**Sprzedaż materacy**

**Dziedzina problemowa**

z tego systemu mogą korzystać firmy zajmujące się sprzedażą dowolnych towarów, ale ten

system jest przeznaczony dla firm sprzedających materace.

**Ceł**

ten system pomoże ułatwić pracę związaną ze sprzedażą, logistyką itp.

**Zakres odpowiedzialności systemu**

system ten jest odpowiedzialny za sprzedaż materaców, tworzenie zgłoszeń, wysyłanie ich do

magazynu, tworzenie zamówień, przygotowywanie dokumentów i wysyłanie towarów do

klientów.

**Użytkownicy systemu**

klient, właściciel, dział sprzedaży, pracownicy magazynu, dział logistyki, księgowość

**Wymagania niefunkcjonalne**

generalnie system musi pracować prawidłowo, ale mimo wszystko mogą być problemy związane

z podłączeniem do baz danych dla pracy z Bitrix 24.

**Opis przyszłej ewolucji systemu**

być może w przyszłości system automatycznie przesyła dokumenty między działami i

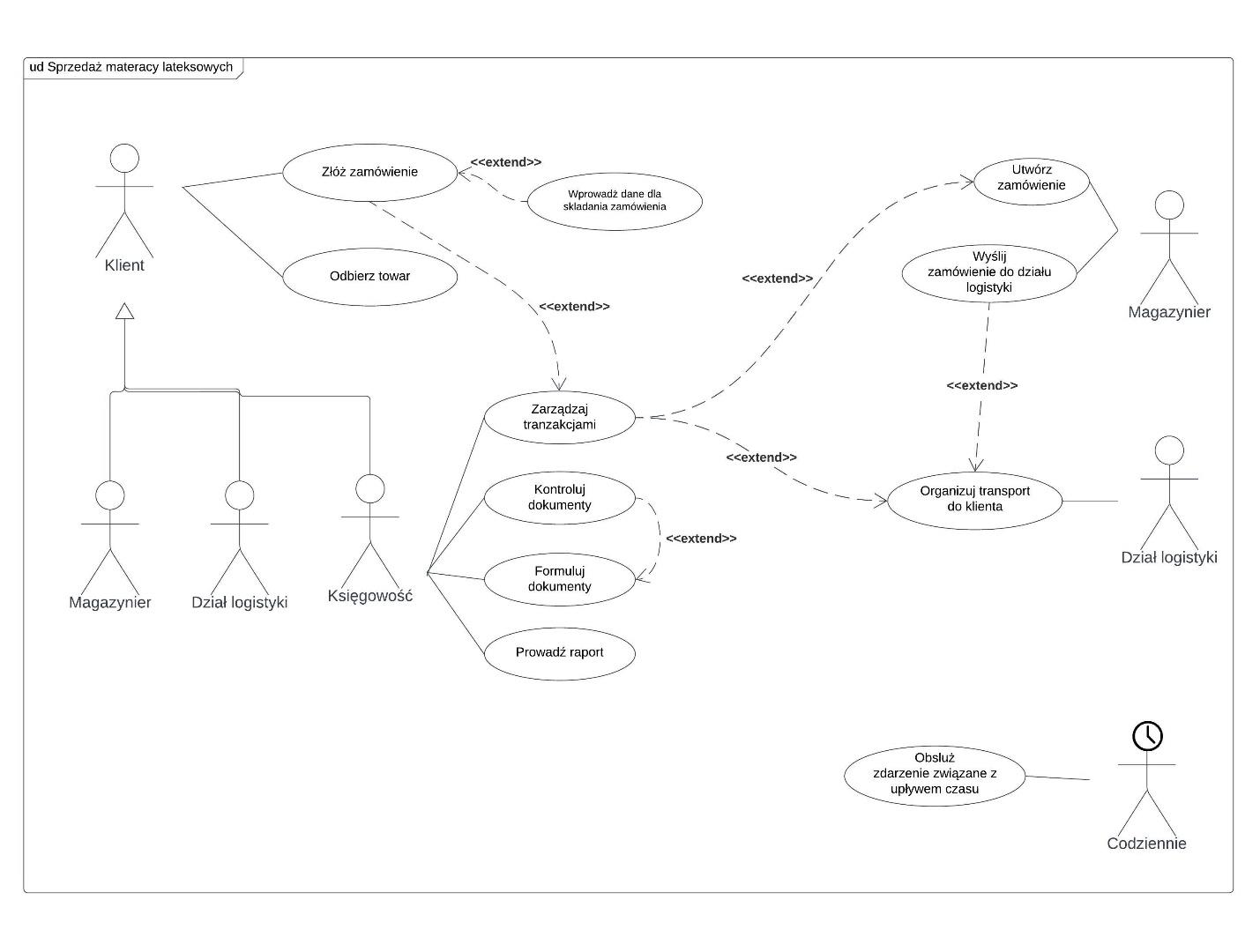
automatycznie weryfikuje płatność.

**Wymagania użytkownika:**

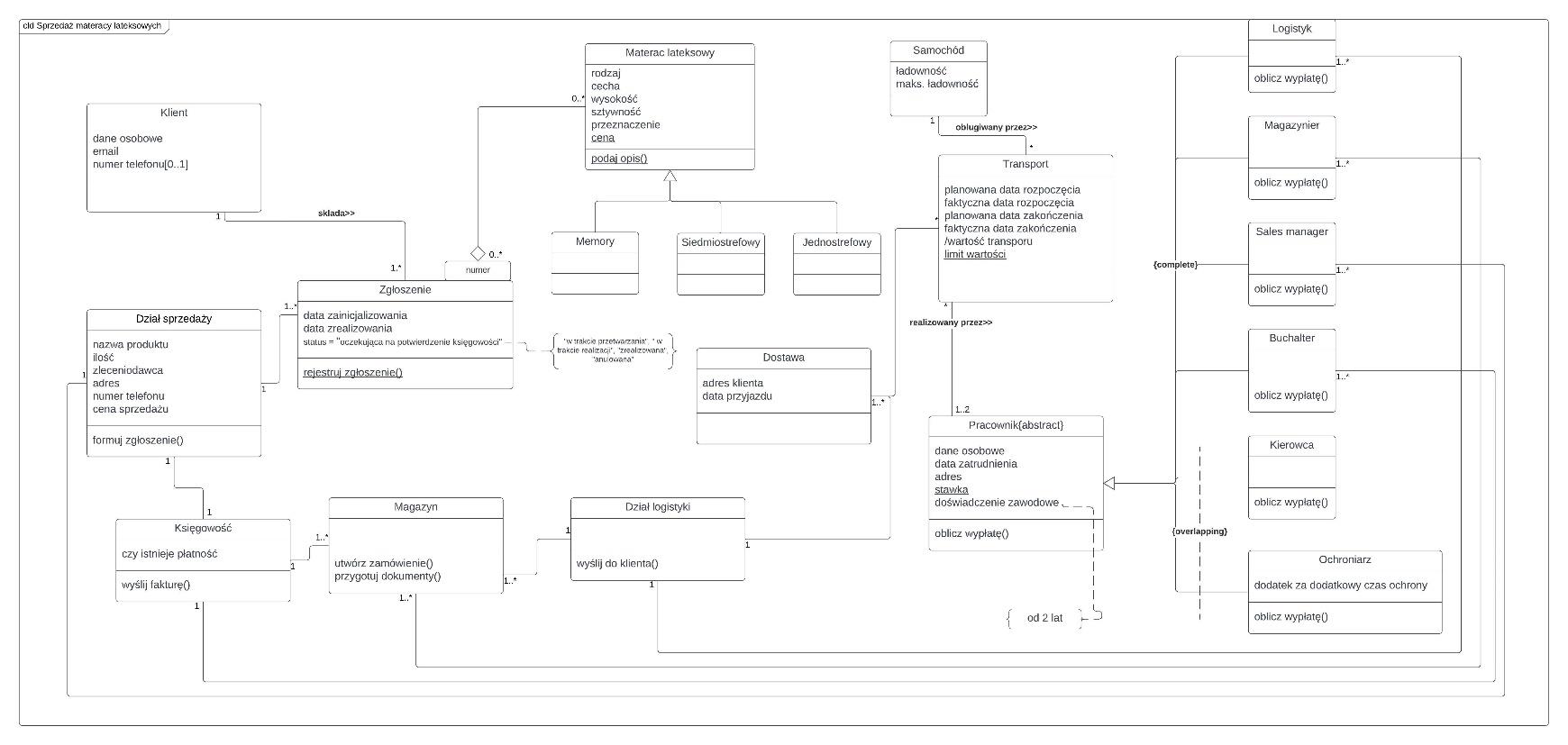
1. W systemie przechowujemy dane pracowników: dane osobowe, data zatrudnienia, adres, stawka dla wszystkich pracowników oraz doświadczenie zawodowe od 2 lat. Są 6 pracowników firmy: logistyk, magazynier, sales manager, buchalter, kierowca oraz ochroniarz. Co miesiąc jest obliczana wypłata, a dla ochroniarza dodatkowo jest uwzględniony dodatek za dodatkowy czas ochrony. Jest możliwość że pracownik może pełnić rolę kierowcy i ochroniarza. Każdy pracownik jest przydzielony do swojego działu.
2. Sklep posiada materacy różnego typu, opisywane przez: rodzaj, cecha, wysokość, sztywność, przeznaczenie, VAT dla wszystkich materacy, cena oraz brutto. Po wybraniu materaca jest wyświetlony cały opis. Materacy są podzielone na: Memory, Siedmiostrefowy oraz Jednostrefowy.
3. Po wejściu na stronę są przechowywane dane klienta: dane osobowe, email oraz opcjonalnie numer telefonu. Klient składa zamówienie jakie przechowuje: data zainicjalizowania, data zrealizowania. Zgłoszenie jest składane za pomocą numeru materaca jaki wybrał klient. Po składaniu rejestruje się zgłoszenie.
4. Po składaniu, zgłoszenie jest zapisane w historię gdzie przechowujemy datę formowania zgłoszenia i datę zakończenia oraz przechodzi do działu sprzedaży gdzie przechowujemy: nazwa produktu, ilość, zleceniodawca, adres, numer telefonu, cena sprzedaży. Dział sprzedaży formuje zgłoszenie
5. Księgowość sprawdza czy istnieje płatność. Jeżeli wszystko w porządku to wysyła fakturę do magazynu. Magazyn tworzy zamówienie i przygotowuje dokumenty. Dział logistyki otrzymuje wszystkie dokumenty i wysyła towar klientowi na jakim jest go adres i data przyjazdu.
6. Dla dostawy w systemie jest wybierany jeden samochód i jeden kierowca i jednym ochroniarzem. Dla każdego transportu przechowujemy takie dane: planowana data rozpoczęcia, faktyczna data rozpoczęcia, planowana data zakończenia, faktyczna data zakończenia, wartość transportu oraz go limit jaki jest dla wszystkich transportów. Gdy wartość transportu przekracza limit to przydzielamy ochroniarza.

System powinien wspomagać swoich użytkowników w realizacji zadań, takich jak:

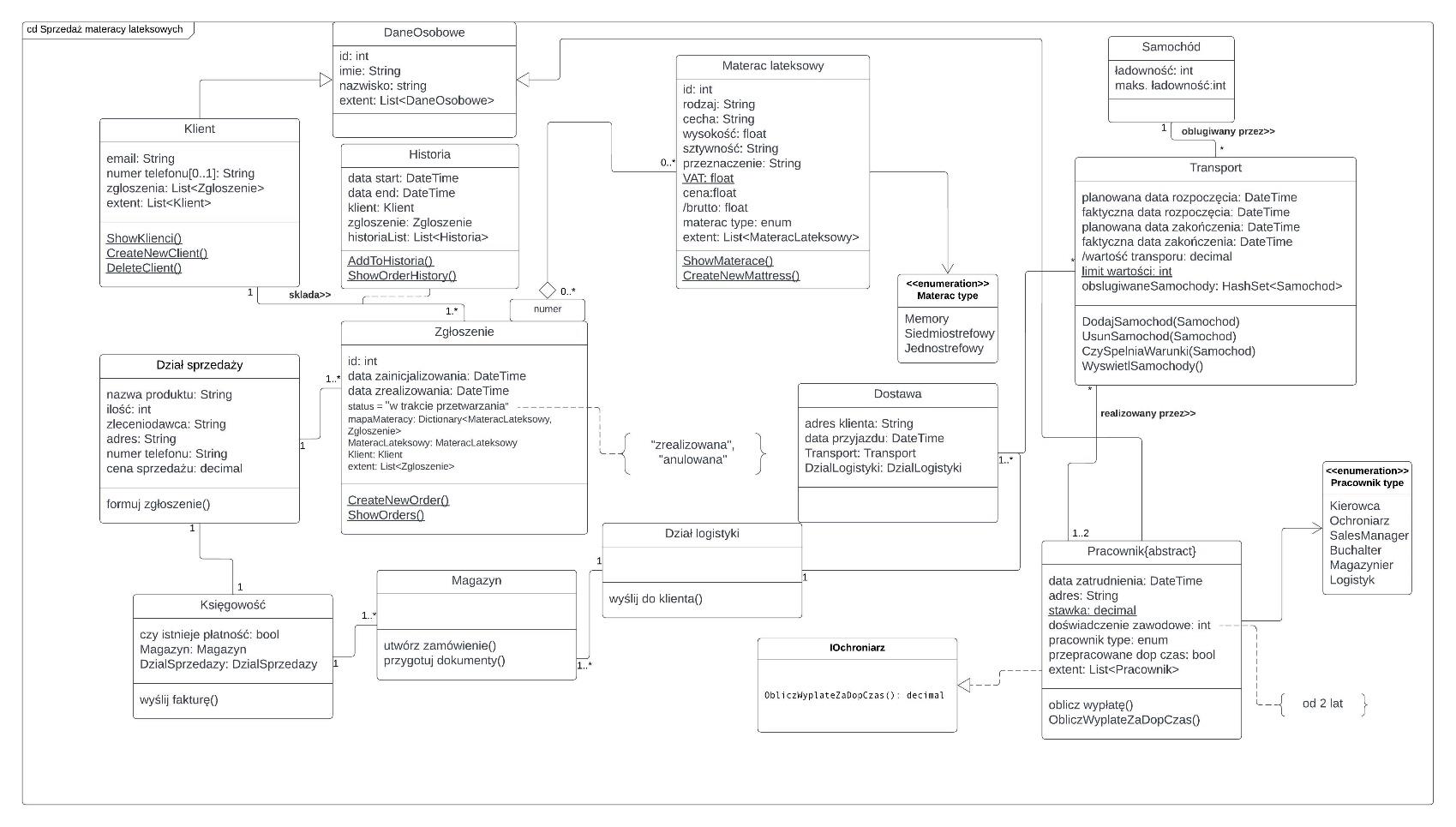
* Kontrolowanie, formułowanie dokumentów oraz prowadzenie raportu (księgowość);
* Zarządzanie transakcjami;
* Utworzenie zamówienia oraz wysyłanie go do działu logistyki (magazyn);
* Organizacja transportu do klienta (dział logistyki);
* Złożenie zamówienia, wprowadzenie kryteria dla wyszukiwania pasującego towaru oraz odebranie
* (klient);
* Obsługiwanie zdarzeń związane z upływem czasu (codziennie).

**Diagram przypadku użycia**

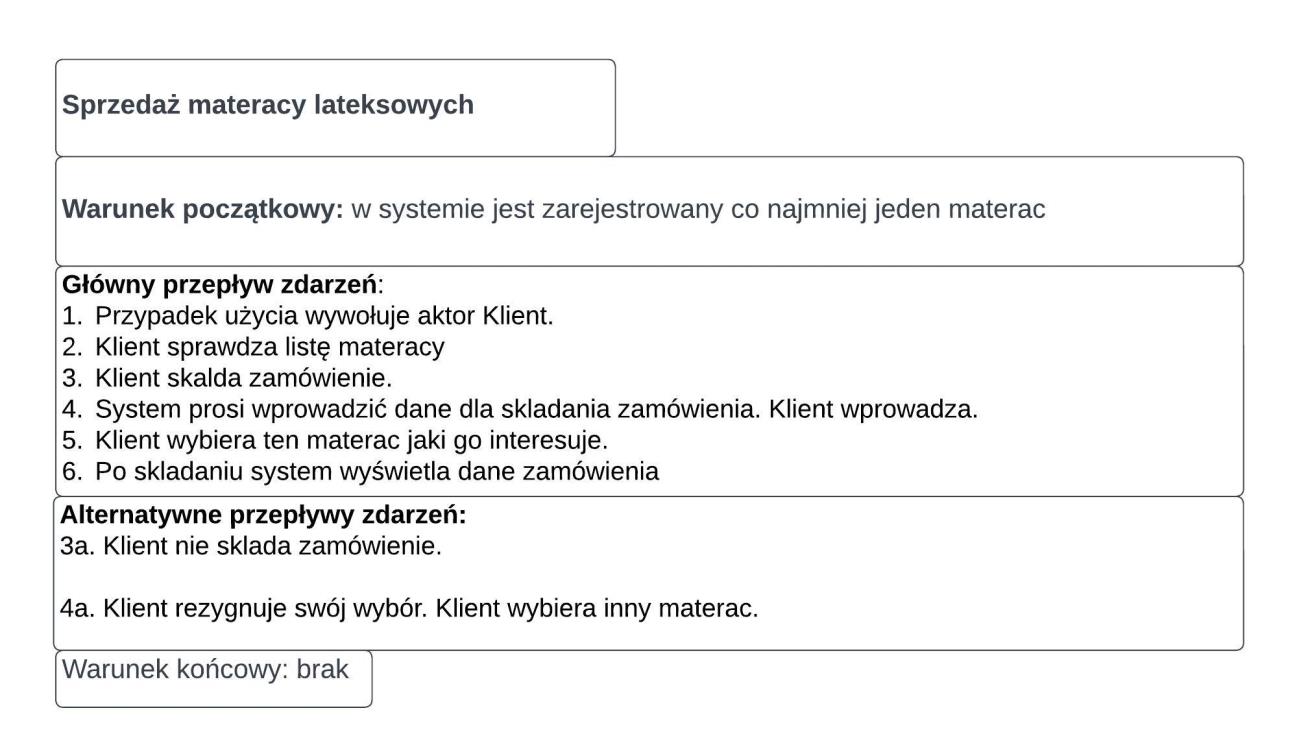
**Diagram klas – analityczny**



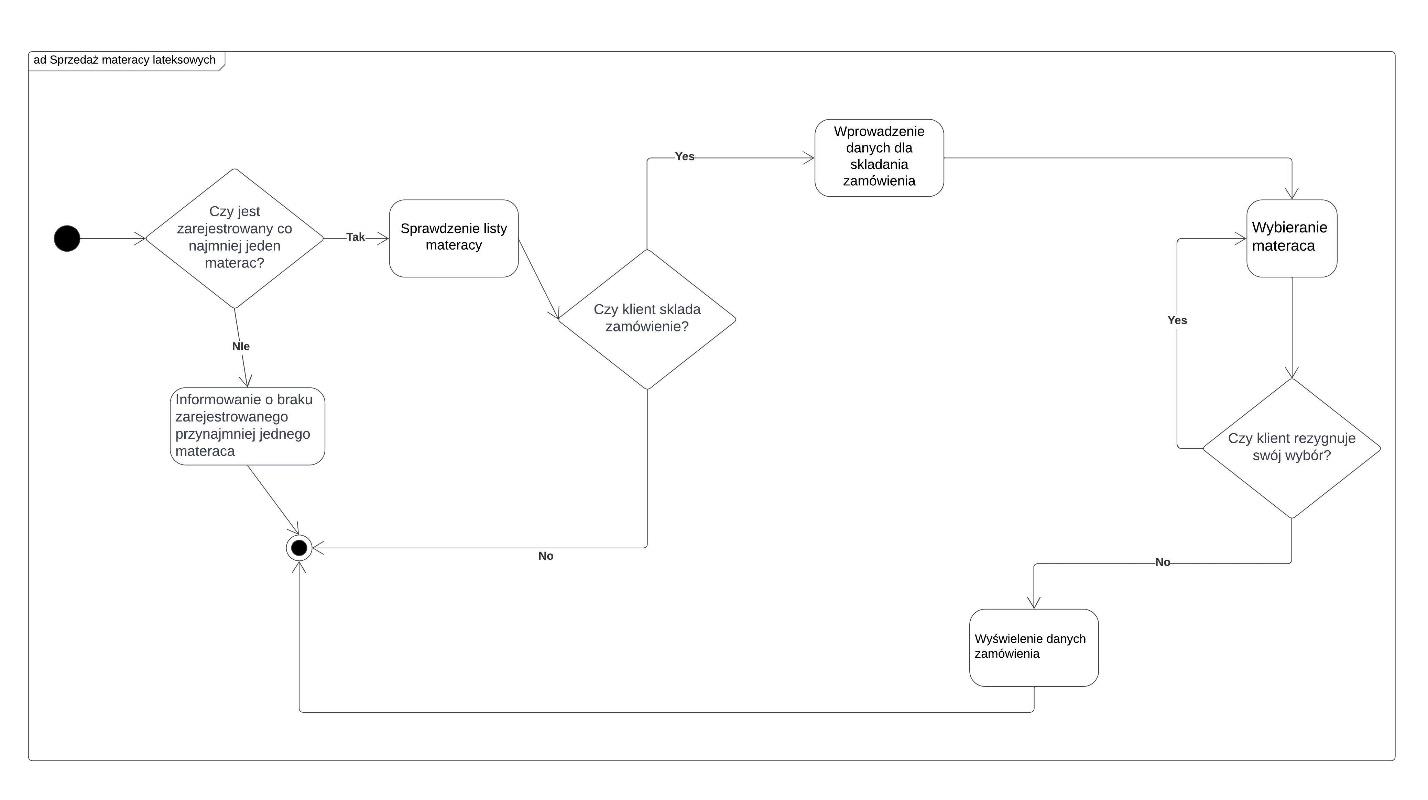
**Diagram klas – projektowy**



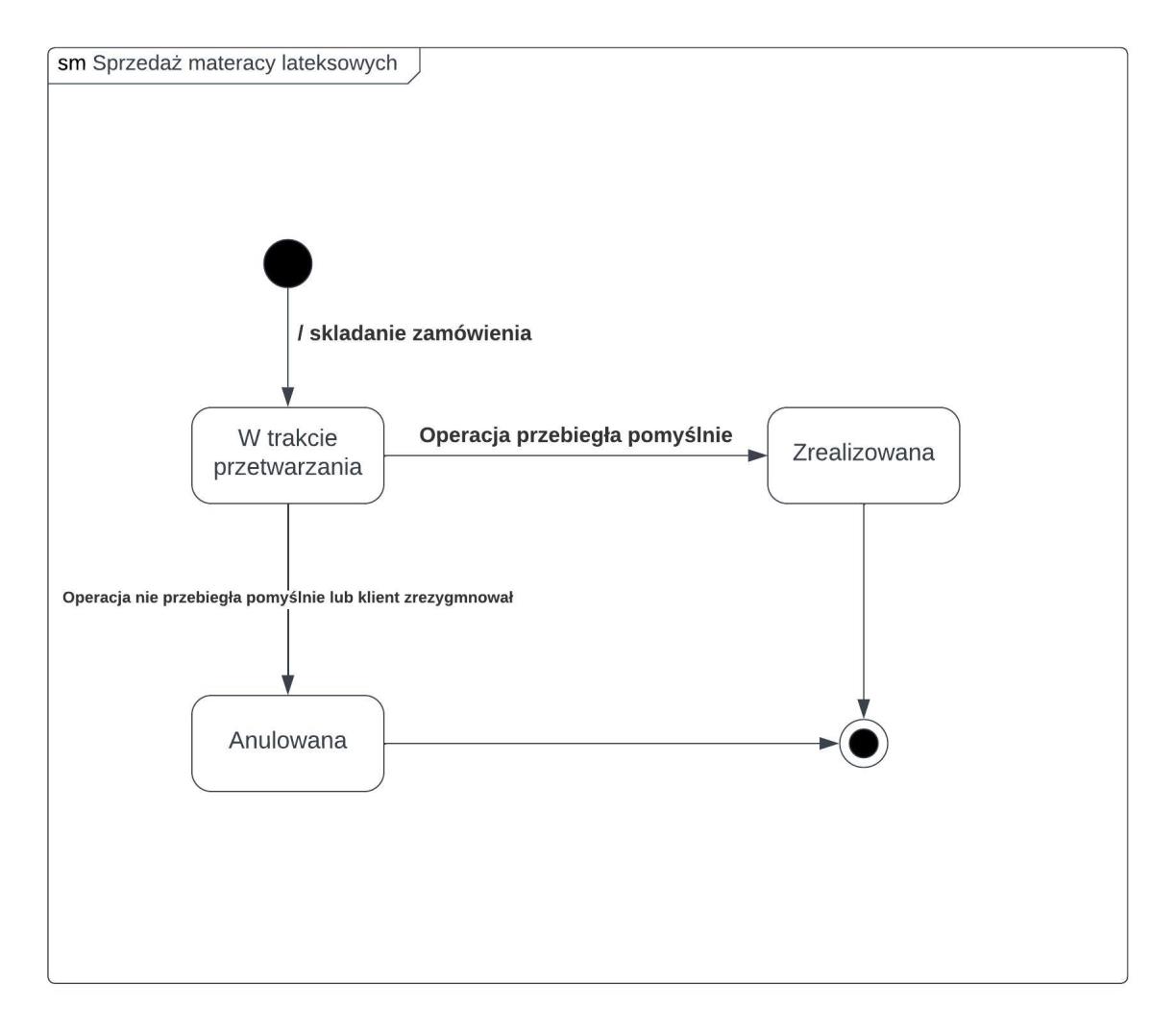
**Scenariusz przypadku użycia**



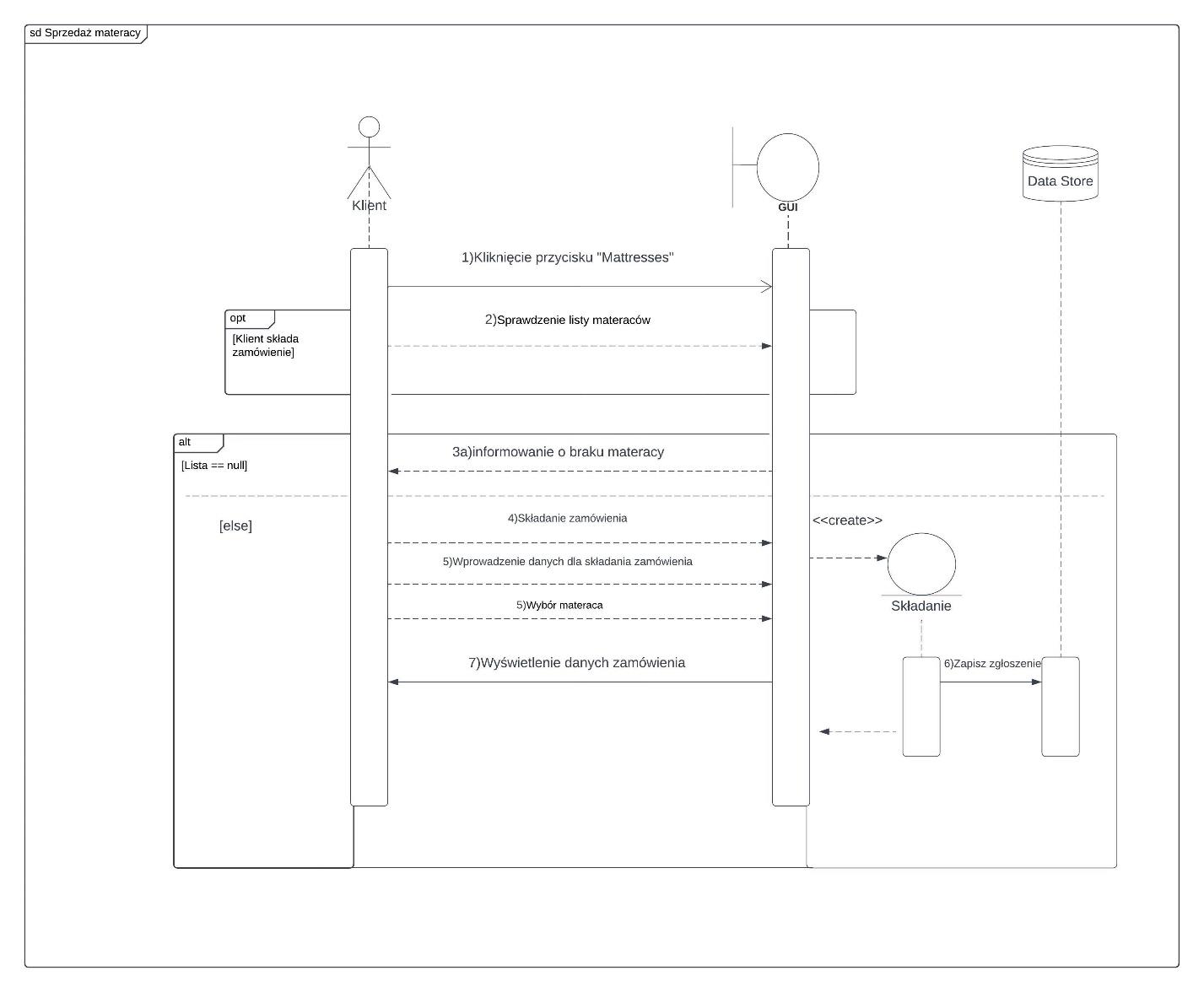
**Diagram aktywności dla przypadku użycia**

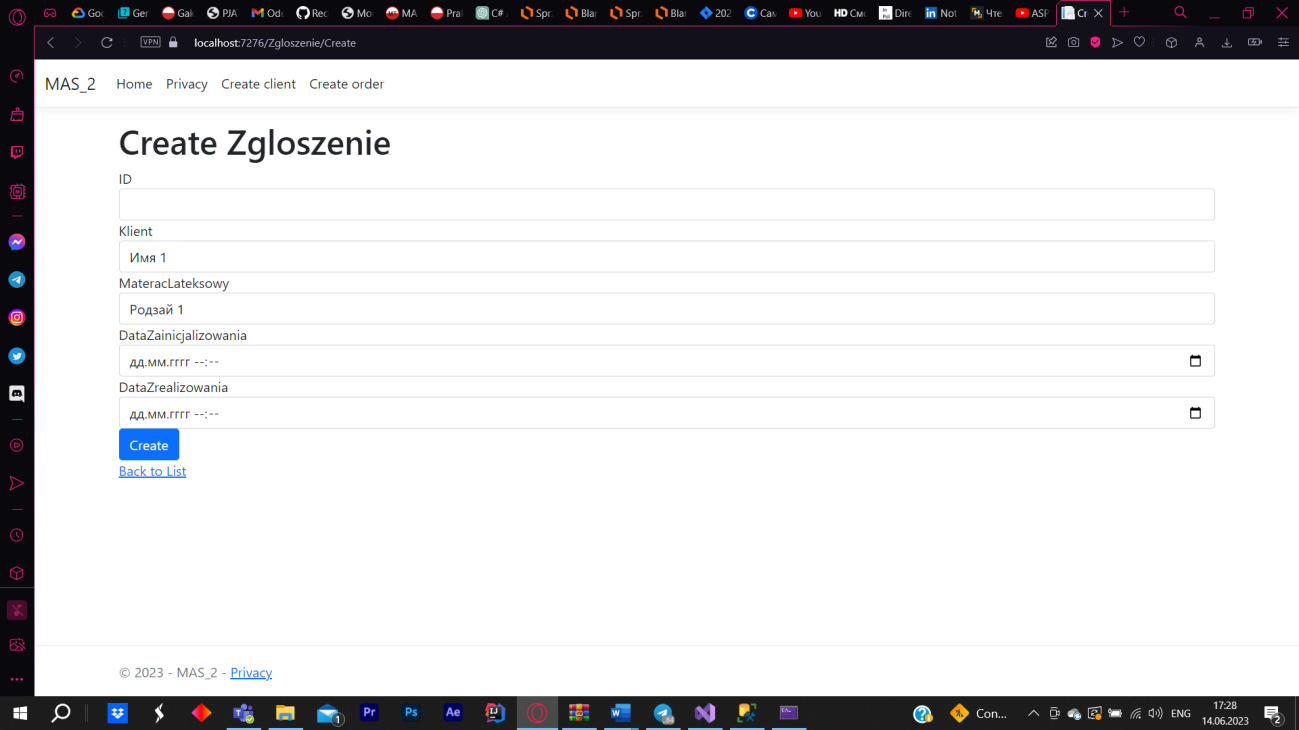


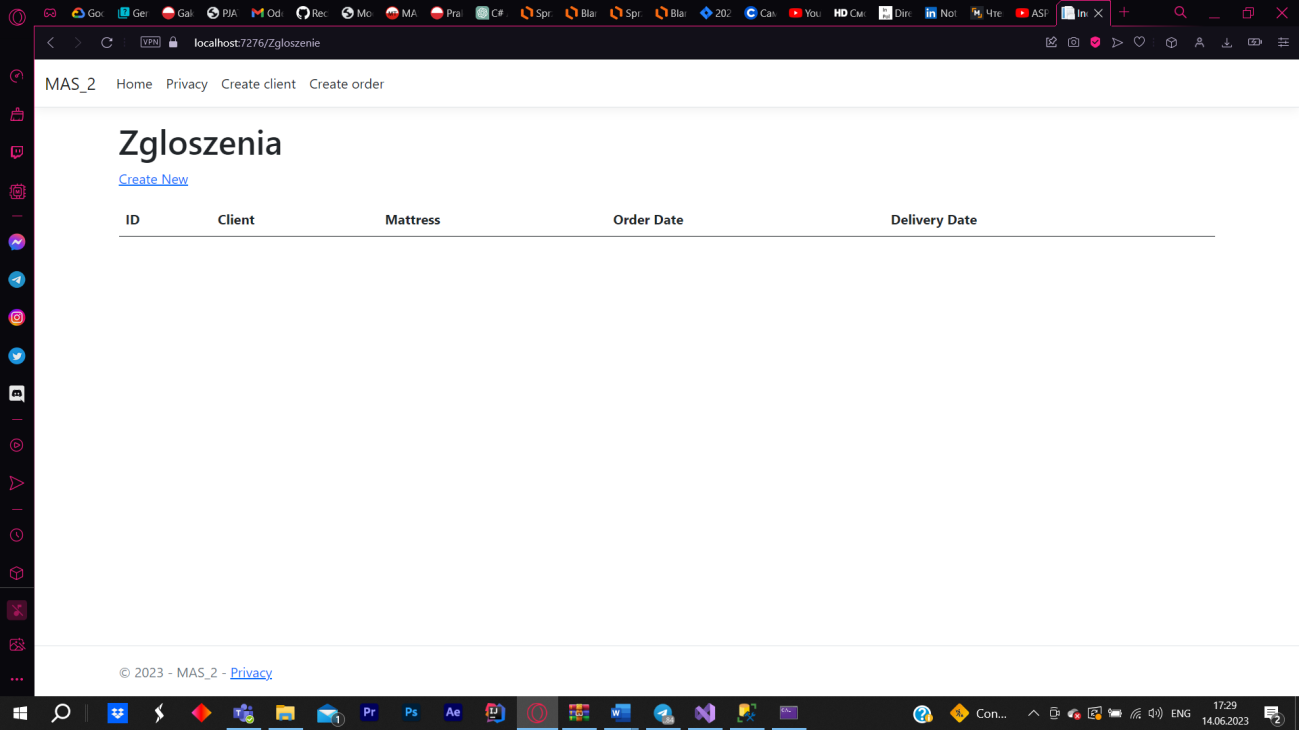
**Diagram stanu dla klasy**



**Diagram sekwencji dla przypadku użycia**



**Projekt GUI**



**Omówienie decyzji projektowych i skutków analizy dynamicznej**

**1. Wybrane decyzje projektowe**

**1.1 Django Framework MVT jako wzorzec projektowy**

W moim projekcie zdecydowalem się wykorzystać wzorzec projektowy MVT (Model-View-Template). MVT pomaga w separacji logiki biznesowej, warstwy prezentacji i zarządzania danymi. Dzięki temu aplikacja staje się bardziej odułowa, elastyczna i łatwa do zarządzania.

**1.2 Wybór języka programowania Python**

Do implementacji logiki biznesowej w moim projekcie wybrałem język programowania Python. Python jest językiem niesilnie typowanym, procedurowo-obiektowym i zapewnia wiele zaawansowanych funkcji, które ułatwiają rozwój aplikacji.

**1.3 Baza danych MySQL**

Jako system zarządzania bazą danych wybrałem MySQL. MySQL jest potężnym DBMS, który zapewnia nie tylko niezawodne przechowywanie danych, ale także zaawansowane funkcje związane z transakcjami, skalowalnością i bezpieczeństwem. Wybór MySQL pozwala efektywnie zarządzać danymi aplikacji.

**1.4 Migracje bazy danych**

Aby utrzymać spójność struktury bazy danych z modelem danych w projekcie, zdecydowałem się wykorzystać migracje bazy danych. Migracje pozwalają w łatwy sposób wprowadzać zmiany w strukturze bazy danych i utrzymywać jej wersjonowanie. Dzięki temu można bezpiecznie i łatwo aktualizować bazę danych wraz z rozwojem aplikacji.

**2. System szablonów:**

Silnik szablonów Django pozwala oddzielić logikę prezentacji od logiki biznesowej. Szablony to pliki HTML ze specjalną składnią, która umożliwia osadzanie dynamicznych treści i struktur kontrolnych. Obejmuje:

**2.1 Generowanie dynamicznego HTML**

Generowanie dynamicznego HTML-a: Django templates pozwalają na wstawianie zmiennych, instrukcji warunkowych i pętli bezpośrednio w kodzie HTML. Możesz użyć znacznika {{ }} do wstawienia wartości zmiennej, {% %} do instrukcji warunkowych i pętli, a także innych funkcji dostępnych w Django.

**2.2 Przekazywanie danych do kontrolerów**

Przekazywanie danych do kontrolerów: Django templates umożliwiają przekazywanie danych z szablonów do kontrolerów. Możesz użyć formularzy HTML do wprowadzania danych, a następnie przesłać je do kontrolera przy użyciu metody POST. W kontrolerze możesz odczytać przesłane dane i podjąć odpowiednie działania.

**2.3 Obsługa zdarzeń użytkownika**

Obsługa zdarzeń użytkownika: Django templates mogą obsługiwać zdarzenia użytkownika, takie jak kliknięcia, przekazywanie danych i wiele innych. Możesz użyć JavaScriptu w swoim kodzie HTML, aby reagować na te zdarzenia i wywoływać odpowiednie funkcje.

**2.4 Wyświetlanie danych z bazy danych**

Wyświetlanie danych z bazy danych: Django templates pozwalają na wyświetlanie danych pobranych z bazy danych. Możesz przekazać dane z widoku do szablonu i użyć ich w kodzie HTML.

**3. Użycie w projekcie dodatkowych elementów**

W projekcie wykorzystano także następujące elementy:

• Asocjacja z atrybutem: Wykorzystanie atrybutów w klasach do określania asocjacji między

różnymi klasami lub encjami w modelu danych.

• Kompozycja: Tworzenie hierarchii obiektów, gdzie obiekty podrzędne są związane z obiektem

nadrzędnym i zależą od niego.

• Asocjacja kwalifikowana: Określanie dodatkowych atrybutów lub kluczy dla asocjacji między

klasami, które nadają im dodatkowe znaczenie lub ograniczenia.

• Atrybut opcjonalny: Ustalanie, że pewne atrybuty mogą mieć wartość None lub nie być obecne w obiekcie.

• Ekstensja: Określanie, jak długo trwa istnienie obiektów w systemie.

• Atrybut złożony: Wykorzystywanie atrybutów składających się z kilku elementów, które

reprezentują bardziej złożone informacje.

• Atrybut powtarzalny: Określanie, że pewne atrybuty mogą mieć wiele wartości, nie tylko

pojedynczą.

• Atrybut klasowy: Przypisywanie atrybutów do klas, które mają wpływ na wszystkie instancje

tych klas.

• Atrybut pochodny: Tworzenie nowych atrybutów na podstawie istniejących atrybutów lub

innych czynników.

• Metoda klasowa: Tworzenie metod, które są związane z klasami, a nie z konkretnymi

instancjami tych klas.

• Przesłonięcie, przeciążenie: Zmiana zachowania lub dodawanie nowych funkcji do dziedziczonej klasy przez podklasę.

• Wielodziedziczenie: Tworzenie hierarchii klas, w której podklasa dziedziczy cechy po więcej niż

jednej klasie nadrzędnej.

**4. Podsumowanie**

Decyzje takie jak wybór wzorca MVT, języka Python oraz bazy danych MySQL miały kluczowy wpływ na architekturę i funkcjonalność aplikacji. Analiza dynamiczna plików pozwala generować dynamiczny interfejs użytkownika, przekazywać dane do kontrolerów, obsługiwać zdarzenia użytkownika i wyświetlać dane z bazy danych. Dodatkowo, w projekcie wykorzystano elementy takie jak asocjacja z atrybutem, kompozycja, asocjacja kwalifikowana, atrybuty opcjonalne, ekstensja, atrybuty złożone, atrybuty powtórne, atrybuty klasowe, atrybuty pochodne, metody klasowe, przesłonięcie, przeciążenie oraz wielodziedziczenie. Dzięki tym decyzjom projektowym i analizie dynamicznej, aplikacja jest elastyczna, skalowalna i dostarcza bogate funkcje dla użytkowników.