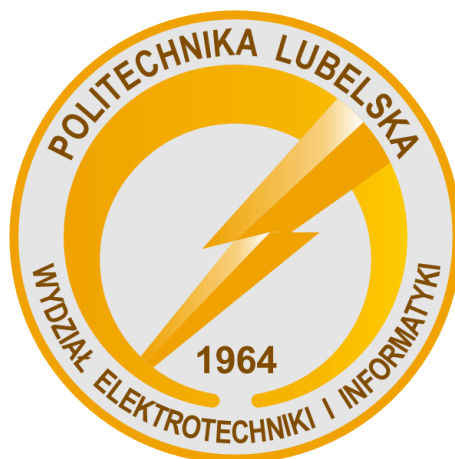


POLITECHNIKA LUBELSKA

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„ 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. Podział prac.....	7
Rozdział 2. Projekt Systemu.....	9
2.1. Wymagania funkcjonalne i нефункционалне.....	9
2.2. Diagram przypadków użycia.....	12
2.3. Diagramy BPMN.....	13
2.4. Diagramy sekwencji.....	19
2.5. Diagramy aktywności.....	22
2.6. Diagram EER.....	24
Rozdział 3. Implementacja systemu.....	25
3.1. Kontrolery.....	25
3.2. Encje.....	27
3.3. Usługi.....	29
3.4. Struktura widoków.....	30
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.....	42
Rozdział 5. Wnioski.....	45
Rozdział Bibliografia.....	46
Rozdział Spis rysunków.....	47
Rozdział Spis listingów.....	48

1. Wstęp

Rozpowszechnienie się dostępu do internetu oraz szybki rozwój technologiczny 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 cosemestralnych 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”[LIT1] i „Vulcan” [LIT2], których dostępność sprawiła, że szybko stały się monopolistami na polskim rynku. Obie wspomniane aplikacje posiadają podobne funkcjonalności. 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 inne funkcjonalności: 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 RODO.

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

1.1. Cel i zakres pracy

Celem niniejszej pracy jest stworzenie dziennika elektronicznego, który będzie ułatwiał pracę nauczycieli oraz komunikację uczniów i opiekunów ucznia ze szkołą oraz umożliwiał ocenę nauczycieli po skończonym 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 użytkownika z istniejących dzienników elektronicznych,
4. zaprojektowanie diagramów,
5. utworzenie bazy danych,
6. wykonanie strony obsługującej dziennik elektroniczny,
7. testy aplikacji.

Celem niniejszej pracy inżynierskiej było zaprojektowanie i stworzenie dziennika elektronicznego zgodnego z wymaganiami RODO. Praca składa się z pięciu rozdziałów. Dziennik elektroniczny ma na celu ułatwić komunikację pomiędzy szkołą a opiekunem, usprawnić pracę nauczyciela oraz umożliwić opiekunom stałą obserwację postępów uczniów. 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 model-widok-kontroler (ang. Model-View-Controller – MVC). Interfejs użytkownika dostępny jest przez przeglądarkę Google chrome i Mozilla Firefox. 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 piątym rozdziale.

Wstęp

The aim of this engineering project was to design and create an electronic journal in line with the requirements of the GDPR. The work consists of five chapters. The e-journal aims to facilitate communication between the school and the guardian, improve the work of the teacher and enable the guardians to constantly monitor the progress of the students. 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 Google Chrome and Mozilla Firefox. 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 fifth chapter.

Wstęp

1.2. Podział prac

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

Tabela 1.1. Podział prac

Zakres pracy			Krzysztof Barczak	Dariusz Głowacki	Marcin Górski
Wstęp			-	+	-
Cel i zakres prac			+	-	-
Projekt Systemu	Wymagania funkcjonalne i нефункционалне		+	-	-
	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	-	+	-
		Podstrona wstawianie oceny	+	-	-
		Podstrona wiadomości	+	-	-
	Testy jednostkowe		-	-	+
Wnioski			+	+	-

2. Projekt Systemu

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

2.1. Wymagania funkcjonalne i niefunkcjonalne

W niniejszym rozdziale przedstawiono wymagania funkcjonalne i niefunkcjonalne, które musi spełnić opracowywany w ramach pracy dziennik elektroniczny. Głównym zadaniem dziennika będzie wspomaganie zarządzaniem zajęciami w szkole.

W opracowywanym projekcie zdefiniowano następujące wymagania funkcjonalne:

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

Projekt Systemu

- f) odbierania uwag ucznia,
- 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) pisanie uwag ucznia,
- 4) wychowawca ma mieć możliwość:
- a) logowanie/wylogowanie,
 - b) usprawiedliwianie nieobecności,
 - c) wysyłanie/odbieranie wiadomości,
 - d) dodawania/edytowanie/usuwanie oceny z zachowania,
- 5) administrator/dyrektor ma mieć możliwość:
- a) dodawania uczniów/nauczycieli/rodziców,
 - b) edytowania uczniów/nauczycieli/rodziców,
 - c) usuwania uczniów/nauczycieli/rodziców,
 - d) przypisania/usunięcia ucznia z klasy,
 - e) dodania podziału klas,
 - f) obsługi ocen semestralnych oraz promocja do klasy,
 - g) zaliczenia/niezaliczenia roku,
 - h) obsługi planu zajęć,
 - i) oglądania ocen nauczycieli ocenianych przez uczniów.

Projekt Systemu

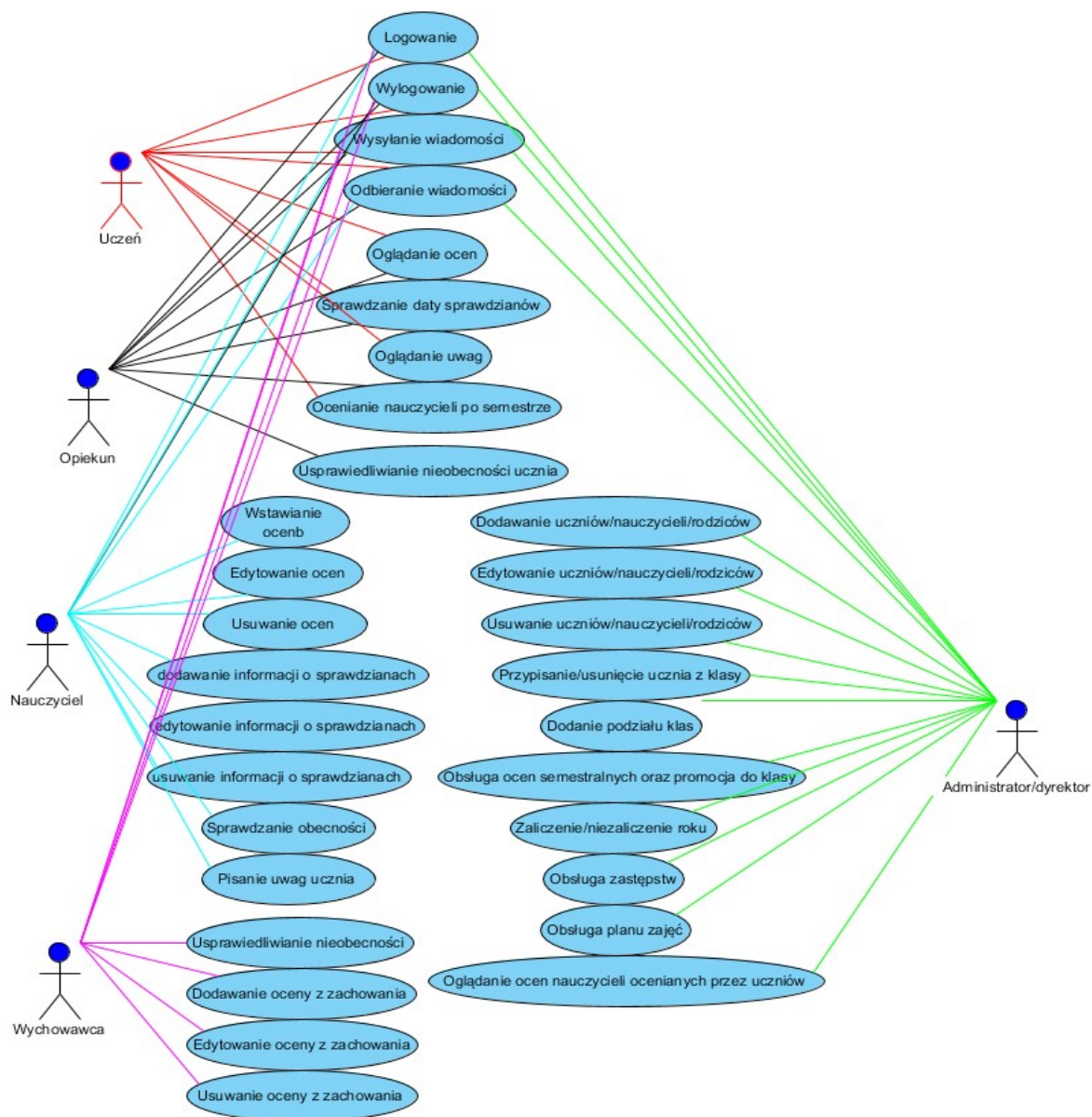
Ze względu na konieczność zapewnienia powszechnego dostępu do systemu użytkownikom dysponującym różnego rodzaju sprzętem i oprogramowaniem zdecydowano się na udostępnienie interfejsu użytkownika poprzez przeglądarkę.

W związku z tym zostały zdefiniowane następujące wymagania niefunkcjonalne:

- 1) aplikacja ma działać na przeglądarkach Google Chrome (wersja 65.0.3325.181 i nowsze) i Firefox (wersja 59 i nowsze) i przeglądarce mobilnej (android 5.0 i nowsze),
- 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

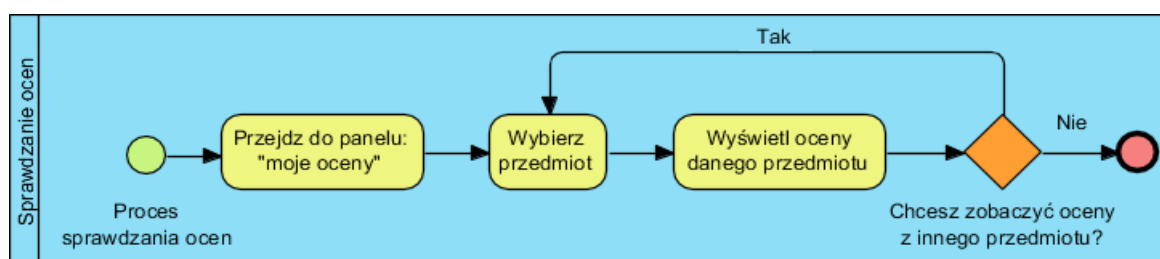
Dziennik elektroniczny może być używany przez różnych użytkowników, każdy z nich może wykonywać inne czynności. Na rysunku 2.1 przedstawiono diagramy przypadków użycia systemu.



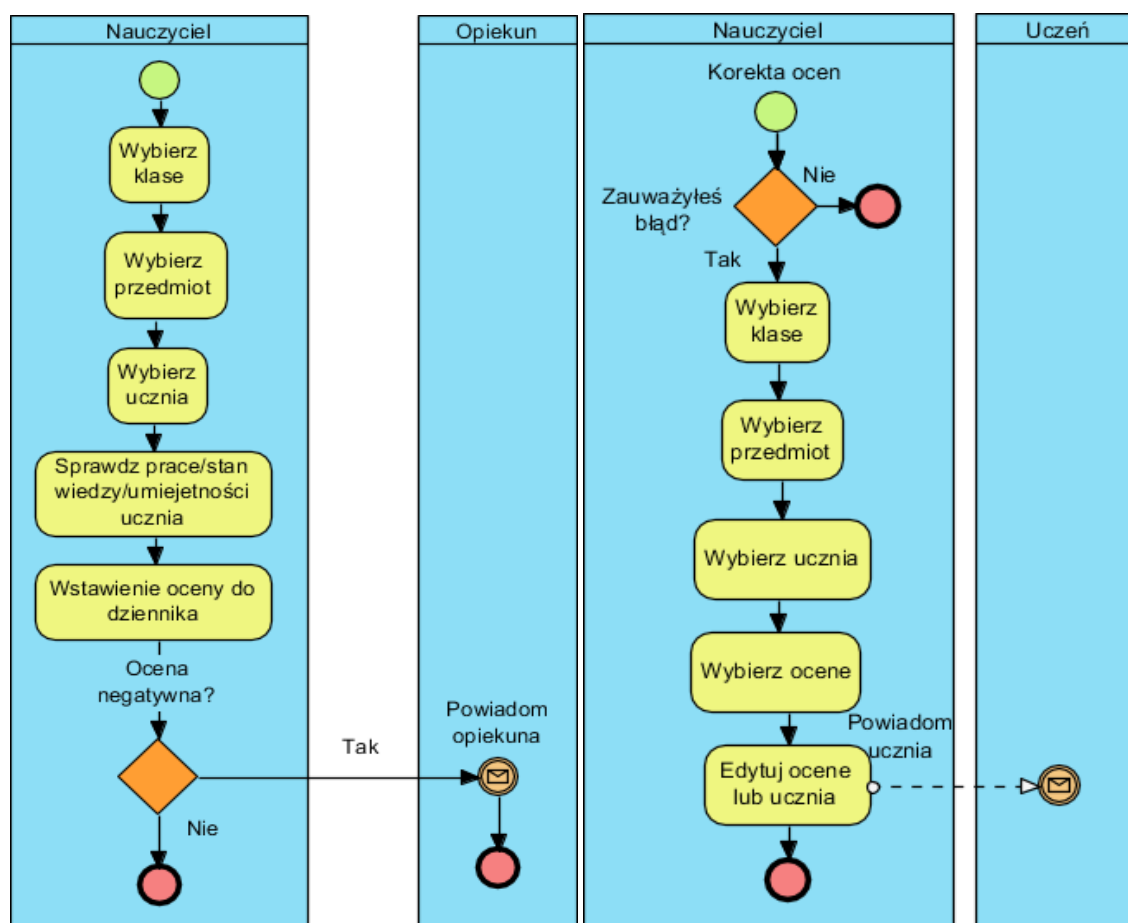
Rysunek 2.1. Diagram przypadków użycia

2.3. Diagramy BPMN

Opracowanie projektu obejmuje identyfikację procesów biznesowych zachodzących w szkole. Procesy te opisano za pomocą diagramów BPMN. Najważniejsze procesy zostały zobrazowane w niniejszym podrozdziale. Na rysunku 2.2 przedstawiono proces sprawdzania ocen widoczny z poziomu ucznia. Rysunek 2.3 przedstawia proces wystawiania ocen przez nauczyciela. Na rysunku 2.4 umieszczono diagram obrazujący proces poprawiania ocen wcześniej już wprowadzonych do systemu.



Rysunek 2.2. Sprawdzanie ocen

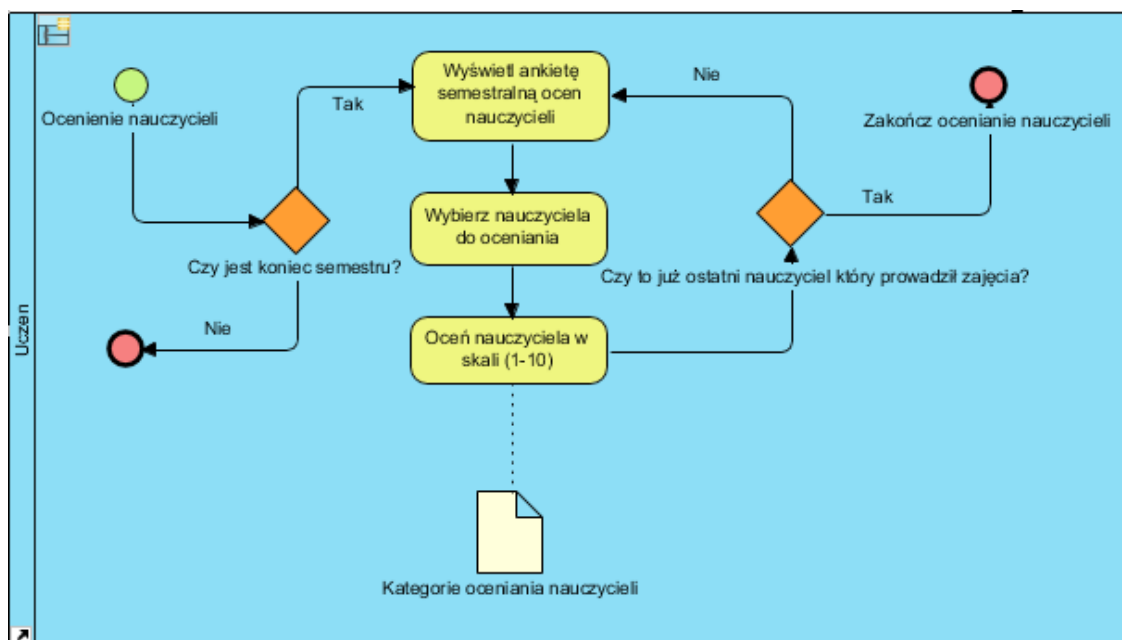


Rysunek 2.3. Wstawianie ocen

Rysunek 2.4. Korekta ocen

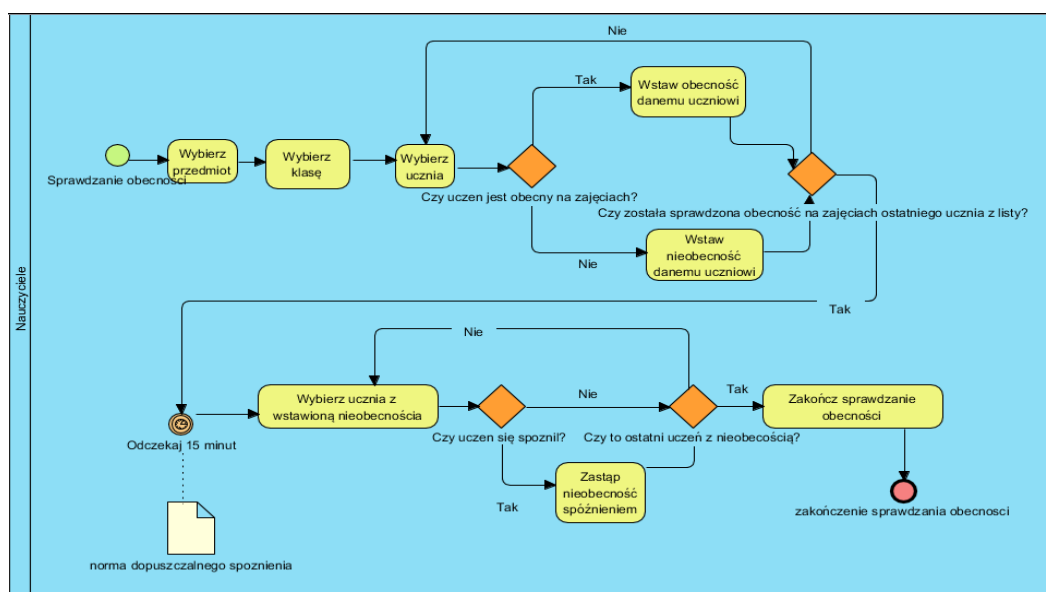
Projekt Systemu

Diagram przedstawiający ocenianie nauczycieli został przedstawiony na rysunku 2.5. Nasza praca wyróżnia się tym, że zaimplementowaliśmy ankietę pod koniec semestru, która będzie oceniać pracę nauczycieli za cały semestr.



Rysunek 2.5. Ocenianie nauczycieli

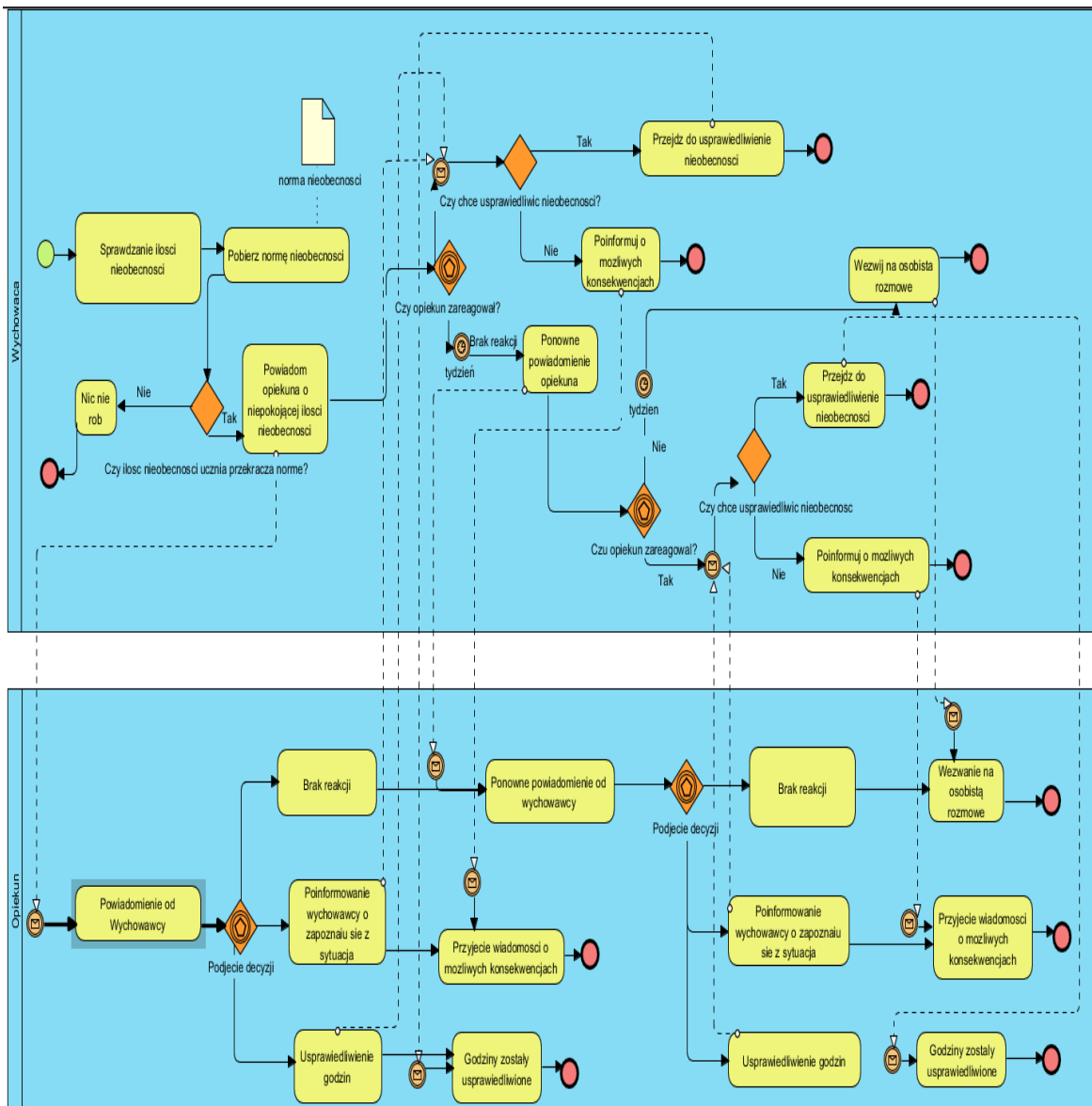
Częstym problem z którym borykają się nauczyciele jest kwestia nieobecności uczniów na zajęciach. Kontrola absencji uczniów została zobrazowana za pomocą diagramu BPMN na rysunku 2.6.



Rysunek 2.6. Sprawdzanie obecności

Projekt Systemu

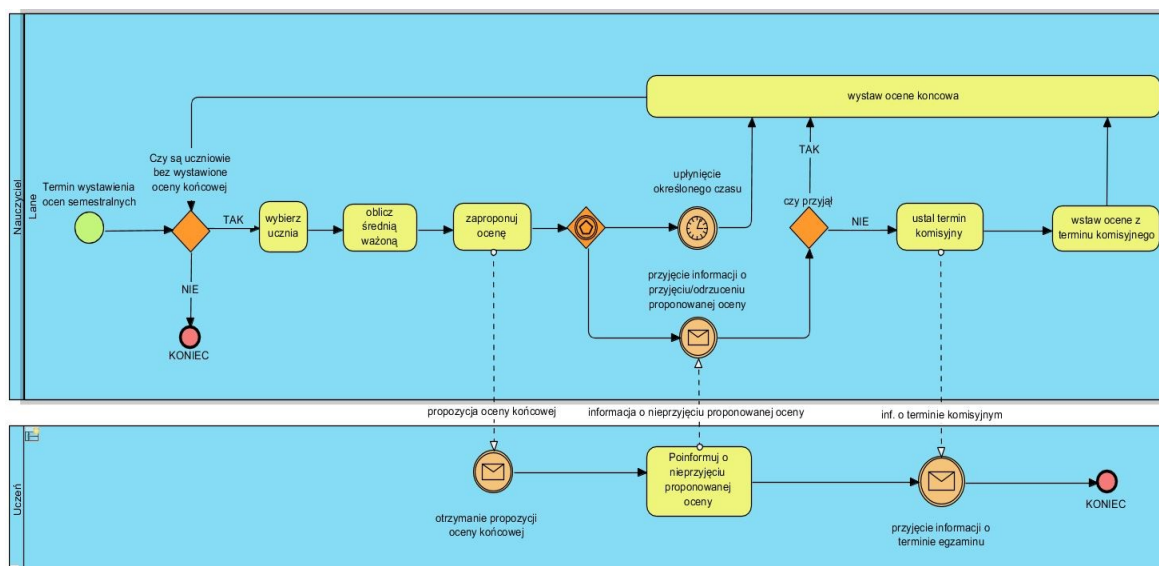
Diagram przedstawiający kontrole nieobecności przedstawia rysunek 2.7. Zaprezentowane są zdarzenia, co się dzieje w przypadku braku nieobecności ucznia.



Rysunek 2.7. Kontrola nieobecności

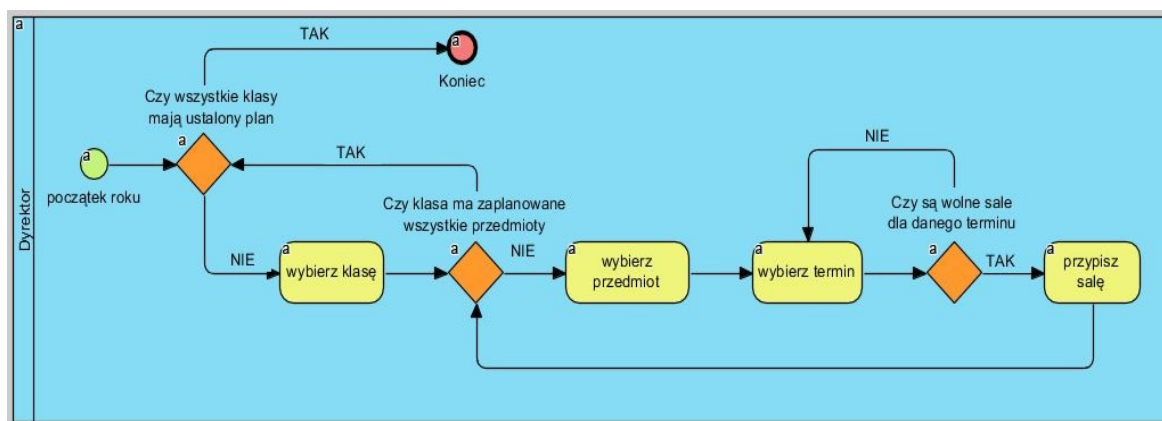
Projekt Systemu

Co semestr w szkołach jest wystawiana ocena podsumowująca pracę ucznia, dlatego zaimplementujemy taką funkcję. Diagram przedstawiający oceny semestralne ucznia umieszczono na rysunku 2.8.



Rysunek 2.8. Oceny semestralne

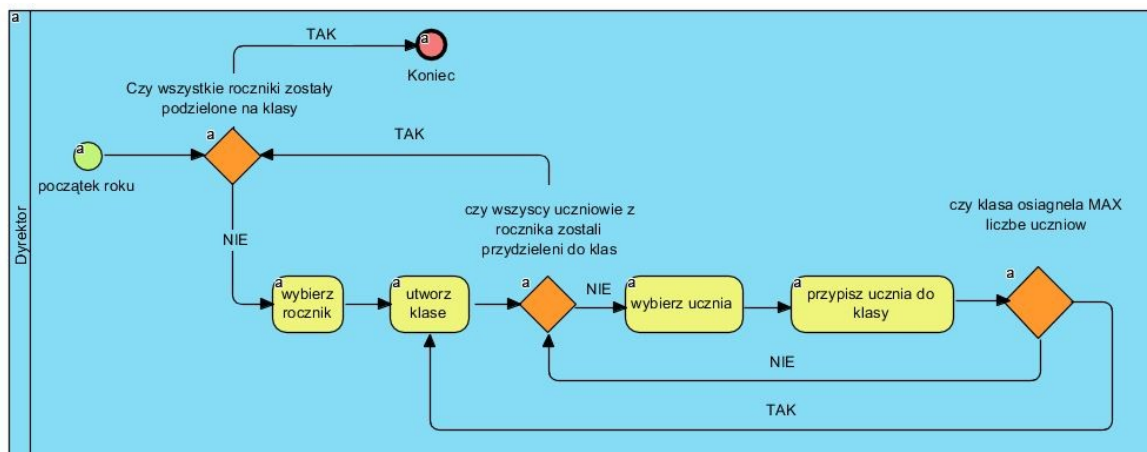
Praca nauczycieli i nauka uczniów musi być zorganizowana, żeby spotykali się w tym samym czasie podczas zajęć. Diagram przedstawiający tworzenie planu zajęć został zilustrowany na rysunku 2.9.



Rysunek 2.9. Tworzenie planu zajęć

Projekt Systemu

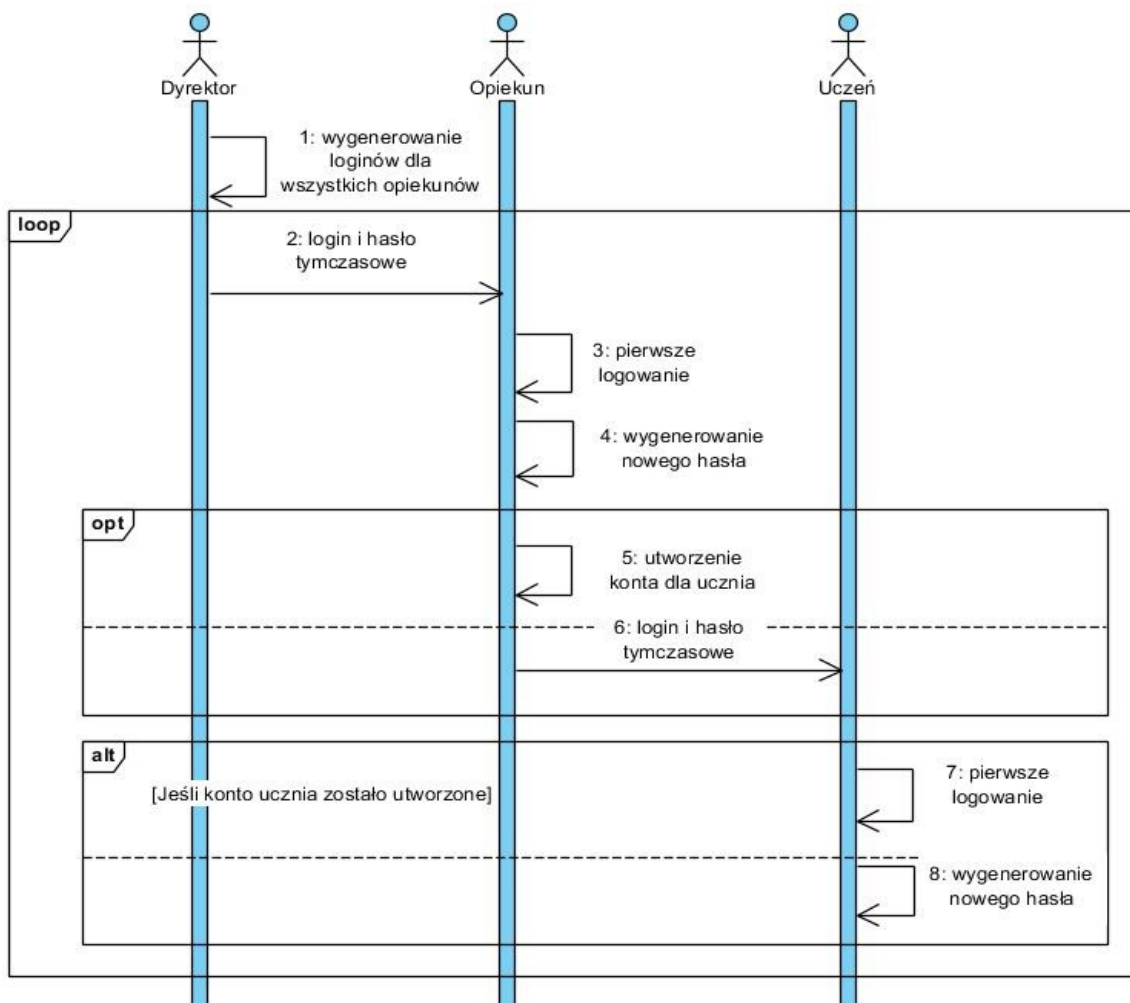
Uczniów jest zazwyczaj dużo i nie można prowadzić zajęć ze wszystkimi naraz, dlatego przypisuje się ich do poszczególnych wolnych klas. Diagram przedstawiający podział uczniów na klasy przedstawiono na rysunku 2.10.



Rysunek 2.10. Tworzenie klas

2.4. Diagramy sekwencji

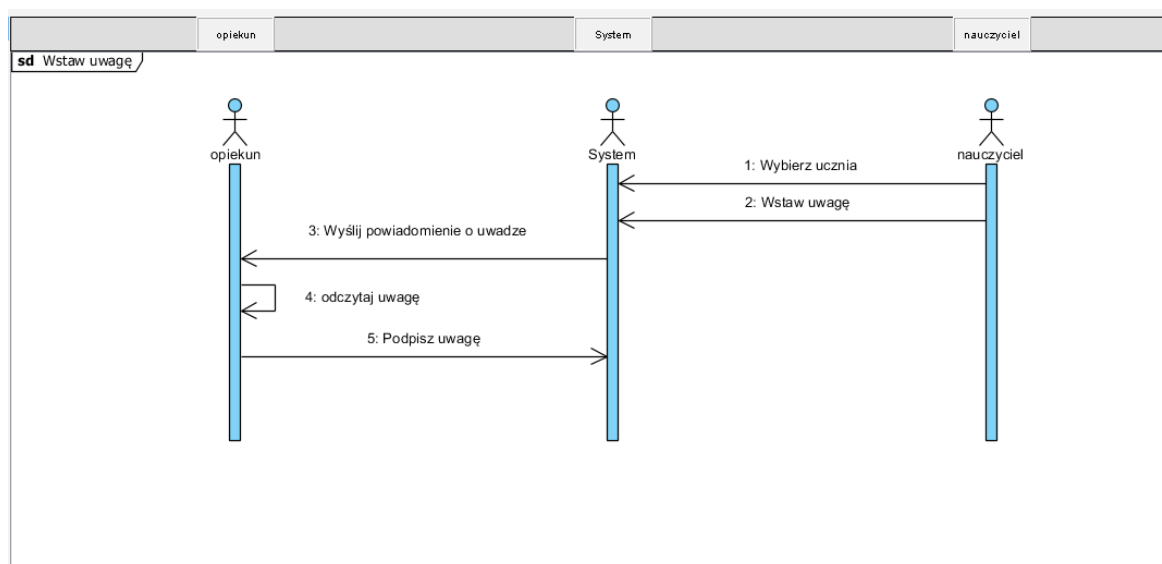
Diagramy sekwencji wspomagają zobrazowanie i identyfikację przepływu komunikatów wewnątrz systemu. Na rysunku 2.11 przedstawiono diagram sekwencji definiujący przepływ komunikatów w procesie tworzenia kont użytkowników.



Rysunek 2.11. Tworzenie kont

Projekt Systemu

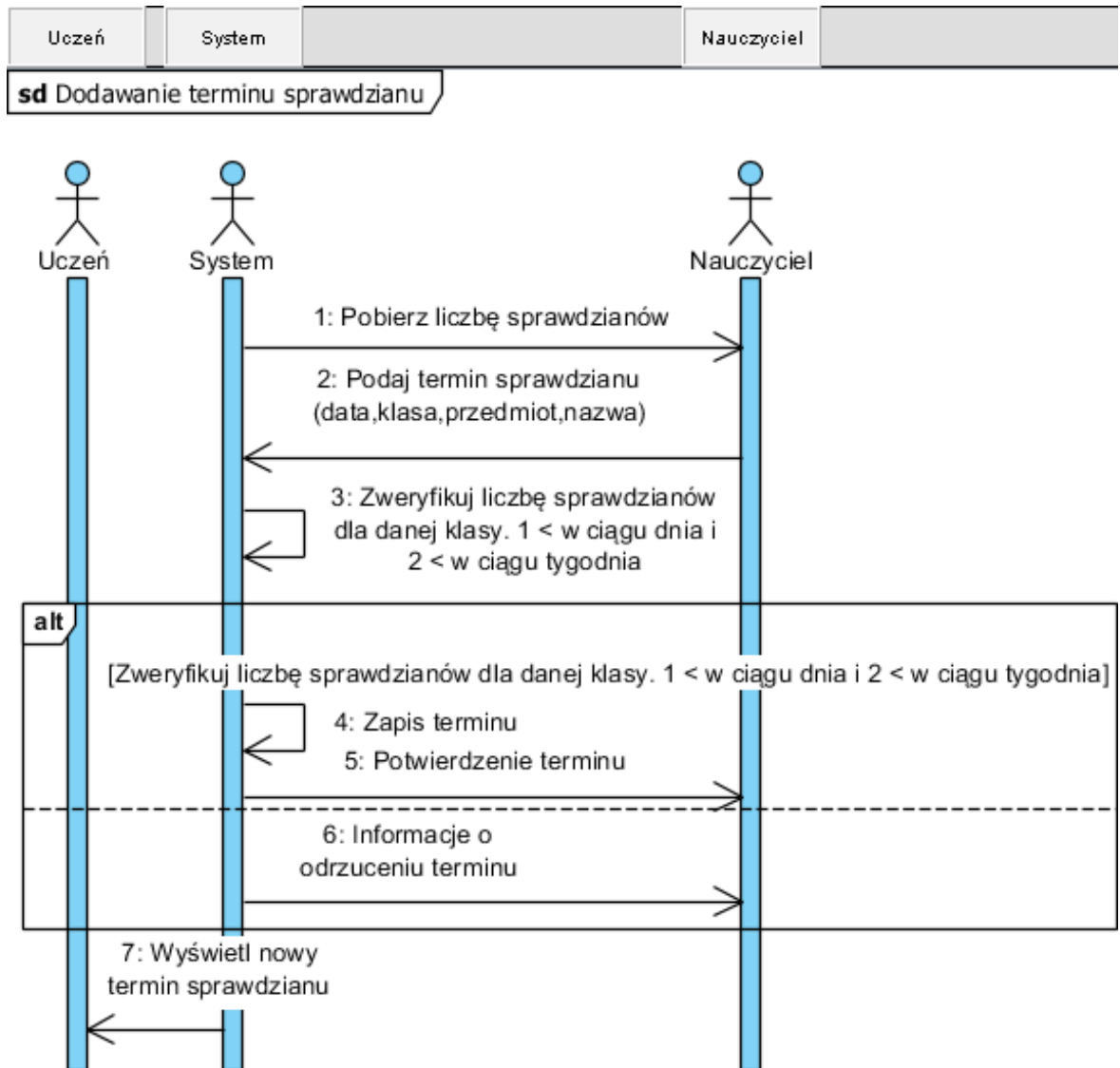
Za pośrednictwem systemu nauczyciel może kontaktować się z opiekunem ucznia. Rysunek 2.12 przedstawia przepływ komunikatu – uwagi od nauczyciela do opiekuna ucznia.



Rysunek 2.12. Wstawianie uwag

Projekt Systemu

Diagram sekwencji przedstawiony na rysunku 2.13 ilustruje przepływ komunikatów podczas procesu dodawania terminów sprawdzianów.

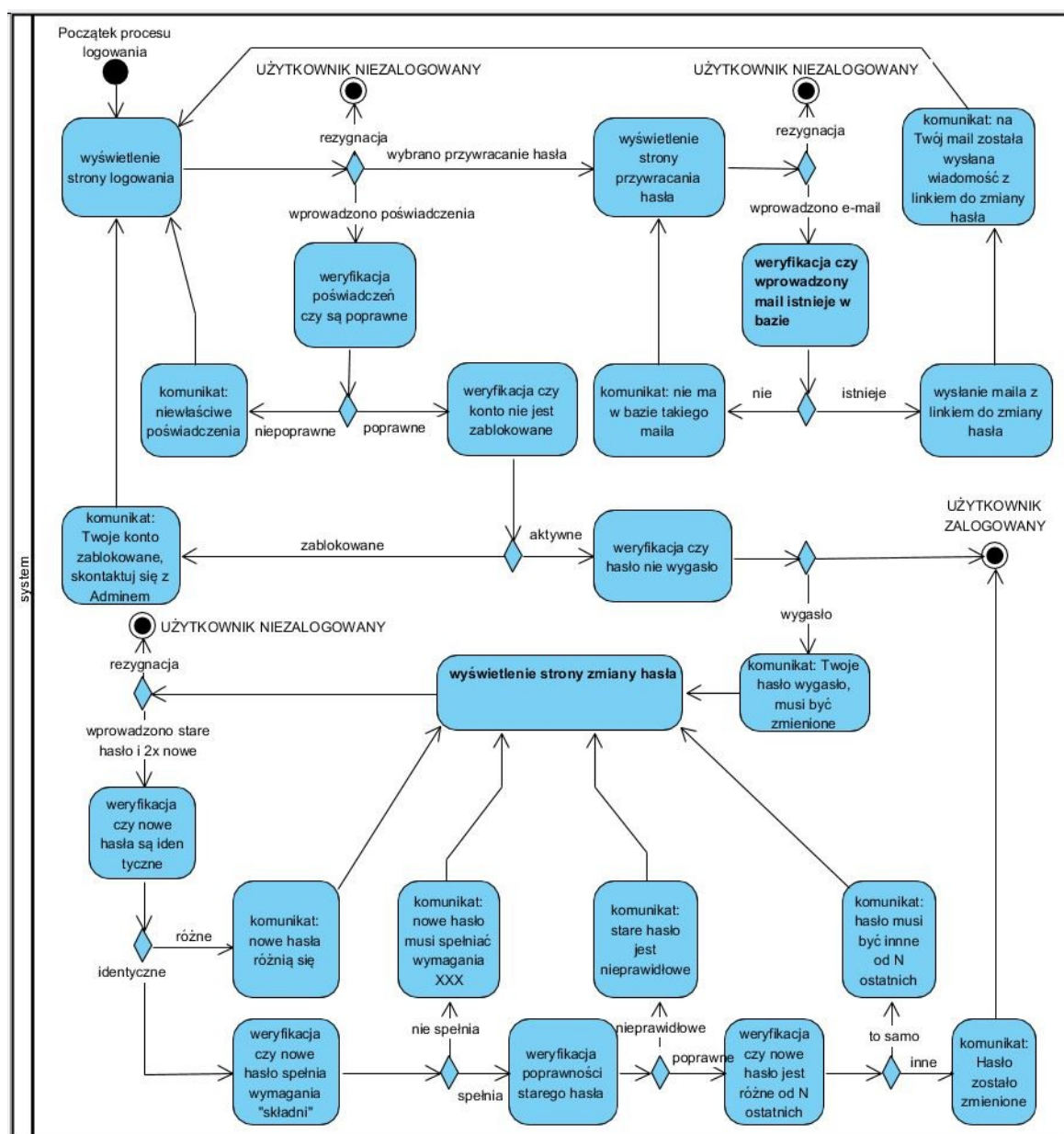


Rysunek 2.13. Dodawanie terminów sprawdzianów

Projekt Systemu

2.5. Diagramy aktywności

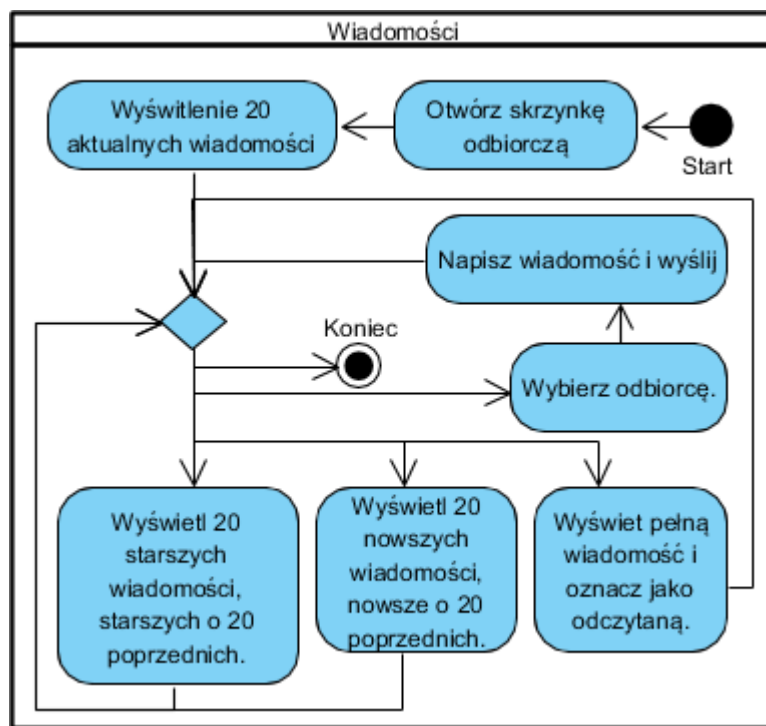
Żeby dostać się do swojego konta, potrzebny jest system, który zweryfikuje wchodzące dane. Jeśli są one prawidłowe, pozwala użytkownikowi korzystać z aplikacji. Implementowana funkcjonalność jest wprowadzona w celu zapewnienia bezpieczeństwa danych. Szczegółowy przebieg procesu logowania został przedstawiony na rysunku 2.14.



Rysunek 2.14. Przebieg procesu logowania

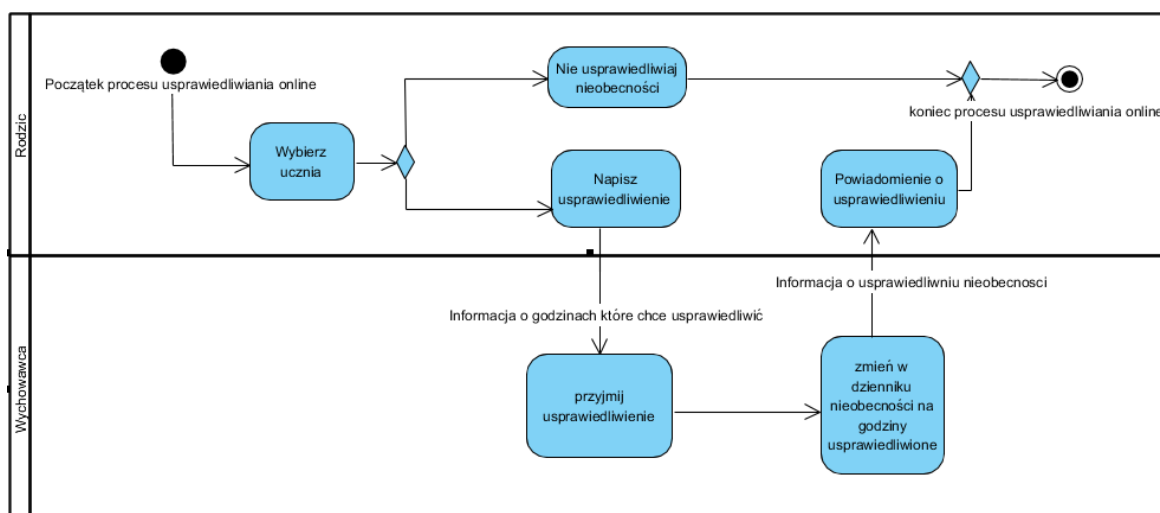
Projekt Systemu

Diagram przedstawiający pisanie oraz odbieranie wiadomości przedstawiono na rysunku 2.15.



Rysunek 2.15. Pisanie wiadomości

Często jest tak, że ucznia nie ma na zajęciach z nie wiadomo jakich przyczyn. W przypadku braku obecności opiekun może go usprawiedliwić. Diagram przedstawiający usprawiedliwienia nieobecności ucznia ukazuje rysunek 2.16.

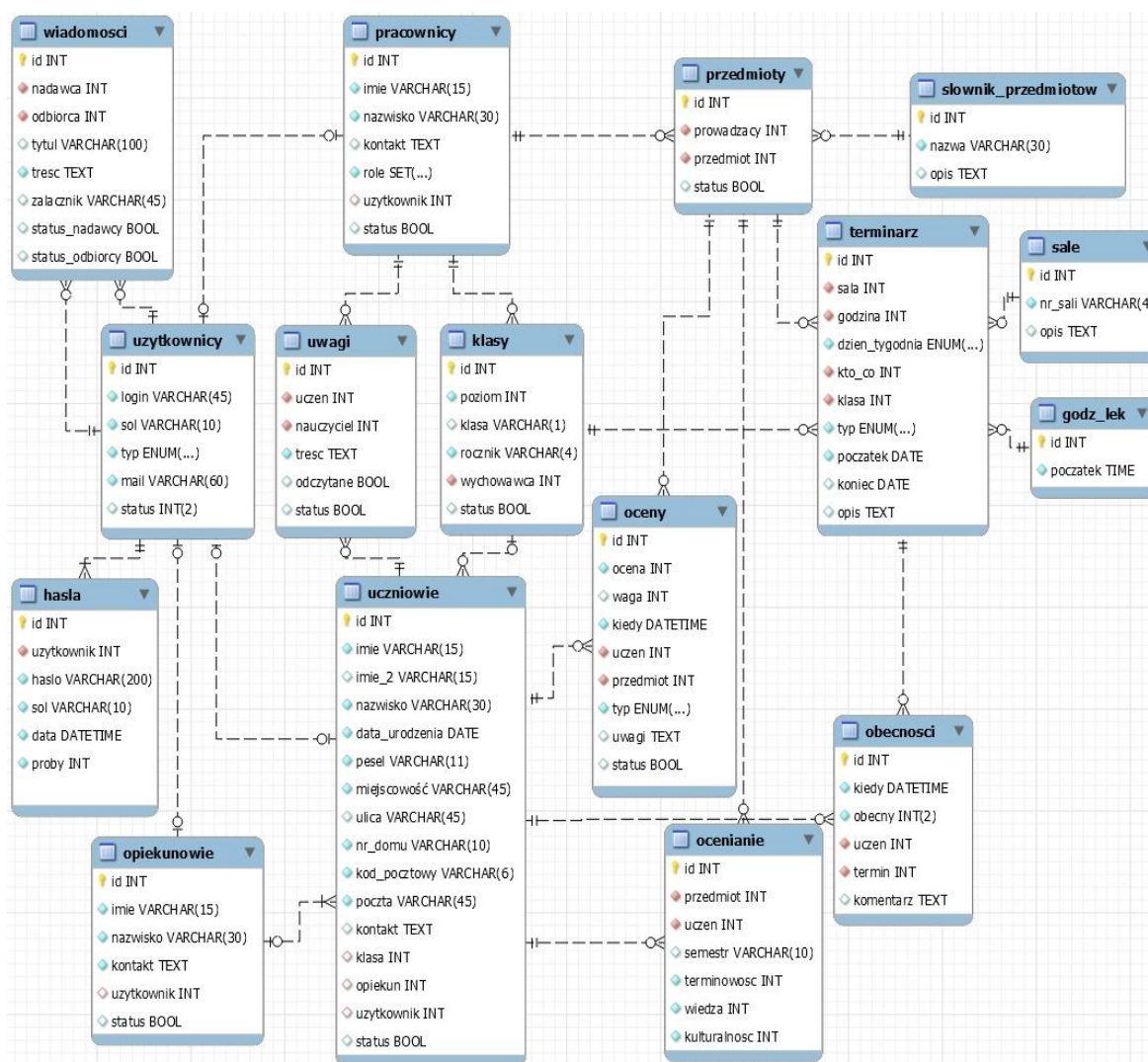


Rysunek 2.16. Usprawiedliwianie online

Projekt Systemu

2.6. Diagram EER

W celu zapewnienia możliwości przechowywania danych wykorzystana będzie relacyjna baza danych. Diagram przedstawiający tabele, kolumny, typy zmiennych oraz relacje między nimi ukazany jest na rysunku 2.17.



Rysunek 2.17. Relacyjna baza danych

3. Implementacja systemu

W tym rozdziale przedstawiony jest opis głównych elementów systemu opracowywanego w ramach niniejszej pracy. Najważniejsze części składowe kodu możemy podzielić na: kontrolery, encje (ang. entities), usługi (ang. utils), strukturę widoków (HTML) 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 omawianym systemie 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” [LIT]. 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/uzytkownicy/{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

```
<?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

Funkcjonalności 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;

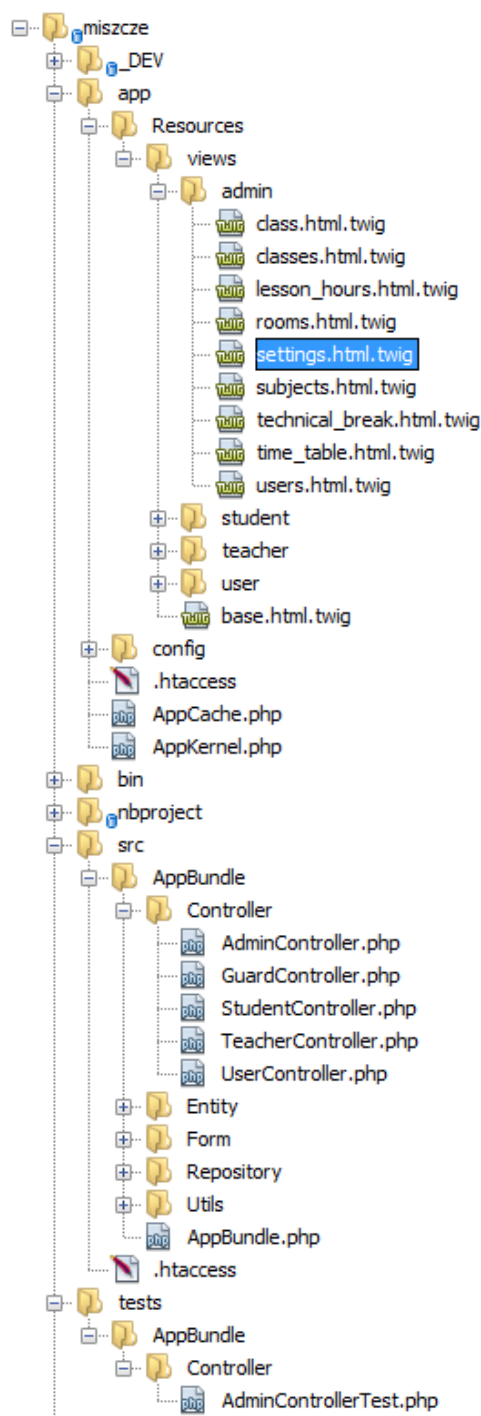
        $messages=$this->em->createQuery(
            "SELECT count(w.id) countMessages "
            "FROM AppBundle\Entity\Wiadomosci w "
            "JOIN AppBundle\Entity\Uzytkownicy u "
            "WITH w.odbiorca=u.id "
            "WHERE w.odbiorca=".$sessionUserId." AND w.odczytana=0 AND w.statusOdbiorcy=0"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

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

Implementacja systemu

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-uczeń 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].początek|date('H:i') }} -
24         {{ lessonHours[i].początek|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 funkcjonalności, w pracy przedstawiono jedynie wybrane testy prezentujące poprawne działanie kluczowych funkcjonalności 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 z 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 funkcjonalnością podstrony terminarza jest dodawanie terminów lekcji do planu zajęć. Funkcjonalność ta została przetestowana poprzez zrealizowanie scenariuszy 4.1 oraz 4.2.

Scenariusz testowy 4.1:

- a) dodawanie nowego terminu,
- b) zalogowany użytkownik posiadający rolę administratora, otworzony formularz dodawania terminarza (rys. 4.1),

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) dodany nowy termin,
- d) 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) 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) 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:

- dodawanie nowego terminu,
- użytkownik posiadający rolę administratora zalogowany, otworzony formularz dodawania terminarza (rys. 4.1),

w bazie danych istnieją: klasa, nauczyciel oraz przedmiot,

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,

- komunikat informujący użytkownika o rodzaju popełnionego błędu, brak dodania nowego terminu, formularz dodawania nowego terminu wypełniony dotychczasowymi danymi umożliwiającymi poprawienie błędnych wpisów,

- 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,

- brak dodania terminu, wyświetlenie komunikatu błędu poniżej wypełnionego formularza dodawania nowego terminu (rys. 4.4),

- 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 31.10.2018

Koniec 01.10.2018

Opis

Wyślij

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:

- logowanie do aplikacji,
- użytkownik posiadający uprawnienia administratora znający swój login i hasło, otworzony formularz logowania,
- zalogowanie do aplikacji,
- użytkownik wprowadza poprawny login oraz hasło (rys. 4.5),
- użytkownik zalogowany, wyświetlony komunikat o poprawnym zalogowaniu (rys. 4.6),
- test zakończony pomyślnie.

The screenshot shows the login interface of the 'Dziennik elektroniczny' application. At the top, there is a dark blue header with a home icon on the left and a right arrow icon on the right. Below the header is a large blue area containing a white login form. The form has the title 'Logowanie' in bold black text. It contains two input fields: 'Login' with the text 'admin' and 'Hasło' with five dots. Below these fields is a blue button labeled 'Wyślij'. At the bottom of the blue area, 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 home icon, a grid of navigation buttons (Użytkownicy, Klasy, Klasa, Przedmioty, Godziny lekcyjne, Sale, Plan zajęć, Ustawienia, Sprawdzanie obecności, Wstawienie oceny, Spóźnienia, Plan zajęć, Uwagi i pochwały, Obecności, Oceny), and a user status 'Zalogowany jako: admin' with an envelope icon and a notification count '(1)' with a right arrow icon. Below the header is a large blue area with a white main content box. Inside this box, the title 'Strona główna' is displayed in bold black text, followed by a light blue box containing the text 'Zalogowano.'. At the bottom of the blue area, there is a dark blue footer with the text 'Dziennik elektroniczny'.

Rysunek 4.6. Logowanie zakończone sukcesem

Testy aplikacji

Scenariusz testowy 4.4:

- a) logowanie do aplikacji,
- b) użytkownik nie zna loginu lub hasła lub jednocześnie loginu i hasła,
- c) komunika informujący użytkownika o wprowadzeniu niepoprawnych danych autoryzacyjnych,
- d) użytkownik wprowadza niepoprawne dane autoryzacyjne (rys. 4.5),
- e) logowanie wstrzymane, wyświetlenie komunikatu poniżej wypełnionego formularza logowania (rys. 4.7),
- f) test zakończony pomyślnie.

The screenshot shows a web application interface with a dark blue header bar containing 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 a light gray border. Inside the white box, the title 'Logowanie' is displayed in a large, bold, black font. Below the title are two input fields: the first is labeled 'Login' and contains the text 'admin'; the second is labeled 'Hasło' and contains six dots. To the left of the 'Hasło' field is a blue button with the text 'Wyślij' in white. Below these fields is a pink rectangular box containing the text 'Błędny login lub hasło.' in a dark red font. At the bottom of the blue area is a dark blue horizontal bar with the text 'Dziennik elektroniczny' in white.

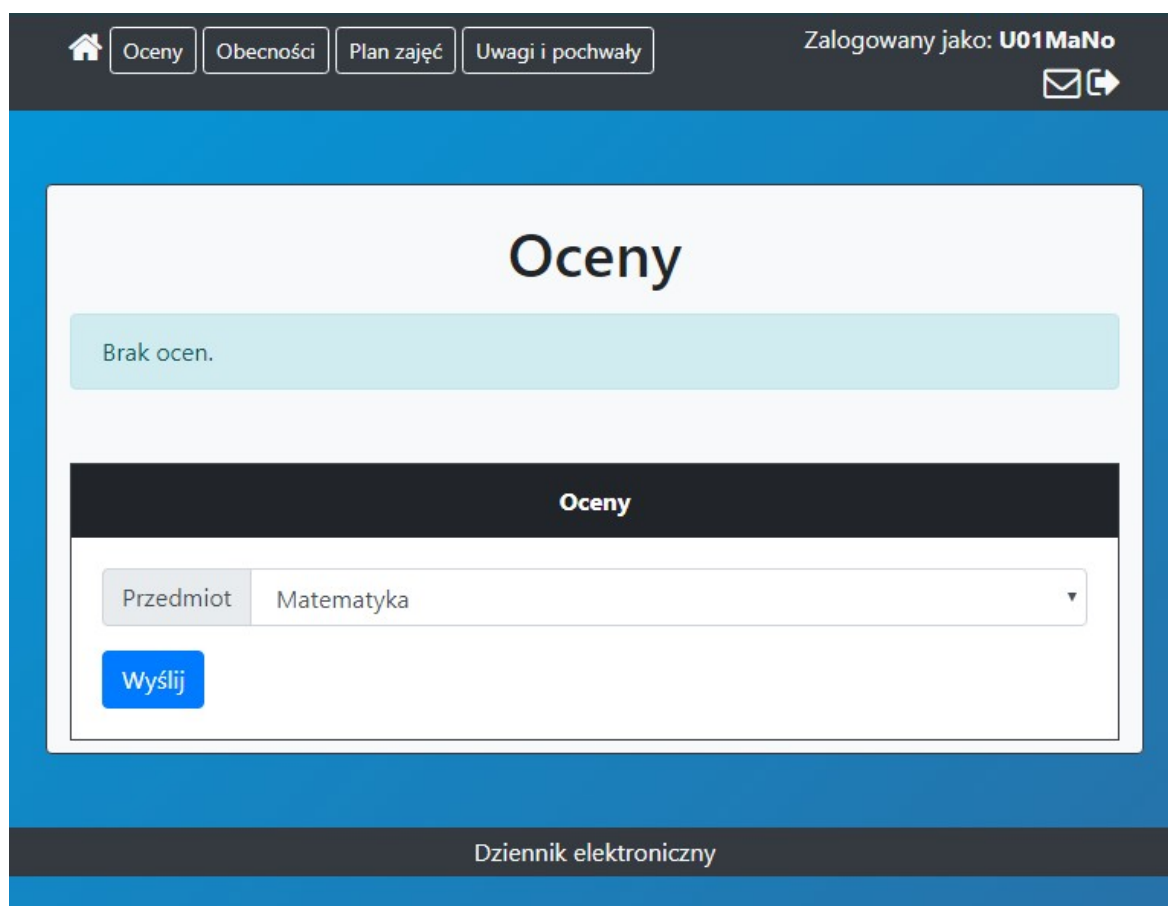
Rysunek 4.7. Logowanie zakończone niepowodzeniem

Scenariusz testowy 4.5:

- a) logowanie do aplikacji,

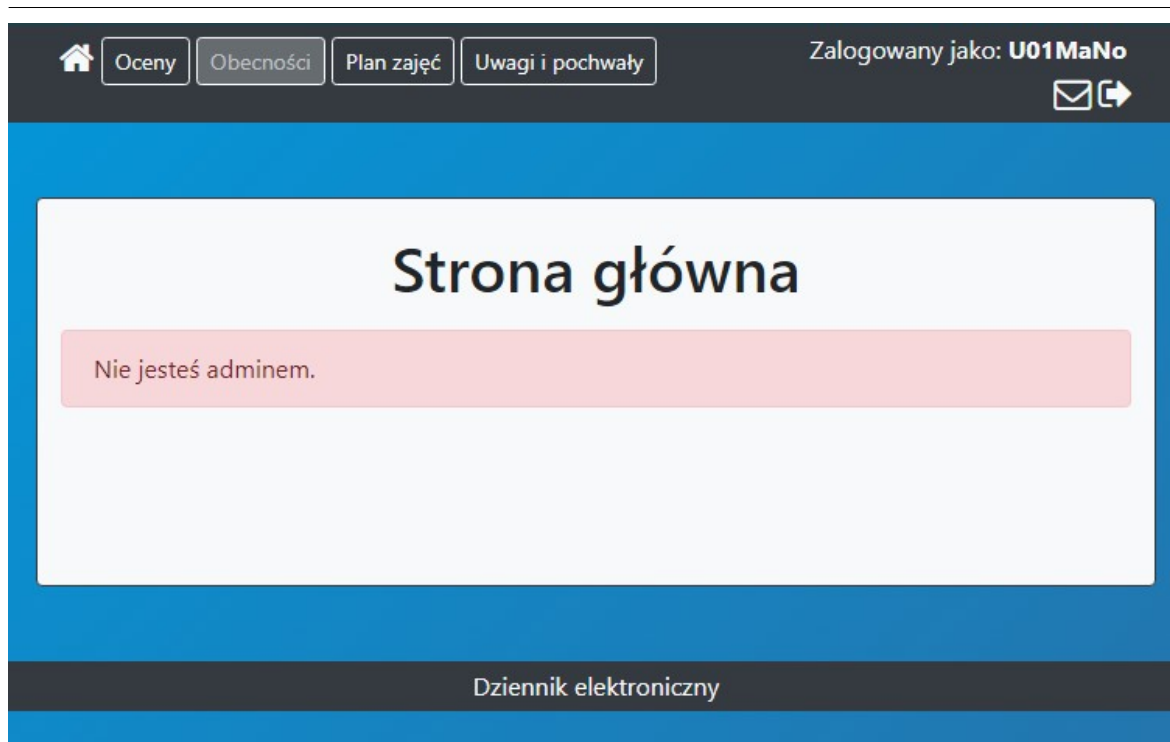
Testy aplikacji

- b) użytkownik zalogowany (rys. 4.8), użytkownik posiada uprawnienia ucznia, użytkownik nie posiada uprawnień administratora, użytkownik zna adres panelu administratora,
- c) niedopuszczenie użytkownika do panelu administratora, komunikat informujący użytkownika o braku uprawnień,
- d) użytkownik w pasku adresu przeglądarki wprowadza adres panelu administratora: <http://localhost:8000/admin/uzytkownicy#>,
- e) użytkownik nie zostaje przekierowany do panelu administratora, wyświetla się komunikat o braku odpowiednich uprawnień (rys. 4.9),
- f) test zakończony pomyślnie.



Rysunek 4.8. Zalogowany użytkownik o uprawnieniach ucznia

Testy aplikacji



Rysunek 4.9. Zabezpieczenie funkcjonalności 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 testującą kluczowe metody danej klasy. Przykładowy kod klasy testów przedstawiono na listingu 4.1.

Testy aplikacji

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();

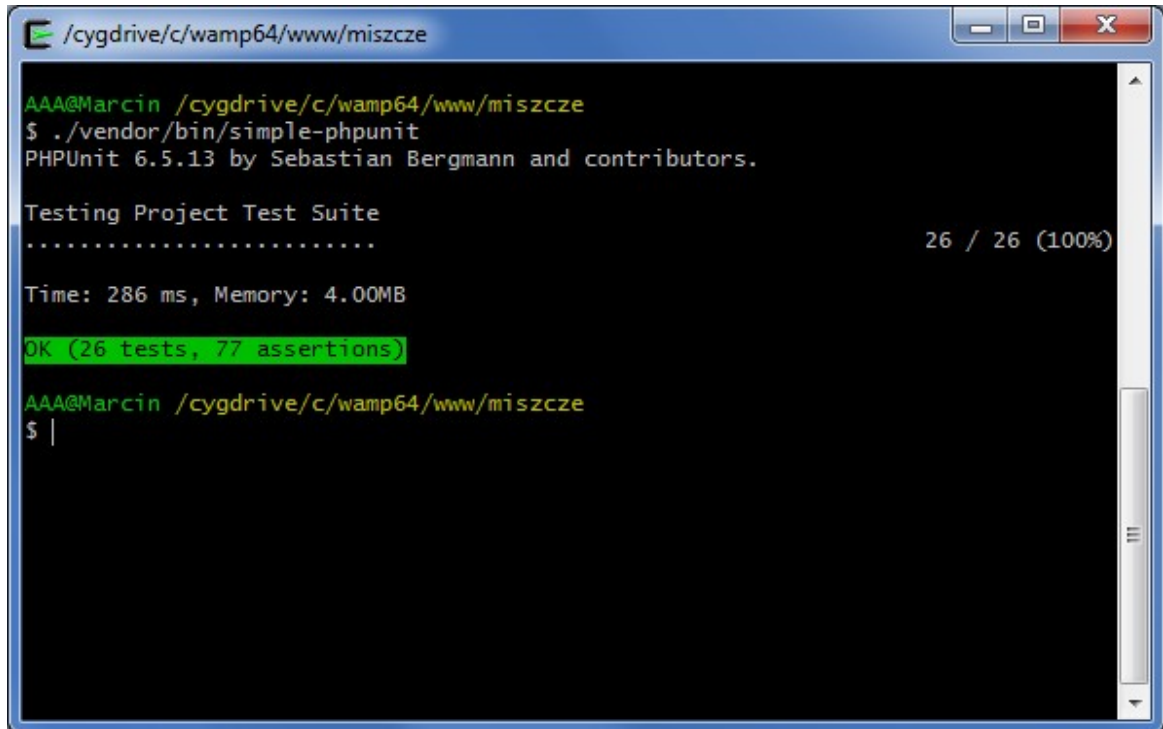
    $client->request('GET','/admin/terminarz');

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

Opracowane testy były uruchamiane cyklicznie po wprowadzeniu kolejnych zmian do kodu aplikacji. W przypadku wystąpienia błędów były one usuwane tak, by testy kończyły się wynikiem pozytywnym. Po opracowaniu całości aplikacji uruchomiono testy ponownie

Testy aplikacji

w celu sprawdzenia czy wszystkie kluczowe metody działają poprawnie. Zrzut ekranu prezentujący wynik wykonania testów został zaprezentowany na rysunku 4.10.

A screenshot of a terminal window with a blue title bar. The title bar text is "/cygdrive/c/wamp64/www/miszcze". The terminal content shows a command prompt session. The user runs the command to execute PHPUnit tests. The output indicates that all 26 tests passed with 77 assertions, and the execution time was 286 ms using 4.00MB of memory. The final result "OK (26 tests, 77 assertions)" is highlighted in green. The prompt returns to the user's shell.

```
/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 opracowano kompletny system dziennika elektronicznego. System ten powstał na podstawie wiedzy uzyskanej od specjalistów z dziedziny edukacji i przeznaczony jest do wspomagania pracy placówek edukacyjnych. Stworzono interfejsy przeznaczone dla wszystkich grup użytkowników. Powszechny i łatwy dostęp do systemu zapewniono przez udostępnienie interfejsu użytkownika poprzez przeglądarkę w postaci dynamicznej strony WWW. Opracowana aplikacja została przetestowana. W trakcie tworzenia aplikacji systematycznie tworzono 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 funkcjonalności.

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, do wspomagania 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. <https://www.librus.pl> [dostęp elektroniczny dnia 29.05.2018r]
2. <https://www.vulcan.edu.pl/> [dostęp elektroniczny dnia 29.05.2018r]
3. <https://helion.pl/ksiazki/symfony-2-od-podstaw-wlodzimierz-gajda,symfo2.htm#format/e>
4. <https://symfony.com/>
5. <https://stackoverflow.com/>

Spis rysunków

Rysunek 2.1. Diagram przypadków użycia.....	11
Rysunek 2.2. Sprawdzanie ocen.....	12
Rysunek 2.3. Wstawianie ocen.....	12
Rysunek 2.4. Korekta ocen.....	12
Rysunek 2.5. Ocenianie nauczycieli.....	13
Rysunek 2.6. Sprawdzanie obecności.....	13
Rysunek 2.7. Kontrola nieobecności.....	14
Rysunek 2.8. Oceny semestralne.....	15
Rysunek 2.9. Tworzenie planu zajęć.....	15
Rysunek 2.10. Tworzenie klas.....	16
Rysunek 2.11. Tworzenie kont.....	17
Rysunek 2.12. Wstawianie uwag.....	18
Rysunek 2.13. Dodawanie terminów sprawdzianów.....	19
Rysunek 2.14. Przebieg procesu logowania.....	20
Rysunek 2.15. Pisanie wiadomości.....	21
Rysunek 2.16. Usprawiedliwianie online.....	21
Rysunek 2.17. Relacyjna baza danych.....	22
Rysunek 3.1. Struktura projektu aplikacji.....	28
Rysunek 3.2. Widok planu zajęć.....	30
Rysunek 4.1. Formularz przed dodaniem terminu.....	33
Rysunek 4.2. Dodawanie terminu.....	33
Rysunek 4.3. Nowy przedmiot na planie lekcji.....	34
Rysunek 4.4. Formularz po wysłaniu błędnych danych.....	35
Rysunek 4.5. Formularz logowania.....	36
Rysunek 4.6. Logowanie zakończone sukcesem.....	36
Rysunek 4.7. Logowanie zakończone niepowodzeniem.....	37
Rysunek 4.8. Zalogowany użytkownik o uprawnieniach ucznia.....	38
Rysunek 4.9. Zabezpieczenie funkcjonalności administratora.....	39
Rysunek 4.10. Wyniki wykonania testów jednostkowych.....	41

Spis listingów

Listing 3.1. Uchwył strony umożliwiający operacje na godzinach lekcyjnych.....	23
Listing 3.2. Formularz dodawania godziny lekcyjnej.....	24
Listing 3.3. Fragment kodu realizujący dodawanie nowej godziny lekcyjnej.....	24
Listing 3.4. Kod umożliwiający usuwanie godziny lekcyjnej.....	24
Listing 3.5. Kod zwracający widok, który prezentuje listę godzin lekcyjnych.....	24
Listing 3.6. Usuwanie sesji użytkownika.....	25
Listing 3.7. Encja do tabeli GodzLek.....	26
Listing 3.8. Usługa message.....	27
Listing 3.9. Widok time_table.....	29
Listing 4.1. Klasa testów "Test Terminarza".....	40