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

Poniższa klasa jest definicją modelu rekordu Unitermu przechowywanego w encji uniterms w bazie danych. Wszelkie sprawy związane z komunikacją z bazą danych jak i wykonywanych na niej operacji wykonywane są za pomocą Mapowania Obiektowo-Relacyjnego. Model unitermu składa się z identyfikatora, nazwy, pierwszego wyrażenia, drugiego wyrażania,

```java
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;

    @Column(nullable = false)
    private String secondExpression;

    @Column
    private String separator;

    @Column
    private String sequentialSeparator;
```

```
    @PrePersist
    @PreUpdate
    private void ensureNameIsSet() {
        if (name == null || name.isEmpty()) {
                name = expression + (secondExpression != null ? " + " +
secondExpression : "");
        }
    }
}
```

Listing 1.1. Klasa Uniterm opisująca model rekordu encji unitermu.

Poniższy listing reprezentuje kontroller realizujący operacje GET oraz POST służące do pobierania i zapisywania rekordów z unitermami zgodnie z architekturą REST.

```
package com.uniterm.backend.controller;

import com.uniterm.backend.model.Uniterm;
import com.uniterm.backend.services.UnitermService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import java.util.List;

@RestController
@RequestMapping("/api/uniterms")
public class UnitermController {
    @Autowired
    private UnitermService unitermService;
    @GetMapping
    public List<Uniterm> getAllUniterms() {
        return unitermService.findAll();
    }
    @GetMapping("/{id}")
    public ResponseEntity<Uniterm> getUnitermById(@PathVariable Long id) {
        return unitermService.findById(id)
                .map(ResponseEntity::ok)
                .orElse(ResponseEntity.notFound().build());
    }
```

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

Listing 1.2. Klasa UnitermController obsługująca żądania GET oraz POST obsługi pobierania lub dodawania nowych rekordów unitermów.

Poniższy listing reprezentuje klasę serwisu obsługującą logikę biznesową komunikacji z bazą danych celem zapisywania lub odczytywania rekordów unitermów.

```
package com.uniterm.backend.services;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.uniterm.backend.model.Uniterm;
import com.uniterm.backend.repositories.UnitermRepository;
import java.util.List;
import java.util.Optional;

@Service
public class UnitermService {
    @Autowired
    private UnitermRepository unitermRepository;
    public List<Uniterm> findAll() {
        return unitermRepository.findAll();
    }
    public Optional<Uniterm> findById(Long id) {
        return unitermRepository.findById(id);
    }
    public Uniterm save(Uniterm uniterm) {
        return unitermRepository.save(uniterm);
    }
    public boolean existsById(Long id) {
        return unitermRepository.existsById(id);
    }
}
```

Listing 1.3. Klasa UnitermService obsługująca logikę biznesową zapisywania i odczytywania rekordów unitermów.

Poniższy listing przedstawia kod odpowiadający za widok okna dialogowego dodawania nowego rekordu unitermu po stronie warstwy prezentacji aplikacji.

```
<h2 mat-dialog-title>{{ isEditMode ? 'Edytuj' : 'Dodaj' }} Wyrażenie</h2>
<mat-dialog-content>
  <form [formGroup]="unitermForm" class="uniterm-form">
    <mat-form-field appearance="fill" class="full-width">
      <mat-label>Nazwa (opcjonalnie)</mat-label>
       <input matInput formControlName="name" placeholder="Opcjonalna nazwa dla
wyrażenia">
    </mat-form-field>
    <div class="section-header">
      <h3>Zrównoleglenie poziome</h3>
    </div>
    <mat-form-field appearance="fill" class="full-width">
      <mat-label>Pierwsza część wyrażenia</mat-label>
         <input matInput formControlName="parallelPart1" placeholder="Pierwsza
część wyrażenia">
    <mat-error *ngIf="unitermForm.get('parallelPart1')?.hasError('required')">
        Wymagane pole
      </mat-error>
    </mat-form-field>
    <mat-form-field appearance="fill" class="full-width">
      <mat-label>Druga część wyrażenia</mat-label>
       <input matInput formControlName="parallelPart2" placeholder="Druga część
wyrażenia">
<mat-error *ngIf="unitermForm.get('parallelPart2')?.hasError('required')">
        Wymagane pole
      </mat-error>
    </mat-form-field>
    <mat-form-field appearance="fill" class="full-width">
      <mat-label>Separator dla wyrażenia poziomego</mat-label>
      <mat-select formControlName="parallelSeparator">
                      <mat-option  *ngFor="let  option  of  separatorOptions"
[value]="option.value">
          {{ option.label }}
        </mat-option>
      </mat-select>
    </mat-form-field>
```

```
    <div class="section-header">
      <h3>Sekwencjonowanie pionowe</h3>
    </div>
    <mat-form-field appearance="fill" class="full-width">
      <mat-label>Pierwsza część wyrażenia</mat-label>
       <input matInput formControlName="sequentialPart1" placeholder="Pierwsza
część wyrażenia">
                                                                    <mat-error
*ngIf="unitermForm.get('sequentialPart1')?.hasError('required')">
        Wymagane pole
      </mat-error>
    </mat-form-field>
    <mat-form-field appearance="fill" class="full-width">
      <mat-label>Druga część wyrażenia</mat-label>
         <input matInput formControlName="sequentialPart2" placeholder="Druga
część wyrażenia">
                                                                    <mat-error
*ngIf="unitermForm.get('sequentialPart2')?.hasError('required')">
        Wymagane pole
      </mat-error>
    </mat-form-field>
    <mat-form-field appearance="fill" class="full-width">
      <mat-label>Separator dla wyrażenia pionowego</mat-label>
      <mat-select formControlName="sequentialSeparator">
                    <mat-option *ngFor="let option of separatorOptions"
[value]="option.value">
          {{ option.label }}
        </mat-option>
      </mat-select>
    </mat-form-field>
  </form>
</mat-dialog-content>
<mat-dialog-actions align="end">
  <button mat-button (click)="onCancel()">Anuluj</button>
   <button mat-raised-button color="primary" [disabled]="unitermForm.invalid"
(click)="onSave()">
    Zapisz
  </button>
</mat-dialog-actions>
```

Listing 1.4. Kod widoku okna dialogowego dodawania nowego unitermu.

Poniższy listing kod odpowiadający za okno dialogowe dodawania nowego rekordu unitermu po stronie warstwy prezentacji aplikacji.

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable, catchError, tap } from 'rxjs';
import { Uniterm } from '../models/uniterm.model';

@Injectable({
  providedIn: 'root',
})
export class UnitermService {
  private apiUrl = 'http://localhost:8080/api/uniterms';
  constructor(private http: HttpClient) {}
  getUniterms(): Observable<Uniterm[]> {
    console.log('Fetching uniterms from:', this.apiUrl);
    return this.http.get<Uniterm[]>(this.apiUrl).pipe(
      tap((data) => console.log('Fetched uniterms:', data)),
      catchError((error) => {
        console.error('Error fetching uniterms:', error);
        throw error;
      })
    );
  }
  getUnitermById(id: number): Observable<Uniterm> {
    return this.http.get<Uniterm>(`${this.apiUrl}/${id}`).pipe(
      tap((data) => console.log(`Fetched uniterm id=${id}:`, data)),
      catchError((error) => {
        console.error(`Error fetching uniterm id=${id}:`, error);
        throw error;
      })
    );
  }
  saveUniterm(uniterm: Uniterm): Observable<Uniterm> {
    console.log('Saving uniterm:', uniterm);
    if (uniterm.id) {
      return this.http
        .put<Uniterm>(`${this.apiUrl}/${uniterm.id}`, uniterm)
```

```
        .pipe(
          tap((data) => console.log('Updated uniterm:', data)),
          catchError((error) => {
            console.error('Error updating uniterm:', error);
            throw error;
          })
        );
    } else {
      return this.http.post<Uniterm>(this.apiUrl, uniterm).pipe(
        tap((data) => console.log('Created uniterm:', data)),
        catchError((error) => {
          console.error('Error creating uniterm:', error);
          throw error;
        })
      );
    }
  }

  deleteUnitermById(id: number): Observable<void> {
    return this.http.delete<void>(`${this.apiUrl}/${id}`);
  }
}
```

Listing 1.5. Kod serwisu odpowiedzialnego za komunikację warstwy prezentacji z utworzonym API.

Poniższa metoda przedstawia funkcję rysującą operacje zrównoleglenia poziomego według danych zawartych w obiekcie unitermu wybranego z listy unitermów.

```
private drawParallelOperation(): void {
    if (!this.ctx) return;
    let contentWidth = 0;
    this.uniterms.forEach((uniterm, index) => {
      let text = uniterm.expression || uniterm.name || '';
      if (!text) text = `Item ${index + 1}`;
      const textWidth = this.ctx!.measureText(text).width;
      contentWidth += textWidth;
  if (index < this.uniterms.length - 1) {
        const separatorWidth = this.ctx!.measureText(this.separator).width;
contentWidth += separatorWidth + 30;
}
```

```
    });
    contentWidth += 60;
    const startX = 50;
    const startY = 90;
    const lineWidth = Math.min(contentWidth, 350);
    this.ctx.beginPath();
    this.ctx.moveTo(startX, startY - 40);
    this.ctx.lineTo(startX + lineWidth, startY - 40);
    this.ctx.lineWidth = 2;
    this.ctx.stroke();
    this.ctx.beginPath();
    this.ctx.moveTo(startX, startY - 40);
    this.ctx.lineTo(startX, startY - 20);
    this.ctx.stroke();
    this.ctx.beginPath();
    this.ctx.moveTo(startX + lineWidth, startY - 40);
    this.ctx.lineTo(startX + lineWidth, startY - 20);
    this.ctx.stroke();
    this.ctx.lineWidth = 1;
    let currentX = startX + 20;
    this.uniterms.forEach((uniterm, index) => {
      let text = uniterm.expression || uniterm.name || '';
      if (!text) text = `Item ${index + 1}`;
      this.ctx!.fillStyle = '#000000';
      this.ctx!.fillText(text, currentX, startY + 5);
      const textWidth = this.ctx!.measureText(text).width;
      currentX += textWidth + 20;
      if (index < this.uniterms.length - 1) {
        this.ctx!.fillStyle = '#555555';
        this.ctx!.fillText(this.separator, currentX - 10, startY + 5);
        currentX += 15;
      }
    });
  }
```

Listing 1.6. Metoda odpowiedzialna za zrenderowania operacji zrównoleglenia poziomego.

Poniższy listing przedstawia metodę odpowiedzialną za zrenderowanie operacji sekwencjonowania pionowego według wybranego z listy obiektu unitermu.

```
  private drawSequentialOperation(): void {
    if (!this.ctx) return;
    const startX = 120;
    const startY = 40;
    const itemHeight = 40;
    this.ctx.lineWidth = 2;
    this.ctx.beginPath();
    this.ctx.moveTo(startX - 30, startY);
    this.ctx.bezierCurveTo(
      startX - 60,
      startY,
      startX - 60,
      startY + this.uniterms.length * itemHeight,
      startX - 30,
      startY + this.uniterms.length * itemHeight
    );
    this.ctx.stroke();
    this.ctx.lineWidth = 1;
    let currentY = startY;
    this.uniterms.forEach((uniterm, index) => {
      let text = uniterm.expression || uniterm.name || '';
      if (!text) text = `Item ${index + 1}`;
      this.ctx!.fillStyle = '#000000';
      this.ctx!.fillText(text, startX, currentY + 15);
      currentY += itemHeight;
      if (index < this.uniterms.length - 1) {
        this.ctx!.fillStyle = '#555555';
        this.ctx!.fillText(this.separator, startX, currentY - 10);
      }
    });
  }
```

Listing 1.7. Metoda odpowiedzialna za zrenderowania operacji sekwencjonowania pionowego.

Poniższa metoda odpowiedzialna jest za obliczenie szerokości renderowanego symbolu zrównoleglenia poziomego tak aby wyrażenia się w nim zmieściły i jego rozmiary były dynamiczne w zależności od przekazanych parametrów.

```
  private calculateBarLength(ctx: CanvasRenderingContext2D): number {
    let totalWidth = 0;
    this.parallelUniterms.forEach((uniterm, index) => {
      const text = this.processText(uniterm.expression || uniterm.name || '');
      totalWidth += ctx.measureText(text).width;
      if (index < this.parallelUniterms.length - 1) {
        const separator = this.getParallelSeparator();
        totalWidth +=
          ctx.measureText(separator).width + this.SEPARATOR_SPACING * 2;
      }
    });
    totalWidth += 80;
    return Math.max(
      this.MIN_BAR_LENGTH,
      Math.min(this.MAX_BAR_LENGTH, totalWidth)
    );
  }
```

Listing 1.8. Metoda odpowiedzialna za obliczenie szerokości elementu zrównoleglenia poziomego.

Poniższa metoda przedstawia operację zamiany zrównoleglenia poziomego na sekwencjonowanie pionowe dla 1 elementu.

```
private drawOption1() {
    if (!this.mergedCtx1) return;
    const ctx = this.mergedCtx1;
    const barLength = this.calculateBarLength(ctx);
    ctx.beginPath();
    ctx.moveTo(50, 30);
    ctx.lineTo(50 + barLength, 30);
    ctx.lineWidth = 2;
    ctx.stroke();
    ctx.beginPath();
    ctx.moveTo(50, 30);
    ctx.lineTo(50, 45);
    ctx.stroke();
    ctx.beginPath();
    ctx.moveTo(50 + barLength, 30);
    ctx.lineTo(50 + barLength, 45);
    ctx.stroke();
    ctx.lineWidth = 1;
```

```
    const sequentialBracketX = 90;
    const bracketHeight = Math.max(35,
      this.sequentialUniterms.length * this.ELEMENT_SPACING
    );
            this.drawSequentialBracket(ctx,   sequentialBracketX,   50,   100,
bracketHeight);
    let y = 65;
    this.sequentialUniterms.forEach((uniterm, idx) => {
      let text = uniterm.expression || uniterm.name || `Item ${idx + 1}`;
      text = this.processText(text);
      ctx.fillStyle = '#000000';
      ctx.fillText(text, sequentialBracketX + 15, y);
      if (idx < this.sequentialUniterms.length - 1) {
        const sequentialSeparator = this.getSequentialSeparator();
        ctx.fillStyle = '#0000FF';
        ctx.fillText(sequentialSeparator, sequentialBracketX + 15, y + 20);
        ctx.fillStyle = '#000000';
        y += this.ELEMENT_SPACING;
      }
    });
    if (this.parallelUniterms.length > 1) {
      const sequentialWidth = this.getMaxSequentialTextWidth(ctx);
      const parallelSeparator = this.getParallelSeparator();
      ctx.fillStyle = '#0000FF';
      ctx.fillText(
        parallelSeparator, sequentialBracketX + sequentialWidth + 30, 70
      );
      const text = this.processText(
        this.parallelUniterms[1]?.expression ||
          this.parallelUniterms[1]?.name ||
          ''
      );
      if (text) {
        ctx.fillStyle = '#000000';
        ctx.fillText(text, sequentialBracketX + sequentialWidth + 50, 70);
      }
    }
  }
```

Listing 1.9. Metoda odpowiedzialna za wyrenderowanie operacji zamiany zrównoleglenia poziomego na sekwencjonowanie pionowe za 1 element.

Poniższa metoda przedstawia operację zamiany zrównoleglenia poziomego na sekwencjonowanie pionowe dla 2 elementu.

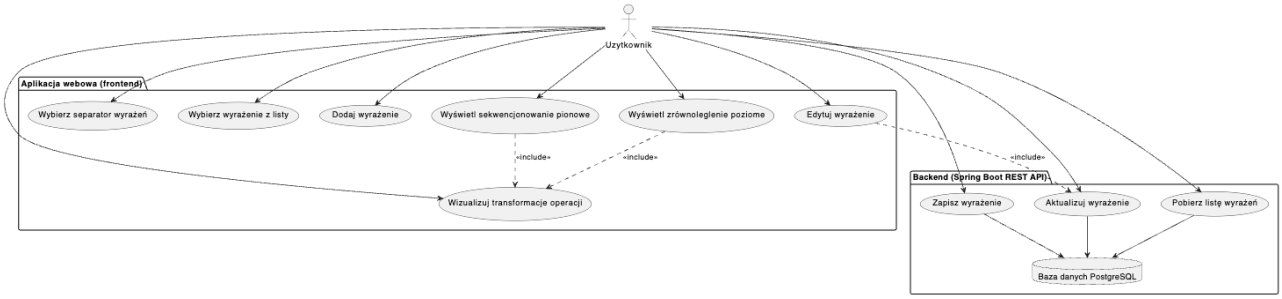
```
  private drawOption2() {
    if (!this.mergedCtx2) return;
    const ctx = this.mergedCtx2;
    const barLength = this.calculateBarLength(ctx);
    ctx.beginPath();
    ctx.moveTo(50, 30);
    ctx.lineTo(50 + barLength, 30);
    ctx.lineWidth = 2;
    ctx.stroke();
    ctx.beginPath();
    ctx.moveTo(50, 30);
    ctx.lineTo(50, 45);
    ctx.stroke();
    ctx.beginPath();
    ctx.moveTo(50 + barLength, 30);
    ctx.lineTo(50 + barLength, 45);
    ctx.stroke();
    ctx.lineWidth = 1;
    if (this.parallelUniterms.length > 0) {
      const text = this.processText(this.parallelUniterms[0]?.expression ||
          this.parallelUniterms[0]?.name ||
          ''
      );
      if (text) {
        ctx.fillStyle = '#000000';
        ctx.fillText(text, 70, 70);
        const firstElementWidth = ctx.measureText(text).width;
        const parallelSeparator = this.getParallelSeparator();
        ctx.fillStyle = '#0000FF';
        ctx.fillText(parallelSeparator, 85 + firstElementWidth, 70);
      } else {
        ctx.fillStyle = '#0000FF';
        const parallelSeparator = this.getParallelSeparator();
        ctx.fillText(parallelSeparator, 100, 70);
      }
    }
    let sequentialX = 150;
    if (this.parallelUniterms.length > 0) {
```

```
        const text = this.processText(this.parallelUniterms[0]?.expression ||
'');
      if (text) {
        const textWidth = ctx.measureText(text).width;
        const separator = this.getParallelSeparator();
        const separatorWidth = ctx.measureText(separator).width;
                    sequentialX = 100 + textWidth + separatorWidth +
this.SEPARATOR_SPACING;
      }
    }
    const bracketHeight = Math.max(
      35,
      this.sequentialUniterms.length * this.ELEMENT_SPACING
    );
    this.drawSequentialBracket(ctx, sequentialX, 50, 100, bracketHeight);
    let y = 65;
    this.sequentialUniterms.forEach((uniterm, idx) => {
      const text = this.processText(
        uniterm.expression || uniterm.name || `Item ${idx + 1}`
      );
      ctx.fillStyle = '#000000';
      ctx.fillText(text, sequentialX + 20, y);
      if (idx < this.sequentialUniterms.length - 1) {
        const sequentialSeparator = this.getSequentialSeparator();
        ctx.fillStyle = '#0000FF';
        ctx.fillText(sequentialSeparator, sequentialX + 20, y + 20);
        ctx.fillStyle = '#000000';
        y += this.ELEMENT_SPACING;
      }
    });
  }
```

Listing 1.10. Metoda odpowiedzialna za wyrenderowanie operacji zamiany zrównoleglenia poziomego na sekwencjonowanie pionowe za 2 element.

2. Diagram przypadków użycia

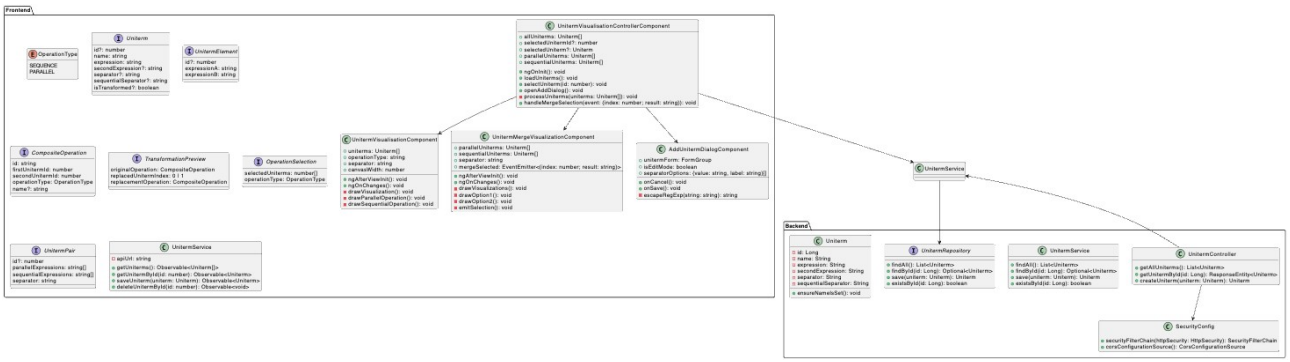
Poniższy rysunek prezentuje diagram przypadków użycia w systemie z podziałem na komunikację użytkownika z warstwą prezentacji (aplikacją webową) oraz warstwą logiki biznesowej (backend).



Rysunek 2.1. Diagram przypadków użycia systemu.

3. Diagram klas

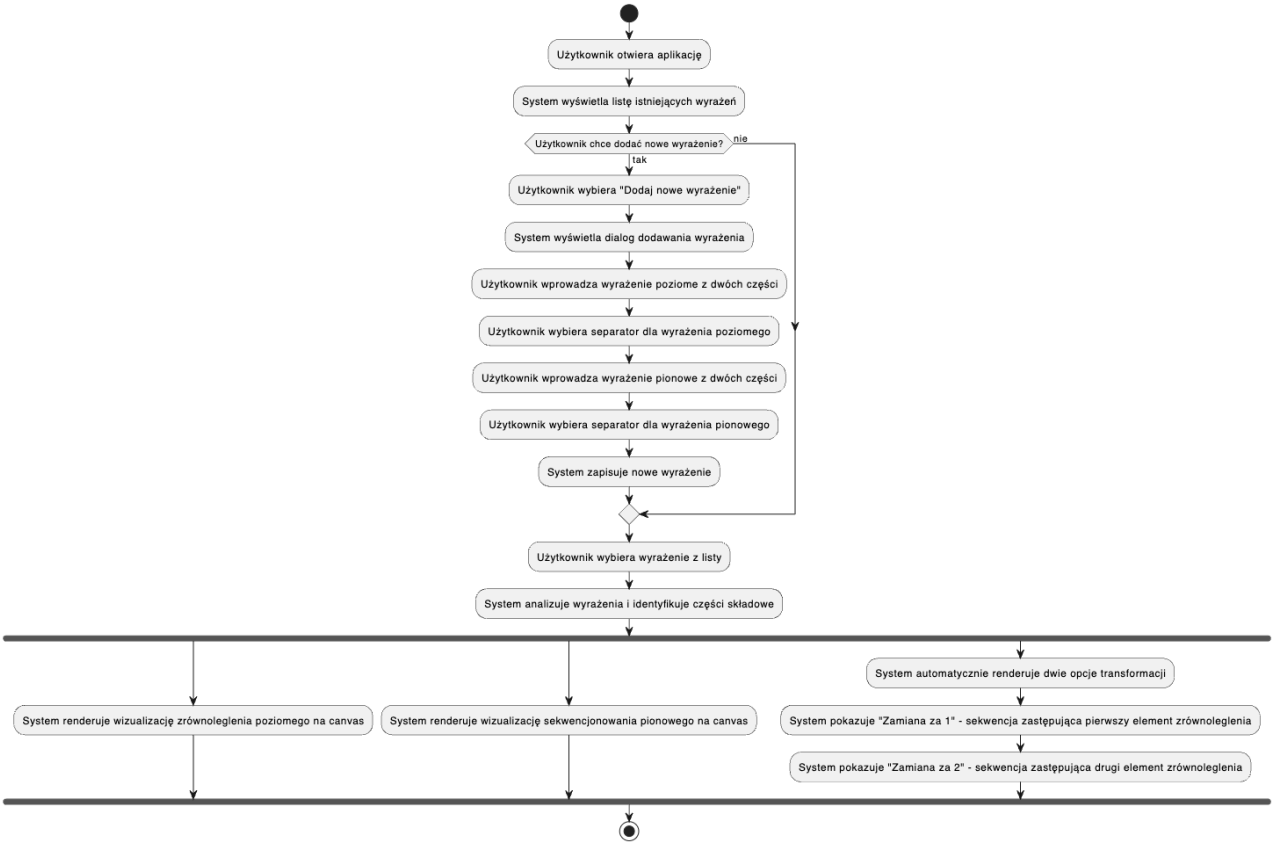
Poniższy rysunek prezentuje diagram klas systemu z podziałem na warstwę prezentacji (frontend) oraz warstwę logiki biznesowej (backend).



Rysunek 3.1. Diagram klas systemu.

4. Diagram aktywności

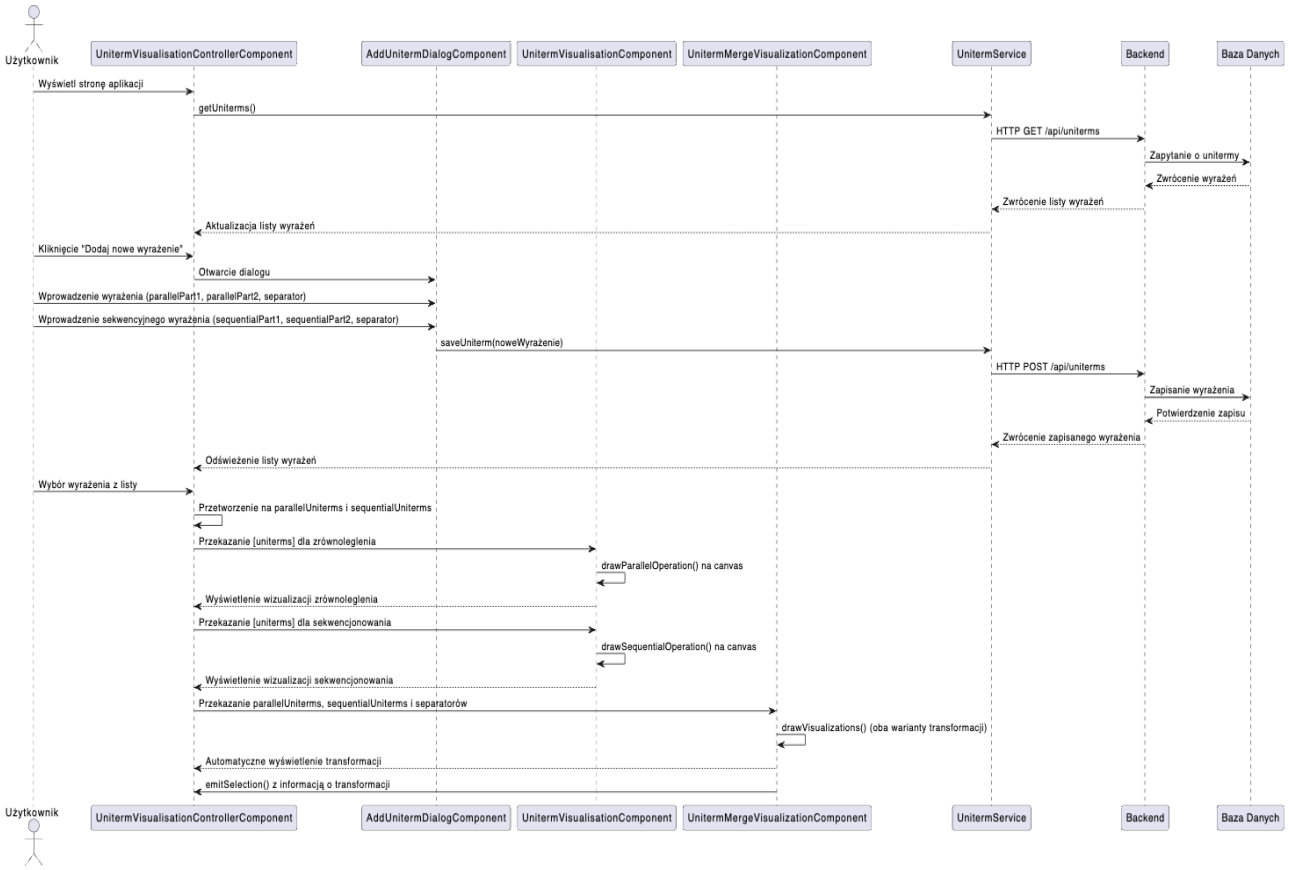
Poniższy rysunek prezentuje diagram aktywności systemu.



Rysunek 4.1. Diagram aktywności systemu.

5. Diagram sekwencji

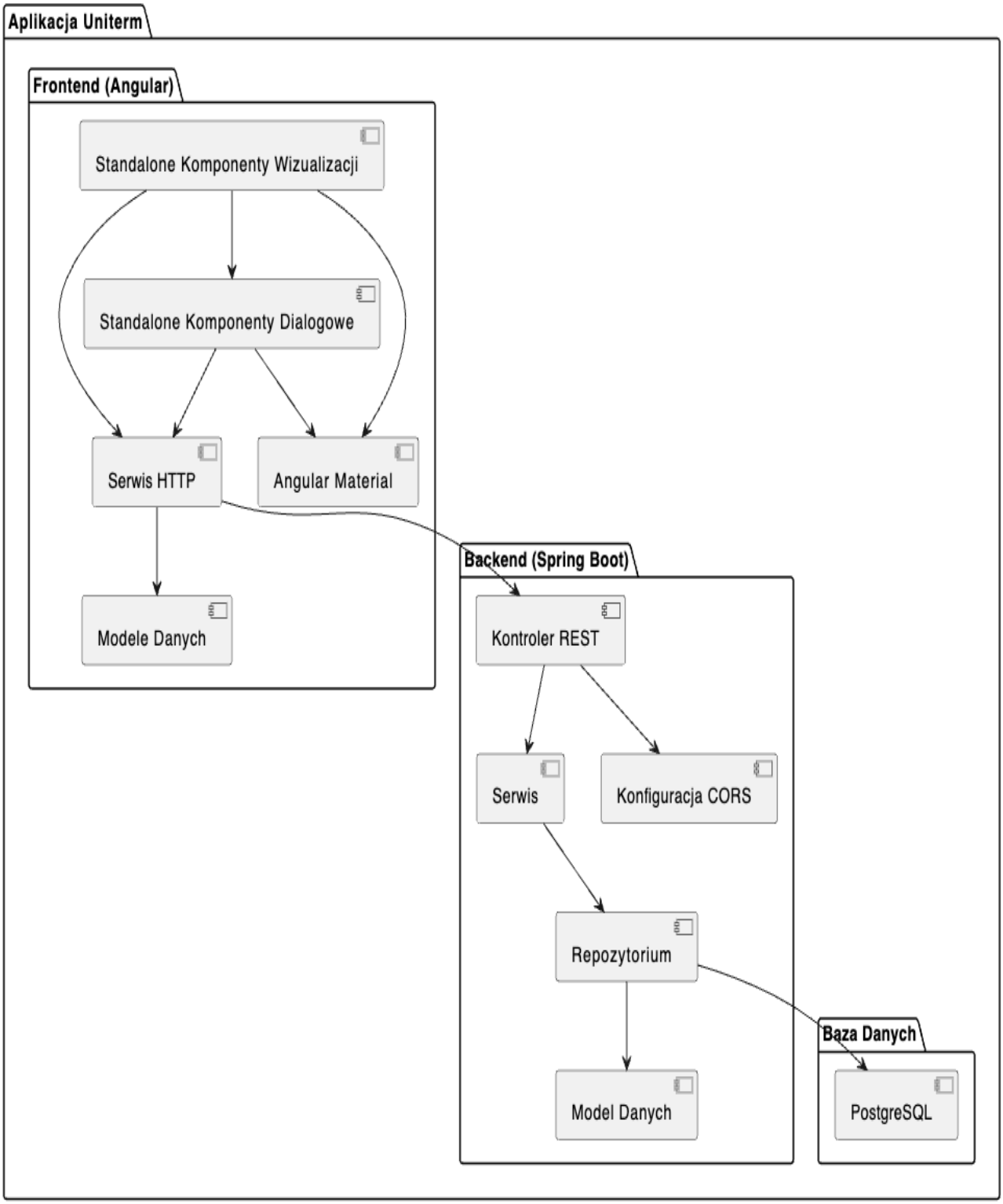
Poniższy rysunek prezentuje diagram sekwencji systemu.



Rysunek 5.1. Diagram sekwencji systemu.

6. Diagram warstw

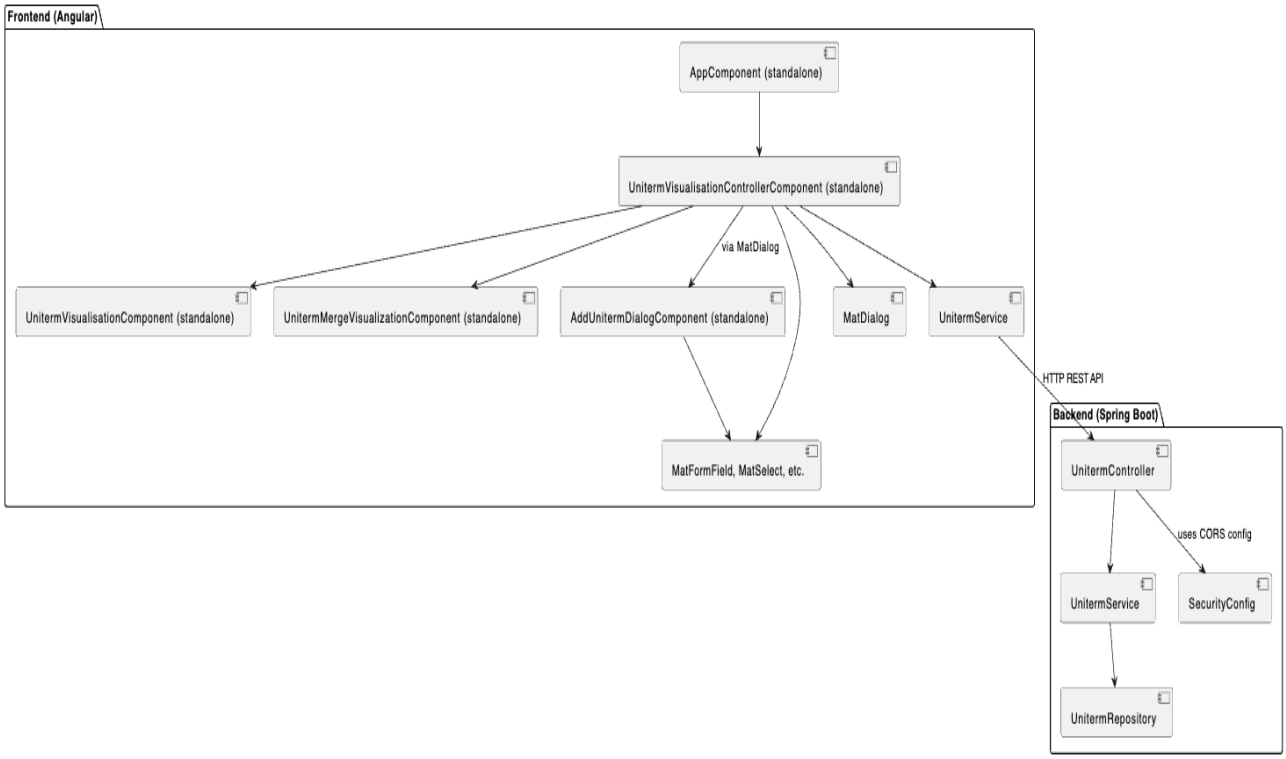
Poniższy rysunek prezentuje diagram warstw systemu.



Rysunek 6.1. Diagram warstw systemu.

7. Diagram komponentów

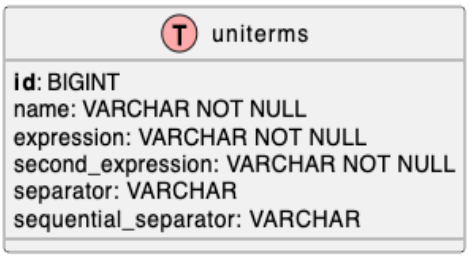
Poniższy rysunek prezentuje diagram komponentów systemu.



Rysunek 7.1. Diagram komponentów systemu.

8. Struktura bazy danych

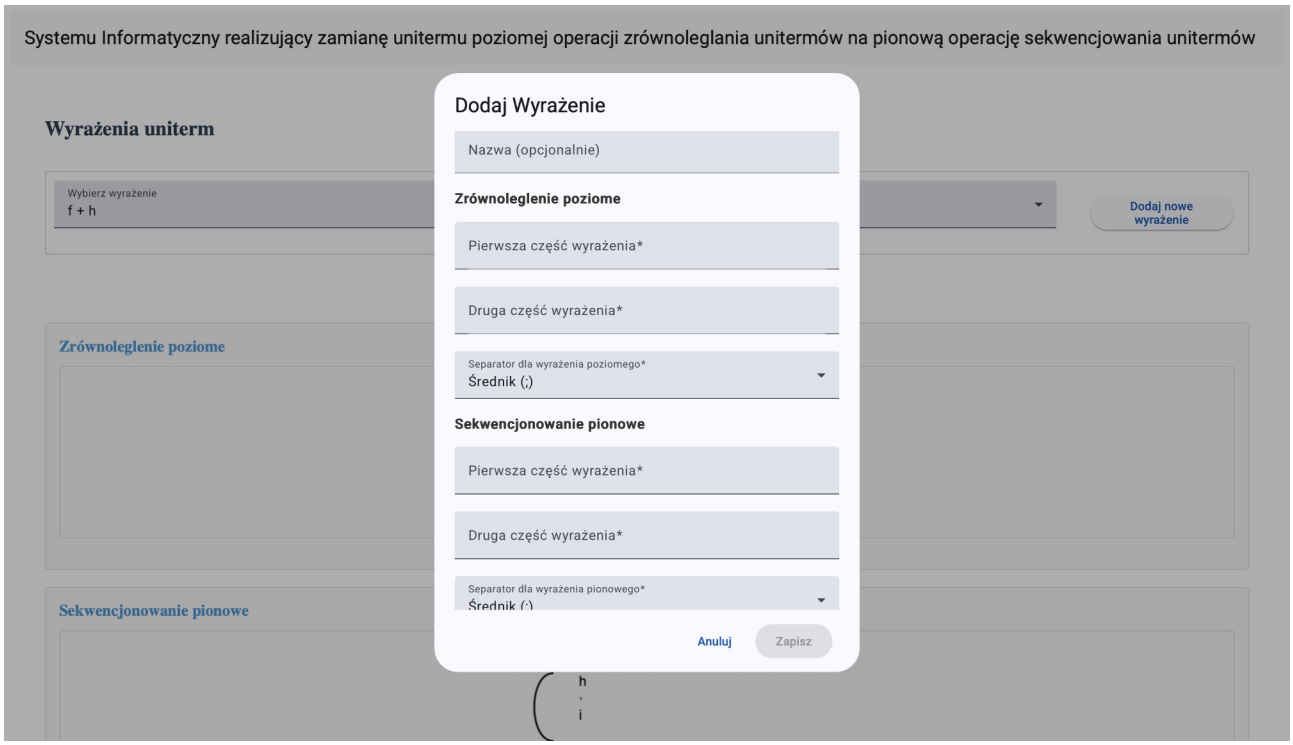
Poniższy rysunek prezentuje strukturę relazyjnej bazy danych systemu.



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

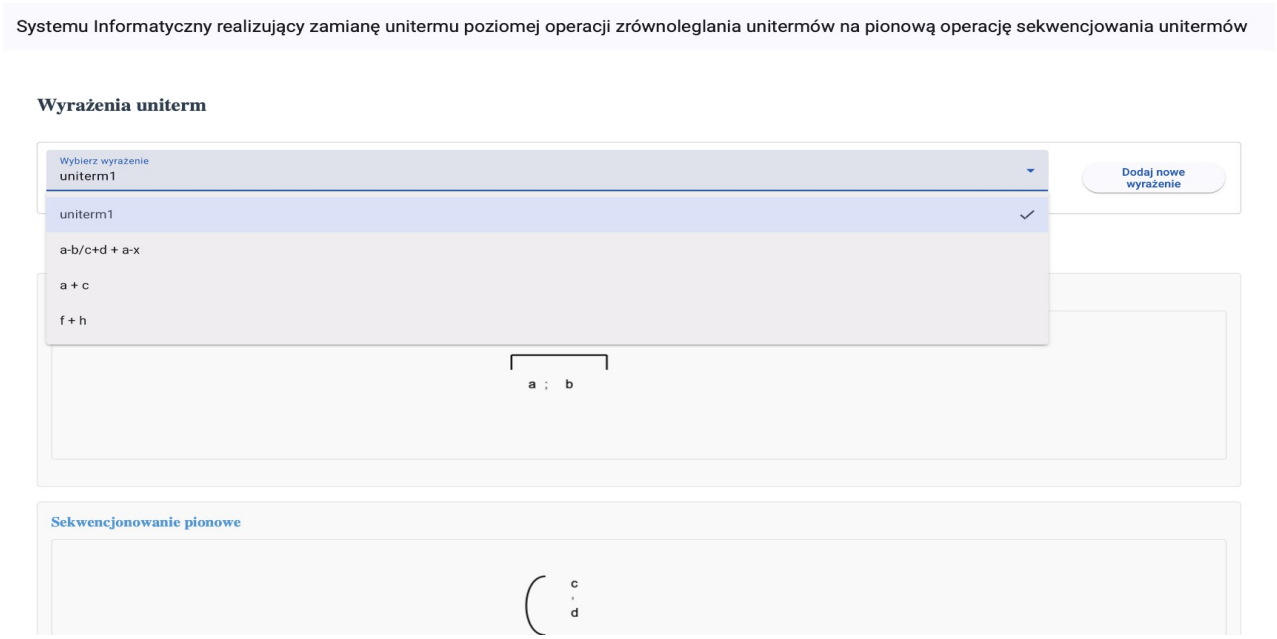
9. Zdjęcia z działania aplikacji

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



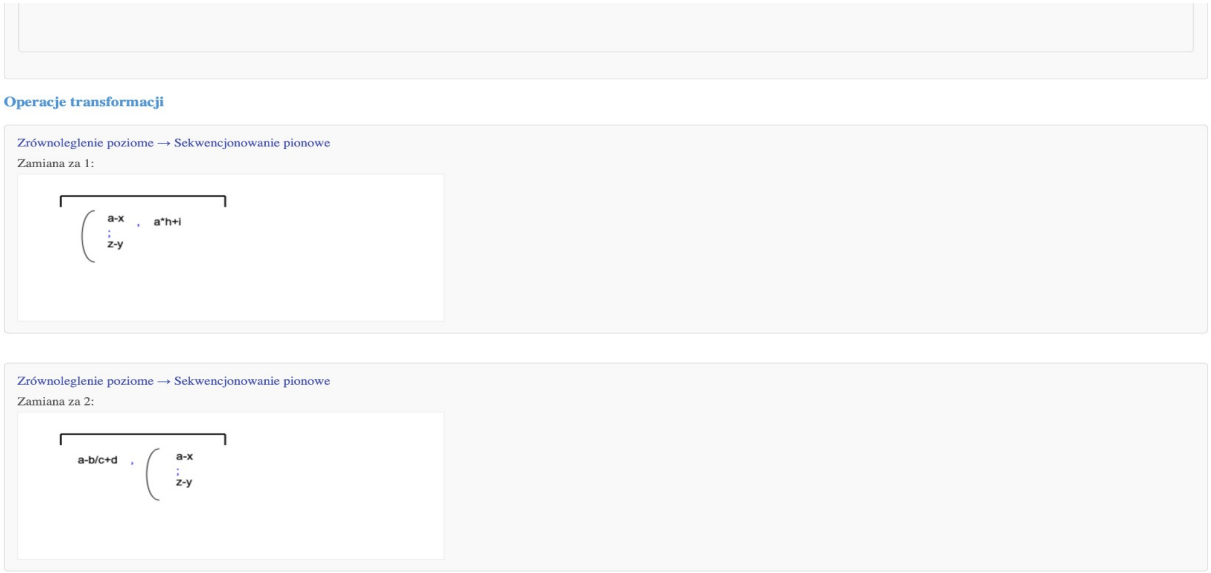
Rysunek 9.1. Dodawanie nowego rekordu do systemu.

Poniższy rysunek przedstawia reprezentację zrównoleglenia poziomego oraz sekwencjonowania pionowego wybranego z listy rekordu zawierającego unitermy i separatory.



Rysunek 9.2. Wynik przedstawienia operacji sekwencjonowania pionowego i zrównoleglenia poziomego po wybraniu rekordu z listy.

Poniższy rysunek przedstawia wynik zamiany zrównoleglenia poziomego na sekwencjonowanie pionowe za 1 i drugi element.



Rysunek 9.3. Wynik zamiany zrównoleglenia poziomego na sekwencjonowanie pionowe.