Wydział Podstawowych Problemów Techniki
Politechnika Wrocławska

Kontrola dostępu do smartfona przy wykorzystaniu urządzenia smartband

Karolina Bąk
Nr indeksu: 244917

Praca inżynierska napisana
pod kierunkiem
Dr inż. Przemysława Błaśkiewicza

Politechnika
Wrocławska

Wrocław 2021

Spis treści

Wstęp

Praca swoim zakresem obejmuje projekt systemu stałej autoryzacji wykorzystujący prostą analizę behawioralną w czasie rzeczywistym na podstawie danych gromadzonych przez inteligentną opaskę dla smartfonów działających pod Androidem. Klucze bezpieczeństwa są rzadkim zjawiskiem w mobilnych systemach. Niska popularność tego rozwiązania może wynikać z ceny tych urządzeń, często niekompatybilnych ze smartfonem metodach połączenia oraz zastępowalności innymi metodami wieloskładnikowej autoryzacji.

Celem pracy jest zaprojektowanie i stworzenie aplikacji o następujących założeniach funkcjonalnych:

- Aplikacja wykorzystuje inteligentną opaskę (pot. smartband) jako klucz bezpieczeństwa;
- Aplikacja pozwala blokować dostęp do innych aplikacji;
- Aplikacja komunikuje się z inteligentną opaską przy wykorzystaniu protokołu Bluetooth Low Energy;
- Aplikacja analizuje zachowanie użytkownika w czasie rzeczywistym;
- Aplikacja działa w tle;
- Aplikacja pozwala wybrać, które aplikacje obejmie ochroną;
- Wszystkie dane użytkownika są przechowywane lokalnie;
- Aplikacja jest odporna na popularne ataki.

Praca składa się z sześciu rozdziałów. W rozdziale 2 przedstawiono dogłębnie problem autoryzacji przy użyciu sprzętowych kluczy w smartfonach oraz braku prywatności wrażliwych danych gromadzonych przez smartbandy. Omówiono szczegółowo dane pobierane z sensorów w opasce oraz smartfonie. Scharakteryzowano zdarzenia, przy których ograniczany jest dostęp do urządzenia. Opisano także rozwiązania podjęte w celu uniemożliwienia sabotażu aplikacji. Przeprowadzono również analizę porównawczą istniejących rozwiązań z realizowanym systemem.

W rozdziale 3 przedstawiono projekt aplikacji w notacji UML. Wykorzystano diagramy przypadków użycia, komponentów oraz stanów. Przedstawiono dokładne scenariusze do przypadków użycia. Omówiono też projekt bazy danych.

W rozdziale 4 określono technologie użyte w implementacji aplikacji: wybrany język programowania oraz wykorzystywane biblioteki. Opisano w pseudokodzie i omówiono algorytmy blokujące dostęp do aplikacji. Wyczerpująco opisano protokoły komunikacji z opaską. Określono także w jaki sposób są analizowane dane o aktywności.

W rozdziale 5 przedstawiono wymagania aplikacji wobec środowiska. Określono także sposób instalacji oraz konfiguracji aplikacji. Rozdział zawiera również przykłady działania dla użytkownika.

Końcowy rozdział jest podsumowaniem uzyskanych wyników.

Analiza zagadnienia

W niniejszym rozdziale omówiono mankamenty związane z uwierzytelnianiem na urządzeniach mobilnych oraz kwestie prywatności wrażliwych danych pochodzących z urządzeń typu smartband. Przedstawiono zarys systemu. Określono, jakie dane będą rejestrowane przez aplikację oraz cel ich gromadzenia. Określono sposoby analizy danych pod kątem wykrywania sytuacji, w których smartfon jest pozostawiony bez nadzoru. Opisano mechanizmy podjęte w celu zabezpieczenia systemu oraz przechowywanych danych. Porównano istniejące rozwiązania z proponowanym w pracy, wskazując na innowacje oraz różnice.

2.1 Przedstawienie problemu

Kwestia bezpieczeństwa smartfonów jest w dzisiejszych czasach niezwykle ważną sprawą. Powszechnie stosowane metody ograniczenia dostępu, takie jak uwierzytelnienie przy użyciu hasła bądź odcisku palca, są niewystaraczające. Szczególnie jest to widoczne przy atakach fizycznych, gdzie na przykład można:

- wykorzystać nieprzytomność użytkownika, by użyć jego odcisku palca;
- poznać hasło w postaci symbolu na podstawie śladów palców na ekranie dotykowym;
- uzyskać dostęp, gdy użytkownik pozostawi odblokowane urządzenie bez opieki.

Zważywszy na fakt, iż telefony komórkowe stają się coraz bardziej powszechne [13] oraz zastępują komputery jako urządzenia wykorzystywane do łączenia się z siecią (około połowa ruchu sieciowego pochodzi z urządzeń mobilnych [4]), przechowują one wiele wrażliwych danych o swoich użytkownikach. Dlatego koniecznością jest wprowadzenie dodatkowego systemu zabezpieczeń, w szczególności wykorzystanie uwierzytelnienia wielopoziomowego, w celu zabezpieczenia urządzenia przed dostępem osób niepowołanych. Często można spotkać się z wykorzystaniem smartfonów jako autentykatorów (kody SMS oraz dedykowane aplikacje) do innych systemów informatycznych. Natomiast do autoryzacji dostępu do smartfona nie wykorzystywane są żadne dodatkowe autentykatory. Głównymi przeszkodami do ich implementacji są:

- niepraktyczność;
- monofunkcyjność.

Nieporęczność fizycznych kluczy bezpieczeństwa objawia się szczególnie w ich formie - są to niewielkie urządzenia przypominające pamięć USB lub kartę płatniczą. Dzięki temu łatwo je zgubić lub o nich zapomnieć, co uniemożliwia użytkownikowi dostęp do systemu. Często też w smartfonie brakuje niezbędnej do odczytania klucza infrastruktury, na przykład czytnika inteligentnych kart czy modułu NFC [6]. Do niepraktyczności tego rozwiązania przyczynia się także wyżej wynieniona monofunkcyjność kluczy. Oferują one jedynie autoryzację użytkownika przy użyciu kluczy kryptograficznych bądź jednorazowych kodów i służą do logowania na stronach internetowych oraz przy autoryzacji w niektórych aplikacjach, co znacznie ogranicza pole ich zastosowania. Dlatego potrzebne jest świeże spojrzenie na technologię kluczy fizycznych, które pozwoli stworzyć przystępne systemy stałej autoryzacji bazujące na wielu czynnikach środowiskowych oraz wykorzystujące powszechnie używane urządzenia by skutecznie zabezpieczyć dane szerokiej bazy użytkowników smartfonów.

Platformy mobilne jako dynamicznie rozwijające się technologie wspierają wachlarz urządzeń peryferyjnych, które są odpowiednie by zostać wykorzystane jako klucz bezpieczeństwa. Jednym z nich jest **inteligentna opaska**, potocznie zwana “smartband” bądź “fitness tracker”. Jest to urządzenie o kształcie zegarka

na rękę monitorujące aktywność użytkownika taką, jak: ilość wykonanych kroków, puls czy sen. Dane gromadzone przez opaski są przesyłane protokołem Bluetooth do aplikacji towarzyszącej udostępnionej przez producenta, skąd zostają przesłane na zewnętrzne serwery. Nie jest to bezpieczne rozwiązanie, zważywszy na: wrażliwość powyższych informacji, powiązanie ich z danymi osobowymi użytkownika oraz fakt, że mogą zostać udostępnione osobom trzecim [1]. Z tego powodu konieczny jest rozwój systemów przechowywania informacji o aktywności pochodzących z inteligentnych opasek, które zapewnią użytkownikowi prywatność i nie będą bezpośrednio powiązane z producentem danego urządzenia.

2.2 Opis aplikacji

Zniwelowanie słabych punktów autentykacji przy użyciu kluczy sprzętowych jest niezwykle ważne przy implementacji tego rozwiązania w urządzeniach mobilnych. Dlatego w pracy skupiono się na usprawnieniu poniższych niedoskonałości tej technologii:

- prawdopodobieństwo utraty urządzenia autoryzującego;
- niska powszechność kluczy sprzętowych;
- ograniczona możliwość stałej autoryzacji.

Proponowana aplikacja opiera się na wykorzystaniu smartbanda jako inteligentnego klucza sprzętowego. Autoryzacja użytkownika odbywa się poprzez analizę danych o aktywności pobieranych z opaski w krótkich odstępach czasu. Po wykryciu sytuacji, gdzie smartfon jest prawdopodobnie poza nadzorem użytkownika, następuje uruchomienie blokady wybranych aplikacji do momentu wprowadzenia poprawnego hasła w aplikacji. Uniemożliwienie dostępu dokonuje się poprzez monitorowanie, która aplikacja znajduje się na pierwszym planie systemu Android. W przypadku wykrycia niedozwolonego programu użytkownik zostaje przeniesiony do aktywności odpowiedzialnej za autoryzację.

Wybór smartbanda do pełnienia funkcji klucza został podyktowany dużą popularnością urządzeń tego typu. Na rynku dostępnych jest wiele niskobudżetowych modeli, które gromadzą dane wystarczające do dość dokładnego określenia aktywności użytkownika. Z tego powodu inteligentne opaski są idealne, by oprzeć o nie system stałej autoryzacji. Dużą zaletą smartbanda jest jego niepozorna forma, czyli zegarek na rękę. Użytkownicy noszą go przez dużą część dnia, a nawet w nocy, przez co znacznie zmniejsza się ryzyko jego utraty bądź kradzieży. Kolejnym atutem tego urządzenia jest fakt, iż pełni ono znacznie więcej funkcji niż klucz sprzętowy. Dzięki temu smartband jest znacznie bardziej praktyczny dla użytkownika.

2.2.1 Charakterystyka gromadzonych danych

Aplikacja opiera się w głównej mierze o informacje rejestrowane przez inteligentną opaskę. Zaliczają się do nich:

- liczba wykonanych kroków danego dnia;
- aktualna wartość pulsu;
- moment zaśnięcia;
- moment zdjęcia opaski.

Powyższe informacje pozwalają określić stan fizyczny użytkownika, co jest kluczowe dla działania aplikacji. Dodatkowo dane o aktywności są uzupełniane o wartość sensora liczącego kroki w telefonie. Dzięki temu możliwa jest detekcja sytuacji, w których osoba eksploatująca może nie być w stanie nadzorować swojego telefonu, na przykład podczas snu bądź po pozostawieniu go na biurku w pracy. Przechowywane są także podstawowe informacje o opasce takie, jak: adres MAC oraz stan baterii. Umożliwia to ponowne połączenie z opaską oraz monitorowanie stanu urządzenia w aplikacji.

Oprócz informacji o aktywności aplikacja przechowuje także listę zainstalowanych aplikacji. Pozwala to użytkownikowi dostosować jej działanie do własnej preferencji. Najważniejszą przechowywaną informacją jest hasz hasła użytkownika, które jest wymagane do odblokowania dostępu do wybranych wcześniej aplikacji.

2.2.2 Analiza aktywności

Ważną częścią pracy jest wykrywanie sytuacji, w których smartfon jest poza nadzorem. Aby było to możliwe aplikacja bada aktywność użytkownika, korzystając z określonych w powyższej podsekcji danych, pod kątem czterech zdarzeń:

- opaska traci połączenie ze smartfonem;
- użytkownik zasypia;
- występują znaczne rozbieżności pomiędzy zarejestrowanymi krokami;
- użytkownik zdejmuje opaskę.

Utrata połączenia wykrywana jest na podstawie metod nasłuchujących zmiany w statusie połączenia Bluetooth. Sen wykrywany jest poprzez otrzymanie powiadomienia z opaski o zarejestrowaniu odpowiedniego zdarzenia. Rozbieżności w rejestrowanych krokach monitorowane są przez porównanie tempa wzrostu kroków mierzonych przez smartbanda oraz telefon, a zdjęcie opaski rozpoznaje się poprzez brak wykrywanego pulsu, bądź poprzez otrzymanie powiadomienia ze smartbanda. W przypadku wykrycia jednej z powyższych sytuacji następuje automatyczne uruchomienie blokady aplikacji.

2.2.3 Zabezpieczenie aplikacji

Aby proponowany system zapewniał ochronę przed dostępem przez osoby niepowołane musi działać nieprzerwanie i być odporny na wyłączenie go przez atakującego. W tym celu usługi aplikacji są zaimplementowane jako *Foreground Service*, by działać stale w tle w zgodzie z limitami obowiązującymi od Androida Oreo[10]. Wykorzystano technologię *Wake Lock*[12] w celu umożliwienia aplikacji pozostania w stanie pełnej sprawności w przypadku, gdy telefon przechodzi w *Doze Mode*[11]. Wdrożono także *BroadcastReceiver*, który jest odpowiedzialny za monitorowanie restartów urządzenia. Po wykryciu ukończonego uruchomienia smartfona, usługa blokująca oraz gromadząca dane są restartowane według stanu sprzed wyłączenia urządzenia.

System jest także odporny na najpopularniejsze podatności w aplikacjach mobilnych związanych z danymi medycznymi[3]. Dzięki lokalnemu przechowywaniu informacji zapewniona jest odporność na ataki za pośrednictwem sieci. Aplikację zabezpieczono przed *Intent spoofing*, dzięki wykorzystaniu jedynie dokładnie sprecyzowanych obiektów Intent oraz zabezpieczeniu komponentów przed otrzymywaniem obiektów Intent z innych aplikacji. By zapewnić bezpieczeństwo gromadzonych danych baza danych oraz plik przechowujący hasło zostały zaszyfrowane przy użyciu algorytmu szyfrowania AES. Natomiast mniej ważne informacje są przechowywane w *SharedPreferences*, do których dostęp ma tylko projektowany system.

2.3 Analiza porównawcza istniejących systemów

Na rynku znajduje się wąskie grono rozwiązań o podobnych funkcjonalnościach. Poniżej zaprezentowano najciekawsze z nich. Określono ich zalety oraz wady, a także porównano je z systemem zaprezentowanym w pracy.

2.3.1 Yubikey

Yubikey [18] to seria nowoczesnych kluczy sprzętowych produkowanych przez Yubico, wykorzystywanych jako część wieloskładnikowej autoryzacji bądź autentykacji bazowanej na jednorazowych hasłach w szerokim gronie serwisów internetowych oraz systemów operacyjnych. Wspierają wiele protokołów kryptograficznych i

autentykacyjnych, w tym: WebAuthn, FIDO2, U2F, smart cardy kompatybilne z PIV oraz Yubico OTP. Modele dedykowane urządzeniom mobilnym do komunikacji ze smartfonem wykorzystują moduł NFC, USB-C oraz złącze Lightning. Autoryzacja odbywa się poprzez umieszczenie klucza w złączu USB-C bądź przystawienie go do tyłu telefonu dla urządzeń z włączonym NFC.

Yubikey posiada wiele zalet. Jest wspierany przez dużą liczbę serwisów i systemów, dzięki czemu wachlarz aplikacji autentykacyjnych oraz kodów SMS czy wiadomości e-mail można zastąpić jednym urządzeniem. Pomaga uniknąć wykradnięcia haseł poprzez phishing czy przechwycenie SMSa. Jest prosty w użyciu dla użytkownika i nie wymaga ładowania. Jest również odporny na wodę oraz zgniecenie.

Dużą wadą Yubikey jest jego cena. Modele zapewniające autoryzację na smartfonach kosztują na tą chwilę minimum 45€ bez podatku VAT[17], czyli około 200 zł. Dla zwykłego użytkownika może być to zbyt duża kwota, gdy może skorzystać z darmowych wariantów dwuskładnikowej autoryzacji. Forma klucza (małe urządenie przypominające pendrive) sprzyja jego łatwemu zgubieniu, co uniemożliwia dostęp do serwisów, które z niego korzystały. By temu zaradzić producent zaleca posiadać zapasowy klucz, co wiąże się z dodatkowym wydatkiem rzędu 200 zł. Nie należy także zapominać o tym, że nie wszystkie telefony wspierają NFC oraz USB-C. Podczas, gdy rynek smartfonów dąży do wdrożenia powszechnie standardu USB-C, w przypadku NFC nie wszędzie jest on potrzebny. Ów moduł służy głównie do płatności mobilnych, dlatego na przykład w Chinach, gdzie powszechny jest system płatności przez kody QR[15], jest po prostu zbędny. Zważając na popularność chińskich telefonów na światowym rynku prawdopodobnym jest, iż nawet nowe modele nie będą wspierać technologii NFC, przez co utrudnią, a nawet uniemożliwią korzystanie z kluczy Yubikey.

W proponowanym rozwiązaniu jako klucz sprzętowy zostało wykorzystane urządzenie, które eliminuje wymienione wyżej wady Yubikey. Inteligentna opaska jest przeznaczona do noszenia na ręce, dzięki czemu ciężej ją zgubić lub ukraść. Smartband komunikuje się ze smartfonem poprzez wykorzystywany powszechnie moduł Bluetooth, co pozwoli wdrożyć system w znacznie szerszym gronie urządzeń. Kolejnym atutem wybranego urządzenia jest jego cena. Inteligentną opaskę można nabyć za mniej niż 100 zł, co sprawia, że jest przystępna dla wielu użytkowników. Najważniejszą różnicą między Yubikey a proponowanym systemem jest sposób autentykacji. Dzięki zastosowaniu opaski można stale autoryzować użytkownika bazując na wielu zmiennych czynnikach w przeciwieństwie do Yubikey, które jest jedynie nośnikiem przechowującym klucz, który jest wykorzystywany przy pojedynczych logowaniach bądź jako dodatkowa autoryzacja przy wrażliwych czynnościach.

2.3.2 Haven

Haven [7] jest darmową aplikacją open-source dla urządzeń działających pod systemem Android zaprojektowaną w celu monitorowania aktywności wokół urządzenia, korzystając z jego wbudowanych sensorów. Przy wykryciu zmian w środowisku aplikacja gromadzi zdjęcia oraz nagrania dźwięku, po czym wysyła je poprzez komunikator Signal do użytkownika. Użytkownik może także zdalnie sprawdzić zarejestrowane dane korzystając z “Tor Onion Service”. Aplikacja została stworzona z myślą o dziennikarzach śledczych, którzy są narażeni na ataki ze strony policji bądź innych intruzów.

Główną zaletą Haven jest to, iż potrafi zastąpić drogie fizyczne systemy bezpieczeństwa. Do korzystania z tego rozwiązania wystarczy stary telefon z Androidem oraz opcjonalnie karta SIM, by zapewnić dostęp do mobilnego Internetu. Zapewnia to użytkownikowi tani, a także łatwy w przenoszeniu system pozwalający monitorować na przykład pokój w hotelu. Pozwala to zdobyć dowody w przypadku ataku typu “evil maid”[14], czyli gdy osoba trzecia uzyskuje fizyczny dostęp do urządzenia, wykorzystując nieobecność właściciela, w celu wykradnięcia danych bądź zainstalowaniu szpiegującego oprogramowania.

Wadą Haven jest zdecydowanie fakt, że nie zapobiega ona atakom, tylko zdobywa dowody ich wystąpienia. Kolejnym mankamentem aplikacji jest jej nadmierna czułość. Wykrywane są mikroruchy smartfona oraz drobne dźwięki, przez co korzystanie z Haven w głośniejszych środowiskach, jak na przykład w biurze może wiązać się z setkami fałszywie wykrytych zdarzeń.

Zaproponowany w pracy system również skupia się na atakach wykonywanych poprzez fizyczny dostęp do urządzenia, lecz w zupełnie inny sposób. Proponowana aplikacja ma na celu wykorzystanie danych ze smartbanda w celu zabezpieczenia smartfona, gdy użytkownik nie jest w stanie nadzorować go samodzielnie. W przeciwieństwie do Haven, które wykorzystuje telefon do zbierania informacji, ale w żadnym stopniu nie korzysta z nich żeby uniemożliwić dostęp do urządzenia. Oba rozwiązania monitorują przeróżne wydarzenia rejestrowane przez dostępne sensory. Podczas, gdy Haven skupia się na środowisku, proponowana aplikacja skupia się na samym użytkowniku. Haven również w żadnym stopniu nie analizuje zbieranych danych, gdyż jego głównym zadaniem jest jedynie raportowanie tego, co dzieje się wokoło. Z kolei proponowane rozwiązanie w pewnym stopniu bierze pod lupę gromadzone dane i na ich podstawie określa, kiedy uruchomić blokadę urządzenia.

2.3.3 Android Management API

Android Management API [8] jest częścią Android Enterprise, inicjatywy dostarczającej deweloperom narzędzi pozwalających budować rozwiązania dla przedsiębiorstw w celu zarządzania flotą mobilnych urządzeń. Program ten jest dedykowany dostawcom usług zarządzania mobilnością w przedsiębiorstwie (EMM). Deweloperzy zapewniają swoim klientom lokalną bądź opartą na chmurze konsolę EMM. Wewnątrz konsoli klienci generują tokeny rejestracji urządzeń oraz tworzą zasady zarządzania (policies). Zasada zarządzania reprezentuje grupę ustawień rządzących zachowaniem zarządzanego urządzenia oraz zainstalowanymi aplikacjami. Następnie urządzenia są zapisywane do systemu przy użyciu wcześniej stworzonych tokenów. Podczas rejestracji, każde urządzenie instaluje aplikację towarzyszącą API, Android Device Policy. Kiedy do danego urządzenia są przyporządkowane zasady, powyższa aplikacja automatycznie wdraża je.

Android Management API daje niespotykane poza aplikacjami systemowymi możliwości zarządzania urządzeniem. Pozwala między innymi na:

- wyłączenie określonych modułów komunikacji takich, jak Bluetooth, Wi-Fi, SMS, rozmowy czy USB;
- blokadę instalacji bądź dezinstalacji aplikacji;
- dostosowanie aplikacji dostępnych w sklepie Play;
- wymuszenie określonych ustawień sieci;
- masowe nadanie pozwoleń aplikacjom;
- określenie sposobów autoryzacji oraz wymogów hasła;
- włączenie aplikacji w trybie Kiosk;
- zdalne wymazanie danych z urządzenia.

Więcej informacji na temat dostępnych polityk można znaleźć w [9].

Główną wadą tego API jest brak możliwości zastosowania go poza przedsiębiorstwami. Urządzeniom korzystającym z tego rozwiązania zasady narzucane są odgórnie przez administratora, więc przy ogromnej liczbie smartfonów, gdzie każdy wymagałby innego zestawu zasad oraz ich aktualizacji na bieżąco, zarządzanie byłoby kłopotliwe. Jest to fundamentalny problem, przez który proponowany system nie może wykorzystać możliwości zabezpieczających Android Management API.

Porównując oba rozwiązania można zauważyć, iż są swoimi przeciwieństwami. Tworzony system jest z założenia czysto lokalny oraz tworzony z myślą o zwykłych użytkownikach, dlatego wszystkie mechanizmy zabezpieczające również muszą być aplikowalne bez udziału zewnętrznych serwisów, a także konfigurowalne przez użytkownika. Z kolei przy wykorzystaniu tego API konieczna jest rejestracja urządzenia w systemie EMM danego przedsiębiorstwa, a zasady dotyczące bezpieczeństwa są narzucane z góry.

Projekt aplikacji

W tym rozdziale przedstawiono szczegółowy projekt systemu korzystając z notacji UML oraz uwzględniając założenia funkcjonalne z rozdziału 1. Scharakteryzowano przypadki użycia oraz towarzyszące im scenariusze. Następnie przedstawiono ogólną strukturę aplikacji w diagramie komponentów. W kolejnej sekcji określono tryby działania aplikacji poprzez diagram stanów. W ostatniej sekcji przedstawiono projekt bazy danych.

3.1 Przypadki użycia

Poniżej przedstawiono ogólny diagram przypadków użycia 3.1. Szczegółowe scenariusze zostały zdefiniowane w odpowiednich podsekcjach tego podrozdziału.

Rysunek 3.1: Diagram przypadków użycia.

3.1.1 Ustal hasło

W tym przypadku użytkownik aplikacji ustala hasło, które jest potrzebne przy odblokowywaniu dostępu do zablokowanych aplikacji. Aktorami są Użytkownik oraz System. Obecny przypadek użycia jest inicjowany przy pierwszym uruchomieniu aplikacji, kiedy nie istnieje jeszcze zaszyfrowany plik, w którym będzie przechowywane hasło. Po wykonaniu przypadku w systemie zostaje zarejestrowane hasło użytkownika, które będzie później wykorzystane w celu dezaktywacji blokady aplikacji. Scenariusz składa się z następującego przepływu głównego:

1. Aplikacja wyświetla formularz zawierający pola Hasło oraz Powtórz hasło.
2. Użytkownik wprowadza identyczne hasła do podanych pól oraz zatwierdza wprowadzone dane przyciskiem znajdującym się poniżej.
3. System sprawdza zgodność haseł oraz czy spełniają wymagane kryteria.
4. System tworzy zaszyfrowany plik i zapisuje w nim hasz hasła.

Alternatywnie:

3. Jeśli wprowadzone hasła nie są zgodne:
    1. Zostaje wyświetlony komunikat o błędzie.
    2. Następuje powrót do kroku numer 2 w głównym przepływie.

3.1.2 Parowanie urządzenia

W tym przypadku użytkownik aplikacji skanuje otoczenie, korzystając z modułu Bluetooth w poszukiwaniu najbliższej opaski MiBand, a następnie nawiązuje z nią pierwsze połączenie. Aktorami są Użytkownik, System oraz Opaska. Przypadek ten występuje przy pierwszym uruchomieniu systemu, kiedy zostało już ustalone hasło odblokowujące. Po zakończeniu system jest sparowany z opaską MiBand, z którą komunikacja jest kluczowym punktem działania systemu. W systemie zostaje także zapisany adres MAC opaski, dzięki czemu będzie można z łatwością ponownie połączyć się z nią. Główny przepływ składa się z następujących kroków:

1. Użytkownik naciska przycisk “Skan”.
2. System rozpoczyna skanowanie urządzeń Bluetooth Low Energy w celu znalezienia Opaski.
3. System wyświetla znalezione Opaski w formie listy.
4. Użytkownik wybiera Opaskę, do której się podłączy poprzez naciśnięcie na jej nazwę.
5. System tworzy więź z wybraną Opaską i inicjuje pierwsze połączenie.

Alternatywnie:

3. Jeśli nie znaleziono żadnej Opaski:
    1. Zostaje wyświetlony komunikat o błędzie.
    2. Następuje powrót do kroku numer 1 w głównym przepływie.

5. Jeśli wystąpi błąd połączenia z Opaską:
    1. Następuje powrót do kroku numer 4 w głównym przepływie.

3.1.3 Wybierz aplikacje do zablokowania

W tym przypadku użytkownik aplikacji wybiera z listy zainstalowanych aplikacji te, które będą blokowane przez system kiedy zostanie uruchomiona blokada. Aktorami są Użytkownik oraz System. Przypadek ten występuje, gdy istnieje plik z hasłem, opaska została sparowana, nie ma uruchomionej blokady oraz Użytkownik uruchomił aplikację. Po zakończeniu System posiada informację, na których aplikacjach uruchamiać blokadę. Główny przepływ składa się z następujących kroków:

1. Użytkownik z poziomu głównego ekranu aplikacji przechodzi do Ustawień.
2. Użytkownik wybiera opcję Zablokowane aplikacje.
3. Użytkownik zaznacza aplikację z listy, którą chce zablokować bądź odblokować.
4. System zapisuje aktualny stan aplikacji.

3.1.4 Zmień hasło

W tym przypadku użytkownik aplikacji zmienia hasło odblokowujące dostęp do zablokowanych aplikacji. Aktorami są Użytkownik oraz System. Przypadek ten występuje, gdy istnieje plik z hasłem, opaska została sparowana, nie ma uruchomionej blokady oraz Użytkownik uruchomił aplikację. Po zakończeniu System posiada informację o nowym haśle, które będzie wykorzystywane od tej pory do odblokowywania dostępu. Główny przepływ składa się z następujących kroków:

1. Użytkownik z poziomu głównego ekranu aplikacji wybiera opcję Ustawienia w dolnej nawigacji.
2. Użytkownik wybiera opcję Zmień hasło.
3. Użytkownik wpisuje odpowiednie wartości do pól Stare hasło, Nowe hasło oraz Powtórz nowe hasło oraz zatwierdza wprowadzone dane.
4. System zapisuje hasz nowego hasła w zaszyfrowanym pliku.

Alternatywnie:

3. Jeśli stare hasło jest błędne lub nowe hasło nie zgadza się z powtórzonym:
    1. System wyświetla komunikat o błędnych wprowadzonych danych.
    2. Następuje powrót do kroku numer 3 w głównym przepływie.
4. Jeśli wystąpi błąd zapisu:
    1. System wyświetla komunikat o błędzie zapisu.
    2. Następuje powrót do kroku numer 3 w głównym przepływie.

3.1.5 Sprawdź stan opaski

W tym przypadku użytkownik aplikacji sprawdza podstawowe informacje na temat opaski oraz jej stan baterii. Aktorami są Użytkownik, System oraz Opaska. Przypadek ten występuje, gdy istnieje plik z hasłem, opaska została sparowana, nie ma uruchomionej blokady oraz Użytkownik uruchomił aplikację. Po zakończeniu Użytkownik zna stan baterii oraz inne informacje o Opasce, a System zyskuje odświeżony stan baterii Opaski. Głowny przepływ składa się z następujących kroków:

1. Użytkownik z poziomu głównego ekranu aplikacji wybiera opcję Opaska w dolnej nawigacji.
2. System wyświetla zapisane wcześniej informacje o Opasce.
3. Przejdź do przypadku Zaktualizuj stan baterii

4. System aktualizuje wartość baterii na ekranie.

Alternatywnie:

2. Jeśli nie zapisano wcześniej informacji o Opasce:
    1. System wyświetla w brakujących polach wartość Nieznane.
    2. Następuje przejście do kroku numer 3 w głównym przepływie.

3. Jeśli nastąpi nagła utrata połączenia z Opaską:
    1. Pomiń następny krok w głównym przepływie.

3.1.6 Sprawdź statystyki aktywności

W tym przypadku użytkownik aplikacji sprawdza proste statystyki zarejestrowanej dziennej aktywności. Aktorami są Użytkownik oraz System. Przypadek ten występuje, gdy istnieje plik z hasłem, opaska została sparowana, nie ma uruchomionej blokady oraz Użytkownik uruchomił aplikację. Po zakończeniu Użytkownik zna ilość zarejestrowanych kroków oraz ostatnią zarejestrowaną wartość pulsu. Główny przepływ składa się z następujących kroków:

1. Użytkownik z poziomu głównego ekranu aplikacji wybiera opcję Statystyki w dolnej nawigacji.
2. System wyświetla ostatnie zapisane wartości dziennych kroków oraz pulsu.
3. System aktualizuje wyświetlane wartości, gdy zostaną zapisane nowe.

3.1.7 Uruchomienie zablokowanej aplikacji

W tym przypadku użytkownik aplikacji próbuje uruchomić aplikację z listy zablokowanych podczas, gdy uruchomiona jest blokada. Aktorami są Użytkownik i System. Przypadek ten występuje, gdy istnieje plik z hasłem, opaska została sparowana oraz aplikacja pracuje w trybie blokady. Po zakończeniu Użytkownik posiada autoryzację do interakcji z blokowanymi aplikacjami, a System przechodzi w tryb monitorowania. Główny przepływ składa się z następujących kroków:

1. Użytkownik uruchamia aplikację z listy zablokowanych.
2. System przenosi Użytkownika w ekran wprowadzania hasła.
3. Użytkownik wprowadza hasło i zatwierdza.
4. System wyłącza blokadę i uruchamia usługę monitorującą.

Alternatywnie:

3. Jeśli Użytkownik wprowadził błędne hasło:
    1. System wyświetla komunikat o błędnym haśle.
    2. Następuje ponowne wykonanie kroku 3 z głównego przepływu.

3.1.8 Zaktualizuj inne zdarzenia

W tym przypadku opaska rejestruje jedno z podejrzanych zdarzeń i przesyła informację o tym do aplikacji. Aktorami są Opaska oraz System. Przypadek ten występuje, gdy istnieje plik z hasłem, opaska została sparowana i jest aktywne połączenie między nią a aplikacją. Po zakończeniu System zyskuje sygnał do uruchomienia blokady. Główny przepływ składa się z następujących kroków:

1. Opaska rejestruje jedno z zaprogramowanych zdarzeń.

2. Opaska przesyła kod zdarzenia do Systemu.
3. System porównuje otrzymany kod z zapisanymi kodami podejrzanych zdarzeń.
4. System uruchamia blokadę.

Alternatywnie:

2. Jeśli nastąpi nagła utrata połączenia:
    1. Następuje przejście do kroku numer 4 w głównym przepływie.
3. Jeśli otrzymany kod nie znajduje się na liście podejrzanych:
    1. Pomiń następny krok głównego przepływu.

3.1.9 Zaktualizuj puls

W tym przypadku opaska wykonuje automatyczny pomiar pulsu i przesyła jego wartość do aplikacji. Aktorami są Opaska oraz System. Przypadek ten występuje, gdy istnieje plik z hasłem, opaska została sparowana i jest aktywne połączenie między nią a aplikacją. Po zakończeniu System zyskuje aktualne dane na temat pulsu do analizy stanu użytkownika. Główny przepływ składa się z następujących kroków:

1. Opaska mierzy puls.
2. Opaska przesyła zarejestrowaną wartość do Systemu.
3. System zapisuje otrzymaną wartość.

Alternatywnie:

2. Jeśli nastąpi nagła utrata połączenia:
    1. System uruchamia blokadę.
3. Jeśli otrzymana wartość wynosi 0:
    1. System uruchamia blokadę.

3.1.10 Zaktualizuj ilość kroków z opaski

W tym przypadku opaska rejestruje wykonane kroki oraz przesyła uaktualnioną wartość dziennych wykonanych kroków do aplikacji. Aktorami są Opaska oraz System. Przypadek ten występuje, gdy istnieje plik z hasłem, opaska została sparowana i jest aktywne połączenie między nią a aplikacją. Po zakończeniu System zyskuje aktualne dane na temat wykonanych kroków, które zostaną wykorzystane później do analizy zachowania użytkownika. Główny przepływ składa się z następujących kroków:

1. Opaska rejestruje wykonanie kroków.
2. Opaska przesyła zaktualizowaną wartość dziennych kroków do Systemu.
3. System zapisuje otrzymaną wartość.

Alternatywnie:

2. Jeśli nastąpi nagła utrata połączenia:
    1. System uruchamia blokadę.
3. Jeśli otrzymana wartość jest równa ostatniemu pomiarowi:
    1. System nie zapisuje otrzymanej wartości.

3.1.11 Zaktualizuj stan baterii

W tym przypadku aplikacja odczytuje aktualny stan baterii z opaski. Aktorami są Opaska oraz System. Przypadek ten występuje, gdy istnieje plik z hasłem, opaska została sparowana i jest aktywne połączenie między nią a aplikacją. Po zakończeniu System zyskuje aktualną wartość stanu baterii. Główny przepływ składa się z następujących kroków:

1. System przesyła do Opaski zapytanie o stan baterii.
2. Opaska przesyła do Systemu aktualny stan baterii.
3. System zapisuje nową wartość stanu baterii.

Alternatywnie:

2. Jeśli nastąpi nagła utrata połączenia:
    1. System uruchamia blokadę.

3.1.12 Zaktualizuj ilość kroków z telefonu

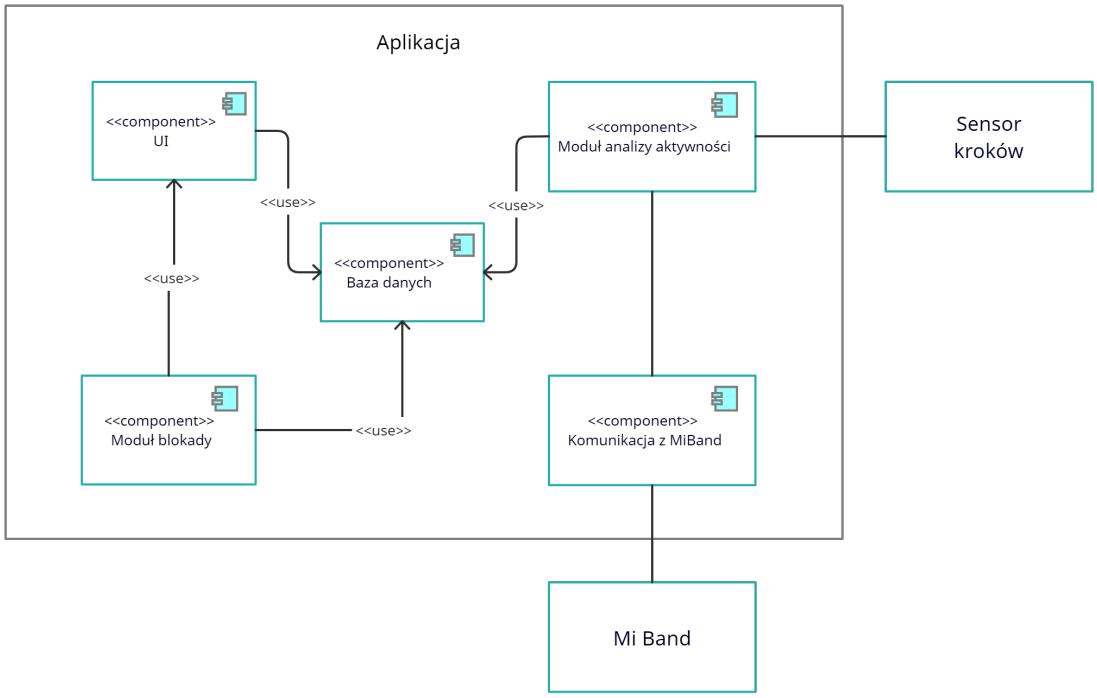
W tym przypadku sensor licznika kroków w smartfonie rejestruje wykonane kroki, a następnie informuje aplikację o aktualizacji swojej wartości. Aktorami są Licznik kroków oraz System. Przypadek ten występuje, gdy istnieje plik z hasłem, opaska została sparowana i jest aktywna usługa monitorująca zachowanie użytkownika. Po zakończeniu System zyskuje informację na temat wykonanych kroków, która zostanie później wykorzystana do analizy zachowania użytkownika.

1. Licznik kroków rejestruje wykonanie kroków.
2. System odczytuje zaktualizowaną liczbę kroków Licznika kroków.
3. System zapisuje odczytaną wartość.

3.2 Diagram komponentów

Na grafice 3.2 przedstawiono ogólny projekt systemu w postaci diagramu komponentów. Jest to uproszczona postać, gdyż implementacja zawiera wiele elementów i bardziej szczegółowe diagramy byłyby zbyt rozległe. Głównym węzłem systemu jest *Aplikacja* zawierająca komponenty: *Baza danych, UI, Moduł blokady, Moduł analizy aktywności* oraz *Komunikacja z Mi Band*. Każdy z nich reprezentuje odpowiednią funkcjonalność systemu.

Komponent *Baza danych* odpowiada za instancję bazy danych oraz wszystkie klasy zapewniające kominikację pomiędzy nią a innymi komponentami. Komponent *UI* jest zbiorem wszystkich klas odpowiedzialnych za wyświetlanie elementów interfejsu oraz aktualizację wyświetlanych danych. Komponent *Moduł blokady* reprezentuje usługi monitorujące aplikacje pierwszoplanowe oraz wykonujące określony zestaw akcji dla każdej z nich. Komponent *Moduł analizy aktywności* zawiera usługę odpowiadającą za analizę zarejestrowanych próbek aktywności użytkownika. Jego działanie jest ściśle powiązane z komponentem *Komunikacja z Mi Band*, który zawiera klasy oraz interfejsy implementujące komunikację Bluetooth z węzłem *Mi Band*. Pozwala to dostarczyć niezbędnych do analizy danych o aktywności użytkownika. Komponent *Moduł analizy aktywności* jest także związany z węzłem *Sensor kroków* zapewniającym dane o wykonanych krokach zarejestrowanych przez smartfon.

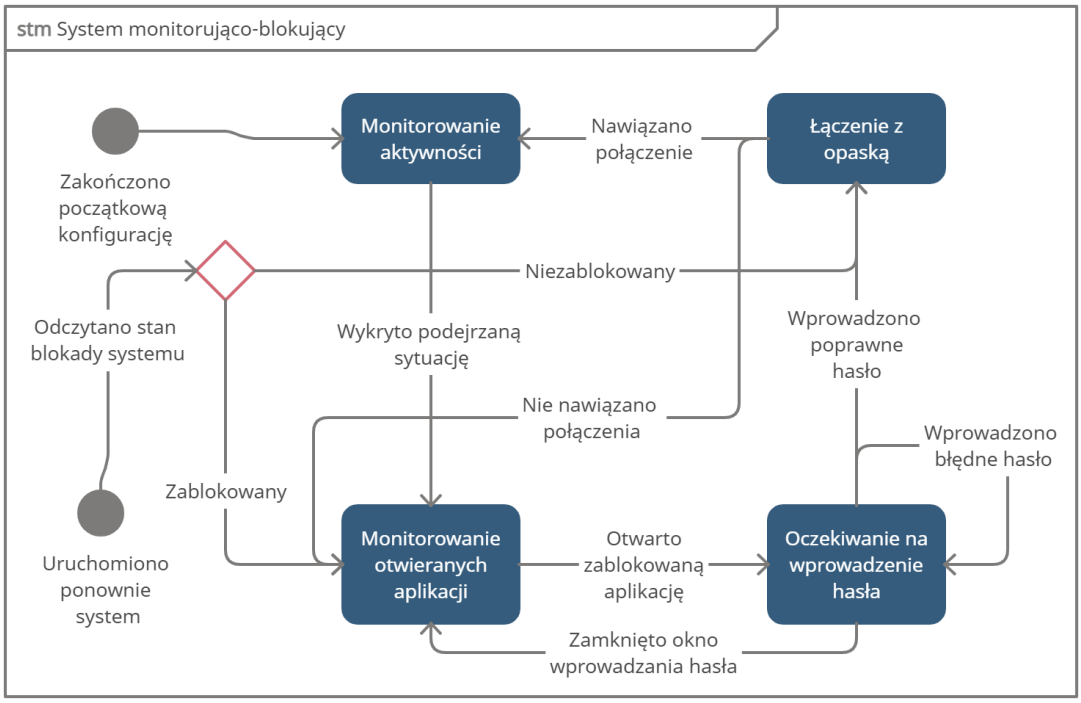


Rysunek 3.2: Diagram komponentów systemu.

3.3 Diagram stanów

Na grafice 3.3 przedstawiono diagram stanów dla systemu monotorująco-blokującego, stanowiącego podstawę działania aplikacji. Diagram ten przedstawia ogólny schemat działania aplikacji w zakresie zabezpieczenia smartfona. Stanami początkowymi są *Zakończono początkową konfigurację* oraz *Uruchomiono ponownie system*. Przedstawiają one dwie sytuacje, w których może nastąpić uruchomienie systemu. W pierwszym przypadku jest to pierwsze uruchomienie aplikacji, gdzie określono hasło i nawiązano połączenie z opaską. Wtedy następuje przejście do stanu *Monitorowanie aktywności*. W przypadku restartu urządzenia wybór stanu, do którego przejdzie aplikacja zależy od ostatniego aktywnego stanu przed wyłączeniem urządzenia. Jeśli była aktywna blokada, aplikacja przejdzie do stanu *Monitorowanie otwieranych aplikacji*. W przeciwnym razie przejdzie do stanu *Łączenie z opaską*.

Następnym analizowanym stanem jest *Monitorowanie aktywności*. W tym stanie aplikacja pobiera dane z opaski i analizuje je pod kątem zdarzeń sugerujących, że użytkownik nie nadzoruje swojego smartfona. Jeśli zostanie wykryte takie zdarzenie następuje przejście do kolejnego analizowanego stanu - *Monitorowanie otwieranych aplikacji*. W tym stanie system skanuje statystyki użycia innych aplikacji w krótkich odstępach czasu. Jeśli zostanie wykryte przejście do aplikacji na liście zablokowanych następuje przejście do stanu *Oczekiwanie na wprowadzenie hasła*. W tym stanie aplikacja wyświetla okno wprowadzania hasła i czeka na akcje użytkownika. Możliwe są trzy przejścia: gdy użytkownik zamknie okno aplikacja powraca do stanu *Monitorowanie otwieranych aplikacji*, gdy wprowadzone hasło jest błędne system pozostaje w tym samym stanie, a gdy wprowadzone zostanie poprawne hasło system przechodzi do stanu *Łączenie z opaską*. W tym stanie aplikacja próbuje nawiązać połączenie z opaską. Gdy zakończy się to sukcesem zachodzi przejście do stanu *Monitorowanie aktywności*. W przeciwnym razie aplikacja powraca do stanu *Monitorowanie otwieranych aplikacji*.



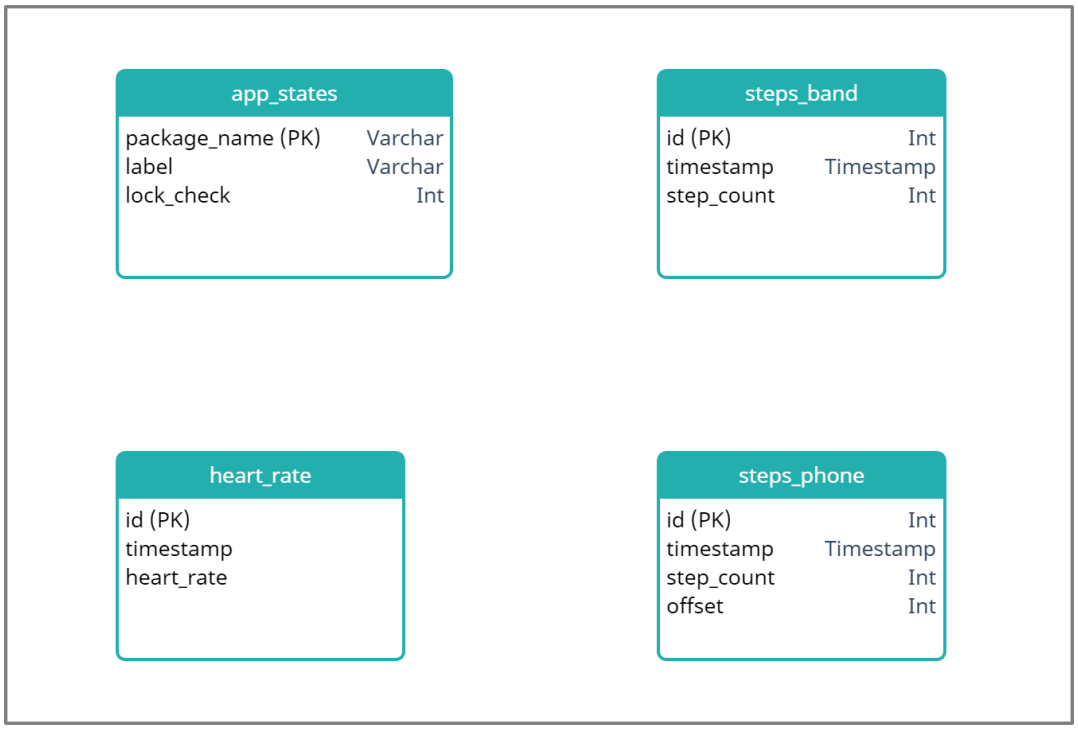
Rysunek 3.3: Diagram stanów systemu monitorująco-blokującego.

3.4 Projekt bazy danych

Na grafice 3.4 przedstawiono tabele znajdujące się w bazie danych aplikacji. Tabele z próbkami aktywności zostały umieszczone w osobnych, niepowiązanych tabelach, by podkreślić rożnice pomiędzy rejestrowanymi rodzajami aktywności. Z kolei tabela zawierająca informacje o blokowanych aplikacjach naturalnie nie ma związku z danymi o aktywności. Proponowana aplikacja na tym stopniu zaawansowania nie wymaga bardziej skomplikowanej struktury bazy.

Tabela *app_states* przechowuje dane pozwalające zidentyfikować każdą zainstalowaną na smartfonie aplikację oraz wartość *lock_check*, która określa, czy dostęp do danej aplikacji będzie możliwy w czasie blokady czy nie - 0 dla dozwolonych aplikacji, a 1 dla niedozwolonych. Łańcuch znaków *label* przedstawia wyświetlaną nazwę aplikacji i jest obecny w tabeli, by umożliwić przyjazne użytkownikowi wyświetlanie jej w interfejsie graficznym.

Tabela *heart_rate* przechowuje zarejestrowane przez opaskę informacje o pulsie wraz z czasem otrzymania danej próbki. Tabela *steps_band* posiada próbki dziennych kroków pobranych z opaski wraz z czasem ich otrzymania. Tabela *steps_phone* zawiera informacje rejestrowane przez sensor kroków w smartfonie wraz z czasem ich zapisu. Dane tej tabeli nieznacznie różnią się od danych z tabeli *steps_band*. Wartość *step_count* to liczba kroków wykonanych danego dnia, natomiast *offset* jest wartością zwracaną przez sensor. Przechowywanie *offsetu* jest konieczne ze względu na sposób, w jaki sensor wykorzystany w aplikacji liczy kroki - są mierzone od ostatniego restartu urządzenia. Dlatego, żeby uzyskać poprawną informację na temat dziennej liczby kroków należy monitorować zmiany w *offsecie*. Wszystkie tabele z danymi o aktywności posiadają autoinkrementowane *id* ze względu na to, że *timestamp* nie zapewnia unikalności Primary Key.



Rysunek 3.4: Tabele w bazie danych aplikacji.

Implementacja aplikacji

W tym rozdziale określono szczegóły implementacyjne aplikacji. Scharakteryzowano wykorzystany język programowania oraz użyte biblioteki. Następnie opisano sposób uniemożliwienia dostępu do blokowanych aplikacji. W kolejnej sekcji opisano dokładnie protokół wykorzystywany do komunikacji z MiBandem 3 wraz z sekwencją autentykującą oraz konfiguracyjną. W ostatniej sekcji określono w jaki sposób aplikacja analizuje dane o aktywności użytkownika.

4.1 Wykorzystane technologie

Aplikacja została napisana przy użyciu języka **Kotlin** w wersji 1.4.31. Język ten, jest tworzony od 2011 roku przez Jetbrains z myślą, by zapewnić przemysłowo silny język obiektowy, który będzie lepszy od Javy ale wciąż będzie z nią w pełni interoperacyjny, aby umożliwić stopniową migrację istniejących już systemów. Głównymi zaletami korzystania z Kotlina w programowaniu na system Android jest zmniejszenie objętości kodu oraz możliwość wykorzystania funkcjonalności języka takich, jak korutyny, funkcje rozrzerzeń czy lambdy w istniejących bibliotekach Androida. Kod Kotlina jest także bardziej czytelny niż Javy, co prowadzi do zmniejszonej liczby błędów.

Do implementacji bazy danych wykorzystano bibliotekę **Room**, tworzącą warstwę abstrakcji nad SQLite, by zapewnić płynny dostęp do bazy, zachowując pełną moc SQLite. Biblioteka ta pozwala między innymi zmniejszyć ilość powtarzalnego, skłonnego do generowania błędów kodu poprzez stosowanie anotacji oraz weryfikować zapytania SQL w czasie kompilacji. Aby zapewnić bezpieczeństwo bazy danych została ona zaszyfrowana przy użyciu biblioteki **SQLCipher**.

Do jeszcze większej optymalizacji kodu wykorzystano bibliotekę **Hilt**, która odpowiada za wstrzykiwanie zależności do klas. W przeciwieństwie do manualnego konstruowania klas wraz z ich zależnościami oraz wykorzystywaniu dodatkowych obiektów do zarządzania nimi, Hilt generuje je sam na podstawie określonych anotacji, a także zapewnia poprawne zarządzanie ich cyklem życia. Dzięki temu przy mniejszej ilości kodu można uzyskać lepszą wydajność w czasie działania aplikacji oraz zmiejszenie liczby błędów podczas kompilacji.

Do implementacji komunikacji z MiBandem została wykorzystana biblioteka **RxJava**, pozwalająca wykorzystać możliwości programowania reaktywnego, czyli paradygmatu programowania polegającego na przetwarzaniu strumieni danych i propagowaniu ich zmian, w aplikacjach na system Android. Przepływ danych pomiędzy aplikacją a opaską został opakowany w obiekty typu *Observable*, co pozwoliło dokładnie monitorować stan wysyłanych danych oraz asynchronicznie oczekiwać odpowiedzi na poszczególne operacje.

Aby usprawnienić wydajność aplikacji wszystkie operacje zapisu do bazy danych oraz komunikacja z opaską są wykonywane w korutynach zoptymalizowanych pod operacje wejścia/wyjścia, dzięki czemu główny wątek aplikacji zostaje odciążony. Wykorzystano również **Data Binding**, aby uprościć odniesienia do elementów UI w kodzie aplikacji, co pozwala zwiększyć odporność na wycieki pamięci oraz wydajność.

4.2 Blokada dostępu do aplikacji

Skuteczny sposób uniemożliwienia dostępu do aplikacji jest jedną z kluczowych części projektowanego systemu. Schemat blokady zaprezentowany w pracy opiera się na monitorowaniu aplikacji na pierwszym planie systemu Android. Aby dostać się do tych informacji wykorzystano usługę *Usage Stats*, która zawiera statystyki korzystania z zainstalowanych aplikacji oraz zdarzenia takie, jak zablokowanie ekranu czy wyświetlenie aktywności danej aplikacji na ekranie. Proponowany system blokady pobiera zdarzenia z ostatnich 5 sekund, a następnie poszukuje w nich typu "MOVE_TO_FOREGROUND". Jeśli takie zdarzenie zostanie odnalezione, a aplikacja, która zainicjowała je, jest na liście aplikacji do zablokowania, następuje przejście do aktywności zawierającej pole do wpisania hasła. W przeciwnym razie system nie reaguje. Jeśli wprowadzone hasło było poprawne, usługa blokująca zostaje zatrzymana i aplikacja nawiązuje próbę połączenia się z opaską. Poniżej przedstawiono zarys algorytmu blokującego w pseudokodzie.

Algorytm 4.1: Blokowanie dostępu do wybranych aplikacji

Dane: Lista zablokowanych aplikacji BA

```
1  begin
2      while blokada jest aktywna do
3          events ← zdarzenia z Usage Stats z ostatnich 5 sekund
4          while events.hasNext() do
5              event ← events.next()
6              if event == "MOVE_TO_FOREGROUND" ∧ event.isIn(BA) then
7                  Uruchom aktywność odpowiedzialną za wpisywanie hasła
8              end if
9          end while
10         Poczekaj 5 sekund
11     end while
12 end
```

4.3 Komunikacja z MiBand 3

W projektowanym systemie główną rolę gra inteligentna opaska. Komunikuje się ona ze smartfonem przy użyciu Bluetooth Low Energy protokołem ATT korzystając z GATT. BLE w porównaniu do klasycznego połączenia Bluetooth wykorzystuje znacznie niższe zasoby energii zachowując podobny zasięg, dzięki czemu znalazło szerokie zastosowanie w urządzeniach peryferyjnych. W poniższych podrozdziałach opisano pokrótce pojęcia ATT i GATT oraz dokładnie przedstawiono zaimplementowany protokół komunikacji z opaską Mi-Band 3.

4.3.1 ATT

Protokół Attribute umożliwia urządzeniu, określonemu jako *serwer*, odsłonić zbiór atrybutów i powiązanych z nimi wartości urządzeniu równorzędnemu, określonemu jako *klient*. Atrybuty odsłonięte przez serwer mogą być odkryte, odczytane bądź nadpisane przez klienta, a także mogą być rozgłaszane przez serwer w ramach powiadomienia lub zasygnalizowania. Atrybut jest dyskretną wartością o trzech właściwościach powiązanych ze sobą:

- typie,
- uchwycie,
- zestawie pozwoleń, które są zdefiniowane przez specyfikację wyższej warstwy wykorzystującą dany atrybut.

Typ atrybutu określa, co reprezentuje dany atrybut poprzez UUID (Universally Unique Identifier), ktore może zostać utworzone przez każdego, a następnie zostać opublikowane. Pozwala to rozpoznać atrybut niezależnie od nadanego mu przez serwer uchwytu. Uchwyt atrybutu jest unikalną, niezerową, 16-bitową wartością, która jednoznacznie identyfikuje dany atrybut w obrębie serwera, pozwalając klientowi odnieść się do niego podczas operacji odczytu oraz zapisu. Pozwolenia mogą być nadane atrybutowi w celu ograniczenia klientowi dostępu do zapisu lub odczytu.

Urządzenie może jednocześnie implementować zarówno rolę serwera jak i klienta oraz obie role mogą funkcjonować współbieżnie oraz komunikować się między sobą. Na każdym urządzeniu Bluetooth może znajdować się maksymalnie jedna instancja serwera.

Wszystkie prośby protokołu Attribute są przesyłane poprzez *nosiciela ATT*. Między urządzeniami może być wielu nosicieli, gdzie każdy z nich korzysta z osobnego kanału L2CAP oraz może mieć inną konfigurację. W przypadku BLE, wykorzystywany jest pojedynczy nosiciel ATT, który używa stałego kanału dostępnego od ustanowienia połączenia ACL. Można jednak skonfigurować dodatkowych nosicieli używając L2CAP. Więcej informacji na temat protokołu ATT można znaleźć w [2].

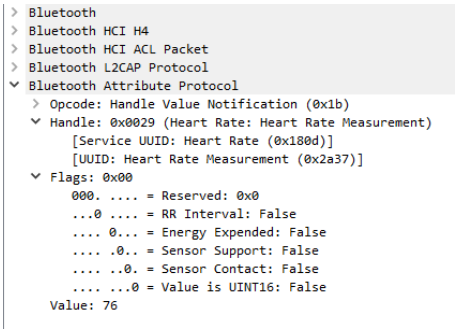
4.3.2 GATT

Profil Generic Attribute (GATT) definiuje framework wykorzystujący protokół Attribute, określający procedury i formaty danych znajdujących się wewnątrz profilu. Zdefiniowane procedury obejmują odkrywanie, odczyt, zapis, powiadamianie oraz sygnalizację. Profil ten został zaprojektowany do wykorzystania przez aplikacje bądź inny profil, aby umożliwić klientowi komunikację z serwerem poprzez opakowanie protokołu ATT w bardziej przystępną formę.

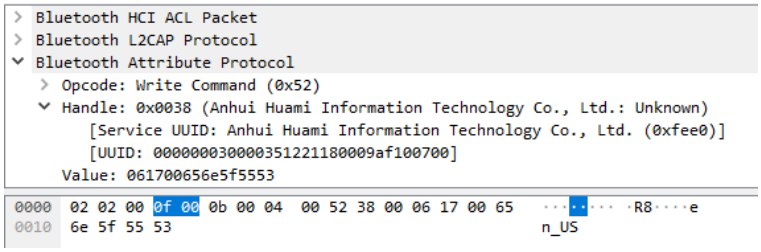
Profil GATT określa strukturę, w której odbywa się wymiana danych. Najwyższym poziomem jest profil zawierający liczne *usługi* będące zbiorem danych oraz przypisanych im zachowań niezbędnych do zapewnienia określonej funkcji. Usługi składają się z *charakterystyk*, z których każda zawiera określoną wartość oraz opcjonalne informacje na jej temat. Usługi oraz charakterystyki wraz ze swoimi komponentami zawierają dane profilu, które są przechowywane w Atrybutach na serwerze. Dzięki wykorzystaniu określonej struktury danych przez GATT możliwe jest przeglądanie dostępnych Usług oraz Charakterystyk, nawet gdy klient nie jest wyspecjalizowany pod dany serwer. Więcej informacji na temat GATT można znaleźć w [2].

4.3.3 Inżynieria wsteczna protokołu komunikacji opaski

Zważywszy na to, iż protokół komunikacji opaski MiBand 3 nie jest jawny, konieczne było zastosowanie inżynierii wstecznej w celu identyfikacji usług oraz charakterystyk wraz z ich funkcjonalnością. Dane do analizy uzyskano przechwytując pakiety ATT między opaską a aplikacją Gadgetbridge [16] przy użyciu sniffera Wireshark [5]. Korzystając z faktu, że Gadgetbridge jest open-source możliwe było użycie kodu źródłowego aplikacji do interpretacji zaobserwowanych przesyłanych wartości. Na poniższych grafikach przedstawiono dwa przykładowe pakiety przechwycone w czasie analizy protokołu.



Rysunek 4.1: Przechwycony pakiet zawierający wartość akcji serca.



Rysunek 4.2: Przechwycony pakiet zawierający konfigurację języka w opasce na angielski.

4.3.4 Wykorzystane usługi i charakterystyki

Na podstawie przeprowadzonych analiz komunikacji, wyszczególniono poniższe usługi i charakterystyki, które posłużyły do realizacji funkcjonalności tworzonej aplikacji.

Usługa 0000fee0-0000-1000-8000-00805f9b34fb

Jest to usługa odpowiadająca w głównej mierze za podstawowe funkcjonalności opaski MiBand 3. Pozwala zmodyfikować ustawienia urządzenia, zapisać informacje o użytkowniku oraz odczytać dane o stanie baterii i aktywności użytkownika. Jest to najczęściej wykorzystywana usługa w aplikacji.

Charakterystyka	Opis
00000003-0000-3512-2118-0009af100700	Charakterystyka pozwalająca na konfigurację ustawień opaski.
00000006-0000-3512-2118-0009af100700	Charakterystyka zawierająca informacje o stanie baterii opaski.
00000007-0000-3512-2118-0009af100700	Charakterystyka przechowująca liczbę wykonanych kroków danego dnia.
00000008-0000-3512-2118-0009af100700	Charakterystyka zawierająca dane użytkownika.
00000010-0000-3512-2118-0009af100700	Charakterystyka zawierająca informacje na temat zdarzeń wykrywanych przez opaskę.
00002a2b-0000-1000-8000-00805f9b34fb	Charakterystyka przechowująca aktualną datę i godzinę,

Tablica 4.1: Wykorzystane charakterystyki z usługi 0000fee0-0000-1000-8000-00805f9b34fb.

Usługa 0000fee1-0000-1000-8000-00805f9b34fb

Jest to usługa zawierająca przede wszystkim charakterystykę wykorzystywaną w procesie autentykacji połączenia oraz parowania. W usłudze tej znajduje się też sporo niezidentyfikowanych charakterystyk, które nie są wymagane do działania aplikacji. Dlatego więc zostały pominięte.

Charakterystyka	Opis
00000009-0000-3512-2118-0009af100700	Charakterystyka wykorzystywana do autentykacji połączenia między opaską a klientem

Tablica 4.2: Wykorzystane charakterystyki z usługi 0000fee1-0000-1000-8000-00805f9b34fb.

Usługa 0000180d-0000-1000-8000-00805f9b34fb

Jest to usługa zdefiniowana przez Bluetooth Special Interest Group odpowiadająca za komunikację między sensorem akcji serca a innym klientem GATT. Za jej pomocą można uzyskać informację o pulsie użytkownika oraz skonfigurować automatyczne pomiary.

Charakterystyka	Opis
00002a37-0000-1000-8000-00805f9b34fb	Charakterystyka wykorzystywana do odczytu aktualnej wartości pulsu użytkownika.
00002a39-0000-1000-8000-00805f9b34fb	Charakterystyka odpowiedzialna za konfigurację sensora akcji serca w opasce.

Tablica 4.3: Wykorzystane charakterystyki z usługi 0000180d-0000-1000-8000-00805f9b34fb.

Usługa 0000180a-0000-1000-8000-00805f9b34fb

Jest to usługa również zdefiniowana przez Bluetooth Special Interest Group. Odpowiada za dostarczenie informacji o urządzeniu. W tym wypadku jest to informacja o numerze seryjnym opaski oraz wersjach sprzętu i oprogramowania.

Charakterystyka	Opis
00002a25-0000-1000-8000-00805f9b34fb	Charakterystyka wykorzystywana do odczytu numeru seryjnego opaski.
00002a27-0000-1000-8000-00805f9b34fb	Charakterystyka wykorzystywana do odczytu wersji sprzętowej opaski.
00002a28-0000-1000-8000-00805f9b34fb	Charakterystyka wykorzystywana do odczytu wersji oprogramowania opaski.

Tablica 4.4: Wykorzystane charakterystyki z usługi 0000180a-0000-1000-8000-00805f9b34fb.

4.3.5 Autentykacja połączenia

Aby konfigurować oraz odczytywać informacje o aktywności z MiBanda wymagana jest prosta autoryzacja. W przeciwnym razie ma dostępu do większości charakterystyk urządzenia, a połączenie zostanie zerwane po 30 sekundach. Sekwencja autentykacyjna wygląda w następujący sposób. Najpierw do opaski należy wysłać klucz, który zostanie wykorzystany do autentykacji połączenia. Wykonuje się to przy pierwszym połączeniu z urządzeniem. Następnie wysyłana jest prośba do MiBanda o podanie losowej liczby. Po jej otrzymaniu należy zaszyfrować ją algorytmem AES korzystając z klucza podanego przy parowaniu opaski, odesłać i przesłać aktualną datę do odpowiedniej charakterystyki. Po otrzymaniu potwierdzenia połączenie jest zatwierdzone i można przejść do sekwencji konfigurującej. Aktualna data przesyłana jest jako tablica bajtów. Dostęp do odpwiednich wartości można uzyskać przez przesunięcie bitowe i maskowanie odpowiednich bitów, co przedstawiono dokładnie na grafice poniżej:

rok & 0xFF	(rok >> 8) & 0xFF	miesiąc & 0xFF	dzień & 0xFF	godzina & 0xFF	minuty & 0xFF	sekundy & 0xFF	dzień tygodnia & 0xFF	0

Rysunek 4.3: Struktura tablicy bajtów zawierającej aktualną datę.

Algorytm 4.2: Parowanie z MiBandem 3

Dane: Tablica bajtów zawierająca klucz wykorzystywany do szyfrowania K, Zmienna logiczna *pair* mówiąca, czy urządzenie jest sparowane

begin
  $authorisationChar \leftarrow 00000009-0000-3512-2118-0009af100700$
  Włącz powiadomienia na $authorisationChar$
  if *pair* **then**
    Wyślij do $authorisationChar \longrightarrow ByteArray(0x01, 0x00) + K$
    if $Otrzymano\ odpowiedz \leftarrow ByteArray(0x10, 0x01, 0x01)$ **then**
      Przejdź do autentykacji
    end if
  end if
end

Algorytm 4.3: Autentykacja z MiBandem 3

Dane: Tablica bajtów zawierająca klucz wykorzystywany do szyfrowania K

begin
  $authorisationChar \leftarrow 00000009-0000-3512-2118-0009af100700$
  $timeChar \leftarrow 00002a2b-0000-1000-8000-00805f9b34fb$
  Włącz powiadomienia na $authorisationChar$
  Wyślij do $authorisationChar \longrightarrow ByteArray(0x02, 0x00)$
  if $Otrzymano\ odpowiedz \leftarrow ByteArray(0x10, 0x02, 0x01, \dots)$ **then**
    $numToEncrypt \leftarrow odpowiedz\,[3:19]$
    Wyślij do $authorisationChar \longrightarrow ByteArray(0x03, 0x00) + AES(K, numToEncrypt)$
    Wyślij do $timeChar$ aktualną datę
  end if
  if $Otrzymano\ odpowiedz \leftarrow ByteArray(0x10, 0x03, 0x01, \dots)$ **then**
    Przejdź do konfiguracji
  end if
end

4.3.6 Sekwencja konfigurująca

Po udanej autoryzacji połączenia należy skonfigurować działanie opaski. Odbywa się to za pomocą sekwencji operacji zapisu, głównie do charakterystyki o UUID równym “00000003-0000-3512-2118-0009af100700” oraz włączeniu powiadamiania na odpowiednich charakterystykach. Po skonfigurowaniu urządzenie będzie automatycznie przesyłać odpowiednie dane o aktywności w najmniejszych możliwych odstępach czasu. Na poniższych grafikach przedstawiono struktury tablic bajtów wysyłanych danych, które były zbyt obszerne, aby przedstawić je w pseudokodzie.

0x64	0x64	0x2F	0x4D	0x4D	0x2F	0x79	0x79	0x79	0x79

Rysunek 4.4: Tablica bajtów zawierająca format wyświetlanej w opasce daty - dd/MM/yyyy.

cel & 0xFF	(cel >> 8) & 0xFF

Rysunek 4.5: Tablica bajtów zawierająca cel aktywności - liczbę kroków do wykonania dziennie.

0x4F	0x00	0x00	rok urodzenia & 0xFF	(rok urodzenia >> 8) & 0xFF	miesiąc & 0xFF	dzień & 0xFF	płeć (0x00 lub 0x01)	wzrost & 0xFF	(wzrost >> 8) & 0xFF	(waga * 200) & 0xFF	(waga * 200 >> 8) & 0xFF	id & 0xFF	(id >> 8) & 0xFF	(id >> 16) & 0xFF	(id >> 24) & 0xFF

Rysunek 4.6: Tablica bajtów zawierająca informacje o użytkowniku.

Algorytm 4.4: Konfiguracja MiBanda 3 - Część 1

begin
- $settingsChar \leftarrow 00000003 - 0000 - 3512 - 2118 - 0009af100700$
- $batteryChar \leftarrow 00000006 - 0000 - 3512 - 2118 - 0009af100700$
- $eventsChar \leftarrow 00000010 - 0000 - 3512 - 2118 - 0009af100700$
- $userChar \leftarrow 00000008 - 0000 - 3512 - 2118 - 0009af100700$
- $hrControlChar \leftarrow 00002a39 - 0000 - 1000 - 8000 - 00805f9b34fb$
- $hrChar \leftarrow 00002a37 - 0000 - 1000 - 8000 - 00805f9b34fb$
- $stepsChar \leftarrow 00000007 - 0000 - 3512 - 2118 - 0009af100700$
- $serialNumChar \leftarrow 00002a25 - 0000 - 1000 - 8000 - 00805f9b34fb$
- $hardwareChar \leftarrow 00002a27 - 0000 - 1000 - 8000 - 00805f9b34fb$
- $softwareChar \leftarrow 00002a28 - 0000 - 1000 - 8000 - 00805f9b34fb$
-
- Włącz powiadomienia na $settingsChar, batteryChar$ oraz $eventsChar$
- Odczytaj wartość z $serialNumChar, hardwareChar, softwareChar$ oraz $batteryChar$
- Ustaw język angielski wysyłając $ByteArray(0x06, 0x17, 0x00, 0x65, 0x6e, 0x5f, 0x55, 0x53)$ do $settingsChar$
- Wyłącz odblokowywanie ekranu w opasce wysyłając $ByteArray(0x06, 0x16, 0x00, 0x00)$ do $settingsChar$
- Wyłącz tryb nocny wysyłając $ByteArray(0x1a, 0x00)$ do $settingsChar$
- Ustaw format daty wysyłając $ByteArray(0x06, 0x1e, 0x00) + format$ do $settingsChar$, gdzie $format$ to opisany powyżej przetworzony łańcuch znaków
- Ustaw format wyświetlanej daty wysyłając $ByteArray(0x06, 0x0a, 0x00, 0x03)$ do $settingsChar$
- Ustaw 24-godzinny zegar wysyłając $ByteArray(0x06, 0x02, 0x00, 0x01)$ do $settingsChar$
- Ustaw przykładowe dane o użytkowniku wysyłając $userInfo$ do $settingsChar$, gdzie $userInfo$ to przetworzone dane o użytkowniku opisane powyżej
- Ustaw jednostki metryczne wysyłając $ByteArray(0x06, 0x03, 0x00, 0x00)$ do $settingsChar$
- Włącz powiadomienia na $userChar$
- Ustaw lokalizację noszenia opaski na lewą rękę wysyłając $ByteArray(0x20, 0x00, 0x00, 0x02)$ do $userChar$
- Ustaw cel fitness wysyłając $ByteArray(0x10, 0x00, 0x00, x[0], x[1], 0x00, 0x00)$ do $userChar$, gdzie x to przetworzona wartość celu
- Ustaw elementy menu opaski wysyłając $ByteArray(0x0a, 0x7f, 0x30, 0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08)$ do $settingsChar$
- Wyłącz w opasce tryb “Nie przeszkadzać” wysyłając $ByteArray(0x09, 0x82)$ do $settingsChar$

end

Algorytm 4.5: Konfiguracja MiBanda 3 - Część 2

begin
- Wyłącz gest obrotu nadgarstka do zmiany pozycji w menu wysyłając $ByteArray(0x06, 0x0d, 0x00, 0x00)$ do *settingsChar*
- Wyłącz gest podniesienia nadgarstka do włączenia ekranu opaski wysyłając $ByteArray(0x06, 0x05, 0x00, 0x00)$ do *settingsChar*
- Włącz wyświetlanie ID dzwoniącego w opasce wysyłając $ByteArray(0x06, 0x10, 0x00, 0x00, 0x01)$ do *settingsChar*
- Wyłącz powiadomienia o celu fitness wysyłając $ByteArray(0x06, 0x06, 0x00, 0x00)$ do *settingsChar*
- Wyłącz powiadomienia o nieaktywności wysyłając $ByteArray(0x08, 0x00, 0x3c, 0x00, 0x04, 0x00, 0x15, 0x00, 0x00, 0x00, 0x00, 0x00)$ do *settingsChar*
- Włącz wspomaganie detekcji snu przez pomiar pulsu wysyłając $ByteArray(0x15, 0x00, 0x01)$ do *hrControlChar*
- Włącz powiadomienie o utracie połączenia wysyłając $ByteArray(0x06, 0x0c, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00)$ do *settingsChar*
- Włącz powiadomienie o nawiązaniu połączenia wysyłając $ByteArray(0x06, 0x01, 0x00, 0x01)$ do *settingsChar*
- Ustaw interwał automatycznego pomiaru pulsu na minimalny wysyłając $ByteArray(0x14, 0x01)$ do *hrControlChar*
- Poproś o alarmy wysyłając $0x0d$ do *settingsChar*
- Włącz powiadomienia na *hrChar* oraz *stepsChar*

end

4.4 Analiza rejestrowanych danych o aktywności

Aby proponowany system działał poprawnie niezbędna jest analiza gromadzonych przez opaskę oraz sensor kroków danych. Aplikacja rozróżnia 4 typy aktywności:

- puls,
- kroki rejestrowane w opasce,
- kroki rejestrowane w telefonie,
- oraz inne zdarzenia.

Próbki pulsu oraz kroków są przechowywane w bazie danych w celu analizy. Natomiast wykrycie zdarzeń innego typu samo w sobie jest przesłanką do zablokowania systemu, więc nie ma potrzeby ich przechowywania. W poniższych akapitach opisano sposoby monitorowania zarejestrowanych próbek i zdarzeń oraz sytuacje, w których jest uruchamiana blokada.

Dane o akcji serca są analizowane w następujący sposób. Każda nadchodząca próbka z pomiarem równym 0 oraz brak zarejestrowania żadnej próbki w ciągu 90 sekund, świadczy o tym, że opaska została zdjęta i jest to przesłanka do blokady systemu. Z testów przeprowadzonych podczas rozwoju aplikacji wynikło, że monitorowanie tych czynników z wykorzystaniem podanych wartości jest ważne dla poprawnego określenia, czy opaska znajduje się na ręce, gdyż MiBand 3 rejestruje zdarzenia z opóźnieniem, więc odpowiednie zdarzenie o zdjęciu opaski nadchodziło po kilku minutach, co nie zapewnia wystarczającego bezpieczeństwa. Z kolei łączenie z opaską również jest obarczone sporym opóźnieniem, dlatego czasem w ciągu 60 sekund (najkrótszy możliwy okres pomiędzy automatycznymi pomiarami pulsu) od włączenia systemu monitorującego nie rejestrowano żadnej próbki pulsu. Z tego powodu okres pomiędzy pobieraniem próbek z bazy musiał zostać

zwiększony do 90 sekund.

Dane o krokach są analizowane na podstawie różnic w tempie wzrostu między próbkami z opaski oraz smartfona. Co 90 sekund system pobiera próbki z ostatnich 90 sekund. Jeśli dla jednego ze źródeł nie zarejestrowano nowych próbek, system pobiera datę ostatniego pomiaru z tego źródła, a następnie dopisuje do listy próbek z drugiego źródła próbki starsze niż 90 sekund i młodsze od ostatniej próbki z pierwszego źródła. Krok ten został podjęty w celu wykrycia sytuacji, gdzie tylko jedno urządzenie porusza się, jednak na tyle wolno, że system myślałby, że tempa wzrostu pomiędzy tymi źródłami są akceptowalne. Tempo wzrostu jest określone jako amplituda wartości kroków z pobranych wcześniej próbek, biorąc poprawkę na możliwość pobrania próbek z okresu, gdy zmienia się data, co powoduje rozpoczęcie liczenia kroków od 0. Akceptowalna wartość amplitudy wynosi $\leqslant 30$. Jest to uwarunkowane tym, że przy niższych wartościach system mógłby aktywować się przy poruszaniu się wraz z telefonem, a przy wyższych system nie zareagowałby w wystarczającym stopniu na podejrzane sytuacje.

Oprócz analizy próbek aktywności system reaguje też na wykrywane zdarzenia. System rejestruje zdjęcie opaski, zaśnięcie oraz utratę połączenia. W przypadku dwóch pierwszych sytuacji, blokada jest uruchamiana natychmiastowo. Natomiast w przypadku utraty połączenia podejmowane są dwie próby ponownego połączenia. Jeśli nie uda się nawiązać ponownie połączenia uruchamiana jest blokada.

Instrukcja obsługi

W tym rozdziale omówiono wymagania wobec środowiska, w którym będzie instalowana będzie opisana w pracy aplikacja. Przedstawiono procedurę instalacji oraz jak przeprowadzić podstawową konfigurację. W następnej sekcji przedstawiono kilka przykładów użycia aplikacji.

5.1 Instalacja i konfiguracja

5.1.1 Wymagania sprzętowe

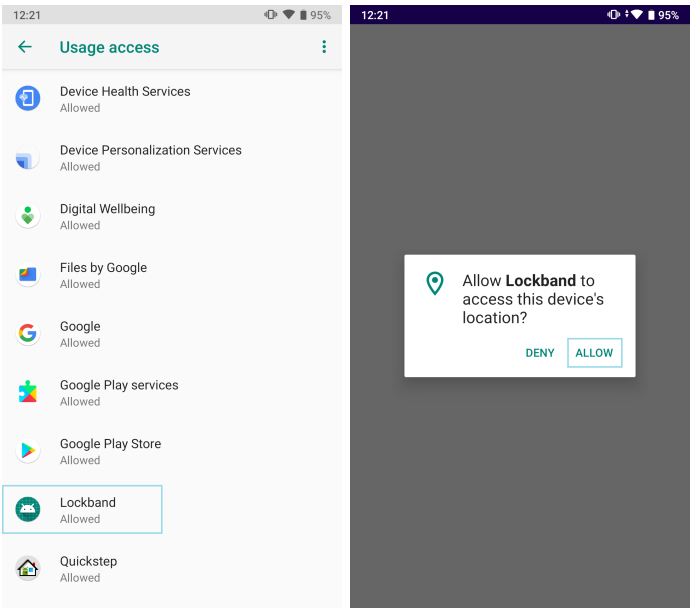
Aplikacja została stworzona pod system Android Pie. Do działania jest wymagany moduł Bluetooth w wersji co najmniej 4.0 oraz posiadanie smartbanda MiBand 3. Do zainstalowania aplikacji potrzeba minimalnie około 23 MB wolnego miejsca.

5.1.2 Instalacja

Aplikację można zainstalować przy użyciu dołączonego do pracy pliku *lockband.apk*. Aby umożliwić instalowanie aplikacji z plików APK należy nadać odpowiedniej aplikacji (w tym przypadku menadżerowi plików) pozwolenie na instalowanie nieznanych aplikacji. Lokalizacja tego ustawienia może się znacząco różnić w zależności od wersji nakładki systemowej. Następnie należy pobrać plik APK na smartfona. W kolejnym kroku należy znaleźć pobrany wcześniej plik korzystając z menedżera plików. Ostatnim krokiem jest stuknięcie w plik APK, by uruchomić instalację.

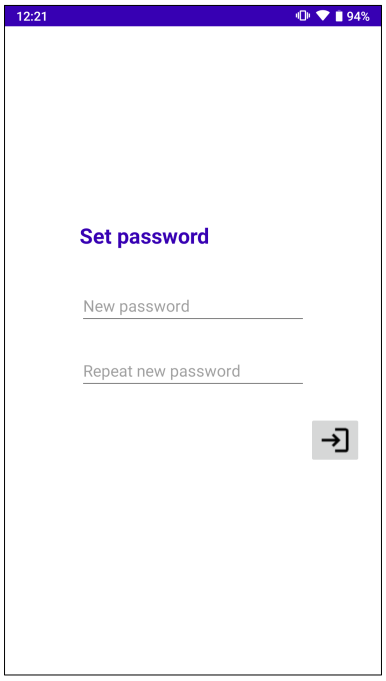
5.1.3 Pierwsze uruchomienie

Podczas pierwszego uruchomienia aplikacja na początku prosi o udzielenie potrzebnych pozwoleń: Lokalizacji, potrzebnej do korzystania z Bluetooth oraz Dostępu do danych o użyciu, koniecznych do monitorowania wyświetlanych aplikacji. Użytkownik zostaje przekierowany do ustawień, gdzie może udzielić pozwolenia na wykorzystanie danych o użyciu. Następnie użytkownik może powrócić do aplikacji, gdzie zostanie wyświetlony monit o udzielenie pozwolenia na dostęp do lokalizacji.



Rysunek 5.1: Udzielanie pozwoleń aplikacji.

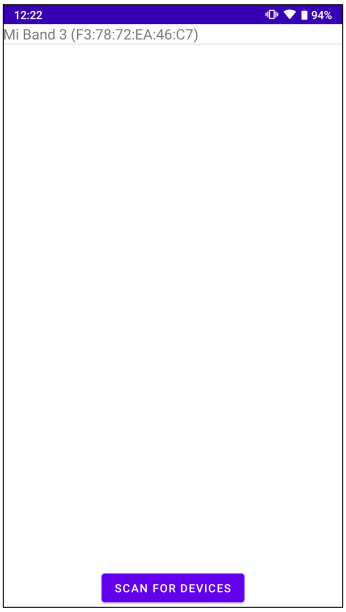
Kolejnym etapem konfiguracji jest ustalenie hasła, które zostanie wykorzystane do odblokowania dostępu do chronionych aplikacji. Użytkownikowi zostaje wyświetlony formularz, w którym należy wprowadzić dwa identyczne hasła, a następnie zatwierdzić je przyciskiem poniżej. Wygląd formularza można zobaczyć w poniższej grafice.



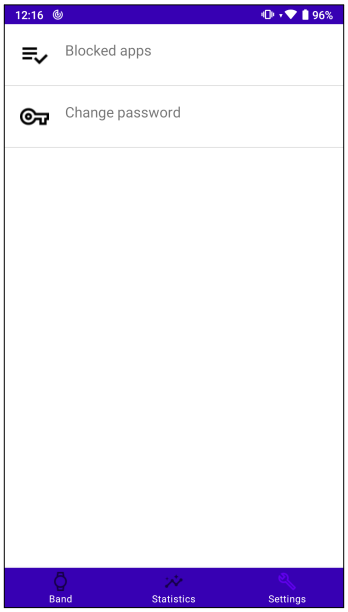
Rysunek 5.2: Formularz kreacji hasła.

Ostatnim etapem konfiguracji jest sparowanie opaski MiBand 3 z aplikacją. W tym celu zostaje wyświetlona aktywność odpowiedzialna za skanowanie pobliskich urządzeń Bluetooth. Po naciśnięciu przycisku *Scan*

for devices zostaje uruchomione skanowanie z nałożonym filtrem na typ urządzenia. Jeśli za pierwszym razem opaska nie zostanie znaleziona należy powtórzyć skan. Po znalezieniu urządzenia zostaje ono wyświetlone na liście. Aby rozpocząć proces parowania należy nacisnąć nazwę danego urządzenia. Po zakończeniu parowania konfiguracja dobiega końca i użytkownik zostaje przeniesiony do głównego menu aplikacji. Inne sprawy konfiguracyjne takie, jak wybranie chronionych aplikacji czy zmiana hasła są dostępne w pozycji *Settings* w menu dolnym aplikacji.



Rysunek 5.3: Lista znalezionych w czasie skanu opasek MiBand 3.



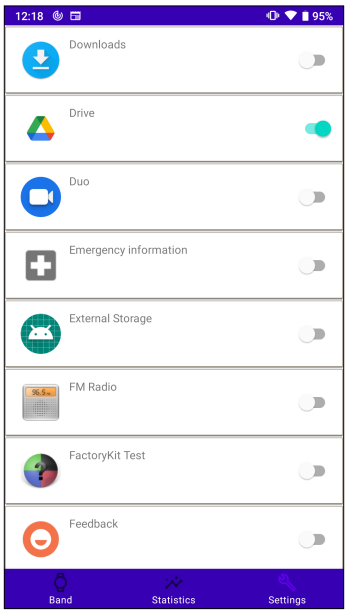
Rysunek 5.4: Ustawienia aplikacji.

5.2 Przykłady użycia

W tej sekcji przedstawiono jak działa aplikacja od strony użytkownika poprzez opis czynności potrzebnych do wykonania określonych zadań, np. sprawdzenie stanu opaski, zmiana blokowanych aplikacji czy sprawdzenie statystyk.

5.2.1 Wybór aplikacji do zablokowania

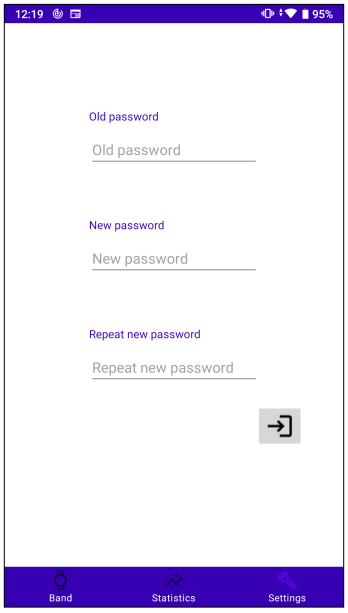
Aby system zapewniał należytą ochronę musi wiedzieć, co jest chronione. W tym celu należy wybrać z listy zainstalowanych aplikacji, te które będą blokowane. Do listy tej można przejść z poziomu panelu ustawień 5.4, wybierając opcję *Blocked apps*. Po naciśnięciu wyświetli się lista wszystkich aplikacji znajdujących się na telefonie. Aby objąć ochroną aplikację należy wykorzystać znajdujący się przy niej przełącznik. Kiedy ma kolor niebieski, aplikacja przy próbie uruchomienia w trakcie blokady będzie przeniosić użytkownika do formularza wprowadzania hasła 5.9.



Rysunek 5.5: Panel wyboru chronionych aplikacji.

5.2.2 Zmiana hasła użytkownika

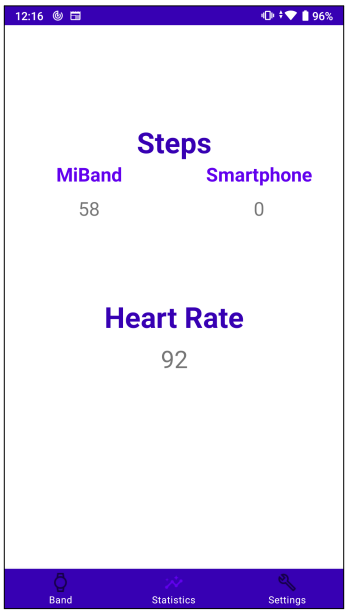
W przypadku, gdy użytkownik chce zmienić hasło wykorzystywane do dezaktywacji blokady, należy udać się do panelu ustawień 5.4 i wybrać opcję *Change password*. Po naciśnięciu pojawi się formularz zmiany hasła, gdzie należy wprowadzić stare hasło oraz dwa razy nowe hasło. Aby zapisać zmiany trzeba wcisnąć przycisk poniżej pól tekstowych.



Rysunek 5.6: Panel zmiany hasła.

5.2.3 Wyświetlenie statystyk aktywności

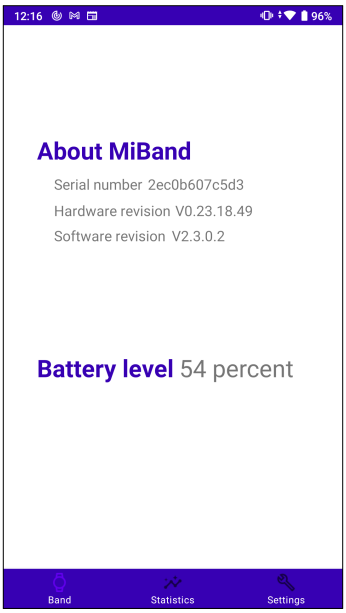
Aplikacja dostarcza użytkownikowi widok, w którym podane są informacje na temat zarejestrowanych danego dnia kroków oraz ostatnia wartość pomiaru pulsu. Aby zobaczyć te dane należy uruchomić aplikację oraz wybrać z dolnego menu pozycję *Statistics*.



Rysunek 5.7: Panel informujący o ostatnich zarejestrowanych aktywnościach.

5.2.4 Wyświetlenie informacji o opasce

Aplikacja zapewnia także panel, w którym można sprawdzić podstawowe informacje o opasce oraz jej stan baterii. Aby uruchomić ten panel należy wejść do głównego widoku aplikacji i wybrać z menu dolnego pozycję *Band*.



Rysunek 5.8: Panel informujący o stanie opaski.

5.2.5 Odblokowanie systemu

Wyłączenie aktywnej blokady odbywa się w na kilka sposobów. Użytkownik może:

- nacisnąć powiadomienie o blokadzie;
- uruchomić aplikację;
- lub podjąć próbę uruchomienia jednej z chronionych aplikacji.

W każdy z tych przypadków system przekieruje użytkownika do formularza wprowadzania hasła. Po wpisaniu poprawnej wartości oraz zatwierdzeniu jej blokada zostaje zdjęta, a następnie uruchamiana jest usługa monitorująca aktywność.

Rysunek 5.9: Formularz wprowadzania hasła.

Podsumowanie

W pracy dokonano analizy problemu wykorzystania fizycznych kluczy bezpieczeństwa w celu zabezpieczenia smartfonów oraz zapewnienia stałej autoryzacji dla systemu Android. Przedstawiono któtki opis proponowanego rozwiązania tego problemu. Następnie przedstawiono istniejące klucze działające z urządzeniami mobilnymi oraz rozwiązania pozwalające zabezpieczyć te urządzenia. Omówiono różnice między nimi a projektowanym systemem. W kolejnym rozdziale przedstawiono projekt aplikacji przy użyciu diagramów UML. W następnym rozdziale opisano wykorzystane do implementacji technologie. Zaprezentowano także szczegóły systemu blokującego dostęp do aplikacji. Opisano również wyczerpująco protokół komunikacji z opaską MiBand 3. Określono też sposoby analizy rejestrowanych danych o aktywności użytkownika. Na końcu przedstawiono instrukcję korzystania z aplikacji dla użytkownika wraz z wymaganiami sprzętowymi oraz procedurą instalacji.

6.1 Uzyskane wyniki

W implementacji aplikacji udało się zrealizować wszystkie wymagania funkcjonalne postawione we wstępie pracy. Jest jednak wiele rzeczy, które wymagają usprawnienia zanim system będzie mógł zostać wykorzystany do prawdziwej ochrony smartfona. Główną luką bezpieczeństwa są ograniczenia sprzętowe.

Opaska MiBand 3 posiada dość mało bezpieczny protokół parujący, gdyż wysyła klucz wykorzystywany później do autentykacji pakietem ATT. Kolejnym mankamentem jest prędkość rejestrowania zdarzeń przez opaskę. Zanim zostanie zarejestrowane zdarzenie zdjęcia opaski mija często kilka minut co sprawia, że jest to praktycznie bezużyteczna informacja. Pomiar akcji serca odbywa się w minimalnych odstępach minutowych, co jest zbyt długim okresem, by szybko reagować na zmiany.

Z kolei system Android wymaga do inwazyjnych czynności takich, jak zamykanie innych aplikacji, aplikacji podpisanej przez system. Utrudnia to bardzo projektowanie aplikacji, które mogłyby być dystrybuowane użytkownikom poprzez Sklep Play. Dlatego w stworzonej aplikacji nie ma stuprocentowej możliwości zablokowania dostępu do innych aplikacji. Obecnie możliwe jest otworzenie stworzonej w pracy aplikacji, gdy zarejestrujemy obecność innej na pierwszym planie. Niestety aplikacje, które zostały wyrzucone z pierwszego planu wciąż są otwarte w tle. Nie można także prawdziwie zablokować wyłączenia formularza wprowadzania hasła, zważywszy na konieczność wykorzystania ekrenu dotykowego do wpisania hasła oraz to, że nawet wyłączenie dotyku nie pomogłoby w urządzeniach posiadających fizyczne przyciski nawigacyjne. Martwiąca jest także możliwość automatycznego zamknięcia działających w tle usług przez system Android. Zostały podjęte wszystkie kroki by temu zapobiec, wykorzystując usługi pierwszoplanowe oraz WakeLock ale minimalne ryzyko wciąż pozostaje. Ponowne uruchomienie aplikacji po restarcie urządzenia działa, lecz nie jest wystarczająco szybkie. W przeprowadzonych testach aplikacja uruchamiała się po kilku minutach od włączenia, co pozostawia wiele możliwości sabotażu systemu.

6.2 Propozycje rozwoju

W przyszłości aplikację można rozwinąć na następujące sposoby:

- Wyłączyć łączność WiFi i dane komórkowe podczas aktywnej blokady.

- Wdrożyć prawdziwe wyłączanie chronionych aplikacji podczas próby ich uruchomienia w czasie blokady.
- Dodać wsparcie dla większej ilości urządzeń typu smartband.
- Stworzyć dokładniejsze statystyki aktywności.
- Upodobnić aplikację do typowej aplikacji fitness, aby zamaskować ją w systemie.
- Dodać automatyczne usuwanie starych danych o aktywności.
- Uzupełnić algorytm blokowania o wykrywanie zdarzeń dodanych w systemie Android 10.
- Usprawnić szybkość wymiany danych ze smartbandem.
- Przeorganizować bazę danych na prawdziwie relacyjny model.

Bibliografia

[1] F. Almenárez-Mendoza, L. Alonso, A. Marín-López, P. Cabarcos. Assessment of fitness tracker security: A case of study. *Proceedings*, 2:1235, 10 2018.

[2] Bluetooth SIG. *Bluetooth Core Specification Version 5.2*, 12 2019. Vol 3, Części F i G.

[3] Y. Cifuentes, L. Beltrán, L. Ramírez. Analysis of security vulnerabilities for mobile health applications. *International Journal of Health and Medical Engineering*, 9(9):1067 – 1072, 2015.

[4] J. Clement. Share of global mobile website traffic 2015-2021. https://www.statista.com/statistics/277125/share-of-website-traffic-coming-from-mobile-devices/.

[5] G. Combs. Wireshark. Uzyskano ze strony produktu https://www.wireshark.org/.

[6] E. De Cristofaro, H. Du, J. Freudiger, G. Norcie. Two-factor or not two-factor? a comparative usability study of two-factor authentication. *USEC*, 09 2013.

[7] Freedom of the Press Foundation, Guardian Project. Haven: Keep watch. Strona internetowa projektu https://guardianproject.github.io/haven/.

[8] Google. Android Management API. Uzyskano z poradnika w dokumentacji Android Management API https://developers.google.com/android/management/introduction.

[9] Google. Android Management API Policies. Uzyskano z dokumentacji Android Management API https://developers.google.com/android/management/reference/rest/v1/enterprises.policies.

[10] Google. Background Service Limitations. Uzyskano z opisu wersji Androida Oreo https://developer.android.com/about/versions/oreo/background.html#services.

[11] Google. Optimize for Doze and App Standby. Uzyskano z poradnika w dokumentacji Androida https://developer.android.com/training/monitoring-device-state/doze-standby.

[12] Google. Wake Lock. Uzyskano z dokumentacji Androida https://developer.android.com/reference/kotlin/android/os/PowerManager.WakeLock.

[13] S. O'Dea. Number of smartphone users worldwide from 2016 to 2026. https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/.

[14] J. Rutkowska. Evil Maid goes after TrueCrypt! Uzyskano z bloga autorki https://blog.invisiblethings.org/2009/10/15/evil-maid-goes-after-truecrypt.html.

[15] H. Shen, C. Faklaris, H. Jin, L. Dabbish, J. I. Hong. 'I Can't Even Buy Apples If I Don't Use Mobile Pay?': When Mobile Payments Become Infrastructural in China. *Proc. ACM Hum.-Comput. Interact.*, 4(CSCW2), Paz. 2020.

[16] A. Shimokawa, C. Pfeiffer, D. Gobbetti. Gadgetbridge. Uzyskano z repozytorium projektu https://codeberg.org/Freeyourgadget/Gadgetbridge.

[17] Yubico Inc. YubiKey 5 NFC. Uzyskano ze strony producenta 08.06.2021 https://www.yubico.com/pl/product/yubikey-5-nfc/.

[18] Yubico Inc. Yubikey for mobile. Uzyskano ze strony producenta https://resources.yubico.com/53ZDUYE6/as/q4bsus-2mej80-29grce/YubiKey˙for˙Mobile˙Solution˙Brief.pdf.

Zawartość płyty CD

Płyta CD dołączona do pracy składa się z następujących elementów:

1. Folder *latex* zawierający kody źródłowe części pisemnej pracy dyplomowej w języku LaTeX wraz z wykorzystywanymi grafikami.
2. Folder *lockband* zawierający kody źródłowe części implementacyjnej pracy dyplomowej w języku Kotlin wraz ze skompilowaną aplikacją mobilną na system Android w formacie APK.
3. Folder *pdf* kryjący w sobie egzemplarz pracy dyplomowej w postaci pliku PDF.