

Gymnázium Christiana Dopplera, Zborovská 45, Praha 5

ROČNÍKOVÁ PRÁCE

Hodnocení jídel v aplikaci jídelních lístků

Vypracoval: Petr Adámek

Třída: 4.C

Školní rok: 2018/2019

Seminář: Seminář z programování

Prohlašuji, že jsem svou ročníkovou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce.

V Praze dne 22.2.2019

Petr Adámek

Obsah

1	Úvod	3
2	Zadání	4
3	Analýza problému	5
3.1	Databáze	5
3.2	Update databáze	6
3.3	Hodnocení na webu	7
4	Technické specifikace	8
4.1	Přechod databáze	8
4.1.1	Zjednodušení textu	8
4.1.2	Propojení oblíbenosti se starou databází	9
4.1.3	Propojení importovaných jídel	9
4.2	Vypisování Webu	10
4.2.1	Průměr	10
4.2.2	Hodnotící tlačítka	11
4.3	Hodnocení	12
4.3.1	Script	12
4.3.2	PHP	12
5	Závěr	14
	Literatura	15
	Přílohy	16

1. Úvod

V této práci rozebereme způsob implementace hodnocení jídel do jídelních lístků webu www.jidelna.cz. Jde o stránky, které zajišťují objednávání jídel přes internet různým jídelnám. Zobrazují jídelní lístky a jiné základní informace o jídelnách. K jednomu účtu může být připojeno více strážníků i z různých jídelen. Kromě objednávání stránky také ukazují, která jídla si člověk vyzvedl a která ne.

Celá práce je vytvořena za spolupráce s firmou Společnost BARDA SW, HW s.r.o., která mimo jiné spravuje tyto stránky. Více informací o nich se můžete dozvědět na stránkách www.barda.cz.

Budeme popisovat přesný způsob implementace a důvody, proč byl zvolen právě tento způsob. Podíváme na výhody a nevýhody, které s sebou přinese. A v neposlední řadě probereme alternativy a důvod, proč byly zavrženy.

2. Zadání







Naším úkolem je vytvořit na webu jidelna.cz čtyři smajlíky na zadané místo (viz obrázek 2.1), jež po stisknutí mají jídlo ohodnotit hodnotou 1-4. Také máme vytvořit smajlíka, který odpovídá průměrnému hodnocení a při přjetí myší přes něj zobrazuje box s podrobným výpočtem průměru (viz obrázek 2.1). Samotné hodnocení jídla bude navázáno na zjednodušený název pokrmu tak, abychom i přes drobné chyby (velká a malá písmena, chybné čárky a háčky, či dvě mezery za sebou ...) byli schopni názvy požadovat za identické. Jednoduše řečeno, nechceme mít dvě hodnocení pro „kaše“ a „Kase“.

Odhadovaným průměrem do budoucnosti bude průměr hodnocení částí jednoho chodu. Což znamená průměr průměrů polévky, hlavního chodu, nápoje a dalších. Ideálně bychom chtěli tento počítat s váženým průměrem, protože dává smysl, aby hlavní jídlo mělo na oblíbenost dne větší vliv než nápoj. Dokonce se tím vyřeší i problém toho, když nechceme, aby se hodnocení dne odvíjelo od hodnocení nápoje.

Při zobrazení historie či dnešního dne jídelního lístku si přejeme ukázat pouze hodnocení tohoto dne bez ohledu na jídla v něm. Odhad oblíbenosti jídla v minulosti by byl pro kohokoliv zbytečnou informací.

Chceme umožnit strážníkům své hodnocení změnit, tudíž musíme nějakým způsobem na stránce zvýraznit, jak již hodnotili. Pokud si z jednoho účtu objednává více strážníků, každý hodnotí individuálně.

Nechceme při hodnocení napovídat. Tudíž průměrné hodnocení zobrazíme až po ohodnocení jídla, popřípadě, když nemůže žádný ze strážníků hlasovat. Hodnocení starší než 10 dní také nebereme v úvahu – vůbec nenecháme lidi hodnotit jídla starší než 10 dní. Vypovídající hodnota takového hodnocení by byla velmi nízká. V minulosti smí člověk hodnotit jen to, co si objednal a vyzvedl. Povolujeme hodnocení pro všechna jídla, která už se vydávají, ale ještě nám nepřišla informace o jejich vyzvednutí. Měli bychom, až dostaneme informace o odebrání jídel z jídelny, odstranit hodnocení, která byla vytvořena lidmi, kteří si jídlo neodebrali.

Oběd					
1 	Polévka	Polévka z vaječné jíšky	1, 3, 9	 700 g    	
	Jídlo	Zapečená ryba se sýrem	4, 7		
	Příloha	opečené brambory	7		
	Doplňek	Okurkový salát			
	Nápoj	voda			

Obrázek 2.1: Náhled hodnocení

3. Analýza problému

Nejprve zmíníme všechny technologie zde použité. Databáze v projektu užitá je MariaDB, jde o SQL databázi, která se chová prakticky stejně jako MySQL. PHP používá Nette Framework a jeho rozšíření Latte, které usnadňuje výpis na web a je přehlednější než samotné PHP. Je nutné podotknout, že dokumentace tohoto frameworku není úplná a pochopit co která funkce a který objekt dělá zabere nezkušenému poměrně dlouho. Naštěstí mají na internetu alespoň zdrojové kódy, které u komplikovanějších věcí, bez popisu funkce nestačí. Javascript je rozšířen o jQuery.

3.1 Databáze

Oblíbenost jídla bude udržována v tabulce oblíbenosti na obrázku 3.1, která se chová jako unikátní identifikátor jídla v jedné jídelně. Obsahuje:

- ID Zařízení, kde je pokrm vydáván.
- Celý název pokrmu a jeho ořezanou verzi. Původní název udržujeme pro případnou zpětnou kompatibilitu v případě přechodu na jiný způsob ořezávání textu.
- Celkový počet hodnocení, sumu hodnocení

Oblíbenost dne uložíme do tabulky která bude obsahovat jen sumu a počet hodnocení. Jako její ID zvolím ID řádky v tabulce, která funguje jako unikátní identifikátor data a času, kdy se toto jídlo vydávalo. Tuto tabulku hodnocení budu vytvářet při prvním hodnocení tohoto času.

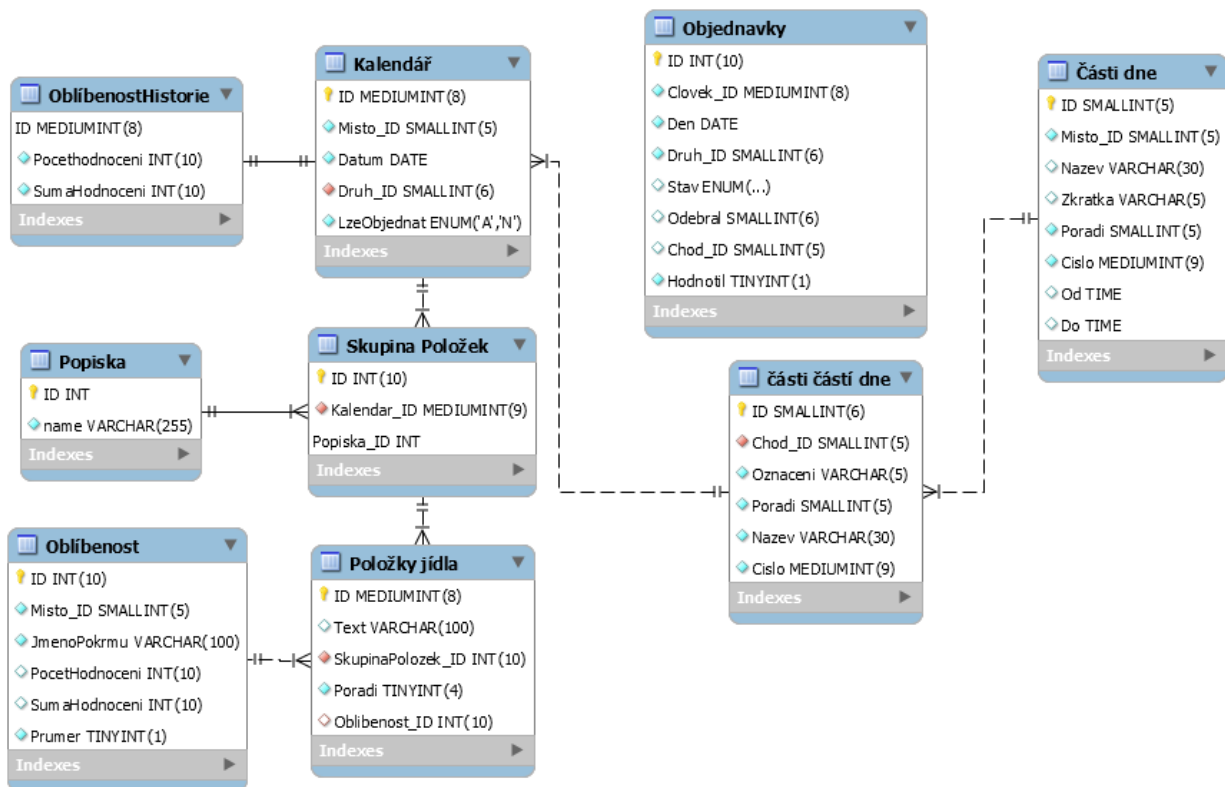
Do každé objednávky bude nutné přidat hodnotu, kterou člověk hodnotil jídlo. Základní hodnota je 0. Při ohodnocení se podíváme na tuto hodnotu a upravíme ji. Pokud byla nulová, navýšíme počet hodnocení, jinak se ho nedotýkáme. Sumu hodnocení změníme jen o rozdíl nové a staré hodnoty.

V případě, že se rozhodneme, že chceme brát v potaz jen hodnocení z nějakého období, dá se z této tabulky objednávek získat.

Pro rychlé hledání hodnocení jídla si udržujeme ID řádku v tabulce oblíbenosti u každého názvu jídla. Příliš mnoho místa nezabere a je mnohem rychlejší než hledat podle stringu a zařízení.

Ořezávání textu vymaže vše kromě písmen a ta obere o diakritiku a budou všechna udržována malá. Speciální znaky, čísla a mezery jsou na začátku a konci odebrány úplně, uvnitř nahrazeny jednou mezerou, aby byla slova stejným způsobem oddělena. Kdybychom tuto část odebrali úplně, mohlo by dojít k nesprávnému propojení slov.

Vhodné místo pro uložení váhy hodnocení jsme nenalezli.



Obrázek 3.1: Schéma Databáze

3.2 Update databáze

Při přechodu na tuto databázovou strukturu budeme muset kromě přidání nových tabulek a sloupců vyplnit ke každému názvu jídla správné ID řádku hodnocení. Protože ořezáváme text, provedeme update databáze z části v PHP. MySQL nepodporuje manipulaci s řetězci dostatečně pro ořezání textu. MariaDB umožňuje použití regulárních výrazů, ale vzhledem k tomu, že jde o jednorázovou operaci, bude jednodušší ořezání vyřešit v PHP.

Při importu nových jídel z jídelen uděláme úplně to samé. Vytvořit nový řádek pokud neexistuje anebo propojit s existujícím.

Hodnocení dní předem nevytváříme. Na to máme v databázi příliš mnoho dní, které určitě nikdo nebude moci ohodnotit a vytvoření ani nalezení zde netrvá ani zdaleka tak dlouho.

3.3 Hodnocení na webu

Přenášení hodnocení z webu do databáze a zpět bude zařízeno pomocí kombinace JavaScriptu a PHP. Při každém ohodnocení budeme muset poslat zprávu na server. Abychom neustále neobnovovali stránku, vyřešíme to přes tzv. AJAX – Asynchronous JavaScript And XML.

Bylo by pěkné, kdybychom zařídili podporu browseru s vypnutým JavaScriptem. K tomu bychom ale museli zjistit, zda má uživatel vypnutý JavaScript a v tom případě nahradit všechny Ajax odkazy přímými odkazy. Pokud jsme to schopni zjistit, konečná změna výpisu nebude ani zdaleka tak složitá. Vložíme celý obrázek do odkazu a data přidáme na konec odkazu namísto uložení do HTML.

Už při vytváření stránky si do HTML uložíme data, která poté budeme odesílat. Jinak nelze zjistit, které jídlo hodnotíme. Chceme odesílat ID objednávky, pod kterou ukládáme hodnocení. Vše ostatní si z této informace dokážeme zjistit. Zpětně potřebujeme získat hodnocení tohoto dne. Při ohodnocení musíme ověřit, že data, která dostáváme od klienta, odpovídají pravdě – nehlasuje za jiného, hodnocení 1-4, tento den opravdu hodnotit mohl. Nelze kontrolovat jen pomocí dat, která sami odešleme do HTML, protože s těmi může uživatel dělat co chce.

4. Technické specifikace

4.1 Přechod databáze

Prvním krokem je vytvoření objektu, který bude obsahovat podstatné funkce pro výpis a hodnocení jídla. Funkce raději ukládáme do objektu než volně do nového namespace, protože už teď vím, že potřebujeme přístup k databázi a usnadní to jakékoliv ukládání proměnných. Navíc se tím zachová konzistence v rámci projektu.

4.1.1 Zjednodušení textu

Potřebujeme funkci, která převede název jídla na jeho ořezanou verzi.

Problém lze vyřešit přes dvě funkce regulárních výrazů. První změní všechna písmena na malá bez diakritiky a změní vše, co není písmenem, na mezeru. Druhá zkrátí všechny řetězce mezer na jednu mezeru. Poté stačí jen zavolat funkci, která odebere mezery na začátku a konci výrazu. V případě, že po odebrání všech mezer nic nezbude, vrátíme původní text. K tomu by nikdy nemělo dojít.

Druhá možnost je chovat se k řetězci jako k poli znaků a projít ho písmeno po písmeni. Protože jde o kódování UTF-8, musíme string specificky konvertovat na pole, jinak se rozdělí po jednom bajtu a charaktery skončí „roztrhané“. Poté procházíme znak po znaku jako index pole. Vytvoříme funkci, která je jen jeden velký switch–case, který vrací buď zmenšené a ořezané písmeno či null. Odstraníme vše, co není písmenem, až po první písmeno. Následně si zapisujeme písmena, která se nám vrátí ořezaná. Když nalezneme něco, co není písmenem, pamatujeme si, že budeme chtít zapsat mezeru. Poté procházíme pole dál a před první písmeno zapíšeme mezeru právě pokud máme zapsáno, že se vrátilo null. Okamžitě musíme změnit stav detekce mezery na neplatný, jinak bychom vypisoval před každé písmeno mezeru. Tím se postaráme o odříznutí mezer na konci a zmenšení počtu po sobě následujících mezer na jednu.

Oboje by se dalo zjednodušit funkcí z Nette Frameworku, která změní všechny znaky na ASCII. Když se podíváme do zdrojového kódu, zjistíme, že jde vlastně o několik regulárních výrazů za sebou. Regulární výraz je obecně pomalou operací, ale vyhodnocení PHP kódu trvá tak dlouho, že rozdíl není tolik znát. Je zde ale neopomenutelná výhoda. Můžeme si být jisti, že se tím vyřeší všechny znaky.

Nevýhoda druhé možnosti je volání další funkce na každém charakteru. Zato projde celý řetězec jen jednou. První metoda projde řetězec dvakrát, odřezávání mezer na začátku a na konci ani zdaleka nemusí projít celý řetězec.

Implementovali jsme možnost druhou, hlavně protože nás napadla jako první.

4.1.2 Propojení oblíbenosti se starou databází

Když už jsme schopni získat všechny informace pro vytvoření řádek v tabulce oblíbenosti, můžeme se pustit do procházení staré databáze.

Prvně naše funkce potřebuje do SQL poslat požadavky na přidání patřičných sloupců a tabulek. Co budeme přidávat bylo zmíněno již v analýze.

Před vymyšlením ořezávání textu bylo vše řešeno jedinou procedurou v SQL, která prošla všechny názvy jídel a spojila je s oblíbeností. Řešení funkcí ušetří čas na komunikaci mezi klientem (PHP skriptem) a databází. Po přidání této podmínky jsme se rozhodli část této procedury ponechat. V PHP projdeme postupně všechny řádky názvů, ořezeme text funkcí výše zmíněnou a spustíme proceduru, která již zajistí zbytek propojení.

Procedura je vcelku jednoduchá, ze vstupních dat zjistí, jaké je ID jídelny, se kterou pracujeme. Tato data jsou uložena v jiné tabulce, kterou specifikují data z první, tudíž je nemá cenu získávat již v PHP. Poté zkontrolujeme, zda jsme našli jídelnu a že jsme dostali nějaký vstupní text. Nakonec se pokusíme pomocí těchto dvou parametrů najít položku v tabulce oblíbenosti. Nikdy bychom neměli dostat více než jednu shodu, ale pokud se objeví, zapíšeme null. V případě jedné shody zapíšeme její ID. V posledním případě vytváříme novou řádku a zapíšeme poslední vytvořené ID.

4.1.3 Propojení importovaných jídel

Jediný způsob, jak do databáze zanést nová jídla, je importem z jídelny. Tudíž zde můžeme zařídit propojení s hodnocením a neřešit nikde jinde.

Pro jednoduchost a uspořádanost si vytvoříme funkci na zjištění ID oblíbenosti. Jediný rozdíl vůči již popsanému SQL skriptu je, že je psána v PHP a jako vstupní parametr máme také ID jídelny. Nejprve ořezeme text, poté získáme potřebné ID způsobem dříve zmíněným.

Toto ID jsme poté vložili do objektu pro položky jídla v projektu, jen mu přidali proměnnou a setter. Již existující insert do databáze se pokouší vložit každou položku objektu do stejnojmenného sloupce v databázi.

4.2 Vypisování Webu

Prvním zádrhelem je, že potřebujeme náš objekt Hodnocení. Jenže ten potřebuje přístup k databázi.

Tudíž máme dvě řešení. Náš objekt uděláme jako rozšíření Presenteru – objektu Nette Frameworku, kterému by se měly, mimo jiné, vstupní parametry zajistit frameworkem. Toto řešení se nám ale nemůže povést zprovoznit. Třída ČástMenu, ve které bychom hodnocení potřebovali Hodnocení, není managována DI kontejnerem, který by tyto spoje jinde zajistil. Tudíž musíme přejít k možnosti druhé. Objekt Hodnocení si předat do konstruktoru objektu ČástMenu. Ta je vytvářena jen v jedné třídě, která už managována DI kontejnerem je, tam si vytvoříme objekt pro Hodnocení a konstruktoru ho předáme ČástMenu. Ale toto také nefunguje. Přehlédli jsme, že v jednu chvíli se ČástMenu tvoří z se souboru Json, kde si stoprocentně oblíbenost sehnat nelze. Jde o statickou funkci, které nevolá konstruktor třídy a tak se Hodnocení nevytvoří.

Nakonec jsme se přiklonili k možnosti vytvořit třídu Hodnocení v komponentě lístku, která se stará o výpis lístku a předali ji do globálních proměnných výpisu. Stačí nám jen tento jeden objekt, protože si při získávání hodnocení nepotřebujeme pamatovat žádné proměnné, nemusíme tedy ani řešit jejich vynulování. Kdybychom mermomocí chtěli objekt Hodnocení v ČástMenu vytvořit, můžeme si předávat údaje pro připojení k databázi.

4.2.1 Průměr

Abychom splňovali vše, co je od nás požadováno, potřebujeme průměr a zprávu, která se zobrazí při přejetí přes smajlíka průměru. Funkci, která se o toto stará, vložíme do objektu, který symbolizuje jeden čas a druh (například oběd 1) v jídelníčku. Bude nám vracet pole s průměrem (zaokrouhleným na celá čísla) a řetězcem, který se vloží do vyskakujícího okna.

Budeme potřebovat zjistit, jak byla jídla hodnocena v minulosti. To zajistíme párem funkcí v našem objektu. Jedna zjišťuje hodnocení spojené s názvem jídla, druhá s časem. Musíme si dát pozor na případ, že se vrátí nulové hodnocení, protože poté bychom dělili nulou a program by spadl!

Funkce, která se stará o minulost, je mnohokrát jednodušší. Vložíme ID času, které je stejné jako ID jeho hodnocení, získáme sumu a počet hodnocení. Z nich je získání průměru a řetězce triviální.

Funkce pro získání hodnocení budoucnost si načte array jídel. Problém je v tom, že jeden řádek odpovídající řekněme polévkám, může mít více členů. Tudíž array jídel je vlastně pole polí jídel. Každý člen nahradíme polem [průměr, počet hodnocení]. Toho dosáhneme tím, že si pro každou položku vyžádáme sumu a počet hodnocení, provedeme výpočet a zapíšeme. Pro výpočet celkového hodnocení potřebujeme sečíst průměry všech členů hodnocení a vydělit počtem. Poté projdeme znovu všechny řádky a vypíšeme průměr a počet hlasů.

Ještě by bylo vhodné vymyslet nějaký rozumný počet desetinných míst, na který budeme

zaokrouhlovat přesné průměry.

Musíme vyřešit, kdy tato tlačítka zobrazovat a tudíž se s touto celou zprávou generovat. Prvně zkontrolujeme, jestli má jídelna zapnuté hodnocení, jestli si zde alespoň jeden ze strážníků může objednávat a zda náhodou některý ze strážníků nemá jídlo neohodnocené. Pokud se díváme více než 10 dní do historie, nemůže hodnotit nikdo a zobrazujeme hodnocení všem strážníkům jídelny.

4.2.2 Hodnoticí tlačítka

Hodnoticí tlačítka vypisujeme pro každého strážníka zvlášť. Navíc musíme splnit několik podmínek. Zaprvé musí mít jídelna zapnuté hodnocení. Pak se musíme podívat, zda se toto jídlo už vydávalo. Toto uděláme funkcí na dvakrát. Prvně zkontrolujeme, zda nejde o minulost či budoucnost, poté je odpověď jasná. Druhým krokem je kontrola času dnes. Navíc se musíme podívat, jestli, když k datu přičteme 10 dní, dostaneme datum budoucnosti – tj, není povoleno hodnotit více než 10 dní zpět. Také se musíme podívat, zda měl strážník toto jídlo objednané.

Poté se ještě podíváme, které tlačítko označíme jako použité.

Poslední vypíšeme do divu v HTML data podstatná pro hodnocení – ID objednávky strážníka. Navíc si ke každému smajlíku poznamenáme, jakým číslem hodnotíme. To by se také dalo vyřešit čtyřmi různými funkcemi v JavaScriptu, ale to je zbytečně složité a nepřehledňuje kód.

4.3 Hodnocení

Chceme, aby se stisknutím jednoho ze čtyř hodnotících smajlíků spustil příslušný script, který si načte data a odešle na server AJAX, který se postará o ohodnocení jídla a vrátí nám nový průměr a string, který potřebujeme napsat do boxu, který se ukáže při přejetí přes smajlíka hodnocení. Musíme v PHP zkontrolovat, jestli je požadavek odeslán z konta, které smí pro tohoto strávnicka hodnotit. Navíc potřebujeme zkontrolovat, zda je objednávka opravdu strávnicka, jídlo si objednal a vyzvedl, splňuje časová omezení a že je hodnocení 1-4. Jinak je nám jedno, co si uživatel s daty udělá, dokud je my nezpracujeme a neupravíme údaje někoho jiného. Také ale musíme zkontrolovat, že nehodnotí nějaké jídlo, které si nevyzvedl/neobjednal.

4.3.1 Script

Script nejprve zkontroluje, zda se hodnotí tlačítkem, které jsme předtím neoznačili jako použité. Toto označení děláme přes tag class. Kontrolujeme, zda se class liší od nepoužitého tlačítka. Uživatel ale pořád může změnit použitá hodnotící tlačítka na nepoužitá, takže by bylo vhodné toto ještě kontrolovat na serveru. Hodnotili bychom stejnou hodnotou a zbytečně bychom prepisovali data.

První věc kterou uděláme, je nalezení dříve použitého tlačítka, tomu změníme class na nepoužité. Změníme to, na které klikl, na použité a spustíme Ajax.

Nejjednodušší pro nás bude, když ze serveru získáme rovnou HTML kód obrázku a divu. Jejich výpis bude o to jednodušší a nebudeme muset řešit změny jak na serveru tak v JS.

Ideálně bychom dostali array o dvou členech. První bude Obrázek, druhý Div. Ze serveru ale můžeme získat pouze text, tudíž se pokusíme data zakódovat do JSON objektu, který snadno rozkódujeme.

V případě, že dostaneme ze serveru error, vrátíme tlačítka do původního stavu.

Zatím nekontrolujeme, že si objednali úplně všichni strávnicki dne. Hodnocení zobrazíme po prvním ohodnocení dne.

4.3.2 PHP

Začneme od nejjednodušší části. Budeme potřebovat funkci, která se podívá do databáze a vrátí nám požadované HTML elementy. Použijeme identický algoritmus jako u výpisu dne, jen namísto průměru vracíme rovnou obrázek, se kterým je průměr spojený a neprovádíme rozdělení později. Stejně tak nevracíme text, ale text obalený v příslušném divu.

Poté funkce pro ohodnocení. Nejprve potřebujeme zjistit, jak člověk hodnotil v minulosti. Máme dvě možnosti. Použít již nějaký existující model v projektu, který zařizuje přístup k objednávkám v databázi, či se jednoduše na řádku v databázi podívat. V projektu jsme ale bohužel model nenašli, tudíž se jednoduše podíváme na data v objednávce a zaměníme

je za nová. Pokud se nové a staré hodnocení rovná, nemáme jak hodnotit a měli bychom vrátit AJAX s errorem.

Nyní ohodnotíme den. Tabulku máme nastavenou tak, že ID kalendáře se rovná ID v této tabulce. Jak zjistíme ID kalendáře? Objednávka v sobě nese datum. Kalendář by měl mít to samé datum. Kalendář a Objednávka mají obě ID části částí (Druh ID) dne. Ale jak si můžeme být jisti, že je tato část jedinečná pro jídelnu? Druh obsahuje ID chodu (části dne) a ten už v sobě má ID jídelny. Tudíž musí ID kalendáře být jedinečné podle ID druhu a data. Nemusíme ani kontrolovat ID zařízení v něm. Zapojil bych vyhledávání ID kalendáře a jeho oblíbenosti do jednoho selectu, ale ještě budeme ID kalendáře potřebovat.

V případě, že je ID kalendáře nulové, musíme vyhodit error. K tomu by nikdy nemělo dojít, ale protože databáze ještě není pořádně provázána za pomoci foreign keys, mohlo by k tomu dojít.

Pokud hodnocení nenalezneme, vytvoříme novou řádku v tabulce s příslušnými informacemi. Pokud řádka existuje, odečteme od nového hodnocení staré a to přičteme k řádce sumy. Když bylo staré nulové, uložíme si jedničku jinak nulu a toto přičteme k počtu hodnocení. Poměrně nově databáze podporuje přičtení hodnot k řádce, není nutné si řádku nejdříve načíst a poté vložit s upravenými hodnotami. To dokonce zabrání možnosti, aby se hodnota změnila poté, co jsme si ji načetli, a tím pádem ignorovali jiné hodnocení.

Nyní musíme změnit hodnocení všech položek jídla. Jak již bylo zmíněno, člověk si nejprve musí sehnat array popisů a teprve z nich získá array položek. Znovu zde je možnost použít již existující modely a funkce v nich nebo si samostatně vše vyžádat z databáze. A ač to znamená horší kompatibilitu při jakékoliv změně databáze, rozhodli jsme se vše vyřešit vlastní cestou.

Načteme si všechny skupiny položek, které ukazují na ID našeho Kalendáře. Stejným způsobem načteme všechny položky jídla, které ukazují na alespoň jednu ze skupin položek. Toho dosáhneme tím, že všechna ID skupin položek načteme do jednoho velkého pole a použijeme IN podmínku. Na pořadí nám nezáleží. Nakonec projdeme všechny ID oblíbenosti v položkách jídla, podle nich upravíme hodnocení. V případě, že nějaká položka oblíbenost nemá, je možné ji vytvořit.

Teď nám zůstává nejtěžší část. Zkontrolovat zda je požadavek přijatelný. Také potřebujeme všechny funkce spustit. Problém je v tom, že jsme ještě nezjistili, jak zařídit, aby se po průchodu Nette routerem spustil požadovaný PHP soubor.

Pro validaci bychom potřebovali session ID = ID člověka, který je aktuálně přihlášený, toto také nevíme jak získat. Mělo by přijít společně s požadavkem z AJAXu, ale protože ho nejsme schopni přijmout, nevíme, jak se přesně získá.

5. Závěr

Zadáním práce bylo hodnocení jídel v aplikaci jídelních lístků na stránce www.jidelna.cz.

Důležité části práce jsou všechny dokončené. Vymysleli jsme, jak reprezentovat data v databázi a zajistili, aby se v ní vytvořily nové sloupce a tabulky. Poté jsme tabulky naplnili takovými hodnotami, aby bylo ohodnocování jídel co nejrychlejší. Zajistili jsme, aby hodnocení fungovalo i pro nové jídelní lísky jídelen. Úspěšně jsme do jídelního lísku vypisovaného na webu přidali požadované informace. Zobrazujeme správně smajlíka, který symbolizuje průměrné hodnocení, a ukážeme přesné hodnocení při přejetí přes něj. Také vypisujeme smajlíky pro ohodnocení a zvýrazňujeme, jak již strážníci hodnotili. Vytvořili jsme funkce, které zařizují ohodnocení jídla a získání dat o hodnocení.

Pro plnou funkčnost programu potřebujeme ještě při ohodnocení jídla z webu spustit námi již připravené funkce. Je nutné zformovat podmínky, kdy je požadavek pro ohodnocení jídla platný. Také nám chybí odstranění neplatného hodnocení z doby, kdy nelze všechny podmínky kontrolovat. Nakonec je potřeba vymyslet, kde v databázi udržovat váhu hodnocení.

Práci nám značně komplikovala neznalost již existujícího projektu, protože jsme nevěděli, co vše již je implementováno, a proto není vhodné znovu vytvářet. Navíc je v projektu použitý framework bez podrobné dokumentace, který neznalému značně zkomplikuje porozumění kódu. Přesto jsme zadání práce skoro celé splnili.

Mnohokrát děkujeme firmě Barda za možnost tuto práci vytvořit.

Literatura

- [1] Documentation: Nette Framework [online]. Nette Foundation, 2019 [cit. 2019-02-20]. Dostupné z: <https://doc.nette.org/en/3.0/>. Dokumentace Nette a Latte.
- [2] PHP: Documentation [online]. PHP Group, 2019 [cit. 2019-02-20]. Dostupné z: <http://php.net/docs.php>.
- [3] JQuery API Documentation [online]. The jQuery Foundation, 2019 [cit. 2019-02-20]. Dostupné z: <https://api.jquery.com/>
- [4] MariaDB Server Documentation [online]. MariaDB, 2019 [cit. 2019-02]. Dostupné z: <https://mariadb.com/kb/en/library/documentation/>

Přílohy

<https://github.com/PetrAda/Jidelna>