Петрозаводский государственный университет
Институт математики и информационных технологий
Кафедра Информатики и математического обеспечения

Направление подготовки бакалавриата

09.03.04 Программная инженерия

Профиль направления подготовки бакалавриата

“Системное и прикладное программное обеспечение”

Отчет о проекте по дисциплине «Разработка для мобильных ОС»

Разработка фитнес-приложения для iOS на фреймворке SwiftUI

Выполнила:
студентка 2 курса группы 22207

А. О. Сергеева _____
подпись

Руководитель:
В.М. Димитров, старший преподаватель

_____
подпись

Итоговая оценка

_____
оценка

Петрозаводск — 2023

Содержание

Введение

В течение осеннего семестра 2023-2024 г. изучалась дисциплина «Разработка для мобильных ОС». В ходе изучения мной были выполнены 4 лабораторных работы для ознакомления с процессом программирования прикладного ПО для мобильных устройств. Главной **целью** итогового проекта является **закрепление полученных навыков**.

В последние годы в мире заметна положительная тенденция обеспокоенности населения своим здоровьем: с каждым днём всё большее количество людей начинает следить за своим питанием и вести здоровый образ жизни. В связи с этим появилась потребность в вспомогательных инструментах. В современном мире, где мобильные устройства являются неотъемлемой частью жизни человека, самым удобным решением было создание мобильного приложения, помогающего отслеживать свои привычки и соблюдать ЗОЖ. На сегодняшний день существует множество приложений, решающих эту проблему, например, YAZIO, FatSecret и др., и их количество тоже растёт, как и спрос на них.

Выбор темы объясняется её актуальностью, а также тем, что сфера мне близка и интересна.

Стек технологий

- среда разработки Xcode 15.0.1;
- язык программирования Swift;
- фреймворк SwiftUI

План проекта

В проекте планируется реализовать следующий функционал:

1. Создание пользовательского пространства, содержащего:
    - текущий рост (вводится пользователем);
    - текущий вес (вводится пользователем);
    - индекс массы тела (ИМТ) (рассчитывается программой)
2. Создание экрана отслеживания:
    - калькулятор потребленных калорий;
    - калькулятор сожженых калорий;
    - рекомендации по питанию

3. Создание экрана активности, состоящего из:

    - списка упражнений;
    - кнопки начала упражнений;
    - таймера

Потребленные калории считаются путём ввода пользователем съеденной за день еды (выбор из предопределенного списка блюд или ввод своего с указанием калорийности).
Потраченные калории включают в себя подсчёт шагов.
Планируется добавить опцию включения/выключения показа рекомендаций по питанию в личном пространстве пользователя.

1 Здоровый образ жизни. ИМТ

Индекс массы тела (ИМТ) — величина, позволяющая оценить степень соответствия массы человека и его роста и тем самым косвенно судить о том, является ли масса недостаточной, нормальной или избыточной. Его расчет позволяет оценить физическое развитие, наличие проблем с весом и риск заболеваний. Лишний вес повышает вероятность развития диабета, сердечно-сосудистых и онкологических заболеваний, проблем с суставами. Недостаток массы тела приводит к снижению иммунитета, частым заболеваниям легких, нарушению менструального цикла. Поэтому ИМТ широко используется в медицине для оценки риска хронических заболеваний. Также он нашел свое в применение в диетологии и фитнес-индустрии.

Формула расчёта ИМТ проста: нужно разделить значение своего веса в килограммах на рост, выраженный в квадратных метрах. Её вывел бельгийский математик Адольф Кетле. Согласно рекомендациям ВОЗ, значения ИМТ делятся на следующие категории:

Индекс массы тела	Соответствие между массой человека и его ростом
16 и менее	Выраженный дефицит массы тела
16—18,5	Недостаточная (дефицит) масса тела
18,5—25	Норма
25—30	Избыточная масса тела (предожирение)
30—35	Ожирение 1 степени
35—40	Ожирение 2 степени
40 и более	Ожирение 3 степени

Рис. 1 – Таблица ИМТ

Формулу можно применять только для ориентировочной оценки физического состояния. Причин этому несколько:

- индекс не учитывает массу мышц;
- показатель не отражает распределение жировой ткани;
- значение может повышаться при отеках;

Не учитываются и многие другие факторы: возраст, пол, особенности обмена веществ, физическая активность. Все это говорит о том, что ИМТ нельзя рассматривать как единственный индикатор здоровья.

Индекс хоть и не годится для постановки точного диагноза, но может помочь при коррекции массы тела. По нему можно определить точку отсчета и прогресс изменений:

- если показатель меньше 18,5, рекомендуется повысить калорийность суточного рациона на 300-500 ккал;
- при ИМТ свыше 25 количество потребляемых килокалорий в день необходимо уменьшить на 500;
- при значениях более 40 врач может порекомендовать снижение калорийности на 700 ккал и выше.

Также можно подобрать объем физической активности.

2 Архитектура приложения

2.1 Язык программирования Swift

Swift — открытый компилируемый язык программирования общего назначения, разработанный и поддерживаемый компанией Apple. Один и тот же код может быть скомпилирован для различных платформ: x86, ARM, WASM и других. Набор инструментов для работы с языком встроен в интегрированную среду разработки Xcode 6 и выше. Swift совместим с Objective-C и C, что делает возможным использование всех трёх языков в рамках одного проекта.

Swift заимствовал довольно многое из Objective-C, который использовался раньше, однако он определяется не указателями, а типами переменных, которые обрабатывает компилятор. По сравнению с Objective-C, Swift более быстрый, простой и устойчивый к ошибкам. Несмотря на то, что язык разработан компанией Apple и в основном предназначен для разработки приложений для macOS, iOS, iPadOS, watchOS и tvOS, он также доступен для Windows и Linux.

Из-за особенностей устройств Apple и их операционной системы для них нужно писать специальный код. В iOS-разработке Swift считается стандартом.

На Swift полностью или частично написано множество мобильных версий приложений и сервисов для iOS, например Firefox, YouTube, WordPress, Uber.

2.2 Описание архитектуры

Базовая архитектура приложения на Swift для iOS с использованием фреймворка SwiftUI включает в себя два файла: <Name>App.swift и ContentView.swift, где <Name> — название проекта, заданное при создании (подробнее о создании проекта см. в главе 3). Интерфейс реализуется за счёт представлений — View. В процессе реализации проекта планируется разработать несколько модулей:

- ContentView.swift — реализация основного представления приложения (переключение между вкладками) и приветственного окна;
- Windows.swift — реализация вкладки профиля, информационного окна и окна редактирования;
- Excercises.swift — реализация экрана активности: вспомогательные структуры, представление со списком упражнений и экран с таймером;

- CounterView.swift — вспомогательные структуры, главное представление экрана отслеживания, представление шкалы прогресса, рекомендационный раздел, экраны добавления калорий;
- Customize.swift — изменения во внешнем виде некоторых элементов интерфейса.

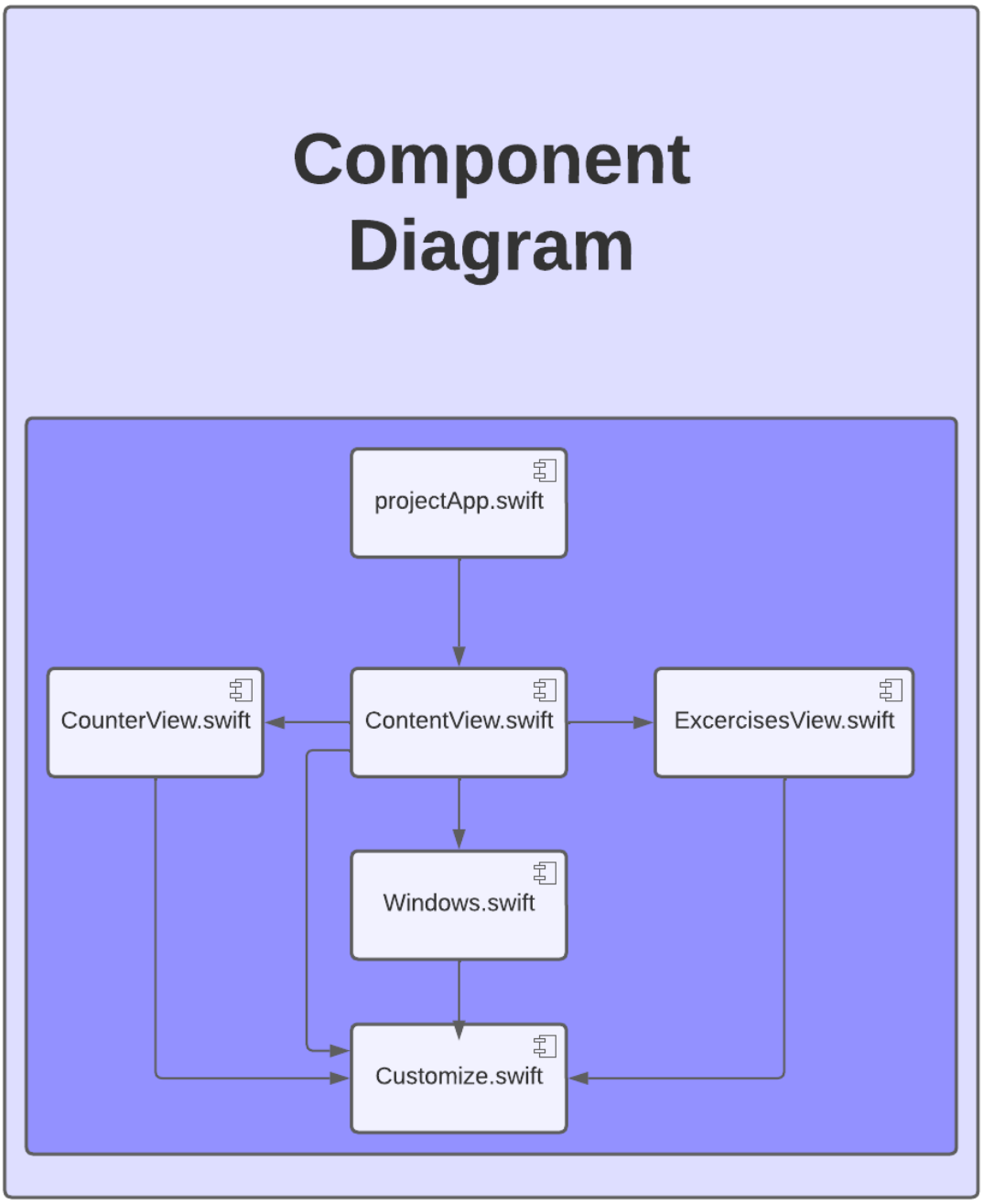

Рис. 2 – Диаграмма компонентов

Для работы с постоянным хранилищем данных приложения используется класс UserDefaults. UserDefaults предоставляет простой способ сохранения и получения данных, но не является подходящим для хранения больших объемов данных или чувствительной информации,

поскольку загружается при запуске приложения, и это может повлиять на время запуска. Взаимодействовать с UserDefaults можно через использование напрямую или с помощью свойства-обертки AppStorage.

Все представления — реализации пользовательского интерфейса — являются структурами, реализующими протокол **View**. Помимо структур-представлений, для хранения данных планируется использовать другие классы и структуры:

- структуры, реализующие протоколы **Codable** и **Hashable**, для хранения информации об упражнении, блюде, потребленных калориях. Протоколы Codable и Hashable необходимы для работы с локальным хранилищем
- классы, реализующие протоколы **ObservableObject**, для хранения информации о всех упражнениях, блюдах, калориях. Протокол ObservableObject необходим для доступа к публичным переменным (Published) — атрибутам класса — из различных структур к одному и тому же объекту

Для кастомизированных объектов в SwiftUI можно использовать расширения (**extention**) — механизмы, которые позволяют добавлять новые функциональности существующим типам данных, таким как классы, структуры, перечисления или протоколы, без изменения их первоначальной реализации.

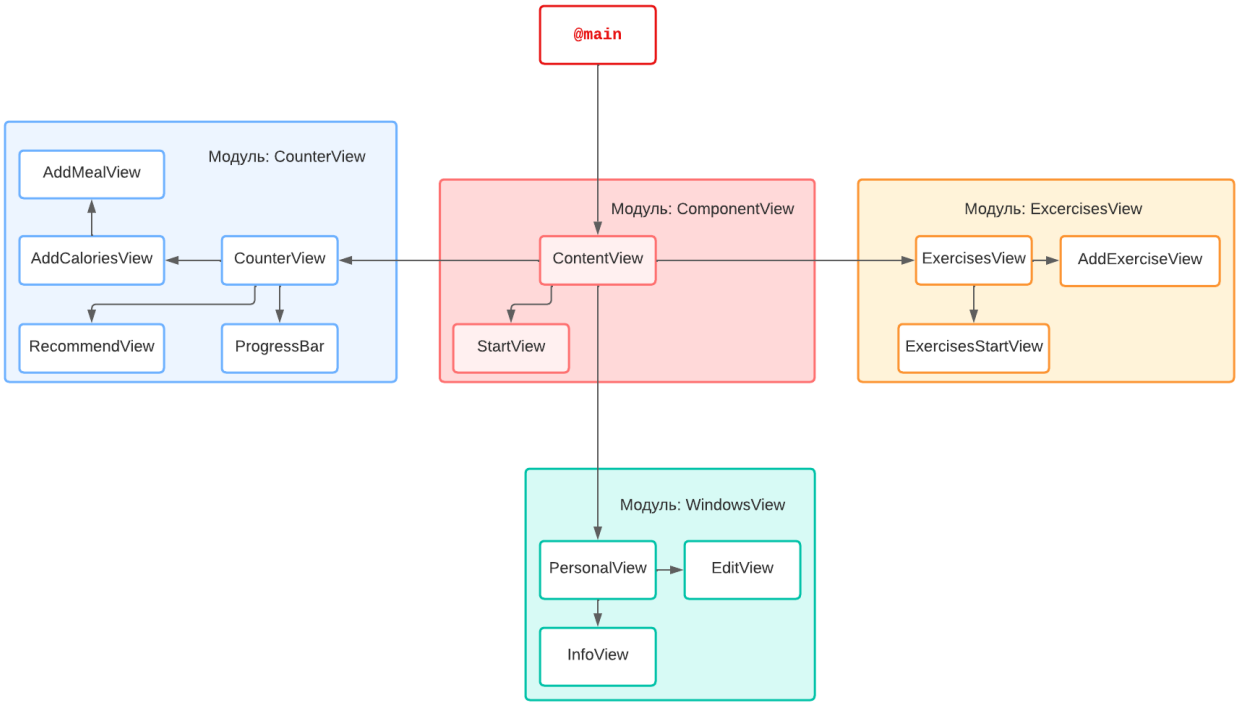

Рис. 3 – Взаимодействие представлений

3 Процесс разработки

3.1 Создание нового проекта

Для начала работы требуется создать проект в среде разработки Xcode[1] (см. рис. 4). Это делается путём нажатия кнопки "Create New Project" при запуске самой IDE. После этого можно выбрать платформу, тип проекта (для мобильной разработки — iOS-приложение) (см. рис. 5) и задать настройки (название проекта, разработчик, язык, фреймворк) (см. рис. 6). После создания проекта в нём уже будет автоматически создана его архитектура и написан примитивный код ("Hello, world!").

Рис. 4 – Стартовый экран среды Xcode

[1]Xcode — интегрированная среда разработки программного обеспечения для платформ macOS, iOS, watchOS и tvOS, разработанная корпорацией Apple.

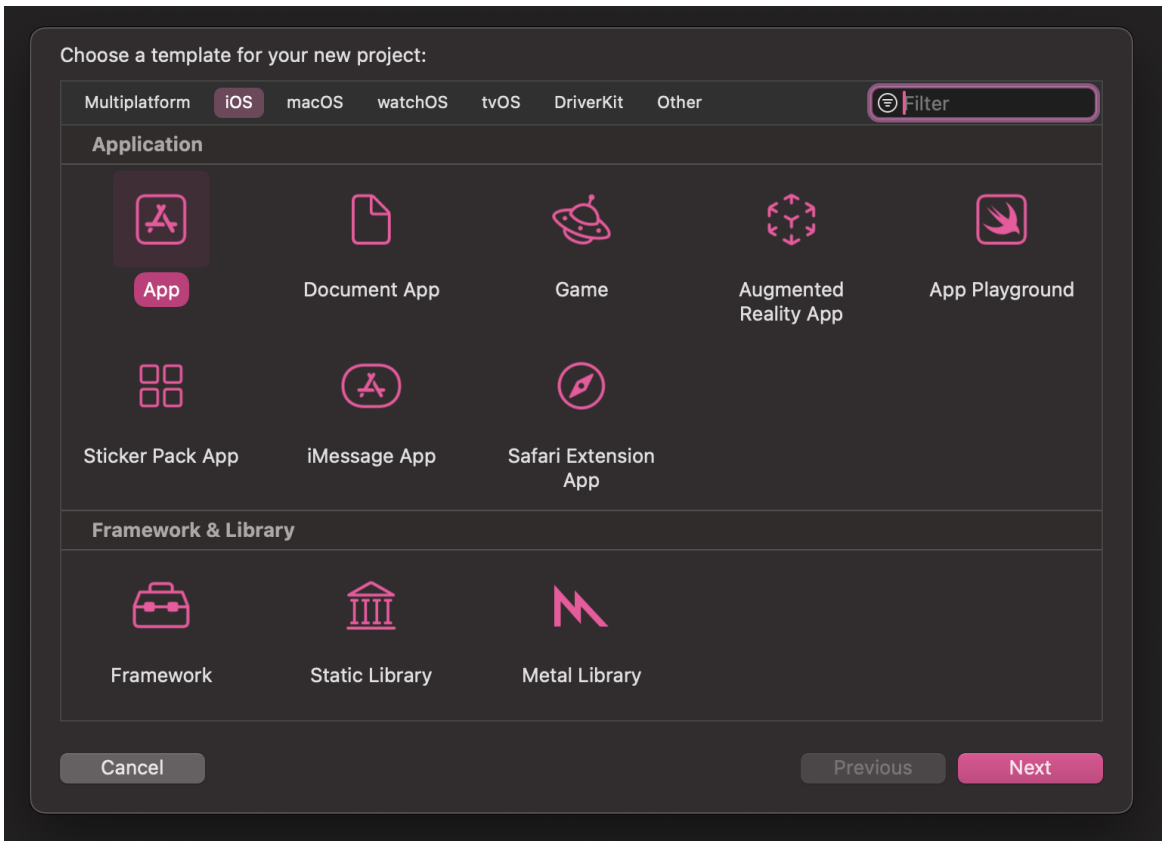

Рис. 5 – Выбор платформы

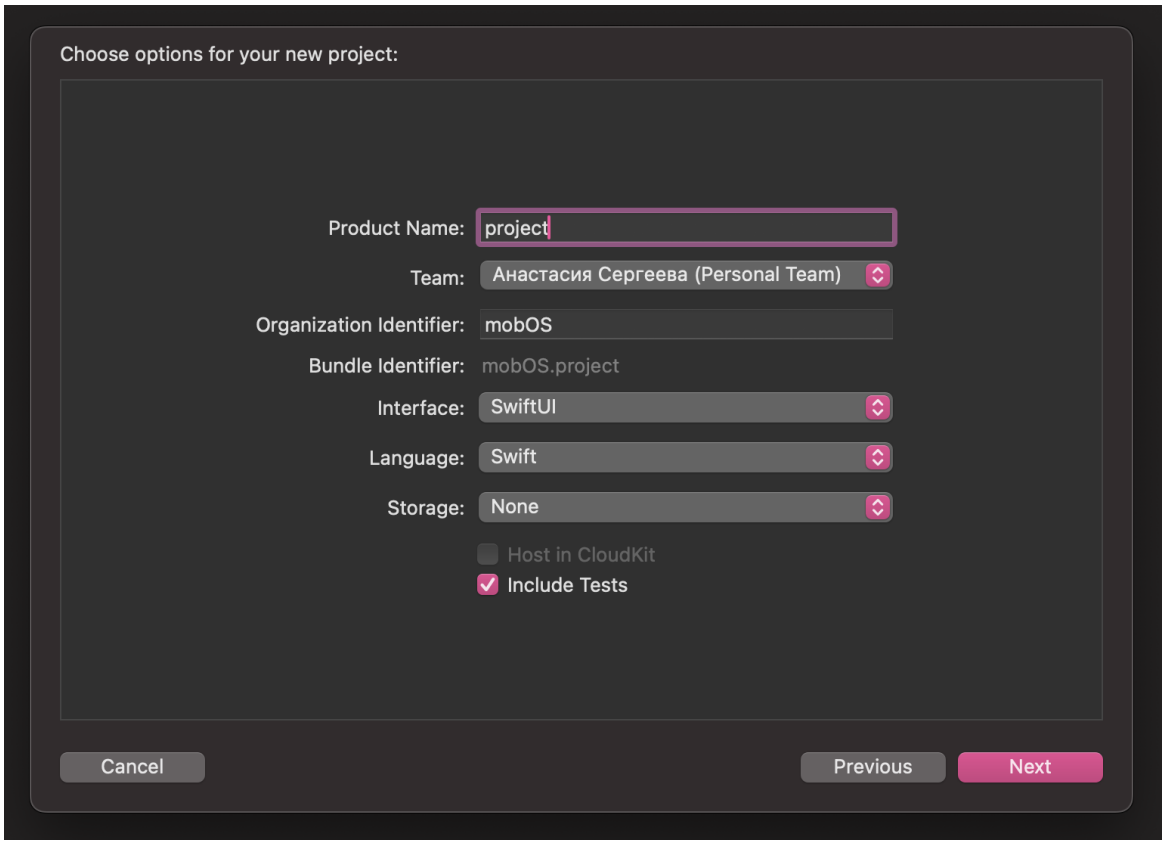

Рис. 6 – Установка настроек

3.2 Разработка модуля ContentView

3.2.1 Приветственное окно

При первом входе в приложение пользователю будет предложено заполнить информацию о себе. Для графической реализации сбора данных была создана структура **StartView**. Стандартная структура, реализующая протокол View, содержит в себе переменную body типа some View[2].

Для проверки первого посещения в свойствах[3] структуры был добавлен AppStorage атрибут с ключом isFirstLaunch типа Bool. Также были добавлены переменные локального хранилища username, userage, userweight, userheight, recommend, heightUnits, weightUnits с одноименными ключами для хранения информации о пользователе.

Для хранения введённых данных были добавлены массив строк inputs, опции радиокнопок. Также в свойствах определены переменные состояния selectedWeight, selectedHeight, weights, heights.

Ввод строк реализован через TextField с привязкой к строке. Для полей, ожидающих только числовых значений, установлена клавиатура из цифр с помощью `.keyboardType(.numberPad)` Расположение объектов на экране реализовано через стопки (VStack, HStack), Spacer'ы. Поля ввода (TextField) появляются по мере ввода пользователем других данных и нажатии на кнопку "Далее"

Если пользователь заполнил не все поля, он получает предупреждение (реализовано с помощью **alert**).

Для реализации кнопок была написана дополнительная структура — RadioButtonView. Внешний вид кнопки обеспечены фигурой Rectangle с закругленными через `.cornerRadius` углами. Размеры прямоугольника регулируются через `.frame`

[2]Возвращаемый тип — не обязательно View, но и всё, что поддерживает этот протокол. Таковыми являются Text, Image, VStack и т. д.

[3]Свойствами структуры называют блок, идущий перед body.

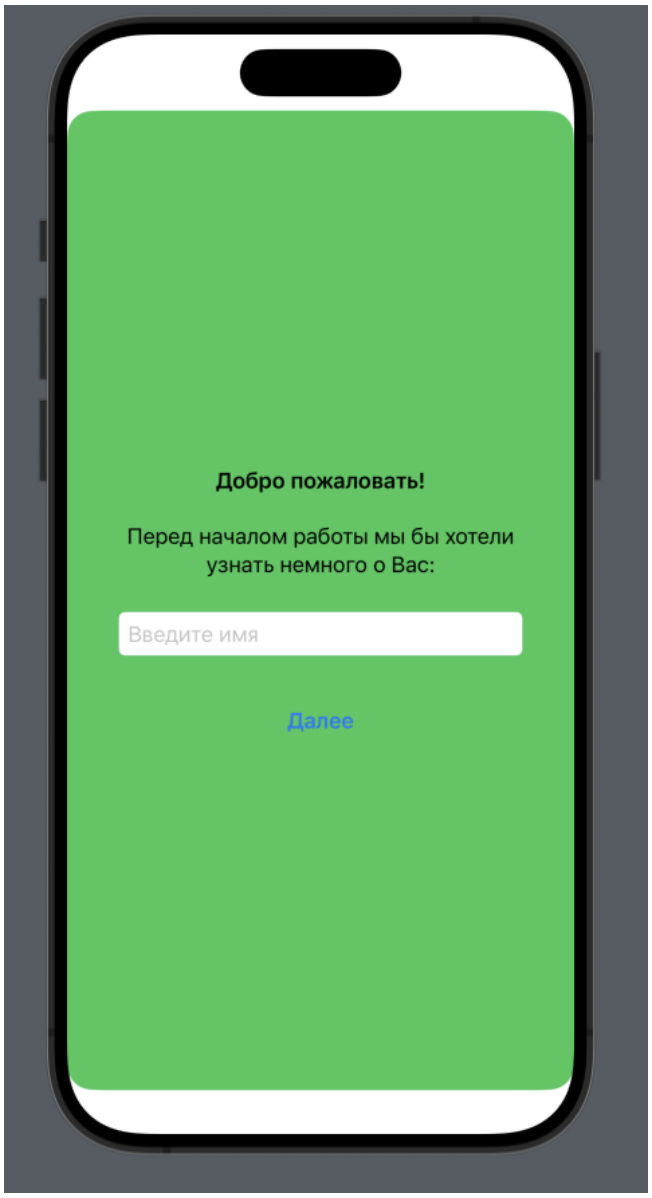

Рис. 7 – Начальная версия приветствия

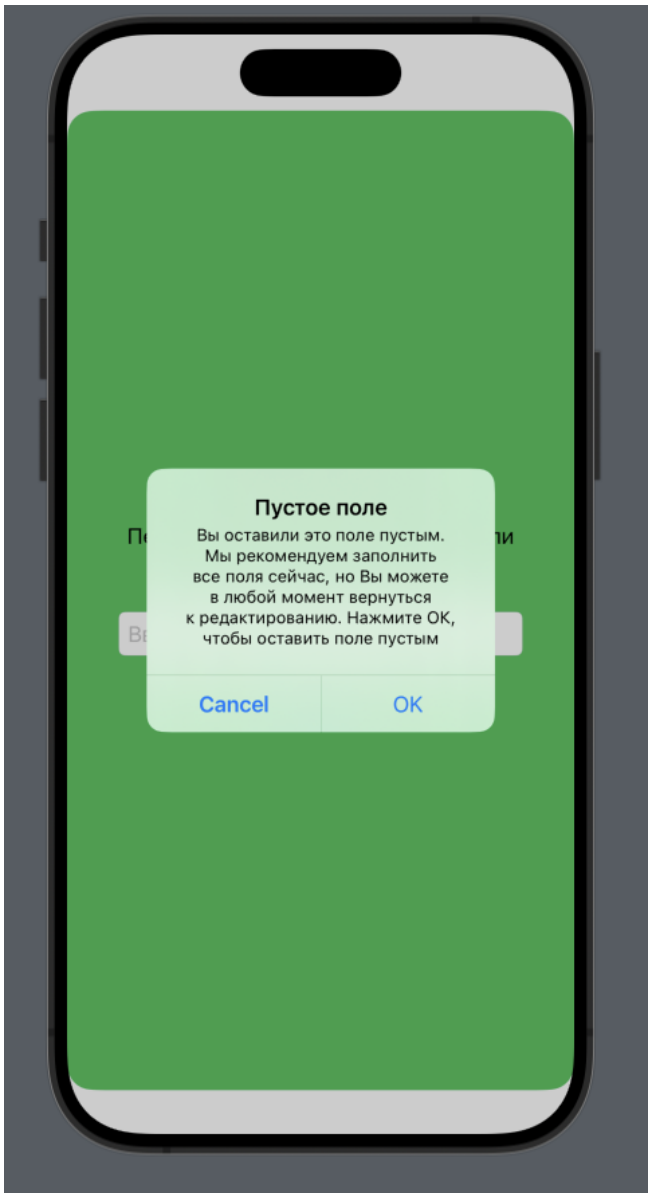

Рис. 8 – Предупреждение

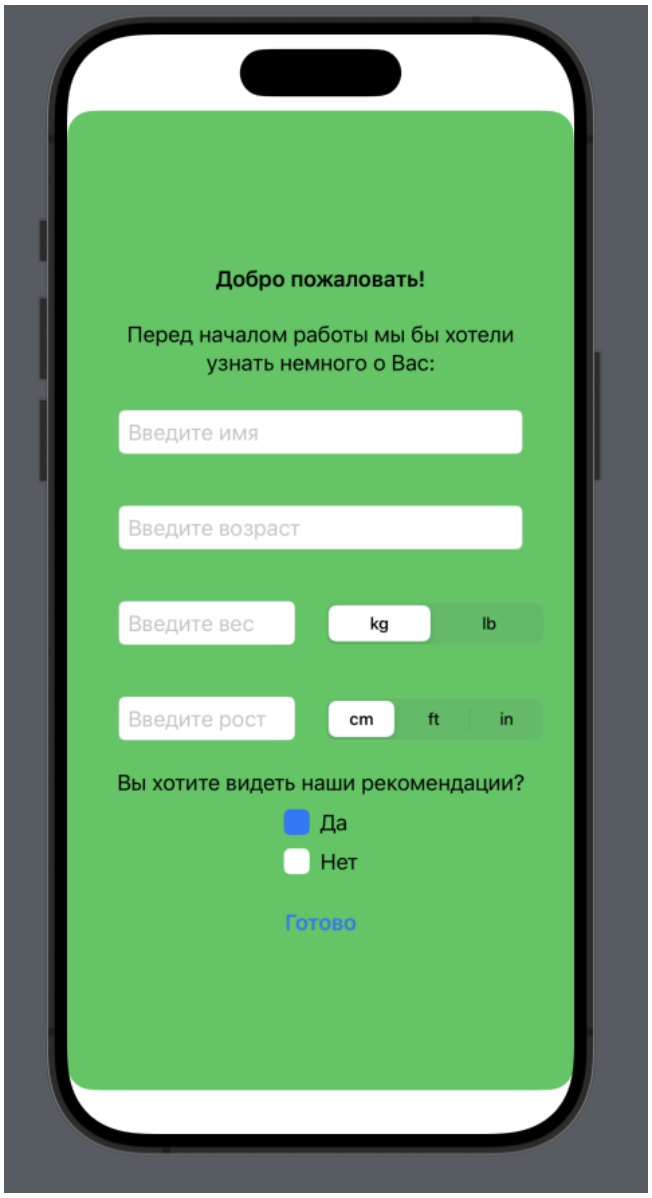

Рис. 9 – Все поля приветственного экрана

3.2.2 Главное представление

Структура **ContentView** представляет собой нижнюю панель для переключения между вкладками. Она реализована с помощью TabView, который содержит три кнопки для трех представлений. Также к структуре привязано приветствие через `.sheet`. Этот объект принимает параметр isPresented — привязываемую логическую переменную-флаг, отображающую состояние показа окна.

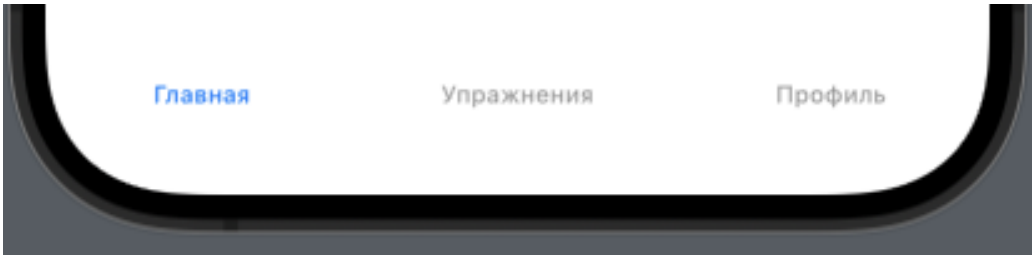

Рис. 10 – TabView

3.3 Разработка модуля WindowsView

3.3.1 Профиль пользователя

В свойствах структуры **PersonalView** были добавлены все переменные данных о пользователе из локального хранилища приложения. Также были описаны переменные состояния показа информационного окна и экрана редактирования.

Была описана функция getBMI, возвращающая Double переменную — высчитанный индекс массы тела по формуле, отталкиваясь от данных пользователя. Все данные приводятся к нужным единицам измерения (кг, м), передаются в формулу и результат округляется до сотых.

Расположение элементов на экране обеспечено так же стопками, Spacer'ами и LazyHGrid, который создаёт сетку по передаваемому шаблону. Элементы сетки обрабатываются циклом ForEach. Устанавливается цвет текста ИМТ в зависимости от его диапазона (см. рис. 1).

Через `.sheet` установлены привязки к информационному окну и экрану редактирования.

3.3.2 Информационное окно

В свойствах структуры **InfoView**определена Binding-переменная **isPresented** — флаг состояния отображения окна.

Text-элементами отображена информация о индексе массы тела. Через сетку LazyHGrid реализована таблица диапазонов ИМТ (см. рис. 1).
Добавлена кнопка, переводящая флаг isPresented в состояние false.

3.3.3 Экран редактирования

В свойствах структуры **EditView** были добавлены все переменные данных о пользователе из локального хранилища приложения. Также были описаны переменные состояния вводимого текста.
В методе **init()** присваиваются значения переменным состояния пользовательских данных.
Через элементы **Form, Section** внутри NavigationView на экране отображены поля ввода изменённой информации о пользователе. Добавлена кнопка, нажатие на которую сохраняет введённые данные в локальном хранилище.

3.4 Разработка модуля CounterView

Перед началом разработки представлений были созданы вспомогательные классы и структуры.

- структура **ConsumedCalories** — информация о потреблённых калориях. Атрибуты:
  - **count: Int** — общее количество
  - **proteins: Int** — масса белков
  - **fats: Int** — масса жиров
  - **carbohydrates: Int** — масса углеводов
- класс **Calories** — информация о потреблённых и потраченных калориях. Класс содержит метод инициализации, который кодирует и сохраняет эту информацию в UserDefaults, а также присваивает начальные значения атрибутам. Атрибуты:
  - **@Published var consumed: ConsumedCalories** — информация о потреблённых калориях
  - **@Published var spent: Int** — информация о потраченных калориях
- структура **Meal** — информация о блюде. Атрибуты:
  - **name: String** — наименование

  - **calories: ConsumedCalories** — калорийность

- класс **Meals** — список блюд. Класс содержит метод инициализации, который присваивает начальные значения атрибутам. Атрибуты:

  - **@Published var list: [Meal]** — список блюд

В структуре **CounterView** расположены основные элементы счётчика: информация о потреблённых калориях, информация о потраченных калориях, информация о пройденных шагах. Все три блока отображены на экране за счёт комбинирования стопок VStack, HStack и Spacer'ов. В блоке потреблённых калорий расположена кнопка, активирующая показ представления **AddCaloriesView**. В этой структуре показан список (List) предопределённых блюд, реализован поиск по названиям и переход на добавление собственного блюда — **AddMealView** структуру.
Добавление собственного блюда предполагает заполнение всех информационных полей, поэтому при оставлении пустой строки пользователь получит уведомление о необходимости ввода всех данных через Alert.
Структура **RecommendView** отображается по желанию пользователя. В структуре представлены рекомендации для пользователя. Изменения в этом представлении со стороны пользователя невозможны.

3.5 Разработка модуля ExercisesView

Перед началом разработки представлений были созданы вспомогательные классы и структуры.

- структура **Excercise** — информация об упражнении. Атрибуты:

  - **name: String** — наименование

  - **duration: Int** — продолжительность (в секундах)

- класс **Exercises** — информация об упражнениях. Класс содержит метод инициализации, который кодирует и сохраняет эту информацию в UserDefaults, а также присваивает начальные значения атрибутам.
  Атрибуты:

  - **@Published var list: [Exercise]** — список упражнений

  - **@Published var spent: Int** — информация о потраченных калориях

Методы:

- **saveData()** — сохраняет в UserDefaults при каждом изменении данные;
- **addExercise(name: String, duration: Int)** — добавляет к атрибуту list упражнение с наименованием name и продолжительностью duration.

3.5.1 Список упражнений

По умолчанию отображена структура **ExercisesView**. В свойствах структуры определен список упражнений exercises — ObservedObject, который отображен с помощью List и цикла ForEach. К каждому элементу прикреплена функция deleteItem, удаляющая элемент из списка. С помощью ссылок навигации (NavigationLink) выполнена привязка структур **AddExerciseView** и **ExercisesStartView**.

3.5.2 Добавление упражнения

3.5.3 Выполнение упражнений

3.6 Косметические дополнения

3.6.1 Установка дополнительных расширений и добавление объектов в проект

Для удобства добавления встроенных изображений в проект было установлено платное расширение SF Symbols, позволяющее быстро искать подходящую иконку и копировать её системное имя либо скачивать сет изображений в нужном формате. В установленном приложении можно было выбрать цвет изображения, толщину обводки а также тип.
Также был загружен собственный шрифт — **Comfortaa**. Для добавления шрифта в проект, в файле Info.plist необходимо было добавить пару ключ – значение:

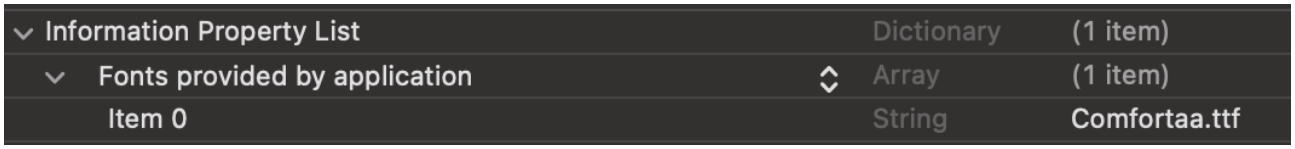

Рис. 11 – Info.plist

3.6.2 Разработка модуля Customize

В модуль Customize была вынесена структура RadioButtonView, а также добавлены два расширения:

1. **extension Color** — расширение, позволяющее не ограничиваться системными цветами, а задавать их в шестнадцатеричном формате. Оно реализует алгоритм сканирования строки в число, а затем разбивает на RGB-сегменты путём маскирования и сдвига битов.

2. **extension View** — расширение, устанавливающее шрифт Comfortaa для всех объектов customText.

После написания расширений большая часть объектов Text была заменена на customText, а также были изменены цвета.

Заключение

В результате проделанной работы был успешно освоен заявленный стек технологий, отработан навык разработки мобильных приложений для iOS с использованием современного фреймворка SwiftUI. План проекта был полностью выполнен, все поставленные задачи были решены.

Список литературы

1. Swift [Электронный ресурс]. / Apple Developer. — URL: https://developer.apple.com/documentation/swift (дата обращения: 15.01.2024)

2. Документация [Электронный ресурс]. / swiftbook. — URL: https://swiftbook.ru/content/docs/ (дата обращения: 27.12.2023)

3. Современные проблемы формирования здорового образа жизни студенческой молодежи : материалы II Международной научно-практической интернет-конференции, 10–12 апреля 2019 г., Минск, Беларусь / БГУ, Фак. социокультурных коммуникаций, Каф. экологии человека ; [редкол.: И. В. Пантюк (отв. ред.) и др.]. – Минск : БГУ, 2019 г. – С. 236-241.