

Wstęp do systemów mobilnych

Aplikacja „Oddaj krew”

Dokumentacja projektowa

Niedziela, 13:30

Skład zespołu:

Dominik Cegiełka	224478	224478@edu.p.lodz.pl
------------------	--------	--

Kamil Zarych	224546	224546@edu.p.lodz.pl
--------------	--------	--

Spis treści

1. Wprowadzenie	3
2. Prezentacja chmury	4
a) Wybrane usługi do budowy aplikacji	4
b) Usługi do prowadzenia testów jakości, wykrywania błędów i publikowania aplikacji	5
c) Wybrane usługi do analizy danych, angażowania użytkowników i wysyłania dopasowanych przekazów marketingowych	6
3. Prezentacja techniczna aplikacji	7
a) Wymagania	7
b) Architektura	7
c) Warstwa logiczna	8
d) Rejestracja i logowanie	9
4. Aspekty bezpieczeństwa zaimplementowane w aplikacji	12
a) Hasła	12
b) Uprawnienia	12
c) Reguły dostępu do bazy danych Cloud Firestore	12
5. Opis rozwiązań chmurowych wykorzystanych do wdrożenia aplikacji	13
a) Firebase Authentication	13
b) Cloud Firestore	14
c) Cloud Functions	15
6. Podręcznik użytkownika	18
a) Uzyskiwanie dostępu do aplikacji	18
b) Rejestracja i logowanie	18
d) Menu	20
e) Ustawienia	21
7. Wnioski	21
8. Doświadczenie po wybranym środowisku	22
9. Bibliografia	23
10. Członkowie zespołu	23




1. Wprowadzenie

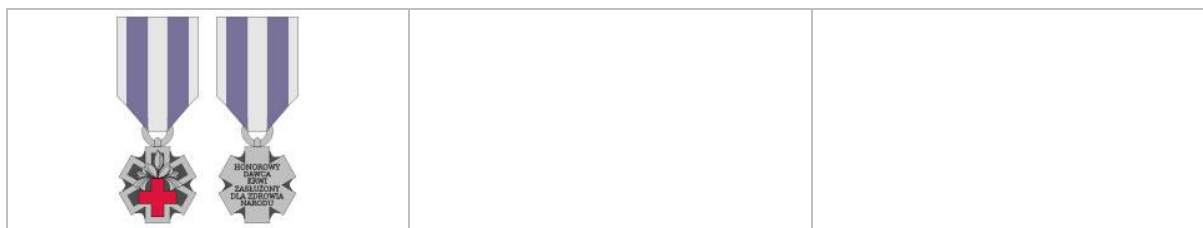
Aplikacja „oddaj krew” została stworzona do przechowywania informacji o krwiodawcach tj. ilości oddanej krwi, uzyskanych odznakach.

Przeznaczona będzie głównie dla dawców krwi oraz punktów poboru w celu dokumentacji ilości oddawanej krwi. Każdy dawca ma swój oddzielny profil, który można stworzyć po uruchomieniu aplikacji. W tym celu korzystamy z formularza rejestracji. Po pomyślnym zarejestrowaniu i potwierdzeniu konta na wskazanym adresie email mamy możliwość zalogowania się do aplikacji.

Po każdorazowym oddaniu krwi, pielęgniarka ma możliwość dodania informacji o pobraniu do profilu dawcy. W tym celu musi wybrać dawcę z listy, wprowadzić datę, rodzaj donacji (*krw pełna, płytki krwi, osocze*) oraz ilość pobranej krwi. Następnie użytkownik dostaje informacje w formie powiadomienia o nowym wpisie.

Na podstawie ilości oddanej krwi przyznawane są odznaki dla dawców, również w zależności od płci.

Nazwa odznaki	Ilość oddanej krwi w litrach	
	Kobiety	Mężczyźni
Zasłużony Honorowy Dawca Krwi 3 ^o 	5	6
Zasłużony Honorowy Dawca Krwi 2 ^o 	10	12
Zasłużony Honorowy Dawca Krwi 1 ^o 	15	18
„Honorowy Dawca Krwi - Zasłużony dla Zdrowia Narodu”	20	20



W aplikacji dostępna są również informacje o nadchodzących wydarzeniach zbiórki krwi. Gdy jesteśmy zwykłym użytkownikiem mamy tylko możliwość przeglądania nadchodzących wydarzeń. Natomiast konto typu „pielęgniarka” ma również możliwość dodawania wydarzeń o zbiórkach krwi. Po dodaniu wydarzenia do użytkowników aplikacji wysyłane jest powiadomienie.

2. Prezentacja chmury

W naszej aplikacji wykorzystaliśmy usługi chmurowe Google Firebase, które odpowiadają za architekturę backendową aplikacji. Firebase jest rozwiązaniem chmurowym typu BaaS (Backend as a Service) oferującym szeroką gamę usług poczynając od budowy aplikacji, czyli autentykacji, baz danych, przechowywania plików, hostingu poprzez testy i wykrywanie błędów aplikacji kończąc na narzędziach służących do analizy danych w celach marketingowych, przesyłania powiadomień czy obsługi reklam w aplikacji.

Usługi te są dostępne dla aplikacji mobilnych iOS i Android, część także dla aplikacji webowych oraz innych środowisk.

W bezpłatnym planie Spark, mamy dostęp do części rozwiązań z ograniczeniami. Natomiast aby skorzystać ze wszystkich funkcjonalności należy aktywować płatny plan Blaze. Koszty zależne są od bieżącego zużycia poszczególnych usług powyżej darmowego limitu. Szczegółowy cennik usług dostępny jest [tutaj](#).

a) Wybrane usługi do budowy aplikacji

Nazwa usługi	Obsługiwane platformy	Opis
Authentication	iOS, Android, Web, C++, Unity, Node.js, Java	Usługa weryfikacji użytkownika obejmująca różne metody uwierzytelniania, np. poprzez mail i hasło, konto Google, Facebook, Twitter itd.
Cloud Firestore	iOS, Android, Web, C++, Unity, Node.js, Java, Python, Go	Elastyczna i skalowalna baza danych NoSQL w chmurze umożliwiająca przechowywanie i synchronizację danych na żywo, obsługę offline oraz wysoce wydajną obsługę zapytań.
Realtime Database	iOS, Android, Web, C++, Unity	Baza danych o niskim opóźnieniu dedykowana produktom, które wymagają synchronizacji statusów pomiędzy klientami w czasie rzeczywistym.

Cloud Functions	iOS, Android, Web, C++, Unity	Bezserwerowe środowisko wykonawcze (execution environment) służące do tworzenia i łączenia usług w chmurze za pomocą prostych funkcji wyzwalanych przez usługi Firebase i żądania HTTPS.
Cloud Storage	iOS, Android, Web, C++, Unity	Usługa bezpiecznego przechowywania na dużą skalę danych i plików takich jak treści generowane przez użytkowników, obrazy, nagrania audio czy wideo.
Hosting	Web	Zestaw narzędzi hostingowych, obejmujący m.in. CDN (Content Delivery Network, usługę zapewniającą wysoką dostępność i wydajność) czy bezpłatny certyfikat bezpieczeństwa SSL.

b) Usługi do prowadzenia testów jakości, wykrywania błędów i publikowania aplikacji

Nazwa usługi	Obsługiwane platformy	Opis
Crashlytics	iOS, Android, Unity	Narzędzie służące do raportowania awarii, które wykrywa i śledzi problemy występujące w aplikacji oraz m.in. reakcje użytkowników na pojawiające się błędy. Crashlytics pozwala również ustawiać powiadomienia o wystąpieniu błędów.
Test Lab	iOS, Android	Test Lab to oparta na chmurze infrastruktura do testowania aplikacji. Za pomocą jednej operacji możesz przetestować aplikację na Androida lub iOS na wielu różnych urządzeniach i konfiguracjach, a wyniki - w tym dzienniki, filmy i zrzuty ekranu - zobaczyć w konsoli Firebase.
Performance Monitoring	iOS, Android, Web	Performance Monitoring to usługa, która pomaga uzyskać wgląd w charakterystykę wydajności aplikacji na iOS, Androida oraz aplikacji webowych.
App Distribution	iOS, Android	Usługa przesyłania przedpremierowej wersji aplikacji do grupy zaufanych testerów. Dzięki szybkiemu umieszczaniu aplikacji na urządzeniach testerów można szybko i często otrzymywać informacje zwrotne z urządzeń.

c) Wybrane usługi do analizy danych, angażowania użytkowników i wysyłania dopasowanych przekazów marketingowych

Nazwa usługi	Obsługiwane platformy	Opis
Analytics	iOS, Android, Web, C++, Unity	Narzędzie służące do analizy danych o zachowaniu użytkowników, przedstawiające informacje za pomocą czytelnych dashboardów i wykresów. Pozwala prowadzić zaawansowaną, rozbudowaną analitykę wielu czynników dzięki możliwości eksportu danych do usługi GCP BigQuery.
Predictions	iOS, Android, Web, C++, Unity	Usługa prognoz, która, dzieląc na grupy użytkowników, pomaga przewidzieć poziom konwersji lub porzuceń aplikacji. Wykorzystuje inteligentne systemy samouczące się i wspiera akcje wykonywane przez inne usługi np. personalizację treści lub częstotliwość wysyłanych wiadomości w In-App Messaging.
Cloud Messaging (FCM)	iOS, Android, Web, C++, Unity	Usługa umożliwiająca przesyłanie bezpłatnych wiadomości do grup lub pojedynczych użytkowników korzystających z różnych platform. Korzystając z FCM, można powiadomić aplikację kliencką, że nowa pocztą e-mail lub inne dane są dostępne do synchronizacji. Można również wysyłać powiadomienia, aby zwiększyć zaangażowanie i utrzymać użytkowników aplikacji.
In-App Messaging	iOS, Android	Usługa angażująca aktywnych użytkowników aplikacji za pomocą komunikatów kontekstowych. Przesyłając wiadomości w aplikacji możemy zaangażować aktywnych użytkowników aplikacji, wysyłając im ukierunkowane, kontekstowe wiadomości, które zachęcają ich do korzystania z kluczowych funkcji aplikacji. Na przykład możesz wysłać wiadomość w aplikacji, aby zachęcić użytkowników do zasubskrybowania, obejrzenia filmu, ukończenia poziomu lub zakupu przedmiotu.

3. Prezentacja techniczna aplikacji

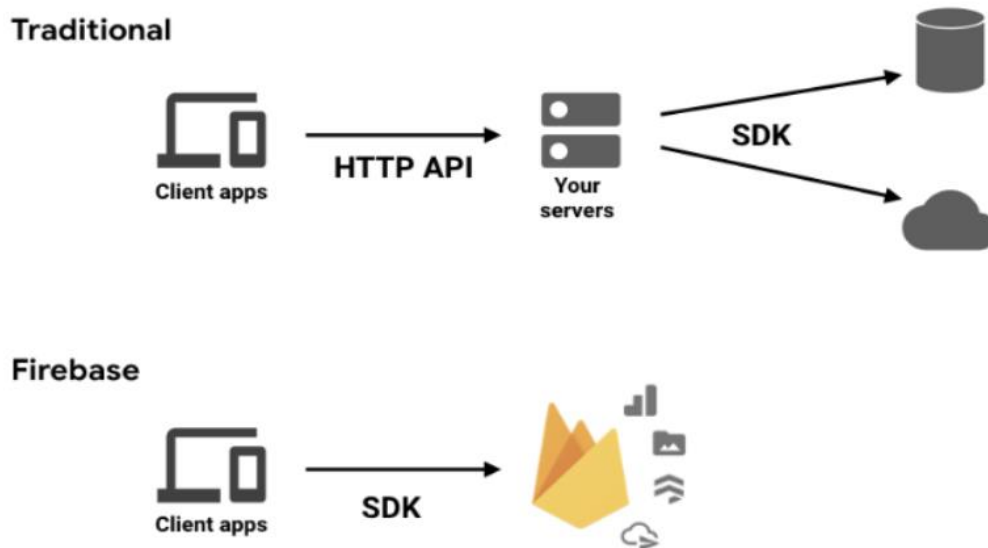
Aplikacja Oddaj krew jest aplikacją mobilną na platformę Android. Została stworzona w Flutterze, który jest rozwiązaniem zaproponowanym przez Google, umożliwiającym tworzenie natywnych aplikacji na iOS i Androida z jednej bazy kodu źródłowego. Aplikacje mobilne implementowane są w języku programowania Dart, wyglądają i zachowują się praktycznie identycznie na obydwu systemach. Ekosystem Fluttera jest dość zaawansowany i rozbudowany oferując wiele różnego rodzaju wtyczek, przez co czas potrzebny na stworzenie aplikacji mobilnej ulega skróceniu.

a) Wymagania

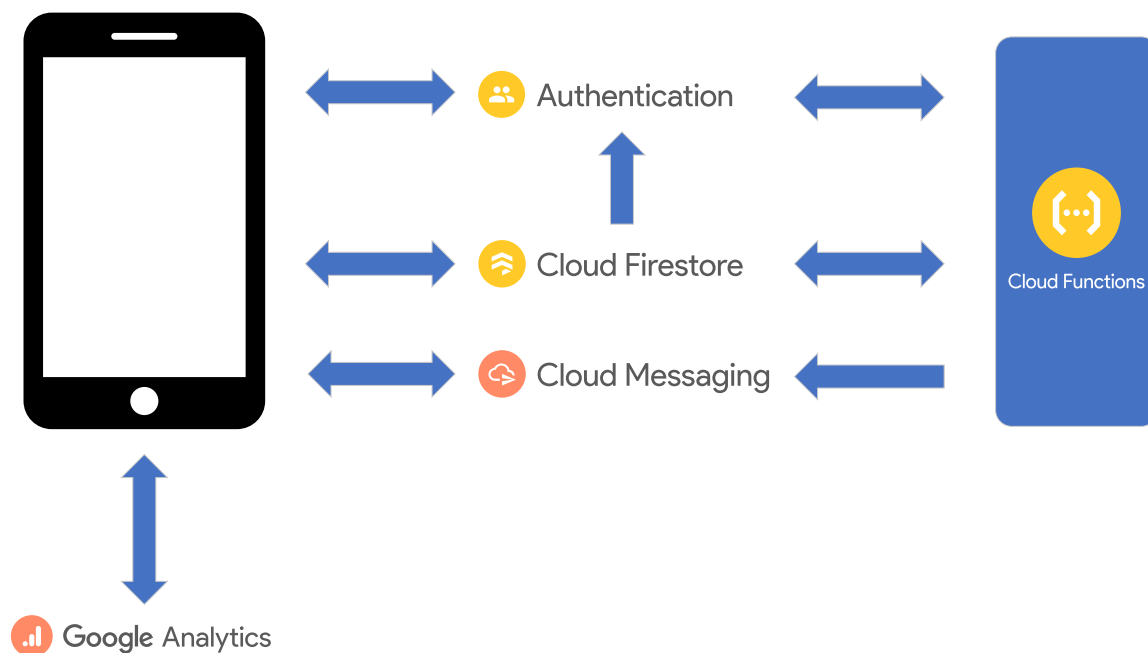
Minimalne wymagania:

Android 5.0, pamięć 1GB, połączenie internetowe

b) Architektura

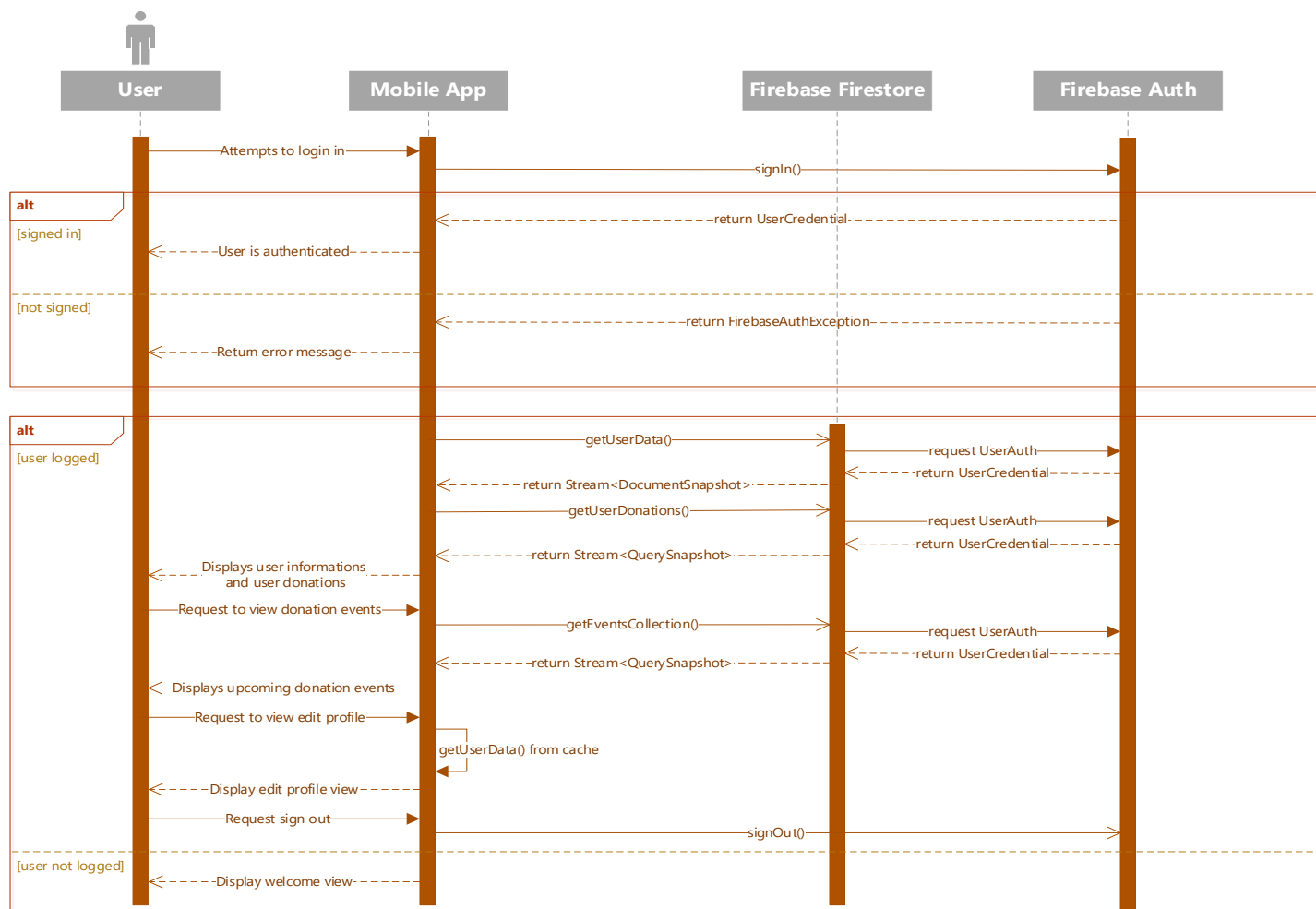


W tradycyjnych aplikacjach klienckich komunikacja odbywa się przy użyciu REST API, które udostępnia serwer aplikacji. W przypadku Firebase komunikacja odbywa się poprzez wykorzystanie Firebase SDK dla konkretnej platformy. W naszym przypadku wykorzystaliśmy zestaw wtyczek FlutterFire, który korzysta z Firebase SDK dla Androida. Tym samym backendem naszej aplikacji jest Firebase, z którym komunikacja odbywa się przy użyciu udostępnionego zestawu narzędzi SDK.



Rys. 1. Diagram architektury systemu

c) Warstwa logiczna



Rys. 2. Diagram komunikacji aplikacji z usługami chmury Firebase

d) Rejestracja i logowanie

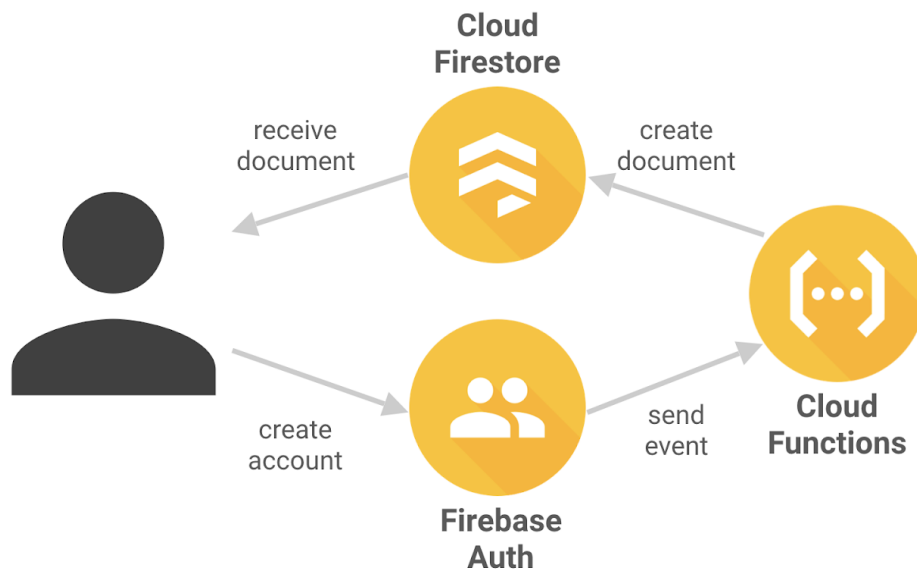
W naszej aplikacji rejestracja i logowanie odbywa się przy użyciu emaila oraz hasła poprzez Firebase Authentication SDK. W procesie logowania i rejestracji przez SDK zwracany jest **UID** użytkownika, który jest warunkiem uzyskania dostępu do danych z bazy danych Firestore oraz przejścia do [strony głównej](#) w aplikacji (w przypadku logowania).

W celu przetworzenia i przekazania komunikatów wyjątków (`FirebaseAuthException`) otrzymanych od usługi autentykacji w przypadku poszczególnych czynności przygotowaliśmy własną klasę **Auth**. Wybrane metody zostały przedstawione poniżej.

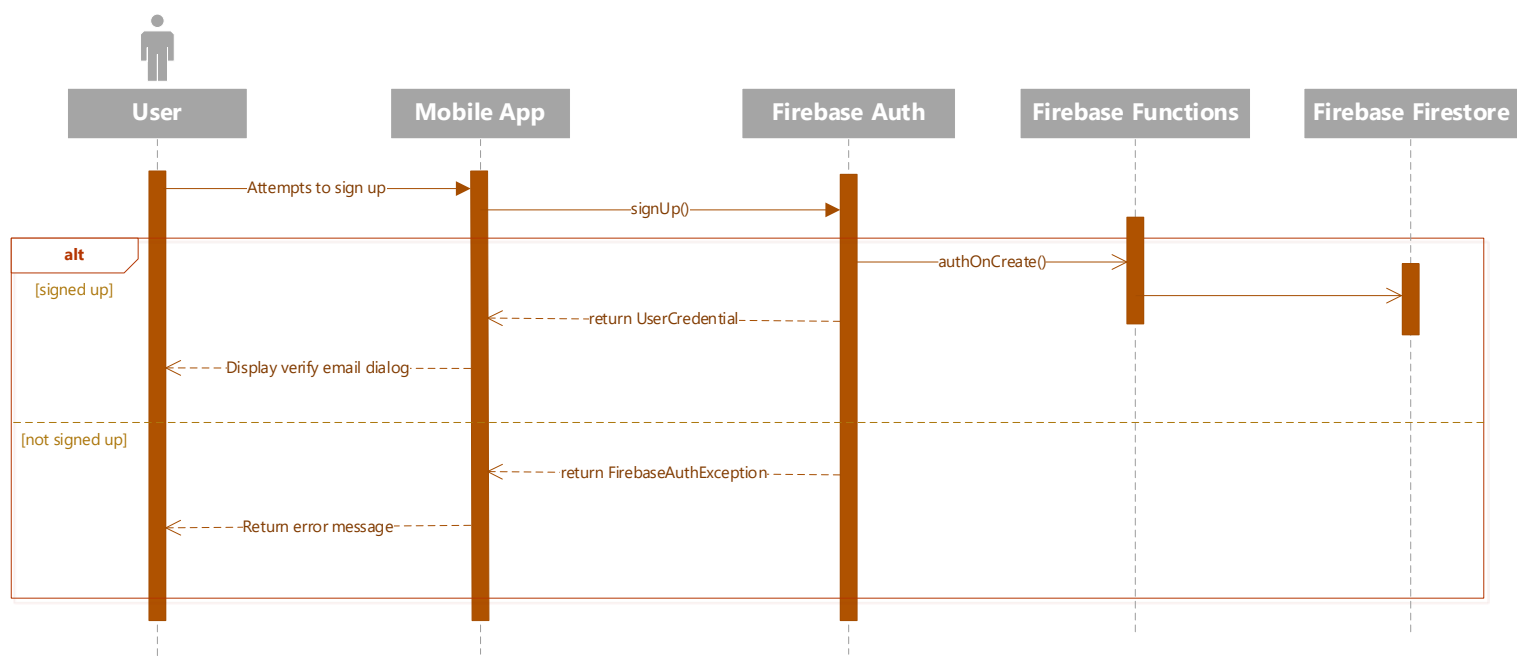
Rejestracja użytkownika:

```
@Override
Future<String> signUp(String email, String password) async {
  UserCredential result;
  try {
    result = await _firebaseAuth.createUserWithEmailAndPassword(
      email: email, password: password);
  } on FirebaseAuthException catch (e) {
    switch (e.code) {
      case 'weak-password':
        return S.current.weakPassword;
        break;
      case 'email-already-user':
        return S.current.emailAlreadyUser;
        break;
      case 'invalid-email':
        return S.current.invalidEmail;
        break;
      case 'operation-not-allowed':
        return S.current.operationNotAllowed;
        break;
      default:
        return S.current.loginError(e.message);
    }
  }
  return result.user.uid;
}
```

Podczas tworzenia konta użytkownika, Firebase Functions tworzy pusty dokument z danymi użytkownika. Identyfikatorem tego dokumentu jest UID użytkownika otrzymany z Firebase Auth.



Rys. 3. Diagram przepływu danych podczas tworzenia konta użytkownika



Rys. 4. Diagram sekwencji - rejestracja użytkownika

Logowanie użytkownika:

```

@Override
Future<String> signIn(String email, String password) async {
    UserCredential result;
    try {
        result = await _firebaseAuth.signInWithEmailAndPassword(
            email: email, password: password);
    } on FirebaseAuthException catch (e) {
        switch (e.code) {

```

```

        case 'user-not-found':
            return S.current.userNotFound;
            break;
        case 'wrong-password':
            return S.current.wrongPassword;
            break;
        case 'user-disabled':
            return S.current.wrongPassword;
            break;
        case 'invalid-email':
            return S.current.invalidEmail;
            break;
        default:
            return S.current.loginError(e.message);
    }
}
return result.user.uid;
}

```

Pobieranie i synchronizacja danych z bazą danych Firestore

Po zalogowaniu się aplikacja pobiera i synchronizuje dane w czasie rzeczywistym z bazy Firestore przy użyciu strumieni (Stream). W naszej aplikacji klasa Repository zawiera zapytania i strumienie które wykorzystywane są do komunikacji bazy Firestore z widżetami. W przypadku kolekcji dokumentów metody zwracają zapytania (Query), które następnie w zależności od potrzeby dodatkowo filtrujemy lub/i sortujemy przed wyświetleniem.

```

Stream<DocumentSnapshot> getUserData() {
    return _firestore
        .collection('users')
        .doc(Auth().getCurrentUser().uid)
        .snapshots();
}

Query getUsers() {
    return _firestore.collection('users');
}

Query getUserDonations() {
    return _firestore.collection('donations').where('userId',
        isEqualTo: _firestore.doc('/users/' + Auth().getCurrentUser().uid));
}

Query getNurseCollections() {
    return _firestore.collection('donations').where('nurseId',
        isEqualTo: _firestore.doc('/users/' + Auth().getCurrentUser().uid));
}

```

4. Aspekty bezpieczeństwa zaimplementowane w aplikacji

a) Hasła

Aplikacja „Oddaj krew” wymaga od użytkowników korzystania z hasła o długości minimum 8 znaków. Przy rejestracji sprawdzana jest długość hasła, gdy zawiera zbyt mało znaków, proces rejestracji kończy się pojawieniem komunikatu.

b) Uprawnienia

Każdy użytkownik posiada własne hasło oraz przydzielone uprawnienie. Uprawnienie są przydzielone na podstawie typu konta. Gdy osoba posiadająca konto jest pielęgniarką, ma w aplikacji możliwość dodania donacji oraz wydarzenia o organizowanej zbiórce. Takich uprawnień nie posiada zwykły użytkownik, który jest tylko dawcą krwi.

c) Reguły dostępu do bazy danych Cloud Firestore

Przy użyciu reguł bezpieczeństwa Cloud Firestore zdefiniowaliśmy dostęp dla użytkowników do danych poprzez sprawdzenie czy użytkownik pobierający dokument posiada odpowiednie uprawnienia do odczytu lub zapisu dokumentu.

Uprawnienia dawcy krwi:

- Odczytywanie i edycja swoich danych personalnych
- Odczytywanie swoich donacji
- Odczytywanie informacji na temat nadchodzących wydarzeń

Uprawnienia pielęgniarki:

- Odczytywanie i edycja swoich danych personalnych
- Odczytywanie danych personalnych wszystkich użytkowników
- Odczytywanie i dodawanie donacji
- Odczytywanie i dodawanie wydarzeń

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {

    match /{document=**} {
      allow read,write: if false;
    }

    function isSignedIn() {
      return request.auth.uid != null
    }

    function isNurse() {
      return
get(/databases/{database}/documents/users/{request.auth.uid}).data.isNurse
    }
  }
}
```

```

function checkDonateAccess(rsc) {
  return rsc.data.userId ==
  /databases/$(database)/documents/users/$(request.auth.uid)
}

match /users/{user} {
  allow read: if isNurse()
  allow read, write: if isSignedIn() && request.auth.uid ==
resource.id
}

match /events/{event} {
  allow read: if isSignedIn()
  allow write: if isSignedIn() && isNurse()
}

match /donations/{donation} {
  allow read: if (isSignedIn() && checkDonateAccess(resource))
  allow read,write: if isSignedIn() && isNurse()
}
}
}

```

5. Opis rozwiązań chmurowych wykorzystanych do wdrożenia aplikacji

Do wykorzystania Firebase w naszej aplikacji mobilnej wykorzystaliśmy zestaw wtyczek FlutterFire, który umożliwia połączenie aplikacji Flutter z usługami Firebase.

a) Firebase Authentication

Usługa backendu umożliwiająca bezpieczne uwierzytelnienie użytkowników aplikacji oraz zarządzanie nimi poprzez łatwe w użyciu pakiety SDK i gotowe biblioteki interfejsu użytkownika. Obsługuje autoryzację za pomocą emaila i hasła, numeru telefonu, popularnych dostawców tożsamości federacyjnej tj. Google, Facebook, Twitter, Github itd. Ponadto Firebase Authentication ściśle integruje się z innymi usługami Firebase i wykorzystuje standardy branżowe OAuth 2.0 i OpenID Connect, dzięki czemu można je łatwo zintegrować z niestandardowym backendem aplikacji.

W naszej aplikacji wykorzystaliśmy obsługę autoryzacji przy pomocy emaila i hasła. Autoryzacja odbywa się poprzez funkcje interfejsu użytkownika zaimplementowane w aplikacji umożliwiając rejestrację, logowanie, przypomnienie oraz zmianę hasła oraz zmianę adresu email. Interfejs ten połączyliśmy z Firebase Authentication SDK poprzez wtyczkę firebase_auth z zestawu FlutterFire.

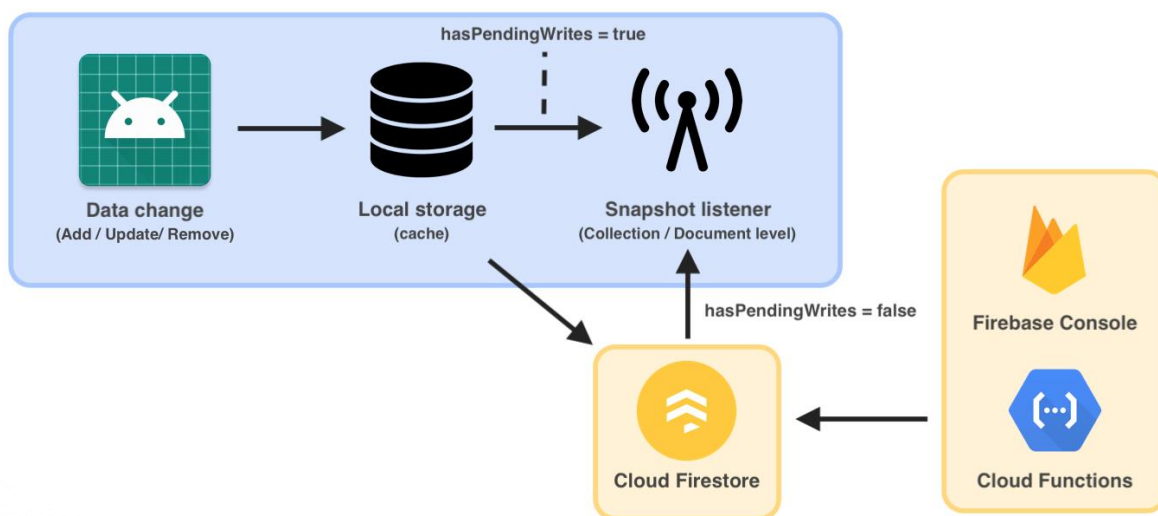
Logowanie użytkownika do aplikacji odbywa się przy użyciu poświadczeń uzyskanych w widoku logowania, które przekazywane są do pakietu SDK Firebase Authentication. Następnie usługi backendu Firebase weryfikują otrzymane poświadczenia i zwracają odpowiedź do klienta. Po pomyślnym zalogowaniu uzyskujemy dostęp do podstawowych informacji o profilu użytkownika i kontrolujemy dostęp użytkownika do danych

przechowywanych w innych produktach Firebase. W przypadku nieprawidłowych danych logowania lub innego błędu SDK zwraca odpowiedni komunikat do widoku logowania.

b) Cloud Firestore

Cloud Firestore to chmurowa baza danych NoSQL, do której iOS, Android i aplikacje internetowe mogą mieć bezpośredni dostęp za pośrednictwem natywnych zestawów SDK. Zgodnie z modelem danych NoSQL Cloud Firestore, dane są przechowywane w dokumentach, które zawierają pola mapowane do wartości. Dokumenty te są przechowywane w kolekcjach, które są kontenerami na dokumenty, które można wykorzystać do porządkowania danych i budowania zapytań. Dokumenty obsługują wiele różnych typów danych, od prostych łańcuchów znakowych i liczb, po złożone, zagnieżdżone obiekty. Można również tworzyć podzbiory w dokumentach i budować hierarchiczne struktury danych, które skalują się w miarę rozwoju bazy danych. Model danych Cloud Firestore obsługuje każdą strukturę danych, która najlepiej pasuje do danej aplikacji.

Dodatkowo, zapytanie w Cloud Firestore jest wyraziste, skuteczne i elastyczne. Nie ma potrzeby pobierania całej kolekcji lub zagnieżdżonych podkolekcji. Można swobodnie dodawać sortowanie, filtrowanie i ograniczenia do zapytań i kursorów, aby paginować wyniki.



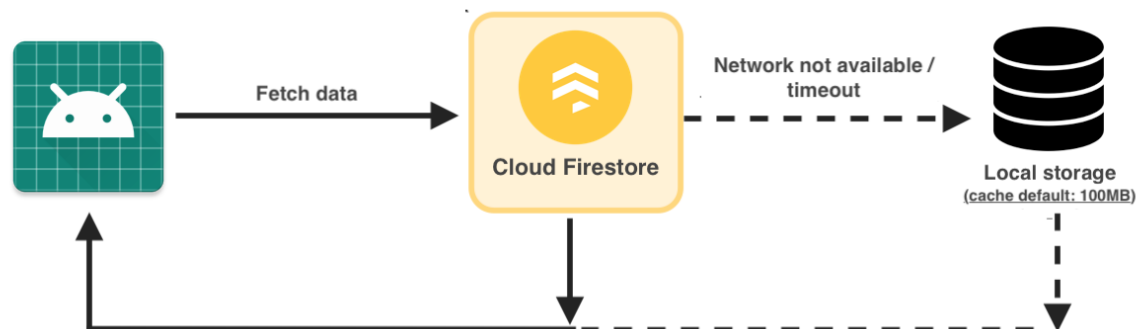
Aby utrzymać dane w aplikacji na bieżąco, bez konieczności pobierania całej bazy danych za każdym razem i generowania wysokich kosztów za usługę, gdy następuje aktualizacja danych w bazie, obserwatorzy „listeners” w czasie rzeczywistym aktualizują dane w pamięci podręcznej. Natomiast jeśli dane zostaną zmienione przez użytkownika, zapisywane są one w pamięci podręcznej i następnie przesyłane do bazy Firestore.

Ścieżka wdrożenia:

1	Integracja zestawów SDK Cloud Firestore	Szybkie dołączenie klientów za pomocą Gradle, CocoaPods lub skryptu.
---	---	--

2	Zabezpiecz swoje dane	Użyj Cloud Firestore Security Rules lub Identity and Access Management (IAM), aby zabezpieczyć swoje dane odpowiednio na potrzeby rozwoju urządzeń mobilnych/stron internetowych i serwerów.
3	Dodaj dane	Twórz dokumenty i kolekcje w swojej bazie danych.
4	Pobierz dane	Twórz zapytania lub używaj listenerów w czasie rzeczywistym do pobierania danych z bazy danych.

System pamięci podręcznej:



Cloud Firestore SDK pobiera dane z serwera Cloud Firestore, jeśli jest to możliwe. Jednak z powodu utraty połączenia z siecią pobieranie danych może się nie powieść. Aby rozwiązać ten problem, Cloud Firestore SDK domyślnie przechowuje wszystkie pobrane dokumenty w lokalnej bazie danych i zwraca dane z pamięci podręcznej jako alternatywę w przypadku problemów z siecią. Rozmiar pamięci podręcznej, który można dostosować, wynosi domyślnie 100 MB.

c) Cloud Functions

Google Cloud Functions to bezserwerowe środowisko do budowania i podłączania usług w chmurze. Dzięki temu mamy możliwość pisania prostych, jednozadaniowych funkcji, które są dołączane do zdarzeń emitowanych z infrastruktury i usług w chmurze. Kod wykonywany jest w pełni zarządzanym środowisku. Nie trzeba udostępniać żadnej infrastruktury ani martwić się o zarządzanie jakimikolwiek serwerami.

Funkcje chmury mogą być napisane przy użyciu JavaScript, Python 3, Go lub Java runtimes na platformie Google Cloud.

d) Firebase Cloud Messaging (FCM)

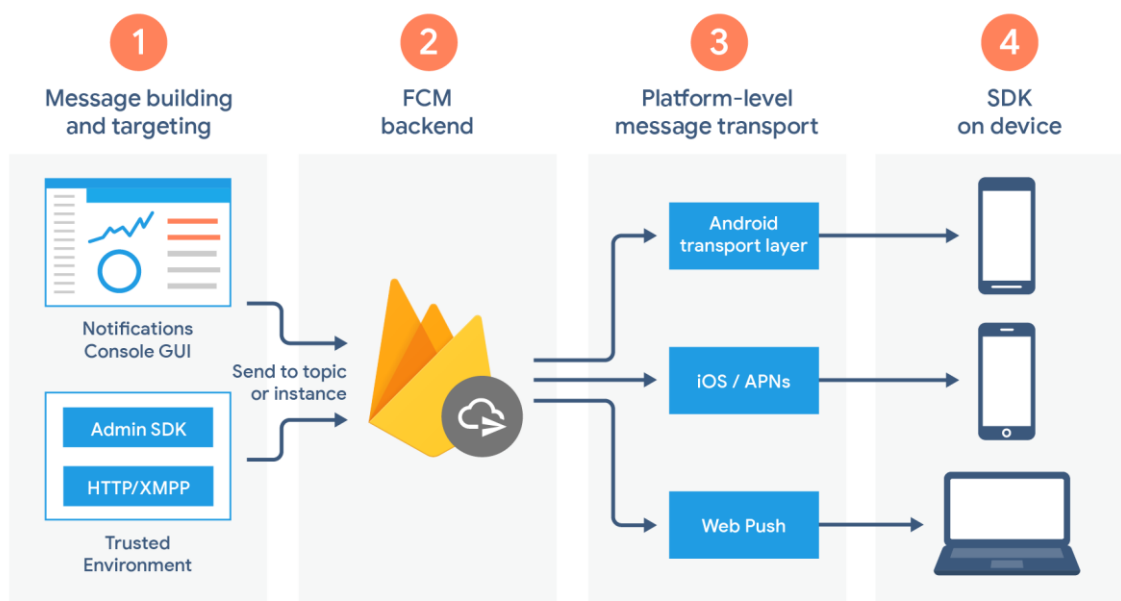
Firebase Cloud Messaging (FCM) to wieloplatformowe rozwiązanie do obsługi wiadomości, które umożliwia niezawodne wysyłanie wiadomości bez żadnych kosztów.

Korzystając z FCM, można powiadomić aplikację kliencką, że nowa poczta e-mail lub inne dane są dostępne do synchronizacji. Można wysyłać powiadomienia, aby zwiększyć ponowne zaangażowanie i utrzymanie użytkowników. W przypadkach użycia, takich jak wiadomości błyskawiczne, wiadomość może przesyłać dane o wielkości do 4 KB do aplikacji klienckiej.

Aby otrzymywać wiadomości z FCM, urządzenie klienckie rejestruje się w celu odbierania komunikatów, poprzez uzyskanie tokenu rejestracji, który jednoznacznie identyfikuje instancję aplikacji.

FCM opiera się na następującym zestawie komponentów, które tworzą, transportują i odbierają komunikaty:

1. Narzędzia do tworzenia lub budowania żądań wiadomości. Kompozytor powiadomień zapewnia opartą na graficznym interfejsie użytkownika opcję tworzenia żądań powiadomień. Aby uzyskać pełną automatyzację i obsługę wszystkich typów wiadomości, trzeba stworzyć żądania wiadomości w środowisku zaufanego serwera, które obsługuje Firebase Admin SDK lub protokoły serwera FCM. Takim środowiskiem może być Cloud Functions dla Firebase, Google App Engine lub własny serwer aplikacji.

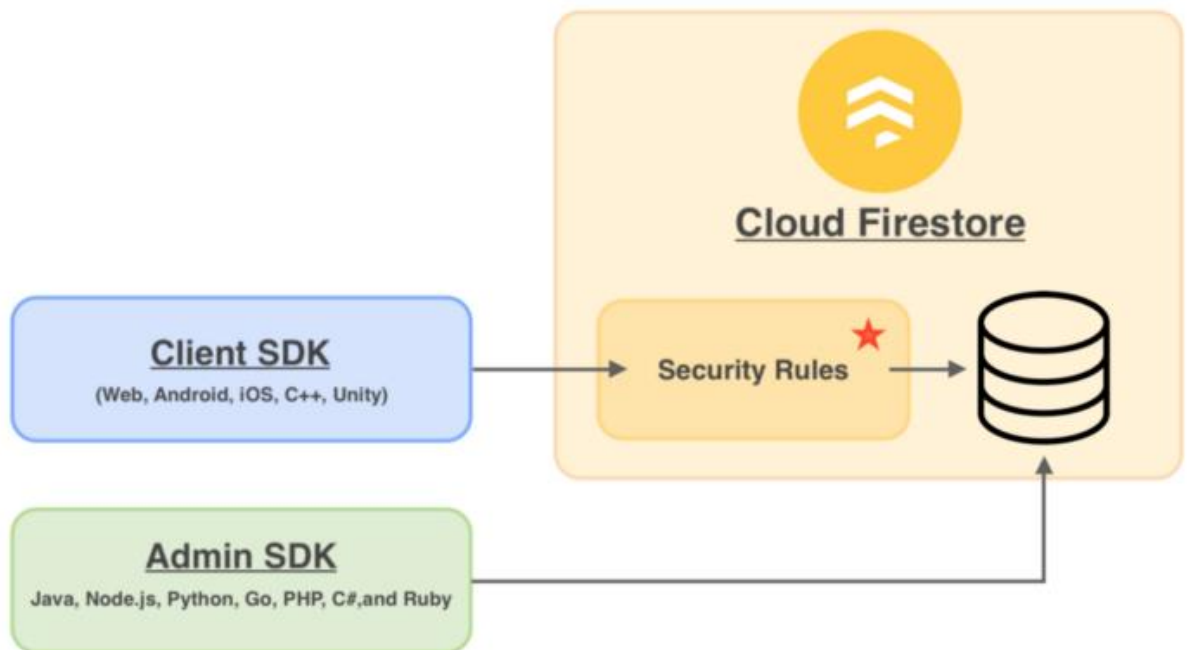


2. Backend FCM, który (między innymi) przyjmuje żądania wiadomości, wykonuje fanout¹ wiadomości za pośrednictwem tematów i generuje metadane wiadomości, tj. identyfikator wiadomości.

¹ Fanout - proces wysyłania wiadomości do wielu urządzeń

3. Warstwa transportowa na poziomie platformy, która kieruje wiadomości do docelowego urządzenia, obsługuje dostarczanie wiadomości i w stosownych przypadkach stosuje konfigurację specyficzną dla platformy. Ta warstwa transportowa obejmuje:
 - Warstwa transportowa systemu Android (ATL) dla urządzeń z systemem Android z usługami Google Play
 - Usługa Apple Push Notification (APN) dla urządzeń z systemem iOS
 - Protokół web push dla aplikacji internetowych
4. FCM SDK na urządzeniu użytkownika, w którym wyświetlane jest powiadomienie lub komunikat jest obsługiwany zgodnie ze stanem pierwszego planu / tła aplikacji i odpowiednią logiką aplikacji.

Opis rozwiązań chmurowych bezpieczeństwa, usług używanych w aplikacji



Firebase udostępnia dwa typy pakietu Cloud Firestore SDK:

- **Client SDK**

Jest przeznaczony dla programistów frontendu w celu uzyskania dostępu do bazy danych i wykonywania akcji CRUD. Wszystkie operacje ograniczone są regułami bezpieczeństwa. Administrator projektu Firebase może ustawić różne reguły, aby zezwolić tylko niektórym użytkownikom (zalogowanym za pośrednictwem uwierzytelniania Firebase) na dostęp do bazy danych, a nawet zabronić usuwania dokumentów w określonej kolekcji.

- Admin SDK

Jest przeznaczony dla programistów backendu w celu uzyskania dostępu do bazy danych Cloud Firestore za pośrednictwem Cloud Functions lub z własnego serwera. Akcje CRUD w tym przypadku nie są sprawdzane przez reguły bezpieczeństwa.

6. Podręcznik użytkownika

a) Uzyskiwanie dostępu do aplikacji

Dostęp do aplikacji realizowany jest za pośrednictwem pliku instalacyjnego .apk lub Sklepu Play.

b) Rejestracja i logowanie

Podczas uruchomienia aplikacji użytkownikowi zostanie wyświetlony ekran **powitalny**. Użytkownik ma do wyboru zalogowanie oraz rejestrację.



Przy pierwszym użyciu musi zarejestrować się do naszej aplikacji naciskając przycisk „zarejestruj się”, a następnie wypełnić wymagane dane w formularzu (adres email oraz hasło). Następnie walidowany jest adres email oraz sprawdzana jest poprawność hasła (czy spełniona jest minimalna długość hasła, oraz czy wprowadzone hasła są jednakowe).

19:53

ZAREJESTRUJ SIĘ

Twój e-mail

Hasło

Potwierdź hasło

ZAREJESTRUJ SIĘ

Masz już konto? Zaloguj się

LUB

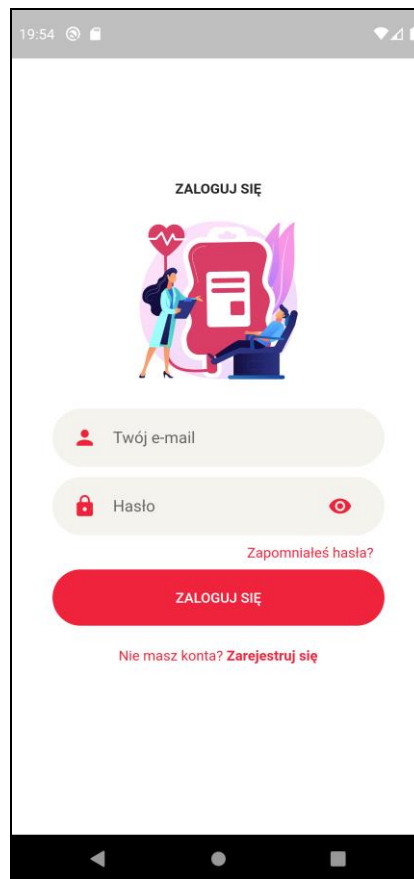
f

t

G+

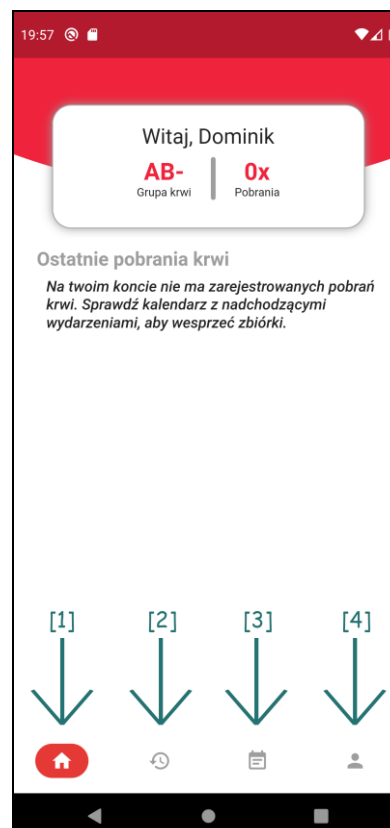
Po wyborze przycisku „Zarejestruj się” użytkownikowi na wskazany w formularzu adres mailowy zostanie przesłany email z potwierdzeniem założenia konta w aplikacji „oddaj krew”.

Po rejestracji konta użytkownik będzie miał możliwość zalogowania.



d) Menu

Po zalogowaniu, na dole ekranu domyślnie wyświetlane jest **menu** aplikacji.



W ramach menu dostępne są następujące sekcje:

- **Strona główna [1]** – widok po zalogowaniu do aplikacji. Wyświetlane są informacje o użytkowniku.
- **Historia donacji lub pobrań [2]** – wyświetlana jest historia donacji użytkownika, lub w przypadku pielęgniarki historia pobrań krwi. W tym miejscu pielęgniarka ma również przycisk, za pomocą którego może dodać nową donację.
- **Nadchodzące wydarzenia [3]** – wyświetlane są informacje o nadchodzących zbiórkach krwi. Pielęgniarka ma możliwość dodania nowego wydarzenia.
- **Profil użytkownika [4]** – po wybraniu tej opcji użytkownikowi zostanie wyświetlony ekran „Profil użytkownika” zawierający dane, które możemy edytować tj.:
 - imię i nazwisko,
 - płeć,
 - telefon kontaktowy,
 - data urodzenia,
 - grupa krwi.

e) Ustawienia

Ustawienia w Profilu użytkownika zawierają następujące sekcje:

- Ustawienia konta
- Powiadomienia
- Przycisk „Wyloguj się”

Zakładka **ustawienia konta** umożliwia:

- Zmianę adresu email,
- Zmianę hasła,
- Zmianę języka aplikacji (dostępny język polski oraz angielski),

Zakładka **powiadomienia** umożliwia:

- włączenie/wyłączenie powiadomień dotyczących zbiórki krwi
- włączenie/wyłączenie powiadomień o aktywności na koncie (pojawienie się nowej donacji).

Wybór przycisku „**Wyloguj się**” spowoduje wylogowanie użytkownika z aplikacji oraz wyświetlenie ekranu **powitalnego**.

7. Wnioski

Aplikacja „Oddaj krew” została stworzona do kontrolowania swojej aktywności w obszarze oddawania krwi, oraz po to, by usprawnić proces rejestracji pobrań przez pielęgniarki.

Każdy użytkownik aplikacji posiada własne konto, odpowiednio zabezpieczone hasłem. Zaimplementowaliśmy autoryzację użytkownika przy pomocy Firebase Authentication, wykorzystując adres email oraz hasło. Logowanie odbywa się przy pomocy poświadczeń pozyskanych z widoku logowania.

Do przechowywania informacji dotyczących użytkowników, donacji oraz wydarzeń wykorzystaliśmy chmurową bazę danych Cloud Firestore. Utworzone zostały trzy kolekcje:

- **donations** – przechowuje informacje dotyczące donacji (ilość oddanej krwi, datę, typ, referencję do danych(dokumentu) pielęgniarki oraz dawcy),
- **events** – przechowuje informację dotyczące wydarzeń (data, typ donacji, rodzaj wydarzenia *'normalna lub pilna zbiórka'*, lokalizacja),
- **users** – przechowuje informacje dotyczące użytkowników (grupa krwi, data urodzenia, imię i nazwisko, płeć, numer telefonu, rodzaj konta *'donor lub pielęgniarka'*).

W naszej aplikacji zostały również zaimplementowane powiadomienia użytkownika z wykorzystaniem Cloud Functions. Użytkownik, któremu zostanie dodana informacja odnośnie donacji otrzymuje powiadomienie jeśli ma aktywne powiadomienia dot. aktywności na koncie. W przypadku dodania nowego wydarzenia, powiadomienie wysyłane jest do użytkowników mających aktywne otrzymywanie powiadomień o zbiórkach krwi,

8. Doświadczenia po wybranym środowisku

Flutter - programy stworzone z wykorzystaniem tego frameworka bazują na języku programowania Dart. Flutter odróżnia się od konkurencji tym, że aplikacje są faktycznie natywne. Część kodu można na przykład napisać w Kotlinie czy Swiftcie.

Jedną z głównych zalet tego środowiska jest możliwość jednoczesnego tworzenia aplikacji na systemy Android i iOS, co znacznie skraca czas implementacji (**piszemy tylko jeden kod**). Dzięki wykorzystaniu własnego silnika renderowania rozłożenie elementów interfejsu jest takie samo na obu platformach, a wygląd kontrolek natywny dla danego środowiska.

Kolejną zaletą jest prostota budowania interfejsu użytkownika, który tworzy się z poziomu kodu. Wszelkie zmiany w kodzie można niemal natychmiastowo podejrzeć w aplikacji uruchomionej na smartfonie, ponieważ Flutter oferuje tzw. „Hot reload”, czyli natychmiastowe przeładowanie treści wyświetlanych na urządzeniu. Ta możliwość znacznie przyspiesza proces tworzenia i testowania oprogramowania.

Flutter ma dużo zalet, ale istnieją również wady. Framework swoją premierę miał w końcówce 2018 roku, dlatego jest to młoda technologia, która ciągle się rozwija. Posiada mnóstwo pluginów, jednak niektóre zawierają błędy, dlatego trzeba je sprawdzić przed dodaniem do projektu. Ponadto wiele ważnych funkcjonalności udostępnionych jest wydaniu beta Fluttera, które wykorzystaliśmy do stworzenia aplikacji powodując częstą potrzebę refaktoryzacji kodu i użycia nowych rozwiązań, gdyż użyte w kodzie są przestarzałe i niezalecane. Dotyczy to również pluginów, które są zgodne z danymi

wydaniami i powstają konflikty przy budowie aplikacji. Do tworzenia aplikacji potrzebna jest również znajomość języka Dart.

Podsumowując, Flutter to bez wątpienia ogromny krok w kierunku tworzenia aplikacji wieloplatformowych. Technologia ta nie jest wolna od wad oraz nie nadaje się do wszystkich zastosowań, jednak może stanowić konkurencję do rozwiązań takich jak chociażby Xamarin.

9. Bibliografia

- <https://rckik.krakow.pl/przywileje-dawcow-2/>
- <https://firebase.flutter.dev/docs/overview>
- <https://firebase.google.com/docs>
- <https://www.webchefs.pl/blog/flutter-wszystko-co-powinienes-wiedziec/>
- <https://flyonthecloud.com/pl/blog/gry-mobilne-firebase-gcp/>
- <https://medium.com/firebase-developers/patterns-for-security-with-firebase-offload-client-work-to-cloud-functions-7c420710f07>
- <https://myrickchow.medium.com/firebase-cloud-firestore-fetching-data-76619dfd3bff> (schematy cache)
- <https://medium.com/@viveky259259/add-firebase-to-flutter-6bc9e2755284>
- <https://firebase.googleblog.com/2017/08/hamilton-app-takes-stage.html>

10. Członkowie zespołu

Dominik Cegiełka, Kamil Zarych.

Nad projektem pracowaliśmy równomiernie, rozkład pracy w skali całego projektu 50:50.