**Содержание**

Введение 3

1. Общая часть 5
   1. Описание предметной области 5
   2. Анализ существующей ситуации 6
   3. Анализ существующих аналогов 9
   4. Обоснование необходимости разработки 11
2. Специальная часть 13
   1. Выбор технологии разработки 13
      1. Выбор подхода к разработке 13
      2. Описание среды разработки 14
      3. Описание языка программирования 16
   2. Разработка спецификаций программного обеспечения 17
      1. Построение диаграмм вариантов использования 17
      2. Проектирование логической модели базы данных 19
   3. Проектирование структуры программного обеспечения 23
      1. Построение диаграммы пакетов 23
      2. Построение контекстной диаграммы классов 24
      3. Построение диаграммы размещения 25
   4. Разработка блок-схемы алгоритма одной из основных функций 26
   5. Проектирование интерфейса пользователя 27
   6. Выбор стратегии тестирования 36
   7. Разработка руководства системного программиста 38

Заключение 39

Библиография 40

Приложение А (обязательное) Техническое задание 42

Приложение Б (обязательное) Руководство системного программиста 43

Приложение В (обязательное) Контекстные диаграммы классов 44

**Введение**

Тема курсового проекта проектирование мобильного приложения «Дневник правильного питания. Персонализация диеты» с использованием искусственного интеллекта.

Цель проектирования программного обеспечения состоит в разработке проекта программного продукта для последующей реализации.

Для достижения поставленной цели необходимо выполнить следующие задачи:

* описание предметной области;
* анализ существующей ситуации;
* анализ существующих аналогов;
* обоснование необходимости разработки;
* выбор технологии разработки;
* разработка спецификаций программного обеспечения;
* проектирование структуры программного обеспечения;
* разработка блок-схемы алгоритма одной из основных функций;
* проектирование интерфейса пользователя;
* выбор стратегии тестирования;
* разработка руководства системного программиста.

Курсовой проект состоит из 2 частей: общей и специальной.

В общей части описывается предметная область, проводится анализ текущей ситуации в этой области, осуществляется обзор существующих аналогов и обосновывается необходимость разработки нового проекта.

Специальная часть включает в себя выбор технологии разработки, разработку спецификаций программного обеспечения, проектирование структуры программы, создание блок-схемы алгоритма одной из основных функций, проектирование интерфейса пользователя, выбор стратегии тестирования и разработку руководства для системных программистов.

1. **Общая часть**
   1. **Описание предметной области**

Дневник питания представляет собой программное решение, предназначенное для отслеживания и анализа пищевого рациона пользователей. Для описания более полного понимания бизнес-процессов был проеден анализ предметной области и составлена модель предметной области.

Модель предметной области дневника питания включает следующие ключевые объекты:

1. пользователь - центральный актер системы, использующий приложение для отслеживания своего рациона и заботы о своем здоровье;
2. продукты - еда и напитки, которые потребляются пользователем;
3. цели и планы - заданные пользователем цели и планы по питанию, которые могут быть заданы либо пользователем, либо определены системой;
4. дневник питания - записи о приеме пищи пользователя;
5. меню пользователя - предположительное меню пользователя;
6. данные о пользователе - данные, необходимые для предоставления рекомендаций пользователю.
   1. **Анализ существующей ситуации**

В современном мире многие задумываются о ведении подсчета КБЖУ и диете благодаря развитию современных тенденций к заботе о здоровье. Приложение поможет автоматизировать данный процесс.

На рисунке 1 представлена функциональная модель AS-IS.

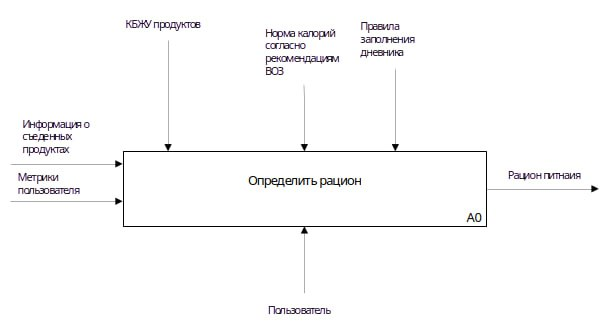


Рисунок 1 – Функциональная модель AS-IS

На рисунках 2, 3, 4 представлена декомпозиция функциональной модели AS-IS.

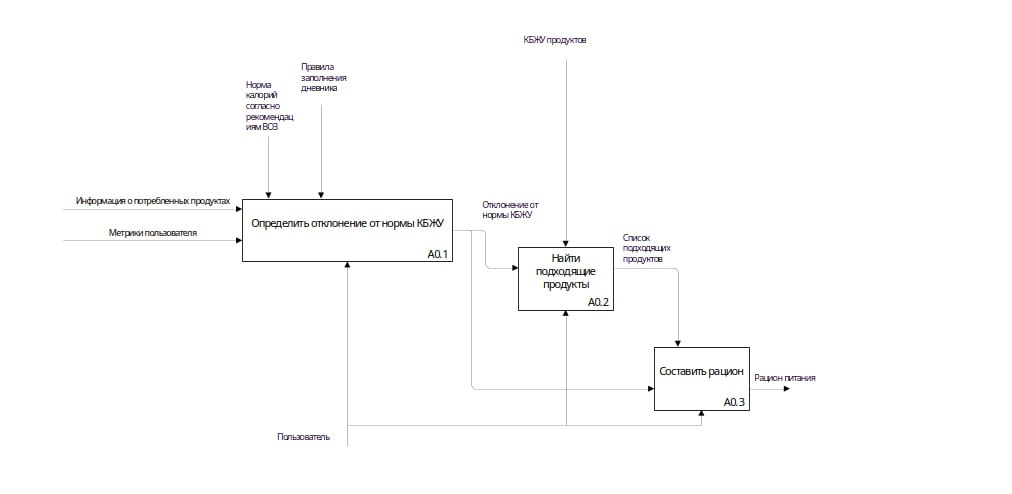


Рисунок 2 – Декомпозиция функциональной модели AS-IS

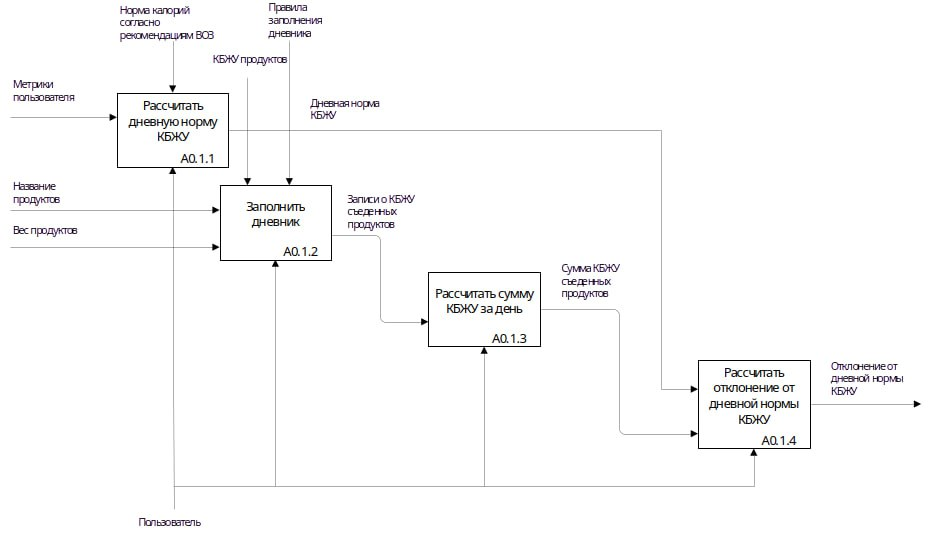


Рисунок 3 – Декомпозиция функциональной модели AS-IS

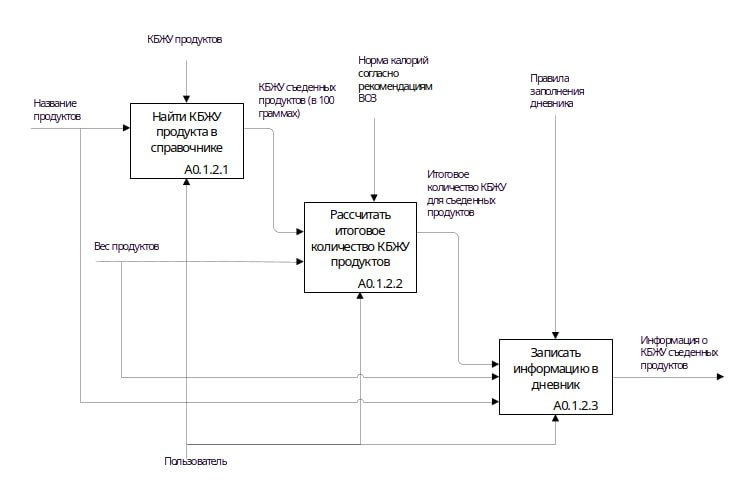


Рисунок 4 – Декомпозиция функциональной модели AS-IS

* 1. **Анализ существующих аналогов**

Целью анализа является выявление преимуществ и недостатков существующих решений, а также определение того, что можно заимствовать или улучшить.

На рынке программного обеспечения в данном сегменте существует несколько ключевых аналогов. Ниже приведены преимущества и недостатки этих приложений.

MyFitnessPal.

* + Преимущества: Обширная база данных продуктов облегчает отслеживание питания, удобный интерфейс и аналитика помогают пользователям оставаться на правильном пути.
  + Недостатки: Ограниченный бесплатный доступ может быть недостаточным для пользователей, а также могут возникать ограничения в настройке рациона.

Lose It!

* + Преимущества: Интеграция с трекерами активности и персонализированные рекомендации по тренировкам облегчают учет калорий и поддержание активного образа жизни.
  + Недостатки: Ограниченные возможности в бесплатной версии и меньшая гибкость в настройке рациона могут быть недостатками для некоторых пользователей.

Cronometer.

* + Преимущества: Анализ макро- и микроэлементов полезен для тех, кто хочет следить за своим питанием более детально. Интеграция с трекерами также удобна.
  + Недостатки: Возможности использования могут быть не так широки, как у коммерческих аналогов, и может потребоваться время для освоения интерфейса.

Open Food Facts.

* + Преимущества: Обновляемая база данных и участие сообщества помогают в поддержании актуальности информации о продуктах.
  + Недостатки: Ограниченные функции отслеживания индивидуального потребления и недостаточное покрытие всех регионов и продуктов могут ограничить его привлекательность для некоторых пользователей.

Исходя из этого, нужно сосредоточиться на создании удобного в использовании приложения с широким набором функций, включая персонализированные рекомендации на основе данных пользователя с использованием искусственного интеллекта. Такой подход позволит предложить конкурентное решение на рынке приложений для здорового питания.

* 1. **Обоснование необходимости разработки**

В ходе анализа рынка коммерческого и свободно распространяемого программного обеспечения были выявлены сильные и слабые стороны конкурирующих программ.

Таким образом, были выявлены преимущества разрабатываемой системы перед аналогами:

* анализ данных с использованием искусственного интеллекта: автоматизированный анализ данных, позволяющий предоставлять более персонализированные рекомендации на основе данных пользователя;
* упрощенный ввод данных: минимизация усилий пользователя при вводе данных, что снижает барьер для использования системы.

На основе данных, изложенных выше, разрабатываемая системы выделяется интеллектуальным анализом данных и упрощенным вводом данных, что предоставляет большую потенциальную аудиторию, и, соответственно, хорошую конкурентоспособность.

На рисунке 5 представлена модель TO-BE информационного процесса после проектирования. А также на рисунках 6 и 7 представлена более расширенная версия модели TO-BE.

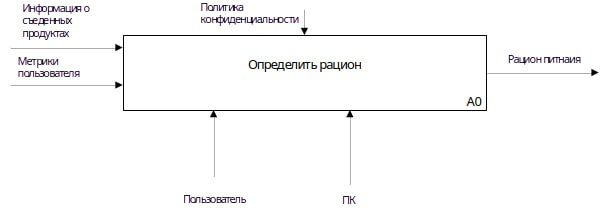


Рисунок 5 - Функциональная модель TO-BE

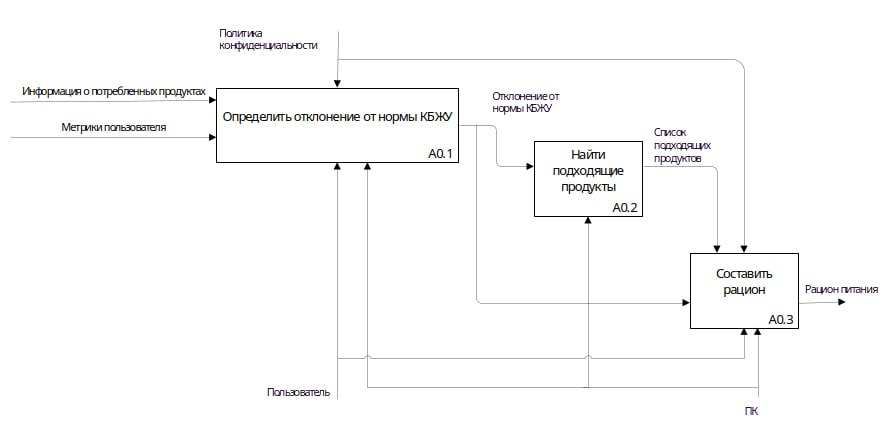


Рисунок 6 - Декомпозиция функциональной модели TO-BE

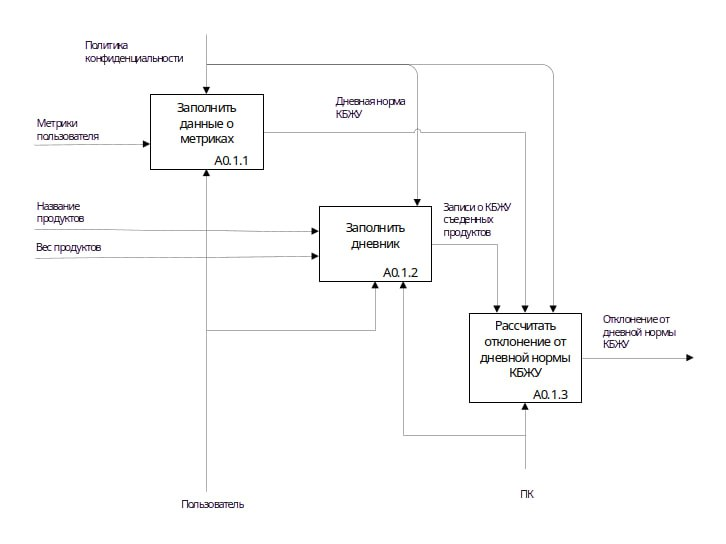


Рисунок 7 - Декомпозиция функциональной модели TO-BE

1. **Специальная часть**
   1. **Выбор технологии разработки**
      1. Выбор подхода к разработке

Рассмотрим два основных подхода к проектированию программного обеспечения: структурный и объектно-ориентированный.

Структурный подход.

* + Программа структурируется в виде иерархии модулей и подпрограмм, что упрощает управление сложностью проекта.
  + Этот подход часто используется в случаях, когда необходимо быстро создать простое решение без введения дополнительных сложностей, которые могут присутствовать в объектно-ориентированном подходе.
  + Зависимости между модулями могут быть легко управляемыми и понятными, что упрощает поддержку и модификацию кода.

Объектно-ориентированный подход.

* + Классы и объекты позволяют моделировать реальные объекты и абстракции, обеспечивая модульность и скрытие деталей реализации.
  + Эти концепции позволяют повторно использовать код, облегчают его понимание и расширение за счет создания иерархии классов и переопределения методов.
  + Объектно-ориентированный подход часто применяется в больших проектах, где необходимо эффективное управление сложностью и возможность легкого масштабирования системы.

Для разработки выбран объектно-ориентированный подход благодаря возможностью лучшего управления сложностью программы и обеспечения более гибкой архитектуры.

* + 1. Описание среды разработки

При выборе технологии разработки необходимо учитывать требования проекта, удобство использования и поддержку, а также экосистему разработки. Visual Studio Code (VS Code) от Microsoft является одним из лучших вариантов. Ниже приведены преимущества этой среды.

Удобный интерфейс и расширяемая функциональность: предлагает интуитивно понятный интерфейс, что делает процесс написания кода эффективным и приятным. Его множество расширений позволяет настраивать среду разработки под конкретные потребности, что важно при интеграции различных инструментов и библиотек, включая работу с искусственным интеллектом и создание конструктора предзаказа.

Поддержка широкого спектра языков программирования и технологий: обладает мощным набором функций, поддерживающих множество языков программирования и технологий. Это позволяет выбирать наиболее подходящие инструменты для решения конкретных задач, что особенно важно при работе с искусственным интеллектом и интеграции различных библиотек и фреймворков.

Активное развитие и поддержка сообщества разработчиков: получает регулярные обновления и исправления ошибок благодаря активному вкладу сообщества разработчиков. Официальная документация и широкая база знаний делают поддержку этой среды разработки удобной и эффективной.

Интеграция с системами контроля версий: удобные инструменты для работы с Git и другими системами контроля версий помогают эффективно управлять кодовой базой проекта и совместно работать с другими разработчиками.

Гибкость настройки через расширения: благодаря огромному количеству доступных расширений, VS Code можно настроить под конкретные потребности проекта, обеспечивая гибкость и возможность выбора подходящих инструментов и библиотек.

Учитывая все эти факторы, VS Code является отличным выбором для разработки, предоставляя удобную среду разработки с широкими возможностями настройки и интеграции различных технологий.

* + 1. Описание языка программирования

Язык программирования C# (C Sharp) — это современный объектно-ориентированный язык программирования, созданный компанией Microsoft. Он является частью экосистемы .NET и часто используется для разработки разнообразных приложений - от настольных до веб-сервисов.

Преимущества C#:

* может быть использован на различных платформах, включая windows, linux и macos.
* предоставляет мощные инструменты разработки, такие как управление памятью, поддержка многопоточности и сильная типизация.
* интегрируется с .NET Framework или .NET Core, что обеспечивает создание масштабируемых и надежных приложений.

Выбор C# для разработки бэкенд-части обоснован, поскольку этот язык программирования предоставляет необходимые инструменты и возможности для создания надежных и масштабируемых веб-приложений.

* 1. **Разработка спецификаций программного обеспечения** 
     1. Построение диаграмм вариантов использования

Диаграммы вариантов использования позволяют наглядно представить ожидаемое поведение системы. Основными понятиями диаграмм вариантов использования являются: действующее лицо, вариант использования, связь.

Действующее лицо – внешняя по отношению к разрабатываемому программному обеспечению сущность, которая взаимодействует с ним с целью получения или предоставления какой-либо информации. Действующими лицами могут быть пользователи, другое программное обеспечение или какие-либо технические средства, взаимодействующие с разрабатываемым программным обеспечением.

Вариант использования – некоторая очевидная для действующего лица процедура, решающая его конкретную задачу. Все варианты использования, так или иначе, связаны с требованиями к функциональности разрабатываемой системы и могут сильно отличаться по объему выполняемой работы.

Связь – взаимодействие действующих лиц и соответствующих вариантов использования.

На рисунке 8 изображена диаграмма вариантов использования.

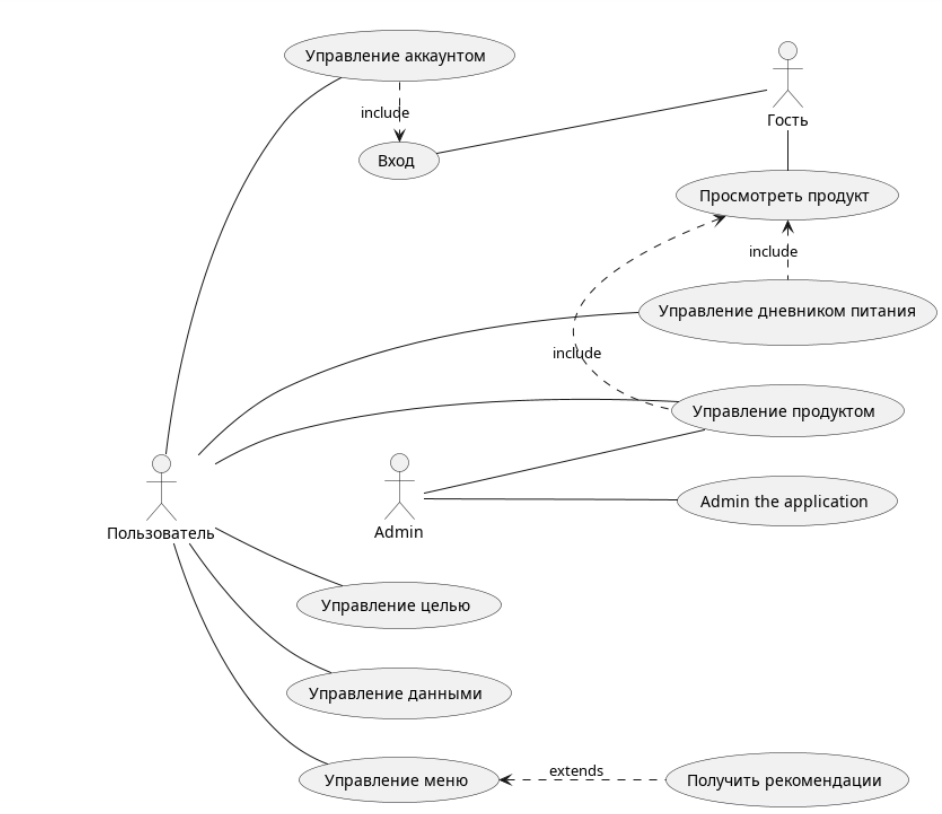
****

Рисунок 8 - Диаграмма вариантов использования

* + 1. Проектирование логической модели базы данных

Логическая модель данных является начальным прототипом будущей базы данных. Основным средством разработки логической модели данных в настоящий момент являются различные варианты ER-диаграмм (Entity-Relationship, диаграммы сущность-связь).

Построение логической схемы базы данных необходимо для определения ее будущей структуры. Таким образом, логическая диаграмма является необходимой.

Логическая схема для базы данных пользователя представлена на рисунке 9.

Эта база содержит информацию об пользователях и токенах обновления.

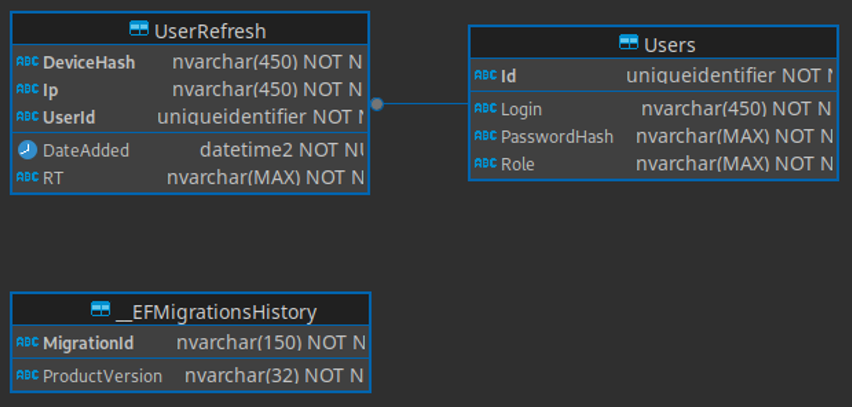


Рисунок 9. Логическая схема БД пользователя

Таблица Users - содержит информацию о пользователях.

Таблица UserRefresh - содержит информацию о токенах обновления пользователя.

Представлена на рисунке 10. Содержит информацию о потребленных продуктов по дням для пользователя.

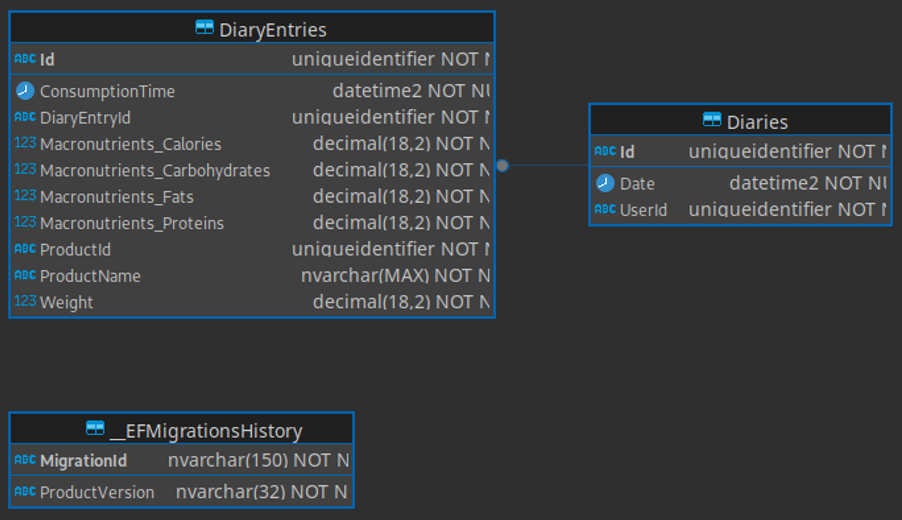


Рисунок 10 - Логическая схема БД дневника питания

Таблица Diary - содержит информацию об общем дневном рационе.

Таблица DiaryEntry - содержит информацию о записи в дневнике пользователя о потребленном продукте.

Логическая схема представлена на рисунке 11. Содержит информацию о планируем меню пользователя.

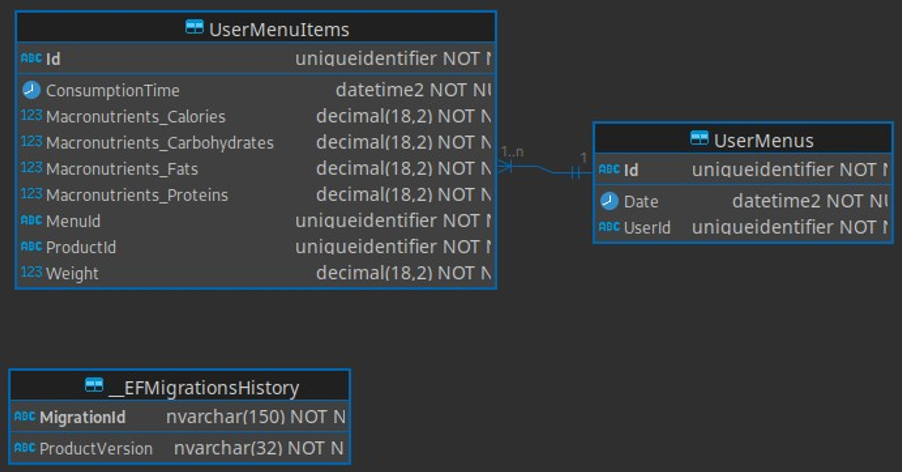


Рисунок 11 - Логическая схема БД меню пользователя

Таблица UserMenu - содержит информацию о меню пользователя.

Таблица UserMenuItem - содержит информацию о конкретном продукте в запланированном меню пользователя.

Логическая схема БД плана пользователя представлена на рисунке 12. Представляет данные о целях пользователя относительно оптимального соотношения КБЖУ для изменения веса.

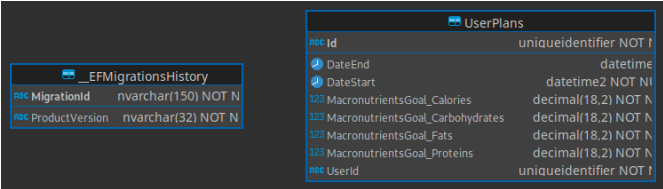


Рисунок 12 - Логическая схема БД плана пользователя

Таблица UserPlan - содержит информацию о плане пользователя по количеству КБЖУ на определенный период.

Логическая схема БД данных пользователя представлена на рисунке 13. Содержит данные о показателях пользователя.

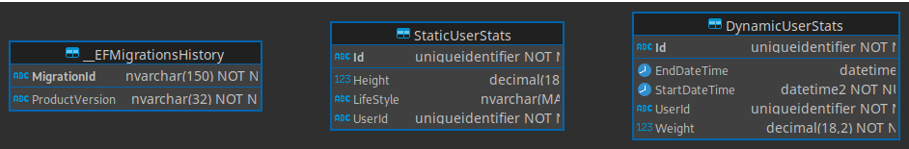


Рисунок 13 - Логическая схема БД данных пользователя

Таблица StaticUserStats - содержит информацию о статичной информации пользователя.

Таблица DynamicUserStats - содержит информацию об изменяющихся данных пользователя.

* 1. **Проектирование структуры программного обеспечения** 
     1. Построение диаграммы пакетов

Диаграмма пакетов показывает, из каких частей состоит проектируемая программная система, и как эти части связаны друг с другом.

На рисунке 14 изображена диаграмма пакетов, которая показывает структуру программной системы и взаимодействие между компонентами.

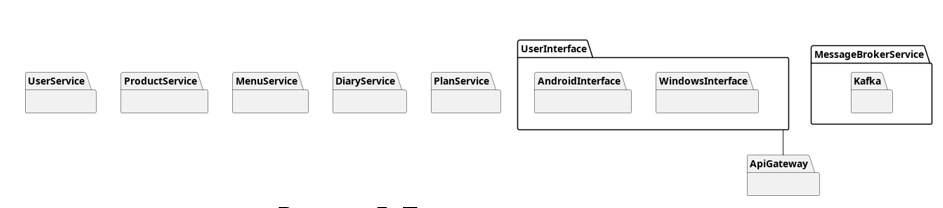


Рисунок 14 - Диаграмма пакетов

* + 1. Построение контекстной диаграммы классов

Диаграмма классов необходима для проектирования будущей структуры основных классов программы и их взаимодействия для последующей реализации.

Диаграммы классов представлены в приложении В.

* + 1. Построение диаграммы размещения

Диаграмма размещения отражает физические взаимосвязи между программными и аппаратными компонентами системы. Каждой части аппаратных средств системы. Соединения узлов означают наличие в системе соответствующих коммуникационных каналов. Внутри узлов указывают размещенные на данном оборудовании программные компоненты разрабатываемой программной системы, сохраняя указанные на диаграмме компонентов отношения зависимости.

На рисунке 16 изображена диаграмма размещения.

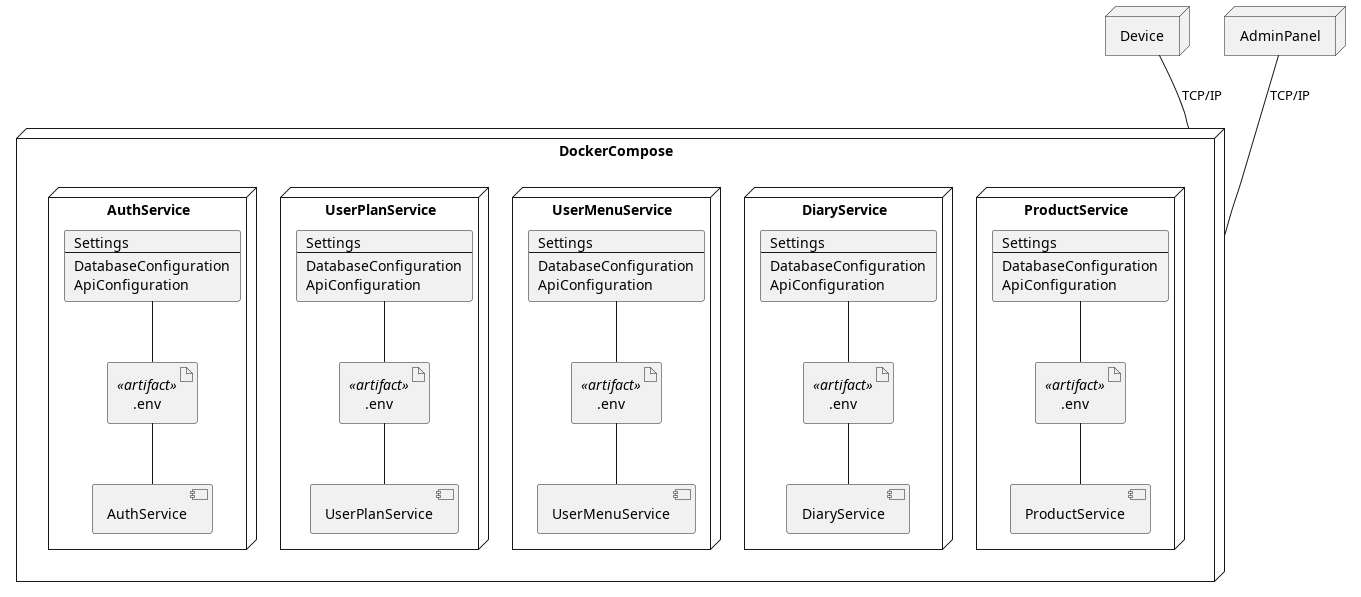
****

Рисунок 16 - Диаграмма размещения

Ее назначение – представление в графическом виде инфраструктуры проектируемого приложения.

* 1. **Разработка блок-схемы алгоритма одной из основных функций**

Блок-схемой называют графическое представление алгоритма, в котором он изображается в виде последовательности связанных между собой функциональных блоков, каждый из которых соответствует выполнению одного или нескольких действий.

На рисунке 17 изображена блок-схема, которая представляет собой логическую структуру процесса проверки аутентификации.

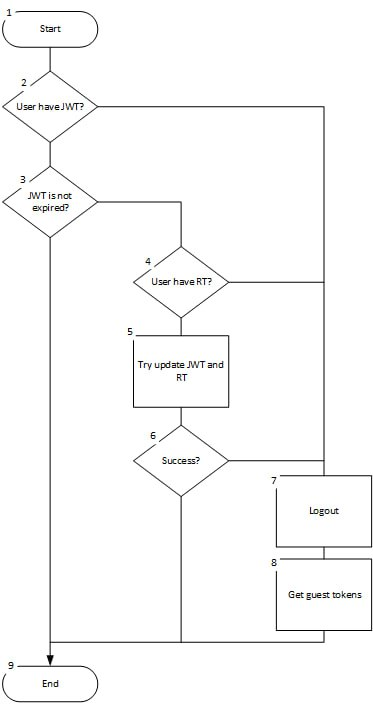


Рисунок 17 - Блок-схема алгоритма аутентификации

* 1. **Проектирование интерфейса пользователя**

Пользовательский интерфейс – интерфейс, обеспечивающий передачу информации между пользователем-человеком и программно-аппаратными компонентами компьютерной системы. Интерфейс представляет собой совокупность средств и методов, при помощи которых пользователь взаимодействует с различными машинами, устройствами и аппаратурой.

Сравнение пользовательских моделей интерфейсов процедурного и объектно-ориентированных типов приведено в таблице 1.

Таблица 1 – Сравнение пользовательских моделей интерфейсов

|  |  |
| --- | --- |
| **Процедурно-ориентированные пользовательские интерфейсы** | **Объектно-ориентированные пользовательские интерфейсы** |
|  | |
| Обеспечивают пользователей функциями, необходимыми для выполнения задач | Обеспечивают пользователям возможность взаимодействия с объектами |
| Акцент делается на задачи | Акцент делается на входные данные и результаты |
| Пиктограммы представляют приложения, окна или операции | Пиктограммы представляют объекты |
| Содержание папок и справочников отображается с помощью таблиц и списков | Папки и справочники являются визуальными контейнерами объектов |

Таким образом, процедурно-ориентированные интерфейсы сконцентрированы на выполнении конкретных задач и предоставлении функциональности, в то время как объектно-ориентированные интерфейсы больше ориентированы на взаимодействие пользователя с объектами и отображение структуры данных.

Кроме подразделения интерфейсов на различные категории, также существуют различные типы интерфейсов.

Сравнение различных типов интерфейсов приведено в таблице 2.

Таблица 2 – Сравнение различных типов интерфейсов

|  |  |  |  |
| --- | --- | --- | --- |
| **Критерий сравнения** | **Интерфейс меню** | **Интерфейс со свободной навигацией** | **Интерфейс прямого манипулирования** |
|  | | | |
| Тип интерфейса | Процедурно-ориентированный | Процедурно-ориентированный | Объектно-ориентированный |
| Простота освоения | Довольно прост в освоении | Прост в освоении, но пользователь может запутаться в пиктограммах | Прост в освоении, но пользователь может запутаться в пиктограммах |
| Элементы управления | Выбор команд осуществляется в иерархичном интерфейсе меню, а также может быть вызван «горячими» клавишами | Элементы управления вызываются по кнопке | Элементы управления могут перетаскиваться по форме |
| Взаимодействие с пользователем | Организует статичное и удобное и быстрое взаимодействие с пользователем | Позволяет осуществлять визуальную связь с пользователем | Позволяет осуществлять визуальную связь с пользователем |

На основе проведенного анализа сделан вывод о том, что наиболее подходящим для проектирования является интерфейс со свободной навигацией, поскольку он сочетает в себе простоту освоения с возможностью визуальной связи с пользователем и более свободный подход к навигации.

Оценка интерфейса пользователем проводится по следующим критериям:

* Простота освоения и запоминания операций системы.
* Скорость достижения результатов при использовании системы.
* Субъективная удовлетворенность при эксплуатации системы (удобство работы, утомляемость и т.д.).

Пользовательская модель интерфейса – это совокупность обобщенных представлений конкретного пользователя или некоторой группы пользователей о процессах, происходящих во время работы программы или программной системы.

Существуют три различные модели пользовательского интерфейса: модель программиста, модель пользователя и программная модель.

Сравнение различных моделей пользовательского интерфейса приведено в таблице 3.

Таблица 3 – Сравнение различных типов пользовательского интерфейса

|  |  |  |
| --- | --- | --- |
| **Критерий сравнения** | **Модель пользователя** | **Программная модель** |
|  | | |
| Сложность освоения | Низкая | Высокая |
| Наличие привычных элементов | Присутствуют в большом количестве | В основном отсутствуют |
| Основные цели проектирования | Удобство использования, скорость работы, наличие привычных общепринятых функций | Функциональность, эффективность, технологичность, внутренняя стойкость |

Наиболее подходящим для проектирования является пользовательский тип интерфейса, поскольку удобство вызова различных команд и привычность использования программы являются определяющими факторами при создании внешнего модуля формирования документов.

Диалог – это процесс обмена информацией между пользователем и программной системой, осуществляемый через интерактивный терминал и по определенным правилам.

В зависимости от вида используемых в конкретном случае синтаксиса и семантики различают три формы диалога: фразовую, директивную, табличную.

Сравнительная характеристика форм диалога представлена в таблице 4.

Таблица 4 - Сравнительная характеристика форм диалога

|  |  |  |
| --- | --- | --- |
| **Фразовая форма** | **Директивная форма** | **Табличная форма** |
|  | | |
| Общение на естественном языке | Использование формального языка | Пользователь выбирает ответ из предложенных программой |
| Повелительные, повествовательные и вопросительные предложения | Команды для инициирования процессов | Простой синтаксис и однозначная семантика |
| Гибкий формат общения с возможностью фиксации фраз | Структурированные команды с идентификатором процесса | Удобство выбора ответа для непрофессионалов и редко пользующихся пользователях |
| Интерактивное общение на естественном языке | Точное и однозначное указание задач | Удобство реализации и понимания |
| Применима в различных контекстах общения | Часто используется в программировании | Предпочтительна для простых операций |

После проведения сравнительной характеристики выбрана директивная форма диалога, т.к. данная форма снижает вероятность ошибок за счет наличия команд и отсутствия вводимой информации.

При проектировании абстрактных диалогов используют графы диалога.

Граф диалога – ориентированный взвешенный граф, каждой вершине которого сопоставлена конкретная картинка на экране (кадр) или определенное состояние диалога, характеризующееся набором доступных пользователю действий. Дуги, исходящие из вершин, показывают возможные изменения состояний при выполнении пользователем указанных действий. В качестве весов дуг указывают условия переходов из состояния в состояние и операции, выполняемые во время перехода.

На рисунках 18 - 23 представлены графы диалога, которые используются для моделирования поведения системы.

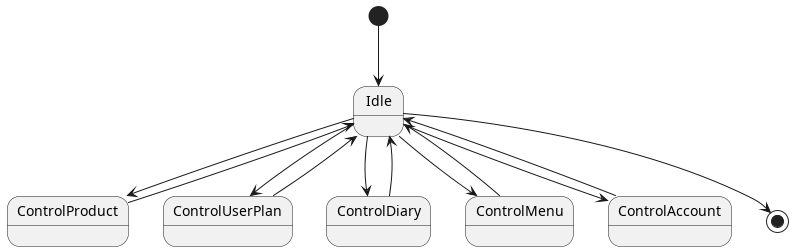


Рисунок 18 – Граф диалога пользовательского интерфейса

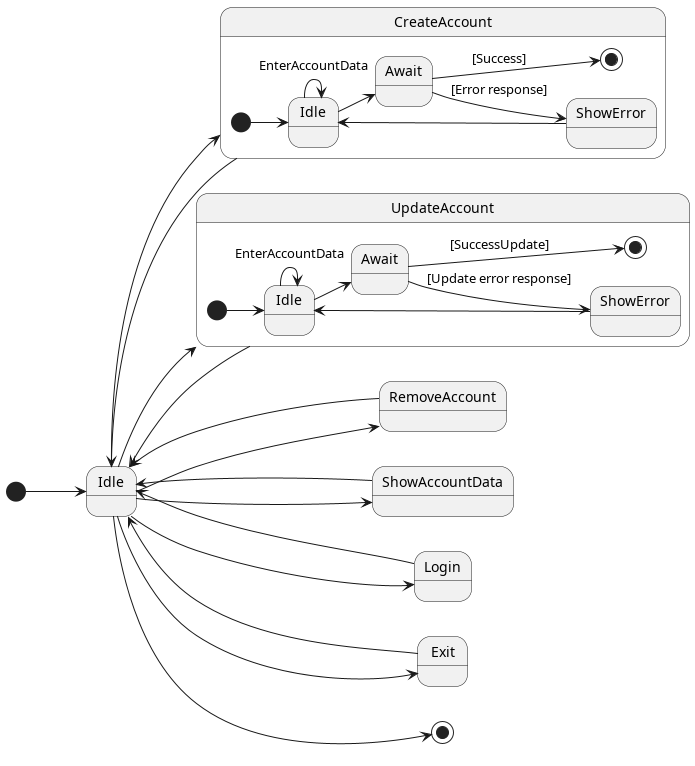


Рисунок 19 – Граф диалога управления аккаунтом

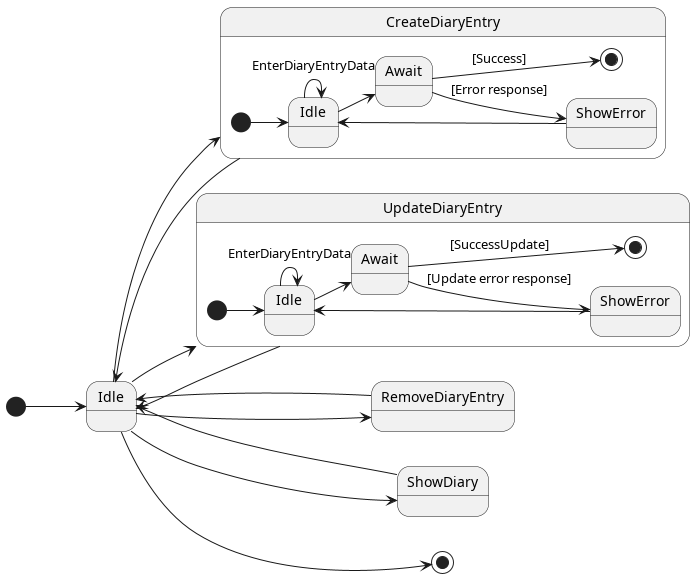


Рисунок 20 – Граф диалога управления дневника питания

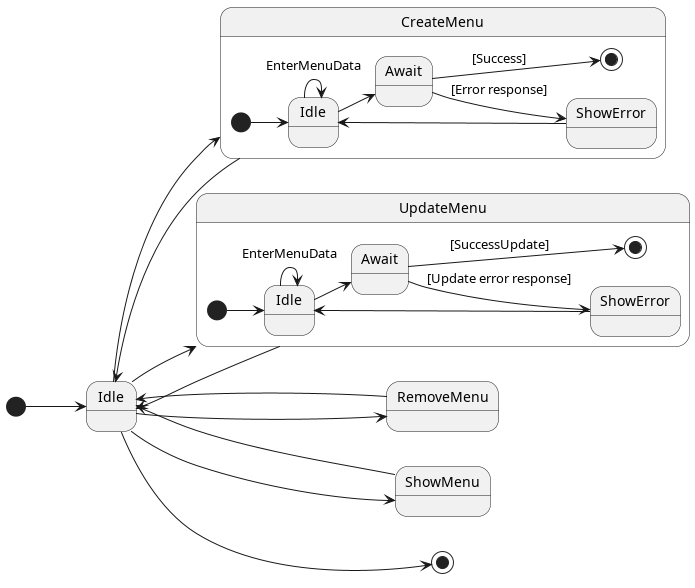


Рисунок 21 – Граф диалога управления меню

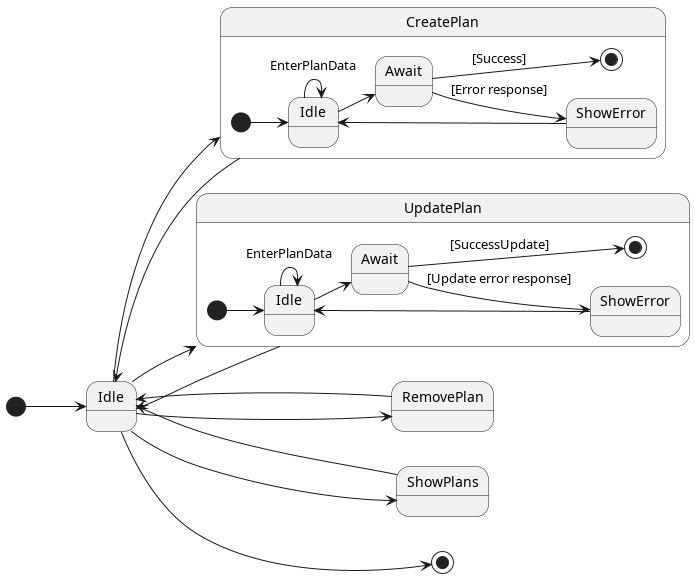


Рисунок 22 – Граф диалога управления планом

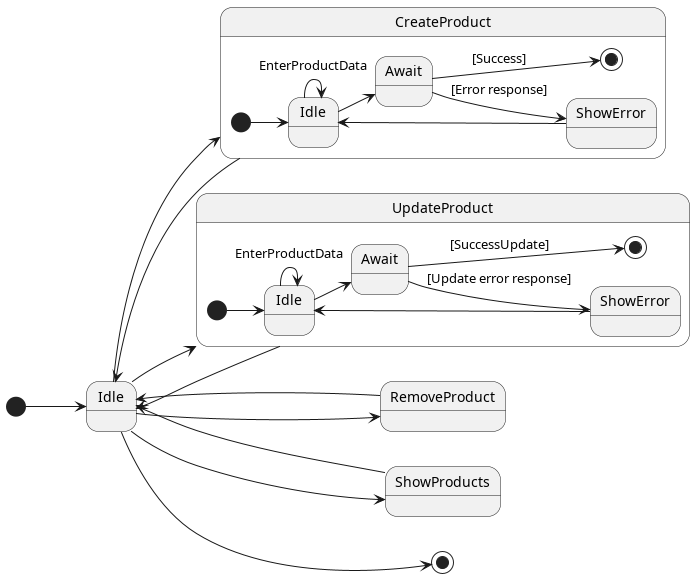


Рисунок 23 – Граф диалога управления продуктами

На рисунках 24 - 29 представлены схемы интерфейса пользователя.

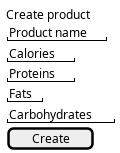


Рисунок 24 – Создание продукта

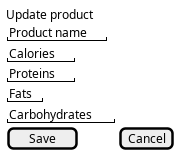


Рисунок 25 – Изменение продукта

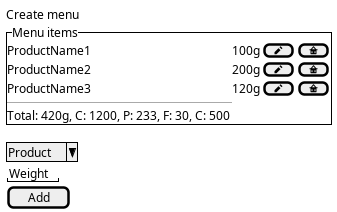


Рисунок 26 – Создание меню



Рисунок 27 – Создание записи в дневнике питания

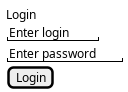


Рисунок 28 – Логин

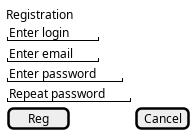


Рисунок 29 – Регистрация

**2.6 Выбор стратегии тестирования**

Стратегия тестирования определяет, насколько выбранный тест удовлетворяет ожиданиям тестирования. Под стратегией тестирования понимают совокупность тех систематических методов, которые используют для создания или отбора тестов, включаемых в тестовый комплект.

Сравнительная характеристика тестирования представлена в таблице 5.

Таблица 5 – Сравнительная характеристика тестирования

|  |  |
| --- | --- |
| **Структурное тестирование** | **Функциональное тестирование** |
|  | |
| Проверка правильности работы программного кода и базы данных, а также производительности сайта | Проверка соответствия бизнес-требованиям приложения и его функциональности |
| Программный код, база данных, производительность сайта | Функциональные возможности приложения, результаты действий пользователя |
| Проверка внутренних структур и механизмов приложения | Проверка видимых результатов действий пользователя |
| Проверка корректности кода, базы данных и производительности | Проверка соответствия функциональных требований приложения |
| Требует технической экспертизы для понимания кода и базы данных | Может быть более доступным для представителей бизнеса |
| Тестирование кода, базы данных, производительности | Тестирование пользовательских сценариев и функциональности |

Таким образом, функциональное тестирование является предпочтительным выбором для обеспечения соответствия, удовлетворения потребностей пользователей и обеспечения высокого уровня качества продукта.

В таблице 6 представлено тестирование.

Таблица 6 - Тестирование

|  |  |  |
| --- | --- | --- |
| **Функция** | **Вход** | **Ожидаемый результат** |
|  | | |
| Авторизация | Корректный логин и пароль | Токен доступа и обновления |
| Авторизация | Некорректный пароль | Статус запроса - запрещено |
| Создание аккаунта | Корректные логин, пароль и почта | Успешный запрос |
| Создание аккаунта | Дублирующий логин | Статус запроса - неверный запрос. Пользователь с таким логином уже существует. |
| Создание аккаунта | Логин содержит менее 4 символов | Статус запроса - неверный запрос. Логин должен состоять не менее чем из 4 символов. |
| Создание аккаунта | Логин содержит более 25 символов | Статус запроса - неверный запрос. Логин должен состоять максимум из 25 символов. |
| Создание записи в дневнике питания | Корректные данные продукта и веса | Успешный запрос |
| Создание записи в дневнике питания | Отрицательные или нулевые значения для потребленного веса продукта | Статус - неверный запрос. Вес потребленного продукта должен быть более 0. |
| Создание записи в дневнике питания | Идентификатор продукта неверный | Статус - неверный запрос. Невозможно найти продукт. |
| Удаление записи в дневнике питания | Корректные данные | Успешный запрос |

**2.7 Разработка руководства системного программиста**

Руководство системного программиста разрабатывается на основании ГОСТ 19.503-79 и должно содержать следующие разделы:

* общие сведения о программе;
* структура программы;
* настройка программы;
* проверка программы;
* дополнительные возможности;
* сообщения системному программисту.

Руководство системного программиста приведено в приложении Б.

**Заключение**

В ходе выполнения курсового проекта все поставленные задачи были выполнены, таким образом, можно считать, что поставленная цель была достигнута.

В соответствии с проведенным анализом был выбран объектно-ориентированный подход к проектированию программного обеспечения.

Была выбрана среда VS Code для разработки, так как она предоставляет удобную среду разработки с широкими возможностями настройки и интеграции различных технологий.

Выбран C# для разработки, так как этот языки программирования предоставляет необходимые инструменты и возможности.

Для достижения поставленной цели были выполнены следующие задачи:

* описание предметной области;
* анализ существующей ситуации;
* анализ существующих аналогов;
* обоснование необходимости разработки;
* выбор технологии разработки;
* разработка спецификаций программного обеспечения;
* проектирование структуры программного обеспечения;
* разработка блок-схемы алгоритма одной из основных функций;
* проектирование интерфейса пользователя;
* выбор стратегии тестирования;
* разработка руководства системного программиста.

**Библиография**

|  |
| --- |
| Нормативно-правовые акты: |
| 1. ГОСТ 7.1-2003 Библиографическое описание документа. Общие требования и правила составления |
|  |
| 1. ГОСТ 34.602-89 Межгосударственный стандарт. ИТ Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы 2. ГОСТ 19.503-79 Межгосударственный стандарт. ЕСПД Руководство системного программиста |
| 1. ГОСТ 19.102-77 Межгосударственный стандарт. ЕСПД Стадии разработки |
| 1. ГОСТ Р 2.105-2019 Общие требования к текстовым документам |
| Научные источники: |
| 1. Kapferer S., Zimmermann O. Domain-Driven Service Design: Context Modeling, Model Refactoring and Contract Generation //Service-Oriented Computing: 14th Symposium and Summer School on Service-Oriented Computing, SummerSOC 2020, Crete, Greece, September 13-19, 2020 14. – Springer International Publishing, 2020. – С. 189-208. |

|  |
| --- |
| Электронные ресурсы:   1. DeadlineMadness. Использование диаграммы классов UML при проектировании и документировании программного обеспечения // Habr.com: русскоязычный веб-сайт в формате системы коллективных блогов, 2021. URL: https://habr.com/ru/articles/572234/ (дата обращения: 03.04.2024). |
| 1. DeadlineMadness. Использование диаграммы вариантов использования UML при проектировании программного обеспечения // Habr.com: русскоязычный веб-сайт в формате системы коллективных блогов, 2021. URL: https://habr.com/ru/articles/566218/ (дата обращения: 28.03.2024). |
| 1. Giretti, A. (2023). Introduction to Application Development Best Practices. In: Coding Clean, Reliable, and Safe REST APIs with ASP.NET Core 8. Apress, Berkeley, CA. URL: <https://doi.org/10.1007/978-1-4842-9979-1_3> (дата обращения: 05.02.2024). |
| 1. OTUS. Практическое DDD. Часть 1: Создание правильных основ. URL: <https://habr.com/ru/companies/otus/articles/726628> (дата обращения: 05.02.2024). |

**Приложение А**

**(обязательное)**

**Техническое задание**

А.1 Техническое задание на проектирование для разработки мобильного приложения "дневник правильного питания. персонализация диеты" с использованием ИИ в количестве 15 страниц.

**Приложение Б**

**(обязательное)**

**Руководство системного программиста**

В.1 Руководство системного программиста для разработки мобильного приложения "дневник правильного питания. персонализация диеты" с использованием ИИ в количеств 8 страниц.

**Приложение В**

**(обязательное)**

**Контекстные диаграммы классов**

На рисунках В.1 – В.5 представлены схемы диаграмм классов.

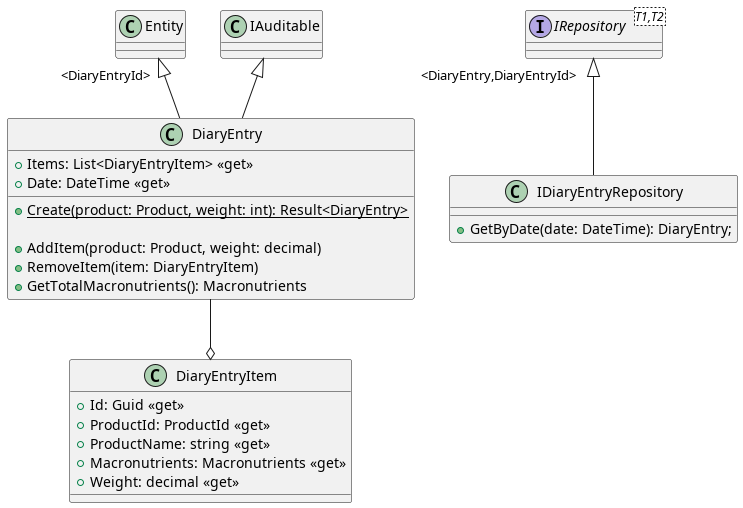


Рисунок В.1 – Диаграмма классов сервиса дневника питания

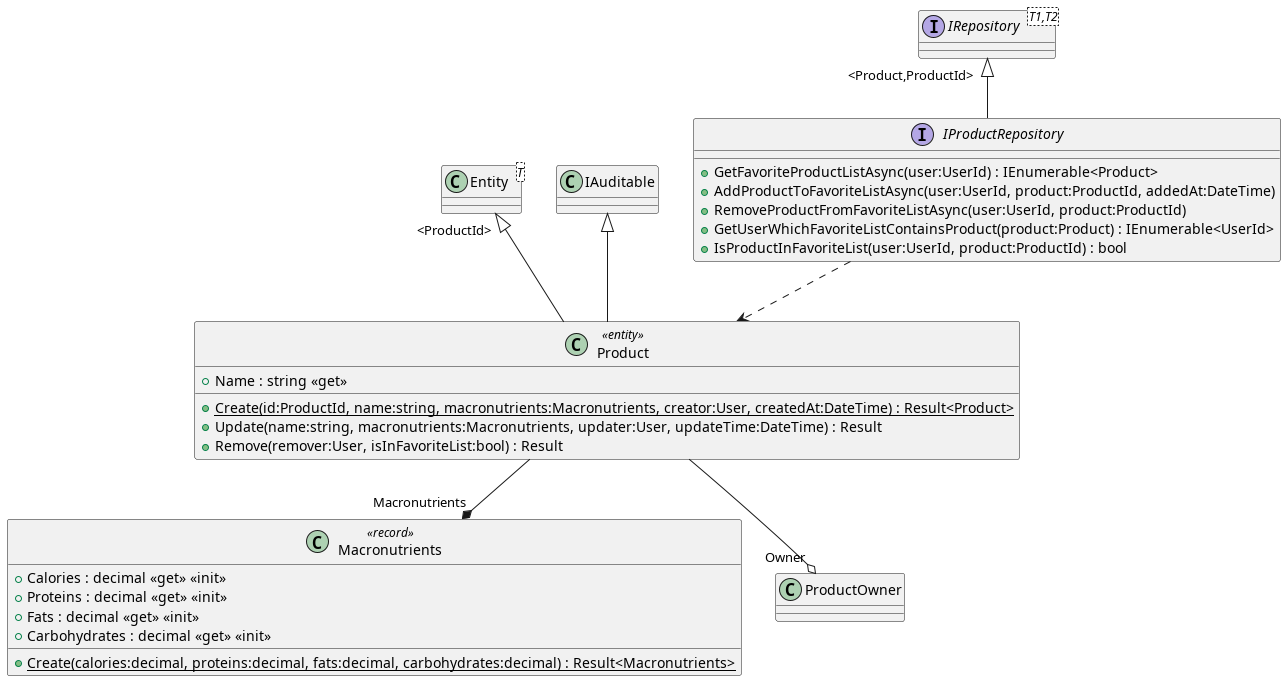


Рисунок В.2 – Диаграмма классов сервиса продукта

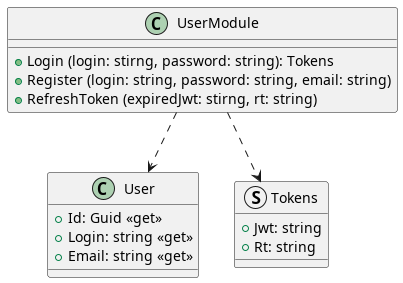


Рисунок В.3 – Диаграмма классов сервиса авторизации

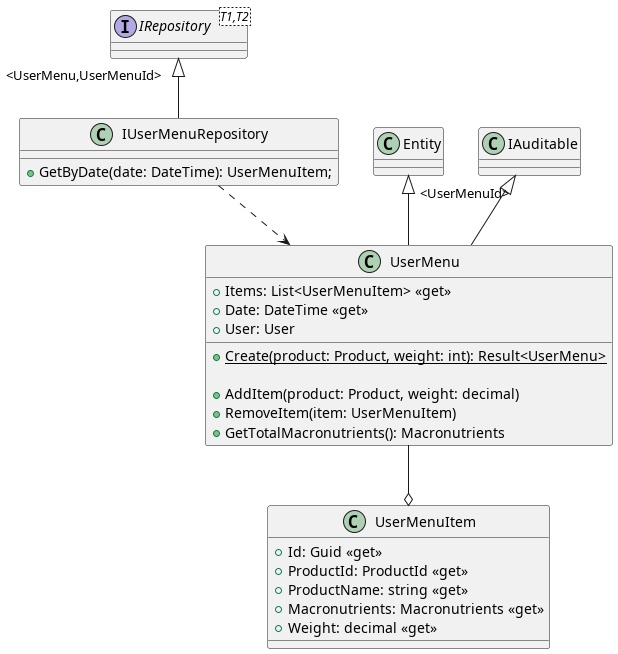


Рисунок В.4 – Диаграмма классов сервиса меню пользователя

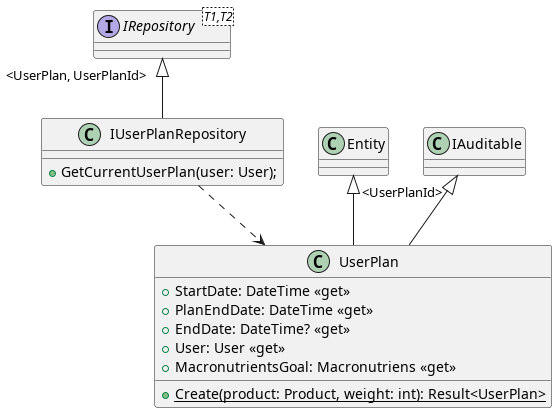


Рисунок В.5 – Диаграмма классов плана пользователя