Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Bakalářská práce

Vývoj mobilní aplikace pro NRZP ČR pro iOS a Android

Jakub Dašek

© 2024 ČZU v Praze

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Jakub Dašek

Informatika

Název práce

Vývoj mobilní aplikace pro NRZP ČR pro iOS a Android

Název anglicky

Development of a mobile application for the NRZP ČR for iOS and Android

---

Cíle práce

Navrhnout a vyvinout mobilní aplikaci pro iOS a Android pro Národní radu osob se zdravotním postižením. Aplikace bude primárně sloužit k přehledu užitečných informací a informování o novinkách vydaných předsedou NRZP ČR. Aplikace by měla být přístupná a přizpůsobitelná pro uživatele s různými druhy postižení, barvoslepým a nevidomým.

Metodika

Práce sestává ze dvou částí, teoretické a praktické.

V rámci praktické části práce bude provedeno studium relevantních informačních zdrojů a prozkoumány potřeby uživatelů se zdravotním postižením.

Praktická část práce bude spočívat v návrhu a implementace aplikace pro potřeby NRZP. Bude provedena identifikace klíčových funkcí, které by aplikace měla poskytovat, jak z hlediska přizpůsobitelnosti pro nejrůznější postižení, tak k identifikaci nejdůležitějších funkcí.

S ohledem na přístupnost a přizpůsobitelnost bude proveden návrh uživatelského rozhraní, přičemž bude využito konzultací s odborníky z RNZP. Následně bude provedena implementace aplikace pomocí frameworku React. Aplikace bude cílena na použití na mobilních zařízeních s OS Android a iOS.

Výsledná aplikace bude prakticky nasazena a otestována, budou shrnuty poznatky z jejího vývoje a navrženy možnosti dalšího možného budoucího rozšíření či úprav.

Doporučený rozsah práce

35-40 stran

Klíčová slova

React, mobilní aplikace, přístupnost, iOS, Android, programování, postižení

Doporučené zdroje informací

https://react.dev/
https://sd.blackball.lv/library/Learning_React_(2020).pdf
https://www.igi-global.com/book/user-centered-software-development-blind/218295
https://www.thinkmind.org/download.php?articleid=achi_2012_2_50_20125

Předběžný termín obhajoby

2023/24 LS – PEF

Vedoucí práce

Ing. Jiří Brožek, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 28. 11. 2023

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 9. 2. 2024

doc. Ing. Tomáš Šubrt, Ph.D.

Děkan

V Praze dne 14. 03. 2024

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Vývoj mobilní aplikace pro NRZP ČR pro iOS a Android" jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 15.3.2024 _____

Poděkování

Rád bych touto cestou poděkoval vedoucímu práce, panu Ing. Jiřímu Brožkovi, Ph.D. za vedení bakalářské práce. Jeho věcné a rychlé připomínky byly pro mě významnou pomocí k dokončení této práce. Děkuji také Mgr. Sabrině Pliskové, MPA, ředitelce odboru komunikace NRZP ČR za její čas a konzultace a za příležitost zpracovat práci na téma, které by mohlo být přínosným.

Vývoj mobilní aplikace pro NRZP ČR pro iOS a Android

Abstrakt

Práce si dává za úkol vytvořit prototyp mobilní aplikace pro Národní radu osob se zdravotním postižením České republiky (NRZP ČR), která by nabídla přehled aktualit a důležitých informací pro osoby se zdravotním postižením co nejpřístupněji a nejjednodušeji. V teoretické části byla rozebrána tématika různých očních postižení a obecné přístupnosti mobilních aplikací. Dále se práce zabývá tématikou různých přístupů k vývoji mobilních aplikací, která vybírá nejvhodnější řešení k vývoji aplikace. V praktické části byl představen vývojový proces od požadavků na vývoj až po podrobný popis kódu React Native, který zajišťuje esenciální funkce pro chod aplikace. Následuje pohled na výsledné uživatelské rozhraní a design celé aplikace. Aplikace byla otestována a vznikl funkční prototyp, který demonstruje možnost využití aplikace pro zlepšení dostupnosti informací pro osoby se zdravotním postižením.

Klíčová slova: React Native, přístupnost, iOS, Android, mobilní aplikace, programování, postižení, NRZP ČR

Development of a mobile application for the NRZP ČR for iOS and Android

Abstract

The objective of this bachelor thesis is to create a prototype mobile application for the National Council of People with Disabilities of the Czech Republic (NRZP ČR), which would provide an overview of all news and important information to people with disabilities in the most accessible and simplest way possible. The theoretical part discusses the topics of various visual impairments and the general accessibility of mobile applications. Furthermore, the work explores different approaches to mobile app development and, based on that, picks the most suitable software to create the app. The practical part introduces the development process from requirements and a detailed description of the React Native code that ensures essential functions for the applications' operation. This is followed by an overview of the final user interface and the design of the entire application. Finally, the app is tested, and a functional prototype, demonstrating the possibility of using the application to inform people with disabilities, is created.

Keywords: React Native, Accessibility, iOS, Android, mobile application, programming, people with disabilities, NRZP ČR

Obsah

1 Úvod

Rozvoj moderních technologií má primárně za cíl zjednodušení, či zvýšení efektivity jakékoli lidské činnosti, a tedy zlepšení kvality života běžné populace. Pro osoby se zdravotním postižením, je obsluha těchto zařízení náročnější, někdy i neřešitelným problémem. Přitom právě tito lidé – lidé se zdravotním postižením – potřebují nejvíce pomoci kompenzovat sníženou schopnost pohybu a orientace. Pokud se vozíčkář včas dozví o Euroklíči, bude vědět, kde může bez problému zaparkovat. Pro osoby se zdravotním postižením je informovanost stěžejní.

Národní rada osob se zdravotním postižením ČR zastupuje zájmy zdravotně postižených. Jedním z důležitých úloh je vydávání informací předsedy Národní rady osob se zdravotním postižením, kterými informuje osoby se zdravotním postižením o důležitých změnách v mnoha aspektech života týkající se jejich života.

Tato práce popisuje práci a funkci Národní rady osob se zdravotním postižením a poskytuje pohled na to, proč je informovanost osob se zdravotním postižením důležitá a jaké jsou momentálně kanály poskytující tuto službu. Dívá se také na přístup k vývoji s ohledem na různá postižení zraku a poskytuje základní přehled doporučení k tvorbě přístupné aplikace.

Práce dále nabízí přehled a srovnání klasických i moderních technologií pro tvorbu mobilních aplikací. Poskytuje elementární přehled a vysvětlení pojmů a technologií potřebných k vývoji multiplatformní aplikace pomocí softwaru React Native.

Praktická část práce se zaobírá požadavky na funckionalitu a uživatelské rozhraní aplikace. Všechny funkce a základní komponenty aplikace jsou vysvětleny spolu s ukázkami kódů samotné aplikace. Výsledné uživatelské rozhraní a funkce jsou názorně ukázány a popsány.

Práce má za cíl informační činnost Národní rady osob se zdravotním postižením rozšířit a vytvořit prototyp aplikace, která osobám se zdravotním postižením poskytne tyto informace rychle a jednoduše. Bez toho, aniž by museli otevírat prohlížeč a chodit na webové stránky.

2 Cíl práce

Cílem práce je vytvořit prototyp aplikace, která bude jednoduše a rychle informovat osob se zdravotním postižením o změnách v legislativě a dalších aktualitách, které se jich týkají. Všechny nové aktuality se zobrazí jako notifikace v systému. Aplikace je vyvíjena s ohledem na různé druhy postižení, aby byla přístupná co největším počtu osob.

Finálním produktem je prototyp aplikace, který demonstruje, jak umožňuje osobám se zdravotním postižením snadný a rychlý přistup k informacím, které vydává Národní rada osob se zdravotním postižením ČR, jenž momentálně chybí. Cílem je také vytvořit mobilní aplikaci, která dodržuje zásady přístupné mobilní aplikace.

2.1 Metodika

Práce zkoumá činnost NRZP ČR a na základě toho nabízí řešení, které může doplnit a rozvinout část jejího působení. Pomocí relevantních zdrojů zkoumá základní principy vývoje přístupné aplikace. Dále je provedeno bádání v oblasti přístupu k modernímu vývoji mobilních aplikací a základy technologií potřebné k tvorbě aplikace.

Je proveden přehled všech požadavků, které jsou kladeny na výsledný prototyp aplikace. Pomocí získaných poznatků jsou vyvíjeny všechny funkce aplikace s ohledem na získané vědomosti o přístupnosti aplikací. Během celého vývoje je aplikace testována na zařízení s operačním systémem iOS. Pro zajištění správného chodu je aplikace otestována i na platformě Android.

3 Příprava na vývoj aplikace pro OZP pro NRZP ČR

První část této kapitoly se zabývá Národní radou osob se zdravotním postižením České republiky (dále NRZP ČR), jejími okruhy zájmů, projekty a jejím přínosem. Následně se zaobírá různými druhy zrakového postižení, a jak přistupovat při tvorbě aplikace, aby byla optimálně přístupná co nejširšímu okruhu lidí. Dále zkoumá aktuální přístupy k vývoji aplikací a poskytuje přehled a základy technologií, které budou k vývoji aplikace potřeba.

3.1 Národní rada osob se zdravotním postižením ČR

NRZP ČR je zastřešující nezisková organizace, která od roku 2000 zastupuje zájmy osob se zdravotním postižením při jednáních se státními a veřejnými institucemi. Těchto osob je v ČR přes 1 milion. NRZP ČR je celostátní organizace a v každém kraji má své profesionální pracoviště [1].

Organizace zejména:

- Poskytuje registrované sociální služby dle zákona č. 108/2006 Sb., se zaměřením na sociální poradenství, cílem je pomoc s řešením nepříznivých situací, do kterých se mohou dostat lidé v důsledku neznalosti svých práv a povinností.
- Tvoří podmínky pro podporu a inkluzi sociálně znevýhodněných skupin občanů.
- Vytváří zázemí pro veřejně prospěšné aktivity občanů se zdravotním postižením.
- Napomáhá k tvorbě souhrnných opatření a služeb, které pomáhají osobám se zdravotním postižením návrat či udržení se na trhu práce.
- Realizuje akce a kampaně, které obecně napomáhají ke zlepšení kvality života osob se zdravotním postižním.
- Vydává časopisy, informační materiály a publikace [2].

3.1.1 Ukázka projektů NRZP ČR

Mezi konkrétní projekty pořádané NRZP ČR patří například Inkluzivní začleňování osob se zdravotním postižením a speciálně vzdělávacími potřebami, který se zabývá osvětou a poradenstvím technických opatření. Díky tomuto projektu došlo například k přidáním

bezbariérových spojů na území Jihočeského kraje či zlepšení parkovacích místo pro osoby se zdravotním postižením (dále OZP) [3].

Dalším projektem je Informační činnost NRZP ČR k podpoře vyrovnávání příležitostí OZP, který má za cíl zvyšovat povědomí v oblasti diskriminace z důvodu zdravotního postižení či nepříznivého zdravotního stavu a podporuje obecnou informovanost v oblasti vyrovnávání příležitostí pro OZP. To realizuje skrze různé akce, časopis MOSTY nebo web NRZP ČR [3].

3.1.2 **Aktuality NRZP ČR**

Jedním z důležitých aspektů, které se nachází ne webu NRZP ČR jsou aktuality o změnách v legislativě, nových novelách, návrzích, a dalších informacích důležitých pro OZP. Tyto informace jsou vydávány téměř denně předsedou NRZP ČR Mgr. Václavem Krásou. Tyto aktuality jsou obrovským přínosem pro OZP. Na webu se jich momentálně nachází více než 2410 a neexistuje žádný obdobný projekt, který by takto pravidelně informoval osoby se zdravotním postižením. Pro ukázku je uvedeno několik aktualit [4]:

- 30. 11. 2023 – předseda Krása informoval o hlasování poslanecké sněmovny o zařazení bodu o příspěvku na péči [5].
- 13. 12. 2023 – tisková zpráva a žádost o zlepšení přístupnosti Výtoňského železničního mostu [6].
- 13. 11. 2023 – pan Krása radí a informuje, jak platit a komunikovat ceny energií pro držitele průkazů ZTP a ZTP/P [7].
- 7. 9. 2023 – informace o velké novele zákona o sociálních službách, kde se projednával například bod „posílení role sociální práce na ORP“ [8].

Z těchto ukázek je zřejmé, že rozsáhlost a škála informací je obrovská a unikátní v tom, kolik informací z různých aspektů života OZP nabízí.

3.1.3 **Informovanost osob se zdravotním postižením**

Informovaností OZP se v ČR zabývá právě NRZP ČR. Mezi další instituce a portály poskytující informace OZP patří například web Vlády ČR, na kterém nalezneme spíše kontakty odkazující se na mezinárodní organizace zabývající se různými typy postižení [9].

Dále se dají nalézt informace na webech krajů ČR, například stránky Jihočeského kraje [10]. Dále také v Národním zdravotnickém informačním portále [11]. Ty se ale zabývají spíše obecnou informovaností a právy těchto osob. NRZP ČR je unikátní v tom, že poskytuje nejnovější informace o všech změnách v legislativě, novelách a další relevantních informací pro OZP [1].

Kromě již zmíněných informací předsedy, NRZP také vydává tištěný časopis, kde se kromě důležitých informací tito lidé dozvědí o tom, jak správě cvičit či relaxovat, nebo jako se na problematiku lidí s postižením dívají vlivní a úspěšní lidé, kteří činí rozhodnutí, která ovlivňují každodenní život lidí s postižením [1].

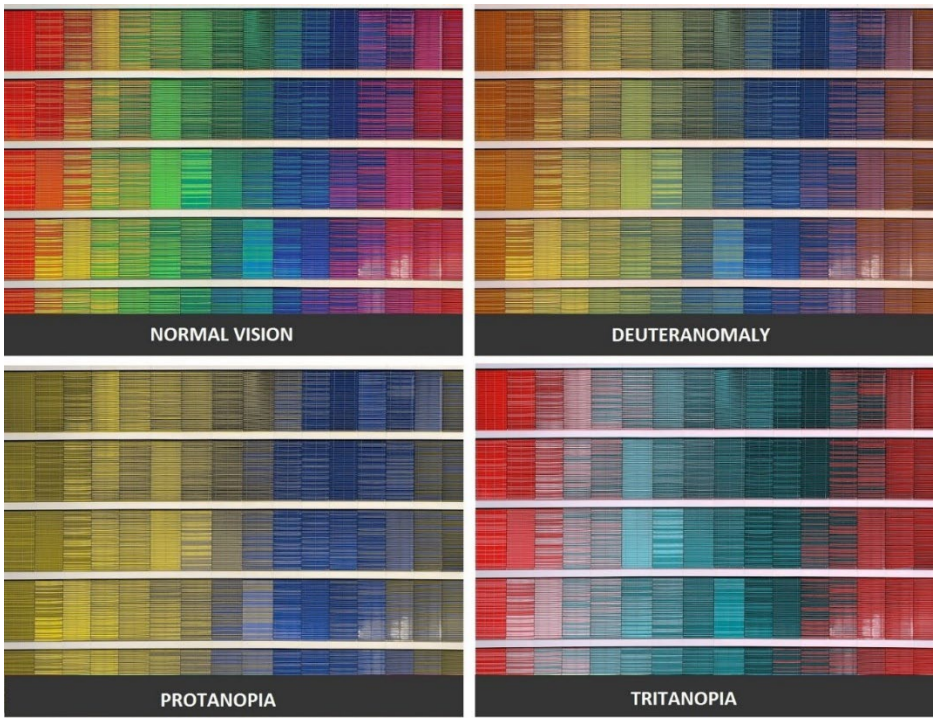
Národní rada osob se zdravotním postižením ČR hájí zájmy a práva těchto lidí a je střechovou organizací pro 97 organizací a spolků lidí se zdravotním postižením. Mezi její hlavní pilíře mimo legislativu a poradenství patří právě informování lidí s postižením [1]. Pokud se k nim informace o nové legislativě, nových bezbariérových trasách, odborných seminářích či poradenských centrech nedostanou, nemají žádnou hodnotu.

3.2 Zrakové postižení s ohledem k přístupnosti aplikací

3.2.1 Porucha barvocitu a barvoslepost

Porucha barvocitu, často v češtině nesprávně zaměňováno za barvoslepost, není neschopnost vidět barvy [12]. Jde o postižení, odborně nazývané daltonismus, které trápí přibližně 8,5 % lidí na světě. U můžu je tato porucha mnohem častější než u žen, vyskytuje se u 1 z 20. U žen touto poruchou trpí jen 1 z 200 [13].

Člověk s tímto postižením má problém rozeznat rozdíl mezi barvami a jejich odstíny.
Existuje několik typů poruchy barvocitu, nejčastějším je deuteranopie, kdy postižený nevnímá zelenou barvu a protanopie, kdy nevnímá červenou barvu. Třetím, méně častým, je tritanopie, kde je problém s modrou a žlutou barvou. Pro lepší představu přikládám obrázek, jak jsou vnímány barvu pro osoby bez zrakového postižení (vlevo nahoře) a lidmi s poruchou barvocitu [12].



Obr. 1: Poruchy barvocitu *[14]*

3.2.2 **Slabozrakost**

Slabozrakost je porucha vidění, jenž se dělí na typy podle závažnosti:

- Ztráta centrálního vidění – špatná schopnost vidět objekty v centru zraku.
- Noční slepota – špatná schopnost vidění za nízkého světla.
- Ztráta periferního vidění – špatná schopnost zřít objekty na okrajích zorného pole.
- Rozmazané a nejasné vidění [15].

3.2.3 **Nevidomost**

Nejvážnější postižení zraku, slepota nebo nevidomost je stav, při kterém nemocný nevnímá světlo [16].

Dělíme na:

- Praktická slepota
- Úplná slepota

U osob s nevidomostí je téměř nemožné ovládat zařízení jako je telefon bez další asistence. Je tedy potřeba, aby byla aplikace správně naprogramována a čtečka správně četla informace, které se zobrazují [16].

3.2.4 **Ostatní oční vady**

Ostatními očními vadami jako jsou například refrakční vady (krátkozrakost, dalekozrakost, šilhání a další). Většina z nich se dá často léčit nebo alespoň korigovat [17]. Je možné poskytnout několik funkcí, jak aplikaci lépe zpřístupnit osobám s očními vadami.

3.2.5 **Jak pomoci osobám se zrakovým postižením s přístupností aplikace?**

Symboly

Jde o zástupný znak, který může doprovázet např. zpětnou vazbu systému, jakým je například úspěšné dokončení, error, upozornění atp. Velmi často se setkáváme, že aplikace nás upozorní pouze například zezelenáním rámečku. To ale nestačí pro osoby s poruchou barvocitu. Tento problém se dá vyřešit právě pomocí symbolů jako je například křížek či výstražný symbol. Lepším řešením je k symbolům přidat i popis [18].

Podtržený text

U hypertextových odkazů je vhodné nezapomínat na podtržení textu. Pokud je text pouze oddělen barevně, může být nečitelný pro osoby s poruchou barvocitu. Také je dobré nepojmenovávat odkazy slovem „odkaz“, pokud čte obsah čtečka, přečetla by slovo odkaz dvakrát. Vhodné je i vytvoření dostatečné mezery mezi odkazy. Když je u sebe více podtržených odkazů najednou, stává se text špatně čitelným všem uživatelům [19].

Popisky

Popisky je třeba používat u všeho, co je na obrazovce znázorněno jako obrázek, symbol či barevné zvýraznění. Doprovodný popis umožní uživatelům informaci rozeznat, i když trpí poruchou barvocitu nebo využívají hlasové asistence [18].

Označení povinného pole

Velmi podobné jako popisky či symboly. Při formulářích je nutno symbolem či popisem označit povinné pole a nespoléhat se pouze na barevné označení. [12].

Zvýraznit důležitá tlačítka

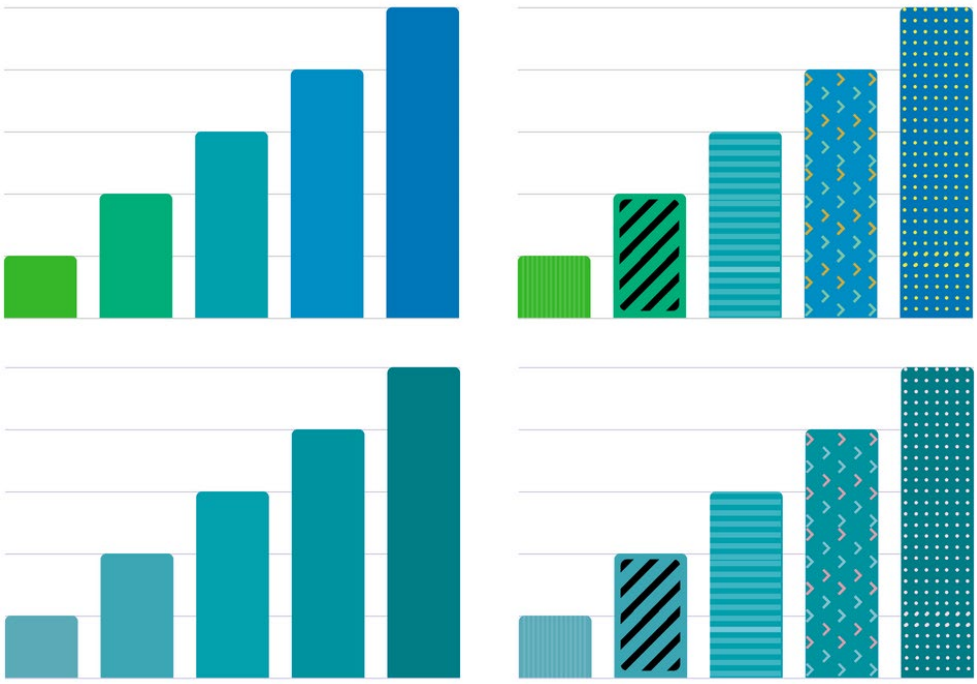
V případě výskytu více tlačítek blízko sebe, je vhodné to důležitější zvětšit. Další možností je například tlačítka odlišit kontrastem, použit jinou velikost či tloušťku písma, aby byl uživatel naveden do cílového místa [12].

Kontrast

Při práci s kontrastem jde o poměr kontrastu mezi pozadím a popředím. Maximální možný kontrast je 21:1 (černá a bílá). Při vývoji přístupné aplikace je nejmenší možný kontrast 4,5:1 u textu a u odkazů 3:1. V případě nedodržení tohoto pravidla se stává text špatně čitelným [20].

Vzory a textury

Pro odlišení velkých barevných bloků či grafů je vhodným řešením použít různě vzory a textury. Jeden blok může mít například diagonální pruhy, druhý by byl čtvercovaný, třetí by obsahoval kruhy atp. Na obrázku 2 představuje první řádek to, jak vidí graf osoba bez zrakového postižení. Druhý řádek představuje, jak vidí graf osoba s tritanopií [12].



Obr. 2: Grafy s texturami – normální zrak (první řádek) x tritanopie (druhý řádek) *[12]*

3.3 Vývoj aplikace

Tato kapitola se zabývá pojmy jako nativní a multiplatformní vývoj a dojde k vyhodnocení, který přístup je vhodný pro aplikaci NRZP ČR. Dalším pojmem je hybridní aplikace a s tím spojený webový vývoj aplikací (HTML, CSS a JavaScript) a knihovna React. Všechny tyto pojmy a technologie jsou klíčové pro porozumění prostředí, ve které bude aplikace vyvíjena.

3.3.1 Nativní vývoj

Při nativivním vývoji se využívá nativního přístupu ke zvolenému zařízení – vyvíjíme software přímo pro konkrétní platformu. Aplikace tedy komunikuje přímo s celým souborem nástrojů pro vývoj softwaru (SDK).

Aplikace a funkce zařízení (přístup k GPS, kameře, přístup k souborům aj.) tedy komunikují přímo se zařízením a není mezi nimi žádná další vrstva (nadstavba). Toto je zároveň i hlavní výhodou nativního vývoje, vyplývají z toho další výhody jako:

- Rychlost aplikace – software využívá celý výkon zařízení,
- Zabírá méně paměti – díky absenci nadstavby pro vývoj.
- Jednodušší a rychlejší reakce na aktualizace OS.
- Přístup k nativním funkcím [21].

Naopak největší nevýhodou je logicky vývoj pouze pro jeden systém, pro iOS i Android musíme využít jiného prostředí i programovacích jazyků z toho vyplývají tyto hlavní nevýhody:

- Dražší a delší vývoj.
- Znalost několika jazyků a prostředí.
- Náročnější testování a údržba – i když velmi dovře reaguje na aktualizace, je pořád potřeba hlídat a udržovat verze na všech platformách zvlášť [21].

3.3.2 Webový vývoj

Při webovém vývoji není potřeba programovat aplikace přímo na míru platformy. Zároveň jde o způsob, díky kterému lze aplikaci spustit na jakékoliv platformě, která podporuje webový prohlížeč a je připojena k internetu. Jsou psány, stejně jako webové stránky, pomocí technologií HTML, CSS a JavaScript.

Dříve byla největším problémem absence nativních funkcí. To dnes lze vyřešit například pomocí progresivní webové aplikace, které umožňují číst a zapisovat soubory na zařízení, poskytuje přístup k Bluetooth aj. Dokonce zvládne do určité míry pracovat i offline. Má to ale své omezení a stále chybí větší podpora u prohlížečů Mozilla Firefox a Safari. Jedná se o velice dobré řešení pro nenáročné aplikace. [22]

3.3.3 **Hybridní vývoj**

Hybridní aplikace je velmi podobná klasické webové aplikaci, na rozdíl od ní má ale přístup k operačnímu systému a hardwaru zařízení, stejně jako aplikace při nativním vývoji. Slučuje tedy výhody (i nevýhody) nativního i hybridního přístupu [23]. Hybridní aplikace jsou vyvíjeny, podobně jako webové, skrze technologie a jazyky HTML5, CSS a JavaScript, vytvořený kód je ale následně zarámován do platforem jako je například Apache Cordova, která už se stará, o komunikaci aplikace a dané olatformy, stále ale běží ve webovém prohlížeči zařízení [24].

3.3.4 **Multiplatformní vývoj**

Při multiplatformním vývoji je vytvářena aplikace, jež je cílena na více platforem. To samozřejmě zvládne již zmíněný webový nebo hybridní přístup, rozdíl je ale v tom, že multiplatformní aplikace fungují na nativním kódu dané platformy.

Multiplatformní vývoj je je sice nejnovějším přístupem, avšak v poslední době stoupá na popularitě a čím dál více startupů a firem vyvíjí aplikaci právě tímto způsobem [25].

Výhody multiplatformního vývoje:

- Je jednoduché zasáhnout velký počet osob. Díky možnosti spustit aplikaci na různých platformách, je jednoduší dostat aplikaci ke všem osobám, nehledě na jejich operačním systému.
- Je to levnější. Jednoduše je levnější napsat kód jednou a pak ho spustit všude, anglický koncept „write once, run everywhere“.
- Sjednocený design – na obou platformách je uživatelské rozhraní (dále i jako UI – User Interface) stejné a jsou tedy schopni ovládat aplikaci na každém zařízení stejně.
- Je to rychlejší. Psát jen jeden kód je samozřejmě nejen levnější, ale i rychlejší [26].

Nevýhoda multiplatformního vývoje:

- Častější problémy s chodem aplikace. Jelikož je kód psán obecně pro všechny platformy, problémy jsou častější než u nativního vývoje [26].
- Menší kontrola nad specifickými požadavky [27].

Multiplatformní vývoj se rozhodně nehodí do každého vývoje, jak je se možné dočíst ve zdroji [27], kde multiplatformní vývoj nesplňoval specifické požadavky a nákroky firmy na aplikaci. Avšak v případě vývoje aplikace pro NRZP je potřeba dostat co nejjednotnější a co nejvíce přístupnou aplikaci co se týče designu, tak i k dostání na různá zařízení a platformy. Proto je multiplatformní vývoj v tomto případě vhodným řešením.

3.4 Technologie potřebné pro vývoj v React Native

V dalších kapitolách jsou popsány technologie často využívané k vývoji multiplatformních aplikací. Konkrétně se zaměřuje na vývoj v softwaru React Native, který byl vybrán jako ideální zástupce multiplatformního vývoje pro účely aplikace pro NRZP ČR.

3.4.1 HTML

Zkratka HTML (HyperText Markup Language) znamená česky hypertextový značkovací jazyk. Používá se pro tvorbu webových stránek, v dnešní době se používá verze HTML 5, která byla vydána roku 2008 [28].

HTML kód obsahuje tagy/značky, které určují, jakou text bude mít formu a jak se bude kód zobrazovat. Tagy jsou buď párové nebo nepárové. Mezi párové patří například tag `<h1></h1>`, uvnitř této značky by se umístil hlavní nadpis stránky nebo `<body></body>`, který je tělem stránky - mezi tagy body se umísťuje jakýkoliv obsah. Vše, co je od začátku do konce tagu, například `<h1>Hlavní nadpis</h1>` se nazývá elementem.

Příkladem nepárového tahu je <img>, což je značka pro obrázek.

Ukázka jednoduchého HTML kódu:

```
<!DOCTYPE html>
<html>
     <head>
          <title>Název stránky</title>
     </head>
     <body>
          <h1>NRZP ČR</h1>
          <p>NRZP má na starost<p>
     </body>
</html>
```

Tento kód zobrazí stránku s názvem “Název stránky” a v oknu prohlížeče bychom viděli nadpis s hodnotou NRZP ČR a pod ním text: NRZP má na starost [29].

3.4.2 CSS

CSS – kaskádové styly (anglicky Cascading Style Sheets), je technologie, která v praxi doplňuje HTML. Stará se o vzhled a rozvržení HTML kódu, přes CSS tedy můžeme měnit například barvu textu, font, velikost, rozmístit odstavce do sloupců a mnoho dalších pokročilejších úprav [30].

Aplikovat CSS lze například umístěním tagů `<style></style>` do HTML tagu `<head></head>` a následně mezi tag `<style></style>` vložit například značku odstavce p a do složených závorek umístit atributu, vypadalo by to následovně:

Ukázka jednoduchého HTML a CSS kódu:

```
<!DOCTYPE html>
<html>
     <head>
          <style>
               p {
               color: blue;
               }
          </style>
          <title>Název stránky</title>
     </head>
     <body>
          <h1>NRZP ČR</h1>
          <p>NRZP má na starost<p>
     </body>
</html>
```

Nyní bychom dostali stejnou stránku jako v kapitole o HTML s tím rozdílem, že by byl text „NRZP má na starost“ modrý.

3.4.3 **JavaScript**

JavaScript (dále i jako JS) je programovací jazyk, který se nejčastěji používá jako jazyk skriptovací právě při tvorbě webů. [31] Při tvorbě webů se skript – program psaný v JS může psát přímo do HTML kódu.

Ukázka kódu:

```
<!DOCTYPE html>
<html>
     <body>
          <h1>NRZP ČR</h1>
          <p>Zde se zobrazí normální text</p>
          <p id="demo"></p>
          <script>
          document.getElementById("demo").innerHTML  =  "Text
          JavaScript!";
          </script>
     </body>
</html>
```

V tomto kódu se kromě nadpisu a odstavce psaného v HTML zobrazí jako druhý odstavec „Text JavaScript!“, díky skriptu, který volá element s id demo [32].

JavaScript se ale dá používat i na straně serveru, například jako Node.js, při tvorbě aplikace ho tedy nevyužijeme pouze jako skriptovací jazyk pro psaní aplikace na straně klienta, ale i na straně serveru. Na JS existují různé knihovny, které usnadňují a zefektivňují vývoj webových aplikací. Jde například o Vue.js, Angular či React. [33].

3.4.4 **Node.js**

Node.js je multiplatformní Javascriptové prostředí, které umožňuje spouštět kód JS i mimo webový prohlížeč. Využívá se tedy hlavně ke spuštění kódu JS na straně serveru. Umí tedy například generovat dynamický obsah na stránce, otevřít, číst a psát soubory na straně serveru, sbírat data či přidávat, mazat a měnit data v databázi [34].

3.4.5 **React**

React je knihovna pro web a uživatelské rozhraní. Používá značkovací syntaxi JSX. Jak takový kód vypadá je možno vidět na následující ukázce:

```
const user = {
  name: 'Hedy Lamarr',
  imageUrl: 'https://i.imgur.com/yXOvdOSs.jpg',
  imageSize: 90,
};

export default function Profile() {
  return (
    <>
      <h1>{user.name}</h1>
      <img
        className="avatar"
        src={user.imageUrl}
        alt={'Photo of ' + user.name}
        style={{
          width: user.imageSize,
          height: user.imageSize
        }}
      />
    </>
  );
}
```
[35]

I když je syntaxe trochu pozměněna, stále je vidět základ HTML, CSS i JS. Skript by nám zobrazil obrázek 3:

Obr. 3: Výstup kódu React *[35]*

Komponenty

Komponenty jsou jedním ze základních konceptů – tvoří základ UI Reactu a později i React Native, který z něj vychází [36].

Komponenta je stavěcí blok uživatelského rozhraní. Podívejme se na tento HTML kód:

```
<article>
  <h1>My First Component</h1>
  <ol>
    <li>Components: UI Building Blocks</li>
    <li>Defining a Component</li>
    <li>Using a Component</li>
  </ol>
</article>
```

Tento kód by se dal v Reactu zapsat jednoduše jako `<TableOfContents />`, který by mohl být následně použit kdekoliv na stránce. Komponentu je třeba definovat, to se děje pomocí JS, následně je potřeba komponentu importovat, a nakonec exportovat na místě, kde je žádoucí ji zobrazit. Výsledek je tedy již v syntaxi nazývané JSX.

JSX Syntaxe

JSX je syntaxové rozšíření JS, umožňuje psaní značkování podobné HTML uvnitř JS souborů. Na první pohled tedy spojuje HTML a JS kód dohromady, ve skutečnosti se ale nejedná o syntaxi HTML ale právě již JSX syntaxi [37].

V čem se tedy liší JSX od HTML a jaká má pravidla:

1. Zabalení elementů
    a. Pokud je třeba zobrazit více než jeden element, je žádoucí ho zabalit do jednoho rodičovského tagu.
    b. Takový tag může být buď `<div></div>` nebo prázný tag `<> </>`, takzvaný fragment.
2. Uzavírání tagů
    a. V JSX musí být všechny tagy uzavřené. I nepárový tag se musí uzavřít.
    b. Například tag `<img>` se mění na `<img />`, u párových tagů musí být uzavírací tag také, `<li>Položka seznamu</li>`.

3. Velbloudí notace
    a. Velbloudí notace není vyžadována, ale je vhodné ji dodržovat. Při tvorbě proměnných není možné využívat pomlčky a podobné znaky, jelikož stále pracujeme s JS.
    b. Místo class bychom tedy například použili className.

React sám na svých webových stránkách nabízí konvertor z HTML do JSX, který může kdokoliv zdarma použít: https://transform.tools/html-to-jsx.

3.4.6 **React Native**

Kombinací nativního vývoje a knihovny React, je právě React Native. React Native je software sloužící pro vytváření mobilních aplikací pro operační systémy iOS i Android. Při psaní kódu je tedy aplikace spustitelná hned z obou operačních systémů [38].

Jak funguje React Native?

React Native nezobrazuje pouze webovou aplikaci, ale skládá se z nativních komponent platformy, pro kterou aplikaci navrhujeme. Narozdíl od Reactu obsahuje nové komponenty jako je <Text></Text> či <View></View>, tagy z HTML zde tedy uplatnit nelze. Každá komponenta je tedy existující nativní komponenta, a to je následně renderována do UI. To vše ale za použití JavaScriptu [38].

Architektura

Architektura React Native lze rozdělit do dvou částí. První z nich je tzv. Rendering v češtině se používá výraz renderování. To je proces, který se stará o vytváření a aktualizování UI. Další termíny nejsou oficiálně přeloženy do češtiny a jsou tedy uváděny v angličtině s vysvětlením. Druhou částí jsou Build Tools (jakýkoliv script nebo software vytvořen pro programovací jazyk), který se stará o co nejnižší čas spuštění, spravuje používání paměti a velikost aplikace [39].

Rendering

Rendering dále dělíme na 5 částí:

1. Fabric: Jedná se o nový systém renderování, základním principem je sjednotit renderovací logiku C++ a zlepšit kompatibilitu z různými hostujícími platformami (Andoid, iOS aj. [40].)
2. Render, Commit and Mount: Tento renderovací proces se stará o to, aby se logika Reactu dostala na hostující platformu. To se nazývá Render Pipeline, česky Renderovací roura [41].
3. Cross-Platform Implementation: Je multiplatformním řešením algoritmu, který pomocí C++ komunikuje s Reactem a hostující platformou [42].
4. View Flattening: Tento algoritmus slouží k zamezení renderování „hlubokých uspořádaných stromů" – deep layout trees. Funguje například tak, že pokud existuje komponenta, která obsahuje obrázek, který je zapuštěn v další komponentě, tak v případě, že obsahuje například pouze margin – odsazení, tak s tím pracuje React jako s jednou komponentou. To znovu zamezí zbytečnému používání procesoru [43].
5. Threading Model: Jedná se o model rozdělení načítání API z JavaScriptu na několik vrstev, které zajišťují bezpečné renderování [44].

Základní a nativní komponenty

Pro zobrazení komponenty na iOS i Androidu je využíván takzvaný view, což je jednoduchý blok uživatelského rozhraní. Určitý čtverec či jiný obrazec na obrazovce (sám o sobě nezobrazí nic), který zvládne zobrazit text, obrázky nebo reagovat na uživatelský vstup. Tyto views se tedy starají o zobrazení všeho, co je v aplikaci vidět [45].

Nativní komponenty

Při vytváření aplikace se mohou již zmíněné views zobrazit na Androidu pomocí jazyků Kotlin či Java, pro iOS šlo zase o Swift nebo Objective-C. S React Native je ale možné vyvolat views s JS pomocí React component. Při běhu vytváří React Native komponenty korespondující pro Android i iOS views komponenty. React Native přichází s novými komponenty a nevyužívá stejné komponenty jako React. [45].

Základní komponenty

React native má několik základních komponent (anglicky Core Components). Mezi ně patří:

- <View>: Jedná se o blokový element, který lze přirovnat k HTML tagu ve kterém nejde scrollovat (rolovat - nelze přejet dolů nebo nahoru, jedná se statický obraz).
- <Text>: Zobrazí řetězce textu podobně jako HTML tag <p>.
- <Image>: Zobrazí různé typy obrázků, v HTML <img>.
- <ScrollView>: Blokový element, ve kterém lze scrollovat a může zobrazit několik komponent, v HTML .
- <TextInput> Umožní uživateli zadat text, v HTML <input type="text"> [45].

Hooks

React Native Hooks umožňuje využívat stavy komponenty bez nutnosti vytvářet třídy. Jedná o novinku představenou ve verzi 16.8, jejímž hlavním přínosem je větší jednoduchost ve sdílení stavové logiky a celkové složitosti kódu. Mezi základní hooks patří například useState, useEffect či useContext. Praktickým využitím hooks je například použití při nastavování různých stylů měnících se na základě určité inicializace globálně, v naši práci jsou přímo využity pro nastavení světlého a tmavého motivu napříč aplikací [46].

API

Application Programming Interface (API), česky aplikační programové rozhraní, je sada pravidel a specifikací, umožňující komunikaci různých zařízení mezi sebou. V kontextu aplikací a internetových služeb obecně se používá také REST API, které se dají použít v kontextu React Native a knihovny Axios pro práci s HTTP požadavky [47].

Axios

Axios je Hypertext Transfer Protocol (HTTP) klient pro node.js a webový prohlížeč. Ze serverové části node.js používá standartní http požadavky a na části webové používá XMLHttpRequests, které umožňuje webovým aplikacím komunikaci prostřednictvím protokolu HTTP. [48] V praxi se dá použít například pro automatickou implementaci článků/stránek z webové stránky a jejich následné zobrazení do aplikace.

Expo Go

Expo Go je tzv. sandbox, česky doslovně přeloženo jako pískoviště, který slouží k vývoji a testování na softwaru React Native. Aplikace Expo Go lze stáhnout na Android i iOS skrze Google Play a App store. Aplikace vyvíjeny v tomto rozhraní neslouží k následné distribuci, nabízí ale otevřený a jednoduchý prostor, kde může vývojář jednoduše postavit prototyp aplikace. I proto se používá označení pískoviště [49].

Tento přehled poskytl základní porozumění technologií a funkcí, které React Native nabízí a které budou použity v tvorbě aplikace v praktické části práce.

4 Proces tvorby aplikace

Tato kapitola se zabývá řešením vývoje aplikace, od použitých nástrojů, rozvržení a jednotlivých funkcionalit. Jsou zobrazeny všechny obrazy, které aplikace nabízí. Následně je aplikace otestována a optimalizována pro platformu Android. Na konci kapitoly je shrnutí toho, co bylo dosaženo a co je potřeba v budoucnu dodělat, aby mohla být aplikace spuštěna.

4.1 Nastavení vývojového prostředí

Jak už bylo zmíněno v teoretické části práce, k vývoji byl použit softwarový framework React Native (dále také jako RN). Pro nastavení vývojového prostředí bylo potřeba nainstalovat Node.js, jelikož RN používá JavaScript. Pro psaní kódu byl použit volně stažitelný editor Visual Studio Code (VS Code).

Pro vývoj byla použita již zmíněná platforma Expo Go, pro zobrazení aplikace bylo potřeba také stáhnout aplikaci Expo Go z Apple Store pro iOS nebo Google Play pro Android.

V momentě, kdy byl nainstalován Node.js, byl skrze terminál ve VS Code spuštěn příkaz: npx create-expo-app NázevAplikace. Tím se nám vytvořil nový projekt se všemi důležitými soubory. Ujistili jsme se, že se nacházíme v adresáři s právě vytvořeným projektem a spustíme projekt příkazem npx expo start, v nové verzi by stačilo pouze npx expo.

Tím bylo nastaveno vývojové prostředí. Pro zobrazení vyvíjené aplikace v mobilu stačilo projekt otevřít naskenováním QR kódu, který byl vygenerován v terminálu.

4.2 Struktura aplikace

Na začátek bylo potřeba definovat, jaké funkce bude aplikace obsahovat, a co by měla zobrazovat. Níže je výpis všech obrazovek a funkcí, které systém zobrazí, a jejich krátký popis.

WelcomeScreen.js

Jedná se o první obrazovku, kterou uživatel uvidí po načtení aplikace.

Požadavky na funkcionalitu:

- Zobrazení loga NRZP ČR.
- Uvítací text do aplikace.
- Zobrazí se pouze při prvním spuštění.
- Možnost prvotního nastavení velikosti textu a motivu (tmavý/světlý).
- Možnost uložení nastavení a pokračování do aplikace.

HomeScreen.js

- Domovská obrazovka, která se zobrazí po nastavení ve WelcomeScreen a při dalším spuštění jako první obrazovka.
- Obsahuje hlavičku s logem a menu.
- Obsahuje nejdůležitější upozornění/aktualitu a další 3 poslední aktuality s možností přejít na danou aktualitu a možnost otevřít stránku se všemi aktualitami.
- Je přístupná z menu.

NewsScreen.js

- Obrazovka zobrazuje všechny aktuality.
- Aktuality si volá na základě Axios API z webové stránky.
- Při přidání nové aktuality je odeslána uživateli notifikace.
- Je přístupná z menu.

AboutScreen.js

- Obrazovka zobrazuje základní informace o Národní radě osob se zdravotním postižením České republiky.

- Je přístupná z menu.
- Zobrazuje partnery NRZP ČR.

SettingsScreen.js

- Obrazovka zobrazuje nastavení textu a motivu.
- Umožňuje uživateli nastavit velikost textu a barvu motivu a následně uložit.
- Je přístupná z menu.

InfoScreen.js

- Obrazovka zobrazuje v plném rozsahu vybranou informaci/aktualitu.
- Je přístupná z domovské obrazovky.

Obrazovky, které nejsou samotné obrazovky, ale jsou esenciální či doplňují ostatní obrazovky aplikace (hlavička, scrollování aj.):

App.js

- Esenciální část aplikace, v každém RN projektu obsahuje hlavní komponentu, která se renderuje při startu aplikace.
- Obsahuje import, navigaci všech obrazovek a hlavičku aplikace.
- Hlavička obsahuje logo NRZP ČR a ikonu menu.

CustomDrawerContent.js

- Je přístupná z hlavičky skrze ikonu menu.
- Komponenta zobrazuje položky menu:
    - Hlavní stránka
    - Aktuality
    - Nastavení
    - O NRZP ČR

CustomScrollView.js

- Komponenta zobrazuje posuvník ke scrollování (rolování) v aplikaci.

- Jedná se o vlastní komponentu, která se přizpůsobuje velikosti obrazovky a je implementovatelná skrz všechny obrazovky.

TextSizeContext.js

- Komponenta obsahuje logiku změny velikost textu v aplikaci.
- Je nastavitelná skrze obrazovky FirstSetupScreen a SettingsScreen, nastavení si zapamatuje a promítne se do všech obrazovek v aplikaci.

ThemeContext.js

- Komponenta obsahuje logiku dvou motivů – světlého a tmavého.
- Je nastavitelná skrze obrazovky FirstSetupScreen a SettingsScreen, nastavení si zapamatuje a promítne do všech obrazovek v aplikaci.

theme.js

- Komponenta uchovává všechny barvy pro světlý a tmavý motiv, které jsou použity v logice nastavení světlého a tmavého motivu.

Další součástí aplikace, které jsou potřeba k vývoji jsou:

Assets

- Jedná se o složku, která obsahuje obrázky a další mediální soubory, které v aplikace používáme.
- V případě naší aplikace je zde vloženo logo NRZP ČR a loga partnerů NRZP ČR.

node_modules

- Další esenciální část aplikace, složka, která obsahuje balíčky a knihovny aplikace
- V našem případě není potřeba manuálně konfigurovat, aktualizuje se sama při instalování balíčků/knihoven skrz terminál.

App.json

- Obsahuje nastavení aplikace jako je jméno, ikony aplikace, je zde možné specifikovat tyto nastavení zvlášť pro každý operační systém.

Další soubory, které jsou vytvořeny samy při vytvoření jakéhokoliv nového projektu, jako je babel.config.js, package-lock.json a package.json, není potřeba v rámci projektu manuálně nastavovat.

4.3 Tvorba vlastních funkcí a základy aplikace

V této kapitole se podrobně podíváme na tvoření vlastních funkcí a nastavení souboru App.js, ve kterých definujeme funkce jako jsou například notifikace či změna motivu a textu. Tyto funkce jsou esenciální pro celou aplikaci, neboť jsou využity v obrazovkách aplikace, které pak vidí uživatel. Kapitola podrobněji vysvětluje, jak kód v React Native vypadá. Mimo jiné tyto funkce slouží i k lepší přístupnosti a přizpůsobitelnosti uživatelského prostředí. Budou zde poskytnuty a vysvětleny řádky kódu definující tyto funkce.

4.3.1 Nastavení souboru App.js

Jak již bylo zmíněno výše v kapitole struktura aplikace, tento soubor se spouští hned při spuštění aplikace uživatelem, je tedy první komponentou. Několik funkcí se odvolává na další obrazovky a soubory, které jsou vysvětleny na dalších stránkách, pokaždé je na to upozorněno.

Jako první byla nastavena „DrawerScreens“, která využije navigační knihovnu React Navigation k vytvoření postranního navigačního menu (drawer menu) a zajišťuje tedy možnost přecházet mezi různými obrazovkami.

```
function DrawerScreens() {
  return (
    <Drawer.Navigator
      initialRouteName="Welcome"
      drawerContent={props => <CustomDrawerContent {...props} />}
      screenOptions={{ header: props => <CustomHeader {...props} /> }}
    >
        <Drawer.Screen name="Welcome" component={WelcomeScreen} options={{
headerShown: false, drawerItemStyle: { height: 0 } }} />
      <Drawer.Screen name="Info" component={InfoScreen} />
      <Drawer.Screen name="Home" component={HomeScreen} />
      <Drawer.Screen name="News" component={NewsScreen} />
      <Drawer.Screen name="Settings" component={SettingsScreen} />
      <Drawer.Screen name="About" component={AboutScreen} />
    </Drawer.Navigator>
```

```
  );
}
```

`<Drawer.Navigator>` nám tedy umožňuje přechod mezi obrazovkami pomocí menu, initialRouteName nám jako první zavolá obrazovku WelcomeScreen, drawerContent volá funkci CustomDrawerContent, která je definována v souboru CustomDrawerContent.js, ta se stará o funkčnost a vzhled již zmíněného menu.

ScreenOptions zase specifikuje a volá hlavičku z CustomHeader, která je definována v rámci App.js. Následně `<Drawer.Screen>` definuje jednotlivé obrazovky. U komponentu WelcomeScreen bylo také definováno, aby se nezobrazovala hlavička a uživatel tak prošel vždy prvotním nastavením.

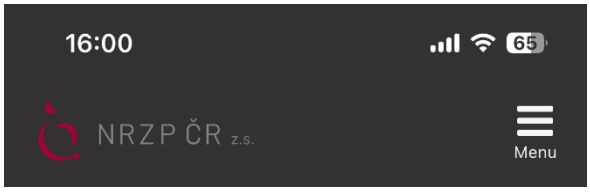
Vlastní hlavička celé aplikace byla definována v rámci App.js tímto kódem:

```
const CustomHeader = ({ navigation }) => {
  const { textSize } = useTextSize();
  const { theme } = useContext(ThemeContext);
  let activeColors = colors[theme];

  const dynamicStyles = StyleSheet.create({
    headers: {
      backgroundColor: activeColors.primaryContrast,
    },
    menuIcon: {
      fontSize: textSize + 10,
      color: activeColors.primaryT,
    },
    menuText: {
      color: activeColors.primaryT,
      fontSize: textSize - 5,
    },
    menuButton: {
      alignItems: 'center',
    },
  });

  return (
    <SafeAreaView style={dynamicStyles.headers}>
        <StatusBar barStyle={theme === 'dark' ? 'light-content' : 'dark-
content'} />
```

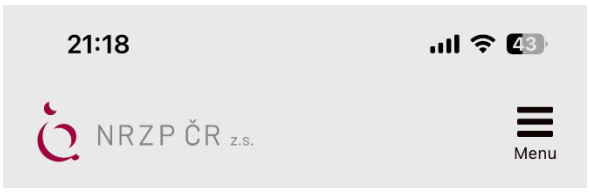
```
      <View style={styles.headerContainer}>
        <Image source={require('./assets/nrzp.png')} style={styles.logo} />
             <TouchableOpacity  onPress={()  =>  navigation.toggleDrawer()}
style={dynamicStyles.menuButton}>
          <FontAwesome name="bars" style={dynamicStyles.menuIcon} />
          <Text style={dynamicStyles.menuText}>Menu</Text>
        </TouchableOpacity>
      </View>
    </SafeAreaView>
  );
};

const styles = StyleSheet.create({
  headerContainer: {
    flexDirection: 'row',
    justifyContent: 'space-between',
    alignItems: 'center',
    paddingHorizontal: 20,
    height: 75,
  },
  logo: {
    width: 150,
    height: 50,
  },
});
```

CustomHeader byl nejdříve deklarován a iniciován. Použití useTextSize je vysvětleno v kapitole Nastavení funkce velikosti textu a použití `useContext(ThemeContext)` a `colors[theme]` je vysvětleno v kapitole Nastavení funkce světlého a tmavého motivu. DynamicStyles slouží k nastavení stylu hlavičky, bylo zde využito dvou motivů a různé velikosti textu.

Následně se volá `<SafeAreaView>`, který zajišťuje, aby se hlavička správně zobrazovala na koncovém zařízení a nekolidovala s ostatní objekty systému. StatusBar mění styl stavové lišty (hodiny, ikona wi-fi aj.) podle nastaveného motivu. Dále už se pomocí `<View>` zobrazí nejdříve logo a ikonu menu, které nám otevře a zavře postranní menu po kliknutí. Byla zde použita ikona z knihovny FontAwesome.
Výsledná hlavička v obou barevných motivech:



Obr. 4: Hlavička tmavého motivu (vlastní zdroj)



Obr. 5: Hlavička světlého motivu (vlastní zdroj)

Následující kód vrací obsah, který zajišťuje, že všechny obrazovky správně zobrazí navigaci, motiv a velikost textu a zajišťuje, jaká obrazovka se zobrazí při prvním spuštění.

```
  return (
    <ThemeProvider>
      <TextSizeProvider>
        <NavigationContainer>
       {isFirstLaunch ? <MainStack initialRouteName="Welcome" /> : <MainStack
initialRouteName="Home" />}
        </NavigationContainer>
      </TextSizeProvider>
    </ThemeProvider>
  );
```

Nedílnou součástí každého souboru v RN aplikaci jsou importy knihoven, komponent a import samotných obrazovek. Bez toho není možné používat navigaci, menu, obrázky, text, hooky aj. V souboru App.js to vypadá takto:

```
import React, { useEffect, useState, useContext } from 'react';
import { StatusBar, SafeAreaView, View, Image, TouchableOpacity, StyleSheet,
Text } from 'react-native';
import { NavigationContainer } from '@react-navigation/native';
import { createDrawerNavigator } from '@react-navigation/drawer';
import { createStackNavigator } from '@react-navigation/stack'; // Import
createStackNavigator
import { FontAwesome } from '@expo/vector-icons';
import WebViewScreen from './WebViewScreen'; // Ensure WebViewScreen is
imported
import { TextSizeProvider, useTextSize } from './TextSizeContext';
import CustomDrawerContent from './CustomDrawerContent';
import { ThemeProvider, ThemeContext } from './Contexts/ThemeContext';
import { colors } from './config/theme';
import * as Notifications from 'expo-notifications';
import axios from 'axios';
import AsyncStorage from '@react-native-async-storage/async-storage';

// Screen imports
```

```
import WelcomeScreen from './WelcomeScreen';
import NewsScreen from './NewsScreen';
import HomeScreen from './HomeScreen';
import SettingsScreen from './SettingsScreen';
import AboutScreen from './AboutScreen';
import FullArticleScreen from './FullArticleScreen';
import InfoScreen from './InfoScreen';
```

4.3.2 Nastavení funkce světlého a tmavého motivu

Nyní je představen kód, který spravuje světlý a tmavý motiv.

```
import React, { createContext, useState, useEffect } from 'react';
import AsyncStorage from '@react-native-async-storage/async-storage';

export const ThemeContext = createContext();

export const ThemeProvider = ({ children }) => {
  const [theme, setTheme] = useState('light');

  useEffect(() => {
    (async () => {
      const storedTheme = await AsyncStorage.getItem('theme');
      if (storedTheme) {
        setTheme(storedTheme);
      }
    })();
  }, []);

  const toggleTheme = async () => {
    const newTheme = theme === 'light' ? 'dark' : 'light';
    setTheme(newTheme);
    await AsyncStorage.setItem('theme', newTheme);
  };

  return (
    <ThemeContext.Provider value={{ theme, toggleTheme }}>
      {children}
    </ThemeContext.Provider>
  );
};
```

Jedná se o kód souboru ThemeContext.js, tento kód začíná importováním hooků a AsyncStorage k uložení dat, které byly použiti k funkci správy motivů.

Byl vytvořen který slouží pro sdělení informací o aktuálně zvoleném tématu. Byla definována komponenta ThemeProvider, u které je nativně nastavený motiv „light“, tedy světlý. UseState slouží k uchování motivu. UseEffect slouží k načtení operace uloženého tématu z AsyncStorage. AsyncStorage uchovává informace o tom, který motiv je zrovna nastaven. Díky tomu zůstává motiv po uložení na všech stránkách i po druhém spuštění stejný. Funkce toggleTheme zajišťuje přepínání mezi motivy a následně uloží do AsyncStorage. ThemeContext.Provider pak umožňuje všem vnořeným komponentám přístup k datům aktuálního motivu a funkce ‘toggleTheme‘.
Dalším souborem zajišťující správnou funkcionalitu je theme.js:

```
export const colors = {

    light: {
        primary: '#F6F6F6',
// další barvy
    },
    dark: {
        primary: '#000005',
// další barvy
}
```

V tomto souboru byly uloženy všechny barvy a byly přiřazeny hodnoty. Tyto barvy je možné použít napříč celou aplikací. Komponentě je přiřazena barva, například activeColors.primary, a při světlém tématu se zobrazí bílá barva, u tmavého černá.

Také je potřeba, jak již bylo zmíněno v kapitole zabývající se App.js obalit MainStack komponentou <ThemeProvider>, aby tato funkce fungovala napříč všemi soubory a obrazy. Následně stačí v obrazech, kde si přejeme, aby uživatel mohl nastavení motivu změnit, přidat funkci pro změnu motivu „toggleTheme“ a přizpůsobení barev tématu `let active Colors = colors[theme];` které na základě tématu vyberou barvu. Následně byly také přidány hooky:

```
  const { theme, toggleTheme } = useContext(ThemeContext);
```

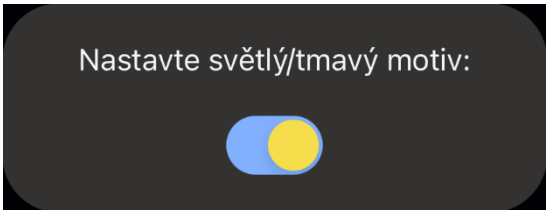
nakonec stačilo přidat následující přepínač:

```
              <Switch
```

```
                trackColor={{ false: '#767577', true: '#81b0ff' }}
                thumbColor={theme === 'dark' ? '#f5dd4b' : '#f4f3f4'}
                onValueChange={handleThemeToggle}
                value={theme === 'dark'}
            />
```

Při stisknutí na přepínač se tedy změní aktuální motiv. Styl přepínače je nastaven tak, aby se měnil podle daného tématu a poskytoval tím zpětnou vazbu.

Obr. 7: Přepínač ve světlém motivu (vlastní zdroj)



Obr. 6: Přepínač v tmavém motivu (vlastní zdroj)

Když je toto nastaveno, stačí u všech obrazovek, kde je žádoucí, aby se motiv mohl měnil, naimportovat:

```
import { colors } from './config/theme';
import { ThemeContext } from "./Contexts/ThemeContext";
```

A definovat:

```
  const { theme } = useContext(ThemeContext);
  let activeColors = colors[theme];
```

barvu jednotlivých komponent je možno měnit pomocí stylů, např.:

```
      backgroundColor: activeColors.primary,
```

4.3.3 Nastavení funkce velikosti textu

Další funkcí, která je potřeba implementovat na všechny obrazovky, je různá velikost textu podle nastavení uživatele. O to se postará kód v souboru TextSizeContext.js:

```
import React, { createContext, useContext, useState, useEffect } from 'react';
import AsyncStorage from '@react-native-async-storage/async-storage';

const TextSizeContext = createContext();

export const useTextSize = () => useContext(TextSizeContext);

export const TextSizeProvider = ({ children }) => {
  const [textSize, setTextSize] = useState(18);
```

```
  useEffect(() => {
    const loadTextSize = async () => {
      const savedTextSize = await AsyncStorage.getItem('textSize');
      if (savedTextSize !== null) {
        setTextSize(Number(savedTextSize));
      }
    };

    loadTextSize();
  }, []);

  useEffect(() => {
    AsyncStorage.setItem('textSize', textSize.toString());
  }, [textSize]);

    const    increaseTextSize    =    ()    =>    setTextSize(currentSize    =>
Math.min(currentSize + 2, 30));
    const    decreaseTextSize    =    ()    =>    setTextSize(currentSize    =>
Math.max(currentSize - 2, 10));

  return (
        <TextSizeContext.Provider    value={{    textSize,    increaseTextSize,
decreaseTextSize }}>
      {children}
    </TextSizeContext.Provider>
  );
};
```

Na začátek bylo znovu potřeba naimportovat potřebné hooky (useState, useEffect aj.) a znovu AsyncStorage pro ukládání dat na zařízení.

Dále byl vytvořen TextSizeContext, který umožní komponentám přístup k datům. Byl definován vlastní hook useTextSize, který zajistí, že kdykoliv ho použijeme v komponentě, umožní jí číst a měnit velikost textu.
Kontext TextSizeProvider pak uchovává velikost textu pomocí useState, automaticky je nastavena na 18. UseEffect načítá uloženou velikost textu z úložiště a při nalezení hodnoty aktualizuje velikost komponenty. Druhý useEffect poté vloží novou hodnotu do AsyncStorage, díky tomu si při dalším spuštění aplikace pamatuje zvolenou velikost.

Funkce increaseTextSize a decreaseTextSize pak slouží ke změně velikosti textu, zároveň je nastaveno minimum a maximum. TextSizeContext.Provider obaluje vnořené komponenty a poskytuje přístup k datům k definovaným funkcím.

Stejně jako v případě motivu bylo třeba obalit celý MainStack a NavigationContainer v App.js pomocí TextSizeProvider aby nám kód fungoval napříč celou aplikací.

Dále bylo potřeba přidat funkce, aby si uživatel mohl nastavit na své obrazovce velikost textu. Do obrazů WelcomeScreen a SettingsScreen byly přidány následující řádky kódu:

```
import React, { useState, useContext } from 'react';
 const [textSize, setTextSize] = useState(18);
```

Tímto kódem byly definovány textSize a setTextSize v rámci tohoto obrazu.

```
    const increaseTextSize = () => setTextSize(currentSize => currentSize <
30 ? currentSize + 2 : 30);
    const decreaseTextSize = () => setTextSize(currentSize => currentSize >
10 ? currentSize - 2 : 10);
```

Definice increaseTextSize a decreaseTextSize aby bylo možno v rámci tohoto obrazu měnit velikost.

```
<View style={dynamicStyles.buttonContainer}>
 <TouchableOpacity style={dynamicStyles.button1} onPress={increaseTextSize}>
    <Text style={dynamicStyles.buttonText}>+ Větší</Text>
 </TouchableOpacity>
 <TouchableOpacity style={dynamicStyles.button1} onPress={decreaseTextSize}>
     <Text style={dynamicStyles.buttonText}>- Menší</Text>
 </TouchableOpacity>
</View>
```

Tento kód nám vytvoří tlačítka, která pomocí definovaných funkcí zvětší nebo zmenší velikost textu. Definovanému stylu v rámci obrazovky byla nastavena hodnota:

```
  const styles = StyleSheet.create({
        buttonText: {
            fontSize: textSize,
        },
    });
```

Velikost textu v tlačítku se tedy bude měnit dynamicky podle toho, jak veliký text uživatel nakliká. Tuto hodnotu bylo potřeba přiřadit každé UI komponentě pro dynamickou změnu velikosti.

V dalších souborech/obrazech bylo potřeba naimportovat useTextSize, definovat textSize a přiřadit hodnotu textSize dané komponentě.

```
import { useTextSize } from './TextSizeContext';
// další kód
  const { textSize } = useTextSize();
  const styles = StyleSheet.create({
    date: {
      fontSize: textSize,
    },
    });
```

4.3.4 **Nastavení vlastního scrollování**

Dalším problémem, který bylo potřeba vyřešit, je možnost scrollování (rolování) u obrazů tak, aby se obrazy i rolovací lišta správně zobrazily na všech platformách.

Platforma React Native umožňuje naimportovat nativní komponentu ScrollView, sama o sobě je však nespolehlivá a při testování se na různých obrazovkách nezobrazovala správně. Problém byl vyřešen rozšířením standartní funkcionality ScrollView:

```
const CustomScrollView = ({ children }) => {
        const   [scrollIndicatorPosition,   setScrollIndicatorPosition]   =
useState(0);
    const [scrollIndicatorHeight, setScrollIndicatorHeight] = useState(0);
```

Tímto kódem byla zajištěna pozici lišty, inicializovaná hodnota je 0.

```
    const handleScroll = (event) => {
      const contentHeight = event.nativeEvent.contentSize.height;
      const scrollViewHeight = event.nativeEvent.layoutMeasurement.height;
      const yOffset = event.nativeEvent.contentOffset.y;
         const  indicatorHeight  =  (scrollViewHeight  /  contentHeight)  *
scrollViewHeight;
      const position = (yOffset / contentHeight) * scrollViewHeight;

      setScrollIndicatorHeight(indicatorHeight);
      setScrollIndicatorPosition(position);
```

Pomocí funkce handleScroll je nastavena a počítána pozice a výšky lišty na základě toho, kde se uživatel nachází, a jak velká je daná obrazovka. Poslední dva řádky uchovávají pozici a výšku lišty.

```
const styles = StyleSheet.create({
    container: {
        flex: 1,
        paddingHorizontal: 15,
        paddingVertical: 10,
        position: 'relative',
        backgroundColor: activeColors.primary,
    },
    customScrollIndicator: {
        position: 'absolute',
        right: 2,
        width: 4,
        backgroundColor: activeColors.primaryT,
        borderRadius: 2,
    },
});
```

Pomocí stylů byly prováděny poslední úpravy toho, jak chceme, aby se nám zobrazila lišta na obrazovkách. Mimo jiné byl implementován activeColors, aby se barva lišty přizpůsobila právě zvolenému motivu.

```
return (
    <View style={styles.container}>
        <ScrollView
            onScroll={handleScroll}
            scrollEventThrottle={16}
            showsVerticalScrollIndicator={false}
            contentContainerStyle={{ paddingRight: 20 }}
        >
            {children}
        </ScrollView>
        <View
            style={[
                styles.customScrollIndicator,
                {
                    height: scrollIndicatorHeight,
                    top: scrollIndicatorPosition,
```

```
                    },
                ]}
            />
        </View>
    );
```

Komponenta `<View>` následně skrývá základní lištu a pomocí onScroll dynamicky aktualizuje pozici a výšku lišty, `{children}` nám slouží k opětovnému použití lišty pro obsah na jiných obrazovkách.

Na zvolené obrazovce stačí obalit kód tagem `<CustomScrollView>`, následně ho naimportovat: `import CustomScrollView from './CustomScrollView';` a vlastní lišta je zobrazena.

4.3.5 Nastavení menu aplikace

Pro definování obrazovek a stylu vlastního menu byl vytvořen soubor CustomDrawerContent.js, který obsahuje informace o položkách, které se zobrazí v menu:

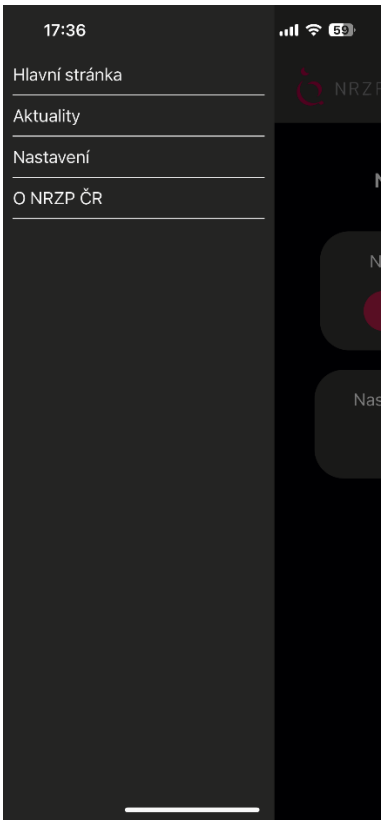
```
<DrawerContentScrollView {...props} style={{ backgroundColor: activeColors.menu }}>
      <DrawerItem routeName="Home" label="Hlavní stránka" />
      <DrawerItem routeName="News" label="Aktuality" />
      <DrawerItem routeName="Settings" label="Nastavení" />
      <DrawerItem routeName="About" label="O NRZP ČR" />
    </DrawerContentScrollView>
```

Bylo také potřeba znovu nastavit styly tak, aby se měnili podle motivu a velikosti textu:

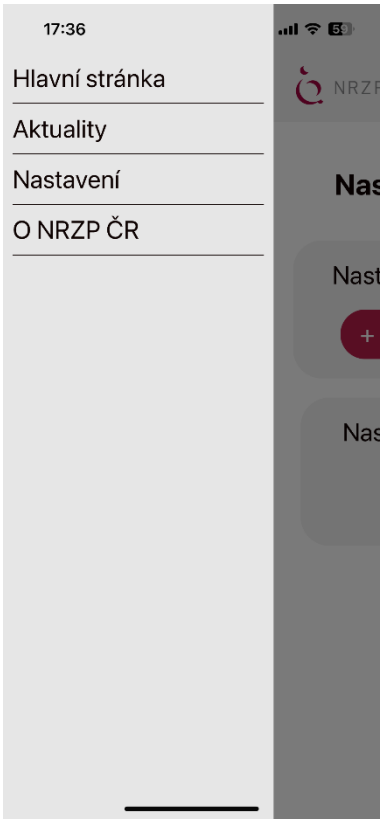
```
  const styles = StyleSheet.create({
    text: {
      fontSize: textSize,
      padding: 10,
      color: activeColors.primaryT,
    },
    separator: {
      height: 1,
      backgroundColor: activeColors.primaryT,
      marginHorizontal: 10,
    },
  });
```

O navigaci se stará funkce DrawerItem. Po kliknutí na položku menu naviguje podle routeName.

```
const DrawerItem = ({ routeName, label }) => (
  <>
    <TouchableOpacity
      onPress={() => navigation.navigate(routeName)}
    >
      <Text style={styles.text}>{label}</Text>
    </TouchableOpacity>
    <View style={styles.separator} />
  </>
```



Obr. 9: Menu v tmavém motivu, základní velikost textu (vlastní zdroj)



Obr. 8: Menu ve světlém motivu, maximální velikost textu (vlastní zdroj)

4.3.6 Nastavení článků pomocí Axios, API a notifikace

Jelikož je potřeba zobrazit články NRZP ČR z webové stránky, je využit klient node.js Axios pro demonstraci získávání článku z API endpointu 'https://newsapi.org/v2/top. Ze stránky newsapi.org jsme získali unikátní API klíč.

```
useEffect(() => {
```

```
    const fetchNews = async () => {
      try {
           const  response  =  await  axios.get('https://newsapi.org/v2/top-
headlines', {
          params: {
            country: 'us',
            apiKey: '8bc7295afdff4b02a6f292a006ab1995',
          }
        });
        if (response.data.articles) {
          setArticles(response.data.articles);

                              const     notifiedTitlesString     =     await
AsyncStorage.getItem('notifiedTitles');
                     const    notifiedTitles    =    notifiedTitlesString    ?
JSON.parse(notifiedTitlesString) : [];

             const  newArticles  =  response.data.articles.filter(article  =>
!notifiedTitles.includes(article.title));

          newArticles.forEach(async (article, index) => {
            await Notifications.scheduleNotificationAsync({
              content: {
                title: "Nový článek!",
                body: article.title,
                data: { url: article.url },
              },
              trigger: { seconds: (index + 1) * 2 },
            });

            notifiedTitles.push(article.title);
          });

                             await     AsyncStorage.setItem('notifiedTitles',
JSON.stringify(notifiedTitles));
        }
      } catch (error) {
        console.error('Error fetching news:', error);
      }
    };

    fetchNews();
    const intervalId = setInterval(fetchNews, 60000 * 60);

    return () => clearInterval(intervalId);
  }, []);
```

Znovu je použita AsyncStorage pro uložení již zveřejněných, v tomto případě oznámených, článků. Zajišťuje to, že nedostaneme notifikaci o novém článku dvakrát. Zprávy se načtou vždy při načtení komponenty a také se automaticky aktualizují každou hodinu. Tyto články se pak zobrazí v obrazu NewsScreen.js, který se stará o jejich otevírání v novém obrazu a design.

Aby zařízení umožňovalo posílat notifikace, je potřeba získat povolení zařízení. Je tedy esenciální přidat tuto část kódu do App.js:

```
async function registerForPushNotificationsAsync() {
      const      {      status:      existingStatus      }      =      await
Notifications.getPermissionsAsync();
  let finalStatus = existingStatus;
  if (existingStatus !== 'granted') {
    const { status } = await Notifications.requestPermissionsAsync();
    finalStatus = status;
  }
  if (finalStatus !== 'granted') {
    alert('Failed to get push token for push notification!');
    return;
  }
   const token = (await Notifications.getExpoPushTokenAsync({ experienceId:
'@Projekt/Projekt' })).data;
  console.log(token);
}
```

4.4 Ukázky jednotlivých obrazů aplikace

V této kapitole jsou zobrazeny všechny obrazy aplikace a podoba uživatelského rozhraní. Každý obraz je znázorněn snímky obrazovky ze zařízení platformy iOS, na kterém byl proveden vývoj i testování.

4.4.1 Uvítací obrazovka při prvním spuštění

Obrazovka, kterou aplikace zobrazí při prvním spuštění má v projektu název WelcomeScreen.

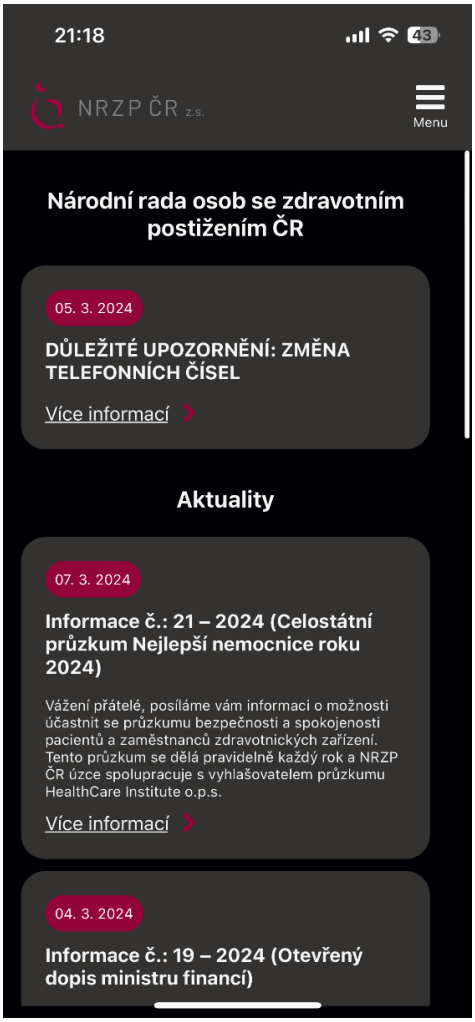
Obr. 10: Uvítací obrazovka ve světlém motivu, větší velikost textu (vlastní zdroj)

Obr. 11: Uvítací obrazovka v tmavém motivu, standartní velikost textu (vlastní zdroj)

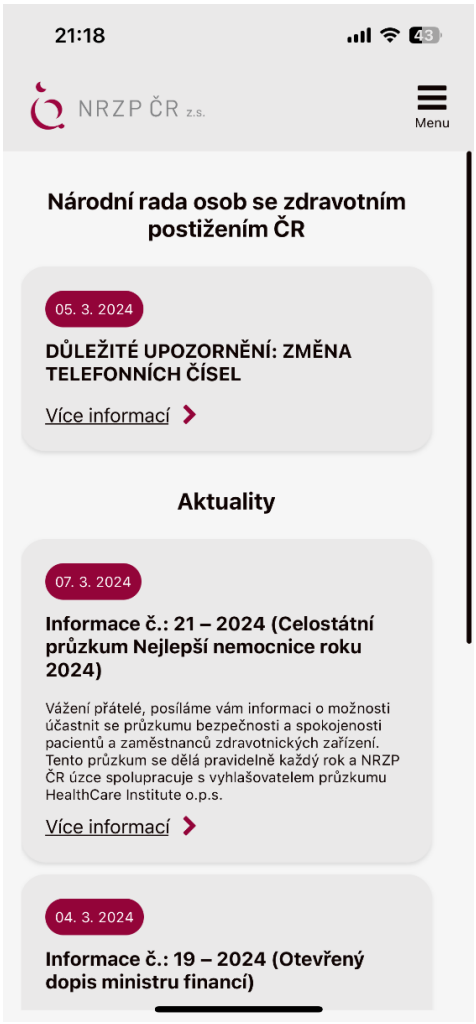
Na obrazcích výše je vidět výstup, který vidí uživatel. Tato obrazovka slouží k přivítání nového uživatele. Jak již zbylo zmíněno v kapitole o nastavení souboru App.js, je zobrazena pouze při prvním spuštění. Její primární funkcí je nabídnout uživateli hned při spuštění možnost přizpůsobení aplikace pro jeho potřeby, aby se mu co nejlépe ovládala. Přináší tedy možnost změnit velikost textu a změnit motiv. Po následném uložení se uživatel přemístí na domovskou stránku aplikace.

4.4.2 **Domovská stránka**

Domovská obrazovka aplikace je první, co uživatel uvidí, když už jednou projde prvotním nastavením. Soubor nese název HomeScreen.



Obr. 13:Domovská obrazovka ve tmavém motivu (vlastní zdroj)

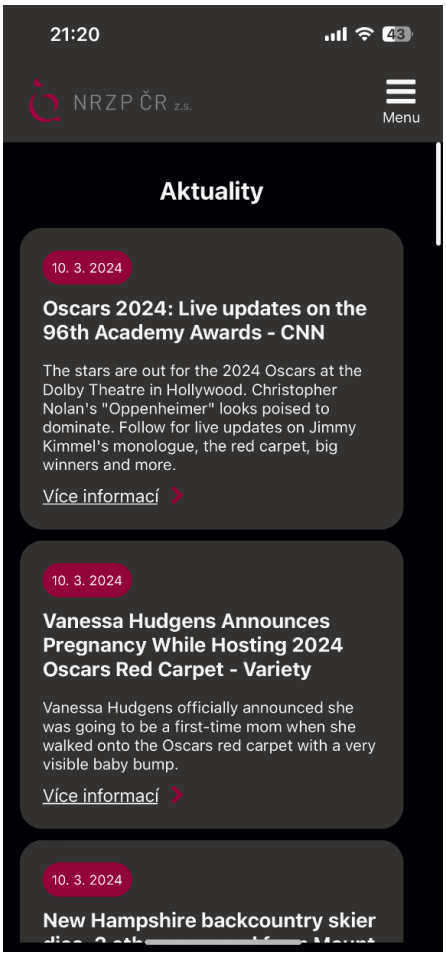


Obr. 12: Domovská obrazovka ve světlém motivu (vlastní zdroj)

Tato stránka slouží k zobrazení nejdůležitější aktuality/oznámení, ta se zobrazí hned jako první, dále jsou zobrazeny tři poslední aktuality. Všechny aktuality lze otevřít a dole je možnost přejít na obrazovku se všemi aktualitami. Je zde také už vidět hlavička celé aplikace, která obsahuje logo NRZP ČR a postranní menu k navigaci.

4.4.3 **Stránka s aktualitami**

Na této stránce – NewsScreen – jsou automaticky volány nejnovější aktuality pomocí API klíče, jelikož zatím nebylo možné v rámci práce napojit API na stránky NRZP ČR, je pro demonstraci napojeno na newssapi.org, které zdarma poskytují tuto službu.



Obr. 14: Prototyp automatických aktualit – zprávy z 10.3.2024, tmavý motiv aplikace (vlastní zdroj)

Uživatelské rozhraní této stránky je přizpůsobeno stylu celé aplikace. Bere si data z webové stránky pomocí API klíče, které následně pomocí stylů změníme tak, aby zapadaly do celkového designu aplikace. Momentálně můžeme vidět články z amerických novin. Funkcionalita a front-end zobrazení už jsou nastaveny.

4.4.4 Stránka s nastavením aplikace

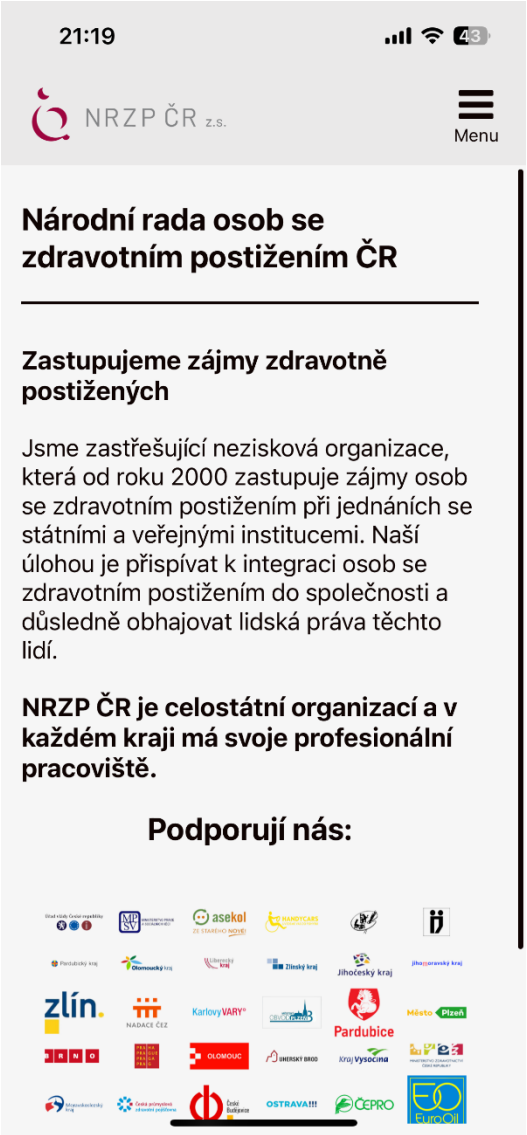
Tato stránka je velmi podobná první uvítací obrazovce. Zobrazuje stejné nastavení motivu a textu. Slouží tedy ke změně prvotního nastavení uživatele, které je jednoduše dostupné z menu. V aplikaci se jedná o soubor SettingsScreen.

světlý motiv a standartní text (vlastní zdroj)

Obr. 16: Obrazovka nastavení aplikace, tmavý motiv a téměř maximální text (vlastní zdroj)

4.4.5 Stránka s informacemi o NRZP ČR

V naši aplikaci má název AboutScreen – zobrazuje základní informace o NRZP ČR a loga všech partnerů.



Obr. 18: Obrazovka o NRZP ČR, světlý motiv (vlastní zdroj)

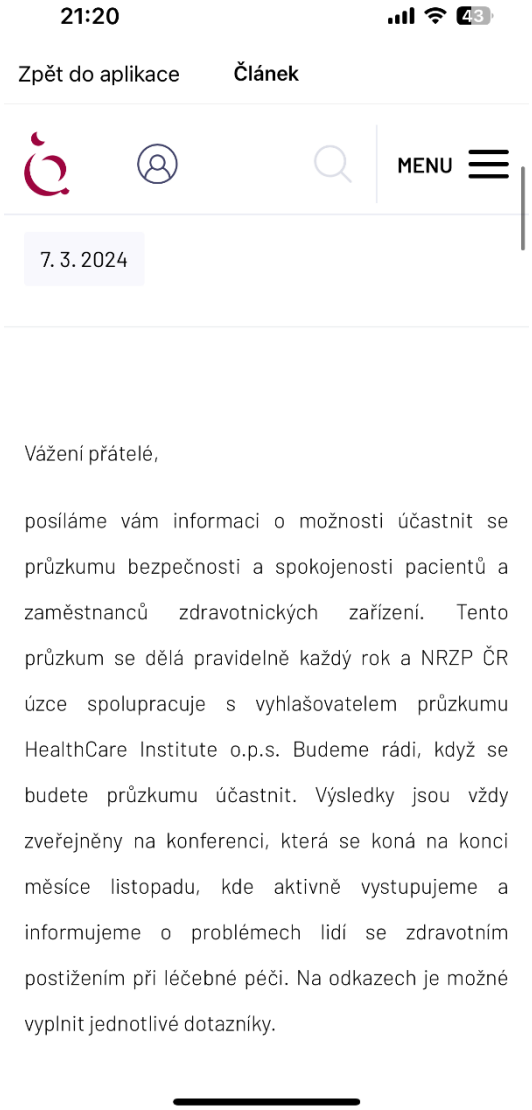
Obr. 17: Obrazovka o NRZP ČR, tmavý motiv (vlastní zdroj)

4.4.6 Stránky zobrazující celý otevřený článek

Tato stránka je přístupná, když uživatel otevře článek. Využívá technologie WebView, která pomocí URL webu zobrazí v aplikaci webovou stránku. Má nastavenou vlastní hlavičku, ze

které lze z funkce WebView odejít zpět. Jedná se o jednoduché řešení, jak zobrazit článek. Dále je vyobrazena ukázka stránky – akutality, která nevyužívá technologie WebView ale byla plnohodnotně převedena do aplikačního rozhraní.

Obr. 20: Aktualita/Upozornění plnohodnotně zobrazena v aplikaci (vlastní zdroj)



Obr. 19: Aktualita zobrazena skrz technologii WebView (vlastní zdroj)

4.5 Testování aplikace na platformě Android

Na platformě iOS byla aplikace testována a používána po celou dobu vývoje. V této kapitole byl proveden průchod aplikací a případné řešení problémů na platformě Android. Pro kontrast s novým zařízením iOS bylo použito levnější a starší zařízení Realme 7.

4.5.1 Spuštění aplikace

Hned při naskenování QR kódu skrze aplikaci Expo Go, byl po načtení zobrazen error: `ERROR TypeError:_RNGestureHandlerModule.default.flushOperations is not a function (it is undefined), js engine: hermes`
Jednoduchým nainstalováním knihovny pomocí příkazu `npm install react-native-gesture-handler` v terminálu byl error vyřešen.

4.5.2 Průchod aplikací a řešení problémů

Na první pohled byly všechny komponenty větší než na iOS a při zvětšení textu pár komponent mizí z obrazovky. Dalším problémem byla neviditelná statusová lišta (ikona Wi-Fi, baterie ad.) při světlém motivu aplikace. To je způsobeno tím, že na iOS se pozadí lištu přizpůsobovalo barvě aplikace, na zařízení s Androidem zůstalo vždy černé. Žádný další problém ale při důkladném průchodu aplikací nenastal.

Problém se komponentou StatusBar byl vyřešen přidáním kódu, který mění pozadí lišty i na zařízení, které to nedělá automaticky. Přidali jsem tedy tuto část kódu uvedenou níže a vše fungovalo správně.

```
  <StatusBar
        backgroundColor={theme   ===   'dark'   ?   activeColors.primary   :
activeColors.primaryContrast}
    barStyle={theme === 'dark' ? 'light-content' : 'dark-content'}
  />
```

Problém s příliš velkým textem byl vyřešen tak, že v rámci souboru TextSizeContext byla snížena maximální velikost textu na platformě Android na 24. Při porovnání na různých

zařízeních, je tato hodnota skoro shodná (lehce větší) než hodnota 30 na iOS. Do souboru TextSizeContext.js byly přidány následující řádky:

```
import { Platform } from 'react-native';
const maxTextSize = Platform.OS === 'android' ? 24 : 30;
  const increaseTextSize = () => {
    const newSize = Math.min(textSize + 2, maxTextSize);
    setTextSize(newSize);
```

Na platformě iOS byla zachována maximální velikost 30 a na Android byla snížena na 24. Při následném průchodu aplikací byly znovu vyzkoušeny všechny další funkce včetně notifikací a nebyly nalezeny žádné další problémy.

Obr. 21: Porovnání – maximální velikost textu na zařízení se systémem Android (vlastní zdroj)

Obr. 22: Porovnání – maximální velikost textu na zařízení se systímem iOS (vlastní zdroj)

4.6 Vyhodnocení a diskuse

V současném stavu se podařilo vytvořit funkční front-end (uživatelskou část) aplikace. Aplikace funguje v rámci Expo Go na obou plánovaných platformách bez problémů. Stejně tak všechny obrazy a funkce fungují bez potíží. Podařilo se implementovat několik funkcí, které činí aplikaci dobře přístupnou pro lidi s různými vadami očí. Zároveň byly při vývoji zachovány všechny praktiky pro tvorbu přístupné aplikace.

Nyní je potřeba společně s NRZP ČR a vývojáři webových stránek NRZP ČR nastavit automatické načítání aktualit skrze napojení aplikace na webové stránky. Aplikace je ale na to z uživatelské části velmi dobře připravena. Po dokončení a konfiguraci všech potřebných prací na serverové části aplikace (aktuality a notifikace) bude aplikace připravena na finální testování, převedení do nového vývojářského prostředí a následné nasazení na koncová zařízení.

5 Závěr

V průběhu této práce bylo dosaženo několika klíčových milníků. V teoretické části práce se nám podařilo charakterizovat činnost NRZP ČR a tím získat přehled, jak a proč danou aplikaci vyvíjet. Bylo provedeno bádání v oblasti tvorby přístupných mobilních aplikací převážně s ohledem na zraková postižení. Na základě získaných vědomostí a informací ohledně dnešního přístupu k tvorbě mobilních aplikací a jejich technologií, byl vytvořen základ pro vývoj mobilní aplikace pro platformy iOS a Android.

Vymezením struktury aplikace a charakterizování funkcí byl vytvořen podklad pro následný vývoj. Podařilo se nám implementovat všechny požadované funkce zajišťující přístupnost a fungování aplikace, jako je možnost změny motivu a velikost textu a funkční uživatelské rozhraní. Tyto funkce přispívají k lepší ovladatelnosti aplikace pro osoby s různými zrakovými postiženími. Stejně tak jsme zachovali všechny praktiky zjištěné v teoretické části, jako jsou například popisky či zvýraznění tlačítek.

Funkce aktualit NRZP ČR spolu s notifikacemi byla nastavena pouze demonstračně. Nebylo možné se při tvorbě práce dostat k požadovaným datům, které by tyto funkce nastavovaly s obsahem webu NRZP ČR.

Podařilo se vytvořit funkční a přístupné uživatelské prostředí, ale vzhledem k chybějícímu napojení na web NRZP ČR aplikace zatím nemůže být implementována, nýbrž funguje jako demonstrační prototyp.

6 Zdroje

[1] N. ČR, „NRZP,“ 2024. [Online]. Available: https://nrzp.cz/. [Přístup získán 15 Leden 2024].

[2] N. ČR, „O NRZP,“ 2024. [Online]. Available: https://nrzp.cz/o-nrzp/. [Přístup získán 16 Leden 2024].

[3] N. ČR, „Inkluzivní začleňování osob se zdravotním postižením a speciálně vzdělávacími potřebami,“ 2024. [Online]. Available: https://nrzp.cz/projekty/inkluzivni-zaclenovani-osob-se-zdravotnim-postizenim-a-specialne-vzdelavacimi-potrebami/. [Přístup získán 17 Leden 2024].

[4] N. ČR, „Archiv aktualit,“ 2024. [Online]. [Přístup získán 18 Leden 2024].

[5] N. ČR, „Informace č.: 106 – 2023 (Hlasování Poslanecké sněmovny o zařazení bodu o příspěvku na péči),“ 2023. [Online]. Available: https://nrzp.cz/2023/11/30/informace-c-106-2023-hlasovani-poslanecke-snemovny-o-zarazeni-bodu-o-prispevku-na-peci/. [Přístup získán 18 Leden 2024].

[6] M. Mgr. Sabrina Plisková, „TISKOVÁ ZPRÁVA – Žádáme přístupný železniční most na Výtoni,“ 13 Prosinec 2023. [Online]. Available: https://nrzp.cz/2023/12/13/tiskova-zprava-zadame-pristupny-zeleznicni-most-na-vytoni/. [Přístup získán 18 Leden 2024].

[7] M. V. Krása, „Informace č.: 97 – 2023 (Jak s placením elektrické energie v novém roce),“ 13 Listopad 2023. [Online]. Available: https://nrzp.cz/2023/11/13/informace-c-97-2023-jak-s-placenim-elektricke-energie-v-novem-roce/. [Přístup získán 2 Únor 2024].

[8] M. V. Krása, „Informace č.: 74 – 2023 (Velká novela zákona o sociálních službách),“ 7 Září 2023. [Online]. Available: https://nrzp.cz/2023/09/07/informace-c-74-2023-velka-novela-zakona-o-socialnich-sluzbach/. [Přístup získán 2 Únor 2024].

[9] V. ČR, „Vládní výbor pro osoby se zdravotním postižením,“ 2024. [Online]. Available: https://vlada.gov.cz/cz/ppov/vvzpo/prehled-nejdulezitejsich-mezinarodnich-instituci-a-organizaci-monitorujicich-situaci-zdravotne-postizenych-osob-v-jednotlivych-zemich-323/. [Přístup získán Leden 20 2024].

[10] K. ú. J. kraje, „OBČANÉ SE ZDRAVOTNÍM POSTIŽENÍM,“ 2024. [Online]. Available: https://www.kraj-jihocesky.cz/cs/obcan-urady/obcane-se-zdravotnim-postizenim. [Přístup získán 2 Únor 2024].

[11] M. z. Č. republiky, „Práva osob se zdravotním postižením,“ 2024. [Online]. Available: https://www.nzip.cz/clanek/241-prava-osob-se-zdravotnim-postizenim. [Přístup získán 2 Únor 2024].

[12] R. Stevens, „Designing Apps with Colour Blind Users in Mind,“ 18 Říjen 2018. [Online]. [Přístup získán 25 Leden 2024].

[13] National Eye Institute, „At a glace: Color Blindness,“ 15 Říjen 2023. [Online]. Available: https://www.nei.nih.gov/learn-about-eye-health/eye-conditions-and-diseases/color-blindness. [Přístup získán 25 Leden 2024].

[14] R. Stevens, „Designing Apps with Colour Blind Users in Mind,“ 18 Říjen 2018. [Online]. Available: https://resource.esriuk.com/wp-content/uploads/1_1.jpg. [Přístup získán 25 Leden 2024].

[15] National Eye Institute, „At a glance: Low Vision,“ 15 Listopad 2023. [Online]. Available: https://www.nei.nih.gov/learn-about-eye-health/eye-conditions-and-diseases/low-vision. [Přístup získán 25 Leden 2024].

[16] K. Novotná, „Slepota a její příčiny,“ Olomouc, 2015.

[17] Ministerstvo zdravotnictví ČR a Ústav zdravotnických informací a statistiky ČR, „Oči,“ 2024. [Online]. Available: https://www.nzip.cz/kategorie/48-oci. [Přístup získán 25 Leden 2024].

[18] D. Gupta, „How to Design Accessibility App for Visually Impaired?,“ 22 Srpen 2022. [Online]. Available: https://appinventiv.com/blog/design-accessibility-app-for-visually-impaired/. [Přístup získán 2 Únor 2024].

[19] Scope, „How to write hyperlink text for better web accessibility,“ 2024. [Online]. Available: https://business.scope.org.uk/article/how-to-write-better-link-text-for-accessibility. [Přístup získán 26 Leden 2024].

[20] „WCAG Color Contrast Checker,“ 2024. [Online]. Available: https://accessibleweb.com/color-contrast-checker/. [Přístup získán 29 Únor 2024].

[21] M. Novotný, „Vývoj multiplatformních mobilních aplikací v React Native,“ Masarykova Univerzita, Brno, 2023.

[22] P. LePage a S. Richard, „What are Progressive Web Apps?,“ 1 Červen 2020. [Online]. Available: https://web.dev/articles/what-are-pwas. [Přístup získán 10 Únor 2024].

[23] M. Novotný, „Vývoj multiplatformních mobilních aplikací v React Native,“ Brno, 2023.

[24] „Cordova,“ [Online]. Available: https://cordova.apache.org/. [Přístup získán 20 Únor 2024].

[25] A. Manchanda, „The Ultimate Guide to Cross Platform App development Freamworks in 2024,“ 7 September 2023. [Online]. Available: https://www.netsolutions.com/insights/cross-platform-app-frameworks-in-2019/. [Přístup získán 13 Únor 2024].

[26] V. Mannotra, „How to build Cross-Platform Mobile Apps,“ 8 Srpen 2023. [Online]. Available: https://www.browserstack.com/guide/build-cross-platform-mobile-apps. [Přístup získán 12 Únor 2024].

[27] G. Peal, „React Native at Airbnb,“ Airbnb Tech Blog, 19 Červen 2018. [Online]. Available: https://medium.com/airbnb-engineering/react-native-at-airbnb-f95aa460be1c. [Přístup získán 13 Únor 2024].

[28] D. Janovský, „HTML příručka,“ [Online]. Available: https://www.jakpsatweb.cz/html/. [Přístup získán 8 Únor 2024].

[29] W3SSchools, „HTML Tutorial,“ [Online]. Available: https://www.w3schools.com/html/. [Přístup získán 8 Únor 2024].

[30] „Learn to style HTML using CSS,“ Mozilla.org, [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/CSS. [Přístup získán 12 Únor 2024].

[31] „JavaScript,“ 25 Září 2023. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript. [Přístup získán 20 Únor 2024].

[32] „HTML Javascript,“ [Online]. Available: https://www.w3schools.com/html/html_scripts.asp. [Přístup získán 20 Únor 2024].

[33] „KTERÝ JS FRAMEWORK ZVOLIT PRO VÝVOJ SINGLE-PAGE WEBOVÉ APLIKACE,“ 9 Duben 2023. [Online]. Available:

https://www.rascasone.com/cs/blog/javascript-frameworky-spa-jednostrankove-web-aplikace. [Přístup získán 20 Únor 2024].

[34] „NodeJS Intro,“ [Online]. Available: https://www.w3schools.com/nodejs/nodejs_intro.asp. [Přístup získán 20 Únor 2024].

[35] Meta, „React,“ [Online]. Available: https://react.dev/. [Přístup získán 20 Únor 2024].

[36] Meta, „Your First Component,“ [Online]. Available: https://react.dev/learn/your-first-component. [Přístup získán 18 Únor 2024].

[37] Meta, „Writing Markup with JSX,“ [Online]. Available: https://react.dev/learn/writing-markup-with-jsx. [Přístup získán 15 Únor 2024].

[38] Meta, „React Native,“ [Online]. Available: https://reactnative.dev/. [Přístup získán 15 Únor 2024].

[39] Meta, „Architecture Overview,“ [Online]. Available: https://reactnative.dev/architecture/overview. [Přístup získán 20 Únor 2024].

[40] Meta, „Fabric,“ [Online]. Available: https://reactnative.dev/architecture/fabric-renderer. [Přístup získán 20 Únor 2024].

[41] Meta, „Render, Commit, and Mount,“ [Online]. Available: https://reactnative.dev/architecture/render-pipeline. [Přístup získán 20 Únor 2024].

[42] Meta, „Cross Platform Implementation,“ [Online]. Available: https://reactnative.dev/architecture/xplat-implementation. [Přístup získán 26 Únor 2024].

[43] Meta, „View Flattening,“ [Online]. Available: https://reactnative.dev/architecture/view-flattening. [Přístup získán 20 Únor 2024].

[44] MEta, „Threading Model,“ [Online]. Available: https://reactnative.dev/architecture/threading-model. [Přístup získán 20 Únor 2024].

[45] Meta, „Core Components and Native Components,“ [Online]. Available: https://reactnative.dev/docs/intro-react-native-components. [Přístup získán 20 Únor 2024].

[46] K. Vekariya a A. Nayak, „How to Build a React Native App Using React Native Hooks?,“ 3 Leden 2024. [Online]. Available:

https://www.bacancytechnology.com/blog/react-native-hooks-to-build-app. [Přístup získán 29 Únor 2024].

[47] Amazon Web Services, Inc., „What is a RESTful API?," [Online]. Available: https://aws.amazon.com/what-is/restful-api. [Přístup získán 29 Únor 2024].

[48] „Getting Started," Axios, [Online]. Available: https://axios-http.com/docs/intro. [Přístup získán 29 Únor 2024].

[49] 650 Industries Inc., „Expo Go," Expo Docs, [Online]. Available: https://docs.expo.dev/get-started/expo-go/. [Přístup získán 29 Únor 2024].

[50] N. ČR, „Informační činnost NRZP ČR k podpoře vyrovnávání příležitostí OZP," 2024. [Online]. Available: https://nrzp.cz/projekty/novy-projekt/. [Přístup získán 17 Leden 2024].

7 Seznam obrázků, tabulek, grafů a zkratek

8 Přílohy

K práci je přiložen archiv Prototyp aplikace NRZP ČR, který obsahuje všechny zdrojové kódy a soubory programu.