Szczecinek, dnia 07.10.2024

ZŁOMET PIOTR STEFANIAK

NIP: 6730004189

ul. Słowiańska 8-34,

78-400 Szczecinek

Załącznik nr 4

**SPECYFIKACJA**

**PROCEDURA WYBORU: „Bee Trend**”

**OPIS PRZEDMIOTU ZAMÓWIENIA**

**Specyfikacja techniczna:**

Celem projektu systemu informatycznego jest stworzenie inteligentnej platformy wspomagającej zarządzanie cyklem życia pasiek z wykorzystaniem zaawansowanej architektury opartej na technologii IoT (Internet of Things) oraz algorytmach uczenia maszynowego (ML), ze szczególnym naciskiem na wykrywanie, monitorowanie i walkę z kluczowymi chorobami pszczół, takimi jak warroza (Varroa destructor) oraz zgnilec amerykański (Paenibacillus larvae). System ten ma działać w rozproszonej infrastrukturze, umożliwiając analizę danych w czasie rzeczywistym, generowanie inteligentnych rekomendacji oraz automatyzację procesu wykrywania zagrożeń zdrowotnych w ulach.

**1. Moduł IoT do monitorowania pasiek**

Kluczową rolą systemu będzie zbieranie i przetwarzanie danych z szerokiego zestawu czujników IoT rozmieszczonych w ulach. Na podstawie tej infrastruktury sensorycznej, system będzie monitorował krytyczne parametry środowiskowe i behawioralne, które mogą wskazywać na ryzyko wystąpienia chorób. System wykorzystuje:

* **Czujniki temperatury i wilgotności** – do bieżącej analizy mikroklimatu w ulu. Wiadomo, że podwyższona wilgotność oraz nieodpowiednia temperatura są czynnikami sprzyjającymi rozwojowi zgnilca amerykańskiego.
* **Czujniki ruchu** – śledzące aktywność pszczół wewnątrz i na zewnątrz ula, co pozwala na identyfikację anomalii w zachowaniu, mogących wskazywać na infekcje Varroa destructor.
* **Czujniki wagi** – pomagające w monitorowaniu poziomu produkcji miodu oraz ewentualnych strat w populacji, co jest pośrednim wskaźnikiem kondycji kolonii.
* **Kamery na podczerwień** – umożliwiające analizę wizualną stanu ula, szczególnie w nocy, oraz wspierające algorytmy analizy obrazu w celu wykrycia pasożytów.

**1.1 Interfejs aplikacyjny REST**

Aby efektywnie zarządzać danymi pochodzącymi z różnych urządzeń IoT, kluczowe jest zaprojektowanie i wdrożenie wydajnego **interfejsu aplikacyjnego REST (Representational State Transfer)**. Interfejs ten będzie pełnił rolę centralnego mechanizmu komunikacyjnego pomiędzy czujnikami IoT a serwerem aplikacyjnym.

**1.1.1. Generyczna metoda transferu danych**

Jednym z głównych wymagań dotyczących interfejsu REST jest jego uniwersalność i elastyczność. Ponieważ system będzie współpracował z wieloma typami czujników i urządzeń IoT o różnej specyfikacji technicznej, kluczowe jest zaprojektowanie **jednej generycznej metody transferu danych**, która umożliwi przesyłanie nieustrukturyzowanych danych z różnych źródeł w sposób spójny i jednolity.

* **Unifikacja formatu danych**: Dane z różnych czujników (np. wartości temperatury, wilgotności, wagi, obraz z kamery) muszą być znormalizowane do wspólnego formatu, który pozwoli na ich łatwe przetwarzanie i analizę w dalszych etapach. Każdy czujnik będzie przesyłał dane w formacie JSON (JavaScript Object Notation), który zapewnia lekkość i łatwość serializacji danych oraz ich przetwarzania w aplikacjach webowych.
* **REST jako architektura komunikacji**: REST API będzie oparte na metodach HTTP, takich jak POST (do przesyłania nowych danych), GET (do pobierania zebranych danych), PUT (do aktualizacji istniejących danych) oraz DELETE (do usuwania danych, jeśli zajdzie taka potrzeba). Interfejs REST będzie również wykorzystywał typowe struktury zasobów REST, gdzie każde urządzenie IoT będzie reprezentowane przez unikalny URI (Uniform Resource Identifier).

**1.1.2. Zarządzanie nieustrukturyzowanymi danymi**

Czujniki IoT, szczególnie te o charakterze wizualnym (np. kamery), generują nieustrukturyzowane dane, takie jak obrazy, które wymagają specjalnego przetwarzania. Interfejs REST będzie odpowiedzialny za transfer tych danych w surowej formie, a następnie serwer aplikacyjny przeprowadzi ich dalszą obróbkę (np. analiza obrazów za pomocą algorytmów machine learning).

* **Przykład metody POST**: Czujniki będą regularnie przesyłać dane do centralnego serwera, korzystając z metody POST, wysyłając dane w formacie JSON. Każde urządzenie będzie dołączało do przesyłanych danych metadane, takie jak czas odczytu, identyfikator czujnika oraz lokalizację ula. Na przykład:

{

"device\_id": "sensor\_temp\_001",

"timestamp": "2024-10-01T14:30:00Z",

"temperature": 35.5,

"hive\_id": "hive\_001"

}

W przypadku kamery na podczerwień, dane obrazu będą przesyłane w formacie binarnym (np. jako strumień danych), a serwer aplikacyjny będzie odpowiedzialny za ich dalszą analizę.

**1.1. Transfer i przetwarzanie danych w systemie**

Po otrzymaniu danych z czujników przez interfejs REST, dane te zostaną przekazane do systemu przetwarzania zdarzeń opartego na architekturze kolejek zdarzeń (np. Apache Kafka lub RabbitMQ), co zapewni asynchroniczne przetwarzanie i skalowalność systemu.

* **Przetwarzanie asynchroniczne**: System kolejkowania zdarzeń pozwoli na rozdzielenie procesu zbierania danych z czujników od ich przetwarzania, co zwiększa wydajność oraz odporność na awarie. Dzięki temu, dane z wielu czujników mogą być przetwarzane równocześnie, a ewentualne opóźnienia w dostarczeniu danych nie będą wpływały na działanie całego systemu.
* **Obróbka danych na serwerze aplikacyjnym**: Serwer aplikacyjny będzie odpowiedzialny za przetwarzanie zebranych danych, w tym za ich analizę w czasie rzeczywistym (np. wykrywanie anomalii w danych środowiskowych lub zachowaniach pszczół), a także za ich przechowywanie w relacyjnej bazie danych, takiej jak PostgreSQL lub MySQL.

Projekt zakłada stworzenie, hosting oraz utrzymanie relacyjnej bazy danych, która będzie centralnym elementem systemu gromadzenia i przetwarzania danych z czujników IoT w pasiekach. Baza danych zostanie zaprojektowana w oparciu o technologie open-source, preferowane są **PostgreSQL** oraz **MySQL**, które zapewniają skalowalność, niezawodność oraz szerokie wsparcie dla analizy danych. Kluczowym elementem tego projektu jest stworzenie bazy danych zgodnie z notacją **ERD (Entity-Relationship Diagram)**, co umożliwi precyzyjne modelowanie relacji między poszczególnymi encjami (tabelami) w systemie.

**2. Stworzenie relacyjnej bazy danych**

Relacyjna baza danych będzie zaprojektowana w sposób umożliwiający gromadzenie i analizę danych z różnorodnych źródeł, takich jak czujniki IoT oraz wyniki algorytmów uczenia maszynowego. Baza będzie musiała być skalowalna i zdolna do obsługi dużych wolumenów danych pochodzących z wielu pasiek, jednocześnie zapewniając szybki dostęp do wyników analiz i raportów.

**2. Notacja ERD i modelowanie encji**

**Entity-Relationship Diagram (ERD)** jest kluczowym narzędziem w projektowaniu struktury bazy danych. Diagram ten pozwala na wizualizację encji (tabel) oraz relacji między nimi, co ułatwia zarówno proces projektowania, jak i późniejszą implementację.

W kontekście systemu pasiek, kluczowe encje w bazie danych to:

**2.1. Odczyty z aparatury IoT**

* **Opis encji**: Ta encja będzie przechowywać surowe dane pochodzące z czujników IoT umieszczonych w ulach. Dane te mogą obejmować takie parametry jak temperatura, wilgotność, waga, aktywność pszczół, obrazy z kamer na podczerwień oraz dane dźwiękowe.
* **Struktura danych**: Dane z czujników IoT będą przesyłane w formie nieustrukturyzowanej (np. w formacie JSON), ale będą powiązane z konkretnym ulem. Kluczowe pola to:
  + device\_id – identyfikator czujnika
  + timestamp – czas odczytu
  + sensor\_data – dane surowe (np. wartości temperatury, obrazy, nagrania dźwiękowe)
  + hive\_id – identyfikator ula, do którego dane się odnoszą

**2.2. Dane pasieki i poszczególnych uli**

* **Opis encji**: Encja ta będzie przechowywać szczegółowe informacje o pasiekach i poszczególnych ulach. Każdy ul w systemie będzie zarejestrowany jako osobna jednostka, co umożliwi indywidualne monitorowanie warunków w każdym z nich.
* **Struktura danych**:
  + hive\_id – unikalny identyfikator ula
  + hive\_name – nazwa ula
  + location – współrzędne GPS lub lokalizacja pasieki
  + beekeeper\_id – identyfikator właściciela pasieki
  + infrastructure\_status – status techniczny ula (np. gotowość do pracy, zaplanowane serwisy)

**2.3. Encja pasieki (relacja z encją ula)**

* **Opis encji**: Ta encja będzie modelować całą pasiekę, która składa się z wielu uli. Encja ta będzie miała relację typu **jeden do wielu** z encją ula, ponieważ jedna pasieka może składać się z wielu uli.
* **Struktura danych**:
  + apiary\_id – unikalny identyfikator pasieki
  + apiary\_name – nazwa pasieki
  + location – lokalizacja pasieki (adres, współrzędne GPS)
  + hive\_count – liczba uli w pasiece
  + beekeeper\_id – identyfikator właściciela pasieki

**2.4. Dane partnerów/użytkowników platformy oraz ich typy**

* **Opis encji**: Encja użytkowników będzie zawierała informacje o osobach korzystających z platformy, w tym właścicielach pasiek, administratorach, a także potencjalnych partnerach biznesowych (np. naukowcy, badacze).
* **Struktura danych**:
  + user\_id – unikalny identyfikator użytkownika
  + user\_name – imię i nazwisko użytkownika
  + user\_role – rola użytkownika na platformie (np. administrator, właściciel pasieki, badacz)
  + contact\_info – dane kontaktowe użytkownika
  + permissions – uprawnienia użytkownika w systemie (np. dostęp do danych, możliwość edycji, możliwość administracji)

**2.5. Encje związane z cykliczną analizą algorytmiczną ML**

* **Opis encji**: Ta encja będzie przechowywać wyniki cyklicznej analizy danych z algorytmów uczenia maszynowego. Analizy te będą wykonywane na podstawie danych z czujników IoT i mają na celu wykrywanie anomalii, przewidywanie problemów zdrowotnych w pasiekach oraz optymalizację zarządzania.
* **Struktura danych**:
  + ml\_analysis\_id – unikalny identyfikator analizy ML
  + analysis\_timestamp – czas przeprowadzenia analizy
  + hive\_id – identyfikator ula, którego dotyczy analiza
  + analysis\_result – wynik analizy (np. wykryte anomalie, rekomendacje działań)
  + ml\_algorithm – nazwa zastosowanego algorytmu ML (np. CNN dla analizy obrazu, RNN dla analizy dźwięku)
  + sensor\_data\_reference – odniesienie do odczytów z czujników, na podstawie których wykonano analizę

### 3. Utworzenie rozproszonego magazynu zdarzeń biznesowych i asynchronicznego systemu przetwarzania zdarzeń

W ramach projektu zakłada się stworzenie **rozproszonego magazynu zdarzeń biznesowych** oraz asynchronicznego systemu przetwarzania tych zdarzeń, które będą reprezentować kluczowe operacje pochodzące z realnego świata (np. odczyty z czujników IoT, takie jak rejestracja temperatury, poziomu hałasu czy wilgotności). System ten będzie służył jako interfejs pośredniczący pomiędzy zintegrowanymi urządzeniami (np. czujniki IoT) a bazą danych oraz serwerem aplikacyjnym. Ponadto, pełnić będzie rolę bufora, zapewniającego skalowalną wydajność aplikacji oraz niezawodność w przetwarzaniu danych w czasie rzeczywistym.

**3.1. Rozproszony magazyn zdarzeń biznesowych**

Rozproszony magazyn zdarzeń biznesowych to centralny komponent, w którym gromadzone będą wszystkie zdarzenia związane z operacjami systemu. Zdarzenia te obejmują różnorodne dane pochodzące z czujników IoT zainstalowanych w ulach, takie jak:

* **Odczyt temperatury** wewnątrz ula.
* **Rejestracja poziomu hałasu** w ulu (np. dźwięki emitowane przez pszczoły).
* **Odczyt poziomu wilgotności**, co ma kluczowe znaczenie dla utrzymania odpowiednich warunków w ulu.
* **Odczyt wagi ula**, co pozwala na monitorowanie poziomu produkcji miodu.

Dane te są zbierane w czasie rzeczywistym i muszą być przesyłane oraz przetwarzane w sposób asynchroniczny, aby zapewnić ciągłość działania systemu nawet przy dużym natężeniu ruchu danych.

**3.2. Asynchroniczne przetwarzanie zdarzeń**

**Asynchroniczne przetwarzanie zdarzeń** to kluczowa cecha systemu, która zapewnia niezawodność oraz elastyczność w obsłudze zdarzeń w czasie rzeczywistym. Oznacza to, że dane napływające z różnych źródeł (np. czujników IoT) nie muszą być przetwarzane natychmiastowo przez serwer aplikacyjny, ale mogą być tymczasowo przechowywane w buforze (kolejce) do momentu, gdy system będzie gotowy do ich przetworzenia.

**3.1. Magazyn zdarzeń a system kolejkowania**

Magazyn zdarzeń będzie zintegrowany z systemem kolejkowania, który będzie pełnił funkcję głównego mechanizmu orkiestracji wymiany danych między serwerem aplikacyjnym a bazą danych. System kolejkowania:

* **Zgromadzi wszystkie zdarzenia** (np. odczyty z czujników IoT) i umieści je w kolejce zdarzeń, co pozwoli na ich przetwarzanie w odpowiedniej kolejności.
* **Zwiększy niezawodność systemu** poprzez wprowadzenie buforowania danych – w przypadku opóźnień lub chwilowej niedostępności bazy danych, zdarzenia będą przechowywane w kolejce i przetwarzane, gdy system zostanie przywrócony do pełnej funkcjonalności.
* **Zminimalizuje opóźnienia** w przetwarzaniu danych poprzez efektywną dystrybucję zdarzeń do odpowiednich usług aplikacji.

**3.2. Przykład przetwarzania zdarzeń**

* Każde zdarzenie, takie jak odczyt temperatury, będzie traktowane jako pojedyncza wiadomość w systemie kolejkowania.
* Zdarzenie zostanie umieszczone w kolejce (np. w systemie **Apache Kafka** lub **RabbitMQ**), gdzie oczekuje na przetworzenie przez serwer aplikacyjny.
* Serwer aplikacyjny odczytuje zdarzenie, przetwarza dane (np. wykonuje analizę odczytu temperatury) i zapisuje wynik do bazy danych.

**4. Bufor danych i skalowalność**

System kolejkowania pełni również rolę **bufora danych**, który pozwala na rozdzielenie procesów zbierania danych od ich przetwarzania, co gwarantuje wysoką skalowalność systemu:

* **Skalowalność pionowa**: W przypadku zwiększonego napływu danych, system może dynamicznie zwiększać liczbę procesów odpowiedzialnych za odbiór zdarzeń (np. w serwerach aplikacyjnych), co pozwala na równoczesne przetwarzanie większej liczby zdarzeń.
* **Skalowalność pozioma**: W przypadku bardzo dużej liczby zdarzeń, system kolejkowania (np. Apache Kafka) może być rozproszony na wiele węzłów (klastrów), co zapewnia równomierne rozłożenie obciążenia i gwarantuje stabilną wydajność nawet przy ogromnych ilościach danych.

**5. Rekomendowane technologie: Apache Kafka i RabbitMQ**

Do implementacji systemu przetwarzania zdarzeń rekomendowane są dwie technologie: **Apache Kafka** oraz **RabbitMQ**, które oferują niezawodność, skalowalność oraz możliwość zarządzania danymi w sposób asynchroniczny.

**5.1. Apache Kafka**

**Apache Kafka** to rozproszony system strumieniowego przetwarzania danych, który jest idealnym rozwiązaniem do obsługi dużych ilości zdarzeń w czasie rzeczywistym. Kafka umożliwia:

* **Trwałe przechowywanie zdarzeń** – zdarzenia mogą być przechowywane w systemie Kafka przez określony czas (parametryzowana retencja danych), co pozwala na późniejszą analizę i replikację danych.
* **Asynchroniczne przetwarzanie strumieniowe** – Kafka pozwala na przetwarzanie strumieni danych w sposób rozproszony, umożliwiając łatwe skalowanie systemu przy rosnącej liczbie zdarzeń.
* **Redundancja i niezawodność** – Kafka zapewnia replikację zdarzeń na wiele węzłów, co zwiększa odporność systemu na awarie.

**5.2. RabbitMQ**

**RabbitMQ** to system kolejkowania wiadomości, który pozwala na niezawodną wymianę zdarzeń między różnymi komponentami systemu. RabbitMQ oferuje:

* **Efektywną dystrybucję zdarzeń** – RabbitMQ umożliwia routing zdarzeń do odpowiednich odbiorców, w zależności od typu zdarzenia lub priorytetu.
* **Możliwość potwierdzenia odbioru wiadomości** – RabbitMQ pozwala na niezawodne przetwarzanie zdarzeń dzięki mechanizmowi potwierdzania, co oznacza, że zdarzenia nie zostaną usunięte z kolejki, dopóki nie zostaną prawidłowo przetworzone.
* **Wsparcie dla wielu protokołów** – RabbitMQ obsługuje różne protokoły komunikacyjne (np. AMQP, MQTT), co zapewnia elastyczność w integracji z różnorodnymi systemami IoT.

**6. Trwałość i parametryzacja retencji obiektów**

Jednym z kluczowych wymagań wobec systemu magazynu zdarzeń jest **trwałość zdarzeń**, co oznacza, że wszystkie zdarzenia powinny być przechowywane w sposób trwały do momentu ich pełnego przetworzenia lub archiwizacji. Możliwość parametryzacji retencji obiektów w magazynie zdarzeń pozwala na dostosowanie okresu przechowywania zdarzeń w zależności od potrzeb systemu:

* **Parametryzacja retencji**: Zdarzenia mogą być przechowywane przez określony czas (np. 7 dni), co umożliwia ich analizę w późniejszym czasie oraz reagowanie na ewentualne błędy w przetwarzaniu danych.
* **Archiwizacja zdarzeń**: Po zakończeniu okresu retencji, zdarzenia mogą być automatycznie usuwane z systemu lub archiwizowane w innym repozytorium w celu długoterminowego przechowywania.

System wyposażony będzie w zaawansowane moduły analityczne, które będą przetwarzać dane pochodzące z czujników IoT (Internet of Things) umieszczonych w ulach i stosować algorytmy uczenia maszynowego do wykrywania anomalii oraz zagrożeń związanych z chorobami pszczół. Algorytmy będą analizować dane z różnych sensorów, takie jak obrazy z kamer, dźwięki z czujników akustycznych, odczyty temperatury i wilgotności, aby zidentyfikować potencjalne zagrożenia dla kolonii pszczół i zaproponować odpowiednie interwencje.

**7.Projekt systemu do przetwarzania i analizy nieustrukturyzowanych danych IoT z ulów – architektura, dane i algorytmy uczenia maszynowego**

Projekt zakłada opracowanie kompleksowego systemu analitycznego, który przetwarza i analizuje dane pochodzące z urządzeń IoT zainstalowanych w ulach. System składa się z serwera aplikacji odpowiedzialnego za gromadzenie, przetwarzanie oraz analizę zarówno danych strukturalnych, jak i nieustrukturyzowanych, wykorzystując algorytmy uczenia maszynowego (ML). Jego głównym celem jest monitorowanie stanu zdrowia kolonii pszczół, wykrywanie anomalii oraz generowanie raportów i powiadomień w czasie rzeczywistym.

**7.1. Architektura serwera aplikacji i przetwarzania danych**

**7.1.1. Webservice REST API**

Dane z czujników IoT, takich jak kamery na podczerwień, czujniki temperatury, wilgotności, hałasu i inne sensory środowiskowe, będą przesyłane na serwer aplikacyjny za pomocą REST API. Dane te będą dostarczane w formacie JSON, co zapewnia ich lekką i szybką transmisję oraz łatwość przetwarzania. Serwer będzie przetwarzał dane zarówno w trybie asynchronicznym, jak i synchronicznym, aby maksymalizować efektywność i umożliwić obsługę dużych strumieni danych.

**7.1.2. Przetwarzanie danych strukturalnych**

Dane strukturalne to uporządkowane informacje, które posiadają określoną strukturę i są łatwo przechowywane w relacyjnych bazach danych. Przykłady danych strukturalnych obejmują:

* **Odczyty z czujników IoT**: Numeryczne dane z czujników temperatury, wilgotności, hałasu czy wagi ula, które są zapisywane w formie tabelarycznej w bazie danych.

{

"sensor\_id": "temp\_sensor\_001",

"hive\_id": "hive\_001",

"timestamp": "2024-10-02T10:30:00Z",

"temperature": 35.5,

"humidity": 65.2

}

* **Dane historyczne**: Wyniki wcześniejszych analiz, które są przechowywane, aby umożliwić analizę trendów oraz prognozowanie przyszłych zdarzeń.

{

"hive\_id": "hive\_001",

"analysis\_id": "ml\_analysis\_100",

"analysis\_timestamp": "2024-10-01T14:00:00Z",

"anomaly\_detected": true,

"temperature": 35.5,

"humidity": 60.0,

"ml\_result": "Varroa detected"

}

**7.1.3. Przetwarzanie danych nieustrukturyzowanych**

Dane nieustrukturyzowane to dane, które nie mają jasno określonej struktury, przez co trudniej je przetwarzać i przechowywać w relacyjnych bazach danych. W ramach systemu będą to:

* **Obrazy z kamer na podczerwień**: Przesyłane jako binarne dane w kodowaniu Base64, przetwarzane następnie przez algorytmy rozpoznawania wzorców (CNN) w celu wykrywania anomalii w ulu.

{

"camera\_id": "infrared\_cam\_001",

"hive\_id": "hive\_001",

"timestamp": "2024-10-02T11:00:00Z",

"image\_data": "iVBORw0KGgoAAAANSUhEUgAAAoAAAA..."

}

* **Nagrania dźwiękowe**: Analizowane pod kątem nietypowych zmian w emisji dźwięków przez pszczoły, co może sugerować problemy zdrowotne w kolonii.

{

"audio\_sensor\_id": "audio\_sensor\_001",

"hive\_id": "hive\_001",

"timestamp": "2024-10-02T10:30:00Z",

"audio\_data": "UklGRiQAAABXRUJQVlA4IC4AAAA..."

}

7.2 **Moduł analityczny oparty na algorytmach uczenia maszynowego (ML)**

**7.2.1. Wykrywanie warrozy (Varroa destructor)**

Warroza, wywoływana przez pasożytniczego roztocza Varroa destructor, jest jednym z głównych zagrożeń dla zdrowia pszczół. Moduły analityczne systemu będą korzystać z zaawansowanych technologii, aby zidentyfikować infestacje na wczesnym etapie.

* **Computer Vision**: System będzie korzystał z kamer na podczerwień zamontowanych w ulach, które będą przekazywały strumień danych wideo. Algorytmy **Convolutional Neural Networks (CNN)**, trenowane na danych obrazowych przedstawiających pasożyty Varroa destructor, będą analizować obrazy, aby automatycznie rozpoznawać obecność roztoczy na ciałach pszczół. Dzięki temu, możliwe będzie wczesne wykrycie infekcji, co pozwoli na szybszą interwencję i zmniejszenie strat w kolonii.
* **Analiza behawioralna**: Czujniki ruchu i aktywności pszczół zainstalowane w ulach dostarczą danych, które będą analizowane przez system pod kątem zmian w normalnych wzorcach zachowania kolonii. Niska aktywność lotna, zmniejszenie ruchu wewnątrz ula lub zmiany w schematach poruszania się pszczół mogą wskazywać na rozwijającą się infekcję pasożytniczą. Algorytmy będą monitorować te zmiany, by zidentyfikować anomalia związane z infestacją Varroa destructor.
* **Rekomendacje interwencyjne**: Na podstawie wyników analiz obrazu i behawioralnych, system wygeneruje zalecenia dotyczące środków terapeutycznych. Wśród rekomendacji mogą znaleźć się:
  + **Zabiegi chemiczne**: System może zasugerować zastosowanie środków chemicznych, takich jak amitraz, kwas mrówkowy czy kwas szczawiowy, które są stosowane w zwalczaniu roztoczy Varroa.
  + **Interwencje mechaniczne**: Algorytmy mogą również rekomendować zastosowanie specjalnych ramek wyłapujących roztocza, co jest mechaniczną metodą walki z pasożytami.

**7.2.2. Wykrywanie zgnilca amerykańskiego (Paenibacillus larvae)**

Zgnilec amerykański to choroba bakteryjna atakująca larwy pszczół, której szybkie rozpoznanie jest kluczowe dla ochrony kolonii. System analityczny będzie korzystać z kombinacji różnych technologii, aby monitorować warunki sprzyjające rozwojowi zgnilca oraz wykrywać jego objawy kliniczne.

* **Monitorowanie parametrów środowiskowych**: Dane z czujników wilgotności i temperatury wewnątrz ula będą analizowane przez system w celu zidentyfikowania warunków, które sprzyjają rozwojowi Paenibacillus larvae. Zbyt wysoka wilgotność i nieodpowiednia temperatura mogą stwarzać idealne warunki dla rozwoju tej choroby. System wykryje takie anomalie i zaleci działania prewencyjne, np. poprawę wentylacji lub dostosowanie temperatury w ulu.
* **Analiza akustyczna**: Czujniki dźwięku umieszczone w ulach będą rejestrować poziom hałasu oraz analizować charakterystyczne dźwięki emitowane przez kolonie pszczół. Zmiany w tonie lub intensywności dźwięków, zwłaszcza śpiewu królowej pszczół, mogą wskazywać na problemy zdrowotne w kolonii, takie jak infekcje bakteryjne. Algorytmy ML będą analizować te dane, wykrywając wszelkie anomalie akustyczne, które mogą być sygnałem rozwijającej się choroby.
* **Wykrywanie objawów klinicznych**: Algorytmy analizy obrazu będą przetwarzać dane z kamer zamontowanych w ulach, aby identyfikować objawy zgnilca amerykańskiego, takie jak zmiany w strukturze larw, wygląd plastrów lub inne oznaki choroby. System rozpozna charakterystyczne wzorce wskazujące na zgnilec i wygeneruje odpowiednie rekomendacje dotyczące działań interwencyjnych.
* **Rekomendacje dotyczące walki z zgnilcem**: Po wykryciu ryzyka infekcji, system wygeneruje zalecenia dotyczące kroków, które należy podjąć, aby opanować sytuację. Wśród możliwych działań będą:
  + **Izolacja zakażonych uli**: System może zalecić izolację dotkniętych chorobą uli, aby zapobiec rozprzestrzenianiu się infekcji.
  + **Usunięcie i dezynfekcja ramek**: System może zaproponować usunięcie zakażonych ramek z ula i ich odpowiednią dezynfekcję w celu wyeliminowania bakterii Paenibacillus larvae.
  + **Zastosowanie środków chemicznych**: W przypadku zaawansowanej infekcji, algorytmy mogą zasugerować użycie środków chemicznych, takich jak tlenek etylenu, który jest skuteczny w walce z bakteriami.
  + **Kontrole higieny sprzętu**: System może również rekomendować kontrole higieny sprzętu pszczelarskiego oraz rotację ramek, co pomoże w zapobieganiu dalszym infekcjom.

7.3. Przykład danych strukturalnych systemu informatycznego:

**7.3.1. Analiza obrazu (Computer Vision)**

Wykorzystując algorytmy CNN, system analizuje obrazy z kamer na podczerwień, wykrywając:

* **Larwy Varroa**: System analizuje obrazy w celu wczesnej identyfikacji pasożytniczych roztoczy Varroa destructor.
* **Wirus zdeformowanego skrzydła**: Algorytmy rozpoznają wizualne objawy choroby pszczół, umożliwiając szybkie reagowanie.

{

"hive\_id": "hive\_001",

"image\_id": "img\_2024-10-02\_110000",

"timestamp": "2024-10-02T11:00:00Z",

"anomaly\_detected": true,

"detected\_issue": "Varroa detected",

"confidence\_level": 0.95

}

**7.3.2. Analiza dźwięku**

Algorytmy ML analizują sygnały dźwiękowe z uli, wykrywając zmiany w tonie oraz intensywności dźwięków, które mogą świadczyć o stresie lub chorobach w kolonii pszczół.

{

"hive\_id": "hive\_001",

"audio\_id": "audio\_2024-10-02\_103000",

"timestamp": "2024-10-02T10:30:00Z",

"anomaly\_detected": false,

"detected\_issue": "No anomalies detected",

"confidence\_level": 0.85

}

**7.3.3. Wykrywanie anomalii temperatury**

Algorytmy porównują aktualne odczyty z czujników temperatury z danymi historycznymi, aby wykryć nietypowe odchylenia, np. nagłe wzrosty lub spadki temperatury, które mogą wskazywać na problemy z wentylacją lub inne zagrożenia.

{

"hive\_id": "hive\_001",

"timestamp": "2024-10-02T11:30:00Z",

"temperature": 37.0,

"anomaly\_detected": true,

"detected\_issue": "Temperature spike",

"confidence\_level": 0.90

}

**7.4. Moduł predykcji i analiza historyczna**

Moduł predykcyjny oparty na algorytmach ML będzie przetwarzał zarówno dane w czasie rzeczywistym, jak i dane historyczne zebrane w trakcie działania systemu. Celem tego modułu będzie przewidywanie potencjalnych zagrożeń i anomalie w pasiekach, co pozwoli na proaktywne działania zapobiegawcze. Moduł ten pozwoli na identyfikowanie długoterminowych trendów w warunkach środowiskowych ula oraz zdrowiu kolonii pszczół.

Struktura predykcji obejmie:

* Prognozowanie rozwoju chorób na podstawie bieżących warunków i wcześniejszych odczytów.
* Wykrywanie trendów, takich jak zmiany w aktywności pszczół, które mogą wskazywać na nadchodzące problemy zdrowotne lub środowiskowe.
* Sugestie dotyczące optymalnych działań prewencyjnych.

**7.5. Harmonogramowanie i uruchamianie algorytmów ML**

Algorytmy ML będą mogły być uruchamiane automatycznie i ręcznie:

* **Automatyczne uruchamianie**: Za pomocą narzędzi takich jak **cron**, algorytmy będą uruchamiane cyklicznie w określonych przedziałach czasowych (np. codziennie lub co godzinę), aby przeprowadzać regularną analizę danych i monitorować sytuację w czasie rzeczywistym.
* **Ręczne uruchamianie**: Użytkownicy będą mogli uruchomić algorytmy ręcznie poprzez odpowiednie wywołanie metod HTTP za pomocą REST API. Możliwe będzie wywoływanie analizy na żądanie w dowolnym momencie, co zapewni elastyczność w zarządzaniu pasieką.

**7.6. Preferowane technologie i języki programowania**

W celu zapewnienia optymalnej wydajności i skalowalności systemu, do implementacji algorytmów oraz serwera aplikacji zaleca się użycie następujących technologii:

* **Java**: Używana do tworzenia serwerów aplikacyjnych, które są skalowalne i wydajne, szczególnie w przypadku dużych systemów backendowych.
* **Python**: Język preferowany do implementacji algorytmów uczenia maszynowego ze względu na wsparcie dla zaawansowanych bibliotek ML, takich jak **TensorFlow** i **PyTorch**, co umożliwia łatwą implementację i trenowanie algorytmów.

**8. Stworzenie serwera aplikacji mobilnej – system Back-End z usługą „Storage”**

Serwer aplikacji mobilnej będzie pełnił rolę systemu Back-End, który odpowiada za zarządzanie danymi użytkowników, pasiek oraz integrację z aplikacją mobilną. Serwer będzie odpowiedzialny za przechowywanie i przetwarzanie danych użytkowników, infrastruktury uli oraz powiązanych procesów operacyjnych, jak również zapewnienie obsługi dla systemu GDPR, hashowanie haseł, oraz segmentację klientów. W celu zapewnienia optymalnej wydajności i redukcji latencji połączeń, system zostanie zintegrowany z **Content Delivery Network (CDN)**, co umożliwi szybki dostęp do danych i obsługę aplikacji mobilnej.

**8.1 Konta użytkowników B2C, zgody regulaminowe i segmentacja użytkowników**

* **Konta użytkowników B2C**: Serwer będzie obsługiwał proces rejestracji oraz logowania użytkowników. Wszystkie hasła będą przechowywane w postaci bezpiecznego skrótu za pomocą algorytmu **SHA-512**, co zapewni ich odpowiednie zabezpieczenie.
* **Zgody marketingowe i regulaminowe**: System zapewni pełną obsługę zgodną z **GDPR**, w tym przechowywanie zgód użytkowników na przetwarzanie danych oraz zgód marketingowych.
* **Segmentacja użytkowników**: Użytkownicy będą podzieleni na segmenty w celu personalizacji treści oraz wyświetlania odpowiednich informacji w aplikacji mobilnej.

**8.2 Obiekty reprezentujące ul i pasiekę**

* **Relacja z kontami użytkowników**: Obiekty reprezentujące pasiekę i ule będą powiązane z kontami użytkowników. Każdy ul i pasieka będą przechowywać szczegółowe informacje dotyczące stanu infrastruktury, nadchodzących inspekcji oraz odczytów z czujników.
* **Obiekt ula** powinien zawierać następujące dane strukturalne:
  + **Stan infrastruktury IT**: Informacje o kondycji technologii zainstalowanych w ulach, w tym czujników IoT, łączności, dostępności sieci oraz ewentualnych błędach sprzętowych lub problemach technicznych.
  + **Data nadchodzącego okna serwisowego**: Informacje o nadchodzących inspekcjach i konserwacjach systemów zainstalowanych w ulu.
  + **Status ula**: Aktualny status działania ula, np. „Aktywny”, „Wymaga serwisu”, „Czekanie na inspekcję”.
  + **Odczyty uśrednione**: Zestawienie danych o średnich odczytach (temperatury, wilgotności, wagi, hałasu) per dzień, miesiąc, rok.

**8.3 Content Management System (CMS)**

System będzie zawierał **prosty CMS**, który umożliwi zarządzanie statycznymi informacjami wyświetlanymi w aplikacji mobilnej. CMS będzie zawierał funkcje personalizacji treści w zależności od segmentu użytkownika. Treści będą przechowywane na serwerze aplikacji, a integracja z CDN (np. **Cloudflare** lub **Akamai**) zapewni optymalizację wydajności oraz redukcję opóźnień HTTP pomiędzy klientem a serwerem aplikacji.

**8.4 Przetwarzanie danych z serwera analitycznego**

Serwer aplikacji mobilnej będzie umożliwiał pobieranie danych z serwera analitycznego, zarówno na żądanie użytkownika, jak i w ustalonych interwałach czasowych. Dane będą obejmowały:

* Informacje o obecnych i uśrednionych parametrach ula (np. temperatura, wilgotność).
* Status infrastruktury IoT.
* Powiadomienia push informujące o nieplanowanych zdarzeniach.
* Informacje o nadchodzących lub przeprowadzonych inspekcjach.

**9. Stworzenie aplikacji mobilnej**

Aplikacja mobilna zostanie zaprojektowana z użyciem technologii **cross-platform** (preferowane: **Flutter**, **Cordova** lub **React Native**) w celu zapewnienia wydajności i możliwości uruchamiania na różnych platformach mobilnych (iOS, Android). Aplikacja mobilna będzie wykorzystywać dane przetwarzane przez serwer Back-End, a jej funkcjonalności obejmą:

* **Logowanie i rejestracja użytkowników**.
* **Zarządzanie pasieką**: Podgląd danych dotyczących uli, odczytów czujników oraz statusu infrastruktury.
* **Personalizowane powiadomienia push** dotyczące stanu ula, inspekcji oraz potencjalnych zagrożeń.
* **Przegląd danych historycznych**: Użytkownicy będą mogli przeglądać dane historyczne, takie jak średnie wartości odczytów per dzień, miesiąc i rok.
* **Integracja z CMS**: Aplikacja będzie prezentować spersonalizowaną treść zarządzaną przez CMS po stronie serwera.

**10. Logika działania systemu**

**10.1 Opracowanie logiki czujników i krytycznych parametrów**

Opracowanie logiki działania systemu dla projektu na podstawie dostarczonej dokumentacji techniczno-naukowej obejmuje stworzenie dokładnej specyfikacji działania czujników IoT, w tym:

* **Dokładność pomiarowa**: Określenie dokładności pomiarowej czujników (np. temperatura, wilgotność, hałas) oraz parametrów krytycznych do badań naukowych.
* **Parametry krytyczne**: Zdefiniowanie wartości progowych dla poszczególnych parametrów środowiskowych, które będą aktywować alerty lub rekomendacje działań w systemie.

**10.2 Podział uprawnień w systemie**

System będzie obsługiwał różne role użytkowników z różnymi poziomami uprawnień, w tym:

* **Użytkownik badań**: Dostęp do odczytów z czujników oraz wizualizacji danych.
* **Administrator systemu**: Pełne uprawnienia do całym systemem.

**10.3 Interfejsy użytkowników**

System będzie obsługiwał różne interfejsy użytkownika, w tym:

* **Niezalogowany użytkownik**: Podgląd najważniejszych parametrów instalacji bez dostępu do szczegółowych danych.
* **Zalogowany użytkownik**: Dostęp do pełnych funkcji systemu, w tym przegląd danych z czujników, stan infrastruktury oraz pełna funkcjonalność zarządzania.

**10.4 Wysyłanie alertów i notyfikacji**

System będzie posiadał funkcję wysyłania automatycznych alertów do użytkowników (mail, SMS, push) w przypadku wykrycia nieprawidłowych parametrów instalacji, takich jak:

* Przekroczenie krytycznych wartości odczytów czujników.
* Nieplanowane zdarzenia związane z infrastrukturą pasieki.

**11. Zarządzanie interfejsami webowymi i mobilnymi**

Interfejsy dla aplikacji webowej i mobilnej będą obejmowały:

* **Dostosowane widoki użytkowników**: Każdy typ użytkownika (np. badacz, administrator) będzie miał dedykowany widok interfejsu, dostosowany do jego funkcji.
* **Wizualizacje danych**: Dane z czujników będą prezentowane w formie wykresów oraz tabel, co umożliwi łatwe zrozumienie stanu pasieki i infrastruktury.

### 12. ****Hosting i utrzymanie serwera bazy danych****

System wymaga wydajnego, skalowalnego i bezpiecznego serwera bazy danych, który będzie hostowany w chmurze lub na dedykowanych serwerach fizycznych, zależnie od wymagań operacyjnych i przyszłej ekspansji projektu.

#### 12.1 ****Skalowalność****

Serwer bazy danych musi być zdolny do dynamicznego skalowania, aby obsłużyć rosnącą liczbę danych zbieranych z czujników IoT i zwiększającą się liczbę pasiek podłączonych do systemu. Technologie **PostgreSQL** i **MySQL** oferują narzędzia do replikacji oraz partycjonowania danych, co umożliwia skalowanie zarówno w pionie (poprzez zwiększenie zasobów serwera), jak i w poziomie (poprzez rozproszenie obciążenia na wiele serwerów). Oznacza to, że baza danych będzie w stanie elastycznie dostosowywać się do rosnącego wolumenu danych, zapewniając niezawodność i szybkość dostępu.

#### 12.2 ****Bezpieczeństwo danych****

Ponieważ dane z czujników IoT mają krytyczne znaczenie dla operacji pasiek, konieczne jest wdrożenie zaawansowanych mechanizmów ochrony danych. Baza danych musi oferować **szyfrowanie danych w spoczynku oraz w trakcie transferu**, aby zabezpieczyć je przed nieautoryzowanym dostępem. Kluczową funkcją jest również zgodność z przepisami **RODO (GDPR)**, co obejmuje odpowiednie zarządzanie danymi osobowymi oraz mechanizmy autoryzacji i kontroli dostępu, które ograniczają możliwość manipulacji danymi wyłącznie do uprawnionych użytkowników.

#### 12.3 ****Wysoka dostępność (High Availability)****

W celu zapewnienia ciągłości pracy systemu, serwer bazy danych musi gwarantować **wysoką dostępność (HA)**, minimalizując przestoje operacyjne. **PostgreSQL** wspiera konfiguracje High Availability oraz mechanizmy **automatycznego failoveru**, które zapewniają, że w przypadku awarii jednego z serwerów, jego funkcje zostaną natychmiast przejęte przez inny serwer. Dzięki temu baza danych będzie stale dostępna, nawet w przypadku krytycznych awarii sprzętowych lub sieciowych.

### 13. ****Zarządzanie danymi i interfejs użytkownika****

System będzie umożliwiał kompleksowe zarządzanie danymi za pomocą dedykowanego interfejsu użytkownika dostępnego w aplikacji mobilnej i webowej.

#### 13.1 ****Baza danych i analiza historyczna****

Baza danych będzie przechowywać dane historyczne, co umożliwi użytkownikom analizowanie trendów, szczególnie w kontekście zdrowia kolonii pszczół. Przechowywane informacje będą obejmowały odczyty z czujników, wyniki analiz obrazu oraz parametry związane z warunkami środowiskowymi w ulach. Dane te będą dostępne w postaci raportów, które użytkownicy będą mogli filtrować według parametrów takich jak:

* Aktywność pszczół
* Wyniki analizy obrazowej z kamer na podczerwień
* Dane o populacji pasożyta Varroa czy wilgotności

#### 13.2 ****Powiadomienia i alerty****

System będzie generował powiadomienia w czasie rzeczywistym, gdy wykryje nieprawidłowości lub potencjalne ryzyko dla zdrowia pszczół. Alerty będą wysyłane za pomocą powiadomień push do aplikacji mobilnej lub wiadomości e-mail, w zależności od ustawień użytkownika. Pozwoli to pszczelarzom na natychmiastowe podjęcie działań interwencyjnych, co znacząco zmniejszy ryzyko strat w pasiekach.

#### 13.3 ****Aplikacja mobilna i webowa****

Użytkownicy będą mieli zdalny dostęp do systemu za pośrednictwem aplikacji mobilnej oraz webowej. W preferowanych technologiach do budowy interfejsu aplikacji mobilnej wykorzystane będą narzędzia **Flutter** lub **React Native**, co zapewni natywną obsługę zarówno na platformie Android, jak i iOS. Zdalny dostęp umożliwi monitorowanie danych z uli, analizę bieżących trendów i otrzymywanie rekomendacji dotyczących działań zapobiegawczych.

### 14. ****Automatyzacja działań interwencyjnych****

System nie tylko wykrywa anomalie w czasie rzeczywistym, ale również umożliwia automatyzację działań interwencyjnych.

#### 14.1 ****Automatyczne alerty****

W przypadku wykrycia krytycznych anomalii, takich jak gwałtowny wzrost liczby roztoczy Varroa lub niesprzyjające warunki do rozwoju zgnilca amerykańskiego, system automatycznie wygeneruje **alerty o wysokim priorytecie**, informując użytkownika o konieczności natychmiastowego podjęcia działań. Tego typu alerty będą wysyłane w czasie rzeczywistym, aby umożliwić szybką reakcję, minimalizując potencjalne straty.

#### 14.2 ****Inteligentne harmonogramy****

System będzie automatycznie planować harmonogramy inspekcji na podstawie historycznych danych i analizy ryzyka. Algorytmy będą analizować zgromadzone dane, identyfikując okresy, w których ryzyko wystąpienia chorób pszczół jest najwyższe, co pozwoli na efektywne zarządzanie zasobami pszczelarza. Dzięki temu inspekcje będą przeprowadzane w optymalnych momentach, kiedy są najbardziej potrzebne.

#### 14.3 ****Rekomendacje dynamiczne****

Na podstawie bieżących danych z czujników i wyników analizy, system będzie dynamicznie dostosowywał rekomendacje dotyczące leczenia i prewencyjnych działań. Oznacza to, że system będzie na bieżąco analizować zmieniające się warunki w ulach i automatycznie sugerować odpowiednie interwencje, takie jak zastosowanie chemicznych środków ochrony zdrowia pszczół lub poprawa wentylacji ula.

**15. Wdrożenie projektu i dokumentacja techniczna**

Projekt zakończy się pełnym wdrożeniem systemu, obejmującym:

* **Instalację serwerów aplikacji mobilnej i webowej**.
* **Dokumentację techniczną** zawierając

**Podział projektu na etapy – szczegółowy opis wdrożeń**

Projekt został podzielony na trzy główne etapy, które obejmują rozwój i wdrożenie poszczególnych elementów systemu monitorowania pasiek opartego na IoT, algorytmach uczenia maszynowego oraz aplikacjach webowych i mobilnych. Każdy etap kończy się pełnym wdrożeniem komponentów systemu.

**Etap 1: Pobieranie i przesyłanie danych z czujników do bazy danych**

Pierwszy etap projektu skupia się na uruchomieniu infrastruktury IoT, zbieraniu danych z czujników oraz ich przesyłaniu do centralnej bazy danych hostowanej na serwerze zewnętrznym. Kluczowe zadania w tym etapie obejmują:

1. **Pobieranie danych z czujników IoT**:
   * Dane będą pobierane z czujników zamontowanych w ulach co 1 minutę. Czujniki będą monitorować takie parametry jak temperatura, wilgotność, waga ula, aktywność pszczół oraz poziom hałasu.
   * Dane będą przypisywane do konkretnego ula w systemie, co jest kluczowe dla późniejszej analizy indywidualnych parametrów każdego ula.
2. **Kolejkowanie i wysyłanie danych**:
   * Dane zbierane co minutę będą kolejkowane w lokalnym buforze, aby następnie co 30 minut wysyłać je na serwer zewnętrzny, gdzie zostaną zapisane w centralnej bazie danych. Kolejkowanie zapewnia optymalizację przesyłu danych oraz zwiększa odporność systemu na chwilowe przerwy w łączności.
3. **Przypisanie danych do konkretnych uli**:
   * Każdy zestaw danych będzie zawierał unikalny identyfikator ula, co pozwoli na przeprowadzanie analizy dla poszczególnych uli. Dane te będą uporządkowane w bazie, umożliwiając łatwy dostęp do historii pomiarów oraz dalsze analizy.
4. **Wdrożenie i testy**:
   * Wdrożenie tego etapu będzie obejmowało instalację czujników w pasiekach, konfigurację połączeń sieciowych oraz uruchomienie mechanizmu kolejkowania i wysyłania danych na serwer zewnętrzny. Testy będą obejmowały sprawdzenie poprawności gromadzenia i przesyłania danych oraz ich zapisywania w bazie danych.
   * Ważnym elementem będzie monitorowanie działania systemu, aby wychwycić ewentualne problemy, takie jak utrata danych lub nieregularne przesyłanie.

**Etap 2: Algorytmy predykcyjne i uczenie maszynowe (ML)**

Drugi etap projektu obejmuje wdrożenie zaawansowanych algorytmów predykcyjnych oraz algorytmów uczenia maszynowego (ML), które będą automatycznie analizować dane przesyłane z czujników. W tym etapie kluczowe jest wdrożenie systemu alertów w przypadku wykrycia anomalii oraz uwzględnienie uwag z pierwszego etapu projektu.

1. **Rozwój algorytmów predykcyjnych**:
   * Algorytmy predykcyjne zostaną opracowane w celu analizy danych środowiskowych i behawioralnych zebranych z uli. Ich zadaniem będzie przewidywanie przyszłych anomalii lub niepożądanych stanów w oparciu o bieżące oraz historyczne dane.
   * Przykładami algorytmów będą modele predykcyjne oceniające wzrost populacji pasożyta Varroa destructor, zmiany w aktywności pszczół, anomalia temperatury, wilgotności i wagi ula, które mogą wskazywać na rozwój chorób lub inne zagrożenia.
2. **Uczenie maszynowe (ML)**:
   * Algorytmy uczenia maszynowego, zarówno nadzorowane, jak i nienadzorowane, zostaną wdrożone, aby automatycznie analizować dane i wykrywać anomalie. Modele ML będą trenowane na danych historycznych w celu identyfikacji wzorców sugerujących potencjalne zagrożenia zdrowotne dla kolonii pszczół.
   * **Analiza obrazu (Computer Vision)**: Algorytmy CNN (Convolutional Neural Networks) będą analizować obrazy z kamer na podczerwień, aby wykrywać obecność pasożytów Varroa oraz innych anomalii zdrowotnych pszczół.
3. **System alertów i powiadomień**:
   * System zostanie skonfigurowany tak, aby automatycznie generować powiadomienia (push, e-mail) w przypadku wykrycia nieprawidłowości w danych, takich jak nagły wzrost temperatury, spadek aktywności pszczół, czy obecność pasożytów.
   * Algorytmy będą stale monitorować dane, a w przypadku przekroczenia określonych progów, użytkownicy zostaną natychmiast poinformowani o potencjalnych problemach.
4. **Uwzględnienie uwag z pierwszego etapu**:
   * Wnioski wyciągnięte podczas testów i wdrożeń pierwszego etapu, takie jak optymalizacja częstotliwości zbierania danych, opóźnienia w transmisji czy jakość przesyłanych danych, zostaną uwzględnione podczas rozwijania algorytmów predykcyjnych i ML.
5. **Wdrożenie i testy**:
   * Wdrożenie obejmuje instalację algorytmów na serwerze, ich integrację z bazą danych oraz systemem powiadomień. Testy będą obejmować ocenę skuteczności algorytmów w wykrywaniu anomalii oraz wydajność systemu w analizie dużych ilości danych.

**Etap 3: Aplikacja webowa i mobilna**

Trzeci etap projektu zakłada stworzenie aplikacji webowej i mobilnej, które umożliwią użytkownikom interakcję z systemem, monitorowanie danych, podgląd alertów oraz modyfikowanie parametrów systemu.

1. **Tworzenie aplikacji webowej**:
   * Aplikacja webowa będzie głównym narzędziem do **wizualizacji danych**. Pozwoli użytkownikom na podgląd bieżących i historycznych odczytów z uli, a także danych analitycznych dotyczących zdrowia pszczół.
   * **Funkcjonalność podglądu danych historycznych**: Użytkownicy będą mieli dostęp do danych zebranych w ciągu dnia, miesiąca, czy roku, co pozwoli na analizę długoterminowych trendów w pasiece.
   * **Podgląd alertów i alarmów**: W aplikacji zostaną wyświetlane wszystkie alarmy generowane przez system, z informacją o czasie ich wystąpienia oraz rekomendacjach dotyczących działań naprawczych.
   * **Modyfikacja ustawień parametrów**: Użytkownicy będą mogli dostosowywać parametry dotyczące częstotliwości pomiarów, progi alarmowe oraz czas wysyłania danych na serwer.
   * **Eksport danych do XLS**: Aplikacja umożliwi eksport danych w formacie XLS, co pozwoli na dalszą analizę danych w programach takich jak Excel.
2. **Tworzenie aplikacji mobilnej**:
   * Aplikacja mobilna zostanie zbudowana w technologii **cross-platform** (np. Flutter), aby działać zarówno na systemach iOS, jak i Android.
   * Funkcjonalność aplikacji mobilnej będzie obejmować te same możliwości, co aplikacja webowa, z dodatkiem możliwości **lokalizacji platformy za pomocą GPS**. Użytkownicy będą mieli dostęp do wizualizacji danych w czasie rzeczywistym oraz powiadomień o alarmach.
3. **Integracja GPS**:
   * Aplikacja mobilna będzie integrować funkcjonalność GPS, co pozwoli na monitorowanie lokalizacji uli i pasiek w czasie rzeczywistym. To ułatwi identyfikację problematycznych lokalizacji lub weryfikację statusu pasiek w terenie.
4. **Wdrożenie i testy**:
   * Wdrożenie aplikacji webowej i mobilnej będzie obejmowało instalację na serwerach, integrację z bazą danych oraz testy użytkowe. Testy będą polegały na ocenie funkcjonalności, szybkości działania oraz łatwości obsługi dla końcowego użytkownika.

**POZOSTAŁE POSTANOWIENIA**

1. Zamawiający nie dopuszcza możliwości składania ofert częściowych ani ofert wariantowych.
2. Oferta ze strony Wykonawcy musi spełniać wszystkie wymogi stawiane w zapytaniu ofertowym i być złożona na wzorze oferty dołączonym do niniejszego zapytania.
3. Zamawiający zastrzega sobie prawo do unieważnienia postępowania na każdym etapie bez podawania przyczyny.
4. Decyzja Zamawiającego o odrzuceniu oferty jest decyzją ostateczną.
5. W przypadku, gdy wybrany Wykonawca odstąpi od podpisania umowy z Zamawiającym, możliwe jest podpisanie przez Zamawiającego umowy z kolejnym Wykonawcą, który w postępowaniu uzyskał kolejną najwyższą liczbę punktów.
6. Zamawiający jest uprawniony do poprawienia w tekście oferty oczywistych omyłek pisarskich, niezwłocznie zawiadamiając o tym danego Oferenta. W przypadku rozbieżności, co do kwoty oferty, za cenę oferty Zamawiający przyjmuje kwotę wpisaną słownie.
7. Zamawiający informuje, iż w umowie o realizację zamówienia będą zapisy:
8. Zastrzegające do **100% kar umownych na rzecz Zamawiającego** na okoliczność niewykonania lub nienależytego wykonania zobowiązania przez Wykonawcę m.in. w następujących sytuacjach:
9. Nieosiągnięcia przez Wykonawcę wskaźników realizacji umowy wskazanych przez Zamawiającego w umowie i niniejszej procedurze;
10. Realizacji przez Wykonawcę umowy wbrew ustalonym harmonogramom;
11. Nieprzedstawiania dokumentów realizacji umowy w terminach wskazanych w umowie;
12. Przedkładania w toku realizacji umowy fałszywych oświadczeń lub podrobionych, przerobionych lub stwierdzających nieprawdę dokumentów lub też popełnienie oszustwa;
13. Innych przypadków niewykonywania przez Wykonawcę umowy zgodnie z jej postanowieniami lub działań, lub zaniechań Wykonawcy uniemożliwiających Zamawiającemu prawidłowe realizowanie wsparcia.
14. Rozszerzające odpowiedzialność Wykonawcy na okoliczności, za które na mocy ustawy odpowiedzialności nie ponosi (art. 473 § 1 kc);
15. Zastrzegające Zamawiającemu możliwość potrącania naliczonych kar umownych z wynagrodzenia Wykonawcy;
16. Zastrzegające Zamawiającemu możliwość dochodzenia od Wykonawcy odszkodowania przekraczającego wysokość kar umownych, na zasadach ogólnych;
17. O przeniesieniu pełni autorskich praw majątkowych do wszelkich materiałów wytworzonych i wykorzystanych podczas realizacji umowy. Wykonawcy nie będzie przysługiwać dodatkowe wynagrodzenie z tego tytułu.
18. Zamawiający informuje, że z tytułu realizacji zamówienia publicznego przez Wykonawcę, Zamawiający ponosi pełną odpowiedzialność finansową, która przekracza określone w umowie łączącej strony wynagrodzenie Wykonawcy.
19. Zamawiający informuje, że terminy płatności wynagrodzenia Wykonawcy uzależnione będą od terminu wpłynięcia na konto Zamawiającego środków przeznaczonych na pokrycie wydatków związanych z realizacją umowy na etapie, w którym uczestniczył Wykonawca i mogą ulegać opóźnieniom.
20. Zamawiający przewiduje możliwość wypłacania wykonawcy zaliczek w trakcie realizacji zamówienia, uzależnionych m.in. od postępu i prawidłowości realizacji zamówienia oraz kompletności i poprawności przedkładanych przez wykonawcę dokumentów. Jednakże w każdym przypadku decyzja o wypłacie zaliczek należy wyłącznie do Zamawiającego.
21. Zamawiający informuje, iż w umowie o realizację zamówienia znajdą się zapisy przewidujące możliwość dokonywania istotnych zmian postanowień umowy w zakresie:
22. Terminu realizacji umowy;
23. Harmonogramu realizacji umowy;