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

Opis najważniejszych elementów systemu został podzielony na dwie warstwy – prezentacji czyli warstwy odpowiadającej za graficzne przedstawienie dokonywanych operacji w systemie oraz logiki biznesowej przedstawiającej sposób przechowywania danych dotyczących unitermów w systemie i dostarczania ich do warstwy prezentacji.

1.1. Najważniejsze części kodu warstwy prezentacji

Poniższy listing przedstawia interfejsy opisujące model obiektów odpowiedzialnych za przedstawienie unitermu w postaci graficznej.

```
export enum OperationType {
  SEQUENCE = 'SEQUENCE',
  PARALLEL = 'PARALLEL',
  UNION = 'UNION',
}

export interface Uniterm {
  id?: number;
  name: string;
  description?: string;
  elements: UnitermElement[];
  operationType: OperationType;
  isTransformed?: boolean;
}

export interface UnitermElement {
  id?: number;
  expressionA: string;
  expressionB: string;
  expressionC?: string;
  x?: number;
  y?: number;
  width?: number;
  height?: number;
}
```

Listing 1.1. Interfejsy opisujące model reprezentowanego unitermu.

Poniższy listing przedstawia metody odpowiedzialne za komunikację z warstwą logiki biznesowej aplikacji, które pobierają listę dostępnych unitermów, pozwalają na zapis unitermu lub usunięcie unitermu (funkcja rozwojowa, nie zaimplementowana w widoku graficznym).

```
import { Injectable } from "@angular/core";
import { HttpClient } from "@angular/common/http";
import { Observable, of } from "rxjs";
import { OperationType, 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[]> {
    return this.http.get<Uniterm[]>(this.apiUrl);
  }
  getUnitermById(id: number): Observable<Uniterm> {
    return this.http.get<Uniterm>(`${this.apiUrl}/${id}`);
  }
  saveUniterm(uniterm: Uniterm): Observable<Uniterm> {
    if (uniterm.id) {
      return this.http.put<Uniterm>(`${this.apiUrl}/${uniterm.id}`, uniterm);
    } else {
      return this.http.post<Uniterm>(this.apiUrl, uniterm);
    }
  }
  deleteUniterm(id: number): Observable<void> {
    return this.http.delete<void>(`${this.apiUrl}/${id}`);
  }
  previewTransformation(uniterm: Uniterm): Uniterm {
    if (uniterm.operationType === OperationType.PARALLEL) {
      return {
        ...uniterm,
        operationType: OperationType.SEQUENCE,
        isTransformed: true,
        elements: [...uniterm.elements]
      };
    }
    return uniterm;
  }
  createTransformedUniterm(uniterm: Uniterm): Observable<Uniterm> {
    const transformed = this.previewTransformation(uniterm);
    return this.saveUniterm(transformed);
  }
}
```

Listing 1.2. Serwis obsługujący komunikację z warstwą logiki biznesowej.

Poniższy listing przedstawia klasę odpowiedzialną za dostarczenie widoku renderowanej operacji przekształcenia unitermu do odpowiedniego komponentu. Rendering oparty jest na użyciu dostępnych metod z pakietu CanvasRenderingContext2D, który umożliwia precyzyjne prowadzenie linii oraz zakrzywiania linii.

```
import {
  Component,
  ElementRef,
  Input,
  OnChanges,
  OnInit,
  SimpleChanges,
  ViewChild,
} from '@angular/core';
import { CommonModule } from '@angular/common';
import {
  Uniterm,
  OperationType,
  UnitermElement,
} from '../models/uniterm.model';
@Component({
  selector: 'app-uniterm-visualization',
  standalone: true,
  imports: [CommonModule],
  template: ` <div class="visualization-container">
      <canvas #unitermCanvas width="800" height="600"></canvas>
    </div>
  `, styles: [
    `.visualization-container {
       border: 1px solid #ccc;
       margin: 10px 0;
       background-color: white;
     }`,],
})
export class UnitermVisualizationComponent implements OnInit, OnChanges {
  @Input() uniterm?: Uniterm;
  @Input() fontFamily: string = 'Arial';
  @Input() fontSize: number = 14;
  @ViewChild('unitermCanvas', { static: true })
  canvasRef!: ElementRef<HTMLCanvasElement>;
  private ctx!: CanvasRenderingContext2D;
```

```
ngOnInit() {
    this.ctx = this.canvasRef.nativeElement.getContext('2d')!;
    this.redraw();
  }
  ngOnChanges(changes: SimpleChanges) {
    if (changes['uniterm'] || changes['fontFamily'] || changes['fontSize']) {
      this.redraw();
    }
  }
  redraw() {
    if (!this.ctx || !this.uniterm) return;
    this.ctx.clearRect(
      0,
      0,
      this.canvasRef.nativeElement.width,
      this.canvasRef.nativeElement.height
    );
    this.ctx.font = `${this.fontSize}px ${this.fontFamily}`;
    if (this.uniterm.operationType == OperationType.PARALLEL) {
      this.drawParallelUniterm();
    } else if (this.uniterm.operationType == OperationType.SEQUENCE) {
      this.drawSequentialUniterm();
    }
  }
  drawParallelUniterm() {
    if (!this.uniterm || !this.ctx) return;
    const elements = this.uniterm.elements;
    if (elements.length === 0) {
      this.ctx.fillText('Dodaj elementy do unitermu', 50, 50);
      return;
    }
    const startX = 50;
    const startY = 100;
    const padding = 20;
    let combinedText = '';
    elements.forEach((element, index) => {
      combinedText += `${element.expressionA} ; ${element.expressionB}`;
      if (index < elements.length - 1) {
        combinedText += ' ; ';
      }
    });
```

```
    const textWidth = this.ctx.measureText(combinedText).width;
    const boxWidth = textWidth + padding * 2;
    const boxHeight = this.fontSize * 2;
    this.ctx.fillText('zrównoleglenie poziome', startX, startY - 20);
    this.ctx.beginPath();
    this.ctx.moveTo(startX, startY);
    this.ctx.lineTo(startX + boxWidth, startY);
    this.ctx.moveTo(startX, startY);
    this.ctx.lineTo(startX, startY + boxHeight / 2);
    this.ctx.moveTo(startX + boxWidth, startY);
    this.ctx.lineTo(startX + boxWidth, startY + boxHeight / 2);
    this.ctx.stroke();
    this.ctx.fillText(
      combinedText,
      startX + padding,
      startY + boxHeight / 2 + 5
    );
  }
  drawSequentialUniterm() {
    if (!this.uniterm || !this.ctx) return;
    const elements = this.uniterm.elements;
    if (elements.length === 0) return;
    const startX = 100;
    const startY = 50;
    const padding = 20;
    const spacing = 30;
    let currentY = startY;
    elements.forEach((element, index) => {
         this.ctx.fillText(element.expressionA, startX + padding, currentY +
this.fontSize);
      currentY += spacing;
      this.ctx.fillText(';', startX + padding, currentY + this.fontSize);
      currentY += spacing;
         this.ctx.fillText(element.expressionB, startX + padding, currentY +
this.fontSize);
      if (index < elements.length - 1) {
        currentY += spacing * 2;
      }
    });
    this.drawCurvedBracket(
      startX - 10,
```

```
      startY - 10,
      currentY - startY + this.fontSize * 2
    );
    this.ctx.fillText(
      'sekwencjonowanie pionowe',
      startX,
      currentY + this.fontSize * 3
    );
  }
  drawCurvedBracket(x: number, y: number, height: number) {
    this.ctx.beginPath();
    this.ctx.moveTo(x, y);
    this.ctx.bezierCurveTo(
      x - 20,
      y + height / 4,
      x - 20,
      y + (3 * height) / 4,
      x,
      y + height
    );
    this.ctx.stroke();
  }
  drawVerticalBracket(x: number, y: number, height: number, width: number) {
    this.ctx.beginPath();
    this.ctx.moveTo(x, y);
    this.ctx.lineTo(x, y + height);
    this.ctx.moveTo(x, y);
    this.ctx.lineTo(x + width, y);
    this.ctx.moveTo(x, y + height);
    this.ctx.lineTo(x + width, y + height);
    this.ctx.stroke();
  }
}
```

Listing 1.3. Klasa odpowiadająca za komponent realizujący operację rysowania operacji sekwencjonowania pionowego lub zrównoleglenia poziomego unitermów.

1.2. Najważniejsze części kodu warstwy logiki biznesowej

Poniższy listing przedstawia kod klasy REST API kontrolera, który umożliwia na obsługę generowanych przez warstwę prezentacji żądań pobrania listy unitermów czy dodania nowego unitermu czy edycji istniejącego unitermu poprzez szereg adnotacji i przekazania żądania do klasy serwisu odpowiadającej za głębszą logikę biznesową.

```
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);
    }
```

```
    @PutMapping("/{id}")
        public ResponseEntity<Uniterm> updateUniterm(@PathVariable Long id,
@RequestBody Uniterm uniterm) {
        if (!unitermService.existsById(id)) {
            return ResponseEntity.notFound().build();
        }
        uniterm.setId(id);
        return ResponseEntity.ok(unitermService.save(uniterm));
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deleteUniterm(@PathVariable Long id) {
        if (!unitermService.existsById(id)) {
            return ResponseEntity.notFound().build();
        }
        unitermService.deleteById(id);
        return ResponseEntity.noContent().build();
    }
}
```

Listing 1.4. Klasa odpowiadająca za obsługę żądań generowanych po stronie warstwy prezentacji.

Poniższy listing przedstawia kod serwisu odpowiadającego za manipulację rekordami dotyczącymi unitermów w bazie danych za pomocą klasy repozytorium będącego nakładką ORM.

```
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);
    }
    public void deleteById(Long id) {
        unitermRepository.deleteById(id);
    }
}
```

Listing 1.5. Klasa serwisu odpowiadająca za obsługę komunikacji z bazą danych.

Poniższy listing przedstawia klasę przedstawiającą model zapisywanego w bazie danych unitermu, którego tabela relacyjnej bazy danych jest automatycznie tworzona w przypadku uruchomienia systemu.

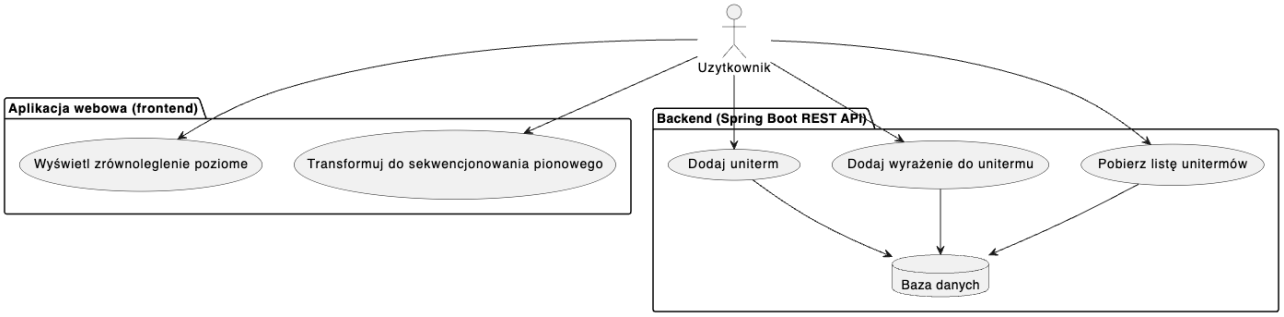
```
package com.uniterm.backend.model;

import jakarta.persistence.*;
import java.util.ArrayList;
import java.util.List;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

@Entity
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Builder
```

```java
@Table(name = "uniterms")
public class Uniterm {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(nullable = false)
    private String name;

    private String description;

    @Enumerated(EnumType.STRING)
    @Column(nullable = false)
    private OperationType operationType;

    @Builder.Default
    @OneToMany(cascade = CascadeType.ALL, orphanRemoval = true)
    @JoinColumn(name = "uniterm_id")
    private List<UnitermElement> elements = new ArrayList<>();

    public void addElement(UnitermElement element) {
        elements.add(element);
    }

    public void removeElement(UnitermElement element) {
        elements.remove(element);
    }
}
```

Listing 1.6. Klasa odpowiedzialna za reprezentację encji unitermu w bazie danych.

2. Diagram przypadków użycia

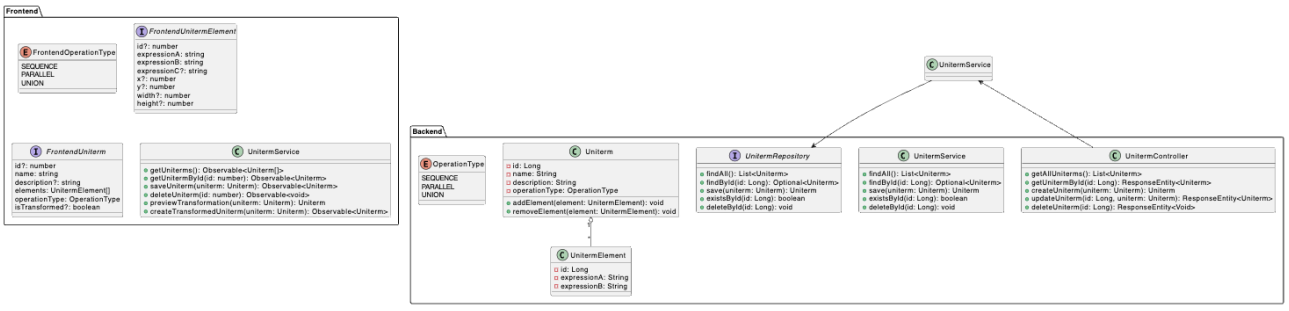
Poniższy rysunek prezentuje diagram przypadków użycia w systemie z podziałem na komunikację użytkownika z warstwą prezentacji (aplikacją webową) oraz warstwą logiki biznesowej (backend).



Rysunek 2.1. Diagram przypadków użycia systemu.

3. Diagram klas

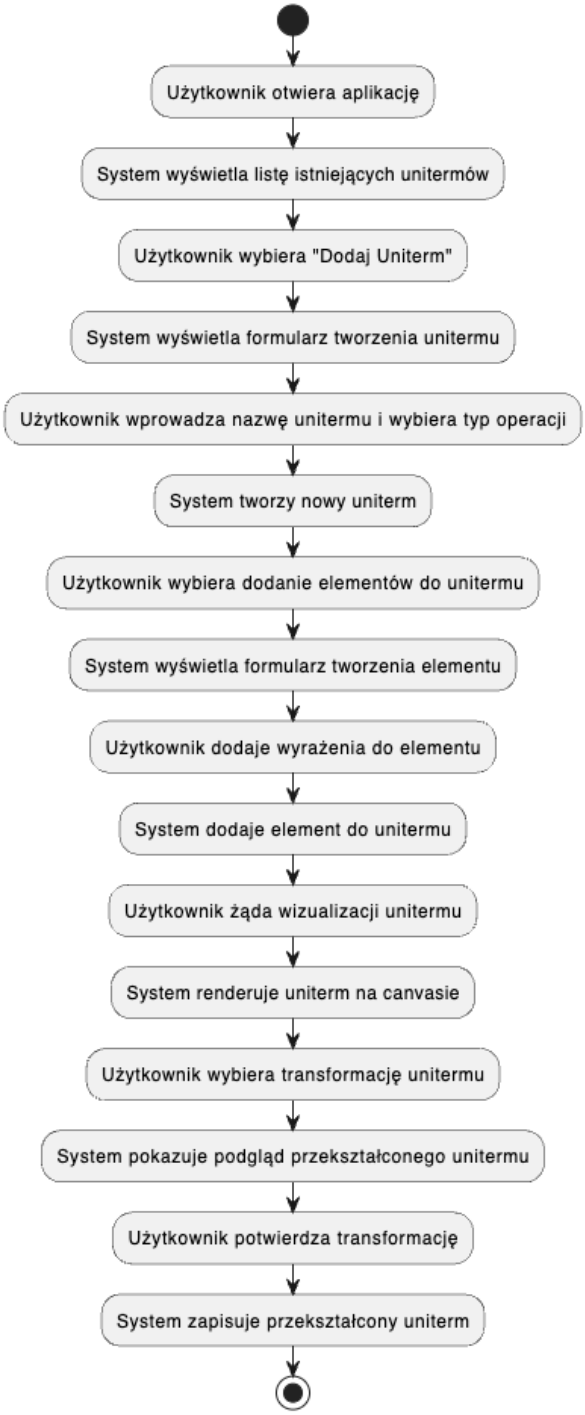
Poniższy rysunek prezentuje diagram klas systemu z podziałem na warstwę prezentacji (frontend) oraz warstwę logiki biznesowej (backend).



Rysunek 3.1. Diagram klas systemu.

4. Diagram aktywności

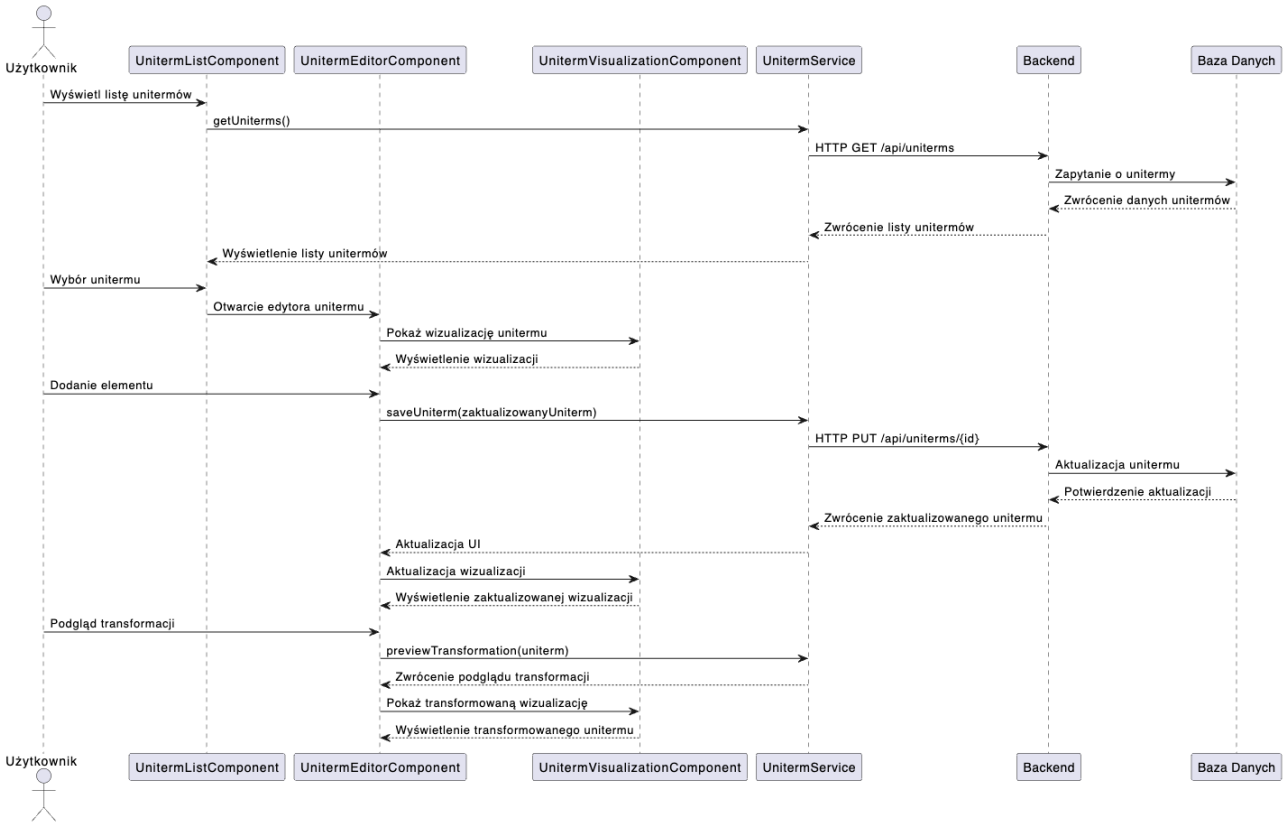
Poniższy rysunek prezentuje diagram aktywności systemu.



Rysunek 4.1. Diagram aktywności systemu.

5. Diagram sekwencji

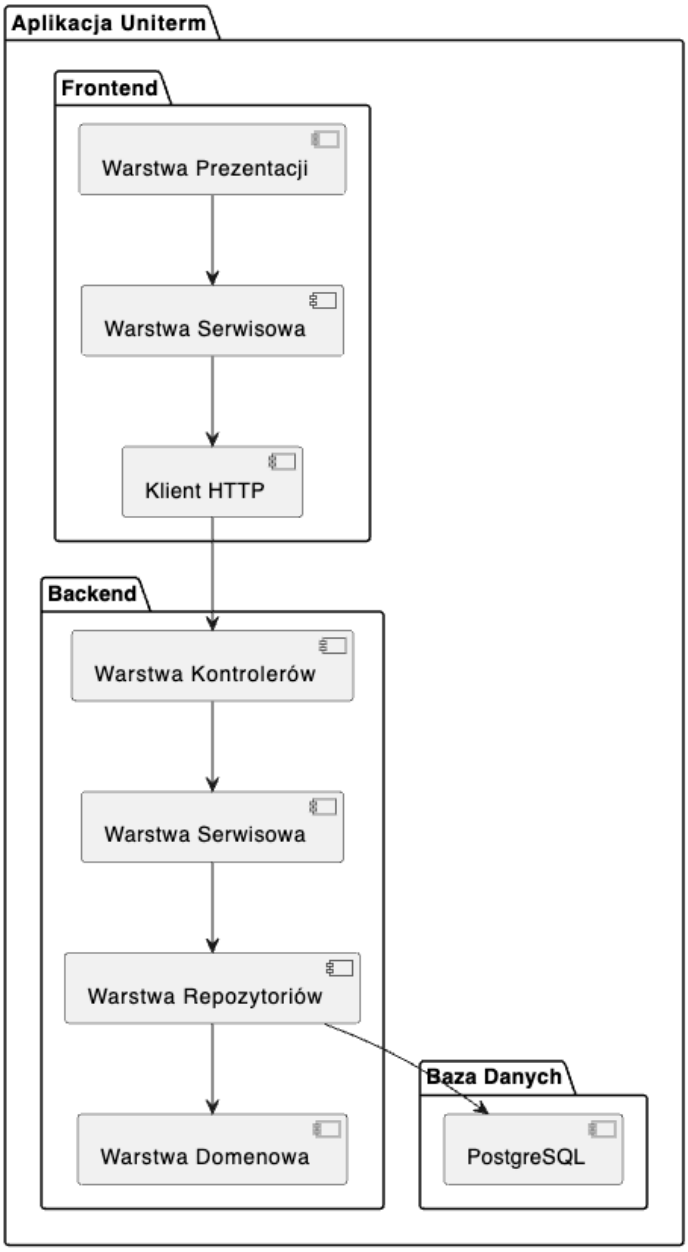
Poniższy rysunek prezentuje diagram sekwencji systemu.



Rysunek 5.1. Diagram sekwencji systemu.

6. Diagram warstw

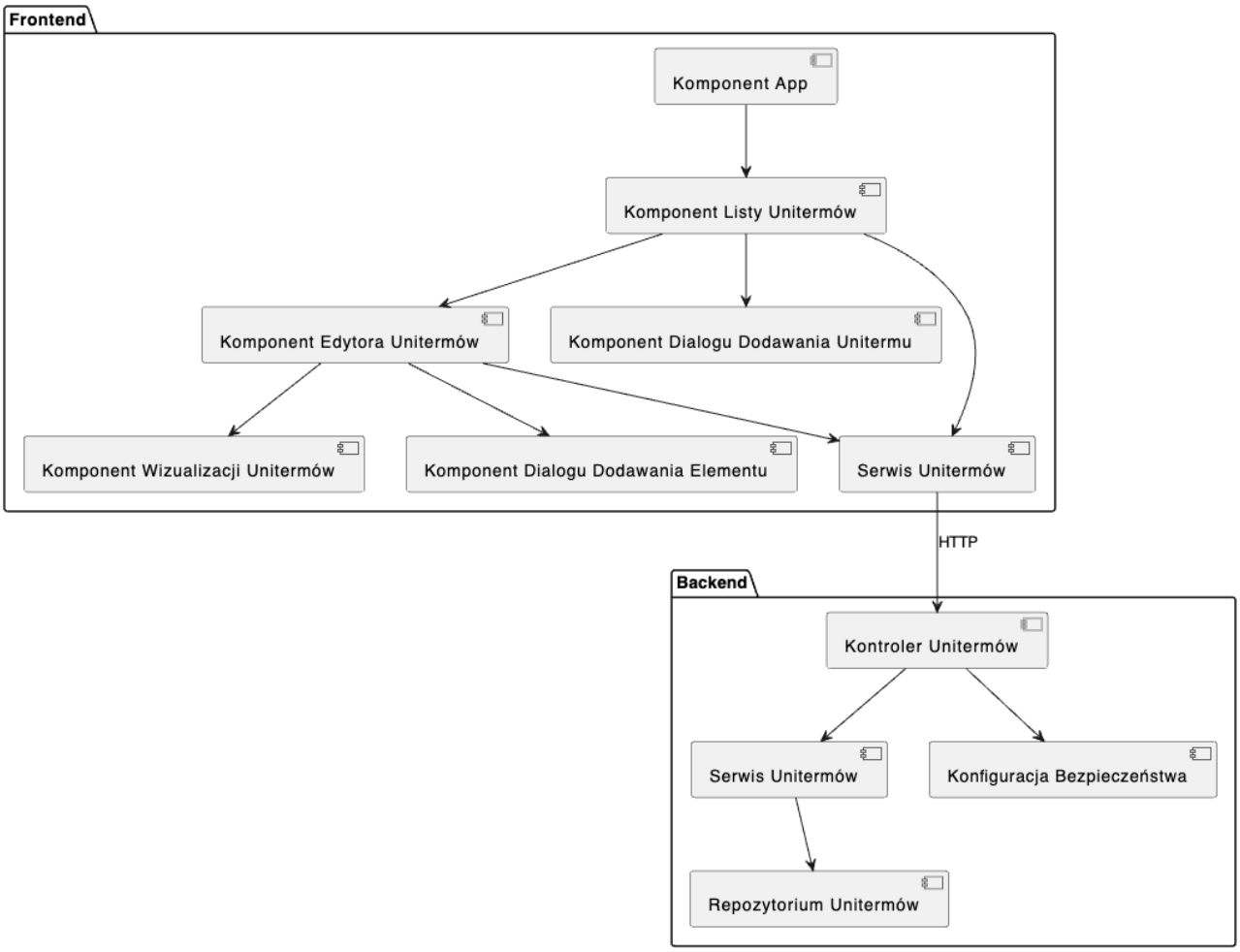
Poniższy rysunek prezentuje diagram warstw systemu.



Rysunek 6.1. Diagram warstw systemu.

7. Diagram komponentów

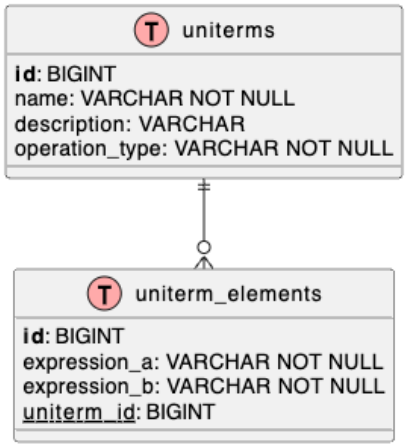
Poniższy rysunek prezentuje diagram komponentów systemu.



Rysunek 7.1. Diagram komponentów systemu.

8. Struktura bazy danych

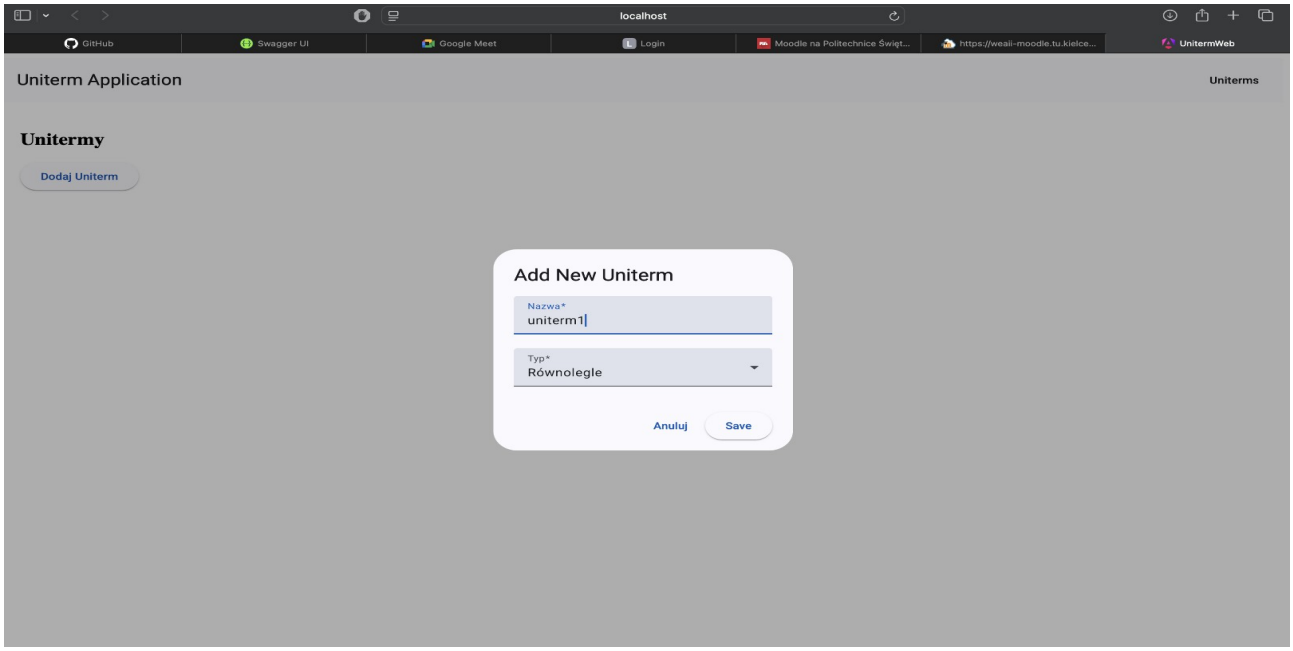
Poniższy rysunek prezentuje strukturę relazyjnej bazy danych systemu.



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

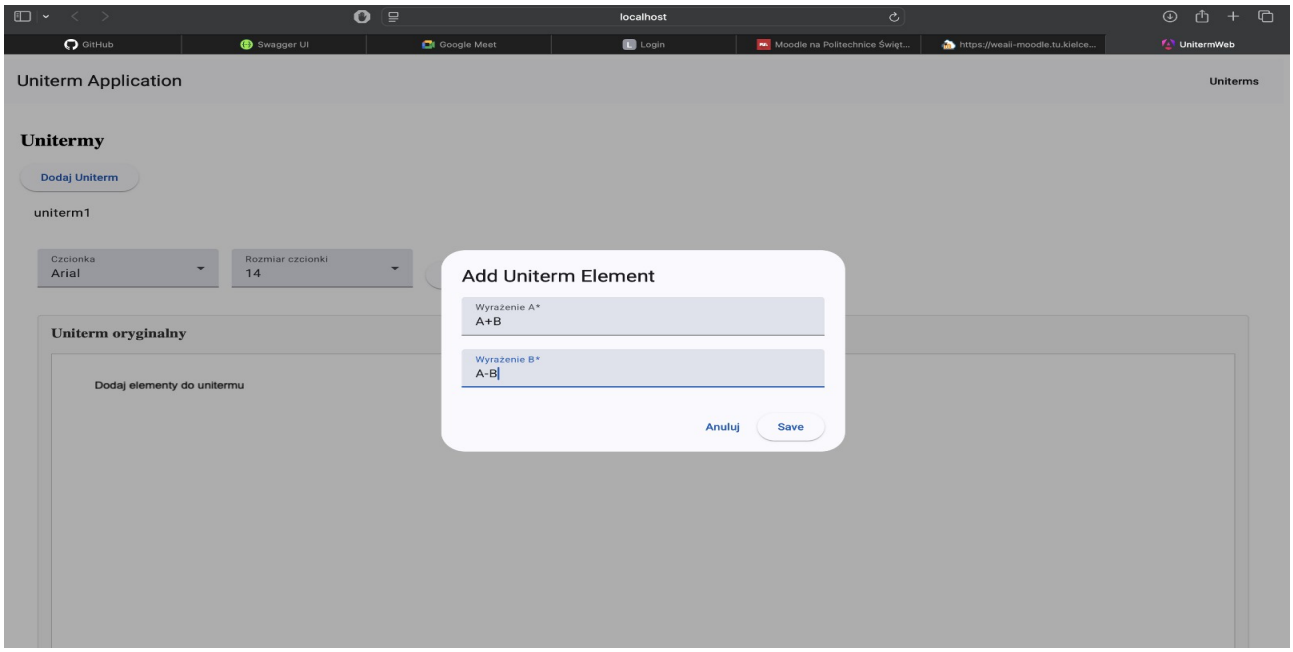
9. Zdjęcia z działania aplikacji

Poniższy rysunek przedstawia dodawanie nowego unitermu do systemu.



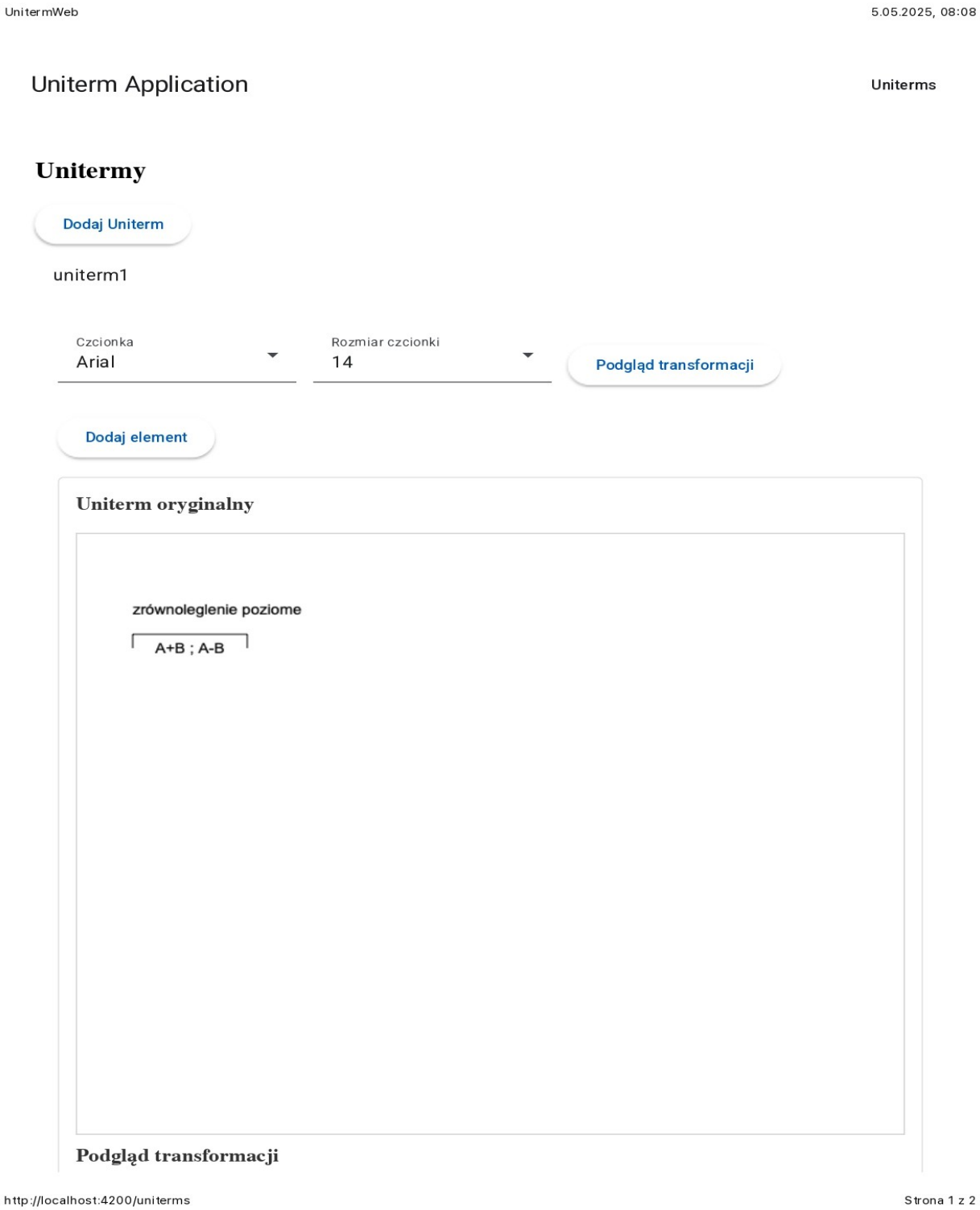
Rysunek 9.1. Dodawanie nowego unitermu.

Następnie, poniższy rysunek prezentuje dodawanie nowych wyrażeń do wybranego unitermu z pobranej listy unitermów w systemie.



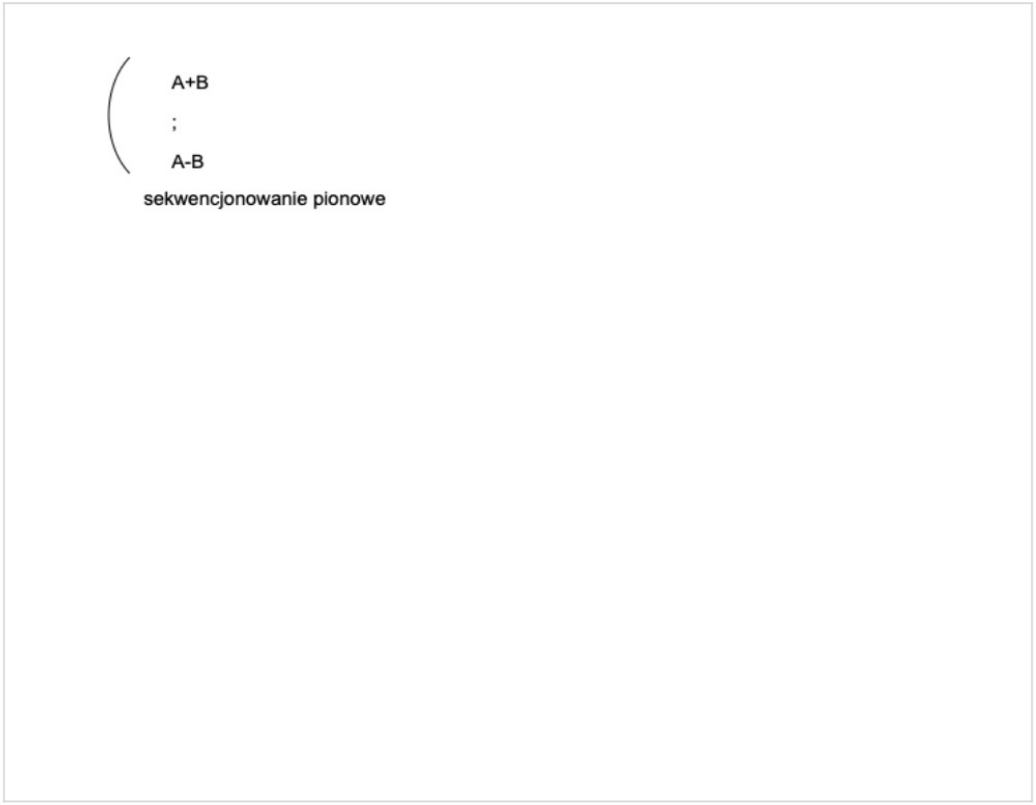
Rysunek 9.2. Dodawanie nowych wyrażeń do unitermu.

Poniższy rysunek przedstawia wyświetlenie operacji zrównoleglenia poziomego unitermu.



Rysunek 9.3. Prezentacja wyświetlonego zrównoleglenia poziomego unitermu.

Poniższy rysunek przedstawia wyświetlenie operacji transformacji zrównoleglenia poziomego unitermu na sekwencjonowanie pionowe unitermu.



Elementy

Wyrażenie A	Wyrażenie B
A+B	A-B

Rysunek 9.4. Prezentacja wyświetlonego sekwencjonowania pionowego unitermu.