Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

# Webová aplikácia na správu cenníkov

Bakalárska práca

2021
Adam Gonšenica

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

# Webová aplikácia na správu cenníkov

Bakalárska práca

| | |
|---|---|
| Študijný program: | aplikovaná informatika |
| Študijný odbor: | Informatika |
| Školiace pracovisko: | Katedra aplikovanej informatiky |
| Školiteľ: | RNDr. Marek Nagy, PhD |
| Konzultant: | Ing. Peter Kello, PhD |

Bratislava, 2021
Adam Gonšenica


Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

# ZADANIE ZÁVEREČNEJ PRÁCE

**Meno a priezvisko študenta:** Adam Gonšenica
**Študijný program:** aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
**Študijný odbor:** informatika
**Typ záverečnej práce:** bakalárska
**Jazyk záverečnej práce:** slovenský
**Sekundárny jazyk:** anglický

**Názov:** Webová aplikácia na správu cenníkov
*Web application for the pricelist management*

**Anotácia:** Aplikácia spracúva ceny produktov, ktoré prichádzajú ako POST request v JSON formáte a zobrazuje ich v reálnom čase na strediskových obrazovkách. Spracovanie requestu nesmie spomaľovať systém a musí byť spracované do 100 ms. Ceny sa ukladajú do PostgrSQL databázy. Aby bola zaručená rýchlosť spracovania pri viacerých requestoch naraz, musí byť request spracovaný postupne. Po spracovaní requestu je nutné ceny premietnuť do obrazoviek pomocou WebSocket technológie.

**Cieľ:**
1. Vytvorenie vhodného návrhu a implementácia DB pre aplikáciu.
2. Vytvorenie vhodného mechanizmu na postupné spracovanie requestov a použitie dátových štruktúr pre čo najrýchlejšie spracovanie.
3. Nájdenie a zapracovanie technológie pre zobrazovanie nových cien v reálnom čase na viacerých miestach naraz.
4. Analýza a implementácia pre front-end aplikácie
5. Rozdelenie aplikácie podľa MVC alebo iného vhodného návrhového vzoru
6. Použite technológií Node.js a express.js pre API časť aplikácie.

**Poznámka:** Softvérové nástroje: Node.js, socket.io, express.js, react, vue, angular, pug,...
Aplikácia bude vyvíjaná pre potreby spoločnosti TMR, a.s.,
ktorej konzultantom bude Ing. Peter Kello, PhD.

**Vedúci:** RNDr. Marek Nagy, PhD.
**Katedra:** FMFI.KAI - Katedra aplikovanej informatiky
**Vedúci katedry:** prof. Ing. Igor Farkaš, Dr.

**Dátum zadania:** 28.09.2021

**Dátum schválenia:** 30.09.2021 doc. RNDr. Damas Gruska, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

**Poďakovanie:** Chcem sa poďakovať školiteľovi RNDr. Marekovi Nagyovi PhD. a konzultantovi Ing. Petrovi Kellovi, PhD za pomoc, cenné rady a čas, ktorý mi venovail počas tvorby mojej bakalárskej práce.

# Abstrakt

Slovenský abstrakt v rozsahu 100-500 slov, jeden odstavec. Abstrakt stručne sumarizuje výsledky práce. Mal by byť pochopiteľný pre bežného informatika. Nemal by teda využívať skratky, termíny alebo označenie zavedené v práci, okrem tých, ktoré sú všeobecne známe.

**Kľúčové slová:** jedno, druhé, tretie (prípadne štvrté, piate)

# Abstract

Abstract in the English language (translation of the abstract in the Slovak language).

**Keywords:**

# Obsah

# Zoznam obrázkov

# Zoznam tabuliek

# Úvod

Bakalársku prácu robíme pre firmu Tatry Moutain Resorts as. Firma Tatry Moutain Resorts as. sú prevádzkovateľom turistických rezortov ako napríklad horských stredísk, hotelov, aquaparkov, zábavných parkov a iných turistických služieb v regióne strednej a východnej Európy.

Lyžiarske strediská majú obmedzené kapacity, ich vyťaženie je počas týždňa nerovnomerné či už vplyvom, počasia, dňa v týždni, sviatkov či prázdnin. Preto spoločnosť Tatry Moutain Resorts potrebovala nástroj na manažovanie vyťažeností stredísk. Týmto nástrojom je dynamická cena lístka, respektíve flexibilné cenníky lístkov . Toto riešenie by malo zabezpečiť rovnomernejšie rozptýlenie lyžiarov v strediskách. Výpočet cien bude mať na starosti firma tretej strany a budú prichádzať ako POST request v Json formáte. Cenníky však môžu veľmi rýchlo nabrať na objeme dát, pretože spoločnosť Tatry Moutain Resorts prevádzkuje momentálne 9 stredísk a v budúcnosti budú určite ďalšie pribúdať. Každé stredisko má rôzne typy lístkov v celkovom počte okolo 6 no a v okrajových prípadoch, bude treba spracovať cenníky pre celú sezónu, čo je cca 350 dní. Keď sa to všetko vynásobí zistíme, že množstvo dát na spracovanie je objemné a bude treba zabezpečiť postupné spracovanie.

Úlohov aplikácie je to zabezpečiť čo najefektívnejšie, ceny spracovať a premietnuť do obrazoviek pomocou WebSocket technológie. Aplikácia by mala mať vlastných používateľov, ako aj administrátorov. Používateľské rozhranie na prihlásenie používateľov, nastavenie voľných dní, to sú tie v ktorých sa nelyžuje. Pre admina by malo obsahovať nástroj na manažment používateľov. Obrazovky na zobrazenie aktuálnych cenníkov v lyžiarskom stredisku a obrazovky na zobrazenie aktuálnych podmienok ako sú počasie, kamery alebo iné dôležité informácie.

Hlavná časť tejto bakalárskej práce je rozdelená na tri kapitoly. V prvej kapitole opisujeme použité technológie. V druhej kapitole opíšeme architektúru, databázový model, algoritmické riešenie efektívneho spracovania cien. Tretia kapitola opisuje proces implementácie, testovania.

# Kapitola 1

# Východiská

Na efektívne riešenie zadania tejto bakalárskej práce je potreba rôzne technológie. V tejto kapitole si opíšeme a vysvetlíme použité technológie, odôvodníme si konkrétny výber, prípadne porovnáme s alternatívnymi možnostiam. V závere východiskovej kapitoly je opísaný aktuálny stav a existujúce podobné systémy.

## 1.1 Serverový jazyk

Webové aplikácie rastú na popularite, pretože sú ľahšie na naprogramovanie udržiavanie a zabezpečenie. Taktiež sú ľahko prístupné pomocou webového prehliadača, ktorý nájdeme na väčšine zariadení s prístupom na internet. Takže nepotrebujú nič inštalovať a ani vyvíjať rôzne verzie pre rôzne operačné systémy.

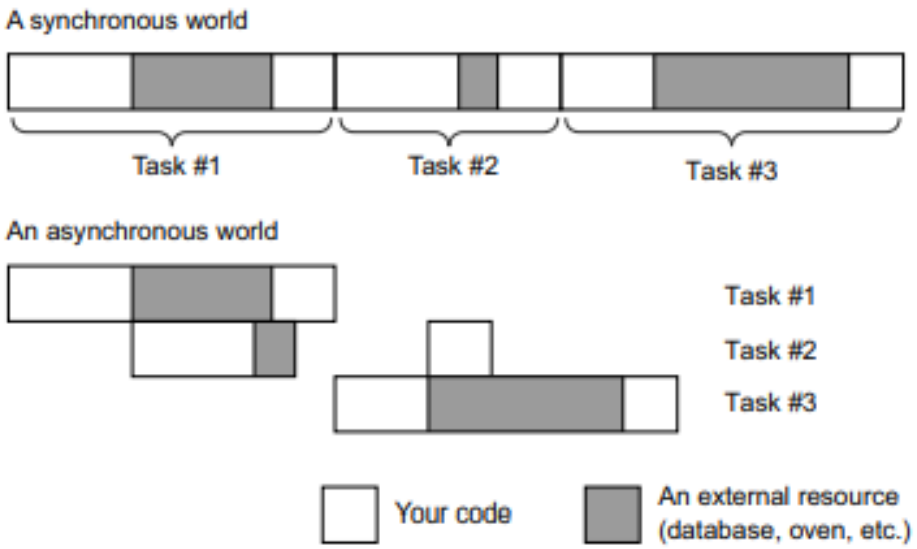
Medzi najdôležitejšie podmienky výberu serverového jazyka je schopnosť spracovať viacero používateľov efektívne, taktiež výkon a v neposlednom rade samotná ľahkosť implementácie. Rozhodovali sme sa medzi jazykmi PHP, Python a Node.js, čo serverová implementácia JavaScriptu.

### 1.1.1 Node.js

Javascript nemusí operovať len vo webovom prehliadači, môžeme ho využiť aj ako každý serverový jazyk. Najrozšírenejšou takouto implementáciou je Node.js. Sekcia je spracovaná podla knihy [10]. Spĺňa teda Node.js naše podmienky na serverový jazyk?

Ako sme už vyššie spomenuli Node.js je naprogramovaný v Javascripte. Je postavený na Google Chrome V8 JavaScript engine, ktorý je všeobecne známy svojim výkonom. [12]

Ďalšou výhodou Node.js je jeho schopnosť spracovať viacero udalostí naraz, pochádzajúca z asynchrónnosti Javascriptu. Javascript využíva takzvaný event loop teda cyklus udalostí, v praxi to znamená, že keď si prehliadač vyžiada request od nášho servera a počas toho nám príde ďalší request, pri čom oba requesty potrebujú nejaké




Obr. 1.1: Porovnanie asynchrónneho kódu (napr Node.js) s synchrónnmy kódom. Aasynchrónny kód sa môže dokončiť oveľa rýchlejšie, hoci ho nikdy nevykonávame paralelne.

dáta z databázy, kým prvý request získava dáta, druhý sa môže začať tiež vybavovať. Možme to vidieť graficky znázorňené na obrázku 1.1. Takéto možnosti nemajú iné riešenia defaultne implementované, častokrát na spracovanie viacero requestov naraz je potrebné nakúpiť dodatočný hardware.

V neposlednom rade výhodou Node.js je framework Express.js, ktorý vie veľmi efektívne zjednodušiť programovanie backendu webovej aplikácie.

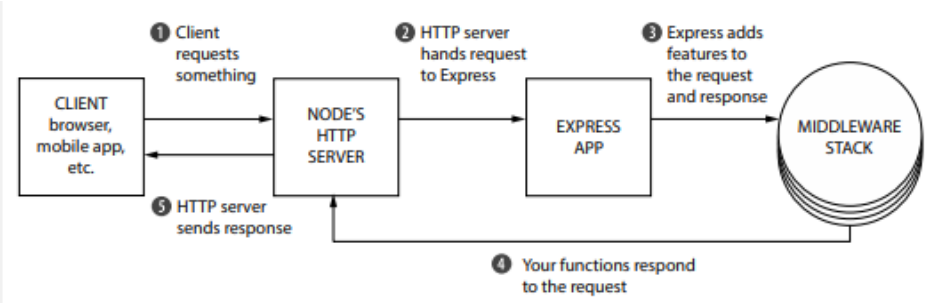
Node.js teda spĺňa všetky nami stanovené kritéria, preto je vhodný kandidát na využitie v tomto projekte.

## 1.2 Frameworky a moduly

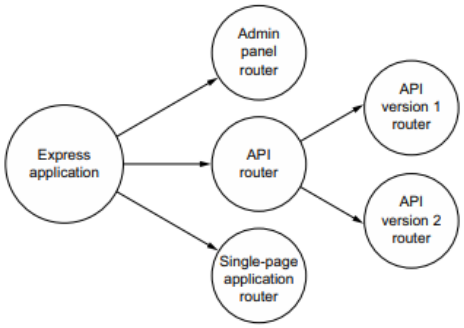
Node.js má k dispozícií veľké množstvo frameworkov a modulov tretích strán, ktoré sú často krát open source, s veľkou základňou aktívnych vývojárov. Taktiež sú vďaka častému využitiu dobre odladené a otestované. V tejto časti si opíšeme tie, ktoré sú vhodné na realizáciu našej aplikácie.

### 1.2.1 Express.js

Express je relatívne malý framework implementovaný v Node.js. Ako už má v názve pomáha s budovaním aplikácii v expresnom čase. Sekcia je spracovaná podla knihy [10]. Express.js pri budovaní API pridáva užitočnú funkcionalitu. Uľahčuje prácu na viacerých miestach a to hlavne organizáciu funkcionality aplikácii použitím takzvaných middleware funkcii, správu ciest API endpointov, takzvaný routing, pridáva uži-




Obr. 1.2: Spracovanie requestu pomocou middleware funkcii.




Obr. 1.3: Diagram ukazujúci príklad routingu vo velkej aplikácii

točné funkcie na prácu s http protokolom a taktiež uľahčuje renderovanie dynamického HTML. Ďalej bližšie špecifikujem túto funkcionalitu:

- Middleware – Node.js nám na spracovanie requestú dáva funkciu s ktorou môžeme ďalej pracovať. Request prichádza do tejto funkcie a response vraciame z tejto funkcie. Middleware je pojem, pod ktorým sa myslí to, že namiesto jednej velkej request spracovávajúcej funkcie, máme viacero menších, z ktorých každá vykoná svoju menšiu časť roboty. Vzniká tak akási chain of responzibilities. Znázornené to možme vidieť na obrázku 1.2

- Routing – Pri spracovaní requestov musíme brať do úvahy konkrétnu URL a http metódu na ktorú nám bol request poslaný. Routing v Express nám pomáha jednoducho nasmerovať URL cesty a http metódy k správnym funkciám. Na správne nasmerovanie používame takzvané routery po anglicky routers. Pri velkých aplikáciách je možné na seba nasmerovať viacero routerov. Vzniká tak prehľadná stromová štruktúra ako je vidno na obrázku 1.3.

Express samozrejme nefunguje samostatne. Súčasťou Node.js ekosystému sú rôzne ďalšie užitočné moduly tretej strany, ktoré s ním spolupracujú.

### 1.2.2 EJS

EJS, teda embedded JavaScript je jeden z najjednoduchších a najpopulárnejších zobrazovacích enginov. Dovoľuje nám generovať dynamicky vyplnené HTML pomocou Javascriptu. Má jednoduchú syntax pomocou EJS tagov ako napríklad <% %> pre vykonanie Javascriptu vo vnutry tagu, alebo <%= %> pre vloženie hodnoty v tagu do html. Taktiež podporuje vkladanie ďalších ejs súborov. [5]

Funkcionalita je znázornená na príklade 1.1. Kde sa dynamicky vyplní meno a vloží súbor bio.ejs.

```
1 var obj = {
2 name:"Adam"
3 }
4 <\% if(obj.name){ \%>
5   <h1> <\%= obj.name \%> </h1>
6 <\% } \%>  >
7 <\%- include("bio" ) \%>
```

Kód 1.1: Príklad EJS

### 1.2.3 Nodemailer

Nodemailer je modul pre Node.js aplikácie, ktorý umožňuje jednoduché posielanie mailov. [7]

### 1.2.4 Axios

Axios je modul pre node.js, krorý umožňuje jednoduché http requesty. [1]

### 1.2.5 Bcryptjs

Bcryptjs je modul ktorý umožňuje zašifrovať dáta napríklad heslá. Používa šifrovaciu metódu bcrypt s použitím takzvanej soli, po anglicky salt. Je velmi odolný aj proti brute force útokom. [2]

### 1.2.6 Socket.io

Socket.io je modul, ktorý dovoluje obojstrannu komunikáciu medzi serverom a prehliadačom v realnom čase. [9]

### 1.2.7 Compression

Compression je modul ktorý dokáže zmenšiť velkosť tiel responsov. Používa sa ako middleware funkcia, a s jeho využitím vieme zmenšiť objem dát a zvýšiť rýchlosť spra-

covania requestov. [3]

## 1.3 Web server

Server je počítač komunikujúci s ostatnými počítačmi, ktorým odosiela požadované informácie. Tieto počítače sa nazývajú klientami. Web server je server pripojený na internet, ktorý pomocou http protokolu príjma requesty od klientov, ako napríklad internetový prehliadač, a odosiela http odpovede ako napríklad HTML stránku alebo dáta v JSON formáte ako napríklad pri volaní API. Sekcia je spracovaná podla sránky [6].

Hoci Node.js aplikácia s využitím Express.js frameworku vie fungovať ako samostatný web server, s nasadením webovej aplikácie do produkcie sa oplatí mať osobitný web server na ktorom našu Node.js webovú aplikáciu spustíme. V súčasnosti najpoužívanejšími riešeniami pre web servery sú Apache a Nginx. Z toho druhý menovaný je síce menej používaný, ale plynulo vytláča konkurenta a rastie na podiely na trhu.

### 1.3.1 Nginx

Nginx je http a proxy server ktorý sa využíva na viacero úloh a dokáže zvýšiť výkonnosť Node.js aplikácie a to s využitím na:

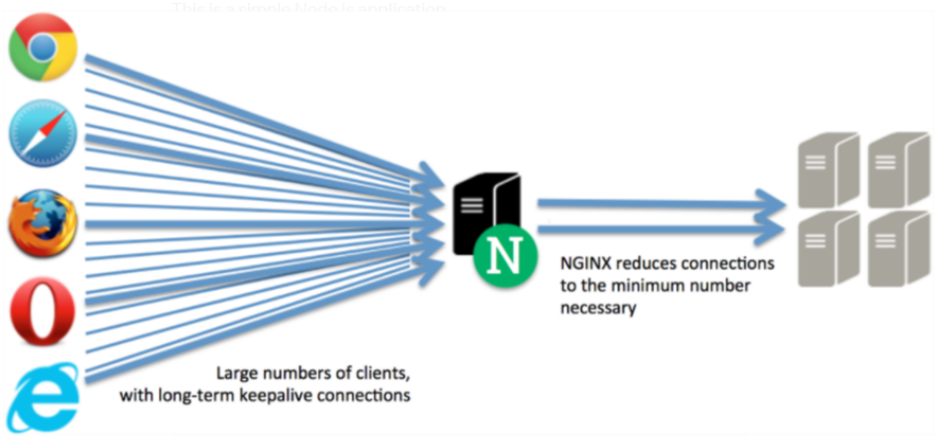
- Reverzný proxy server – Keď sa návštevnosť webovej aplikácie zvyšuje, najlepším prístupom na zlepšenie výkonu je použitie NginX ako reverzného proxy servera pred serverom Node.js na vyrovnávanie prenosu medzi servermi, ako možme vidieť na obrázku 1.4. Toto je hlavný prípad použitia NginX v aplikáciách Node.js.

- Vyvažovanie záťaže – zlepšuje výkon a zároveň znižuje záťaž na backendové služby odosielaním požiadaviek klientov, ktoré má splniť ktorýkoľvek server s prístupom k požadovanému súboru.

- Ukladanie statického obsahu do vyrovnávacej pamäte – Poskytovanie statického obsahu v aplikácii Node.js a používanie NginX ako reverzného proxy servera zdvojnásobuje výkon aplikácie.

## 1.4 Databáza

Každá ozajstná aplikácia potrebuje spôsob na uloženie perzistentných údajov do pevnej pamäte a následne, keď ich treba, čo najefektívnejšie načítanie späť do operačnej pamäte. Za týmto účelom existujú databázové systémy. Sekcia je spracovaná podla knihy




Obr. 1.4: Využitie Ngnix web servera na zvýšenie výkonnosti Node.js aplikácie

[10]. Na výber máme medzi relačnými databázami takzvané relational databases, alebo nerelačnými databázami takzvanými non-relational databases:

- Relačné databázy sa podobajú tabuľkám. Dáta sú štruktúrované a každý záznam je v podtate riadok tabulky. Väčšina relačných databáz sa dopytujú nejakov formou SQL jazyka čo je skratka pre structured query language, teda štruktúrovaný dopytovací jazyk. V Node.js je najrozsiahlejší a najpoužívanejší modul na prácu s relačnými databázami Sequelize.

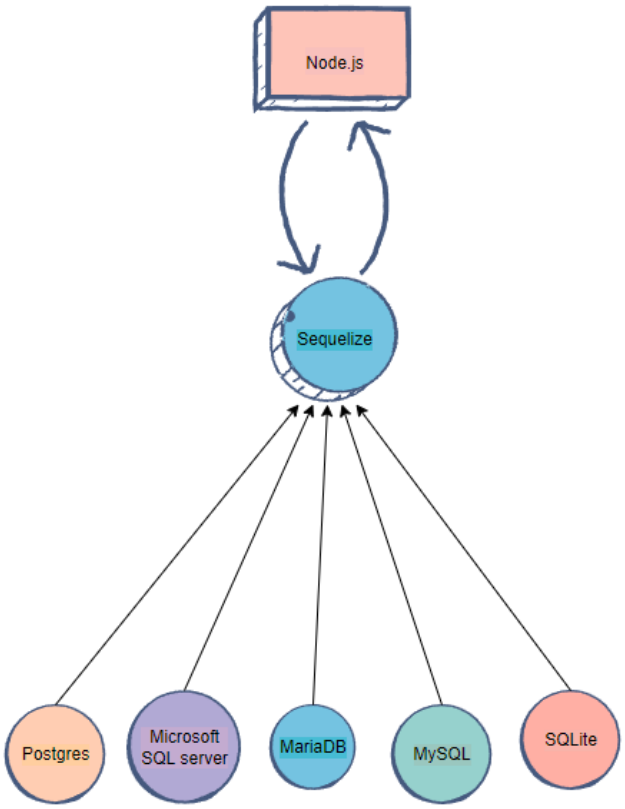
- Nerelačné databázy sú odlišné od relačných, nie sú štruktúrované ako tabuľky. Sú menej robustné, nemajú stĺpce ani tabuľky. Majú kolekcie, čo je niečo ako súbor dokumentov respektíve zoznam. Záznamy sú potom dokumenty v kolekcii. Dokumenty nemajú nejakú špecifickú štruktúru ako napríklad stĺpce v tabuľke. V Node.js je najrozsiahlejší a najpoužívanejší modul na prácu s relačnými databázami Mongoose.

Z týchto možností sme pre našu aplikáciu vybrali Relačnú databázu, dôvodu našich predchádzajúcich skúsenosti s SQL jazykmi najme s PostgreSQL, preto ako SQL jazyk sme si vybrali pravé tento.

### 1.4.1 Sequelize

Sequelize je ORM(Object–relational mapping) pre Node.js umožňujúci komunikáciu s viacerými SQL jazykmi, ako možme vidieť na obrázku 1.5. Podporuje transakcie, relácie, migrácie, lazy loading aj eager loading. [8]

Modelovanie databázy pomocou Sequelize je jednoduché, model v sequelize reprezentuje vlastne tabuľku v databáze. Pri definovaní modelu môžeme zadať rôzne para-




Obr. 1.5: Komunikácia ORM Sequelize s podporovanými SQL jazykmi

metre ako typ, dovolenie null hodnoty, primárny kľuč, unikátnosť hodnoty. Samotný model následne za nás vygeneruje samotné SQL.

Podobne aj pri dopytoch do databázy stačí z referencie na model zavolať jednu z funkcii ako napríklad create, find, findAll, update. Taktiež môžeme pridať parametre ako atribúty, agregácie, zgrupovanie, where klauzulu a podobne. Následne modul za nás vygeneruje SQL a vykoná dopyt do databázy.

## 1.5 Dynamické programovanie

Na spracovanie veľkého množstva cien, potrebujeme optimálne riešenie, použijeme teda metódu dynamického programovania:

Kľúčom k zníženiu množstva práce, ktorú robíme, je zapamätať si niektoré z minulých výsledkov, aby sme sa vyhli prepočítavaniu výsledkov, ktoré už poznáme. Jednoduchým riešením je uložiť výsledky do tabuľky, keď ich nájdeme. Potom predtým, ako vypočítame nové výsledky, najprv skontrolujeme tabuľku, či už nie je známy výsledok. Ak už v tabuľke existuje výsledok, použijeme namiesto prepočítania hodnotu z tabuľky. [11]

## 1.6 Dátové štruktúry

Pri príchode cenníkov ich potrebujeme spracovať postupne podľa priority. Na toto spracovanie potrebujeme implementáciu prioritného frontu. Sekcia je spracovaná podla knihy [11].

### 1.6.1 Binárna halda

Binárna halda je implementácie prioritného frontu, ktorá využíva dátovú štruktúru halda. Halda je binárny strom, ktorý musí spĺňať tieto dve podmienky:

- každý vrchol (okrem koreňa) má hodnotu kľúča väčšiu alebo rovnú ako hodnota kľúča v jeho predkovi, takže v koreni je minimálna hodnota kľúča z celého stromu, teda prvok s najväčšou prioritou.
- je to skoro úplný binárny strom, a to preto, že okrem najnižšej úrovne sú všetky úrovne úplné a v najnižšej úrovni sú všetky vrcholy umiestňované len zľava

Pri spracovaní cenníkov podla priority budeme potrebovať tieto metódy:

- Pridanie prvku - prvok sa pridá na koniec haldy. Následne sa povymieňa v strome smerom nahor a to až kým nenatrafi na menšieho predka alebo sa dostane až do koreňa stromu.
- Odobranie prvku s najväčšou prioritou - to je prvok v koreni haldy. Následne sa koreň nahradí posledným prvkom v halde a povymieňa sa v strome smerom nadol a to až kým vrchol už nemá ani jedného syna, alebo obaja synovia už nemajú menšiu hodnotu ako prvok.

### 1.6.2 Modul Priorityqueue

Modul priorityqueue implementuje binárnu haldu v JavaScripte ako BinaryHeap(comparator) - comparator definuje porovnanie podľa ktorého sa prvky v binárnej halde budú usporadúvať. Modul má implementované obe potrebné metódy a to push() na pridanie prvku a pop() odobranie prvku s najväčšou prioritou

## 1.7 Aktuálny stav

Aktuálne sa v lyžiarskych strediskách spoločnosti Tatry Moutain Resort používaju pevné cenniky. Sezóna je väčšinov rozdelena na tri časti a v nich je pevne nastavena cena listkov v zavislosti na ocakavanu vyťaženosť strediska.

## 1.8 Exitujúce podobné systémy

Dynamická tvorba cien, je cenová stratégia, pri ktorej podniky stanovujú flexibilné ceny produktov alebo služieb na základe aktuálnych požiadaviek trhu. Podniky sú schopné meniť ceny na základe algoritmov, ktoré zohľadňujú ceny konkurentov, ponuku a dopyt a ďalšie vonkajšie faktory na trhu. Dynamická tvorba cien je bežnou praxou vo viacerých odvetviach, ako je pohostinstvo, cestovný ruch, zábava, maloobchod, elektrina a verejná doprava. Každé odvetvie pristupuje k dynamickej tvorbe cien trochu inak na základe svojich individuálnych potrieb a dopytu po produkte. Tu je niekoľko spoločností, ktoré majú podobný systém implementovaný: [4]

- Ticketmaster - Dynamické ceny za koncerty a iné podujatia sú často založené na dopyte trhu a je to stratégia, ktorú veľké spoločnosti ako Ticketmaster používajú už roky.

- Tímy NFL - Keďže 16 tímov NFL prešlo z pevných cien, je zrejmé, že dynamické stanovovanie cien v športe je trendom v celom odvetví. Dnes sa univerzitné a profesionálne športové tímy spoliehajú na rôzne modely pri obsadzovaní miest v závislosti od počasia, dátumu alebo počtu zľavnených miest. Niektoré tímy tiež pracujú na maximalizácii výnosov znížením počtu zľavnených a bezplatných vstupeniek. Toto rozhodnutie im pomohlo pomaly zvyšovať základňu držiteľov sezónnych vstupeniek a prilákať verných fanúšikov, ktorí sú ochotní zaplatiť viac.

# Záver

Na záver už len odporúčania k samotnej kapitole Záver v bakalárskej práci podľa smernice [**?**]: „V závere je potrebné v stručnosti zhrnúť dosiahnuté výsledky vo vzťahu k stanoveným cieľom. Rozsah záveru je minimálne dve strany. Záver ako kapitola sa nečísluje.“

Všimnite si správne písanie slovenských úvodzoviek okolo predchádzajúceho citátu, ktoré sme dosiahli príkazmi `\glqq` a `\grqq`.

V informatických prácach niekedy býva záver kratší ako dve strany, ale stále by to mal byť rozumne dlhý text, v rozsahu aspoň jednej strany. Okrem dosiahnutých cieľov sa zvyknú rozoberať aj otvorené problémy a námety na ďalšiu prácu v oblasti.

Abstrakt, úvod a záver práce obsahujú podobné informácie. Abstrakt je kratší text, ktorý má pomôcť čitateľovi sa rozhodnúť, či vôbec prácu chce čítať. Úvod má umožniť zorientovať sa v práci skôr než ju začne čítať a záver sumarizuje najdôležitejšie veci po tom, ako prácu prečítal, môže sa teda viac zamerať na detaily a využívať pojmy zavedené v práci.

# Literatúra


[1] Axios. [Citované 2022-1-15] Dostupné z https://www.npmjs.com/package/axios.

[2] Bcryptjs. [Citované 2022-1-15] Dostupné z https://www.npmjs.com/package/bcryptjs.

[3] Compression. [Citované 2022-1-15] Dostupné z https://www.npmjs.com/package/compression.

[4] Dynamicpricing. [Citované 2022-1-15] Dostupné z https://softjourn.com/insights/dynamic-pricing-in-ticketing.

[5] Ejs. [Citované 2022-1-15] Dostupné z https://www.npmjs.com/package/ejs.

[6] Nginx. [Citované 2022-1-15] Dostupné z https://blog.logrocket.com/how-to-run-a-node-js-server-with-nginx/.

[7] Nodemailer. [Citované 2022-1-15] Dostupné z https://www.npmjs.com/package/nodemailer.

[8] Sequelize. [Citované 2022-1-15] Dostupné z https://sequelize.org/v7/.

[9] Socket.io. [Citované 2022-1-15] Dostupné z https://www.npmjs.com/package/socket.io.

[10] Evan Hahn. *Express in Action: Writing, building, and testing Node. js applications.* Simon and Schuster, 2016.

[11] Bradley N Miller and David L Ranum. *Problem solving with algorithms and data structures using python Second Edition.* Franklin, Beedle & Associates Inc., 2011.

[12] Hezbullah Shah. Node. js challenges in implementation. *Global Journal of Computer Science and Technology*, 2017.

# Príloha A: obsah elektronickej prílohy

V elektronickej prílohe priloženej k práci sa nachádza zdrojový kód programu a súbory s výsledkami experimentov. Zdrojový kód je zverejnený aj na stránke `http://mojadresa.com/`.

Ak uznáte za vhodné, môžete tu aj podrobnejšie rozpísať obsah tejto prílohy, prípadne poskytnúť návod na inštaláciu programu. Alternatívou je tieto informácie zahrnúť do samotnej prílohy, alebo ich uviesť na obidvoch miestach.

# Príloha B: Používateľská príručka

V tejto prílohe uvádzame používateľskú príručku k nášmu softvéru. Tu by ďalej pokračoval text príručky. V práci nie je potrebné uvádzať používateľskú príručku, pokiaľ je používanie softvéru intuitívne alebo ak výsledkom práce nie je ucelený softvér určený pre používateľov.

V prílohách môžete uviesť aj ďalšie materiály, ktoré by mohli pôsobiť rušivo v hlavnom texte, ako napríklad rozsiahle tabuľky a podobne. Materiály, ktoré sú príliš dlhé na ich tlač, odovzdajte len v electronickej prílohe.