МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение

высшего образования

**«Северный (Арктический) федеральный университет имени М.В. Ломоносова»**

Высшая школа информационных технологий и автоматизированных систем

(наименование высшей школы / филиала / института / колледжа)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

|  |  |  |
| --- | --- | --- |
| По дисциплине | | Разработка мобильных приложений |
|  | | |
| На тему | Разработка мобильного приложения Water Mate для ОС Android на языке | |
| программирования Kotlin | | |
|  | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ф.И.О. обучающихся | Наименование направления подготовки | Курс | Группа | Ф.И.О. руководителя, должность |
| Архаров Никита Михайлович | 09.03.02 Информационные системы и технологии | 3 | 351018 | Латухина Екатерина Александровна, старший преподаватель кафедры ВиПМ |
| Громов Никита Андреевич | 09.03.02 Информационные системы и технологии | 3 | 351018 | Латухина Екатерина Александровна, старший преподаватель кафедры ВиПМ |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Отметка о зачете |  |  |  |  |
|  |  | (отметка прописью) |  | (дата) |
| Руководитель |  |  |  | Е. А. Латухина |
|  |  | (подпись руководителя) |  | (инициалы, фамилия) |

Архангельск 2023

Оглавление

[Введение 3](#_Toc138338260)

[1 Описание предметной области и постановка задачи 4](#_Toc138338261)

[1.1 Описание предметной области и постановка задачи 4](#_Toc138338262)

[1.2 Анализ требований к приложению 4](#_Toc138338263)

[1.3 Описание идеи и состава приложения 5](#_Toc138338264)

[1.4 Сравнительный анализ существующих мобильных приложений 6](#_Toc138338265)

[2 Разработка мобильного приложения Water Mate 11](#_Toc138338266)

[2.1 Выбор программных средств 11](#_Toc138338267)

[2.2 Разработка прототипа 12](#_Toc138338268)

[2.3 Проектирование 17](#_Toc138338269)

[2.4 Интерфейс 19](#_Toc138338270)

[2.5 Разработка функционала 20](#_Toc138338271)

[2.6 Тестирование 24](#_Toc138338272)

[Заключение 26](#_Toc138338273)

[Список используемых источников 27](#_Toc138338274)

[Приложение А. Прототип приложения 28](#_Toc138338275)

[Приложение Б. Листинг кода курсовой работы 29](#_Toc138338276)

Введение

В современном мире здоровье и благополучие каждого человека становятся все более приоритетными ценностями. Оптимальное функционирование организма требует поддержания гидратации и уровня электролитов внутри оптимальных пределов. Однако, недостаток воды и дисбаланс электролитов могут привести к серьезным проблемам со здоровьем, включая дегидратацию различной степени тяжести.

С целью помощи в диагностике и коррекции дегидратации и электролитного баланса, нашей командой из двух человек было разработано мобильное приложение под названием Water Mate. Это приложение предоставляет удобный и эффективный инструмент для определения степени дегидратации и необходимой коррекции электролитов.

Основной целью данной работы является детальное описание разработанного мобильного приложения Water Mate и его функциональности. Мы также поставили перед собой следующие задачи:

- разработать пользовательский интерфейс приложения, обеспечивающий удобство использования и понятность;

- реализовать механизм ввода и обработки персональных данных пользователей, необходимых для расчета степени дегидратации и коррекции электролитов;

- создать модели данных для представления степени дегидратации, коррекции электролитов и определения необходимого объема воды для пациентов различного возраста и веса;

- разработать алгоритмы расчета и вывода рекомендаций по дегидратации, электролитам и уровню потребления воды;

- построить файловую структуру приложения, учитывающую модели данных и пользовательский интерфейс.

**1 Описание предметной области и постановка задачи**

* 1. Описание предметной области и постановка задачи

Предметной областью данной работы является профилактика и контроль дегидратации с помощью мобильных приложений.

Для профилактики и контроля дегидратации необходимо регулярно пить достаточное количество воды в соответствии с индивидуальными потребностями организма. Однако многие люди забывают или игнорируют свою жажду, не замечают признаков дегидратации или не имеют доступа к качественной питьевой воде. В этих случаях мобильные приложения могут помочь отслеживать уровень гидратации, напоминать о необходимости пить воду и рассчитывать оптимальную суточную дозу воды.

### Анализ требований к приложению

Для разработки мобильного приложения по профилактике и контролю дегидратации необходимо определить функциональные и нефункциональные требования к нему. Функциональные требования описывают основные возможности приложения, которые должны удовлетворять потребности пользователей. Нефункциональные требования описывают характеристики качества приложения, такие как производительность, безопасность, удобство использования и другие.

Функциональные требования к приложению:

- приложение должно позволять пользователю вводить свои персональные данные, такие как пол, возраст, вес;

- приложение должно рассчитывать оптимальную суточную дозу воды для пользователя на основе его персональных данных и рекомендаций Всемирной организации здравоохранения (ВОЗ);

- приложение должно рассчитывать суточную дозу воды при дегидратации для пользователя на основе его персональных данных и степени дегидратации;

- приложение должно отслеживать уровень гидратации пользователя на основе его потребления воды и других жидкостей (соки, чай, кофе и другие).

Нефункциональные требования к приложению:

- приложение должно быть совместимо с мобильной платформой Android;

- приложение должно иметь простой и интуитивно понятный интерфейс, который обеспечивает удобство использования для пользователей разного возраста и уровня компьютерной грамотности;

- приложение должно обеспечивать безопасность и конфиденциальность персональных данных пользователей;

- приложение должно иметь высокую производительность и низкое потребление ресурсов мобильного устройства, обеспечивая быстрый и стабильный запуск и работу приложения;

- приложение должно иметь возможность работать в оффлайн-режиме.

1.3 Описание идеи и состава приложения

Идея приложения заключается в том, чтобы предоставить пользователю удобный и эффективный инструмент для профилактики и контроля дегидратации с помощью мобильного устройства.

Состав приложения будет включать следующие основные компоненты:

- ввод данных – позволяет пользователю вводить свои персональные данные, такие как возраст, вес, рост;

- расчет дозы воды – рассчитывает оптимальную суточную дозу воды для пользователя на основе его персональных данных и рекомендаций ВОЗ, а также суточную дозу воды при дегидратации на основе его персональных данных и степени дегидратации;

- оценка уровня дегидратации;

- советы по электролитам.

1.4 Сравнительный анализ существующих мобильных приложений

Для проведения сравнительного анализа были выбраны четыре мобильных приложения, решающих поставленную задачу: Waterllama, Water Time Drink Tracker & Reminder, HydroCoach и Plant Nanny. Для каждого приложения была приведена краткая характеристика и описание основных возможностей. Результаты сравнения занесены в таблицу 1.

Waterllama – это мобильное приложение, которое помогает пользователю пить достаточно воды каждый день. Приложение рассчитывает индивидуальную суточную дозу воды на основе данных пользователя. Приложение отслеживает потребление воды и других жидкостей, напоминает о необходимости пить воду, предлагает альтернативные источники жидкости и предупреждает о признаках дегидратации. Приложение также имеет игровой элемент: пользователь может выбрать своего виртуального питомца-ламу, которая будет мотивировать его пить воду и реагировать на его действия.

Water Time Drink Tracker & Reminder – это мобильное приложение, которое помогает пользователю контролировать свой уровень гидратации. Приложение рассчитывает оптимальную суточную дозу воды на основе данных пользователя и его активности. Приложение отслеживает потребление воды и других напитков, напоминает о необходимости пить воду, предлагает полезные советы по гидратации и предупреждает о признаках дегидратации. Приложение также имеет статистику потребления воды и уровня гидратации в виде графиков и диаграмм.

HydroCoach – это мобильное приложение, которое помогает пользователю пить достаточно воды для поддержания здоровья. Приложение рассчитывает индивидуальную суточную дозу воды на основе данных пользователя и его целей (похудение, повышение иммунитета и другие). Приложение отслеживает потребление воды и других напитков, напоминает о необходимости пить воду, предлагает разнообразные рецепты напитков и предупреждает о признаках дегидратации. Приложение также имеет статистику потребления воды и уровня гидратации в виде графиков и диаграмм.

Plant Nanny – это мобильное приложение, которое помогает пользователю пить достаточно воды с помощью виртуальных растений. Приложение рассчитывает оптимальную суточную дозу воды на основе данных пользователя и его активности. Приложение отслеживает потребление воды и других напитков, напоминает о необходимости пить воду, предлагает разнообразные рецепты напитков и предупреждает о признаках дегидратации. Приложение также имеет игровой элемент: пользователь может выбрать свое виртуальное растение, которое будет расти и цвести в зависимости от его потребления воды.

Таблица 1 – Сравнение приложений-конкурентов

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Мобильное приложение | Основной функционал | Отличительные особенности | Стоимость | Доступные платформы | Офлайн доступ | Реклама |
| Waterllama | Расчёт суточной дозы воды, отслеживание потребления воды и других жидкостей, напоминание о необходимости пить воду, предупреждение о признаках дегидратации. | Виртуальный питомец-лама, который мотивирует пользователя пить воду и реагирует на его действия. | Бесплатно | Android,  iOS | Да | Нет |
| Water Time Drink Tracker & Reminder | Расчёт суточной дозы воды, отслеживание потребления жидкости, напоминание о необходимости пить воду, полезные советы по гидратации, предупреждение о признаках дегидратации. | Статистика потребления воды и уровня гидратации в виде графиков и диаграмм. | Бесплатно | Android,  iOS | Да | Да |

Таблица 1 – Сравнение приложений-конкурентов (окончание)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Мобильное приложение | Основной функционал | Отличительные особенности | Стоимость | Доступные платформы | Офлайн доступ | Реклама |
| HydroCoach | Расчёт суточной дозы воды, отслеживание потребления воды и других напитков, напоминание о необходимости пить воду, разнообразные рецепты напитков, предупреждение о признаках дегидратации. | Статистика потребления воды и уровня гидратации в виде графиков и диаграмм. Возможность установить цели по гидратации (похудение, повышение иммунитета и другие). | Бесплатно (премиум-версия за 4.99$) | Android,  iOS | Да | Да |
| Plant Nanny | Расчет суточной дозы воды, отслеживание потребления жидкости, напоминание о необходимости пить воду, предложение разнообразных рецептов напитков, предупреждение о признаках дегидратации. | Виртуальное растение, которое растет и цветет в зависимости от потребления воды пользователя. Возможность выбрать из разных видов растений. | Бесплатно (покупки внутри приложения) | Android,  iOS | Да | Нет |
| Water Mate | Вычисление степени дегидратации, расчет коррекции электролитов, расчет суточной потребности в жидкости, расчет потребности в жидкости для устранения дегидратации. | Простота интерфейса, направленность на работников медицинской сферы. | Бесплатно | Android | Да | Нет |

Сравнивая существующие мобильные приложения по профилактике и контролю дегидратации, можно сделать вывод, что они имеют общий функционал, но различаются по отличительным особенностям, стоимости и наличию рекламы. Некоторые приложения используют игровые элементы для мотивации пользователя пить воду (Waterllama, Plant Nanny), другие приложения предоставляют статистику и графики для визуализации уровня гидратации (Water Time Drink Tracker & Reminder, HydroCoach), а также возможность установить цели по гидратации (HydroCoach). Некоторые приложения бесплатные, но имеют рекламу (Water Time Drink Tracker & Reminder, HydroCoach), другие бесплатные, но имеют покупки внутри приложения (Plant Nanny), а одно приложение полностью бесплатное и без рекламы (Waterllama).

На основе сравнительного анализа можно сформулировать следующие преимущества и недостатки существующих мобильных приложений по профилактике и контролю дегидратации:

Преимущества:

- приложения помогают пользователю пить достаточно воды каждый день, учитывая его индивидуальные потребности и рекомендации ВОЗ;

- приложения отслеживают уровень гидратации пользователя на основе его потребления воды и других жидкостей, а также цвета мочи и других сигналов;

- приложения напоминают пользователю о необходимости пить воду с заданной периодичностью или по определенным сигналам, а также предлагают альтернативные источники жидкости;

- приложения предупреждают пользователя о признаках дегидратации и рекомендуют действия по ее устранению или обращению к врачу;

- приложения визуализируют статистику потребления воды и уровня гидратации пользователя в виде графиков, диаграмм и других.

- приложения используют игровые элементы для мотивации пользователя пить воду (Waterllama, Plant Nanny) или возможность установить цели по гидратации (HydroCoach).

Недостатки:

- приложения имеют рекламу (Water Time Drink Tracker & Reminder, HydroCoach) или покупки внутри приложения (Plant Nanny), которые могут раздражать пользователя или ограничивать его доступ к полному функционалу приложения.

Исходя из всего вышесказанного можно сделать вывод о том, что такое приложение как Water Mate нуждается в разработке, так как у людей нет инструментов для определения степени дегидратации, ее устранения и коррекции электролитов.

**2 Разработка мобильного приложения Water Mate**

2.1 Выбор программных средств

Для разработки приложения Water Mate были выбраны программные средства, описанные ниже.

Средство прототипирования. Для создания прототипа интерфейса приложения был выбран онлайн-сервис Figma. Figma позволяет создавать интерактивные прототипы с различными элементами дизайна, а также совместно работать над проектом в режиме реального времени. Figma имеет бесплатный тарифный план для индивидуальных пользователей и небольших команд, а также интегрируется с другими сервисами, такими как Slack и GitHub. В таблице 2 представлено сравнение Figma с другими популярными средствами прототипирования: Adobe XD и Sketch.

Таблица 2 – Сравнение средств прототипирования

|  |  |  |  |
| --- | --- | --- | --- |
| Критерий | Figma | Adobe XD | Sketch |
| Цена | Бесплатно для 3 проектов и 2 редакторов | Бесплатно для 1 проекта и 2 ГБ хранилища | $99 в год за лицензию |
| Платформа | Веб-приложение | Windows, macOS | macOS |
| Интерактивность | Да | Да | Нет |
| Совместная работа | Да | Да | Нет |
| Интеграция с другими сервисами | Да | Да | Нет |

Исходя из сравнения, можно сделать вывод, что Figma является наиболее подходящим средством прототипирования для разработки приложения Water Mate.

Другие программные средства. Для разработки приложения Water Mate также были использованы следующие программные средства:

- IDE Android Studio. Android Studio — это официальная среда разработки для платформы Android, которая предоставляет все необходимые инструменты для создания, тестирования и отладки приложений [2]. Android Studio поддерживает язык программирования Kotlin [1], а также имеет множество плагинов и библиотек, которые упрощают разработку.

- Система контроля версий GitHub. GitHub — это популярный сервис для хранения и управления кодом, который использует систему контроля версий Git. GitHub позволяет отслеживать изменения в коде, совместно работать над проектом, а также интегрироваться с другими сервисами, такими как Figma и Slack.

2.2 Разработка прототипа

Приложение включает несколько экранов, каждый из которых играет важную роль в обеспечении полноценного функционала.

Первый экран - экран приветствия - позволяет ознакомиться с основными возможностями приложения, включая определение степени дегидратации, требуемое количество электролитов, ежедневную потребность в воде, а также рекомендации по коррекции дегидратации. Первый экран представлен на рисунке 1.

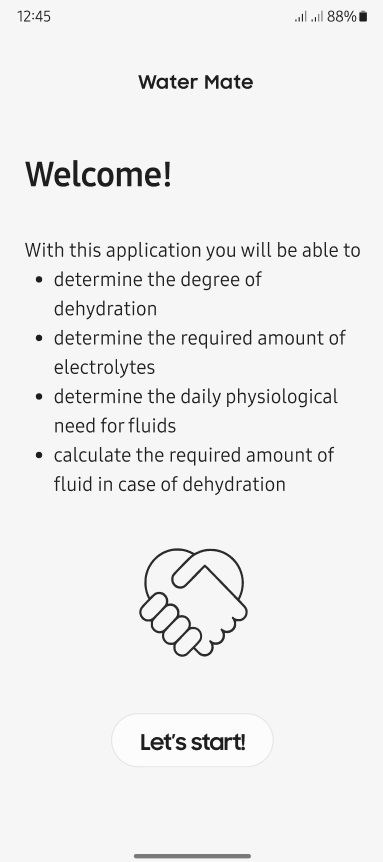


Рисунок 1 – Экран приветствия

Далее следуют четыре экрана, изображенные на рисунках 2-5: «Dehydration», «Electrolites», «Water», «Dehydration treatment», на которых пользователь вводит соответствующие данные, чтобы определить степень дегидратации, требуемое количество электролитов, ежедневную потребность в воде и рекомендации по устранению дегидратации. Заполняя эти формы, пользователь получает рассчитанные показатели, которые помогают ему контролировать свое гидратационное состояние и принимать соответствующие меры.

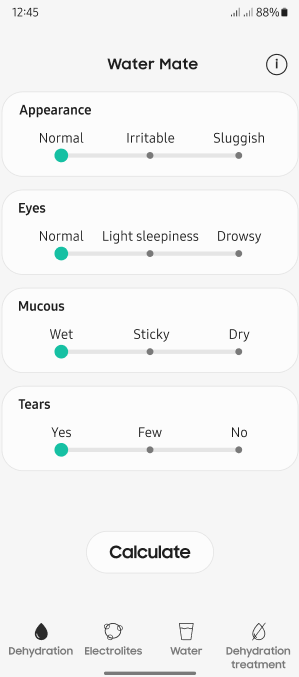


Рисунок 2 – Экран «Dehydration»

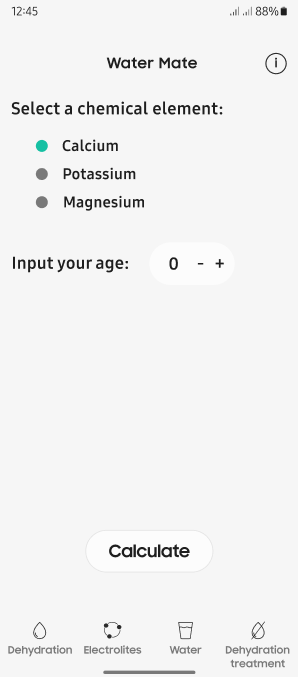


Рисунок 3 –Экран «Electrolites»

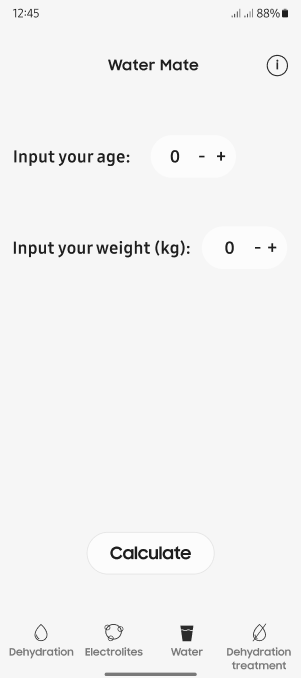


Рисунок 4 – Экран «Water»

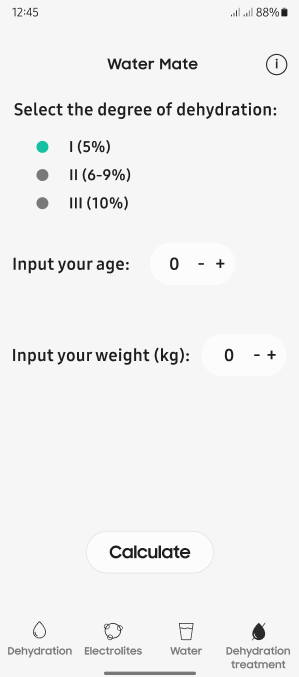


Рисунок 5 – Экран «Dehydration treatment»

Также, каждый из перечисленных четырех экранов, содержит экран показа ответа. Все они идентичны, за исключением содержащейся в них информации. На рисунке 6 приведен пример такого экрана.



Рисунок 6 – Пример экрана вывода информации

Такая структура экранов в приложении Water Mate обеспечивает логичную последовательность действий и предоставляет необходимую информацию для эффективного контроля гидратации и поддержания здоровья пользователей.

Кроме того, приложение предоставляет экран «О разработчиках», который содержит информацию о команде разработчиков. Это позволяет пользователям узнать больше о людях, стоящих за приложением, и создает доверие к его надежности и качеству. Экран показан на рисунке 7.

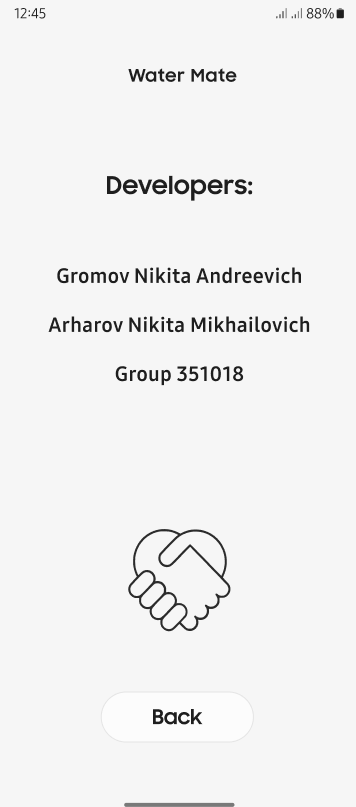


Рисунок 7 – Экран «О разработчиках»

2.3 Проектирование

Для разработки мобильного приложения Water Mate была выбрана архитектура MVVM (Model-View-ViewModel), которая позволяет разделить логику приложения на три слоя: модель, представление и модель представления.

Модель (Model) – это слой данных, который содержит бизнес-логику и состояние приложения. Модель отвечает за хранение, обработку и предоставление данных для других слоев. В проекте Water Mate модель состоит из классов, представляющих сущности, такие как User, Dehydration, Electrolyte и т.д., а также из классов-репозиториев, которые обеспечивают доступ к локальным или удаленным источникам данных.

Представление (View) – это слой пользовательского интерфейса, который отвечает за отображение данных на экране и взаимодействие с пользователем. Представление не содержит бизнес-логики и зависит только от модели представления. В проекте Water Mate представление состоит из активностей (Activity) и фрагментов (Fragment), которые содержат элементы управления (View), такие как кнопки, текстовые поля, списки и т.д.

Модель представления (ViewModel) – это слой посредника между моделью и представлением, который отвечает за связывание данных между ними. Модель представления получает данные от модели, преобразует их в формат, удобный для отображения на представлении, и обновляет представление при изменении данных. Также модель представления обрабатывает действия пользователя, передавая их модели или вызывая другие компоненты приложения. В проекте Water Mate модель представления состоит из классов ViewModel, которые содержат свойства (LiveData или Observable) [4], подписанные на данные модели, и методы для выполнения операций над данными.

Диаграмма классов, представленная на рисунке 8 показывает структуру классов в проекте Water Mate и связи между ними. На диаграмме изображены атрибуты и методы классов, а также виды связей: наследование, ассоциация, агрегация и композиция.

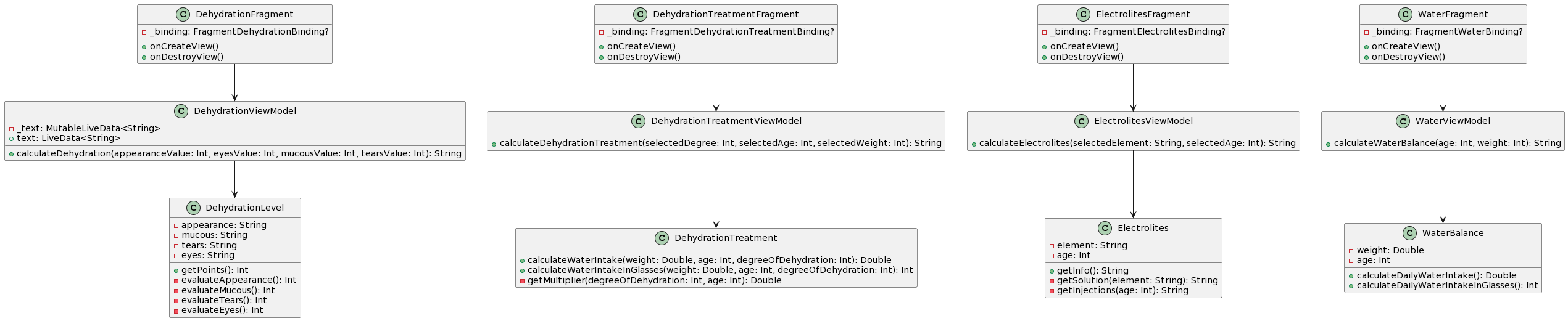


Рисунок 8 – Диаграмма классов

Диаграмма последовательности, представленная на рисунке 9 - показывает взаимодействие объектов во времени, то есть последовательность сообщений, передаваемых между объектами.

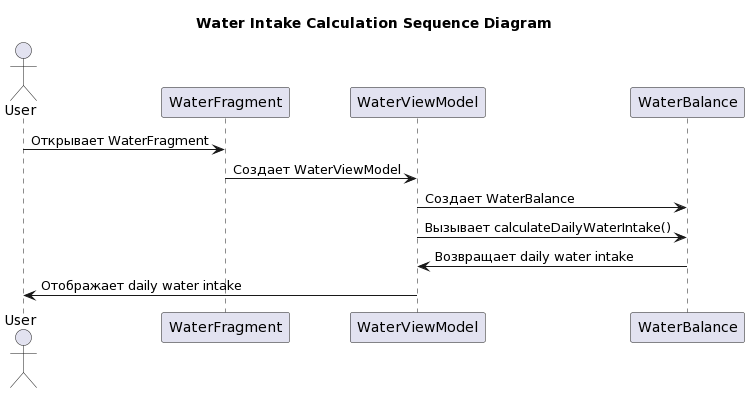


Рисунок 9 – Диаграмма последовательности

2.4 Интерфейс

Дизайн приложения незамысловатый, понятный любому пользователю. Простота дизайна обусловлена тем, что в приложении не должно быть ничего, что могло бы отвлекать пользователя от взаимодействия.

На каждом из экранов есть свой контент. Ниже приведено подробное описание содержимого каждого элемента отображения.

Экран «Dehydration». На этом экране пользователь вводит данные о степени дегидратации. Есть четыре формы ввода:

- Appearance: пользователь выбирает одно из трех состояний: Normal, Irritable, Sluggish.

- Eyes: пользователь выбирает одно из трех состояний: Normal, Light sleepiness, Drowsy.

- Mucous: пользователь выбирает одно из трех состояний: Wet, Sticky, Dry.

- Tears: пользователь выбирает одно из трех состояний: Yes, Few, No.

Экран «Electrolites». На этом экране пользователь выбирает элемент химического элемента и вводит свой возраст. Форма ввода состоит из двух полей:

- Select a chemical element: пользователь выбирает один из трех химических элементов: Calcium, Potassium, Magnesium.

- Input your age: пользователь вводит свой возраст в числовом формате.

Экран «Water». На этом экране пользователь вводит свой возраст и вес. Форма ввода состоит из двух полей:

- Input your age: пользователь вводит свой возраст в числовом формате.

- Input your weight: пользователь вводит свой вес в числовом формате.

Экран «Dehydration treatment» (Коррекция дегидратации). На этом экране пользователь выбирает степень дегидратации, вводит свой возраст и вес. Форма ввода состоит из трех полей:

- Select the degree of dehydration: пользователь выбирает одну из трех степеней дегидратации: I (5%), II (6-9%), III (10%).

- Input your age: пользователь вводит свой возраст в числовом формате.

- Input your weight: пользователь вводит свой вес в числовом формате, округляя до целых.

Каждый экран предоставляет пользователю возможность ввести необходимые данные для расчета соответствующих показателей. После нажатия кнопки «Calculate» на каждом экране открывается экран с рассчитанными результатами, отображающими информацию о степени дегидратации, уровне электролитов или рекомендованном потреблении воды.

2.5 Разработка функционала

Рассмотрим состав модулей, структуру классов, примеры программного кода реализации алгоритмов, описание ресурсов и скриншоты пользовательского интерфейса в режиме разработчика.

Модули приложения:

- «custom\_utils» – модуль, содержащий пользовательские утилиты, необходимые для настройки и адаптации элементов пользовательского интерфейса. В состав модуля входят следующие классы: CustomArrayAdapter.kt – пользовательский адаптер для работы с массивами данных, CustomSeekbar.kt – пользовательский виджет, реализующий функциональность ползунка с возможностью настройки параметров [3].

- «models» – модуль, содержащий модели данных, необходимые для работы с основными функциями приложения. В состав модуля входят следующие классы: DehydrationLevel.kt – модель для определения степени дегидратации, DehydrationTreatment.kt – модель для расчета рекомендуемого уровня воды при лечении дегидратации, Electrolites.kt – модель для работы с данными по электролитам, WaterBalance.kt – модель для расчета рекомендуемого уровня потребления воды.

- «ui» – модуль, содержащий пользовательский интерфейс приложения, разделенный на фрагменты. В состав модуля входят следующие классы: DehydrationFragment.kt – класс фрагмента, отвечающий за отображение и взаимодействие с формой ввода данных о дегидратации, DehydrationViewModel.kt – класс ViewModel, связывающий фрагмент DehydrationFragment с моделью DehydrationLevel, DehydrationTreatmentFragment.kt – класс фрагмента, отвечающий за отображение и взаимодействие с формой ввода данных о лечении дегидратации, DehydrationTreatmentViewModel.kt – класс ViewModel, связывающий фрагмент DehydrationTreatmentFragment с моделью DehydrationTreatment, ElectrolitesFragment.kt – класс фрагмента, отвечающий за отображение и взаимодействие с формой ввода данных об электролитах, ElectrolitesViewModel.kt – класс ViewModel, связывающий фрагмент ElectrolitesFragment с моделью Electrolites, WaterFragment.kt – класс фрагмента, отвечающий за отображение и взаимодействие с формой ввода данных о потреблении воды, WaterViewModel.kt – класс ViewModel, связывающий фрагмент WaterFragment с моделью WaterBalance.

В качестве примера будет описан алгоритм определения степени дегидратации, представленный в листинге 1. Класс DehydrationLevel представляет собой реализацию логики оценки степени дегидратации на основе различных факторов, таких как внешний вид, состояние слизистой оболочки, наличие слез и состояние глаз. В конструкторе класса принимаются значения этих факторов.

Метод getPoints() вычисляет общее количество баллов, основываясь на оценке каждого фактора. Он вызывает приватные методы evaluateAppearance(), evaluateMucous(), evaluateTears() и evaluateEyes(), которые выполняют оценку по каждому из факторов и возвращают соответствующие баллы.

Приватные методы evaluateAppearance(), evaluateMucous(), evaluateTears() и evaluateEyes() используют выражение when [5] для сопоставления значений входных параметров с конкретными значениями и возвращают соответствующие баллы в зависимости от оценки. Например, метод evaluateAppearance() оценивает внешний вид и возвращает 0 для значения «Normal», 1 для значения «Irritable» и 2 для значения «Sluggish».

Листинг 1 – Сокращенный класс уровня дегидратации

class DehydrationLevel(

private val appearance: String,

private val mucous: String,

private val tears: String,

private val eyes: String

) {

fun getPoints(): Int {

}

private fun evaluateAppearance(): Int {

}

}

private fun evaluateMucous(): Int {

}

}

private fun evaluateTears(): Int {

}

}

private fun evaluateEyes(): Int {

}

}

}

На рисунке 10 представлен скриншот пользовательского интерфейса фрагмента DehydrationFragment в режиме разработчика.

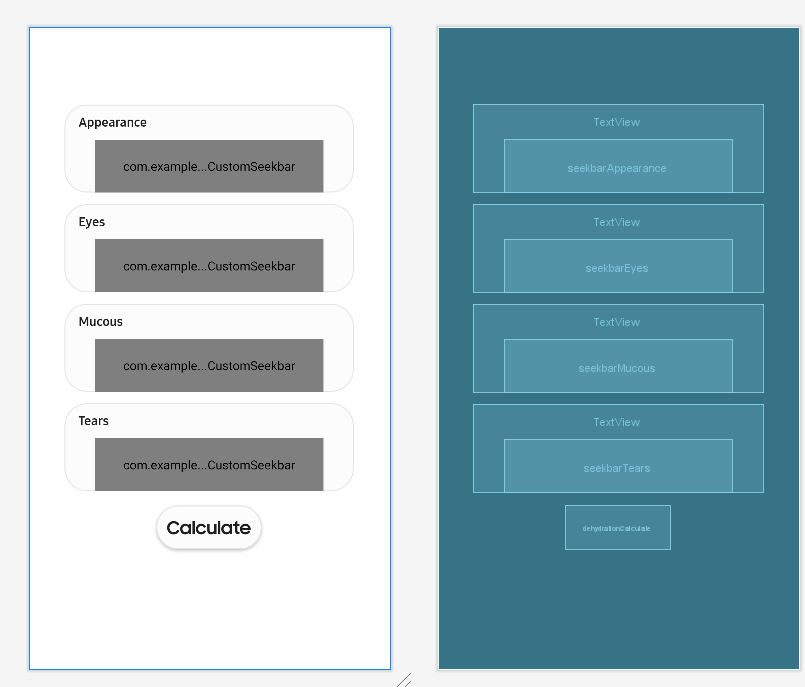


Рисунок 10 – Пользовательский интерфейс DehydrationFragment

2.6 Тестирование

Тестирование проводилось на двух реальных устройствах: Google Pixel 4 XL и Samsung Galaxy S22.

Характеристики Google Pixel 4 XL:

- операционная система: Android 13;

- процессор: Qualcomm Snapdragon 855;

- оперативная память: 6 ГБ;

- встроенная память: 64 ГБ;

- разрешение экрана: 1440 x 3040 пикселей;

- диагональ экрана: 6.3 дюйма;

- версия Bluetooth: 5.0;

- версия Wi-Fi: 5 (802.11 b/g/n/ac);

- батарея: 3700 мАч.

Характеристики Samsung Galaxy S22:

- операционная система: Android 13;

- процессор: Samsung Exynos 2200;

- оперативная память: 8 ГБ;

- встроенная память: 256 ГБ;

- разрешение экрана: 1080 x 2400 пикселей;

- диагональ экрана: 6.4 дюйма;

- версия Bluetooth: 5.2;

- версия Wi-Fi: 802.11 a/b/g/n/ac;

- батарея: 3700 мАч.

Весь функционал приложения Water Mate успешно протестирован на указанных выше устройствах. Приложение корректно работает на обоих смартфонах и предоставляет пользователю возможность вводить необходимые данные, рассчитывать показатели и получать соответствующие рекомендации. Интерфейс приложения адаптирован под различные экраны и разрешения, обеспечивая удобное взаимодействие с пользователем.

На данный момент публикации приложения Water Mate в магазинах приложений не проводилось. Разработанное приложение было предметом внутреннего тестирования и апробации командой разработчиков. Основной упор в данной стадии разработки делался на функциональность и пользовательский опыт. Результаты тестирования показали, что приложение успешно выполняет свою основную функцию и предоставляет необходимые рекомендации для определения степени дегидратации и коррекции электролитов.

Поскольку приложение Water Mate на данный момент не было опубликовано в магазинах приложений, статистика загрузок и обзоров недоступна. Однако, на основе результатов внутреннего тестирования, можно сделать вывод о том, что приложение обладает потенциалом привлечь внимание аудитории.

Таким образом, разработанное приложение Water Mate успешно прошло тестирование на устройствах Google Pixel 4 XL и Samsung Galaxy S22, демонстрируя стабильную работу и соответствие заявленному функционалу. Дополнительное тестирование и апробация приложения в реальных клинических условиях могут быть рекомендованы для дальнейшего совершенствования и улучшения приложения перед его публикацией в магазинах приложений.

Заключение

В данной курсовой работе было разработано мобильное приложение под названием Water Mate, предназначенное для помощи в диагностике и коррекции дегидратации и электролитного баланса. Введение в работе описывает актуальность проблемы поддержания гидратации и уровня электролитов для здоровья и благополучия каждого человека.

Основная цель работы состояла в детальном описании разработанного мобильного приложения Water Mate и его функциональности. Для достижения этой цели были поставлены следующие задачи: разработка пользовательского интерфейса, реализация механизма ввода и обработки персональных данных, создание моделей данных, разработка алгоритмов расчета и вывода рекомендаций, а также построение файловой структуры приложения.

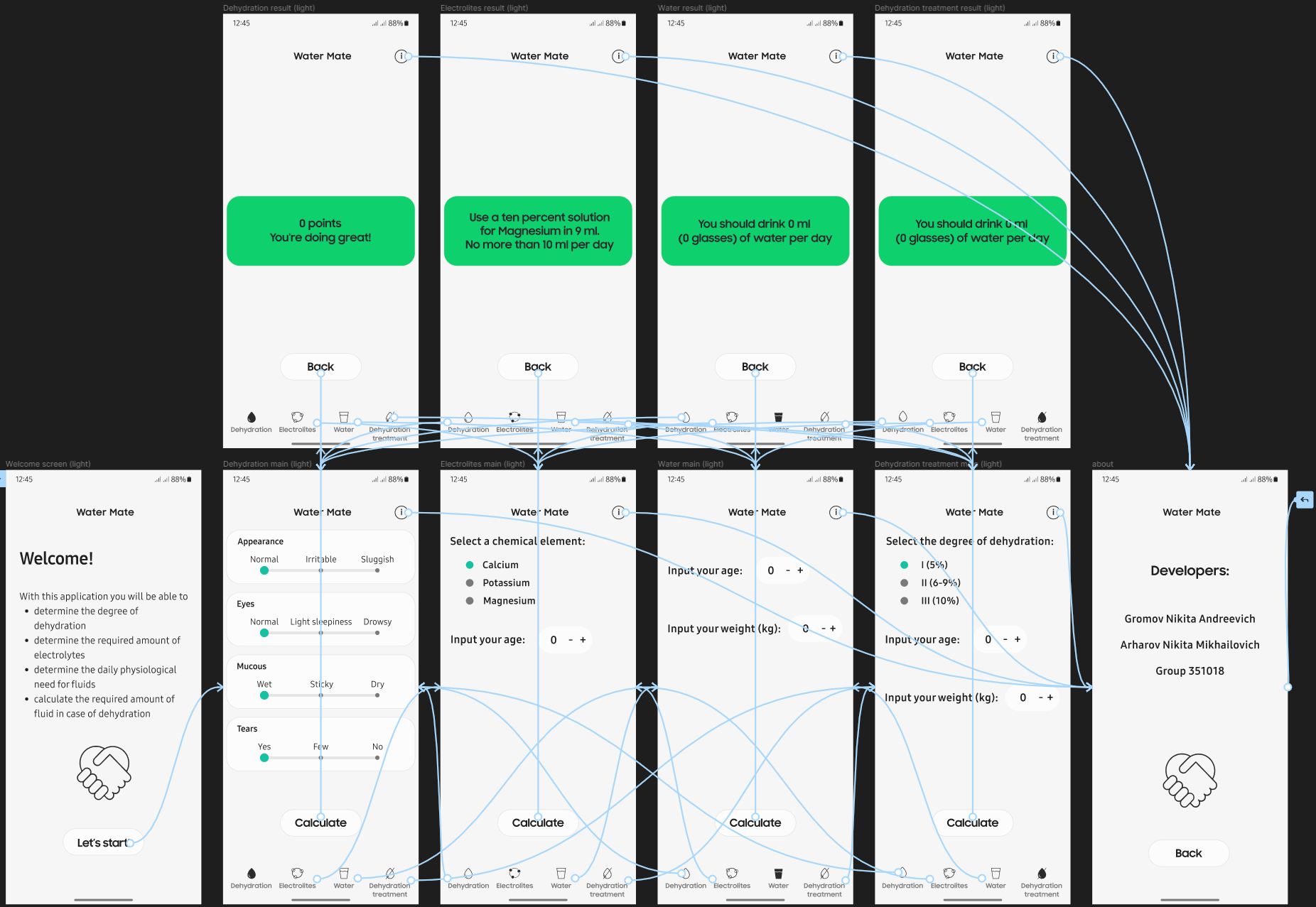
В ходе выполнения работы был разработан интерактивный прототип мобильного приложения, который обеспечивает удобство использования и понятность. Мобильное приложение было реализовано на языке программирования Kotlin для платформы Android. Оно позволяет пользователям вводить необходимые данные для расчета степени дегидратации, коррекции электролитов и определения необходимого объема воды. После обработки данных приложение выдает рекомендации и выводит результаты на экран.

Таким образом, выполнение поставленных задач и достижение поставленных целей позволило успешно разработать мобильное приложение Water Mate, которое предоставляет удобный инструмент для определения степени дегидратации и коррекции электролитов. Это приложение может быть полезным в повседневной практике и способствовать улучшению здоровья и благополучия пользователей.

Список используемых источников

1. Официальная документация Kotlin «Kotlin для Android» [Электронный ресурс]. - Текст: электронный. - URL: https://kotlinlang.ru/docs/android-overview.html (дата обращения: 29.05.2023). - Режим доступа: с загл. экрана.
2. Официальная документация Kotlin «Создание вашего первого мобильного приложения» [Электронный ресурс]. - Текст: электронный. - URL: https://kotlinlang.ru/docs/multiplatform-mobile-create-first-app.html (дата обращения: 18.05.2023). - Режим доступа: с загл. экрана.
3. Статья «Kotlin for Android Developers» [Электронный ресурс]. - Текст: электронный. - URL: https://medium.com/@mohit\_kumar/kotlin-for-android-developers-1-6f8d7a6f8b0c (дата обращения: 18.05.2023). - Режим доступа: с загл. экрана.
4. Статья «Современная Android разработка на Kotlin. Часть 1» [Электронный ресурс]. - Текст: электронный. - URL: https://habr.com/ru/articles/341602/ (дата обращения: 15.05.2023). - Режим доступа: с загл. экрана.
5. Статья на сайте «Android Developers» «Kotlin for Android» [Электронный ресурс]. - Текст: электронный. - URL: https://developer.android.com/kotlin (дата обращения: 06.06.2023). - Режим доступа: с загл. экрана.

## Приложение А. Прототип приложения



Приложение Б. Листинг кода курсовой работы

Листинг 2 – Листинг WelcomeScreen

|  |
| --- |
| package com.example.watermate  import android.content.Intent  import android.media.MediaPlayer  import androidx.appcompat.app.AppCompatActivity  import android.view.View  import android.os.Bundle  import android.widget.ListView  import android.widget.Button  import androidx.appcompat.app.AppCompatDelegate  import com.example.watermate.custom\_utils.CustomArrayAdapter  @Suppress("DEPRECATION")  class WelcomeScreen : AppCompatActivity() {  private val PREF\_FIRST\_RUN = "first\_run"  override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  overridePendingTransition(0, 0)  window.decorView.systemUiVisibility = View.SYSTEM\_UI\_FLAG\_FULLSCREEN  supportActionBar?.hide()  AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE\_NIGHT\_NO)  setContentView(R.layout.activity\_welcome\_screen)  val isFirstRun = getSharedPreferences("MyPrefs", MODE\_PRIVATE)  .getBoolean(PREF\_FIRST\_RUN, true)  if (isFirstRun) {  val abilitiesArray = resources.getStringArray(R.array.possibilities\_list)  val listView = findViewById<ListView>(R.id.possibilities)  val adapter = CustomArrayAdapter(this, android.R.layout.simple\_list\_item\_1, abilitiesArray)  listView.adapter = adapter  val pop = MediaPlayer.create(this, R.raw.pop)  val button = findViewById<Button>(R.id.start)  button.setOnClickListener {  pop.start()  val intent = Intent(this, FunctionalActivity::class.java)  startActivity(intent)  getSharedPreferences("MyPrefs", MODE\_PRIVATE)  .edit()  .putBoolean(PREF\_FIRST\_RUN, false)  .apply()  }  } else {  val intent = Intent(this, FunctionalActivity::class.java)  startActivity(intent)  }  }  } |

Листинг 3 - Листинг FunctionalActivity

|  |
| --- |
| package com.example.watermate  import android.content.Intent  import android.os.Bundle  import android.view.View  import android.widget.ImageView  import com.google.android.material.bottomnavigation.BottomNavigationView  import androidx.appcompat.app.AppCompatActivity  import androidx.navigation.findNavController  import androidx.navigation.ui.AppBarConfiguration  import androidx.navigation.ui.setupActionBarWithNavController  import androidx.navigation.ui.setupWithNavController  import com.example.watermate.databinding.ActivityFunctionalBinding  @Suppress("DEPRECATION")  class FunctionalActivity : AppCompatActivity() {  private lateinit var binding: ActivityFunctionalBinding  override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  overridePendingTransition(0, 0)  window.decorView.systemUiVisibility = View.SYSTEM\_UI\_FLAG\_FULLSCREEN  binding = ActivityFunctionalBinding.inflate(layoutInflater)  setContentView(binding.root)  supportActionBar?.hide()  val navView: BottomNavigationView = binding.navView  val navController = findNavController(R.id.nav\_host\_fragment\_activity\_functional)  val appBarConfiguration = AppBarConfiguration(  setOf(  R.id.navigation\_dehydration, R.id.navigation\_electrolites,  R.id.navigation\_water, R.id.navigation\_dehydration\_treatment  )  )  setupActionBarWithNavController(navController, appBarConfiguration)  navView.setupWithNavController(navController)  val button = findViewById<ImageView>(R.id.information)  button.setOnClickListener {  val intent = Intent(this, InfoActivity::class.java)  startActivity(intent)  }  }  } |

Листинг 4 – Листинг InfoActivity

|  |
| --- |
| package com.example.watermate  import android.content.Intent  import android.media.MediaPlayer  import androidx.appcompat.app.AppCompatActivity  import android.view.View  import android.os.Bundle  import android.widget.Button  @Suppress("DEPRECATION")  class InfoActivity : AppCompatActivity() {  override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  overridePendingTransition(0, 0)  window.decorView.systemUiVisibility = View.SYSTEM\_UI\_FLAG\_FULLSCREEN  setContentView(R.layout.activity\_info)  supportActionBar?.hide()  val pop = MediaPlayer.create(this, R.raw.pop)  val button = findViewById<Button>(R.id.back)  button.setOnClickListener {  pop.start()  val intent = Intent(this, FunctionalActivity::class.java)  startActivity(intent)  }  }  } |

Листинг 5 – Листинг WaterBalance

|  |
| --- |
| package com.example.watermate.models  class WaterBalance(private val weight: Double, private val age: Int) {  fun calculateDailyWaterIntake(): Double {  val multiplier = when {  age < 1 -> 80.0  age in 1..10 -> (age + 40).toDouble()  else -> when {  age < 14 -> 40.0  age < 30 -> 35.0  age < 55 -> 30.0  else -> 25.0  }  }  return weight \* multiplier  }  fun calculateDailyWaterIntakeInGlasses(): Int {  val waterIntake = calculateDailyWaterIntake()  val glassVolume = 250  return (waterIntake / glassVolume).toInt()  }  } |

Листинг 6 – Листинг WaterViewModel

|  |
| --- |
| package com.example.watermate.ui.water  import androidx.lifecycle.ViewModel  import com.example.watermate.models.WaterBalance  class WaterViewModel : ViewModel() {  fun calculateWaterBalance(age: Int, weight: Int): String {  val waterBalance = WaterBalance(weight.toDouble(), age)  val dailyWaterIntake = waterBalance.calculateDailyWaterIntake() / 1000  val dailyWaterIntakeInGlasses = waterBalance.calculateDailyWaterIntakeInGlasses()  return "You should drink $dailyWaterIntake Liters\n($dailyWaterIntakeInGlasses glasses) of water per day"  }  } |

Листинг 7 – Листинг DehydrationFragment

|  |
| --- |
| package com.example.watermate.ui.dehydration  import android.media.MediaPlayer  import android.os.Bundle  import android.view.LayoutInflater  import android.view.View  import android.view.ViewGroup  import androidx.fragment.app.Fragment  import androidx.lifecycle.ViewModelProvider  import com.example.watermate.R  import com.example.watermate.databinding.FragmentDehydrationBinding  class DehydrationFragment : Fragment() {  private var \_binding: FragmentDehydrationBinding? = null  override fun onCreateView(  inflater: LayoutInflater,  container: ViewGroup?,  savedInstanceState: Bundle?  ): View {  val binding = FragmentDehydrationBinding.inflate(inflater, container, false)  val root: View = binding.root  val seekbarAppearance = binding.seekbarAppearance  seekbarAppearance.labels = resources.getStringArray(R.array.labels\_appearance)  seekbarAppearance.max = 2  val seekbarEyes = binding.seekbarEyes  seekbarEyes.labels = resources.getStringArray(R.array.labels\_eyes)  seekbarEyes.max = 2 |

Листинг 7 – Листинг DehydrationFragment (окончание)

|  |
| --- |
| val seekbarMucous = binding.seekbarMucous  seekbarMucous.labels = resources.getStringArray(R.array.labels\_mucous)  seekbarMucous.max = 2  val seekbarTears = binding.seekbarTears  seekbarTears.labels = resources.getStringArray(R.array.labels\_tears)  seekbarTears.max = 2  val dehydrationMain = binding.fragmentDehydrationMain  val dehydrationResult = binding.fragmentDehydrationResult  val calculateDehydration = binding.dehydrationCalculate  val backDehydration = binding.dehydrationBack  val viewModel = ViewModelProvider(this).get(DehydrationViewModel::class.java)  val pop = MediaPlayer.create(requireContext(), R.raw.pop)  calculateDehydration.setOnClickListener {  pop.start()  val appearanceValue = seekbarAppearance.progress  val eyesValue = seekbarEyes.progress  val mucousValue = seekbarMucous.progress  val tearsValue = seekbarTears.progress  val resultStr = viewModel.calculateDehydration(appearanceValue, eyesValue, mucousValue, tearsValue)  binding.dehydrationResultCard.text = resultStr  dehydrationMain.visibility = View.GONE  dehydrationResult.visibility = View.VISIBLE  }  backDehydration.setOnClickListener {  pop.start()  dehydrationMain.visibility = View.VISIBLE  dehydrationResult.visibility = View.GONE  }  return root  }  override fun onDestroyView() {  super.onDestroyView()  \_binding = null  }  } |