МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«До захисту допущено»
В.о. завідувача кафедри

_____ Ігор ШЕЛЕХОВ
(підпис)

червня 202_ р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 - Комп'ютерних наук,
освітньо-професійної програми «Інформатика»
на тему: «Інформаційна система моделювання класифікації регіонів за показниками діджиталізації»
здобувача групи ІН-06-2 Крижанівського Дениса Олесьовича

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Денис КРИЖАНІВСЬКИЙ
_____
(підпис)

Керівник, Наталія БАРЧЕНКО
кандидат технічних наук, доцент

_____
(підпис)

Суми – 2024

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри

Ігор ШЕЛЕХОВ

_____
(підпис)

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавра

зі спеціальності 122 - Комп'ютерних наук, освітньо-професійної програми «Інформатика»
здобувача групи ІН-06-2 Крижанівського Дениса Олесьовича

1. Тема роботи: «Інформаційна система моделювання класифікації регіонів за показниками діджиталізації»
затверджую наказом по СумДУ від *«22» квітня 2024 р. № 0414-VI*
2. Термін здачі здобувачем кваліфікаційної роботи *до 09 червня 2024 року*
3. Вхідні дані до кваліфікаційної роботи _____
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)
1) Аналіз предметної області, постановка й формування завдань дослідження. 2) Огляд і вибір технологій, що можна вибрати для аналізу індексів діджиталізації. 3) Розробка інформаційної системи для розрахунку індексу діджиталізації та аналіз результатів.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)_____
6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «06» травня 2024 р.

Завдання прийняв до виконання _____ Керівник _____
(підпис) (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Аналіз проблеми предметної області, постановка й формування завдань дослідження*	06.05.24-10.05.24	
2	*Огляд технологій, що можливо використати для вирішення задачі*	11.05.24-13.05.24	
3	*Розробка інформаційної системи для класифікації регіонів*	14.05.24-25.05.24	
4	*Аналіз функціональності системи та отриманих результатів*	26.05.24-27.05.24	

5	*Оформлення пояснювальної записки до кваліфікаційної роботи*	28.05.24-31.05.24	

Здобувач вищої освіти _____ Керівник _____

(підпис) (підпис)

АНОТАЦІЯ

Записка: 63 стр., 17 рис., 1 додаток, 30 використаних джерел.

Обґрунтування актуальності теми роботи – Тема кваліфікаційної роботи є актуальною, оскільки присвячена розв'язанню задачі класифікації регіонів України за показниками діджиталізації шляхом використання апарату нечіткого логічного виведення для аналізу субіндексів діджиталізації.

Об'єкт дослідження — класифікації регіонів за показниками діджиталізації.

Мета роботи — розробка системи для моделювання класифікації регіонів за показниками діджиталізації.

Методи дослідження — апарат нечіткої логіки та ретроспективний аналіз показників діджиталізації.

Гіпотеза — Використання нечіткої логіки для класифікації регіонів за показниками діджиталізації оцінювати рівень цифрового розвитку регіонів та підвищити ефективність прийняття управлінських рішень.

Результати — розроблено інформаційну систему, яка здатна аналізувати та класифікувати регіони за показниками діджиталізації. Система підтримує CRUD операції для роботи з сутностями системи, автоматичну побудову ієрархічної структури змінних. Результати аналізу субіндексів корелюють з реальними значеннями індексів діджиталізації, відзначається необхідність у подальшому удосконаленні системи та набору правил.

ІНФОРМАЦІЙНА СИСТЕМА, НЕЧІТКА ЛОГІКА, ПОКАЗНИКИ ДІДЖИТАЛІЗАЦІЇ, JAVA, FUZZYLITE, JAVAFX.

ЗМІСТ

ВСТУП

Актуальність. Інформаційні технології грають ключову роль у визначенні економічного та соціального прогресу регіонів. З огляду на це, важливість дослідження та розробки методів моделювання для класифікації регіонів за показниками діджиталізації набуває особливої актуальності. Використання інформаційні технології для оцінки та класифікації регіонів дозволить зробити обґрунтовані висновки щодо рівня їх діджиталізації.

Об’єкт дослідження. Класифікації регіонів за показниками діджиталізації.

Предмет дослідження. Система для створення ефективної моделі класифікації регіонів на основі різних показників діджиталізації.

Гіпотеза. Використання нечіткої логіки для класифікації регіонів за показниками діджиталізації оцінювати рівень цифрового розвитку регіонів та підвищити ефективність прийняття управлінських рішень.

Структура. Дана робота складається зі вступу, аналітичного огляду, постановки задачі, вибір методу розв’язання поставленої задачі, опису програмної реалізації інформаційної системи, висновків, списку використаних джерел та додатків.

1 АНАЛІТИЧНИЙ ОГЛЯД

1.1 Дослідження предметної області

Показники діджиталізації є інструментами, що дозволяють оцінити, наскільки інформаційні технології інтегровані в різні аспекти життя регіону. Вони включають різноманітні метрики, які демонструють глибину впровадження цифрових технологій у повсякденне життя, економіку, освіту та управління регіонів [1].

Одним із важливих аспектів показників діджиталізації є оцінка рівня цифрової інфраструктури. Це охоплює доступність та якість широкосмугового інтернету, розгортання мереж 5G та наявність різних форм цифрових комунікацій. Розгляд цих показників дозволяє зрозуміти, наскільки регіони забезпечені необхідною технічною основою для реалізації цифрових ініціатив.

Ще один ключовий аспект – це рівень цифрової грамотності населення. Важливо аналізувати, наскільки обізнане населення з цифровими технологіями, як часто ці технології використовуються в повсякденному житті, а також рівень IT-освіти серед громадян. Рівень цифрової грамотності впливає на спроможність населення ефективно користуватися цифровими послугами та технологіями.

Важливим є також аналіз використання цифрових технологій у бізнесі. Це включає оцінку того, як місцеві компанії інтегрують цифрові технології у свою діяльність, включаючи електронну комерцію, цифровий маркетинг та автоматизацію процесів. Рівень впровадження цифрових технологій у бізнесі може значно впливати на економічний розвиток регіону.

Останній аспект – це рівень розвитку цифрових урядових послуг. Це стосується ефективності використання урядом цифрових технологій для надання послуг громадянам. Показники в цій сфері включають розгортання електронного уряду, наявність цифрових документів, онлайн-сервісів та інших форм цифрового управління. Розвиток цифрових урядових послуг

може значно покращити ефективність та доступність управління для громадян.

1.2 Визначення актуальності проекту

Показники діджиталізації відіграють важливу роль у багатьох аспектах сучасного суспільства та економіки, від планування розвитку регіонів до стратегічного управління.

Оцінка показників діджиталізації дозволяє урядам та місцевій адміністрації виявляти ключові сфери для інвестицій та вдосконалення. Наприклад, якщо аналіз показує низький рівень доступності широкосмугового інтернету в певному регіоні, це може стати пріоритетом для розвитку інфраструктури. Таким чином, ці показники допомагають формувати обґрунтовані рішення щодо розподілу ресурсів та пріоритетів розвитку.

Аналіз показників діджиталізації може сприяти визначенню потенціалу для зростання цифрового бізнесу та стимулювання інновацій. Наприклад, високий рівень цифрової грамотності населення та добре розвинута цифрова інфраструктура можуть створювати сприятливі умови для стартапів та приваблювати інвестиції в технологічний сектор. Таким чином, показники діджиталізації відіграють ключову роль у формуванні економічної стратегії регіону.

Використання цих показників допомагає у визначенні впливу цифрових технологій на якість життя громадян. Наприклад, розвиток цифрових урядових послуг може підвищити доступність та ефективність публічних послуг, що, в свою чергу, поліпшує загальний рівень задоволеності громадян. Також це може впливати на освітні можливості, забезпечуючи ширший доступ до інформації та ресурсів.

Для урядів та організацій показники діджиталізації є важливим інструментом для розробки стратегій розвитку цифрової економіки та суспільства. Вони дозволяють здійснювати обґрунтоване планування та

визначати напрямки, в яких потрібно зосередити зусилля для досягнення максимального ефекту від цифровізації.

1.3 Постановка задачі

Основними завданнями цієї роботи є:

1. Проаналізувати існуючі технології, що здатні виконувати роль математичного апарату.
2. Розробити інформаційну систему, здатну вирішити задачу класифікації регіонів за показниками діджиталізації.
3. Під час розробки врахувати необхідність у легкому внесенні змін до створюваної системи.
4. Забезпечити систему потужним математичним апаратом, здатним працювати з складними структурами даних.
5. Протестувати систему та отримати результати розрахунку показників діджиталізації.
6. Проаналізувати отримані результати.

2 ВИБІР МЕТОДУ РОЗВ’ЯЗАННЯ ЗАДАЧІ

2.1 Математичний апарат

Розробка ефективної моделі класифікації регіонів за показниками діджиталізації вимагає уважного вибору відповідних технологій та підходів. Це завдання передбачає створення системи, здатної аналізувати різноманітні дані, класифікувати їх та надавати точні та об'єктивні висновки. Особливу увагу при цьому заслуговує використання нечіткої логіки як математичного апарату моделювання.

З одного боку, є підходи, такі як машинне навчання та штучний інтелект, які передбачають використання алгоритмів для аналізу великих даних та ідентифікації закономірностей. Інші методики включають статистичний аналіз для визначення тенденцій і кореляцій між різними показниками діджиталізації, а також розробку систем прийняття рішень на основі встановлених правил.

Однак, особливу перевагу представляє використання нечіткого логічного виведення [2] (рис. 2.1). Такий підхід дозволяє досягнути гнучкості у визначенні критеріїв [6], особливо важливої при оцінці таких неоднозначних показників, як рівень цифрової грамотності або якість інтернет-покриття. Нечітка логіка ефективно справляється з неточними або неповними даними, надаючи збалансовані висновки. Крім того, моделі, побудовані на основі нечіткої логіки, легше інтерпретувати та зрозуміти, оскільки вони використовують умови та правила, близькі до людського мислення, і водночас адаптивні до змінних умов.

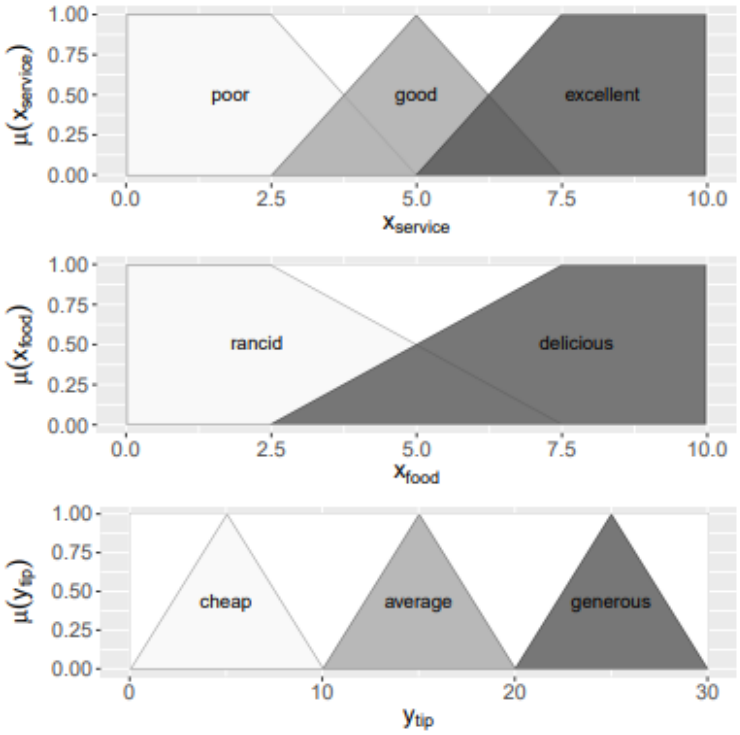

Рисунок 2.1 – Приклад використання нечіткої логіки

Таким чином, обираючи нечітку логіку як основний інструмент для розробки моделі класифікації, можна досягти високої ефективності та точності у висновках, що є критично важливим для об'єктивної оцінки та класифікації регіонів за рівнем їх діджиталізації.

.Для реалізації технології, заснованої на нечіткій логіці з використанням Java, можна розглянути кілька фреймворків і бібліотек, що надають необхідні інструменти та функціональні можливості. Ось декілька з них:

jFuzzyLogic: jFuzzyLogic є іншою потужною бібліотекою для реалізації нечітких систем у Java. Цей фреймворк зосереджений на забезпеченні точності та ефективності в реалізації нечітких систем [3]. Він дозволяє легко інтегрувати нечіткі правила та логіку в Java-програми, забезпечуючи високу продуктивність і точність.

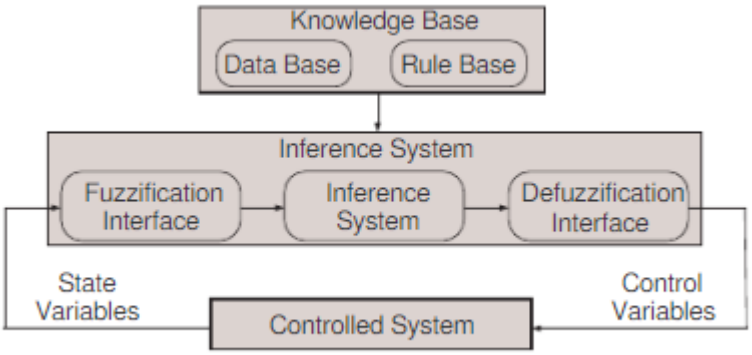

Рисунок 2.2 – Реалізація апарату нечіткої логіки у jFuzzyLogic

FuzzyLite є відкритим інструментом для реалізації нечітких логічних систем, який може бути використаний у багатьох програмних мовах, включаючи Java. Цей фреймворк надає зручний та гнучкий спосіб створення, тестування та впровадження нечітких контролерів та систем [4].

FuzzyJ Toolkit є ще одним варіантом для реалізації нечіткої логіки в Java. Цей фреймворк включає набір інструментів для створення нечітких логічних систем та обробки нечітких запитів. FuzzyJ забезпечує розширений API для створення, валідації та виконання нечітких правил [5], а також підтримує інтеграцію з іншими Java-додатками.

При порівнянні FuzzyLite, jFuzzyLogic та FuzzyJ Toolkit з точки зору гнучкості, простоти використання та ефективності, можна виділити ряд важливих характеристик цих фреймворків, які допоможуть визначити їхню придатність для різних видів проектів.

FuzzyLite вирізняється своєю високою гнучкістю, дозволяючи легко адаптувати логіку до потреб конкретного проекту. Він підтримує різноманітні нечіткі оператори та методи виведення. jFuzzyLogic також надає значну гнучкість, особливо з його підтримкою Fuzzy Control Language, який є корисним для опису нечітких систем. FuzzyJ Toolkit, хоча і є гнучким, може бути трохи менш адаптивним порівняно з двома іншими фреймворками, оскільки його основний фокус – на обробці нечітких запитів і виконанні нечітких правил.

З точки зору простоти використання, FuzzyLite вирізняється своїм інтуїтивним інтерфейсом і легкістю інтеграції, роблячи його ідеальним вибором для тих, хто вперше звертається до нечіткої логіки. jFuzzyLogic також має переваги у легкості використання, зокрема завдяки підтримці FCL, що дозволяє користувачам легко описувати нечіткі системи. FuzzyJ Toolkit може вимагати трохи більше часу для освоєння, але все ще забезпечує солідний функціонал для роботи з нечіткою логікою.

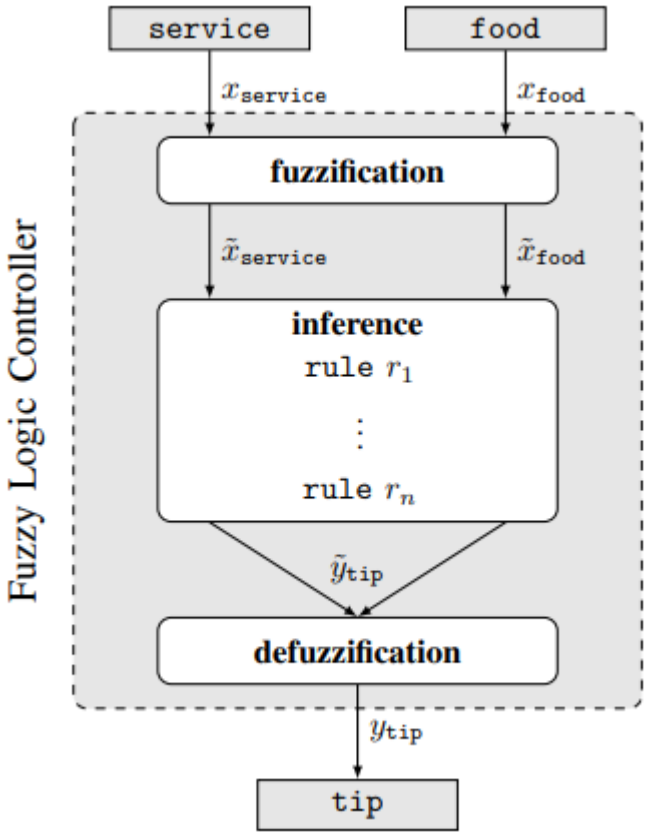

Рисунок 2.3 – реалізація апарату нечіткої логіки у FuzzyLite

У плані ефективності, всі три фреймворки демонструють хороші результати. FuzzyLite і jFuzzyLogic оптимізовані для швидкого виконання, що робить їх придатними для додатків, які вимагають високої продуктивності. FuzzyJ Toolkit також ефективний у обробці нечітких правил і запитів, хоча його продуктивність може трохи відрізнятися від інших двох фреймворків, залежно від конкретного використання.

Загалом, з урахуванням потреби в обробці складних і часто неоднозначних даних у проектах класифікації регіонів за показниками діджиталізації, FuzzyLite виступає як надійний, гнучкий і користувацьки-зручний інструмент. Його можливості дозволяють ефективно реалізувати складні нечіткі логічні системи, необхідні для точного та об'єктивного аналізу діджиталізації регіонів.

Перш за все, висока гнучкість FuzzyLite дозволяє адаптувати логіку нечітких систем до конкретних потреб і особливостей діджиталізації регіонів. Це означає, що можна ефективно обробляти різноманітні та неоднозначні дані, які часто зустрічаються при аналізі рівня діджиталізації.

Простота використання є іншою вагомою перевагою FuzzyLite. Інтуїтивний інтерфейс та легка інтеграція з Java роблять цей фреймворк доступним навіть для тих, хто вперше стикається з нечіткою логікою. Це знижує поріг входження та спрощує процес розробки.

Нарешті, ефективність FuzzyLite гарантує, що системи, побудовані на його основі, будуть швидко обробляти дані, надаючи результати в режимі реального часу, що є критично важливим для додатків, зорієнтованих на аналіз даних.

2.2 Графічний інтерфейс

У рамках проекту розробки системи для моделювання класифікації регіонів за показниками діджиталізації особлива увага приділяється функціональності графічного інтерфейсу. Хоча до дизайну інтерфейсу немає високих вимог, важливо забезпечити, щоб він був інтуїтивно зрозумілим та ефективним у використанні.

Головна мета графічного інтерфейсу полягає у забезпеченні зручності роботи з аналітичною моделлю. Користувач повинен мати можливість не тільки взаємодіяти з моделлю в її стандартному вигляді, але й мати змогу редагувати параметри моделі або логіку розрахунків без необхідності звертатися до розробника. Це надасть кінцевим користувачам більшу

гнучкість та контроль над процесом аналізу, дозволяючи їм адаптувати систему під змінювані умови або особливі вимоги дослідження.

Графічний інтерфейс системи буде створено за допомогою JavaFX, фреймворка для розробки багатофункціональних клієнтських додатків на Java. Використання JavaFX обґрунтовано декількома важливими перевагами.

JavaFX забезпечує широкий спектр можливостей для створення інтерактивних інтерфейсів користувача, що є особливо важливим для проекту, де необхідно ефективно взаємодіяти з складними аналітичними моделями. Фреймворк надає широкі можливості для створення графічних елементів, у тому числі відображення 2D та 3D графіків (рис. 2.4), що може бути корисним для аналізу результатів роботи системи. Це дозволяє створити зручний та інформативний інтерфейс для користувачів.

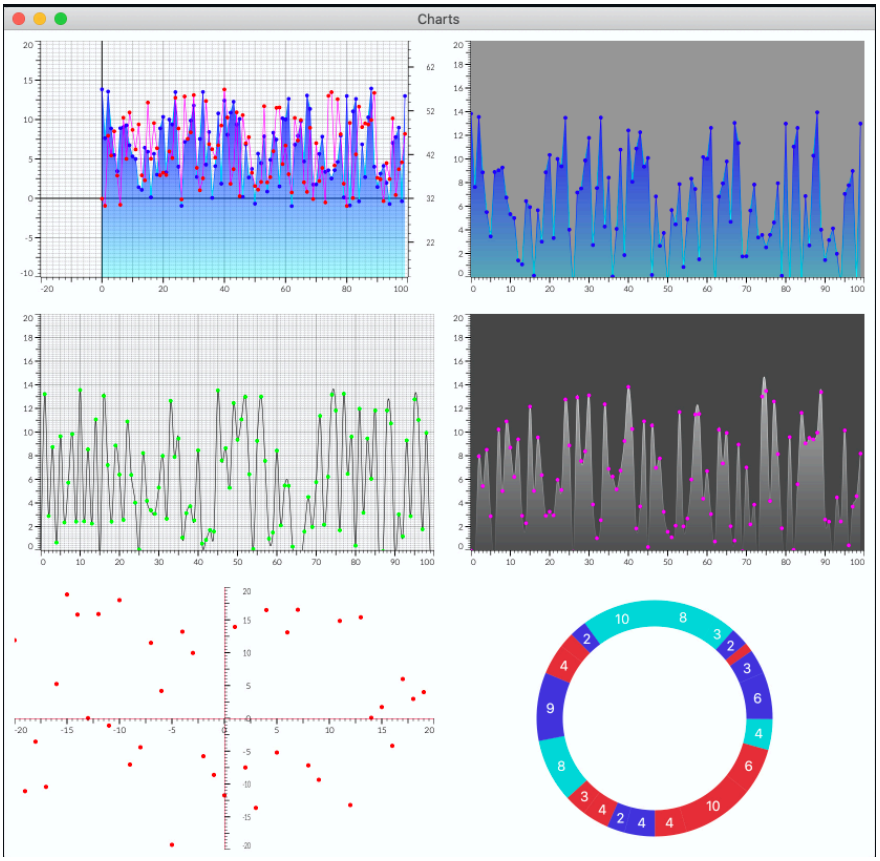

Рисунок 2.4 – Можливості JavaFX у побудові графіків

JavaFX можна легко інтегрувати з іншими Java фреймворками, що значно спрощує розробку системи. Ця особливість JavaFX дозволяє ефективно поєднати бібліотеку FuzzyLite з візуальними компонентами, які

бачить користувач. Завдяки цій інтеграції, можна безпосередньо відображати результати роботи нечіткої логіки на графічному інтерфейсі, забезпечуючи користувачам інтуїтивно зрозуміле та зручне управління параметрами моделі. Таким чином, JavaFX не тільки полегшує реалізацію багатофункціональних інтерфейсів, але й сприяє глибшій інтеграції розрахункових модулів з користувацьким середовищем, забезпечуючи ефективну взаємодію між логікою програми та її візуалізацією.

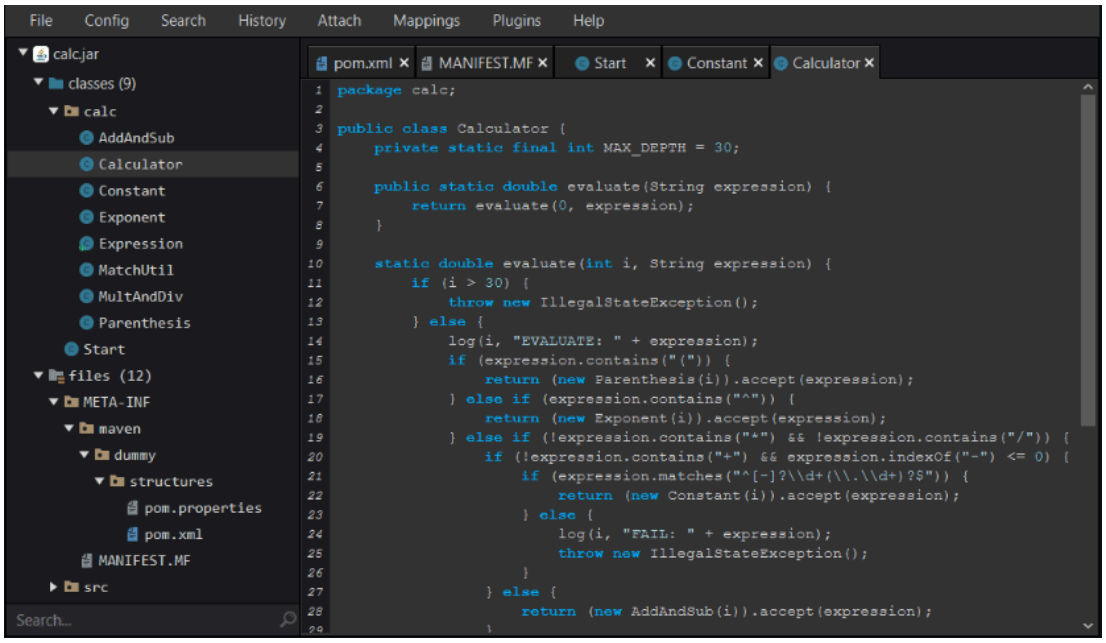

Рисунок 2.4 – Приклад інтерфейсу, створеного за допомогою JavaFX

Також JavaFX є доволі простим у використанні для розробників, що істотно спрощує процес розробки інтерфейсу. Завдяки багатофункціональному API та широкому набору вбудованих компонентів, розробники можуть швидше створювати комплексні графічні інтерфейси, значно скорочуючи час, необхідний для розробки. Це дозволяє не тільки зберегти час на етапі створення інтерфейсу, але й гнучко реагувати на зміни вимог до системи, швидко вносячи необхідні корективи до структури інтерфейсу.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Система CRUD операцій та математичний апарат

Оскільки система правил та набір змінних можуть змінюватися з часом, надзвичайно важливо розробити систему з можливістю легкого додавання, редагування та видалення цих сутностей. Це забезпечує гнучкість і адаптивність системи до змінних вимог та умов експлуатації. З цієї причини, раціонально в першу чергу розробити функціонал для CRUD (Create, Read, Update, Delete) операцій. Впровадження такого функціоналу дозволить швидко та ефективно керувати компонентами системи, що є критично важливим для підтримки її відповідності до актуальних аналітичних завдань.

Початково необхідний набір правил можна додати до готової системи, використовуючи саме цей функціонал. Реалізація можливостей CRUD для управління правилами та змінними дозволить ввести початкові дані безпосередньо через систему, забезпечуючи простий та інтуїтивний спосіб налаштування. Такий підхід забезпечить більшу гнучкість та зручність при налаштуванні системи під конкретні потреби, надаючи можливість швидко адаптувати набір правил до специфічних вимог та умов використання.

Дана система працює з двома ключовими сутностями: змінними та наборами правил. Кожна з цих сутностей має свою специфічну модель, що детально відображено на рис. 3.5.

Модель змінної включає кілька основних елементів, які забезпечують її функціональність:

- Назва змінної: ідентифікує змінну в системі.
- Функція належності: визначає, як вхідні дані відносяться до можливих значень цієї змінної.
- Діапазон даних: встановлює межі, в яких можуть коливатися значення змінної.
- Можливі значення: специфікує конкретні значення, які може приймати змінна.

Модель набору правил складається з наступних компонентів:

- Назва набору правил: дає унікальне ім'я для групи правил.
- Опис алгоритмів кон'юнкції, диз'юнкції, імплікації, та активації: визначає логічні операції, що використовуються для виведення висновків з даних.
- Масив правил: зберігає правила, описані в синтаксисі FuzzyLite, що дозволяє системі обробляти вхідні дані та приймати рішення.

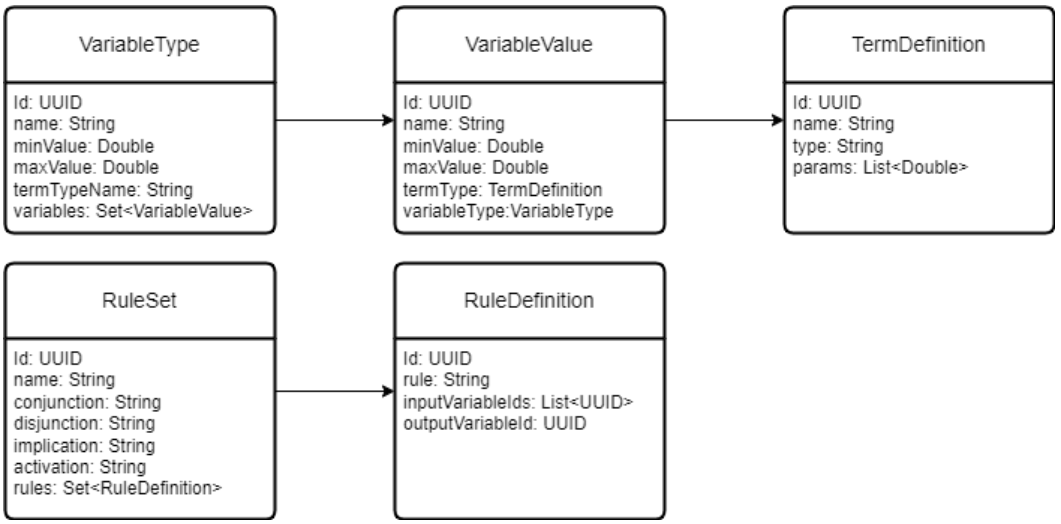

Рисунок 3.5 – Структура існуючих сутностей

У нашій системі всі дані сутностей, таких як змінні та правила, зберігаються у вигляді локальних файлів. Збереження даних здійснюється за допомогою механізму серіалізації, що дозволяє ефективно управляти структурованими даними. Для роботи з файлами були розроблені спеціалізовані класи: FileStorage та AbstractStorage.

Клас FileStorage відповідає за взаємодію з файловими потоками та реалізує функціонал збереження даних. Використання цього класу забезпечує надійне зберігання інформації у вигляді файлів на локальному диску, що дозволяє системі швидко відновлювати стан сутностей при повторних запусках.

AbstractStorage є абстрактним класом, який надає базовий функціонал для обробки даних сутностей. Цей клас дозволяє нащадкам працювати з представленнями сутностей, мінімізуючи необхідність безпосереднього

взаємодії з файлами. Такий підхід сприяє абстракції та вищому рівню управління даними, уникаючи залежності бізнес-логіки від конкретних методів зберігання даних.

Всі існуючі дані сутностей надходять до класу Analyser, який відповідає за інтеграцію та використання функціоналу фреймворку FuzzyLite. Цей клас грає ключову роль у підготовці та обробці даних для використання в рамках нечіткої логіки. Основне завдання Analyser полягає у підготовці представлень змінних та правил, щоб вони могли бути використані в процесі обчислення.

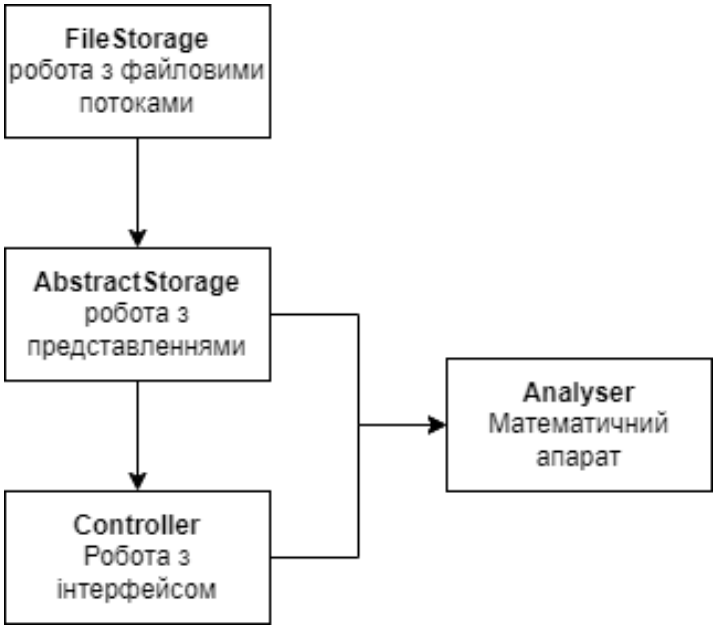

Рисунок 3.6 – Архітектура системи

Змінні в системі ієрархічно сортуються згідно з їх роллю в аналітичному процесі — від змінних для вхідних даних до змінної результату. Таке сортування базується на тому, як змінні використовуються у створених правилах, забезпечуючи логічну послідовність у процесах обчислень.

Після організації змінних та наборів правил, вони ініціалізуються у логічному ядрі бібліотеки FuzzyLite. Це дозволяє системі ефективно керувати обчислювальними процесами на основі нечіткої логіки. На графічний інтерфейс виводиться список існуючих вхідних змінних, де користувач може ввести їх значення. Після введення значень відбувається розрахунок результату, який базується на визначених параметрах та правилах.

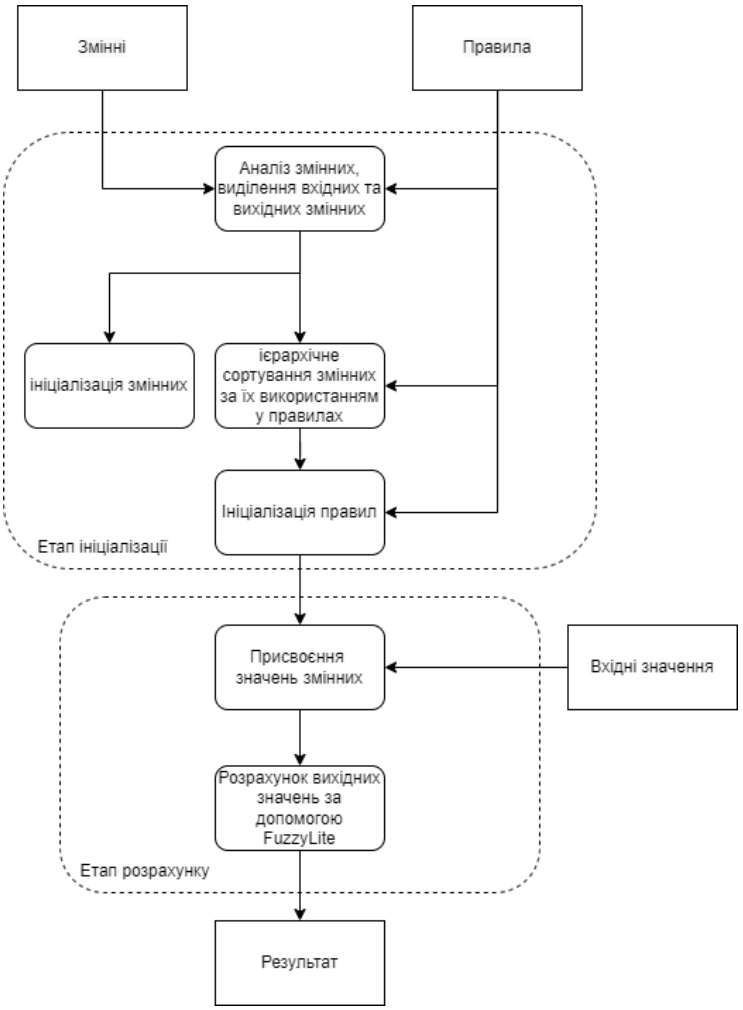

Рисунок 3.7 – Принцип роботи з апаратом нечіткої логіки

3.2 Графічний інтерфейс

Інтерфейс системи складається з п’яти вікон (рис. 3.8–3.12). Початкове вікно містить у собі список вхідних змінних, редактор їх значень, вікно з результатом обчислень. Також є дві кнопки, що відкривають вікна для редагування змінних або наборів правил.

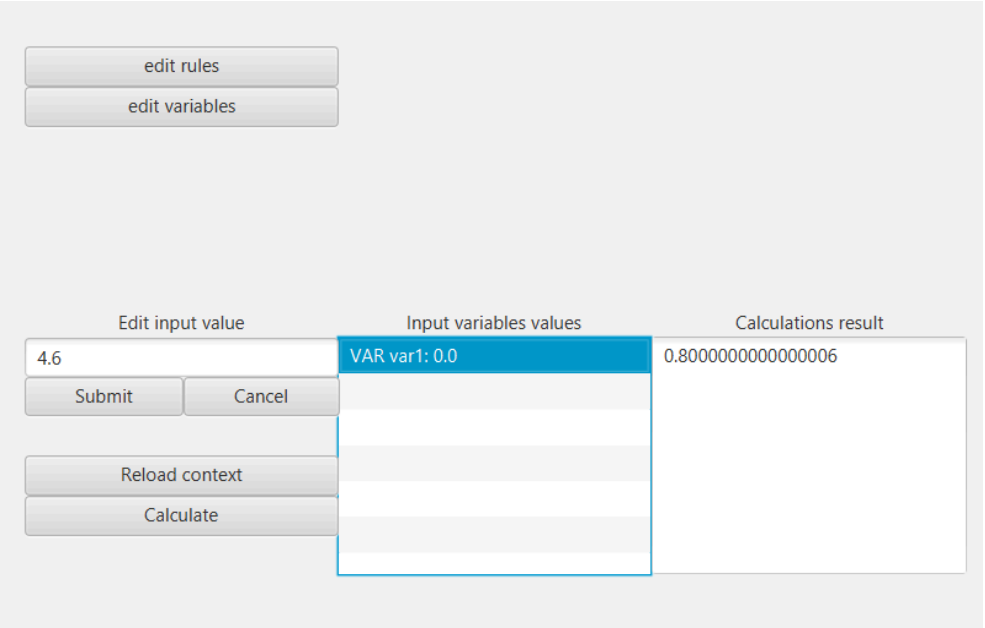

Рисунок 3.8 – Головне вікно

У вікні редагування наборів правил є список існуючих наборів правил, редактор для створення, редагування та видалення існуючих наборів, а також кнопка, що відкриває редактор існуючих правил для відповідного набору.

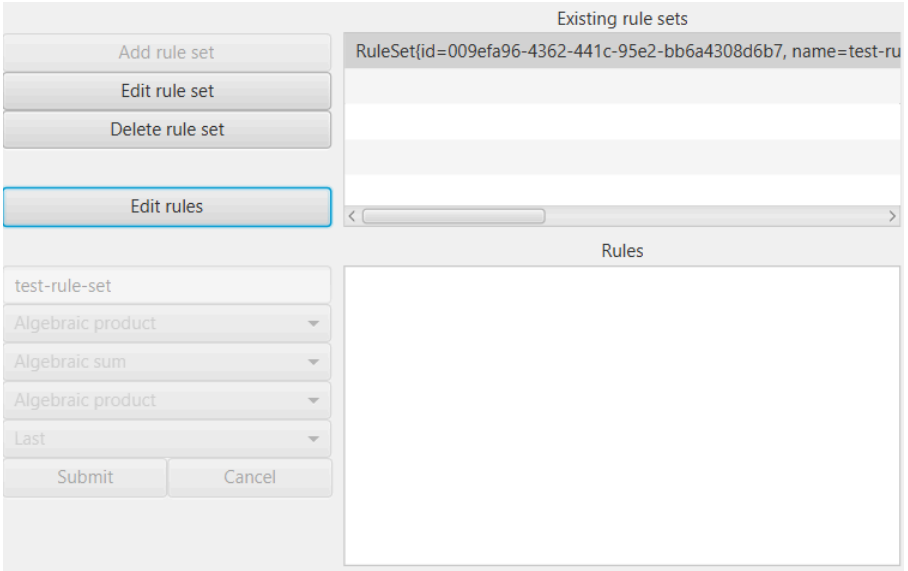

Рисунок 3.9 – Редактор наборів правил

Редактор правил містить у собі список існуючих правил, відповідних до обраного набору. Тут присутній конструктор нових правил, вікно для перегляду синтаксису створених правил, та кнопки для виконання CRUD операцій над правилами.

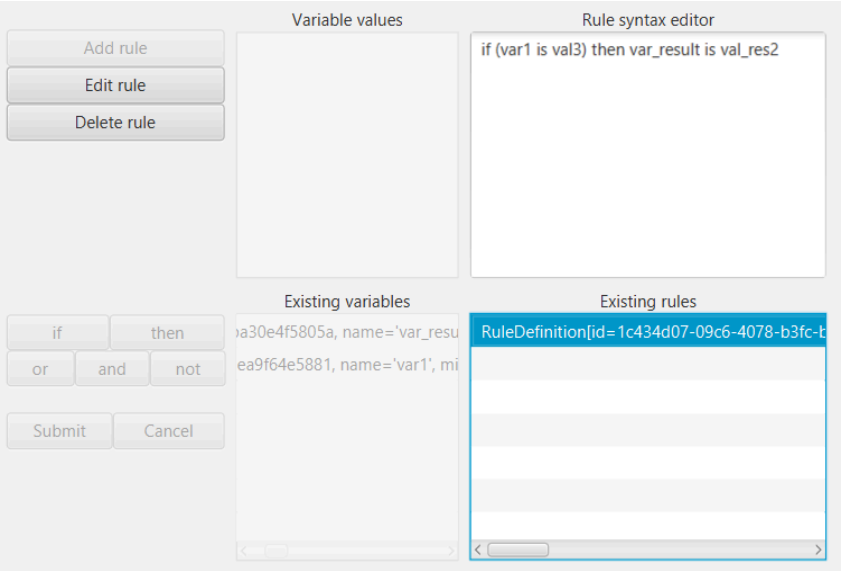

Рисунок 3.10 – Редактор правил

Редактор змінних аналогічний до редактору правил, містить у собі функціонал для CRUD операцій над змінними, список існуючих змінних та кнопку, що відкриває редактор можливих значень змінної.

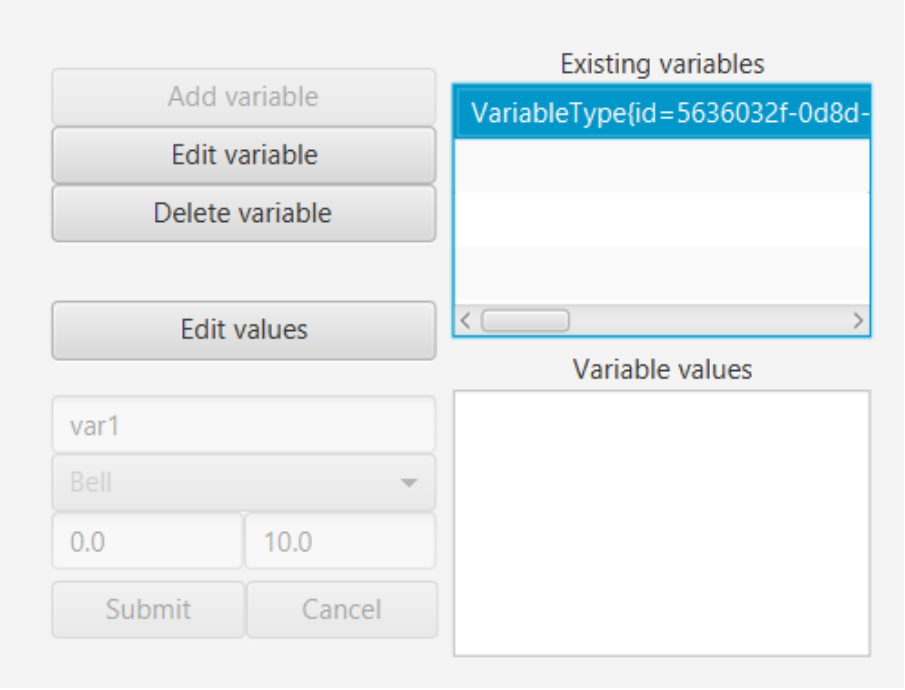

Рисунок 3.11 – Редактор змінних

Редактор значень відповідає за редагування значень відповідної змінної та перегляд вже існуючих значень.

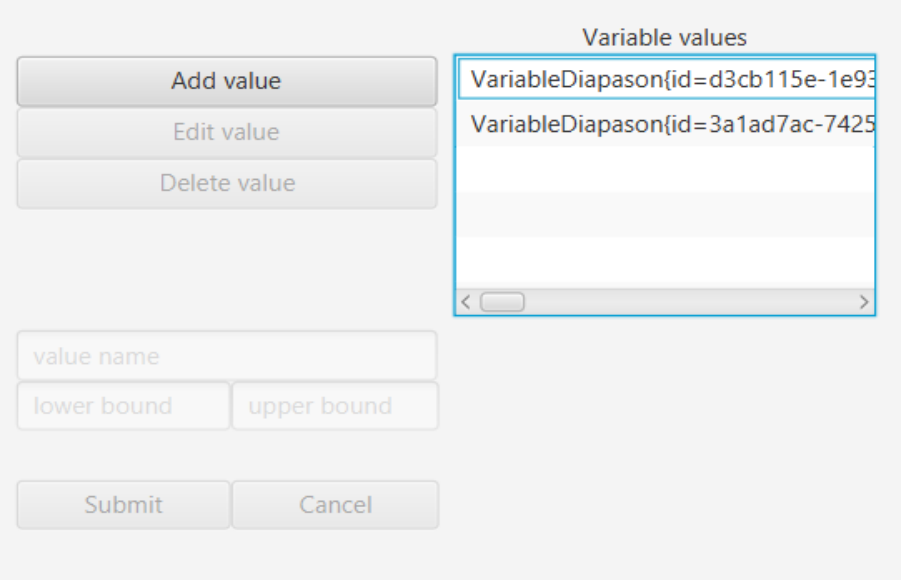

Рисунок 3.12 – Редактор значень змінних

3.3 Можливості системи в області нечіткої логіки

Реалізована система надає широкі можливості для роботи з нечіткою логікою, використовуючи функціонал бібліотеки FuzzyLite. На разі система підтримує такі терми:

- Triangle (Трикутник)
- Gaussian (Гаусівський)
- Bell (Дзвін)
- Spike (Шип)
- Rectangle (Прямокутник)

На рис. 3.13 зеленим кольором виділені терми, що підтримуються системою на даний момент [4].

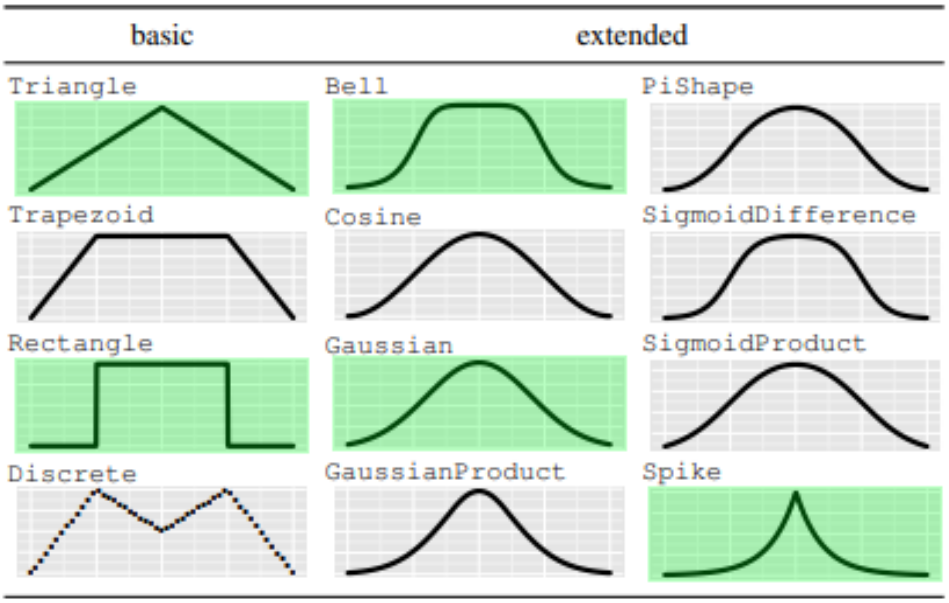

Рисунок 3.13 – Підтримувані терми

Система також підтримує ряд S-норм та T-норм. Підтримувані S-норми включають:

- Algebraic sum (Алгебраїчна сума)
- Maximum (Максимум)
- Bounded difference (Обмежена різниця)

Для обчислення перетинів нечітких множин система підтримує наступні T-норми:

- Algebraic product (Алгебраїчний добуток)
- Minimum (Мінімум)
- Bounded sum (Обмежена сума)

На рис. 3.14 зеленим кольором виділені норми, що підтримуються системою на даний момент [4]. Чорним кольором позначено наближення значень до нуля, а білим - до одиниці.

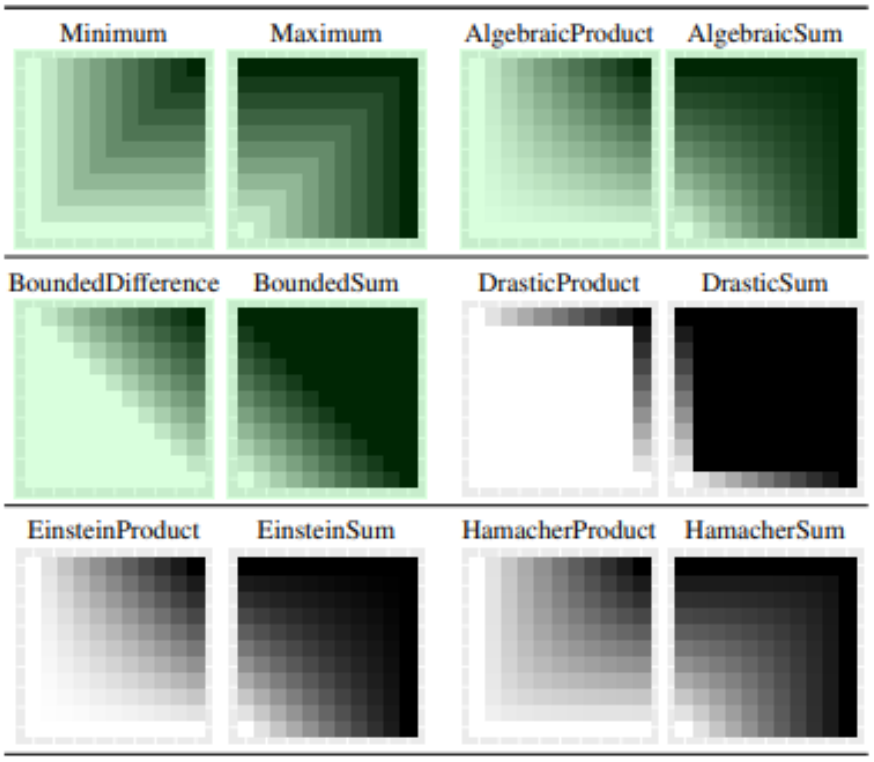

Рисунок 3.14 – Підтримувані S-норми та T-норми

Система підтримує побудову ієрархічної структури правил та автоматично розпізнає, які правила та змінні належать до якого рівня ієрархії. Також частково підтримується редагування правил з прямим використанням синтаксису правил FuzzyLite

3.4 Тестування системи

Для тестування системи були використані показники субіндексів усіх областей України за 2022 рік. Кожен з індексів представлено змінною, яка має п'ять значень: низький, нижче середнього, середній, вище середнього та високий.

Область	Інституційна спроможність	Розвиток інтернету	Розвиток ЦНАП	Режим «без паперів»	Цифрова освіта	Візитівка області	Проникнення базових електронних послуг	Галузева цифрова трансформаці	Класт	Показник Індексу
Черкаська	0,744	0,771	0,855	0,803	0,312	0,6	0,759	0,387	1	0,716
Івано-Франківська	0,81	0,896	0,853	0,162	0,124	0,8	0,78	0,338	1	0,683
Хмельницька	0,861	0,609	0,719	0,653	0,165	0,4	0,596	0,458	1	0,61
Київська	0,71	0,377	0,824	0,711	0,258	1	0,61	0,21	1	0,588
Харківська	0,794	0,615	0,703	0,553	0,152	0,48	0,544	0,325	1	0,571
Вінницька	0,9	0,743	0,852	0,894	0,778	0,4	0,696	0,721	2	0,769
Волинська	0,6	0,733	0,911	0,928	0,256	0	0,772	0,716	2	0,72
Житомирська	0,38	0,909	0,895	0,859	0,349	0,4	0,646	0,403	2	0,692
Дніпропетровська	1	0,993	0,972	0,952	0,694	1	0,754	0,924	3	0,916
Тернопільська	1	0,992	0,83	0,933	0,718	1	0,834	0,999	3	0,91
Одеська	0,9	0,841	0,705	0,958	0,702	1	0,877	0,808	3	0,836
Полтавська	0,8	0,894	0,905	0,967	0,88	0,6	0,737	0,56	3	0,814
Львівська	0,89	0,833	0,822	0,924	0,562	0,8	0,68	0,82	3	0,799
Рівненська	1	0,907	0,875	0,891	0,195	0,8	0,705	0,573	3	0,794
Закарпатська	0,571	0,868	0,841	0,85	0,372	1	0,766	0,504	3	0,756
Чернівецька	0,37	0,633	0,852	0,39	0,13	0,4	0,596	0,211	4	0,54
Кіровоградська	0,213	0,388	0,664	0,531	0,163	0	0,614	0,154	4	0,431
Луганська	0,093	0,1	0,743	0,459	0,262	0,4	0,623	0,224	4	0,404
Запорізька	0,331	0,405	0,385	0,063	0,155	0,4	0,595	0,261	4	0,37
Сумська	0,44	0,471	0,763	0,569	0,605	0	0,622	0,321	5	0,534
Чернігівська	0,493	0,612	0,629	0,596	0,267	0,4	0,4	0,507	5	0,522
Херсонська	0,441	0,428	0,787	0,693	0,589	0,4	0,438	0,066	5	0,5
Донецька	0,325	0,252	0,631	0,527	0,396	0,68	0,596	0,345	5	0,469
Миколаївська	0,11	0,53	0,467	0,497	0,509	0,6	0,427	0,3	5	0,431

Рисунок 3.15 – Вхідні значення [8]

На основі цих даних та існуючих субіндексів діджиталізації було ініціалізовано список вхідних змінних та змінну результату (рис. 3.16).

Рисунок 3.16 – Ініціалізовані змінні та їх функції належності

Система правил складається з наборів правил типу "if var_1 is low then result is low" для кожного значення кожного субіндексу (рис. 3.17).

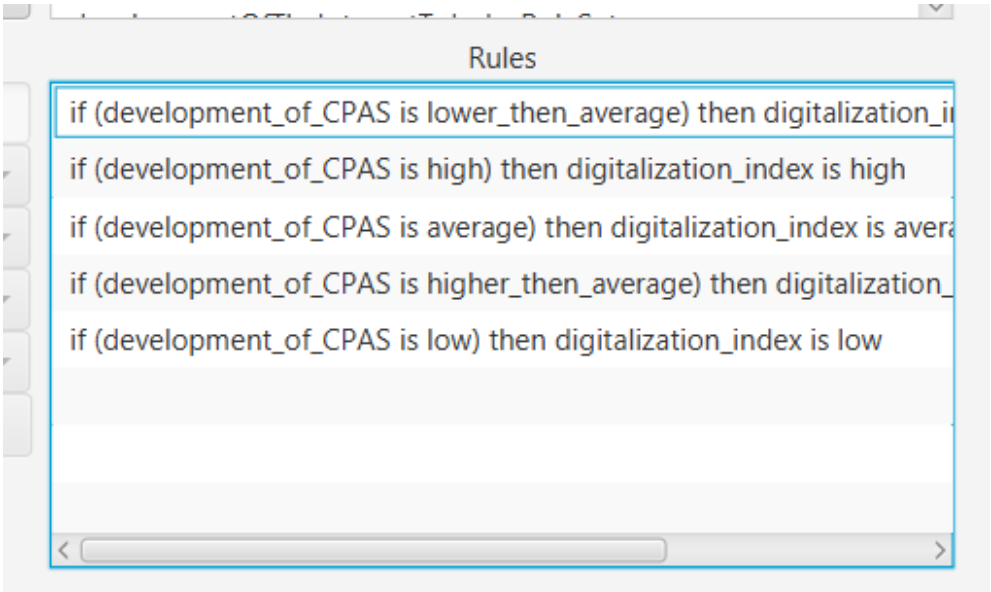

Рисунок 3.17 – Приклад блоку правил

Це дозволяє системі виконувати аналіз на основі нечіткої логіки, враховуючи різні комбінації значень змінних.

Таблиця 3.1 – порівняння отриманих та фактичних значень

Область	**Фактичне значення індексу**	**Отримане значення індексу**
Черкаська	0,716	0,589
Івано-Франківська	0,683	0,498
Хмельницька	0,61	0,534
Київська	0,588	0,57
Харківська	0,571	0,512
Вінницька	0,769	0,574
Волинська	0,72	0,504
Житомирська	0,692	0,574
Дніпропетровська	0,916	0,64
Тернопільська	0,91	0,652
Одеська	0,836	0,653

Продовження табл. 3.1

Область	**Фактичне значення індексу**	**Отримане значення індексу**
Полтавська	0,814	0,614
Львівська	0,799	0,628
Рівненська	0,794	0,585
Закарпатська	0,756	0,619
Чернівецька	0,54	0,505
Кіровоградська	0,431	0,503
Луганська	0,404	0,511
Запорізька	0,37	0,502
Сумська	0,534	0,505
Чернігівська	0,522	0,574
Херсонська	0,5	0,575
Донецька	0,469	0,579
Миколаївська	0,431	0,515

Таблиця 3.2 – класифікація регіонів за показниками діджиталізації

Область	**Класифікація за фактичним індексом**	**Класифікація за отриманим індексом**
Черкаська	середній	середній
Івано-Франківська	середній	середній
Хмельницька	середній	середній
Київська	середній	середній
Харківська	середній	середній
Вінницька	вище середнього	вище середнього
Волинська	вище середнього	середній

Продовження табл. 3.2

Область	**Класифікація за фактичним індексом**	**Класифікація за отриманим індексом**
Житомирська	вище середнього	середній
Дніпропетровська	високий	вище середнього
Тернопільська	високий	вище середнього
Одеська	високий	вище середнього
Полтавська	високий	вище середнього
Львівська	високий	вище середнього
Рівненська	високий	вище середнього
Закарпатська	високий	вище середнього
Чернівецька	низький	нижче середнього
Кіровоградська	низький	нижче середнього
Луганська	низький	нижче середнього
Запорізька	низький	нижче середнього
Сумська	нижче середнього	нижче середнього
Чернігівська	нижче середнього	середній
Херсонська	нижче середнього	нижче середнього
Донецька	нижче середнього	середній
Миколаївська	нижче середнього	нижче середнього

У ході тестування були отримані результати для кожної проаналізованої області (табл. 3.1). Результати виявилися відмінними від фактичних індексів, що дуже сильно вплинуло на класифікацію (табл. 3.2). Однак, отримані індекси діджиталізації демонструють кореляцію з реальними показниками [8]. Наприклад, області з високими показниками субіндексів (вище середнього та високий) отримали відповідні високі (відносно інших отриманих результатів) результати в системі. Області з низькими

показниками субіндексів (нижче середнього та низький) мали відповідні низькі результати.

Це підтверджує, що розроблена система здатна адекватно відображати реальний стан діджиталізації в регіонах, враховуючи різні аспекти та показники, хоча отримані результати й не є правдивими.

Основною причиною розбіжності між результатами системи та реальними показниками є недосконаліть існуючої системи правил. Поточні правила не враховують усіх особливостей та нюансів кожного субіндексу, що впливає на точність аналізу. Також на точність розрахунків вплинула неможливість обрати алгоритми агрегації та дефазифікації для вихідних змінних у системі. На даний момент у системі використовується агрегація вихідних значень за S-нормою максимума, а дефазифікація – за центроїдом. Це також пояснює наближення отриманих даних до середнього значення, що спричинило таку похибку.

Для досягнення більш точних та надійних результатів необхідно допрацювати систему правил, зробивши їх більш детальними та специфічними для кожного субіндексу. У подальшому також слід додати додаткові функції до системи, що дозволить покращити її гнучкість та адаптивність до різних умов. Це забезпечить більш точний аналіз та класифікацію регіонів за показниками діджиталізації, враховуючи їхні унікальні особливості та потреби.

ВИСНОВОК

У ході виконання кваліфікаційної роботи було виконано такі завдання:

1. Проаналізовано доступні технології для вирішення задачі. У якості математичного апарату обрано апарат нечіткого логічного виведення.
2. Розроблено інформаційну систему, здатну вирішити задачу класифікації регіонів за показниками діджиталізації.
3. Враховано необхідність у легкому внесенні змін до створюваної системи.
4. Система була забезпечена потужним математичним апаратом, здатним працювати з складними структурами даних.
5. Протестовано систему та отримано результати розрахунку показників діджиталізації.
6. Отримані результати було проаналізовано.

Надалі планується доповнення системи додатковим необхідним функціоналом, який забезпечить більш широкі можливості для аналізу та виведення висновків. Одним із ключових етапів подальшої роботи стане занесення системи правил та змінних, які необхідні для розрахунку індексів діджиталізації регіонів. Це дозволить системі повноцінно функціонувати в заданій області, забезпечуючи точність та об'єктивність аналітичних розрахунків.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Зайченко, Катерина С. "Діджиталізація економік та суспільства: світові тенденції." Національний університет "Одеська політехніка", Одеса, Україна.

2. Zadeh, L.A.: Fuzzy Sets: Information and Control; Vol. 8, No. 3, 1965 – 338–353 с.

3. Cingolani, Pablo. "jFuzzyLogic: A Robust and Flexible Fuzzy-Logic Inference System Language Implementation." International Journal of Computational Intelligence Systems, Vol. 6, Supplement 1 (2013), 61-75

4. Rada-Vilela, Juan, PhD. "The FuzzyLite Libraries for Fuzzy Logic Control." FuzzyLite Limited, Wellington, New Zealand

5. FuzzyJ ToolKit Documentation [Електронний ресурс]. URL: http://rorchard.github.io/FuzzyJ/

6. Кравець, П., Киркало, Р. "Системи прийняття рішень з нечіткою логікою." Кафедра інформаційних систем та мереж, Національний університет “Львівська політехніка”, Львів, Україна.

7. Сайт “OpenJFX.io”, документація щодо бібліотеки JavaFX [Електронний ресурс], URL: https://fxdocs.github.io/docs/html5 (Дата звернення 19.05.2024)

8. Барченко, Н., Любчак, В., Великодний, Д. “ВИБІР МЕТОДА КЛАСТЕРИЗАЦІЇ З МЕТОЮ АНАЛІЗУ ПОКАЗНИКІВ ЦИФРОВИХ ТРАНСФОРМАЦІЙ РЕГІОНІВ УКРАЇНИ”, Інформаційні технології та суспільство. Випуск 2 (8). 2023

9. Кузнєцов, Ю.Б.: Нечіткі системи в прикладах і задачах: Видавництво ХНУ імені В. Н. Каразіна; Харків, 2006 – 200 с.

10. Васильєва, Т.А., Пічкур, В.Д.: Нечітке моделювання в економіці: КНЕУ; Київ, 2002 – 256 с.

11. Pedrycz, W., Gomide, F.: An Introduction to Fuzzy Sets: Analysis and Design: MIT Press; 1st edition (September 17, 1998) – 560 с.

12. Красільников, О.А., та ін.: Теорія нечітких множин і нечітка логіка: Наукова думка; Київ, 2003 – 248 с.

13. Pedrycz, W., Gomide, F.: An Introduction to Fuzzy Sets: Analysis and Design: MIT Press; 1st edition (September 17, 1998) – 560 с.

14. Negnevitsky, M.: Artificial Intelligence: A Guide to Intelligent Systems: Pearson Education; 3rd edition (April 6, 2011) – 504 с.

15. Klir, G.J., Yuan, B.: Fuzzy Sets and Fuzzy Logic: Theory and Applications: Prentice Hall; 1st edition (January 1, 1995) – 592 с.

16. Mehdi Rahmani-Andebili: Applications of Fuzzy Logic in Planning and Operation of Smart Grids (Power Systems): Springer; 1st ed. 2021 edition (May 25, 2021) – 236 с.

17. Jang, J.-S.R., Sun, C.-T., Mizutani, E.: Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence: Prentice Hall; 1st edition (January 1, 1997) – 614 с.

18. Дубровська, Н.І.: Основи теорії нечітких множин: Академперіодика; Київ, 2005 – 280 с.

19. Mendel, Jerry M.: Uncertain Rule-Based Fuzzy Systems: Introduction and New Directions: Springer; 2nd edition (September 28, 2017) – 684 с.

20. Bonissone, Piero P., Subbu, Rajendra, Lizzi, Roberto: A Framework for Decision Making with Incomplete, Uncertain, and Linguistic Information: Springer; 1st edition (January 1, 2010) – 297 с.

21. Nguyen, Hung T., Walker, Elbert A.: A First Course in Fuzzy Logic: Chapman and Hall/CRC; 3rd edition (November 15, 2005) – 440 с.

22. Lee, Chuen-Chien: Fuzzy Logic in Control Systems: Fuzzy Logic Controller—Part I & II: IEEE Transactions on Systems, Man, and Cybernetics; Vol. 20, No. 2, 1990 – 404–435 с.

23. Yager, Ronald R., Zadeh, Lotfi A.: An Introduction to Fuzzy Logic Applications in Intelligent Systems: Springer; 1st edition (November 1, 1992) – 356 с

24. Kosko, Bart: Fuzzy Thinking: The New Science of Fuzzy Logic:

Hyperion; Reprint edition (December 1, 1994) – 336 с.

25. Bonissone, Piero P., Subbu, Rajendra, Lizzi, Roberto: A Framework for Decision Making with Incomplete, Uncertain, and Linguistic Information: Springer; 1st edition (January 1, 2010) – 297 с.

26. Петренко, П.П.: Системи підтримки прийняття рішень на основі нечіткої логіки: Видавництво ДНУ; Дніпро, 2013 – 280 с.

27. Олексієнко, А.П.: Застосування нечіткої логіки у вирішенні економічних задач: Видавництво ХНУ імені В.Н. Каразіна; Харків, 2016 – 280 с.

28. Лазаренко, І.В.: Нечіткі моделі в системах керування: Видавництво ЧНУ імені Юрія Федьковича; Чернівці, 2018 – 300 с.

29. Олійник, В.В.: Принципи та методи нечіткої логіки: Видавництво "Вища школа"; Київ, 2010 – 280 с.

30. Драгомир, О.В.: Нечітке моделювання в економіці: Видавництво КНЕУ; Київ, 2015 – 260 с.

ДОДАТОК

VariableType.class

```
public class VariableType extends ModelBase {

    private final UUID id;
    private final String name;
    private Double minValue;
    private Double maxValue;
    private String termTypeName;
    private Set<VariableValue> variables = new HashSet<>();

    public VariableType(final String name, final Double minValue, final Double
maxValue, final String term) {
        this.id = UUID.randomUUID();
        this.name = name;
        this.minValue = minValue;
        this.maxValue = maxValue;
        this.termTypeName = term;
    }

    public VariableType(final UUID id, final String name, final Double
minValue, final Double maxValue, final String term) {
        this.id = id;
        this.name = name;
        this.minValue = minValue;
        this.maxValue = maxValue;
        this.termTypeName = term;
    }

    public UUID getId() {
        return this.id;
    }

    public String getName() {
        return this.name;
    }

    public Double getMinValue() {
        return this.minValue;
    }

    public void setMinValue(final Double minValue) {
        this.minValue = minValue;
    }

    public Double getMaxValue() {
        return this.maxValue;
    }

    public void setMaxValue(final Double maxValue) {
        this.maxValue = maxValue;
    }

    public String getTermTypeName() {
        return this.termTypeName;
    }

    public void setTermTypeName(final String termTypeName) {
        this.termTypeName = termTypeName;
    }

    public Set<VariableValue> getVariables() {
        return this.variables;
```

```
    }

    public void setVariables(final Set<VariableValue> variables) {
        this.variables = variables;
    }

    public void addVariables(final Set<VariableValue> variables) {
        this.variables.addAll(variables);
    }

    public void removeVariables(final Set<VariableValue> variables) {
        this.variables.removeAll(variables);
    }

    @Override
    public boolean equals(final Object o) {
        if (this == o) return true;
        if (!(o instanceof final VariableType that)) return false;
        return Objects.equals(getId(), that.getId())
            && Objects.equals(getName(), that.getName())
            && Objects.equals(getMinValue(), that.getMinValue())
            && Objects.equals(getMaxValue(), that.getMaxValue())
            && Objects.equals(getTermTypeName(), that.getTermTypeName())
            && Objects.equals(getVariables(), that.getVariables());
    }

    @Override
    public int hashCode() {
        return Objects.hash(getId(), getName(), getMinValue(), getMaxValue(),
getTermTypeName(), getVariables());
    }

    @Override
    public String toString() {
        return "VariableType{" +
            "id=" + this.id +
            ", name='" + this.name + '\'' +
            ", minValue=" + this.minValue +
            ", maxValue=" + this.maxValue +
            ", term=" + this.termTypeName +
            ", variables=" + this.variables +
            '}';
    }
}

VariableValue.class
public class VariableValue extends ModelBase {

    private final UUID id;
    private final String name;
    private final VariableType type;
    private final TermDefinition term;
    private Double minValue;
    private Double maxValue;

    public VariableValue(final String name, final TermDefinition term, final
VariableType type, final Double minValue, final Double maxValue) {
        this.id = UUID.randomUUID();
        this.name = name;
        this.term = term;
        this.type = type;
        this.minValue = minValue;
        this.maxValue = maxValue;
    }
```

```
    public VariableValue(final UUID id, final String name, final TermDefinition
term, final VariableType type, final Double minValue, final Double maxValue) {
        this.id = id;
        this.name = name;
        this.term = term;
        this.type = type;
        this.minValue = minValue;
        this.maxValue = maxValue;
    }

    public UUID getId() {
        return this.id;
    }

    public String getName() {
        return this.name;
    }

    public VariableType getType() {
        return this.type;
    }

    public Double getMinValue() {
        return this.minValue;
    }

    public Double getMaxValue() {
        return this.maxValue;
    }

    public TermDefinition getTerm() {
        return this.term;
    }

    public void setMinValue(final Double minValue) {
        this.minValue = minValue;
    }

    public void setMaxValue(final Double maxValue) {
        this.maxValue = maxValue;
    }

    @Override
    public boolean equals(final Object o) {
        if (this == o) return true;
        if (!(o instanceof final VariableValue variable)) return false;
        return Objects.equals(getId(), variable.getId())
            && Objects.equals(getName(), variable.getName())
            && Objects.equals(getTerm(), variable.getTerm())
            && Objects.equals(getType().getName(),
variable.getType().getName())
            && Objects.equals(getMinValue(), variable.getMinValue())
            && Objects.equals(getMaxValue(), variable.getMaxValue());
    }

    @Override
    public int hashCode() {
        return Objects.hash(getId(), getName(), getTerm(), getType().getName(),
getMinValue(), getMaxValue());
    }

    @Override
    public String toString() {
```

```
        return "VariableDiapason{" +
            "id=" + this.id +
            ", name='" + this.name + '\'' +
            ", term=" + this.term.toString() +
            ", type=" + this.type.getName() +
            ", minValue=" + this.minValue +
            ", maxValue=" + this.maxValue +
            '}';
    }
}
```

```
TermDefinition.class
public class TermDefinition extends ModelBase {

    private final UUID id;
    private final String name;
    private final String type;
    private final List<Double> params = new ArrayList<>();

    public TermDefinition(final UUID id, final String type, final String name)
{
        this.id = id;
        this.type = type;
        this.name = name;
    }

    @Override
    public UUID getId() {
        return this.id;
    }

    public String getName() {
        return this.name;
    }

    public String getType() {
        return this.type;
    }

    public List<Double> getParams() {
        return this.params;
    }

    public void addParam(final Integer index, final Double param) {
        this.params.add(index, param);
    }

    @Override
    public boolean equals(final Object o) {
        if (this == o) return true;
        if (!(o instanceof final TermDefinition that)) return false;
        return Objects.equals(getId(), that.getId())
            && Objects.equals(getName(), that.getName())
            && Objects.equals(getType(), that.getType())
            && Objects.equals(getParams(), that.getParams());
    }

    @Override
    public int hashCode() {
        return Objects.hash(getId(), getName(), getType(), getParams());
    }

    @Override
    public String toString() {
```

```
        return "TermDefinition{" +
            "id=" + this.id +
            ", name='" + this.name + '\'' +
            ", type='" + this.type + '\'' +
            ", params=" + this.params +
            '}';
    }
}


RuleSet.class
public class RuleSet extends ModelBase {

    private final UUID id;
    private String name;
    private String conjunction;
    private String disjunction;
    private String implication;
    private String activation;
    private Set<RuleDefinition> rules = new HashSet<>();

    public RuleSet(final String name, final String conjunction, final String
disjunction, final String implication, final String activation) {
        this.id = UUID.randomUUID();
        this.name = name;
        this.conjunction = conjunction;
        this.disjunction = disjunction;
        this.implication = implication;
        this.activation = activation;
    }
    public RuleSet(final UUID id, final String name, final String conjunction,
final String disjunction, final String implication, final String activation) {
        this.id = id;
        this.name = name;
        this.conjunction = conjunction;
        this.disjunction = disjunction;
        this.implication = implication;
        this.activation = activation;
    }

    public UUID getId() {
        return this.id;
    }

    public String getName() {
        return this.name;
    }

    public String getConjunction() {
        return this.conjunction;
    }

    public String getDisjunction() {
        return this.disjunction;
    }

    public String getImplication() {
        return this.implication;
    }

    public String getActivation() {
        return this.activation;
    }
```

```
    public Set<RuleDefinition> getRules() {
        return this.rules;
    }

    public void setName(final String name) {
        this.name = name;
    }

    public void setConjunction(final String conjunction) {
        this.conjunction = conjunction;
    }

    public void setDisjunction(final String disjunction) {
        this.disjunction = disjunction;
    }

    public void setImplication(final String implication) {
        this.implication = implication;
    }

    public void setActivation(final String activation) {
        this.activation = activation;
    }

    public void setRules(final Set<RuleDefinition> rules) {
        this.rules = rules;
    }

    public void addRules(final List<RuleDefinition> rules) {
        this.rules.addAll(rules);
    }

    public void removeRules(final List<RuleDefinition> rules) {
        rules.forEach(this.rules::remove);
    }

    @Override
    public boolean equals(final Object o) {
        if (this == o) return true;
        if (!(o instanceof final RuleSet ruleSet)) return false;
        return Objects.equals(getId(), ruleSet.getId())
            && Objects.equals(getName(), ruleSet.getName())
            && Objects.equals(getConjunction(), ruleSet.getConjunction())
            && Objects.equals(getDisjunction(), ruleSet.getDisjunction())
            && Objects.equals(getImplication(), ruleSet.getImplication())
            && Objects.equals(getActivation(), ruleSet.getActivation())
            && Objects.equals(getRules(), ruleSet.getRules());
    }

    @Override
    public int hashCode() {
        return Objects.hash(getId(), getName(), getConjunction(),
getDisjunction(), getImplication(), getActivation(), getRules());
    }

    @Override
    public String toString() {
        return "RuleSet{" +
            "id=" + this.id +
            ", name=" + this.name +
            ", conjunction=" + this.conjunction +
            ", disjunction=" + this.disjunction +
            ", implication=" + this.implication +
            ", activation=" + this.activation +
```

```
            ", rules=" + this.rules +
            '}';
    }
}

RuleDefinition.class
public record RuleDefinition(
    UUID id,
    String rule,
    List<UUID> inputVariableIds,
    UUID outputVariableId) implements Serializable {

    public RuleDefinition(final String rule, final List<UUID> inputIds, final
UUID outputId) {
        this(UUID.randomUUID(), rule, inputIds, outputId);
    }
}
FileStorage.class
public abstract class FileStorage<T extends ModelBase> {

    static final File STORAGE = new File("storage");
    protected final File subStorage;
    private final T emptyInstance;

    public static void preInit() {
        STORAGE.mkdir();
    }

    public List<T> readAll() throws IOException, ClassNotFoundException {
        if(subStorage.exists() && subStorage.isDirectory()) {
            if (!this.subStorage.exists()) {
                this.subStorage.mkdir();
            }
            ArrayList<T> entities = new ArrayList<>();
            for (File record : subStorage.listFiles()) {
                FileInputStream fis = new FileInputStream(record);
                ObjectInputStream oin = new ObjectInputStream(fis);
                T entity = (T) oin.readObject();
                entities.add(entity);
                fis.close();
                oin.close();
            }
            return entities;
        } return new ArrayList<T>();
    }

    public T read(T record) throws IOException, ClassNotFoundException {
        if(subStorage.exists() && subStorage.isDirectory()) {
            if (!this.subStorage.exists()) {
                this.subStorage.mkdir();
            }
            FileInputStream fis = new FileInputStream(getFileName(record));
            ObjectInputStream oin = new ObjectInputStream(fis);
            T entity = (T) oin.readObject();
            fis.close();
            oin.close();
            return entity;
        }
        return this.emptyInstance;
    }

    public void write(T record) throws IOException {
        FileOutputStream fos = new FileOutputStream(getFileName(record));
        ObjectOutputStream oos = new ObjectOutputStream(fos);
```

```
        oos.writeObject(record);
        oos.flush();
        oos.close();
    }

    public void delete(T record){
        File recordToDelete = new File(getFileName(record));
        if(recordToDelete.exists()) {
            recordToDelete.delete();
        }
    }

    public String getFileName(T record) {
        return
STORAGE.getName()+"//"+this.subStorage.getName()+"//"+record.getId().toString()
+".record";
    }

    protected FileStorage(final T emptyInstance, final String dir) {
        this.emptyInstance = emptyInstance;
        this.subStorage = new File(STORAGE.getName().concat("//").concat(dir));
    }
}

AbstractStorage.class
public abstract class AbstractStorage<T extends ModelBase> extends
FileStorage<T> {

    protected final HashMap<UUID, T> buffer;

    public void init() throws IOException, ClassNotFoundException {
        this.subStorage.mkdir();
        super.readAll().forEach(t -> this.buffer.put(t.getId(), t));
    }

    protected AbstractStorage(final T emptyInstance, final String dir) {
        super(emptyInstance, dir);
        this.buffer = new HashMap<>();
    }

    @Override
    public List<T> readAll() {
        return
Collections.unmodifiableList(this.buffer.values().stream().toList());
    }

    public boolean existsById(final UUID uuid) {
        return this.buffer.containsKey(uuid);
    }


    public T readById(final UUID uuid) {
        return this.buffer.get(uuid);
    }

    @Override
    public T read(final T record) {
        return this.buffer.get(record.getId());
    }

    @Override
    public void write(final T record) throws IOException {
        this.buffer.put(record.getId(), record);
        super.write(record);
```

```
    }

    @Override
    public void delete(final T record) {
        this.buffer.remove(record.getId());
        super.delete(record);
    }
}


RuleSetStorage.class
public class RuleSetStorage extends AbstractStorage<RuleSet> {

    public RuleSetStorage(final RuleSet emptyInstance, final String dir) {
        super(emptyInstance, dir);
    }
}

VariableTypeStorage.class
public class VariableTypeStorage extends AbstractStorage<VariableType> {

    public VariableTypeStorage(final VariableType emptyInstance, final String
dir) {
        super(emptyInstance, dir);
    }
}

Analyser.class
public class Analyser {

    private final Engine fuzzyLogicEngine;

    private final List<VariableType> inputVars = new ArrayList<>();
    private final List<VariableType> outputVars = new ArrayList<>();

    private final Set<InputVariable> initedInputVars = new HashSet<>();
    private final Set<OutputVariable> initedOutputVars = new HashSet<>();

    private final List<Set<UUID>> variableIdsByLevelList = new ArrayList<>();

    private int totalVariables = 0;
    private int totalLevels = 0;

    public Analyser() throws InvocationTargetException, NoSuchMethodException,
InstantiationException, IllegalAccessException {
        this.fuzzyLogicEngine = new Engine();
        this.fuzzyLogicEngine.setName("mainLogicEngine");
        this.setUp();
        this.preInitVariables();
        this.postInitVariables();
        this.initRules();
    }

    public Set<String> getInputVarNames() {
        return this.initedInputVars.stream()
            .map(Variable::getName)
            .collect(Collectors.toSet());
    }

    public Engine getFuzzyLogicEngine() {
        return this.fuzzyLogicEngine;
    }

    private void setUp() {
```

```
        final List<RuleSet> ruleSets =
App.getContext().getRuleSetStorage().readAll();
        final List<VariableType> variables =
App.getContext().getVariableStorage().readAll();

        final List<RuleDefinition> allRules = new ArrayList<>();
        ruleSets.forEach(rs -> allRules.addAll(rs.getRules()));

        final Set<UUID> inputs = new HashSet<>();
        allRules.stream()
            .map(RuleDefinition::inputVariableIds)
            .forEach(inputs::addAll);
        final Set<UUID> outputs = allRules.stream()
            .map(RuleDefinition::outputVariableId)
            .collect(Collectors.toSet());
        final Set<UUID> varsIntersection = inputs.stream()
            .filter(outputs::contains)
            .collect(Collectors.toSet());

        outputs.removeAll(varsIntersection);
        inputs.removeAll(varsIntersection);

        this.inputVars.addAll(variables.stream()
            .filter(v -> inputs.contains(v.getId()))
            .toList());
        this.outputVars.addAll(variables.stream()
            .filter(v -> outputs.contains(v.getId()))
            .toList());
    }

    private void preInitVariables() {
        final List<VariableType> variables =
App.getContext().getVariableStorage().readAll();
        for (VariableType variable : variables) {
            if (this.inputVars.contains(variable)) {
                final InputVariable initedInVar = new
InputVariable(variable.getName(), variable.getMinValue(),
variable.getMaxValue());
                variable.getVariables().forEach(inVarValue -> {
                    final Optional<? extends Class<? extends Term>> termType =
App.TERMS.stream()
                        .filter(t ->
variable.getTermTypeName().equals(t.getKey()))
                        .map(Pair::getValue)
                        .findFirst();
                    if (termType.isPresent()) {
                        final Term term =
TermFactory.createTerm(inVarValue.getName(), termType.get(),
inVarValue.getMinValue(), inVarValue.getMaxValue());
                        initedInVar.addTerm(term);
                    }
                });
                this.fuzzyLogicEngine.addInputVariable(initedInVar);
                this.initedInputVars.add(initedInVar);
            } else if (this.outputVars.contains(variable)) {
                final OutputVariable initedOutVar = new
OutputVariable(variable.getName(), variable.getMinValue(),
variable.getMaxValue());

                initedOutVar.setDefuzzifier(new Centroid());
                initedOutVar.setAggregation(new Maximum());
                variable.getVariables().forEach(outVarValue -> {
                    final Optional<? extends Class<? extends Term>> termType =
App.TERMS.stream()
```

```
                        .filter(t ->
variable.getTermTypeName().equals(t.getKey()))
                        .map(Pair::getValue)
                        .findFirst();
                    if (termType.isPresent()) {
                        final Term term =
TermFactory.createTerm(outVarValue.getName(), termType.get(),
outVarValue.getMinValue(), outVarValue.getMaxValue());
                        initedOutVar.addTerm(term);
                    }
                });
                this.fuzzyLogicEngine.addOutputVariable(initedOutVar);
                this.initedOutputVars.add(initedOutVar);
            } else {
                final OutputVariable initedOutVar = new
OutputVariable(variable.getName(), variable.getMinValue(),
variable.getMaxValue());
                final InputVariable initedInVar = new
InputVariable(variable.getName(), variable.getMinValue(),
variable.getMaxValue());
                variable.getVariables().forEach(varValue -> {
                    final Optional<? extends Class<? extends Term>> termType =
App.TERMS.stream()
                        .filter(t ->
variable.getTermTypeName().equals(t.getKey()))
                        .map(Pair::getValue)
                        .findFirst();
                    if (termType.isPresent()) {
                        final Term term =
TermFactory.createTerm(varValue.getName(), termType.get(),
varValue.getMinValue(), varValue.getMaxValue());
                        initedOutVar.addTerm(term);
                        initedInVar.addTerm(term);
                    }
                });
                this.fuzzyLogicEngine.addOutputVariable(initedOutVar);
                this.initedOutputVars.add(initedOutVar);
                this.fuzzyLogicEngine.addInputVariable(initedInVar);
                this.initedInputVars.add(initedInVar);
            }
            this.totalVariables++;
        }
    }

    private void postInitVariables() {
        final Set<RuleDefinition> allRules = new HashSet<>();
        App.getContext().getRuleSetStorage().readAll().stream()
            .map(RuleSet::getRules)
            .forEach(allRules::addAll);
        final VariableType resultVar = this.outputVars.get(0);
        this.variableIdsByLevelList.add(0, Set.of(resultVar.getId()));
        int variablesSorted = 1;
        int currentLevel = 1;
        while (variablesSorted != this.totalVariables) {
            final Set<VariableType> currentLevelVars = new HashSet<>();
            final int fcl = currentLevel -1;
            allRules.stream()
                .filter(r ->
this.variableIdsByLevelList.get(fcl).contains(r.outputVariableId()))
                .map(RuleDefinition::inputVariableIds)
                .forEach(ivs -> ivs
                    .forEach(iv ->
currentLevelVars.add(App.getContext().getVariableStorage().readById(iv))));
            variablesSorted += currentLevelVars.size();
```

```
            currentLevel++;
        }
        this.totalLevels = currentLevel;
    }

    private void initRules() throws NoSuchMethodException,
InvocationTargetException, InstantiationException, IllegalAccessException {
        final List<RuleSet> ruleSets =
App.getContext().getRuleSetStorage().readAll();
        for (RuleSet ruleSet : ruleSets) {
            final RuleBlock ruleBlock = new RuleBlock(ruleSet.getName());
            final List<Rule> ruleList = ruleSet.getRules().stream()
                .map(rd -> Rule.parse(rd.rule(), this.fuzzyLogicEngine))
                .toList();
            ruleBlock.setRules(ruleList);
            final Optional<? extends Class<? extends TNorm>> conjunction =
App.T_NORMS.stream()
                .filter(cjn -> cjn.getKey().equals(ruleSet.getConjunction()))
                .map(Pair::getValue)
                .findFirst();
            if (conjunction.isPresent()) {

ruleBlock.setConjunction(conjunction.get().getConstructor().newInstance());
            }
            final Optional<? extends Class<? extends SNorm>> disjunction =
App.S_NORMS.stream()
                .filter(djn -> djn.getKey().equals(ruleSet.getDisjunction()))
                .map(Pair::getValue)
                .findFirst();
            if (disjunction.isPresent()) {

ruleBlock.setDisjunction(disjunction.get().getConstructor().newInstance());
            }
            final Optional<? extends Class<? extends TNorm>> implication =
App.T_NORMS.stream()
                .filter(djn -> djn.getKey().equals(ruleSet.getImplication()))
                .map(Pair::getValue)
                .findFirst();
            if (implication.isPresent()) {

ruleBlock.setImplication(implication.get().getConstructor().newInstance());
            }
            final Optional<? extends Class<? extends Activation>> activation =
App.TRIGGERS.stream()
                .filter(act -> act.getKey().equals(ruleSet.getActivation()))
                .map(Pair::getValue)
                .findFirst();
            if (activation.isPresent()) {

ruleBlock.setActivation(activation.get().getConstructor().newInstance());
            }
            this.fuzzyLogicEngine.addRuleBlock(ruleBlock);
        }
    }

    public Double analyse(final Map<String, Double> inputs) {
        inputs.keySet()
            .forEach(input -> this.fuzzyLogicEngine
                .getInputVariable(input)
                .setValue(inputs.get(input)));
        this.fuzzyLogicEngine.process();
        if (this.totalLevels > 2) {
            for (int level = this.totalLevels - 1; level > 0; level--) {
                final Set<UUID> varIds =
```

```
this.variableIdsByLevelList.get(level);
                final int fl = level;
                final Set<VariableType> levelVars = varIds.stream()
                    .filter(vi ->
this.variableIdsByLevelList.get(fl).contains(vi))
                    .map(vi ->
App.getContext().getVariableStorage().readById(vi))
                    .collect(Collectors.toSet());
                levelVars.forEach(v -> {
                    final double outputValue =
this.fuzzyLogicEngine.getOutputValue(v.getName());
                    this.fuzzyLogicEngine.setInputValue(v.getName(),
outputValue);
                });
                this.fuzzyLogicEngine.process();
            }
        }
        final double result =
this.fuzzyLogicEngine.getOutputValue(this.outputVars.get(0).getName());
        return result;
    }
}

Context.class
public class Context {

    private final VariableTypeStorage variableStorage;
    private final RuleSetStorage ruleSetStorage;

    public Context(
        final RuleSetStorage ruleSetStorage,
        final VariableTypeStorage variableStorage) {
        this.ruleSetStorage = ruleSetStorage;
        this.variableStorage = variableStorage;
    }

    public VariableTypeStorage getVariableStorage() {
        return this.variableStorage;
    }

    public RuleSetStorage getRuleSetStorage() {
        return this.ruleSetStorage;
    }
}
App.class
public class App extends Application {

    public static final VariableType EMPTY_VAR_TYPE = new VariableType(null,
null, null, null);
    public static final VariableValue EMPTY_VAR_DIAPASON = new
VariableValue(null, null, null, null, null);
    public static final RuleSet EMPTY_RULE_SET = new RuleSet(null, null, null,
null, null);
    public static final RuleDefinition EMPTY_RULE = new RuleDefinition(null,
null, null);

    public static final List<Pair<String, Class<? extends Term>>> TERMS =
List.of(
        new Pair<>("Triangle", Triangle.class),
        new Pair<>("Bell", Bell.class),
        new Pair<>("Gaussian", Gaussian.class));
    public static final List<Pair<String, Class<? extends TNorm>>> T_NORMS =
List.of(
        new Pair<>("Algebraic product", AlgebraicProduct.class),
```

```
        new Pair<>("Minimum", Minimum.class),
        new Pair<>("Bounded difference", BoundedDifference.class));
    public static final List<Pair<String, Class<? extends SNorm>>> S_NORMS =
List.of(
        new Pair<>("Algebraic sum", AlgebraicSum.class),
        new Pair<>("Maximum", Maximum.class),
        new Pair<>("Bounded sum", BoundedSum.class));
    public static final List<Pair<String, Class<? extends Activation>>>
TRIGGERS = List.of(
        new Pair<>("Lowest", Lowest.class),
        new Pair<>("Highest", Highest.class),
        new Pair<>("First", First.class),
        new Pair<>("Last", Last.class),
        new Pair<>("General", General.class));

    private static Context context;

    @Override
    public void start(Stage stage) throws IOException, ClassNotFoundException {
        initContext();
        FileStorage.preInit();
        FXMLLoader fxmlLoader = new
FXMLLoader(App.class.getResource("main-view.fxml"));
        Scene scene = new Scene(fxmlLoader.load(), 650, 450);
        stage.setTitle("test");
        stage.setScene(scene);
        stage.show();
    }

    private static void initContext() throws IOException,
ClassNotFoundException {
        context = new Context(
            new RuleSetStorage(EMPTY_RULE_SET, "rules"),
            new VariableTypeStorage(EMPTY_VAR_TYPE, "variables"));
        context.getVariableStorage().init();
        context.getRuleSetStorage().init();
    }

    public static Context getContext() {
        return context;
    }

    public static void main(String[] args) {
        launch();
    }
}

MainMenuController.class
public class MainMenuController implements Initializable {

    private Analyser analyser = null;
    private VariableView selectedVariable = null;

    @FXML
    private ListView<VariableView> inputVarsList;
    @FXML
    private TextField inputValField;
    @FXML
    private TextArea resultLog;

    public void editVariableType() throws IOException {
        FXMLLoader fxmlLoader = new
FXMLLoader(App.class.getResource("variables-view.fxml"));
        Scene scene = new Scene(fxmlLoader.load(), 400, 310);
```

```
        Stage stage = new Stage();
        stage.setTitle("Variable type manager");
        stage.setScene(scene);
        stage.setMaxHeight(330);
        stage.setMaxWidth(420);
        stage.show();
    }

    public void editRules() throws IOException {
        FXMLLoader fxmlLoader = new
FXMLLoader(App.class.getResource("ruleset-view.fxml"));
        Scene scene = new Scene(fxmlLoader.load(), 620, 420);
        Stage stage = new Stage();
        stage.setTitle("Rule sets manager");
        stage.setScene(scene);
        stage.setMaxHeight(420);
        stage.setMaxWidth(620);
        stage.show();
    }

    public void reloadAnalyser() throws InvocationTargetException,
NoSuchMethodException, InstantiationException, IllegalAccessException {
        this.analyser = new Analyser();

this.inputVarsList.getItems().setAll(this.analyser.getInputVarNames().stream()
            .map(vn -> new VariableView(vn).setValue(0.0D))
            .toList());
    }

    public void selectVariable() {
        this.selectedVariable =
this.inputVarsList.getSelectionModel().getSelectedItem();

    }

    public void submitChanges() {
        if (this.selectedVariable != null) {
            final double value =
Double.parseDouble(this.inputValField.getText());
            this.selectedVariable.setValue(value);
            this.selectedVariable = null;
        }
    }

    public void cancelChanges() {
        if (this.selectedVariable != null) {
            this.inputValField.clear();
            this.selectedVariable = null;
        }
    }

    public void analyse() {
        final Map<String, Double> inputMap = new HashMap<>();
        this.inputVarsList.getItems().forEach(vw -> inputMap.put(vw.getName(),
vw.getValue()));
        final Double result = this.analyser.analyse(inputMap);
        this.resultLog.setText(result.toString());
    }

    @Override
    public void initialize(final URL url, final ResourceBundle resourceBundle)
{

    }
```

```
    public static class VariableView {

        private final String name;
        private Double value;

        public VariableView(final String name) {
            this.name = name;
        }

        public VariableView setValue(final Double value) {
            this.value = value;
            return this;
        }

        public String getName() {
            return this.name;
        }

        public Double getValue() {
            return this.value;
        }

        @Override
        public String toString() {
            return "VAR " + this.name + ": " + this.value.toString();
        }
    }
}
```

RuleSetViewController.class

```
public class RuleSetViewController implements Initializable {

    private RuleSet selectedRuleset = EMPTY_RULE_SET;

    @FXML
    private ListView<RuleSet> existingSetsList;
    @FXML
    private ListView<RuleDefinition> rulesList;
    @FXML
    private TextField idField;
    @FXML
    private TextField setNameField;
    @FXML
    private ComboBox<String> conjunctionBox;
    @FXML
    private ComboBox<String> disjunctionBox;
    @FXML
    private ComboBox<String> implicationBox;
    @FXML
    private ComboBox<String> activationBox;
    @FXML
    private Button addSetButton;
    @FXML
    private Button editSetButton;
    @FXML
    private Button deleteSetButton;
    @FXML
    private Button editRulesButton;
    @FXML
    private Button submitButton;
    @FXML
    private Button cancelButton;
```

```
    @Override
    public void initialize(final URL url, final ResourceBundle resourceBundle)
{
        this.loadRuleSets();
        this.resetFields();
        this.idField.setVisible(false);


this.conjunctionBox.setItems(FXCollections.observableList(T_NORMS.stream()
            .map(Pair::getKey)
            .toList()));

this.disjunctionBox.setItems(FXCollections.observableList(S_NORMS.stream()
            .map(Pair::getKey)
            .toList()));

this.implicationBox.setItems(FXCollections.observableList(T_NORMS.stream()
            .map(Pair::getKey)
            .toList()));

this.activationBox.setItems(FXCollections.observableList(TRIGGERS.stream()
            .map(Pair::getKey)
            .toList()));
    }

    public void addRuleSet() {
        this.selectedRuleset = null;
        this.existingSetsList.setDisable(true);
        this.addSetButton.setDisable(true);
        this.setNameField.setDisable(false);
        this.conjunctionBox.setDisable(false);
        this.disjunctionBox.setDisable(false);
        this.implicationBox.setDisable(false);
        this.activationBox.setDisable(false);
        this.submitButton.setDisable(false);
        this.cancelButton.setDisable(false);
        this.idField.clear();
        this.setNameField.clear();
        this.conjunctionBox.getSelectionModel().clearSelection();
        this.disjunctionBox.getSelectionModel().clearSelection();
        this.implicationBox.getSelectionModel().clearSelection();
        this.activationBox.getSelectionModel().clearSelection();
    }

    public void editRuleSet() {
        if (this.selectedRuleset != null) {
            this.existingSetsList.setDisable(true);
            this.addSetButton.setDisable(true);
            this.setNameField.setDisable(false);
            this.conjunctionBox.setDisable(false);
            this.disjunctionBox.setDisable(false);
            this.implicationBox.setDisable(false);
            this.activationBox.setDisable(false);
            this.submitButton.setDisable(false);
            this.cancelButton.setDisable(false);
            this.idField.setText(this.selectedRuleset.getId().toString());
            this.setNameField.setText(this.selectedRuleset.getName());

this.conjunctionBox.getSelectionModel().select(this.selectedRuleset.getConjunct
ion());

this.disjunctionBox.getSelectionModel().select(this.selectedRuleset.getDisjunct
ion());
```

```
this.implicationBox.getSelectionModel().select(this.selectedRuleset.getImplicat
ion());

this.activationBox.getSelectionModel().select(this.selectedRuleset.getActivatio
n());
        }
    }

    public void deleteRuleSet() {
        if (this.selectedRuleset != null) {
            App.getContext().getRuleSetStorage().delete(this.selectedRuleset);
            this.selectedRuleset = null;
            this.resetFields();
            this.loadRuleSets();
        }
    }

    public void selectExistingRuleset() {
        this.selectedRuleset =
this.existingSetsList.getSelectionModel().getSelectedItem();
        this.idField.setText(this.selectedRuleset.getId().toString());
        this.setNameField.setText(this.selectedRuleset.getName());

this.conjunctionBox.getSelectionModel().select(this.selectedRuleset.getConjunct
ion());

this.disjunctionBox.getSelectionModel().select(this.selectedRuleset.getDisjunct
ion());

this.implicationBox.getSelectionModel().select(this.selectedRuleset.getImplicat
ion());

this.activationBox.getSelectionModel().select(this.selectedRuleset.getActivatio
n());

        this.addSetButton.setDisable(true);
        this.editSetButton.setDisable(false);
        this.deleteSetButton.setDisable(false);
        this.editRulesButton.setDisable(false);
    }

    public void editRules() throws IOException {
        if (this.selectedRuleset != null) {
            RuleViewController.ruleSet = this.selectedRuleset;
            FXMLLoader fxmlLoader = new
FXMLLoader(App.class.getResource("rule-view.fxml"));
            Scene scene = new Scene(fxmlLoader.load(), 700, 450);
            Stage stage = new Stage();
            stage.setTitle("Rule manager");
            stage.setScene(scene);
            stage.setMaxHeight(450);
            stage.setMaxWidth(700);
            stage.show();
        }
    }

    public void submitChanges() throws IOException {
        if (this.idField.getText().isBlank()) {
            final RuleSet newRuleSet = new RuleSet(
                this.setNameField.getText(),
                this.conjunctionBox.getSelectionModel().getSelectedItem(),
                this.disjunctionBox.getSelectionModel().getSelectedItem(),
                this.implicationBox.getSelectionModel().getSelectedItem(),
                this.activationBox.getSelectionModel().getSelectedItem());
```

```
            App.getContext().getRuleSetStorage().write(newRuleSet);
        } else {
            final RuleSet editedRuleSet = new RuleSet(
                UUID.fromString(this.idField.getText()),
                this.setNameField.getText(),
                this.conjunctionBox.getSelectionModel().getSelectedItem(),
                this.disjunctionBox.getSelectionModel().getSelectedItem(),
                this.implicationBox.getSelectionModel().getSelectedItem(),
                this.activationBox.getSelectionModel().getSelectedItem());
            App.getContext().getRuleSetStorage().write(editedRuleSet);
        }
        this.selectedRuleset = null;
        this.resetFields();
        this.loadRuleSets();
    }

    public void cancelChanges() {
        this.selectedRuleset = null;
        this.resetFields();
        this.loadRuleSets();
    }

    private void resetFields() {
        this.selectedRuleset = null;
        this.existingSetsList.setDisable(false);
        this.addSetButton.setDisable(false);
        this.editRulesButton.setDisable(true);
        this.deleteSetButton.setDisable(true);
        this.setNameField.setDisable(true);
        this.conjunctionBox.setDisable(true);
        this.disjunctionBox.setDisable(true);
        this.implicationBox.setDisable(true);
        this.activationBox.setDisable(true);
        this.submitButton.setDisable(true);
        this.cancelButton.setDisable(true);
        this.idField.clear();
        this.setNameField.clear();
        this.conjunctionBox.getSelectionModel().clearSelection();
        this.disjunctionBox.getSelectionModel().clearSelection();
        this.implicationBox.getSelectionModel().clearSelection();
        this.activationBox.getSelectionModel().clearSelection();
    }

    private void loadRuleSets() {
        final List<RuleSet> ruleSets =
App.getContext().getRuleSetStorage().readAll();
        this.existingSetsList.getItems().clear();
        this.existingSetsList.getItems().setAll(ruleSets);
    }
}

RuleViewController.class
public class RuleViewController implements Initializable {

    private RuleDefinition selectedRule = EMPTY_RULE;
    private VariableType selectedVariable = EMPTY_VAR_TYPE;
    private RuleBuilder ruleBuilder = null;
    private RuleDefinition rule = null;
    static RuleSet ruleSet = null;

    @FXML
    private TextField idField;
    @FXML
    private TextArea editor;
```

```
    @FXML
    private ListView<VariableType> variableList;
    @FXML
    private ListView<VariableValue> valuesList;
    @FXML
    private ListView<RuleDefinition> existingRulesList;
    @FXML
    private Button addRuleButton;
    @FXML
    private Button editRuleButton;
    @FXML
    private Button deleteRuleButton;
    @FXML
    private Button ifOperatorButton;
    @FXML
    private Button thenOperatorButton;
    @FXML
    private Button andOperatorButton;
    @FXML
    private Button orOperatorButton;
    @FXML
    private Button notOperatorButton;
    @FXML
    private Button submitButton;
    @FXML
    private Button cancelButton;

    @Override
    public void initialize(final URL url, final ResourceBundle resourceBundle)
{
        this.ruleBuilder = null;
        this.resetFields();
        this.loadRules();
    }

    public void addRule() {
        this.ruleBuilder = new RuleBuilder();
        this.existingRulesList.setDisable(true);
        this.variableList.setDisable(true);
        this.valuesList.setDisable(true);
        this.addRuleButton.setDisable(true);
        this.editRuleButton.setDisable(true);
        this.deleteRuleButton.setDisable(true);
        this.ifOperatorButton.setDisable(false);
        this.thenOperatorButton.setDisable(true);
        this.andOperatorButton.setDisable(true);
        this.orOperatorButton.setDisable(true);
        this.notOperatorButton.setDisable(true);
        this.submitButton.setDisable(true);
        this.cancelButton.setDisable(false);
    }

    public void editRule() {}

    public void deleteRule() {
         if (this.selectedRule != null) {
             ruleSet.removeRules(List.of(this.selectedRule));
             this.selectedRule = null;
             this.existingRulesList.setDisable(false);
             this.variableList.setDisable(true);
             this.valuesList.setDisable(true);
             this.addRuleButton.setDisable(true);
             this.editRuleButton.setDisable(false);
             this.deleteRuleButton.setDisable(false);
```

```
            this.ifOperatorButton.setDisable(true);
            this.thenOperatorButton.setDisable(true);
            this.andOperatorButton.setDisable(true);
            this.orOperatorButton.setDisable(true);
            this.notOperatorButton.setDisable(true);
            this.submitButton.setDisable(true);
            this.cancelButton.setDisable(true);
        }
    }

    public void addIfOperator() {
        this.ruleBuilder.initStatement();
        this.variableList.setDisable(false);
        this.valuesList.setDisable(true);
        this.ifOperatorButton.setDisable(true);
        this.thenOperatorButton.setDisable(true);
        this.andOperatorButton.setDisable(true);
        this.orOperatorButton.setDisable(true);
        this.notOperatorButton.setDisable(false);
    }

    public void addThenOperator() {
        this.ruleBuilder.finalizeCondition();
        this.variableList.setDisable(false);
        this.valuesList.setDisable(true);
        this.ifOperatorButton.setDisable(true);
        this.thenOperatorButton.setDisable(true);
        this.andOperatorButton.setDisable(true);
        this.orOperatorButton.setDisable(true);
        this.notOperatorButton.setDisable(true);
    }

    public void addAndOperator() {
        this.ruleBuilder.and(new AndOperator());
        this.variableList.setDisable(false);
        this.valuesList.setDisable(true);
        this.ifOperatorButton.setDisable(true);
        this.thenOperatorButton.setDisable(true);
        this.andOperatorButton.setDisable(true);
        this.orOperatorButton.setDisable(true);
        this.notOperatorButton.setDisable(true);
    }

    public void addOrOperator() {
        this.ruleBuilder.or(new OrOperator());
        this.variableList.setDisable(false);
        this.valuesList.setDisable(true);
        this.ifOperatorButton.setDisable(true);
        this.thenOperatorButton.setDisable(true);
        this.andOperatorButton.setDisable(true);
        this.orOperatorButton.setDisable(true);
        this.notOperatorButton.setDisable(true);}

    public void addNotOperator() {
        this.ruleBuilder.not();
        this.variableList.setDisable(true);
        this.valuesList.setDisable(false);
        this.ifOperatorButton.setDisable(true);
        this.thenOperatorButton.setDisable(true);
        this.andOperatorButton.setDisable(true);
        this.orOperatorButton.setDisable(true);
        this.notOperatorButton.setDisable(true);
    }
```

```
    public void submitChanges() throws IOException {
        if (ruleSet != null) {
            ruleSet.addRules(List.of(this.rule));
            App.getContext().getRuleSetStorage().write(ruleSet);
            this.rule = null;
            this.ruleBuilder = null;
            this.resetFields();
            this.loadRules();
        }
    }

    public void cancelChanges() {
        this.rule = null;
        this.ruleBuilder = null;
        this.resetFields();
        this.loadRules();
    }

    public void addVariableToRule() {
        final VariableType variableType =
this.variableList.getSelectionModel().getSelectedItem();
        if (variableType != null) {
            this.selectedVariable = variableType;
            this.valuesList.getItems().setAll(variableType.getVariables());
            if (this.ruleBuilder.isFinalized()) {
                this.ruleBuilder.initConclusion(new
ConclusionStatement(variableType));
                this.variableList.setDisable(true);
                this.valuesList.setDisable(false);
                this.ifOperatorButton.setDisable(true);
                this.thenOperatorButton.setDisable(true);
                this.andOperatorButton.setDisable(true);
                this.orOperatorButton.setDisable(true);
                this.notOperatorButton.setDisable(true);
            } else {
                this.ruleBuilder.when(new PredicateStatement(variableType));
                this.variableList.setDisable(true);
                this.valuesList.setDisable(false);
                this.ifOperatorButton.setDisable(true);
                this.thenOperatorButton.setDisable(true);
                this.andOperatorButton.setDisable(true);
                this.orOperatorButton.setDisable(true);
                this.notOperatorButton.setDisable(false);
            }
        }
    }

    public void addValueToRule() {
        final VariableValue variableValue =
this.valuesList.getSelectionModel().getSelectedItem();
        if (variableValue != null) {
            if (this.ruleBuilder.isFinalized()) {
                this.rule = this.ruleBuilder.then(variableValue);
                this.submitButton.setDisable(false);
                this.variableList.setDisable(true);
                this.valuesList.setDisable(true);
                this.ifOperatorButton.setDisable(true);
                this.thenOperatorButton.setDisable(true);
                this.andOperatorButton.setDisable(true);
                this.orOperatorButton.setDisable(true);
                this.notOperatorButton.setDisable(true);
            } else {
                this.ruleBuilder.is(variableValue);
                this.variableList.setDisable(true);
```

```
                this.valuesList.setDisable(true);
                this.ifOperatorButton.setDisable(true);
                this.thenOperatorButton.setDisable(false);
                this.andOperatorButton.setDisable(false);
                this.orOperatorButton.setDisable(false);
                this.notOperatorButton.setDisable(true);
            }
            this.selectedVariable = null;
            this.valuesList.getItems().clear();
        }
    }

    public void selectExistingRule() {
        final RuleDefinition rule =
this.existingRulesList.getSelectionModel().getSelectedItem();
        if (rule != null) {
            this.idField.setText(this.selectedRule.id().toString());
            this.editor.setText(this.selectedRule.rule());
            this.selectedRule = rule;
            this.existingRulesList.setDisable(false);
            this.variableList.setDisable(true);
            this.valuesList.setDisable(true);
            this.addRuleButton.setDisable(true);
            this.editRuleButton.setDisable(false);
            this.deleteRuleButton.setDisable(false);
            this.ifOperatorButton.setDisable(true);
            this.thenOperatorButton.setDisable(true);
            this.andOperatorButton.setDisable(true);
            this.orOperatorButton.setDisable(true);
            this.notOperatorButton.setDisable(true);
            this.submitButton.setDisable(true);
            this.cancelButton.setDisable(true);
        }
    }

    public void finishManualChanges() {}

    private void resetFields() {
        this.existingRulesList.setDisable(false);
        this.variableList.setDisable(true);
        this.valuesList.setDisable(true);
        this.addRuleButton.setDisable(false);
        this.editRuleButton.setDisable(true);
        this.deleteRuleButton.setDisable(true);
        this.ifOperatorButton.setDisable(true);
        this.thenOperatorButton.setDisable(true);
        this.andOperatorButton.setDisable(true);
        this.orOperatorButton.setDisable(true);
        this.notOperatorButton.setDisable(true);
        this.submitButton.setDisable(true);
        this.cancelButton.setDisable(true);
    }

    private void loadRules() {
        this.existingRulesList.getItems().setAll(ruleSet.getRules());
        final List<VariableType> variableTypes =
App.getContext().getVariableStorage().readAll();
        this.variableList.getItems().setAll(variableTypes);
    }
}

VariableDiapasonController.class
public class VariableDiapasonController implements Initializable {
```

```
    static VariableType selectedVariable = EMPTY_VAR_TYPE;
    private VariableValue selectedValue = EMPTY_VAR_DIAPASON;

    @FXML
    private ListView<VariableValue> existingValuesList;

    @FXML
    private TextField idField;
    @FXML
    private TextField nameField;
    @FXML
    private TextField minValField;
    @FXML
    private TextField maxValField;
    @FXML
    private Label termTypeLabel;
    @FXML
    private Button addValButton;
    @FXML
    private Button editValButton;
    @FXML
    private Button deleteValButton;
    @FXML
    private Button submitButton;
    @FXML
    private Button cancelButton;

    @Override
    public void initialize(final URL url, final ResourceBundle resourceBundle)
{
        this.loadValues();
        this.resetFields();
    }

    public void addValue() {
        this.selectedValue = null;
        this.addValButton.setDisable(true);
        this.editValButton.setDisable(true);
        this.deleteValButton.setDisable(true);
        this.nameField.setDisable(false);
        this.minValField.setDisable(false);
        this.maxValField.setDisable(false);
        this.submitButton.setDisable(false);
        this.cancelButton.setDisable(false);
        this.idField.clear();
        this.nameField.clear();
        this.minValField.clear();
        this.minValField.setText("0");
        this.maxValField.clear();
        this.maxValField.setText("1");
        this.termTypeLabel.setText(selectedVariable.getTermTypeName());
    }


    public void editValue() {
        if (this.selectedValue != null) {
            this.addValButton.setDisable(true);
            this.editValButton.setDisable(true);
            this.deleteValButton.setDisable(true);
            this.nameField.setDisable(false);
            this.minValField.setDisable(false);
            this.maxValField.setDisable(false);
            this.submitButton.setDisable(false);
            this.cancelButton.setDisable(false);
```

```
            this.idField.setText(this.selectedValue.getId().toString());
            this.nameField.setText(this.selectedValue.getName());

this.minValField.setText(this.selectedValue.getMinValue().toString());

this.maxValField.setText(this.selectedValue.getMaxValue().toString());
            this.termTypeLabel.setText(selectedVariable.getTermTypeName());
        }
    }

    public void deleteValue() throws IOException {
        if (this.selectedValue != null) {
            selectedVariable.removeVariables(Set.of(this.selectedValue));
            App.getContext().getVariableStorage().write(selectedVariable);
            this.selectedValue = null;
            this.loadValues();
            this.resetFields();
        }
    }

    public void selectExistingValue() {
        this.selectedValue =
this.existingValuesList.getSelectionModel().getSelectedItem();
        this.idField.setText(this.selectedValue.getId().toString());
        this.nameField.setText(this.selectedValue.getName());
        this.termTypeLabel.setText(selectedVariable.getTermTypeName());
        this.minValField.setText(this.selectedValue.getMinValue().toString());
        this.maxValField.setText(this.selectedValue.getMaxValue().toString());
        this.addValButton.setDisable(true);
        this.editValButton.setDisable(false);
        this.deleteValButton.setDisable(false);
    }

    public void submitChanges() throws IOException {
        if (this.idField.getText().isBlank()) {
            final UUID id = UUID.randomUUID();
            final TermDefinition termDefinition = new TermDefinition(
                id,
                selectedVariable.getTermTypeName(),
                this.nameField.getText() + "-" + id);
            termDefinition.addParam(0,
Double.parseDouble(this.minValField.getText()));
            termDefinition.addParam(1,
Double.parseDouble(this.maxValField.getText()));
            final VariableValue newValue = new VariableValue(
                id,
                this.nameField.getText(),
                termDefinition,
                selectedVariable,
                Double.parseDouble(this.minValField.getText()),
                Double.parseDouble(this.maxValField.getText()));
            selectedVariable.addVariables(Set.of(newValue));
        } else {
            final TermDefinition termDefinition = new TermDefinition(
                UUID.fromString(this.idField.getText()),
                selectedVariable.getTermTypeName(),
                this.nameField.getText() + "-" + this.idField.getText());
            termDefinition.addParam(0,
Double.parseDouble(this.minValField.getText()));
            termDefinition.addParam(1,
Double.parseDouble(this.maxValField.getText()));
            final VariableValue editedValue = new VariableValue(
                UUID.fromString(this.idField.getText()),
                this.nameField.getText(),
```

```
                termDefinition,
                selectedVariable,
                Double.parseDouble(this.minValField.getText()),
                Double.parseDouble(this.maxValField.getText()));
            selectedVariable.addVariables(Set.of(editedValue));
        }
        App.getContext().getVariableStorage().write(selectedVariable);
        this.selectedValue = null;
        this.resetFields();
        this.loadValues();
    }

    public void cancelChanges() {
        this.selectedValue = null;
        this.resetFields();
        this.loadValues();
    }

    private void resetFields() {
        this.existingValuesList.setDisable(false);
        this.nameField.setDisable(true);
        this.minValField.setDisable(true);
        this.maxValField.setDisable(true);
        this.addValButton.setDisable(false);
        this.editValButton.setDisable(true);
        this.deleteValButton.setDisable(true);
        this.submitButton.setDisable(true);
        this.cancelButton.setDisable(true);
        this.idField.clear();
        this.nameField.clear();
        this.termTypeLabel.setText("");
        this.minValField.clear();
        this.maxValField.clear();
    }

    private void loadValues() {
        final Set<VariableValue> variables = selectedVariable.getVariables();
        this.existingValuesList.getItems().setAll(variables);
    }
}

VariableTypeMenuController.class
public class VariableTypeMenuController implements Initializable {

    private VariableType selectedVariable = EMPTY_VAR_TYPE;

    @FXML
    private ListView<VariableType> existingVariablesList;
    @FXML
    private ListView variableValuesList;

    @FXML
    private TextField idField;
    @FXML
    private TextField nameField;
    @FXML
    private ComboBox<String> termSelector;
    @FXML
    private TextField minValField;
    @FXML
    private TextField maxValField;
    @FXML
    private Button addButton;
    @FXML
```

```
    private Button editButton;
    @FXML
    private Button editValButton;
    @FXML
    private Button deleteButton;
    @FXML
    private Button submitButton;
    @FXML
    private Button cancelButton;

    @Override
    public void initialize(final URL url, final ResourceBundle resourceBundle)
{
        this.loadVariables();
        this.resetFields();
        this.idField.setVisible(false);

        this.termSelector.setItems(FXCollections.observableList(TERMS.stream()
            .map(Pair::getKey)
            .toList()));
    }

    public void addVariable() {
        this.selectedVariable = null;
        this.existingVariablesList.setDisable(true);
        this.addButton.setDisable(true);
        this.nameField.setDisable(false);
        this.termSelector.setDisable(false);
        this.minValField.setDisable(false);
        this.maxValField.setDisable(false);
        this.submitButton.setDisable(false);
        this.cancelButton.setDisable(false);
        this.idField.clear();
        this.nameField.clear();
        this.minValField.clear();
        this.minValField.setText("0");
        this.maxValField.clear();
        this.maxValField.setText("1");
    }

    public void editVariable() {
        if (this.selectedVariable != null) {
            this.existingVariablesList.setDisable(true);
            this.addButton.setDisable(true);
            this.editButton.setDisable(true);
            this.editValButton.setDisable(false);
            this.deleteButton.setDisable(true);
            this.nameField.setDisable(false);
            this.termSelector.setDisable(false);
            this.minValField.setDisable(false);
            this.maxValField.setDisable(false);
            this.submitButton.setDisable(false);
            this.cancelButton.setDisable(false);
            this.idField.setText(this.selectedVariable.getId().toString());
            this.nameField.setText(this.selectedVariable.getName());

this.minValField.setText(this.selectedVariable.getMinValue().toString());

this.maxValField.setText(this.selectedVariable.getMaxValue().toString());
        }
    }

    public void deleteVariable() {
        if (this.selectedVariable != null) {
```

```
App.getContext().getVariableStorage().delete(this.selectedVariable);
            this.selectedVariable = null;
            this.resetFields();
            this.loadVariables();
        }
    }

    public void editVariableValues() throws IOException {
        if (this.selectedVariable != null) {
            FXMLLoader fxmlLoader = new
FXMLLoader(App.class.getResource("variable-diapasons-view.fxml"));
            Scene scene = new Scene(fxmlLoader.load(), 440, 310);
            Stage stage = new Stage();
            stage.setTitle("Variable values manager");
            stage.setScene(scene);
            stage.setMaxHeight(330);
            stage.setMaxWidth(440);
            stage.show();
            VariableDiapasonController.selectedVariable =
this.selectedVariable;
        }
    }

    public void selectExistingVariable() {
        this.selectedVariable =
this.existingVariablesList.getSelectionModel().getSelectedItem();
        this.idField.setText(this.selectedVariable.getId().toString());
        this.nameField.setText(this.selectedVariable.getName());

this.termSelector.getSelectionModel().select(this.selectedVariable.getTermTypeN
ame());

this.minValField.setText(this.selectedVariable.getMinValue().toString());

this.maxValField.setText(this.selectedVariable.getMaxValue().toString());

        this.addButton.setDisable(true);
        this.editButton.setDisable(false);
        this.deleteButton.setDisable(false);
        this.editValButton.setDisable(false);
    }

    public void submitChanges() throws IOException {
        if (this.idField.getText().isBlank()) {
            final VariableType newVariableType = new VariableType(
                this.nameField.getText(),
                Double.parseDouble(this.minValField.getText()),
                Double.parseDouble(this.maxValField.getText()),
                this.termSelector.getSelectionModel().getSelectedItem());
            App.getContext().getVariableStorage().write(newVariableType);
        } else {
            final VariableType editedVariableType = new VariableType(
                UUID.fromString(this.idField.getText()),
                this.nameField.getText(),
                Double.parseDouble(this.minValField.getText()),
                Double.parseDouble(this.maxValField.getText()),
                this.termSelector.getSelectionModel().getSelectedItem());
            App.getContext().getVariableStorage().write(editedVariableType);
        }
        this.selectedVariable = null;
        this.resetFields();
        this.loadVariables();
    }
```

```
    public void cancelChanges() {
        this.selectedVariable = null;
        this.resetFields();
        this.loadVariables();
    }

    private void resetFields() {
        this.existingVariablesList.setDisable(false);
        this.nameField.setDisable(true);
        this.termSelector.setDisable(true);
        this.minValField.setDisable(true);
        this.maxValField.setDisable(true);
        this.editButton.setDisable(true);
        this.editValButton.setDisable(true);
        this.deleteButton.setDisable(true);
        this.submitButton.setDisable(true);
        this.cancelButton.setDisable(true);
        this.idField.clear();
        this.nameField.clear();
        this.termSelector.getSelectionModel().clearSelection();
        this.minValField.clear();
        this.maxValField.clear();
    }

    private void loadVariables() {
        final List<VariableType> variableTypes =
App.getContext().getVariableStorage().readAll();
        this.existingVariablesList.getItems().clear();
        this.existingVariablesList.getItems().addAll(variableTypes);
    }
}
```