EKONOMICKÁ UNIVERZITA V BRATISLAVE

FAKULTA HOSPODÁRSKEJ INFORMATIKY

Evidenčné číslo: 103004/B/2022/36124048425768964

OFFLINE WEBOVÉ STRÁNKY

Bakalárska práca

2022 **Adam Kmec**

EKONOMICKÁ UNIVERZITA V BRATISLAVE

FAKULTA HOSPODÁRSKEJ INFORMATIKY

OFFLINE WEBOVÉ STRÁNKY

Bakalárska práca

Študijný program:	Hospodárska informatika
Študijný odbor:	Hospodárska informatika
Školiace pracovisko:	Katedra aplikovanej informatiky
Školiteľ:	Ing. Igor Bandurič, PhD.

Bratislava 2022 **Adam Kmec**

Ekonomická univerzita v Bratislave
Fakulta hospodárskej informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Adam Kmec
Študijný program: hospodárska informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: Bakalárska záverečná práca
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Offline webové stránky

Anotácia: Cieľom práce je navrhnúť a implementovať jednoduchý webový formulár (stránku), ktorá dokáže fungovať aj v offline režime. Dáta, ktoré sú do stránky vložené počas nedostupnosti siete/internetu sa prenesú na server po pripojení. Práca nájde vhodnú technológiu aj koncept.

Vedúci: Ing. Igor Bandurič, PhD.
Katedra: KAI FHI - Katedra aplikovanej informatiky FHI
Vedúci katedry: Ing. Mgr. Peter Schmidt, PhD.

Dátum zadania: 06.04.2021

Dátum schválenia: 06.04.2021

Ing. Mgr. Peter Schmidt, PhD.
vedúci katedry

Čestné vyhlásenie

Čestne vyhlasujem, že záverečnú prácu som vypracoval samostatne a že som uviedol všetku použitú literatúru.

Dátum:

Poďakovanie

Za odborné rady, pripomienky a pomoc pri písaní tejto práce ďakujem môjmu školiteľovi Ing. Igor Bandurič, PhD. Svojej rodine, ktorá stála vždy pri mne a kamarátom, ktorí boli mojou oporou aj v tých najstresovejších chvíľach počas písania tejto záverečnej práce.

ABSTRAKT

KMEC, Adam: *Offline webové stránky*. – Ekonomická univerzita v Bratislave. Fakulta hospodárskej informatiky; Katedra aplikovanej informatiky. – Vedúci záverečnej práce: Ing. Igor Bandurič, PhD. Bratislava: FHI EU, 2022, 42 strán.

Cieľom záverečnej práce, na tému offline webové stránky, je navrhnúť a implementovať jednoduchý webový formulár (stránku), ktorá dokáže fungovať aj v offline režime. Dáta, ktoré sú do stránky vložené počas nedostupnosti siete/internetu sa prenesú na server po pripojení. Výstupom bakalárskej práce by malo byť možné riešenie tejto problematiky a jej nasledovné uskutočnenie. Práca je rozdelená do troch kapitol. Prvá kapitola je venovaná vysvetleniu kľúčových pojmov, ktoré túto prácu vystihujú. Vymedzujeme v nej offline podporu webovej stránky a rôzne technológie, s ktorými môžeme dosiahnuť túto podporu. V druhej časti sa charakterizuje metodika práce a ciele našej práce. Záverečná kapitola sa zaoberá vytvorením jednoduchého webového formulára a stručným popisom toho, ako je možné na spomínanom webovom formulári urobiť túto offline podporu. Výsledkom riešenia danej problematiky je celkový pohľad na to, aký prínos má táto problematika pre nás a aj obyčajných používateľov webu.

Kľúčové slová: offline, web storage, localStorage, javascript

ABSTRACT

KMEC, Adam: *Offline websites*. - University of Economics in Bratislava. Faculty of Business Informatics; Department of Applied Informatics. - Thesis supervisor: Ing. Igor Bandurič, PhD. Bratislava: FHI EU, 2022, 42 pages.

The aim of the final work, on the topic of offline websites, is to design and implement a simple web form (page) that can work offline. Data that is entered into the site during network / internet unavailability will be transferred to the server after connection. The output of the bachelor thesis should be a possible solution to this issue and its subsequent implementation. The work is divided into three chapters. The first chapter is devoted to explaining the key concepts that describe this work. It defines offline website support and the various technologies with which we can provide this support. The second part characterizes the methodology of work and the goals of our work. The final chapter is about creating a simple web form and a brief description how it is possible to make this offline support on the mentioned web form. if the solution to the problem is an overall view of the benefits of this issue for us and ordinary web users.

Key words: offline, web storage, localStorage, javascript

OBSAH

Zoznam skratiek

ADO	-Active Data Objects
ADO	-ActiveX Data Objects
API	-Application programming interface
ASP	-Active Server Pages
CORS	-Cross-origin resource sharing
CSS	-Cascading Style Sheets
DOM	-Document Object Model
GB	-gigabajt
GHz	-gigaherz
HTML	-HyperText Markup Language
HTTP	-Hypertextový prenosový protokol
IDE	-Integrated Development Enviroment
JSON	-JavaScript Object Notation
KB	-kilabajt
MB	-megabajt
NPM	-Node Package Manager
PHP	-Hypertext Preprocessor
RAM	-Random Access Memory
RDBMS	-Relational Database Management System
SQL	-Structured Query Language
SSL	-Secure Socket Layer
URI	-Uniform resource identifier
URL	-Uniform resource Locator
WWW	-World Wide Web

Zoznam ilustrácií

Úvod

V dnešnej dobe sú moderné technológie čoraz viac využívané a stretávame sa s nimi na každom kroku. Súčasný svet nebude vedieť čo pojem offline webová stránka vôbec je alebo čo to znamená. Pritom každý používateľ webu sa s touto stránkou stretáva denne na bežnom poriadku. Táto technológia je neodmysliteľnou súčasťou webovej stránky a bez nej by sme si už ani nevedeli predstaviť ako by web vyzeral bez týchto technológií a metód. Mnoho firiem, ktoré vynaložia časť zo svojho kapitálu na reklamy ktoré môžeme vidieť na webových stránkach práve sú závislé na tieto technológie.

Cieľom našej práce bude bližšie čitateľom objasniť túto problematiku a jej význam a celkový dopad na web ako taký. V prvej kapitole sa zaoberáme využitím danej problematiky doma, ale aj v zahraničí. Popisujeme v nej teoretické východiská, ktoré sú pre túto prácu dôležité a to, offline podporu a metódy, ktoré môžeme využívať a využívame pri riešení tejto problematiky. Uvádzame tiež najdôležitejšie a najpoužívanejšie technológie, s ktorými túto problematiku dokážeme vyriešiť. Druhá kapitola vymedzuje cieľ našej záverečnej práce a popisuje metodiku, ktorú sme použili v rámci nášho projektu. Obsahom tretej kapitoly sú výsledky práce a záverečná diskusia. Vychádzajúc z teoretických východísk v tejto kapitole predstavujeme webovú stránku, ktorú sme vytvorili. Ukážka webovej stránky je názorne predvedená pri pripojení na internet a tiež offline.

Na základe nadobudnutých teoretických ale aj praktických poznatkov sú podľa nášho názoru offline webové stránky niečo, čo je verejnosťou nie veľmi známa problematika a pritom je veľmi dôležitá pri tvorbe veľkej a hlavne dobre funkčnej webovej stránky.

1. Súčasný stav riešenej problematiky doma a v zahraničí

Existuje mnoho prototypov a rôznych technológií pre možnosti ukladania webových aplikácií, ktoré sú offline. Jednými z prvých pokusov bola napríklad technológia cookies alebo riešenie, na ktoré sme potrebovali ďalší softvér ako Flash alebo Google Gears. Cookies fungovali na princípe, že čítali údaje na strane servera, pričom modernejšie technológie ich čítali na strane používateľa webovej aplikácie. Tieto technológie boli na trhu už od roku 1994 a boli zavedené ako beta verzia v známom prehliadači Netscape.

V súčasnosti cookies nahradili oveľa modernejšie spôsoby ako zapisovať dáta do offline aplikácií ako napríklad SQLite, ktorý sa natívne obsiahol do prehliadača, lenže kvôli problémom so štandardizáciou sa od neho čoraz viac upúšťa. Ako najjednoduchším sa v dnešnej dobe využíva Web Storage, ktorý umožňuje aplikáciám ukladať dáta natrvalo cez okná alebo len pre jednu reláciu na jednej karte. Sofistikovanejšie riešenie je IndexedDB, ktoré je nízkoúrovňové API nad databázovými štruktúrami s transakciami a kurzormi. Najnovší prírastok je Cache API ktoré je špecializované riešenie na uchovávanie párov žiadosť/odpoveď.

Offline webové stránky využívajú tieto technológie, aby dosiahli to, že webová stránka aj po vypadnutí internetového pripojenia, užívateľovi bude môcť zabezpečiť zapísanie dát na server stránky. Zabezpečuje sa to tým, že tieto dáta sú uložené dočasne alebo navždy (podľa typu technológie) v počítači užívateľa a po opätovnom pripojení na internet sa tieto dáta odošlú na server stránky. Toto riešenie sa zahŕňa bežne v každej progresívnejšej webovej aplikácii, aby sa zabezpečila lepšia používateľská skúsenosť.

Zotrvanie údajov v akomkoľvek úložisku je spravované prehliadačom a v predvolenom nadstavení môže byť vymazané bez súhlasu koncového používateľa. Na to slúži takzvané Storage API, ktoré dáva aplikáciám metódu na ukladanie údajov, ak to používateľ povolí.

1.1 Offline podpora

Offline podpora sa stáva požiadavkou mnohých webových aplikácií, keďže v dnešnej dobe sa využíva napríklad pri podpore reklám na mobilných zariadeniach, tabletoch, či osobných počítačoch. Bežná mylná predstava je, že bez internetového pripojenia sú webové aplikácie zbytočné. Natívne aplikácie môžeme napísať tak, aby mali na dátovom úložisku nášho zariadenia nainštalované všetko, čo potrebujú od kódu aplikácie až po samotné úložisko. Toto dosiahneme pomocou HTML5, aby aplikácie boli funkčné aj vtedy, keď je používateľské zariadenie odpojené. Offline verzia webovej aplikácie nemusí ponúkať plnú funkčnosť, no určité funkcie môžu byť stále dostupné.

Aplikácia, ktorá je založená na HTML a chce byť užitočná v offline režime, musí mať prístup k lokálnemu úložisku v zariadení, aby sa údaje zadané používateľom v oknách HTML mohli uložiť lokálne a zároveň synchronizované so serverom aplikácie ak bude k dispozícii pripojenie. Musíme zabezpečiť, aby server odosielal klientovi iba nespracované údaje bez značiek HTML. Preto by sme na vývoj tohto rozhrania mali použiť jazyky HTML, javascript, CSS a backend v našom preferovanom jazyku ako napríklad java, NET, PHP a údaje vo formáte JSON by sme mali odosielať zo servera klientovi a späť. Je vhodné vytvoriť dátovú vrstvu v kóde javascript, ktorá bude zodpovedná za všetku dátovú komunikáciu. Ak je dostupné internetové pripojenie, dátová vrstva odošle požiadavky na server, v opačnom prípade získa údaje z lokálneho úložiska. [7]

Webové prehliadače majú boolean vlastnosť window.navigator.onLine, ktorá sa používa na kontrolu pripojenia internetu. Táto vlastnosť musí vrátiť hodnotu false ak používateľský agent nebude kontaktovať sieť, keď používateľ sleduje odkazy alebo keď skript požaduje vzdialenú stránku alebo inak vráti hodnotu true. Ak náhodou táto vlastnosť nebude fungovať správne kvôli malej podpore tejto vlastnosti mnohých webových prehliadačov alebo kvôli tomu, že internet nie je stabilný, musíme napísať vlastný skript, ktorý bude pravidelne volať Ajax a bude sa pokúšať pripojiť k vzdialenému serveru, ktorý je vždy spustený ako napríklad google.com. Ak táto požiadavka zlyhá, je viac než pravdepodobné, že naša webová aplikácia je taktiež odpojená od internetu. [7]

1.2 Web Storage

Web storage je samostatné API jazyka HTML5 a môžeme ju taktiež nazvať aj ako DOM storage alebo Local Storage. Zo všetkých technológií na strane klienta je web storage najrýchlejší na osvojenie a najjednoduchší na použitie. Je vhodný pre začiatočníkov, ktorý sa snažia tieto technológie ešte len naučiť alebo je vhodný pre malé aplikácie, či databázy. Zameriava sa na nastavovanie a získavanie jednoduchých hodnôt pomocou kľúčov. Ako príklad sme si uviedli že, hodnotu Janko dáme do kľúča s menami alebo hodnotu 50 do kľúča s rokmi. Komplexnejšie dáta ako objekty nie sú podporované touto technológiou. Tieto údaje sa vo všeobecnosti pohybujú od 5 do 10 MB.

Pomocou tejto technológie môžu webové aplikácie ukladať údaje lokálne v prehliadači používateľa. Web storage má oveľa väčšie úložisko ako jeho predchodcovia a informácie sa nikdy neprenášajú na server. Ukladá sa podľa domény a protokolu. To znamená, že všetky stránky môžu ukladať, pristupovať a čerpať dáta z jedného zdroja. [1]

Táto metóda je síce podporovaná vo väčšine prehliadačov, ale jeho aktívny vývoj bol spomalený z dôvodu problémov s výkonom. Ako lepšia alternatíva sa považuje takzvaná indexovaná databáza, v ktorej môžeme ukladať, indexovať a vyhľadávať dátové objekty ako napríklad indexedDB. Avšak pre začiatočníkov je táto metóda stále veľmi používaná a húfne rozšírená. [2]

Web storage ponúka dva objekty, a to sú localStorage a sessionStorage, pričom obidva sú široko podporované. Túto technológiu môžeme nájsť v prehliadačoch:

- Internet Explorer 8
- Google Chrome, Google Chrome Android
- Mozilla Firefox, Mozilla Firefox for Android
- Safari, Safari on iOS
- Microsoft Edge
- Opera, Opera Android
- WebView Android
- Samsung Internet
- Deno

Ak ho náhodou prehliadač z nejakého dôvodu nepodporuje, tak tento problém sa dá vyriešiť pomocou jednoduchých polyfillov. Pollyfill je kód, ktorý implementuje funkciu do webového prehliadača bez toho, aby ten prehliadač podporoval danú funkciu. [3]

1.2.1 Predchodca Web Storage

Či už chceme alebo nie s cookies sa stretávame stále, keď si prehliadame internet. Sú to malé textové súbory, ktoré sú vytvorené webovými stránkami a sú uložené lokálne v počítači osoby, čo umožňuje webovej stránke zapamätať si informácie o tejto osobe pri ďalších návštevách web stránky. Táto funkcia sa dodnes považuje za štandard a je neoddeliteľnou súčasťou internetových funkcií. Využívame ich pri internetových nákupných košíkoch alebo pri automatickom vypĺňaní formulárov.

Ak osoba interagovala s webovou stránkou jej internetový prehliadač, pripojí sa k serveru webovej lokality priradenému k odkazu stránky a odošle mu požiadavku, na ktorú server odpovie. Tieto požiadavky sú vo forme HTTP žiadosti čo môže spomaľovať našu webovú aplikáciu. V prípade hypertextového odkazu server vo všeobecnosti odpovie webovou stránkou spojenou s hypertextovým odkazom. V júni 1994 Lou Montulli vymyslel tento koncept a vyriešil problém bezstavového prehliadania internetu. Tento názov sa odvodil od termínu „magické cookies“, ktorý sa už dávno v kruhoch informatiky používal na opis paketov dát prijatých programom a vrátených nezmenených. V podstate táto metóda funguje pomerne rovnako v dnešnej dobe ako na začiatku keď bola vyvinutá. Majú kapacitu 4 KB dát a kvôli bezpečnosti je najlepšie pre našu webovú aplikáciu, aby mala zriadené SSL pripojenie a týmto naše dáta budú chránené. [4]

Cookies programátori stále používajú a budú využívať kvôli viacerým možnostiam, ktoré táto technológia ponúka. Webová lokalita je jednou z výhod, ktoré cookies ponúka. Tá sa uloží v počítači osoby na prispôsobenie preferencií špeciálne pre užívateľa webu. Pre príklad si uvedieme, že osoba si môže nastaviť jazyk na viacjazyčnej webovej stránke a trvalý súbor cookies zaznamenávajúci túto preferenciu sa uloží do počítača. To umožňuje webovej stránke automaticky načítať preferovaný jazyk osoby pri ďalších návštevách. Alebo napríklad majitelia stránok si pomocou cookies vedia odkontrolovať aký veľký pohyb osôb je na ich stránke. Táto informácia pomôže majiteľom opraviť problémy s malou návštevnosťou web stránky a dokonca si môžu odkontrolovať koľko stálych navštevovateľov majú a podobne. [4]

1.2.2 LocalStorage

LocalStorage je API jazyka javascript a je totožné s jeho bratom sessionStorage. Pomocou nej môžeme ukladať údaje bez toho, aby nám vypršala platnosť na ich vymazanie. Údaje sa po zatvorení prehliadača nevymažú a budú v ňom dostupné už navždy, pokiaľ ich užívateľ manuálne nevymaže alebo programátor môže špecifické údaje nastaviť na to, aby sa vymazali za určitý čas.

Je podobný a môže sa používať rovnakým spôsobom ako sa používajú trvalé súbory cookies ale podobné ukladanie relácií prináša niektoré významné výhody oproti používaniu cookies. Táto metóda je spojená s funkcionalitou offline aplikácií, aby bolo možné, že používateľ bude môcť upravovať uložené súbory prostredníctvom webovej aplikácie, keď momentálne nie je pripojený k sieti. Údaje, ktoré sú v nej uložené zároveň nemôžeme zabudnúť rozumne zašifrovať, pretože pre užívateľa sú voľne viditeľné. Mohlo by to predstavovať možné bezpečnostné riziko, pretože akýkoľvek používateľ by mohol vidieť toto webové úložisko.

Tieto údaje sa ukladajú počas relácie na stránke alebo vo webovom prehliadači. Údaje, ktoré boli načítané v režime súkromné prehliadanie alebo inkognito sa vymažú po zatvorení poslednej karty tohto režimu podobne ako v sessionStorage. [5]

1.2.3 SessionStorage

SessionStorage je podobný ako localStorage a používa také isté API. Hlavný rozdiel je v tom, že zatiaľ čo platnosť údajov, ktoré si ukladá localStorage nevyprší, tak údaje v sessionStorage sa napríklad po zatvorení prehliadača alebo stránky vymažú. Ak sa načíta dokument na konkrétnej karte vo webovom prehliadači, tak tieto údaje z dokumentu sú platné pokiaľ užívateľ nezavrie konkrétnu kartu alebo samotný prehliadač.

Otvorením tej istej stránky alebo dokumentu na novej karte sa vytvoria nové údaje, ktoré sú s väčšou hodnotou kontextu ako karta, ktorá bola otvorená predtým. Duplikovaním kariet sa skopírujú údaje z predchádzajúcej karty. Zatvorením karty alebo webového prehliadača používateľ vymaže všetky údaje ktoré boli uložené na kartách. Údaje uložené v sessionStorage sú špecifické pre protokol stránky. [6]

SessionStorage má veľké využitie v bankovníctve alebo pri elektronických platbách. Ako príklad si môžeme uviesť, že ak má užívateľ otvorene okná pre tú istú transakciu nákupného košíka, je možné že sa uskutoční nákup v jednom okne a zároveň tento nákup sa spracuje v oboch oknách z dôvodu zdieľania údajov naprieč dvoma oknami.

1.2.4 Využitie Web Storage vo veľkých spoločnostiach

Web storage sa často využíva vo veľkých korporátoch ako napríklad Twitter, Google (Alphabet) alebo Amazon. Tieto spoločnosti striedajú tieto technológie kvôli efektívnosti zápisu dát na server, načítania a zrýchlenia stránky a tým spríjemňujú užívateľský pobyt na stránke. Taktiež striedajú tieto metódy, kde sú efektívnejšie, a to či už na desktope alebo na zariadení ako tablet, mobil a mnoho ďalších.

Pre mobilné zariadenia, základná vyhľadávacia stránka Google využíva pomerne často localStorage a to tým, že ukladá obrázky base64 a ďalšie súbory typu CSS. Pre každú nasledujúcu požiadavku, ktorá sa odošle na stránke používa takzvaný javascript na vloženie blokov <style> hneď za názov stránky v hlavičke dokumentu s hodnotami CSS z localStorage. Pri základnom vyhľadávaní na počítači Google tieto údaje ukladá inač ako na mobile. Najprv použije sessionStorage a tým vie, že to budú dočasné údaje. Ak sa pozrieme na nespracované údaje JSON uložené jednoduchým vyhľadávaním môžeme vidieť hlavne súbory CSS a HTML, ktoré sú uložené s niekoľkými sledovacími dátami. [3]

Je prekvapivé, že najlepšie stránky na internete stále nevyužívajú plnohodnotne webové úložisko na zrýchlenie svojich stránok a zníženie požiadaviek HTTP. Efektívne požiadavky a rozhrania zippier nemusia byť veľkým problémom pre tieto stránky na počítači, ale nie je to dôvod, prečo by sme tieto úložiská nemohli mať aj na mobilnom zariadení a aj na počítači. Niektoré z dôvodov, prečo zaznamenávame veľké využitie webového úložiska iba na mobilných zariadeniach sú :

- Identifikátori URI údajov (obrázky na pozadí s kódovaním base64) používane v CSS fungujú len s modernými prehliadačmi. Táto technika má limity a nepríjemnosti až po Internet Explorer 9.
- Mobilné latencie sú oveľa vyššie, takže do vyrovnávacej pamäte na mobilných zariadeniach sa môže výrazne urýchliť používateľské rozhranie. [3]

1.3 IndexedDB

IndexedDB je databáza na ukladanie transakčných objektov v prehliadači. Transakčný je práve v tom, že akcie, ktoré sa v ňom vykonajú sú zoskupené do transakcií. Buď sa tieto transakcie všetky vykonajú úspešne alebo zlyhajú. Tento transakčný aspekt tejto metódy sa najlepšie opíše na príklade. Ak užívateľ webu, ktorý sa volal napríklad Adam, chce poslať svojmu kamarátovi Janovi 5 EUR, tak stránka má za úlohu dve veci. Za prvé, odpočítať Adamovi z účtu peniaze a za druhé ich Janovi pripočítať. Ak prvá akcia zlyhá kvôli tomu, že Adam nemá dostatok peňazí na účte, tak zlyhá aj druhá akcia, ktorá mala pripočítať Janovi peniaze na účet.

IndexedDB využíva indexy, preto je zaradená medzi indexované databázy. Index môžeme pridať do ľubovoľného skladu objektov a získať len nami požadovanú informáciu. Môžeme tak vyhľadať napríklad, ak hľadáme posledných 20 užívateľov, ktorí boli prihlásení do našej aplikácie. Taktiež táto databáza je založená len na prehliadači. K údajom je možné pristupovať bez ohľadu na to, či je užívateľ pripojený alebo nie. Ak sme zmenili niečo v lokálnej databáze, tak sa na serveri neprejaví žiadna zmena. Jediný spôsob ako lokálne zmeny v databáze zmeniť aj na serveri je preniesť tieto zmeny manuálne na server.

Rovnako to funguje aj v prípade storage. Táto databáza ukladá údaje ako páry, kľúč-hodnota, ale zároveň ponúka aj transakčnú manipuláciu s objektmi. Dokonca vytvára indexy uložených objektov pre rýchle vyhľadávanie. Pomocou webového úložiska môžeme ukladať iba reťazce a toto môžeme obísť pomocou príkazov JSON stingify() a parse(), aby sme týmto reťazcom poskytli určitú štruktúru. Prostredníctvom nej môžeme priamo ukladať a indexovať bežné objekty JavaScript. [7]

Pomocou IndexedDB môžeme pristupovať k údajom asynchrónne, takže pri prístupe k veľkým objektom na disku nedochádza k zamrznutiu používateľského rozhrania. IndexedDB takisto používa udalosti DOM pre všetky upozornenia. Úspešné udalosti, takzvane „nebublajú“ zatiaľ čo chybové udalosti áno. Používatelia budú mať pocit, že aplikácia reaguje čo by nebolo v prípade, ak by sme tento prípad riešili pomocou Web Storage. Podobne ako pri Web Storage je prístup k databázam IndexedDB regulovaný politikou rovnakého pôvodu. [7]

1.3.1 Princíp NoSQL

Je založený na princípe NoSQL databázy. Na rozdiel od tradičných databáz ako napríklad MySQL alebo SQL Server, ktoré majú klasické tabuľky, v ktorých sa nachádzajú preddefinované riadky a stĺpce, databáza úložiska objektov ukladá objekty. Každá databáza môže obsahovať viacero objektových skladov a každý z nich môže obsahovať mnoho objektov. Tieto „objekty“ môžu byť objekty javascriptu, čísla, bloby alebo väčšina ďalších jednotiek údajov, ktoré javascript môže spracovať.

V skratke NoSQL sa vyznačuje štyrmi hlavnými znakmi.

- Agnostická schéma – s NoSQL databázou nie je potrebná schéma čo vám dáva slobodu ukladať priamo informácie bez toho, aby ste si museli dopredu vytvoriť a navrhnúť schému.
- Nerelačná – vzťahy v databáze a spojenia medzi nimi vytvárajú tabuľky údajov.
- Komoditný hardvér – môžeme použiť aj lacné štandardné servery. Nepotrebujeme drahý špecializovaný hardvér.
- Vysoko distribuovateľné – to znamená že dokážu uchovávať a spracovávať informácie na viac ako jednom zariadení. [8]

1.4 SQLite

SQLite je softvérový balík, ktorý je vo verejnej doméne a poskytuje systém správy relačných databáz alebo inak pod skratkou RDBMS. Relačné databázové systémy sa používajú na ukladanie presne definovaných záznamov vo veľkých tabuľkách. Okrem týchto dvoch vecí dokáže databáza aj spracovávať komplexné príkazy a dotazy, ktoré kombinujú dáta z viacerých tabuliek a vytvárajú zostavy a súhrny týchto údajov.

Na jednej strane máme databázové systémy ako napríklad Oracle Database, IBM DB2 a Microsoft SQL Server, ktoré sú komerčné a na druhej strane máme MySQL alebo PostgreSQL, ktoré sú produkty s otvoreným zdrojovým kódom. Tiež sa nazývajú ako „open source“.

SQLite je vstavaná databáza. Namiesto toho, aby bežala ako samostatný proces, tak ona skôr koexistuje v „symbióze“ vo vnútri aplikácie, ktorej slúži v rámci svojho procesného priestoru. Jeho kód je prepletený alebo je vložený ako súčasť programu, ktorý ho hostí. Preto nie je žiadna potreba konfigurácie siete ani jej správa. Tým pádom nemusíme riešiť žiadne brány firewall alebo adresy, o ktoré by sa mala táto metóda starať. [9]

Najlepšia vlastnosť tejto metódy je jej všestrannosť. Je to databáza, programovacia knižnica a nástroj príkazového riadka ako aj zároveň vynikajúci vzdelávací nástroj, ktorý poskytuje dobrý úvod do relačných databáz pre študentov. To isté platí aj o tom, že ju dokážeme využiť na vývoj a testovanie, ba dokonca aj ako vyrovnávaciu pamäť. Môžeme uchovávať konfiguračné údaje alebo vďaka využitiu svojej binárnej kompatibility naprieč platformami môže dokonca fungovať ako formát súboru aplikácie. [9]

Lite v názve SQLite znázorňuje jeho jednoduchosť v oblasti nastavení, administratívy a využitia zdrojov. Nepotrebuje žiaden klient alebo serverovú architektúru na to, aby táto databáza fungovala. Celý databázový engine je integrovaný do akejkoľvek aplikácie, ktorá potrebuje prístup k databáze. Jediná vec, ktorá sa zdieľa medzi aplikáciami a zdrojom je jeden databázový súbor, ktorý je uložený na disku. Má to len jedinú nevýhodu. Je navrhnutý tak, aby riešil lokalizované úložiská, a to znamená, že nie je vhodný pre situácie, keď viacero klientskych počítačov potrebuje prístup k centralizovanej databáze. [10]

SQLite používa systém dynamického typu pre tabuľky to znamená, že engine umožňuje vložiť akúkoľvek hodnotu do takmer akéhokoľvek stĺpca bez ohľadu na typ. Taktiež dokáže manipulovať s viacerými databázami čo umožňuje spracovávať príkazy, ktoré by viaceré databázy nedokázali. A ako najzaujímavejšia schopnosť tohto softvéru je vytváranie databáz vo vyrovnávacej pamäti súborov, čo urobí to, že transakcie sú veľmi rýchle ak máme dostatok pamäte RAM, a tým sú skvelé na ukladanie dočasných tabuliek alebo iných prechodných údajov. [10]

1.5 WebSQL

WebSQL je technológia na vykonávanie SQL príkazov. Výsledky tejto technológie sa zobrazujú cez internet. Spomínaná technológia bola vyvinutá na univerzite v Minnesote a bola postavená pomocou ASP a ADO od firmy Microsoft. Akceptuje len príkazy jazyka SQL z webového prehliadača ako súčasť HTTP žiadosti, vykoná príkaz, ktorý mu bol daný a napokon vytvorí súbor HTML s výsledkami a tento súbor odošle naspäť.

Kvôli tomu, že webový server generuje statické HTML súbory, WebSQL bude fungovať v akomkoľvek webovom prehliadači. Je tiež veľmi vhodný pre tých, čo pristupujú na internet cez dial-up spojenie. [11]

Na pripojenie k WebSQL databáze je potrebné zadanie správneho mena a hesla. Po tomto sa užívateľovi zobrazí biela obrazovka, kde vypíše SQL príkaz napríklad na

vypísanie údajov z databázy a po zadaní správneho príkazu mu WebSQL vypracuje zadaný príkaz. Po tomto si užívateľ dokáže tieto údaje stiahnuť a takto si ich dokáže napríklad používateľ premiestniť na inú SQL databázu. [11]

1.6 PouchDB

PouchDB je open source databáza javascript inšpirovaná Apache CouchDB, ktorá je navrhnutá tak, aby dobre fungovala v prehliadači. PouchDB bol vytvorený aby pomohol webovým vývojárom vytvárať aplikácie ktoré fungujú rovnako dobre offline ako online.

Táto metóda má výhodu ako väčšina moderných asynchrónnych rozhraní javascript API a tou je, že používa Promises na modelovanie všetkých svojich asynchrónnych operácií, čo znamená že môžeme použiť Rx.Observable.fromPromise() na prispôsobenie všetkých API. Týmto táto technológia ľahšie interaguje s databázovým kódom a všetko je zabalené a koordinované prostredníctvom pozorovateľných operátorov. Toto je len jedna s mála funkcií pouchDB. Okrem toho dokáže vytvoriť podporu pre reaktívnu databázu. Je to zároveň úložisko dokumentov bez schém, takže to znamená že pred zápisom údajov do nej nemusíme definovať a vytvárať rôzne schémy a tabuľky. [12]

Môžeme o nej ba dokonca povedať, že je mapová a redukčná databáza čo znamená, že ak chceme do nej zadať údaje, musíme najskôr definovať ako funguje projekcia alebo funkcia mapovania. Tento objekt sa nazýva aj ako návrhový dokument, ktorý musíme zahrnúť ako súčasť dotazu. Zároveň má taktiež niektoré vlastné redukčné operácie. Tým pádom môžeme vykonávať ďalšie veci ako napríklad súčet alebo štatistické operácie. [12]

1.7 Lawnchair

Lawnchair je ľahká knižnica javascript a je jednou z možných odpovedí na ukladanie klientskej databázy, pretože používa jednoduché priradenie páru názov a hodnota, ktoré idú cez JavaScript a popritom získava hodnoty cez JSON. Môže byť tiež vybavený mnohými adaptérmi, ak niektoré prehliadače neakceptujú iné technológie ukladania na strane klienta a ak by zlyhali tak vtedy nastúpi lawnchair. [13]

Lawnchair uľahčuje fragmentáciu medzi technológiami ukladania a poskytuje vývojárom jednoduchý prístup k ukladaniu hodnôt. Navyše má veľmi malých 6 KB, keď je minifikovaný a ešte menší, keď sú dáta zazipované. To môže byť veľmi prospešné pri

vytváraní offline archívu používateľských interakcií a prípadne pri opakovaní interakcií a spúšťaní používateľských aktivít, keď používateľ znovu získa sieťové pripojenie. [13]

1.8 Porovnanie technológií

Porovnávanie týchto technológií je v podstate ťažšie ako sa zdá. Pri výbere tej najvhodnejšej z nich pre našu webovú aplikáciu sme si ich museli najprv porovnať. Niektorí programátori použijú tú, ktorá im je najbližšia a v ktorej majú dlhodobé skúsenosti a ktorá im najviac sedí. Ale väčšina sa zhodne na tom, že lepšie je posúdiť aký je veľký daný projekt, na ktorom ideme robiť, čo od toho projektu očakávame, či má byť ten projekt optimalizovaný a „bežať ako hodinky“ alebo stačí nejaká jednoduchšia metóda, ktorá nie je zbytočne až tak komplikovaná. Po usúdení týchto javov si urobíme prieskum aké technológie a metódy sú dostupné na trhu, z ktorých si môžeme vyberať. Po tomto prieskume sme zistili, že najlepšie je pre našu malú webovú aplikáciu urobiť to čo najjednoduchšie, keďže nepotrebujeme nič komplikované a zbytočne zdĺhavé.

Vybrali sme si štyri metódy na skúmanie a to sú:

- localStorage
- sessionStorage
- indexedDB
- webSQL.

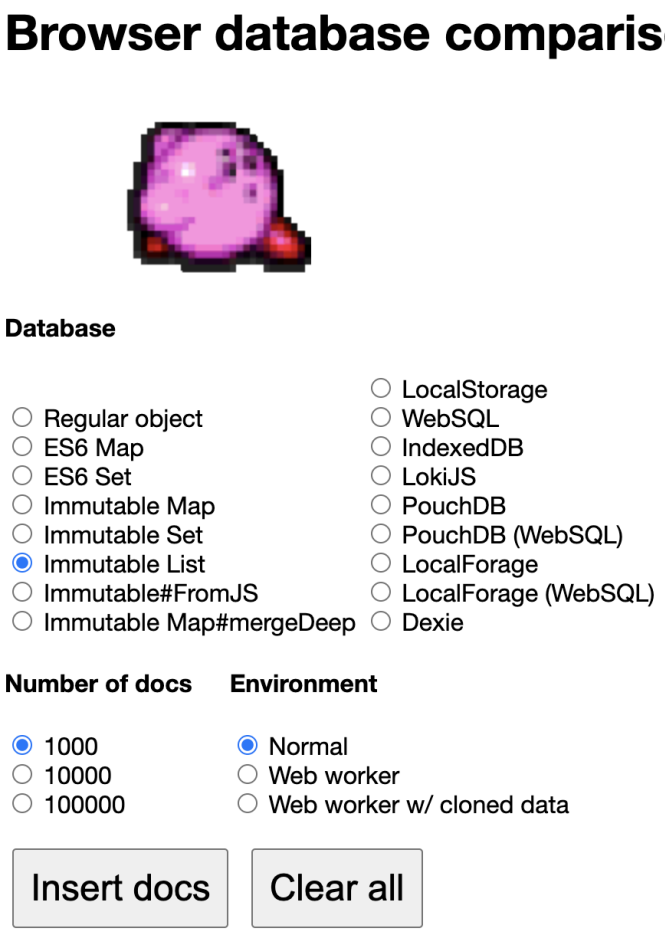
Predtým ako ich budeme porovnávať sme si museli určiť podľa čoho ich budeme hodnotiť, ktorá je pre nás tá najlepšia. A preto sme si určili tri kritéria, podľa ktorých sme porovnávali naše štyri opísané technológie a napokon aj vybrali tú najvhodnejšiu pre náš projekt. Tie tri kritéria, ktoré sme si zvolili sú kapacita technológie, rýchlosť akou sa zapisujú dáta do databázy a ako posledným kritériom je celkové zotrvanie dát v týchto technológiách. Kapacita technológie a zotrvanie dát v technológií bolo asi to najjednoduchšie, keďže tieto informácie sú voľne dostupné na internete alebo v knižných zdrojoch. Problém je s rýchlosťou zápisu dát, pretože na výpočet sme potrebovali buď pomocnú web stránku (viď obrázok 1), ktorá nám vypíše za koľko milisekúnd sa objekt načítal do nami vybratej technológie alebo potrebujeme napísať kód, ktorý nám vypočíta túto požiadavku.

1.8.1 Využité prostriedky na porovnávanie

Toto porovnávanie sme vykonávali na notebooku, ktorý má názov Apple MacBook Pro z roku 2019. Je to 13 palcová verzia tohto prenosného počítača ktorý má 4-jadrový Intel Core i5 2,4 GHz procesor, 8 GB 2133 MHz LPDDR3 pamäte RAM a grafickú kartu, ktorá je Intel Iris Plus Graphics 655 1536 MB. Táto informácia je dôležitá, keďže rýchlosť zápisu údajov sa bude líšiť podľa niekoľkých faktorov. Tieto faktory môžu byť napríklad také, že niekto môže použiť na toto testovanie iný hardvér, softvér, kód, operačný systém, iné rýchlosti internetového pripojenia, či je pripojený káblom alebo bezdrôtovo alebo napríklad v tej dobe boli iné znalosti o problematike tohto typu a v budúcnosti to môže byť inač kvôli novým inováciám v tejto problematike a tak ďalej.

Testovanie sme taktiež mohli vykonať na mnohých webových prehliadačoch ale my sme si zvolili len jeden, a to Google Chrome. Toto rozhodnutie sme zvolili kvôli tomu, že podľa webovej stránky statista.com je Chrome najpoužívanejší webový prehliadač na trhu. Až 65% užívateľov čo je 6 z 10 ľudí ktorý používajú nejaký internetový prehliadač majú práve Chrome. Takže je najlogickejšie, že sme si zvolili ten najviac populárny.

Obrázok 1 - Pomocná testovacia stránka, zdroj: Autor práce



1.8.2 Výsledky porovnávania technológie LocalStorage

Ako prvé z kritérií sme zisťovali veľkosť technológie. LocalStorage podľa stránky web.dev má kapacitu okolo 5 až 10 MB dát, ktoré sa odlišujú podľa toho, aký webový prehliadač používateľ našej webovej aplikácie používa. Pri Safari 8 je to maximálne 5 MB pričom pri Google Chrome je to zasa 10 MB.

Ohľadom zotrvačnosti dát v tomto úložisku sme zistili, že tieto dáta v tejto metóde v nej zotrvávajú navždy pokiaľ ich manuálne nevymažeme alebo im programátor nenaprogramuje nejakú „životnosť“, po ktorej sa tieto položky vymažú zo samotného úložiska.

Čo sa týka rýchlosti zápisu faktov na úložisko, museli sme si prv napísať v skratke náš kód (viď obrázok 2). Tento jednoduchý kód sme spustili na webovom prehliadači známy ako Google Chrome. Kód sme naprogramovali tak, že ako prvé sme si vytvorili 10 000 položiek bez žiadnych vložených informácií a ako ďalší krok sme si vytvorili jednoduchú slučku, kde sme zapísali tieto vložené položky do konzoly čo v našom prípade bolo 10 000. Na riadku 22 až 24 sme zadali a vybrali dáta z localStorage. Po tomto sme už len napísali príkaz na získanie času zápisu na začiatku a na konci merania, ktoré sme v závere už len od seba odčítali a vyšiel nám výsledok. Výsledok bol viditeľný v konzole vývojového prostredia (viď obrázok 3).

Obrázok 2 – Ukážka kódu na testovanie localStorage, zdroj: Autor práce

```
<script>
    var count = 10000;

    var data = {
        items: []
    };

    for (var i = 0; i < count; i++) {
        data.items.push(i);
    }
    console.log('data: ', data);

    var start = performance.now();

    localStorage.setItem('Test', JSON.stringify(data));
    var cache = localStorage.getItem('Test');
    cache = cache ? JSON.parse(cache) : {};

    var end = performance.now();

    console.log('Zápis sa urobil za ' + (end - start) + 'ms.');
</script>
```

Obrázok 3 - Ukážka výstupu kódu, zdroj: Autor práce

1.8.3 Výsledky porovnávania technológie SessionStorage

Spomínali sme, že localStorage a sessionStorage sú ako keby dvojičky. Tým pádom kapacita tejto technológie je totožná s kapacitou localStorage ktorá je od 5 do 10 MB.

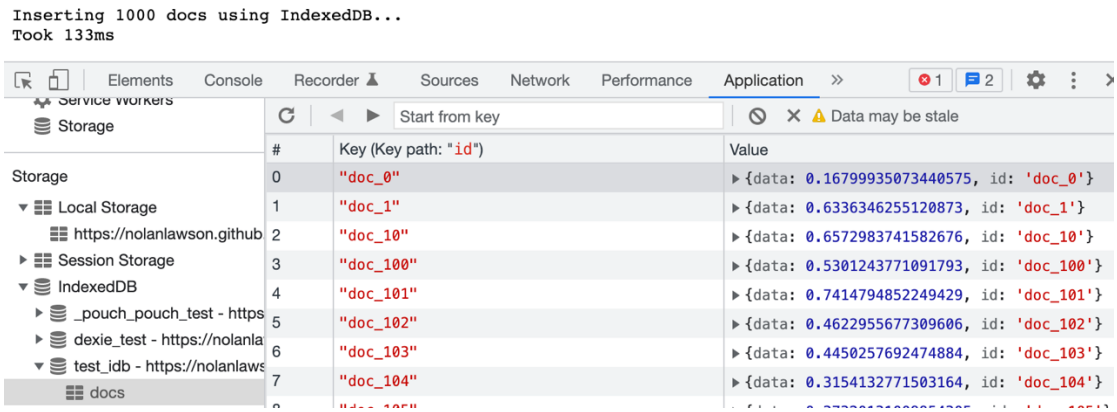
To isté môžeme povedať aj o zápise dát, keďže odlišnosť tejto technológie od svojej dvojičky je zanedbateľná. Jediné v čom sa odlišuje sessionStorage a localStorage je zotrvanie dát v celkovom úložisku. Dáta v ňom zotrvávajú len pokiaľ máme otvorenú webovú stránku alebo webový prehliadač. Ak jedno z týchto veci zatvoríme a prerušíme spojenie, tak sa dáta na tejto technológií vymažú. Tým pádom sme zistili, že táto technológia nám vôbec nevyhovuje pre náš projekt.

1.8.4 Výsledky porovnávania technológie IndexedDB

Pri indexedDB je to kúsok zložitejšie. Kapacita pri tejto technológií je omnoho väčšia ako pri localStorage a sessionStorage. Táto kapacita sa zväčša pohybuje okolo 10 MB až po 2 GB. Pre každý webový prehliadač existujú iné pravidlá ohľadom kapacity indexedDB. Zobrali sme si pre príklad prehliadač Mozilla Firefox. Tento prehliadač má pravidlo, že ak dáta presiahnu limit 50 MB, tak Firefox požiada používateľa o uvoľnenie ďalšieho miesta na disku. Toto pravidlo má zväčša každý prehliadač ale môže sa stať, že nejaký má iné pravidlo. V podstate táto technológia má ako keby neobmedzenú kapacitu, keďže jej veľkosť závisí od veľkosti pevného disku a od operačného systému používateľa.

Pri zotrvaní dát je to jednoduché. Je také isté ako pri localStorage a sessionStorage. A ako posledné je zápis objektov. Túto vlastnosť má indexedDB dosť dobrú, čo sme si mohli prečítať na rôznych fórach a v mnohých knihách. Ale testovanie nám ukázalo niečo úplne iné. Môže to byť tým, že testovacia stránka, ktorú sme používali na toto porovnávanie nemá optimalizovaný kód alebo autor tejto stránky nemal dostatočne informácie o riešení tejto problematiky. Na obrázku (viď obrázok 4) sme mohli vidieť zapísané objekty v úložisku technológie indexedDB pričom rýchlosť zápisu a počet zapísaných objektov sme mohli vidieť v ľavom hornom rohu obrázka.

Obrázok 4 - Ukážka objektov v technológií IndexedDB, zdroj: Autor práce



1.8.5 Výsledky porovnávania technológie WebSQL

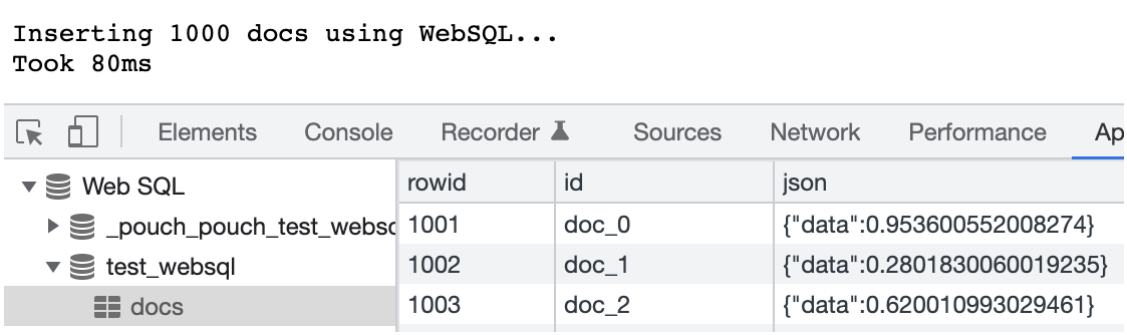
Dôvod prečo sme si na porovnávanie vybrali WebSQL a nie SQLite je ten, že s WebSQL máme viac skúseností. Taktiež sa nachádza na našej testovacej stránke a aby sme nemali skreslené výsledky a všetko bolo narovnako vybrali sme si práve túto technológiu. V podstate obidva sú rovnaké čo sa týka reakcie na príkazy jazyka SQL. Tým pádom máme porovnanie z každého kúska, keďže z indexovaných databáz máme napríklad IndexedDB a z technológií, ktoré pracujú na základe SQL enginu máme WebSQL.

Kapacita tejto technológie je 5 MB. Zdroje pri tejto technológií ukazujú na to, že limit sa mení podľa toho, aký webový prehliadač používame alebo na akom zariadení sme, keďže kapacita je iná na počítači a iná na mobile. Tento limit sa dá zväčšiť, ak k tomu pridáme rôzne pluginy, ktoré môžeme nájsť na internete.

Zotrvanie dát je také isté ako v predošlých technológiách. Dáta sa nevymažú do tej doby, kým ich manuálne vymažeme.

Čo sa týka rýchlosti zápisu objektov alebo dát na túto technológiu, tak v tom je podstatne rýchlejšia ako IndexedDB a kúsok pomalšia ako localStorage (viď obrázok 5).

Obrázok 5 - Ukážka objektov v technológií WebSQL, zdroj: Autor práce



1.9 Výsledky porovnávania

Pri vyberaní správnej technológie sme sa pozerali hlavne na to, aby rýchlosť zápisu pri najmenšom počte položiek bola čo najrýchlejšia. Takisto pri výbere nemusíme až tak dbať na kapacitu technológie, keďže v našom prípade sa budeme zaoberať malou webovou aplikáciu, kde nie je potrebná veľká kapacita. Čo sa týka zotrvania dát, tak budeme trvať na tom, že chceme, aby dáta zotrvávali v databáze čo najdlhšie. Túto podmienku spĺňa každá jedna technológia okrem sessionStorage.

Výber v našom prípade bol jednoznačný, a to localStorage, keďže každú našu podmienku spĺňal či už tým, že je najrýchlejší pri zápise menších počtov položiek alebo tým že v ňom zotrvávajú dáta aj naďalej pokiaľ ich manuálne nevymažeme.

Obrázok 6 - Výsledky porovnávania, zdroj: Autor práce

Vložených 1 000 objektov			
Číslo testu	localStorage	indexedDB	webSQL
1	15	157	199
2	14	118	89
3	12	117	97
4	14	122	86
5	14	143	88
6	15	130	94
7	15	122	92
8	12	208	88
9	11	121	93
10	14	118	91

Vložených 10 000 objektov			
Číslo testu	localStorage	indexedDB	webSQL
1	97	913	188
2	109	1 210	201
3	85	1 253	197
4	83	1 235	198
5	85	1 274	204
6	84	1 258	196
7	85	1 246	207
8	85	1 284	200
9	82	1 272	195
10	85	1 241	210

Vložených 100 000 objektov			
Číslo testu	localStorage	indexedDB	webSQL
1	924	10 407	2 020
2	952	13 384	1 350
3	819	14 916	1 399
4	923	14 115	1 609
5	820	13 685	1 635
6	904	10 334	1 511
7	861	14 896	1 502
8	804	14 153	1 612
9	813	14 472	1 535
10	835	13 778	1 492

2. Cieľ práce, metodika práce a metódy skúmania

Naša bakalárska práca je zameraná na problematiku využitia offline webovej stránky. V tejto kapitole budeme rozoberať cieľ našej záverečnej práce, podrobne popisovať použitú metodiku práce a zároveň aj použité metódy spracovania práce. Na základe realizovaného projektu sme sa rozhodli navrhnúť a implementovať jednoduchý webový formulár.

2.1 Cieľ práce

Cieľom práce je navrhnúť a implementovať jednoduchý webový formulár (stránku), ktorá dokáže fungovať aj v offline režime. Dáta, ktoré sú do stránky vložené počas nedostupnosti siete/internetu sa prenesú na server po pripojení. Práca nájde vhodnú technológiu aj koncept.

2.2 Metodika práce

- Ako prvý krok sme si analyzovali všetky populárne a verejnosťou často používané dátové úložiska na strane používateľa / klienta.
- Ako ďalší krok pre nás bolo analyzovať si tie štyri najpoužívanejšie a najpopulárnejšie technológie a porovnať ich medzi sebou. Urobili sme to pomocou pomocnej webovej stránky, ktorá bola voľne dostupná na internete a ktorú autor sprístupnil verejnosti presne kvôli takémuto dôvodu na testovanie a skúšanie, aby sme zistili ktorá z týchto úložísk je najlepšia a koľko ktorá zvládne.
- Po tomto kroku sme zistili, že pre našu webovú aplikáciu je najlepšie použiť technológiu localStorage. Tá mala spomedzi všetkých ostatných metód ten najrýchlejší zápis, dátové úložisko pre našu aplikáciu je viac než dostačujúce a samozrejme jej zotrvačnosť dát sa nesmie zabudnúť.
- Túto metódu ktorú sme využívali pre našu webovú stránku bolo potrebné napísať v jazyku javascript. Nasledovne sme si vytvorili jednoduchú webovú stránku, kde sme vytvorili dve polia, do ktorých sa dalo písať nami ľubovoľné údaje, ktoré sme mali za cieľ vložiť do databázy a do našej samotnej technológie. A ako druhé sme tam vložili obyčajné tlačidlo, na ktoré keď klikneme, tak údaje z polí sa vložia do vyššie spomínaných technológií.
- Ďalší krok bolo samozrejme vytvoriť javascript server, ktorý prepojí našu webovú aplikáciu a MySQL databázu. Zvolili sme si preto node.js ktorý je

všeobecne používaný pri tvorbe webových aplikácií a s ním sme si nainštalovali rôzne knižnice ako Express, CORS a mnoho ďalších, ktoré nám umožnili túto aplikáciu spojazdniť.

- Databázu do ktorej sa nám tieto údaje zapisujú sme si zvolili MySQL, pretože nám bola blízka svojím rozhraním, štruktúrou a hlavne máme v nej skúsenosti z minulosti takže nám nebola neznáma.
- Ako posledný a najdôležitejší krok bol určite prepojiť tieto tri technológie medzi sebou pomocou programovacieho jazyka javascript.

2.3 Metódy skúmania

Pri riešení problematiky a analýze rôznych zdrojov sa nám najviac osvedčili tie od zahraničných autorov. Porovnávali sme si technológie pomocou štyroch kritérií, ktoré sme si určili a na následné skonštruovanie webovej aplikácie sme použili overované zdroje, čo boli v našom prípade buď knihy alebo internetové zdroje.

3. Výsledky práce a diskusia

V predchádzajúcich kapitolách sme vysvetlili aký je súčasný stav riešenia problematiky doma a v zahraničí, aké metódy a postup sme využívali na riešenie tejto problematiky a vysvetlili sme pojmy súvisiace s našou témou ako je offline podpora webovej aplikácie / stránky a ako túto podporu vieme cez aké technológie vykonať, prípadne spojazdniť. V tejto kapitole zasa zhodnotíme k čomu sme dospeli, čo by sa dalo vylepšiť alebo poňať lepšie a nakoniec čo sa nám podarilo urobiť a či sa to podarilo podľa našich očakávaní.

Zároveň táto kapitola obsahuje popis našej výslednej offline webovej stránky, ktorá mala hlavnú úlohu a tou je že, tento webový formulár dokáže fungovať aj v offline režime. Dáta, ktoré sú uložené v našej vybratej technológií, ktorou je localStorage budú odoslané jednoducho cez server do databázy MySQL. Súčasťou je samozrejme aj popis servera, databázy MySQL a programovacieho jazyka, v ktorom je napísaná aplikácia.

3.2 Vývojové prostredie

Vývojové prostredie tiež nazývané ako IDE je softvér, ktorý používame pri programovaní. Je to prostredie pre programátorov, aby vyvíjali aplikácie a mohli programovať bez toho aby v živom prostredí čokoľvek porušili. Nasadzujú sa na ňom rôzne zmeny prostredia predtým ako sa prostredie dostane na napríklad živú webovú stránku. Je to softvérový balík, ktorý sa používa a je navrhnutý, tak aby maximalizoval produktivitu a efektivitu programátora. Predstavuje testovacie miesto, kde si vývojári aplikácií môžu testovať hocičo a bez obáv čo má vplyv na koncových užívateľov samotnej aplikácie, ktorý používajú ju v reálnom čase. Vo väčšine sa používa s localhostom alebo na serveri, kde sa stiahne zdrojový kód webovej stránky.

Pre vývojové prostredie na náš záverečný projekt sme si vybrali prostredie Visual Studio Code. Toto bezplatné vývojové prostredie od firmy Microsoft je najviac používané vývojové prostredie. Až 70% developerov používa Visual Studio Code ako ich hlavné vývojové prostredie.

3.3 Použité programovacie jazyky

Na tvorbu tejto webovej stránky sme zvolili prevažne programovací jazyk, ktorý sa nazýva javascript. Ako druhý sme zvolili samozrejme jazyk html, kvôli samotnej tvorbe webovej stránky, ktorý je nevyhnutný na tvorbu webových stránok.

3.3.1 Javascript

Je to programovací jazyk, ktorý je prevažne využívaný v moderných prehliadačoch, hrách, stolných počítačoch, konzolách, tabletoch či mobilných telefónoch. Všetky tieto spomínané hardvéry zahŕňajú takzvaných tlmočníkoch javascript vďaka čomu je javascript najrozšírenejším programovacím jazykom v histórii. Tento jazyk sa mnohí musia naučiť kvôli tomu, že sa v ňom píše napríklad v HTML špecifikácia obsahu webových stránok alebo v CSS špecifikácia prezentácie a tak ďalej. Je to vysokoúrovňový, dynamický, netypizovaný, interpretovaný programovací jazyk, ktorý je vhodný pre objektovo orientované a funkčné štýly programovania. Odvodzuje si syntax z Javy a svoje prvotriedne funkcie zo Scheme a dedičnosť má založenú na prototype od Self. [14]

Tento jazyk sme hlavne použili kvôli tomu, že je jednoduchý na osvojenie a písanie, je prehľadný a čo je najdôležitejšie pomocou neho sme dokázali naprogramovať localStorage a ako druhé, prostredníctvom tohto jazyka sme programovali node.js server.

3.3.2 HTML

Je to najnákladnejší článok ako postaviť webovú stránku. Tento jazyk definuje význam a štruktúru webového obsahu. Podobné technológie okrem jazyka HTML sa vo všeobecnosti používajú na opis vzhľadu alebo prezentácie webovej stránky ako napríklad jazyk CSS alebo sa používajú na funkčnosť a správanie webu ako napríklad vyššie spomínaný javascript. Hypertext označuje odkazy, ktoré navzájom spája webové stránky dohromady, či už je to v rámci jednej webovej lokality alebo medzi viacerými lokalitami. Odkazy sú základným aspektom tohto jazyka a odovzdaním na internet a jeho prepojením so stránkami, ktorý vytvorili iní ľudia sa stávame aktívnym účastníkom WWW. [15]

3.4 Server a použité knižnice

Pre našu webovú aplikáciu sme zvolili server node.js. Samozrejme ako knižnice sme museli nainštalovať MySQL kvôli databáze, CORS, bodyparser a pravdaže aj express.

3.4.1 Node.js

Mnohí možno ani nevedia, že node používajú alebo ho majú nainštalovaný v stolnom počítači a nemajú o tom ani poňatia. Je veľmi populárny a mladý, keďže debutoval v roku 2009. Je druhým najsledovanejším projekom na GitHub a má viac ako 15 000 komunitných modulov publikovaných v NPM správcovi balíkov. Predstavuje platformu postavenú na

javascripte a webovom prehliadači. Používa sa na jednoduché vytváranie rýchlych a škálovateľných sieťových aplikácií. Ako ďalšie využíva udalosťami riadený a neblokujúci vstup / výstup model, vďaka ktorému je ľahký a efektívny. Je ideálny pre dátovo náročné aplikácie v reálnom čase, ktoré bežia na distribuovaných zariadeniach. [18]

Tento server je jeden z najviac používaných javascript serverov nielen pri tvorbe webu, ale aj pre aplikácie a backend celkovo. Preto je samozrejme, že sme si zvolili práve tento server.

3.4.2 MySQL

MySQL má viac ako 10 miliónov inštalácií a pravdepodobne je tou najpopulárnejšou široko používanou databázou pre riadiaci systém a pre webové servery. Táto databáza bola vyvinutá v polovici 90. rokov 20. storočia a je to vyspelá technológia, ktorá poháňa mnoho dnešných najnavštevovanejších internetových destinácií. Jeden z dôvodov prečo je MySQL tak populárne je fakt, že podobne ako PHP je jeho používanie bezplatné. Mnohé iné databázy sú cenovo neprijateľné. Zaujímavé je to, že síce je bezplatný, ale je zároveň extrémne výkonný a výnimočne rýchly. Dokáže bežať aj na tých najnákladnejších hardvéroch a sotva to zaťaží systémové prostriedky. Je taktiež vysoko škálovateľný čo znamená, že rastie s našou webovou stránkou. Základom je štruktúrovaná zbierka záznamov alebo údajov uložených v počítačovom systéme a organizované tak, aby sa v nich dalo rýchlo vyhľadávať. SQL v mene MySQL znamená jazyk, ktorý je voľne založený na angličtine a taktiež sa tento jazyk používa aj v iných databázach, ba dokonca aj v komerčných ako sú napríklad Oracle alebo Microsoft SQL Server. [16]

Keďže táto databáza je dostupná zadarmo a máme najviac skúseností s touto databázou bolo jasné, že dáme prednosť tejto databáze pred mnoho ostatnými.

3.4.3 CORS

CORS je jednoduchý spôsob ako vytvárať požiadavky HTTP z jedného miesta na druhé. Toto je úplná triviálna vec v iných programovacích jazykoch, ale na strane klienta v jazyku javascript keďže už roky výslovne bráni svojou politikou rovnakého pôvodu prehliadača. V tomto prípade CORS znie ako takzvaný proti rečník. Kladie servery pevne a zodpovedá za to, kto môže zadávať požiadavky a aké typy žiadosti sú povolené. Server má možnosť sprístupniť svoje API všetkým klientom alebo malým počtom klientov ba dokonca zamedziť prístup všetkým klientom. CORS funguje pomocou hlavičky požiadavky

a odpovede. Prehliadač a server používajú HTTP žiadosti ako oznámiť, že sa požiadavky „krížia“. Pomocou hlavičiek server môže určiť ktorí klienti môžu pristupovať k API a ktoré metódy HTTP alebo hlavičky HTTP sú povolené a či sú v žiadosti povolené napríklad súbory cookies a tak ďalej. Funguje ako taký dozorca serveru. [17]

Túto knižnicu sme samozrejme museli zaradiť do nášho servera, keďže bez tohto by nám naša webová aplikácia nešla.

3.4.4 Bodyparser

Bežná potreba všetkých druhov webových aplikácií je prijímať vstup od používateľa. Pre príklad dajme tomu, že chcete prijať pomocou značky HTML <input type=“file“>. Stačí jeden riadok kódu, ktorý pridá middlevérový komponent bodyParser(). Je to mimoriadne užitočný komponent a je to zároveň súhrn troch ďalších menších komponentov ako sú json(), urlencoded() a multipart(). Taktiež poskytuje pre webovú aplikáciu vlastnosť req.body na použitie pri analýze požiadaviek JSON. [17]

Túto knižnicu sme využili hlavne preto, že sme v HTML kóde použili JSON.stringify a potrebovali sme vytiahnuť tieto dáta na strane servera. Preto sme použili len riadok kódu na vyriešenie tohto jednoduchého problému.

3.4.5 Express

Tento webový rámec je postavený na Connect, a poskytuje nástroje a štruktúru vďaka ktorým je písanie webových aplikácií jednoduchšie a rýchlejšie. Express ponúka jednotný systém zobrazenia, ktorý nám umožňuje použiť takmer akýkoľvek nástroj šablón ktorý chceme a dokonca jednoduché nástroje na odpovedanie pomocou rôznych formátov údajov, prenos súborov, smerovanie adries URL a mnoho ďalších vecí. V porovnaní s inými aplikačnými rámcami ako sú Django, Ruby alebo Rails zaberá Express oveľa menej úložiska. Filozofia tohto rámca je, že aplikácie sa značne líšia vo svojich požiadavkách a implementáciách a ľahký rámec nám umožní vytvoriť presne to, čo potrebujeme. Zameriava sa skôr na menšie, modulárne časti funkčnosti než na monolitické rámce. [17]

Pravdaže túto knižnicu sme nainštalovali a zaradili medzi naše knižnice kvôli svojej môžeme tomu povedať neodlučiteľnosti od node.js.d. Každý ak vytvára nejakú webovú aplikáciu a programuje v javascripte a využíva node.js použije taktiež aj Express.

3.5 Predstavenie a ukážka webovej stránky

Aby sme spustili našu webovú stránku / aplikáciu, ktorú sme urobili pre tento záverečný projekt musíme urobiť pre to 3 kroky a to sú:

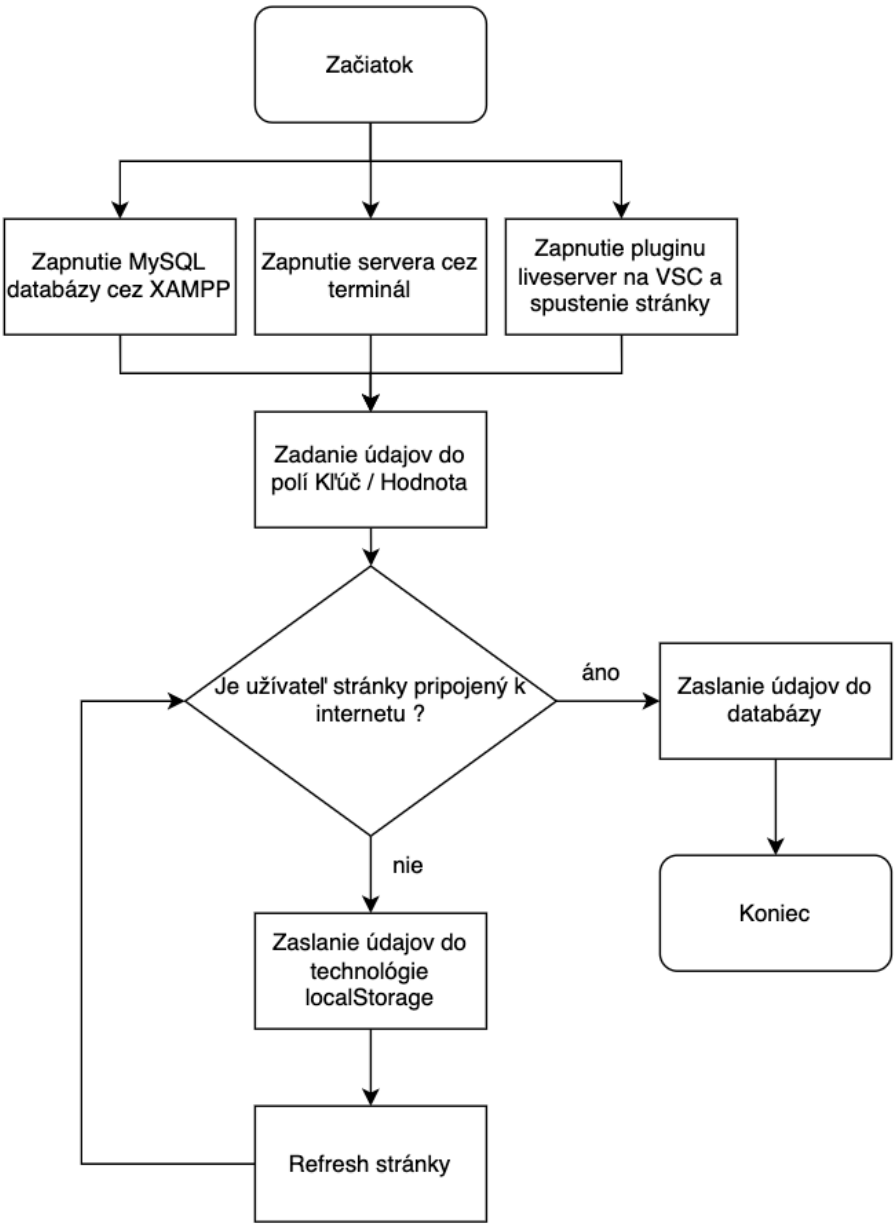
- Spustiť node.js serverové prostredie cez konzolu v počítači.
- Zapnúť naše vývojové prostredie a v ňom na plugine liveserver spustiť náš súbor index.html.
- A ako posledné zapnúť XAMPP server, kde spustíme našu MySQL databázu.

Ak tieto kroky splníme, tým sa naša webová aplikácia sa spustí a môžeme ju začať využívať naplno.

3.5.1 Vývojový diagram webovej aplikácie

Tento diagram nám pomôže s grafickým znázornením ako naša webová aplikácia funguje (viď. obrázok 7). Vyjadríme v ňom jednotlivé kroky a proces ako táto aplikácia prebieha a beží. Tento druh diagramov je často využívaný v informatike a používa sa najmä pri opisovaní procesov, počas programovania na analýzu programu, na návrhy aplikácií, dokumentáciu či riadenie procesov. Obsahujú rôzne tvary ako napríklad kosoštvorec, pomocou ktorého vetvíme postup v závislosti na splnení požiadavky / podmienky, alebo obdĺžnik, kruh a mnoho ďalších tvarov.

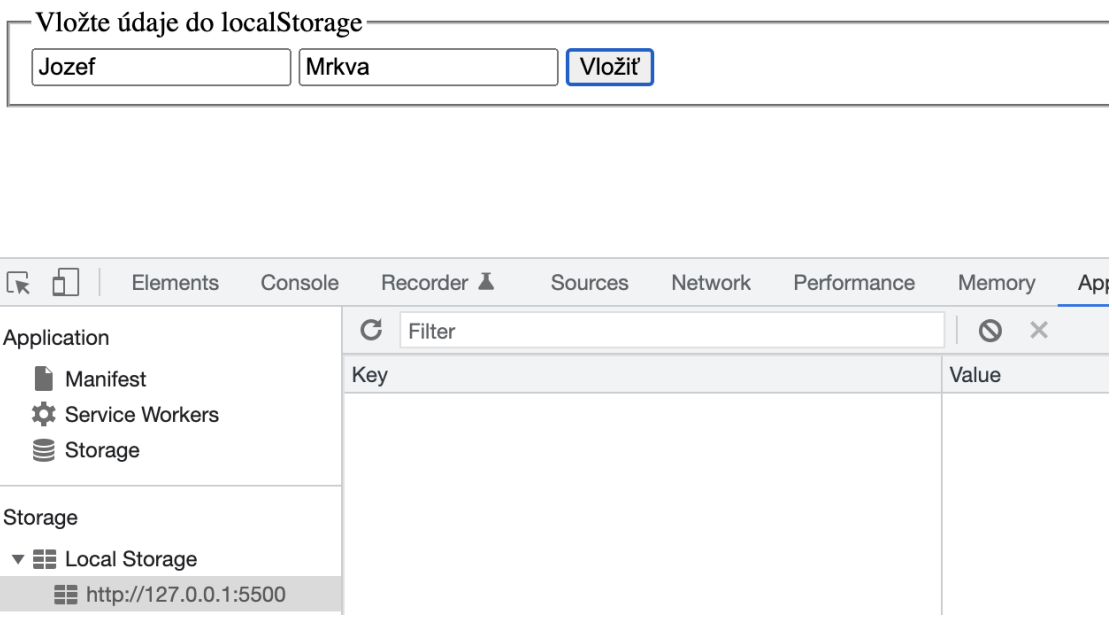
Obrázok 7 – Vývojový diagram webovej aplikácie, zdroj: Autor práce



3.5.2 Správanie webovej stránky ak je pripojená k internetu

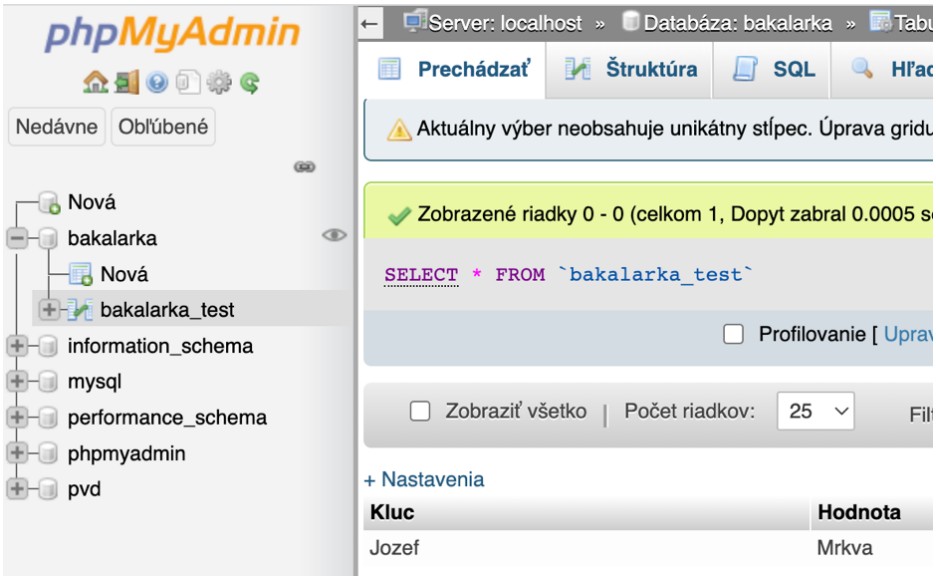
Táto ukážka zahŕňa ako sa naša webová aplikácia správala, keď bola pripojená k internetu. Naša webová stránka funguje nasledovne, prv vložíme ľubovoľné hodnoty do polí, ktoré sú na to určené (viď. Obrázok 8). Ako ďalší krok pomocou tlačidla „Vložiť“ tieto hodnoty / údaje vložíme do technológie localStorage.

Obrázok 8 - Ukážka našej webovej stránky a technológiou localStorage v konzole, zdroj: Autor práce



Následne, ak sme pripojení k internetu, tak sa tieto údaje automaticky uložia do MySQL databázy do vopred pred vytvorenej databázy s názvom „bakalarka“, v ktorej je tabuľka taktiež s názvom „bakalarka“. V tejto tabuľke máme vytvorené aj dva riadky, ktoré máme pomenované „kluc“ a „hodnota“, do ktorých sa naše vložené údaje z našej webovej stránky zapíšu (viď obrázok 9).

Obrázok 9 - Ukážka MySQL databázy, zdroj: Autor práce



Z pohľadu servera to nie je nič zaujímavé, keďže je tam iba vyobrazené to, ako sme sa museli dostať do toho správneho priečinku, kde sa nachádza náš javascript súbor so serverom. Tento priečinok sa volal mysql-test a presunuli sme sa tam pomocou príkazu cd (change directory). Ak už sme sa v ňom nachádzali, tak iba jednoducho pomocou jedného príkazu sme spustili náš server, a to príkazom node server.js a následne sme tým dostali odpoveď na akom porte sa nachádzame a či spojenie bolo úspešné. Ak by spojenie nebolo úspešné, konzola nám vypíše chybu a popis čo je chybné s našim pripojením alebo prípadne kódom (viď. obrázok 10).

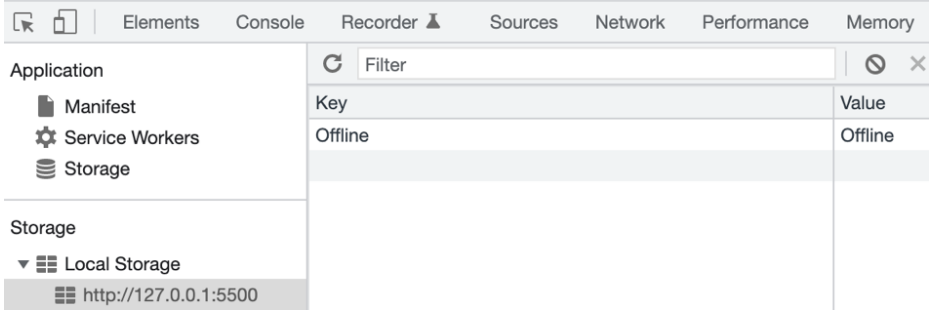
Obrázok 10 - Pohľad zo strany servera, zdroj: Autor práce

```
kmecadam@MBPuzivtelaAdam mysql-test % node server.js
Example app listening on port 3000!
Connected!
{ array_items: [ { key: 'Jozef', value: 'Mrkva' } ] }
```

3.5.3 Správanie webovej stránky ak je offline

Správanie našej webovej aplikácie v režime offline je nasledovné. Údaje zapísané do polí na písanie sa uložia do nášho webového úložiska (v našom prípade localStorage), kde sú uložené do tej doby, do kedy ich napríklad my manuálne nevymažeme alebo sa nepripojíme naspäť k internetu (viď. obrázok 11). Ak sa opätovne pripojíme na internet a refreshneme stránku, tak tieto údaje nám znova z tohto úložiska zmiznú a presunú sa na MySQL databázu (viď. obrázok 12).

Obrázok 11 – Vložené údaje v offline režime, zdroj: Autor práce



Obrázok 12 - Ukážka údajov zapísaných v databáze po pripojení na internet, zdroj: Autor práce

3.6 Diskusia

Pri tejto záverečnej práci sa nám podarilo ukázať ako vytvoríme jednoduchý webový formulár, vytvorenie servera a databázy. Samozrejme táto práca má svoje nedostatky. Jedným z nich je úplne banalita toho, že webová stránka vyzerá strašne jednoducho. Tento nedostatok by sa dal kľudne a jednoducho vyriešiť rôznymi úpravami, a to v novom súbore CSS, kde by sme si upravili ako vyzerá fieldset, tlačidlo na ktoré klikáme pri odoslaní údajov, pole do ktorého vpisujeme údaje a celkovo úprava samotnej stránky, kde by sme vedeli všetko dať na stred obrazovky a nie to nechať, tak ako máme vľavo hore. Ale keďže naše zadanie záverečnej práce to nevyžadovalo a nebolo to potrebné, tak sme tento nedostatok ani neriešili. Po ďalšie si tu môžeme spomenúť fakt, že pri samotnom posielaní údajov v offline režime, kedy nie sme pripojení na internet a po opätovnom pripojení na internet sa naše údaje z localStorage presunú do databázy len vtedy, ak refreshneme stránku. Tento nedostatok by sa dal vyriešiť viacerými spôsobmi, ale jedným z nich by mohol byť, že by sme do kódu pridali príkaz window.addEventListener().

Medzi naše prínosy určite patrí, že sme si objasnili v tejto práci dôležitosť offline podpory webových stránok a priblížili sme čitateľom bližšie túto problematiku.Táto tvorba webovej stránky by sa dala aj naďalej vyvíjať a dala by sa z toho urobiť aj nejaká funkčná a užitočná webová aplikácia pre niekoho, kto si potrebuje napríklad niečo zapísať (napríklad nejakú pripomienku) a táto stránka by mu to dokázala urobiť aj v offline režime a mnoho ďalších.

Záver

Touto prácou sme sa snažili vyzdvihnúť poznanie offline podpory, ktorú využíva široká verejnosť pri prehliadaní webových stránok. Niektorí užívatelia internetu a špecialisti webových prehliadačov majú o danej problematike základné vedomosti, avšak zvyšok bežnej populácie nepozná termín offline webová stránka.

Záverečná práca je zložená z teoretickej a praktickej časti. Prvá kapitola vysvetľovala pojmy súvisiace s našou prácou a tými sú offline podpora a technológie využívané pri riešení tejto offline podpory. V druhej kapitole sme našu pozornosť upriamili na vymedzenie cieľu, metodiky, ktorou sme postupovali pri skonštruovaní nášho projektu a metódy, ktoré sme využili. Záverečná časť je venovaná samotnému projektu, v ktorom popisujeme získané výsledky. Táto problematika ako sme už spomínali nie je veľmi populárna a známa medzi bežnými užívateľmi internetu, ale naša práca mala za úlohu zdôrazniť a poukázať na túto problematiku a venovať jej pozornosť. Rozoberali sme v nej najbežnejšie a najpoužívanejšie úložiská na strane používateľa a vysvetlili sme v skratke akú históriu mal tento typ úložiska. Tieto úložiská sme porovnávali medzi sebou pomocou nami zvolených kritérií a výsledkom tohto porovnávania bol výber tej najlepšej pre náš malý webový formulár. Túto technológiu sme použili na našej webovej stránke, ktorú sme si vytvorili úplne jednoducho a ktorú sme prepojili so serverom a databázou. Nakoniec sme si predviedli ako to funguje a čo sa nám pri tejto práci podarilo a čo zas nie.

Pri tejto problematike bolo veľmi podstatné, aby sa jej venovalo viac času a aby dostala oveľa viac priestoru a prostriedkov čo sme aj dokázali v tejto práci. Veríme, že táto práca pomôže bežnej populácií jednoducho pochopiť čo sú offline webové stránky a ako jednoducho ich vytvoriť a naprogramovať.

Zoznam použitej literatúry

[1] W3SCHOOLS. HTML Web Storage API. [cit. 2021-11-12] Dostupné na internete: <https://www.w3schools.com/html/html5_webstorage.asp>

[2] ZAKAS, Nicholas. JavaScript for Web Developers 2nd Edition. Wiley Publishing Inc., 2014. ISBN 978-0-470-22780-0

[3] HALES, Wesley. HTML5 and JavaScript Web Apps. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2013. ISBN 978-1-449-32051-5

[4] Editor WARF, Barney. The SAGE Encyclopedia of THE INTERNET. SAGE Publications, Ltd., 2018. ISBN 978-1-4739-2661-5

[5] MDN. Window.localStorage. [cit. 2021-11-12] Dostupné na internete: <https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage>

[6] MDN. Window.sessionStorage. [cit. 2021-11-12] <https://developer.mozilla.org/en-US/docs/Web/API/Window/sessionStorage>

[7] GAMOV, Viktor, Anatole TARTOVSKY, Victor RASPUTNIS a Yakov FAIN. Enterprise Web Development. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2014. ISBN 978-1-449-35681-1

[8] FOWLER, Adam. NoSQL for DUMMIES. John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030-5774, 2015. ISBN 978-1-118-90574-6

[9] OWENS, Mike a Grant ALLEN. The Definitive Guide to SQLite. Paul Manning, 2010. ISBN 978-1-4302-3225-4

[10] KREIBICH, A. Jay. Using SQLite. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, 2010. ISBN 978-0-596-52118-9

[11] ALLEN, N. Grove. WebSQL: An Interactive Web Tool for Teaching Structured Query Language. [cit. 2022-05-01] <https://aisel.aisnet.org/cgi/viewcontent.cgi?article=1874&context=amcis2000>

[12] ATENCIO, Luis, Paul, P. DANIELS. RxJS in Action. Manning Publications Co., 2017. ISBN 9781617293412

[13] PERCIVAL, John. HTML5 Advertising. Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013, 2013. ISBN 978-1-4302-4602-2

[14] FLANAGAN, David. JavaScript The Definitive Guide. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, 2011. ISBN 978-0-596-80552-4

[15] MDN. HTML: HyperText Markup Language. [cit. 2022-04-05] Dostupné na internete: <https://developer.mozilla.org/en-US/docs/Web/HTML>

[16] NIXON, Robin. Learning PHP, MySQL & JavaScript with JQUERY, CSS & HTML5. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, 2018. ISBN 978-1-491-97891-7

[17] HOSSAIN, Monsur. CORS IN ACTION Creating and consuming cross-origin APIs. Manning Publications Co., 2015. ISBN 9781617291821

[18] RAJLICH, Nathan, T. J. HOLOWAYCHUK, Marc HARTER, Mike CANTELON. Node.js IN ACTION. Manning Publications Co., 2014. ISBN 9781617290572