Gymnázium, Praha 6, Arabská 14

Předmět Programování

ROČNÍKOVÝ PROJEKT

Webová aplikace – Zdravá výživa

Daniela Pilková – IV.E 2023/24

Prohlašuji, že jsem jediným autorem tohoto projektu, všechny citace jsou řádně označené a všechna použitá literatura a další zdroje jsou v práci uvedené. Tímto dle zákona 121/2000 Sb. (tzv. Autorský zákon) ve znění pozdějších předpisů uděluji bezúplatně škole Gymnázium, Praha 6, Arabská 14 oprávnění k výkonu práva na rozmnožování díla (§ 13) a práva na sdělování díla veřejnosti (§ 18) na dobu časově neomezenou a bez omezení územního rozsahu.

V …… dne …… Daniela Pilková ……..……

Obsah

1. Abstract

Tento ročníkový projekt se zaměřuje na tvorbu webové stránky, která se zaměřuje na zdravé stravování. Uživatelům umožňuje vytvářet zdravé recepty a sdílet je s ostatními uživateli. Aplikace také umožňuje uživateli vytvořit měsíční plán, kde může spravovat svůj jídelníček na každý den v měsíci. Může si vybrat, kolikrát chce za den jíst a na základě počtu kalorií se mu vygenerují recepty, které tento atribut splňují. Aplikace také obsahuje užitečné tipy, jak si udržet zdravý životní styl.

2. Zadání

Téma: Webová aplikace – zdravá výživa

Autor: Daniela Pilková

Popis:

Cílem mého projektu je vytvořit webovou aplikaci, která bude sloužit jako receptář. Člověk si vybere, kolik kilokalorií chce zkonzumovat, kolikrát denně chce jíst. Na základě toho se mu vygeneruje denní jídelníček i s recepty. Jestli se mu jídlo nepozdává, může ho vyměnit za jiné. Aplikace bude obsahovat také tipy a rady, jak se zdravě stravovat a případně jak správně cvičit.

Platforma & Technologie: Visual Studio Code, Node.js, Django, Python, CSS, HTML, Cloudinary, Bootstrap

3. Technologie a architektura

3.1. Django

Django je vysokoúrovňový webový framework napsaný v jazyce Python, který podporuje rychlý vývoj a čistý, pragmatický design. Následuje filozofii "vše v jednom balení" (batteries-included), což znamená, že se snaží zahrnout vše potřebné pro vývoj webových aplikací přímo do svých základů. Django je open-source a zdarma k použití.

Klíčové vlastnosti Djanga:

1. Architektura MVC: Django následuje architektonický vzor Model-View-Controller (MVC), ačkoli ho nazývá Model-View-Template (MVT). Toto pomáhá udržovat čistý a znovupoužitelný kód.
2. ORM (Object-Relational Mapping): Django poskytuje vrstvu ORM, která umožňuje vývojářům komunikovat s databází pomocí objektů v Pythonu, namísto přímých SQL dotazů. To zjednodušuje operace s databází a činí kód přenositelnějším mezi různými databázovými systémy.
3. Administrační rozhraní: Django automaticky generuje upravitelné administrační rozhraní pro správu dat aplikace. To ušetřuje vývojářům čas při vytváření panelu správy pro operace Create, Read, Update, Delete (CRUD).
4. Routing URL: Django používá dispečer URL k přesměrování příchozích webových požadavků na příslušný pohled na základě URL vzorů definovaných v URL konfiguraci projektu.
5. Autentizace a autorizace: Django poskytuje robustní mechanismy autentizace a autorizace, včetně autentizace uživatelů, oprávnění a skupin.

Celkově je Django široce používán vývojáři k vytváření různých webových aplikací, od jednoduchých webových stránek po složité systémy na úrovni podniku. Jeho rozsáhlá dokumentace, aktivní komunita a ekosystém třetích stranových balíčků ho dělají populární volbou pro webové vývojové projekty.

3.1.1. Django-allauth

Django-allauth je populární balíček pro Django, který poskytuje komplexní a snadno použitelné řešení pro autentizaci a správu uživatelských účtů ve webových aplikacích. Tento

balíček kombinuje různé funkce, jako je autentizace pomocí sociálních médií, registrace uživatelů, správa hesel, správa e-mailových adres a mnoho dalšího.

Mezi klíčové vlastnosti django-allauth patří:

1. Podpora pro různé způsoby autentizace: django-allauth podporuje autentizaci pomocí různých mechanismů, včetně klasického uživatelského jména a hesla, přihlášení pomocí sociálních médií (např. Facebook, Twitter, Google, GitHub atd.) a autentizace pomocí e-mailových ověření.
2. Flexibilní konfigurace: Balíček poskytuje možnost konfigurace a přizpůsobení autentizačních a registrace formulářů, šablon a chování podle potřeb konkrétní aplikace.
3. Jednoduchá integrace: Django-allauth lze snadno integrovat do existujících aplikací nebo nových projektů. Poskytuje jasně definované rozhraní API a podporuje běžné funkcionality, jako jsou odesílání e-mailů s ověřovacími odkazy a správa uživatelských relací.
4. Bezpečnost: Balíček se stará o bezpečnostní aspekty autentizace a správy uživatelských účtů, včetně správného ukládání a ověřování hesel, omezení počtu neúspěšných pokusů o přihlášení a dalších opatření k ochraně proti útokům.
5. Dokumentace a komunita: Django-allauth má dobře zdokumentovanou sadu funkcí a nabízí aktivní komunitu uživatelů a vývojářů, kteří poskytují podporu a přispívají k rozvoji balíčku.

Django-allauth je oblíbeným řešením pro autentizaci a správu uživatelských účtů v aplikacích postavených na Djangu. Jeho široká funkcionalita a snadná použitelnost usnadňují vývojářům vytváření bezpečných a uživatelsky přívětivých webových aplikací.

3.1.2. Django-cloudinary-storage

Django-Cloudinary-Storage je knihovna pro Django, která umožňuje snadnou integraci služby Cloudinary pro správu médií (obrázků, videí atd.) ve webových aplikacích postavených na Djangu. Cloudinary je cloudová služba specializovaná na správu a optimalizaci multimediálního obsahu, poskytující širokou škálu funkcí pro manipulaci s obrázky a videi, včetně ukládání, transformace, optimalizace a doručování médií.

Hlavní vlastnosti Django-Cloudinary-Storage:

1. Snadná integrace: Knihovna poskytuje jednoduchou a přímou integraci Cloudinary do projektů Django. Stačí nastavit konfiguraci knihovny a připojit ji k Cloudinary účtu.

2. Ukládání a správa médií: Django-Cloudinary-Storage umožňuje jednoduché ukládání médií do Cloudinary, a to včetně obrázků, videí a dalších multimediálních souborů. Díky tomu se uživatelé nemusí starat o složité aspekty správy a údržby souborů.
3. Automatická optimalizace: Cloudinary poskytuje funkce automatické optimalizace obrázků a videí pro rychlejší načítání a nižší datovou náročnost. Django-Cloudinary-Storage umožňuje využívat tyto funkce přímo ve webových aplikacích postavených na Djangu.
4. Pokročilé funkce manipulace s obrázky: Cloudinary nabízí širokou škálu pokročilých funkcí pro manipulaci s obrázky, jako je změna velikosti, oříznutí, filtry, efekty a mnoho dalšího. Tyto funkce jsou snadno dostupné a použitelné přímo z webových aplikací Django pomocí Django-Cloudinary-Storage.
5. Škálovatelnost a výkon: Cloudinary je navržen tak, aby byl škálovatelný a výkonný, což znamená, že může snadno zvládat zpracování velkého množství multimediálního obsahu bez ztráty výkonu. Django-Cloudinary-Storage využívá tuto škálovatelnost a umožňuje vývojářům stavět webové aplikace s vysokým výkonem.

Celkově lze Django-Cloudinary-Storage považovat za užitečný nástroj pro vývojáře Django, kteří chtějí využívat výhody Cloudinary pro správu a optimalizaci multimediálního obsahu ve svých webových aplikacích. Jeho snadná integrace, široká funkcionalita a škálovatelnost přispívají k rychlému a efektivnímu vývoji webových aplikací s bohatým multimediálním obsahem.

3.1.3. Django-crispy-forms

Django-crispy-forms je balíček pro Django, který usnadňuje vytváření formulářů v HTML s pomocí Django formulářů. Tento balíček přidává vrstvu nad běžné Django formuláře a poskytuje jednoduchý způsob, jak definovat a stylizovat formuláře pomocí Bootstrap nebo jiných CSS frameworků.

Hlavní vlastnosti Django-crispy-forms zahrnují:

1. Snadná integrace s Django formuláři: Django-crispy-forms poskytuje jednoduchý způsob, jak integrovat se stávajícími formuláři v aplikaci Django. Stačí přidat crispy tagy do šablon HTML a definovat stylizaci formulářů.
2. Podpora pro různé CSS frameworky: Balíček podporuje nejen Bootstrap, ale i další CSS frameworky, jako je Foundation nebo Materialize. To umožňuje vývojářům snadno přizpůsobit vzhled svých formulářů podle preferencí a požadavků projektu.

3. Jednoduché definování stylů: Django-crispy-forms umožňuje definovat stylizaci formulářů pomocí jednoduchých konfiguračních souborů nebo přímo v kódu šablon. To umožňuje vývojářům snadno spravovat vzhled formulářů a zjednodušuje údržbu kódu.
4. Dokumentace a komunita: Django-crispy-forms má dobře zdokumentovanou sadu funkcí a nabízí aktivní komunitu uživatelů a vývojářů, kteří poskytují podporu a přispívají k rozvoji balíčku. Django-crispy-forms je užitečným nástrojem pro vývojáře

Django, kteří chtějí snadno vytvářet a stylizovat formuláře v HTML s pomocí CSS frameworků. Jeho snadná integrace, podpora pro různé frameworky a jednoduché definování stylů přispívají k efektivnímu vývoji webových aplikací s atraktivními a uživatelsky přívětivými formuláři.

3.1.4. Django-richtextfield

Django-richtextfield je balíček pro Django, který umožňuje snadnou integraci bohatého textového editoru do webových aplikací postavených na Djangu. Tento balíček poskytuje pole formuláře, které umožňuje uživatelům vkládat a upravovat formátovaný text pomocí bohatých textových funkcí, jako jsou různé styly písma, odrážky, obrázky, odkazy atd.

Hlavní vlastnosti Django-richtextfield:

1. Jednoduchá integrace: Django-richtextfield je snadno integrovatelný do formulářů v aplikacích Django. Stačí definovat pole typu richtextfield ve formulářové třídě a přidat ho do šablony.
2. Bohaté funkce textového editoru: Balíček poskytuje bohaté funkce textového editoru, které umožňují uživatelům formátovat text pomocí různých stylů písma, odrážek, číslování, vkládat obrázky, vytvářet odkazy a další.
3. Podpora pro různé textové editory: Django-richtextfield podporuje různé textové editory, včetně populárních editorů jako je TinyMCE, CKEditor, Summernote atd. Uživatelé mohou vybrat preferovaný editor podle svých potřeb a preferencí.
4. Flexibilní konfigurace: Balíček poskytuje možnost konfigurace vlastností textového editoru, jako jsou povolené funkce, výchozí nastavení, možnosti rozšíření atd.
5. Dokumentace a komunita: Django-richtextfield má dobře zdokumentované funkce a nabízí podporu od komunity uživatelů a vývojářů, což usnadňuje používání balíčku a řešení případných problémů.

Django-richtextfield je užitečným nástrojem pro vývojáře Django, kteří potřebují snadno integrovat bohatý textový editor do svých webových aplikací. Jeho jednoduchá integrace, bohaté funkce editoru a flexibilní konfigurace přispívají k vytváření atraktivních a uživatelsky přívětivých formulářů s formátovaným textem.

3.1.5. Generic Editing Views

V kontextu webových aplikací postavených na frameworku Django jsou "Generic Editing Views" (Obecné zobrazení úprav) součástí balíčku django.views.generic.edit, který poskytuje předdefinované pohledy pro operace úprav dat v databázi, jako jsou vytváření, aktualizace a mazání záznamů. Tyto obecné pohledy usnadňují vývojářům vytváření funkcí pro úpravu dat bez potřeby psaní značného množství vlastního kódu.

Některé z nejčastěji používaných generických pohledů:

1. CreateView (Pohled pro vytvoření): Poskytuje formulář pro vytvoření nového záznamu v databázi. Tento pohled je vhodný pro situace, kdy uživatelé potřebují vytvářet nové záznamy, jako například přidání nového článku, příspěvku nebo produktu do systému.
2. ListView (Pohled seznamu): Zobrazuje seznam všech dostupných záznamů v databázi. Tento pohled je užitečný pro prezentaci dat ve strukturované podobě, například seznamu článků, uživatelů nebo produktů.
3. DetailView (Detailní pohled): Poskytuje detailní informace o konkrétním záznamu v databázi. Tento pohled je vhodný pro zobrazení podrobností o jednom záznamu, například detaily článku, uživatele nebo produktu.
4. DeleteView (Pohled pro smazání): Poskytuje možnost odstranění konkrétního záznamu z databáze. Tento pohled umožňuje uživatelům odstranit záznam, který již není potřebný, například odstranění nežádoucího článku, uživatele nebo produktu.
5. UpdateView (Pohled pro aktualizaci): Poskytuje formulář pro úpravu existujícího záznamu v databázi. Tento pohled je užitečný pro situace, kdy uživatelé potřebují upravit již existující záznam, například aktualizovat informace o článku, uživateli nebo produktu.

Tyto generické pohledy jsou základními stavebními bloky webových aplikací postavených na frameworku Django a usnadňují vývojářům implementaci CRUD operací (Create, Read, Update, Delete) s daty. Díky nim mohou vývojáři rychle vytvářet robustní a efektivní webové aplikace s minimálním opakováním kódu.

3.2. Cloudinary

Cloudinary je cloudová služba pro správu digitálního obsahu, která poskytuje kompletní řešení pro ukládání, optimalizaci, manipulaci a doručování obrázků a videí na webových stránkách a mobilních aplikacích. Tato platforma umožňuje vývojářům snadno nahrávat, ukládat a spravovat multimediální soubory v cloudu, což snižuje zátěž na serverech a zlepšuje výkon a rychlost načítání webových stránek.

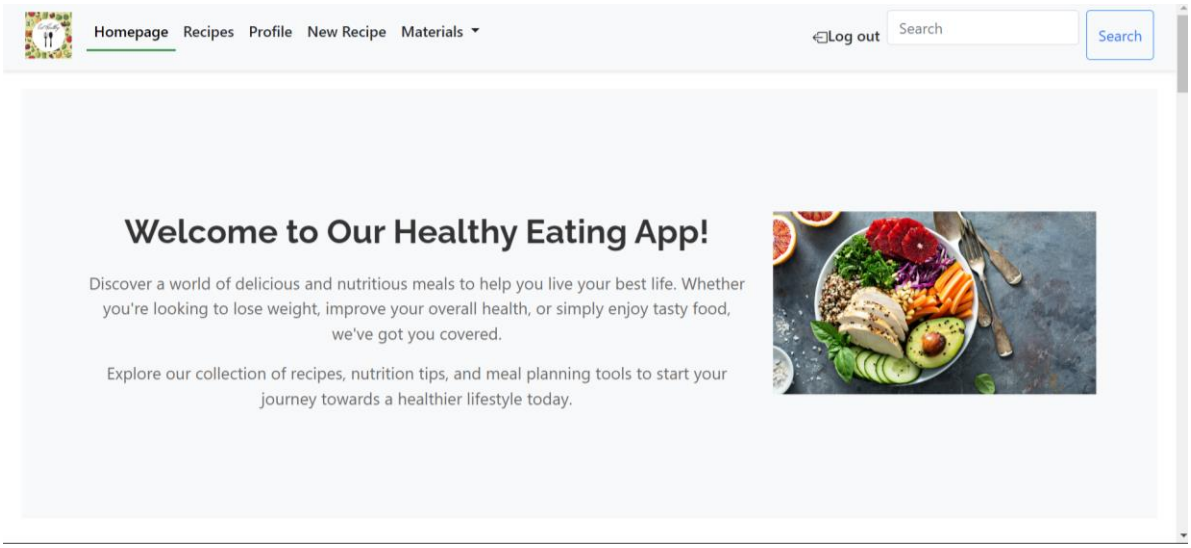
Některé klíčové vlastnosti a funkce Cloudinary:

1. Ukládání souborů v cloudu: Cloudinary umožňuje vývojářům nahrávat obrázky a videa do cloudového úložiště, což snižuje potřebu vlastního serverového úložiště a zjednodušuje správu digitálního obsahu.
2. Optimalizace obrázků a videí: Služba automaticky optimalizuje nahrávané obrázky a videa, aby minimalizovala jejich velikost a zlepšila rychlost načítání webových stránek.
3. Manipulace s obrázky: Cloudinary poskytuje širokou škálu funkcí pro manipulaci s obrázky, včetně změny velikosti, oříznutí, rotace, filtry, efekty a další.
4. Doručování obsahu: Služba umožňuje doručovat multimediální obsah pomocí CDN (Content Delivery Network), což zajišťuje rychlé načítání obrázků a videí na stránkách po celém světě.
5. Podpora pro různé platformy a jazyky: Cloudinary poskytuje rozhraní API a knihovny pro různé platformy a programovací jazyky, včetně JavaScriptu, Pythonu, Ruby, PHP a dalších.
6. Bezpečnostní funkce: Služba poskytuje možnosti ochrany digitálního obsahu, včetně možnosti vodoznaku, ochrany přístupu a omezení velikosti nahrávaných souborů.

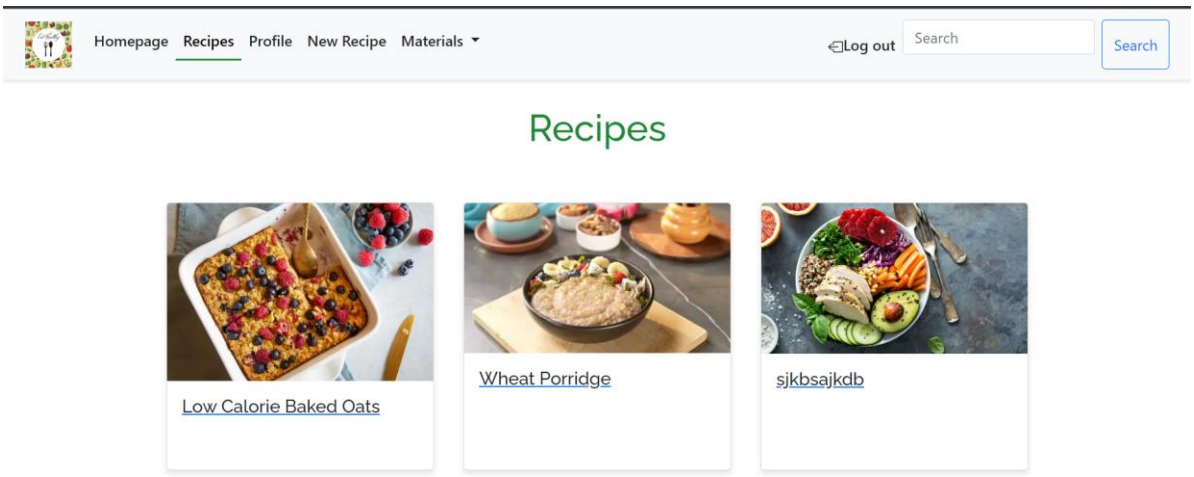
Cloudinary je široce využívanou službou v oblasti webového vývoje a poskytuje kompletní a efektivní řešení pro správu digitálního obsahu na internetu. Díky svým pokročilým funkcím a jednoduchému rozhraní je oblíbenou volbou mezi vývojáři, kteří potřebují rychle a efektivně spravovat multimediální soubory ve svých projektech.

4. O aplikaci

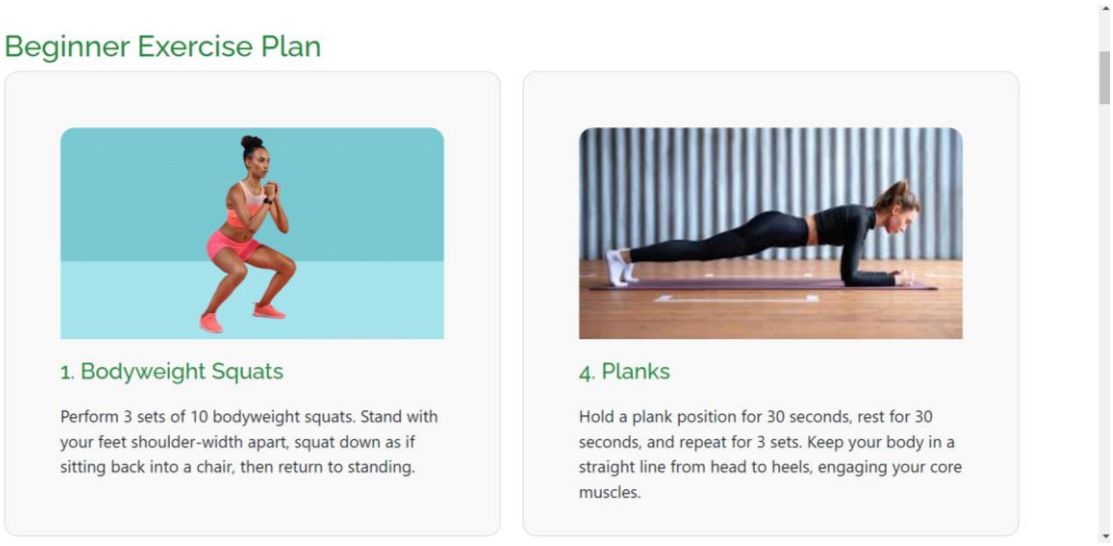
Jedná se o webovou aplikaci, která slouží jako receptář a plánovač denního stravování. Aplikace má několik částí. Pro všechny uživatele, i ty nepřihlášené, je k dispozici domovská stránka, receptář a materiály týkající se zdravého životního stylu.



Obr 1. Homepage

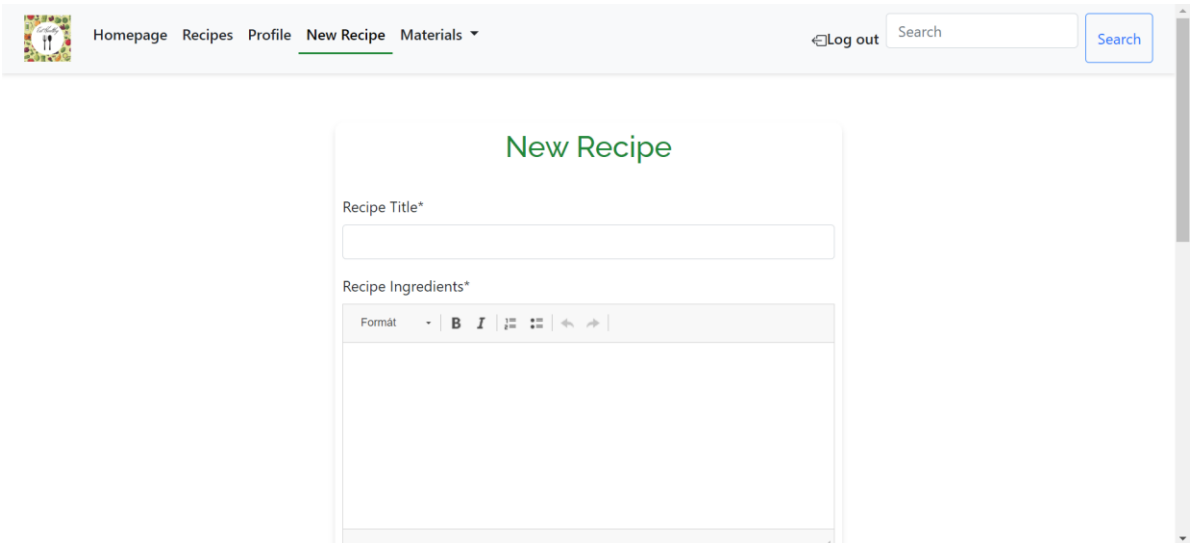


Obr 2. Receptář

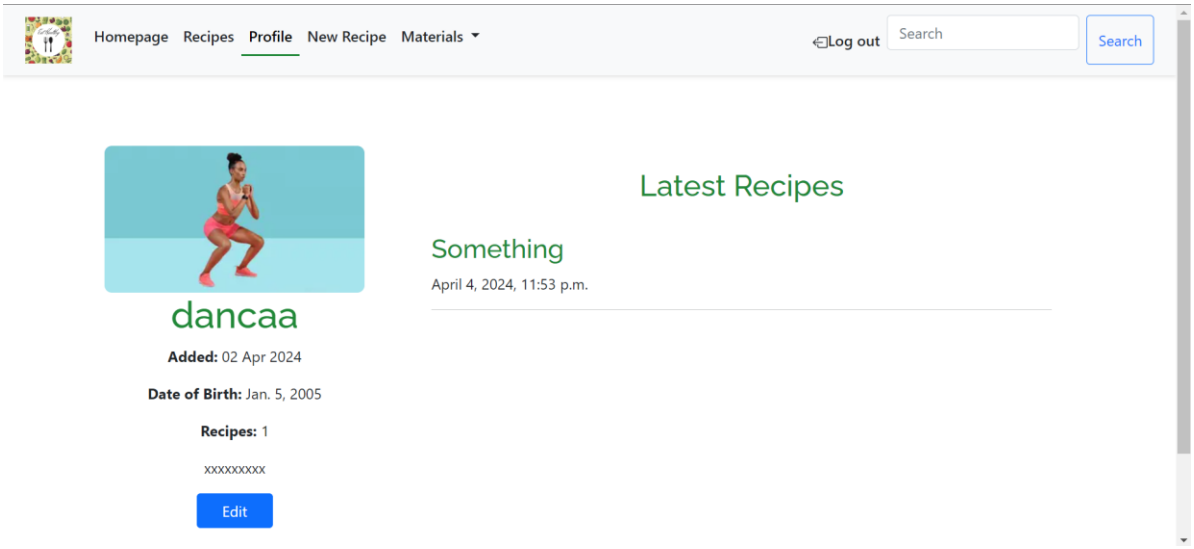


Obr 3. Excercise

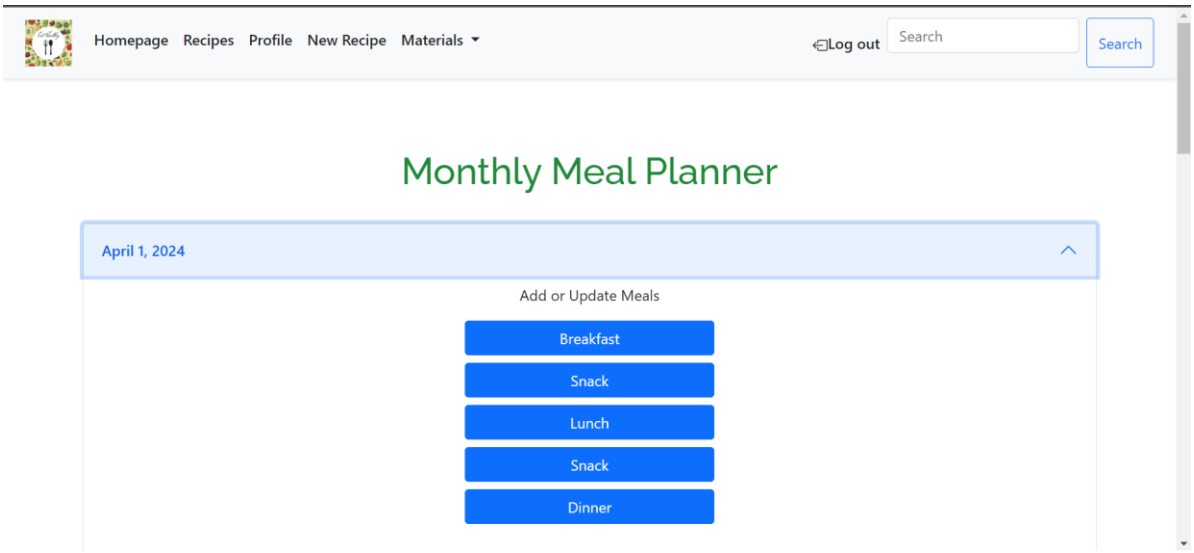
Pro přihlášené uživatele je pak k dispozici stránka umožňující recepty přidávat a následně je také upravovat, pak je jim umožněno si vytvořit profil, nebo si vytvořit jídelníček na měsíc dopředu. Přihlášení funguje přes e-mail nebo uživatelské jméno. Aplikace je uživatelsky přívětivá, je plně responzivní a příjemná vzhledově.



Obr 4. Nový recept



Obr 5. Profil



Obr 6. Denní plán

5. Jednotlivé části aplikace

V mém projektu Django je celkem 5 aplikací. Jedna z nich je aplikace *homepage*, která se stará o správu domovské stránky. Tato aplikace obsahuje jediný template s názvem *homepage.html*. V souboru *views.py* této aplikace je definována třída *Homepage*, která má za úkol zobrazovat obsah domovské stránky. Tato třída obsahuje jedinou metodu, která vrací seznam tří receptů z databáze, které budou zobrazeny na domovské stránce.

Druhou aplikací je *recipes*, která se zabývá správou všech aspektů souvisejících s recepty. To zahrnuje vytváření nových receptů, jejich úpravu a odstranění. Navíc aplikace *recipes* poskytuje seznam receptů, který slouží jako receptář. V kapitole "Recipes" bude podrobněji popsáno, jak tato aplikace funguje a jakým způsobem je možné využívat její funkcionalitu.

Aplikace *socials* se stará o správu uživatelských profilů. V rámci této aplikace máme dva šablony: *profile.html* a *edit_profile.html*. V souboru *forms.py* se nachází formulář sloužící k editaci uživatele. V souboru *models.py* je definována jedna třída *Profile*, která reprezentuje uživatelský profil. Soubor *views.py* obsahuje dvě třídy pohledů: *Socials*, která zajišťuje zobrazení uživatelského profilu, a *EditProfile*, která umožňuje uživatelům editovat svůj profil.

Aplikace *day_plan* slouží k plánování denního jídelníčku a obsahuje dva templaty. První z nich je *day_plan.html*, druhý template je *create_recipe.html*. V souboru *models.py* je definována třída *Meal*. V souboru *views.py* najdeme tři třídy. První z nich, *DayPlan*, druhou třídou je *GetMeal* a poslední třídou je *NewMeal.*

Poslední aplikací je *materials*, která se stará o správu materiálů. Tato aplikace obsahuje tři templaty s názvy *eat_healthy.html, excercise.html, mental_health*. V souboru *views.py* této aplikace jsou definovány třídy *EatHealthy, Excercise, MentalHealth*, které mají za úkol zobrazovat obsah těchto stránek, které jsou definovány zmíněnými templaty.

5.1. Recipes

5.1.1. Templates

Tento soubor obsahuje několik templatů, které jsou využívány v rámci aplikace pro správu receptů. První z nich, *recipes.html*, funguje jako přehledný seznam receptů, kde každý recept je prezentován jako karta s názvem a obrázkem. Uživatelé mohou procházet seznam a kliknutím na konkrétní recept se dostanou na stránku s detaily receptu.

Druhý template, *new_recipe.html*, je určený pro vytváření nových receptů. Obsahuje formulář, ve kterém uživatelé mohou vyplnit název, obrázek, popis, kalorie, typ kuchyně,

ingredience a postup receptu. Po vyplnění a odeslání formuláře se nový recept uloží do databáze a zobrazí se v seznamu receptů.

Třetí template, *edit_recipe.html*, slouží k úpravě existujících receptů. Podobně jako předchozí formulář, umožňuje uživatelům měnit informace o receptu. Po odeslání formuláře se změny uloží do databáze a recept se aktualizuje.

Čtvrtý template, *detail.html*, poskytuje detailní pohled na konkrétní recept. Zobrazuje obrázek, název, autora, datum vytvoření, popis, kalorie, typ kuchyně, ingredience a postup. Pokud je uživatel vlastníkem receptu, má možnost jej upravit nebo smazat.

Poslední template, *delete_recipe.html*, slouží k potvrzení smazání receptu. Uživatelé musí potvrdit své rozhodnutí a po potvrzení se recept odstraní z databáze. Tyto templaty usnadňují uživatelům prohlížení, vytváření, úpravu a mazání receptů v rámci aplikace pro správu receptů, přičemž každý z nich poskytuje specifické funkcionality a možnosti interakce s recepty.

5.1.2. Forms.py

Soubor *forms.py* obsahuje jeden formulář *NewRecipeForm*. Tento formulář slouží k vytváření nových receptů v rámci aplikace pro správu receptů. Uživatelé mohou vyplnit název receptu, seznam ingrediencí, postup přípravy, obrázek receptu, typ jídla, typ kuchyně a počet kalorií. Pole pro zadání ingrediencí a postupu přípravy jsou implementována jako bohatá textová pole s použitím widgetu *RichTextWidget*, který umožňuje formátování textu včetně obrázků a formátovacích prvků.

Formulář obsahuje následující pole:

1. Název receptu: Umožňuje uživatelům zadat název receptu.
2. Ingredience: Pole pro výčet ingrediencí receptu, kde uživatelé mohou zadat seznam ingrediencí.
3. Postup přípravy: Pole pro popis postupu přípravy receptu, kde uživatelé mohou zadat podrobný postup.
4. Obrázek receptu: Umožňuje uživatelům připojit obrázek k receptu.
5. Typ jídla: Pole umožňující uživatelům vybrat typ jídla, například snídaně, oběd nebo večeře.
6. Typ kuchyně: Umožňuje uživatelům vybrat typ kuchyně, ke které recept patří, například italská, asijská nebo mexická.
7. Počet kalorií: Umožňuje uživatelům zadat počet kalorií receptu.

5.1.3. Models.py

Soubor *models.py* obsahuje pouze jednu třídu *Recipe*. *Recipe* je součástí aplikace pro správu receptů a slouží k reprezentaci jednotlivých receptů. Každý recept je spojen s

uživatelem, který ho vytvořil, pomocí pole user, které je implementováno jako cizí klíč k modelu User z frameworku Django.

Atribut title představuje název receptu, který je omezen na maximálně 300 znaků a slouží k identifikaci a popisu receptu. Ingredience receptu jsou uloženy v poli ingredients, které je implementováno jako pole bohatého textu. To umožňuje uživatelům zadávat podrobné informace o potřebných surovinách pro přípravu receptu.

Postup přípravy receptu je uložen v poli instructions, které je rovněž implementováno jako bohaté textové pole. Uživatelé mohou zde zadat podrobný postup krok za krokem. Pro zobrazení obrázku receptu slouží pole image, které je typu *ResizedImageField*. Uživatelé mohou nahrát obrázek receptu, který je následně zobrazen spolu s ostatními informacemi o receptu.

Typ jídla, ke kterému recept patří, je určen v poli *meal_type*, které umožňuje volbu z možností jako snídaně, svačina, oběd nebo večeře. Pole *cuisine_types* definuje typ kuchyně, ke které recept patří, a nabízí uživatelům možnosti jako česká, francouzská, italská atd.

Informace o počtu kalorií receptu jsou uloženy v poli *calories*, které je typu *IntegerField*. Uživatelé mohou zadat počet kalorií, což poskytuje informaci o energetické hodnotě receptu. Pole *posted_date* obsahuje datum a čas, kdy byl recept vytvořen nebo posledně upraven.

Třída *Order* definuje výchozí pořadí, ve kterém jsou recepty zobrazeny. Recepty jsou v základním nastavení řazeny podle data jejich vytvoření sestupně, takže nejnovější recepty jsou zobrazeny první.

Metoda *__str__* definuje textovou reprezentaci receptu, která bude použita při zobrazení receptu v administrátorském rozhraní a jiných místech, kde je vyžadována textová reprezentace. Vrací název receptu ve formátu řetězce.

5.1.4. Views.py

Tento soubor obsahuje několik tříd, které řídí chování aplikace v rámci správy receptů. Každá třída pohledu poskytuje specifickou funkcionalitu a interakci s daty receptů. Třída Recipes řídí zobrazení seznamu receptů uživatelům. Umožňuje uživatelům procházet tento seznam a vyhledávat recepty podle klíčových slov.

Specifikuje název šablony, ve které se seznam receptů zobrazí - v našem případě je to šablona s názvem *recipes.html,* umístěná ve složce *recipes.* Dále definuje model, kterým je v tomto případě model *Recipe*, obsahující veškeré informace o receptech.

Tato třída také nastavuje kontextový objekt pro šablonu, kterým je seznam receptů, a pojmenovává ho *recipes*. Metoda *get_queryset()* se používá k získání querysetu, tj. seznamu objektů, který bude zobrazen. Nejprve získává klíčové slovo z URL parametru *query*, které určuje, jaké recepty mají být zobrazeny. Pokud je klíčové slovo k dispozici, provede filtrování

receptů podle názvu, instrukcí, typu kuchyně a typu jídla podle zadaného klíčového slova. Pokud uživatel nezadá žádné klíčové slovo do pole pro vyhledávání, budou zobrazeny všechny recepty dostupné v databázi.

```
class Recipes(ListView):
    template_name = "recipes/recipes.html"
    model = Recipe
    context_object_name = "recipes"

    def get_queryset(self, **kwargs):
        q = self.request.GET.get("query")
        if q:
            recipes = self.model.objects.filter(
                Q(title__icontains=q)
                | Q(instructions__icontains=q)
                | Q(cuisine_types__icontains=q)
                | Q(meal_type__icontains=q)
            )
        else:
            recipes = self.model.objects.all()
        return recipes
```

Kód 1.: Třída Recipes

Třída *Detail* poskytuje detailní zobrazení konkrétního receptu. Zobrazuje název receptu, obrázek, popis, typ jídla, typ kuchyně a počet kalorií. Pokud je uživatel vlastníkem receptu, má možnost editovat nebo smazat recept.

```
class Detail(DetailView):
    template_name = "recipes/detail.html"
    model = Recipe
    context_object_name = "recipe"
```

Kód 2.: Třída Detail

Další třídou je *NewRecipe*, která uživatelům umožňuje vytvářet nové recepty. Po vyplnění formuláře a jeho odeslání se nový recept uloží do databáze a následně se zobrazí v seznamu receptů.

Určuje název šablony, která bude použita pro zobrazení formuláře pro vytvoření nového receptu - v našem případě se jedná o šablonu s názvem *new_recipe.html*, umístěnou ve složce *recipes.*

Dále specifikuje model, který bude třída vytvářet - v tomto případě je to model *Recipe,* který obsahuje veškeré informace o receptu, jako je název, ingredience, postup, obrázek atd.

Formulář, který bude použit pro zadání informací o novém receptu, je definován pomocí třídy formuláře *NewRecipeForm*.

Po úspěšném odeslání formuláře se uživatel přesměruje zpět na seznam receptů pomocí URL adresy */recipes/.* Metoda *form_valid()* se používá k ověření platnosti formuláře a přidání informace o uživateli, který vytvořil recept, do instance receptu před jeho uložením do databáze. Tímto způsobem se zajišťuje, že každý nový recept bude spojen s konkrétním uživatelem, který ho vytvořil.

```
class NewRecipe(LoginRequiredMixin, CreateView):
    template_name = "recipes/new_recipe.html"
    model = Recipe
    form_class = NewRecipeForm
    success_url = "/recipes/"

    def form_valid(self, form):
        form.instance.user = self.request.user
        return super(NewRecipe, self).form_valid(form)
```

Kód 3.: Třída NewRecipe

Třída *Delete* umožňuje uživatelům odstranit recept z databáze. Před smazáním receptu je vyžadováno potvrzení uživatele.

```
class Delete(LoginRequiredMixin, UserPassesTestMixin, DeleteView):
    model = Recipe
    success_url = "/recipes/"

    def test_func(self):
        return self.request.user == self.get_object().user
```

Kód 4.: Třída Delete

Třída *Edit* umožňuje uživatelům upravit existující recept. Po odeslání formuláře se provedené změny uloží do databáze a recept je aktualizován.

template_name: Specifikuje název šablony, která bude použita pro zobrazení formuláře úpravy receptu. V našem případě je to šablona s názvem *edit.html*, umístěná ve složce *recipes.*

model: Určuje model, který bude tato třída upravovat. V našem případě je to model Recipe, který reprezentuje jednotlivé recepty.

form_class: Definuje třídu formuláře, která bude použita pro úpravu receptu. Zde je použita třída *NewRecipeForm*, která obsahuje pole a validace pro úpravu receptu.

success_url: Specifikuje URL adresu, na kterou bude uživatel přesměrován po úspěšném odeslání formuláře úpravy receptu. V našem případě je to */recipes/,* což znamená, že uživatel bude přesměrován zpět na seznam receptů.

test_func: Metoda *test_func()* provádí test, zda je aktuálně přihlášený uživatel vlastníkem upravovaného receptu. Tato metoda ověřuje, zda je ID přihlášeného uživatele shodné s ID uživatele, který vytvořil upravovaný recept. Pokud ano, uživatel má oprávnění k úpravě receptu, jinak není oprávněn a přístup je zamítnut.

```
class Edit(LoginRequiredMixin, UserPassesTestMixin, UpdateView):
    template_name = "recipes/edit.html"
    model = Recipe
    form_class = NewRecipeForm
    success_url = "/recipes/"

    def test_func(self):
        return self.request.user == self.get_object().user
```

Kód 5: Třída Edit

5.2. Socials

Aplikace *socials* slouží k správě uživatelských profilů. Obsahuje dva templaty, *profile.html*, která zobrazuje uživatelský profil, a *edit_profile.html*, která umožňuje uživatelům editovat svůj profil pomocí formuláře. V souboru *forms.py* se nachází formulář sloužící k editaci uživatele.

V souboru *models.py* je definována třída *Profile*, která obsahuje tři metody. Nejvýznamnější z nich automaticky vytváří profil pro nově registrovaného uživatele. Profil je vytvořen automaticky po registraci uživatele.

Soubor *views.py* obsahuje dvě třídy. *Socials,* která slouží k zobrazení uživatelského profilu. Metoda této třídy získává data pro šablonu *profile.html*. Profil uživatele je získán z databáze podle identifikátoru uživatele (primární klíč) z URL adresy. Následně jsou data o profilu a instance formuláře *ProfileForm* přidány do kontextu a předány šabloně.

```
class Socials(TemplateView):
    template_name = "socials/profile.html"

    def get_context_data(self, **kwargs):
        profile = Profile.objects.get(user=self.kwargs["pk"])
        context = {
            "profile": profile,
            "form": ProfileForm(instance=profile),
        }

        return context
```

Kód 6.: Třída Socials

EditProfile, která umožňuje uživatelům editovat svůj profil. Vyžaduje, aby uživatel byl přihlášený a zároveň testuje, zda aktuálně přihlášený uživatel odpovídá uživateli, jehož profil se snaží editovat. Pokud uživatel projde testem, může editovat svůj profil. Metoda *form_valid* nastavuje cílovou URL pro přesměrování po úspěšném odeslání formuláře na adresu zobrazení profilu s odpovídajícím uživatelem.

```
class EditProfile(LoginRequiredMixin, UserPassesTestMixin, UpdateView):
    template_name = "socials/edit_profile.html"
    form_class = ProfileForm
    model = Profile

    def form_valid(self, form):
        self.success_url = f'/profiles/user/{self.kwargs["pk"]}'
        return super().form_valid(form)

    def test_func(self):
        return self.request.user == self.get_object().user
```

Kód 7.: Třída EditProfile

5.3. Day_plan

Aplikace *day_plan* slouží k plánování denního jídelníčku a obsahuje dva templaty. První z nich je *day_plan.html*, který zobrazuje plánovač denního jídelníčku s seznamem dnů v aktuálním měsíci. Uživatel má zde možnost přidat jídlo do plánovače pro každý den, a to jak snídani, tak oběd, večeři nebo svačinu. Druhý template, *create_recipe.html*, umožňuje uživatelům vyhledat a vybrat recept pro přidání do plánovače. Obsahuje formulář pro vyhledávání receptů podle klíčových slov a možnost nastavení maximálního počtu kalorií.

V souboru *models.py* je definována třída *Meal*, která reprezentuje jednotlivá jídla v plánovači. Tato třída obsahuje informace o uživateli, receptu, typu jídla a datu.

V souboru *views.py* najdeme tři třídy. Třída *DayPlan* je klíčovým prvkem naší aplikace pro plánování denního jídelníčku. Její hlavní funkcí je zobrazení plánovače pro aktuální měsíc, který umožňuje uživatelům organizovat své jídla podle typu a dne v týdnu. V rámci metody *get_context_data()* třídy *DayPlan* je nejprve získán aktuální datum pomocí modulu *datetime*. Poté jsou vypočítány všechny dny v aktuálním měsíci pomocí funkce *monthrange()* z modulu *calendar.* Tímto způsobem jsou vytvořeny seznamy dnů, které jsou následně předány do šablony pro zobrazení plánovače.

Dalším krokem je načtení informací o plánovaných jídlech pro daného uživatele a aktuální měsíc. To je provedeno pomocí filtru *Meal.objects.filter()* nad modelem *Meal,* který

obsahuje data o plánovaných jídlech. Jídla jsou filtrována podle uživatele (*self.request.user*) a data, která odpovídají seznamu dnů v aktuálním měsíci.

Nakonec jsou získaná data o jídlech seřazena podle typu jídla v pořadí: snídaně, svačina, oběd, svačina, večeře. To je provedeno pomocí funkce *reorder()*, která definuje pořadí typů jídel. Seřazená data jsou pak předána do kontextu a následně zobrazena v šabloně *day_plan/day_plan.html*. Tento přístup umožňuje uživatelům snadno organizovat své jídelní plány a sledovat svou stravu v průběhu celého měsíce, což přispívá k lepšímu zdraví a životnímu stylu.

```
class DayPlan(LoginRequiredMixin, TemplateView):
    template_name = "day_plan/day_plan.html"

    def get_context_data(self, **kwargs):
        today = datetime.date.today()
        days_in_mon = calendar.monthrange(today.year, today.month)[1]

        days = [
            datetime.date(today.year, today.month, day)
            for day in range(1, days_in_mon + 1)
        ]

        meals = Meal.objects.filter(
            user=self.request.user, meal_date__in=days
        ).order_by(reorder(meal_type=["breakfast", "snack", "lunch", "snack",
"dinner"]))

        context = {"days": days, "meals": meals}

        return context
```

Kód 8.: Třída DayPlan

Druhou třídou je *GetMeal*, která umožňuje uživatelům vyhledávat recepty pro konkrétní typ jídla a přidávat je do plánovače jídelníčku. Její hlavní funkcionalitou je zpracování požadavků na vyhledávání receptů podle klíčových slov a nastavení maximálního počtu kalorií.

V metodě *get_context_data()* třídy *GetMeal* jsou nejprve získány parametry z URL adresy, zejména maximální počet kalorií (*calories*) a klíčové slovo pro vyhledávání (*query*). Poté je provedena kontrola, zda uživatel zadal klíčové slovo pro vyhledávání receptů. Pokud ano, aplikace filtruje recepty podle tohoto klíčového slova a zároveň zohledňuje nastavení maximálního počtu kalorií.

Recepty jsou filtrovány podle názvu, ingrediencí, typu kuchyně a instrukcí. Pokud uživatel nezadá klíčové slovo pro vyhledávání, ale specifikuje maximální počet kalorií, aplikace zobrazí recepty, které splňují toto kritérium a jsou určeny pro konkrétní typ jídla.

Pokud ani klíčové slovo pro vyhledávání, ani maximální počet kalorií nejsou zadány, aplikace zobrazí všechny recepty pro daný typ jídla. Pokud jsou nalezeny vhodné recepty, je náhodně vybrán jeden z nich a přidán do kontextu pro zobrazení ve formuláři pro přidání jídla do plánovače. V opačném případě je do kontextu přidán pouze typ jídla a datum, aby uživatel mohl pokračovat ve vyhledávání nebo se vrátit zpět bez vybrání konkrétního receptu.

```
class GetMeal(TemplateView):
    template_name = "day_plan/create_recipe.html"

    def get_context_data(self, **kwargs):
        calories = self.request.GET.get("calories")
        query = self.request.GET.get("search")

        if query:
            if not calories:
                calories = 1000

            calories = int(calories)

            recipes = Recipe.objects.filter(
                  Q(title__icontains=query)
                | Q(ingredients__icontains=query)
                | Q(cuisine_types__icontains=query)
                | Q(instructions__icontains=query)
                & Q(calories__lte=calories)
                & Q(meal_type=kwargs["meal_type"])
            )

        elif calories:
            recipes = Recipe.objects.filter(
                calories__lte=calories, meal_type=kwargs["meal_type"]
            )

        else:
            if len(Recipe.objects.all()) < 1:
                recipes = []
            else:
                recipes = Recipe.objects.filter(meal_type=kwargs["meal_type"])

        if len(recipes) > 0:
            recipe = random.choice(recipes)
            context = {
                "meal_date": kwargs["meal_date"],
                "meal_type": kwargs["meal_type"],
                "recipe": recipe,
            }

        else:
```

```
            context = {
                "meal_date": kwargs["meal_date"],
                "meal_type": kwargs["meal_type"],
            }
        return context
```

Kód 9.: Třída GetMeal

Poslední třídou je třída *NewMeal*, která zpracovává přidání nového jídla do plánovače a ukládá informace o jídle do databáze. V metodě *post()* této třídy jsou nejprve získány potřebné informace z URL adresy, konkrétně primární klíč (*pk*) receptu, datum jídla (*meal_date*) a typ jídla (*meal_type*). Poté je načten recept z databáze podle získaného primárního klíče. Následně je vytvořeno nebo aktualizováno záznam v databázi tabulky *Meal*.

Pokud záznam s daným datem a typem jídla již existuje, jsou aktualizovány informace o tomto jídle (například pokud uživatel změnil recept). Pokud záznam neexistuje, je vytvořen nový záznam s informacemi o uživateli, receptu, datu jídla a typu jídla. Poté je uživatel přesměrován na domovskou stránku aplikace pomocí metody *HttpResponseRedirect* a návratového URL definovaného pomocí funkce reverse, která vrátí URL pro zobrazení plánovače jídelníčku.

Tímto způsobem třída *NewMeal* zajišťuje efektivní zpracování přidání nového jídla do plánovače a ukládání relevantních informací o jídle do databáze, což přispívá k plynulému fungování naší aplikace pro plánování jídelníčku.

```
class NewMeal(View):
    def post(self, *args, **kwargs):
        pk = kwargs["pk"]
        recipe = Recipe.objects.get(pk=pk)
        meal_date = kwargs["meal_date"]
        meal_type = kwargs["meal_type"]

        meal, created = Meal.objects.update_or_create(
            meal_date=meal_date,
            meal_type=meal_type,
            defaults={
                "user": self.request.user,
                "recipe": recipe,
                "meal_date": meal_date,
            },
        )

        return HttpResponseRedirect(reverse("day_plan"))
```

Kód 10.: Třída NewMeal

6. Závěr

V mé aplikaci jsem popsala jednotlivé části, které zahrnují správu receptů, uživatelských profilů, plánování denního jídelníčku a správu materiálů souvisejících se zdravým životním stylem. Každá část aplikace poskytuje uživatelům specifickou funkcionalitu a přispívá k celkovému uživatelskému zážitku.

V aplikaci pro správu receptů jsme vytvořili komplexní systém pro vytváření, prohlížení, úpravu a mazání receptů. Uživatelé mohou snadno procházet seznam receptů, vyhledávat recepty podle klíčových slov a přidávat nové recepty pomocí intuitivního formuláře. Díky bohatým textovým polím a možnosti připojit obrázek k receptu mohou uživatelé detailně popsat a vizualizovat své recepty.

Aplikace pro správu uživatelských profilů umožňuje uživatelům spravovat své osobní údaje a prezentovat svůj profil ostatním uživatelům. Uživatelé mohou editovat svůj profil, změnit své údaje a nahrát profilový obrázek. Tato funkcionalita přispívá k interaktivitě a sociálnímu aspektu naší aplikace.

Plánovač denního jídelníčku poskytuje uživatelům prostředek k organizaci a plánování jejich stravovacích návyků. Uživatelé mohou jednoduše plánovat své jídla na každý den a vyhledávat recepty odpovídající jejich preferencím. Tato funkcionalita přispívá k lepšímu zdraví a životnímu stylu našich uživatelů.

Aplikace pro správu materiálů nabízí uživatelům zdroje a informace související se zdravým životním stylem. Uživatelé mohou procházet různé materiály týkající se stravování, cvičení a duševního zdraví a získat tak užitečné tipy a rady pro zlepšení svého životního stylu.

Celkově moje aplikace poskytuje uživatelům komplexní prostředí pro správu jejich receptů, zdraví a zdravého životního stylu.

7. Zdroje

1. Django dokumentace: Django. Dostupné z: https://www.djangoproject.com/
2. Django Allauth dokumentace: Django Allauth. Dostupné z: https://django-allauth.readthedocs.io/en/latest/installation.html
3. Django Crispy Forms dokumentace: Django Crispy Forms. Dostupné z: https://django-crispy-forms.readthedocs.io/en/latest/install.html
4. Django Reorder na PyPI: PyPI. Dostupné z: https://pypi.org/project/django-reorder/
5. Dokumentace Django Class-Based Views Mixins: Django. Dostupné z: https://docs.djangoproject.com/en/4.1/topics/class-based-views/mixins/
6. Cloudinary: Cloudinary. Dostupné z: https://cloudinary.com/
7. Django ORM Cookbook: Agiliq. Dostupné z: https://books.agiliq.com/projects/django-orm-cookbook/en/latest/query_relatedtool.html
8. Django Generic Display Views: Django. Dostupné z: https://docs.djangoproject.com/en/5.0/topics/class-based-views/generic-display/
9. Django HTTP Redirect Examples: Real Python. Dostupné z: https://realpython.com/django-http-redirect/
10. Django RichTextField na PyPI: PyPI. Dostupné z: https://pypi.org/project/django-richtextfield/
11. Django Cloudinary Storage na PyPI: PyPI. Dostupné z: https://pypi.org/project/django-cloudinary-storage/
12. Canva: Canva. Dostupné z: https://www.canva.com/design/
13. Recept na Low Calorie Baked Oats: Matt's Fit Chef. Dostupné z: https://mattsfitchef.com/low-calorie-baked-oats/
14. Recept na Wheat Porridge: Yummy Valley. Dostupné z: https://yummy-valley.com/blog/wheat-porridge-recipe/
15. Top 10 Tips of Healthy Living: Medium. Dostupné z: https://medium.com/@sumayyashireen567/top-10-tips-of-healthy-living-a68b87653c09
16. Watch Your Portion Sizes: Power Sculpt Fitness. Dostupné z: https://powersculptfitness.wordpress.com/2013/01/13/watch-your-portion-sizes/
17. How to Stay Hydrated: Healthy Food. Dostupné z: https://www.healthyfood.com/advice/how-to-stay-hydrated/
18. Benefits of Squats: Real Simple. Dostupné z: https://www.realsimple.com/health/fitness-exercise/workouts/squat-form
19. Benefits of Push-Ups: Real Simple. Dostupné z: https://www.realsimple.com/benefits-of-push-ups-7563143
20. Lunge Exercise Guide: Healthy Food. Dostupné z: https://www.healthyfood.com/advice/how-to-stay-hydrated/
21. Walking Planks: Tom's Guide. Dostupné z: https://www.tomsguide.com/features/i-did-50-walking-planks-a-day-for-a-week-heres-what-happened-to-my-abs
22. 30-Day Jumping Jack Challenge: NordicTrack. Dostupné z: https://www.nordictrack.co.uk/learn/your-30-day-jumping-jack-challenge/
23. Bicycle Crunches: Steel Supplements. Dostupné z: https://steelsupplements.com/blogs/steel-blog/how-to-do-bicycle-crunches-form-muscles-worked)
24. Running Outdoors: NBC News. Dostupné z: https://www.nbcnews.com/better/health/what-you-need-know-taking-your-run-outdoors-ncna872491
25. Jump Squats: Men's XP. Dostupné z: https://www.mensxp.com/health/fitness/35410-what-are-jump-squats-and-how-to-do-them.html
26. Push-Up Variations: Hy-Vee. Dostupné z: https://www.hy-vee.com/recipes-ideas/advice-how-tos/wellness/health-fitness/push-up-variations
27. Walking Lunges: Hit My Macros. Dostupné z: https://hitmymacros.com/exercises/walking-lunges/
28. Plank with Leg Lift: Men's Health. Dostupné z: https://www.menshealth.com/fitness/a20694666/plank-leg-lift/
29. Russian Twists: PopSugar. Dostupné z: https://www.popsugar.com/fitness/how-do-russian-twist-43932884
30. Sprinting Tips: Ladder. Dostupné z: https://ladder.sport/pages/sprinting-tips/
31. Plyometric Push-Ups: Steel Supplements. Dostupné z: https://steelsupplements.com/blogs/steel-blog/how-to-do-plyo-push-ups-form-benefits
32. Pistol Squats: Freeletics. Dostupné z: https://www.freeletics.com/en/blog/posts/freeletics-exercises-pistol-squats/
33. Single-Leg Romanian Deadlifts: ALO Moves. Dostupné z: https://blog.alomoves.com/movement/how-to-do-single-leg-romanian-deadlift
34. Hanging Leg Raises: Women's Health. Dostupné z: https://www.womenshealthmag.com/fitness/a41960561/how-to-do-hanging-leg-raise/
35. Renegade Rows: CoachWeb. Dostupné z: https://www.coachweb.com/dumbbell-exercises/5799/how-to-do-a-renegade-row
36. How to Listen to Your Body: Joshua Hook. Dostupné z: https://joshuanhook.com/2019/10/02/how-to-listen-to-your-body-4-keys/
37. Benefits of Cooking from Home: The Recipe. Dostupné z: https://www.therecipe.com/10-reasons-you-should-be-cooking-from-home-more-often/
38. Avoiding Emotional Eating: Aspire Health Plan. Dostupné z: https://www.aspirehealthplan.org/tips-avoiding-emotional-eating/
39. Practicing Moderation: Just Mind. Dostupné z: https://justmind.org/how-to-practice-moderation/
40. Yoga and Meditation: Everyday Yoga. Dostupné z: https://www.everydayyoga.com/blogs/guides/yoga-meditation
41. Getting Active: American Heart Association. Dostupné z: https://www.heart.org/en/healthy-living/fitness/getting-active/get-real-about-getting-active
42. Healthy Sleep Tips: National Sleep Foundation. Dostupné z: https://www.sleepfoundation.org/sleep-hygiene/healthy-sleep-tips
43. Essential Self-Care Practices: Wellness Road Psychology. Dostupné z: https://www.wellnessroadpsychology.com/blog/10-essential-self-care-practices
44. Listening to Music Mindfully: Greater Good in Education. Dostupné z: https://ggie.berkeley.edu/practice/listening-to-music-mindfully
45. iOS Screen Time: MakeUseOf. Dostupné z: https://www.makeuseof.com/tag/ios-screen-time/

8. Seznam obrázků

9. Seznam ukázek kódu