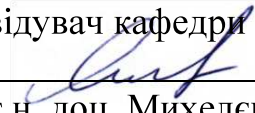


МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ
ІМЕНІ АДМІРАЛА МАКАРОВА

Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій

"Допущений до захисту"
Завідувач кафедри ІУСТ


к.т.н, доц. Михелєв І.Л.
"___" _____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти "магістр"
(освітньо-кваліфікаційний рівень)

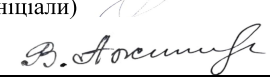
на тему **Дослідження методів теорії масового обслуговування та
розробка проекту інформаційної системи апарату Миколаївської
обласної ради**

Виконав: студент 6 курсу, групи 6145м
напряму підготовки (спеціальності)
124 – "Системний аналіз".

(шифр і назва напряму підготовки, спеціальності)

Галатюк Д.А.

(прізвище та ініціали)

Керівник Ажищев В.Ф. 

(прізвище та ініціали)

м. Миколаїв – 2024 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ
ІМЕНІ АДМІРАЛА МАКАРОВА

Інститут, факультет Навчально-науковий інститут комп'ютерних наук та управління проектами

Кафедра Інформаційних управляючих систем та технологій

Рівень вищої освіти другий (магістерський)

Спеціальність 124 – "Системний аналіз"

Освітня програма "Системний аналіз"

"ЗАТВЕРДЖУЮ"

Гарант освітньої програми

д.ф-м.н, проф. Ушкац М.В.

"___" _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття ступеня вищої освіти "магістр"

Студенту Галатюк Данило Андрійович

1. Тема проекту (роботи) Дослідження методів теорії масового обслуговування та розробка проекту інформаційної системи апарату Миколаївської обласної ради
керівник проекту (роботи) Ажищев Віктор Федорович,
затверджені наказом вищого навчального закладу від "08" 10 2024 року № 1055-уч.

2. Строк подання студентом проекту (роботи) 15 грудня 2024 року.

3. Вихідні дані до проекту (роботи) ДСТУ щодо обробки інформації, літературні джерела, технічна документація на існуючі аналоги систем, матеріали практики.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):

- Титульний аркуш, завдання на дипломну роботу, анотація (українською, англійською), зміст, перелік умовних означень, символів, одиниць та термінів (при необхідності).
- Вступ.
- Аналіз предметної області і постановка задачі.
- Розробка проекту автоматизованої системи.
- Реалізація проекту автоматизованої системи (технічна документація по проекту).
- Розділ з охорони праці.

- Висновки.
- Список використаної літератури.
- Додатки: загальний опис підсистеми, керівництво користувача, керівництво адміністратора, програма і методика випробувань, графічні інтерфейси, код програми.

5. Перелік графічного матеріалу

- 5.1 Структура документообігу обласної ради.
- 5.2 Концепції розробки ІС.
- 5.3 Діаграма рівнів використання.
- 5.4 Концептуальна модель даних.
- 5.5 Логічна модель даних.
- 5.6 Схема комп'ютерної мережі.
- 5.7. Екранні форми (графічний інтерфейс).

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1.	Ажищев В.Ф.	05.09.2023	22.12.2023
2.	Ажищев В.Ф.	23.12.2023	15.06.2024
3.	Ажищев В.Ф.	15.06.2024	25.09.2024
4.	Ажищев В.Ф.	25.09.2024	14.11.2024
5.	Гурець Н.В.	25.09.2024	14.11.2024
6.	Гурець Н.В.	25.09.2024	14.11.2024

7. Дата видачі завдання 05 вересня 2024 р.

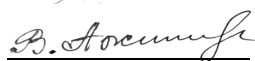
КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1.	Вивчення і опис предметної області	22.12.2024	
2.	Розробка концепції ІС	15.03.2024	
3.	Розробка проектних рішень по ІС	15.06.2024	
4.	Реалізація проекту, розробка робочої документації	25.09.2024	
5.	Вирішення питань з охорони праці	14.11.2024	
6.	Виконання графічної частини роботи	14.11.2024	
7.	Оформлення пояснювальної записки	14.11.2024	
8.	Подання роботи на кафедральний захист	16.11.2024	
9.	Подання роботи рецензенту	20.11.2024	
10.	Подання роботи на державний захист	15.12.2024	

Студент

 Галатюк Д.А.

Керівник проекту (роботи)

 к.т.н., доцент Ажищев В.Ф.

АНОТАЦІЯ

Дана магістерська робота присвячена науковому дослідженню на тему "Дослідження методів теорії масового обслуговування та розробка проекту інформаційної системи апарату Миколаївської обласної ради".

Метою даної роботи є підвищення ефективності ведення документообігу обласної ради шляхом створення інформаційної системи, що дозволяла б спільну швидку роботу та ефективний контроль документів.

Задачами дослідження є аналіз функціоналу та проблем систем електронного документообігу, автоматизація роботи з документами, забезпечення спільної роботи з даними, забезпечення безпеки і надійності зберігання інформації.

Об'єкт дослідження – система документообігу державних структур.

Предмет дослідження – методи теорії масового обслуговування.

Науковою новизною є вибір раціонального методу для розробки проекту системи масового обслуговування документообігу обласної ради.

Дана дипломна робота викладена на 135 сторінках друкованого тексту, містить 21 рисунок, 3 таблиці та має 3 додатки які містять технічне завдання, текст виконаної програми, інструкцію користувача. Список літератури з 13 найменувань.

Робота виконана на українській мові.

Ключові слова: теорія масового обслуговування; моделі масового обслуговування; кількість каналів обслуговування; черги; лінійні та нелінійні моделі обслуговування; системи управління; інформаційні технології в органах влади; архітектура інформаційної системи; автоматизація адміністративних послуг; електронний документообіг.

ABSTRACT

This master's work is devoted to scientific research on "Research methods of queuing theory and the development of the draft information system for processing requests of deputies of the regional council."

The purpose of this work is to improve the efficiency of the document management of the regional council by creating an information system that would allow for a common quick work and effective control of documents.

The objectives of the study are to analyze the functional and problems of electronic document management systems, automate the work with documents, provide joint work with data, ensure the security and reliability of information storage.

The object of research is the system of document circulation of state structures.

The subject of the research are methods of queuing theory.

The scientific novelty of the work is the choice of a rational method for the development of a draft system of mass servicing of the document management of the regional council.

This thesis is presented in 135 pages of printed text, contains 21 figures, 3 tables, and has 3 applications containing the technical task, the text of the program, the instruction. List of literature from 13 titles.

The work was done in Ukrainian.

Keywords: theory of queuing; queuing models; number of service channels; queues; linear and nonlinear service models; management systems; information technologies in government bodies; information system architecture; automation of administrative services; electronic document management.

ЗМІСТ

Вступ	
1. Дослідження методів теорії масового обслуговування	
1.1 Актуальність та постановка задачі	
1.2 Дослідження методів теорії масового обслуговування	
1.3 Аналіз структури та вибір методів теорії масового обслуговування.....	
2. Розробка проекту інформаційної системи	
2.1 Загально системні рішення.....	
2.2 Діаграми UML	
2.3 Рішення з інформаційного забезпечення	
2.4 Рішення з організаційного забезпечення	
3. Розробка робочої документації	
3.1 Загальний опис системи.....	
3.2 Методика випробувань системи управління	
3.3 Розробка програми та методики випробувань, порядок здавання та приймання системи.....	
4. Розрахунок економічної ефективності впровадження ПЗ	
4.1 Вступ.....	
4.2 Розрахунок витрат на створення та експлуатацію ПЗ інформаційної системи	
4.3 Економічна ефективність розробки і впровадження системи	
5. Охорона праці	
5.1 Загальні визначення та актуальність питання охорони праці	
5.2 Аналіз небезпечних і шкідливих факторів при роботі з ПК	
5.3 Розрахунок системи загального освітлення приміщення з ПК	
5.4 Розробка заходів щодо забезпечення сприятливих умов праці при роботі з персональним комп'ютером	
6. Охорона навколишнього середовища	
6.1 Вступна частина	
6.2 Забруднення навколишнього середовища підприємствами	
6.3 Основні заходи щодо зниження забруднення атмосферного повітря викидами автомобільного транспорту	
6.4 Заходи запобігання забруднення навколишнього середовища застарілою комп'ютерною технікою підприємства.....	
Висновки	
Список використаної літератури	
Додаток А – Технічне завдання	
Додаток Б – Інструкція з експлуатації	
Додаток В – Текст програми	

ВСТУП

Одним із визначальних факторів впливу на результат управлінського процесу є діяльність органів місцевого самоврядування, які стоять найближче до громадян і місцевих громад. В Україні гарантується місцеве самоврядування, що здійснюється територіальними громадами селищ, міст, як безпосередньо, так і через сільські, селищні, міські ради і їхні виконавчі органи, а також через районні й обласні ради, що представляють загальні інтереси територіальних общин сіл, селищ і міст.

У сучасному українському суспільстві діяльність як органів державної влади, так і місцевого самоврядування трансформується у бік якісних змін за рахунок використання новітніх інформаційних технологій. Новітні інформаційні технології – це сукупність сучасних електронних технічних засобів і програмного забезпечення, а також організаційних форм і методів їх застосування в інформаційній роботі, яка спрямована на ефективне використання інформаційних ресурсів. В органах місцевого самоврядування новітні інформаційні технології представлені не тільки засобами інформатизації на основі комп'ютерів, різними методами вирішення завдань але й принципово новими формами діяльності даних структур.

Нові інформаційні технології мають ряд відмінних рис і переваг порівняно з традиційними інформаційними технологіями. Основні переваги новітніх технологій полягають у тому, що вони: переводять інформаційний обмін на без паперовий рівень; прискорюють підготовку документів, підвищують якість їх оформлення; автоматизують більшість функцій в інформаційних і комунікаційних процесах; замінюють рутинну працю на творчу; перетворюють масиви документів у інформаційні ресурси розвитку й вирішення складних проблем життєдіяльності суспільства.

Важливим компонентом новітніх інформаційних технологій є методи інформаційної роботи, які адаптовані до можливостей електронних засобів комунікації й обробки інформації в органах місцевого самоврядування. Методи визначають найбільш ефективні шляхи й способи вирішення на основі

комп'ютера таких завдань інформаційної роботи, як цільовий пошук і збір, оперативне оброблення, надійність збереження, перероблення й ефективне використання інформації. Залежно від сфери інформаційної діяльності форми застосування новітніх інформаційних технологій мають свої особливості. У вирішенні комунікативних завдань використовується електронна пошта, телекомунікаційні мережі, у професійній діяльності – автоматизовані робочі місця.

Застосування новітніх технологій в органах місцевого самоврядування забезпечить: загальне оброблення документів, збереження документів; дистанційну роботу співробітників з документами; уведення, персональне оброблення й передавання документів; роздрукування текстів документів; обмін інформацією між базами даних; інформаційну підтримку рішень; надання громадянам потрібних інформаційних послуг.

Основними перспективними напрямками провадження новітніх технологій у діяльність органів місцевого самоврядування в Україні є такі: створення та широке запровадження у діловодну практику автоматизованої інформаційної системи аналізу вхідних, вихідних та внутрішніх документів; яка забезпечить зниження випадків втрати документів; оперативність проходження та виконання документа; підвищить якість документування завдяки шаблонам документів, які зберігаються у пам'яті персонального комп'ютера; широке використання електронних інформаційних ресурсів інших органів державної влади.

Впровадження "електронного уряду" та "електронної демократії" в Україні передбачає залученням широких кіл громадськості до різних видів діяльності державних установ шляхом електронних голосувань, опитувань не виключаючи інших форм вирішення питань управління державою. Використання сучасних інформаційно-комунікаційних технологій забезпечить співпрацю органів місцевого самоврядування з громадянами, бізнесом, громадськими організаціями, зробить їх діяльність прозорою і відкритою, наповнить новим змістом службові комунікації, встановить новий порядок взаємовідносин між різними організаційними структурами.

В Україні потрібно ширше використовувати досвід європейських країн впровадження електронного урядування.

Удосконалення та підвищення ефективності роботи органів місцевого самоврядування в значній мірі залежить від використання сучасних інформаційних технологій, які забезпечують інформування громадськості про їх діяльність; надання он-лайн-послуг, консультацій; здійснюють залучення громадян до прийняття рішень; громадський контроль за діяльністю.

Нові інформаційні технології містять різні електронні засоби опрацювання, збереження, управління інформацією, надання її замовнику. Ефективне їх використання в органах місцевого самоврядування потребує відповідного сучасного технічного й програмного забезпечення, створення громадських точок доступу до Інтернету, які забезпечать співпрацю з громадянами.

Метою даної роботи є підвищення ефективності ведення документообігу обласної ради шляхом створення інформаційної системи, що дозволяла б спільну швидку роботу та ефективний контроль документів.

Задачами дослідження є аналіз функціоналу та проблем систем електронного документообігу, автоматизація роботи з документами, забезпечення спільної роботи з даними, забезпечення безпеки і надійності зберігання інформації.

Об'єкт дослідження – система документообігу державних структур.

Предмет дослідження – методи теорії масового обслуговування.

Науковою новизною є вибір раціонального методу для розробки проекту системи масового обслуговування документообігу обласної ради.

1. ДОСЛІДЖЕННЯ МЕТОДІВ ТЕОРІЇ МАСОВОГО ОБСЛУГОВУВАННЯ.

1.1 Актуальність та постановка задачі.

Документи використовуються на всіх етапах розвитку суспільства, в будь-яких правових системах і реалізуються на різних матеріальних носіях. З часом все змінюється – носії інформації, специфіка "закріплення" інформації на них, види обов'язкових реквізитів, які надають такій інформації юридичний статус документа. З'явилася така форма подання інформації, як електронний документ.

Багато в чому ефективність ведення документації залежить від часу, який витрачається на проходження документу відповідних адміністративних етапів, перед тим як він буде відправлений на виконання. У виконавчих органах державного самоуправління спостерігається, що ефективність ведення документації, на основі якої здійснюється прийняття важливих адміністративних рішень, є вкрай низькою через необхідність вручну створювати звіти про документи та здійснювати пошук і редагування певних відомостей.

Наявність непотрібної паперової тяганини, необхідність відправлення документів через проміжні відділи, затягування важливих адміністративних рішень – все це є реальною проблемою сьогодення в органах державного самоуправління, через що розвиток різних галузей, за які відповідальний державний апарат, в даному регіоні сповільнюється. Вирішення даної проблеми є актуальним через те, що облік та виконання адміністративних рішень з використанням інформаційної підсистеми має пришвидшитись, а з ним і розвиток певних галузей.

Мета даної роботи є підвищення ефективності ведення документообігу шляхом розробки проекту інформаційної системи документообігу на основі методів теорії масового обслуговування. Дана система буде забезпечувати автоматизацію роботи з документами, спільну роботу з даними, безпеку і надійність зберігання документів, швидкий пошук потрібного документу, його редагування, супровід і отримання інформаційних довідок або передачу іншим

особам в електронному або в паперовому вигляді. Також підвищення оперативності, контролю щодо дотримання законодавств при прийнятті рішень обласної ради. В наслідок цього очкується поліпшення якості та мінімізація часових витрат на підготовку рішень і складання документів, спрощення та прискорення доступу до інформації.

Об'єкт дослідження – дослідження методів теорії масового обслуговування, аналіз та ефективний вибір методів для проектування системи.

1.2 Дослідження методів теорії масового обслуговування.

Предмет ТМО – системи масового обслуговування (СМО) і мережі масового обслуговування. Під СМО розуміють динамічну систему, призначену для ефективного обслуговування випадкового потоку заявок при обмежених ресурсах системи. Узагальнена структура СМО наведена на рисунку 1.1.

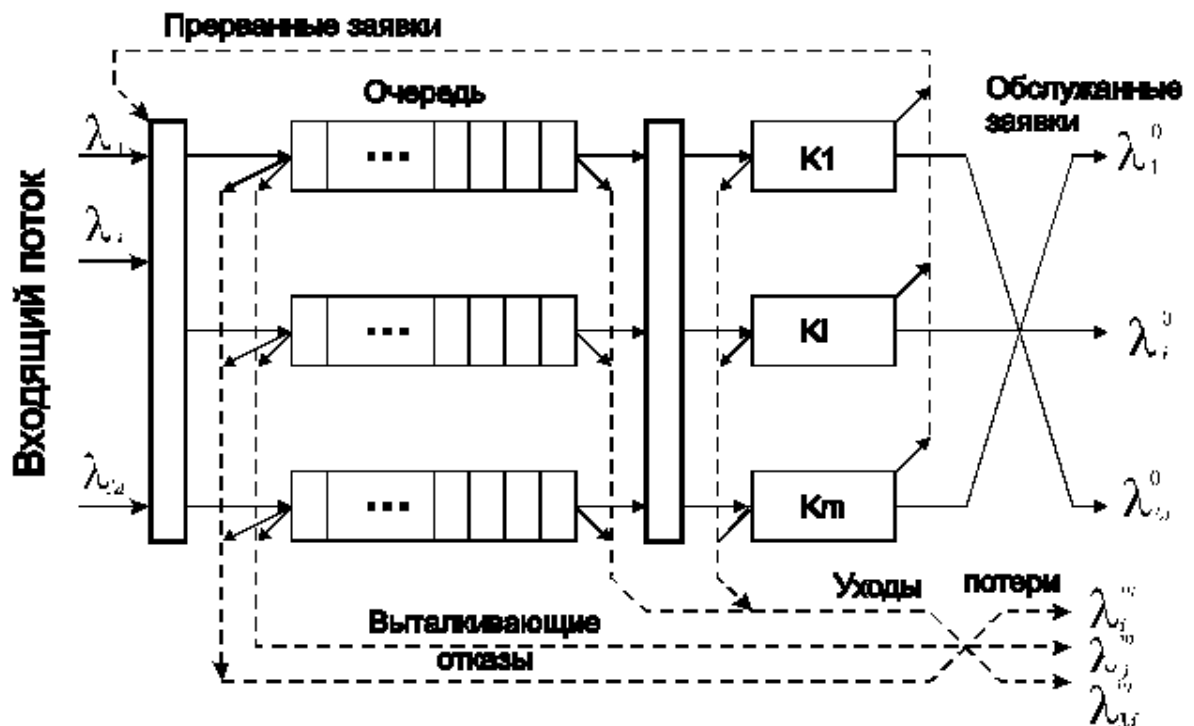


Рисунок 1.1 – Схема СМО.

Однорідні заявки що надходять на вхід СМО в залежності від причини що породжує їх діляться на типи, інтенсивність потоку заявок типу i ($i = \overline{1, M}$) позначено λ_i . Сукупність заявок всіх типів – вхідний потік СМО.

Обслуговування заявок виконується m каналами. Розрізняють універсальні та спеціалізовані канали обслуговування. Для універсального каналу типу j вважається відомими функції розподілу $F_{ij}(t)$ тривалості обслуговування заявок довільного типу. Для спеціалізованих каналів функції розподілу тривалості обслуговування каналів заявок деяких типів є невизначеними, призначення цих заявок на даний канал.

В якості процесу обслуговування можуть бути представлені різні по своїй фізичній природі процеси функціонування економічних, виробничих, технічних та інших систем, наприклад заявки на обробку інформації ЕОМ від віддалених терміналів і т.д. При цьому характерним для роботи таких об'єктів є випадкове поводження заявок (вимог) на обслуговування і завершення обслуговування у випадкові моменти часу.

Q -схеми можна досліджувати аналітично і імітаційними моделями. Останнє забезпечує більшу універсальність.

Розглянемо поняття масового обслуговування. У будь-якому елементарному акті обслуговування можна виділити дві основні складові: очікування обслуговування заявкою і власне обслуговування заявки. Це можна відобразити у вигляді деякого i -ого приладу обслуговування Π_i , що складається з накопичувача заявок, в якому може знаходитися одночасно $l_i = 0 \dots L_i^H$ заявок, де L_i^H – ємність i -ого накопичувача, і каналу обслуговування заявок, k_i .

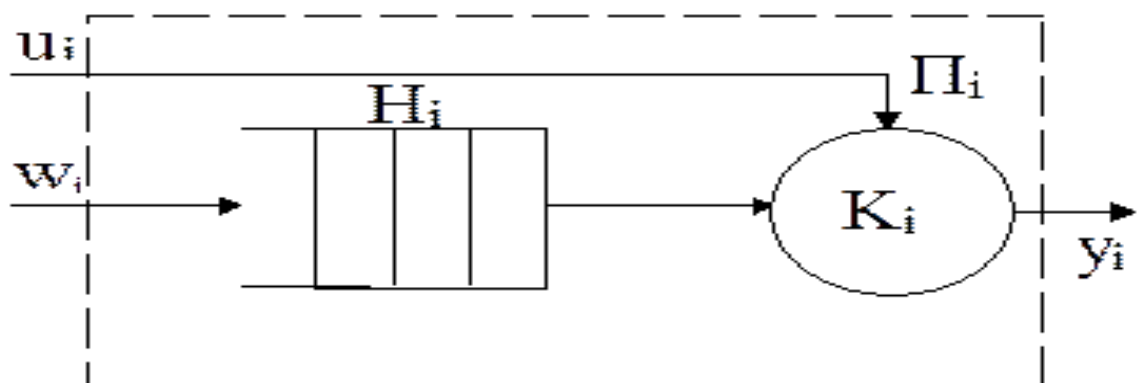


Рисунок 1.2 – Схема приладу СМО

На кожен елемент приладу обслуговування Π_i надходять потоки подій: в накопичувач H_i потік заявок w_i ; на канал k_i – потік обслуговування u_i .

Потоком подій Π_n називається послідовність подій, що відбуваються одне за іншим в якісь випадкові моменти часу. Розрізняють потоки однорідних і неоднорідних подій.

Однорідний Π_n характеризується тільки моментами надходження цих подій (що викликають моменти) і задається послідовністю

$$\{t_n\} = \{0, t_1, t_2, \dots, t_n, \dots\},$$

де t_n – момент надходження n -ої події – невід'ємне дійсне число.

Однорідний Π_n може бути також заданий у вигляді послідовності проміжків часу між n -им і $(n-1)$ -им подіями $\{t_n\}$.

Неоднорідним Π_n називається послідовність

$$\{t_n, f_n\},$$

де t_n – викликають моменти; f_n – набір ознак події.

Наприклад, може бути задана приналежність до того чи іншого джерела заявок, наявність пріоритету, можливість обслуговування тим або іншим типом каналу і т.п.

Розглянемо однорідний Π_n , для якого $\{t_n\}$ – випадкові величини, незалежні між собою. Тоді Π_n називається потоком з обмеженим післядією.

Π_n називається ординарним, якщо ймовірність того, що на малий інтервал часу D_t , що примикає до моменту часу t потрапляє більше однієї події $P_{\geq 1}(T, D_t)$ зневажливо мала.

Якщо для будь-якого інтервалу D_t подія $P_0(T, D_t) + P_1(T, D_t) + P_{\geq 1}(T, D_t) = 1$, ймовірність потрапляння на інтервал D_t рівно однієї події.

Стаціонарним Π_n називається потік, для якого ймовірність появи того чи іншого числа подій на інтервалі часу t залежить від довжини цієї ділянки і не залежить від того, де на осі часу $0 \dots t$ узятий цю ділянку.

Для Π_n справедливо $0 \cdot P_0(T, D_t) + 1 \cdot P_1(T, D_t) = P_1(T, D_t)$ – середнє число

подій на інтервалі D_t . Середнє число подій, що настають на ділянці D_t в одиницю часу становить $\frac{P_1(T, D_t)}{D_t}$.

Стосовно до елементарного каналу обслуговування k_i можна вважати що інтервали часу між моментами появи заявок на вході k_i утворюють підмножину некерованих змінних, а потік обслуговування u_i , тобто інтервали часу між початком і закінченням обслуговування заявки утворюють підмножину керованих змінних.

Заявки, обслужених каналом k_i та заявки, покинувши прилад Π_i з різних причин не обслужених, утворюють вихідний потік y_i .

Процес функціонування приладу обслуговування Π_i можна представити як процес зміне станів його елементів у часі $Z_i(t)$. Перехід у новий стан для Π_i означає зміну кількості заявок, які в ньому знаходяться (в каналі k_i і накопичувачі H_i). Таким чином вектор станів для Π_i має вигляд:

$$Z_i = (Z_i^H; Z_i^K),$$

де Z_i^H – стан накопичувача ($Z_i^H = 0$ – накопичувач порожній, $Z_i^H = 1$ – у накопичувачі одна заявка, $Z_i^H = L_i^H$ – накопичувач зайнятий повністю);

Z_i^K – стан каналу k_i ($Z_i^K = 0$ – канал вільний, $Z_i^K = 1$ – канал зайнятий).

Q -схеми реальних об'єктів утворюються композицією багатьох елементарних приладів обслуговування Π_i . Якщо k_i різних приладів обслуговування з'єднані паралельно, то має місце багатоканальне обслуговування (багатоканальна Q -схема), а якщо прилади Π_i та їх паралельні композиції з'єднані послідовно, то має місце багатofазна обслуговування (багатofазна Q -схема).

Таким чином для завдання Q -схеми необхідно оператор сполучення R , що відображає взаємозв'язок елементів структури.

Зв'язок в Q -схемі зображують у вигляді стрілок (ліній потоку, що відображають напрямок руху заявок).

Розрізняють розімкнуті і замкнуті Q -схеми.

У розімкнутій вихідний потік не може знову вступити на який-небудь елемент, тобто зворотний зв'язок відсутній.

Власними (внутрішніми) параметрами Q -схеми будуть кількість фаз L_ϕ . Кількість каналів у кожній фазі L_{kj} , $j=1\dots L_\phi$. Кількість накопичувачів кожної фази L_{kj} , $k=1\dots L_\phi$. Ємність i -ого накопичувача L_i^H .

Слід зазначити, що в теорії масового обслуговування в залежності від ємності накопичувача застосовують наступну термінологію:

1. Системи з втратами ($L_i^H = 0$, накопичувач відсутній).
2. Системи з очікуванням (в яких є накопичувач нескінченної ємності для буферизації вимог, при цьому вимоги утворюють чергу).
3. Системи з обмеженою ємністю накопичувача H_i (змішані).

Позначимо всю сукупність власних параметрів Q -схеми як підмножина H .

Для завдання Q -схеми також необхідно описати алгоритми її функціонування, які визначають правила поведінки заявок в різних неоднозначних ситуаціях.

Залежно від місця виникнення таких ситуацій розрізняють алгоритми (дисципліни) очікування заявок в накопичувачі H_i і обслуговування заявок каналом k_i . Неоднорідність потоку заявок враховується за допомогою введення класу пріоритетів. Залежно від динаміки пріоритетів Q -схеми розрізняють статичні і динамічні.

Статичні пріоритети призначаються заздалегідь і не залежать від станів Q -схеми, тобто вони є фіксованими в межах вирішення конкретної задачі моделювання. Динамічні пріоритети виникають при моделюванні. Виходячи з правил вибору заявок з накопичувача H_i на обслуговування каналом k_i можна виділити відносні і абсолютні пріоритети. Відносний пріоритет означає, що заявка з вищим пріоритетом, що надійшла в накопичувач H , очікує закінчення обслуговування представляє заявки каналом k_i і тільки після цього займає канал. Абсолютний пріоритет означає, що заявка з вищим пріоритетом, що

надійшла в накопичувач, перериває обслуговування каналом k_i заявки з більш низьким пріоритетом і самі займає канал (при цьому витіснена з k_i заявка може або покинути систему, або може бути знову записана на якесь місце в H_i).

Необхідно також знати набір правил, за якими заявки залишають H_i і k_i :

для H_i – або правила переповнення, або правила догляду, пов'язані із закінченням часу очікування заявки в H_i ;

для k_i – правила вибору маршрутів або напрямів догляду.

Крім того, для заявок необхідно задати правила, за якими вони залишаються в каналі k_i , тобто правила блокувань каналу. При цьому розрізняють блокування k_i по виходу і по входу. Такі блокування відображають наявність керуючих зв'язків у Q -схемою, що регулюють потік заявок залежно від станів Q -схеми. Набір можливих алгоритмів поведінки заявок в Q -схемі можна представити у вигляді деякого оператора алгоритмів поведінки заявок A .

Таким чином Q -схема, що описує процес функціонування СМО будь-якої складності однозначно задається у вигляді набору множин:

$$Q = \langle W, U, H, Z, R, A \rangle.$$

Процесом обслуговування в СМО можуть бути представлені різні за фізичну природу процеси функціонування економічних, виробничих, технічних та інших систем, наприклад потоки товарів, потоки продукції, потоки деталей, потоки клієнтів і т.п.

Для будь-якої системи масового обслуговування характерна наявність трьох відмінних властивостей:

1. Об'єктів, у яких може виникнути потреба в задоволенні деяких заявок.
2. Агрегатів, призначених для задоволення заявок на обслуговування.
3. Спеціальної організації прийому в систему заявок та їх обслуговування.

1.3 Аналіз структури та вибір методів ТМО.

Скликання в обласній раді проходить наступним чином:

- Депутати обласної ради скликають сесію.
- Обговорюють всі нерозглянуті питання, або нові заявки від громадських груп.
- У разі якщо заявка прийнята на розгляд, голова комісії оголошує про голосування даного рішення.
- У разі якщо більшість голосів "за" приймається рішення щодо даної заявки.

Нове рішення на першій стадії буде містити такі пункти:

- Дата прийняття рішення.
- Номер сесії на якій прийняли рішення.
- Номер скликання на якій прийняли рішення.
- Текст рішення, який описує область дії та постанову ради.

Дата прийняття рішення – порядковий номер календарного дня, порядковий номер або найменування календарного місяця і порядковий номер календарного року.

Сесія – період, протягом якого рада збирається для обговорень законопроектів або просто поточних подій.

Номер сесії – це порядковий номер, що вказує на скликання ради.

Номер скликання – це порядковий номер, що вказує скликання ради у поточному році.

Текст рішення – це зв'язана і повністю послідовна сукупність символів, яка описує сукупність рішення ради, її повноваження и постанову.

Процес розгляду нових постанов має вигляд:

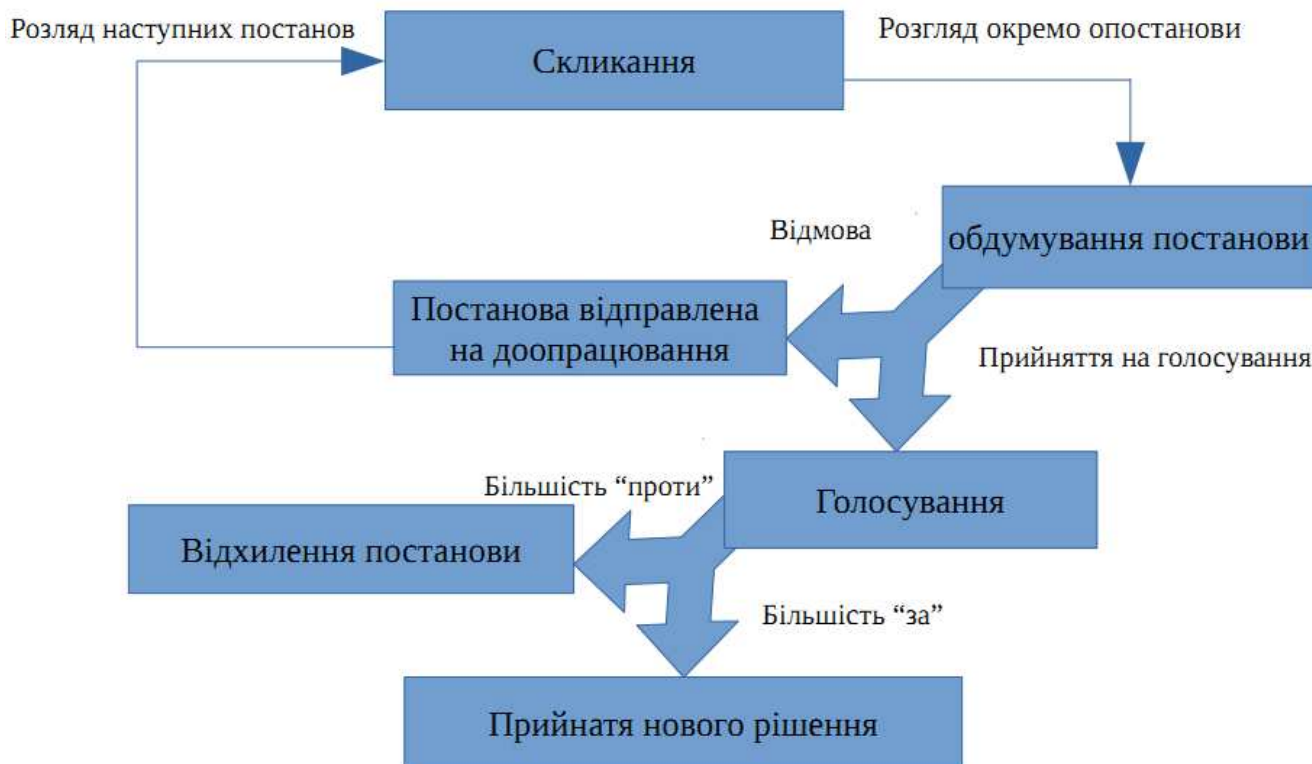


Рисунок 1.3 – Структура прийняття рішення на скликанні

Розглядаючи даний документообіг можна виділити 4-ри етапи:

1. Збір проектів (чернеток). На даному етапі проекти заносяться у базу даних.

2. Обговорення проектів, проекти які були допущенні до голосування формують повістку дня, проекти які були відхилені відправляються на доопрацювання.

3. Голосування. На даному етапі проект можуть ухвалити (в разі чого проект становиться документом), або відхилити.

4. Реєстрація. Етап запису документів в базу даних.

Аналізуючи данні етапи можна сказати що система є багатофазною. Розглянемо багатофазні системи більш детально.

Процес обробки запитів у багатьох реальних системах складається з їх послідовної обробки в декількох обслуговуючих пристроях. Системи обслуговування такого виду з'явилися прототипом мереж масового обслуговування і отримали назву багатофазних СМО.

Багатофазні СМО прийнято кодувати у вигляді послідовності відповідних

символів.

Символ A_1 описує вхідний потік на вхід ланцюжки з послідовних обслуговуючих пристроїв.

Символи B_k , n_k , t_k , $k = 1 \dots r$ описують відповідно розподіл часу обслуговування, число паралельних каналів і число місць в буфері для очікування/

Символи A_1 , B_k , $k = 1 \dots r$ приймають значення в тих же множинах, як і відповідні символи в описі однофазних систем.

Символ $*$ означає перехід запиту на вхід наступного обслуговуючого пристрою але закінчення його обслуговування у попередньому. У певному сенсі цей символ є заміником символу, що задає вид вхідного потоку у відповідну систему обслуговування, оскільки входить потік в дану СМО визначається виходять потоком запитів з попередньої СМО.

Якщо будь-який символ m_k приймає кінцеве значення, то виникає питання про поведінку багатофазної СМО в ситуації, коли буфер перед відповідною однофазної СМО вже повний, а на вхід цієї СМО надходить наступний запит. Зазвичай розглядаються два варіанти: вступник запит втрачається й чинить запит залишається на приладі, де він закінчив обслуговування, тимчасово блокуючи подальшу роботу цього приладу.

Відомі результати для широкого кола багатофазних СМО досить вичерпно описані. Коротко торкнемося СМО з обмеженням за довжиною черги.

Граф станів даної СМО представляє собою схему загибелі і розмноження (рис. 1.4).

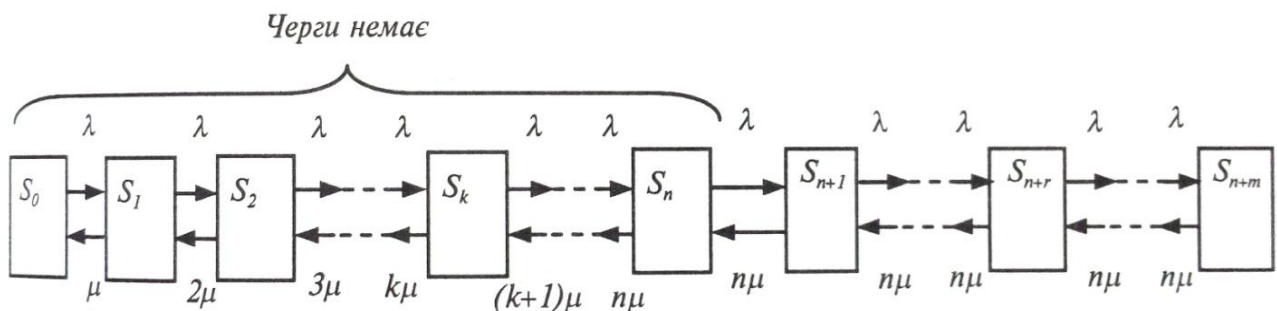


Рисунок 1.4 – Розмічений граф станів багатоканальної СМО з обмеженням за довжиною черги

Стани системи нумеруються за числом заявок, що знаходяться в системі:

S_0 – канали вільні, черги немає;

S_1 – один канал зайнято, решта вільні, черги немає;

S_2 – два канали зайнято, решта вільні, черги немає;

S_k – k каналів зайнято, решта вільні, черги немає;

S_n – зайнято всі n каналів, черги немає;

S_{n+1} – зайнято всі n каналів, одна заявка стоїть у черзі;

S_{n+r} – зайнято всі n каналів, r заявок стоять у черзі;

S_{n+m} – зайнято всі n каналів, m заявок стоять у черзі.

Процес утворення черг та затримок у багатоканальних СМО описано моделями, що наведені в таблиці 1.1.

Таблиця 1.1 – Багатоканальна СМО з обмеженням за довжиною черги

Основні параметри		Розрахункові формули
1.	Приведена щільність заявок	$Z = \frac{\lambda}{\mu}.$
2.	Імовірність станів СМО	$p_0 = \left[1 + \frac{Z}{1!} + \frac{Z^2}{2!} + \dots + \frac{Z^n}{n!} + \frac{Z^{n+1}}{n!(n-Z)} \right]^{-1};$ $p_1 = \frac{Z}{1!} \cdot p_0; \quad p_2 = \frac{Z^2}{2!} \cdot p_0; \quad \dots; \quad p_n = \frac{Z^n}{n!} \cdot p_0;$ $p_{n+1} = \frac{Z^{n+1}}{n \cdot n!} \cdot p_0; \quad p_{n+r} = \frac{Z^{n+r}}{n^r \cdot n!} \cdot p_0; \quad p_{n+m} = \frac{Z^{n+m}}{n^m \cdot n!} \cdot p_0,$ де m – кількість місць для очікування; r – кількість заявок у черзі; $r \leq m$.
3.	Імовірність відмови	$P_{\text{відм}} = p_{n+m} = \frac{Z^{n+m}}{n^m \cdot n!} \cdot p_0.$
4.	Відносна пропускна здатність	$q = 1 - P_{\text{відм}} = 1 - \frac{Z^{n+m}}{n^m \cdot n!} \cdot p_0.$
5.	Абсолютна пропускна здатність	$Q = \lambda \cdot q = \lambda \cdot \left(1 - \frac{Z^{n+m}}{n^m \cdot n!} \cdot p_0 \right)$

Характеристики функціонування		Математична модель
1.	Середнє число заявок у черзі	$\bar{n}_{\text{чрз}} = 1 \cdot p_{n+1} + 2 \cdot p_{n+2} + 3 \cdot p_{n+3} + \dots + m \cdot p_{n+m} =$ $= \frac{Z^{n+1}}{n \cdot n!} \cdot p_0 \cdot \frac{1 - (m+1) \cdot \alpha^m + m \cdot \alpha^{m+1}}{(1-\alpha)^2}.$
2.	Середнє число заявок у системі	$\bar{n}_c = \bar{n}_{\text{чрз}} + \bar{k}.$
3.	Середнє число зайнятих каналів	$k = \frac{Q}{\mu} = Z \cdot \left(1 - \frac{Z^{n+m}}{n^m \cdot n!} \cdot p_0 \right).$
4.	Середній час очікування у черзі	$\bar{t}_{\text{чрз}} = \frac{\bar{n}_{\text{чрз}}}{\lambda} = \frac{Z^n}{n \cdot n!} \cdot p_0 \cdot \frac{1 - (m+1) \cdot \alpha^m + m \cdot \alpha^{m+1}}{(1-\alpha)^2}.$
5.	Середній час перебування у системі	$\bar{t}_c = \bar{t}_{\text{чрз}} + \bar{t}_{\text{обс}} = \bar{t}_{\text{чрз}} + \frac{q}{\mu}.$

2. РОЗРОБКА ПРОЕКТУ ІНФОРМАЦІЙНОЇ СИСТЕМИ.

2.1 Загально системні рішення.

Інформаційна система керуючого справами виконавчого апарату ради повинна забезпечувати:

- 1) Зберігання зареєстрованих рішень в базі даних.
- 2) Зміна, додавання та редагування рішень в базу даних.
- 3) Швидкий пошук за критеріями.
- 4) Налагодження та редагування постійних комісій.
- 5) Налагодження та редагування постійних класифікаторів.
- 6) Налагодження та редагування пов'язаних комісій.
- 7) Генерація документів відповідно до обраного рішенням.

Виходячи із специфіки операцій, до інформаційної системи, що розробляється, пред'явлені наступні вимоги:

- функціональні можливості ІС повинні дозволяти керуючому справами виконавчого апарату ради здійснювати в автоматизованому режимі документообіг;
- базове (системне) програмне забезпечення повинне дозволяти роботу в реальному режимі часу, і допускати проведення у великих кількостях операцій введення/виведення, читання, запису, виведення на друк;

Прикладне програмне забезпечення повинне забезпечити:

- розмежування прав доступу до інформації;
- максимально швидкий пошук інформації;
- швидку генерацію звітів.

Користування інформаційною системою має бути простим і наочним, а робота з її використанням повинна знижувати кількість помилок, що допускаються.

2.1.1 Аналіз предметної області.

Згідно з Конституцією України, Закону "Про місцеве самоврядування в Україні" та іншими нормативними актами обласної ради Миколаївської

облдержадміністрації (далі рада) є органом місцевого самоврядування, що представляють спільні інтереси територіальної громади області, з правом юридичної особи, наділений власними повноваженнями, в межах яких він діє самостійно.

Рада приймає рішення в межах повноважень, наданих їй законодавством України, забезпечує та здійснює контроль за їх виконанням. Рішення ради є обов'язковими для виконання всіма розташованими на його території органами виконавчої влади, об'єднаннями громадян, підприємствами, установами та організаціями незалежно від форм власності, посадовими особами, а також громадянами, які постійно або тимчасово проживають на території ради.

Рішення – це розумова модель майбутньої індивідуальної або групової діяльності.

Для підготовки найбільш досконалих за формою і структурою проектів, що забезпечують максимально повне і точне відповідність форми нормативних приписів їх змісту, доступність, простоту і видимість нормативного матеріалу, вичерпний охоплення регульованих питань використовується законодавча техніка.

Правила законодавчої техніки різноманітні, але спільна ідея – створити найкращі умови і максимальні зручності для правильного тлумачення і застосування нормативного акта, досягти повноти, точності, доступності і компактності правових положень.

Відповідно до Закону України "Про всеукраїнський та місцевий референдуми".

Референдум – це спосіб прийняття громадянами України шляхом голосування законів України, інших рішень з важливих питань загальнодержавного та місцевого значення.

В Україні до основних стадій законотворчості ставляться:

- 1) Законодавча ініціатива.
- 2) Обговорення законопроекту.
- 3) Прийняття рішення.
- 4) Оприлюднення рішення.

Таким чином прийняття закону в ради зображено на рисунку 2.1.



Рисунок 2.1 – Модель прийняття законів.

Рада повинна контролювати виконання рішення та мати архіви цих законів. Але з ростом розміру документів система її документообігу ускладнюється, і вже стає непросто контролювати архів складений з паперу.

Документ – це засіб закріплення різними способами на відповідному матеріалі інформації про факти, події, явища об'єктивної дійсності та розумову діяльність людини.

Накопичення паперових документів може призвести до їх втрати, пошкодженню, ускладнити їх передачу. Природним чином перед керівництвом встає питання про систематизацію, створенні механізму поширення та архівації документів. Враховуючи сучасний технічний розвиток, ці проблеми найбільш оптимальним чином може вирішити комп'ютерна інформаційна система.

Схему документообігу представлена на рисунку 2.2.

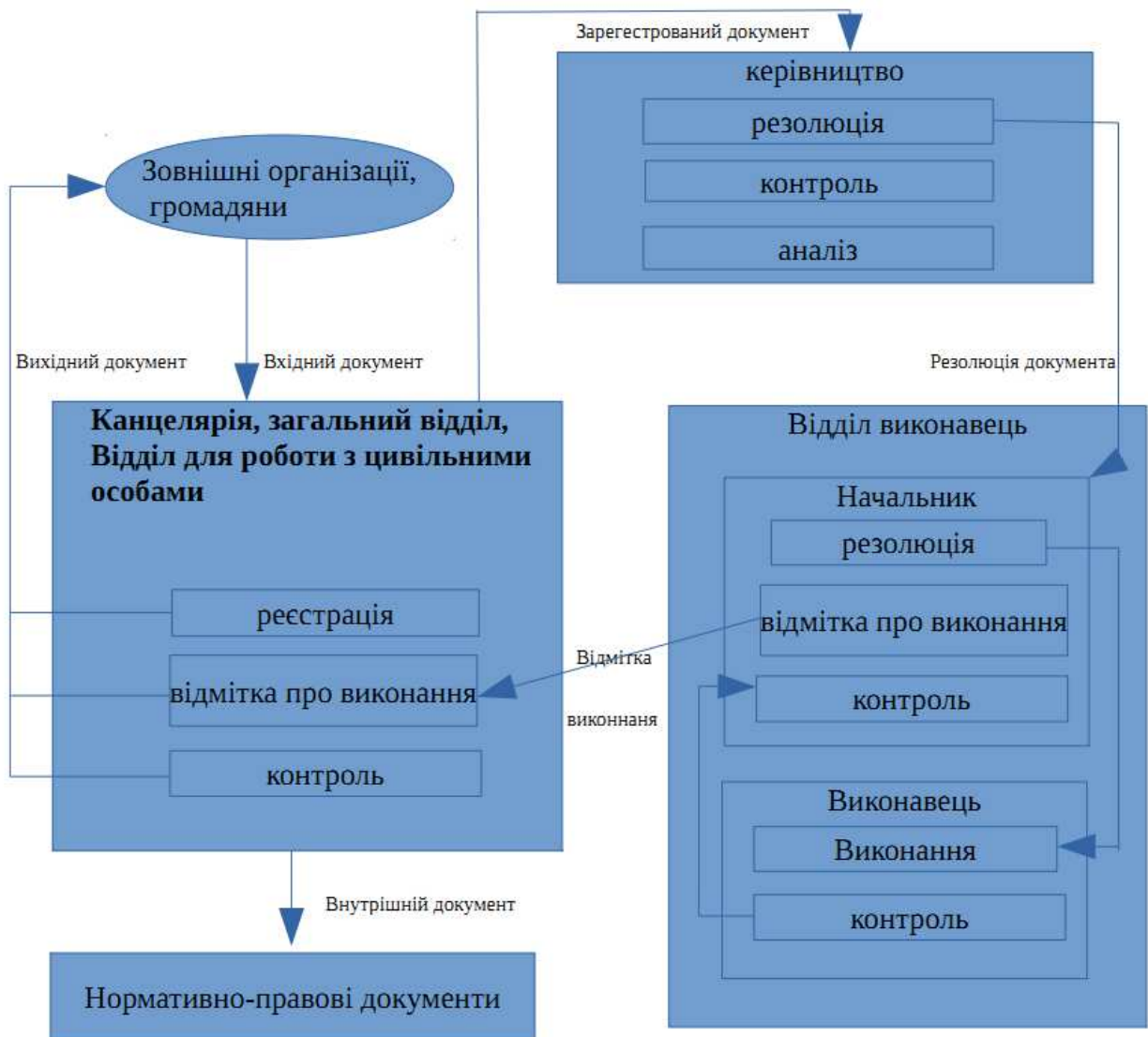


Рисунок 2.2 – Структурна схема документообігу.

Виходячи зі схеми документообігу в ради, можна виділити ключові концепції які стануть основними факторами для реалізації документообігу.

- 1) Обговорені рішення на референдумі, спираючись на потреби громадськості.
- 2) Ухвалення всіх нормативних актів стосовно рішення, і передача документів до керівництва.
- 3) Установка класифікатора, і призначення відповідальної комісії за виконання даного документа.
- 4) Створення необхідних документів, які будуть описувати рішення прийняте радою.
- 5) Контроль виконання рішення ради.

Рішення у раді проходить наступним чином:

- 1) Депутати ради скликають засідання.
- 2) Обговорюють всі нерозглянуті або нові заявки від громадських груп.
- 3) У разі якщо заявка пройшла розглядання, оголошується про голосування даного рішення.
- 4) У разі якщо більшість голосів проголосувало "за" тоді піймається рішення щодо даної заявки.

Нове рішення на першій стадії буде містити такі пункти:

- 1) Дата прийняття рішення.
- 2) Номер сесії на якій прийняли рішення.
- 3) Номер скликання на якій прийняли рішення.
- 4) Текст рішення, який описує область дії та постанову ради.

Дата прийняття рішення – порядковий номер календарного дня, порядковий номер або найменування календарного місяця і порядковий номер календарного року.

Сесія – період, протягом якого рада збирається для обговорень законопроектів або просто поточних подій.

Номер сесії – це порядковий номер, що вказує на скликання ради.

Номер скликання – це порядковий номер, що вказує на скликання ради у поточному році.

Текст рішення – це зв'язана і повністю послідовна сукупність символів, яка описує сукупність рішення ради, її повноваження и постанову.

Процес розгляду нових постанов зображений на рисунку 2.3.

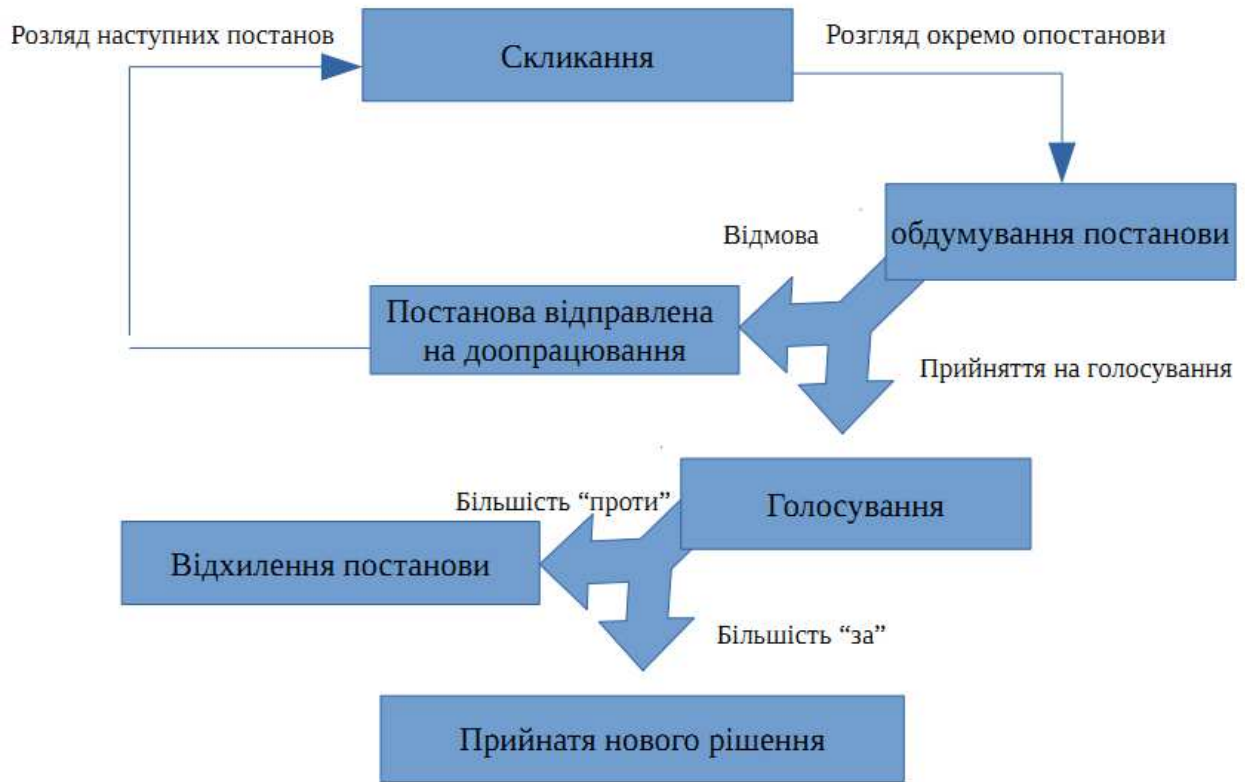


Рисунок 2.3 – Структура прийняття рішення на скликанні ради.

Наступний етап у створенні та реалізації нового рішення ради, це передача його керівництву яке має розподілити:

- 1) Яка комісія повинна контролювати дане рішення.
- 2) Установка класифікатора даного рішення.
- 3) Призначити відповідального за дане рішення.

Комісія – група осіб або орган, що формується у складі групи осіб для виконання яких-небудь певних функцій (виборча комісія, лічильна комісія, ревізійна комісія, приймальна комісія) або проведення спеціальних заходів.

Класифікатор – систематизований перелік найменованих об'єктів, кожному з яких у відповідність надано унікальний код. Класифікація об'єктів проводиться згідно з правилами розподілу заданої множини об'єктів на підмножини (класифікаційні угруповання) відповідно до встановлених ознаками їх відмінності або подібності. Застосовується в автоматизованих системах управління і обробки інформації. Класифікатор є стандартним кодовою мовою документів, фінансових звітів та автоматизованих систем.

У раді існують наступні класифікатори:

- 1) Інвестиційна та зовнішньоекономічна діяльність.
- 2) Соціальний захист
- 3) Безпека
- 4) Охорона навколишнього середовища
- 5) Наука та освіта
- 6) Медицина
- 7) Культура та мистецтво
- 8) Сфера послуг
- 9) Спорт
- 10) Розваги та відпочинок.
- 11) Засоби масової інформації.
- 12) Об'єднання громадян.

Структура Миколаївської облдержадміністрації на яку буде покладено контроль нового рішення зображено на рисунку 2.4.

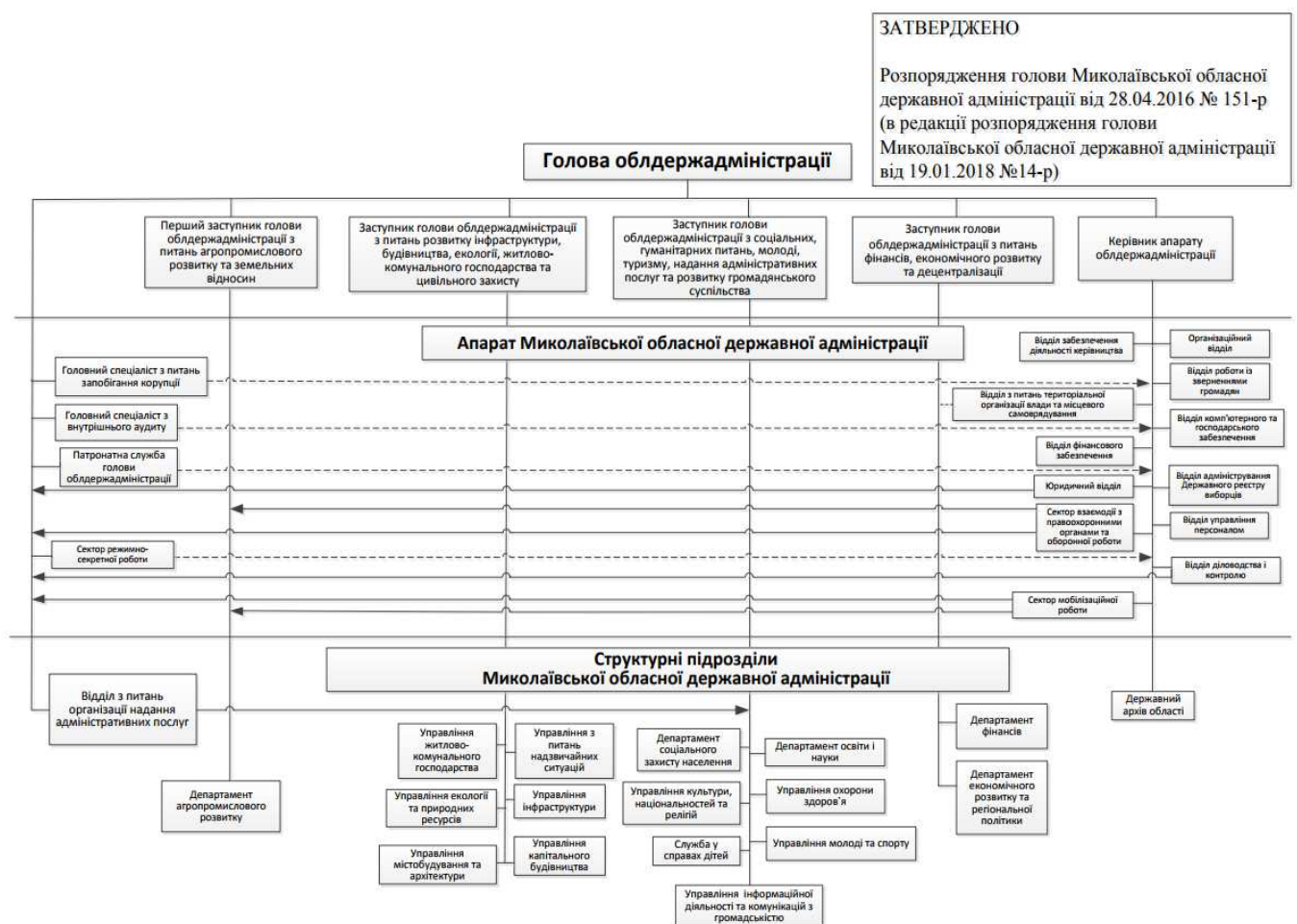


Рисунок 2.4 – Структура Миколаївської облдержадміністрації.

Після всіх стадій підготовки документа який містить нове рішення, починається контроль і виконання даного документа. Документ має юридичну силу і являється постановкою ради. Документ повинен містити в собі дату прийняття рішення, номер сесії на якій прийнято рішення, номер скликання, класифікатор, ПБ відповідального за виконання даного документа, назва комісії відповідальної за виконання даного документа та її аббревіатура. Також рішення може мати пов'язані рішення, в разі чого в документі має бути описано інформація про пов'язане рішення з даної установкою.

2.1.2 Розробка діаграми варіантів використання.

Розробка діаграми варіантів використання переслідує мету:

- 1) Визначити загальні межі і контекст модельованої наочної області на початкових етапах проектування системи.
- 2) Сформулювати загальні вимоги до функціональної поведінки проектованої системи.
- 3) Розробити початкову концептуальну модель системи для її подальшої деталізації у формі логічних і фізичних моделей.
- 4) Підготувати початкову документацію для взаємодії розробників системи з її замовниками і користувачами.

Суть даної діаграми полягає в наступному: проектована система представляється у вигляді множини сутностей або акторів, що взаємодіють з системою за допомогою варіантів використання. При цьому актором або дійовою особою називається будь-яка сутність, що взаємодіє з системою ззовні. У свою чергу, варіант використання служить для опису сервісів, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, ініційованих системою при діалозі з актором. При цьому нічого не говориться про те, яким чином буде реалізована взаємодія акторів з системою.

Основними поняттями діаграм варіантів використання є: діюча особа, варіант використання, зв'язок.

Діюча особа (актор) – зовнішня стосовно розроблювального програмного забезпечення сутність, що взаємодіє з ним з метою одержання або надання якої-

небудь інформації. Діючими особами можуть бути користувачі, інше програмне забезпечення або які-небудь технічні засоби, взаємодіючі з розроблювальним програмним забезпеченням.

Варіант використання – деяка очевидна для діючої особи, процедура, що вирішує його конкретне завдання. Всі варіанти використання, так чи інакше, пов'язані з функціональними вимогами до розроблювальної системи й можуть сильно відрізнятися за обсягом виконуваної роботи.

Зв'язок – взаємодія діючих осіб і відповідних варіантів використання.

Діаграма варіантів використання інформаційної системи зображена на рисунку 2.5.

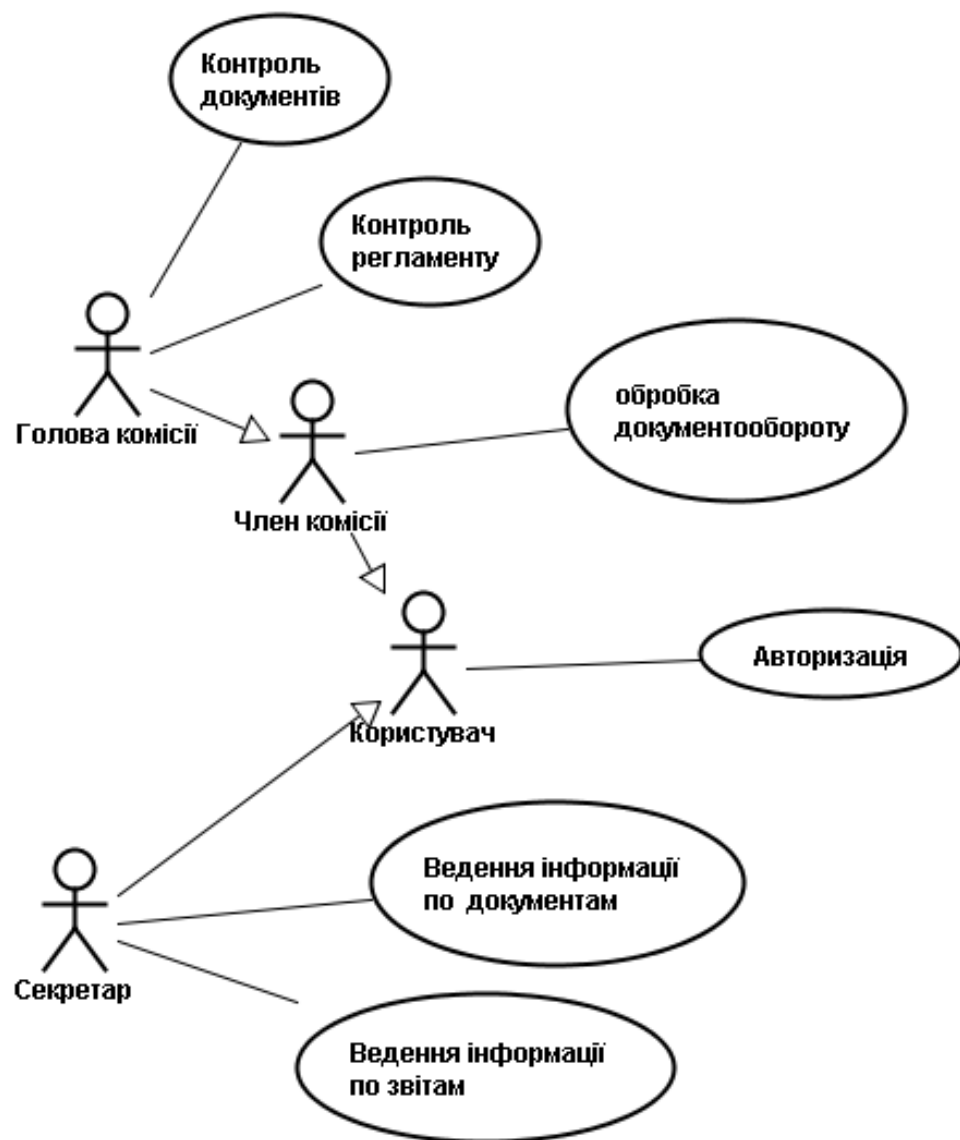


Рисунок 2.5 – Діаграма варіантів використання ІС обробки депутатських запитів.

2.2 Діаграми UML.

2.2.1 Діаграма послідовності.

Діаграма послідовності – діаграма яка відображає взаємодії об'єктів впорядкованих за часом. Зокрема, такі діаграми відображають задіяні об'єкти та послідовність відправлених повідомлень. Основними елементами діаграми послідовності є позначення об'єктів (прямокутники), вертикальні лінії, що відображають протягом часу при діяльності об'єкта, і стрілки, що показують виконання дій об'єктами. На даній діаграмі об'єкти розташовуються зліва направо. Її недоліком є те, що вона займає багато місця. На рисунку 2.6 зображену діаграму послідовності роботи з карткою рішень.

2.2.2 Діаграма переходів стану.

На основі схеми ієрархії класів можливо зробити ієрархію переходів стану яка являє собою, роботу кожного класу окремо та показує у який послідовності працює клас.

Модель класу MainFrame представлено на рисунку 2.7.

Модель класу CardDescision представлено на рисунку 2.8.

Модель класу DatabaseAccess представлено на рисунку 2.9.

Модель класу DatabaseMainFrame представлено на рисунку 2.10.

Модель класу DatabaseCardDescision представлено на рисунку 2.11.

Модель класу HTMLCreator представлено на рисунку 2.12.

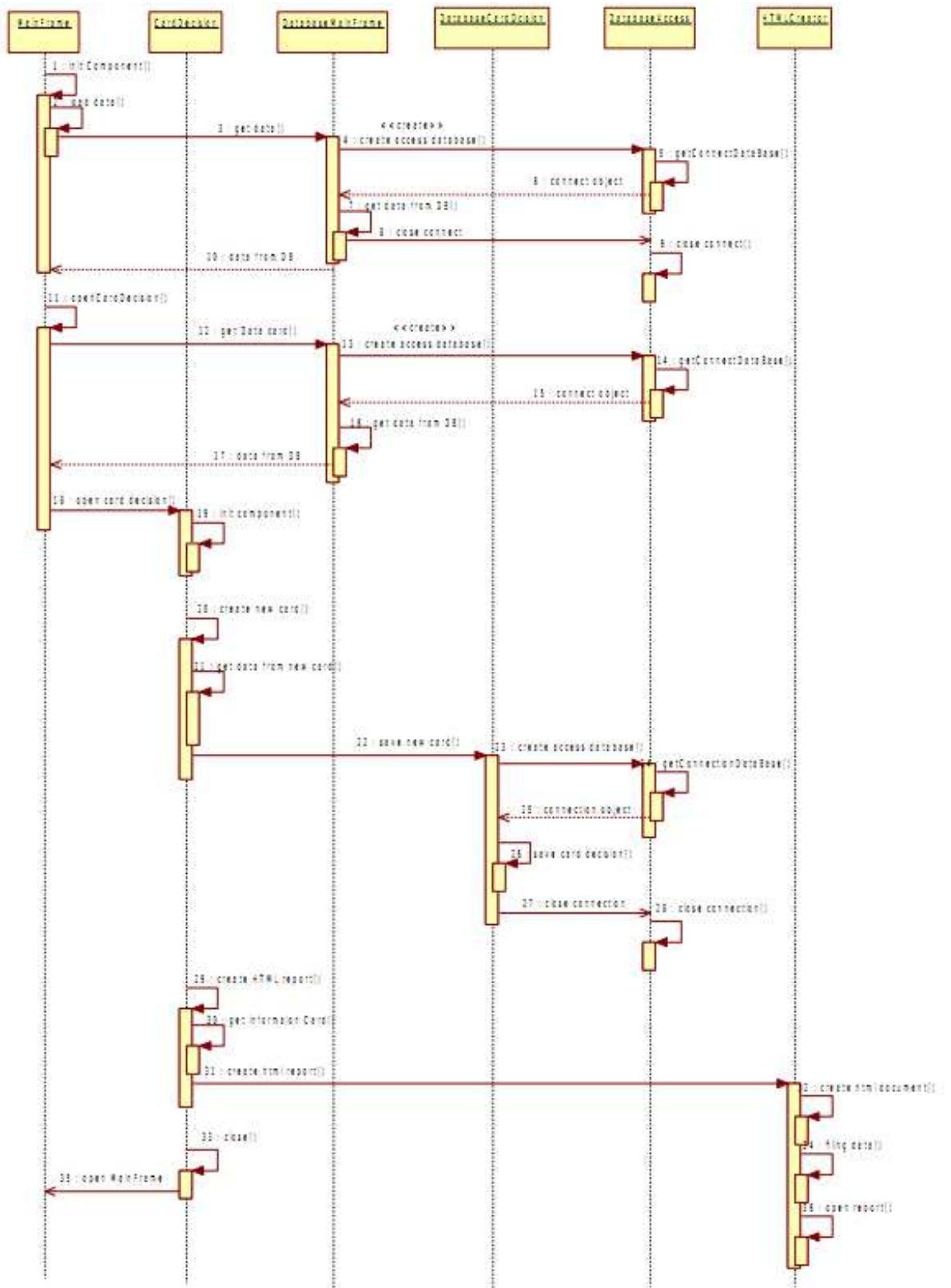


Рисунок 2.6 – Діаграма послідовності.

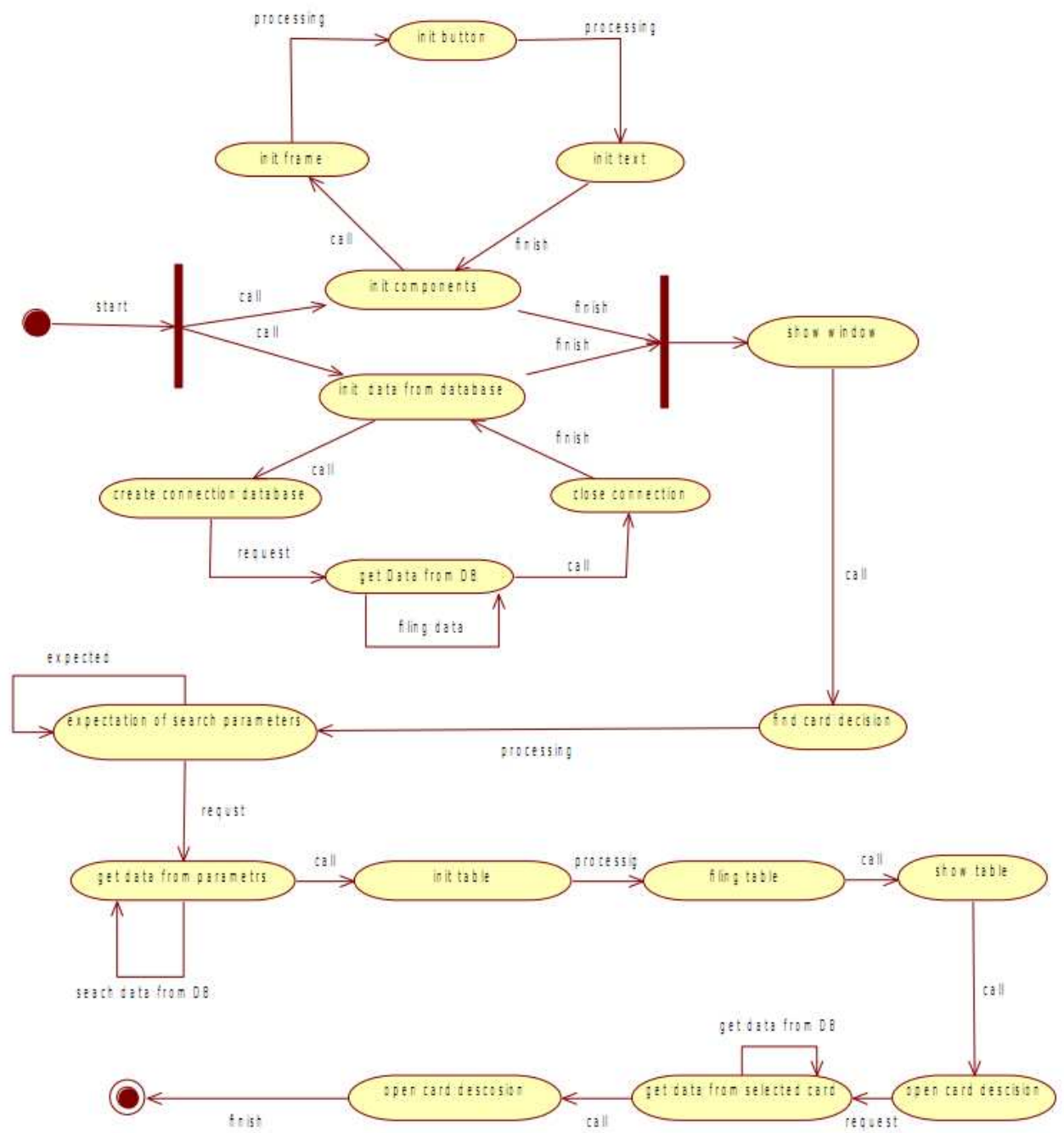


Рисунок 2.7 – Модель класу MainFrame.

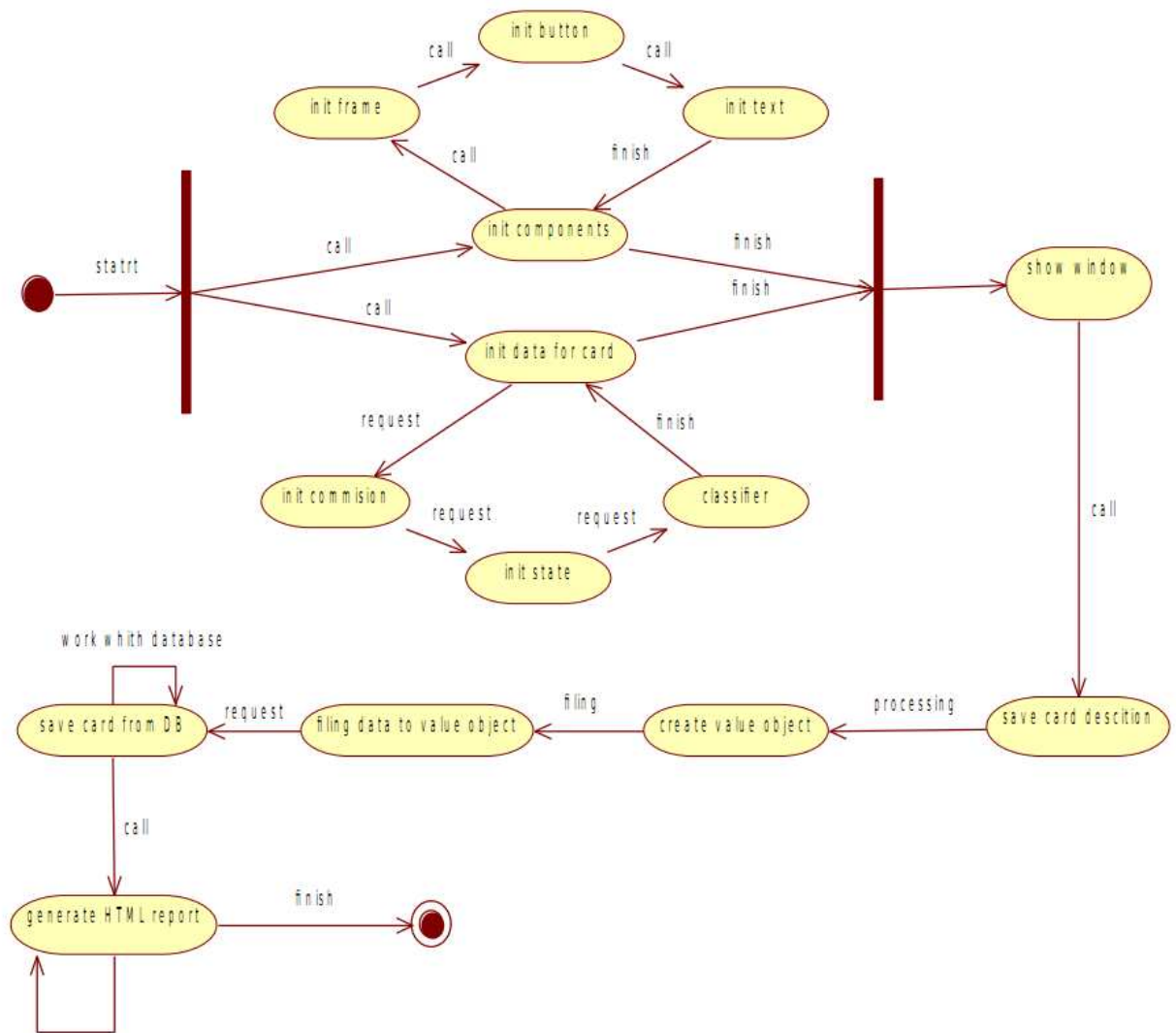


Рисунок 2.8 – Модель класу CardDescision.

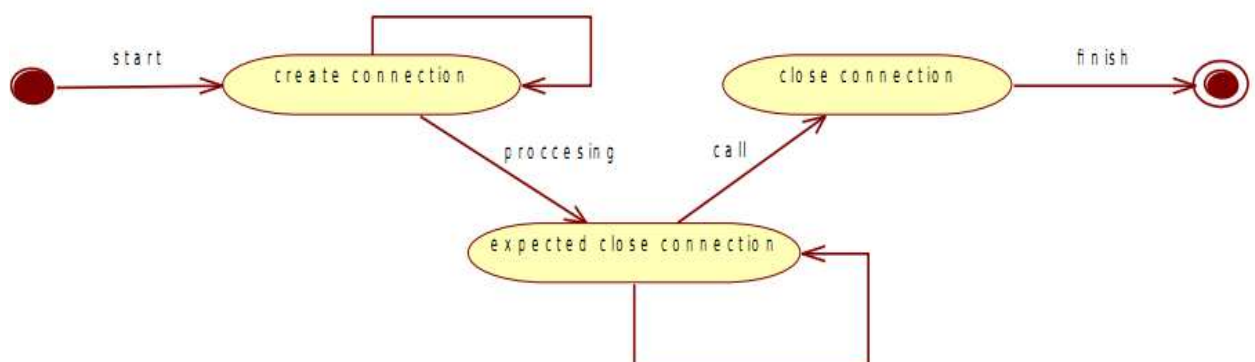


Рисунок 2.9 – Модель класу DatabaseAccess.

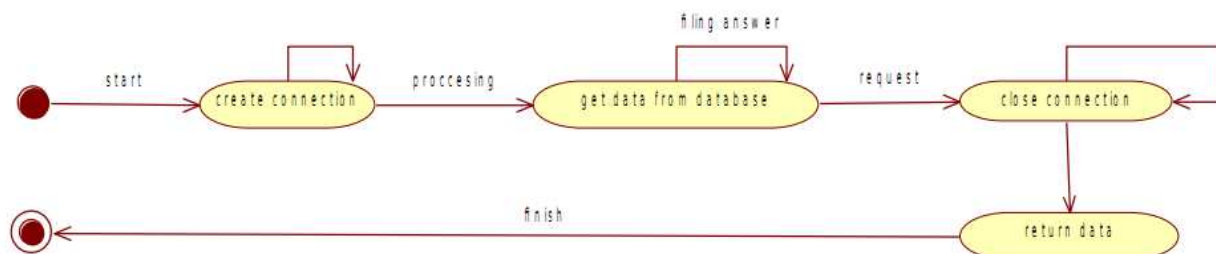


Рисунок 2.10 – Модель класу DatabaseMainFrame.

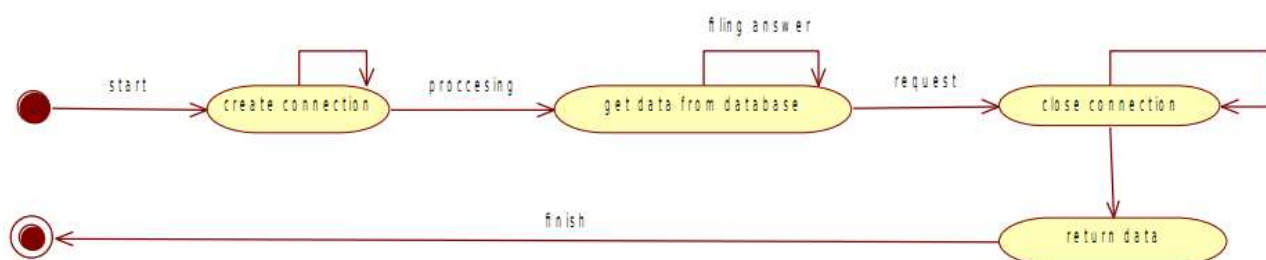


Рисунок 2.11 – Модель класу DatabaseCardDescision.



Рисунок 2.12 – Модель класу HTMLCreator.

2.2.3 Концептуальна модель системи.

Завдяки концептуальній моделі даних в нотації Пітера Чена (ERD) можна дуже зручно та зрозуміло представити сутності, зв'язки між ними та атрибути сутностей.

В нотації Чена сутності зображуються прямокутниками. Сутності також можуть бути слабкими, тобто такими, які не можуть існувати без інших. Вони в даній нотації зображуються прямокутником з подвійною границею. Атрибути сутності мають вигляд еліпсів, поєднаних з даною сутністю простою лінією.

Якщо атрибут ключовий, він підкреслюється двома рисками. Зв'язки в даній нотації позначаються ромбом, на двох кінцях якого пишеться тип зв'язку: 1-1 – якщо один до одного (коли одному примірнику з одної сутності відповідає один примірник з іншої), 1-M – якщо один до багатьох (коли одному примірнику з сутності може відповідати багато примірників з іншої), M-M – якщо багато до багатьох (коли декільком примірникам однієї сутності відповідає декілька примірників іншої).

Для заданої предметної області – інформаційної системи депутатських звернень побудуємо діаграму сутність-зв'язок(ERD), представлену на рисунку 2.13.

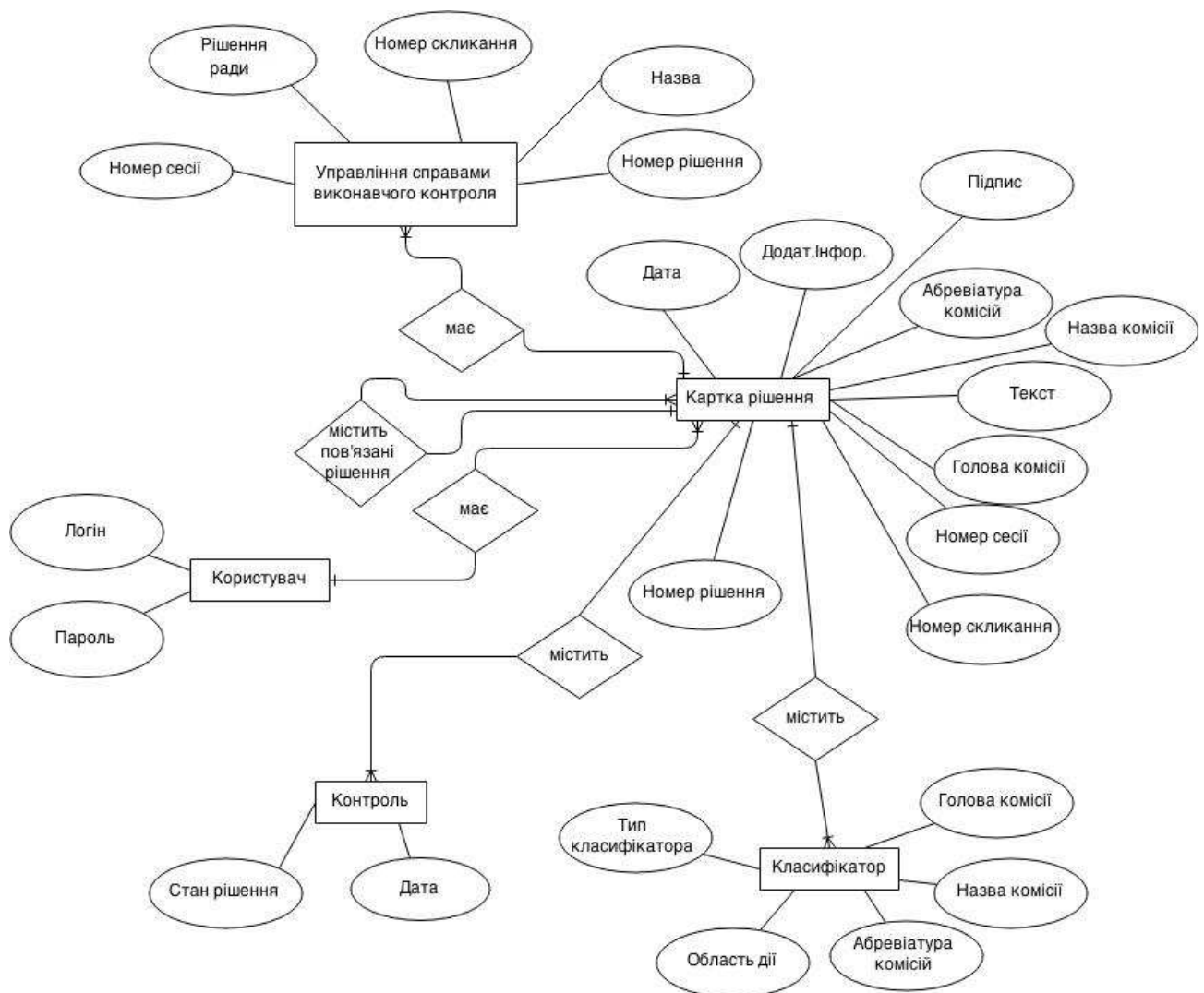


Рисунок 2.13 – Концептуальна модель даних.

2.2.4 Логічна модель даних системи.

Логічна модель даних створюється в термінах СУБД, але є незалежною від СУБД і статичною. На даній моделі вказуються сутності, тип зв'язку між ними, асоціюючи таблиці, що можуть виникнути під час аналізу даної моделі, а також типи даних полів сутностей. В даній моделі сутності позначаються у вигляді прямокутників, в яких вказуються їх поля і типи полів. Сутності також можуть бути слабкими, тобто такими, що не можуть існувати без інших. Типи даних можуть бути п'яти типів: *S* – текст, *N* – число, *D* – тип "дата", *Blob* – великий двійковий об'єкт та *O* – інший.

Дослідивши постановку задачі, потоки інформації, необхідні для функціонування системи можна зробити висновки про структуру і взаємини таблиць, що входять у загальну базу даних. На рисунку 2.14 зображена логічна схема даних, що використовується в системі.

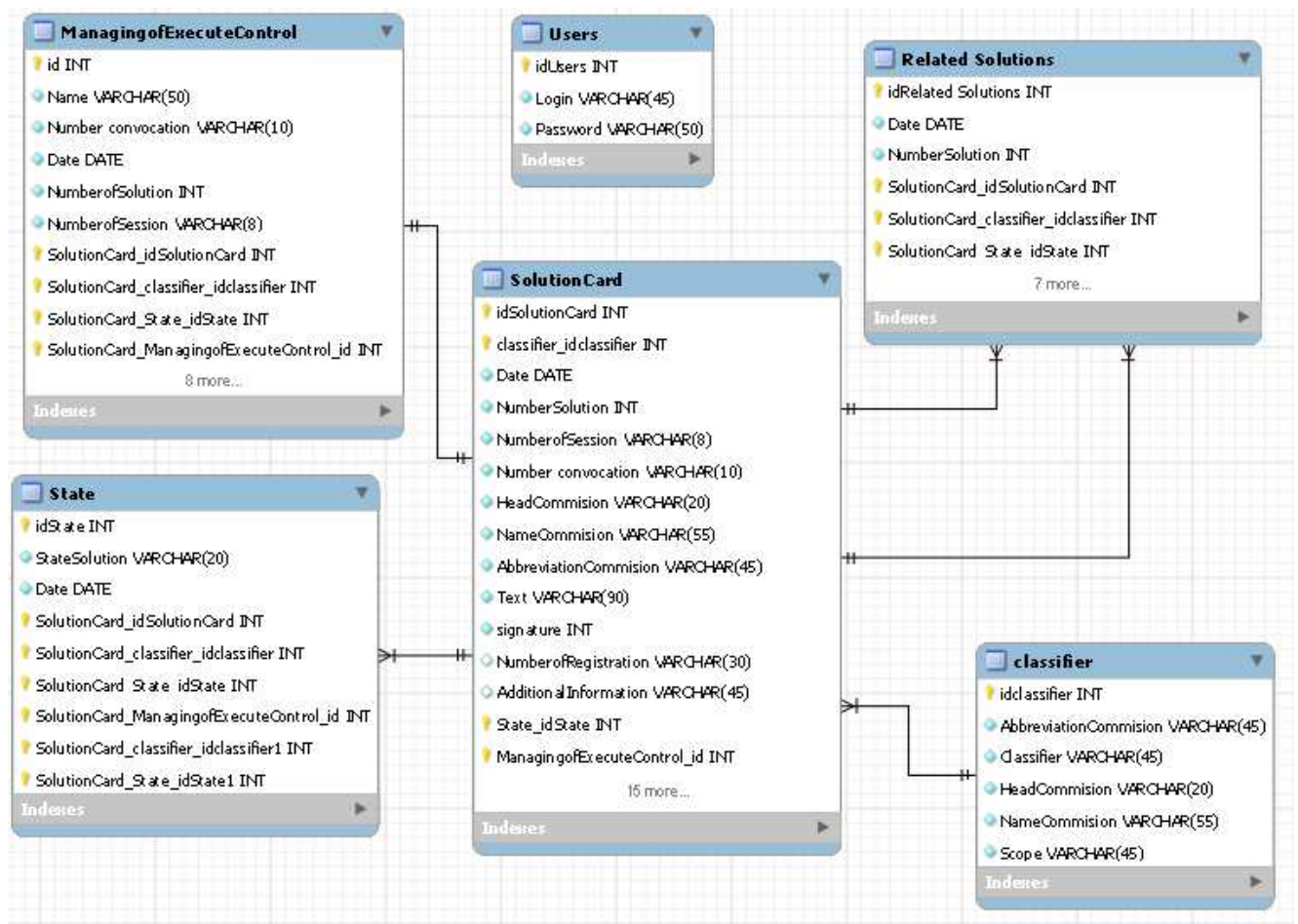


Рисунок 2.14 – Логічна модель даних.

2.3 Рішення з інформаційного забезпечення.

Логічне проектування дозволяє сформувати об'єкти їх атрибути і взаємозв'язки з урахуванням вимог ліквідації надмірності, нормалізації, цілісності й інше, а також з урахуванням вимог прикладної постановки і незалежності даних.

Вивчивши постановку задачі, потоки інформації, необхідні для функціонування системи можна зробити висновки про структуру і взаємини таблиць, що входять у загальну БД.

Зображені наступні сутності:

Рада: скликає сесію та приймає рішення. Має зв'язок з сесією, рішенням, Управляючий справами виконавчого апарату.

Ця сутність має наступні властивості

- Комісії.
- Класифікатори.

Сесія ради: На сесії приймають рішення. Має зв'язок з рішенням.

Ця сутність має наступні властивості:

- Дата.
- Номер сесії.
- Номер рішення.
- Номер скликання.

Управляючий справами виконавчого апарату: виконує контроль документів, приймає класифікатор та виконавчу комісію під кожне рішення. Має зв'язок з карта рішення.

Ця сутність має наступні властивості:

- Комісія.
- Класифікатор.
- Стан.

Карта рішення: виконує функцію сховища для рішення, яка має в собі всі властивості прийнятого рішення. Має зв'язок з рішенням. Ця сутність має наступні властивості

- Дата.

- Номер сесії.
- Номер рішення.
- Номер скликання.
- Назва.
- Класифікатор.
- Стан.
- Пов'язанні рішення.
- Комісія.

Рішення: сутність яка з'являється на сесії ради, яка має юридичну силу та повинна виконуватися за установою.

Ця сутність має наступні властивості

- Назва.
- Призначення.
- Виконавча комісія.

2.4 Рішення з організаційного забезпечення.

Дана система розробляється для Миколаївської обласної ради з метою підвищення ефективності роботи персоналу, ведення операцій з картами рішення. Користувачами системи, що розробляється, будуть: управляючий справами виконавчого апарату. Організаційна структура не змінилась. Були розширені функціональні можливості користувачів системи.

3. РОЗРОБКА РОБОЧОЇ ДОКУМЕНТАЦІЇ.

3.1 Загальний опис системи.

3.1.1 Призначення системи.

Дана автоматизована система призначена для автоматизації робочого місця керуючого справами виконавчого апарату Миколаївської обласної ради.

Метою впровадження системи є повна автоматизація роботи керуючого справами виконавчого апарату Миколаївської обласної ради, що повинне привести до збільшення ефективності роботи.

Розроблена система надає можливість найбільш повного використання можливості комп'ютера для роботи. При розробці було прийняте рішення про створення бази даних системи.

Створена система виконує наступні функції:

- створення і редагування карт рішень;
- забезпечення контролю карт рішень;
- забезпечення спільного використання інформації;
- відтік прибуткових і видаткових рішень;
- формування звітів і відомостей;
- авторизація надійності системи.

3.1.2 Опис системи.

До складу системи входять БД, та ОС Windows подібні або Unix подібні на робочих станціях із клієнтським додатком;

Модуль забезпечує повне управління над існуючими даними такими як комісії, класифікатори, пов'язані рішення. Користувачеві надається можливість створення, редагування і видалення карт рішень, встановлення асоціації між ними, прив'язувати їх до вже існуючих рішень, створювати звіт в електронному вигляді.

Інформація взаємодіє між модулем системи і БД, що здійснюється кожним разом, у залежності, яка необхідна інформація береться зі сховища даних.

Банк даних – сукупність інформації системи, що надходить від усіх модулів,

і забезпечує вільний доступ усім користувачам, що має до нього право доступу. Обмін даними відбувається автоматично з використанням локальної мережі підприємства.

3.2 Методика випробувань системи управління.

Об'єкт випробувань

Об'єктом випробувань є розроблений програмний модуль системи управління документообігом.

Ціль випробувань

Метою проведення випробувань є перевірка відповідності програми технічному завданню.

Вимоги до програми

Вимоги до програми задаються в документі "Технічне завдання".

Функціональні вимоги

- створення і редагування карт рішень;
- забезпечення контролю карт рішень;
- забезпечення спільного використання інформації;
- відтік прибуткових і видаткових рішень;
- формування звітів і відомостей;
- авторизація надійності системі;

Головними вимогами є правильне виконання цих функцій, наочність і надійне зберігання результатів.

3.3 Розробка програми та методики випробувань, порядок здавання та приймання системи.

Автоматизована система повинна бути протестована відповідно до програми і методики випробувань.

Об'єктом тестування є автоматизована системи для, що призначена для керуючого справами виконавчого апарату Миколаївської обласної ради.

Мета тестування – оцінити відповідність програмного забезпечення вимогам, які були сформульовані при постановці завдання, виявити помилки проектування. Якщо достатній рівень відповідності програмного забезпечення був досягнутий, відсутні грубі помилки, то програмне забезпечення приймається до експлуатації.

Для здійснення тестування надається наступні документи:

- технічне завдання (Додаток А);
- проект автоматизації;
- тексти програм (Додаток В);
- описання програм.

У процесі роботи, замовнику надається відповідний повний комплект документів, що обґрунтовує та описує проведення робіт, надається технічне завдання, проектні матеріали по всіх етапах робіт.

Представники замовника встановлюють відповідність проведених робіт технічному завданню, повноту і коректність виконання проектних функцій.

4. РОЗРАХУНОК ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.

4.1 Вступ.

Сучасні комп'ютери, які приймають, запам'ятовують і передають інформацію, здійснюють контроль, являють собою важливий технічний засіб інформаційної системи. Сучасні засоби програмування дозволяють вирішувати задачі, що стояли в нещодавно перед цілими колективами наукових співробітників.

Розвиток нових технологій і ріст застосування обчислювальної техніки привели до широкого застосування ЕОМ для вирішення проблем, пов'язаних з обробкою інформації, її контролем і аналізом. Сьогодні на основі засобів обчислювальної техніки розробляються і впроваджуються різні автоматизовані інформаційні системи (проектування, технологічної підготовки виробництва, електронної демократії, навчання).

Використання даного програмного продукту приведе до значної економії, що виражається в наступних показниках:

- підвищення рівня знань;
- аналіз отриманої інформації;
- економія часу на обробку інформації.

Найбільш важливим моментом для розробника, з економічної точки зору, є процес формування вартості програмного продукту. Очевидно, що вона являє собою дуже специфічний товар з безліччю особливостей.

Рішення з програмного забезпечення системи обробки запитів депутатів обласної ради Миколаївської облдержадміністрації вимагає одноразових витрат на розробку ПЗ, придбання необхідних технічних засобів і поточних витрат на функціонування системи. Відношення цієї економії до витрат на створення системи характеризує економічну ефективність капітальних вкладень. Економічні показники визначаються по діючим на момент розрахунку оптовим цінам, тарифам і ставкам заробітної плати.

4.2 Розрахунок витрат на створення та експлуатацію ПЗ інформаційної системи.

Витрати на розробку ПЗ системи складаються з витрат на зарплату розробника, на амортизацію ЕОМ, на якій виконується розробка, на експлуатацію цієї ЕОМ, на засоби розробки та витрат на матеріали і комплектуючі.

Припустимо розробка програмного забезпечення виконується програмістом, місячний оклад якого складає 15000 грн. Додаткова заробітна плата складає 20 % від основної. Виходячи з цього, основна і додаткова заробітна плата розроблювача системи 18000 грн./міс., а вартість сучасної ПК складає 24000 грн. (приведена вартість машини на базі Intel Core i5: Intel Core i5-7400 (3.0-3.5 ГГц)/RAM 16 ГБ/HDD 1 ТБ/nVidia GeForce GTX1050 Ti, 4 ГБ/без ОД/LAN/без ОС. При вартості кіловат-години електроенергії рівної 4,32 грн. (для населення, яке споживає понад 100 кВт·год до 600 кВт·год електроенергії на місяць), розраховується вартість розробки програми.

Витрати на допоміжні матеріали приведені в табл. 4.1.

Таблиця 4.1 – Витрати на допоміжні матеріали

Пункти витрат	Сума грн.
Папір	100,00
Заправлення картриджа до принтера	120,00
Канцтовари	180,00
Література	2000,00
Непередбачені витрати	500,00
Разом:	2800,00

Вартість програми розрахуємо по формулі:

$$C_{np} = (Z_{zn} + Z_{cz} + Z_{ze} + Z_e) \cdot T + Z_m, \quad (4.1)$$

де T – тривалість розробки, міс.

Z_{zn} – основна і додаткова заробітна плата обслуговуючого персоналу, грн.

Z_{cz} – відрахування на соціальні заходи (38 % від основної і додаткової заробітної плати), грн.

Z_{ze} – загальногосподарські витрати (10 % від основної заробітної плати),

грн.

Z_e – витрати на електроенергію.

Z_m – витрати на основні і допоміжні матеріали.

Z_e при споживанні потужності 0.5 кВт, тривалості роботи на місяць, рівної $22 \cdot 6 = 132$ години, вартості кіловат-години електроенергії 4,32 грн обчислимо:

$$Z_e = 132 \cdot 4,32 \cdot 0,5 = 285,12 \text{ грн.}$$

Таблиця 4.2. – Витрати на розробку системи.

Найменування витрат	Одиниця виміру	Кількість
Тривалість розробки	Місяць	1
Основна і додаткова заробітна плата	Гривня	18000
Відрахування на соціальні заходи	Гривня	6840
Загальногосподарські витрати	Гривня	1500
Витрати на допоміжні матеріали	Гривня	2800
Витрати на електроенергію	Гривня	110,88

Відповідно до формули 4.1 вартість програми:

$$C_{np} = (18000 + 6840 + 1500 + 285,12) \cdot 1 + 2800 = 29250,88 \text{ грн.}$$

Амортизаційні відрахування на устаткування складають 60 % балансової вартості в рік:

$$A_{об} = 24000 \cdot 0,6 = 14400 \text{ грн.}$$

У масштабах підприємства річні витрати на основні і допоміжні матеріали (носії інформації, папір) визначаються в розмірі 5 % вартості основного устаткування:

$$B_m = 24000 \cdot 0,06 = 1200 \text{ грн.}$$

Річний обсяг робіт ПК у годинах визначається в такий спосіб:

$$\Phi_m = 264,5 \cdot T_z, \quad (4.2)$$

де T_z – середнє місячне завантаження устаткування (близько 4 годин);

264,5 – середня кількість робочих днів у році.

Отже, річний обсяг роботи ПЕОМ складе:

$$\Phi_m = 264,5 \cdot 4 = 1058 \text{ годин.}$$

Витрати на електроенергію Z_e при 1058 годинах роботи устаткування в рік складуть

$$Z_e = 1058 \cdot 1,68 \cdot 0,5 = 888,72 \text{ грн.}$$

Експлуатаційні витрати для ПК за рік складуть:

$$Z_{zp} = A_{об} + B_{м} + \Phi_{м} = 14400 + 1200 + 888,72 = 16488,72 \text{ грн.}$$

Отже, у перший рік витрати на створення й експлуатацію програми складуть:

$$Z_{се} = C_{np} + Z_{zp} = 29250,88 + 16488,72 = 45739,60 \text{ грн.}$$

4.3 Економічна ефективність розробки і впровадження системи.

Основним показником економічної ефективності функціонування системи є підвищення ефективності керування інформацією у вигляді зниження витрат на керування при одночасному збільшенні швидкості і якості одержання потрібного результату.

Крім багатьох інших негативних ефектів, ручна обробка інформації спричиняє наступні негативні економічні ефекти:

- високі витрати на складування паперових документів (сейфи, шафи, папки й ін.);
- підвищені витрати на канцтовари;
- витрати, пов'язані з роботою виявлення раніше допущених помилок (людський фактор).

До числа основних факторів, що визначають приріст прибутку в зв'язку з упровадженням програми, відносяться:

- підвищення продуктивності праці;
- вивільнення робочого часу.

Крім того, не піддається прямій грошовій оцінці підвищення оперативності керування, якість одержуваних результатів, поліпшення організації праці і т.д.

Обов'язковою умовою визначення економічної ефективності програми є порівнянність усіх показників у часі, за цінами й іншими нормами, використовуваними для визначення показників, за змістом і колом елементів витрат.

Визначимо пряму економічну ефективність, ґрунтуючись на тому, що

впровадження програмного продукту вивільняє 0,7 працівника (за експертною оцінкою фахівців підприємства).

Зарплата 0,7 працівника в рік складає:

$$15000 \cdot 12 \cdot 0,7 = 126000 \text{ грн.}$$

Річний економічний ефект розраховується по формулі:

$$\mathcal{E}_{\text{год}} = \Delta C_n - E_n \cdot k,$$

де ΔC_n – вивільнені кошти після впровадження програмного продукту мінус експлуатаційні витрати: $\Delta C_n = 126000 - 16488,72 = 109511,3$ грн.;

E_n – коефіцієнт ефективності (дорівнює коефіцієнту амортизації – 0,6);

k – одноразові витрати на впровадження продукту (45739,60).

$$\mathcal{E}_{\text{год}} = 109511,3 - 0,6 \cdot 45739,60 = 82067,52 \text{ грн.}$$

Строк окупності системи розраховується по формулі:

$$T = \frac{k}{\Delta C} = \frac{45739,60}{109511,3} \approx 0,42 \text{ років.}$$

Отже строк окупності системи складає приблизно 5 місяців.

5. ОХОРОНА ПРАЦІ.

5.1 Загальні визначення та актуальність питання охорони праці.

Охорона праці є системою правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних та лікувально-профілактичних заходів та засобів, спрямованих на збереження здоров'я і працездатності людей в процесі праці.

Як відомо, цілком безпечних і нешкідливих виробництв не існує, тому основною задачею охорони праці є зведення до мінімуму імовірність ураження чи захворювання працюючого з одночасним забезпеченням комфорту при максимальній продуктивності праці. Основний принцип у галузі охорони праці – це пріоритет життя і здоров'я працівника по відношенню до результатів виробничої діяльності. При цьому роботодавці зобов'язані забезпечувати здорові і безпечні умови праці для своїх працівників, вживати всіх заходів щодо запобігання ризиків на робочих місцях, навчати працівників правилам і нормам охорони праці і виконувати інші вимоги законодавства.

Турбота про охорону праці є справою державної важливості. 14 жовтня 1992 року був прийнятий Закон України "Про охорону праці", у якому закріплені найважливіші положення в області охорони праці, і який поширюється на всіх суб'єктів підприємницької діяльності незалежно від форм власності.

Стаття 4 Закону "Про охорону праці" формулює принципи державної політики у сфері охорони праці. Ці принципи в силу вказаного їх визначення, слід кваліфікувати і як принципи інституту охорони праці, які підлягають застосуванню при відсутності відповідних конкретних правових норм, і як такі, що визначають напрямки подальшого розвитку інституту охорони праці, і обов'язки держави, які вона взяла на себе і має додержувати при здійсненні своїй політики у майбутньому. З урахуванням цього, принципи зазначені статтею 4 Закону "Про охорону праці" мають певне регулятивне значення. В статті закріплені такі принципи:

- пріоритет життя і здоров'я працівників відносно результатів виробничої

діяльності підприємства, повна відповідальність власника за створення безпечних і нешкідливих умов праці;

- комплексне вирішення завдань охорони праці на засадах національних програм з цих питань, а також з урахуванням інших напрямків економічної і соціальної політики, досягнень в галузі науки, техніки охорони здоров'я;

- соціальний захист працівників, повне відшкодування шкоди особам потерпілим від нещасних випадків на виробництві та професійних захворювань;

- установлення єдиних нормативів з охорони праці для всіх підприємств незалежно від форм власності і видів їх діяльності;

- використання економічних методів управління охороною праці. проведення політики пільгового оподаткування, то сприяє створенню безпечних та нешкідливих умов праці, участь держави в фінансуванні заходів з охорони праці;

- здійснення навчання населення, професійної підготовки та підвищення кваліфікації працівників з питань охорони праці;

- забезпечення координації діяльності державних органів, установ, організацій та об'єднань громадян, що вирішують різні проблеми охорони здоров'я, гігієни та безпеки праці, а також співробітництво.

Державне керування органами охорони праці здійснюють: Кабінет Міністрів України, державний комітет України по нагляду за охороною праці, міністерство й інші централізовані органи, місцеві ради народних депутатів і їхні виконавчі комітети. Державний нагляд за дотриманням законодавства й інших нормативних актів по охороні праці здійснюють: державний комітет України по нагляду за охороною праці, державний комітет України по атомній і радіаційній безпеці, органи державного пожежного нагляду, органи і служби санітарно епідемічного нагляду. А трудові колективи в особі своїх уповноважених і профспілки здійснюють суспільний контроль за охороною праці.

Вищий нагляд за точним виконанням законів про працю міністерствами і відомствами, підприємствами, установами, громадянами і посадовими особами

покладений на Генерального Прокурора України.

Велике значення в створенні здорових і безпечних умов праці має стандартизація. Вона дозволяє вжити діючих заходів по підвищенню технічного рівня й упорядкуванню розробки нормативно-технічної документації по безпеці праці. З 1972 року створюється Система стандартів по безпеці праці (ССБП), що являє собою комплекс великого числа взаємозалежних стандартів, спрямованих на забезпечення безпеки праці. Ця система встановлює загальні вимоги і норми по видах небезпечних і шкідливих виробничих факторів, загальні вимоги безпеки до виробничого устаткування і процесів, вимоги до засобів захисту працюючих, методи оцінки безпеки праці.

Охорона праці – один з центральних інститутів трудового права. Він має виключно практичне значення. Недодержання вимог охорони праці створює небезпеку для здоров'я і життя працівників. У свою чергу і ті, кого законодавець називає власником або уповноваженим ним органом, несуть сувору, у тому числі й кримінальну, відповідальність за порушення правил охорони праці.

Однією із характерних особливостей сучасного розвитку суспільства є зростання сфер діяльності людини, в яких використовуються інформаційні технології. Широке розповсюдження отримали персональні комп'ютери. Однак їх використання загостило проблеми збереження власного та суспільного здоров'я, вимагає удосконалення існуючих та розробки нових підходів до організації робочих місць, проведення профілактичних заходів для запобігання розвитку негативних наслідків впливу ПК на здоров'я користувачів. Тому питання охорони праці при роботі з ПК є досить актуальним.

5.2 Аналіз небезпечних і шкідливих факторів при роботі з ПК.

З точки зору впливу небезпечних і шкідливих факторів при роботі з ПК можна виділити наступні:

- вплив електромагнітного та електростатичного випромінювань;
- вплив недостатньої освітленості робочого місця;
- вплив електричного струму;
- вплив шуму;

- вплив несприятливих умов мікроклімату;
- вплив нераціонального розташування устаткування і неправильної організації робочого місця.

Тривале перебування людини в зоні комбінованого впливу різних несприятливих факторів може призвести до професійного захворювання. Розглянемо більш детально вплив означених факторів на користувача ПК.

1) Вплив електромагнітного та електростатичного випромінювань.

При роботі ПК виникають два типи випромінювань: електростатичне та електромагнітне. Навколо електростатично-зарядженого монітора підвищується концентрація пилу. Електромагнітне випромінювання створюється магнітними котушками. Результати медичних досліджень показують, що електризований пил викликає запалення шкіри, а електромагнітне випромінювання приводить до порушення біологічних процесів (порушення сну, зниження кров'яного тиску, уповільнення серцевої діяльності, зміни складу крові).

2) Вплив недостатньої освітленості робочого місця.

Напружена зорова робота внаслідок нераціонального освітлення може бути причиною функціональних порушень у зоровому аналізаторі і привести до розладу зору, а у важких випадках – і до повної втрати. Втома органів зору залежить від ступеня напруженості процесів, що супроводжують зорове сприйняття. При роботі оператора ПК одним з основних навантажень на організм є навантаження на зір, тому дуже важливо приділити особливу увагу освітленості робочого місця.

У залежності від джерела світла виробниче висвітлення може бути трьох видів: природним, штучним і сполученим. При роботі з ПК вдень у ясну погоду використовується природне освітлення, яке здійснюється через світлові віконні прорізи. При нестачі природного світла, а також при роботі у вечірній час використовується штучне освітлення, джерелами якого є газорозрядні лампи. При такому освітленні величезне значення має правильне розташування джерел світла, їхній тип і кількість.

Правильна освітленість робочого місця сприяє високій працездатності користувача ПК і зводить до мінімуму шкідливий вплив на органи зору.

Освітленість робочого місця, що рекомендується для користувача ПК складає 400–750 лк.

3) Вплив електричного струму.

Робота комп'ютерів і периферійних пристроїв можлива завдяки електричному струмові, до джерел якого вони підключаються, тому існує небезпека ураження користувача струмом. Щоб уникнути нещасних випадків варто строго дотримуватися правил електробезпечності.

Небезпека електричного струму на відміну від інших небезпек збільшується за рахунок того, що людина не в змозі без спеціальних приладів виявити напругу дистанційно, а також швидкоплинності поразки – небезпека виявляється, коли людина вже уражена. Ураження струмом приводить до різних порушень в організмі, викликаючи як місцеву поразку тканин і органів, так і загальне ураження організму. Людина відчуває дратівну дію перемінного струму промислової частоти силою 0,6-1,5 мА і постійного струму 5-7 мА. Ці струми не представляють серйозної небезпеки для організму людини, тому що при такій силі струму можливе самостійне звільнення людини від контакту зі струмоведучими частинами. Для перемінного струму промислової частоти сила невідпускаючого струму знаходиться в межах 6-20 мА. Постійний струм не викликає невідпускаючого ефекту, але приводить до сильних болючих відчуттів, сила такого струму 15-80 мА і більше.

4) Вплив шуму.

Шумом прийнято називати всякий небажаний для людини звук, що заважає сприйняттю корисних сигналів. Шум являє собою безладне сполучення звуків різної інтенсивності і частоти. Він викликає труднощі з розпізнанням колірних сигналів, знижує швидкість сприйняття кольору, гостроту зору, зорову адаптацію, порушує сприйняття візуальної інформації, зменшує на 5-12% продуктивність праці. Тривалий вплив шуму з рівнем звукового тиску 90 дБ знижує продуктивність праці на 30-60%.

Робота ПК може супроводжуватися шумами, викликаними деякими рухливими пристроями усередині комп'ютера, а також підключеними до нього. До таких пристроїв можна віднести вентилятори, пристрої читання дисків,

принтери. Виникаючий шум, у залежності від інтенсивності, може привести до підвищення нервової напруги і зниження працездатності.

5) Вплив несприятливих умов мікроклімату.

Мікрокліматичні параметри (температура, відносна вологість і швидкість руху повітря) впливають на самопочуття і здоров'я людини, а також на надійність роботи засобів обчислювальної техніки.

При підвищеній температурі і вологості в приміщенні відбувається перегрівання тіла, що погіршує самопочуття і працездатність. Варто враховувати, що ПК, які знаходяться в приміщенні, сприяють збільшенню температури.

Для комфортної роботи приймається температура $23 (\pm 2)^{\circ}\text{C}$ й $18 (\pm 2)^{\circ}\text{C}$ в теплий і холодний час року відповідно, при відносній вологості $55 (\pm 5)\%$.

6) Вплив нераціонального розташування устаткування і неправильної організації робочого місця.

Робоче місце – це частина простору, у якій користувач ПК здійснює трудову діяльність і проводить велику частину робочого часу. Важливу роль грає планування робочого місця, яке повинне задовольняти вимогам зручності виконання робіт і економії енергії і часу користувача, зручності обслуговувань ЕОМ, дотримання правил охорони праці.

При плануванні робочого місця необхідно враховувати зони досяжності при розташуванні моніторів, клавіатури, маніпулятора "миша". Ці зони встановлюються на підставі антропологічних даних людського тіла. Під час роботи користувач ПК не повинен піддаватися впливові сторони подразників, якими можуть бути неадекватне фарбування пристроїв ЕОМ або приміщення.

5.3 Розрахунок системи загального освітлення приміщення з ПК.

Рациональне освітлення приміщень – один з найбільш важливих факторів, від яких залежить ефективність трудової діяльності людини. Гарне освітлення необхідне для виконання більшості задач.

До систем освітлення висувають наступні вимоги:

– відповідність рівня освітленості робочих місць характеру виконуваної

зорової роботи;

- досить рівномірний розподіл яскравості на робочих поверхнях і в навколишньому просторі;
- сталість освітленості в часі;
- оптимальна спрямованість випромінюваного освітлювальними приладами світлового потоку;
- довговічність, економічність, електро- та пожежабезпечність, естетичність, зручність і простота експлуатації.

В приміщенні з ПК необхідно організувати змішане освітлення (комбінування природного і штучного освітлення). Природне освітлення створюється бічним освітленням через вікно. Штучне освітлення використовується при недостатньому природному освітленні. У даному приміщенні використовується загальне штучне освітлення. Розрахунок виконаємо за методом світлового потоку з урахування потоку, віддзеркаленого від стін та стелі.

Необхідно зробити розрахунок загального освітлення виробничої приміщення. Довжина $A = 10$ м, ширина $B = 7$ м, висота $H = 3$ м. Висота робочої поверхні $h_p = 1$ м. Для освітлення використовується світильні УПМ-15. Мінімальна освітленість лампи наколювання по нормах $E_{min} = 100$ лк. Коефіцієнт відображення стелі $S_n = 50\%$, стін $S_c = 30\%$, робочої поверхні $S_p = 10\%$. Напруга мережі 220 В.

Відстань від стелі до робочої поверхні:

$$H_0 = H - h_p = 3 - 1 = 2 \text{ м.}$$

Відстань від стелі до світильника:

$$h_c = 0,2 \cdot H_0 = 0,2 \cdot 2 = 0,4 \text{ м.}$$

Висота підвісу світильника над освітленою поверхнею:

$$h = H_0 - h_c = 2 - 0,4 = 1,6 \text{ м.}$$

Висота підвісу світильника над підлогою:

$$H_n = h + h_p = 1,6 + 1 = 2,6 \text{ м.}$$

Для досягнення найбільшої рівномірності освітлення приймаємо

відношення $L/h = 1,5$.

Відстань між центрами світильників:

$$L = 1,5 \cdot h = 1,5 \cdot 1,6 = 2,4 \text{ м.}$$

Необхідна кількість світильників:

$$N = S/L^2 = (10 \cdot 7) / (2,4)^2 \approx 12.$$

Приймаємо кількість світильників: 12 (3 ряди по 4 світильники).

Індекс приміщення:

$$i = (A \cdot B) / (h \cdot (A + B)) = (10 \cdot 7) / (1,6 \cdot (10 + 7)) = 2,57$$

(найближче значення в таблиці 2,5).

По таблиці коефіцієнтів використання світлового потоку при $i = 2,5$, $S_n = 50\%$, $S_c = 30\%$, $S_p = 10\%$ для світильника типу "УПМ-15" коефіцієнт використання світлового потоку $\eta = 0,59$.

Світловий потік однієї лампи:

$$\Phi = (E_{min} \cdot S \cdot K_z \cdot Z) / (N \cdot \eta) = (100 \cdot 70 \cdot 1,5 \cdot 1,1) / (12 \cdot 0,59) = 1631 \text{ лм,}$$

де K_z – коефіцієнт запасу, який враховує запилення світильників і знос джерел світла в процесі експлуатації; для приміщень, освітлюваних люмінесцентними лампами $K_z = 1,5$ за умови чищення світильників не рідше двох разів на рік;

Z – коефіцієнт нерівномірності освітлення ($Z = 1,1$);

За знайденим світовим потоком із таблиці "Параметри ламп накаливання загального призначення з розрахунковими напругами 130 і 220 В" обираємо лампу Г 215-225-150 накаливання потужністю 150 Вт, яка має світовий потік 2090 лк, що найбільше близький до розрахункового.

Фактична освітленість:

$$E_{\phi} = E_{min} \cdot \Phi_{\lambda} / \Phi_p = 100 \cdot 2090 / 1631 = 128,14 \text{ лк.}$$

Загальна потужність:

$$P_{заг} = P_{л} \cdot N = 150 \cdot 12 = 1800 \text{ Вт} = 1,8 \text{ кВт.}$$

Таким чином, для оптимальної роботи в приміщенні необхідно встановити 3 ряди по 4 світильники типу Г 215-225-150. Загальна потужність усіх світильників складе 1,8 кВт.

5.4 Розробка заходів щодо забезпечення сприятливих умов праці при роботі з персональним комп'ютером.

При виконанні робіт на комп'ютерах необхідно дотримуватись вимог загальної та даної інструкції з охорони праці.

До самостійної роботи на комп'ютерах допускаються особи, які пройшли медичний огляд, навчання по професії, вступний інструктаж з охорони праці та первинний інструктаж з охорони праці на робочому місці. В подальшому вони проходять повторні інструктажі з охорони праці на робочому місці один раз на півріччя, періодичні медичні огляди один раз на два роки.

Під час роботи на комп'ютерах можуть діяти такі небезпечні та шкідливі фактори, як:

- фізичні;
- психофізіологічні.

Основним обладнанням робочого місця користувача комп'ютера є монітор, системний блок та клавіатура.

Робочі місця мають бути розташовані на відстані не менше 1,5 м від стіни з вікнами, від інших стін на відстані 1 м, між собою на відстані не менше 1,5 м. Відносно вікон робоче місце доцільно розташовувати таким чином, щоб природне світло падало на нього збоку, переважно зліва.

Робочі місця слід розташовувати так, щоб уникнути попадання в очі прямого світла. Джерела освітлення рекомендується розташовувати з обох боків екрану паралельно напрямку погляду. Для уникнення світлових відблисків екрану, клавіатури в напрямку очей користувача, від світильників загального освітлення або сонячних променів, необхідно використовувати антиполюсові сітки, спеціальні фільтри для екранів, захисні козирки, на вікнах – жалюзі.

Фільтри з металевої або нейлонової сітки використовувати не рекомендується, тому що сітка спотворює зображення через інтерференцію світла. Найкращу якість зображення забезпечують скляні поляризаційні фільтри. Вони усувають практично всі відблиски, роблять зображення чітким і контрастним.

При роботі з текстовою інформацією (в режимі введення даних та редагування тексту, читання з екрану) найбільш фізіологічним правильним є зображення чорних знаків на світлому (чорному) фоні.

Монітор повинен бути розташований на робочому місці так, щоб поверхня екрана знаходилася в центрі поля зору на відстані 400-700 мм від очей користувача. Рекомендується розміщувати елементи робочого місця так, щоб витримувалася однакова відстань очей від екрана, клавіатури, тексту.

Зручна робоча поза при роботі з комп'ютером забезпечується регулюванням висоти робочого столу, крісла та підставки для ніг. Раціональною робочою позою може вважатися таке положення, при якому ступні працівника розташовані горизонтально на підлозі або підставці для ніг, стегна зорієнтовані у горизонтальній площині, верхні частини рук – вертикальні. Кут ліктьового суглоба коливається в межах 70-90°, зап'ястя зігнуті під кутом не більше ніж 20°, нахил голови 15-20°.

Для нейтралізації зарядів статичної електрики в приміщенні, де виконується робота на комп'ютерах, в тому числі на лазерних та світлодіодних принтерах, рекомендується збільшувати вологість повітря за допомогою кімнатних зволожувачів. Не рекомендується носити одяг з синтетичних матеріалів.

Вимоги безпеки перед початком роботи:

- увімкнути систему кондиціонування в приміщенні;
- перевірити надійність встановлення апаратури на робочому столі. Повернути монітор так, щоб було зручно дивитися на екран – під прямим кутом (а не збоку) і трохи зверху вниз, при цьому екран має бути трохи нахиленим, нижній його край ближче до оператора;
- перевірити загальний стан апаратури, перевірити справність електропроводки, з'єднувальних шнурів, штепсельних вилок, розеток, заземлення захисного екрана;
- відрегулювати освітленість робочого місця;
- відрегулювати та зафіксувати висоту крісла, зручний для користувача

нахил його спинки;

- приєднати до системного блоку необхідну апаратуру. Усі кабелі, що з'єднують системний блок з іншими пристроями, слід вставляти та виймати при вимкненому комп'ютері;

- ввімкнути апаратуру комп'ютера вимикачами на корпусах в послідовності: монітор, системний блок, принтер (якщо передбачається друкування);

- відрегулювати яскравість свічення монітора, мінімальний розмір світної точки, фокусування, контрастність. Не слід робити зображення надто яскравим, щоб не втомлювати очей.

Рекомендується:

- яскравість свічення екрана – не менше 100 кг/м²;
- відношення яскравості монітора до яскравості оточуючих його поверхонь в робочій зоні – не більше 3:1;

- мінімальний розмір точки свічення не більше 0,4 мм для монохромного монітора і не менше 0,6 мм для кольорового, контрастність зображення знаку – не менше 0,8.

При виявленні будь-яких несправностей роботу не розпочинати, повідомити про це керівника.

Вимоги безпеки під час виконання роботи:

- необхідно стійко розташовувати клавіатуру на робочому столі, не опускати її хитання. Під час роботи на клавіатурі сидіти прямо, не напружуватися;

- для забезпечення несприятливого впливу на користувача пристроїв типу "миша" належить забезпечувати вільну велику поверхню столу для переміщення "миші" і зручного упору ліктьового суглоба;

- не дозволяються по сторонні розмови, подразнюючі шуми;

- періодично при вимкненому комп'ютері прибирати ледь змоченою мильним розчином бавовняною ганчіркою порох з поверхонь апаратури. Екран ВДТ та захисний екран протирають ганчіркою, змоченою у спирті. Не

дозволяється використовувати рідинні або аерозольні засоби чищення поверхонь комп'ютера.

Забороняється:

- самостійно ремонтувати апаратуру. Ремонт апаратури здійснюється спеціалістами з технічного обслуговування комп'ютера;
- класти будь-яку предмети на апаратуру комп'ютера;
- закривати будь-чим вентиляційні отвори апаратури, що може призвести до її перегрівання і виходу з ладу.

Для зняття статичної електрики рекомендується час від часу доторкатися до металевих поверхонь.

Розташувати принтер необхідно поруч з системним блоком таким чином, щоб з'єднувальний шнур не був натягнутий. Забороняється ставити принтери на системний блок.

Для досягнення найбільш чистих, з високою розподільністю зображень і щоб не зіпсувати апарат, має використовуватися папір, вказаний в інструкції до принтера. При застряганні паперу потрібно відкрити кришку і обережно витягнути лоток з папером.

Згідно з інструкцією фірми-виробника потрібно дотримуватися правил зберігання картриджа.

Забороняється:

- зберігати картриджі без упаковки;
- ставити картриджі вертикально;
- перевертати картридж етикеткою донизу;
- відкривати кришку валика і доторкатися до нього;
- самому заповнювати використаний картридж.

Вимоги безпеки після закінчення роботи:

- закінчити та записати у пам'ять комп'ютера файл, що знаходиться в роботі;
- вимкнути принтер та інші периферійні пристрої. Штепсельні вилки витягнути з розеток. Накрити клавіатуру кришкою запобігання попаданню в неї

пилу;

- прибрати робоче місце;
- ретельно вимити руки теплою водою з милом;
- вимкнути кондиціонер, освітлення і загальне електроживлення;

пройти в спеціально обладнаному приміщенні сеанс психофізіологічного розвантаження і зняття втоми з виконанням спеціальних вправ аутогенного тренування.

6. ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА.

6.1 Вступна частина.

У наш час захист навколишнього середовища висувається на перший план. Наслідки недостатньої уваги до проблеми можуть бути катастрофічними. Мова йде не тільки про добробут людства, а про його виживання. Особливо тривожно те, що деградація природного середовища і може виявитися необоротною.

Під охороною навколишнього середовища розуміють сукупність міжнародних, державних і регіональних правових актів, інструкцій і стандартів, які доводять загальні юридичні вимоги до кожного конкретного забруднювача і забезпечують його зацікавленість у виконанні цих вимог, конкретних природоохоронних заходів по втіленню в життя цих вимог.

Тільки якщо всі ці складові частини відповідають один одному за змістом і темпами розвитку, тобто складаються в єдину систему охорони навколишнього природного середовища, можна розраховувати на успіх.

Оскільки не була вирішена вчасно завдання охорони природи від негативного впливу людини, тепер все частіше постає завдання захисту людини від впливу змінилася природного середовища. Обидва ці поняття інтегруються в терміні "охорона навколишнього (людини) природного середовища".

Охорона навколишнього природного середовища складається з:

- правової охорони, що формулює наукові екологічні принципи у вигляді юридичних законів, обов'язкових для виконання;
- матеріального стимулювання природоохоронної діяльності з ціллю зробити її економічно вигідною для підприємств;
- інженерної охорони, що розробляє природоохоронну та ресурсозберігаючу технологію і техніку.

Основними принципами охорони навколишнього природного середовища є:

- пріоритет забезпечення сприятливих екологічних умов для життя, праці та відпочинку населення;

- науково обґрунтоване поєднання екологічних і економічних інтересів суспільства;
- облік законів природи і можливостей самовідновлення і самоочищення її ресурсів;
- недопущення незворотних наслідків для охорони природного середовища і здоров'я людини;
- право населення та громадських організацій на своєчасну і достовірну інформацію про стан навколишнього середовища та негативному впливі на неї і на здоров'я людей різних виробничих об'єктів;
- невідворотність відповідальності за порушення вимог природоохоронного законодавства.

6.2 Забруднення навколишнього середовища підприємствами.

Забруднення навколишнього середовища — надходження в природне середовище будь-яких твердих, газоподібних речовин, мікроорганізмів або енергій (у вигляді звуків, шумів, електромагнітного й радіоактивного випромінювання) у кількостях, що викликають зміну складу й властивостей компонентів природи й (або) чинять шкідливий вплив на людину, флору й фауну.

Види забруднень навколишнього середовища різні й різноманітні: викиди в атмосферу хімічних сполук і сумішей; надходження у водне середовище всіляких виробничих і комунально-побутових відходів, влучення в неї нафтових продуктів; засмічування ландшафтів сміттям і пакувальними матеріалами; забруднення полів, луків, лісів і водних об'єктів пестицидами; підвищення рівня іонізуючої радіації, виробничих і побутових шумів, вібрацій, а також нагромадження тепла в атмосфері.

У результаті господарської діяльності людини в атмосфері з'являється велика кількість забруднюючих речовин. Взаємодія атмосферного повітря з водою і ґрунтом призводить до якісних і кількісних змін усієї біосфери в цілому, підсилюючи і прискорюючи небажані зміни складу і структури атмосферного

повітря, клімату Землі. Найбільш сильні зміни клімату і якості атмосферного повітря спостерігаються у великих містах. Якщо кисень в атмосферному повітрі буде дуже сильно забруднений усілякими речовинами, то поступово у всього живого на Землі буде скорочуватися термін життя, доки не зменшиться до винищення всіх і всього.

Різноманітне втручання людини в природні процеси в біосфері можна згрупувати за наступними видами забруднень, розуміючи під ними будь-які небажані для екосистем антропогенні зміни:

- інгредієнтне (інгредієнт – складова частина складного з'єднання або суміші) забруднення як сукупність речовин, кількісно або якісно далеких природнім біогеоценозам;

- параметричне забруднення (параметр навколишнього середовища – одне з його властивостей, наприклад рівень шуму, освітленості, радіації і т.д.), пов'язане зі зміною якісних параметрів навколишнього середовища;

- біоценотичне забруднення, що полягає у впливі на склад і структуру популяції живих організмів;

- стаціонально-деструкційне забруднення (стація – місце проживання популяції, деструкція – руйнування), яке являє собою зміну ландшафтів екологічних систем у процесі природокористування.

Відомо більше 200 забруднюючих атмосферу речовин, причому в міру освоєння нових технологічних процесів і створення нових продуктів виробництва число забруднюючих речовин зростає. Цьому сприяють також реакції в атмосфері й інших середовищах між забруднюючими речовинами під впливом сонячного світла, окислювально-відновних процесів і інших реакцій. Стрімкий розвиток усіх галузей промисловості, енергетики, транспорту, збільшення чисельності населення й урбанізація, хімізація всіх сфер діяльності людини привели до визначених змін навколишнього природного середовища, у тому числі несприятливих, що полягають головним чином у забрудненні біосфери. Вплив шкідливих речовин антропогенного походження на природне середовище, а так само відгук середовища на ці впливи стають глобальними, всеосяжними.

У великих містах до числа основних джерел забруднення атмосферного повітря відноситься автотранспорт. Гази двигунів, що відходять, містять складну суміш, їх більш ніж двісті компонентів, серед яких чимало канцерогенів. Найбільш значимі фактори негативного впливу автомобільного транспорту на людину і навколишнє середовище наступні:

- забруднення повітря;
- забруднення навколишнього середовища;
- шум, вібрація;
- виділення тепла (розсіювання енергії).

Основна причина забруднення повітря полягає у неповному і нерівномірному згорянні палива. Всього 15% його витрачається на рух автомобіля, а 85% "летить на вітер". До того ж камери згоряння автомобільного двигуна – це своєрідний хімічний реактор, що синтезує отруйні речовини і викидає їх в атмосферу.

Рухаючись зі швидкістю 80-90 км./год. в середньому автомобіль перетворює на вуглекислоту стільки ж кисню, скільки 300-350 чоловік. Але справа не тільки в вуглекислоті. Річний вихлоп одного автомобіля – це 800 кг окису вуглецю, 40 кг окислів азоту і більше 200 кг різних вуглеводнів. У цьому наборі дуже підступна окис вуглецю. Через високу токсичність її допустима концентрація в атмосферному повітрі не повинна перевищувати 1 мг/м³.

Рівень загазованості магістралей залежить від інтенсивності руху автомобілів, ширини і рельєфу вулиці, швидкості вітру, частки вантажного транспорту та автобусів в загальному потоці і інших факторів. При інтенсивності руху 500 транспортних одиниць на годину концентрація окису вуглецю на відкритій території на відстані 30-40 м від автомагістралі знижується в 3 рази і досягає норми.

Ускладнене розсіювання викидів автомобілів на тісних вулицях. У результаті практично всі жителі міста відчувають на собі шкідливий вплив забрудненого повітря.

Чадний газ і окисли азоту, настільки інтенсивно виділяються на перший погляд безневинним голубуватим димком глушника автомобіля – ось одна з

основних причин головних болів, втоми, невмотивованого роздратування, низькою працездатності. Сірчистий газ здатний впливати на генетичний апарат, сприяючи безпліддя і природженої потворності, а всі разом ці чинники ведуть до стресів, нервовим проявам, прагненню до усамітнення, байдужості до найближчих людей. У великих містах також більш широко поширені захворювання органів кровообігу і дихання, інфаркти, гіпертонія і новоутворення. За розрахунками фахівців, "внесок" автомобільного транспорту в атмосферу становить до 90% по окису вуглецю і 70% по окису азоту. Автомобіль також додає в ґрунт і повітря важкі метали та інші шкідливі речовини.

В умовах сильного міського шуму відбувається постійна напруга слухового аналізатора. Це викликає збільшення порогу чутності (10 дБ для більшості людей з нормальним слухом) на 10-25 дБ.

Високі рівні шуму в міському середовищі є одним з агресивних подразників центральної нервової системи. Міський шум справляє негативний вплив і на серцево-судинну систему. Ішемічна хвороба серця, гіпертонічна хвороба, підвищений вміст холестерину в крові зустрічаються частіше в осіб, що проживають у гучних районах.

Найбільші рівні шуму 90-95 дБ відзначаються на магістральних вулицях міст з середньою інтенсивністю руху 2-3 тис. і більше транспортних одиниць на годину.

Рівень вуличних шумів обумовлюється інтенсивністю, швидкістю і характером (складом) транспортного потоку. Крім того, він залежить від планувальних рішень (поздовжній і поперечний профіль вулиць, висота і щільність забудови) і таких елементів благоустрою, як покриття проїзної частини та наявність зелених насаджень. Кожен з цих факторів здатний змінити рівень транспортного шуму в межах до 10 дБ.

Шум, що виникає на проїжджій частині магістралі, поширюється не тільки на при магістральну територію, але і вглиб житлової забудови. Так, в зоні найбільш сильного впливу шуму знаходяться частини кварталів і мікрорайонів, розташованих уздовж магістралей загальноміського значення (еквівалентні рівні

шуму від 67,4 до 76,8 дБ). Рівні шуму, заміряні в житлових кімнатах при відкритих вікнах, орієнтованих на зазначені магістралі, всього на 10-15 дБ нижче.

6.3 Основні заходи щодо зниження забруднення атмосферного повітря викидами автомобільного транспорту.

Увага вчених, інженерів, агрономів, інших фахівців і громадськості зосереджено на усуненні всіх видів забруднюючих речовин, що представляють найбільшу небезпеку для здоров'я людей. Способи й шляхи боротьби із забрудненнями навколишнього середовища різноманітні: пристрої очисних споруджень, установка фільтрів, створення й впровадження безвідхідних технологій, замкнутих циклів водокористування, заміна хімічних способів боротьби зі шкідниками й хворобами сільськогосподарських і лісових рослин біологічними, звукопоглинаючі пристрої, використання відходів для одержання корисної продукції, пошуки нових видів палива, використання для технічних і побутових потреб тепла геотермальних джерел, сили вітру, морських припливів, сонячного тепла, установлення режиму користування автомобільним і авіаційним транспортом

Проблема забруднення навколишнього середовища автомобільним транспортом цілком очевидно є глобальною. По всьому світу кількість автомобілів з кожним днем збільшується в геометричній прогресії. Все більше і більше людей мають свою власну машину. Але багато хто зовсім не замислюється про те, до чого все це врешті-решт приведе. Для того, щоб зберегти людству автомобіль, необхідно якщо не виключити, то звести до мінімуму шкідливі викиди. Робота в цьому напрямі ведеться в усьому світі і дають певні результати. Автомобілі випускаються в даний час в промислово розвинених країнах, викидають шкідливих речовин у 10-15 разів менше, ніж 10-15 років тому. У всіх розвинених країнах відбувається посилення нормативів на шкідливі викиди при роботі двигуна. У 2000 р. введені більш суворі норми. Відбувається не тільки кількісне посилення норм, але і їх якісна зміна. Постійно розширюється перелік речовин, зміст яких має знаходитися під контролем.

В даний час йде боротьба з автомобільною небезпекою. Конструюються фільтри, розробляються нові види пального, що містять менше свинцю.

Загальні шляхи зниження екологічних збитків від транспорту полягають в наступному:

- оптимізація руху міського транспорту
- розробка альтернативних енергетичних джерел;
- опалювання і очищення органічного палива;
- створення (модифікація) двигунів, що використовують альтернативні палива;
- захист від шуму;
- економічні ініціативи з управління автомобільним парком і рухом.

6.4 Заходи запобігання забруднення навколишнього середовища застарілою комп'ютерною технікою підприємства.

Природоохоронною є будь-яка діяльність, спрямована на збереження якості навколишнього середовища на рівні, що забезпечує стійкість біосфери. До неї відноситься як великомасштабна, здійснювана на загальнодержавному рівні діяльність, так і діяльність окремих підприємств.

Існують два основних напрямки природоохоронної діяльності підприємств. Перший – очищення шкідливих викидів. Цей шлях "у чистому вигляді" малоефективний, тому що з його допомогою далеко не завжди вдається цілком припинити надходження шкідливих речовин у біосферу. До того ж скорочення рівня забруднення одного компонента навколишнього середовища веде до посилення рівня забруднення іншого.

Другий напрямок – усунення самих причин забруднення, що вимагає розробки маловідходних, а в перспективі й безвідходних технологій виробництва, які дозволяють комплексно використовувати вихідну сировину й утилізувати максимум шкідливих для біосфери речовин.

Однак далеко не для всіх галузей знайдені прийнятні техніко-економічні рішення по різкому скороченню кількості відходів, що утворюються, і їхньої

утилізації, тому в цей час доводиться працювати по обох зазначених напрямках.

Людство знаходиться в постійній залежності від процесу модернізації і заміни електронної техніки. Виробництво офісного устаткування, комп'ютерів і мобільних пристроїв з кожним роком нестримно збільшується. Проте є в прогресу і неприваблива сторона – застарілу апаратуру, що вийшла з використання, необхідно кудись дівати – виникає поняття "електронне сміття", і про його об'єми можна лише здогадуватися.

На відміну від органічних відходів, техногенне сміття неможливо утилізувати, використовуючи природні процеси регенерації. Більш того, враховуючи присутність в багато компонентах важких металів і отруйних речовин, спалювати його подібно до побутових відходів теж не можна. Переробка застарілої електроніки вимагає значних зусиль і є чималою проблемою. Ігнорувати її неможливо, тому що вона несе серйозну загрозу для екології.

Навіть по приблизним оцінках з 1991 р. до України було завезено більше 10 млн. одиниць персональної і оргтехніки загальною масою понад 400 тис. тон і ще порядку 20 млн. мобільних телефонів (близько 2400 т). Причому в сучасної електроніки є своя специфіка – вона в набагато більшій мірі складається з неліквідних матеріалів. Так, один комп'ютер класу 286 або 386 в комплекті з ЕПТ-монітором орієнтування містить від 10 до 20 кг скла, 3 кг АВС-пластика, приблизно стільки ж чорних металів, по 200 г міді і алюмінію, всього 1 г срібла і не більше 10 міліграма золота. Тобто в основному йдеться про матеріали, вторинне використання яких неможливе, і тому вартість переробки значно перевищує вартість отримуваних металів.

Окрему проблему утилізації представляють кінескопи моніторів – із-за вмісту отруйних речовин їх не можна навіть розібрати – як і ядерні відходи, їх слід лише ховати в спеціальних могильниках. Розвинені країни задихаються від електронного сміття, оскільки одних лише неробочих кінескопів і моніторів в них скупчилося більше 100 млн. Те ж саме відноситься і до батарей живлення і акумуляторів, що містять такі метали, як свинець, кадмій, цинк і нікель.

У Японії, США і країнах ЄС останніми роками активно розвивається

напрям утилізації старих комп'ютерів. У США компанія Dell почала переробку застарілої техніки восени 2002 р. У Германії щорік викидається 2,2 млн. комп'ютерів, принтерів і моніторів. Середньостатистичний німець повністю або частково оновлює свій ПК кожні 2-3 роки і вимушений платити за утилізацію старого (залежно від моделі) від 15 до 30 євро.

В наш час німецький концерн Siemens і японська компанія Fujitsu мають намір створити спільне підприємство по утилізації персональних комп'ютерів – вони планують побудувати заводи в п'яти містах Німеччини. Є в намірах також створення аналогічних виробництв у Великобританії, Франції і Скандинавських країнах. На думку ініціаторів проекту, централізована переробка ПК поліпшить екологічну обстановку і сприятиме збільшенню попиту на нові моделі.

У Японії Sony організувала службу по утилізації застарілих персональних комп'ютерів своєї марки. Компанія створила спеціальну мережу по їх збору і доставці, а вартість переробки одного ПК складає 4-5 тис. ієн (\$40).

У нашій державі, на відміну від індустріально розвинених країн, проблема переробки електронного сміття практично пущена на самоплив – утилізація застарілої техніки є турботою виключно її власників, тому найчастіше устаткування закінчує свої дні на звалищі.

Згідно із законом для списання і утилізації ПК і офісного устаткування юридична особа повинна звернутися до одного із спеціалізованих підприємств, які займаються виробництвом, ремонтом і обслуговуванням персональної техніки, з проханням про проведення експертизи, і отримати висновок про те, що дана модель морально застаріла, знята з виробництва і ремонту не підлягає. Лише після цього можна укласти договір з відповідною організацією про утилізацію відходів.

Ліцензію на утилізацію відходів, що містять дорогоцінні метали, видає Міністерство фінансів, кольорові і чорні – Міністерство промислової політики, а полімери і отруйні матеріали – Міністерство екології і природних ресурсів. Але, як ми вже з'ясували, персональні комп'ютери містять всі ці речовини. Тому в їх відношенні діє такий порядок: устаткування необхідно здати підприємству, що має дозвіл на роботу з дорогоцінними металами. Там здійснюються розбирання

техніки і сортування матеріалів, які потім передаються компаніям, на законних підставах що займається переробкою певного вигляду відходів.

Утилізація застарілої офісної техніки вимагає грошей, проте такі витрати неминучі. У силах же законодавчих органів зробити цей процес по можливості простим і ефективним. При розумній організації переробка "електронного сміття" може стати економічно вигідною діяльністю і сприяти створенню додаткових робочих місць.

ВИСНОВКИ

На базі проведених досліджень теорії масового обслуговування, в процесі виконання дипломної роботи було спроектовано інформаційну систему обробки депутатських звернень, що дозволило підвищити ефективність ведення адміністративної документації.

Для роботи з даною системою було створене програмне забезпечення для обліку документообігу та формування на його основі картки рішення. Даний засіб розроблений для спрощення роботи керівника відділу роботи з документами. В роботі представлено етапи створення інформаційної системи та програмного засобу для роботи з нею.

Програмне забезпечення інформаційної системи здійснює організацію зручного пошуку записів за певними критеріями, створення звіту на їх основі, зручне представлення вхідних і вихідних даних через графічний інтерфейс. Розроблене програмне забезпечення може використовувати користувач з базовими знаннями про роботу з персональним комп'ютером.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Котляр, М. М. Теорія масового обслуговування: підручник / М. М. Котляр. — К.: Вид-во КНУ, 2017. — 312 с.
2. Шейко, В. І. Теорія масового обслуговування та її застосування в інформаційних технологіях / В. І. Шейко, М. В. Іванова. — Львів: Видавничий центр ЛНУ, 2016. — 450 с.
3. Мельничук, А. С. Інформаційні системи та технології в обчислювальній техніці / А. С. Мельничук. — Харків: Вища школа, 2015. — 280 с.
4. Грінберг, Ю. М. (2020). Розробка інформаційних систем: підручник. Київ: Кондор.
5. Шакирзян, А. Р. (2019). Інформаційні системи і технології: теорія та практика. Львів: Вид-во Львівської політехніки.
6. Петренко, О. В. (2018). Інформаційні системи та технології: основи проектування і розробки. Харків: Харківський національний університет радіоелектроніки.
7. Смирнов, О. В. (2021). Моделювання та розробка інформаційних систем. Київ: Видавництво «Освітня книга».
8. Теорія масового обслуговування: основи, методи та моделі. (2023). — URL: www.mos-theory.com.
9. Борисова, Н. В. (2023). Електронний документообіг в системі державного управління: теорія і практика. Київ: Видавництво "Юридична думка".
10. Петров, А. О. (2022). Інформаційні технології в органах державної влади: сучасні підходи до електронного документообігу. Науковий журнал "Інформаційні технології та право", 45(2), 98-112.
11. Мельник, І. А. (2021). Електронне урядування в Україні: нормативно-правові засади та практика впровадження. Київ: Нова Книга.
12. Єрмоленко, В. А. (2020). Електронний документообіг в державному управлінні: теоретичні засади та інновації. Журнал "Державне управління", 34(3), 56-67.

13. Литвиненко, Ю. В. (2019). Автоматизація процесів електронного документообігу в органах влади: стратегічні аспекти. Херсон: Видавничий дім "Освіта".

ТЕХНІЧНЕ ЗАВДАННЯ

Назва: "Інформаційна система обробки депутатських запитів".

Головна задача інформаційної системи – автоматизація обліку депутатських запитів, редагування та створення звітів про їх зміст з різними варіантами відбору.

1. Підстави для розробки: документ "Завдання на дипломний проект (роботу)" у відповідності до бланку завдання.

2. Призначення розробки.

Інформаційна системи призначена спростити створення звітів, сортування інформації про депутатські рішення, вибір для звіту документів з певними атрибутами, класифікаторами.

3. Вимоги до ПЗ ІС.

Вимоги до функціональних характеристик:

1. Зберігання зареєстрованих рішень в базі даних.
2. Зміна, додавання та редагування рішень в базу даних.
3. Швидкий пошук за критеріями.
4. Налагодження та редагування постійних комісій.
5. Налагодження та редагування постійних класифікаторів.
6. Налагодження та редагування пов'язаних комісій.
7. Генерація документів відповідно до обраного рішенням.

Вимоги до надійності.

Для підтримки надійне функціонування продукту "Інформаційна система обробки депутатських запитів", необхідно заповнювати усі обов'язкові поля облікової картки документу. У випадку не заповнення обов'язкового поля програма буде повідомляти користувача за допомогою спеціального вікна з текстом помилки. Прочитавши повідомлення та натиснувши кнопку "ОК",

користувач знову отримає можливість працювати з ІС та вводити дані.

Умови експлуатації.

Для роботи з ПЗ ІС, користувач повинен володіти базовими навиками роботи з персональним комп'ютером. У використанні ПЗ має інтуїтивно зрозумілий інтерфейс, тому спеціальна кваліфікація користувачеві не потрібна.

До умов експлуатації можна віднести температуру повітря від 10°C до 35°C. При цьому вологість повітря не повинна перевищувати 70%. Це забезпечить нормальну роботу комп'ютера та відповідно програмного продукту.

Вимоги до складу та параметрів технічних засобів.

Для нормального функціонування ПЗ необхідний наступний мінімальний технічний склад:

- процесор Athlon AMD 2,4 ГГц і вище;
- оперативна пам'ять 2 Gb і більше;
- жорсткий диск 300 Gb і вище.
- монітор;
- клавіатура, миша.

Вимоги до інформаційної й програмної сумісності.

Для нормальної роботи програмного ПЗ на ПК має бути встановлена операційна система Microsoft Windows 7 / 10/ XP / Vista, а також віртуальна машина Java – Java JRE 6, або Java SDK 6.

Інтерфейс для ПЗ створений за допомогою мови програмування Java, тобто нові модулі, які будуть написані та додані до програми, також повинні бути реалізовані за допомогою цієї мови.

Також програма взаємодіє з базою даних, де зберігається уся інформація. База даних була створена за допомогою мови SQL та розміщена на сервері MySQL, тому даний сервер і база даних також повинні бути встановлені на комп'ютері.

Вимоги до маркування та пакування.

Якщо програмний продукт треба транспортувати та встановити на іншому комп'ютері, необхідно папку, де розміщено програму, додати до архіву за допомогою архіваторів RAR або ZIP. Це зменшує обсяги програми та додає зручності при транспортуванні.

Вимоги до транспортування й зберігання.

Транспортування даного програмного засобу можна здійснювати за допомогою Flash або інших накопичувачів інформації. Після транспортування на інший комп'ютер, програма не потребує інсталяції, але повинні бути встановлені вищезазначені програмні засоби. Програма розміщується на жорсткому диску комп'ютера, після чого вона придатна для роботи. Зберігати програму на CD/DVD дисках можна впродовж 2 років, на Flash-накопичувачі – впродовж 3 років.

4. Вимоги до програмної документації.

Документація до програмного засобу повинна містити додатки програмного коду продукту "Інформаційна система обробки депутатських запитів", опис програми, програму і методику випробувань, а також, інструкцію користувача.

5. Техніко-економічні показники.

Впровадження даного програмного засобу в роботу дозволяє автоматизувати створення звітів по класифікаторам, значно підвищивши ефективність ведення, використання, а також, зберігання депутатських рішень. Продуктивність праці користувача збільшується приблизно в 3-4 рази.

Програмний засіб буде використовуватися щоденно.

6. Стадії та етапи розробки.

Стадії та етапи розробки відповідають календарному плану, затвердженому кафедрою ІУСТ.

6. Послідовність контролю та прийняття.

Перед впровадженням ПЗ ІС його тестує розробник, а після цього тестування проводять користувачі, які будуть працювати з цим програмним засобом. Якщо у останніх не виникає ніяких зауважень, то програма впроваджується, в іншому випадку розробник доробляє програму і тестування проводиться повторно.

ІНСТРУКЦІЯ З ЕКСПЛУАТАЦІЇ

1. Установка програми.

Програма працює під управлінням ОС Windows Windows 7 / 10/ XP / Vista. Для запуску програми необхідно скопіювати папку TownCouncil на жорсткий диск.

Апаратні засоби повинні задовольняти наступним вимогам: процесор AMD Athlon і вище, 2Гб оперативної пам'яті.

2. Запуск програми відбувається шляхом запуску файлу TownCouncil.jar, що знаходиться в робочому каталозі програми.

3. Інструкція для використання функцій інтерфейсу.

Головне вікно інтерфейсу зображено на рисунку Б 1.

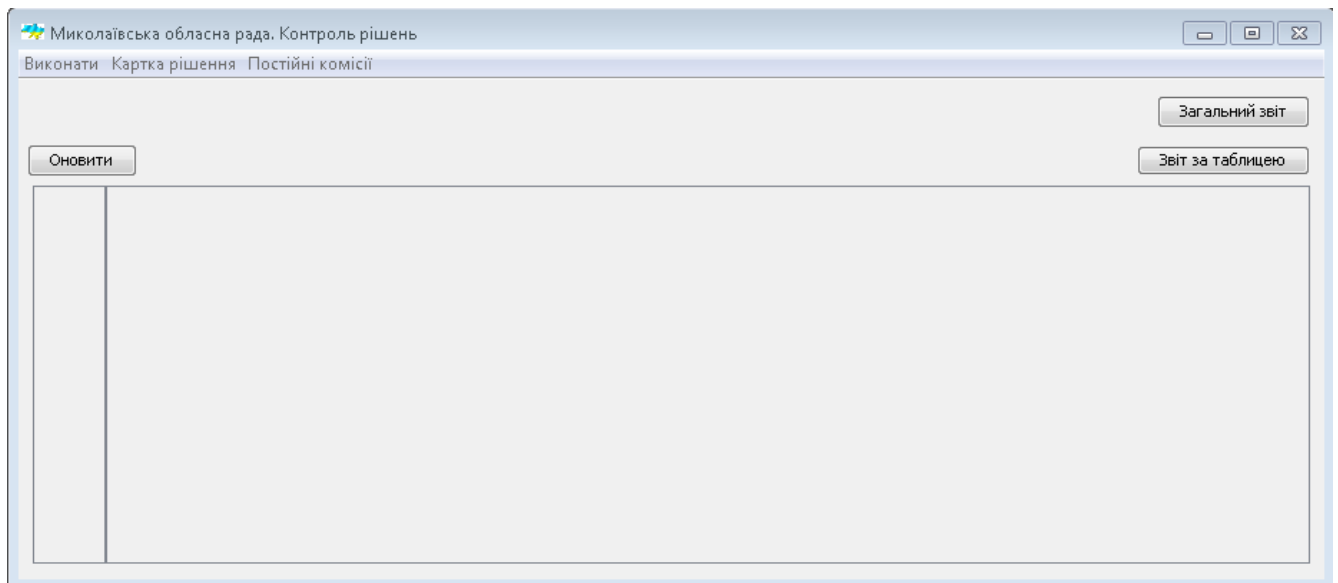


Рисунок Б1 – Головне меню

Заведення нового рішення зображено на рисунку Б2.

Інтерфейс пошуку зображено на рисунку Б3.

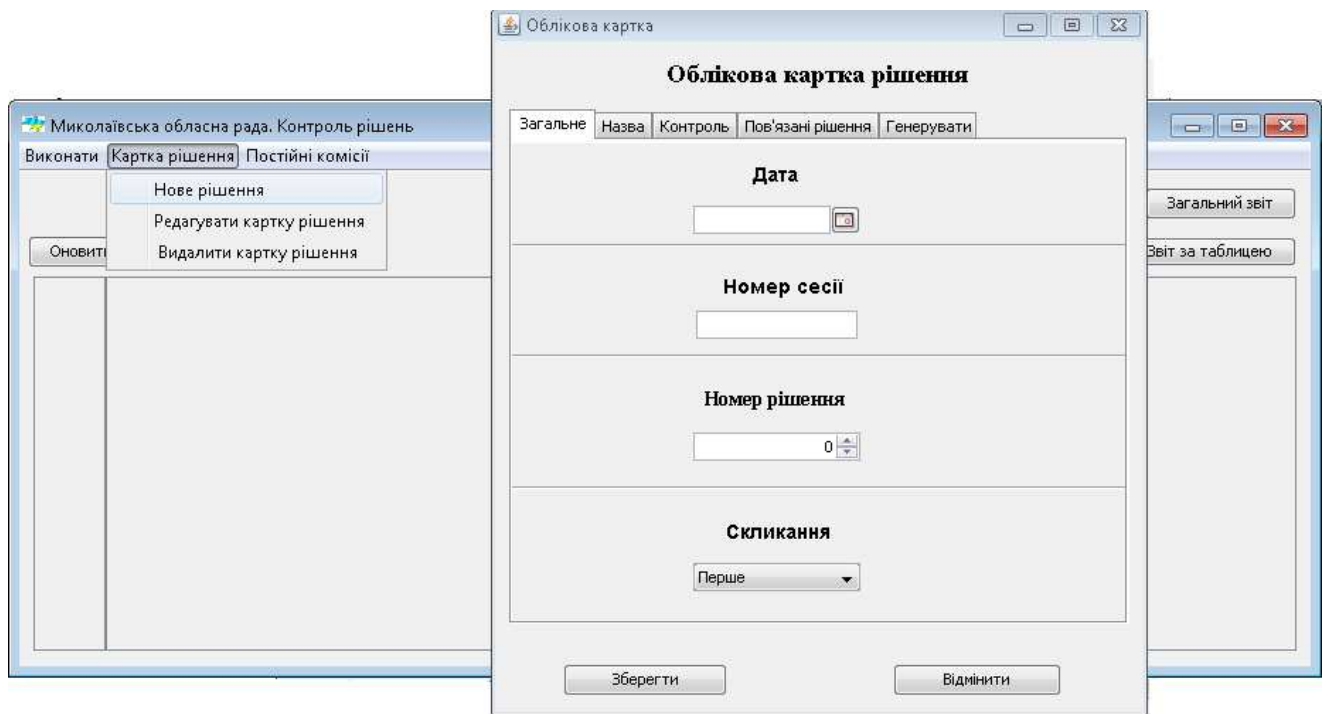


Рисунок Б2 – Занесення нового рішення в базу даних

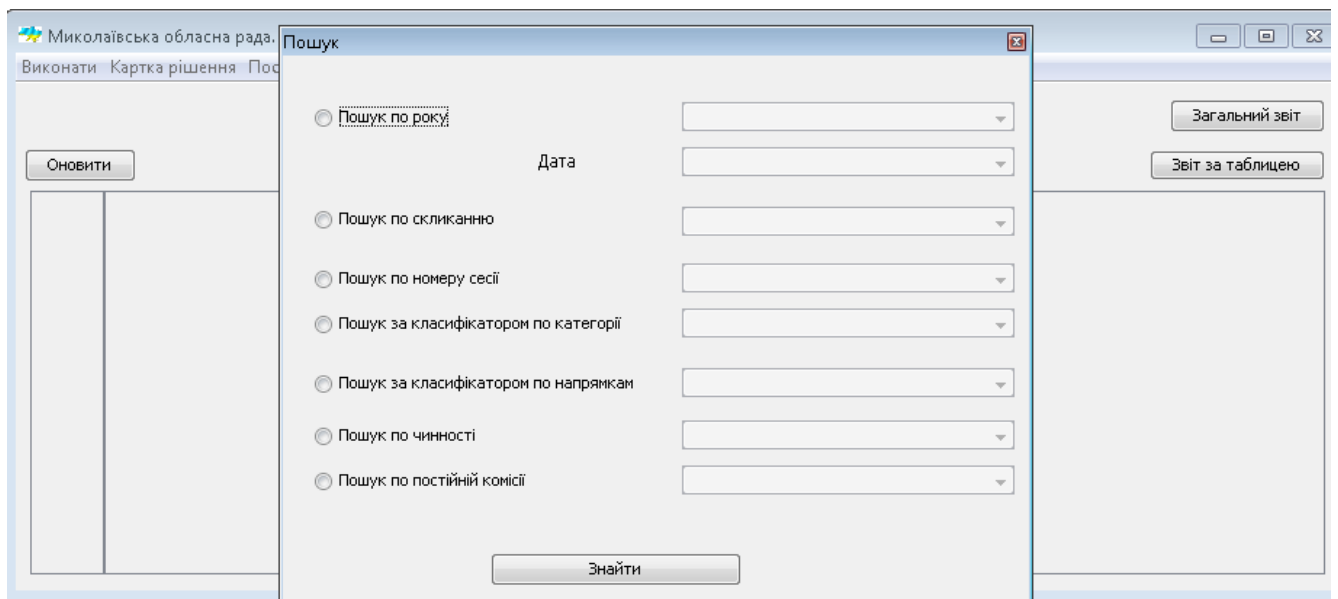


Рисунок Б3 – Пошук по базі даних

ТЕКСТ ПРОГРАМИ

```
Клас MainFrame
package townCouncil.MainFrame;
import Excel.ExcelJTable;
import dao.*;
import java.awt.Image;
import java.awt.Toolkit;
import java.awt.TrayIcon.MessageType;
import java.io.IOException;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Comparator;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.DefaultCellEditor;
import javax.swing.DefaultComboBoxModel;
import javax.swing.JComboBox;
import javax.swing.JOptionPane;
import javax.swing.JRadioButton;
import javax.swing.JTable;
import javax.swing.ListSelectionModel;
import javax.swing.RowSorter.SortKey;
import javax.swing.SortOrder;
import javax.swing.event.ListSelectionEvent;
import javax.swing.event.ListSelectionListener;
import javax.swing.plaf.basic.BasicBorders;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.TableRowSorter;
import townCouncil.CardDecision.CardDecision;
import vo.InformationCard;

/**
 * Данный класс реализует первоначальное окно для отображение базы данных,
 * создание или редактирование карт решений.
 */
* @author Галич Максим
*/
public class MainFrame extends javax.swing.JFrame {

    Logger logger = Logger.getLogger();
    /**
     * Объект для добавления и удаления имён комиссий, инициализация происходит в
     * конструкторе, вызовы смотри ниже по коду.
     */
    private DefaultComboBoxModel defaultListNames;
```



```

/**
 * Объект для добавления и удаления головы комиссии, инициализация происходит
 * в конструкторе, вызовы смотри ниже по коду.
 */
privateDefaultComboBoxModeldefaultListHead;
/**
 * Объект для добавления и удаления аббревиатур комиссий, инициализация
 * происходит в конструкторе, вызовы смотри ниже по коду.
 */
privateDefaultComboBoxModeldefaultListAbbreviation;
/**
 * Переменная которая хранит в себе состояние вызванного действия, далее
 * используется в switch, Смотри метод добавление новых значений в комиссии.
 */
privateintchooseAction;
/**
 * Объект класса DataBaseMainFrame, необходим для работы с базой данных.
 */
privateDataBaseMainFramedbmf;
/**
 * Объект типа DefaultTableModel, необходим для создания таблицы.
 */
privateDefaultTableModeltableModel;
/**
 * Коллекция содержащая в себе компоненты типа JRadioButton.
 *
 * @see #initBoxes()
 */
private ArrayList<JRadioButton> boxes;
/**
 * Коллекция содержащая в себе компоненты типа JComboBox.
 *
 * @see #initBoxes()
 */
private ArrayList<JComboBox> comboBoxes;
private DefaultTableModel classifierTableModel;
private ArrayList<Integer> countNewClassifierState;
private ArrayList<Integer> countNewCommission;
private ArrayList<Integer> countNewMoreClassifier;
private ArrayList<Integer> countNewClassifier;
private DefaultTableModel relatedTableModel;
private DefaultTableModel moreClassifierDefaultModel;
private DefaultTableModel classifierTableEdit;
private boolean isSelectedDateOption = false;
//hren
private ArrayList<Object[]> tempObjs;
private static Object reportClassifier;
private static String reportName;
//id vzatogo elementa s tablicu

```

```

static int idRow;

public static Object getReportClassifier() {
    return reportClassifier;
}

/**
 * Инициализация всех форм проекта Swing.
 */
public MainFrame() {
    Toolkit toolkit = Toolkit.getDefaultToolkit();
    Image image =
toolkit.getImage(getClass().getResource("/source/image/icon.png"));
    setIconImage(image);
//    Dimension screenSize = toolkit.getScreenSize();
//    int screenWidth = screenSize.width;
//    int screenHeight = screenSize.height;
//    setSize (screenWidth,screenHeight);
    initComponents();
    initObjects();
    initBoxs();
}

/**
 * Метод инициализирует объекты которые будут включительно работать в данном
 * классе.
 */
private void initObjects() {
    dbmf = new DataBaseMainFrame();
    boxs = new ArrayList<>();
    comboBoxs = new ArrayList<>();
    countNewClassifierState = new ArrayList<>();
    countNewCommision = new ArrayList<>();
    countNewMoreClassifier = new ArrayList<>();
    countNewClassifier = new ArrayList<>();
    this.setLocationRelativeTo(null);
}

/**
 * Метод добавляет в коллекции объекты с которыми будет опирировать два
 * метода в данном классе.Это conntrolerViewRadioButton и
 * conntrolerViewComboBox Там где написано Magicka не модифицируете так как
 * строки которые передаются в метод setName() играют ключевую роль в
 * генерации SQL запроса в Базу данных.
 */
private void initBoxs() {
    boxs.add(seachConvocation);
    boxs.add(seachSession);
    boxs.add(seachClassifier);
}

```

```

        boxes.add(seachState);
        boxes.add(seachCommision);
        boxes.add(seachYear);
        boxes.add(seachAddClassifier);
comboBoxs.add(listConvocation);
        comboBoxs.add(listClassifier);
        comboBoxs.add(listState);
        comboBoxs.add(listCommision);
        comboBoxs.add(listSession);
        comboBoxs.add(listYear);
        comboBoxs.add(listDateOption);
        comboBoxs.add(listAddClassifier);
/**
 * Next code is a Magicka. Do not modify him ! //
 */
listConvocation.setName("Скликання");
listClassifier.setName("Класифікатор");
listState.setName("Чіність");
listCommision.setName("Абревіатура");
listSession.setName("Номер сесії");
listYear.setName("Рік");
listAddClassifier.setName("Класифікатор_додатковий");
    }

    private void initListCommision() {
        try {
defaultListNames = new DefaultComboBoxModel(dbmf.getCommisionName());
            defaultListHead = new DefaultComboBoxModel(dbmf.getCommisionHead());
            defaultListAbbreviation = new
DefaultComboBoxModel(dbmf.getCommisionAbbreviation());
        } catch (SQLException ex) {
            System.out.println(ex.getMessage());
            ex.printStackTrace();
            JOptionPane.showMessageDialog(this, "Сбой работы с базой данных,
перезапустите приложения и попробуйте снова", "Ошибка работы с базой данных",
MessageType.ERROR.ordinal());
        }
    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        seachFrame = new javax.swing.JFrame();
        seachDatabase = new javax.swing.JButton();
        listYear = new javax.swing.JComboBox();
        listConvocation = new javax.swing.JComboBox();
        seachConvocation = new javax.swing.JRadioButton();
        seachSession = new javax.swing.JRadioButton();

```

```

seachClassifier = new javax.swing.JRadioButton();
seachState = new javax.swing.JRadioButton();
listClassifier = new javax.swing.JComboBox();
listState = new javax.swing.JComboBox();
seachCommision = new javax.swing.JRadioButton();
listCommision = new javax.swing.JComboBox();
listSession = new javax.swing.JComboBox();
seachYear = new javax.swing.JRadioButton();
seachAddClassifier = new javax.swing.JRadioButton();
listAddClassifier = new javax.swing.JComboBox();
jLabel1 = new javax.swing.JLabel();
listDateOption = new javax.swing.JComboBox();
jMenuBar2 = new javax.swing.JMenuBar();
jMenu2 = new javax.swing.JMenu();
jMenu3 = new javax.swing.JMenu();
settingCommittee = new javax.swing.JFrame();
jScrollPane2 = new javax.swing.JScrollPane();
listHeadCommision = new javax.swing.JList();
jScrollPane3 = new javax.swing.JScrollPane();
listNamesCommision = new javax.swing.JList();
jScrollPane4 = new javax.swing.JScrollPane();
listAbbreviation = new javax.swing.JList();
jLabel3 = new javax.swing.JLabel();
jLabel4 = new javax.swing.JLabel();
jLabel5 = new javax.swing.JLabel();
jLabel6 = new javax.swing.JLabel();
addHeadCommision = new javax.swing.JButton();
removeHeadCommision = new javax.swing.JButton();
addNamesCommision = new javax.swing.JButton();
renoveNamesCommision = new javax.swing.JButton();
addAbbreviationCommision = new javax.swing.JButton();
removeAbbreviationCommision = new javax.swing.JButton();
newValue = new javax.swing.JFrame();
textNewValue = new javax.swing.JLabel();
valueField = new javax.swing.JTextField();
createNewNotation = new javax.swing.JButton();
editClassifier = new javax.swing.JFrame();
jScrollPane6 = new javax.swing.JScrollPane();
classifierTable = new javax.swing.JTable();
saveClassfierModel = new javax.swing.JButton();
cancelClassfierModel = new javax.swing.JButton();
addRowClassifier = new javax.swing.JButton();
removeRowClassifier = new javax.swing.JButton();
relatedCommision = new javax.swing.JFrame();
jScrollPane5 = new javax.swing.JScrollPane();
relatedTable = new javax.swing.JTable();
addCommisionState = new javax.swing.JButton();
removeCommisionState = new javax.swing.JButton();
saveCommisionState = new javax.swing.JButton();

```

```

cancel = new javax.swing.JButton();
frameMoreClassifier = new javax.swing.JFrame();
jScrollPane7 = new javax.swing.JScrollPane();
tableMoreClassifier = new javax.swing.JTable();
addMoreClassifier = new javax.swing.JButton();
removeMoreClassifier = new javax.swing.JButton();
saveMoreClassifier = new javax.swing.JButton();
cancelMoreClassifier = new javax.swing.JButton();
frameMoreClassifier1 = new javax.swing.JFrame();
jScrollPane8 = new javax.swing.JScrollPane();
tableMoreClassifier1 = new javax.swing.JTable();
addMoreClassifier1 = new javax.swing.JButton();
removeMoreClassifier1 = new javax.swing.JButton();
saveMoreClassifier1 = new javax.swing.JButton();
cancelMoreClassifier1 = new javax.swing.JButton();
jScrollPane1 = new javax.swing.JScrollPane();
jTable1 = new javax.swing.JTable();
searchParametrValue = new javax.swing.JTextField();
searchParametrValue.setVisible(false);
searchParametrText = new javax.swing.JLabel();
jButton1 = new javax.swing.JButton();
jButton2 = new javax.swing.JButton();
jScrollPane9 = new javax.swing.JScrollPane();
jTable2 = new javax.swing.JTable();
jButton3 = new javax.swing.JButton();
jMenuBar1 = new javax.swing.JMenuBar();
mainMenu = new javax.swing.JMenu();
searchMenu = new javax.swing.JMenuItem();
exitMenu = new javax.swing.JMenuItem();
cardDecisionMenu = new javax.swing.JMenu();
createCardDecision = new javax.swing.JMenuItem();
openCardDecision = new javax.swing.JMenuItem();
DeleteCardDecision1 = new javax.swing.JMenuItem();
optionMenu = new javax.swing.JMenu();
jMenuItem1 = new javax.swing.JMenuItem();
jMenuItem3 = new javax.swing.JMenuItem();
jMenuItem5 = new javax.swing.JMenuItem();
jMenuItem2 = new javax.swing.JMenuItem();
jMenuItem4 = new javax.swing.JMenuItem();

searchFrame.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
searchFrame.setTitle("Пошук");
searchFrame.setMinimumSize(new java.awt.Dimension(520, 400));
searchFrame.setResizable(false);
searchFrame.setType(java.awt.Window.Type.UTILITY);
searchFrame.setLocationRelativeTo(null);

searchDatabase.setText("Знайти");

```

```

searchDatabase.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        searchDatabaseActionPerformed(evt);
    }
});

listYear.setEnabled(false);
listYear.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        listYearActionPerformed(evt);
    }
});

listConvocation.setEnabled(false);

searchConvocation.setText("Пошук по скликанню");
searchConvocation.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        searchConvocationActionPerformed(evt);
    }
});

searchSession.setText("Пошук по номеру сесії");
searchSession.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        searchSessionActionPerformed(evt);
    }
});

searchClassifier.setText("Пошук за класифікатором по категорії");
searchClassifier.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        searchClassifierActionPerformed(evt);
    }
});

searchState.setText("Пошук по чинності");
searchState.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        searchStateActionPerformed(evt);
    }
});

listClassifier.setEnabled(false);
listClassifier.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        listClassifierActionPerformed(evt);
    }
});

```

```

listState.setEnabled(false);

seachCommision.setText("Пошук по постійній комісії");
seachCommision.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
seachCommisionActionPerformed(evt);
    }
});

listCommision.setEnabled(false);

listSession.setEnabled(false);
listSession.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        listSessionActionPerformed(evt);
    }
});

seachYear.setText("Пошук по року");
seachYear.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        seachYearActionPerformed(evt);
    }
});

seachAddClassifier.setText("Пошук за класифікатором по напрямкам");
seachAddClassifier.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        seachAddClassifierActionPerformed(evt);
    }
});

listAddClassifier.setEnabled(false);
listAddClassifier.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        listAddClassifierActionPerformed(evt);
    }
});

jLabel1.setFont(new java.awt.Font("Tahoma", 0, 12)); // NOI18N
jLabel1.setText("Дата");

listDateOption.setMaximumRowCount(10);
listDateOption.setEnabled(false);
listDateOption.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        listDateOptionActionPerformed(evt);
    }
});

```

```

        javax.swing.GroupLayout seachFrameLayout = new
javax.swing.GroupLayout (seachFrame.getContentPane());
        seachFrame.getContentPane().setLayout (seachFrameLayout);
        seachFrameLayout.setHorizontalGroup(

seachFrameLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(seachFrameLayout.createSequentialGroup())

.addGroup(seachFrameLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
NG)
.addGroup(seachFrameLayout.createSequentialGroup())
        .addGap(143, 143, 143)
        .addComponent(seachDatabase,
javax.swing.GroupLayout.PREFERRED_SIZE, 173, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(0, 135, Short.MAX_VALUE)
        .addGroup(seachFrameLayout.createSequentialGroup())
        .addGap(18, 18, 18)

.addGroup(seachFrameLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
        .addComponent(seachConvocation,
javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(seachSession,
javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(seachClassifier,
javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addComponent(seachCommision,
javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addComponent(seachState,
javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(seachYear,
javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addComponent(seachAddClassifier,
javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addGroup(seachFrameLayout.createSequentialGroup())
        .addComponent(jLabel1,
javax.swing.GroupLayout.PREFERRED_SIZE, 41, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(30, 30, 30)))
        .addGap(28, 28, 28)

.addGroup(seachFrameLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
NG)
        .addComponent(listConvocation, 0,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addComponent(listClassifier, 0,

```



```

javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addComponent(listSession, 0,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addComponent(listState, 0,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addComponent(listCommision, 0,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addComponent(listAddClassifier, 0,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addComponent(listYear, 0,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addComponent(listDateOption, 0,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))
        .addContainerGap()
    );
    seachFrameLayout.setVerticalGroup(

seachFrameLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(seachFrameLayout.createSequentialGroup()
            .addGap(32, 32, 32)

.addGroup(seachFrameLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
            .addComponent(listYear)
            .addComponent(seachYear, javax.swing.GroupLayout.PREFERRED_SIZE,
0, Short.MAX_VALUE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(seachFrameLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(jLabell1, javax.swing.GroupLayout.PREFERRED_SIZE, 19,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(listDateOption,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(18, 18, 18)

.addGroup(seachFrameLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
            .addComponent(seachConvocation)
            .addComponent(listConvocation, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(18, 18, 18)

.addGroup(seachFrameLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(seachSession)
            .addComponent(listSession, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))

```

```

        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

        .addGroup(seachFrameLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

            .addComponent(seachClassifier)
            .addComponent(listClassifier, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(18, 18, 18)

        .addGroup(seachFrameLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

            .addComponent(seachAddClassifier)
            .addComponent(listAddClassifier,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(13, 13, 13)

        .addGroup(seachFrameLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

            .addComponent(seachState)
            .addComponent(listState, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

        .addGroup(seachFrameLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

            .addComponent(seachCommision)
            .addComponent(listCommision,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
49, Short.MAX_VALUE)
            .addComponent(seachDatabase)
            .addContainerGap())
    );

    listDateOption.getAccessibleContext().setAccessibleName("dateOption");

    jMenu2.setText("File");
    jMenuBar2.add(jMenu2);

    jMenu3.setText("Edit");
    jMenuBar2.add(jMenu3);

    settingCommittee.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE
);

    settingCommittee.setAlwaysOnTop(true);
    settingCommittee.setMinimumSize(new java.awt.Dimension(714, 380));

```

```

settingCommittee.setResizable(false);
    settingCommittee.setLocationRelativeTo(null);

    jScrollPane2.setViewportViewView(listHeadCommision);

    jScrollPane3.setViewportViewView(listNamesCommision);

jScrollPane4.setViewportViewView(listAbbreviation);

    jLabel3.setFont(new java.awt.Font("Tahoma", 1, 11)); // NOI18N
    jLabel3.setText("Голова комісії");

    jLabel4.setFont(new java.awt.Font("Tahoma", 1, 11)); // NOI18N
jLabel4.setText("Повна назва");

    jLabel5.setFont(new java.awt.Font("Tahoma", 1, 11)); // NOI18N
    jLabel5.setText("Скорочена назва");

    jLabel6.setFont(new java.awt.Font("Times New Roman", 1, 18)); // NOI18N
    jLabel6.setText("Каталог постійних комісій");

    addHeadCommision.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/source/image/plus.jpg"))); // NOI18N
    addHeadCommision.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            addHeadCommisionActionPerformed(evt);
        }
    });

    removeHeadCommision.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/source/image/minus.jpg"))); // NOI18N
    removeHeadCommision.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            removeHeadCommisionActionPerformed(evt);
        }
    });

    addNamesCommision.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/source/image/plus.jpg"))); // NOI18N
    addNamesCommision.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            addNamesCommisionActionPerformed(evt);
        }
    });

    removeNamesCommision.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/source/image/minus.jpg"))); // NOI18N
    removeNamesCommision.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {

```

```

        removeNamesCommisionActionPerformed(evt);
    }
});

addAbbrevationCommision.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/source/image/plus.jpg"))); // NOI18N
addAbbrevationCommision.addActionListener(new java.awt.event.ActionListener()
{
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        addAbbrevationCommisionActionPerformed(evt);
    }
});

removeAbbrevationCommision.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/source/image/minus.jpg"))); // NOI18N
removeAbbrevationCommision.addActionListener(new
java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        removeAbbrevationCommisionActionPerformed(evt);
    }
});

javax.swing.GroupLayout settingCommitteeLayout = new
javax.swing.GroupLayout(settingCommittee.getContentPane());
settingCommittee.getContentPane().setLayout(settingCommitteeLayout);
settingCommitteeLayout.setHorizontalGroup(

settingCommitteeLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(settingCommitteeLayout.createSequentialGroup()

.addGroup(settingCommitteeLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(settingCommitteeLayout.createSequentialGroup()
        .addGroup(settingCommitteeLayout.createSequentialGroup()
            .addGroup(settingCommitteeLayout.createSequentialGroup()
                .addGroup(settingCommitteeLayout.createSequentialGroup()
                    .addComponent(addHeadCommision, javax.swing.GroupLayout.PREFERRED_SIZE, 40,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(removeHeadCommision,
javax.swing.GroupLayout.PREFERRED_SIZE, 40, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addComponent(jScrollPane2,
javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.DEFAULT_SIZE, 174,
Short.MAX_VALUE)))
            .addGroup(settingCommitteeLayout.createSequentialGroup()

```

```

        .addGap(112, 112, 112)
        .addComponent(jLabel3)))
    .addGap(18, 18, 18)

    .addGroup(settingCommitteeLayout.createParallelGroup(javax.swing.GroupLayout.Alignment
    .LEADING)

        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
settingCommitteeLayout.createSequentialGroup())

    .addGroup(settingCommitteeLayout.createParallelGroup(javax.swing.GroupLayout.Alignment
    .TRAILING, false)

        .addComponent(jScrollPane3,
javax.swing.GroupLayout.PREFERRED_SIZE, 216, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
settingCommitteeLayout.createSequentialGroup())
            .addComponent(addNamesCommision,
javax.swing.GroupLayout.PREFERRED_SIZE, 40, javax.swing.GroupLayout.PREFERRED_SIZE)

    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addComponent(renoveNamesCommision,
javax.swing.GroupLayout.PREFERRED_SIZE, 40, javax.swing.GroupLayout.PREFERRED_SIZE)))

    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 16,
Short.MAX_VALUE)
    .addGroup(settingCommitteeLayout.createParallelGroup(javax.swing.GroupLayout.Alignment
    .LEADING, false)

        .addGroup(settingCommitteeLayout.createSequentialGroup())
            .addComponent(addAbbreviationCommision,
javax.swing.GroupLayout.PREFERRED_SIZE, 40, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(removeAbbreviationCommision,
javax.swing.GroupLayout.PREFERRED_SIZE, 40, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addComponent(jScrollPane4,
javax.swing.GroupLayout.PREFERRED_SIZE, 150, javax.swing.GroupLayout.PREFERRED_SIZE)))
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
settingCommitteeLayout.createSequentialGroup())
            .addGap(61, 61, 61)
            .addComponent(jLabel4)

    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addComponent(jLabel5)
        .addGap(29, 29, 29)))
    .addGap(73, 73, 73))
    .addGroup(settingCommitteeLayout.createSequentialGroup())
        .addGap(241, 241, 241)
        .addComponent(jLabel6)

```

```

.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
    );
    settingCommitteeLayout.setVerticalGroup(

settingCommitteeLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(settingCommitteeLayout.createSequentialGroup())
        .addContainerGap()
        .addComponent(jLabel6)
        .addGap(27, 27, 27)

.addGroup(settingCommitteeLayout.createParallelGroup(javax.swing.GroupLayout.Alignment
.BASILINE)
    .addComponent(jLabel3)
    .addComponent(jLabel4)
    .addComponent(jLabel5, javax.swing.GroupLayout.PREFERRED_SIZE, 21,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(9, 9, 9)

.addGroup(settingCommitteeLayout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
    .addComponent(jScrollPane3, javax.swing.GroupLayout.DEFAULT_SIZE,
263, Short.MAX_VALUE)
    .addComponent(jScrollPane2,
javax.swing.GroupLayout.PREFERRED_SIZE, 0, Short.MAX_VALUE)
    .addComponent(jScrollPane4,
javax.swing.GroupLayout.PREFERRED_SIZE, 0, Short.MAX_VALUE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(settingCommitteeLayout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
    .addComponent(renoveNamesCommision)
    .addComponent(addAbbrevationCommision,
javax.swing.GroupLayout.PREFERRED_SIZE, 34, javax.swing.GroupLayout.PREFERRED_SIZE)
    .addComponent(removeAbbrevationCommision)
    .addComponent(addHeadCommision)
    .addComponent(removeHeadCommision)
    .addComponent(addNamesCommision,
javax.swing.GroupLayout.PREFERRED_SIZE, 34, javax.swing.GroupLayout.PREFERRED_SIZE)))
    );

newValue.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
newValue.setAlwaysOnTop(true);
newValue.setMinimumSize(new java.awt.Dimension(300, 110));
newValue.setResizable(false);
newValue.setLocationRelativeTo(null);

textNewValue.setText("jLabel8");

```

```

        createNewNotation.setText("Додати");
        createNewNotation.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                createNewNotationActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout newValueLayout = new
javax.swing.GroupLayout(newValue.getContentPane());
        newValue.getContentPane().setLayout(newValueLayout);
        newValueLayout.setHorizontalGroup(

newValueLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
newValueLayout.createSequentialGroup()
                .addGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)

.addGroup(newValueLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
)
                .addComponent(valueField, javax.swing.GroupLayout.PREFERRED_SIZE,
274, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGroup(newValueLayout.createSequentialGroup()
                    .addGap(91, 91, 91)
                    .addComponent(createNewNotation))
                .addComponent(textNewValue, javax.swing.GroupLayout.PREFERRED_SIZE, 228,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap())
            );
        newValueLayout.setVerticalGroup(

newValueLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(newValueLayout.createSequentialGroup()
                .addGap(6, 6, 6)
                .addComponent(textNewValue)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(valueField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
14, Short.MAX_VALUE)
                .addComponent(createNewNotation))
            );

        editClassifier.setAlwaysOnTop(true);
        editClassifier.setMinimumSize(new java.awt.Dimension(924, 580));
        editClassifier.setType(java.awt.Window.Type.UTILITY);

jScrollPane6.setViewportViewView(classifierTable);

```

```

saveClassifierModel.setText("Зберегти");
saveClassifierModel.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        saveClassifierModelActionPerformed(evt);
    }
});

cancelClassifierModel.setText("Відмінити");
cancelClassifierModel.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        cancelClassifierModelActionPerformed(evt);
    }
});

addRowClassifier.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/source/image/plus.jpg"))); // NOI18N
addRowClassifier.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        addRowClassifierActionPerformed(evt);
    }
});

removeRowClassifier.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/source/image/minus.jpg"))); // NOI18N
removeRowClassifier.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        removeRowClassifierActionPerformed(evt);
    }
});

javax.swing.GroupLayout editClassifierLayout = new
javax.swing.GroupLayout(editClassifier.getContentPane());
editClassifier.getContentPane().setLayout(editClassifierLayout);
editClassifierLayout.setHorizontalGroup(

editClassifierLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(editClassifierLayout.createSequentialGroup()
        .addComponent(jScrollPane6, javax.swing.GroupLayout.PREFERRED_SIZE,
904, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(0, 20, Short.MAX_VALUE)
        .addGroup(editClassifierLayout.createSequentialGroup()
            .addGap(273, 273, 273)
            .addComponent(saveClassifierModel,
javax.swing.GroupLayout.PREFERRED_SIZE, 177, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(18, 18, 18)
            .addComponent(cancelClassifierModel,
javax.swing.GroupLayout.PREFERRED_SIZE, 217, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        .addGroup(editClassifierLayout.createSequentialGroup()

```



```

        .addContainerGap()
        .addComponent(addRowClassifier,
javax.swing.GroupLayout.PREFERRED_SIZE, 40, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addComponent(removeRowClassifier,
javax.swing.GroupLayout.PREFERRED_SIZE, 35, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addContainerGap())
    );
    editClassifierLayout.setVerticalGroup(

editClassifierLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(editClassifierLayout.createSequentialGroup())
        .addContainerGap()
        .addComponent(jScrollPane6, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(editClassifierLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.L
EADING)
        .addComponent(addRowClassifier)
        .addComponent(removeRowClassifier))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
105, Short.MAX_VALUE)

.addGroup(editClassifierLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.B
ASELINE)
        .addComponent(saveClassifierModel)
        .addComponent(cancelClassifierModel)))
    );

relatedCommission.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOS
E);
relatedCommission.setAlwaysOnTop(true);
    relatedCommission.setMinimumSize(new java.awt.Dimension(855, 550));
    relatedCommission.setResizable(false);

jScrollPane5.setViewportView(relatedTable);

addCommisionState.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/source/image/plus.jpg"))); // NOI18N
    addCommisionState.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
addCommisionStateActionPerformed(evt);
        }
    });

removeCommisionState.setIcon(new

```

```

javax.swing.ImageIcon(getClass().getResource("/source/image/minus.jpg"))); // NOI18N
    removeCommisionState.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            removeCommisionStateActionPerformed(evt);
        }
    });

    saveCommisionState.setText("Зберегти");
    saveCommisionState.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            saveCommisionStateActionPerformed(evt);
        }
    });

    cancel.setText("Відмінити");
    cancel.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            cancelActionPerformed(evt);
        }
    });

    javax.swing.GroupLayout relatedCommissionLayout = new
    javax.swing.GroupLayout(relatedCommission.getContentPane());
    relatedCommission.getContentPane().setLayout(relatedCommissionLayout);
    relatedCommissionLayout.setHorizontalGroup(

relatedCommissionLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(relatedCommissionLayout.createSequentialGroup()
            .addComponent(jScrollPane5, javax.swing.GroupLayout.PREFERRED_SIZE,
835, Short.MAX_VALUE)
            .addGroup(relatedCommissionLayout.createSequentialGroup()
                .addComponent(addCommisionState,
javax.swing.GroupLayout.PREFERRED_SIZE, 40, javax.swing.GroupLayout.PREFERRED_SIZE)

                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.PREFERRED_SIZE, Short.MAX_VALUE)
                .addComponent(removeCommisionState,
javax.swing.GroupLayout.PREFERRED_SIZE, 40, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(4)
            .addGroup(relatedCommissionLayout.createSequentialGroup()
                .addGap(324, 324, 324)
                .addComponent(saveCommisionState)
                .addGap(33, 33, 33)
                .addComponent(cancel, javax.swing.GroupLayout.PREFERRED_SIZE, 86,
javax.swing.GroupLayout.PREFERRED_SIZE)

```

```

        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
    );
    relatedCommissionLayout.setVerticalGroup(

relatedCommissionLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(relatedCommissionLayout.createSequentialGroup()
            .addContainerGap()
            .addComponent(jScrollPane5, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(relatedCommissionLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)
.addComponent(removeCommisionState)
            .addComponent(addCommisionState))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
34, Short.MAX_VALUE)
.addGroup(relatedCommissionLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.BASELINE)
            .addComponent(saveCommisionState)
            .addComponent(cancel))
            .addContainerGap())
    );

frameMoreClassifier.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CL
OSE);

frameMoreClassifier.setAlwaysOnTop(true);
frameMoreClassifier.setMinimumSize(new java.awt.Dimension(472, 413));
frameMoreClassifier.setResizable(false);
frameMoreClassifier.setType(java.awt.Window.Type.UTILITY);

tableMoreClassifier.setModel(new javax.swing.table.DefaultTableModel(
    new Object [][] {

        },
    new String [] {
"Классификатор по напрямкам"
    }
) {
    Class[] types = new Class [] {
        java.lang.String.class
    };

    public Class getColumnClass(int columnIndex) {
        return types [columnIndex];
    }
});

```

```

jScrollPane7.setViewportViewView(tableMoreClassifier);

addMoreClassifier.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/source/image/plus.jpg"))); // NOI18N
addMoreClassifier.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
addMoreClassifierActionPerformed(evt);
    }
});

removeMoreClassifier.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/source/image/minus.jpg"))); // NOI18N
removeMoreClassifier.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        removeMoreClassifierActionPerformed(evt);
    }
});

saveMoreClassifier.setText("Збергти");
saveMoreClassifier.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        saveMoreClassifierActionPerformed(evt);
    }
});

cancelMoreClassifier.setText("Відмінити");
cancelMoreClassifier.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        cancelMoreClassifierActionPerformed(evt);
    }
});

javax.swing.GroupLayout frameMoreClassifierLayout = new
javax.swing.GroupLayout(frameMoreClassifier.getContentPane());
frameMoreClassifier.getContentPane().setLayout(frameMoreClassifierLayout);
frameMoreClassifierLayout.setHorizontalGroup(

frameMoreClassifierLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(frameMoreClassifierLayout.createSequentialGroup()
        .addComponent(jScrollPane7)
        .addGroup(frameMoreClassifierLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(addMoreClassifier, javax.swing.GroupLayout.PREFERRED_SIZE, 34,
javax.swing.GroupLayout.PREFERRED_SIZE)

```

```

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addComponent(removeMoreClassifier,
javax.swing.GroupLayout.PREFERRED_SIZE, 37, javax.swing.GroupLayout.PREFERRED_SIZE)))
        .addContainerGap())
        .addGroup(frameMoreClassifierLayout.createSequentialGroup()
        .addGap(126, 126, 126)
        .addComponent(saveMoreClassifier)
        .addGap(48, 48, 48)
        .addComponent(cancelMoreClassifier,
javax.swing.GroupLayout.PREFERRED_SIZE, 87, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );
        frameMoreClassifierLayout.setVerticalGroup(

frameMoreClassifierLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

        .addGroup(frameMoreClassifierLayout.createSequentialGroup()
        .addContainerGap()
        .addComponent(jScrollPane7, javax.swing.GroupLayout.PREFERRED_SIZE,
280, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(frameMoreClassifierLayout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.BASELINE)

                .addComponent(addMoreClassifier)
                .addComponent(removeMoreClassifier))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 49,
Short.MAX_VALUE)

        .addGroup(frameMoreClassifierLayout.createParallelGroup(javax.swing.GroupLayout.Alignm
ent.BASELINE)

                .addComponent(saveMoreClassifier)
                .addComponent(cancelMoreClassifier))
        .addContainerGap())
        );

frameMoreClassifier1.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_C
LOSE);

        frameMoreClassifier1.setAlwaysOnTop(true);
        frameMoreClassifier1.setMinimumSize(new java.awt.Dimension(472, 413));
        frameMoreClassifier1.setResizable(false);
        frameMoreClassifier1.setType(java.awt.Window.Type.UTILITY);

        tableMoreClassifier1.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {

                },

```

```

        new String [] {
            "Класифікатор по категорії"
        }
    ) {
        Class[] types = new Class [] {
            java.lang.String.class
        };

        public Class getColumnClass(int columnIndex) {
            return types [columnIndex];
        }
    });
    jScrollPane8.setViewportViewView(tableMoreClassifier1);

    addMoreClassifier1.setIcon(new
    javax.swing.ImageIcon(getClass().getResource("/source/image/plus.jpg"))); // NOI18N
    addMoreClassifier1.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            addMoreClassifier1ActionPerformed(evt);
        }
    });

    removeMoreClassifier1.setIcon(new
    javax.swing.ImageIcon(getClass().getResource("/source/image/minus.jpg"))); // NOI18N
    removeMoreClassifier1.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            removeMoreClassifier1ActionPerformed(evt);
        }
    });

    saveMoreClassifier1.setText("Зберегти");
    saveMoreClassifier1.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            saveMoreClassifier1ActionPerformed(evt);
        }
    });

    cancelMoreClassifier1.setText("Відмінити");
    cancelMoreClassifier1.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            cancelMoreClassifier1ActionPerformed(evt);
        }
    });

    javax.swing.GroupLayout frameMoreClassifier1Layout = new
    javax.swing.GroupLayout(frameMoreClassifier1.getContentPane());
    frameMoreClassifier1.getContentPane().setLayout(frameMoreClassifier1Layout);
    frameMoreClassifier1Layout.setHorizontalGroup(

```

```

frameMoreClassifier1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

        .addGroup(frameMoreClassifier1Layout.createSequentialGroup()

            .addContainerGap()


        .addGroup(frameMoreClassifier1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

            .addComponent(jScrollPane8)

            .addGroup(frameMoreClassifier1Layout.createSequentialGroup()

                .addComponent(addMoreClassifier1,
javax.swing.GroupLayout.PREFERRED_SIZE, 34, javax.swing.GroupLayout.PREFERRED_SIZE)

            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

                .addComponent(removeMoreClassifier1,
javax.swing.GroupLayout.PREFERRED_SIZE, 37, javax.swing.GroupLayout.PREFERRED_SIZE)))

            .addContainerGap()

        .addGroup(frameMoreClassifier1Layout.createSequentialGroup()

            .addGap(126, 126, 126)

            .addComponent(saveMoreClassifier1)

            .addGap(48, 48, 48)

            .addComponent(cancelMoreClassifier1,
javax.swing.GroupLayout.PREFERRED_SIZE, 87, javax.swing.GroupLayout.PREFERRED_SIZE)

            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))

        );

frameMoreClassifier1Layout.setVerticalGroup(

frameMoreClassifier1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

        .addGroup(frameMoreClassifier1Layout.createSequentialGroup()

            .addContainerGap()

            .addComponent(jScrollPane8, javax.swing.GroupLayout.PREFERRED_SIZE,
280, javax.swing.GroupLayout.PREFERRED_SIZE)

            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

        .addGroup(frameMoreClassifier1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

            .addComponent(addMoreClassifier1)

            .addComponent(removeMoreClassifier1))

            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
49, Short.MAX_VALUE)

        .addGroup(frameMoreClassifier1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

            .addComponent(saveMoreClassifier1)

            .addComponent(cancelMoreClassifier1))

            .addContainerGap()

        );

```

```

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
setTitle("Миколаївська обласна рада. Контроль рішень");
setForeground(new java.awt.Color(255, 255, 255));

jTable1.setMaximumSize(size());
jTable1.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
jTable1MouseClicked(evt);
    }
    public void mouseEntered(java.awt.event.MouseEvent evt) {
        jTable1MouseEntered(evt);
    }
});
jScrollPane1.setViewportViewView(jTable1);

searchParametrValue.setEnabled(false);
searchParametrValue.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        searchParametrValueActionPerformed(evt);
    }
});

searchParametrText.addFocusListener(new java.awt.event.FocusAdapter() {
    public void focusGained(java.awt.event.FocusEvent evt) {
        searchParametrTextFocusGained(evt);
    }
});

jButton1.setText("Загальний звіт");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});

jButton2.setText("Звіт за таблицю");
jButton2.setInheritsPopupMenu(true);
jButton2.setMaximumSize(new java.awt.Dimension(0, 0));
jButton2.setMinimumSize(new java.awt.Dimension(0, 0));
jButton2.setPreferredSize(new java.awt.Dimension(0, 0));
jButton2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton2ActionPerformed(evt);
    }
});

jTable2.setMaximumSize(size());
jTable2.addMouseListener(new java.awt.event.MouseAdapter() {

```



```

        public void mouseClicked(java.awt.event.MouseEvent evt) {
            jTable2MouseClicked(evt);
        }
        public void mouseEntered(java.awt.event.MouseEvent evt) {
            jTable2MouseEntered(evt);
        }
    });
    jScrollPane9.setViewportViewView(jTable2);

    jButton3.setText("Оновити");
    jButton3.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton3ActionPerformed(evt);
        }
    });

    mainMenu.setText("Виконати");

    seachMenu.setText("Пошук");
    seachMenu.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            seachMenuActionPerformed(evt);
        }
    });
    mainMenu.add(seachMenu);

    exitMenu.setText("Вихід");
    exitMenu.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            exitMenuActionPerformed(evt);
        }
    });
    mainMenu.add(exitMenu);

    jMenuBar1.add(mainMenu);

    cardDecisionMenu.setText("Картка рішення");

    createCardDecision.setText("Нове рішення");
    createCardDecision.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            createCardDecisionActionPerformed(evt);
        }
    });
    cardDecisionMenu.add(createCardDecision);

    openCardDecision.setText("Редагувати картку рішення");
    openCardDecision.setToolTipText("");
    openCardDecision.addActionListener(new java.awt.event.ActionListener() {

```

```

        public void actionPerformed(java.awt.event.ActionEvent evt) {
            openCardDecisionActionPerformed(evt);
        }
    });
    cardDecisionMenu.add(openCardDecision);

    DeleteCardDecision1.setText("Видалити картку рішення");
    DeleteCardDecision1.setToolTipText("");
    DeleteCardDecision1.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            DeleteCardDecision1ActionPerformed(evt);
        }
    });
    cardDecisionMenu.add(DeleteCardDecision1);

jMenuBar1.add(cardDecisionMenu);

    optionMenu.setText("Постійні комісії");

    jMenuItem1.setText("Каталог постійних комісій");
    jMenuItem1.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jMenuItem1ActionPerformed(evt);
        }
    });
    optionMenu.add(jMenuItem1);

    jMenuItem3.setText("Облікова карта постійних комісій");
    jMenuItem3.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jMenuItem3ActionPerformed(evt);
        }
    });
    optionMenu.add(jMenuItem3);

    jMenuItem5.setText("Класифікатор по категорії");
    jMenuItem5.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jMenuItem5ActionPerformed(evt);
        }
    });
    optionMenu.add(jMenuItem5);

    jMenuItem2.setText("Профіль постійної комісії");
    jMenuItem2.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jMenuItem2ActionPerformed(evt);
        }
    });

```

[illegible]

```

        .addComponent(jButton1))))
        .addContainerGap()
    );
    layout.setVerticalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
    layout.createSequentialGroup())
        .addContainerGap()

    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(jButton1)
        .addGroup(layout.createSequentialGroup()
            .addGap(4, 4, 4)
            .addComponent(seachParametrText,
    javax.swing.GroupLayout.PREFERRED_SIZE, 16, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addComponent(seachParametrValue,
    javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
        .addComponent(jButton2, javax.swing.GroupLayout.PREFERRED_SIZE,
    21, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(jButton3))
        .addGap(6, 6, 6)
    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(jScrollPane9, javax.swing.GroupLayout.DEFAULT_SIZE,
    264, Short.MAX_VALUE)
        .addComponent(jScrollPane1,
    javax.swing.GroupLayout.PREFERRED_SIZE, 0, Short.MAX_VALUE))
        .addContainerGap()
    );

    pack();
} // </editor-fold>

/**
 * Открывает окно поиска в базе данных по критериям
 *
 * @param evt
 */
private void seachMenuActionPerformed(java.awt.event.ActionEvent evt) {
listDateOption.setSelectedIndex(-1);
    listConvocation.setEnabled(false);
    listSession.setEnabled(false);
    listClassifier.setEnabled(false);
    listAddClassifier.setEnabled(false);
    listState.setEnabled(false);
    listCommision.setEnabled(false);

```

```

        seachFrame.setVisible(true);
    }

    private void listYearActionPerformed(java.awt.event.ActionEvent evt) {
        Object selectedItem = listYear.getSelectedItemAt();
        String search = (String) selectedItem;
        try {
listDateOption.setModel(dbmf.getOptionDate(search));
        } catch (SQLException ex) {
            Logger.getLogger(MainFrame.class.getName()).log(Level.SEVERE, null, ex);
        }
        isSelectedDateOption = false;
listDateOption.setEnabled(true);
    }

    private void jTable1MouseClicked(java.awt.event.MouseEvent evt) {
    }

    private void jTable1MouseEntered(java.awt.event.MouseEvent evt) {
    }
/**
 * Метод для поиска и создания таблицы по критериям которые были выбраны.С
 * начала проверяется какой JComboBox isEnabled() возвращает true.После
 * этого записывается 2 переменные name и value.Где name это имя объекта а
 * value это значение которое было выбранно в JComboBox.Далее идёт проверка
 * на пустую строку и null.В случае если всё хорошо в классе
 * DatabaseMainFraim вызывается метод getTable(String name,Object value)
 * который возвращает объект коллекции типа ArrayList<Object[]> и создаётся
 * таблица по критериям поиска.
 *
 * @param evt
 */
    private void seachDatabaseActionPerformed(java.awt.event.ActionEvent evt) {
        String name = null;

        Object value = null;

        if (listDateOption.getSelectedItemAt() != null) {
            // System.out.println(listDateOption.getSelectedItemAt().toString());
            listDateOption.setEnabled(false);
        }
        seachConvocation.setSelected(false);
        seachSession.setSelected(false);
        seachClassifier.setSelected(false);
seachAddClassifier.setSelected(false);
        seachState.setSelected(false);
        seachCommision.setSelected(false);

//        else {

```

```

//      listDateOption.getSelectedItemAt() ;
//      }
for (JComboBox jComboBox : comboBoxes) {
    if (jComboBox.isEnabled()) {
        name = jComboBox.getName();
        value = jComboBox.getSelectedItemAt();
    }
}
if (value == null) {
    JOptionPane.showMessageDialog(this, "Выберите параметр для поиска",
"Ошибка работы", MessageType.ERROR.ordinal());
}
try {
    seachParametrValue.setVisible(true);
    seachParametrText.setText(name);
    if (isSelectedDateOption) {

        String month = (String) listDateOption.getSelectedItemAt();
        String year = (String) listYear.getSelectedItemAt();
seachParametrValue.setText(month + "." + year);

    } else {
        seachParametrValue.setText((String) value);
    }
} catch (ClassCastException ex) {
    Integer a = (Integer) value;
seachParametrValue.setText(String.valueOf(a));
}
try {
    seachFrame.dispose();
    ArrayList<Object[]> list = null;

    if (isSelectedDateOption) {
        String month = (String) listDateOption.getSelectedItemAt();
        String year = (String) listYear.getSelectedItemAt();
        list = dbmf.getTable(name, value, month + "." + year);
    } else {
        list = dbmf.getTable(name, value, null);
    }
    ///
    reportClassifier = value;
    reportName = name;
    ///

    if (tableModel == null) {
initTable();

        // initNumberTable(jTable1.getRowCount());
    }
}

```

```

        cleanRowTable();
        //-----//
        int[] tmpArrIndexes = new int[list.size()];
        for (int i = 0, counter = 1; i < tmpArrIndexes.length; i++, counter++) {
            tmpArrIndexes[i] = counter;
        }

        for (Object[] objects : list) {
            addRowToTable(objects);
        }

        initNumberTable(list.size());
        tempObjs = list;
        //-----//
    } catch (Exception ex) {
        ex.printStackTrace();
        JOptionPane.showMessageDialog(this, "Сбой работы с базой данных,
        перезапустите приложения и попробуйте снова", "Ошибка работы с базой данных",
        MessageType.ERROR.ordinal());
    }
}

/**
 * Метод который открывает карту решения.В данном случае карта будет пустой
 * для заполнения.
 *
 * @see CardDecision
 * @param evt
 */
private void createCardDecisionActionPerformed(java.awt.event.ActionEvent evt) {
    setEnabled(false);
    CardDecision cardDecision = new CardDecision();
    cardDecision.openCardDecision();
    setEnabled(true);
}

/**
 * Метод выбора компонента JRadioButton.Если компонент был вызван то
 * соответствующий список обращается в базу данных и заполняет JComboBox В
 * данном методе работает 2 объекта <seachConvocation> и <listConvocation>
 *
 * @param evt
 */
private void seachConvocationActionPerformed(java.awt.event.ActionEvent evt) {
    if (seachConvocation.isSelected()) {
        try {
            conntrolerViewRadioButton(seachConvocation);
            conntrolerViewComboBox(listConvocation);
            listConvocation.setModel(dbmf.getConvocation());
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }
}

```

```

        JOptionPane.showMessageDialog(this, "Сбой работы с базой данных,
        перезапустите приложения и попробуйте снова", "Ошибка работы с базой данных",
        MessageType.ERROR.ordinal());
    }
}
/**
 * Метод выбора компонента JRadioButton.Если компонент был вызван то
 * соответствующий список обращается в базу данных и заполняет JComboBox В
 * данном методе работает 2 объекта <seachSession> и <listSession>
 *
 * @param evt
 */
private void seachSessionActionPerformed(java.awt.event.ActionEvent evt) {
    if (seachSession.isSelected()) {
        conntrolerViewRadioButton(seachSession);
        conntrolerViewComboBox(listSession);
        try {
            listSession.setModel(dbmf.getSessionNumber());
        } catch (SQLException ex) {
            ex.printStackTrace();
            JOptionPane.showMessageDialog(this, "Сбой работы с базой данных,
            перезапустите приложения и попробуйте снова", "Ошибка работы с базой данных",
            MessageType.ERROR.ordinal());
        }
    }
}
/**
 * Метод выбора компонента JRadioButton.Если компонент был вызван то
 * соответствующий список обращается в базу данных и заполняет JComboBox В
 * данном методе работает 2 объекта <seachClassifier> и <listClassifier>
 *
 * @param evt
 */
private void seachClassifierActionPerformed(java.awt.event.ActionEvent evt) {
    if (seachClassifier.isSelected()) {
        conntrolerViewRadioButton(seachClassifier);
        conntrolerViewComboBox(listClassifier);
        try {
            listClassifier.setModel(dbmf.getClassifier());
        } catch (SQLException ex) {
            ex.printStackTrace();
            JOptionPane.showMessageDialog(this, "Сбой работы с базой данных,
            перезапустите приложения и попробуйте снова", "Ошибка работы с базой данных",
            MessageType.ERROR.ordinal());
        }
    }
}

```



```

* Метод выбора компонента JRadioButton.Если компонент был вызван то
* соответствующий список обращается в базу данных и заполняет JComboBox В
* данном методе работает 2 объекта <seachState> и <listState>
*
* @param evt
*/
private void seachStateActionPerformed(java.awt.event.ActionEvent evt) {
    if (seachState.isSelected()) {
        conntrolerViewRadioButton(seachState);
        conntrolerViewComboBox(listState);
        try {
            listState.setModel(dbmf.getState());
        } catch (SQLException ex) {
            ex.printStackTrace();
            JOptionPane.showMessageDialog(this, "Сбой работы с базой данных,
перезапустите приложения и попробуйте снова", "Ошибка работы с базой данных",
MessageType.ERROR.ordinal());
        }
    }
}
/**
* Метод выбора компонента JRadioButton.Если компонент был вызван то
* соответствующий список обращается в базу данных и заполняет JComboBox В
* данном методе работает 2 объекта <seachCommision> и <listCommision>
*
* @param evt
*/
private void seachCommisionActionPerformed(java.awt.event.ActionEvent evt) {
    if (seachCommision.isSelected()) {
        conntrolerViewRadioButton(seachCommision);
        conntrolerViewComboBox(listCommision);
        try {
            listCommision.setModel(dbmf.getCommision());
        } catch (SQLException ex) {
            ex.printStackTrace();
            JOptionPane.showMessageDialog(this, "Сбой работы с базой данных,
перезапустите приложения и попробуйте снова", "Ошибка работы с базой данных",
MessageType.ERROR.ordinal());
        }
    }
}
/**
* Метод выбора компонента JRadioButton.Если компонент был вызван то
* соответствующий список обращается в базу данных и заполняет JComboBox В
* данном методе работает 2 объекта <seachYear> и <listYear>
*
* @param evt
*/
private void seachYearActionPerformed(java.awt.event.ActionEvent evt) {

```

```

        if (seachYear.isSelected()) {
            conntrolerViewRadioButton(seachYear);
            conntrolerViewComboBox(listYear);
            try {
                listYear.setModel(dbmf.getListYear());
            } catch (SQLException ex) {
                ex.printStackTrace();
                JOptionPane.showMessageDialog(this, "Сбой работы с базой данных,
перезапустите приложения и попробуйте снова", "Ошибка работы с базой данных",
MessageType.ERROR.ordinal());
            }
        }
    }
}

/**
 * Открыте выбранной карты решения, естественно куча магии но вкратце мы
 * берём значения id с таблицы, которую выбрали, далее мы записываем её в
 * переменную и вызываем метод, который собирает всю информацию про карточку
 * с указанным ID.Далее мы открываем фрейм карты решения и передаём ей
 * ValueObject с заполненными данными про уже существующую карту решения.
 *
 * @param evt
 */
// ListSelectionListener lslistener = new ListSelectionListener() {
//     @Override
//     public void valueChanged(ListSelectionEvent e) {
//         Object idSelectedItem = null;
//         idSelectedItem = jTable1.getValueAt(jTable1.getSelectedRow(), 0);
//         MainFrame.idRow = Integer.parseInt(idSelectedItem.toString());
//     }
// };

private void openCardDecisionActionPerformed(java.awt.event.ActionEvent evt) {
    Object idSelectedItem = null;
    try {
        int row = jTable1.getSelectedRow();
        idSelectedItem = tableModel.getValueAt(row, 0);
    } catch (NullPointerException ex) {
        JOptionPane.showMessageDialog(this, "Не выбрана существующия карта решения", "Ошибка
работы", MessageType.ERROR.ordinal());
    }
    try {
        InformationCard card = dbmf.getInformationCard((int) idSelectedItem);
        CardDecision cardDecision = new CardDecision();
        cardDecision.openCardDecision(card);
    } catch (SQLException ex) {
        JOptionPane.showMessageDialog(this, "Сбой работы с базой данных,
перезапустите приложения и попробуйте снова", "Ошибка работы с базой данных",

```

```

MessageType.ERROR.ordinal());
        ex.printStackTrace();
    }
}

/**
 * Открытие фрейма редактирования постоянных комиссий.
 *
 * @param evt
 */
private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {
    initListCommision();
    listNamesCommision.setModel(defaultListNames);
    listHeadCommision.setModel(defaultListHead);
    listAbbreviation.setModel(defaultListAbbreviation);
    settingCommittee.setVisible(true);
}

/**
 * Добавления новой строки в головы комиссий Переменной chooseAction
 * присваивается значения 0.
 *
 * @param evt
 */
private void addHeadCommisionActionPerformed(java.awt.event.ActionEvent evt) {
    valueField.setText("");
    chooseAction = 0;
    textNewValue.setText("Вкажіть нову голову комісії");
    newValue.setVisible(true);
}

/**
 * Удаление выбранного элемента со списка и таблицы голов комиссий
 *
 * @param evt
 */
private void removeHeadCommisionActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        String deletedValue = (String) listHeadCommision.getSelectedValue();
        dbmf.deleteCommisionHead(deletedValue);
        defaultListHead.removeElement(deletedValue);
    } catch (SQLException ex) {
        JOptionPane.showMessageDialog(this, "Сбой работы с базой данных,
        перезапустите приложения и попробуйте снова", "Ошибка работы с базой данных",
        MessageType.ERROR.ordinal());
        ex.printStackTrace();
    }
}

/**
 * Добавления новой строки в имена комиссий Переменной chooseAction
 * присваивается значения 1.
 *

```

```

    * @param evt
    */
    private void addNamesCommisionActionPerformed(java.awt.event.ActionEvent evt) {
        valueField.setText("");
        chooseAction = 1;
        textNewValue.setText("Вкажіть нове ім'я комісії");
        newValue.setVisible(true);
    }
    /**
     * Удаление выбранного элемента со списка и таблицы имён комиссий
     *
     * @param evt
     */
    private void removeNamesCommisionActionPerformed(java.awt.event.ActionEvent evt) {
        try {
            String deletedValue = (String) listNamesCommision.getSelectedValue();
            dbmf.deleteCommisionName(deletedValue);
            defaultListNames.removeElement(deletedValue);
        } catch (SQLException ex) {
            ex.printStackTrace();
            JOptionPane.showMessageDialog(this, "Сбой работы с базой данных,
перезапустите приложения и попробуйте снова", "Ошибка работы с базой данных",
MessageType.ERROR.ordinal());
        }
    }
    /**
     * Добавления новой строки в Аббревиатуры комиссий Переменной chooseAction
     * присваивается значения 2.
     *
     * @param evt
     */
    private void addAbbrevationCommisionActionPerformed(java.awt.event.ActionEvent
evt) {
        valueField.setText("");
        chooseAction = 2;
        textNewValue.setText("Вкажіть нову аббревіатуру комісії");
        newValue.setVisible(true);
    }
    /**
     * Удаление выбранного элемента со списка и таблицы аббревиатур комиссий
     *
     * @param evt
     */
    private void removeAbbrevationCommisionActionPerformed(java.awt.event.ActionEvent
evt) {
        try {
            String deletedValue = (String) listAbbreviation.getSelectedValue();
            dbmf.deleteCommisionAbbreviation(deletedValue);
            defaultListAbbreviation.removeElement(deletedValue);
        }
    }

```

```

        } catch (SQLException ex) {
            ex.printStackTrace();
            JOptionPane.showMessageDialog(this, "Сбой работы с базой данных,
перезапустите приложения и попробуйте снова", "Ошибка работы с базой данных",
MessageType.ERROR.ordinal());
        }
    }

    /**
     * Метод который добавляет в соответствующий список новое значения, нужный
     * список проверяется в switch на переменную chooseAction.
     *
     * @param evt
     */
    private void createNewNotationActionPerformed(java.awt.event.ActionEvent evt) {
        try {
            String value = valueField.getText();
            if (value.equals("")) {
                JOptionPane.showMessageDialog(this, "Вы не ввели новое значения для
таблицы", "Ошибка", MessageType.ERROR.ordinal());
                return;
            }
            switch (chooseAction) {
                case 0:
                    dbmf.addCommisionHead(value);
                    defaultListHead.addElement(value);
                    break;
                case 1:
                    dbmf.addCommisionName(value);
                    defaultListNames.addElement(value);
                    break;
                case 2:
                    dbmf.addCommisionAbbreviation(value);
                    defaultListAbbreviation.addElement(value);
                    break;
            }
        } catch (SQLException ex) {
            ex.printStackTrace();
            JOptionPane.showMessageDialog(this, "Сбой работы с базой данных,
перезапустите приложения и попробуйте снова", "Ошибка работы с базой данных",
MessageType.ERROR.ordinal());
        }
        newValue.dispose();
    }

    private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {
        if (classifierTableModel == null) {
            initEditClassifier();
        }
        editClassifier.setVisible(true);
    }

```

```

    }

    private void addRowClassifierActionPerformed(java.awt.event.ActionEvent evt) {
        //ADD Row
        classifierTableModel.addRow(new Object[]{"", "", "", ""});
        countNewClassifierState.add(classifierTable.getRowCount());
    }

    private void removeRowClassifierActionPerformed(java.awt.event.ActionEvent evt) {
        try {
            //Remove Row
            int selectionRow = classifierTable.getSelectedRow();
            if (countNewClassifierState.contains(selectionRow)) {
                classifierTableModel.removeRow(selectionRow);
                countNewClassifierState.remove(selectionRow);
                return;
            }
            String classifier = (String) classifierTable.getValueAt(selectionRow, 0);
            String commision_1 = (String) classifierTable.getValueAt(selectionRow, 1);
            String commision_2 = (String) classifierTable.getValueAt(selectionRow, 2);
            String commision_3 = (String) classifierTable.getValueAt(selectionRow, 3);
            String[] arrayValues = new String[]{
                classifier, commision_1, commision_2, commision_3
            };
            checkToIncorrectData(arrayValues);
            if (!arrayValues[0].isEmpty()) {
                dbmf.deleteClassifierState(arrayValues);
            }
            classifierTableModel.removeRow(selectionRow);
        } catch (SQLException ex) {
            ex.printStackTrace();
            JOptionPane.showMessageDialog(this, "Сбой работы с базой данных, перезапустите приложения и попробуйте снова", "Ошибка работы с базой данных",
                MessageType.ERROR.ordinal());
        }
    }

    private void saveClassifierModelActionPerformed(java.awt.event.ActionEvent evt) {
        if (!countNewClassifierState.isEmpty()) {
            for (Integer index : countNewClassifierState) {
                try {
                    Object[] arrayData = new Object[4];
                    arrayData[0] = classifierTableModel.getValueAt(index - 1, 0);
                    arrayData[1] = classifierTableModel.getValueAt(index - 1, 1);
                    arrayData[2] = classifierTableModel.getValueAt(index - 1, 2);
                    arrayData[3] = classifierTableModel.getValueAt(index - 1, 3);
                    dbmf.addClassifierState(arrayData);
                } catch (SQLException ex) {

```

```

        System.out.println(ex);
        //JOptionPane.showMessageDialog(this, "Сбой работы с базой данных,
перезапустите приложения и попробуйте снова", "Ошибка работы с базой данных",
MessageType.ERROR.ordinal());
    }
}

countNewClassifierState.clear();
editClassifier.dispose();
}

private void jMenuItem3ActionPerformed(java.awt.event.ActionEvent evt) {
    relatedCommission.setVisible(true);
    if (relatedTableModel == null) {
        initRelatedTable();
    }
}

private void addCommisionStateActionPerformed(java.awt.event.ActionEvent evt) {
    relatedTableModel.addRow(new Object[]{null, null, null});
    countNewCommision.add(relatedTable.getRowCount());
}

private void saveCommisionStateActionPerformed(java.awt.event.ActionEvent evt) {
    if (!countNewCommision.isEmpty()) {
        for (Integer index : countNewCommision) {
            try {
Object[] arrayData = new Object[3];
                arrayData[0] = relatedTableModel.getValueAt(index - 1, 0);
                arrayData[1] = relatedTableModel.getValueAt(index - 1, 1);
                arrayData[2] = relatedTableModel.getValueAt(index - 1, 2);
                dbmf.saveRelatedCommision(arrayData);
            } catch (SQLException ex) {
                System.out.println(ex);
                //JOptionPane.showMessageDialog(this, "Сбой работы с базой данных,
перезапустите приложения и попробуйте снова", "Ошибка работы с базой данных",
MessageType.ERROR.ordinal());
            }
        }
    }

    countNewCommision.clear();
    relatedCommission.dispose();
}

private void removeCommisionStateActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        int selectedItem = relatedTable.getSelectedRow();
        if (countNewCommision.contains(selectedItem)) {
            relatedTableModel.removeRow(selectedItem);
        }
    }
}

```

```

        countNewCommision.remove(selectedItem);
        return;
    }
    String headCommision = (String) relatedTableModel.getValueAt(selectedItem,
0);
    String nameCommision = (String) relatedTableModel.getValueAt(selectedItem,
1);
    String abbreviation = (String) relatedTableModel.getValueAt(selectedItem,
2);

    String[] arrayData = new String[]{headCommision, nameCommision,
abbreviation};
    relatedTableModel.removeRow(selectedItem);
    dbmf.removeRelatedCommision(arrayData);
} catch (SQLException ex) {
    System.out.println(ex);
    //JOptionPane.showMessageDialog(this, "Сбой работы с базой данных,
перезапустите приложения и попробуйте снова", "Ошибка работы с базой данных",
MessageType.ERROR.ordinal());
}
}

private void cancelMoreClassifierActionPerformed(java.awt.event.ActionEvent evt) {
    frameMoreClassifier.dispose();
}

private void jMenuItem4ActionPerformed(java.awt.event.ActionEvent evt) {
    if (moreClassifierDefaultModel == null) {
        initMoreTable();
    }
    frameMoreClassifier.setVisible(true);
}

private void addMoreClassifierActionPerformed(java.awt.event.ActionEvent evt) {
    moreClassifierDefaultModel.addRow(new Object[]{null});
    countNewMoreClassifier.add(tableMoreClassifier.getRowCount());
}

private void saveMoreClassifierActionPerformed(java.awt.event.ActionEvent evt) {
    if (!countNewMoreClassifier.isEmpty()) {
        for (Integer index : countNewMoreClassifier) {
            try {
                Object arrayData = moreClassifierDefaultModel.getValueAt(index -
1, 0);

                if (arrayData == null) {
                    return;
                }
                dbmf.saveMoreCommision(arrayData);
            } catch (SQLException ex) {
                Logger.getLogger(MainFrame.class.getName()).log(Level.SEVERE,

```



```

null, ex);
        }
    }
    countNewMoreClassifier.clear();
    frameMoreClassifier.dispose();
}

private void removeMoreClassifierActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        int selectedIndex = tableMoreClassifier.getSelectedRow();
        if (countNewMoreClassifier.contains(selectedIndex - 1)) {
            moreClassifierDefaultModel.removeRow(selectedIndex);
            countNewMoreClassifier.remove(selectedIndex);
            return;
        }

        Object data = moreClassifierDefaultModel.getValueAt(selectedIndex, 0);
        dbmf.removeMoreClassifier(data);
        moreClassifierDefaultModel.removeRow(selectedIndex);
    } catch (SQLException ex) {
        ex.printStackTrace();
        JOptionPane.showMessageDialog(this, "Сбой работы с базой данных,
перезапустите приложения и попробуйте снова", "Ошибка работы с базой данных",
MessageType.ERROR.ordinal());
    }
}

private void seachParametrTextFocusGained(java.awt.event.FocusEvent evt) {
}

private void exitMenuActionPerformed(java.awt.event.ActionEvent evt) {
    System.exit(0);
}

private void cancelClassfierModelActionPerformed(java.awt.event.ActionEvent evt) {
    editClassifier.dispose();
}

private void cancelActionPerformed(java.awt.event.ActionEvent evt) {
    relatedCommission.dispose();
}

private void seachParametrValueActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void seachAddClassifierActionPerformed(java.awt.event.ActionEvent evt) {
    if (seachAddClassifier.isSelected()) {
        conntrolerViewRadioButton(seachAddClassifier);
    }
}

```

```

        conntrolerViewComboBox(listAddClassifier);
    try {
        listAddClassifier.setModel(dbmf.getAddClassifier());
    } catch (SQLException ex) {
        ex.printStackTrace();
        JOptionPane.showMessageDialog(this, "Сбой работы с базой данных,
        перезапустите приложения и попробуйте снова", "Ошибка работы с базой данных",
        MessageType.ERROR.ordinal());
    }
}

private void addMoreClassifierActionPerformed(java.awt.event.ActionEvent evt) {
    classifierTableEdit.addRow(new Object[]{null});
    countNewClassifier.add(classifierTableEdit.getRowCount());
}

private void removeMoreClassifierActionPerformed(java.awt.event.ActionEvent evt)
{
    try {
        int selectedIndex = tableMoreClassifier1.getSelectedRow();
        if (countNewClassifier.contains(selectedIndex - 1)) {
            classifierTableEdit.removeRow(selectedIndex);
            countNewClassifier.remove(selectedIndex);
            return;
        }

        Object data = classifierTableEdit.getValueAt(selectedIndex, 0);
        dbmf.removeClassifier(data);
        classifierTableEdit.removeRow(selectedIndex);
    } catch (SQLException ex) {
        ex.printStackTrace();
        JOptionPane.showMessageDialog(this, "Сбой работы с базой данных,
        перезапустите приложения и попробуйте снова", "Ошибка работы с базой данных",
        MessageType.ERROR.ordinal());
    }
}

private void saveMoreClassifierActionPerformed(java.awt.event.ActionEvent evt) {
    if (!countNewClassifier.isEmpty()) {
        for (Integer index : countNewClassifier) {
            try {
                Object arrayData = tableMoreClassifier1.getValueAt(index - 1, 0);
                if (arrayData == null) {
                    return;
                }
                dbmf.saveClassifier(arrayData);
            } catch (SQLException ex) {
                Logger.getLogger(MainFrame.class.getName()).log(Level.SEVERE,
                null, ex);
            }
        }
    }
}

```

```

        }
    }
}
countNewClassifier.clear();
frameMoreClassifier1.dispose();
}

private void cancelMoreClassifier1ActionPerformed(java.awt.event.ActionEvent evt)
{
    frameMoreClassifier1.dispose();
}

private void jMenuItem5ActionPerformed(java.awt.event.ActionEvent evt) {
    if (classifierTableEdit == null) {
        initEditClassifierOne();
    }
    frameMoreClassifier1.setVisible(true);
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        Excel.Excel3.main(null);
    } catch (IOException ex) {
        Logger.getLogger(MainFrame.class.getName()).log(Level.SEVERE, null, ex);
    } catch (SQLException ex) {
        Logger.getLogger(MainFrame.class.getName()).log(Level.SEVERE, null, ex);
    }
    try {
        DataBaseExelReport baseExelReport = new DataBaseExelReport();
baseExelReport.search();
    } catch (SQLException ex) {
        Logger.getLogger(MainFrame.class.getName()).log(Level.SEVERE, null, ex);
    }
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

    try {
        ExcelJTable.createxls(tempObjs, reportClassifier);
    } catch (IOException ex) {
        Logger.getLogger(MainFrame.class.getName()).log(Level.SEVERE, null, ex);
    } catch (SQLException ex) {
        JOptionPane.showMessageDialog(this, "Таблица пуста", "Выберете в поиске критерий для наполнения таблицы", MessageType.ERROR.ordinal());
    }
}
}

```

```

private void listSessionActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void listClassifierActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void listAddClassifierActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void listDateOptionActionPerformed(java.awt.event.ActionEvent evt) {
    isSelectedDateOption = true;

}

private void jTable2MouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
}

private void jTable2MouseEntered(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
}

private void DeleteCardDecision1ActionPerformed(java.awt.event.ActionEvent evt) {
    Object idSelectedItem = null;
    try {
        int row = jTable1.getSelectedRow();
        idSelectedItem = tableModel.getValueAt(row, 0);
    } catch (NullPointerException ex) {
        JOptionPane.showMessageDialog(this, "Не выбрана существующия карта
решения", "Ошибка работы", MessageType.ERROR.ordinal());
    }
    try {
        String msg = "Картка рішення видалена";
        JOptionPane.showMessageDialog(null, msg, "Інформація",
JOptionPane.OK_CANCEL_OPTION);
        dbmf.deleterow((int) idSelectedItem);
    } catch (SQLException ex) {
        Logger.getLogger(MainFrame.class.getName()).log(Level.SEVERE, null, ex);
    }
}

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    String name = reportName;

    Object value = reportClassifier;

```

```

//      for (JComboBox jComboBox : comboBoxes) {
//          if (jComboBox.isEnabled()) {
//              name = jComboBox.getName();
//              value = jComboBox.getSelectedItem();
//          }
//      }
//      if (value == null) {
//          JOptionPane.showMessageDialog(this, "Выберите параметр для поиска",
"Ошибка работы", MessageType.ERROR.ordinal());
//      }
try {
    seachFrame.dispose();
    ArrayList<Object[]> list = null;

    if (isSelectedDateOption) {
        String month = (String) listDateOption.getSelectedItem();
        String year = (String) listYear.getSelectedItem();
        list = dbmf.getTable(name, value, month + "." + year);
    } else {
        list = dbmf.getTable(name, value, null);
    }

    if (tableModel == null) {
        initTable();

        // initNumberTable(jTable1.getRowCount());
    }
    cleanRowTable();
//-----//
    int[] tmpArrIndexes = new int[list.size()];
    for (int i = 0, counter = 1; i < tmpArrIndexes.length; i++, counter++) {
        tmpArrIndexes[i] = counter;
    }

    int tmpCounter = 0;
    for (Object[] objects : list) {
        // objects[0] = tmpArrIndexes[tmpCounter];
        addRowToTable(objects);
        //tmpCounter++;
    }
    initNumberTable(list.size());
    tempObjs = list;
//-----//
} catch (Exception ex) {
    ex.printStackTrace();

    JOptionPane.showMessageDialog(this, "Сбой работы с базой данных,
перезапустите приложения и попробуйте снова", "Ошибка работы с базой данных",

```

```

MessageType.ERROR.ordinal());
    }
}

private void checkToIncorrectData(String[] arrayValues) {
    for (String string : arrayValues) {
        if (string == null) {
            string = "";
        }
    }
}

/**
 * Метод создаёт объект таблицы, инициализирует все столбцы, Также создаёт
 * объект типа ListSelectionModel для установки одиночного выбора строки в
 * таблице.Метод вызывается один раз, при создании таблицы.
 *
 * Для изменения порядка, или имён в таблице следует менять только методы
 * tableModel.addColumn("some text");
 */
private void initTable() {
    tableModel = (DefaultTableModel) jTable1.getModel();
    tableModel.addColumn("ID");
    tableModel.addColumn("Дата");
tableModel.addColumn("№ пішення");
    tableModel.addColumn("Номер сесії");
    tableModel.addColumn("Номер скликання");
    tableModel.addColumn("Назва ");
    tableModel.addColumn("Назва комісії");
    tableModel.addColumn("Чинність");

    //jTable1.removeColumn(null);
    jTable1.getColumnModel().getColumn(0).setPreferredWidth(0);
    jTable1.getColumnModel().getColumn(0).setMinWidth(0);
    jTable1.getColumnModel().getColumn(0).setMaxWidth(0);
    jTable1.getColumnModel().getColumn(1).setPreferredWidth(75);
    jTable1.getColumnModel().getColumn(1).setMaxWidth(75);
    jTable1.getColumnModel().getColumn(2).setPreferredWidth(78);
    jTable1.getColumnModel().getColumn(2).setMaxWidth(78);
    jTable1.getColumnModel().getColumn(3).setPreferredWidth(75);
    jTable1.getColumnModel().getColumn(3).setMaxWidth(75);
    jTable1.getColumnModel().getColumn(4).setPreferredWidth(115);
    jTable1.getColumnModel().getColumn(4).setMaxWidth(116);
    jTable1.getColumnModel().getColumn(5).setPreferredWidth(475);
    jTable1.getColumnModel().getColumn(6).setPreferredWidth(90);
    jTable1.getColumnModel().getColumn(6).setMaxWidth(90);
    jTable1.getColumnModel().getColumn(7).setPreferredWidth(85);
    jTable1.getColumnModel().getColumn(7).setMaxWidth(85);
}

```

```

//      TableRowSorter tableRowSorter = new TableRowSorter(tableModel);
//      tableRowSorter.setSortTable(1, true);
//      tableRowSorter.setSortTable(3, true);
//      ArrayList<SortKey> sortedKey = new ArrayList<>();
//      sortedKey.add(new SortKey(1, SortOrder.DESENDING));
//      sortedKey.add(new SortKey(1, SortOrder.DESENDING));
//      tableRowSorter.setSortKeys(sortedKey);
//      tableRowSorter.toggleSortOrder(1);
//      tableRowSorter.toggleSortOrder(3);
//      tableRowSorter.setSortsOnUpdates(true);
//      jTable1.setRowSorter(tableRowSorter);

ListSelectionModel selectionModel = jTable1.getSelectionModel();
selectionModel.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
//      jTable1.getSelectionModel().addListSelectionListener(lslistener);
}

private void initNumberTable(int rowCount) {
    try {
        jTable2 = new javax.swing.JTable();
        DefaultTableModel model = new DefaultTableModel();
        model.setColumnIdentifiers(new Object[]{
            "№П/п"});
        for (int count = 0; count < rowCount; count++) {
            Object[] obj = new Object[]{count + 1};
            model.insertRow(count, obj);
        }
        jTable2.setModel(model);
jScrollPane9.setViewportViewView(jTable2);
        jTable2.setVisible(true);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

/**
 * Метод очищает таблицу от всех элементов
 *
 * Не модифицировать. Do not modify !
 */
private void cleanRowTable() {
    int rows = tableModel.getRowCount();
    for (int i = rows - 1; i >= 0; i--) {
        tableModel.removeRow(i);
    }
}

/**
 * Метод добавляет массив элементов в строку, Столбцы описанны в методе

```

```

    * initTable()
    *
    * Значения индексов массива должны совпадать со значением
    * инициализированной таблице
    *
    * @param objects - массив содержащий строку таблицы
    */
private void addRowToTable(Object[] objects) {
    tableModel.addRow(objects);
}

/**
 * Метод котролёр который блокирует все остальные radioButtons кроме того
 * который был включен.
 *
 * @param radioButton - объект JRadioButton которые необходимо оставить
 * выбранным
 */
private void conntrolerViewRadioButton(JRadioButton radioButton) {
    for (JRadioButton jRadioButton : boxs) {
        jRadioButton.setSelected(false);
    }
    radioButton.setSelected(true);
}

/**
 * Метод контролёр который блокирует все объекты компонентов JComboBox
 * описанные в коллекции ArrayList<JComboBox> comboBox, Блокировка проходит
 * методом установки setEnabled(false)
 *
 * @param comboBox -параметр который необходимо оставить включённым
 * setEnabled(true). Тоесть каждому компоненту JRadioButton соответствует свой
 * JComboBox.
 */
private void conntrolerViewComboBox(JComboBox comboBox) {
    for (JComboBox jComboBox : comboBoxs) {
        jComboBox.setEnabled(false);
    }
    comboBox.setEnabled(true);
    isSelectedDateOption = false;
}

public static void main(String args[]) {
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Windows".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    }
}

```



```

        }
    }
} catch (ClassNotFoundException | InstantiationException |
IllegalAccessException | javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(MainFrame.class.getName()).log(java.util.logging.Le
vel.SEVERE, null, ex);
    }

    java.awt.EventQueue.invokeLater(new Runnable() {
        @Override
        public void run() {
            new MainFrame().setVisible(true);
        }
    });
}

// Variables declaration - do not modify
private javax.swing.JMenuItem DeleteCardDecision1;
private javax.swing.JButton addAbbrevationCommision;
private javax.swing.JButton addCommisionState;
private javax.swing.JButton addHeadCommision;
private javax.swing.JButton addMoreClassifier;
private javax.swing.JButton addMoreClassifier1;
private javax.swing.JButton addNamesCommision;
private javax.swing.JButton addRowClassifier;
private javax.swing.JButton cancel;
private javax.swing.JButton cancelClassfierModel;
private javax.swing.JButton cancelMoreClassifier;
private javax.swing.JButton cancelMoreClassifier1;
private javax.swing.JMenu cardDecisionMenu;
private javax.swing.JTable classifierTable;
private javax.swing.JMenuItem createCardDecision;
private javax.swing.JButton createNewNotation;
private javax.swing.JFrame editClassifier;
private javax.swing.JMenuItem exitMenu;
private javax.swing.JFrame frameMoreClassifier;
private javax.swing.JFrame frameMoreClassifier1;
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton2;
private javax.swing.JButton jButton3;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JLabel jLabel6;
private javax.swing.JMenu jMenu2;
private javax.swing.JMenu jMenu3;
private javax.swing.JMenuBar jMenuBar1;

```

```
private javax.swing.JMenuBar jMenuBar2;
private javax.swing.JMenuItem jMenuItem1;
private javax.swing.JMenuItem jMenuItem2;
private javax.swing.JMenuItem jMenuItem3;
private javax.swing.JMenuItem jMenuItem4;
private javax.swing.JMenuItem jMenuItem5;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JScrollPane jScrollPane2;
private javax.swing.JScrollPane jScrollPane3;
private javax.swing.JScrollPane jScrollPane4;
private javax.swing.JScrollPane jScrollPane5;
private javax.swing.JScrollPane jScrollPane6;
private javax.swing.JScrollPane jScrollPane7;
private javax.swing.JScrollPane jScrollPane8;
private javax.swing.JScrollPane jScrollPane9;
private javax.swing.JTable jTable1;
private javax.swing.JTable jTable2;
private javax.swing.JList listAbbreviation;
private javax.swing.JComboBox listAddClassifier;
private javax.swing.JComboBox listClassifier;
private javax.swing.JComboBox listCommision;
private javax.swing.JComboBox listConvocation;
private javax.swing.JComboBox listDateOption;
private javax.swing.JList listHeadCommision;
private javax.swing.JList listNamesCommision;
private javax.swing.JComboBox listSession;
private javax.swing.JComboBox listState;
private javax.swing.JComboBox listYear;
private javax.swing.JMenu mainMenu;
private javax.swing.JFrame newValue;
private javax.swing.JMenuItem openCardDecision;
private javax.swing.JMenu optionMenu;
private javax.swing.JFrame relatedCommission;
private javax.swing.JTable relatedTable;
private javax.swing.JButton removeAbbreviationCommision;
private javax.swing.JButton removeCommisionState;
private javax.swing.JButton removeHeadCommision;
private javax.swing.JButton removeMoreClassifier;
private javax.swing.JButton removeMoreClassifier1;
private javax.swing.JButton removeRowClassifier;
private javax.swing.JButton renoveNamesCommision;
private javax.swing.JButton saveClassfierModel;
private javax.swing.JButton saveCommisionState;
private javax.swing.JButton saveMoreClassifier;
private javax.swing.JButton saveMoreClassifier1;
private javax.swing.JRadioButton seachAddClassifier;
private javax.swing.JRadioButton seachClassifier;
private javax.swing.JRadioButton seachCommision;
private javax.swing.JRadioButton seachConvocation;
```

```

private javax.swing.JButton seachDatabase;
private javax.swing.JFrame seachFrame;
private javax.swing.JMenuItem seachMenu;
private javax.swing.JLabel seachParametrText;
private javax.swing.JTextField seachParametrValue;
private javax.swing.JRadioButton seachSession;
private javax.swing.JRadioButton seachState;
private javax.swing.JRadioButton seachYear;
private javax.swing.JFrame settingCommittee;
private javax.swing.JTable tableMoreClassifier;
private javax.swing.JTable tableMoreClassifier1;
private javax.swing.JLabel textNewValue;
private javax.swing.JTextField valueField;
// End of variables declaration

private void initEditClassifier() {
    try {
        ListSelectionModel selectionModel = classifierTable.getSelectionModel();
        selectionModel.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);

        classifierTableModel = (DefaultTableModel) classifierTable.getModel();
        classifierTableModel.addColumn("Класифікатор по категорії");
        classifierTableModel.addColumn("Постійна комісія");
        classifierTableModel.addColumn("Додаткова постійна комісія");
        classifierTableModel.addColumn("Додаткова постійна комісія");

        JComboBox classifierModel = new JComboBox(dbmf.getClassifierList());
        JComboBox commision = new JComboBox(dbmf.getCommisionAbbreviation());
        classifierTable.getColumnModel().getColumn(0).setCellEditor(new
DefaultCellEditor(classifierModel));
        classifierTable.getColumnModel().getColumn(1).setCellEditor(new
DefaultCellEditor(commision));
        classifierTable.getColumnModel().getColumn(2).setCellEditor(new
DefaultCellEditor(commision));
        classifierTable.getColumnModel().getColumn(3).setCellEditor(new
DefaultCellEditor(commision));

        ArrayList<Object[]> classifierStateList = dbmf.getClassifierState();

        for (Object[] objects : classifierStateList) {
            classifierTableModel.addRow(objects);
        }
    } catch (SQLException ex) {
        ex.printStackTrace();
        JOptionPane.showMessageDialog(this, "Сбой работы с базой данных,
перезапустите приложения и попробуйте снова", "Ошибка работы с базой данных",
MessageType.ERROR.ordinal());
    }
}

```

```

private void initRelatedTable() {
    try {
        ListSelectionModel selectionModel = relatedTable.getSelectionModel();
        selectionModel.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);

        relatedTableModel = (DefaultTableModel) relatedTable.getModel();
        relatedTableModel.addColumn("Голова комісії");
        relatedTableModel.addColumn("Назва комісії");
        relatedTableModel.addColumn("Аббревіатура");

        JComboBox headCommision = new JComboBox(dbmf.getCommisionHead());
        JComboBox nameCommision = new JComboBox(dbmf.getCommisionName());
        JComboBox abbreviation = new JComboBox(dbmf.getCommisionAbbreviation());

        relatedTable.getColumnModel().getColumn(0).setCellEditor(new
DefaultCellEditor(headCommision));
        relatedTable.getColumnModel().getColumn(1).setCellEditor(new
DefaultCellEditor(nameCommision));
        relatedTable.getColumnModel().getColumn(2).setCellEditor(new
DefaultCellEditor(abbreviation));

        ArrayList<Object[]> list = dbmf.getRelatedCommision();
        for (Object[] objects : list) {
            relatedTableModel.addRow(objects);
        }
    } catch (SQLException ex) {
        ex.printStackTrace();
        JOptionPane.showMessageDialog(this, "Сбой работы с базой данных, перезапустите
приложения и попробуйте снова", "Ошибка работы с базой данных",
MessageType.ERROR.ordinal());
    }
}

private void initMoreTable() {
    try {
        moreClassifierDefaultModel = (DefaultTableModel)
tableMoreClassifier.getModel();
        ArrayList<String> list = dbmf.getMoreClassifier();
        for (String value : list) {
            moreClassifierDefaultModel.addRow(new Object[]{value});
        }
    } catch (SQLException ex) {
        ex.printStackTrace();
        JOptionPane.showMessageDialog(this, "Сбой работы с базой данных,
перезапустите приложения и попробуйте снова", "Ошибка работы с базой данных",
MessageType.ERROR.ordinal());
    }
}

```

```

    }
}

private void initEditClassifierOne() {
    try {
        classifierTableEdit = (DefaultTableModel) tableMoreClassifier1.getModel();
        ArrayList<String> list = dbmf.getClassifierEditebale();
        for (String value : list) {
            classifierTableEdit.addRow(new Object[]{value});
        }
    } catch (SQLException ex) {
        ex.printStackTrace();
        JOptionPane.showMessageDialog(this, "Сбой работы с базой данных,
перезапустите приложения и попробуйте снова", "Ошибка работы с базой данных",
MessageType.ERROR.ordinal());
    }
}
}

```