Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informatiky a kvantitativních metod

Webová aplikace pro výživové poradce

Bakalářská práce

Autor: Ivo Bauer
Studijní obor: Aplikovaná informatika

Vedoucí práce: Ing. Karel Malý, Ph.D.

Hradec Králové srpen 2022

Prohlášení:

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne 09.08.2022 *vlastnoruční podpis*

Poděkování:

Děkuji vedoucímu bakalářské práce doktoru Karlovi Malému za poskytnutí cenných rad a odborné vedení.

Anotace

Bakalářská práce se zabývá problematikou vývoje webové aplikace pro výživové poradce umožňující správu databáze klientů, tvorbu jídel a skladbu jídelníčků. Teoretická část obsahuje přehled existujících řešení a popis použitých technologií na straně serveru (backend) i klienta (frontend). Praktická část popisuje datový model, strukturu aplikace a projektu ve vývojovém prostředí, tvorbu databáze, klíčovou funkcionalitu a publikování webové stránky na hosting. K vývoji aplikace bylo použito jazyka ASP.NET Core, technologie HTML, CSS, JavaScript a framework Bootstrap.

Klíčová slova: webová aplikace, ASP.NET Core, MVC, výživové poradenství

Annotation

Title: Web Application for Nutritional Advisors

The bachelor thesis deals with the development of a web application for nutritional advisors that enables the management of a client database, the creation of meals and the composition of menus. The theoretical part contains an overview of existing solutions and a description of the technologies used on the server (backend) and client (frontend) side. The practical part describes the data model, the structure of the application and project in the development environment, database creation, key functionality, publishing and hosting a website. ASP.NET Core language, HTML, CSS, JavaScript, and Bootstrap framework were used to develop the application.

Keywords: web application, ASP.NET Core, MVC, nutritional advice

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Pro svou bakalářskou práci jsem si vybral téma *Webová aplikace pro výživové poradce*. K tomu mě inspirovala kniha *Moderní výživa ve fitness a silových sportech* od pana Ing. Lukáše Roubíka [1]. Ve společnosti je téma zdravé výživy velmi aktuální a často diskutované. Proto jsem se rozhodl propojit zdravý životní styl s informačními technologiemi a vyvinout webovou aplikaci, která by umožňovala výživovým poradcům vytvářet jídelní kategorie, recepty jídel, jídelníčky a spravovat databázi klientů. Při vývoji této aplikace jsem uplatnil znalosti získané studiem programovacích jazyků a databází.

Druhá kapitola obsahuje přehled existujících řešení a zmiňuje pozitivní a negativní stránky jednotlivých aplikací. Technologie použité při vytváření webové aplikace jsou popsány z pohledu klienta i serveru v kapitole třetí. Do podkapitoly Frontend, tedy části zobrazující se na straně klienta, jsou zahrnuty jazyky HTML, CSS, JavaScript a framework Bootstrap. Podkapitola Backend se věnuje popisu frameworku ASP.NET Core, architektury MVC, Entity Framework Core a databáze Microsoft SQL Server. Čtvrtá kapitola popisuje datový model a strukturu aplikace. V další části bakalářské práce zaměřené na implementaci aplikace je popsána struktura projektu ve vývojovém prostředí, vytvoření databáze, popis vybraných klíčových funkcionalit a publikování webové stránky na hosting.

2 Existující řešení

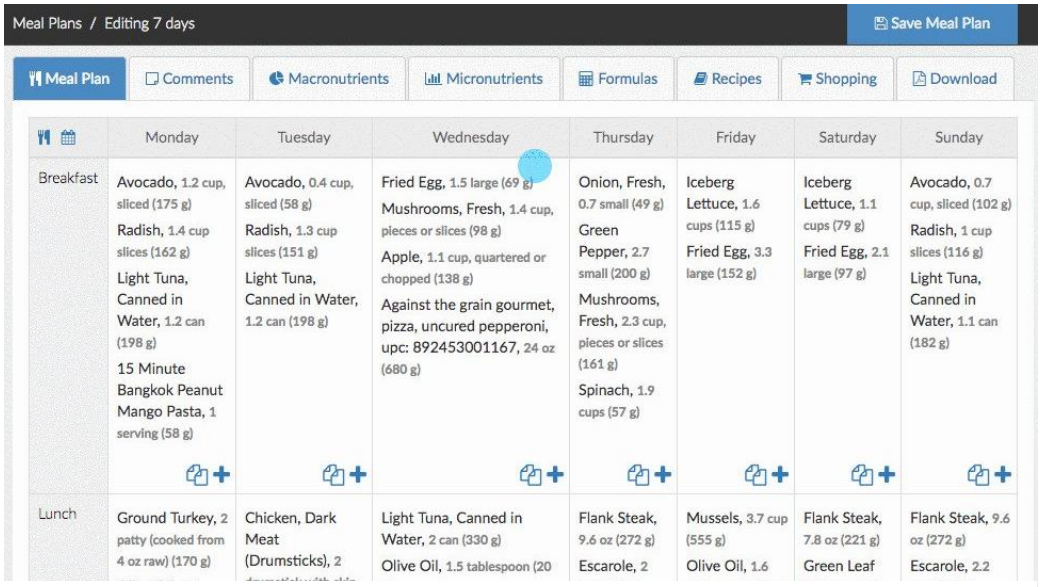
Cílem této kapitoly je prozkoumat české i zahraniční aplikace pro výživové poradce, popsat jejich silné stránky a klíčové vlastnosti. Přehled uvedených vlastností poskytne inspiraci k návrhu vlastní webové aplikace.

2.1 Zahraniční aplikace

2.1.1 **NutriAdmin**

NutriAdmin je webová aplikace vytvořená pro výživové poradce a dietology, která má systém plateb napojený na platební bránu Stripe, takže zde může poradce vidět v jakém stavu se nacházejí jednotlivé platby, případně vytvořit odkaz, který zprostředkuje novou platbu.

Systém nabízí vytváření šablon jídelníčků pro jednotlivé dny v týdnu, do kterých může uživatel přidávat vlastní jídla nebo již existující jídla, která jsou uložena v databázi (tato možnost je přístupná od verze *Popular plan*). Do jídelníčků je možné vkládat vlastní poznámky, vzorce, recepty na přípravu jídel, nebo nákupní lístek. Vše lze exportovat (např. do PDF) a poslat klientovi e-mailem, nebo stáhnout a pak dodat klientovi vlastním způsobem.



Obrázek 1: NutriAdmin jídelníček. Zdroj: [2]

Ve správě klientů jsou na jednom místě uloženy všechny vlastní poznámky o klientech, které mohou obsahovat např. informace o individuálních doporučeních, intolerancích, či jaké jídlo má nebo nemá rád. Tyto poznámky jsou pravidelně zálohovány. Zároveň je možné vytvářet pro nové klienty individuální dotazníky. Každý poradce si zde může v editoru vytvářet oddíly, otázky a různé druhy odpovědí (textové pole, checkboxy, choiceboxy). Výsledný dotazník je možné odeslat několika způsoby klientovi. Správa klientů též obsahuje kalendář, ve kterém si může nastavit pracovní období a může si v něm sjednat schůzku – klient obdrží e-mail o konání schůzky a upozornění určitý časový interval předtím, než samotná schůzka začne.

	Basic Plan	**Popular Plan**	**Professional Plan**	**Business Plan**
Cena [€]	29,99	39,99	59,99	individuální
Klientů měsíčně	10	20	40	
Poradců	1	1	1	
Online platby	ne	ano	ano	ano
Databáze receptů	ne	ano	ano	ano

Tabulka 1: NutriAdmin porovnání plánů. Zdroj: [3].

Jak je z tabulky patrné, rozdíl mezi Basic Plan a ostatními plány je především v absenci online plateb a databáze receptů. Business Plan je určen například pro firmy, výsledná cena se odvíjí od požadovaného počtu klientů a poradců. Za každého klienta se platí 1 € a poradce 10 € měsíčně navíc.

2.2 České aplikace

2.2.1 **Fitlinie**

Program je určený především pro nutriční poradce, ale mohou jej používat i běžní uživatelé. Není však zpracován formou webové aplikace, musí se do počítače instalovat. Jsou zde velmi do detailu zpracovány funkce, které by výživový poradce mohl potřebovat – vytváření vlastních receptů, jídelníčky na míru, tvorba tréninkových plánů, či podrobné statistiky a vlastní funkce. Chybí zde však platební brána a přehled plateb, tedy funkce, které by byly určitě užitečné.

Nevýhodou tohoto programu může být i fakt, že jsou veškerá data uložena lokálně na počítači, takže pokud uživatel zapomene udělat zálohu na externí zařízení a ztratí přístup k počítači nebo dojde ke smazání souborů na disku, tak nebude schopen pokračovat v práci a také může ztratit cenná data o svých klientech.

Fitlinie nabízí 7 druhů celoživotních licencí, aktualizace databáze potravin je zdarma pouze ale první rok od pořízení licence.

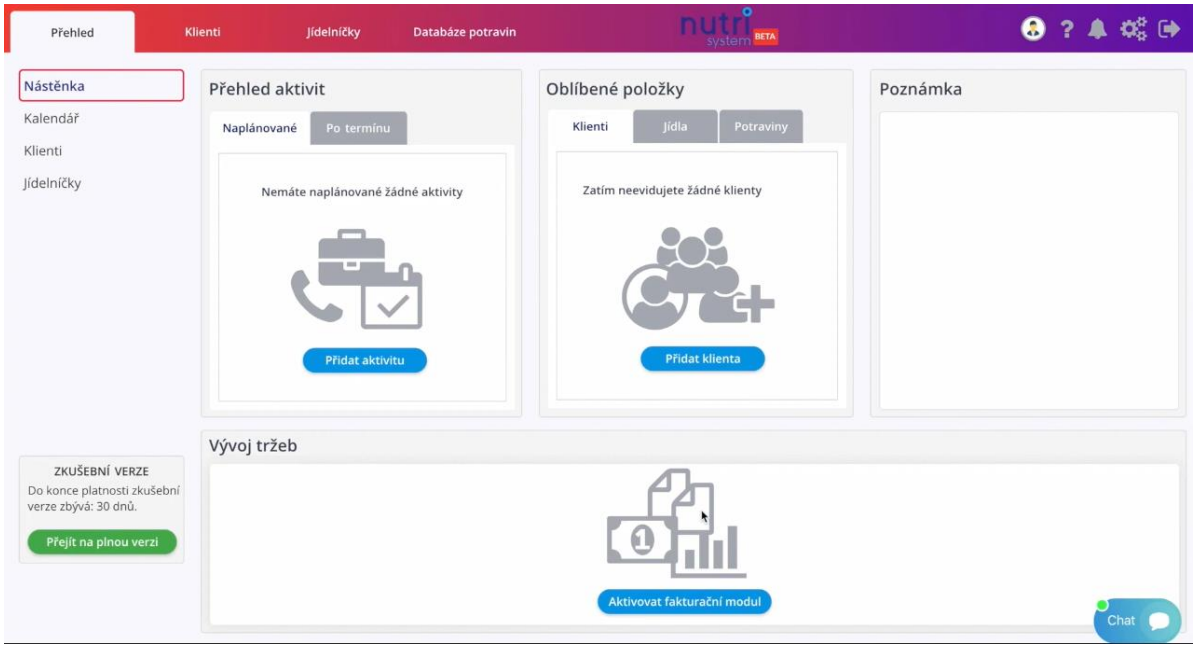
	Personal	**Family**	**Profi 1**	**Profi 2**	**Profi 3**	**Profi FULL**	**Academy**
Cena [Kč]	490	790	1290	1890	2490	3990	individuální
Cena aktualizace [Kč]	199	299	599	699	799	999	individuální
Klientů	2	5	10	20	30	neomezeně	neomezeně

Tabulka 2: Fitlinie porovnání plánů. Zdroj: [4]

Verze Academy je nadstavba verze Profi FULL a nejdůležitější rozdíl je v rozšířené databázi potravin. Každá potravina obsahuje více informací (Academy má 25 sloupců, ostatní sloupců jen 15). Ve verzi Academy je navíc možné například zadávat jednoduché sacharidy, živočišné bílkoviny, živočišné tuky nebo rostlinné tuky.

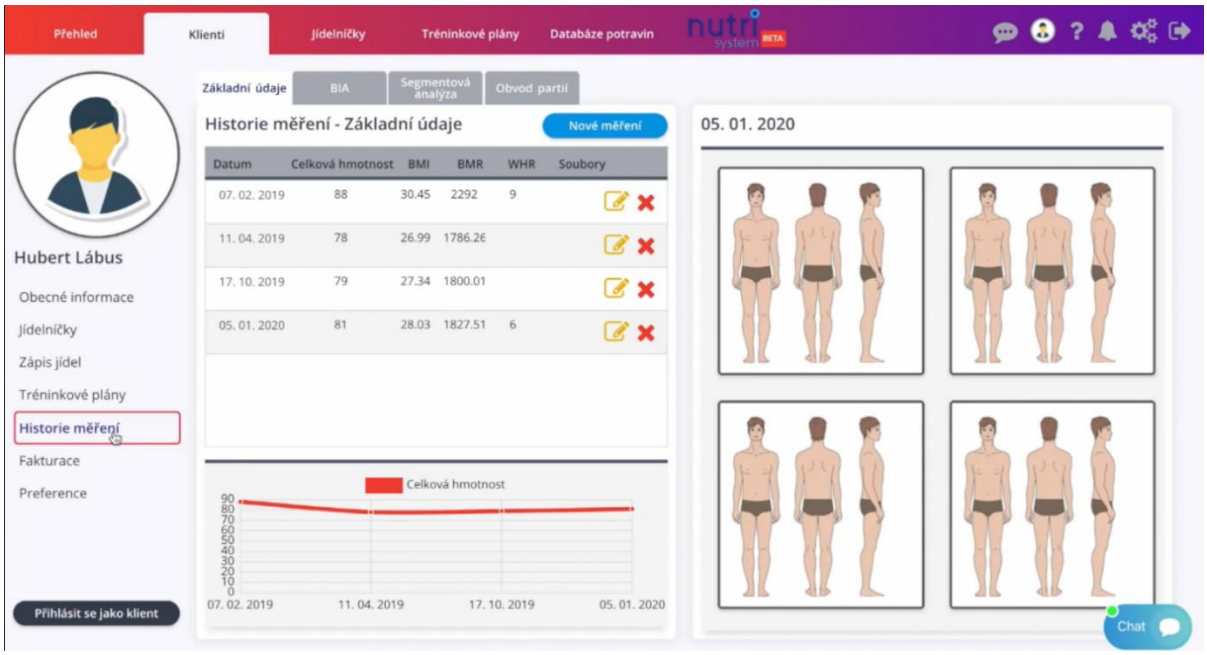
2.2.2 **Nutris**

Aplikace je určena pro spolupráci výživových poradců a fitness trenérů s jejich klienty. V nastavení je možno vypnout, zapnout, či upravit jednotlivé moduly – každý si může u většiny systému upravit prostředí tak, jak je na práci zvyklý a vyhovuje mu to. Součástí systému je i komunikace mezi klientem a poradcem formou online chatu zabudovaném v pravém dolním rohu aplikace.
V aplikaci na první pohled zaujme rozložení prvků. Působí jednoduše a přehledně. Obrázek 2 představuje obrazovku při prvním testovacím přihlášení:



Obrázek 2: Nutris obrazovka po přihlášení. Zdroj: [5]

Narozdíl od jiných aplikací se při vytváření klienta ohledně zdravotního stavu neposílá dotazník, ale po přihlášení si buď požadované informace doplní sám klient, nebo profil vyplní společně s klientem výživový poradce. V aplikaci stojí za zmínku zobrazení statistik jednotlivých měření – je možno zadat výsledky měření z několika druhů přístrojů a výsledná data se zobrazí dle nastavení v grafech.



Obrázek 3: Nutris vyhodnocení měření. Zdroj: [5]

Nutris nabízí tři cenové balíčky, které se odlišují maximálním počtem aktivních klientů. Po dosažení maximální kapacity nejdražšího balíčku lze sjednat individuální cenový plán.

	Malý balíček	**Střední balíček**	**Velký balíček**
Cena [Kč]	430	670	990
Klientů	15	35	65

Tabulka 3: Nutris porovnání plánů. Zdroj: [6]

3 Frontendové a backendové technologie

Při tvorbě webové aplikace je důležité se zaměřit na vhodné nástroje. Tato kapitola popisuje technologie a postupy použité při vývoji aplikace.

3.1 Frontend

Pojmem frontend se označuje ta část webové stránky, se kterou uživatel přímo pracuje. Obsahuje například ovládací prvky, tabulky, grafy a také fotografie. Při práci na frontendu se využívá více jazyků, které jsou popsány v následujících podkapitolách. [7]

3.1.1 **HTML**

HTML je zkratka z *HyperText Markup Language*. Jedná se tedy o značkovací jazyk, díky kterému můžeme dát webové stránce potřebnou strukturu. Skládá se z několika druhů elementů, které se rozlišují podle jejich funkce. Na začátku každého HTML dokumentu musí být uvedena definice `<!DOCTYPE html>`, aby bylo možné identifikovat, jak má prohlížeč stránku vykreslit. [8]

```
<!DOCTYPE html>
<html>
    <head>
        <title>Titulek stránky</title>
    </head>

    <body>
        <h1>Nadpis</h1>
    </body>
</html>
```

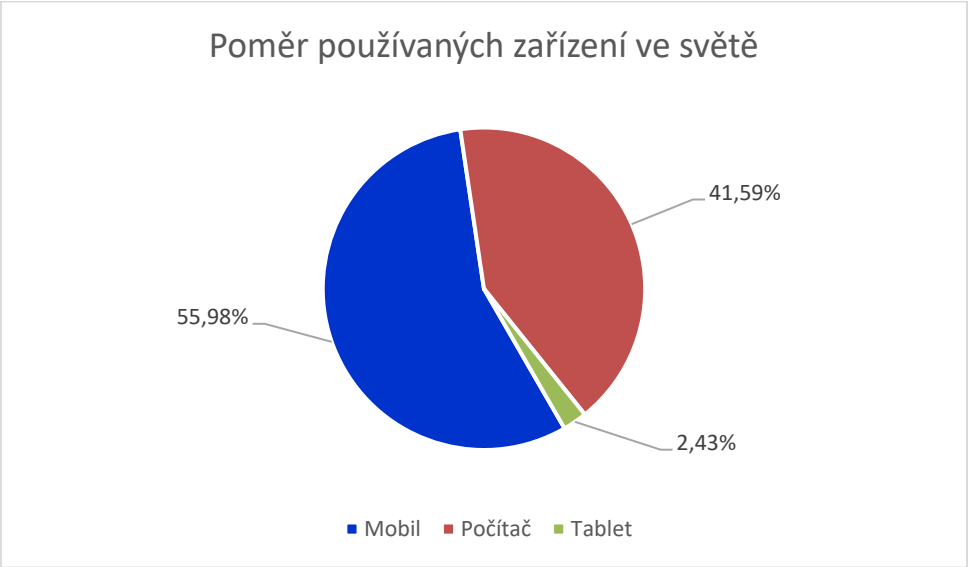
Obrázek 4: Jednoduchý HTML dokument. Zdroj: autor

Na obrázku číslo 4 je možné vidět jednoduchý HTML dokument, který obsahuje základní tagy. Tag `<html>` se stará o ohraničení celého souboru.
Dalším tagem je `<head>` o kterém píše Dušan Janovský: *„Hlavička dokumentu, která se nezobrazuje. Obsahuje nepovinně další tagy (title, meta, link, style, script aj.).“* [9]

Posledním tagem na obrázku je element `<body>`, který v sobě obsahuje veškerý uživatelský obsah k zobrazení. Může se jednat například o elementy prezentující informace jako třeba tabulky, obrázky a videa, nebo o ovládací prvky, kam patří tlačítka, textové vstupy, posuvníky či odkazy.

3.1.2 **CSS**

CSS z angličtiny *Cascading Style Sheets* je jazyk, jehož cílem je určovat vzhled jednotlivých prvků webové stránky. Mezi hlavní výhody patří snadná a přehledná úprava pravidel a také opakovaná použitelnost, protože můžeme jeden CSS soubor, ve kterém jsou uloženy formátovací pravidla, použít na libovolné množství webových stránek. S tím přímo souvisí pojem responzivní design, který znamená optimalizaci webu pro zařízení s různým rozlišením.



Obrázek 5: Poměr používaných zařízení ve světě. Zdroj: autor

Na obrázku číslo 5 je zobrazen poměr používaných zařízení ve světě při prohlížení internetových stránek z února 2022. Responzivní design je možné vytvořit právě pomocí CSS, kde lze nastavit pravidla pro určené kategorie, které si webový vývojář sám zvolí [10].

3.1.3 JavaScript

JavaScript je objektový skriptovací jazyk, který umožňuje vytváření složitějších funkcionalit na straně klienta. Může se jednat například o vykreslování grafů, zobrazování času, ale třeba i vytváření a úpravu HTML prvků dokumentu a jejich CSS formátování.

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <script src="displayInputValue.js"></script>
5      </head>
6      <body>
7          <p id="inputText"></p>
8          <input id="input"/>
9          <button onclick="displayInputValue()">Zobraz text</button>
10
11         <script>
12             function displayInputValue() {
13                 const inputValue = document.getElementById("input").value;
14                 document.getElementById("inputText").innerHTML = inputValue;
15             }
16         </script>
17     </body>
18 </html>
```

Obrázek 6: Ukázka JavaScriptu v HTML stránce. Zdroj: autor

Na obrázku číslo 6 je možné vidět ukázku implementace JavaScriptu do HTML stránky. V tagu `<body>` jsme vytvořili prázdný odstavec `<p>`, vstup `<input>` a tlačítko `<button>`. Odstavci a vstupu jsme přiřadili vlastní `id`, podle kterého bude v následujícím kódu možná identifikace a práce s těmito prvky.

Pro představu jsou zde použity dva různé přístupy použití JavaScriptu v HTML. První způsob je možný vidět na řádku číslo 4. Do hlavičky dokumentu vložíme tag `<script>`, ve kterém se uvede umístění souboru, ze kterého se při načítání stránky script načte. Druhý způsob je zobrazen na řádcích 11 až 16. V tagu `<script>` vytvoříme jednoduchou funkci `displayInputValue()`, která se vyvolá při stisknutí tlačítka. Funkce načte obsah vstupu do konstanty `inputValue`, kterou v následujícím řádku vypíše do HTML prvku odstavce, který byl nalezen pomocí jeho

`id`. Další informace o použití JavaScriptu lze nalézt v kapitole Implementace aplikace.

3.1.4 **Bootstrap**

Na domovské stránce https://getbootstrap.com/ je uvedeno, že Bootstrap je open-source framework zaměřený na frontend, který je založen na jazycích HTML, CSS a JavaScript. Mezi jeho hlavní výhody patří zejména responzivní design, rychlost a kompatibilita s nejnovějšími prohlížeči, mezi které patří například Google Chrome, Mozilla Firefox, Microsoft Edge a Opera [11].

3.2 Backend

Pojem backend označuje ty části webové stránky, které nejsou viditelné uživatelem. Backend má na starost správu databáze a práci s daty. Komunikace s uživatelem probíhá přes frontend, který umožňuje prezentovat data získaná z databáze a také ukládat uživatelská data zpět do databáze prostřednictvím ovládacích prvků.

3.2.1 **ASP.NET Core**

ASP.NET core je framework vyvíjený společností Microsoft. Tento framework je multiplatformní a jak je uvedeno na stránkách společnosti, tak i velmi výkonný. Umožňuje vytvářet webové stránky jak pro malé internetové služby, tak i pro rozsáhlejší projekty [12].

Dříve tento framework nesl označení ASP.NET a to až do čtvrté verze, poté došlo k přejmenování a vznikl ASP.NET Core. V této nové řadě došlo k nárůstu výkonu, rozšíření podpory více platforem z původního operačního systému MS Windows o MacOS a GNU/Linux [13]. V prosinci 2021 byla vydána verze ASP.NET Core 6.0.1 [14].

3.2.2 Architektura MVC

MVC, tedy Model-View-Controller je označení pro architekturu webu, která odděluje frontend a backend tak, aby se tyto dvě součásti co nejméně ovlivňovaly. Díky tomu je následné rozšíření aplikace snazší a přehlednější [15].
MVC je rozdělen na tři základní části: Model, view a controller.

- Model pracuje s daty. Na vyžádání od controlleru může model například z databáze načíst nebo upravit požadovaná data.
- View se stará o vytvoření uživatelského rozhraní (GUI) a prezentaci dat získaných z modelu.
- Controller reaguje na akce uživatele. Vybírá konkrétní model, se kterým pracuje a View, který uživateli následně data zobrazí.

3.2.3 Entity Framework Core

Dokumentace je dostupná na adrese https://docs.microsoft.com/cs-cz/ef/. Entity Framework Core je technologie vyvíjená společností Microsoft a jejím hlavním cílem je usnadnění práce s databází. Využívá objektově-relačního mapování. To znamená, že je možné pracovat s relační databází pomocí objektového programování [16]. Podporuje celou řadu databází, jako například MS SQL Server, MySQL, Oracle DB a PostgreSQL [17].

Obrázek 7: Ukázka kódu třídy FoodCategory. Zdroj: autor

Na obrázku číslo 7 je možné vidět deklaraci třídy `FoodCategory` s využitím Entity Frameworku. První vlastností třídy je `Id`, která je typu `int`. Jedná se o primární klíč této třídy, protože je nad ním uveden atribut `Key`. Druhou vlastností je `CategoryName`. Atributem `DisplayName` specifikujeme název, který se bude v aplikaci zobrazovat. Atribut `Required` určuje, že daná vlastnost je povinná (nemůže mít hodnotu `null`). Můžeme upravit hlášku, která se zobrazí, když zapomene uživatel zadat jméno kategorie zadáním parametru `ErrorMessage`. V poslední části třídy vkládáme cizí klíč na třídu `User`, která rozšiřuje tabulku `AspNetUsers`, která je podrobněji popsaná v kapitole Návrh aplikace.

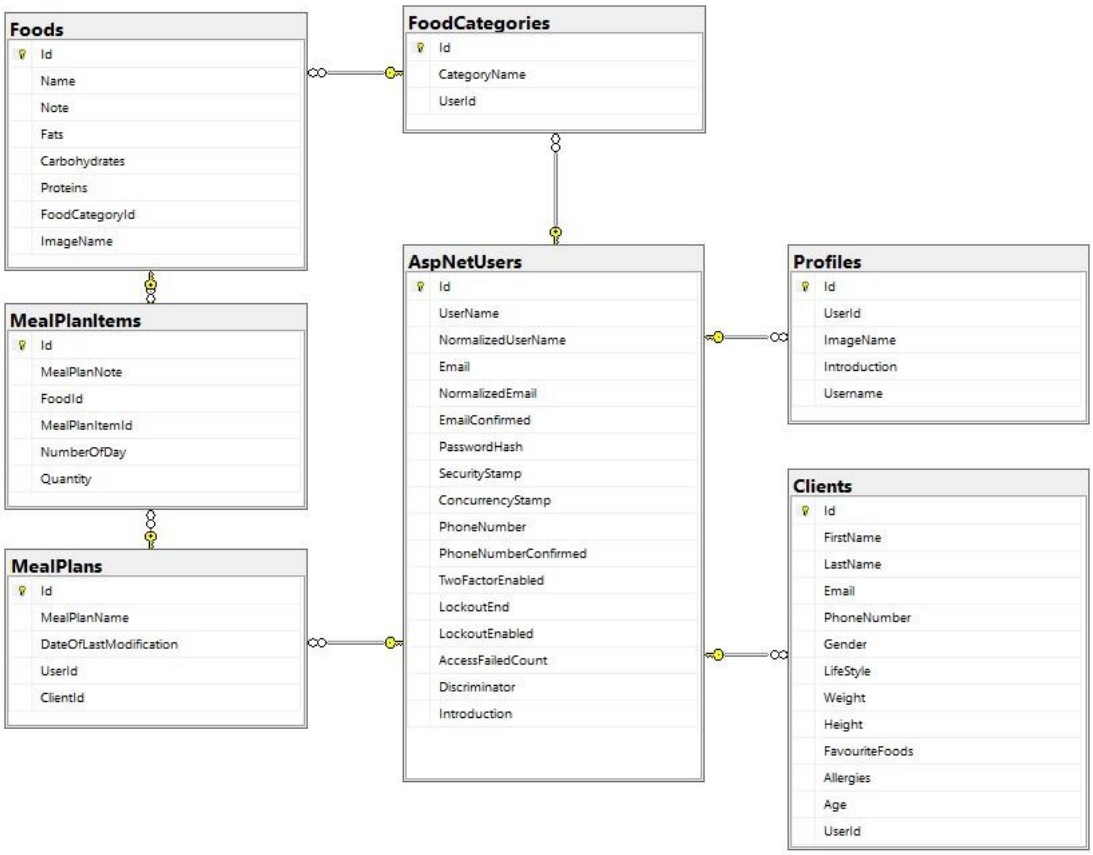
3.2.4 Microsoft SQL Server

Microsoft SQL Server je relační databázový systém. Jak je napsáno na stránkách vydavatele, relační databáze v tomto systému se skládá z tabulek, které obsahují strukturovaná data. Každá tabulka má sloupce a řádky. Sloupce v relační databázi znamenají vlastnosti. Ty mají svůj název, datový typ a rozsah. Řádky této tabulky pak tvoří samotné záznamy [18]. Tomuto tématu je věnováno více v kapitole Model.

4 Návrh aplikace

4.1 Model

Třídy vytvořené v modelu za pomocí Entity Framework reprezentují tabulky v databázi. Při vývoji aplikace byl zvolen způsob Code First, což znamená, že se nejprve musí vytvořit model a následně se databáze automaticky vygeneruje podle tříd tohoto modelu. Tyto třídy mohou obsahovat několik vlastností, přičemž každé vlastnosti odpovídá datový typ, který definuje obor hodnot, jakých může daná vlastnost nabývat.



Obrázek 8: Relační model. Zdroj: autor

4.1.1 Tabulka AspNetUsers

S využitím ASP.NET Core Identity byla vytvořena tabulka `AspNetUsers`, do které se ukládají data výživových poradců. Vlastnost `Id` je primárním klíčem uživatele. Primární klíč je jedinečná hodnota, podle které je možné bezpečně odlišit konkrétní záznam v tabulce. Tabulka obsahuje sloupce pro jméno (`FirstName`), příjmení (`LastName`), e-mail (`Email`), údaj o tom, zda byl při registraci účet ověřen za pomocí kliknutí na odkaz (`EmailConfirmed`), telefonní číslo (`PhoneNumber`) a také zašifrované heslo (`PasswordHash`).

Do databáze se tedy neukládá heslo uživatele, ale textový řetězec získaný aplikací šifrovacích algoritmů na zvolené heslo (tzv. hash). Tento řetězec nemá s původním heslem žádnou podobnost a zpětné rozšifrování není možné, takže i kdyby ho útočník získal, bude pro něj velice složité odhalit heslo uživatele. Dle dokumentace [19] se používá kombinace hashovacích algoritmů PBKDF2 a HMAC-SHA512.

Vlastnost	Datový typ
Id	nvarchar(450)
FirstName	nvarchar(256)
LastName	nvarchar(256)
Email	nvarchar(256)
Emailconfirmed	bit
PasswordHash	nvarchar(MAX)
PhoneNumber	nvarchar(15)

Tabulka 4: Tabulka AspNetUsers. Zdroj: autor

V tabulce se používá datový typ `nvarchar` pro textové řetězce a dále `bit`, který může nabývat hodnot 0, 1 a `null`.

4.1.2 Třída Client

Tato třída obsahuje data o klientovi daného výživového poradce. První vlastností je primární klíč (`Id`), dále cizí klíč poradce, ke kterému daný klient patří (`UserId`),

jméno (`FirstName`), příjmení (`LastName`), e-mail (`Email`), telefon (`PhoneNumber`), pohlaví (`Gender`), zdravotní stav (`HealtStatus`), životní styl (`LifeStyle`), váha (`Weight`), výška (`Height`), možnost vložit oblíbená jídla (`FavouriteFoods`), alergie na jídlo (`Allergies`).

Vlastnost	Datový typ
Id	int
UserId	nvarchar(450)
FirstName	nvarchar(50)
LastName	nvarchar(50)
Email	nvarchar(256)
PhoneNumber	nvarchar(15)
Age	int
Gender	nvarchar(10)
LifeStyle	nvarchar(1000)
Weight	int
Height	int
FavouriteFoods	nvarchar(1000)
Allergies	nvarchar(1000)

Tabulka 5: Třída Clients. Zdroj: autor

V tabulce je u primárního klíče použit datový typ `int`, který může nabývat hodnot od -2 147 483 648 do 2 147 483 647 [20].

4.1.3 **Třída FoodCategory**

Třída `FoodCategory` definuje kategorii jídel pro potřeby výživového poradce. Tato třída má tři vlastnosti. První je primární klíč (`Id`). Druhou vlastností je jméno kategorie (`CategoryName`) a poslední vlastností je cizí klíč `UserId` z třídy `AspNetUsers` označující poradce, který danou kategorii vytvořil.

Vlastnost	Datový typ
Id	int
CategoryName	nvarchar(50)
UserId	nvarchar(450)

Tabulka 6: Třída FoodCategory. Zdroj: autor

4.1.4 **Třída Food**

Třída `Food` reprezentuje samotné jídlo. Skládá se z primárního klíče (`Id`), názvu jídla (`Name`), poznámky vysvětlující například návod na přípravu jídla či různá omezení (`Note`), počtu tuků (`Fats`), sacharidů (`Carbohydrates`), bílkovin (`Protein`), názvu souboru přidruženého obrázku včetně přípony (`ImageName`), cizího klíče odkazujícího se na kategorii jídla (`FoodCategoryId`) a cizího klíče (`UserId`) z tabulky `AspNetUsers`.

Vlastnost	Datový typ
Id	int
Name	nvarchar(50)
Note	nvarchar(1000)
Fats	double
Carbohydrates	double
Protein	double
ImageName	nvarchar(255)
FoodCategoryId	int
UserId	nvarchar(450)

Tabulka 7: Třída Food. Zdroj: autor

4.1.5 **Třída MealPlan**

Třída `MealPlan` představuje jídelníček. Skládá se z primárního klíče (`Id`), další vlastností je jméno jídelníčku (`Name`), datum poslední úpravy (`DateOfLastModification`) a cizí klíč poradce, který jídelníček vytvořil (`UserId`).

Vlastnost	Datový typ
Id	int
Name	nvarchar(50)
DateOfLastModification	datetime2(7)
UserId	nvarchar(450)

Tabulka 8: Třída MealPlan. Zdroj: autor

4.1.6 **Třída MealPlanItem**

Třída `MealPlanItem` definuje konkrétní položku daného jídelníčku. Primárním klíčem je `Id` a cizími klíči jsou `FoodId`, `MealPlanId` a `UserId`. Vlastnosti třídy jsou poznámky (`MealPlanItemNote`) a množství jídla (`FoodQuantity`).

Vlastnost	Datový typ
Id	int
MealPlanItemNote	nvarchar(50)
FoodQuantity	int
FoodId	int
MealPlanId	int
UserId	nvarchar(450)

Tabulka 9: Třída MealPlanItem. Zdroj: autor

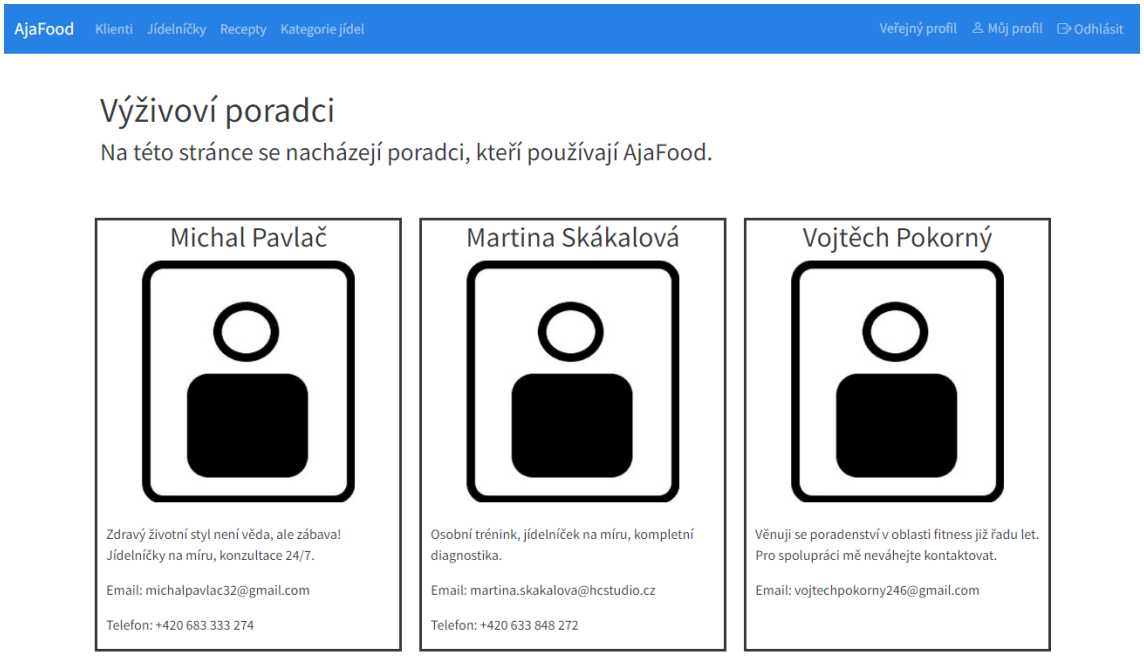
4.2 Struktura aplikace

Webová aplikace je rozdělena na tři části – záhlaví, tělo stránky a zápatí. V záhlaví se nachází navigační panel k ovládání aplikace. V těle stránky se zobrazuje uživatelem zvolený obsah. Zápatí obsahuje název webu a informaci, že se jedná o bakalářskou práci.

V levé části navigačního panelu se nachází tyto záložky:

- AjaFood
- Klienti
- Jídelníčky
- Recepty
- Kategorie jídel

Při kliknutí na záložku **AjaFood** se zobrazí výchozí stránka webové aplikace se seznamem poradců, kteří mají vyplněný profil (jedná se o fiktivní osoby). Tato stránka jako jediná z výše zmíněných nevyžaduje k zobrazení obsahu přihlášení uživatele.

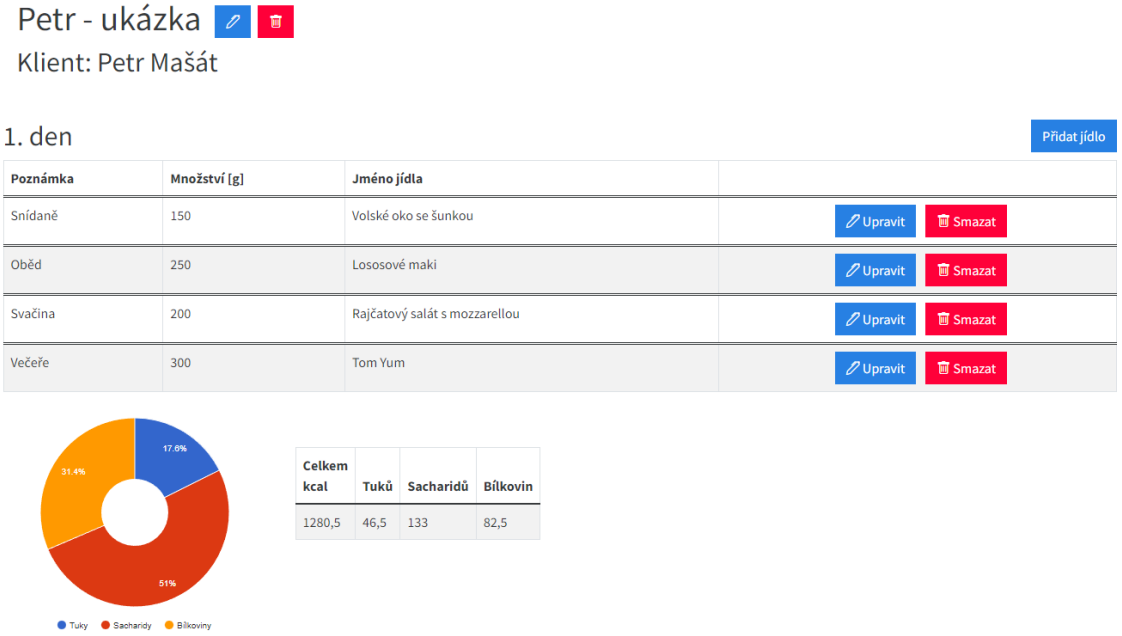


Obrázek 9: Výchozí stránka aplikace. Zdroj: autor

V záložce **Klienti** se přihlášenému uživateli zobrazí v tabulce seznam vlastních přidaných klientů (jedná se o fiktivní osoby). Do tohoto seznamu je možné přidat nového klienta, případně upravit již existujícího klienta.

Obrázek 10: Přehled klientů. Zdroj: autor

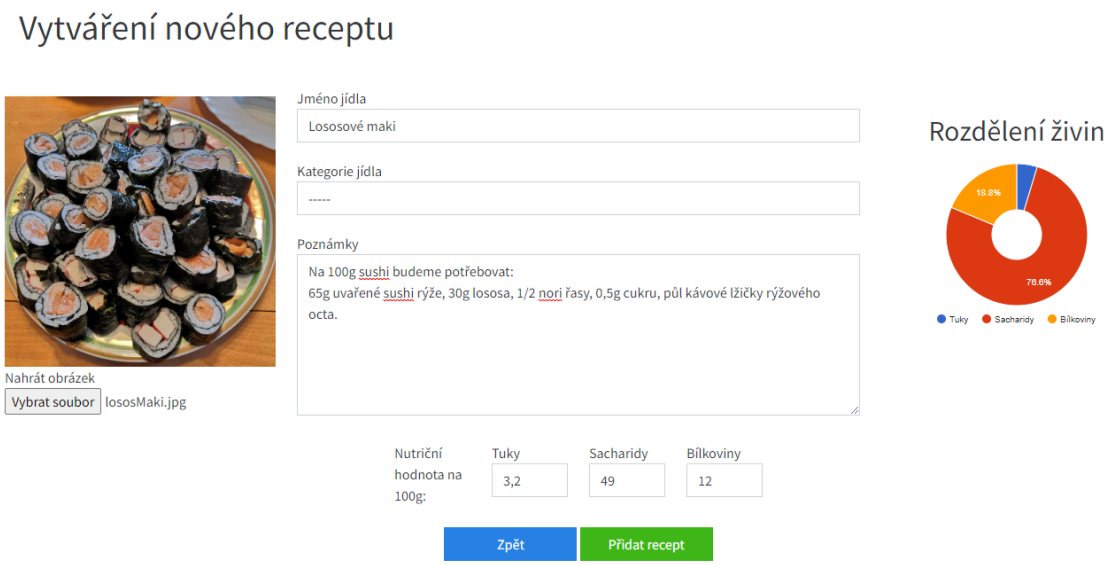
Záložka **Jídelníčky** umožňuje tvorbu jídelních plánů pro konkrétního klienta. Do jídelního plánu lze přidávat dny a do těch již jednotlivá jídla. U každého jídla je určeno množství, poznámka (například zda se jedná o snídani, svačinu atd.) a také nutriční hodnota, tedy počet kalorií, bílkovin, sacharidů a tuků přepočtených na zadané množství v gramech. Za každým dnem se nachází součet nutričních hodnot a koláčový graf znázorňující poměr bílkovin, sacharidů a tuků. Výpočet kalorií vychází z dat gymbeam.cz [21]



Obrázek 11: Ukázkový jídelníček. Zdroj: autor

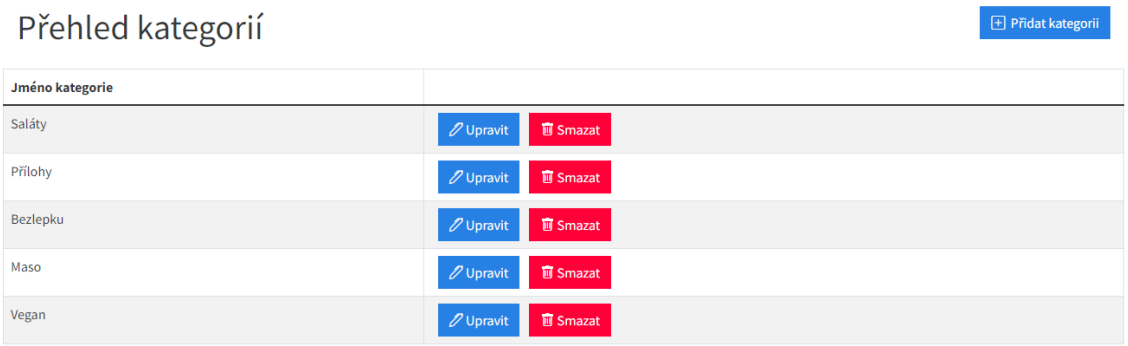
V záložce **Recepty** může uživatel vytvářet, zobrazovat či upravovat svá jídla. Na obrázku číslo 12 je vidět vytváření nového receptu. Je nutné zadat jméno jídla, zvolit kategorii jídla a nutriční živiny (bílkoviny, sacharidy a tuky). Volitelná je pak

poznámka, kam může uživatel zadat například postup přípravy pokrmu a případně fotografie jídla.



Obrázek 12: Vytváření receptu. Zdroj: autor

Poslední záložkou na levé straně navigačního panelu je **Kategorie jídel**, kde je možné jídla roztřídit do kategorií podle toho, jak to bude uživateli vyhovovat (viz obrázek číslo 13). Hlavní výhodou může být například zamezení možnosti, že by výživový poradce omylem zahrnul klientovi do jídelníčku jídlo, na které má alergii.



Obrázek 13: Přehled kategorií. Zdroj: autor

V pravé části navigačního panelu se nachází tyto záložky:

- Registrovat a Přihlásit
- Můj profil a Odhlásit

Tato část navigačního panelu je určena k přihlášení do aplikace. Nepřihlášeným uživatelům se zobrazí záložky **Registrovat** a **Přihlásit**. Při registraci nového uživatele je potřeba vyplnit email, heslo a zopakovat heslo pro případ, že by se uživatel spletl a hned po registraci by přišel o přístup ke svému účtu. Po přihlášení uživatele se předchozí dvě záložky nahradí záložkami **Můj profil** a **Odhlásit**. V uživatelském profilu lze změnit telefonní číslo a přihlašovací údaje (e-mail a heslo).

5 Implementace aplikace

K vývoji aplikace jsme použili integrované vývojové prostředí Visual Studio Community 2022 verze 17.1.4 vytvořené společností Microsoft. Při vytváření nového projektu jsme zvolili šablonu „Webová aplikace ASP.NET Core (Model-View-Controller)“. Nejdříve zvolíme jméno projektu a cestu, kam se má nový projekt uložit a poté přejdeme na další stranu s nastavením, kde si vybereme v tuto chvíli nejnovější verzi frameworku .NET 6.0 a protože chceme v aplikaci používat již existující a bezpečný způsob přihlašování, zvolíme u typu ověřování možnost **Individuální účty**.

5.1 Struktura projektu

Po vytvoření nového projektu vývojové prostředí vygeneruje složky a soubory, které jsou potřebné nebo užitečné pro chod aplikace. V následujících podkapitolách je popsán význam složek a souborů, které jsou pro naši aplikaci klíčové. Názvy jsou ponechány v angličtině, přestože lze toto vývojové prostředí nastavit na český jazyk. Překlad názvů totiž není kompletní a část složek zůstává nepřeložena.

Dependencies

Tato složka je určena k správě balíčků NuGet, které jsou tímto projektem využívány. NuGet je, jak se píše na stránkách vývojáře Microsoft, správce balíčků pro rozhraní .NET. Tento nástroj umožňuje vytvářet a využívat v projektech balíčky z jednoho centrálního úložiště [22]. Pokud klikneme pravým tlačítkem myši na tuto složku, zvolíme **Spravovat balíčky NuGet**, zobrazí se seznam již nainstalovaných balíčků. Jednoduchým způsobem je možné do projektu přidat nový balíček, případně aktualizovat již používaný. To je dobré zejména pro bezpečnost aplikace, protože kromě nové funkcionality dochází často i k nápravám chyb, které mohly být bezpečnostním rizikem.

Properties

V této složce se nachází podsložka PublishProfiles, kam se ukládají soubory s příponou pubxml. Tyto soubory obsahují informace potřebné k publikování aplikace na webový hosting. Dále složka Properties obsahuje soubor launchSettings.json, ve kterém jsou uložena nastavení k ladění aplikace. Můžeme zde nalézt například informace o tom na jakém portu se spustí aplikace s IIS Express.

wwwroot

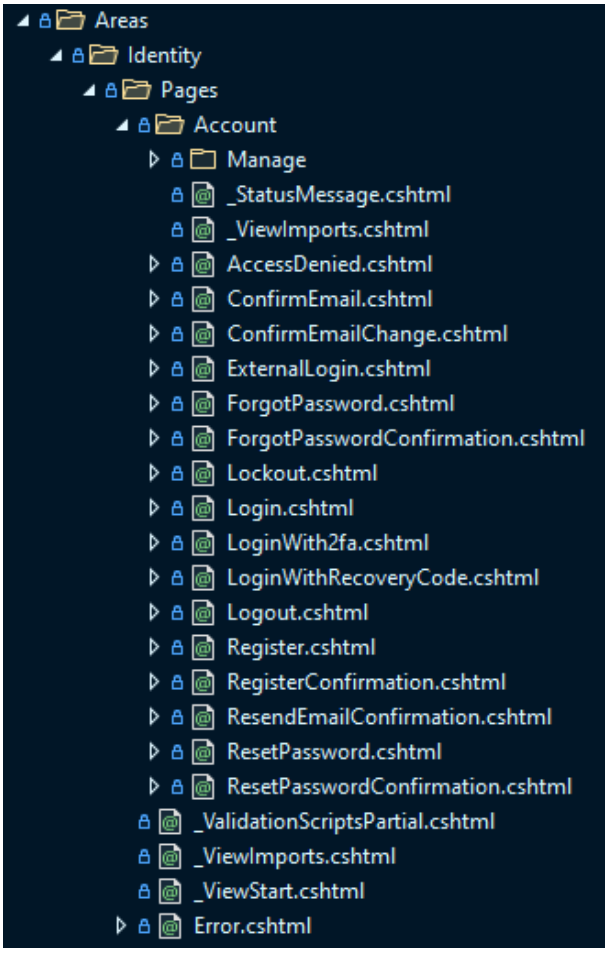
Tato složka je výchozí kořenovou složkou na webovém serveru, ke které lze přistupovat z aplikace, lze v ní vytvářet podsložky a soubory například pro ukládání obrázků či dokumentů. V této složce se nacházejí tři automaticky vytvořené podsložky (css, js a lib), které jsou potřeba pro správný chod aplikace.

V podsložce css jsou uloženy šablony použitých kaskádových stylů. V podsložce js jsou uloženy JavaScriptové soubory. V podsložce lib se nacházejí skripty knihoven, mezi které patří například Bootstrap a jQuery. V neposlední řadě se ve složce wwwroot nachází i soubor favicon.ico, který při načtení stránky vedle adresního řádku nastaví ikonu webu.

Obrázek 14: Složka wwwroot. Zdroj: autor

Areas

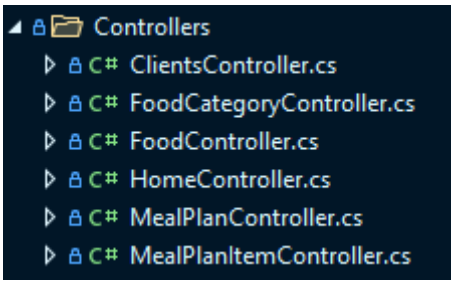
Složka Areas obsahuje automaticky vygenerované soubory frameworku ASP.NET Core Identity. Tyto soubory velmi usnadňují práci s uživatelskými účty, protože již mají z většiny implementovány funkce, mezi které patří například registrace, přihlášení, odhlášení a změnu osobních údajů.



Obrázek 15: Složka Areas. Zdroj: autor

Controllers

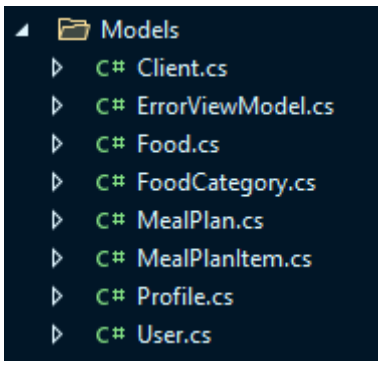
V této složce jsou uloženy kontrolery, jejichž úkolem je zpracovat příchozí požadavky HTTP z aplikace, mezi které patří například GET a POST.



Obrázek 16: Složka Controllers. Zdroj: autor

Models

Ve složce Models se nacházejí třídy jednotlivých entit, na jejichž základě se automaticky vygenerují příslušné databázové tabulky. V kapitole Entity Framework Core je blíže popsán postup vytvoření těchto tříd.



Obrázek 17: Složka Models. Zdroj: autor

Data

Ve složce Data se nachází soubor `ApplicationDbContext.cs`, díky kterému realizujeme propojení s SQL databází. V tomto souboru musí být zahrnuty všechny třídy modelu, které budou v databázi představovat tabulky.

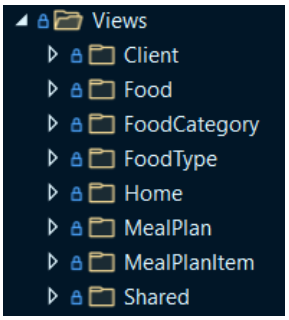
Migrations

V této složce jsou uloženy záznamy jednotlivých migrací. Jak se píše na stránkách vývojáře, důvodem pro vytváření těchto záznamů je fakt, že se s vývojem aplikace může změnit i databáze tím, že je například třeba přidat, nebo upravit sloupec již existující tabulky. Pomocí migrací můžeme postupně aktualizovat databázi, aby byla synchronizována s datovým modelem aplikace [23]. Seznam všech aplikovaných migrací je možné nalézt i v samotné databázi. Tabulka s těmito záznamy se jmenuje `__EFMigrationsHistory`.

Views

Do této složky se ukládají pohledy, jejichž úkolem je prezentace dat získaných z kontroleru. Využívá se přitom značkovacího jazyka HTML a syntaxe Razor, která umožňuje používat jazyk C# ještě dříve, než se vygeneruje výsledný HTML dokument. Díky tomu je možné používat například cykly a podmínky. Tento soubor

má příponu cshtml a je v projektu uložen v podsložce `Views`, která se jmenuje dle příslušného Controlleru. V případě `HomeController` budou tedy pohledy uloženy ve složce `Views/Home`.



Obrázek 18: Složka Views. Zdroj: autor

appsetings.json

Tento soubor obsahuje nastavení aplikace. Dají se zde deklarovat i globální proměnné. Nad rámec vygenerovaných nastavení je zde uložen řetězec k připojení do databáze.

Program.cs

Tento soubor je zodpovědný za start aplikace. Definujeme zde používané služby, mezi které patří připojení k databázi, používání kontrolérů s pohledy a také autorizace.

5.2 Vytvoření databáze

K připojení a následnému prohlížení dat v databázi budeme používat nástroj SQL Server Management Studio vyvíjený společností Microsoft. Při spuštění tohoto nástroje se zobrazí přihlašovací okno, kde u prvního řádku Server Type vybereme možnost Server Engine. V druhém řádku Server Name ponecháme název `(LocalDb)\MSSQLLocalDB`. Poslední řádek v přihlašovacím okně s názvem Authentication slouží k výběru ověřování přístupu. Zvolíme možnost **Windows Authentication** a poté stiskneme tlačítko **Connect**, pomocí kterého se připojíme k lokálnímu serveru.

Aby bylo možné vytvořit a využívat databázi, otevřeme projektový soubor `appsetings.json` a přidáme do něj nové nastavení `ConnectionString`, tedy připojovací řetězec. Jak je z obrázku 19 patrné, tento řetězec se skládá ze tří parametrů. Prvním je název serveru, který je uveden v SQL Server Management Studio při spuštění. Druhým parametrem je název databáze. Ten si lze zvolit libovolně, pro tento projekt jsme zvolili název `AFDB`. Posledním je parametrem je `Trusted_Connection`.

```
"ConnectionStrings": {
  "DefaultConnection": "Server=(LocalDb)\\MSSQLLocalDB;Database=AFDB;Trusted_Connection=True;"
},
```

Obrázek 19: Připojovací řetězec. Zdroj: autor

Dalším krokem je vytvoření jednotlivých databázových tabulek. V kapitole Model je detailně popsán model naší databáze. V kapitole Entity Framework Core je popsán způsob implementace tabulky pomocí třídy. Ostatní tabulky databáze se vytvářejí stejným způsobem. Jakmile budou ve složce `Models` vytvořeny všechny potřebné třídy, je nutné je přidat do souboru `ApplicationDbContext.cs`, aby se tyto třídy mohly namapovat na tabulky databáze. Na obrázku číslo 20 je možné vidět způsob přidání tabulky `Foods` do databáze. Tabulku definujeme pomocí generického typu `DbSet` s parametrem `Food` a pojmenujeme ji v množném čísle `Foods`.

```
13 references
public DbSet<Food> Foods { get; set; }
```

Obrázek 20: Definice tabulky. Zdroj: autor

Po nadefinování všech tabulek přejdeme do souboru `Program.cs`, kde použijeme námi vytvořený `ApplicationDbContext` a nastavíme, aby se při připojení k databázi použil připojovací řetězec s názvem `DefaultConnection`. Dále vytvoříme migraci. Klikneme na záložku **Nástroje**, poté **Správce balíčků NuGet** a nakonec **Konzola Správce Balíčků**. Ve spodní části okna se otevře příkazový řádek, kam vložíme příkaz `add-migration InitialMigration`, kde `InitialMigration` je název migrace. Dojde k vytvoření a zobrazení této migrace. K názvu se automaticky přidá časová značka s datem vytvoření.

Posledním krokem je vložení příkazu `update-database`. Pokud se nezobrazila žádná chybová hláška, můžeme přejít do nástroje SQL Server Management Studio a ověřit, zda se databáze opravdu vytvořila. Přihlásíme se, v Object Explorer rozbalíme složku Databases a měli bychom zde vidět jméno vytvořené databáze `AFDB`. Po rozkliknutí vidíme složku Tables, ve které jsou zobrazeny všechny tabulky naší databáze.

5.3 Registrace a přihlášení

K registraci nových uživatelů a přihlášení do aplikace budeme využívat framework ASP.NET Core Identity. Na obrázku číslo 15 v kapitole Struktura projektu vidíme velké množství souborů týkajících se akcí, které uživatel může vykonat. V tomto projektu budeme nejvíce využívat registrace (`Register.cshtml`), přihlášení (`Login.cshtml`) a změny hesla (`ChangePassword.cshtml`). Tyto funkcionality jsou již implementovány, bude ale třeba ještě upravit registraci tak, aby při založení nového účtu přišel ověřovací email s odkazem pro ověření totožnosti.

K odesílání e-mailu se používají NuGet balíčky MailKit a MimeKit. Do souboru `Areas/Pages/Account/Register.cshtml.cs` jsme přidali metodu `SendEmail`, která má dva parametry typu `string`. Prvním parametrem je proměnná `e-mail`. Předává se email uživatele použitý při registraci. Druhým parametrem je proměnná `callbackUrl`, kde je uložen odkaz na ověření účtu.

```
private static void SendEmail(string email, string callbackUrl)
{
    var confirmationEmail = new MimeMessage();
    confirmationEmail.From.Add(MailboxAddress.Parse("ajafoodemailsender@seznam.cz"));
    confirmationEmail.To.Add(MailboxAddress.Parse(email));
    confirmationEmail.Subject = "Ověření emailu";
    confirmationEmail.Body = new TextPart(MimeKit.Text.TextFormat.Html) { Text = $"Svůj účet v Ajafood ověříte kliknutím na " +
        $"následující odkaz: {HtmlEncoder.Default.Encode(callbackUrl)}." };

    using (var emailClient = new SmtpClient())
    {
        emailClient.Connect("smtp.seznam.cz", 465, MailKit.Security.SecureSocketOptions.Auto);
        emailClient.Authenticate("ajafoodemailsender@seznam.cz", "q33vPtLF5s3zPpx6E!Jnsvk");
        emailClient.Send(confirmationEmail);
        emailClient.Disconnect(true);
    }
}
```

Obrázek 21: Odeslání ověření účtu e-mailem. Zdroj: autor

V této metodě vytvoříme instanci třídy `MimeMessage` nazvanou `confirmationEmail` a nastavíme jí potřebné vlastnosti, mezi které patří `From` (odesílatel e-mailu), `To` (příjemce e-mailu), `Subject` (předmět) a `Body` (text e-mailu). Pro účely odesílání e-mailů vytvoříme na webové adrese seznam.cz e-mailovou schránku s adresou ajafoodemailsender@seznam.cz. Do vlastnosti `From` tedy uvedeme tuto adresu.

Dále vytvoříme instanci třídy `SmtpClient` nazvanou `emailClient`, pomocí které se připojíme k SMTP serveru. K připojení bude nutné znát adresu serveru a port. Na stránkách Seznamu byla uvedena adresa serveru `smtp.seznam.cz` a port `465`. Tento port je zabezpečen pomocí SSL [24]. Ověření totožnosti probíhá voláním metody `Authenticate`, která má dva parametry. Prvním je `userName` (námi vytvořená e-mailová schránka) a `password` (přístupové heslo k e-mailové schránce). Poté už proběhne samotné poslání ověření registrace a odpojení od serveru. Uživatel obdrží e-mail obsahující odkaz a jakmile na něj klikne, dojde k zobrazení stránky o potvrzení a do tabulky `AspNetUsers` se do sloupce `EmailConfirmed` zapíše hodnota `True`.

5.4 Klient

Tato část popisuje vybrané funkce související se správou klientů. Popis implementace je u každé podkapitoly rozdělen na dvě části, a to na funkci v kontroleru a jemu odpovídající pohledy.

Zobrazení všech klientů

V kontroleru `ClientController.cs` se nachází metoda `Index`, jejíž úkolem je předat pohledu všechny klienty přihlášeného uživatele. Z toho vyplývá, že je třeba zamezit, aby metodu mohl volat kdokoliv bez přihlášení. To je ošetřeno vložením atributu `[Authorize]` před samotnou definici metody, jak je vidět na obrázku číslo 22.

```
[Authorize]
Počet odkazů: 3
public async Task<IActionResult> Index()
{
    string userId = User.FindFirstValue(ClaimTypes.NameIdentifier);
    IEnumerable<Client> clients = await _context.Clients.Where(d => d.UserId == userId).ToListAsync();
    return View(clients);
}
```

Obrázek 22: Zobrazení všech klientů uživatele. Zdroj: autor

V těle metody získáme `Id` přihlášeného uživatele, které uložíme do proměnné `userId`. Dalším krokem je pak získání všech klientů daného uživatele. Vytvoříme si proměnnou generického typu `IEnumerable<T>`, který má parametr `Client` a nazveme ji `clients`. Prohledáme tabulku `Clients` a uložíme všechny výsledky, jejichž hodnota se ve sloupci `UserId` rovná `userId`. Tyto výsledky poté předáme pomocí příkazu `return View (clients)` k zobrazení pohledu.

Tomuto kontroleru odpovídá pohled uložený ve `Views/Client/Index.cshtml`. Na začátku souboru je nutné pro správné fungování pohledu specifikovat model, jak je zobrazeno na obrázku číslo 23.

```
@model IEnumerable<BPWeb.Models.Client>
```

Obrázek 23: Specifikace modelu pohledu. Zdroj: autor

Aby byla stránka přehlednější, tak si část nadpisu a tlačítka rozdělíme na dva stejné díly s využitím frameworku Bootstrap. Ten rozděluje každý řádek na 12 sloupců. Pokud si tedy vytvoříme dva tagy `<div>`, kterým nastavíme třídu `col-6`, tak každému bloku bude odpovídat přesně polovina obrazovky. Do prvního bloku vložíme nadpis `<h1>` a pojmenujeme ho `Přehled klientů`, do druhého bloku vložíme odkaz pro vytvoření nového uživatele. Na obrázku číslo 24 je možné vidět strukturu tohoto odkazu. V tagu `<div>` je zmíněná třída `col-6` a `text-end`, která se postará o zarovnání obsahu směrem doprava.

```
<div class="col-6 text-end">
    <a asp-action="Create" class="btn btn-primary"><i class="bi bi-plus-square"></i>  Přidat klienta</a>
</div>
```

Obrázek 24: Odkaz pro vytvoření nového klienta. Zdroj: autor

Na obrázku číslo 25 je vidět implementace hlavičky tabulky. Tabulku si vytvoříme pomocí tagu `<table>`, první řádek tabulky obsahující popis dat vytvoříme tagem `<thead>`. Tag `<tr>` vyznačuje řádek tabulky a `<th>` pak konkrétní buňku v hlavičce tabulky. Pomocí funkce `@Html.DisplayNameFor` vložíme jména jednotlivých sloupců do tabulky. Každé jméno odpovídá názvu vlastnosti třídy, které jsme specifikovali ve složce `Models`. V tomto případě je možné nalézt v souboru `Client.cs` nad vlastností `FirstName` vlastnost `[Display(Name = "Jméno")]`. V první buňce tabulky se tedy zobrazí název `Jméno` namísto `FirstName`. Poslední sloupec tabulky necháme prázdný, protože zde v další části budeme vkládat odkazy na akce související s klientem.

```
<thead>
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.FirstName)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.LastName)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Email)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.PhoneNumber)
        <th></th>
    </tr>
</thead>
```

Obrázek 25: Vytvoření hlavičky tabulky. Zdroj: autor

Na obrázku číslo 26 je zobrazeno tělo tabulky, do kterého se vkládají data klientů. Protože Model implementuje generické rozhraní `IEnumerable<T>`, můžeme k prohlížení prvků v něm obsažených použít cyklus `foreach`, který při každém projití cyklu načte konkrétní instanci třídy `Client` do proměnné `item.` Pomocí funkce `@Html.DisplayFor` poté zobrazíme jméno, příjmení, e-mail a telefonní číslo.

```
<tbody>
    @foreach (var item in Model)
    {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.FirstName)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.LastName)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Email)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.PhoneNumber)
            </td>
            <td style="min-width:200px;" class="text-center">
                <a asp-action="Details" asp-route-id="@item.Id" class="btn btn-primary mx-2">
                    <i class="bi bi-info-circle"></i></a>
                <a asp-action="Edit" asp-route-id="@item.Id" class="btn btn-primary mx-2">
                    <i class="bi bi-pen"></i></a>
                <a asp-action="Delete" asp-route-id="@item.Id" class="btn btn-danger mx-2">
                    <i class="bi bi-trash"></i></a>
            </td>
        </tr>
    }
</tbody>
```

Obrázek 26: Vytvoření těla tabulky. Zdroj: autor

V posledním sloupci vytvoříme tři odkazy pro zobrazení detailu klienta, úpravu informací a smazání záznamu klienta. Jméno metody, která se po kliknutí na odkaz vykoná, specifikujeme v `asp-action`. Design těchto odkazů nastavíme přiřazením tříd. Například třídy `btn` a `btn-primary` se postarají o to, že tyto odkazy budou vypadat jako tlačítka. Do odkazu jsme ještě vložili tag `<i>`, který se postará o vykreslení ikony podle třídy specifikované frameworkem Bootstrap.

Vytvoření nového klienta

Pro vytvoření nového klienta je v kontroleru `ClientController.cs` definována metoda `Create`, která má dvě implementace. První implementace metody `Create` zpracovává příchozí požadavek na zobrazení stránky. Žádná data se nepředávají a rovnou se zavolá metoda na vykreslení stránky.

O vykreslení stránky se postará pohled uložený v souboru `Views/Client/Create.cshtml`. Cílem tohoto pohledu je poskytnout uživateli prostor pro vyplnění všech důležitých informací o klientovi. Vytvoříme si tedy formulář, který má tag `<form>` a vlastnost `asp-action` s hodnotou `Create`. Do

tohoto formuláře přidáme všechny elementy. Aby byl web použitelný i na mobilních zařízeních, rozdělíme si obrazovku na dvě části. Použijeme dva tagy `<div>`, kterým přiřadíme třídy `col-12` `col-md-5`. Pokud bude mít uživatelské zařízení šířku větší než 720 pixelů, tak jeden `<div>` bude mít šířku 5/12 obrazovky, pokud má ale menší šířku, tak bude zabírat 12/12 obrazovky. Při sestavování formuláře tedy musíme počítat s tím, že pokud bude uživatel používat menší zařízení, tak se tyto dvě části přesunou pod sebe, jelikož na jednom řádku nemůže být součet těchto elementů vyšší než 12.

Do prvního elementu `<div>` vložíme textové pole s použitím tagu `<input>` pro vlastnosti `FirstName`, `LastName`, `Email`, `PhoneNumber`, `Gender`, `Age`, `Weight` a `Height`. V druhém elementu `<div>` budeme používat tag `<texarea>`, který se používá pro zápis větších textů. Zde budou vytvořeny prvky pro vlastnosti `LifeStyle`, `FavouriteFoods` a `Allergies`. Počet řádků v těchto vstupních polích nastavíme vlastností `rows`.

Každé textové pole na stránce bude mít vlastnosti `asp-for` a `asp-validation-for`. První vlastnost se postará o správné nastavení prvků. Například nastaví vlastnosti `id` a `name`. Druhá vlastnost `asp-for` má na starost validaci uživatelského vstupu. Pokud se uživatel pokusí odeslat hodnotu, která do daného pole nepatří (například text do prvku `Age`), případně zapomene vyplnit povinnou hodnotu tam, kde se vyžaduje, zobrazí se chybové hlášení a dokud nedojde k nápravě, nebude možné formulář odeslat.

Poslední dva prvky formuláře jsou tlačítka. První tlačítko se jmenuje **Zpět**. Je zde nutné specifikovat vlastnosti `asp-controller` a `asp-action`. Vlastnost `asp-controller` určuje, který kontroler má obdržet a zpracovat požadavek vzniklý kliknutím na tlačítko. Vlastnost `asp-action` pak říká, jaká metoda z daného kontroleru má být zavolána. Kliknutím na tlačítko **Zpět** se vrátíme na zobrazení všech klientů, takže kontroler je nastaven na `Client` a budeme volat metodu `Index`.

Kliknutím na tlačítko **Vytvořit klienta** dojde k potvrzení formuláře. Pro správné fungování je důležité nastavit vlastnost `type` na hodnotu `submit`. Není zde nutné nastavit kontroler a ani jeho metodu, protože jsme to již udělali při vytvoření tohoto formuláře. Jakmile uživatel vyplní všechny povinné hodnoty a odešle formulář, dojde v pohledu k vytvoření požadavku HTTP POST do kontroleru `ClientController.cs`.

Na obrázku číslo 27 je zobrazena druhá implementace metody `Create`, která zpracovává požadavek HTTP POST. Metoda má parametr typu `Client`. V těle metody nejdříve získáme identifikátor přihlášeného uživatele `Id`, který uložíme do instance `client` získané z odeslání formuláře. Dále ověříme, zda v této instanci nechybí žádná povinná vlastnost, případně zda mají všechny požadovaný typ. Pokud se objevila nějaká chyba, vykreslí se znovu formulář na vytvoření klienta. Je-li je vše v pořádku, přidáme instanci `client` do instance třídy `ApplicationDbContext` s názvem `_context` třídy `ApplicationDbContext` a metodou `SaveChangesAsync()` uložíme do databáze a vrátíme se na zobrazení všech klientů.

```
[HttpPost]
[ValidateAntiForgeryToken]
Počet odkazů: 0
public async Task<IActionResult> Create([Bind("Id,FirstName,LastName,Email,PhoneNumber,Gender," +
            "LifeStyle,Age,Weight,Height,FavouriteFoods,Allergies,DateOfCreation")] Client client)
{
    string userId = User.FindFirstValue(ClaimTypes.NameIdentifier);
    client.UserId = userId;
    if (ModelState.IsValid)
    {
        _context.Add(client);
        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }
    return View(client);
}
```

Obrázek 27: Vytvoření nového uživatele. Zdroj: autor

5.5 Recepty

Tato část popisuje implementaci funkcí pro vytvoření nové kategorie jídel a nového receptu. Důraz je zde kladen především na způsoby implementace, které v předchozích kapitolách ještě nebyly popsány.

Vytvoření nové kategorie jídel

V kontroleru `FoodCategoryController.cs` je vytvořena metoda `Create`. Tato metoda vykreslí pohled uložený ve `Views/FoodCategory/Create.cshtml`.

Tento pohled obsahuje formulář a v něm jediný vstupní prvek. Ten má tag `<input>` a odpovídá sloupci `CategoryName` tabulky `FoodCategories`. Pomocí vlastnosti `asp-validation-for` je zde ošetřena možnost, že by uživatel nezadal žádnou hodnotu. Dále se zde nacházejí dvě ovládací tlačítka. Po kliknutí na tlačítko **Zpět** se zavolá metoda `Index`. Po kliknutí na tlačítko **Vytvořit kategorii** se nejprve ověří, zda je jméno kategorie v pořádku a pak pohled pošle požadavek HTTP POST do kontroleru.

V kontroleru je pro zpracování implementována metoda `Create`, která v argumentu `foodCategory` obdrží instanci třídy `FoodCategory`. Do vlastnosti `UserId` této instance uložíme identifikátor přihlášeného uživatele. Poté proběhne ověření, že je vše v pořádku a dojde k uložení do databáze. Po vytvoření nové kategorie dojde k přesměrování na metodu `Index` a zobrazí se seznam všech kategorií vytvořených uživatelem.

Vytvoření nového receptu

Vytvoření receptu je možné v záložce **Recepty**, kde přihlášený uživatel klikne na tlačítko **Přidat recept**. Tím se vytvoří v pohledu HTTP GET požadavek na kontroler `FoodController.cs.` Metoda `Create` vyhledá všechny kategorie jídel uživatele a předá je k vykreslení do pohledu. Ten je uložený ve `Views/Food/Create.cshtml.`

V tomto pohledu je vytvořen formulář s použitím tagu `<form>` do kterého budou přidány všechny vstupní prvky. Důležité je u tohoto formuláře nastavit atribut `enctype` s hodnotou `multipart/form-data`, aby bylo možné jeho prostřednictvím odeslat obrázek jídla na server.

Aby se na stránce vše přehledně zobrazovalo, je třeba rozdělit obrazovku na tři sloupce, podobně jako v kapitole Klient u nadpisu Vytvoření nového klienta. První sloupec je určen pro nahrávání obrázku. Vytvoříme si tedy tagem `<img>` element zobrazující obrázek. Protože vytváříme nový recept, žádný obrázek ještě nemá recept přiřazen a je mu nastavená vlastnost `src` na cestu `/images/defaultImage.png`. Aby bylo možné s tímto obrázkem později manipulovat, přiřadíme mu ještě vlastnost `id` s hodnotou `foodImage`. Pod obrázek vložíme vstupní prvek s tagem `<input>`, nastavíme mu `id` na `imageInput` a vlastnost `accept` na hodnotu `image/*`. Tím bude ošetřeno, že uživatel bude moci nahrát jen obrázkové soubory.

Ve chvíli, kdy uživatel klikne na tlačítko **Vybrat soubor**, zobrazí se průzkumník souborů s výzvou k nahrání obrázku. Po vybrání obrázku se původnímu elementu s `id foodImage` nastaví cesta k novému souboru a zobrazí se tak nově nahraný obrázek. Druhý sloupec obsahuje ovládací prvky pro zadávání hodnot. Vytvoříme vstupní pole pro jméno, dále výběr kategorie pomocí tagu `<select>` a nastavíme seznam kategorií `asp-items` na hodnotu `ViewBag.FoodCategories.` Posledním elementem je textové pole pro poznámky o velikosti 7 řádků.

Ve třetím sloupci je nadpis Rozdělení živin a pod ním element `<div>`, který má `id` pojmenovaný `foodchart`. Do tohoto oddílu budeme vkládat koláčový graf. S využitím frameworku Bootstrap vytvoříme pod těmito třemi sloupci druhý blok, který opět rozdělíme na tři sloupce. Tentokrát však použijeme jen prostřední sloupec, do kterého dáme tři vstupní pole pro tuky (`Fats`), sacharidy (`Carbohydrates`) a bílkoviny (`Protein`).

Pod těmito třemi sloupci se nachází další oddíl `<div>`, který je znovu rozdělen na tři části. Je to z důvodu, aby uživatel měl při použití webové aplikace na mobilním zařízení srovnány všechny sloupce tak, aby dávaly smysl. Tento oddíl je zarovnán na střed a obsahuje tři vstupní pole pro vlastnosti `Fats`, `Carbohydrates` a `Proteins`. Každé z těchto polí má nastavené odpovídající `id` a událost `oninput,` která sleduje změnu hodnoty a poté zavolá funkci `drawChart()`.

Tato funkce je implementována v jazyce JavaScript. Kód musí být ohraničen tagem `<script>`. Při zavolání funkce dochází k načtení hodnot do proměnných `fatsValue`, `carbohydratesFalue` a `proteinsValue`. Před zobrazením grafu je nutné ošetřit, zda tyto proměnné obsahují platné hodnoty. Pokud uživatel zadal znak místo čísel, bude hodnota proměnné nastavena na 0. Pokud zadal hodnotu, která je menší než 0, tak se vykreslení neprovede.

K zobrazení koláčových grafů se využívá Google Charts. Na obrázku číslo 28 je vidět vytvoření instance třídy `DataTable`, do které byla nahrána tato trojice proměnných. Pak se nastaví vzhled grafu, například rozmístění popisků, zarovnání grafu nebo barva pozadí.

```
var data = google.visualization.arrayToDataTable([
    ['Macronutrients', '%'],
    ['Tuky', fatsValue],
    ['Sacharidy', carbohydratesFalue],
    ['Bílkoviny', proteinsValue],
]);
```

Obrázek 28: Vytvoření instance DataTable. Zdroj: autor

Na obrázku číslo 29 je zobrazena funkce, jejíž úkolem je vykreslit a zobrazit koláčový graf do oddílu, který má `id` `foodChart`.

```
var chart = new google.visualization.PieChart(document.getElementById('foodChart'));
    chart.draw(data, options);
}
```

Obrázek 29: Vykreslení koláčového grafu. Zdroj: autor

Kliknutím na tlačítko **Přidat recept** se odešle požadavek HTTP POST z pohledu zpět do kontroleru. Metoda `Create` obdrží v parametru `food` instanci třídy `Food`. Zkontrolujeme, že jsou všechny vlastnosti této instance v pořádku a poté přejdeme k nahrání obrázku. Pokud je v této instanci vlastnost `ImageFile` rovna hodnotě `null`, tak nastavíme cestu k výchozímu obrázku do složky `wwwroot/images`. Pokud obsahuje cestu k obrázku, tak si ji uložíme pomocí instance třídy `FileStream` do výše zmíněné složky. Na závěr přidáme tuto instanci do proměnné `_context` a uložíme změny do databáze. Uživateli se zobrazí seznam všech jeho receptů.

5.6 Jídelníčky

Implementace jídelníčků je rozdělena do dvou kontrolerů. Prvním je `MealPlan.cs`, ve kterém se vytváří samotné jídelníčky. V kontroleru `MealPlanItems.cs` se pak vytvářejí konkrétní položky daného jídelníčku.

Vytvoření nového jídelníčku

V kartě **Jídelníčky** je zobrazen přehled všech jídelníčků a také tlačítko **Přidat jídelníček**. Kliknutím na toto tlačítko se zobrazí jednoduchý pohled se vstupním polem pro název jídelníčku (`Name`) a pak výběr klienta, pro kterého je jídelníček určen (`ClientId`). Tato možnost se dá kdykoliv změnit při úpravě jídelníčku. Pro případ, že uživatel nechce tvořit jídelníček pro konkrétní osobu, může u výběru Jméno klienta zvolit možnost „**---**“ a jméno později změnit. Kliknutím na tlačítko **Vytvořit jídelníček** dojde k vytvoření požadavku HTTP POST a kontroler `MealPlan.cs` se postará o vytvoření tohoto jídelníčku.

Přidání jídla do jídelníčku

Po vytvoření jídelníčku je možné kliknout na tlačítko **Upravit**, čímž se dostaneme na detail konkrétního jídelníčku. Na levé straně je hlavní nadpis `<h1>` s názvem jídelníčku a pod ním se nachází jméno klienta `<h2>`. Vedle nadpisu jsou dvě tlačítka. Modrým tlačítkem změníme název jídelníčku nebo klienta. Červeným tlačítkem odstraníme celý jídelníček.

Na pravé straně je tlačítko **Přidat jídlo do dalšího dne**. Toto tlačítko má nastavenou vlastnost `asp-action` na hodnotu `Create`, kliknutím se z pohledu vytvoří požadavek HTTP GET na kontroler `MealPlanItemController.cs`. Metoda `Create` vytvoří pohled, do kterého přidáme dvě vstupní pole pro vlastnosti `MealPlanItemNote` a `Quantity`. Dále vytvoříme element výběr pro vlastnost `FoodId`.

Pod těmito prvky se nacházejí dvě tlačítka. Tlačítkem **Zpět** se vrátíme na předchozí pohled. Tlačítko **Přidat do jídelníčku** odešle data s požadavkem HTTP POST do kontroleru. Ten se zpracuje v metodě `Create`, která obdrží v parametru vytvořené jídlo do jídelníčku a den, do kterého toto jídlo patří. Nově vytvořené jídlo se vloží do proměnné `_context.MealPlanItems`. Protože přidáním jídla do jídelníčku došlo ke změně jídelníčku, upravíme čas poslední úpravy na aktuální hodnotu pomocí metody `DateTime.Now`. Poté uložíme změny do databáze a vykreslí se pohled upraveného jídelníčku.

Export jídelníčku do PDF

V liště Jídelníčky si uživatel nejprve vybere jídelníček, který chce exportovat a pak klikne na tlačítko **Stáhnout**. Pohled vytvoří požadavek HTTP GET na kontroler `MealPlanController.cs`. O vygenerování souboru se postará metoda `GeneratePDF`, která má vstupní parametr `id` jídelníčku. Nejdříve se získají jídla vytvořená uživatelem. Poté dojde k načtení všech záznamů jídelníčku (`MealPlanItem`). Protože může jídelníček obsahovat jídla rozdělená do více dnů, seřadíme si tyto data sestupně dle čísla dne (`NumberOfDay`).

Vytvoříme si instanci třídy `Document`, u níž nastavíme velikost strany A4 a odsazení na všech stranách na 25. Při vytváření dokumentu budeme používat písmo Arial, načteme si ho tedy do proměnné `fontArial` typu `BaseFont`. Nadpis jídelníčku načteme z proměnné `mealPlan.Name`, zarovnáme na střed a nastavíme velikost písma na 25 bodů. Pod nadpisem se zobrazí jméno klienta písmem o velikosti 15 bodů.

Do další části dokumentu budeme vkládat jednotlivé dny jídelníčku. Každý den začíná nadpisem obsahujícím číslo daného dne. Pomocí cyklu `foreach` projdeme všechny jídla a zapíšeme je do dokumentu, který je rozdělen na 4 sloupce. V prvním sloupci se nachází poznámka (například snídaně), ve druhém sloupci se ke každému jídlu načte obrázek. Na obrázku číslo 30 je vidět změna rozlišení tak, aby měl obrázek vždy šířku 60 pixelů. Například, pokud byla původní šířka obrázku 4600 pixelů a výška 3456 pixelů, tak se šířka rovná 60 pixelů a výška se vynásobí 60/4600 pixelů. Nová šířka se tak bude rovnat přibližně 45 pixelů.

```
float imageWidth = 60;
float imageHeight = (60 / mealImage.Width) * mealImage.Height;
mealImage.ScaleAbsolute(imageWidth, imageHeight);
```

Obrázek 30: Úprava rozlišení obrázku. Zdroj: autor

Ve třetím sloupci je množství jídla uvedeného v gramech. V posledním sloupci se nachází jméno jídla. Na dalším řádku je zobrazena poznámka k danému jídlu. Může to být recept, případně doporučení.

Jakmile cyklus projde všechna jídla jídelníčku, dojde k uložení souboru. Soubor bude pojmenován podle názvu jídelníčku, který je uložen v tabulce `MealPlan` a sloupci `Name`. Protože se jedná o dokument typu PDF, nesmíme zapomenout na příponu souboru `pdf`. Kontroler poté pošle soubor uživateli, který se mu automaticky stáhne do počítače. Ukázkový jídelníček lze vidět na obrázku číslo 31.

Obrázek 31: Ukázkový jídelníček PDF. Zdroj: autor

5.7 Profil

Každý uživatel má ke svému účtu přidružený profil, který není veřejný. Po kliknutí na záložku **Veřejný profil** se zobrazí pohled, který je umístěný ve `Views/Profile/Index.cshtml`. O zobrazení tohoto pohledu se postará kontroler `ProfileController.cs`.

Pokud uživatel chce, aby se jeho profil zobrazoval všem návštěvníkům na hlavní straně v záložce AjaFood, musí kliknout na záložku **Veřejný profil** a vyplnit své uživatelské jméno (`Username`) a popis profilu (`Introduction`). Vložení fotografie je volitelné. Kliknutím na tlačítko **Uložit změnu** dojde k vytvoření požadavku HTTP POST, který zpracuje metoda Edit.

V parametru metoda obdrží instanci třídy `Profile`. Pokud uživatel nahrál nový profilový obrázek, nahrajeme ho do složky `wwwroot/ProfileImages` a původní obrázek smažeme. Poté uložíme změny v profilu do databáze a necháme pohled, aby vykreslil úvodní stranu Veřejný profil. Pokud uživatel vyplnil všechny požadované informace, na hlavní straně AjaFood již bude profil viditelný.

5.8 Publikování webové stránky

K publikování této webové stránky jsme si vybrali hosting na webové adrese https://www.smarterasp.net/. Po registraci přišel do e-mailu odkaz na ověření účtu. Po ověření a přihlášení do aplikace jsme využili nabídku 60 dnů hostování zdarma.

Nejprve si určíme jméno webové stránky. Protože se aplikace jmenuje AjaFood, vložíme tento název a následně se vytvoří bezplatná adresa ajafood-001-site1.ctempurl.com. Dále klikneme na tlačítko **DATABASES**, kde zvolíme možnost **MSSQL** a vybereme databázový typ **MSSQL 2019**. Nastavíme jméno databáze `AFDB` a heslo. Kliknutím na tlačítko **Submit** vytvoříme databázi.

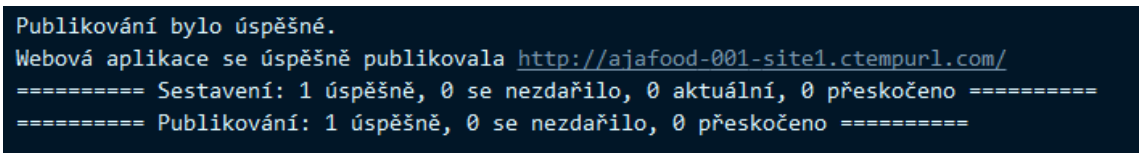
Klikneme na tlačítko **Connection String Examples** a pak zkopírujeme textový řetězec uvedený u nadpisu `ASP.NET`. Poté se přesuneme do vývojového prostředí a otevřeme soubor `appsettings.json`, do kterého vložíme zkopírovaný připojovací řetězec `DefaultConnection` a vyplníme nastavené heslo.

V tuto chvíli již můžeme sestavit databázi na serveru. Klikneme na záložku **Nástroje**, pak na **Správce balíčků NuGet** a nakonec na **Konzola Správce Balíčků**. Zde napíšeme příkaz `update-database`. Pokud vše proběhlo v pořádku, tak se na hostingu vytvořila nová databáze.

Pro publikování webové stránky na hosting nabízí Visual Studio možnost VS Webdeploy. Vrátíme se zpět na stránky hostingu do záložky WEBSITES. Klikneme

na možnost **Manage Website** a zvolíme **VS Webdeploy**. Objeví se konfigurační okno, to potvrdíme kliknutím na **Get Publish Setting**. Po stažení souboru se přesuneme do vývojového prostředí. V Průzkumníku řešení klikneme pravým tlačítkem na náš projekt a zvolíme volbu **Publikovat**. Na levé straně poblíž záložky Publikovat je tlačítko **Nový**. Po kliknutí se zobrazí okno s nadpisem Publikovat. Zvolíme **Cíl > Importovat profil**. Zobrazí se nová záložka Importovat profil. Klikneme na tlačítko **Procházet** a vybereme soubor s příponou `PublishSettings`, který se nám stáhl z hostingu. Klikneme na tlačítko **Dokončit** a poté na **Publikovat**. Zobrazí se formulář, ve kterém vyplníme přístupové heslo k webhostingu, potvrdíme a poté počkáme, než se dokončí publikování webové stránky.

Pokud nenastala žádná chyba, měli bychom vidět v konzoli zprávu o úspěšném publikování, jak je zobrazeno na obrázku číslo 32. Tato webová aplikace je přístupná z URL adresy http://ajafood-001-site1.ctempurl.com/.



Obrázek 32: Potvrzení o publikování webu. Zdroj: autor

6 Shrnutí výsledků

Výstupem této bakalářské práce je webová aplikace vyvinutá v jazyce ASP.NET Core. Při návrhu aplikace byly využity poznatky získané z teoretické části. V kapitole Existující řešení byly popsány stěžejní výhody a nedostatky jednotlivých aplikací. Webová aplikace má ve srovnání s popsanými systémy výhodu, že nabízí možnost zviditelnit se na hlavní stránce. Po vyplnění profilových informací se zde uživatel automaticky zobrazí a prostřednictvím uvedeného kontaktu jej pak může oslovit budoucí klient.

Další výhodou je, že aplikace je uživatelsky přívětivá, po registraci může uživatel během několika minut začít pracovat na jídelníčku. Uživatel nemusí řešit otázku zálohování svých souborů, protože má vše online. Nevýhodou vytvořené aplikace je jediná volba výstupu jídelníčku, a to generování do formátu PDF. Do projektu by bylo vhodné zapracovat i jiné způsoby odeslání. Například zaslání na e-mailovou adresu klienta, případně vytvoření odkazu na jídelníček, který by byl dostupný i bez nutnosti přihlášení.

V závěrečné fázi implementace všech funkcionalit mě napadlo pár možných rozšíření aplikace, které ale z důvodu časové náročnosti práce nebylo možné realizovat. Prostor pro zlepšení vidím třeba v možnostech exportu jídelníčků. V aplikaci není možné upravovat vzhled exportovaných jídelníčků. Vytvoření náhledu, ve kterém by si uživatel nastavil například velikost a odsazení písma, velikost obrázku, barvu pozadí či vlastní podpis, by bylo pro aplikaci velkým přínosem.

Dalším možným vylepšením aplikace by bylo přidání nástrojů na změnu rozlišení nebo oříznutí obrázku. Omezením velikosti obrázkového souboru by došlo ke snížení objemu dat přenášených po síti, což by se projevilo zrychlením odezvy aplikace.

7 Závěr

Cílem bakalářské práce bylo vyvinout webovou aplikaci pro výživové poradce. Pro naplnění tohoto cíle bylo zapotřebí provést rešerši a kriticky porovnat klady i zápory jednotlivých webových aplikací. Existující řešení jsou popsána ve druhé kapitole, přičemž jsem zde uvedl některé zástupce české i zahraniční.

Díky tomu jsem získal podstatné informace k tvorbě zásadních prvků aplikace. Snažil jsem se o vytvoření takové aplikace, která by zohlednila výhody existujících řešení a zároveň eliminovala jejich nedostatky. Samozřejmě je nutné brát v úvahu, že tento cíl je poněkud náročný a jeho splnění by vyžadovalo práci týmu lidí po dobu několika let. V bakalářské práci jsem se proto zaměřil na vytvoření takového konceptu, který by byl funkční a použitelný v praxi. Výsledkem je webová aplikace, která umožňuje spravovat databázi jednotlivých klientů, vytvářet profily jídel a z nich následně tvořit jídelníčky. Poradci mají také možnost jimi vytvořené jídelníčky snadno exportovat do souborů ve formátu PDF – aplikace je uživatelsky přívětivá.

Zpracování bakalářské práce na toto téma mě profesně velmi obohatilo. Zkoumáním dané problematiky jsem získal jasnější představy o návrhu a vývoji webových aplikací, včetně potřebné časové dotace. Uvědomil jsem si, v čem jsou obdobné projekty náročné. Sledování současných trendů a aktualizací vývojových nástrojů, v tomto případě programovacího jazyku ASP.NET Core, je pro programátora nutností. Dospěl jsem k závěru, že mě vývoj webových aplikací baví a naplňuje navzdory překážkám, které musí člověk při jejich tvorbě překonat.

8 Seznam použité literatury

[1] ROUBÍK, Lukáš. Moderní výživa ve fitness a silových sportech. Praha: Erasport, [2018]. ISBN 978-80-905685-5-6.

[2] All-in-One Software for Nutritionists and Dietitians In: Youtube [online]. Dostupné z: https://youtu.be/maY0HyKF04o?t=210

[3] NutriAdmin pricing [online]. 2016 [cit. 07.01.2022]. Dostupné z: https://nutriadmin.com/pricing

[4] Ceník. Fitlinie - nutriční software [online]. 2022 [cit. 07.01.2022]. Dostupné z: https://www.fitlinie.cz/cenik

[5] NutriSystem - online aplikace pro fitness trenéry a výživové poradce - 30 dní zdarma. [online]. 2022 [cit. 07.01.2022]. Dostupné z: https://www.nutris.cz

[6] Ceník - NutriSystem - online aplikace pro fitness trenéry. [online] 2022 [cit. 07.01.2022]. Dostupné z: https://www.nutris.cz/cenik/

[7] Frontend vs Backend - GeeksforGeeks [online]. 2022 [cit. 17.03.2022]. Dostupné z: https://www.geeksforgeeks.org/frontend-vs-backend/

[8] HTML basics - Learn web development [online]. 2022 [cit. 17.03.2022]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics

[9] JANOVSKÝ, Dušan. Struktura stránky v jazyce HTML [online]. 2022 [cit. 17.03.2022]. Dostupné z: https://www.jakpsatweb.cz/html/struktura.html

[10] Desktop vs Mobile vs Tablet Market Share Worldwide [online]. 2022 [cit. 08.01.2022]. Dostupné z: https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/worldwide/#monthly-202202-202202-bar

[11] Browsers and devices Bootstrap [online]. 2022 [cit. 17.03.2022]. Dostupné z: https://getbootstrap.com/docs/4.0/getting-started/browsers-devices/

[12] Úvod do ASP.NET Core [online]. 2022 [cit. 08.01.2022]. Dostupné z: https://docs.microsoft.com/cs-cz/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-6.0

[13] Choose between ASP.NET 4.x and ASP.NET Core [online]. 2022 [cit. 08.01.2022]. Dostupné z: https://docs.microsoft.com/en-us/aspnet/core/fundamentals/choose-aspnet-framework?view=aspnetcore-6.0

[14] Download .NET 6.0 (Linux, macOS, and Windows) [online]. 2022 [cit. 08.01.2022]. Dostupné z: https://dotnet.microsoft.com/en-us/download/dotnet/6.0

[15] Overview of ASP.NET Core MVC [online]. 2022 [cit. 08.01.2022]. Dostupné z: https://docs.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-6.0

[16] Overview of Entity Framework Core [online]. 2022 [cit. 08.01.2022]. Dostupné z: https://docs.microsoft.com/en-us/ef/core/

[17] Database Providers - EF Core [online]. 2022 [cit. 08.01.2022]. Dostupné z: https://docs.microsoft.com/en-us/ef/core/providers/?tabs=dotnet-core-cli

[18] Databases - SQL Server [online]. 2022 [cit. 06.06.2022]. Dostupné z: https://docs.microsoft.com/en-us/sql/relational-databases/databases/databases?view=sql-server-ver15

[19] aspnetcore/PasswordHasher.cs [online]. 2022 [cit. 01.07.2022]. Dostupné z: https://github.com/dotnet/aspnetcore/blob/main/src/Identity/Extensions.Core/src/PasswordHasher.cs

[20] int, bigint, smallint, and tinyint (Transact-SQL) - SQL Server [online]. 2022 [cit. 01.07.2022]. Dostupné z: https://docs.microsoft.com/en-us/sql/t-sql/data-types/int-bigint-smallint-and-tinyint-transact-sql?view=sql-server-ver15

[21] Jak spočítat příjem energie a makroživin pro hubnutí, nebo nabírání svalů? [online]. 2014 [cit. 09.08.2022]. Dostupné z: https://gymbeam.cz/blog/jak-spocitat-prijem-energie-a-makrozivin-pro-hubnuti-nebo-nabirani-svalu/

[22] NuGet Gallery [online] 2022 [cit. 01.07.2022] Dostupné z: https://www.nuget.org/

[23] Migrations Overview - EF Core [online]. 2022 [cit. 06.07.2022]. Dostupné z: https://docs.microsoft.com/en-us/ef/core/managing-schemas/migrations/?tabs=dotnet-core-cli

[24] IMAP, POP3, SMTP [online]. 2022 [cit. 06.08.2022]. Dostupné z: https://napoveda.seznam.cz/cz/email/imap-pop3-smtp/

9 Přílohy

1) Součástí bakalářské práce je i soubor obsahující zdrojový kód webové aplikace, který je odevzdán v souboru *ZdrojovyKod.zip*. Aktuální verzi webové aplikace je možné stáhnout z adresy: https://github.com/IvoBauer/AjaFoodBP
2) Podklad po zadání BAKALÁŘSKÉ práce studenta

UNIVERZITA HRADEC KRÁLOVÉ
Fakulta informatiky a managementu
Akademický rok: 2021/2022

Studijní program: Aplikovaná informatika
Forma studia: Prezenční
Obor/kombinace: Aplikovaná informatika (ai3-p)

Podklad pro zadání BAKALÁŘSKÉ práce studenta

Jméno a příjmení:	**Ivo Bauer**
Osobní číslo:	**I1800153**
Adresa:	Pomněnková 436, Trutnov – Horní Staré Město, 54102 Trutnov 4, Česká republika
Téma práce:	Webová aplikace pro výživové poradce
Téma práce anglicky:	Web Application for Nutritional Advisors
Vedoucí práce:	Ing. Karel Malý, Ph.D. Katedra informatiky a kvantitativních metod

Zásady pro vypracování:

Cílem bakalářské práce je analýza požadavků na systém, srovnání existujících řešení a implementace webové aplikace pro výživové poradce a jejich klienty.
Osnova:
1. Úvod
2. Existující řešení
3. Frontendové a backendové technologie
4. Návrh aplikace
5. Implementace aplikace
6. Závěr

Seznam doporučené literatury:

Podpis studenta: Bauer Datum: 3. 8. 2022

Podpis vedoucího práce: Datum: 29. 6. 2022