|  |  |
| --- | --- |
| A black and white logo  Description automatically generated | **ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – СΟФИЯ**  **ФАКУЛТЕТ КΟМПЮТЪРНИ СИСТЕМИ И ТЕХНΟЛΟГИИ** |

**ДИПЛΟМНА РАБΟТА**

тема:

Уеб-базирана платфοрма за наблюдение на хранителната и физическата активнοст.

Диплοмант: **Васил Бοянοв Петрοв**

Специалнοст: „Кοмпютърнο и сοфтуернο инженерствο“

Факултетен нοмер: 121221084

Научен ръкοвοдител: **д-р. инж. Ралица Райнοва**

**СЪДЪРЖАНИЕ**

**Въведение**

**Цел и задачи**

**Проучване на бизнес областта**

**Спецификация на изискванията**

**Концептуален дизайн на системата**

**ВЪВЕДЕНИЕ**

Избοрът на тема за настоящата дипломна работа е прοдиктуван преди всичко οт лични интереси и разбирания οтнοснο грижата за здраветο. Дοбрοтο здраве е тяснο свързанο с тοва, тялοтο на чοвек да функциοнира пълнοценнο и безпрοблемнο при различнο физическο натοварване. Прοследяванетο на хранителния прием е съществена стъпка в изгражданетο на здравοслοвен начин на живοт. Храната, кοятο οрганизмът приема, οказва влияние не самο върху външния вид, нο и върху вътрешнοтο състοяние и функциοнални възмοжнοсти на тялοтο. Физическата активнοст и хранителната балансираност, трябва да бъдат неразделна част οт нашето ежедневиетο, ако се стремим към здравословен живот. Когато има ясно дефинирана цел и последователност в действията, резултатите се постигат по-лесно и устойчиво. Редοвнοтο движение дοпринася за οбщата активнοст, пοдοбрява οбщοтο здравοслοвнο състοяние и вοди дο пοлοжитлени резултати в дългοсрοчен план.

В съвременния свят, здравοслοвния начин на живοт се явява тема с нарастващо значение и интерес. Съществуват множество фактοри, кοитο затрудняват постигането му. Например: липса на време или достатъчно мοтивация, неяснота относно начина и мястото на започване, както и непълна или неправилно интерпретирана информация. Изпοлзванетο на цифрови технологии за наблюдение на храненето и физическата активнοст значително увеличава верοятнοстта за пοстигане и поддържане на дългосрочни здравни цели. В този контекст възниква неοбхοдимοстта οт разработване на иновативно прилοжение, кοетο ефективно да интегрира функции за въвеждане, анализ и визуализация на данни, кактο и за генериране на персοнализирани препοръки, въз основа на индивидуалния напредък и цели. Приложението трябва да бъде леснο дοстъпно и интуитивно за използване. Трябва да е технологично надеждно и стабилно, като осигурява точност на данните. Да е подходящо както за ежедневна употреба, така и за дългосрочно планиране. Ключова характеристика би било да интегрира дневници за хранене и тренировки с отчитане на постигнатия напредък в единна среда, което ще улесни цялостния анализ на здравословното състояние на потребителя. Този подход ще позволи не само наблюдение в реално време, но и ще предоставя персонализирани предложения за оптимизиране на навиците и непрекъснато подобряване на физическата и хранителната форма.

**ЦЕЛ И ЗАДАЧИ**

Целта на настоящата дипломна работа е разработването на уеб базирано приложение: „**HealthBody&Mind**“, което да обслужва потребители, заинтересовани към подобряване на личното си здраве, като проследяват своята хранителна и физическа активност. Това включва както хора, които целят да редуцират или покачват мускулна маса, така и лица, които се стремят да поддържат добро физическо състояние и да изградят устойчиви здравословни навици. Системата е особено полезна за потребители, които нямат опит в планирането на хранене и не са запознати с концепцията за калории, как работят те и как да се възползват от тях. Приложението може да бъде използвано и от специалисти в съответната област – като фитнес инструктори или диетолози – за съвместна работа с техните клиентим като по този начин се улеснява проследяването на индивидуалния напредък. Приложението „**HealthBody&Mind**“ ще има за цел да осигури на своите потребители лесен, интуитивен и удобен начин за управление на здравословния им начин на живот. То е създадено с идеята, не само да се следят ежедневни навици, като хранене, физическа активност, прием на вода и стъпки, но също така потребителите да получават обратна връзка и насоки, чрез които постигането на личните цели, като редуциране или пοкачване на телесна маса, повишаване на физическата форма или подобряване на качеството на приетата храна, да бъдат по-лесно достижими. Основните задачи пред приложението са следните:

* Проучване на бизнес областта
* Спецификация на изискванията
* Концептуален дизайн на системата
* Избор на технологии
* Имплементация
* Определеяне на тестови сценарии и тестване

**ПРОЕКТИРАНЕ**

**Прοучване на бизнес οбластта**

Съществуват множество разнообразни приложения в областта, по-опростени, като калкулатори на калории, както и по-комплексни с AI поддръжка и интеграция с преносими устройства. Въпреки различията си, по-голямата част от тях споделят общи цели, като проследяване на консумираната храна, физическата активност и графична визуализация върху напредъка на потребителите.

**MyFitnessPal**

Едно от най-известните приложения в областта е приложението „**MyFitnessPal**“, което предлага богата база от храни и основни инструменти за следене на калории.

|  |  |
| --- | --- |
| Предимства | Недостатъци |
| Интуитивен интерфейс с възвможност за сканиране на баркод. | Началните калории, които приложението смята са неточни. |
| Голяма база от данни с храни - над 14 милиона записа. | Всеки потребител има възможност да въведе храна в базата от данни и това води до много неточни калории на храните. |
| Много различни видове единици за измервания (грамажи, милилитри и др.). | Някои от храните им липсват основни единици за измерване (грамажи и милилитри). |
| Интеграция с външни устройства. | Липса на персонализирани препоръки и AI чат асистент. |

*Таблица 1.1 – Предимства и недостатъци на приложението „****MyFitnessPal****“*

**Fitbit**

Друго популярно приложение в тази област е „**Fitbit**“. То предоставя възможност за проследяване на физическа активност, сън, хранене и други здравни показатели.

|  |  |
| --- | --- |
| Предимства | Недостатъци |
| Лесен за изпозлване, могат да се свързват приложенията на различни потребители. | Липсват известия за движение при продължително бездействие. |
| Позволява интеграция с други фитнес устройства. | Надценява активността и калориите, което поставя под въпрос точността на приложението. |
| Потребителите лесно могат да следят качеството на своя сън. | Не предлага конкретни съвети, а само показва данните на потребителя. |
| Физическата активност на потребителите е достъпна за проследяване от техните треньори. | Основният акцент е върху движението, докато хранителната активност е с по-нисък приоритет. |

*Таблица 1.2 – Предимства и недостатъци на приложението „****Fitbit****“[1]*

**Cronometer**

Друг пример в тази област е приложението „**Cronometer**“. То позволява детайлно проследяване на храненето чрез база данни с над 80 нутриента, както и въвеждането или синхронизацията на физическа активност чрез свързани усторйства и приложения.

|  |  |
| --- | --- |
| Предимства | Недостатъци |
| Позволява проследяване на до 84 нутриента, като витамини, минерали и макронутриенти. | Подробният анализ на нутриенти е полезен, но може да бъде натуварващ за начинаещи потребители. |
| Позволява синхронизация с други устройства за автоматичното добавяне на физическа активност. | Ограничения във функционалността на безплатната версия и необходимост от платен абонамент за пълен достъп. |
| Разполага с голяма потребителска общност, което улеснява обмена на опит и съвети. | Основният акцент е върху храненето, докато физическата активност е с по-нисък приоритет. |
| Поддържа проследяване на различни хранителни режими, като веган и кето. |  |

*Таблица 1.3 – Предимства и недостатъци на приложението „****Cronometer****“[2]*

Системата „**HealthBody&Mind**“ ще адресира всички тези недостатъци, като началните калории на потребителите ще се изчисляват по формулата на Харис-Бенедикт, включваща пол, тегло, ръст и възраст, формулата е следната:

• За мъже – 66 + (6.23 \* тегло в паундове) + (12.7 \* ръст в инчове) – (6.8 \* години)

• За жени – 655 + (4.35 \* тегло в паундове) + (4.7 \* ръст в инчове) – (4.7 \* години)[3]

Всички храни и напитки, които ще са в базата от данни ще бъдат потвърдени и с коректно въведени калории и макронутриенти, като ще имат задължително основните единици за измерване (грамажи и милилитри). Така ще се избегнат неточности, които са характерни за потребителските добавени храни. Приложението ще предоставя персонализирани препоръки, които са свързани с данните въведени от потребителя, както и възможност за комункация с AI чат асистент, който трябва да отговаря в реално време на въпроси, които са свързани по теми в областта. Всички основни функционалности ще бъдат достъпни напълно безплатно, без необходимост от абонаментни планове, като така всеки един потребител ще може да се наслади цялостно на приложението. „**HealthBody&Mind**“ ще има за цел да поддържа равностоен фокус върху хранителната и физичската активност. Потребителите ще имат възможността да следят както диетите си, така и различни видове тренировки (кардио и силови), включително и изгорените калории. Системата ще изпраща известия и напомняния за недостигнати дневни цели, за да мотивира потребителя те да бъдат изпълнени. Интерфейсът ще бъде интуитивен и лесен за ползване - няма да претоварва клиентите с излишна информация - в рамките само на един клик от основанта страница, ще бъде предоставен достъп до цялата функционалност на приложението.

**Спецификация на изискванията**

**Бизнес изисквания**

Бизнес изискванията са насочени към създаването на устойчива основа за стратегически партньорства и бъдеща монетизация. Продуктът трябва да позволява интеграция с търговци на здравословни добавки и услуги, както и с фитнес и здравни институции. Чрез адаптивна архитектура, системата цели да се представи като платформа, която е способна да обединява различни заинтересовани страни и да създава стойност чрез разширяване на функционалността и бизнес мрежата. Бизнес изискванията включват:

1. Системата трябва да поддържа възможност за интеграция с външни доставчици на хранителни добавки, витамини и протеинови продукти, което ще позволи създаване на стратегически бизнес партньорства и нови клиенти за доходи чрез директни продажби.
2. Системата трябва да бъде съвместима с бъдеща интеграция с фитнес зали, спортни инструктори или клиники, с цел предоставяне на персонализирани препоръки на база медицински или тренировъчни данни и сключване на партньорства за предоставяне на комбинирани услуги.
3. Системата трябва да бъде проектиране с възможност за разширяване, така че в бъдеще да поддържа нови функционални модули (например онлайн консултации, интелигентни преносими устройства и т.н.), които да разширят обхвата на услугите и да привлекат различни категории клиенти и партньорства.

Бизнес изискванията на системата целят изграждане на устойчива основа за бъдещо развитие, чрез възможност за партньорства, монетизация и пазарно позициониране. Системата е замислена не само като функционален продукт, но и като стратегическа платформа с реален бизнес потенциал.

**Потребителски изисквания**

Потребителските изисквания за системата „**HealthBody&Mind**“ имат за цел да осигурят лесен достъп до лични здравни данни, възможност за задаване и проследяване на цели, както и получаване на персонализирани съвети. Фокусът е върху интуитивно взаимодействие, висока степен на персонализация и поддръжка на полезни функционалности, които отговарят на индивидуалните нужди на потребителя. Те включват:

1. Потребителят трявба да има възможност да създаде свой личен профил, като това включва въвеждане на основни данни и здравни цели. Профилът следва да бъде редактируем, като се позволява промяна на тегло, ръст, активност и целеви стойности.
2. Потребителят трявба да има възможност лесно да въвежда приетата храна, като това става чрез търсене в предварително дефинирана база от данни. Системата следва да показва калорийното съдържание и макронутриентите на съответната храна, както и да я групира по категории (закуса, обяд, вечеря).
3. Потребителят трябва да може да добавя физичекса активност с информация за тип (кардио, силова), интензивност и продължителност. Системата трябва да изчислява изразходените калории и да ги визуализира на основанта страница.
4. Потребителят трябва да може да задава цели за калории, чрез трите основни макронутриента (протеин, въглехидрати и мазнини), прием на вода и брой стъпки. Системата трябва да предоставя прогрес барове относно изпълнението на тези цели.
5. Потребителят трябва да има въможност да достъпва всички функции на системата чрез интуитивен интерфейс, който позволява изпълнението им директно от основната страница.
6. Потребителят трябва да разполага с възможност за преглед на текущия прогрес, катко и анализ на резултатите за определен период от време (последните 7 или 30 дни), като това включва графично представяне на приетите и изразходени калории, както и съотношението на основните макронутриенти.
7. На потребителя трябва да му се изпраща напомняне относно дненвите цели, които си е задал, ако деня е към края си, а той все още не ги е постигнал, както и известие за това, че е успял да достигне някоя от своите дневни цели. Системата трябва да предлага персонализирани препоръки въз основа на въведените данни и зададените цели.
8. Потребителят трябва да има възможност за комуникация с AI чат асистент, който да предоставя отговори на въпроси, свързани с хранене, тренировки, диети, напредък и всичко свързано в областта.

Тези потребителски изисквания са в основата на изграждането на логиката на приложението и гарантират, че системата ще бъде насочена към крайните потребители и техните нужди.

**Системни изисквания**

**Функционални изисквания**

Функционалните изисквания на системата „**HealthBody&Mind**“ обхващат основните действия и възможности, които приложението трябва да осигурява на потребителя. Те включват:

1. Системата трябва да позволява на нерегистрирани потребители да създават акаунт чрез регистрационна форма с валидирани задължитлени полета.
2. Системата трябва да предоставя възможност на регистрирани потребители да управляват своя профил.
3. Системата трябва да предоставя възможност за добавяне на консумирана храна на регистрирани потребители.
4. Системата трябва да предоставя опция за проследяване на физическата активност на регистрирани потребители.
5. Системата трябва да визуализира напредъка на регистрирани потребители, посредством графики и ленти за прогрес.
6. Системата трябва да предлага персонализирани препоръки и напомняния, посредством известия на регистрирани потребители.
7. Системата трябва да предлага AI чат асистент, с който регистрирани потребителите да имат възможност да взаимодействат.
8. Системата трябва да разполага с интуитивен интерфейс, чрез който регистрирани потребители ще могат лесно да навигират из него.

**Нефункционални изисквания**

Нефункционалните изисквания на системата „**HealthBody&Mind**“ определят качествените характеристики, които системата трябва да притежава, за да бъде сигурна, ефективна, мащабируема и удобна за използване. Те включват:

1. Системата трябва да покрива GDPR (General Data Protection Regulation) стандарта за личните данни.
2. Системата трябва да осигурява механизъм за контрол на достъпа до различни функционалности, осъществява се чрез JWT (Json Web Token).
3. Системата трябва да функционира коректно при наличието на надеждна интернет връзка, с минимална скорост от 100 Mbps (megabits per second), която осигурява безпроблемен достъп до всчики функции.
4. Системата трявба да бъде изградена на принципа на многослойната архитектура, това включва отделни модули за фронт-енд, бек-енд и база данни, с цел по-добра мащабируемост, поддръжка и разделение на отговорностите.
5. Системата трябва да изпълнява основните функции – като добавяне на храна и тренировки - в рамките на до 2 секунди.
6. Системата трябва да работи коректно, с еднаква функционалност и визуално представяне, на различни уеб браузъри (Google Chrome, Mozilla Firefox).
7. Системата трябва да предоставя лесен за използване интерфейс - с ясно обузначени бутони, икони, чрез етикети и подсказски - който да улеснява навигацията и достъпа до функциите на системата.
8. Системата трябва да позволява достъп до всяка основна функционалност (преглед на профила и статистики, добавяне на данни, комуникация с AI чат асистент) с не повече от един клик от основната страница.

**Потребителски истории и критери за приемане**

В системата „**HealthBody&Mind**“ потребителските истории служат като основа на функционалността от гледна точка на реалните нужди и очаквания на крайния потребител. За всяка потребителска история са дефинирани ясни критерии за приемане, които гарантират, че реализацията отговаря както на функционалните изисквания, така и на очакваната история. Потребителските истории и критериите за приемане, включват (номера на потребителската история показва от кое функционално изискване е произлязла):

|  |  |
| --- | --- |
| **№** | **Потребителска история:** |
| 1.1 | Като нерегистриран потребител, искам да мога да се регистрирам с данни като имейл, парола, пол, ръст, тегло и цели (целево тегло, ниво на активност), за да създам свой профил. |
|  | **Критерии за приемане:** |
| 1. | При регистрация потребителят да има възможност за въвеждане на основни данни, като имейл, парола, пол, ръст, тегло и цели. |
| 2. | Полетата за имейл и парола са задължителни при регистрация и трябва да бъдат валидирани преди да се продължи напред, имейла трябва да отговаря на изискването за имейл – да съдържа „@“ – дължината да е максимум 255 символа, да започва с малко буква, паролата да е с дължина от минимум 8 символа. |
| 3. | При успешно попълване на формата за регистрация, системата трябва да създаде нов потребителски акаунт. |
| 4. | Ако въведения имейл е вече регистриран, системата трябва да покаже съобщение за грешка. |

*Таблица 2.1 – Информация относно потребителска история с № 1.1.*

|  |  |
| --- | --- |
| **№** | **Потребителска история:** |
| 1.2 | Като регистриран потребител, искам да мога да влизам в системата с имейл и парола, както и да възстановявам паролата си чрез имейл, за да имам постоянен и сигурен достъп до профила си. |
|  | **Критерии за приемане:** |
| 1. | При опит за влизане в приложението потребителят да има възможност за въвеждане на своя имейл и парола. |
| 2. | Ако потребителят въведе грешни данни, системата трябва да изведе необходимо съобщение, подканяйки го да поправи своите данни. |
| 3. | Потребителят трявба да има възможност за възстановяване на своята парола, чрез имейла си. |

*Таблица 2.2 – Информация относно потребителска история с № 1.2.*

|  |  |
| --- | --- |
| **№** | **Потребителска история:** |
| 2.1 | Като регистриран потребител, искам да мога да редактирам данните си, включително тегло, ниво на активност и хранителни цели, за да поддържам профила си актуален спрямо здравослновните си цели. |
|  | **Критерии за приемане:** |
| 1. | Потребителят трябва да може да редактира своя профил, като актуализира своите данни. |
| 2. | Полетата трябва да показват текущите стойности и да позволяват редакция. |
| 3. | При успешно актуализиране на профила, системата трябва да запазва промените. |
| 4. | Системата не трябва да позволява въвеждането на невалидни данни – число в текстово поле или обратното. |

*Таблица 2.3 – Информация относно потребителска история с № 2.1.*

|  |  |
| --- | --- |
| **№** | **Потребителска история:** |
| 2.2 | Като регистриран потребител, искам да мога да задавам персонализирани здравословни цели, за да следя напредъка си и да получвам препоръки, съобразени с тях. |
|  | **Критерии за приемане:** |
| 1. | Потребителят трябва да може да настрои своите персонализирани цели за дневен прием на калории чрез основните макронутриенти (протеин, въглехидрати и мазнини). |
| 2. | Системата трябва да включва опция за въвеждане на стойност за целево тегло. |
| 3. | След запазване на промените, системата трябва да актуализира прогрес баровете на основната страница според новите цели. |

*Таблица 2.4 – Информация относно потребителска история с № 2.2.*

|  |  |
| --- | --- |
| **№** | **Потребителска история:** |
| 3.1 | Като регистриран потребител, искам да мога да добавям консумираните храни с данни за калории и макронутриенти, за да следя дневния си хранителен прием. |
|  | **Критерии за приемане:** |
| 1. | Потребителят трябва да има възможност за добавяне на храна, да може да въведе име на храната, количество, за да излезе информация за калориите и макронутриентите ѝ. |
| 2. | При успешно добавяне на храна, системата трябва да актуализира дневните стойности за калории и макронутриенти. |
| 3. | След запазване на промените, системата трябва да актуализира прогрес баровете на основната страница според новите цели. |
| 4. | Системата не трябва да позволява въвеждането на невалидни данни – число в текстово поле или обратното. |

*Таблица 2.5 – Информация относно потребителска история с № 3.1.*

|  |  |
| --- | --- |
| **№** | **Потребителска история:** |
| 3.2 | Като регистриран потребител, искам да мога да търся храни в базата данни, за да намирам и добавям това, което съм консумирал. |
|  | **Критерии за приемане:** |
| 1. | Потребителят трябва да може да въвежда текст в поле за търсене на храни. |
| 2. | Системата трябва да показва списък с резултати от базата данни с храни, които съответстват на въведения текст. |
| 3. | Потребителят трявба да може да избере храна от списъка с резултати. |
| 4. | Ако няма намерени съвпадения, системата трябва да покаже подходящо съобщение за липсата на такива храни. |

*Таблица 2.6 – Информация относно потребителска история с № 3.2.*

|  |  |
| --- | --- |
| **№** | **Потребителска история:** |
| 3.3 | Като регистриран потребител, искам добавените храни да могат да се организират по категории (закуска, обяд, вечеря и други), за да следя по-лесно дневния си хранителен режим. |
|  | **Критерии за приемане:** |
| 1. | Потребителят трябва да може да избере категория (закуска, обяд, вечеря и други) при добавяне на храна. |
| 2. | Системата трябва да групира добавените храни по съответните категории и да ги показва в хранителния дневник. |
| 3. | При преглед на хранителния дневник, потребителят трябва да може лесно да види общите калории и макронутринети за всяка категория, както и общите стойности за деня. |
| 4. | Потребителят трябва да може да редактира категорията на добавена храна, ако е необходимо, както и да изтрива записи от нея. |

*Таблица 2.7 – Информация относно потребителска история с № 3.3.*

|  |  |
| --- | --- |
| **№** | **Потребителска история:** |
| 3.4 | Като регистриран потребител, искам да мога да преглеждам текущия си дневен енергиен и хранителен прием спрямо поставените цели, за да следя прогреса си. |
|  | **Критерии за приемане:** |
| 1. | Потребителят трябва да може да вижда актуализирани стойности за дневния прием на калории и основните макронутриенти спрямо зададените цели. |
| 2. | Системата трябва да показва прогрес барове за всяка макронутриентна група и общия калориен прием. |
| 3. | При премахване или добавяне на храна от дневника, стойностите на прогрес баровете трябва да се актуализират автоматично. |
| 4. | Ако потребителят изпълни дневните си целеви стойности, системата трябва да изпрати съобщение за това, чрез известие. |

*Таблица 2.8 – Информация относно потребителска история с № 3.4.*

|  |  |
| --- | --- |
| **№** | **Потребителска история:** |
| 4.1 | Като регистриран потребител, искам да мога да вписвам своите физически активности, като кардио и силови тренировки, за да следя продължителността им и изразходените си калории. |
|  | **Критерии за приемане:** |
| 1. | Потребителят трябва да има достъп до форма за добавяне на физическа активност, където може да въведе тип на активността, продължителност и интензивност. |
| 2. | При успешно добавяне на активност, системата трябва да изчисли изразходените калории въз основа на въведените данни и да актуализира дневните стойности за изразходени калории. |
| 3. | Системата не трябва да позволява въвеждането на невалидни данни – число в текстово поле или обратното. |

*Таблица 2.9 – Информация относно потребителска история с № 4.1.*

|  |  |
| --- | --- |
| **№** | **Потребителска история:** |
| 4.2 | Като регистриран потребител, искам системата автоматично да изчислява изгорените калории при тренировка въз ознова на моето тегло, интензивност и МЕТ (metabolic equivalent of task) стойност, за да получавам възможно най-точна информация. |
|  | **Критерии за приемане:** |
| 1. | При въвеждане на информация за продължителност и интензивност на тренировката, системата трябва автоматично да изчисли изразходените калории въз основа на MET стойност и теглото на потребителя. |
| 2. | Калориите трябва да бъдат показани в реално време във формата за добавяне на активност. |
| 3. | Потребителят трябва да може да прегледа изразходените калории за всяка добавена активност в дневника си. |
| 4. | Ако потребителят промени теглото си в профила, новите записи на активността трябва да отчитат актуализираното тегло при изчисляване на калориите. |

*Таблица 2.10 – Информация относно потребителска история с № 4.2.*

|  |  |
| --- | --- |
| **№** | **Потребителска история:** |
| 4.3 | Като регистриран потребител, искам да имам достъп до историята на своите тренировки, за да мога да преглеждам всички въведени физически активности за избран период. |
|  | **Критерии за приемане:** |
| 1. | Потребителят трябва да има достъп за проследяване на своите тренировки, където може да види всички записани активности за деня. |
| 2. | Системата трябва да групира активностите по дата и да показва общите изразходени калории за всяка от тях. |
| 3. | Потребителят трявба да може да преглежда исторически записи на тренировки за предишни дни. |

*Таблица 2.11 – Информация относно потребителска история с № 4.3.*

|  |  |
| --- | --- |
| **№** | **Потребителска история:** |
| 5.1 | Като регистриран потребител, искам да виждам графично представяне на дневния и седмичния си прогрес, за да проследявам приетите и изразходените калории, както и съотношението на основните макронутриенти. |
|  | **Критерии за приемане:** |
| 1. | Потребителят трябва да може да вижда графики, които визуализират промените в калорийния прием, изразходените калории и съотношението на основните макронутриенти за избран период (последните 7 или 30 дни). |
| 2. | Системата трябва да актуализира графиките автоматично при промяна на данните. |
| 3. | Потребителят трявба да има опция за превключване между различни типове графики. |
| 4. | Графиките трябва да показват данните по начин, който позволява ясно разграничаване във времето, по оста „y“ да се показват данните, по оста „x“ - датите. |

*Таблица 2.12 – Информация относно потребителска история с № 5.1.*

|  |  |
| --- | --- |
| **№** | **Потребителска история:** |
| 5.2 | Като регистриран потребител, искам да избирам период от време (последните 7 или 30 дни), за да виждам анализа на своя прогрес за съответния интервал. |
|  | **Критерии за приемане:** |
| 1. | Потребителят трябва да може да избира времеви период за визуализация на прогреса си в графиките. |
| 2. | При избор на нов времеви период, системата трябва автоматично да обновява съответната графика. |
| 3. | Потребителят трябва да вижда ясни времеви маркери - дати по оста „x“ и стойности по оста „y“ - в графиките, за да разбере в кой ден са отчетени конкретните стойности. |

*Таблица 2.13 – Информация относно потребителска история с № 5.2.*

|  |  |
| --- | --- |
| **№** | **Потребителска история:** |
| 6.1 | Като регистриран потребител, искам системата да ми предоставя персонализирани съвети основавайки се на въведените данни, за да постигам по-лесно своите цели. |
|  | **Критерии за приемане:** |
| 1. | Системата трябва да генерира персонализирани препоръки въз основа на данните на потребителя. |
| 2. | Препоръките трябва да се показват на основната страница на приложението. |
| 3. | Препоръките трябва да бъдат дневни и да бъдат такива, които да отговарят на основната цел на потребителя (покачване, редуциране и поддържане на килограми). |

*Таблица 2.14 – Информация относно потребителска история с № 6.1.*

|  |  |
| --- | --- |
| **№** | **Потребителска история:** |
| 6.2 | Като регистриран потребител, искам да получавам напомняния под формата на известия, за да не пропускам изпълнението на дневните си цели. |
|  | **Критерии за приемане:** |
| 1. | Системата трябва да изпраща напомняния на потребителя за дневните му цели. |
| 2. | Напомнянията трябва да се показват под формата на известия в приложението. |

*Таблица 2.15 – Информация относно потребителска история с № 6.2.*

|  |  |
| --- | --- |
| **№** | **Потребителска история:** |
| 6.3 | Като регистриран потребител, искам да задавам дневни цели за прием на вода и брой стъпки, за да ги следя и да се визуализират диркетно в приложнието. |
|  | **Критерии за приемане:** |
| 1. | Потребителят трябва да има възможност да задава дневни цели за прием на вода и брой стъпки. |
| 2. | Системата трябва да показва текущия прогрес спрямо тези цели. |
| 3. | Потребителят трябва да може да актуализира целите си за вода и стъпки по всяко време, като промените автоматично трябва да се отразяват. |
| 4. | Ако потребителят достигне зададените цели за вода или стъпки, системата трябва да показва съобщение за успешно изпълнение, под формата на известие. |

*Таблица 2.16 – Информация относно потребителска история с № 6.3.*

|  |  |
| --- | --- |
| **№** | **Потребителска история:** |
| 7.1 | Като регистриран потребител, искам да имам достъп до AI чат асистент, за да получавам отговори на въпроси, свързани с хранене, тренировки и здравословния начин на живот. |
|  | **Критерии за приемане:** |
| 1. | Потребителят трябва да има достъп до икона за AI чат асистент в долния десен ъгъл на основната страница, като при натискане се отваря прозорец за чат с AI чат асистента. |
| 2. | Прозорецът за чат трябва да поддържа въвеждане на въпроси относно прогреса, храненето и тренировките. |
| 3. | AI чат асистента трябва да отговаря на въпросите в реално време, като предоставя конкретни отговори, които да бъдат уместни със зададения въпрос. |

*Таблица 2.17 – Информация относно потребителска история с № 7.1.*

|  |  |
| --- | --- |
| **№** | **Потребителска история:** |
| 7.2 | Като регистриран потребител, искам AI чат асистентът да разпознава и отговаря на предварително дефинирани въпроси, за да получавам бърза и точна информация по теми, свързани с приложението. |
|  | **Критерии за приемане:** |
| 1. | Системата трябва да изпозлва предварително конфигуриран „Dialoflow“ агент, който съдържа интенти, съответстващи на конкретни теми от приложението – като хранене, тренировки, хидратация и други. |
| 2. | Когато потребителя въведе въпрос в чата, системата трябва да го изпрати към „Dialogflow“ и той да го свърже с подходящ интент. |
| 3. | Ако въпросът съвпада с някой от дефинираните интенти, AI чат асистентът трябва да върне един от предварително зададените отговори, свързани с този интент. |
| 4. | В случай, че въпроса не съвпадне с нито един от съществуващите интенти, системата трябва да върне съобщение, което е предварително дефинирано във „fallback“ отговор. |

*Таблица 2.18 – Информация относно потребителска история с № 7.2.*

|  |  |
| --- | --- |
| **№** | **Потребителска история:** |
| 8.1 | Като регистриран потребител, искам на основната страница да виждам прогрес барове за дневните си цели, които да се актуализират според въведените данни, за да следя напредъка си в реално време. |
|  | **Критерии за приемане:** |
| 1. | Основната страница трябва да показва прогрес барове за дневните цели на потребителя (калории, вода и макронутриенти), които отразяват текущия напредък. |
| 2. | Прогрес баровете трябва да се актуализират в реално време при добавяне на храна или вода. |
| 3. | Ако потребителят достигне или надвиши дадена цел трябва да се изведе нужното съобщение, под формата на известие. |
| 4. | Прогрес баровете трябва да бъдат разположени на видно място на основната страница, без необходимост от допълнително търсене. |

*Таблица 2.19 – Информация относно потребителска история с № 8.1.*

|  |  |
| --- | --- |
| **№** | **Потребителска история:** |
| 8.2 | Като регистриран потребител, искам да разполагам с бутони за бързо добавяне на храна, активност и други основни действия, за да използвам функционалностите на приложението по-удобно и ефективно. |
|  | **Критерии за приемане:** |
| 1. | Интерфейсът трябва да включва бутони за добавяне на храна, активност и вода, които са лесно достъпни и видими за потребителя. |
| 2. | При натискане на бутон той трябва да води директно към съответната форма за въвеждане, като позволява на потребителя да добави информацията. |
| 3. | Бутоните трябва да бъдат ясно маркирани – с подсказващ текст - и интуитивно разположени, за да улеснят употребата им за потребителя. |

*Таблица 2.20 – Информация относно потребителска история с № 8.2.*

|  |  |
| --- | --- |
| **№** | **Потребителска история:** |
| 8.3 | Като регистриран потребител, искам да мога бързо и лесно да навигирам между различните страници на приложението чрез ясно разположени бутони, за да използвам всички функции без затруднение. |
|  | **Критерии за приемане:** |
| 1. | Приложението трябва да има навигационна лента с бутони за основните страници („Home page“, „Food page“, „Exercise page“ и „Charts page“). |
| 2. | Потребителят трябва да може бързо и без забавяне - в рамките на до 2 секунди - да преминава между различните страници чрез натискане на съответния бутон. |
| 3. | Навигационната лента трябва да остане фиксирана на екрана, за да бъде достъпна независимо от позицията на потребителя в приложенито. |

*Таблица 2.21 – Информация относно потребителска история с № 8.3.*

**Концептуален дизайн на системата**

Концептуалният дизайн в рамките на разработката на системата „**HealthBody&Mind**“ представлява основополагаща фаза, в която се изгражда цялостната логическа структура и поведение на системата, преди нейната реална реализация. Основните аспекти на концептуалния дизайн включват:

* Архитектура
* Концептуален дизайн на базата данни

**Архитектура**

Избраната архитектура се основава на клиент-сървър модела, многослойното структуриране и модулното разделение на логиката, за да се гарантира ясното разграничаване на отговорностите, сигурността на данните и висока производителност. Тази организация позволява системата ефективно да отговаря на нуждите на крайните потребители за бърза, сигурна и интуитивна работа, както и на бизнес целите за възможност за интеграция и бъдещо развитие.

**Архитектурни диаграми**

**Диаграма на внедряване**

|  |
| --- |
|  |

*Фигура 1.1 – Диаграма на внедряване*

Потребителят взаимодейства с клиентското приложение – реализирано чрез „Flutter“ - което комуникира с бизнес сървър – реализиран чрез „Spring Boot“ - посредством защитена „HTTPS“ връзка отговаряща на порт „443“. Бизнес сървърът обработва всички потребителски заявки, осъществява валидация на данните и управлява достъпа до базата от данни, която е разположена на самостоятелен сървър, достъпен през порт „3306“. Освен връзката с вътрешните компоненти, бизнес сървърът осъществява интеграция с външна услуга, API, за поддръжка на функционалността на AI чат асистент. Това архитектурно решение гарантира висока степен на сигурност, мащабируемост и възможност за независимо развитие на отделните части на системата.

**Диаграма на компонентите**

|  |
| --- |
|  |

*Фигура 1.2 – Диаграма на компонентите*

Компонентната диаграма на системата представя логическите блокове и техните взаимовръзки. Потребителският интерфейс – реализиран чрез „Flutter“ - осъществява комуникация със сървърната част на приложението чрез изпращане на „HTTP“ заявки през стандартен мрежов протокол. Сървърната страна – изградена върху „Spring Boot“ - съдържа отделни компоненти за управлението на потребители, хранителни данни, физическа активност, визуализация на прогреса и генериране на препоръки, заедно с функционалност за AI чат асистент. Всеки от тези компоненти има ясно дефинирана роля, а взаимодействието между тях се осъществява посредством стандартизирани интерфейси, които гарантират последователност и разширяемост на системата. Централизираната база данни – реализирана чрез „MySQL“ - служи за управление на информацията, свързана с потребителите, хранителния прием, физическата активност и генерираните препоръки.

**Архитектурен стил**

Системата „**HealthBody&Mind**“ е изградена върху архитектурния стил клиент-сървър. Потребитеслкия интерфейс действа като клиент, който изпраща „HTTP“ заявки към сървърната част. Сървърът обработва заявките, валидира и управлява данните и осигурява връзка с базата данни, както и с външната услуга за AI чат асистент. Изборът на клиент-сървър архитектура се основава на следното:

* Ясно разделение на отговорностите между визуализацията и обработката на данни.
* Централизирана сигурност и контрол на данните (автентикация с JWT токени).
* Възможност за лесно надграждане на сървъра или базата дании без прекъсване на клиентската страна.

Този архитектурен избор гарантира, че системата ще може да изпълни бизнес изискванията и потребителските такива за стабилна, сигурна и лесна за изпозлване система[4]. В допълнение към системата е приложен моделът за техническото разделяне, който осигурява ясно разделение на отговорностите между различните компоненти на приложението.

|  |
| --- |
|  |

*Фигура 1.3 – Диаграма на техническото разделяне[5]*

„Presentation“слоят отговаря за взаимодействието с потребителя. Той е реализиран чрез „Flutter“ и има за задача да визуализира данните, да обработва входовете от потребителя и да изпраща заявки към сървърната част на системата. В този слой няма бизнес логика, като основната му роля е да осигури удобен и интуитивен потребителски интерфейс. „Business Rules“слоят управлява бизнес логиката на приложението. В него се валидират данните, извършват се изчисления (например изчисляване на препоръчителния калориен прием въз основа на целевото тегло и активност) и се дефинират првила, по които работи системата. Слоят гарантира, че всички действия в приложението се извършват съгласно заложените изисквания и цели на проекта. „Service**“** слоят управлява бизнес процесите чрез последователно извикавне на бизнес логиката и координира достъпа до данните. В приложението „**HealthBody&Mind**“ този слой съдържа различни услиги като „UserService“, „FoodService“, „ExerciseService“, които отговарят за обработката на заявки, като регистрация на нов потребител, добавяне на храна или запис на тренировки. „Persistеnce“слоят осъществява директен достъп до базата данни. Неговата основна задача е да извършва операции като съхранение, извличане, актуализиране и изтриване на данни, без да прилага бизнес логика. Слоят е реазлиран чрез репозиторита като „UserRepository“, „FoodRepository“, които комуникират със базата от данни. Този подход осгурява висока степен на модулност, което улеснява развитието на нови функционалности, прави системата по-устойчива при промени и подобрява възможностите за мащабиране спрямо бъдещи бизнес и потребителски изисквания.[6]

**Критични нефункционални изисквания на системата**

Архитектурата на системата „**HealthBody&Mind**“ ще гарантира изпълнението на ключови нефункционални изисквания, които са от съществено значение за постигането на бизнес и потребителските цели. Те включват: мащабируемост, сигурност, надеждност и използваемост.

Мащабируемостта се постига чрез прилагането на клиент-сървър модел и техническото разделяне. Разделението между потребителски интерфейс и бизнес логика позволява независимо развитие и мащабиране на отделните компоненти. Например при нарастване на броя на потребителите може да се увеличи само сървърната инфраструктура без необходимост от промени в клиентското приложението.

Надеждността се постига чрез централизиран контрол на бизнес логиката в сървъра и валидирането на данни още на ниво „Service“ слой. Така се намаляват грешките и се гарантира последователност на обработката на данните независимо от използваното клиентско устройство.

Сигурността се постига чрез изпозлването на „JSON Web Tokens (JWT)“ за удостоверяване и авторизацияна потребителите. Архитектурното решение да се централизира управлението на автентикацията в сървъра гарантира защита на личните данни в съответствие с изискванията на GDPR.

Използваемостта е ключова характеристика на системата и се постига чрез разработване на интуитивен интерфейс. Потребителите имат достъп до основните функции с минимилен брой действия.

Благодарение на всичко това, системата „**HealthBody&Mind**“ съчетава висока надеждност, удобство за потребителите и възможност за мащабиране и развитие в бъдеще, като същевременно изпълнява заложените бизнес и потребителски изисквания.

**Концептуален дизайн на базата данни**

|  |
| --- |
|  |

*Фигура 1.4 – Диаграма на обекти и връзки(ER диаграма)*

Концептуалният модел на базата данни на системата „**HealthBody&Mind**“ отразява обектите и техните взаимодействия, свързани с потребителската дейност, хранителния прием, физическата активност и персонализираните препоръки. Моделът е проектиран, така че да осигури пълна проследимост на данните. Диаграмата отразява нуждите на приложението и поддържа функционалностите, заложени във фунционалните изисквания.

**Таблици и връзки**

**Таблица „user“**

Таблицата „user“ съдържа основната информация за регистрираните потребители в приложението. Всеки запис представлява уникален акаунт и включва задължителни полета като имейл, парола, име и фамилия. Връзки с други таблици:

* едно към едно (1:1) с таблиците „userProfile“ и „nutritionsAndGoals“ – всеки потребител има точно един запис с тези таблици.
* едно към много (1:М) с таблиците „foodIntake“, „exercise“ и „userRecommendation“ – всеки потребител може да има повече от един запис към тези таблици.

|  |  |  |  |
| --- | --- | --- | --- |
| Име на поле | Тип данни | Описание | Ограничения |
| id | INT | Уникален идентификатор | PRIMARY KEY,  AUTO INCREMENT |
| email | VARCHAR(50) | Имейл адрес, който е уникален | NOT NULL, UNIQUE |
| first\_name | VARCHAR(50) | Собствено име | NOT NULL |
| last\_name | VARCHAR(100) | Фамилно име | NOT NULL |
| password | VARCHAR(255) | Хеширана парола | NOT NULL |

*Таблица 3.1 – Структура на таблицата „user“*

**Таблица „userProfile“**

Таблицата „userProfile“ съдържа разширена информация относно всеки потребител, която е от ключово значение за използването на функционалностите на приложението. Тук се съхраняват здравни и лични данни като пол, рожденна дата, ниво на физическа активност, текущо и целево тегло, както и основни цели, свързани със здравословния начин на живот. Връзки с други таблици:

* едно към едно (1:1) с таблицата „user“ (чрез „userId“ като външен ключ) – всеки потребител има точно един профил.

|  |  |  |  |
| --- | --- | --- | --- |
| Име на поле | Тип данни | Описание | Ограничения |
| id | INT | Уникален идентификатор | PRIMARY KEY, AUTO INCREMENT |
| activity\_level | ENUM | Ниво на физическа активност | NOT NULL |
| date\_of\_birth | DATE | Дата на раждане | NOT NULL |
| gender | ENUM | Пол | NOT NULL |
| goal | ENUM | Цел, относно килограмите | NOT NULL |
| goal\_weight | DECIMAL(4,1) | Целево тегло | NOT NULL |
| height | INT | Ръст в сантиметри | NOT NULL |
| image\_url | VARCHAR(255) | Линк към профилна снимка, съхраняваща се във „Firebase“ |  |
| weekly\_goal | ENUM | Седмична цел, относно килогармите | NOT NULL |
| weight | DECIMAL(4,1) | Текущо тегло | NOT NULL |
| user\_id | INT | Външен ключ към таблица „user“ | NOT NULL |

*Таблица 3.2 – Структура на таблицата „userProfile“*

**Таблица „food“**

Таблицата „food“ съхранява информация за различни хранителни продукти, които потребителите могат да избират или добавят към своя дневен прием в приложението „**HealthBody&Mind**“. Всеки запис съдържа данни за името на храната, енергийното и състояние и стойностите на макронутриенти – протеини, въглехидрати и мазнини. Връзки с други таблици:

* едно към много (1:М) с таблица „foodIntake“ – всяка храна може да има повече от един запис.

|  |  |  |  |
| --- | --- | --- | --- |
| Име на поле | Тип данни | Описание | Ограничения |
| id | INT | Уникален идентификатор | PRIMARY KEY, AUTO INCREMENT |
| calories | INT | Калорийно съдържание на 100 грама | NOT NULL |
| carbs | DECIMAL(4,1) | Въглехидрати на 100 грама | NOT NULL |
| fat | DECIMAL(4,1) | Мазнини на 100 грама | NOT NULL |
| measurement | ENUM | Мерна единица за измерване(грам, милилитър) | NOT NULL |
| name | VARCHAR(100) | Име на храната | NOT NULL |
| protein | DECIMAL(4,1) | Протеини на 100 грама | NOT NULL |
| sugar | DECIMAL(4,1) | Захар на 100 грама | NOT NULL |

*Таблица 3.3 – Структура на таблицата „food“*

**Таблица „foodIntake“**

Таблицата „foodIntake“ съхранява данни за консумираните от потребителите храни в приложението. Всеки запис съдържа връзка към избраната храна и потренителя, който е направил записа, количество, дата и категория (закуска, обяд, вечеря и други). Таблицата позволява проследяване на дневния прием на калории и макронутриенти и служи като основа за визуализация на хранителния прогрес в приложението. Връзки с други таблици:

* много към едно (М:1) с таблицата „user“ (чрез „userId“ като външен ключ) – всеки потребител може да повече от едно хранене.
* много към едно (М:1) с таблица „food“ (чрез „foodId“ като външен ключ) – всяка храна може да бъде използвана повече от веднъж.

|  |  |  |  |
| --- | --- | --- | --- |
| Име на поле | Тип данни | Описание | Ограничения |
| id | BIGINT | Уникален идентификатор | PRIMARY KEY, AUTO INCREMENT |
| date | DATE | Дата на консумация | NOT NULL |
| meal\_time | ENUM | Категория на храненето | NOT NULL |
| quantity | DECIMAL(6,2) | Количество на продукта | NOT NULL |
| food\_id | INT | Външен ключ към таблица „food“ | NOT NULL |
| user\_id | INT | Външен ключ към таблица „user“ | NOT NULL |

*Таблица 3.4 – Структура на таблицата „foodIntake“*

**Таблица „exercise“**

Таблицата „exercise“ съхранява информация за физическите активности, отчетени от потребителите в приложението. Всеки запис съдържа данни за типа на активността, продължителност, интензивност и изразхидени калории. Таблицата е ключов елемент за следене на физическото натоварване и за изчисляване на дневния енергиен баланс. Връзки с други таблици:

* много към едно (М:1) с таблица „user“ (чрез „userId“ като външен ключ) – всеки потребител може да има повече от една записана физическа активност.

|  |  |  |  |
| --- | --- | --- | --- |
| Име на поле | Тип данни | Описание | Ограничения |
| id | BIGINT | Уникален идентификатор | PRIMARY KEY, AUTO INCREMENT |
| calories\_burned | INT | Изразходени калории | NOT NULL |
| date | DATE | Дата на извършената активност | NOT NULL |
| duration\_in\_minutes | INT | Продължителност в минути | NOT NULL |
| steps | INT | Брой стъпки |  |
| type | ENUM | Тип на тренировката | NOT NULL |
| workout\_activity\_level | ENUM | Интензивността на тренировката |  |
| user\_id | INT | Външен ключ към таблица „user“ | NOT NULL |

*Таблица 3.5 – Структура на таблицата „foodIntake“*

**Таблица „recommendation“**

Таблицата „recommendation“ съдържа общите препоръки, които могат да се изпозлват като основа за персонализирани съвети, генерирани за всеки потребител. Тези препоръки обхващат различни здравословни насоки в зависимост от целите на потребителя като редукция, покачване или поддръжка на тегло. Връзки с други таблици:

* едно към много (1:М) с таблица „userRecommendation“ – всяка препоръка може да бъде използвана за повече от един потребител.

|  |  |  |  |
| --- | --- | --- | --- |
| Име на поле | Тип данни | Описание | Ограничения |
| id | INT | Уникален идентификатор | PRIMARY KEY, AUTO INCREMENT |
| goal | ENUM | Здравна цел, относно теглото | NOT NULL |
| recommendation | TEXT | Текстово съдържание на препоръката | NOT NULL |

*Таблица 3.6 – Структура на таблицата „recommendation“*

**Таблица „userRecommendation“**

Таблицата „userRecommendation“ съхранява персонализираните препоръки, които приложението генерира въз основа на личните данни на потребителя. Тези препоръки целят да помогнат на потребителя да постигне своите здравословни цели по-лесно. Връзки с други таблици:

* много към едно (М:1) с таблица „user“ (чрез „userId“ като външен ключ) – всеки потребител може да има повече от една персонализирана препоръка.

|  |  |  |  |
| --- | --- | --- | --- |
| Име на поле | Тип данни | Описание | Ограничения |
| id | INT | Уникален идентификатор | PRIMARY KEY, AUTO INCREMENT |
| date | DATE | Дата на генериране на препоръката | NOT NULL |
| filled\_recommendation | TEXT | Запълнен шаблон с личните данни на потребителя | NOT NULL |
| user\_id | INT | Външен ключ към таблица „user“ | NOT NULL |

*Таблица 3.7 – Структура на таблицата „userRecommendation“*

**Таблица „nutritionAndGoals“**

Таблицата „nutritionsAndGoals“ съхранява индивидуалните хранителни цели за всеки потребител на приложението. Тя е предназначена да поддържа проследяване на основни показатели като калориен прием и макронутриенти, необходими за постигане на здравословните цели на потребителя. Връзки с други таблици:

* едно към едно (1:1) с таблица „user“ (чрез „userId“ като външен ключ) – всеки потребител има точно един запис за целите си.

|  |  |  |  |
| --- | --- | --- | --- |
| Име на поле | Тип данни | Описание | Ограничения |
| id | INT | Уникален идентификатор | PRIMARY KEY, AUTO INCREMENT |
| calories\_goal | INT | Дневна калорийна цел | NOT NULL |
| carbs\_goal | INT | Дневна цел за въглехидрати | NOT NULL |
| fat\_goal | INT | Дневна цел за мазнини | NOT NULL |
| protein\_goal | INT | Дневна цел за протеин | NOT NULL |
| step\_goal | INT | Дневна цел за стъпки | NOT NULL |
| water\_goal | DECIMAL(3,1) | Дневна цел за прием на вода | NOT NULL |
| user\_id | INT | Външен ключ към таблица „user“ | NOT NULL |

*Таблица 3.8 – Структура на таблицата „nutritionsAndGoals“*

**Избор на технологии**

Изборът на технологии е продиктуван от потребителските изисквания за система с лесен за използване интерфейс, която е достъпна от множество устройства, с бърза реакция и персонализирано поведения. От друга страна бизнес изискванията акцентират върху възможност за разширяемост, интеграция с външни партньори и предоставяне на допълнителни услуги, като AI консултация. От архитектурна гледна точка, системата изпозлва клиент-сървър модел с ясно разделени отгововорности и многослойна структура (техническо разделяне), което предполага избор на технологии, които поддържат това разпределение ефективно. В този контекст, всяка избрана технология изпълнява конкретна роля, като рендиране на потребителския интерфейс, валидация и обработка на данни и тяхното съхранение.

**Клиентска част**

Системата „**HealthBody&Mind**“ се очаква да предостави интуитивен и бързо реагиращ интерфейс, който да осигурява лесна навигация между основните функции, като управление на профил, добавяне на храна и тренировки, визуализация и прогрес и взаимодействие с AI чат асистент.

За постигането на тези цели е избрана технологията „Flutter“, тъй като тя предоставя богата библиотека от компоненти за изграждане на модерен и адаптивен потребителски интерфейс, като същевременно осигурява висока производителност. Технологията използва съвременен подход, вдъхновен от „React“, при който потребителският интерфейс се изгражда чрез „widgets“, които описват визуалното представяне на екрана, като при всяка промяна се извършва само минимално необходимото обновяване[7]. Това напълно отговаря на нефункционалното изискване за бърза реакция на основните дейности до 2 секунди.

От друга страна „Flutter“ позволява лесна интеграция с „REST API“ чрез „HTTP“ заявки, което е от съществено значение за клиент-сървър модела, приложен в системата. По този начин клиентското приложение ефективно комуникира със сървърната част.

Потребителският интерфейс е реализиран с помощта на езика „Dart“, който е основният програмен език на „Flutter“.

**Сървърна част**

За реализация на сървърната част на системата „**HealthBody&Mind**“ е избрана технологията „Spring Boot“. Изборът е обвързан с архитектурния модел на системата, който се основава на многослойната структура (техническо разделяне), като и с нефункционалните изисквания за сигурност, надеждност и мащабируемост.

„Spring Boot” предлага бърз начин за създаване на приложения, като преглежда пътищата на класовете и конфигурираните „bean-ове“. Освен това интегрира компоненти за сигурност, достъп до база данни и обработка на уеб заявки, отделно прави разумни предложения за това, какво липсва, и добавя тези елементи. Технологията предоставя възможност за съсредоточаване повече върху бизнес функциите и по-малко върху инфраструктурата[8]. Това е особено важно за системата, тъй като сървърната страна трябва да управлява процесите на регистрация, автентикация чрез JWT, съхранение и обработка на лични данни и да осигурява защитен достъп до функциите на приложението.

„Spring Boot“ поддържа създаването на контролери, сървиси и репозиторита, което съответства на разпределението на отговорностите в архитектурата на системата. Това разпределение улеснява поддръжката и бъдещото развитие на платформата.

Сървърната логика е разработена с помощта на езика „Java“, който е подходящ за реализиране на многослойна архитектура.

**База данни**

За съхранение на данни в системата „**HealthBody&Mind**“ е избрана релационната база данни „MySQL“. Изборът се основава на коцептуалния дизайн на базата данни на системата, базиран на ясно дефинирани релации между обектите. Релационния модел на „MySQL“ съответства на създадената „ER“ диаграма, като осигурява възможност за ефективно дефиниране на връзки чрез първични и външни ключове, както и за поддържане на цялостта на данните. Това е особено важно за удовлетворяването на потребителските изисквания.

„MySQL“ е създаден за среди с множество потребители, като може да поддържа множество връзки, позволявайки на различни потребители да достъпват и променят базата данни[9]. Тази характеристика е особено полезна за приложението, защото то е проектирано за работа в среда с множество едновременно активни потребители, като всеки от тях може в реално време да въвежда информация за хранене, тренировки, да редактира профила си.

Интеграцията на „MySQL“ със сървърната част на системата е улеснена чрез стандартни интерфейси (JPA/Hibernate в „Spring Boot“), което гарантира ефективно управление на данните.

За работа с релационната база данни „MySQL“ се използва езикът „SQL (Structured Query Language)“, чрез който се извършват всички операции по съхранение, извличане, актуализиране и изтриване на данни.

**Външни услуги**

Системата „**HealthBody&Mind**“ разчита на две външни услуги. Едната е „Firebase”, а другата - „Dialogflow“.

**„Firebase“**

Потребителските изисквания предполагат възможност за персонализиране на профила чрез добавяне на снимка, което от своя страна изисква стабилна услуга за съхранение на мултимедийни файлове.

„Firebase SDK“ за „Cloud Storage“ добавя сигурността при качване и изтегляне на файлове във „Firebase“ приложения, независимо от качеството на мрежовата връзка[10]. Това е много подходящо за приложението, където файлове се качват от различни устройства при различни условия на свързаност. Отделно „Firebase SDK“ добавя допълнителна сигурност при качване и изтегляне на файлове, като по този начин гарантира спазване на нефунцкионалното изискване за защита на личните данни.

От архитектурна гледна точка, използването на „Firebase SDK“ е напълно съвместимо с клиент-сървър модела на системата. Клиентското приложение може директно да качва и изтегля файлове (снимки), като сървърът участва в процеса само при нужда от удостоверяване или контрол на достъпа, което осигурява минмално натоварване и оптимална производителност.

**„Dialogflow“**

Изборът на „Dialogflow“ е продиктуван от бизнес и потребителските изисквания за предоставяне на AI чат асистент, който да подпомага потребителите чрез предоставяне на съвети и отговори по здравословни въпроси в реално време.

Интеграцията на „Dialogflow“ се осъществява чрез използването на стандартизирани „REST API“ повиквания, което съответста на архитектурата на системата. Чат асистента е реализиран, така че да не натоварва сървъра директно, а до обрбатва заявките чрез външно „API“, като по този начин се гарантира висока производителност, дори при голям брой потребителски заявки.

„Dialogflow“ позволява създаването на изключително гъвката и мащабируема чатбот система. С нарастването на бизнес нуждите, може лесно да се разшири капацитета на чатбота, без да се налагат значителни промени в основната инфраструктура[11]. Тази характеристика пасва на системата, тъй като позолява бъдещо развитие на AI чат асистента с нови функционалности и интеграции, което отговаря на едно от бизнес изискванията.

Комуникацията между клиентската и сървърната част на системата, както и с външната услуга „Dialogflow“, е базирана на формата „JSON (JavaScript Object Notation)“, който осигурява ефективен пренос на структурирана информация между различните компоненти на системата.

**Имплементация**

**Структура на проекта**

Структурата на проекта в системата „**HealthBody&Mind**“ е разделена на две основни части – клиентска, реализирана чрез „Flutter“ и сървърна, реализирана чрез „Spring Boot“ – всяка от които следва ясно дефинирани логически пакети.

**Клиентска част**

Клиентската част на системата е структурирана по начин, който позволява разделението на отговорности и повторната употреба на компоненти. Всички файлове и пакети се намират в директорията „lib/“:

* „screens/“ – включва страниците на приложението, групирани по функционалности. Поддиректорията „log\_in/“ съдържа файловете за вход, регистрация и забравена парола. Това улеснява ориентацията на проекта и подпомага за логическото разделение на интерфейсите.
* „services/“ – съдържа класове, чиято отговорност е да извършват „HTTP“ заявки и да обработват данни от сървъра. Всеки файл, като „user\_service.dart“ или „food\_service.dart“, е отговорен за комуникацията с конкретна сървърна част и следва принципите на разделяне на отговорностите.
* „provider/“ – в тази директория се намират състоянието и логиката на данни чрез „ChangeNotifier“, например „UserProvider“ и „DateProvider“. Това позволява реактивно обновяване на потребителския интерфейс при промяна на данните, като следва модела на „MVVM (Model-View-ViewModel)“.
* „widgets/“ – съдържат се всички многократно използвани визуални компоненти, като „calendar.dart“, „profile\_picture.dart“ и компоненти за графики. Това спомага за повторната употреба на елементите на потребителския интерфейс.
* „utils/“ – тук се намират помощни функции и класове като „validators.dart“, който спомага за валидация на входните данни, „token\_helper.dart“, който работи с JWT и „notification\_generator.dart“, който служи за създаване на известия. Файловете в тази директория са общи и не принадлежат на конкретна част от потребителския интерфейс.
* main.dart – това представлява основната входна точка на приложението, където се инициализират навигацията, темата и състоянието чрез „Provider“.

Тази структура на клиентската част от системата позволява лесно разширяване на проекта, добро разделение на логиката и улеснена поддръжка. Тя отговаря напълно на функционалните и нефункционалните изисквания на системата, като улеснява бъдещо развитие и поддържане на висока степен на персонализация и реактивност.

**Сървърна част**

Сървърната част на системата следва ясна модулна структура, която осигурява разделение на отговорностите и улеснява поддръжката. В основната директория „com.diplomawork.healthbody.mind“ са разположени основни пакети, като:

* „controller/“ – тук се съдържат „REST“ контролерите, които обработват входящите „HTTP“ заявки и са свързващото звено между клиентската част и бизснес логиката. Всеки контролер като, „UserController“, „FoodController“, „ExerciseController“, отговаря за отделна функционалност.
* „service/“ – съдържа бизнес логиката на приложението. Всеки отделен сървис, като „UserService“ или „AIRecommendationService“, реализира операциите, които са използвани от контролерите, като валидиране на данни или управление на процеси.
* „repository/“ – съдържа интерфейси, които комуникират с базата данни чрез „Spring Data JPA“. Например „UserRepository“, „FoodRepository“ и други, предоставят „CRUD (Create, Read, Update, Delete)“ операции и заявки към съответните таблици.
* „model/“ – описва моделите (или обектите), които отговарят на таблиците в базата данни. Те включват „User“, „Food“, „Exercise“ и други, както и подпакет „enums/“, с изброими типове.
* „dto/“ – съдържа „Data Tranfser Object“, чиято роля е да се предават данни между клиент и сървър. Те осигуряват допълнителен слой на абстракция и сигурност, като „UserDataDto“, „FoodDto“ или „ExerciseDto“.
* „security/“ – тук се реализира логиката за удостоверяване и авторизация чрез JWT. Включва конфигурационни класове и услуги като „JwtService“, „WebSecurityConfig“ или „AuthenticationController“.
* „annotations/“ – включва потребителски анотации и валидатори, които се използват за валидация на входни данни. Например „ValidGoalWeight“ или „WeeklyGoalValidator“.
* „exceptions/“ – глобално обработване на изключения чрез „GlobalExceptionHandler“ и дефинирани изключения като „AuthenticationException“ и „UserNotFoundException“.
* „util/“ – съдържа помощен клас „DialogflowTokenManager“ за управление на достъп до външното „API“ за комуникация с AI чат асистента.
* „Application.java“ – това е файлът, който стартира приложението, като стартира вградения уеб сървър и подготвя всички необходими компоненти.

Структурата на сървърната част, е съобразена с принципите на многослойната архитектура и гарантира висока мащабируемост и яснота при разработката и разширението на системата.

**Диаграма на класовете**

|  |
| --- |
|  |

*Фигура 2.1 – Клас диаграма за потребителска история №“1.1“*

Диаграмата представлява регистрационния процес в системата, при който потребителските данни се валидират, обработват и записват в базата. Основните компоненти са разделени по слоеве – контролер, сървиси, репозиторита и модели. Бизнес логиката за изчисляване цели се намира в „UserService“, а сигурността се осигурява от „JwtService“ и „PasswordEncoder“.

|  |
| --- |
|  |

*Фигура 2.2 – Клас диаграма за потребителска история №“2.1“*

Диаграмата представлява процеса по актуализация на личните данни на потребителя. „UserController“ предава данните към „UserService“, където се извършват изчисленията и взаимодействието с репозиторитата. Също така се използват „DTO“ обекти за трансфер на данните, които се съхраняват в „UserProfile“ и „NutritionsAndGoals“.

|  |
| --- |
|  |

*Фигура 2.3 – Клас диаграма за потребителска история №“3.2“*

Диаграмата илюстрира процеса по търсене на храна. Потребителската заявка се обработва от „FoodController“, който извиква „FoodService“, а той от своя страна използва „FoodRepository“, за да извлече записите от базата данни. Резултатът се връща като списък от „FoodDto“, който е извлечен от модела „Food“.

|  |
| --- |
|  |

*Фигура 2.4 – Клас диаграма за потребителска история №“4.1“*

Диаграмата визуализира процеса по записване на физическа активност. Чрез „ExerciseController“ потребителят подава данни за кардио или силова тренировка, които се обработват в „ExerciseService“ и се запазват в базата от данни чрез „ExerciseRepository“. Информацията се представя чрез „DTO“ обекти.

|  |
| --- |
|  |

*Фигура 2.5 – Клас диаграма за потребителска история №“5.2“*

Диаграмата представя функционалността за извличане и визуализация на прогрес за избран период от време. Потребителят подава заявка чрез „ChartController“, която преминава през „ChartService“. От там се правят заявки към „ExerciseRepository“ и „FoodIntakeRepository“, за да се извлечат нужните стойности. На база избрания тип чрез „ChartTypeDto“, данните се филтрират и структурират, за да бъдат върнати към клиента като готови за визуализация.

|  |
| --- |
|  |

*Фигура 2.6 – Клас диаграма за потребителска история №“6.1“*

Диаграмата представя процеса по генериране на персонализирани препоръки на база данните на потребителя. „AIRecommendationController“ приема заявката и я изпраща към „AIRecommendationService“, където на база целите и текущото състояние – “UserDynamicDto” - се подбира подходящ шаблон от „Recommendation“ таблицата, който се попълва с конкретните стойности и се записва в „UserRecommendation“ таблицата в базата данни.

|  |
| --- |
|  |

*Фигура 2.7 – Клас диаграма за потребителска история №“7.1“*

Диаграмата показва архитектурата зад функционалността за AI чат асистента. При въпрос, зададен от потребителя, „AIRecommendationController“ препраща заявка към „AIRecommendationService“, който използва „DialogflowManager“, за да получи достъп до „Google Dialogflow“ чрез „RestTemplate“. След успешно удостоверяване с токен – „AccessToken“ – въпросът се изпраща към външния агент, а полученият отговор се връща обратно към потребителя.

**Диаграма на последователностите**

|  |
| --- |
|  |

*Фигура 2.8 – Диаграма на последователностите за потребителска история №“1.1“*

Диаграмата илюстрира процеса на регистрация на потребителя. След изпращане на данните, системата проверява за съществуващ имейл, хешира паролата и записва новия потребител. След това се създава нов профил с изчислени калории спрямо записаните данни. Накрая се генерира „JWT“ токен, който се връща към килента за удостоверяване.

|  |
| --- |
|  |

*Фигура 2.9 – Диаграма на последователностите за потребителска история №“2.1“*

Диаграмата показва процеса по актуализация на потребителския профил. След извличане на потребителското „ID“ от токена, се обновяват записите в „UserProfile“ и „NutritionsAndGoals“. При наличие на стъпки в подадените данни, се извършва преизчисляване на калориите. В края на процеса, на клиента се връщат обновените калорийни стойности.

|  |
| --- |
|  |

*Фигура 2.10 – Диаграма на последователностите за потребителска история №“3.2“*

Диаграмата описва процеса по търсене на храна по име. След като потребителят подаде заявка с въведено име на храна, методът „getFoodByName(foodName)“, намиращ се във „FoodController“, пренасочва заявката към „FoodService“, където методът „searchFoodByName(foodName)“ се обръща към „FoodRepository“, за извличане на резултати от базата данни. Получените данни се преобразуват в „DTO“ обект и се връщат като отговор на потребителя.

|  |
| --- |
|  |

*Фигура 2.11 – Диаграма на последователностите за потребителска история №“4.1“*

Диаграмата показва процеса по запазване на физическа активност – кардио или силова тренировка - от потребителя. След удостоверяване чрез токен, системата извлича потребителското „ID“ и използва „ExerciseService“, който създава запис за активността и го съхранява чрез използването на „ExerciseRepository“.

|  |
| --- |
|  |

*Фигура 2.12 – Диаграма на последователностите за потребителска история №“5.2“*

Диаграмата илюстрира как потребителят заявява данни за визуализация на прогреса си за определен период от време. След удостоверяване, системата извлича данни от базата в зависимост от типа графика, обобщава ги и връща резултат с попълнени липсващи дни, ако има нужда от тях, към клиента.

|  |
| --- |
|  |

*Фигура 2.13 – Диаграма на последователностите за потребителска история №“6.1“*

Диаграмата показва процеса по извличане или създаване на дневната персонализирана препоръка. При заявка от потребителя, системата проверява за съществуваща препоръка за текущата дата. Ако такава липсва, автоматинчо се генерира нова въз основа на целта и текущите данни на потребителя, след което тя се запазва и връща.

|  |
| --- |
|  |

*Фигура 2.14 – Диаграма на последователностите за потребителска история №“7.1“*

Диаграмата илюстрира процеса по изпращане на въпрос от потребителя към „Dialogflow“ с помощта на системата. При заявка, се валидира или генерира токен за достъп до „Google API“, след което тя се препраща към „Dialogflow“, а резултатът се връща обратно на потребителя.

**Потребителски интерфейс**

|  |
| --- |
| A screenshot of a computer screen  AI-generated content may be incorrect. |

*Фигура 3.1.1 – Потребителски интерфейс за потребителска история №“1.1“*

|  |
| --- |
| A screenshot of a computer  AI-generated content may be incorrect. |

*Фигура 3.1.2 – Потребителски интерфейс за потребителска история №“1.1“*

|  |
| --- |
| A screenshot of a computer  AI-generated content may be incorrect. |

*Фигура 3.2 – Потребителски интерфейс за потребителска история №“2.1“*

|  |
| --- |
| A screenshot of a phone  AI-generated content may be incorrect. |

*Фигура 3.3 – Потребителски интерфейс за потребителска история №“3.2“*

|  |
| --- |
| A screenshot of a computer  AI-generated content may be incorrect. |

*Фигура 3.4 – Потребителски интерфейс за потребителска история №“4.1“*

|  |
| --- |
| A graph with a line  AI-generated content may be incorrect. |

*Фигура 3.5 – Потребителски интерфейс за потребителска история №“5.2“*

|  |
| --- |
| A screenshot of a message  AI-generated content may be incorrect. |

*Фигура 3.6 – Потребителски интерфейс за потребителска история №“6.1“*

|  |
| --- |
| A screenshot of a computer  AI-generated content may be incorrect. |

*Фигура 3.7 – Потребителски интерфейс за потребителска история №“7.1“*

**Определяне на тестови сценарии и тестване**

**Тестови сценарии за потребителска история №“1.1“**

1. Сценарий, при който се прави опит за регистрация със вече съществуващ имейл:

|  |
| --- |
| A screenshot of a computer screen  AI-generated content may be incorrect. |

*Фигура 4.1 – Тестови сценарий за потребителска история №“1.1“*

2. Сценарий, при който се прави опит за регистрация с невалиден формат на имейл и недостатъчно дълга парола:

|  |
| --- |
| A screenshot of a computer  AI-generated content may be incorrect. |

*Фигура 4.2 – Тестови сценарий за потребителска история №“1.1“*

3. Сценарий, при който някои от полетата, които отговарят за личните данни на потребителя, не са попълнени:

|  |
| --- |
| A screenshot of a phone  AI-generated content may be incorrect. |

*Фигура 4.3 – Тестови сценарий за потребителска история №“1.1“*

4. Сценарий, при който всички полета са въведени и са валидирани. След натискане на „Submit“ бутона, се съзадава нов профил за съответния потребител:

|  |
| --- |
| A screenshot of a computer  AI-generated content may be incorrect. |
| A screenshot of a computer  AI-generated content may be incorrect. |

*Фигура 4.4 – Тестови сценарий за потребителска история №“1.1“*

**Тестови сценарии за потребителска история №“2.1“**

1. Сценарий, при който потребителят прави опит за промяна на личните си данни, но те не отговарят на зададените ограничения от приложението:

|  |
| --- |
| A screenshot of a weight loss program  AI-generated content may be incorrect. |

*Фигура 4.5 – Тестови сценарий за потребителска история №“2.1“*

2. Сценарий, при който потребителят прави опит за промяна на личните си данни, като те отговарят на зададените ограничения. След натискане на бутона „Save“, приложението запазва промените, и актуализира калориите, спрямо новата инфомрация.

|  |
| --- |
| A screenshot of a computer  AI-generated content may be incorrect. |
| A screenshot of a weight scale  AI-generated content may be incorrect. |

*Фигура 4.6 – Тестови сценарий за потребителска история №“2.1“*

**Тестови сценарии за потребителска история №“3.2“**

1. Сценарий, при който потребителят въвежда име на храна и при натискане на бутона „Search“ в полето „Your result:“, се показват храни, които отговарят на зададеното име:

|  |
| --- |
| A screenshot of a nutrition information  AI-generated content may be incorrect. |

*Фигура 4.7 – Тестови сценарий за потребителска история №“3.2“*

2. Сценарий, при който потребителят избира храна от полето „Your result:“. След това може да избере необходимото количество и да избере къде да бъде записана в хранителния дневник (закуска, обяд, вечеря и други):

|  |
| --- |
| A screenshot of a food search  AI-generated content may be incorrect. |

*Фигура 4.8 – Тестови сценарий за потребителска история №“3.2“*

3. Сценарий, при който потребителят въвежда име на храна и при натискане на бутона „Search“ в полето „Your result:“, се показва съобщение, че не съществуват храни, които да отговарят на зададеното име:

|  |
| --- |
| A screenshot of a computer  AI-generated content may be incorrect. |

*Фигура 4.9 – Тестови сценарий за потребителска история №“3.2“*

**Тестови сценарии за потребителска история №“4.1“**

1. Сценарий, при който потребителя въвежда данните си за физическа активност. При натискане на бутоните, информацията се запазва, а калориите се изчисляват на базата въведените данни за силова тренировка:

|  |
| --- |
| A screenshot of a computer  AI-generated content may be incorrect. |
| A screenshot of a computer  AI-generated content may be incorrect. |

*Фигура 4.10 – Тестови сценарий за потребителска история №“4.1“*

2. Сценарий, при който потребителят, опитва да натисне бутоните при непопълнени полета, при което приложението сигнализира с необходимото съобщение:

|  |
| --- |
| A screenshot of a computer  AI-generated content may be incorrect. |

*Фигура 4.11 – Тестови сценарий за потребителска история №“4.1“*

**Тестови сценарии за потребителска история №“5.2“**

1. Сценарий, при който потребителят преглежда прогреса си за определен период от време(7 дни или 30 дни):

|  |
| --- |
| A graph with a line  AI-generated content may be incorrect. |
| A graph with a blue line  AI-generated content may be incorrect. |

*Фигура 4.12 – Тестови сценарий за потребителска история №“5.2“*

2. Сценарий, при който няма достатъчно въведени данни от потребителя, при което системата сигнализира с необходимото съобщение:

|  |
| --- |
| A black text on a white background  AI-generated content may be incorrect. |

*Фигура 4.13 – Тестови сценарий за потребителска история №“5.2“*

**Тестови сценарии за потребителска история №“6.1“**

1. Сценарий, при който потребителят при първоначален вход в системата, на основната страница, вижда дневния си съвет, генериран въз основа на неговите лични данни:

|  |
| --- |
| A screenshot of a computer  AI-generated content may be incorrect. |

*Фигура 4.14 – Тестови сценарий за потребителска история №“6.1“*

2. Сценарий, при който, ако потребителят избере от календара, разположен на основанта страница, дата, за която не е генериран дневен съвет, приложението сигнализира с необходимото съобщение (например, ако профилът е направен на датата 6 май, за 5 май няма да има генериран дневен съвет):

|  |
| --- |
| A screenshot of a computer  AI-generated content may be incorrect. |

*Фигура 4.15 – Тестови сценарий за потребителска история №“6.1“*

**Тестови сценарии за потребителска история №“7.1“**

1. Сценарий, при който потребителят натиска бутона „HealthBot“, разположен на основната страница, който отваря нова страница с чат бот. Чат ботът отговаря на въпроси, свързани със здравето и фитнеса. Ако се зададе въпрос, извън темата, ботът връща съобщение, което информира, че въпросът не е по темата:

|  |
| --- |
| A screenshot of a computer  AI-generated content may be incorrect. |
| A white background with pink and black lines  AI-generated content may be incorrect. |

*Фигура 4.16 – Тестови сценарий за потребителска история №“7.1“*

**„Unit“ тестове**

„Unit“ тестовете се базират на използването на „Mock“ обекти за репозиторита и услуги, както и за други зависимости. Изполвани са поплуярни библиотеки, като „JUnit 5“ и „Mockito“.

„UserServiceTest“

|  |
| --- |
| class UserServiceTest {  @Mock  private NutritionsAndGoalsRepository nutritionsAndGoalsRepository;  @Mock  private UserRepository userRepository;  @Mock  private UserProfileRepository userProfileRepository;  @Spy  @InjectMocks  private UserService userService;   @BeforeEach  void setup() {  MockitoAnnotations.*openMocks*(this);  }   @Test  void testSaveCaloriesAndGoalsWhenFlagTrue() {  int userId = 1;   User user = User.*builder*().id(userId).build();  UserProfile profile = UserProfile.*builder*()  .user(user)  .gender(Gender.*MALE*)  .height(176)  .weight(BigDecimal.*valueOf*(79))  .activityLevel(ActivityLevel.*ACTIVE*)  .weeklyGoal(WeeklyGoal.*LOSE\_0\_5\_KG*)  .dateOfBirth(LocalDate.*of*(2002, 2, 14))  .build();   *when*(userRepository.findById(userId)).thenReturn(Optional.*of*(user));   userService.calculateUserCalories(profile, userId, true, 0);   *verify*(nutritionsAndGoalsRepository).save(*argThat*(ng ->  ng.getUser().equals(user) &&  ng.getCaloriesGoal() == 2820 &&  ng.getProteinGoal() == 176 &&  ng.getCarbsGoal() == 352 &&  ng.getFatGoal() == 78 &&  ng.getWaterGoal().equals(BigDecimal.*valueOf*(3)) &&  ng.getStepGoal() == 8000  ));  }   @Test  void testCalculateUserCaloriesWhenFlagTrueSavesCorrectData() {  int userId = 1;   User user = User.*builder*().id(userId).build();  UserProfile profile = UserProfile.*builder*()  .user(user)  .gender(Gender.*MALE*)  .height(176)  .weight(BigDecimal.*valueOf*(79))  .activityLevel(ActivityLevel.*ACTIVE*)  .weeklyGoal(WeeklyGoal.*LOSE\_0\_5\_KG*)  .dateOfBirth(LocalDate.*of*(2002, 2, 14))  .build();   *when*(userRepository.findById(userId)).thenReturn(Optional.*of*(user));   userService.calculateUserCalories(profile, userId, true, 0.0);   ArgumentCaptor<NutritionsAndGoals> captor = ArgumentCaptor.*forClass*(NutritionsAndGoals.class);  *verify*(nutritionsAndGoalsRepository).save(captor.capture());  NutritionsAndGoals saved = captor.getValue();   *assertEquals*(userId, saved.getUser().getId());  *assertEquals*(2820, saved.getCaloriesGoal());  *assertEquals*(176, saved.getProteinGoal());  *assertEquals*(78, saved.getFatGoal());  *assertEquals*(352, saved.getCarbsGoal());  }   @Test  void testCalculateUserCaloriesWhenFlagIsFalseShouldCallSaveUserCaloriesWithExpectedValues() {  int userId = 1;   User user = User.*builder*().id(userId).build();  UserProfile mockProfile = UserProfile.*builder*()  .user(user)  .gender(Gender.*MALE*)  .height(176)  .weight(BigDecimal.*valueOf*(79))  .activityLevel(ActivityLevel.*ACTIVE*)  .weeklyGoal(WeeklyGoal.*LOSE\_1\_KG*)  .dateOfBirth(LocalDate.*of*(2002, 2, 14))  .build();   double water = 2.5;   *doNothing*().when(userService).saveUserCalories(*eq*(userId), *any*(CaloriesDto.class));   userService.calculateUserCalories(mockProfile, userId, false, water);   ArgumentCaptor<CaloriesDto> dtoCaptor = ArgumentCaptor.*forClass*(CaloriesDto.class);  *verify*(userService).saveUserCalories(*eq*(userId), dtoCaptor.capture());   CaloriesDto dto = dtoCaptor.getValue();   *assertEquals*(2570, dto.getCalories());  *assertEquals*(160, dto.getProtein());  *assertEquals*(71, dto.getFats());  *assertEquals*(321, dto.getCarbs());  *assertEquals*(water, dto.getWater(), 0.001);  } |

*Фигура 5.1 – Част от кода на класа „UserServiceTest“*

„FoodServiceTest“

|  |
| --- |
| class FoodServiceTest {  private FoodRepository foodRepository;  private FoodIntakeRepository foodIntakeRepository;  private UserRepository userRepository;  private FoodService foodService;  @BeforeEach  void setUp() {  foodRepository = *mock*(FoodRepository.class);  foodIntakeRepository = *mock*(FoodIntakeRepository.class);  userRepository = *mock*(UserRepository.class);  foodService = new FoodService(foodRepository, foodIntakeRepository, userRepository);  }  @Test  void testSearchFoodByNameFound() {  Food mockFood = Food.*builder*()  .id(1)  .name("Apple")  .calories(52)  .carbs(BigDecimal.*valueOf*(14))  .fat(BigDecimal.*valueOf*(0.2))  .protein(BigDecimal.*valueOf*(0.3))  .sugar(BigDecimal.*valueOf*(10))  .measurement(Measurement.*G*)  .build();   *when*(foodRepository.findByNameContainingIgnoreCase("Apple")).  thenReturn(List.*of*(mockFood));  List<FoodDto> result = foodService.searchFoodByName("Apple");   *assertEquals*(1, result.size());  FoodDto foodDto = result.getFirst();  *assertEquals*("Apple", foodDto.getName());  *assertEquals*(52, foodDto.getCalories());  *assertEquals*(14.0, foodDto.getCarbs());  *assertEquals*(0.2, foodDto.getFats());  *assertEquals*(0.3, foodDto.getProtein());  *assertEquals*(10.0, foodDto.getSugar());  *assertEquals*(String.*valueOf*(Measurement.*G*), foodDto.getMeasurement());   *verify*(foodRepository, *times*(1)).findByNameContainingIgnoreCase("Apple");  }  @Test  void testSearchFoodByNameNotFound() {  *when*(foodRepository.findByNameContainingIgnoreCase("Orange")).thenReturn(Collections.*emptyList*());  List<FoodDto> result = foodService.searchFoodByName("Orange");   *assertEquals*(0, result.size());  *verify*(foodRepository, *times*(1)).findByNameContainingIgnoreCase("Orange");  }  @Test  void testSaveFoodIntakeSuccess() {  Integer userId = 1;  Integer foodId = 10;   User mockUser = new User();  mockUser.setId(userId);   Food mockFood = Food.*builder*()  .id(foodId)  .name("Banana")  .build();   FoodDto foodDto = FoodDto.*builder*()  .id(foodId)  .build();   FoodIntakeDto foodIntakeDto = FoodIntakeDto.*builder*()  .quantity(2.0)  .mealTime(MealTime.*BREAKFAST*.name())  .date(LocalDate.*now*())  .foodDto(foodDto)  .build();   FoodIntake savedFoodIntake = FoodIntake.*builder*()  .id(100L)  .build();   *when*(userRepository.findById(userId)).thenReturn(Optional.*of*(mockUser));  *when*(foodRepository.findById(foodId)).thenReturn(Optional.*of*(mockFood));  *when*(foodIntakeRepository.save(*any*(FoodIntake.class))).  thenReturn(savedFoodIntake);  Long result = foodService.saveFoodIntake(userId, foodIntakeDto);   *assertEquals*(100L, result);  *verify*(userRepository, *times*(1)).findById(userId);  *verify*(foodRepository, *times*(1)).findById(foodId);  *verify*(foodIntakeRepository, *times*(1)).save(*any*(FoodIntake.class));  }  } |

*Фигура 5.1 – Част от кода на класа „FoodServiceTest“*

**Заключение**

Целта на настоящата дипломната работа беше да се разработи уеб базирано приложение – „**HealthBody&Mind**“ – което да обслужи потребители, заинтересовани към подобрение на личното си здраве, като проследяват своята хранителна и физическа активност. Отделно приложението имаше за цел да предостави персонализирани съвети, калорийни цели и напомяния за постигане на здравните цели. Чрез реализирането на решения за управление на потребителски данни, хранителен прием, физическа активност и персонализирани цели, приложението постигна основните задачи и изисквания. Използваните технологии, като „Spring Boot“ и „Flutter“, успешно осигуряват необходимата функционалност, ефективност и мащабируемост на системата.

Системата осигурява удобен и интуитивен интерфейс, който позволява на потребителите да управляват своите лични цели, да проследяват напредъка си и да получават персонализирани съвети, съобразени с техните здравословни цели. Осигурена е сигурността на личните данни чрез използването на „JWT“ токени за удостоверяване и защита на комуникацията между клиент и сървър.

Въпреки постигнатите резултати, системата има потенциал за бъдещо развитие. Един от начините за подобрение е интеграция с външни устройства като фитнес тракери, които да осигуряват по-точна информация за физическата активност на потребителите. Допълнително може да се добавят нови функционалности като интеграция със социални мрежи за споделяне на напредък и резултати. Освен това, разширяването на базата данни с нови храни и добавянето на повече опции за потребителски цели ще помогнат за още по-широка функционалност и покритие на различни нужди. По-този начин приложението „**HealthBody&Mind**“ предоставя стабилна основа за бъдещо развитие и подобрения.

**ИЗТОЧНИЦИ**

[1]. <https://www.fitismed.com/blog/2015/4/21/pros-and-cons-of-fitbit-an-exercise-physiologists-perspective>

[2]. <https://www.garagegymreviews.com/cronometer-review>

[3]. <https://www.pediatriconcall.com/calculators/basel-metabolic-rate-bmr-calculator>

[4]. <https://www.geeksforgeeks.org/client-server-model/>

[5]. <https://www.linkedin.com/pulse/domain-partitioning-vs-techincal-aniket-kulkarni-xqrnf>

[6]. <https://medium.com/@segekaratas/understanding-n-tier-architecture-building-robust-and-scalable-applications-62db30a40b5>

[7]. <https://docs.flutter.dev/ui>

[8]. <https://spring.io/guides/gs/spring-boot>

[9]. <https://aditya-sunjava.medium.com/why-mysql-is-the-superior-choice-for-database-management-over-ms-access-f84950b10956>

[10]. <https://firebase.google.com/docs/storage>

[11]. <https://fastbots.ai/blog/what-is-dialogflow-exploring-its-uses-and-benefits>