AKADEMIA GÓRNICZO-HUTNICZA

IM. STANISŁAWA STASZICA W KRAKOWIE



Wydział Inżynierii Metali i Informatyki Przemysłowej



PRACA DYPLOMOWA INŻYNIERSKA

pt.

„Opracowanie aplikacji mobilnej wspomagającej przyjmowanie leków przez osoby chore na stwardnienie rozsiane”

Imię i nazwisko dyplomanta: **Karol Skóra**

Kierunek studiów: **Informatyka Stosowana**

Nr albumu: 260468

Promotor: dr hab. inż. Łukasz Rauch

Recenzent: dr inż. Krzysztof Regulski

Podpis dyplomanta: Podpis promotora:

Kraków 2015

„Uprzedzony o odpowiedzialności karnej na podstawie art. 115 ust. 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.): „Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystyczne wykonanie albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, wideogram lub nadanie.”, a także uprzedzony o odpowiedzialności dyscyplinarnej na podstawie art. 211 ust. 1 ustawy z dnia 27 lipca 2005 r. Prawo o szkolnictwie wyższym (t.j. Dz. U. z 2012 r. poz. 572, z późn. zm.) „Za naruszenie przepisów obowiązujących w uczelni oraz za czyny uchybiające godności studenta student ponosi odpowiedzialność dyscyplinarną przed komisją dyscyplinarną albo przed sądem koleżeńskim samorządu studenckiego, zwanym dalej "sądem koleżeńskim"”, oświadczam, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i że nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.”

Kraków, dnia …… Podpis dyplomanta…………….

Spis treści

[1. Wstęp 4](#_Toc467423813)

[2. Problematyka stwardnienia rozsianego 5](#_Toc467423814)

[3. Istniejące rozwiązania 7](#_Toc467423815)

[4. Projekt oprogramowania 12](#_Toc467423816)

[a. Założenia funkcjonalne i platformowe 12](#_Toc467423817)

[b. Diagramy i wzorce projektowe 14](#_Toc467423818)

[c. Użyte biblioteki 17](#_Toc467423819)

[5. Implementacja 18](#_Toc467423820)

[6. Testy 31](#_Toc467423821)

[7. Podsumowanie 36](#_Toc467423822)

[8. Literatura 36](#_Toc467423823)

# Wstęp

Gwałtowny rozwój techniki oraz postępująca miniaturyzacja spowodowały, iż w dzisiejszych czasach większość z nas we własnej kieszeni posiada telefon komórkowy, który swoją mocą obliczeniową przewyższa komputery stacjonarne sprzed kilku lat. Te wielofunkcyjne urządzenia dawno przestały być tylko narzędziem do dzwonienia i wysyłania wiadomości tekstowych. Większość nowoczesnych smartphone’ów bez problemu radzi sobie z wyświetlaniem grafiki 3D, skomplikowanymi obliczeniami, wyświetlaniem notyfikacji użytkownikowi czy harmonogramowaniem planu dnia. Najczęściej wszystkie wymienione wyżej funkcję służą do dostarczania nam rozrywki, jednak coraz większa część producentów sprzętu oraz oprogramowania zaczyna zauważać w urządzeniach mobilnych bardzo duży potencjał do ułatwiania życia ludziom chorym i niepełnosprawnym. W ostatnich latach zaczęło powstawać bardzo dużo aplikacji mobilnych, których głównym przeznaczeniem jest pomoc np. osobom niewidomym w poruszaniu się po mieście. Przykładem takich rozwiązań może być „Seeing Assistant move” czyli aplikacja, która potrafi wyszukać trasę, pomaga w poruszaniu się osobie niewidomej, rejestruje przebytą drogę czy opisuje obecną lokalizację. Innym ciekawym zastosowaniem nowoczesnych technologii w pomocy osobom chorym mogą być aplikacje porządkujące wspomnienia i ułatwiające sporządzanie notatek dla osób cierpiących na chorobę Alzheimera.

Dlatego też w niniejszej pracy podjęto próbę stworzenia aplikacji wspomagającej grupę osób, która do tej pory, przynajmniej w momencie tworzenia tej pracy, była pomijana przez twórców oprogramowania czyli dla osób chorych na stwardnienie rozsiane. Stwardnienie rozsiane jest poważną i nieuleczalną chorobą układu nerwowego, w której dochodzi do nieodwracalnego uszkodzenia tkanki nerwowej. Jedyną szansą dla chorych jest regularne przyjmowanie leków z rodziny interferonów, które zwalniają postęp choroby, łagodzą jej objawy i dają szansę na normalne życie – osobiste i zawodowe. Lek przyjmowany jest w postaci zastrzyku w określone miejsce ciała według przyjętego, rygorystycznego harmonogramu. Pacjent zobligowany jest do prowadzenia dziennika, w którym prowadzi ewidencję przyjmowanych dawek oraz notuje skutki uboczne towarzyszące terapii.

Celem pracy było stworzenie aplikacji mobilnej wspomagającej chorego w przyjmowaniu leku. Głównym założeniem projektu było całkowite zastąpienie niewygodnego w prowadzeniu, nieczytelnego oraz łatwego do zgubienia papierowego dzienniczka opisywanym programem. Aplikacja miała ponadto pełnić funkcję wspomagającą w planowaniu procesu leczenia, zarządzaniu zasobami leku oraz ułatwiać kontakt z lekarzem lub pielęgniarką prowadzącą pacjenta. Program jest dedykowany na urządzenia z obecnie najbardziej popularnym oraz jednym z najbardziej zaawansowanych i niezawodnych mobilnych systemów operacyjnych dostępnych na rynku czyli systemem Android.

W pierwszych dwóch rozdziałach nakreślono krótko problematykę stwardnienia rozsianego, opis choroby, jej przebiegu wraz z najpopularniejszymi objawami oraz dokonano przeglądu istniejących rozwiązań, które w mniejszym lub większym stopniu mogą wspomóc osoby chore. Kolejny rozdział został poświęcony przedstawieniu założeń funkcjonalnych jakie ma spełniać opisywana aplikacja oraz wymaganiom platformowym i ograniczeniom stawianym oprogramowaniu. Następne rozdziały opisują sposób implementacji, opis problemów jakie musiały zostać rozwiązane w trakcie realizacji pracy, a także diagramy oraz zastosowane techniki dobrego programowania. Na sam koniec przedstawiono testy funkcjonalne i jednostkowe jakim został poddany projekt, aby spełniać rygorystyczne normy niezawodności, jakie są stawiane przed aplikacją spełniającą funkcje medyczne oraz dokonano krótkiego podsumowania projektu.

# Problematyka stwardnienia rozsianego

**Stwardnienie rozsiane (inaczej SM) to choroba wpływając na mózg i rdzeń kręgowy, w wyniku jej działania osoba chora traci kontrolę nad mięśniami oraz ma problemy z czuciem oraz odbieraniem bodźców zewnętrznych.** Przy chorobie tej nerwy w mózgu i rdzeniu kręgowym zostają uszkodzone przez własny układ odpornościowy. Stan taki, jest nazywany chorobą autoimmunologiczną, czyli chorobą, **z powodu której system odpornościowy organizmu, który prawidłowo działając namierza i niszczy substancje obce dla organizmu, zaczyna atakować zdrowe tkanki.** W przypadku SM układ immunologiczny atakuje mózg oraz rdzeń kręgowy– składniki ośrodkowego układu nerwowego.

Centralny układ nerwowy złożony jest z nerwów, które działają na zasadzie przesyłania impulsów elektrycznych. Każdy z nich obtoczony jest substancją zwaną otoczką mielinową, która izoluje nerwy i pomaga w przenoszeniu impulsów elektrycznych pomiędzy mózgiem a resztą ciała. Dzięki tym informacjom, możliwe jest kontrolowanie ruchów mięśni kontrolujących między innymi poruszanie się i mówienie, a także odczuwanie bodźców zewnętrznych dochodzących do naszego organizmu.

**Stwardnienie rozsiane zaczerpnęło swoją nazwę od narastania chorych komórek w mózgu i rdzeniu kręgowym**. Powstają one, gdy ochronna i izolacyjna otoczka mielinowa, która osłania nerwy zostaje uszkodzona. Bez mieliny przesył sygnałów zostaje zakłócony lub całkowicie wstrzymany. Następuje nagłe załamanie komunikacji wewnątrz organizmu będące powodem objawów SM.

SM według Towarzystwa Stwardnienia Rozsianego jest, poza urazami po wypadkach, najczęstszą przyczyną niepełnosprawności neurologicznej osób dorosłych do 50 roku życia. **Stwardnienie rozsiane występuje dwa do trzech razy częściej u kobiet niż u mężczyzn**, a jego występowanie jest bardzo rzadko spotykane przed okresem dojrzewania. Zwiększone ryzyko rozwoju choroby występuje szczególnie u osób nastoletnich do wieku 50 lat i stopniowo obniża się w kolejnych latach. Lekarze nie są pewni co wywołuje stwardnienie rozsiane, ale istnieją badania sugerujące, że wpływ na możliwość zachorowania ma genetyka, środowisko otaczające daną osobę, a nawet wirus.

Objawy stwardnienia rozsianego różnią się w zależności od osoby i często zmieniają się w czasie u tego samego pacjenta. **Najczęstsze z nich obejmują:**

* osłabienie mięśni,
* pogorszoną koordynację,
* zamazane lub zamglone widzenie,
* ból oczu,
* podwójne widzenie.

Zaburzenia widzenia występują u około 60% pacjentów. Jako syndrom początkowy pojawiają się u co trzeciego chorego. Przejawiają się jako pozagałkowe zapalenie nerwu wzrokowego oraz zaburzenie czynności mięśni okoruchowych. Pierwsza przypadłość powoduje głównie pogorszenie ostrości widzenia, zaburzenie percepcji kolorów oraz ból gałki ocznej. Utrudnienia w sposobie widzenia trwają najczęściej do kilku dni. Z reguły uszkodzenia nerwu wzrokowego są odwracalne. Drugie schorzenie przejawia się oczopląsem, podwójnym widzeniem oraz zezem.

W miarę postępu choroby, objawy mogą ponadto obejmować sztywnienie mięśni oraz bóle, trudności z kontrolowaniem oddawania moczu lub problemy z umiejętnościami poznawczymi.

Dostępny jest dość szeroki wachlarz nowoczesnych leków, które coraz skuteczniej zmniejszają częstotliwość i nasilenie objawów. Niektóre z nich mogą również spowalniać postęp niektórych rodzajów choroby. Jednym z dostępnych leków jest interferon beta 1b, który jest lekiem uwzględnionym w opisywanej pracy z uwagi przede wszystkim, na fakt, iż osoba zainteresowana aplikacją używa właśnie tego leku.

Interferon beta-1b jest rodzajem białka z grupy interferonów. Jest on stosowany wśród pacjentów, u których po raz pierwszy wystąpiły objawy wskazujące na wysokie ryzyko wystąpienia SM. Kuracja polega na wstrzykiwaniu substancji w odpowiednie ściśle określone miejsce na ciele co 48 godzin. Kolejne obszary zastrzyków są ustalane w taki sposób, aby odstęp pomiędzy kolejnymi dawkami był jak największy, z uwagi na możliwe niepożądane objawy. Lek może powodować także bardzo nieprzyjemne skutki uboczne takie jak:

* objawy grypopodobne (dreszcze, gorączka, bóle mięśniowo-stawowe, bóle głowy, osłabienie, nudności) najsilniejsze po pierwszych dawkach leku i ustępują w trakcie dalszego leczenia.
* niedokrwistość,
* stany zapalne i inne objawy w miejscu wstrzyknięcia
* depresja
* bezsenność, wymioty,
* zaburzenia czynności tarczycy.

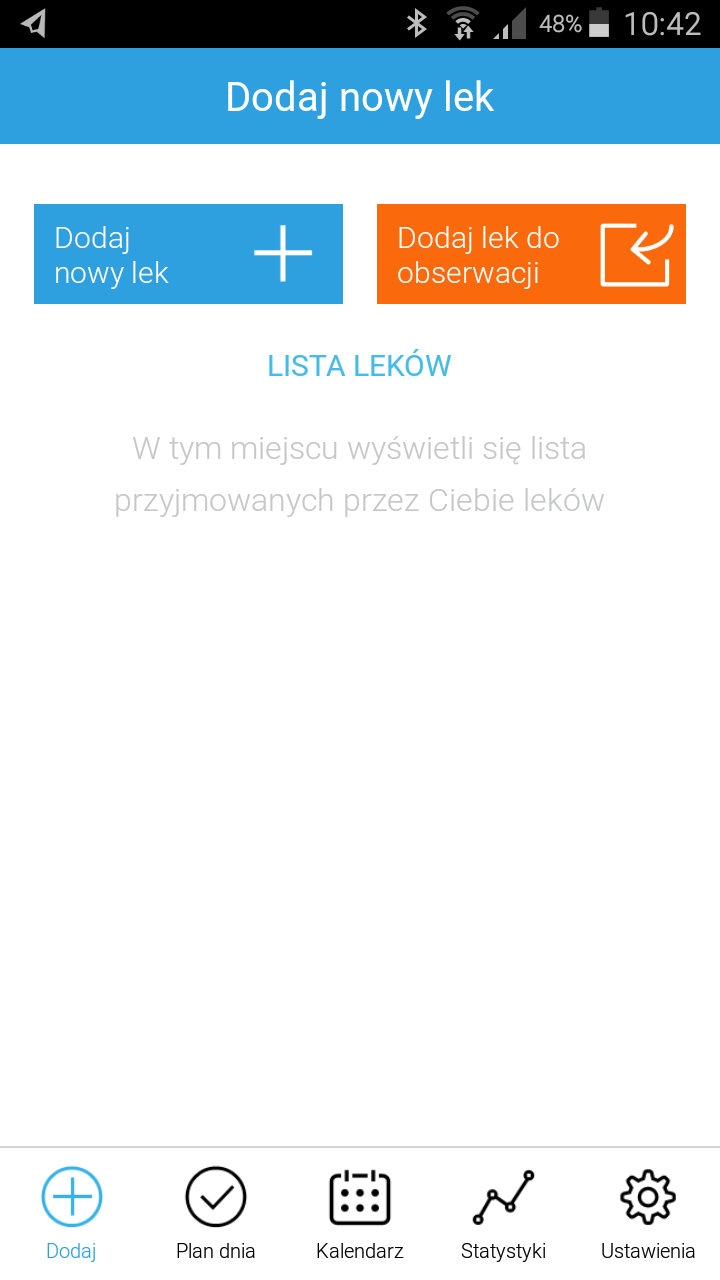
Dlatego bardzo ważnym czynnikiem ułatwiającym skuteczne i w miarę bez objawowe leczenie są regularne zastrzyki według harmonogramu, zapisywanie wszystkich objawów i częste konsultacje z lekarzem. We wszystkich powyższych czynnościach ma pomagać budowana aplikacja.

# Istniejące rozwiązania

W momencie pisania pracy istnieje dość duży wachlarz aplikacji, które mogą wspomóc osoby, chore na stwardnienie rozsiane, przyjmujące interferon. Są to jednak programy ogólnego przeznaczenia, pisane z myślą o jak największej grupie odbiorców i nie posiadają funkcji typowych dla tego konkretnego zastosowania. Mogą one dlatego jedynie wspomóc pacjenta, ale nie są w stanie całkowicie zastąpić papierowego dzienniczka. W poniższym rozdziale dokonano przeglądu istniejących rozwiązań oraz krótkiego opisu wybranych najciekawszych pozycji.

* Moje leczenie

Aplikacja pozwala w prosty sposób zaplanować harmonogram przyjmowania leków lub wizytę u lekarza. Dodatkowo za pomocą notyfikacji, przypomina o każdej dawce leku i wydarzeniu. Umożliwia także bliskim na sprawdzenie czy pacjent bierze regularnie swoje leki. Program ten osoby chore na stwardnienie rozsiane mogą wykorzystać do przypominania o kolejnym zastrzyku lub wizycie u lekarza, jednak nie posiada bardzo istotnych funkcji z punktu widzenia osoby cierpiącej na SM takich jak dodawanie niepożądanych objawów, generowanie oraz wysyłanie raportu do lekarza czy wskazywanie miejsca kolejnej dawki. Program nie posiada także w swojej bazie leków interferonu, co także jest dużym utrudnieniem.



Rysunek Główny ekran aplikacji Moje leczenie

* My Therapy

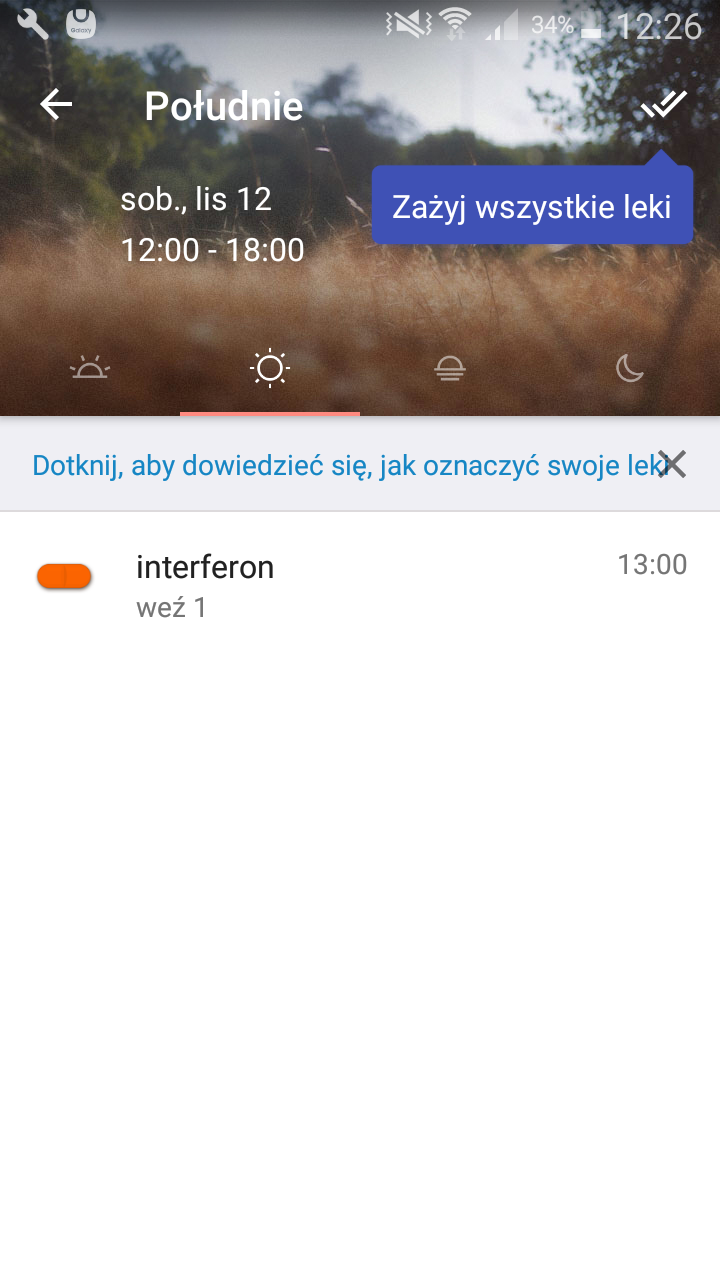
Aplikacja pozwala na dodanie harmonogramu zastrzyków, przypomina o zbliżającej się dawce leku oraz pozwala na wysyłanie bliskim powiadomienia, jeżeli pacjent pominie dawkę leku. Umożliwia też wprowadzenie dodatkowych parametrów takich jak ciśnienie krwi czy wyniki badań. Posiada także dodatkowo, bardzo ważną dla osób chorych na stwardnienie rozsiane, funkcję wygenerowania i wydrukowania miesięcznego raportów z leczenia. Wciąż jednak brakuje funkcji typowych dla opisywanego w pracy zastosowania czyli podglądu miejsca zastrzyku, zapisywania szczegółowych informacji związanych z daną dawką leku czy wysłanie raportu z dowolnej ilości zastrzyków drogą elektroniczną.



Rysunek : GŁÓWNY EKRAN APLIKACJI MY THERAPY

* Medisafe

Aplikacja podobnie jak prezentowane wcześniej posiada wiele funkcji ogólnego przeznaczenia, takie jak przypominanie o wzięciu kolejnej dawki leku, wprowadzenie dodatkowych danych o zdrowiu pacjenta, wysyłanie przypomnienia, także do innych wybranych osób, które pomagają pacjentowi w przyjmowaniu leków, generowanie raportów z leczenia i możliwość ich szybkiego wysyłania do lekarza, a także po darmowej rejestracji archiwizowanie historii zastrzyków po stronie serwera. Tak jak w poprzednich aplikacjach brakuje jednak wielu potrzebnych funkcji.



Rysunek : GŁÓWNY EKRAN APLIKACJI MEDISAFE

* Dzienniczek pacjenta

Papierowy dzienniczek, zaprojektowany przez producenta leku, jest najczęściej stosowanym i jedynym obecnie istniejącym kompleksowym rozwiązaniem dla pacjentów. Umożliwia on planowanie dni kolejnych dawek, pokazuje schemat kolejnych zastrzyków oraz posiada część przeznaczoną na zapisywanie ewentualnych problemów z przyjmowaniem leku oraz notowania skutków ubocznych. Pacjent musi jednak samemu pamiętać o zaaplikowaniu zastrzyku w danym dniu i o konkretnej ustalonej godzinie oraz o uzupełnieniu zapasu leku. Może jedynie wspomóc się jednym z wyżej opisanych rozwiązań lub zwykłym kalendarzem w telefonie. Papierowy dzienniczek, nie ułatwia także komunikacji z lekarzem oraz archiwizowania zastrzyków. Pacjent musi zabierać ze sobą dzienniczek na każdą konsultację a ewentualne zagubienie dzienniczka, w którym zapisana jest duża część procesu leczenia może być sporym problemem. Biorąc jednak pod uwagę sposób działania istniejących aplikacji wspomagających jest obecnie najlepszym sposobem na organizowanie leczenia.

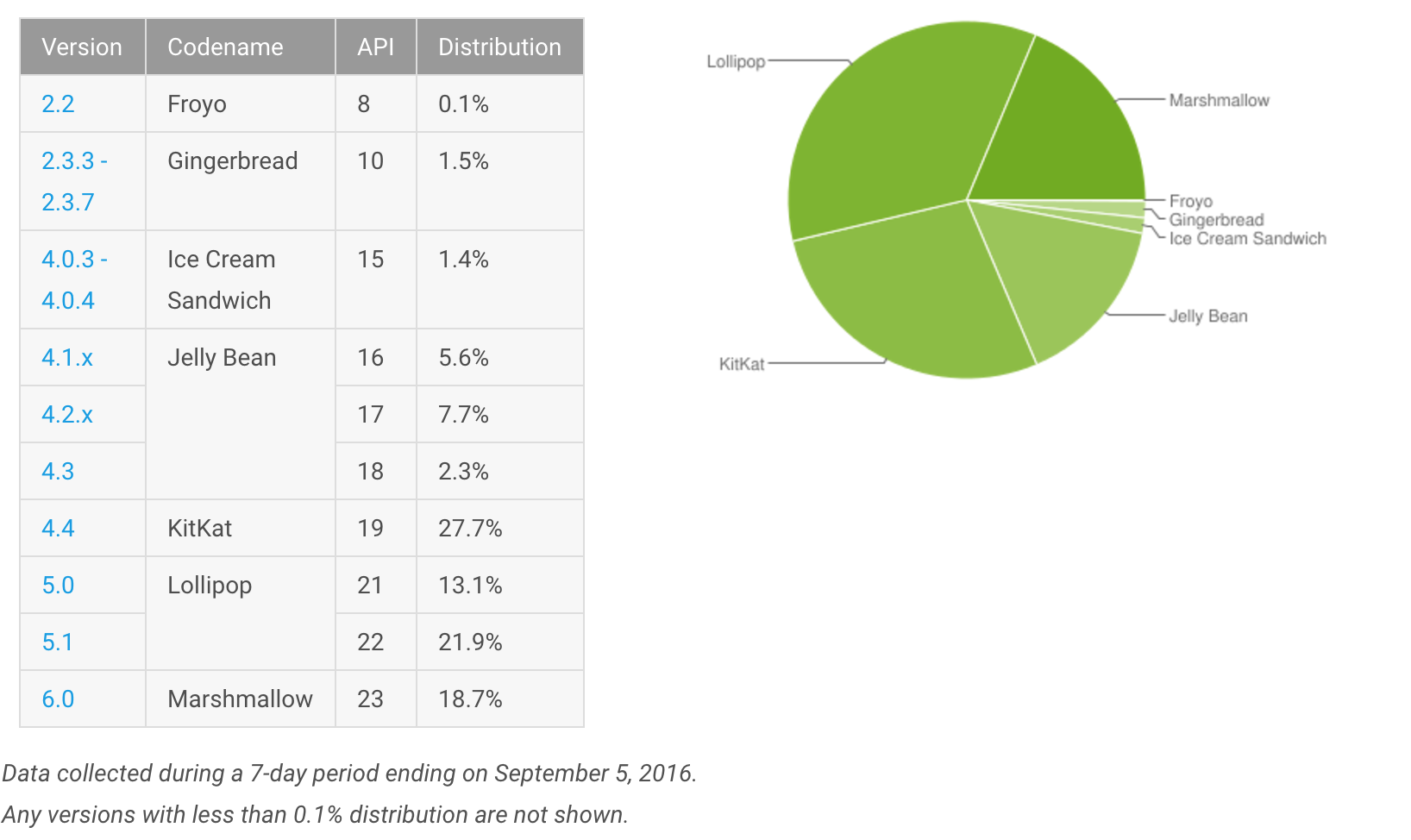
# Projekt oprogramowania

## Założenia funkcjonalne i platformowe

Celem pracy jest stworzenie aplikacji wspomagającej chorego w przyjmowaniu leku interferon beta 1b. Aplikacja ma pozwalać na zdefiniowanie harmonogramu powiadomień o zbliżającym się czasie aplikacji leku oraz wskazywać miejsce wykonania zastrzyku. Użytkownik ma możliwość wprowadzić dodatkowe informacje takie jak ewentualne niepożądane objawy czy problemy w aplikacji zastrzyku. Dodatkowo aplikacja ma udostępniać prosty sposób zarządzania posiadanymi zasobami leku oraz przypominać o kończącym się ich zapasie. Możliwe ma być przeglądanie historii terapii wraz ze szczegółowymi informacjami o objawach czy innych dolegliwościach im towarzyszących. Program ma pozwalać ponadto na wygenerowaniu łatwego w interpretacji raportu dla lekarza prowadzącego bądź pielęgniarki i wysłanie uprzednio wygenerowanego pliku w wiadomości sms lub email. Bardzo ważną funkcją, z uwagi na bardzo często występujące u osób cierpiących na stwardnienie rozsiane problemy ze wzrokiem, ma być także możliwość zmiany wyglądu aplikacji na tzw. wysoki kontrast, który jest bardziej czytelny. W celu łatwiejszego zarządzania projektem oraz pracami w poszczególnych jego etapach, przy konsultacji z osobą zainteresowaną aplikacją sporządzono historie użytkowników na podstawie powyższego opisu. Następnie każdy z punktów wymaganych do pełnego zakończenia prac nad projektem wypunktowano pod względem ważności oraz czasochłonności w odniesieniu do całości projektu. Pozwoliło to na lepszą organizację harmonogramu prac i ewentualną redukcję opóźnień lub zmianę wymagań. Wyżej opisywane historie użytkowników prezentują się następująco:

* Jako użytkownik :
  + Chcę mieć możliwość wprowadzenia i zapamiętania moich danych w celu łatwiejszego i szybszego zatwierdzania zastrzyków, generowania raportów oraz ich wysyłania (3pkt)
  + Chcę mieć możliwość wprowadzenia pożądanej godziny wykonywania zastrzyków oraz dnia pierwszej aplikacji w celu zarządzania procesem leczenia (1pkt)
  + Chcę mieć możliwość utrzymywania notyfikacji o zbliżającym się zastrzyku w celu przypomnienia mi o kolejnej dawce (5pkt)
  + Chcę mieć możliwość podglądu miejsca zbliżającego się zastrzyku aby zaaplikować lek w odpowiednie miejsce ciała(13pkt)
  + Chcę mieć możliwość odłożenia notyfikacji jeżeli w danym momencie nie mogę wykonać zastrzyku(3pkt)
  + Chcę mieć możliwość wygenerowania raportu z jednego lub wielu zastrzyków na raz w celu łatwego dokumentowania procesu leczenia (5pkt)
  + Chcę mieć możliwość wysłania wygenerowanego raportu do lekarza lub pielęgniarki na podany wcześniej adres email lub nr telefonu w celu łatwiejszej i szybszej komunikacji (1pkt)
  + Chcę mieć możliwość zmiany wprowadzonych wcześniej ustawień w celu łatwiejszej personalizacji aplikacji(3pkt)
  + Chcę mieć wgląd do pomocy/samouczka w razie problemów z obsługą aplikacji(3pkt)
* Jak lekarz lub pielęgniarka:
  + Chcę mieć możliwość otrzymania w łatwej do odczytania formie raportu z zastrzyków, w którym zawarte będą miejsca aplikacji leku oraz ewentualne niepożądane objawy

Dodatkowo założono, że aplikacja będzie współpracować z urządzeniami, na których zainstalowano system operacyjny Android w wersji przynajmniej 4.4 KitKat. Wybrana minimalna wersja pozwala na zastosowanie w pisanej aplikacji wszystkich niezbędnych funkcji, przy zachowaniu stosunkowo dużego wsparcia dla większości użytkowników. Biorąc pod uwagę dane z września 2016r. jak pokazuje poniższa grafika, aplikacja wspiera 81.4% wszystkich urządzeń z zainstalowanym systemem Android.



Rysunek 4: Fragmentacja systemu android wrzesień 2016r

Tworząc aplikację mobilną, należy także zawsze zwrócić szczególną uwagę na wydajność, z uwagi na ograniczone zasoby sprzętowe. Założono iż miejsce kolejnego zastrzyku będzie wyświetlane na modelu 3D co spowoduje dużą czytelność tego ekranu dla użytkownika. Jest to jednak, miejsce potencjalnego spowolnienia działania aplikacji, szczególnie na telefonach z gorszymi parametrami sprzętowymi co należy wziąć pod uwagę podczas implementacji oraz testowania.

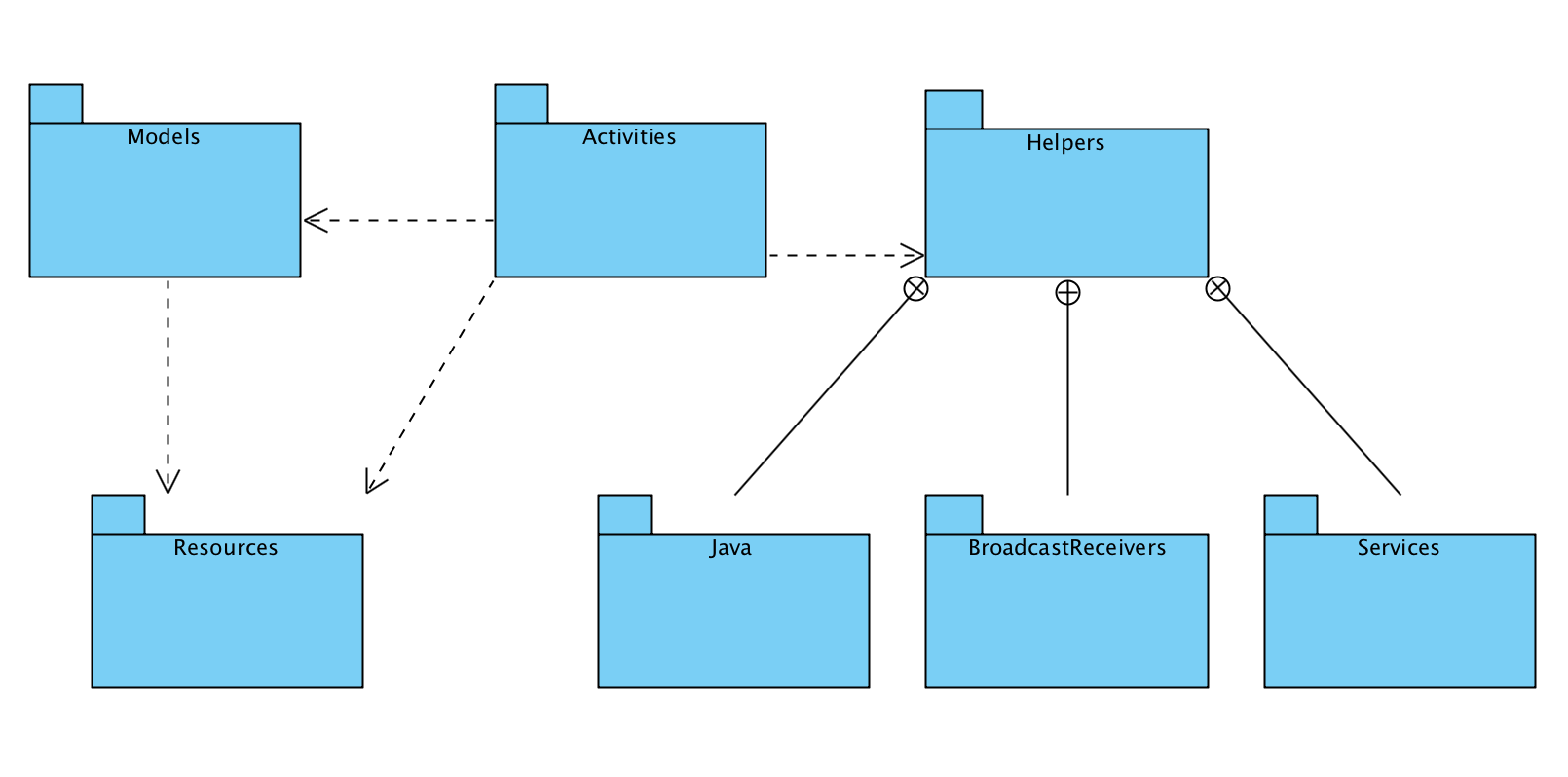
## Diagramy i wzorce projektowe

Dobre oprogramowanie wymaga przemyślanego projektu. Dlatego przed przystąpieniem do prac związanych z implementacją, wykonano diagramy UML kluczowych elementów systemu oraz zaprojektowano strukturę całego projektu.

Klasy używane w projekcie zostały podzielone na kilka grup:

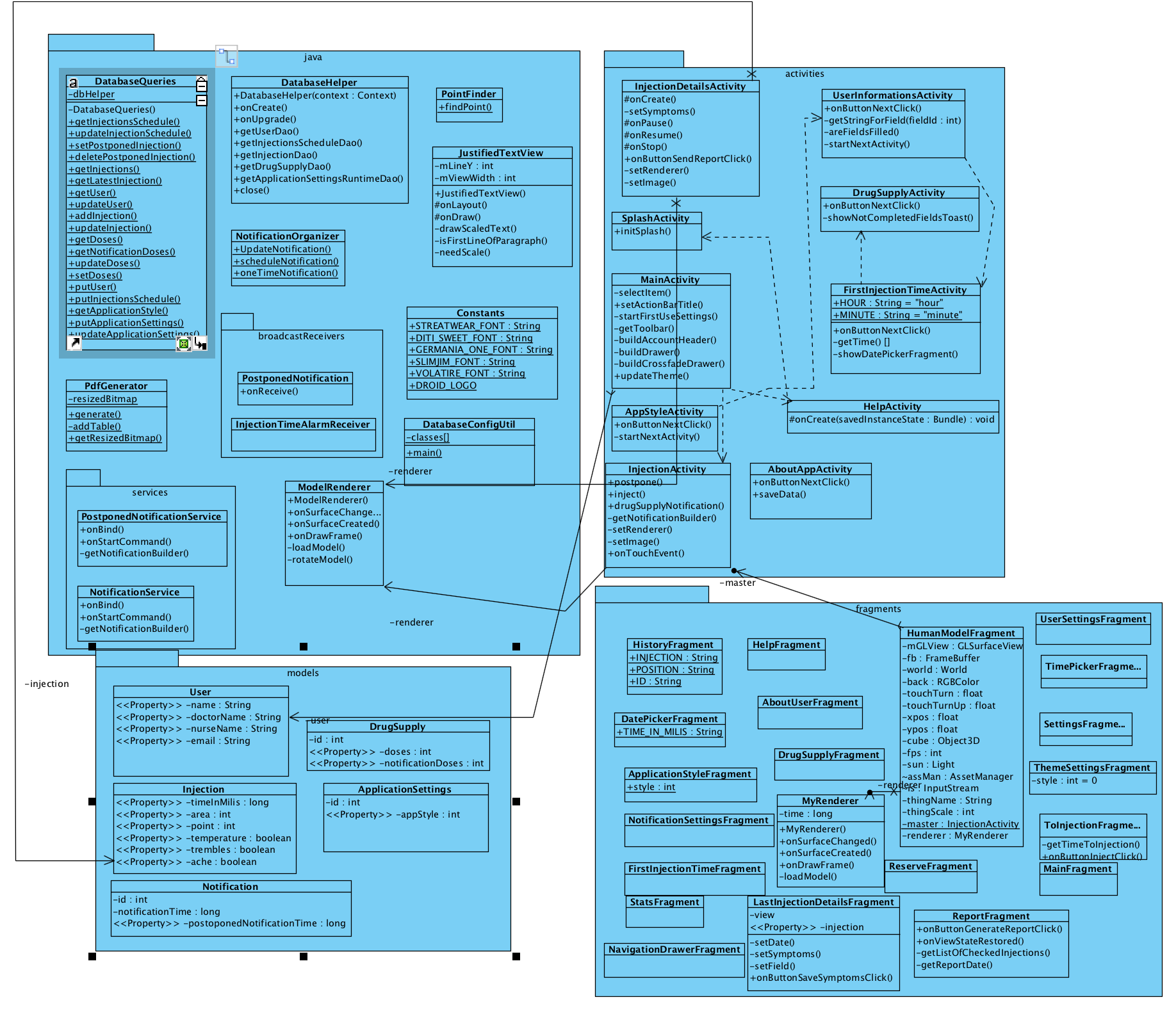
* Activities/Fragments– klasy aktywności i fragmentów, czyli klasy związane z jednym pojedynczym ekranem lub jego częścią, odpowiedzialne za komunikację pomiędzy klasami modelowymi a interfejsem użytkownika, zawierające całą z tym związaną logikę oraz definiujące sposób wyświetlania interfejsu użytkownika
* Models- Klasy modelowe zawierające klasy encyjne mapowane na tabele w bazie danych oraz logikę biznesową związaną z tymi encjami.
* Java/ BroadcastReceivers/ Services- Wszystkie klasy pomocnicze np. zarządzające notyfikacjami, odbierające notyfikacje oraz wykonujące operację na obiektach klas encyjnych takich jak uaktualnienie stanu leku czy informacji o użytkowniku. Są to klasy używane w wielu miejscach aplikacji, dzięki temu kod jest bardziej przejrzysty, a zmiana logiki związanej z działaniem jednej z nich nie ponosi za sobą wielu zmian w projekcie.
* Resources- pliki XML definiujące wygląd interfejsu użytkownika, pliki graficzne rastrowe i wektorowe oraz pliki XML zawierające wszystkie napisy używane w projekcie.

Przyjęty podział dobrze obrazuje zamieszczony poniżej diagram pakietów.



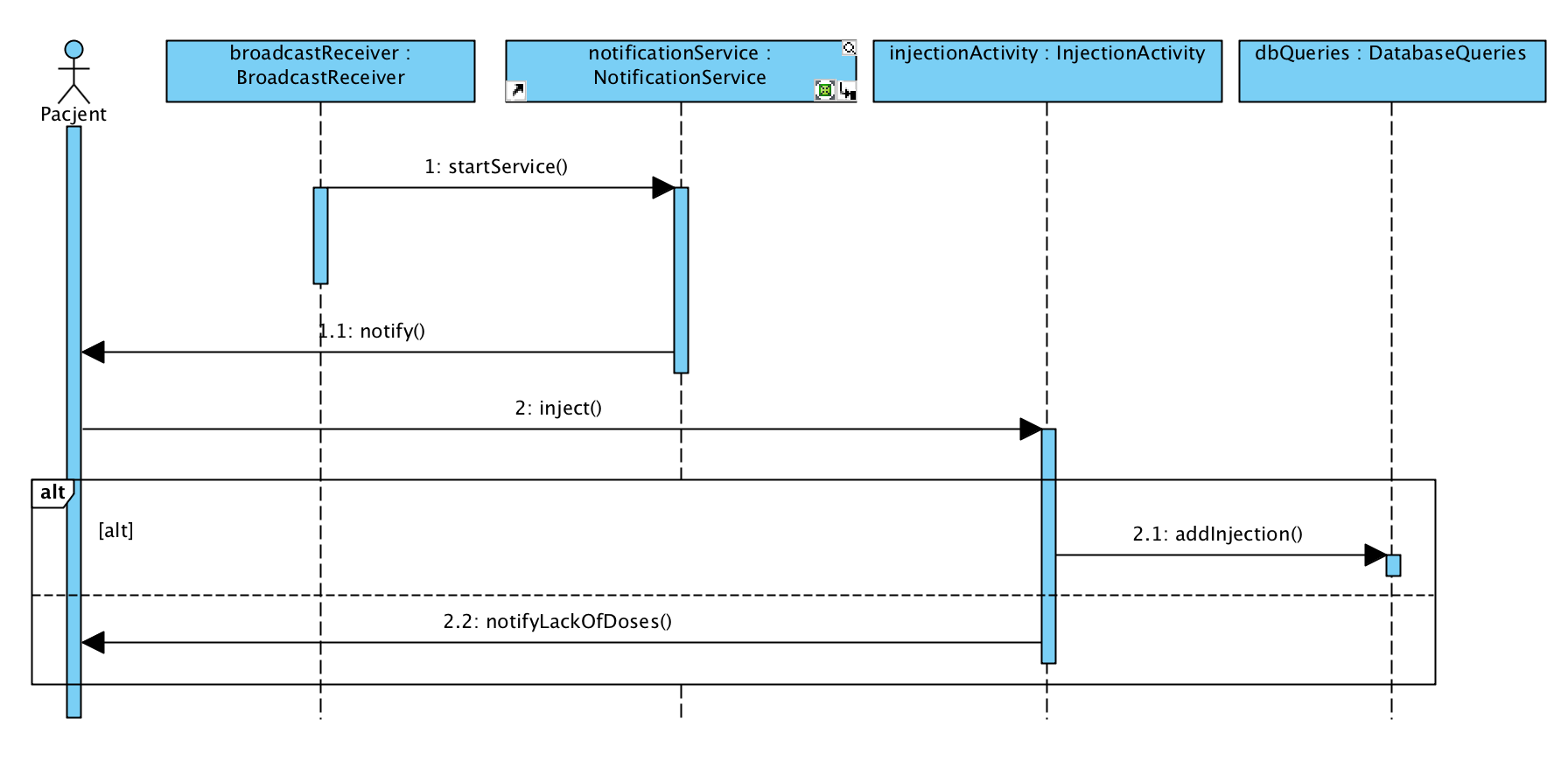
Rysunek : Diagram relacji pakietów w aplikacji

Dodatkowo wykonano diagram klas na którym pokazano najważniejsze klasy w projekcie oraz najbardziej istotne relacje między nimi.



Rysunek : Diagram pokazujący relację miedzy klasami w projekcie

Kluczowymi funkcjami jakie ma spełniać aplikacja, są wykonywanie zastrzyku oraz notyfikowanie o zbliżającym się czasie kolejnej dawki, dlatego wykonano dla nich dodatkowo diagram sekwencji. Dzięki temu łatwo zobaczyć przepływ sterowania pomiędzy obiektami, a także ogólny schemat algorytmu.



Rysunek : Diagram sekwencji dla notyfikacji oraz procesu wykonywania zastrzyku

## Użyte biblioteki

Podczas implementacji aplikacji, starano się używać jak najczęściej funkcji dostępnych w oficjalnym SDK systemu Android bez używania dużej ilości bibliotek zewnętrznych. Takie podejście gwarantuje łatwe utrzymanie i rozwój aplikacji w przyszłości poprzez uniezależnienie kodu od nieoficjalnych funkcji. Wybrano jednak kilka zewnętrznych bibliotek, które w znacznych stopniu ułatwiły pracę nad projektem oraz poprawiły czytelność kodu oraz ogólny wygląd aplikacji.

* JPCT-AE

PCT-AE jest najważniejszą biblioteką użytą w aplikacji, udostępnia ona łatwy i spójny interfejs ułatwiający pisanie aplikacji mobilnych wykorzystujących grafikę 3D. Jest lekka i bardzo wydajna z uwagi na to, że została oparta na OpenGL ES czyli oficjalnym API dla grafiki 3D w systemie Android. jPCT obsługuje Android 1.5 i wyższe i może być używany z OpenGL ES 1.x, jak również z OpenGL ES 2.0.

* MPAndroidChart

MPAndroidChart to biblioteka przeznaczona do wyświetlania wykresów. W opisywanym projekcie została użyta do stworzenia ekranu wyświetlającego wykres podsumowujący częstość występowania każdego rodzaju niepożądanych objawów. Wykresy tworzone za pomocą tej biblioteki są wysoce konfigurowalne, interaktywne i łatwe w tworzeniu.

* ORMLite

ORMLite to biblioteka znacznie ułatwiająca pracę z bazą danych. Jest to biblioteka mapująca obiekty występujące w obiektowym kodzie projektu na relacyjną bazę danych SQLLite, co umożliwia uniknięcie pisania zapytań SQL w obiektowym kodzie aplikacji . Dzięki temu praca z bazą danych staje się dużo łatwiejsza a kod jest łatwiejszy w czytaniu i utrzymaniu.

* Material Drawer

Material Drawer jest lekką i wysoce konfigurowalną biblioteką umożliwiającą uzyskanie efektów identycznych jak w specyfikacji material design na wersjach systemu niższych niż 5.0 Lollipop. Dzięki tej bibliotece udało się uzyskać przede wszystkim bogate w opcje i przejrzyste dwustopniowe wysuwane menu oraz kilka mniejszych efektów.

* JUnit

Biblioteka służąca do pisania powtarzalnych testów jednostkowych. W aplikacji posłużyła głównie jako biblioteka pomocnicza w tworzeniu automatycznych testów funkcjonalnych

* Espresso

Biblioteka służąca do pisania automatycznych niezawodnych testów interfejsu użytkownika. W aplikacji była główną biblioteką używaną do przetestowania najważniejszych funkcji programu.

* iText

iText jest biblioteką, która pozwala w łatwy i czytelny tworzyć, dostosowywać, kontrolować i zmieniać dokumenty w formacie Portable Document Format (PDF). Generowanie dokumentów i raportów może odbywać się na podstawie danych z pliku XML lub bazy danych. W aplikacji powyższa biblioteka posłużyła do zaimplementowania możliwości wygenerowania raportu z dowolnej ilości zastrzyków.

# Implementacja

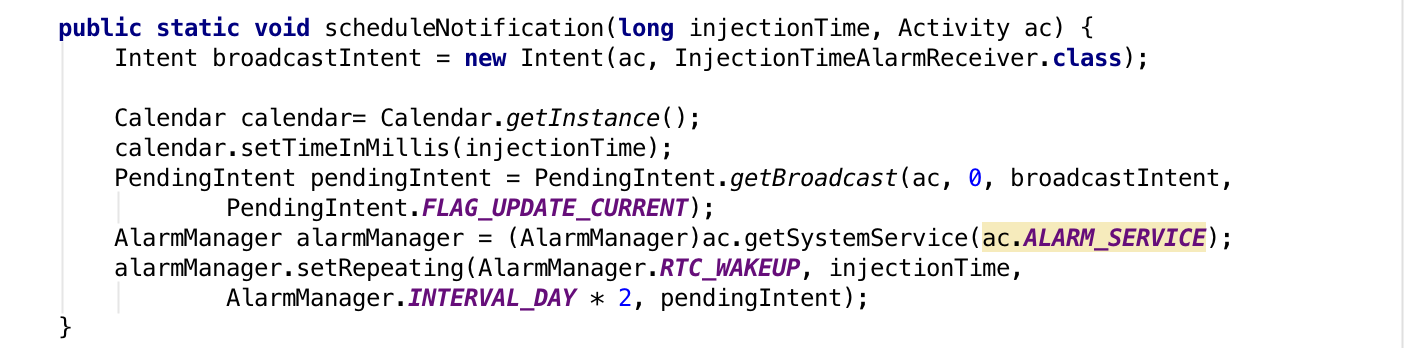
Proces implementacji, rozpoczęto od stworzenia ekranów odpowiedzialnych za zebranie niezbędnych wiadomości o użytkowniku takich jak: adres email do wysyłania raportów, data pierwszego zastrzyku oraz godzina notyfikacji czy obecna ilość dawek leku posiadanych przez pacjenta. Głównymi zadaniami tych klas są zebranie informacji od użytkownika, przetworzenie ich i zapisanie w bazie danych oraz przejście do następnego ekranu z pytaniami. Poniżej zaprezentowano przykładowy fragment kodu, który w tym przypadku, jest odpowiedzialny za ustawienie godziny notyfikacji.



Powyższym fragment przedstawia pojedynczą aktywność. **Activity** czyli aktywność, to jedna z podstawowych i najczęściej używanych klas w Androidzie. Klasy po niej dziedziczące odpowiedzialne są za interakcję z użytkownikiem, tworzenie okna naszej aplikacji i uruchamianie innych komponentów oraz przechodzenie do innych aktywności. Zazwyczaj jedna podklasa Activity reprezentuje jedno okno naszej aplikacji.

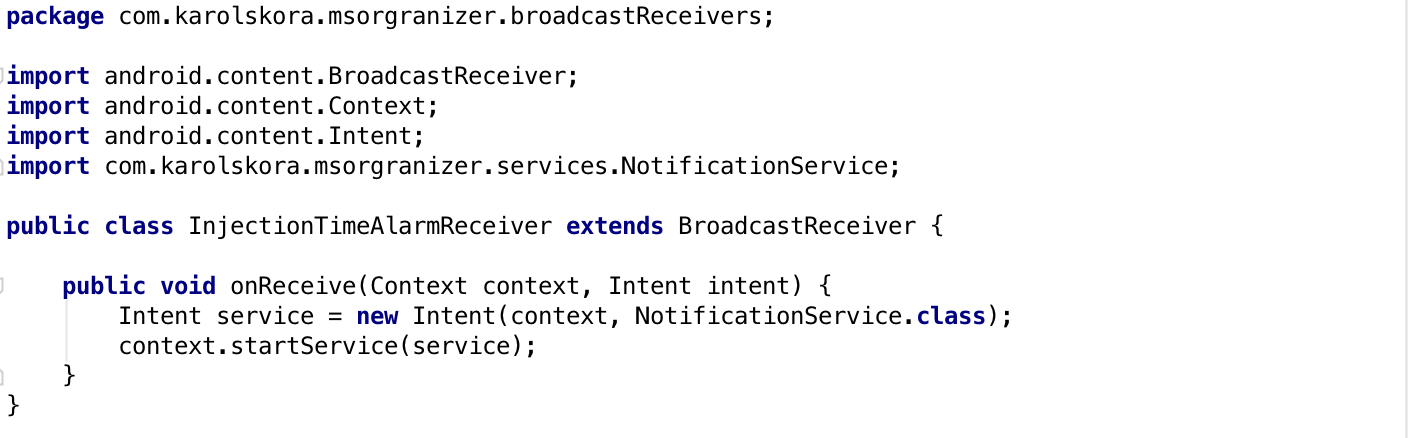
Po utworzeniu aktywności, w metodzie onCreate(), która jest podstawową metodą związaną z cyklem życia aktywności w Androidzie, ustawiany jest styl aplikacji zgodny z wcześniejszym wyborem a następnie wskazywany jest plik xml definiujący wygląd ekranu. Po wciśnięciu przez użytkownika przycisku, odczytywany jest podany czas notyfikacji, zapisywany w obiekcie klasy Bundle w celu przekazania go kolejnej aktywności, a następnie uruchamiany jest fragment odpowiedzialny za ustawienie daty pierwszego zastrzyku. Warto zwrócić uwagę, na fragment kodu, w którym sprawdzana jest wersja systemu użytkownika, a następnie w zależności od niej używana jest odpowiednia metoda, prawidłowa dla danej wersji SDK. Jest to bardzo częsty przypadek, jeżeli aplikacji musi wspierać starsze wersje systemu. Pozostałe ekrany pokazywane przy pierwszym uruchomieniu aplikacji działają analogicznie do powyższego przykładu więc nie będą dodatkowo omawiane.

Jak wspomniano w rozdziale dotyczącym procesu projektowania aplikacji kluczową funkcją jest notyfikowanie pacjenta o zbliżającym się czasie leku. Poniżej zaprezentowano i omówiono fragment kodu odpowiedzialny za omawianą funkcję.



Powyższa metoda uruchamiania jest w przypadku ustawiania nowej notyfikacji. Tworzony jest obiekt kalendarza, a czas ustawiany jest na podany wcześniej przez użytkownika czas notyfikacji. Następnie tworzony jest obiekt klasy AlarmManager w której konstruktorze przekazywany jest parametr wcześniej utworzonego obiektu klasy PendingIntent, wskazującego jakie działania mają zostać podjęte w przyszłości w czasie alarmu. Klasa AlarmManager zapewnia dostęp do usług systemu alarmowego. Pozwala to zaplanować uruchomienie aplikacji w pewnym momencie w przyszłości. Gdy włączy się alarm przekazany podczas tworzenia obiektu obiekt klasy PendingIntent automatycznie uruchamia docelowy komponent, jeżeli nie został już wcześniej uruchomiony. Zarejestrowane alarmy są zatrzymane, gdy urządzenie jest w trybie uśpienia ale opcjonalnie mogą wzbudzić urządzenie w czasie alarmu. AlarmManager posiada blokadę procesora dopóki metoda obiektu klasy AlarmReceiver OnReceive () jest wykonywana. Gwarantuje to, że telefon nie zostanie uśpiony przed zakończeniem wykonywania zadania.

Następnym elementem związanym z uruchomieniem notyfikacji jest klasa InjectionTimeAlarmReceiver, której wcześniej wspominana metoda orReceive() uruchamiana jest po rozpoczęciu alarmu.

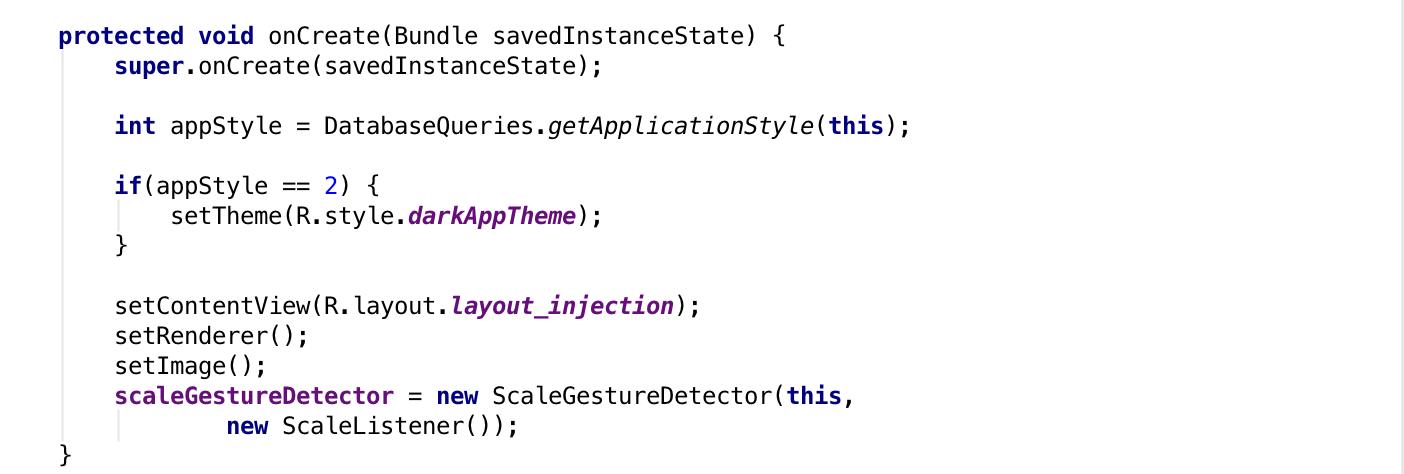


Klasa ta dziedziczy po BroadcastReceiver, która jest bazowa dla wszystkich klas uruchamianych przez metodę intencji sendBroadcast() lub przez metody klasy AlarmManager. W metodzie onReceive() tworzony jest obiekt intencji z przekazanym obiektem docelowym NotificationService oraz uruchamiana jest metoda startService() na obiekcie klasy Context, która włącza działanie serwisu. Kod serwisu został przedstawiony poniżej.

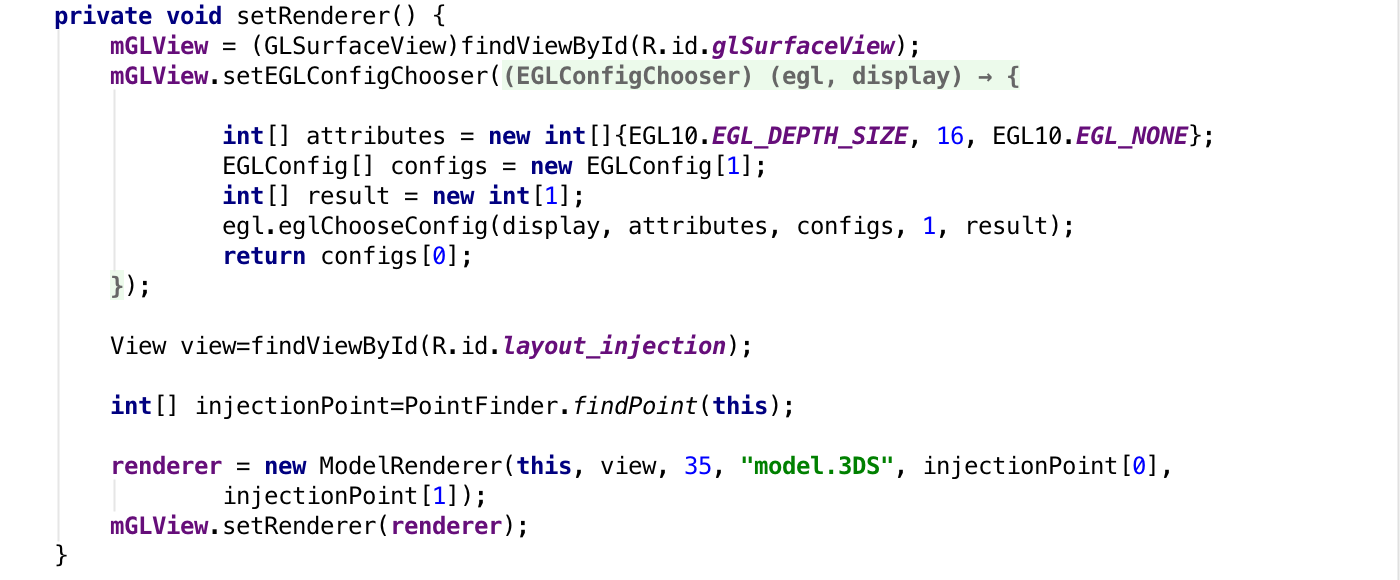


Klasa dziedziczy po klasie Service, czyli komponencie służącym do uruchamiania zadań w tle, nie wpływających na działanie systemu z punktu widzenia użytkownika. Aplikacja może uruchomić usługę i nadal działać w tle, nawet jeśli użytkownik przełączy się na inną aplikację. Service jest często używany do obsługi komunikacji sieciowej, zapisywania do pliku, czy tak jak w tym przypadku do uruchamiania notyfikacji. Powyższy service tworzy nową notyfikację z odpowiednią treścią i poprzez uruchomienie metody notify obiektu klasy NotificationManager wyświetla zbudowaną wcześniej notyfikację użytkownikowi.

Bardzo ważnym elementem programu jest czytelne wskazywanie miejsca kolejnego zastrzyku. Zdecydowano, że najlepszym wyborem będzie model 3D człowieka, który poprzez odpowiednie ustawienie i przybliżenie kamery będzie wskazywał miejsce danej dawki. Jak wspomniano w podrozdziale dotyczącym projektu oprogramowania, zdecydowano się na skorzystanie z biblioteki JPCT AE, która udostępnia łatwiejszy interfejs do pracy z niskopoziomową biblioteką graficzną OpenGL ES. Model człowieka osadzono na standardowym komponencie GLSurfaceView służącym do wyświetlania treści wygenerowanej przy pomocy OpenGL ES. Poniżej przedstawiono fragmenty kodu związane z wyświetlaniem modelu oraz omówiono najważniejsze fragmenty.

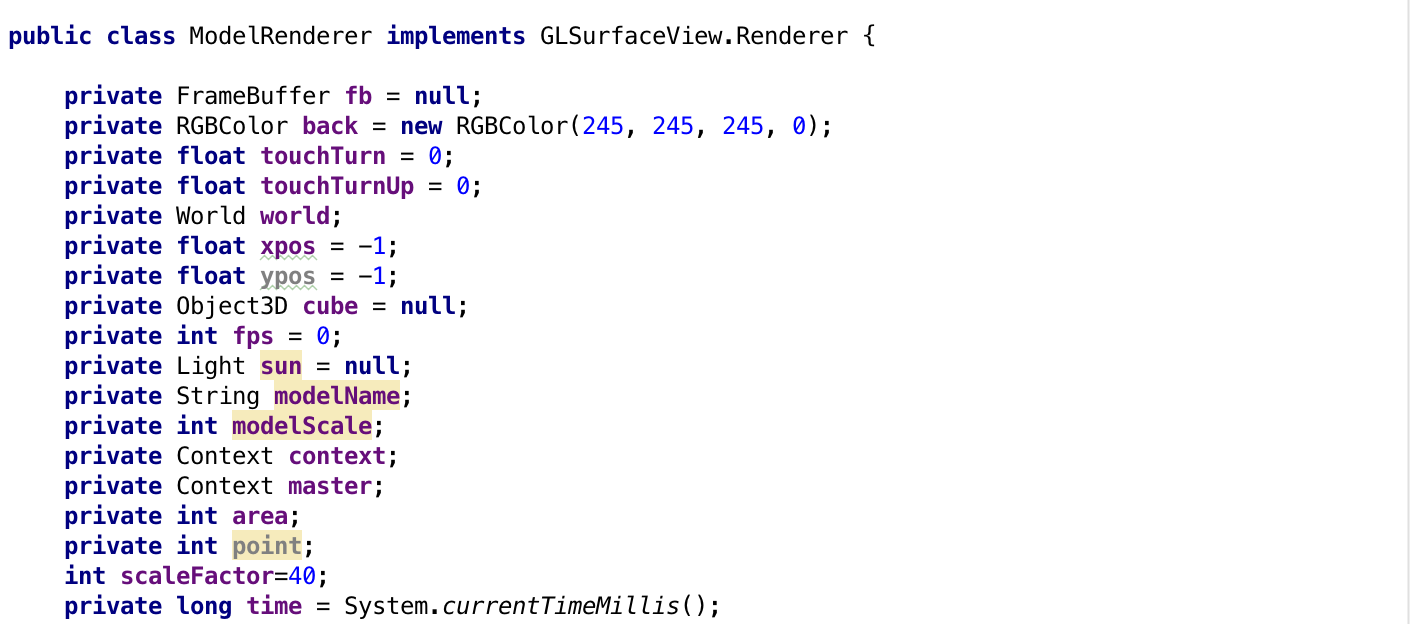


W metodzie onCreate() aktywności odpowiedzialnej za wykonanie zastrzyku wywoływana jest metoda setRenderer(), do której przeniesiono algorytm odpowiedzialny za dodanie modelu do ekranu.



Obiekt komponentu GLSurfaceView jest znajdywany poprzez metodę findViewById() i przypisywany do zmiennej mGLView. Następnie w metodzie setEGLConfigChooser następuje dodatkowa konfiguracja komponentu i ustawiana jest między innymi pożądana głębia kolorów. Następnie tworzony jest obiekt renderer typu ModelRenderer, w którego kosntruktorze przekazywana jest docelowa skala modelu, nazwa pliku z modelem do wczytania oraz miejsce wskazania zastrzyku.

Ostatnim etapem konfiguracji jest przekazanie w metodzie setRenderer() obiektu mGLView utworzonego wcześniej obiektu renderer. Klasa ModelRenderer została przedstawiona i umówiona poniżej.

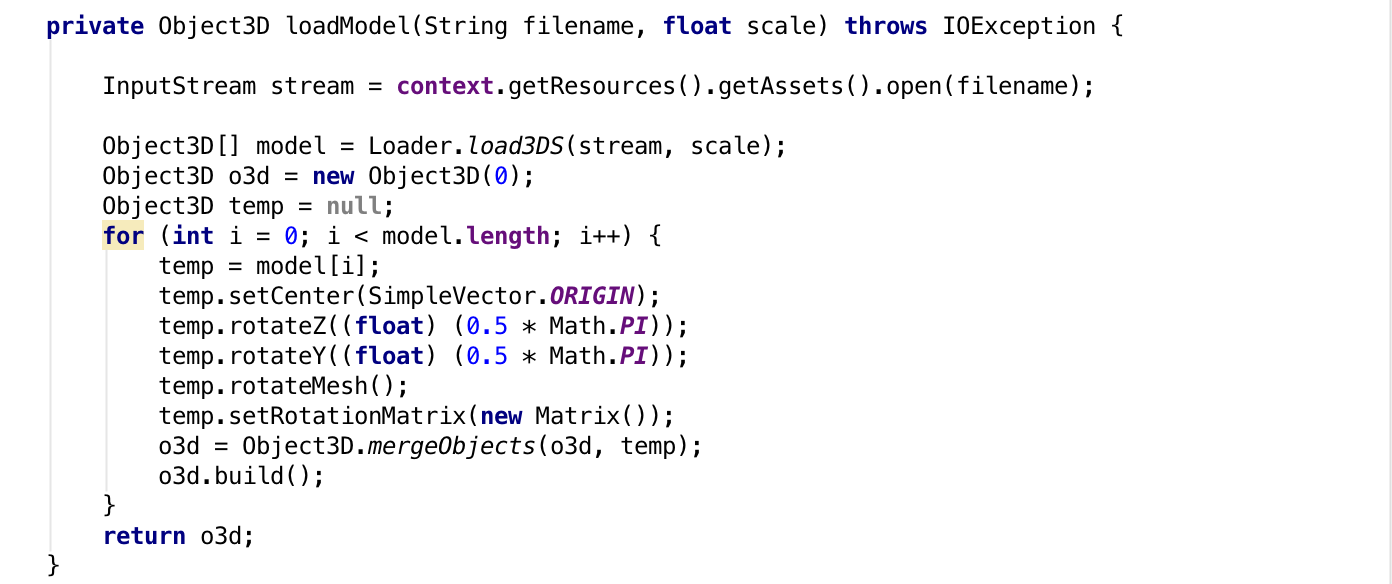


Klasa implementuje interfejs Renderer który jest ogólnym interfejsem implementowanym przez klasy odpowiedzialne za generowanie grafiki. Interfejs renderer deklaruje 3 abstrakcyjne metody, które zostały zaimplementowane w omawianej klasie:

* onDrawFrame()
* onSurfaceChanged()
* onSurfaceCreated()



W konstruktorze ustawiane są przekazane parametry oraz wczytany zostaje wybrany przez użytkownika styl aplikacji. Następnie metoda loadModel() inicjalizuje zmienną cube modelem wczytanym z pliku. Ostatnią operacją wykonywaną w konstruktorze jest dodanie możliwości poruszania modelem poprzez przesuwanie palca po ekranie. W tym celu na przekazanym w konstruktorze parametrze klasy GLSurfaceView wywoływana jest metoda setOnTouchListener() i poprzez użycie klasy anonimowej zdefiniowany zostaje sposób reagowania na gesty użytkownika.



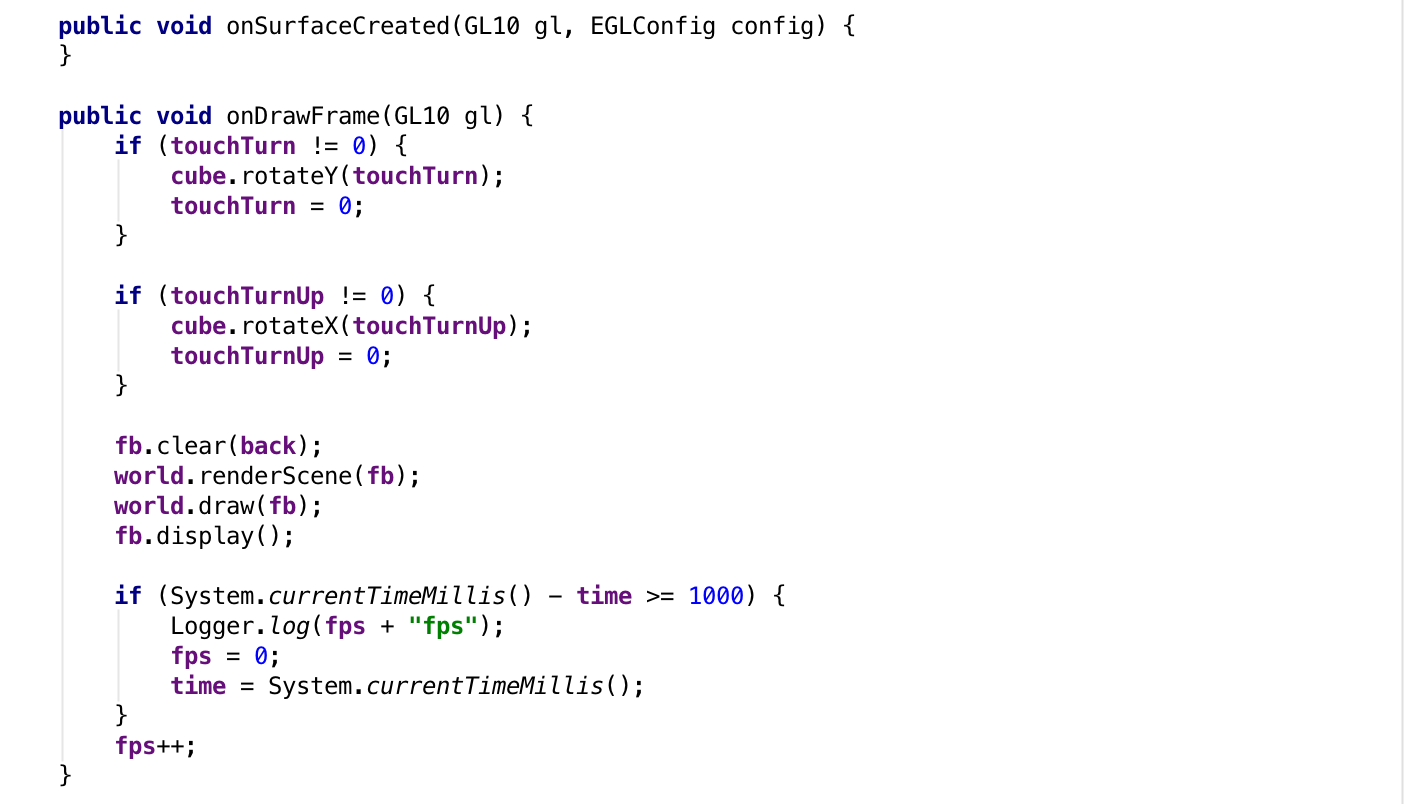
Model jest przechowywany jako plik o rozszerzeniu 3ds, który jest formatem plików używanym przez oprogramowanie Autodesk 3ds Max, przeznaczonym do modelowania i animacji 3D. Wewnątrz funkcji loadModel() metoda load3DS() wczytuje model człowieka jako tablice modeli 3D. Następnie dla każdego z elementów zostają wykonane operacje ustawiania środka oraz rotacji względem osi Z oraz Y, po to aby model został ustawiony w odpowiedniej pozycji, czyli przodem do patrzącego. Ostatecznie po wykonaniu opisanych operacji gotowy obiekt zwracany jest jako obiekt klasy Object3D.



metoda onSurfaceViewChanged() jest odpowiedzialna za ustawianie światła, odpowiedniego koloru tła oraz wczytanie i nałożenie tekstur. Tworzony jest obiekt klasy World, która jest najważniejszą klasą w bibliotece i zawiera obiekty i źródła światła, które definiują całą scenę. Następnie na wcześniej utworzonym obiekcie dodawane jest odpowiednie światło oraz tworzony jest obiekt słońca ze zdefiniowaną jasnością i typem światła. Ostatnimi etapami są nałożenie odpowiednich wcześniej przygotowanych tekstur oraz ustawienie modelu w zależności od miejsca zastrzyku. Fragment kodu metody rotateModel() odpowiedzialnej za odpowiednie ustawienie modelu został przedstawiony poniżej.



W powyższej metodzie sprawdzany jest obszar wykonywanego zastrzyku i w zależności od niego, ustawiana jest odpowiednia pozycja kamery oraz rotacja modelu względem osi Y. Umożliwia to przybliżenie kamery na pożądany obszar ciała modelu.



Ostatnimi omawianymi metodami zaimplementowanymi w klasie ModelRenderer są funkcje onSurfaceCreated() oraz onDrawFrame(). Metoda onSurfaceCreated, jak pokazano w powyższym fragmencie kodu, pozostaje pusta. Dzieje się tak dlatego, że cały proces generowania grafiki został przeniesiony do wyżej pisanej metody onSurfaceChanged(). Dzięki temu, przy każdej zmianie, cała scena generowana jest na nowo. W metodzie onDrawFrame() natomiast ustawiany jest bufor ramki służący do przechowywania informacji o pojedynczej ramce obrazu, oraz dodatkowo liczona jest ilość ramek na sekundę, co przyda się podczas testów wydajności generowania grafiki na słabszych urządzeniach.

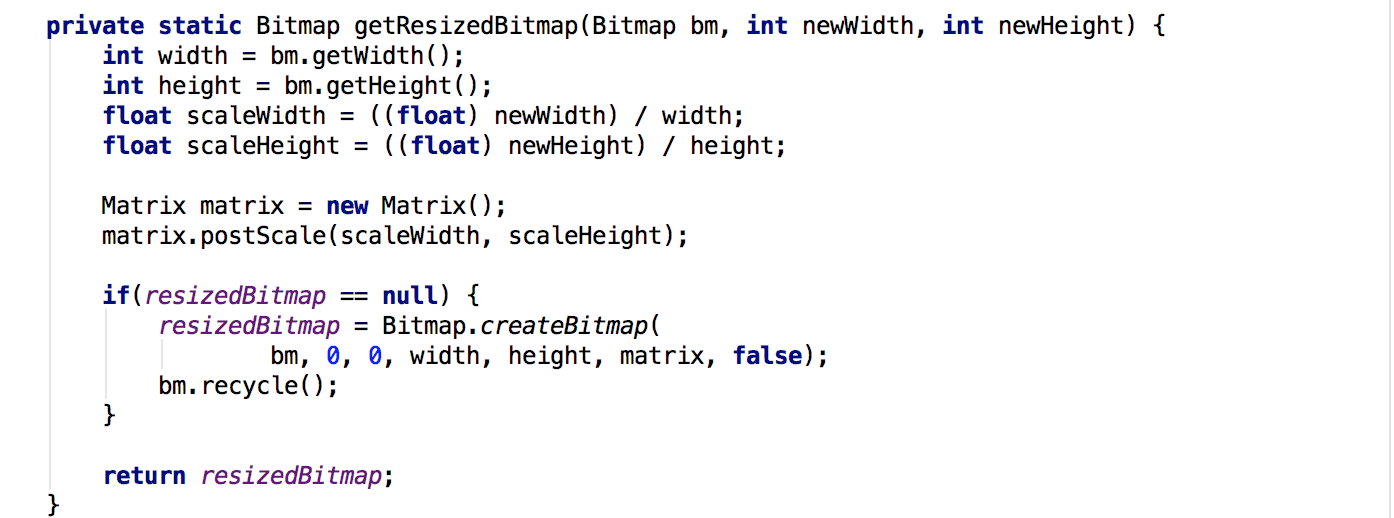
Ostatnim omawianym fragmentem programu jest generator raportów. Jak wspomniano w rozdziale dotyczącym wykorzystanych bibliotek, do generowania raportów wykorzystano bibliotekę iText. W aplikacjach tworzonych na urządzenia z zainstalowanym systemem Android 4.4 lub nowszym, istnieje wbudowana klasa PrintedPdfDocument umożliwiająca wygenerowanie pliku pdf. Zdecydowano się jednak, na użycie dodatkowej biblioteki, ze względu na dużo większą czytelność kodu, oraz dodatkowe możliwości jakie oferuje wspomniana biblioteka.



Klasą odpowiedzialną za tworzenie dokumentów PDF jest PdfGenerator. Jest to klasa posiadająca metodę statyczną generate(), która jest publiczną metodą, odpowiedzialną za wygenerowanie dokumentu oraz kilka prywatnych metod pomocniczych, uczestniczących w generowaniu dokumentu. Wewnątrz metody generate() otwierany jest nowy plik z nazwą podaną jako parametr metody. Następnie tworzony jest obiekt klasy FileOutputStream, odpowiedzialnej za pisanie do pliku. Po stworzeniu strumienia i wygenerowaniu nowego dokumentu, z bazy danych pobierane są niezbędne dane o użytkowniku, a następnie tworzony jest nagłówek wraz z tytułem zawierającym Imię oraz nazwisko użytkownika, a także informację o dacie wygenerowania raportu. Na końcu dla każdego zastrzyku z listy przekazanej do raportu, w metodzie addTable(), tworzona jest osobna tabela ze szczegółowymi danymi konkretnego zastrzyku.



Metoda addTable() służy do wygenerowania tabeli dla konkretnego zastrzyku. Na początku tworzony jest obiekt klasy PdfTable reprezentujący całą tabelę. Następnie tworzona są zmienna kalendarza przechowująca datę zastrzyku oraz zmienna tekstowa, w której umieszczono napis z datą zastrzyku. W następnym kroku tworzony jest obiekt klasy PdfCell reprezentujący pojedynczą komórkę w tabeli i dodawana jest do niego treść uprzednio utworzonej zmiennej przechowującej datę zastrzyku. Później tworzone są komórki zawierające nagłówki dla objawów i miejsca zastrzyku oraz generowana jest bitmapa o odpowiednim rozmiarze wskazująca miejsce zastrzyku. Służy do tego pomocnicza metoda getResizedBitmap(), która została przedstawiona poniżej. Ostatnim elementem jest dodanie listy z wypunktowanymi niepożądanymi objawami. Tak wygenerowany jest obiekt jest wpisywany do dokumentu jako pojedyncza tabela.



Metoda getResizedBitmap(), jak wspomniano powyżej, jest funkcją pomocniczą, zwracającą obiekt bitmapy o zmienionych rozmiarach. Wewnątrz niej, pobierane są aktualne wymiary bitmapy oraz ustalana jest skala. Następnie metoda postScale() obiektu klasy Matrix zmienia wymiary według wcześniej ustalonej skali, a na końcu, przetworzony obiekt jest zwracany jako obiekt klasy Bitmap.

# Testy

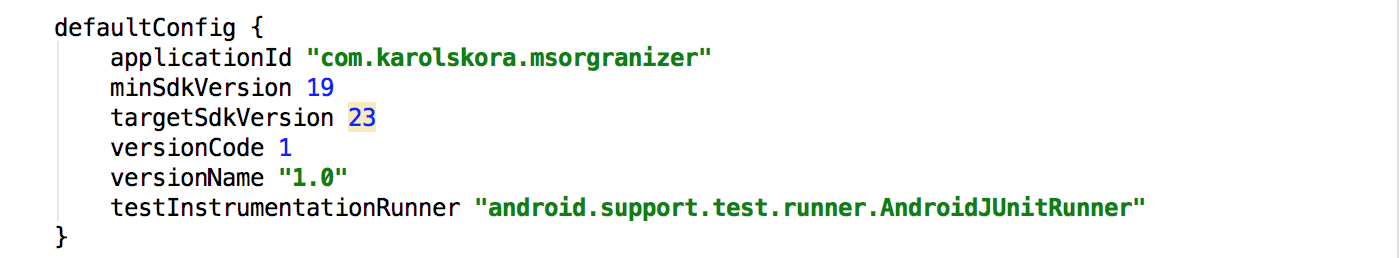
Jak wspomniano we wstępie aplikacja, która ma służyć do celów medycznych musi działać niezawodnie, dlatego wykonano szereg testów funkcjonalnych, sprawdzających czy wszystkie najważniejsze funkcje programu działają poprawnie. Testy były uruchamiane po każdej zmianie w kodzie aplikacji, dzięki temu można było mieć pewność, że nowy kod nie wprowadził żadnego niepożądanego zachowania do aplikacji. Poniżej przedstawiono przykładowe testy a następnie umówiono najważniejsze fragmenty.



Powyższy test jest odpowiedzialny za sprawdzenie przebiegu pierwszego uruchomienia aplikacji. Klasa testowa jest oznaczona adnotacją @LargeTest, która informuje, że test powinien być uruchomiony jako część większych testów. W każdej klasie testowej dodano dodatkowo adnotację @RunWith() z podanym aliasem wersji klasy uruchamiającej testy(AndroidJUnit4). Dzięki temu zabezpieczono się od sytuacji w przyszłości, kiedy zmieni się domyślna wersja tej klasy. W powyższym teście zaimplementowano dodatkowo pomocniczą metodę clearAppInfo() oznaczoną adnotacją @Before. Metoda ta zostaje uruchomiona przed każdym uruchomieniem testów zdefiniowanych w klasie. Wewnątrz niej zostaje wyczyszczona baza danych aplikacji, dzięki czemu program zawsze rozpoczyna działanie od ekranów wyświetlanych tylko przy pierwszym uruchomieniu. Wewnątrz pojedynczego testu, oznaczonego adnotacją @test, znajdują się wszystkie operacje wykonywane w trakcie testowania. W tym przypadku, test służy do sprawdzenia przebiegu wszystkich ustawień początkowych. Pierwszą instrukcją wykonaną w teście jest metoda statyczna sleep() klasy SystemClock. W tym przypadku wstrzymuje ona wykonywanie wszystkich instrukcji do czasu zakończenia wyświetlania animacji przy uruchomieniu programu. Następnie metody onView(), wczytują kolejno wszystkie komponenty, uczestniczące w danym przypadku użycia, bazując na ich identyfikatorach przypisanych w plikach definiujących wygląd interfejsu użytkownika. Po wczytaniu pożądanego komponentu, metoda perform() wykonuje na nim zdefiniowane akcje. W powyższych przykładzie są to metody:

* replaceText() – metoda podmieniająca tekst w komponencie
* scrollTo()- metoda przesuwająca ekran na dany komponent tak, aby był w całości widoczny. Bez wykonania tej akcji nie można przeprowadzić żadnej innej interakcji z komponentem, który domyślnie nie jest widoczny na ekranie.
* click() – Metoda wykonująca kliknięcie na wskazanym komponencie
* closeSoftKeyboard()- Metoda zamykająca klawiaturę ekranową

Aby uruchomić testy, w pliku konfiguracyjnym build.gradle należy dodatkowo zdefiniować parametr testInstrumentationRunner wskazujący domyślną klasę uruchamiającą.



Jednym z najważniejszych testów jest test sprawdzający proces wykonywania zastrzyku oraz dodania niepożądanych objawów. Jego kod przedstawiono poniżej.

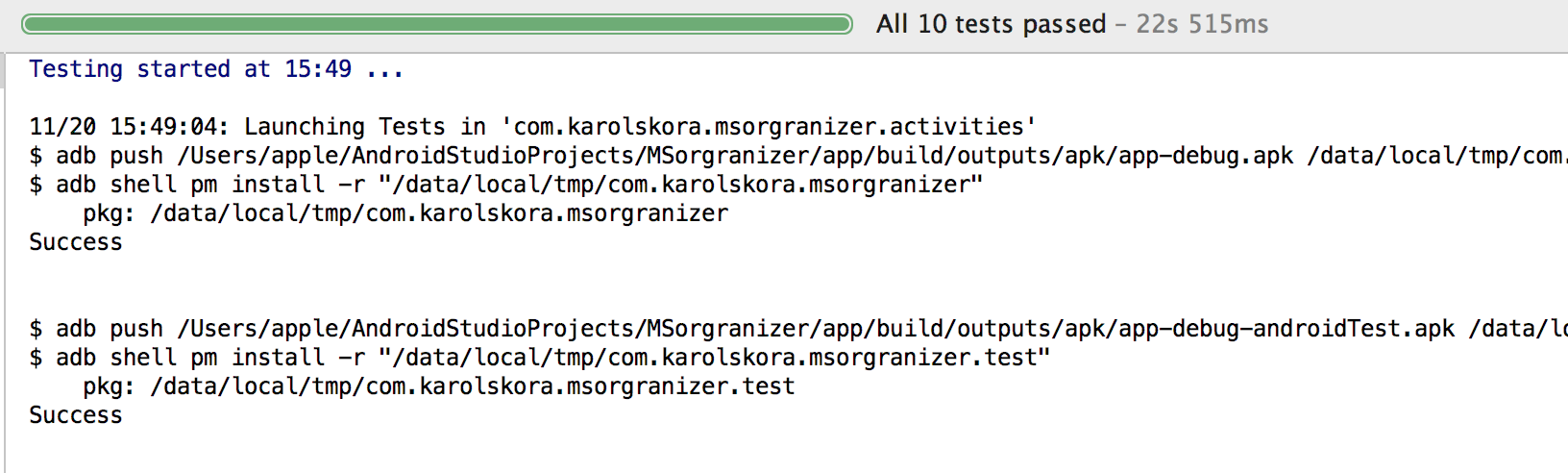


W powyższej klasie testowej zaimplementowano jedną metodę testową o nazwie injectionTest(), oznaczoną adnotacją @Test. Wewnątrz metody skorzystano z funkcji opisanych powyżej, użytych także podczas testowania pierwszego uruchomienia programu. Na początek wczytywany jest przycisk odpowiedzialny za wykonanie zastrzyku oraz wykonywana jest na nim akcja click(). Następnie na ekranie zastrzyku wykonywana jest akcja wstrzyknięcia. Ostatnim etapem jest dodanie testowych objawów jakie wystąpiły po podaniu danej dawki leku.

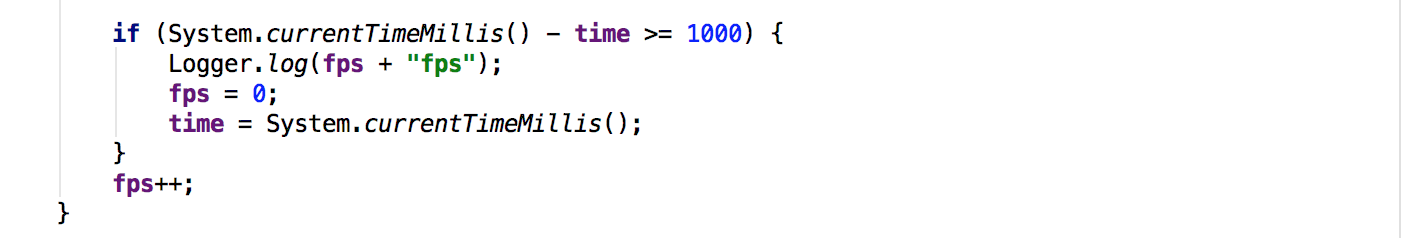
Analogicznie wykonano testy wszystkich najważniejszych funkcji aplikacji. Ostatecznie powstało 10 testów sprawdzających poprawność:

* ustawień początkowych
* wykonania zastrzyku i dodania objawów ubocznych
* przejścia po wszystkich elementach menu
* sprawdzenia historii zastrzyków oraz szczegółów jednego z nich
* zarządzania zapasami leku: zmiana ilości leku, aktualizacja notyfikacji o kończących się zasobach
* wygenerowania raportu dla lekarza i wyświetlenia powstałego pliku
* zmiany ustawień:
  + informacji o użytkowniku
  + czasu powiadomień o zastrzyku
  + wyglądu aplikacji
* skorzystania z ekranów pomocy

Po poprawnym wykonaniu wszystkich testów środowisko Android studio pokazuje na ekranie informację o sukcesie wraz z czasem trwania testów.



Dodatkowo sprawdzono także czy słabsze urządzenia poradzą sobie ze wskazywaniem zastrzyku w formie modelu 3D. W tym celu w klasie ModelRenderer dodano fragment kodu umożliwiający monitorowanie ilości klatek na sekundę w czasie rzeczywistym podczas działania programu na danym urządzeniu.





Rysunek : Przykładowe komunikaty generowane podczas wyświetlania modelu

Następnie uruchomiono program na kilku urządzeniach testowych w celu sprawdzenia ich wydajności podczas generowania modelu. Zebrane dane przedstawiono poniżej w formie tabeli.

# Podsumowanie

# Literatura

* 1. Android Developer. [Online]. Dostępny w Internecie: <http://www.developer.android.com>
  2. Dawn Griffiths, David Griffiths, *Head First Android Development A Brain-Friendly Guide,* wydawnictwo O'Reilly Media, 2016
  3. Andrzej Stasiewicz, *Android Studio. Podstawy tworzenia aplikacji*, wydawnictwo Helion, 2015
  4. Anders Göransson, *Android. Aplikacje wielowątkowe. Techniki przetwarzania*, wydawnictwo Helion, 2015
  5. Erik Hellman, *Platforma Android. Nowe wyzwania*, wydawnictwo Helion 2014
  6. Poradnik użycia biblioteki JPCT [Online]. Dostępny w Internecie: <http://www.jpct.net/wiki/>
  7. Oficjalna dokumentacja biblioteki JPCT [Online]. Dostępny w Internecie: <http://www.jpct.net/jpct-ae/doc/>
  8. JUnit - Testy jednostkowe w Javie [Online]. Dostępny w Internecie: <http://javastart.pl/static/efektywne-programowanie/junit/>
  9. Antonio Pachon Ruiz, *Mastering Android Application Development*, O'Reilly Media, 2015