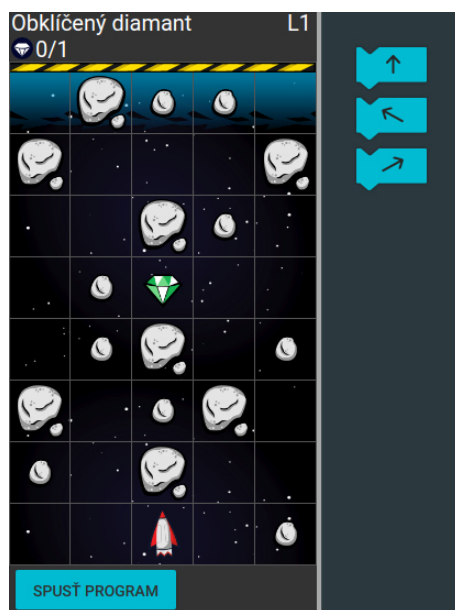
Tretiemu účastníkovi neprišlo intuitívne rozbaľovanie úloh levelu. Takže pri piatej úlohe testovacieho scenára najskôr nevedel, ako spustiť úlohu, ktorá je v cyklickej výzve, po bližšom preskúmaní komponenty na to ale prišiel. Druhý účastník si vybral úlohu, ktorá mala obmedzený počet príkazov, čo je znázornené nad hracou mriežkou ikonou 2.1, a nerozumel, prečo mu bloky stmavli a nemôže ich ďalej používať.

V poslednej fáze testovania účastníci hodnotili aplikáciu na škále od 1 (najhoršie hodnotenie) do 7 (najlepšie hodnotenie). Priemerom hodnotenia piatich testerov je 5, čo ukazuje, že používatelia boli s rozhraním viac spokojní ako nespokojní. Pri otázke, či by si vedeli túto hru predstaviť používať, dvaja odpovedali kladne, jednému vadilo detské prostredie, jeden sa na to pozeral ako na nástroj, ako pri pohovoroch vyfiltrovať ľudí bez analytického myslenia. Poslednému sa nepáčilo užívateľské prostredie.

2.4.2 Návrhy riešení

Po uvedení a rozobraní problémov nasleduje časť návrhov možných riešení, ktorými by bolo možné tieto nedostatky eliminovať.

1. Najväčším problémom testovacích účastníkov boli pre nich nepochopené príkazové bloky: Vpravo, Vľavo. Účastníci navrhli zmeniť názov na Diagonálne vpravo a Diagonálne vľavo. Tohoto názvu ľahko porozumejú programátori, ale obávam sa, že 12 ročné deti by s takýmto názvom mali rovnaký problém ako



Obr. 2.3: Vylepšená úloha Obklúčený diamant

predtým testovací účastníci. Preto navrhujem použiť namiesto slov značky: ↖ a ↗. Obr: 2.3

2. Problém určenia, kedy má raketka zastaviť, môžeme vyriešiť pridaním praporku, alebo žltó-čiernej výstražnej pásy na konci hracieho poľa 2.3, aby používateľ vedel, kde hra končí a kde už nemôže ísť ďalej.
3. Rozbaľovanie úloh jednotlivých levelov sa dá jednoducho nastaviť, aby bolo defaultne otvorené. Treba však otestovať, či to vyhovuje aj ostatným používateľom.
4. Ďalším návrhom na vylepšenie aplikácie bolo umiestniť legendu príkazov niekde na okraj hry.
5. Programátori sú zvyknutí písať kód v angličtine, preto navrhli pridať možnosť zmeny jazyka aplikácie. V aktuálnej verzii, pokiaľ sú pod českou doménou, musia skladať bloky cyklov: dokud, pokud. Ak by si používateľ mohol zmeniť jazyk na an-

glický, presúval by bloky: while, do-while, ktorým programátor lepšie rozumie.

6. Posledný zaujímavý návrh spočíval v pridaní úvodného videa na domovskú stránku, v ktorom by autori vysvetlili budúcemu hráčovi princíp a bodovanie hry.

Z užívateľského testovania vyplynuli dve hlavné požiadavky na novú natívnu aplikáciu. Prvou požiadavkou je zmeniť bloky vpravo, vľavo. Druhou je zvýrazniť koniec hracieho poľa. Navrhla som vzhľad takto upraveného hracieho pola. Názornú ukážku môžete vidieť na obrázku 2.3. Na mobilnej aplikácii treba tento návrh otestovať.

3 Mobilná aplikácia

Nasledujúca kapitola v prvej časti popisuje technológiu použitú pri vývoji novej natívnej aplikácie. V ďalších častiach sa venuje implementačným postupom a funkcionalitám.

3.1 React a React Native

React Native je javascriptový framework pre tvorbu multiplatformových mobilných aplikácií. To znamená, že umožňuje vývojárom vytvárať aplikácie pre iOS i Android. Za jeho vznikom a neustálym vývojom stojí Facebook, Inc. [8] Je založený na javascriptovej knižnici React, ktorá slúži na tvorbu užívateľských rozhraní webových stránok.

React-Native nevyužíva na vykreslenie komponenty služby HTML tagov ako React. Namiesto nich sa používajú špeciálne komponenty poskytované frameworkom. Napríklad namiesto HTML značky `div` (3.1) sa v Native používa komponenta `View` (3.2).

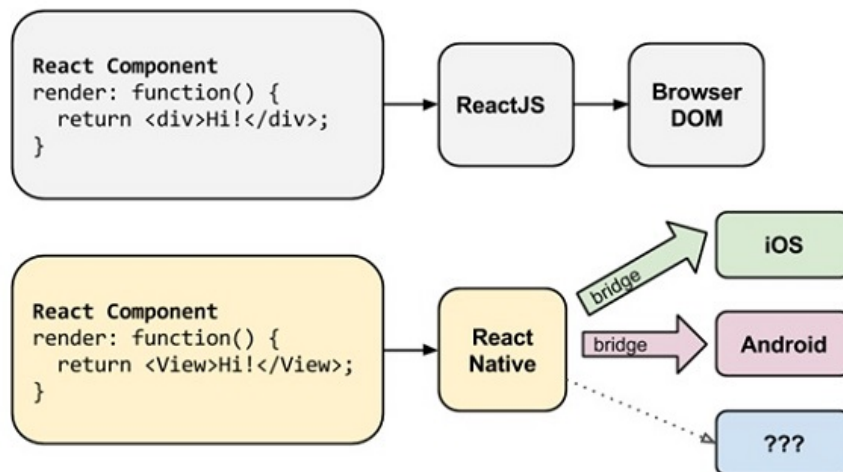
```
1 { const element = (  
2   <div>  
3     <h1>Hello , world!</h1>  
4   </div>  
5 ) ;}
```

Kód 3.1: Základná React komponenta

```
1 { const element = (  
2   <View>  
3     <Text>Hello , world!</Text>  
4   </View>  
5 ) ;}
```

Kód 3.2: Základná React-Native komponenta

V každej aplikácii typicky bežia dva typy vlákien – JavaScript vlákno a natívne vlákna. Natívne vlákna pozostávajú z hlavného vlákna a niekoľkých vlákien bežiacich na pozadí. JavaScript vlákno je špecifické pre React Native. Jeho úlohou je vykonávať javascriptový kód, ktorý sa stará o business logiku aplikácie v samostatnom



Obr. 3.1: Schéma architektúry React Native a porovnanie s ReactJS. Zdroj:[9]

JavaScript engine¹. Tieto dva typy vlákien spolu nikdy nekomunikujú priamo a ani sa navzájom neblokujú. Je medzi nimi akýsi most (z angl. bridge), ktorý je jadrom React Native.

3.2 Základná kostra projektu

Ako som už spomínala v sekcii 2.1, webová aplikácia je napísaná s použitím Reactu. Mobilná verzia hry Robomise bude naprogramovaná v React-Native. Vďaka tomu, že React aj React-Native sú javascriptové frameworky, môžem znova použiť časť webového kódu. V tejto práci budem písať hlavne o odlišnostiach týchto dvoch implementácií.

Prvý súbor, ktorý sa zavolá, sa volá `Index.js`. V tomto súbore sa nachádza `AppRegistry` trieda, ktorá registruje hlavnú komponentu celej aplikácie.

React-Native aplikácia je zložená z komponent. Komponenta reprezentuje celú obrazovku alebo jej časť. Každá komponenta môže byť zložená z dcérskych komponent. Pre svoje fungovanie potrebuje dáta.

1. JavaScript engine je program alebo interpret, ktorý spúšťa javascriptový kód

Predávanie dát si je možné predstaviť dvoma spôsobmi, pomocou props a state.

Props si je možné predstaviť ako vstup funkcie. Komponenta prijme vstup (props), spracuje ho a zobrazí výstup, nesmie však meniť svoje props [9].

Na dáta, ktoré sú premenlivé, sa používa state. Ten predstavuje stav, v ktorom sa objekt aktuálne nachádza. Komponenta môže svoj stav inicializovať a následne ho aktualizovať kedykoľvek potrebuje.

Napríklad chceme v aplikácii nemenný blikajúci text. V tomto prípade je text props, inicializuje sa iba raz a ostáva nemenný. To, či je text viditeľný alebo nie, sa mení časom. Preto je hodnota vypnutý, zapnutý uložená v state.

Podľa toho, či komponenta pracuje iba s props alebo aj so state, rozdeľujeme komponenty na prezentačné (stateless) a kontajnery (stateful).

Prezentačné komponenty sa zaoberajú tým, ako veci vyzerajú. Popisujú, ako ma vyzeráť UI. Prijímajú props z komponent, ktoré ich používajú a na ich základe vykresľujú obsah. V mobilnom projekte sú tieto komponenty v zložke components.

Hlavnou úlohou kontajnerov je starať sa o logiku zobrazovania. Môžu obsahovať prezentačné komponenty, ale aj ďalšie kontajnery. Slúžia ako zdroje dát, ktoré ďalej predávajú svojim komponentám. Kontajnery sú dostupné v zložke containers.

Okrem vyššie spomínaných zložiek sa v mobilnom projekte nachádza súbor package.json, v ktorom je zoznam použitých knižníc, a zložka pages, ktorá obsahuje kontajnery reprezentujúce celé obrazovky. Napríklad súbor PractisePage.js predstavuje celú hráčku plochu hry Robomise. Obsahuje SpaceGameContainer, v ktorom sa raketa pohybuje, a BlocklyEditorContainer, v ktorom používateľ spája príkazové bloky.

3.3 Navigácia

Mobilná aplikácia nie je zložená iba z jednej obrazovky. Preto je potrebné implementovať logiku, ktorá určí kedy a za čo vymeniť jednu obrazovku za druhú. Pre navigáciu medzi obrazovkami (pages) je

3. MOBILNÁ APLIKÁCIA

použitá knižnica `react-router-native` ². Túto knižnicu som vybrala pre jej intuitívnosť, schopnosť držať si históriu prechodov a podobnosť s webovou navigáciou. Alternatívnou voľbou bola rozšírená knižnica `react-navigation`.³

```
1 <NativeRouter>
2   <AppContainer>
3     <Switch>
4       <Route exact path="/" component={TasksTableContainer}/>
5       <Route exact path="/tasks" component={TasksTableContainer}/>
6       <Route path="/task/:taskId" component={PracticePage}/>
7     </Switch>
8   </AppContainer>
9 </NativeRouter>
```

Kód 3.3: Router v `index.js`

Použitie routeru v mobilnej aplikácii môžete vidieť v kóde 3.3. Route je najdôležitejšia komponenta knižnice. Jej zodpovednosťou je vykresliť komponentu podľa aktuálnej path ⁴. Ak sa v kóde 3.3 path rovná `/tasks`, router vykreslí komponentu `TasksTableContainer`. Ak sa cesta zmení na `/task/:taskId`, router vykreslí komponentu `PracticePage`.

`Switch` slúži na vykreslenie prvého dcérskeho Route, ktorého path je rovnaká ako aktuálna path ⁵.

Link poskytuje navigáciu hlbšie v aplikácii. Pomocou parametra `to:String` mení aktuálnu path.

3.4 Správa stavov

Užívateľ môže svojím vstupom ovplyvňovať časti aplikácie. Zachovávať konzistenciu a reagovať na tieto zmeny môže byť náchylné k chybám. Tento problém sa snažia vyriešiť state management knižnice, ktoré využívajú jednosmerný tok dát.

2. Dostupná z <https://www.npmjs.com/package/react-router-native>

3. <https://facebook.github.io/react-native/docs/navigation>

4. <https://reacttraining.com/react-router/native/api/Route>

5. <https://reacttraining.com/react-router/native/api/Switch>

Pre správu stavov mobilnej aplikácie bola zvolená knižnica `react-redux`⁶, a to hlavne pre jej veľkú podporu react vývojárov. Táto knižnica funguje na princípe globálneho úložiska dát. Toto úložisko je pre aplikáciu jediným zdrojom dát. Tie predáva komponentám. Tieto dáta sú nemenné a pokiaľ ich chceme zmeniť, musíme ich prepísať inou hodnotou.

Komponenty posielajú akcie do reduxových reducerov, tie na základe informácií z danej akcie zmenia reduxové úložisko. Akonáhle sa zmenia dáta v reduxovom úložisku, komponenta, ktorej redux tieto dáta predáva, na tieto zmeny reaguje.

3.5 Api

Mobilná aplikácia nemá dáta, ako napríklad zoznam úloh, zabudované priamo v sebe. Pre prístup k zdrojom používa REST API⁷, ktoré je dostupné na adrese: `https://robomise.cz/learn/api/`. Funkcie, ktoré pracujú s API, sa nachádzajú v zložke `api`.

Na HTTP požiadavky⁸ je použitá knižnica `axios`⁹. Jej hlavnou výhodou je vykonávanie automatického transformovania JSON¹⁰ dát[10]. V aplikácii sú použité dve asynchrónne metódy `axios.get()`, ukážka v kóde 3.4, a `axios.post()`, ukážka v kóde 3.5.

```
1  async get(urlPath) {
2    let options = await this.getRequestOptions()
3    let ret = await this.axios.get(urlPath, options)
4    return ret
5  }
```

Kód 3.4: Metóda `get` v `api.js`

```
1  async post(urlPath, data) {
2    let options = await this.getRequestOptions()
3    let ret = await this.axios.post(urlPath, data, options)
```

6. <https://react-redux.js.org/>

7. Representational State Transfer Application Programming Interface

8. HTTP requests

9. <https://www.npmjs.com/package/axios>

10. JavaScript Object Notation

3. MOBILNÁ APLIKÁCIA

```
4   return ret
5 }
```

Kód 3.5: Metóda post v `api.js`

Vyššie uvedené metódy sa potom používajú vo funkciách v `/api/index.js`. Napríklad v ukážke 3.6 sa používa vyššie spomenutá metóda `get`, ktorou v tomto prípade získame informácie o hracích plochách hry.

```
1 export function fetchWorld(url) {
2   let api = Api.getInstance()
3   return api.get(url).then(response => response.data);
4 }
```

Kód 3.6: Funkcia `fetchWorld`

Problémom bolo, že, aj napriek korektne implementovaným HTTP požiadavkam, mi pri post metóde server vracal: chybový kód 403¹¹.

Problémom bol chýbajúci CSRF¹² token, ktorým sa zaisťuje bezpečnosť severu, aby posielali post požiadavky len autorizovaní klienti.

Najskôr som pomocou webových stránok¹³ zistila, ako zahrnúť CSRF token do všetkých HTTP požiadavok. Ukážka priradenia v kóde 3.8.

```
1 Axios.defaults.headers.common['X-CSRF-TOKEN'] = token;
```

Kód 3.7: Priradenie tokenu do `axios`

Potom som chcela využiť elegantné riešenie¹⁴, ktoré bolo založené na použití knižníc `axios-cookiejar-support` a `tough-cookie`. Bohužiaľ som stále naspäť dostávala chybový kód 403. Pri použití výpisu do konzoly som zistila, že token je naďalej prázdny.

Všimla som si, že v `response`¹⁵ požiadavky `get` sú `headers`, ktoré obsahujú hodnotu žiadaného CSRF tokena. Ten som pomocou `re-`

11. Forbidden error - Nepovolený prístup

12. Cross-site Request Forgery

13. <https://github.com/axios/axios/issues/2024>

14. Dostupné na <https://github.com/axios/axios/issues/943#issuecomment-378219600>

15. Odpoveď

gexu¹⁶ získala a priradila do `Axious.defaults`. Kód 3.8.

```
1 let x = await this.axios.get("/", this.requestOptions);  
2 let regex = /(?!<=csrftoken=)\w+(?!=;)/;  
3 this.token = regex.exec(x.headers["set-cookie"])[0][0]
```

Kód 3.8: Získanie CSRF tokenu

Pre overenie som znova vypísala token do konzoly. Tentokrát bol už správne nastavený, ale server mi stále vracal kód 403. Potom som prišla na to, že je potreba nastaviť ešte `Referer`¹⁷. Po jeho nastavení na adresu `https://robomise.cz` post požiadavka prebehla úspešne.

3.6 Obrázky a Štýly

16. Regulárneho výrazu

17. URI, z ktorej bola webová stránka navštívená

Bibliografia

1. *Forecast and Trends* [online]. Cisco, 2018 [cit. 2018-11-07]. Dostupné z: <https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html>.
2. *Reports* [online]. Statcounter, 2018 [cit. 2018-11-07]. Dostupné z: <https://statcounter.com/demo/reports/>.
3. VAŠKOVÁ, Eva. *Testování uživatelského prožitku nástroje pro HA Cluster* [online]. 2018 [cit. 2019-05-10]. Diplomová práce. Masarykova univerzita, Fakulta informatiky, Brno. Vedúci práce Marek GRÁC.
4. KRUG, Steve. *Nenuťte uživatele přemýšlet!: praktický průvodce testováním a opravou chyb použitelnost*. Computer Press. 2010.
5. EFFENBERGER, Tomáš. *Adaptabilní systém pro výuku základů programování* [online]. 2018 [cit. 2019-05-10]. Diplomová práce. Masarykova univerzita, Fakulta informatiky, Brno. Vedúci práce Radek PELÁNEK.
6. *Checklist for Planning Usability Studies* [online]. Hoa Loranger, 2016 [cit. 2019-05-07]. Dostupné z: <https://www.nngroup.com/articles/usability-test-checklist/>.
7. *Why You Only Need to Test with 5 Users* [online]. Jakob Nielsen, 2010 [cit. 2019-05-07]. Dostupné z: <https://www.nngroup.com/articles/task-scenarios-usability-testing/>.
8. *Facebook Open Source*. Facebook, Inc., 2019. Dostupné tiež z: <https://opensource.facebook.com/>.
9. *Writing Cross-Platform Apps with React Native*. Bonnie Eisenman, 2016. Dostupné tiež z: <https://www.infoq.com/articles/react-native-introduction>.
10. *HTTP requests using Axios*. Flavio Copes, 2018. Dostupné tiež z: <https://flaviocopes.com/axios/>.