Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Bakalářská práce

Návrh a implementace Progresivní webové aplikace

Ladislav Topoľský

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Ladislav Topolský

Informatika

Název práce

Návrh a implementace Progresivní webové aplikace

Název anglicky

Design and implementation of Progressive web application

Cíle práce

Cílem bakalářské práce je tvorba aplikace prostřednictvím open-source frontend knihovny Javascriptu React. Dalším cílem je implementace technologie progresivních webových aplikací vytvořením service worker, manifestu a zajištění bezpečného šifrovaného připojení formou protokolu HTTPS. Následně budou otestovány podmínky stanovené v Google PWA seznamu pomocí vývojářského nástroje Google Lighthouse.

Metodika

Teoretická část bakalářské práce bude obsahovat analýzu a rešerše odborných zdrojů z oblasti progresivních webových aplikací a open-source frontendovej knihovny Javascriptu React.

Praktická část bakalářské práce bude obsahovat návrh a implementaci postupů vývoje aplikace pro zajištění splnění požadavků ze seznamu Progresívních webových aplikací.

Doporučený rozsah práce

45

Klíčová slova

React, Progresivní webové aplikace, Service worker, Manifest, HTTPS

Doporučené zdroje informací

ALEX BANKS, EVE PORCELLO. Learning React, Modern Patterns for Developing React Apps. Sebastopol: O'Reilly Media, Inc, 2020. ISBN 978-1-492-05172-5

ANTHONY ACCOMAZZO, NATE MURRAY, ARI LERNER, CLAY ALLSOPP, DAVID GUTMAN, AND TYLER MCGINNIS. Fullstack React, The Complete Guide to ReactJS and Friends. \newline, 2020.

DEAN ALAN HUME. Progressive Web Apps. Shelter Island: Manning Publications Co, 2018. ISBN: 9781617294587

DENNIS SHEPPARD. Beginning Progressive Web App Development. Tinley Park: Apress, 2017. ISBN-13: 978-1-4842-3090-9

CHRIS LOVE. Progressive Web Application Development by Example. Birmingham: Packt Publishing, 2018. ISBN 978-1-78712-542-1.

SCOTT DOMES. Progressive Web Apps with React. Birmingham: Packt Publishing, 2017. ISBN 978-1-78829-755-4

Předběžný termín obhajoby

2021/22 LS – PEF

Vedoucí práce

RNDr. Alexander Galba

Garantující pracoviště

Katedra informačních technologií

Elektronicky schváleno dne 20. 8. 2021

doc. Ing. Jiří Vaněk, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 5. 10. 2021

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 08. 03. 2022

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Návrh a implementace Progresivní webové aplikace" 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 14.3.2022 _____

Poděkování

Rád bych touto cestou poděkoval panu RNDr. Alexanderovi Galbovi za odborné rady, kontrolu a konzultace ohledně mé bakalářské práce. Dále bych chtěl poděkovat panu Ing. Martinovi Havránkovi, Ph.D. za nápad na kategorii testováni.

Návrh a implementace Progresivní webové aplikace

Abstrakt

Bakalářská práce má za cíl, přiblížit návrh a implementaci aplikace, prostřednictvím technologie Progresivních webových aplikací. Aplikace je napsaná pomocí knihovny React, která je open-source JavaScriptová knižnice pro vývoj front-endových aplikací. V práci jsem zabezpečil, aby aplikace využívala protokol HTTPS a prostřednictvím vytvoření service workera a manifestu, zabezpečila dodatečnou funkcionalitu. Následně testuji splněné požadavky prostřednictvím vývojářského nástroje Google Lighthouse.

Klíčová slova: React, Progresivní webová aplikace, Service worker, Manifest, HTTPS, Google Lighthouse

Design and implementation of Progressive web application

Abstract

This bachelor thesis focuses on design and implementation of Progressive web application. This application is written in React, open-source JavaScript library used for developing front-end applications. In this thesis is used HTTPS protocol for secure communication. By creating service worker and manifest files, we provided the requirements for additional functionality. Then all the requirements are tested by developer tool Google Lighthouse.

Keywords: React, Service worker, Manifest, Progressive web application, HTTPS, Google Lighthouse

Obsah

Seznam obrázků

Seznam tabulek

Seznam použitých zkratek

FCM – Firebase Cloud Messaging

SPA – Single Page Application

PWA – Progresívní webová aplikace

MITM – Man in the Middle

DOM – Document Object Model

WAI – Web Accessibility Initiative

WCAG – Web Content Accessibility Guidelines

SEO – Search Engine Optimization

URL – Uniform Resource Locator

iOS – iPhone Operating System

1 Úvod

Technologie, v oblasti webového vývoje, se neustále mění a přizpůsobují se aktuálním potřebám uživatelů. Se stále narůstajícím počtem mobilních zařízení, bylo nevyhnutné vytvořit technologii, která by mohla konkurovat nativním mobilním aplikacím. Progresívní webové aplikace přicházejí s řešením, jak poskytnout funkce a výhody nativní aplikace v internetovém prohlížeči. Poskytují webovým aplikacím možnosti, jak pracovat v režime off-line, posílat uživatelům push notifikace anebo možnost nainstalovat si ji přímo do paměti zařízeni.

Skutečně progresívní na těchto aplikacích, je postupné zpřístupnění funkcionalit na základě výkonu zařízeni, rychlosti připojeni anebo verze internetového prohlížeče. Jde ale o aplikaci, která se výkonově chová jak nativní i když podmínky internetového připojení nejsou optimální jak ve většině případů. Jeden ze způsobů, jak zabezpečit rychlost načítání, je prostřednictvím zálohováni časti aplikace do paměti zařízeni. Jádro celé progresivní aplikace je ve funkci service workera, která dokáže pracovat mimo prohlížeč a poskytuje širokou škálu funkcí díky kterým se odlišuje progresivní webová aplikace od „klasické“ webový aplikace.

Aby tohle všechno mohlo fungovat, musí aplikace obsahovat zabezpečené připojení, protože funkce, které obsahuje service worker, by mohli lehko lidé zneužít k útoku na aplikaci a mohlo by dojít k průlomu zabezpečení.

Díky vývoji progresivních webových aplikací dokázali firmy zvýšit svůj zisk a dokázat, že tento druh vývoje je ve mnoha případech optimálním řešením při vývoji nové aplikace.

2 Cíl práce a metodika

2.1 Cíle práce

Cílem téhle bakalářské práce, je vytvořit funkční aplikaci, prostřednictvím open-source front-endové knižnice JavaScriptu, React. Následně, implementovat všechny požadované funkce technologií Progresivních webových aplikací, vytvořením service workera a manifestu a taktéž zabezpečení bezpečného šifrovaného připojení přes protokol HTTPS. Všechny podmínky stanovené v Google PWA checkliste následně otestovat přes vývojářský nástroj Google Lighthouse.

2.2 Metodika

Teoretická část bakalářské práce bude obsahovat analýzu a rešerše odborných zdrojů z oblasti Progresivních webových aplikací a knihovny React. V praktické časti bakalářské práce budou, na základě informací z teoretické časti, aplikované postupy vývoje aplikace na zabezpečení seznamu požadavek pro PWA a využitím nástroje budou této požadavky otestované.

3 Úvod do problematiky PWA

Neexistuje oficiální definice pro Progresívní webové aplikace. Pojem Progresivní webová aplikace, nebo ve zkratce PWA, se začal používat v roce 2015. Většinou by měl splňovat určitá generická kritéria. Ve wikipedií, je termín PWA definovaný jako webové aplikace, které jsou jak obyčejné web stránky, ale uživatel je vnímá jako klasické aplikace, anebo nativní mobilní aplikace. Tenhle typ aplikace se pokouší kombinovat funkce moderních internetových prohlížečů s výhodami mobilního rozhraní. Na internetovém portálu developer.mozilla.org, je uložená sbírka webové technické dokumentace MDN Web Docs, anebo kdysi MDN (Mozilla Developer Network). Na téhle stránce, je termín PWA definovaný, jako Progresívní webové aplikace, které využívají moderní webové API (Aplikační programové rozhraní), spolu se strategií na progresivní vylepšení pro poskytnutí multiplatformních webových aplikací s tak měr nativním aplikačním prostředím.[8][9] Nativní aplikace na mobilním zařízení poskytují mnohem lepší uživatelskou zkušenost. Po prvotním stáhnutí této aplikace, je čas načtení obsahu tak měr nulový. Při zhoršeném a pomalém připojení na internet aplikace ví poskytnout většinu svojí funkcionality. Díky těmto bodům se uživatelé mohli spolehnout na kvalitní a bezproblémovou uživatelskou zkušenost. Což má za následek i extrémně vysoký počet aplikací. Podle internetového portálu statistica.com kombinovaný počet aplikací na čtyřech největších trzích byl 6.8 milionu aplikací k prvnímu čtvrtletí 2021.[3]

V roce 2015 Alex Russel, který pracoval pro Google jako inženýr, který vyvíjel internetový prohlížeč Google Chrome, publikoval článek, který jako první definoval vlastnosti PWA. Tyto body by se daly shrnout do tří hlavních bodů:[5]

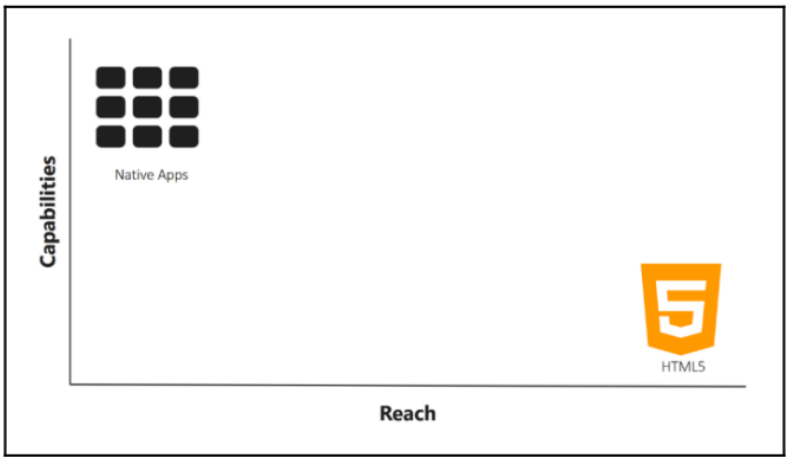
Rychlé – schopné vykreslit objekty uživatelského rozhraní na zařízení uživatele během méně než pár vteřin.

Spolehlivé – i přes horší připojení k internetu nebo na starých mobilních zařízeních poskytnout uživateli spolehlivé aplikační prostředí.

Poutavé – díky push notifikacím uživatelé jsou obeznámeni co se děje v aplikaci i když zrovna internetový prohlížeč není spuštěn. Uživatelé si umí nainstalovat aplikaci přímo na jejich domovskou obrazovku. Vývojáři umí nastavit vzhled ikony nebo vzhled obrazovky při spouštění.[4]

3.1 Výhody PWA

Mezi největší výhodu, jakou vývoj progresivní webové aplikace má je počet potencionálních uživatelů aplikace. Podle statistica.com počet mobilních zařízení připojených k internetu je téměř 15 miliard.[4] To představuje také tento graf, který zobrazuje rozdíl dosahu a funkcionality mezi webovými a nativními aplikacemi. Symbol HTML5 představuje webové aplikace. Na rozdíl od nativních aplikací ty webové se snadno najdou. Vyhledávací algoritmus je lepší a jeho uživatelské rozhraní je navrženo způsobem, aby uživatel věděl, co hledat. Algoritmus vyhledávání zadanou frázi vyhledává i na podstránkách stránek.[5]



Obrázek 1: Graf dosahu a možností

Další velká výhoda je možnost přidat potřebné funkcionality PWA do již existující webové aplikace postupně a dle potřeby. Kolekce těchto technologií lze aplikovat na internetové stránky bez ohledu na jakých technologiích a službách byla vyvíjena. To, co umožňuje technologií PWA komunikovat s jakýmkoli základem je to, že jsou založeny na HTML, CSS a JavaScriptu, jako téměř všechny webové stránky na internetu.[3] PWAs fungují všude, avšak rozlišují se v množství funkcionality, které umí na daném zařízení, přes daný prohlížeč poskytnout. Na nejstarších zařízeních s malým výkonem zobrazí pouze statickou webovou stránku a jak se zlepšuje výkon zařízení a je dostupnější novější verze prohlížeče tak poskytují širší funkcionalitu. Tento proces postupného zlepšování se jmenuje progresivní zlepšení.[4] Uživatelská zkušenost se zlepšuje prostřednictvím kolekce vlastností, které dodávají aplikací funkce, aby upoutali uživatele bez ohledu na kvalitu a rychlost připojení k internetu.[4] Podle výzkumu, padesát tři procent uživatelů odejde z internetové stránky, trvá-li doba načítání déle než 3 sekundy. Technologie Service Worker

dokáže ukládat části webové aplikace do mezi paměti, což má za následek rapidní snížení délky načítání. Aplikace pak dokáže fungovat i v režimu off-line.[5]

3.2 Bezpečný kontext(HTTPS)

Podle definice uvedené na MDN Web Docs, bezpečný kontext je Okno nebo Pracovník (Worker), které splňují určitý stupeň ověření a zabezpečení. Hlavním úkolem je předejít bezpečnostním útokům MITM, aby nedošlo k získání přístupu k API a dalšímu ohrožení zařízení uživatele. Připojení na webovou aplikaci musí být prostřednictvím zabezpečeného připojení. Pokud je internetová stránka zajištěna umožní to provádět zabezpečené požadavky. Je to nutnost, neboť většina funkcí PWA jako geolokace nebo service workery jsou dostupné pouze prostřednictvím zabezpečeného připojení HTTPS.[10][11]

3.3 Service worker

Podle definice, service worker se chová jako proxy server, který je mezi webovou aplikací a internetovým prohlížečem a připojením na internet. Jednou z jeho úloh je vytvoření efektivního používání aplikace i v režimu offline. Zachycuje síťové požadavky a na základě toho provádí operace vzhledem k dostupnosti připojení k internetu. Aktualizuje soubory, které jsou na serveru a kontroluje, aby byly vždy aktuální. Poskytuje možnost běžet na pozadí v systému a umožňuje uživateli odesílat push-notifikace.[12] Service worker je script, který běží na pozadí a umožňuje zachytit síťové požadavky, push-notifikace, aktualizuje soubory a poskytuje offline režim pro aplikace. V případě, že jej internetový prohlížeč nepodporuje, tak se aplikace chová jako klasická internetová stránka.[3] Service worker nemá přístup k DOM. Běží na vlastním vláknu, aby neblokoval chod UI. Je to prostředník mezi aplikací a internetem. Provede operaci, na kterou byl na programován a výsledek pošle aplikaci.[4] Service worker je jádro aplikace PWA a bez něho by funkcionality PWA nebylo možné implementovat. Podpora technologie napříč internetovými prohlížeči se zlepšuje, aktuálně ho podporují téměř všechny a nejvíce funkcí podporuje Chrome.[13]

3.4 Manifest

Manifest webové aplikace je soubor ve formátu JSON, který prohlížeči sděluje informace o progresivní webové aplikaci a o tom, jak by se měla chovat, když je

nainstalována na počítači nebo mobilním zařízení uživatele. Typický soubor manifestu obsahuje název aplikace, ikony, které by měla aplikace používat, a adresu URL, která by se měla otevřít při spuštění aplikace. Soubory Manifest jsou podporovány ve všech hlavních internetových prohlížečích s výjimkou poskytování pouze částečné podpory v Safari.

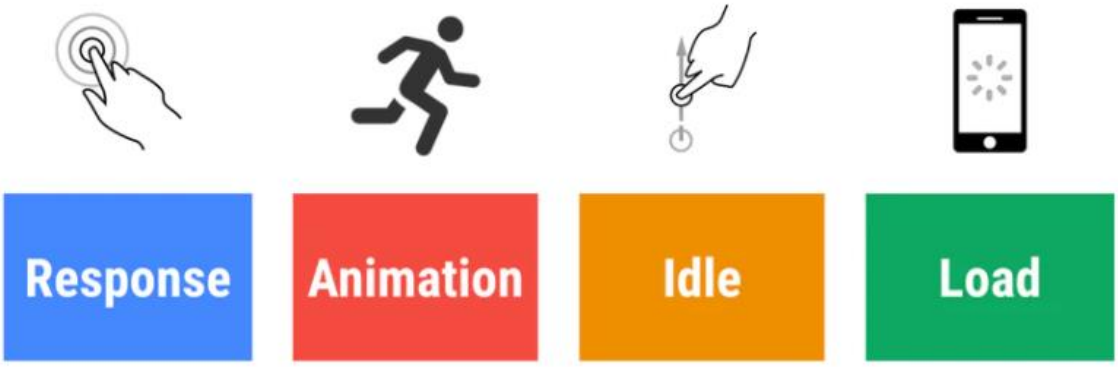
3.5 Kostra aplikace

Kostra aplikace je minimální obsah kódu pro soubory HTML, CSS a JavaScriptu, nutné pro zobrazení a funkčnost uživatelského rozhraní. Poskytnutí responzivního chodu aplikace a konzistentnímu výkonu aplikace, a to i při uložení do paměti počítače, dojde-li ke ztrátě připojení na internet. Na základě toho, toto rozhraní není na novo staženo z internetu při opětovné návštěvě uživatele, ale pouze nezbytný obsah aplikace. Tato daná architektura pro vývoj aplikace je jeden ze způsobů poskytnutí rychlých časů načítání a poskytnutí uživatelské zkušenosti na úrovni nativní aplikace. Pro aplikace, které využívají způsob vykreslování obsahu na obrazovce uživatele prostřednictvím techniky SPA, je daná architektura výborný způsob, jak to zajistit. Prostřednictvím service workera se uživatelské rozhraní uloží do paměti zařízení a dynamicky zobrazuje obsah aplikace prostřednictvím JavaScriptu. Táto architektura je vhodná pro typ aplikací a internetových stránek, jejichž rozhraní se nemění. Tento koncept sám o sobě nepodněcuje využití konkrétního frameworku. Množství moderních JavaScriptových frameworků a knihoven vede vývojáře k rozdělení prvků mezi navigační prvky aplikace, prvky, které zobrazují obsah a samotný obsah aplikace, který se na základě interakce uživatele s aplikací dynamicky mění. Výhodou je, že při opakovaných návštěvách aplikace jsou časy načítání extrémně rychlé. To vede k uživatelské zkušenosti na úrovni nativní aplikace s možností chodu v režimu offline. Data uložená v paměti zařízení jsou opětovně dynamicky vykreslována na obrazovce, díky čemuž je docíleno nižší spotřeby mobilních dat. [15]

3.6 RAIL Model

RAIL je model vývoje aplikace zaměřený na uživatele, který poskytuje strukturu pro optimální výkon. Tento model dělí uživatelské interakce na akce a pomáhá definovat jejich optimální časy provedení. RAIL reprezentuje čtyři rozdílné aspekty životních cyklů webové aplikace. Jsou to odezva (response), animace (animation), nečinnost (idle) a načítání (load). Uživatel má rozdílná očekávání, jak rychle se tyto akce provedou v závislosti na

kontextu.[16]



Obrázek 1: Kategorie modelu RAIL

V obecnějším smyslu je doba odezvy rozdělena do 3 limitních kategorií:

0.1 sekundy: Je to limit, po který má uživatel pocit, že operace jsou provedeny instantně, a proto žádný další feedback kromě výsledného obsahu není nutný. Jsou to objekty na obrazovce, se kterými se přímo manipuluje, jako například zvýraznění sloupce v tabulce. V ideálním případě by takovou odezvu měla i akce seřazení tabulky, aby nedocházelo k pocitu u uživatele, že danou tabulku řadí on.

1 sekunda: Do jedné sekundy je čas, kdy myšlenkový pochod uživatele zůstává nepřerušený i když odezva je patrná. Není potřebný dodatečný feedback, ale uživatel má pocit, že zobrazený obsah není přímo prostřednictvím jeho zadávané akce, ale výsledek operace, kterou muselo zařízení zpracovat. V případě seřazení tabulky, je-li to čas do 1 sekundy, se program jeví jako rychlý a responzivní.

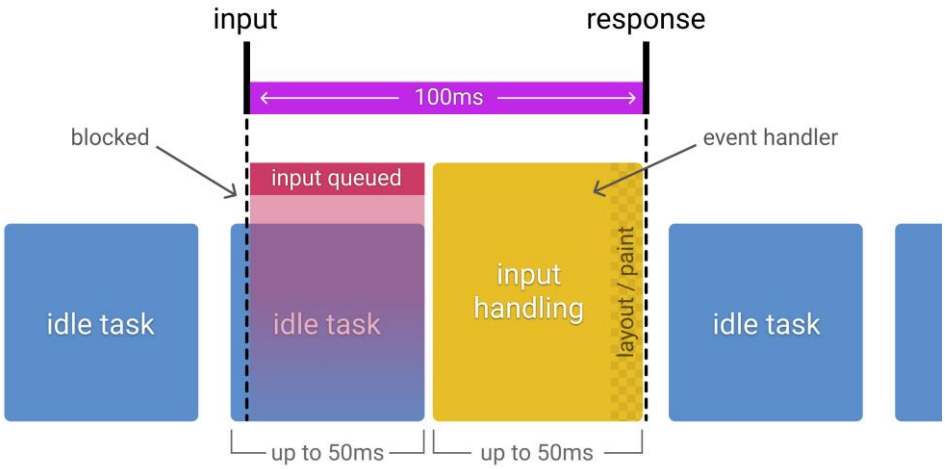
10 sekund: Tento limit je hranice k udržení pozornosti uživatele. Pro odezvy déle než 10 sekund by měl uživatel možnost provádět jiné akce, zatímco na pozadí by program dokončil danou akci. Feedback během čekání je obzvlášť důležitý pro uživatele, aby věděl kdy došlo k dokončení, ideálně prostřednictvím procentuálního zobrazení průběhu. Daná akce předpokládá změnu v myšlení uživatele na základě zahájení nového úkolu.[34]

RAIL model dále definuje ještě jeden limit od 0 do 16 milisekund, který je zaměřen na animace v aplikaci. Uživatel vnímá animace za plynulé, pokud 60 obrazů za sekundu se vykreslí na obrazovce, to znamená 16 milisekund na obraz.

3.6.1 Odezva (Response)

Časový limit pro odezvu je pod hranicí 100 milisekund. Ale model říká, aby čas byl pouhých 50 milisekund. Je to proto, že kromě zaregistrování akce od uživatele běží na pozadí

aplikace procesy. V tomto důsledku se předpokládá, že doba na zpracování a vykreslení je 50 milisekund. Tento čas je optimální pro akce objektů aplikace, jako například stlačení tlačítka nebo zapnutí animace. Pro akce trvající déle, než 50ms je ideální zpětná vazba.



Obrázek 2: Grafické zobrazení odezvy

3.6.2 Animace (Animation)

Pro zajištění plynulosti se vyžaduje rychlost času vykreslení do 16ms na jeden snímek. Čas vyhrazený pro aplikaci je pouhých 10ms, neboť doba vykreslení v prohlížeči je 6ms. V době vykreslení aplikace by měla provádět pouze tento proces, v horším případě počet procesů na pozadí minimalizovat.

3.6.3 Nečinnost (Idle)

Cíl je maximalizovat počet procesů, které aplikace zpracovává v období nečinnosti uživatele. Čas na zpracování by ovšem neměl být větší než 50ms, aby nedošlo k narušení schopnosti zpracování vstupu od uživatele. V případě, že dojde k interakci, měla by mít největší prioritu v systému.

3.6.4 Načítání (Load)

Cílem je aplikaci optimalizovat relativně k výkonu zařízení a rychlosti připojení k internetu individuálního uživatele. Při prvotním načtení aplikace, by se měl čas pohybovat pod hranicí 5 vteřin pro střední třídu mobilních zařízení s rychlostí připojení 3G. Opětovné načítání by nemělo přesáhnout dobu delší než 2 sekundy.[16]

3.7 Rychlost a výkon aplikace

Uživatel vnímá rychlost, jakým aplikace zpracovává operace individuálně, na základě výkonu jeho zařízení a podle rychlosti připojení na internet. Na základě studie, kde bylo přes deset tisíc mobilních zařízení, byla průměrná doba načítání internetové stránky devatenáct vteřin přes 3G připojení. Při srovnání stránek jejichž doba načítání byla pět vteřin jejich zisk z reklam byl dva krát vyšší než u stránek s dobou načítání devatenáct vteřin. Navíc byla o 25 % vyšší míra zobrazení reklam a o 70 % vyšší průměrná doba, kterou uživatel strávil na stránce. Pokud doba načítání trvá déle než 3 sekundy, 53 % uživatelů zavře danou stránku. Je to dáno také tím, že jeden ze dvou lidí očekává čas načítání menší než dvě sekundy. Z toho 46 % lidí zařadilo čekání na načtení stránky mezi vlastnosti stránky jako nejméně přijatelné. Jsou tři hlavní faktory, které zpomalují čas načítání internetové stránky: velikost souborů stránky, počet požadavků na server a v jakém pořadí se jednotlivý obsah na stránce načte. Průměrná velikost dat obsahu na internetových stránkách je 1.49MB, což představuje délku čekání než se obsah načte sedm sekund přes 3G připojení. Internetové stránky mají průměrný počet 214 požadavků na server, z toho téměř polovina je spojena s reklamou.[17] RAIL Model a PWA poskytují vhodné způsoby pro zrychlení načítání. Jeden ze způsobů je eliminovat zdroje blokující renderování. Cíl je snížit počet URL odkazů, a to vymezením nezbytných URL odkazů, seřadit je za sebou a odložit načítání zbývajících odkazů až po prvotním vykreslením aplikace.[18] Mezi další způsoby optimalizace času načítání aplikace je i technologie lazy-loading (líné načítání). Podporuje ji většina moderních internetových prohlížečů. V případě, že technologii prohlížeč nepodporuje, ignoruje ji bez komplikací. Obrázky jsou nejvíce vyžadovaným typem obsahu internetových stránek. Všechny obrázky, které jsou ihned viditelné na obrazovce uživatele se načítají normálně a zbytek se načte, když se dostanou na viditelnou část obrazovky. Obrázky jsou načteny dostatečně v předstihu, na základě rychlosti internetového připojení uživatele, ještě před vstupem na obrazovku. Na zařízeních s rychlostí připojení k internetu 4G je velikost stránky, na které je obsah, 1250px a na zařízeních s rychlostí 3G je velikost 2500px.[19]

3.8 Responzivnost PWA

Podmínka, kterou musí splňovat jakákoli Progresivní webová aplikace, je být responzivní. Je to soubor postupů, které umožňují internetovým stránkám změnit jejich rozložení a vzhled tak, aby vyhovovaly různým šířkám obrazovky a rozlišení. Termín

responzivní design poprvé použil Ethan Marcotte v roce 2010 a popsal použití tří technik v kombinaci. Je důležité si uvědomit, že responzivní webový design není samostatnou technologií. Je to termín používaný k popsání přístupu k webovému designu nebo skupina osvědčených postupů, která se používá k vytvoření rozložení, které dokáže reagovat na velikost obrazovky zařízení. Jedna technika byla využití Media Querries, která umožňuje spustit sérii testů ke zjištění velikosti obrazovky uživatele a selektivně použít CSS k přizpůsobení stránky tak, aby odpovídala potřebám uživatele. Mezi moderní způsoby přizpůsobení rozložení obsahu obrazovky uživatele patři například Flexbox nebo Grid, které jsou responzivní automaticky. Poskytují jednodušší nastavení internetové stránky pro vytvoření responzivního designu. [20][21]

3.9 React

Je to open-source knihovna pro vytváření uživatelských rozhraní, kterou vytvořili a spravují vývojáři ve Facebooku. Umožňuje vytvářet zapouzdřené komponenty, které spravují svůj stav a spolu vytvářejí komplexní rozhraní. Spolu s knihovnou React-DOM, poskytuje React vývojářům možnost tvorby webových aplikací s širokou škálou možností jako u jiných frameworků. React-DOM vykresluje jednotlivé komponenty uživatelského rozhraní přes internetový prohlížeč na obrazovku uživatele. React-DOM využívá takzvanou virtuální DOM, kde ideální nebo virtuální představa uživatelského rozhraní uložená v paměti a synchronizovaná s reálnou DOM. Tento proces se nazývá usměrnění (reconciliation). Tento přístup umožňuje psaní kódu aplikace deklarativní. V Reacte se zadá stav, v jakém by mělo být uživatelské rozhraní a to zajistí, aby se DOM shodoval s tímto stavem. Tímto se abstrahuje manipulace s atributy, zpracování událostí a manuální aktualizace modelu DOM. [15][16][17]

3.9.1 Single-page-application (SPA)

React se využívá k tvorbě jedno stranových aplikací (SPA). Největší rozdíl oproti více stranové aplikací je, že celý obsah je na jedné stránce. Prostřednictvím skupiny pohledů umožňuje načtení rozdílného obsahu v aplikaci. [18]

Obrázek 3: Princíp SPA

V jedno stranové aplikaci, internetový prohlížeč načte pouze jeden HTML dokument. Jak se uživatel pohybuje v aplikaci, tak JavaScript překresluje nové uživatelské rozhraní na základě vybraného pohledu. React používá své objekty, které jsou odlišné od klasických objektů DOM. Objekt v React je popis, jak má vypadat objekt DOM. Poskytuje instrukce k vytvoření objektu pro DOM rozhraní internetového prohlížeče. [1]

3.9.2 Komponenty v Reacte

Komponenty jsou nezávislé a znovu použitelné části kódu. Mají stejný smysl jako funkce v JavaScriptu, ale jsou izolovány a vytvářejí HTML obsah prostřednictvím render funkce. Komponenty mohou být vykresleny na konkrétní objekt ve struktuře DOM prostřednictvím React-DOM. Komponenty nám umožňují znovu použít stejnou strukturu a poté můžeme tyto struktury naplnit různými daty. Data, která komponenty obsahují, jsou vkládána prostřednictvím hodnot zvaných vlastnosti. Syntakticky se shodují s argumenty funkcí v JavaScriptu. Existují dva typy komponent, funkční a třídní.[19][20]

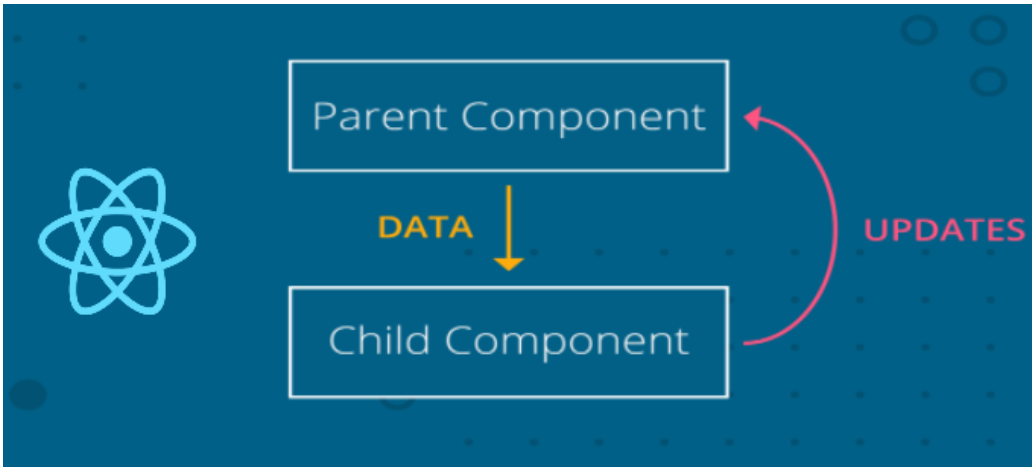
3.9.3 JSX

Je to rozšíření jazyka JavaScript, který nám umožňuje definovat objekty v Reacte pomocí syntaxe značkovacího jazyka s funkcemi a možnostmi programovacího jazyka

JavaScriptu. React zahrnuje skutečnost, že vykreslovací logika je neodmyslitelně spojena s logikou uživatelského rozhraní: jak se zpracovávají události, jak se stav časem mění a jak se údaje připravují k zobrazení. Namísto toho, aby uměle odděloval technologie rozdělováním značkovací a aplikační logiky do samostatných souborů, React odděluje odpovědnosti (Separation of Concerns – SoC) do jednotek zvaných „komponenty“, které obsahují obě logiky. Ne všechny internetové prohlížeče podporují nejnovější funkce a syntaxi JavasScriptu a žádný nepodporuje JSX. Proto prostřednictvím procesu kompilování překonvertujeme jazyk JSX na kód, který umí internetový prohlížeč interpretovat a zobrazit na obrazovce uživatele. React využívá kompilátor Babel.[1][16]

3.9.4 Jednosměrný tok údajů

Jednosměrný tok dat je technika, která se nachází primárně ve funkčním reaktivním programování. Znamená to, že data mají pouze jediný způsob, jak se mohou dostat k ostatním částem aplikace. React nepodporuje obousměrné vázání, aby se zachovala architekturu čistého toku dat. Hlavní výhodou tohoto přístupu je, že data proudí přes aplikaci jedním směrem, což umožňuje lepší kontrolu dat.



Obrázek 4: Princip jednosměrného toku dat

React ukládá data do objektu zvaný stav(state), na jehož základě se pak vykresluje a přizpůsobuje. Tento objekt přináleží vždy jen jedné komponentě. Jakékoliv provedené změny mohou mít vliv pouze na komponenty pod ním. Na jeho děti. Změna stavu komponenty nikdy neovlivní jeho rodiče ani sourozence, ovlivní to pouze děti. To je hlavní důvod, proč je stav ve stromu komponent často posouván nahoru, aby jej bylo možné sdílet mezi komponenty, které k němu potřebují přístup.[23][24]

3.10 Google Lighthouse

Google Lighthouse je open-source nástroj od Googlu pro měření jednotlivých předpokladů, které musí splňovat webová aplikace. Poskytuje sadu metrik, které pomáhají při vytváření PWA s téměř nativní uživatelskou zkušeností. Existují tři typy způsobů, jak použít tento nástroj. Prostřednictvím CLI, přes kartu Auditu v Chrome DevTools a jako plugin do prohlížeče Google Chrome. Momentálně Google Lighthouse umožňuje provádět testy, které rozděluje do 5 kategorií: Progresivní webové aplikace, výkon, dostupnost a nejlepší praktiky a SEO. Na základě výsledků testů vygeneruje bodové ohodnocení pro každou kategorii. První stupeň nastavení, které tento nástroj nabízí, je volba zařízení mezi stolním počítačem nebo mobilním zařízením. Další nastavení, které tento nástroj nabízí je simulované snižování výkonu zařízení (throttling) pro získání potřebných statistik, jak se bude chovat zařízení v suboptimálních podmínkách. Pro získání nejpřesnějších výsledků se doporučuje vybrat možnost vymazat obsah paměti. Tato možnost vymaže veškerou paměť ještě před spuštěním testů.[4][24]

V první kategorii Lighthouse kontroluje věci jako registrace service workera, absence režimu off-line, rychlost načítání aplikace na 3G připojení, nakonfigurovaní úvodní obrazovky nebo možnost instalace aplikace do paměti zařízení. Ve druhé kategorii, kde se testuje výkon aplikace, mezi zajímavé části těchto testů patří například první vykreslení aplikace nebo první interakce. Lighthouse pak nabízí příležitosti, jak urychlit aplikaci. Například prostřednictvím omezení odkazů v hlavičce aplikace, které brání vykreslení, kontroluje velikost a polohu obrázků, aby nebyly vykresleny mimo obrazovku.[4][25]

Třetí kategorie je přístupnost, kde nástroj provádí několik testů, aby ověřil, jak velmi přístupná daná aplikace je. Přístupnost znamená dostupnost aplikace pro všechny, a aby ji mohl ovládat kdokoli. WAI připravila podrobné pokyny, které se nazývají WCAG a jsou rozděleny do 4 kategorií: vnímatelnost, funkčnost, pochopitelnost a robustnost. Zlepšení přístupnosti vede ke zvýšení počtu lidí s handicapem, který budou schopní aplikaci používat. V kontextu PWA a přístupnosti zabezpečení chodu aplikace i při velmi pomalém nebo nespolehlivém internetovém připojení, zlepší přístupnost a umožní většímu počtu lidí aplikaci používat. Čtvrtá kategorie jsou nejlepší praktiky, kde nástroj kontroluje různé druhy a způsoby použití technologií podle zažitých pravidel. Je to soubor 16 ověřených postupů, které se zaměřují především na bezpečnostní aspekty a moderní standardy webového vývoje. Lighthouse analyzuje, zda se používají HTTPS a HTTP/2, kontroluje, zda data pocházejí z bezpečných zdrojů, a hodnotí bezpečnost používaných knihoven JavaScriptu. A poslední

kategorie je SEO, kde testy zjišťují, jak dobře mohou webové stránky nebo aplikace prohledávat vyhledávací nástroje a zobrazovat je ve výsledcích vyhledávání. [4][32][33]

3.11 Shrnutí

V této kapitole jsem se věnoval problematice Progresivních webových aplikací, kde jsem vysvětlil, co přesně tento pojem zahrnuje. Jasně jsem si vymezil, co všechno musí taková aplikace splňovat, aby mohla být považována za PWA. Přiblížil jsem její výhody, zaměřil se na kostru aplikace a doporučený model vývoje RAIL. Dále jsem přiblížil JavaScriptovou knihovnu React, ve které aplikaci budu vyvíjet. Nakonec jsem představil jeden z nástrojů pro testování parametrů dané aplikace Google Lighthouse, kterým budu aplikaci testovat.

4 Interaktivní školský diář

V praktické části mojí práce navrhnu a naprogramuji aplikaci pro studenty. Konkrétně se bude jednat o interaktivní školský diář, kde využiji benefity vývoje progresivní webové aplikace. Cílem vývoje této aplikace se bude, co nejvíc přiblížit funkcionalitou ale i vzhledově nativní aplikaci. Využiji dostupné API a ověřím jejich schopnost, jak věrně umí vykompenzovat fakt, že se jedná o webovou aplikaci. Prostřednictvím Google Lighthouse provedu měření, na základě kterého objektivně zhodnotím část splněných požadavek. Kód aplikace bude přístupný na „https://github.com/lacinko/interactive-school-planner“.

4.1 Požadavky

4.1.1 Funkční požadavky

FP_01: Jako student, chci mít možnost si vytvořit účet v aplikaci, abych se věděl přihlásit do aplikace a mohl si prohlédnout své hodiny, úkoly a připomínky.
FP_02: Jako student, chci mít možnost si zobrazit celý školský semestr ve formě seznamu, kde budou jednotlivé dni seřazené do logických celků v podobě týdnů.
FP_03: Jako student, chci po stlačení konkrétního dne zobrazit detailní časový přehled dne, rozděleného do logických celků v podobě hodin. Jednotlivé hodiny budou mít přiděleny mnou zadané hodiny a jejich případné úkoly.
FP_04: Jako student, chci mít možnost si zadat do diáře své vlastní předměty, jejich časové rozpětí, konkrétní čas vyučování a typ hodiny.
FP_05: Jako student, chci mít možnost, zadat si úkoly k jednotlivým předmětům na konkrétní datum.
FP_06: Jako student, chci vidět seznam předmětů a jejich úkolů na samostatné obrazovce.

4.1.2 Vzhledové požadavky

VP_01: Aplikace je responzivní a poskytuje plnohodnotnou uživatelskou zkušenost. Funguje na většině moderních prohlížečích, na stolních ale i mobilních zařízeních.
VP_02: Aplikace se zobrazuje správně na širokém spektru velikosti obrazovek.

4.1.3 PWA požadavky

PWAP_01: Aplikaci je možné si nainstalovat do paměti zařízení a spouštět pomocí ikony na domovské obrazovce zařízení.

PWAP_02: Aplikace funguje off-line, v případě výpadku připojení k internetu aplikace funguje bez přerušení.
PWAP_03: Aplikace odesílá notifikace uživatelům o úkolech na předměty.

4.2 Uživatelské rozhraní

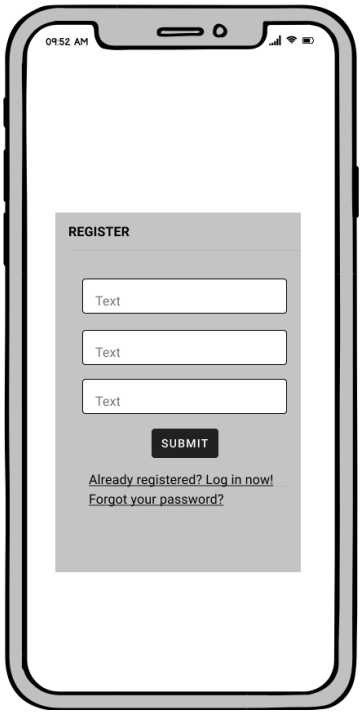
4.2.1 Obrazovky

Seznam všech obrazovek použitých v aplikaci:

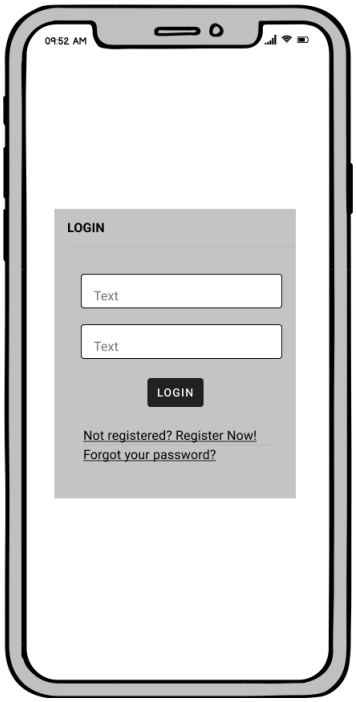
a) **Obrazovka přihlášení** – slouží na k přihlášení do aplikace po zadání uživatelských údajů.
b) **Obrazovka registrace** – slouží k registraci uživatelů.
c) **Obrazovka zapomenutého hesla** – slouží k obnově hesla, když si ho uživatel nepamatuje.
d) **Obrazovka seznamu semestru –** zobrazuje semestr ve formě týdnů, ale také podrobný denní přehled rozdělený do hodin, kde zobrazuje hodiny daného studenta.
e) **Obrazovka na přidáváni předmětů –** slouží k přidávání předmětů do školního diáře.
f) **Obrazovka na přidávání úkolů –** slouží k přidávání úkolů k jednotlivým předmětům.
g) **Obrazovka listovaného přehledu předmětů a úkolů –** zobrazuje listovaný přehled předmětů semestru a jejich úkolů.

4.3 Wireframe

K zobrazení jednotlivých obrazovek a rozdělení prvků uživatelského rozhraní jsem vytvořil wireframe návrhy. Tyto návrhy jsem vytvořil v programu Figma. Typ wireframe modelu jsem si zvolil high fidelity. Jedná se model s vysokou přesností, který je velmi blízký výslednému produktu.

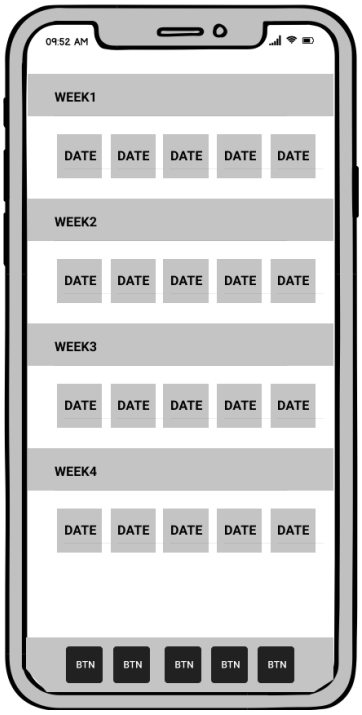


Obrázek 5: Obrazovka registrace

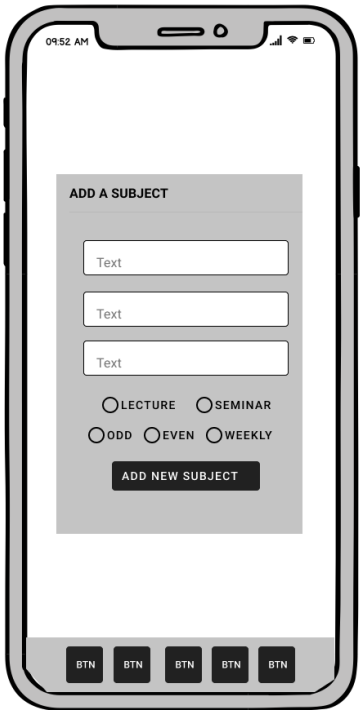


Obrázek 6: Obrazovka přihlášení

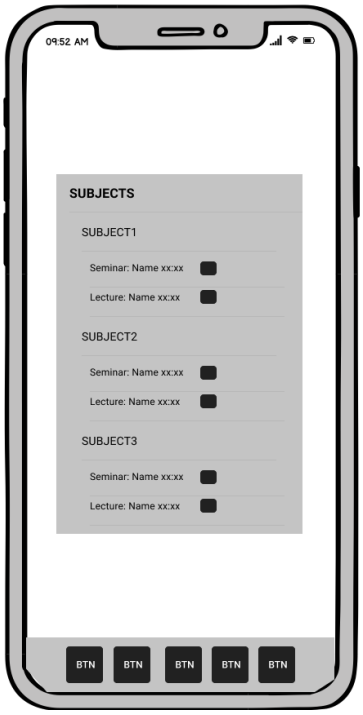
Obrázek 7: Obrazovka pro obnovení hesla



Obrázek 8: Obrazovka seznamu semestru



Obrázek 9: Obrazovka pro přidávání předmětů



Obrázek 10: Obrazovka seznamu předmětů

Obrázek 11: Obrazovka pro přidávání úloh

4.4 Architektura aplikace

4.4.1 Frontend

Ke tvorbě UI byla použita knižnice ReactJS, pro správu dat v aplikaci byla použitá knižnice Redux Toolkit, směrování v aplikaci zabezpečuje knižnice Reach Router. Ke tvorbě časových objektů byla využita knižnice Moment, generovaní UUID bylo pomocí knižnice UUID. Vzhled jednotlivých elementů je stylizovaný pomocí CSS frameworku Tailwind a prostřednictvím preprocesoru CSS Sass.

4.4.2 Backend

Pro hostování aplikace jsem si zvolil doménu, kterou poskytuje služba Firebase. Pro autentifikaci jsem si zvolil Firebase Authentication. Databáze aplikace je NOSQL, dokumentová databáze Firestore.

4.4.3 Souborová struktura

- src/components – obsahuje komponenty uživatelského rozhraní
- src/logic – obsahuje soubory, kde je napsána aplikační logika a konfigurační soubory
- src/pages – obsahuje jednotlivé stránky, které jsou vykreslovány v aplikaci
- src/store – obsahuje všechny soubory pro správu dat Redux Toolkit

- src/styles – obsahuje stylizační soubory preprocesoru Sass a konfigurační soubory frameworku Tailwind
- public/images – obsahuje ikony aplikace
- public – obsahuje json soubor manifest, soubory service workera a soubory pro offline režim

4.4.4 Zhotovení aplikace

Ke zhotovení aplikace je použit nástroj ViteJS. Je to předkonfigurovaný nástroj Rollup, který zabalí soubory aplikace do vysoce optimalizovaných souborů pro webový prohlížeč. Ke kontrole kvality kódu byl použit nástroj ESLint, který analyzuje kód a upozorňuje na syntaktické a stylistické chyby. Na automatické formátování programovacího jazyka byl použit nástroj Prettier.

4.5 Implementace funkčních požadavků

V této sekci se budu věnovat splnění požadavků, které jsem si definoval v předchozí sekci. Jejich splněním vytvořím uživatelské rozhraní pro aplikaci.

4.5.1 Autentifikace

Aplikace poskytuje tři obrazovky, na kterých si uživatel umí vytvořit účet, umí se přihlásit a ví si nechat přeposlat nové heslo, pokud ho zapomněl. Vytvořený formulář pro sběr přihlašovacích údajů je dočasně uloží do stavu a pak prostřednictvím thunk API z Redux Toolkit je asynchronně odešle na server. Na serveru je zpracuje Firebase Authentification, který ověří jejich správnost a odešle odpověď zpět klientovi. Firebase ukládá přihlašovací údaje do lokální cache paměti, díky čemuž zůstane uživatel přihlášen i když dojde ke ztrátě připojení na internet. Ukázka obrazovek v příloze č.1.[30]

4.5.2 Zobrazení semestru

Přehled celého semestru je zobrazen prostřednictvím čtyř komponent: Semestr, Week, Day, Hour. Po zadání data začátku semestru a počtu týdnů v semestru, aplikace vygeneruje strukturu. Je dělená do logických celků od týdnů, dnů až na hodiny. Jednotlivá komponenta vykresluje na displeji uživatele obsah. Nejmenší logické celky, hodiny v sobě nesou informace, které obsahují údaje jako: ID, čas a subjekt. Stylizování obsahu je prostřednictvím CSS tříd v Tailwindu. Po kliknutí na konkrétní den, je zobrazen podrobný

rozpis dne v čtyřiceti-pěti minutových intervalech. Pokud daný interval obsahuje předmět nebo úlohu je zobrazen na obrazovce. Ukázka obrazovek v příloze č.2

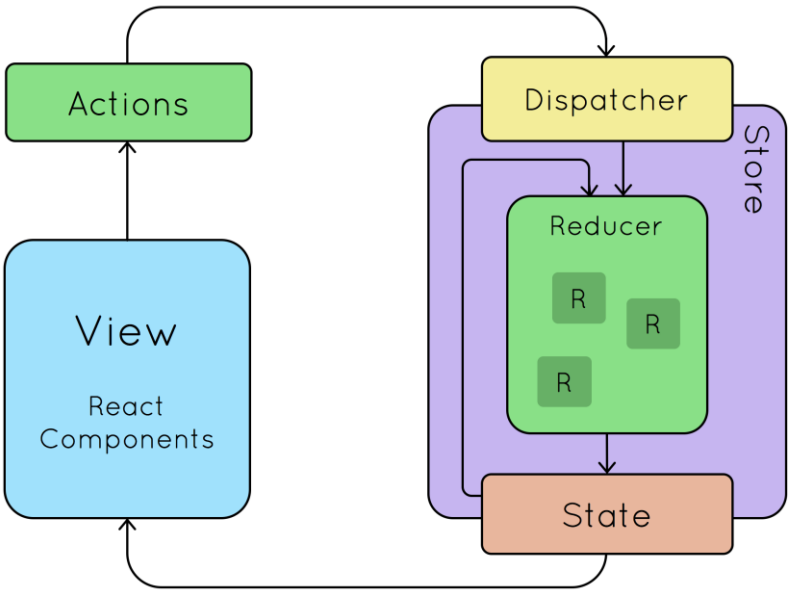
4.5.3 Přidání předmětů a přidání úloh

V aplikaci jsou pro přidání studentových předmětů a úkolů, vytvořeny dvě separátní stránky. Obě stránky obsahují své komponenty, ve kterých je vytvořen formulář pro zadávání podrobností ohledně předmětů a úkolů. Tyto komponenty svůj obsah od uživatele ukládají v Redux store, což je vlastně objekt, ve kterém jsou uložena všechna data aplikace. Ukázka obrazovek v příloze č.3

4.5.4 Redux Toolkit

Redux je knihovna pro správu stavu v aplikaci. Funguje na principu poskytování centralizovaného úložiště pro data pod názvem store. To umožňuje jednotlivým komponentům přistupovat k těmto datům přímo. Model Reduxu bychom mohli rozdělit do následujících kroků:

1. Uživatel interakcí s komponentou vyvolá změnu stavu.
2. Když je vyžadována změna stavu, komponent zavolá akci.
3. Reducer zpracuje akci a aktualizuje stav v store na základě obsahu akce.
4. Komponenta vykresluje obsah, který má ze storu a dojde-li k jeho změně, tak překreslí komponentu.



Obrázek 12: Diagram fungování Reduxu[31]

Redux Toolkit je oficiální doporučený přístup pro psaní Redux logiky. Poskytuje zjednodušení a výrazné snížení množství kódu pro sestavení logiky. [31]

4.5.5 Seznam předmětů

V aplikaci si umí student zobrazit seznam přidaných předmětů pro daný semestr. Je to na samostatné stránce, kde je komponenta SubjectList, která tyto předměty zobrazuje. Komponenta si data tahá z Redux storu.

4.6 Implementace vzhledových požadavků

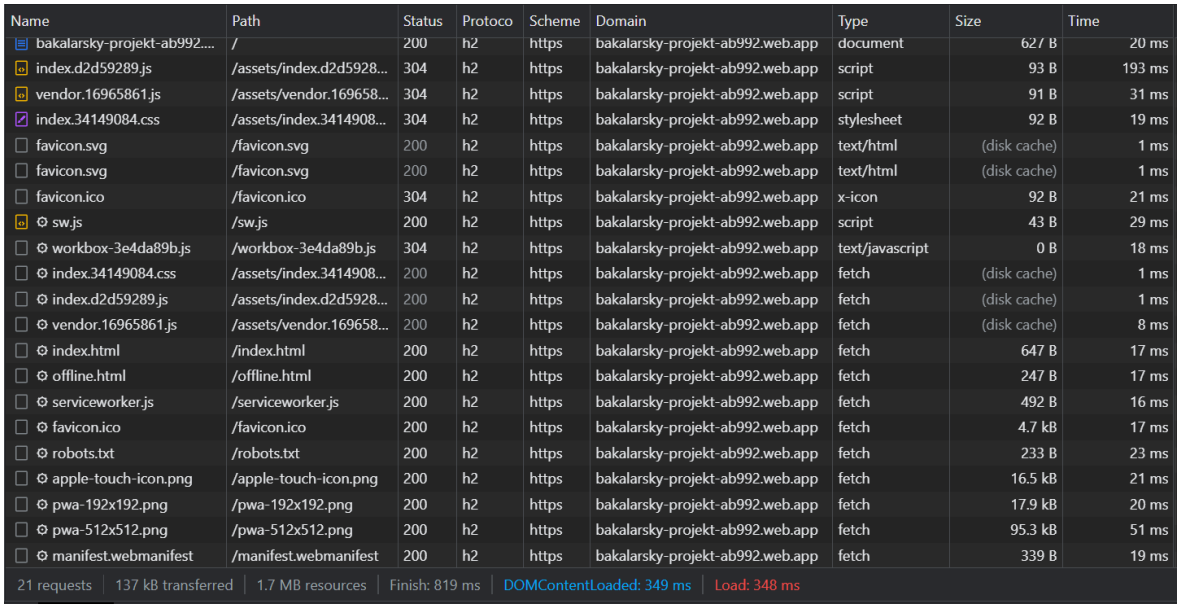
V aplikaci jsou vytvořena dvě uživatelská rozhraní, jedno pro velké obrazovky, jako monitory na noteboocích nebo samostatné monitory k stolovým počítačům, a jedno pro malé obrazovky jako jsou mobily nebo tablety. Aplikace je vyvíjena postupem mobile first, což znamená, že nejprve se vytvoří rozhraní pro menší zařízení a potom přes media querries v CSS se aplikují třídy pro větší zařízení.

4.7 Implementace PWA požadavků

Pro úspěšnou implementaci požadavků kategorie Progresivních webových aplikací, je nutné nejprve vytvořit architekturu a využití technologií pro podporu PWA. Backend technologie, kterou jsem si vybral, mi umožňuje dosáhnout mého cíle, který jsem si stanovil. Firebase je platforma pro vývoj webových a mobilních aplikací na Google Cloud Platform. Jejich model je Backend as a Service, takže nabízí širokou škálu služeb jako: Firestore databázi, Firebase autentifikaci, Firebase hosting, Firebase Cloud Messaging pro posílání zpráv.

4.7.1 HTTPS anebo HTTP/2

Jedna z podmínek, jak umožnit nainstalování webové aplikace do paměti zařízení, je aplikaci poskytnout uživatelům přes protokol HTTPS nebo HTTP/2. HTTP/2 je nová verze protokolu HTTP. Oproti staré verzi se jedná o binární protokol, který odesílá striktně definované binární struktury zvané rámce, pro přenos informací. I když daný standard nevyžaduje šifrování přenosu, většina moderních prohlížečů podporuje přenos protokolu HTTP/2 pouze přes TLS pod názvem h2.



Name	Path	Status	Protoco	Scheme	Domain	Type	Size	Time
bakalarsky-projekt-ab992....	/	200	h2	https	bakalarsky-projekt-ab992.web.app	document	627 B	20 ms
index.d2d59289.js	/assets/index.d2d5928...	304	h2	https	bakalarsky-projekt-ab992.web.app	script	93 B	193 ms
vendor.16965861.js	/assets/vendor.169658...	304	h2	https	bakalarsky-projekt-ab992.web.app	script	91 B	31 ms
index.34149084.css	/assets/index.3414908...	304	h2	https	bakalarsky-projekt-ab992.web.app	stylesheet	92 B	19 ms
favicon.svg	/favicon.svg	200	h2	https	bakalarsky-projekt-ab992.web.app	text/html	(disk cache)	1 ms
favicon.svg	/favicon.svg	200	h2	https	bakalarsky-projekt-ab992.web.app	text/html	(disk cache)	1 ms
favicon.ico	/favicon.ico	304	h2	https	bakalarsky-projekt-ab992.web.app	x-icon	92 B	21 ms
sw.js	/sw.js	200	h2	https	bakalarsky-projekt-ab992.web.app	script	43 B	29 ms
workbox-3e4da89b.js	/workbox-3e4da89b.js	304	h2	https	bakalarsky-projekt-ab992.web.app	text/javascript	0 B	18 ms
index.34149084.css	/assets/index.3414908...	200	h2	https	bakalarsky-projekt-ab992.web.app	fetch	(disk cache)	1 ms
index.d2d59289.js	/assets/index.d2d5928...	200	h2	https	bakalarsky-projekt-ab992.web.app	fetch	(disk cache)	1 ms
vendor.16965861.js	/assets/vendor.169658...	200	h2	https	bakalarsky-projekt-ab992.web.app	fetch	(disk cache)	8 ms
index.html	/index.html	200	h2	https	bakalarsky-projekt-ab992.web.app	fetch	647 B	17 ms
offline.html	/offline.html	200	h2	https	bakalarsky-projekt-ab992.web.app	fetch	247 B	17 ms
serviceworker.js	/serviceworker.js	200	h2	https	bakalarsky-projekt-ab992.web.app	fetch	492 B	16 ms
favicon.ico	/favicon.ico	200	h2	https	bakalarsky-projekt-ab992.web.app	fetch	4.7 kB	17 ms
robots.txt	/robots.txt	200	h2	https	bakalarsky-projekt-ab992.web.app	fetch	233 B	23 ms
apple-touch-icon.png	/apple-touch-icon.png	200	h2	https	bakalarsky-projekt-ab992.web.app	fetch	16.5 kB	21 ms
pwa-192x192.png	/pwa-192x192.png	200	h2	https	bakalarsky-projekt-ab992.web.app	fetch	17.9 kB	20 ms
pwa-512x512.png	/pwa-512x512.png	200	h2	https	bakalarsky-projekt-ab992.web.app	fetch	95.3 kB	51 ms
manifest.webmanifest	/manifest.webmanifest	200	h2	https	bakalarsky-projekt-ab992.web.app	fetch	339 B	19 ms

21 requests | 137 kB transferred | 1.7 MB resources | Finish: 819 ms | DOMContentLoaded: 349 ms | Load: 348 ms

Obrázek 13: Tabulka načtených souborů při návštěvě domény.

Na obrázku č.13 je zobrazená tabulka zobrazující uložení aplikací, na internetové doméně „bakalarsky-projekt-ab992.web.app“, hosting zajišťuje Firebase a přenos je přes protokol h2. [32]

4.7.2 Manifest

Nezbytnou součástí je přítomnost souboru manifest, který obsahuje informace o webové aplikaci. Jsou to informace pro OS, aby věděli jako aplikaci při instalaci do systému pojmenovat, nebo cesty k ikonám. Pro vytvoření manifest souboru jsem si zvolil plugin VitePWA, který generuje soubor sám, stačí jen specifikovat zmíněné parametry. Soubor ukládá s příponou „.webmanifest“, která klasifikuje mime typ jako „application/manifest+json“.

4.7.3 Service Worker

Plugin VitePWA využívá knihovnu Workbox od Googlu, jejímž prostřednictvím vygeneruje soubor service workera. Tato knihovna zapouzdřuje Service Worker API a Cache Storage API a poskytuje vývojářům rozhraní pro snadnější práci. Hlavním úkolem je nahrání souborů cache paměti zařízení. Proces distribuce aktualizací aplikace a service workera, by se dal rozdělit do následujících kroků:

1. S novou verzí aplikace je vytvořen nový service worker s odlišným revizním číslem pro každý soubor.

2. Když uživatel otevře aplikaci, service worker je stažen a následně je porovnán s předchozí verzí. V případě, že je odlišný, je nainstalován a vyžaduje se obnovení aplikace.
3. Protože se revizní číslo souborů změnilo, tak soubory jsou odstraněny z cache paměti a jsou staženy znovu.
4. Následně aplikace je připravena k používání úplnou podporou v režimu off-line.

Aplikace obsahuje dva service workery, jeden je generován pluginem a druhý je napsán manuálně. Druhý service worker zajišťuje podporu push notifikací. Ukázka instalace na mobilní zařízení je v příloze č.4

4.7.4 Režim off-line

Service worker ukládá soubory specifikované ve funkci „precacheAndRoute()“, která je z knihovny Workbox. Tato funkce očekává množinu objektů. Každý objekt obsahuje hodnotu url a hodnotu revision.

```
e.precacheAndRoute(
  [
    { url: "assets/index.c999c70d.css", revision: null },
    { url: "assets/index.dc9c45b4.js", revision: null },
    { url: "assets/vendor.1ed398a3.js", revision: null },
    { url: "index.html", revision: "fdf6a743117cba3e0cdacda7c20ac67d" },
    { url: "offline.html", revision: "d9e981fa5ac1224ab845356fa8ff7116" },
    {
      url: "serviceworker.js",
      revision: "6634f97c6e23b90281534cef816fa5c1",
    },
    { url: "sw-auth.js", revision: "ba0915803d959a1479326d55d6fcb4e7" },
    { url: "sw-msg.js", revision: "kd03275803db192d2f78b8c65d6fxcd61" },
    { url: "favicon.ico", revision: "e40467793db192d2f78b8c6dfa928151" },
    { url: "robots.txt", revision: "cd9cd94aaa699e0a16e692b6bb16f672" },
    {
      url: "apple-touch-icon.png",
      revision: "7440bda34bde128b64a6c003a688ea4d",
    },
```

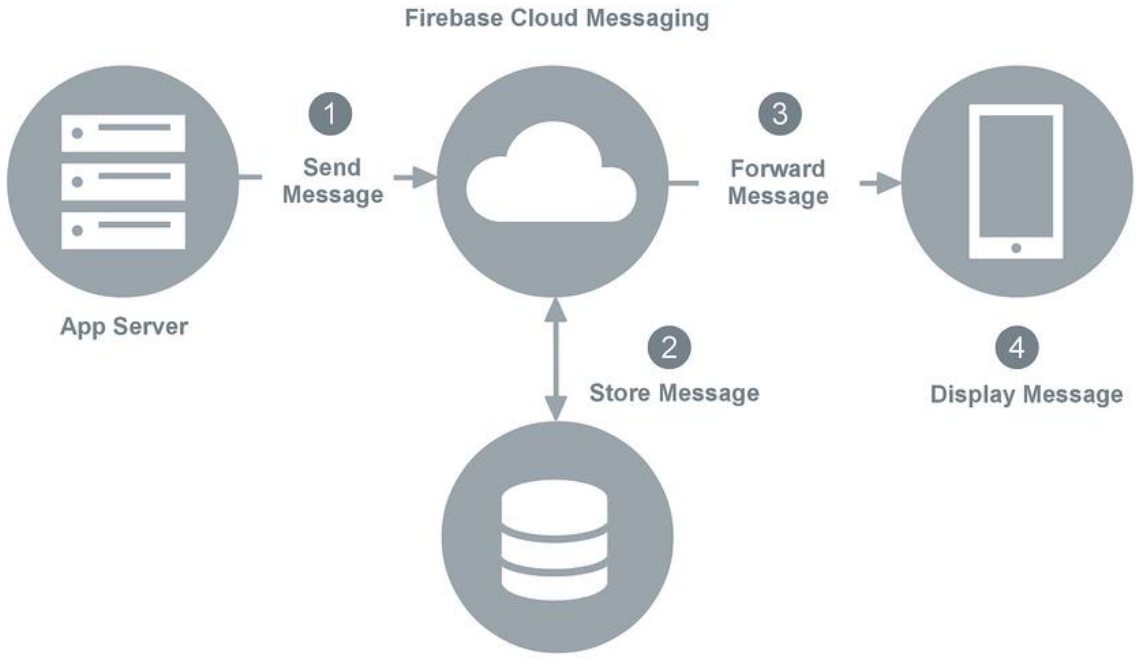
Obrázek 14: Množina objektů pro zálohování

Stav aplikace je zachován v paměti CacheStorage, proto je chod aplikace možný i bez připojení k internetu. Použitá databáze Firestore Database, podporuje uchovávání údajů v režimu off-line. Tato funkce ukládá do vyrovnávací paměti kopii dat Cloud Firestore, která aplikace aktivně používá, proto aplikace může přistupovat k údajům, když je zařízení off-line. Údaje uložené ve vyrovnávací paměti může uživatel zapisovat a číst. Když se zařízení

vrátí do režimu online, Cloud Firestore se synchronizuje a také všechny místní změny provedené aplikací s backendem Cloud Firestore. [33]

4.7.5 Push notifikace

Funkci push notifikací poskytuje aplikaci služba Firebase Cloud Messaging (FCM). Proces využívá knihovnu firebase/messaging. Po prvotní inicializaci prostřednictvím API klíčů je celý proces registrace zapouzdřen do jedné funkce – getToken(). Funkce si vyžádá oprávnění k zaregistrování service workera a k registrování služby. Aplikace kontaktuje FCM a předá mu údaje jako: ID odesílatele, klíč API a ID aplikace, aby získala registrační token. Aplikační server uloží registrační token do vyrovnávací paměti pro následnou komunikaci s aplikací. Po tomto může aplikace přijímat zprávy nebo odesílat zprávy z aplikačního serveru. Když aplikace již nechce přijímat zprávy z aplikačního serveru, může serveru odeslat požadavek na odstranění registračního tokenu. Pokud je aplikace odinstalována ze zařízení, FCM to zjistí a automaticky upozorní aplikační server, aby odstranil registrační token. Ukázka push notifikace je v příloze č.5



Obrázek 15: Diagram komunikace mezi serverem a klientem přes FCM[35]

Nahoře zobrazený diagram popisuje schéma přenosu zprávy. Postup bychom mohli rozdělit do následujících kroků:

1. Aplikační server odešle zprávu FCM.

2. Pokud zařízení není dostupné, server FCM uloží zprávu pro pozdější přenos. Zprávy se uchovávají v úložišti FCM maximálně čtyři týdny.
3. Když je zařízení dostupné, FCM přepošle zprávu do aplikace na daném zařízení.
4. Aplikace přijme zprávu z FCM, zpracuje ji a zobrazí uživateli. [34]

4.8 Testování

K testování aplikace byl použit nástroj Google Lighthouse. V tabulce č.1 je uveden seznam použitých technologií a postupů v aplikaci. Nástroj spouští automatizované testování aplikace. Testuje určité parametry a aspekty dle seznamu požadavků z kategorie podmínek PWA, který vypracovali vývojáři od Googlu jeho podrobný popis je dostupný i na internetu. Aplikace byla otestována také v dalších kategoriích: výkon, přístupnost, osvědčené postupy a SEO.

Framework / UI knihovny	React, Reach Router, Redux Toolkit, MomentJS, Toastify-JS, UUID, VitePWA Plugin, Tailwind
Nástroj na sdružování modulů	Vite
Service Worker	Aplikační Kostra + ukládání dat do mezipaměti s Workboxem
Výkonnostní modely	HTTP/2 a Server Push
Vykreslování	Vykreslování na straně klienta
API	Firebase Platform API
Hosting	Firebase

Tabulka 1: Seznam použitých technologií a postupů

4.8.1 Podmínky testování

Zařízení, na kterém byly tyto testy provedeny, byl emulovaný mobilní telefon Moto G4. V tabulce č.2 je uveden seznam podmínek za jakých byly testy prováděny. Jednalo se o simulované zhoršení výkonu zařízení a jeho připojení k internetu. Ale také rozsah datového sběru, jádro prohlížeče a podmínky načítání.

Emulované zařízení	Moto G4
Výkon CPU/paměti	1477
CPU throttling	4x zpomalení
Axe verze	4.2.3
Podmínky načítaní	Počáteční načtení stránky
Typ připojení	Pomalá 4G síť
Datový sběr	Načtení stránky pouze jednou
Síťový throttling	150ms TCP RTT
Síťová propustnost	1638.4Kbps
Jádro prohlížeče	Chromium 98.0.4758.102

Tabulka 2: Podmínky testování

Dané seskupení testů zobrazuje výsledky jednoho prvotního načtení aplikace. Připojení na internet bylo za podmínek zhoršeného přenosu dat, konkrétně se jednalo o pomalé 4G připojení s rychlostí připojení 1,638.4Kbps a dobou odezvy 150ms. Očekávané předpoklady jsou kompletní splnění požadavků ze seznamu obou kategorií k dosažení optimalizovaného stavu PWA.

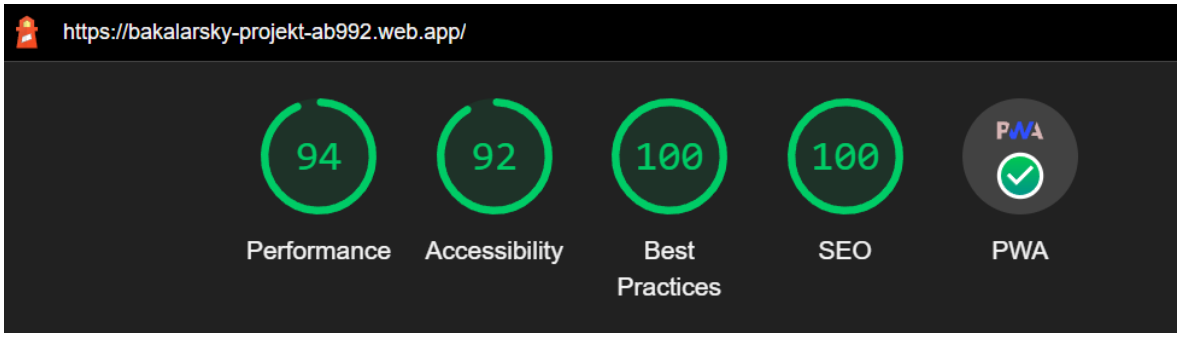
INSTALOVATELNÉ

- Manifest webové aplikace a service worker splňují požadavky na instalaci

PWA OPTIMALIZOVÁNO

- Registrace service worker, který řídí aplikaci a počáteční adresu URL.
- Nakonfigurováno pro vlastní úvodní obrazovku
- Nastaví barvu motivu pro adresní řádek.
- Obsah má správnou velikost pro zobrazovanou oblast
- Obsahuje značku <meta name="viewport"> s šířkou nebo počátečním měřítkem
- Poskytuje platnou apple-touch ikonu pro zařízení iOS
- Manifest má maskovatelnou ikonu

4.8.2 Kategorie PWA

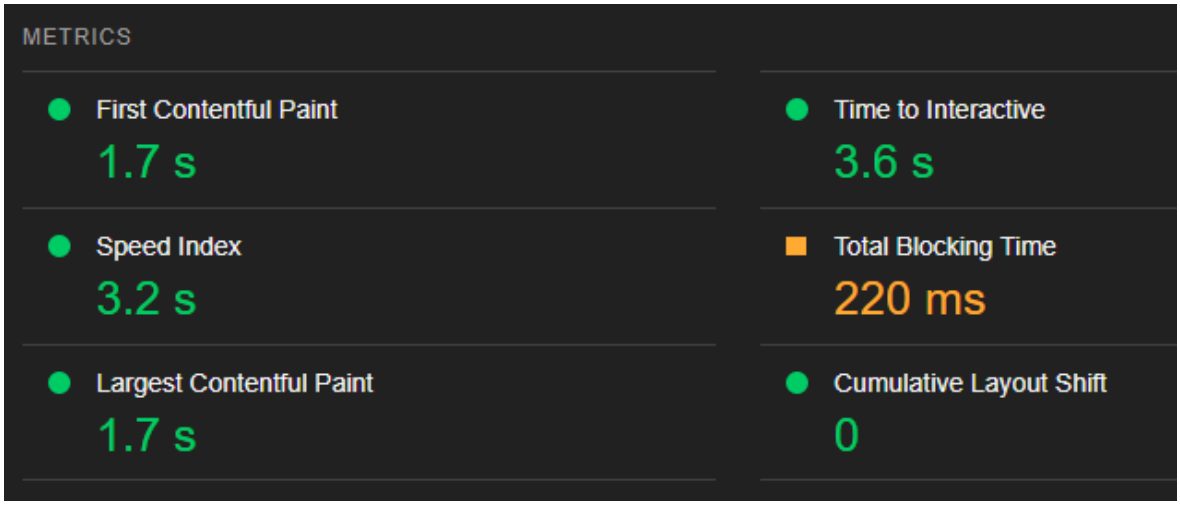


Obrázek 16: Výsledné hodnoty testování

Kategorie PWA je rozdělena do 2 částí. Na rozdíl od jiných kategorií nemá bodové ohodnocení. Hodnocení je na základě 3 odznaků podle míry splnění seznamu PWA. Pokud nedojde ke splnění první části, daná webová aplikace nemůže být nainstalována, a tedy se nepovažuje prohlížeči za PWA. Druhá kategorie se soustředí na aspekty aplikace, jejichž splněním se daná aplikace považuje za optimalizovanou. Aplikace Interaktivní studentský diář splňuje všechny uvedené požadavky. Lighthouse zároveň uvádí ještě 3 dodatečná ověření, ale musí být ověřena manuálně.

4.8.3 Výkon

Důležité metriky, kterými se vývoj PWA zabývá, ale nejsou přímo zahrnuty v kategorii PWA, je Výkon. V tomto testu daná aplikace dosáhla časů, které jsou z hlediska uživatelské zkušenosti vyhovující.

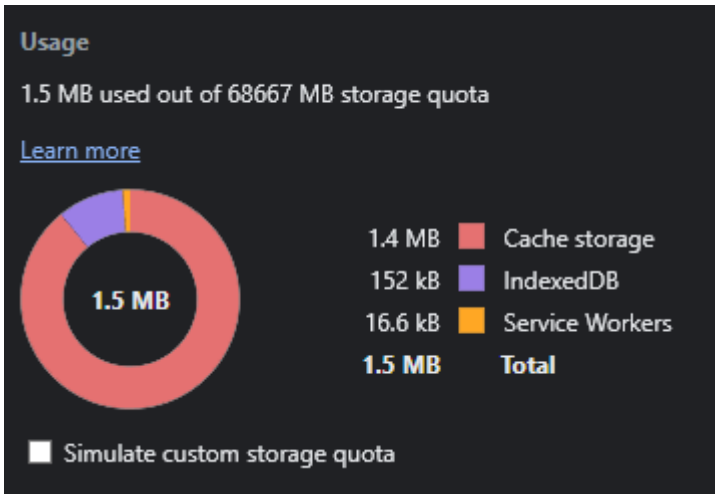


Obrázek 17: Naměřené hodnoty v kategorii výkonu

Celková doba blokování je celková doba trvání aplikace, aby byla připravena k interakci uživatele. Vzhledem k tomu, že v tomto případě uživatel již byl přihlášen, tak volání na backend služby Firebase Auth a čekání na odpověď způsobilo mírné zvýšení času.

4.8.4 Velikost

Další parametr, na který jsem se zaměřil ve svém testování, byla velikost aplikace na paměťovém médiu zařízení. Svá zjištění jsem provedl přes rozhraní Storage Manager v aplikačním rozhraní Storage API, pomocí funkce estimate(). Samotná velikost aplikace je pouze odhadem, jelikož je několik faktorů, které ovlivňují zjištěnou hodnotu. Jeden z důvodů, proč se jedná pouze o odhad, jsou použity rozdílné techniky komprese a dalším je cíleně přidávání bajtů navíc, aby se zamezilo uniknutí přesné velikosti. Velikost aplikace jsem ověřil i v Chrome DevTools, kde se výsledná hodnota shodovala.



Obrázek 18: Velikost aplikace v paměti zařízení

5 Výsledky

Z poznatků nabytých při návrhu a implementaci PWA jsem si uvědomil, že již při samotném návrhu aplikace je důležitá selekce technologií použitých v aplikaci. Přesto jsem během implementace požadavků narazil na množství problémů. Můj stanovený cíl byl, se přiblížit vzhledem i funkcionalitou k nativní aplikaci a splnit všechny požadavky ze seznamu PWA. Cíl se mi díky dostupným technologiím podařilo splnit.

Instalace do paměti zařízeni a přidáni ikony na plochu

Aby mohla být aplikace nainstalována a přidána na plochu jako nativní jsem se dopracoval vytvořením manifest a service worker souborů. Manifest obsahuje údaje potřebné k instalaci jako název aplikace, jméno autora, popis aplikace, verze aplikace,

seznam ikon a jiných souborů. Service worker je technologie, která umožnila uložení obsahu aplikace do cache paměti, vytvořila komunikační prostředí mezi rozhraním prohlížeče a internetem. Z důvodu bezpečnosti aplikace musí být hostována na doméně která poskytuje šifrování HTTP protokolu. Mnou zvolená doména a její poskytovatel hostingu podporuje šifrovaný HTTP protokol verze 2 – h2.

Off-line režim

Na off-line režim jsem musel zajistit uživatelům zůstat přihlášen i přes ztrátu připojení. Autentifikace služby Firebase poskytuje metody, které zapouzdrují celou logiku a umožňují pomocí funkce onAuthStateChanged() perzistentní přihlášení uživatele. Funkce poslouchá na změnu uživatele i když je aplikace off-line a výsledku docílí perzistentní přihlášení uživatele. Firestore databáze naopak poskytuje metody, jejichž volání umožní vytvoření dočasného uložení dat do paměti cache. Přičemž aktivně poslouchá na změnu stavu připojení k internetu. Při napojení dojde k synchronizaci nahráním změn provedených na datech v aplikaci na databázi na serveru.

Push notifikace

Využitím FCM bylo splnění zasílání push notifikací snadné. Služba přímo využívá dvě aplikační rozhraní Push API a Notifications API a poskytuje komunikační kanál pro zasílání obsahu ve formě zpráv na zařízení. Ve výsledku jsem docílil možnosti zasílat notifikace, když uživatel aplikaci používal ale i když byla vypnuta.

Seznam PWA optimalizovaných požadavek

Vytvořením maskované ikony, doplněním dodatečných atributů do manifest souboru a doplněním meta značek do záhlaví webu jsem docílil splnění části požadavků potřebných pro optimalizaci PWA.

5.1 Výhody vývoje PWA

Jako nespornou výhodu vývoje takového druhu aplikace vidím jednotný kód pro všechny druhy zařízení a všechny druhy OS. Je to ušetření času, ujištění konzistence funkcionality napříč rozdílnými druhy OS. Snadný přístup pro uživatele přímo přes internetový prohlížeč. Možnost umístění takové aplikace také na aplikační obchody App Store nebo Google Play Store.

5.2 Nevýhody vývoje PWA

Aplikace i přes všechny funkce stále je v jejím základu aplikace, která běží uprostřed internetového prohlížeče. Tím je do určité míry limitována možnostmi prostředí. Nativní vývoj umožňuje tvořit detailnější řešení a umožňuje využívat určité hardwarové části, které jsou při webovém vývoji zatím nedostupné.

6 Závěr

V mé práci jsem prozkoumal možnosti vývoje Progresivních webových aplikací. Prvotním definováním požadavků jsem vytvořil obraz o tom, jaké funkce bude aplikace nabízet. Následně poté jsem vytvořil wireframe, který reprezentoval model uživatelského rozhraní a komponent. Poté jsem definoval architekturu aplikace s cílem vytvořit funkční celek, který je schopen splnit všechny uvedené požadavky. Následovala implementace požadavků, kde jsem si uvědomil, že samotná progresivní webová aplikace je jen určitá skupina technologií, která ve výsledku vytváří to progresivní zlepšení webové aplikace. Cílem bylo zaměření se na funkcionality, jaké doposud uměly nabídnout pouze nativní aplikace. Kombinací technologií jako je manifest, service worker a protokol h2, jsem vytvořil instalovatelnou aplikaci s vlastní ikonou na ploše, jako je tomu u nativní varianty. Prostřednictvím Firebase backend platformy jsem uměl udělat chod aplikace i při absenci připojení na internet, umožnit zůstat přihlášen a manipulovat s daty. Ve výsledku docílit uživatelské zkušenosti jako při nativní aplikaci. Do aplikace jsem úspěšně implementoval zasílání push notifikací ze serveru přímo na zařízení přes službu Firebase Cloud Messaging. V závěru jsem testoval aplikaci přes Google Lighthouse. Ten ji vyhodnotil v pěti kategoriích: výkon, přístupnost, doporučené praktiky, SEO a PWA. Jednalo se o automatizovaný test za programem stanovených podmínek. Bylo emulováno zařízení, které mělo zhoršení výkonu a připojení k internetu s cílem prověřit schopnost aplikace i v nepříznivých podmínkách. Ve všech kategoriích aplikace dosáhla bodového hodnocení nad 90 z možných 100 bodů. V poslední kategorii splnila všechny podmínky PWA a dosáhla odznaku optimalizované PWA aplikace.

Vývoj Progresivních webových aplikací nabízí produkt, který je rovnocenný aplikacím vyvíjeným nativně. Takový vývoj se může aplikovat nejen na nové aplikace, ale také na klasické webové aplikace, které se umí přeměnit na progresivní. Výhodou je poměrně jednoduchá implementace. Jako další výhody vidím jednotný kód pro celou aplikaci, z čehož vyplývá ušetření času i prostředků při vývoji. Nevýhodou je, že existuje skupina internetových prohlížečů, které nemají podporu PWA. V takovém případě aplikace v prohlížeči funguje stejně, ale nemůže být nainstalována do paměti zařízení. Nevýhodou je, že daná aplikace se spouští v prohlížeči, u kterých dochází k nekonzistentní podpore určitých funkcionalit napříč různýma internetovými prohlížeči. V aplikaci by se daly doplnit dodatečné funkce, které by prozkoumaly schopnosti a možnosti webového rozhraní. Jak

využít mikrofon nebo kameru mobilního zařízení k ukládání hlasových připomínek nebo pořizování fotografií studijních materiálů. Porovnání vůči nativní aplikaci a změření výkonu a rychlosti. Prozkoumat bezpečnost a možná rizika vyplývající z využití service workera a technologií spojených s vývojem PWA. Udělat ucelený přehled testování na rozdílných zařízeních a systémech s cílem poskytnout detailní přehled podpory jednotlivých technologií.

7 Seznam použitých zdrojů

7.1 Seznam literatury

1. ALEX BANKS, EVE PORCELLO. Learning React, Modern Patterns for Developing React Apps. Sebastopol: O'Reilly Media, Inc, 2020. ISBN 978-1-492-05172-5
2. ANTHONY ACCOMAZZO, NATE MURRAY, ARI LERNER, CLAY ALLSOPP, DAVID GUTMAN, AND TYLER MCGINNIS. Fullstack React, The Complete Guide to ReactJS and Friends. \newline, 2020.
3. DEAN ALAN HUME. Progressive Web Apps. Shelter Island: Manning Publications Co, 2018. ISBN: 9781617294587
4. DENNIS SHEPPARD. Beginning Progressive Web App Development. Tinley Park: Apress, 2017. ISBN-13: 978-1-4842-3090-9
5. CHRIS LOVE. Progressive Web Application Development by Example. Birmingham: Packt Publishing, 2018. ISBN 978-1-78712-542-1.
6. SCOTT DOMES. Progressive Web Apps with React. Birmingham: Packt Publishing, 2017. ISBN 978-1-78829-755-4

7.2 Internetové zdroje

7. ANON., 2022. Biggest app stores in the world 2021 | Statista. Statista [online] [cit. 28 . 02 2022]. Dostupné z: https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/
8. KOTULIAK, IVAN. Progresívne webové aplikácie. In: . Presentation. B.m. 2019.
9. About MDN Web Docs - The MDN project | MDN. Developer.mozilla.org [online]. 2022 [cit. 28-02-2022]. Dostupné z: https://developer.mozilla.org/en-US/docs/MDN/About
10. Progressive web apps (PWAs) | MDN. Developer.mozilla.org [online]. 2022 [cit. 28-02-2022]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps
11. Secure contexts - Web security | MDN. Developer.mozilla.org [online]. 2022 [cit. 28-02-2022]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/Security/Secure_Contexts

12. Service Worker API - Web APIs | MDN. Developer.mozilla.org [online]. 2022 [cit. 28-02-2022]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API
13. Service Workers: an Introduction | Web Fundamentals | Google Developers. Google Developers [online]. 2022 [cit. 28-02-2022]. Dostupné z: https://developers.google.com/web/fundamentals/primers/service-workers
14. Progressive Web Apps: Escaping Tabs Without Losing Our Soul - Infrequently Noted. Infrequently Noted [online]. 2022 [cit. 28-02-2022]. Retrieved z: https://infrequently.org/2015/06/progressive-apps-escaping-tabs-without-losing-our-soul/
15. Web Fundamentals | Google Developers. Google Developers [online]. 2022 [cit. 28-02-2022]. Dostupné z: https://developers.google.com/web/fundamentals/architecture/app-shell
16. Measure performance with the RAIL model. web.dev [online]. 2022 [cit. 28-02-2022]. Dostupné z: https://web.dev/rail/
17. Thinkwithgoogle.com [online]. 2022 [cit. 28-02-2022]. Dostupné z: https://www.thinkwithgoogle.com/intl/en-154/marketing-strategies/app-and-mobile/need-mobile-speed-how-mobile-latency-impacts-publisher-revenue/
18. Eliminate render-blocking resources. web.dev [online]. 2022 [cit. 28-02-2022]. Dostupné z: https://web.dev/render-blocking-resources/
19. Browser-level image lazy-loading for the web. web.dev [online]. 2022 [cit. 28-02-2022]. Dostupné z: https://web.dev/browser-level-image-lazy-loading/https://alistapart.com/article/responsive-web-design/
20. Responsive design - Learn web development | MDN. Developer.mozilla.org [online]. 2022 [cit. 28-02-2022]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Responsive_Design#media_queries
21. React – A JavaScript library for building user interfaces. Reactjs.org [online]. 2022 [cit. 28-02-2022]. Dostupné z: https://reactjs.org/
22. Virtual DOM and Internals – React. Reactjs.org [online]. 2022 [cit. 28-02-2022]. Dostupné z: https://reactjs.org/docs/faq-internals.html
23. CHINNATHAMBI, KIRUPA. Creating a Single-Page App in React using React Router. kirupa.com [online]. 2022 [cit. 28-02-2022]. Dostupné z:

https://www.kirupa.com/react/creating_single_page_app_react_using_react_router.htm
24. React Components. W3schools.com [online]. 2022 [cit. 28-02-2022. Dostupné z: https://www.w3schools.com/react/react_components.asp
25. Components and Props – React. Reactjs.org [online]. 2022 [cit. 28-02-2022]. Dostupné z: https://reactjs.org/docs/components-and-props.html#props-are-read-only
26. Introducing JSX – React. Reactjs.org [online]. 2022 [cit. 28-02-2022]. Dostupné z: https://reactjs.org/docs/introducing-jsx.html
27. Unidirectional Data Flow - GeeksforGeeks. GeeksforGeeks [online]. 2022 [cit. 28-02-2022]. Dostupné z: https://www.geeksforgeeks.org/unidirectional-data-flow/
28. Understanding "state" in React Components - Thinkster. Thinkster.io [online]. 2022 [cit. 28-02-2022]. Dostupné z: https://thinkster.io/tutorials/understanding-react-state
29. Lighthouse PWA Analysis Tool | Web | Google Developers. Google Developers [online]. 2022 [cit. 28-02-2022]. Dostupné z: https://developers.google.com/web/ilt/pwa/lighthouse-pwa-analysis-tool
30. Using Google Lighthouse to audit your web application. Flexiple.com [online]. 2022 [cit. 28-02-2022]. Dostupné z: https://flexiple.com/developers/using-google-lighthouse-to-audit-your-web-application/#section4
31. Scandiweb.com [online]. 2022 [cit. 28-02-2022]. Dostupné z: https://scandiweb.com/blog/improve-website-accessibility-with-progressive-web-apps/
32. Google Lighthouse – SEO Glossary | Searchmetrics. Searchmetrics [online]. 2022 [cit. 28-02-2022]. Dostupné z: https://www.searchmetrics.com/glossary/google-lighthouse/
33. Response Time Limits: Article by Jakob Nielsen. Nielsen Norman Group [online]. 2022 [cit. 28-02-2022]. Dostupné z: https://www.nngroup.com/articles/response-times-3-important-limits/
34. Firebase Authentication | Firebase Documentation. Firebase [online]. 2022 [cit. 28-02-2022]. Dostupné z: https://firebase.google.com/docs/auth

35. Redux is Dead: Long Live Redux Toolkit. OpenReplay Blog [online]. 2022 [cit. 28-02-2022]. Dostupné z: https://blog.openreplay.com/redux-is-dead-long-live-redux-toolkit
36. An introduction to HTTP/2 - SSL.com. SSL.com [online]. 2022 [cit. 28-02-2022]. Dostupné z: https://www.ssl.com/article/an-introduction-to-http2/
37. Access data offline | Firebase Documentation. Firebase [online]. 2022 [cit. 28-02-2022]. Dostupné z: https://firebase.google.com/docs/firestore/manage-data/enable-offline
38. Firebase Cloud Messaging - Xamarin. Docs.microsoft.com [online]. 2022 [cit. 28-02-2022]. Dostupné z: https://docs.microsoft.com/en-us/xamarin/android/data-cloud/google-messaging/firebase-cloud-messaging#setup_fcm

8 Přílohy

Příloha č.1 – Obrazovky autentifikace aplikace

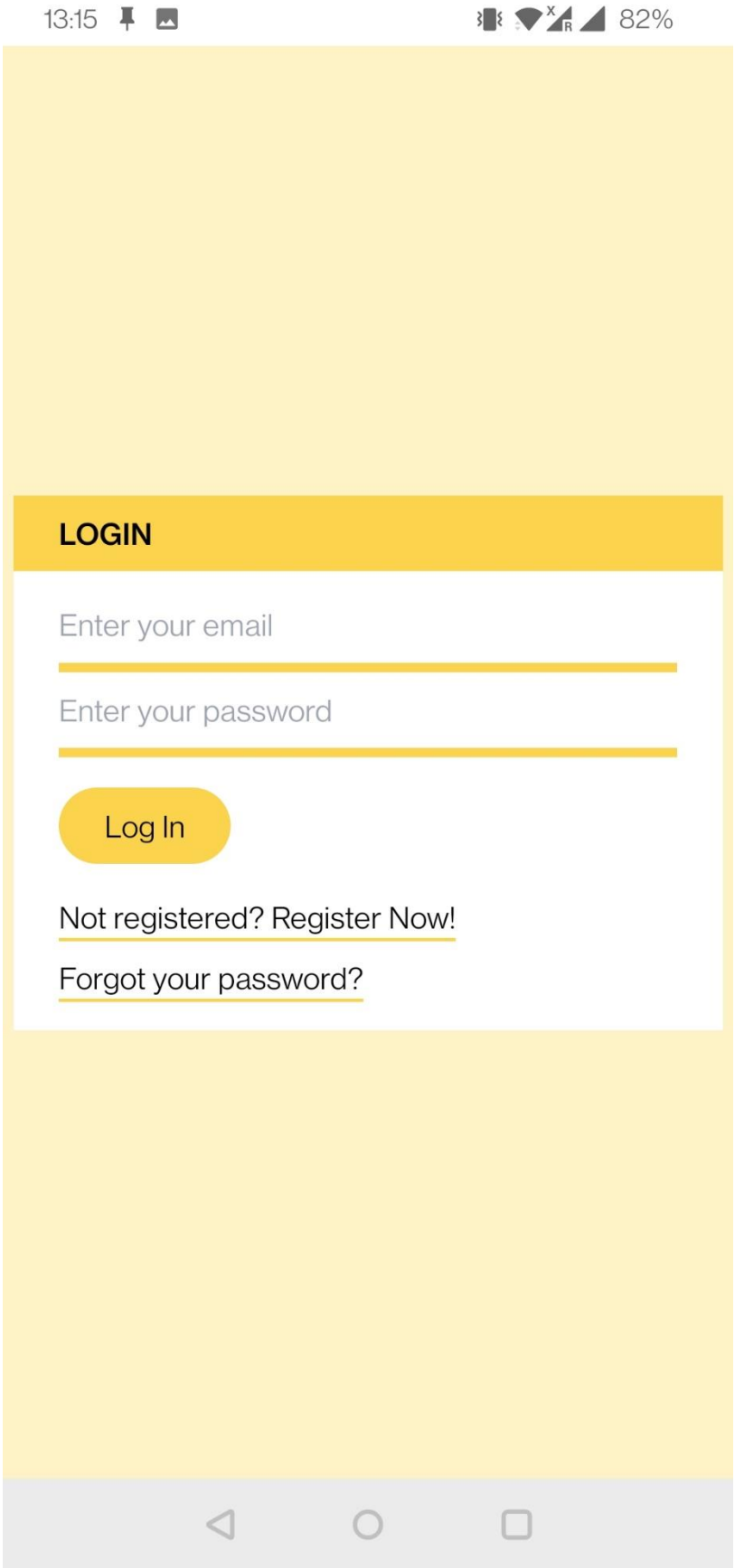
Příloha č.2 – Obrazovka přidáni předmětu

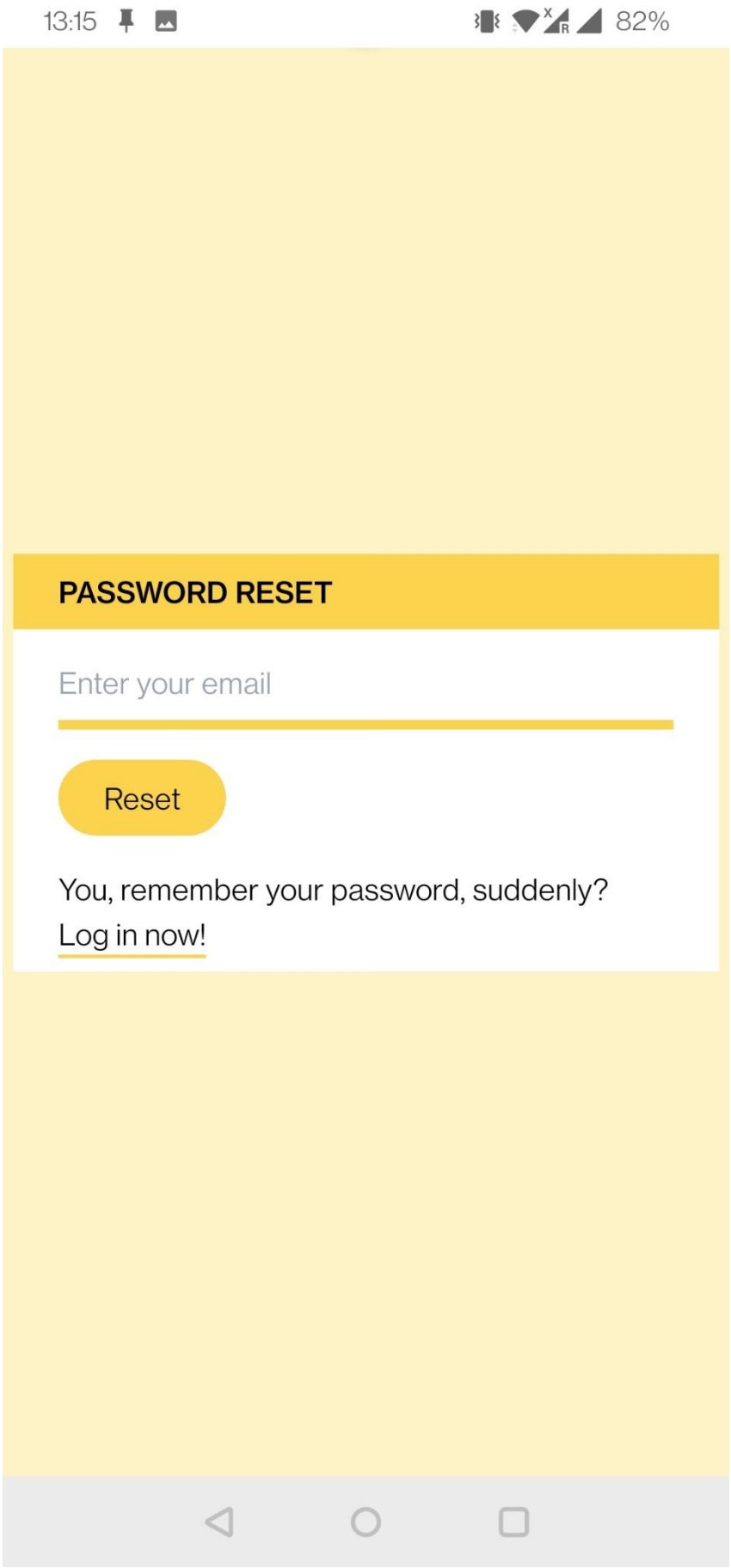
Příloha č.3 – Obrazovka přidáni předmětu a úloh

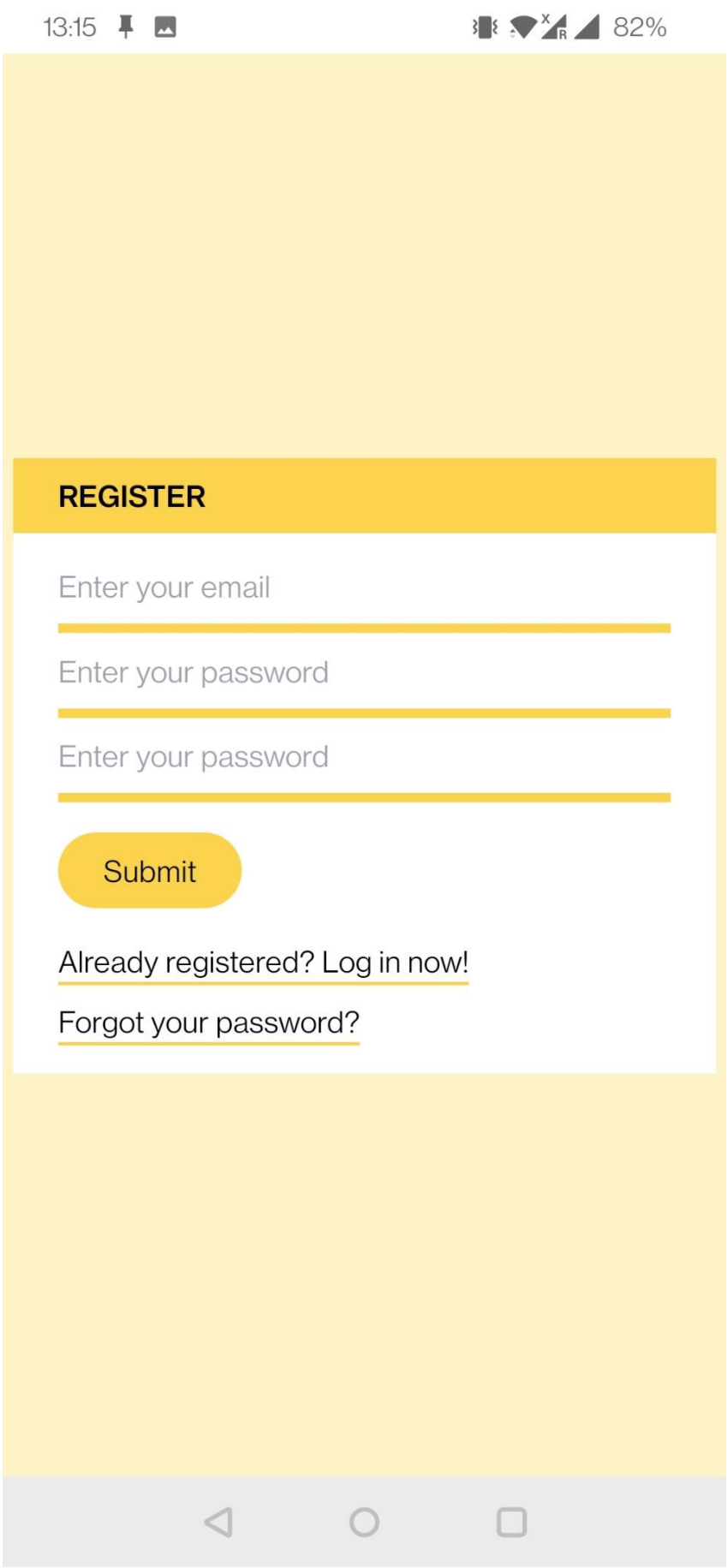
Příloha č.4 – Obrazovka Instalování aplikace

Příloha č.5 – Obrazovka test push notifikací

Příloha č.1 – Obrazovky autentifikace aplikace



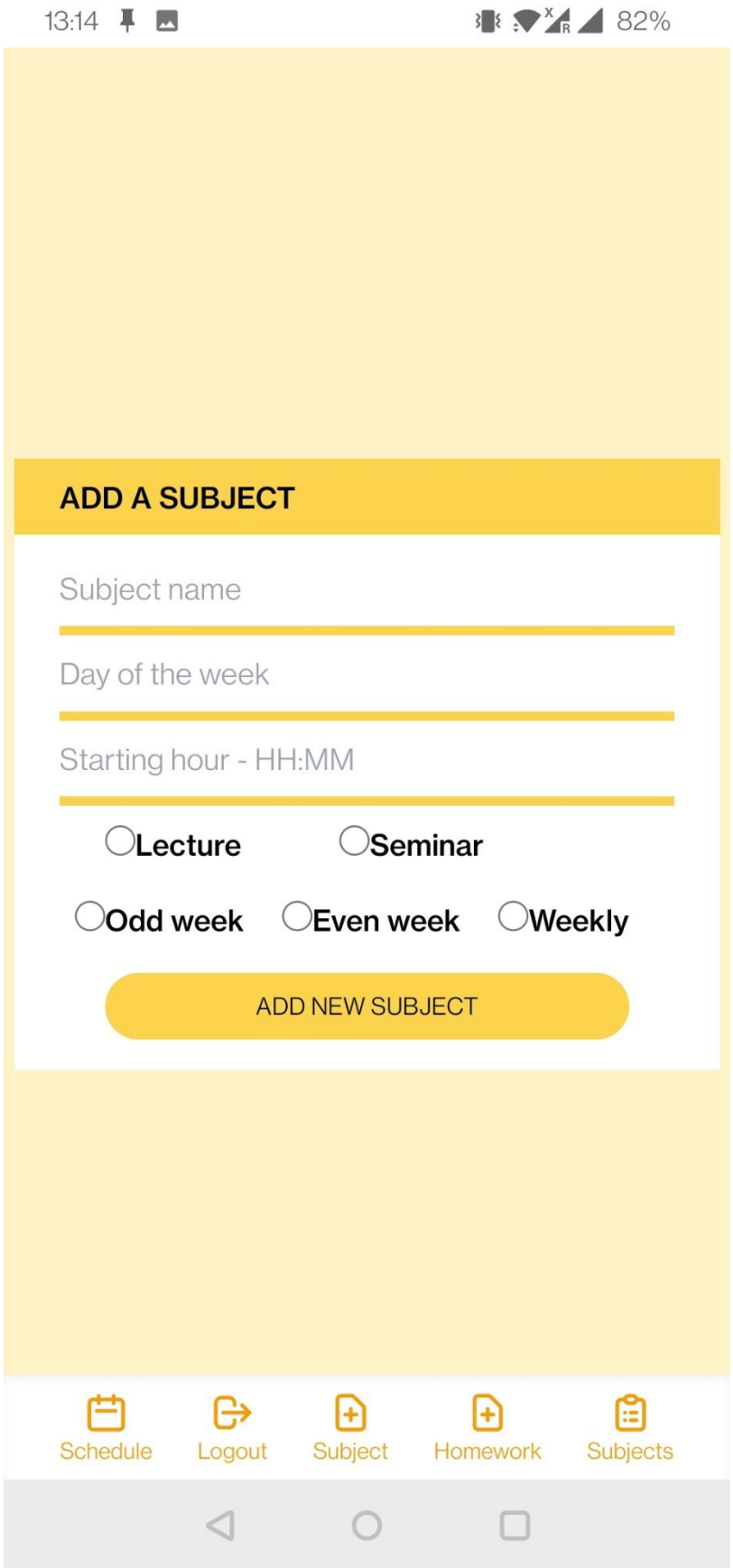

13:15
82%
PASSWORD RESET
Enter your email
Reset
You, remember your password, suddenly?
Log in now!

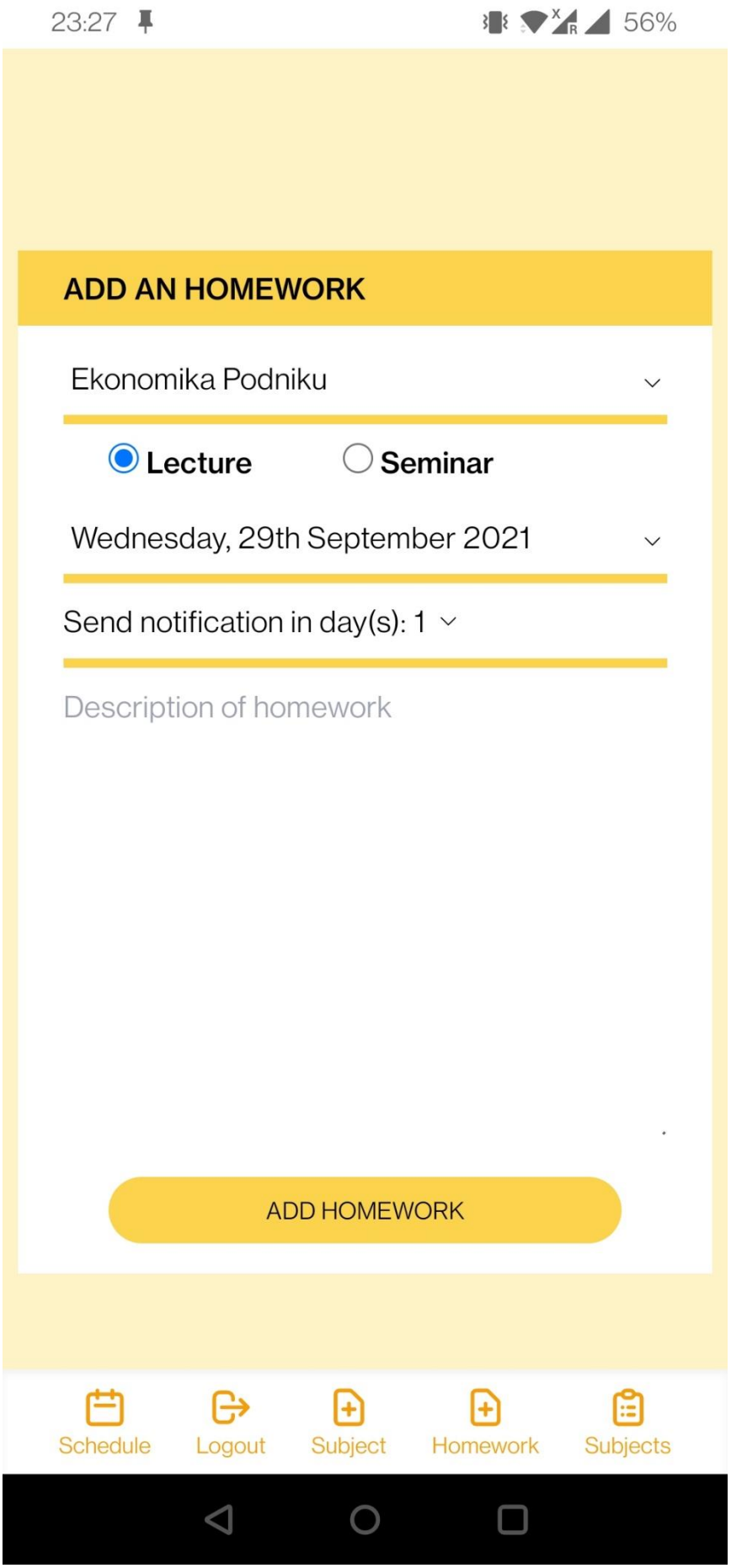

13:15
82%
REGISTER
Enter your email
Enter your password
Enter your password
Submit
Already registered? Log in now!
Forgot your password?

Příloha č.2 – Obrazovka seznam semestru

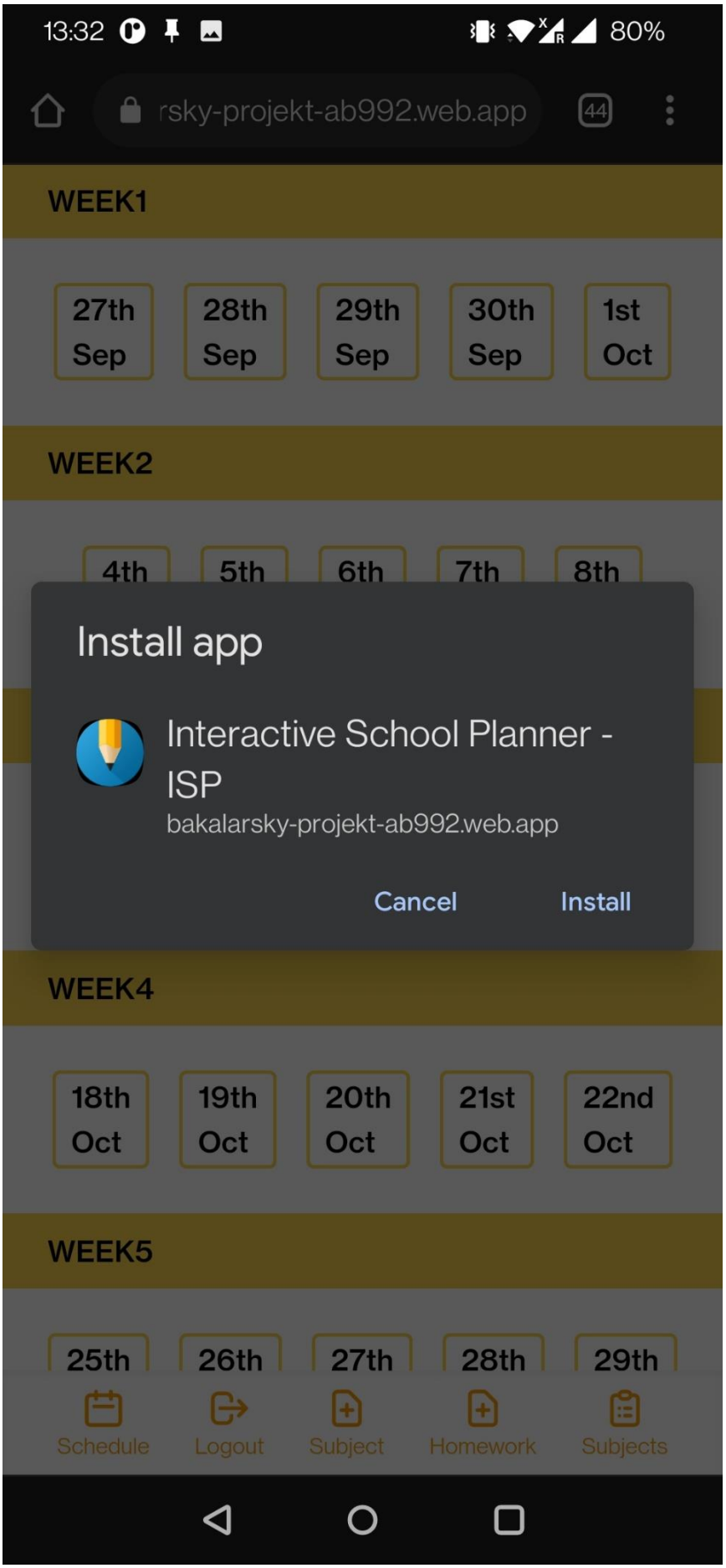

13:16
82%
WEEK1
27th Sep
28th Sep
29th Sep
30th Sep
1st Oct
Tuesday, 28th September 2021
08:45
09:30
Ekonomika Podniku
09:30
10:15
Ekonomika Podniku
10:15
10:30
10:30
11:15
Základy účetnictví
11:15
12:00
Základy účetnictví
12:00
12:15
12:15
13:00
13:00
13:45
13:45
Schedule
Logout
Subject
Homework
Subjects

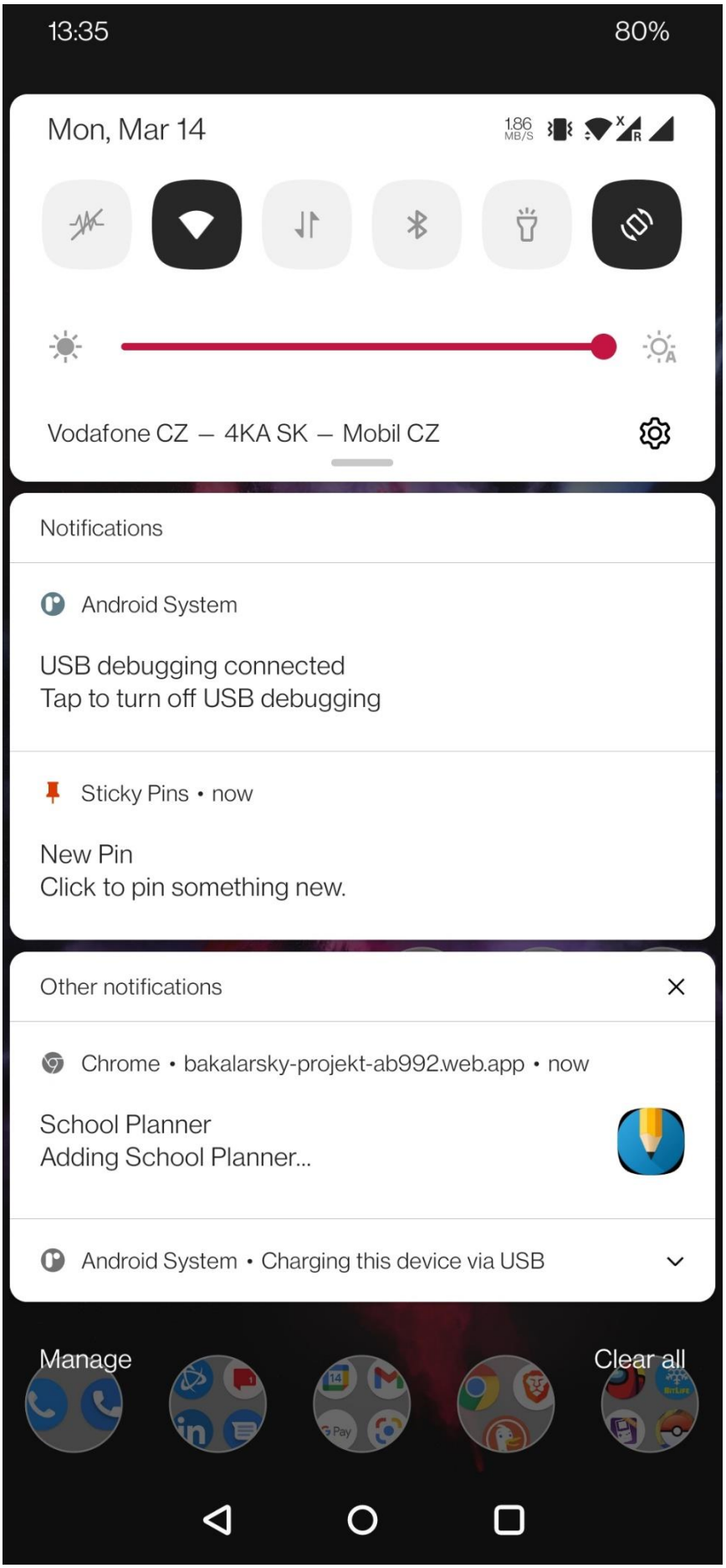
Příloha č.3 – Obrazovka přidáni předmětu a úloh

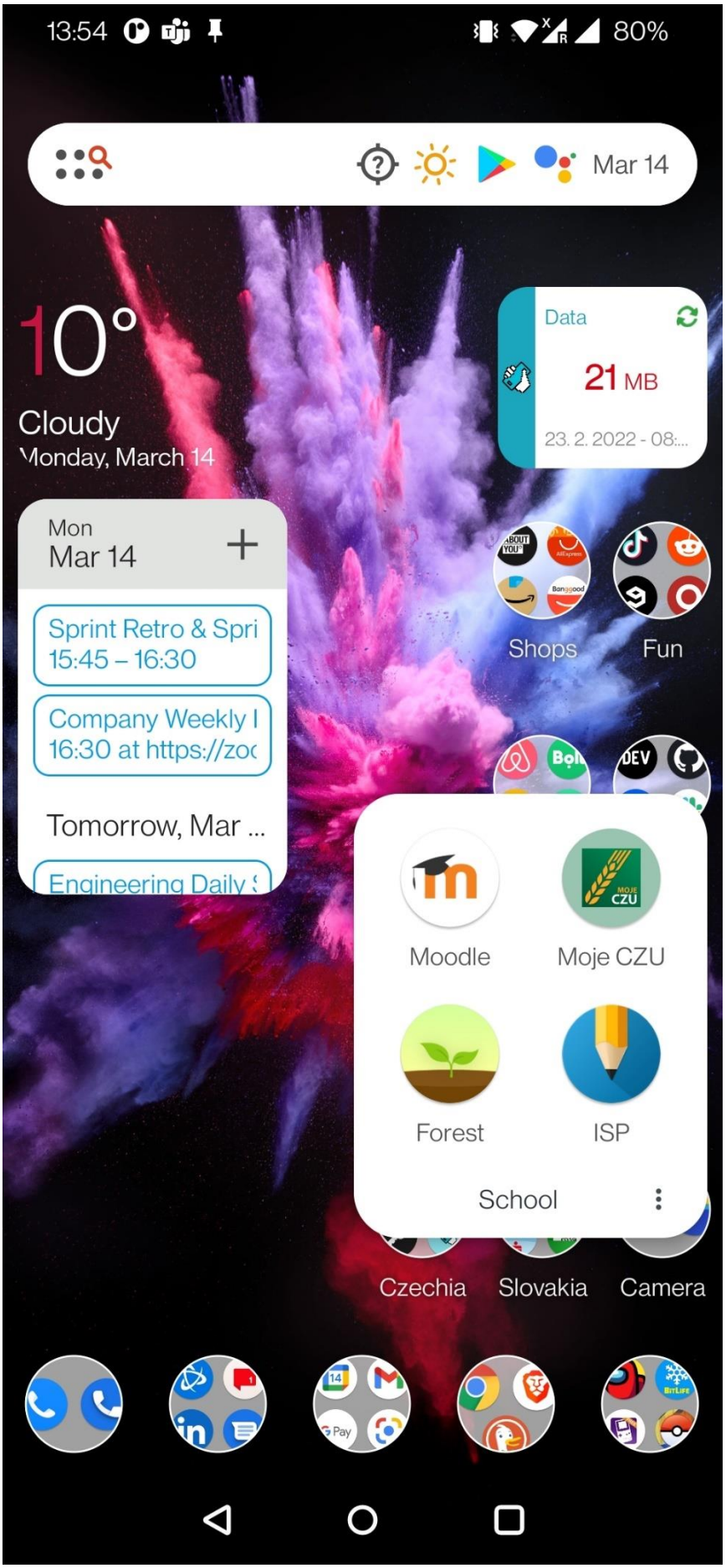



23:27
56%
ADD AN HOMEWORK
Ekonomika Podniku
Lecture
Seminar
Wednesday, 29th September 2021
Send notification in day(s): 1
Description of homework
ADD HOMEWORK
Schedule
Logout
Subject
Homework
Subjects

Příloha č.4 – Obrazovka Instalování aplikace


13:32
80%
rsky-projekt-ab992.web.app
44
WEEK1
27th Sep
28th Sep
29th Sep
30th Sep
1st Oct
WEEK2
4th
5th
6th
7th
8th
Install app
Interactive School Planner - ISP
bakalarsky-projekt-ab992.web.app
Cancel
Install
WEEK4
18th Oct
19th Oct
20th Oct
21st Oct
22nd Oct
WEEK5
25th
26th
27th
28th
29th
Schedule
Logout
Subject
Homework
Subjects


13:35
80%
Mon, Mar 14
1.86 MB/S
Vodafone CZ — 4KA SK — Mobil CZ
Notifications
Android System
USB debugging connected
Tap to turn off USB debugging
Sticky Pins • now
New Pin
Click to pin something new.
Other notifications
Chrome • bakalarsky-projekt-ab992.web.app • now
School Planner
Adding School Planner...
Android System • Charging this device via USB
Manage
Clear all


13:54
80%
Mar 14
10°
Cloudy
Monday, March 14
Data
21 MB
23. 2. 2022 - 08:...
Mon
Mar 14
Sprint Retro & Spri
15:45 – 16:30
Company Weekly I
16:30 at https://zoc
Tomorrow, Mar ...
Engineering Daily
Shops
Fun
Moodle
Moje CZU
Forest
ISP
School
Czechia
Slovakia
Camera

Príloha č.5 – Obrazovka test push notifikací