

**TEMAT**            Aplikacja pogodowa - YoungPogosia  
**GRUPA**            Michał Liebner, Sebastian Maraszek

### KRÓTKI OPIS PROJEKTU:

Przygotowanie serwisu, który będzie zbierał dane dotyczące pogody z różnych zewnętrznych serwisów, zapisywał w bazie interesujące użytkownika rekordy, a następnie udostępniał zebrane dane przy pomocy własnego API. Z tym serwisem komunikowała będzie się aplikacja mobilna, która będzie odpowiadała za przedstawianie danych pogodowych w przyjazny dla użytkownika sposób.

### SPECYFIKACJA WYMAGAŃ:

SERWIS	APLIKACJA
<ul style="list-style-type: none"><li>• zbieranie danych pogodowych z co najmniej jednego zewnętrznego źródła</li><li>• przechowywanie danych pogodowych z miejsca zamieszkania użytkownika przez co najmniej rok</li><li>• przechowywanie danych pogodowych dla 10 różnych-wybranych przez użytkownika miejsc przez co najmniej rok</li><li>• pobieranie z zewnętrznego API prognoz pogody dla dowolnego, wybranego przez użytkownika miasta</li><li>• umożliwienie użytkownikowi wyboru jakie dane pogodowe mają być zapisywane w bazie danych (wilgotność, ciśnienie, temperatura odczuwalna, temperatura, prędkość wiatru, widoczność)</li><li>• Udostępnienie użytkownikowi dokumentacji serwisu, oraz obrazu dockerowego, który umożliwi użytkownikowi synchronizację z serwisem bez użycia dedykowanej przez nas aplikacji</li><li>• hostowanie serwisu w chmurze</li></ul>	<ul style="list-style-type: none"><li>• wyświetlanie danych pogodowych w formacie godzinowym, dniowym</li><li>• wyświetlanie pogody za pomocą serwisu używając wyszukiwarki</li><li>• wyświetlanie pogody za pomocą lokalizacji użytkownika</li><li>• zapisywanie w serwisie miejsc i wyświetlanie ich w liście do szybkiego przełączania się między nimi</li><li>• wyświetlanie szczegółowych warunków atmosferycznych (wilgotność, ciśnienie, temperatura odczuwalna, temperatura, prędkość wiatru, widoczność)</li><li>• zastosowanie najnowszych technologii i trendów projektowania aplikacji w celu dostarczenia klientowi aplikacji o zmiennym interfejsie, w zależności od upodobań użytkownika.</li><li>• możliwość podstawowego korzystania z aplikacji nawet w trybie offline</li><li>• dopasowywanie UI do aktualnej pogody</li><li>• hostowanie aplikacji w serwisie Firebase</li></ul>

**WSTĘPNY HARMONOGRAM PROJEKTU:**

DATA	WYDARZENIE
08.10.2021 - 17.10.2021	wybranie zespołu projektowego, określenie używanych technologii, przygotowanie wstępnych założeń projektowych, konsultacje wewnątrz zespołowe dotyczące realizowanego projektu.
17.10.2021 - 24.10.2021	Przygotowanie specyfikacji wymagań, WBS, oraz przedstawienie klientowi cech charakterystycznych wybranych technologii
24.10.2021 - 31.10.2021	Stworzenie diagramów UML prezentujących działanie systemu, wyszczególnienie ryzyk towarzyszących realizacji projektu YoungPogosia.
<b>ITERACJA I</b>	
31.10.2021 - 07.10.2021	Inicjalizacja obu części projektu, stworzenie bazy danych, uzyskanie danych niezbędnych do autoryzacji z zewnętrznymi API. Instalacja niezbędnych do działania aplikacji paczek z npm. Pierwsza struktura aplikacji.
07.10.2021 - 14.10.2021	Przygotowanie interfejsu strony głównej aplikacji oraz modułu komunikacji z zewnętrznym serwisem
14.10.2021 - 21.10.2021	Przygotowanie interfejsu listy miejsc oraz wyszukiwania. Przygotowanie pierwszej wersji API, oraz projektu bazy danych
<b>ITERACJA II</b>	
31.10.2021 - 07.10.2021	Ukończenie API, wsparcie techniczne dotyczące integracji, stworzenie dokładnej dokumentacji. Połączenie z serwerem aplikacji.
07.10.2021 - 14.10.2021	Refaktoryzacja kodu oraz naprawa błędów przed wystawieniem obu części aplikacji.
14.10.2021 - 21.10.2021	Wystawienie obu części aplikacji do serwisów hostingowych, przygotowanie obrazów dockerowych z serwisem. Przygotowanie narzędzi umożliwiających osobom trzecim kontrybucje do kodu poprzez narzędzia CI. Wystawienie produkcyjnej wersji.

## **CECHY CHARAKTERYSTYCZNE I POWODY UŻYCIA WYBRANYCH TECHNOLOGII:**

### **FastAPI:**

- Jeden z najszybszych frameworków webowych do Pythona
- Bardzo szybki development
- Automatyczne generowanie dokumentacji
- Opierające się na znanych i często używanych standardach (OpenAPI JSON Schema)

Fast API umożliwia stworzenie REST API w sposób bardzo szybki i wygodny dla developera. Jedynym poważnym konkurentem do Fast API był Django Rest Framework, jednak odrzuciliśmy tę kandydaturę ze względu na to, że framework Fast API jest zdecydowanie lżejszy, łatwiejszy do produkcyjnego użytkowania, a także posiadający asynchroniczne widoki, w przeciwieństwie do DRF.

### **MongoDB:**

- nierelacyjna baza danych
- Każda baza danych zawiera kolekcje, które z kolei zawierają dokumenty
- każdy dokument może być inny i mieć różną liczbę pól.
- Rozmiar i zawartość każdego dokumentu mogą się od siebie różnić.
- Struktura dokumentu jest bardziej zgodna ze sposobem, w jaki programiści konstruują swoje klasy i obiekty w odpowiednich językach programowania.
- Wiersze (lub dokumenty, jak wywoływane w MongoDB) nie muszą mieć wcześniej zdefiniowanego schematu.
- środowiska MongoDB są bardzo skalowalne.

Wybraliśmy MongoDB z tego względu, że ta baza daje nam dużo większą elastyczność niż w przypadku baz relacyjnych (możliwość działania aplikacji bez konieczności bolesnych produkcyjnie migracji). Kolejnym ważnym punktem jest szybkość implementacji, oraz fakt, że Mongo umożliwia zapisywanie danych w sposób bardziej przystający do logiki biznesowej, gdyż w aplikacji nie mamy zagnieżdżonych relacji obiektów, tylko pojedyncze rekordy zawierające informacje o pogodzie. Mongo umożliwia też lepsze skalowanie horyzontalne, niż w przypadku innych typów baz.

### **React:**

- Komponenty tworzone mogą być używane wielokrotnie.
- JS jest wszechstronny i może być używany z dowolnie wybranym frameworkiem.
- Oferuje wysoką wydajność, ponieważ jest oparty na virtual DOM.
- Umożliwia budowę dynamicznego interfejsu.
- Prostota w obsłudze oraz w .
- Jest stabilnym rozwiązaniem, ponieważ stoi za nim duża społeczność, która nieustannie się rozwija.

React to deklaratywna, wydajna i elastyczna biblioteka JavaScript do budowania interfejsów użytkownika. Pozwala komponować złożone interfejsy użytkownika z małych i odizolowanych fragmentów kodu zwanych „komponentami”. Ze względu na doświadczenie zespołu z technologią oraz szeroko dostępną oraz przystępną dokumentację, postanowiliśmy wykorzystać React w naszym projekcie.

#### **MUI:**

- Liczne komponenty łatwe do implementacji w stronie.
- CSS w JS pozwala na pisanie w prostszy sposób reaktywnego interfejsu.
- Możliwość konfigurowania komponentów do własnych potrzeb.
- System Grid zapewniający responsywny layout.
- “Tree Shaking” - usuwanie zbędnego kodu z paczki w wersji produkcyjnej .
- Jest stabilnym rozwiązaniem, ponieważ stoi za nim duża społeczność, która nieustannie się rozwija.

MUI (dawniej Material UI) zapewnia solidną, konfigurowalną i dostępną bibliotekę podstawowych i zaawansowanych komponentów, umożliwiającą zbudowanie własnego systemu projektowania i szybsze tworzenie aplikacji React. Komponenty w MUI są tworzone zgodnie z językiem projektowania Material Design stworzonym przez Google, dzięki temu prezentują się w atrakcyjny sposób. Biblioteka pomoże bezproblemowo zaimplementować nowoczesny i responsywny interfejs w aplikacji.