VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

SYSTÉM PRO ŘÍZENÍ VÝROBY VE FORMĚ PWA

A PRODUCTION MANAGEMENT SYSTEM IN THE FORM OF A PWA

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. DAMIÁN GLADIŠ**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. VLADIMÍR BARTÍK, Ph.D.**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání diplomové práce

148329

Ústav: Ústav informačních systémů (UIFS)
Student: **Gladiš Damián, Bc.**
Program: Informační technologie a umělá inteligence
Specializace: Informační systémy a databáze

Název: **Systém pro řízení výroby ve formě PWA**
Kategorie: Informační systémy
Akademický rok: 2022/23

Zadání:

1. Prostudujte principy tvorby Progressive Web Application (PWA).
2. Analyzujte požadavky na systém pro řízení výroby ve strojírenské firmě zahrnující evidenci chyb zaměstnanců, včetně prioritizace jednotlivých zakázek na základě různých kritérií. Aplikace by měla být funkční na různých typech zařízení.
3. Navrhněte aplikaci dle požadavků, použijte k tomu vhodné modelovací prostředky.
4. Po konzultaci s vedoucím implementujte systém dle požadavků a otestujte funkčnost na vhodném vzorku dat.
5. Zhodnoťte dosažené výsledky a další možnosti pokračování tohoto projektu.

Literatura:

- Ater, T.: Building Progressive Web Apps: Bringing the Power of Native to the Browser (1st Edition). O'Reilly Media, 2017. ISBN 978-1491961650.
- Hume, D. A.: Progressive Web Apps (1st. ed.). Manning Publications Co., USA, 2017. ISBN 978-1617294587.
- Love, C.: Progressive web application development by example: develop fast, reliable and engaging user and experiences with progressive web applications. Packt Publishing Ltd., 2018. ISBN: 978-1787125421.
- Hajian, M.: Progressive Web Apps with Angular: Create Responsive, Fast and Reliable PWAs Using Angular. Apress, 2019. ISBN: 978-1484244494

Při obhajobě semestrální části projektu je požadováno:
Body 1 až 3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Bartík Vladimír, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1.11.2022
Termín pro odevzdání: 17.5.2023
Datum schválení: 24.10.2022

Abstrakt

Táto diplomová práca sa zaoberá rozborom, návrhom a implementáciou systému pre riadenie výroby vo forme PWA. Diplomová práca popisuje návrh tejto aplikácie so zameraním sa na vykonávanie a prioritizovanie výrobných zákaziek a na evidenciu dokumentácie práce a chýb zamestnancov. V kapitolách je popísaný rozbor vyššie uvedených požiadaviek, na základe ktorého bol spracovaný návrh a implementácia tohto systému.

Abstract

This thesis deals with the analysis, design and implementation of a production management system in the form of PWA. The diploma thesis describes the design of this application with a focus on the execution and prioritization of production orders and the record of work documentation and employee errors. The chapters describe the analysis of the above-mentioned requirements, on the basis of which the design and implementation of this system was processed.

Kľúčové slová

Systém na riadenie výroby, PWA, strojárenská firma, prioritizovanie výrobných zákazok, React, Service Worker

Keywords

Production management system, PWA, engineering company, prioritization of production orders, React, Service Worker

Citácia

GLADIŠ, Damián. *Systém pro řízení výroby ve formě PWA*. Brno, 2023. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vladimír Bartík, Ph.D.

Systém pro řízení výroby ve formě PWA

Prehlásenie

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne pod vedením pána Ing. Vladimíra Bartíka, PhD. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Damián Gladiš
17. mája 2023

Poďakovanie

Týmto by som sa chcel poďakovať môjmu vedúcemu práce Ing. Vladimírovi Bartíkovi, PhD. za vedenie v mojej diplomovej práci, cenné rady a všetkú pomoc. Taktiež by som chcel poďakovať Ing. Štefanovi Čabrovi a Ing. Jaroslavovi Fábrymu odbornú konzultáciu pri návrhu a implementácii.

Obsah

Kapitola 1

Úvod

Mobilné zariadenia sa každým rokom čoraz viac vyvíjajú a tak podniky a firmy ich čoraz viac a viac používajú ako primárny prístup na internet. Tak vzniká nový prístup k tvorbe webstránok aj na úrovni mobilných zariadení. Na webstránke si návštevník vyhľadá konkrétne informácie danej služby či programu, no na ich doručenie je už vytvorená samostatná natívna aplikácia.

Progresívne webové aplikácie prepájajú vlastnosti natívnych aplikácií, ktorými sú funkčnosť v offline režime, náhly prístup a vlastnosti webových aplikácií, ktorými sú odkazovanie pomocou URL, menšia pamäťová náročnosť či vyhnutie sa obchodom aplikácií rôznych operačných systémov.

Hlavným cieľom mojej diplomovej práce je vytvorenie systému pre riadenie výroby v strojárenskej firme za pomoci technológie PWA, ktorá bude uľahčovať komunikáciu medzi vedúcim pracovníkom a jeho podriadenými, a následne tak časovo urýchli jednotlivé fázy výroby v danej firme. Webová aplikácia bude aj evidovať chyby pri výrobe, či už interné alebo externé, ktoré poskytnú firme spätnú väzbu, na základe ktorej firma podnikne dané opatrenia. Výrobné zákazky budú zoradené podľa priority, ktorá bude daná vedúcim zamestnancom alebo dátumom dodania.

Kapitola 2 obsahuje stručný prehľad existujúcich aplikácií, ich výhody a nevýhody, a zdôrazňuje vlastnosti, ktoré by mala spĺňať vyvíjaná progresívna webová aplikácia.

Kapitola 3 obsahuje popis vývoja a vlastností PWA aplikácie pomocou viacerých moderných technológií, jej stratégie ukladania dát a zhodnotenie zvolenej technológie k implementácii. Okrem technológií pre klientsku stranu tiež obsahuje aj technológie na vývoj serverovej strany aplikácie.

Kapitola 4 obsahuje problematiku dokumentácie a riadenia procesu výroby v strojárenskej firme, jeho štruktúre, popis UML diagramu, ktorý vysvetľuje aké prípady použitia majú jednotlivý používatelia systému.

Kapitola 5 obsahuje návrh progresívnej webovej aplikácie, požiadavky, ktoré musí spĺňať, ako bude vyzerať vstup pre webovú aplikáciu, ako bude vyzerať jej navigačné menu, ako sa budú prioritizovať výrobné zákazky, ako bude vyzerať zadávanie interných a externých chýb,zadávanie problémov pri jednotlivých zákazkách. Obsahuje aj architektúru progresívnej webovej aplikácie.

V kapitole 6 je popísaná implementácia systému progresívnej webovej aplikácie, ktorej časti sú implementácia klientskej a serverovej strany, offline režimu a push notifikácií. V rámci implementácie klientskej strany sú popísané aj jednotlivé sekcie, v ktorých sa používateľ pohybuje.

Napokon, v kapitole 7 je zhrnuté testovanie tejto PWA aplikácie. Obsahuje testovanie na požiadavky systému, ktoré boli zadané na začiatku a zhodnotenú funkčnosť aplikácie. V rámci sekcií kapitoly 7 je popísané testovanie pracovníkom výroby a vedúcim pracovníkom výroby.

Kapitola 2

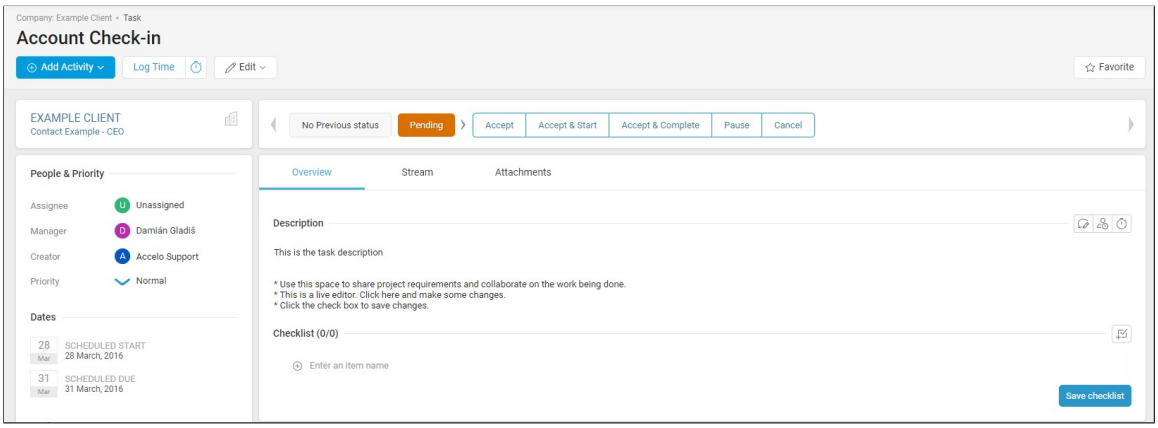
Prehľad existujúcich aplikácií

V tejto kapitole sú rozobraté existujúce aplikácie na riadenie výroby v podniku, ich vlastnosti a služby. Týchto aplikácií existuje strašne veľa, s veľmi podobnými vlastnosťami, tak sú rozobrané tie, ktoré sa od seba líšia viac. Cieľom tohto prieskumu je zistiť vlastnosti týchto aplikácií, uviesť ich výhody a nevýhody, a určiť tie vlastnosti, ktoré by mala zahrňovať aj samotná vyvíjaná aplikácia.

2.1 Accelo

Accelo poskytuje nástroje a zdroje na zobrazenie a správu projektov, plánovania, cenových ponúk, fakturácie a pod., a to všetko v rámci jednej aplikácie. Snaží sa ušetriť čas prácou v danej firme a rozvíjať služby danej firmy.

Táto aplikácia neobsahuje neplatenú verziu a každý jej používateľ musí zaplatiť mesačne približne 38€.



Obr. 2.1: Ukážka programu Accelo

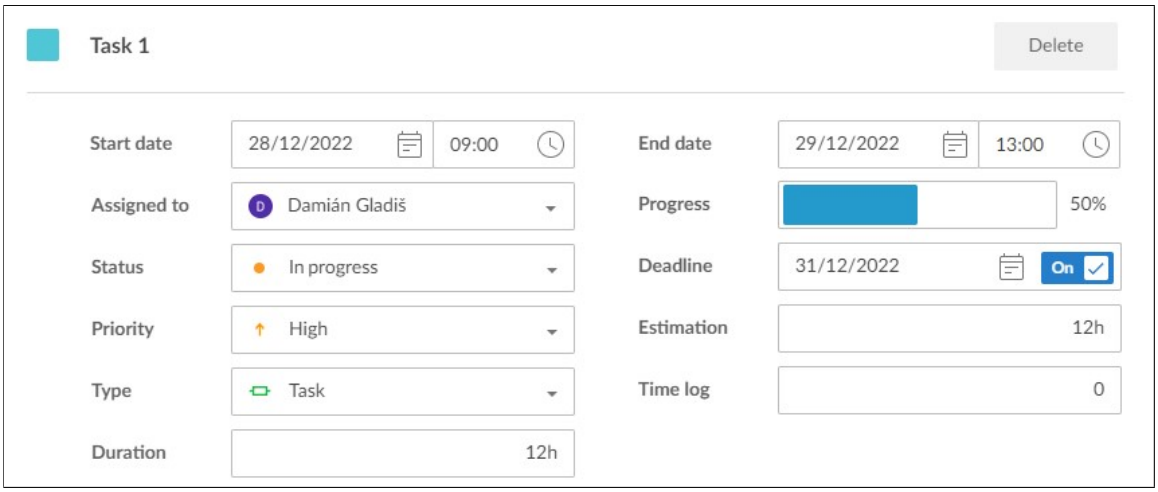
Výhodou tejto webovej aplikácie je jej prehľadnosť, poskytuje spravovanie viacerých spoločností naraz a obsahuje všetko, čo väčšia firma potrebuje k svojej prevádzke a aby upevnila svoju pozíciu na trhu. Zaoberá sa hlavne ziskom a pracuje na dlhodobom vzťahu s ich klientmi. Ďalšou výhodou je, že je kompatibilná na skoro väčšine operačných systémoch. Tiež, táto aplikácia je schopná vypočítať časový a finančný rozsah prípadne spôsobenej škody, ak na úlohe pracovali viacerí zamestnanci.

Nevýhodou je vnútro-podniková komunikácia, kde si musí vedúci danej úlohy a jeho vykonávateľ písať hromadu emailov, a nevedia to riešiť priamo v kalendári, kde je daná úloha umiestnená. Ďalšou nevýhodou je, že nemá offline režim, a tým pádom užívateľ nie je schopný hneď zadať informácie o prebiehajúcej úlohe, aj keď nie je momentálne pripojený na internet.

2.2 GanttPRO

GanttPRO je online softvér Ganttovho diagramu, ktorý umožňuje efektívne vytvoriť a spravovať komplexnejšie projektové plány, sledovať priebeh týchto projektov a komunikovať s ostatnými členmi daného projektu.

Táto aplikácia tiež neobsahuje neplatenú verziu a každý jej používateľ musí zaplatiť mesačne približne 13€.



Obr. 2.2: Ukážka programu GanttPRO

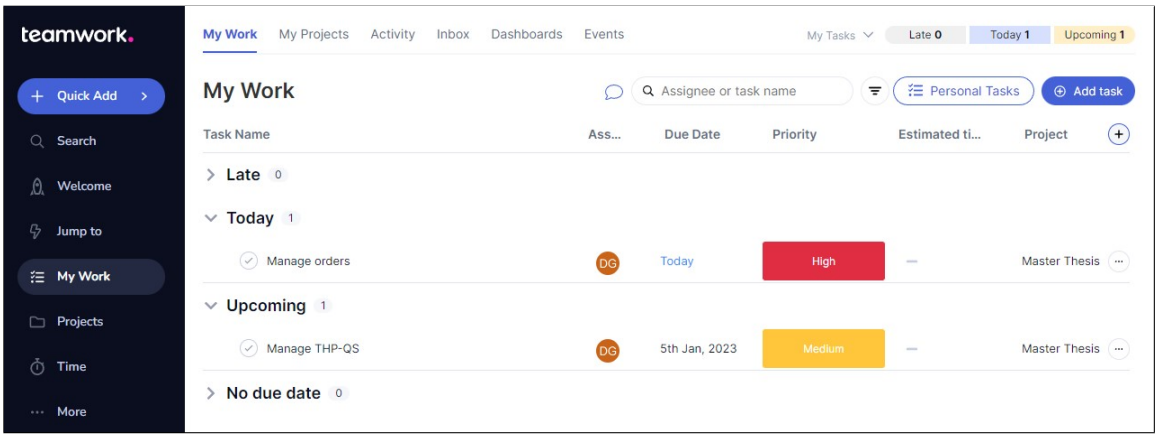
Výhodou pri tejto webovej aplikácii je možnosť nastavenia priblíženia časového zobrazenia úloh v kalendári a lepší prehľad jednotlivých podúloh hlavnej úlohy vďaka zobrazeniu na Ganttovom diagrame. Rovnako táto aplikácia sa nachádza aj v podobe pre mobilné zariadenia. Ďalšou výhodou je selekcia notifikácií, ktoré bude užívateľ dostávať.

Nevýhodou je vnútro-podniková komunikácia, kde je možné vzniknuté problémy pridať len do popisku, bez zaslania notifikácie alebo písaním emailov prostredníctvom tretej strany. Táto aplikácia nepodporuje fungovanie v offline režime, je nutné pripojenie na internet. Nepodporuje ani zaznamenávanie finančného ohodnotenia jednotlivých úloh, bez čoho nie je možné vypočítať prípadné vzniknuté škody.

2.3 Teamwork

Teamwork je popredná svetová platforma projektového manažmentu, ktorá je navrhnutá pre spoločnosti, ktoré chcú vytvárať, spravovať a sledovať rôzne komplexné projekty. Ponúka rôzne formy zobrazenia úlohy, či už v usporiadanom zozname, vo forme tabuli alebo vo forme dosiek, ktoré predstavujú jednotlivé fázy projektu.

Táto aplikácia tiež neobsahuje neplatenú verziu, používateľ si musí zaplatiť približne 18€ za mesiac.



Obr. 2.3: Ukážka programu Teamwork

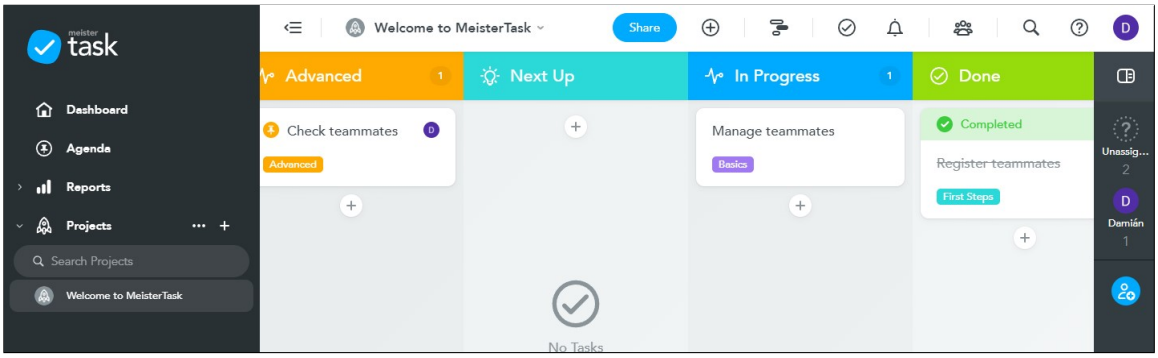
Táto aplikácia ponúka vo forme výhod prehľadnosť, zobrazenie úloh v rôznych časových mierkach, nezávislosť na platforme, finančné ohodnotenie úloh či zamestnancov a nastavenie notifikácií podľa užívateľských požiadavok.

Medzi nevýhody patrí neschopnosť pridania projektov a úloh z iných súborov, je možné pridať len používateľsjé profily alebo kontakty. Rovnako, ako pri predošlej aplikácií, nie je zabezpečená podpora pre prácu a nahrávanie informácií v offline režime.

2.4 MeisterTask

MeisterTask je intuitívny multiplatformový nástroj, ktorý zlepšuje štandard spolupráce na tímových projektoch. Užívateľské prostredie je navrhnuté pomocou dosiek, ktoré sú vhodné aj na softvérové sprinty a iné typy odvetví v rámci tímových projektov.

Táto aplikácia obsahuje bezplatnú verziu, ktorú je možné vyskúšať, ale za štatistiky projektov a podobné funkcionality je nutné si priplatiť 8,25€ mesačne.



Obr. 2.4: Ukážka programu MeisterTask

Výhodu je možnosť pridania konverzácie alebo aktivity v rámci danej úlohy, a je možnosť aj pridania prílohy k danej úlohe. Medzi ďalšie výhody patrí nastavenie priority jednotli-

vým úlohám, prehľadnosť, platformová nezávislosť a načítanie projektov z iných aplikácií a súborov.

Medzi nevýhody patrí to, že neobsahuje niektoré vlastnosti PWA (práca v offline režime,...) a nie je možné si prechádzať jednotlivé úlohy v projekte v rôznych časových zobrazení v základnom ani v pro verzii. Rovnako ako pri predošlej aplikácii, nepodporuje zaznamenávanie finančného ohodnotenia jednotlivých úloh.

2.5 Zhodnotenie prehľadu aplikácií

Aplikácie majú spoločne viacero znakov. Zobrazujú jednotlivé úlohy, kto ich vykonáva a kto je vedúcim projektu, časový rozsah danej úlohy a jej podúloh. Skoro všetky aplikácie nie sú bezplatné, a tak každý jeden ich používateľ si musí zaplatiť, ak ich chce používať.

Ich výhodami je prehľadnosť a jednoduchosť zobrazenia jednotlivých projektov. Spravovanie týchto projektov je nenáročné a pochopiteľné bežnému užívateľovi.

Medzi nevýhody patrí to, že viaceré aplikácie nepodporujú offline režim, ktorý je v rámci strojárenskej výroby podstatný, ak zamestnanec momentálne nie je pripojený k internetu, no potrebuje zdokumentovať práve vykonávanú úlohu bez toho, aby sa k tomu musel neskôr vracať.

Po tejto rešerši je navrhnutá aplikácia tak, aby vzniknuté problémy pri jednotlivých úlohách upozornili osoby týkajúce sa danej úlohy, a tak aby neviazla komunikácia medzi vedúcim projektu a ich vykonávateľmi. Aplikácia zahrňuje aj dokumentáciu jednotlivých krokov v procese výroby, a aj prípadne vzniknuté chyby/škody jednotlivých zamestnancov.

Kapitola 3

Teoretický rozbor témy

V tejto kapitole sú zahrnuté informácie o technológiách, ktoré je nutné analyzovať ešte pred návrhom a následnou implementáciou aplikácie. Vysvetľuje aký typ aplikácie je PWA a z čoho pozostáva.

3.1 Progresívne webové aplikácie

PWA sú v jednoduchosti vytvorené pomocou známych technológií - HTML, CSS a Javascript, no ponúkajú vyššiu úroveň pre ich používateľa. Poukazujú na súbor nazývaný *Manifest*, ktorý obsahuje dôležité informácie o obsahu webovej aplikácie, vrátane jej vzhľadu, pozadia, farieb, ikony, predvolenej orientácii a pod. Používajú aj dôležitú funkciu, nazývanú *Service Workers*, ktorá zohráva úlohu vo využívaní sieťových požiadavkách a tak dokážeme vytvárať lepšie webové aplikácie. Tieto webové aplikácie je možné „uložiť“ na domovskej obrazovke vo vašom zariadení a budú sa tváriť rovnako ako natívne aplikácie, takže prístup k webovej aplikácii je zabezpečený jednoducho stlačením na jej odkaze[20].

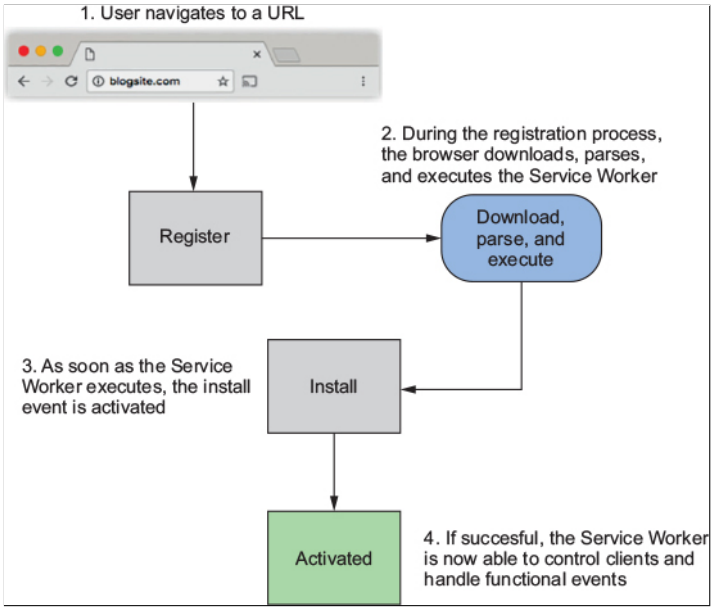
Tieto aplikácie by mali poskytovať možnosť pracovať aj v „offline“ režime. Práve *Service Workers* ukladajú časti webovej stránky do vyrovnávacej pamäti, aby boli neskôr k dispozícii, ak zariadenie nebude pripojené k internetovej sieti. Vytvorenie PWA vám umožní osloviť širšie publikum návštevníkov, pretože oproti natívnym aplikáciám sú platformovo nezávislé a dokážete ich pustiť na každom zariadení, v ktorom je webový prehliadač[9].

3.1.1 Service Workers

Service Workers zohrávajú významnú úlohu v progresívnych webových aplikáciach, lebo sú zodpovední za ukladanie do vyrovnávacej pamäti v offline režime, PUSH notifikácie, synchronizácia na pozadí a pod. Majú svoj životný cyklus, ktorý je znázornený na obrázku 3.1 sa skladá z 3 fáz:

- Registrácia
- Inštalácia
- Aktivácia

V nasledujúcich podsekciách sa rozoberie každá z týchto fáz[4].



Obr. 3.1: Životný cyklus Service Workers [9]

Registrácia

Service Worker, narozdiel od bežného Javascriptu, beží na pozadí, mimo hlavného vlákna, kde beží používateľské rozhranie používateľa. Ešte pred jeho prvotným použitím ho musíme zaregistrovať. Na overenie, či webový prehliadač podporuje Service Worker, môžeme použiť kód uvedený nižšie[9]:

```
<script>
// Register the service worker
if ('serviceWorker' in navigator) {
     navigator.serviceWorker.register('/sw.js')
        .then(function(registration) {
    // Registration was successful
    console.log('ServiceWorker registration successful with scope: ',
    registration.scope);
}).catch(function(err) {
    // Registration failed :(
    console.log('ServiceWorker registration failed: ', err);
    });
}
</script>
```

Ak objekt „navigator“ obsahuje vlastnosť „serviceWorker“, teda jeho podporu, zaregistruje súbor Service Worker s názvom „sw.js“ a vypíše rozsah platnosti jeho registrácie. Ak sa to nepodarí,vypíše chybu do konzoly prehliadača [9].

Inštalácia

Ak už máme Service Worker zaregistrovaný, musíme ho ešte nainštalovať. V tomto kroku sa skript stiahne a webový prehliadač sa ho pokúsi nainštalovať. Inštalácia Service Workera prebehne iba v jednom z týchto dvoch prípadov:

- Service Worker ešte predtým nebol nainštalovaný
- Skript Service Worker bol zmenený

Ak máme Service Worker nainštalovaný, spustí sa udalosť „install“. Na túto udalosť sa môžeme zamerať a pri jej zachytení vykonať nejakú úlohu, napríklad uložiť statické dáta našej aplikácie.

Pomocou metódy „open()“ z rozhrania CacheAPI otvoríme prístup a pridáme potrebné URL adresy do vyrovnávacej pamäti. Musíme použiť metódu „event.waitUntil()“, ktorá zabezpečí aby inštalácia Service Workera nebola úspešná, ak zlyhá ukladanie do vyrovnávacej pamäti[5].

Aktivácia

Po úspešnej inštalácii, Service Worker je v nainštalovanom stave a čaká, aby prevzal kontrolu nad stránkou, ktorú ešte spravuje aktuálny Service Worker. Následne prejde do stavu „activate“ v nejakom z týchto prípadov:

- Žiadny iný Service Worker už nie je aktívny
- Ak sa pri obsluhe udalosti „install“ zavolá metóda „self.skipWaiting()“
- Ak používateľ obnoví webovú aplikáciu

Metóda skipWaiting() môže vyzerať nasledovne:

```
    self.addEventListener('install', function (event) {
    self.skipWaiting();

    event.waitUntil(
           // static assets caching
      );
});
```

Udalosť „activate“ bude spustená, ak bude Service Worker aktívny. Aj túto udalosť je možné zaznamenať a pri jej zachytení vykonať zadanú úlohu pre webovú aplikáciu. Aktívny Service Worker teraz dokáže spracovávať udalosti, ako je načítanie, odosielanie a synchronizácia[4].

3.1.2 Stratégie ukladania dát do vyrovnávacej pamäte

Ako už bolo vyššie spomenuté, Service Worker, ktorý beží na pozadí webového prehliadača, spravuje aj dáta a obsluhuje viacero funkcií. Zachytávanie a ukladanie týchto dát je jednou z týchto funkcií. Service Worker ponúka prostredie natívnej aplikácie a tým pádom môže poskytovať dáta alebo vykonať určité funkcie pri zlom internetovom pripojení alebo v úplnom offline režime. Poznáme 5 rôznych stratégií, ktorými môže Service Worker pracovať:

- **Stale-While-Revalidate** - kontroluje odozvu vo vyrovnávacej pamäti. Ak sa dáta nachádzajú vo vyrovnávacej pamäti, tak sa odpoveď odošle práve odtiaľ a dáta v nej sa aktualizujú. Ak nie, Service Worker ju načíta z odpovede servera a uloží do vyrovnávacej pamäti. Táto stratégia dobre pracuje nie len s načítavaním dát zo strany servera, ale aj opačným smerom, ukladaním dát na server.
- **Cache first, then Network** - kontroluje najprv vyrovnávajúcu pamäť, ak tam je nejaká predošlá požiadavka už uložená, zoberie ju a vráti ako odpoveď. Ak nie je, pošle požiadavku na server, odpoveď vráti a uloží do vyrovnávajúcej pamäti pre opätovné použitie.
- **Network first, then Cache** - je presným opakom predošlej stratégie. Stále sa najprv dotazuje na server, ak uspeje, odpoveď vráti a uloží ju do vyrovnávajúcej pamäti. Ak nie je možné sa dotazovať na server, odpoveď sa vráti z vyrovnávajúcej pamäti.
- **Cache only** - posiela požiadavky iba do vyrovnávajúcej pamäti, nedotazuje sa na server
- **Network only** - posiela požiadavky iba na server, nedotazuje sa do vyrovnávajúcej pamäti[3].

3.1.3 Manifest

Súbor Manifestu môže mať valiteľný názov, no bežne sa tento súbor nazýva „manifest.json“ a je umiestnený v koreňovom adresári webovej aplikácie. Jeho prípona podľa špecifikácie by mala byť „.webmanifest“, no webové prehliadače poskytujú podporu aj pre JSON súbory, čo zjednodušuje prácu vývojárov. Príklad takéhoto Manifestu môže vyzerať takto[15]:

```
{
  "name": "Overleaf.io",
  "short_name": "Overleaf",
  "theme_color": "#d3d3d3",
  "background_color": "#69d369",
  "display": "browser",
  "Scope": "/",
  "start_url": "/",
  "icons": [
    {
      "src": "images/icons/icon-128x128.png",
      "sizes": "128x128",
      "type": "image/png"
    }
  ],
  "splash_pages": null
}
```

Ak už je Manifest súbor vytvorený a správne uložený, je možné ho pridať do sekcii hlavičky našej webovej aplikácie napríklad takto[1]:

```
<link rel="manifest" href="/path/to/manifest.json">
```

3.1.4 Výhody PWA

Prvým bodom je určite správa aplikácií. Natívne aplikácie si užívatelia vedia nainštalovať cez obchody jednotlivých operačných systémov, napr. Google Play alebo App Store. Tieto obchody požadujú, aby uverejnené aplikácie splňovali rôzne požiadavky a obmedzenia, ktoré sa môžu líšiť a každý takýto obchod ma tieto podmienky odlišné vzhľadom na cieľ, ktorý chcú ponúknuť používateľovi. Taktiež, proces zverejnenia môže byť zdĺhavý, preto časovo rýchlejšie šírenie PWA aplikácií mimo týchto obchodov za pomoci URL je oveľa lepšou voľbou.

Ďalšou výhodou je multiplatformovosť. Na trhu je veľa rôznych značiek, každá s rôznym operačným systémom a vývojovými nástrojmi. Rozdielne vývojové procesy môžu ovplyvniť funkcionalitu aj cenu týchto aplikácií. Pre aplikácie PWA to nie je žiadny problém, keďže na ich prevádzkovanie stačí moderný webový prehliadač s podporou HTML, CSS a JavaScriptu.

3.1.5 Nevýhody PWA

Napriek veľmi dobrým výhodam PWA aplikácie, aj oni majú pár obmedzení a nedostatkov. Tieto aplikácie sú pomerné nové a ešte stále sa vyvíjajú, hlavne ich funkcionalita. PWA aplikácie nam poskytnú len toľko, čo nám dovolí webový prehliadač. Medzi hlavné funkčné opatrenia webového prehliadaču Google Chrome patria napríklad prístup ku kamere alebo mikrofónu, prístup ku kontaktom alebo SMS správam. Keďže tieto aplikácie najviac závisia na webových prehliadačoch, užívateľské zariadenie musí podporovať ich novšie verzie.

3.2 IndexedDB

V rámci PWA aplikácie je možné pracovať aj offline, je potrebné tieto dáta niekam ukladať. Ak webová aplikácia odošle požiadavka cez sieť na stranu servera a používateľ momentálne nemá potrebné internetové pripojenie, ako odpoveď mú príde dáta z vyrovnávajúcej pamäti. IndexedDB je databáza typu NoSQL, resp. kľúč-hodnota, ktorá dokáže poskytnúť priestor pre ukladanie až do výšky 50% kapacity pevného disku. Rovnako chráni údaje, ktoré majú rovnaký pôvod, aby mohli byť len odpoveďou len pre tento pôvod. Podpora tejto databázy je zastúpená vo všetkých najpoužívanejších prehliadačoch a má byť čoskoro podporovaná aj v menej používaných.

Každá vytvorená databáza môže existovať iba v stanovenej verzii a v žiadnej inej. Operácie v takejto databáze prebiehajú formou transakcií, čo predstavuje spôsob interakcie s údajmi v databáze. Každá operácia musí byť uskutočnená v rámci danej transakcie.

Vytvorenie takejto databázy pozostáva z nasledujúcich krokov:

1. Otvoríme databázu pod daným názvom
2. V databáze vytvoríme objekt pre ukladanie
3. Transakcia sa spustí a požiada o vykonanie databázovej operácie, napr. uloženie alebo načítanie dát
4. Počká sa na dokončenie operácie podľa volanej udalosti z DOM - Document Object Model

5. Spracujú sa výsledky - je možné ich nájsť uložené v objekte, ktorý prislúcha požiadavke

Táto databáza je dostupná práve z Service Workera, ktorej dostupnosť si vieme overiť pomocou príkazu *self.IndexedDB*[21].

3.3 MVVM

Vzor MVVM k vzorom MVC a MVP prišiel ako alternatíva. Väčšina argumentov, ktoré podporujú MVVM, je založená na skutočnosti, že View a stav View v predchádzajúcich prístupoch (MVC/MVP) sú stále prepojené s Model do takej miery, že je ťažké ich otestovať individuálne. Toto prepojenie zasahuje do všeobecného princípu modulárneho programovania.

ViewModel vo vzore MVVM nahrádza Presenter a Controller. Povinnosti ViewModel a View sú teraz odlišné. Je bežné prezentovať vzor MVVM lineárnym spôsobom, ako to je znázornené na obrázku 3.2 . Dôvodom tohto zoskupenia je poukázanie na zmenu priradených úloh, ktoré sa vykonávajú v každej časti vzoru, a poukázať na tok informácií a údajov. Model zostáva prevažne rovnaký ako v dizajne MVP. Stále je zodpovedný za prístup k rôznym zdrojom údajov (napr. súborom, databázom alebo serverom). Vo všeobecnosti má model tendenciu byť veľmi tenký v implementácii MVVM.

Obr. 3.2: Vzor Model-View-ViewModel (MVVM) [10]

V MVVM sa väčšina kódu nachádza v ViewModel. Koncept modelu ViewModel spočíva v tom, že tento komponent predstavuje spôsob, akým sa od zobrazenia očakáva (stav zobrazenia) a od ktorého sa očakáva, že sa bude správať podľa interakcií používateľa (logika zobrazenia). Je to model pohľadu v tom zmysle, že popisuje súbor princípov a štruktúr, ktoré prezentujú špecifické údaje získané prostredníctvom Model. ViewModel zabezpečuje komunikáciu medzi View a Modelom odovzdávaním všetkých potrebných údajov z View do Modelu vo forme, ktorú model dokáže spracovať. Overenie sa vykonáva v komponente ViewModel. V tomto vzore komponenty pracujú v súpravách po dvoch. View si je vedomý ViewModelu, aktualizuje vlastnosti ViewModelu a sleduje všetky zmeny, ktoré sa v ňom vyskytnú. ViewModel si nie je vedomý existencie View. Podobne Model nepozná ViewModel (alebo samotný pohľad), ale je to iba ViewModel, ktorý má prístup k Modelu. ViewModel odovzdáva udalosti a údaje modelu tak, ako sú tlačené zobrazením vo formulároch, ktoré model dokáže interpretovať. ViewModel sleduje všetky zmeny vytvorené modelom a následne do View vloží všetky potrebné signály podľa View a obchodných pravidiel.

Jedným z kľúčových bodov je, že MVVM je veľmi flexibilný a vývojári ho môžu implementovať podľa viacerých návrhov. Táto flexibilita je jednou zo silných stránok tohto vzoru[10].

3.4 Vývojové rámce PWA

V tejto sekcii sú porovnané štruktúry, vlastnosti,výhody a nevýhody známejších vývojových rámcov progresívnych webových aplikácií a na konci je uvedené zdôvodnenie vybraného vývojového rámca.

3.4.1 React

React je knižnica pre JavaScript, typu open-source, ktorá je zložená z komponentov a používa sa najmä na vývoj front-endov. Pre lepšie pochopenie, JavaScript je programovací jazyk, používaný na vývoj webových aplikácií a webových stránok, ktorý im poskytuje lepšiu dynamickosť, škálovateľnosť a efektívnosť. Každá webová stránka, ktorá neobsahuje len statické dáta, ale aj napr. nejakú animáciu či video, pravdepodobne k tomu využíva Javascript.

Ak už máme predstavu o jazyku Javascript, tak si vysvetlíme čo je to JavaScript knižnica. Je to knižnica s vopred napísanými kódmi, v jazyku Javascript, ktorá pozostáva zo vzorov, funkcií a komponentov, ktoré hrajú významnú rolu pre vývj webovej stránky. Vývoj Reactu využíva funkcie JavaScriptu pre mnohé zo svojich vzorov, ako napríklad JSX, čo uľahčuje písanie a pridávanie HTML v Reacte. Primárnym cieľom Reactu je minimalizovať výskyt chýb v procese vývoja používateľského rozhrania.

React obsahuje JSX, rozšírenie syntaxe JavaScriptu, vďaka ktorému je mimoriadne jednoduché kódovať pomocou funkcií JavaScriptu a prvkov HTML vedľa seba[8].

Medzi výhody Reactu patrí:

- **Viazanosť údajov** - predstavuje jednoduchosť v tom, že každý je schopný nazrieť do zmien vykonaných v akýchkoľvek uložených dátach
- **Jednoduchý na učenie** - oproti Angular a Vue, je jednoduchší na pochopenie a urýchľuje tým vývojový čas hlavne väčších aplikácií.
- **Znovupoužiteľné komponenty** - vývojári po vytvorení danej komponenty ju môžu znovu použiť niekoľkokrát aj pre inú aplikáciu s rovnakou funkcionalitou. Ako predošlá výhoda, to značne skráti čas vývoja.
- **Deklaratívnosť** - zmeny v dátach je možné vykonať pomocou Reactu, čo zautomatizuje aktualizácie zmien v používateľskom rozhraní UI aplikácie. Pre zmeny nie je potrebné nič ďalšie vykonávať.
- **Intuitívnosť** - aj pre vývojára, ktorý začína s Reactom, je ľahké pochopeniť rozloženie používateľského rozhrania a ako s ním pracovať.
- **Optimalizácia SEO** - umožňuje vytvárať vizuálne prívetivé používateľské nastavenia SEO, ktoré majú podporu naprieč viacerými nástrojmi a prehliadačmi.
- **Podporné komponenty** - je ideálnym na naviazanie HTML s JavaScriptom. React predstavuje sprodstredkovateľa, ktorý obsahuje „Document Object Model“ a rozhodne ako sa v ňom dosiahnu požadované zmeny[7].

Oproti množstvu výhod má aj pár nevýhod:

- **React a JSX** - Ako je vyššie spomenuté, pre komunikáciu HTML a Javascriptu React využíva JSX. Pre niektorých môže byť toto rozšírenie mätúce a zavádzajúce. Je jedným z rozšírení, ktoré potrebujú čas na pochopenie.
- **Nesprávna dokumentácia** - React je stále meniaca sa technológia a nové knižnice, ako Reflux a Redux, urýchľujú proces vývoja aplikácií, no existuje možnosť, že výsledný produkt nie je zhodný s požadovaným návrhom, čo predstavuje pre vývojára ďalšiu námahu s integráciou nástrojov. Sami si musia napísať pokyny, čo a ako bolo v aplikácii aktualizované.
- **Rýchlosť vývoja** - s takým rýchlym vývojom je ťažké držať krok so všetkými aktualizáciami a novými pracovnými spôsobmi. je to výhodou aj nevýhodou, no z väčšej časti výhodou, keďže zlepšuje výkon a zvyšuje úroveň vyvíjanej aplikácie.
- **Úzke zameranie na UI** - pokrýva iba časti UI - používateľského rozhrania, takže k týmto častiam je potrebné pridať ďalšie nástroje a technológie aby tvorili kompletnú sadu[7].

3.4.2 Angular

Angular predstavuje vývojový rámec webových aplikácií a podobne ako React, je založený na jazyku TypeScript a je verejne dostupný. Bežne sa používa na vytváranie jednostránkových aplikácií - SPA[11].

V rámci vývoju progresívnych webových aplikácií poskytuje štruktúru, na ktorej vývojári môžu spoločne pracovať. Takto zostanú aj väčšie vyvíjané aplikácie nezastaralé a udržiavané[7].

Angular ponúka viacero výhod:

- **Jazyk Typescript** - priehľadný kód, škálovateľnosť a lepšie nástroje pre vývoj, ktoré väčšie podnikové aplikácie vyžadujú
- **Architektúra MVC** - izoluje používateľské rozhranie a logiku, poskytuje odpoveď používateľovi vytvorenú kontrolérom
- **Vylepšený dizajn** - v rámci väčších webových aplikácií je potrebné spravovať aj obrovské množstvo komponentov, pre ktoré Angular ponúka efektívny spôsob po začatí navrhovacieho procesu
- **Spravovanie závislostí a služieb** - využíva službu „Dependency Injection“, ktorá rozdeľuje pracovné zaťaženie medzi viacero dostupných služieb. Objekt, ktorý je závislý, nebude spravovaný klientskym servisom, ale uhlovým injektorom. Pri riešení PWA má injektor na starosti generovanie inštancií rôznych služieb.

Angular má rovnako aj viacero nevýhod:

- **Sofistikovaná a zdĺhavá fráza** - pri tak veľkej rozmanitosti tohto nástroja je možné, že niekedy to môže byť až zdrvujúce. Angular ponúka širokú paletu konceptov, materiálov, komponentov, modulov, a práve kvôli tomu mu je ťažšie porozumieť, ako pre React alebo Vue, ktoré majú len jeden komponent.
- **Ťažšie pochopiteľný** - začlenie sa do vývoju s použitím Angular je oveľa ťažšie aj pre vývojárov, ktorí už JavaScript poznajú a tiež hlavne aj preto, že je potrebné pokryť viacero nevyhnutných súčastí.

- **Obmedzené možnosti SEO** - nízka dostupnosť v rámci prehľadávania je významnou nevýhodou používania Angular. Rovnako je to aj hlavnou nevýhodou pri použití Angular v rámci vývoja PWA[7].

3.4.3 Vue

Vue jedným z ďalších vývojových rámcov pre vývoj progresívnych webových aplikácií. Oproti vyššie spomínaným rámcov, React a Angular, bolo Vue štruktúrované tak, aby jeho adaptácia bola čo najjednoduchšia. Tvorí ho základná knižnica, ktorá obsahuje zobrazovaciu vrstvu a tá komunikuje s ostatnými knižnicami alebo aplikáciami. No má aj spoločné vlastnosti ako React a Angular, vytvára používateľské rozhrania - UI pomocou komponentov, ktoré sa zmiešavajú dokopy[7].

Výhody pri použití Vue:

- **Je efektívny a zaberá málo miesta** - v režime „Lightweight“ má len 18 až 21Kb, vďaka čomu ho vývojári veľmi využívajú. Aj napriek malej veľkosť, je dostatočne rýchly, ľahký na načítanie a inštaláciu, čo oproti Angularu zvyšuje dosah SEO. Vďaka tomu sú optimalizácie takýchto programov výrazne kratšie.

- **Ľahšie pochopiteľný** - Má základnú štruktúru, vďaka čomu je krivka učenia menej strmá a je jednoduchšie vzniknuté chyby vyhľadať a opraviť.

- **Dobrý výkon** - pracuje s virtuálnym DOM a je ostražitejší, venuje väčšiu pozornosť vzniknutým chybám

- **Lahšia integrácia** - s podporou JavaScriptu je možné ho integrovať do akejkoľvek aplikácie, buď novej alebo už existujúcej. Podobne ako React, využíva šablónu a virtuálny DOM - Document Object Model.

- **Flexibilita** - pre spoločnosti, ktoré sa zaoberajú vývojom webu Vue napomáha vytvárať šablóny HTML a JavaScript, ktoré je možné prezerať vo webovom prehliadači.

- **Udržiavateľnosť** - Vue zjednodušuje prácu vývojárom, keď napomáha pri správe aktualizácií a inej údržbe aplikácií.

- **Výpočtové vlastnosti** - nevyžaduje žiadny ďalší závislý kód pri úpravach prvkov UI a následne vykonaných výpočtoch.

- **Obojsmerná väzba údajov a reaktivita** - pre vývoj webu je obojsmerná reaktívna väzba údajov veľmi dôležitá, pretože pripája JavaScript k virtuálnemu DOM a opačne a reaktívne viazanie údajov ich udržiava stále aktuálne a to aj bez ľudského zásahu.

- **Prívetivý pre jeho používateľov aj začiatočníkov** - ak poznajú Javascript, tak využívanie šablón a štýlov, je podobné ako pri iných rámcovy typu JavaScript. Medzi ďašie výhody patrí ponuka a rozsiahlosť dokumentácie VUE, v ktorej vývojár nájde popis k všetkým dôležitým údajom, od inštalácie po škálovateľnosť danej aplikácie.

  Začiatočník prevažne začína učením sa základov 3 dôležitých jazykov v oblasti webových aplikácií: HTML, CSS a JavaScript. JSX je naviazané na HTML a bežné pre použitie.

Aj Vue má svoje nevýhody:

- **Menšia komunita** - je ešte v rannom a rýchlom štádiu vývoja a je zatiaľ menej používanejší ako React alebo Angular. Pri tak rýchlom vývoji ak niekde nájdete nie úplne dávno pridané informácie, je možné, že už budú zastaralé. Aj hľadanie riešenia pre niektoré problémy môže trvať dlhšie ako len sa pozrieť na známe fóra. Zatiaľ neobsahuje ani dostatok knižníc a rozšírení.
- **Viac ako flexibilné** - ak sa pracuje na väčšom projekte, kde je širší tím vývojárov, tak prílišná sloboda môže byť závažným problémom, ktorá môže viesť k nekonzistentnosti, oneskoreniu projektu a následnom zvýšení výdajov na projekt. Ako PWA vývojový rámec nemá takú podporu pre moduly, komponenty a iné rámce tohto typu, čo je dosť veľkou nevýhodou Vue.

3.4.4 Zhodnotenie

Pre aplikáciu, ktorá bude obsahovať viacero modulov na načítanie čiarového kódu, na správu dát ktoré sa posielajú a prijímajú zo servera, a bude to webová aplikácia, kde bude možné sa prepínať medzi komponentami, bol zvolený vývojový rámec React. Je vhodnejší pre väčšie webové aplikácie s častejšie premenlivými údaje. Tiež disponuje širokou škálou balíčkov, ktoré je ľahké nainštalovať a naimportovať oproti ostatným frameworkom.

3.5 Node.js

V prípade, kedy klient potrebuje informácie o niečom od klientskej strany aplikácie, požiadavka sa odošle najprv na stranu servera, kde prebieha spracovanie alebo kalkulácie na overenie zadanej požiadavky, prebehne jej vykonanie a príslušná odpoveď sa odošle na stranu klienta. Na obslúženie takýchto požiadaviek sa používa framework JavaScriptu - Node.js.

Je to multiplatformové Javascript prostredie a knižnica na spúšťanie webových aplikácií mimo klientskeho prehliadača. Je ideálna pre aplikácie, ktoré sú dátovo náročné, pretože využíva asynchrónny model riadený udalosťami.

Pre použitie Node.js na serverovej strane aplikácie existuje viacero benefitov:

- Node.js je postavený na stroji Google Chrome V8, preto je jeho čas spustenia ale aj chod programu veľmi rýchly.
- K dispozícii máme viac ako 50 tisíc balíkov, ktoré je možné importovať podľa potrebnej funkcionality, čo rapídne šetrí čas vývoja.
- Pracuje v asynchrónnom režime, čo znamená, že je úplne neblokujúci.
- Skracuje čas načítania zvukových alebo obrazových dát, pretože existuje lepšia synchronizácia kódu medzi serverom a klientom[19].

3.6 NestJS

NestJS je rámec Node.js, ktorý sa predovšetkým používa s jazykom TypeScript[2] na vytváranie škálovateľných a efektívnych aplikácií na serverovej strane aplikácie. Ide o webovú aplikáciu, ktorá sa spolieha hlavne na silné serverové rámce HTTP. Je to nový rámec

Node.js, ktorý nielen imituje, ale aj rieši nedostatky predošlých rámcov Node.js. Na začiatku nového projektu Node.js je NestJS oveľa lepšou voľbou ako ExpressJS[16], pretože je postavený na dizajne, ktorý je zložený z niekoľko jednoduchých komponentov (moduly, ovládače a s lužby). Vďaka tomu je rozdelenie aplikácií na mikroslužby oveľa jednoduchšie.

NestJS sa skladá z 3 základných komponentov:

- **Ovládače** - Spracovávajú prichádzajúce požiadavky a odosielajú odpovede klientskej strane aplikácie.
- **Moduly** - používajú viac štruktúrovaný kód a oddeľujú opakovane používané logické časti od funkcionálnych.
- **Poskytovatelia \služby** - odstraňujú logiku a komplexnosť od používateľa. Je možné ich vložiť do iných služieb, napr. ovládačov apod.

NestJS rovnako ponúka veľa skvelých funkcií:

- Je jednoduché sa ho naučiť a používať.
- Využíva TypeScript - silne typovaný jazyk, ktorý je nadmnožinou JavaScriptu.
- Výkonné rozhranie príkazového riadka (CLI), ktoré slúži najmä na zjednodušenie vývoja a zvýšenie produktivity.
- Podporuje desiatky modulov, ktoré sa ľahko spolupracujú s bežnými technológiami, ako napr TypeORM, Mongoose,GraphSQL, apod.
- Skvelý na jednotkové testovanie.
- Vytvorený pre rozsiahle veľkopodnikové aplikácie.
- ...

NestJS výrazne napomáha pri napredovaní vývoja aplikácií s jej správnym usporiadaním[12].

3.7 ORM a TypeORM

Pre pochopenie čo je to TypeOrm, músíme najprv pochopiť čo znamená skratka ORM v JavaScripte.

3.7.1 JavaScript ORM

ORM je skratka pre objektovo-relačné mapovanie. Je to technika programovania, ktorá má prostriedky na komunikáciu s databázou, pomocou programovacích jazykov, ktoré sú objektovo orientované. ORM generuje objekty, nazývané entity, ktoré sa mapujú virtuálne na tabuľky v databáze. Takýmto spôsobom jednoducho pridáme, upravíme, načítame alebo zmažeme akékoľvek hodnoty v tabuľke, kde sú zložitejšie dotazy nad danou databázou spracované optimalizovaným spôsobom. Objektovo-relačný mapovač je knižnica kódu, ktorá zapuzdruje kód, ktorý pracuje s údajmi namiesto dotazov v SQL. Môže byť napísaný v akomkoľvek jazyku a pracovať s objektami v rovnakom jazyku, ktorý sa momentálne používa k vývoji aplikácii. Výhodou použitie ORM je aj ochrana proti útokom typu SQL Injection, kde knižnica filtruje údaje za nás.

3.7.2 TypeORM

TypeORM je jeden z najobľúbenejších JAvaScriptových ORM a jednou z najpopulárnejších ORM knižníc pre TypeScript projekty. Podporuje širokú škálu najnovších funkcií od JavaScriptu, ES5-8, a zvládne fungovať na mnohých platformách, vrátane vyššie spomínaného Node.js. Vývojárom tak poskytuje vytváranie rozných aplikácie, ktorá pracujú len s niekoľkými tabuľkami alebo aj s rozsiahlymi databázami. Od iných JavaScriptových ORM sa TypeORM líší v podporovaní hlavných vzorov, Data Mapper a Active Record, čo umožňuje vývoj aplikácií s vysokou kvalitou, škálovaním a veľmi dobrým udržiavaním aplikácií. Týmito dôvodmi tvorí TypeOrm flexibilitu, ktorú vývojari potrebujú a využívajú[14].

Kapitola 4

Analýza požiadaviek

V tejto kapitole sa venujem logickým požiadavkám webovej aplikácie, konkrétne diagramu prípadov použitia a vysvetleniu, z akých rolí pozostáva a aké funkcie majú dané role. Ďalšou časťou v tejto kapitole je dokumentácia a riadenie procesu výroby v strojárenskej firme IMA Schelling Slovakia, čomu sa firma venuje, s čím pracuje a z čoho pozostáva dokumentácia jednotlivých častí výroby a riadenie takejto výroby.

4.1 Požiadavky firmy IMA Schelling Slovakia s.r.o.

Firma IMA Schelling Slovakia s.r.o. bola založená v roku 2005, ako malý rodinný podnik v Rakúsku. Časom sa postupne rozrástla do ďalších krajín, ako je Slovensko, Nemecko a Poľsko. Táto firma sa zaoberá hlavne aj týmito predmetmi činnosti:

- Zváračské práce
- Výroba a montáž oceľových konštrukcií
- Povrchové úpravy kovo lakovaním
- Kovoobrábanie
- Výroba účelových strojov mimo elektročastí
- Inžinierske činnosti a súvisiace technické poradenstvo

Firma kladie veľký dôraz na kvalitu a dátum dodania vykonanej práce, od ktorej zavisí aj kvalitný výber zamestnancov.

4.1.1 Technická príprava výroby vo firme IMA Schelling Slovakia s.r.o.

Technická príprava výroby v tejto firme začína dopytom a pozostáva z 3 častí, z administratívnej, výrobnej a evidencii chýb. V administratívnej časti pracovník technickej prípravy výroby, ďalej ako TPV, spracuje cenovú ponuku, ktorú následne zašle zákazníkovi na posúdenie. Zákazník môže cenovú ponuku odmietnuť, ale ak ju prijme, pracovník TPV spracuje výkresovú dokumentáciu k cenovej ponuke, založí k nej viacero protokolov, vrátane zákazky, kusovníka a výrobného postupu. Následne sa k založenej zákazke objedná potrebný materiál od dodávateľov a posunie do výroby.

Vo výrobnej časti pracovník TPV odovzdá výrobnú zákazku vedúcemu výroby. Na oddelení Prípravy dorazí objednaný materiál, ktorý ďalej putuje na pracoviská podľa zadaného výrobného postupu. Na pracoviskách si zamestnanci načítavajú výrobné operácie označené čiarovým kódom, ktoré momentálne vykonávajú. Tieto údaje sa zapisujú priamo do tabuľkového súboru, z ktorého vieme zistiť, kto na danej výrobnej zákazke pracoval, ako dlho a kedy na nej pracoval.

Po vykonaní všetkých výrobných operácií sa výrobná zákazka odošle na pracovisko Oddelenie Technickej Kontroly, ďalej OTK, kde zamestnanec skontroluje, či bol správne dodržaný výrobný postup a skontroluje rozmery podľa výkresovej dokumentácie.

Následne vyrobená zákazka putuje na oddelenie Lakovne, kde sa opracuje lakovaním. Po opracovaní na oddelení Expedícii sa výrobná časť zákazky ukončí, zabalí sa, odošle k zákazníkovi.

V časti dokumentácie sa popri evidencii práce zamestnancov spracovávajú 3 typy chýb:

- **Interné** - Na oddelení OTK zamestnanec pri zistení nesprávne dodržaného postupu založí záznam o chybe, a podľa uložených výrobnej operácií je možné zistiť, kde nastala chyba a kto ju zapríčinil.
- **Externé** - Počas výrobného procesu dôjde k zisteniu chyby na strane dodávateľa, kde bol napr. zle dodaný materiál alebo sa dodanie výrobnej zákazky omeškápre chýbajúci materiál, prípadne zákazník poslal zle zadanú objednávku, ktorá už je vo výrobnom procese
- **Reklamácie** - Zákazník po prijatí vyrobenej zákazky zistí chybu, ktorá nebola odhalená v rámci výrobného postupu ani na oddelení OTK

Tieto chyby je potrebné správne a efektívne ukladať, rovnako ako aj prácu zamestnancov a zoznam výrobných zákaziek podľa nastavených priorít.

4.1.2 Dokumentácia a riadenie procesu výroby v strojárenskej firme

V tejto podkapitole je detailnejšie popísaný postup dokumentácie práce a chýb zamestnanca a riadenie výrobných zákaziek vrátane postupu riešenia pri vzniku problémov v rámci výrobných zákaziek.

Dokumentácia jednotlivých častí výroby

Proces výroby v strojárenskej firme pozostáva z viacerých častí. Začína to objednaním vhodných materiálov, ktoré sú potrebné k ďalšiemu opracovaniu. Kroky týchto opracovaní je dôležité dokumentovať a kontrolovať, aby sme predišli k chybám a aby sa až po poslednom kroku v procese výroby nezistilo, že je to od začiatku zlý kus. Popis a vysvetlenie k týmto chybám je napísaná v sekcii vyššie.

Tieto chyby je potrebné zaznamenávať a z ich štatistiky vie strojárenská firma vyvodiť ďalšie kroky, potrebné k zlepšeniu kvality výroby v ich firme. Okrem chýb sa zaznamenáva aj dokumentácia odvedenej práci zamestnanca a jeho sebakontrola. Ďalej sa zaznamenávajú meracie protokoly rôznych typov, podľa požadovaných vlastností konečného výrobku.

Riadenie výroby

Kým sa zo vstupného neopracovaného materiálu stane finálny výrobok, prejde viacerými oddeleniami, na ktorom sa vykoná časť v rámci jeho výrobného procesu, ktorý patrí pod vý-

robnú zákazku. Pri každej zákazke je potrebné zaznamenať čo sa práve deje, kde sa daná zákazka nachádza a v akom je stave, či náhodou nenastal nejaký problém, ktorý by oneskoril celý výrobný proces tejto zákazky. V takom prípade je nutné, aby zamestnanec dal vedieť vedúcemu výroby v čom nastala chyba a aby sa problém čo najskôr vyriešil.

V rámci výrobnej zákazky je uvedený aj dátum dodania, podľa ktorého vieme, ktoré zákazky musia byť odbavené ako prvé. Každá zákazka na rôznych oddeleniach strávi rôzne veľa času, tak je potrebné aby vedel vedúci výroby týmto zákazkám určiť ich vlastnú prioritu, ktorá bude vyššia ako dátum dodania.

4.2 Diagram prípadov použitia

Ako prvé pri návrhu aplikácie je popísaná skupina vlastností a funkcií dotknutých užívateľov tohto systému, ktoré sú znázornené na obrázku 4.1 a popísané v jednotlivých podsekciách.

4.2.1 Pracovník výroby

Pracovník výroby môže v aplikácii v rámci časti výrobných zákaziek prezerať, vyhľadávať a vykonávať výrobné zákazky zadané vedúcim pracovníkom výroby a taktiež môže v prípade vzniknutia problému daný problém priradiť k výrobnej zákazke a priradené problémy prezerať. V rámci ďalšej časti dokáže pridávať a prezerať záznamy dokumentácie. Taktiež dokáže pozmeniť tie záznamy, ktoré neboli uzavreté vedúcim pracovníkom výroby. Svoju prácu si vie prezrieť aj vo vypracovaných štatistikách.

4.2.2 Vedúci pracovník výroby

Vedúci pracovník výroby môže v aplikácii v rámci časti výrobných zákaziek prezerať, vyhľadávať a vykonávať výrobné zákazky a oproti bežnému pracovníkovi výroby vie tieto výrobné zákazky nahrávať a upravovať. V prípade vzniknutia problému vie rovnako ho priradiť k výrobnej zákazke a oproti bežnému pracovníkovi výroby je schopný mu pridať aj riešenie. V rámci ďalšej časti vedúci pracovník vie záznamy dokumentácie nielen zaznamenávať a prezerať, no vie ju aj uzatvárať, upravovať a mazať. Pre obe tieto časti aj spravuje ich nastavenia, do ktorých patria všetky potrebné informácie pre ich správnu evidenciu. Vyhodnotenú štatistiku tejto práce vie prezerať detailnejšie.

4.2.3 Admin

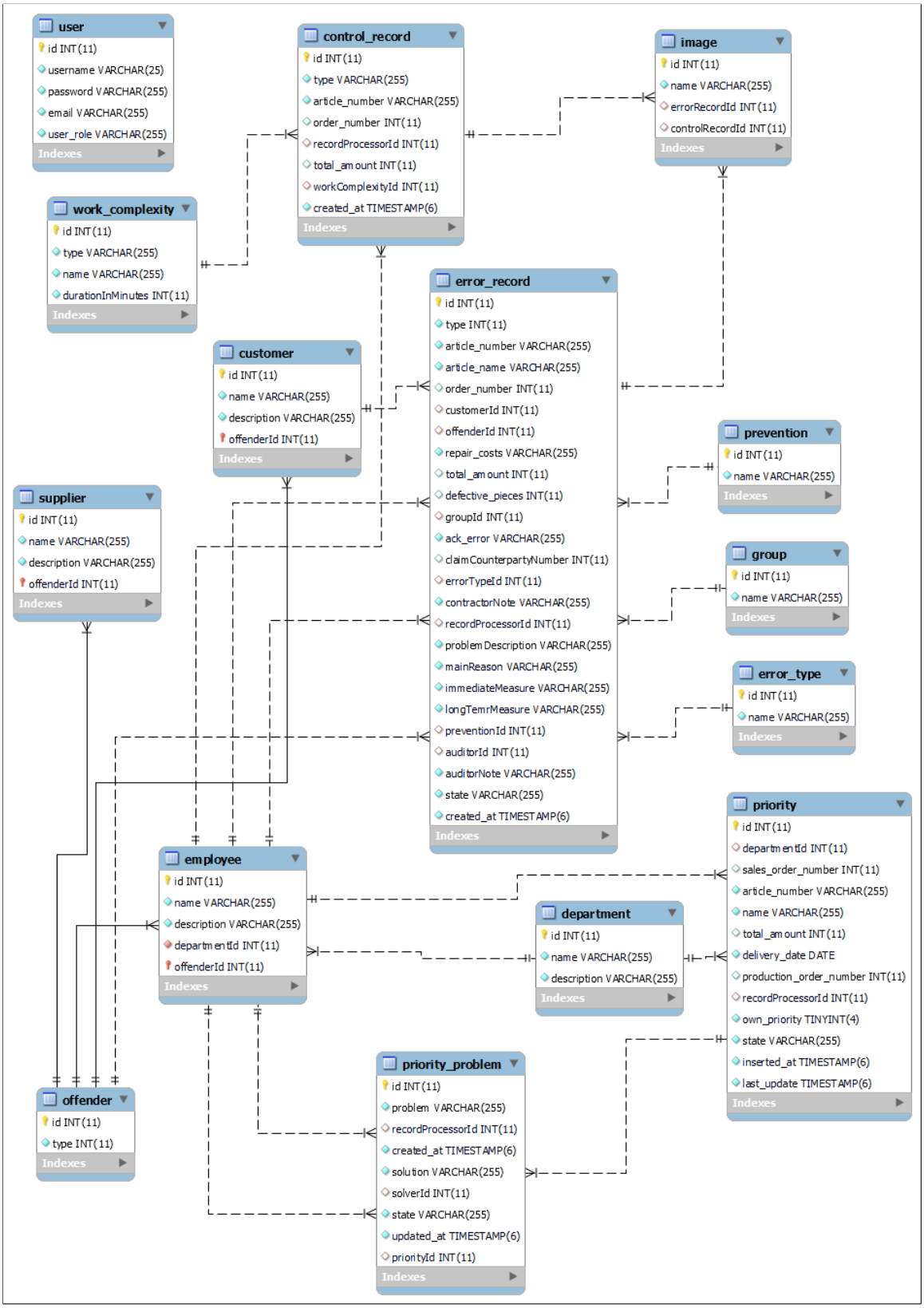
Admin má rovnakú funkcionalitu ako Vedúci pracovník výroby a naviac spravuje aj účty všetkých používateľov v aplikácii.

Obr. 4.1: Diagram prípadov použitia

Kapitola 5

Návrh riešenia

V tejto kapitole sa venujem fyzickému návrhu webovej aplikácie, konkrétne samotnému návrhu užívateľského rozhrania progresívnej webovej aplikácie, ktorá sa skladá z 2 časti. V prvej časti je popísané, ako bude možné zobrazovať, prioritizovať a upravovať výrobné zákazky podľa zadaných požiadavok a v druhej časti je uvedená evidencia dokumentácie práce a chýb zamestnancov, z ktorých budú následne vyhodnotené ich štatistiky. Na začiatku je tiež popísaný diagram vzťahu entít (ER), znázornený na obrázku 5.1, ktorý nám ukazuje, ako ľudia, resp. pracovníci, objekty alebo koncepty, navzájom súvisia v rámci tejto webovej aplikácie. Následne je táto štruktúra tejto schémy vysvetlená v sekcii 5.1.



Obr. 5.1: Schéma relačnej databázy

5.1 Popis schémy relačnej databázy

V tejto sekcii sú vysvetlené vzťahy medzi jednotlivými entitami, resp. tabuľkami, v ktorých sa nachádzajú údaje,ktoré je potrebné evidovať. Postupnosť týchto entít je zoradená podľa závislostí. Najprv sú vysvetlené jednoduché entity, následne entity, ktoré sú závislé na iný entitách.

Entita *user*

V tejto entite sú uložené údaje o používateľoch pre prihlásenie do aplikácie. Entita obsahuje údaje o používateľovi ako prihlasovacie meno a heslo, email a používateľskú rolu, podľa ktorej ma prihlásený používateľ udelený prístup, čo všetko môže vykonávať.

Entita *department*

V tejto entite sú uložené údaje o oddeleniach, ako je názov a popis. Pomocou tejto entity vieme aj vytriediť zamestnacov, ktorí pracujú na danom oddelení.

Entita *offender*

V tejto entite sú uložené údaje o vinníkoch, ktorí sú priradení konkrétnym chybám. Vinníkom môže byť *employee* - zamestnanec firmy, *customer* - zákazník alebo *supplier* - dodávateľ. V rámci údajov sa ukladá identifikačné číslo vinníka a konkrétny typ vinníka, ktorým je 1 z 3 vyššie uvedených.

Entita *customer*

Táto entita obsahuje údaje o zákazníkovi, ako je jeho názov a popis zákazníka. Pri vytvorení nového zákazníka sa najprv vytvorí vinník s typom zákazník, a následne je vytvorený zákazník s daným identifikačným číslom vinníka.

Entita *supplier*

Táto entita obsahuje údaje o dodávateľovi, ako je jeho názov a popis dodávateľa. Rovnako ako aj pri predošlej entite, pri vytvorení nového dodávateľa sa najprv vytvorí vinník s typom dodávateľ, a následne je vytvorený dodávateľ s daným identifikačným číslom vinníka.

Entita *employee*

Táto entita obsahuje údaje o zamestnancovi, ako je jeho meno, popis a oddelenie, kde zamestnanec pracuje. Rovnako ako aj pri predošlej entite, pri vytvorení nového zamestnanca sa najprv vytvorí vinník s typom zamestnanec, a následne je vytvorený zamestnanec s daným identifikačným číslom vinníka.

Entita *prevention*

V tejto entite je uložený názov zabránenia opakovaniu vzniknutej chyby, ktorú daný vinník spôsobil. Následne má používateľ pri vložení nové chybového záznamu na výber z uložených chýb.

Entita *group*

V tejto entite je uložený názov skupiny, do ktorej spadá konkrétny chybový záznam. Následne má používateľ pri vložení nové chybového záznamu na výber z uložených skupín.

Entita *error_type*

V tejto entite je uložený typ danej chyby, do ktorej spadá konkrétny chybový záznam. Používateľ pri vzniknutí chyby zaeviduje jeden typ z uložených v tejto tabuľke.

Entita *image*

Táto entita obsahuje údaje o obrázkoch, ktoré sú uložené na serveri a slúžia ako fotodokumentácia k vloženému záznamu o chybe alebo pri kontrolnom zázname.

Entita *error_record*

Táto entita reprezentuje záznam o chybe a obsahuje najviac vzťahov od iných entít. Pri vložení nového chybového záznamu je potrebné evidovať typ vzniknutej chyby, číslo a názov artikla, číslo výrobnej zákazky, zákazníka, ktorý si danú zákazku objednal, vinníka, ktorý spôsobil danú chybu, ďalej vyčíslené spôsobené škody, celkový počet objednaných kusov a počet poškodených kusov, skupinu chyby, uznanie chyby, číslo reklamačnej protistrany v prípade reklamácie, zadávateľa a jeho poznámku k chybe a hlavný popis vzniknutého problému.

Vyššie spomenuté údaje sú potrebné k vytvoreniu chybového záznamu. Po vložení vedúci pracovník skontroluje vložené údaje a ku chybe pridá krátkodobé a dlhodobé riešenie, zabránenie opakovaniu chyby, poznámku a upraví stav chyby, či je chyba ešte stále otvorená alebo uzavretá.

Entita *work_complexity*

Táto entita reprezentuje časovú náročnosť dielu z výrobnej zákazky, ktorá je potrebná k jej riadnej dokumentácii na oddelení OTK. Zamestnanec pri zadávaní kontrolného záznamu si zvolí akú časovú náročnosť má diel, ktorý kontroloval.

Entita *control_record*

Táto entita je ďalšou, ktorá je závislá na iných entitách. Pri vložení kontrolného záznamu je potrebné evidovať typ kontroly, číslo zákazky a artikla, zadávateľa záznamu, počet kontrolovaných kusov a časovú náročnosť kontrolovaných dielov.

Entita *priority*

Táto entita reprezentuje uložené výrobné zákazky, ktoré obsahujú údaje o oddelení, kde sa momentálne výrobná zákazka nachádza, číslo predajnej a výrobnej zákazky, názov zákazky, počet kusov k výrobe, dátum dodania, pracovníka výroby, ktorý na nej momentálne pracuje, stav prioritnej výroby a celkový stav výrobnej zákazky. Vďaka tejto entite sa pracovníci ľahko zorientujú v tom, na čom majú momentálne pracovať.

Entita *priority_problem*

Pri každej výrobnej zákazke môže nastať nejaký problém. V tejto entite sú uložené údaje o vzniknutom problém, ako je popis problému, meno zadávateľa a dátum vloženia problému. Tieto údaje vloží pracovník výroby a bude čakať na odpoveď vedúceho pracovníka výroby, ktorá zahrňuje popis riešenia problému, riešiteľa a momentálny stav problému.

5.2 Architektúra aplikácie

Architektúra vyvíjanej aplikácie sa skladá z klientskej, teda front-end, strany a serverovej, teda back-end, strany. V rámci front-endu je použitá architektúra Reactu, ktorá je popísaná v nasledujúcej sekcii a back-end zastupuje architektúra NestJS, ktorá už bola spomenutá v kapitole 3, a nižšie je popis použitia na strane servera.

5.2.1 Klientska strana - Front-End

Knižnica React obsahuje množstvo funkcií, ktoré napomáhajú s vývojom UI, teda používateľského rozhrania. Obrovskou výhodou Reactu je, že nie je potrebné používať nejaký preddefinovaný architektonický vzor, ako pri iných knižniciach JavaScriptu. Sami si zvolíme aká bude štruktúra vyvíjanej aplikácie.

Architektúra Reactu

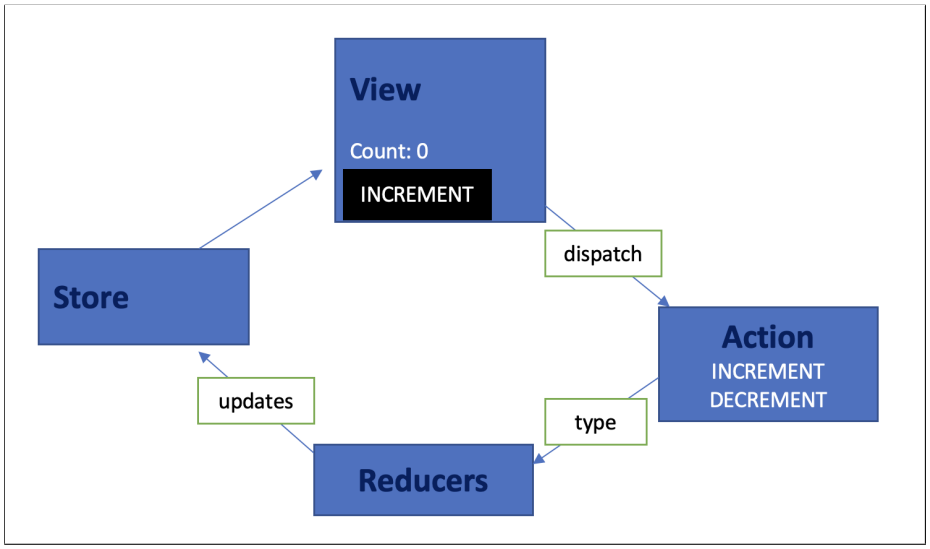
Je to súbor komponentov, ktoré majú zodpovednosť za vytvorenie používateľského rozhrania aplikácie. Na pohľad sa tvári ako organizácia základného kódu pre daný projekt. Architektúra Reactu obsahuje viacero komponentov UI, ako sú formuláre, tlačidlá, služby API, služby pre centrálne riadenie stave a pod. Práve pre tieto spomínané komponenty je React najpreferovanejšiou knižnicou JavaScriptu na vytváranie front-endu. Jedným z takýchto vzor je aj vzor **Redux**[6].

Redux je kontajner stavov, ktorý sa vyznačuje svojou predvídateľnosťou tak, aby aplikácie v JavaScripte sa správali súdržne naprieč klientskou a serverovou stranou, natívnymi prostrediami a aby dobre testovateľné. Vrátane závislostí zaberá iba 2kB, takže nezväčšuje veľkosť aktív aplikácie. Stav aplikácie je uložený v *Store* a každý komponent má prístup k akémukoľvek stavu, uloženému v *Store*.

Jasne povedané, používa sa na udržiavanie a aktualizáciu uložených dát vo vyvíjanej aplikácií, ktoré sa zdieľajú naprieč komponentami, pričom je to celé nezávislé od týchto komponentov. Bez Reduxu by dáta boli podriadené komponentám a museli by byť tam, kde sú potrebné.

Síce má Redux svoju samostatnosť, a je použiteľný s rôznymi ďalšími frameworkami, ako Angular, Vue, Inferno a pod., najčastejšie je používaný práve s Reactom, lebo React bol navrhnutý ako koncept životných cyklov a stavov. V Reacte sa stav premennej upravuje pomocou funkcie *setState*, nie je možné ho upraviť priamo. Takto spolu zdieľajú rovnakú správu objektov[13].

Architektúra Reduxu je znázornená nasledujúcom vývojovom diagrame:



Obr. 5.2: Vývojový diagram architektúry Redux [13]

Na diagrame 5.2 vidíme tri hlavné časti architektúry:

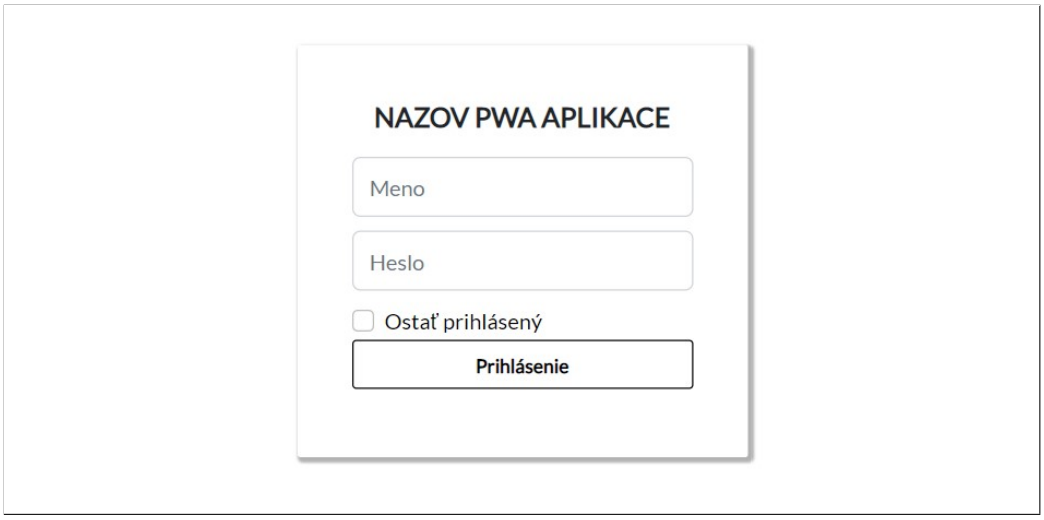
- **Action** - je jediným zdrojom informácií pre *Store*, a prenáša veľké množstvo údajov z aplikácie do pamäti. Základom je atribút *type*, ktorý označuje typ vykonávanej akcie. Rieši to problémy, ak chceme vykonať iba jednu akciu namiesto viacerých.
- **Store** - predstavuje strom objektov. Obsahuje stav aplikácie uložený v stavovom kontajneri, ktorý je maximálne jeden v rámci aplikácie. K vytvorenému *Store* musí byť vytvorený *Reducer*.
- **Reducer** - je čistá funkcia, ktorý zakaždým vráti rovnaký výstup daného vstupu. Nie je potrebné asynchrónne volanie alebo globálnu premennú. Vstupom sú 2 premenné: akcia a predchádzajúci stav. Výstupom je redukcia premenných na jednu aktualizovanú entitu stavu.

5.2.2 Serverova strana - Back-End

NestJS sa hlavne používa na vytvorenie aplikácii na strane servera pre obslúženie klientskej strany aplikácií Node.js. Umožňuje zostaviť aplikáciu s ľubovoľne stanovenými podmienkami, čo je veľkou výhodou. Je jednoduché sa v ňom zamotať a tak je potrebné si zjednotiť požidavky, ktoré od aplikácie vyžadujeme. Poskytuje užitočné nástroje ako Dependency Injection, rôzne filtre, ovládače, moduly, ochrany a podobne[17].

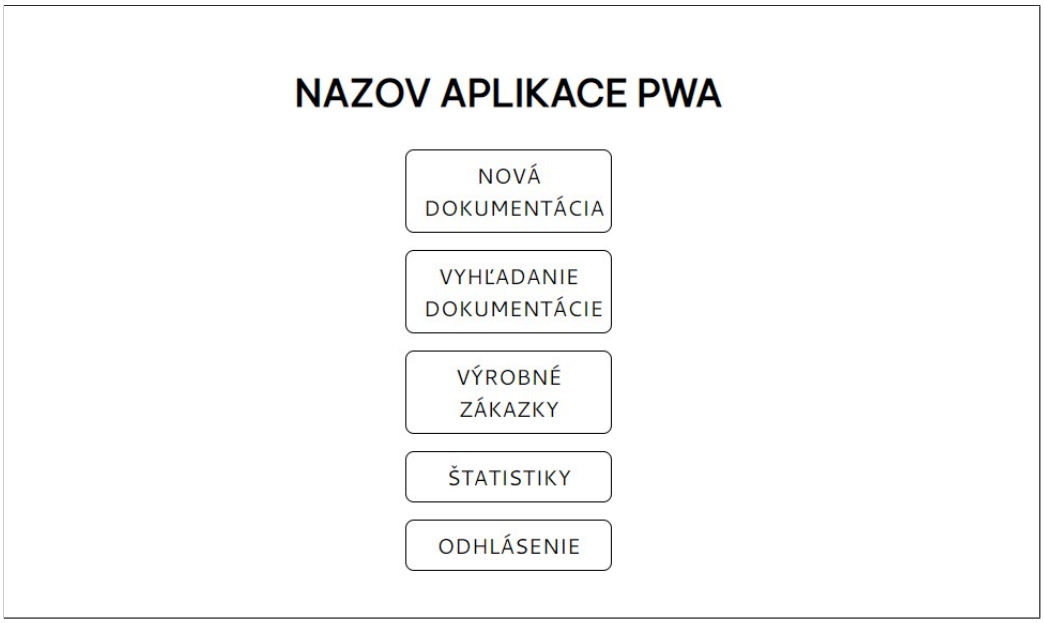
5.3 Užívateľské rozhranie

Vzhľadom na to, že táto aplikácia ma slúžiť ľudom v strojárenskej firme, je rozmiestnenie a zobrazenie jednotlivých častí webovej aplikácii pomerne jednoduché. Po otvorení webovej aplikácie je užívateľ povinný sa prihlásiť. Na prihlásenie sú potrebné len základné údaje, meno a heslo, ktorého návrh je zobrazený na obrázku 5.3.



Obr. 5.3: Návrh stránky pre prihlásenie

Po prihlásení je užívateľ presmerovaný do hlavného navigačného menu, ktoré je znázornené na obrázku 5.4, v ktorom si zvolí časť evidencie dokumentácie, časť vyhľadávania zaznamenanej dokumentácie, časť výrobných zákaziek alebo je používateľ schopný odhlásiť sa. V prípade, že je prihlásený vedúci pracovník výroby alebo admin, vidí aj vyhodnotené štatistiky práce a chýb pracovníkov.



Obr. 5.4: Návrh navigačného menu

5.3.1 Nová dokumentácia

V tejto časti používateľ zadáva jednotlivé informácie výrobnej zákazky, ktoré sú znázornené na obrázku 5.5. V prípade artikla a čísla výrobnej zákazy, vie pracovník zadať tieto hodnoty pomocou naskenovania čiarového kódu, ako je znázornené na obrázku 5.6, kde sa užívateľovi zobrazí okno s textom čo má naskenovať a aj výber dostupných zariadení, pomocou ktorých

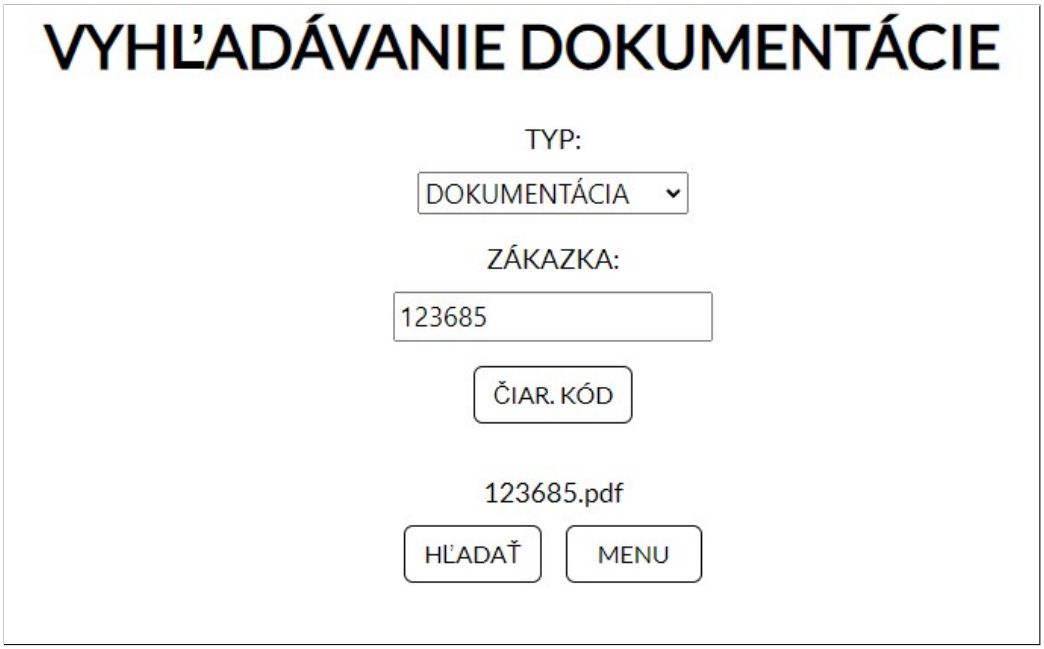
je schopný čiarový kód výrobnej zákazky naskenovať. Príklad časti výrobnej zákazky je znázornený v prílohe A.

Obr. 5.5: Návrh pridanie novej dokumentácie

Obr. 5.6: Návrh skenovania čiarového kódu

5.3.2 Vyhľadanie dokumentácie

Pod touto časťou si užívateľ bude môcť vyhľadať informácie k výrobnej zákazke podľa zadania jej identifikačného čísla alebo aspoň jeho podreťazca. Rovnako aj tu je používateľ schopný toto identifikačné číslo naskenovať pomocou čiarového kódu. Keďže je viacero typov dokumentácie, ktorú pracovník eviduje, zvolí si konkrétny typ dokumentácie a tú si presne vyhľadá. Ukážka je znázornená na obrázku 5.7.



Obr. 5.7: Návrh vyhľadávania dokumentácie

5.3.3 Výrobné zákazky

V tejto časti si pracovník môže prezerať výrobné zákazky podľa dátumu dodania, podľa ich čísla artikla alebo čísla výrobnej zákazky. Ďalej to je počet kusov k výrobe, problémy k danej zákazke, jej celkový stav (problém, pracuje sa alebo dokončená) a aj či nejaká zákazka nemá určenú prioritu vedúcim výroby.

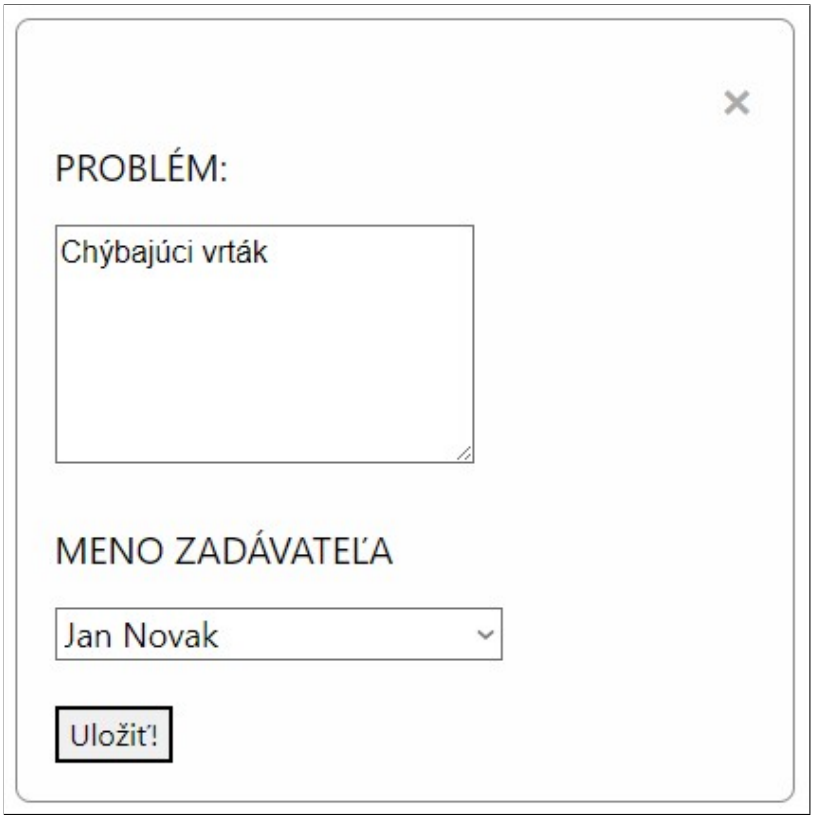
Pri zobrazení problému červená farba zobrazuje nový problém, oranžová farba zobrazuje, že problém je v štádiu riešenia, zelená farba zobrazuje, že problém je vyriešený a šedá zobrazuje pridanie problému.

Pri stave výrobnej zákazky výkričník v červenom poli značí, že pri danej zákazke vznikol problém. Točiaci sa kruh značí, že na zákazke sa pracuje a „fajka“ v zelenom poli znamená, že zákazka je dokončená na danom oddelení.

Všetky tieto stavy sú znázornené na obrázku 5.8.

Obr. 5.8: Návrh zobrazenia výrobných zákaziek

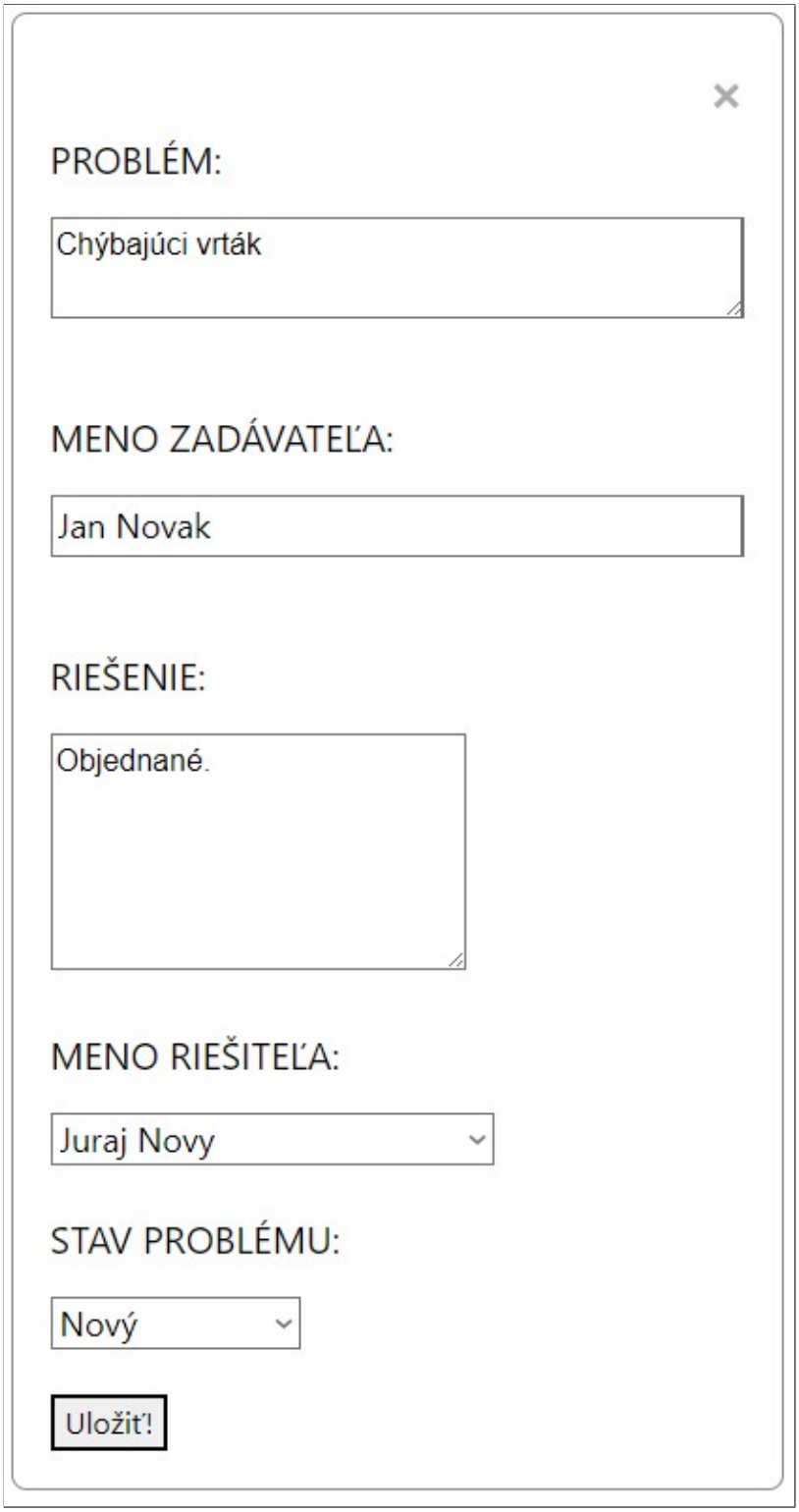
Pri pridaní problému je dôležité zadať aký problém nastal a kto problém zadáva. Príklad zadania problému je znázornený na obrázku 5.9.



Obr. 5.9: Návrh pridania problému

Pri pridaní riešenia k vzniknutému problému sa len zobrazí aký problém nastal a kto ho zadal. V rámci úpravy je dôležité, aby vedúci pracovník zadal aké je riešenie daného

problému, ktorý vedúci pracovník ho rieši/vyriešil podľa stavu problému v riešení/vyriešený. Príklad zadania riešenia k vzniknutému problému je znázornený na obrázku 5.10.



Obr. 5.10: Návrh pridania riešenia k vzniknutému problému

5.3.4 Vyhodnotenie štatistík

Na základe vloženia jednotlivých dát z evidencie dokumentácie práce a chýb zamestnancov zobrazíme ich jednotlivé štatistiky, pričom pracovníci budú vidieť štatistiky len s uvedenými identifikačným číslami a vedúci pracovníci budú vidieť aj ich mená a budú vedieť usúdiť, ktorý pracovník ako pracuje. V prvej zobrazenej štatistike budú zobrazené počty chýb, v ktorých bol pracovník vinníkom a v druhej štatistike budú zobrazené počty dokumentácií, ktoré pracovník zaevidoval.

Kapitola 6

Implementácia

V tejto kapitole sa venujem implementácii klientskej a serverovej strane aplikácií.V rámci klientskej strany je implementácia rozdelená do 4 Sekcií, Evidencia chýb a práce zamestnancov pod sekciou OTK, Výrobné zákazky pod sekciou Priority, Vyhodnotená práca zamestnancov pod sekciou Štatistiky a v rámci Nastavení sú implementované súvisiace komponenty, potrebné pre správne fungovanie aplikácie.

6.1 Implementácia serverovej strany

Po nainštalovaní potrebných balíčkov bol pomocou príkazu *nest new backend* vytvorená základná štruktúra aplikácie, ktorá neobsahovala žiadne služby, ovládače ani entity, len hlavný modul, v ktorom sa spustí serverom, ktorý načúva nastavenému portu. Časť tohto kódu vyzerá nasledovne:

```
...
const app = await NestFactory.create(AppModule,{ cors: true,logger});
...
await app.listen(4001);
...
```

V hlavnom module sa zavolá funkcia *forRoot* z modulu *TypeOrmModule*, ktorá zabezpečí napojenie aplikácie k databáze.

Vytvorenie tabuliek v databáze bolo formou „code-first“, čo znamená, že zdrojový adresár obsahuje adresáre, s konkrétnou entitou, modulom, ovládačom a službou. To značne uľahčuje správu tabuliek v danej databáze. V rámci hlavného modulu boli importované aj iné moduly, ako *MulterModule*, ktorý zabezpečuje ukladanie nahrávaných súborov do zvoleného adresára v aplikácii. Ďalším modulom je *ServeStaticModule*, ktorý sprístupňuje uložené súbory používateľovi. Pre načítanie a uloženie dát z tabuľkového súboru programu Excel sa stará balíček *xlsx*.

Prihlasovací modul obsahuje aj stratégie pre používateľské profily, ktoré im dovoľujú sa dotazovať z klientskej strany podľa zadanej úrovni.

Pre nainštalovanie potrebných balíčkov serverovej strany pomocou príkazu *npm install* je možné aplikáciu spustiť. Pre spustenie je potrebné zadať príkaz *npm run start*.

6.2 Implementácia klientskej strany

Po nainštalovaní potrebných balíčkov bol pomocou príkazu *npx create-react-app frontend –template cra-template-pwa* vytvorená základná štruktúra aplikácie React, obohatená o vlastnosti PWA aplikácie. V ďalších podsekciách je rozpísaná súslednosť krokov, ako sa klientska strana aplikácie vyvíjala.

6.2.1 Prihlasovanie do aplikácie

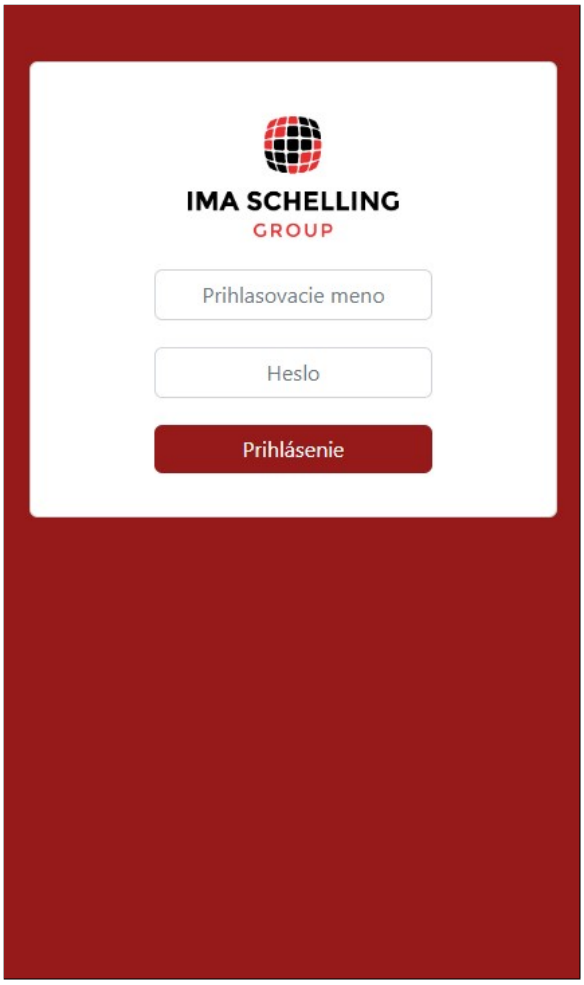
V rámci prihlasovania sa na overenie bezpečnosti používateľa používa JWT - Json Web Token[18], ktorý využíva 2 typy tokenov:

- *access* - prístupový token, ktorý kontroluje oprávnenie používateľa na požiadavok
- *refresh* - obnovovací token, ktorý sa používa na generovanie nového prístupového tokenu. Obyvkle má prístupový token nastavený dátum expirácie, a po jej uplynutí by sa musel používateľ znova autentifikovať, aby tento token získal.

V tejto časti sú implementované vyššie spomínané stratégie, ktoré sú vytvorené pre prihlasovanie, pre získanie obnovovacieho tokenu a pre spracovanie dotazov z klientskej strany podľa používateľského profilu. Nižšie vidíme stretégiu overenia používateľa pre kontrolu dotazu na úrovni profilu Admin :

```
async validate(req,payload: any) {
  var user = await this.userService.getUserByName(req.username);
  if(!user || user.user_role!="Admin"){
      throw new UnauthorizedException();
  }
  return { userId: user.id, username: user.username,
      email:user.email, user_role:user.user_role};
}
```

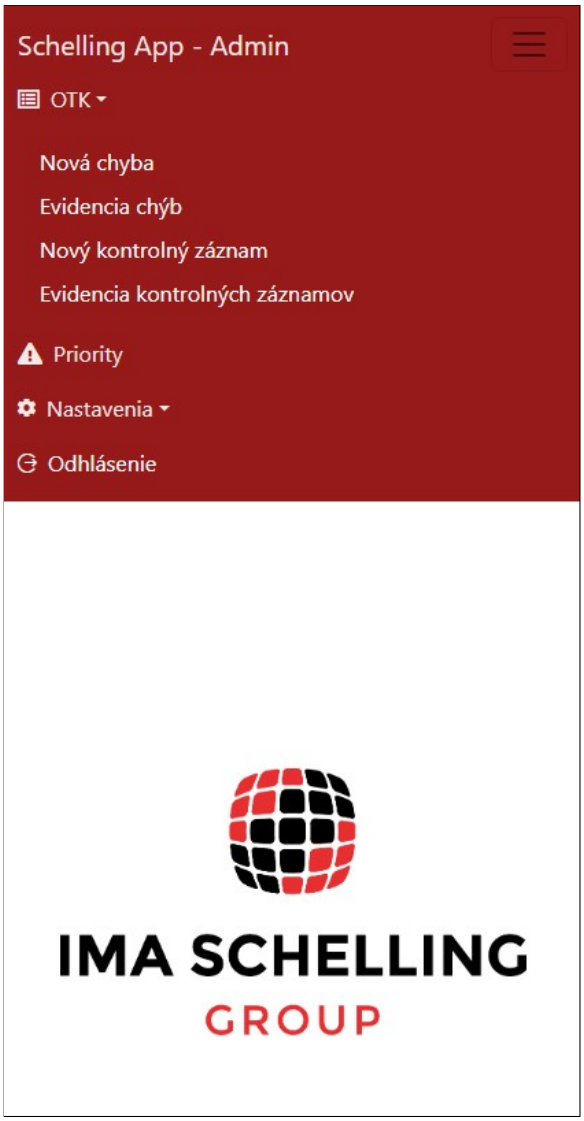
Nainštalovanie potrebných balíčkov klientskej strany spustíme pomocou príkazov *npm install* a *npm install serve*. Pre spustenie aplikácie zadáme príkazy *npm run build* a následne *serve -s build*. Zobrazí sa nám prihlasovací formulár, znázornený na obrázku 6.1. Po prihlásení sa získané tokeny uchovávajú v lokálnej pamäti, ktoré sa pri ďalšej požiadavke prikladajú do hlavičky požiadavky a kontrolujú na serverovej strane aplikácie.



Obr. 6.1: Prihlasovanie do aplikácie

6.2.2 Navigačné menu

Po prihlásení sa používateľ dostane na domovskú obrazovku, ktorej jediným obsahom je navigačné menu a logo firmy. V tomto navigačnom menu si používateľ zvolí, kam sa chce dostať. Meno pre používateľský profil Pracovník výroby je znázornené na obrázku 6.2. V sekcii Nastavenia je znázornené rozsiahlejšie navigačné menu pre používateľský profil Admin.



Obr. 6.2: Navigačné menu pre používateľský profil Pracovník výroby

6.2.3 Sekcia Nastavenia

Po kliknutí na sekciu Nastavenia má používateľ na výber, ktorý komponent chce spravovať. Používateľ Pracovník výroby sekciu Nastavenia nemá prístupnú. Používateľ pracovník nemá prístupnu iba podsekciu „Používatelia“, ktorú má na starosti Admin. Takýmito komponentami sú napr. oddelenia, zamestnanci, zákazníci a dodávatelia, a mnoho ďalších. Táto sekcia je zobrazení na obrázku 6.3.

Obr. 6.3: Sekcia Nastavenia

6.2.4 Sekcia OTK

V rámci sekcie OTK má používateľ viacero možností. Pracovník výroby môže zaznamenávať chyby a v rámci kontrolných záznamov ich môže pridávať a upravovať. Vedúci pracovník výroby tieto chyby dokáže aj zmazať alebo upraviť, a dokáže aj zmazať kontrolné záznamy. Príklad pridávania chyby je znázornený na obrázku 6.4.

Obr. 6.4: Pridanie chyby

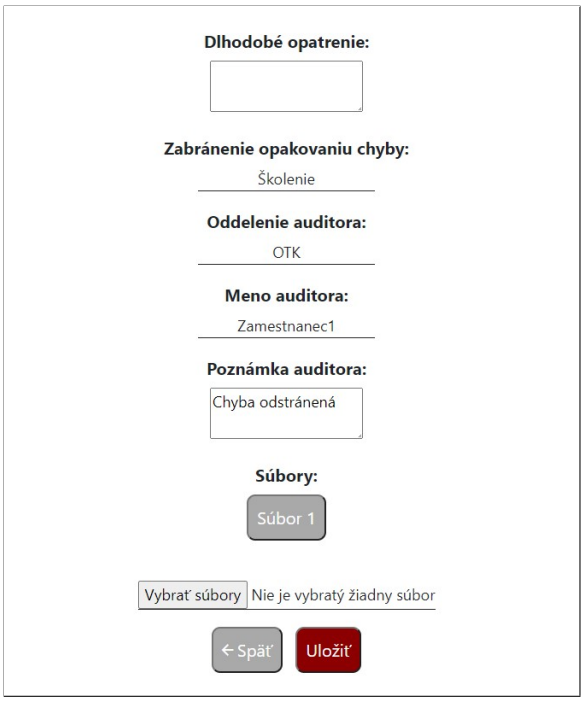
Pri procese spracovania chyby do aplikácie používateľ môže číslo zákazky a artiklu naskenovať pomocou čiarového kódu, čo mu uľahčí prácu s dlhším vypisovaním. Príklad takého skenovania je znázornený na obrázku 6.5. Pri tom zadáva aj ďalšie hodnoty potrebné k evidencii, ako meno zadávateľa a meno vinníka, typ spracovanej chyby, popis problému. K tomu vie priložiť aj fotodokumentáciu.

Obr. 6.5: Skenovanie čísla artikla

Po zaevidovaní chýb je možné si tieto chyby prezerať, prípadne upraviť alebo vymazať. Zaevidované chyby sú znázornené na obrázku 6.6.

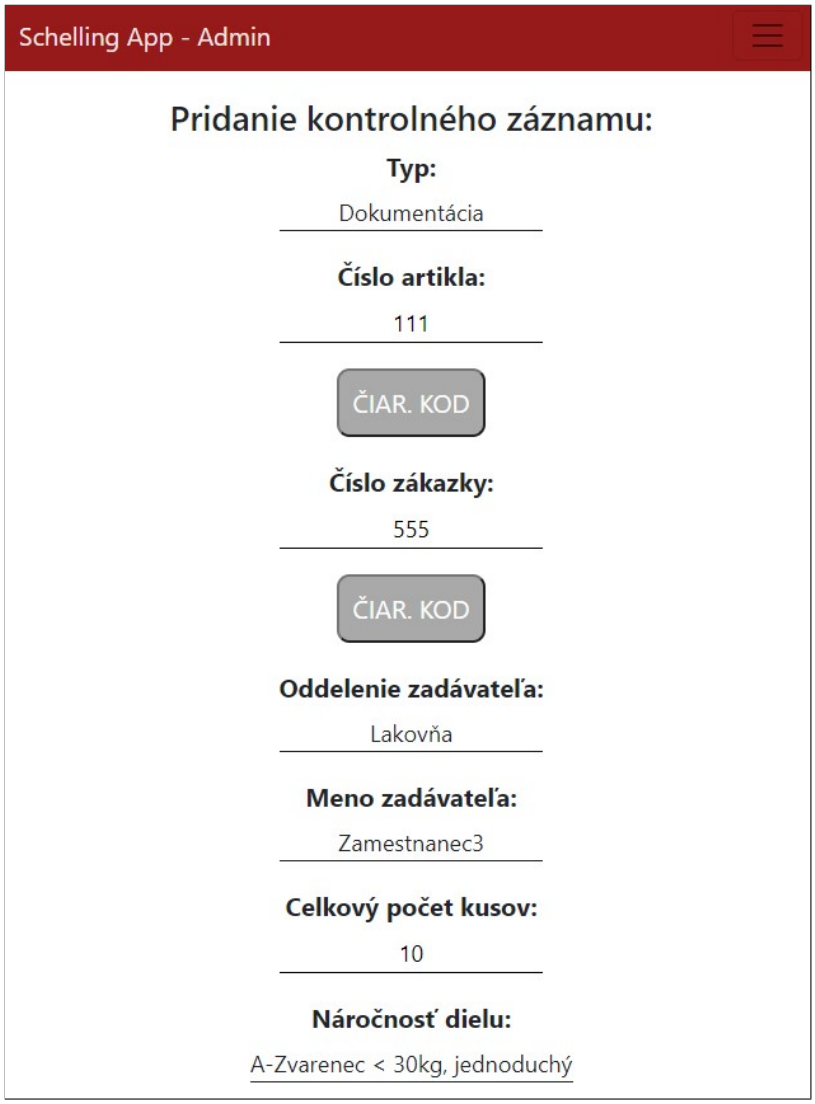
Obr. 6.6: Evidencia chýb

V rámci úpravy chyby vie používateľ prezerať pridané súbory, čo je znázornené na obrázku 6.7.



Obr. 6.7: Prezeranie súborov

Ďalším hlavným bodom sekcie OTK je pridávanie kontrolných záznamov. Tu si vie rovnako pomôcť naskenovaním čísla artiklu a zákazky. Okrem základných hodnôt tu používateľ uvádza aj počet meraných kusov a zvolí kategóriu náročnosti, do ktorého diel zákazky patrí. Príklad pridávania kontrolného záznamu je znázornený na obrázku 6.8.



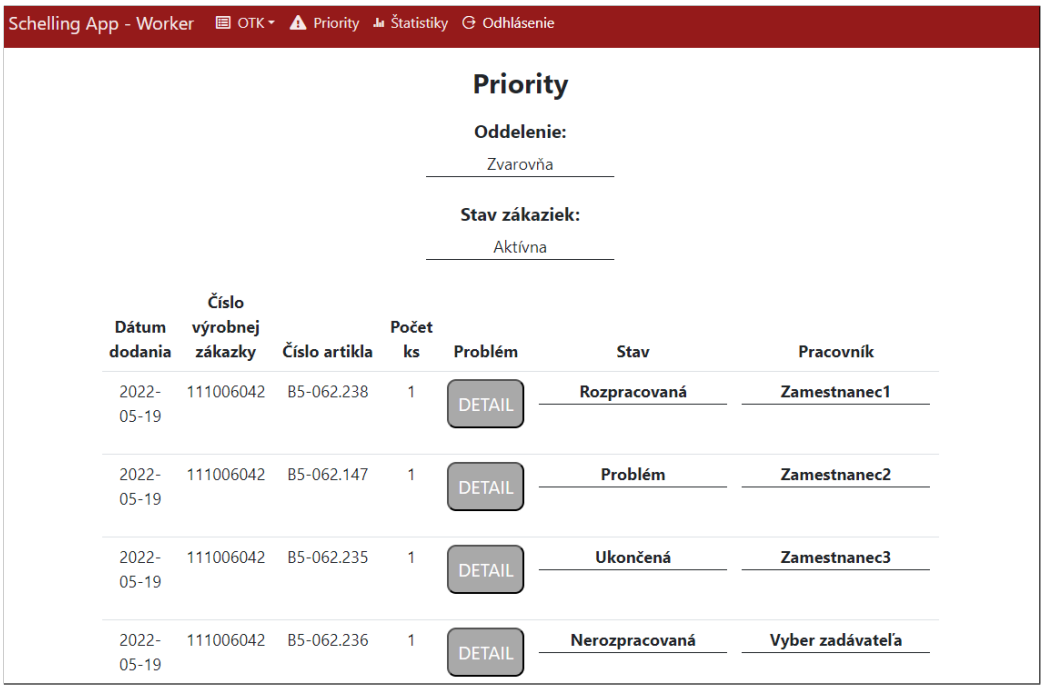
Obr. 6.8: Pridanie kontrolného záznamu

Po zaevidovaní kontrolných záznamov je možné si tieto chyby prezerať, prípadne upraviť alebo vymazať. Zaevidované kontrolné záznamy sú znázornené na obrázku 6.6.

Obr. 6.9: Evidencia kontrolných záznamov

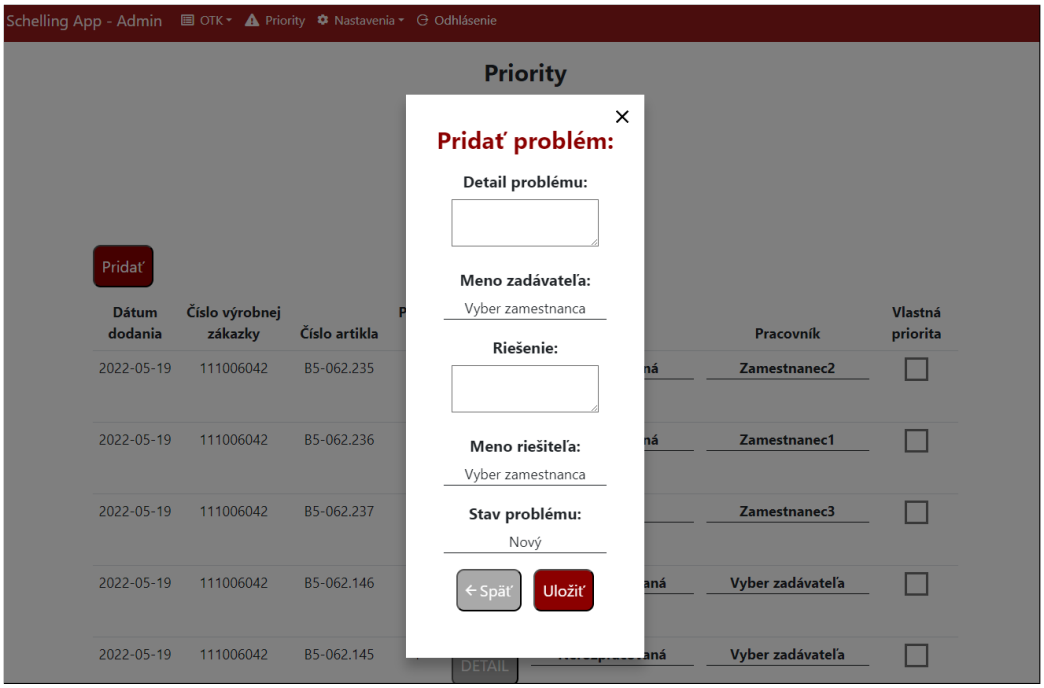
6.2.5 Sekcia Priority

V rámci sekcie Priority sa pracovníkovi výroby ukážu aktuálne výrobne zákazky, ktoré môže prezerať podľa zvoleného oddelenia a stavu zákaziek. Taktiež môžu nastaviť stav zákazky, pracovníka, ktorý na zákazke momentálne pracuje a pridať problém, ak nejaký vznikol. Vlastná priorita sa pracovníkovi výroby zobrazuje len na prezeranie. Príklad zobrazenia výrobných zákaziek pre úroveň Pracovník výroby je znázornený na obrázku nižšie:



Obr. 6.10: Zobrazenie výrobných zákaziek pre úroveň Pracovník výroby

V tabuľke zobrazenia výrobných zákaziek v stĺpci Problém po rozkliknutí tlačidla „Detail“ Pracovník výroby dokáže prezerať evidované problémy alebo pridať nový problém. Formulár na pridanie problému má nasledovný vzhľad:



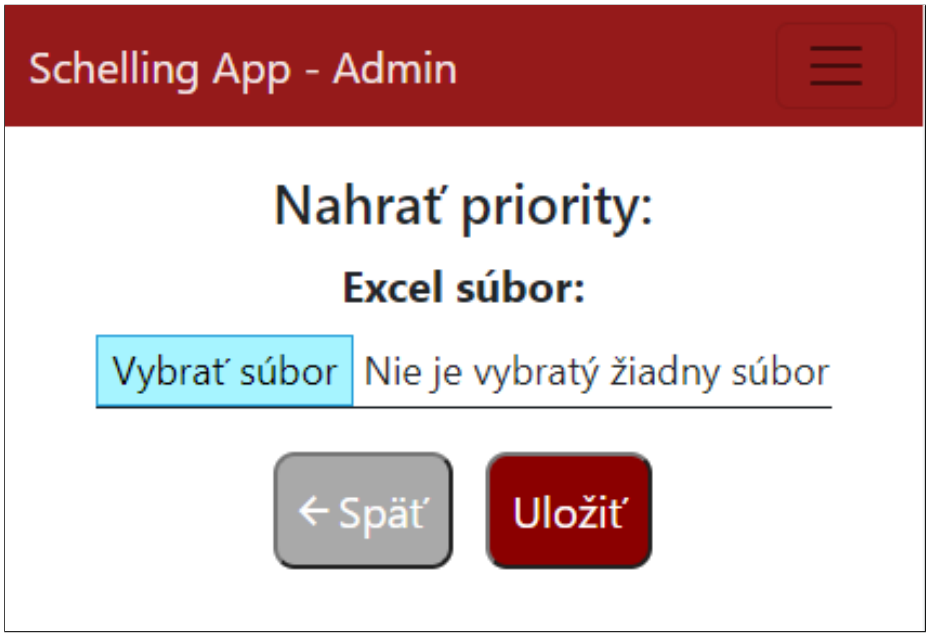
Obr. 6.11: Formulár pridania problému

Vedúcemu pracovníkovi sa naviac zobrazí aj možnosť nahrania zákaziek v podobe súboru programu Excel, v ktorom sú pravidelne tieto výrobné zákazky aktualizované. Podoba, v akej sú tieto výrobné zákazky uložené, vyzerá nasledovne:

	A	B	C	D	E	F	G
1	Predajna_zakazka	Cislo_artikla	Pocet_kusov	Datum_dodania	Vyrobna_zakazka	Nazov	Oddelenie
2	111006042_100	S11062.146	1	19.5.2023	119017197	Zakazka1	Zvarovňa
3	111006042_40	S11062.147	5	22.5.2023	119017191	Zakazka2	Lakovňa
4	111006042_50	S11062.148	4	22.5.2023	119017192	Zakazka3	OTK
5	111006042_60	S11062.149	1	22.5.2023	119017193	Zakazka4	Zvarovňa
6	111006042_70	S11062.150	1	22.5.2023	119017194	Zakazka5	Lakovňa
7	111006042_80	S11062.151	2	22.5.2023	119017195	Zakazka6	OTK
8	111006042_90	S11062.152	9	23.5.2023	119017196	Zakazka7	Zvarovňa
9	111005834_30	S11062.153	10	25.5.2023	119016589	Zakazka8	Lakovňa
10	111005995_10	S11062.154	5	26.5.2023	119017006	Zakazka9	OTK

Obr. 6.12: Výrobne zákazky v súbore programu Excel

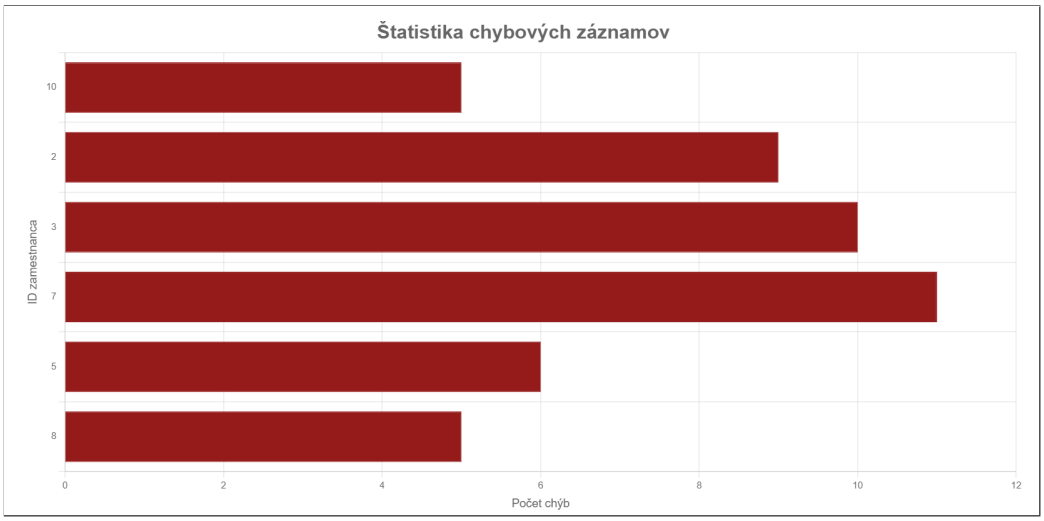
Vedúci pracovník aktualizované výrobné zákazky skontroluje, prípadne nastaví správne hodnoty, ktorá má výrobná zákazka obsahovať a daný súbor nahraje na server skrz formulár, ktorý je znázornený na obrázku 6.13.



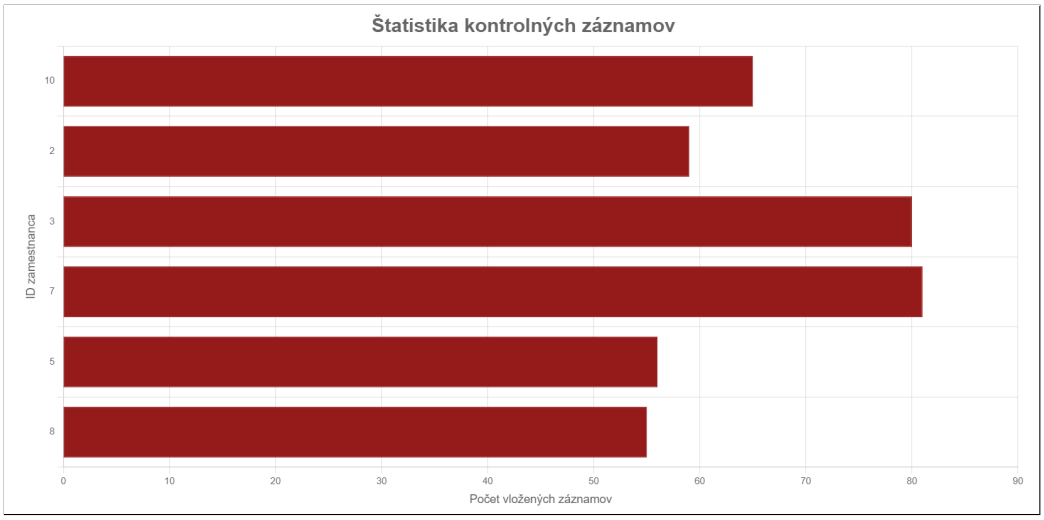
Obr. 6.13: Formulár na nahranie výrobných zákaziek

6.2.6 Sekcia Štatistiky

K tejto sekcii majú prístup všetci prihlásení používatelia. Sú tu vykreslené štatistiky v grafickej podobe. Na prvom grafe, zobrazený na obrázku 6.14, je znázornený pomer počtu chybových záznamov, v ktorých bol daný pracovník výroby vinníkom. Na druhom grafe, zobrazený na obrázku 6.15, je znázornený pomer počtu kontrolných záznamov, ktoré pracovník výroby vykonal. Z dôvodu ochrany osobných údajov je zviditeľnené iba identifikačné číslo zamestnanca. Z týchto údajov môže vedúci pracovník výroby získať informácie kto a ako kvalitne svoju prácu vykonával.



Obr. 6.14: Grafické znázornenie štatistiky chybových záznamov

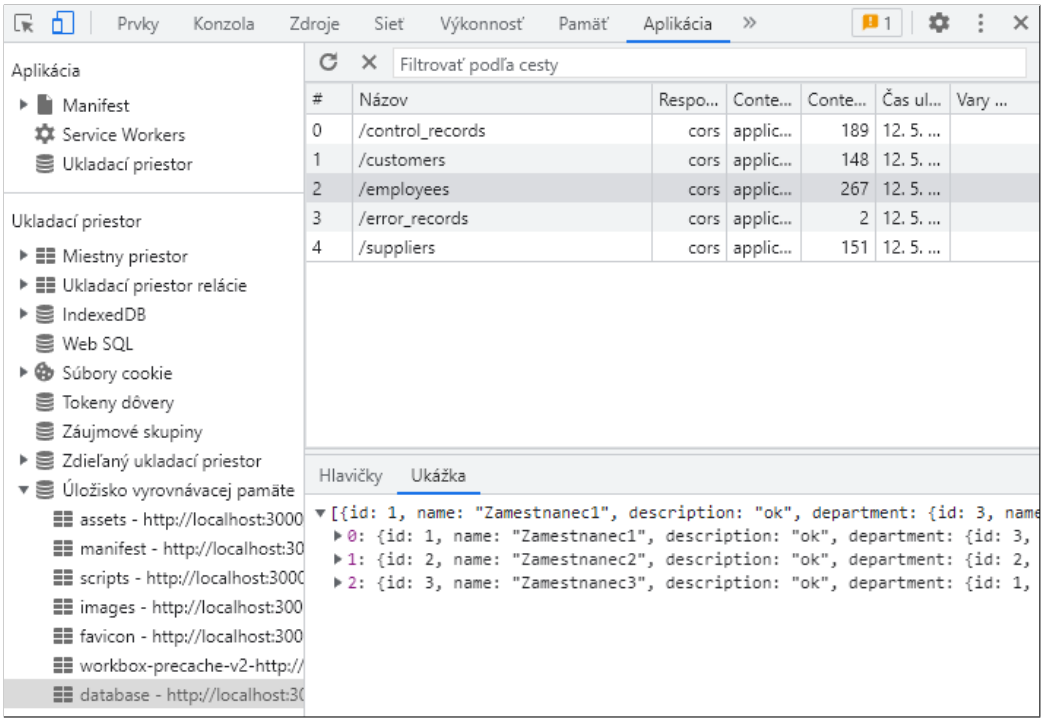


Obr. 6.15: Grafické znázornenie štatistiky kontrolných záznamov

6.3 Implementácia offline režimu

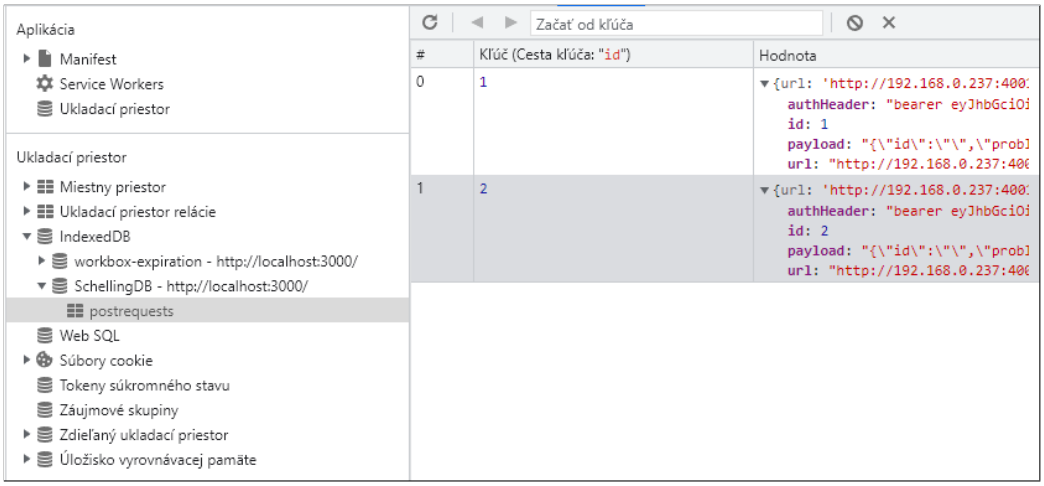
V rámci vlastností progresívnej webovej aplikácie boli implementované viaceré rozšírenia. Pracovník výroby dokáže pracovať aj s aplikáciou s nie príliš dobrým internetovým pripojením a aj bez pripojenia. Údaje, ktoré sú potrebné pre nahratie chybových alebo kontrolných záznamov, sa neukladajú len do „state“ v architektúre Redux, ale aj do internej pamäti webového prehliadača nazývanej „IndexedDb“, ktorá ma podobu ukladania dát kľúč-hodnota.

Pre prihlásenie je potrebné pracovať ešte v online režime, resp. s pripojením na internet. Po prihlásení sa potrebné dáta pomocou žiadostí odoslú na server prvýkrát a webový prehliadač si ich uloží do svojej vyrovnávacej pamäte. Ak sa požiadavka odošle druhý raz, najprv sa skontroluje vyrovnávajúca pamäť, či neobsahuje potrebné údaje. Ak áno, a stav týchto údajov je platný, uložené dáta sa vrátia ako odpoveď. Ak sú údaje zastaralé, pamäť tieto údaje zaktualizuje. Na obrázku 6.16 vidíme takto uložené dáta vo webovom prehliadači. V inom prípade sa požiadavok odošle na server.



Obr. 6.16: Uložená dáta vo vyrovnávacej pamäti

Vyššie spomínané ukladanie dát je zamerané na načítanie dát uložených v databáze. Pre ukladanie dát sa skontroluje možnosť odoslania požiadaviek na stranu servera. Ak to je možné, požiadavky sa odošlú. Ak to možné nie je, dáta sú uložené do pamäti „IndexedDb“ na potrebný čas a periodickým vyzývaním sa kontroluje možnosť ich odoslania. Po pripojení k serveru sa dáta odošlú a z pamäti sa vymažú. Na obrázku 6.16 vidíme takto uložené dáta vo webovom prehliadači, ktoré čakajú na odoslanie.

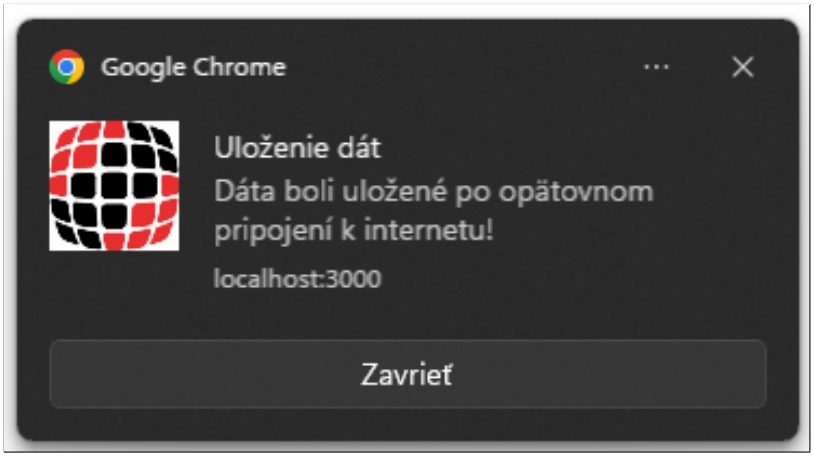


Obr. 6.17: Uložená dáta vo vyrovnávacej pamäti čakajúce na odoslanie

Všetky tieto úkony sprostredkováva správne nastavený Service Worker, ktorý sa najprv nainštaluje a aktivuje vo webovom prehliadači a pri správnom nastavení tieto údaje zachytí a primerane na nich odošle odpoveď.

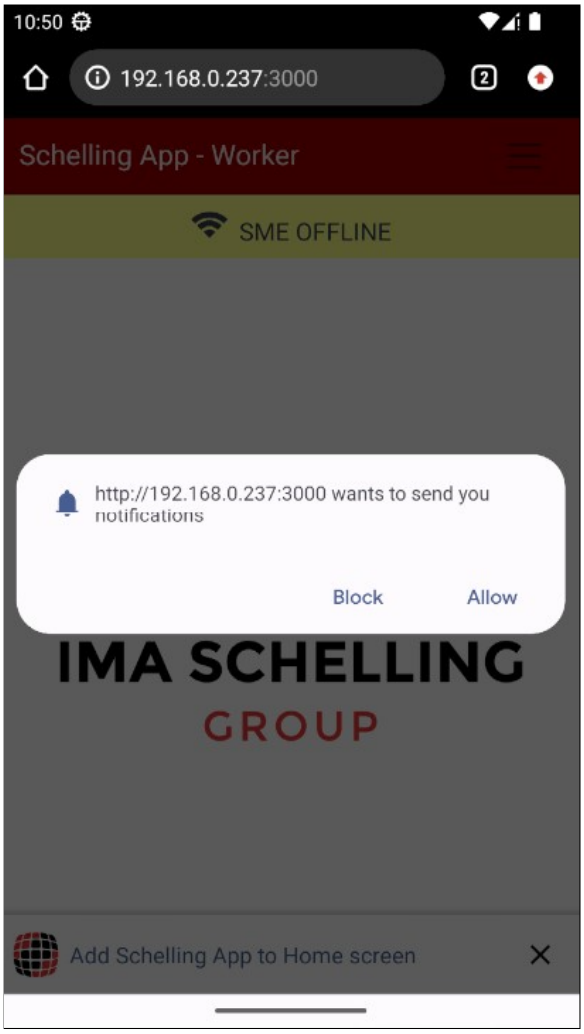
6.4 Implementácia push notifikácií

V predchádzajúcej sekcii bolo vysvetlené ako sa požiadavky ukladajú do databázy prehliadača a potom odosielajú na server, ak je to možné. Po odoslaní požiadavku Service Worker čaká na odpoveď, ktorá príde zo strany servera. Túto odpoveď zobrazí v podobe notifikácie, ktorú zašle používateľovi a on si ju dokáže prečítať. Príklad kladnej odpovede je znázornený na obrázku 6.18. Tieto notifikácie si musí ale používateľ vo webovom prehliadači povoliť. Väčšina dostupných prehliadačov tieto upozornenia podporujú.



Obr. 6.18: Notifikácia po uložení dat

Na obrázku 6.19 je možné vidieť, ako povoliť takéto notifikácie na mobilnom zariadení. Keďže je aplikácie správne nastavená a implementovaná, dole v pozadí vidíme aj ponuku na nainštalovanie tejto progresívnej webovej aplikácie.

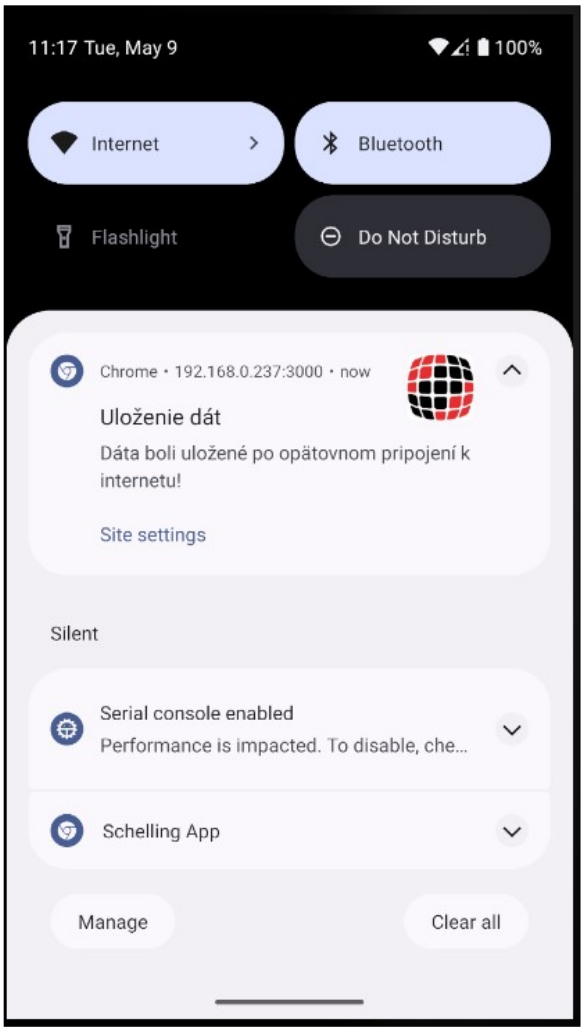


Obr. 6.19: Povolenie zasielania notifikácií na mobilnom zariadení

Kapitola 7

Testovanie

Takto navrhnutá a implementovaná aplikácia bola otestovaná vo webovom prehliadači ako na počítači, resp „desktope“, tak aj na mobilnom zariadení. Aplikácia sa chová rovnako ako na desktope aj mobilnom zariadení. Oproti desktopovému prehliadaniu notifikácia, ktorá je prijatá po uložení dát, je uložená medzi upozorneniami a vyzerá nasledovne:



Obr. 7.1: Notifikácia na mobilnom zariadení

Aplikácia je rozdelená do niektorých častí a každá z týchto častí bola otestovaná na oboch zariadeniach. Rovnako bola vyvinutá aplikácia otestovaná aj vo firemnom prostredí. Pracovníci postupne prešli jednotlivé časti a operácie, ktoré im aplikácia ponúka. Pracovníci výroby testovali hlavne vytváranie kontrolných a chybových záznamov so skutočnými dátami. Vedúci pracovníci testovali úpravu a mazanie týchto záznamov, nahrávanie priorít, kde výrobné zákazky obsahovali reálne dáta z oddelenia technickej prípravy výroby. V ďalšej sekcii je podrobnejšie rozpísané testovanie užíva.

7.1 Testovanie aplikácie užívateľom

Po implementácii bola aplikácia predstavená pracovníkom výroby, resp. užívateľom aplikácie. Testovanie prebiehalo tak, že užívateľ vykonával jednotlivé prípady užitia, ktoré sú popísané v kapitole 4.

7.1.1 Testovanie aplikácie pracovníkom výroby

Prvým testovacím užívateľom bol pracovník výroby. V rámci prípadu užitia pridania nového chybového záznamu bolo odhalené, že číslo artikla pozostáva aj zo špeciálnych znakov. Táto chyba bola opravená správnym nastavením režimu v rámci nastavení pre skenovanie čiarového kódu. Po zhodnotení vedúcim pracovníkom výroby boli zmenené práva pre pracovníka výroby, ktorý môže chybový záznam po vložení aj upraviť, lebo je potrebné do záznamu doplniť aj dôležité informácie, ktoré boli buď zistené neskôr alebo ich zákazník pri reklamácii dodatočne poskytol.

Pri testovaní vykonávania a pridávania problémov k výrobným zákazkam boli práva pracovníka výroby rozšírene o úpravu evidovaných problémov, pretože pre vedúceho pracovníka výroby je dôležitá spätná väzba okolo riešeného problému.

Pre ostatné prípady užitia pracovníka výroby boli podmienky splnené a diskutovalo sa o budúcom vývoji aplikácie, ktorá by mu mohla poskytovať exportovanie záznamov do csv súborov.

7.1.2 Testovanie aplikácie vedúcim pracovníkom výroby

Následne aplikáciu testoval aj vedúci pracovník výroby. V rámci pridávania chybových záznamov bol pridaný aj jeho stav, ktorý môže meniť len vedúci pracovník výroby alebo admin. Záznam môže byť v stave otvorený, ktorý značí, že chyba ešte nebola vyriešená a uzavretý, ktorý naznačuje, že vedúci pracovník chybu skontroloval a je považovaná za ukončenú.

V rámci pridávania výrobných zákaziek bol pozmenený spôsob pridávania. Keďže táto strojárenská firma má vysoký objem výrobných zákaziek, nebolo by možné, aby ich vedúci pracovník výroby nahadzoval ručne. Po konzultácii bolo dohodnuté, že sa výrobné zákazky budú pridávať z tabuľkového súboru, kde sú tieto výrobné zákazky uložené a zmeny sa spracujú len ak boli zmenené údaje k danej výrobnej zákazke alebo výrobná zákazka ešte nebola uložená v databáze. Rovnako bolo dohodnuté, aby prehľadnosť údajov k výrobným zákazkám bola vyššia, boli ikony v návrhu zamenené za zobrazenie konkrétnych údajov.

V rámci práce s aplikáciou vedúci pracovník poznamenal, aby bolo stále jasne viditeľné, aký typ pracovníka je prihlásený a do hlavičky pri názve aplikácie bol pridaný aj typ prihláseného pracovníka. Pre ostatné prípady užitia vedúceho pracovníka výroby boli podmienky splnené a ďalším vývojom v budúcnosti by boli aj nastavenia štatistík, ktoré

by boli zhodnotené po časovo dlhšej dobe používania aplikácie a taktiež aj pridanie ďalších modulov aplikácie.

Kapitola 8

Záver

Cieľom tejto diplomovej práci bolo vytvoriť progresívnu webovú aplikáciu, ktorá bude napomáhať pri riadení výroby v strojárenskej firme a bude uľahčovať evidenciu dát pracovníkom výroby aj bez pripojenia na internet. V rámci zadania boli splnené všetky jeho požiadavky.

Na začiatku bol analyzovaný a stručne popísaný prehľad existujúcich riešení, ich výhody a nevýhody a na základe toho boli vyhodnotené požiadavky, aké má spĺňať aj vyvíjaný systém.

V rámci teoretického rozboru bola popísaná problematika funkčnosti PWA, jej výhody a nevýhody, stratégie akými môže pracovať, akým smerom sa bude vývoj PWA aplikácie napredovať, návrhový vzor a následne aj problematika dokumentácie a riadenia procesu výroby v strojárenskej firme.

V časti návrhu riešenia je popísaný UML diagram pre daný systém, ako bude vyzerať užívateľské rozhranie progresívnej webovej aplikácie a čo konkrétne bude obsahovať, akým spôsobom bude prebiehať prioritizácia výrobných zákaziek a ako sa budú riešiť vzniknuté problémy v rámci výrobných procesov daných zákaziek. Ďalej je popísaná časť, čo a ako budú pracovníci dokumentovať svoju prácu, prípadne vzniknuté chyby v rámci výrobného procesu danej zákazky, ako sa bude kontrolovať práca zamestnancov a ako si vedúci pracovník prezrie vyhodnotenú prácu svojich podriadených.

V rámci implementácie je popísaný postup akým spôsobom bola implementovaná serverova a klientska strana. Pod klienstkou stranou ako bolo implementované používateľské rozhranie a aké možnosti pre jednotlivé úrovne používateľov ponúka a aj implementácia rozšírení obyčajnej webovej aplikácie o vlastnosti PWA.

V časti testovania bola aplikácia odskúšaná na desktopovom aj mobilnom zariadení a rovnako si aplikáciu vyskúšali aj zamestnanci z strojárenskej firmy s reálnymi dátami.

V rámci ďalšieho rozšírenia by bolo možné pridať do webovej aplikácie aj ďalšie moduly, ktoré takáto firma využíva, ako napríklad evidencia kontroly lakovania, údržby strojov, kalibrovania meradiel alebo školení zamestnancov. Vývoj progresívnych webových aplikácií rýchlo napreduje, a tak by bolo možné pridať aj ich ďalšie rozšírenia podľa požiadaviek, ktoré by ešte viac uľahčili prácu v technickej príprave výroby.

Táto diplomová práca mi priniesla mnoho prospešných informácií o moderných webových aplikáciach a rozšírila moje vedomosti v tejto oblasti. Taktiež sa mi podarilo zlepšiť svoje schopnosti v jednotlivých fázach tohto veľkého projektu, ktorý prinesie úžitok v strojárenskom odvetví.

Literatúra

[1] Alligator.io. *Introduction to Progressive Web Apps (PWAs): Service Worker & Manifest* [online]. 2016. Dostupné z: https://www.digitalocean.com/community/tutorials/js-intro-progressive-web-apps.

[2] Bierman, Gavin and Abadi, Martín and Torgersen, Mads. Understanding TypeScript. In: Springer. *ECOOP 2014 – Object-Oriented Programming*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, sv. 8586, s. 257–281. Lecture Notes in Computer Science. ISBN 3662442019.

[3] Chidume Nnamdi. *5 Service Worker Caching Strategies for Your Next PWA App*. 2020. Dostupné z: https://blog.bitsrc.io/5-service-worker-caching-strategies-for-your-next-pwa-app-58539f156f52.

[4] Chimezie Enyinnaya. *Demystifying The Service Worker Lifecycle* [online]. DigitalOcean, 2019. Dostupné z: https://www.digitalocean.com/community/tutorials/demystifying-the-service-worker-lifecycle.

[5] Chris Love. *Progressive Web Application Development by Example: Develop fast, reliable, and engaging user experiences for the web*. 2. vyd. Packt Publishing, 2018. ISBN 9781787282346.

[6] Darlington Gospel. *React.js Architecture Pattern: Implementation + Best Practices*. 2023. Dostupné z: https://www.knowledgehut.com/blog/web-development/react-js-architecture.

[7] Flora. *Angular Vs Vue Vs React: What Is The Best PWA Framework For 2022?* 2022. Dostupné z: https://bsscommerce.com/blog/best-pwa-framework-angular-vue-react/.

[8] Hareem Mohsin. *An introduction to react programming language*. 2021. Dostupné z: https://invozone.com/blog/an-introduction-to-react-programming-language/.

[9] Hume, Dean Alan. *Progressive Web Apps*. 1. vyd. USA: Manning Publications Co., 2017. ISBN 1617294586.

[10] Kouraklis John. *MVVM as Design Pattern*. 1. vyd. Október 2016. ISBN 978-1-4842-2213-3.

[11] Mohit Joshi. *Angular vs React vs Vue: Core Differences*. 2022. Dostupné z: https://www.browserstack.com/guide/angular-vs-react-vs-vue.

[12] Nazhim Kalam. *Why You Should Use NestJS for Backend Development?* 2021. Dostupné z: https://enlear.academy/why-you-should-use-nestjs-as-your-backend-framework-bd1ff1acce5d.

[13] Neo Ighodaro. *Understanding Redux: A tutorial with examples.* 2022. Dostupné z: https://blog.logrocket.com/understanding-redux-tutorial-examples/.

[14] Oyetoke Tobi. *TypeORM: Object-relational mapping with Node.js.* 2022. Dostupné z: https://blog.logrocket.com/typeorm-object-relational-mapping-node-js/.

[15] Pete LePage, François Beaufort, Thomas Steiner. *Add a web app manifest* [online]. 2018. Dostupné z: https://web.dev/add-manifest/.

[16] Rick L. *Express. Js: Guide Book on Web Framework for Node. Js.* 1. vyd. CreateSpace Independent Publishing Platform, 2016. ISBN 9781533320018. Dostupné z: https://books.google.sk/books?id=gU7hjwEACAAJ.

[17] Royi Benita. *Clean Node.js Architecture —With NestJs and TypeScript.* 2022. Dostupné z: https://betterprogramming.pub/clean-node-js-architecture-with-nestjs-and-typescript-34b9398d790f.

[18] Sebastian Peyrott and AuthO. *The JWT Handbook.* 1. vyd. Auth0 Inc, 2016. ISBN Self-published. Dostupné z: https://assets.ctfassets.net/2ntc334xpx65/o5J4X472PQUI4ai6cAcqg/13a2611de03b2c8edbd09c3ca14ae86b/jwt-handbook-v0_14_1.pdf.

[19] Taha Sufiyan. *What is Node.js: A Comprehensive Guide.* 2023. Dostupné z: https://www.simplilearn.com/tutorials/nodejs-tutorial/what-is-nodejs.

[20] Tal Ater. *Building Progressive Web Apps: Bringing the Power of Native to the Browser.* 1. vyd. O'Reilly Media, 2017. ISBN 9781491961650.

[21] Uday Hiwarale. *How to use IndexedDB to build Progressive Web Apps.* 2018. Dostupné z: https://medium.com/jspoint/indexeddb-your-second-step-towards-progressive-web-apps-pwa-dcbcd6cc2076.

Príloha A

Príklad časti listu výrobnej zákazky

Datum : 08.12.2022 [13:21]
Datum : vtimko
(Kopie)

IMA Schelling Slovakia s.r.o.
Materiálový List

Strana : 1 / 2
Comp. : 100

Cislo artikla : 614.103.2023 — Vyrobna zakazka : 111098207

Nazov : MASCHINENT. 330 OF SCHW.

Roh

Selektionscd : DIS Dispomaschine

Sklad : 110200
Planovany dt. zaciatku : 22.12.2022
Planovany dt. ukoncenia : 25.01.2023
Objednane mnozstvo : 1 pcs

614.103.2023

111098207

Lagerort	Artikla	Nazov	Menge-Pro-Stück		Gesamtmenge
	614.103.2008	TISCHPLATTE 330 OF	0	0,00	0,0000 pcs
	614.103.2024	LUFTZULEITUNGSROHR LKT 330	0	0,00	1,0000 pcs
	614.104.2079	RIPPE OF LUFTKANAL	0	0,00	10,0000 pcs
WE-FERTIGU	614.103.2011	BRENNZUSCHN. INNEN 330 OF	0	0,00	0,0000 pcs
	614.103.2012	BRENNZUSCHN.AUSSEN 330 OF	0	0,00	0,0000 pcs
	614.103.2023-01	SADA VYP. PRE 614.103.2023	0	0,00	1,0000 pcs

GBO S.R.O
614.103.2008 ..... 1ks
614.103.2011 ..... 1ks
614.103.2012 ..... 1ks
Makro S355J2+N 5240/100/20 ....1ks
Makro S355J2+N 100/ 90/20 ....2ks
pieskovat

111022064 20 16.12.2022

Príloha B

Obsah priloženého pamäťového média

Priložené CD obsahuje `xgladi00_dp.zip` súbor, ktorý po rozbalení má nasledovnú štruktúru:

- `master-thesis-schelling/` - obsahuje zdrojové súbory progresívnej webovej aplikácie
  - `frontend/` - obsahuje zdrojové súbory klientskej strany aplikácie
  - `backend/` - obsahuje zdrojové súbory serverovej strany aplikácie
  - `db/` - obsahuje súbory pre inicializáciu databázy
- `pdf/` - obsahuje pdf súbor s diplomovou prácou
- `pdf_latex/` - obsahuje zdrojové súbory diplomovej práce napísanej v jazyku LaTeX