



AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI,
INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ

Praca dyplomowa

System zarządzania biurem rachunkowym

Accounting office management system

Autor:

Mikołaj Kozieł

Kierunek studiów:

Informatyka

Specjalizacja:

Inżynieria Oprogramowania i Systemów

Opiekun pracy:

dr inż. Mirosław Gajer

Kraków 2021

Spis treści

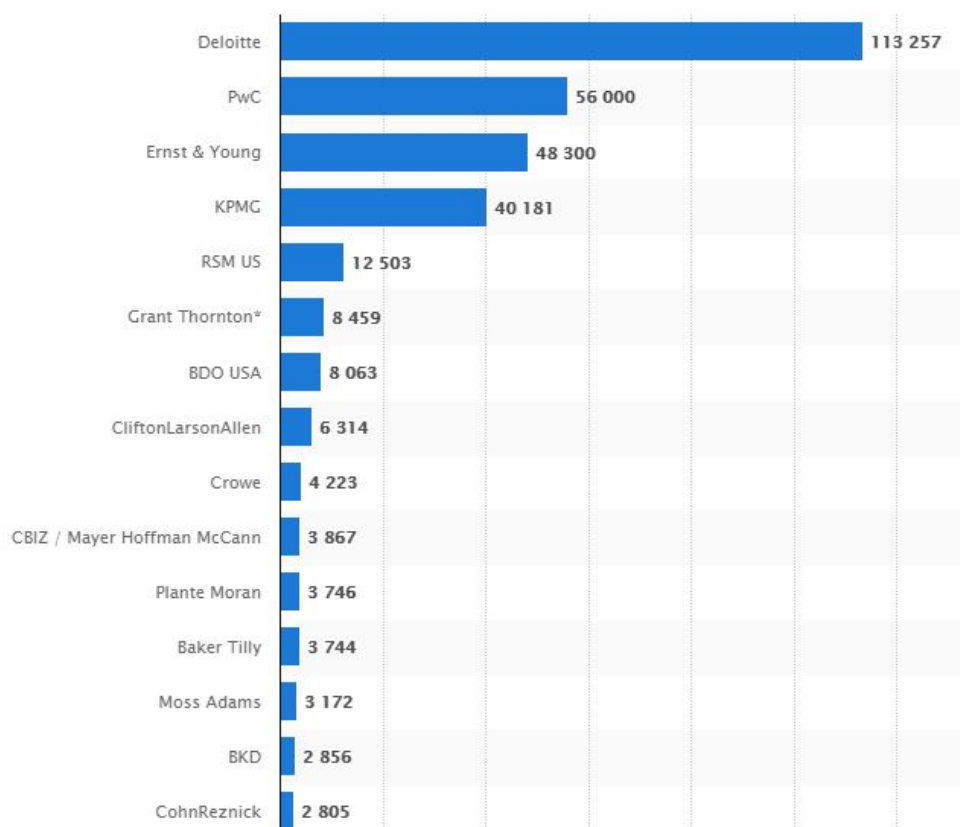
1	Cel prac i wizja produktu	3
1.1	Charakterystyka problemu	3
1.2	Wizja produktu	4
1.3	Analiza zagrożeń	5
2	Przegląd istniejących rozwiązań	6
2.1	Pliki tekstowe	6
2.2	BQE Core Suite	6
2.3	Karbon	7
2.4	Xero Practice Manager	8
2.5	Podsumowanie	9
3	Zakres funkcjonalności	11
3.1	Charakterystyka użytkowników systemu	11
3.2	Wymagania funkcjonalne	11
3.3	Wymagania нефункционалне	12
3.4	Ustalone priorytety	12
4	Wybrane aspekty realizacji	13
4.1	Studium wykonalności - analiza technologiczna	13
4.2	Architektura aplikacji	15
4.2.1	Serwer	15
4.2.2	Aplikacja frontendowa	16
4.2.3	Baza danych	16
4.2.4	System plików	17
4.3	Funkcjonalności	17
4.3.1	Rejestracja użytkownika	17
4.3.2	Autoryzacja i Autentykacja	17
4.3.3	Zarządzanie pracownikami	22
4.3.4	Zarządzanie pracą	23
4.3.5	Zarządzanie klientami	24
4.3.6	Kalendarz	24
4.3.7	Przesyłanie dokumentów	25
4.3.8	Zarządzanie profilem użytkownika	27
4.4	Baza danych	28
4.4.1	Połączenie z bazą	28
4.4.2	Schemat	28
4.4.3	Tabele	29

5	Kolejność prac	32
5.1	Milestone 1 - Projekt	33
5.2	Milestone 2 - Prototyp	33
5.3	Milestone 3 - Pełna funkcjonalność	34
5.4	Milestone 4 - Poprawki	34
6	Wyniki projektu	35
6.1	Wskazanie wyników projektu	35
6.2	Prezentacja wyników	35
6.2.1	Okna rejestracji	35
6.2.2	Okno logowania	38
6.2.3	Okno domowe zalogowanego użytkownika	38
6.2.4	Okno zarządzania pracą	40
6.2.5	Okno zarządzania pracownikami	40
6.2.6	Okno profilu podwładnego	41
6.2.7	Okno zarządzania klientami	42
6.2.8	Okno profilu klienta	42
6.2.9	Okno zarządzania firmami klienckimi	43
6.2.10	Okno profilu firmy klienckiej	43
6.2.11	Okno zarządzania profilem zalogowanego użytkownika	44
6.3	Propozycje dalszych prac	45
6.4	Wnioski	45
	Spis tabel	46
	Spis rysunków	47
	Bibliografia	49

1 Cel prac i wizja produktu

Rachunkowość to dziedzina mająca bardzo duże znaczenie w prowadzeniu biznesu. Każda firma musi prowadzić księgowość oraz zarządzać finansami. Wiele z nich stawia jednak na outsourcing usług finansowo-księgowych. Takie rozwiązanie jest bardziej korzystne niż prowadzenie rachunkowości samodzielnie.[1]

Rachunkowość to jedna z wielu branż, które rozwijają się w bardzo szybkim tempie. Przynosi ona ponad 110 miliardów dolarów dochodów w samych Stanach Zjednoczonych. 15 największych firm w USA zatrudnia ponad 300 tys. pracowników, natomiast ponad 46% całej branży rachunkowej to licencjonowani księgowi.[2]



Rysunek 1: Diagram pracowników największych firm rachunkowych w Stanach Zjednoczonych w 2020

Źródło: [3]

1.1 Charakterystyka problemu

Biura rachunkowe również potrzebują pomocy w zarządzaniu pracą. Wiele obowiązków związanych z ich działaniem mogłoby zostać ułatwione przy pomocy odpowiednich

narzędzi.

Ponad połowa (56%) księgowych przyznaje, że technologia zwiększa ich wydajność.

(Accounting Statistics & Facts[4])

Wynika z tego, że księgowi doceniają zasługi technologii w ich branży. Rozwój informatyki doprowadził wszakże do m.in. szybszego przekazywania informacji oraz łatwiejszego przechowywania danych. [5]

Takie obowiązki jak kontakt z klientami, zarządzanie ich danymi oraz zarządzanie pracownikami zabiera cenny czas, który mógłby być wykorzystany do innych zadań. A zakres działalności biura rachunkowego jest bardzo różnorodny. [6] Obejmuje wiele czynności, które są czasochłonne i wymagają precyzji, przez co księgowi nie mają wiele wolnego czasu na zarządzanie biurem. [7]

Niestety jednak na chwilę obecną najczęściej obowiązki tego typu są wykonywane przy użyciu programów, które zupełnie nie są odpowiednio na to przygotowane. Niektórymi z przykładów mogą być:

- dokumenty Microsoft Word
- arkusze Microsoft Excel
- pliki PDF

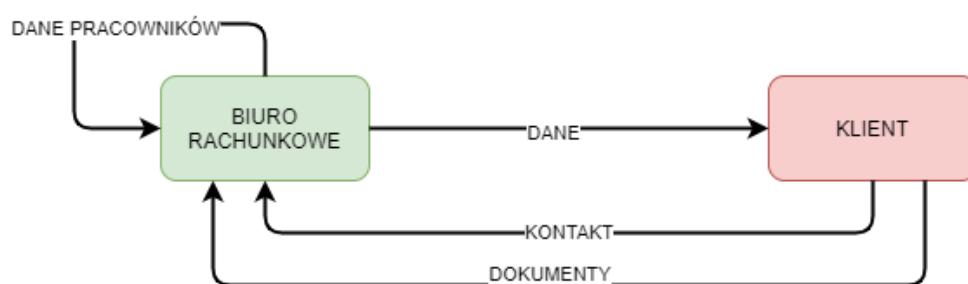
Wielu księgowych widzi braki w systemach i niedostosowanych do ich obowiązków programach, ale są zmuszeni do ich używania.

1.2 Wizja produktu

Celem pracy jest stworzenie projektu i implementacja systemu umożliwiającego zarządzanie biurem rachunkowym. System będzie miał za zadanie ułatwić prowadzenie takiej działalności. Czynności, takie jak m.in. zarządzanie danymi klienta i obiegiem dokumentów w firmie oraz kontakt z klientem, zostaną wykonane przez ten system.

Tabela 1: Analiza zagrożeń

Zagrożenie	Prawd.	Skutek
Problemy techniczne z pojemnością bazy danych	wysokie	3
Zagrożenie dla procesu - klient może często zmieniać zdanie	niskie	1
Zagrożenie dla produktu - serwer bazy danych przestanie działać	niskie	3
Wyciek danych	średnie	1
Zmiana polityki i przepisów dotyczących dostępu do bazy danych	niskie	3



Rysunek 2: Schemat koncepcyjny końcowego produktu

Źródło: Opracowanie własne

1.3 Analiza zagrożeń

Tworząc i udostępniając użytkownikom aplikację webową możliwe, że będziemy musieli się zmierzyć z poniższymi zagrożeniami:

Skutki: 3 - nie ma możliwości używania aplikacji, 2 - utrudnione używanie aplikacji, 1 - brak utrudnień w używaniu aplikacji

2 Przegląd istniejących rozwiązań

W tym rozdziale zostanie przeprowadzona analiza produktów już istniejących na rynku i porównanie ich z systemem finalnym tego projektu.

2.1 Pliki tekstowe

Wiele biur rachunkowych używa różnego rodzaju plików tekstowych w zarządzaniu pracą biurem rachunkowym. Najbardziej powszechne pliki tekstowe używane w tym celu to:

- Programy Microsoft Office(np. Microsoft Word, Microsoft Excel)
- pliki PDF
- Dokumenty i Arkusze Google

Pliki tekstowe są bardzo przydatne w prowadzeniu biura rachunkowego. Po odpowiednim przygotowaniu mogą służyć jako narzędzia do śledzenia czasu pracy pracowników, zarządzania nimi, zarządzaniu dokumentami klientów i wielu innych. Jednakże ich użycie często może powodować problemy m.in. z ich:

- przechowywaniem
- bezpieczeństwem
- konwersją
- przesyłaniem do odpowiednich podmiotów

Wymienione powyżej punkty to kluczowe sprawy w zarządzaniu biurem rachunkowym. Dlatego też powinna być zalecona zmiana narzędzi używanych na bardziej odpowiednie do tego zadania.

2.2 BQE Core Suite

BQE Core Suite jest to system do zarządzania biznesem.[8]



Rysunek 3: Logo BQE Core

Źródło: [9]

Ma wiele funkcjonalności przydatnych w tym celu, m.in.:

- Śledzenie czasu i wydatków
- Zarządzanie projektem
- Zarządzanie zasobami ludzkimi
- Generowanie danych do faktury
- Generowanie faktur e-mail z e-płatnościami

BQE Core Suite nie jest jednak narzędziem dedykowanym dla biur rachunkowych. Niektóre z jego funkcjonalności nie mają zastosowania w biurze rachunkowym. Inne jak np. przesyłanie dokumentów i kalendarz spotkań, które nadawałyby się w takim biurze, nie są dostępne w tym narzędziu.

Dlatego jest to narzędzie prawie idealne do zarządzania biurem rachunkowym. Jednakże jego przeznaczenie jest zbyt szerokie. Jak w każdym narzędziu niededykowanym dla danej branży, istnieje możliwość poprawy.

2.3 Karbon

Karbon to system do zarządzania pracą biura rachunkowego.[10]

Ma wiele funkcjonalności przydatnych w tym celu, m.in.:

- Zarządzanie pracownikami zdalnymi



Rysunek 4: Logo Karbon

Źródło: [11]

- Planowanie wydajności
- Standaryzowane procedury
- Zbieranie danych

Karbon jest narzędziem dedykowanym dla biur rachunkowych. Skupia on jednak największą uwagę na zarządzaniu pracą, przez co nie uwzględnia wielu funkcjonalności przydatnych w zarządzaniu biurem rachunkowym.

To narzędzie ma zbyt okrojone przeznaczenie, aby móc pokryć 100% zapotrzebowania biur rachunkowych.

2.4 Xero Practice Manager

Xero Practice Manager to oprogramowanie do zarządzania praktykami księgowymi, które usprawnia przepływy pracy i zwiększa wydajność.[12]

Ma wiele funkcjonalności przydatnych w tym celu, m.in.:

- Zarządzanie pracownikami
- Zarządzanie pracą pracowników
- Tworzenie spersalizowanych raportów



Rysunek 5: Logo Xero Practice manager

Źródło: [13]

- Zintegrowanie z produktami firmy Xero

Xero Practice Manager jest narzędziem bardzo rozbudowanym. Jest ono jednak skupione najbardziej na praktykach księgowych, a nie zarządzaniu biurem rachunkowym. Nie posiada wielu funkcjonalności, które są niezbędne do ułatwienia tego zadania, takich jak np.: przysyłania dokumentów oraz zarządzania zasobami ludzkimi.

To narzędzie, podobnie jak opisany w poprzednim podrozdziale Karbon, ma zbyt okrojone przeznaczenie i z tego powodu nie jest idealnym narzędziem do zarządzania biurem rachunkowym.

2.5 Podsumowanie

Każde z tych narzędzi jest wyspecjalizowane do pewnego obszaru działalności. Nie jest to jednak zarządzanie biurem rachunkowym. Z tego powodu zakres każdego z nich nie pokrywa się w 100% z wymaganiami takiej działalności.

Poniższa tabela przedstawia porównanie funkcjonalności rozwiązań, opisanych w tym rozdziale.

Tabela 2: Porównanie funkcjonalności porównywanych rozwiązań

Funkcjonalność	Pliki tekstowe	BQE Core Suite	Karbon	Xero Practice Manager
Śledzenie czasu pracy	?	✓	×	✓
Zarządzanie projektem	?	✓	×	×
Zarządzanie zasobami ludzkimi	?	✓	×	×
Dane do faktury	?	✓	×	×
Standaryzowane procedury	?	×	✓	×
Tworzenie spersalizowanych raportów	?	×	×	✓
Bezpieczeństwo	×	✓	✓	✓
GUI	×	✓	✓	✓
Przesyłanie dokumentów	×	×	×	×

Legenda: ✓ - funkcjonalność zawarta w narzędziu, × - funkcjonalność nie dostępna w narzędziu, ? - funkcjonalność dostępna po odpowiednim przygotowaniu narzędzia

Powyższa tabela jasno pokazuje, że żadne z tych narzędzi nie wspiera wszystkich funkcjonalności potrzebnych w zarządzaniu biurem rachunkowym. Nie są to więc narzędzia idealne do wykorzystania przy tym zadaniu. W takiej sytuacji zawsze istnieje możliwość poprawy i tym zajmie się projekt przygotowywany w ramach tej pracy.

3 Zakres funkcjonalności

Funkcjonalności tego systemu muszą spełniać szeroki wachlarz wymagań, narzucony przez różnej wielkości biura rachunkowe.

3.1 Charakterystyka użytkowników systemu

Użytkownicy naszego systemu będą się dzielić na dwie kategorie, które będą się dzielić na łącznie cztery podkategorie:

- Managerowie i pracownicy biura rachunkowego
 - USER - pracownik biura rachunkowego
 - ADMIN - manager w biurze rachunkowym
 - AO_ADMIN - administrator biura rachunkowego
- Pracownicy firm klienckich biur rachunkowych
 - CLIENT - pracownik z firmy będącej klientem biura rachunkowego

3.2 Wymagania funkcjonalne

W ramach naszego projektu zostaną zrealizowane następujące funkcjonalności:

1. Autoryzacja i autentykacja użytkowników - system do zarządzania użytkownikami i ich autoryzacji.
2. Zarządzanie danymi klientów - system do zarządzania danymi klientów, ich dokumentami oraz spotkaniami z nimi
3. Zarządzanie pracownikami - system do zarządzania pracownikami, podwładnymi oraz ich czasem pracy i przypisywaniem do nich klientów
4. Obieg dokumentów w firmie - system kontrolowania obiegu dokumentów między biurem rachunkowym i klientami
 - 4.1. Potwierdzenie przesyłania dokumentów między klientami a biurem rachunkowym

3.3 Wymagania niefunkcjonalne

Oprócz powyższych funkcjonalności nasza aplikacja powinna zapewniać:

- bezpieczeństwo
- przejrzysty i estetyczny interfejs
- intuicyjną obsługę
- reaktywność aplikacji frontendowej

3.4 Ustalone priorytety

Rozwój systemu został podzielony na cztery etapy. Poniższy plan jest poglądowy i możliwe, że ulegnie zmianie.

1. Etap 1

- Zaprojektowanie bazy danych i jej wstępna implementacja.
- Skonfigurowanie środowiska i projektu.
- Zaprojektowanie aplikacji frontendowej.
- Zaprojektowanie architektury serwera.

2. Etap 2

- Wstępne zaimplementowanie serwera.
- Zaimplementowanie autentykacji i autoryzacji.
- Połączenie serwera z bazą danych.
- Wstępne zaimplementowanie aplikacji frontendowej.

3. Etap 3

- Zaimplementowanie endpointów serwera.
- Zaimplementowanie poszczególnych stron aplikacji frontendowej.

4. Etap 4

- Poprawienie wyglądu ogólnego aplikacji.
- Ulepszenie funkcji serwera.

4 Wybrane aspekty realizacji

W poniższym rozdziale będą opisane wybrane elementy realizacji tego projektu. Zostaną zatem przedstawione tutaj informacje m.in. o architekturze całego systemu oraz jej poszczególnych komponentach, funkcje spełniane przez system oraz schemat i opis bazy danych użytej w tym projekcie.

4.1 Studium wykonalności - analiza technologiczna

Technologie, które zostały wybrane do realizacji projektu:

- Server: **Java**[14] + **Spring Boot** [15]

Wybór odpowiedniej technologii dla serwera to kluczowa sprawa dla tego projektu, ponieważ na niej jest oparty cały plan działania. Po przeanalizowaniu dostępnych opcji dwie technologie wydawały się najbardziej odpowiednie: Java i Node.js. Obie to popularne, dojrzałe technologie z ogromnymi środowiskami programistów wspierającymi ich rozwój.

Wybrałem jednak Javę wraz z jej frameworkiem Spring Boot. Razem zapewniają wszystko, czego potrzebuje ten projekt. [16]



Rysunek 6: Java

Źródło: [17]



Rysunek 7: Spring Boot

Źródło: [18]

- Frontend: **Angular** [19]

Patrząc na nowoczesne rozwiązania w aplikacjach webowych, doszedłem do wniosku, że najlepszym wyborem będzie framework Angular.

Jest to open-sourcowy framework utrzymywany przez Google. Jest prawie co roku ulepszany i rozwijany wraz z kolejnymi wersjami tego projektu. W tym momencie jest jednym z najpopularniejszych przedsięwzięć tego typu. [20]

Dwaj najpoważniejsi konkurenci Angulara to React JS i Vue.js. Jednak pierwszy z nich to biblioteka, a nie pełnoprawny MVC framework, jakim jest Angular i ponadto potrzebuje do działania wielu zewnętrznych bibliotek. Drugi zaś jest dość nowym frameworkiem, który wciąż jest rozwijany i ma również podobne wady jak React JS. [21]



Rysunek 8: Angular

Źródło: [22]

- Baza Danych: **MySQL** [23]

Baza danych dla tego projektu będzie posiadać bardzo skomplikowane relacje, dlatego też relacyjna baza danych wydaje się najodpowiedniejszym wyborem. Spośród wielu takowych wybrana została technologia MySQL. Jest to baza danych zajmująca pierwsze miejsce wśród najpopularniejszych w ankiecie portalu StackOverflow w 2020 roku. Ponadto jest ona bezpieczna i gwarantuje zgodność z regułami ACID. [24]



Rysunek 9: MySQL

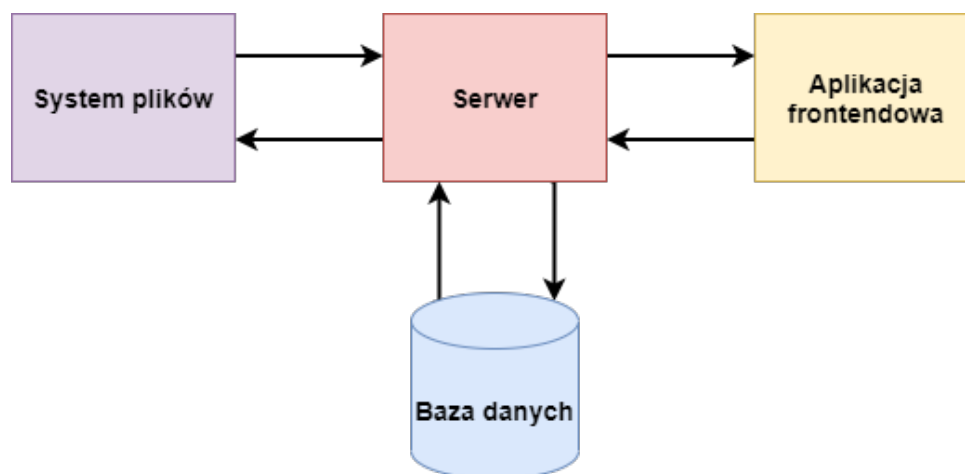
Źródło: [25]

4.2 Architektura aplikacji

Konstrukcja ogólna systemu będzie polegać na typowej architekturze typu serwer-klient. Można wydzielić cztery główne komponenty tego systemu:

1. Serwer
2. Aplikacja frontendowa
3. Baza danych
4. System plików

Relacje między nimi zostały przedstawione na poniższym diagramie.



Rysunek 10: Architektura

Źródło: Opracowanie własne

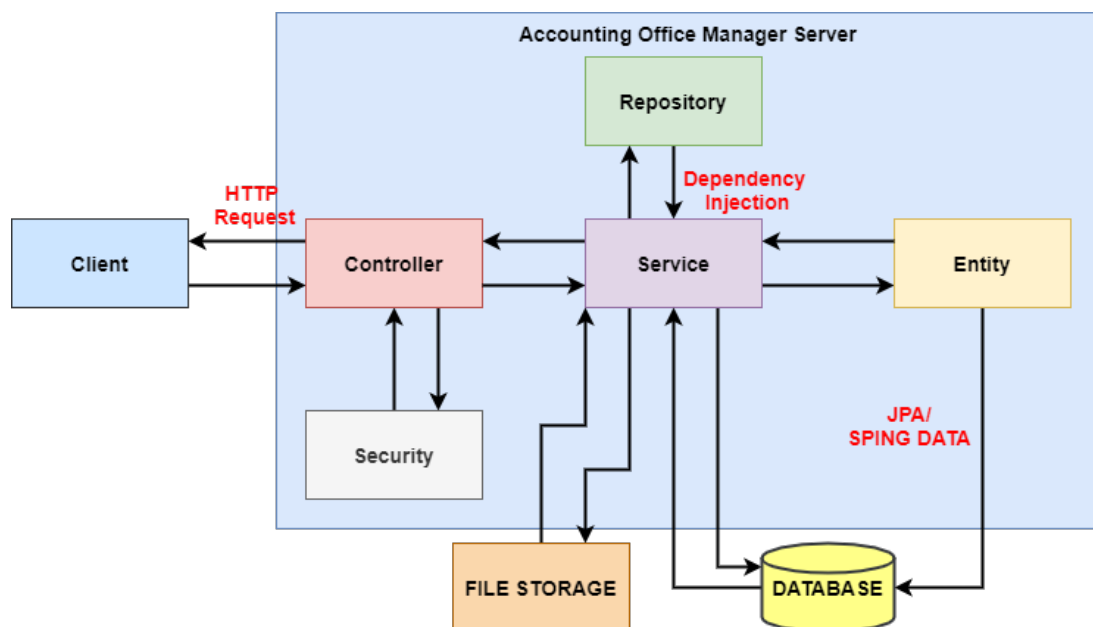
Jak widać na diagramie, głównym komponentem systemu będzie Serwer, który będzie zarządzał komunikacją między pozostałymi elementami.

4.2.1 Serwer

Serwer będzie przyjmować zapytania od aplikacji frontendowej, a następnie po zapytaniach do bazy danych lub systemu plików i przetworzeniu odpowiedzi od nich, wysyłać do niej odpowiedni Response.

Jego architektura będzie opierać się na frameworku Spring Boot, który będzie wykorzystywany w jego implementacji. Mamy zatem pięć modułów zapewniających bezpieczeństwo i komunikację z aplikacją frontendową, bazą danych i systemem plików.

1. Controller - przyjmuje i odpowiada na zapytania HTTP od Klienta (w tym przypadku aplikacji frontendowej) oraz komunikuje się z modułami Service i Security
2. Security - moduł odpowiadający za autoryzację użytkownika i autentykację zapytań HTTP
3. Service - w zakres jego zadań będzie odpowiadanie na zapytania do bazy danych i komunikację z systemem plików oraz za komunikację z modułami Repository i Entity
4. Entity - moduł zajmujący się reprezentacją danych z bazy danych
5. Repository - moduł odpowiedzialny za mapowanie do bazy danych, jest on wstrzykiwany do modułu Service



Rysunek 11: Architektura serwera

Źródło: Opracowanie własne

4.2.2 Aplikacja frontendowa

Budowa aplikacji frontendowej opiera się na architekturze technologii Angular.

4.2.3 Baza danych

Baza danych będzie posiadać konstrukcję typową dla baz danych MySQL. Więcej na ten temat zostanie opisanych w podrozdziale subsection 4.4 Baza danych.

4.2.4 System plików

System plików, wykorzystywany w tym projekcie, nie posiada większych wymagań architektonicznych. Jedynymi, które musi spełniać to: możliwość tworzenia katalogów oraz możliwość zapisywania i odczytywania plików.

4.3 Funkcjonalności

System posiada 8 funkcjonalności. Zostały one opisane w następujących podrozdziałach:

4.3.1 Rejestracja użytkownika

Rejestracja nowych użytkowników jest podzielona na trzy wersje:

1. Z publicznej wersji strony dostępnej dla każdego, można zarejestrować tylko i wyłącznie użytkownika, będącego administratorem biura rachunkowego, jednocześnie rejestrując biuro rachunkowe.
2. Druga wersja to rejestracja nowych pracowników biura rachunkowego, dostępna dla każdego admina biura rachunkowego. Tutaj może dodać nowych użytkowników, będących pracownikami biura rachunkowego, do którego należy admin.
3. Trzecia wersja to rejestracja firmy-klienta oraz jej pracowników. Ta opcja jest również dostępna dla każdego admina biura rachunkowego.

Podczas procesu rejestracji, trzeba wypełnić wszystkie pola w formularzu, a następnie zatwierdzić wprowadzone dane przez kliknięcie przycisku Submit.

4.3.2 Autoryzacja i Autentykacja

Autoryzacja i autentykacja została zaimplementowana przy użyciu biblioteki frameworku Spring Boot o nazwie spring-boot-starter-security po stronie serwera. Funkcjonalności te korzystają z pomocy JSON Web Tokens(JWT). Znajdują się one w większości w module SecurityConfig.

Autoryzacja Strona logowania użytkowników jest odpowiedzialna za autoryzację po stronie aplikacji frontendowej. Zawiera ona typowy formularz logowania, w którym użytkownik, chcący uzyskać dostęp do systemu, wpisuje do formularza swój login oraz hasło, a następnie potwierdza przyciskiem Submit.

Klasa AuthController jest odpowiedzialna po stronie serwera za otrzymywanie requestów autoryzacji. Zapytania do niej są mapowane poprzez adres "/auth". Zawiera ona tylko jedną metodę login. Proces autoryzacji jest wykonywany przez funkcję authenticate() obiektu klasy AuthenticationManager.

```
1  @PostMapping("login")
2  public ResponseEntity<User> login(@RequestBody @Valid User
   request) {
3      try {
4          Authentication authenticate = authenticationManager
5              .authenticate(
6                  new UsernamePasswordAuthenticationToken(
7                      request.getUsername(), request.
8                          getPassword()
9                  );
10
11         User user = (User) authenticate.getPrincipal();
12
13         HttpHeaders responseHeaders = new HttpHeaders();
14         responseHeaders.add(HttpHeaders.AUTHORIZATION,
15             jwtTokenUtil.generateAccessToken(user));
16         responseHeaders.add(HttpHeaders.
17             ACCESS_CONTROL_EXPOSE_HEADERS,
18             HttpHeaders.AUTHORIZATION);
19
20         return ResponseEntity.ok()
21             .headers(responseHeaders)
22             .body(user);
23     } catch (BadCredentialsException ex) {
24         return ResponseEntity.status(HttpStatus.UNAUTHORIZED).
25             build();
26     }
27 }
```

Widać tutaj również użycie klasy Principal, w celu uzyskania informacji zwrotnych od

metody autoryzującej. Podczas procesu autoryzacji generowany jest JWT Token, używany później w procesie autentykacji. Służy do tego funkcja `generateAccessToken()` z klasy `JwtTokenUtil`. Używany do tego jest builder `Jwts` z biblioteki `io.jsonwebtoken`.

```
1 public String generateAccessToken(User user) {
2     final String jwtIssuer = "ao.com";
3     return Jwts.builder()
4         .setSubject(format("%s,%s", user.getUser_id(), user.
5             getUsername()))
6         .setIssuer(jwtIssuer)
7         .setIssuedAt(new Date())
8         .setExpiration(new Date(System.currentTimeMillis() +
9             5 * 60 * 60 * 1000)) // 5 hours
10        .signWith(SignatureAlgorithm.HS512, jwtSecret)
11        .compact();
12 }
```

Autentykacja Guard `AuthGuard` jest odpowiedzialny po stronie aplikacji frontendowej za autentykację. Sprawdza on stan zalogowania aktualnego użytkownika. W przypadku osoby niezalogowanej przekierowuje ją na stronę do logowania.

Po stronie serwera zaś za autentykację odpowiedzialna jest klasa `JwtTokenFilter`. Jest ona wywoływana po otrzymaniu każdego requestu a przed wywołaniem funkcji mapującej użyty w nim endpoint. W funkcji `configure()` klasy `SecurityConfig` można zobaczyć wszystkie reguły dostępności endpointów systemu oraz wywołanie przechwytywania http requestów do serwera przez `JwtTokenFilter`:

```
1 // Permissions on endpoints
2 http.authorizeRequests()
3     // Public endpoints
4     .antMatchers("/auth/**").permitAll()
5     // Private endpoints
6     .antMatchers("/ao/").hasAnyRole(
7         RoleEnum.AO_ADMIN.toString())
8     .antMatchers("/calendar/").hasAnyRole(
9         RoleEnum.ADMIN.toString(),
10        RoleEnum.AO_ADMIN.toString(),
11        RoleEnum.USER.toString(),
12        RoleEnum.CLIENT.toString())
13     .antMatchers("/cc/register").hasAnyRole(
14        RoleEnum.ADMIN.toString(),
```

```

15         RoleEnum.AO_ADMIN.toString())
16     .antMatchers("/cc/").hasAnyRole(
17         RoleEnum.ADMIN.toString(),
18         RoleEnum.AO_ADMIN.toString(),
19         RoleEnum.USER.toString(),
20         RoleEnum.CLIENT.toString())
21     .antMatchers("/client/").hasAnyRole(
22         RoleEnum.ADMIN.toString(),
23         RoleEnum.AO_ADMIN.toString(),
24         RoleEnum.USER.toString())
25     .antMatchers("/company/").hasAnyRole(
26         RoleEnum.ADMIN.toString(),
27         RoleEnum.AO_ADMIN.toString(),
28         RoleEnum.USER.toString(),
29         RoleEnum.CLIENT.toString())
30     .antMatchers("/documents/").hasAnyRole(
31         RoleEnum.ADMIN.toString(),
32         RoleEnum.AO_ADMIN.toString(),
33         RoleEnum.USER.toString(),
34         RoleEnum.CLIENT.toString())
35     .antMatchers("/employee/").hasAnyRole(
36         RoleEnum.ADMIN.toString(),
37         RoleEnum.AO_ADMIN.toString(),
38         RoleEnum.USER.toString(),
39         RoleEnum.CLIENT.toString())
40     .antMatchers("/user/").hasAnyRole(
41         RoleEnum.ADMIN.toString(),
42         RoleEnum.AO_ADMIN.toString(),
43         RoleEnum.USER.toString(),
44         RoleEnum.CLIENT.toString())
45     .antMatchers("/work-log/").hasAnyRole(
46         RoleEnum.ADMIN.toString(),
47         RoleEnum.AO_ADMIN.toString(),
48         RoleEnum.USER.toString())
49     .anyRequest().authenticated();
50
51 // Add JWT token filter
52 http.addFilterBefore(
53     jwtTokenFilter,
54     UsernamePasswordAuthenticationFilter.class
55 );

```

Klasa `JwtTokenFilter` rozszerza klasę `OncePerRequestFilter`. W związku z tym nadpisuje ona funkcję `doFilterInternal()`. Ta funkcja zawiera dwie części:

- autentykacja tokena JWT - sprawdzenie czy request zawiera token oraz jego walidacja:

```
1  @Override
2  protected void doFilterInternal(HttpServletRequest request,
3                                HttpServletResponse response,
4                                FilterChain chain) throws
5                                ServletException, IOException {
6      // Get authorization header and validate
7      final String header = request.getHeader(HttpHeaders.
8          AUTHORIZATION);
9      if (isEmpty(header) || !header.startsWith("Bearer ")) {
10         chain.doFilter(request, response);
11         return;
12     }
13
14     // Get jwt token and validate
15     final String token = header.split(" ")[1].trim();
16     if (!jwtTokenUtil.validate(token)) {
17         chain.doFilter(request, response);
18         return;
19     }
20     ...
21 }
```

- autentykacja użytkownika - pobranie informacji na temat właściciela tokenu JWT z bazy danych oraz autentykacja jego i jego roli w systemie:

```
1  ...
2  // Get user identity and set it on the spring security
3  context
4  User userDetails = null;
5  try {
6      userDetails = userRepo.findByUsername(jwtTokenUtil.
7          getUsername(token));
8  } catch (Exception ignored){
9  }
```

```

8
9     UsernamePasswordAuthenticationToken authentication = new
        UsernamePasswordAuthenticationToken(
10         userDetails, null,
11         ofNullable(userDetails).map(User::getAuthorities).
            orElse(of())
12     );
13
14     authentication
15         .setDetails(new WebAuthenticationDetailsSource().
            buildDetails(request));
16
17     SecurityContextHolder.getContext().setAuthentication(
        authentication);
18     chain.doFilter(request, response);
19 }

```

4.3.3 Zarządzanie pracownikami

Jedną z funkcjonalności naszego systemu jest zarządzanie pracownikami biura rachunkowego. Po stronie aplikacji frontendowej dwa komponenty są odpowiedzialne za tą funkcję:

- Employees Management Component - zawiera informacje na temat wszystkich pracowników, podległych aktualnie zalogowanemu użytkownikowi. Jego funkcja inicjalizująca używa endpointu /employee/admin/:id, gdzie id oznacza user_id aktualnie zalogowanego użytkownika:

```

1  currentUser: User;
2  employees: Array<Employee>;
3
4  ngOnInit(): void {
5      this.userService.getCurrentUser().subscribe(user =>{
6          this.currentUser = user;
7          this.eService.getEmployeesForAdmin(this.currentUser.id)
            .subscribe(response=>{
8              this.employees = response;
9              this.spinnerFlag += 1;
10          })
11      })

```

12 | }

- Employee Info Component - zawiera informacje na temat konkretnego pracownika, jego podwładnych i klientów. Zawiera również możliwość:

- zmiany roli danego pracownika - endpoint `/user/updateRole/id`

```
1  changeRole(){
2      this.userService.changeRole({
3          user_id: this.employee.id,
4          role_id: this.selectedRole
5      }).subscribe(x=>{
6          refreshComponent(this.router);
7      })
8  }
```

- przypisywania mu klientów - endpoint `/client/employee/`

```
1  assignClient(){
2      this.userService.assignClient({
3          employee_id: this.employee.id,
4          client_id: this.selectedClient
5      }).subscribe(x=>{
6          refreshComponent(this.router);
7      })
8  }
```

Komponenty te są dostępne tylko dla adminów biura rachunkowego.

4.3.4 Zarządzanie pracą

W naszym systemie będzie możliwość również zarządzania czasem pracy pracowników biura rachunkowego. Strona tej funkcjonalności będzie zatem dostępna tylko dla użytkowników będących pracownikami lub adminami biura rachunkowego.

Okno tej funkcji jest podzielone na dwie części. W pierwszej części znajduje się licznik czasu pracy wraz z przyciskami na wystartowanie, wstrzymanie i zakończenie liczenia.

```
1  startTimer() {
2      this.date = new Date();
3      this.interval = setInterval(() => {
```



```

4      if (this.time === 0) {
5          this.time++;
6      } else {
7          this.time++;
8      }
9      this.display=this.transform(this.time)
10 }, 1000);
11 }

```

W drugiej części zaś znajduje się lista do tej pory zarejestrowanego czasu pracy aktualnie zalogowanego pracownika.

4.3.5 Zarządzanie klientami

Zarządzanie klientami to możliwość dostępna dla wszystkich pracowników biura rachunkowego. Składa się ona z 2 komponentów:

1. Client Management Component - lista wszystkich klientów z ich podstawowymi danymi aktualnie zalogowanego pracownika
2. Client Info Component - podstrona wymienionej wyżej cechy zawierająca informacje szczegółowe klienta oraz listę dokumentów przesłanych przez niego do biura rachunkowego

4.3.6 Kalendarz

Funkcjonalność ta dostępna jest dla każdego z użytkowników. Strona ją reprezentująca zawiera kalendarz w widoku miesięcznym lub tygodniowym z wydarzeniami przypisanymi dla aktualnie zalogowanego użytkownika. Można więc tutaj zapisać daty spotkań między pracownikiem biura rachunkowego, jak i przypomnienia terminów składnia poszczególnych dokumentów.

Po stronie aplikacji frontendowej używa biblioteki angular-calendar. Wybór między widokiem miesiąca i tygodnia wygląda następująco w kodzie html:

```

1 <div [ngSwitch]="view">
2   <mw1-calendar-month-view
3     *ngSwitchCase="CalendarView.Month"
4     [viewDate]="viewDate"

```

```

5      [events]="events"
6      [refresh]="refresh"
7      [activeDayIsOpen]="activeDayIsOpen"
8      class="calendar_own"
9  >
10 </mwl-calendar-month-view>
11 <mwl-calendar-week-view
12     *ngSwitchCase="CalendarView.Week"
13     [viewDate]="viewDate"
14     [events]="events"
15     [dayEndHour]="dayEndHour"
16     [dayStartHour]="dayStartHour"
17     [hourSegments]="4"
18     [hourSegmentHeight]="15"
19     class="calendar_own"
20 >
21 </mwl-calendar-week-view>
22 </div>

```

4.3.7 Przesyłanie dokumentów

Jednym z głównych zadań tego systemu jest przesyłanie dokumentów między klientami a biurem rachunkowym oraz przechowywanie ich w systemie plików serwera. Plik jest przesyłany od aplikacji frontendowej do serwera jako multipart/form-data. Jednym z argumentów jest plik typu MultipartFile oraz informacje na jego temat jako obiekt klasy Document.

Plik jest zapisywany w systemie plików przy pomocy funkcji store() przyjmującej jako argumenty:

- MultipartFile file - plik do zapisania
- Document document - obiekt zawierający informacje na temat pliku do zapisania
- String modifiedName - zmodyfikowana nazwa pliku zawierająca między innymi informacje na temat daty zapisu pliku

Funkcja ta została przedstawiona poniżej:

```

1 public String store(MultipartFile file, Document document, String
    modifiedFileName) {

```

```

2      try {
3          if (file.isEmpty()) {
4              throw new StorageException("Failed to store empty
5                  file.");
6          }
7          Path destinationFile = this.rootLocation.resolve(
8              Paths.get(Objects.requireNonNull(modifiedFileName
9                  )))
10             .normalize().toAbsolutePath();
11         if (!destinationFile.getParent().equals(this.rootLocation
12             .toAbsolutePath())) {
13             throw new StorageException("Cannot store file outside
14                 current directory.");
15         }
16         try (InputStream inputStream = file.getInputStream()) {
17             Files.copy(inputStream, destinationFile,
18                 StandardCopyOption.REPLACE_EXISTING);
19         }
20         this.setFileMetaData(destinationFile, document);
21         return destinationFile.toString();
22     }
23     catch (IOException e) {
24         throw new StorageException("Failed to store file.", e);
25     }
26 }

```

Potwierdzenie przesyłu dokumentów Funkcja store() zawiera wywołania kilku innych, m.in. setFileMetaData(). Jest to funkcja, która dopisuje metadane do plików w formacie pdf, potwierdzających przekazanie dokumentu do biura rachunkowego. Metadane dopisywane przez nią to:

- transfer-date - data przekazania dokumentu do biura rachunkowego
- transfer-client - user_id użytkownika przesyłającego dany dokument
- transfer-company - company_id firmy do której oryginalnie należał dokument

Do dodawania metadanych do plików pdf została wykorzystana biblioteka Apache PDFBox:

```

1 public void setFileMetaData(Path path, Document document) throws
    IOException {

```

```
2     PDDocument pdfDoc = PDDocument.load(path.toFile());
3     pdfDoc.getDocumentInformation().setCustomMetadataValue("
        transfer-date", new Date().toString());
4     pdfDoc.getDocumentInformation().setCustomMetadataValue("
        transfer-client", String.valueOf(document.getClient().
            getUser_id()));
5     pdfDoc.getDocumentInformation().setCustomMetadataValue("
        transfer-company", String.valueOf(document.getCompany().
            getCompany_id()));
6     pdfDoc.save(path.toFile());
7 }
```

4.3.8 Zarządzanie profilem użytkownika

Zarządzanie profilem użytkownika jest dostępne ze strony startowej użytkownika po kliknięciu odpowiedniego kafelka. Po wejściu na stronę profilu widzimy informacje o aktualnie zalogowanym użytkowniku. Znajduje się tu również rozwijany panel, w którym po wpisaniu starego hasła i dwukrotnie nowego, istnieje możliwość zmiany hasła.

4.4.3 Tabele

- **User** przechowuje informacje o użytkownikach

Tabela 3: Struktura tabeli User

Nazwa	Typ	Null	Klucz główny	Klucz obcy
user_id	int(11)	Nie	Tak	Nie
first_name	varchar(50)	Nie	Nie	Nie
last_name	varchar(50)	Nie	Nie	Nie
username	varchar(50)	Nie	Nie	Nie
company_id	int(11)	Nie	Nie	Tak
password	varchar(50)	Nie	Nie	Nie

- **Employee** przechowuje dodatkowe informacje o użytkownikach będących pracownikami biur rachunkowych

Tabela 4: Struktura tabeli Employee

Nazwa	Typ	Null	Klucz główny	Klucz obcy
user_id	int(11)	Nie	Nie	Tak
admin_id	int(11)	Nie	Nie	Tak

- **Client** przechowuje dodatkowe informacje o użytkownikach będących pracownikami firm klienckich biur rachunkowych

Tabela 5: Struktura tabeli Client

Nazwa	Typ	Null	Klucz główny	Klucz obcy
user_id	int(11)	Nie	Nie	Tak
employee_id	int(11)	Nie	Nie	Tak

- **Roles** przechowuje informacje o rolach użytkowników systemu

Tabela 6: Struktura tabeli Roles

Nazwa	Typ	Null	Klucz główny	Klucz obcy
role_id	int(11)	Nie	Tak	Nie
name	varchar(10)	Nie	Nie	Nie

- **UserRole** tabela łącząca tabele User i Roles

Tabela 7: Struktura tabeli UserRole

Nazwa	Typ	Null	Klucz główny	Klucz obcy
user_id	int(11)	Nie	Nie	Tak
role_id	int(11)	Nie	Nie	Tak

- **Company** przechowuje informacje o firmach użytkowników

Tabela 8: Struktura tabeli Company

Nazwa	Typ	Null	Klucz główny	Klucz obcy
company_id	int(11)	Nie	Tak	Nie
name	varchar(50)	Nie	Nie	Nie

- **AccountingOffice** przechowuje dodatkowe informacje o firmach będących biurami rachunkowymi

Tabela 9: Struktura tabeli AccountingOffice

Nazwa	Typ	Null	Klucz główny	Klucz obcy
company_id	int(11)	Nie	Nie	Tak

- **ClientCompany** przechowuje dodatkowe informacje o firmach będących firmami klienckimi biur rachunkowych

Tabela 10: Struktura tabeli ClientCompany

Nazwa	Typ	Null	Klucz główny	Klucz obcy
company_id	int(11)	Nie	Nie	Tak
ao_id	int(11)	Nie	Nie	Tak

- **Documents** przechowuje informacje o dokumentach przesłanych przez użytkowników systemu

Tabela 11: Struktura tabeli Documents

Nazwa	Typ	Null	Klucz główny	Klucz obcy
document_id	int(11)	Nie	Tak	Nie
company_id	int(11)	Nie	Nie	Tak
client_id	int(11)	Nie	Nie	Tak
name	varchar(250)	Nie	Nie	Nie
path	varchar(250)	Nie	Nie	Nie
description	varchar(250)	Nie	Nie	Nie

- **WorkLog** przechowuje informacje o dzienniku pracy użytkowników będących pracownikami biur rachunkowych

Tabela 12: Struktura tabeli WorkLog

Nazwa	Typ	Null	Klucz główny	Klucz obcy
worklog_id	int(11)	Nie	Tak	Nie
user_id	int(11)	Nie	Nie	Tak
date	datetime	Nie	Nie	Nie
duration	bigint(11)	Nie	Nie	Nie

- **Calendar** przechowuje informacje o eventach w kalendarzach użytkowników

Tabela 13: Struktura tabeli Calendar

Nazwa	Typ	Null	Klucz główny	Klucz obcy
calendar_id	int(11)	Nie	Tak	Nie
start_date	datetime	Nie	Nie	Nie
end_date	datetime	Nie	Nie	Nie
title	varchar(250)	Nie	Nie	Nie
all_day	bit(1)	Nie	Nie	Nie

- **CalendarUser** tabela łącząca tabele User i Calendar

Tabela 14: Struktura tabeli CalendarUser

Nazwa	Typ	Null	Klucz główny	Klucz obcy
calendar_id	int(11)	Nie	Nie	Tak
user_id	int(11)	Nie	Nie	Tak

5 Kolejność prac

Narzędzia wykorzystane w zarządzaniu projektem:

- GitHub [27]
- Sourcetree [28]
- Trello [29]



Rysunek 13: GitHub

Źródło: [30]



Rysunek 14: Trello

Źródło: [31]

Ponadto została wykorzystana praktyka podziału harmonogramu na milestoney i sprinty, aby usprawnić proces powstawania systemu.

Zgodnie z początkowymi założeniami istnieją cztery milestoney: Projekt, Prototyp, Pełna Funkcjonalność oraz Poprawki. Każdy z ich zawiera różną ilość sprintów.

Sprinty w tym projekcie miały długość 1 tygodnia lub 2 tygodni w zależności od skomplikowania przyjętego na dany okres zadania. Udało się utrzymać tę zasadę do końca wykonywania systemu.

5.1 Milestone 1 - Projekt

Cel: Zaplanowanie przyszłych prac, stworzenie wizji projektu

Czas trwania: 6 tygodni (01 luty - 14 marca)

Kolejne sprinty oraz ich cele w tym milestone zostały opisane poniżej:

- **Sprint 1** - Analiza tematu projektu oraz wymagań narzucanych przez niego. Planowanie przyszłych prac oraz kolejności ich wykonywania.
- **Sprint 2** - Projektowanie architektury całkowitej systemu oraz poszczególnych komponentów. Analiza technologiczna i wybór odpowiednich technologii.
- **Sprint 3** - Projekt bazy danych i jej wstępna implementacja.
- **Sprint 4** - Konfiguracja środowiska. Konfiguracja projektów aplikacji frontendowej oraz serwera. Inicjalizacja repozytoriów w systemie GitHub.

5.2 Milestone 2 - Prototyp

Cel: Stworzenie działającego prototypu aplikacji.

Czas trwania: 8 tygodni (15 marca - 9 maja)

Kolejne sprinty oraz ich cele w tym milestone zostały opisane poniżej:

- **Sprint 5** - Wstępna implementacja serwera. Zapoznanie się z frameworkiem Spring Boot.
- **Sprint 6** - Wstępna implementacja aplikacji frontendowej. Inicjalizacja potrzebnych modułów i komponentów.
- **Sprint 7** - Wstępna implementacja połączenia z bazą danych.
- **Sprint 8** - Implementacja procesu autoryzacji użytkowników przy pomocy frameworku Spring Boot.
- **Sprint 9** - Implementacja procesu autentykacji użytkowników przy pomocy frameworku Spring Boot.

5.3 Milestone 3 - Pełna funkcjonalność

Cel: Dostarczenie w pełni funkcjonalnej aplikacji

Czas trwania: 12 tygodnie (10 maja - 1 sierpnia)

Kolejne sprinty oraz ich cele w tym milestone zostały opisane poniżej:

- **Sprint 10** - Implementacja endpointów dla modułu user, employee i client. Implementacja Work Management Component i Employee Component.
- **Sprint 11** - Implementacja endpointów dla modułu company, accounting office i client_company. Implementacja Companies Management Component i User Profile Component.
- **Sprint 12** - Implementacja endpointów dla modułu roles. Implementacja Client Management Component.
- **Sprint 13** - Implementacja endpointów dla modułu worklog. Implementacja WorkLog Component.
- **Sprint 14** - Implementacja endpointów dla modułu documents. Implementacja Documents Component.
- **Sprint 15** - Implementacja endpointów dla modułu calendar. Implementacja Calendar Component.
- **Sprint 16** - Implementacja Register Component, Register CC Component i Register Client Component.
- **Sprint 17** - Implementacja Add Client Component, Add Event Component i Add Employee Component.

5.4 Milestone 4 - Poprawki

Cel: Poprawienie ogólnego wyglądu aplikacji

Czas trwania: 4 tygodnie (2 sierpnia - 30 sierpnia)

Kolejne sprinty oraz ich cele w tym milestone zostały opisane poniżej:

- **Sprint 18** - Optymalizacja aplikacji frontendowej.
- **Sprint 19** - Praca nad dokumentacją.

6 Wyniki projektu

Rozdział ten zawiera wyniki naszego projektu oraz możliwe dalsze ścieżki jego rozwoju.

6.1 Wskazanie wyników projektu

Wynikiem tego projektu jest System Zarządzania Biurem Rachunkowym. Składa się on z trzech współpracujących ze sobą części:

- **Aplikacja frontendowa** - <https://github.com/mikkoziel/AccountingOfficeManager>, repozytorium zawiera gotową do skompilowania aplikację frontendową. Wymaga zainstalowanego Angulara oraz managera pakietów npm.
- **Serwer** - <https://github.com/mikkoziel/AccountingOfficeManagerServer>, repozytorium zawiera gotowy do skompilowania serwer. Wymaga zainstalowanej Javy w wersji 11 oraz narzędzie Maven.
- **Baza danych** - <https://github.com/mikkoziel/AccountingOfficeManagerDB>, repozytorium zawiera dwa skrypty sql do zaimportowania do bazy danych:
 - zawiera komendy potrzebne do inicjalizacji wszystkich tabel i niezbędnych danych
 - dodatkowo zawiera komendy do inicjalizacji przykładowych danych

6.2 Prezentacja wyników

Ostateczny wygląd aplikacji składa się z następujących okien:

6.2.1 Okna rejestracji

Zgodnie z założeniami, w systemie są cztery komponenty odpowiadające za rejestrację różnych aktorów systemu:

1. biura rachunkowego i jego AO_ADMINA
2. użytkowników typu USER będących pracownikami biura rachunkowego
3. firmy klienckiej

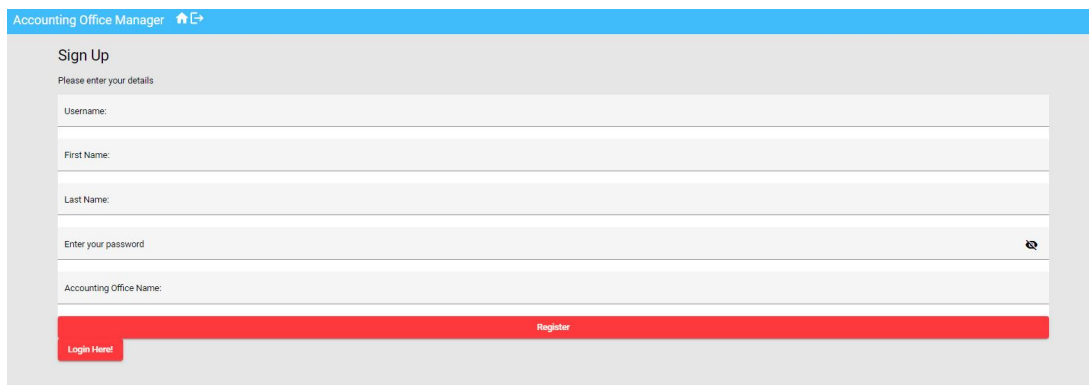
4. pracowników firmy klienckiej

Każdy z tych komponentów ma odpowiadającą mu stronę lub okno. Każde z nich zawiera formularz do wypełnienia oraz dwa przyciski:

- do zatwierdzenia procesu rejestracji
- do anulacji procesu rejestracji lub do przejścia do procesu logowania

Okna te wyglądają następująco:

1. Strona rejestracji biura rachunkowego i jego administratora- strona zawiera formularz z 5 polami oraz przyciski do zatwierdzenia procesu rejestracji i drugi do przejścia na stronę logowania.



Rysunek 15: Strona rejestracji biura rachunkowego i jego administratora

Źródło: Zrzut ekranu systemu

2. Okno rejestracji pracowników biura rachunkowego - strona zawiera formularz z 4 polami oraz przyciski do zatwierdzenia i drugi do anulowania procesu rejestracji.

Add Employee

Username:

First Name:

Last Name:

Enter your password

Submit

Cancel

Rysunek 16: Okno rejestracji pracowników biura rachunkowego

Źródło: Zrzut ekranu systemu

3. Strona rejestracji firmy klienckiej biura rachunkowego - strona zawiera formularz z 1 polem oraz przycisk do zatwierdzenia procesu rejestracji.

Accounting Office Manager

Register Client Company

Please enter details

Name:

Register

Rysunek 17: Strona rejestracji firmy klienckiej biura rachunkowego

Źródło: Zrzut ekranu systemu

4. Okno rejestracji pracowników firmy klienckiej biura rachunkowego - strona zawiera formularz z 4 polami oraz przyciski do zatwierdzenia i drugi do anulowania procesu rejestracji.

Rysunek 18: Okno rejestracji pracowników firmy klienckiej biura rachunkowego

Źródło: Zrzut ekranu systemu

6.2.2 Okno logowania

Okno logowania jest bardzo podobne do stron rejestracji. Zawiera w sobie formularz z dwoma polami do uzupełnienia: username oraz password, gdzie użytkownik wpisuje swoje dane, aby zakończyć proces logowania. Na stronie znajdują się również dwa przyciski do zatwierdzenia wprowadzonych danych oraz do przejścia do strony rejestracji biura rachunkowego i jego AOAdmina.

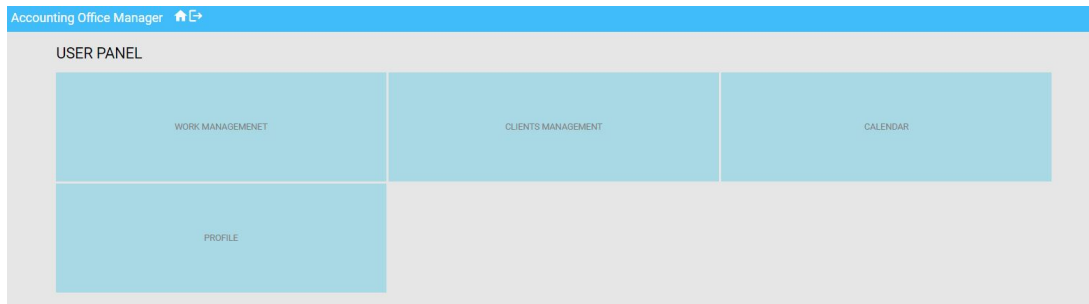
Rysunek 19: Strona logowania

Źródło: Zrzut ekranu systemu

6.2.3 Okno domowe zalogowanego użytkownika

Składa się ono z tabeli przycisków prowadzących do konkretnych funkcjonalności systemu. W zależności od roli użytkownika przycisków może być od 3 do 7. Wspólne dla wszystkich są przyciski CALENDAR i PROFILE.

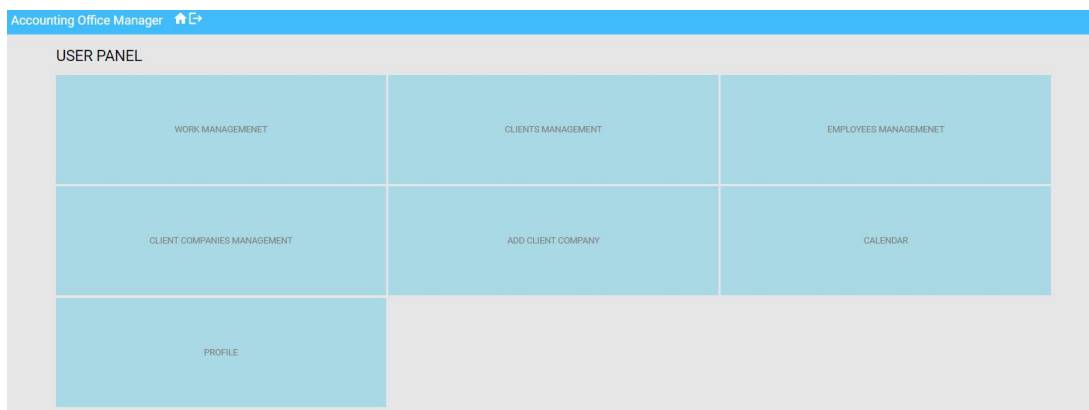
- Wszyscy pracownicy biura rachunkowego mają dwa dodatkowe przyciski: WORK MANAGEMENT i CLIENTS MANAGEMENT.



Rysunek 20: Strona domowa zalogowanego użytkownika, wersja dla USER

Źródło: Zrzut ekranu systemu

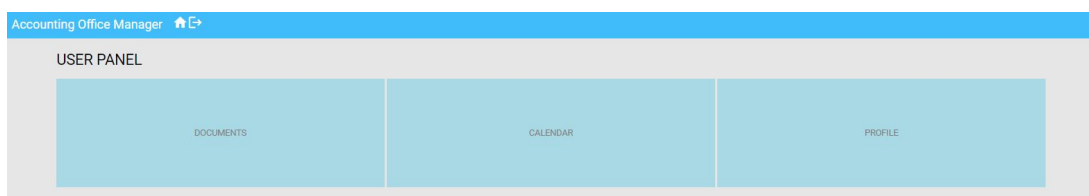
- Dla Admina i AO_Admina wyświetlane są jeszcze przyciski: EMPLOYEE MANAGEMENT, CLIENT COMPANIES MANAGEMENT oraz ADD CLIENT COMPANY.



Rysunek 21: Strona domowa zalogowanego użytkownika, wersja dla ADMIN i AO ADMIN

Źródło: Zrzut ekranu systemu

- Użytkownicy z rolą CLIENT mają dodatkowy przycisk: DOCUMENTS.



Rysunek 22: Strona domowa zalogowanego użytkownika, wersja dla CLIENT

Źródło: Zrzut ekranu systemu

6.2.4 Okno zarządzania pracą

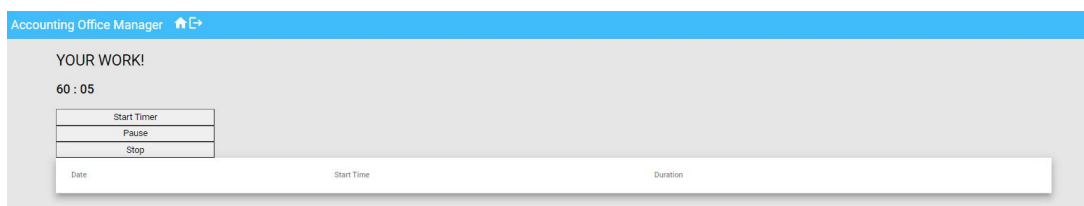
Okno zarządzania pracą jest podzielone na dwie części: licznik czasu pracy oraz tabelę z historią zapisanych czasów pracy użytkownika.

Pierwsza z nich składa się z licznika odliczającego od zera czas pracy sterowanego za pomocą trzech przycisków:

- START - odpowiada za wystartowanie licznika
- PAUSE - odpowiada za wstrzymanie odliczania przez licznik
- STOP - odpowiada za zastopowanie licznika oraz zapisanie jego odczytów do bazy danych

Druga z nich składa się z tabeli zawierającej historię zapisanych czasów pracy użytkownika pobranej z bazy danych. Każdy wiersz tabeli to osobny rekord. Tabela zawiera trzy kolumny:

- Date - data zapisu rekordu
- Start Time - czas wystartowania zapisywania rekordu
- Duration - czas trwania zapisywanego rekordu



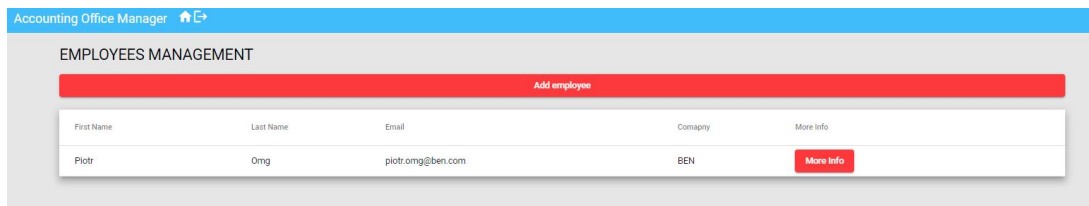
Rysunek 23: Strona zarządzania pracą

Źródło: Zrzut ekranu systemu

6.2.5 Okno zarządzania pracownikami

Okno zarządzania pracownikami składa się z przycisku "Add employee" oraz tabeli z pracownikami będącymi podwładnymi aktualnie zalogowanego użytkownika. Przycisk otwiera okno pozwalające na rejestrację pracowników biura rachunkowego. Tabela zawiera następujące kolumny:

- First Name
- Last Name
- Email
- Company
- More Info - przycisk przenoszący do strony profilu konkretnego podwładnego



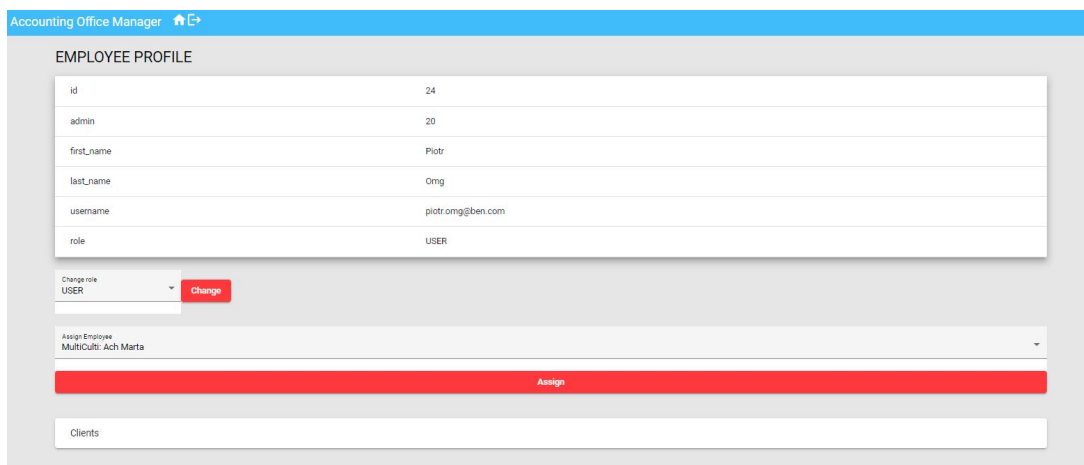
Rysunek 24: Strona zarządzania pracownikami

Źródło: Zrzut ekranu systemu

6.2.6 Okno profilu podwładnego

Okno profilu podwładnego jest podzielone na cztery sekcje:

- tabela zawierająca informację na temat podwładnego
- pole wyboru pozwalające na zmianę roli danego użytkownika. Dostępne opcje to: "USER" i "ADMIN". Jest ono widoczne tylko dla użytkownika będącego przełożonym podwładnego.
- pole wyboru z auto uzupełnieniem pozwalające przypisać podwładnemu podwładnych z listy użytkowników będących zarządzanych przez aktualnie zalogowanego użytkownika. Jest ono widoczne tylko dla użytkownika będącego przełożonym podwładnego.
- lista klientów przypisanych do danego podwładnego.



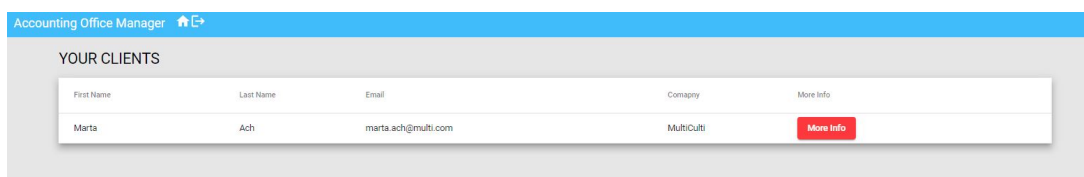
Rysunek 25: Strona profilu podwładnego

Źródło: Zrzut ekranu systemu

6.2.7 Okno zarządzania klientami

Okno zarządzania klientami składa się z tabeli z klientami przypisanymi do aktualnie zalogowanego użytkownika. Tabela zawiera następujące kolumny:

- First Name
- Last Name
- Email
- Company
- More Info - przycisk przenoszący do strony profilu konkretnego klienta



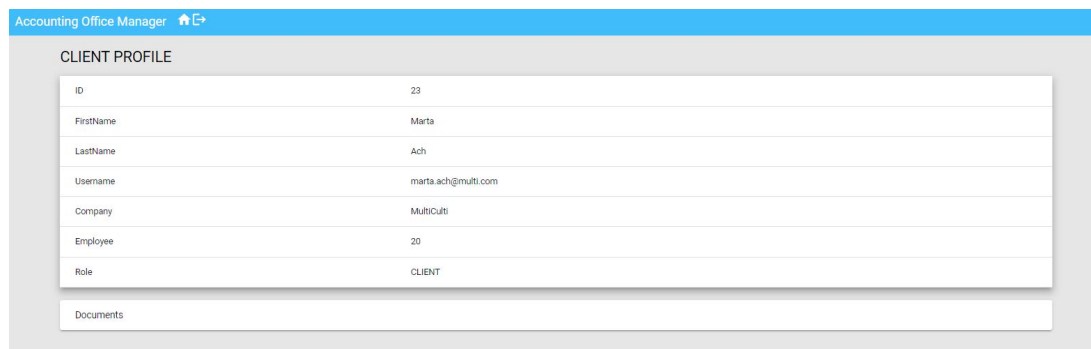
Rysunek 26: Strona zarządzania klientami

Źródło: Zrzut ekranu systemu

6.2.8 Okno profilu klienta

Okno profilu klienta jest podzielone na dwie sekcje:

- tabela zawierająca informację na temat danego klienta
- lista dokumentów przypisanych do danego klienta.



CLIENT PROFILE	
ID	23
FirstName	Marta
LastName	Ach
Username	marta.ach@multi.com
Company	MultiCulti
Employee	20
Role	CLIENT
Documents	

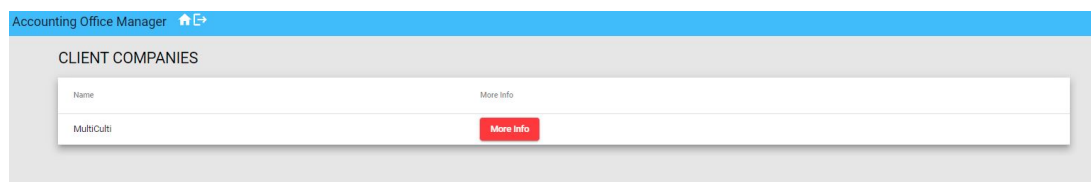
Rysunek 27: Strona profilu klienta

Źródło: Zrzut ekranu systemu

6.2.9 Okno zarządzania firmami klienckimi

Okno zarządzania firmami klienckimi składa się z tabeli z firmami klienckimi przypisanymi do biura rachunkowego aktualnie zalogowanego użytkownika. Tabela zawiera następujące kolumny:

- Name
- More Info - przycisk przenoszący do strony profilu konkretnej firmy klienckiej



CLIENT COMPANIES	
Name	More Info
MultiCulti	More Info

Rysunek 28: Strona zarządzania firmami klienckimi

Źródło: Zrzut ekranu systemu

6.2.10 Okno profilu firmy klienckiej

Okno profilu firmy klienckiej jest podzielone na trzy sekcje:

- tabela zawierająca informację na temat danej firmy klienckiej

- przycisk "Add client" otwierający okno rejestracji pracowników firmy klienckiej
- lista dokumentów przypisanych do danej firmy klienckiej

First Name	Last Name	Email	Company	More Info
Marta	Ach	marta.ach@multi.com	MultiCulti	More Info

Rysunek 29: Strona profilu firmy klienckiej

Źródło: Zrzut ekranu systemu

6.2.11 Okno zarządzania profilem zalogowanego użytkownika

Okno zarządzania profilem zalogowanego użytkownika jest podzielone na trzy sekcje:

- tabela zawierająca informację na temat obecnie zalogowanego użytkownika
- przycisk "Show employee profile" lub "Show client profile" prowadzący na stronę profilu pracowniczego, lub klienckiego danego użytkownika
- formularz zmiany hasła

Rysunek 30: Strona zarządzania profilem zalogowanego użytkownika

Źródło: Zrzut ekranu systemu

6.3 Propozycje dalszych prac

System jest otwarty na rozszerzenia i dodawanie kolejnych funkcjonalności systemu. Propozycje dalszych prac obejmują m.in.:

- Dodanie funkcjonalności kontaktu klienta i biura rachunkowego w formie np. czatu
- Ulepszenie endpointów, polepszając szybkości wczytywania niektórych komponentów
- Dodanie funkcjonalności dodawania danych specjalnych użytkowników i firm
- Dodanie stylów wyglądu aplikacji frontendowej systemu

6.4 Wnioski

Praca nad tym projektem zakończyła się stworzeniem systemu posiadającego wszystkie niezbędne funkcjonalności do zarządzania biurem rachunkowym.

Udało się m.in.:

- zapewnić bezpieczeństwo aplikacji poprzez zastosowanie systemu autoryzacji i autentykacji użytkowników
- zaimplementować funkcjonalności zawarte w istniejących już rozwiązaniach, takich jak: zarządzanie pracownikami, zarządzanie czasem pracy oraz zarządzanie danymi klientów
- zaimplementować funkcjonalność przesyłania dokumentów między użytkownikami systemu, oraz ich przechowywania w systemie plików
- zaimplementować system potwierdzenia wysyłki dokumentów poprzez wprowadzenie odpowiednich metadanych do właściwości pliku przesyłanego

Z tego powodu jest on bardziej odpowiedni do postawionego przed nim zadania niż istniejące rozwiązania opisane w rozdziale 2 tej pracy. W związku z tym założenia projektu zostały spełnione.

Spis tabel

1	Analiza zagrożeń	5
2	Porównanie funkcjonalności porównywanych rozwiązań	10
3	Struktura tabeli User	29
4	Struktura tabeli Employee	29
5	Struktura tabeli Client	29
6	Struktura tabeli Roles	29
7	Struktura tabeli UserRole	30
8	Struktura tabeli Company	30
9	Struktura tabeli AccountingOffice	30
10	Struktura tabeli ClientCompany	30
11	Struktura tabeli Documents	30
12	Struktura tabeli WorkLog	31
13	Struktura tabeli Calendar	31
14	Struktura tabeli CalendarUser	31

Spis rysunków

1	Diagram pracowników największych firm rachunkowych w Stanach Zjednoczonych w 2020	3
2	Schemat koncepcyjny końcowego produktu	5
3	Logo BQE Core	7
4	Logo Karbon	8
5	Logo Xero Practice manager	9
6	Java	13
7	Spring Boot	13
8	Angular	14
9	MySQL	14
10	Architektura	15
11	Architektura serwera	16
12	Schemat Bazy Danych	28
13	GitHub	32
14	Trello	32
15	Strona rejestracji biura rachunkowego i jego administatora	36
16	Okno rejestracji pracowników biura rachunkowego	37
17	Strona rejestracji firmy klienckiej biura rachunkowego	37
18	Okno rejestracji pracowników firmy klienckiej biura rachunkowego	38
19	Strona logowania	38
20	Strona domowa zalogowanego użytkownika, wersja dla USER	39
21	Strona domowa zalogowanego użytkownika, wersja dla ADMIN i AO ADMIN	39
22	Strona domowa zalogowanego użytkownika, wersja dla CLIENT	39

23	Strona zarządzania pracą	40
24	Strona zarządzania pracownikami	41
25	Strona profilu podwładnego	42
26	Strona zarządzania klientami	42
27	Strona profilu klienta	43
28	Strona zarządzania firmami klienckimi	43
29	Strona profilu firmy klienckiej	44
30	Strona zarządzania profilem zalogowanego użytkownika	44

Bibliografia

- [1] Organizacja pracy w biurze rachunkowym
Anna Chojnacka-Komorowska
Finanse, Rynki Finansowe, Ubezpieczenia 94 (1):289-297
Wydawnictwo Naukowe Uniwersytetu Szczecińskiego
2018
Wersja na dzień: 08.09.2021
<https://www.ceeol.com/search/article-detail?id=763989>
- [2] 37 Amazing Accounting Statistics Showing the Power of Numbers [2021]
Ana Djurovic
February 23, 2021
Wersja na dzień: 25.07.2021
<https://goremotely.net/blog/accounting-statistics/>
- [3] Number of employees of the leading accounting firms in the United States in 2020
Statista
2021
Wersja na dzień: 9.09.2021
<https://www.statista.com/statistics/188939/number-of-employees-of-leading-us-accounting-firms-2011/>
- [4] Accounting Statistics & Facts
Milena
21.06.2021
Wersja na dzień: 25.07.2021
<https://balancingeverything.com/accounting-statistics/>
- [5] Wykorzystanie narzędzi informatycznych i mediów społecznościowych w zarządzaniu relacjami z klientami
Michał Koziół, Anna Karaś, Paweł Bełzowski
ZARZĄDZANIE I JAKOŚĆ
Zeszyty Naukowe Małopolskiej Wyższej Szkoły Ekonomicznej w Tarnowie
Tom 44 Nr 4 (2019)
30.12.2019

Wersja na dzień: 08.09.2021

<https://doi.org/10.25944/znmwse.2019.04.105120>

[6] Funkcjonowanie Biura Rachunkowego

Zofia Sawicka-Kłuźniak

Zachodniopomorski Uniwersytet Technologiczny

2011

Wersja na dzień: 08.09.2021

https://wneiz.pl/nauka_wneiz/frfu/32-2011/FRFU-32-633.pdf

[7] Ryzyko w działalności biur rachunkowych

Elżbieta Klamut

Społeczna Akademia Nauk

Przedsiębiorczość i Zarządzanie

2012, 13, z. 15, Problemy Zarządzania, 11–34

Wersja na dzień: 25.07.2021

<http://bazekon.icm.edu.pl/bazekon/element/bwmeta1.element.ekon-element-000171577768>

[8] BQE Core Suite

Wersja na dzień: 09.09.2021

<https://www.bqe.com/>

[9] BQE Core Suite Logo

Wersja na dzień: 09.09.2021

<https://www.bqe.com/company/news-press-releases>

[10] Karbon

Wersja na dzień: 09.09.2021

<https://karbonhq.com/>

[11] Karbon Logo

Wersja na dzień: 09.09.2021

<https://www.g2.com/products/karbon-2019-08-27/reviews>

- [12] Xero Practice Manager
Wersja na dzień: 09.09.2021
<https://www.xero.com/au/xero-practice-manager/>
- [13] Xero Practice Manager Logo
Wersja na dzień: 09.09.2021
<https://fyidocs.com/nz/xero-practice-manager-2/>
- [14] Java 11 - java.com
Wersja na dzień: 09.09.2021
<https://www.java.com/pl/download/>
- [15] Spring Boot - spring.io
Wersja na dzień: 09.09.2021
<https://spring.io/projects/spring-boot/>
- [16] Differences Between Java vs Node JS
Priya Pdamkar
Wersja na dzień: 28.07.2021
<https://www.educba.com/java-vs-node-js/>
- [17] Java Logo
Wersja na dzień: 09.09.2021
<https://www.java.com/pl/download/>
- [18] Spring Boot Logo
Wersja na dzień: 09.09.2021
<https://www.pngaaa.com/detail/2459579>
- [19] Angular 11 - angular.io
Wersja na dzień: 09.09.2021
<https://angular.io/>
- [20] 15 Suprising Stats About Angular
2muchcoffee
27.02.2020

Wersja na dzień: 28.07.2021

<https://2muchcoffee.com/blog/15-surprising-stats-about-angular/>

[21] Angular vs React vs Vue: Which Framework to Choose in 2021

Shaumik Daityari

15.03.2021

Wersja na dzień: 28.07.2021

<https://2muchcoffee.com/blog/15-surprising-stats-about-angular/>

[22] Angular Logo

Wersja na dzień: 09.09.2021

<https://angular.io/>

[23] MySQL - mysql.com

Wersja na dzień: 09.09.2021

<https://www.mysql.com/>

[24] Top 10 Databases to Use in 2021

Md Kamaruzzaman

20.01.2021

Wersja na dzień: 28.07.2021

<https://towardsdatascience.com/top-10-databases-to-use-in-2021-d7e6a85402ba>

[25] MySQL Logo

Wersja na dzień: 09.09.2021

<https://www.mysql.com/>

[26] Spring Boot Architecture

Md Kamaruzzaman

20.01.2021

Wersja na dzień: 28.07.2021

<https://www.javatpoint.com/spring-boot-architecture>

[27] GitHub - github.com

Wersja na dzień: 09.09.2021

<https://github.com/>

[28] Sourcetree

Wersja na dzień: 09.09.2021

<https://www.sourcetreeapp.com/>

[29] Trello - trello.com

Wersja na dzień: 09.09.2021

<https://trello.com/>

[30] logo GitHub - clipart.info

Wersja na dzień: 09.09.2021

<https://clipart.info/github-logo-png-4934>

[31] Trello logo

Wersja na dzień: 09.09.2021

<https://logos-world.net/trello-logo/>

Wszystkie powyższe odnośniki były aktywne w dniu: 25 lipca 2021 roku.