Министерство образования Новосибирской области ГБПОУ НСО «Новосибирский авиационный технический колледж имени Б.С. Галущака»

Разработка мобильного приложения  
«Дневник здоровья»

Пояснительная записка к курсовому проекту

ПМ.01 Разработка модулей программного обеспечения для компьютерных систем

МДК 01.03 Разработка мобильных приложений

НАТКиГ.202200.010.000ПЗ

Выполнил: Овчаров Д. В.

2023

Содержание

[Введение 5](#_Toc138598147)

[1 Исследовательский раздел 6](#_Toc138598148)

[1.1 Описание предметной области 6](#_Toc138598149)

[1.2 Образ клиента 6](#_Toc138598150)

[1.3 Сценарии 7](#_Toc138598151)

[1.4 Сбор и анализ прототипов 7](#_Toc138598152)

[2 Проектирование приложения 14](#_Toc138598153)

2.1 [UI/UX дизайн проекта 14](#_Toc138598154)

[2.2 Выбор технологии, языка и среды программирования 16](#_Toc138598155)

[3 Разработка мобильного приложения 17](#_Toc138598156)

[3.1 Разработка базы данных 17](#_Toc138598157)

[3.2 Описание используемых плагинов 17](#_Toc138598158)

[3.3 Описание разработанных процедур и функций 18](#_Toc138598159)

[4 Тестирование 20](#_Toc138598160)

[4.1 Протокол тестирования дизайна приложения 20](#_Toc138598161)

[4.2 Протоколтестированияфункционалаприложения 20](#_Toc138598162)

[Заключение 24](#_Toc138598189)

[Библиография 25](#_Toc138598190)

[Приложение А (обязательное) Техническое задание 26](#_Toc138598191)

[Приложение Б (справочное) Словарь данных 32](#_Toc138598206)

[Приложение В (справочное) Листинги разработанных функций 33](#_Toc138598207)

Введение

Людям трудно запомнить всё, что с ними происходило, что они принимали за время болезни, особенно, если их болезнь длится уже продолжительное время. Дневник здоровья поможет вести записи о состоянии на день и принимаемых лекарствах. Также определение болезни станет проще, т.к. на основе данных дневника проще выявить болезнь.

Тема здоровья будет актуальна всегда, ведь только здоровый человек сможет полноценно функционировать, а в мегаполисах от людей требуется практически всё время быть здоровым.

Целью курсового проекта является создание мобильно приложения для ведения дневника здоровья.

Задачами курсового проекта являются:

* изучение предметной области темы;
* определение целевой аудитории;
* определения назначения приложения путем создания сценариев использования;
* рассмотрение прототипов для выявления необходимых функций;
* проектирование приложения;
* разработка мобильного приложения;
* тестирование приложения.

Объект исследования – приложения-дневники.

Предмет исследования – изучение функционала и принципов функционирования приложений-дневников.

1 Исследовательский раздел

## 1.1 Описание предметной области

Здоровье – это одна из самых важных вещей в жизни человека. Болеющий человек не сможет работать, а значит и обеспечивать свою жизнь, поэтому его задачей является быстрое определение болезни и её лечение.

Ведение дневника здоровья поможет отслеживать изменения в жизнедеятельности человека, а также поможет определить больному, когда он заболел. В дневнике здоровья человек может записать различные сведения о своем здоровье, такие как следующие сведения:

* самочувствие;
* беспокойства;
* симптомы желудочно-кишечного тракта;
* настроение;
* тревога и т.д.

На приеме у врача пациенту придется вспомнить все сведения о своем здоровье. Запоминать это всё самому очень проблематично, ибо история болезни может быть слишком большой, а различных симптомов слишком много, поэтому для этого отлично подойдет дневник здоровья, в котором человек может расписать по дням все проявляемые в эти дни симптомы.

Дневник здоровья поможет вести записи о состоянии здоровья. Это позволяет систематизировать данные о здоровье человека, что при отслеживании изменений в здоровье, на приеме у врача, а также при самостоятельном определении болезни.

## 1.2 Образ клиента

Клиентами являются болеющие люди и люди, отслеживающие изменения в жизнедеятельности.

Мужчины и женщины болеют примерно одинаково, поэтому приложение предназначено для обоих полов.

Возраст нашей аудитории начинается с 16 лет, потому что обычно с этого времени люди начинают сами следить за своим здоровьем.

## 1.3 Сценарии

Человек, который недавно заболел, хочет сходить на прием к врачу и, чтобы не забыть какие-либо данные о болезни, он может скачать себе приложение, чтобы их записать.

Давно болеющий человек хочет сходить к доктору, но перед этим ему необходимо систематизировать информацию о его болезни, расписать её по дням. Данное приложение поможет ему сформировать историю болезни, по которой определение болезни будет проще и точнее.

Человеку необходимо следить за изменением своего здоровья из-за недавней болезни. В данном случае ему подойдет это приложение.

## 1.4 Сбор и анализ прототипов

«Моё здоровье»

На рисунке 1 представлен календарь.



Рисунок 1 – Страница «Календарь»

У приложения есть шапка, информационная часть и навигационное меню в левой части экрана. Шапку приложения составляют название раздела и меню. В навигационном меню имеются кнопки для перехода между экранами «Календарь», «Мониторинг показателей». Внутри информационной части приложение содержит календарь, в котором можно создать напоминание. На рисунке 2 изображен мониторинг показателей.



Рисунок 2 – Страница «Мониторинг показателей»

На данной странице можно выбрать, какой список показателей показать. На рисунке 3 изображена список показаний пульса.

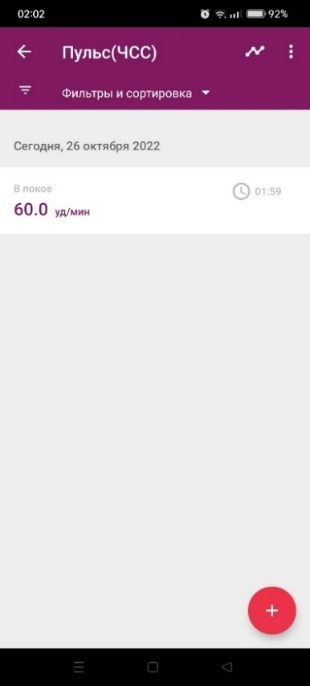


Рисунок 3 – Список показаний пульса

На данной странице можно добавить новую запись в список и просматривать список. Исходя из рассмотрения данного приложения можно сделать вывод:

Плюсы приложения:

* оснащённость ключевыми функциями.

Минусы приложения:

* при переходе на страницу «Календарь» приложение вылетает;
* небольшое количество регистрируемых данных.

«Здоровье.ру»

Это приложение отличается от предыдущего приложения более приятным дизайном, а также дневник здоровья здесь больше. Приложение принимает информацию не с помощью заполнения форм, а с помощью выбора вариантов ответов. На рисунке 4 представлена настройка дневника.

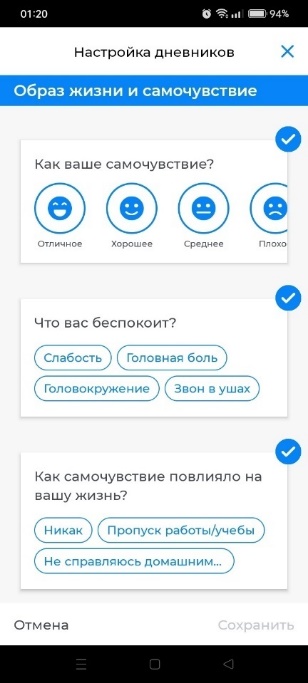


Рисунок 4 – Настройка дневника

На данной странице мы выбираем, какие вопросы у нас будут задаваться в дневнике. На рисунке 5 изображен дневник здоровья.



Рисунок 5 – Дневник здоровья

На данной странице мы можем выбрать день записи, просмотреть и изменить данные.

Исходя из рассмотрения данного приложения можно сделать вывод:

Плюсы приложения:

* оснащённость главными функциями;
* приятный дизайн, с использование 2 цветов;
* настраиваемый дневник здоровья, который позволяет отсечь ненужные вопросы;
* приложение принимает информацию с помощью выбора варианта ответа.

«Daylio»

Это приложение является дневником настроения. Как и в предыдущем, приложение принимает информацию с помощью выбора варианта ответа.

На рисунке 6 изображены записи.

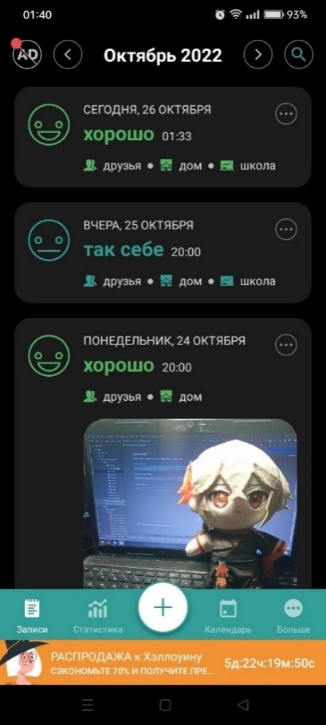


Рисунок 6 – Записи

У приложения есть шапка, информационная часть и навигационное меню в нижней части экрана. В навигационном меню имеются кнопки для перехода между экранами «Записи», «Статистика», «Календарь». Внутри информационной части приложение содержит дневник записей, которые можно выбрать и изменить. На рисунке 7 представлена страница «статистика».

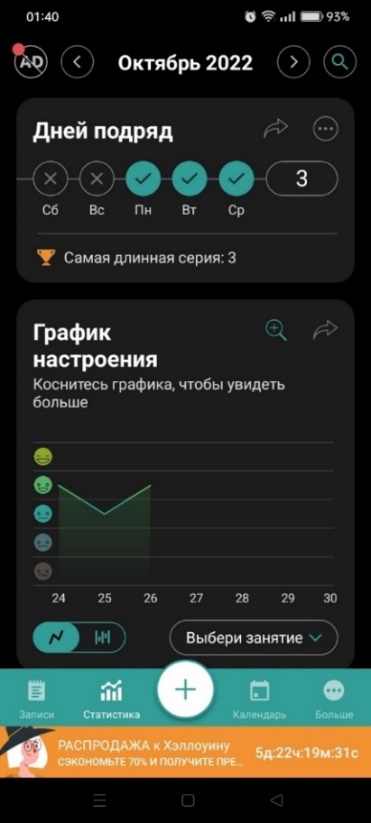


Рисунок 7 — Статистика записей

На этой странице нам представлен график с осями в виде уровня настроения и даты.

На рисунке 8 изображен календарь.

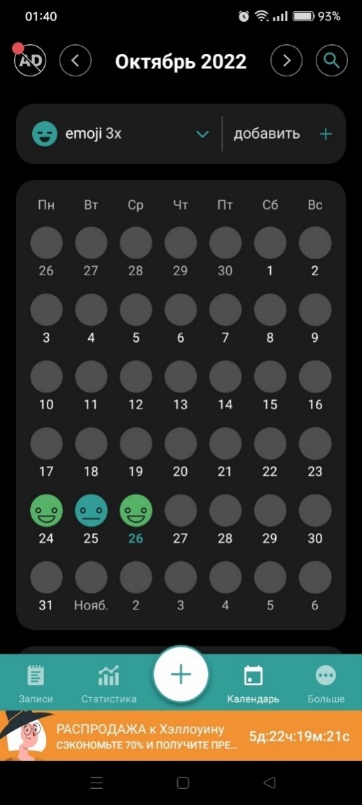


Рисунок 8 – Календарь

На данной странице нам представлен календарь, в котором дни имеют свои записи. Мы можем выбрать день и редактировать запись этого дня.

Исходя из рассмотрения данного приложения можно сделать вывод:

Плюсы приложения:

* оснащённость главными функциями;
* настраиваемое количество данных, которое мы можем выбрать;
* приложение принимает информацию с помощью выбора варианта ответа.

Минусы:

* плохо-адаптированный дизайн темной темы.

«Дневник Здоровья»

На рисунке 9 изображен дизайн приложения «Дневник Здоровья»



Рисунок 9 – Дизайн приложения «Дневник Здоровья»

Данное приложение по функционалу похоже на первое, в отличие от него имеет более приятный дизайн, выдержанный в 2 цветах. На данной странице показаны параметры, которые ведутся в дневнике. Их можно просмотреть и записать.

Исходя из рассмотрения данного приложения можно сделать вывод:

Плюсы приложения:

* оснащённость ключевыми функциями;
* приятный дизайн с использованием 2 цветов.

Минусы приложения:

* небольшое количество регистрируемых данных.

«Дневник здоровья: показатели»

На рисунке 10 изображен дизайн приложения «Дневник здоровья: показатели».



Рисунок 10 – Дизайн приложения «Дневник здоровья: показатели»

На данной странице нам представлен дневник здоровья в виде таблицы. Нажав снизу на кнопку, у нас откроется форма добавления новой записи, после заполнения которой у нас появится новая запись в таблице.

Плюсы приложения:

* оснащённость ключевыми функциями;
* дизайн выдержан в 2 цветах.

Минусы приложения:

* небольшое количество регистрируемых данных.

В заключении ключевыми особенностями в разработке являются: возможность регистрации и авторизации, просмотр предыдущих записей и добавление новых записей через опрос пользователя.

2 Проектирование приложения

## 2.1 UI/UX дизайн проекта

Дизайн проекта разработан в программе Figma, потому что это приложение специализировано для создания дизайна мобильных приложений.

Для проекта определены основные экраны:

* главный;
* добавления записи в дневник;
* просмотра записи из дневника;
* аналитики записей.

Цветовая гамма выбрана по тематике больницы. Больница ассоциируется у людей больше с белым и синим. Здания больниц обычно имеют белый цвет, а форма персонала состоит из белого и синего цветов. Поэтому в приложении принято использовать белый цвет и оттенки синего цвета для светлой темы и черный с оттенками синего цвета для темной темы. На рисунке 11 изображена главная страница приложения.

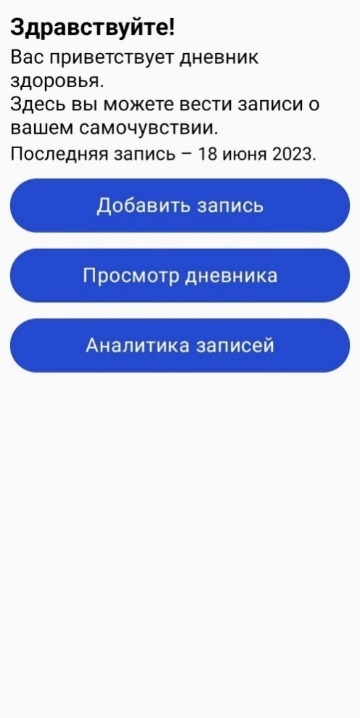


Рисунок 11 – Главная страница

На данной странице мобильного приложения предоставлена возможность добавить новую запись в дневник, просмотреть запись из дневника, и посмотреть сводку по записям за месяц. При нажатии на кнопку «Добавить запись» откроется страница добавления записи, при нажатии на кнопку «Просмотр дневника» – страница просмотра записи.

На рисунке 12 изображена страница добавления записи.

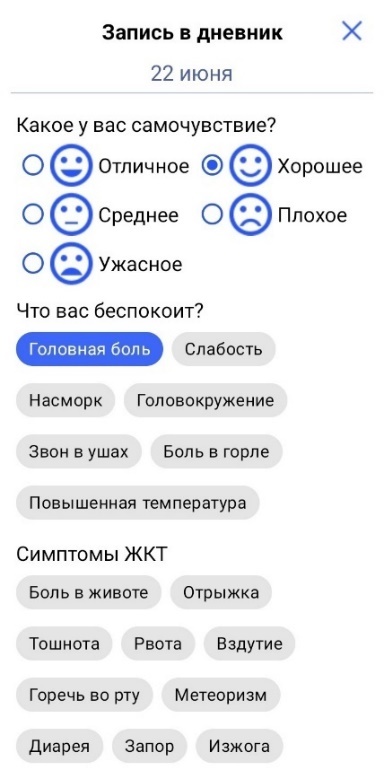


Рисунок 12 – Страница добавления записи

На данной страниценаходятся различные вопросы о самочувствии пользователя, на которые пользователь отвечает. При нажатии самой нижней кнопки на странице «Сохранить» данные опроса сохраняются.

На рисунке 13 изображена страница просмотра записей.

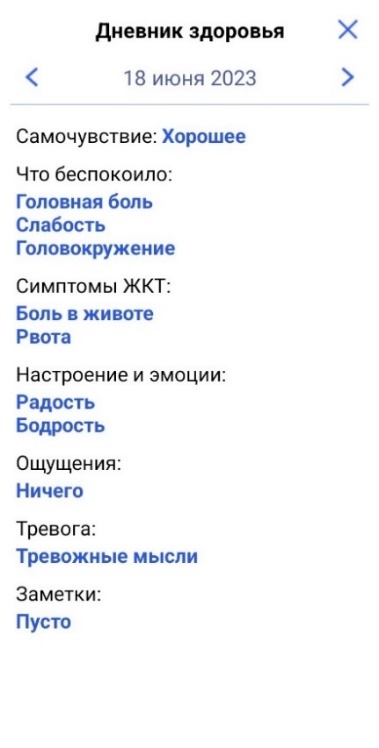


Рисунок 13 – Страница просмотра записей

На данной странице пользователь может выбрать дату записи и просмотреть выбранную запись. На рисунке 14 изображена страница аналитики записей.

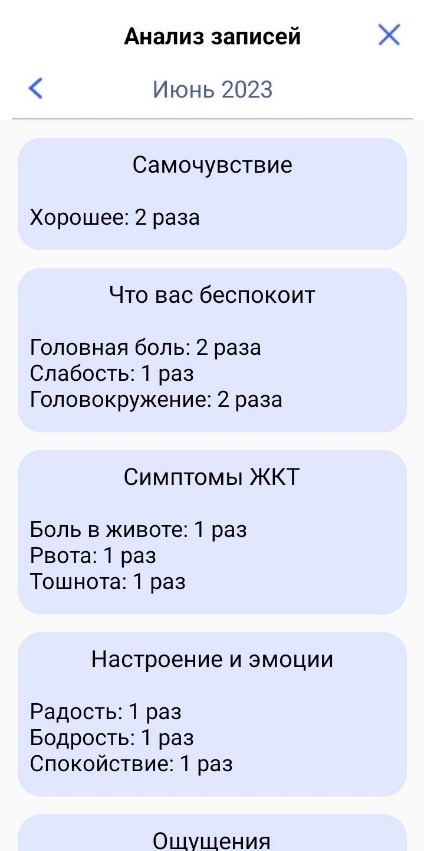


Рисунок 14 – Страница аналитики записей

На данной странице пользователь может выбрать месяц и просмотреть сводки записей за выбранный месяц.

## 2.2 Выбор технологии, языка и среды программирования

Средой программирования выбрана программа Android Studio, потому что она позволит нам создать приложение, которое будет поддерживать большинство телефонов. Языком программирования является Java. Java – строго типизированный объектно-ориентированный язык программирования общего назначения, в котором будет легко реализовать данную работу.

Используемый сервис для базы данных – SQLite. Он выбран из-за необходимости создания для каждого пользователя свою базу данных. SQLite позволяет реализовать эту возможность.

3 Разработка мобильного приложения

## 3.1 Разработка базы данных

В проекте использована СУБД SQLite для запросов к базе данных. На рисунке 15 изображена ER-диаграмма базы данных.

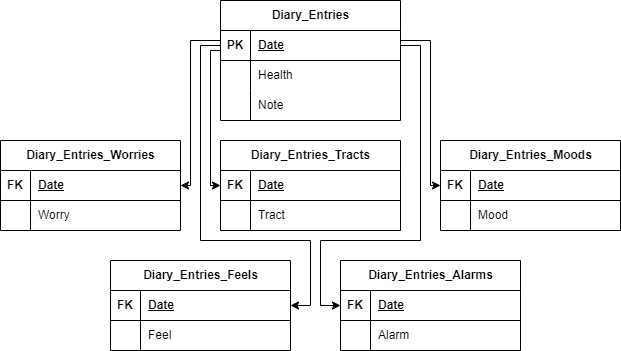


Рисунок 15 – ER-диаграмма

Благодаря SQLite реализовано добавление записей в дневник, вывод определенных записей и вывод записей за месяц. В таблице «Diary\_Entries» хранятся записи о самочувствии и заметки, в таблице «Diary\_Entries\_Worries» – записи о том, что беспокоит пользователя, в таблице «Diary\_Entries\_Tracts» – записи о симптомах ЖКТ, в таблице «Diary\_Entries\_Moods» – записи о настроении, в таблице «Diary\_Entries\_Feels» – записи о ощущениях, в таблице «Diary\_Entries\_Alarms» – записи о тревоге. В приложении Б приведен словарь данных базы данных.

## 3.2 Описание используемых плагинов

При реализации приложения использованы встроенные плагины. На рисунке 16 изображен список используемых плагинов.



Рисунок 16 – Используемые плагины

Используемые плагины предназначены для работы приложения с новым API и для реализации дизайна.

## 3.3 Описание разработанных процедур и функций

В приложении разработаны следующие функции:

* получение даты последней записи;
* добавление/перезаписывание записи;
* загрузка текущей записи при добавлении;
* просмотр записи за выбранную дату;
* просмотр сводки записей за выбранный месяц.

На листинге 1 представлен код, отвечающий за получение даты последней записи.

Листинг 1 – Получение даты последней записи

private void getLastDate(){  
 SQLiteDatabase db = getBaseContext().openOrCreateDatabase("diary.db",*MODE\_PRIVATE*, null);  
 db.execSQL("CREATE TABLE IF NOT EXISTS Diary\_Entries (date TEXT PRIMARY KEY, health Text, note Text)");  
 db.execSQL("CREATE TABLE IF NOT EXISTS Diary\_Entries\_Worries (date TEXT, worry TEXT)");  
 db.execSQL("CREATE TABLE IF NOT EXISTS Diary\_Entries\_Tracts (date TEXT, tract TEXT)");  
 db.execSQL("CREATE TABLE IF NOT EXISTS Diary\_Entries\_Moods (date TEXT, mood TEXT)");  
 db.execSQL("CREATE TABLE IF NOT EXISTS Diary\_Entries\_Feels (date TEXT, feel TEXT)");  
 db.execSQL("CREATE TABLE IF NOT EXISTS Diary\_Entries\_Alarms (date TEXT, alarm TEXT)");  
  
 Cursor query = db.rawQuery("SELECT date FROM Diary\_Entries",null);*//получение списка дат* tvLastDate = findViewById(R.id.*tvLastDate*);  
 if(query.getCount()>0){*//если есть данные* query.moveToLast();*//перейти к последнему* String dateString = query.getString(0);*//получение даты* DateTimeFormatter formatter = DateTimeFormatter.*ofPattern*("yyyy-MM-dd");  
 LocalDate date = LocalDate.*parse*(dateString,formatter);*//переход из строки в дату* dateString = date.getDayOfMonth() + " ";*//формирование строки для вывода* switch (date.getMonthValue()){  
 case 1: dateString += "января "; break;  
 case 2: dateString += "февраля "; break;  
 case 3: dateString += "марта "; break;  
 case 4: dateString += "апреля "; break;  
 case 5: dateString += "мая "; break;  
 case 6: dateString += "июня "; break;  
 case 7: dateString += "июля "; break;  
 case 8: dateString += "августа "; break;  
 case 9: dateString += "сентября "; break;  
 case 10: dateString += "октября "; break;  
 case 11: dateString += "ноября "; break;  
 case 12: dateString += "декабря "; break;  
 }  
 dateString += date.getYear() + ".";  
 tvLastDate.setText(dateString);  
 }  
}

При получении даты происходит получение списка всех дат из таблицы Diary\_Entries, из которой берется последняя и форматируется в строку, после чего устанавливается в TextView.

При добавлении записи происходит получение выбранных данных, удаление записей текущей даты и добавление выбранных данных.

При просмотре записи с помощью запросов получаем данные за выбранную дату, и они выводятся пользователю.

При просмотре сводки записей за выбранный месяц с помощью запросов получаем данные за выбранный месяц, считаем их количество и выводим их пользователю.

В приложении В приведены листинги, на которых представлены код функций, отвечающих за добавление/перезаписывание записи, за просмотр записи выбранной даты и за просмотр сводки записей выбранного месяца.

.

4 Тестирование

## 4.1 Протокол тестирования дизайна приложения

Для проведения тестирования дизайна приложения выбрано устройство realme C25s с разрешением экрана 720x1600.

Проведем тестирование оптимального размера кнопок. Сверим разработанный прототип с итоговым результатом. По итогу тестирования все кнопки и элементы дизайна отображается корректно.

Проведем тестирование на читабельность текста. Сверим размер шрифта в разработанном прототипе со шрифтом в итоговом результате. По итогу тестирования все надписи прошли проверку.

Проведем тестирование цветового сочетания. Сверим разработанный дизайн с дизайном итогового приложения. Дизайн приложения соответствует разработанному дизайну, подходит для этого приложения и приятен глазу.

В результате тестирования дизайна все тест пройдены и дизайн приложения корректно отображается.

## 4.2 Протоколтестированияфункционалаприложения

Для тестирования функционала разработаны тест-кейсы. В таблицах представлены разработанные тест-кейсы.

Таблица 1 – Test Case 1

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Дымовое тестирование | | |
| **Функция:** | Запуск приложение | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  | |
| **Шаги теста:** | |  | |
| Открыть приложение | | Корректный запуск приложения | Пройден |
| **Постусловие:** | |  | |
| Продолжение работы | | Загрузка главного экрана | Пройден |

Таблица 2 – Test Case 2

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Критическое тестирование | | |
| **Функция:** | Работа переходов в приложении | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  | |
| Запуск приложения | | Корректный запуск приложения | Пройден |
| **Шаги теста:** | |  | |
| Нажатие кнопки перехода на другой экран | | Корректный переход на этот экран | Пройден |
| Нажатие кнопки перехода на главный экран | | Корректный переход на главный экран | Пройден |
| **Постусловие:** | |  | |
| Продолжение работы | | Приложение работает и функционирует | Пройден |

Таблица 3 – Test Case 3

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Критическое тестирование | | |
| **Функция:** | Добавление записи без наличия сегодняшней записи | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  | |
| Запуск приложения | | Корректный запуск приложения | Пройден |
| **Шаги теста:** | |  | |
| Нажатие кнопки перехода на экран добавления | | Корректный переход на экран добавления | Пройден |
| Выбираем вносимые данные | | Вносимые элементы выбраны | Пройден |
| Нажатие на кнопку сохранения | | Данные сохраняются корректно и происходит переход на главный экран | Пройден |
| **Постусловие:** | |  | |
| Продолжение работы | | Приложение работает и функционирует | Пройден |

Таблица 4 – Test Case 4

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Критическое тестирование | | |
| **Функция:** | Добавление записи с наличием сегодняшней записи | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  | |
| Запуск приложения | | Корректный запуск приложения | Пройден |
| Добавить запись | | Корректное добавление | Пройден |
| **Шаги теста:** | |  | |
| Нажатие кнопки перехода на экран добавления | | Корректный переход на экран добавления, заранее внесенные данные выбраны | Пройден |
| Меняем вносимые данные | | Вносимые элементы изменены | Пройден |
| Нажатие на кнопку сохранения | | Данные сохраняются вместо старых корректно и происходит переход на главный экран | Пройден |
| **Постусловие:** | |  | |
| Продолжение работы | | Приложение работает и функционирует | Пройден |

Таблица 5 – Test Case 5

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Критическое тестирование | | |
| **Функция:** | Просмотр записи | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  | |
| Запуск приложения | | Корректный запуск приложения | Пройден |
| Добавить запись | | Корректное добавление | Пройден |
| **Шаги теста:** | |  | |
| Нажатие кнопки перехода на экран просмотра | | Корректный переход на экран просмотра, сегодняшние данные выведены | Пройден |
| Сменяем дату с помощью кнопок смены даты | | Дата изменяется, выводятся данные выбранной даты | Пройден |
| **Постусловие:** | |  | |
| Продолжение работы | | Приложение работает и функционирует | Пройден |

Таблица 6 – Test Case 6

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Критическое тестирование | | |
| **Функция:** | Аналитика записи | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  | |
| Запуск приложения | | Корректный запуск приложения | Пройден |
| Добавить запись | | Корректное добавление | Пройден |
| **Шаги теста:** | |  | |
| Нажатие кнопки перехода на экран аналитики | | Корректный переход на экран аналитики, выводятся данные за текущий месяц | Пройден |
| Сменяем месяц с помощью кнопок смены месяца | | Месяц изменяется, выводятся данные за выбранный месяц | Пройден |
| **Постусловие:** | |  | |
| Продолжение работы | | Приложение работает и функционирует | Пройден |

В итоге тестирования функционала все тест-кейсы прошли проверку корректно.

## Заключение

Результатом курсового проекта является разработка мобильного приложения, выполняющее роль дневника здоровья.

Данное приложение позволяет осуществить добавление записей в дневник. Также пользователь может переписать сегодняшнюю запись. Пользователь также может просмотреть определенную запись, выбрав дату записи. Кроме того, пользователь может посмотреть данные записей за выбранный месяц.

Дизайн приложения разработан в Figma. Приложение разработано на языке Java, с использованием системы управления базы данных SQLite, которая позволила реализовать создание базы данных и работу с локальной базой данных.

В ходе выполнения проекта решены следующие задачи:

* определения назначения приложения путем создания сценариев использования;
* исследование и изучение требований для дневников здоровья путем рассмотрения прототипов;
* разработка структуры приложения, определяющая дизайн приложения, основные модули и связи между ними;
* разработка приложения, включающая в себя создание кода и реализацию функциональности;
* тестирования приложения путем выполнения тест-кейсов.

Во время разработки возникли трудности с добавлением поля для даты и поиска по этому полю. Проблема была решена использованием для этого поля типа строки и форматированием дат в определённый формат.

## Библиография

1. ГОСТ Р 2.105–2019 ЕСКД. Общие требования к текстовым документам. – Москва: Стандартинформ, 2019. — 36 с.
2. Figma [Электронный ресурс]. – Прототипирование проектов. – URL: https://www.figma.com/ (дата обращения: 25.10.2022)
3. Metanit [Электронный ресурс]. – Работа с базами данных SQLite. – URL: https://metanit.com/java/android/14.5.php (дата обращения: 10.02.2023)
4. Betacode.net [Электронный ресурс]. – Руководство Android Chip и ChipGroup. – URL: https://betacode.net/12611/android-chip-chipgroup (дата обращения: 20.12.2022)
5. AndroidDevelopers [Электронныйресурс]. – Documentation. – URL: https://developer.android.com/docs (Дата обращения: 05.10.2022)
6. AndroidDevelopers [Электронныйресурс]. – ArrayList. – URL: https://developer.android.com/reference/java/util/ArrayList (Дата обращения: 05.10.2022)
7. Metanit [Электронный ресурс]. –Хранение дат и времени в базе данных. – URL: https://metanit.com/sql/sqlite/6.3.php (дата обращения: 10.02.2023)

Приложение А

(обязательное)

Техническое задание

Министерство образования Новосибирской области

ГБПОУ НСО «Новосибирский авиационный технический колледж

имени Б.С. Галущака»

РАЗРАБОТКА мобильного ДНЕВНИКА ЗДОРОВЬЯ

НАТКиГ.202200.010.000ПЗ

Выполнил:

студент группы

ПР-20.101

Овчаров Д.В.

2023

Содержание

[1 Введение 24](#_Toc138430217)

[2 Назначение разработки 26](#_Toc138430218)

[3 Требования к мобильному приложению 26](#_Toc138430219)

[3.1 Требования к функциональным характеристикам 26](#_Toc138430220)

[3.2 Требования к надёжности 27](#_Toc138430221)

[3.3 Условия эксплуатации 27](#_Toc138430222)

[3.4 Требования к составу и параметрам технических средств 27](#_Toc138430223)

[3.5 Требования к информационной и программной совместимости 27](#_Toc138430224)

[3.6 Требования к защите информации 28](#_Toc138430225)

[3.7 Требования к маркировке и упаковке 28](#_Toc138430226)

[4 Требования к программной документации 28](#_Toc138430227)

[5 Технико-экономические показатели 28](#_Toc138430228)

[6 Стадии и этапы разработки 28](#_Toc138430229)

[7 Порядок контроля и приёмки 29](#_Toc138430230)

1 Введение

Настоящее техническое задание распространяется на разработку приложения «Мобильного дневника здоровья», используемого для записи и просмотра данных о состоянии здоровья.

Наименование приложения: «Дневник Здоровья».

Краткая характеристика области применения: приложение для людей, заботящихся о своем здоровье. Приложение позволяет пользователю записывать данные о своем состоянии здоровья и просматривать их.

Основанием для проведения разработки является Протокол №6 от 21 февраля 2022 года.

Наименование темы разработки – «Разработка мобильного дневника здоровья».

Условное обозначение темы разработки – «Дневник здоровья».

2 Назначение разработки

Основное назначение мобильного приложения заключается в:

* ведении дневника здоровья (добавлении записей в дневник);
* просмотре записей о состоянии здоровья.

Лицо, которое может работать с данной программой – пользователь. Он может добавлять записи в дневник здоровья и просматривать свои записи.

3 Требования к мобильному приложению

## 3.1 Требования к функциональным характеристикам

Требованиями к составу выполняемых функций являются:

* добавление записей о состоянии здоровья;
* просмотр записей дневника.

Пользователь выбирает варианты ответов в опросе. На основе выбранных вариантов ответов формируются входные данные. Выходные данные должны быть организованы в виде текста, выводимого в форме просмотра.

## 3.2 Требования к надёжности

Обеспечение устойчивого функционирования должно выполняться:

* организацией бесперебойного питания оборудования пользователя;
* использованием лицензионного программного обеспечения.

Приложение должно контролировать входную информацию с помощью:

* выбора вариантов ответов на опрос;
* операций изменения и сохранения.

## 3.3 Условия эксплуатации

Обслуживание приложения включает в себя:

* исправление ошибок;
* добавление новых пунктов в дневник здоровья, изменение базы данных.

## 3.4 Требования к составу и параметрам технических средств

Для работы приложения на устройстве требуется:

* платформа Android;
* 256 МБ памяти.

## 3.5 Требования к информационной и программной совместимости

Для работы приложения необходимаверсия Android 5.0 (SDK 21 – Lollipop). Приложение будет поддерживать только русский язык.

## 3.6 Требования к защите информации

База данных хранится на телефоне пользователя. Доступ к базе имеет только пользователь.

## 3.7 Требования к маркировке и упаковке

Требования к маркировке и упаковке не предъявляются.

4 Требования к программной документации

Состав программной документации должен включать в себя:

* техническое задание;
* пояснительная записка.

5 Технико-экономические показатели

Экономические преимущества разработки и ориентировочная экономическая эффективность не рассчитываются.

6 Стадии и этапы разработки

На таблице 1 перечислены стадии разработки приложения.

Таблица 1 – Стадии разработки

|  |  |  |
| --- | --- | --- |
| Название этапа | Срок, даты | Отчётность |
| 1 | 2 | 3 |
| Определение цели и задач, объекта и предмета исследования | 20.09.2022 – 02.10.2022 | Пояснительная записка |
| Описание предметной области | 03.10.2022 – 25.10.2022 | Пояснительная записка |
| Выбор технологии, языка и среды программирования | 26.10.2022 – 18.11.2022 | Пояснительная записка |
| Оформление технического задания | 19.11.2022 – 6.12.2022 | Техническое задание |
| Проектирование UI/UX дизайна | 10.01.2022 – 20.02.2022 | Макет дизайна приложения |
| Разработка мобильного приложения | 21.02.2023 – 24.04.2023 | Программный продукт |

Продолжение таблицы 1

|  |  |  |
| --- | --- | --- |
| Разработка базы данных | 21.02.2023 – 24.03.2023 | Программный продукт |
| Отладка и тестирование приложения | 25.04.2023 – 18.05.2023 | Тексты программных компонентов |
| Оформление документации | 19.05.2023 – 25.06.2023 | Программная  документация |
| Защита | 26.06.2023 |  |

7 Порядок контроля и приёмки

Виды испытаний – защита курсового проекта.

Общими требованиями к приёмке являются:

* техническое задание;
* пояснительная записка;
* программный продукт;
* презентация.

Приложение Б

(справочное)

Словарь данных

Таблица 1 – Сущность «Diary\_Entries»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ключ | Поле | Тип данных | Обязательное | Примечание |
| PK | Date | Text | Да | Дата записи |
|  | Health | Text | Да | Самочувствие |
|  | Note | Text | Нет | Заметка |

Таблица 2 – Сущность «Diary\_Entries\_Worries»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ключ | Поле | Тип данных | Обязательное | Примечание |
| PK | Date | Text | Да | Дата записи. Внешний ключ к таблице Diary\_Entries |
|  | Worry | Text | Да | Что беспокоит |

Таблица 3 – Сущность «Diary\_Entries\_Tracts»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ключ | Поле | Тип данных | Обязательное | Примечание |
| PK | Date | Text | Да | Дата записи. Внешний ключ к таблице Diary\_Entries |
|  | Tract | Text | Да | Симптомы ЖКТ |

Таблица 4 – Сущность «Diary\_Entries\_Moods»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ключ | Поле | Тип данных | Обязательное | Примечание |
| PK | Date | Text | Да | Дата записи. Внешний ключ к таблице Diary\_Entries |
|  | Mood | Text | Да | Настроение |

Таблица 5 – Сущность «Diary\_Entries\_Feels»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ключ | Поле | Тип данных | Обязательное | Примечание |
| PK | Date | Text | Да | Дата записи. Внешний ключ к таблице Diary\_Entries |
|  | Feel | Text | Да | Ощущения |

Таблица 5 – Сущность «Diary\_Entries\_Alarm»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ключ | Поле | Тип данных | Обязательное | Примечание |
| PK | Date | Text | Да | Дата записи. Внешний ключ к таблице Diary\_Entries |
|  | Alarm | Text | Да | Тревога |

Приложение В

(справочное)

Листинги разработанных функций

Листинг 1 – Добавление/перезаписывание записи

chipSave.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 String dateString = getDate();*//Получение строки даты* if(rgFeel1.getCheckedRadioButtonId()!=-1 || rgFeel2.getCheckedRadioButtonId()!=-1){*//выбрано самочувствие* int rbInt=-1;  
 if(rgFeel1.getCheckedRadioButtonId()==-1){*//если есть выбранный RadioButton в первой группе* rbInt = rgFeel2.getCheckedRadioButtonId();*//получить id* }  
 else{*//если есть во второй группе* rbInt = rgFeel1.getCheckedRadioButtonId();*//получить id* }  
 RadioButton rb = findViewById(rbInt);*//получение RadioButton по id* String health = rb.getText().toString();*//Получение значения о самочуствии* String note = editTextNote.getText().toString();*//получение заметок* SQLiteDatabase db = getBaseContext().openOrCreateDatabase("diary.db", *MODE\_PRIVATE*,null);  
 db.execSQL("DELETE FROM Diary\_Entries WHERE date = '"+dateString+"'");*//удаление всех записей за текущую дату* db.execSQL("DELETE FROM Diary\_Entries\_Worries WHERE date = '"+dateString+"'");  
 db.execSQL("DELETE FROM Diary\_Entries\_Tracts WHERE date = '"+dateString+"'");  
 db.execSQL("DELETE FROM Diary\_Entries\_Moods WHERE date = '"+dateString+"'");  
 db.execSQL("DELETE FROM Diary\_Entries\_Feels WHERE date = '"+dateString+"'");  
 db.execSQL("DELETE FROM Diary\_Entries\_Alarms WHERE date = '"+dateString+"'");  
 db.execSQL("INSERT OR IGNORE INTO Diary\_Entries VALUES ('"+ dateString +"', '"+health+"', '"+note+"')");*//Добавление инфо о самочуствии и заметках* List<Integer> idsChipWorries = chipGroupWorries.getCheckedChipIds();*//Получение списка id выбранных Chip* for (Integer id:idsChipWorries){*//Проход по всем id выбранных Chip* Chip chip = chipGroupWorries.findViewById(id);*//получение Chip* db.execSQL("INSERT OR IGNORE INTO Diary\_Entries\_Worries VALUES ('"+dateString+"','"+chip.getText().toString()+"')");*//Добавление инфо о беспокойствах* }  
 List<Integer> idsChipTracts = chipGroupTracts.getCheckedChipIds();*//Получение списка id выбранных Chip* for (Integer id:idsChipTracts){*//Проход по всем id выбранных Chip* Chip chip = chipGroupTracts.findViewById(id);*//получение Chip* db.execSQL("INSERT OR IGNORE INTO Diary\_Entries\_Tracts VALUES ('"+dateString+"','"+chip.getText().toString()+"')");*//Добавление инфо о симптомах ЖКТ* }  
 List<Integer> idsChipMoods = chipGroupMoods.getCheckedChipIds();*//Получение списка id выбранных Chip* for (Integer id:idsChipMoods){*//Проход по всем id выбранных Chip* Chip chip = chipGroupMoods.findViewById(id);*//получение Chip* db.execSQL("INSERT OR IGNORE INTO Diary\_Entries\_Moods VALUES ('"+dateString+"','"+chip.getText().toString()+"')");*//Добавление инфо о настроении* }  
 List<Integer> idsChipFeels = chipGroupFeels.getCheckedChipIds();*//Получение списка id выбранных Chip* for (Integer id:idsChipFeels){*//Проход по всем id выбранных Chip* Chip chip = chipGroupFeels.findViewById(id);*//получение Chip* db.execSQL("INSERT OR IGNORE INTO Diary\_Entries\_Feels VALUES ('"+dateString+"','"+chip.getText().toString()+"')");*//Добавление инфо о ощущении* }  
 List<Integer> idsChipAlarms = chipGroupAlarms.getCheckedChipIds();*//Получение списка id выбранных Chip* for (Integer id:idsChipAlarms){*//Проход по всем id выбранных Chip* Chip chip = chipGroupAlarms.findViewById(id);*//получение Chip* db.execSQL("INSERT OR IGNORE INTO Diary\_Entries\_Alarms VALUES ('"+dateString+"','"+chip.getText().toString()+"')");*//Добавление инфо о тревоге* }  
 Intent intent = new Intent(AddActivity.this, MainActivity.class);*//переход на главную страницу* startActivity(intent);  
 }else{*//если не указано самочувствие* Toast.*makeText*(getBaseContext(),"Укажите ваше самочувствие",Toast.*LENGTH\_LONG*).show();  
 }  
 }  
});

Листинг 2 – Просмотр записи за выбранную дату

private void loadActivity(LocalDate date){  
 setDate(date);  
 if(LocalDate.*now*().equals(date))*//если выбранна текущая даты* btnDateNext.setVisibility(View.*GONE*);*//Убрать кнопку выбора следующей даты* else  
 btnDateNext.setVisibility(View.*VISIBLE*);*//Вернуть кнопку выбора следующей даты* DateTimeFormatter formatter = DateTimeFormatter.*ofPattern*("yyyy-MM-dd");  
 String dateString = date.format(formatter);*//Преобразование даты в строку* SQLiteDatabase db = getBaseContext().openOrCreateDatabase("diary.db",*MODE\_PRIVATE*, null);*//получение данных за выбранную дату* Cursor query = db.rawQuery("SELECT health,note FROM Diary\_Entries WHERE date ='"+dateString+"'",null);  
 Cursor queryWorry = db.rawQuery("SELECT worry FROM Diary\_Entries\_Worries WHERE date = '"+dateString+"'",null);  
 Cursor queryTract = db.rawQuery("SELECT tract FROM Diary\_Entries\_Tracts WHERE date = '"+dateString+"'",null);  
 Cursor queryMood = db.rawQuery("SELECT mood FROM Diary\_Entries\_Moods WHERE date = '"+dateString+"'",null);  
 Cursor queryFeel = db.rawQuery("SELECT feel FROM Diary\_Entries\_Feels WHERE date = '"+dateString+"'",null);  
 Cursor queryAlarms = db.rawQuery("SELECT alarm FROM Diary\_Entries\_Alarms WHERE date = '"+dateString+"'",null);  
  
 String health = "";  
 String note = "";  
 if(query.getCount()>0){*//если есть записи* query.moveToFirst();  
 health = query.getString(0);*//заполнение инфо. о самочуствии и заметках* note = query.getString(1);  
 }  
 if (health.isEmpty())*//если нет записи о самочувствии* tvHealth.setText("Нет записи");*//вывод, что нет записи* else  
 tvHealth.setText(health);*//заполнение самочуствия* if(note.isEmpty())*//если нет заметок* tvNote.setText("Пусто");*//выводим пусто* else  
 tvNote.setText(note);*//вывод заметки* String worry= "";  
 for(int i =0; i<queryWorry.getCount();i++){*//получение всех записей о беспокойствах* queryWorry.moveToPosition(i);  
 if (queryWorry.getString(0)!=null)*//если есть значение* if(worry.isEmpty())*//если строка о беспокойствах пуста* worry += queryWorry.getString(0);  
 else*//если есть данные в строке* worry +="\n"+queryWorry.getString(0);  
 }  
 if(worry.isEmpty())*//если нет данных* worry = "Ничего";  
 tvWorry.setText(worry);  
  
 String tract = "";  
 for(int i=0;i<queryTract.getCount();i++){*//получение всех записей о симптомах ЖКТ* queryTract.moveToPosition(i);  
 if(tract.isEmpty())*//если строка о симптомах пуста* tract = queryTract.getString(0);  
 else*//если есть данные в строке* tract += "\n" + queryTract.getString(0);  
 }  
 if (tract.isEmpty())*//если нет данных* tract = "Ничего";  
 tvTract.setText(tract);  
  
 String mood = "";  
 for(int i=0;i<queryMood.getCount();i++){*//получение всех записей о настроении* queryMood.moveToPosition(i);  
 if(mood.isEmpty())*//если строка о настроении пуста* mood = queryMood.getString(0);  
 else*//если есть данные в строке* mood += "\n" + queryMood.getString(0);  
 }  
 if (mood.isEmpty())*//если нет данных* mood = "Ничего";  
 tvMood.setText(mood);  
  
 String feel = "";  
 for(int i=0;i<queryFeel.getCount();i++){*//получение всех записей о ощущениях* queryFeel.moveToPosition(i);  
 if(feel.isEmpty())*//если строка о ощущениях пуста* feel = queryFeel.getString(0);  
 else*//если есть данные в строке* feel += "\n" + queryFeel.getString(0);  
 }  
 if (feel.isEmpty())*//если нет данных* feel = "Ничего";  
 tvFeel.setText(feel);  
  
 String alarm = "";  
 for (int i =0; i<queryAlarms.getCount();i++){*//получение всех записей о тревоге* queryAlarms.moveToPosition(i);  
 if(alarm.isEmpty())*//если строка о тревоге пуста* alarm = queryAlarms.getString(0);  
 else*//если есть данные в строке* alarm += "\n" + queryAlarms.getString(0);  
 }  
 if (alarm.isEmpty())*//если нет данных* alarm = "Ничего";  
 tvAlarm.setText(alarm);  
}

Листинг 3 – Просмотр сводки записей за выбранный месяц

private void loadActivity(LocalDate date) {  
 setDate(date);  
 if (LocalDate.*now*().getMonth().equals(date.getMonth()))*//если выбрань текущий месяц* btnDateNext.setVisibility(View.*GONE*);*//убрать кнопку выбора следующего месяца* else  
 btnDateNext.setVisibility(View.*VISIBLE*);*//вернуть кнопку выбора следующего месяца* LocalDate firstDay = date.withDayOfMonth(1);  
 LocalDate lastDay = date.withDayOfMonth(date.lengthOfMonth());*//получение первой и последней даты месяца* DateTimeFormatter formatter = DateTimeFormatter.*ofPattern*("yyyy-MM-dd");  
 String dateString = date.format(formatter);  
 SQLiteDatabase db = getBaseContext().openOrCreateDatabase("diary.db", *MODE\_PRIVATE*, null);*//поиск данных за выбранный период* Cursor query = db.rawQuery("SELECT health FROM Diary\_Entries WHERE date between '"+firstDay+"' and '"+lastDay+"'",null);  
 Cursor queryWorry = db.rawQuery("SELECT worry FROM Diary\_Entries\_Worries WHERE date between '"+firstDay+"' and '"+lastDay+"'",null);  
 Cursor queryTract = db.rawQuery("SELECT tract FROM Diary\_Entries\_Tracts WHERE date between '"+firstDay+"' and '"+lastDay+"'",null);  
 Cursor queryMood = db.rawQuery("SELECT mood FROM Diary\_Entries\_Moods WHERE date between '"+firstDay+"' and '"+lastDay+"'",null);  
 Cursor queryFeel = db.rawQuery("SELECT feel FROM Diary\_Entries\_Feels WHERE date between '"+firstDay+"' and '"+lastDay+"'",null);  
 Cursor queryAlarms = db.rawQuery("SELECT alarm FROM Diary\_Entries\_Alarms WHERE date between '"+firstDay+"' and '"+lastDay+"'",null);  
 ArrayList<String> healthList = new ArrayList<>();  
 ArrayList<String> worryList = new ArrayList<>();  
 ArrayList<String> tractList = new ArrayList<>();  
 ArrayList<String> moodList = new ArrayList<>();  
 ArrayList<String> feelList = new ArrayList<>();  
 ArrayList<String> alarmList = new ArrayList<>();  
 while(query.moveToNext()){*//загрузка данных в ArrayList* healthList.add(query.getString(0));  
 }  
 while(queryWorry.moveToNext()){*//загрузка данных в ArrayList* worryList.add(queryWorry.getString(0));  
 }  
 while (queryTract.moveToNext()){*//загрузка данных в ArrayList* tractList.add(queryTract.getString(0));  
 }  
 while (queryMood.moveToNext()){*//загрузка данных в ArrayList* moodList.add(queryMood.getString(0));  
 }  
 while (queryFeel.moveToNext()){*//загрузка данных в ArrayList* feelList.add(queryFeel.getString(0));  
 }  
 while (queryAlarms.moveToNext()){*//загрузка данных в ArrayList* alarmList.add(queryAlarms.getString(0));  
 }  
 String health = "";  
 String worry = "";  
 String tract = "";  
 String mood = "";  
 String feel = "";  
 String alarm = "";  
 while(!healthList.isEmpty()) {*//Если есть данные в ArrayList* String currentElem = healthList.get(0);  
 if(health.isEmpty()){*//если в строке нет данных* health = currentElem +": " + writeWithDeclension(Collections.*frequency*(healthList,currentElem));*//самочувствие + количество повторений этой записи в ArrayList* }else{  
 health +="\n"+ currentElem +": " + writeWithDeclension(Collections.*frequency*(healthList,currentElem));*//аналогично* }  
 while (healthList.remove(currentElem));*//удаление записей добавленного значения* }  
 while(!worryList.isEmpty()) {*//Если есть данные в ArrayList* String currentElem = worryList.get(0);  
 if(worry.isEmpty()){*//если в строке нет данных* worry = currentElem +": " + writeWithDeclension(Collections.*frequency*(worryList,currentElem));*//беспокойство + количество повторений этой записи в ArrayList* }else{  
 worry +="\n"+ currentElem +": " + writeWithDeclension(Collections.*frequency*(worryList,currentElem));*//аналогично* }  
 while (worryList.remove(currentElem));*//удаление записей добавленного значения* }  
 while(!tractList.isEmpty()) {*//Если есть данные в ArrayList* String currentElem = tractList.get(0);  
 if(tract.isEmpty()){*//если в строке нет данных* tract = currentElem +": " + writeWithDeclension(Collections.*frequency*(tractList,currentElem));*//симптом ЖКТ + количество повторений этой записи в ArrayList* }else{  
 tract +="\n"+ currentElem +": " + writeWithDeclension(Collections.*frequency*(tractList,currentElem));*//аналогично* }  
 while (tractList.remove(currentElem));*//удаление записей добавленного значения* }  
 while(!moodList.isEmpty()) {*//Если есть данные в ArrayList* String currentElem = moodList.get(0);  
 if(mood.isEmpty()){*//если в строке нет данных* mood = currentElem +": " + writeWithDeclension(Collections.*frequency*(moodList,currentElem));*//настроение + количество повторений этой записи в ArrayList* }else{  
 mood +="\n"+ currentElem +": " + writeWithDeclension(Collections.*frequency*(moodList,currentElem));*//аналогично* }  
 while (moodList.remove(currentElem));*//удаление записей добавленного значения* }  
 while(!feelList.isEmpty()) {*//Если есть данные в ArrayList* String currentElem = feelList.get(0);  
 if(feel.isEmpty()){*//если в строке нет данных* feel = currentElem +": " + writeWithDeclension(Collections.*frequency*(feelList,currentElem));*//ощущение + количество повторений этой записи в ArrayList* }else{  
 feel +="\n"+ currentElem +": " + writeWithDeclension(Collections.*frequency*(feelList,currentElem));*//аналогично* }  
 while (feelList.remove(currentElem));*//удаление записей добавленного значения* }  
 while(!alarmList.isEmpty()) {*//Если есть данные в ArrayList* String currentElem = alarmList.get(0);  
 if(alarm.isEmpty()){*//если в строке нет данных* alarm = currentElem +": " + writeWithDeclension(Collections.*frequency*(alarmList,currentElem));*//тревога + количество повторений этой записи в ArrayList* }else{  
 alarm +="\n"+ currentElem +": " + writeWithDeclension(Collections.*frequency*(alarmList,currentElem));*//аналогично* }  
 while (alarmList.remove(currentElem));*//удаление записей добавленного значения* }  
 if(!health.isEmpty()){*//Если строка заполнена* tvHealth.setText(health);  
 }else{*//Если нет данных* tvHealth.setText("Данные за период отсутствуют");  
 }  
 if(!worry.isEmpty()){*//Если строка заполнена* tvWorry.setText(worry);  
 }else{*//Если нет данных* tvWorry.setText("Данные за период отсутствуют");  
 }  
 if(!tract.isEmpty()){*//Если строка заполнена* tvTract.setText(tract);  
 }else{*//Если нет данных* tvTract.setText("Данные за период отсутствуют");  
 }  
 if(!mood.isEmpty()){*//Если строка заполнена* tvMood.setText(mood);  
 }else{*//Если нет данных* tvMood.setText("Данные за период отсутствуют");  
 }  
 if(!feel.isEmpty()){*//Если строка заполнена* tvFeel.setText(feel);  
 }else{*//Если нет данных* tvFeel.setText("Данные за период отсутствуют");  
 }  
 if(!alarm.isEmpty()){*//Если строка заполнена* tvAlarm.setText(alarm);  
 }else{*//Если нет данных* tvAlarm.setText("Данные за период отсутствуют");  
 }  
}  
private String writeWithDeclension(int num){*//выбор склонения в зависимости от числа* if(num%10==0 || num%10==1 || (num%10>=5 && num%10<=9) || (num%100>=10 && num%100<=20)){  
 return num + " раз";  
 }else{  
 return num + " раза";  
 }  
}