Spis treści

[WSTĘP 1](#_Toc159745287)

[1. ANALIZA WYMAGAŃ 2](#_Toc159745288)

[1.1. Wymagania biznesowe 2](#_Toc159745289)

[1.2. Diagram modeli domenowych 4](#_Toc159745290)

[1.3. Diagram przypadków użycia 6](#_Toc159745291)

# WSTĘP

# ANALIZA WYMAGAŃ

## Wymagania biznesowe

Budowę każdego systemu komputerowego, należy rozpocząć od ustalenia problemu, który ma on rozwiązać. Następnie należy dostrzec możliwe rozwiązania i wybrać jedno z nich. Powszechnie przyjętym sposobem na opis dlaczego organizacja implementuje system są tzw. wymagania biznesowe skupiające się na celach biznesowych[[1]](#footnote-1). Poniżej przedstawiona jest analiza wymagań biznesowych klienta.

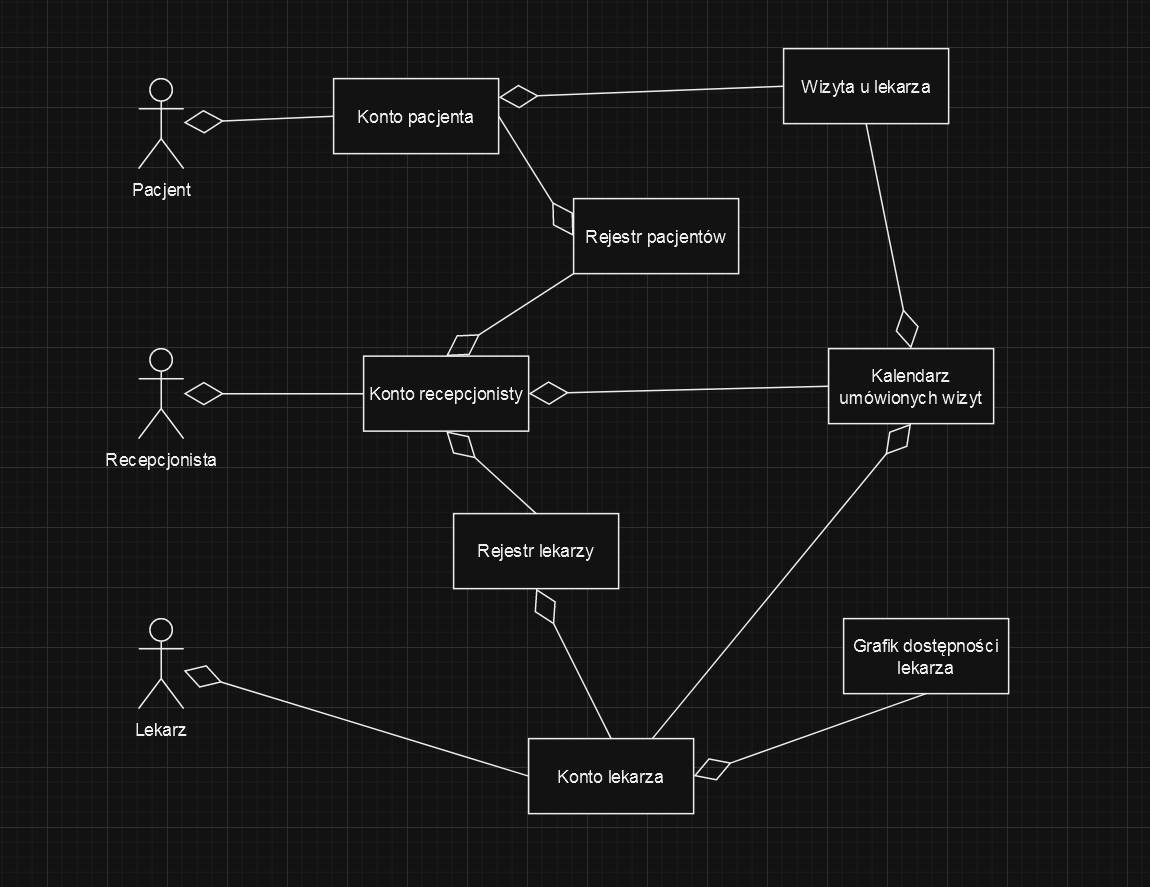
Przychodnia o nazwie *Great Health* potrzebuje systemu, który odciąży jej pracowników w rejestracji pacjentów, tym samym zwiększając wydajność, zmniejszając zapotrzebowanie na rekrutację nowej kadry oraz poprawiając konkurencyjność. Obecnie każdy pacjent musi się najpierw zarejestrować w przychodni poprzez stawienie się fizycznie w przychodni oraz przedstawienie swojego dowodu osobistego. Wówczas recepcjonista prowadząc rejestr pacjentów w formie papierowej, wpisuje nowo zarejestrowaną osobę na liście. Ponadto, pracownicy muszą prowadzić rejestr lekarzy. Podobnie jak z pacjentami, każdy lekarz musi być wpisany w papierowym rejestrze, a w przypadku zwolnienia, z niego wypisany. Oprócz wyżej wspomnianych rejestrów, prowadzony jest grafik dostępności lekarzy, ponieważ większość z nich pełni swoje obowiązki również w innych placówkach. Kiedy recepcjonista placówki ma już przed sobą rejestr pacjentów, lekarzy oraz grafik dostępności specjalistów, może on przyjmować osobiście stawionych pacjentów proszących o umówienie na wizytę z konkretnym lekarzem, o konkretnej godzinie. Informacje te są przechowywane w formie kalendarza. W przypadku jakichkolwiek zmian terminów, pracownik na recepcji musi wykreślić daną osobę z kalendarza i znaleźć jej nowy termin. Cały ten proces, jest żmudny i łatwo o popełnienie w nim błędu, co może skutkować pewnym zamieszaniem oraz opóźnieniami. Zaś z perspektywy samych lekarzy, również nie jest to najlepszy system zarządzania, ponieważ są oni informowani o każdej zmianie z pewnym opóźnieniem. W idealnej sytuacji, lekarze wiedzą kiedy mają pacjenta, a kiedy nie, najszybciej jak tylko się da, oraz nie muszą być o tym informowani przez pracowników z recepcji. Pracownicy na recepcji powinni tylko i wyłącznie weryfikować tożsamość oraz obecność pacjentów gotowych na wizytę, a następnie przekierowywać ich do konkretnego gabinetu. Zaś z perspektywy pacjentów, najlepiej byłoby nie wychodzić z domu, aby umówić się do lekarza oraz mieć jasny obraz wolnych terminów na każdy dzień.

## Diagram modeli domenowych

Na podstawie wyżej opisanych wymagań biznesowych przychodni, należy przełożyć je na wymagania użytkownika[[2]](#footnote-2). Najlepiej zrobić to w postaci diagramów przypadków użycia. Jednakże najpierw pomocnym jest zacząć od zdefiniowania modeli domenowych, które tworzyć będą system[[3]](#footnote-3). Tworząc taki diagram, warto trzymać się pewnych zasad. Są to[[4]](#footnote-4):

* Koncentracja na obiektach występujących w domenie biznesowej
* Używanie agregacji w celu odzwierciedlenia relacji między obiektami
* Zorganizowanie obiektów wokół kluczowych abstrakcji w domenie biznesowej
* Wykorzystywanie modeli domenowych w celu utworzenia słownika projektu
* Utworzenie diagramu modeli domenowych przed diagramem klas tak, aby w tym drugim uniknąć wieloznaczności używanych terminów

Na podstawie powyższych rekomendacji utworzony został diagram modeli domenowych, który znaleźć można na rysunku 1.1. Wyszczególnionych zostało trzech aktorów – pacjent, recepcjonista oraz lekarz. Dla każdego z nich, utworzony został obiekt będącym kontem w systemie. Konto pacjenta zawiera listę wizyt u lekarzy oraz jest częścią rejestru pacjentów. Natomiast konto recepcjonisty ma dostęp do rejestru pacjentów oraz rejestru lekarzy. Pracownik na recepcji będzie również zarządzać kalendarzem umówionych wizyt, dlatego też jego konto powiązane jest z kalendarzem umówionych wizyt. Ostatnim kontem, jest konto lekarza, do którego przypisany jest grafik dostępności, zaś samo konto jest częścią rejestru lekarzy oraz kalendarza umówionych wizyt.



Rys. 1.1 Diagram modeli domenowych

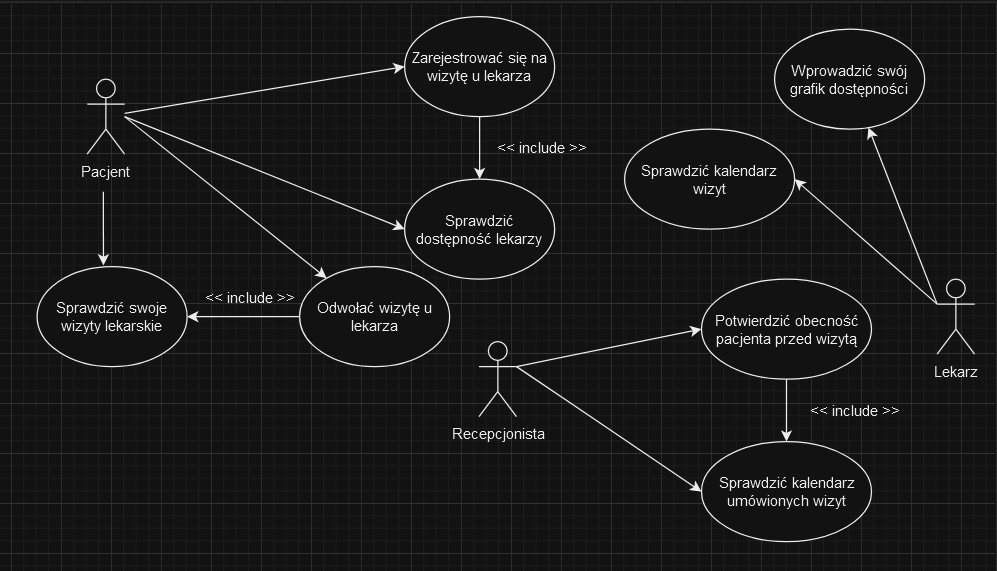
## Diagram przypadków użycia

Kiedy utworzony już został diagram modeli domenowych, można rozpocząć tworzenie diagramu przypadków użycia. Ten rodzaj diagramu pozwala uchwycić wymagania zachowania systemu tak, aby możliwym było go zaprojektowanie[[5]](#footnote-5). Ponadto, „k*ażdy przypadek użycia powinien zostać udokumentowany co najmniej trzema elementami:*

1. *opisem stanu początkowego (warunków początkowych),*
2. *scenariuszem,*
3. *opisem oczekiwanego stanu końcowego”[[6]](#footnote-6)*

Diagram przypadków użycia składa się z graficznego odwzorowania aktorów systemu oraz samych przypadków użycia, w których to aktorzy biorą udział[[7]](#footnote-7). Ponadto, diagram może zawierać również dwa dodatkowe typy relacji: *extend* oraz *include[[8]](#footnote-8)*. Pierwszy rozszerza przypadek użycia o scenariusz alternatywny. Drugi zaś oznacza, że jeden z przypadków użycia składa się z jednego lub kilku innych.

Na podstawie opisanych zasad, utworzonego diagramu modeli domenowych  
na rysunku 1.1, oraz opisu wymagań biznesowych, wykonany został diagram przypadków użycia – rysunek 1.2, który zawiera trzech aktorów. Pierwszym z nich jest pacjent. Ma on możliwość umówienia się na wizyty u lekarza. Częścią tego przypadku użycia jest również sprawdzenie dostępności lekarzy. Pacjent ma również możliwość odwołać umówioną wcześniej wizytę u lekarza. W celu odwołania wizyty, użytkownik musi najpierw znaleźć swoją wizytę w systemie poprzez sprawdzenie swoich wizyt lekarskich. Kolejnym aktorem jest lekarz. Ma on możliwość wprowadzić swój grafik dostępności oraz sprawdzić kalendarz wizyt z pacjentami. Ostatni aktor to recepcjonista. Jego zadaniem jest potwierdzić obecność pacjent tuż przed wizytą lekarską. Aby to zrobić, recepcjonista musi mieć możliwość sprawdzenia kalendarza umówionych wizyt.



Rys. 1.2 Diagram przypadków użycia

## Diagram klas

1. Karl E Wiegers, Joy Beatty, Specyfikacja oprogramowania. Inżynieria wymagań. Wydanie III, Helion, 2014, s. 35 [↑](#footnote-ref-1)
2. Karl E Wiegers, Joy Beatty, Specyfikacja oprogramowania. Inżynieria wymagań. Wydanie III, Helion, 2014, s. 35 [↑](#footnote-ref-2)
3. Doug Rosenberg, Matt Stephens, Use Case Driven Object Modeling with UML. Theory and Practice. Appress, 2007, s. 25 [↑](#footnote-ref-3)
4. Doug Rosenberg, Matt Stephens, Use Case Driven Object Modeling with UML. Theory and Practice. Appress, 2007, s. 26 [↑](#footnote-ref-4)
5. Doug Rosenberg, Matt Stephens, Use Case Driven Object Modeling with UML. Theory and Practice. Appress, 2007, s. 49 [↑](#footnote-ref-5)
6. Jarosław Żeliński, Analiza Biznesowa. Praktyczne modelowanie organizacji, Helion, 201, s. 74 [↑](#footnote-ref-6)
7. Karl E Wiegers, Joy Beatty, Specyfikacja oprogramowania. Inżynieria wymagań. Wydanie III, Helion, 2014, s. 172 [↑](#footnote-ref-7)
8. Karl E Wiegers, Joy Beatty, Specyfikacja oprogramowania. Inżynieria wymagań. Wydanie III, Helion, 2014, s. 179 [↑](#footnote-ref-8)