

POLITECHNIKA LUBELSKA

Wydział Elektrotechniki i Informatyki

Kierunek informatyka



PRACA INŻYNIERSKA

Dziennik elektroniczny zgodny z wymaganiami RODO

An electronic grade book compatible with GDPR requirements

Promotor:

Dr inż. Grzegorz Koziel

Dyplomanci:

Krzysztof Barczak nr albumu: 84138

Dariusz Głowacki nr albumu: 84147

Marcin Górski nr albumu: 84628

Lublin 2019

OŚWIADCZENIE

Oświadczam, że przedstawiona praca inżynierska została napisana przeze mnie osobiście, a przytoczone w niej cytaty oraz dane źródłowe zostały udokumentowane zgodnie z wymogami prawa autorskiego.

Jednocześnie wyrażam zgodę na wykorzystywanie fragmentów wydrukowanej wersji mojej pracy pt. „Dziennik elektroniczny zgodny z wymaganiami RODO” w publikacjach naukowych wykonywanych przez pracowników Politechniki Lubelskiej, za zgodą Dyrektora Instytutu na zasadach wynikających z Ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz. U. 2000 r. Nr 80, poz. 904, z późn. zm.). Potwierdzam zgodność tekstu drukowanego z zapisem w formie elektronicznej.

.....

(podpis dyplomanta)

.....

(podpis dyplomanta)

.....

(podpis dyplomanta)

Spis treści

Rozdział 1. Wstęp.....	4
1.1. Cel i zakres pracy.....	5
1.2. Streszczenie pracy.....	6
1.3. Abstract.....	6
1.4. Wykorzystane technologie i narzędzia.....	7
1.5. Podział prac.....	9
Rozdział 2. Projekt Systemu.....	10
2.1. Wymagania funkcjonalne i нефункционалне.....	10
2.2. Diagram przypadków użycia.....	13
2.3. Diagramy BPMN.....	14
2.4. Diagramy sekwencji.....	20
2.5. Diagramy aktywności.....	23
2.6. Diagram EER.....	25
Rozdział 3. Implementacja systemu.....	26
3.1. Kontrolery.....	26
3.2. Encje.....	28
3.3. Usługi.....	30
3.4. Struktura widoków.....	31
Rozdział 4. Testy aplikacji.....	34
4.1. Testy funkcjonalne.....	34
4.1.1. Podstrona terminarza.....	35
4.1.2. Podstrona logowania.....	38
4.2. Testy jednostkowe.....	43
Rozdział 5. Wnioski.....	45
Bibliografia.....	46
Spis rysunków.....	47
Spis listingów.....	48

1. Wstęp

Szybki rozwój technologiczny oraz rozpowszechnienie dostępu do internetu wymusiły szereg zmian i przekształceń w wielu dziedzinach życia społecznego, zarówno w skali globalnej jak i lokalnej. Wspomniane zjawisko nie ominęło również szkoły, które zmieniła swoje tradycyjne oblicze. Obecnie komunikacja na linii szkoła – rodzic i uczeń weszła w nową fazę rozwoju, nie ograniczając się już wyłącznie do co semestralnych spotkań tzw. „wywiadówek”. Aktualnie prawie każda placówka edukacyjna posiada dziennik elektroniczny, który umożliwia rodzicom stałą i bieżącą obserwację postępów ucznia w nauce, usprawnia pracę nauczycieli, ułatwia kontakt szkoły z opiekunami dzieci, a także pozwala na śledzenie wydarzeń z życia szkoły.

W wyniku analizy aktualnego rynku dzienników elektronicznych wyłoniono dwa najbardziej popularne i rozpowszechnione dzienniki elektroniczne „Librus”[7] i „Vulcan”[10], których dostępność sprawiła, że szybko stały się monopolistami na polskim rynku. Obie wspomniane aplikacje posiadają podobne funkcje. Należą do nich takie możliwości jak: sprawdzanie ocen, obecności oraz innych osiągnięć ucznia.

Opisywany w niniejszej pracy program również będzie spełniał te kryteria mając jednocześnie następującą funkcję: semestralna ocena pracy nauczycieli przez uczniów w kilku aspektach np. pod względem skuteczności nauczania czy dodatkowych umiejętności. Warto podkreślić, że będzie on zgodny z nowym rozporządzeniem o ochronie danych osobowych czyli RODO¹.

Pragniemy złożyć serdeczne podziękowania Panu dr inż. Grzegorzowi Koziełowi za cenne uwagi i życzliwość okazaną nam w trakcie pisania niniejszej pracy.

¹ Rozporządzenie Parlamentu Europejskiego i Rady (UE) 2016/679 z dnia 27 kwietnia 2016 r. w sprawie ochrony osób fizycznych w związku z przetwarzaniem danych osobowych i w sprawie swobodnego przepływu takich danych oraz uchylenia dyrektywy 95/46/WE (ogólne rozporządzenie o ochronie danych).[9]

1.1. Cel i zakres pracy

Celem niniejszej pracy jest stworzenie dziennika elektronicznego mającego za zadanie ułatwienie pracy nauczycielom oraz komunikację uczniów i opiekunów ucznia ze szkołą. Będzie on również umożliwiał ocenianie nauczycieli po każdym zakończonym semestrze.

Zakres pracy obejmuje:

- 1) zapoznanie się z istniejącymi technologiami webowymi oraz zgłębienie wiedzy na temat ich działania,
- 2) analiza funkcjonalności istniejących dzienników elektronicznych,
- 3) zapoznanie się z doświadczeniami użytkowników istniejących już dzienników elektronicznych,
- 4) zaprojektowanie diagramów,
- 5) utworzenie bazy danych,
- 6) wykonanie strony obsługującej dziennik elektroniczny,
- 7) testy aplikacji.

1.2. Streszczenie pracy

Praca składa się z pięciu rozdziałów. Rozdział pierwszy zawiera wprowadzenie do problematyki pracy, omówienie wykorzystywanych technologii i narzędzi oraz podział zakresu prac pomiędzy poszczególnymi autorami. Do drugiego rozdziału pracy wybrano i przedstawiono ważne diagramy takie jak diagram przypadków użycia, BPMN, sekwencji, aktywności oraz EER. W trzecim rozdziale zaprezentowano implementację systemu który został zrealizowany zgodnie z wzorcem MVC gdzie interfejs użytkownika dostępny jest przez przeglądarkę internetową. W czwartym rozdziale przeprowadzono testy, które zostały zrealizowane w celu sprawdzenia poprawności działania aplikacji. Wykonane testy zostały podzielone na dwa główne typy: testy funkcjonalne oraz testy jednostkowe. Do sprawdzenia poprawności systemu przeprowadzono szereg testów. Wnioski przedstawiono w ostatnim, piątym rozdziale.

1.3. Abstract

The work consists of five chapters. The first chapter includes an introduction to the issues of work, a description of technologies and tools used, and a division of work between authors. For the second chapter of the work, important diagrams such as the use case diagram, BPMN, sequence, activity and EER have been selected and presented. The third chapter presents the implementation of the system that was implemented in accordance with the model-view-controller (MVC) model. The user interface is available through web browser. In the fourth chapter tests were carried out to verify the correctness of the application. The tests were divided into two main types: functional tests and unit tests. A series of tests was carried out to check the correctness of the system. Applications are presented in the last, fifth chapter.

1.4. Wykorzystane technologie i narzędzia

Podczas pisania pracy wykorzystano technologie i narzędzia niezbędne do utworzenia aplikacji dziennika elektronicznego. Są to darmowe i popularne technologie, do których ma dostęp każdy i które umożliwiają tworzenie aplikacji webowych.

„Bootstrap jest darmowym i obecnie jednym z najpopularniejszych frameworków HTML i CSS na świecie. Pozwala na stosunkowo szybkie tworzenie interfejsów stron internetowych i aplikacji przeglądarkowych, a to wszystko dzięki wstępnie przygotowanym arkuszom CSS oraz modułom HTML i JavaScript. Do najważniejszych cech Bootstrapa można zaliczyć schludną i ładną typografię, style dla większości elementów HTML, wspieranie się językiem JavaScript, a konkretnie jego biblioteką jQuery, oraz solidną siatkę responsywną zaprojektowaną według koncepcji Mobile-First². Bootstrap jest kompatybilny ze wszystkimi najnowszymi wersjami przeglądarek Opera, Safari, Firefox, Google Chrome, Internet Explorer oraz Edge³.”[6]

Do ostylowania aplikacji internetowej zastosowano CSS czyli "Kaskadowe arkusze stylów składają się z kilku specyfikacji określających różne poziomy i profile. Każdy poziom powstał na bazie poprzedniego i zawiera jakieś nowości w stosunku do poprzednika. Specyfikacje te mają nazwy CSS 1, CSS 2 oraz CSS3.”[1] Wdrożenie tej technologii umożliwia modyfikację widoku strony.

PHP jest to język, który interpretuje się po stronie serwera, można dzięki niemu obsługiwać dane wysyłać oraz je odbierać oraz bezpiecznie połączyć się z bazą danych co jest niezbędne w tej pracy inżynierskiej. „PHP pozwala stworzyć niezbędne funkcje takie jak obsługa formularzy. PHP to skryptowy język programowania służący głównie do tworzenia stron internetowych. PHP jest rozprowadzany na otwartej licencji i każdy może pobrać za darmo jego kopię, zainstalować i używać bez żadnych ograniczeń zarówno do celów prywatnych jak i komercyjnych. Język jest prosty w nauce i umożliwia tworzenie profesjonalnych dynamicznych stron internetowych.”[11]

Symfony jest frameworkiem MVC (model, widok i kontroler) napisany w języku PHP. Wspomniany framework jest w pełni obiektowy, podobnie jak i inne języki wysokiego poziomu. Symfony wspiera podstawowe potrzeby aplikacji webowych takie jak wyświetlanie podstron, obsługiwanie bazy danych, obsługę formularzy oraz treści.

Wstęp

Dodatkowo posiada niezależne moduły, które wspierają aplikację webową. Rzeczą wyróżniającą framework, są zaawansowane widoki. „W Symfony 2 domyślnym językiem przetwarzania widoków jest Twig. Pliki widoków mają podwójne rozszerzenie .html.twig.”[5]

Dane aplikacji przechowywane są w relacyjnej bazie danych MySQL. „MySQL to prawdziwy system bazodanowy, wspierający zaawansowane techniki replikacji danych i mogący przechowywać bez problemu miliardy rekordów. ”[3]

NetBeans jest środowiskiem wspierające wiele języków programowania, między innymi do technologii webowych czyli html, css, PHP i MySQL. Wspomniany powyżej NetBeans wspomaga pisanie kodu oraz zawiera strukturę katalogów, dzięki której można zarządzać całym projektem. Oprogramowanie pozwala również na doinstalowanie wtyczek, które dodatkowo ułatwiają prace nad pisanym kodem.

Wstęp

1.5. Podział prac

Podział prac wykonanych przez poszczególnych autorów przedstawiono w Tabeli 1.1. W przypadku gdy numer rozdziału został wskazany przy wielu autorach oznacza to, że rozdział ten był tworzony wspólnie przez wymienione osoby.

Tabela 1.1. Podział prac

Zakres pracy			Krzysztof Barczak	Dariusz Głowacki	Marcin Górski
Wstęp			+	+	+
Projekt Systemu	Wymagania funkcjonalne i нефункционалne		+	-	-
	Diagram przypadków użycia		+	+	+
	Diagramy BPMN		+	+	+
	Diagramy sekwencji		+	+	+
	Diagramy aktywności		+	+	+
	Diagram EER		-	+	-
Implementacja systemu	Kontrolery		-	-	+
	Encje		-	+	+
	Usługi		-	-	+
	Struktura widoków		+	-	+
Testy aplikacji	Testy funkcjonalne	Podstrona terminarz	+	-	-
		Podstrona logowanie	-	+	-
	Testy jednostkowe		-	-	+
Wnioski			+	+	+

2. Projekt Systemu

Przed przystąpieniem do prac mających na celu stworzenia systemu podjęto się opracowania projektu. Na podstawie przeglądu rynku oraz konsultacji ze specjalistami z dziedziny oświaty, zidentyfikowano potrzeby rynku oraz procesy biznesowe realizowane w placówkach oświatowych. W konsekwencji opracowano wymagania funkcjonalne jak również нефunkcjonalne oraz uszczegółowiono projekt za pomocą diagramów aktywności i sekwencji. Finalnym etapem było opracowanie struktury bazy danych.

2.1. Wymagania funkcjonalne i нефunkcjonalne

Głównym zadaniem dziennika będzie wspomaganie zarządzaniem zajęciami w szkole. W niniejszym rozdziale przedstawiono wymagania funkcjonalne i нефunkcjonalne, które musi spełnić opracowywany w ramach pracy dziennik elektroniczny. W opracowywanym projekcie zdefiniowano następujące wymagania funkcjonalne:

- 1) uczniowie mają mieć możliwość:
 - a) logowania/wylogowania,
 - b) podglądu ocen z przedmiotów oraz ocen z zachowania,
 - c) sprawdzania daty sprawdzianów,
 - d) wysyłania/odbierania wiadomości,
 - e) podglądu do uwag wystawionych przez nauczyciela,
 - f) oceniania nauczycieli po semestrze,
- 2) rodzice mają mieć możliwość:
 - a) logowania/wylogowania,
 - b) podglądu ocen z przedmiotów oraz ocen z zachowania,
 - c) sprawdzania daty sprawdzianów,
 - d) wysyłania/odbierania wiadomości,
 - e) usprawiedliwiania nieobecności ucznia,

Projekt Systemu

- f) odbierania uwag wystawionych przez nauczyciela,
- 3) nauczyciele mają mieć możliwość:
- a) logowania/wylogowania,
 - b) dodawania/edytowania/usuwania ocen oraz typu ocen z przedmiotów,
 - c) dodawania/edytowania/usuwanie informacji o sprawdzianach,
 - d) sprawdzania obecności,
 - e) wysyłania/odbierania wiadomości,
 - f) wystawiania uwag uczniom,
- 4) wychowawca ma mieć możliwość:
- a) logowania/wylogowania,
 - b) usprawiedliwiania nieobecności,
 - c) wysyłania/odbierania wiadomości,
 - d) dodawania/edytowania/usuwania oceny z zachowania,
- 5) administrator/dyrektor ma mieć możliwość:
- a) dodawania kont uczniów/nauczycieli/rodziców,
 - b) edytowania kont uczniów/nauczycieli/rodziców,
 - c) usuwania kont uczniów/nauczycieli/rodziców,
 - d) przypisania/usunięcia ucznia z klasy,
 - e) dodania podziału klas,
 - f) obsługi ocen semestralnych oraz promowania uczniów do następnej klasy,
 - g) zaliczenia/niezaliczenia uczniom roku,
 - h) obsługi planu zajęć,
 - i) podglądu do ocen nauczycieli wystawianych przez uczniów.

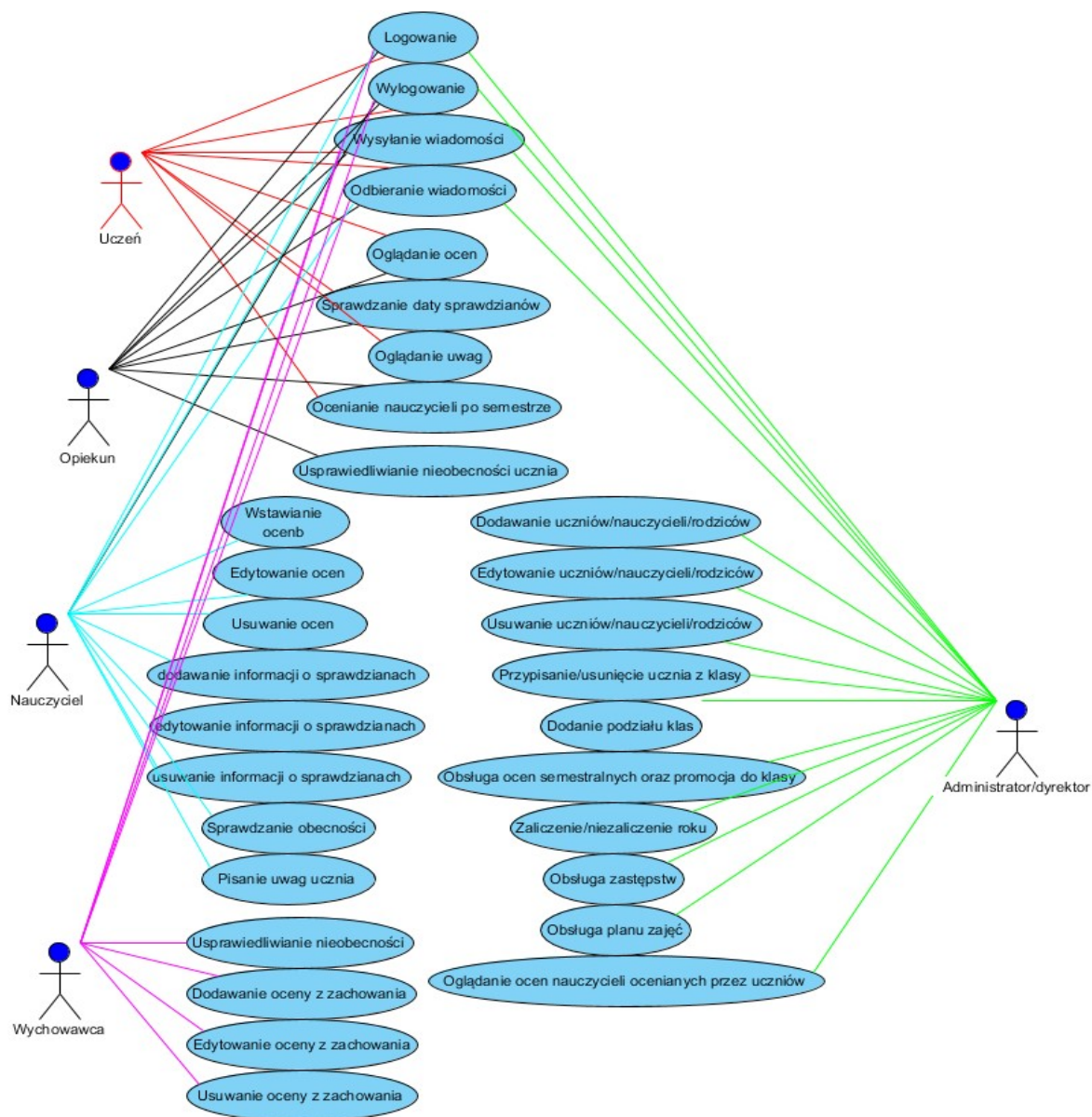
Projekt Systemu

Ze względu na konieczność zapewnienia powszechnego dostępu do systemu użytkownikom dysponującym różnego rodzaju sprzętem i oprogramowaniem zdecydowano się na udostępnienie interfejsu użytkownika poprzez przeglądarkę. W związku z powyższym zostały zdefiniowane następujące wymagania niefunkcjonalne:

- 1) aplikacja ma działać na przeglądarkach Google Chrome (wersja 65 i nowsze) oraz Firefox (wersja 59 i nowsze) zarówno w wersjach desktopowych jak i mobilnych,
- 2) aplikacja ma być napisana w języku PHP,
- 3) aplikacja ma używać bazy danych MySQL,
- 4) aplikacja ma być responsywna na urządzenia:
 - a) telefony: 320 px,
 - b) tablety: 768 px,
 - c) monitory: 1240 px.

2.2. Diagram przypadków użycia

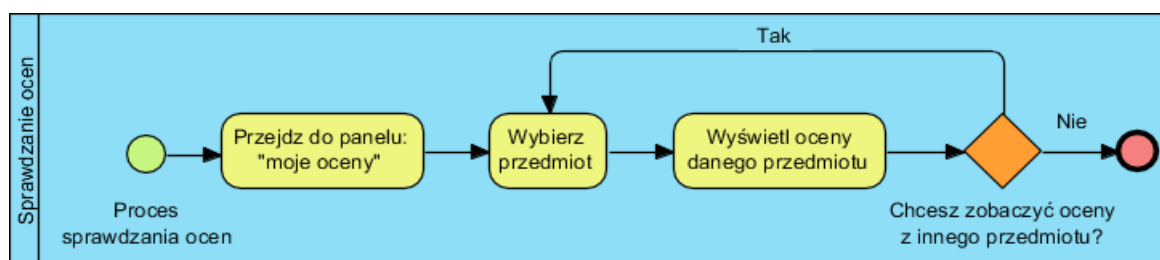
Dziennik elektroniczny może być używany przez różnych użytkowników, przy czym każdy z nich może wykonywać inne czynności. Na Rysunku 2.1 przedstawiono diagramy przypadków użycia systemu.



Rysunek 2.1. Diagram przypadków użycia

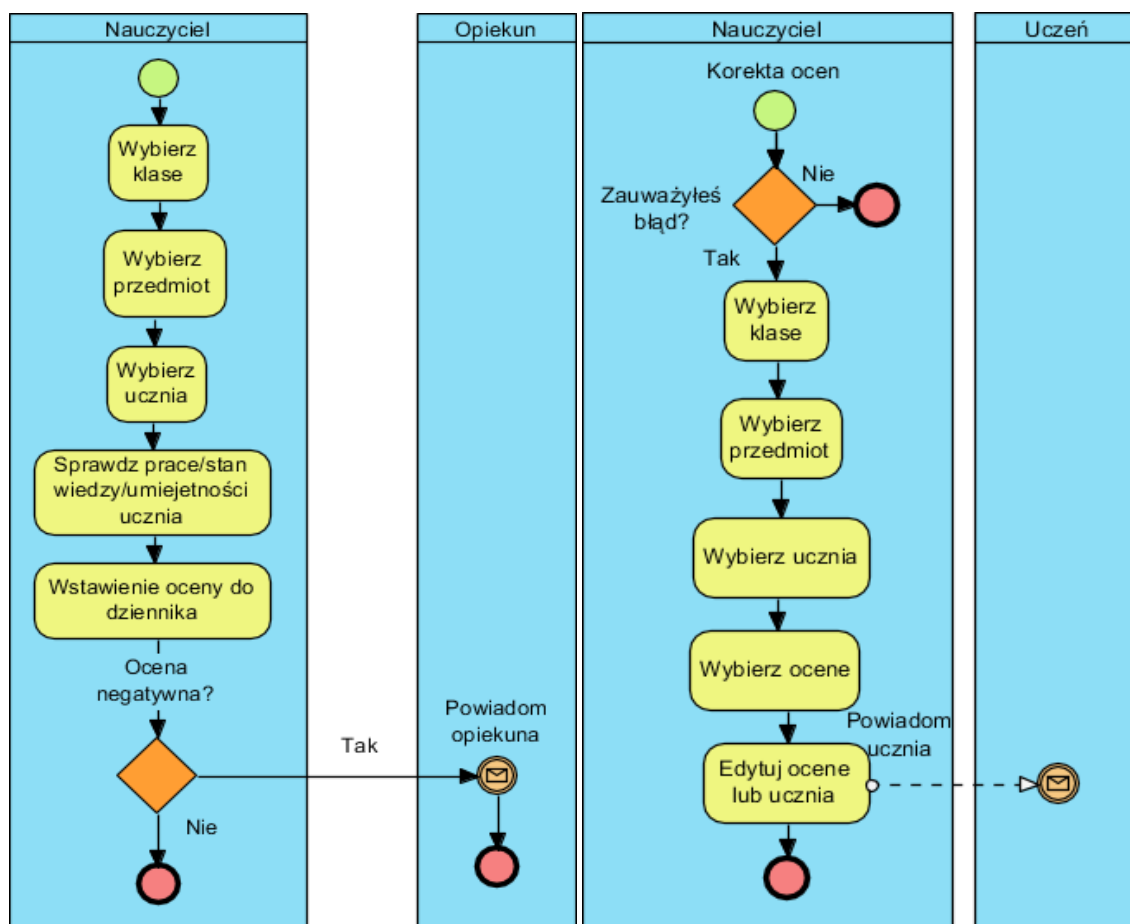
2.3. Diagramy BPMN

Opracowanie projektu obejmuje identyfikację procesów biznesowych zachodzących w szkole. Procesy te opisano za pomocą diagramów BPMN. „BPMN to standard opracowany przez organizację Object Management Group (OMG). Jego pełna nazwa to Business Process Model and Notation. Podstawowym celem tego standardu jest dostarczenie notacji do opisywania procesów biznesowych, która jest czytelna i zrozumiała zarówno dla biznesowych "użytkowników", monitorujących procesy i zarządzających nimi, dla analityków, którzy przeprowadzają biznesową analizę procesów, jak i programistów, odpowiedzialnych za ich techniczną implementację.”[2] Najważniejsze procesy zostały zobrazowane w niniejszym podrozdziale. Na Rysunku 2.2 przedstawiono proces sprawdzania ocen widoczny z poziomu ucznia. Rysunek 2.3 wskazuje na proces wystawiania ocen przez nauczyciela. Na Rysunku 2.4 umieszczono diagram obrazujący proces poprawiania ocen wcześniej wprowadzonych do systemu.



Rysunek 2.2. Sprawdzanie ocen

Projekt Systemu

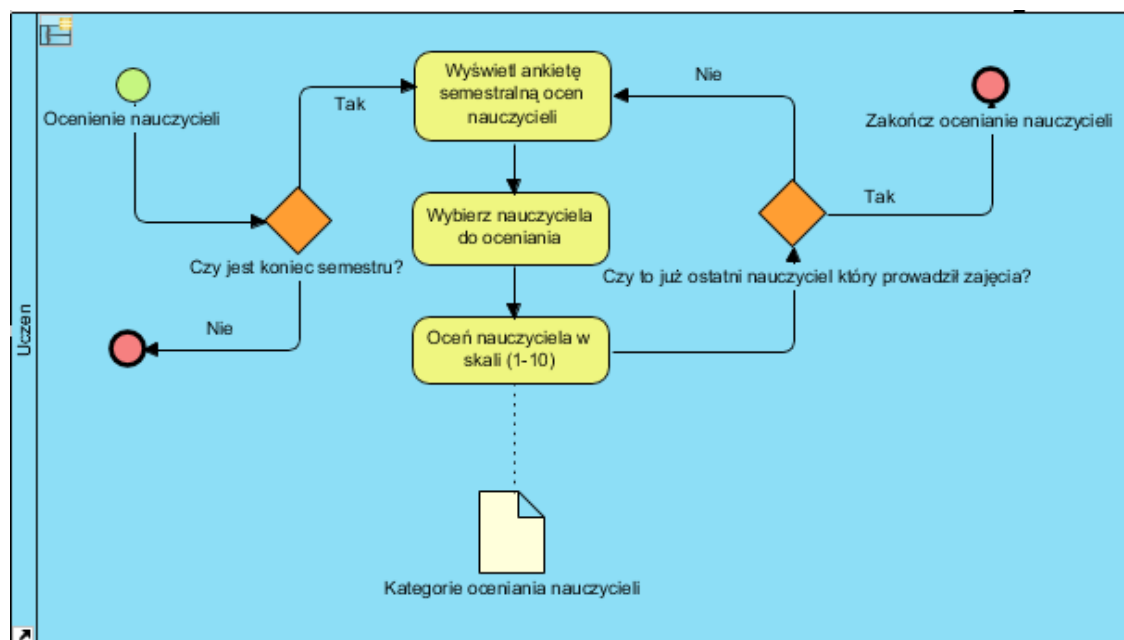


Rysunek 2.3. Wstawianie ocen

Rysunek 2.4. Korekta ocen

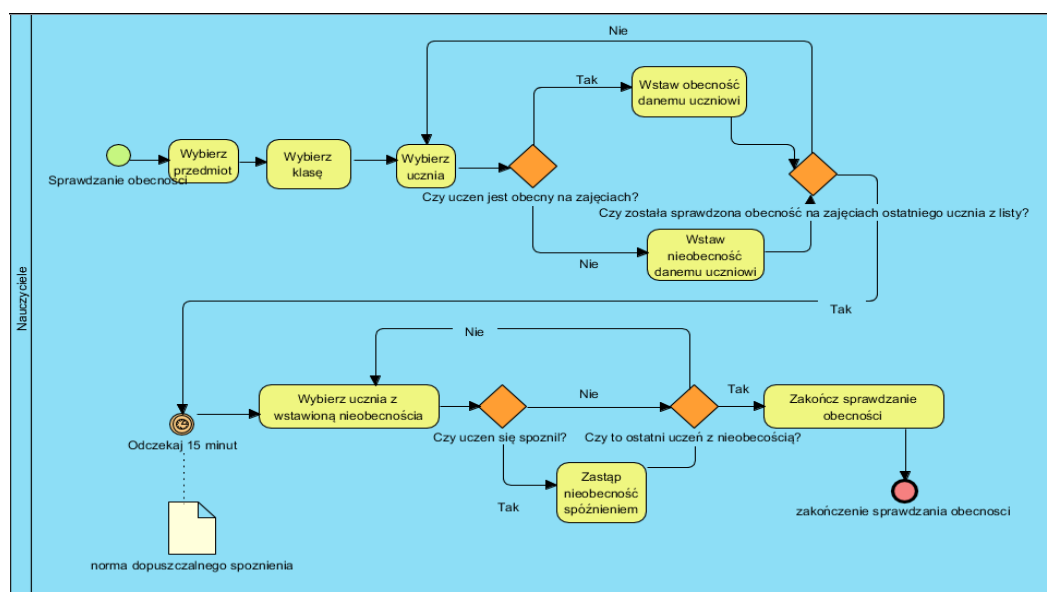
Projekt Systemu

Diagram przedstawiający ocenianie nauczycieli został przedstawiony na Rysunku 2.5. Omawiana praca inżynierska wyróżnia się ankietą wypełnianą pod koniec semestru, która umożliwia ocenę pracy nauczycieli za cały ten okres.



Rysunek 2.5. Ocenianie nauczycieli

Częstym problem z którym borykają się nauczyciele jest kwestia nieobecności uczniów na zajęciach. Kontrola absencji uczniów została zobrażowana za pomocą diagramu BPMN na Rysunku 2.6.



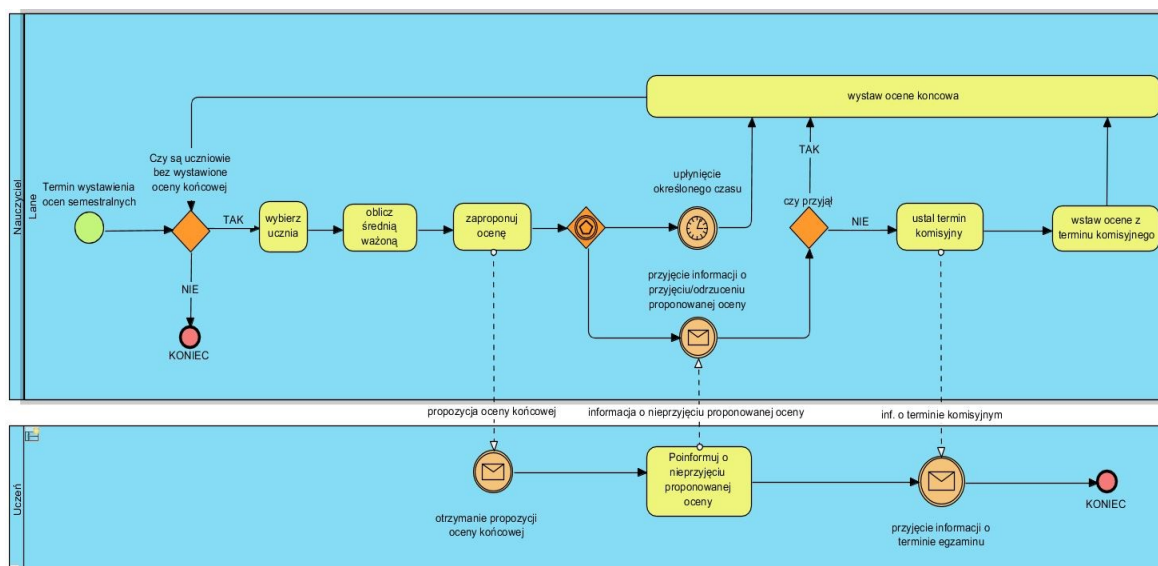
Rysunek 2.6. Sprawdzanie obecności

Diagram obrazujący proces kontroli nieobecności ucznia został przedstawiony na Rysunku 2.7. System ma wspomagać wysyłanie komunikatów wychowawcy do ucznia oraz jego opiekuna.



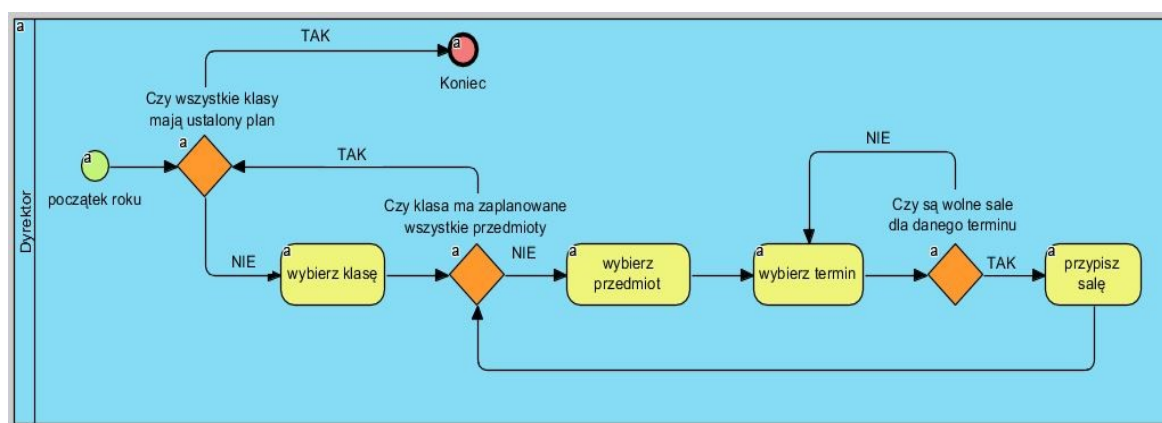
Projekt Systemu

W związku z faktem że w szkołach co semestr praktykowane jest wystawianie oceny podsumowującej pracę ucznia, w niniejszej aplikacji została zaimplementowana podobna funkcja. Diagram przedstawiający oceny semestralne ucznia umieszczono na Rysunku 2.8.



Rysunek 2.8. Oceny semestralne

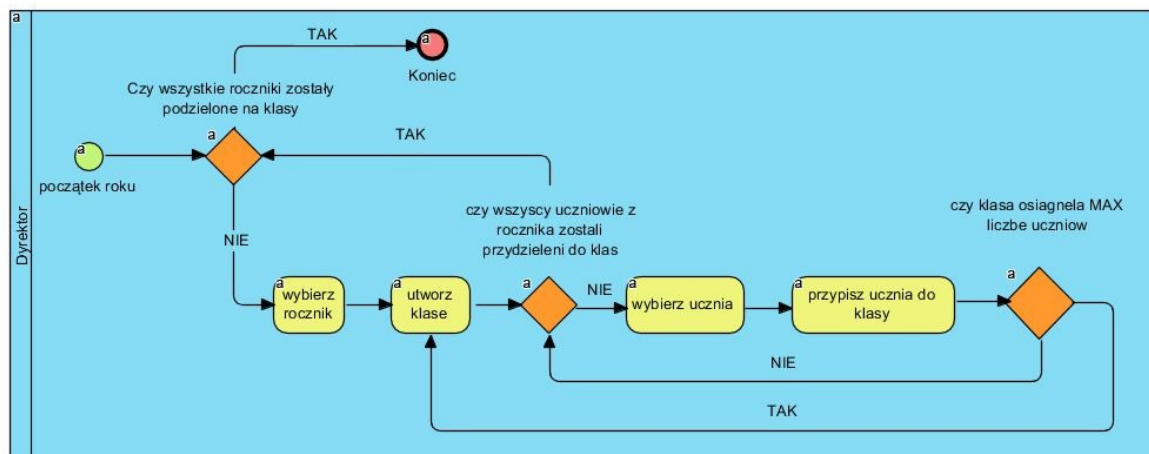
Potrzeba dobrej organizacji planu zajęć dla uczniów i nauczycieli również wymaga stworzenia dodatkowej funkcjonalności. Diagram przedstawiający tworzenie planu zajęć został zilustrowany na Rysunku 2.9.



Rysunek 2.9. Tworzenie planu zajęć

Projekt Systemu

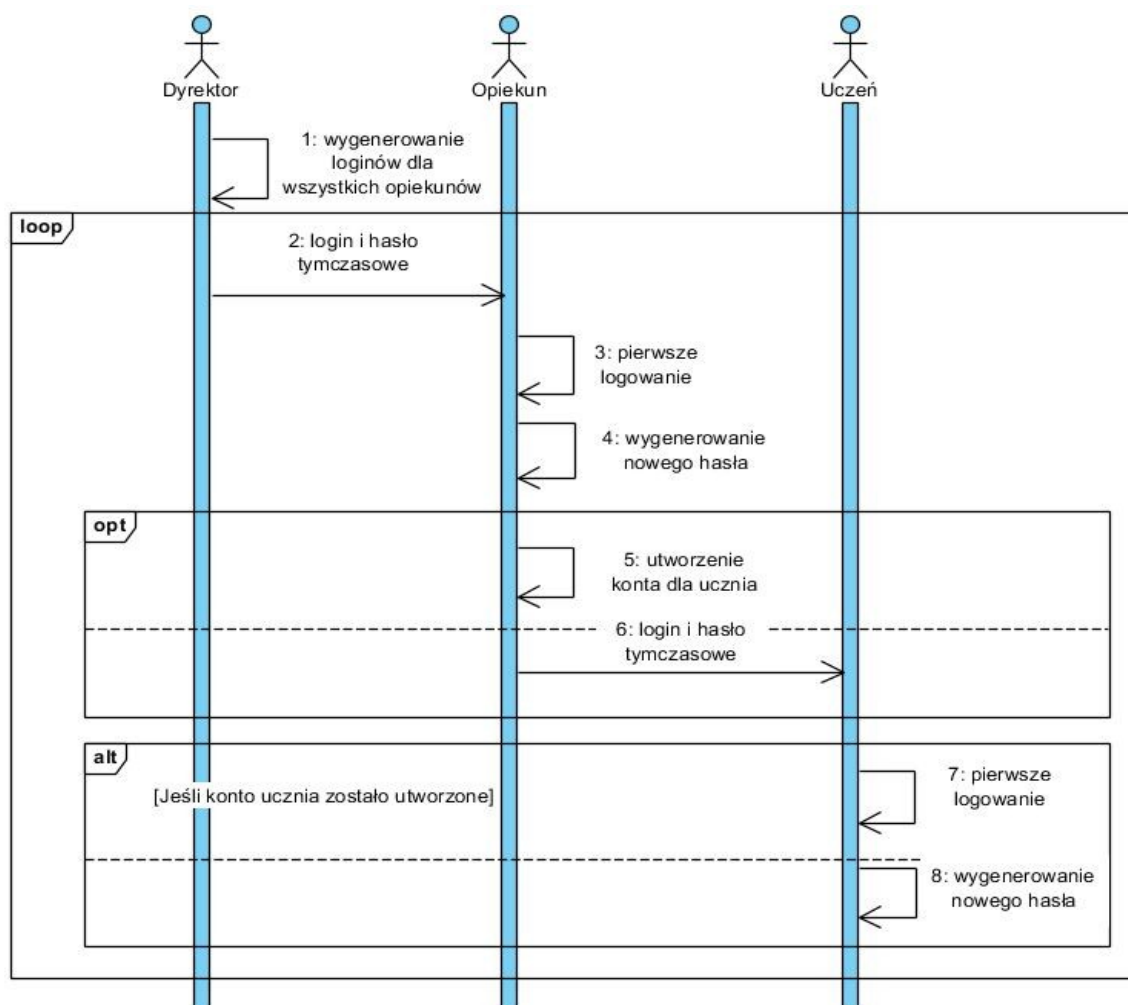
System umożliwia podział ogółu uczniów na poszczególne klasy. Diagram przedstawiający podział uczniów na klasy przedstawiono na Rysunku 2.10.



Rysunek 2.10. Tworzenie klas

2.4. Diagramy sekwencji

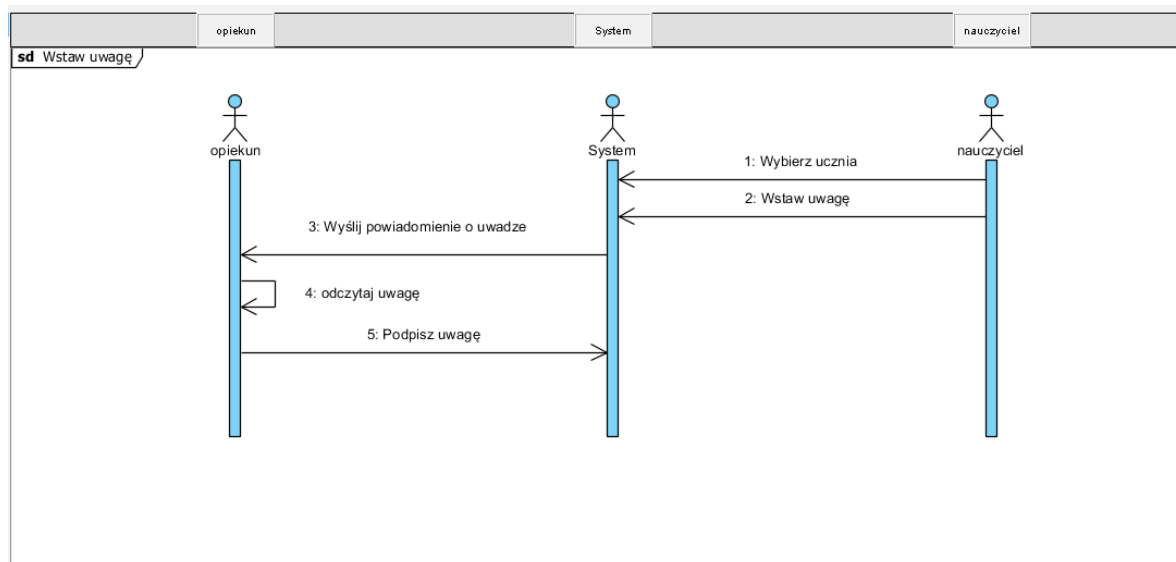
Diagramy sekwencji wspomagają zobrazowanie i identyfikację przepływu komunikatów wewnątrz systemu. Na Rysunku 2.11 przedstawiono diagram sekwencji definiujący przepływ komunikatów w procesie tworzenia kont użytkowników.



Rysunek 2.11. Tworzenie kont

Projekt Systemu

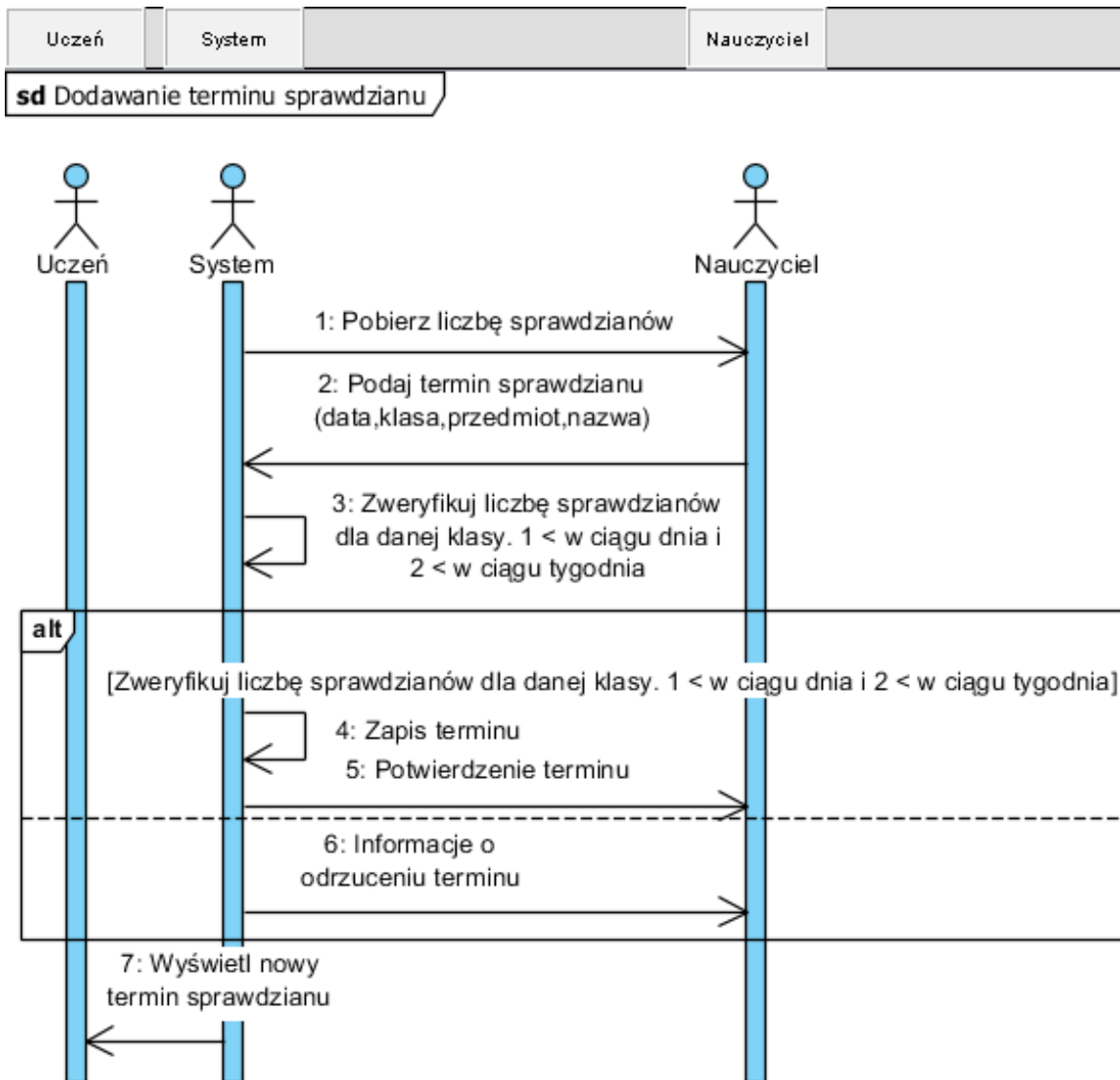
Za pośrednictwem systemu nauczyciel może kontaktować się z opiekunem ucznia. Rysunek 2.12 przedstawia przepływ komunikatu – uwagi od nauczyciela do opiekuna ucznia.



Rysunek 2.12. Wstawianie uwag

Projekt Systemu

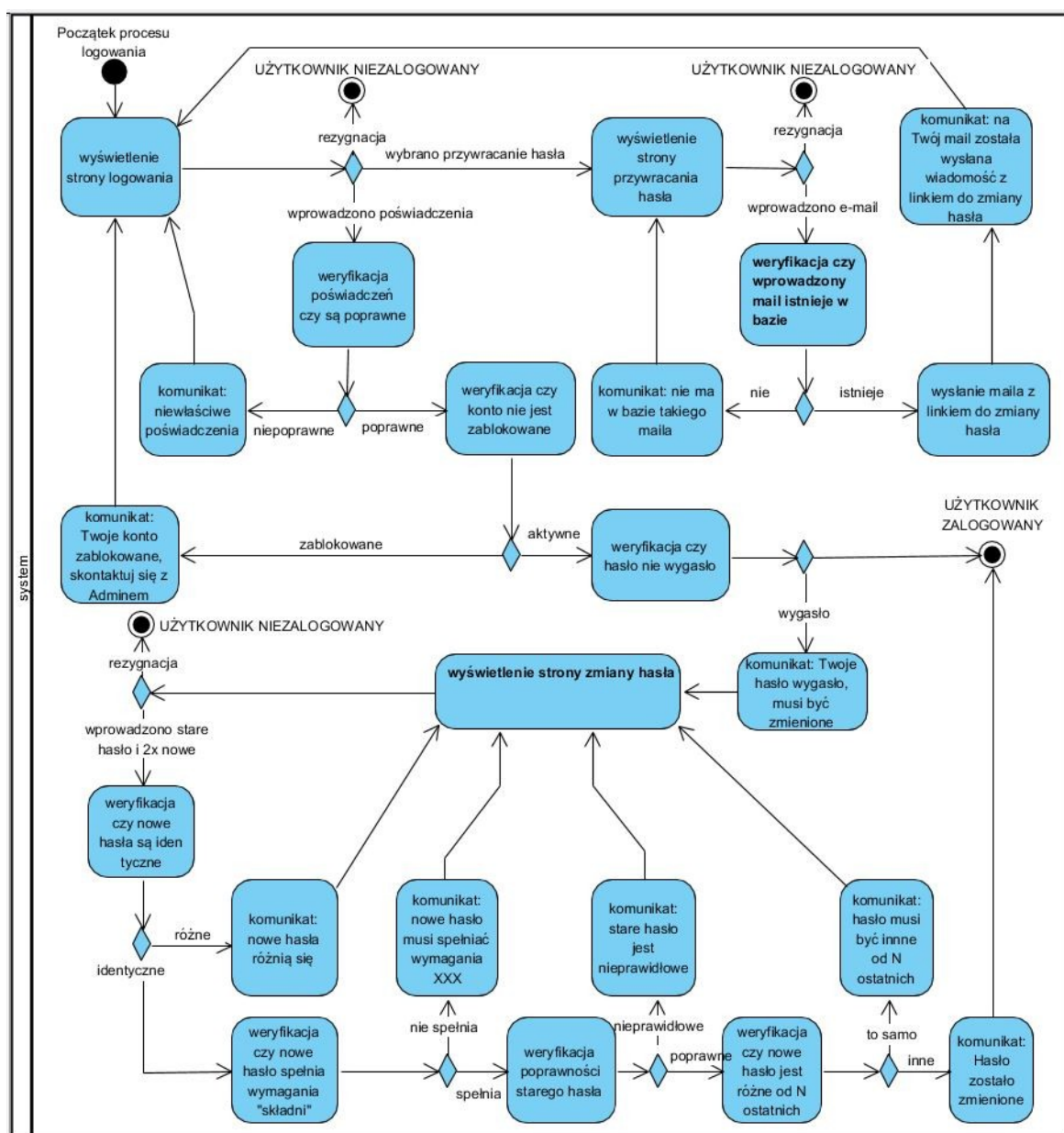
Diagram sekwencji przedstawiony na Rysunku 2.13, ilustruje przepływ komunikatów podczas procesu dodawania terminów sprawdzianów zgodnie z wstępnie zdefiniowanym limitem sprawdzianów, które mogą się odbyć w ciągu jednego dnia oraz jednego tygodnia.



Rysunek 2.13. Dodawanie terminów sprawdzianów

2.5. Diagramy aktywności

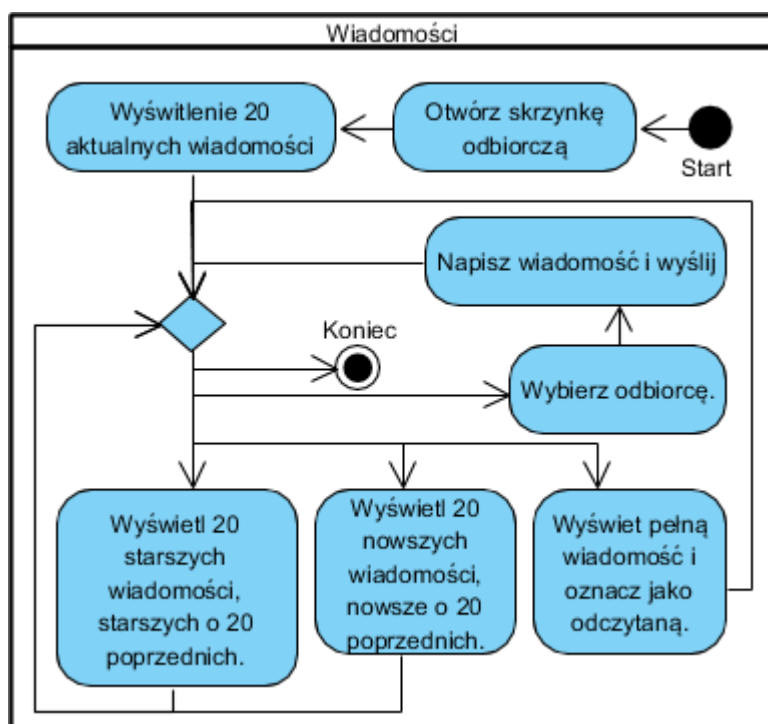
Aplikacja w celu umożliwienia dostępu do konta weryfikuje wprowadzone dane autoryzacyjne. Jeśli są one prawidłowe, pozwala użytkownikowi korzystać z aplikacji. Implementowana funkcjonalność została wprowadzona w celu zapewnienia bezpieczeństwa danych. Szczegółowy przebieg procesu logowania został przedstawiony na Rysunku 2.14.



Rysunek 2.14. Przebieg procesu logowania

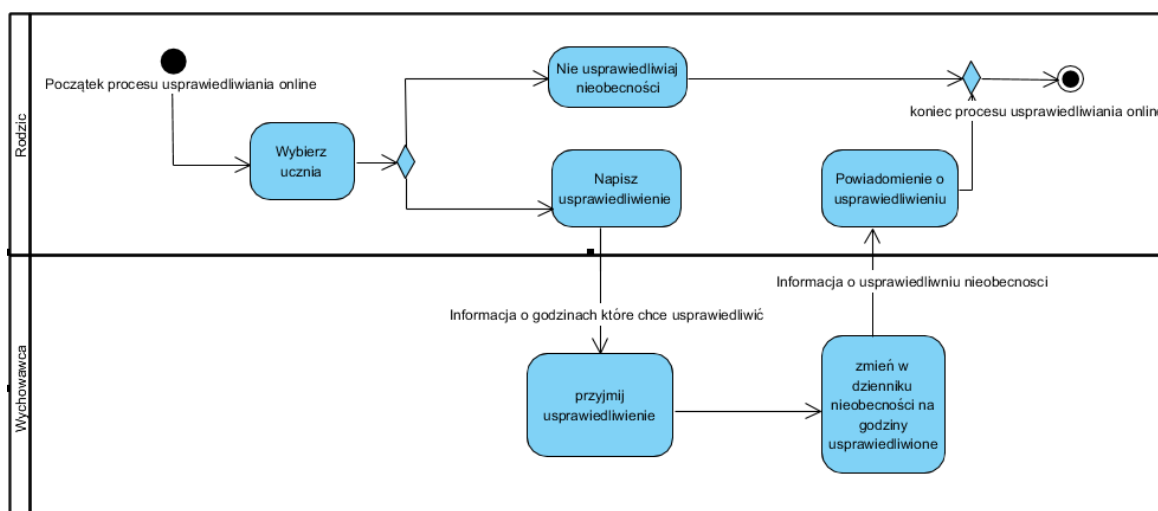
Projekt Systemu

Diagram przedstawiający proces wysyłania oraz odbierania wiadomości przedstawiono na Rysunku 2.15. Dostęp do tej funkcjonalności posiadają wszyscy użytkownicy systemu.



Rysunek 2.15. Pisanie wiadomości

Kolejny rysunek obrazuje możliwość usprawiedliwienia nieobecności ucznia w szkole przez opiekuna oraz podanie przyczyny absencji. Diagram przedstawiający usprawiedliwienia nieobecności ucznia ukazuje Rysunek 2.16.



Rysunek 2.16. Usprawiedliwianie online

2.6. Diagram EER

The diagram illustrates a database schema for a school system. It includes the following tables and their attributes:

- wiadomosci**: id INT(11), nadawca INT(11), odbiorca INT(11), tytul VARCHAR(100), tresc TEXT, zalacznik VARCHAR(45), status_nadawcy TINYINT(1), status_odbiorcy TINYINT(1), odczytana TINYINT(1).
- pracownicy**: id INT(11), imie VARCHAR(15), nazwisko VARCHAR(30), kontakt TEXT, role SET(...), uzytkownik INT(11), status TINYINT(1).
- klasy**: id INT(11), poziom INT(11), klasa VARCHAR(1), rocznik VARCHAR(4), wychowawca INT(11), status TINYINT(1).
- przedmioty**: id INT(11), prowadzacy INT(11), przedmiot INT(11), status TINYINT(1).
- sownik_przed...**: id INT(11), nazwa VARCHAR(30), opis TEXT.
- uzytkownicy**: id INT(11), login VARCHAR(45), sol VARCHAR(10), typ VARCHAR(20), mail VARCHAR(60), status INT(2).
- uwagi**: id INT(11), uczen INT(11), nauczyciel INT(11), tresc TEXT, odczytane TINYINT(1), status TINYINT(1).
- ocenianie**: id INT(11), uczen INT(11), prowadzacy INT(11), ocena INT(11).
- godz_lek**: id INT(11), poczatek DATETIME.
- terminarz**: id INT(11), sala INT(11), godzina INT(11), dzien_tygodnia VARCHAR(13), kto_co INT(11), klasa INT(11), typ VARCHAR(10), poczatek DATE, koniec DATE, opis TEXT.
- hasla**: id INT(11), uzytkownik INT(11), haslo VARCHAR(200), sol VARCHAR(10), data DATETIME, proby INT(11).
- uczniowie**: id INT(11), imie VARCHAR(15), imie_2 VARCHAR(15), nazwisko VARCHAR(30), data_urodzenia DATE, pesel VARCHAR(11), numer_legitymacji INT(11), miejscowosc VARCHAR(45), ulica VARCHAR(45), nr_domu VARCHAR(10), kod_poczty VARCHAR(6), poczta VARCHAR(45), kontakt TEXT, klasa INT(11), opiekun INT(11), uzytkownik INT(11), status TINYINT(1).
- zajecia**: id INT(11), temat VARCHAR(255), opis TEXT, termin INT(11), data DATETIME.
- obecnosci**: id INT(11), obecny INT(2), uczen INT(11), zajecia INT(11).
- oceny**: id INT(11), ocena DECIMAL(2,1), waga INT(11), kiedy DATETIME, uczen INT(11), przedmiot INT(11), typ VARCHAR(10), uwagi TEXT, status TINYINT(1).
- sale**: id INT(11), nr_sali VARCHAR(4), opis TEXT.
- const**: id INT(11), name VARCHAR(255), value VARCHAR(255).
- opiekunowie**: id INT(11), imie VARCHAR(15), nazwisko VARCHAR(30), kontakt TEXT, uzytkownik INT(11), status TINYINT(1).

Relationships are indicated by lines with crow's foot notation symbols (1:1, 1:M, etc.).

Rysunek 2.17. Relacyjna baza danych

3. Implementacja systemu

W niniejszym rozdziale przedstawiony został opis głównych elementów systemu opracowywanego w ramach pracy inżynierskiej. Najważniejsze części składowe kodu możemy podzielić na: kontrolery, encje (ang. entities), usług (ang. utils), strukturę widoków (html.twig) oraz na komponenty realizujące dostęp do bazy danych MySQL.

System został zrealizowany zgodnie z wzorcem model-widok-kontroler (ang. Model-View-Controller – MVC). Interfejs użytkownika dostępny jest przez przeglądarkę. Backend został zrealizowany w języku PHP z wykorzystaniem frameworka Symfony.

3.1. Kontrolery

Częścią systemu odpowiedzialną za przechwytywanie wszystkich żądań ze strony WWW są kontrolery. W opisywanym systemie dokonano podziału kontrolerów ze względu na role obsługiwanych użytkowników. W omawianej aplikacji kontrolerami są klasy: UserController, AdminController, StudentController, GuardController oraz TeacherController. Z kolei zagłębiając się w strukturę konkretnego kontrolera możemy wyodrębnić funkcje (metody, klasy) odpowiedzialne za czynności, jakie może wykonywać użytkownik posiadający określoną rolę, np. administrator może zarządzać salami lekcyjnymi. W funkcjach możemy wyodrębnić „uchwyty stron” oraz „hierarchię komend”[8]. Przykładowy uchwyt do funkcji lessonHoursAction umożliwiającej zarządzanie godzinami lekcyjnymi przedstawia Listing 3.1. Sposób dodawania nowych godzin lekcyjnych zaprezentowano na Listingu 3.2.

Listing 3.1. Uchwyt strony umożliwiający operacje na godzinach lekcyjnych

```
/**
 * @Route("/admin/uzytownicy/{formType}/{id}/{delete}", name="users",
 * defaults={"formType"="pracownik","id"="0","delete"="0"})
 */
```

Implementacja systemu

Listing 3.2. Formularz dodawania godziny lekcyjnej

```
if($id!=0) $formValue=$this->getDoctrine()->getRepository(GodzLek::class)->find($id);

$form=$this->createFormBuilder($formValue)
    ->setMethod('POST')
    ->add('poczatek',TimeType::class,['widget'=>'single_text'])
    ->add('submit',SubmitType::class)
    ->getForm();
```

Na Listingu 3.3 umieszczony został kod odpowiedzialny za dodawanie nowego rekordu, natomiast kod umożliwiający usuwanie rekordów przedstawiono na Listingu 3.4.

Listing 3.3. Fragment kodu realizujący dodawanie nowej godziny lekcyjnej

```
//dodawanie
if($request->isMethod('post') && $id==0){
    $lessonHours=new GodzLek();
    $lessonHours->setPoczatek(new DateTime($_POST['form']['poczatek']));

    $entityManager->persist($lessonHours);
    $entityManager->flush();
}
```

Listing 3.4. Kod umożliwiający usuwanie godziny lekcyjnej

```
}else if($delete==1 && $id!=0){
    $entityManager->createQuery(
        "DELETE AppBundle\Entity\GodzLek g " .
        "WHERE g.id=".$id
    )
    ->getResult();

    return $this->redirectToRoute('lesson_hours');
}
```

Kod zwracający widok do użytkownika w postaci html przedstawia Listing 3.5.

Listing 3.5. Kod zwracający widok, który prezentuje listę godzin lekcyjnych

```
return $this->render('admin/lesson_hours.html.twig',[
    'form'=>$form->createView(),
    'lessonHours'=>$lessonHours
]);
```

Listing 3.6. Usuwanie sesji użytkownika

```
/**
 * @Route ("/wyloguj", name="logout")
 */
public function logoutAction() {
    //usuwanie sesji użytkownika
    $this->get('session')->remove('user');
    $this->get('session')->remove('student');
    $this->get('session')->remove('admin');
    $this->get('session')->remove('teacher');
    $this->get('session')->remove('classTeacher');

    $this->get('session')->set('info', 'Wylogowano. ');

    return $this->redirectToRoute('login', [], 201);
}
```

W kontrolerze UserController niezbędny przy wylogowywaniu jest kod odpowiedzialny za usuwanie sesji użytkownika, którego przedstawiono w Listingu 3.6.

3.2. Encje

W celu zarządzania danymi dostępnymi w bazie danych, stworze są klasy-encje, które reprezentują tabele z całą ich strukturą. Każdej tabeli w bazie danych przypisana jest odpowiednia klasa encji. Pozwala to na obiektowe podejście do bazy danych. Przykład kodu tworzącego encję przedstawiono na Listingu 3.7.

Implementacja systemu

Listing 3.7. Encja do tabeli GodzLek

```
1 <?php
2
3 namespace AppBundle\Entity;
4
5 use Doctrine\ORM\Mapping as ORM;
6
7 /**
8  * GodzLek
9  *
10  * @ORM\Table(name="godz_lek")
11  * @ORM\Entity
12  */
13 class GodzLek
14 {
15     /**
16      * @var \DateTime
17      *
18      * @ORM\Column(name="poczatek", type="datetime", nullable=false)
19      */
20     private $poczatek;
21
22     /**
23      * @var integer
24      *
25      * @ORM\Column(name="id", type="integer")
26      * @ORM\Id
27      * @ORM\GeneratedValue(strategy="IDENTITY")
28      */
29     private $id;
30
31
32
33     /**
34      * Set poczatek
35      *
36      * @param \DateTime $poczatek
37      *
38      * @return GodzLek
39      */
40     public function setPoczatek($poczatek)
41     {
42         $this->poczatek = $poczatek;
43
44         return $this;
45     }
46
47     /**
48      * Get poczatek
49      *
50      * @return \DateTime
51      */
52     public function getPoczatek()
53     {
54         return $this->poczatek;
55     }
56
57     /**
58      * Get id
59      *
60      * @return integer
61      */
62     public function getId()
63     {
64         return $this->id;
65     }
66 }
67
```

Implementacja systemu

3.3. Usługi

Funkcje, które powtarzają się w wielu kontrolerach, umieszczone zostały w sekcji projektu „Utils”. Przykładem takiej usługi jest klasa Message, która liczy wszystkie nieodczytane wiadomości i zapisuje je w sesji każdego użytkownika w celu wyświetlenia tej liczby. Kod odpowiedzialny za działanie usługi Message przedstawiony został na Listingu 3.8

Listing 3.8. Usługa message

```
class Message{

    public $em;

    public function __construct($em){
        $this->em=$em;
    }

    public function count(){
        $session=new Session(new PhpBridgeSessionStorage());
        $sessionUserId=$session->get('user');

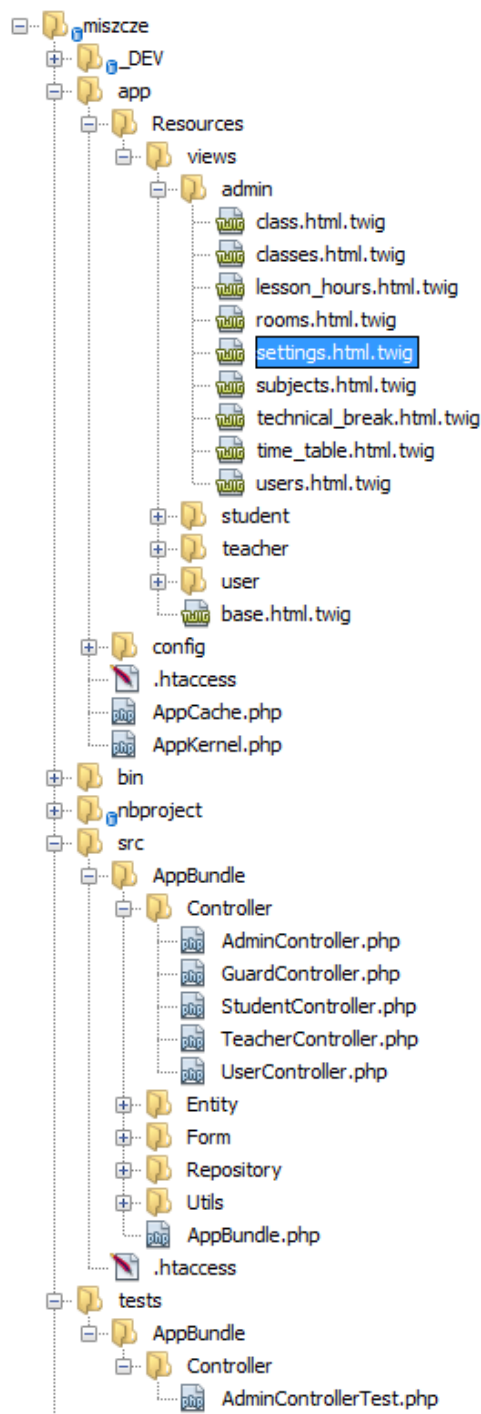
        if(isset($sessionUserId))
            $sessionUserId=$session->get('user')['user']->getId();
        else
            $sessionUserId=0;

        $messages=$this->em->createQuery(
            "SELECT count(w.id) countMessages ".
            "FROM AppBundle\Entity\Wiadomosci w ".
            "JOIN AppBundle\Entity\Uzytkownicy u ".
            "WITH w.odbiorca=u.id ".
            "WHERE w.odbiorca=".$sessionUserId." AND w.odczytana=0 AND w.statusOdbiorcy=0"
        )
        ->getResult();

        if($messages[0]['countMessages']>0)
            $session->set('newMessages',$messages[0]['countMessages']);
        else if(isset($session))
            $session->remove('newMessages');
    }
}
```

3.4. Struktura widoków

Zgodnie z modelem MVC zaimplementowano szablon Twig, który generuje widoki i przyjmuje zmienne przekazane przez kontroler. Zgodnie z dobrą praktyką struktura widoków odpowiada strukturze kontrolerów (Rys.3.1)



Rysunek 3.1. Struktura projektu aplikacji

Implementacja systemu

Przykładowym widokiem dostępnym w dzienniku dla użytkownika z rolą ucznia jest plan zajęć. Fragment kodu z widoku `time_table` zaprezentowano na Listingu 3.9, natomiast widok zwracany w przeglądarce zaprezentowany został na Rysunku 3.2.

Listing 3.9. Widok `time_table`

```
4  (* block body *)
5  <h1 class="text-center my-3">Plan zajęć</h1>
6
7  (* if lessons is not empty *)
8  <table class="table table-hover table-light border border-dark table-responsive-md">
9    <thead class="thead-dark">
10     <tr>
11       <th>Godz.</th>
12       <th>Poniedziałek</th>
13       <th>Wtorek</th>
14       <th>Środa</th>
15       <th>Czwartek</th>
16       <th>Piątek</th>
17     </tr>
18   </thead>
19   <tbody>
20     (* for i in 0..lessonHours|length-1 *)
21     <tr>
22       <td>
23         {{ lessonHours[i].poczatek|date('H:i') }} -
24         {{ lessonHours[i].poczatek|date_modify("+45 min")|date('H:i') }}
25       </td>
26       <td>
27         (* if lessons[i]['poniedzialek'] is not null *)
28         {{ lessons[i]['poniedzialek'].klasa.poziom }}{{ lessons[i]['poniedzialek'].klasa.klasa }} -
29         {{ lessons[i]['poniedzialek'].ktoCo.przedmiot.nazwa }}
30         (* endif *)
31       </td>
32       <td>
33         (* if lessons[i]['wtorek'] is not null *)
34         {{ lessons[i]['wtorek'].klasa.poziom }}{{ lessons[i]['wtorek'].klasa.klasa }} -
35         {{ lessons[i]['wtorek'].ktoCo.przedmiot.nazwa }}
36         (* endif *)
37       </td>
38       <td>
39         (* if lessons[i]['sroda'] is not null *)
40         {{ lessons[i]['sroda'].klasa.poziom }}{{ lessons[i]['sroda'].klasa.klasa }} -
41         {{ lessons[i]['sroda'].ktoCo.przedmiot.nazwa }}
42         (* endif *)
43       </td>
44       <td>
45         (* if lessons[i]['czwartek'] is not null *)
46         {{ lessons[i]['czwartek'].klasa.poziom }}{{ lessons[i]['czwartek'].klasa.klasa }} -
47         {{ lessons[i]['czwartek'].ktoCo.przedmiot.nazwa }}
48         (* endif *)
49       </td>
50       <td>
51         (* if lessons[i]['piatek'] is not null *)
52         {{ lessons[i]['piatek'].klasa.poziom }}{{ lessons[i]['piatek'].klasa.klasa }} -
53         {{ lessons[i]['piatek'].ktoCo.przedmiot.nazwa }}
54         (* endif *)
55       </td>
56     </tr>
57   </tbody>
58 </table>
59 (* endif *)
60 (* endblock *)
```


Implementacja systemu

Oceny

Obecności

Plan zajęć

Uwagi i pochwały

Zalogowany jako: U01MaNo

Plan zajęć

Godz.	Poniedziałek	Wtorek	Środa	Czwartek	Piątek
08:00 - 08:45	1a - Matematyka			1a - Wychowanie fizyczne	
08:55 - 09:40	1a - Polski	1a - Niemiecki	1a - Matematyka	1a - Wychowanie fizyczne	
09:40 - 10:25	1a - Informatyka	1a - Matematyka	1a - Matematyka	1a - Polski	1a - Matematyka
10:35 - 11:20	1a - Informatyka	1a - Geografia	1a - Geografia	1a - Polski	1a - Matematyka
11:30 - 12:15	1a - Biologia	1a - Biologia	1a - Matematyka	1a - Polski	1a - Polski
12:25 - 13:10	1a - Niemiecki				1a - Geografia
13:20 - 14:05					

Dziennik elektroniczny

Rysunek 3.2. Widok planu zajęć

4. Testy aplikacji

Przeprowadzone testy zostały zrealizowane w celu sprawdzenia poprawności działania aplikacji. Wykonane testy zostały podzielone na dwa główne typy: testy funkcjonalne oraz testy jednostkowe. Testy funkcjonalne wykonywane były przy bezpośrednim kontakcie użytkownika z interfejsem użytkownika, natomiast testy jednostkowe przeprowadzone były automatycznie przez skrypty testujące po ich uruchomieniu przez programistę. Zaletą przeprowadzonych testów jednostkowych jest możliwość zautomatyzowania procedury testowej oraz wykrycia błędów w kodzie na wczesnym etapie implementacji.

Ze względu na dużą złożoność systemu oraz liczne funkcje, w pracy przedstawiono jedynie wybrane testy prezentujące poprawne działanie kluczowych funkcji systemu. Zrezygnowano z prezentowania testów prowadzonych według scenariuszy alternatywnych oraz sytuacji wyjątkowych. Testy te zostały przeprowadzone a wyniki wszystkich testów były pozytywne.

4.1. Testy funkcjonalne

Testy funkcjonalne zostały zrealizowane tak by przetestować kompletną funkcjonalność systemu. Ze względu na dużą objętość testów w pracy przedstawiono tylko najważniejsze przypadki testowe. Każdy ze wspomnianych przypadków testowych opracowano według następującego scenariusza testowego:

- a) nazwa testowanej funkcjonalności,
- b) stan początkowy aplikacji, warunki wstępne i dane wejściowe,
- c) oczekiwane rezultaty,
- d) kroki wykonania testu,
- e) uzyskany rezultat,
- f) podsumowanie testu.

Testy aplikacji

4.1.1. Podstrona terminarza

Kluczową funkcją podstrony terminarza jest dodawanie terminów lekcji do planu zajęć. Funkcja ta została przetestowana poprzez zrealizowanie scenariuszy 4.1 oraz 4.2.

Scenariusz testowy 4.1:

- a) nazwa testowanej funkcjonalności: dodawanie nowego terminu,
- b) stan początkowy aplikacji: zalogowany użytkownik posiadający rolę administratora, otworzony formularz dodawania terminarza (Rys. 4.1),
warunki wstępne : w bazie danych istnieją klasa, nauczyciel oraz przedmiot,
dane wejściowe: czas początku i końca zajęć, dzień tygodnia, okres w jakim zajęcia będą się cyklicznie powtarzały, sala, nauczyciel, przedmiot, typ zdarzenia, opis,
- c) oczekiwane rezultaty: dodany nowy termin,
- d) kroki wykonania testu:
użytkownik wybiera: klasę, nauczyciela, przedmiot, salę,
użytkownik wprowadza: czas początku i końca zajęć, dzień tygodnia, okres w jakim zajęcia będą się cyklicznie powtarzały,
użytkownik klika przycisk „Wyślij”
- e) uzyskany rezultat: termin dodany, wyświetlone potwierdzenie dodania terminu, formularz dodawania terminu wyczyszczony (Rys. 4.2), na formularzu prezentującym plan lekcji widoczna jest nowa lekcja (Rys. 4.3),
- f) podsumowanie testu: test zakończony pomyślnie.

Testy aplikacji

Dodawanie terminu

Sala	1	⌵
Godzina	09:40	⌵
Dzień tygodnia	Piątek	⌵
Przedmiot prowadzący	Fizyka Arnold Wróbel	⌵
Klasa	1a	⌵
Typ	Plan	⌵
Początek	01.10.2018	✖
Koniec	31.10.2018	✖
Opis		
<button>Wyślij</button>		

Rysunek 4.1. Formularz przed dodaniem terminu

Dodawanie terminu

Sala	1	⌵
Godzina	08:00	⌵
Dzień tygodnia	Poniedziałek	⌵
Przedmiot prowadzący	Matematyka Grzegorz Adamczuk	⌵
Klasa	1a	⌵
Typ	Plan	⌵
Początek	dd.mm.rrrr	
Koniec	dd.mm.rrrr	
Opis		
<button>Wyślij</button>		

Dodano nowy termin do bazy.

Rysunek 4.2. Dodawanie terminu

Testy aplikacji

Klasa 1a

Godz.	Poniedziałek	Wtorek	Środa	Czwartek	Piątek	Sobota
08:00 - 08:45	Matematyka			Wychowanie fizyczne		
08:55 - 09:40	Polski	Niemiecki	Matematyka	Wychowanie fizyczne		
09:40 - 10:25	Informatyka	Fizyka	Matematyka	Polski	Fizyka	
10:35 - 11:20	Informatyka	Geografia	Geografia	Polski	Matematyka	
11:30 - 12:15	Biologia	Biologia	Matematyka	Polski	Polski	
12:25 - 13:10	Niemiecki				Geografia	

Info: w przypadku dwóch lub więcej terminów dla danej klasy, dnia tygodnia oraz godziny, wyświetli się najstarszy termin dla danego miejsca.

Rysunek 4.3. Nowy przedmiot na planie lekcji

Scenariusz testowy 4.2:

- a) nazwa testowanej funkcjonalności: dodawanie nowego terminu,
- b) stan początkowy aplikacji: użytkownik posiadający rolę administratora zalogowany, otworzony formularz dodawania terminarza (Rys. 4.1),
warunki wstępne : w bazie danych istnieją: klasa, nauczyciel oraz przedmiot,
dane wejściowe: błędnie podana jedna lub wiele z wymienionych wartości: czas początku i końca zajęć, dzień tygodnia, okres w jakim zajęcia będą się cyklicznie powtarzały, sala, nauczyciel, przedmiot, typ zdarzenia, opis,
- c) oczekiwane rezultaty: komunikat informujący użytkownika o rodzaju popełnionego błędu, brak dodania nowego terminu, formularz dodawania nowego terminu wypełniony dotychczasowymi danymi umożliwiającymi poprawienie błędnych wpisów,
- d) kroki wykonania testu:
użytkownik wybiera: klasę, nauczyciela, przedmiot, salę,
użytkownik wprowadza: czas początku i końca zajęć, dzień tygodnia, okres w jakim zajęcia będą się cyklicznie powtarzały,
użytkownik klika przycisk „Wyślij”,

Testy aplikacji

- e) uzyskany rezultat: brak dodania terminu, wyświetlenie komunikatu błędu poniżej wypełnionego formularza dodawania nowego terminu (Rys. 4.4),
- f) podsumowanie testu: test zakończony pomyślnie.

Dodawanie terminu

Sala	1	⌵
Godzina	09:40	⌵
Dzień tygodnia	Piątek	⌵
Przedmiot prowadzący	Fizyka Arnold Wróbel	⌵
Klasa	1a	⌵
Typ	Plan	⌵
Początek	31.10.2018	✖
Koniec	01.10.2018	✖
Opis		
<button>Wyślij</button>		
Koniec nie może być wcześniejszy od początku.		

Rysunek 4.4. Formularz po wysłaniu błędnych danych

4.1.2. Podstrona logowania

Poprawnie działające uwierzytelnianie jest kluczowe w bezpiecznym działaniu aplikacji. Z tego względu przeprowadzono testy poprawności działania logowania według następujących scenariuszy testowych 4.3, 4.4 oraz 4.5.

Scenariusz testowy 4.3:

- a) nazwa testowanej funkcjonalności: logowanie do aplikacji,
- b) stan początkowy aplikacji: użytkownik posiadający uprawnienia administratora znający swój login i hasło, otworzony formularz logowania,
warunki wstępne : w bazie istnieją: administratora
dane wejściowe: poprawny login i hasło administratora
- c) oczekiwane rezultaty: zalogowanie do aplikacji,

Testy aplikacji

- d) kroki wykonania testu: użytkownik wprowadza poprawny login oraz hasło (Rys. 4.5),
- e) uzyskany rezultat: użytkownik zalogowany, wyświetlony komunikat o poprawnym zalogowaniu (Rys. 4.6),
- f) podsumowanie testu: test zakończony pomyślnie.

The screenshot shows a web application interface for a 'Dziennik elektroniczny' (Electronic Journal). At the top, there is a dark blue header with a home icon on the left and a user icon on the right. Below the header is a large blue banner. In the center of the banner is a white box titled 'Logowanie' (Login). Inside this box, there are two input fields: 'Login' with the text 'admin' and 'Hasło' (Password) with masked characters '.....'. Below these fields is a blue button labeled 'Wyślij' (Send). At the bottom of the page, there is a dark blue footer with the text 'Dziennik elektroniczny'.

Rysunek 4.5. Formularz logowania

The screenshot shows the main page of the 'Dziennik elektroniczny' application after a successful login. The top dark blue header now includes a navigation menu with buttons: 'Użytkownicy', 'Klasy', 'Klasa', 'Przedmioty', 'Godziny lekcyjne', 'Sale', 'Plan zajęć', 'Ustawienia', 'Sprawdzanie obecności', 'Wstawienie oceny', 'Spóźnienia', 'Plan zajęć', 'Uwagi i pochwały', 'Obecności', and 'Oceny'. On the right side of the header, it says 'Zalogowany jako: admin' with a user icon and a notification icon showing '(1)'. Below the header is a large blue banner with the title 'Strona główna' (Main Page). In the center of the banner is a light blue box with the text 'Zalogowano.' (Logged in.). At the bottom of the page, there is a dark blue footer with the text 'Dziennik elektroniczny'.

Rysunek 4.6. Logowanie zakończone sukcesem

Testy aplikacji

Scenariusz testowy 4.4:

- a) nazwa testowanej funkcjonalności: logowanie do aplikacji,
- b) stan początkowy aplikacji: użytkownik nie zna loginu lub hasła lub jednocześnie loginu i hasła,
warunki wstępne: w bazie istnieją: administrator
dane wejściowe: nieprawidłowy login i/lub hasło
- c) oczekiwane rezultaty: komunika informujący użytkownika o wprowadzeniu niepoprawnych danych autoryzacyjnych,
- d) kroki wykonania testu: użytkownik wprowadza niepoprawne dane autoryzacyjne (Rys. 4.5),
- e) uzyskany rezultat: logowanie wstrzymane, wyświetlenie komunikatu poniżej wypełnionego formularza logowania (Rys. 4.7),
- f) podsumowanie testu: test zakończony pomyślnie.

The screenshot shows a web application interface for logging in. At the top, there is a dark blue header bar with a home icon on the left and a right arrow icon on the right. Below the header is a large blue rectangular area. In the center of this area is a white box with the title "Logowanie" in bold black text. Inside the white box, there are two input fields: the first is labeled "Login" and contains the text "admin"; the second is labeled "Hasło" and contains six dots. Below these fields is a blue button labeled "Wyślij". At the bottom of the white box, there is a pink error message that reads "Błędny login lub hasło." Below the white box, there is a dark blue footer bar with the text "Dziennik elektroniczny" in white.

Rysunek 4.7. Logowanie zakończone niepowodzeniem

Testy aplikacji

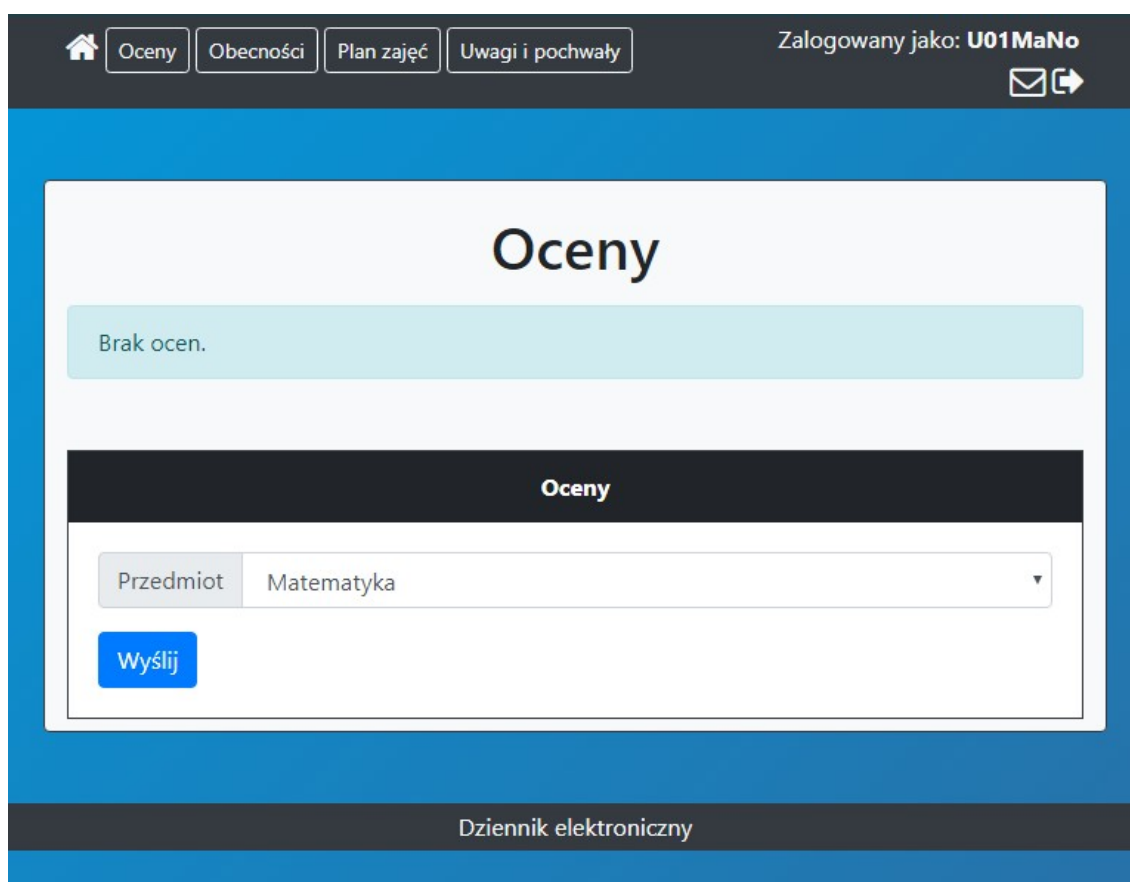
Scenariusz testowy 4.5:

- a) nazwa testowanej funkcjonalności: logowanie do aplikacji,
- b) stan początkowy aplikacji: użytkownik zalogowany (Rys. 4.8), użytkownik posiada uprawnienia ucznia, użytkownik nie posiada uprawnień administratora, użytkownik zna adres panelu administratora,

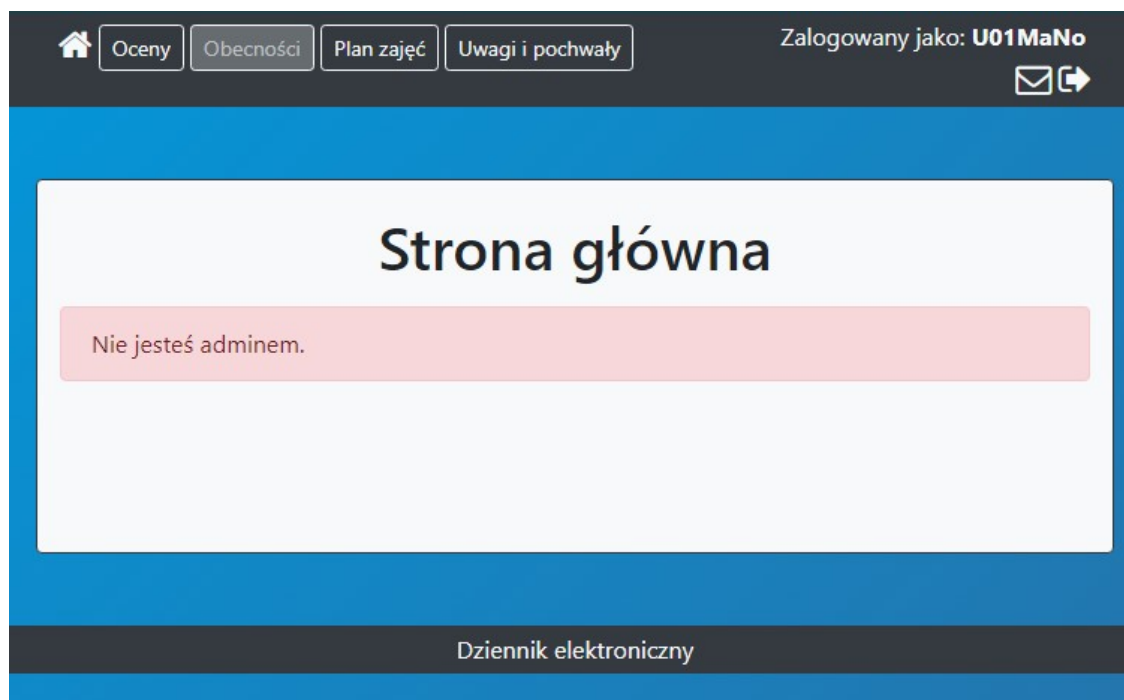
warunki wstępne : w bazie danych istnieją: uczeń, administrator

dane wejściowe: zalogowany uczeń
- c) oczekiwane rezultaty: niedopuszczenie użytkownika do panelu administratora, komunikat informujący użytkownika o braku uprawnień,
- d) kroki wykonania testu: użytkownik w pasku adresu przeglądarki wprowadza adres panelu administratora: <http://localhost:8000/admin/uzytkownicy#>,
- e) uzyskany rezultat: użytkownik nie zostaje przekierowany do panelu administratora, wyświetla się komunikat o braku odpowiednich uprawnień (Rys. 4.9),
- f) podsumowanie testu: test zakończony pomyślnie.

Testy aplikacji



Rysunek 4.8. Zalogowany użytkownik o uprawnieniach ucznia



Rysunek 4.9. Zabezpieczenie funkcji administratora

4.2. Testy jednostkowe

Testy jednostkowe zostały zrealizowane za pomocą klas frameworka Symfony 3. Dla każdej z klas kontrolera i modelu opracowano niezależną klasę testów jednostkowych sprawdzającą kluczowe metody danej klasy. Przykładowy kod klasy testów przedstawiono na Listingu 4.1.

Listing 4.1. Klasa testów "Test Terminarza"

```
//zalogowany admin z parametrem dodawanie
//testujemy czy zwróci nam strona odpowiedni kod HTTP poprawność tekstu tagi h2
public function testTimeTableAction() {
    $client=static::createClient();

    $session=$client->getContainer()->get('session');
    $session->set('admin',true);
    $session->save();
    $crawler=$client->request('GET','/admin/terminarz');

    $this->assertEquals(200,$client->getResponse()->getStatusCode());
    $this->assertContains('Dodawanie terminu',$crawler->filter('#container h2')->text());
}

//zalogowany admin z parametrem edycja
//testujemy czy zwróci nam strona odpowiedni kod HTTP poprawność tekstu tagi h2
public function test2TimeTableAction() {
    $client=static::createClient();

    $session=$client->getContainer()->get('session');
    $session->set('admin',true);
    $session->save();
    $crawler=$client->request('GET','/admin/terminarz/0/0/1');

    $this->assertEquals(200,$client->getResponse()->getStatusCode());
    $this->assertContains('Edytowanie terminu',$crawler->filter('#container h2')->text());
}

//zalogowany admin z parametrem klasa 1a
//testujemy czy zwróci nam strona odpowiedni kod HTTP poprawność tekstu tagi h2
public function test3TimeTableAction() {
    $client=static::createClient();

    $session=$client->getContainer()->get('session');
    $session->set('admin',true);
    $session->save();
    $crawler=$client->request('GET','/admin/terminarz/1/a');

    $this->assertEquals(200,$client->getResponse()->getStatusCode());
    $this->assertContains('Klasa 1a',$crawler->filter('#container h2')->text());
}

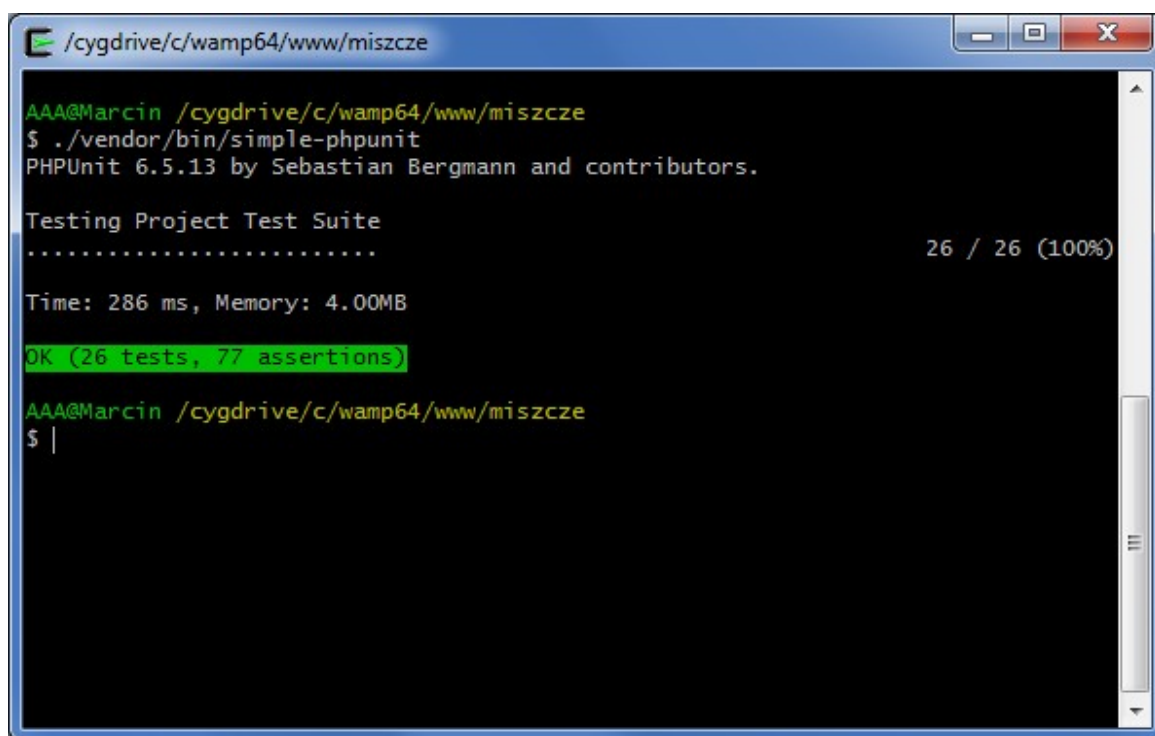
//niezalogowany admin
//testujemy czy zwróci nam strona odpowiedni kod HTTP poprawność tekstu tagi h2
public function test4TimeTableAction() {
    $client=static::createClient();

    $client->request('GET','/admin/terminarz');

    $this->assertEquals(302,$client->getResponse()->getStatusCode());
}
```

Testy aplikacji

Opracowane testy były uruchamiane cyklicznie po wprowadzeniu kolejnych zmian do kodu aplikacji. W przypadku wystąpienia błędów były one eliminowane tak, by testy kończyły się ostatecznie z wynikiem pozytywnym. Po opracowaniu całości aplikacji uruchomiono testy ponownie w celu sprawdzenia czy wszystkie kluczowe metody działają poprawnie. Zrzut ekranu prezentujący wynik wykonania testów został zaprezentowany na Rysunku 4.10.



```
/cygdrive/c/wamp64/www/miszcze
AAA@Marcin /cygdrive/c/wamp64/www/miszcze
$ ./vendor/bin/simple-phpunit
PHPUnit 6.5.13 by Sebastian Bergmann and contributors.

Testing Project Test Suite
.....
26 / 26 (100%)

Time: 286 ms, Memory: 4.00MB

OK (26 tests, 77 assertions)

AAA@Marcin /cygdrive/c/wamp64/www/miszcze
$ |
```

Rysunek 4.10. Wyniki wykonania testów jednostkowych

5. Wnioski

W wyniku przeprowadzonych prac stworzono kompletny system dziennika elektronicznego. Wspomniany system powstał na podstawie wiedzy uzyskanej od specjalistów z dziedziny edukacji a jego zadaniem jest wspomaganie pracy placówek edukacyjnych. Powszechny i łatwy dostęp do systemu zapewniono przez udostępnienie interfejsu użytkownika poprzez przeglądarkę w postaci dynamicznej strony WWW. Stworzono również interfejsy przeznaczone dla wszystkich grup użytkowników.

Opracowana aplikacja została przetestowana. W trakcie tworzenia aplikacji systematycznie tworzone i wykonywano testy jednostkowe, co pozwoliło na szybkie identyfikowanie błędów a co za tym idzie sprawne i tanie ich usuwanie. Finalna wersja systemu została poddana testom manualnym weryfikującym kompletność systemu oraz poprawność działania wszystkich funkcji.

Cel pracy został w pełni zrealizowany poprzez zaprojektowanie, zaimplementowanie oraz przetestowanie systemu dziennika elektronicznego. Opracowany system jest gotowym narzędziem, które można wdrożyć w dowolnej placówce edukacyjnej, w celu usprawnienia jej pracy. Rozwiązanie prezentowane w pracy jest skalowalne i posiada możliwości rozbudowy. Planowany jest dalszy rozwój prezentowanego dziennika elektronicznego poprzez dodanie do niego modułu raportującego.

Bibliografia

- [1] Bowers Michael. Synodinos Dionysios. Sumner Victor, HTML5 i CSS3. Zaawansowane wzorce projektowe, Helion, Gliwice 2012
- [2] Drejewicz Szymon, Zrozumieć BPMN. Modelowanie procesów biznesowych, Helion, Gliwice 2012
- [3] DuBois Paul, MySQL. Vademecum profesjonalisty, Helion, Gliwice 2014
- [4] Gajda Włodzimierz, PHP, MySQL i MVC. Tworzenie witryn WWW opartych na bazie danych, Helion, Gliwice 2010
- [5] Gajda Włodzimierz, Symfony 2 od podstaw, Helion, Gliwice 2012
- [6] Kortas Michał, Bootstrap. Praktyczne projekty, Helion, Gliwice 2016
- [7] <https://www.librus.pl> [dostęp elektroniczny dnia 29.05.2018r]
- [8] <http://www.php.pl/Wortal/Artykuly/Proces-tworzenia-aplikacji/Tlumaczenia/Projektowanie-Kontrolera/Angielskie-nazwy-wzorcow> [dostęp elektroniczny dnia 20.10.2018]
- [9] <http://prawo.sejm.gov.pl/isap.nsf/download.xsp/WDU20180001000/T/D20181000L.pdf>
- [10] <https://www.vulcan.edu.pl> [dostęp elektroniczny dnia 29.05.2018r]
- [11] <https://rk.edu.pl/pl/co-jest-php/> [dostęp elektroniczny dnia 15.01.2019]

Spis rysunków

Rysunek 2.1. Diagram przypadków użycia.....	13
Rysunek 2.2. Sprawdzanie ocen.....	14
Rysunek 2.3. Wstawianie ocen.....	15
Rysunek 2.4. Korekta ocen.....	15
Rysunek 2.5. Ocenianie nauczycieli.....	16
Rysunek 2.6. Sprawdzanie obecności.....	16
Rysunek 2.7. Kontrola nieobecności.....	17
Rysunek 2.8. Oceny semestralne.....	18
Rysunek 2.9. Tworzenie planu zajęć.....	18
Rysunek 2.10. Tworzenie klas.....	19
Rysunek 2.11. Tworzenie kont.....	20
Rysunek 2.12. Wstawianie uwag.....	21
Rysunek 2.13. Dodawanie terminów sprawdzianów.....	22
Rysunek 2.14. Przebieg procesu logowania.....	23
Rysunek 2.15. Pisanie wiadomości.....	24
Rysunek 2.16. Usprawiedliwianie online.....	24
Rysunek 2.17. Relacyjna baza danych.....	25
Rysunek 3.1. Struktura projektu aplikacji.....	31
Rysunek 3.2. Widok planu zajęć.....	33
Rysunek 4.1. Formularz przed dodaniem terminu.....	36
Rysunek 4.2. Dodawanie terminu.....	36
Rysunek 4.3. Nowy przedmiot na planie lekcji.....	37
Rysunek 4.4. Formularz po wysłaniu błędnych danych.....	38
Rysunek 4.5. Formularz logowania.....	39
Rysunek 4.6. Logowanie zakończone sukcesem.....	39
Rysunek 4.7. Logowanie zakończone niepowodzeniem.....	40
Rysunek 4.8. Zalogowany użytkownik o uprawnieniach ucznia.....	42
Rysunek 4.9. Zabezpieczenie funkcji administratora.....	42
Rysunek 4.10. Wyniki wykonania testów jednostkowych.....	44

Spis listingów

Listing 3.1. Uchwyty strony umożliwiające operacje na godzinach lekcyjnych.....	26
Listing 3.2. Formularz dodawania godziny lekcyjnej.....	27
Listing 3.3. Fragment kodu realizujący dodawanie nowej godziny lekcyjnej.....	27
Listing 3.4. Kod umożliwiający usuwanie godziny lekcyjnej.....	27
Listing 3.5. Kod zwracający widok, który prezentuje listę godzin lekcyjnych.....	27
Listing 3.6. Usuwanie sesji użytkownika.....	28
Listing 3.7. Encja do tabeli GodzLek.....	29
Listing 3.8. Usługa message.....	30
Listing 3.9. Widok time_table.....	32
Listing 4.1. Klasa testów "Test Terminarza".....	43