Warszawa, 15.01.2019

POLITECHNIKA WARSZAWSKA

WYDZIAŁ ELEKTRONIKI I TECHNIK INFORMACYJNYCH

SYSTEMY TELEMEDYCZNE

Dokumentacja końcowa

Prowadzący przedmiot: dr inż. Robert Kurjata

Prowadzący projekt: mgr inż. Tomasz Olszewski

Autorzy: Tomasz Kopacz i Adam Jackowski

Spis treści

1. Temat

Aplikacja mobilna - podręczny przewodnik pacjenta z kalendarzem, przypominający o zaleconych lekach i pomiarach (np. temperatury, ciśnienia) oraz umożliwiający zapis wyników pomiarów. Aplikacja nosi nazwę *Kalendarz pacjenta*.

2. Cel projektu

Celem pracy jest zaprojektowanie i zaimplementowanie aplikacji zapisującej wyniki pomiarów parametrów życiowych, przypominającej o przyjęciu leków danego dnia oraz podającej informacje o umówionej wizycie lekarskiej.

Głównym założeniem jest ułatwienie funkcjonowania osobom, które przyjmują dużo leków o różnych porach. Aplikacja jest adresowana również do pacjentów wykonujących regularne pomiary ciśnienia czy temperatury, ponieważ umożliwia zapis wyników badań. Interfejs użytkownika powinien być prosty i intuicyjny, a funkcjonalności łatwo dostępne.

3. Ogólny opis funkcjonowania aplikacji

Aplikacja składa się z okna głównego oraz okien dodatkowych funkcjonalności. W oknie głównym wyświetlane są wszystkie zadania ustalone na wybrany dzień w kolejności chronologicznej. Użytkownik po wykonaniu zadania może oznaczyć je jako wykonane. W przypadku pomiarów musi podać wartość uzyskanych wyników. Użytkownik może dodać nowe zadanie wciskając przycisk “*Plus*”. Następuje przekierowanie z okna głównego do okna wyboru typu zadania, a następnie do okna szczegółowych informacji o zadaniu. Aplikacja daje możliwość wstępnej analizy danych w postaci wykresów wykonanych pomiarów. Ponadto, w momencie, gdy nadchodzi pora wykonania zadania, w telefonie użytkownika jest to sygnalizowane poprzez wysyłanie przypomnienia, nawet gdy aplikacja jest wyłączona.

4. Dokumentacja techniczna

4.1. Narzędzia pracy

Wybrane środowisko pracy (IDE) to *Android Studio 3.2.1.* Aplikacja jest napisana w języku JAVA. Interfejs graficzny jest przygotowany za pomocą języka znacznikowego XML. Projekt wykorzystuje proste zapytania bazodanowe SQL. Budowanie projektu odbywa się za pomocą narzędzia *Gradle*. Współpraca przy tworzeniu projektu jest wspomagana poprzez system kontroli wersji Git, a repozytorium zdalne umieszczone jest na portalu *Gitlab*.

4.2. Wymagania

Aplikacja jest przeznaczona dla urządzeń typu Android wyposażonych w wersję KitKat lub nowszą.

4.3. Instrukcja kompilacji

1. Należy przejść do folderu zawierającego pliki projektu, w tym narzędzie do budowania projektów *Gradle*.
2. W folderze projektu należy uruchomić wiersz poleceń.
3. Następnie należy wpisać polecenie "*gradlew tasks*". W konsoli pojawi się lista zadań możliwych do wykonania.
4. W celu kompilacji należy wpisać polecenie "*gradlew build*". Projekt zostanie skompilowany oraz zbudowany.
5. W celu wgrania aplikacji na telefon należy włączyć na nim opcje programisty, a następnie podłączyć do komputera. W wierszu poleceń należy wpisać polecenie "*gradlew installDebug*".

4.4. Wykorzystane biblioteki zewnętrzne

Room – biblioteka umożliwiająca wykonywanie zapytań do bazy danych przechowywanej w urządzeniu.

SwipeMenuListView – biblioteka umożliwiająca sprawne przeglądanie elementów w postaci listy oraz wykonywanie na nich dodatkowej akcji w postaci przesuwania na bok.

GraphView – biblioteka umożlwiająca rysowanie wykresów.

4.5. Architektura projektu

Kod źródłowy programu został podzielony za względu na pełnione funkcje. Każdy pakiet zawiera ściśle ze sobą powiązane klasy i interfejsy, które wspólnie tworzą moduł odpowiedzialny za ściśle określoną funkcjonalność:

- Pakiet *mainwindow* - odpowiada za przedstawienie okna z listą zadań ustalonych na aktualny dzień. Umożliwia też przejścia do innych widoków, takich jak formularz dodania nowego zadania, lista zadań w dowolnym dniu z kalendarza, wyświetlenie wykresów z wynikami pomiarów oraz okna z informacjami. Pakiet zawiera również okna dialogowe wyświetlające dodatkowe informacje o zadaniu oraz umożliwiające podanie wartości pomiarów

- Pakiet *newtaskwindow* – odpowiada za dodanie nowych zadań do kalendarza. Zadania mogą być różnego typu: pomiar, przypomnienie o leku, wizyta u lekarza. Wraz z dodaniem zadania ustalane jest przypomnienie (notyfikacja), które zostanie wyświetlone użytkownikowi dokładnie o godzinie, w której powinien je wykonać
- Pakiet *chartwindow* - odpowiada za wyświetlenie wykresów
- Pakiet *data* - zawiera implementację bazy danych oraz umożliwia komunikację z nią
- Pakiet *notification* – zawiera klasy odpowiadające za ustalanie notyfikacji i wysyłanie ich o określonej porze
- Pakiet *utils* - zawiera dodatkowe klasy, pełniące funkcje pomocnicze, np. formatowanie tekstu na datę

4.6. Okno główne

MainActivity - jest to aktywność stanowiąca główne okno aplikacji. Odpowiada za wyświetlanie listy zadań ustalonych na dany dzień. Zostało to wykonywane poprzez umieszczenie we wnętrzu tej aktywności odpowiedniego widoku, zwanego fragmentem (klasa *Fragment*). Tutaj określony zostaje również pasek narzędzi, który umożliwia przejście do okien: wyboru dnia z kalendarza, rysowania wykresów czy wyświetlania informacji o programie.

- *onCreate()* - ustawienie domyślnego wyglądu okna
- *changeFragment(Fragment fragment)* - zmiana widoku
- *onCreateOptionsMenu(Menu menu)* - utworzenie paska zadań
- *onOptionsItemSelected(MenuItem item)* - interakcja użytkownika z paskiem zadań

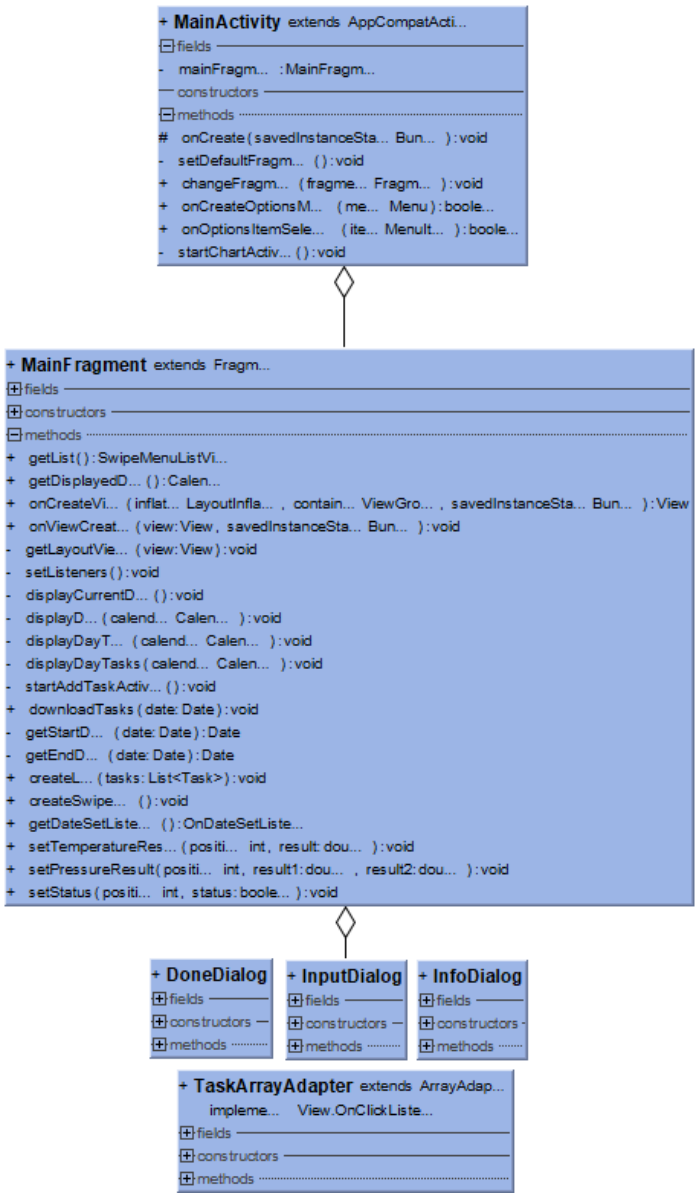
MainFragment – domyślny fragment umieszczony we wnętrzu *MainActivity.* Znajdują się w nim metody służące do inicjowania listy zadań i jej wyświetlenia. Tworzenie listy wykonywane jest poprzez pobranie informacji z bazy danych. Przepisaniem danych z bazy do listy zadań zajmuje się adapter *TaskArrayAdapter.* Zapewniono interakcję użytkownika z zadaniami znajdującymi się na liście - zaimplementowano obiekty (*listenery*), nasłuchujące czy wykonana została określona akcja, taka jak naciśnięcie, długie przytrzymanie, przesunięcie. W ich wyniku wykonywane są odpowiednie funkcje.

- *onCreate()* - ustawienie domyślnego wyglądu i ustawienie *listenerów*
- *downloadTasks(Date date)* - pobranie zadań z bazy danych na podstawie podanej daty i wyświetlenie ich na liście

W wyniku naciśnięcia wyświetlane jest okno dialogowe z informacjami o zadaniu, po długim przytrzymaniu ukazuje się okno dialogowe z możliwością wprowadzenia wartości lub oznaczenia zadania jako wykonanego, a po przesunięciu elementu w lewo następuje usunięcie zadania zarówno z listy jak i bazy danych.

We fragmencie *MainFragment* zamieszczono także przycisk i zdefiniowano *listenera,* który reaguje na jego wciśnięcie. Przycisk ten umożliwia przejście do okna dodania nowego zadania.

Uproszczony diagram UML:



4.7. Okno dodania nowego zadania

AddTaskActivity – aktywność umożliwiająca użytkownikowi dodanie nowego zadania. Po jej otwarciu ukazuje widok prezentujący 3 typy zadań: pomiar, lek, wizyta. Użytkownik wybiera jeden z nich. Wtedy zostaje przekierowany do kolejnego widoku. Jego oczom ukazuje się formularz, gdzie należy wprowadzić dodatkowe informacje (np. w przypadku wizyty lekarskiej należy podać imię i nazwisko lekarza, godzinę i datę). Dla każdego typu zadania prezentowany jest inny formularz.

- *onCreate()* - ustawienie domyślnego wyglądu okna
- *changeFragment(Fragment fragment)* - zmiana widoku

TaskTypeFragment – domyślny fragment początkowy aktywności AddTaskActivity. Odpowiada za prezentację 3 typów zadań do wyboru. Po kliknięciu na jeden z nich następuje przekierowanie do kolejnego widoku z odpowiednim formularzem.

- *showMeasurementView()* – przekierowanie do formularza pomiaru
- *showDrugView()* – przekierowanie do formularza leku
- *showExaminationView()* – przekierowanie do formularza badania

MeasurementFragment, DrugFragment, ExaminationFragment – fragmenty zawierające formularze, w których podaje się dane definiujące nowe zadanie.

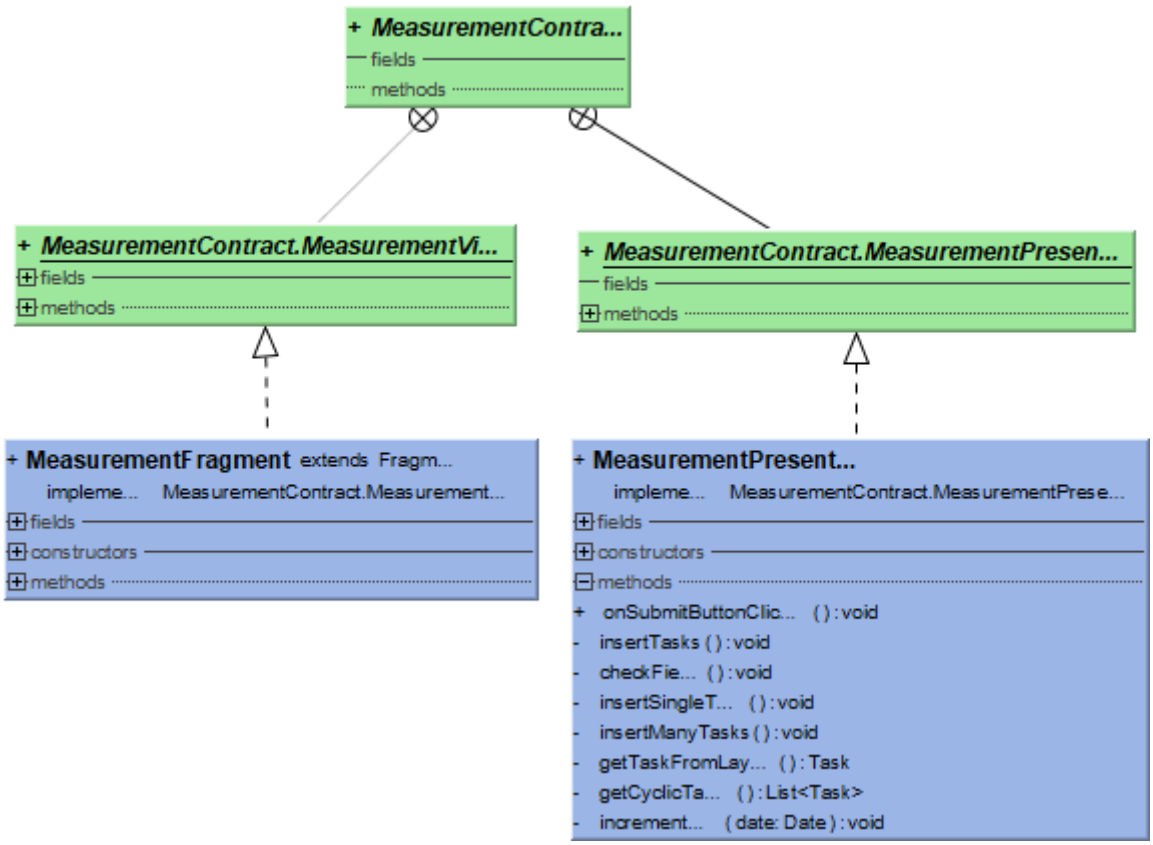
MeasurementPresenter, DrugPresenter, ExaminationPresenter – klasy prezenterów. Zawierają logikę przetwarzania danych z formularzy. Na podstawie podanych przez użytkownika informacji tworzony jest nowy obiekt zadania. Po kliknięciu na przycisk potwierdzenia następuje dodanie zadania do bazy danych.

- *getTaskFromLayout()* – utworzenie obiektu zadania na podstawie danych z formularzy
- *checkFields()* – walidacja wprowadzonych danych i obsługa błędów
- *onSubmitButtonClicked()* – funkcja wywoływana po kliknięciu na przycisk „AKCEPTUJ”
- *insertTasks()* – dodanie zadań do bazy danych

MeasurementContract, DrugContract, ExaminationContract – interfejsy, których zadaniem jest ścisłe połączenie klas odpowiadających za widok (fragmenty) z klasami odpowiadającymi za logikę (prezenterzy). Stosowanie takich interfejsów ułatwia implementację wzorca *Model-View-Presenter.*

Uproszczony diagram UML:

Przedstawiono diagram dla modułu dotyczącego jednego typu zadania: pomiaru. Dla pozostałych typów można wygenerować analogiczne diagramy. Ponadto, przedstawiony poniżej fragment, podobnie jak wszystkie pozostałe w tym podrozdziale, zawierają się w aktywności *AddTaskActivity.* Taką zależność prezentuje diagram podobny do tego przedstawionego w poprzednim podrozdziale, dlatego nie jest tu zaprezentowany.



4.8. Okno zawierające wykresy

ChartActivity – aktywność przechowująca domyślny wygląd. Jej głównym zadaniem jest ustawienie fragmentu definiującego wygląd okna zawierającego wykresy.

- *setDefaultFragment(Fragment fragment)* – ustawienie domyślnego fragmentu

ChartFragment, ChartPresenter – odpowiada za tworzenie wykresów na podstawie danych z pomiarów. Tworzone są dwa wykresy: temperatury oraz ciśnienia. Na tym drugim rysowane są dwa przebiegi. Jeden z nich dotyczy ciśnienia skurczowego, a drugi rozkurczowego.

- *onCreate()* – tworzenie domyślny wyglądu wykresów
- *drawChart(LiveData<List<Tasks>> list)* – tworzenie wykresu na podstawie listy zadań

ChartContract **–** interfejs definiujący podstawowe metody fragmentu i prezentera, umożliwiające ich komunikację

4.9. Okno informacji

AppInfoDialog – klasa definiująca okno dialogowe, służące do wyświetlenia informacji o aplikacji, takich jak numer wersji i autorzy.

- *infoDialog()* – zwraca obiekt klasy *AlertDialog*, w którym znajdują się odpowiednie informacje

4.10. Baza danych

Task – klasa definiująca obiekt zadania. Zawiera takie pola jak: status, typ zadania, znacznik czasowy (data i godzina), wyniki pomiarów, imię i nazwisko lekarza i wiele innych. Obiekty tej klasy reprezentują encje przechowywane w bazie danych. Klasa posiada kilka konstruktorów i metody dostępowe (gettery i settery).

TaskDAO **–** interfejs definiujący metody odpowiadające zapytaniom SQL do bazy danych. Interfejs ten jest opatrzony etykietą *@Dao*. Dzięki niej biblioteka *Room* generuje metody, służące do wykonywania zapytań do bazy danych.

- *getAll()* – pobranie wszystkich zadań z bazy danych. Pobrane zadania są uporządkowane rosnąco według daty
- *insert(), update(), delete()* – metody służące do wykonania operacji CRUD na bazie danych

TaskDatabase – klasa stanowiąca obiekt bazy danych. Jest stworzona według wzorca *Singleton*, więc istnieje tylko jeden taki obiekt w trakcie działania aplikacji.

TaskRepository – klasa dostępowa do interfejsu *TaskDao.* Umożliwia bardziej złożone operacje na danych bazodanowych. Tutaj wywoływane są również metody interfejsu *TaskDao,* które ze względu na to, że odwołują się do bazy danych, muszą zostać wywołane w oddzielnych wątkach. Dzięki temu praca aplikacji nie zostanie zaburzona.

4.11. Alarmy i notyfikacje

TaskAlarm – w tej klasie następuje zdefiniowanie alarmu, na podstawie którego w określonym momencie ma pojawić się notyfikacja. Czas wykonania zadania przechowywany jest w obiekcie *Task*. Zostaje on przekazany do menedżera alarmów w urządzeniu. Jednak żeby notyfikacja została

wyświetlona, potrzebne jest również zdefiniowanie odbiornika, który wywoła alarm, gdy nadejdzie odpowiednia pora.

- *setAlarm(Context context, Task, task)* – wysyła alarm dotyczący określonego zadania do menedżera alarmów urządzenia

TaskAlarmReceiver – odbiornik alarmów. Gdy wywołany zostanie alarm, odbiornik go odbierze i wyśle powiadomienie. W odbiorniku następuje implementacja „wyglądu" notyfikacji: wyświetlany tekst, ikona, postać wibracji czy kolor diody powiadamiającej. Zdefiniowana zostaje także akcja otwarcia aplikacji *Kalendarz pacjenta* w momencie kliknięcia na notyfikację.

- *onReceive(Context context, Intent intent)* – funkcja wywoływana po otrzymaniu alarmu
- *createNotification(Context context, Intent intent)* – tworzenie powiadomienia i definicja jego wyglądu
- *sendNotification(Context context, String title, String message)* – wysyłanie powiadomienia

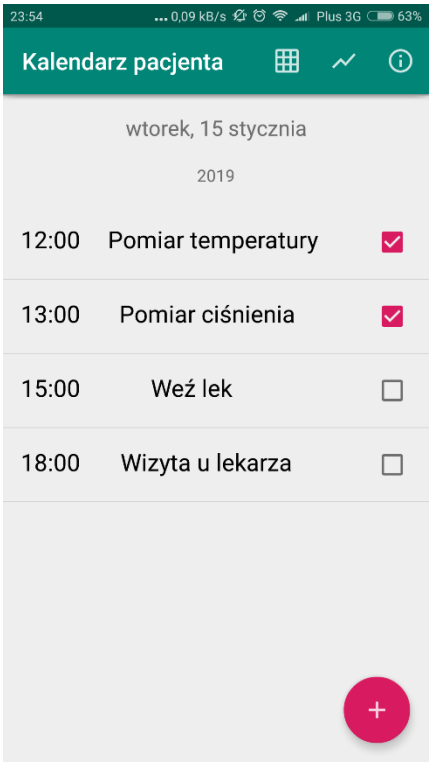
5. Instrukcja użytkownika

5.1. Instalacja

W celu zainstalowania aplikacji na telefonie należy wgrać plik z rozszerzeniem *.apk* na urządzenie. Aby było to możliwe, należy wcześniej zezwolić urządzeniu na instalowanie aplikacji z nieznanych źródeł. Następnie należy połączyć urządzenie z komputerem i skopiować plik *.apk.* Gdy plik zostanie przeniesiony, należy go uruchomić i kliknąć „INSTALUJ”.

5.2. Okno główne aplikacji

Po uruchomieniu aplikacji ukazuje się ekran główny. Zawiera trzy części:



Pasek menu – zawiera nazwę aplikacji 3 ikony:

- kalendarz - po naciśnięciu pojawia się okno z kalendarzem. Wybór dnia spowoduje wyświetlanie się zadań ustalonych na ten dzień
- wykres - po naciśnięciu nastąpi przejście do okna wykresów. Okno umożliwia przeglądanie ostatnich pomiarów ciśnienia i temperatury z dowolnego okresu
- informacje – po naciśnięciu pojawia się okno informacji o aplikacji

Data oraz lista zadań

Ta część stanowi główny widok okna. Wyświetlana jest aktualna data, którą można zmienić korzystając z ikonki kalendarza w pasku menu. Poniżej daty wyświetlana jest lista zadań. Po pierwszym uruchomieniu aplikacji jest pusta, a zamiast zadań wyświetla się tekst „BRAK ZADAŃ”.

Po dodaniu zadań (instrukcja, jak to zrobić znajduje się niżej) zostaną one wyświetlone na liście. Zamknięcie aplikacji nie powoduje utraty danych, ponieważ są one zapisywane w pamięci urządzenia.

Każda pozycja na liście składa się z kilku komponentów: po lewej stronie widoczna jest godzina wykonania zadania. W środkowej części znajduje się tekst typu zadania, a po prawej stronie *„checkbox”*, informujący, czy zadanie zostało wykonane.

Każde zadanie z listy można oznaczyć jako wykonane (lub pomiar jako zmierzony). W tym celu należy nacisnąć na odpowiednią pozycję i przytrzymać do czasu pojawienia się okna dialogowego. Następnie należy postępować zgodnie z informacją widoczną w oknie dialogowym.

Po krótkim naciśnięciu na zadanie z listy pojawi się okno dialogowe z informacją o danym zadaniu.

Zadanie można również usunąć. Aby to zrobić, należy przesunąć zadanie palcem w stronę lewej krawędzi ekranu. Po prawej stronie pojawi się czerwona ikona. Po naciśnięciu na nią, zadanie zostanie usunięte.

Przycisk ”Dodaj zadanie”

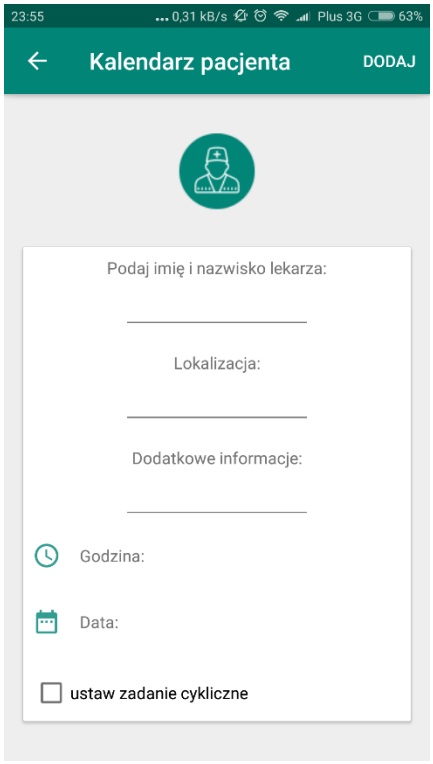
Przycisk ma okrągły kształt z znakiem “+” w środku. Służy do dodawania zadań do listy. Po naciśnięciu otwiera się nowe okno z wyborem zadania.

5.3. Dodawanie zadań

Po pierwszym uruchomieniu aplikacji nie są zdefiniowane żadne zadania. Aby dodać nowe zadanie należy w oknie głównym nacisnąć przycisk z ikoną „+”. Otwiera się wówczas nowe okno, służące do wyboru typu zadania. Składa się z trzech części, odpowiadających trzem rodzajom zadań: pomiar, przypomnienie o leku, przypomnienie o wizycie u lekarza. Należy wybrać jedną z tych opcji i ją nacisnąć.

Pojawi się okno z formularzem do wypełnienia. Zawiera on istotne informacje o zadaniu. Dla każdego zadania formularz jest inny, jednak w przypadku każdego zadania należy koniecznie wybrać godzinę oraz datę jego wykonania. Wykonuje się to poprzez naciśnięci na ikonkę zegara lub kalendarza.

Ciekawą funkcjonalnością jest możliwość ustawienia zadania cyklicznego. Umożliwia ona dodanie wielu jednakowych zadań odbywających się o tej samej godzinie w pewnym przedziale czasowym. Na przykład, jeżeli chcemy, aby aplikacja przypominała nam o wzięciu leku codziennie o godzinie 12:00 przez dwa tygodnie, nie musimy dodawać czternastu zadań osobno. Zamiast tego należy zaznaczyć pole „ustaw zadanie cykliczne” znajdujące się na dole formularza. Pojawi się wówczas dodatkowe pole z ikonką kalendarza, które służy do określenia końcowej daty wykonywania zadania.



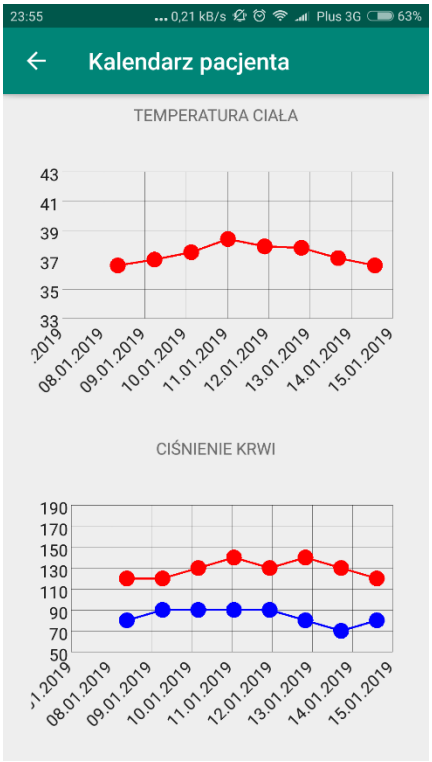
Aby dodać zadanie lub zadania do listy należy nacisnąć przycisk “DODAJ”, znajdujący się u góry na pasku menu po prawej stronie. Po jej naciśnięciu zadania zostaną dodane do bazy i nastąpi powrót do ekranu głównego. Zostanie wyświetlona lista zadań na wybrany wcześniej dzień.

5.4. Notyfikacje

Za każdym razem, gdy nastaje pora wykonania zadania, w urządzeniu generowane jest przypomnienie o jego wykonaniu. Danego dnia o podanej godzinie w smartfonie pojawia się notyfikacja, że należy wykonać pomiar, wziąć lek lub zbliża się wizyta u lekarza. Wraz z wystąpieniem notyfikacji telefon wibruje, a jeżeli urządzenie posiada diodę, to świeci się ona kolorem morskim. Notyfikacja pojawia się nawet gdy aplikacja *Kalendarz pacjenta* nie jest aktywna a ekran jest wygaszony.

5.5. Wykresy

Aplikacja umożliwia wstępną analizę danych z wykonanych pomiarów w postaci automatycznego generowania wykresów. Aby je wyświetlić należy w oknie głównym kliknąć ikonkę wykresu znajdującą się w pasku menu. Wyświetli się wówczas nowe okno, a w nim dwa wykresy: temperatury oraz ciśnienia. W przypadku ciśnienia na jednym wykresie prezentowane są jednocześnie przebiegi ciśnienia skurczowego i rozkurczowego. Na osiach poziomych znajdują się daty wykonania danego pomiaru, a na osiach pionowych uzyskane wartości. Użytkownik może przewijać wykresy oraz je przybliżać/oddalać.



5.6. Informacje

W celu uzyskania dodatkowych informacji o aplikacji, należy przejść do odpowiedniego okna. W tym celu należy kliknąć ikonkę „i” znajdującą się w pasku menu ekranu głównego.

6. Podział zadań

Adam Jackowski:

- okno główne - wyświetlanie listy zadań, interakcje użytkownika z elementami listy, okna do wpisywania wyników badania i do oznaczenia zadania jako wykonane
- okno kalendarza
- okno z informacjami

Tomasz Kopacz:

- okno dodawania nowych zadań: wybór rodzaju zadania, okna z formularzami
- okno z wykresami
- tworzenie notyfikacji
- utworzenie bazy danych