Одеський національний політехнічний університет
Інститут комп'ютерних систем
Кафедра системного програмного забезпечення

Пояснювальна записка

до дипломної роботи бакалавра

на тему: «Інформаційна система з електробезпеки з можливістю консультування»

Виконав: студент IV курсу, групи АС-121
напряму підготовки
6.050103 – Програмна інженерія
Куценко Д. С.
Керівник: Тройніна А. С.
Рецензент: _____

Одеса – 2016

Одеський національний політехнічний університет

Інститут комп'ютерних систем
Кафедра системного програмного забезпечення
Освітньо-кваліфікаційний рівень бакалавр
Напрям підготовки 6.050103 – Програмна інженерія

ЗАТВЕРДЖУЮ

Завідувач кафедри СПЗ

_____(В.В. Любченко)

«____» _____ 2016 року

ЗАВДАННЯ

НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Куценку Дмитру Сергійовичу

(прізвище, ім'я, по-батькові)

1. Тема роботи Інформаційна система з електробезпеки з можливістю консультування

Керівник роботи Тройніна Анастасія Сергіївна ст. викладач

(прізвище, ім'я, по-батькові, науковий ступінь, вчене звання)

затверджені наказом ректора ОНПУ від «_27_» 04 2016 року № 246- в

2. Строк подання студентом роботи _16.06.2016_

3. Вихідні дані до роботи у відповідності до технічного завдання на розробку

4. Зміст розрахунково-пояснювальної записки (перелік всіх питань, які необхідно розробити)

Вступ. Функціональні вимоги до системи. План виконання проекту. Проектування системи. Програмнареалізація. Тестуваннясистеми. Висновки. Список використаних джерел.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

У відповідності зі слайдами електронної презентації.

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 18.01.16

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Специфікація вимог	18.01.16 – 22.02.16	Виконано
2	Планування проекту	23.02.16 – 07.03.16	Виконано
3	Проектування проекту	08.03.16 – 01.04.16	Виконано
4	Реалізація проекту	01.04.16 – 30.05.16	Виконано
5	Тестування програми	30.05.16 – 06.06.16	Виконано
6	Введення в експлуатацію	06.06.16 – 02.07.16	Виконано

Студент _____ Куценко Д. С
(підпис) (прізвище та ініціали)

Керівник роботи _____ Тройніна А. С.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи: 81 с., 33 рис., 20 табл., 9 джерел.

Метою роботи є розробка системи для поліпшення пошуку інформації та організації консультації з електробезпеки.

Методи розробки базуються на технології JAVA, сервері бази даних MySQL і експертної системи JESS.

Як результат роботи виконана програмна реалізація інформаційної системи з електробезпеки з можливістю консультування або скорочено «ZEVS».

Ключові слова: експертна система, MySQL, JAVA, JESS, ZEVS.

ABSTRACT

The aim is to develop a system to improve information search and organization of consultations on electrical safety.

The methods of making technology based on JAVA, database server MySQL and expert system JESS.

As a result, of the implemented software implementation of information system with the possibility of electrical short or counseling «ZEVS».

Keywords: expert system, MySQL, JAVA, JESS, ZEVS.

ЗМІСТ

ВСТУП

Вся інформація, яка нас оточує, надзвичайно велика. С часом ця інформація збільшує свої розміри. Тому має місце проблема обробки інформації. Деякі організації зберігають інформацію в звичайних папках для документів, але більша частина все ж використовує комп'ютеризований спосіб. І вже сьогодні без баз даних неможливо представити роботу не однієї програми, тому зараз набагато доцільніше зберігати інформацію в електронному вигляді. Вартість зберігання інформації в файлах дешевше, ніж на папері. Бази даних допомагають зберігати, структурувати інформацію і витягувати оптимальним для користувача чином.

Використання клієнт/серверних технологій дозволяють зберегти значні кошти, а головне і час для отримання необхідної інформації, а також спрощують доступ і ведення.

Будь-яка система, навіть невелика, потребує постійного контролю. Незалежно від якості її налаштування та якості апаратного забезпечення в її складі не можна покладатися на можливість її самостійної стабільної роботи. Навіть наявність відповідальної особи, не може гарантувати стабільність її роботи, особливо у випадку значної складності або масштабності.

Оскільки людина не може постійно шукати проблему тої чи іншої проблеми, є виправданим застосування інформаційних систем, які дозволяють виявляти неполадки на ранніх етапах та отримувати про них якомога детальнішу інформацію. При цьому, обробка та аналіз отриманої інформації покладається на відповідальну особу.

Одним із способів автоматизації пошуку помилки є відтворення роботи людини-експерта. Це стало можливим з використанням експертних систем, які оперуються на базу знань про можливі неполадки. Але корисність систем на основі експертних знань, напряму залежить від якості баз знань.

Головна мета цієї дипломної роботи є скорочення часу на пошук необхідної інформації з електробезпеки. Дана система, моніторить вимоги з електробезпеки і приймає рішення про допустимість проведення робіт. Рішення програма буде

приймати виходячи з правил, які будуть описані на основі експертної системи.

Експертна система акумулює знання технічних фахівців про виявлення причин аномальної роботи. Найпростішим варіантом експертної системи є контекстно-залежна система допомоги. Більш складні експертні системи, бази знань, що володіють елементами штучного інтелекту.

Об’єктом автоматизації даної роботи є процес пошуку інформації з електробезпеки. Суб’єктами є персонал, який відповідає за електробезпеку на підприємстві, або будь-який користувач якому необхідно знайти інформацію та отримати рекомендації з електробезпеки.

Пояснювальна записка до дипломної роботи містить наступні розділи відповідно до етапів розробки програмних продуктів: специфікація та аналіз вимог, планування розробки, проектування програмної системи, опис роботи алгоритмів, тестування програмної системи.

1 ФУНКЦІОНАЛЬНІ ВИМОГИ ДО СИСТЕМИ

1.1 Опис предметної області і аналогів системи

Розробка полягає у створенні клієнт/серверної системи, яка поліпшує безпеку роботи з електрообладнанням. Дана система, розглядає вимоги з електробезпеки і приймає рішення про допустимість проведення робіт.

Програмний продукт повинен мати функції для роботи з базою правил, виконувати консультацію користувача, вміти розмежувати доступ за допомогою авторизації та реєстрації.

Система прийняття рішень повинна підвищити безпеку шляхом надання рекомендацій користувачу. За допомогою цієї програми користувачам буде зрозуміло, чи можна допускати працівників до роботи.

Для створення бази знань необхідно провести добування знань на основі відповідей експерта.

Було вирішено виділити наступні інструменті розробки баз знань:

- JESS;
- OpenL Tablets;
- JBoss Drools;
- IBM JRules.

JESS. Jess – двигун правил, написаний мовою Java та призначений для використання у програмних продуктах, написаних також мовою Java. Розроблений Ернестом Фрідманом-Хілом у національній лабораторії Сандія (Sandia National Labs, США) у 1995 році.

Jess був створений як реалізація CLIPS мовою програмування Java. У подальшому при розробці були реалізовані нові можливості, але Jess досі зберігає зворотну сумісність із CLIPS. Основі принципи його роботи лежать у порівнянні набору правил з набором фактів. Порівняння фактів та правил здійснюється за алгоритмом Rete. База знань при використанні Jess задається у

текстовому вигляді. Код Jess не є відкритим. Для використання Jess у комерційних продуктах потрібно придбати ліцензію, для учбових закладів та лабораторій ліцензія є безкоштовною.

OpenL Tablets. OpenL Tablets – система управління бізнес-правилами на основі таблиць прийняття рішень. Ця система розробляється компанією Exigen Services та написана мовою Java.

Дана система підтримує деякі додаткові типи подання знань. Окрім таблиць прийняття рішень підтримуються також дерева.

OpenL Tablets не надає жодних засобів візуалізації знань, засобів для перевірки якості знань, обмежені перевіркою синтаксичних помилок та типізації.

JBoss Drools. JBoss Drools – система управління бізнес-правилами з відкритим вихідним кодом розробляється компанією Red Hat і розповсюджується за ліцензією ASL 2 (Apache License 2.0.)

Даний продукт передбачає подання знань у вигляді фактів та правил, як і в Jess виконання відбуваються за допомогою алгоритму Rete. Правила описуються за допомогою спеціального діалекту мови XML.

IBM JRules. IBM JRules – система управління бізнес-правилами розробляється французькою компанією ILOG. Дана система підтримує подання знань як у вигляді фактів та правил, так і у вигляді таблиць прийняття рішень. Цей продукт має закритий вихідний код, для використання потрібна ліцензія.

Для проведення критичного аналізу пропонуються наступні критерії:

- форма подання знань – спосіб збергігання та обробки даних;
- відкритий вихідний код – наявність відкритого коду системи;
- інструменти редагування бази знань – наявність способів редагування бази знань;
- контроль якості знань – наяіність функціоналу контролю якості знань;

- візуалізація знань – наявність функціоналу графічного представлення знань;
- перевірка суперечливості знань – перевірка знань на супереслевість один одному;
- перевірка над достатності знань – перевірка вхідних даних для однозначного вирішення задачі;
- перевірка повноти знань – характеризує адекватність моделі, яка використовується для опису даної област.

У табл. 1.1 наведена порівняльна характеристика розглянутих засобів розробки баз знань.

Таблиця 1.1 - Порівняльна характеристика інструментів розробки баз знань

Редактор	Jess	OpenL Tablets	JBoss Drools	IBM JRules
Форма подання знань	Правила та факти	Таблиця рішень	Правила та факти	Правила та факти
Відкритий вихідний код	Ні	Ні	Так	Ні
Інструменти редагування бази знань	Середовище Eclipse, текстовий редактор	Microsoft Word, Excel	Середовище Eclipse, текстовий редактор	Середовище Eclipse, Microsoft Word, Excel
Контроль якості знань	Ні	Обмеж.	Обмеж.	Обмеж.
Візуалізація знань	Ні	Ні	Так	Так
Перевірка суперечливості знань	Ні	Так	Так	Так
Перевірка над достатності знань	Ні	Ні	Ні	Ні
Перевірка повноти знань	Ні	Ні	Ні	Так

Для аналізу функціоналу існуючих програмних продуктів було виділено наступні програми:

- MIXER;
- ExpSystem PC;
- ACE.

MIXER. Система надає допомогу програмістам в написанні мікропрограм для розробленої Texas Instruments HBIC TI990. По заданому опису прошивки система отримує оптимізовані прошивки для TI990. MIXER містить знання з мікропрограмування для TI990, взяті з керівництва і з аналізу прошивки керуючого ПЗУ TI990. Сюди відносяться знання про те, як перетворювати введені опису в набори проміжних операцій, як виділити відповідні регістри під змінні і як перетворити проміжні операції в набори мікрооперацій.

MIXER використовує ці знання, щоб визначити, які микрооперации є кращими для реалізації вбудованого. Система являє знання у вигляді правил і даних, володіє уніфікацією, керованої механізмом виведення, і динамічним поверненням. MIXER реалізована на мові Пролог. Вона була розроблена в Токійському університеті і доведена до рівня демонстраційного прототипу.

ExpSystem PC. Ця система допоможе людям не настільки добре розбираються в комплектуючих для персонального комп'ютера підібрати собі прийнятну конфігурацію майбутнього комп'ютера. У програму вводяться дані про вимогу до системи і максимальну суму грошей, передбачуваної для покупки.

Так само дана експертна система стане в нагоді для менеджерів комп'ютерних салонів що б автоматизувати свою роботу і зробити автоматизоване робоче місце для своїх покупців. Для підбору комплектуючих використовується два алгоритми розрахунку: для "багатого" і "бідного" покупця. Данная ЕС була розроблена в грудні 2006 року.

ACE. Система ACE визначає несправності в телефонної мережі і дає рекомендації щодо необхідного ремонту і відновлювальним заходам. Система працює без людського втручання, аналізуючи зведення-звіти про стан, одержувані щодня за допомогою програми, яка стежить за ходом ремонтних робіт в кабельній мережі.

ACE виявляє несправні телефонні кабелі і потім вирішує, чи потребують вони в планово-попереджувальному ремонті і вибирає, який тип ремонтних робіт найімовірніше буде ефективним.

ACE запам'ятовує свої рекомендації в спеціальній базі даних, до якої користувач має доступ. Система приймає рішення, застосовуючи знання щодо телефонних станцій, повідомлення системи та стратегії аналізу мереж. Подання знань в системі засновано на правилах; використовується схема управління за допомогою прямої ланцюжка міркувань.

Вона розроблена в Bell Laboratories. ACE пройшла дослідну експлуатацію і доведена до рівня комерційної експертної системи.

Для проведення критичного аналізу пропонуються наступні критерії:

- пошук інформації – наявність функції швидкого пошуку інформації;
- редагування – наявність функціоналу редагування даних системи;
- косультація користувача – наявність функції консультування користувача;
- змінення тексту – наявність функції змінення відображення текстової інформації.

У таблиці 1.2 представлено порівняльний аналіз готових систем з системою що розробляється.

Таблиця 1.2 – Порівняльний аналіз систем

Функції	Програмні продукти			
	ZEVS	MIXER	ExpSystem PC	ACE
Пошук інформації	+	+	+	-
Редагування	+	-	-	-
Консультація користувача	+	+	+	+
Змінення тексту	+	-	-	-

В процесі пошуку програмних продуктів було виявлено, що немає систем-аналогів, які б повністю відтворювали функціонал розроблюваної системи. Та чи інша програма, яка була знайдена частково відтворює функціонал але не повністю, і це є великим плюсом розроблюваної системи.

1.2 Варіанти використання системи

Користувач даної системи може виконувати наступні дії:

- Пошук інформації з електробезпеки;
- Отримання консультації;
- Додавання, видалення та редагування користувачів;
- Додавання, видалення та редагування правил експертної системи;
- Додавання, видалення та редагування інформаційної бази;
- Змінення типу та розміру шрифту;
- Авторизація;
- Реєстрація.

На рис. 1.1 представлені формалізовані вимоги до програмної системи, що розробляється у вигляді діаграми прецедентів.

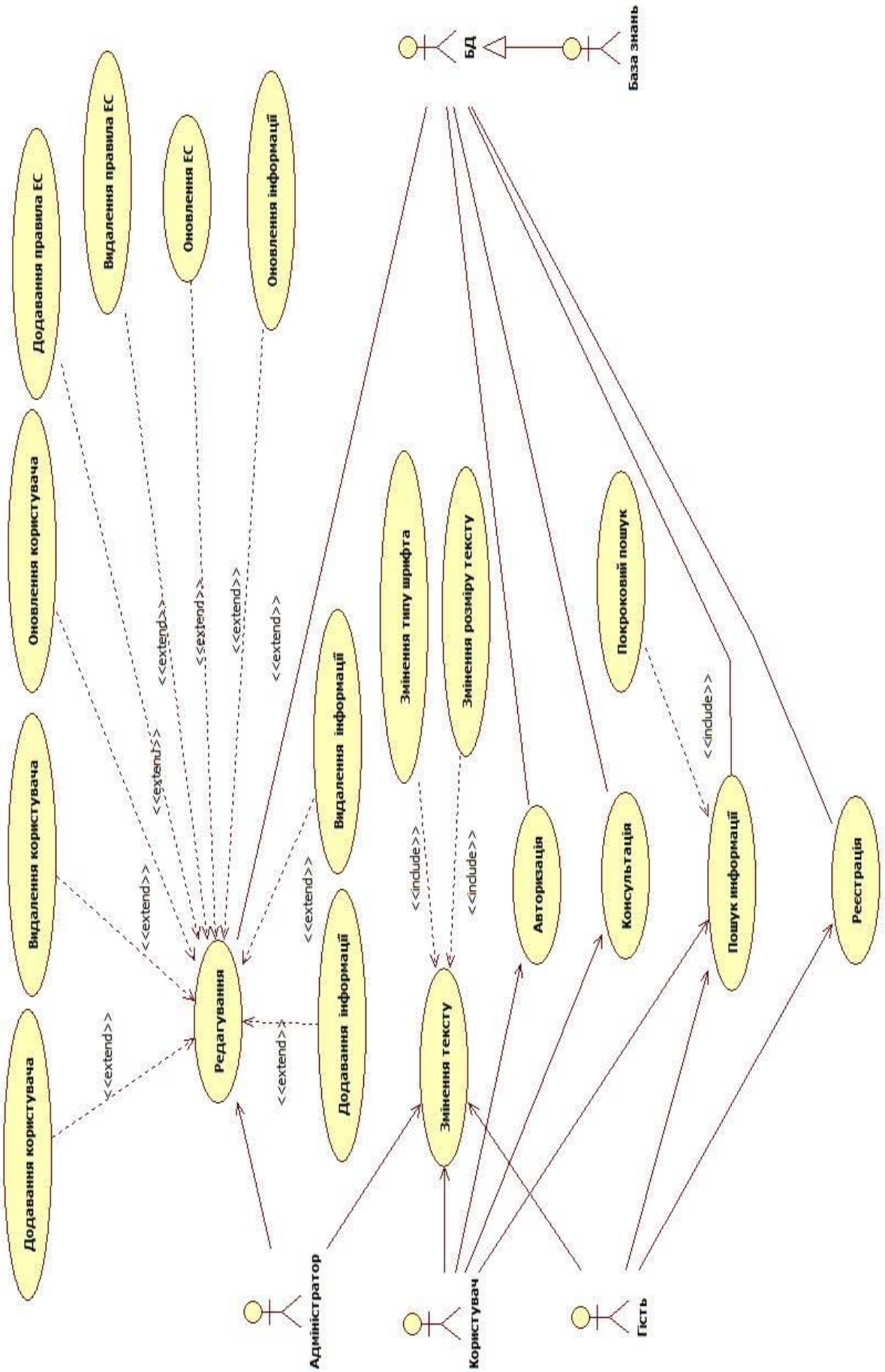

Рисунок 1.1 – Діаграма прецедентів

Адміністратор – актор, який раніше зареєстрований в системі і має особливі права доступу, він відповідає за супровід системи, в його арсеналі всі необхідні функція для роботи с даними.

Користувач – актор, який вже раніше був зареєстрований в системі, він може шукати інформацію та отримувати консультацію.

Гість – актор, який не зареєстрований в системі, його функціонал обмежений і складається лише з пошуку необхідної інформації.

Редагування – прецедент к якому має доступ лише Адміністратор, він відповідає за додавання, змінення та оновлення даних системи, дані дії Адміністратор може виконати в разі виявлення якихось недоліків.

Змінення тексту – прецедент за допомогою якого можна змінити відображення текстової інформації.

Консультація – у разі виникнення питань з електробезпеки, користувач може скористатися цією функцією, яка в ході інтерв'ю відповість на питання, що цікавлять користувача, і в результаті система допоможе вирішити якусь проблему.

Реєстрація – цей прецедент відповідає за реєстрацію нового користувача в системі.

Авторизація – прецедент, який відповідає за вхід в систему раніше зареєстрованого користувача.

1) Прецедент *«Реєстрація»*

Актори: Гість, база даних.

Передумова: Гість має доступ до локальної або глобальної мережі, з'єднання стабільне без розривів. Сервер запущений і готовий до виконання поставлених завдань.

Основний успішний сценарій:

1) Гість запускає програму «ZEVS».
2) Система ZEVS відображає вікно (форму) з проханням авторизуватися або зареєструватися.
3) Гість вибирає пункт «Зареєструватися»
4) Система ZEVS відображає вікно Реєстрації з полями Логін, Пароль**,** Прізвище, Ім’я, по батькові та поле повторного вводу паролю.
5) Гість вводить бажаний Логін. Система аналізує введенні данні та підтверджує.
6) Гість вводить бажаний Пароль**.** Система аналізує введенні данні та підтверджує.
7) Гість вводить бажане Прізвище та Ім’я, по батькові **.** Система аналізує введенні данні та підтверджує.
8) Гість повторно вводить пароль для підтвердження. Система аналізує введенні данні та підтверджує.
9) Система підтверджує реєстрацію нового користувача.

Альтернативний сценарій:

5a. Гість ввів Логін, який вже зайнятий другим користувачем. Повідомлення про помилку. Повернення до пункту 5, основного сценарію.

5b.Гість використовував неприпустимі символи при введенні Логіна. Повідомлення про помилку. Повернення до пункту 5, основного сценарію.

6a. Гість ввів дуже короткий Пароль. Повідомлення про помилку. Повернення до пункту 6, основного сценарію.

6b. Гість використовував неприпустимі символи при введенні Паролю. Повідомлення про помилку. Повернення до пункту 6, основного сценарію.

7а. Гість ввів неправильний Пароль. Повідомлення про помилку. Повернення до пункту 7, основного сценарію.

8а.Система не може додати нового користувача до бази. Повідомлення про помилку. Повернення до пункту 3, основного сценарію.

2) Прецедент *«Авторизація»*

Актори: Гість, база даних.

Передумова: Гість був раніше зареєстрований, має доступ до локальної мережі або глобальної мережі.

Основний успішний сценарій:

1) Користувач запускає програму «ZEVS».
2) Система ZEVS відображає вікно (форму) з проханням авторизуватися або зареєструватися.
3) Гість вводить свій Логін та Пароль.
4) Система аналізує введенні данні та підтверджує вхід.

Альтернативний сценарій:

4а. Система знаходить помилку в введених даних. Повідомлення про помилку. Повернення до пункту 3, основного сценарію.

3) Прецедент *«Консультація з електробезпеки»*

Актори: Користувач, база даних.

Передумова: Користувач виконав вхід у систему та має доступ до локальної або глобальної мережі.

Основний успішний сценарій:

1) Користувач вибирає пункт меню «Консультація»
2) Система ZEVS відображає вікно (форму) з проханням вибрати групу правил зі списку.

3) Користувач вибирає потрібний пункт. Система аналізує та підтверджує вибір.
4) Система починає опитування користувача для вирішення проблеми.
5) Користувач відповідає на питання Системи. Система аналізує та підтверджує.
6) Система виводить кінцевій результат опитування.

Альтернативний сценарій:

6а.Користувач неправильно відповідає на питання системи. Повторння запитання. Повернення до пункту 5, основного сценарію.

4) Прецедент *«Редагування»*

Актори: Адміністратор, база даних.

Передумова: Адміністратор виконав вхід у систему, та має права доступу до бази даних, також доступ до локальної або глобальної мережі.

Основний успішний сценарій:

1) Адміністратор вибирає пункт меню «Адміністрування».
2) Система ZEVS відображає вікно (форму) авторизування.
3) Користувач вводить необхідні дані. Система аналізує та підтверджує.
4) Система ZEVS відображає вікно (форму) з переліком функцій для роботи з даними.
5) Користувач вибирає потрібний пункт. Система аналізує та підтверджує вибір.
6) Користувач вводить необхідні данні. Система аналізує та підтверджує.
7) Система виводить повідомлення про успішне редагування даних.

Альтернативний сценарій:

4а.Користувач вводить некоректні дані. Повідомлення про помилку. Повернення до пункту 3, основного сценарію.

7а.Система не відредагувала данні. Повідомлення про помилку. Повернення до пункту 3, основного сценарію.

5) Прецедент *«Змінення тексту»*

Актори: Користувач

Передумова: Користувач виконав вхід у систему.

Основний успішний сценарій:

1) Система ZEVS відображає вікно (форму) з пунктами тип шрифту та розмір шрифту.
2) Користувач вводить або вибирає необхідні данні.
3) Система аналізує та змінює текс в текстовому полі.

6) Прецедент *«Пошук інформації з електробезпеки»*

Актори: Користувач, база даних.

Передумова: Користувач виконав вхід у систему, та має права доступу, також доступ до локальної або глобальної мережі.

Основний успішний сценарій:

1) Користувач вибирає пункт меню «Пошук інформації»
2) Система ZEVS відображає вікно (форму) з полем для вводу даних пошуку.
3) Користувач вводить потрібні данні. Система аналізує та підтверджує .
4) Система шукає та виводить необхідні данні .

Альтернативний сценарій:

3а.Адміністратор вводить некоректні дані. Повідомлення про помилку. Повернення до пункту 3, основного сценарію.

4а.Система не виводить данні. Повідомлення про помилку. Повернення до пункту 3, основного сценарію.

1.3 Нефункціональні вимоги до системи

Нефункціональні вимоги – описують, як повинна працювати система або програмний продукт, і якими властивостями або характеристиками вона повинна володіти. Як правило, кажучи про нефункціональні вимоги, найчастіше говорять про атрибути якості.

Сформулюємо вимоги до програмної системи за ISO 9126 з такими атрибутами якості:

- Функціональність - набір атрибутів, що відносяться до суті набору функцій та їх конкретних властивостям. Функціями є ті, які реалізують встановлені або передбачувані потреби;
- Ефективність - набір атрибутів, що відносяться до співвідношення між рівнем якості функціонування програмного забезпечення і обсягом використовуваних ресурсів при встановлених умовах;
- Переносимість (мобільність) - набір атрибутів, що відносяться до здатності програмного забезпечення бути перенесеним з одного оточення в інше.

В табл. 1.3 представлені атрибути якості програмної системи за ISO 9126.

Таблиця 1.3 - Нефункціональні вимоги програмної системи

Набір характеристик (за ISO 9126)	Властивості програмного забезпечення
Функціональність	Здатність до взаємодії (interoperability): Система активно взаємодіє з базою даних, яка містить у собі базу знань**,** яка необхідна при прийнятті рішень.

Продовження таблиці 1.3

Набір характеристик (за ISO 9126)	Властивості програмного забезпечення
Функціональність	Захищеність (security): У зв'язку з тим що база даних є найважливішою складовою системи доступ строго обмежений, за допомогою розділення прав.
Ефективність	Часові характеристики (time behaviour): Додавання даних в базу має здійснюється з максимальною швидкістю, і воно повинно бути < 5 секунд. Отримання даних з бази, мабуть найголовніше, так як на цьому зав'язана робота системи час отримання даних повинне бути < 3 секунд Використання ресурсів (resource utilisation): Іза того що мова java досить вимоглива до пам'яті, програмна система не повинна споживати < 1 гб пам'яті
Переносимість (мобільність)	Адаптованість (adaptability): Програмна система буде написана мовою java, оскільки мова java платформонезалежна дана програма зможе запускатися на платформах з різними операційними системами.

2 ПЛАН ВИКОНАННЯ ПРОЕКТУ

2.1 Оцінка тривалості розробки

Методика UCP (Use Case Points) дозволяє врахувати нефункціональні вимоги, організаційні ризики, компетенцію при оцінці і інші критерії.

В основі UCP лежить методика Feature points (оцінка на основі функціональних точок системи), проте вона значно спрощена для застосування не експертами Feature points. На відміну від Feature points, UCP враховує нефункціональні вимоги, організаційні ризики, компетенцію при оцінці і інші критерії.

Метод складається з 5 етапів:

- Етап 1. Оцінка акторів;
- Етап 2. Оцінка варіантів використання;
- Етап 3. Оцінка технічних факторів;
- Етап 4. Оцінка зовнішніх факторів;
- Етап 5. Результуючі оцінки.

Етап 1. Оцінка акторів

У методі UCP розрізняють три типи акторів, відповідно до складності проектування та реалізації інтерфейсу взаємодії системи з ними.

У таблиці 2.1 представлені визначення типів акторів.

Таблиця 2.1 - Визначення типів акторів

Тип актора	Вага	Примітка
Простий	1	Інша система з певним API (REST, SOAP, dll)
Середній	2	Інша система, взаємодія з якою виконується за певним протоколу (наприклад, TCP / IP)
Складний	3	У більшості випадків взаємодія з користувачем за допомогою GUI або веб-сторінок

У системі присутній два найважливіших актора це Користувач і База даних. Виходячи з наявних даних можна визначити, що: Користувач і Адміністратор це складний тип актора, так як саме він взаємодіє з системою за допомогою графічного інтерфейсу (GUI).

База даних це середній тип актора, тому що система має клієнт / серверну архітектуру і доступ до бази буде здійснюватися по мережевим протоколам передачі даних. У табл. 2.2 представлений короткий висновок.

Таблиця 2.2 - Короткий висновок

Найменування актора	Тип актора	Вага
Користувач	Складний	3
Адміністратор	Складний	3
БД	Середній	2

Тепер зробимо скориговану оцінку наявних акторів.

$$UAW = 3*1 + 3*1 + 2*1 = 3+ 3 + 2 = 8. \quad (2.1)$$

Етап 2. Оцінка варіантів використання

Тип варіанти використання найпростіше визначити відповідно до кількості транзакцій (неподільних операцій) в ньому, але наша система часто взаємодіє з базою даних тому краще використовувати інший підхід, а саме підхід, який базується на кількості об'єктів в базі даних (БД), які змінюються в його рамках (табл. 2.3).

Таблиця 2.3 - Визначення типів варіантів використання за кількістю змінених об'єктів в БД.

Тип варіанти використання	Вага	Кількість об'єктів в БД
Простий	5	1
Середній	10	2
Складний	15	3 и більше

Визначаємо тип варіантів використання, опис представлено в табл.2.4

Таблиця 2.4 - Визначення варіантів використання

Найменування варіанта використання	Тип варіанти використання	Вага	Опис
Авторизація	Простий	5	Простий варіант використання, складається всього лише з перевірки даних, без будь-яких змін.
Реєстрація	Простий	5	Також як і авторизація простий варіант використання, але все ж відбувається зміна даних в базі за рахунок додавання нового користувача.
Консультація	Простий	5	Простий варіант використання, з БД не відбувається ніяких змін крім отримання даних.
Змінення тексту	Простий	5	Як і у випадку з консультацією це простий варіант без змін БД.

Продовження таблиці 2.4

Найменування варіанта використання	Тип варіанти використання	Вага	Опис
Пошук інформації	Середній	10	Змін в БД не передбачає але реалізація функції на середньому рівні.
Редагування	Складний	15	Найскладніший варіант використання так як відбувається безпосередня робота з БД, в результаті чого дані в БД будуть активно змінятся.

І так, зробимо обчислення нескорегованої оцінки варіантів використання. Для обчислення нескорегованої оцінки варіантів використання (UUCW) слід підрахувати кількості варіантів використання кожного типу, помножити ці кількості на відповідні вагові коефіцієнти і знайти суму отриманих творів.

Показник UCP обчислюють за формулою:

$$UCP = UAW + UUCW \quad (2.2)$$

Загальна кількість варіантів використання 6, з них 4 простих ,1 середній і 1 складний.

UUCW = 4*5 + 1*10+ 1*15 = 30 + 15 = 45.

UAW = 8

UUCW = 45

UCP = UAW + UUCW.

UCP = 8 + 45 = 53.

Етап 3. Оцінка технічних факторів

Оцінка технічних факторів дає коефіцієнт для оцінки складності архітектури застосування (табл. 2.5). Оцінка проводиться за шкалою від 0 до 5, де 0 означає відсутність впливу, 3 - середній вплив, 5 - сильний вплив на розробку.

Таблиця 2.5 - Технічні фактори

Фактор	Опис	Вага	Пояснення	Оцінка
T1	Розподіленість системи	2	Інформує про необхідність системи в розподілених обчисленнях	0
T2	Час відгуку	1	Визначає ефективність системи з точки зору часу відгуку, потоку робіт і т.п.	5
T3	Ефективність кінцевого користувача	1	Визначає ефективність користувача з точки зору його (її) сприйняття	3
T4	Складність обробки	1	Визначає, чи будуть застосовуватися складні алгоритми для обробки даних	0
T5	Фокус на повторному використанні коду	1	Визначає, чи будуть розміщені елементи коду системи використовуватися знову	0
T6	Простота інсталяції	0,5	Визначає спосіб призначення та простоту інсталяції для кінцевого користувача, або необхідності в фахівця для установки системи	3
T7	Простота використання	0,5	Зазначає узгодженість інтерфейсу з його потребами	5
T8	Портативність	2	Визначає, чи має застосування працювати в різних середовищах	3

Продовження таблиці 2.5

Фактор	Опис	Вага	Пояснення	Оцінка
T9	Простота зміни	1	Визначає, буде будуватися система таким чином, щоб спростити її модифікації в майбутньому	3
T10	Паралельні обчислення	1	Інформує, матимуть в системі місце паралельні обчислення	0
T11	Способи захисту	1	Визначає вимагає система спеціальні засоби захисту даних і системи	3
T12	Доступ до третьої сторони	1	Визначає ступінь використання системи зовнішніми системами або акторами	3
T13	Потреба в спеціальному навчанні	1	Визначає, чи потрібно організувати тренінги для користувачів	0

Тепер на основі даних представлених в табл. 2.5 зробимо обчислення технічного чинника UCP.

Сума добутків вагових коефіцієнтів і оцінок для кожного з технічних факторів визначає показник TFactor. Оцінка технічного фактора обчислюється за формулою:

$$TCF = 0,6 + (0,01 * TFactor) \quad (2.3)$$

де TFactor = 28

Результат оцінки технічного фактора:

$$TCF = 0,6 + (0,01 * 28) = 0,6 + 0,28 = 0,88$$

Етап 4. Оцінка зовнішніх факторів

Оцінка зовнішніх факторів дає коефіцієнт для організаційних ризиків при розробці (табл. 2.6). Оцінка проводиться за шкалою від 0 до 5, де 0 означає відсутність впливу, 3 - середній вплив, 5 - сильний вплив на розробку.

Таблиця 2.6 - Зовнішні чинників

Фактор	Опис	Вага	Пояснення	Оцінка
F1	Знайомство з процесом розробки	1,5	Визначає, знайома чи команда з предметною областю і технічними аспектами вирішення проблеми клієнта. Особливу увагу слід приділити знання методології, в якій виконується проект, а також знання мов моделювання системи	3
F2	Досвід подібних проектів	0,5	Загальне уявлення про досвід команди в розробці програмного забезпечення	0
F3	Досвід об'єктно-орієнтованої розробки	1	Досвід в проектуванні об'єктно-орієнтованих додатків, а також в підтримку засобів для розробки інформаційних систем	5
F4	Досвідченість провідного аналітика	0,5	Здібності аналітика отримати вимоги від клієнта і знання по задач, які буде вирішувати система	3
F5	Мотивація	1	Здатність команди займатися призначеної завданням	5
F6	Стабільність вимог	2	Визначає, чи не будуть вимоги часто змінюватися	5

Продовження таблиці 2.6

Фактор	Опис	Вага	Пояснення	Оцінка
F7	Часткова зайнятість працівників	-1	Визначає, наскільки велика частка працівників часткової зайнятості	0
F8	Складність мови програмування	-1	Визначає, наскільки складно вивчити мову програмування: 0 - легко, за тиждень можна опанувати; 1 - не менше 2 тижні потрібно для вивчення мови; 2 - мінімум місяць потрібно для вивчення мови; 3 - потрібен спеціальний тренінг з мови; 4 - потрібен спеціальний тренінг і допомогу під час виконання проекту; 5 - складно, потрібні тільки досвідчені люди.	1

Зробимо обчислення зовнішніх факторів UCP. Сума добутків вагових коефіцієнтів і оцінок для кожного з зовнішніх факторів визначає показник EFactor.

Оцінка зовнішнього фактора обчислюється за формулою:

$$EF = 1,4 + (-0,03 * EFactor) \quad (2.4)$$

де EFactor = 22

Результат обчислення зовнішніх факторів:

EF = 1,4 + (-0,03 * 22) = 1,4 -0,66 = 0,74

Етап 5. Результуючі оцінки

Скориговані UCP обчислюються за формулою:

$$UCP = UCP * TCF * EF \quad (2.5)$$

$$AUCP = UCP * TCF * EF = 53 * 0,88 * 0,74 = 35 \quad (2.6)$$

Для визначення тривалості розробки потрібно знати, якою кількістю робочих годин відповідає один UCP. Для цього підраховується кількість чинників з безлічі F1 - F8, оцінки яких за абсолютним значенням перевищують 3. Якщо результат 2 або менше, то для розрахунку тривалості розробки приймається, що одному UCP відповідає 20 робочих годин. Якщо результат 3 або 4, то одному UCP відповідає 28 робочих годин. Якщо результат перевищує 4, то рекомендується переглянути умови виконання проекту. У разі неможливості це зробити приймається, що одному UCP відповідає 36 робочих годин.

Результат з безлічі F1 - F8 дорівнює 5, в слідстві чого одному UCP буде відповідає 36 робочих годин. В результати отримуємо 35 * 36 = 1260 годин.

Для розробки даної системи необхідно 1260/9 год що приблизно дорівнює 140 дням.

2.2 Оцінка ризиків

Ідентифікація ризиків - це виявлення ризиків, здатних вплинути на проект, і документальне оформлення їх характеристик.

Діаграма Ішикави (cause-effect diagram) - графічний інструмент, який дозволяє наочно і систематизовано аналізувати взаємозв'язку наслідків (effects) і причин (causes), які породжують ці наслідки або впливають на них. Ще ці діаграми називають «діаграмами рибного кістяка» (fishbone diagram) по їх зовнішню схожість зі скелетом риб.

Але яке б ім'я не використовувалося, необхідно пам'ятати, що цінність цього методу полягає в сприянні категоризації і структуризації безлічі потенційних

причин, а також ідентифікації найбільш імовірною кореневої причини для слідства досліджуваного.

Діаграма Ішикави представлена на рис. 2.1

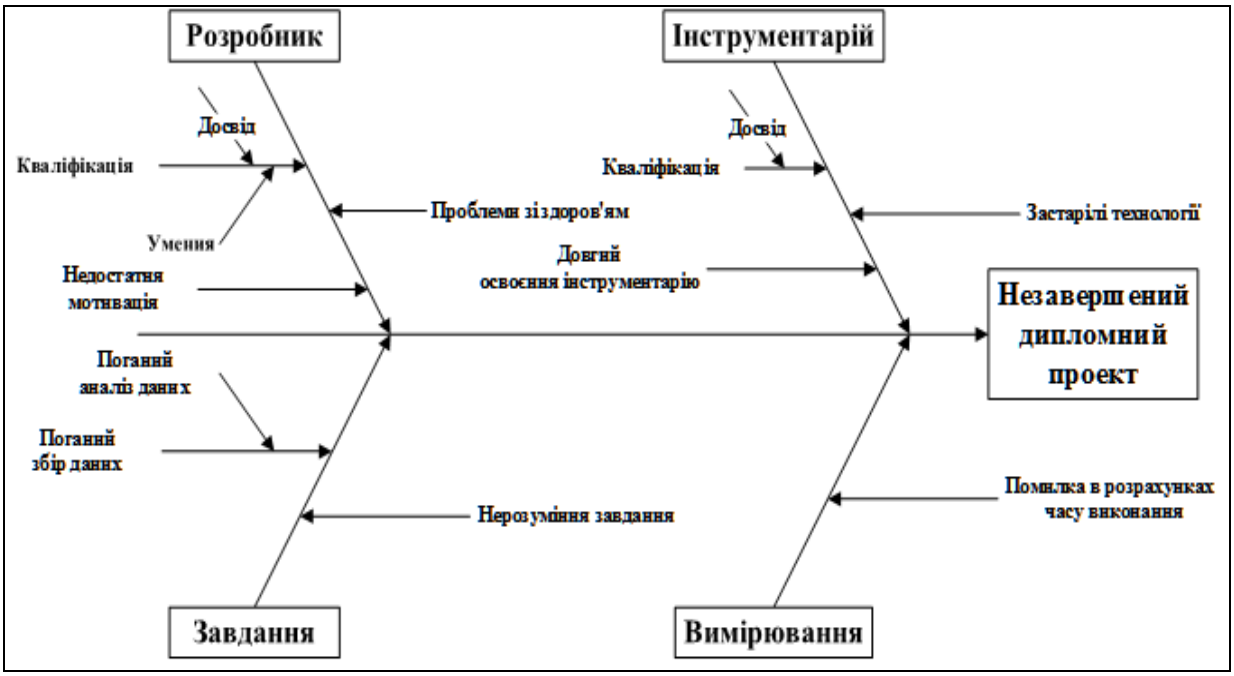

Рисунок 2.1 – Діаграма Ішикави

Головна проблема дипломного проекту, це незакінчений дипломний проект. Найважливіші можливі причини невиконання диплома це нерозуміння завдання проекту, проблеми пов'язані з інструментарієм, і проблеми з самим розробником.

Всі з перерахованих факторів можуть призвести до невиконання проекту якщо вони якимось чином виконуватися.

Для визначення рангу ризику використовується матриця ймовірностей і наслідків (табл. 2.7). Ранг ризику визначається множенням ваги ймовірності і значущості наслідків.

Таблиця 2.7 - Ранг ризику і матриця ймовірностей і наслідків

Вплив	Імовірність					
		1	2	3	Ранг ризику	
	3	3	6	9		-високий
	2	2	4	6		-середній
	1	1	2	3		-низькій

Для оцінки ризиків необхідна точна і адекватна інформація. Використання неточної інформації веде до помилок в оцінці. Невірна оцінка ризику також є ризиком. Для початку зробимо оцінку рангу ризиків (див. табл. 2.8).

Таблиця 2.8 – Оцінка ранга ризиків

Причина	Опис	Оцінка
Інструментарій: Кваліфікація Застарілі технології Довге освоєння технології	Низька кваліфікація працівника, застарілий інструментарій довгий освоєння технології розробки може призвести до невиконання диплома	Вплив (3) * Імовірність (3) = 9
Розробник: Кваліфікація Проблем зі здоров'ям Недостатня мотивація	Розробник це жива людина, у нього є свої бажання і потреби. Кожна з представлених причин може безпосередньо вплинути на процес розробки	Кваліфікація = Вл (3) * Ве (3) = 9 Проблеми зі здоров'ям= Вл (3) * Ве (3) = 9 Недостатня мотивація = Вл (3) * Ве (1) = 3

Продовження таблиці 2.8

Причина	Опис	Оцінка
Завдання: Нерозуміння завдання Поганий збір і аналіз даних	Найголовніше при створенні проекту це вміння розуміти завдання і аналізувати отримані дані, це відіграє важливу роль у проекті	Нерозуміння завдання = Вл (3) * Ве (2) = 6 Поганий збір і аналіз даних = Вл (3) * Ве (2) = 6
Вимірювання: Помилка розрахунку часу виконання	Після збору і аналізу завдання команда розробників починає розраховувати необхідний час розробки, це середньо впливає на процес розробки.	Помилка розрахунку часу виконання = Вл (2) * Вага (2) = 4

Зробимо якісний аналіз ризиків і складемо детальний опис ризиків з самими великими рангами, а саме рангами 6 і 9. Під ранг 6 і 9 у нас підпадають такі причини:

- кваліфікація;
- застарілі технології;
- довгий освоєння технології;
- проблем зі здоров'ям;
- нерозуміння завдання;
- поганий збір і аналіз даних;
- помилка розрахунку часу виконання.

Результатом аналізу ризиків є їх опис, опис представлено в табл. 2.8 – 2.14

Таблиця 2.8 – Картка опису ризику

Номер: R-1	Категорія: Організаційний
Причина: Кваліфікація	Умови: Брак знань в тій чи іншій області
Слідство: Низька продуктивність роботи	Вплив: Збільшення витрат часу на розробку
Вірогідність: Дуже ймовірно	Ступінь впливу: Критичний
Близькість: Дуже близько	Ранг:9
Вхідні дані: «Зміст проекту», «План забезпечення ресурсами»	

Таблиця 2.9 – Картка опису ризику

Номер: R-2	Категорія: Технологічний
Причина: Застарілі технології	Умови: Вибір поганої технології розробки на організаційному етапі
Слідство: Низька продуктивність роботи	Вплив: Збільшення витрат часу і сил на розробку
Вірогідність: Дуже ймовірно	Ступінь впливу: Критичний
Близькість: Дуже близько	Ранг:9
Вхідні дані: «Зміст проекту», «План забезпечення ресурсами»	

Таблиця 2.10 – Картка опису ризику

Номер: R-3	Категорія: Технологічний
Причина: Довге освоєння технології	Умови: Вибір поганої технології розробки на організаційному етапі
Слідство: Низька продуктивність роботи	Вплив: Збільшення витрат часу і сил на розробку
Вірогідність: Дуже ймовірно	Ступінь впливу: Критичний

Продовження таблиці 2.10

Близькість: Дуже близько	Ранг:9
Вхідні дані: «Зміст проекту», «План забезпечення ресурсами»	

Таблиця 2.11 – Картка опису ризику

Номер: R-4	Категорія: Технологічний
Причина: Проблем зі здоров'ям	Умови: Усе що може вплинути на імунну систему людини
Слідство: Неможливість виконання проекту	Вплив: Вплив на проект в цілому
Вірогідність: Дуже ймовірно	Ступінь впливу: Критичний
Близькість: Дуже близько	Ранг:9
Вхідні дані: «Зміст проекту», «План забезпечення ресурсами»	

Таблиця 2.12 – Картка опису ризику

Номер: R-5	Категорія: Організаційний
Причина: Нерозуміння завдання	Умови: Несвоєчасні консультації з дипломним керівником
Слідство: Низька продуктивність роботи	Вплив: Збільшення витрат часу і сил на розробку
Вірогідність: Дуже ймовірний	Ступінь впливу: Критичний
Близькість: Дуже скоро	Ранг:6
Вхідні дані: «План забезпечення ресурсами»	

Таблиця 2.13 – Картка опису ризику

Номер: R-6	Категорія: Організаційний
Причина: Поганий збір даних	Умови: Несвоєчасні консультації з дипломним керівником
Слідство: Низька продуктивність роботи	Вплив: Збільшення витрат часу і сил на розробку
Вірогідність: Дуже ймовірний	Ступінь впливу: Критичний
Близькість: Дуже скоро	Ранг:6
Вхідні дані: «План забезпечення ресурсами»	

Таблиця 2.14 – Картка опису ризику

Номер: R-7	Категорія: Організаційний
Причина: Поганий аналіз даних	Умови: Несвоєчасні консультації з дипломним керівником
Слідство: Низька продуктивність роботи	Вплив: Збільшення витрат часу і сил на розробку
Вірогідність: Дуже ймовірний	Ступінь впливу: Критичний
Близькість: Дуже скоро	Ранг:6
Вхідні дані: «План забезпечення ресурсами»	

Заплановані операції з реагування на ризики повинні відповідати серйозності ризику, бути економічно ефективними в рішенні проблеми, своєчасними, реалістичними в контексті проекту і узгодженими з усіма учасниками.

Можливі чотири методи реагування на ризики:

- ухилення від ризику (risk avoidance)
- передача ризику (risk transference)
- зниження ризику (risk mitigation)
- прийняття ризику (risk acceptance).

Ухилення від ризику передбачає зміну плану управління проектом таким чином, щоб виключити загрозу, викликану негативним ризиком, захистити мету проекту від наслідків ризику або послабити мету, що знаходяться під загрозою. Деякі ризики, що виникають на ранніх стадіях проекту, можна уникнути за допомогою уточнення вимог, отримання додаткової інформації або проведення експертизи.

Передача ризику - це перекладення негативних наслідків загрози з відповідальністю за реагування на ризик на третю сторону. Передача ризику просто переносить відповідальність за його управління іншій стороні, але ризик при цьому нікуди не дівається. Передача ризику практично завжди передбачає виплату премії за ризик стороні, що приймає на себе ризик.

Зниження ризиків - це зниження ймовірності та / або наслідків негативного ризикованого події до прийнятних меж. Вживання заходів щодо зниження ймовірності настання ризику або його наслідків часто виявляються більш ефективними, ніж зусилля щодо усунення негативних наслідків, що вживаються після настання події ризику.

І, нарешті, прийняття ризику означає, що команда проекту усвідомлено прийняв рішення не змінювати план управління проектом в зв'язку з ризиком або не виявила відповідної стратегії реагування. Команда буде змушена приймати всі «невідомі ризики».

Реакція на ризики представлена в табл. 2.15

Таблиця 2.15 – Реакція на ризики

Номер картки	Реакція	Опис
R-1	Зниження ризику	Аналіз аналогів розробки з метою набору досвіду.
R-2	Прийняття ризику	Не знайдено способів рішення проблеми
R-3	Прийняття ризику	Не знайдено способів рішення проблеми
R-4	Прийняття ризику	Не знайдено способів рішення проблеми
R-5	Усунення ризику	Перерозподіл графіку часу для своєчасного консультування
R-6	Усунення ризику	Перерозподіл графіку часу для своєчасного консультування
R-7	Усунення ризику	Перерозподіл графіку часу для своєчасного консультування

2.3 Розробка плану робіт проекту

Планування робіт проекту слід починати з визначення процесів, які повинні бути реалізовані в проекті. Відповідно до стандарту ISO / IEC 12207 всі процеси життєвого циклу програмного забезпечення розділені на групи (рис. 2.2).

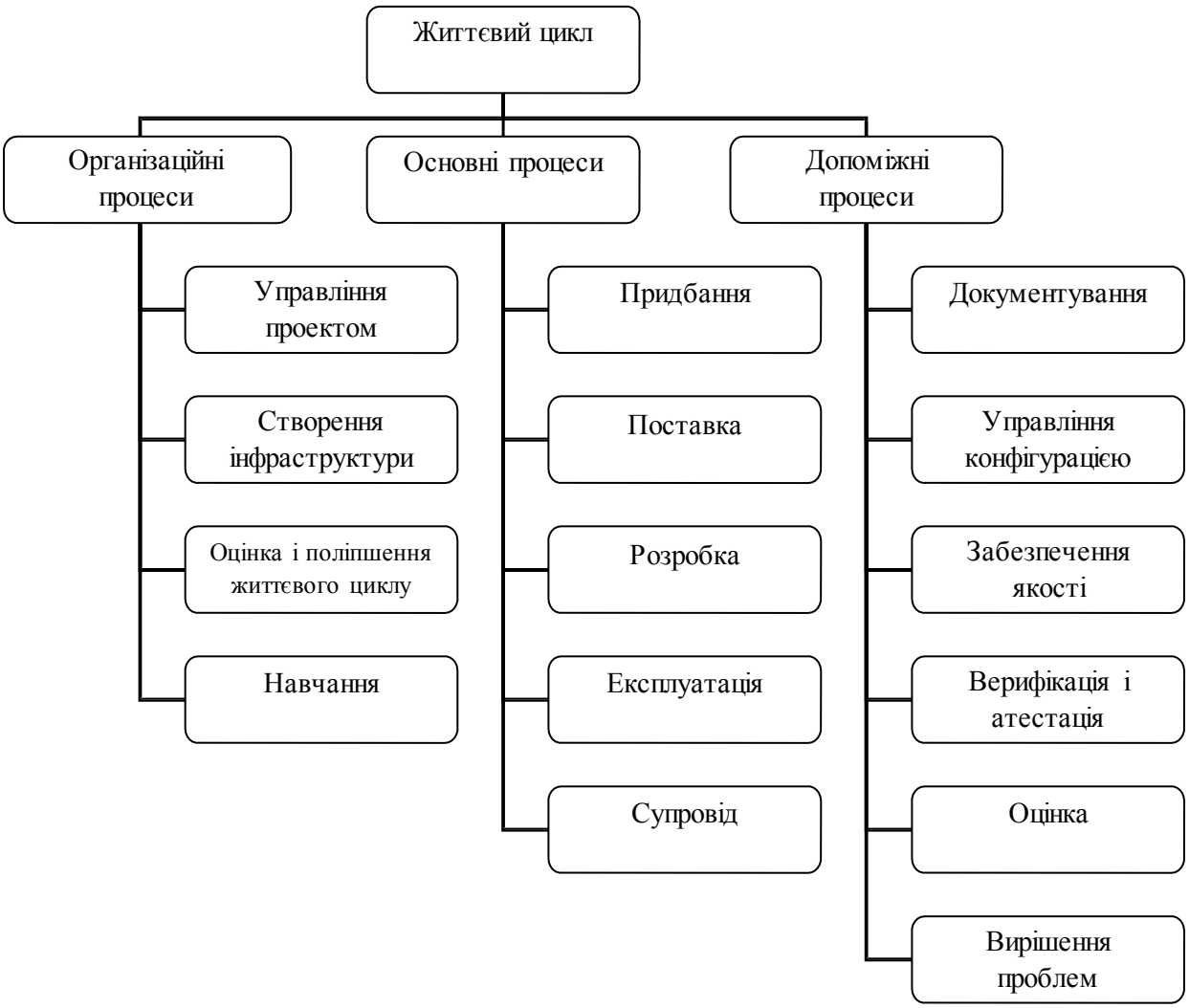

Рисунок 2.2 – Процеси життєвого циклу програмного забезпечення

Для планування робіт, пов'язаних з виконанням дипломної роботи, обов'язковим безліччю процесів є процеси розробки або супроводу, навчання, управління проектом, документування та забезпечення якості. Решта процеси вводиться в розгляд при необхідності.

Процес розробки визначає дії і завдання, які виконуються розробником в процесі створення програмного забезпечення та його компонентів відповідно до заданих вимог. Декомпозиції робіт процесу розробки пропонується виконувати на основі варіантів використання.

Декомпозиції робіт організаційних і допоміжних процесів пропонується виконувати з групами процесів. Декомпозицію (WBS) будемо виконувати в програмі Microsoft Project.

На рисунку 2.3 зображена ієрархічна структура WBS.

	Режим задачи	Название задачи	Длительнс	Начало	Окончани
1		**Специфікація вимог до програмної системи**	26 дней	Пн 18.01.16	Пн 22.02.16
2		Аналіз предметної області	6 дней	Пн 18.01.16	Пн 25.01.16
3		Аналіз програмних аналогів системи	5 дней	Вт 26.01.16	Пн 01.02.16
4		Визначення прецендентів системи	15 дней	Вт 02.02.16	Пн 22.02.16
5		**Планування проекту**	10 дней	Вт 23.02.16	Пн 07.03.16
6		Розрахунок методом UCP	5 дней	Вт 23.02.16	Пн 29.02.16
7		Побудова плану проекту	5 дней	Вт 01.03.16	Пн 07.03.16
8		**Проектування системи**	19 дней	Вт 08.03.16	Пт 01.04.16
9		Вибір інструментарію для розробки сисеми	6 дней	Вт 08.03.16	Вт 15.03.16
10		Проектування графічного інтерфейсу користувача	6 дней	Ср 16.03.16	Ср 23.03.16
11		Проектування архітектури системи	7 дней	Ср 23.03.16	Пт 01.04.16
12		**Реалізація програмної системи**	41 дней	Пт 01.04.16	Пн 30.05.16
13		Реалізація бек-енду програми	32 дней	Пн 30.05.16	Ср 13.07.16
14		Реалізація фронт-енду програми	9 дней	Ср 13.07.16	Вт 26.07.16
15		**Тестування програми**	5 дней	Пн 30.05.16	Пн 06.06.16
16		Розробка тест-кейсів	3 дней	Пн 06.06.16	Чт 09.06.16
17		Проведення тестування	2 дней	Чт 09.06.16	Пн 13.06.16
18		**Введення в експлуатацію**	41 дней	Пн 13.06.16	Вт 09.08.16

Рисунок 2.3 – Ієрархічна структура WBS, розроблена в Microsoft Project

У сумі на розробку необхідно витратити 101 день що приблизно дорівнює 3 місяцям. В результаті отримуємо такий розподіл часу.

Розподіл часу представлено в табл. 2.16

Таблиця 2.16 - Розподіл часу

Елемент WBS першого рівня	Стандартний розподіл
Етап підготовки	15%
Організаційні процеси	5%
Реалізація	60%
Завершення	20%

Для наочності побудуємо діаграму Ганта. Діаграма Ганта є графіком, на якому по горизонталі розміщена шкала часу, а по вертикалі розташований список завдань (рис. 2.4). Довжина відрізків, що позначають завдання, пропорційна тривалості завдань. Поруч з відрізками може відображатися додаткова інформація, наприклад, назви задіяних в них ресурсів. Склад діаграми визначається її налаштуванням.

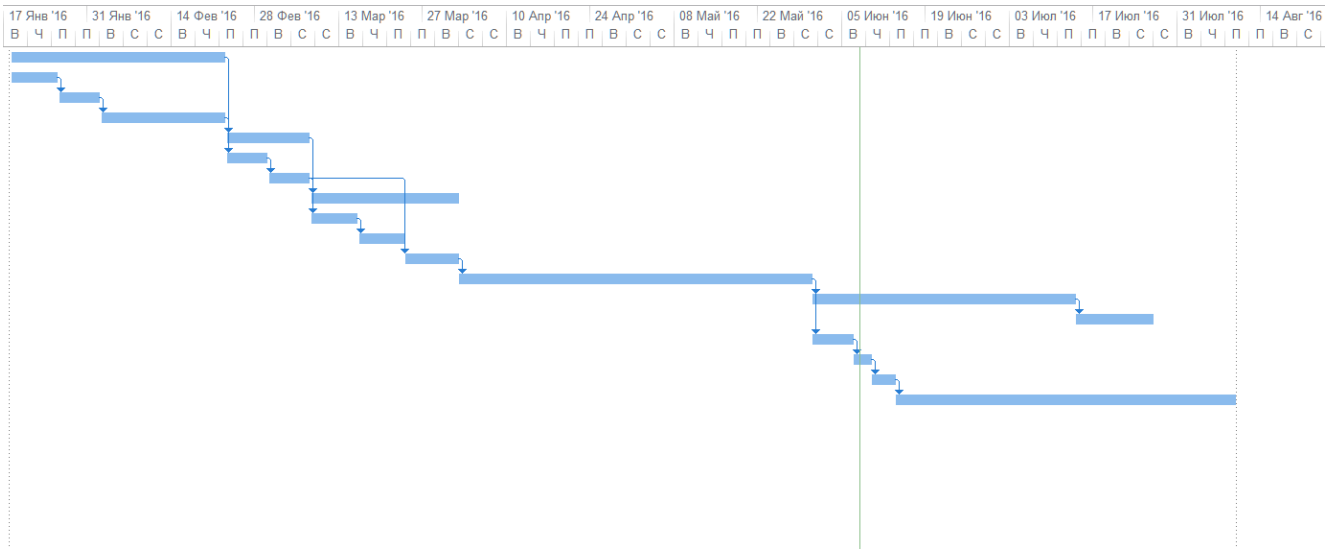

Рисунок 2.4 – Діаграма Ганта

3 ПРОЕКТУВАННЯ СИСТЕМИ

3.1 Проектування архітектури системи

Для того щоб побачити, які об’єкти беруть участь во взаємодії, і якими повідомленнями обмінюються для досягнення мети треба побудувати діаграми взаємодій.

Діаграма взаємодії - це діаграма, на якій представлено взаємодія, що складається з безлічі об'єктів і відносин між ними, включаючи і повідомлення, якими вони обмінюються. Цей термін застосовується до видів діаграм з акцентом на взаємодії об'єктів (діаграмах кооперації, послідовності і діяльності).

На діаграмах послідовностей увага акцентується насамперед на тимчасовій впорядкованості повідомлень. Це дає читачеві наочну картину, що дозволяє зрозуміти розвиток потоку управління в часі.

Діаграма кооперації акцентує увагу на організації об'єктів, які беруть участь у взаємодії. Це дає користувачеві ясне візуальне уявлення про по струмі управління в контексті структурної організації кооперуються об'єктів.

Діаграми діяльності дозволяють моделювати складний життєвий цикл об'єкта, з переходами з одного стану (діяльності) в інше. Також цей вид діаграм може бути використаний і для опису динаміки сукупності об'єктів. Ми будемо використовувати саме діаграму діяльності.

Діаграми діяльності зображені на рисунках 3.1 – 3.5

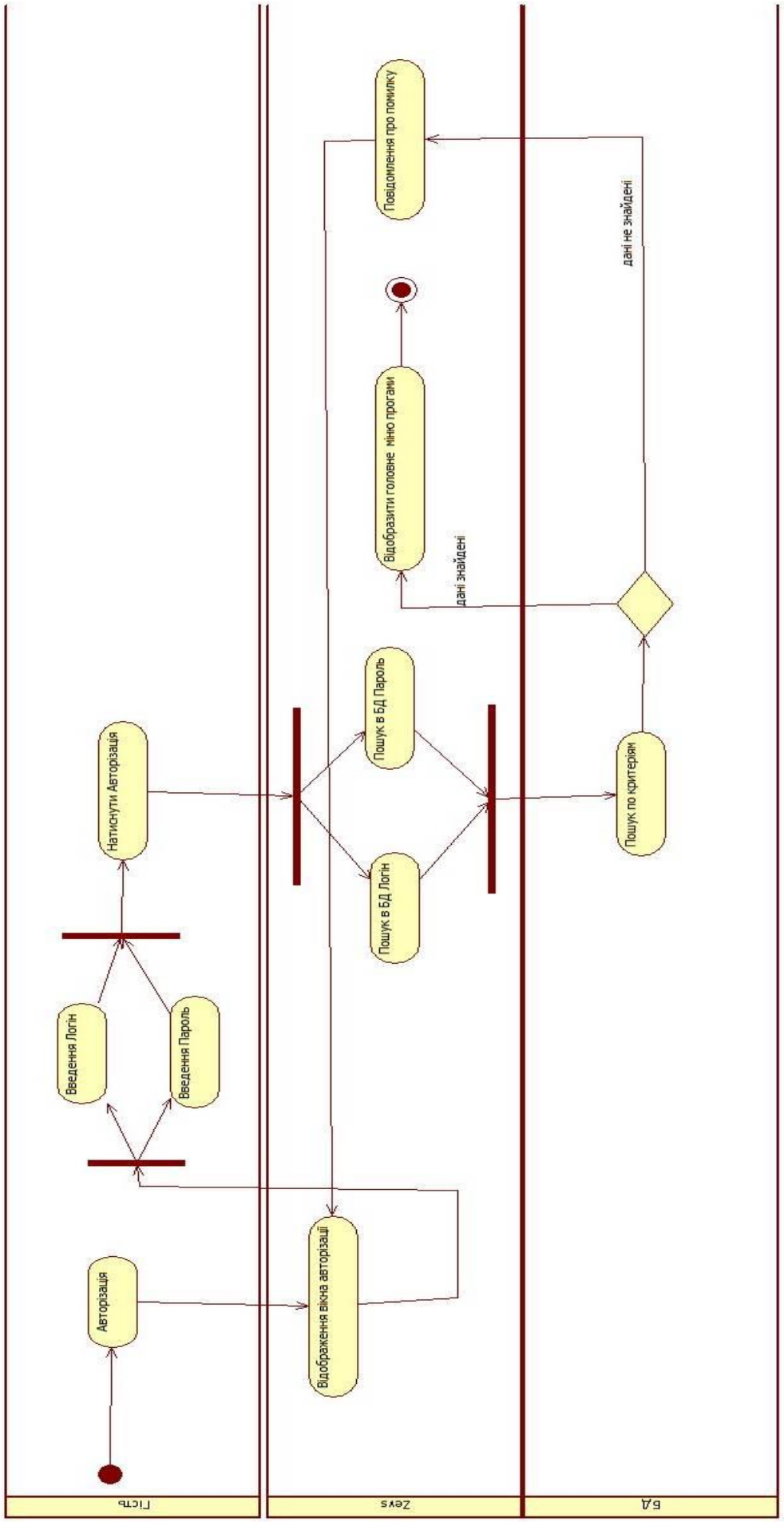

Рисунок 3.1 – Діаграма діяльності «Авторизація»

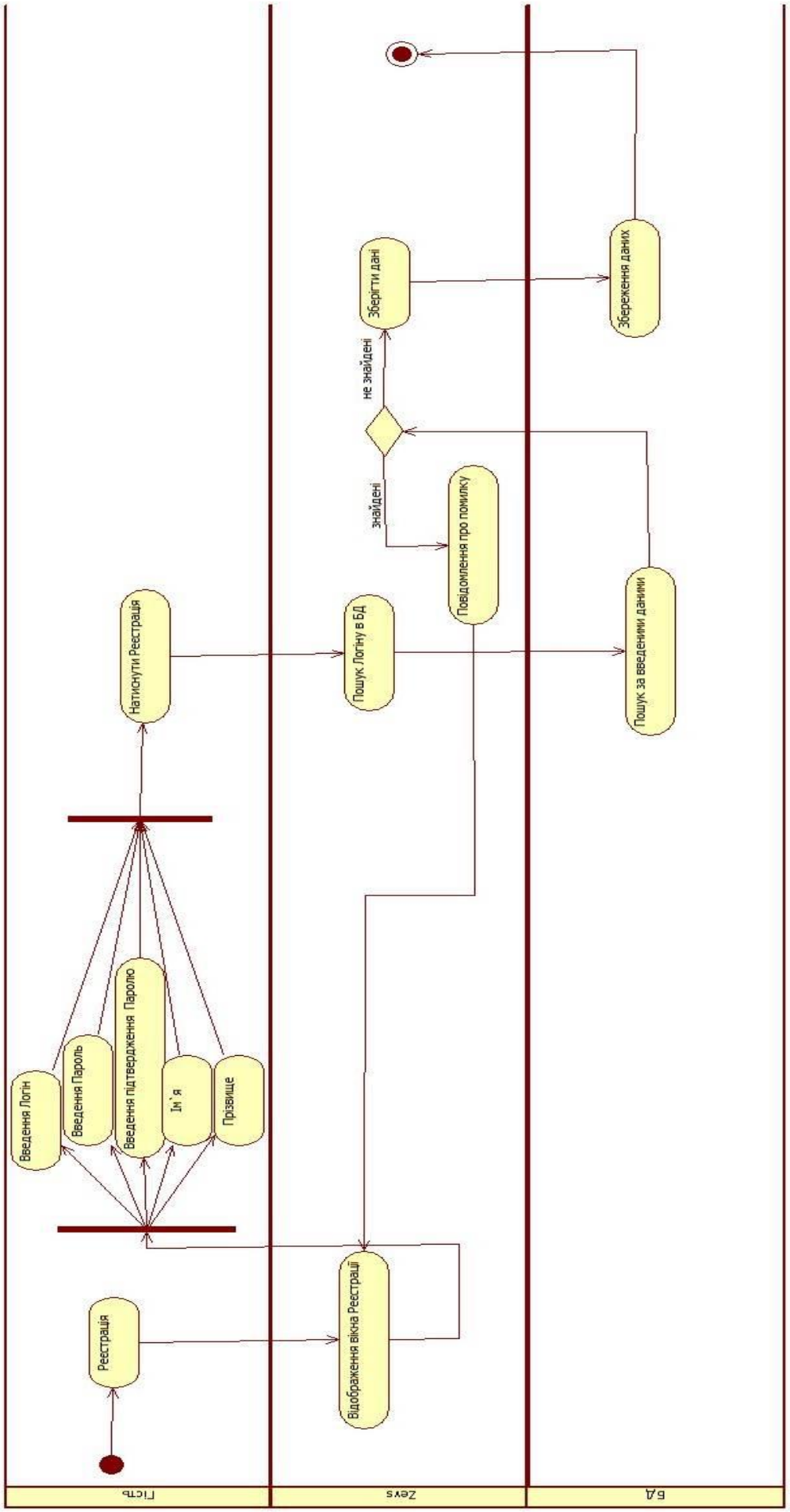

Рисунок 3.2 – Діаграма діяльності «Реєстрація»

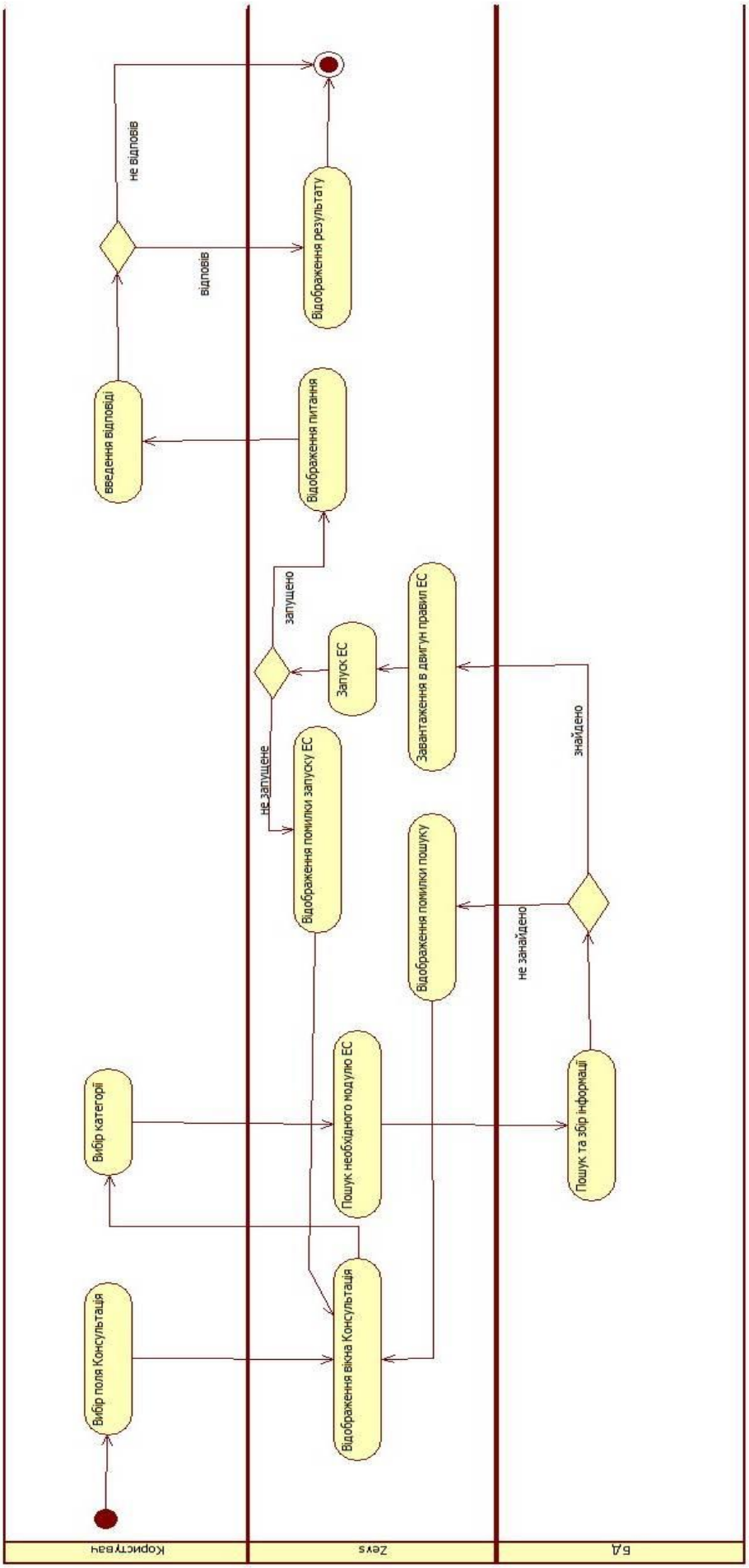

Рисунок 3.3 – Діаграма діяльності «Пошук інформації з електробезпеки»

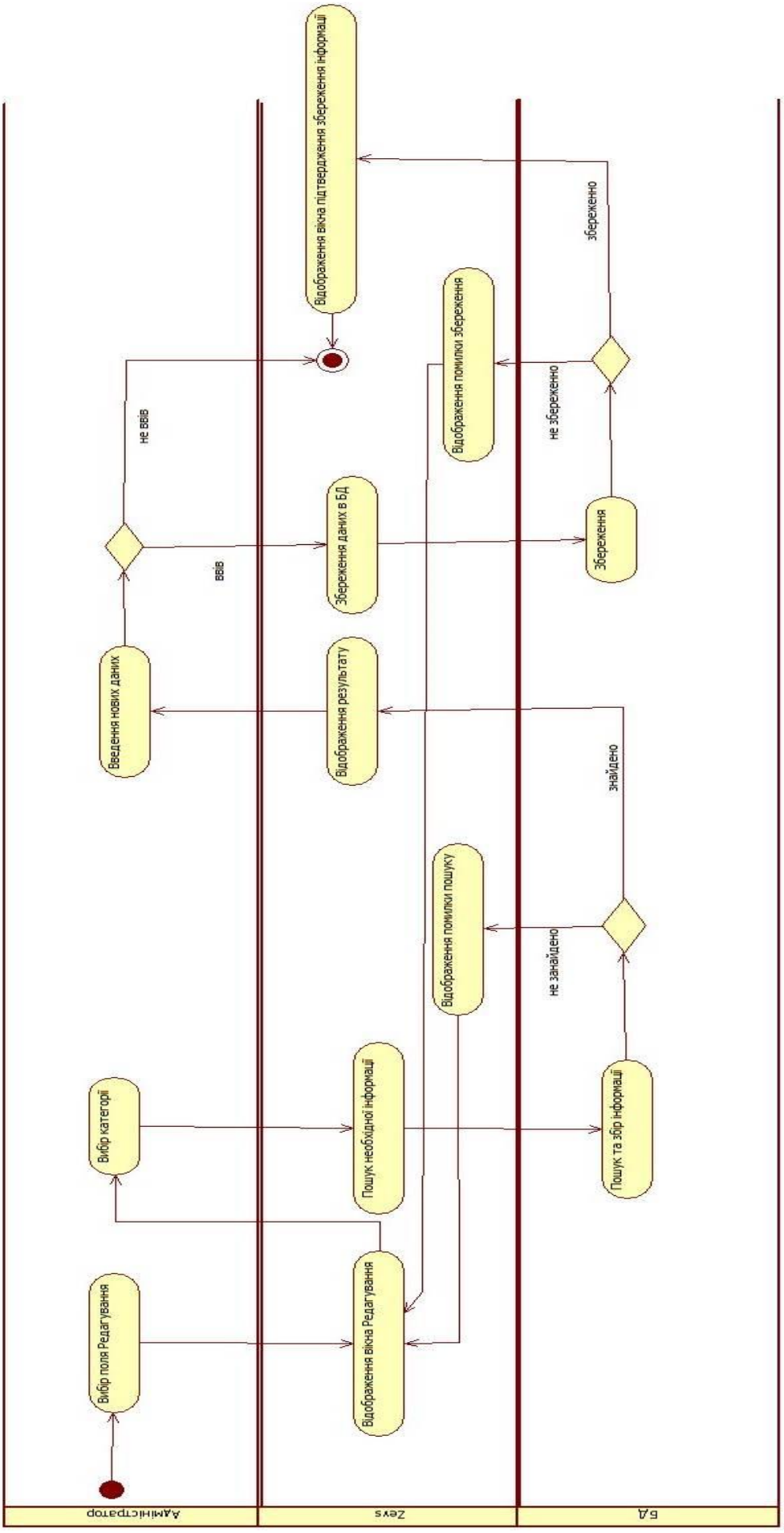

Рисунок 3.4 – Діаграма діяльності «Редагування»

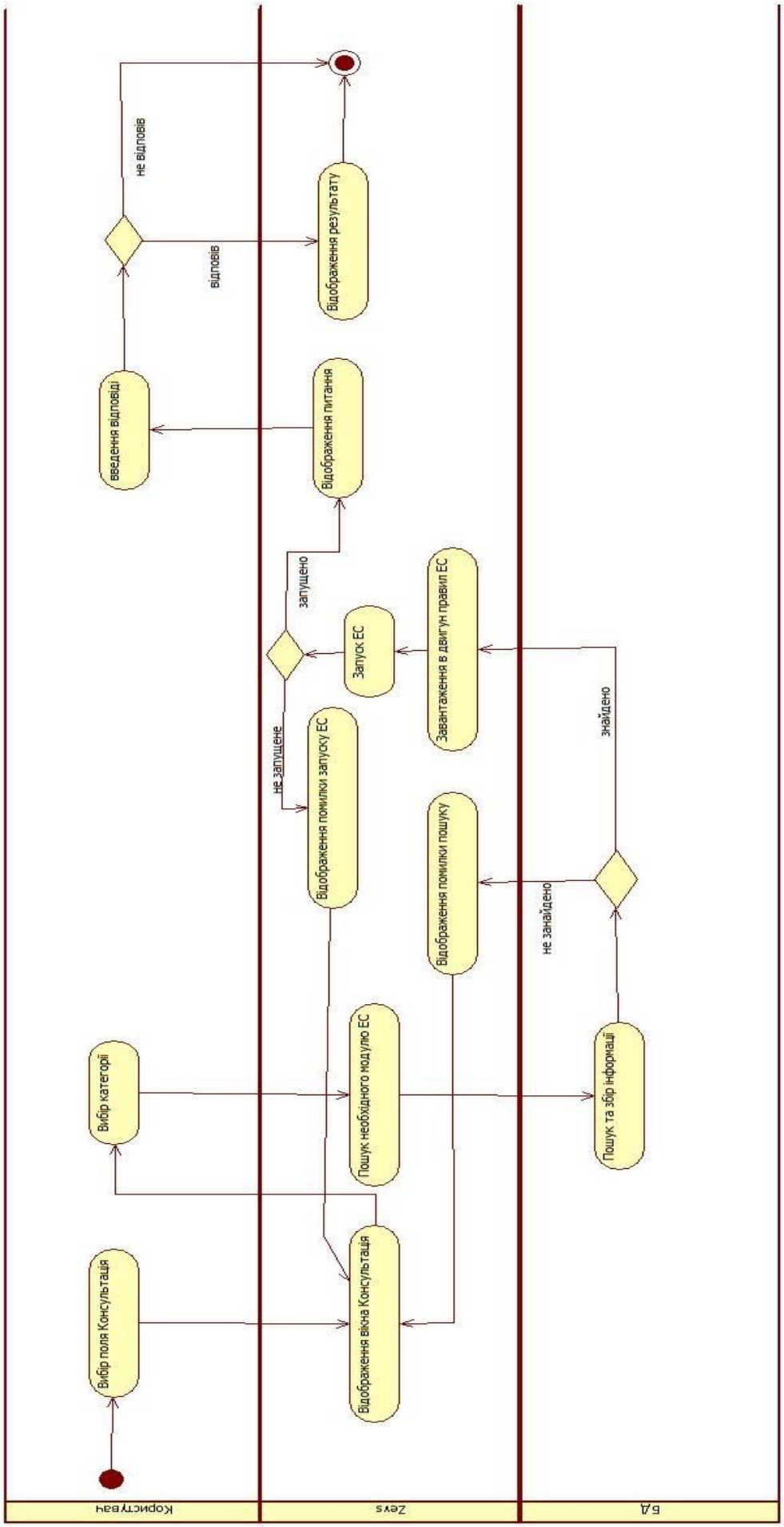

Рисунок 3.5 - Діаграма діяльності «Консультація з електробезпеки»

3.2 Розробка діаграми програмних класів

Так як система з електробезпеки використовуватиметься звичайними користувачами, найважливішим критерієм буде простий інтуїтивно зрозумілий інтерфейс. Для нашої системи будемо використовувати такий патерн: Model-View-Controller (MVC).

Патерн Model-View-Controller основна мета застосування цієї концепції полягає у відділенні бізнес-логіки (моделі) від її візуалізації (уявлення, виду). За рахунок такого поділу підвищується можливість повторного використання. Найбільш корисно застосування даної концепції в тих випадках, коли користувач повинен бачити ті ж самі дані одночасно в різних контекстах та / або з різних точок зору.

Виходячи з опису прецедентів були виявлені наступні програмні класи:

- Registration;
- ConnnectionDB;
- Authorization;
- Workspace;
- CheckData.

На рис. 3.6 зображенно діаграмму програмних класів розроблюваної системи.

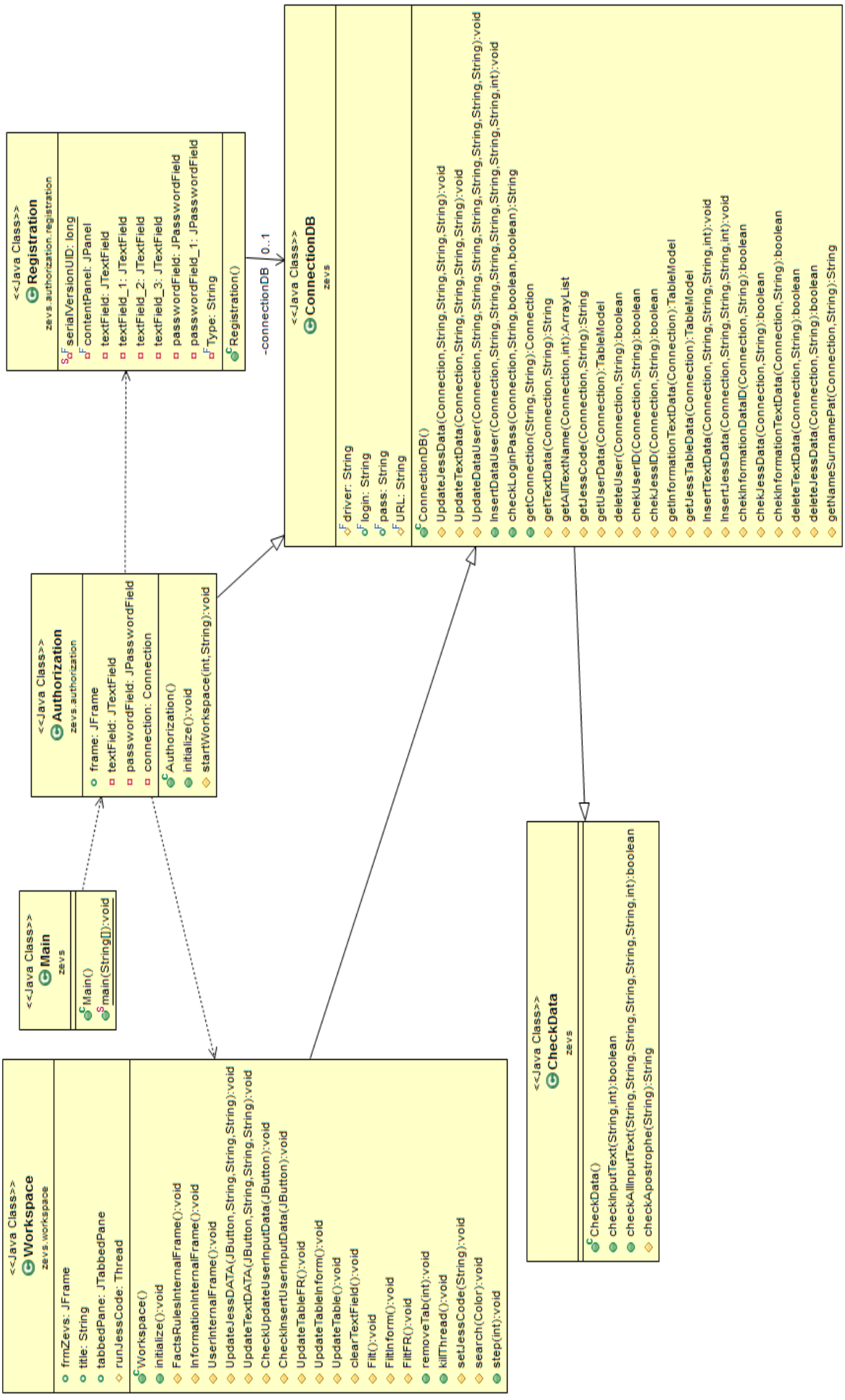

Рисунок 3.6 – Діаграма програмних класів

Для кращого розуміння даної діаграми зробимо її опис.

Клас Registration – містить у собі необхідний функціонал для реєстрації нового користувача системи.

Клас Workspace – відповідає за відображення головного робочого вікна програми, саме з ним користувач буде процювати писля авторизації.

Клас Authorization – містить у собі функціонал для авторизації раніше зареєстрованого користувача системи.

Клас ConnnectionDB – відповідає за весь необхідний функціонал для роботи с базою даних.

Клас CheckData – містить у собі функціонал для перевірки введених користувачем даних.

Система ZEVS використовує реляційну базу даних MySQL, в ній зберігаються всі необхідні данні для коректної роботи програми а саме:

Таблиця користувачів – ця таблиця зберігає в собі всіх користувачів системи, саме з нею виконаються операції реєстрації та авторизації.

Таблиця експертної системи – ця таблиця зберігає у собі правила експертної, які написані мовою CLIPS та які згодом будуть завантаженні в двигун ЕС JESS. Використовується для проведення консультації користувача.

Таблиця довідкового матеріалу – ця таблиця зберігає у собі весь довідковий матеріал з електробезпеки, який користувач може використовувати для ознайомлення.

На рис. 3.7 зображена структура реляційної бази даних ZEVS.

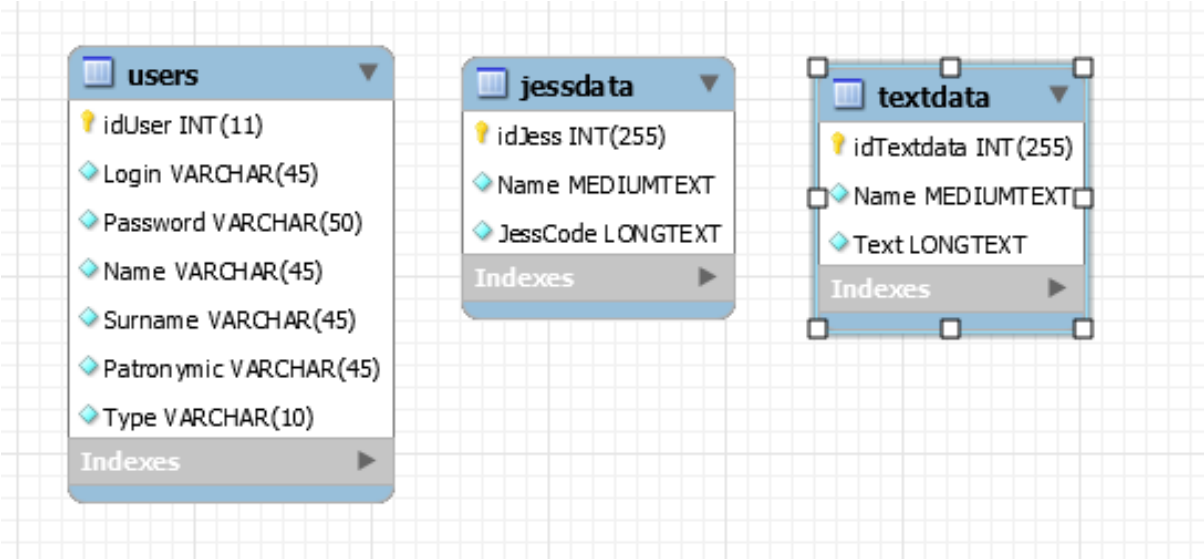

Рисунок 3.7 – Структура реляційної бази даних.

Як видно на рис. 3.7 між таблицями відсутній будь-який зв'язок, вони повністю незалежні один від одного.

3.3 Проектування алгоритмів

Одним із самих складних алгоритмів системи є пошук інформації по ключовому слову. Як тільки користувач ввів ключове слово, програма починає аналізувати текст в формі, якщо знаходиться збіг, вона повертає номер позиції найденого слова і виділяє його жовтім кольором. Цей процес повторюється доки в процесі пошуку не вернеться -1.

Блок-схему алгоритму пошуку по ключовому слову зображено на рис. 3.8

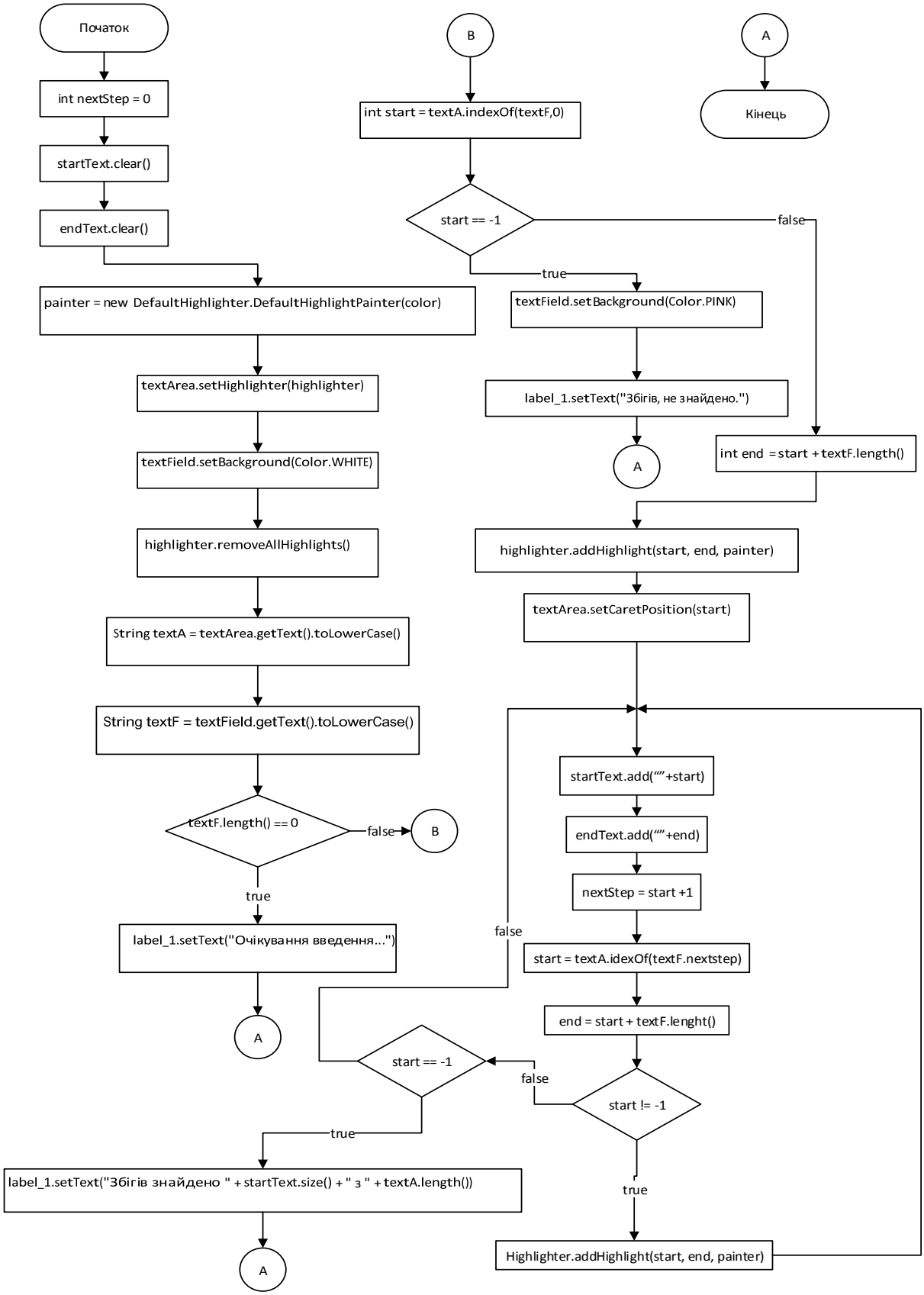

Рисунок 3.8 – Блок-схема алгоритму пошуку по ключовому слову

3.4 Проектування інтерфейсу користувача

Дуже часто розробка інтерфейсу - це вже запорука успіху. І не так важливо, який у вас продукт. У всіх випадках головне завдання інтерфейсу - зробити взаємодію користувача з продуктом якомога простішим.

Тому було вирішено розробити якомога простіший інтерфейс користувача. При проектуванні інтерфейсу системі були використанні деякі принципи:

- всі елементи програми мають стандартну кольорову палітру та вид
- для зручності читання текстової інформації було вирішено додати функціонал змінення типу та розміру шрифту.

Далі розглянемо декілька графічних інтерфейсів користувача.

На рис. 3.9 зображено вікна авторизації та реєстрації.

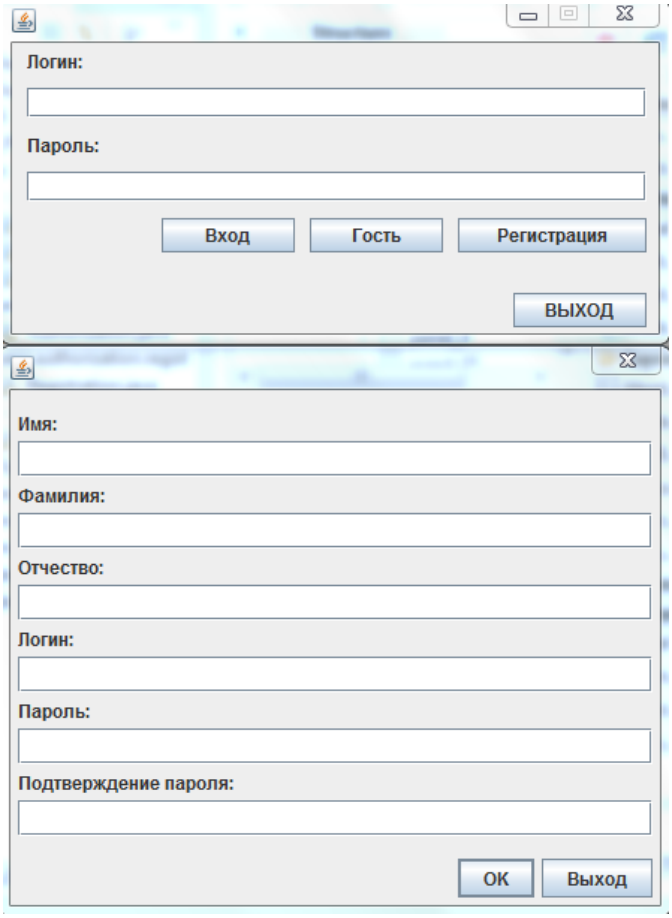

Рисунок 3.9 – Вікна авторизації та реєстрації

Після авторизації ми бачимо головне вікно програми, для того щоб весь робочій інтерфейс відобразився ввійдемо під адміністратором. На рис. 3.9 зображено головне робоче вікно програми.

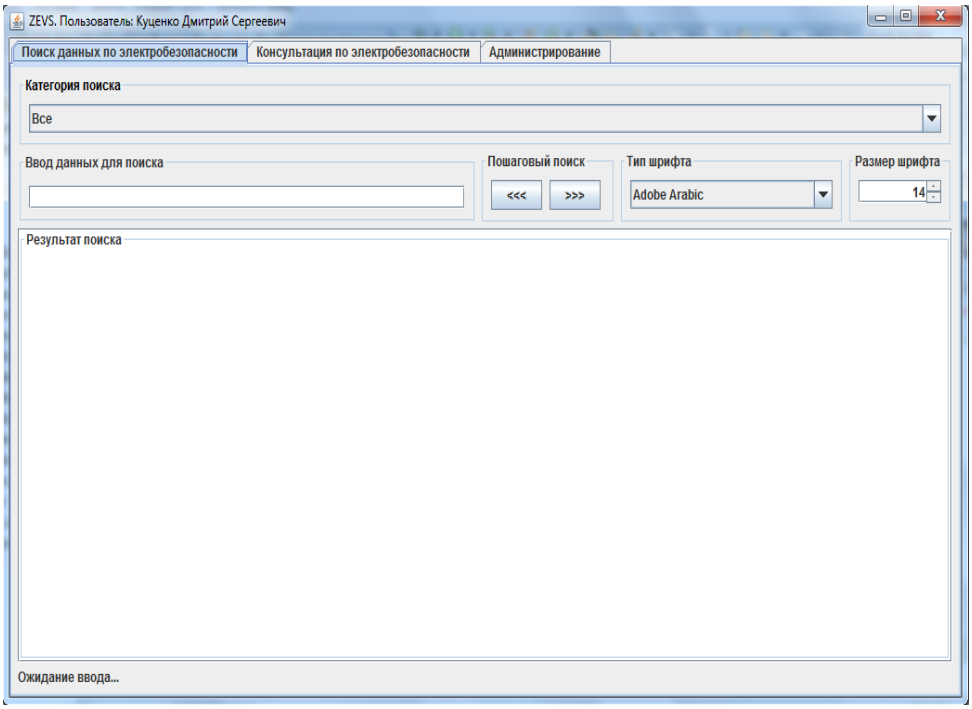

Рисунок 3.10 – Головне робоче вікно програми

Як видно на рис. 3.10 робоче вікно дуже просте і інтуїтивно зрозуміле, також присутні функції змінення типу і розміру шрифту для зручності читання інформації з електробезпеки.

Далі розглянемо інтерфейс консультації користувача (див. рис. 3.11). Для цього потрібно натиснути на вкладку «Консультація з електробезпеки».

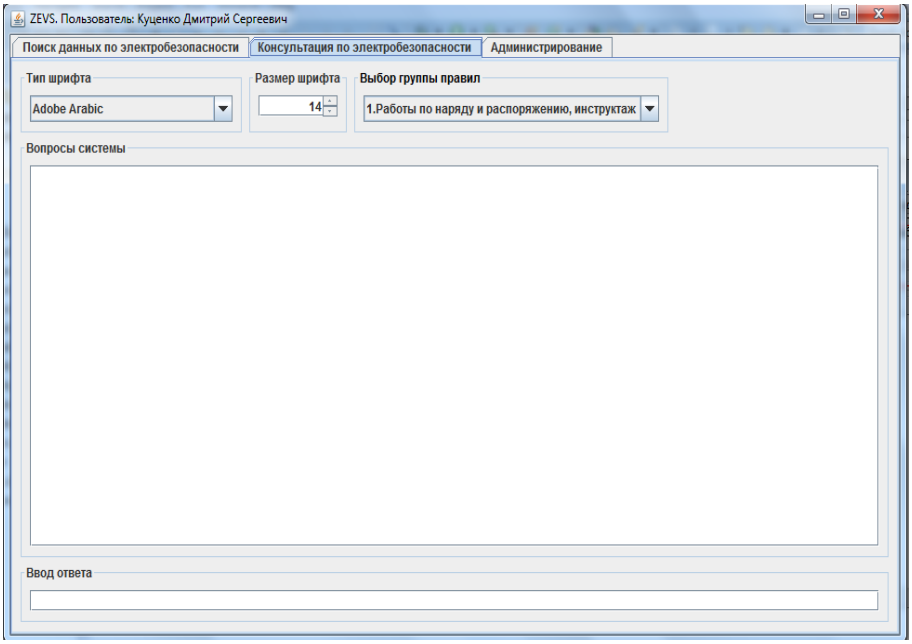

Рисунок 3.11 – Інтерфейс консультації користувача

Як і в попередньому випадку інтерфейс дуже простий і зрозумілий також присутні функції змінення тексту .

Далі зображено інтерфейс адміністратора програми, він трохи відрізняються але він такий же простий. Для того щоб відобразились вікна редагування, спочатку потрібно ввести логін та пароль адміністратора бази даних (див. рис. 3.12).

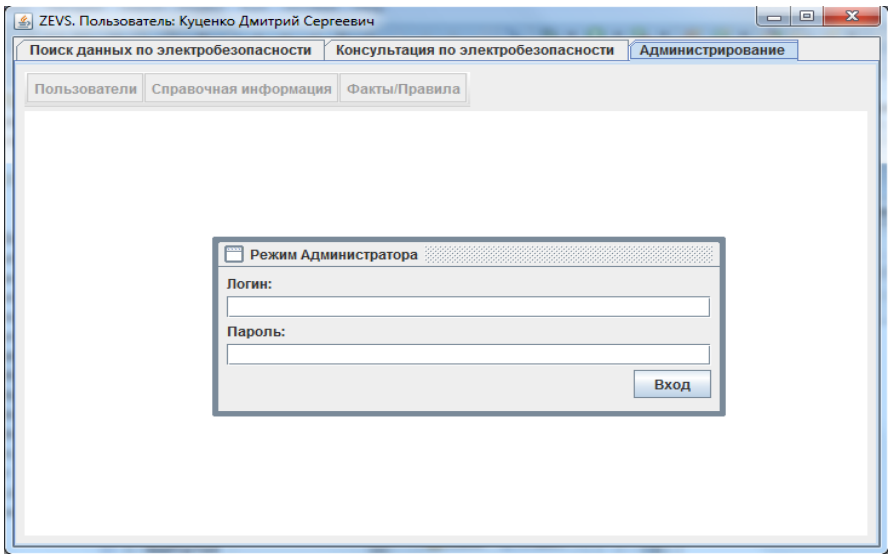

Рисунок 3.12 – Вікно входу в режим адміністратора

Після того як буде введений коректний логін та пароль відобразяться три вікна для редагування даних системі. Головна відмінність інтерфейсу в тому що кожне із вікон знаходиться всередині головного вікна і є незалежними один від одного (див. рис. 3.13).

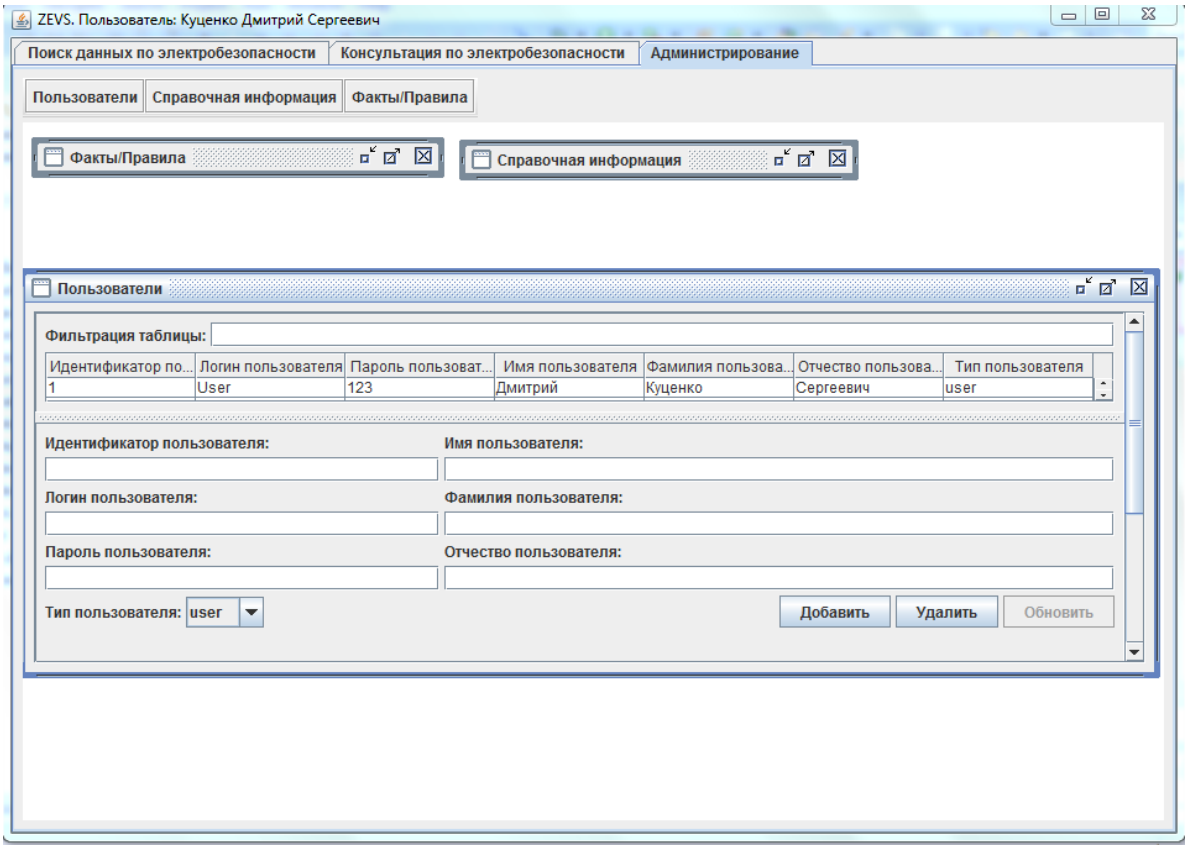

Рисунок 3.13 – Вікна редагування інформації бази даних

З допомогою функції адміністрування можна редагувати користувачів, довідкову інформацію та факти і правила експертної системі.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ

4.1 Опис програмних технологій

Розроблювана програма представляє собою клієнт/серверну систему що для виконання використовує JVM. Так як програма використовує JVM, перед використанням її потрібно завантажити з офіційного сайту. В процесі розробки використовувалась найостанніша версія JDK що дає можливість використовувати самі актуальні можливості Java SE. Розробка велась мовою Java та JESS (CLIPS).

CLIPS (від англ. C Language Integrated Production System). CLIPS є однією з найбільш широко використовуваних інструментальних середовищ для розробки експертних систем завдяки своїй швидкості, ефективності і безкоштовності.

Для розробки графічного інтерфейсу використовувалась стандартна бібліотека Java Swing та плагін WindowBuilder.

Swing - інструментарій для создания графічного інтерфейсу користувача (GUI) мовою програмування Java.

Swing розробляли для забезпечення функціонального набору програмних компонентів для создания графічного інтерфейсу користувача, значний його плюс універсальність інтерфейсу створення програм на всіх платформах.

WindowBuilder - візуальний дизайнер інтерфейсів на Swing, GWT та GWT. Для платформи Eclipse виконаний у вигляді плагіна. На даний момент WindowsBuilder поширюється вільно.

4.2 Опис програмних бібліотек

Jess (Java Expert System Shell) - це оболонка для розробки експертних систем повністю написаний на мові Java. Використовуючи Jess, можна побудувати Java програму з можливістю обробки даних на основі знань, представлених у вигляді правил. На даний момент, Jess - один з найбільш легких і швидких оболонок для експертних систем.

Скриптова мова Jess все ще сумісна з CLIPS, тобто більшість скриптів Jess

працюватимуть в CLIPS і навпаки. Як і CLIPS, ядро Jess використовує Rete алгоритм для обробки фактів і правил, який є дуже ефективним і швидкодіючим при вирішенні завдань множинного порівнювання (many-to-many matching problem). Jess також включає в себе можливість створення, управління і виклику методів Java об'єктів.

JDBC (англ. Java DataBase Connectivity) - переносний незалежний промисловий стандарт взаємодії Java-додатків з різними СУБД, реалізований у вигляді пакета java.sql, що входить до складу Java SE.

JDBC заснований на концепції так званих драйверів, що дозволяють отримувати з'єднання з базою даних по спеціально описаного URL. Драйвери можуть завантажуватися динамічно (під час роботи програми). Завантажившись, драйвер сам реєструє себе і викликається автоматично, коли програма вимагає URL, що містить протокол, за який драйвер відповідає.

4.2 Опис мови JESS

Система Jess як і Clips, заснована на правилах. Jess є декларативною мовою, тобто програма складається з набору інструкцій. Так само можна вважати, що Jess є інтерактивним інтерпретаторів. Система може працювати в режимі реального часу і - в пакетному режимі. Пакетний режим передбачає, що один і кілька файлів можуть працювати і виконуватися одночасно. Структура Jess представлена на рис. 4.1.

В основі механізму виведення Jess лежить Rete-алгоритм, який підвищує ефективність і швидкодію на завданнях множинного порівняння. Rete-алгоритм являє собою дуже швидкий засіб порівняння з шаблонами. Алгоритм запам'ятовує результат останнього тестування знань і заново перевіряє тільки знову з'явилися факти.

Даний алгоритм реалізований за допомогою побудови системи вузлів, кожен з яких представляє одне або кілька перевірок фактів для кожного правила.

Факт, який додається або видаляється з бази знань, обробляється цією системою вузлів. В основі цієї системи лежать конкретні дії.

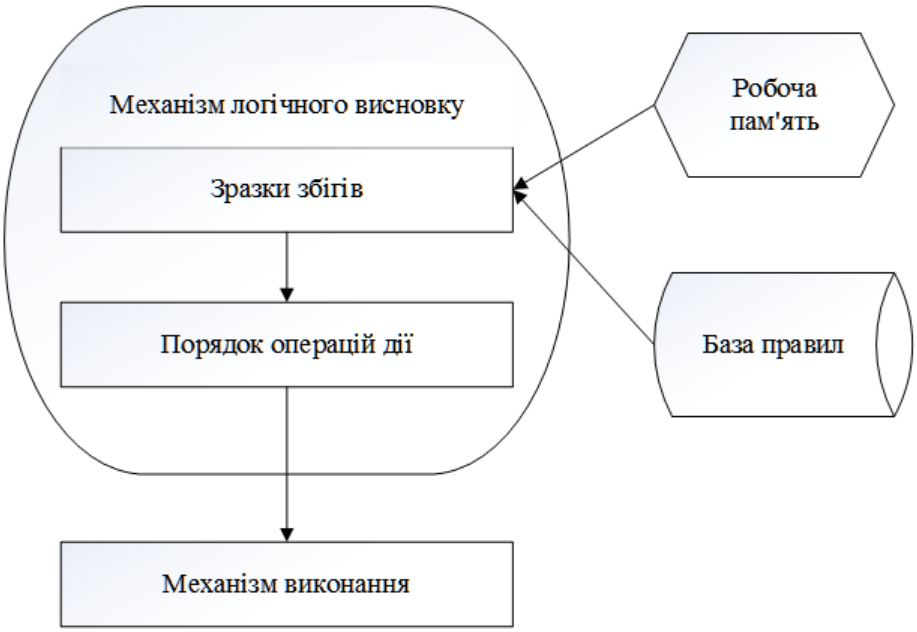

Рисунок 4.1 – Схема роботи системи Jess

Подання знань і накопичення баз знань в системі Jess відбувається за допомогою правил і фактів. Розглянемо коротко їх синтаксис. Синтаксис факту наведено на рис. 4.2. Синтаксис і вид записи правила в Jess праведний на рис. 4.3. Фактично в цьому випадку записується факт і відповідна послідовність виконання функцій.

```
(deftemplate <name> [<comment>] (slot| multislot <slotname>
(<slotoptions>)*)*)
```

Рисунок 4.2 – Запис факту на мові Jess

```
(defrule <name> [<comment>]<fact>* => <function>*)
```

Рисунок 4.3 – Запис правила на мові Jess

5 ТЕСТУВАННЯ СИСТЕМИ

5.1 Функціональне тестування

Тестування програмного забезпечення – це процес перевірки відповідності заявлених до продукту вимог і реально реалізованої функціональності, здійснюваний шляхом спостереження за його роботою в штучно створених ситуаціях і на обмеженому наборі тестів, обраних певним чином.

Існують два принципи тестування програми:

- функціональне тестування (тестування «чорного ящика»);
- структурне тестування (тестування «білого ящика»).

Тестування «чорного ящика» - один з видів тестування, спрямованого на перевірку відповідностей функціональних вимог ПО до його реальних характеристик. Основний завданням функціонального тестування є підтвердження того, що розробляється програмний продукт володіє всім функціоналом, необхідним замовником.

Тестування "білий ящик" виконується з метою виявлення проблем у внутрішній структурі програми. Це вимагає від перевіряючого глибокого знання внутрішньої структури і, отже, не може бути виконано звичайним користувачем. Загальне завдання такого тестування - забезпечити перевірку кожного кроку за алгоритмом програми.

Основна перевага всіх типів стратегій тестування "білий ящик": при тестуванні береться до уваги структура всієї програми, що полегшує виявлення помилок навіть у тому випадку, коли специфікації програмного забезпечення недостатньо певні або неповні.

У табл. 5.1 зображено множина варіантів тестування чорного ящика.

Таблиця 5.1 – Множина варіантів функціонального тестування

Прецедент	Авторизація	
Дії	Очікуваний результат	Результат тестування (вдалий(passed)/невдалий (failed))
1. Перехід до пункту авторизації.	Відображення вікна авторизації	Passed
2. Користувач вводить існуючу у БД Логін, вводить пароль раніше зареєстрованого користувача та натискає на кнопку «Авторизація»	Система шукає Логін користувача у БД Система перевіряє коректність введеного Користува-чем пароля з паролем знайденого користувача. Відображення робочого меню програми.	Passed
1. Перехід до пункту авторизації.	Відображення вікна авторизації	Passed

Продовження таблиці 5.1

Прецедент	Авторизація	
Дії	Очікуваний результат	Результат тестування (вдалий(passed)/невдалий (failed))
2. Користувач вводить існуючу у БД Логін, вводить пароль раніше зареєстрованого користувача та натискає на кнопку «Авторизація»	Система шукає Логін користувача у БД Система не знаходить Логін користувача у БД Система виводить повідомлення про помилку.	Passed
1. Перехід до пункт у авторизації.	Відображення вікна авторизації	Passed

Продовження таблиці 5.1

Прецедент	Авторизація	
Дії	Очікуваний результат	Результат тестування (вдалий(passed)/невдалий (failed))
2. Користувач вводить існуючу у БД Логін, вводить пароль раніше зареєстрованого користувача та натискає на кнопку «Авторизація»	Система шукає Логін користувача у БД Система знаходить Логін користувача у БД Система перевіряє введений пароль з наявним в БД. Паролі не співпадають. Повідомлення про помилку	Passed
Прецедент	Реєстрація	
Дії	Очікуваний результат	Результат тестування (вдалий(passed)/невдалий (failed))
1. Перехід до пункт у Реєстрація.	Відображення вікна реєстрації	Passed

Продовження таблиці 5.1

Прецедент	Реєстрація	
Дії	Очікуваний результат	Результат тестування (вдалий(passed)/невдалий (failed))
2. Користувач вводить ПІБ, логін, пароль, підтвердження паролю та натискає Зареєструватися	Перевірка даних на коректність вводу. Система шукає в БД введений логін. Система не знаходить Логін в БД. Система зберігає дані в БД Відображення повідомлення про успішну реєстрацію	Passed
1. Перехід до пункту Реєстрація.	Відображення вікна реєстрації	Passed
2. Користувач вводить ПІБ, логін, пароль, підтвердження паролю та натискає Зареєструватися	Перевірка даних на коректність вводу. Система шукає в БД введений логін. Система знаходить Логін в БД. Відображення повідомлення про помилку.	Passed

Продовження таблиці 5.1

Прецедент	Реєстрація	
Дії	Очікуваний результат	Результат тестування (вдалий(passed)/невдалий(failed))
1. Перехід до пункту Реєстрація.	Відображення вікна реєстрації	Passed
2. Користувач вводить ПІБ, логін, пароль, підтвердження паролю та натискає Зареєструватися	Перевірка даних на коректність вводу. Підтвердження паролю не співпадає. Відображення повідомлення про помилку.	Passed
1. Перехід до пункту Реєстрація.	Відображення вікна реєстрації	Passed
2. Користувач вводить ПІБ, логін, пароль, підтвердження паролю та натискає Зареєструватися	Перевірка даних на коректність вводу. Пропущене поле для вводу. Відображення повідомлення про помилку.	Passed

Продовження таблиці 5.1

Прецедент	Пошук інформації	
Дії	Очікуваний результат	Результат тестування (вдалий(passed)/невдалий (failed))
1. Користувач натиснув пункт Пошук інформації.	Відображення вікна пошуку інформації	Passed
2. Користувач вводить необхідну інформацію для пошуку.	Система шукає ключові слова. Система знаходить дані. Відображення результату пошуку.	Passed
1. Користувач натиснув пункт Пошук інформації.	Відображення вікна пошуку інформації	Passed
2. Користувач вводить необхідну інформацію для пошуку.	Система шукає ключові слова. Система не знаходить дані. Відображення результату повідомлення що дані не знайдені.	Passed

Продовження таблиці 5.1

Прецедент	Консультація	
Дії	Очікуваний результат	Результат тестування (вдалий(passed)/невдалий (failed))
1. Користувач натиснув пункт Консультація.	Відображення вікна Консультації	Passed
2. Користувач вибирає необхідну групу правил.	Система шукає в БД групу правил консультації. Система знаходить дані та завантажує їх в двигун ЕС.	Passed
3. Користувач відповідає на запитання	ЕС запам'ятовує відповідь користувача. ЕС порівнює відповіді з правилами. Система виводить результат консультації.	Passed
1. Користувач натиснув пункт Консультація.	Відображення вікна Консультації	Passed

Продовження таблиці 5.1

Прецедент	Консультація	
Дії	Очікуваний результат	Результат тестування (вдалий(passed)/невдалий (failed))
2. Користувач вибирає необхідну групу правил.	Система шукає в БД групу правил консультації. Система не знаходить дані. Відображення повідомлення про помилку	Passed
Прецедент	Редагування	
Дії	Очікуваний результат	Результат тестування (вдалий(passed)/невдалий (failed))
1. Адміністратор натиснув пункт Редагування	Відображення вікна Редагування	Passed
2. Адміністратор добавляє змінює або видаляє дані.	Система аналізує та підтверджує введення. Система зберігає у БД нові данні. Система виводить підтвердження змінення даних.	Passed

Продовження таблиці 5.1

Прецедент	Редагування	
Дії	Очікуваний результат	Результат тестування (вдалий(passed)/невдалий (failed))
1. Адміністратор натиснув пункт Редагування	Відображення вікна Редагування	Passed
2. Адміністратор добавляє змінює або видаляє дані.	Система аналізує та підтверджує введення. Система не може зберегти дані у БД . Система виводить повідомлення про помилку.	Passed

5.2 Інструкція з розгортання

Крок 1. Для початку роботи необхідно встановити JAVA, скачати його можна з сайту http://www.java.com/ru/download/windows_manual.jsp

Крок 2. Встановити MySQL Community Server, для зручності використання рекомендується встановити також MySQL Workbench. Скачати можна з офіціального сайту https://dev.mysql.com/downloads/mysql/

Крок 3. В процесі установки MySQL Community Server, буде запропоновано створити нового користувача це потрібно обов'язково зробити. Для цього треба ввести наступні данні (див. рис. 5.1).

В поле Password обов'язково потрібно ввести наступній пароль: 1357905ZEVSAC121

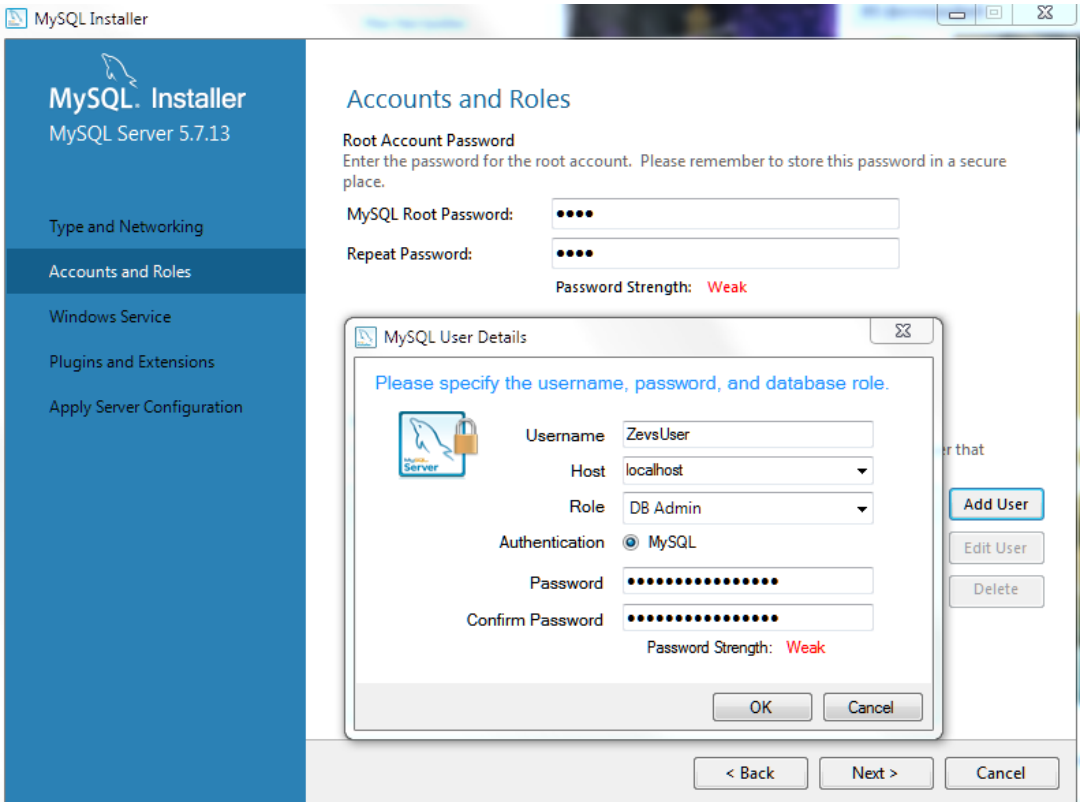

Рисунок 5.1 – Створення нового користувача

Крок 3. Запустити програму MySQL Workbench та увійти в режимі адміністратора, та змінити права доступу у користувача, для цього потрібно виконати наступні дії:

- вибрати вкладку Users and Privileges, та натиснути на користувача ZevsUser
- вибрати вкладку Administrative Roles та зняти виділення з усіх полів крім Custom
- в вкладці Custom зняти виділення з усіх полів крім Select, Insert та натиснути Apply

Результат змінення прав доступу зображено на рис. 5.2

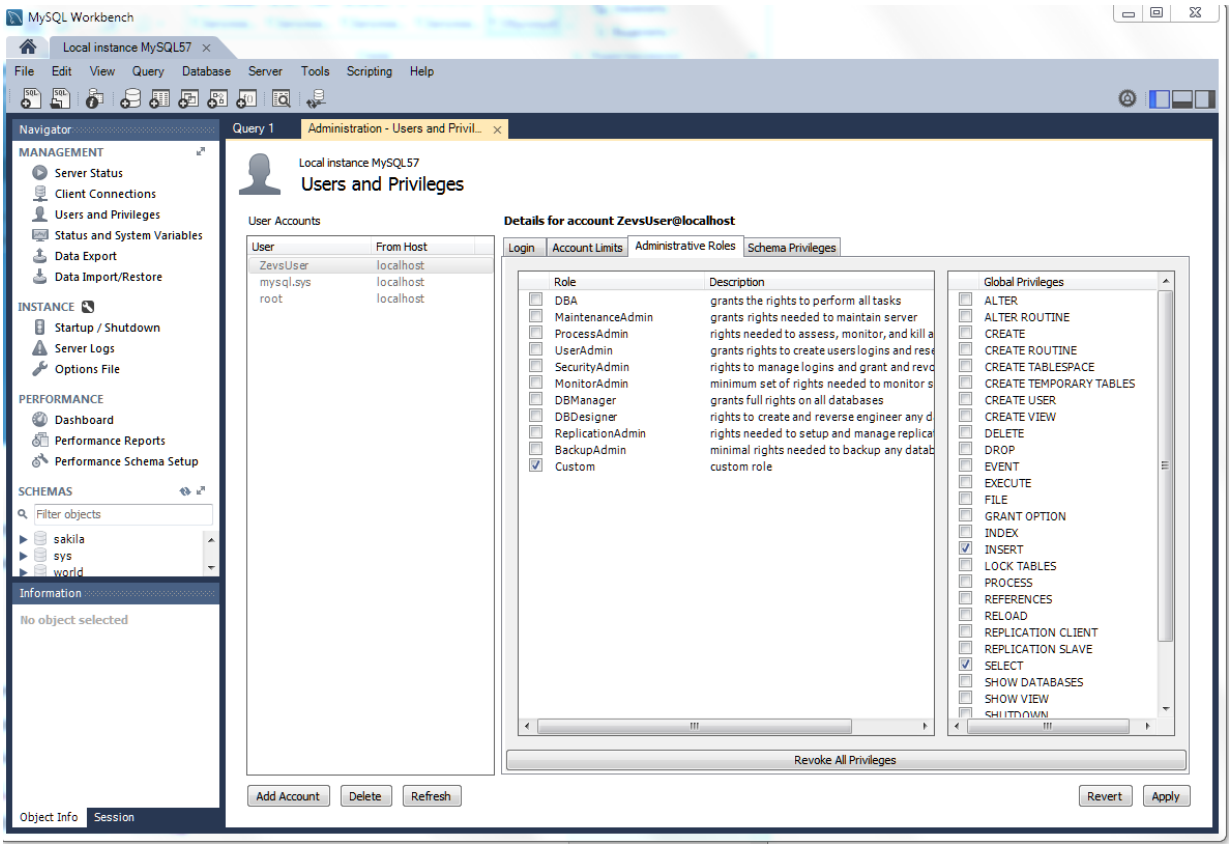

Рисунок 5.2 – Змінені права доступу користувача

Крок 4. Скачати та встановити дамп бази даних, він зберігає у собі всі необхідні дані для роботи програми. Скачати можна з репрезенторію проекту по наступному лінку:

https://github.com/ScieteX/ZEVS/blob/master/ZEVS.sql?raw=true

Встановити дамп можна з допомогою MySQL Workbench -> Data Import, або команди $ mysql -u username -p -h localhost DATA-BASE-NAME < data.sql

На рис. 5.3 зображено процес встановлення дампу бази даних.

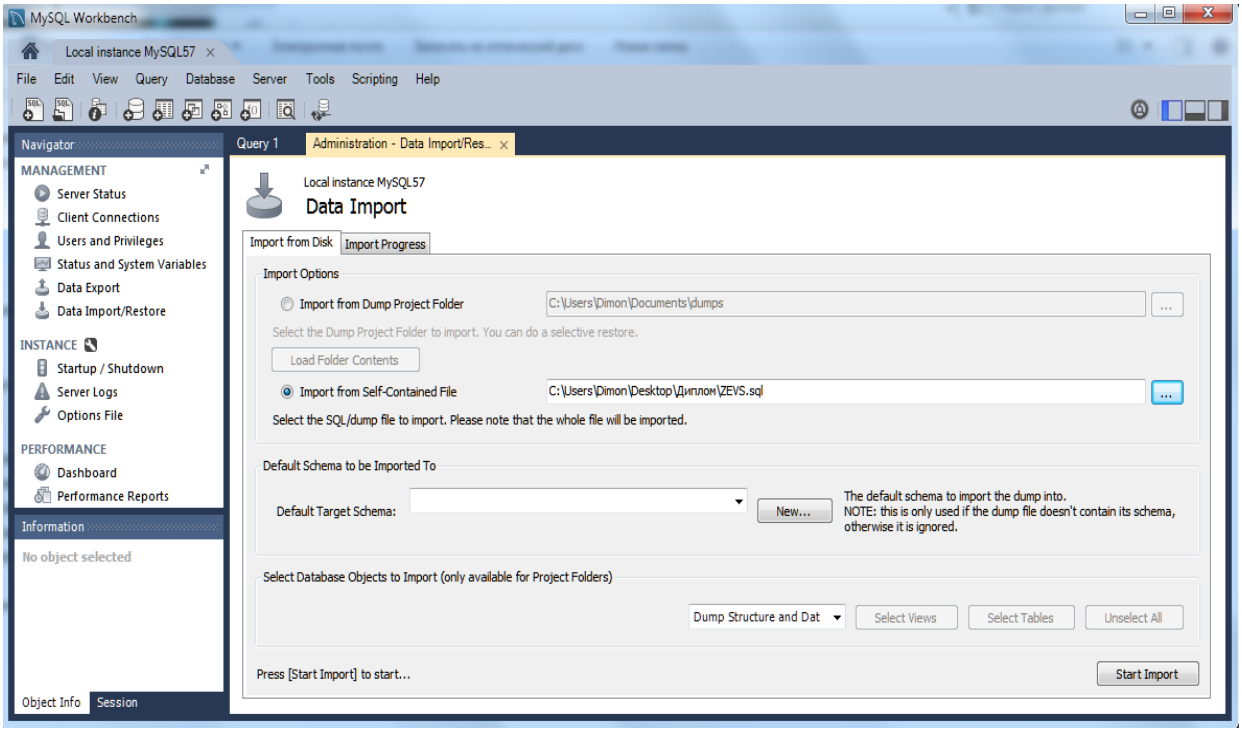

Рисунок 5.3 – Встановлення дампу бази данних

Крок 5. Скачати та запустити саму програму ZEVS. Скачати можна з репрезенторію проекту по наступному лінку:

https://github.com/ScieteX/ZEVS/blob/master/ZEVS.jar?raw=true

5.3 Інструкція користувача

Для роботи с програмою будемо використовувати наступні Логіни/паролі для входу:

- Режим користувача – User/123
- Режим гостя – кнопка гість у вікні авторизації
- Режим адміністратора – root/123 та root/root для адміністрування.

Розглянемо самий простий варіант роботи, а саме режим гість. Для цього в вікні авторизації натиснемо кнопку «Гість» на рис. 5.4 зображено форму авторизації.

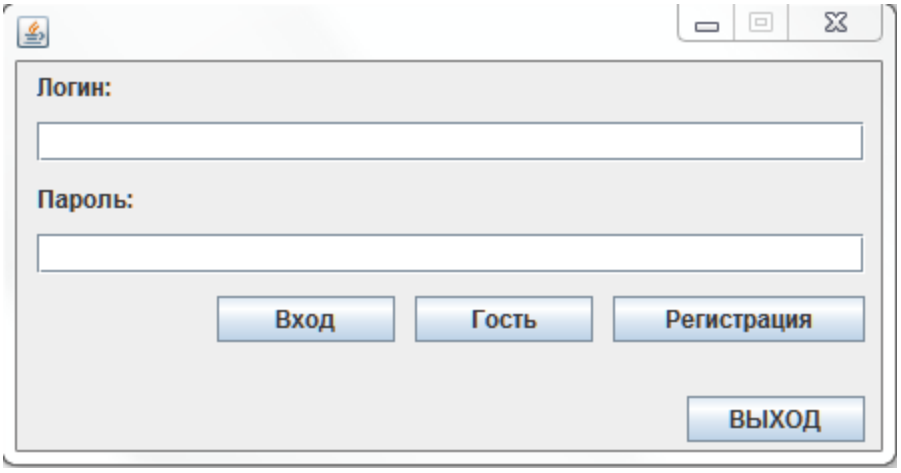

Рисунок 5.4 – Авторизація

Після натискання кнопки «Гість» відображається головне вікно програми (див. рис. 5.5)

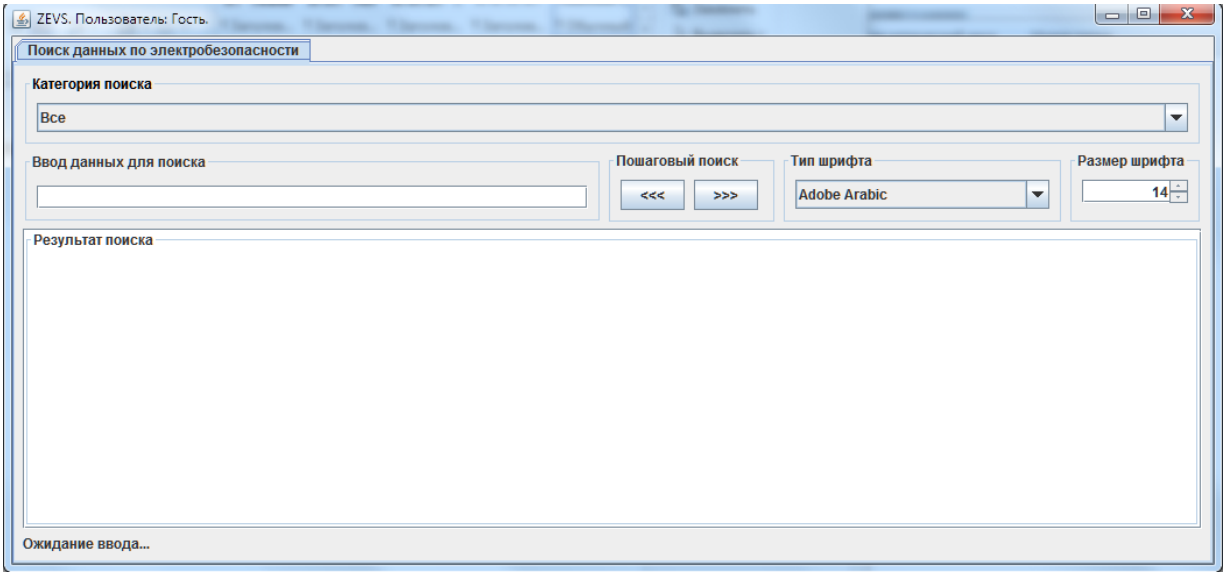

Рисунок 5.5 – Головне вікно програми

В режимі «Гість» доступні функціонал складається лише с пошуку інформації з електробезпеки. Консультація та адміністрування не доступні в режимі «Гість».

Для початку пошуку необхідно вибрати категорію та ввести ключове слово. Якщо необхідно виконати покроковий пошук треба натиснути відповідні кнопки (див. рис. 5.6)

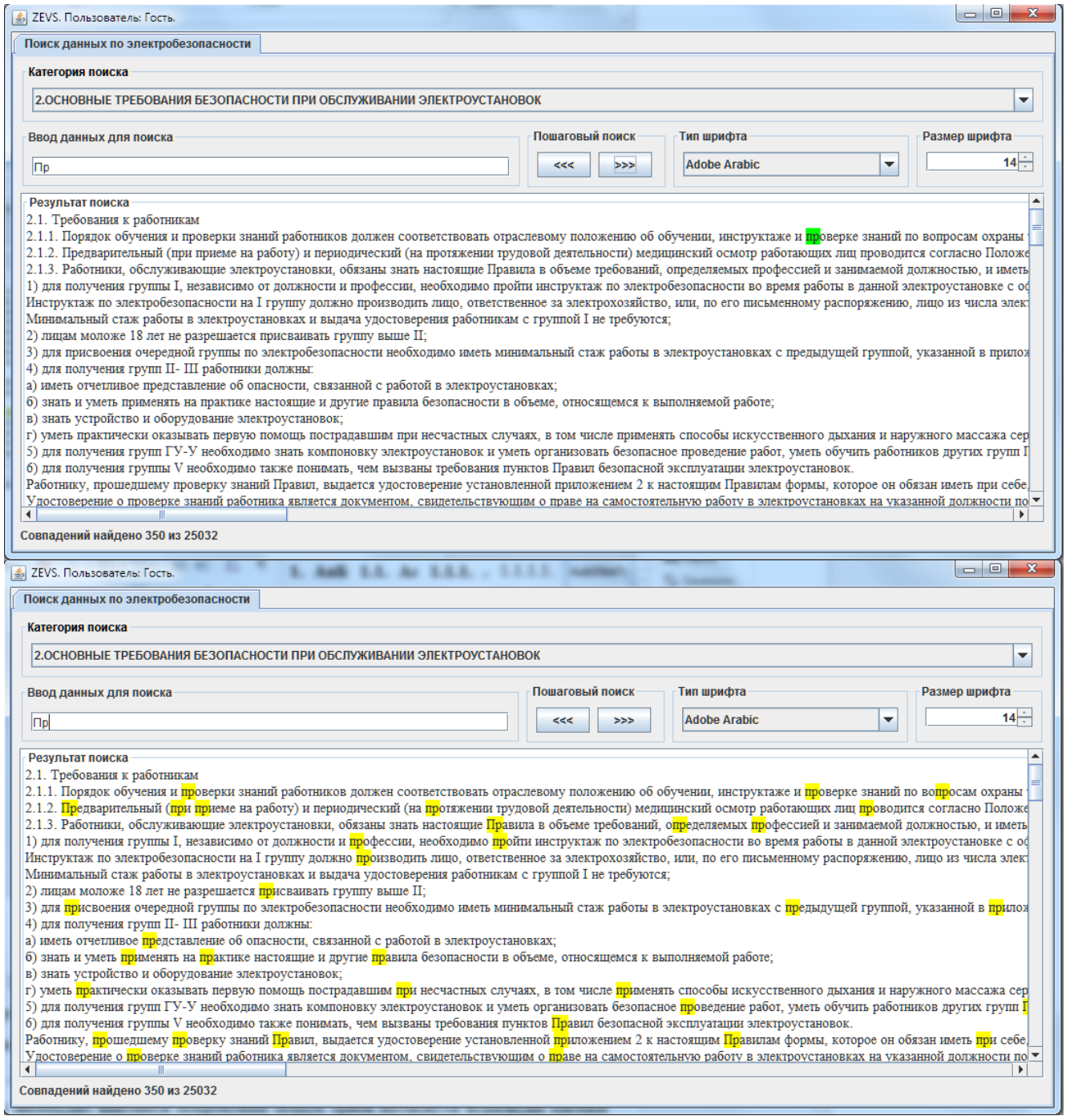

Рисунок 5.6 – Процес пошуку інформації

Як видно на рисунку вище, в процесі пошуку було знайдено 350 збігів.

Далі увійдемо в режим «Користувач» та випробуємо функцію консультації. В режимі «Користувач» функціонал стандартний і складається з пошуку інформації та консультування. На рис. 5.7 зображено процес консультування.

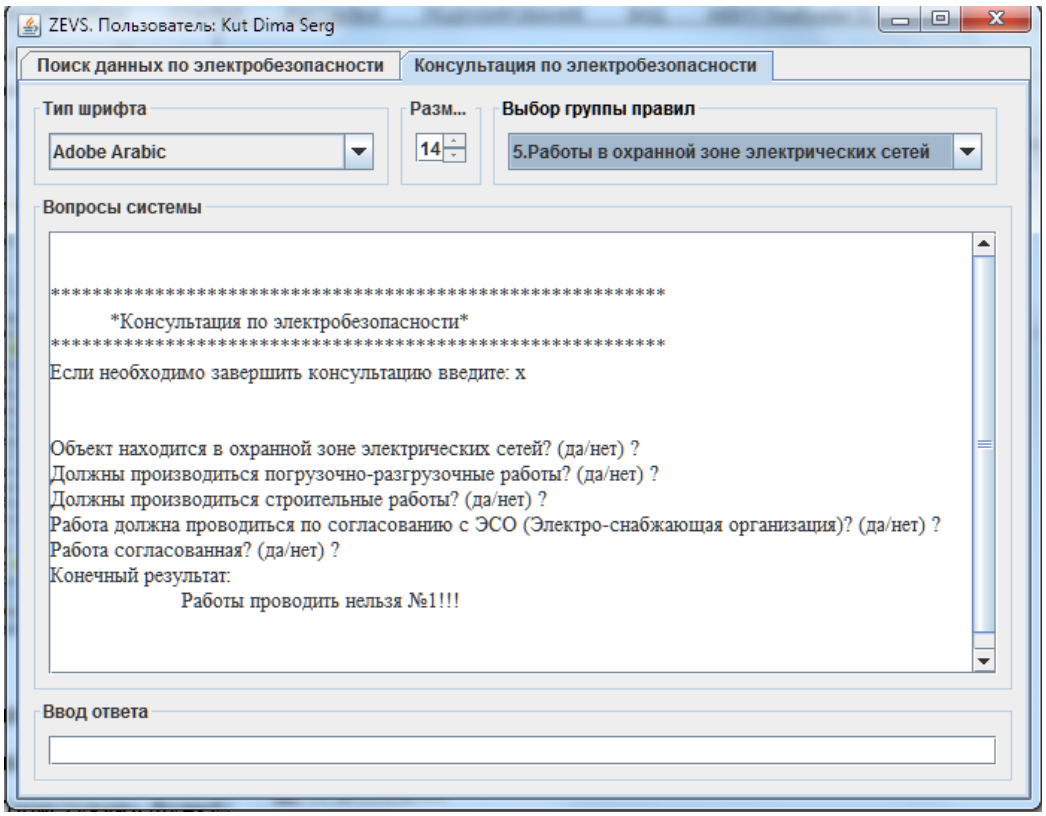

Рисунок 5.7 – Процес консультування

Після вибору групи правил, система почне задавати користувачу питання на які необхідно відповідати точно з рекомендацій. Після закінчення опитування користувачу буде виведений кінцевій результат о дозволі проведення робіт.

Тепер увійдемо в режим «Адміністратор» в цьому режимі функціонал такий же як і у режиму «Гість» плюс можливість редагування даних.

Редагування складається з додавання, видалення, та оновлення бази даних системи. Адміністратор може редагувати користувачі, базу правил експертної системи та довідкову інформацію з електробезпеки.

На рис. 5.8 зображено процес входу в режим «Адміністратора»

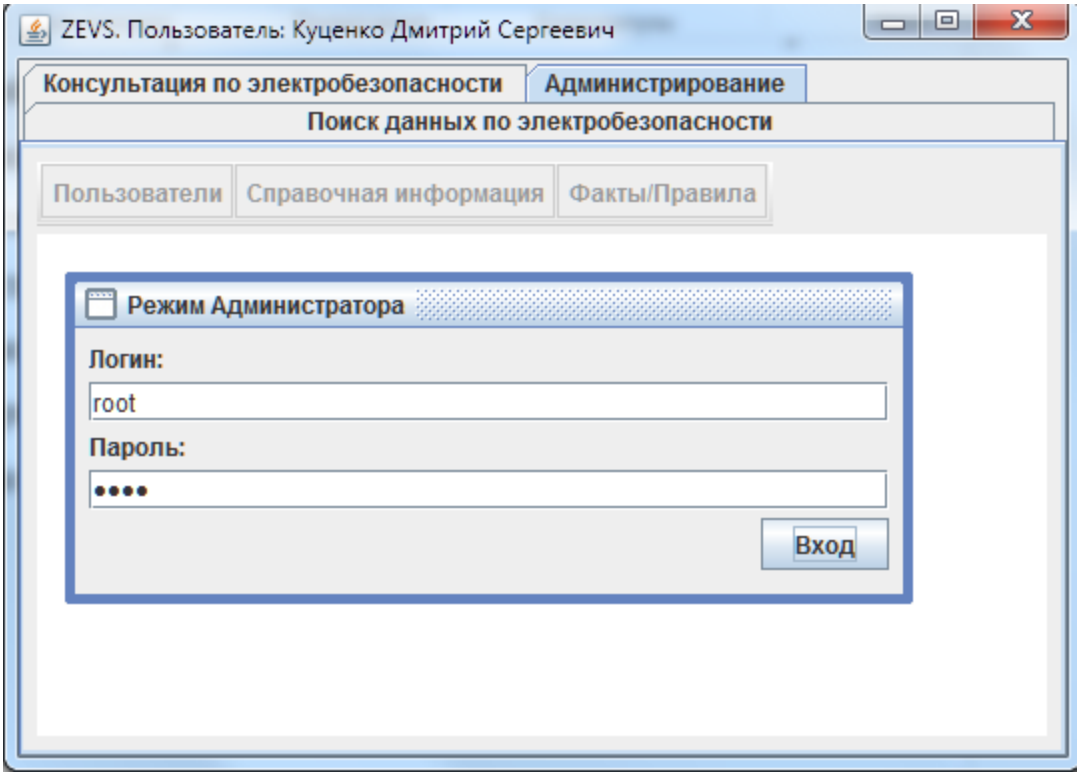

Рисунок 5.8 – Процес авторизації адміністратора

Відразу після авторизації з’являться три вікна для редагування (див. рис. 5.9). Для прикладу роботи виберемо редагування користувачі і змінимо деякі данні. На рисунка 5.10 – 5.12 зображено процес змінення даних.

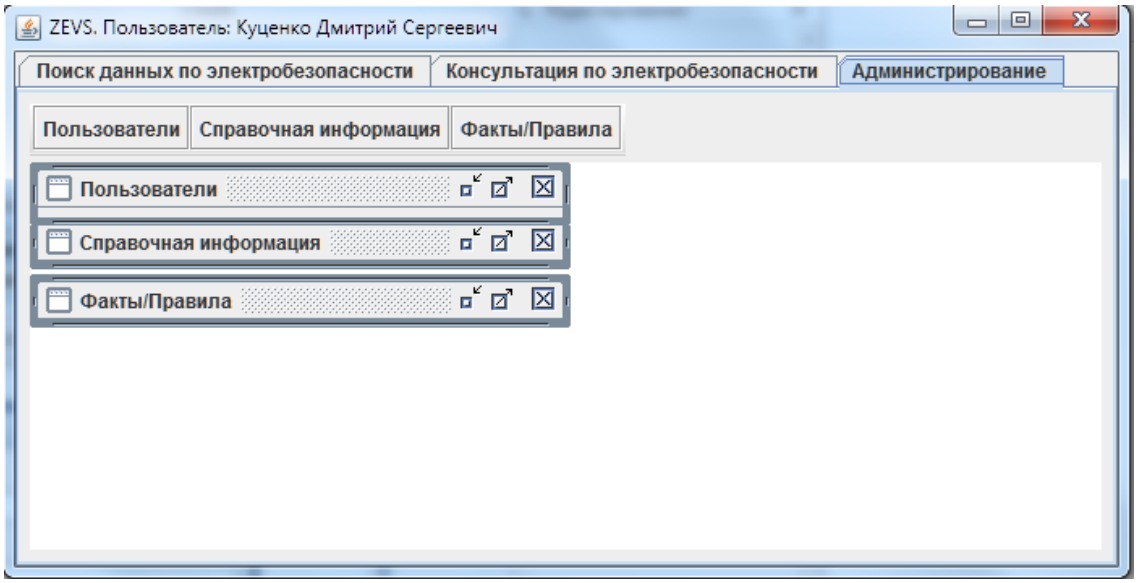

Рисунок 5.9 – Вікна редагування даних

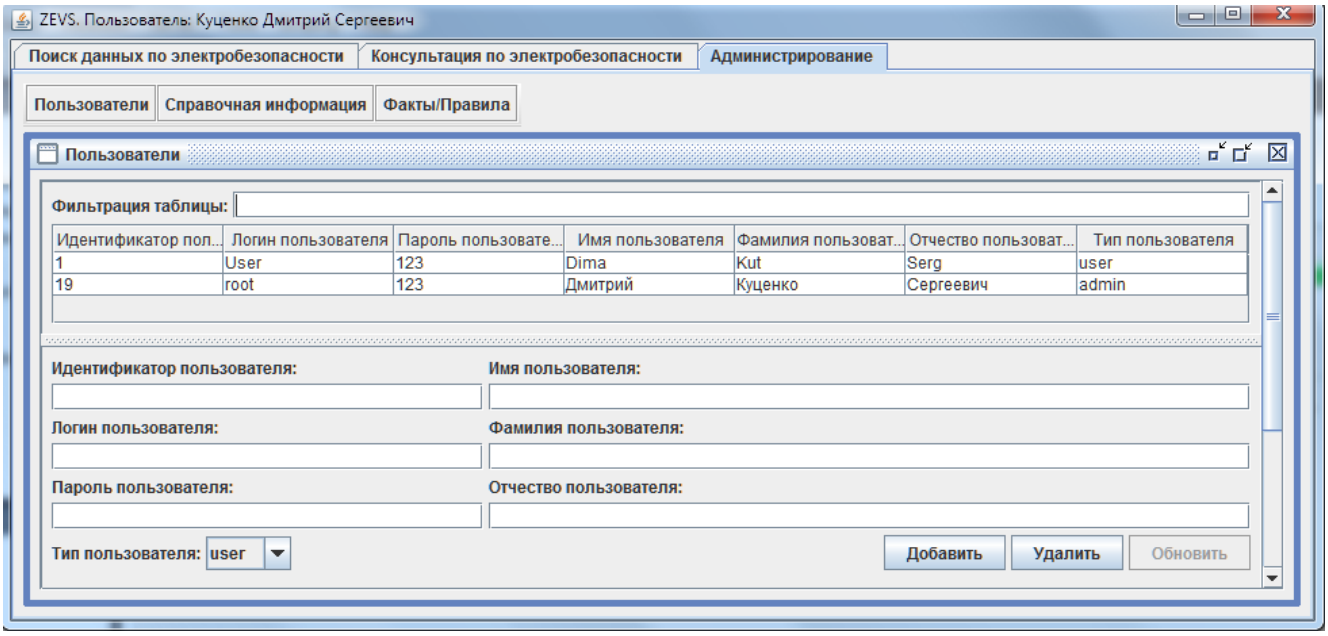

Рисунок 5.10 – Змінення даних користувача

Як видно на рис. 5.10 у користувача с логіном User неправильно написані ім’я, прізвище, та ім’я по батькові. Змінимо ці данні на інші, для цього натискаємо на користувача та вводимо необхідні змінення.

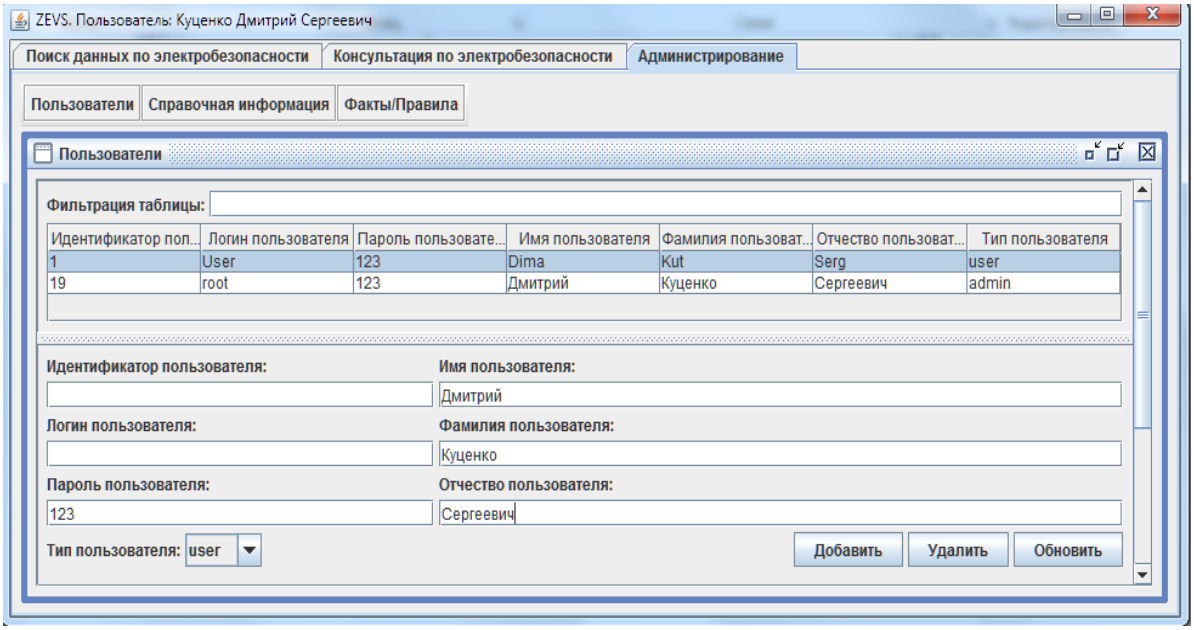

Рисунок 5.11 – Введення нових даних.

Після вводу необхідних даних необхідно натиснути на кнопку «Оновити». На рис. 5.12 зображено результат оновлення даних.

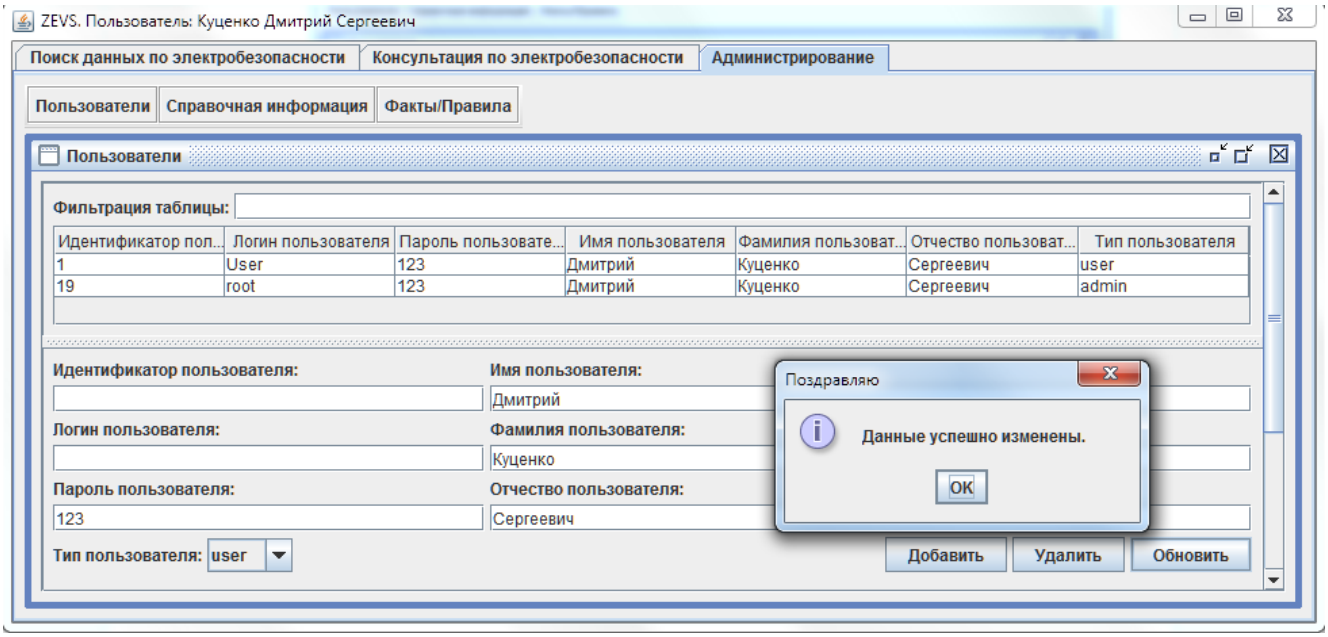

Рисунок 5.12 – Результат оновлення інформації

Тепер ми бачимо що дані успішно оновлені і відповідають правилам написання.

5.5 Проведення експерименту

Для перевірки ефективності розробленого продукту проводився процес заміри часу необхідного для отримання інформації з електробезпеки. Без використання програми, пошук необхідної інформації з електробезпеки займав більше 2 годин, в цей час увійшов пошук необхідної літератури, аналіз а також збір коректних правил електробезпеки. В результаті вимірювань було виявлено що розроблена програма скорочує процес пошуку в рази, тепер час пошуку займає менше 20 хвилин що показує зменшення часу пошуку на 84%, що є більш ніж хорошим результатом.

ВИСНОВКИ

В результаті виконання даної роботи було розроблено програмний продукт інформаційної системи з електробезпеки с можливістю консультування.

Причиною для розробки стали наступні проблеми:

- зі збільшенням електрообладнання, збільшується складність контролю;
- високий ступінь складності контролю електробезпеки;
- високі вимоги до кваліфікації відповідальної особи.

В процесі виконання роботи було виконано наступні задачі а саме:

- виявлення специфікації вимог програмного продукту;
- проведення аналізу раніше знайдених специфікацій вимог;
- створення плану розробки з урахуванням специфікацій вимог;
- розробка архітектури програмної системи;
- вибір технологій та інструментів розробки системи.

В процесі виконання було створено клієнт-серверну програму, в результаті чого дана система може одночасно працювати з вилкою кількість користувачі.

За серверну частину відповідає MySQL Community Server, яка зберігає у собі списки користувачів, довідкову інформацію та базу знань експертної системи.

Клієнтська частина програми велась мовою Java з використанням бібліотеки JESS для роботи з експертною системою та бібліотекою Java Swing у середовищі Eclipse KEPLER.

Розробка експертної системи була проведена за наступною схемою:

- Вивчення предметної області електробезпеки;
- Пошук джерел знань (експертів);
- Вибір методів добування знань;
- Оцінка якості здобутих знань;
- Формування на основі здобутих знань бази знань ЕС.

В результаті експерименту було виявлено, що система значно зменшує час пошуку інформації з електробезпеки. Без використання системи ZEVS пошук необхідної інформації з електробезпеки займав більше 2 годин, в цей час увійшов пошук данних, аналіз а також збір коректних правил з електробезпеки. З використанням розроблюваної системи процес пошуку інформації займав менше 20 хвилин.

Тепер не потрібно перечитувати велику кількість документів, які стосуються електробезпеки, достатньо пройти консультацію або виконати пошук за ключовим словом.

Завдяки наявності власного графічного інтерфейсу користувача можливе застосування розробленого продукту у вигляді самостійного інструмента розробки інформаційної системі з можливостю консультування. Крім цього, незважаючи на орієнтацію на предметну область, завдяки універсальності форми подання знань та методів контролю можливе застосування продукту для розробки баз знань у будь-якій предметній області.

СПИСОК ВИКОРИСТАННИХ ДЖЕРЕЛ

1. Васильев А.Н.: Java. – СПб.: Питер, 2011. – 396 с.

2. Хабибуллин И.: Разработка Web-служб средствами Java. – СПб.: БХВ Петербург, 2003. – 400 с.

3. Хабибуллин И.: Создание распределенных приложений на Java 2. – СПб.: БХВ-Петербург, 2002. – 692 с.

4. Кубенский А.: Создание и обработка структур данных в примерах на Java. - СПб.: БХВ-Петербург, 2001. 336 с.

5. Попов Э.В. Статические и динамические экспертные системы / Э.В. Попов, И.Б. Фоминых, Е.В. Кисель, М.Д. Шапот – М.: «Финансы и статистика», 1996. – 318 с.

6. Гаврилова Т.А. Базы знаний интеллектуальных систем / Т.А. Гаврилова, В.Ф.Хорошевский – СПб.: Питер, 2005. – 384 с.

7. Джексон П. Введение в экспертные системы / П. Джексон – М.: Вильямс, 2001. – 624 с.

8. Джарратано Дж. Экспертные системы. Принципы разработки и программирование, 4е изд. / Джарратано Дж., Райли Г. – М.: Вильямс, 2007. – 1152 с.

9. Субботін С. О. Подання й обробка знань у системах штучного інтелекту та підтримки прийняття рішень: Навчальний посібник / С. О. Субботін – Запоріжжя: ЗНТУ, 2008. – 341 с.

Додаток А. ОПИС МОДУЛЯ ЄКСПЕРТНОЇ СИСТЕМИ ZEVS

Група правил «Роботи за нарядом і розпорядженням, інструктаж»

Якщо U > 1000в, робота повинна виконуватися в порядку технічної експлуатації і розпорядження на роботу не оформлено і наряд на роботу не оформлений, то роботу виконувати не можна.

Якщо U > 1000в і робота повинна виконуватися за нарядом, і немає спостерігача, то роботу виконувати не можна.

Якщо U > 1000в і робота повинна виконуватися за нарядом, і немає керівника, то роботу виконувати не можна.

Якщо робота повинна виконуватися за нарядом, і наряд на роботу не оформлений, то роботу виконувати не можна.

Якщо робота повинна виконуватися за розпорядженням, і немає спостерігача, то роботу виконувати не можна.

Якщо робота повинна виконуватися за розпорядженням, і розпорядження на роботу не оформлено, то роботу виконувати не можна.

Якщо не для кожного працівника є відмітка про цільовий інструктажі, то роботу виконувати не можна.

Якщо не для кожного працівника дата наступної перевірки з охорони праці <WorkEnd_time, то роботу виконувати не можна.

Група правил «Правила, що залежать від зовнішніх умов»

Якщо приміщення особливо небезпечне і якщо тип робіт - перемикання, і кількість працівників >= 2, і немає таких двох людей, щоб одна людина була з кваліфікацією > = 2, а інший з кваліфікацією >= 3, то роботу виконувати не можна.

Якщо приміщення з підвищеною небезпекою і якщо тип робіт - перемикання, і кваліфікація кожного працівника бригади < 3, то роботу виконувати не можна.

Якщо приміщення особливо небезпечне, і якщо тип робіт - перемикання, кількість працівників < 2, то роботу виконувати не можна.

Якщо приміщення з підвищеною небезпекою або приміщення особливо небезпечне і вид робіт - висотні, і кількість працівників < 2, то роботу виконувати не можна.

Якщо приміщення без підвищеної небезпеки і тип робіт - зняття показань приладів обліку, і U > 1000в і кваліфікація кожного працівника бригади <3, то роботу виконувати не можна.

Група правил «Робота з приладами обліку»

Якщо напруга від 42в до 1000в і робота - заміна приладу обліку, і немає індикатора напруги чи ні діелектричних рукавичок, чи ні інструменту з ізолюючими рукоятками, то роботу виконувати не можна.

Якщо напругу понад 1000 В і робота - заміна приладу обліку, і немає покажчика напруги чи ні ізолюючої штанги, то роботу виконувати не можна.

Якщо тип роботи - зняття приладу обліку трансформаторного включення і немає наряду на виконання роботи, то роботу виконувати не можна.

Якщо тип роботи - зняття приладу обліку трансформаторного включення і число працівників в бригаді < 2 або не оформлений наряд, або не оформлене розпорядження, то роботу виконувати не можна.

Якщо тип роботи - зняття приладу обліку трансформаторного включення і кваліфікація кожного працівника бригади < 4 або не оформлений наряд, або не оформлене розпорядження, то роботу виконувати не можна.

Якщо тип роботи - зняття приладу обліку безпосереднього включення і кваліфікація працівника < 3 не оформлене розпорядження, то роботу виконувати не можна.

Якщо тип роботи - зняття показань приладу обліку або приладу електровимірювання і число працівників = 1, і кваліфікація працівника < 3, то роботу виконувати не можна.

Група «Захисні засоби»

Якщо напругу понад 1000 В і немає одного з додаткових захисних засобів, необхідних для даної роботи, то роботу виконувати не можна.

Якщо не дата наступного випробування будь-якого ЕЗС (Електрозащітние кошти) з одного з двох списків, представлених нижче > CurrentDate, то роботи виконувати не можна.

Якщо неможливо встановити огорожу струмоведучих частин і замість огорожі використовуються ізолюючі накладки і число працівників < 2 або група кваліфікації кожного працівника < 3 , то роботу виконувати не можна.

Група «Роботи в охоронній зоні електричних мереж»

Якщо об'єкт знаходиться в охоронній зоні електричних мереж (ЕС), і робота повинна проводитися за погодженням з ЕСО (Електро-постачальна організація) і немає узгодження, то роботу виконувати не можна.

Якщо повинні проводитися будівельні роботи, і місце проведення робіт знаходиться в охоронній зоні ЕС, то без узгодження виконання цих робіт з організацією, яка експлуатує ЕС, і отримання дозволу на їх виконання роботи виконувати не можна.

Якщо повинні проводитися вантажно-розвантажувальні роботи і місце проведення робіт знаходиться в охоронній зоні ЕС, то без узгодження виконання цих робіт з організацією, яка експлуатує ЕС, і отримання дозволу на їх виконання роботи виконувати не можна.

Додаток В. ПРОГРАМНИЙ КОД ЕКСПЕРТНОЇ СИСТЕМИ

Група правил «Роботи за нарядом і розпорядженням, інструктаж»

```
(deftemplate av  (slot a) (slot v))
(deffunction ask-question (?question $?allowed-values)
   (printout t ?question crlf)
   (bind ?answer (read))
   (if (lexemep ?answer)
      then
         (bind ?answer (lowcase ?answer)))
    (if (= ?answer \x)
        then
        (printout t "ОТМЕНЕНО ПОЛЬЗОВАТЕЛЕМ!!!" crlf)
            (halt)
            (reset)
            (clear)
        else
   (while (not (member$ ?answer ?allowed-values)) do
       (if (= ?answer \x)
        then
       (halt)
       (reset)
       (clear)
       (printout t "ОТМЕНЕНО ПОЛЬЗОВАТЕЛЕМ!!!" crlf)
       (break))
       (printout t ?question crlf)
       (bind ?answer (read))
       (if (lexemep ?answer)
          then
           (bind ?answer (lowcase ?answer)))))
     ?answer
)

(deffunction yes-or-no-p (?question)
(bind ?response (ask-question ?question да нет))
(if (or (eq ?response да) (eq ?response y))
then
TRUE
```

```
else
FALSE)
)
(defrule answer-1
  (declare (salience 4))
 (not (av (a voltage   ) (v ?)))
  (not (result ?))
=>
(if (yes-or-no-p  "Напряжение больше 1000В? (да/нет) ? ")
then
(assert (av (a voltage) (v voltage_high)))
else
(assert (av (a voltage) (v voltage_low)))
))

(defrule answer-2
  (declare (salience 6))
  (not (av (a work   ) (v ?)))
  (not (result ?))
=>
(assert (av (a work    ) (v (ask-question  "Работа должна выполняться по
(наряду или распоряжению или тэ)? "  наряду  распоряжению тэ))))
)



(defrule answer-3
  (declare (salience -1))
 (not (av (a work_inst  ) (v ?)))
  (not (result ?))
=>
(if (yes-or-no-p  "Распоряжение на работу  оформлено? (да/нет) ? ")
then
(assert (av (a work_inst ) (v work_inst_yes)))
else
(assert (av (a work_inst ) (v work_inst_no)))
))
```

```
(defrule answer-4
  (declare (salience 5))
 (not (av (a work_livery ) (v ?)))
  (not (result ?))
=>
(if (yes-or-no-p  "Наряд на работу оформлен? (да/нет) ? ")
then
(assert (av (a work_livery) (v work_livery_yes)))
else
(assert (av (a work_livery) (v work_livery_no)))
))

(defrule answer-5
  (declare (salience 7))
 (not (av (a watch_man   ) (v ?)))
  (not (result ?))
=>
(if (yes-or-no-p  "Есть наблюдающий человек? (да/нет) ? ")
then
(assert (av (a watch_man) (v watch_man_yes)))
else
(assert (av (a watch_man) (v watch_man_no)))
))

(defrule answer-6
  (declare (salience -1))
 (not (av (a director  ) (v ?)))
  (not (result ?))
=>
(if (yes-or-no-p  "Есть руководитель? (да/нет) ? ")
then
(assert (av (a director) (v director_yes)))
else
(assert (av (a director) (v director_no)))
))
```

```
(defrule answer-7
  (declare (salience 8))
 (not (av (a target_briefing  ) (v ?)))
  (not (result ?))
=>
(if (yes-or-no-p  "У каждого работника есть отметка о целевом инструктаже?
(да/нет) ? ")
then
(assert (av (a target_briefing) (v target_briefing_yes)))
else
(assert (av (a target_briefing) (v target_briefing_no)))
))

(defrule answer-8
  (declare (salience 9))
 (not (av (a work_end_time  ) (v ?)))
  (not (result ?))
=>
(if (yes-or-no-p  "Не для каждого работника просрочена дата проверки знаний
по охране труда? (да/нет) ? ")
then
(assert (av (a work_end_time) (v work_end_time_yes)))
else
(assert (av (a work_end_time) (v work_end_time_no)))
))

(defrule result-1
       (and(av (a voltage) (v voltage_high))
           (av (a work   ) (v тэ))
           (av (a work_inst ) (v work_inst_no))
           (av (a work_livery) (v work_livery_no)))
        (not (result ?))
         =>
(assert (result "Работы проводить нельзя!!!"))
)

(defrule result-2
```

```
       (and(av (a voltage) (v voltage_high))
           (av (a work   ) (v наряду))
           (av (a watch_man) (v watch_man_no)))
        (not (result ?))
         =>
(assert (result "Работы проводить нельзя!!!"))
)
(defrule result-3
       (and(av (a voltage) (v voltage_high))
           (av (a work   ) (v наряду))
           (av (a director) (v director_no)))
        (not (result ?))
         =>
(assert (result "Работы проводить нельзя!!!"))
)

(defrule result-4
       (and(av (a work   ) (v наряду))
           (av (a work_livery) (v work_livery_no)))
        (not (result ?))
         =>
(assert (result "Работы проводить нельзя!!!"))
)

(defrule result-5
       (and(av (a work   ) (v распоряжению))
           (av (a watch_man) (v watch_man_no)))
        (not (result ?))
         =>
(assert (result "Работы проводить нельзя!!!"))
)

(defrule result-6
       (and(av (a work   ) (v распоряжению))
           (av (a work_inst ) (v work_inst_no)))
        (not (result ?))
         =>
```

```
(assert (result "Работы проводить нельзя!!!"))
)

(defrule result-7
       (exists(av (a target_briefing) (v target_briefing_no)))
        (not (result ?))
         =>
    (assert (result "Работы проводить нельзя!!!"))
)

(defrule result-8
       (exists(av (a work_end_time) (v work_end_time_yes)))
        (not (result ?))
         =>
    (assert (result "Работы проводить нельзя!!!"))
)
(defrule ok-result ""
    (declare (salience -10))
   (not (result ?))
   =>
   (assert (result "Всё в порядке. Необходимые работы можно выполнять."))
)


(defrule print-result ""
   (declare (salience 10))
   (result ?item)
    =>
   (printout            t "Конечный результат:")
   (printout            t crlf)
   (format t            " %s%n%n%n" ?item)
)
(defrule system-banner ""
(declare (salience 10) )
 =>
(printout t crlf crlf)
(printout  t  "*************************************************************"
```

```
crlf)
(printout t "          *Консультация по электробезопасности*" crlf)
(printout  t  "*************************************************************"
crlf)
(printout t "Если необходимо завершить консультацию введите: \\x" crlf)
(printout t crlf crlf)
)
```

Група правил «Правила, що залежать від зовнішніх умов»

```
(deftemplate av  (slot a) (slot v))
(deffunction ask-question (?question $?allowed-values)
   (printout t ?question crlf)
   (bind ?answer (read))
   (if (lexemep ?answer)
      then
         (bind ?answer (lowcase ?answer)))
    (if (= ?answer \x)
        then
        (printout t "ОТМЕНЕНО ПОЛЬЗОВАТЕЛЕМ!!!" crlf)
            (halt)
            (reset)
            (clear)
        else
   (while (not (member$ ?answer ?allowed-values)) do
       (if (= ?answer \x)
        then
       (halt)
       (reset)
       (clear)
       (printout t "ОТМЕНЕНО ПОЛЬЗОВАТЕЛЕМ!!!" crlf)
       (break))
      (printout t ?question crlf)
      (bind ?answer (read))
      (if (lexemep ?answer)
         then
          (bind ?answer (lowcase ?answer)))))
     ?answer
)
```

```
(deffunction yes-or-no-p (?question)
(bind ?response (ask-question ?question да нет))
(if (or (eq ?response да) (eq ?response y))
then
TRUE
else
FALSE)
)
(defrule answer-1
  (declare (salience 4))
 (not (av (a voltage   ) (v ?)))
  (not (result ?))
=>
(if (yes-or-no-p  "Напряжение больше 1000В? (да/нет) ? ")
then
(assert (av (a voltage) (v voltage_high)))
else
(assert (av (a voltage) (v voltage_low)))
))

(defrule answer-2
  (declare (salience 9))
  (not (av (a room  ) (v ?)))
  (not (result ?))
=>
(assert (av (a room   ) (v (ask-question  "Помещение (особо опасное(оо) или
с повышенной опасностью(спо) или без повышенной опасности(бпо))? "  оо спо
бпо))))
)

(defrule answer-3
  (declare (salience 8))
 (not (av (a work_type  ) (v ?)))
  (not (result ?))
=>
(assert (av (a work_type   ) (v (ask-question  "Тип работ (переключение или
```

```
снятие показаний приборов учета (сппу))? "  переключение сппу))))
)

(defrule answer-4
  (declare (salience 5))
 (not (av (a work_mans ) (v ?)))
  (not (result ?))
=>
(if (yes-or-no-p  "Количество работников меньше 2? (да/нет) ? ")
then
(assert (av (a work_mans) (v work_mans_yes)))
else
(assert (av (a work_mans) (v work_mans_no)))
))

(defrule answer-5
  (declare (salience 6))
 (not (av (a work_man_skill   ) (v ?)))
  (not (result ?))
=>
(if  (yes-or-no-p    "Есть двое человек, чтобы один человек был с квалификацией больше или равной 2, а другой с квалификацией больше или равной 3? (да/нет) ? ")
then
(assert (av (a work_man_skill) (v work_man_skill_yes)))
else
(assert (av (a work_man_skill) (v work_man_skill_no)))
))

(defrule answer-6
  (declare (salience 7))
 (not (av (a all_work_man_skill   ) (v ?)))
  (not (result ?))
=>
(if  (yes-or-no-p    "Квалификация каждого работника бригады меньше 3? (да/нет) ? ")
then
```

```
(assert (av (a all_work_man_skill) (v all_work_man_skill_yes)))
else
(assert (av (a all_work_man_skill) (v all_work_man_skill_no)))
))

(defrule answer-7
  (declare (salience 4))
 (not (av (a view_w  ) (v ?)))
  (not (result ?))
=>
(if (yes-or-no-p  "Вид работ – высотные? (да/нет) ? ")
then
(assert (av (a view_w) (v view_w_yes)))
else
(assert (av (a view_w) (v view_w_no)))
))

(defrule result-1
       (and(av (a room  ) (v oo))
           (av (a work_type  ) (v переключение))
           (av (a work_mans) (v work_mans_yes)))
        (not (result ?))
         =>
(assert (result "Работы проводить нельзя!!!"))
)

(defrule result-2
       (and(av (a room  ) (v oo))
           (av (a work_type  ) (v переключение))
           (av (a work_mans) (v work_mans_no))
           (av (a work_man_skill) (v work_man_skill_no))
        )
        (not (result ?))
         =>
(assert (result "Работы проводить нельзя!!!"))
)
(defrule result-3
```

```
        (and(av (a room  ) (v спо))
            (av (a work_type  ) (v переключение))
            (av (a all_work_man_skill) (v all_work_man_skill_yes))
         )
         (not (result ?))
          =>
(assert (result "Работы проводить нельзя!!!"))
)

(defrule result-4
        (and(av (a room  ) (v бпо))
            (av (a work_type  ) (v сппу))
            (av (a voltage) (v voltage_high))
            (av (a all_work_man_skill) (v all_work_man_skill_yes))
         )
         (not (result ?))
          =>
(assert (result "Работы проводить нельзя!!!"))
)
(defrule result-5
        (and(av (a view_w) (v view_w_yes))
            (av (a work_mans) (v work_mans_yes))
          (or (av (a room  ) (v спо))
              (av (a room  ) (v оо)))
         )
         (not (result ?))
          =>
(assert (result "Работы проводить нельзя!!!"))
)
(defrule ok-result ""
    (declare (salience -10))
   (not (result ?))
   =>
   (assert (result "Всё в порядке. Необходимые работы можно выполнять."))
)
```

```
(defrule print-result ""
   (declare (salience 10))
   (result ?item)
    =>
   (printout            t "Конечный результат:")
   (printout            t crlf)
   (format t            " %s%n%n%n" ?item)
)
(defrule system-banner ""
(declare (salience 10) )
 =>
(printout t crlf crlf)
(printout  t  "*************************************************************"
crlf)
(printout t "          *Консультация по электробезопасности*" crlf)
(printout  t  "*************************************************************"
crlf)
(printout t "Если необходимо завершить консультацию введите: \\x" crlf)
(printout t crlf crlf)
)
```

Група правил «Робота з приладами обліку»

```
(deftemplate av  (slot a) (slot v))
(deffunction ask-question (?question $?allowed-values)
   (printout t ?question crlf)
   (bind ?answer (read))
   (if (lexemep ?answer)
      then
         (bind ?answer (lowcase ?answer)))
    (if (= ?answer \x)
        then
        (printout t "ОТМЕНЕНО ПОЛЬЗОВАТЕЛЕМ!!!" crlf)
            (halt)
            (reset)
            (clear)
        else
   (while (not (member$ ?answer ?allowed-values)) do
       (if (= ?answer \x)
```

```
        then
       (halt)
       (reset)
       (clear)
       (printout t "ОТМЕНЕНО ПОЛЬЗОВАТЕЛЕМ!!!" crlf)
       (break))
      (printout t ?question crlf)
      (bind ?answer (read))
      (if (lexemep ?answer)
         then
          (bind ?answer (lowcase ?answer)))))
     ?answer
)

(deffunction yes-or-no-p (?question)
(bind ?response (ask-question ?question да нет))
(if (or (eq ?response да) (eq ?response y))
then
TRUE
else
FALSE)
)
(defrule answer-1
  (declare (salience 5))
 (not (av (a voltage   ) (v ?)))
  (not (result ?))
=>
(if (yes-or-no-p  "Напряжение от 42в до 1000в? (да/нет) ? ")
then
(assert (av (a voltage) (v voltage_low)))
else
(assert (av (a voltage) (v voltage_high)))
))

(defrule answer-2
  (declare (salience 4))
  (not (av (a work   ) (v ?)))
```

```
  (not (result ?))
=>
(assert (av (a work   ) (v (ask-question  "Тип работы (снятие прибора учета трансформаторного включения(спутв) или снятие прибора учета непосредственного включения(спунв) снятие показаний прибора учета(сппу) или электроизмерительного прибора(эп))? "  спутв  спунв сппу эп))))
)



(defrule answer-3
  (declare (salience 2))
 (not (av (a work_inst  ) (v ?)))
  (not (result ?))
=>
(if (yes-or-no-p  "Распоряжение на работу  оформлено? (да/нет) ? ")
then
(assert (av (a work_inst ) (v work_inst_yes)))
else
(assert (av (a work_inst ) (v work_inst_no)))
))

(defrule answer-4
  (declare (salience 3))
 (not (av (a work_livery ) (v ?)))
  (not (result ?))
=>
(if (yes-or-no-p  "Наряд на работу оформлен? (да/нет) ? ")
then
(assert (av (a work_livery) (v work_livery_yes)))
else
(assert (av (a work_livery) (v work_livery_no)))
))

(defrule answer-5
  (declare (salience 6))
 (not (av (a indicator   ) (v ?)))
```

```
  (not (result ?))
=>
(if (yes-or-no-p  "Есть индикатор напряжения ? (да/нет) ? ")
then
(assert (av (a indicator) (v indicator_yes)))
else
(assert (av (a indicator) (v indicator_no)))
))

(defrule answer-6
  (declare (salience 7))
 (not (av (a insulating_gloves  ) (v ?)))
  (not (result ?))
=>
(if (yes-or-no-p  "Есть диэлектрические перчатки ? (да/нет) ? ")
then
(assert (av (a insulating_gloves) (v insulating_gloves_yes)))
else
(assert (av (a insulating_gloves) (v insulating_gloves_no)))
))

(defrule answer-7
  (declare (salience 8))
 (not (av (a insulating_handles  ) (v ?)))
  (not (result ?))
=>
(if (yes-or-no-p  "Есть инструмент с изолирующими рукоятками ? (да/нет) ?
")
then
(assert (av (a insulating_handles) (v insulating_handles_yes)))
else
(assert (av (a insulating_handles) (v insulating_handles_no)))
))

(defrule answer-8
  (declare (salience 9))
 (not (av (a work_replacement  ) (v ?)))
```

```
  (not (result ?))
=>
(if (yes-or-no-p  "Работа – замена прибора учета? (да/нет) ? ")
then
(assert (av (a work_replacement) (v work_replacement_yes)))
else
(assert (av (a work_replacement) (v work_replacement_no)))
))
(defrule answer-9
  (declare (salience 8))
 (not (av (a insulating_rod  ) (v ?)))
  (not (result ?))
=>
(if (yes-or-no-p  "Есть изолирующая штанга ? (да/нет) ? ")
then
(assert (av (a insulating_rod) (v insulating_rod_yes)))
else
(assert (av (a insulating_rod) (v insulating_rod_no)))
))
(defrule answer-10
  (declare (salience 8))
 (not (av (a work_mans ) (v ?)))
  (not (result ?))
=>
(if (yes-or-no-p  "Количество работников в брегаде меньше 2? (да/нет) ? ")
then
(assert (av (a work_mans) (v work_mans_yes)))
else
(assert (av (a work_mans) (v work_mans_no)))
))

(defrule answer-11
  (declare (salience 8))
 (not (av (a all_work_man_skill   ) (v ?)))
  (not (result ?))
=>
(if  (yes-or-no-p   "Квалификация  каждого  работника  бригады  меньше  4?
```

```
(да/нет) ? ")
then
(assert (av (a all_work_man_skill) (v all_work_man_skill_yes)))
else
(assert (av (a all_work_man_skill) (v all_work_man_skill_no)))
))

(defrule answer-12
  (declare (salience 8))
 (not (av (a worker_qualification   ) (v ?)))
  (not (result ?))
=>
(if (yes-or-no-p  "Квалификация работника меньше 3? (да/нет) ? ")
then
(assert (av (a worker_qualification ) (v worker_qualification_yes)))
else
(assert (av (a worker_qualification ) (v worker_qualification_no)))
))

(defrule answer-13
  (declare (salience 8))
 (not (av (a number_of_workers   ) (v ?)))
  (not (result ?))
=>
(if (yes-or-no-p  "Число работников равно 1? (да/нет) ? ")
then
(assert (av (a number_of_workers ) (v number_of_workers_yes)))
else
(assert (av (a number_of_workers ) (v number_of_workers_no)))
))

(defrule result-1
       (and(av (a voltage) (v voltage_low))
           (av (a work_replacement) (v work_replacement_yes))
        (or(av (a indicator) (v indicator_no))
           (av (a insulating_gloves) (v insulating_gloves_no))
           (av (a insulating_handles) (v insulating_handles_no))
```

```
        ))
        (not (result ?))
         =>
(assert (result "Работы проводить нельзя!!!"))
)

(defrule result-2
       (and(av (a voltage) (v voltage_high))
           (av (a work_replacement) (v work_replacement_yes))
        (or(av (a indicator) (v indicator_no))
           (av (a insulating_rod) (v insulating_rod_no))
        ))
        (not (result ?))
         =>
(assert (result "Работы проводить нельзя!!!"))
)
(defrule result-3
       (and(av (a work   ) (v спутв))
           (av (a work_livery) (v work_livery_no)))
        (not (result ?))
         =>
(assert (result "Работы проводить нельзя!!!"))
)

(defrule result-4
       (and(av (a work   ) (v спутв))
           (av (a work_mans) (v work_mans_yes))
        (or(av (a work_inst ) (v work_inst_no))
           (av (a work_livery) (v work_livery_no))
        ))
        (not (result ?))
         =>
(assert (result "Работы проводить нельзя!!!"))
)

(defrule result-5
       (and(av (a work   ) (v спутв))
```

```
           (av (a all_work_man_skill) (v all_work_man_skill_yes))
         (or(av (a work_inst ) (v work_inst_no))
           (av (a work_livery) (v work_livery_no))
         ))
         (not (result ?))
          =>
(assert (result "Работы проводить нельзя!!!"))
)

(defrule result-6
       (and(av (a work   ) (v спунв))
         (or(av (a worker_qualification ) (v worker_qualification_yes))
           (av (a work_inst ) (v work_inst_no))
         ))
         (not (result ?))
          =>
(assert (result "Работы проводить нельзя!!!"))
)

(defrule result-7
       (and(av (a number_of_workers ) (v number_of_workers_yes))
           (av (a worker_qualification ) (v worker_qualification_yes))
         (or(av (a work   ) (v сппу))
           (av (a work   ) (v эп))
         ))
         (not (result ?))
          =>
(assert (result "Работы проводить нельзя!!!"))
)

(defrule ok-result ""
    (declare (salience -10))
   (not (result ?))
   =>
   (assert (result "Всё в порядке. Необходимые работы можно выполнять."))
)
```

```
(defrule print-result ""
   (declare (salience 10))
   (result ?item)
    =>
   (printout            t "Конечный результат:")
   (printout            t crlf)
   (format t            " %s%n%n%n" ?item)
)
(defrule system-banner ""
(declare (salience 10) )
 =>
(printout t crlf crlf)
(printout  t  "************************************************************"
crlf)
(printout t "          *Консультация по электробезопасности*" crlf)
(printout  t  "************************************************************"
crlf)
(printout t "Если необходимо завершить консультацию введите: \\x" crlf)
(printout t crlf crlf)
)
```

Група «Захисні засоби»

```
(deftemplate av  (slot a) (slot v))
(deffunction ask-question (?question $?allowed-values)
   (printout t ?question crlf)
   (bind ?answer (read))
   (if (lexemep ?answer)
      then
         (bind ?answer (lowcase ?answer)))
    (if (= ?answer \x)
        then
        (printout t "ОТМЕНЕНО ПОЛЬЗОВАТЕЛЕМ!!!" crlf)
            (halt)
            (reset)
            (clear)
        else
   (while (not (member$ ?answer ?allowed-values)) do
```

```
        (if (= ?answer \x)
         then
        (halt)
        (reset)
        (clear)
        (printout t "ОТМЕНЕНО ПОЛЬЗОВАТЕЛЕМ!!!" crlf)
        (break))
       (printout t ?question crlf)
       (bind ?answer (read))
       (if (lexemep ?answer)
          then
           (bind ?answer (lowcase ?answer)))))
     ?answer
)

(deffunction yes-or-no-p (?question)
(bind ?response (ask-question ?question да нет))
(if (or (eq ?response да) (eq ?response y))
then
TRUE
else
FALSE)
)
(defrule answer-1
  (declare (salience 9))
 (not (av (a voltage   ) (v ?)))
  (not (result ?))
=>
(if (yes-or-no-p  "Напряжение свыше 1000В? (да/нет) ? ")
then
(assert (av (a voltage) (v voltage_high)))
else
(assert (av (a voltage) (v voltage_low)))
))

(defrule answer-2
  (declare (salience 8))
```

```
 (not (av (a work  ) (v ?)))
  (not (result ?))
=>
(if (yes-or-no-p    "Есть одно из дополнительных защитных средств,
необходимых для данной работы? (да/нет) ? ")
then
(assert (av (a work ) (v work_yes)))
else
(assert (av (a work ) (v work_no)))
))


(defrule answer-3
  (declare (salience 7))
 (not (av (a live_parts_guard ) (v ?)))
  (not (result ?))
=>
(if (yes-or-no-p   "Возможно установить ограждение токоведущих частей?
(да/нет) ? ")
then
(assert (av (a live_parts_guard) (v live_parts_guard_yes)))
else
(assert (av (a live_parts_guard) (v live_parts_guard_no)))
))

(defrule answer-4
  (declare (salience 6))
 (not (av (a insulating_lining   ) (v ?)))
  (not (result ?))
=>
(if (yes-or-no-p  "Вместо ограждения используются изолирующие накладки ?
(да/нет) ? ")
then
(assert (av (a insulating_lining) (v insulating_lining_yes)))
else
(assert (av (a insulating_lining) (v insulating_lining_no)))
))
```

```
(defrule answer-5
  (declare (salience 5))
 (not (av (a work_mans ) (v ?)))
  (not (result ?))
=>
(if (yes-or-no-p  "Количество работников в брегаде меньше 2? (да/нет) ? ")
then
(assert (av (a work_mans) (v work_mans_yes)))
else
(assert (av (a work_mans) (v work_mans_no)))
))

(defrule answer-6
  (declare (salience 4))
 (not (av (a all_work_man_skill   ) (v ?)))
  (not (result ?))
=>
(if  (yes-or-no-p    "Квалификация  каждого  работника  бригады  меньше  3?
(да/нет) ? ")
then
(assert (av (a all_work_man_skill) (v all_work_man_skill_yes)))
else
(assert (av (a all_work_man_skill) (v all_work_man_skill_no)))
))

(defrule result-1
       (and(av (a voltage) (v voltage_high))
           (av (a work ) (v work_no))
        )
        (not (result ?))
         =>
(assert (result "Работы проводить нельзя!!!"))
)

(defrule result-2
       (and(av (a live_parts_guard) (v live_parts_guard_no))
```

```
            (av (a insulating_lining) (v insulating_lining_yes))
         (or(av (a work_mans) (v work_mans_yes))
            (av (a all_work_man_skill) (v all_work_man_skill_yes))
         ))
         (not (result ?))
          =>
(assert (result "Работы проводить нельзя!!!"))
)
(defrule ok-result ""
    (declare (salience -10))
   (not (result ?))
   =>
   (assert (result "Всё в порядке. Необходимые работы можно выполнять."))
)


(defrule print-result ""
   (declare (salience 10))
   (result ?item)
    =>
   (printout             t "Конечный результат:")
   (printout             t crlf)
   (format t             " %s%n%n%n" ?item)
)
(defrule system-banner ""
(declare (salience 10) )
 =>
(printout t crlf crlf)
(printout  t  "*************************************************************"
crlf)
(printout t "          *Консультация по электробезопасности*" crlf)
(printout  t  "*************************************************************"
crlf)
(printout t "Если необходимо завершить консультацию введите: \\x" crlf)
(printout t crlf crlf)
)
```

Група «Роботи в охоронній зоні електричних мереж»

```
(deftemplate av  (slot a) (slot v))
(deffunction ask-question (?question $?allowed-values)
   (printout t ?question crlf)
   (bind ?answer (read))
   (if (lexemep ?answer)
      then
         (bind ?answer (lowcase ?answer)))
    (if (= ?answer \x)
        then
        (printout t "ОТМЕНЕНО ПОЛЬЗОВАТЕЛЕМ!!!" crlf)
            (halt)
            (reset)
            (clear)
        else
   (while (not (member$ ?answer ?allowed-values)) do
       (if (= ?answer \x)
        then
       (halt)
       (reset)
       (clear)
       (printout t "ОТМЕНЕНО ПОЛЬЗОВАТЕЛЕМ!!!" crlf)
       (break))
      (printout t ?question crlf)
      (bind ?answer (read))
      (if (lexemep ?answer)
         then
          (bind ?answer (lowcase ?answer)))))
     ?answer
)

(deffunction yes-or-no-p (?question)
(bind ?response (ask-question ?question да нет))
(if (or (eq ?response да) (eq ?response y))
then
TRUE
else
```

```
FALSE)
)
(defrule answer-1
  (declare (salience 9))
 (not (av (a security_zone   ) (v ?)))
  (not (result ?))
=>
(if (yes-or-no-p  "Объект находится в охранной зоне электрических сетей?
(да/нет) ? ")
then
(assert (av (a security_zone) (v security_zone_yes)))
else
(assert (av (a security_zone) (v security_zone_no)))
))

(defrule answer-2
  (declare (salience 6))
 (not (av (a work ) (v ?)))
  (not (result ?))
=>
(if  (yes-or-no-p   "Работа должна проводиться по согласованию с ЭСО
(Электро-снабжающая организация)? (да/нет) ? ")
then
(assert (av (a work ) (v work_yes)))
else
(assert (av (a work ) (v work_no)))
))

(defrule answer-3
  (declare (salience 5))
 (not (av (a work_matching ) (v ?)))
  (not (result ?))
=>
(if (yes-or-no-p  "Работа согласованная? (да/нет) ? ")
then
(assert (av (a work_matching ) (v work_matching_yes)))
else
```

```
(assert (av (a work_matching ) (v work_matching_no)))
))

(defrule answer-4
  (declare (salience 7))
 (not (av (a construction_works   ) (v ?)))
  (not (result ?))
=>
(if (yes-or-no-p  "Должны производиться строительные работы? (да/нет) ? ")
then
(assert (av (a construction_works ) (v construction_works_yes)))
else
(assert (av (a construction_works ) (v construction_works_no)))
))

(defrule answer-5
  (declare (salience 8))
 (not (av (a cargo_handling  ) (v ?)))
  (not (result ?))
=>
(if  (yes-or-no-p   "Должны производиться погрузочно-разгрузочные работы?
(да/нет) ? ")
then
(assert (av (a cargo_handling) (v cargo_handling_yes)))
else
(assert (av (a cargo_handling) (v cargo_handling_no)))
))

(defrule result-1
       (and(av (a security_zone) (v security_zone_yes))
           (av (a work ) (v work_yes))
           (av (a work_matching ) (v work_matching_no)))
        (not (result ?))
         =>
(assert (result "Работы проводить нельзя №1!!!"))
)
```

```
(defrule result-2
       (and(av (a construction_works ) (v construction_works_yes))
           (av (a security_zone) (v security_zone_yes))
       )
        (not (result ?))
         =>
(assert (result "Без согласования выполнения этих работ с организацией, эксплуатирующей ЭС, и получения разрешения на их выполнение работы выполнять нельзя!!!"))
)
(defrule result-3
       (and(av (a cargo_handling) (v cargo_handling_yes))
           (av (a security_zone) (v security_zone_yes))
        )
        (not (result ?))
         =>
(assert (result "Без согласования выполнения этих работ с организацией, эксплуатирующей ЭС, и получения разрешения на их выполнение работы выполнять нельзя!!!"))
)
(defrule ok-result ""
    (declare (salience -10))
   (not (result ?))
   =>
   (assert (result "Всё в порядке. Необходимые работы можно выполнять."))
)
(defrule print-result ""
   (declare (salience 10))
   (result ?item)
    =>
   (printout            t "Конечный результат:")
   (printout            t crlf)
   (format t            " %s%n%n%n" ?item)
)
(defrule system-banner ""
(declare (salience 10) )
 =>
```

```
(printout t crlf crlf)
(printout  t  "************************************************************"
crlf)
(printout t "          *Консультация по электробезопасности*" crlf)
(printout  t  "************************************************************"
crlf)
(printout t "Если необходимо завершить консультацию введите: \\x" crlf)
(printout t crlf crlf)
)
```