

*Akademia Górnictwo-Hutnicza im. Stanisława Staszica w Krakowie  
Wydział Inżynierii Metali i Informatyki Przemysłowej*

# **Dokumentacja Techniczna Aplikacja Gabinetu Dentystycznego**

**Wykonali:**

**Mateusz Kuzera**

**Wiktor Kowalski**

**Mateusz Ługowski**

Kierunek Informatyka Techniczna  
Studia I stopnia stacjonarne



Semestr III, rok akademicki 2022/2023  
Grupa laboratoryjna 3

**Kraków 2022/2023**

## **Spis treści:**

- 1. Słownik pojęć i terminów**
- 2. Specyfikacja projektu**
  - a. Koncepcja systemu**
  - b. Opis systemu**
  - c. Podział systemu**
  - d. Przepływ danych w systemie**
- 3. Metodyka wytwarzania**
- 4. Wymagania funkcjonalne i niefunkcjonalne**
- 5. Diagramy UML**
  - a. Diagram przypadków użycia**
  - b. Diagramy wymagań**
  - c. Diagramy aktywności**
  - d. Diagram klas**
  - e. Diagramy sekwencji**
  - f. Diagram komponentów**
  - g. Diagram wdrożenia**
- 6. Wykaz zastosowanych framework-ów, bibliotek, środowiska, technologii oraz pozostałych rozwiązań informatycznych**
- 7. Przykłady współpracy zespołu (github, trello)**
- 8. Testy jednostkowe – 2 przykłady**
- 9. Przykład refaktoryzacji kodu – 3 przykłady**
- 10. Wzorzec projektowy**
- 11. Instrukcja użytkowania programu**
- 12. Podsumowanie**

## 1. Słownik najważniejszych pojęć i terminów

**menu** - obiekt klasy Okno służący jako podstawę do całego programu

**logging\_menu** - obiekt klasy Logging, służy do wysyłania emaili

**mailbox** - obiekt klasy Mail, służy do wysyłania maili

**window** - obiekt klasy RenderWindow(SFML), służy jako okno które się wyświetla

**ev** - obiekt klasy Event(SFML), służy do "wyłapywania" eventów (np. wcisnięcie przycisku na klawiaturze)

**video\_mode** - obiekt klasy VideoMode(SFML), służy do ustawienia parametrów okna, np. rozdzielczości.

**MousePosWindow** - obiekt klasy sf::Vector2i(SFML), jest to wektor (int,int) określający położenie myszki na oknie

**font1** - obiekt klasy sf::Font(SFML), służy jako nośnik czcionki używanej w programie

**text1, text2, text3, text4, text5** - obiekty klasy Textline, służące do określania i wyświetlania napisów tekstów

**b1, b2, b3, b4, b5, b6, b7, b8, b9** - obiekty klasy Button służące do określania i wyświetlania przycisków tekstowych

**background\_photo, logo** - obiekty klasy sf::Texture(SFML) służącej jako nośnik zdjęcia.

**icon** - obiekt klasy sf::Image(SFML) służącej jako nośnik ikony.

**sprite, sprite2** - obiekt klasy sf::Sprite(SFML), służącej do wyświetlania na oknie.

**input\_bar1, input\_bar2, input\_bar3, input\_bar4, input\_bar5, input\_bar6, input\_bar7** - obiekty klasy Input\_bar, służącej jako wejście tekstu

**site** - obiekt klasy sites, służącej do wskazywania na aktualną stronę wyświetlana

**assistant** - obiekt klasy Assistant

**logged\_user** - obiekt klasy Pracownik

**adm** - obiekt klasy Admin

**dentist** - obiekt klasy Dentist

**pacient\_list** - obiekt klasy Pacjenci, służy do kompozycji wektora pacjentów

**worker\_list** - obiekt klasy Pracownicy, służy do kompozycji wektora Pracowników

**kalendarz** - obiekt klasy Kalendarz

**finanse** - obiekt klasy Finanse

**mag** - obiekt klasy Magazyn

**kartoteka2** - obiekt klasy Kartoteka

## 2. Specyfikacja projektu

### Koncepcja systemu

Koncepcja systemu zakłada utworzenie wielofunkcyjnej aplikacji, mającej na celu usprawnienie działania gabinetu dentystycznego. System znaczco poprawi organizację zadań pracowników w gabinecie dentystycznym, jak również spowoduje oszczędności w czasie, co przełoży się na większą efektywność pracy.

### Opis systemu

System opiera się na wielu klasach, odpowiadających za poszczególne elementy w aplikacji. Aplikacja zakłada możliwość tworzenia konta przez użytkownika - pracownicy gabinetu dentystycznego. Użytkownik dostaje możliwość zmiany maila i hasła. Zalogowany pracownik posiada możliwość przeglądania kalendarza, dodania wizyty do kalendarza i modyfikacji wizyt pacjentów. System zakłada modyfikację wizyt pacjentów jako modyfikację bazy danych z informacjami o pacjentach i wizytach. Dodana jest również możliwość wysyłania maili z informacją o nadchodzącej wizycie dzień przed jej terminem.

### Podział systemu

System składa się z części dostosowanej dla użytkownika - interfejs graficzny i części niewidocznej, odpowiadającej za działanie systemu. Aplikacja podzielona jest na klasy, działające kompatybilnie ze sobą. Istnieją kasy odpowiedzialne za wyświetlanie się menu użytkownika, są to klasy które głównie korzystają z bibliotek SFML. Pozostałe klasy w aplikacji gabinetu dentystycznego tworzą wszystkie funkcje takie jak możliwość logowania i zakładania konta, kalendarz, dodawanie i edycja wizyt. Aplikacja posiada również oddzielny moduł umożliwiający wysyłanie maili.

## Przepływ danych w systemie

System zakłada możliwość wprowadzania danych i ich magazynowania w bazie danych. Dane wprowadzane są przez użytkownika (pracownik gabinetu dentystycznego), przez klawiaturę. System zakłada możliwość logowania. W tym przypadku wprowadzone dane są weryfikowane z danymi zapisanymi w bazie danych i zwracany jest komunikat o poprawnym, bądź niepoprawnym logowaniu. Również podczas wysyłania maili dane zawierające informacje o nadchodzącej wizycie zostają wysłane przez pocztę elektroniczną do pacjenta. Do poprawnego wysłania maila należy odczytać odpowiedni adres email pacjenta, znajdujący się w bazie danych, jak również informacje o zbliżającej się wizycie.

## 3. Metodyka wytwarzania

### Opis modelu

Do realizacji projektu aplikacji gabinetu dentystycznego użyliśmy **modelu kaskadowego**. Polega on na wykonywaniu podstawowych czynności jako odrębnych faz projektowych, kolejno po sobie. W naszym projekcie po określeniu głównych celów projektowych przeszliśmy do pierwszego etapu, którym było planowanie systemu, w tym jego specyfikacji oraz analiza systemu i wymagań. Następnie zaprojektowaliśmy poszczególne struktury w naszej aplikacji i przeszliśmy do implementacji kodu. Kolejnym etapem było usprawnienie działania kodu i poprawienie jego przejrzystości. Ostatecznie przeprowadziliśmy testy poszczególnych elementów. Kroki, które wykonaliśmy idealnie odwzorowują definicję modelu kaskadowego.

## Przyczyny wyboru modelu kaskadowego

Model kaskadowy zakłada wykonywanie faz następujących kolejno po sobie. Takie rozwiązanie idealnie sprawdziło się przy pracy nad projektem aplikacji. Realizując projekt, spotykaliśmy się na cotygodniowych zajęciach z inżynierii oprogramowania i poznawaliśmy kolejne etapy procesu projektowego. Następnie wykonywaliśmy kroki poznane na zajęciach, odnosząc się do projektu aplikacji gabinetu dentystycznego.

Kroki które wykonaliśmy:

1. Planowanie systemu (w tym specyfikacja wymagań).
2. Analiza systemu (w tym analiza wymagań i studium wykonalności).
3. Projekt systemu (poszczególnych struktur itp.).
4. Implementacja (wytworzenie kodu).
5. Testowanie (poszczególnych elementów systemu oraz elementów połączonych w całość).
6. Pielęgnacja powstałego systemu.

## 4. Wymagania funkcjonalne i niefunkcjonalne

Wymagania użyte podczas tworzenia projektu dzielą się na dwa podstawowe obszary. Są to wymagania funkcjonalne i niefunkcjonalne. Aplikacja ma na celu usprawnienie zarządzania gabinetem zaoszczędzając czas pracowników.

Def:

**Wymagania Funkcjonalne** – opisują zachowania opisywanej funkcji rozwiązania, które realizują wymagania biznesowe. Wymaganie funkcjonalne jest ogólnie utożsamiane z wymaganiem użytkownika, dotyczy tego co ma realizować system: jakie ma spełniać funkcje, jakich dostarczać usług, jak zachowywać się w określonych sytuacjach.

**Wymagania Niefunkcjonalne (pozafunkcjonalne)** – są dopełnieniem wymagań funkcjonalnych opisując cechy rozwiązania pod kątem jakościowym, np. wydajności czy niezawodności. Wymagania dotyczące koniecznych zasobów, ograniczeń czasowych, niezawodności, bezpieczeństwa, przenośności, współpracy z określonymi narzędziami i środowiskami, zgodności z normami i standardami, a także przepisami prawnymi, w tym dotyczącymi tajności i prywatności.

**Wymagania funkcjonalne** założone podczas realizacji projektu aplikacji gabinetu dentystycznego:

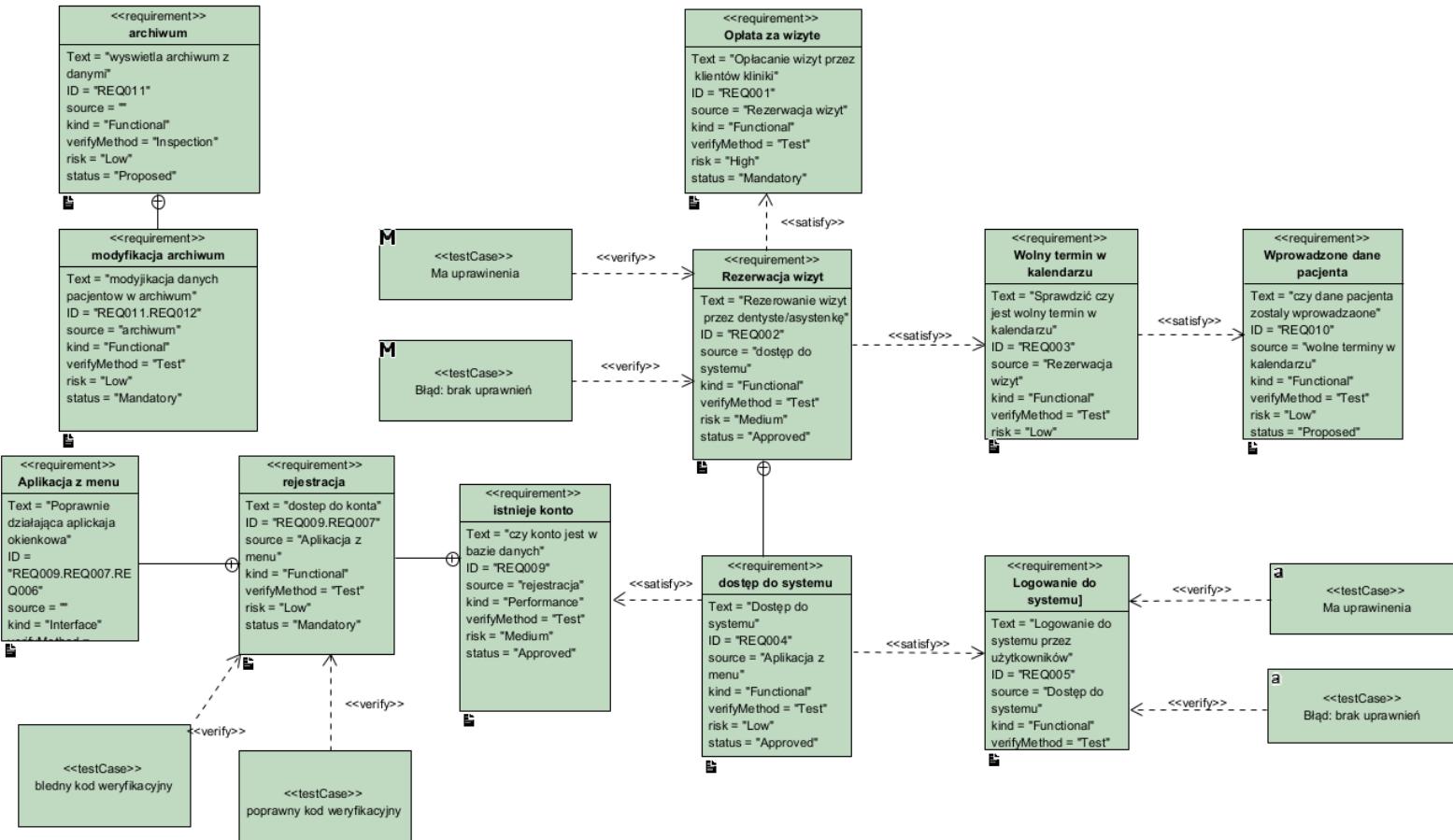
1. System powinien oferować możliwość założenia konta użytkownikowi.
2. Użytkownik powinien mieć możliwość zalogowania się do systemu przy użyciu swojej nazwy użytkownika i hasła.
3. System powinien móc zmieniać informacje na temat pracowników (login, hasło, adres email).
4. System powinien zawierać bazę danych z informacjami o pacjentach i ich wizytach.
5. Użytkownik powinien mieć możliwość modyfikacji archiwum(bazy danych).
6. Użytkownik powinien dodać nową wizytę do kalendarza.
7. System powinien wysłać wiadomość e-mail do pacjenta, z informacją o zbliżającej się wizycie.
8. Użytkownik powinien móc zarządzać magazynem tj. modyfikować ilość sprzętu w magazynie
9. Użytkownik powinien mieć dostęp do historii zabiegów
10. System powinien móc przechowywać dane użytkowników
11. System powinien generować kod weryfikacyjny do odzyskania konta

**Wymagania niefunkcjonalne** założone podczas realizacji projektu aplikacji gabinetu dentystycznego:

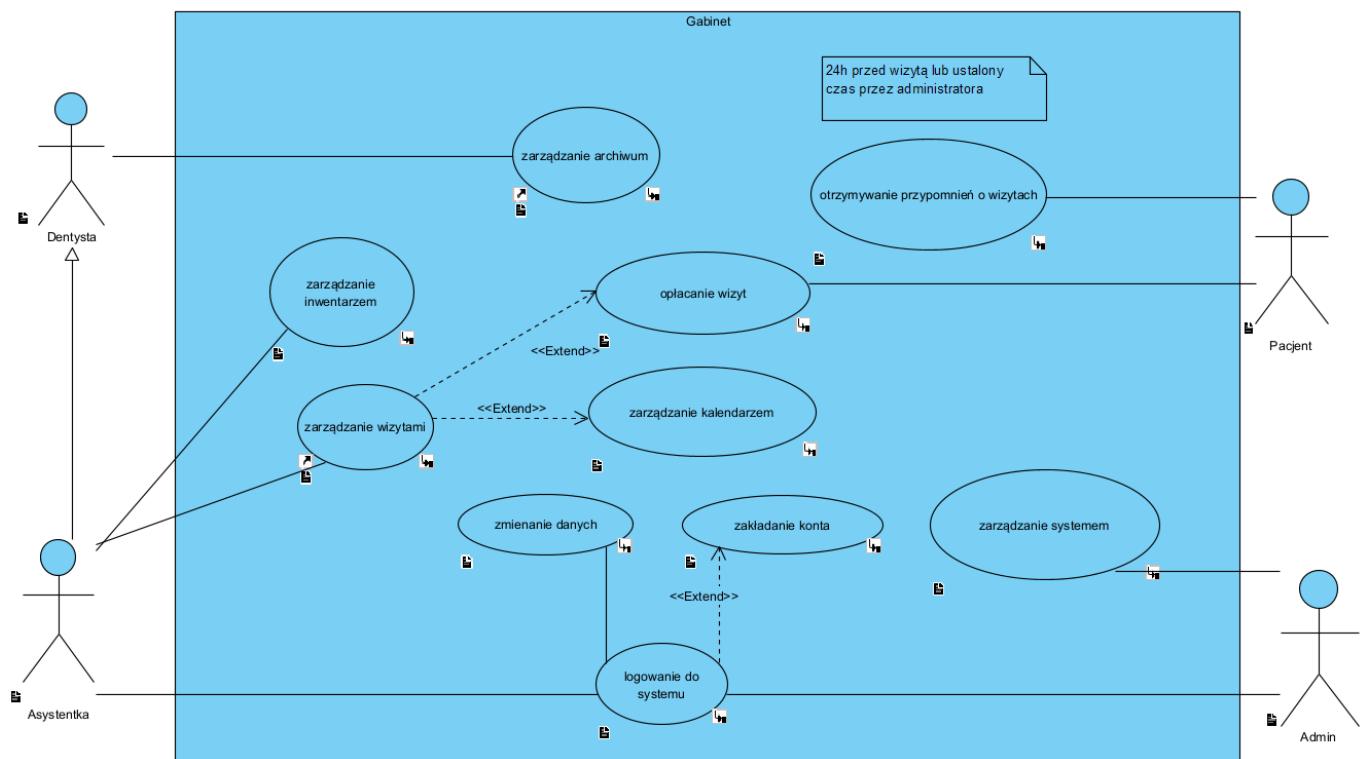
1. Prosty i przejrzysty interfejs
2. Wydajność systemu – czas reakcji na zapytania ze strony użytkowników
3. Bezpieczeństwo – zabezpieczenie danych podczas rejestracji i logowania
4. Użyteczność – łatwość korzystania z aplikacji, szybkość uczenia się obsługi aplikacji.
5. Intuicyjność interfejsu – krótki czas na naukę obsługi aplikacji

## 5. Diagramy UML

### Diagramy wymagań



## Diagram przypadków użycia



### Opis aktorów:

- **Dentysta** zajmuje się głównie zarządzaniem archiwum(tj. sprawdzaniem oraz edytowaniem historii choroby pacjenta). Dodatkowo posiada dostęp do kalendarza oraz dostęp do zarządzania wizytami. Aby Dentysta mógł z tego korzystać musi posiadać konto oraz musi być zalogowany w systemie.

### Przypadki użycia:

- |                                |                          |                       |
|--------------------------------|--------------------------|-----------------------|
| -zarządzanie archiwum          | -zarządzanie kalendarzem | -modyfikacja konta    |
| -zarządzanie inwentarzem       | -logowanie do systemu    | -zarządzanie wizytami |
| -sprawdzenie stanu finansowego |                          |                       |

- **Asystentka** zajmuje się rezerwacją wizyt, ich edycją oraz odwoływaniem. Dodatkowo ma dostęp do kalendarza który pokazuje jej rozpiszę wizyt. Jest odpowiedzialna za inwentarz (sprawdza oraz uzupełnia inwentarz poprzez dodawanie brakujących rzeczy do listy zakupów). Aby asystentka mogła z tego korzystać musi posiadać konto oraz musi być zalogowana w systemie.

Przypadki użycia:

-logowanie do systemu	-zarządzanie inwentarzem	-sprawdzenie stanu finansowego
-zarządzanie wizytami	-zarządzanie kalendarzem	-modyfikacja konta

- **Pacjent** otrzymuje powiadomienia drogą mailową o nadchodzących wizytach w gabinecie stomatologicznym. .

Przypadki użycia:

-otrzymywanie przypomnień o wizytach

- **Administrator** zajmuje się zarządzaniem systemem (tj. pełna możliwość edycji wszystkich danych w bazie) oraz zarządza pracownikami. Posiada możliwość modyfikacji konta i wizyt. Musi być zalogowany do systemu aby móc korzystać z tych możliwości systemie.

Przypadki użycia:

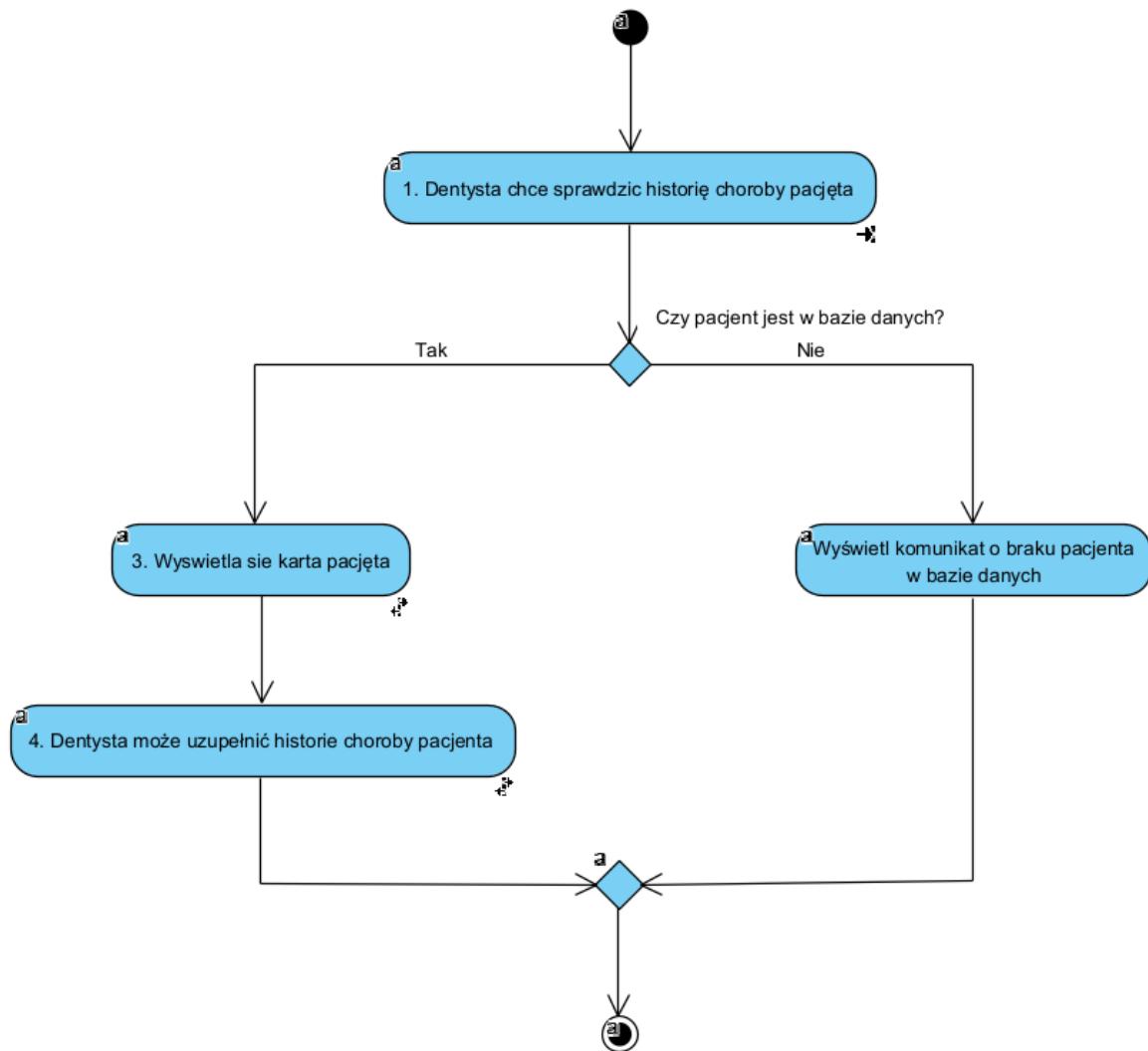
-logowanie do systemu	-zarządzanie inwentarzem	-modyfikacja konta
-zarządzanie wizytami	-zarządzanie systemem	-zarządzanie kalendarzem

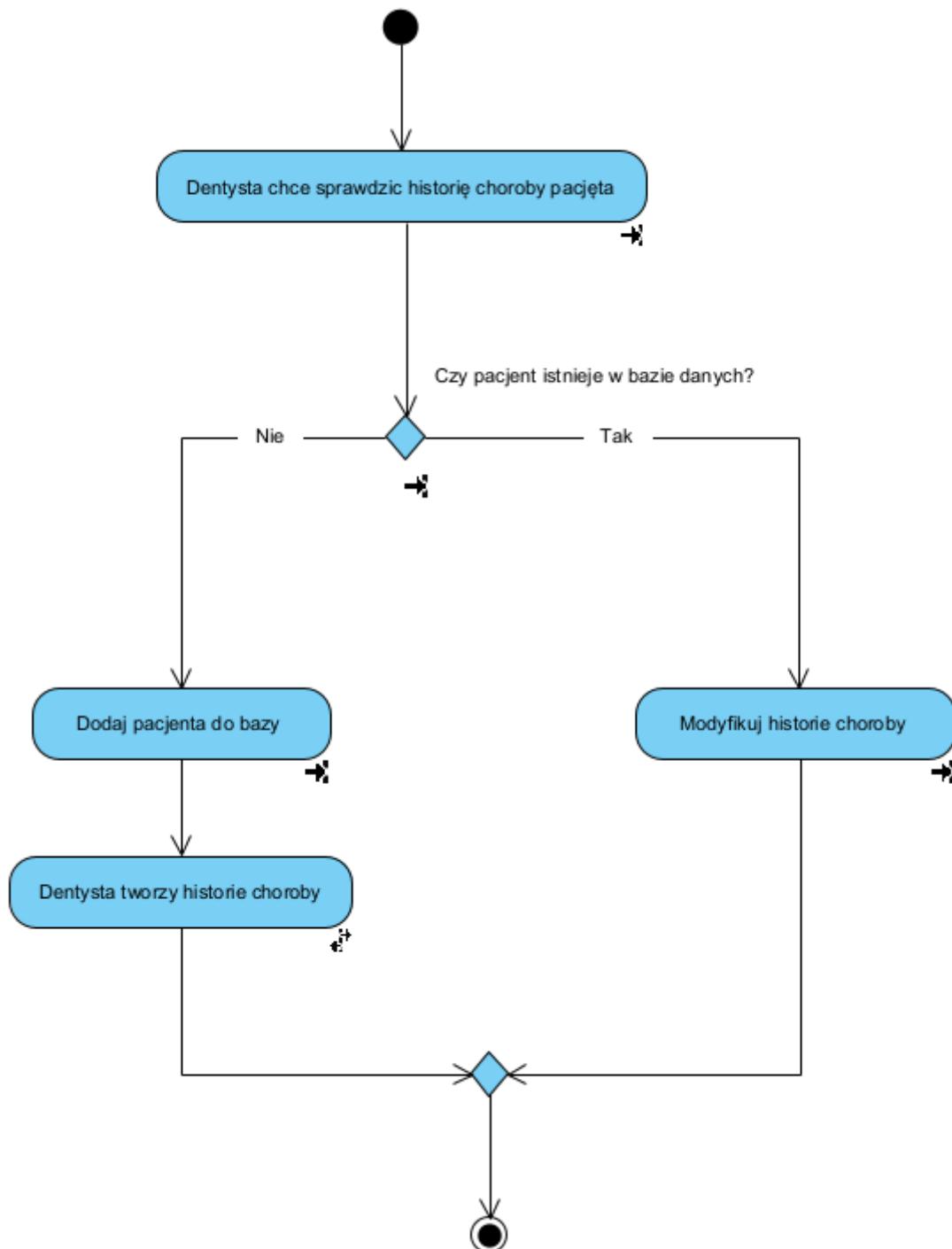
## Opis przypadków użycia:

- **Zarządzanie inwentarzem** - Uprawnieni użytkownicy, w naszym przypadku dentysta, asystentka i administrator systemu mają możliwość sprawdzenia oraz uzupełnienia inwentarza poprzez dodawanie lub edycję stanu magazynu.
- **Zarządzanie archiwum** - Polega na modyfikacji historii choroby czy zabiegów pacjenta. Uprawnia do takich działań posiada dentysta i administrator.
- **Zarządzanie wizytami** - Asystentka, dentysta, oraz administrator mają możliwość dodania wizyty, czyli wpisania w kalendarz godziny i daty wizyty wraz z danymi pacjenta. Dodatkowo istnieje możliwość edycji aktualnych wizyt.
- **Zarządzanie kalendarzem** - Uprawnieni użytkownicy mogą wyświetlić kalendarz z wizytami oraz posiadają możliwość dodania kolejnej wizyty.
- **Zakładanie konta** - Pracownik gabinetu dentystycznego może założyć konto w aplikacji. Dokonuje tego przy pomocy swojego maila i wybranego przez siebie loginu i hasła. Po otrzymaniu kodu weryfikacyjnego w mailu oraz podaniu go do aplikacji może poprawnie się zalogować.
- **Logowanie do systemu** - Użytkownik z kontem może zalogować się do systemu przy użyciu swojego loginu i hasła, podanych podczas zakładania konta.
- **Zmiana danych** - Zalogowany użytkownik posiada możliwość ustawienia swojego imienia i nazwiska oraz zmianę obecnego mailu czy hasła do konta w aplikacji. Dodatkowo w przypadku zapomnienia hasła przez użytkownika ma on możliwość odzyskania go.
- **Zarządzanie systemem** - Jest to przypadek użycia przypisany dla administratora systemu, który posiada możliwość modyfikacji wszystkich danych użytkowników, wizyt w kalendarzu oraz archiwum.
- **Otrzymywanie przypomnień o wizytach** - Pacjent otrzymuje powiadomienie o nadchodzącej dzień przed wizytą. Powiadomienie przychodzi na adres email podany przez pacjenta i zapisany w bazie danych,

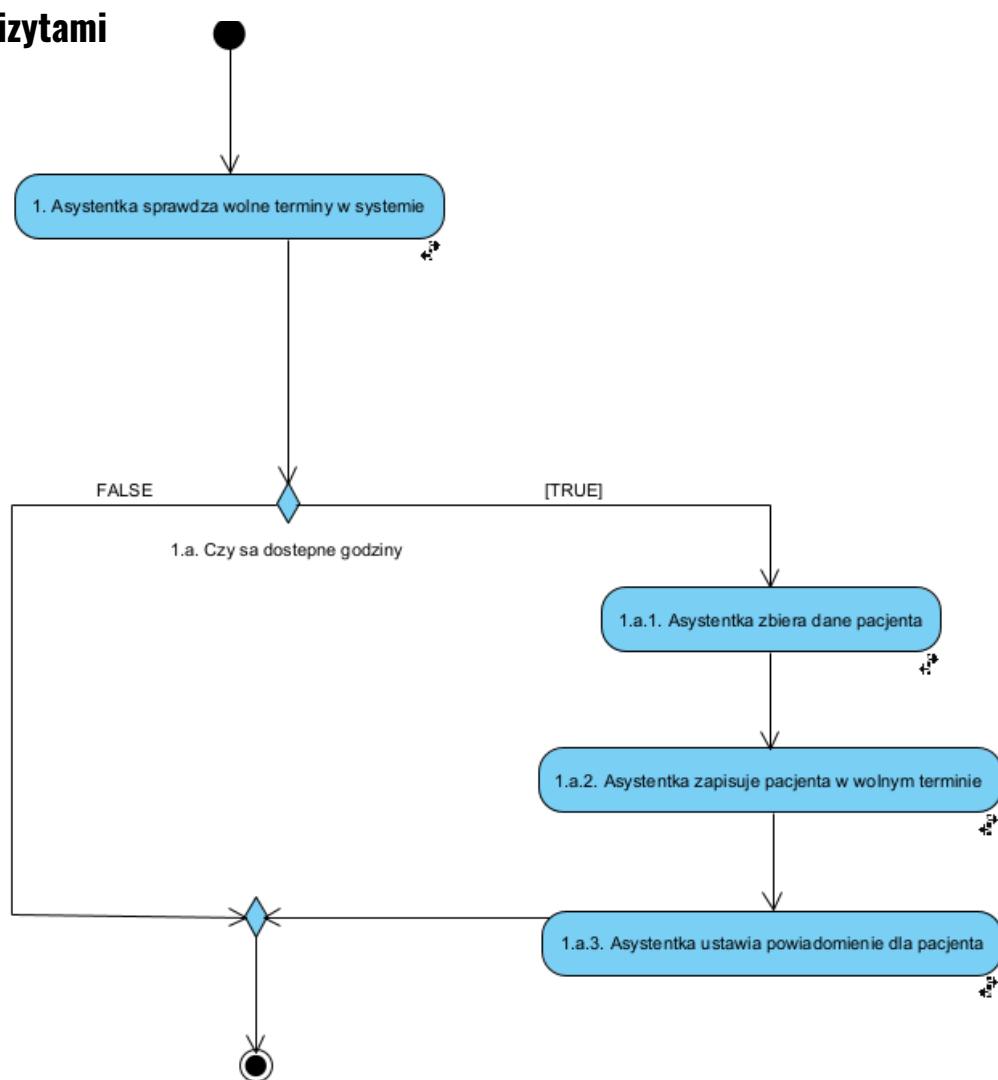
## Diagramy aktywności

### Zarządzanie archiwum

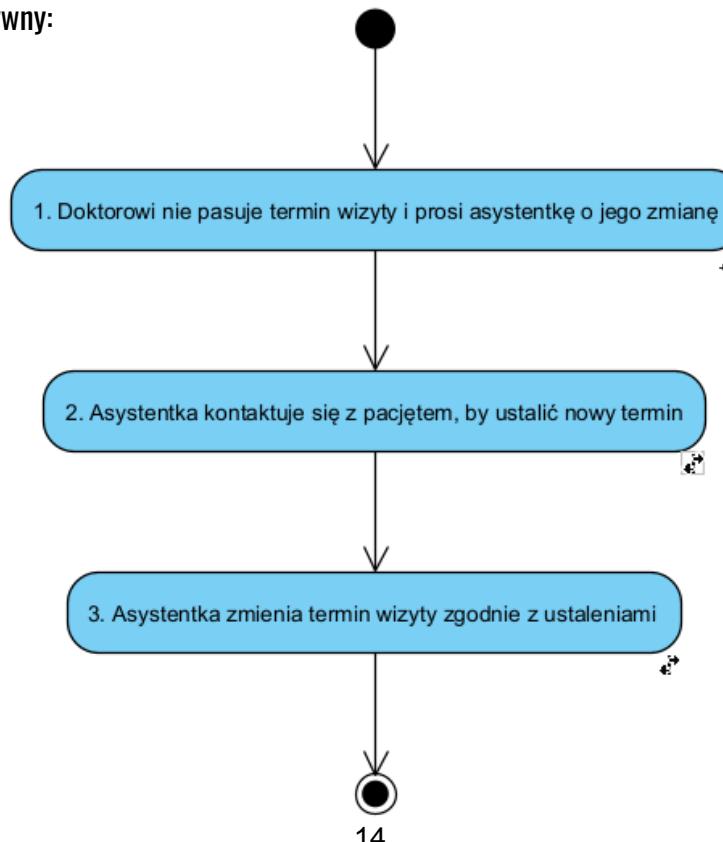


**scenariusz alternatywny:**

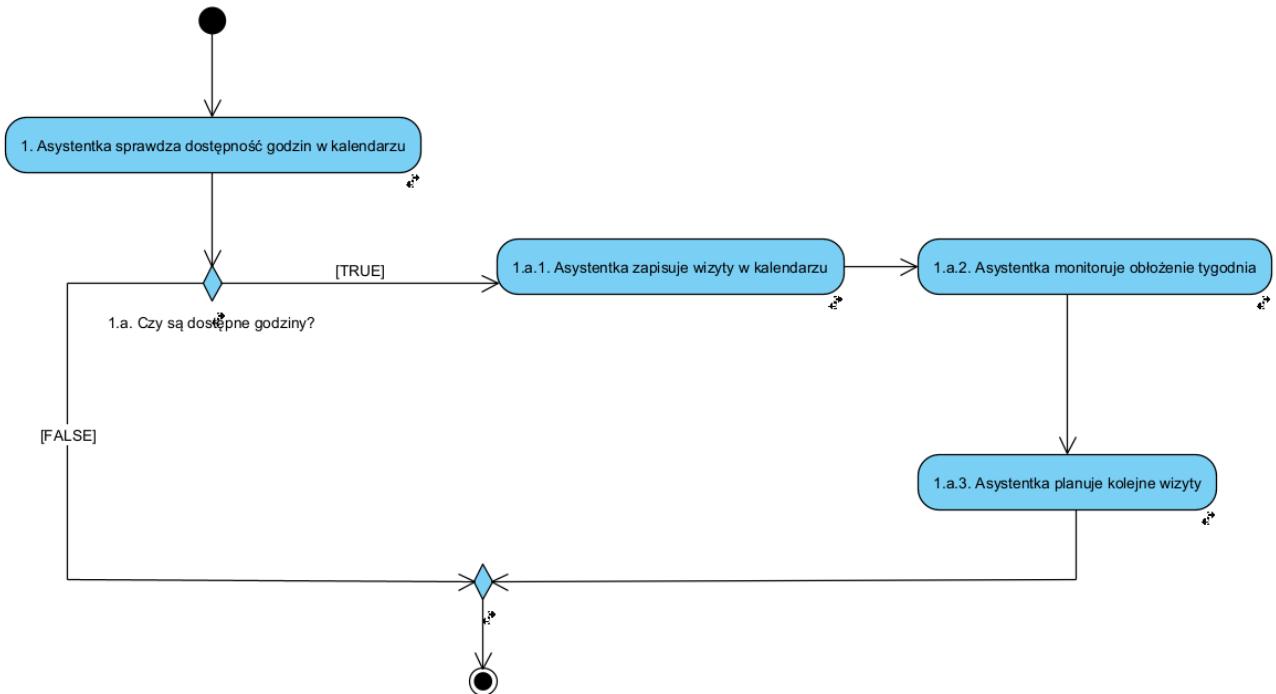
## Zarządzanie wizytami



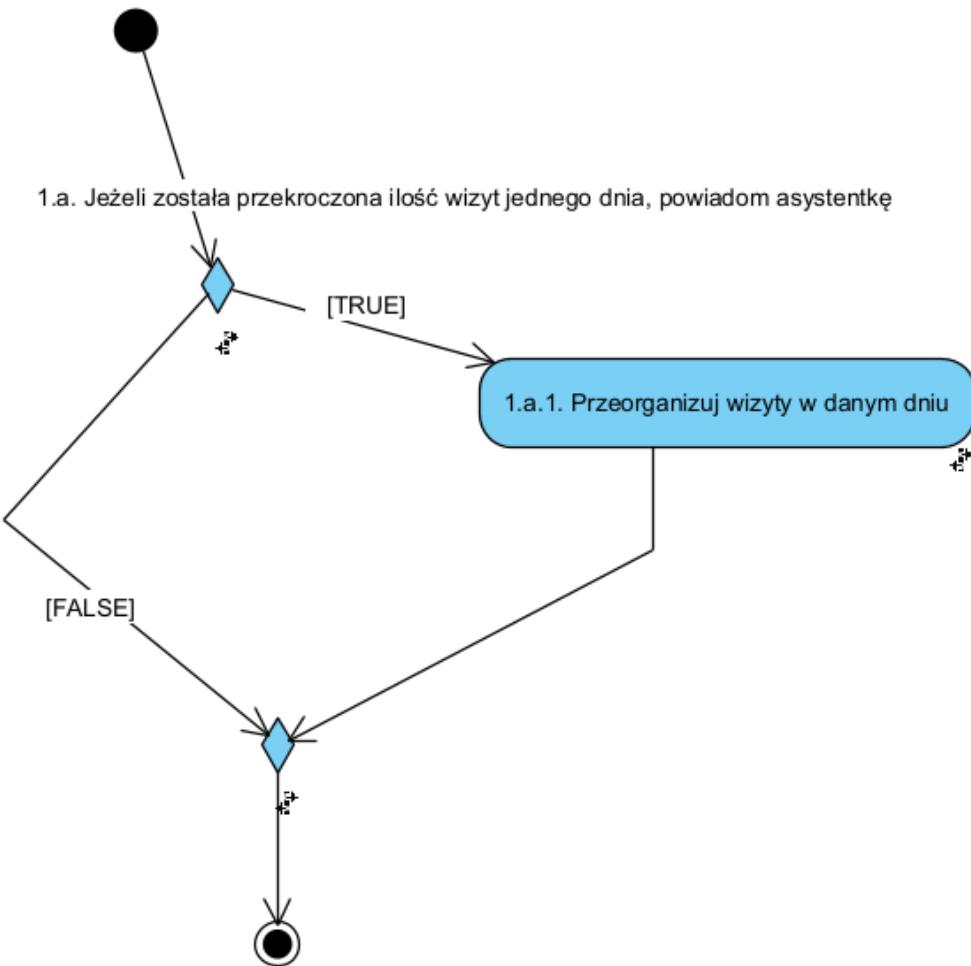
scenariusz alternatywny:



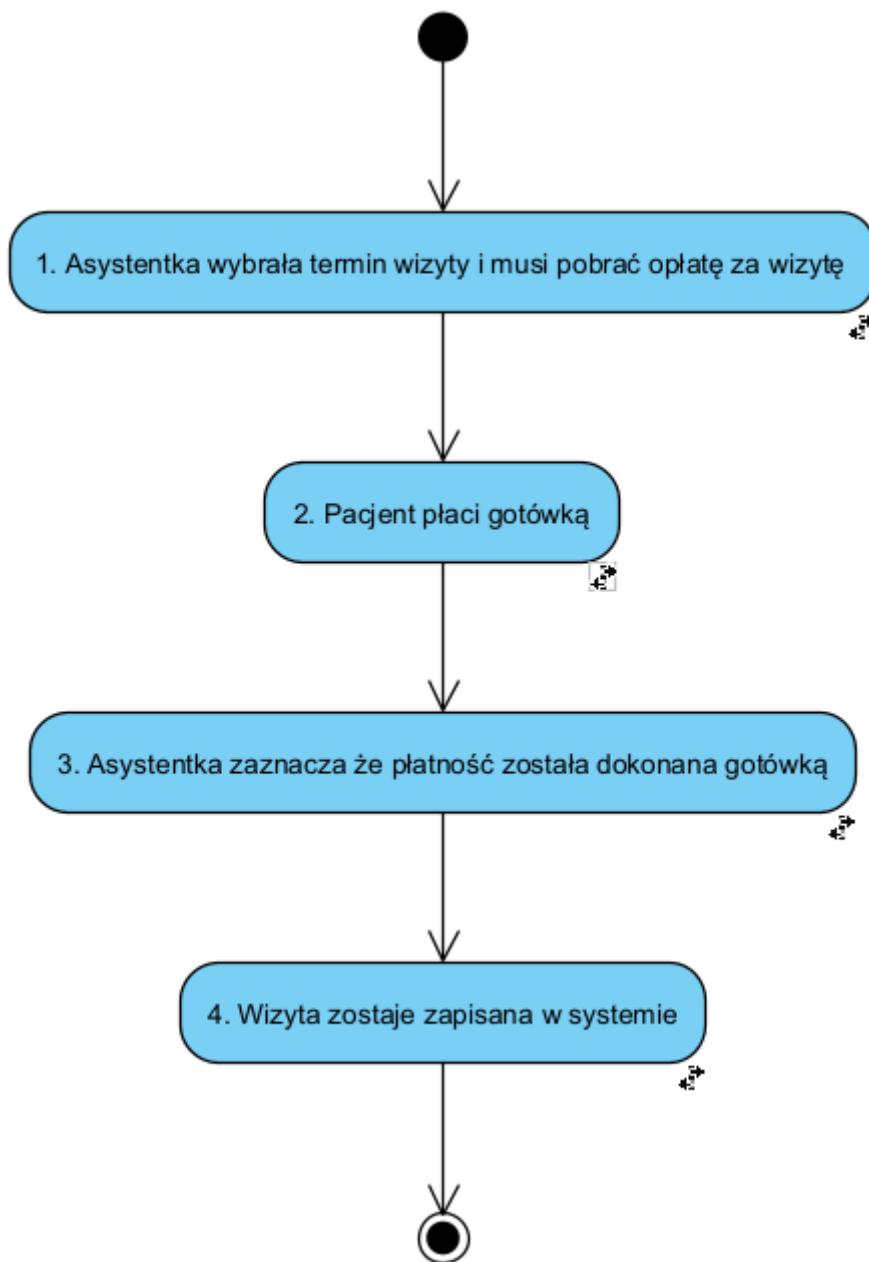
## Zarządzanie kalendarzem

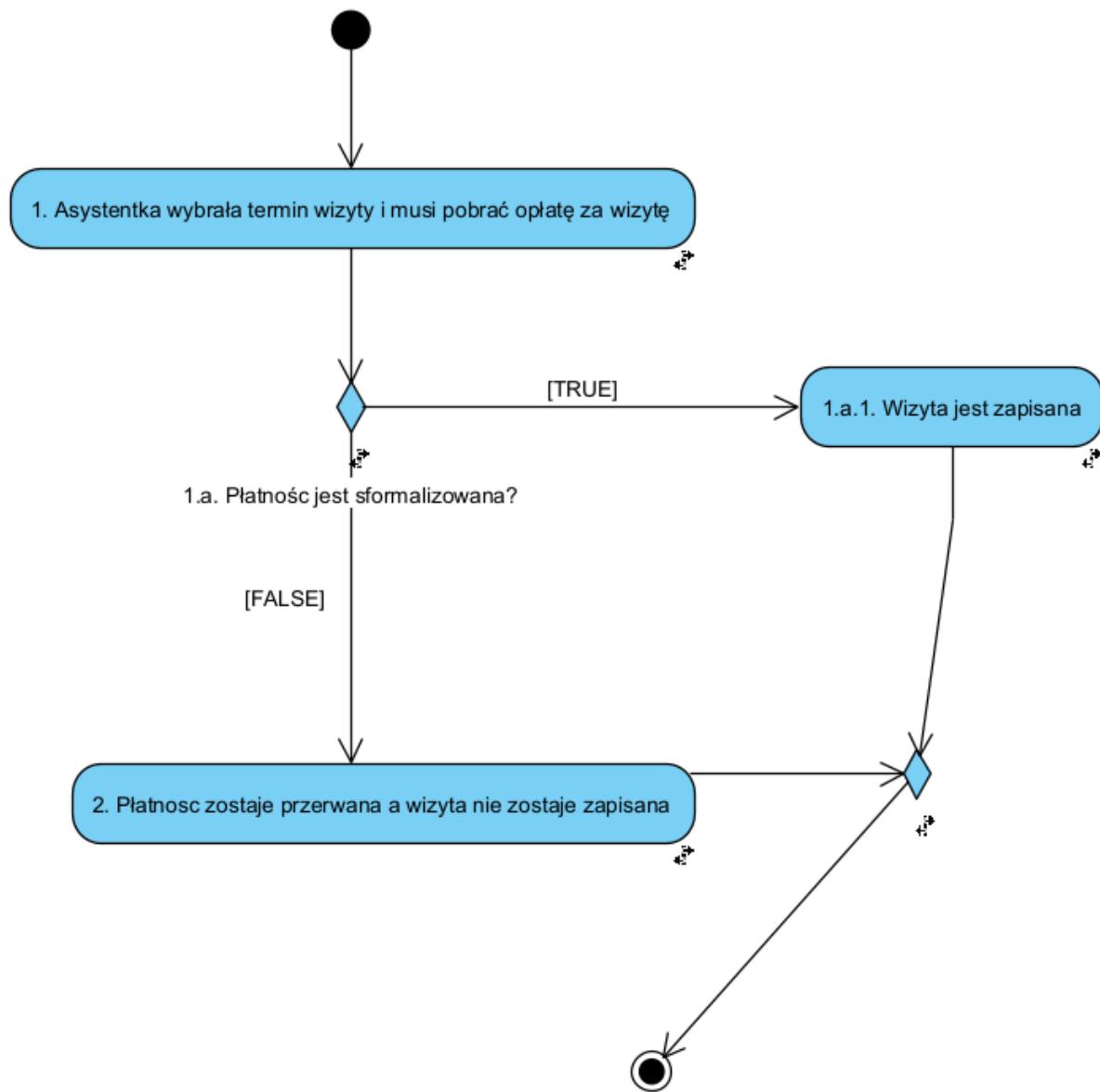


scenariusz alternatywny:

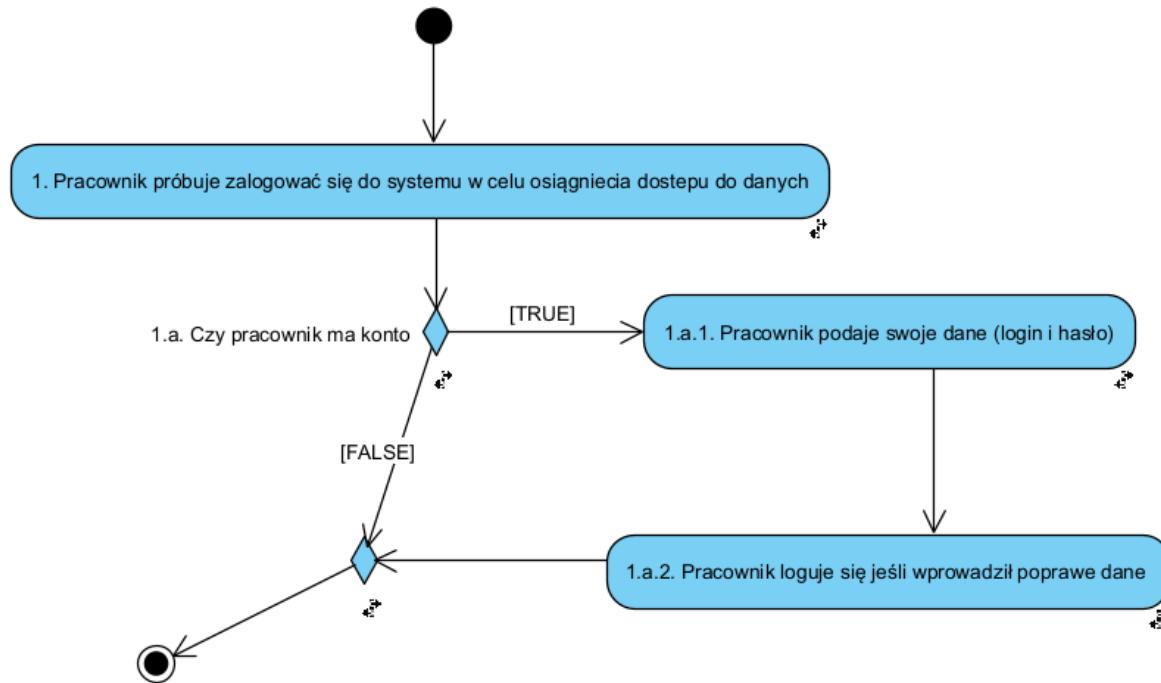


## Opłacanie wizyt

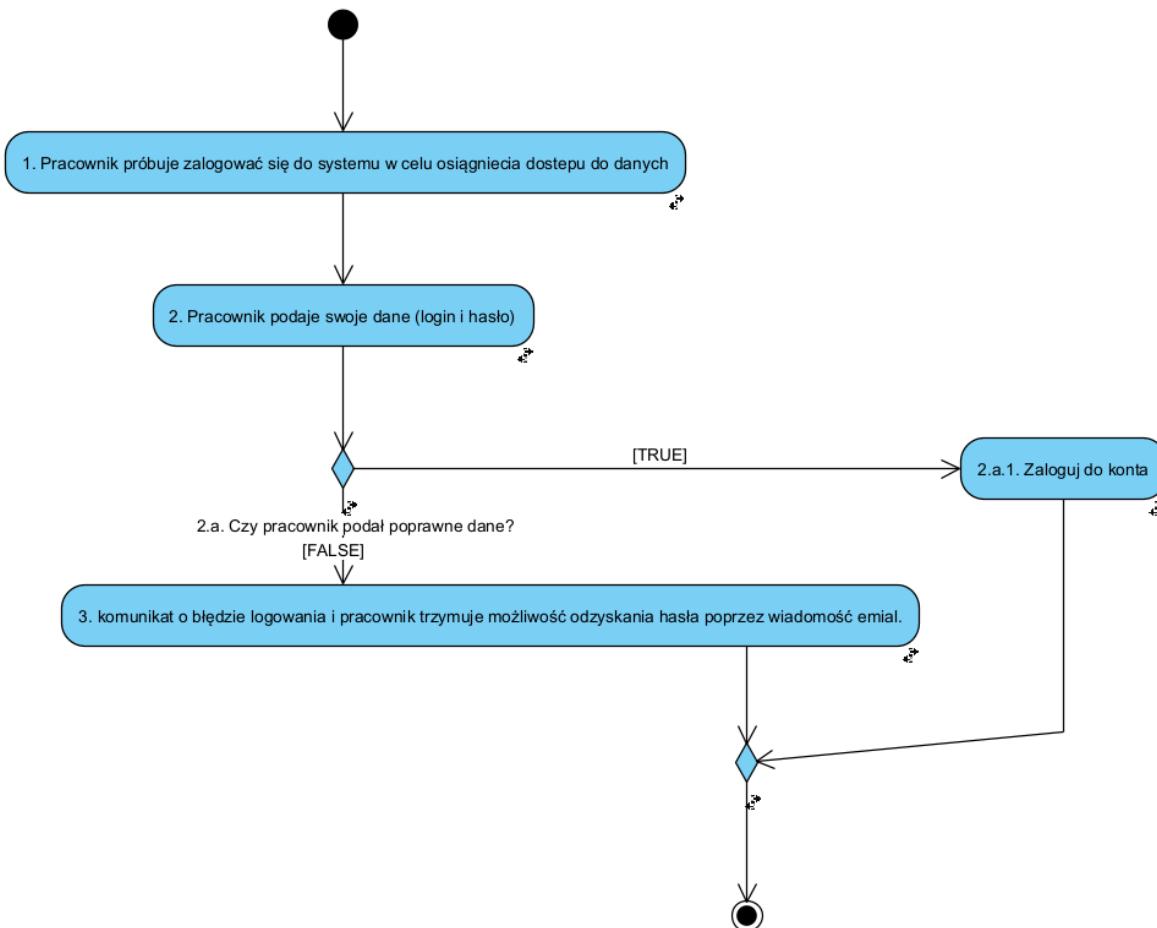


**scenariusz alternatywny:**

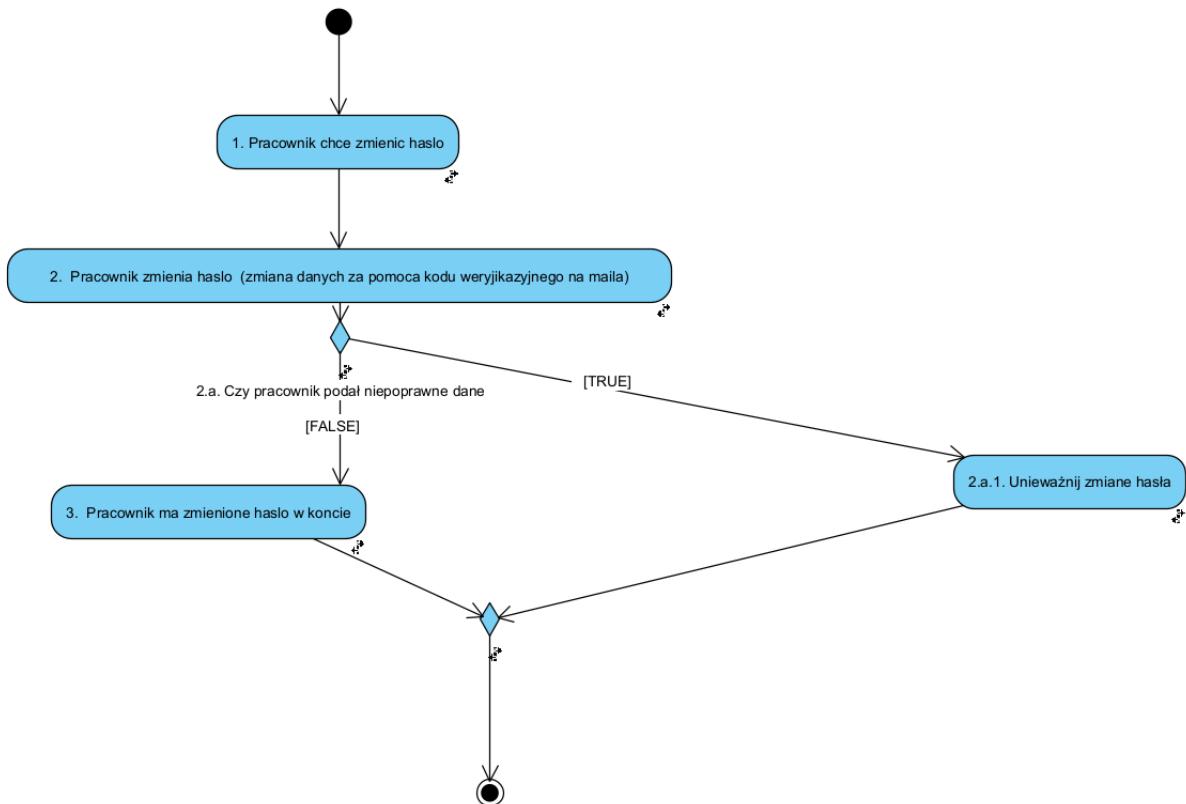
## Logowanie do systemu



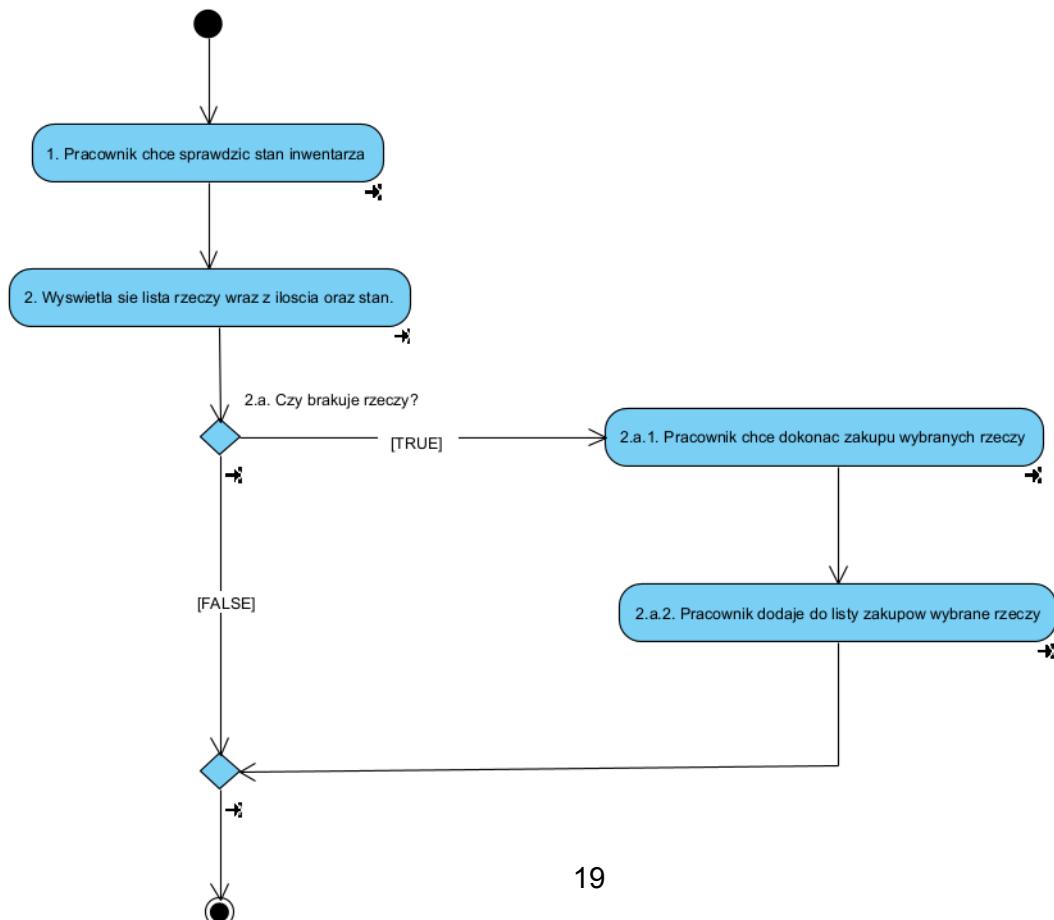
scenariusz alternatywny:



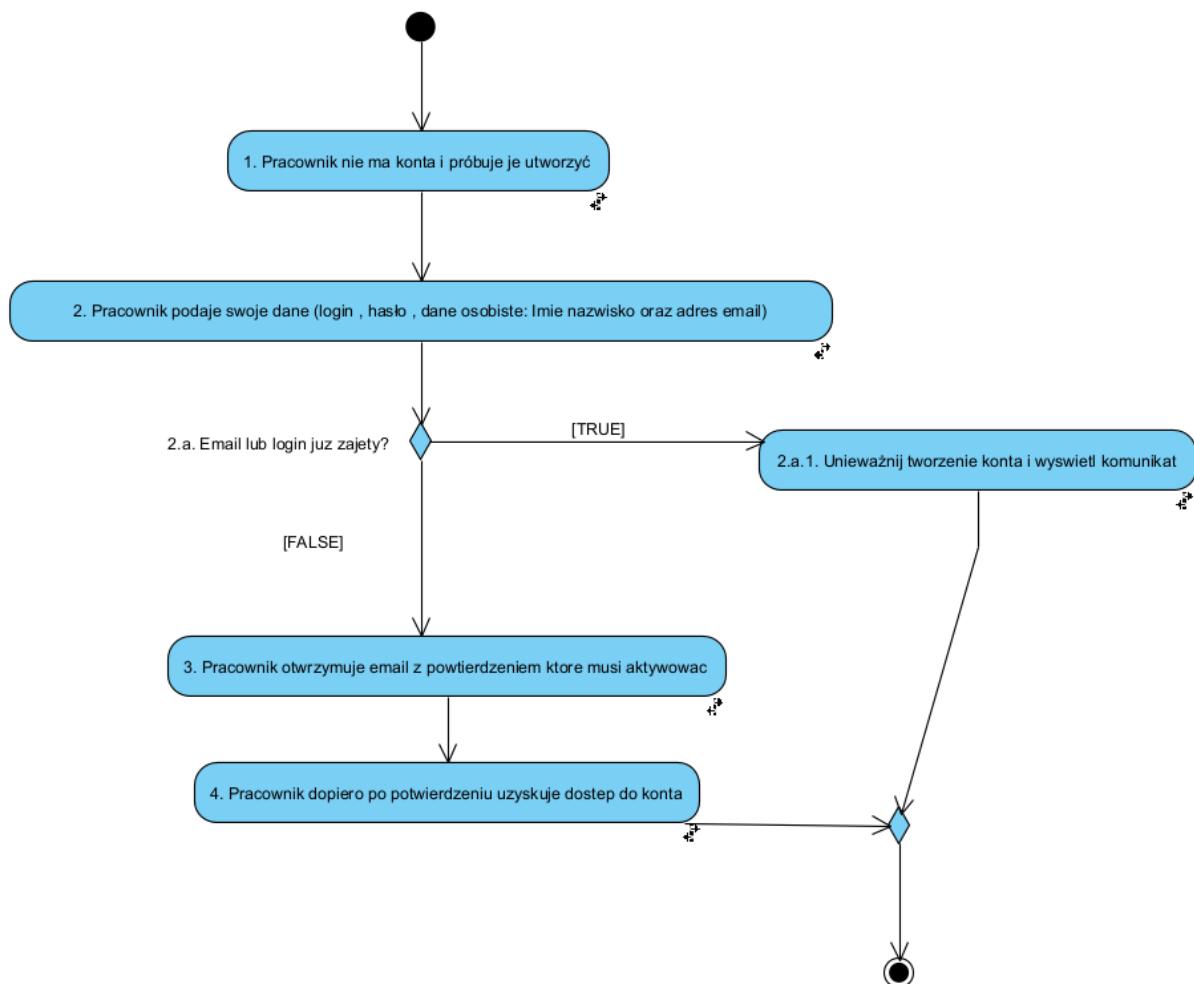
## Zmiana danych



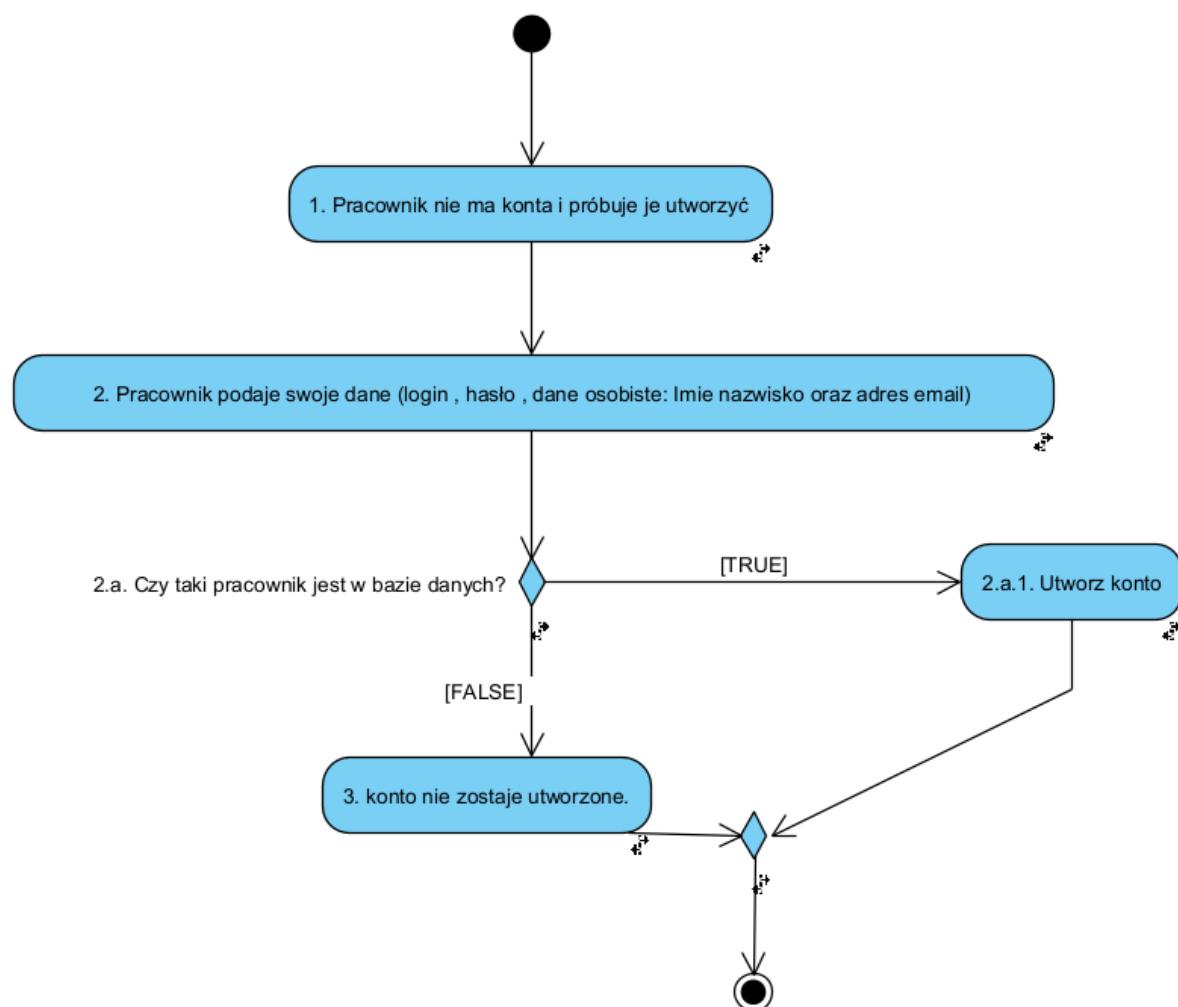
## Zarządzanie inwentarzem



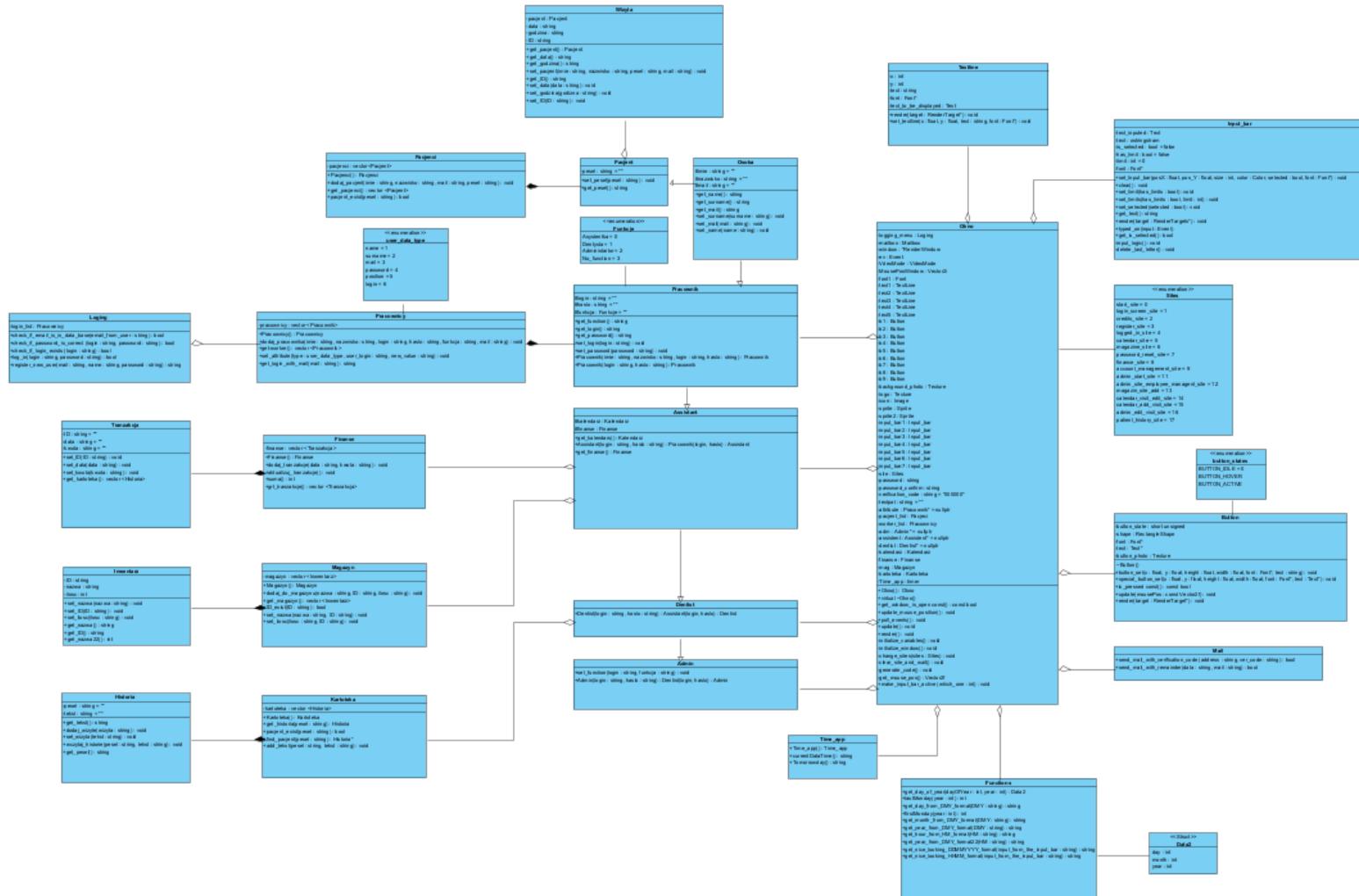
## Zakładanie konta



**scenariusz alternatywny:**



## Diagram klas



## Poszczególne fragmenty diagramu klas:

<pre> <b>Osoba</b> #imie : string = "" #nazwisko : string = "" #mail : string = ""  +get_name() : string +get_surname() : string +get_mail() : string +set_surname(surname : string) : void +set_mail(mail : string) : void +set_name(name : string) : void </pre>	<p>Klasa <i>Osoba</i> odpowiedzialna jest za podstawę każdej kolejnej klasy (np. <i>Assistant</i>) która dziedziczy po tej klasie.</p>
<pre> <b>Pacjent</b> -pesel : string = ""  +set_pesel(pesel : string) : void +get_pesel() : string </pre>	<p>Klasa <i>Pacjent</i> odpowiedzialna jest za dodatkowe informacje o pacjencie. Jest dziedziczona po <i>Osobie</i>.</p>
<pre> <b>Pacjenci</b> -pacjenci : vector&lt;Pacjent&gt;  +Pacjenci() : Pacjenci +dodaj_pacjenta(imie : string, nazwisko : string, mail : string, pesel : string) : void +get_pacjenci() : vector &lt;Pacjent&gt; +pacjent_exist(pesel : string) : bool </pre>	<p><i>Pacjenci</i> odpowiedzialna jest za wektor pacjentów (lista wszystkich pacjentów).</p>

<pre> <b>Wizyta</b> -pacjent : Pacjent -data : string -godzina : string -ID : string  +get_pacjent() : Pacjent +get_data() : string +get_godzina() : string +set_pacjent(imie : string, nazwisko : string, pesel : string, mail : string) : void +get_ID() : string +set_data(data : string) : void +set_godzina(godzina : string) : void +set_ID(ID : string) : void </pre>	<p><b>Klasa <i>wizyta</i></b> odpowiedzialna jest za określenie parametrów wizyty.</p>
<pre> <b>Pracownik</b>   V #login : string = "" #haslo : string = "" #funkcja : Funkcje = ""  +get_function() : string +get_login() : string +get_password() : string -set_login(login : string) : void +set_password(password : string) : void +Pracownik(imie : string, nazwisko : string, login : string, haslo : string) : Pracownik +Pracownik(login : string, haslo : string) : Pracownik </pre>	<p><b>Klasa <i>Pracownik</i></b> odpowiedzialna jest za dołączenie dodatkowych informacji na temat Pracownika.</p>
<pre> <b>&lt;&lt;enumeration&gt;&gt;</b> <b>Funkcje</b> Asystentka = 0 Dentysta = 1 Administartor = 2 No_function = 3 </pre>	<p><b>Klasa <i>Funkcje</i></b> jest to klasa typu enum, określa ona wszystkie możliwe opcje jakimi może być użytkownik.</p>
<pre> <b>Pracownicy</b> -pracownicy : vector &lt;Pracownik&gt; +Pracownicy() : Pracownicy +dodaj_pracownika(imie : string, nazwisko : string, login : string, haslo : string, funkcja : string, mail : string) : void +get_worker() : vector &lt;Pracownik&gt; +set_attribute(type : user_data_type, user_login : string, new_value : string) : void +get_login_with_mail(mail : string) : string </pre>	<p><b>Klasa <i>Pracownicy</i></b> odpowiedzialna jest za wektor pracowników (pełną listę pracowników).</p>

<pre>&lt;&lt;enumeration&gt;&gt; user_data_type  name = 1 surname = 2 mail = 3 password = 4 position = 5 login = 6</pre>	<p>Klasa <i>user_data_type</i> jest to klasa typu enum, służy przy niektórych funkcjach w celu wybrania określonego parametru pracownika.</p>
<pre>Logging  -login_list : Pracownicy  +check_if_email_is_in_data_base(email_from_user : string) : bool +check_if_password_is_correct (login : string, password : string) : bool +check_if_login_exists (login : string) : bool +log_in(login : string, password : string) : bool +register_new_user(mail : string, name : string, password : string) : string</pre>	<p>Klasa <i>Logging</i> odpowiedzialna jest za logowanie oraz rejestrację użytkowników.</p>
<pre>Assistant  #kalendarz : Kalendarz #finanse : Finanse  +get_kalendars() : Kalendarz +Assistant(login : string, haslo : string) : Pracownik(login, haslo) : Assistant +get_finanse() : Finanse</pre>	<p>Klasa <i>Assistant</i> dziedziczy po klasie <i>Pracownik</i> i ma dodatkowe funkcje korzystania z Kalendarza i Finansów.</p>
<pre>Finanse  -finanse : vector &lt;Tranzakcja&gt;  +Finanse() : Finanse +dodaj_tranzakcje(data : string, kwota : string) : void +aktualizuj_tranzakcje() : void +suma() : int +grt_tranzakcje() : vector &lt;Tranzakcja&gt;</pre>	<p>Klasa <i>Finanse</i> jest to klasa odpowiedzialna za historię przychodów gabinetu. Zawiera wektor <i>Transakcji</i>.</p>

<pre> <b>Tranzakcja</b> -ID : string = "" -data : string = "" -kwota : string = ""  +set_ID(ID : string) : void +set_data(data : string) : void +set_kwota(kwota : string) : void +get_kartoteka() : vector &lt;Historia&gt; </pre>	<p><b>Klasa Transakcja</b> odpowiedzialna jest za poszczególne transakcje.</p>
<pre> <b>Magazyn</b> -magazyn : vector &lt;Inwentarz&gt;  +Magazyn() : Magazyn +dodaj_do_magazynu(nazwa : string, ID : string, ilosc : string) : void +get_magazyn() : vector &lt;Inwentarz&gt; +ID_exist(ID : string) : bool +set_nazwa(nazwa : string, ID : string) : void +set_ilosc(ilosc : string, ID : string) : void </pre>	<p><b>Klasa Magazyn</b> jest odpowiedzialna za magazyn rzeczy. Posiada ona wektor <b>Inwentarzy</b> (rzeczy na stanie).</p>
<pre> <b>Inwentarz</b> -ID : string -nazwa : string -ilosc : int  +set_nazwa(nazwa : string) : void +set_ID(ID : string) : void +set_ilosc(ilosc : string) : void +get_nazwa() : string +get_ID() : string +get_nazwa22() : int </pre>	<p><b>Klasa Inwentarz</b> jest odpowiedzialna za poszczególny przedmiot znajdujący się w magazynie.</p>
<pre> <b>Dentist</b> ↓ +Dentist(login : string, haslo : string) : Assistant(login, haslo) : Dentist </pre>	<p><b>Klasa Dentist</b> jest to klasa dziedzicząca po klasie <b>Assistant</b> oraz zawierająca dodatkową funkcję zarządzania historią choroby pacjenta.</p>

<pre>class Kartoteka { private:     vector&lt;Historia&gt; -&gt; kartoteka; public:     Kartoteka() : Kartoteka();     Historia get_historia(string pesel);     bool pacjent_exist(string pesel);     Historia* find_pacjent(string pesel);     void add_tekst(string pesel, string tekst); };</pre>	Klasa <i>Kartoteka</i> jest to klasa posiadająca dostęp do wektora który zawiera wszystkie dane o historii choroby każdego pacjenta w systemie.
<pre>class Historia { private:     string -&gt; pesel;     string -&gt; tekst; public:     Historia();     string get_tekst();     void dodaj_wizyta(string wizyta);     void set_wizyta(string tekst);     void wczytaj_historie(string pesel, string tekst);     string get_pesel(); };</pre>	Klasa <i>Historia</i> jest to klasa opisująca poszczególnego pacjenta a dokładnie jego historię chory czy wizyt.



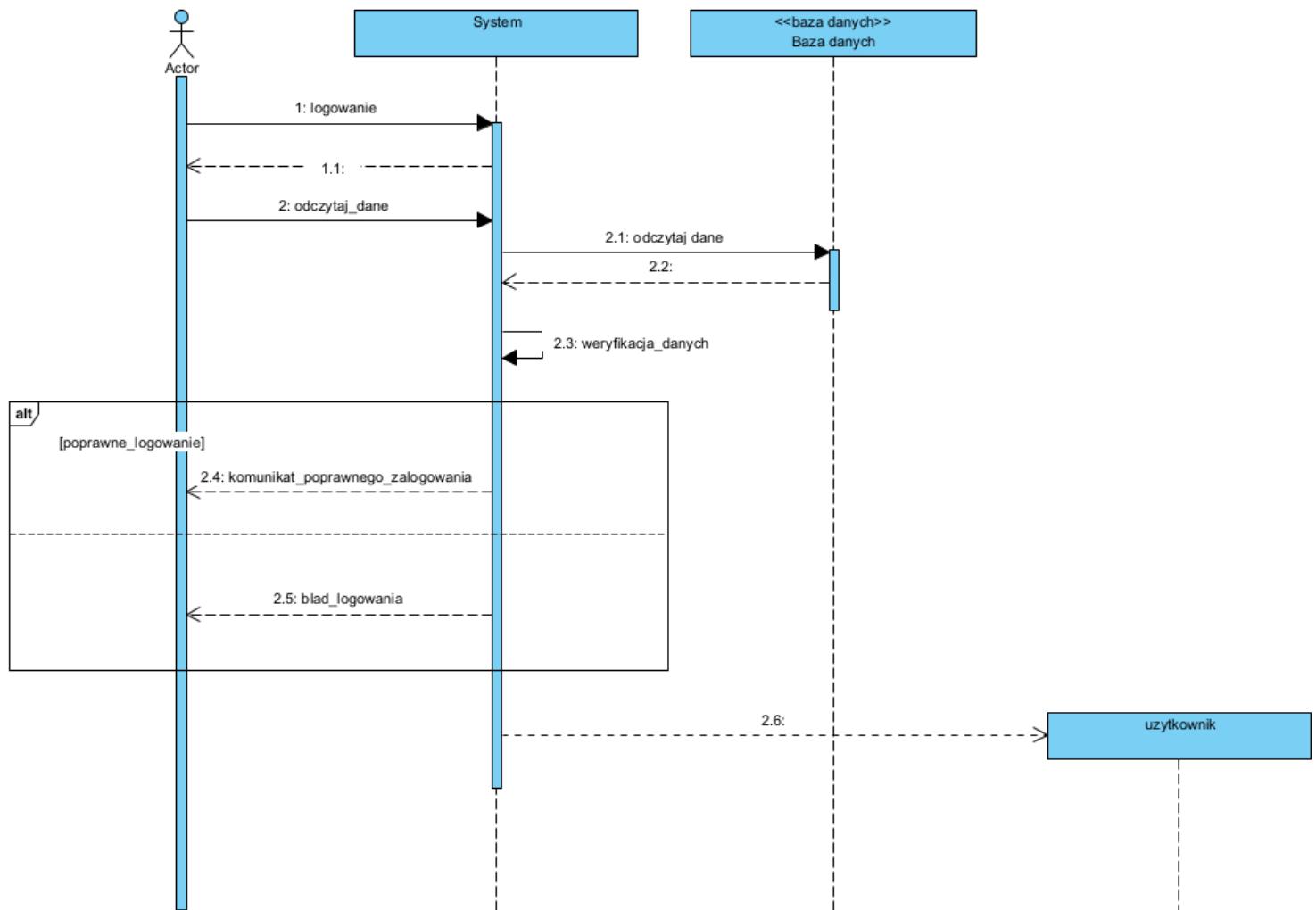
<pre> <b>Textline</b> -<i>x</i> : int -<i>y</i> : int -<i>text</i> : string -<i>font</i> : Font* -<i>text_to_be_displayed</i> : Text +<i>render(target : RenderTarget*)</i> : void +<i>set_textline(x : float, y : float, text : string, font : Font*)</i> : void </pre>	<p>Klasa <i>Textline</i> odpowiada za generowanie określonego napisu.</p>
<pre> <b>Input_bar</b> -<i>text_inputed</i> : Text -<i>text</i> : ostrinjam -<i>is_selected</i> : bool = false -<i>has_limit</i> : bool = false -<i>limit</i> : int = 0 -<i>font</i> : Font* +<i>set_Input_bar(posX : float, pos_Y : float, size : int, color : Color, selected : bool, font : Font*)</i> : void +<i>clear()</i> : void +<i>set_limit(has_limits : bool)</i> : void +<i>set_limits(has_limits : bool, limit : int)</i> : void +<i>set_selected(selected : bool)</i> : void +<i>get_text()</i> : string +<i>render(target : RenderTargets*)</i> : void +<i>typed_on(input : Event)</i> +<i>get_is_selected()</i> : bool -<i>input_logic()</i> : void -<i>delete_last_letter()</i> : void </pre>	<p>Klasa <i>Input_bar</i> odpowiedzialna jest za "wejście danych" do systemu. Głównym zadaniem klasy jest pobieranie danych wprowadzanych z klawiatury.</p>
<pre> &lt;&lt;enumeration&gt;&gt; <b>Sites</b> start_site = 0 login_screen_site = 1 credits_site = 2 register_site = 3 logged_in_site = 4 calendar_site = 5 magazine_site = 6 password_reset_site = 7 finance_site = 8 account_management_site = 9 admin_start_site = 11 admin_site_employee_managent_site = 12 magazin_site_add = 13 calendar_visit_edit_site = 14 calendar_add_visit_site = 15 admin_edit_visit_site = 16 patient_history_site = 17 </pre>	<p>Klasa <i>Sites</i> określa jakie strony może generować metoda <i>render</i> z klasy <i>Okno</i>.</p>

<pre> classDiagram     class Button {         -button_state : short unsigned         -shape : RectangleShape         -font : Font*         -text : Text*         -button_photo : Texture         ~Button()         +button_set(x : float, y : float, height : float, width : float, font : Font*, text : string) : void         +special_button_set(x : float, y : float, height : float, width : float, font : Font*, text : Text*) : void         +is_pressed const() : const bool         +update(mousePos : const Vector2f) : void         +render(target : RenderTarget*) : void     } </pre>	<p>Klasa <b>Button</b> odpowiedzialna jest za generowanie określonego przycisku który może służyć jako zwykły przycisk lub w połączeniu z klasą <b>Input_bar</b> jako przycisk z wejściem.</p>
<pre> classDiagram     class button_states {         &lt;&lt;enumeration&gt;&gt;         BUTTON_IDLE = 0         BUTTON_HOVER         BUTTON_ACTIVE     } </pre>	<p>Klasa <b>button_states</b> jest to klasa typu enum określająca stany jakie może mieć przycisk.</p>
<pre> classDiagram     class Mail {         +send_mail_with_verification_code (address : string, ver_code : string) : bool         +send_mail_with_remainder(data : string, mail : string) : bool     } </pre>	<p>Klasa <b>Mail</b> odpowiada za wysyłanie maili.</p>
<pre> classDiagram     class Time_app {         +Time_app() : Time_app         +currentDataTime() : string         +Tomorrowday() : string     } </pre>	<p>Klasa <b>Time_app</b> odpowiedzialna jest za dostarczanie daty.</p>

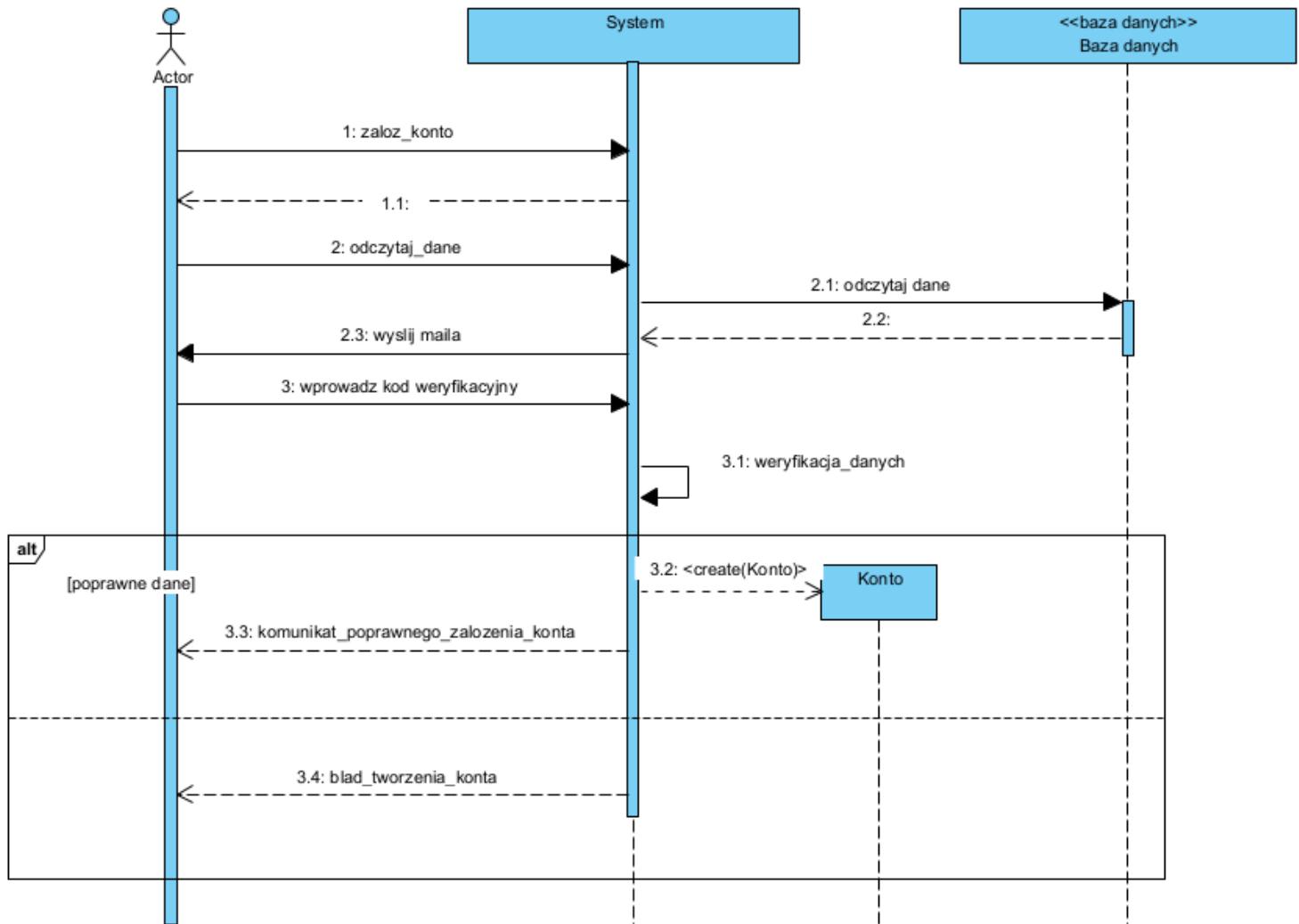
<pre>Functions +get_day_of_year(dayOfYear : int, year : int) : Data2 +lastMonday(year : int) : int +get_day_from_DMY_format(DMY : string) : string +firstMonday(year : int) : int +get_month_from_DMY_format(DMY : string) : string +get_year_from_DMY_format(DMY : string) : string +get_hour_from_HM_format(HM : string) : string +get_year_from_DMY_format22(HM : string) : string +get_nice_looking_DDMYYYYY_format(input_from_the_input_bar : string) : string +get_nice_looking_HHMM_format(input_from_the_input_bar : string) : string</pre>	Jest to header zawierający dodatkowe funkcje pomocne dla klasy Okno
<pre>&lt;&lt;Struct&gt;&gt; Data2 day : int month : int year : int</pre>	Jest to struktura określająca format daty.

## Diagramy sekwencji

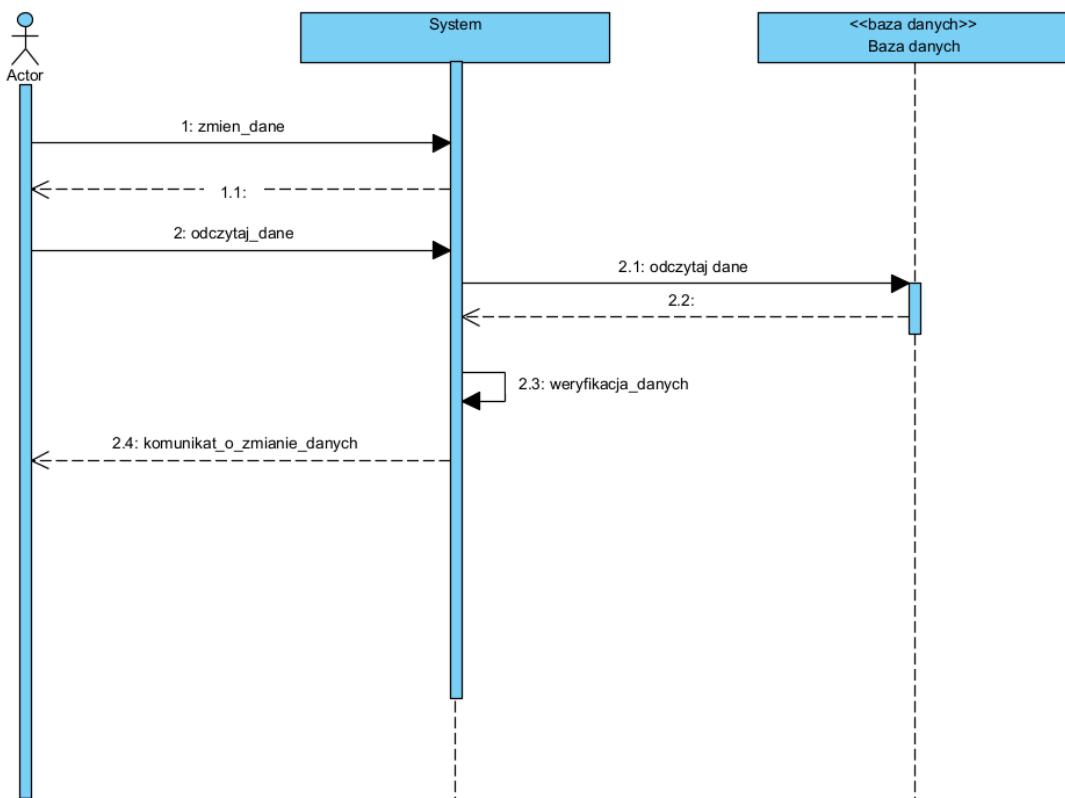
### Logowanie do systemu



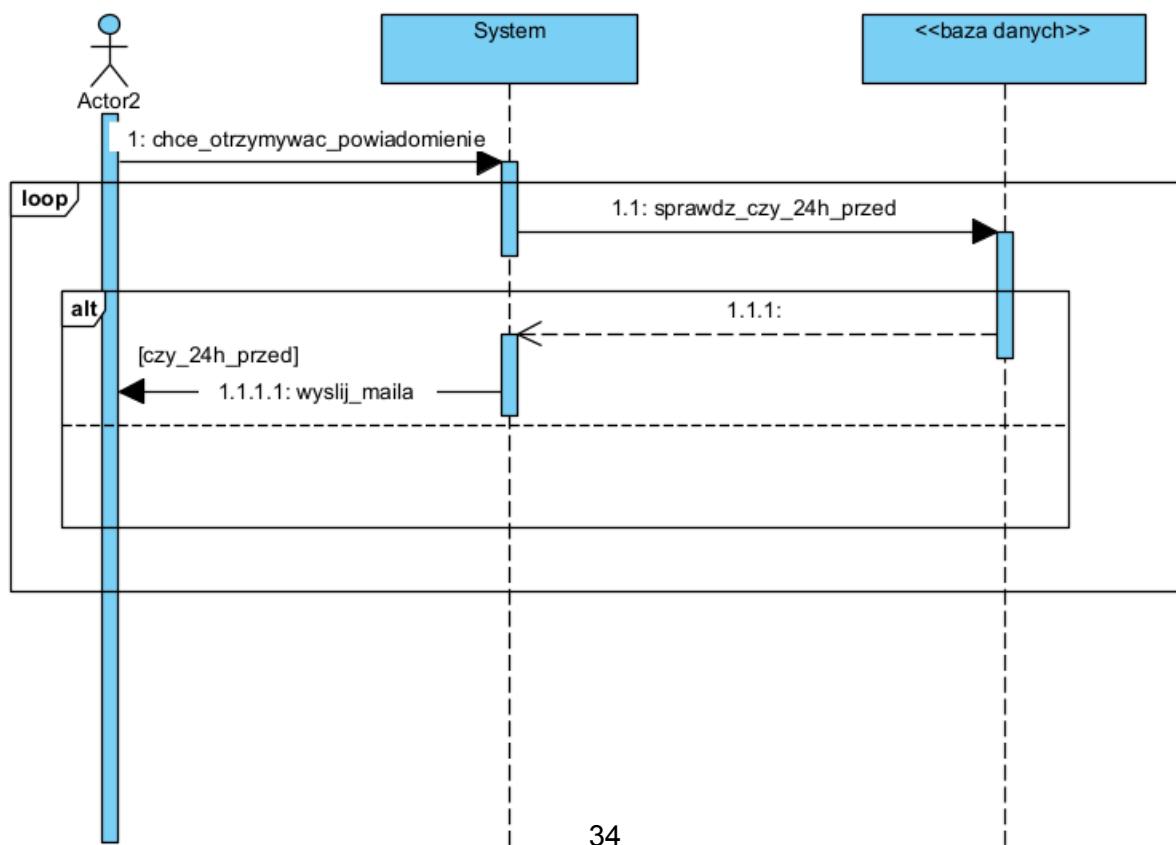
## Zakładanie konta



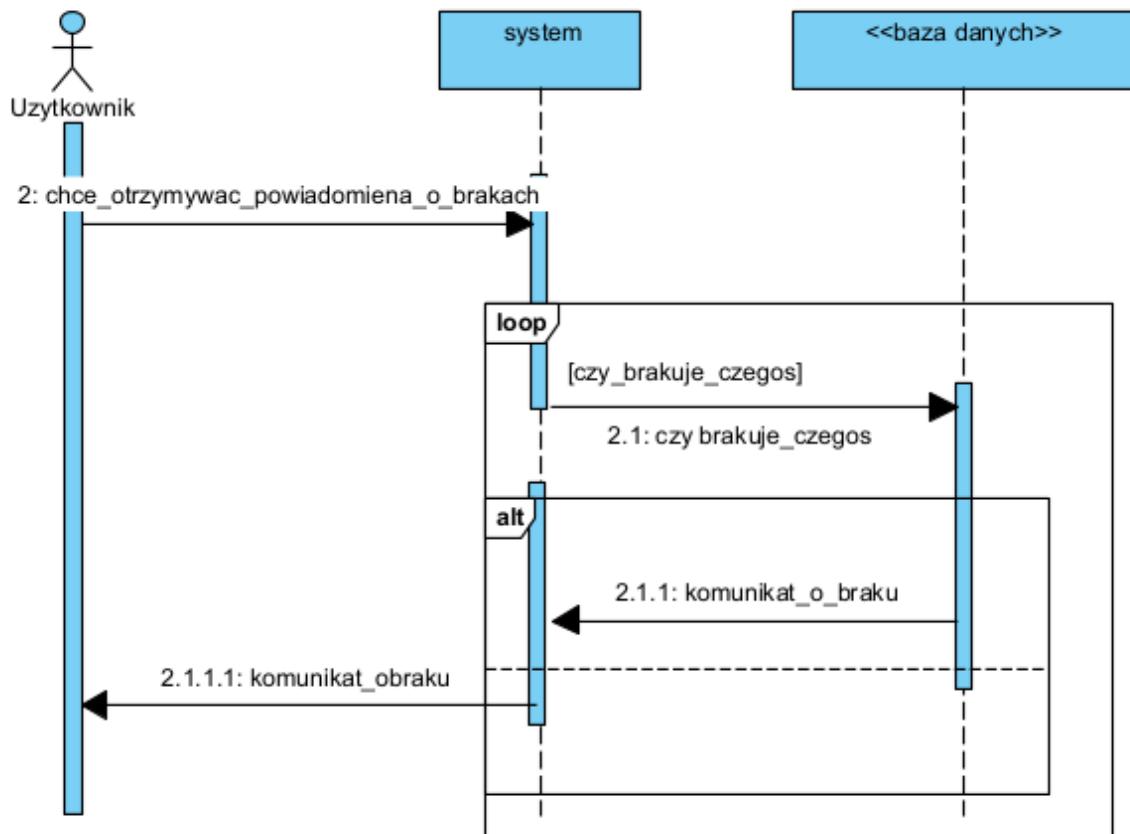
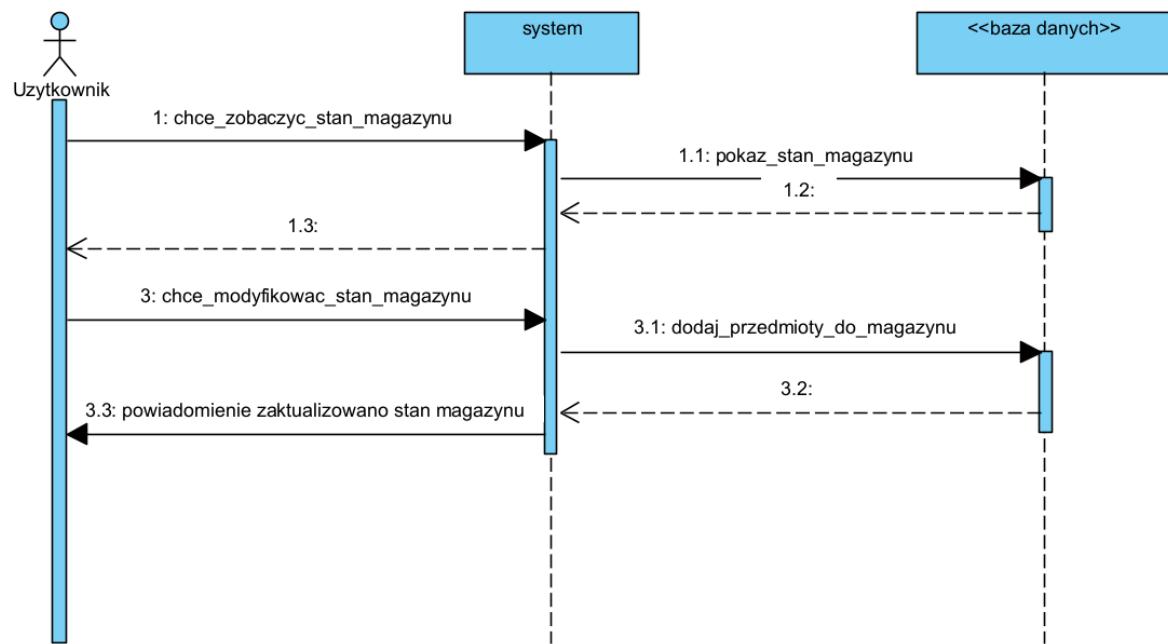
## Modyfikacja danych konta



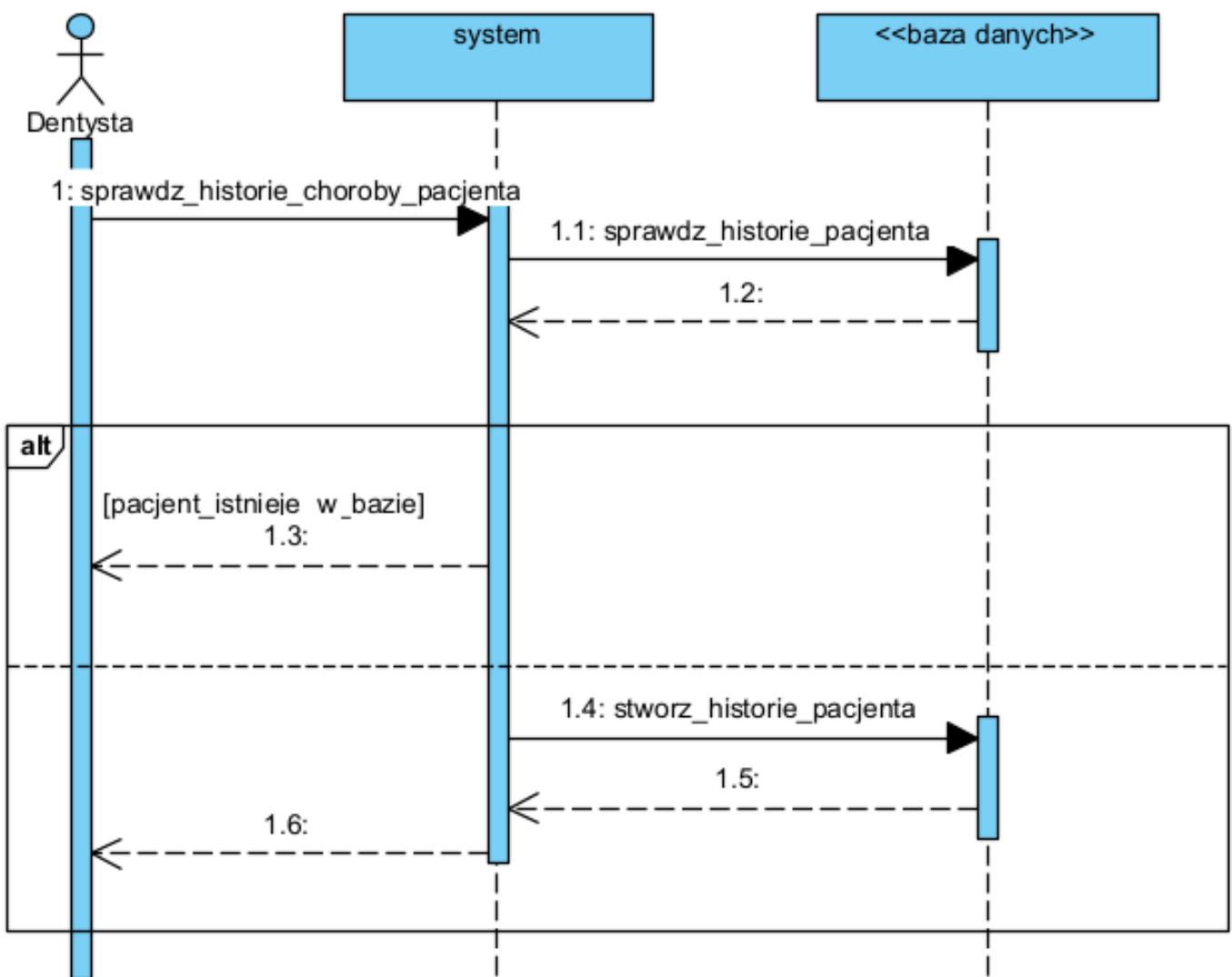
## Otrzymywanie przypomnień o wizytach



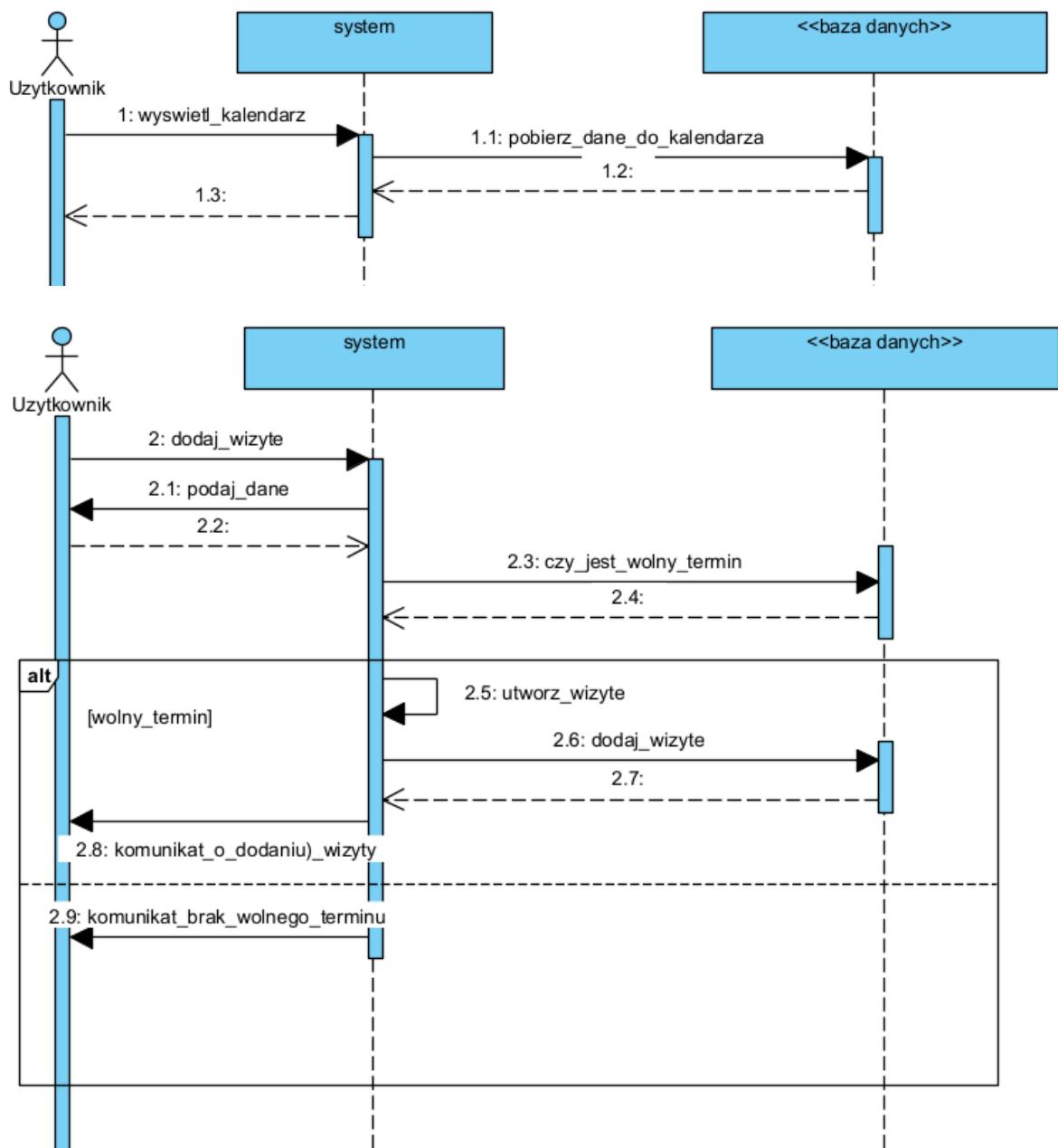
## Zarządzanie inwentarzem



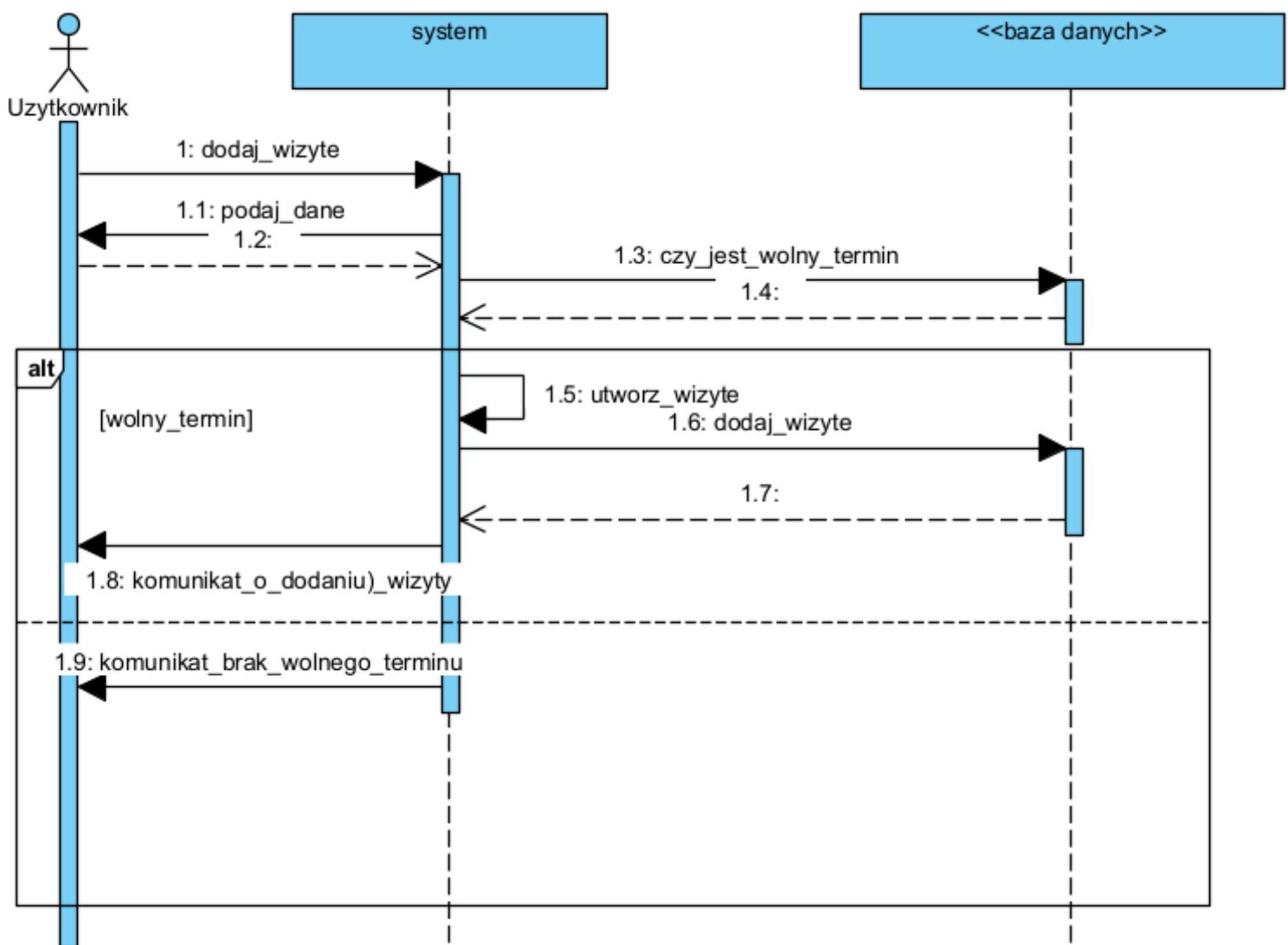
## Zarządzanie archiwum



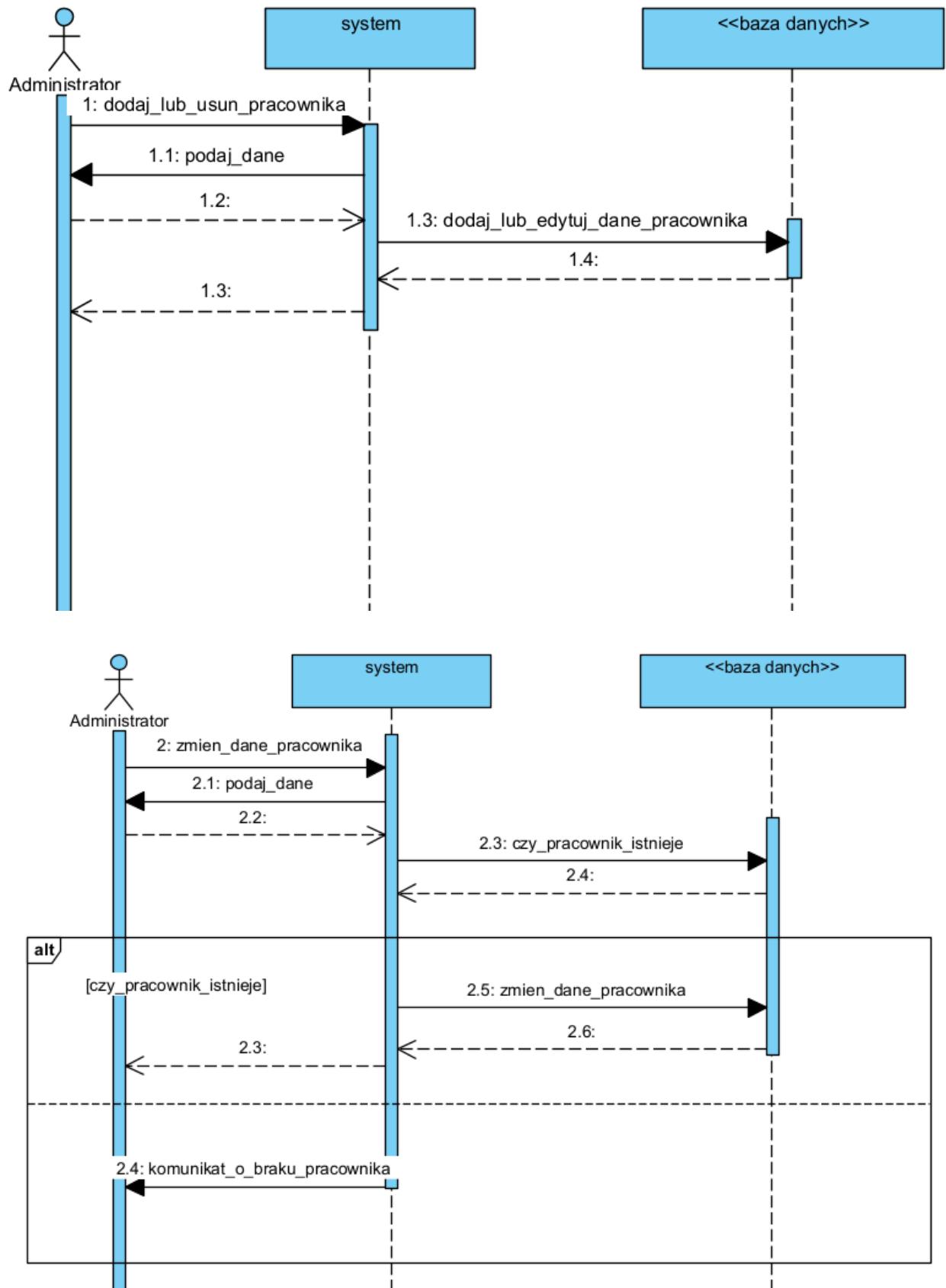
## Zarządzanie kalendarzem



## Zarządzanie wizytami

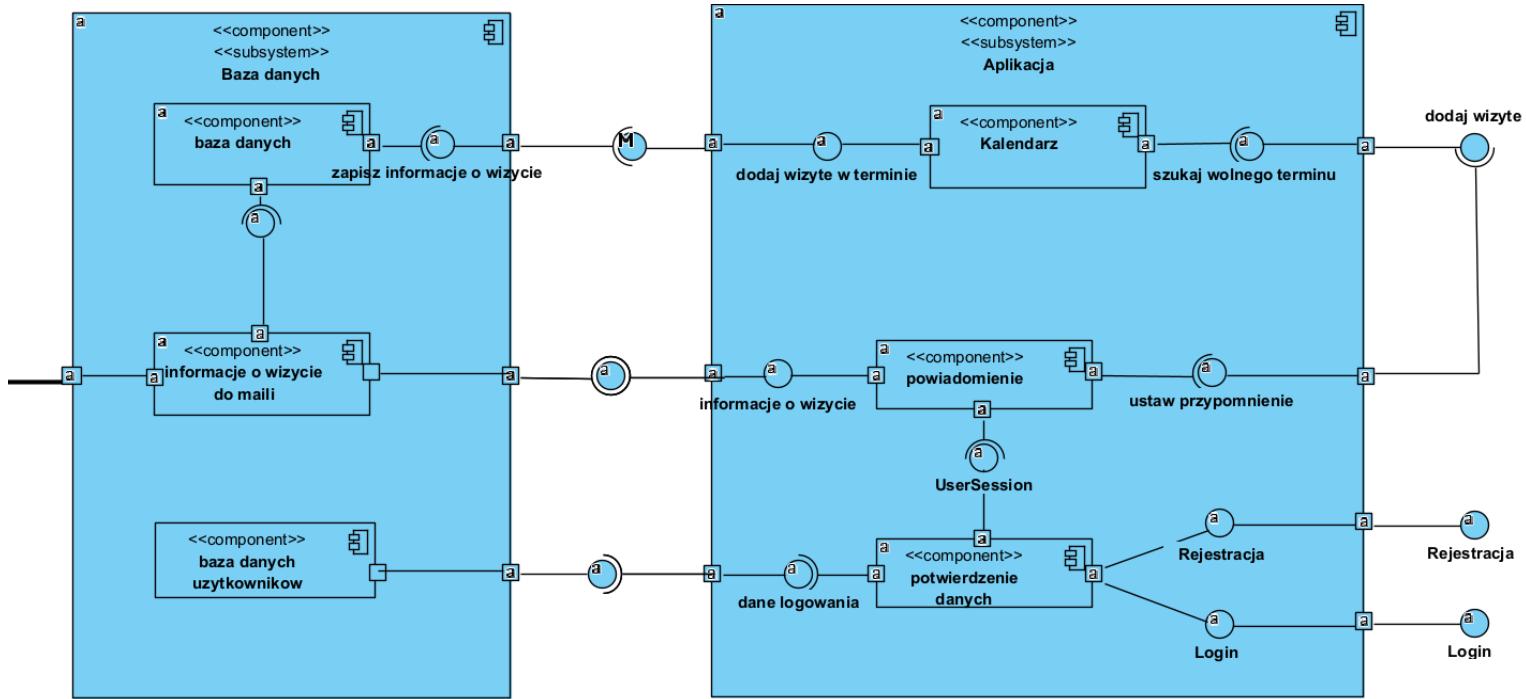


## Zarządzanie systemem

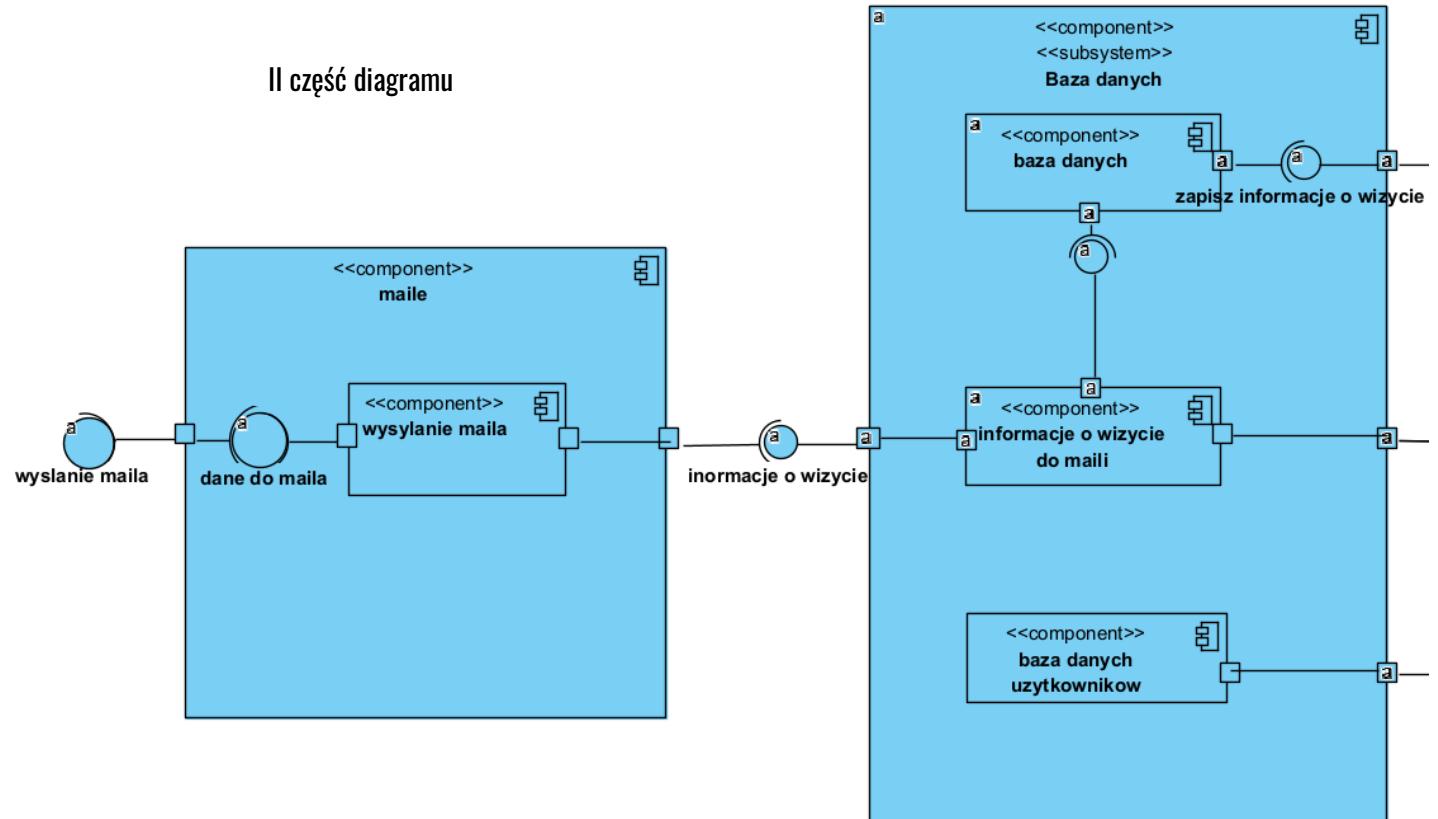


## Diagram komponentów

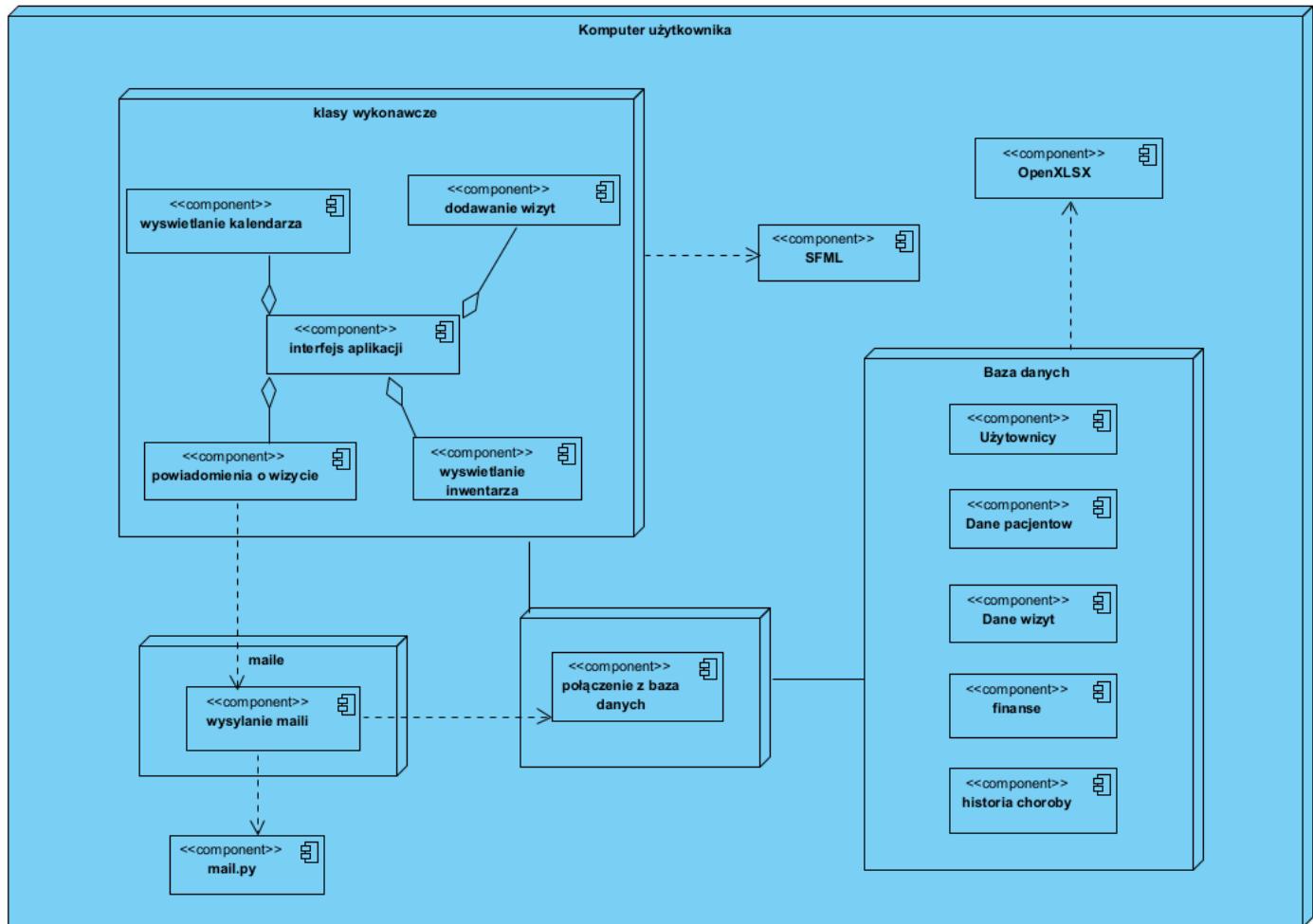
### I część diagramu



### II część diagramu



## Diagram wdrożenia



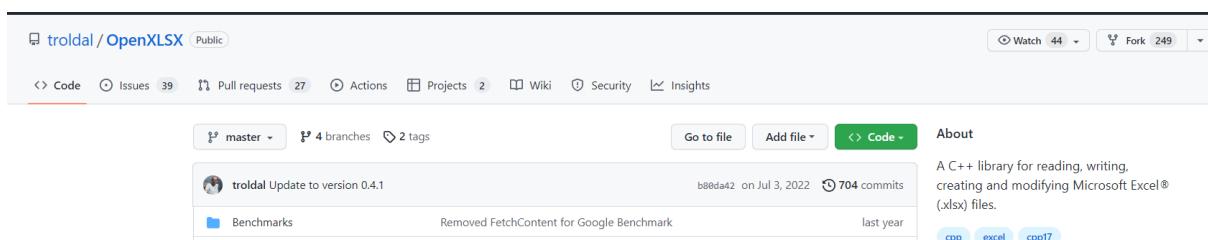
## 6. Wykaz zastosowanych rozwiązań informatycznych

### Biblioteki:

**SFML** - jest to biblioteka graficzna do języka C++. Za pomocą sfml możesz napisać program 2D. W naszym przypadku użyliśmy tej biblioteki do tworzenia wszystkich elementów graficznych w naszej aplikacji, łącznie z interfejsem i wszystkimi przyciskami.

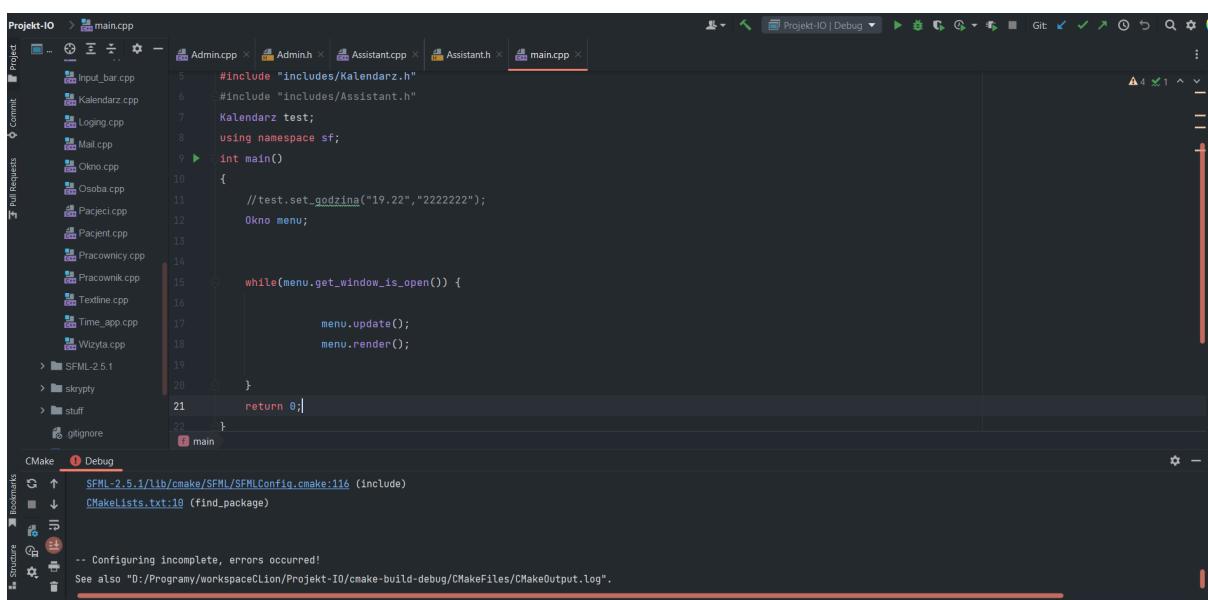


**OpenXLSX** - jest to biblioteka służąca do modyfikacji danych z programu Microsoft Excel. Użyliśmy jej do przetwarzania danych pobranych z tabel w excelu i ich zmian. W tabelach znajdują się dane pacjentów, dane o wizytach czy dane do logowania. Dzięki OpenXLSX w łatwy sposób mogliśmy te dane modyfikować.



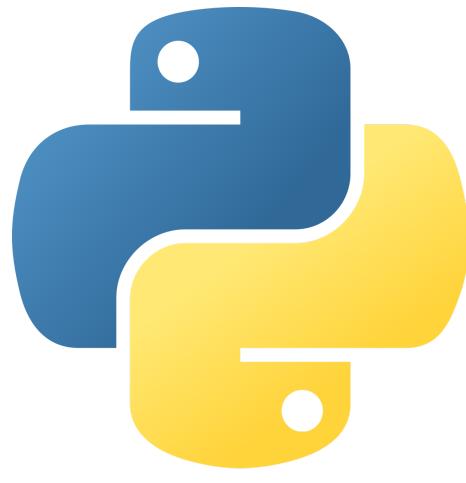
## Środowisko:

**Clion** - wieloplatformowe zintegrowane środowisko programistyczne języka C++ produkcji JetBrains. Clion Umożliwił nam efektywną pracę nad naszym projektem dzięki wielu wbudowanym narzędziom, między innymi posiada podpowiedzi pojawiające się przy tworzeniu kodu czy możliwość pobierania aktualnej wersji z GitHuba. Interfejs środowiska jest bardzo przejrzysty i intuicyjny co pozwoliło uniknąć domyślania się i szukania odpowiednich funkcji, a to przełożyło się na przyspieszenie pracy nad naszym projektem.



## Technologie:

**C++** - Język programowania, którego użyliśmy w największym stopniu do stworzenia naszego projektu. Wykorzystaliśmy go do napisania wszystkich klas i interfejsu użytkownika. Wybrałyśmy język C++ ze względu na możliwość stworzenia w nim graficznej aplikacji przy użyciu odpowiednich bibliotek.



**Python** - Mimo, że w większości naszego projektu używaliśmy C++, python znalazło swoje miejsce w segmencie do wysyłania maili. Python bardzo ułatwił nam to zadanie, które przy użyciu C++ wydawało się niemal niewykonalne.

## Inne użyte rozwiązania:

- Przykładowe powiadomienie na poczcie



Klinika Dentystyczna <gabinetdentystycznyiodent@gmail.com>

do mnie ▾

Dzień dobry,

Przypominamy o wizycie w dniu: 19.01.2023

Proszę o informację zwrotną z potwierdzeniem wizyty

Prozrawiamy,

Zespół IO-Dent

- **Baza danych w excelu**

	A	B	C	D	E	F
1	//	//	//	//	//	//
2	Mateusz	Lugowski	222222222222	20.01.2023	15.00	lugowski.mat2
3	Mateusz	Lugowski	222222222222	12.01.2023	11.11	lugowski.mat3
4	Mateusz	Lugowski	222222222222	19.01.2023	09.00	lugowski.mat4
5	Wiktor	Kowalski	111111111111	19.01.2023	14.00	wiktor.k.20025
6	Wiktor	Kowalski	111111111111	20.01.2023	14.00	wiktor.k.20026
7	Wiktor	Kowalski	111111111111	21.01.2023	08.10	wiktor.k.20027
8	Amelka	Knap	333333333333	19.01.2023	10.00	amelia244@i8
9	Michał	Piekarski	444444444444	20.01.2023	12.30	michalpiekarski9
10	Mateusz	Kuzera	555555555555	19.01.2023	11.00	mateusz.kuzera10

Fragment bazy danych z danymi dodanymi do kalendarza pacjentów. Od lewej imię, nazwisko, numer pesel, data i godzina umówionej wizyty oraz adres email pacjenta.

- **Photoshop**



Użyliśmy tego programu do tworzenia grafik widniejących w naszym projekcie aplikacji gabinetu dentystycznego. Po prawie widnieje jedna z grafik, przedstawiająca logo naszego gabinetu.

## 7. Przykłady współpracy zespołu

### GitHub Desktop:

The screenshot shows the GitHub Desktop application interface. On the left, there's a sidebar with a list of commits from the 'Projekt-IO' repository, including changes related to adding and returning visitors, starting a calendar, and fixing bugs. The main area displays a 'Quick fix' for a file named 'Okno.cpp'. The diff shows several lines of C++ code being modified, with some lines highlighted in red and green. The commit message for the current branch ('Kalendarz') is 'MKuzera, ironnext22 - 6 days ago'.

GitHub Desktop oferuje graficzny interfejs użytkownika do korzystania z systemu Git, dzięki czemu nie musieliszy polegać na poleceniach tekstowych pisanych w konsoli poleceń Git. Pozwolił nam na sprawne przełączanie się między branchami czy sprawdzanie historii commitów. Umożliwił cofnięcie ostatniego commita czy łatwą zmianę gałęzi, w celu wykonania np. testu nowego rozwiązania.

**Wszystkie gałęzie (Branches) w naszej aplikacji:**

The screenshot shows a list of branches in a GitHub interface. At the top, it says 'Default branch' with a 'main' branch listed. Below that is a section titled 'Your branches' containing 'OPEN\_XLSX' and 'Mati\_test' branches. Under 'Active branches', there are several branches: 'Kalendarz', 'Kalendarz\_v012', 'magazyn', 'OPEN\_XLSX', and 'MENU\_INIT\_VER'. Each branch entry includes the last update time and a 'New pull request' button.

## GitHub:

**Contributors 3**

Contributor	Avatar
ironnext22	
MKuzera	
MTX77 MTX	

GitHub pozwolił nam na jednoczesną pracę nad projektem, z własnych komputerów oddalonych od siebie. Każdy z nas mógł pracować na swoim branchu i następnie ukończone rozwiązanie dodać do projektu. Umożliwiał w łatwy sposób obserwację naszej pracy i postępów co motywowało nas do dalszej pracy. Głównie korzystaliśmy z jego desktopowej wersji - GitHub Desktop.

Pseudonimy pod jakimi pracowaliśmy:

ironnext22 - Wiktor Kowalski

MTX77 - Mateusz Ługowski

Mkuzera - Mateusz Kuzera

Foldery i pliki w naszej aplikacji, wraz z ostatnimi commitami i ostatnią datą aktualizacji:

IO PU	Add files via upload	3 months ago
OpenXLSX	podpięty OpenXLSX	last week
Package	nazwa okna git	2 days ago
SFML-2.5.1	podpięty sfml	2 months ago
cmake-build-debug	nazwa okna	2 days ago
includes	format v2	2 days ago
skrypty	skrypty	last week
stuff	Buttons updated with new look, also new background.	2 weeks ago
.gitignore	123	last week
CMakeLists.txt	Added	2 days ago
LICENSE	Initial commit	3 months ago
README.md	Update README.md	2 weeks ago
arial.ttf	Dodano czcionke , okno dziala , na biezaco pobiera i wyświetla kody ...	2 weeks ago
main.cpp	kalendarz uprawnienia	2 days ago

## Trello:

Głównym zastosowaniem Trello było monitorowanie postępów oraz tworzenie listy zadań do wykonania. Aplikacja umożliwiła nam w łatwy sposób zaplanowanie i realizację założeń w odpowiedniej kolejności.

Monitorowanie postępów przy pomocy list trello:

The image displays three Trello boards side-by-side, each with a pink header and a white body. The first board is titled 'Kod Zrobione' and contains cards for tasks like 'rejestracja za pomocą emaila loginu i hasła - imię , nazwisko , role ustaw na default' and 'blad maila dla pacjentow'. The second board is titled 'Diagramy zrobione' and shows a complex sequence diagram with multiple nodes and arrows. The third board is titled 'Dokumentacja zrobione' and lists various documents such as 'Słownik pojęć i terminów', 'Specyfikacja projektu', and 'Wymagania funkcjonalne i niefunkcjonalne'.

Kod Zrobione	Diagramy zrobione	Dokumentacja zrobione
rejestracja za pomocą emaila loginu i hasła - imię , nazwisko , role ustaw na default	Sequence diagram showing interactions between multiple components.	Słownik pojęć i terminów
blad maila dla pacjentow		Specyfikacja projektu
zmiana zmien_nazwisko(string naziwsko login)		Metodyka wytwarzania
zmiana imienia w bazie - zmien_imie(string imie, string login)		Wymagania funkcjonalne i niefunkcjonalne
admin: zarzadzanie pracownikami		Wykaz zastosowanych framework-ów, bibliotek, środowiska, technologii
zmien haslo (string haslo , string login)		Testy jednostkowe
+ Dodaj kartę	+ Dodaj kartę	Przykład refaktoryzacji kodu
		Wzorzec projektowy
		Instrukcja użytkowania programu
		+ Dodaj kartę

## Przykłady naszych list:

**Do zrobienia** ...

- baza dancyh dla magazynu: lista z rzeczami [przedmiot/ilosc]
- kalendarz
- admin: zarzadzanie kalendarzem
- admin: zarzadzanie magazynem
- dentysta: edycja historii choroby
- asystentka: dodawanie pacjentów
- usun\_osobe\_offline(login)
- dodac te log\_os\_offline etc do admin\_pracownicy\_site
- usunac fmt chyba xD
- naprawic wyswietlanie ladne , cos z sfml (Okno::set\_table\_for\_admin\_site)
- fix: multiclick -> one click
- refresh .xsls

+ Dodaj kartę 

**WAZNE** ...

- naprawic blad z (klik ma byc jednoklikiem a nie serią klikniec)
- Dodac zabezpieczenie przy tworzeniu konta (czy nie istanieje juz taki email) bo sie wywala

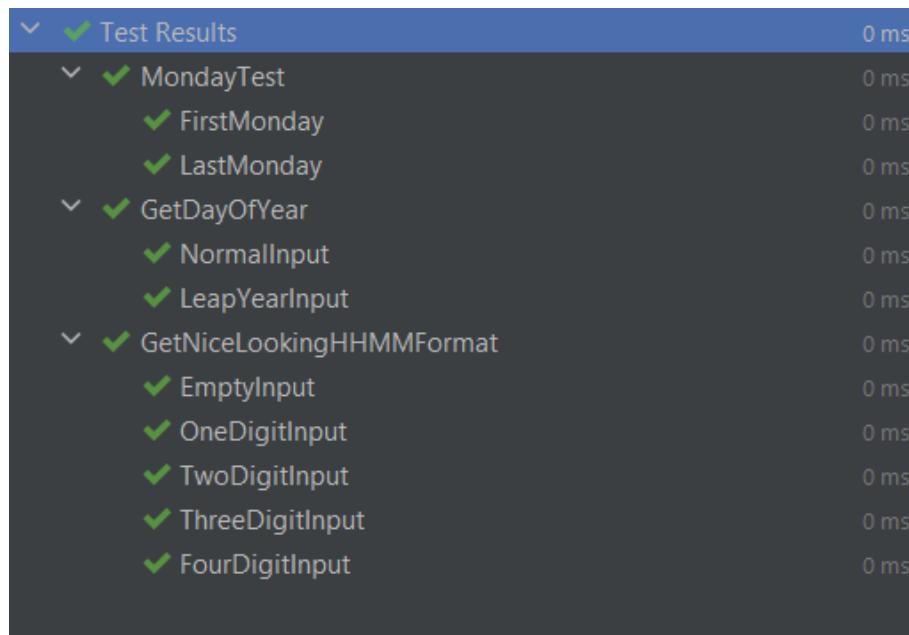
+ Dodaj kartę 

**na potem** ...

- wysyłanie emaili 24h przed wizyta
- wysyłanie emaili z verification key

+ Dodaj kartę 

## 8. Testy jednostkowe



```
#include "gtest/gtest.h"
#include <gtest/gtest.h>
#include "../includes/Functions.h"
TEST(GetNiceLookingHHMMFormat, EmptyInput) {
    std::string input = "";
    EXPECT_EQ("HH.MM", get_nice_looking_HHMM_format( input_from_the_input_bar: input));
}

TEST(GetNiceLookingHHMMFormat, OneDigitInput) {
    std::string input = "1";
    EXPECT_EQ("1H.MM", get_nice_looking_HHMM_format( input_from_the_input_bar: input));
}

TEST(GetNiceLookingHHMMFormat, TwoDigitInput) {
    std::string input = "11";
    EXPECT_EQ("11.MM", get_nice_looking_HHMM_format( input_from_the_input_bar: input));
}

TEST(GetNiceLookingHHMMFormat, ThreeDigitInput) {
    std::string input = "111";
    EXPECT_EQ("11.1M", get_nice_looking_HHMM_format( input_from_the_input_bar: input));
}

TEST(GetNiceLookingHHMMFormat, FourDigitInput) {
    std::string input = "1111";
    EXPECT_EQ("11.11", get_nice_looking_HHMM_format( input_from_the_input_bar: input));
}
```

```
#include "gtest/gtest.h"
#include "../includes/Functions.h"
TEST(GetDayOfYear, NormalInput) {
    int dayOfYear = 32;
    int year = 2020;
    Date2 expected = { .day: 1, .month: 2, .year: 2020};
    EXPECT_EQ(expected, get_day_of_year(dayOfYear, year));
}
TEST(GetDayOfYear, LeapYearInput) {
    int dayOfYear = 60;
    int year = 2020;
    Date2 expected = { .day: 29, .month: 2, .year: 2020};
    EXPECT_EQ(expected, get_day_of_year(dayOfYear, year));
}
```

```
#include "gtest/gtest.h"
#include "../includes/Functions.h"
TEST(MondayTest, FirstMonday)
{
    EXPECT_EQ(firstMonday( year: 2022), 3);
    EXPECT_EQ(firstMonday( year: 2023), 2);
    EXPECT_EQ(firstMonday( year: 2024), 1);
}
TEST(MondayTest, LastMonday)
{
    EXPECT_EQ(lastMonday( year: 2022), 360);
    EXPECT_EQ(lastMonday( year: 2023), 359);
    EXPECT_EQ(lastMonday( year: 2024), 364);
}
```

## 9. Przykład refaktoryzacji kodu

Metoda `get_nice_looking_HH_MM_format` zajmowała ponad 45 linijek.

Przed:

```
std::string get_nice_looking_HHMM_format(std::string input_from_the_input_bar){  
  
    std::string newstr = "", str2;  
    if(input_from_the_input_bar == ""){return "HH.MM";}  
  
    else if(input_from_the_input_bar.length() == 1){  
        str2 = input_from_the_input_bar[0];  
        newstr.append( str: str2);  
        newstr.append( s: ".MM");  
        return newstr;  
    }  
    else if (input_from_the_input_bar.length() == 2){  
        str2 = input_from_the_input_bar[0];  
        newstr.append( str: str2);  
        str2 = input_from_the_input_bar[1];  
        newstr.append( str: str2);  
        newstr.append( s: ".MM");  
        return newstr;  
    }  
}
```

```
else if (input_from_the_input_bar.length() == 3){  
    str2 = input_from_the_input_bar[0];  
    newstr.append( str: str2);  
    str2 = input_from_the_input_bar[1];  
    newstr.append( str: str2);  
    newstr.append( s: ".");  
    str2 = input_from_the_input_bar[2];  
    newstr.append( str: str2);  
    newstr.append( s: "M");  
    return newstr;  
}  
else if (input_from_the_input_bar.length() == 4){  
    str2 = input_from_the_input_bar[0];  
    newstr.append( str: str2);  
    str2 = input_from_the_input_bar[1];  
    newstr.append( str: str2);  
    newstr.append( s: ".");  
    str2 = input_from_the_input_bar[2];  
    newstr.append( str: str2);  
    str2 = input_from_the_input_bar[3];  
    newstr.append( str: str2);  
    return newstr;  
}  
}
```

## Po: Zajmuje ponad połowę mniej miejsca.

```
std::string get_nice_looking_HHMM_format(std::string input_from_the_input_bar){

    if(input_from_the_input_bar == ""){return "HH.MM";}
    std::string newstr = "", str2;
    int j = 0;

    for(int i = 0; i<input_from_the_input_bar.length();i++){
        if(i==2){
            newstr.append( s: ".");
        }
        str2 = input_from_the_input_bar[i];
        newstr.append( str: str2);
        j++;
    }
    if(j==1){ newstr.append( s: "H.MM");}
    else if(j==2){ newstr.append( s: ".MM");}
    else if(j==3){ newstr.append( s: "M");}
    return newstr;
}
```

## Przed

```
void Okno::clear_site_and_wait(){ /// Makes breaks between jumping through sites and clears all drawable variables
    std::this_thread::sleep_for( rime: std::chrono::milliseconds( rep: 20));

    b1.button_set( x: 0, y: 0, height: 0, width: 0, font: &font1, text: "");
    b2.button_set( x: 0, y: 0, height: 0, width: 0, font: &font1, text: "");
    b3.button_set( x: 0, y: 0, height: 0, width: 0, font: &font1, text: "");
    b4.button_set( x: 0, y: 0, height: 0, width: 0, font: &font1, text: "");
    b5.button_set( x: 0, y: 0, height: 0, width: 0, font: &font1, text: "");
    b6.button_set( x: 0, y: 0, height: 0, width: 0, font: &font1, text: "");
    b7.button_set( x: 0, y: 0, height: 0, width: 0, font: &font1, text: "");
    b8.button_set( x: 0, y: 0, height: 0, width: 0, font: &font1, text: "");
    b9.button_set( x: 0, y: 0, height: 0, width: 0, font: &font1, text: "");
    text1.Textline_set( x: 0, y: 0, text: "", Charakter_size: 0, font: &font1);
    text2.Textline_set( x: 0, y: 0, text: "", Charakter_size: 0, font: &font1);
    text3.Textline_set( x: 0, y: 0, text: "", Charakter_size: 0, font: &font1);
    text4.Textline_set( x: 0, y: 0, text: "", Charakter_size: 0, font: &font1);
    text5.Textline_set( x: 0, y: 0, text: "", Charakter_size: 0, font: &font1);
    input_bar1.set_Input_bar( posX: 0, posY: 0, size: 0, color: sf::Color::White, selected: false, font: &font1);
    input_bar2.set_Input_bar( posX: 0, posY: 0, size: 0, color: sf::Color::White, selected: false, font: &font1);
    input_bar3.set_Input_bar( posX: 0, posY: 0, size: 0, color: sf::Color::White, selected: false, font: &font1);
    input_bar4.set_Input_bar( posX: 0, posY: 0, size: 0, color: sf::Color::White, selected: false, font: &font1);
    input_bar5.set_Input_bar( posX: 0, posY: 0, size: 0, color: sf::Color::White, selected: false, font: &font1);
    input_bar6.set_Input_bar( posX: 0, posY: 0, size: 0, color: sf::Color::White, selected: false, font: &font1);
    input_bar1.clear();
    input_bar2.clear();
    input_bar3.clear();
    input_bar4.clear();
    input_bar5.clear();
    input_bar6.clear();
```

## Po

```
void Okno::clear_site_and_wait(){ /// Makes breaks between jumping through
    std::this_thread::sleep_for( rime: std::chrono::milliseconds( rep: 20));

    clear_buttons();
    clear_textlines();
    clear_input_bars();
```

## Przed:

```
// Create a new instance of MailMessage class
System::SharedPtr<MailMessage> message = System::MakeObject<MailMessage>();
System::SharedPtr<SmtpClient> client = System::MakeObject<SmtpClient>();

// Set subject of the message, Html body and sender information
message->set_Subject(u"New message created by Aspose.Email for .NET");
message->set_From(System::MakeObject<MailAddress>(L"from@domain.com", L"Sender Name", false));
message->set_Body(System::String(u"<b>This line is in bold.</b> <br/> <br/>") + u"<font color=blue>This line is in blue color</font>");
message->set_BodyEncoding(System::Text::Encoding::get_ASCII());
// Add TO recipients and Add CC recipients
message->get_To()->Add(System::MakeObject<MailAddress>(L"to1@domain.com", L"Recipient 1", false));
message->get_To()->Add(System::MakeObject<MailAddress>(L"to2@domain.com", L"Recipient 2", false));
message->get_CC()->Add(System::MakeObject<MailAddress>(L"cc1@domain.com", L"Recipient 3", false));

// Specify your mailing host server, Username, Password, Port # and Security option
client->set_Host(u"mail.server.com");
client->set_Username(u"username");
client->set_Password(u"password");
client->set_Port(587);
client->set_SecurityOptions(Aspose::Email::Clients::SecurityOptions::SSLExplicit);

try
{
// Send this message
client->Send(message);
}
```

## Po:

```
import gmail
# Twój adres e-mail i hasło
username = 'gabinetdentystycznyiodent@gmail.com'
password = 'ppbrnlloepwfgjau'
# ironnext22 *

def mail1(mail = "X", data=""):
    client = gmail.GMail('Klinika Dentystyczna<' + username + '>', password)
    msg = gmail.Message('Wizyta', to='pacjent<' + mail + '>', text='Dzień dobry,\nPrzypominamy o wizycie w dniu: ' + data +
        '\nProszę o informacje zwrotne z potwierdzeniem wizyty\n\nPrzrawiamy.\nZespół IO-Dent')
    client.send(msg)

# Adres e-mail odbiorcy
obd = 'wiktor.k.2002icloud.com'
obd = open("mail2.txt", "r")
data = open("data.txt", "r")
mail1(obd.read(), data.read())
```

## Przed:

```
#include <iostream>
#include "MENU.h"
using namespace sf;
int main()
{
    sf::RenderWindow window_menu( sf::VideoMode( 800, 600, 32 ), "NIE DLACZEGO" );
    MENU menu(window_menu.getSize().x, window_menu.getSize().y);
    window_menu.setFramerateLimit(60);

    View view(FloatRect (0.0f, 0.0f, 800, 600)); //view_size.x view_size.y
    //
    while( window_menu.isOpen() ) {
        Event ev{};

        while (window_menu.pollEvent(ev)) {
            switch (ev.type) {
                case Event::KeyReleased:
                    switch (ev.key.code) {
                        case Keyboard::Up:
                            menu.MoveUp();
                            break;

                        case Keyboard::W:
                            menu.MoveUp();
                            break;

                        case Keyboard::Down:
                            menu.MoveDown();
                            break;

                        case Keyboard::S:
                            menu.MoveDown();
                            break;
                        case Keyboard::Return:
                            window_menu.close();
                            break;
                    }
                    break;
                case Event::Closed:
                    window_menu.close();
                    return 0;
            }
        }
    }
}
```

## Po:

```
int main()
{
    Okno menu;
    while(menu.get_window_is_open()) {

        menu.update();
        menu.render();

    }
    return 0;
}
```

## 10. Wzorzec projektowy

*„Wzorzec opisuje problem, który powtarza się wielokrotnie w danym środowisku, oraz podaje istotę jego rozwiązania w taki sposób, aby można było je zastosować miliony razy bez potrzeby powtarzania tej samej pracy”*

Christopher Alexander „A pattern language”, 1977

Podczas pracy nad naszym projektem użyliśmy wzorca projektowego Fasada. Nasz projekt był wystarczająco rozbudowany, żeby użycie wzorca projektowego przyspieszyło napisanie kodu. Wzorzec fasada polega na stworzeniu oddzielnej klasy wywołującej poszczególne funkcje biblioteki lub frameworka. Użyliśmy biblioteki graficznej SFML, która umożliwia tworzenie wielu obiektów, a za pomocą klasy fasada moglibyśmy sprawniej nimi zarządzać. Zamiast wiązać bezpośrednio kod z wieloma klasami składowymi SFML, tworzymy klasę fasady, która by wszystko społała.

Przykład opisujący idealnie wykorzystania fasady na podstawie kodu ze strony [refactoring.guru](http://refactoring.guru):

```
// Oto klasy złożonego frameworku od zewnętrznego dostawcy
// służącego konwersji wideo. Nie mamy wpływu na ten kod, więc
// nie możemy go uproszcić.

class VideoFile
// ...

class OggCompressionCodec
// ...

class MPEG4CompressionCodec
// ...

class CodecFactory
// ...

class BitrateReader
// ...

class AudioMixer
// ...
```

```

// Tworzymy klasę fasada która ukryje złożoność frameworku za
// prostym interfejsem. Takie podejście jest kompromisem
// pomiędzy funkcjonalnością, a łatwością użycia.
class VideoConverter is
    method convert(filename, format):File is
        file = new VideoFile(filename)
        sourceCodec = (new CodecFactory).extract(file)
        if (format == "mp4")
            destinationCodec = new MPEG4CompressionCodec()
        else
            destinationCodec = new OggCompressionCodec()
        buffer = BitrateReader.read(filename, sourceCodec)
        result = BitrateReader.convert(buffer, destinationCodec)
        result = (new AudioMixer()).fix(result)
        return new File(result)

// Klasy aplikacji nie są zależne od masy klas wchodzących w
// skład frameworku. Ponadto, w przypadku konieczności wymiany
// frameworku na inny, trzeba będzie zmodyfikować jedynie klasę
// fasada.
class Application is
    method main() is
        convertor = new VideoConverter()
        mp4 = convertor.convert("funny-cats-video.ogg", "mp4")
        mp4.save()

```

Fasada to klasa stanowiąca prosty interfejs dla złożonego podsystemu.

**Przykłady wykorzystania fasady podczas tworzenia naszej aplikacji.**

## Klasa Textline

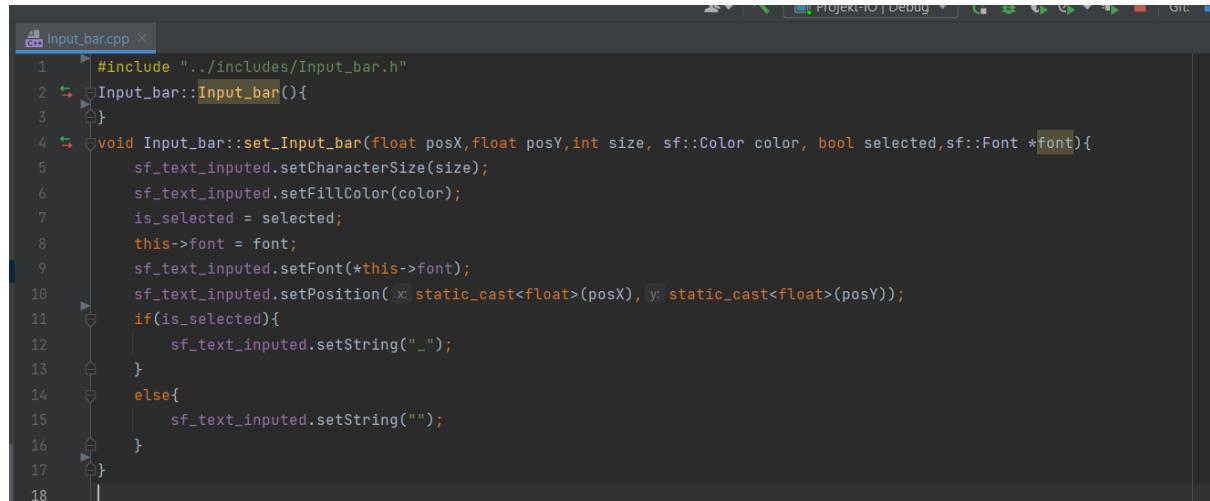
```

void Textline::Textline_set(float x, float y, std::string text, int Charakter_size, sf::Font *font) {
    this->text_to_be_displayed.setString(text);
    this->font = font;
    this->text_to_be_displayed.setFont(*this->font);
    this->text_to_be_displayed.setFillColor( color: sf::Color{ color: text_color });
    this->text_to_be_displayed.setCharacterSize( size: Charakter_size );
    this->text_to_be_displayed.setPosition(x,y);
    this->text_to_be_displayed.setScale( factors: sf::Vector2f( X: 0.5f , Y: 0.5f));
    this->text_to_be_displayed.setOutlineColor( color: sf::Color{ color: out_line_color });
    this->text_to_be_displayed.setOutlineThickness( thickness: 10 );
}

void Textline::render(sf::RenderTarget *target) {
    target->draw( drawable: this->text_to_be_displayed );
}

```

## Klasa Input\_bar



```

1 #include "../includes/Input_bar.h"
2 Input_bar::Input_bar(){
3 }
4 void Input_bar::set_Input_bar(float posX, float posY, int size, sf::Color color, bool selected, sf::Font *font){
5     sf_text_inputed.setCharacterSize(size);
6     sf_text_inputed.setFillColor(color);
7     is_selected = selected;
8     this->font = font;
9     sf_text_inputed.setFont(*this->font);
10    sf_text_inputed.setPosition(x: static_cast<float>(posX), y: static_cast<float>(posY));
11    if(is_selected){
12        sf_text_inputed.setString("_");
13    }
14    else{
15        sf_text_inputed.setString("");
16    }
17 }
18

```

## Klasa Button



```

1 void Button::button_set(float x, float y, float height, float width, sf::Font* font, std::string text){
2     button_photo.loadFromFile( filename: "../stuff/button.png");
3
4     this->button_state = BUTTON_IDLE;
5     this->shape.setPosition(sf::Vector2f( x: x, y: y));
6     this->shape.setSize(sf::Vector2f( x: width, y: height));
7     this->shape.setTexture( texture: &button_photo);
8     this->font = font;
9     this->text.setFont(*this->font);
10    this->text.setString(text);
11    this->text.setFillColor( color: sf::Color{ color: text_color });
12    this->text.setOutlineThickness( thickness: 4 );
13    this->text.setOutlineColor( color: sf::Color{ color: out_line_color });
14    this->text.setCharacterSize( size: 40 );
15
16    this->text.setPosition( // sets the position of TEXT into the button
17        x: this->shape.getPosition().x + this->shape.getSize().x/1.8- this->text.getCharacterSize()*text.length()/3.0,
18        y: this->shape.getPosition().y + (this->shape.getSize().y/1.5) - this->text.getCharacterSize()
19    );
20

```

## 11. Instrukcja użytkowania programu

### Podstawowe zasady korzystania z aplikacji:

- I. Z aplikacji mogą korzystać wyłącznie osoby, które posiadają konto i są zalogowane.
- II. Aby założyć konto, podaj swój adres email i utwórz hasło.
- III. System sprawdzi Twoje dane, a następnie założy Ci konto i uzyskasz możliwość logowania.

### Poniżej znajduje się instrukcja użytkowania:

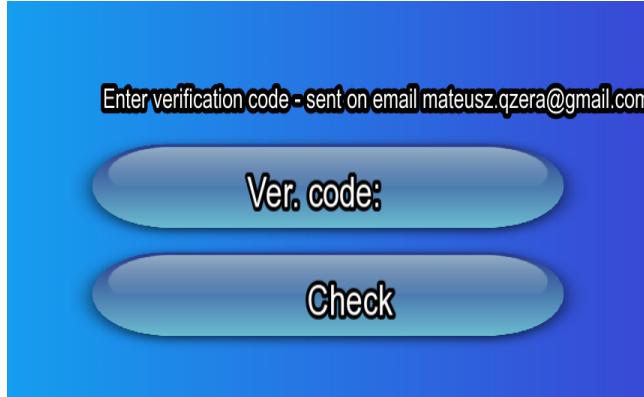
	<h4>Logowanie do systemu</h4> <ol style="list-style-type: none"><li>1. naciśnij przycisk Log in aby przejść na stronę logowania</li></ol>
	<ol style="list-style-type: none"><li>2. wprowadź poprawne dane do wcześniej założonego konta w pola Login oraz Password</li><li>3. wciśnij Log in aby się zalogować</li></ol>



### Zakładanie konta

1. Przejdź stronę logowania
2. Przejdź na zakładkę Register
3. Wprowadź dane
4. Wciśnij przycisk Register

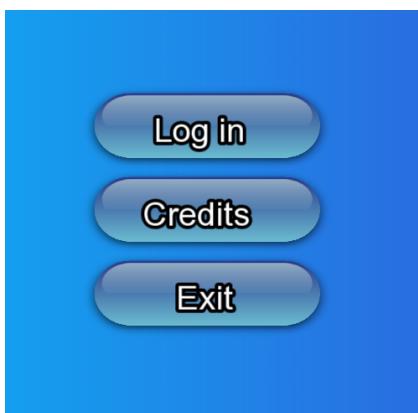
 Klinika Dentystyczna <gabinetdentystycznyiodent@gmail.com>  
do mnie ▾  
Twój kod weryfikacyjny: 1452



5. Wprowadź kod weryfikacyjny wysłany na podany email
6. Naciśnij przycisk Check



7. Po wprowadzeniu poprawnego kodu wyskoczy komunikat z poprawnie założonym kontem

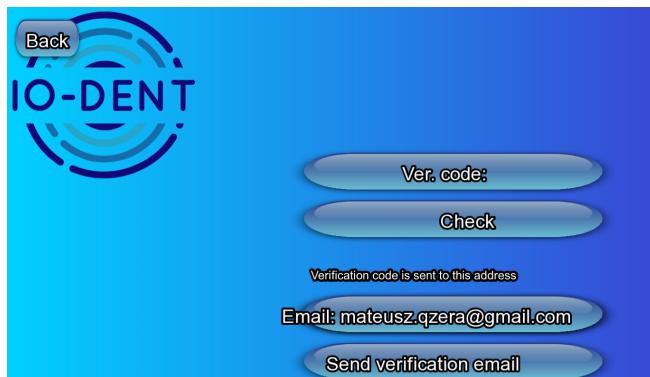


## Odzyskiwanie hasła

1. Przejdź na stronę logowania



2. Przejdź na stronę "Forgot Password"



3. Wpisz e-mail do konta

Klinika Dentystyczna <gabinetdentystycznyident@gmail.com>  
do mnie ▾  
Twój kod werifikacyjny: 1452

4. Na ten adres e-mail został wysłany kod aktywacyjny



5. wpisz kod w odpowiednie pole
6. zmień hasło



### Zarządzanie własnym kontem

1. Zaloguj się na swoje konto



2. Przejdź na stronę "My account"



3. wybierz odpowiednie pole
4. wprowadź nową wartość i naciśnij przycisk zmiany



## Zarządzanie magazynem



1. Zaloguj się na swoje konto



2. Przejdź na stronę "Magazine"

ID	Nazwa	Ilosc
10	Płomby	15
5	Krzesło dentystyczne	1
1	Wązki	100
2	Wierko	5
3	Kubeczki	100

3. Wciśnij przycisk "Add"

4. Dodaj nazwę, ID i ilość przedmiotów

5. Wciśnij przycisk "Edit"

6. Wpisz nazwę, ID i zmodyfikuj ilość przedmiotów



	Monday 21/12/2023	Tuesday 31/12/2023	Wednesday 4/1/2023	Thursday 5/1/2023	Friday 6/1/2023	Saturday 7/1/2023	Sunday 8/1/2023
8:00-9:00							
9:00-10:00							
10:00-11:00							
11:00-12:00							
12:00-13:00							
13:00-14:00							
14:00-15:00							
15:00-16:00							

	Monday 16/12/2023	Tuesday 17/12/2023	Wednesday 18/12/2023	Thursday 19/12/2023	Friday 20/12/2023	Saturday 21/12/2023	Sunday 22/12/2023
8:00-9:00							
9:00-10:00			Mateusz Lugowski mateusz.lugowski.10@gmail.com 078/0				
10:00-11:00			Aneta Kowalska aneta.kowalska.44@gmail.com 078/0				
11:00-12:00			Mateusz Kowalski mateusz.kowalski.6@gmail.com 078/0				
12:00-13:00				Michał Piotr michalpiotr1991@gmail.com 078/0			
13:00-14:00							
14:00-15:00					Wojciech Kowalski wojciech.kowalski.2002@gmail.com 078/0		
15:00-16:00						Mateusz Lugowski lugowski.mateusz.10@gmail.com 078/0	

## Kalendarz - jak postugiwać się

### 1. Zaloguj się na swoje konto

### 2. Przejdz na stronę “Calendar”

### 3. Poruszaj się za pomocą przycisków “Next” i “Prev”



## Zarządzanie kalendarzem

	Monday 16.1.2023	Tuesday 17.1.2023	Wednesday 18.1.2023	Thursday 19.1.2023	Friday 20.1.2023	Saturday 21.1.2023	Sunday 22.1.2023
8.00-9.00							
9.00-10.00			Mateusz Lugowski lugowski.mateusz.02@gmail.com 08:45			Witold Kawecki witor.kawiecki.2002@gmail.com 09:15	
10.00-11.00			Agnieszka Górska agnieszka.gorska.02@gmail.com 09:30				
11.00-12.00			Mateusz Kuzera mateusz.qzera@gmail.com 10:10				
12.00-13.00				Marcin Pacholski marcin.pacholski@gmail.com 12:00			
13.00-14.00							
14.00-15.00					Witold Kawecki witor.kawiecki.2002@gmail.com witor.kawiecki.2002@gmail.com 09:30		
15.00-16.00						Mateusz Lugowski lugowski.mateusz.02@gmail.com 09:30	

Back

Name:	
Surname:	
Email:	
Pesel:	
Date: DD.MM.YYYY	
Hour: HH.MM	
Paid: (use ,) ZL	
Add Visit	

Back

Edit Visit	
Date: DD.MM.YYYY	
Hour: HH.MM	
Change	

1. Zaloguj się na swoje konto
2. Przejdź na stronę “Calendar”

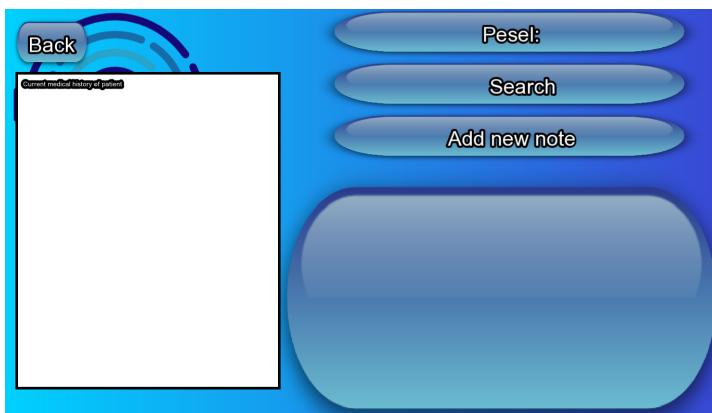
3. Wciśnij przycisk “Add”
4. Do edycji danych wybierz ID wizyty i kliknij przycisk “Edit”

5. ad3. Dodaj dane pacjenta

6. ad4. Ustaw nowe dane

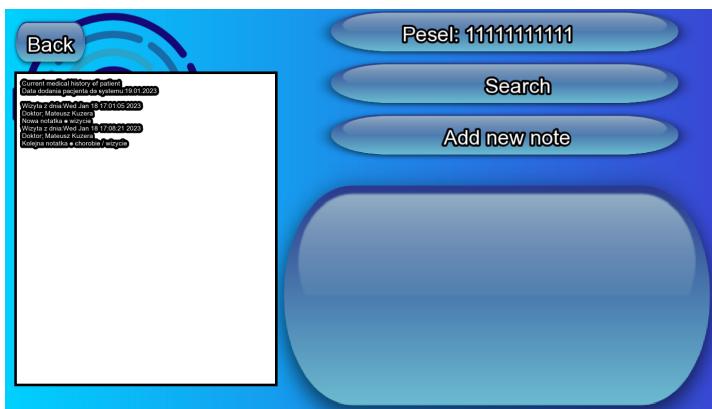


## Zarządzanie historią pacjenta

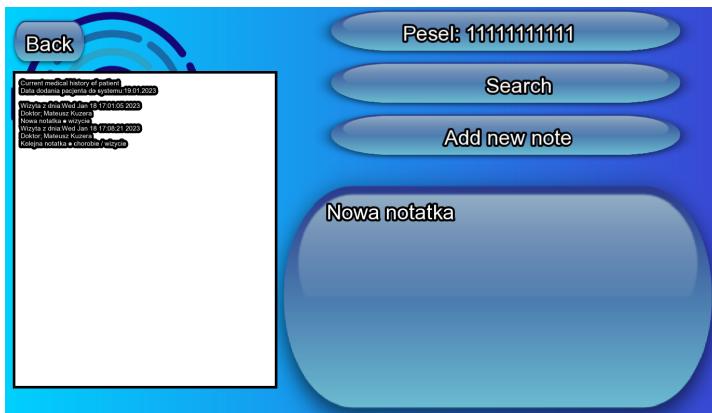


1. Zaloguj się na swoje konto
2. Przejdź na stronę "Patients"

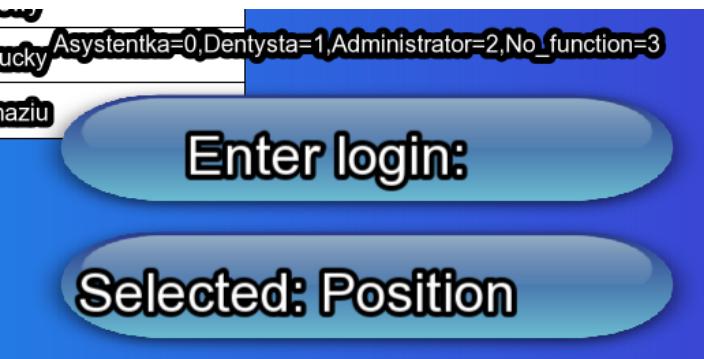
3. Wpisz pesel pacjenta



4. Kliknij przycisk "Search"



5. Dodaj nową notatkę



## Zarządzanie pracownikami przez administratora

1. Zaloguj się na konto z uprawnieniami Administratora.
2. Naciśnij przycisk *Employees* aby przejść do strony zarządzania pracownikami
3. Wpisz login wybranego pracownika do edycji.
4. Wybierz na pasku *Selected* atrybut do zmiany
5. Wprowadź nową wartość tego atrybutu na pasku *New:*
6. Naciśnij przycisk *Change!* aby zmienić atrybut
7. Uwaga! W celu zmienienia funkcji pracownika należy wybrać na pasku *Selected Function* a następnie w pasku *New:* wprowadzić odpowiedni numer



Logged as: Doktor  
Mateusz Kuzera  
Position: Dentysta  
Email: mateusz.qzera@gmail.com  
Today is: Wed Jan 18 17:22:48 2023  
Have a good day

Send emails  
Patients  
Magazine  
Calendar  
My account  
Finance  
Log out

### **Wysyłanie e maili przypominających o wizycie**

1. Zaloguj się na konto z uprawnieniami
2. W celu wysłania e-maila z przypomnieniem należy wcisnąć przycisk *Send emails*.
3. Po chwili pacjenci którzy mają wizytę następnego dnia otrzymają maila.

Logged as: Doktor  
Mateusz Kuzera  
Position: Dentysta  
Email: mateusz.qzera@gmail.com  
Today is: Wed Jan 18 17:23:25 2023  
Have a good day

Send emails  
Patients  
Magazine  
Calendar  
My account  
Finance  
Log out

### **Jak się wylogować**

1. Zaloguj się na konto
2. Wciśnij przycisk *Log out*
3. Zostaniesz wylogowany i przekierowany do strony początkowej

## 12. Podsumowanie

### Opis prac

Pracę nad projektem zaczęliśmy w październiku, kiedy powstały pierwsze diagramy opisujące naszą aplikację. Sukcesywnie tworzyliśmy kolejne diagramy. Tworzenie każdego diagramu poprzedzone było poznaniem techniki jego wykonania oraz poznaniem podstawowych zasad jego stosowania. W kolejnych tygodniach zaplanowaliśmy jak będzie wyglądała implementacja naszego kodu i ustaliliśmy szczegóły funkcjonalności klas. W międzyczasie wybraliśmy odpowiednie środowisko programistyczne dla języka C++ i dla naszego projektu. Powstało też nasze repozytorium na GitHubie, w którym zamieściliśmy pierwsze linijki kodu. Przez następne tygodnie stopniowo dodawaliśmy kolejne elementy kodu i w razie potrzeby modyfikowaliśmy funkcjonalności klasy. Niecały miesiąc przed deadlinem zaczęliśmy tworzyć dokumentację. W tym czasie poprawiliśmy niektóre diagramy i dalej udoskonalałyśmy nasz kod. Prace skończyłyśmy z większością założonych zadań wypełnionych. Zostawiliśmy jednak możliwość dalszego rozwoju projektu, ponieważ najlepsze aplikacje można udoskonalać bezterminowo.

**Filmik prezentujący działanie aplikacji gabinetu dentystycznego IO-DENT.**

[LINK do filmiku](#)

### Napotkane problemy

W trakcie realizacji projektu napotkaliśmy szereg problemów związanych z wieloma aspektami, jednak od razu staraliśmy się je rozwiązać, aby prace szły w odpowiednim tempie. Jednym z najczęściej pojawiających się problemów była nasza niewystarczająca wiedza, aby zrealizować konkretny element w kodzie. Radziliśmy sobie z tym czytając fora internetowe takie jak Stackoverflow, oglądając poradniki na YouTube czy czytając dokumentację. Kolejnym pojawiającym się problemem była komunikacja, niezrozumienie fragmentu kodu czy pomysłu osoby z zespołu. Jednak bardzo szybko rozwiązaliśmy ten problem umawiając się na realizację kodu w konkretnych zasadach i spotkania na komunikatorze Discord. Ostatecznie mimo napotkanych problemów, udało nam się doprowadzić projekt do końca.