OpenWeatherMap

Programowanie Systemów Mobilnych

18.11.2024

Arkadiusz Bodziony, Martin Klimuk, Mikołaj Śnieżko
Politechnika Krakowska, WIŚiE

1 Etap I

1.1 Wstępny opis działania programu

Projekt zakłada stworzenie aplikacji mobilnej, która prezentuje aktualne dane pogodowe oraz prognozę na podstawie informacji z **API OpenWeatherMap**. Aplikacja będzie posiadała następujące funkcjonalności:

- **Wybór lokalizacji:**
  - Możliwość ręcznego wpisania nazwy miasta.
  - Automatyczne lokalizowanie użytkownika za pomocą GPS.
- **Prezentacja danych pogodowych:**
  - Wybór zakresu czasowego prognozy.
  - Wyświetlanie ostrzeżeń (np. burza, silny wiatr, ograniczona widoczność dla kierowców).
- **Dodatkowe opcje:**
  - Sugestie dotyczące ubioru w zależności od pogody (np. parasol, kurtka przeciwwiatrowa).
  - Prezentacja wykresu indeksu UV z informacją o ryzyku oparzeń skóry.
- **Ostrzeżenia meteorologiczne:**
  - Wysokie ciśnienie atmosferyczne.
  - Ograniczona widoczność.

1.2 Diagram przypadków użycia

1.3 Wymagania funkcjonalne i niefunkcjonalne

1.3.1 Wymagania funkcjonalne

- **Wybór miejsca do sprawdzenia pogody (wpisanie nazwy miasta/współrzędnych)**
  - **Wejście:** Nazwa miasta.
  - **Wyjście:** Lista dostępnych miast lub prognoza pogody dla wybranej lokalizacji.
  - **Wymagania:** Użytkownik musi mieć możliwość wprowadzenia nazwy miasta lub współrzędnych geograficznych (współrzędne mogą być dostarczone ręcznie lub przez GPS).
- **Automatyczne znalezienie pogody dzięki wbudowanej lokalizacji GPS w smartfonie.**
  - **Wejście:** Automatycznie zebrane dane GPS (szerokość i długość geograficzna).
  - **Wyjście:** Prognoza pogody dla bieżącej lokalizacji użytkownika.
  - **Wymagania:** Wymaga aktywnej funkcji GPS w urządzeniu mobilnym. Aplikacja musi mieć dostęp do lokalizacji urządzenia i wykorzystać ją do pobrania prognozy pogody.
- **Wybór dodatkowej opcji (checkbox) dla wskazówek dotyczących odpowiedniego ubioru do pogody.**
  - **Wejście:** Zaznaczenie odpowiedniej opcji w interfejsie użytkownika (checkbox).

- **Wyjście:** Wskazówki dotyczące odpowiedniego ubioru, np. parasol, kurtka przeciwwiatrowa.
- **Wymagania:** Aplikacja powinna analizować dane pogodowe (np. deszcz, temperatura) i dostarczać rekomendacje dotyczące ubioru. Wymaga interakcji z użytkownikiem w celu zaznaczenia preferencji.

- **Wyświetlanie potencjalnych zagrożeń (np. burza, silny wiatr – wyświetlić „!" wraz z informacją).**
  - **Wejście:** Dane pogodowe z API (np. informacje o silnym wietrze, burzy).
  - **Wyjście:** Powiadomienie użytkownika o zagrożeniu (np. wyświetlenie ikony „!").
  - **Wymagania:** Aplikacja powinna analizować dane pogodowe i generować powiadomienia o zagrożeniach pogodowych. Wymagany jest dostęp do odpowiednich danych z API oraz odpowiednia obsługa powiadomień.

- **Wyświetlenie informacji o wysokim ciśnieniu.**
  - **Wejście:** Dane o ciśnieniu atmosferycznym z API.
  - **Wyjście:** Informacja o wysokim ciśnieniu (np. "Wysokie ciśnienie, może wpłynąć na samopoczucie").
  - **Wymagania:** Aplikacja powinna wyświetlać ostrzeżenia o wysokim ciśnieniu, korzystając z odpowiednich danych pogodowych.

- **Wyświetlenie informacji o małej widoczności – ostrzeżenie kierowców.**
  - **Wejście:** Dane o widoczności z API.
  - **Wyjście:** Ostrzeżenie o małej widoczności (np. "Ostrzeżenie: ograniczona widoczność, zachowaj ostrożność podczas jazdy").
  - **Wymagania:** Aplikacja musi analizować dane o widoczności i wyświetlać odpowiednie ostrzeżenie. Wymaga monitorowania danych pogodowych.

1.3.2 Wymagania niefunkcjonalne

- **Wydajność:**
  - Aplikacja powinna działać płynnie na urządzeniach mobilnych z minimum 2 GB pamięci RAM.
  - Czas ładowania danych z API nie powinien przekraczać 5 sekund przy stabilnym połączeniu internetowym.

- **Skalowalność:**
  - System musi być przygotowany na zwiększoną liczbę użytkowników w przyszłości, bez zauważalnej utraty wydajności.
  - Struktura kodu powinna umożliwiać łatwą rozbudowę o dodatkowe funkcjonalności w przyszłości.

- **Kompatybilność:**
  - Aplikacja powinna być kompatybilna z systemami Android (od wersji 8.0 Oreo).
  - Interfejs użytkownika powinien dostosowywać się do różnych rozdzielczości ekranów (responsywność).

- **Dostępność:**
  - Interfejs aplikacji powinien być prosty w obsłudze, także dla osób bez doświadczenia z zaawansowanymi aplikacjami mobilnymi.
  - Powinny być uwzględnione podstawowe zasady projektowania dostępności, takie jak kontrast kolorów i możliwość obsługi aplikacji za pomocą czytników ekranu.
- **Bezpieczeństwo:**
  - Wszystkie połączenia z API powinny być szyfrowane (protokół HTTPS).
  - Aplikacja nie powinna przechowywać wrażliwych danych użytkownika lokalnie na urządzeniu.
- **Niezawodność:**
  - Aplikacja powinna działać stabilnie, minimalizując ryzyko awarii i niespodziewanych zamknięć.
  - W przypadku braku połączenia z internetem aplikacja powinna informować użytkownika o problemach z pobieraniem danych i umożliwiać ponowne próby połączenia.

1.4 Wybranie systemu kontroli wersji oraz platformy hostingu dla niej, utworzenie repozytorium

W celu integracji naszej wspólenj pracy oraz zapobieganiu pomyłek związanych z powielaniem kodów, używamy systemu kontroli wersji - **Git**. Za jego pomocą możemy na bieżąco aktualizować swoje kody oraz zauważać zmiany wprowadzone przez innych członków grupy. Jako platformę hostingową wybraliśmy **GitHub**.

- Opcje sklonowania repozytorium:
  - – git clone `https://github.com/mesniezek/OpenWeatherApp.git`
  - Przy dodawaniu nowego projektu w IDE, zaznaczyć opcję "**Get from VCS**"
- Kolejność wykonywania komend w celu pobrania aktualnej wersji głównego brancha **master**:
  - Kolejność wykonywania komend w celu pobrania aktualnej wersji głównego brancha **master**:
    - – git fetch origin
    - – git checkout master
    - – git pull origin master
  - Kolejność wykonywania komend w celu utworzenia/przeniesienia się na swojego brancha do pracy:
    - – git checkout -b <nazwa-brancha>
  - Kolejność wykonywania komend w celu zacommitowania i spushowania zmian:
    - – git add .
    - – git commit -m "Opis wprowadzonych zmian"
    - – git push origin <nazwa-brancha>

Przed podjęciem jakichkolwiek działań warto zapoznać się ze szczegółowym opisem działania poszczególnych komend. Do tego poza oficjalną dokumentacją, polecić możemy poradnik od firmy **Atlassian** - `https://www.atlassian.com/git`. Ponadto obecnie co raz mniej popularne staje się korzystanie z wiersza poleceń do tych działań, toteż warto zapoznać się z możliwościami swojego IDE w kwestii kontroli wersji. Przykładowy materiał pokazujący możliwości interfejsu użytkownika - `https://www.youtube.com/watch?v=Xzw6ttOIpnE&ab_channel=DanClarke`.

1.5 Raport ze stosowania metodologii programowania zwinnego

https://open-weather-map-java.atlassian.net/jira/software/projects/OWM/boards/1