“Plate Path”

СОФТУЕРЕН АРХИТЕКТУРЕН ДОКУМЕНТ

Автори на проект 1:
Десимира Пламенова Димитрова - 471221086
Йоан Емилов Петров - 471222102
Калоян Огнянов Първанов - 471221115
Мертол Адемов Бекиров - 471221092
Веселин Руменов Инзов - 471221103
Кристиан Диез Петров - 471221071

ИСТОРИЯ НА ПРОМЕНИТЕ

НОМЕР НА ДОКУМЕНТА:	ПУБЛИКАЦИЯ/ПРОМЯНА:	ДАТА НА ПУБЛИКАЦИЯ/ПРОМЯНА:
1	v1	29.10.2023

Съдържание

Въведение.....3
Предназначение..... 3
Обхват..... 3
Актьори.....4
Използвани термини и символи..... 5
Източници.....5
Архитектурен обзор.....5
Нефункционални изисквания..... 21

Въведение

В епоха, в която здравословният начин на живот и кулинарното откривателство са станали първостепенни, приложението за планиране на хранене „Plate Path“ се очертава като универсално и интуитивно решение. Plate Path е софтуерно приложение, предназначено да даде възможност на потребителите да изготвят персонализирани планове за хранене, да избират от разнообразен набор от рецепти и да се впуснат в кулинарно пътешествие, съобразено с техните индивидуални предпочитания, диетични изисквания и хранителни цели. Този иновативен проект се стреми да революционизира начина, по който хората подхождат към планирането на хранене и проследяването на храненето, като предоставя безпроблемно, удобно за потребителя и богато на функции изживяване.

Софтуерната архитектура на Plate Path е структурирана така, че да отговаря на високите стандарти за производителност, надеждност и сигурност, очаквани от едно модерно приложение за планиране на хранене. То гарантира, че потребителите могат безпроблемно да навигират през огромно хранилище от рецепти, да персонализират планове за хранене и без усилие да проследяват своя хранителен прием. Тази архитектура е проектирана да предоставя приятно потребителско изживяване, като насърчава ангажираността и удовлетворението на потребителите.

Ключовите архитектурни елементи на Plate Path включват компоненти за генериране на план за хранене, управление на рецепти, удостоверяване и оторизация на потребителите, съхранение и извличане на данни и съвместимост между платформи. Тези елементи са преплетени, за да улеснят основните функционалности на приложението, като същевременно поддържат основни нефункционални изисквания като мащабируемост, поддръжка и сигурност.

Участници

- Разработчици
- Потребители: Хората, които ще използват PlatePath за планиране на храненето си.

Предназначение

Този документ служи като ръководство за архитектурата на софтуера PlatePath, който има за цел да помогне на потребителите при планирането на техните хранения. Документът включва информация за разработката, тестването и поддръжката на приложението.

Обхват

Основната цел на проекта Plate Path е да разработи удобно за потребителя и богато на функции софтуерно приложение за планиране на хранене, което позволява на потребителите да създават, персонализират и управляват своите планове за хранене.

Софтуерът има за цел да рационализира процеса на планиране на хранене, да осигури достъп до разнообразна и хранително точна база данни с рецепти и да помогне на потребителите да постигнат своите диетични и хранителни цели.

Този документ описва архитектурата на PlatePath през целия му жизнен цикъл, който включва следните етапи:

- Разработка: Създаване на софтуерната архитектура и функционалност на PlatePath.
- Тестване: Проверка и валидност на функциите и ефективността на приложението.
- Поддръжка: Поддръжка, актуализиране и подобрения на PlatePath след излизането му в експлоатация.

Актьори

Всяка заинтересована страна се интересува от различни характеристики на системата. Ето списък на ролите на заинтересованите страни, взети предвид при разработването на архитектурата, описана в този документ.

1. **Крайни потребители:**
   - Описание:

   Крайните потребители са лицата, които използват софтуера за планиране на хранене Plate Path, за да създават планове за хранене, да откриват рецепти и да проследяват своето хранене. Те идват от различен произход и имат различни диетични предпочитания и здравни цели.
   - Отговорности:

   Предоставяне на обратна връзка за потребителското изживяване, докладване на проблеми или грешки и реалност за активността на потребителя чрез използване на приложението.
   - Притеснения:
     - Удобство за потребителя и лесна навигация.
     - Гъвкавост и персонализиране на планирането на храненето.
     - Наличие на разнообразни и привлекателни рецепти.
     - Поверителността на данните и сигурността на личната информация.
     - Достъпност на уеб приложение.

2. **Разработчици:**
   - Описание:

   Разработчиците с лицата, отговорни за изграждането и поддръжката на софтуера Plate Path. Работите върху кода, архитектурата и цялостната функционалност на софтуера.

- Отговорности:

Разработване на код, проектиране на системна архитектура, осигуряване на машинност, оптимизиране на производителността и поддържане на целостта на софтуера.

- Притеснения :
    - Мащабируемост на системата за посрещане на нарастващ брой потребители.
    - Поддръжка и модулност на кодовата база.
    - Интегриране на API на трети страни (напр. бази данни за хранилище, проследяване на фитнес).
    - Ефективност при съхранение и извличане на данни.

3. Системни администратори :

- Описание:

Системните администратори управляват сървърната инфраструктура, следят производителя на системата и гарантират достоверността и надеждността на софтуера Plate Path.

- Отговорности:

Поддръжка на сървъра, наблюдение на системни мерки, сигурност за архивиране, възстановяване на данни и осигуряване на пълната работа на системата.

- Притеснения :
    - Работно време и надеждност на системата.
    - Мониторинг и поддръжка на сървърна инфраструктура.
    - Мерки за сигурност, включително архивиране и възстановяване на данни.

Използвани термини и символи

- API: Приложение за програмиране на интерфейс.
- HTTP: Протокол за предаване на хипертекст.
- MVC: Архитектурен шаблон за създаване на потребителски интерфейс.
- GIT: Система за контрол на версиите на разработка.
- ORM: Библиотека, с която се извършват заявки и манипулации върху база данни, като се използва обектно-ориентиран подход.
- UML: Стандартен визуален моделиращ език, който се използва за документиране на бизнес процеси и софтуерни архитектури.
- CSRF: Атака, която принуждава потребителя да изпълни нежелани действия в уеб приложение, в което той е автентифициран.
- xUnit: инструмент за създаване и тестване на софтуер

Източници

1. https://www.eatthismuch.com/
2. https://fdc.nal.usda.gov/
3. https://plantuml.com/

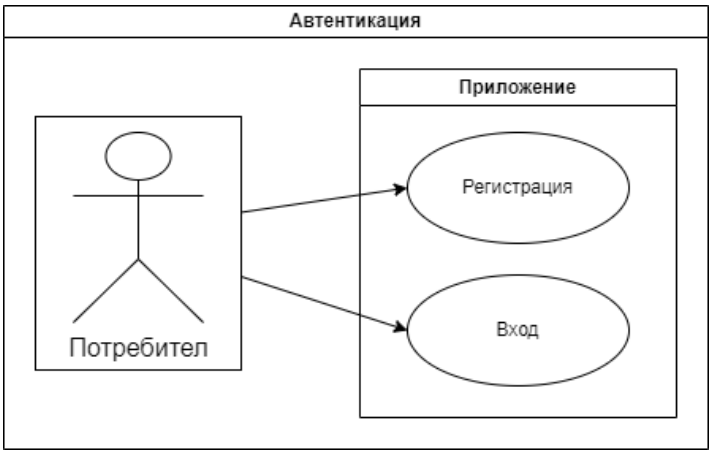
Архитектурен обзор

1. Use-case изглед

В този изглед се представят функционалните възможности на софтуера. Разгледани са основните сценарии от гледната точка на потребителите (ползвателите), които са представени чрез UML Use-case диаграми.

1.1. Автентикация:

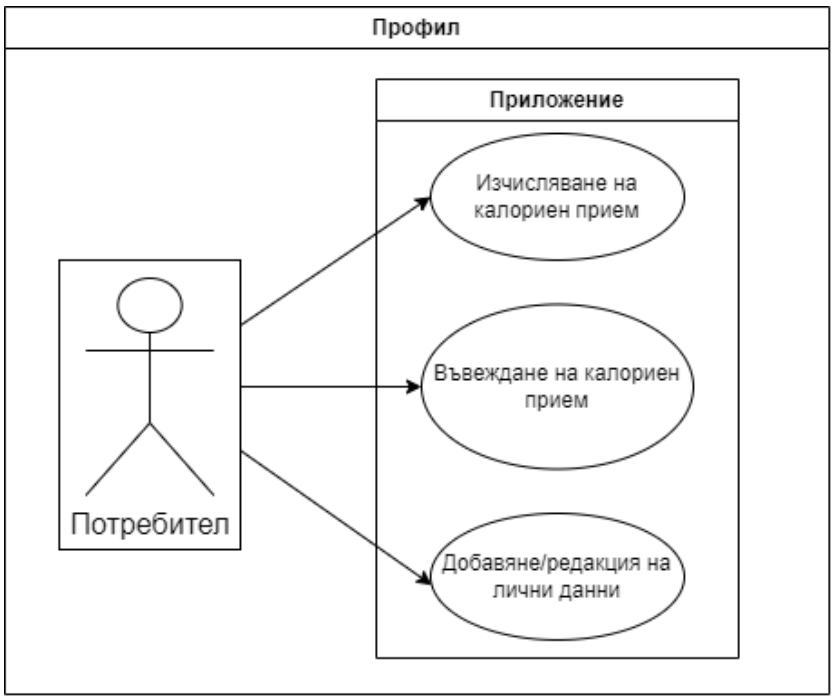
Всеки потребител ще получава достъп до системата, използвайки своя потребителски профил, който трябва да създаде предварително. След регистрация, потребителят може да влезе в системата, както е онагледено в следната диаграма:



- При случая на ползване “Регистрация” потребителят въвежда данните си, за да създаде нов профил в системата.
- При случая на ползване “Вход” потребителят използва данните на вече регистрирания му профил, за да влезе в системата и да получи достъп до нейните функционалности.

1.2. Профил:

Всеки потребител може да достъпи страница, където да разглежда и променя детайли относно своя профил. Детайлите включват въвеждане или изчисляване на калориен прием и промяна на личните данни.



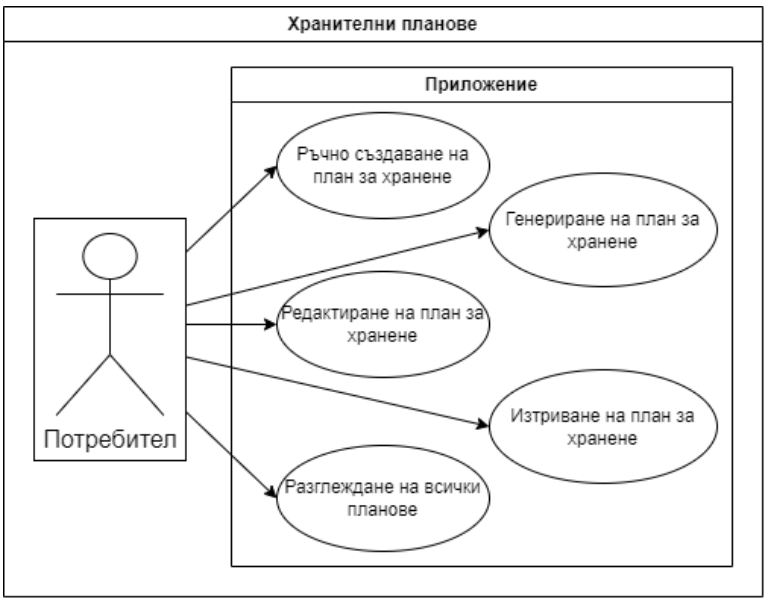
- Изчисляването или въвеждането на калориен прием се използва за генериране на хранителни планове в другите части на системата.
- Ако потребителят не знае какъв калориен прием цели, може да го изчисли, използвайки вградения в системата калкулатор.
- Потребителят може да редактира личните данни, които е предоставил за себе си в своя профил.

1.3. Хранителни планове:

Основната функционалност на приложението е свързана със създаването на хранителни планове. След въвеждане на дневния си калориен прием в секция “Профил”, всеки потребител има достъп до секцията с хранителни планове. Там, след въвеждане на брой хранения на ден, приложението може автоматично да генерира хранителен план, който да се приближава максимално до желания му калориен прием, като едновременно с това спазва ограничението за броя хранения.

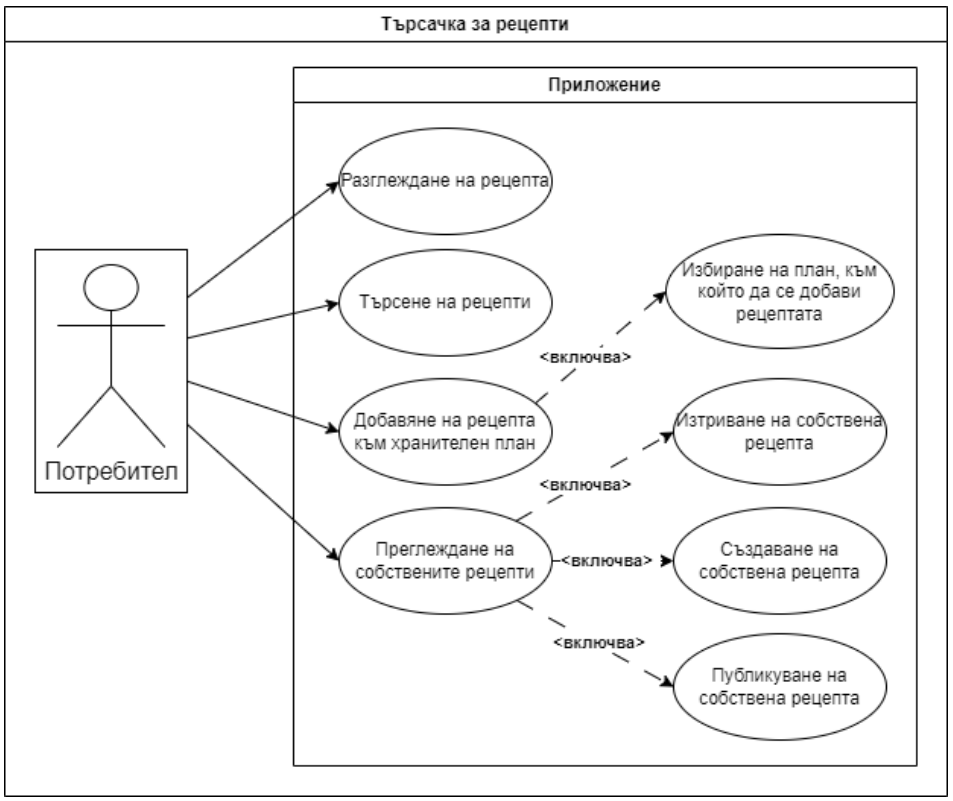
Освен автоматично, хранителните планове могат да бъдат създавани и редактирани ръчно. В този случай, потребителят може да подбира от търсачката за ястия, като в готовия план за хранене автоматично се изчисляват общите калорийни стойности, както и хранителната стойност на всички ястия в плана. Ако потребителят иска да следва план, който е използвал и преди, но не иска да го създава отново, той може да разгледа списък с всички планове, които е създавал или генерирал досега. Ако някой от плановете не е нужен, или не се хареса на потребителя, той може да го изтрие.

Всички досега разгледани функции са онагледени в следната диаграма:



1.4. Браузър за рецепти:

Потребителите имат възможността да ползват търсачката за рецепти, за да търсят ястия, които да добавят към своите хранителни планове.

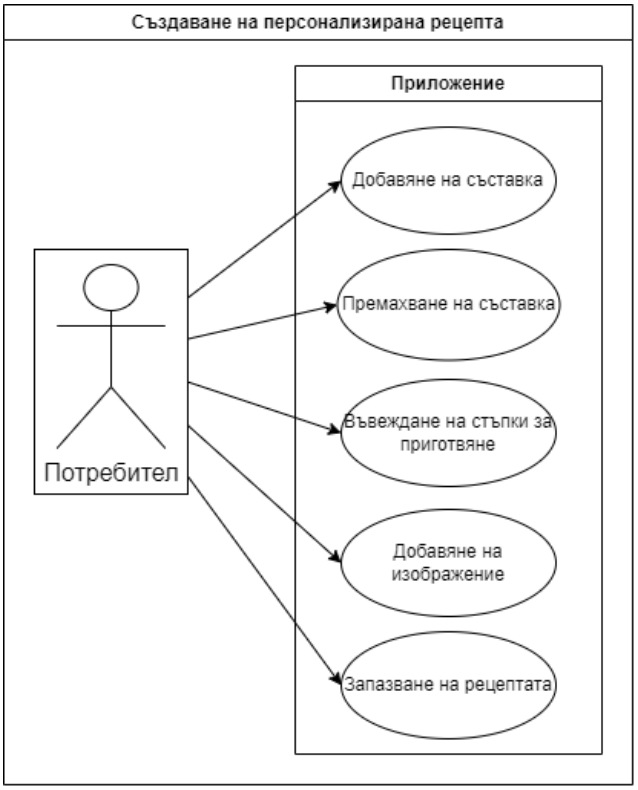


Потребителят може да:

- Разглежда дадена рецепта;
- Търси рецепти - по име или по други критерии;

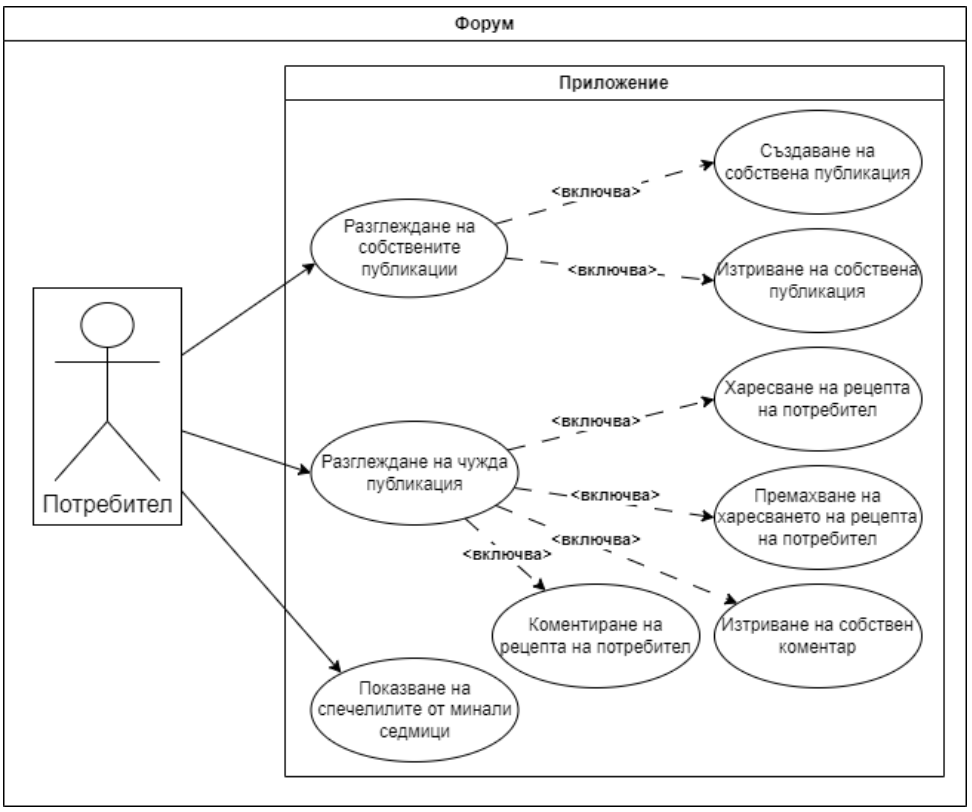
- Добавя рецепти към хранителен план, като трябва да посочи и към кой план да бъде добавена;
- Да преглежда собствените си рецепти, където може да създава, изтрива и публикува собствените си рецепти.

При създаването на персонализирана рецепта, потребителят трябва да избере какви продукти са нужни за направата й, както и техните количества. Освен това трябва да се въведат кратки стъпки за приготвянето на ястието. Също така, може да се добави снимка към рецептата (незадължително). След тези стъпки, потребителят може да запази рецептата с избрано от него име.



1.5. Форум

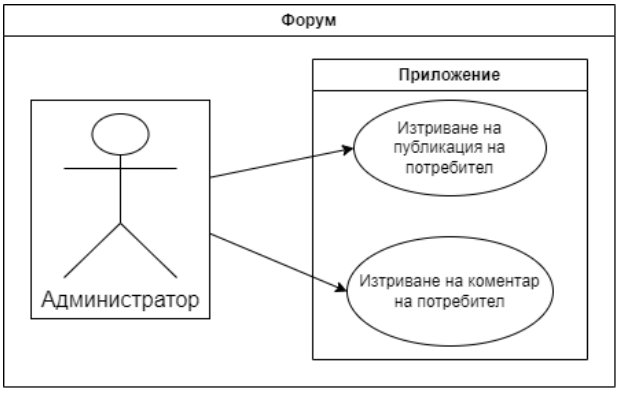
Потребителите могат да използват форума, за да публикуват своите рецепти, като по този начин ги споделят с другите потребители. Те също могат да дискутират и оценяват рецептите на другите, като по този начин се създава седмична класация. В края на всяка седмица, най-високо оценената рецепта се добавя в официалната база данни на системата, като по този начин тя става достъпна за всички потребители. Тези случаи на ползване са онагледени в следните диаграми:



В горната диаграма са описани случаите на ползване на обикновения потребител на системата. Той може да:

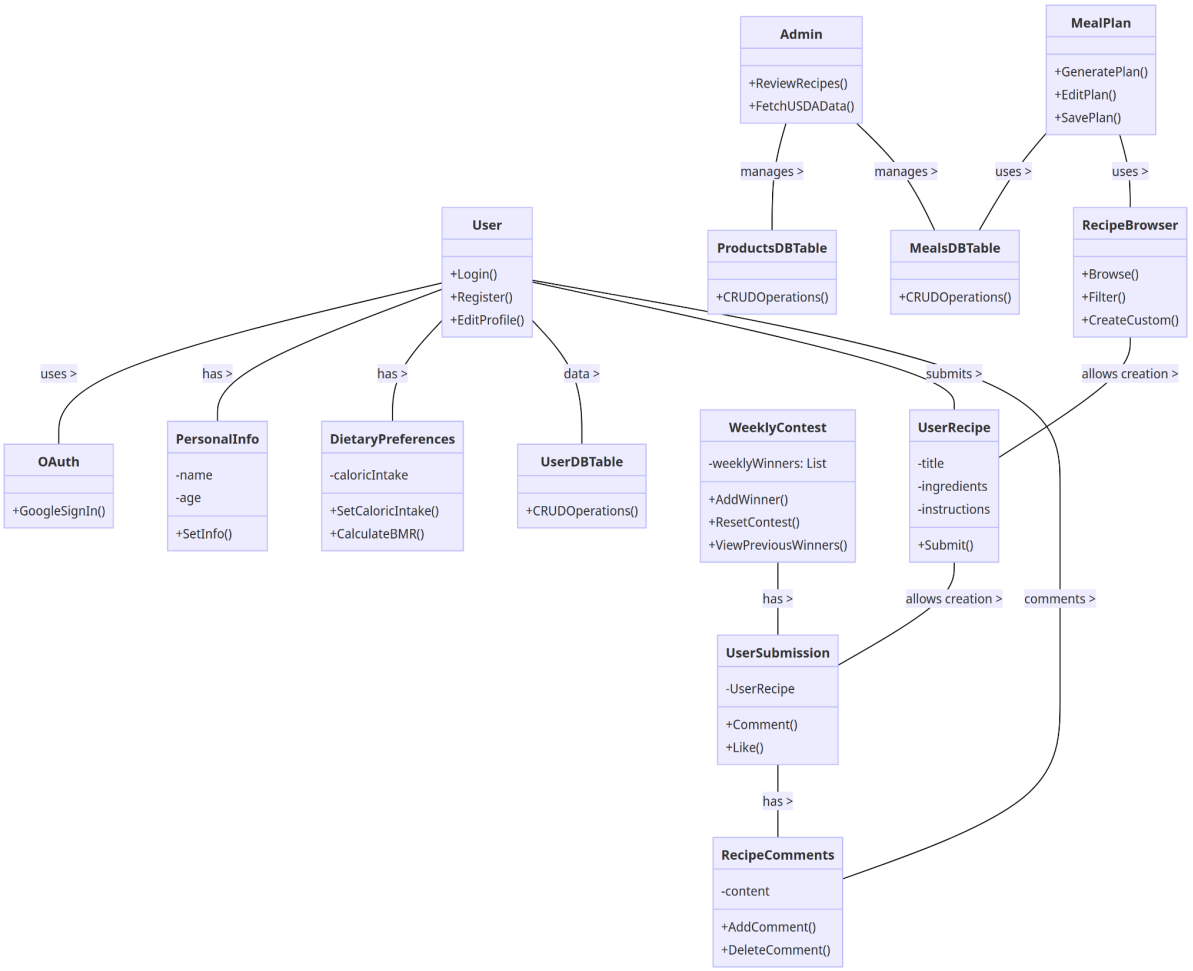
- Разглежда собствените си публикации, където може да ги създава и изтрива;
- Разглежда рецептите на други потребители, където може да ги харесва/премахва харесване и да добавя или премахва собствени коментари за рецептите;
- Да разглежда рецептите, спечелили предишните седмични състезания.

Някои потребители могат да се възползват от свободата си и да използват неуместен език или да публикуват рецепта с нерелевантно съдържание. За да се отсяват тези публикации, администраторите ще могат да преглеждат рецептите на потребителите, и ако се налага, да ги изтриват, както и да изтриват коментари на потребител, ако сметнат, че съдържанието им е неуместно или ненужно.. По този начин се филтрират качествените публикации от публикации тип “спам”.



2. Логически изглед

В логическия изглед се разглежда функционалността, която системата предоставя под формата на класове, пакети и др. Показва се логическата свързаност на компонентите на архитектурата.



В нея могат да се разгледат аспекти като автентикация, промяна на лични данни за потребителите, както и за участието им в седмичните състезания във форума.

Отразени са промените, които администраторите могат да правят по съществуващата база от продукти, както и техните права да филтрират потребителски кандидатури във форума.

3. Процесен изглед

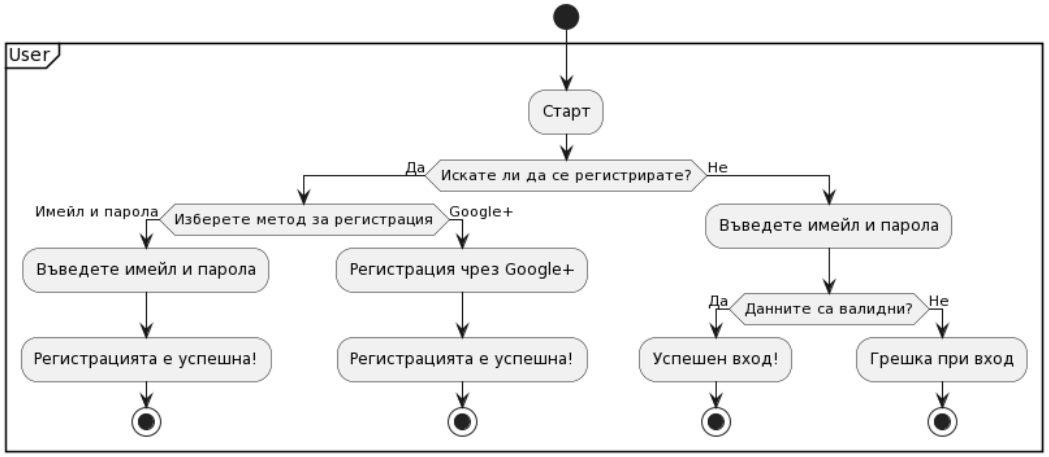
3.1. Алгоритъм за изчисление на нужния калориен прием:

Процесът започва със събирането на лични данни от потребителя. Първоначално потребителят трябва да предостави информация за своята възраст. Тази информация е важна, тъй като възрастта влияе върху основната обмяна на веществата (BMR). Следва въвеждането на текущото тегло на потребителя. Потребителят също така въвежда своята височина. Полът на потребителя е от решаващо значение за изчисленията, тъй като мъжете и жените имат различни формули за изчисление на BMR. Потребителят избира от предложените нива на активност - например “липса на активност”, “слаба активност”, “умерена активност”, “висока активност”, “екстремно висока активност”. Въз основа на предоставените данни, системата изчислява BMR на потребителя чрез подходящата формула. След като получи BMR, системата коригира това число въз основа на избраното ниво на активност на потребителя, за да определи общия дневен енергиен разход (TDEE). На последния етап, системата показва на потребителя препоръчителния му дневен калориен прием.

3.2. Вход / Регистрация

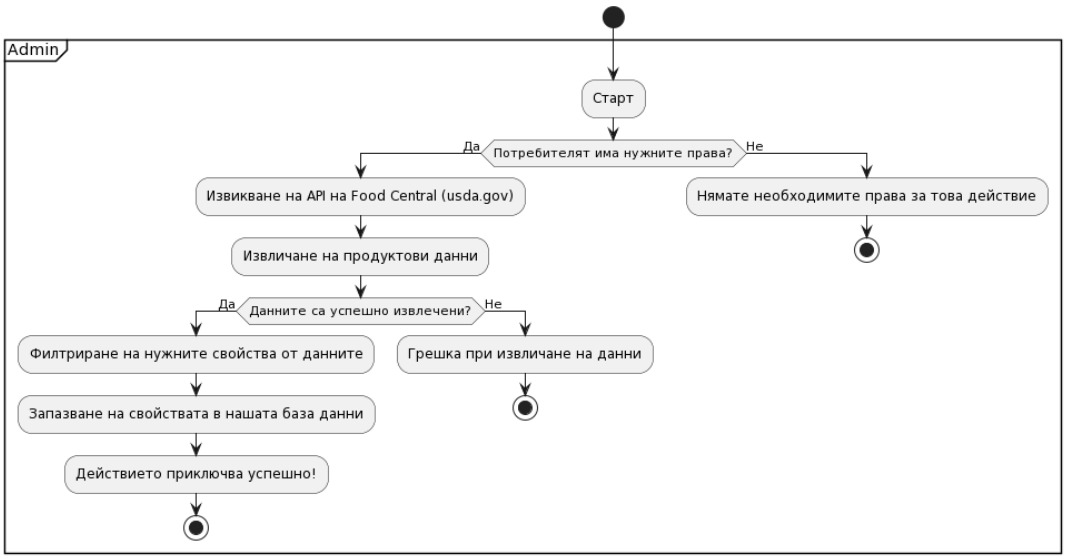
Ако потребителят избере да се регистрира, може да избере между два начина: чрез имейл и парола или чрез Google+. Ако избере регистрация с имейл и парола: Въвежда имейл адрес и парола. Регистрацията приключва успешно. Ако избере регистрация чрез Google+: Процедурата протича автоматично, като данните се извличат от Google+ акаунта. Регистрацията приключва успешно. Ако потребителят не иска да се регистрира, въвежда имейл адрес и парола. Системата проверява валидността на данните. Ако данните са коректни, потребителят влиза успешно в

системата. Ако данните не са валидни, извежда се съобщение за грешка и процесът приключва.



3.3. Извличане на продукти и техните хранителни стойности

Системата проверява дали потребителят има нужните права за извършване на действието. Ако потребителят има нужните права, системата извиква API на Food Central (usda.gov) за извличане на продуктови данни. Системата извлича продуктови данни от Food Central. Системата проверява дали данните са успешно извлечени. Системата филтрира и извлича само нужните свойства от извлечените данни. Тези свойства се запазват в нашата база данни. Действието приключва успешно. Ако има проблем при извличане на данни: Извежда се съобщение за грешка и процесът приключва. Ако потребителят няма нужните права: Извежда се съобщение, че няма необходимите права за това действие и процесът приключва.



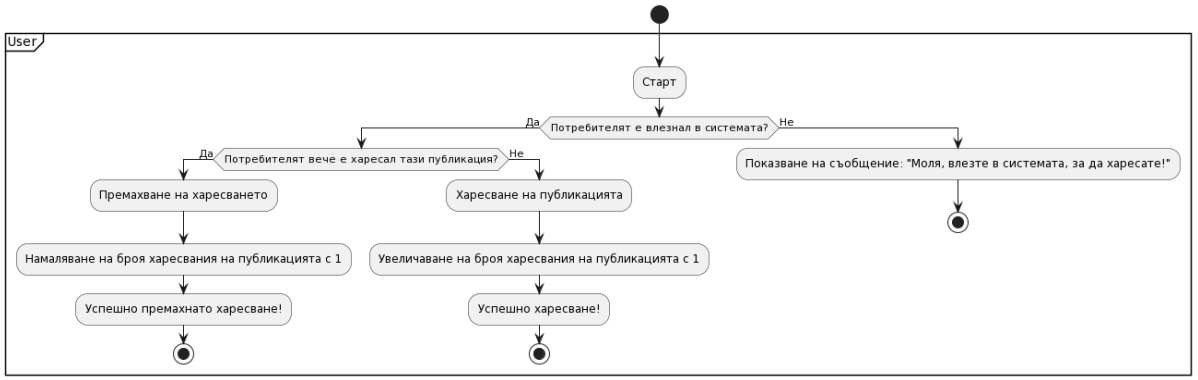
3.4. Публикуване на рецепта във форума (седмично състезание)

Потребителят започва процеса на публикуване на ястие. Системата проверява дали потребителят е влезнал в системата. Ако потребителят е влезнал, системата проверява дали потребителят вече е публикувал ястие за текущото състезание.

Ако потребителят е публикувал ястие: Извежда се съобщение, че потребителят вече е публикувал ястие и процесът приключва. Ако потребителят не е публикувал ястие: Потребителят въвежда информация за ястието. Ястието се публикува във форума. Извежда се съобщение за успешно публикуване и процесът приключва. Ако потребителят не е влезнал: Извежда се съобщение, че трябва да влезе в системата, за да публикува ястие и процесът приключва.

3.5. Харесване или махане на харесване на публикация.

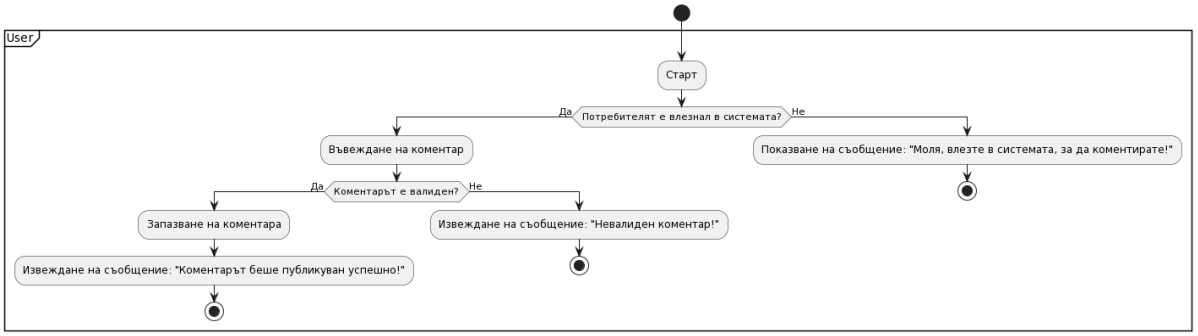
Системата проверява дали потребителят е влезнал в системата. Ако потребителят е влезнал се прави проверка дали е харесал публикацията, ако потребителят е харесал публикацията, броят на харесванията на публикацията се намалява с 1. Извежда се съобщение за успешно премахнато харесване и процесът приключва. Ако потребителят не е харесал публикацията: Публикацията се харесва. Броят на харесванията на публикацията се увеличава с 1. Извежда се съобщение за успешно харесване и процесът приключва. Ако потребителят не е влезнал: Извежда се съобщение, че трябва да влезе в системата, за да хареса публикацията и процесът приключва.



3.6. Коментиране на публикация

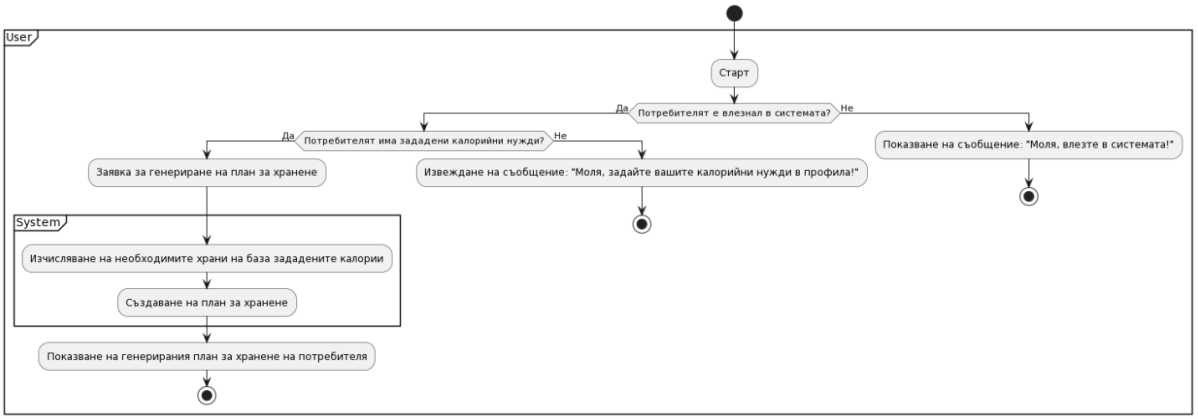
Системата проверява дали потребителят е влезнал в системата. Ако потребителят е влезнал се въвежда коментар. Системата проверява дали коментарът е валиден (например, дали не е празен, не съдържа обидни думи и т.н.). Ако коментарът е валиден се запазва. Извежда се съобщение за успешно публикуван коментар и процесът приключва. Ако коментарът не е валиден - извежда се съобщение за

невалиден коментар и процесът приключва. Ако потребителят не е влезнал: Извежда се съобщение, че трябва да влезе в системата, за да коментира и процесът приключва.



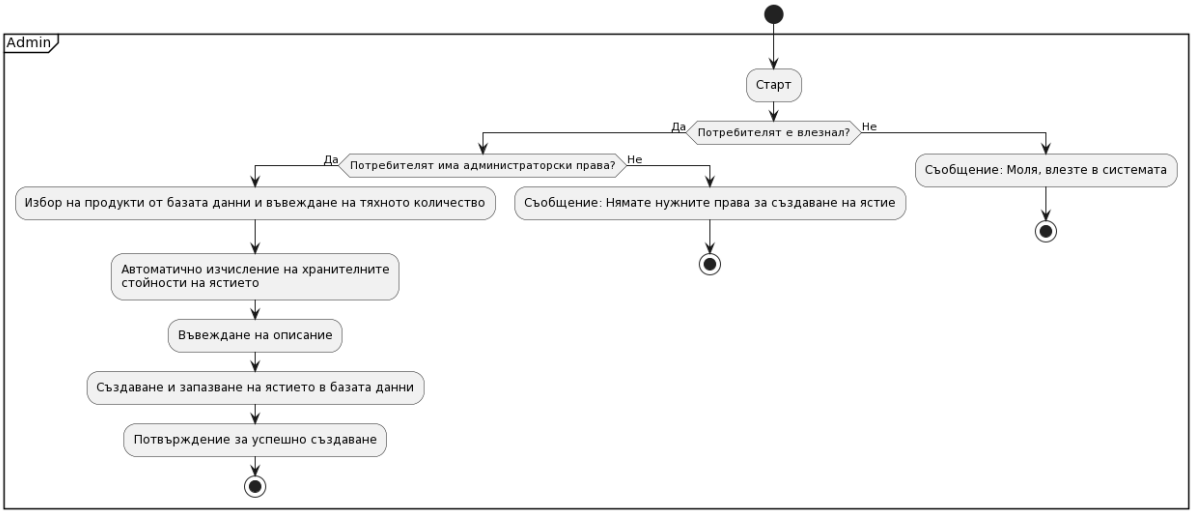
3.7. Генериране на план за хранене

Системата проверява дали потребителят е влезнал в системата. Системата проверява дали потребителят има зададени калорийни нужди в профила си. Ако има зададени калорийни нужди: Потребителят изпраща заявка за генериране на план за хранене. Системата изчислява необходимите храни на база зададените калории и създава план за хранене. Генерираният план за хранене се показва на потребителя и процесът приключва. Ако няма зададени калорийни нужди: Извежда се съобщение, че потребителят трябва да зададе своите калорийни нужди в профила си и процесът приключва. Ако потребителят не е влезнал: Извежда се съобщение, че потребителят трябва да влезе в системата и процесът приключва.



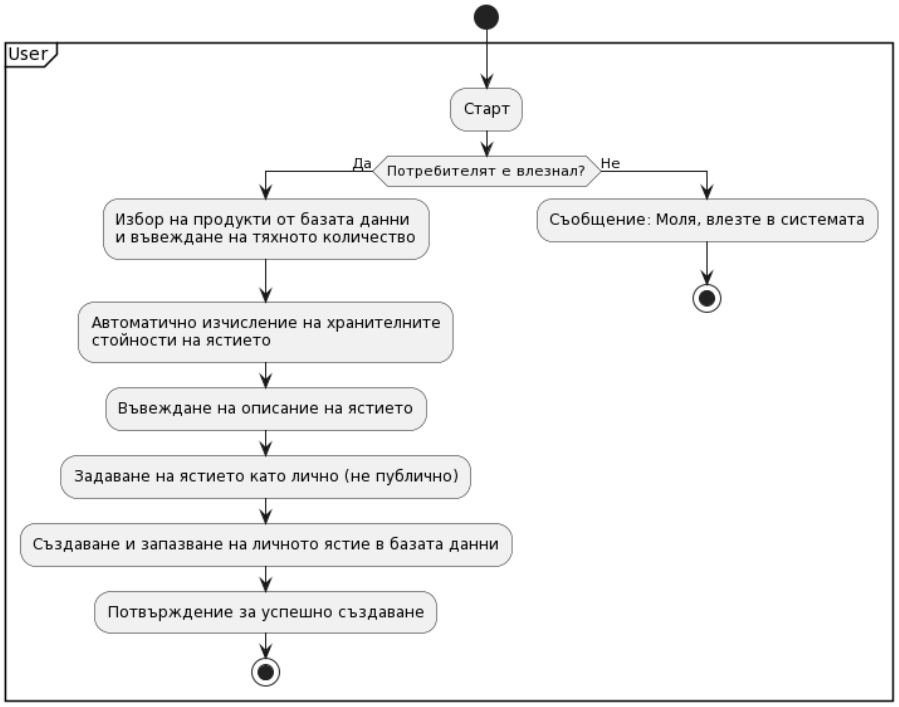
3.8. Създаване на ястие (рецепта) от администратор

Проверка дали потребителят е влезнал в системата. Ако НЕ, процесът се прекратява и се показва съобщение "Моля, влезте в системата". Проверка дали потребителят има правата на администратор. Ако НЕ, процесът се прекратява и се показва съобщение "Нямате нужните права за създаване на ястие". Администраторът избира продукти и определя тяхното количество, което ще бъде използвано в ястието. На базата на избраните продукти и тяхното количество се изчисляват общите хранителни стойности на ястието. Администраторът въвежда описание на ястието, което съдържа информация за него и начина на приготвяне. Ястието се създава и записва в базата данни. Администраторът получава съобщение за успешно създаване на ястието.



3.9. Създаване на лично (от потребител) ястие

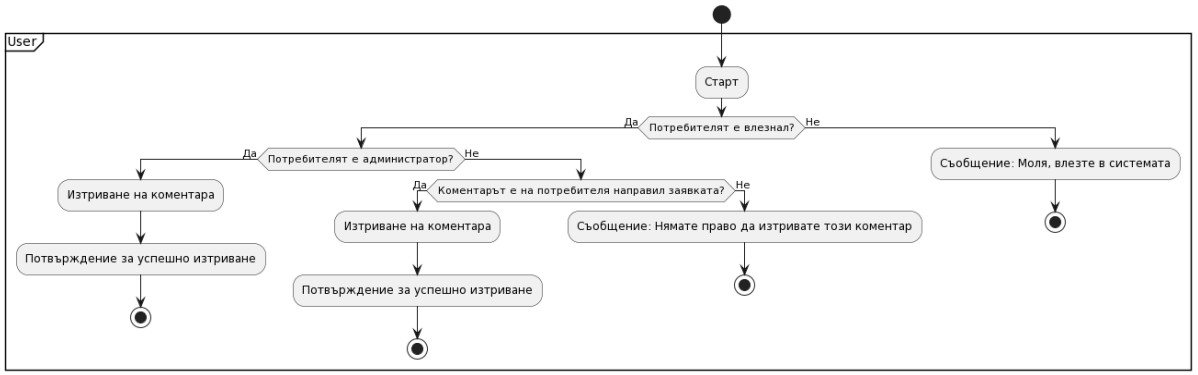
Проверка дали потребителят е влезнал в системата. Ако НЕ, процесът се прекратява и се показва съобщение "Моля, влезте в системата". Потребителят избира продукти и определя тяхното количество за ястието. На базата на избраните продукти и тяхното количество се изчисляват хранителните стойности на ястието. Потребителят въвежда описание за ястието. Потребителят определя, че ястието е само за лична употреба и не е публично. Ястието се създава и записва в базата данни като лично. Потребителят получава съобщение, че личното ястие е успешно създадено.



3.10. Изтриване на публикация от форума за седмичния конкурс по ястия

Проверка дали потребителят е влезнал в системата. Ако НЕ, процесът се прекратява и се показва съобщение "Моля, влезте в системата". Проверка дали потребителят има администраторски права. Ако ДА, потребителят изтрива

публикацията. След успешно изтриване, се показва съобщение за успешно изтриване. Ако НЕ, проверка дали публикацията, която потребителят иска да изтрие, е направена от него. Ако ДА, публикацията се изтрива и се показва съобщение за успешно изтриване. Ако НЕ, процесът се прекратява и се показва съобщение "Нямате право да изтривате тази публикация".



4. Изглед на данните

Описание на Модела на Данни за Форум за Хранене:

Моделът на данни представлява структурата на базата данни, използвана в приложението за форума за хранене. Той включва следните основни същности и връзките между тях:

Продукт (Product)
Id: Уникален идентификатор на продукта (целочислен).
Carbs: Съдържание на въглехидрати (десетичен тип).
Fat: Съдържание на мазнини (десетичен тип).
Protein: Съдържание на протеини (десетичен тип).

Рецепта (Recipe)
Id: Уникален идентификатор на рецептата (целочислен).
Description: Описание на рецептата (текстов тип).
DateOfCreation: Дата на създаване на рецептата (дата и време).
UserId: Идентификатор на потребителя, създал рецептата (външен ключ).
Products: Списък на продуктите, включени в рецептата (външен ключ към Product).
IsCustomRecipe: Показва дали рецептата е персонализирана (булев тип).

Потребител (User)
Id: Уникален идентификатор на потребителя (целочислен).
CustomRecipes: Списък на персонализираните рецепти, създадени от потребителя (външен ключ към Recipe).
Comments: Списък на коментарите, направени от потребителя (външен ключ към Comment).
Likes: Списък на харесванията, дадени от потребителя (външен ключ към Like).
CaloricIntake: Дневен калориен прием на потребителя (целочислен).
Age: Възраст на потребителя (целочислен).
Weight: Тегло на потребителя (десетичен тип).
Height: Височина на потребителя (десетичен тип).
SavedDayPlans: Списък на запазените дневни планове за хранене (външен ключ към DayPlan).

Коментар (Comment)
Id: Уникален идентификатор на коментара (целочислен).
Content: Съдържание на коментара (текстов тип).
UserId: Идентификатор на потребителя, направил коментара (външен ключ).
ParentCommentId: Идентификатор на родителския коментар, ако има такъв (външен ключ).

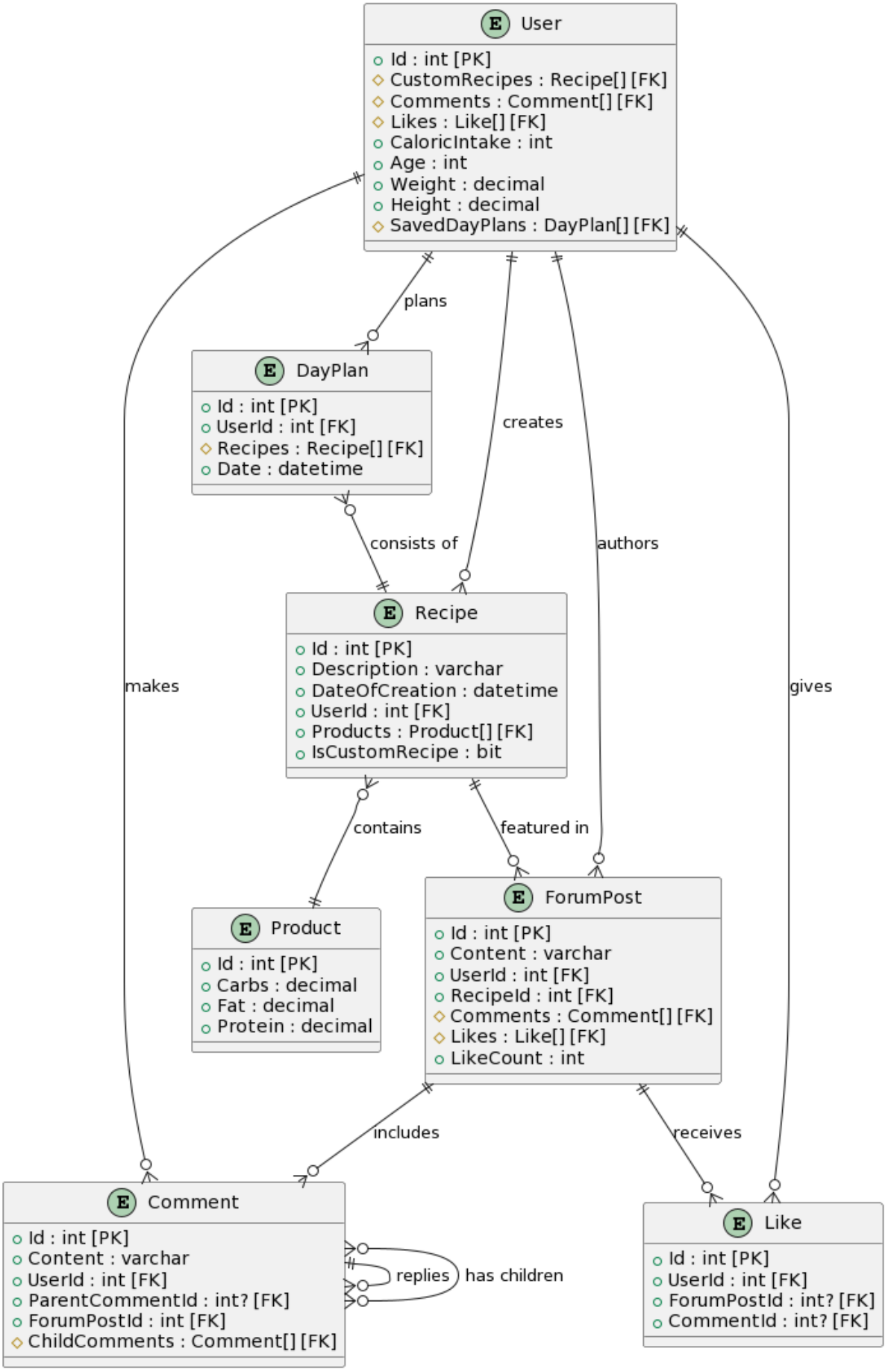
ForumPostId: Идентификатор на форум поста, към който принадлежи коментара (външен ключ).
ChildComments: Списък на дъщерни коментари (външен ключ).

Харесване (Like)
Id: Уникален идентификатор на харесването (целочислен).
UserId: Идентификатор на потребителя, дал харесването (външен ключ).
ForumPostId: Идентификатор на форум поста, ако харесването е за пост (външен ключ).
CommentId: Идентификатор на коментара, ако харесването е за коментар (външен ключ).

Форум Пост (ForumPost)
Id: Уникален идентификатор на форум поста (целочислен).
Content: Съдържание на поста (текстов тип).
UserId: Идентификатор на потребителя, автор на поста (външен ключ).
RecipeId: Идентификатор на рецептата, свързана с поста (външен ключ).
Comments: Списък на коментарите в поста (външен ключ).
Likes: Списък на харесванията на поста (външен ключ).
LikeCount: Брой на харесванията на поста (целочислен).

Дневен План (DayPlan)
Id: Уникален идентификатор на дневния план (целочислен).
UserId: Идентификатор на потребителя, създал плана (външен ключ).
Recipes: Списък на рецептите в плана (външен ключ към Recipe).
Date: Дата на плана (дата и време).

Връзките между същностите позволяват създаването на комплексна структура, която да отразява взаимоотношенията между продукти, рецепти, потребители, коментари и харесвания в рамките на форума.


User
Id : int [PK]
CustomRecipes : Recipe[] [FK]
Comments : Comment[] [FK]
Likes : Like[] [FK]
CaloricIntake : int
Age : int
Weight : decimal
Height : decimal
SavedDayPlans : DayPlan[] [FK]
plans
DayPlan
Id : int [PK]
UserId : int [FK]
Recipes : Recipe[] [FK]
Date : datetime
creates
consists of
authors
makes
gives
Recipe
Id : int [PK]
Description : varchar
DateOfCreation : datetime
UserId : int [FK]
Products : Product[] [FK]
IsCustomRecipe : bit
contains
featured in
Product
Id : int [PK]
Carbs : decimal
Fat : decimal
Protein : decimal
ForumPost
Id : int [PK]
Content : varchar
UserId : int [FK]
RecipeId : int [FK]
Comments : Comment[] [FK]
Likes : Like[] [FK]
LikeCount : int
includes
receives
Comment
Id : int [PK]
Content : varchar
UserId : int [FK]
ParentCommentId : int? [FK]
ForumPostId : int [FK]
ChildComments : Comment[] [FK]
replies
has children
Like
Id : int [PK]
UserId : int [FK]
ForumPostId : int? [FK]
CommentId : int? [FK]

5. Изглед на внедряването

Azure - Облачната платформа, която хоства уеб приложението и базата данни. В рамките на Azure са разгърнати следните компоненти:

Уеб Приложение (ASP.NET 6) - Основният софтуер, който служи като сървърна част на системата и обработва заявки от клиенти.

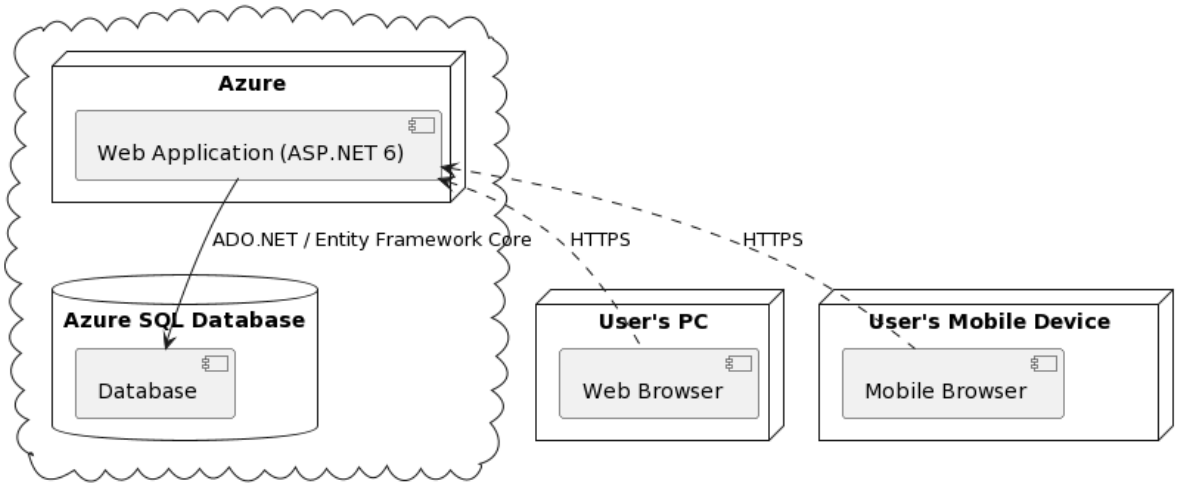
Azure SQL Database - Базата данни, която управлява всички данни, свързани с приложението, включително потребителска информация, данни за постове и др.

Потребителски Компютър - Устройство на потребителя, от което той достъпва уеб приложението чрез уеб браузър. Свързването с уеб приложението се извършва посредством HTTPS протокола за сигурна комуникация.

Мобилно Устройство на Потребителя - Мобилни устройства, които също достъпват уеб приложението чрез мобилен браузър, използвайки HTTPS за защитена връзка.

Комуникацията между Уеб Браузърите (и на десктоп, и на мобилно устройство) и Уеб Приложението се осъществява през Интернет с помощта на HTTPS протокол, който осигурява криптиране на данните и защита на информацията.

Уеб Приложението комуникира директно с Базата Данни чрез ADO.NET или Entity Framework Core, които предоставят интерфейс за управление на данните и изпълнение на различни операции като заявки, актуализации и други транзакции с данни.



6. Изглед на имплементацията

Изгледът на имплементацията показва структурата на слоевете в приложението и тяхното взаимодействие:

Презентационен слой (Presentation Layer) - Състои се от контролери, изгледи и модели за изгледи. Презентационният слой използва ASP.NET MVC за обработка на заявките на потребителите и представянето на уеб страници.

Бизнес логика (Business Logic Layer) - Включва сервизите RecipeService, ForumService, ProfileService и AdminService, които се занимават с обработката на бизнес правилата и операциите.

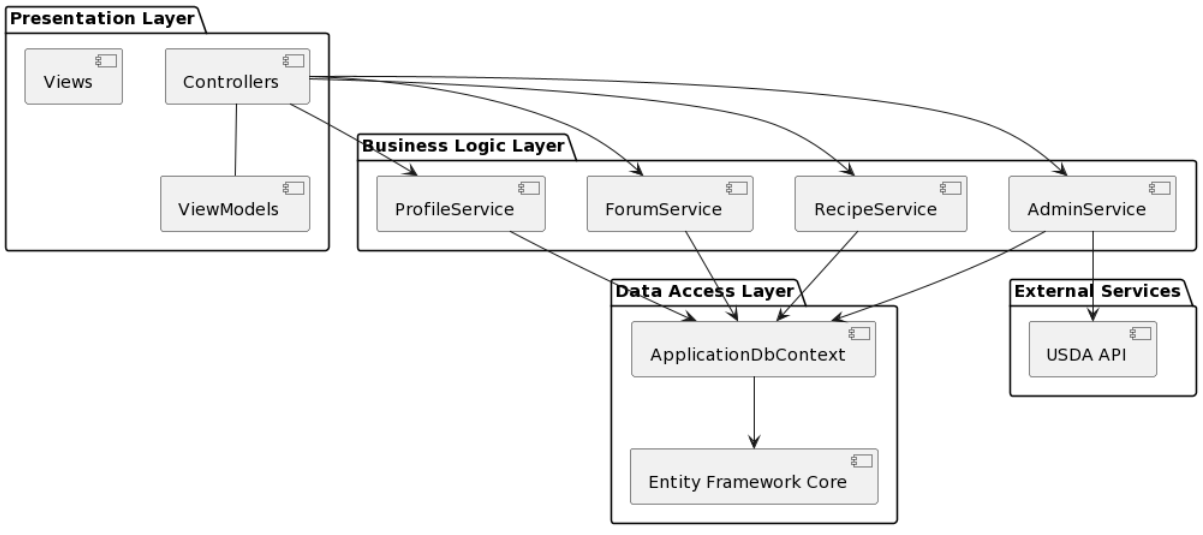
Слой на достъп до данни (Data Access Layer) - Определя се от ApplicationDbContext и използва Entity Framework Core за комуникация с базата данни.

Външни услуги (External Services) - Външният API на USDA се използва изключително от AdminService за извличане на информация за продуктите, което подчертава, че този слой има привилегии за достъп до определени външни ресурси.

Контролерите в презентационния слой взаимодействат директно с бизнес логиката. Сервизите в бизнес слоя управляват взаимодействието както със слоя на достъп до данни, така и с външни услуги когато е необходимо.

Приложението управлява идентификацията с помощта на ASP.NET Identity и OAuth 2.0 с вход чрез Google+. Авторизацията се контролира от бизнес слоя с помощта на атрибути.

Конфигурациите за връзка с базата данни се намират в настройките на Azure Web App, като и базата данни, и уеб приложението са разположени на платформата Azure.



Нефункционални изисквания

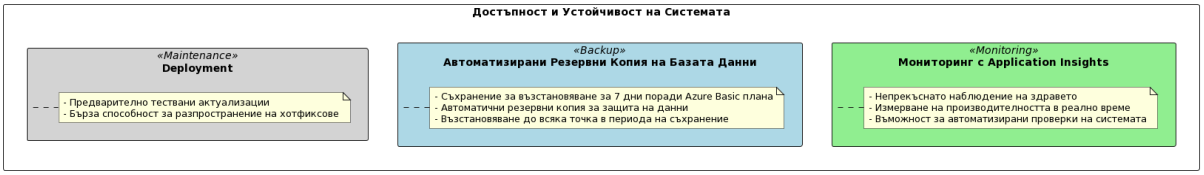
1. Достъпност

Системата ни използва набор от стратегии за увеличаване на своята устойчивост и осигуряване на постоянна работоспособност.

Мониторинг с Application Insights: Системата е оборудвана със средства за наблюдение, които предоставят постоянен мониторинг на нейното състояние. Това включва измерване на производителността в реално време и автоматизирани системни проверки за бързо откриване на потенциални проблеми.

Автоматизирани Резервни Копия на Базата Данни: Данните са защитени с автоматизирани резервни копия, които осигуряват възможност за възстановяване при необходимост. Съхраняването на данните се извършва в Azure и е настроено на Basic план, който предлага седемдневен период на ретенция.

Deployment: Поддържаме ефективен процес за разгръщане на нови версии на софтуера, като всеки нов релийз предварително се тества за да се минимизира времето на престой. В случай на критични проблеми, системата е способна да приема и разпространява хотфиксове с бързина и ефективност.



2. Разширяемост

Ще използваме MVC архитектурен шаблон, разделяйки функционалността на приложението на модули и компоненти, които могат да бъдат разширявани и подменяни лесно.

Ще се придържаме към добрите практики на програмиране, като например принципите на SOLID (Single Responsibility, Open/Closed, Liskov Substitution, Interface Segregation, Dependency Inversion) - за лесна поддръжка и разширение на кода.

Ще използваме интерфейси и абстракции за лесна подмяната на компонентите, без да се наруши функционалността на останалата част от приложението.

Ще поддържаме добра документация на кода, архитектурата и интерфейсите на приложението, за по-лесна поддръжка при бъдещи разширения.

Ще използваме Git - система за контрол на версиите, за да следим промените в кода и да съхраняваме историята на разработката.

3. Производителност

В момента нашето уеб приложение се хоства на Azure Free service tier, което подчертава стремежа към икономия на разходите по време на началния етап, когато все още няма активни потребители. Този план е адекватен за нашите текущи нужди, като предоставя основни услуги без допълнителни разходи.

При подготовката за привличане на потребители сме готови да преминем към по-мощни планове за услуги на Azure, които ще активират функции като:

Dynamic Scaling: Автоматично регулиране на ресурсите в отговор на промени в трафика.

Higher Service Tiers: Предоставяне на повече CPU и RAM за подобрение на времето за реакция при необходимост.

Data Replication и Caching Strategies: За подобряване на управлението на натоварванията и ускоряване на достъпа до данни с увеличаването на търсенето.

С помощта на Application Insights разполагаме с основен мониторинг, който ни позволява своевременно да откриваме и разрешаваме проблеми с производителността. Това подреждане ни осигурява да бъдем едновременно икономични и готови за мащабиране, готови да се развиваме заедно с изискванията на приложението.

4. Сигурност

За автентикация и авторизация ще използваме .NET Identity и OAuth 2.0 за вход в системата с Google акаунт или имейл и парола. Различните потребителски роли на приложението ще допринесат за ограничаването на достъпа до системата. Ще използваме принципа “най-малките привилегии”, при който потребителите имат само тези права, които са необходими за изпълнението на техните задачи и нищо повече. Този принцип е важен за защитата на данните и системите от злоупотреби и атаки. Ще поддържаме резервни копия на данните, за да гарантираме достъпността им при атаки или други инциденти. Ще използваме параметризирани заявки и ORM (ObjectRelational Mapping) слой, за да предотвратим SQL инжекции. Ще съхраняваме паролите на потребителите хеширани с добавете сол (salt) за допълнителна сигурност. Ще използваме механизми за предотвратяване на Cross-Site Request Forgery (CSRF) атаки, като например използването на CSRF токени.

5. Възможност за тестване

Според функционалностите на приложението тестовете ще представляват :

Ръчни тестове: те ще бъдат използвани за проверка на интерфейса на потребителското приложение и неговата употреба. Тестерите ще извършват ръчни тестове за вход и изход от потребителски профили, като симулират действието на реални потребители. Ще се тества функционалността за създаване на рецепти, като се

пробват различни входни данни и сценарии: Списък с налични продукти на потребителя . Списъка с продукти, като се добавят и изтриват елементи като се проверява коректността на данните.

За автоматизираните тестове ще използваме xUnit, защото е най-подходящо за нашето приложение. За тестване на увеличаването на хранителните планове ще се използват интегрирани тестове, които автоматично създават хранителен план и проверяват дали той отговаря на очакванията. Автоматично изчисляване на дневния калориен прием: Този вид тестове ще бъдат тествани с xUnit, които автоматично проверяват коректността на изчисленията в приложението. За тестване на функционалността за добавяне на рецепти в "Любими" ще се използват Интегрирани тестове, които автоматично добавят рецепта и проверяват дали тя е успешно добавена.

6. Интероперабилност

Системата ще е съвместима с външно API за извличане на рецепти и информация за продукта, като ще може да се адаптира към промени, като поддържа версии, като гарантира, че може да взаимодейства както с по-стари, така и с по-нови версии на API.

Софтуерът ще се интегрира с базата данни за съхраняване на потребителски предпочитания, планове за хранене и други подходящи данни.

Ще има механизми за поддържане на базата данни в синхрон с външни източници, като планирани актуализации на данни или синхронизация в реално време.

Приложението трябва редовно да се тества и актуализира, за да се осигури съвместимост с най-новите версии на външни системи, с които взаимодейства.

7. Използваемост

Започваме с провеждане на потребителски изследвания, които включват интервюта и анкети с потенциални потребители на приложението. Това ще ни помогне да разберем техните нужди, предпочитания и проблеми. Създаваме удобен и интуитивен потребителски интерфейс. Включваме лесна навигация, подредба на информацията и визуален дизайн, който е приятен за окото. Позволяваме на потребителите да се регистрират и влизат в приложението по лесен начин, например чрез Google акаунт или електронна поща. Възможност за създаване на лични профили, където потребителите могат да настройват предпочитания, цели и лични данни. Включване на обучение, което да помага на новите потребители да научат как да използват приложението. Използване на визуална йерархия, за да подчертаем важните елементи на страниците и функционалността, като например бутони за добавяне на храна или следене на калории. Потребителите предоставят обратна връзка и докладват за проблеми. Активно реагираме на техните коментари и подобрения. Извършване на тестове с реални потребители, за да оценим използваемостта на приложението и да идентифицираме проблеми в употребата. Поддръжка на различни устройства: приложението работи добре на екрани с различни размери и разделителни способности.