Uniwersytet Zielonogórski

Wydział Informatyki, Elektrotechniki i Automatyki

Praca dyplomowa

Kierunek: Informatyka

Modernizacja ogólnouczelnianego systemu informatycznego do zarządzania oraz ewidencji publikacji (SKEP)

Dawid Konarczak

Promotor:

Dr hab. inż. Remigiusz Wiśniewski Prof. UZ

Zielona Góra, styczeń 2024

Streszczenie

Niniejsza praca stanowi kompleksową analizę oraz modernizację Systemu Komputerowej Ewidencji Publikacji (SKEP), który pełni kluczową rolę w zarządzaniu i rejestrowaniu publikacji. Obecna wersja systemu, oparta na przestarzałych technologiach, stwarza liczne wyzwania, zarówno w kwestii wprowadzania i edycji danych, jak i w kontekście zagrożeń związanych z bezpieczeństwem informacji. Głównym założeniem projektu jest stworzenie zupełnie nowej wersji SKEP w formie nowoczesnej aplikacji webowej, która spełniać będzie najwyższe standardy wydajności, bezpieczeństwa oraz dostępności. Celem jest nie tylko zapewnienie bezpiecznej infrastruktury, ale także elastyczności i przenośności, umożliwiającej użytkownikom korzystanie z systemu na różnych platformach. Wewnątrz projektu backend systemu bazuje na języku PHP w połączeniu z frameworkiem Laravel [1], co gwarantuje efektywne zarządzanie danymi oraz udostępnianie aplikacji użytkownikom. Frontend natomiast wykorzystuje popularną bibliotekę Bootstrap 5, co zapewnia nie tylko atrakcyjny wizualnie, ale także intuicyjny interfejs użytkownika, co znacząco podnosi jakość obsługi systemu. Baza danych, na której operuje projekt, to Oracle SQL [2], umożliwiając efektywne zarządzanie danymi w bazie oraz ich wprowadzanie. Ważnym aspektem modernizacji jest zapewnienie bezpieczeństwa danych i ograniczenie dostępu jedynie do upoważnionych użytkowników. W tym celu projekt korzysta z istniejącego systemu PracNet, który stanowi wewnętrzny system kont pracowniczych Uniwersytetu Zielonogórskiego. Ta integracja zapewnia nie tylko ochronę danych, ale również pełną kontrolę nad nimi, przy jednoczesnym ograniczeniu dostępu do zatrudnionych pracowników uczelni. Projekt ma na celu znaczną poprawę wydajności i użyteczności Systemu Komputerowej Ewidencji Publikacji na Uniwersytecie Zielonogórskim. Jego realizacja przyczyni się do efektywniejszego zarządzania publikacjami, poprawy pracy uczelni oraz zabezpieczenia cennych danych instytucji, co przekłada się na jej sukces i rozwój.

Słowa kluczowe: Bootstrap, PHP, SKEP, Laravel, CSS, SQL

Spis treści

Spis rysunków

Rozdział 1

Wstęp

1.1. System SKEP

System Komputerowej Ewidencji Publikacji (SKEP) aktualnie działa na przestarzałym systemie stworzonym w narzędziu Oracle Forms Builder [3] (część aplikacji z której korzysta Biblioteka Uniwersytetu Zielonogórskiego do wprowadzania publikacji), który to jest tzw. RAD-em (z ang. Rapid Application Development), aplikacja ta umożliwia szybkie tworzenie interfejsu użytkownika który umożliwia dostęp do bazy danych Oracle SQL. Początkowo aplikacja nazywała się 'Interactive Application Facility' i została udostępniona w roku 1981. Aktualny system na Uniwersytecie Zielonogórskim powstał w 2000 roku (korzysta z wersji Oracle Forms Builder z roku 1996) i od tamtej pory nie był aktualizowany (poza drobnymi aktualizacjami związanymi ze zmianą funkcji formularzy lub ich dodawaniu jednak te zmiany były czysto kosmetyczne i nie powodowały naprawy luk bezpieczeństwa samej aplikacji), co po pewnym czasie zaczęło powodować problemy z przenoszeniem programu oraz aktualizacją systemu operacyjnego użytkownika docelowego, gdyż aplikacja stworzona w 2000 nie jest uruchamiana przez najnowszy system operacyjny Windows 11. Stwarza to ogromne ryzyko bezpieczeństwa w postaci luk zabezpieczeń systemu operacyjnego (ostatnia wersja systemu Windows, która bezproblemowo uruchamia aplikacje stworzone w Oracle Forms Builder z roku 1996 to Windows 7), właśnie dlatego aby zapewnić maksymalną przenośność, wydajność oraz bezpieczeństwo system SKEP zostanie przeniesiony do aplikacji webowej zbudowanej na frameworku PHP oraz Laravel w istniejącej już aplikacji do wyświetlania publikacji dla pracow-

ników badawczych oraz badawczo-dydaktycznych zatrudnionych na Uniwersytecie Zielonogórskim. Sam system SKEP służy do ewidencji publikacji na Uniwersytecie Zielonogórskim czyli:

- artykuły w czasopismach (AWCZ);
- wydawnictwa Zwarte (WZ);
- wydawnictwa Zwarte Część (WZCZ);
- konferencje (KONF);
- redakcje (RED).

Elementy podane powyżej są wyświetlanie w wygodnej dla użytkownika formie na jego indywidualnej podstronie w systemie SKEP, do której użytkownicy mogą w łatwy sposób przejść poprzez link znajdujący się na profilu pracownika uczelni. Dostęp do zestawienia publikacji danego naukowca ma każda osoba nawet spoza uczelni, wystarczy, że posiada ona przeglądarkę oraz dostęp do internetu i przejdzie na adres: `https://pers.uz.zgora.pl/` a następnie znajdzie, korzystając z wyszukiwarki, interesującego ją naukowca, po czym przejdzie za pomocą linku na profilu pracownika do systemu SKEP, gdzie zostanie jej przedstawiony dorobek naukowy wybranej osoby.

1.2. Cel i zakres pracy

Celem pracy było zaprojektowanie webowego systemu do wprowadzania danych o publikacjach pracowników naukowych uczelni tak, aby wprowadzanie danych było możliwe z każdego urządzenia które, posiada dostęp do najnowszej wersji przeglądarki oraz internetu, dzięki czemu będzie można bezproblemowo aktualizować systemy operacyjne komputerów BU a co za tym idzie zwiększyć bezpieczeństwo.

Praca swym zakresem obejmowała:

- Analizę działania aktualnego systemu SKEP (Oracle Forms);
- Zaprojektowanie aplikacji na podstawie poprzedniej wersji systemu SKEP;
- Implementację aplikacji zgodnie z założeniami poprzedniej wersji SKEP.

1.3. Struktura pracy

Niniejsza praca została podzielona na cztery rozdziały. Tenże rozdział stanowi wprowadzenie ogólne do tematyki pracy.

Rozdział 2. jest przedstawieniem aktualnego systemu SKEP oraz omówieniem jego działania wraz z planowanymi zmianami w systemie oraz technologiami, które zostały zastosowane w pracy.

Rozdział 3. to etap pracy przedstawiający praktyczne rozwiązania napotkanych problemów inżynieryjnych, które wystąpiły podczas wdrażania rozwiązania oraz jego testowania.

Rozdział 4. jest krótkim podsumowaniem co zostało wykonane oraz jakie są możliwości dalszego rozwoju aplikacji.

Rozdział 2

Wprowadzenie teoretyczne

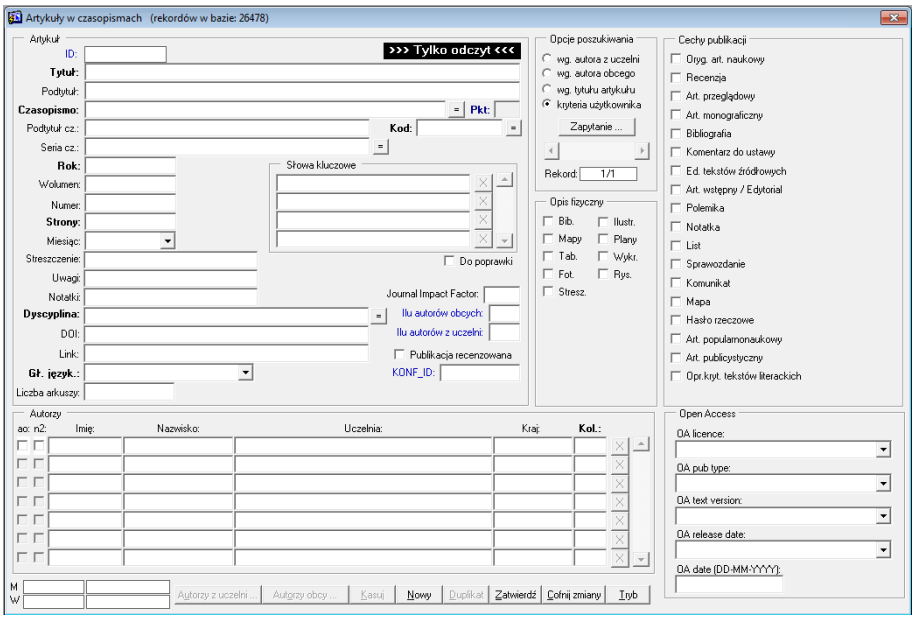
2.1. Czym jest i jak działa SKEP?

SKEP "System Komputerowej Ewidencji Publikacji", to uniwersytecka baza danych przechowująca informacje odnośnie dorobku naukowego pracowników, doktorantów oraz studentów Uniwersytetu Zielonogórskiego. Poza właściwym katalogowaniem osiągnięć naukowych, system zawiera wiele innych przydatnych informacji, w tym m.in. szczegółowe dane konkretnej publikacji czy czasopisma (aktualne wskaźniki parametryczne, liczba punktów ministerialnych, itp.). Dane te są niezwykle przydatne zarówno do przeprowadzenia oceny uczelni (dorobek naukowy dyscyplin), jak i pracowniczej (dla konkretnej osoby). Ocena odbywa się na podstawie punktów, które przydzielone są do danej pracy naukowej. Artykuły w czasopismach otrzymają taką liczbę punktów, jaką ustaliło Ministerstwo Edukacji i Nauki na początku danego roku kalendarzowego. Przykładowo jeśli praca naukowa pojawiła się w 2023 roku, to obowiązuje tą pracę "lista punktów"za rok 2023. Głównym i najważniejszym dokumentem w tym przypadku jest wykaz czasopism naukowych i recenzowanych materiałów z konferencji międzynarodowych, na podstawie którego określana jest liczba przydzielonych punktów dla danej pracy naukowej (artykułu w czasopiśmie lub konferencji). Większość uczelni korzysta z takiego systemu oceny pracowników, gdyż punkty przyznawane podczas ewaluacji stanowią jedną z kilku składowych oceny dziedziny prowadzonej na uczelni, co przekłada się na prestiż samej uczelni.

Jak widać, SKEP jest jednym z bardziej istotnych systemów wspomagających pracę Uniwersytetu. Dlatego też istotne, aby system był bezpieczny i szybki. Stąd właśnie zrodził się pomysł na niniejszą pracę dyplomową, która ma na celu usprawnienie funkcjonalności oraz zwiększenie bezpieczeństwa SKEP.

2.2. Analiza aktualnej struktury SKEP

Aktualnie SKEP jest oparty na rozwiązaniu Oracle Forms, w którym dane są wprowadzane za pomocą aplikacji przygotowanej w Oracle Forms Builder z roku 1996. Rozwiązanie to jest problematyczne ze względu na brak możliwości łatwego przenoszenia aplikacji między systemami oraz ze względu na potencjalne luki bezpieczeństwa, które mogą umożliwić osobom trzecim edycję oraz wprowadzanie publikacji bez wiedzy Biblioteki Uniwersyteckiej. Poniżej omówiono aktualny układ pól oraz praktyczną funkcjonalność wybranych elementów systemu SKEP.



Rysunek 2.1. Rzeczywisty wygląd aktualnej aplikacji

Na rysunku 2.1 przedstawiono formularz Oracle Forms do wprowadzania artykułów w czasopismach. Układ elementów w tym formularzu można podzielić na siedem różnych elementów, każdy z nich obsługuje inne elementy publikacji. Najważniejsze z siedmiu elementów są dwa czyli "Artykuł"(lewe górne pola) oraz "Autorzy"(lewe dolne pola pod zakładką "Artykuł"), te elementy odpowiadają za wprowadzanie da-

nych na temat AWCZ takich jak tytuł, podtytuł, czasopismo w którym artykuł został opublikowany, rok publikacji, strony na których znajduje się artykuł, dyscyplina oraz główny język, kod - są to najważniejsze pola z całej zakładki "Artykuł". Oprócz najważniejszych pól znajdują się tu również mniej istotne pola takie jak słowa kluczowe artykułu, podtytuł artykułu, wolumen, numer lub nawet wewnętrzne notatki i uwagi, które nie są widoczne dla użytkownika docelowego w webowej reprezentacji artykułu. Pola najbardziej istotne zaznaczone są pogrubioną czcionką.

Kolejnym istotnym elementem formularza do wprowadzania AWCZ jest element "Autorzy"w którym to dokonuje się wprowadzania autorów danego artykułu którzy później zostaną przedstawieni w webowej wersji opisu bibliograficznego. Autorów użytkownik wprowadza poprzez dopisanie ich do listy, przy czym podczas wpisywania autorzy są kojarzeni z autorami z uczelni z tabeli, w której znajduje się lista pracowników uczelni lub, jeśli autor jest oznaczony jako autor obcy (lewa strona checkbox o nazwie "ao"), wtedy autorzy są pobierani z tabeli "autorzy_obcy", w której to wpisywani są autorzy spoza uczelni, jeśli autor obcy nie znajduje się na liście korzysta się z innego formularza w którym wprowadza się właśnie autorów obcych uczelni. Dane z zakładki "Autorzy"następnie są wpisywani do tabeli "autorzy_publikacje", gdzie przechowywane są ID publikacji oraz ID autorów, tabela jest połączona za pomocą kluczy obcych do listy pracowników uczelni oraz do listy autorów z obcych uczelni. Rozwiązanie to będzie za wszelką cenę utrzymywane w wersji webowej aplikacji do wprowadzania publikacji, gdyż pozwala ono na wygodne wprowadzanie danych oraz zapewnia brak duplikatów w bazie jak i utrzymuje porządek w strukturach bazy.

Pozostało pięć elementów do omówienia, ‘Opis fizyczny‘ oraz ‘Cechy publikacji‘, te dwa elementy odpowiadają za określenie, jakie fizyczne elementy wchodzą w skład danego artykułu oraz czym cechuje się dana publikacja - czy jest na przykład artykułem popularnonaukowym. W obydwu zakładkach zasada działania jest taka sama i wystarczy zaznaczyć checkbox, aby w bazie została zapisana jedynka przy odpowiednim polu odnoszącym się do właśnie tej cechy lub opisu fizycznego, dane te nie są wyświetlane w opisie bibliograficznym w webowej wersji wyświetlającej publikacje.

Zakładka ‘Open Access‘ odpowiada za wybór na jakiej licencji dana publikacja

została opublikowana i jest to informacja, która znajduje się w docelowym opisie bibliograficznym publikacji. Do wyboru są elementy z listy (poza polem ‘OA date‘, które jest datą), dane które w nich (listach) się znajdują są pobierane z bazy danych a następnie przedstawiane właśnie w formie wygodnego pola wyboru.

U dołu strony dodawania publikacji znajdują się przyciski, które pozwalają na odpowiednią manipulacją stanem formularza oraz umożliwiają podgląd do danych na temat, kto ostatnio modyfikował aktualnie dodawany lub edytowany wpis oraz kto dany wpis dodał (pod warunkiem że znajdujemy się w trybie edycji), pola pokazujące osoby modyfikujące oraz wprowadzające to pola obok liter ‘M‘ oraz ‘W‘, gdzie odpowiednio pierwsza odpowiada słowu ‘modyfikował‘ a druga słowu "wprowadził"a w polu stosowane są skróty od użytkowników, którzy wprowadzali dane do bazy (na podstawie kont stworzonych bezpośrednio w Oracle data base). Obok skróconych nazw użytkownika znajduje się data modyfikacji oraz wprowadzenia również odpowiednio przy literach "M"oraz "W". Dalej znajdują się już przyciski które umożliwiają wybór autora z uczelni oraz autora obcego (oba przyciski mają dostęp do danych w bazie i wyświetlają je odpowiednio w formie osobnego okna z listą autorów obcych lub z uczelni). Następnie są przyciski "Kasuj", "Nowy"oraz "Duplikat"które umożliwiają jak sama nazwa wskazuje na wykasowanie aktualnie edytowanego/wprowadzanego wpisu, na utworzenie nowego wpisu lub zdublowanie aktualnie wprowadzanego wpisu, tak aby można było go niezależnie modyfikować ale nie przepisywać danych które zostały wprowadzone już wcześniej. Ostatecznie na końcu są już przyciski "Zatwierdź", "Cofnij zmiany"oraz "Tryb", przy czym pierwszy z nich odpowiada za zapis wprowadzonych danych do bazy, drugi cofa dokonane zmiany podczas edycji a ostatni odpowiada za tryb w jakim aplikacja się znajduje, czyli pozwala na włączenie lub wyłączenie trybu "tylko do odczytu", który zabezpiecza użytkownika przed przypadkowym nadpisaniem lub edycją danych, które niekoniecznie chciał edytować.

Pozostałym elementem do omówienia jest element "Opcje poszukiwania"który pozwala użytkownikowi na filtrowanie danych znajdujących się w bazie danych oraz pokazuje na którym rekordzie z ilu znajduje się aktualnie użytkownik oraz umożliwia na szybkie przechodzenie między filtrowanymi rekordami w bazie danych. Użytkownik może przeszukiwać dane na podstawie własnych kryteriów (opcja "kryteria użyt-

kownika") lub wybrać jeden z trzech najczęściej używanych kryteriów (według autora z uczelni, według autora z uczelni obcej lub według tytułu artykułu), użytkownik aby dokonać filtrowania po wprowadzeniu warunków filtrowania klika przycisk "Wykonaj", który to pojawia się po wciśnięciu przycisku "Zapytanie".

Jak widać aplikacja jest aktualnie bardzo intuicyjna dla użytkownika docelowego i taki zamysł (aplikacja musi być intuicyjna) będzie najważniejszym elementem niniejszej pracy, tak aby aktualni użytkownicy po przejściu na nową wersję aplikacji webowej mogli w jak najszybszym czasie i w jak najbardziej wygodnej formie rozpocząć pracę z nową wersją systemu, która ma być również łatwo przenośna oraz bezpieczna a przede wszystkim nowoczesna, a co za tym idzie musi spełniać nowoczesne standardy aplikacji webowej oraz być jak najmniej zawodna i stabilna oraz dokonywać autoryzacji użytkownika docelowego tak, aby osoby trzecie nie mogły dokonywać edycji danych w bazie.

2.3. Zastosowane technologie oraz ich wpływ na bezpieczeństwo

W tej części pracy zostaną omówione wykorzystane technologie które używane są w projekcie webowej aplikacji SKEP. Najważniejszą technologią która będzie wykorzystywana jest język PHP [4], aplikacja korzystać będzie z wersji PHP 7.2.34, czyli wersja języka PHP z pierwszego października dwa tysiące dwudziestego roku, a więc wersja sprzed 3 lat (w momencie pisania pracy tj. pierwszy listopad dwa tysiące dwudziestego trzeciego roku). Dzięki zastosowaniu jednej z najnowszych wersji języka PHP mamy gwarancję bezpieczeństwa oraz odporności na luki CVE, które występują jednak tylko przy niektórych funkcjach jednak funkcje te nie są wykorzystywane w aplikacji, dzięki czemu można powiedzieć, że strona będzie odporna na ewentualne ataki z zewnątrz. Dodatkowo jest opcja wyłączenia ogólnego dostępu do strony autoryzacji oraz ukrycie linku dostępowego oraz jego blokada dla użytkowników spoza sieci uczelni, co jeszcze bardziej zwiększyłoby bezpieczeństwo (docelowo można nawet zablokować dostęp wewnątrz uczelni do adresów tylko z biblioteki uniwersyteckiej).

2.3.1. Laravel

Kolejną technologią, która jest zastosowana w aplikacji jest framework Laravel w wersji 6.0.3 z roku dwa tysiące dziewiętnastego z trzeciego września, która to nadal otrzymuje łatki luk bezpieczeństwa, dzięki czemu Laravel staje się kolejną warstwą zabezpieczającą aplikacje nie tylko przed błędami samego PHP jak i przed błędami samego programisty (Laravel dostarcza wiele łatwych w użyciu narzędzi, które między innymi zabezpieczają aplikację przed SQL Injection jak i przed wykonywaniem kodu PHP/HTML/JS po stronie serwera), co znacząco podnosi poziom bezpieczeństwa aplikacji przed wykradaniem lub modyfikacją danych, ale również zabezpiecza innych użytkowników aplikacji przed atakami chociażby XSS (z ang. Cross Site Scripting, czyli wstrzykiwanie kody Javascript [5] w przykładowo pola wyświetlane użytkownikowi, dzięki czemu atakujący może w łatwy sposób wykradać dane użytkownika docelowego aplikacji bez jego wiedzy i zgody) lub innymi które stawiają dane użytkownika w jak i jego samego niebezpieczeństwo.

Poza zaletami związanymi ze zwiększonym bezpieczeństwem framework daje wiele przydatnych narzędzi które można praktycznie wykorzystać w aplikacji, co przekłada się na zmniejszony nakład pracy programisty oraz na szybszy czas realizacji docelowego projektu, mówiąc prościej Laravel daje możliwość szybkiego, wygodnego oraz bezpiecznego budowania aplikacji webowej, zajmując się kwestiami, które normalnie zajęłyby programiście wiele czasu oraz wymagały dużego nakładu pracy. Framework laravel korzysta z Symfony, czyli innego frameworka który jest również wykorzystywany do tworzenia aplikacji PHP-owych, obydwa frameworki korzystają z wzorca projektowego MVC (Model - View - Controller), co również wpływa na szybkość i wygodę pracy programisty a jednocześnie jest zgodne z najnowszymi standardami na rynku IT, wzorzec ten skutecznie pozwala podzielić aplikację na wcześniej wymienione widoki, modele oraz kontrolery, dzięki czemu można zachować przejrzystą strukturę aplikacji oraz stosować praktyki czystego kodu takie jak SOLID, KISS i do nich podobne.

Laravel jest o tyle wygodny, że pozwala pominąć czasochłonny proces przygotowywania Symfony i niejako umożliwia nam tworzenie aplikacji od razu bez potrzeby konfiguracji oraz układania plików oraz folderów w odpowiedni sposób. Symfony stosuje się zazwyczaj w dużych projektach, z których korzystać będzie wielu użyt-

kowników końcowych gdyż daje ono dużą elastyczność tworzenia rozwiązania webowego, zaś Laravel jest używany w mniejszych projektach, z których korzystać będzie mniejsza liczba ludzi i jest jednocześnie mniej elastyczny od Symfony gdyż ma z góry narzucone zasady i struktury projektu i właśnie z tych dwóch powodów Laravel został wykorzystany do stworzenia całego systemu PERS oraz SKEP (wersja wyświetlająca opisy bibliograficzne), w którym również docelowo znajdzie się aplikacja SKEP przez którą wprowadzane będą dane na temat publikacji do bazy danych, aby później mogły one zostać wyświetlone w SKEP (wersji wyświetlającej opisy bibliograficzne). Podsumowując, można wymienić takie zalety Laravela jak:

- **Elokwentny ORM (Object-Relational Mapping):** Laravel wykorzystuje Eloquent, który umożliwia pracę z bazą danych za pomocą prostego i czytelnego kodu PHP. Ułatwia to zarządzanie danymi i zapytaniami do bazy danych.
- **System routingu:** Laravel oferuje elastyczny i intuicyjny system routingu, co ułatwia definiowanie ścieżek URL i mapowanie ich do odpowiednich kontrolerów.
- **System szablonów Blade:** Blade to silnik szablonów używany w Laravelu, który zapewnia przejrzystą i wydajną składnię dla generowania widoków. Wsparcie dla dziedziczenia, sekcji i komponentów sprawia, że jest to potężne narzędzie do tworzenia interfejsów użytkownika.
- **Autentykacja i autoryzacja:** Laravel dostarcza gotowe mechanizmy autentykacji i autoryzacji, co ułatwia zabezpieczanie aplikacji oraz zarządzanie uprawnieniami użytkowników.
- **Middleware:** Middleware w Laravelu pozwala na przetwarzanie żądań HTTP przed dotarciem do kontrolera. To doskonałe narzędzie do wykonywania operacji takich jak autentykacja, logowanie, zabezpieczanie dostępu, itp.
- **Zintegrowane narzędzia do obsługi zapytań HTTP:** Laravel udostępnia klasę Request i Response, co ułatwia zarządzanie danymi HTTP, walidację żądań i generowanie odpowiedzi.

2.3.2. Bootstrap

Aplikacja korzysta również z Bootstrapa [6], czyli biblioteki CSS, która została stworzona oraz jest nadal utrzymywana przez programistów Twittera. Bootstrap umożliwia tworzenie eleganckiego designu aplikacji webowej w bardzo szybki i wygodny sposób. Sam bootstrap nie wpływa docelowo na bezpieczeństwo aplikacji, jednak umożliwia bardzo łatwe tworzenie nowocześnie wyglądających formularzy do wprowadzania danych jak i daje szanse na łatwe ostylowanie pozostałych elementów aplikacji, tak aby ta wyglądała profesjonalnie. Wykorzystana wersja Bootstrapa w projekcie to wersja 4.3.1, czyli wersja z trzynastego lutego dwa tysiące dziewiętnastego roku. Najważniejsze cechy Bootstrapa jakie można wymienić to:

- **System siatkowy:** Bootstrap oferuje elastyczny i skalowalny system siatkowy oparty na klasach CSS, co ułatwia tworzenie responsywnych układów strony, dostosowanych do różnych rozmiarów ekranów.
- **Gotowe komponenty:** Biblioteka zawiera wiele gotowych komponentów takich jak nawigacje, przyciski, formularze, panele, karty, modale i wiele innych. Dzięki nim projektanci i deweloperzy mogą szybko tworzyć UI.
- **Stylizacja:** Bootstrap dostarcza predefiniowane klasy CSS, które pozwalają na łatwe dostosowanie wyglądu elementów, takie jak kolory, typografia, marginesy, wypełnienia i wiele innych.
- **RWD (Responsive Web Design):** Bootstrap jest zaprojektowany z myślą o responsywnym projektowaniu stron, co oznacza, że strony internetowe tworzone przy użyciu Bootstrap automatycznie dostosowują się do różnych urządzeń, takich jak telefony komórkowe, tablety i desktopy.
- **Wsparcie dla różnych przeglądarek:** Bootstrap zapewnia spójny wygląd i zachowanie w wielu popularnych przeglądarkach internetowych, co ułatwia rozwijanie stron bez konieczności długotrwałego testowania i dostosowywania.
- **Biblioteka JavaScript:** Bootstrap zawiera bibliotekę JavaScript, która umożliwia dodawanie interaktywności do stron internetowych, takie jak karuzele, akordeony, okna modalne, rozwijane menu i wiele innych.

- **Mobilne menu nawigacyjne:** Biblioteka oferuje gotowe rozwiązania do tworzenia responsywnych menu nawigacyjnych, co ułatwia nawigację na stronie internetowej, zwłaszcza na urządzeniach mobilnych.
- **Wsparcie dla niestandardowych projektów:** Bootstrap jest elastyczny i umożliwia dostosowanie do niestandardowych projektów. Deweloperzy mogą dostosować i nadpisać stylowanie oraz komponenty, aby spełnić konkretne wymagania projektu.
- **Wsparcie społeczności i dokumentacja:** Bootstrap cieszy się ogromną społecznością użytkowników i deweloperów, co oznacza, że istnieje wiele dostępnych źródeł wiedzy, dokumentacji oraz gotowych rozwiązań, które ułatwiają pracę.

Wszystkie powyższe zalety Bootstrap-a powodują, że jest on używany w poniższym projekcie a jego zalety wpłyną na szybsze oraz bardziej efektywne implementowanie elementów frontendu aplikacji.

2.3.3. Oracle Database

Projekt potrzebuje również bazy danych, w której to będą znajdować się wszystkie publikacje wprowadzane przez użytkownika, w tym przypadku oczywistym wyborem jest baza danych od Oracle, gdyż Oracle Forms są bardzo silnie skorelowane z bazą danych tego samego producenta, dodatkowo nie trzeba będzie wprowadzać wielu zmian na poziomie relacji między tabelami, co będzie wymagało mniejszego nakładu pracy, gdyż aktualny stan bazy danych nie różni się mocno od stanu docelowego, który będzie wykorzystywany przez aplikację webową, do głównych zalet Oracle Database można zaliczyć:

- **Niezawodność i wydajność:** Oracle DB jest znany z wysokiej niezawodności i wydajności. Jest to kluczowe w przypadku systemów, które wymagają ciągłej dostępności i efektywnego zarządzania danymi.
- **Zarządzanie transakcjami:** Oracle DB oferuje zaawansowany system zarządzania transakcjami, co sprawia, że jest idealnym rozwiązaniem do aplikacji wymagających spójności danych i kontroli nad transakcjami.

- **Wsparcie dla wielu platform:** Oracle DB jest dostępny na wielu platformach, co umożliwia organizacjom wybór odpowiedniego systemu operacyjnego i sprzętu.
- **Wsparcie dla języka SQL:** Oracle DB stosuje standardowy język zapytań SQL, co ułatwia pracę z bazą danych i umożliwia integrację z wieloma aplikacjami i narzędziami.
- **Zgodność i standardy:** Oracle DB zgodny jest z wieloma standardami i regulacjami branżowymi, co jest istotne w przypadku organizacji podlegających rygorystycznym przepisom i przepisom prawno-finansowym.

Dzięki tym zaletom oraz przez fakt, iż ta baza danych używana jest w większości systemów informatycznych Uniwersytetu Zielonogórskiego, jest ona również wykorzystywana w niniejszym projekcie.

2.4. Technologie pomocnicze

W tej sekcji poruszony zostanie temat innych mniej istotnych jednak nadal potrzebnych i przydatnych bibliotek, które zostały użyte w projekcie. Tworzenie aplikacji dzięki tym dodatkom staje się o wiele łatwiejsze oraz o wiele bardziej wygodne i intuicyjne, dzięki czemu docelowo można skupić uwagę na najważniejszych elementach systemu a mniej istotne rzeczy pozostawić do zrealizowania poprzez użycie właśnie wyżej wymienionych zewnętrznych bibliotek.

Skuteczne przeszukiwanie list wyboru w aplikacji jest kluczowe dla sprawnego i wydajnego funkcjonowania aplikacji i z tego względu należy posiadać opcję wyszukiwarki w liście, implementacja własnego rozwiązania (wraz z polami edycyjnymi oraz stylami CSS) zajęłaby naprawdę sporo czasu a implementacja tego rozwiązania w JavaScript pochłonęła by dużą ilość pracy z tego względu w projekcie została zastosowana biblioteka JavaScript o nazwie "SELECT2"[7]. Jest to prosta biblioteka JavaScript, która została wręcz stworzona na potrzeby poniższej pracy dyplomowej, gdyż umożliwia łatwą oraz sprawną implementację opcji wyszukiwarki w polach wyboru, z listy a co za tym idzie umożliwia na sprawną nawigację w zbiorach danych, które w tych listach się znajdują. Dzięki zastosowaniu tej biblioteki zaoszczędzone

zostaje sporo czasu i pracy, które to można poświęcić na optymalizację i ulepszanie pozostałych elementów aplikacji, które nie muszą zostać stworzone od podstaw ręcznie, gdyż obarczone są bardzo specyficznymi wymaganiami projektowymi i żadna dostępna na rynku technologia aktualnie nie spełnia ich. Bibliotekę tę można zainstalować na trzy sposoby:

- **Źródła CDN (Content Delivery Network)** - w nagłówku głównego pliku html w aplikacji należy w odpowiednich tagach html podać adresy do styli oraz do kodu javascript z którego korzysta select2.
- **Poprzez Bower (menedżer pakietów javascript)** - instalacja odbywa się poprzez wykonanie jednej prostej komendy 'bower install select2'.
- **Ręczna instalacja** - proces podobny do instalacji poprzez źródła CDN, jednak różni się tym, iż należy pobrać pliki na serwer (maszynę lokalną) i w tagach html podać ścieżki do w. w. plików.

Jak widać instalacja jest bardzo prosta a pola, które powinny posiadać opcję wyszukiwania, muszą mieć przypisaną klasę o nazwie 'search_select' po tym zabiegu można korzystać z wyszukiwania w polach wyboru.

Sprawne działanie aplikacji wymaga również, aby użytkownik na bieżąco był informowany o tym, co się dzieje po stronie systemu tak, aby mógł ewentualne problemy w łatwy sposób zgłosić administratorowi. Przejrzysty i szybki feedback aplikacji do użytkownika jest kluczowy również po to, aby użytkownik wiedział że aplikacja działa poprawnie a nie zawiesza się na czas, gdy dochodzi do zapisywania zmian lub do wpisania jakichś zmian, w momencie wprowadzenia danych potrzebny jest jasny komunikat, czy dane zostały poprawnie zapisane w bazie oraz czy po drodze nie doszło do innego błędu, który uniemożliwił wprowadzenia danych do bazy (przykładowo użytkownik wpisał niepoprawny separator dziesiętny). Problem czytelnych komunikatów rozwiązuje biblioteka o nazwie 'Sweetalert2'[8], jest to prosta biblioteka która posiada w sobie gotowe do użycia modale oraz tak zwane 'toasty' czyli małe (zazwyczaj pojawiające się w wybranym rogu przeglądarki) okienka z komunikatami wraz z animacją. Instalacja sweetalert2 odbywa się analogicznie do select2. Biblioteka daje spore możliwości konfiguracyjne, co z pewnością jest zaletą.

Rozdział 3

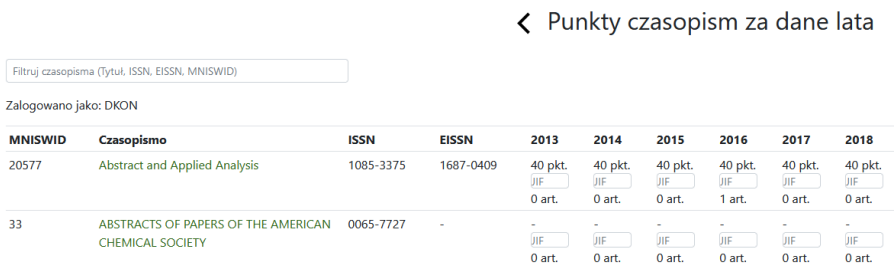
Modernizacja systemu SKEP

W niniejszym rozdziale omówiono praktyczne aspekty modernizacji systemu SKEP. Na podstawie struktury systemu (szczegółowo przedstawionej w poprzednim rozdziale), opracowano koncepcję zmodernizowanego systemu tak, aby zachować jak najbardziej zbliżoną strukturę bazy danych oraz jak najbardziej przyjazny interfejs użytkownika końcowego. Ważnym aspektem była również wydajność aplikacji oraz czas wczytywania danych, które później przedstawiane są użytkownikowi, optymalizacja będzie wymagać wiele kompromisów.

3.1. Czasopisma

Największą wadą dotychczasowego systemu był problem z synchronizacją listy czasopism z tą, która dostarczana jest co roku przez Ministerstwo Nauki i Szkolnictwa Wyższego gdyż do tej pory czasopisma były wpisywanie ręcznie do tabeli czasopism co generowało częste błędy oraz sporadyczne duplikaty czasopism znajdujących się w bazie, należało dokonać szybkiej integracji listy ministerstwa z naszą wewnętrzną listą oraz przypisanie wszystkich artykułów do odpowiadających im czasopism przy zachowaniu, nie ministerialnych czasopism, w których też są publikowane prace naukowe na Uniwersytecie Zielonogórskim. Dodatkowo należało wziąć pod uwagę fakt, iż mogą pojawiać się nowe artykuły w czasopismach, które nie są jeszcze wpisane ani do listy ministerstwa ani do naszych wewnętrznych baz, przez co należy stworzyć system który skutecznie będzie umożliwiał dopisywanie nowych czasopism do istniejącej już listy oraz zabezpieczyć je przed ewentualnym usunięciem

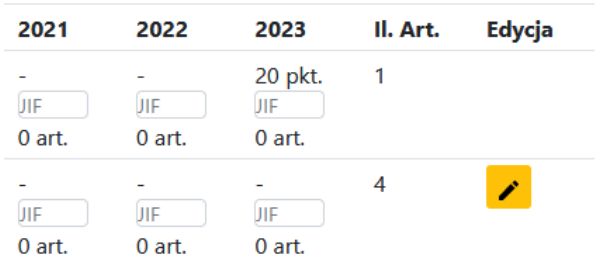
podczas corocznej synchronizacji oraz udostępnić narzędzie do modyfikowania ich danych (w przypadku błędów lub zmian czasopisma). Czasopisma posiadają również dodatkowe atrybuty, które powinny zostać uwzględnione w module wprowadzania czasopism, gdyż aktualnie dane te wpisywane są do poszczególnych artykułów w czasopiśmie a nie są przypisywane do danego czasopisma co powoduje że w bazie mogą pojawiać się duble oraz błędnie wprowadzone dane, co przełoży się na błędne opisy bibliograficzne oraz zwiększoną objętość bazy danych. Rysunek 3.1 przedstawia fragment modułu systemu do wprowadzania danych do bazy danych.



Rysunek 3.1. Wycinek głównego elementu modułu czasopism

Należy wyszczególnić dwa rodzaje pól znajdujących się na wyżej wymienionym rysunku, pole odpowiedzialne za filtrowanie czasopism (lewy górny róg) oraz pole do wprowadzania wartości liczbowej JIF (z ang. Journal Impact Factor), całość jest zamknięta w tabeli, która posiada cztery najważniejsze kolumny, które identyfikują czasopismo (Id znajdujące się na liście ministerialnej, Tytuł czasopisma, ISSN oraz EISSN) po których następują już poszczególne lata zaczynając od roku 2013 aż do aktualnego roku (na dzień dzisiejszy 2023). Kolumny z latami posiadają w sobie pola do edycji JIF dla danego roku (zapisanego u góry) oraz ilości punktów, które znajdują się na liście Ministerstwa Edukacji Narodowej, jeżeli czasopismo w danym roku nie istniało lub nie posiada wpisanych punktów we wspomnianej liście wtedy przypisywana jest kreska oznaczająca brak danych, jednak pole JIF jest aktywne i można wpisać do niego wartość za dany rok. Poszczególne lata posiadają również informację, ile w danym roku znajduje się czasopism, ma to na celu ułatwienie pracy użytkownikowi gdyż w łatwy sposób jest on w stanie określić czy powinien szukać a następnie wpisać (jeśli istnieje) wartość JIF w danym roku, w innym przypadku wymagałoby to ręcznego przeszukania listy artykułów w celu znalezienia artykułów, które znajdują się w danym roku w czasopiśmie i dopiero po znalezieniu takowego

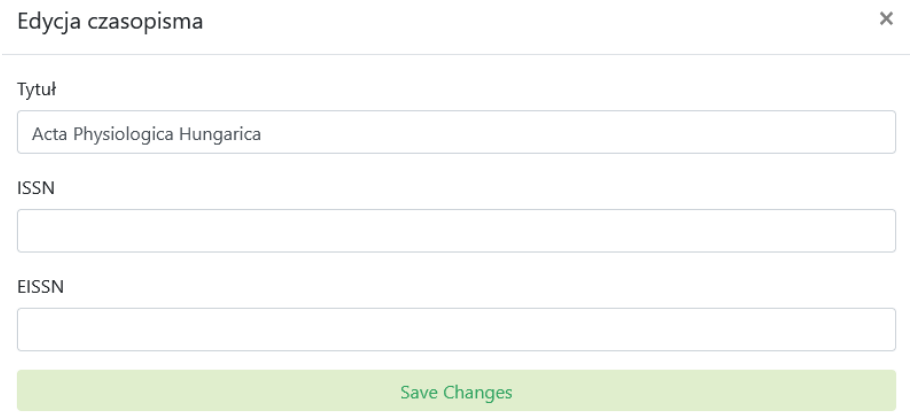
wpisać wartość, jest to bez wątpienia uciążliwe i podatne na błędny ze strony użytkownika (może nie zauważyć czasopisma lub niepoprawnie dokona filtrowania czasopism). Kliknięcie na tytuł czasopisma przenosi do wyszukiwarki SKEP, która od razu ustawiona jest na filtrowanie czasopisma, na które kliknął użytkownik, jest to o tyle wygodne, że dzięki temu użytkownik może od razu zweryfikować swoją pracę i upewnić się co do poprawności wpisanych oraz wyświetlanych danych w opisie bibliograficznym artykułów w aktualnie filtrowanym czasopiśmie.



Rysunek 3.2. Wycinek prawej strony tabeli modułu obsługi czasopism

Rysunek 3.2 przedstawia prawą stronę tabeli odpowiedzialnej za wprowadzanie danych do czasopism, jak widać znajdują się tutaj dodatkowe dwie kolumny (poza kolumnami z latami). Pierwsza z nich odpowiada za wyświetlenie ilości artykułów w wybranym czasopiśmie (wartość sumaryczna za wszystkie lata w zakresie od 2013 do 2023 w tym przypadku), druga kolumna odpowiada za wyświetlanie okna edycji czasopisma, jednak przycisk ten pokazywany jest jedynie w czasopismach, które zostały wpisane ręcznie do bazy danych. Pojawia się tutaj pierwszy istotny problem, w jaki sposób rozpoznać i co najważniejsze zabezpieczyć takowe czasopismo przed usunięciem lub nadpisaniem podczas corocznej aktualizacji listy czasopism. Zabezpieczenie takiego czasopisma jest podwójne, podczas wprowadzania ręcznego czasopisma do bazy danych czasopismo otrzymuje bardzo wysoki identyfikator, który jest połączeniem liczby 999000000 oraz kolejnych id znajdujących się w starej tabeli przechowującej, czasopisma która została zastąpiona przez tę, która obsługiwania jest przez powyższy moduł. Przykładowo czasopismo, które widać na rysunku 3.2 posiada id równe 999010651 czyli wartości 999000000 dodanej do kolejnego wolnego numeru w sekwencji obsługującej starą tabelę czasopism czyli 10651. Dzięki takiej operacji można, używając identyfikatora, rozpoznać czy czasopismo zostało wpisane przez użytkownika i odpowiednio ten fakt obsłużyć w skrypcie odpowiedzialnym za

aktualizację oraz wpisanie ponowne danych do tabeli czasopism, jednak aby system był jak najbardziej bezpieczny, został zastosowany dodatkowy atrybut, który posiada wartość numeryczną ‘1‘ lub ‘null‘, w pierwszym przypadku wartość ‘1‘ oznacza czasopismo wpisane ręcznie i w takiej sytuacji pojawia się przycisk umożliwiający edycję czasopisma, w przypadku wartości ‘null‘ przycisk nie zostanie będzie renderowany przez moduł a co za tym idzie edycja będzie niemożliwa.



Rysunek 3.3. Wycinek głównego elementu modułu czasopism

Okno edycji czasopisma jest modalem który pokazuje się na ekranie użytkownika po wciśnięciu przycisku edycji na czasopiśmie, które posiada taką opcję. Okno umożliwia edycję tylko trzech parametrów: tytułu, ISSN oraz EISSN. Rysunek 3.3 przedstawia okno dla przykładowego czasopisma, które posiada opcję edycji (zostało wprowadzone ręcznie). Zapisywanie danych odbywa się asynchronicznie dzięki zastosowaniu techniki AJAX, czyli dokonywania połączenia między użytkownikiem a serwerem bez przeładowywania strony. Rozwiązanie to umożliwia bardzo sprawne i wygodne obsługiwanie aplikacji. Przy zapisywaniu zmian w prawym górnym rogu pojawia się komunikat o powodzeniu lub niepowodzeniu wykonanej akcji (w tym przypadku zapisywania danych do bazy danych). Anulowanie operacji edycji odbywa się poprzez wciśnięcie klawisza ESC lub poprzez kliknięcie na krzyżyk w prawym górnym rogu okna edycji.

Wpisywanie danych na temat wartości JIF do bazy odbywa się również asynchronicznie, jednak w tym przypadku użytkownik nie musi akceptować zmian, a wystarczy że wpisze wartość a następnie wyjdzie z edycji wybranego pola tak aby fokus na to pole zniknął, w tym momencie moduł uznaje, że użytkownik skończył edycję wartości i dokonuje asynchroniczne zapytanie do bazy, aby ta przechowała wartość z pola. Przyjrzymy się jak to rozwiązanie wygląda w praktyce.

Listing 3.1. Kod odpowiedzialny za zapis JIF

```
function updateJIF(mniswId, year, value) {
    [...]
    $.ajax({
        type: 'PUT',
        headers: {
            'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')
        },
        url: "aktualizuj-jif?mnisw_id=" + mniswId + "&year=" + year + "&value=" + value,
        success:function(data){
            [...]
        },
        error:function(data) {
            [...]
        }
    })
    [...]
}
```

Listing 3.1 przedstawia fragment kodu który jest odpowiedzialny za wprowadzanie wartości JIF do bazy danych. Dane do serwera są przekazywane poprzez wartości podawane w adresie docelowym zdefiniowanym w pliku odpowiedzialnym za routing ruchu w aplikacji. Ważną kwestią jest przekazanie w nagłówku żądania token-u CSRF, gdyż jeżeli nie zostanie on przekazany system, uzna żądanie za nieuprawnione i wyświetli informację o wygaśnięciu formularza, co w tym przypadku spowoduje zwrócenie kodu statusu HTML 419 i wyzwoli zdarzenie błędu, które wyświetli użytkownikowi końcowemu błąd przy zapisie danych i prośbę o skontaktowanie się z administratorem, aby ten zbadał problem. Listing 3.2 z kolei przedstawia fragment kodu w pliku widoku (plik blade). Parametry pokroju ‘mnisw_id‘, ‘year‘ oraz ‘value‘ są przekazywane jako parametry funkcji nadrzędnej o nazwie ‘UpdateJIF‘.

Wywołanie funkcji odbywa się poprzez parametr ‘onchange‘ tagu ‘input‘ w HTML, gdy tylko zostanie wykryta zmiana dochodzi do wywołania funkcji wraz z przekaza-

niem jej parametrów. Jak widać poniżej identyfikator czasopisma pobierany jest ze zmiennej ‘journal‘ a całość znajduje się w pętli foreach, która dla każdego elementu przekazanej tablicy czasopism generuje definicję wywołania funkcji wraz z parametrami tego wywołania, dzięki czemu każde poszczególne czasopismo ma swoją indywidualną definicję wywołania wraz z informacją na temat roku, wartość zostaje pobrana dopiero po zmianie wartości pola i wtedy cały pakiet zmiennych przekazywany jest do funkcji ‘updateJIF‘, która to z kolei przekazuje dane do kontrolera a ten z kolei wykorzystując definicję wpisu do bazy w modelu czasopisma wpisuje dane do bazy danych. Pomyślne wykonanie operacji komunikowane jest poprzez toast w prawym górnym rogu ekranu, jeśli dojdzie do błędu użytkownik w identyczny sposób jest o nim informowany jednak ze zmienioną treścią komunikatu oraz jego kolorem.

Listing 3.2. Wywołanie funkcji ‘updateJIF‘

```
1 [...]
2 <input placeholder="JIF"
3        onchange="updateJIF({{ $journal->mnisw_id }}, {{ $year }}, this.value)"
4        [...]>
5 [...]
```

Duża liczba czasopism powoduje pewne problemy z optymalizacją aplikacji i pewne kompromisy musiały zostać dokonane w celu poprawienia wydajności wczytywania danych. Podstawowym rozwiązaniem problemu optymalizacji czasu ładowania strony jest podział na podstrony z czego każda z podstron pokazuje tylko dziesięć czasopism na raz, dzięki tej paginacji aplikacja działa o wiele szybciej. Paginacja działa również asynchronicznie.

Pojawia się jeden problem związany z podziałem czasopism, mianowicie podczas filtrowania czasopism może dojść do sytuacji, w której użytkownik znajduje się na podstronie wyższej niż wynosi numer wszystkich podstron po przefiltrowaniu zbioru danych, czyli po prostu użytkownik wyjdzie poza zakres listy, co nie spowoduje błędu, jednak na stronie nie znajdzie się żaden wpis, przykładowo - użytkownik znajduje się aktualnie na stronie z numerem pięć czyli aktualna ilość rekordów zwróconych wynosi przynajmniej 51 gdyż użytkownik widzi czasopisma na podstronie piątej, jednak gdy użytkownik dokonał filtrowania czasopism po numerze ISSN ilość zwróconych rekordów wynosi jeden, czyli na podstronie piątej nie znajdują się żadne

rekordy a jedynie na stronie pierwszej. Rozwiązaniem tego problemu jest sprawdzenie liczby stron po przefiltrowaniu i jeśli strona, na której aktualnie znajduje się użytkownik, przekracza liczbę wszystkich stron zwróconych po filtrowaniu oznacza to, że użytkownik znajduje się poza zakresem, w tym momencie system uznaje stronę z najwyższym indeksem za stronę poprawną, którą należy wyświetlić użytkownikowi i tą właśnie stronę pokazuje.

Listing 3.3. Fragment kodu sprawdzający zakres zbioru danych

```
[...]
$page = request()->get('page', 1);
$perPage = 10;

$slices = $journals->slice(($page - 1) * $perPage, $perPage);

$paginator = new LengthAwarePaginator(
    $slices,
    $journals->count(),
    $perPage,
    $page,
    ['path' => url()->current()]
);

if($request->ajax()) {
    if ($page > $paginator->lastPage()) {
        $page = $paginator->lastPage();
    }

    $slices = $journals->slice(($page - 1) * $perPage, $perPage);

    $paginator = new LengthAwarePaginator(
        $slices,
        $journals->count(),
        $perPage,
        $page,
        ['path' => url()->current()]
    );
}

$journals = $paginator->items();
[...]
```

Wszystko dzieje się w czasie rzeczywistym po stronie serwera i użytkownik nie musi podejmować żadnych dodatkowych kroków, po prostu, jeśli ilość danych jest zbyt mała, aby jakiekolwiek wpisy znalazły się na stronie, na której się aktualnie znajduje, to zostaje on przekierowany na ostatnią stronę z jakimikolwiek wpisami

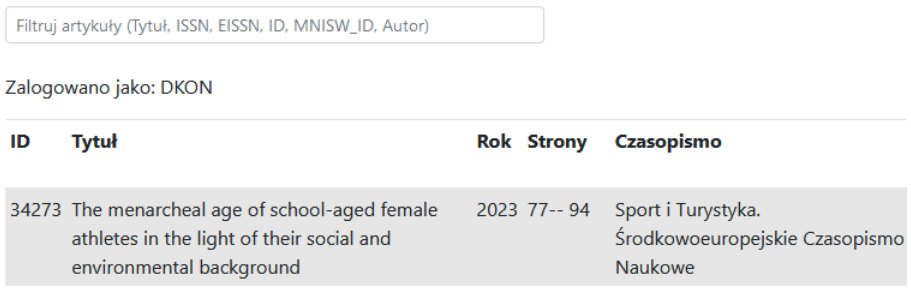
(przynajmniej jednym). Praktyczne rozwiązanie problemu zostało przedstawione na listingu 3.3 na którym widać fragment kodu odpowiedzialny za podział danych na podstrony za pomocą klasy 'LengthAwarePaginator', która to jest wbudowana w kod źródłowy frameworku Laravel, jest to idealny przykład na to, jak bardzo niektóre gotowe rozwiązania mogą pomagać w tworzeniu aplikacji webowych. Algorytm sprawdza czy żądanie jest asynchroniczne (linia 15), jeśli jest to sprawdzona zostaje strona która została wybrana przez użytkownika i jeśli przekracza ona ostatnią stronę aktualnego paginatora to wartość zmiennej '$page' zostaje przypisana do wartości ostatniej strony paginatora, następnie dokonywany jest już zwykły podział na podstrony i na końcu zostaje wybrana strona na podstawie zmiennej '$page' (linia 22 tworzy nowy obiekt paginacji i przekazuje do niego parametry). Podany fragment kodu kończy się na linii 31, gdzie znajduje się już przypisanie elementów paginatora do zmiennej '$journals', która później zostaje przekazana do widoku a tam zostaje obsłużona i wyświetlane są czasopisma. Warunek w linii 15 jest o tyle istotny, iż optymalizuje on działanie aplikacji (zakres sprawdzany jest tylko i wyłącznie wtedy gdy żądanie wykonywane jest asynchronicznie).

Dzięki tym implementacjom systemu udało się znacząco poprawić wydajność oraz przejrzystość systemu, jednak, co najważniejsze, praca osób wprowadzających dane nie wymaga już wielokrotnego wprowadzania wartości JIF dla danego artykułu, lecz wystarczy, że zostanie wartość ta wpisana do czasopisma a system sam skojarzy czasopismo z artykułem i wyświetli wartość JIF (o ile dane czasopismo taką informację posiada), pozwala to również pozbyć się niepotrzebnie zduplikowanych danych, co zmniejsza rozmiar bazy - w skrócie nowa implementacja wprowadzania czasopism jest wygodniejsza dla użytkownika, gdyż ten nie wykonuje jednej pracy wielokrotnie dla jednego artykułu, lecz wpisuje wartość tylko raz oraz zmniejsza ilość duplikatów w bazie danych. Oczywiście opisane tutaj rozwiązania nie były jedynymi potrzebnymi zmianami, jednak aby zachować odpowiedni format oraz długość pracy inżynierskiej, należało pominąć kilka drobniejszych etapów, takich jak przepisywanie identyfikatorów czasopism ze starej listy na nową czy też przeszukiwanie bazy danych za pomocą kodu PL/SQL [9] w celu znalezienia duplikatów oraz skuteczne zastąpienie ich id nowymi z listy Ministerstwa Edukacji Narodowej tak, aby zachować integralność opisów bibliograficznych a jednocześnie przejść na nową listę.

Pominięty również został skrypt odpowiedzialny za pobieranie oraz wprowadzanie danych do bazy danych z API Ministerstwa Edukacji Narodowej czyli informacje, które później są przedstawiane właśnie w wyżej wspomnianej tabeli czasopism wraz z możliwością wpisywania JIF, jednak nie wpływają one bezpośrednio na działanie wyżej opisywanego systemu.

3.2. Artykuły w czasopismach

Wszystkie powyższe implementacje zostały stworzone z myślą o zupełnie nowym module wprowadzania artykułów w czasopismach, bazuje on na poprzedniej wersji stworzonej w Oracle Forms, jednak duża jego część została zmodyfikowana a pozostał jedynie sam zamysł oraz ułożenie elementów interfejsu aplikacji tak, aby użytkownik końcowy mógł, bez problemu oraz bez wymogu nauki nowego systemu, zacząć korzystać z aplikacji. Na rysunku 3.4 widać część tabeli wyświetlającej artykuły w czasopismach.



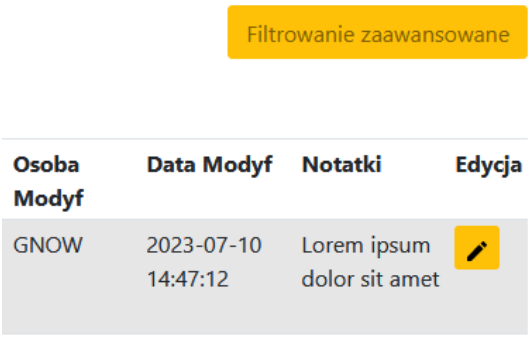
Rysunek 3.4. Wycinek głównego elementu wprowadzania AWCZ

Fragment, który widać na rysunku 3.4, przedstawia podstawowe informacje na temat publikacji AWCZ takie jak: "Tytuł", "Rok", "Strony", "Czasopismo"oraz "ID", są to informacje, po których można w łatwy sposób zidentyfikować artykuł. Powyżej tabeli widzimy pole edycyjne, które odpowiada za wprowadzania fraz filtrowania, jest to jedna z dwóch opcji przeszukiwania zbioru danych i to pole odpowiada za podstawowe wyszukiwanie - szuka ciągu znaków z pola edycyjnego w każdym z wierszy tabeli (tylko widoczne kolumny oraz ISSN i EISSN biorą udział w filtrowaniu). Filtrowanie zaawansowane zostanie przedstawione w dalszej części rozdziału po przedstawieniu całej tabeli wyświetlającej artykuły w czasopismach. Na rysunku 3.5 przedstawiona została środkowa część tabeli wyświetlającej AWCZ, można tu-

taj znaleźć informacje o autorach z uczelni jak i autorach obcych pochodzących z innych uczelni, ale jednocześnie biorących udział w pracy nad publikacją. Autorzy wyświetlani są po przecinkach w dwóch osobnych kolumnach - "Autor Swoj", czyli kolumna wyświetlająca autorów z Uniwersytetu Zielonogórskiego oraz w kolumnie "Autor Obcy"wyświetlającej wszystkich autorów spoza uczelni po przecinku. W kolumnach została zachowana kolejność autorstwa, która ustalana jest na podstawie zmiennej w bazie danych.

Rysunek 3.5. Środkowy wycinek głównego elementu wprowadzania AWCZ

Na wcześniej wspomnianym rysunku znajdują się jeszcze trzy dodatkowe kolumny, "Kod", czyli kolumna odpowiedzialna za wyświetlanie kodu ministerialnego, który dana publikacja ma przypisany, "Osoba Wprow", czyli identyfikator osoby, która wprowadziła do bazy danych wpis, który aktualnie jest oglądany przez użytkownika, "Data Wprow", czyli data wprowadzenia publikacji do bazy danych.



Rysunek 3.6. Końcowy wycinek głównego elementu wprowadzania AWCZ

Na rysunku 3.6 widać ostatni fragment tabeli AWCZ wraz z przyciskiem odpowiedzialnym za zaawansowane filtrowanie artykułów w czasopismach. Fragment ten posiada informacje o osobie modyfikującej wpis wraz z datą tej modyfikacji oraz dodatkowo przedstawiane są notatki zapisane wraz z artykułem. Ostatnia kolumna po prawej stronie posiada przycisk umożliwiający edycję danej publikacji.

Filtrowanie zaawansowane umożliwia przeszukanie bazy na podstawie aż dwunastu pól, do których należą: **tytuł**, **rok**, **strony**, **czasopismo**, **ISSN**, **EISSN**, **kod**,

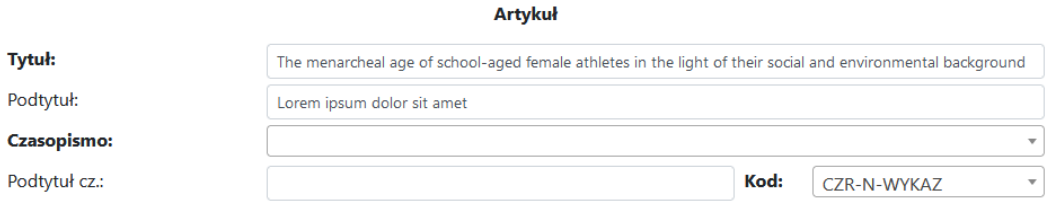
autor z uczelni, **autor spoza uczelni**, **osoba wprowadzająca**, **osoba modyfikująca**, **notatki**. W przypadku, w którym zapełnione są dwa pola lub więcej podczas filtrowania danych brane pod uwagę są wszystkie pola posiadające wartość.

Rysunek 3.7. Fragment formularza do filtrowania zaawansowanego

Po wprowadzeniu przez użytkownika danych na podstawie, których ma dojść do filtrowania, użytkownik wciska przycisk "Filtruj"w celu wykonania operacji, strona nie zostaje przeładowana gdyż filtrowanie odbywa się asynchronicznie tak, aby ułatwić użytkownikowi pracę z systemem oraz jednocześnie zadbać o wydajność aplikacji. Na rysunku 3.7 przedstawiono fragment formularza odpowiedzialnego za zaawansowane filtrowanie publikacji. Formularz 'wysuwa' się spod przycisku 'Filtrowanie zaawansowane' po wciśnięciu niniejszego przycisku.

Edycja wybranego artykułu odbywa się poprzez specjalnie przygotowany do tego celu 'modal' pojawiający się po wciśnięciu przycisku edycji znajdującego się po prawej stronie tabeli przy każdej publikacji. Wciśnięcie przycisku powoduje pokazanie się okna modal na środku ekranu o wielkości zbliżonej do całego okna przeglądarki (90 vw - z ang. viewport width - czyli 90 procent szerokości widoku). Panel edycji został podzielony na fragmenty, podobnie jak ma to miejsce w oryginalnej aplikacji stworzonej w Oracle Forms. Na rysunku 3.8 przedstwiony został wycinek fragmentu "Artykuł"odpowiedzialnego za wprowadzanie podstawowych informacji na temat aktualnie edytowanego wpisu. Są to najważniejsze pola i określają one tytuł oraz podtytuł AWCZ oraz czasopismo wraz z jego podtytułem, w którym artykuł został opublikowany. W poniższej części można również wybrać kod ministerialny dla aktualnie edytowanego wpisu.

Kolejnym istotnym elementem całego systemu wprowadzania oraz edycji artykułów w czasopismach jest zakładka "Słowa kluczowe", która to odpowiada za wprowadzanie słów kluczowych dla danej publikacji. Przy implementacji tej funkcjonalności pojawiło się kilka problemów, dla których rozwiązania zostaną przedstawione



Rysunek 3.8. Fragment formularza do edycji artykułu

poniżej. Rysunek 3.9 przedstawia zakładkę odpowiedzialną za wprowadzanie słów kluczowych.

Rysunek 3.9. Element formularza odpowiedzialny za wprowadzanie słów kluczowych

Można rozróżnić tutaj trzy podstawowe elementy:

- **pole dodaj** - odpowiedzialne za wpisywanie wartości słowa kluczowego tak, aby wartość ta została później przeniesiona do listy skąd trafi do bazy danych,
- **przycisk '+'** - odpowiedzialny za zatwierdzenie podanej wartości słowa kluczowego oraz przeniesienie go do listy gdzie później podczas zapisywania trafi do bazy danych,
- **lista ze słowami kluczowymi** - czyli miejsce, w którym wyświetlane są wszystkie wpisane do listy aktualnego AWCZ słowa kluczowe wraz z przyciskami odpowiedzialnymi za ich usuwanie.

Procedura wprowadzania słowa kluczowego wygląda następująco:

- **użytkownik wprowadza wartość tekstową słowa kluczowego w pole edycyjne**,
- **po wprowadzeniu wartości należy wcisnąć przycisk '+' w celu wprowadzenia słowa kluczowego do listy**,

- **gdy powstaje potrzeba usunięcia słowa kluczowego, należy wcisnąć czerwony ‘X‘.**

Listing 3.4 przedstawia fragmenty kodu odpowiedzialne za implementację rozwiązania wprowadzania słów kluczowych do listy pokazanej na rysunku 3.9. Słowa kluczowe, które zostały pobrane z bazy danych i przekazywane są w formie listy do funkcji, która uruchamia funkcjonalność po stronie użytkownika oraz obsługuje zdarzenia usuwania wraz z dodawaniem słów kluczowych. Algorytm sprawdza słowa kluczowe czy nie zostały powtórzone (uniemożliwia użytkownikowi wpisania drugi raz tego samego słowa kluczowego), sprawdzanie odbywa się poprzez wyliczenie wartości funkcji skrótu dla danego słowa kluczowego (tekstu) a następnie ustawienia tej wartości jako identyfikatora w liście słów kluczowych dla aktualnej publikacji. Funkcja skrótu, która została tutaj wykorzystana to MD5, która to w tym przypadku będzie jak najbardziej wystarczająca (szanse na kolizję tej funkcji skrótu są małe) gdyż nie jest od niej istotne bezpieczeństwo aplikacji (MD5 nie nadaje się do szyfrowania haseł ze względu na łatwość celowego wywołania kolizji). W momencie wykrycia przez algorytm powtórzenia słowa kluczowego pole do wpisywania wartości zmienia kolor na czerwony, poniżej pola wyświetla się stosowny komunikat o tym, że słowo kluczowe już istnieje w tym zbiorze oraz samo słowo nie zostaje dodane drugi raz (za to zachowanie odpowiadają linie 19 oraz 20). Jeśli słowo kluczowe nie istnieje w liście, to zostaje ono dodane i wyświetlone w liście. Funkcja sprawdzająca powtórzenia słów kluczowych zwraca uwagę na wielkość znaków (jest ‘case sensitive‘).

Usuwanie słów kluczowych z listy odbywa się po wciśnięciu przycisku ‘X‘, każde słowo kluczowe i odpowiadający mu przycisk posiada przypisaną swoją funkcję anonimową. Przypisanie dokonywane jest za pomocą funkcji ‘keywordReload‘.

Listing 3.4. Fragment kodu odpowiedzialny za słowa kluczowe

```
1 [...]
2 keywords: function (kwrds) {
3     [...]
4     if($('.keyword-form').length) {
5         [...]
6         for (var i = 0; i < kwrds.length; i++) {
7             spark.append(kwrds[i]['slowo']);
8             md5Hash = spark.end();
9             keywrds.push({id: md5Hash, name: kwrds[i]['slowo']});
```

```
10          }
11          [...]
12          this.addEvent(keywordForm, 'submit', addKeyword, false);
13          function addKeyword()
14          {
15              event.preventDefault(); //disable default event
16              if (keywordInput.value.trim() === '') return; //check if value is provided
17              spark.append(keywordInput.value.trim());
18              md5Hash = spark.end();
19              const existingKeyword = keywrds.find(kwrd => kwrd.id === md5Hash);
20              if(existingKeyword) {
21                  [...]
22              } else {
23                  keywrds.push({id: md5Hash, name: keywordInput.value.trim()});
24                  [...]
25              }
26          }
27          [...]
28          function keywordsReload() {
29              var close = keywords.querySelectorAll(".delete-button");
30              for (var i = 0; i < close.length; i++) {
31                  close[i].onclick = function () {
32                      var div = this.parentElement;
33                      var id = div.attributes["data-key"].value;
34                      kwrds_deleted.push(keywrds.find((kw) => kw.id === id));
35                      keywrds = removeObjectWithId(keywrds, id.toString());
36                      div.remove();
37                  }
38              }
39          }
40      }
41  [...]
```

Funkcja dla wszystkich słów kluczowych znajdujących się w liście ‘keywords‘ dodaje event ‘onclick‘ (wywoływany po wciśnięciu przycisku), który to podczas wywołania uruchamia wcześniej wspomnianą funkcję anonimową (linie 31 do 37). Tymczasowy identyfikator słowa kluczowego w liście znajduje się w atrybucie html ‘data-key‘, który w linii 33 zostaje pobrany dla aktualnego przycisku, w momencie gdy dane słowo kluczowe zostaje usunięte na podstawie tego identyfikatora zostaje rozpoznane i przeniesione z listy słów kluczowych do listy usuniętych słów kluczowych. Taka implementacja wynika z potrzeby usuwania słów kluczowych również w bazie danych, a gdyby nie lista z informacją o usuniętych elementach, to w efekcie nie istniałaby możliwość weryfikacji w bazie danych, które elementy istnieją i powinny zostać a które nie. W momencie zapisywania zmian w bazie algorytm sprawdza li-

stę usuniętych słów kluczowych a następnie wykonuje zapytania ‘SELECT‘ w celu sprawdzenia czy dane słowo kluczowe istnieje czy nie, jeśli istnieje zostaje usunięte za pomocą ‘DELETE‘ z bazy, ma to jedną ogromną zaletę, wykonywane są tylko konieczne usunięcia z bazy danych i nie jest dokonywane masowe usuwanie słów kluczowych a następnie ponowne ich wprowadzanie do bazy danych (duplikowanie co najmniej jednej transakcji za każdym razem), takie operacje spowodowałyby niepotrzebny przyrost kopii bazy używanej do ewentualnego przywracania poprzednich wersji bazy, gdyż każda transakcja na bazie danych zostaje zapisana, aby później w razie potrzeby mogła zostać cofnięta (nie tyczy się to jedynie zapytania ‘SELECT‘, które po prostu nie modyfikuje żadnych danych a jedynie je zwraca).

Największym problemem, który pojawił się podczas implementacji algorytmu obsługującego słowa kluczowe, to niejako przechodzenie słów kluczowych między czasopismami. Przykład:

- Użytkownik przechodzi do pierwszej publikacji AWCZ na liście która posiada słowa kluczowe: ‘**Veni**‘, ‘**Vidi**‘.
- Dopisane zostaje jedno słowo kluczowe ‘**Vici**‘.
- Zmiany w edycji czasopisma zostają zapisane.
- Użytkownik przechodzi do edycji drugiego AWCZ na liście, które posiada słowa kluczowe ‘**Non**‘, ‘**omnis**‘ i które poprawnie zostają wyświetlone na początku.
- Wprowadzone zostaje nowe słowo kluczowe ‘**moriar**‘ jednak podczas zapisywania nagle pojawiają się słowa kluczowe z poprzedniego AWCZ czyli: ‘**Veni**‘, ‘**Vidi**‘ oraz ‘**Vici**‘ a wprowadzone przez użytkownika słowo kluczowe znika a po zapisaniu zmian dla AWCZ słowa kluczowe z czasopisma pierwszego przechodzą do czasopisma drugiego, co powoduje błąd poprawności danych w bazie danych (w AWCZ drugim znajdują się słowa kluczowe które nie należą do niego).

Na czym polega problem? W dokumentacji javascript [10] można znaleźć informacje na temat metody ‘addEventListener‘ [11], metoda ta ustawia funkcję na celu zdarzenia (EventTarget), która zostanie wywołana w momencie wykonania podanego rodzaju zdarzenia na wybranym celu. Celem w tym przypadku jest formularz

dodawania słowa kluczowego (‘keywordForm‘) a docelowym zdarzeniem jest ‘submit‘, czyli moment wciśnięcia przycisku dodawania przez użytkownika. Każdy obiekt DOM (z ang. obiektowy model dokumentu) posiada listę zdarzeń, które są do niego przypisane a funkcja ‘addEventListener‘ po prostu dopisuje (jeśli dane zdarzenie nie występuje na liście) do listy zdarzeń podane zdarzenie. Do metody można przekazać funkcję, funkcja może być anonimowa lub może być po prostu zwykłą funkcją (zadeklarowaną wcześniej). Przekazanie tradycyjnego podprogramu nie stanowi problemu, gdyż zostaje przekazany identyfikator podprogramu, jednak podczas podania w ‘addEventListener‘ literału funkcyjnego wcześniej wspomniana metoda zacznie tworzyć za każdym razem zupełnie nowe odniesienia w liście zdarzeń do, z pozoru identycznych funkcji, kolejnych podprogramów, co w powyższym rozwiązaniu oznacza, że przy drugim otwarciu okna edycji powstają już dwie, identyczne funkcje (jednak z innymi wartościami zmiennych), które później są wykonywane przy każdym otrzymaniu zdarzenia ‘submit‘. Ilość tworzonych listenerów jest wprost proporcjonalna do ilości otwartych przez użytkownika okien edycji AWCZ co w skrócie oznacza, że na każde otwarcie okna przypada, kolejna, nowa funkcja (logicznie identyczna jednak z innymi wartościami zmiennych). Dochodzi więc do multipleksowania funkcji obsługi zdarzeń. Najprostszym i najbardziej oczywistym rozwiązaniem jest użycie metody ‘removeEventListener‘ [12] w momencie zamykania okna, jednak z powodu iż przekazywana zostaje funkcja anonimowa użycie wcześniej wspomnianej metody nic nie da, ponieważ wymagane jest podanie referencji do funkcji (jej identyfikatora) a z racji specyfiki algorytmu nie jest możliwe stworzenie globalej funkcji która zajmie się obsługą słów kluczowych w liście (podprogram ‘addKeyword‘ zdefiniowany jest lokalnie w funkcji ‘keywords‘), przez co jego referencja (identyfikator) nie znajdują się w miejscu obsługi zamykania okna edycji AWCZ. Element, który widoczny jest globalnie to sam formularz słów kluczowych, a więc należy zaimplementować funkcję, która na podstawie odniesienia się do formularza będzie w stanie dodać zdarzenie, oraz funkcję, która będzie umożliwiała całkowite usunięcie wszystkich zdarzeń (odpowiedniego typu) danego elementu.

Listing 3.5. Funkcja przypisująca zdarzenia

```
var _eventHandlers = {};
[...]
addEvent: function(node, event, handler, capture) {
    if (!(node in _eventHandlers)) {
        // _eventHandlers stores references to nodes
        _eventHandlers[node] = {};
    }
    if (!(event in _eventHandlers[node])) {
        // each entry contains another entry for each event type
        _eventHandlers[node][event] = [];
    }
    // capture reference
    _eventHandlers[node][event].push([handler, capture]);
    node.addEventListener(event, handler, capture);
},
[...]
```

Na listingu 3.5 przedstawiona została funkcja dodająca nowe zdarzenie do listy zdarzeń podanego w parametrze funkcji elementu. Podprogram przyjmuje cztery parametry (‘node‘, ‘event‘, ‘handler‘ oraz ‘capture‘) gdzie ‘node‘ to element, do którego zostanie przypisany event, ‘event‘ to rodzaj zdarzenia, którego program ma nasłuchiwać (np. ‘submit‘, ‘onclick‘, ‘mouseover‘ etc.), ‘handler‘ to funkcja, która ma zostać wywołana gdy dane zdarzenie zostanie wykryte, ‘capture‘, czyli czy dane zdarzenie ma zostać wysłane ‘w dół‘ drzewa DOM czy w górę drzewa (‘Event Bubbling‘ lub ‘Event Capturing‘) [13]. Obie funkcje korzystają z globalnej zmiennej ‘_eventHandlers‘, która to przechowuje informacje o wszystkich zdarzeniach zarejestrowanych poprzez tę funkcję w celu późniejszej możliwości usunięcia tych zdarzeń z wybranego elementu DOM. Na samym końcu tradycyjną metodą zostaje zapisane zdarzenie.

W przypadku funkcji ‘removeAllEvents‘ (listing 3.6) sprawa jest prostsza, gdyż funkcja ta ma za zadanie jedynie usunąć wszystkie zdarzenia utworzone na wybranym elemencie, informacje nt. zdarzeń pobiera również ze zmiennej ‘_eventHandlers‘ po to, aby móc skutecznie wywołać metodę ‘removeEventListener‘. Podprogram przyjmuje jedynie dwa parametry, ‘node‘ czyli element, z którego mają zostać usunięte wszystkie zdarzenia oraz ‘event‘ czyli rodzaj zdarzeń, które mają zostać usunięte, następnie funkcja iteruje poprzez listę ‘_eventHandlers‘ i w momencie zgodności rodzaju zdarzenia usuwa go z listy zdarzeń za pomocą metody ‘removeEventListener‘.

Listing 3.6. Funkcja usuwająca wszystkie zdarzenia

```
[...]
removeAllEvents: function(node, event) {
    if (node in _eventHandlers) {
        var handlers = _eventHandlers[node];
        if (event in handlers) {
            var eventHandlers = handlers[event];
            for (var i = eventHandlers.length; i--;) {
                var handler = eventHandlers[i];
                node.removeEventListener(event, handler[0], handler[1]);
            }
        }
    }
},
[...]
```

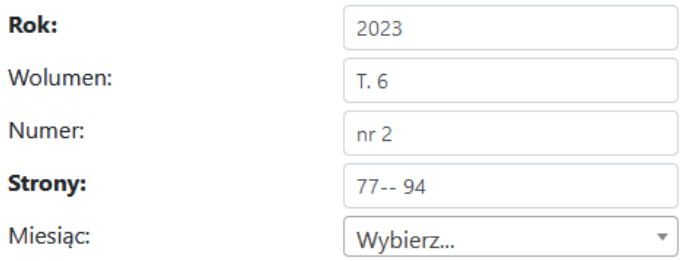
Takie rozwiązanie umożliwia usuwanie wszystkich zdarzeń wybranego elementu DOM nawet, gdy taka konieczność zajdzie poza aktualnym zakresem programu, czyli w momencie zamykania okna edycji istnieje możliwość usunięcia wszystkich zdarzeń z listy dla formy służącej do wprowadzania słów kluczowych, w ten sposób unika się multipleksowania funkcji obsługi zdarzeń. Istniało też inne rozwiązanie, a mianowicie istnieje opcja usuwania samego elementu drzewa DOM, powoduje to usunięcie wszystkich zdarzeń, które zostały przypisane do tego elementu DOM, jednak jest to rozwiązanie nieeleganckie i wymaga tworzenia elementów DOM za pomocą kodu javascript, co nie jest zalecane w przypadku pisania czystego oraz wydajnego kodu. Funkcja usuwania zaś może zostać wywołana z poziomu innego pliku javascript, co umożliwia na dowolne tworzenie oraz usuwanie zdarzeń na elementach.

Listing 3.7 przedstawia fragment pliku ‘awcz_edit_script.blade.php‘, w którym to znajduje się metoda obsługująca zdarzenie zamknięcia okna edycji. Widać tutaj, że funkcja usuwania zdarzeń została również użyta na formularzu wprowadzania autorów, który działa bardzo podobnie z jedną różnicą, autorzy są wybierani z listy autorów a nastepnie dane o nich są wpisywane do listy podobnej jak lista słów kluczowych. Do obu wywołań funkcji przekzywany jest element drzewa DOM, którego akcja ma dotyczyć (‘kwForm‘ oraz ‘auForm‘) oraz rodzaj zdarzenia, które ma zostać usunięte (w tym przypadku ‘submit‘). W ten sposób problem multipleksowania funkcji obsługi zdarzeń został rozwiązany.

Listing 3.7. Kod obsługujący zdarzenie zamknięcia okna edycji

```
[...]
$('#editAWCZModal').on('hidden.bs.modal', function () {
    let kwForm = document.querySelector('.keyword-form');
    let auForm = document.querySelector('.authors-form');
    Pers.removeAllEvents(kwForm, 'submit');
    Pers.removeAllEvents(auForm, 'submit');
    $('#czasopismo').empty();
})
[...]
```

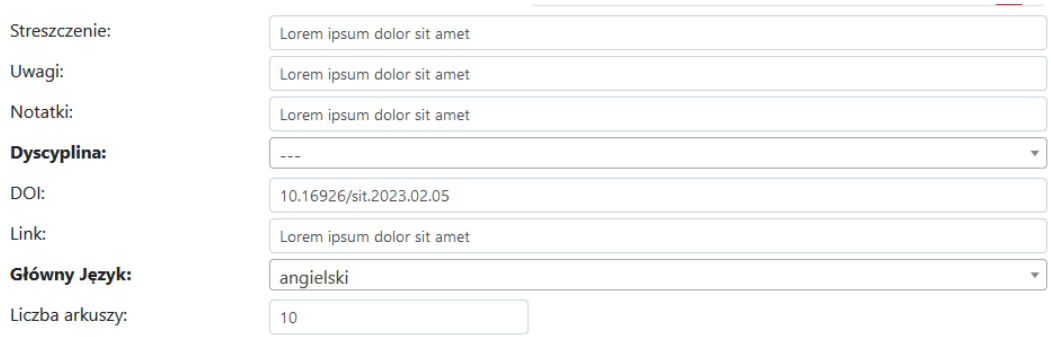
Rysunek 3.10 przedstawia fragment umożliwiający użytkownikowi wprowadzenie danych na temat roku, wolumenu, numeru oraz stron, na których artykuł się znajduje czy też miesiąca. Wszystkie pola są polami tekstowymi poza polem 'Miesiąc', które jest listą miesięcy z której użytkownik może wybrać interesujący go miesiąc.



Rysunek 3.10. Element formularza

Poniżej słów kluczowych oraz roku, wolumenu, numeru, stron oraz miesiąca znajdują się pola odpowiadające za wprowadzanie uwag, notatek oraz streszczenia danego artykułu, gdzie zaraz po nich pojawiają się pola odpowiedzialne za wybór dyscypliny do której należy AWCZ, pole DOI umożliwiające wpisanie numeru identyfikacyjnego aktualnego artykułu, pole link, które daje możliwość wprowadzenia linku do przykładowo strony webowej czasopisma, pole wielokrotnego wyboru określające, jaki język jest językiem wiodącym w wybranym artykule oraz liczba arkuszy danej publikacji. Całość widoczna jest na rysunku 3.11.

Zakładka 'Opis fizyczny' umożliwia użytkownikowi edycję parametrów opisu fizycznego za pomocą checkboxów takich jak np: rysunki, mapy, plany, tabele, fotografie itp. Zakładka wraz z zakładką 'Cechy publikacji' widoczna jest na rysunku 3.12. Cechy publikacji, które widoczne są obok opisu fizycznego umożliwiają użytkownikowi edytowanie danych dotyczących tego czy dana publikacja jest na przykład recenzją, artykułem przeglądowym, polemika, notatką, listem, artykułem popular-

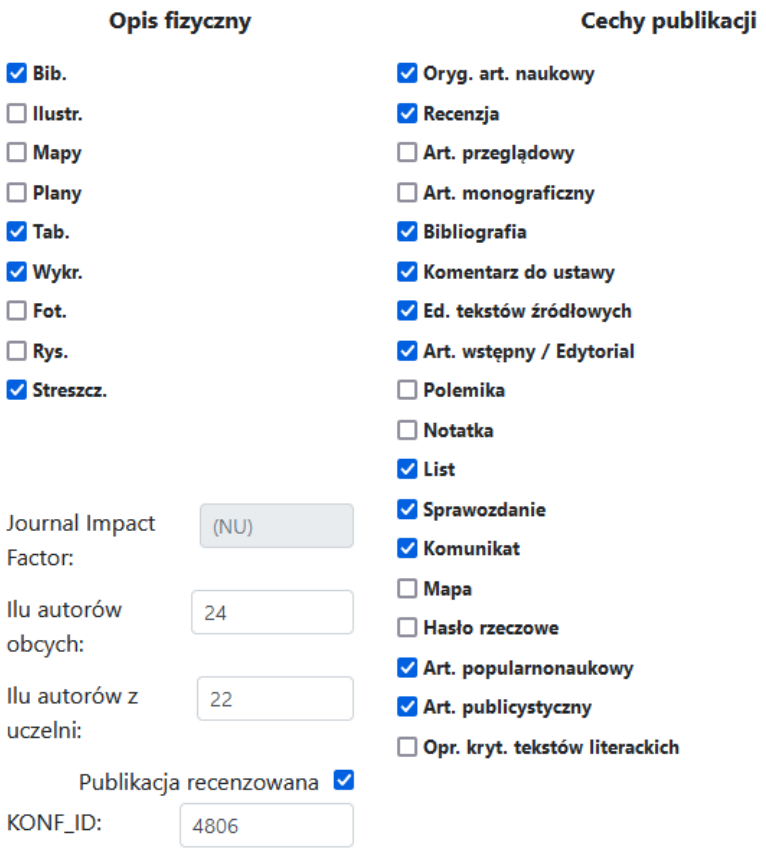


Rysunek 3.11. Element formularza

nonaukowym, artykułem publicysztycznym lub innym rodzajem artykułu w czasopiśmie.

W lewym dolnym rogu rysunku 3.12 widoczne są dodatkowe pola. Pole ‘Journal Impact Factor‘, ze względu na zmienioną strukturę wyświetlania oraz przekazywania do modułu wyświetlającego opisy bibliograficzne, nie jest już używane, jednak w celu zachowania estetycznego wyglądu aplikacji, a co najważniejsze jak najbardziej zbliżonego do oryginalnej formy ułożenia elementów, zostało zachowane i jedynie wyłączone (możliwość edycji wartości została zablokowana). Pola ‘Ilu Autorów obcych‘ oraz ‘Ilu autorów z uczelni‘ w starszych publikacjach nadal są używane, z tego względu należało je zachować, w nowszych publikacjach pola są zbędne, gdyż system sam oblicza liczbę autorów. Checkbox o nazwie ‘Publikacja recenzowana‘ pozwala oznaczyć publikację jako ‘w trakcie recenzji‘, co pozwala na jej ukrycie do czasu zakończenia recenzji. Pole ‘KONF_ID‘ umożliwia wpisanie identyfikatora konferencji, której artykuł dotyczy, w tym celu użytkownik musi znać ID odpowiedniej konferencji a następnie wpisać tę wartość do poniższego pola.

Rysunek 3.13 przedstawia pola odpowiedzialne za wprowadzanie danych na temat otwartego dostępu do aktualnie edytowanej publikacji. Pierwsze pole ‘OA Licence‘ umożliwia wybór rodzaju otwartej licencji, na podstawie której publikacja została opublikowana. Pole ‘OA pub type‘ określa rodzaj publikacji, która objęta jest otwartym dostępem. ‘OA text version‘ odpowiada za wybór wersji tekstu. Pole ‘OA release date‘ oznacza datę, kiedy licencja otwartego dostępu została wypuszczona. Wszystkie wymienione pola są polami wielokrotnego wyboru, których dane (opcje wyboru) pobieranie są z bazy danych. Ostatnie pole ‘OA date‘ oznacza datę publikacji licencji otwartego dostepu i jest jedynym polem, które nie posiada wielo-

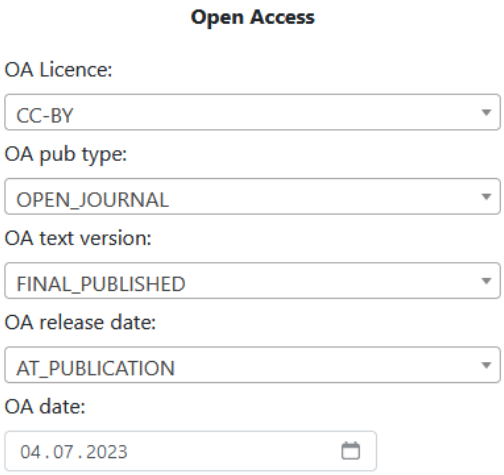


Rysunek 3.12. Element formularza - cechy oraz opis fizyczny publikacji

krotnego wyboru a za to jest polem typu ‘date‘.

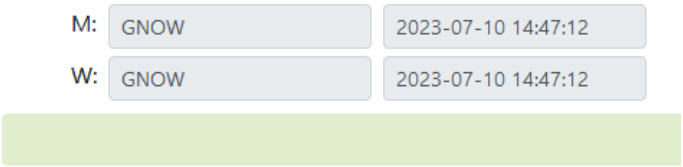
W lewym dolnym rogu okna edycji publikacji AWCZ zaraz nad przyciskiem zapisującym zmiany znajdują się cztery pola z wyłączoną edycją (rysunek 3.14) znajdują się informacje dotyczące osób, które modyfikowały lub edytowały aktualnie edytowany wpis raz z datą i godziną edycji.

Ostatnim elementem znajdującym się w formularzu edycji artykułu w czasopiśmie jest zakładka ‘Autorzy‘ odpowiedzialna za wprowadzanie, usuwanie oraz modyfikację danych na temat autorów oraz ich kolejność bądź ewentualnych afiliacji obcych lub zmiany nazwiska autora (rysunek 3.15). Można rozróżnić dwa większe elementy - formularz do wprowadzania autorów oraz listę wpisanych już autorów (wczytywaną na początku edycji z bazy danych). Formularz do wprowadzania autorów umożliwia jedynie wprowadzenie kolejności autora oraz zaznaczenia afiliacji obcej lub zmienionego nazwiska (odpowiednio pole ‘ao‘ i ‘n2‘ - pola te mogą przyjąć wartość true lub false). Na liście autorów z kolei znajdują się informacje o występujących już autorach publikacji (Imie, Nazwisko, Uczelnia z której pochodzą, Kraj z



Rysunek 3.13. Element formularza - ‘Open Access‘

którego pochodzą oraz kolejność w publikacji wraz z polami ‘ao‘ oraz ‘n2‘). Po prawej stronie każdego wpisu pojedynczego autora znajdują się przyciski umożliwiające usunięcie (czerwony ‘X‘) autora bądź edycję jego danych (jedynie pola ‘ao‘, ‘n2‘ oraz ‘Kol.‘ mogą zostac zmodyfikowane).



Rysunek 3.14. Element formularza - osoby modyfikujące

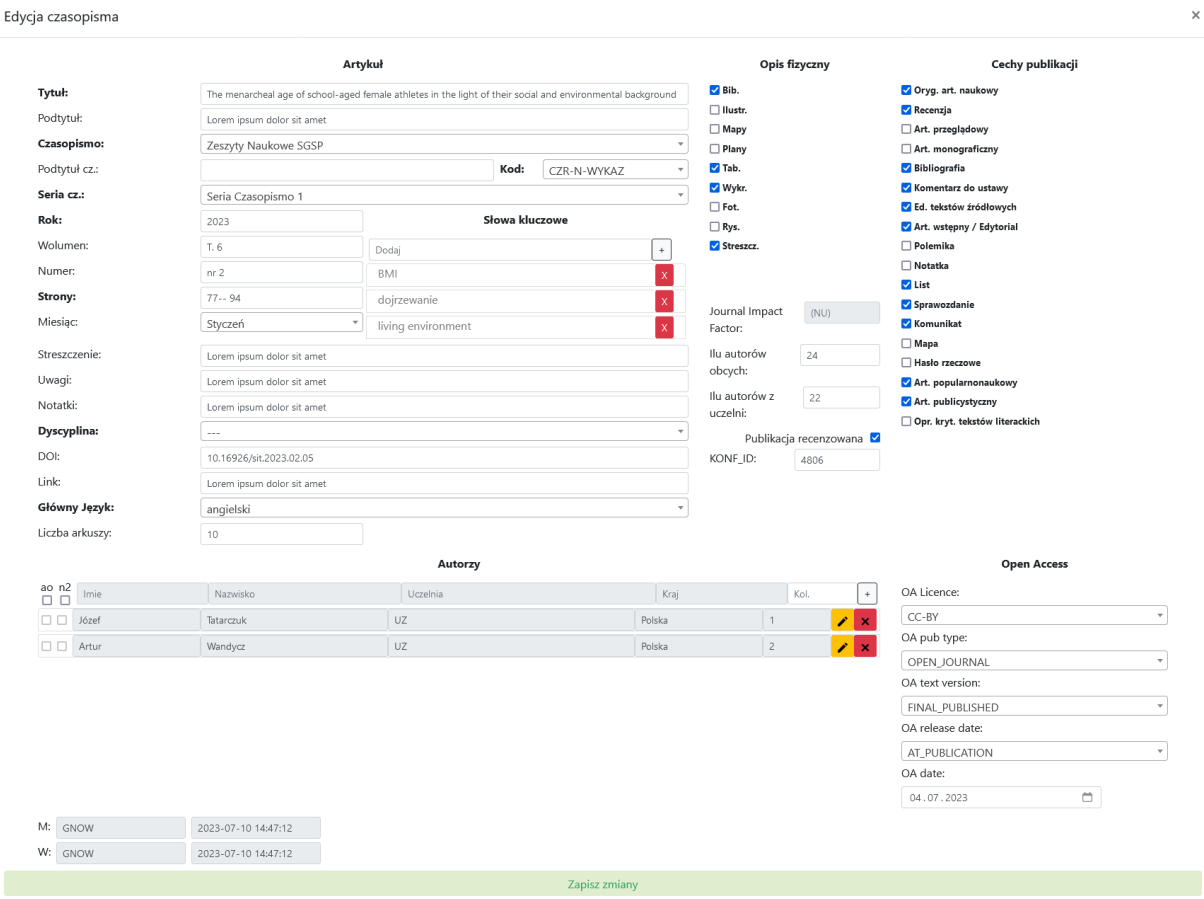
Podczas edycji pojedynczego autora przycisk edycji znika a w jego miejsce pojawia się przycisk akceptcji zmian, skrypt pilnuje aby pola ‘Kol.‘ u każdego autora różniły się od siebie (dwóch autorów nie może pojawić się na tej samej pozycji w opisie bibliograficznej stąd zasadność tego zabezpieczenia). Po dokonanej edycji użytkownik wciska przycisk zapisu w celu zapisania zmian kolejności a system dokonuje modyfikacji w liście po weryfikacji poprawności danych. System poza poprawnością danych weryfikuje kolejność autorów oraz automatycznie ją dobiera tak aby przy dodawaniu autorów użytkownik nie musiał za każdym razem wpisywać kolejności autora. Użytkownik ma możliwość ręcznej edycji kolejności.

Rysunek 3.15. Element formularza - osoby modyfikujące

Cały formularz widoczny jest na rysunku 3.16. Użytkownik, który chce zapisać zmiany, wciska zielony przycisk ‘Zapisz zmiany‘, po wciśnięciu którego dane z formularza są dzielone na trzy części:

- **Dane publikacji** - są to takie pola jak ‘Tytuł‘, ‘Czasopismo‘, ‘Rok‘, czyli główne i najważniejsze pola edycyjne całego formularza, w skład danych wchodzi zakładka ‘Artykuł‘, ‘Opis Fizyczny‘, ‘Cechy publikacji‘ oraz ‘Open Access‘.
- **Autorzy** - wszyscy autorzy wpisani w zakładce ‘Autorzy‘ zostają przekazani do funkcji javascript która asynchronicznie dokonuje analizy poprawności oraz, jeśli dane są poprawne, wpisuje je do bazy danych (odpowiednio szuka oraz usuwa lub dodaje autorów jeśli to konieczne oraz zmienia dane na temat kolejności jeli takie zmiany nastąpiły).
- **Słowa kluczowe** - dane z tej zakładki to same słowa kluczowe, lista ze słowami kluczowymi usuniętymi oraz ze słowami kluczowymi istniejącymi przekazana zostaje do funkcji która najpierw sprawdza, które słowa kluczowe należy usunąć, następnie je usuwa i na końcu dodaje wszystkie słowa kluczowe które nie występowały w bazie danych.

W ten sposób formularz zostaje zapisany, dane zostają wprowadzone do bazy danych, a użytkownik jest informowany odpowiednim komunikatem o sukcesie zapisania danych bądź też ewentualnym błędzie jeśli takowy wystąpi. Powyższy formularz jest elastyczny i uniwersalny, co oznacza, że można go przystosować do innego rodzaju publikacji np. wydawnictwa zwartego, konferencji, redakcji itp. więc powyższe rozwiązanie ma możliwość łatwej propagacji na pozostałe elementy systemu SKEP w celu jego szybkiej i kompleksowej modernizacji, należy jedynie zachować odpowiednią strukturę bazy danych, zgodną z poprzednim systemem.



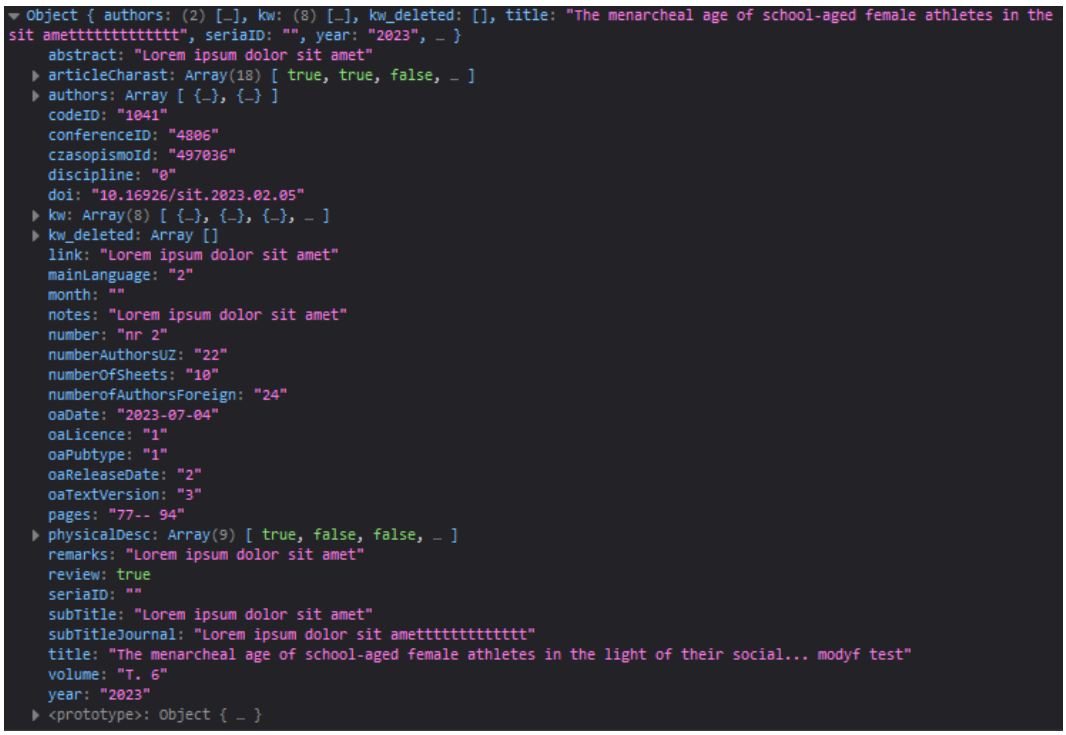
Rysunek 3.16. Formularz edycji AWCZ

3.3. Testowanie działania aplikacji

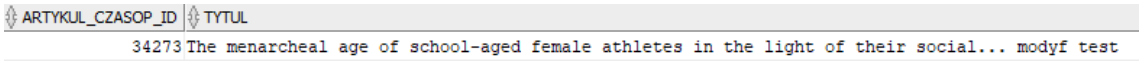
Do testowania aplikacji został wykorzystany dziennik konsoli, do którego to aplikacja podaje informacje na temat zapisywanych danych, oczywiście rozwiązanie to jest wykorzystywane jedynie w wersji deweloperskiej systemu a wersja produkcyjna nie posiada możliwości wyświetlania informacji nt. zapisywanej aktualnie wersji artykułu w czasopiśmie lub innego rodzaju publikacji. Rysunek 3.17 przedstawia obiekt AWCZ, który jest przekazywany do kontrolera AWCZ a następnie za pomocą modelu zapisywany do bazy danych. Widoczne na rysunku jest dodatkowe 5 list:

- ‘articleCharast‘ - lista odpowiadająca za cechy publikacji (’Recenzja’, ’Oryg. art. naukowy’ itp.);
- ‘authors‘ - lista odpowiadająca za przechowywanie wszystkich danych o autorach;

- ‘kw‘ - lista odpowiadająca za przechowywanie słów kluczowych, które są aktywne czyli są wyświetlane w opisie bibliograficznym;
- ‘kw_deleted‘ - lista odpowiadająca za przechowywanie słów kluczowych, które zostały usunięte w celu weryfikacji czy znajdują się one w bazie, jeśli tak zostają one usunięte;
- ‘physicalDesc‘ - lista odpowiadająca z opis fizyczny publikacji (‘mapy‘, ‘ilustracje‘, ‘fotografie‘ itp.);



Rysunek 3.17. Dziennik konsoli po zapisie danych w wersji deweloperskiej aplikacji



Rysunek 3.18. Formularz edycji AWCZ

Po poprawnej weryfikacji danych w dzienniku konsoli, dane są porównywane z danymi zapisanymi w bazie danych tak aby mieć pewność iż doszło do poprawnego zapisu danych, weryfikacja była dokonywana ręcznie. Na rysunku 3.18 widać powyższy przykład zapisany poprawnie po modyfikacji tytułu publikacji.

Rozdział 4

Podsumowanie

Dzięki zaimplementowaniu powyższych rozwiązań technologicznych udało się skutecznie zoptymalizować oraz zabezpieczyć system SKEP (część wprowadzania danych do bazy danych). Formularz AWCZ może z łatwością zostać przeniesiony na inne rodzaje publikacji, gdyż publikacje nie różnią się istotnie zamysłem opisu bibliograficznego oraz dzięki sposobowi implementacji powyższego rozwiązania, które zaprojektowane zostało z myślą o przenośności i elastyczności formularza. Implementacja postawiła wiele trudnych i skomplikowanych problemów inżynierskich, które zostały w sprytny, skuteczny, bezpieczny oraz, co najważniejsze, wydajny sposób rozwiązane tak, aby zachować integralność danych w bazie danych, nie wykonywać zbędnych transakcji na bazie danych oraz aby utrzymać formę zbliżoną do oryginalnej aplikacji oraz utrzymać założenia oryginalnej implementacji systemu SKEP (części do wprowadzania publikacji). Modyfikacja sposobu zapisywania czasopism do bazy danych oraz ich odpowiednie połączenie za pomocą relacji tabel umożliwiła na zmniejszenie duplikatów w bazie danych oraz pozwoliła na szybsze i sprawniejsze wprowadzanie publikacji bez dużego pola manewru do popełnienia błędu przez użytkownika końcowego. Aplikacja, dzięki swojemu ścisłemu połączeniu z ekosystemem SKEP oraz PraNet, jest bardzo bezpieczna oraz zamknięta tylko na środowisko uniwersytetu, co skutecznie utrudnia modyfikację oraz manipulację danych przez osoby trzecie. Zastosowanie frameworku Laravel umożliwia również duże możliwości rozbudowy systemu, dla przykładu implementacja wydawnictw zwartych sprowadza się jedynie do zmiany niektórych pól formularza oraz do modyfikacji zapytań do bazy danych, które to obsługiwać będą w/w. wydawnictwa zwarte.

Bibliografia

[1] Christopher John Pecoraro. *Mastering Laravel*. Packt Publishing, 2015.

[2] Sanjay Mishra and Alan Beaulieu. *Mastering Oracle SQL, 2nd Edition*. O'Reilly Media, 2004.

[3] Michał Widera. *Oracle Form Builder*. Helion, 2001.

[4] Jon Duckett. *PHP MySQL: Server-side Web Development*. Helion, 2023.

[5] Marijn Haverbeke. *Eloquent JavaScript*. No starch press, 2018.

[6] Jeppe Schaumburg Jensen. *The Missing Bootstrap 5 Guide*. Packt Publishing, 2022.

[7] Kevin Brown and Alexander Weissman. *Select2 The jQuery replacement for select boxes*. `https://select2.org/`, 11 2023.

[8] Limon Monte. *Sweetalert2 A beautiful, responsive, customizable, accessible (WAI-ARIA) replacement for JavaScript's popup boxes - Zero dependencies*. `https://sweetalert2.github.io/v9.html`, 11 2023.

[9] Steven Feuerstein and Bill Pribyl. *Oracle PL/SQL Programming: Covers Versions Through Oracle Database 12c*. O'Reilly Media, 2014.

[10] Mozilla. *MDN web docs*. `https://developer.mozilla.org/en-US/docs/Web/JavaScript`, 11 2023.

[11] Mozilla. *MDN web docs - addEventListener*. `https://developer.mozilla.org/en-US/docs/Web/API/EventTarget/addEventListener`, 11 2023.

[12] Mozilla. *MDN web docs - removeEventListener*. `https://developer.mozilla.org/en-US/docs/Web/API/EventTarget/removeEventListener`, 11 2023.

[13] Anuradha Aggarwal. *Event Capturing Vs Event Bubbling In Javascript*. `https://anuradha.hashnode.dev/event-capturing-vs-event-bubbling-in-javascript`, 11 2023.