Pomocnik kierowcy

1. Cel i opis projektu

Aplikacja mobilna służąca kierowcom którzy są w trudnej sytuacji, wymagającej natychmiastowej pomocy. Aplikacja będzie pobierać lokalizacje innych użytkowników, którzy znajdują sie w pobliżu oraz ich aktualne wyposażenie jakim dysponują do pomocy.

Aplikacja ma na celu pomoc innym kierowcom, którzy potrzebują pomocy np. przy wymianie koła ale nie posiadają odpowiednich narzędzi takich jak podnośnik czy klucz. Aplikacja będzie działać w systemie Android oraz iOS.

2. Analiza rynku oprogramowania w obszarze realizowanego projektu

Brak aplikacji konkurencyjnych.
Na rynku aplikacji mobilnych na systemy Android i iOS nie ma aplikacji, która mogłaby być konkurencyjną. Aplikacja „Pomocnik kierowcy" będzie pierwszą, która niesie pomoc poprzez wykorzystanie drugiego kierowcy w najbliższej okolicy. Aplikacja jest skierowana zarówno dla kierowców samochodów osobowych jak i dla kierowców zawodowych.

3. Wybór typu procesu wytwórczego. (W5HH principle) + uzasadnienie

Podczas realizacji projektu zostanie zastosowany model wytwarzania zwinnego (ang. Agile software development). Pozwala on na kontrolę ciągle zmieniającego się rynku. Ważnym aspektem tej metody jest bezpośredni kontakt z klientem oraz zasadnicza miara postępu, czyli działające oprogramowanie szybko dostarczone do odbiorców.

Wytwarzanie zwinne pozwala użytkownikom mieć wpływ na rozwój aplikacji. Dzięki temu procesowi implementacja nie spowoduje zaburzenia całego procesu wytwórczego, a nowa funkcjonalność zostanie dodana w możliwie krótkim czasie.
Ponadto Agile umożliwia regularną poprawę jakości dostarczanego oprogramowania, poprzez rozwiązywanie napotkanych błędów. Poza tym jest to model idealny dla małych zespołów, w których nie ma problemów ze sprawną komunikacją.

W5HH:
- Why – aplikacja powstaje, ponieważ zespół zauważył, że brakuje produktu, który pozwoliłby szybko i sprawnie nieść pomoc potrzebującym kierowcom w niewielkich usterkach pojazdu.

- What - zostanie stworzona aplikacja, zgodnie z wytycznymi zawartymi w tym projekcie.

- When - aplikacja ma mieć w pełni działające podstawowe funkcjonalności i być gotowa do wydania dla specjalnie wyselekcjonowanych klientów jako alpha w dniu 20 czerwca 2018.

-Who- projekt zostanie stworzony w małym, 3-osobowym zespole.

-Where- projekt nie wymaga pracy w jednym miejscu. Komunikacja pomiędzy członkami zespołu może odbywać się przez internet.

-How-Na początku zostanie przeprowadzona analiza wymagań. Opracowany zostanie sposób działania aplikacji. Kolejnym etapem będzie przygotowanie schematu blokowego, który umożliwi późniejszą implementację. Następnie zostanie przeprowadzona analiza poprawności napisanego kodu oraz przeprowadzone testy jednostkowe.

-How-W projekt będą zaangażowane 3 osoby. Do ukończenia i działania projektu potrzebne będą serwer bazodanowy i stacje robocze zaopatrzone w konieczne środowiska programistyczne(tj. Android Studio, Eclipse, Xcode). Wszyscy członkowie zespołu posiadają własne stacje robocze.
Po utworzeniu działającej wersji aplikacji z podstawowymi funkcjami i wyboru tylko jednej usterki zostanie udostępniona publiczna

wersja beta, następnie na podstawie opinii użytkowników poprawione zostaną błędy oraz dodane najbardziej pożądane funkcjonalności. Po zakończeniu tego etapu prac stopniowo będzie dodawane wsparcie dla innych usterek. Aplikacja pozwoli na łatwe zgłaszanie błędów i uwag za pomocą wbudowanego narzędzia. Dystrybucja będzie się odbywać za pośrednictwem Sklepu Play (Android) oraz App Store (iOS). Aplikacja będzie programowana w języku Java co pozwoli na łatwe wdrożenie w systemie Android oraz w języku Swift dla urządzeń korzystających z systemu iOS.

4. Analiza wymagań: zakres funkcjonalności, wymagania funkcjonalne i niefunkcjonalne, analiza MoSCoW

funkcjonalne:

1.Must
-Automatyczne pobieranie lokalizacji urządzeń
-Automatyczne udostępnianie lokalizacji urządzenia
-Możliwość wyboru pomocy
-Możliwość wyboru udzielenia lub odmówienia pomocy
-Możliwość kontaktu z udzielającym lub potrzebującym pomocy

2.Should
-Możliwość ustawienia autoaktualizacji udostępniania lokalizacji urządzenia
-Możliwość zgłaszania problemów z poziomu aplikacji
-Możliwość wyciszenia powiadomień

3.Could
-Możliwość autosynchronizacji wielu usterek w jednym widoku.
-Ostrzeżenia przy wyborze udzielenia pomocy w jednym czasie w kilku miejscach
-Edycja kolorów w widoku
-Historia udzielania pomocy
-Możliwość logowania przez facebook lub google+ i zachowanie ustawień w chmurze

4.Won’t

-Personalizacja interfejsu
-Synchronizacja z innymi aplikacjami
-Port Windows Phone

niefunkcjonalne:

1.Must
-Kompatybilność z Android 4.4+
-W pełni działająca baza danych przechowująca lokalizacje urządzeń

2.Should
-Przejrzysty interfejs użytkownika (Pozytywne opinie od przynajmniej połowy użytkowników)
-Wysoka stabilność (max 5% użytkowników zgłaszających błędy)
-Mały rozmiar (poniżej 75MB)

3.Could
-Rozbudowanie bazy danych o możliwość przechowywania historii udzielania pomocy przez użytkowników
-Szybkie działanie aplikacji (Pozytywne opinie od przynajmniej 60% użytkowników)

4.Won't
-Możliwość wyłączenia przycisku udzielenia pomocy
-Możliwość kontaktu z poprzednim użytkownikiem, który udzielił nam pomocy, ale jest poza zasięgiem aplikacji

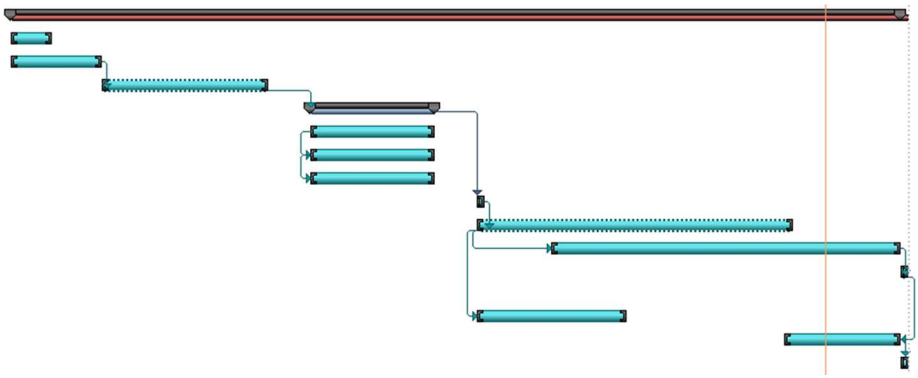
5. Oszacowanie pracochłonności metodą Cocomo 2

komponenty	98	**FC**	niska zł	średnia zł	wysoka zł	n-liczba	sr-liczba	wy-liczba	suma							
cechy funkcyjne (14)	32	we	3	4	6	3	1	1	19							
PF	95,06	wy	4	5	7	3	0	1	19							
LOC	3232,04	pliki	7	10	15	2	0	1	29							
Sj	14,39	int zew	5	7	10	0	3	0	21							
B	1,05	zap	3	4	6	0	1	1	10							
wsp. kosztu	1,83							**suma**	**98**							
		Fi	**32**													
E Cocomo2	**18,5**	pyt.1	pyt.2	pyt.3	pyt.4	pyt.5	pyt.6	pyt.7	pyt.8	pyt.9	pyt.10	pyt.11	pyt.12	pyt.13	pyt.14	
E Cocomo1	14,7	3	4	1	1	1	3	2	3	2	5	1	1	1	4	
		di														
		pyt.1	pyt.2	pyt.3	pyt.4	pyt.5	pyt.6	pyt.7	pyt.8	pyt.9	pyt.10	pyt.11	pyt.12	pyt.13	pyt.14	pyt.15
		1,40	1,16	1,00	1,11	1,06	1,15	0,87	1,00	1,17	0,91	1,00	0,95	0,91	1,00	1,04
		Sj														
		typowość	elastyczność	ryzyko	spójność	dojrzałość cmm										
		3,72	1,01	2,83	2,19	4,64										

Według analizy metodą Cocomo 2 wykonanie projektu zajmie około 18,5 osobomiesięcy.

6. Harmonogram procesu wytwórczego

Projekt	**77 dn**	**wto, 06.03.18**	**śro, 20.06.18**	
Planowanie	5 dn	wto, 06.03.18	sob, 10.03.18	
Analiza rynku	9 dn	wto, 06.03.18	pią, 16.03.18	
Analiza wymaga	15 dn	sob, 17.03.18	czw, 05.04.18	3
Projekt aplikacji	**11 dn**	**śro, 11.04.18**	**śro, 25.04.18**	**4**
Projekt strukt	11 dn	śro, 11.04.18	śro, 25.04.18	
Projekt danyc	11 dn	śro, 11.04.18	śro, 25.04.18	6RR
Projekt interfa	11 dn	śro, 11.04.18	śro, 25.04.18	7RR
Zakończenie pro	1 dzień	wto, 01.05.18	wto, 01.05.18	5
Kodowanie	28 dn	wto, 01.05.18	czw, 07.06.18	9
Testowanie	30 dn	czw, 10.05.18	śro, 20.06.18	10RR
Koniec testów i kodowania	1 dzień	czw, 21.06.18	czw, 21.06.18	11
Promocja	14 dn	wto, 01.05.18	pią, 18.05.18	10RR
Dokumentacja	10 dn	czw, 07.06.18	śro, 20.06.18	12ZZ
Zakończenie projek	1 dzień	czw, 21.06.18	czw, 21.06.18	14



7. Analiza i zarządzanie ryzykiem

Nazwa zagrożenia	Prawdopodobieństwo	skutki
Niezadowolenie użytkowników	25%	2
Niedokładny opis wymagań	35%	1
Przesunięcie terminu	4%	2
Niedyspozycja członka zespołu	10%	3
Zmiana wymagań	18%	3
Większa złożoność projektu niż przewidywana	15%	4
Awaria sprzętu, systemu lub narzędzi projektowych	4%	1
Zmniejszenie budżetu produktu	15%	3
Odejście pracownika	10%	3
Pojawienie się konkurencyjnego produktu	10%	3
Brak zainteresowania produktem	25%	2

Rozwiązania problemów:

1. Niezadowolenie użytkowników

   Użytkownicy będą mieli możliwość wysyłania pomysłów zastrzeżeń co do działania programu z poziomu aplikacji. Pozwoli to na szybkie wprowadzenie poprawek i zwiększenie satysfakcji użytkowników.

2. Niedokładny opis wymagań

   W razie wątpliwości co do wymagań pracownicy mogą skontaktować się z liderem. Aby zapewnić bezproblemową komunikację lider powinien

mieć stałe połączenie z internetem, co można zapewnić przez wykupienie pakietu danych mobilnych.

3. Przesunięcie terminu

Gdyby z jakiegoś powodu konieczne było przyspieszenie wydania aplikacji można skrócić fazę testów wewnętrznych i wydać aplikację jako wczesną wersję (wersja alpha). Jeśli budżet na to pozwoli to można również przyspieszyć proces wytwórczy zatrudniając dodatkowego pracownika. Na wypadek takiej ewentualności każdy pracownik musi prowadzić dokładną dokumentację pracy.

4. Niedyspozycja członka zespołu

Ponieważ cała praca może być wykonywana zdalnie wszyscy członkowie zespołu będą w stanie kontynuować pracę w razie niegroźnych urazów lub chorób. Na wypadek poważniejszych niedyspozycji wszyscy pracownicy muszą prowadzić dokładną dokumentację pracy. Na jej podstawie pozostali członkowie zespołu mogą pracować w zastępstwie.

5. Zmiana wymagań

Gdyby na podstawie feedbacku użytkowników konieczna była duża zmiana w projekcie na późnym etapie produkcji to wprowadzenie jej wymagałoby dużego nakładu pracy i czasu. Aby zminimalizować negatywne skutki kod powinien mieć budowę możliwie najbardziej modularną.

6. Większa złożoność projektu niż przewidywana

Należy zgromadzić ilość pieniędzy, która pozwoliłaby na zatrudnienie doświadczonego freelancera lub studenta, który pomoże w ukończeniu projektu.

7. Awaria sprzętu

Należy zakupić ubezpieczenie dla sprzętu i dokonywać częstych backupów w wielu różnych lokalizacjach (kopie zapasowe w chmurze i na dyskach twardych).

8. Zmniejszenie budżetu

Znalezienie możliwie najtańszych dostawców usług bazodanowych oraz

przyspieszenie wydania aplikacji. Może to wymagać konieczność wykluczenia niektórych funkcji z pierwszej wersji programu.

9. Odejście pracownika

   Należy szybko zastąpić pracownika . Ze względu na niewielki budżet projektu należy zatrudnić możliwie najtańszą siłę roboczą (studenta).

10. Pojawienie się konkurencyjnego produktu na rynku

    Konieczne będzie zbadanie funkcjonalności konkurencyjnej aplikacji i dodanie ich do naszej, a także uatrakcyjnić ją pod względem estetycznym.

11. Brak zainteresowania produktem

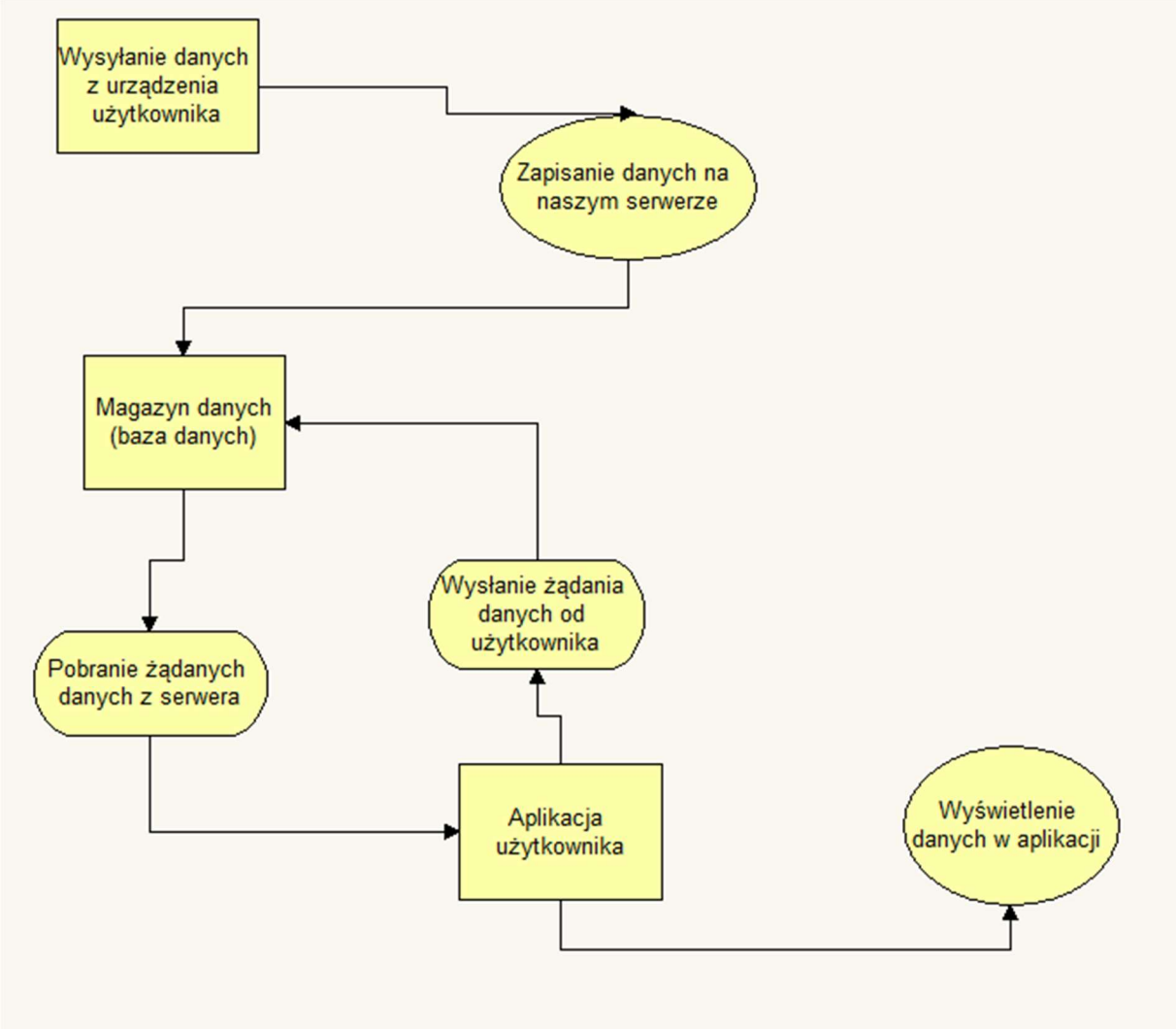
    Prowadzenie agresywnego marketingu mającego na celu przekonanie użytkowników do wyboru naszej aplikacji. Np. kampanie reklamowe na portalach społecznościowych oraz współpraca z firmami spedycyjnymi, kurierskimi.

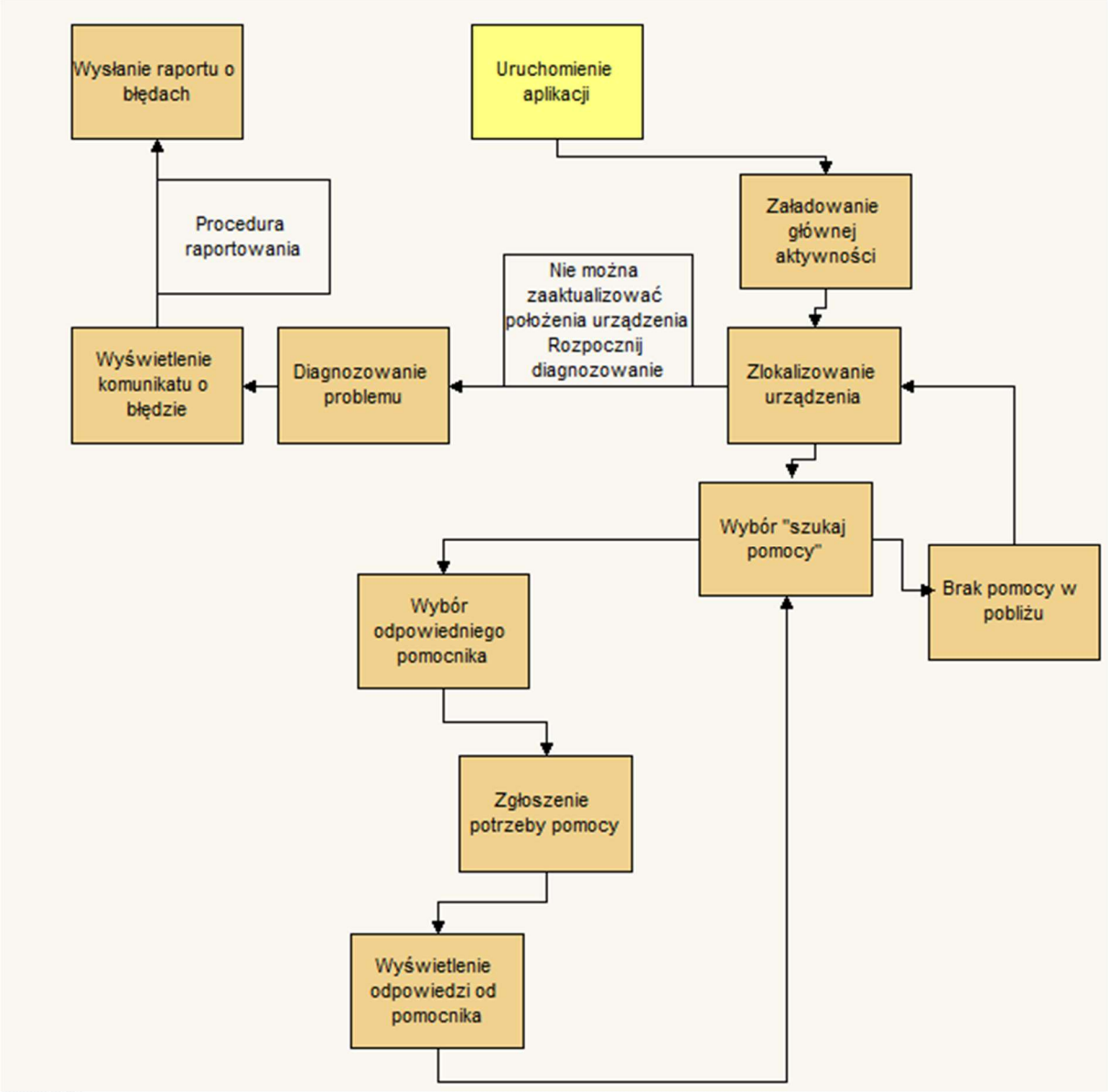
8. Wspomaganie jakości produktu - zastosowane metody i narzędzia

Aplikacja będzie posiadała wbudowane narzędzie, dzięki któremu użytkownik będzie mógł zgłosić napotkane błędy oraz przesłać swoje uwagi dotyczące aplikacji. Regularnie analizowane będą także opinie użytkowników w aplikacjach Sklep Play oraz App Store. Głównym narzędziem do kontroli projektu jest system kontroli wersji GIT, dzięki któremu widzimy dokumentację projektu w każdej chwili. Kod jest tworzony zgodnie z zasadami DRY, KISS i YAGNI.

9. Modelowanie analityczne: diagramy: przepływu danych, przypadków użycia wraz z krótkim opisem

Diagram przepływu danych:



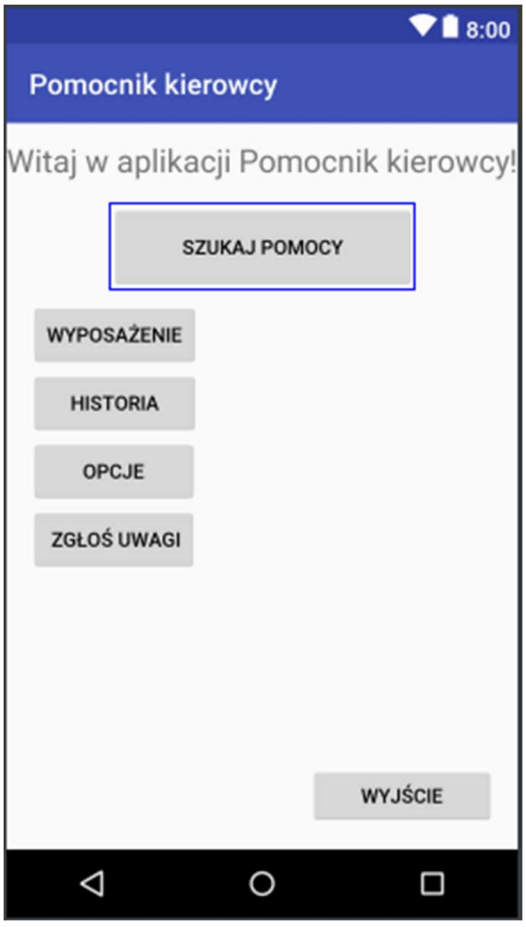


Użytkownik wybierając na początku w ustawieniach autoaktualizacje lokalizacji będzie wysyłał do bazy swoją aktualną pozycję w odpowiednich odstępach czasowych. Za każdym razem gdy wywoływana jest opcja „Szukaj pomocy” aplikacja łączy się z serwerem pobierając aktualne pozycje innych użytkowników przebywających w pobliżu, posiadających odpowiednie wyposażenie, które pozwoli na pomoc. Po wyborze odpowiedniego pomocnika, dostaje on powiadomienie o możliwości udzielenia pomocy, może pomóc lub odmówić.

10.Projekt architektury

Ponieważ system będzie składał się z 2 aplikacji o różnym przeznaczeniu wykorzystamy 2 architektury.
Aplikacja do pobierania i obróbki danych od klienta (użytkownika) będzie działać w architekturze przepływowej, której ostatecznym celem będzie baza danych.
Aplikacja dla klienta będzie działać według architektury skoncentrowanej na danych. Będzie ona przede wszystkim pobierać i wyświetlać przetworzone wcześniej dane z bazy danych.
Aplikacja od strony serwera będzie musiała pobrać dane od użytkownika, zapisać je w bazie danych, a następnie przekształci i wyśle na format zrozumiały dla aplikacji klienta.
Obie aplikacje napisane zostaną w Javie. Pozwoli to na łatwą implementację w systemach mobilnych.
Wybrane środowisko dla napisania programu od strony serwera to Eclipse, ponieważ jest ono bardzo dobrze wspierane oraz cały czas rozwijane.

11.Projekt interfejsu



12.Kodowanie

Ponieważ program ma działać w systemie android oraz iOS, należy wybrać środowisko i język, które pozwolą na sprawne pisanie i testowanie kodu.
Najpopularniejszymi środowiskami pozwalającymi na programowanie pod system android, dostępnymi obecnie na rynku są:
Android Studio-Java
IntelliJ IDEA Ultimate-Java
NetBeans z pluginem NBAndroid-Java
Eclipse z Eclipse for Android Developers-Java
Visual Studio z Xamarin-C#
Najlepiej wspieranymi środowiskami z tej listy są Android Studio, IntelliJ IDEA. Android Studio zapewnia wszystkie funkcjonalności niezbędne do stworzenia naszej aplikacji, więc nie ma potrzeby kupować IntelliJ IDEA Ultimate. Na korzyść Android Studio przemawia również fakt, że jest to oficjalne środowisko do tworzenia programów pod androida, wspierane przez Google i nie wymaga żadnych opłat.
Z tych powodów wykorzystane zostanie Android Studio.
Pod system iOS najpopularniejsze środowiska programistyczne dostępne na rynku to:
AppCode-Swift
Apple Xcode-Swift
Apple Xcode zapewnia wszystkie funkcjonalności, więc skupimy się na korzystaniu z tego środowiska. AppCode co prawda wspiera język Swift, ale dopiero od niedawna, wiec możliwe jest że nie działa on jeszcze tak wydajnie i bezproblemowo.

13.Testowanie

Naszą aplikację będziemy testować pod względem jakości kodowania (metoda białej skrzynki), jak i funkcjonalności aplikacji (metoda czarnej skrzynki). Od strony funkcjonalności przetestujemy w głównej mierze

funkcjonalności związane z przesyłem danych pomiędzy aplikacją kliencką, a serwerem, a także pozostałych interakcji użytkownika. Postaramy się poddać testom wszystkie możliwe funkcjonalności oraz każdy fragment kodu naszego projektu w celu poprawienia wyników działania aplikacji, zwiększenia jej wydajności i niezawodności. Ostatnim etapem testowania będzie przekazanie działającej już aplikacji klientom jako wersja beta w celu uzyskania ocen jakościowych i ewentualnych propozycji dalszego rozwoju.

14.Pielęgnacja

Platformami na której program zostanie wydany są Sklep Play oraz App Store. Umożliwi nam to dostęp do bardzo dużej bazy klientów, a użytkownikom łatwą instalację na urządzeniach z systemem android jak i iOS.
Aplikacja zostanie wydana pierwotnie jako wczesna wersja posiadająca tylko niezbędne funkcje. Użytkownicy będą mieli możliwość zgłaszania feedbacku z poziomu aplikacji. Pozwoli to na skupienie się na wdrażaniu w pierwszej kolejności najbardziej pożądanych funkcjonalności i dynamiczną zmianę projektu w razie konieczności.
Po wydaniu wersji 1.0 zawierającej wszystkie funkcjonalności zawarte w wymaganiach aplikacja będzie dalej rozwijana na podstawie opinii użytkowników. Kolejne wersje będą najpierw wydawane jako beta (dostępna jako oddzielna aplikacja dla chętnych). Pozwoli to przetestować wprowadzone zmiany pod kątem błędów i satysfakcji użytkowników zanim zaktualizowana zostanie główna wersja. Istotne będzie również uwzględnienie opinii oraz ocen
na stronie aplikacji w Sklep Play oraz App Store.

15.Podręcznik użytkownika

Aplikację należy pobrać ze Sklep Play lub App Store. Po pobraniu aplikacja zainstaluje się automatycznie. Po włączeniu aplikacji jeśli użytkownik jest podłączony do internetu, oraz ma włączoną lokalizacje

urządzenia zaktualizuje się baza pomocników w pobliżu oraz zostanie udostępniona i zapisana w bazie lokalizacja urządzenia. Po kliknięciu w przycisk “Szukaj pomocy” wyświetli się kilka opcji jakie musisz wybrać z listy aby przejść dalej m.in. jakiej pomocy oczekujesz. Następnie wyświetli się lista osób, które znajdują się w pobliżu. Po wybraniu pomocnika wyświetli się do niego kontakt. Użytkownik może skonfigurować odpowiednio swoje wyposażenie poprzez kliknięcie w przycisk „wyposażenie”. Pod przyciskiem „Historia” użytkownik może zobaczyć spis osób którym udzielił pomocy lub kto udzielił użytkownikowi pomocy. W zakładce opcji użytkownik może zmienić ustawienia takie jak automatyczna aktualizacja lokalizacji urządzenia, zmiana sposobu powiadomień czy też mniej istotne jak zmiana szaty kolorystycznej.
W zakładce “Zgłoś uwagi” użytkownik ma możliwość przesłać uwagi, opinie i błędy znalezione w programie. Z rozwijanej listy należy wybrać kategorię błędu i w polu poniżej wpisać treść wiadomości.