cizzz

# DOKUMENTACJA PROJEKTOWA

Sklep z podzespołami elektronicznymi

|  |  |
| --- | --- |
| Autorzy: |  |
| Status: | roboczy |
| Wersja: | 0.1 |
| Data modyfikacji: | 14.04.2024 |
| Abstrakt: | Dokument zawiera dokumentację metodyki, wymagań oraz projekt oprogramowania dla systemu Sklepu z podzespołami elektronicznymi. |

Historia zmian:

|  |  |  |  |
| --- | --- | --- | --- |
| **Wersja** | **Data** | **Kto** | **Opis** |
| 0.1 | 03.03.2024 |  | Założenie dokumentu |
| 0.2 | 14.04.2024 |  | Przygotowanie rejestru produktowego i sprintów |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Plan zadań w projekcie

## Rejestr produktowy

**User stories użytkowników: Administratora i Klienta.**

**User Stories dla Administratora**

* Jako administrator, chcę móc logować się do panelu administracyjnego, aby zarządzać sklepem.
  + Kryteria akceptacji:
    - możliwość zalogowania się za pomocą nazwy użytkownika i hasła;
    - dostęp do panelu tylko po autoryzacji;
    - komunikat błędu przy nieudanej próbie logowania.
* Jako administrator, chcę zarządzać danymi produktów, aby aktualizować, dodawać lub usuwać produkty z oferty.
  + Kryteria akceptacji:
    - możliwość dodawania nowych produktów;
    - edycja istniejących produktów;
    - usuwanie produktów;
    - przeglądanie listy produktów.
* Jako administrator, chcę zarządzać czasowymi promocjami, aby przyciągać więcej klientów poprzez oferty specjalne.
  + Kryteria akceptacji:
    - tworzenie nowych promocji;
    - modyfikacja istniejących promocji;
    - usuwanie promocji;
    - przeglądanie aktywnych promocji.
* Jako administrator, chcę zarządzać danymi klientów, aby mieć aktualne i kompletne informacje o użytkownikach sklepu.
  + Kryteria akceptacji:
    - przeglądanie listy klientów i ich zakupów, generowanie raportów sprzedaży;
    - edycja danych klientów;
    - usuwanie klientów z systemu.

**User Stories dla Klienta**

* Jako klient, chcę się zalogować, aby mieć dostęp do mojego profilu i historii zakupów.
  + Kryteria akceptacji:
    - możliwość zalogowania się za pomocą e-maila i hasła;
    - przypomnienie zapomnianego hasła;
    - dostęp do profilu tylko dla zalogowanych użytkowników.
* Jako klient, chcę przeglądać listę produktów, aby wybrać i kupić te, które mnie interesują.
  + Kryteria akceptacji:
    - przeglądanie listy produktów z opcjami sortowania, wyszukiwania i filtrowania;
    - wyświetlanie szczegółowych informacji o produkcie;
    - możliwość dodania produktu do koszyka.
* Jako klient, chcę korzystać z koszyka, aby gromadzić produkty przed zakupem.
  + Kryteria akceptacji:
    - dodawanie produktów do koszyka;
    - modyfikacja ilości produktów w koszyku;
    - usuwanie produktów z koszyka;
    - wyświetlanie sumy ceny za wszystkie produkty w koszyku.
* Jako klient, chcę dokonać zakupu poprzez proces złożenia zamówienia.
  + Kryteria akceptacji:
    - złożenie zamówienia;
    - podanie danych do wysyłki;
    - wybór metody płatności;
    - potwierdzenie zamówienia i otrzymanie potwierdzenia mailem.

**Sprint 1**

**Cel: konfiguracja środowiska pracy i stworzenie szkieletu aplikacji.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Przypadek użycia** | **Nazwa** | **Priorytet** | **Ilość godzin** |
| PU01 | Przygotowanie środowiska deweloperskiego i narzędziowego dla zespołu | Bardzo Wysoki | 24 |
| PU02 | Projektowanie architektury systemu i bazy danych | Wysoki | 8 |
| PU03 | Logowanie administratora | Bardzo Wysoki | 8 |
| PU04 | Rejestracja i logowanie klienta | Wysoki | 8 |

Po realizacji **Sprintu 1** uzyskamy system logowania działający dla administratorów i klientów z podstawowym interfejsem użytkownika.

**Sprint 2**

**Cel: implementacja podstawowych funkcji związanych z użytkownikami oraz wprowadzenie podstawowych funkcji zarządzania produktami.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Przypadek użycia** | **Nazwa** | **Priorytet** | **Ilość godzin** |
| PU05 | Zarządzanie danymi produktów (CRUD): dodawanie,, wyświetlanie, usuwanie, edytowanie | Bardzo Wysoki | 16 |
| PU06 | Zarządzanie kontami użytkowników: usuwanie, edytowanie | Bardzo Wysoki | 12 |

Po realizacji **Sprintu 2** uzyskamy funkcje zarządzania produktami i użytkownikami z możliwością edycji i usuwania.

**Sprint 3**

**Cel: rozwój funkcjonalności związanych z produktami.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Przypadek użycia** | **Nazwa** | **Priorytet** | **Ilość godzin** |
| PU07 | Wyszukiwanie produktów | Wysoki | 16 |
| PU08 | Sortowanie produktów | Wysoki | 8 |
| PU09 | Filtrowanie produktów | Wysoki | 16 |
| PU10 | Szczegółowe widoki produktów | Wysoki | 8 |
| PU11 | Recenzje i oceny produktów | Średni | 12 |

Po realizacji **Sprintu 3** uzyskamy rozszerzone funkcje zarządzania produktami z wyszukiwaniem i filtrowaniem.

**Sprint 4**

**Cel: implementacja koszyka zakupowego i procesu składania zamówień.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Przypadek użycia** | **Nazwa** | **Priorytet** | **Ilość godzin** |
| PU12 | Dodawanie i usuwanie produktów do koszyka | Wysoki | 12 |
| PU13 | Modyfikacja ilości produktów w koszyku | Wysoki | 8 |
| PU14 | Składanie i realizacja zamówienia | Wysoki | 12 |
| PU15 | Dodanie metod płatności i integracja z zewnętrznymi systemami płatniczymi. | Wysoki | 20 |

Po realizacji **Sprintu 4** uzyskamy funkcjonalność koszyka zakupowego i procesu realizacji zamówienia.

**Sprint 5**

**Cel: przygotowanie aplikacji do wdrożenia i uruchomienia.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Przypadek użycia** | **Nazwa** | **Priorytet** | **Ilość godzin** |
| PU16 | Generowanie raportów sprzedaży w panelu administratora | Średni | 12 |
| PU17 | Zarządzanie promocjami | Średni | 8 |
| PU18 | Zarządzanie danymi klientów | Niski | 8 |

Po realizacji **Sprintu 5** uzyskamy w pełni funkcjonalny sklep internetowy z możliwościami zarządzania, zakupów oraz raportowania.

## Rejestr Sprintów

### Rejestr Sprintu 1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **PU** | **Opis** | **Ilość godzin** | **pn** | **wt** | **śr** | **czw** | **pt** |
| PK, BS, PH | Spotkanie zespołowe, wybór szkieletu aplikacji, wybór rodzaju bazy danych, wybór IDE | 4 | 1 |  |  |  |  |
| PK, PH | Spotkanie zespołu graficznego, omówienie wstępnej koncepcji graficznej | 3 | 1 |  |  |  |  |
| ZH | Konfiguracja IntelliJ 2023.2.5 Communiti Edition dla każdego członka zespołu, instalacja środowisk (Spring Boot 3.2.3), przygotowanie obrazów, dystrybucja kluczy, ustanowienie repozytorium kodu | 8 |  | 1 |  |  |  |
| PU02 | Projektowanie modeli danych dla użytkowników i produktów, definiowanie relacji i schematu bazy danych; projektowanie ogólnej architektury aplikacji, w tym podziału na warstwy i główne komponenty systemu |  |  |  |  |  |  |
| Implementacja klas encji dla użytkowników i produktów w Javie, konfiguracja Spring Data JPA dla operacji na bazie danych |  |  |  |  |  |  |
| Stworzenie podstawowego interfejsu użytkownika oraz struktury nawigacji – projektowanie wzoru *User Interface* przy użyciu HTML5, CSS3 i JavaScript |  |  |  |  |  |  |
| Implementacja podstawowej nawigacji po stronie aplikacji (menu, przeglądanie produktów, strona logowania / rejestracji) |  |  |  |  |  |  |
| PU03 | Projektowanie i implementacja *User Interface* formularza logowania dla administratorów, przygotowanie warstwy frontend |  |  |  |  |  |  |
| Stworzenie mechanizmu autentykacji i sesji administratora, przygotowanie warstwy backend |  |  |  |  |  |  |
| Integracja systemu logowania z bazą danych H2 dla przechowywania danych o użytkownikach |  |  |  |  |  |  |
| Implementacja bezpiecznego przechowywania haseł |  |  |  |  |  |  |
| Testowanie funkcjonalności logowania |  |  |  |  |  |  |
| PU04 | Projektowanie i implementacja *User Interface* formularza rejestracji i logowania dla klientów, przygotowanie warstwy frontend |  |  |  |  |  |  |
| Stworzenie mechanizmu rejestracji, autentykacji i sesji klienta, przygotowanie warstwy backend |  |  |  |  |  |  |
| Integracja systemu logowania z bazą danych H2 dla przechowywania danych o użytkownikach |  |  |  |  |  |  |
| Implementacja mechanizmów bezpieczeństwa |  |  |  |  |  |  |
| Testowanie funkcjonalności logowania |  |  |  |  |  |  |

### Rejestr Sprintu 2

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **PU** | **Opis** | **Ilość godzin** | **pn** | **wt** | **śr** | **czw** | **pt** |
| PU05 | Projektowanie i implementacja *User Interface* formularza dodawania, wyświetlania, edycji i usuwania produktów, przygotowanie warstwy frontend |  |  |  |  |  |  |
| Tworzenie formularza w panelu administracyjnym do wprowadzania danych produktu (opcja „Dodaj produkt”), edycji produktu, dodanie opcji „Usuń” w interfejsie użytkownika panelu administracyjnego przy każdym produkcie, przygotowanie warstwy backend |  |  |  |  |  |  |
| Integracja formularza z bazą danych |  |  |  |  |  |  |
| Zapewnienie walidacji danych na formularzu |  |  |  |  |  |  |
| Implementacja potwierdzenia dodania, modyfikacji, usunięcia produktu |  |  |  |  |  |  |
| Testy jednostkowe dla funkcji dodawania, wyświetlania, edycji i usuwania produktów, |  |  |  |  |  |  |
| PU06 | Projektowanie i implementacja *User Interface* do zarządzania kontami użytkowników |  |  |  |  |  |  |
| Przygotowanie warstwy backend |  |  |  |  |  |  |
| Integracja z bazą danych |  |  |  |  |  |  |
| Implementacja potwierdzenia modyfikacji, usunięcia konta |  |  |  |  |  |  |
| Testy jednostkowe |  |  |  |  |  |  |

### Rejestr Sprintu 3

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **PU** | **Opis** | **Ilość godzin** | **pn** | **wt** | **śr** | **czw** | **pt** |
| PU07 | Projektowanie i implementacja *User Interface* do wyszukiwania produktów na podstawie nazwy, kategorii, ceny, dodanie opcji wyszukania w interfejsie użytkownika na stronie z listą produktów, przygotowanie warstwy frontend |  |  |  |  |  |  |
| Przygotowanie warstwy backend |  |  |  |  |  |  |
| Integracja funkcji wyszukiwania z bazą danych |  |  |  |  |  |  |
| Testy jednostkowe dla funkcji wyszukiwania |  |  |  |  |  |  |
| PU08 | Projektowanie i implementacja *User Interface* do sortowania produktów na podstawie nazwy, kategorii, ceny, dodanie opcji wyszukania w interfejsie użytkownika na stronie z listą produktów, przygotowanie warstwy frontend |  |  |  |  |  |  |
| Przygotowanie warstwy backend |  |  |  |  |  |  |
| Integracja logiki sortowania z bazą danych |  |  |  |  |  |  |
| Testy dla funkcji sortowania |  |  |  |  |  |  |
| PU09 | Projektowanie i implementacja *User Interface* do filtrowania produktów na podstawie nazwy, kategorii, ceny, dodanie opcji filtrowania w interfejsie użytkownika na stronie z listą produktów, przygotowanie warstwy frontend |  |  |  |  |  |  |
| Przygotowanie warstwy backend |  |  |  |  |  |  |
| Integracja logiki filtrowania w bazie danych |  |  |  |  |  |  |
| Testy dla funkcji filtrowania |  |  |  |  |  |  |
| PU10 | Projektowanie i implementacja szczegółowej strony produktu, przygotowanie warstwy frontend |  |  |  |  |  |  |
| Przygotowanie warstwy backend |  |  |  |  |  |  |
| Integracja z bazą danych dla wyświetlenia kompletnych informacji |  |  |  |  |  |  |
| Testy dla szczegółowego widoku produktu |  |  |  |  |  |  |
| PU11 | Projektowanie i implementacja *User Interface* do dodawania i wyświetlania recenzji i ocen, przygotowanie warstwy frontend |  |  |  |  |  |  |
| Przygotowanie warstwy backend |  |  |  |  |  |  |
| Integracja systemu ocen i recenzji z bazą danych |  |  |  |  |  |  |
| Testy |  |  |  |  |  |  |

### Rejestr Sprintu 4

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **PU** | **Opis** | **Ilość godzin** | **pn** | **wt** | **śr** | **czw** | **pt** |
| PU12 | Projektowanie i implementacja interfejsu dla dodawania i usuwania produktów z koszyka, przygotowanie warstwy frontend |  |  |  |  |  |  |
| Przygotowanie warstwy backend |  |  |  |  |  |  |
| Implementacja logiki zarządzania stanem koszyka na serwerze i w przeglądarce użytkownika |  |  |  |  |  |  |
| Testy jednostkowe i integracyjne |  |  |  |  |  |  |
| PU13 | Projektowanie i implementacja interfejsu dla modyfikacji ilości produktów w koszyku, przygotowanie warstwy frontend |  |  |  |  |  |  |
| Przygotowanie warstwy backend |  |  |  |  |  |  |
| Implementacja mechanizmów aktualizacji ilości produktów w koszyku bez przeładowywania strony |  |  |  |  |  |  |
| Testy jednostkowe i integracyjne |  |  |  |  |  |  |
| PU14 | Projektowanie i implementacja *User Interface* dla procesu składania zamówienia, przygotowanie strony podsumowania zamówienia, przygotowanie warstwy frontend |  |  |  |  |  |  |
| Przygotowanie warstwy backend |  |  |  |  |  |  |
| Integracja z bazą danych (z systemem zarządzania stanem produktów dla aktualizacji dostępności po złożeniu zamówienia) |  |  |  |  |  |  |
| Implementacja wysyłania potwierdzeń zamówienia na adres email klienta |  |  |  |  |  |  |
| Testy całego procesu zamówienia |  |  |  |  |  |  |
| PU15 | Projektowanie i implementacja interfejsu API dla obsługi różnych metod płatności, przygotowanie warstwy frontend |  |  |  |  |  |  |
| Przygotowanie warstwy backend |  |  |  |  |  |  |
| Testy |  |  |  |  |  |  |

### Rejestr Sprintu 5

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **PU** | **Opis** | **Ilość godzin** | **pn** | **wt** | **śr** | **czw** | **pt** |
| PU16 | Projektowanie i implementacja *User Interface* do zbierania danych sprzedażowych i generowania raportów, przygotowanie warstwy frontend |  |  |  |  |  |  |
| Przygotowanie warstwy backend |  |  |  |  |  |  |
| Integracja funkcji raportowania z bazą danych |  |  |  |  |  |  |
| Testy jednostkowe i integracyjne |  |  |  |  |  |  |
| PU17 | Projektowanie i implementacja *User Interface* do zarządzania promocjami, przygotowanie warstwy frontend |  |  |  |  |  |  |
| Integracja logiki promocji z systemem sklepu i bazą danych (m.in. aktualizacja cen produktów w czasie promocji), przygotowanie warstwy backend |  |  |  |  |  |  |
| Implementacja walidacji czasu trwania promocji i automatyczne wyłączanie zakończonych promocji |  |  |  |  |  |  |
| Testy jednostkowe i integracyjne |  |  |  |  |  |  |
| PU18 | Projektowanie i implementacja *User Interface* do zarządzania danymi klientów, przygotowanie warstwy frontend |  |  |  |  |  |  |
| Przygotowanie warstwy backend (przeglądanie, edycja danych klientów, usuwanie kont klientów) |  |  |  |  |  |  |
| Integracja z bazą danych |  |  |  |  |  |  |
| Testy jednostkowe i integracyjne |  |  |  |  |  |  |

## Założenia

Środowisko: Spring Boot 3.2.3

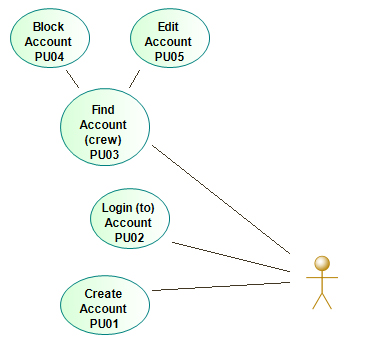
Java: version 17

IDE: IntelliJ 2023.2.5 Community Edition

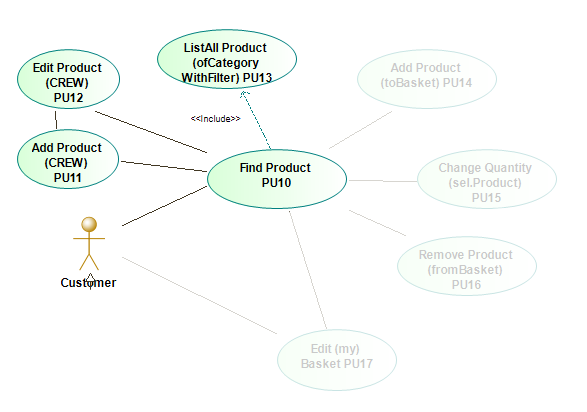
Baza danych:  
 środowiska developerskie: H2

# Opis wymagań funkcjonalnych systemu

## Model przypadków użycia



Rysunek 1 : Diagram przypadków użycia dla Sprintu 1 - Account (1)



Rysunek 1 : Diagram przypadków użycia dla Sprintu 1 - Product (1)

## Scenariusze przypadków użycia dla sprintu 1

**PU01: Create Account**

{rola: Niezalogowany}

01: Aktor wybiera zakładkę [ GET: / account / create\_account ]

02: System wyświetla widok [ VIEW: account / create\_account ]

03: Aktor wprowadza informacje [ CLASS Account: mail, password, firstName, lastName, status: 1,   
 ew. adress, ew NIP , ew. zdjęcie ]

04: Aktor klika Zapisz [ FORM: account, POST: account / create\_account ( account ) ]

05: System waliduje dane [ METHOD: account . create\_account\_POST ( account ) ]

[ dane prawidłowe ] ( wolny e-mail )

06: System zapisuje dane w bazie [ account ]

07: System aktualizuje [ status ostatniej akcji ] [ INFO: użytkownik zapisany + last Id]

08: System wyświetla [ status ostatniej akcji ] w widok [ / ]

**[SUCCES]**

[ dane nieprawidłowe ]

07: System aktualizuje [ status ostatniej akcji ] [ INFO: błąd formularza ]

08: System wyświetla [ status ostatniej akcji ] w widok [ account / create\_account ]

**PU02: Login (to) Account**

{rola: Niezalogowany}

01: Aktor wybiera zakładkę [ GET: / account / login ]

02: System wyświetla widok [ VIEW: account / login ]

03: Aktor wprowadza informacje [ mail, password ]

04: Aktor klika Login [ FORM: account / login ( account ) ]

05: System waliduje dane [ METHOD: account . login\_POST ]

[ dane prawidłowe ]

06: System zapisuje dane w bazie [ Przypisanie accoundId do SESJI ]

07: System aktualizuje [ status ostatniej akcji ] [ INFO: witaj ponownie: + userName ]

08: System wyświetla [ status ostatniej akcji ] w widok [ / ]

**[SUCCES]**

[ dane nieprawidłowe ]

`07: System aktualizuje [ status ostatniej akcji ] [ INFO: nieprawidłowe dane logowania ]

`08: System wyświetla [ status ostatniej akcji ] w widok [ account / login ]

`01: Aktor wybiera zakładkę [ GET: / account / przypomnij hasło ]

’02: System wyświetla widok [ View: / account / password\_reset ] -> **PU01A**

**PU03: Find Account**

{rola: Crew}

01: Aktor wybiera zakładkę [ GET: / account / find\_account ]

02: System wyświetla widok [ VIEW: account / find\_account ]

03: Aktor wprowadza informacje [ mail lub firstName, lub lastName lub NIP ]

04: System wyszukuje listę kont [ METHOD: account . find\_account( data ) ]

05: System przygotowuje listę kont do wyświetlenia [ MODEL: accountList ]

06: System wyświetla widok [ VIEW: / account / list ( accountList )]

07: Aktor zaznacza konto [ accountId ]

08: -> Aktor wywołuje żądanie [ GET: / account / edit / { accountId } ]

**[ SUCCESS ]**

**PU05: Edit Account**

{rola: Crew, selected: accountId }

01: Aktor wybiera zakładkę [ GET: / account / edit / accountId ]

02: System pobiera z bazy obiekt account [ by accountId ]

03: System wyświetla widok [ VIEW: account / edit ( account = account ) ]

03: Aktor wprowadza informacje [ account ]

04: Aktor klika zapisz [ FORM: account, POST: / account / edit ( account ) ]

05: System waliduje dane [ METHOD: account . edit ( accountId , account ) ]

[ dane zgodne ]

06: System zapisuje do bazy account

07: System aktualizuje [ status ostatniej akcji ] [ INFO: dane zapisane: + userName ]

08: System wyświetla [ status ostatniej akcji ] w widok [ / ]

**[ SUCCESS ]**

[ dane niezgodne ]

`07: System aktualizuje [ status ostatniej akcji ] [ INFO: dane niezgodne ]

`08: System wywołuje -> GET: / account / edit / accountId

**PU04: Block Account**

{rola: Crew, selected: accountId }

01: Aktor wybiera zakładkę [ GET: / account / block / accountId ]

02: System pobiera z bazy obiekt account [ by accountId ]

03: System ustawia account.status = 0

04: system zapisuje account w bazie

**[ SUCCESS ]**

**PU10: Find Product**

{rola: User}

01: Aktor wybiera zakładkę [ GET: / product / list + / ( categoryId ) / ( filter ) ]

02: System pobiera productList dane z bazy

03: System wyświetla [ VIEW: / product / list ( productList ) ]

04: Aktor wybiera produktId

05: -> Aktor wywołuje żądanie [ GET: / product / info / { productId } ]

**[ SUCCESS ]**

**PU10: Edit Product**

{rola: Crew}

01: Aktor wybiera zakładkę [ GET: / product / edit / { productId } ]

02: System pobiera product dane z bazy

03: System wyświetla wypełnioną formatkę [ VIEW: / product / edit ( product ) ]

04: Aktor edytuje dane [ product ]

04: Aktor klika zapisz [ FORM: product, POST: / product / edit ( product ) ]

05: System waliduje dane [ METHOD: account . edit ( account ) ]

[ dane zgodne ]

06: System zapisuje do bazy account

07: System aktualizuje [ status ostatniej akcji ] [ INFO: produkt zapisany ]

08: System wyświetla [ status ostatniej akcji ] w widok [ / product ]

**[ SUCCESS ]**

[ dane niezgodne ]

`07: System aktualizuje [ status ostatniej akcji ] [ INFO: błąd formularza ]

`08: System wyświetla [ status ostatniej akcji ]

`09: przekierowanie -> 01: GET: / product / edit / { productId }

**PU11: Add Product**

{rola: Crew}

01: Aktor wybiera zakładkę [ GET: / product / edit / { NULL } ]

02: System wywołuje tworzy nowy\_produkt

03: System zapisuje w bazie nowy\_produkt

04: System pobiera nowy identyfikator productId

05: System przekierowuje na: [ GET: / product / edit / { productId } ]

**PU13: = PU10**

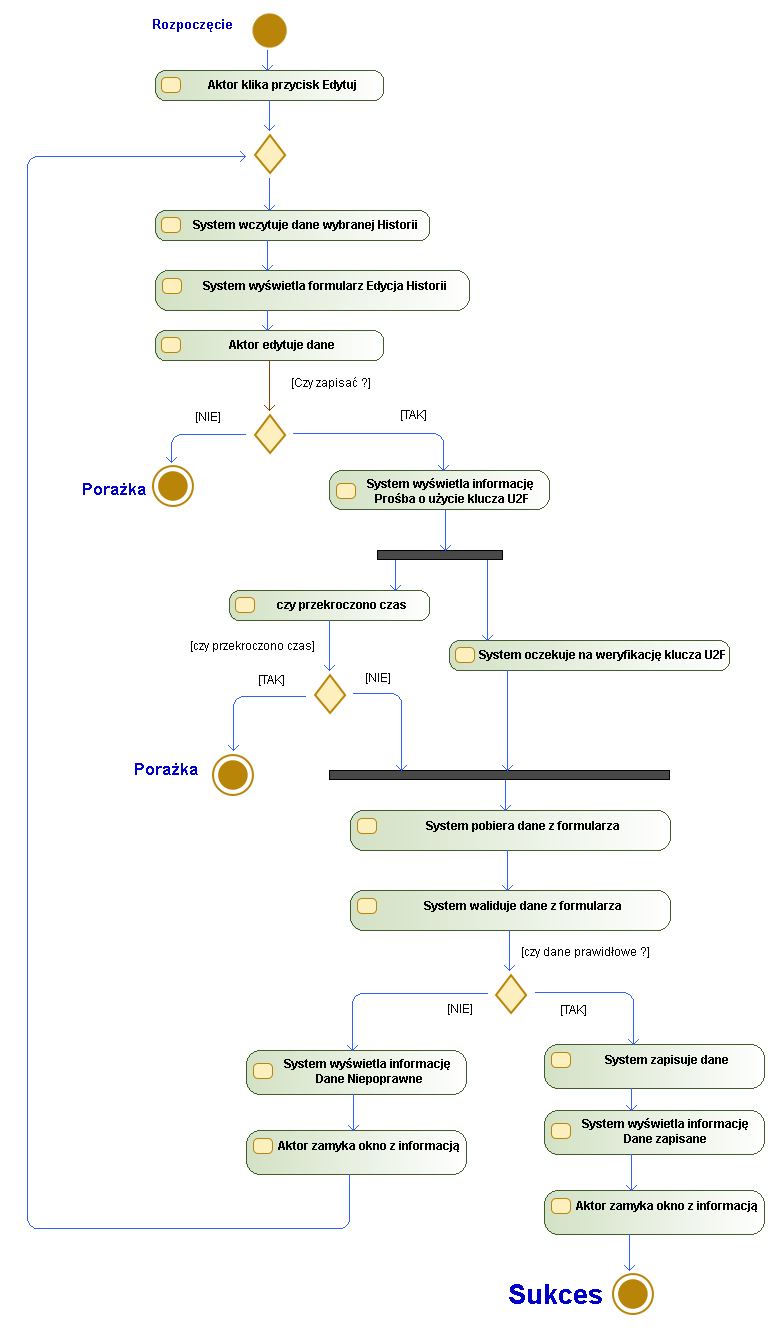


Diagram czynności PU2: Edytuj Historię [ Identyfikator Historii ]

**PU2: Edytuj Historię [ Identyfikator Historii ]**

{rola: Pisarz, aktywna zakładka [Historia ] wybrany element [ Historia ]}

01: Aktor klika przycisk [ Edytuj ]

02: System wczytuje dane wybranej [Historii ]

03: System wyświetla formularz [ Edycja Historii ]

04: Aktor edytuje dane

05: Aktor klika [ Anuluj ]

06: idz do → **PU 03 Znajdz Historię**

**[ NIEPOWODZENIE ]**

**PU2: Edytuj Historię [ Identyfikator Historii ] – scenariusz alternatywny**

05’: Aktor klika [ Zapisz zmiany ]

06’: System wyświetla informację [ Prośba o użycie klucza U2F ]

07’: System oczekuje na weryfikację klucza U2F

[ przekroczony czas weryfikacji ]

08’: idz do **PU 03 Znajdz Historię**

**[ NIEPOWODZENIE ]**

**– scenariusz alternatywny**

[ klucz zgodny ]

08’’: System pobiera dane z formularza

09’’: System waliduje dane

[ dane prawidłowe ]

10’’: System zapisuje [ Historię ]

11’’: System wyświetla informację [ Dane zapisane ]

12’’: Aktor klika [ Zamknij ] ( lub [ ESC ] )

13’’: idz do → **PU 03 Znajdz Historię**

**[ SUKCES ]**

**– scenariusz alternatywny**

[ dane niepoprawne ]

10’’’: System wyświetla komunikat [ błąd w formularzu ]

11’’’: Aktor klika [ Zamknij ] ( lub [ ESC ])

12’’’: idz do → 02

**PU04: Dodaj Wydarzenie**

{rola: Zarządca, aktywna zakładka [Historia ] wybrany element [ Historia ] }

01: Aktor wybiera przycisk [ Dodaj Wydarzenie ]

03: System wyświetla formatkę [ Nazwa Wydarzenie ]

04: Aktor wprowadza [ Nazwę Wydarzenia ]

05: Aktor klika przycisk [ Zapisz ]

06: System waliduje wprowadzone dane

[ dane prawidłowe ]

07: System zapisuje nowe wydarzenie

08: System zachowuje nowy identyfikator

[ SUKCES ]

09: idz do → **PU 05 Edytuj Wydarzenie [ Identyfikator Wydarzenia ]**

**PU01: Dodaj Wydarzenie – scenariusz alternatywny**

[ dane niepoprawne ]

09: System wyświetla komunikat [ niewłaściwa nazwa ]

**10: idz do → 03**

**PU5: Edytuj Wydarzenie [ Identyfikator Wydarzenia ]**

{rola: Zarządca, aktywna zakładka [Wydarzenia ] wybrany element [ Wydarzenie ]}

01: Aktor klika przycisk [ Edytuj ]

02: System wczytuje dane wybranego [ Wydarzenie ]

03: System wyświetla formularz [ Edycja Wydarzenie ]

04: Aktor edytuje dane

05: Aktor klika [ Zapisz zmiany ]

06: System waliduje dane

[ dane prawidłowe ]

07: System zapisuje [ Wydarzenie ]

08: System wyświetla informację [ Dane zapisane ]

**[ SUKCES ]**

**– scenariusz alternatywny**

[ dane niepoprawne ]

07’: System wyświetla informację [ błąd w formularzu ]

08’: Aktor zamyka informację

09’: idz do → 04

**PU06: Szukaj Wydarzenie**

{rola: Gracz}

01: Aktor wybiera zakładkę [ Wydarzenie ]

02: Aktor wybiera przycisk [ Szukaj Wydarzenia ]

03: System pobiera listę [ opisów Wydarzeń ]

04: System wyświetla listę [ opisów Wydarzeń ]

05: Aktor wybiera element [ Wydarzenie ] z listy

[ SUKCES ]

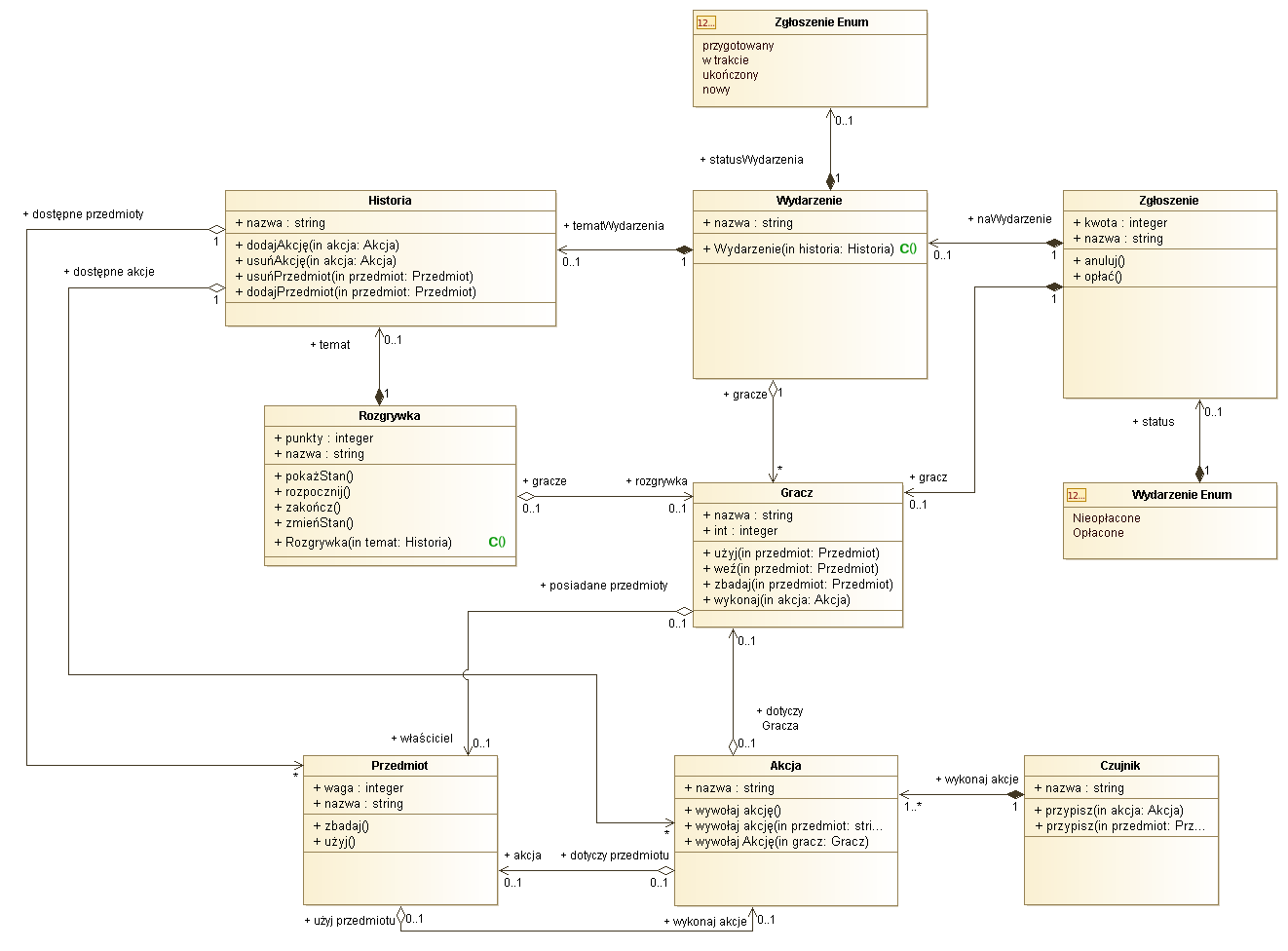
– scenariusz alternatywny

05’: Aktor klika [ Wyjście ]

06’: idz do zakładki [ Wydarzeń ]

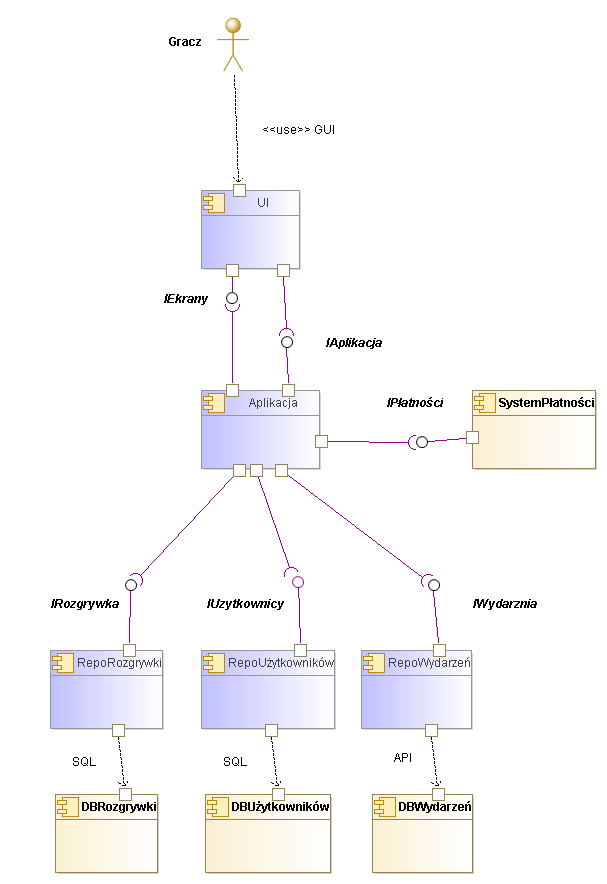
[ PORAŻKA ]

# Słownik dziedziny problemu



Rysunek 4 : Diagram klas słownika dla obszaru wydarzenia

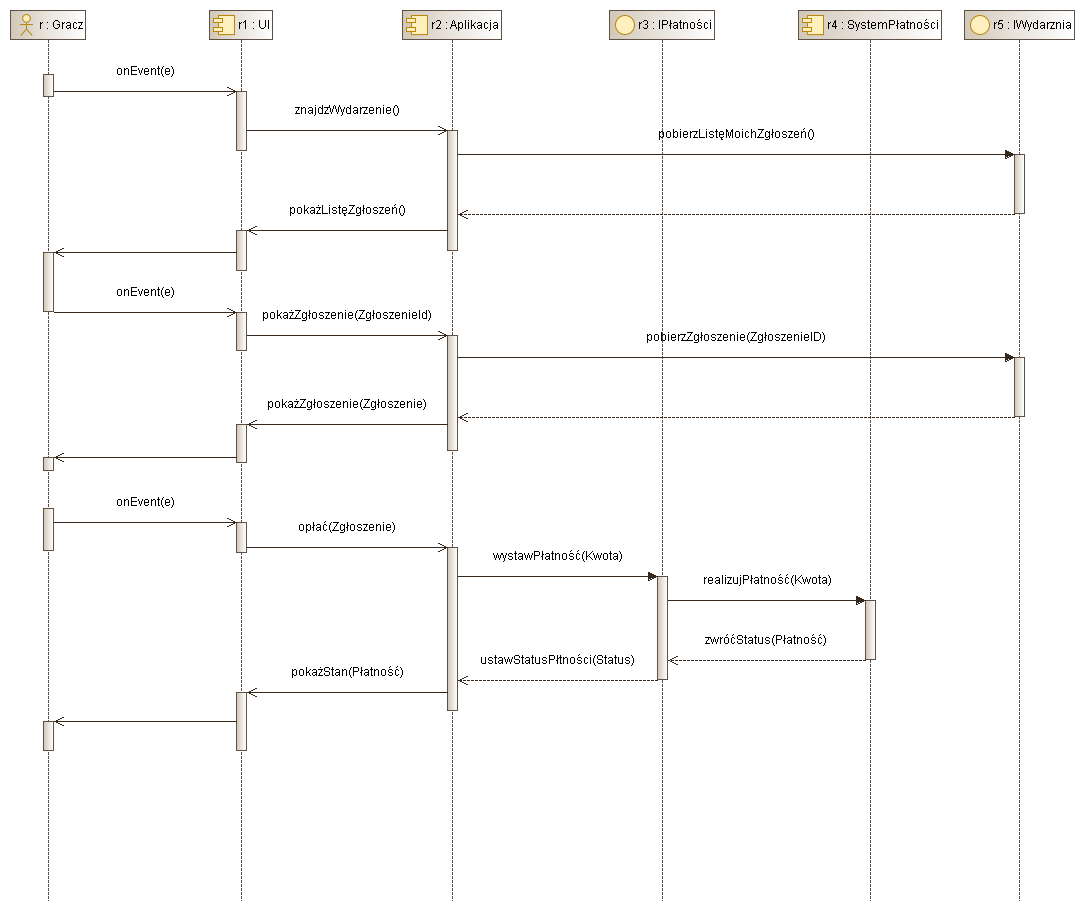
# Projekt struktury systemu



Rysunek 5 : Diagram komponentów systemu zarządzania LARP

zastosowałem strukturę 3 warstw, pierwsza warstwa to warstwa komunikacji z użytkownikiem UI, druga to logika aplikacji przetwarzająca zdarzenia od użytkownika oraz przekazująca odpowiedzi systamu do użytkownika oraz logika dziedziny - główny mózg aplikacji, która używa warstwy 3 interfejsów dostarczonych przez adaptery do utrawalania danych. Adaptery przetwarzają dane i utrwalają je w w bazach danych.

# Projekt dynamiki systemu

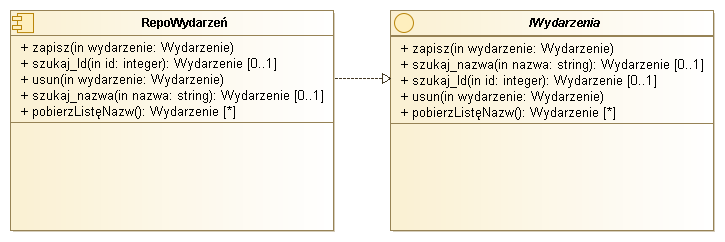


Rysunek 6 : Diagram sekwencji dla realizacji przypadku użycia PU06 i PU09

Użytkownik wykonuje czynność PU06 – odnajduje Zgłoszenie, System znajduje zgłoszenie i pokazuje na ekranie. Uzytkownik chcąc je opłacić – pobiera informacje o zgłoszeniu, a następnie opłaca je korzystając z zewnętrznego systemu płatności – który ustawia status płatności zgodnie z odpowiedzią banku, korzystając z przygotowanego przez bank API.

# Struktura kodu systemu

## Realizacja interfejsów



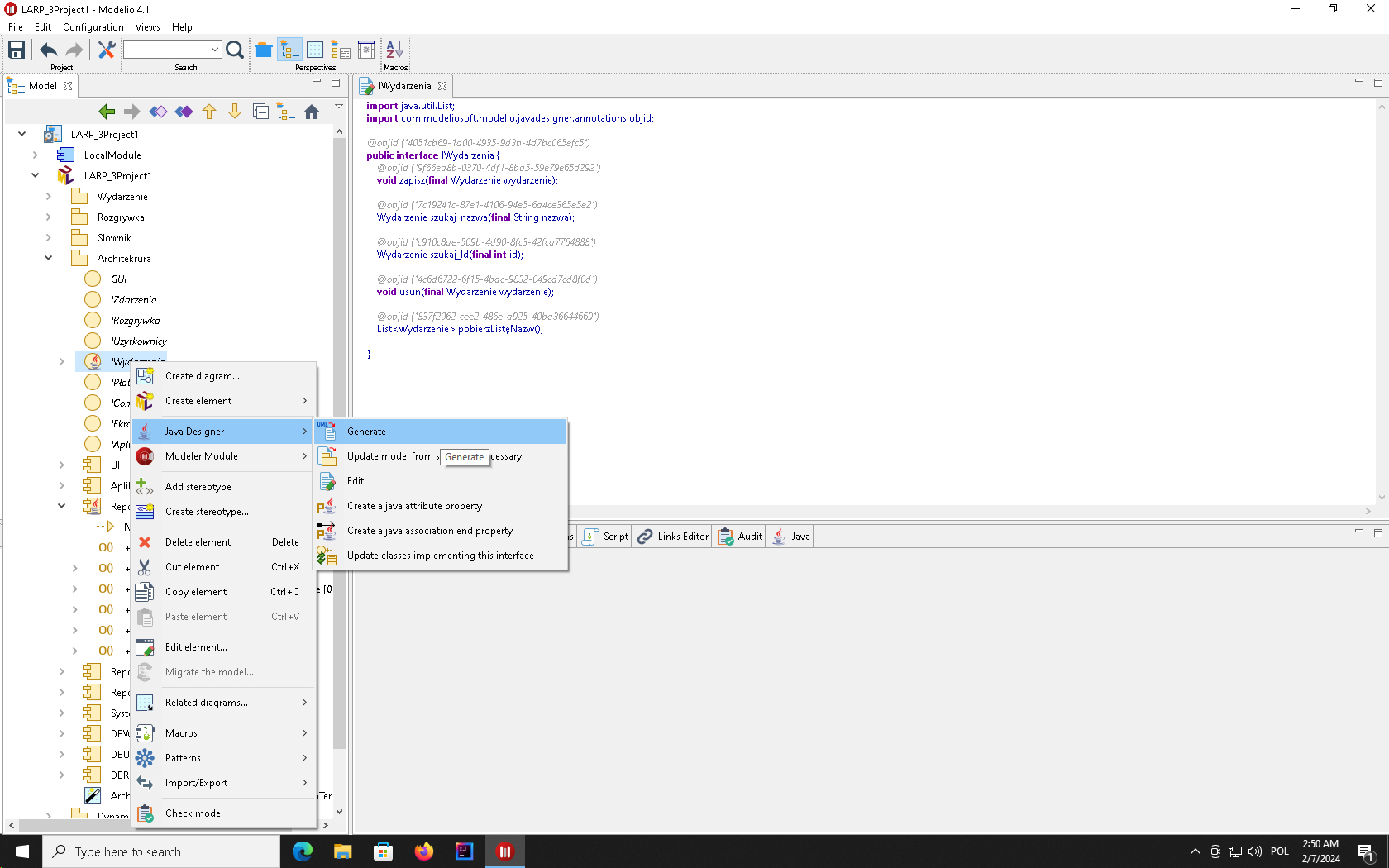
Rysunek 7 : Diagram klas przedstawiający realizację interfejsu IWydarzenia

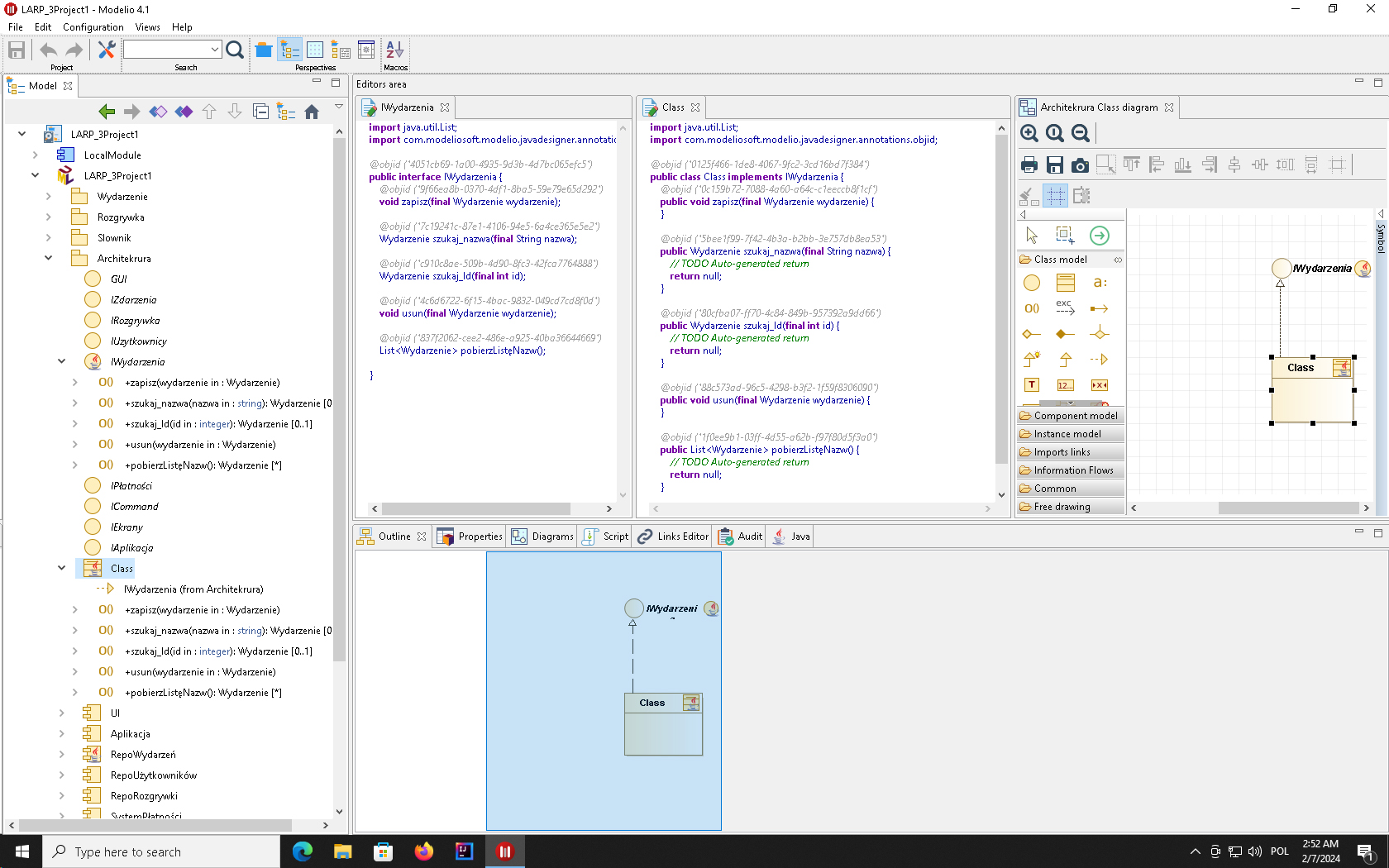
## Generacja kodu

Kod interfejsu IWydarzenia oraz klasy go implementującej Class, w języku programowania Java, wygenerowany narzędziem JavaDesigner .

Aby to zrobić należy zainstalować Moduł JavaDesigner, a następnie mając interfejs utworzyć diagram na którym klasa implementująca będzie implementowała interfejs. Następnie klikając lewym przyciskiem myszy można wybrać generuj zarówno dla klasy jak i dla interfejsu.

Po wybraniu Edit zobaczymy wygenerowana klasę.





## Interfejs - kod

**import** java.util.List;

**import** com.modeliosoft.modelio.javadesigner.annotations.objid;

*@objid ("4051cb69-1a00-4935-9d3b-4d7bc065efc5")*

**public** **interface** IWydarzenia {

*@objid ("9f66ea8b-0370-4df1-8ba5-59e79e65d292")*

**void** zapisz(**final** Wydarzenie wydarzenie);

*@objid ("7c19241c-87e1-4106-94e5-6a4ce365e5e2")*

Wydarzenie szukaj\_nazwa(**final** String nazwa);

*@objid ("c910c8ae-509b-4d90-8fc3-42fca7764888")*

Wydarzenie szukaj\_Id(**final** **int** id);

*@objid ("4c6d6722-6f15-4bac-9832-049cd7cd8f0d")*

**void** usun(**final** Wydarzenie wydarzenie);

*@objid ("837f2062-cee2-486e-a925-40ba36644669")*

List<Wydarzenie> pobierzListęNazw(); }

## Interfejs - realizacja

**import** java.util.List;

**import** com.modeliosoft.modelio.javadesigner.annotations.objid;

*@objid ("0125f466-1de8-4067-9fc2-3cd16bd7f384")*

**public** **class** Class **implements** IWydarzenia {

*@objid ("0c159b72-7088-4a60-a64c-c1eeccb8f1cf")*

**public** **void** zapisz(**final** Wydarzenie wydarzenie) {

}

*@objid ("5bee1f99-7f42-4b3a-b2bb-3e757db8ea53")*

**public** Wydarzenie szukaj\_nazwa(**final** String nazwa) {

*// TODO Auto-generated return*

**return** null;

}

*@objid ("80cfba07-ff70-4c84-849b-957392a9dd66")*

**public** Wydarzenie szukaj\_Id(**final** **int** id) {

*// TODO Auto-generated return*

**return** null;

}

*@objid ("88c573ad-96c5-4298-b3f2-1f59f8306090")*

**public** **void** usun(**final** Wydarzenie wydarzenie) {

}

*@objid ("1f0ee9b1-03ff-4d55-a62b-f97f80d5f3a0")*

**public** List<Wydarzenie> pobierzListęNazw() {

*// TODO Auto-generated return*

**return** null;

}