POLITECHNIKA ŚLĄSKA

WYDZIAŁ INŻYNIERII MATERIAŁOWEJ

Kierunek: Informatyka Przemysłowa  
Profil praktyczny  
Rodzaj studiów: stacjonarne I stopnia

Projekt inżynierski  
Krzysztof MOCHOCKI

projekt i implementacja  
Systemu raportującEGO braki pomiędzy serwisami agregującymi publikacje naukowe napisany

Design and implementation of  
reporting system for missing articles between science publications aggregation sites

Kierujący pracą: Recenzent:

Dr inż. Adrian SMAGÓR dr hab. inż. Roman PRZYŁUCKI, prof. PŚ

Katowice, czerwiec 2021 r.

Spis treści

1. Wstęp 4

1.1. Inspiracja 4

1.2. Cel 4

2. Zakres Pracy 5

2.1. Przegląd Literaturowo - Technologiczny 5

2.1.1. Podobne Rozwiązania 5

2.1.2. Aktualny Model Synchronizacji 5

2.1.3. Dostępne źródła wiedzy w alternatywnych serwisach 7

2.2. Przegląd wybranych technologii 10

2.2.1. System operacyjny 10

2.2.2. Język programowania 10

2.2.3. Styl 11

2.2.4. System kontroli wersji 11

2.2.5. Dokumentacja 12

2.2.6. System budowy 13

2.2.7. Środowisko deweloperskie 13

2.2.8. Biblioteki 13

# Wstęp

## Inspiracja

Inspiracją dla niniejszego projektu inżynierskiego było dostrzeżenie problemu dotykającego pracowników naukowych Politechniki Śląskiej, którzy, aby móc dzielić się wynikami swoich prac z resztą świata nauki, zmuszeni są eksponować swój dorobek w różnych agregatach publikacji naukowych. Niestety w związku z różną datą pojawienia się agregatorów, oraz natłokiem innych zadań, ciężko jest dopilnować, aby wszystkie serwisy posiadały wszystkie publikacje.

## Cel

Celem projektu jest dostarczenie narzędzia pozwalającego zautomatyzować cały proces wyszukiwania brakujących publikacji wśród znanych i obsługiwanych serwisów.

Wśród obsługiwanych portali, powinny znaleźć się te, które posiadają najwięcej zbiorów, oraz te, które cieszą się największą popularnością. Dodatkowym kryterium jest możliwość dodania lub edycji danych w przeszukiwanym przez aplikację portalu. Przy założonych kryteriach wyłaniają się 4 portale:

* DOROBEK, dostępny pod adresem *https://www.bg.polsl.pl/expertus/new/bib*
* ORCID, dostępny pod adresem *https://orcid.org*
* SCOPUS, dostępny pod adresem *https://www.scopus.com/*
* Web Of Science, dostępny pod adresem *https://webofknowledge.com*

Docelowo program, ma generować prosty w interpretacji raport w powszechnie znanym formacie, pozwalający na jednoznaczne wskazanie, zbiorów publikacji, które należy uzupełnić w poszczególnym portalu internetowym. Formatem spełniającym te warunki oraz dodatkowo zapewniający przenaszalność i wiele narzędzi w jego obsłudze, jest otwarty format opracowany przez firmę *Microsoft* dla arkuszy kalkulacyjnych: *xlsx.* Biorąc pod uwagę możliwość pojawienia się nowych podobnych serwisów internetowych, projekt aplikacji powinien zapewniać łatwą rozszerzalność   
o nowe strony.

# Zakres Pracy

## Przegląd Literaturowo - Technologiczny

### Podobne Rozwiązania

W domenie publicznej niestety nie udało się znaleźć podobnych rozwiązań, co może być zrozumiałe z uwagi na charakter korzystania z tego typu serwisów. Wnioskując na podstawie dokumentacji, dostarczonej przez wyżej wymienione portale, programowalny interfejs jest udostępniany do integracji z wewnętrznymi zasobami ośrodków naukowych.

Zasoby tego typu, jeżeli będą otwarte, to nie będą skoncentrowane na twórczości pracowników Politechniki Śląskiej, lecz na swoich naukowcach. Fakt ten uniemożliwia korzystanie z potencjalnych portali, jako narzędzia do diagnozowania luk we wspieranych portalach, dla obcych badaczy.

### Aktualny Model Synchronizacji

W momencie rozpoczęcia prac brakowało narzędzi pozwalających na sprawną synchronizację publikacji naukowych pracowników Politechniki Śląskiej   
w internetowych bazach publikacji, takich jak: DOROBEK, SCOPUS, czy ORCID, ponieważ narzędzia, które miały im to umożliwić nie zapewniały wystarczających możliwości.

Dla pracowników Politechniki Śląskiej, punktem wyjścia ręcznego porównania spisów swoich publikacji w internetowych bazach, jest baza DOROBEK. Baza ta powinna zawierać wszystkie publikacje, pracowników Politechniki Śląskiej. Aby możliwe było w prosty i systematyczny sposób porównywać dane pomiędzy portalami potrzebna jest funkcjonalność prostego zbierania danych, czego baza DOROBEK nie zapewnia.

Naturalnym podejściem osoby niepotrafiącej programować, byłoby ręczne kopiowanie tytułów i wklejane ich do innych wyszukiwarek, celem weryfikacji ich istnienia. Temu podejściu nie sprzyja jednak mocno dynamiczny wygląd strony, który uniemożliwia sprawne zaznaczanie, interesującego ciągu znaków kursorem.

Alternatywą dla pierwszego sposobu mogłoby być skorzystanie z generacji pliku CSV, który jednak jest mocno niekompletny, między innymi przez brakujące numery ORCID, znacznie ułatwiające wyszukiwanie, a dane zarówno w nim, jak i na stronie są nierzadko zapisane w nieprzystępnej formie.

Często widywaną praktyką jest wpisywanie tytułu w języku polskim   
i nowożytnym po przecinku, średniku lub kropce, co generuje kolejny problem. Problem powodujący niejednoznaczność w dalszych poszukiwaniach, ponieważ pojawiają się dwie różne wartości dla jednego nie tablicowego pola. Strona, oprócz pliku w formacie CSV oferuje również generowanie pliku w formacie RTF, który względem poprzedniego pliku ma przewagę   
w postaci kompletnych danych.

Największe trudności związane z automatycznym poborem danych, z bazy DOROBEK jest brak możliwości zdalnego generowania plików CSV i RTF, ponieważ są one generowane przez skrypty klienckie, co wymusza przetwarzanie gołego kodu strony w języku HTML wraz ze wszystkimi stylami i definicjami skryptów klienckich.

Domyślną postawą podczas komunikacji strony internetowej wraz z bazą danych, jest korzystanie z programowalnego interfejsu aplikacji   
(ang. API – Application Programmable Interface), za pomocą protokołu *REST*, czy *GRPC*. Można przypuszczać, że w przypadku tego serwisu, również zachodzi taka wymiana, co potwierdza szybka analiza ruchu sieciowego strony. Skorzystanie jednak z wyłonionego zapytania, poprzez wyszukanie największego typu metody POST (Rysunek 1), również nie napawa optymizmem, ponieważ zapytanie to, zwraca kawałek strony internetowej wygenerowanej po stronie serwera.

Ponadto zapytanie zbudowane przez stronę, wygląda na pozór na zaszyfrowane (Rysunek 1), ponieważ w rzeczywistości, są to trudno identyfikowalne nazwy wprowadzonych pól. Stosowanie takiej praktyki sprawia, że wykorzystanie tego zapytania przez zewnętrzny serwis wymaga zarówno wcześniejszego zgadywania przez programistę za pomocą prób i błędów znaczenia poszczególnych pól, a także żmudnego i wymagającego sporych zasobów przetwarzania dostarczonych danych w formacie HTML.

Wielokrotne próby komunikacji z administratorem strony zakończyły się kategoryczną odmową dostępu do jakiegokolwiek punktu dostępowego dla osób trzecich, co wymusza na osobie zainteresowanej automatyzacją procesu weryfikacji spójności danych w innych serwisach na skorzystanie z przetwarzanie danych uzyskanych ze źródła przeznaczonego dla użytkownika przeglądarki internetowej.

Kolejne serwisy, między innymi ORCID, oraz SCOPUS zapewniają bardzo obszerne dokumentacje, z przykładami oraz dokładnymi objaśnieniami,   
w przeciwieństwie do bazy DOROBEK są przyjaźnie nastawione do użytkownika   
i oferują zwięzły panel użytkownik.

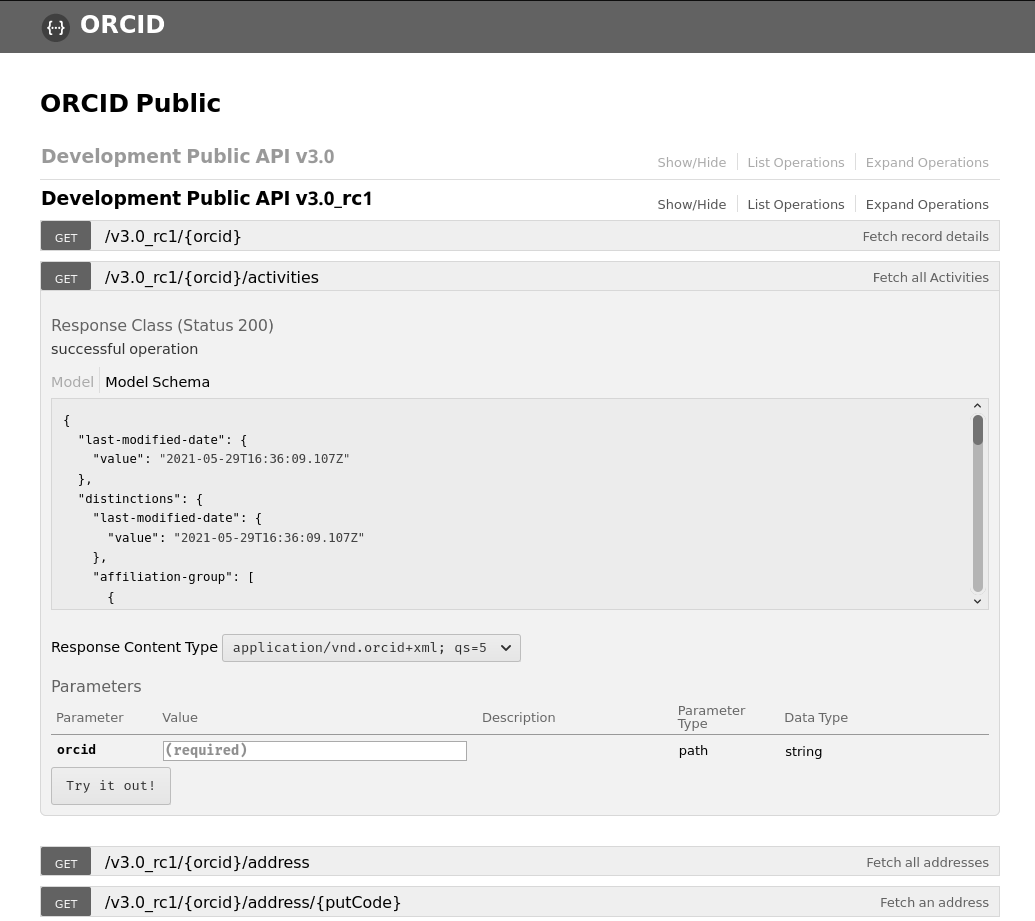


Rysunek zapytanie na jednym polu - autorze, wygenerowane przez serwis (skala szarości)

### Dostępne źródła wiedzy w alternatywnych serwisach

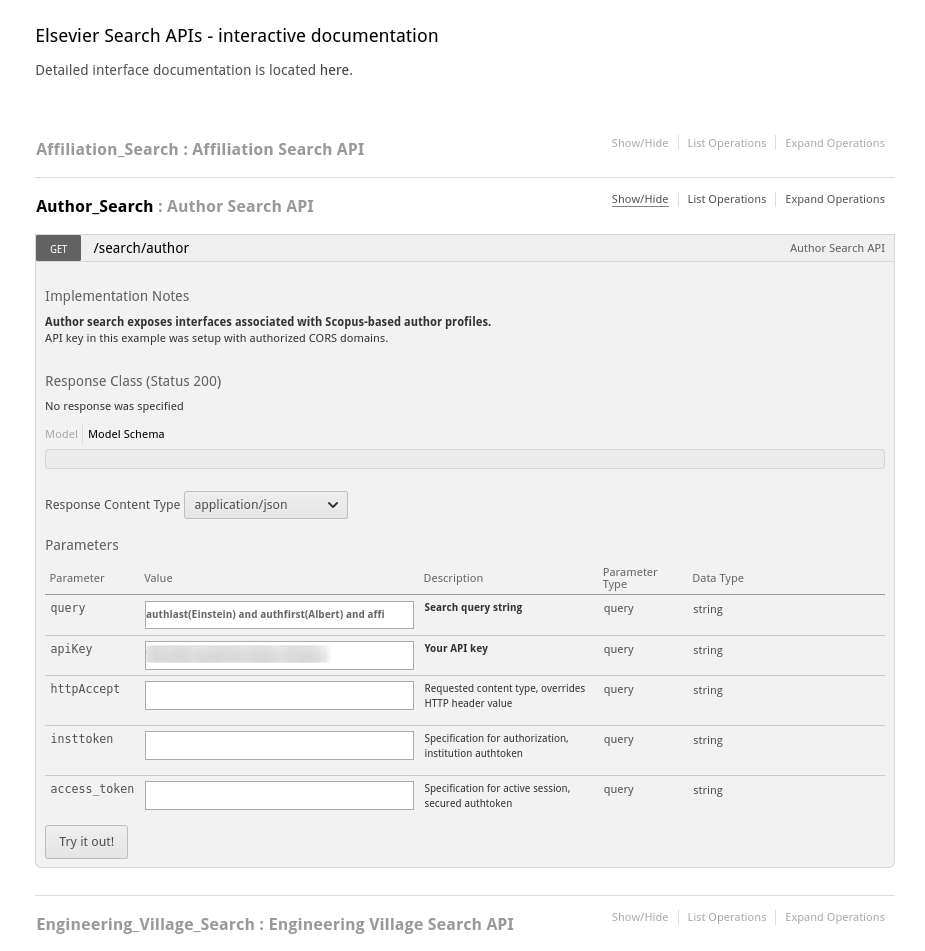
#### ORCID

Serwis ORCID zapewnia programiście szerokie wsparcie w zakresie obsługi dostarczonych przez serwis narzędzi. Przekonać można się o tym, logując się na stronie, a następnie przechodząc do zakładki narzędzie deweloperskich   
(ang. ‘Developer Tools’), która znajduje się w poręcznym miejscu przy ikonie użytkownika w prawym górnym rogu. Po przejściu na witrynę deweloperską, strona od razu przedstawia przydatne hiperłącza, kierujące zainteresowanego do obszernej dokumentacji wystawionego interfejsu sieciowego. Istnieje możliwość, zarejestrowania swojej aplikacji. Daje to możliwość wykorzystania z jednego   
z dostępnych protokołów podwyższających bezpieczeństwo komunikacji: *OAuth* lub *OAuth 2.*0. Skorzystanie z bezpieczniejszego połączenia jest wymagane przez serwis, aby móc przeprowadzić proces logowanie i wykonywać operacje na koncie poza stroną internetową.



Rysunek podgląd specyfikacji technicznej dostarczonej przez serwis ORCID (skala szarości)

Dzięki wykorzystaniu otwartego i sprawdzonego standardu tworzenia dokumentacji *Swagger*, (Rysunek 2) dokumentacja interfejsu sieciowego jest   
w znanej formie. Dodatkowo twórcy umożliwili przetestowanie punktów końcowych interfejsu, co umożliwia programiście zaplanowanie sposobu przetwarzania przychodzących danych. Kolejną istotną rzeczą jest wersjonowanie punktów końcowych. Daje to programiście spore poczucie bezpieczeństwa, że model przychodzących danych lub wygląd zapytania, nie zostanie nagle zmieniony. Polityka serwisu polega na dodaniu kolejnej wersji API w przypadku zmiany czegokolwiek   
w budowie zapytań lub w modelu przychodzących danych.



Rysunek podgląd specyfikacji technicznej dostarczonej przez serwis SCOPUS (skala szarości)

#### SCOPUS

Serwis SCOPUS za wyjątkiem wersjonowania dostarcza te same równie przyjemną interaktywną dokumentację, co serwis ORCID. Jednakże w porównaniu do poprzedniego zapewnia konkretne przypadki użycia swojego interfejsu   
z wykorzystaniem różnych języków, między innymi *Python*, czy *JavaScript*. Ponadto interaktywna dokumentacja generuje od razu polecenia *cURL*, które można wkleić do linii poleceń w systemie Linux i szybko sprawdzić sposób działania dostępnych punktów końcowych.

## Przegląd wybranych technologii

### System operacyjny

Manjaro – dystrybucja systemu operacyjnego GNU/Linux bazująca na dystrybucji Arch Linux, który słynie ze ścisłego trzymania się reguły KISS   
(ang.”Keep It Simple, Stupid”). Pośrednio wynika z tej zasady, jak wygląda zarządzanie pakietami w tej dystrybucji. Mimo istnienia menadżera pakietów *pacman*, spora część oprogramowania wymaga ręcznego budowania przez użytkownika na swoim urządzeniu. Wymóg ten miał spory wpływ na wybór zależności do samego programu, ponieważ wybór bibliotek dedykowanych pod aktualny system jest bardzo mały. Wymusza to na programiście wybór bibliotek i technologii, które są możliwe do zastosowania również na nietypowych systemach i cechujące się wysoką przenaszalnością. Implikuje to poprawienie przenaszalności całego projektu na różne platformy.

### Język programowania

C++ – elastyczny, szybki i wydajny język programowania, dający programiście dużo możliwości oraz stawiający sporo wyzwań podczas pisania w nim. Sam, spadkobierca języka C, jest dość stary i posiada wiele wersji, które, szczególnie te współczesne, mocno się od siebie różnią, stąd konieczność na wstępie jej sprecyzowania. Niniejszy projekt używa najnowszej wersji – 20, oznaczanej, jako C++20, lub C++2a w przypadku wydań eksperymentalnych.

Jednakże wersja języka nie jest jedyną rzeczą konieczną do sprecyzowania przy jego omawianiu. Konieczny jest jeszcze wybór kompilatora, który będzie zamieniał go na język maszynowy. Na wybranej platformie, dostępne są dwa popularne kompilatory: GCC oraz CLANG. Ostatecznym czynnikiem mającym wpływ na wybór kompilatora było wcześniejsze doświadczenia twórcy w pracy z GCC.

Jednym z wielu argumentów za wyborem tego języka jest jego wyżej wspomniana elastyczność. Jest to jeden z niewielu języków dający możliwość korzystania z wielu paradygmatów. Program tutaj opisywany, nie skupia się na kurczowym trzymaniu się jednego z nich, lecz oscyluje wokół trzech: obiektowego, funkcyjnego oraz [szablonowego] meta-programowania.

### Styl

*Notacja wężowa* (ang. *sneak case*) – zaraz obok wielbłądziej jeden   
z najbardziej rozpoznawalnych stylów używanych w kodzie. Wyborem stojącym za skorzystaniem z tej notacji jest chęć dostosowania się do tej, dostarczonej z biblioteki standardowej. Jedynym odstępstwem od jej ścisłego przestrzegania, są nazwy argumentów w szablonach, gdzie również w bibliotece standardowej można spotkać się z notacją wielbłądzią.

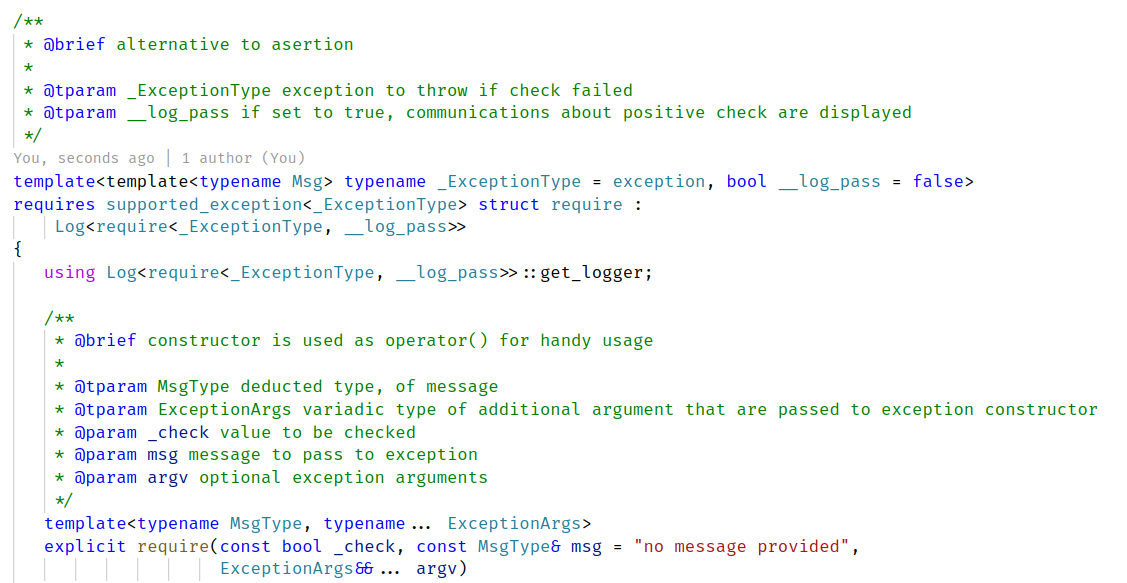
Oprócz samej notacji warto jest zadbać o jednolite formatowanie w całym projekcie, co poprawia czytelność i zwiększa wrażenie spójności wszystkich jego elementów. Celem zapewnienia zgodnego formatowania, należało zdecydować się na dedykowane, oprogramowanie. Najpopularniejszym, otwartym, rozwiązaniem jest program z pakietu CLANG o nazwie *clang-tidy*. Konfiguracja jest bardzo prosta, oparta na pojedynczym pliku tekstowym. Po uruchomieniu, program rekurencyjnie wyszukuje pliki według zadanego wyrażenia regularnego, a następnie podmienia na sformatowane.

### System kontroli wersji

git – popularny program, służący do kontroli wersji. Przy bardziej złożonych projektach jest to narzędzie obowiązkowe, z uwagi na łatwą możliwość powrotu do poprzednich wersji, możliwość łatwej współpracy z wieloma osobami, a także łatwą kontrolę zależności w postaci podmodułów. W połączeniu z serwisem GitHub, służącym, jako zdalne repozytorium, zapewnia bezpieczny ekosystem.

### Dokumentacja

Doxygen – narzędzie do automatycznego generowania dokumentacji w formie strony internetowej, plików w formacie *man* lub rtf. Korzyścią tego rozwiązania jest brak konieczności ręcznego tworzenia dokumentacji w osobnych plikach. Cała treść dokumentacji jest zapisana w formie komentarzy w całym kodzie źródłowym. Dodatkowo, wbudowane programy, do wspomagania programisty, takie jak *IntelliSense* korzystają z tych komentarzy, aby móc wyświetlić pomocne informacje, podczas tworzenia nowego kodu.



Rysunek fragment kodu z komentarzami do wygenerowania dokumentacji



Rysunek fragment wygenerowanej strony internetowej

### System budowy

cmake – otwarte, wieloplatformowe narzędzie do kontroli przebiegu budowy projektu oraz planowania testów. Oprogramowanie wykonuje instrukcje na podstawie przygotowanych plików, napisanych w dedykowanym języku skryptowym, za pomocą, którego można definiować cele do zbudowania w postaci bibliotek oraz plików wykonywalnych.

Dostarczony język daje również możliwość tworzenia niestandardowych celów budowy, co daje spore możliwości w zakresie tworzenia narzędzi wspomagających proces budowy, czy proces wytwarzania kodu.

W trakcie tworzenia niniejszej aplikacji została wydana nowsza wersja programu *cmake* – 3.20. Aplikacja natomiast używa wersji 3.19.

### Środowisko deweloperskie

Visual Studio Code – otwarty, wieloplatformowy edytor tekstowy zawierający zestaw narzędzi wspierających programistę. Jego mocnymi stronami jest ilość wtyczek dostępnych bezpośrednio z edytora, tworzonych przez społeczność. Dzięki nim edytor zapewnia wsparcie nawet najbardziej egzotycznym językom, zarówno   
w zakresie podpowiadania składni, debugowania, kontroli wersji czy budowania   
i uruchamiania.

### Biblioteki

#### Wspierająca; Narzędziowa

boost – otwarta, wieloplatformowa biblioteka narzędziowa, lustrzana do biblioteki standardowej, jednocześnie rozszerzająca ją o wiele funkcjonalności.   
W przypadku większych projektów jest to, niemalże obowiązkowa pozycja na liście bibliotek.

Argumentem stojącym za wykorzystaniem tej biblioteki jest również powszechność jej użycia w różnych projektach, co sprawia, że najczęściej znajduje się już zainstalowana w systemie. Wadą tej biblioteki, jest wymagający próg wejścia, zadany przez autorów. Przy domyślnych ustawieniach biblioteka jest kompilowana   
w całości, co przy słabszych maszynach, może okazać dość długim procesem.

#### Testowa

boost-ext/μt – eksperymentalna część wcześniej wspomnianej biblioteki boost, zawierająca pakiet narzędzi do tworzenia testów, wraz z raportowaniem. Biblioteka jest stworzona w duchu standardu C++17, dając możliwość pisania testów korzystając z dobrodziejstw najnowszych standardów.

Biblioteka posiada możliwość organizacji kodu, dzięki zaimplementowanym zestawom (ang. suites). Wewnątrz jednego zestawu można umieścić różne konfiguracje aplikacji, a następnie napisać różne przypadki testowe (ang. test cases), celem przetestowania założonych ustawień. Ciekawe jest podejście autorów, do sposobu deklaracji testów, które polega na używaniu literałów ciągów znakowych.

W samych testach, jako sprawdzeń, używa się bibliotecznych funkcji. Są one statycznie rozwiązywalne, z uwagi na oparcie o szablony, przeciwieństwie do biblioteki matki, która to używa makr z języka C. Takie rozwiązanie powoduje wykrycie części błędów, jeszcze na etapie kompilacji.

#### Sieciowa

drogoncpp – otwarta, sieciowa biblioteka napisana w nowoczesnym standardzie języka C++, dedykowana do pisania aplikacji serwerowych, jednakże równie dobrze sprawdza się, jako biblioteka kliencka.

Posiada pełne wsparcie dla rozwiązań wielowątkowych, a także posiada wsparcie dla korutyn. Bliskie trzymanie się biblioteki standardowej sprawia, że integracja z dostępnym interfejsem aplikacji jest bezproblemowa, a użytek intuicyjny. Użytkowanie biblioteki poprawia również dość szeroka dokumentacja, a także dostarczone przez społeczność seria przykładów użycia

© Elsevier B.V. . Dokumentacja Interfejsu Sieciowego SCOPUS. 2021. 30 06 2021 <https://dev.elsevier.com/search.html>.

an-tao. Repozytorium projektu drogoncpp. 10 04 2021. 31 06 2021 <https://github.com/an-tao/drogon>.

Bertocci, Vittorio. Bezpieczeństwo protokołu OAuth 2.0. 08 01 2019. 30 06 2021 <https://auth0.com/blog/oauth2-implicit-grant-and-spa/>.

Dokumentacja języka C++. 11 08 2020. 30 06 2021 <https://en.cppreference.com/w/>.

Gauniyal, Abhinav. Repozytorium projektu rang. 24 11 2018. 31 06 2021 <https://github.com/agauniyal/rang>.

Heesch, Dimitri van. Strona główna narzędzia Doxygen. 2021. 30 06 2021 <https://www.doxygen.nl/index.html>.

Microsoft. Specyfikacja techniczna formatu XLSX. n.d. 30 06 2021 <https://support.microsoft.com/pl-pl/office/specyfikacje-i-ograniczenia-programu-excel-1672b34d-7043-467e-8e27-269d656771c3>.

—. Strona główna projektu Visual Studio Code. 30 06 2021. 30 06 2021 <https://github.com/microsoft/vscode>.

Müller, Philip. Strona dystrybucji Manjaro. 2011. 30 06 2021 <https://manjaro.org/>.

ORCID. Dokumentacja Interfejsu Sieciowego ORCID. n.d. 30 06 2021 <https://pub.orcid.org/v3.0/>.

Repozytorium projektu boost-ext/μt. 13 05 2021. 30 06 2021 <https://github.com/boost-ext/ut>.

SMARTBEAR. Strona domowa Swagger. n.d. 30 06 2021 <https://swagger.io/>.

Specyfikacja notacji wężowej. 19 05 2021. 30 06 2021 <https://en.wikipedia.org/wiki/Snake\_case>.

Specyfikacja notacji wielbłądziej. 24 05 2021. 30 06 2021 <https://en.wikipedia.org/wiki/Camel\_case>.

Specyfikacja plików CSV. n.d. 30 06 2021 <https://en.wikipedia.org/wiki/Comma-separated\_values>.

SPLENDOR; Biblioteka Główna Politechniki Śląskiej. Strona Główna Wyszukiwarki Bazy Dorobek. 29 05 2021. 30 06 2021 <https://www.bg.polsl.pl/expertus/new/bib/>.

Strona główna programu git. 2021. 30 06 2021 <https://git-scm.com/>.

Strona projektu Boost. 11 12 2020. 30 06 2021 <https://www.boost.org/>.

The Clang Team. Dokumentacja programu clang-tidy. 2021. 30 06 2021 <https://clang.llvm.org/extra/clang-tidy/>.

Vinet, Judd, Aaron Griffin and Levente Polyák. Strona dystrybucji Archlinux. 2002. 30 06 2021 <https://archlinux.org/>.

[Rysunek 1 zapytanie na jednym polu - autorze, wygenerowane przez serwis (skala szarości) 7](file:///Z:\home\raidg\proj\polsl\antybiurokrata\engeener_docx\engeneer_project.docx#_Toc73326280)

[Rysunek 2 podgląd specyfikacji technicznej dostarczonej przez serwis ORCID (skala szarości) 8](file:///Z:\home\raidg\proj\polsl\antybiurokrata\engeener_docx\engeneer_project.docx#_Toc73326281)

[Rysunek 3 podgląd specyfikacji technicznej dostarczonej przez serwis SCOPUS (skala szarości) 9](file:///Z:\home\raidg\proj\polsl\antybiurokrata\engeener_docx\engeneer_project.docx#_Toc73326282)

[Rysunek 4 fragment kodu z komentarzami do wygenerowania dokumentacji 12](file:///Z:\home\raidg\proj\polsl\antybiurokrata\engeener_docx\engeneer_project.docx#_Toc73326283)

[Rysunek 5 fragment wygenerowanej strony internetowej 12](file:///Z:\home\raidg\proj\polsl\antybiurokrata\engeener_docx\engeneer_project.docx#_Toc73326284)