

# Aplikacja webowa automatyzująca proces rekrutacji do szkół ponadpodstawowych

(A web application automating  
the recruitment process for secondary schools)

Julia Noczyńska

Praca inżynierska

**Promotor:** dr Wiktor Zychla

Uniwersytet Wrocławski  
Wydział Matematyki i Informatyki  
Instytut Informatyki

25.01.2025



## Streszczenie

Celem niniejszej pracy jest zaprojektowanie i implementacja aplikacji internetowej wspomagającej proces rekrutacji do szkół ponadpodstawowych. Aplikacja udostępnia kandydatom takie funkcjonalności, jak: składanie wniosków, obliczanie punktów rekrutacyjnych oraz wgląd do oferty edukacyjnej. System oferuje również interfejs dla administratorów szkolnych, który wspiera zarządzanie procesem przez szkoły. Do implementacji aplikacji wykorzystano między innymi React.js do stworzenia warstwy frontendowej oraz Node.js do warstwy backendowej aplikacji.

---

The aim of this study is to design and implement a web application supporting the secondary school recruitment process. The application provides candidates with features such as: application submitting, calculating recruitment points and access to the educational offer. The system also offers an interface for school staff, which supports the management process by schools. The implementation of the application uses React.js for the frontend and Node.js for the backend layer of the application.



# Spis treści

<b>1. Wprowadzenie</b>	<b>9</b>
1.1. Motywacja . . . . .	9
1.2. Porównanie z innymi implementacjami . . . . .	10
1.2.1. Nabór Szkoły ponadpodstawowe VULCAN . . . . .	10
1.2.2. Kompleksowy System Rekrutacji do szkół ponadpodstawowych	10
1.3. Plan pracy . . . . .	11
<b>2. Część dla użytkowników</b>	<b>13</b>
2.1. Poziomy dostęp użytkowników aplikacji . . . . .	13
2.2. Wymagania funkcjonalne . . . . .	13
2.2.1. Gość . . . . .	13
2.2.2. Użytkownik . . . . .	14
2.2.3. Administrator główny . . . . .	14
2.2.4. Administrator szkolny . . . . .	14
2.3. Wymagania niefunkcjonalne . . . . .	14
2.4. Sposób instalacji . . . . .	15
2.4.1. Wymagania wstępne . . . . .	15
2.4.2. Wersje z których korzystano podczas tworzenia aplikacji . . .	15
2.4.3. Instalacja . . . . .	15
2.5. Podręcznik użytkownika . . . . .	16
2.5.1. Terminy . . . . .	16
2.5.2. Oferta Edukacyjna . . . . .	17
2.5.3. Rejestracja oraz logowanie . . . . .	17

2.5.4.	Rejestracja kandydata . . . . .	18
2.5.5.	Widok strony głównej zalogowanego użytkownika . . . . .	19
2.5.6.	Składanie wniosku . . . . .	20
2.5.7.	Dodawanie wyników . . . . .	21
2.5.8.	Obliczanie punktów rekrutacyjnych . . . . .	22
2.5.9.	Status aplikacji . . . . .	22
2.6.	Podręcznik administratora szkolnego . . . . .	23
2.6.1.	Menu szkół . . . . .	23
2.6.2.	Menu profili . . . . .	23
2.6.3.	Dodawanie oraz edycja profilu . . . . .	23
2.6.4.	Ranking kandydatów . . . . .	25
2.7.	Podręcznik dla administratora głównego . . . . .	26
2.7.1.	Edycja szkół oraz terminów . . . . .	27
2.7.2.	Uruchamianie Naboru . . . . .	27
<b>3.</b>	<b>Część dla programistów</b>	<b>29</b>
3.1.	Użyte technologie . . . . .	29
3.1.1.	Język programowania . . . . .	29
3.1.2.	Backend . . . . .	29
3.1.3.	Frontend . . . . .	30
3.1.4.	Testy . . . . .	30
3.1.5.	Baza danych . . . . .	31
3.1.6.	System kontroli wersji . . . . .	31
3.1.7.	Diagramy i wykresy . . . . .	31
3.2.	Frontend . . . . .	31
3.2.1.	Atomic Design Pattern . . . . .	31
3.2.2.	Feature-based Structure . . . . .	32
3.2.3.	Testy E2E . . . . .	32
3.2.4.	Stylowanie . . . . .	33
3.2.5.	React . . . . .	34

3.3. Backend . . . . .	35
3.3.1. Architektura MVC . . . . .	35
3.3.2. Wzorzec repozytorium . . . . .	35
3.3.3. Wzorzec serwisów . . . . .	36
3.3.4. Middlewares . . . . .	36
3.3.5. Autoryzacja oparta na rolach . . . . .	36
3.4. Baza danych . . . . .	37
3.4.1. Encje szkoły . . . . .	38
3.4.2. Encja profili . . . . .	38
3.4.3. Encja ocen . . . . .	38
3.4.4. Encja użytkowników . . . . .	40
3.4.5. Encja aplikacji . . . . .	41
3.5. Statystki kodu . . . . .	41
<b>4. Algorytmy i struktury danych</b>	<b>43</b>
4.1. Algorytm Naboru . . . . .	43
4.1.1. Idea . . . . .	43
4.1.2. Pseudokod . . . . .	44
4.1.3. Oszacowanie złożoności . . . . .	45
4.1.4. Podsumowanie algorytmu . . . . .	46
<b>5. Podsumowanie</b>	<b>47</b>
5.1. Co udało się zrobić . . . . .	47
5.2. Plany na rozwój aplikacji . . . . .	48
<b>Bibliografia</b>	<b>49</b>





# Rozdział 1.

## Wprowadzenie

### 1.1. Motywacja

Rekrutacja do szkół to złożony proces, które obejmuje zarządzanie dużą ilością danych oraz ich przetwarzanie zgodnie z ustalonymi kryteriami, takimi jak wyniki egzaminów, oceny na świadectwie, preferencje kandydatów oraz kryteria rekrutacyjne oddziałów.

Składa się on z kilku etapów.

Pierwszy etap polega na przygotowaniu oferty edukacyjnej - czyli listy oddziałów (lub szerzej - profili) w poszczególnych szkołach wraz z kryteriami naboru.

Następnie kandydaci składają wnioski, w ramach których wybierają interesujące ich oddziały/profile w szkołach. Bywa, że stosowane jest ograniczenie liczby możliwych do wybrania oddziałów/profilu/szkół, choć ma to charakter historyczny i nie ma uzasadnienia w przypadku procesu elektronicznego. Kandydaci ustalają również swoje własne priorytety dla każdego profilu.

Na etap ten składa się również dostarczenie wyników kandydatów ze świadectwa oraz z egzaminu, na podstawie których będą później obliczane zdobyte punkty. W oparciu o zdobyte przez kandydatów punkty, tworzone są listy rankingowe dla poszczególnych oddziałów.

Drugim etapem rekrutacji jest rozpatrzenie wszystkich wniosków. W ustalonym terminie przeprowadzany jest proces naboru, podczas którego tworzone są listy rankingowe kandydatów aplikujących do danych oddziałów/profilu.

Za kwalifikujących się aplikantów uznaje się te osoby, które mają najwięcej punktów oraz mieszczą się w limicie dostępnych miejsc dla oddziału/profilu. Pozytywnie rozpatrywane jest tylko zgłoszenie o najwyższym priorytecie spośród kwalifikujących się aplikacji. Takie rozwiązanie zapobiega blokowaniu miejsc dla innych kandydatów.

Ponadto, po zakończeniu tego procesu, uruchamiane są tury uzupełniające, w których cały proces przeprowadzany jest ponownie dla oddziałów/profilu, w których zostały wolne miejsca.

Ręczne przeprowadzenie takiego procesu jest czasochłonne i podatne na błędy. Automatyzacja istotnie zwiększa jego efektywność oraz minimalizuje ryzyko wystąpienia błędów.

Celem niniejszej pracy jest zaprojektowanie oraz implementacja systemu informatycznego realizującego proces elektronicznego naboru do szkół ponadpodstawowych. System jest aplikacją internetową. Jego zadaniem jest wspieranie wszystkich etapów rekrutacji, począwszy od składania wniosków rekrutacyjnych przez kandydatów, po tworzenie list rankingowych i przydział uczniów do oddziałów.

Docelowymi użytkownikami systemu są prawni opiekunowie kandydatów, którzy w ich imieniu składają wnioski, a także administracja szkolna odpowiedzialna za zarządzanie oddziałami w placówkach szkolnych oraz za kontrolę procesu rekrutacyjnego.

Kod źródłowy systemu dostępny jest w repozytorium projektu pod adresem:

<https://github.com/julgitt/Electronic-School-Enrollment-System>

## 1.2. Porównanie z innymi implementacjami

### 1.2.1. Nabór Szkoły ponadpodstawowe VULCAN

<https://www.vulcan.edu.pl/samorzady/oprogramowanie/systemy-naborowe>

System stworzony został przez firmę VULCAN. Używany jest m.in. w Warszawie, Powiecie Łódzkim Wschodnim oraz na Śląsku.

Porównując obie implementacje, można zauważyć różnicę w podejściu do składania wniosków. W mojej aplikacji jest podejście zbiorcze do kandydatów. Opiekun prawny rejestruje każde dziecko oraz ma wygodny dostęp do wniosków wszystkich kandydatów nad którymi sprawuje opiekę. W systemie Nabór Szkoły ponadpodstawowe VULCAN podejście jest indywidualne - kandydat podaje swój pesel oraz rejestruje się otrzymując dostęp do swojego konta, wraz ze złożonym wnioskiem rekrutacyjnym.

### 1.2.2. Kompleksowy System Rekrutacji do szkół ponadpodstawowych

<https://rekrutacje.edu.wroclaw.pl/omikron-public/offer/search>

Następnym systemem, który różni się od proponowanej przeze mnie implementacji, jest system naboru elektronicznego wykorzystywany we Wrocławiu. Systemy używane w Polsce na szeroką skalę korzystają z profilu zaufanego, co zapewnia wysoki poziom bezpieczeństwa. Moja aplikacja upraszcza proces logowania wykorzystując uwierzytelnianie za pomocą loginu i hasła.

Obie aplikacje udostępniają podobny zestaw dodatkowych funkcjonalności dostępnych dla kandydata, takich jak harmonogram czy oferta edukacyjna.

Nie znalazłam informacji o tym, czy użytkownikami tych systemów może być również administracja szkolna.

W aplikacji, którą stworzyłam jest dostępny interfejs dla administratorów szkolnych, którzy mogą zarządzać swoimi placówkami w systemie rekrutacji. Moja aplikacja udostępnia również interfejs dla administratorów głównych, którzy w wygodny sposób mogą nadzorować proces rekrutacji, a także generować raport z przeprowadzonego naboru.

Ze względu na brak dostępu do kodu źródłowego innych implementacji, porównanie na poziomie technicznym nie było możliwe.

### 1.3. Plan pracy

W pierwszym rozdziale przedstawiono cel oraz dokonano porównania z innymi istniejącymi implementacjami.

Rozdział drugi zawiera część dla użytkownika, w której zawarte są wymagania funkcjonalne i нефункционалне, instrukcja instalacji oraz podręcznik użytkownika, administratora szkolnego i administratora głównego.

W trzecim rozdziale opisano część dla programisty, w której przybliżono użyte w projekcie technologie i zastosowane rozwiązania, a także przedstawiono diagramy związków encji dla użytej w projekcie bazy danych.

Czwarty rozdział zawiera opis algorytmu naboru, wraz z jego pseudokodem oraz oszacowaną złożonością.

Piąty rozdział zawiera podsumowanie, w którym jest opisane, co udało się zrealizować oraz jakie są plany na dalszy rozwój aplikacji.



## Rozdział 2.

# Część dla użytkowników

### 2.1. Poziomy dostęp użytkowników aplikacji

System rekrutacji został zaprojektowany z uwzględnieniem różnych poziomów dostępu użytkowników, aby zapewnić bezpieczeństwo danych i funkcjonalności adekwatne do roli użytkownika.

- **Gość** – osoba odwiedzająca aplikację bez zalogowania, posiadająca ograniczone uprawnienia.
- **Użytkownik** – opiekun prawny kandydata, który korzysta z systemu w celu rejestracji kandydata i składania aplikacji w jego imieniu.
- **Administrator główny** – osoba odpowiedzialna za zarządzanie procesem rekrutacji, np. ustalanie terminów, dodawanie szkół oraz uruchamianie procesu naboru.
- **Administrator szkolny** – osoba zarządzająca ofertą edukacyjną oraz kryteriami rekrutacyjnymi oddziałów w swoich placówkach.

### 2.2. Wymagania funkcjonalne

#### 2.2.1. Gość

- Możliwość zalogowania się do swojego konta przy użyciu swojej nazwy użytkownika lub adresu e-mail oraz hasła.
- Możliwość utworzenia nowego konta za pomocą formularza rejestracji konta użytkownika.
- Dostęp do informacji o ustalonych terminach związanych z naborem.
- Dostęp do oferty edukacyjnej.

### 2.2.2. Użytkownik

- Dodawanie, usuwanie oraz edycja kandydatów.
- Dla każdego kandydata:
  - Składanie aplikacji do wybranych oddziałów.
  - Wysyłanie formularza z ocenami kandydata.
  - Kalkulator punktów do wybranego profilu.
  - Dostęp do statusu złożonych wniosków.

### 2.2.3. Administrator główny

- Uruchamianie naboru.
- Wgląd do wyników naboru wraz z możliwością wyeksportowania pliku w formacie .csv.
- Zarządzanie terminami rekrutacji.
- Dodawanie, usuwanie i edycja szkół.

### 2.2.4. Administrator szkolny

- Dodawanie, usuwanie i edycja oddziałów/profilu w szkołach, do których dany administrator ma uprawnienia.
- Definiowanie kryteriów rekrutacyjnych do dodawanych/edytowanych oddziałów/profilu. Są nimi oceny z przedmiotów na świadectwie brane pod uwagę przy rekrutacji.
- Wgląd w listy rankingowe kandydatów dla poszczególnych profili.
- Możliwość odrzucania aplikacji kandydatów przyjętych w poprzednich turach (co odpowiada za sytuację, gdy użytkownik nie dopełnił formalności, co wiąże się ze zrezygnowaniem ze szkoły).

## 2.3. Wymagania niefunkcjonalne

- **Bezpieczeństwo:** Hasła użytkowników, są przechowywane w sposób bezpieczny z wykorzystaniem biblioteki haszującej hasła - Bcrypt[5].
- **Dostosowanie wyglądu:** Motyw strony aplikacji dostosowuje się do ustawień systemowych. Jeżeli użytkownik preferuje ciemny motyw w swoim systemie, aplikacja automatycznie dostosowuje się do tych preferencji.

## 2.4. Sposób instalacji

### 2.4.1. Wymagania wstępne

- Zainstalowane środowisko Node.js oraz menedżer pakietów npm.
- Zainstalowany system baz danych.
- Utworzona baza danych, do której zaimportowana zostanie kopia zapasowa bazy danych.

### 2.4.2. Wersje z których korzystano podczas tworzenia aplikacji

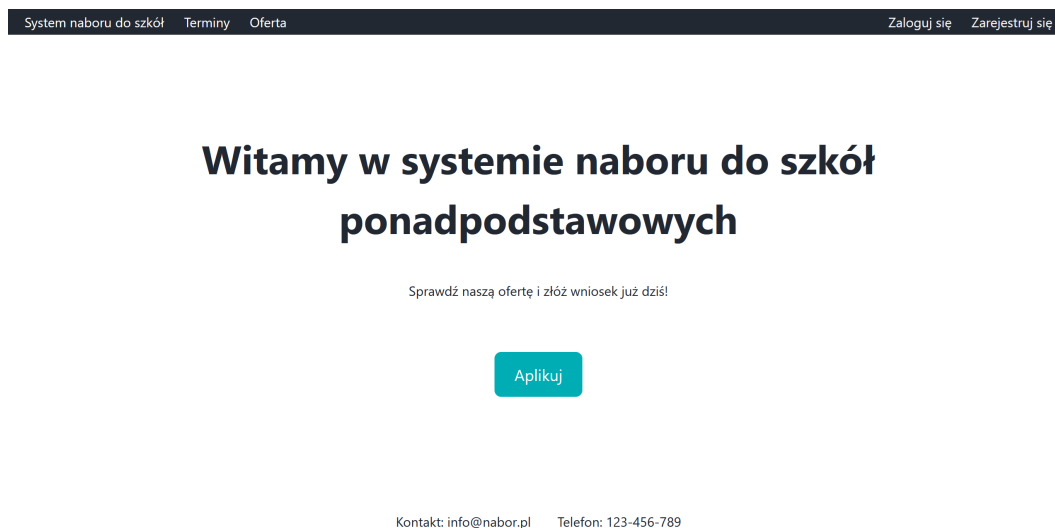
- Node.js: 20.11.1[2]
- npm: 10.2.4[3]
- PostgreSQL[15] oraz pgAdmin4[29] w wersji 14.5 (testowane również na wersji 17.2)

### 2.4.3. Instalacja

1. Sklonuj repozytorium znajdujące się pod adresem:  
`https://github.com/julgitt/Electronic-School-Enrollment-System`
2. Przejdź do katalogu projektu.
3. Aplikacja jest podzielona na dwie warstwy: frontendową w katalogu `/client` oraz backendową w katalogu `/server`. Należy przejść do obu podkatalogów i uruchomić polecenie `npm install` w każdym z nich, aby zainstalować brakujące zależności.
4. Plik `/server/.env` zawiera parametry konfiguracyjne, które muszą być uzupełnione, aby zapewnić prawidłowe działanie aplikacji.
5. Przejdź do katalogu `public/database/db_backups`. W tym miejscu znajdują się pliki z kopiami zapasowymi bazy danych. Należy zaimportować jedną z nich do utworzonej wcześniej bazy danych. Aby zaimportować dane użyłam polecenia `psql -U <DB_USER> -d <DB_NAME> -f <nazwa pliku z kopią bazy danych>`. Zmienne `<DB_USER>` `<DB_NAME>` pochodzą z pliku `/server/.env`.
6. Aby uruchomić warstwę frontendową przejdź do podfolderu `/client` i uruchom polecenie `npm start`
7. Aby uruchomić warstwę backendową przejdź do podfolderu `/server` i uruchom polecenie `npm start`
8. Aby wejść na stronę przejdź pod adres: `http://localhost:5173/`

## 2.5. Podręcznik użytkownika

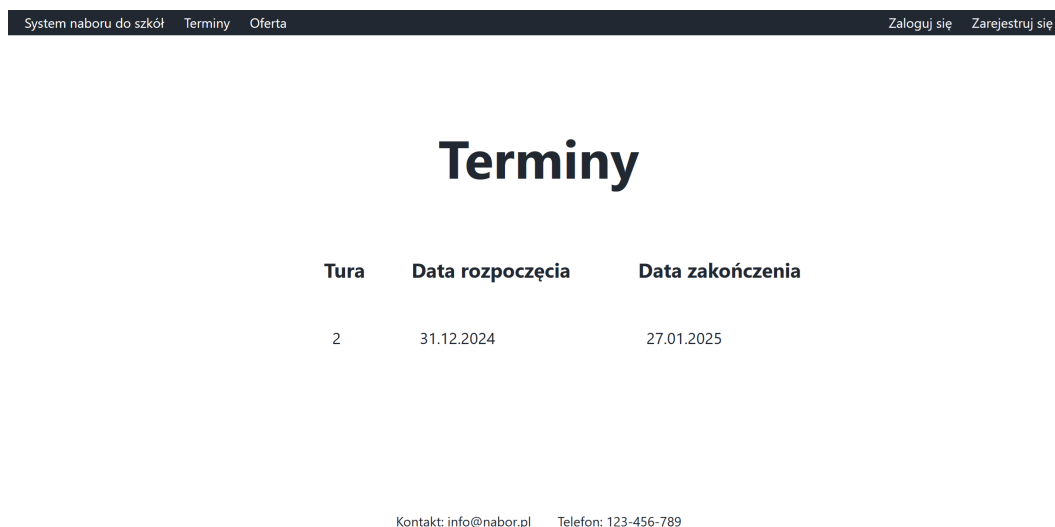
Użytkownik po wejściu na stronę główną jako gość, zobaczy główny panel zachęcający do złożenia aplikacji oraz nagłówek aplikacji na którym znajdują się dostępne funkcjonalności. Widok strony głównej przedstawiony jest na rysunku 2.1. Korzystanie z każdej z dostępnych funkcjonalności zostało opisane w tym poradniku.



Rysunek 2.1: Strona główna

### 2.5.1. Terminy

Widok terminów, widoczny na rysunku 2.2, wyświetla terminy przyszłych tur naboru ustalone przez administratora systemu.



Rysunek 2.2: Widok terminów



### 2.5.2. Oferta Edukacyjna

Oferta edukacyjna, przedstawiona na rysunku 2.3, jest funkcjonalnością dostępną zarówno dla zalogowanych użytkowników, jak i dla gości. Jest to wyszukiwarka profili, w której istnieje możliwość filtrowania po wielu czynnikach, takich jak: nazwa szkoły, nazwa profilu, przedmioty, które są w kryteriach rekrutacyjnych. Profile można również posortować pod względem ilości osób składających aplikację do danego oddziału, aby sprawdzić jakie jest zainteresowanie danym profilem.

System naboru do szkół
Terminy
Oferta
Zaloguj się
Zarejestruj się

### Filtry:

język|

język obcy nowożytny

język polski

dodaj filtrowanie nazwy profilu

dodaj filtrowanie nazwy szkoły

### Sortowanie:

☒ Sortuj rosnąco po popularności
☐ Sortuj malejąco po popularności

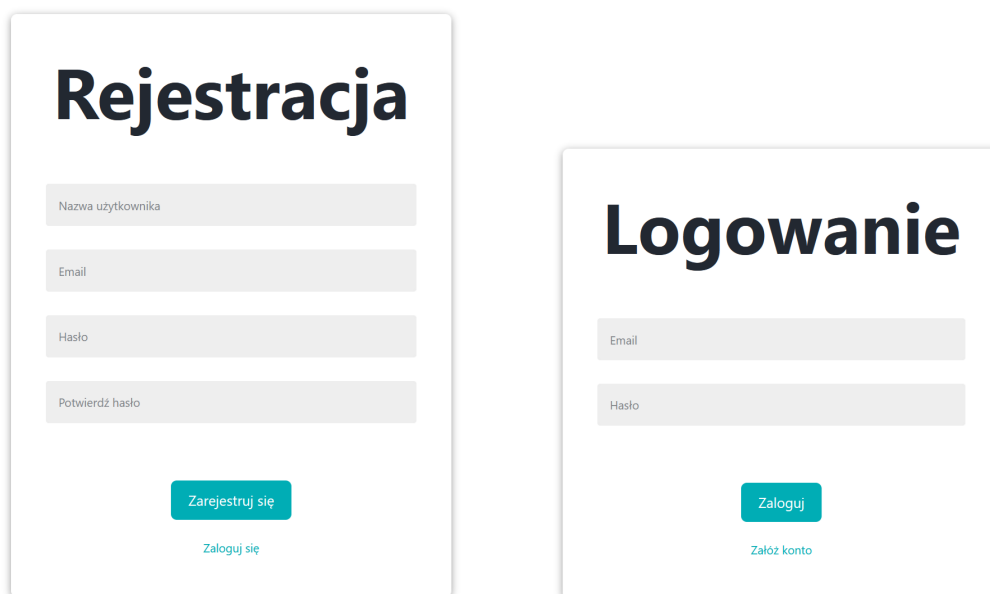
### Wyniki:

Nazwa profilu	Nazwa Szkoły	Liczba kandydatów
informatyczny	Technikum nr 24	0
biologiczny	Technikum nr 24	0
historyczny	Liceum ogólnokształcące nr 1	0

Rysunek 2.3: Oferta Edukacyjna

### 2.5.3. Rejestracja oraz logowanie

Aby utworzyć konto należy przejść do widoku rejestracji 2.4a i uzupełnić potrzebne dane. Po zakończonej rejestracji użytkownik zostaje przekierowany do strony logowania 2.4b. Aby się zalogować, należy podać prawidłową nazwę użytkownika lub adres email oraz hasło.



The image shows two side-by-side forms. The left form is titled 'Rejestracja' (Registration) and contains four input fields: 'Nazwa użytkownika' (Username), 'Email', 'Hasło' (Password), and 'Potwierdź hasło' (Confirm password). Below the fields is a teal button labeled 'Zarejestruj się' (Register) and a link 'Zaloguj się' (Login). The right form is titled 'Logowanie' (Login) and contains two input fields: 'Email' and 'Hasło'. Below the fields is a teal button labeled 'Zaloguj' (Login) and a link 'Załóż konto' (Create account).

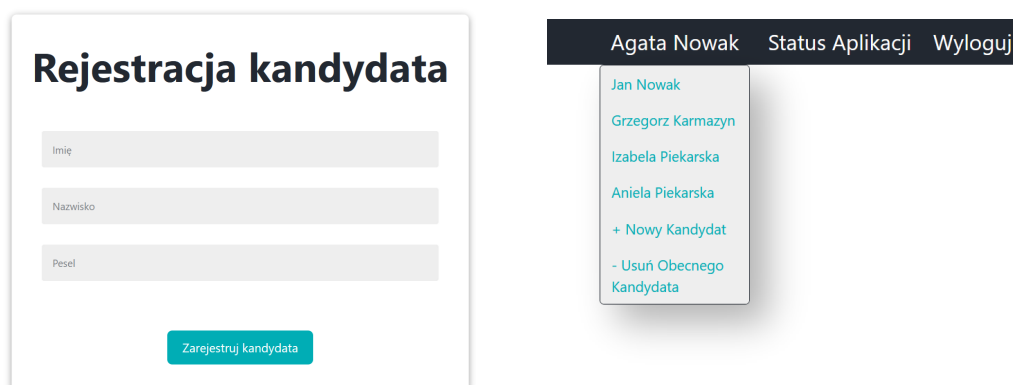
(a) Formularz rejestracji

(b) Formularz logowania

Rysunek 2.4

#### 2.5.4. Rejestracja kandydata

Po pierwszym zalogowaniu użytkownik zostaje poproszony o zarejestrowanie pierwszego kandydata 2.5a. Należy podać imię i nazwisko kandydata oraz numer pesel.



The image shows two components. On the left is a form titled 'Rejestracja kandydata' (Candidate registration) with three input fields: 'Imię' (First name), 'Nazwisko' (Last name), and 'Pesel'. Below the fields is a teal button labeled 'Zarejestruj kandydata' (Register candidate). On the right is a navigation bar with three items: 'Agata Nowak', 'Status Aplikacji', and 'Wyloguj'. Below the bar is a dropdown menu with the following items: 'Jan Nowak', 'Grzegorz Karmazyn', 'Izabela Piekarska', 'Aniela Piekarska', '+ Nowy Kandydat', '- Usuń Obecnego Kandydata'.

(a) Formularz rejestracji kandydata

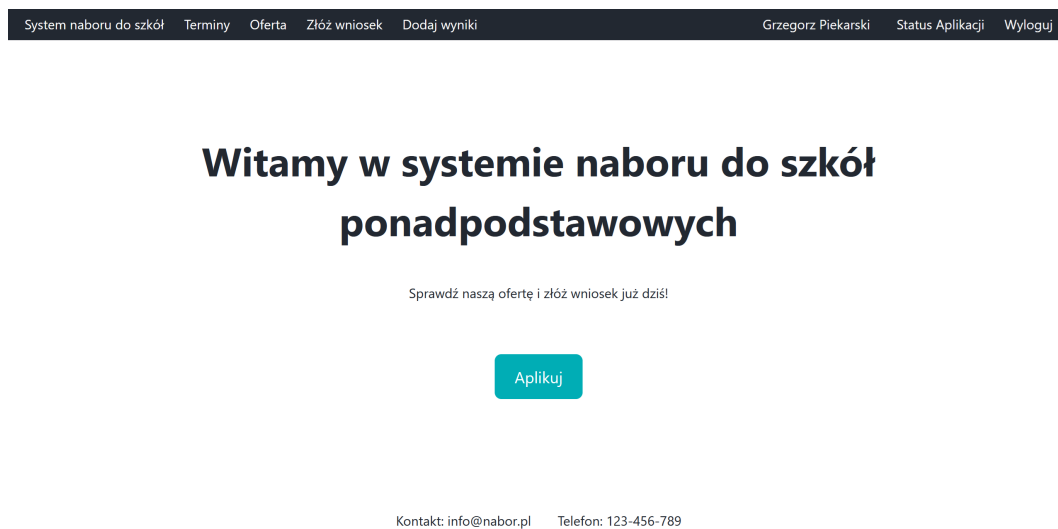
(b) Menu kandydatów

Rysunek 2.5

Po rejestracji kandydata, na pasku nawigacji po prawej stronie będą widoczne jego imię i nazwisko. Kandydatami można zarządzać za pomocą rozwijanego menu 2.5b. Po kliknięciu imienia i nazwiska obecnego kandydata wyświetli się drop-down, z którego można przełączać się pomiędzy kandydatami, oraz dodawać lub usuwać ich z systemu.

### 2.5.5. Widok strony głównej zalogowanego użytkownika

Po zalogowaniu, na pasku nawigacji pojawiają się dodatkowe opcje. Jeżeli aktualnie jest otwarty proces rekrutacyjny, wyświetla się opcja składania wniosku rekrutacyjnego oraz możliwość wprowadzenia wyników z egzaminu oraz ze świadectwa.



Rysunek 2.6: Widok strony głównej zalogowanego użytkownika

Użytkownik korzysta z funkcji w kontekście aktualnie wybranego kandydata. Na przykład, w sytuacji przedstawionej na rysunku 2.6, składanie wniosku będzie się odbywało w imieniu kandydata, który nazywa się Grzegorz Piekarski.

### 2.5.6. Składanie wniosku

Po kliknięciu w link "Złóż wniosek", znajdujący się na pasku nawigacyjnym, użytkownik zostaje przekierowany do formularza wyboru profili 2.7. W formularzu wprowadza szkoły oraz zaznacza profile, do których chce aplikować.

Rysunek 2.7: Formularz wyboru profili podczas składania wniosku

Aby dodać nową szkołę, należy kliknąć przycisk z symbolem plusa. Na formularzu pojawi się nowe pole tekstowe, w które wpisujemy nazwę szkoły. Podczas wpisywania nazwy, pod polem tekstowym wyświetli się lista proponowanych szkół, które pasują do wprowadzonego wyszukiwania. Kliknięcie na jedną z propozycji uzupełni pole tekstowe.

Następnie należy zaznaczyć, do których profili chcemy złożyć wniosek oraz nadać im priorytet za pomocą pola tekstowego znajdującego się po prawej od nazwy profilu. Mniejsza liczba oznacza wyższy priorytet. Należy pamiętać, że spośród wszystkich zakwalifikowanych aplikacji, jedynie ta o najwyższym priorytecie jest uznawana za przyjętą. Pozostałe aplikacje są odrzucone, aby nie blokować miejsc dla innych kandydatów.

Domyślnie możliwe jest aplikowanie do maksymalnie sześciu szkół. Nie ma natomiast ograniczenia na liczbę profili, do których aplikuje kandydat. To ograniczenie ma charakter arbitralny, od strony technicznej nie ma przeciwwskazań, aby przystosować aplikację do przyjmowania wniosków do dowolnej liczby szkół.

Formularz można zapisać za pomocą przycisku "Zapisz". Dopóki nie zakończy się proces składania wniosków, jest on dostępny do edycji.

### 2.5.7. Dodawanie wyników

Po kliknięciu w link "Dodaj wyniki", znajdujący się na pasku nawigacyjnym, użytkownik zostaje przekierowany do formularza wpisywania wyników 2.8a. W formularzu wprowadza swoje wyniki z egzaminu oraz ze świadectwa. Na ich podstawie będzie liczona punktacja, zdobyta podczas rekrutacji do wybranych profili.

**Wpisz swoje wyniki:**

Z egzaminu		Ze świadectwa	
matematyka:	73	matematyka:	4
język polski:	82	fizyka:	5
język obcy nowożytny:	90	informatyka:	6
		chemia:	5
		biologia:	5
		geografia:	3
		historia:	4
		język polski:	5
		język obcy nowożytny:	6

**Przelicznik punktów rekrutacyjnych**

Technikum nr 24

informatyczny

**Oceny ze świadectwa:**

matematyka	5	Wymagane
fizyka	5	Wymagane
informatyka	5	Wymagane
chemia	4	Opcjonalne
język obcy nowożytny	3	Opcjonalne

**Punkty: 67.8**

(a) Formularz wyników

(b) Przelicznik punktów rekrutacyjnych

Rysunek 2.8

Wyniki uzupełniamy tylko raz. Nie ma możliwości edycji wyników, dlatego należy sprawdzić czy uzupełnione dane są poprawne przed wysłaniem.

Warto nadmienić, że weryfikacja prawidłowości wprowadzonych we wniosku danych (ocen/wyników egzaminów) odbywa się już po złożeniu przez kandydatów dokumentów, w szkołach tzw. *pierwszego wyboru*, czyli tych które na liście preferencji kandydatów znajdują się na pierwszych miejscach. Operatorzy systemu naboru w poszczególnych szkołach, po przyjęciu dokumentów od kandydatów, weryfikują dane wprowadzone przez kandydatów z danymi z przedstawionych dokumentów. Niektóre wcześniejsze systemy elektronicznego naboru przerzucały konieczność przenoszenia danych z dokumentów kandydata do systemu na operatorów szkolnych. We współczesnych systemach przyjęto jednak rozwiązanie, w którym to kandydaci wprowadzają wszystkie dane do systemu, a operatorzy szkolni te dane wyłącznie weryfikują. Nie ma to znaczenia dla technicznej części procesu, jednak usprawnia proces wprowadzania danych do systemu.

### 2.5.8. Obliczanie punktów rekrutacyjnych

Aplikacja udostępnia również funkcjonalność obliczania zdobytych punktów rekrutacyjnych do wybranego profilu. Po dodaniu wyników kandydata, pojawia się pozycja w menu o nazwie "Oblicz Punkty", która przenosi użytkownika do przelicznika punktów rekrutacyjnych 2.8b.

Po przejściu do strony, użytkownik może wybrać szkołę oraz profil, aby przeliczyć punkty osiągnięte dla danego profilu. Oprócz punktów wyświetlane są również przedmioty, należące do kryteriów rekrutacyjnych tego profilu oraz oceny uzyskane z tych przedmiotów przez kandydata.

### 2.5.9. Status aplikacji

Użytkownik ma możliwość zapoznania się ze statusem oraz informacjami o złożonych aplikacjach, klikając w link "Status Aplikacji", który prowadzi do widoku przedstawionego na rysunku 2.9.

System naboru do szkół

Terminy

Oferta

Złóż wniosek

Oblicz punkty

Grzegorz Karmazyn

Status Aplikacji

Wyloguj

Status Aplikacji

Szkoła	Profil	Status	Tura	Priorytet	Data złożenia aplikacji
Technikum nr 24	biologiczny	Przyjęty	2	1	6.01.2025
Technikum nr 24	informatyczny	Odrzucony	2	2	6.01.2025
Liceum ogólnokształcące nr 1	historyczny	Odrzucony	2	3	6.01.2025

Kontakt: info@nabor.pl

Telefon: 123-456-789

Rysunek 2.9: Widok statusów aplikacji

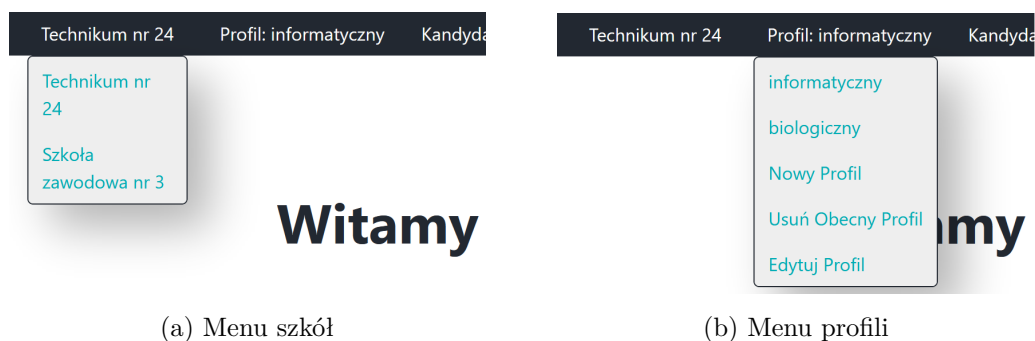
## 2.6. Podręcznik administratora szkolnego

Administrator szkolny rejestruje się oraz loguje w ten sam sposób co zwykły użytkownik. To administrator główny przypisuje odpowiednie uprawnienia w systemie dla tego użytkownika, czyniąc go administratorem szkolnym.

Administrator szkolny może zarządzać więcej niż jedną szkołą.

### 2.6.1. Menu szkół

Po lewej stronie paska nawigacyjnego znajduje się nazwa szkoły, w kontekście której administrator aktualnie pracuje. Klikając na nazwę szkoły, otwiera się menu szkół 2.10a, w którym administrator może przełączać się między szkołami.



Rysunek 2.10

Administrator szkolny nie może usuwać oraz dodawać szkół - tym zajmuje się administrator główny.

### 2.6.2. Menu profili

Obok menu szkół znajduje się nazwa profilu, po kliknięciu którego rozwija się menu pokazane na rysunku 2.10a. Menu profilu pozwala na przełączanie się między profilami, a także na edycję lub usunięcie obecnego profilu oraz na dodanie nowego.

### 2.6.3. Dodawanie oraz edycja profilu

Po przejściu do widoku edycji/tworzenia profilu, wyświetla się formularz 2.11, w którym należy nadać nazwę oraz określić liczbę dostępnych dla oddziału/profilu miejsc.

W tym miejscu warto zaznaczyć, że używane w tej pracy często określenie *oddziału/profilu* wynika z tego jak w rzeczywistości układane są oferty edukacyjne.

**Edycja profilu**

Nazwa profilu

0

**Kryteria**

Przedmiot	Obowiązkowy / Alternatywny	
matematyka	<input type="checkbox"/> O	<input type="checkbox"/> A
fizyka	<input type="checkbox"/> O	<input type="checkbox"/> A
informatyka	<input type="checkbox"/> O	<input type="checkbox"/> A
chemia	<input type="checkbox"/> O	<input type="checkbox"/> A
biologia	<input type="checkbox"/> O	<input type="checkbox"/> A
geografia	<input type="checkbox"/> O	<input type="checkbox"/> A
historia	<input type="checkbox"/> O	<input type="checkbox"/> A
język polski	<input type="checkbox"/> O	<input type="checkbox"/> A
język obcy nowożytny	<input type="checkbox"/> O	<input type="checkbox"/> A

Zapisz

Rysunek 2.11: Formularz edycji profilu

Czasem bywa tak, że wiadomo, iż zostanie utworzony jeden oddział o jakiejś charakterystyce (np. matematyczno-informatyczny) i może do niego aplikować 30 kandydatów. Bywa jednak też i tak, że planowanych oddziałów o takiej charakterystyce jest więcej (na przykład cztery) i wtedy opisuje się je jako jeden profil o 120 miejscach dla kandydatów. W takich przypadkach przypisanie 120 przyjętych kandydatów do poszczególnych czterech oddziałów nie jest już częścią procesu naboru.

Następnie wyborowi podlegają kryteria rekrutacyjne dla profilu. Są to przedmioty, których ocena ze świadectwa będzie brana pod uwagę przy obliczaniu punktacji podczas naboru. Kryteria obowiązkowe są wymagane, natomiast kryteria alternatywne to zbiór przedmiotów, spośród których liczy się tylko ten, z którego kandydat uzyskał najwyższą ocenę.

Wymagane jest podanie czterech kryteriów obowiązkowych. System pozwala na wybór dwóch kryteriów alternatywnych. Jeśli administrator zdecyduje się nie dodawać kryteriów alternatywnych, musi wskazać dodatkowy przedmiot obowiązkowy.



#### 2.6.4. Ranking kandydatów

Po kliknięciu pozycji na pasku nawigacji o nazwie "Kandydaci", wyświetlany jest ranking 2.12 dla aktualnie wybranego profilu.

##### Kandydaci

Aplikacje kwalifikujące się:				
Imię	Nazwisko	Punkty	Status	
Jan	Nowak	105.25	Oczekujący	
Grzegorz	Karmazyn	67.8	Oczekujący	

Aplikacje na liście rezerwowej:				
Imię	Nazwisko	Punkty	Status	
Grzegorz	Piekarski	0	Oczekujący	

Aplikacje zaakceptowane w poprzednich turach:				
Imię	Nazwisko	Punkty	Status	Odrzuć
Agata	Nowak	101.5	Przyjęty	<input type="checkbox"/>
Aniela	Piekarska	71.85	Przyjęty	<input type="checkbox"/>

Rysunek 2.12: Ranking kandydatów

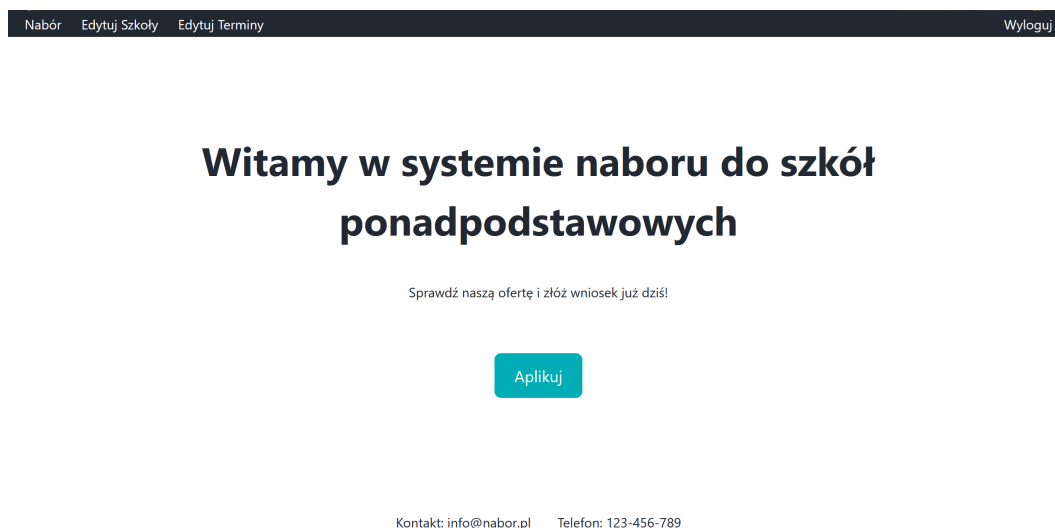
Składa się on z trzech list:

- **Lista aplikacji kwalifikujących się** - grupa kandydatów o największej liczbie punktów, którzy mieszczą się w ilości wolnych miejsc przypisanych do profilu.
- **Lista aplikacji na liście rezerwowej** - kandydaci, którym zabrakło punktów, aby znaleźć się wśród zakwalifikowanych. Kandydaci ci nadal mają szansę na przyjęcie do profilu, jeśli niektórzy z zakwalifikowanych kandydatów dostaną się do innego profilu o wyższym priorytecie.
- **Lista aplikacji zaakceptowanych w poprzednich turach** - grupa kandydatów, którzy zakwalifikowali się w poprzednich turach. Jeżeli któryś z kandydatów nie dostarczył dokumentów potwierdzających chęć zapisania do profilu, można odrzucić jego aplikację za pomocą przycisku w kolumnie o nazwie "Odrzucić".

## 2.7. Podręcznik dla administratora głównego

Aplikacja zapewnia również interfejs użytkownika dla administratora głównego. Aby zostać administratorem aplikacji, należy założyć konto jako zwykły użytkownik, a następnie dodać rekord w tabeli *user\_roles* w bazie danych, który będzie zawierał identyfikator nowo utworzonego użytkownika i rolę "admin".

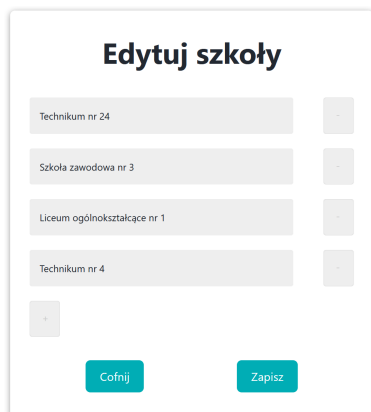
Na pasku nawigacji dla administratora jest dostępna edycja szkół 2.14a oraz terminów tur rekrutacji 2.14b. Ponadto istnieje też strona zarządzania samym naborem, do której można przejść wybierając pozycję "Nabór" znajdującą się na pasku nawigacji.



Rysunek 2.13: Widok strony głównej dla administratora

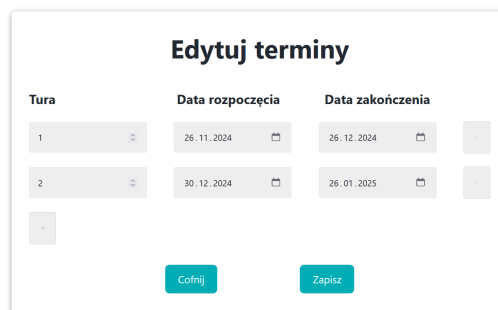
### 2.7.1. Edycja szkół oraz terminów

Administrator jest w stanie usuwać oraz edytować istniejące szkoły, a także dodawać nowe, klikając przycisk z symbolem plusa.



Formularz edycji szkół. Tytuł: **Edytuj szkoły**. Zawiera cztery pola tekstowe z przyciskami minus: "Technikum nr 24", "Szkoła zawodowa nr 3", "Liceum ogólnokształcące nr 1", "Technikum nr 4". Na dole znajduje się przycisk plus i dwa przyciski: "Cofnij" i "Zapisz".

(a) Formularz edycji szkół



Formularz edycji terminów. Tytuł: **Edytuj terminy**. Zawiera tabelę z kolumnami: "Tura", "Data rozpoczęcia", "Data zakończenia".

Tura	Data rozpoczęcia	Data zakończenia	
1	26. 11. 2024	26. 12. 2024	-
2	30. 12. 2024	26. 01. 2025	-
			-

Na dole znajdują się przyciski "Cofnij" i "Zapisz".

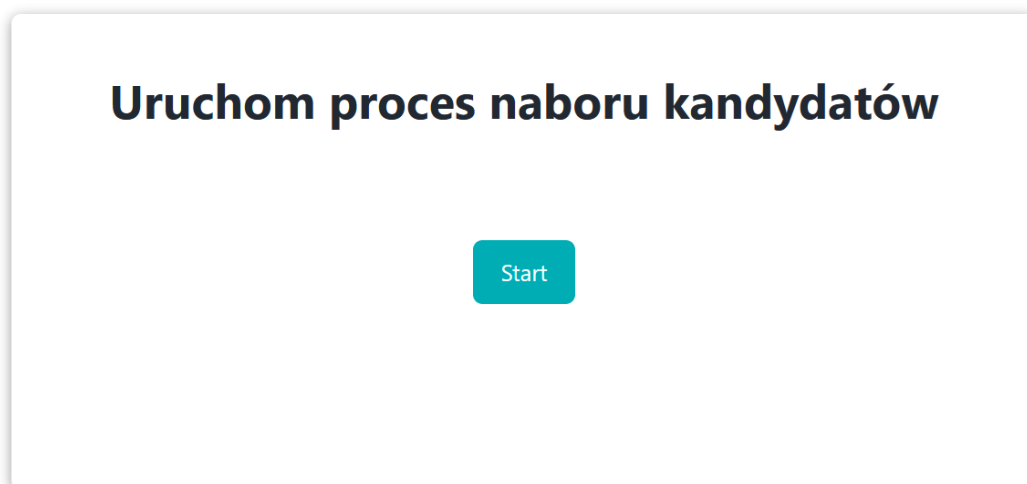
(b) Formularz edycji terminów

Rysunek 2.14

W podobny sposób można edytować terminy składania wniosków w poszczególnych turach.

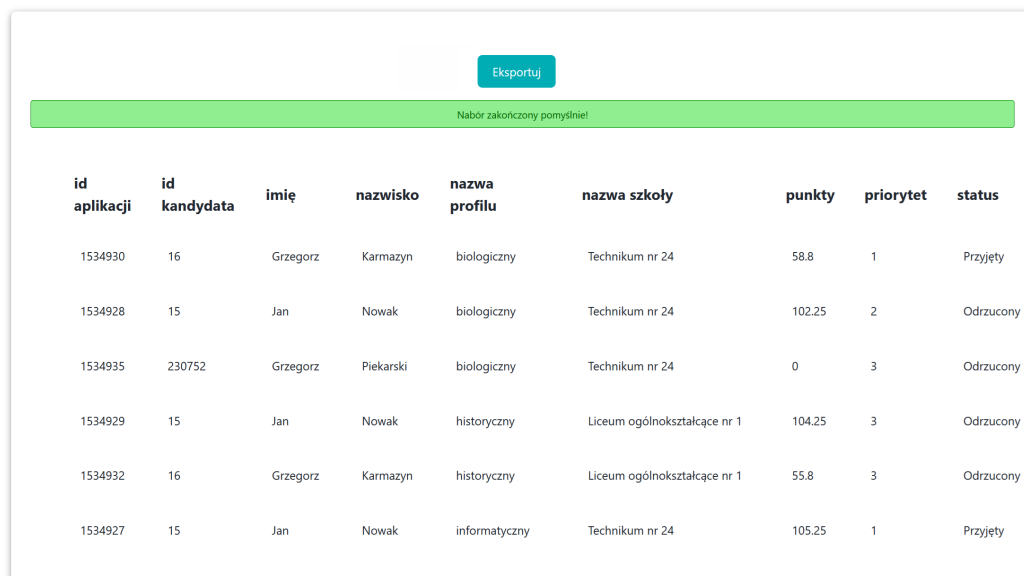
### 2.7.2. Uruchamianie Naboru

Aby uruchomić algorytm naboru należy nacisnąć przycisk "Start"<sup>2.15</sup> widoczny po przejściu do strony zarządzania naborem.



Strona z tytułem **Uruchom proces naboru kandydatów** i przyciskiem **Start**.

Rysunek 2.15: Przycisk służący do uruchomienia naboru



Nabór zakończony pomyślnie!								
Eksportuj								
id aplikacji	id kandydata	imię	nazwisko	nazwa profilu	nazwa szkoły	punkty	priorytet	status
1534930	16	Grzegorz	Karmazyn	biologiczny	Technikum nr 24	58.8	1	Przyjęty
1534928	15	Jan	Nowak	biologiczny	Technikum nr 24	102.25	2	Odrzucony
1534935	230752	Grzegorz	Piekarski	biologiczny	Technikum nr 24	0	3	Odrzucony
1534929	15	Jan	Nowak	historyczny	Liceum ogólnokształcące nr 1	104.25	3	Odrzucony
1534932	16	Grzegorz	Karmazyn	historyczny	Liceum ogólnokształcące nr 1	55.8	3	Odrzucony
1534927	15	Jan	Nowak	informatyczny	Technikum nr 24	105.25	1	Przyjęty

Rysunek 2.16: Informacje o naborze

Po zakończeniu naboru wyświetlana jest tabela zawierająca informacje o wszystkich aplikacjach 2.16, które brały w nim udział. W szczególności znajdują się w niej zaktualizowane statusy tych aplikacji.

Tabelę można wyeksportować jako plik w formacie .csv klikając przycisk "Eksportuj".

## Rozdział 3.

# Część dla programistów

### 3.1. Użyte technologie

#### 3.1.1. Język programowania

**TypeScript** [1] - jest to język programowania będący nadzbiorem języka JavaScript, który transpiluje się do JavaScriptu. Dodaje do JavaScriptu statyczne typowanie oraz modyfikatory dostępu i interfejsy. Statyczne typowanie było dla mnie szczególnie istotne, ponieważ uważam, że poświęcenie niewielkiej ilości dodatkowej uwagi na poprawne typowanie, przekłada się na dużą oszczędność czasu, który w przeciwnym razie poświęciłabym prawdopodobnie na debugowanie kodu. Błędne typowanie skutkuje w TypeScriptie błędami wykrywalnymi już na etapie kompilacji, co pozwala szybciej zidentyfikować problemy, które w przypadku braku statycznego typowania mogłyby zostać przeoczone, prowadząc do nieoczekiwanego działania programu. W zaproponowanej przeze mnie implementacji projektu korzystam z języka TypeScript zarówno w warstwie frontendowej jak i backendowej.

#### 3.1.2. Backend

- **Node.js** [2] - środowisko służące do uruchamiania języka JavaScript poza przeglądarką. W efekcie umożliwia programowanie aplikacji po stronie serwera. Dzięki Node.js można było wybrać jeden i ten sam język dla wytworzenia aplikacji dla obu warstw, frontendu i backendu. Zaletą tego środowiska jest to, że jest asynchroniczne, zatem zapytania do bazy danych czy odczyt plików, nie blokują głównego wątku programu. Dodatkowo, Node.js udostępnia repozytorium pakietów - **npm** [3], które ułatwia zarządzanie zależnościami i pozwala na łatwe dodawanie gotowych modułów do aplikacji.
- **Express** [4] - framework, który znacząco upraszcza zarządzanie żądaniami HTTP oraz pisanie kodu po stronie serwera. Narzędzie zawiera wsparcie dla routingu, parsowania adresu i parametrów, a także zarządzania ciasteczkami czy

sesją. Pozwala to skupić się na implementacji logiki biznesowej, bez konieczności pisania powtarzalnych fragmentów kodu. Ponadto Express obsługuje tzw. *middlewares*, czyli możliwość rozszerzenia potoku obsługi żądań o własne funkcje. Sprzyja to dzieleniu dużych zadań na mniejsze odpowiedzialności. W ten sposób dodaje się na przykład zarządzanie autentykacją czy tworzenie logów audytowych.

### 3.1.3. Frontend

- **React.js** [6] - biblioteka pozwalająca na tworzenie interfejsu użytkownika. React korzysta z tzw. wirtualnego DOM (reprezentacji drzewa dokumentu widocznego w przeglądarce internetowej), który służy jako lekka kopia prawdziwego DOM. Kiedy zmienia się stan aplikacji, React najpierw aktualizuje wirtualny DOM, a następnie porównuje go z rzeczywistym DOM-em, aktualizując jedynie różnice, a nie całą strukturę. Znacząco zwiększa to szybkość renderowania strony. Narzędzie zachęca również do tworzenia architektury w taki sposób, aby korzystać z mniejszych komponentów (obsługa widoku) i hooków (obsługa zmiany stanu i zdarzeń). Istotne jest również to, że w React ważny jest koncept niemutowanych stanów. Polega on na tym, że stan aplikacji nie powinien być modyfikowany bezpośrednio, tylko powinno tworzyć się jego kopię z wprowadzonymi zmianami.
- **Vite** [7] - tzw. bundler modułów - narzędzie służące do budowania i uruchamiania aplikacji frontendowych. Stworzenie projektu z Vite pozwoliło na szybkie, automatyczne utworzenie domyślnej konfiguracji dla aplikacji korzystającej z Reacta oraz TypeScriptu. Zaletą tego narzędzia jest również mechanizm Hot Module Replacement, które pozwala na automatyczne aktualizowanie się dokonanych w kodzie zmian w przeglądarce bez konieczności odświeżania strony. Głównym zadaniem narzędzia jest tłumaczenie napisanego przez programistę kodu TypeScript, na język zrozumiały dla przeglądarki - HTML, CSS oraz JavaScript.
- **CSS** [8] - język służący do pisania stylowania stron internetowych. Definiujemy w nim kolory, marginesy, czcionki oraz układ elementów.
- **SCSS** [9] - rozszerza składnię CSS.

### 3.1.4. Testy

- **Mocha** [11] - framework pozwalający na tworzenie testów jednostkowych oraz integracyjnych. Wspiera testowanie asynchronicznego kodu oraz współpracuje z innymi narzędziami służącymi do tworzenia testów.
- **Sinon** [12] - biblioteka do tworzenia obiektów zastępczych - tzw. mocków, stubów i szpiegów w testach jednostkowych. Pozwala między innymi zastępować

działania funkcji oraz monitorować ich wywołania, co umożliwia testowanie takich aspektów jak liczba wywołań funkcji czy sprawdzenie przekazywanych do niej argumentów.

- **Assert** [13] - moduł udostępniany przez Node.js zawierający zbiór funkcji - asercji - służących do sprawdzania, czy warunki testu są spełnione.
- **Cypress** [14] - framework do automatyzacji testów. W moim projekcie wykorzystałam go do tworzenia testów end-to-end.

### 3.1.5. Baza danych

**PostgreSQL** [15] - relacyjna baza danych, która obsługuje bardziej skomplikowane typy danych, takie jak tablice, albo dane w formacie JSON. Jest to oprogramowanie typu open-source. Ma dużą społeczność, która ciągle pracuje nad ulepszaniem systemu.

### 3.1.6. System kontroli wersji

- **Git** [16] - system kontroli wersji, pozwalający na zarządzanie wersjami kodu.
- **Github** [17] - przechowuje kod źródłowy projektu w repozytorium dostępnym online.

### 3.1.7. Diagramy i wykresy

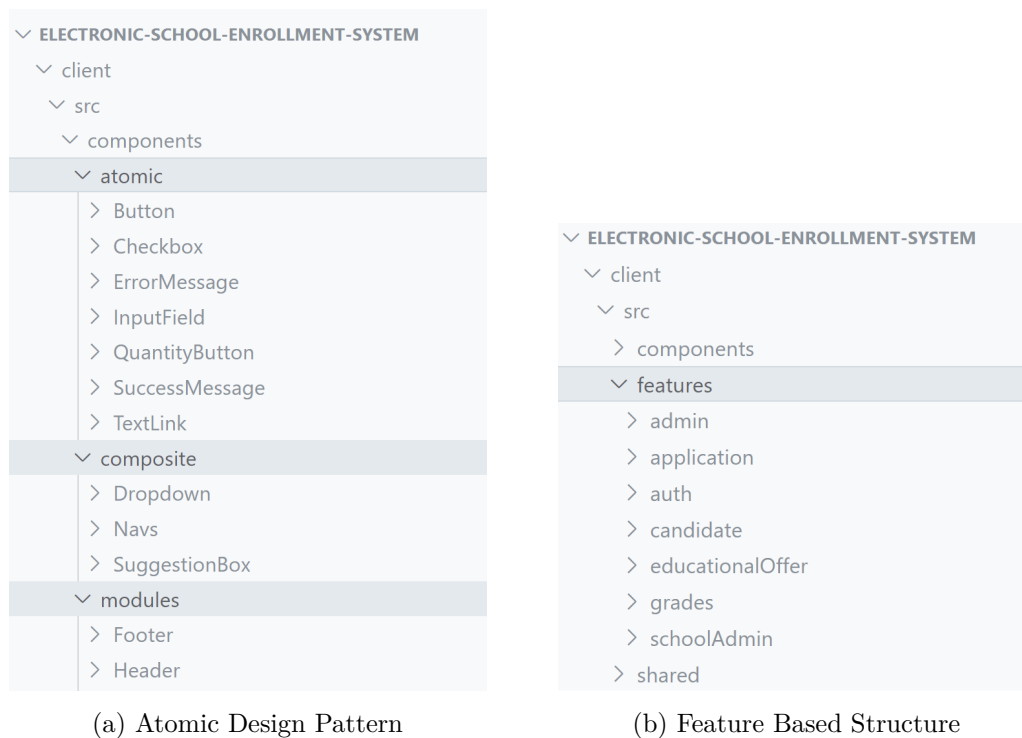
- **Python** [18] z biblioteką **Matplotlib** [19]
- **Visual Paradigm** [20]

## 3.2. Frontend

### 3.2.1. Atomic Design Pattern

Aby zorganizować strukturę plików zastosowałam wzorzec Atomic Design zaproponowany przez Brada Frosta [21]. Zakłada on podział aplikacji frontendowej na kilka zależnych od siebie poziomów.

Pierwszy poziom zwyczajowo nazywany jest Atoms. Definiuje on najmniejsze, najbardziej re-używalne komponenty, takie jak przyciski czy pola tekstowe. Następne poziomy składają się z elementów z poprzedzającego poziomu, coraz bardziej rozbudowując logikę komponentów. Struktura plików przedstawiająca wzorzec Atomic Design przedstawiona jest na rysunku 3.1a.



Rysunek 3.1

Dzięki temu struktura zorganizowana jest w intuicyjny sposób, a ilość powtarzającego się kodu znacznie się zmniejsza. Ponadto wzorzec ten zwiększa skalowalność systemu, ponieważ wprowadzanie kolejnych komponentów opiera się na wykorzystaniu tych już stworzonych. Jeśli zdecydujemy się zmienić jeden z komponentów, aktualizację wystarczy wprowadzić zwykle tylko w jednym miejscu w kodzie.

### 3.2.2. Feature-based Structure

Z czasem rozbudowywania projektu, zwiększała się liczba wprowadzanych funkcjonalności, a co za tym idzie, wzrastała potrzeba oddzielenia od siebie poszczególnych odpowiedzialności w strukturze plików. Wynikło z tego przeorganizowanie plików tak, aby pogrupować je ze względu na funkcjonalności [22]. Strukturę plików użytą w projekcie przedstawiono na rysunku 3.1b.

### 3.2.3. Testy E2E

Testy end-to-end umożliwiają sprawdzenie poprawności działania całej aplikacji, obejmując zarówno warstwę frontendową, odpowiedzialną między innymi za wyświetlanie interfejsu użytkownika, jak i backendową, w tym zapis do bazy danych. Do realizacji testów E2E użyto narzędzia Cypress. W ramach testów sprawdzono funkcjonalność logowania oraz rejestracji kandydatów.



Obecnie nie wszystkie funkcjonalności aplikacji zostały objęte testami e2e, jednak dzięki uprzedniej konfiguracji narzędzi, dodawanie nowych przypadków testowych ogranicza się do tworzenia nowych klas testów.

#### **3.2.4. Stylowanie**

Do stylowania użyłam SCSS Modules [10]. Podejście to zakłada tworzenie modularnych plików SCSS, które określają stylowanie dla danego komponentu.

### 3.2.5. React

#### React Router

Do obsługi nawigacji w aplikacji, użyłam biblioteki React Router [23]. Cała aplikacja jest ładowana tylko raz, natomiast poszczególne komponenty są renderowane w zależności od bieżącego adresu URL. Dzięki temu użytkownicy mogą poruszać się po aplikacji bez potrzeby przeładowywania całej strony. Rozwiązanie to nosi nazwę SPA - Single Page Applications.

Dodawanie nowej trasy polega na skorzystaniu z komponentu Route, któremu przekazujemy ścieżkę oraz odpowiadający jej komponent, który ma być renderowany. Fragment routingu utworzony w opisywanej aplikacji, wygląda w sposób przedstawiony na rysunku 3.2.

```
return (
  <Routes>
    { /* guest */ }
    <Route path="/" element={<Home/>} />
    <Route path="/dates" element={<Deadlines/>} />
    <Route path="/login" element={<Login/>} />
    <Route path="/signup" element={<Signup/>} />
    <Route path="/educationalOffer" element={<EducationalOffer/>} />
    { /* user */ }
    <Route path="/registerCandidate" element={<RegisterCandidate/>} />
    <Route path="/submitApplication" element={<SubmitApplication/>} />
    <Route path="/applicationStatus" element={<ApplicationStatus/>} />
    <Route path="/submitApplicationPastDeadline" element={<SubmitApplicationPastDeadline/>} />
  </Routes>
)
```

Rysunek 3.2: Routing napisany z wykorzystaniem biblioteki React Router

#### Context API

Do przechowywania globalnego stanu użyłam Context API [24]. Jest to narzędzie wbudowane w Reacta. Rozwiązanie to udostępnia komponent Providera, dzięki któremu można określić, które komponenty mają dostęp do globalnego kontekstu, oraz hook useContext pozwalający na pozyskanie danych z kontekstu.

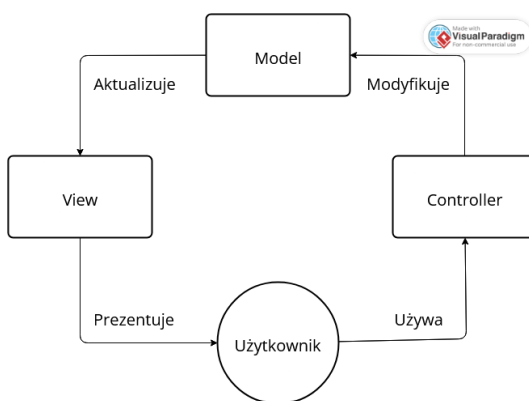
Z wykorzystaniem Context API stworzyłam scentralizowaną obsługę błędów.

### 3.3. Backend

#### 3.3.1. Architektura MVC

Architektura aplikacji opiera się na wzorcu Model View Controller [25]. Zakłada on podzielenie aplikacji na trzy części:

- Model - jest odpowiedzialny za przechowywanie i modyfikację danych. Obejmuje logikę biznesową oraz interakcje z bazą danych.
- View - odpowiada za prezentację danych. Implementacja tej warstwy znajduje się na frontendzie.
- Controller - zajmuje się odbieraniem żądań HTTP od klienta, delegacją zadań w celu ich przetworzenia oraz zwracaniem odpowiednich odpowiedzi.



Rysunek 3.3: Diagram wzorca MVC wykonany w aplikacji Visual Paradigm

#### 3.3.2. Wzorzec repozytorium

```
export class UserRepository {  
  
    async insert(newUser: UserEntity, t: ITask<any>): Promise<UserEntity> {  
        const userInsertQuery = `  
            INSERT INTO users (username, email, password)  
            VALUES ($1, $2, $3)  
            RETURNING id;  
        `;  
  
        const values = [newUser.username, newUser.email, newUser.password];  
        return t.one(userInsertQuery, values);  
    }  
}
```

Rysunek 3.4: Fragment repozytorium użytkownika

Wzorzec projektowy repozytorium [26] dodaje dodatkową abstrakcję, oddzielającą logikę dostępu do bazy danych od reszty aplikacji. Zadaniem repozytoriów jest zarządzanie dostępem do danych i wykonywanie zapytań do bazy danych.

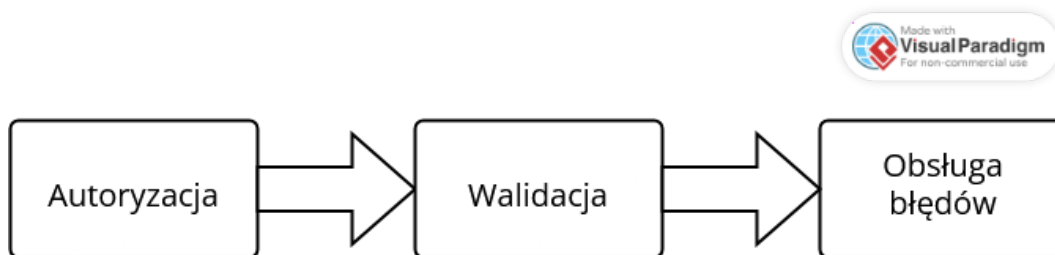
Zastosowanie tego wzorca ułatwiło testowanie logiki biznesowej, ponieważ pozwoliło na mockowanie tej części aplikacji w testach jednostkowych. Takie rozwiązanie zmniejsza koszty utrzymania aplikacji, ponieważ ułatwia refaktoryzację kodu w przyszłości oraz eliminuje redundancję kodu. Fragment implementacji wzorca repozytorium pochodzący z projektu zamieszczony jest na rysunku 3.4.

### 3.3.3. Wzorzec serwisów

Kolejną abstrakcją użytą w aplikacji jest warstwa serwisów [27], które implementują logikę biznesową oraz pośredniczą między kontrolerami a repozytoriami.

### 3.3.4. Middlewares

W aplikacji korzystam z middleware'ów do walidacji, autoryzacji oraz obsługi błędów. Łańcuch wywołań wygląda w sposób, przedstawiony na rysunku 3.5.



Rysunek 3.5: Middlewares

Walidacja żądań HTTP jest zaimplementowana przy użyciu modułu `express-validator`[28].

### 3.3.5. Autoryzacja oparta na rolach

W aplikacji uprawnienia dostępu przydzielane są na podstawie ról należących do użytkownika. Role przechowywane są w bazie danych w tabeli `user-roles`. Użytkownik jest w stanie posiadać kilka ról, choć w projekcie nie było potrzeby przypisywania więcej niż jednej roli dla jednego użytkownika. Obsługiwane przez aplikację role obejmują:

- **user** - rola zalogowanego użytkownika, bez uprawnień do administracji.
- **schoolAdmin** - rola użytkownika zalogowanego jako szkolny administrator.

- **admin** - rola zalogowanego użytkownika, będącego administratorem systemu.

Proces autoryzacji oparty na rolach przebiega następująco:

- Użytkownik loguje się.
- Jeśli dane są poprawne, serwer generuje ciasteczka zawierające informacje o jego identyfikatorze oraz rolach użytkownika. Ciasteczka mają włączone flagi **HttpOnly** oraz **Secure**. Pierwsza z flag uniemożliwia dostęp do ciasteczka z poziomu JavaScriptu po stronie klienta. Flaga Secure z kolei gwarantuje, że ciasteczko jest wysyłane przez zaszyfrowane połączenie HTTPS.
- W przypadku żądania do zasobu wymagającego uprawnień opartych na rolach, żądanie delegowane jest do middleware'a odpowiedzialnego za autoryzację, gdzie podejmowana jest decyzja, czy role użytkownika zawarte w ciasteczku są wystarczające do udzielenia dostępu.
- Po wylogowaniu użytkownika ciasteczka są usuwane.

### 3.4. Baza danych

Aplikacja automatyzująca proces elektronicznego naboru do szkół ponadpodstawowych powinna być w stanie przechowywać dużą ilość danych, które są ze sobą ściśle powiązane. Ilość użytkowników korzystająca z aplikacji może być liczona w tysiącach, a na każdego kandydata przypada dodatkowo kilka złożonych wniosków. Aby umożliwić efektywne przechowywanie oraz zarządzanie informacjami, naturalnym rozwiązaniem było wykorzystanie bazy danych.

Zdecydowałam się na wykorzystanie relacyjnej bazy danych PostgreSQL. Wybrałam bazę relacyjną, ponieważ cenię wysokie zorganizowanie danych i relacji między nimi oraz możliwość określania ścisłych typów dla kolumn w tabelach. Relacyjne bazy danych zapewniają dzięki temu większą integralność danych niż bazy nierelacyjne.

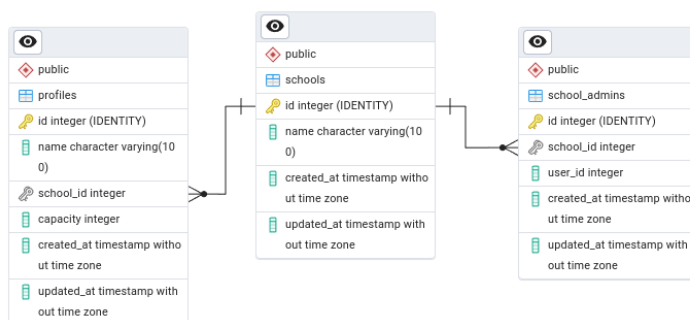
W bazie danych znajduje się jedenaście tabel.

Wszystkie klucze obce mają dodaną kaskadę `ON DELETE`, aby w razie usunięcia jakichś danych, odpowiadające im rekordy w powiązanych tabelach były automatycznie usuwane.

W celu poprawy czytelności, encje występujących w bazie danych zostały pogrupowane w pięciu diagramach. Diagramy wykonano za pomocą narzędzia ERD udostępnianego przez program pgAdmin [29].

### 3.4.1. Encje szkoły

Diagram encji szkoły przedstawiony jest na rysunku 3.6.



Rysunek 3.6: Diagram encji szkoły

Szkoła powiązana jest ze zbiorem profili, które udostępnia w ramach swojej oferty edukacyjnej. Aby odzwierciedlić tę zależność w bazie danych, tabela szkół (*schools*) połączona jest relacją *jeden do wielu* z tabelą profili (*profiles*). Encja szkoły zawiera również informację o nazwie szkoły.

Szkoły posiadają również administratorów, którzy nimi zarządzają. Jeden administrator może zarządzać wieloma szkołami oraz jedna szkoła, może być zarządzana przez wielu administratorów. Z tego względu powstała tabela administratorów szkolnych (*school\_admins*) łącząca encje użytkowników z encją szkół w relacji *wiele do wielu*.

### 3.4.2. Encja profili

Diagram encji profili przedstawiony jest na rysunku 3.7.

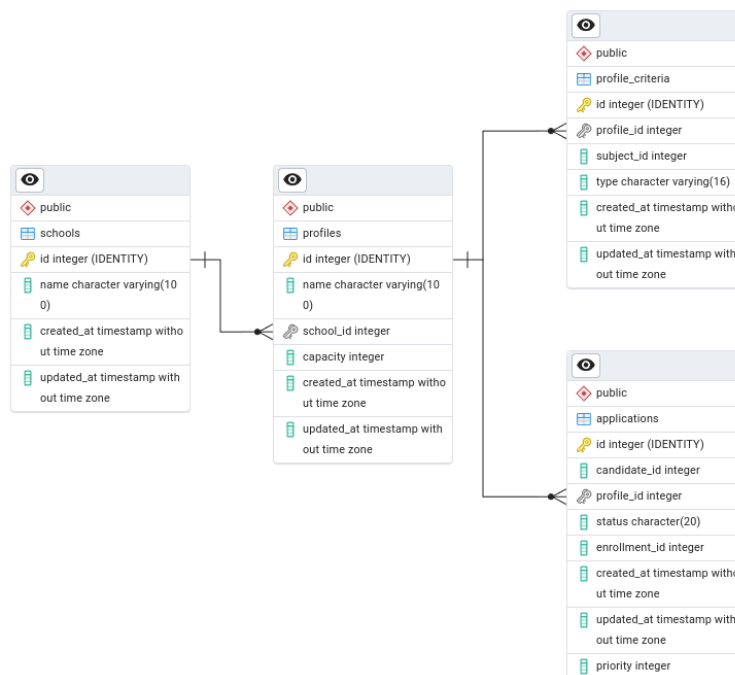
Encja profilu związana jest ze szkołą oraz z aplikacją, co zostało omówione w podsekcjach 4.1.1 oraz 4.1.4. Profile zawierają również kryteria rekrutacyjne nadane przez administratora szkolnego. Są to przedmioty uwzględniane w procesie rekrutacji przy obliczaniu punktów zdobytych przez kandydata do profilu. Kryterium może być określone jako obowiązkowe, albo alternatywne, co odpowiada atrybutowi *type* w tabeli kryteriów (*profiles\_criteria*).

Tabela *kryteriów* łączy tabelę *przedmiotów* oraz profili relacją *wiele do wielu*.

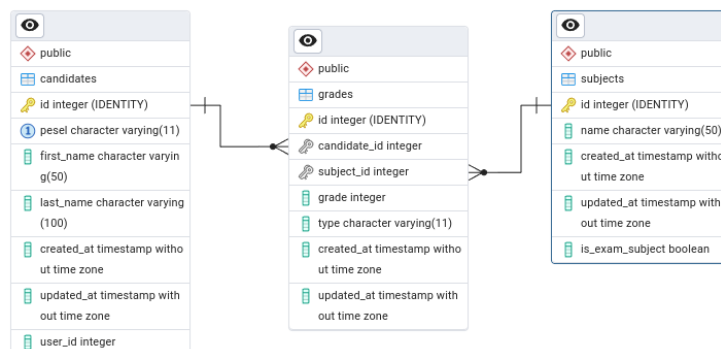
### 3.4.3. Encja ocen

Diagram encji ocen przedstawiony jest na rysunku 3.8.

Kandydat powiązany jest ze zbiorem ocen, które dostał z egzaminu oraz na świadectwie, z tego powodu tabela kandydatów (*candidates*) jest w relacji *jeden do wielu* z tabelą ocen (*grades*). Encja ocen przechowuje informacje o wartości oceny



Rysunek 3.7: Diagram encji profili



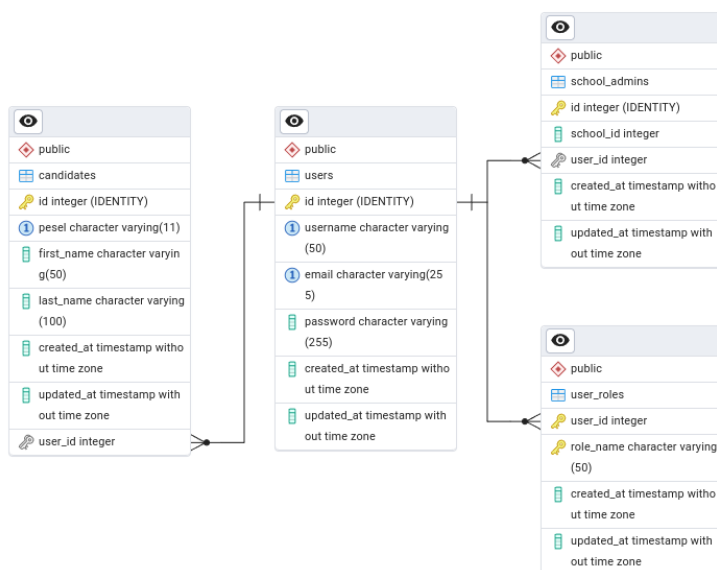
Rysunek 3.8: Diagram encji aplikacji

oraz o typie oceny, który informuje, czy uzyskana ocena jest z egzaminu czy ze świadectwa.

Ocena przypisana jest też do określonego przedmiotu, a z racji tego, że tabela ocen przechowuje oceny wszystkich kandydatów - ocen przypisanych do jednego przedmiotu może być wiele. Z tego względu ocena jest reprezentacją relacji *wiele do wielu* między tabelą kandydatów, a tabelą przedmiotów (*subjects*).

### 3.4.4. Encja użytkowników

Diagram encji użytkowników przedstawiony jest na rysunku 3.9.



Rysunek 3.9: Diagram encji użytkowników

Użytkownikiem systemu może być zwykły użytkownik, który ma nadaną rolę "user". Użytkownik ten jest prawnym opiekunem grupy kandydatów, w imieniu których składa wnioski. Z tego powodu tabela użytkowników (*users*) jest związana z tabelą kandydatów (*candidates*) relacją *jeden do wielu*. Encja kandydata zawiera w sobie dodatkowe informacje o kandydacie, takie jak imię, nazwisko oraz pesel.

W przypadku gdy użytkownik ma nadaną rolę "school\_admin", jest on administratorem szkolnym dla pewnej grupy szkół. Stąd wynika powiązanie użytkownika ze szkołami, które było już dokładniej opisywane w ramach opisywania encji szkół.

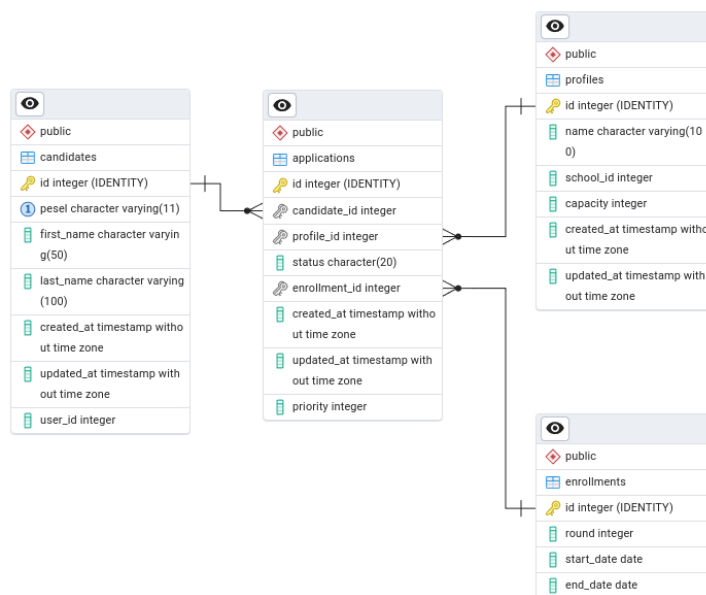
Role użytkownika przechowywane są w tabeli *user\_roles*. Tabela użytkowników jest w relacji *jeden do wielu* z tabelą ról użytkownika, w której zawarta jest nazwa roli.

Zarówno baza danych jak i aplikacja korzystająca z modelu autoryzacji opartej na rolach, uwzględnia możliwość istnienia wielu ról przypisanych do jednego użytkownika, w celu umożliwienia przyszłej rozbudowy systemu lub łatwiejszego wprowadzenia zmian w jego implementacji. Encja użytkownika zawiera nazwę użytkownika, adres email oraz zahaslowane hasło.



### 3.4.5. Encja aplikacji

Diagram encji aplikacji przedstawiony jest na rysunku 3.10.



Rysunek 3.10: Diagram encji aplikacji

Kandydat w danej turze wyborów składa aplikacje do profili, którymi jest zainteresowany. W związku z tym tabele kandydatów, profili oraz naborów (*enrollments*) powiązane są z tabelą aplikacji (*applications*) relacją *jeden do wielu*.

## 3.5. Statystyki kodu

Statystyki wygenerowane za pomocą narzędzia cloc [30] są zawarte na rysunku 3.11. Sumaryczna ilość kodu w projekcie przekracza 8000 linii.

Pokrycie kodu testami dla warstwy logiki biznesowej jest przedstawione na rysunku 3.11c, natomiast liczba testów zawarta jest na rysunku 3.11d. Zostało ono wygenerowane za pomocą narzędzia wbudowanego w środowisko programistyczne Webstorm [31]. Testy e2e uruchamiane są za pomocą komendy: `npx cypress run` wywołanej w katalogu `/client`. Przed uruchomieniem testów e2e należy uruchomić frontend oraz backend aplikacji.

Aby wywołać testy jednostkowe należy uruchomić komendę `npm test` w katalogu `/server`.

```

julia@konkuter:~/praca_inzynierska/electronic-school-enrollment-system/client$ cloc --exclu
de-dir=node_modules --exclude-lang=JSON,SVG,config.ts .
157 text files.
156 unique files.
8 files ignored.

github.com/AlDanial/cloc v 1.98 T=0.10 s (1575.6 files/s, 46197.0 lines/s)
-----
Language             files      blank      comment      code
-----
TypeScript            131         435          60         3271
SCSS                   12          34           0          301
CSS                     1           37           0          139
XML                     4            0           0           90
JavaScript             1            0           0           18
HTML                   1            0           0           13
-----
SUM:                   150         506          60         3832
  
```

(a) Statystyki dla warstwy frontendowej.

```

julia@konkuter:~/praca_inzynierska/electronic-school-enrollment-system/server$ c
loc --exclude-dir=node_modules,@types --exclude-lang=CSV,JSON .
115 text files.
114 unique files.
11 files ignored.

github.com/AlDanial/cloc v 1.98 T=0.09 s (1165.2 files/s, 70236.3 lines/s)
-----
Language             files      blank      comment      code
-----
TypeScript            104         922          506         4608
Text                    1            1            0          292
-----
SUM:                   105         923          506         4900
  
```

(b) Statystyki dla warstwy backendowej.

```

> routes
✓ services 9 files, 100% lines covered
  TS adminService.ts 100% lines covered
  TS applicationService.ts 100% lines covered
  TS candidateService.ts 100% lines covered
  TS enrollmentService.ts 100% lines covered
  TS gradeService.ts 100% lines covered
  TS profileService.ts 100% lines covered
  TS schoolService.ts 100% lines covered
  TS subjectService.ts 100% lines covered
  TS userService.ts 100% lines covered
✓ utils
  
```

(c) Pokrycie testami.

```

✓ should register a new user with hashed password
✓ should throw an error if login is already taken
✓ should throw an error if email is already taken
deleteUser
✓ should successfully delete user

81 passing (284ms)
  
```

(d) Liczba przypadków testowych

Rysunek 3.11

## Rozdział 4.

# Algorytmy i struktury danych

### 4.1. Algorytm Naboru

#### 4.1.1. Idea

1. Tworzenie list rankingowych dla profili:
  - Kandydaci są punktowani na podstawie ocen z egzaminu oraz ocen ze świadectwa z przedmiotów wymaganych przez dany oddział.
  - Dla każdego profilu:
    - (a) Kandydaci są sortowani pod względem uzyskanych punktów.
    - (b) Na podstawie liczby dostępnych miejsc dla profilu, tworzona jest lista kwalifikujących się kandydatów oraz lista rezerwowa.
2. Przydział kandydatów na podstawie priorytetów:
  - Kandydat pozostaje w profilu o najwyższym priorytecie spośród tych, do których się zakwalifikował, reszta aplikacji kandydata zostaje odrzucona.
  - Zwolnione miejsca są uzupełniane kandydatami z dalszych miejsc na listach.
3. Aktualizacja statusu aplikacji na bazie danych.

### 4.1.2. Pseudokod

Kod znajduje się w katalogu `/server/src/services/adminService.ts`.

---

**Algorithm 1** Algorytm Naboru

---

```
procedure nabór (lista przyjętych, listy rezerwowe)
    zaakceptowane    ▷ Mapa zaakceptowanych aplikacji dla kandydatów
    odrzucone        ▷ Tablica odrzuconych aplikacji
    do_rozpatrzenia = lista przyjętych

    while do_rozpatrzenia nie jest pusta do
        rozpatrywana = do_rozpatrzenia.pop()
        id_kandydata = do_rozpatrzenia.kandydat.id
        aktualna = zaakceptowane(id_kandydata)

        if aktualna istnieje oraz ma
            mniejszy priorytet niż rozpatrywana then
                do_odrzucenia = rozpatrywana
                do_zaakceptowania = aktualna
            else
                do_odrzucenia = aktualna
                do_zaakceptowania = rozpatrywana
            end if

        zaakceptowane.set(id_kandydata, do_zaakceptowania)
        if do_odrzucenia nie istnieje then continue
        end if

        odrzucone.push(do_odrzucenia)
        rezerwa = listy_rezerwowe.get(do_odrzucenia.id_profilu)
        nowa_aplikacja = rezerwa.shift()
        if nowa_aplikacja istnieje then
            do_rozpatrzenia.push(nowa_aplikacja)
        end if
    end while
    return zaakceptowane, odrzucone, listy_rezerwowe
end procedure
```

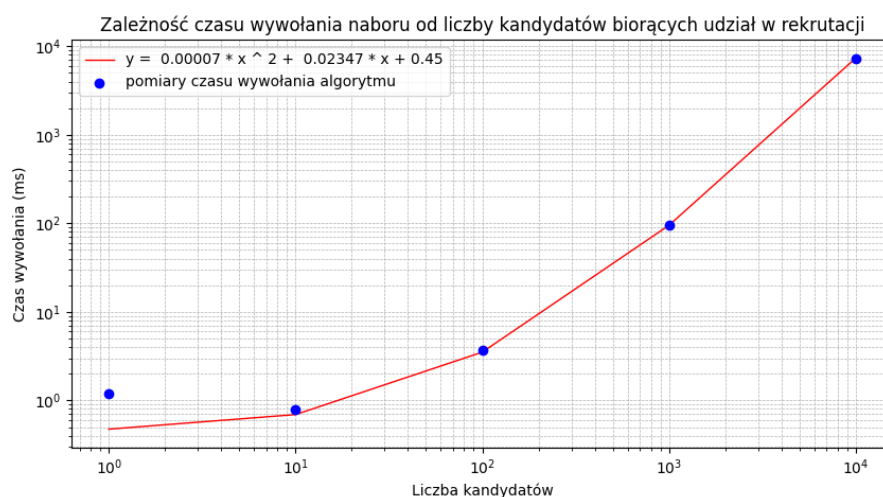
---

### 4.1.3. Oszacowanie złożoności

Wykres 4.1 przedstawia zależność między czasem wykonania algorytmu naboru, a liczbą kandydatów, przy założeniu stałej liczby profili wynoszącej 300, stałej liczby miejsc w profilach wynoszącej 30, oraz stałej liczby aplikacji składanych przez jednego kandydata, która wynosi 6.

Przedstawione pomiary nie uwzględniają czasu poświęconego na odczyt danych z bazy danych, który jest wykonywany na początku algorytmu w celu pozyskania kandydatów, aplikacji oraz profili.

Nie uwzględniono również czasu przeznaczonego na zaktualizowanie aplikacji w bazie danych po zakończonym naborze. Omówienie narzutu wynikającego z zapytań do bazy danych znajduje się w podsumowaniu algorytmu na wykresie 4.2.

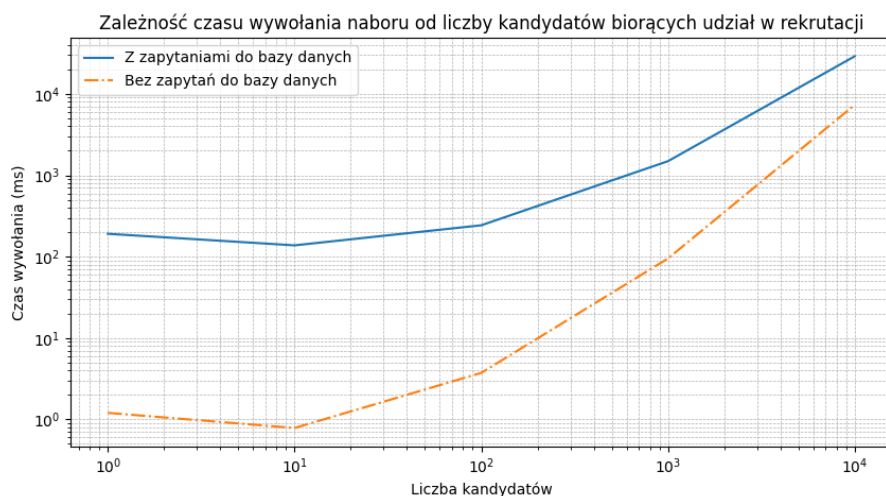


Rysunek 4.1: Wykres stworzony w Pythonie w bibliotece Matplotlib

Wykres wskazuje, że czas wykonania algorytmu można opisać za pomocą funkcji kwadratowej postaci  $a \cdot n^2 + b \cdot n + c$ , gdzie  $n$  oznacza liczbę kandydatów.

Oznacza to złożoność  $O(n^2)$ .

#### 4.1.4. Podsumowanie algorytmu



Rysunek 4.2: Wykres stworzony w Pythonie w bibliotece Matplotlib

Algorytm jest w stanie przeprowadzić nabór dla 1000 kandydatów w czasie 1.5 sekundy, a dla 10000 czas wywołania rośnie do 29 sekund.

Duży narzut czasowy spowodowany jest przez ilość zapytań do bazy danych, które wykonują się na początku algorytmu, aby uzyskać informacje o aplikacjach, kandydatach oraz profilach, a także na końcu algorytmu, gdzie aktualizowane są statusy aplikacji. Wpływ operacji wykonywanych na bazie danych można dostrzec na wykresie 4.2.

Jeżeli weźmiemy pod uwagę jedynie czas wywołania algorytmu, bez krańcowych operacji wykonywanych na bazie danych, otrzymamy czasy: 0.09 sekundy dla 1000 kandydatów oraz 7,3 sekundy dla 10000 kandydatów.

Algorytm ten w założeniu jest wykonywany kilka razy w roku, zatem z powodu niewielkiej częstotliwości uruchamiania algorytmu, nie jest konieczne jego maksymalne zoptymalizowanie. Szukanie optymalnego algorytmu nie było również celem tej pracy. Jest to ciekawy temat na osobny projekt. Do obecnego zastosowania, algorytm spełnia swoje zadanie.

Czas wykonania algorytmu mógłby zostać zmniejszony poprzez ograniczenie liczby zapytań do bazy danych oraz wprowadzenie indeksów do często używanych kolumn.

## Rozdział 5.

# Podsumowanie

### 5.1. Co udało się zrobić

Celem pracy było zaprojektowanie i implementacja systemu informatycznego automatyzującego proces rekrutacji do szkół ponadpodstawowych. W ramach projektu udało się osiągnąć następujące rezultaty:

- Została stworzona aplikacja umożliwiająca składanie wniosków rekrutacyjnych do wybranych oddziałów przez opiekunów kandydatów, łatwy dostęp do przeglądania statusów aplikacji, wgląd w ofertę z możliwością filtrowania i sortowania profili.
- Aplikacja udostępnia również interfejs dla administratorów szkół do zarządzania profilami i ich kryteriami, a także interfejs dla administratora systemu, który ma możliwość edycji szkół oraz jest odpowiedzialny za uruchamianie naboru.
- Rezultat naboru jest wyświetlany na widoku naboru dla administratora. Aplikacja umożliwia również eksport wszystkich, zaktualizowanych w wyniku naboru aplikacji, do pliku z rozszerzeniem .csv, dzięki czemu istnieje możliwość tworzenia prostego raportu z procesu rekrutacji.
- Udało się stworzyć algorytm naboru, przetestować go oraz oszacować jego złożoność.
- Udało się pokryć testami jednostkowymi 100% linii kodu w warstwie logiki biznesowej.
- Stworzono załączek testów E2E, który można łatwo rozszerzyć o więcej przypadków testowych.

## 5.2. Plany na rozwój aplikacji

Aplikacja w obecnym stanie dysponuje bardzo podstawową szatą graficzną. W przyszłości warto rozważyć interfejsu, który byłby atrakcyjny dla użytkowników. Należy przy tym uwzględnić przeznaczenie aplikacji, związane z edukacją, oraz zadbać o to, aby wygląd aplikacji odzwierciedlał jej cel.

Kolejnym krokiem z pewnością będzie rozszerzenie przypadków testowych, aby objąć nimi jak najszerszy zakres funkcjonalności aplikacji.

Następnie można rozszerzyć ilość oferowanych przez aplikację funkcjonalności. Warto wziąć pod uwagę stworzenie większej liczby raportów dla administratorów, być może zawierających statystyki.

Do systemu można również dodać funkcjonalność lokalizacji, co umożliwi filtrowanie szkół oraz sortowania ich pod względem odległości od miejsca lokalizacji kandydata.

Aby uwzględnić różnice w sytuacji, w jakiej znajdują się kandydaci, proces rekrutacji powinien również brać pod uwagę inne kryteria rekrutacyjne, takie jak czynniki socjalne - np. majątność rodziny, wielodzietność czy niepełnosprawność. Ważne jest rozbudowanie aplikacji o obsługę tych dodatkowych kryteriów.

Można byłoby wprowadzić blokadę zabezpieczającą system przed równoległym uruchomieniem dwóch lub więcej instancji algorytmu naboru jednocześnie.

Istotne jest również dodanie innych języków do aplikacji, co zwiększyłoby jej dostępność dla szerszej grupy użytkowników.



# Bibliografia

- [1] Strona główna Typescript, <https://www.typescriptlang.org/>, data dostępu: 03.01.2025
- [2] Strona główna Node.js, <https://nodejs.org/en>, data dostępu: 03.01.2025
- [3] Strona główna npm.js, <https://www.npmjs.com/>, data dostępu: 03.01.2025
- [4] Strona główna Express.js, <https://expressjs.com/>, data dostępu: 03.01.2025
- [5] Strona npm zawierająca opis biblioteki Bcrypt, <https://www.npmjs.com/package/bcrypt>, data dostępu: 03.01.2025
- [6] Strona główna React, <https://react.dev/>, data dostępu: 03.01.2025
- [7] Strona główna Vite, <https://vite.dev/>, data dostępu: 03.01.2025
- [8] Strona główna CSS, <https://vite.dev/>, data dostępu: 03.01.2025
- [9] Strona Sass zawierająca dokumentację SCSS, <https://sass-lang.com/documentation/syntax/>, data dostępu: 12.01.2025
- [10] Repozytorium projektu css-modules, <https://github.com/css-modules/css-modules>, data dostępu: 12.01.2025
- [11] Strona główna frameworku Mocha.js, <https://mochajs.org/>, data dostępu: 06.01.2025
- [12] Strona główna frameworku Sinon.js, <https://sinonjs.org/>, data dostępu: 06.01.2025
- [13] Strona Node.js z dokumentacją modułu Assert, <https://nodejs.org/api/assert.html>, data dostępu: 06.01.2025
- [14] Strona główna narzędzia Cypress, <https://www.cypress.io/>, data dostępu: 07.01.2025
- [15] Strona główna PostgreSQL, <https://www.postgresql.org.pl>, data dostępu: 03.01.2025

- [16] Strona główna systemu Git,  
<https://git-scm.com/>, data dostępu: 26.01.2025
- [17] Strona główna Github, <https://github.com/>, data dostępu: 26.01.2025
- [18] Strona główna języka Python, <https://www.python.org/>, data dostępu: 26.01.2025
- [19] Strona główna biblioteki Matplotlib, <https://matplotlib.org/>, data dostępu: 26.01.2025
- [20] Strona główna Visual Paradigm, <https://www.visual-paradigm.com/>, data dostępu: 18.01.2025
- [21] tytuł: Atomic Design, autor: Brad Frost, wydawca: Brad Frost, rok wydania: 01.01.2016  
Strona Brada Frosta zawierająca książkę:  
<https://atomicdesign.bradfrost.com/>, data dostępu: 21.01.2025
- [22] Strona React opisująca Feature-based Structure,  
<https://legacy.reactjs.org/docs/faq-structure.html>, data dostępu: 21.01.2025
- [23] Strona główna React Router, <https://reactrouter.com/home>, data dostępu: 18.01.2025
- [24] Strona React opisująca Context API, <https://legacy.reactjs.org/docs/context.html#when-to-use-context>, data dostępu: 18.01.2025
- [25] Strona Martina Fowlera opisująca wzorzec Model View Controller,  
<https://martinfowler.com/eaCatalog/modelViewController.html>,  
data dostępu: 18.01.2025
- [26] Strona Martina Fowlera opisująca wzorzec Repozytorium,  
<https://martinfowler.com/eaCatalog/repository.html>,  
data dostępu: 18.01.2025
- [27] Strona Martina Fowlera opisująca wzorzec warstwy serwisów,  
<https://martinfowler.com/eaCatalog/serviceLayer.html>,  
data dostępu: 18.01.2025
- [28] Strona główna modułu express-validator,  
<https://express-validator.github.io/docs/>,  
data dostępu: 20.01.2025
- [29] Strona główna narzędzia pgAdmin, <https://www.pgadmin.org/>, data dostępu: 18.01.2025
- [30] Repozytorium Github z narzędziem cloc, <https://github.com/AlDanial/cloc>, data dostępu: 20.01.2025

- [31] Strona główna Webstorm, <https://www.jetbrains.com/webstorm/>,  
data dostępu: 20.01.2025