POLITECHNIKA ŚWIĘTOKRZYSKA

Wydział Elektrotechniki, Automatyki i Informatyki

Projekt: **Modelowanie i Analiza Systemów Informatycznych**	
Temat: **Modelowanie i analiza systemu informatycznego realizującego zamianę unitermu poziomej operacji zrównoleglania unitermów na pionową operację sekwencjowania unitermów**	
Maksymilian Sowula	Grupa: **1ID21A**

1. Najważniejsze części kodu źródłowego wraz z opisem

1.1. Metoda odpowiedzialna za wizualizację zrównoleglenia poziomego

Jest to metoda odpowiedzialna za graficzną reprezentację zrównoleglenia poziomego. Rysuje poziome linie oraz umieszcza wszystkie wyrażenia (expression) z listy uniterms, rozdzielone separatorem, w jednej linii. Jeśli lista jest pusta, wyświetla komunikat informujący o braku wybranych wyrażeń.

```
public drawParallelOperation() {
    const startX = 50;
    const startY = 100;
    if (this.uniterms.length === 0) {
      this.ctx.fillText("Nie wybrano wyrażeń", startX, startY);
      return;
    }
    this.ctx.beginPath();
    this.ctx.moveTo(startX, startY - 30);
    this.ctx.lineTo(startX + 150, startY - 30);
    this.ctx.stroke();
    this.ctx.beginPath();
    this.ctx.moveTo(startX, startY - 30);
    this.ctx.lineTo(startX, startY - 15);
    this.ctx.stroke();
    this.ctx.beginPath();
    this.ctx.moveTo(startX + 150, startY - 30);
    this.ctx.lineTo(startX + 150, startY - 15);
    this.ctx.stroke();
    const text = this.uniterms.map(u => u.expression).join(` ${this.separator} `);
    this.ctx.fillText(text, startX + 20, startY);
  }
```

Listing 1.1: Metoda drawParallelOperation.

1.2. Metoda odpowiedzialna za wizualizację sekwencjonowania pionowego

Jest to metoda służąca do wizualizacji sekwencjonowania pionowego. Rysuje pionowy przepływ (za pomocą krzywej Beziera) i wypisuje każde wyrażenie jedno pod drugim, oddzielając je separatorem. Przedstawia kolejność wykonywania wyrażeń.

```
public drawSequentialOperation() {
    const startX = 50;
    const startY = 50;
    if (this.uniterms.length === 0) {
      this.ctx.fillText("Nie wybrano wyrażeń", startX, startY);
      return;
    }
    this.ctx.beginPath();
    this.ctx.moveTo(startX, startY);
    this.ctx.bezierCurveTo(
      startX - 20, startY,
      startX - 20, startY + (this.uniterms.length * 30 + 20),
      startX, startY + (this.uniterms.length * 30 + 20)
    );
    this.ctx.stroke();
    this.uniterms.forEach((uniterm, index) => {
      this.ctx.fillText(uniterm.expression, startX + 15, startY + (index * 30) + 20);
      if (index < this.uniterms.length - 1) {
        this.ctx.fillText(this.separator, startX + 15, startY + (index * 30) + 35);
      }
    });
}
```

Listing 1.2: Metoda drawSequentialOperation.

1.3. Metoda odpowiedzialna za wizualizacje zamiany zrównoleglenia poziomego wybranego unitermu na sekwencjonowanie pionowe

Jest to metoda rysująca połączenie zrównoleglenia z sekwencjonowaniem – ilustruje zamianę jednego z elementów struktury równoległej na sekwencję. Działa w dwóch wariantach zależnie od podanego indeksu: pierwszy przypadek rysuje sekwencję z lewej strony, drugi – z

prawej. W obu przypadkach wyrażenia są połączone separatorem i umieszczone w ramach wizualizacji równoległej.

```
public drawMergedVisualization(index: number) {
    const ctx = index === 0 ? this.mergedCtx1 : this.mergedCtx2;
    if (!ctx) return;
    const canvas =
      index === 0
        ? this.mergedCanvas1Ref!.nativeElement
        : this.mergedCanvas2Ref!.nativeElement;
    ctx.clearRect(0, 0, canvas.width, canvas.height);
    ctx.font = '14px Arial';
    const startX = 80;
    const startY = 80;
    if (index === 0) {
      ctx.beginPath();
      ctx.moveTo(startX, startY - 30);
      ctx.lineTo(startX + 200, startY - 30);
      ctx.stroke();
      ctx.beginPath();
      ctx.moveTo(startX, startY - 30);
      ctx.lineTo(startX, startY - 15);
      ctx.stroke();
      ctx.beginPath();
      ctx.moveTo(startX + 200, startY - 30);
      ctx.lineTo(startX + 200, startY - 15);
      ctx.stroke();
      this.drawSequentialOperation(
        ctx,
        this.sequentialUniterms,
        startX + 30,
        startY - 10
      );
      ctx.fillText(this.separator, startX + 100, startY);
      if (this.parallelUniterms.length > 1) {
        ctx.fillText(this.parallelUniterms[1].expression, startX + 120, startY);
      }
    } else {
      ctx.beginPath();
      ctx.moveTo(startX, startY - 30);
```

```
      ctx.lineTo(startX + 200, startY - 30);
      ctx.stroke();
      ctx.beginPath();
      ctx.moveTo(startX, startY - 30);
      ctx.lineTo(startX, startY - 15);
      ctx.stroke();
      ctx.beginPath();
      ctx.moveTo(startX + 200, startY - 30);
      ctx.lineTo(startX + 200, startY - 15);
      ctx.stroke();
      if (this.parallelUniterms.length > 0) {
        ctx.fillText(this.parallelUniterms[0].expression, startX + 20, startY);
      }
      ctx.fillText(this.separator, startX + 80, startY);
      this.drawSequentialOperation(
        ctx,
        this.sequentialUniterms,
        startX + 100,
        startY - 10
      );
    }
  }
```

Listing 1.3: Metoda drawMergedVisualisation.

1.4. Metoda odpowiedzialna za udostępnienie punktu końcowego do pobrania unitermów z bazy danych

Jest to metoda kontrolera udostępniająca punkt końcowy HTTP GET, który zwraca listę wszystkich unitermów z bazy danych. Komunikuje się z warstwą serwisową (unitermService.findAll()).

```
@GetMapping
public List<Uniterm> getAllUniterms() {
   return unitermService.findAll();
}
```

Listing 1.4: Metoda getAllUniterms.

1.5. Metoda odpowiedzialna za udostępnienie punktu końcowego do zapisania unitermu do bazy danych

Jest to metoda kontrolera obsługująca zapytanie HTTP POST, służące do dodawania nowego unitermu do bazy. Dane są przekazywane w ciele żądania i zapisywane poprzez unitermService.save(uniterm).

```
@PostMapping
public Uniterm createUniterm(@RequestBody Uniterm uniterm) {
    return unitermService.save(uniterm);
}
```

Listing 1.5: Metoda createUniterm.

1.6. Klasa definiująca encję Uniterm

Jest to klasa, która definiuje encję JPA, która zawiera pola: id, name, expression (wymagane) oraz description. Dodatkowo, dzięki metodzie @PrePersist i @PreUpdate, jeśli pole name nie jest ustawione, automatycznie przyjmuje wartość pola expression.

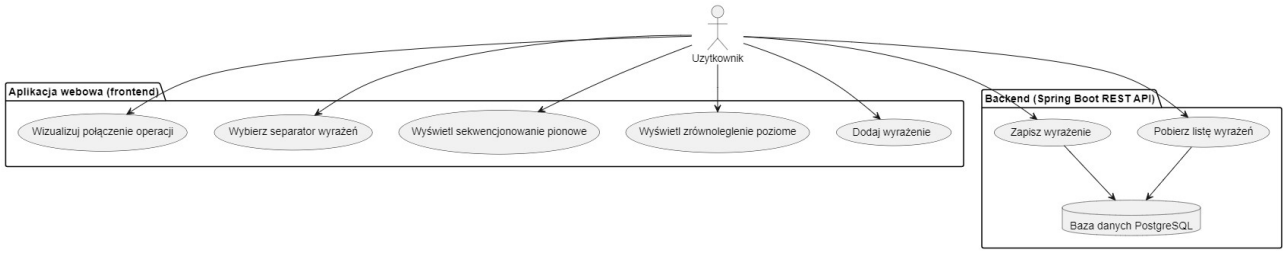
```
package com.uniterm.backend.model;

import jakarta.persistence.*;
import lombok.*;

@Entity
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Builder
@Table(name = "uniterms")
public class Uniterm {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
```

```
    @Column
    private String name;

    @Column(nullable = false)
    private String expression;

    private String description;

    @PrePersist
    @PreUpdate
    private void ensureNameIsSet() {
        if (name == null || name.isEmpty()) {
            name = expression;
        }
    }
}
```

Listing 1.6: Klasa definiująca encję Uniterm.

2. Diagram przypadków użycia

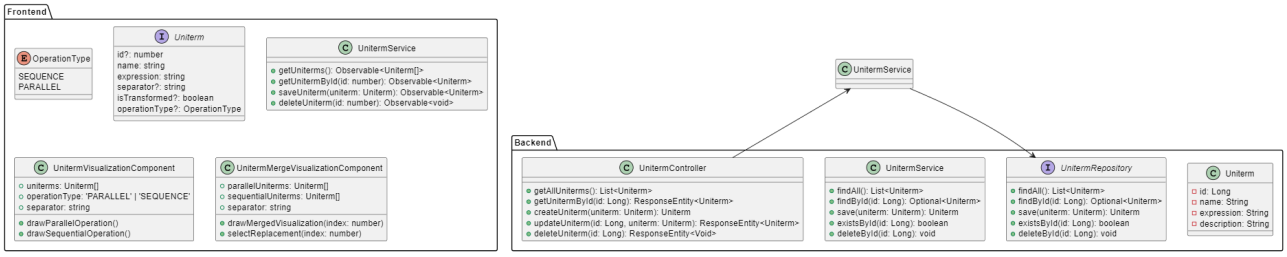
Poniższy rysunek prezentuje diagram przypadków użycia w systemie z podziałem na komunikację użytkownika z warstwą prezentacji (aplikacją webową) oraz warstwą logiki biznesowej (backend).



Rysunek 2.1. Diagram przypadków użycia systemu.

3. Diagram klas

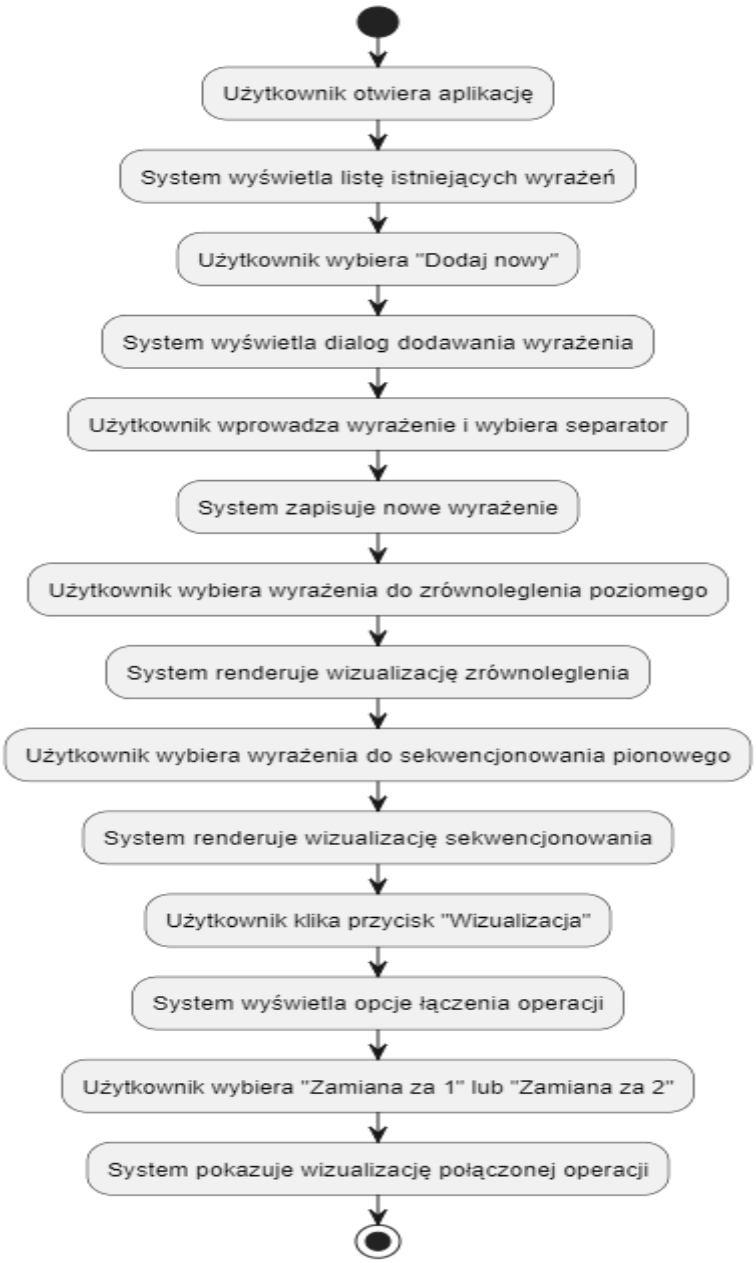
Poniższy rysunek prezentuje diagram klas systemu z podziałem na warstwę prezentacji (frontend) oraz warstwę logiki biznesowej (backend).



Rysunek 3.1. Diagram klas systemu.

4. Diagram aktywności

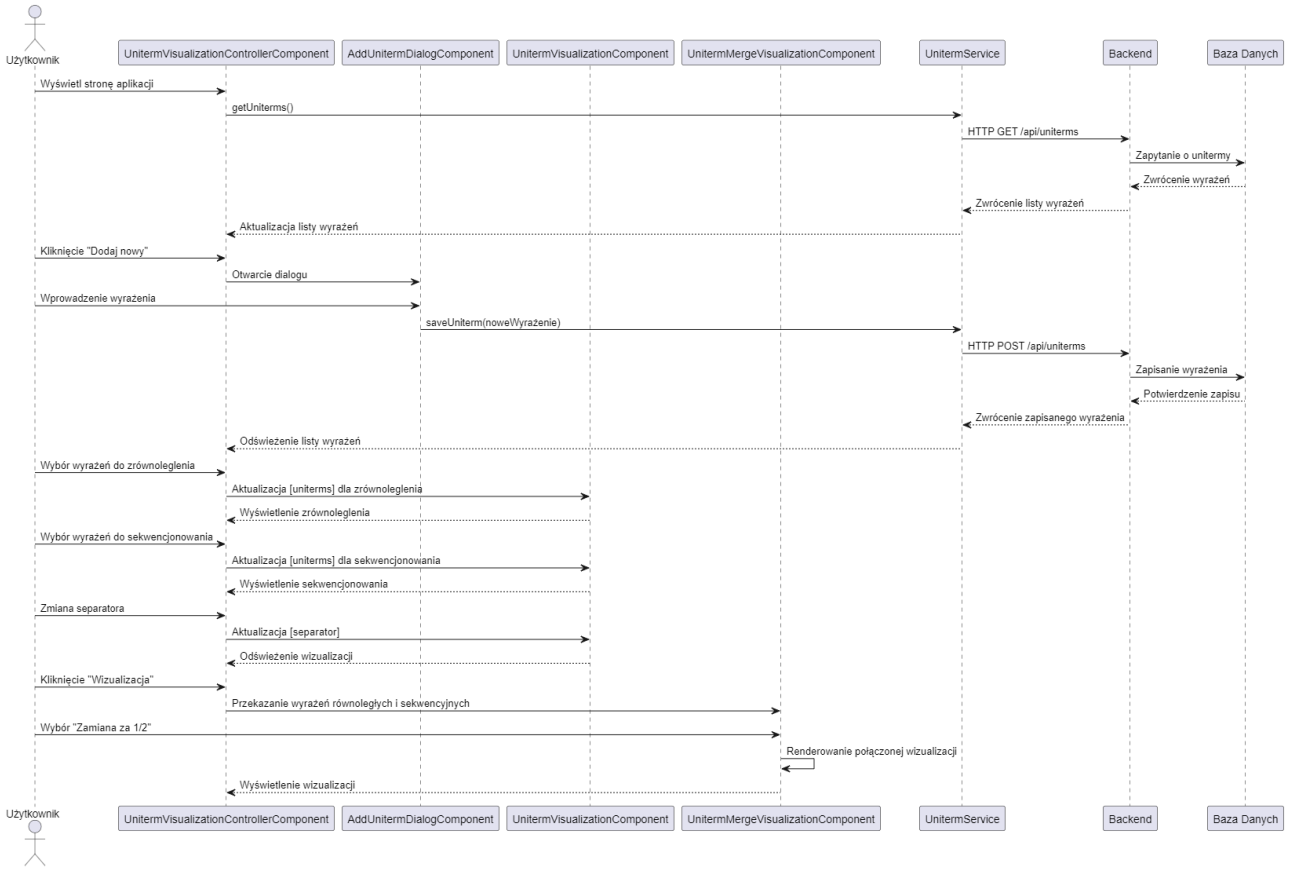
Poniższy rysunek prezentuje diagram aktywności systemu.



Rysunek 4.1. Diagram aktywności systemu.

5. Diagram sekwencji

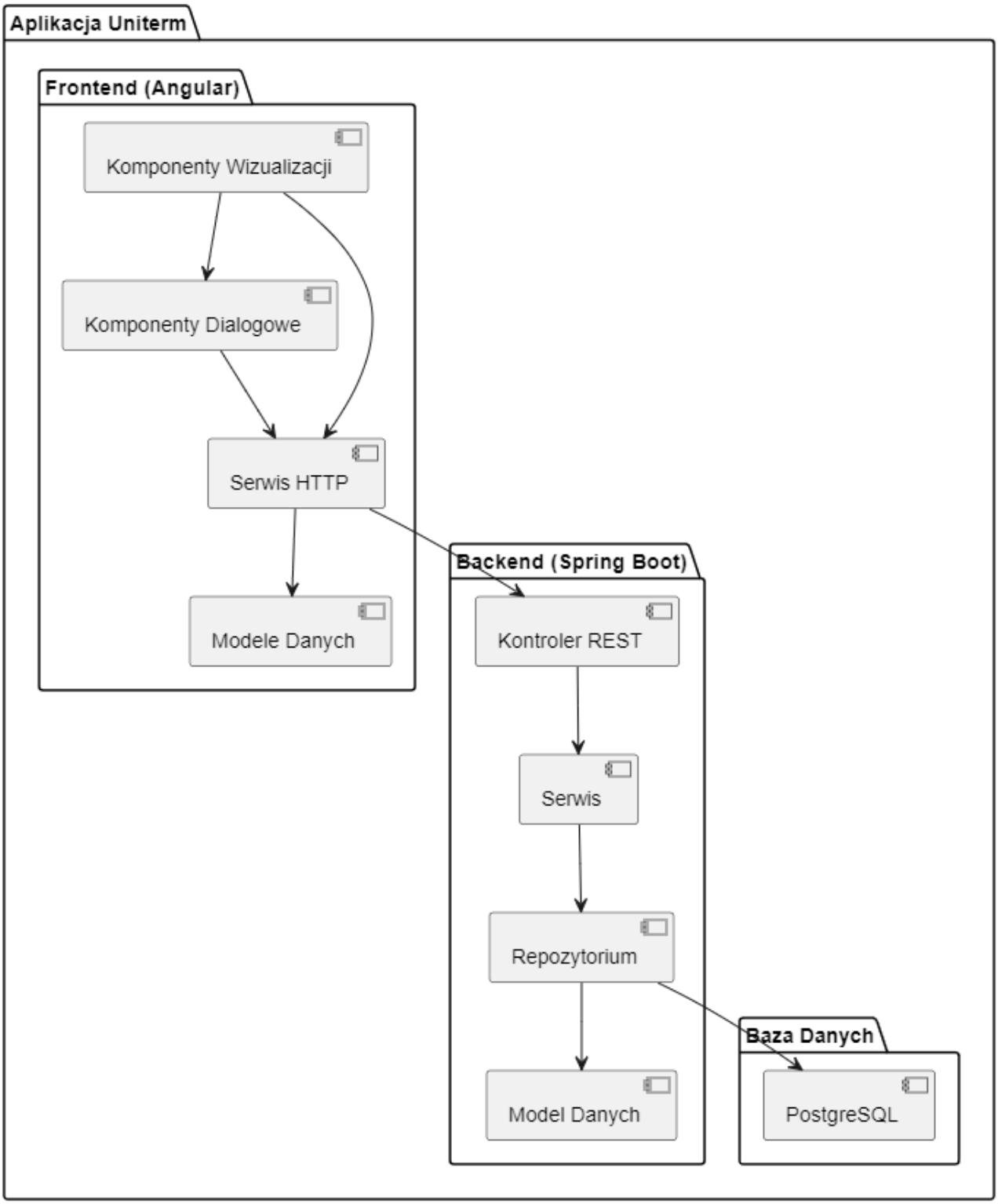
Poniższy rysunek prezentuje diagram sekwencji systemu.



Rysunek 5.1. Diagram sekwencji systemu.

6. Diagram warstw

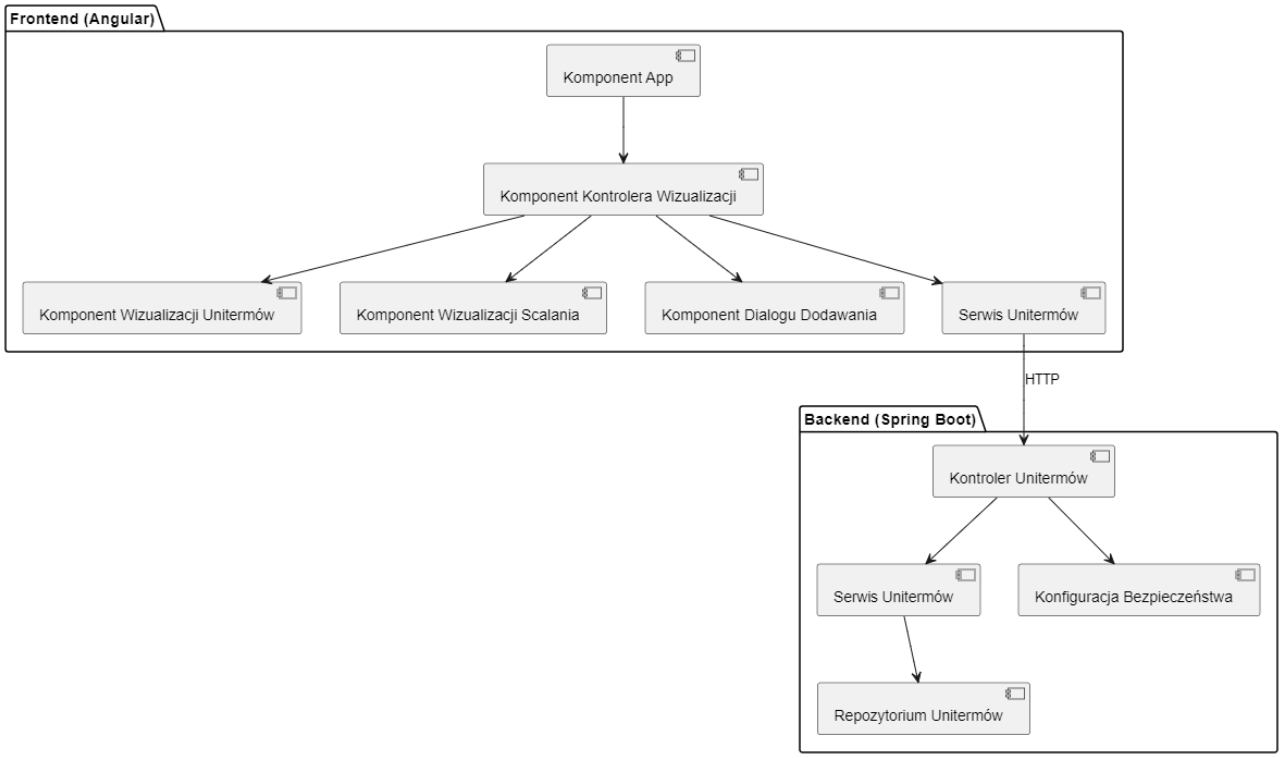
Poniższy rysunek prezentuje diagram warstw systemu.



Rysunek 6.1. Diagram warstw systemu.

7. Diagram komponentów

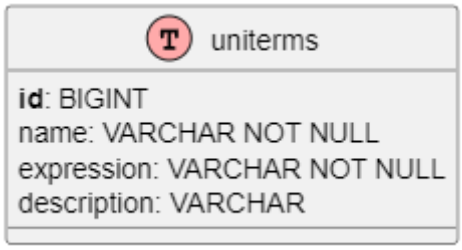
Poniższy rysunek prezentuje diagram komponentów systemu.



Rysunek 7.1. Diagram komponentów systemu.

8. Struktura bazy danych

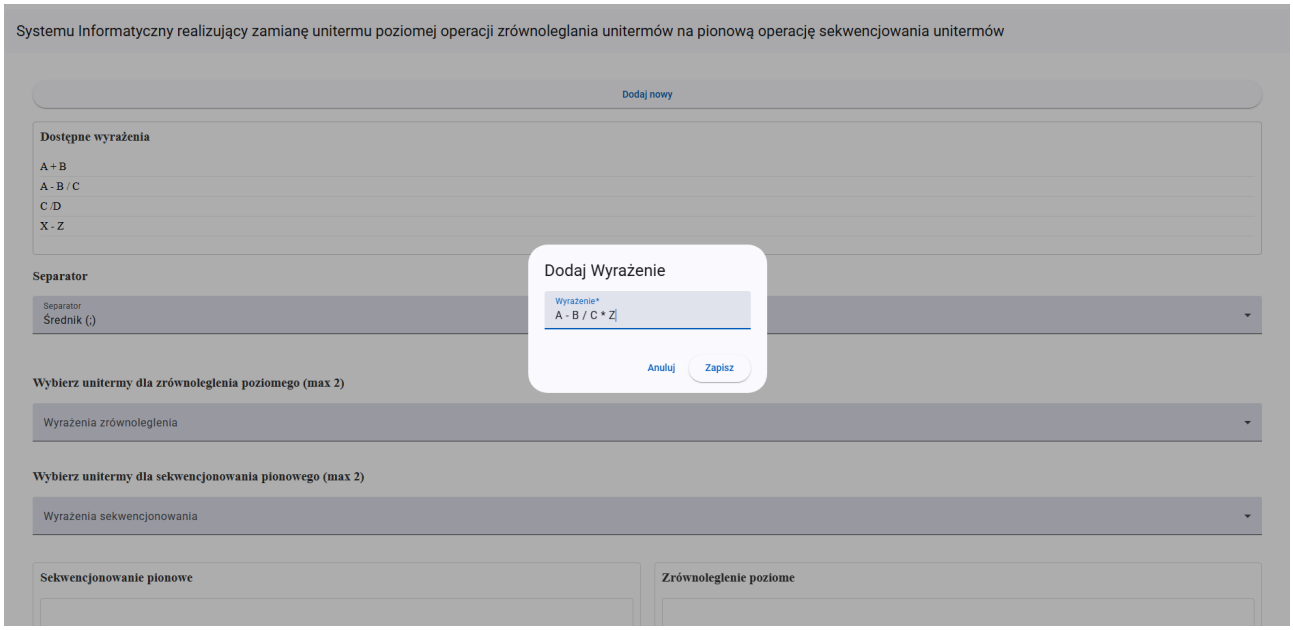
Poniższy rysunek prezentuje strukturę relazyjnej bazy danych systemu.



Rysunek 8.1. Diagram relacji encji w relacyjnej bazie danych systemu.

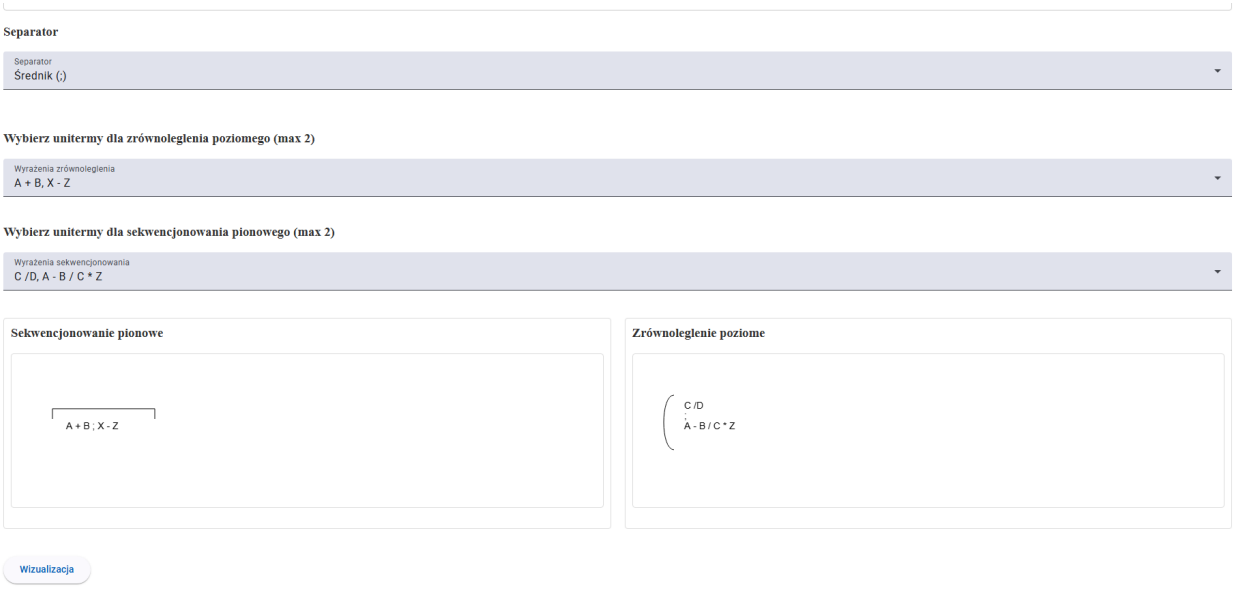
9. Zdjęcia z działania aplikacji

Na poniższym rysunku przedstawiono okno dialogowe dodawania nowego wyrażenia do bazy systemu.



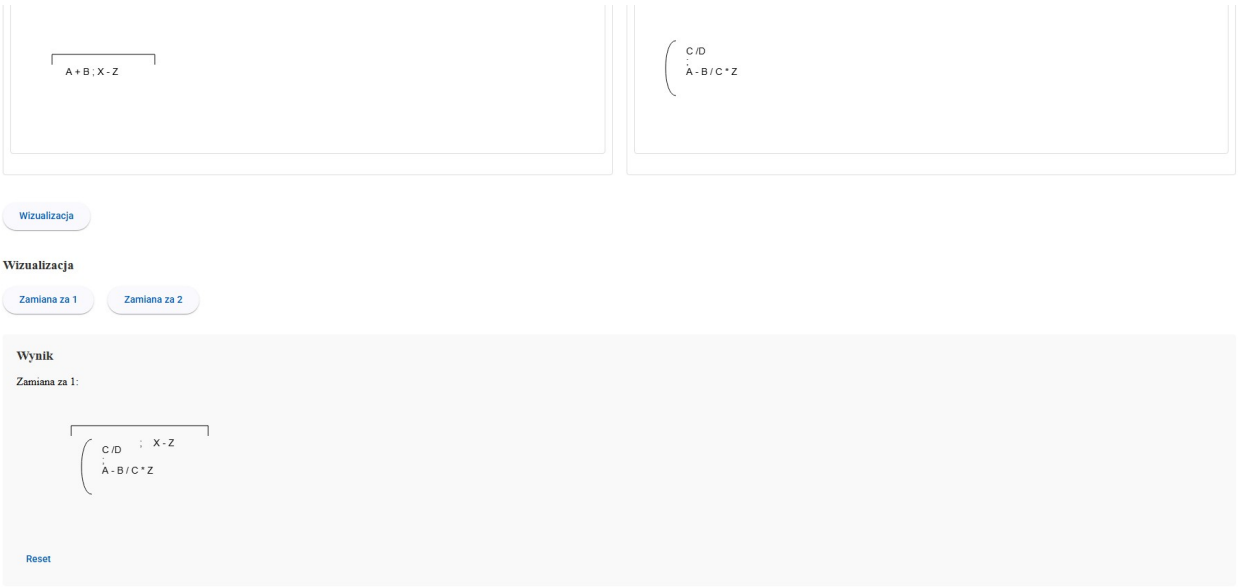
Rysunek 9.1. Dodawanie nowego wyrażenia do systemu.

Na poniższym rysunku przedstawiono wizualizację wybranych wyrażeń w postaci sekwencjonowania pionowego i zrównoleglenia poziomego.



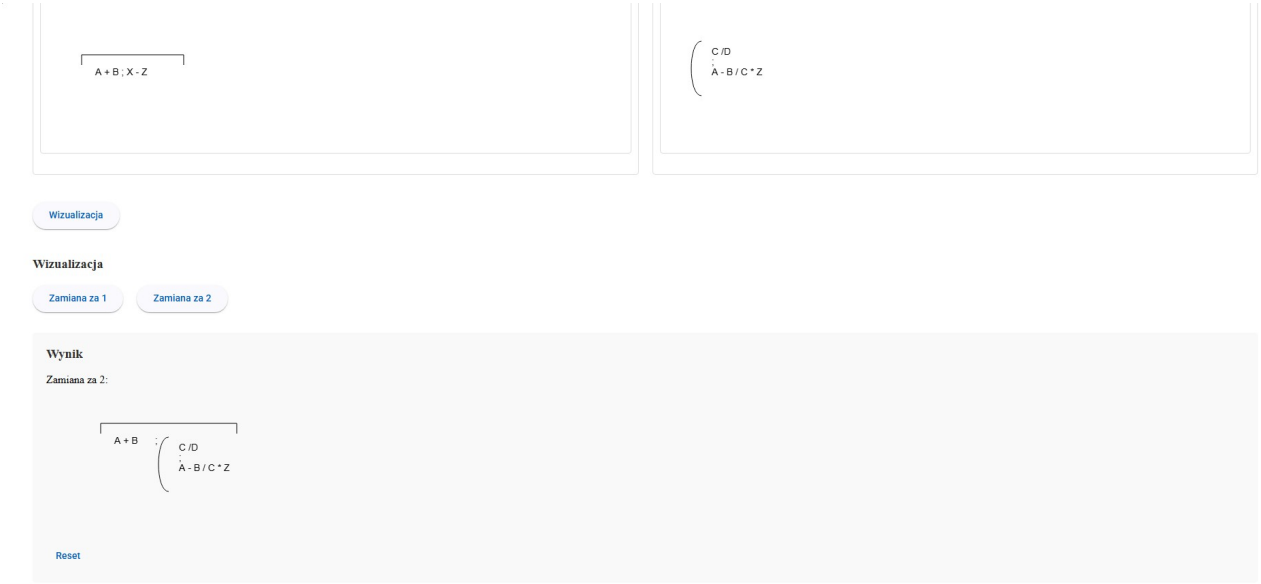
Rysunek 9.2. Wybranie wyrażeń, separatora oraz wyświetlenie operacji.

Poniższy rysunek przedstawia wynik zamiany zrównoleglenia poziomego 1 unitermu na sekwencjonowanie pionowe.



Rysunek 9.3. Zamiana 1 unitermu na zrównoleglenie poziome.

Poniższy rysunek przedstawia wynik zamiany zrównoleglenia poziomego 2 unitermu na sekwencjonowanie pionowe.



Rysunek 9.4. Zamiana 2 unitermu na sekwencjonowanie pionowe.