Vyšší odborná škola, Střední průmyslová škola a Střední odborná škola, Varnsdorf, příspěvková organizace

CHYTRÁ KUCHAŘKA

4. IT

Informační technologie

Imrich Zeleňák

Varnsdorf 2023

Obsah

1 Úvod

Cílem mé praktické maturitní práce bylo vytvořit webovou aplikaci kuchařky s chytrými prvky. Chytrými prvky jsou myšleny takové prvky, které pomohou a usnadní uživateli s výběrem vhodného a pro ně dostupného receptu např.: uživatel si vybere suroviny které má doma a webová aplikace mu z databáze vypíše recepty, které vybrané suroviny obsahují.

Webová stránka by měla splňovat základní kritéria jakožto responzivitu pro různá zařízení (telefony, tablety, …) a přístupnost pro lidi se zrakovým postižením. Také by měla mít moderní design, měla by být pro uživatele na první pohled chytlavá, aby jí nadále využíval a samozřejmě by měla být validní.

Při tvorbě webové “Chytré kuchařky” jsem použil několik technologií.

Použitý software a technologie:

Výsledkem mé práce je dle mého jednoduchý prototyp pro “Chytrou kuchařku”, který by po pár rozšířeních o několik důležitých funkcí jako např.: vyhledávácí pole s našeptávačem, možnost uživatele si vytvořit účet, kde bude moct přidávat vlastní recepty s obrázky, administrativní část pro přidávání, mazání a úpravu dat v databázi, a přidání několika důležitých a zajímavých informací k receptu jako např.: výživové hodnoty, seznam obsažených alergenů, recenze ostatních registrovaných uživatelů nebo třeba uživatelské hodnocení receptů, mohl být zařazen do praktického nasazení a mohl by být využíván širokou veřejností.

V průběhu realizace mého zadání, jsem objevil jednu podobnou, ale již plně funkční webovou kuchařku, která měla také výběr receptů dle zadaných surovin. Nevýhodou již zmíněné kuchařky je ten fakt, že suroviny musíte zapsat do textového pole přesně dle daných kritérii tzn. Na začátku velké písmeno, ostatní písmena malá i s diakritikou. V mé verzi chytrého vyhledávání pouze zaškrtnete políčko s danou surovinou a práce tím pro uživatele končí. Dále stačí jen kliknout na tlačítko filtrování a recepty jsou hned po načtení dat k dispozici.

2 Teoretický úvod

2.1 Tvorba DB

Prvním a zároveň nejdůležitějším úkolem v postupu práce na “Chytré kuchařce” je vytvoření databáze, na které celá webová aplikace stojí. Pro tvorbu databází je velmi často používán software pro tvorbu SŘBD zvaný “MySQL workbench”, který umí z vámi vytvořeného E-R diagramu obsahujícím databázové objekty a relace mezi nimi vygenerovat pomocí reverzního inženýrství SQL skript. Pomocí zmíněného vygenerovaného SQL skriptu lze za pomoci administrativních softwarů pro SŘBD, v mém případě již zmíněný “phpMyAdmin”, vytvořit databázi, která je již připravená na pouhé naplnění daty.

2.2 Připojení a komunikace s DB

Druhým úkolem v postupu práce je potřeba vytvořit v backend programovacím jazyce PHP skript pro komunikaci s danou databází. Jazyk PHP má spousty již vytvořených funkcí, mezi něž spadají i funkce pro komunikaci s databází. V mém případě použiji funkce např.: mysqli_connect a mysqli_query. Funkce mysqli_connect slouží k připojení PHP skriptu s databází. K funkčnosti této funkce je zapotřebí znát adresu serveru, kde se SŘBD nachází, v mém případě aplikace poběží na lokálním serveru, takže adresa serveru bude 127.0.0.1, dále je potřeba znát název databáze, přihlašovací jméno a heslo. Funkce mysqli_query slouží k vyslání SQL příkazu např.: SELECT, INSERT, UPDATE, nebo velmi důležitého prvotního sladění SŘBD s webovou aplikací, a to vysláním příkazu mysqli_query, ve kterém vyšleme příkaz, který zaručí komunikaci ve správném kódování UTF-8. Toto zaručí správné načítání webové stránky a všech dat v ní.

2.3 Frontend

Ve třetím úkolu vytvoříme PHP skript, který všechna data ze SŘBD uloží do proměnných a vygeneruje HTML kód, který již bude tvořit strukturu webové aplikace. Po případné opravě a úpravě všech PHP skriptů, které jak pracují s daty z databáze nebo generují web budeme moci přejít k výslednému kroku, a to ke tvorbě vizuálních prvků aplikace. Mým plánem je vytvořit takzvaný „one page“ web, který se od klasické webové stránky liší tím, že se veškerý kontent webové stránky na kliknutí přegeneruje na kontent jiný. Jelikož se obsah webové stránky liší po přidání dat do databáze, jak přidání receptu do databáze, přidání nové kategorie nebo suroviny, takto generovaná stránka nebude potřebovat změny. Kdyby byl web napsán staticky, webové stránky by se tudíž sami neobnovovali a nerozšiřovali o nová data. Také stylování takovéto webové stránky je poněkud jednodušší z mého pohledu.

3 Praktická část

3.1 Tvorba DB

3.1.1 Evidence entitních typů

Z hlediska zadání potřebujeme evidovat údaje o kategoriích, receptech, surovinách a jejích jednotek potřebných k výrobě. Použil jsem tedy čtyři entitní typy: *kategorie, recept, suroviny a jednotky*.

3.1.2 Entitní typ kategorie

Entitní typ kategorie má přivlastněné dva atributy, což jsou: *id* a *název* (číslo pozdější specifické kategorie a její přidělený název).

3.1.3 Entitní typ suroviny

Entitní typ suroviny má přivlastněné také dva atributy, což jsou: id a název (číslo pozdější specifické suroviny a její název). Později se však tento entitní typ dá rozšířit o další atributy, jakožto nutriční hodnoty suroviny nebo jaké alergeny obsahuje. Pro alergeny bych však nejspíše vytvořil nový entitní typ, který by obsahoval atributy: *id, název.*

3.1.4 Entitní typ jednotky

Entitní typ jednotky obsahuje také dva atributy *id a název*. (číslo specifické jednotky a její název), tento entitní typ bude později využit ve vztahu s entitním typem *recept_has_suroviny* v poměru 1 ku N.

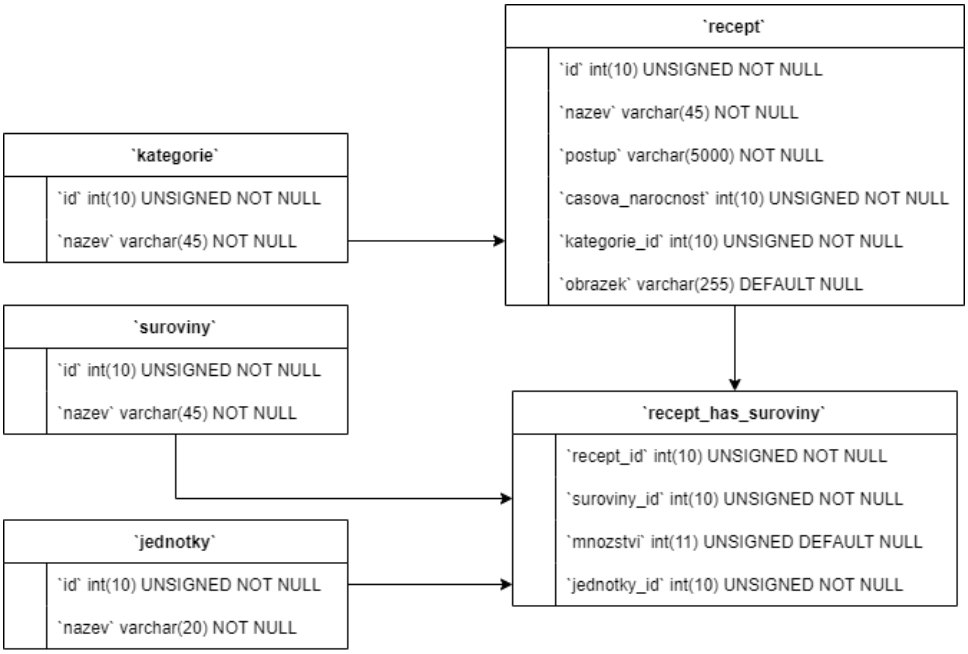
3.1.5 Entitní typ recept

Entitní typ recept obsahuje poněkud více atributů, mezi ně patří: id, název, postup, časová náročnost, obrázek a číslo kategorie, které je získáno z tabulky entitního typu kategorie. Tento vztah zajistíme přidáním atributu *kategorie_id* (cizí klíč, který ukazuje na primární klíč *id* v tabulce kategorie) k tabulce entitního typu recept. Z hlediska kardinality (četnosti) víme, že každý recept je obsažen v jedné kategorii a současně každá kategorie může obsahovat více receptů vyplívá z toho tedy to, že se jedná o typ vztahu 1 ku N. V případě parciality recept musí být obsažen v kategorii (povinný vztah), ale každá kategorie nemusí obsahovat recept (nepovinný vztah). Abych neopomněl v budoucích vylepšených verzích by se dalo oddělit atribut obrázek od tabulky *recept* a vytvořit samostatný entitní typ s názvem *obrázek*, který by obsahoval atributy *id, název* z toho důvodu, aby k jednotlivému receptu mohlo být přiřazeno více obrázku, a ne pouze jeden. Z hlediska kardinality by se jednalo o entitní vztah s tabulkou *recept* typu 1 ku N.

3.1.6 Vztah N ku M a entitní typ recept_has_suroviny

V neposlední řadě je tu spojení entitních typů recept a suroviny. Jelikož z bližšího zkoumání zjistíme, že recept obsahuje více surovin a každá surovina může být obsažena ve více receptech jedná se tedy z hlediska kardinality o vztah mezi entitami typu M ku N. Právě a proto, že se jedná o vztah mezi entitami typu M ku N musíme vztah rozdělit na právě dva vztahy 1 ku N. Tím nám vznikne další entitní typ *recept_has_suroviny,* který obsahuje cizí klíče odkazující na primární klíče právě těchto dvou tabulek. Jelikož se mezi těmito dvěma tabulkami vytvořila třetí tabulka, můžeme jí využít pro řešení problému s určením konkrétního množství a specifické jednotky pro každou surovinu. K tabulce *recept_has_suroviny* tudíž připojíme tabulku s jednotkami tak, že přidáme atribut s názvem *jednotka_id* (odkazující na primární klíč id v tabulce jednotky). Problém s množstvím dané suroviny vyřešíme tím způsobem, že do tabulky přidáme atribut s názvem množství.

Takto vypadá finální E-R diagram po vytvoření všech předchozích úkonů:



1 E-R diagram Chytrá kuchařka

3.2 Komunikace s DB

Po vytvoření DB jsem se přesunul ke tvorbě backend scriptů v jazyce PHP, které umožní komunikaci mezi databázovým serverem a webovou aplikací chytré kuchařky. K této komunikaci jsem musel, jak je již v úvodu zmíněno, využít software XAMPP. XAMPP je jednodušší verze originálního webového serveru Apache, která poskytne webovým vývojářům spustit a otestovat své projekty v lokálním prostředí. Aby byl potenciál plně využit XAMPP také obsahuje databázi MariaDB, kterou také můžete využít.

3.2.1 Třída Database

Jako první jsem si vytvořil soubor database.php, ve kterém se nachází třída Database, který slouží k připojení k databázi a nastavení komunikace na kódování UTF-8. Ve třídě jsem si vytvořil funkci connect(), ve které jsem využil již zmíněných funkcí mysqli_connect a mysqli_query. V souboru index.php jsem vytvořil objekt z třídy Database a uložil ho do proměnné, abych ho mohl nadále využívat v dalších funkcích mysqli_query. Funkčnost této části skriptu jsem otestoval v souboru index.php jednoduchou PHP funkcí var_dump.

```php
    public function __construct()
    {
        $this->hostname = "127.0.0.1"; // Adresa localhost
        $this->database = "chytra_kucharka"; // Název databáze, s níž se chceme spojit
        $this->username = "root"; // Uživatelské jméno pro přihlášení do databáze
        $this->password = ""; // Heslo pro přihlášení do databáze
        $this->port = "3306"; // Číslo portu, na kterém běží databázový server (výchozí pro MySQL)
    }
    public function connect(){
        $this->conn = mysqli_connect($this->hostname, $this->username, $this->password,
$this->database, $this->port); // Pokusíme se připojit k databázi
        if($this->conn == false){ // Pokud se nepodařilo připojit
            echo "Failed to connect: ".mysqli_connect_error(); // Vypíšeme chybovou hlášku
            exit(); // Ukončíme běh skriptu
        }
        mysqli_query($this->conn, "SET CHARACTER SET UTF8"); // Nastavíme kódování pro spojení na UTF-8
        return $this->conn; // Vrátíme objekt představující spojení s databází
    }
```

2 PHP třída Database

3.3 Výběr všech potřebných dat z DB

Pro výběr všech potřebných dat jsem zvolil vytvoření několika tříd s názvy, které odpovídají názvům datových tabulek, ze kterých data berou.

3.3.1 Třída Kategorie

Jako první jsem vytvořil třídu s názvem Kategorie, která pomocí již zmíněné funkce mysqli_query a pomocí již mnou vytvořené třídy Database, která připojí daný skript s databází, vybere všechny daná data z databáze *chytra_kucharka*. K výběru všech potřebných dat, jsem do funkce mysqli_query použil SQL dotaz SELECT v následném znění: *SELECT id, nazev FROM kategorie;* Tento SQL dotaz z již zmíněné databáze *chytra_kucharka* vybere všechna data ze sloupečku v tabulce s názvem *id* a *nazev*. Nadále jsem objekt s daty, který funkce mysqli_query uložil do proměnné a v dalších krocích zpracoval. Pro zpracovaní dat jsem využil funkce jazyka PHP, a to konkrétně funkce foreach. Ve funkci foreach, kterou jsem využil zejména kvůli tomu, že funkce umí zpracovat proměnný počet jednotlivých dat, což má na rozdíl od funkce for značnou výhodu. V této funkci jsem vybraná data uložil do proměnných *id a nazev* a následně jsem je jednoduchým spojení textových řetězců spojil a uložil do pole, aby pro mě bylo jednodušší budoucí zpracování dat. V souboru index.php jsem ze třídy Kategorie následně vytvořil objekt a výsledek mého snažení jsem otestoval funkcí var_dump.

```php
class Kategorie{

    private $kategorie; // Vlastnost pro uchování dat o kategorii
    private $arr; // Vlastnost pro uchování pole kategorií
    private $query; // Vlastnost pro uchování dotazu na kategorie

    public function __construct($db_connect){
        $this->query = "SELECT id, nazev FROM kategorie"; // Nastavení SQL dotazu
        return $this->query = mysqli_query($db_connect, $this->query); // Vykonání dotazu a vrácení výsledku
    }

    public function get_data(){
        foreach($this->query as $this->kategorie){ // Procházení výsledků dotazu
            $id = $this->kategorie["id"]; // Získání ID kategorie
            $nazev = $this->kategorie["nazev"]; // Získání názvu kategorie
            $arr[] = $id."/".$nazev; // Vytvoření řetězce ID/název kategorie a přidání ho do pole kategorií
        }
        return $arr; // Vrácení pole kategorií
```

3 PHP třída Kategorie

3.3.2 Třída Suroviny

Následně po vytvoření a otestování objektu z třídy Kategorie jsem se vrhnul na tvorbu souboru, ve kterém se bude nacházet třída Surovina, kterou v pozdější části využijeme jako soubor dat pro vygenerování seznamu surovin pro náš chytrý vyhledávač. Třída Surovina obsahuje funkci magickou funkci __construct, která se automaticky zavolá při vytváření objektu z dané třídy a pomůže tak s jejím prvotním nastavením. K této funkci přidáme do vstupu proměnnou db_connect, místo které následně v soboru index.php doplníme proměnnou, ve které je uložen objekt ze třídy Database. Poté jsem si jako druhou funkci ve třídě Suroviny vytvořil funkci get_data, která se nachází ve všech ostatních mnou specifikovaných třídách. Funkce get_data má za úkol, jak je již zmíněno, získat data podle specificky zadaného SQL dotazu. V tomto případě, jelikož se jedná o třídu Suroviny, budeme tedy potřebovat takový SQL dotaz, který nám získá všechna potřebná a využitelná data z databázové tabulky Suroviny. Pro takový případ jsem poskládal SQL dotaz následujícího znění: SELECT nazev FROM suroviny. Následně použiji již zmíněnou PHP funkci foreach, abych získaná data dokázal z výsledku funkce mysqli_query dostat a použít. Ve funkci foreach data uložím do proměnných nesoucích stejný název jako získávaná data. Poté pomocí jednoduché vychytávky jazyka PHP data spojím při každém jednom cyklu a vyrobím z nich textový řetězec, který uložím do jednoduchého pole, bych usnadnil následnou manipulaci s těmito daty. Funkčnost této třídy jsem ověřil stejným postupem jako u předchozích. V souboru index.php jsem si z třídy vytvořil objekt, který jsem uložil do proměnné. Tuto proměnnou jsem dosadil do funkce již zmíněné var_dump.

```php
class Surovina{

    private $suroviny; // Proměnná pro uložení dat z SQL
    private $arr;
    private $query; // Proměnná pro SQL dotaz

    public function __construct($db_connect){
        // Nastavení dotazu pro získání názvů surovin z databáze
        $this->query = "SELECT nazev FROM suroviny";
        // Výsledek dotazu se uloží do vlastnosti objektu
        return $this->query = mysqli_query($db_connect, $this->query);
    }
    public function get_data(){
        // Projdeme všechny řádky výsledku dotazu
        foreach($this->query as $this->suroviny){
            // Pro každou surovinu získáme její název a uložíme ho do pole
            $nazev = $this->suroviny["nazev"];
            $arr[] = $nazev;
        }
        // Vrátíme pole s názvy surovin
        return $arr;
    }
}
```

4 PHP třída Surovina

3.3.3 Třída Recept

V dalším kroku jsem si vytvořil soubor Recept, který obsahuje stejnojmennou třídu Recept. Tato třída slouží, podobně jako ostatní třídy, k získání specifických dat z databáze pod názvem *chytra_kucharka*. Pro vytvoření jsem znovu použil magickou funkci _construct jejíž použití jsem zmínil již v předchozím

odstavci, kde se zmiňuji o třídě s názvem Surovina. Do této třídy jsem následně vytvořil druhou funkci, která nese stejný název, jako v již zmíněné třídě Surovina. Funkce get_data v této třídě, je tu pro stejný účel jako v ostatních třídách, což znamená získání dat pomocí SQL dotazu. Do této funkce jsem použil poněkud rozsáhlejší dotaz SQL proto, aby šlo názorně vidět, jak se ten jeden a ten samý SQL dotaz, který získává základní data z databáze, může lišit. V tomto případě jsem použil takzvaný "Alias", což znamená, že následně data nebudou uložena ve výsledku pod názvem stejným, jako se nachází v databázi, ale budou uložena pod tím aliasem. Jak již předchozích podkapitol víme, konkrétně z těch o tvorbě databázového modelu, víme, že tabulka neboli entitní typu s názvem recept obsahuje právě jeden cizí klíč (odkazuje na primární klíč v jiné tabulce). Pro tyto případy existuje v MySQL verze dotazu, kde se dají propojit dvě nebo více tabulek. Pomocí INNER JOIN, který se zapíše v dotazu SELECT, můžeme jednoduše získat data z jiné tabulky právě podle primárního klíče, který se v obou tabulkách rovná. Toto jsem použil abych získal název kategorie, ve které se daný recept nachází a pomocí tohoto jsem poskládal následující SQL dotaz: SELECT recept.id AS ID, recept.nazev AS NÁZEV, recept.postup AS POSTUP, recept.casova_narocnost AS ČAS, kategorie.nazev AS KATEGORIE, recept.obrazek AS OBRAZEK FROM recept INNER JOIN kategorie ON recept.kategorie_id = kategorie.id; Jako neposlední krok ve funkci get_data znovu použiji již zmíněnou funkci foreach a pomocí této funkce si specifická data pod aliasi uložím do proměnných. Tyto proměnné následně spojím znakem "/" a uložím do jednoduchého pole. Funkčnost následně otestuji stejně jako jsem to udělal s předchozími třídami. V souboru index.php vytvořím objekt ze třídy Recept a uložím ho do proměnné. Tuto proměnnou následně vložím do funkce var_dump a ujistím se, že vše funguje tak jak má.

```php
class Recept{

    private $recept_data; // Proměnná pro uložení dat o receptech
    private $data;
    private $query; // Proměnná pro SQL dotaz

    public function __construct($db_connect)
    {
        // Nastavení SQL dotazu pro získání dat o receptech a připojení tabulky kategorie
        $this->query = "SELECT recept.id AS ID, recept.nazev AS NÁZEV, recept.postup AS POSTUP, recept.casova_narocnost AS ČAS, kategorie.nazev AS KATEGORIE,
recept.obrazek AS OBRAZEK FROM recept INNER JOIN kategorie ON recept.kategorie_id = kategorie.id";
        // Provedení dotazu a uložení výsledků do proměnné recept_data
        $this->recept_data = mysqli_query($db_connect, $this->query);
    }
    public function get_data(){
        foreach($this->recept_data as $this->data){
            // Uložení jednotlivých položek receptu do proměnných
            $id = $this->data["ID"];
            $nazev = $this->data["NÁZEV"];
            $postup = $this->data["POSTUP"];
            $cas = $this->data["ČAS"];
            $kategorie = $this->data["KATEGORIE"];
            $obrazek = $this->data["OBRAZEK"];
            // Uložení informací o receptu do pole recept
            $recept[] = $id."/".$nazev."/".$postup."/".$cas."/".$kategorie."/".$obrazek;
        }
        // Vrácení pole s informacemi o receptech
        return $recept;
    }
}
```

5 PHP třída Recept

3.3.4 Třída Recept_suroviny

V posledním kroku vyjímání dat z databáze pod názvem *chytra_kucharka* jsem vytvořil soubor recept_suroviny.php a v něm jsem vytvořil stejnojmennou třídu Recept_suroviny. V této třídě, stejně jako ve třídách zmíněných v předešlých podkapitolách, použiji magickou funkci __construct, která se od funkcí __construct v předešlých kapitolách liší především SQL dotazem, který posílá hned po připojení do databáze. Tento SQL dotaz, který funkce __construct využívají, je ze všech předešlých použitých SQL dotazů nejsložitější. Tento SQL dotaz využívá spojení až tří databázových tabulek. Třídu nesoucí název Recept_suroviny využijeme v pozdější části tvorby webové stránky, abychom získali veškeré suroviny, které daný recept obsahuje. Abychom chtěného výsledku docílili, musíme použít SQL

dotaz následujícího znění: „SELECT jednotky.nazev AS jednotky, suroviny.nazev AS suroviny, recept_has_suroviny.mnozstvi FROM recept_has_suroviny INNER JOIN jednotky ON recept_has_suroviny.jednotky_id = jednotky.id INNER JOIN suroviny ON recept_has_suroviny.suroviny_id = suroviny.id INNER JOIN recept ON recept_has_suroviny.recept_id = recept.id WHERE recept.nazev = „$nazev“;“ Do proměnné s názvem $nazev, ve které se nachází název receptu, který posléze získáme jako výsledek při používání webové aplikace. Výsledkem tohoto dotazu získáme přístup ke všem názvům surovin, jejich množství a také k jednotkám ve kterých je množství dané suroviny uvedeno. Posléze si ve stejné třídě vytvořím druhou funkci, kde získaná data zpracuji. Jak je již řečeno v předešlých podkapitolách, jedná se o funkci get_data. Ve funkci data opětovně použiji funkci foreach, kterou projdeme všechna data získaná již zmíněným SQL dotazem a následně všechna získaná data uložím do proměnných. Tyto proměnné pak spojím pomocí jednoduchého zápisu v jazyce PHP a vytvořím pomocí toho textový řetězec, který spojím znakem „/“. Tento řetězec obsahuje všechna vybraná data, z již zmíněného SQL dotazu a jelikož jsem z těchto dat vytvořil řetězec, bude pro mě jednodušší budoucí manipulace s těmito daty.

```php
class Recept_suroviny{

    private $data;  // Proměnná pro ukládání dat z databáze
    private $recept_data;   // Proměnná pro ukládání dat o receptu
    private $jednotky;  // Proměnná pro ukládání jednotek surovin
    private $mnozstvi;  // Proměnná pro ukládání množství surovin
    private $suroviny;  // Proměnná pro ukládání názvů surovin
    private $recept_has;    // Pole pro ukládání surovin a jejich množství v receptu

    public function __construct($db_connect, $name)
    {
        // SQL dotaz pro získání surovin a jejich množství pro daný recept
        $this->data = mysqli_query($db_connect,"SELECT jednotky.nazev AS jednotky, suroviny.nazev AS suroviny, recept_has_suroviny.mnozstvi FROM recept_has_suroviny INNER JOIN jednotky ON recept_has_suroviny.jednotky_id = jednotky.id INNER JOIN suroviny ON recept_has_suroviny.suroviny_id = suroviny.id INNER JOIN recept ON recept_has_suroviny.recept_id = recept.id WHERE recept.nazev = '$name'");
    }

    public function get_data(){
        // Cyklus pro procházení dat z databáze a ukládání informací do pole $recept_has
        foreach($this->data as $this->recept_data){
            $this->jednotky = $this->recept_data["jednotky"];
            $this->mnozstvi = $this->recept_data["mnozstvi"];
            $this->suroviny = $this->recept_data["suroviny"];
            $this->recept_has[] = $this->mnozstvi." ".$this->jednotky." ".$this->suroviny;
        }
        // Vrácení pole s informacemi o receptech
        return $this->recept_has;
    }
}
```

6 PHP třída Recept_suroviny

3.4 Tvorba a zpracování prvotních dat v index.php

3.4.1 Připojení souboru index.php k databázi

Po vytvoření již zmíněných tříd z předešlých podkapitol jsem dané soubory s třídami importoval do hlavního souboru index.php, a náhle na to ze všech tříd vytvořím stejnojmenné objekty naplněné daty z databáze. Jako první si po importování souboru database.php vytvořím objekt ze třídy Database a uložím ho do proměnné $conn. Tuto proměnnou budu v pozdějších částích kódu využívat v ostatních tvorbách objektů. Dále z vytvořeného objektu zavolám funkci connect, která obsahuje hlavní funkci jazyka PHP pro připojení do databáze zvaná mysqli_connect. Tento objekt je po odzkoušení ve funkci var_dump typem mysqli.

3.4.2 Vytvoření objektu a zpracování dat ze třídy Kategorie

Následně si po importaci souboru kategorie.php vytvořím objekt ze stejnojmenné třídy Kategorie. Při tvorbě tohoto objektu použiji předem vytvořenou funkci get_data, která jak již bylo zmíněno, pošle mnou zadaný SQL dotaz a získá data z databáze, které uloží na výstup této funkce. Tento zmíněný výsledek, jelikož je na výstupu funkce get_data, kterou jsem zavolal hned při tvorbě objektu, uloží do

dané proměnné, kterou jsem pojmenoval stejným názvem jako je název třídy. Ve třídě Kategorie, abych ušetřil několik řádků, které by se jinak opakovali, jsem do magické funkce __construct dal jako povinný parametr atribut hodnoty mysqli. Do této hodnoty dosadím již zmíněný výsledek objektu Database, který jsem uložil pod proměnnou $conn a je taktéž typem mysqli. Tím zaručím, že se mi podaří docílit výsledku při tvorbě objektu ze třídy Kategorie. Následně po získání dat z této třídy a uložení jeho výsledku do proměnné $kategorie, všechna tato ošetřím, abych získal mnou potřebná a využitelná data. K tomuto použiji funkci, již několikrát zmíněnou a využitou u tvorby tříd, s názvem foreach. Funkcí foreach projdu pole s daty, které je uloženo pod proměnnou $kategorie. Každý jeden prvek toho pole je textový řetězec, který jsem sám vytvořil při získávání dat ve funkci get_data dané třídy. Při každém jednom prvku pole se data rozdělí a získám tak dvourozměrné pole, ve které se nachází pospolu uložená číselné id a název kategorie. Tato data následně využiji v další kapitole týkající se tvorby frontend části webové stránky.

3.4.3 Vytvoření objektu a zpracování dat ze třídy Recept

Nyní jsem si naimportoval soubor recept.php, který obsahuje stejnojmennou třídu Recept. Opět, jak jsem již zmínil v předchozí podkapitole, dosadím při tvorbě objektu do parametru magické funkce __construct proměnnou $conn, ve které se nachází objekt typu mysqli, který je vytvořen ze třídy Database. Při tvorbě toho objektu použiji, jak je již zmíněno v předchozí podkapitole, funkci get_data, kterou jsem si vytvořil v dané třídě. Tato funkce také slouží k získání dat dle daného SQL dotazu, který jsem zadal již při tvorbě dané třídy. Výsledek volané funkce je, jak je již zmíněno ve tvorbě tříd, jednoduché pole, které obsahuje informace o receptech. Toto pole s výsledky funkce get_data uložíme do proměnné $recept, kterou v následujících krocích ošetříme a připravíme pro použití ve tvorbě frontend části webové stránky. Ke zpracování dat již poněkolikáté využijeme funkci foreach, při každém prvku zadaného pole použijeme na daný prvek funkci explode a uložený textový řetězec rozdělíme dle znaku „/“, kterým jsme ho při tvorbě ve třídě Recept vytvořili. Vytvořená data uložíme do proměnné, která je polem dat. Dále uložené pole znovu vložíme do funkce foreach a znovu všechny prvky projedeme zmíněnou funkcí. Při každém prvku uložím do pole hodnotu z předešlého průchodu, tím je myšleno, že do proměnné, která je polem ve druhé funkci foreach, uložím data z druhého místa (číslo jedna, když budeme číslovat dle polí) z prvního průchodu funkcí foreach. Uložená data z proměnných dále využiji ve tvorbě frontend části webové stránky, jak je již zmíněno.

3.4.4 Vytvoření objektu a zpracování dat ze třídy Suroviny

Nyní se vrhneme na tvorbu a zpracování dat z objektu vytvořeného ze třídy Suroviny. Jako první krok ve vytváření objektu ze třídy Suroviny není nic jiného než importování souboru, který obsahuje danou třídu, do souboru index.php, kde budeme objekt z již zmíněné třídy vytvářet. K vytvoření objektu bude zapotřebí zas a znovu vložit do magické funkce __construct proměnnou $conn, která obsahuje výsledek funkce mysqli_connect, která je využita ve třídě Database. Po vložení proměnné $conn můžeme zavolat funkci get_data, která je stejná jako v předešlých podkapitolách. Výsledkem funkce get_data je soubor dat z databáze, který získáme mnou zadaným SQL dotazem vyslaným funkcí jazyka PHP, a to konkrétně funkcí mysqli_query. Výsledek daného vytvoření objektu je tedy soubor dat uložený v proměnné, kterou následně využijeme pro tvorbu frontend části webové stránky. Tyto data samostatně nijak neupravuji, pouze při tvorbě frontend části webové aplikace celé pole se všemi surovinami vložím do funkce foreach a použiji je pro generování seznamu surovin, ze kterého si následně uživatel, který chce mnou vytvořené chytré vyhledávání použít, může použít.

3.5 Generování frontendové části webové stránky

Jelikož, a protože se jedná o takzvaný one-page web, webová stránka potřebuje ke své existenci pouze jeden jediný soubor index.php. Tento způsob webové stránky mi umožní dle mého jednodušší pohyb po stránce a také dle mého rychlejší a také není tak rozsáhlý jako webová stránka s několika indexy.

3.5.1 Generování seznamu surovin

Jak jsem již zmínil k hlavní pointě této webové kuchařky, využijeme proměnnou $suroviny, do které jsem uložil výsledek funkce get_data z objektu z třídy Suroviny. V této proměnné, jak již bylo řečeno v předešlé podkapitole, se nachází názvy všech surovin, které jsou obsaženy v receptech a uživatel má možnost si z nich vybrat. Do funkce, již několikrát použité, foreach jsem vložil jednoduché pole s názvy surovin a na každý průchod novou surovinou jsem nastavil generování takzvaného checkboxu. Checkboxu jsem samozřejmě přidal label pro stylizaci a také label pro název. Bohužel nejsem příliš zkušený v oblasti designování webových stránek, a tak jsem využil webové stránky getcssscan.com kde jsem si vyhlédl konkrétně styl checkboxu číslo 17. Tento styl jsem poté implementoval do svého CSS souboru, kde poté budu doplňovat styly dalších vygenerovaných divů, textů a tlačítek. Celou funkci foreach jsem obalil do divu, který následně pomocí frontend technologie zvané Javascript, který vložím na tlačítko filtru, nechám buď ukázat, nebo zmizet.

3.5.2 Generování kategorií

Jako další přichází na řadu generování kategorií, pro generování kategorií využiji funkci foreach, kterou projdu jednoduché pole se všemi názvy kategorií a na výstup funkce nastavím HTML tag div, který obsahuje text s názvem dané kategorie a také href, který zařídí, že po kliknutí na tlačítko se web změní a vypíše seznam pouze takových receptů, které jsou v dané kategorii. Samozřejmě, že href toto sám nezvládne, a tak pro získání receptů použijeme již několikrát zmíněnou funkci mysqli_query, kterým pošleme upravený SQL dotaz, který se ptá na všechny recepty obsažené v dané kategorii. Jakmile se výsledek dotazu dostaví, web vygeneruje dané recepty na místo jim určené.

3.5.3 Generování receptů

Abych navázal na předchozí podkapitolu, seznam receptů závisí na našem filtrování. Pokud filtrujeme recepty dle surovin, vygenerují se recepty, které vybrané suroviny obsahují. Pokud však ale filtrujeme recepty dle vybrané kategorie, budou vygenerovány recepty pouze z dané kategorie. O všechnu tuto práci se starají funkce mysqli_query a foreach. Aby však webová stránka věděla, kde se zrovna nacházíme a jaké recepty má zobrazovat, získáme pomocí proměnné $_GET data z URL adresy webu. Pokud se data za posledním lomítkem rovnají „?page=“ získá data za znakem „=“. Soubor index následně vyhodnotí jednu ze tří podmínek IF jako podmínku s výsledkem true a její tělo provede. Každá podmínka IF obsahuje jiné tělo, což znamená jiný výběr receptů. V podmínkách IF se ptáme, jestli se proměnná $page, ve které jsou uložena data proměnné $_GET, rovná danému textovému řetězci. Pokud se podmínka rovná číslům, vypíše se seznam receptů, obsažených v kategorii, která nese stejné ID jako bylo v proměnné $_GET. Jestliže se podmínka IF rovná textovému řetězci „home“ nebo je proměnná $_GET prázdná, vygeneruje se seznam všech receptů obsažených v databázi. Jestliže se proměnná $_GET rovná jednomu z textových řetězců, obsažených v poli textových řetězců nesoucích názvy receptů, vygenerují se podrobnosti receptu. Mezi podrobnosti receptu patří data získaná z objektu třídy Recept_suroviny a z objektu třídy Recept. Tudíž v podrobnostech receptu nalezneme informace jakožto název, seznam obsažených surovin, postup a časovou náročnost. Jak již bylo zmíněno v příštích verzích by se podrobnosti receptu mohli rozšířit o kolekci obrázků. Jestliže se proměnná $_GET rovná textovému řetězci filtr, vypíše se seznam receptů, které obsahují uživatelem vybrané suroviny. Pokud však uživatel nevybere ani jednu surovinu ze seznamu všech surovin a potvrdí

výběr tlačítkem „Filtrovat“, webová stránka vypíše chybu, že výsledek jeho hledání je bohužel marný, protože uživatel nevybral vhodné suroviny pro vypsání receptů, v našem případě konkrétně žádné.

4 Výsledek práce a závěr

Výsledkem práce je dle mého plně funkční webová aplikace chytré kuchařky, která by po pár verzích, které by jí vylepšili o několik, za mě důležitých funkcí, mohla být nasazena do ostrého provozu. Výsledkem je responzivní, z mého pohledu moderně vypadají, přístupná webová aplikace chytré kuchařky. Kuchařka umí, jak je již zmiňováno průběhu celého dokumentu, výpis receptů z databáze s různými filtry. Mezi filtry patří filtrování dle zadaných surovin, podle něhož kuchařka nese název chytrá, pokud uživatel nezadá žádnou surovinu, místo receptů se na webovou stránku vypíše chybová hláška, která říká, že nebyli nalezeny žádné výsledky hledání a vyzve uživatele k vrácení na hlavní stránku. Dále chytrá kuchařka umí filtrovat dle kategorií, ke kterým jsou recepty přiřazeny. Po kliknutí na recept, jak jsem již zmínil v praktické části dokumentace, se vypíší na webovou stránku všechny informace o daném receptu, které se nachází v databázi. Jedná se o informace, o kterých jsem se taktéž zmínil v praktické části, konkrétně v její podkapitole nesoucí název generování receptů. Ve výpisu detailních informací o receptu jsou data následující: název receptu, který je vygenerovaný jako nadpis, dále se v detailních informacích o receptu nachází seznam surovin potřebných k realizaci daného receptu společně s množstvím a jednotkou, ve které se daná surovina nachází. Jako předposlední informace je zde vygenerován postup realizace receptu a časová náročnost, kterou je myšlená předpokládaná doba výroby.