Fundusze Europejskie
Pomoc Techniczna

Unia Europejska
Fundusz Spójności

Human Smart Cities – Siechnice

Doświadczenia i dobre praktyki współpracy między JST oraz instytucjami badawczymi

Gmina Siechnice

Akademia Górniczo-Hutnicza w Krakowie

AGH

Projekt został zrealizowany w ramach konkursu „Human Smart Cities. Inteligentne miasta współtworzone przez mieszkańców", współfinansowanego przez Unię Europejską ze środków Funduszu Spójności w ramach Programu Operacyjnego Pomoc Techniczna 2014-2020, Ministerstwa Funduszy i Polityki Regionalnej.

W opracowaniu wykorzystano następujące materiały w oparciu o licencje Creative Commons:

- Andrzej Błaszczak – Praca własna, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=98372125
- Alex Proimos, Sydney, Australia – CrossWalk, CC BY 2.0, https://commons.wikimedia.org/w/index.php?curid=25650842
- IMarcoHerrera – Praca własna, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=65280240
- kennymatic – https://www.flickr.com/photos/kwl/2743788633/, CC BY 2.0, https://commons.wikimedia.org/w/index.php?curid=10057066,
- Victor Grigas – Praca własna, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=20348454,
- Lothar Neumann - Praca własna, CC BY-SA 2.0 de, https://commons.wikimedia.org/w/index.php?curid=7512264
- Michał Kud - Praca własna, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=114932256
- Tobias 'ToMar' Maier - Praca własna, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=50065987
- Pixabay - https://pixabay.com/photos/work-typing-computer-notebook-731198/, CC0, https://commons.wikimedia.org/w/index.php?curid=99618818
- Sora Shimazaki, https://www.pexels.com/@sora-shimazaki/

Wydanie trzecie, Listopad 2023

Spis treści

Wprowadzenie

1. O projekcie

Projekt „Realizacja pilotażowego projektu wdrożenia rozwiązań Smart City dla Gminy Siechnice z uwzględnieniem strategii rozwoju gminy" jest realizowany w formule partnerstwa pomiędzy Gminą Siechnice jako Liderem Projektu a Akademią Górniczo-Hutniczą w Krakowie jako Partnerem naukowym.

Celem projektu było pilotażowe wdrożenie przez Gminę szeregu rozwiązań *smart city*, obejmujących:

- inteligentne oświetlenie,
- monitoring jakości powietrza,
- kamery zdolne do wykrywania zdarzeń i obiektów,
- ładowarki EV.

Podstawowym wyzwaniem dla prac badawczych było natomiast przekształcenie danych pochodzących z różnych systemów elektronicznych w informacje użyteczne zarówno do podejmowania decyzji zarówno na szczeblu decyzyjnym miasta/gminy jak szeroko rozumianej partycypacji społecznej.

Narzędzia pozwalające na osiągnięcia tego celu to przede wszystkim narzędzia oraz procedury analizy danych, ale również opracowana we współpracy z Gminą autorska aplikacja, której celem jest usprawnienie komunikacji pomiędzy Władzami Gminy a jej Mieszkańcami. Jej funkcjonalności pozwalają zarówno na informowanie Mieszkańców o działaniach prowadzonych na terenie Gminy, wraz z możliwością wglądu w działanie wdrożonych nowoczesnych systemu oraz publikowania popartych danymi materiałów podnoszących świadomość, jak i na zbieranie od Mieszkańców opinii, uwag i życzeń w sposób utrzymujący ich semantykę, a więc pozwalający na ich bezpośrednie wykorzystanie w procesach decyzyjnych.

Jednym z najważniejszych najważniejszym celów projektu było spójne zaprezentowania części ze zgromadzonych i przetworzonych danych w postaci aplikacji, która umożliwi partycypację społeczną mieszkańców w zarządzaniu miastem. Należy jednak podkreślić również wagę efektów projektu stanowiących osiągnięcia naukowe oraz dobre praktyki prowadzenia badań naukowych, szczególnie tych związanych z analizą i obróbką danych.

2. Uczestnicy projektu

2.1 Gmina Siechnice

W Polsce jest ok. 2500 Jednostek Samorządu Terytorialnego. Wśród nich unikalną społecznością jest gmina Siechnice. Znajduję się ona w Województwie Dolnośląskim przy granicy w miastem Wrocław. Siechnice są miejscem unikalnym w całej Polsce- Jej najbardziej charakterystyczną cechą jest dynamiczny przyrost liczby ludności związanymi z ruchami migracyjnymi oraz przyrostem naturalnym. Zgodnie z ostatnimi danymi rządu polskiego Siechnice – są najmłodsza pod względem wieku mieszkańców gminą w Polsce. Na przestrzeni lat IV kw. 2005–IV. kw. 2022 na terenie gminy nastąpił wzrost liczby mieszkańców – ok. 80,55% procenta (13 373 os. do 24 145 os.). Tak dynamiczny wzrost populacji pociągnął za sobą szereg wyzwań związanych z budową i modernizacją instruktury komunalnej – liniowej i kubaturowej, oraz jednocześnie niesie za sobą szereg wyzwań związanych z procesem zarządzania na poziomie gminy oraz komunikacji na styku gmina – mieszkańcy. Młodzi ludzie wychowani w kulturze dynamicznego przepływu informacji oczekują takiego samego podejścia ze strony władz Siechnic.

2.2 Akademia Górniczo-Hutnicza

Akademia Górniczo-Hutnicza w Krakowie (skrótowo AGH) to polski uniwersytet publiczny, mieszczący się w Krakowie. AGH została założona w 1913 roku, a otwarcie nastąpiłó w 1919 roku. AGH skupia się na innowacyjnych technologiach, a jej profil badawczy obejmuje również dziedziny inżynieryjne, nauki ścisłe, nauki o Ziemi i nauki społeczne.

AGH jest jednyą z 10 polskich uczelni wyższych, które otrzymały tytuł uniwersytetu badawczego. W jego skład wchodzi 16 wydziałów, centrum badawcze – Akademickie Centrum Materiałów i Nanotechnologii AGH, a także inne ośrodki dydaktyczne i jednostki. Oferuje trzy poziomy kształcenia: studia pierwszego stopnia, studia drugiego stopnia i studia trzeciego stopnia (szkoły doktorskie). Uczelnia kształci ponad 20 000 studentów i zatrudnia niemal 2 000 pracowników naukowych (w tym ponad 200 profesorów i ponad 500 doktorów habilitowanych).

W międzynarodowych rankingach, takich jak The Center for World University Rankings 2022-2023 czy Akademicki Ranking Światowych Uniwersytetów 2021, AGH regularnie zajmuje pierwsze miejsce wśród polskich uniwersytetów technicznych.

3. Efekty projektu

W dalszej części opracowania przedstawiono efekty projektu, które leżą w kilku obszarach:

- wdrażania inteligentnych rozwiązań infrastruktury miejskiej,
- rozumienia danych i wdrożenia ich w procesy decyzyjne,
- przekazywania informacji oraz danych Mieszkańcom,
- zwiększenia partycypacji Mieszkańców, poprzez stworzenie wielu przystępnych kanałów zbierania ich opinii.

Problematyka wdrażania inteligentnych rozwiązań w jednostkach samorządu terytorialnego jest istotna, gdyż rozwiązania te są bardzo zróżnicowane i należą do kilku kategorii, lecz bardzo często charakteryzują się wspólnymi cechami:

- możliwość rejestrowania danych, najczęściej o charakterze pomiarowym,
- wyposażenie w moduły komunikacyjne, pozwalające na zdalny dostęp do rejestrowanych danych oraz sterowanie działaniem urządzeń,
- dostarczenie wraz z towarzyszącą aplikacją, pozwalającą na konfigurację urządzeń i śledzenie jej pracy.

Istotnym jest, aby praca z wdrożonymi urządzeniami nie była ograniczona do własnościowej aplikacji producenta – innymi słowy, aby możliwa była agregacja danych pochodzących z danego systemu w jednym, zunifikowanym repozytorium. Warunkiem koniecznym jest tutaj odpowiednia *otwartość* systemu, natomiast w dalszej kolejności należy zadbać o przygotowanie odpowiednich struktur do ich gromadzenia, a także integrację systemów przy pomocy odpowiednich modułów. Zagadnienia te omówiono szerzej w części II (rozdziały 4 oraz 5).

Kolejnym zagadnieniem, stanowiącym inherentny element programu *Human Smart Cities*, jest włączenie mieszkańców w życie miasta. Komunikacja z nimi powinna być dwukierunkowa – oczywistym elementem jest informowanie mieszkańców np. o projektach prowadzonych w mieście, lecz równie cennym elementem jest wiedza zbierana od użytkowników. Można wręcz zaryzykować stwierdzenie, iż stanowi ona naturalne rozszerzenie danych sensorycznych pozyskiwanych z urządzeń IoT. Uwarunkowania związane z tym zagadnieniem, jak również autorskie rozwiązanie opracowane przez naukowców AGH

przedstawiono w części III (rozdziały 6 i 7).

Ostatnią część opracowania (część IV) poświęcono aspektom badawczym, a także utrzymaniu współpracy po zakończeniu realizacji projektu. Przedstawia ona wariant koncepcji „żyjącego laboratorium" (ang. *living lab*; rozdział 8), dostosowany do specyfiki współpracy pomiędzy JST a instytucjami naukowymi, a także przedstawia ramy pozwalające na jej sformalizowanie. W części tej przedstawiono także niezwykle istotny, a często zaniedbywany aspekt organizacji samych badań naukowych (rozdział 9 , szczególnie związanych z pracą z danymi, w sposób gwarantujący płynną współpracę w obrębie samego zespołu badawczego oraz efektywne wykorzystanie efektów badań. Rozdział 10 omawia wypracowae w ramach projektu metody analizy danych, które stanowią jeden z najważniejszych wyników naukowych, opisany już w kilku wydanych oraz przygotowywanych publikacjach o zasięgu międzynarodowym.

II Wdrażanie inteligentnych rozwiązań w JST

4. Proces wdrożenia

4.1 Przygotowanie do procesu wdrożenia

Realizacja Projektu oraz wcześniejsze doświadczenia zespołu związane z wdrażaniem nowoczesnych technologii w JST zaowocowały sformułowaniem szeregu zadań koniecznych do wykonania przed przystąpieniem do inwestycji, podsumowanych m.in. w poz. [6]:

1. identyfikacja i optymalizacja procesów w urzędzie miejskim, które mają mieć swoje odzwierciedlenie w systemie,
2. zdefiniowanie wymagań funkcjonalnych tej aplikacji (obejmują m.in. obliczenia, przetwarzanie danych i inne specyficzne funkcje, które określają, co system i jego składowe mają osiągnąć),
3. określenie zakresu i źródeł danych pozyskiwanych z innych systemów miejskich (jakie konkretnie dane znajdujące się w już działających systemach miejskich będzie musiała wykorzystywać aplikacja, aby realizowała zakładane funkcjonalności),
4. ustalenie zakresu danych eksportowanych do innych systemów i umieszczanych w zbiorze danych otwartych,
5. podjęcie decyzji dot. praw dostępu do aplikacji (na jakich warunkach następowało będzie korzystanie z zasobów aplikacji przez inne systemy działające w mieście).

Jednym ze sposobów na uzyskanie interoperabilności i otwartości systemów jest zapewnienie odpowiednich wymagań dotyczących interfejsów wdrażanych systemów i aplikacji, co szerzej omówiono w seklcji 4.2. Nieodpowiednie spełnienie tych wymagań niesie za sobą ryzyka, co opisują autorzy [6]:

> Warto zwrócić większą uwagę na specyfikację zakresu i sposobu danych w SWZ – dołożenie wymagania wyeksportowania dowolnych danych z reguły nie podnosi kosztu tej aplikacji w chwili jej zamawiania, natomiast stanowi znaczne koszty, gdy zażyczymy sobie tej usługi po jakimś czasie. Zwróćmy uwagę, że koszt modyfikacji systemu nie podlega już ograniczeniom wynikającym z procesu przetargowego, a wprost przeciwnie – jest dyktatem monopolisty. Oznacza to, że każdorazowe rozbudowanie systemu jest całkowicie uzależnione

od pierwotnego dostawcy i miasto jest skazane na przyjęcie jego warunków cenowych.

4.2 Zapewnienie otwartości i interoperabilności systemów

API (Application Programming Interface) to interfejs programowania aplikacji, który umożliwia komunikację między różnymi systemami i usługami. W systemach sensorycznych i/lub sterowania, zaliczanych do kategorii IoT (Internet of Things) API służą do wymiany danych pomiędzy urządzeniami, aplikacjami oraz innymi systemami.

Aby zapewnić interoperabilność i łatwą wymianę danych z innymi systemami, API dla systemów IoT powinny spełniać następujące ogólne wymagania:

1. **Otwarte standardy:** API powinny korzystać z otwartych standardów, takich jak HTTP, HTTPS lub WebSocket, aby umożliwić komunikację z różnymi urządzeniami i systemami.
   Gwarantuje to możliwość nawiązania połączenia, niezależenie od użytech technologii implementacji systemu IoT oraz samej aplikacji. Jednocześnie zapewnia możliwość dalszego rozwoju aplikacji, gdyż niezależnie od rozwiązań informatycznych komunikacja pozostaje niezmieniona, zapewniając kompatybilność i tzw. odporność na przyszłość. Zatem modyfikacje lub zmiana aplikacji w przyszłości nie będzie wymagała zmiany systemu IoT.
2. **Dokumentacja:** API powinny mieć dobrze napisaną i aktualną dokumentację, która umożliwi innym programistom łatwe korzystanie z nich.
   W ramach wybranego protokołu komunikacyjnego występuje zwykle szereg metod, z pomocą których przeprowadzana jest właściwa komunikacja. Jest istotne, aby wszystkie wymagane etapy tej komunikacji były dostatecznie opisane, wraz z użytymi metodami, jak również zaprezentowane z wykorzystaniem przykładów. Przykłady winny zawierać przesyłane dane, żądania i spodziewane odpowiedzi, tak aby mieć pewność jak skutecznie zaimplementować proces komunikacji. Udokumentowane również winny być zależności czasowe: w jakim przedziale czasowym należy spodziewać się odpowiedzi, kiedy odpowiedź jest ważna, czy też kiedy jest błędem. Należy zadbać aby zakres API pokrywał się z zakresem funkcjonalność systemu IoT, tak aby zarówno dane z niego pochodzące, jak również metody manipulacji samymi urządzeniami mogły być w całości zarządzane za jego pośrednictwem. Pewnym ryzykiem jest praktyka wykorzystania częściowego API, z uwagi np. na ograniczenia licencyjne, techniczne, czy też komunikacyjne po stronie dostawcy systemu IoT. Może to prowadzić do późniejszych problemów z dostępem do danych, których właścicielem powinien być zamawiający, ograniczeniami w zarządzaniu urządzeniami, zubożeniem funkcjonalności, czy też koniecznością ponoszenia ukrytych kosztów, np. za utrzymanie lub udostępnienie API.
3. **Bezpieczeństwo:** API powinny być zabezpieczone przed atakami i nieupoważnionym dostępem, np. poprzez stosowanie szyfrowania danych i uwierzytelniania użytkowników.
   Polityka bezpieczeństwa powinna być transparentna i udokumentowana. Również udokumentowane winny być wszelkie jej zależności od systemów trzecich (np. służących do uwiarygodnienia certyfikatów kryptograficznych) oraz przepływy danych do i z systemów trzecich (jakie dane i gdzie są przesyłane).
   Dobrą praktyką jest wymagania audytu bezpieczeństwa. Audyt taki powinien być przeprowadzony dla aktualnej wersji oraz implementacji API przez niezależny podmiot mający doświadczenie w zakresie cyberbezpieczeństwa.

Bezpieczeństwo API jest szczególnie istotne w przypadku zarządzania infrastrukturą krytyczną, transportem, czy też bezpieczeństwem publicznym. Głównym celem jest minimalizacja skutków ewentualnego zaniedbania, czy też celowych działań osób trzecich. Należy również zwrócić uwagę na bezpieczeństwo danych wrażliwych lub osobowych, jeżeli są one udostępniane bądź przesyłane za pomocą API.

4. **Skalowalność:** API powinny być przystosowane do dużych obciążeń i rosnącego ruchu, tak aby nie spowalniać działania systemu ani nie ograniczać możliwości jego rozwoju.
   Zapewnienie skalowalności dostarcza informacji jak bardzo można rozwinąć system w przyszłości, ile urządzeń może obsłużyć i jakie są z tym związane koszty ewentualnej rozbudowy. Skalowalność również powinna być udokumentowana i poparta testami wydajnościowymi zawierającymi wyniki badania przepustowości komunikacji, jak również opóźnień. Jeżeli instalacja jest zrealizowane na infrastrukturze teleinformatycznej zamawiającego należy również określić parametry użytego sprzętu i oprogramowania oraz wytyczne jak proponowane rozwiązanie przeskalować w przypadku osiągnięcia granicy wydajności. Tego rodzaju analizy są niezbędne do oszacowania kosztów rozwoju w przyszłości.
5. **Struktura atrybutów:** Model danych powinien mieć jasno zdefiniowaną strukturę, w której atrybuty są grupowane w odpowiednie sekcje i podsekcje, tak aby były łatwe do odczytania i interpretacji.
   Model danych winien być oparty o dobrze zdefiniowane i ugruntowane standardy przechowywania i przesyłania danych np. JSON, czy też XML. Projekt modelu powinien wziąć pod uwagę różne kategorie pogrupowane w sekcje np. słownik urządzeń, wartości odczytów, sterowanie itp. W ogólnym przypadku struktura modelu danych może być bardziej złożona w postaci drzewa lub grafu.
6. **Semantyka:** Atrybuty powinny mieć jasno zdefiniowaną semantykę, tzn. ich znaczenie powinno być jednoznaczne i zrozumiałe dla różnych urządzeń i systemów.
   Semantyka atrybutów winna być dokładnie opisana w dokumentacji. Dobrą praktyką jest stosowanie odpowiednich nazw, które mają znaczenie i są zwykle opisane w sposób czytelny za pomocą wyrazów jeżyka naturalnego (często w języku angielskim). W przypadku nazw składających się z więcej niż jednego wyrazu należy konsekwentnie stosować spójną konwencję np. CamelCase, snake_case, kebab-case itp. Równie istotna jest semantyka wartości atrybutów tj. rozróżnienie danych poprawnych, i niepoprawnych, błędów odpowiedzi, bądź też błędów z uwagi na przekroczenie limitów czasowych. Wiąże się to również z typami i jednostkami danych opisanymi poniżej.
7. **Typy danych:** Atrybuty powinny mieć odpowiednie typy danych, tak aby ich wartości były jednoznacznie interpretowane przez różne systemy.
   Szczególnie w przypadku typów złożonych, bądź też reprezentujących dane w różnych formatach należy dokładnie określić te formaty. Przykładowo jeżeli typem danych dla reprezentacji znacznika czasowego jest data i godzina należy określić jej format. Formatem może być reprezentacja tekstowa standardu ISO 8601. Wybierając format trzeba również określić dopuszczalną rozdzielczość i dokładność typów danych, jak również dopuszczalne zakresy wartości. W przypadku wartości symbolicznych zakresem wartości będzie zbiór symboli.
8. **Jednostki:** Atrybuty powinny mieć jasno zdefiniowane jednostki, tak aby ich wartości były jednoznacznie interpretowane przez różne systemy.
   Dzięki temu można uniknąć błędów jakościowych (np. *wat* zamiast *watogodzina*) jak i ilościowych (np. *wat* zamiast *miliwat*).

4.3 Uwarunkowania formalne

Gmina Siechnice jako lider projektu by możliwe było zrealizowanie zakresu badawczego przez partnera projektu tj. Akademie Górniczo-Hutniczą w Krakowie, koordynowała przeprowadzenie prac instalacyjnych, montażowych oraz budowalnych. Wykonanie tych prac umożliwiło uruchomienie środowiska pilotażowego i co za tym idzie budowę środowiska badawczo-obliczeniowego, integrację systemów, opracowanie koncepcji integracji danych i wartości dodanych, weryfikacji danych, opracowanie i wdrożenie systemu integrującego dane o mieście oraz budowę narzędzi do naukowej analizy danych.

Wykonawcy prac instalacyjnych, montażowych czy też budowalnych zostali wyłonieni na podstawie przepisów ustawy z dnia 11 września 2019 r. – Prawo zamówień publicznych lub też wewnętrznych regulacji obowiązujących w Urzędzie Miejskim w Siechnicach.

Pracownicy Urzędu Miejskiego w Siechnicach zaangażowani w realizację projektu opracowali dokumentację niezbędną do przeprowadzenia postepowań o udzielenie zamówień publicznych, następnie wyłoniono wykonawców i wykonano instalację systemu kamer cyfrowych oraz czujników jakości powietrza. Ponadto utworzono infrastrukturę w zakresie inteligentnego oświetlenia tj.: zrealizowano roboty budowalne związane z wykonaniem układu liniowego oświetlenia smart w przebiegu ulicy Henryka III w Siechnicach. Dodatkowo w ramach projektu wykonano sterowniki dla oświetlenia smart w ramach ulic: Sportowej, Piłsudskiego, Gimnazjalnej, Jana Pawła II, Kolejowej i Świętego Krzyża w Siechnicach.

W ramach odrębnego projektu realizowanego na terenie gminy, dostarczono i zainstalowano ładowarkę e-car. Dane uzyskane w tym zakresie wykorzystano do analizy.

5. Gromadzenie danych

5.1 Model danych dla systemów pomiarowych

Aby zapewnić możliwość elastycznego gromadzenia wszelkich danych produkowanych przez już wdrożone systemy pomiarowe, a także umożliwić łatwą rozbudowę systemu o nowe rodzaje czujników, zaprojektowano elastyczny schemat danych, zapewniający jednocześnie spójność danych i poprawność ich typów. Schemat bazy danych pomiarowych przedstawiono na rys. 5.1.

Tabela `iot_devices` reprezentuje poszczególne urządzenia w systemie Internetu rzeczy (IoT). Posiada następujące kolumny:

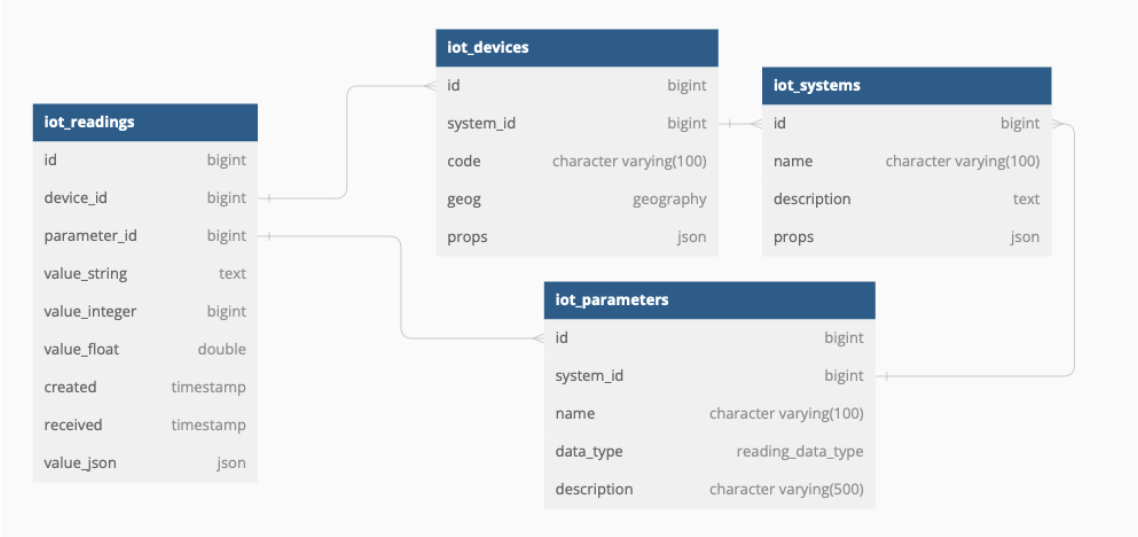
- `id`: unikalny identyfikator urządzenia.
- `system_id`: identyfikator systemu IoT, do którego należy urządzenie.
- `code`: kod reprezentujący urządzenie.
- `geog`: położenie geograficzne urządzenia.
- `props`: obiekt JSON zawierający dodatkowe informacje o urządzeniu.

Tabela `iot_parameters` przedstawia parametry mierzone przez urządzenia w systemie IoT. Posiada następujące kolumny:

- `id`: unikalny identyfikator parametru.
- `system_id`: identyfikator systemu IoT, do którego należy parametr.
- `name`: nazwa parametru.
- `data_type`: typ danych odczytów parametru. Może to być jedna z następujących wartości: „string”, „integer”, „float” lub „json”.
- `description`: opis parametru.

Tabela `iot_readings` reprezentuje odczyty wykonane przez urządzenia w systemie IoT. Posiada następujące kolumny:

- `id`: unikalny identyfikator odczytu.
- `device_id`: identyfikator urządzenia, które wykonało odczyt.
- `parameter_id`: identyfikator parametru, który został zmierzony podczas odczytu.
- `value_string`: wartość ciągu odczytu, jeśli dotyczy.
- `value_integer`: wartość całkowita odczytu, jeśli dotyczy.



Rysunek 5.1: Diagram schematu danych pomiarowych

- `value_float`: wartość zmiennoprzecinkowa odczytu, jeśli dotyczy.
- `created`: znacznik czasu wskazujący, kiedy dokonano odczytu.
- `received`: znacznik czasu wskazujący, kiedy odczyt został odebrany przez system.
- `value_json`: wartość JSON odczytu, jeśli dotyczy.

Tabela `iot_systems` reprezentuje systemy IoT, do których należą urządzenia i parametry. Posiada następujące kolumny:

- `id`: unikalny identyfikator systemu IoT.
- `nazwa`: nazwa systemu IoT.
- `description`: opis systemu IoT.
- `props`: obiekt JSON zawierający dodatkowe informacje o systemie IoT.

5.2 Gromadzenie danych z wdrożonych systemów IoT

Aby wdrożone systemy IoT nie były ograniczone wyłącznie do współracy z dedykowanymi im aplikacjami (zazwyczaj dostarczanymi przez producentów), konieczne jest aby posiadały one odpowiednie interfejsy programistyczne, co szerzej opisano w sekcji 4.2.

Docelowo, dane powinny trafiać do spójnych struktur opisanych w sekcji 5.1. Gromadzenie danych z systemów satelitarnych powinno być wykonywane regularnie, na bieżąco.

Wykorzystano w tym celu program Cron. Pozwala on na automatyzację wykonywania określonych zadań w określonych przez użytkownika momentach. Umożliwia wykonywanie skryptów w tle, co pozwala na wykonywanie różnego rodzaju operacji bez konieczności bezpośredniego udziału użytkownika. Może być ustawiany tak, aby wykonywał określone zadania w danej chwili, w określonych odstępach czasowych lub w określonych dniach tygodnia lub miesiąca. Inne niż pobieranie danych jego zastosowania związane są z często z automatyzacją rutynowych zadań, takich jak backup danych, czyszczenie plików tymczasowych czy aktualizacja oprogramowania.

Wszystkie komponenty przygotowano w języku Python, aby utrzymać spójność logiki i ułatwić późniejszą rozbudowę aplikacji. Każdy z modułów stanowi swoisty „pomost” pomiędzy interfejsami API wdrożónych systemów a jednolitą strukturą danych w aplikacji.

Specyfika każdego z modułów jest więc związana z konstrukcją interfejsu danego sys-

temu, lecz realizowane cele są w większości przypadków zbliżone i obejmują następujące fazy:

- uwierzytelnianie – uzyskanie dostępu do systemu strony trzeciej przez moduł pobierający dane, zazwyczaj w oparciu o dane uwierzytelniajace (ang. *credentials*) w postaci nazwy użytkownika oraz hasła lub klucza API,
- enumeracja urządzeń – pobranie listy urządzeń (np. czujników, lamp, kamer) z systemu; na tym etapie możliwe jest także wykrycie nowych urządzeń i stworzenie dla nich odpowiednich struktur w repozytorium danych, z ewentualnym udziałem użytkownika (np. dla określenia lokalizacji urządzenia, jeżeli nie jest ona udostępniana przez interfejs),
- gromadzenie odczytów – gdzie oprócz oczywistego pobrania danych ze zdalnego systemu i złożenia ich w lokalnej bazie danych istotne jest też odfiltrowanie wartości pobranych przy poprzedniej sesji integracyjnej.

Istotnym aspektem jest też późniejsza rozszerzalność systemu – oprócz odpowiednio skalowalnej struktury danych ważne jest stworzenie ram dla łatwego definiowania modułów integrujących ją z kolejno wdrażanymi systemami IoT.

II Zaangażowanie mieszkańców

6. Komunikacja z mieszkańcami

6.1 Metody efektywnej komunikacji z mieszkańcami

Komunikacja to proces, który decyduje o sprawności i efektywności firmy, instytucji lub organizacji. Komunikacja może dotyczyć poszczególnych komponentów organizacji takich jak działy, zespoły, grupy robocze, ale także, a może przede wszystkim może dotyczyć interesariuszy zewnętrznych (indywidualnych i/lub grupowych). W przypadku jednostek samorządu terytorialnego klienci mogą być dość zróżnicowani, od najbardziej licznych osób prywatnych poprzez organizacje otoczenia gospodarczego, inne jednostki publiczne oraz różnej skali firmy. Dobre i wydajne mechanizmy komunikacji wzmacniają zdolność jednostki publiczne do sprostania rosnącym wymaganiom interesariuszy.

Samorządy muszą szukać szybkich, wydajnych i niedrogich kanałów komunikacji. Jednym z takich mediów jest oczywiście sieć Internet. Przewagą Internetu nad innymi kanałami komunikacji jest jego relatywnie niski koszt, potencjalnie nieograniczony zasięg oraz szybka, dwukierunkowa wymiana informacji.

6.1.1 Metody i platformy komunikacyjne

Internet jest jednak tylko pewnego rodzaju medium, które umożliwia dopiero tworzenie platform komunikacyjnych, stanowiących dopiero o efektywności i opłacalności tego medium, serwujących określone usługi. W literaturze przedmiotu [12, 9] można odnaleźć kategoryzację mediów społecznościowych w świecie biznesu zawierającą:

- blogi, podcasty, wideocasty (treści multimedialne, najczęściej wideo tworzone przez użytkowników platformy),
- portale społecznościowe i wirtualne światy (przestrzeń dla komunikacji użytkowników),
- strony typu wiki oraz platformy, poradniki otwarte bazujące na paradygmacie open-source
- fora dyskusyjne, platformy oceny, portale dla hobbystów
- RSS i widżety (wpływające na przyśpieszenie konsumpcji informacji).

W pracy [12] autorzy wprowadzają również inną klasyfikację dzielącą sposób ko-

munikacji na komunikację bezpośrednią – jeden do jednego (ang. peer-to-peer) oraz „rozgłoszeniową" (jeden do wielu, ang. broadcast), wyróżniając w tej ostatniej kategorii komunikację z informacją zwrotną (komunikacje dwukierunkową) i bez informacji zwrotnej (komunikacje jednokierunkową).

W poszczególnych kategoriach autorzy prac [12, 5] wymieniają następujące metody/rodzaje aktywności komunikacyjnych:

1. Komunikacja bezpośrednia (peer to peer):
    1. e-mail (często wykorzystywany jednak stwarzający problemy natury prawnej, często jego uzy),
    2. e-bok (elektroniczne biuro obsługi klienta, skierowane przede wszystkim na komunikacje z interesariuszami zewnętrznymi),
    3. ePUAP (platforma przekazywania elektronicznych informacji pomiędzy różnymi interesariuszami w sposób oficjalny),
    4. sms, mms, voice message
    5. faks
2. Komunikacja rozgłoszeniowa jednokierunkowa:
    1. strony internetowe (warto zwrócić uwagę, że strony mogą być statyczne tj. tylko serwujące informacje statyczną, jak i dynamiczne / interaktywne pozwalające serwować informacje kontekstową. Tym drugim bliżej do aplikacji internetowych)
    2. systemy publikacji materiałów multimedialnych (audio i video, VOD, Vimeo, Youtube etc.),
    3. systemy powiadamiania sms lub e-mail (to funkcjonalność zwykle wbudowana w istniejące i działające aplikacje),
    4. biuletyny informacyjne (autorzy [12] dodają, iż mają mieć charakter periodyczny),
    5. newslettery,
    6. Biuletyn Informacji Publicznej – BIP (uregulowany prawem zbiór informacji na temat wszystkich obszarów działalności jednostki publicznej).
3. Komunikacja rozgłoszeniowa dwukierunkowa:
    1. media społecznościowe (Facebook, Twitter etc.)
    2. sondaże / ankiety
        1. proste ankiety
        2. metody rankingowe
    3. konsultacje społeczne
        1. zbieranie opinii o charakterze (najczęściej) jakościowym a nie ilościowym.

Powyższa lista stanowi pewien katalog metod komunikacji elektronicznej. Wspólną platformą dla większości z nich jest Internet (poza komunikacją bezpośrednią i ew. mediami społecznościowymi) z dokładniej taka lub inna forma strony / aplikacji internetowej. Tu jednak warto wspomnieć o nieco lekceważoną przez instytucje publiczne platformie jaką stanowią aplikacje internetowe na urządzenia mobilne.

Aplikacje takie stanowią wygodną i dostosowaną do środowiska przetwarzania informacji klienta alternatywę komunikacji jedno lub dwustronnej, bezpośredniej i rozgłoszeniowej. Przykładem takich aplikacji mogą być aplikacje takke jak mObywatel czy mojeIKP dostępne zarówno dla systemów IOS jak i Android (Rys. @fig:app-logos).

Zdecydowanie się na konkretną platformę komunikacyjną oraz metodę komunikacji wymaga zapewnienia odpowiedniej jakości oferowanej usługi komunikacyjnej w odpowiednio długim okresie czasu. W przypadku braku zapewnienia jakości takiej usługi, zostanie ona porzucona przez użytkowników i zysk z jej wprowadzenia będzie niewielki. W kontekście stron (aplikacji) internetowych autorzy pracy [12] wymieniają:

Rysunek 6.1: Logotypy aplikacji mObywatel, MojeIKP

- nowoczesność i atrakcyjność strony WWW (ang. user experience)
- częstotliwość publikowania informacji na stronie WWW,
- czas ładowania strony,
- jakość kodu strony głównej serwisu
- informacja o polityce dotyczącej tzw. ciasteczek,
- jakość interfejsu użytkownika – łatwy dostęp do oferowanych przez aplikację internetową funkcji
- integracja z mediami społecznościowymi oraz usługami typu ePUAP
- multimedialność oferowanych zasobów

Do tego zbioru dodać należy:

- wprowadzanie częstych aktualizacji bezpieczeństwa strony WWW
- zapewnienie wsparcia technicznego dla aplikacji WWW, szczególnie w trakcie awarii systemu
- zapewnienie wsparcia użytkownikom, którzy albo korzystają z aplikacji pierwszy raz albo mają problem z korzystania strony WWW
- wsparcie dla różnych przeglądarek stron internetowych
- wsparcie dla szyfrowania komunikacji tj. protokołu https (w tym wsparcie i aktualizacje certyfikatu uwierzytelniającego)

Wykorzystanie jako platformy komunikacyjnej aplikacji mobilnej, choć często bardziej atrakcyjne dla użytkownika (klienta, mieszkańca) również stawia wymagania związane z zapewnieniem odpowiedniej jakości usługi. Zaliczyć do nich można:

- wprowadzanie aktualizacji nadążających za aktualizacjami systemu operacyjnego pod którego kontrolą działa urządzenie mobilne
- zapewnienie wysokiej jakości interfejsu aplikacji
- zadbanie o responsywność aplikacji – istotne w kontekście komunikacji sieciowej pomiędzy urządzeniem mobilnym a serwisem internetowym dostarczającym dane do aplikacji
- zapewnienie bezpieczeństwa informatycznego oraz prywatności użytkownika
- zapewnienie wsparcia technicznego działania serwisu dostarczającego dane
- zapewnienie wsparcia użytkownikom nie radzącym sobie z obsługą (bądź instalacją) aplikacji.

6.1.2 Funkcje komunikacji

Dyskusja nad formą (lub formami) jaką ma przybrać komunikacja pomiędzy jednostką, firmą, instytucją musi zostać poprzedzona zdefiniowaniem funkcji oraz celu (celów) komu-

nikacji. W szczególności zatem potrzeba zdefiniować jaką role ma pełnić owa komunikacja i w jaki sposób wpisuje się ona w procesy biznesowe instytucji. Na przykład czy wymagany czas reakcji / odpowiedzi ze strony mieszkańca / interesariusza jest ograniczony (np. przez ustawę) bo stanowi element postępowania administracyjnego czy też nie. Trzeba też zdefiniować spodziewany rezultat takiej komunikacji:

- ranking (np. priorytetyzacja budżetu obywatelskiego)
- zalecenia, opinie (konsultacje społeczne)
- załatwienie sprawy urzędowej
- kontakt z inspektorem
- itd.

Następnie zdefiniować, czy komunikacja ma być rozgłoszeniowa, interaktywna, bezpośrednia, a w kolejnym kroku dobrać właściwą platformę realizacji komunikacji np.: kontakt bezpośredni: telefon / sms itd., aplikacja internetowa, media społecznościowe, forum dyskusyjne, wreszcie aplikacja na urządzenia mobilne.

Dopiero w kontekście odpowiedzi na te wszystkie pytania można próbować określić na ile dany sposób komunikacji z interesariuszami / mieszkańcami będzie adekwatny oraz będzie zapewniał odpowiednio wysoką jakość.

6.1.3 Przyjemność komunikowania się

Niezależnie od zdefiniowanego sposobu komunikacji obie strony biorące udział w wymianie informacji powinny być z tej komunikacji zadowolone. Tj. zarówno pracownicy instytucji publicznej / samorządowej powinni widzieć korzyść z użycia wybranej formy/platformy komunikacyjnej jak i taką korzyść powinni dostrzegać interesariusze zewnętrzni. Punktem wyjścia jest zapewnienie odpowiednio wysokiej jakości takiej komunikacji oraz sprawienie, że użytkownicy danej platformy komunikacyjnej będą do niej chętnie powracać. Stan taki można osiągnąć na wiele sposobów, w szczególności można wprowadzić takie uregulowania oraz funkcjonalność usług, że korzystanie z preferowanych kanałów komunikacyjnych stanie się:

- tańsze,
- szybsze,
- bardziej bezpieczne
- itd.

Ponadto warto zadbać by wokół usług komunikacyjnych stworzyła się pewna wspólnota osób z nich korzystających. Można próbować to osiągnąć poprzez wprowadzenie metod pozwalających na interakcje między użytkownikami. Są to przykładowo fora dyskusyjne, komentarze pod opublikowanym tekstem oraz przemyślana integracja z mediami społecznościowymi.

Pomimo wysiłków nad zapewnieniem wysokiej atrakcyjności komunikacji, może się czasem okazać, że owa atrakcyjność nie rekompensuje nakładu pracy na komunikację poszczególnych interesariuszy. W rezultacie komunikacja ta może nie spełniać pokładanych w niej nadziei. Przykładem mogą być różnego rodzaju ankiety, badania opinii, konsultacje społeczne, rankingi, słowem wszystkie te rodzaje aktywności, które wymagają od interesariuszy (mieszkańców) większego niż zwykle wysiłku (lub czasu). W takiej sytuacji warto pomyśleć o zwiększeniu atrakcyjności komunikacji poprzez wprowadzenie różnego rodzaju zachęt do aktywnego w niej udziału. Mogą to być punkty, nagrody (przykładem takiej zachęty jest np. loteria paragonowa), wyróżnienia itp. Mogą to być również programy lojalnościowe, których zadaniem jest nie tyle związać użytkownika za platformą co umożliwić mu akumulację tych zachęt i być może również ich wykorzystanie.

7. Aplikacja do kontaktu z mieszkańcami

7.1 Charakterystyka ogólna

Jednym z najważniejszych efektów praktycznych Projektu jest opracowana aplikacja pozwalająca na komunikację z Mieszkańcami. Jej funkcjonalności pozwalają zarówno na informowanie Mieszkańców o działaniach prowadzonych na terenie Gminy, wraz z możliwością wglądu w działanie wdrożonych nowoczesnych systemu oraz publikowania popartych danymi materiałów podnoszących świadomość, jak i na zbieranie od Mieszkańców opinii, uwag i życzeń w sposób utrzymujący ich semantykę, a więc pozwalający na ich bezpośrednie wykorzystanie w procesach decyzyjnych.

Ponieważ aplikacja – w odróżnieniu od wielu narzędzi budowanych w innych projektach naukowych – ma charakter systemu używanego w trybie ciągłym i dostępnego publicznie (tzw. systemu produkcyjnego), prace te – oprócz charakteru badawczego – musiały również być prowadzone w odpowiednim rygorze inżynierii oprogramowania.

7.2 Opracowanie wymagań oraz wizji rozwiązania

Liczne dyskusje i konsultacje z Liderem Projektu pozwoliły na nakreślenie wizji aplikacji, która spełnia wymagania zarówno w obszarze gromadzenia i zarządzania danymi, jak i – a może przede wszystkim – w zakresie funkcji związanych z komunikacją z Mieszkańcami.

7.2.1 Grupy użytkowników

Zakłada się 4 grupy użytkowników:

1. **Użytkownicy zarejestrowani** – niekoniecznie mieszkańcy Gminy, którzy uzyskują dostęp do danych oznaczonych jako publiczne, ale nie mają możliwości uczestnictwa w procesie decyzyjnym.
2. **Zarejestrowani mieszkańcy Gminy** – użytkownicy, którzy w procesie rejestracji złożyli oświadczenie o byciu mieszkańcem Gminy i podali weryfikowalny adres zamieszkania na jej terenie.

3. **Operatorzy** – pracownicy Urzędu, którzy mają dostęp do funkcjonalności panelu operatorskiego, np. publikowanie treści, zatwierdzanie zgłoszeń, itd.
4. **Administratorzy** – pracownicy, którzy dodatkowo mają dostęp do części administracyjnej (technicznej) panelu operatorskiego.

7.2.2 Sposób rejestracji

Użytkownicy rejestrują się przy pomocy formularza rejestracyjnego. Przykładowym wzorcem takiej funkcjonalności jest formularz rejestracji w systemie budżetu obywatelskiego miasta Krakowa.

Użytkownicy mogą zadeklarować bycie mieszkańcem Gminy, co nadaje im status mieszkańca (patrz wyżej).

Po rejestracji, do aktywacji konta, niezbędne jest potwierdzenie adresu e-mail poprzez kliknięcie linku otrzymywanego drogą e-mailową.

Opcjonalnie możliwe jest dodanie moderacji bazy użytkowników przez operatora.

7.2.3 Funkcje systemu

Kolejne sekcje omawiają wizję leżącą u podstaw zrealizowanych, specyficznych funkcjonalności systemu. Szczegóły ich realizacji przedstawiono w sekcji 7.3.

Prezentacja zrealizowanych inwestycji

Inwestycje prezentowane są na mapie, a użytkownik może po kliknięciu zobaczyć aktualny stan urządzeń oraz dowiedzieć się więcej na temat korzyści poprzez treści redagowane przez Operatorów np. ile energii zużywałoby oświetlenie przed modernizacją.

Zgłaszanie potrzeb przez mieszkańców

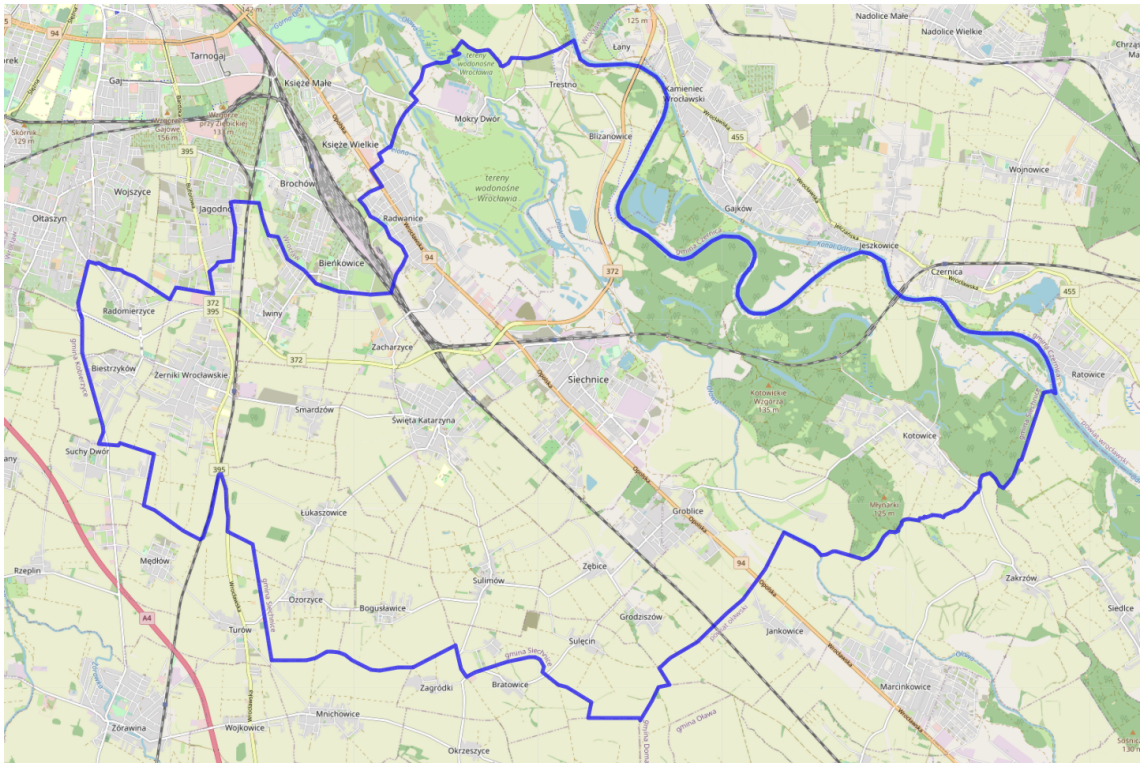
Użytkownicy o statusie mieszkańca mogą zgłaszać potrzeby dotyczące działań (m.in. inwestycji) na terenie Gminy. Zgłoszenia należą do kategorii z określonego katalogu (np. remonty dróg) i są przypisywane do lokalizacji, a tam gdzie jest to uzasadnione, również do poszczególnych obiektów topograficznych (np. konkretny odcinek drogi), dzięki integracji z danymi OpenStreetMap. Do zgłaszania wykorzystywany jest interaktywny interfejs mapowy aplikacji.

Zgłoszenia po przesłaniu widoczne są dla Operatora. Może on podjąć decyzję o opublikowaniu zgłoszenia, dzięki czemu staje się ono widoczne dla innych użytkowników, a ci ze statusem mieszkańca mogą je „podbijać” (np. kciuk w górę/kciuk w dół).

Historie poparte danymi – „stories”

Historie poparte danymi krótkie artykuły, zawierające narrację podnoszącą wiedzę lub świadomość mieszkańców w obszarach związanych z tematyką inteligentnych miast i szerzej zarządzania gminą.

Przykładem takiej funkcjonalności może być artykuł opisujący uwarunkowania finansowania bieżących inwestycji oraz rolę środków pozyskiwanych z podatków mieszkańców w tym procesie. Oprócz treści (pojawiającej się w panelu bocznym lub okienku), na mapie wyświetlane będą dane dotyczące wpływów z PIT zestawione z nakładami Gminy na inwestycje w poszczególnych miejscowościach. Dane będą prezentowane w postaci odpowiednio kolorowanych obszarów granic miejscowości, które użytkownik może klikać i wyświetlać szczegółowe informacje. Efektem tej konkretnej funkcjonalności jest zachęcenie osób które mieszkają na terenie Gminy, ale podatki odprowadzają gdzieś indziej, do zameldowania w Gminie Siechnice.



Rysunek 7.1: Granice Gminy Siechnice użyte do wyboru zakresu importowanych danych mapowych

7.3 Szczegółowa charakterystyka funkcjonalności

Niniejsza sekcja przedstawia najbardziej istotne szczegóły funkcji systemu zaimplementowanych

7.3.1 Mapa Gminy

Aplikacja została wyposażona w szczegółową bazę danych modelującą infrastrukturę oraz obiekty topograficzne na obszarze Gminy Siechnice. Obszar zaimportowany do systemu przedstawiono na rys. 7.1.

Źródłem danych jest serwis OpenStreetMap, a dane zostały zamodelowane w sposób zapewniający jednoznaczną identyfikację semantyki, parametrów oraz lokalizacji każdego z obiektów.

Moduł bazy danych modelujących mapę Gminy jest istotny, ponieważ stanowi podstawę funkcjonowania innych funkcjonalności, realizujących określone szczegółowe zadania.

Stworzona baza danych obejmuje obiekty trzech rodzajów:

- obiekty liniowe, które posiadają kształt w postaci linii (łamanej), łuku lub linii zamkniętej (obszaru) – takie jak drogi, obrysy budynków, granice obszarów zagospodarowania terenu, itd.,
- obiekty punktowe, których wymiary są na tyle niewielkie, że nie został im nadany odrębny kształt – takie jak kosze na śmieci, drzewa, itd.,
- metaobiekty, łączące ww. obiekty w większe struktury – np. zbiór dróg, które składają się na trasę linii autobusowej.

Wszystkie wyróżnione na mapie przyporządkowano do jednej z 25 zdefiniowanych kategorii tematycznych:

1. `aeroway` – elementy infrastruktury lotniczej, takie jak lotniska, hangary, wieże kontroli lotów itp.
2. `amenity` – usługi i udogodnienia dostępne dla ludzi, takie jak sklepy, hotele, restauracje, bankomaty itp.
3. `barrier` – przeszkody fizyczne, takie jak mury, ogrodzenia, bramy itp.
4. `boundary` – granice różnych obszarów, takich jak granice gmin, powiatów, miejscowości, itp.
5. `building` – budynki i inne budowle, takie jak domy, biurowce, kościoły itp.
6. `craft` – różne rodzaje zakładów rzemieślniczych, takich jak szwalnie, stolarnie itp.
7. `emergency` – elementy infrastruktury służącej do ratowania życia i zdrowia ludzi, takie jak szpitale, pogotowia ratunkowe itp.
8. `healthcare` – elementy infrastruktury służącej do opieki zdrowotnej, takie jak szpitale, przychodnie itp.
9. `highway` – różne rodzaje dróg, takie jak autostrady, drogi krajowe, ulice itp.
10. `historic` – elementy o znaczeniu historycznym, takie jak zamki, ruiny, pomniki itp.
11. `landuse` – różne rodzaje zagospodarowania terenu, takie jak rolnictwo, budownictwo mieszkaniowe itp.
12. `leisure` – różne rodzaje obiektów rekreacyjnych, takich jak parki, baseny itp.
13. `man_made` – elementy sztuczne wybudowane przez ludzi, takie jak kopce śmieci, latarnie morskie itp.
14. `military` – elementy infrastruktury wojskowej, takie jak magazyny, bazy wojskowe itp.
15. `natural` – obiekty naturalne, takie jak lasy, góry, jeziora itp.
16. `office` – różne rodzaje biur, takich jak biura rządowe, firmy konsultingowe itp.
17. `place` – nazwane miejsca, takie jak miasta, wsie, osady itp.
18. `power` – elementy infrastruktury związanej z dostarczaniem energii, takie jak elektrownie, linie elektroenergetyczne itp.
19. `public_transport` – elementy infrastruktury transportu publicznego, takie jak przystanki autobusowe, tramwajowe, metro itp.
20. `railway` – elementy infrastruktury kolejowej, takie jak tory, stacje kolejowe itp.
21. `shop` – sklepy, takie jak sklepy spożywcze, sklepy z odzieżą itp.
22. `sport` – obiekty sportowe, takie jak stadiony, boiska itp.
23. `tourism` – obiekty turystyczne, takie jak hotele, muzea itp.
24. `water` – elementy związane z wodą, takie jak jeziora, morza itp.
25. `waterway` – różne rodzaje dróg wodnych, takich jak rzeki, kanały, jeziora itp.

7.3.2 Weryfikacja statusu mieszkańca

Status mieszkańca weryfikowany jest w oparciu o złożone oświadczenie przez Operatora systemu. Aby dostarczyć Operatorowi odpowiednie narzędzia do weryfikacji, oświadczenie wymaga podania adresu zamieszkania na terenie Gminy.

Podany adres weryfikowany jest względem bazy punktów adresowych zawartych w bazie danych modelującej mapę Gminy przy pomocy klas walidacyjnych.

Klasa `PlaceValidator` służy do sprawdzania czy podana miejscowość leży w gminie Siechnice. Klasa ta posiada metodę `__call__`, która jest wywoływana podczas walidacji formularza. W metodzie tej następuje pobranie danych o miejscowościach za pomocą funkcji `osm.places_get()`, a następnie sprawdzenie, czy podana miejscowość znajduje się wśród pobranych miejscowości.

Klasa `StreetValidator` służy do sprawdzania, czy podana ulica znajduje się w miejscowości. Podobnie jak w przypadku klasy `PlaceValidator`, metoda `__call__` pobiera

dane o ulicach za pomocą funkcji `osm.place_streets_get()` i sprawdza, czy podana ulica znajduje się wśród pobranych ulic.

Klasa `AddressValidator` służy do sprawdzania poprawności adresu. Została ona przygotowana do weryfikacji istnienia zarówno adresów z nazwą ulicy jak i bez niej (dla wsi oraz miejscowości bez nazw ulic).

7.3.3 Zgłoszenia obywatelskie

Funkcjonalność zgłoszeń obywatelskich może być elastycznie stosowana do różnych celów. Najbardziej oczywistym i naturalnym jest przekazywanie przez Mieszkańców próśb do Gminy, dotyczących np. elementów infrastruktury wymagających remontu lub modernizacji (np. remonty dróg, wymiana oświetlenia).

Ze względu na możliwość elastycznej konfiguracji, moduł może także być wykorzystywany do zbierania informacji od Mieszkańców, np. dotyczących źródeł ogrzewania, efektywności energetycznej budynków lub chociażby postrzeganej przez nich dostępności komunikacji zbiorowej.

Cechą wyróżniającą moduł jest zaprojektowany i wdrożony sposób łączenia zgłoszeń z dowolnymi obiektami dostępnymi w bazie modelującej mapę Gminy (por. sekcja 7.3.1). Dzięki przyjęciu założenia o szerokiej konfigurowalności aplikacji, osoby obsługującej aplikację mogą zdefiniować zapytania pobierające z bazy obiekty dowolnej kategorii poprzez wybór kategorii oraz dodatkowe przefiltrowanie parametrów obiektów. Dokonuje się tego poprzez stworzenie odpowiedniego opisu w tekstowym pliku konfiguracyjnym aplikacji; przykładowo, zdefiniować można kategorie takie jak przykładowo:

- drogi – poprzez określenie rodzajów dróg uwzględnianych na mapie,
- j.w., ale tylko dróg przy których biegną chodniki – poprzez dodatkowe wymaganie obecności chodnika po jednej lub obu stronach drogi,
- budynki – wszystkie, lub wybranej kategorii (np. budynki użyteczności publicznej),
- przystanki komunikacji zbiorowej.

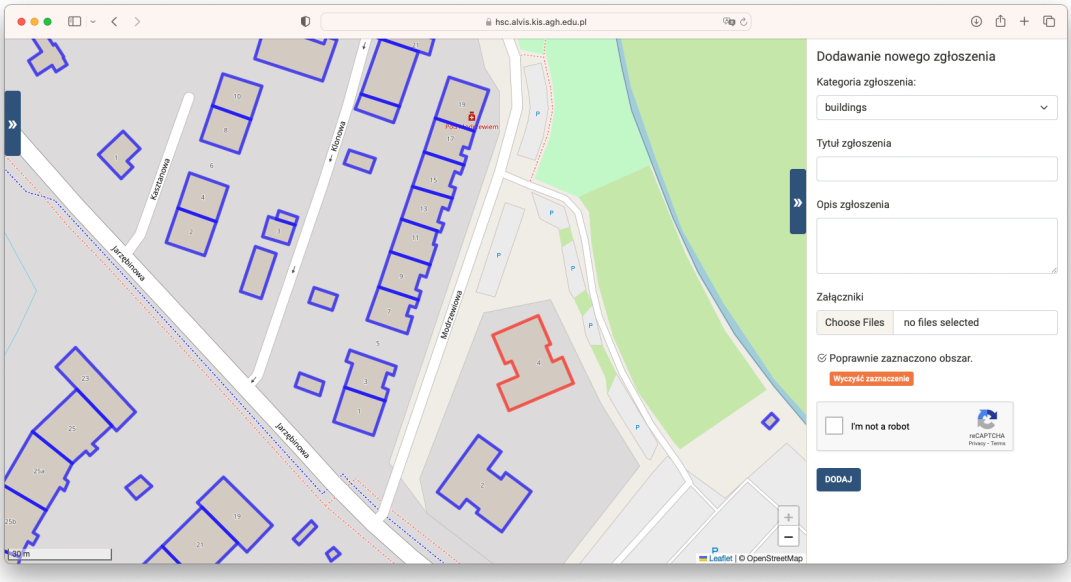
Użytkownik, podczas redagowania zgłoszenia, wybiera jedną z predefiniowanych wcześniej przez operatora kategorii. Przykładowo, jeżeli zgłoszenie dotyczy kategorii „budynki”, podświetleniu ulegają wszystkie budynki, a użytkownik może zaznaczyć wybrany, tak jak to przedstawiono na rys. 7.2.

Jeżeli zgłoszenie dotyczy obiektu liniowego, system umożliwia dodatkowo wybór określonego fragmentu obiektu. Przykładowo, przy zgłoszeniu dla kategorii „drogi”, w prosty sposób zaznaczyć można fragment drogi którego dotyczy zgłoszenie. Na rys. 7.3 przedstawiono proces tworzenia zgłoszenia dotyczącego odcinka ul. Jarzębinowej znajdującego się bezpośrednio przed terenem przedszkola.

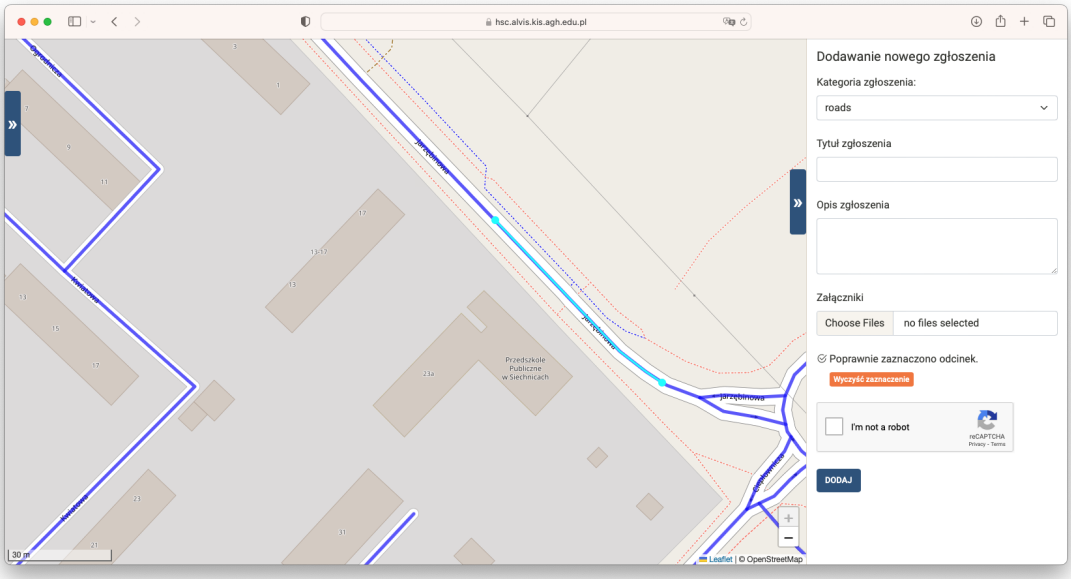
Utworzone zgłoszenia podlegają procesowi formalnego zatwierdzenia. W systemie zaprojektowano i zaimplementowano moduł realizujący obieg sprawy (ang. *workflow*). Każdy projekt może przyjąć jeden z 4 statusów:

- oczekujący na weryfikację – został dodany przez Mieszkańca i oczekuje na weryfikację przez Operatora
- zaakceptowany przez Operatora – dodany do mapy
- odrzucony przez Operatora – odrzucony ze względu na nieprawidłowości w opisie
- w trakcie realizacji – przyjęty do realizacji, co oznacza, że wkrótce pojawi się lub już jest zdefiniowany na stronie *Projekty realizowane*
- anulowany – projekt nie jest już pokazywany na mapie ponieważ nie został przegłosowany.

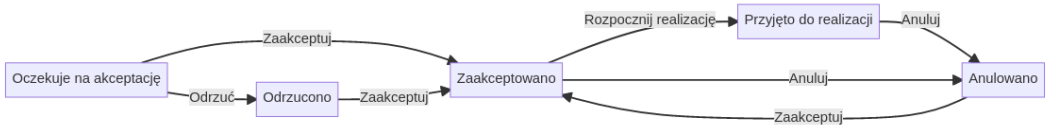
Diagram zmian statusu zgłoszeń przedstawiono na rys. 7.4.



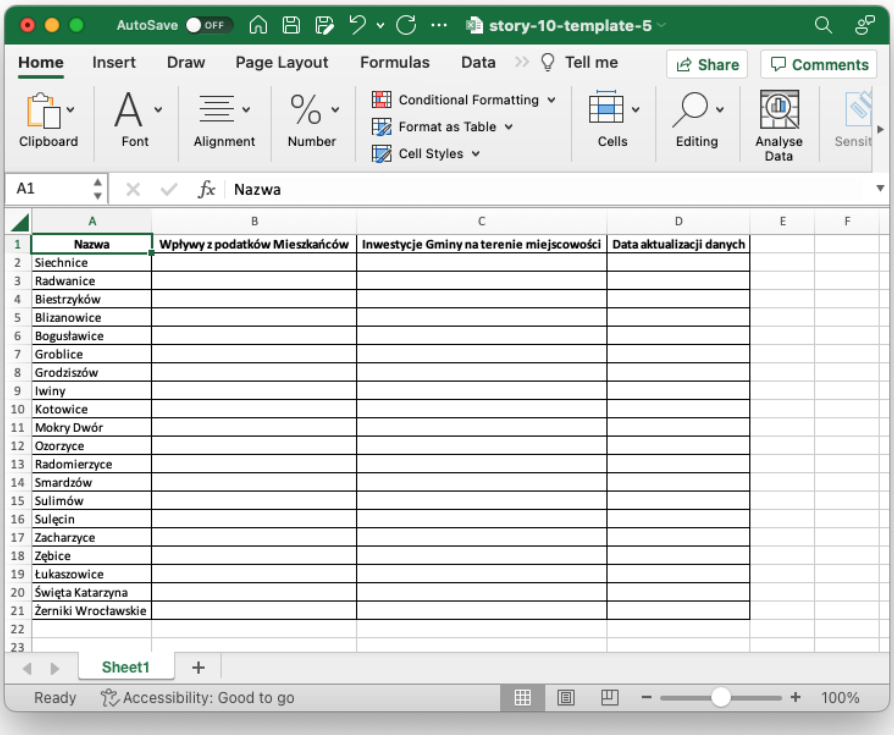
Rysunek 7.2: Przypisywanie zgłoszenia do budynku



Rysunek 7.3: Przypisywanie zgłoszenia do fragmentu drogi



Rysunek 7.4: Diagram zmian stanu zgłoszeń obywatelskich



	A	B	C	D
1	Nazwa	Wpływy z podatków Mieszkańców	Inwestycje Gminy na terenie miejscowości	Data aktualizacji danych
2	Siechnice			
3	Radwanice			
4	Biestrzyków			
5	Blizanowice			
6	Bogusławice			
7	Groblice			
8	Grodziszów			
9	Iwiny			
10	Kotowice			
11	Mokry Dwór			
12	Ozorzyce			
13	Radomierzyce			
14	Smardzów			
15	Sulimów			
16	Sulęcin			
17	Zacharzyce			
18	Zębice			
19	Łukaszowice			
20	Święta Katarzyna			
21	Żerniki Wrocławskie			
22				
23				

Rysunek 7.5: Wygenerowany przez system szablon do uzupełnienia danych

7.3.4 Artykuły oparte o dane

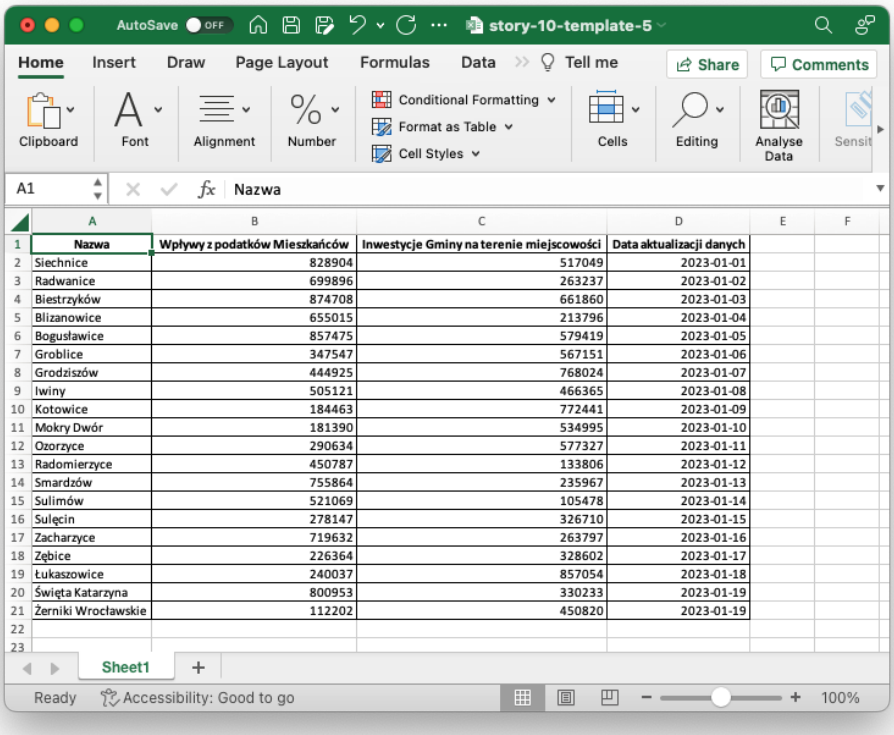
Funkcjonalność dodawania artykułów opartych o dane zaprojektowana została w oparciu o założenia przedstawione w sekcji 7.2.3.

Dla funkcjonalności zaprojektowano następujący schemat użycia:

1. Administrator definiuje w systemie kategorie obiektów (np. granice miejscowości), do których mogą być przyporządkowywane dane poprzez zdefiniowanie odpowiednich zapytań SQL do bazy modelującej mapę Gminy (por. sekcja 7.3.1).
2. Operator tworzy szablon artykułu w postaci pliku tekstowego w formacie YAML. Szablon określa, jakie parametry będą przypisywane do poszczególnych obiektów (np. dochody z podatków oraz inwestycje z gminy), formaty danych (waluty, daty, itp.), a także to jak wartości będą następnie wizualizowane na mapie.
3. System generuje dla operatora szablon w postaci pliku MS Excel, zawierający wolne miejsca do wklejenia odpowiednich wartości parametrów (por. rys. 7.5).
4. Operator uzupełnia wartości w szablonie MS Excel i ładuje plik w odpowiednie miejsce w systemie (por. rys. 7.6).
5. Operator uzupełnia treść artykułu, w postaci formatowanego tekstu, ewentualnie wzbogaconego zdjęciami.
6. Artykuł publikowany jest w systemie w postaci interaktywnej mapy, na której wartości reprezentowane są kolorem w oparciu o określony wzór matematyczny, np.:

$$\text{kolor} = \frac{\text{wpływy}}{\text{inwestycje}}$$

Wizualizację przykładowego artykułu przedstawiono na rys. 7.7.



AutoSave OFF story-10-template-5

Home Insert Draw Page Layout Formulas Data Tell me Share Comments

Clipboard Font Alignment Number Conditional Formatting Format as Table Cell Styles Cells Editing Analyse Data Sensit

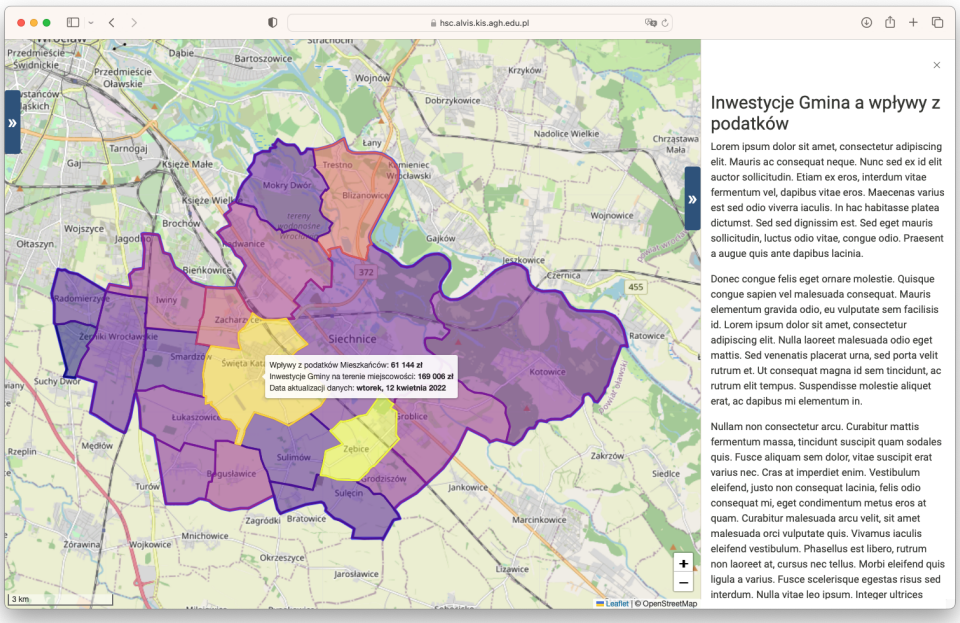
A1 Nazwa

	A	B	C	D	E	F
1	Nazwa	Wpływy z podatków Mieszkańców	Inwestycje Gminy na terenie miejscowości	Data aktualizacji danych		
2	Siechnice	828904	517049	2023-01-01		
3	Radwanice	699896	263237	2023-01-02		
4	Biestrzyków	874708	661860	2023-01-03		
5	Blizanowice	655015	213796	2023-01-04		
6	Bogusławice	857475	579419	2023-01-05		
7	Groblice	347547	567151	2023-01-06		
8	Grodziszów	444925	768024	2023-01-07		
9	Iwiny	505121	466365	2023-01-08		
10	Kotowice	184463	772441	2023-01-09		
11	Mokry Dwór	181390	534995	2023-01-10		
12	Ozorzyce	290634	577327	2023-01-11		
13	Radomierzyce	450787	133806	2023-01-12		
14	Smardzów	755864	235967	2023-01-13		
15	Sulimów	521069	105478	2023-01-14		
16	Sulęcin	278147	326710	2023-01-15		
17	Zacharzyce	719632	263797	2023-01-16		
18	Zębice	226364	328602	2023-01-17		
19	Łukaszowice	240037	857054	2023-01-18		
20	Święta Katarzyna	800953	330233	2023-01-19		
21	Żerniki Wrocławskie	112202	450820	2023-01-19		
22						
23						

Sheet1 +

Ready Accessibility: Good to go 100%

Rysunek 7.6: Uzupełniony danymi szablon MS Excel



Rysunek 7.7: Opublikowany artykuł oparty o dane, wraz z interaktywną mapą

7.3.5 Wdrożenie aplikacji

Aby zapewnić odpowiednią łatwość wdrożenia i przenośność aplikacji, zaprojektowano strukturę wirtualizacyjną opartą o mechanizm konteneryzacji.

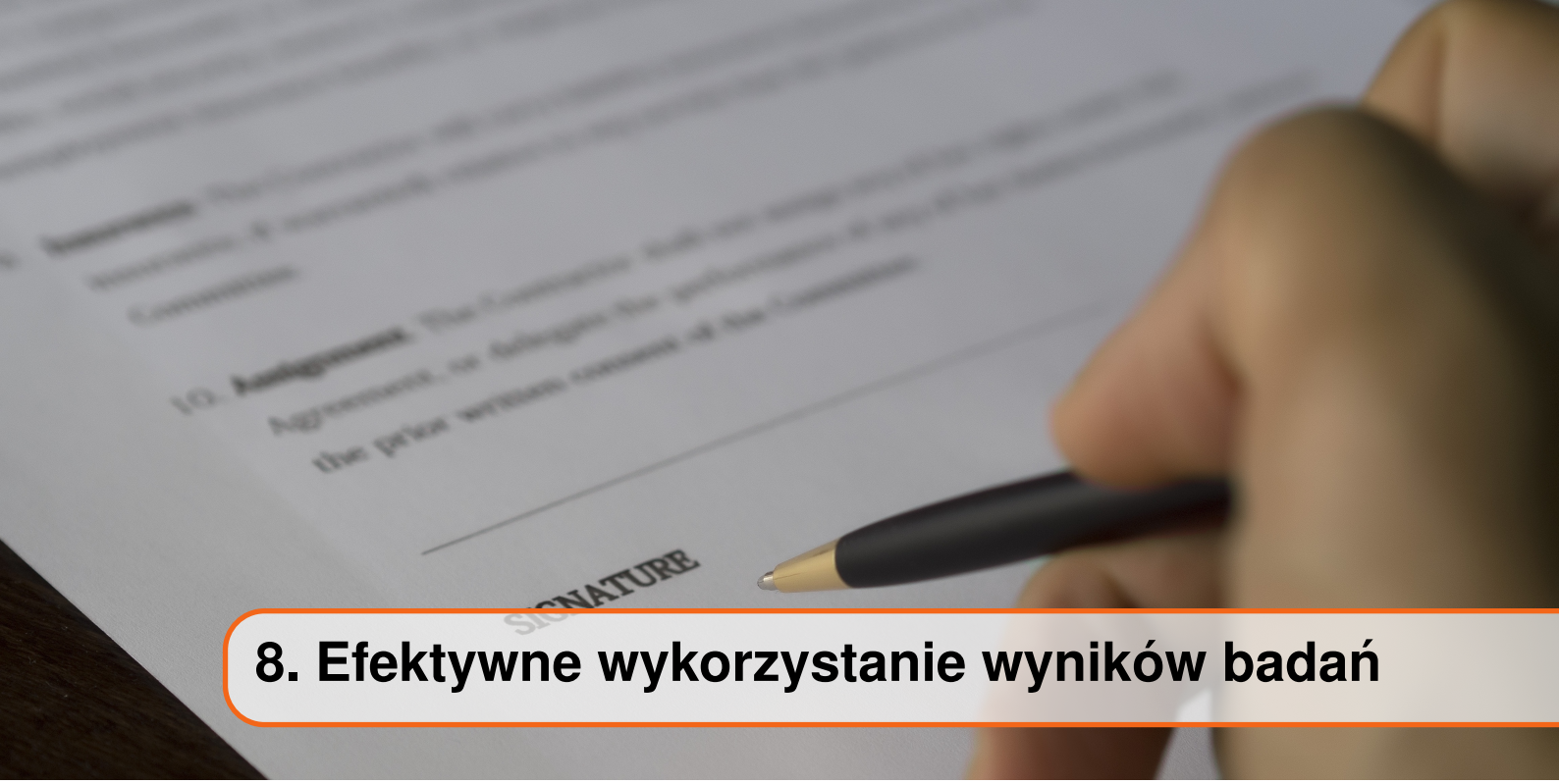
Jej implementacji dokonano przy pomocy systemu Docker. Pozwala on na tworzenie, uruchamianie i dzielenie się aplikacjami w sposób prosty i szybki. Kontenery Docker są niezależnymi, samodzielnymi jednostkami, które zawierają wszystko, czego potrzebują do działania, włączając w to kod, biblioteki, narzędzia oraz system operacyjny. Dzięki temu możliwe jest uruchamianie aplikacji w dowolnym środowisku, bez konieczności dostosowywania go do potrzeb danej aplikacji. Docker umożliwia również łatwe dzielenie się aplikacjami z innymi osobami lub zespołami, co ułatwia pracę nad projektami i zwiększa produktywność.

Strukturę zdefiniowano przy pomocy narzędzia Docker Compose. Umożliwia ono definiowanie i uruchamianie wielu kontenerów jednocześnie za pomocą jednego pliku YAML. Docker Compose ułatwia tworzenie skomplikowanych aplikacji opartych na wielu kontenerach, ponieważ pozwala na łatwe ich uruchamianie, zatrzymywanie i usuwanie za pomocą prostych poleceń. Plik YAML zawiera informacje o kontenerach, ich połączeniach oraz parametrach uruchomienia, dzięki czemu możliwe jest zautomatyzowanie procesu tworzenia i uruchamiania aplikacji. Docker Compose jest szczególnie przydatny przy tworzeniu aplikacji opartych na mikrousługach, ponieważ umożliwia łatwe uruchamianie wielu usług jednocześnie.

Takie przygotowanie wdrożenia aplikacji niesie ze sobą wiele korzyści:

1. **Prosta instalacja:** Docker pozwala na łatwe instalowanie i uruchamianie aplikacji na różnych platformach, co ułatwia przygotowanie wdrożenia na serwerach Gminy Siechnice.
2. **Wirtualizacja na poziomie systemu operacyjnego:** Kontenery Docker pozwalają na izolowanie aplikacji od reszty systemu, co zapewnia lepszą niezawodność i bezpieczeństwo.
3. **Oszczędność zasobów:** Kontenery Docker są lżejsze od tradycyjnych maszyn wirtualnych, co pozwala na oszczędność zasobów i lepsze wykorzystanie mocy obliczeniowej serwerów.
4. **Łatwy rozwój aplikacji:** Docker umożliwia łatwe dzielenie się aplikacjami z innymi osobami lub zespołami, co ułatwia pracę nad projektami i zwiększa produktywność.
5. **Łatwe zarządzanie wieloma kontenerami:** Docker Compose umożliwia łatwe zarządzanie wieloma kontenerami za pomocą jednego pliku YAML, co ułatwia tworzenie skomplikowanych aplikacji opartych na wielu kontenerach.
6. **Proste wdrażanie nowych wersji aplikacji:** Dzięki Dockerowi i Docker Compose łatwo jest wdrażać nowe wersje aplikacji, ponieważ wystarczy zastąpić kontener z aktualną wersją nowym kontenerem z nową wersją aplikacji.

IV Living Lab: trwająca współpraca uczelni z JST



8. Efektywne wykorzystanie wyników badań

8.1 Efekty żyjące a efekty zamrożone

Błąd często popełniany w projektach prowadzonych wspólnie przez JST oraz instytucje badawcze polega na tym, iż efekt prac sprowadza się do raportu obejmującego wnioski wypracowane na podstawie danych zebranych tylko w trakcie realizacji projektu. Skutkiem takiego podejścia jest często sytuacja, w której wypracowane wnioski zostają niejako „zamrożone", a nierzadko dotyczą one dane zebranych wcześniej, a więc już w chwili zakończenia projektu stają się częściowo nieaktualne.

Nowoczesne technologie oferują ogromne możliwości przetwarzania danych, a ich wykorzystanie stało się łatwiejsze dzięki powstaniu bogatych ekosystemów narzędzi analitycznych. Dlatego też długofalowa wartość dodana z realizacji projektu powinna zakładać utrzymanie ciągłości prac – zarówno eksploatacji wypracowanych narzędzi przez JST, jak i możliwości utrzymania przez jednostkę naukową dostępu do danych rejestrowanych przez wdrożone systemy.

W dziedzinach takich jak sztuczna inteligencja czy uczenie maszynowe brak danych stanowi najczęstszy problem spowalniający badania lub prowadzący do ich całkowitego zatrzymania. Dlatego też zarówno prace podczas realizacji projektów, jak i działania podejmowane po ich zakończeniu powinny być nastawione na możliwe utrzymanie dalszej współpracy.

Rezultatem takiego podejścia jest utworzenie tzw. *living labs* – żywych laboratoriów, które zapewniają JST dostęp do wiedzy i doświadczenia naukowców, a naukowcom – dostęp do tak im potrzebnych nowych danych.

8.2 Utrzymanie współpracy pomiędzy JST a jednostkami badawczymi

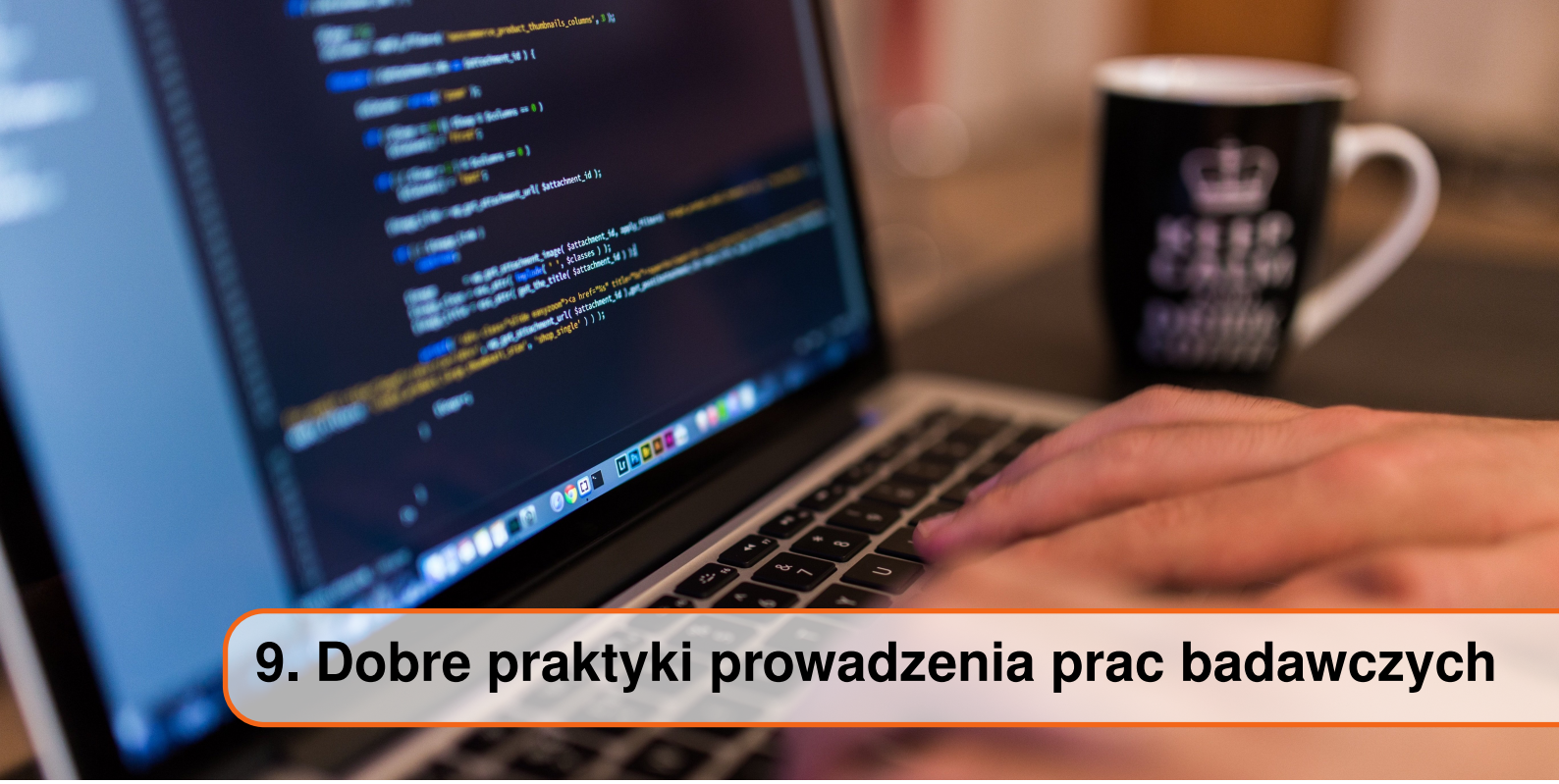
Jedną form utrzymania współpracy, stosowaną często w środowisku akademickim, jest zawarcie tzw. porozumień o współpracy.

Zakres takiej współpracy może obejmować:

- wsparcie merytoryczne pracowników naukowych procesów wdrażania wszelkich rozwiązań inteligentnych oraz służących redukcji użycia energii na terenie JST,
- dalszego rozwoju wytworzonych rozwiązań,
- zapewnienia ciągłości prac badawczych pracowników naukowych poprzez utrzymanie dostępu do danych wytwarzanych przez wdrożone przez JST rozwiązania„
- współpracę nad pozyskaniem finansowania dla kolejnych realizowanych przedsięwzięć,
- wsparcie procesu szerszego upublicznienia rozwiązań,
- wsparcie techniczne np. migracji oprogramowania na infrastrukturę należącą do JST,
- organizację spotkań mających na celu zapoznanie Przedstawicieli JST ze szczegółowymi wynikami realizacji projektu i możliwościami ich wykorzystania,
- adaptację stworzonych rozwiązań w celu maksymalizacji ich przydatności dla JST,
- wsparcie przygotowania różnych form propagacji stworzonych rozwiązań, takich jak wdrożenie w innych JST lub ewentualna komercjalizacja,
- wykorzystywanie danych pochodzących z JST w kolejnych publikacjach naukowych, przede wszystkim o zasięgu międzynarodowym,
- zapewnienie widoczności efektów działań innowacyjnych JST.

Porozumienia o współpracy to forma przyjęta i praktykowana na większości uczelni, które zazwyczaj posiadają jednostki koordynujące te działania. Dodatkowo, podnoszenie widoczności efektów badań może przyczynić się do zbudowania w szerszej skali sieci kompetencji, co może się przełożyć także na zintensyfikowanie współpracy pomiędzy jednostkami badawczymi, a także ułatwić kolejnym JST wybór partnerów dla kolejnych przedsięwzięć.

Przykładowo, jedną z pierwszych publikacji powstałych już po zakończeniu projektu HSC Siechnice jest pozycja [4], przygotowana wspólnie z przedstawicielem Gminy Siechnice, przestawiana na prestiżowej konferencji PP-RAI 2023. Prezentacja zaowocowała zarówno wyraźnym zainteresowaniem innowacyjnymi działaniami prowadzonymi przez Gminę, jak i podniesieniem świadomości dotyczącej obszarów kompetencji zespołów AGH.



9. Dobre praktyki prowadzenia prac badawczych

9.1 Zasady zapewnienia jakości prac naukowych

W projektach takich jak omawiany Aby umożliwić sprawne przechowywanie danych, ich wersjonowanie oraz zachowanie jakości tworzonych narzędzi analitycznych należy zastosować się do poniższych propozycji.

9.1.1 Dokumentowanie kodu i danych na bieżąco

Komórki notatnika JupyterLab pozwalają dodawać dokumentację w formacie Markdown. Wskazanym jest przeplatanie nimi komórek z kodem, tak aby opisać motywację i informacje techniczne dotyczące wszystkich kroków przetwarzania danych, które są podejmowane.

Celowe jest również ustrukturalizowanie notatnika z użyciem sekcji. Można to osiągnąć stosując komórki w formacie Markdown i odpowiednie znaczniki definiujące sekcję.

9.1.2 Odpowiednia struktura katalogów i konwencje użytych nazw

Należī sprecyzować odpowiednią strukturę katalogów wraz konwencją użytych nazw, tak aby zarówno struktura jak i nazwy miały znaczenie semantyczne.

Utworzone pliki z notatnikami powinny mieć konsekwentną konwencję nazw. Nazwa katalogu powinna odpowiadać zakresowi tematycznemu badania, czy też eksperymentu i być czytelna. Nazwy poszczególnych plików w katalogu powinny również reprezentować semantykę swej zawartości. Przykładowo:

```
analiza_ruchu_ulicznego
|-- natezenie_ruchu_z_iot.ipynb
\-- natezenie_ruchu_predykcja.ipynb
```

gdzie `analiza_ruchu_ulicznego` jest nazwą katalogu, a `natezenie_ruchu_z_iot.ipynb` i `natezenie_ruchu_predykcja.ipynb` są nazwami plików notatnika. W ogólnym przypadku w katalogu może znajdować się wiele plików oraz podkatalogi.

Warto zwrócić uwagę również na spójne używanie wielkich i małych liter (w w/w przykładzie tylko małe litery) oraz separatorów wyrazów (w w/w przykładzie jest to znak podkreślenia). Użycie wielkich małych liter jest dopuszczalne, ale ważne jest aby używać ich konsekwentnie. Ułatwia to późniejsze odszukanie katalogu czy też pliku oraz pozwala uniknąć problemu przy automatyzacji odczytu lub też zapisu plików, szczególnie w przypadku, gdy wielkie i małe litery są rozróżnialne przez system plików. Użycie spacji jako separatora wyrazów może prowadzić do trudności w rozróżnieniu, czy w nazwie jest jedna, czy też dwie spacje, podobnie czy jest spacja na początku czy też na końcu nazwy, co w konsekwencji może spowodować trudny do znalezienia błąd w kodzie. Zatem rekomenduje się użycie innych, widocznych znaków np. podkreślenia lub pauzy.

9.1.3 Wprowadzenie systemu kontroli wersji

W inżynierii oprogramowania kontrola wersji (znana również jako kontrola wersji, kontrola źródła lub zarządzanie kodem źródłowym) to klasa systemów odpowiedzialnych za zarządzanie zmianami w programach komputerowych, dokumentach, dużych witrynach internetowych lub innych zbiorach informacji. Kontrola wersji jest elementem zarządzania konfiguracją oprogramowania.

Zmiany są zwykle identyfikowane za pomocą kodu liczbowego lub literowego, określanego jako *numer wersji*, *poziom wersji* lub po prostu *wersja*. Na przykład początkowy zestaw plików to *wersja 1*. Gdy dokonywana jest pierwsza zmiana, wynikowym zestawem jest *wersja 2* i tak dalej. Każda wersja jest powiązana ze znacznikiem czasu i osobą wprowadzającą zmianę. Wersje można porównywać, przywracać, a w przypadku niektórych typów plików łączyć.

Potrzeba logicznego sposobu organizowania i kontrolowania wersji istniała prawie tak długo, jak istniało pisanie, ale kontrola wersji stała się znacznie ważniejsza i bardziej skomplikowana, gdy rozpoczęła się era komputerów. Numeracja wydań książek i rewizji specyfikacji to przykłady z okresu przed komputerowego. Obecnie najbardziej wydajne, a także złożone, systemy kontroli wersji to te używane w tworzeniu oprogramowania, w których zespół ludzi może jednocześnie wprowadzać zmiany w tych samych plikach.

Podstawową korzyścią jest możliwość przechowywania historii i cofania zmian. Daje to programiście więcej możliwości eksperymentowania, redukując możliwość uszkodzenia istniejącego kodu.

Kontrola wersji usprawnia również współpracę, ponieważ może łączyć zmiany dokonane w tym samym czasie przez różnych użytkowników, jak również zarządzać ewentualnymi konfliktami wynikłymi z takich zmian.

Proponowanym systemem kontroli wersji jest Git. Jest to rozproszony system kontroli wersji umożliwiający śledzenie zmian w dowolnym zestawie plików, zwykle używany do koordynowania pracy programistów wspólnie opracowujących kod źródłowy podczas tworzenia oprogramowania. Jego główne cechy to szybkość, integralność danych i obsługę rozproszonych, nieliniowych przepływów pracy.

Git jest wolnym oprogramowaniem o otwartym kodzie źródłowym, dystrybuowanym wyłącznie na licencji GPL-2.0.

9.1.4 Dostosowanie sposobu wersjonowania do specyfiki środowiska

Należī określić i wdrożyć odpowiedni sposób wersjonowania danych w kontekście formatu przechowania danych JupyterLab.

Zastosowanie w/w systemu kontroli wersji do notatników JupyterLab (pliki `.ipynb`) jest problematyczne uwagi na ich wewnętrzną strukturę. Zawiera ona nie tylko kod i dokumentację, ale również rezultaty działania tego kodu (tekst, grafikę, wykresy, diagramy

itp.). Każdorazowe uruchomienie kodu powoduje wygenerowanie nowych treści, które podlegałyby kontroli wersji. Notatnik, w którym nie poczyniono żadnych zmian, a jedynie uruchomiono kod w nim znajdujący się, będzie się kwalifikował jako nowa wersja oprogramowania. W połączeniu z wielodostępnością, z dużym prawdopodobieństwem prowadzi to powstawania konfliktów, czyli rozbieżnych wersji. Zarządzanie takimi rozbieżnymi wersjami jest zwykle żmudne i czasochłonne.

Rozwiązaniem jest zastosowanie pakietu Jupytext. Jest to pakiet oprogramowania w języku Python, który zapewnia dwukierunkową konwersję między notatnikami Jupyter a innymi formatami tekstowymi, takimi jak dokumenty lub skrypty Markdown (`.md`). Reprezentacja tekstowa zawiera tylko część notatnika zawierającą kod i dokumentację, bez wyników działania. Zatem zastosowanie Jupytext oraz wersjonowania plików w formacie Markdown, zamiast `.ipynb` rozwiązuje w/w problem konfliktów. Dzięki dwukierunkowej synchronizacji pliki `.md` są automatycznie synchronizowane z `.ipynb`.

Zatem rekomenduje się:

- nie obejmowanie systemem kontroli wersji plików `.ipynb`,
- wdrożenie synchronizacji za pomocą Jupytext,
- objęcie kontrolą wersji plików `.md` oraz pozostałych niezbędnych do uruchomienia kodu w notatniku,
- wyłączenie plików `.ipynb` z systemu kontroli wersji (poprzez dodanie ich do `.gitignore`).

9.1.5 Automatyczna kontrola poprawności procedur

Należy określić i wdrożyć odpowiedni mechanizm automatycznej kontroli poprawności kodu; testów automatycznych.

Automatyzacja testów oprogramowania polega na wykorzystaniu dodatkowego oprogramowania niezależnego od testowanego oprogramowania do kontrolowania wykonywania testów i porównywania rzeczywistych wyników z przewidywanymi wynikami. Dzięki temu można zautomatyzować niektóre powtarzalne, ale niezbędne zadania w już istniejącym sformalizowanym procesie testowania lub przeprowadzić dodatkowe testy, które trudno byłoby wykonać ręcznie. Automatyzacja testów ma kluczowe znaczenie dla zapewnienia jakości oprogramowania.

Szczegółowe rekomendacje w zakresie testowania kodu opisano w sekcji 9.2 poniżej.

9.1.6 Konfekcjonowanie dojrzałych procedur analitycznych

Po osiągnięciu dojrzałości danej procedury analitycznej, pożądanym może być przekształcenie kodu analitycznego w osobne narzędzie. Wykonywane jest wtedy sparowanie danego notebooka z plikami źródłowymi języka Python. Zalecanym krokiem jest wtedy umożliwienie określania istotnych parametrów danej analizy w postaci argumentów linii poleceń, poprzez wykorzystanie modułu `argparse`).

9.2 Testowanie procedur analitycznych

Dostępne jest wiele pakietów oprogramowania dla języka Python realizujących proces testowania. Jednym z najbardziej popularnym jest pytest. Ponieważ narzędziem analitycznym jest JupyterLab i notatniki, należy również zapewnić odpowiednie podejście do testowania takiego oprogramowania tj.

1. sporządzić testy w osobnych notatnikach (`.ipynb`),
2. umożliwić konstrukcję testów z wykorzystaniem mechanizmów udostępnianych przez pytest,

3. zapewnić zautomatyzowane uruchamianie testów, tak aby raporty z ich działania były dostępne z poziomu JupyterLab.

Aby sprostać powyższym wymaganiom należy:

1. użyć pakietu ipynb do importu i przetwarzania plików notatnika,
2. użyć pakietu importlib do importowania kodu notatnika w notatnikach przeznaczonych do testów.
3. użyć pakietu nbmake, który dodaje możliwość automatyzacji testowania notatników,

Zatem do zbioru notatników realizujących podstawowe funkcje należy dodać notatniki zawierające procedury testowe. Nazwy takich notatników powinny zaczynać się od `test_`.

Przykładowa zawartość komórki notatnika `obliczenia.ipynb`:

```
def moja_funkcja():
    return 1
```

Przykładowa zawartość komórki notatnika testującego `test_obliczenia.ipynb`:

```
import importlib
nb = importlib.import_module("ipynb.fs.defs." + "obliczenia")

assert moja_funkcja() == 1
```

Pierwsza linia to import niezbędnego modułu. Linia 2 realizuje automatyczne uruchomienie notatnika o nazwie `obliczenia.ipynb`. Ostatnia linijka to definicja testu pytest dotyczącego wywołania funkcji `moja_funkcja()`, która musi zwrócić wartość 1. Każda inna wartość spowoduje błąd testu.

Wszystkie notatniki można również uruchomić za pośrednictwem JupyterLab z linii poleceń:

```
pytest --nbmake --overwrite
```

Dodatkowo powyższe polecenie generuje treść wszystkich notatników (opcja `--overwrite`). Uruchomienie jedynie testów można zrealizować za pomocą polecenia:

```
pytest --nbmake --overwrite test*.ipynb
```

9.3 Środowisko prowadzenia prac naukowych

Niezwykle istotnym, a niestety często zaniedbywanym podczas realizacji projektów badawczych aspektem jest zapewnienie odpowiedniej spójności i porządku w pracach naukowych. Jest to szczególnie istotne w projektach i dziedzinach, w których gros eksperymentów prowadzonych jest w formie budowy programów komputerowych. Zagadnienie to obejmuje zarówno problem stałej wymiany doświadczeń pomiędzy naukowcami wchodzącymi w skład zespołu projektowego (i, w efekcie, uniknięcie efektu „silosowania"), jak i ustalenie ram pozwalających na współpracę i ponowne wykorzystanie utworzonych programów komputerowych służących do prowadzenia eksperymentów.

W niniejszej sekcji przedstawiono praktyki wypracowane i wdrożone na samej AGH, których efektem, oprócz uporządkowania strony „warsztatowej" samego projektu, było m.in. uzupełnienie programu studiów o treści związane z tym aspektem pracy badawczej.

9.3.1 Repozytorium kodu

Centralnym punktem jest repozytorium kodu, zarządzane przy pomocy systemu kontroli wersji Git. Pozwala on badaczom i analitykom śledzić zmiany w ich kodzie i zarządzać wspólnymi podprojektami analitycznymi. Posiada on architekturę rozproszoną, co oznacza, że każda kopia kodu programisty jest pełnym repozytorium, które może być używane do udostępniania zmian innym programistom.

Jedną z głównych cech Git jsest jego zdolność do śledzenia zmian w kodzie na przestrzeni czasu. Gdy programista wprowadza zmiany w kodzie, może użyć Git, aby zatwierdzić zmianę, wraz z wiadomością opisującą zmianę. Tworzy to nową wersję w historii kodu. Git pozwala programistom wracać do poprzednich wersji, porównywać zmiany i cofnąć się do wcześniejszych wersji, jeśli to konieczne.

Git oferuje również potężne możliwości tworzenia gałęzi i scalania. Badacze mogą tworzyć osobne gałęzie dla różnych funkcji lub poprawek błędów, a następnie łączyć te zmiany z główną gałęzią, gdy są gotowe. To umożliwia równoległy rozwój i ułatwia integrację zmian od wielu programistów.

9.3.2 Środowisko eksperymentalne

Jako środowisko do wykonywania eksperymentów wybrano platformę Jupyter, a w szczególności system JupyterLab. Jest to interaktywne środowisko programistyczne, które pozwala na tworzenie, edytowanie i udostępnianie dokumentów zawierających kod, wyniki, wizualizacje i opisy tekstowe. JupyterLab jest bardzo popularnym narzędziem wśród naukowców, analityków danych i programistów, ponieważ umożliwia łatwe wizualizowanie i eksplorowanie danych oraz współdzielenie rezultatów z innymi.

Pakiet JupyterLab pozwala na przygotowywanie tzw. notatników (ang. *notebook*), łączących uruchamialny kod z treściami opisowymi, takimi jak tekst, tabele czy wzory matematyczne. Pliki Jupyter Notebook to dokumenty zapisywane w formacie .ipynb, które zawierają kod, wyniki, wizualizacje i opisy tekstowe, a także metadane dotyczące sposobu ich wyświetlania. Pliki te są tworzone i edytowane w programie JupyterLab i mogą być udostępniane innym osobom, dzięki czemu można wspólnie pracować nad projektem i udostępniać rezultaty. Pliki Jupyter Notebook są często używane do tworzenia raportów naukowych, prezentacji danych i instrukcji technicznych, ponieważ umożliwiają łączenie kodu, wyników i opisów w jednym dokumencie.

Wybrana do prac analitycznych i omawiana wyżej platforma JupyterLab to internetowe, interaktywne środowisko programistyczne dla dokumentacji, kodu i danych. Jego elastyczny interfejs pozwala użytkownikom konfigurować i organizować pracę w zakresie: analityki danych, obliczeń naukowych, dziennikarstwa obliczeniowego i uczenia maszynowego, poprzez tworzenie tzw. notatników, w formie dzienników badań.

Prace analityczne przeprowadzane są za pomocą kodu w języku Python wzbogaconego opisem i możliwością interaktywnego generowania zestawień i wykresów. Wyżej wymieniony kod uruchamiany jest w środowisku analitycznym z wykorzystaniem dostępu do danych zgromadzonych w *Magazynie danych*.

JupyterLab, jak również JupyterHub, jest wolnym oprogramowaniem udostępnionym na zmodyfikowanej licencji BSD.

9.3.3 Ogólnodostępny system centralny

Ogólnodostępny system centralny oparto o system JupyterHub, uruchamiany na jednym z serwerów AGH. Dzięki temu różni użytkownicy mają dostęp do narzędzi programistycznych i analitycznych przy pomocy samej przeglądarki internetowej, a jednocześnie dostają do

dyspozycji rozłączne środowiska, których synchronizacja wykonywana jest wedle opisu zawartego w sekcji

JupyterHub to narzędzie do tworzenia wirtualnych środowisk programistycznych, które umożliwiają użytkownikom dostęp do interaktywnych komputerów z zainstalowanym oprogramowaniem, takim jak JupyterLab, z dowolnego miejsca za pośrednictwem przeglądarki internetowej. JupyterHub jest szczególnie przydatny w sytuacjach, gdy wiele osób potrzebuje dostępu do tego samego zestawu narzędzi programistycznych lub gdy użytkownicy nie chcą lub nie mogą instalować oprogramowania na swoich komputerach.

JupyterHub może być wdrożony w pojedynczym kontenerze, co oznacza, że wszystkie potrzebne narzędzia i oprogramowanie są zainstalowane w jednym kontenerze Docker. Takie rozwiązanie jest proste w implementacji i sprawdza się w przypadku niewielkich zastosowań, ale może okazać się ograniczone pod względem skalowalności i elastyczności.

Alternatywnie, JupyterHub może być wdrożony przy użyciu systemu Kubernetes, co pozwala na skalowanie i zarządzanie wirtualnymi środowiskami programistycznymi w sposób elastyczny i skalowalny. System Kubernetes pozwala na tworzenie i zarządzanie klastrami kontenerów, dzięki czemu można łatwo rozszerzać lub zmniejszać zasoby dostępne dla użytkowników JupyterHub w zależności od ich potrzeb. Wdrożenie JupyterHub przy użyciu systemu Kubernetes wymaga więcej pracy i wiedzy technicznej, ale pozwala na lepszą elastyczność i skalowalność w przypadku dużych zastosowań.

9.3.4 Wsparcie dla pracy lokalnej

Zaletą środowiska centralnego jest możliwość uzyskania do niego dostępu przez naukowców bez potrzeby konfigurowania własnego sprzętu ani oprogramowania, co znacząco obniża próg i czas potrzebny na wdrożenie w środowisko analityczne.

Jednakże w pewnych sytuacjach pożądana może być możliwość uruchamiania procedur lokalnie – np. na własnym komputerze, lub na specjalizowanym serwerze obliczeniowym, takim jak wykorzystywane w projekcie przez AGH maszyny wyposażone w wydajne procesory CPU AMD Ryzen oraz GPU Nvidia RTX 3090.

Wsparcie dla pracy rozproszonej osiągnięto przygotowując pakiet oprogramowania będący w stanie zbudować całe środowisko wedle podanego przepisu na wybranej maszynie.

Przygotowany pakiet oparty jest o system Conda. Jest to narzędzie do zarządzania pakietami i środowiskami w systemach operacyjnych z rodziny Unix, takich jak Linux i MacOS. Jest szczególnie przydatne w przypadku projektów naukowych i analitycznych, ponieważ umożliwia łatwe instalowanie, aktualizowanie i usuwanie pakietów oraz tworzenie izolowanych środowisk, w których mogą być uruchamiane różne aplikacje z różnymi wersjami zależności. Dzięki Conda można też łatwo zarządzać zależnościami między pakietami i uniknąć problemów z konfliktami wersji. Narzędzie to jest dostępne dla wielu języków programowania, w tym Python, R, Ruby i wielu innych.

10. Analiza danych

10.1 Znaczenie analizy danych

Oprogramowanie dostarczane z systemami IoT, takimi jak systemy sensoryczne, pozwala najczęściej wyłącznie na wizualizację rejestrowanych parametrów, co ogranicza możliwości ich analizy. Dlatego w projekcie szczególny nacisk położono, obok zbierania danych do spójnych struktur bazodanowych, na opracowanie metod pozwalających na integrację zbiorów pochodzących z różnych źródeł i budowę metod pozwalających na ich efektywną analizę, prowadzącą do automatycznego wyciągania wniosków mających praktyczne znaczenie.

10.2 Porównywanie szeregów czasowych

Do analizy szeregów czaosowych, ze szczególnym uwzględnieniem geolokalizacji rozpatrywane są dwa podstawowe narzędzia:

1. statystyczne, w postaci korelacji krzyżowej, oraz
2. algorytmiczne, w postaci algorytmu DTW (Dynamic Time Warp).

Zostały wykonane następujące eksperymenty.

1. Pozyskanie i obróbka gelokowanych danych o charakterystyce zbliżonej do projektu; dane winne być pozyskane z uwagi na brak danych z projektu we wczesnej fazie realizacji.
2. Dobór odpowiednich komponentów oprogramowania realizujących w/w narzędzia.
3. Analiza DTW oraz wizualizacji wyników.
4. Analiza korelacji krzyżowej oraz przesunięć w dziedzinie częstotliwości.
5. Wizualizacja i eksploracja danych.

Rekomendowane jest użycie języka Python w wersji 3.x wraz z następującymi narzędziami:

1. pandas,
2. matplotlib,
3. dtw-python,

4. numpy,
5. scipy,
6. sweetviz.

Podstawy teoretyczne proponowanego podejścia oparte są m.in. o prace [2, 11].

10.3 Porównywanie parami

Jednym ze sposobów określania priorytetów (tworzenia rankingu) jest metoda porównywania alternatyw parami. Wychodzi ona z założenia, że prościej jest porównywać opcje, warianty decyzyjne, alternatywy w parach, niż próbować od razu wytypować „na raz" kolejność wszystkich możliwych propozycji lub rozwiązań.

Metoda dobrze sprawdza się do budowy małej i średniej wielkości rankingów, w których liczba rozważanych alternatyw nie jest duża (od kilku do kilkunastu alternatyw). Metoda AHP została opisana w szeregu prac [7, 10].

Moduł powinien wspierać obliczenia na danych niekompletnych tj. takich w których porównań pomiędzy niektórymi parami alternatyw brakuje. Pozwoli to z jednej strony na redukcję ilości koniecznych zapytań, na które muszą odpowiedzieć interesariusze, z drugiej strony znacznie uodporni system na problem niekompletnych ankiet tj. takich, w których ankietowany celowo lub niechcący nie odpowiedział na wszystkie zadane pytania. Istnieje co najmniej kilka metod pozwalających na obliczanie takich rankingów [8, 7].

W rozważanym podejściu zakładamy, że wejściem do systemu będą dane niespójne tj. takie dla których ilościowa relacja przechodniości nie zawsze jest spełniona. W praktyce niespójność danych będzie oznaczać, że nie zawsze będzie możliwe znalezienie takiego rankingu by proporcja wartości dwóch alternatyw powiedzmy i-tej i j-tej odpowiadała tej proporcji, którą zasugerował ekspert. Niespójność danych rankingowych nie jest postrzegana jako wada tych danych. Więcej nawet, pewna, ale nie za duża niespójność danych może być korzystna. Dane w wysokim stopniu niespójne sugerują jednak, że ekspert albo nie był pewny swoich opinii albo nie znał się na przedmiocie oceny.

Do pomiaru niespójności danych służą tzw. indeksy niespójności. Wysoka ich wartość może wskazywać na małą wiarygodność rankingu. Moduł rankingowy powinien udostępniać podstawowe metody pomiaru niespójności poszczególnych macierzy rankingowych.

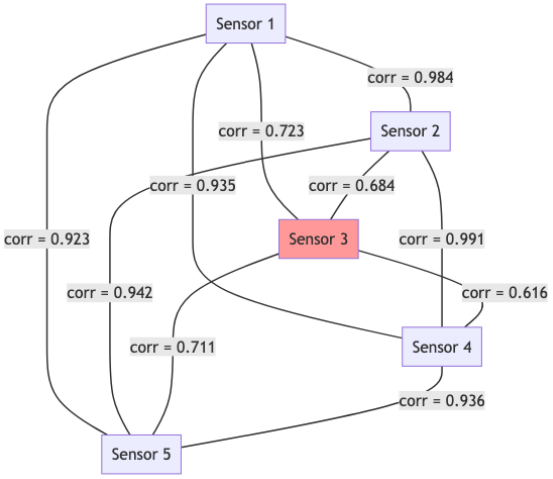
10.4 Integracja zbiorów danych poprzez materializację wykrytych związków

Proponowaną metodą wiązania zbiorów danych podlegających analizie jest materializacja związków pomiędzy obiektami przy pomocy formalnego modelu grafowego. Wykorzystuje ona mechanizm transformacji grafowych do utrwalania semantyki związków wykrytych we wszystkich zbiorach danych wchodzących w skład stworzonego rozwiązania.

Mechanizm ten oparty jest o uogólnioną metodę STGT (ang. *Spatially-Triggered Graph Transformations*), stosowaną wcześniej z powodzeniem m.in. do optymalizacji oświetlenia zewnętrznego[ˆstgt]. Koncepcja ta oparta jest o zbudowanie grafu modelującego wszelkie występujące w systemie obiekty zgodnie z ich semantyką, i z zachowaniem wszystkich istotnych atrybutów danych, a następnie wykonywanie analiz dziedzinowych w oparciu o adekwatne do tychże narzędzia i materializowanie ich jako związki (krawędzie) w grafie.

W proponowanym podejściu STGT proces analizy podzielony jest na fazy:

1. Osobne zbiory danych importowane są do grafu, z zachowaniem semantyki obiektów oraz atrybutów.
2. Na zbiorach danych wykonywane są analizy przy pomocy dedykowanych narzędzi.



Rysunek 10.1: Graf przedstawiający współczynniki korelacji odczytów z pięciu czujników

3. Wykryte związki (np. „lampa X oświetla ulicę Y”) zapisywane są – znów z zachowaniem semantyki – jako krawędzie w grafie.
4. Dalsza analiza prowadzone jest przy pomocy reguł określonych jako tradycyjne transformacje grafowe.
5. Wynikiem jest profilowany graf zawierający odpowiednią wiedzę syntentyczną.

Zgodnie z nazwą, pierwotna koncepcja STGT skupiała się na wykrywaniu związków przestrzennych, do czego wykorzystywane były odpowiednie narzędzia, jak PostGIS czy GeoPandas. Proponowane uogólnienie polega na rozszerzeniu spektrum wykrywanych związków o inne obszary.

Przykładem może być tu analiza szeregów czasowych. Wyobraźmy sobie, że w różnych punktach wdrożono system urządzeń pomiarowych. Każdy z tych czujników produkuje dane o identycznych atrybutach (mierzone parametry), lecz – oczywiście – innych wartościach.

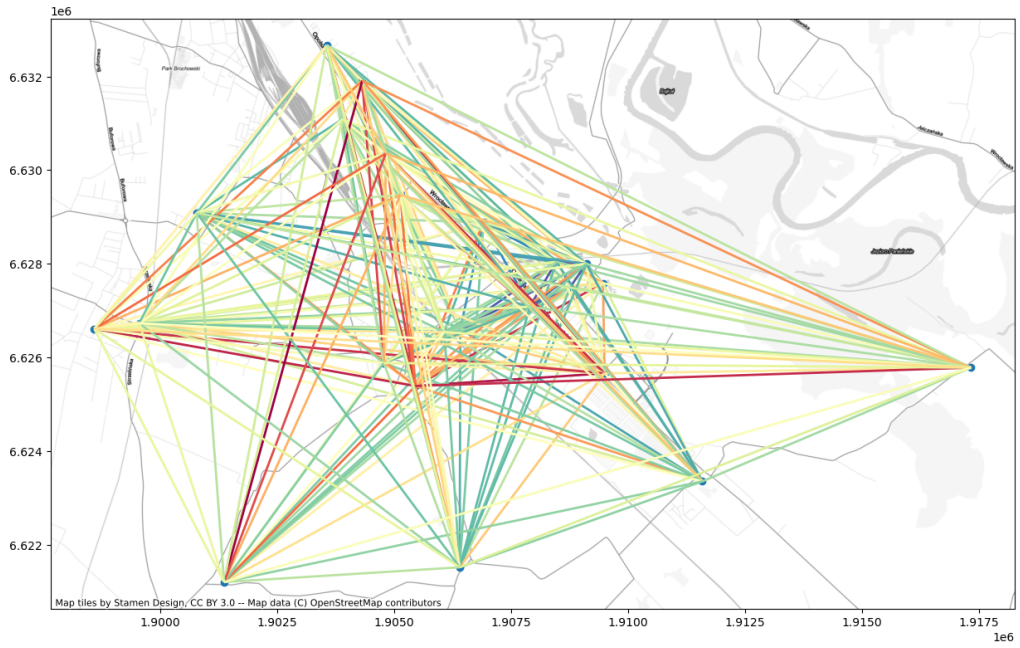
Zebrane dane z czujników mogą być poddane analizie, np. obejmującej to, jak poszczególne odczyty odbiegają od siebie nawzajem. Analizy takiej dokonać można przy pomocy prostych metod, jak budowa macierzy korelacji, lub bardziej złożonych, uwzględniających przesunięcia czasowe, jak np. algorytm DTW (ang. *Dynamic Time Warping*) [1]. Algorytm ten może być szczególnie przydatny w przypadku systemów, w których następuje propagacja wartości – jak np. czujniki jakości powietrza (przemieszczanie się zanieczyszczeń) lub sieci rozsyłowe (np. wodociągowa). Przykładowo, graf ukazany na rys. 10.1 przedstawia wyliczone wartości korelacji dla systemu składającego się z pięciu czujników.

Widać tu, że odczyty z czujnika nr 3 (podświetlony na czerwono) są wyraźnie słabiej skorelowane z innymi czujnikami niż pozostałe. Może to wskazywać na uszkodzenie czujnika, ale także na faktyczne odstępstwo mierzonej wartości (ang. *outlier*).

10.5 Wizualizacja wyników analiz

Istotnym problemem, szczególnie przy pracy z danymi geograficznymi, jest odpowiednia wizualizacja wyników analiz. Pojawia się tu jednak problem skali.

Jeżeli przyjmiemy, że związki w grafie obrazują np. korelację pomiędzy wartościami z czujników, a więc że istnieje parametryzowany związek pomiędzy każdą parą urządzeń,



Rysunek 10.2: Wizualizacja korelacji odczytów czujników poziomu pyłów zawieszonych w Gminie Siechnice

to liczba krawędzi w grafie modelującym system IoT liczący n urządzeń wynosi:

$$e = \frac{n \cdot (n-1)}{2}$$

Oznacza to że przyrost tej liczby jest wykładniczy, więc wizualizacja takiego grafu może być niemożliwa, szczególnie jeżeli uwzględniona miałaby lokalizacja urządzeń.

Rozwiązaniem tego problemu jest reprezentacja wartości powiązań przy pomocy kolorów na mapie.

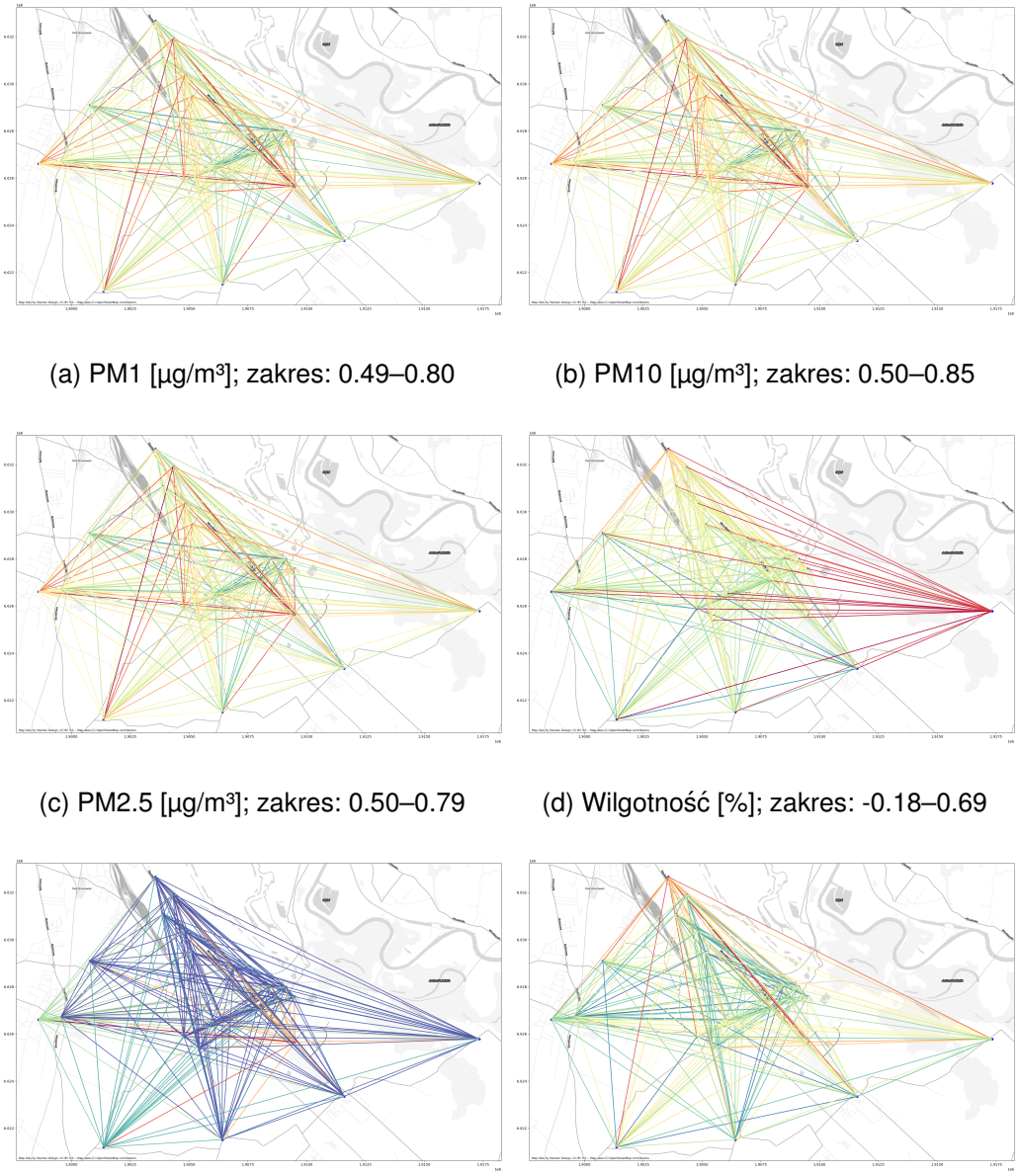
Przykład zastosowania tego podejścia do wizualizacji stopnia korelacji wartości odczytów poziomu pyłów zawieszonych z czujników jakości powietrza w Gminie Siechnice przedstawiono na rys. 10.2.

10.6 Analiza danych sensorycznych

Za przykład do omówienia opracowanej metodyki analizy danych sensorycznych przyjmijmy dane z systemu czujników jakości powietrza. Na rys. 10.3 przedstawiono wizualizację grafów obrazujących korelację Tau Kendalla dla niektórych parametrów przez nie rejestrowanych. Tabela 10.1 przedstawia zakresy wartości współczynnika korelacji dla przedstawionych parametrów. Zakresy zmienności przedstawiono na rys. 10.4.

Przyjrzyjmy się znaczeniu otrzymanych wyników. Porównując parametry poziomu pyłów zawieszonych (PM1, PM2.5, PM10), zarówno zakresy współczynników korelacji, jak i ich rozkład przestrzenny (por. rys. 10.3a, 10.3b i 10.3c) są zbliżone. Oznacza to, iż charakterystyka czujników dla tego parametru jest stabilna, co świadczy pozytywnie o ich jakości, lecz może także być związane z wykorzystaniem dla wszystkich tych parametrów podobnych technologii pomiarowych.

Przy analizie grafu zwraca jednak uwagę względnie niski poziom korelacji czujnika zlokalizowanego przy skrzyżowaniu DK 94 (ul. Opolska) oraz ul. Ciepłowniczej z pozostałymi czujnikami. Po zintegrowaniu grafów korelacji odczytów oraz grafów modelujących



(a) PM1 [µg/m³]; zakres: 0.49–0.80

(b) PM10 [µg/m³]; zakres: 0.50–0.85

(c) PM2.5 [µg/m³]; zakres: 0.50–0.79

(d) Wilgotność [%]; zakres: -0.18–0.69

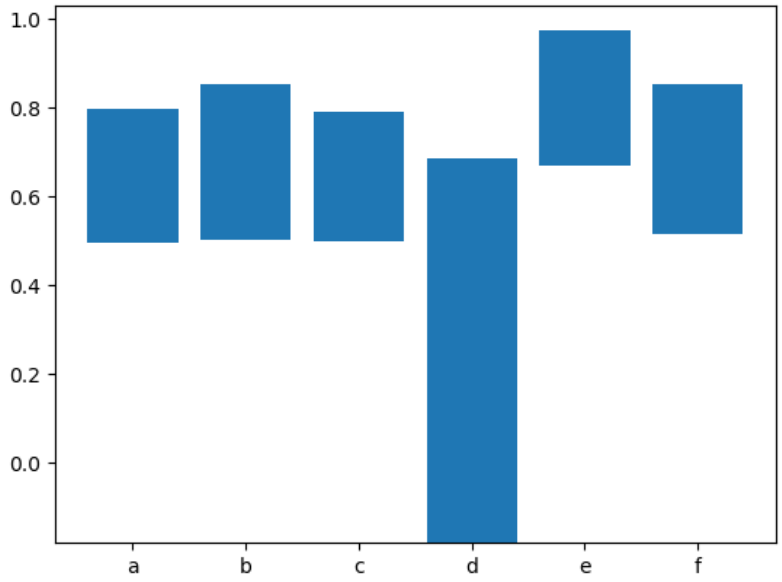
(e) Ciśnienie [hPa]; zakres: 0.67–0.97

(f) Temperatura [℃]; zakres: 0.51–0.85

Rysunek 10.3: Wizualizacja grafów modelujących korelację Tau Kendalla dla danych pozyskanych z systemu monitorowania jakości powietrza

Symbol	Nazwa	Min	Max
a	PM1 [µg/m³]	0.493865	0.797666
b	PM10 [µg/m³]	0.500864	0.851019
c	PM2.5 [µg/m³]	0.496218	0.788663
d	Wilgotność [%]	-0.182255	0.685498
e	Ciśnienie [hPa]	0.668560	0.972545
f	Temperatura [℃]	0.512723	0.853681

Tabela 10.1: Zakresy współczynnika korelacji dla wybranych parametrów jakości powietrza



Rysunek 10.4: Zakresy współczynników korelacji dla poszczególnych parametrów

natężenie ruchu wykrywane przez zainstalowane kamery (poprzez dodatkową materializację związków przestrzennych kamera–droga–czujnik, por. sekcja 10.4) przeprowadzono dalsze analizy, wskazujące na wpływ ruchu pojazdów w tej lokalizacji. Dalsze powiązanie grafu analitycznego z grafem modelującym mapę drogową miasta pokazało, iż ruch na skrzyżowaniu jest regulowany sygnalizacją świetlną, co sprawia że pojazdy zatrzymują się tam często, zazwyczaj przy włączonym silniku.

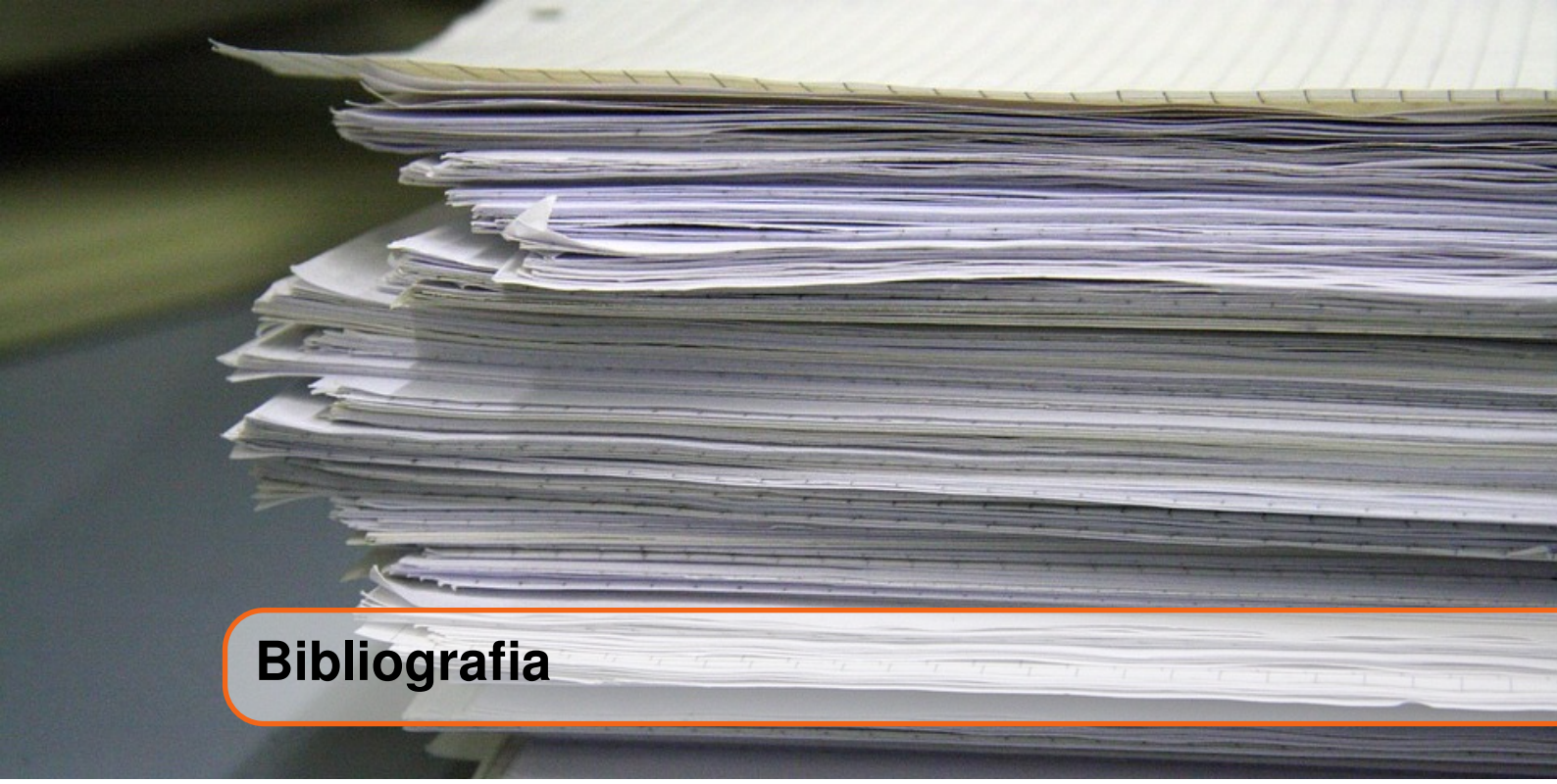
Dane wilgotności (rys. 10.3d) pokazują natomiast wysoki poziom korelacji z wyjątkiem jednego czujnika, zlokalizowanego w szkole w miejscowości Kotowice. Niskie wartości współczynników korelacji tego czujnika z pozostałymi mogłyby wprawdzie być tłumaczone wyższą wilgotnością w tej lokalizacji związaną z lokalnymi uwarunkowaniami, lecz wystąpienie wartości ujemnych wskazuje jednoznacznie na to, iż odczyty te nie poddają się okresowym zmianom wynikającym ze zjawisk meteorologicznych. W odniesieniu do tego parametru tego pojedynczego czujnika może być wymagana kontrola techniczna, połączona najprawdopodobniej z kalibracją.

Rejestrowane wartości ciśnienia (rys. 10.3e) mają podobną charakterystykę, co omawiane wcześniej poziomy pyłów zawieszonych, lecz tu rozbieżności są nieco wyższe, a ich charakterystyka wskazuje na pojawianie się ich okresowo. Poziom zbieżności jest jednak na tyle wysoki, iż nie powinno budzić to wątpliwości związanych z jakością samych odczytów.

W odniesieniu do odczytów temperatury powietrza (rys. 10.3f) zaobserwowano z kolei niższy poziom korelacji w odniesieniu do czujnika zlokalizowanego przy skrzyżowaniu ul. Łąkowej i ul. Piastowskiej w miejscowości Radwanice. Odczyty z czujnika są nieco niższe niż większości pozostałych, lecz zachowują podobną charakterystykę (por. sekcja 10.2).

Niniejszy przykład przedstawia korzyści wynikające z zaproponowanych, holistycznych metod analizy danych, w porównaniu do klasycznych metod statystycznych. Możliwość wykrycia związków zarówno w obrębie jednego systemu sensorycznego, jak i pomiędzy tymi sensorami (ukazana na przykładzie powiązania danych o jakości powietrza z danymi o natężeniu i organizacji ruchu) pozwala na łatwiejsze i bardziej spójne wyciąganie wniosków.

Obserwacje te mogą posłużyć do wspomagania podejmowania bieżących decyzji, ale także do planowania dalszej rozbudowy sieci sensorycznych – np. określania, które lokalizacje powinny zostać wyposażone w dodatkowe czujniki tak aby użyteczność inwestycji była jak największa. Wątek ten poruszono szerzej w jednej z opracowanych w ramach projektu publikacji [3].



Bibliografia

[1] Donald J. Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, AAAIWS'94, pages 359–370, Seattle, WA, July 1994. AAAI Press.

[2] B. F. Chao and C. H. Chung. On Estimating the Cross Correlation and Least Squares Fit of One Data Set to Another With Time Shift. *Earth and Space Science*, 6(8):1409–1415, 2019.

[3] Sebastian Ernst, Leszek Kotulski, Adam Sędziwy, and Igor Wojnicki. Graph-Based Computational Methods for Efficient Management and Energy Conservation in Smart Cities. *Energies*, 16(7):3252, January 2023.

[4] Sebastian Ernst, Konrad Zaworski, Piotr Sokołowski, and Grzegorz Salwa. Lessons Learned from a Smart City Project with Citizen Engagement. In *PP-RAI 2023*, Łódź, 2023.

[5] Krzysztof Hauke. Model komunikacji elektronicznej pomiędzy interesariuszami a jednostkami samorządu terytorialnego na poziomie gminy wykorzystujący technologie WEB 2.0. *Studia Informatica Pomerania*, 41:5–16, 2016.

[6] Leszek Kotulski and Maciej Gnela. Zasady pozyskiwania i udostępniania danych. In *Zarządzanie Danymi w Miastach: Podręcznik Dla Samorządów*. Kraków-Warszawa, 2021.

[7] Konrad Kulakowski. *Understanding the Analytic Hierarchy Process*. Boca Raton, FL, 11 listopada 2020.

[8] Konrad Kułakowski. On the Geometric Mean Method for Incomplete Pairwise Comparisons. *Mathematics*, 8(11):1873, November 2020.

[9] Charlene Li and Josh Bernoff. *Groundswell: Winning in a World Transformed by Social Technologies*. Harvard Business Press, 2011.

[10] Thomas L Saaty. A scaling method for priorities in hierarchical structures. *Journal of Mathematical Psychology*, 15(3):234–281, June 1977.

[11] Giorgino Toni. Computing and Visualizing Dynamic Time Warping Alignments in R: The dtw Package. *Journal of Statistical Software*, 31, August 2009.

[12] Jędrzej Wasiak-Poniatowski. Platformy komunikacji zewnętrznej gmin - analiza porównawcza. *Marketing i Zarządzanie*, 45:233–243, 2016.