НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Навчально-науковий інститут атомної та теплової енергетики

Кафедра цифрових технологій в енергетиці

"До захисту допущено"
Завідувач кафедри
_____ Наталія АУШЕВА
"___" _____ 2026 р.

Дипломна робота
на здобуття ступеня бакалавр

За освітньою програмою "Цифрові технології в енергетиці"

Спеціальності 122 "Комп'ютерні науки"

на тему "Інформаційна системи обліку та моніторингу переміщення радіоактивних відходів"

Виконав: студент 4 курсу, групи ТР-21 Семененко Максим Олексійович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник доцент каф. ЦТЕ, к.військ.н, доцент Андрій ОНИСЬКО _____
(посада, науковий ступінь, вчене звання, ім'я, ПРІЗВИЩЕ) (підпис)

Рецензент доцент каф. ТАЕ, к.т.н, доцент Олександр БАРАНЮК _____
(посада, науковий ступінь, вчене звання, ім'я, ПРІЗВИЩЕ) (підпис)

Н.контроль старший викладач каф. ЦТЕ Ольга БЕСПАЛА _____
(посада, ім'я, ПРІЗВИЩЕ) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ - 2026

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Навчально-науковий інститут атомної та теплової енергетики

Кафедра Цифрових технологій в енергетиці

Рівень вищої освіти перший (бакалаврський)
спеціальність 122 “Комп’ютерні науки”
Освітньо-професійна програма “Цифрові технології в енергетиці”

ЗАТВЕРДЖУЮ
Завідувач кафедри ЦТЕ
_____ Наталія АУШЕВА
(підпис)
“___”_____2026 р.

ЗАВДАННЯ
на дипломну роботу студенту

Семененку Максиму Олексійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи “Інформаційна системи обліку та моніторингу переміщення радіоактивних відходів”
Науковий керівник Онисько Андрій Ілліч, к.військ.н, доцент
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від “28” 05 2026 року № 1992-с

2. Термін подання студентом роботи 09.06.2026 р.
3. Вихідні дані до роботи середовище розробки Ruby on Rails 4.0,бібліотека JavaScript jQuery.
4. Перелік завдань, що підлягають розробці в дипломній роботі: провести аналіз існуючого обліку радіоактивних відходів, обґрунтування вибору засобів розробки системи, розробка системи та реалізація з іншими інформаційними системами».
5. Орієнтовний перелік ілюстративного матеріалу: архітектура програмних засобів, діаграма прецедентів, макет користувацького інтерфейсу, інтерфейс візуалізації результатів

6. Дата видачі завдання 15.09.2025 р.

Календарний план

№ з/п	Назва етапів дипломної роботи	Строк виконання роботи	Примітки
1.	Затвердження теми роботи	13.05.2025	Виконано
2.	Вивчення та аналіз задачі	14.04.2026-19.04.26	Виконано
3.	Розробка архітектури та загальної структури системи	21.04.2026-26.04.26	Виконано
4.	Розробка структур окремих підсистем	28.04.2026-03.05.26	Виконано
5.	Програмна реалізація системи	05.05.2026-10.05.26	Виконано
6.	Захист програмного продукту	11.05.2026-18.05.2026	Виконано
7.	Оформлення пояснювальної записки	14.05.2026	Виконано
8.	Передзахист	02.06.2026	Виконано
9.	Захист	18.06.2026	Виконано

Студент _____ Максим СЕМЕНЕНКО
(підпис) (ім'я, ПРІЗВИЩЕ)

Керівник роботи _____ Андрій ОНИСЬКО
(підпис) (ім'я, ПРІЗВИЩЕ)

АНОТАЦІЯ

Дипломна бакалаврська робота виконана на 64 сторінках комп'ютерного тексту та структурована за п'ятьма основними розділами. Пояснювальна записка містить 35 ілюстративних матеріалів, 26 таблиць, один індивідуальний додаток, а список бібліографічних джерел охоплює 22 найменування.

Мета роботи: Дослідження спрямоване на проектування та програмну реалізацію автоматизованої платформи для ведення інвентаризаційного обліку РАВ у межах України.

Методи та засоби: база даних SQLite, інтегроване середовище розробки NetBeans IDE, мова програмування Ruby, модулі Devise, CanCan, Kaminari та бібліотека JavaScript jQuery.

Результат: інформаційно-аналітична система, що призначений для автоматизації процесів оперативного збору, централізованого оброблення, верифікації та безпечної передачі даних щодо обліку, поточного стану та динаміки руху радіоактивних відходів.

Ключові слова: РАДІОАКТИВНІ ВІДХОДИ, ІНФОРМАЦІЙНО-АНАЛІТИЧНА СИСТЕМА, БАЗА ДАНИХ SQLITE, RUBY, КОРИСТУВАЦЬКІ ЗАПИТИ, API.

ABSTRACT

This 64-page bachelor's thesis comprises five main chapters, supplemented by 35 illustrations, 26 tables, one appendix, and a list of 22 bibliographic sources.

The project focuses on creating and implementing a dedicated software solution aimed at auditing and tracking radioactive waste materials within Ukraine.

Methods and tools: SQLite database, NetBeans IDE integrated development environment, Ruby programming language, Devise modules, CanCan, Kaminari and jQuery JavaScript library.

The result is an information and analytical system designed to automate the processes of operational collection, centralized processing, verification and safe transmission of data on accounting, current state and dynamics of the movement of radioactive waste.

Keywords: RADIOACTIVE WASTE, INFORMATION AND ANALYTICAL SYSTEM, SQLITE DATABASE, RUBY, USER REQUESTS, API.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ,СКОРОЧЕНЬ І ТЕРМІНІВ.....7
ВСТУП…….....8
1. ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ……..…….....10
2. СУЧАСНИЙ СТАН ТА ТЕХНОЛОГІЧНІ КОМПЛЕКСИ ПОВОДЖЕННЯ
3. РАДІОАКТИВНИМИ ВІДХОДАМИ В УКРАЇНІ.....12
2.1.Технологічні регламенти тимчасового зберігання радіоактивних відходів.….....……14
2.2.Технологічні стратегії та довгострокова безпека остаточного захоронення радіоактивних відходів….....……15
3. ЗАСОБИ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ…...……18
3.1. Мова програмування Ruby……18
3.2. Програмна основа Ruby on Rails…...…….....19
3.3. Бібліотека JavaScript jQuery…….....21
4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ…….....24
4.1. Діаграма прецедентів…….....24
4.2. Архітектура програмного продукту……..….....27
4.3. Структура класів програми…….…….....29
4.4. Опис бази даних……...….....31
4.4.1. Даталогічна архітектура бази даних …….....33
4.4.2. Структура таблиць бази даних……36
5. ОПИС РОБОТИ З СИСТЕМОЮ..……...…….....42
5.1. Інсталяція та системні вимоги……...42
5.2. Сценарії роботи користувача з системою.....……..…….....43
ВИСНОВКИ…….62
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....64
ДОДАТОК А.....67

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ДІВ − джерела іонізуючого випромінювання.

РАВ − радіоактивні відходи.

СКБД − об'єктно-реляційна система керування базами даних

CLI (Command-Line Interface) - консольна модель взаємодії із додатком, за якої користувач контролює роботу системи шляхом безпосереднього набору текстових інструкцій у термінальному вікні

ORM (Object-Relational Mapping) - архітектурний інструментарій, що дозволяє програмісту оперувати записами бази даних як стандартними об'єктами бізнес-логіки, повністю нівелюючи потребу у прямому написанні SQL-запитів

SQL (Structured Query Language) - стандартизована мова запитів, що виконує роль для керування даними та взаємодії застосунку з базою даних.

ВСТУП

Масштабне застосування радіоактивних матеріалів в економіці у середині минулого століття спричинило накопичення великої кількості радіоактивних відходів на українських підприємствах. Тому, було визначено, щодо необхідності створення шостих державних міжобласних спеціальних комбінатів: Дніпропетровського, Донецького, Київського, Львівського, Одеського і Харківського, Головним завданням, для забезпечення якого розроблялася система, є моніторинг процесів знешкодження різних видів радіоактивних відходів, їх збирання, транспортування, переробки, захоронення, а також дезактивації забрудненого радіонуклідами спецодягу та ліквідації радіаційних аварій на території України.

Метою дипломної роботи є реалізація системи яка б дозволила впорядковувати, оброблювати та аналізувати данні, що надходять з міжобласних спеціалізованих комбінатів до головного інформаційно-аналітичного центру обліку радіоактивних відходів (далі РАВ).

Державна корпорація Державне спеціалізоване підприємство “Об’єднання “Радон”, аналогічно спеціалізованим компаніям провідних країн, що є членами МАГАТЕ (Bfs – ФРН, NIREX – Великобританія, ANDRA – Франція, ENRESA – Іспанія, ONDRA / NIRAS – Бельгія). Дане об'єднання виступає провідним державним суб'єктом, який здійснює збір, логістику, переробку та довготривале зберігання РАВ неядерного циклу, що виникають у процесі діяльності українських промислових та наукових установ. Регіональні філії та комбінати компанії реалізують комплексні інженерно-технічнізавдання. Вони охоплюють проектування сховищ, створення профільних приладів, безперервний дозиметричний контроль та безпечну експлуатацію майданчиків для ізоляції відходів на всіх технологічних стадіях, зокрема й у межах Чорнобильської зони.

Одним із ключових підрозділів у складі ДАЗВ виступає ДСП «Об'єднання «Радон». Профільний вектор діяльності цієї організації охоплює повний цикл менеджменту неядерних радіоактивних відходів. Окрім цього, фахівці

об'єднання забезпечують проведення аварійно-рятувальних та відновлювальних робіт у разі виникнення радіаційних надзвичайних ситуацій на території України.

Організаційна структура держкорпорації сформована задля ефективного вирішення цих завдань шляхом об'єднання низки ключових суб’єктів. Базовим підрозділом визначено Чорнобильське ДСП «ЦППРВ». Локальний моніторинг та операційну діяльність здійснюють шість міжобласних спецкомбінатів — Одеський, Харківський, Львівський, Київський, Дніпропетровський та Донецький. Крім того, до складу увійшли дві спеціалізовані установи: Науково-технічний центр дезактивації (включаючи поводження з РАВ та ДІВ) і Український радіологічний навчальний центр.

1. ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

Концепція обліку РАВ базується на створенні єдиного інформаційного простору для збору, верифікації, аналітичного оцінювання та транзиту відомостей про поточний стан небезпечних матеріалів. Програмне забезпечення вирішує завдання з прогнозування екологічних ризиків та генерації обґрунтованих сценаріїв для прийняття управлінських рішень. Головною метою впровадження такого підходу є недопущення погіршення екологічного балансу та забезпечення відповідності чинним стандартам екобезпеки.

Система обліку радіоактивних відходів створюється як інформаційно-аналітична система, призначена для забезпечення сучасного науково-технічного рівня збору, обробки та передачі необхідних даних.

Розроблена система обліку орієнтована на автоматизацію діяльності інформаційно-аналітичних підрозділів при державних спеціалізованих комбінатах. Головною функцією програмного забезпечення є ведення детального моніторингу РАВ та джерел іонізуючого випромінювання (ДІВ), що перебувають на зберіганні. Крім того, комплекс дозволяє збирати дані щодо інженерно-дозиметричного стану ізоляційних споруд та локацій фінального захоронення на кожному етапі оперування відходами, зокрема й на об'єктах у межах Чорнобильської зони відчуження.

Враховуючи обсяги надходження, переробки та захоронень радіоактивних відходів, програмний продукт повинен відповідати усім сучасним вимогам безпеки та витримувати навантаження, що в свою чергу забезпечить безперебійну та передбачувану роботу системи.

Основні задачі які повинна виконувати система:

- внесення та відображення даних про нові радіоактивні відходи та джерела іонізуючого випромінювання, що були прийняті до спеціалізованих комбінатів, зберігаються в існуючих сховищах, та пунктах захоронень;
- внесення та відображення даних про перевезення та обробку радіоактивних відходів і джерел іонізуючого випромінювання;
- формування детальних звітів за запитом користувача;

- представлення даних у зручному для сприйняття користувачем вигляді, тобто у вигляді графіків, діаграм та таблиць.

Однією з основних задач які повинна виконувати система це представлення статистично-аналітичних даних для головного інформаційно-аналітичного центру обліку РАВ, що були отримані з шостих державних міжобласних спеціалізованих комбінатів.

2. СУЧАСНИЙ СТАН ТА ТЕХНОЛОГІЧНІ КОМПЛЕКСИ ПОВОДЖЕННЯ З РАДІОАКТИВНИМИ ВІДХОДАМИ В УКРАЇНІ

У контексті забезпечення сталого розвитку та охорони навколишнього природного середовища України одним із найбільш вагомих інженерно-екологічних завдань виступає гарантування безпеки під час поводження з радіоактивними відходами та їх утилізація. Актуальність зазначеної проблематики визначається необхідністю недопущення забруднення навколишнього природного середовища внаслідок неконтрольованого потрапляння РАВ у воду, ґрунти та атмосферне повітря, а також потребою в екологічного оздоровлення регіонів, що постраждали від викиду радіонуклідів у результаті аварії на ЧАЕС.

Процес локалізації токсичних відходів передбачає їхнє повне консервування поза межами біосфери на строк, протягом якого відбудеться самовільний розпад радіоактивних елементів. Практична реалізація подібної ізоляції зазвичай спирається на дві основні методології: ізоляції від біосфери на період, протягом якого відбудеться природний розпад радіонуклідів. На практиці використовують два ключові методи такої ізоляції: зберігання та захоронення, вибір яких залежить від комплексу економічних, технологічних і соціально-політичних чинників [8].

Криза у секторі накопичення радіоактивних залишків набула критичного характеру після катастрофи на ЧАЕС, яка спровокувала появу сотень тисяч кубометрів РАВ різних класифікаційних рівнів. Наразі географія розповсюдження цих речовин охоплює не лише внутрішні об'єкти зони відчуження (включаючи ПТЛРО, ПЗРО та комплекс «Укриття»), а й території за її кордонами. Значна частка даного потенціалу, спираючись на чинні державні стандарти, кваліфікується як довгоіснуючі РАВ. На жаль, частина цих матеріалів депонована з порушенням чинних критеріїв протирадіаційного захисту [9, 10]. Модернізація безпекових підходів до поводження з РАВ в Україні відбувалася з огляду на світові тренди та методологічні рекомендації

МАГАТЕ. Сучасні концептуальні рішення були успішно закріплені в оновленому законодавчому полі нашої держави.

Таблиця 1.1. Розподіл обсягів твердих радіоактивних відходів в Україні

Місцезнаходження	Обсяг, тис. куб. метрів	Відсоток від загального обсягу РАВ, % в Україні
Атомні електростанції	33,17	1,2
Об'єкт "Укриття"	600	22
Зона відчуження	1913	70,2
Пункти захоронення відходів дезактивації	171	6,3
Чорнобильська АЕС	2,5	0,1
Сховища спецкомбінатів	5	0,2

Таблиця 1.2. Розподіл обсягів рідких радіоактивних відходів в Україні

Місцезнаходження	Обсяг, тис. куб. метрів	Відсоток від загального обсягу РАВ, % в Україні
Атомні електростанції	18,57	44,2
Об'єкт "Укриття"	2,5	6
Дослідницькі ядерні реактори	0,37	0,9
Чорнобильська АЕС	20	47,6
Сховища спецкомбінатів	0,62	1,5

Підсумовуючи, можна констатувати критичний рівень накопичення РАВ у державі та постійне зростання їхньої кількості. Це вимагає створення надійної інфраструктури для їх депонування. Головними деструктивними факторами тут є застарілі підходи до моніторингу, незавершеність організаційних реформ та дефіцит фінансування. Відсутність адекватного реагування на ці виклики провокує низку негативних наслідків:

- **Екологічні ризики:** посилення шкідливого впливу іонізуючого випромінювання на суспільство та біосферу в контексті ліквідації спадщини Чорнобиля.

- **Соціальний аспект:** потенційне зростання радіофобії та психологічного тиску в соціумі на тлі подальшої розбудови атомного сектору.
- **Стратегічні загрози:** Гальмування реалізації положень Енергетичної стратегії України на період до 2050 року, що створює економічний тягар для нащадків та стримує євроінтеграційний поступ держави у сфері радіаційного захисту.

2.1. Технологічні регламенти тимчасового зберігання радіоактивних відходів

Етап проміжного утримання таких матеріалів вимагає безперервного інженерно-технічного супроводу та радіаційного аудиту ізоляційних споруд. Саме тому в процесі депонування РАВ.відбувається опромінення персоналу, який здійснює обслуговування об'єктів та контроль їхнього стану, а також існує постійний ризик аварійного витоку радіоактивних речовин.

На території України поширеною практикою є організація приповерхневих локацій для депонування РАВ. Конструктивно такі об'єкти реалізують як залізобетонні модулі, спеціальні траншеї, мікрошахти, підземні колодязі або штучні кургани. Запобігання виходу радіоізотопів за межі цих споруд забезпечується за рахунок комплексу спеціалізованих інженерних бар'єрів. Під цим терміном розуміють сукупність штучно створених конструкційних елементів, які блокують міграцію небезпечних речовин та ізолюють джерела іонізуючого випромінювання від екосистеми

Довгострокове депонування небезпечних залишків супроводжується загрозою пошкодження верхньої ізоляційної оболонки, що може дестабілізувати внутрішнє середовище сховища. Саме тому розробка теоретико-технічних рішень для підтвердження довговічності інженерних споруд за таких умов є пріоритетним напрямом досліджень. Важливим складником цієї системи безпеки виступає герметичний контейнер, який слугує первинним фізичним бар'єром

на шляху міграції радіоактивних елементів.

Функціональне призначення контейнерів охоплює підтримку безпекових бар'єрів на всіх стадіях життєвого циклу менеджменту відходів. Сюди належать процеси збору вихідних матеріалів, їх транспортування, сепарації, переробки та остаточного поховання у глибоких геологічних формаціях або приповерхневих накопичувачах.

Оскільки захисна тарна система десятиліттями перебуває в умовах агресивного середовища — під впливом гідрологічних факторів, барометричного тиску, радіолітичного нагріву та мікробіологічної корозії, — її проектування та експлуатація жорстко регламентуються державними стандартами безпеки. Оскільки чинні нормативні документи вимагають остаточного захоронення довгоіснуючих РАВ виключно у глибоких геологічних формаціях, усі державні міжобласні спецкомбінати наразі переходять на технологію тимчасового зберігання відходів.

2.2. Технологічні стратегії та довгострокова безпека остаточного захоронення радіоактивних відходів

З погляду екологічної безпеки та довгострокових економічних витрат метод глибокого геологічного поховання є найбільш обґрунтованим. З огляду на це, сучасні державні програми та пріоритети в галузі менеджменту радіоактивних матеріалів активно адаптуються під розвиток цього стратегічного напряму.

Надійна довготривала локалізація нуклідів у материкових формаціях досягається лише за умови відсутності активного дренажу з боку ґрунтових вод. До категорії середовищ, що задовольняють критерії остаточного депонування РАВ, належать поклади туфу, базальтові та гранітні пласти, шари глини, а також соляні куполи. Об'єми радіоактивних матеріалів, які передбачається захоронити за даною технологією, суттєво відрізняються в кожному окремому

випадку. На цей показник безпосередньо впливає тип обраної інженерної споруди та її внутрішня конфігурація.

Фундаментом довгострокової безпеки будь-якого сховища РАВ є система багатобар'єрного захисту. Вона поєднує в собі інженерні (штучні) та природні (геологічні) бар'єри. Якщо один із бар'єрів зазнає деградації, інші мають стримати міграцію радіонуклідів.

Одним із рішень є інженерні бар'єри (Engineered Barrier System - EBS). Вони складаються з наступних стратегій.

Перша стратегія − матриця відходів (іммобілізація). Радіоактивні відходи переводяться у стійку тверду форму. Для високоактивних відходів застосовують вітрифікацію (сплавлення з боросилікатним або фосфатним склом) або синрок (синтетична кам'яна кераміка). Це мінімізує швидкість вилуговування радіонуклідів підземними водами.

Наступна стратегія − контейнер (каністра). Герметична ємність із корозійностійких матеріалів (чиста мідь, титан, високолегована сталь або чавун). Завдання контейнера − забезпечити повну ізоляцію на період від кількох тисяч до десятків тисяч років, поки розпадається основна маса найбільш активних короткоживучих ізотопів.

І остання − буферний матеріал (засипка). Простір між контейнером і скельною породою заповнюється бентонітовою глиною. При контакті з вологою бентоніт розбухає, створюючи наднизьку водопроникність, та працює як хімічний фільтр (іони радіонуклідів сорбуються на глиняних мінералах).

Також, відомо про природні (геологічні) бар'єри. Вони включають у себе масивну товщу стабільної геологічної формації (на глибині 300–1000 метрів). Головні вимоги до породи:

− низька гідравлічна провідність (відсутність інтенсивного руху підземних вод);

− тектонічна та сейсмічна стабільність регіону;

− висока сорбційна здатність породи щодо радіонуклідів.

Основними типами порід у світовій практиці є кристалічні щити (граніти − Фінляндія, Швеція), глиняні формації (Бельгія, Франція) та соляні пласти (США).

Таким чином, остаточне захоронення радіоактивних відходів є фінальною та найбільш критичною стадією циклу поводження з ядерними матеріалами. На відміну від тимчасового зберігання, яке розраховане на контрольований термін (зазвичай до 50–100 років), остаточне захоронення передбачає повну та безповоротну ізоляцію радіонуклідів від біосфери на геологічні періоди часу (від тисяч до сотень тисяч років) без наміру їх подальшого вилучення та без потреби постійного технічного обслуговування з боку людини.

Стратегія остаточного захоронення − це перехід від концепції “активного управління та контролю” (що потребує фінансових та людських ресурсів майбутніх поколінь) до концепції “пасивної безпеки”, де за рахунок правильного підбору фізико-хімічних матриць, інженерних конструкцій та геологічного середовища гарантується екологічна безпека планети на епохи вперед.

3. ЗАСОБИ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Даний розділ присвячено детальному опису програмної реалізації розробленої системи обліку радіоактивних відходів, для реалізації якої була використана сучасна мова програмування Ruby. Також, було використано сучасні засоби розробки які дозволяють створювати динамічні web-сторінки та забезпечують функціональність ресурсу

3.1. Мова програмування Ruby

Мова програмування Ruby — це інтерпретована мова високого рівня. Вона містить незалежну від операційної системи реалізацію багатопотоковості, має строгу динамічну типізацію, вбудований автоматичний збирач сміття та низку інших можливостей. Наявні інструменти підтримують кілька парадигм програмування, насамперед об'єктно-орієнтовану. сторія створення Ruby бере свій початок у лютому 1993 року завдяки ініціативі японського програміста Юкіхіро Мацумото. Автор прагнув синтезувати переваги існуючих на той час мов програмування, щоб запропонувати IT-спільноті гнучке рішення для підвищення продуктивності праці та спрощення написання коду [1].

Архітектура Ruby базується на мультипарадигмовому підході. Мова успішно поєднує процедурний стиль (де функції та змінні, оголошені поза контекстом користувацьких класів, автоматично інтегруються у базовий клас Object) та об'єктно-орієнтовану концепцію, за якої абсолютно кожна сутність інтерпретується як об'єкт. Фундаментом розробки став відомий принцип «найменшого подиву», згідно з яким логіка виконання коду має збігатися з очікуваннями інженера. Водночас у реаліях Ruby ця теза розкривається не під час перших кроків освоєння синтаксису, а на етапі глибокого проектування систем. За визначенням самого Юкіхіро Мацумото, першочерговим завданням

було створення комфортного робочого середовища без непередбачуваних аномалій насамперед для самого автора технології.

Однак після поширення мови автор із подивом дізнався, що мислення багатьох програмістів є схожим, і для них концепція «найменшої несподіванки» повністю збіглася з його власним баченням. [2,3].

Ruby також успадкувала ідеологію мови програмування Perl у контексті надання розробнику можливостей досягнення одного й того самого результату кількома різними способами. Пріоритетним завданням під час проектування архітектури мови виступало вивільнення інженерного потенціалу від монотонних рутинних операцій, які обчислювальна система здатна реалізувати з вищою ефективністю та швидкістю. Особливий акцент було зроблено на спрощенні щоденних завдань розробника, таких як системне адміністрування та парсинг текстових масивів. На відміну від низькорівневих машинно-орієнтованих стеків, орієнтованих суто на швидкість виконання інструкцій, цей інструмент створювався з фокусом на людино-орієнтований підхід. Оскільки будь-яке програмне забезпечення створюється людьми і для користувачів, ключовий пріоритет зміщується у бік оптимізації та економії інтелектуальних зусиль самої команди розробки. Незважаючи на потенційне зростання навантаження на обчислювальні потужності процесора, дана технологія орієнтована на оперативне та просте розв'язання інженерних задач.

Сукупність специфічних правил кодингу та внутрішньої логіки мови сформували стійке поняття — «Шлях Ruby» (The Ruby Way). Аналізуючи цей концепт, Хел Фултон визначає його ключові складові. Серед них: дотримання розумної лаконічності без надмірного спрощення структури, робота за правилом мінімізації несподіваних результатів (POLA), компроміс у питанні продуктивності на користь зручності програмування, а також гнучка стандартизація, що позбавлена зайвого бюрократизму та педантизму. Головною причиною програмування автори вважають потребу створювати корисні та красиві програмні продукти. У цілому ці принципи не мають жорсткого формального визначення, через що даний термін іноді використовується для критичного аналізу технології. [3].

3.2. Програмна основа Ruby on Rails

Для реалізації системи були використані сучасні засоби розробки які дозволяють створювати динамічні web-сторінки та забезпечують функціональність ресурсу.

Ruby on Rails регламентує фундаментальні принципи розробки програмного забезпечення, які охоплюють такі ключові положення:

- **стандартизація структури:** архітектура вебзастосунку базується на готових шаблонах фреймворку й не потребує проєктування унікальної конфігурації з нуля;
- **семантична гнучкість:** синтаксис мови Ruby дозволяє використовувати легкочитну нотацію для точного визначення бізнес-логіки та семантики програмних модулів;
- **мінімізація надлишковості:** технологія надає ефективні механізми повторного використання компонентів, що дозволяють мінімізувати дублювання коду в межах проєкту (реалізація принципу *Don't Repeat Yourself* — DRY, «не повторюйся»);
- **пріоритет домовленостей:** за замовчуванням у системі застосовуються типові домовленості щодо конфігурації, характерні для більшості сучасних вебзастосунків (реалізація принципу *Convention over Configuration* — «домовленості над конфігурацією»). При цьому явна специфікація налаштувань потрібна лише у нестандартних або специфічних випадках.

Розвиток екосистеми Rails супроводжувався появою численних плагінів, що суттєво розширюють можливості базової платформи. Такі технології, як Sass та CoffeeScript, продемонстрували високу ефективність і з часом стали частиною стандартного постачання фреймворку. Окремі зовнішні компоненти, хоч і не входять до офіційного ядра системи, сприймаються спільнотою як безальтернативні рішення для щоденної розробки. Зокрема, це стосується інструменту RSpec, який де-факто є головним засобом для організації автоматизованого тестування додатків [2,3].

Реліз архітектурної гілки Rails 3 ознаменував перехід до вираженої модульної структури, що супроводжувалося виокремленням багатьох системних компонентів у незалежні бібліотеки. Це зумовлено їхнім швидшим розвитком порівняно з самим фреймворком, а також прагненням полегшити загальну архітектуру ядра Rails.

Кількість розробників, які зробили хоча б одну модифікацію в код Rails на початок 2013 року, склало 2 782 людини.

Деякі модулі: Devise (для аутентифікації), CanCan (для авторизації), Kaminari (для розділення записів, видобутих з бази даних, або елементів масиву по сторінках), Inherited Resources [8] (для спрощення створення контролерів з стандартним CRUD-кодом), FFaker (для випадкової генерації тестових наборів даних у веб-додатках), friendly_id (дозволяє створювати людино-зрозумілі веб-адреси), Active Admin (для створення панелей адміністрування), CommunityEngine [9] (для створення соціальних мереж).

3.3. Бібліотека JavaScript jQuery

Фреймворк клієнтського сценарію jQuery

Технологія jQuery репрезентує кросплатформену JavaScript-бібліотеку з відкритим кодом. Її перша офіційна презентація відбулася на початку 2006 року в межах конференції BarCamp NYC завдяки розробці Джона Ресіга (John Resig). За актуальними статистичними відомостями аналітичної платформи W3Techs, зазначений інструмент інтегровано до архітектури переважної більшості високонавантажених та найбільш відвідуваних веб-ресурсів глобальної мережі.[2]. Ця технологія залишається однією з найпоширеніших бібліотек JavaScript.

Ключова функція jQuery спрямована на спрощення кросбраузерного пошуку та вибірки об'єктів DOM, що реалізується шляхом використання селекторів CSS та XPath. Паралельно цей інструмент виступає ефективним середовищем для конструювання Ajax-додатків, керування подіями та

інтеграції нескладних візуальних ефектів. Специфіка роботи платформи спирається на патерн проектування, де базовий метод при зверненні повертає свій власний об'єкт. Такий підхід дозволяє будувати лаконічний і читабельний код завдяки каскадній побудові функцій. Обов'язковою умовою для використання jQuery є завантаження актуального пакета бібліотеки та його наступне підключення до структури веб-сторінки. Для цього в заголовок HTML-документа додається такий фрагмент програмного коду: <script type="text/javascript" src="/jquery.js"></script>, де jquery.js — ім'я файлу, що містить код бібліотеки. Після ініціалізації розробнику стають доступні всі можливості базового функціоналу jQuery, серед яких: функції ядра, робота із селекторами та атрибутами, обхід дерева DOM, маніпуляції елементами й CSS-властивостями, обробка подій, візуальні ефекти, взаємодія з Ajax та допоміжні утиліти.

Інтеграція технологічного стеку AJAX дозволила забезпечити безшовну комунікацію та транзит інформаційних масивів між серверною мовою Ruby та клієнтськими сценаріями JavaScript. Використання асинхронних запитів забезпечує плавне відображення динамічних елементів, що є критично важливим для сучасних веб-розробок.

В основі парадигми AJAX лежить модель побудови інтерфейсів, яка дозволяє веб-додатку взаємодіяти із сервером «непомітно» для користувача. Клієнтський модуль в автоматичному режимі імпортує виключно цільові пакети відомостей за запитом оператора, зберігаючи стабільність та цілісність відображення інтерфейсу в браузері. Окреслена модель виступає базовим елементом загальної концепції DHTML [4]. AJAX функціонує не як відокремлена технологічна одиниця, а як системний патерн, що інтегрує потенціал декількох стандартів розробки. Створення інтерактивного фронтенд-середовища на базі AJAX забезпечується комплексом взаємопов'язаних алгоритмів. Зокрема, гнучка зміна контенту й архітектури сторінки виконується інструментами DHTML. Безпосередній фоновий обмін даними з сервером без переривання сесії реалізується за допомогою інтерфейсу XMLHttpRequest. При цьому динамічне підключення скриптів і обробка масивів інформації

відбуваються безпосередньо всередині DOM-моделі із застосуванням полегшеного формату JSON. Реалізація цих підходів гарантує створення адаптивних веб-інтерфейсів із миттєвим відгуком системи під час інтенсивного оперування базами даних.

Завдяки асинхронній природі AJAX оператор має можливість безперешкодно взаємодіяти з інтерфейсом системи безпосередньо під час опрацювання запиту серверною частиною. Процес обміну інформацією з сервером протікає непомітно для користувача і не потребує оновления всього вікна браузера, за винятком випадків, коли архітектурою додатка закладено візуальну індикацію з’єднання. Будь-які інтерактивні маніпуляції трансформуються в низькорівневі операції з деревом DOM (Document Object Model). Через цей механізм коригуються доступні інформаційні блоки, що призводить до миттєвої перебудови відображуваного контенту. На цьому ж рівні система перехоплює та інтерпретує події введення, включаючи кліки миші та натискання клавіш. Узгодженість візуального оформлення всіх компонентів програмного продукту та оптимізація доступу до елементів DOM досягається шляхом інтеграції каскадних таблиць стилів (CSS). При цьому об'єкт XMLHttpRequest виступає ключовим інструментом фонового транзиту даних та обробки поточних клієнт-серверних запитів. [4].

4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Для реалізації системи інвентаризації радіоактивних відходів, була обрана одна з найпопулярніших на сьогоднішній день програмних основ – Ruby on Rails. Ruby on Rails — об'єктно-орієнтований фреймворк для створення вебзастосунків, написаний мовою програмування Ruby. Ruby on Rails надає архітектурний шаблон «Модель-Представлення-Контролер» (Model-View-Controller, MVC) для вебзастосунків, а також забезпечує їхню інтеграцію з вебсервером і сервером баз даних.

4.1. Діаграма прецедентів

Концептуально діаграма варіантів використання (use case diagram) представляє собою графічну модель, що описує взаємодію зовнішніх суб'єктів (акторів) із функціональними блоками системи в межах визначеного проектного контуру. Головне завдання такого моделювання полягає у формалізації сервісів, які архітектура розроблюваного програмного забезпечення має надати кінцевому користувачу або суміжним модулям. Кожен окремий прецедент задає специфічний алгоритм та послідовність операцій під час транзиту даних між актором та платформою.

Важливо зауважити, що на даному етапі інженерного аналізу внутрішні механізми та особливості програмної реалізації коду залишаються інваріантними та не регламентуються. Специфікація мови UML виокремлює кілька базових типів логічного зв'язку для декомпозиції цих процесів, зокрема: механізми асоціації, відношення включення (include) та розширення (extend), а також патерни узагальненняПри цьому загальні властивості варіантів використання можуть бути представлені трьома різними способами, а саме — за допомогою відношень включення, розширення та узагальнення. Відношення

асоціації є одним із фундаментальних понять у мові UML і певною мірою використовується під час побудови всіх графічних моделей систем у формі канонічних діаграм.

Включення (include) у мові UML — це різновид відношення залежності, за якого базовий прецедент обов'язково містить у собі функціонал іншого (включеного) варіанта використання.

Загальна специфікація use case зазвичай структурується за допомогою трьох базових підходів: патернів розширення, включення або ієрархічного узагальнення. У методології UML базовим інструментом виступає зв'язок асоціації, який є невід'ємним компонентом проектування більшості графічних моделей та структурних схем. Поняття включення (include) відображає специфічну форму залежності, за якої логіка головного прецеденту беззастережно інтегрує в себе алгоритм роботи допоміжного сценарію. Сама по собі залежність (dependency) фіксує таку причинно-наслідкову сукупність об'єктів, де модифікація параметрів суверенного (незалежного) компонента автоматично викликає трансформацію пов'язаного з ним елемента. Натомість відношення розширення (extend) регламентує взаємодію, за якої додатковий функціонал інтегрується в основний процес не перманентно, а виключно за умови настання конкретних тригерних подій або виконання особливих вимог системи.

В запропонованій системі присутні дві сутності: адміністратор спеціалізованого комбінату та головний адміністратор. Головний адміністратор має більш розширені можливості порівняно з регіональним адміністратором. Зокрема він може спостерігати дані по РАВ які відносяться до будь-якого спеціалізованого комбінату у той час коли регіональний адміністратор має можливість дивитися дані лише по спеціалізованому комбінату до якого він відноситься. Також відрізняється формат звітів які автоматично генеруються для головного адміністратора та адміністратора спеціалізованого комбінату [6].

На рисунку 4.1 представлена діаграма прецедентів, що відображає відношення між головними акторами та прецедентами у нашій системі.

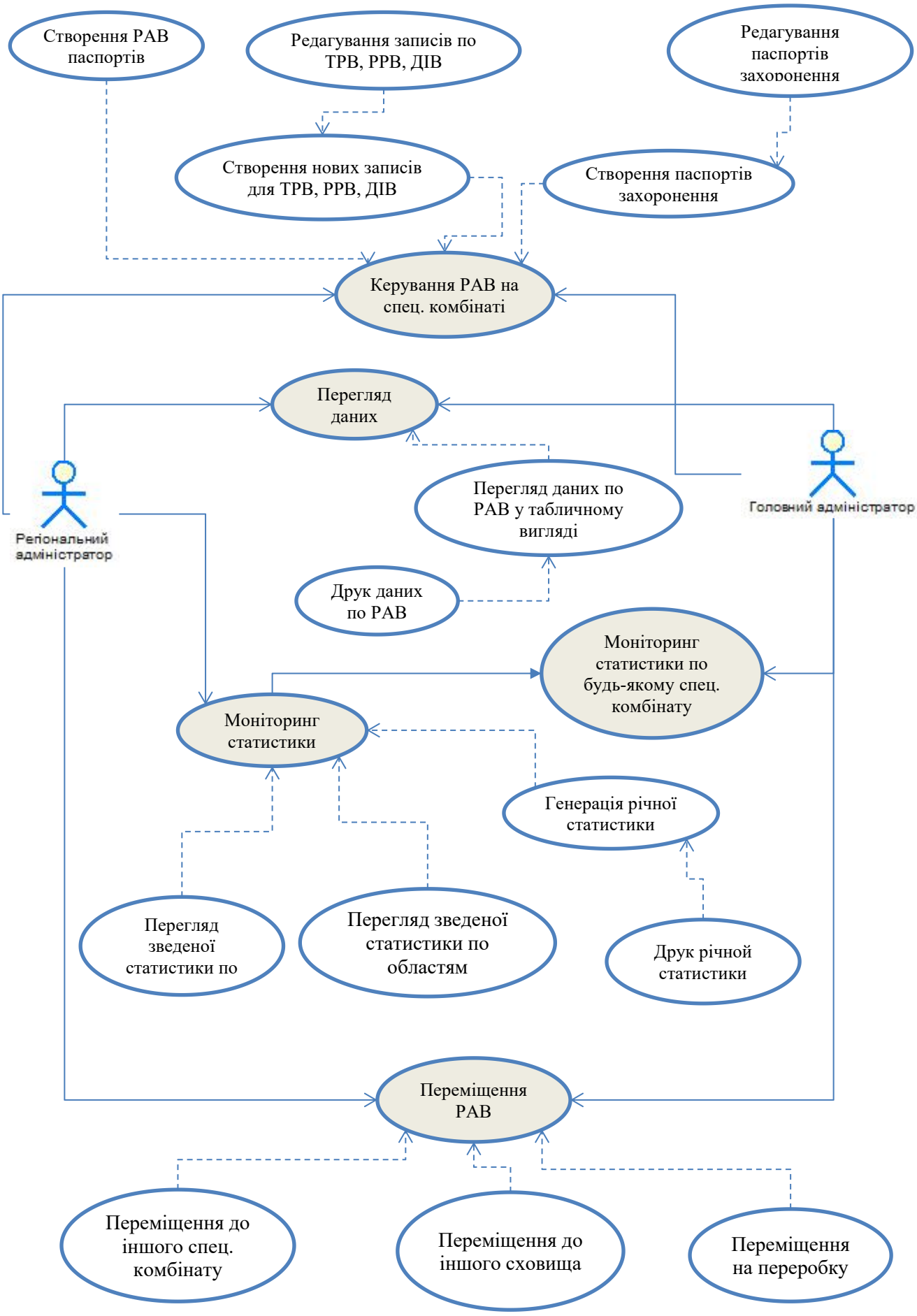


Рисунок 4.1 – Діаграма прецедентів

Таким чином, дана діаграма дозволила спроектувати систему взаємодії між учасниками для порівняння даних по радіоактивним відходам, їх моніторинг, наповнення сховищ та надання звітів для розподілу відходів по підприємствам.

4.2. Архітектура програмного продукту

Ruby on Rails слідує архітектурній схемі модель-представлення-контролер (MVC), яка реалізує розділення між логікою предметної області від логіки вводу та логіки представлення, зв’язаної з графічним інтерфейсом користувача. Зазначений архітектурний патерн передбачає декомпозицію структури програмного продукту на три автономні, але взаємозалежні компоненти: об'єктну модель даних, клієнтське представлення та контролер обробки логіки. Ключова мета впровадження такого підходу полягає в ізоляції бізнес-логіки додатка від його графічного інтерфейсу. Завдяки цьому забезпечується високий рівень інваріантності модулів: будь-які ревізії та зміни візуального оформлення не порушують стабільність операцій з базами відомостей, а модернізація внутрішніх алгоритмів моделі може виконуватися без примусової реструктуризації фронтенд-частини.

Основне призначення шаблону полягає в оптимізації архітектури ПЗ задля полегшення майбутніх процесів модернізації та адаптації системи до нових технічних вимог. Завдяки високій автономності компонентів стає можливим їхній ревз у межах інших програмних рішень. У контексті проектування високонавантажених або розгалужених інформаційних платформ використання MVC гарантує чітку структуризацію кодової бази, що дозволяє ефективно керувати архітектурною складністю системи на всіх етапах її життєвого циклу.

Архітектурний каркас вебдодатків на базі Ruby on Rails формується з трьох фундаментальних модулів: об'єктної моделі, інтерфейсного представлення та контролера обробки подій (рис. 4.2). Рівень моделі відповідає за надання іншим компонентам системи структурованого об'єктно-

орієнтованого інтерфейсу до даних (наприклад, до карток реєстрації РАВ, реєстрів ізоляційних споруд або класифікаторів захисних контейнерів). При цьому сутності моделі інкапсулюють у собі логіку вибірки, оновлення та збереження записів у реляційній СУБД. Використовуючи переваги динамічної типізації мови Ruby, програмісту достатньо реалізувати успадкування користувацького класу від базової абстракції ActiveRecord: Base. Фреймворк самостійно змапує створені класи із відповідними таблицями бази даних, автоматично згенерувавши необхідні атрибути об'єктів на основі наявних полів табличної схеми. Компонент відображення (view) відповідає за генерацію графічного інтерфейсу, візуалізуючи для оператора інформаційні масиви, що надходять від контролера. Окрім цього, цей рівень забезпечує транзит дій та запитів користувача щодо модифікації даних назад до керуючого модуля. Важливо підкреслити, що безпосередньо шар представлення повністю ізольований від сховища даних і самостійно не виконує жодних операцій запису чи коригування полів у базі даних.

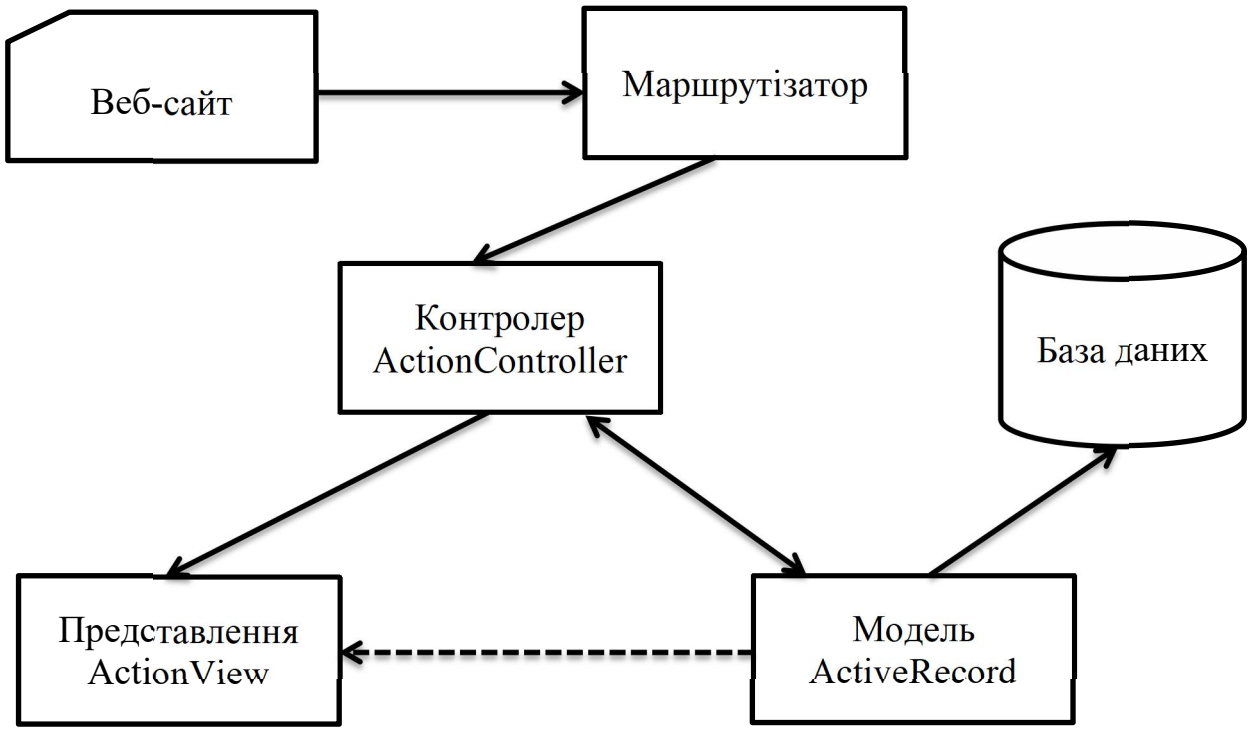


Рисунок 4.2 – Архітектура програмного продукту

У фреймворку Ruby on Rails рівень представлення проектується на основі спеціалізованих HTML-шаблонів. Конструктивно вони репрезентують файли розмітки, які містять інтегровані блоки виконуваного коду Ruby (технологія Embedded Ruby або ERb). Результат обчислень, згенерований цими вбудованими скриптами, динамічно впроваджується безпосередньо у тіло HTML-сторінки, яка згодом транслюється у браузер кінцевого користувача. Крім того, архітектура Rails дозволяє реалізувати модульний підхід: окремі компоненти відображення можуть імпортувати ізольовані фрагменти інших шаблонів (*partials*), а також рендеритися всередині глобальних макетів (*layouts*) вищого ієрархічного рівня. [3,4].

Керуючим ядром додатка є компонент контролера, орієнтований на обробку користувацьких дій та маршрутизацію запитів. Він виконує функції посередника: вилучає об'єкти з реляційних моделей для їхнього подальшого представлення користувачеві та інкапсулює дані інтерфейсу для оновлення полів у базі даних. Програмна реалізація контролерів у Rails базується на створенні класів, що є нащадками системного типу ActionController: Base. Доступні ззовні відкриті методи трактуються архітектурою як екшени (*actions*). Як правило, певний action жорстко пов'язаний із відповідним графічним представленням. Наочним прикладом є обробка URL-адреси admin/list: у цьому сценарії платформа активує метод list контролера AdminController, транслюючи користувачеві згенерований заздалегідь файл відображення list.html.erb.

4.3. Структура класів програми

Діаграма класів системи інвентаризації РАВ (рис. 4.3) відображає статичну структуру інформаційної системи, визначає ключові сутності предметної області, їхні атрибути, методи, а також типи зв’язків між ними. Архітектура спроєктована із дотриманням принципів SOLID та об’єктно-орієнтованого проектування (ООП), що забезпечує масштабованість та легкість підтримки програмного продукту.

Основними структурними компонентами (класами) системи є:

1. WasteContainer (клас контейнера РАВ) – представляє собою центральну сутність системи, яка моделює фізичну упаковку з радіоактивними відходами. Він скаладється з атрибутів: унікального ідентифікатора (id: UUID), серійного номера (serialNumber: String), категорії відходів (wasteClass: WasteCategory), поточна активність (activityLevel: Double), дата герметизації (sealDate: Date) та поточний статус (status: ContainerStatus).

2. RadionuclideComposition (склад радіонуклідів) – клас, що деталізує ізотопний склад всередині кожного контейнера. Він пов'язаний з класом WasteContainer зв'язком композиції (Composition), оскільки ізотопний склад не може існувати окремо від самого контейнера з відходами. Містить інформацію про конкретні ізотопи (^{137}Cs, ^{90}Sr тощо) та їхню питому вагу.

3. StorageLocation (Місце зберігання / Сховище) – моделює фізичну структуру сховища РАВ (зони, секції, комірки). Він пов'язаний з WasteContainer асоціативним зв'язком агрегації (Aggregation) типу "один до багатьох" (1 ... *), оскільки сховище містить багато контейнерів, але при видаленні контейнера зі сховища сам об'єкт сховища не руйнується.

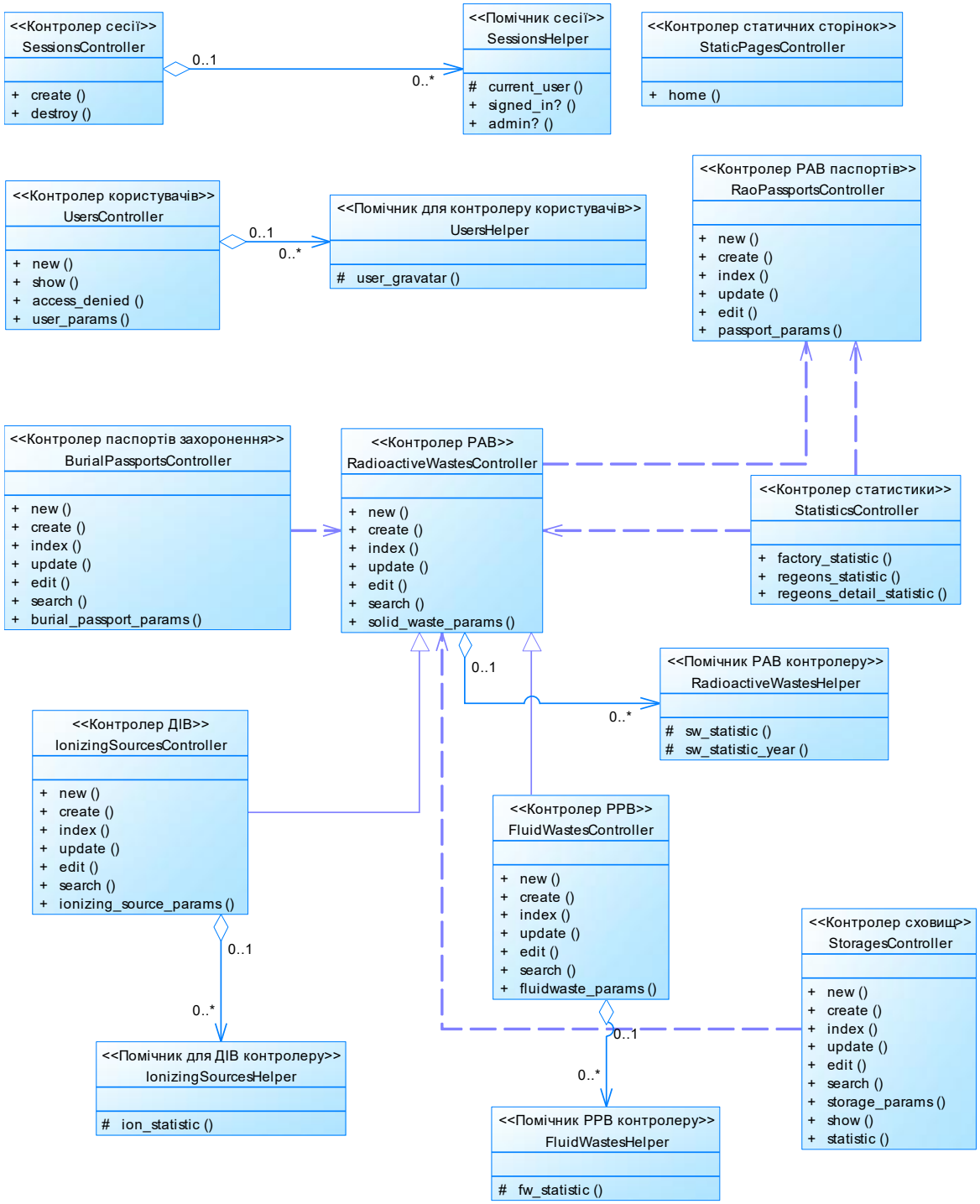


Рисунок 4.3 – Структура класів програмного модулю

Проєктування діаграми класів дозволило формалізувати наступні бізнес-правила системи:

– напрямки стрілок на діаграмі вказують на те, що об’єкт логістики (InventoryLog) має прямий доступ до об’єкта WasteContainer для зчитування його характеристик під час інвентаризації;

− у сі ключові атрибути класів (такі як рівні радіації, координати) мають модифікатор доступу private (-). Зміна їхнього стану можлива виключно через публічні (+) сетери після проходження валідації на відповідність правилам радіаційної безпеки.

Таким чином, описана структура класів є основою для генерації схем реляційної бази даних (через ORM-технології) та подальшої реалізації бізнес-логіки серверної частини інформаційної системи.

4.4. Опис бази даних

Для збереження інформації про РАВ було обрано СУБД SQLite, яка є полегшеною реляційною системою керування базами даних. В основі обраного рішення лежить компактний програмний рушій, що забезпечує підтримку більшої частини операторів стандарту SQL-92. Повна відкритість кодової бази SQLite, яка поширюється у форматі *public domain*, знімає будь-які бар'єри для її розгортання. Інженери можуть абсолютно безкоштовно модифікувати та впроваджувати цю систему автоматизації обліку як у комерційних, так і в науково-дослідних некомерційних цілях.

Функціонування SQLite базується на безсерверній моделі, що виключає використання клієнт-серверної архітектури. Програмний модуль СУБД інтегрується безпосередньо в ядро розробленого комплексу, функціонуючи всередині його адресного простору без запуску сторонніх служб чи сервісів. Замість мережевих запитів взаємодія з даними реалізується через прямі локальні виклики функцій (API). Такий підхід дозволяє позбутися латентності при транзиті пакетів, суттєво прискорює обробку запитів до таблиць обліку РАВ та оптимізує структуру програмного забезпечення.

Характерною рисою SQLite є консолідація всіх компонентів СУБД (включаючи табличні схеми, індекси, тригери та безпосередньо записи) всередині одного стандартного файлу, що розміщується на локальному диску робочої станції чи сервера. Спрощена архітектура керування паралельним

доступом реалізована шляхом повного монопольного блокування цього файлового сховища безпосередньо перед ініціалізацією та виконанням будь-якої транзакційної операції.

Відповідність фундаментальним критеріям ACID гарантується завдяки генерації спеціалізованого файлу транзакційного журналу. Архітектура СУБД дозволяє декільком незалежним процесам або легковаговим потокам паралельно й безперешкодно виконувати операції зчитування з одного файлу сховища. Натомість модифікація чи внесення нових записів блокується, якщо в цей момент відбуваються інші обчислювальні процеси. У разі виникнення конфлікту доступу операція запису переривається, а викликаючому додатку надходить відповідний код системної помилки або ж ініціюється автоматичний цикл повторних запитів протягом заданого таймауту. Стандартний дистрибутив також містить консольну утиліту sqlite3, яка виступає повноцінною клієнтською частиною для демонстрації можливостей ядра системи. Цей інструмент функціонує на базі інтерфейсу CLI, забезпечуючи пряму взаємодію з локальною базою даних через стандартні системні виклики ОС [5].

Специфіка програмного рушія робить SQLite універсальним інструментом. Вона однаково успішно інтегрується як у компактні вбудовані системи моніторингу, так і в серверні конфігурації, забезпечуючи стабільну роботу з локальними базами даних різного об'єму.

Специфікація та функціональний потенціал SQLite:

- масштабування та об’єми даних: архітектурно закладена можливість оперування сховищами терабайтного масштабу, а також коректна обробка великих текстових полів та бінарних об'єктів (BLOB) гігабайтних розмірів;
- компактність рушія: дистрибутив відрізняється мінімальним розміром кодової бази, де повна збірка займає менше 350 КБ, а базова версія - до 200 Кб;
- продуктивність та API: швидкість обробки типових реляційних транзакцій є вищою порівняно з громіздкими клієнт-серверними аналогами, що в поєднанні з лаконічним інтерфейсом API спрощує написання коду;
- сумісність та надійність: написання ядра на мові ANSI C у поєднанні зі 100-відсотковим тестовим покриттям гарантує стабільність роботи. Платформа

є автономною, не має системних залежностей і підтримує безшовне перенесення між Windows, Linux, macOS та іншими Unix-системами.

4.4.1. Даталогічна архітектура бази даних

На стадії логічного проектування виконується трансформація концептуальної (інфологічної) структури предметної області у даталогічну модель, яка безпосередньо адаптована під функціональні можливості обраної СУБД SQLite. Цей етап конвертації схем прийнято класифікувати як процедуру мапування або відображення моделей. Логічна структура даних. (рисунок 4.4) є концепт описує структуру сховища на логічному рівні з орієнтацією на архітектуру обраної платформи. Процедурі даталогічного моделювання обов'язково передує обґрунтування та вибір конкретної системи керування базами даних. Оскільки будь-яке програмне рішення висуває власні специфічні вимоги до побудови логічних схем, першочерговим завданням є детальний аналіз особливостей СУБД та виявлення факторів, що детермінують фінальну топологію таблиць.

Головним архітектурним чинником тут виступає підтримуваний тип логічної моделі даних. У сучасній інженерній практиці розгортання інформаційно-моніторингових комплексів лідируючі позиції утримують саме реляційні СУБД (зокрема, інтегрована у даному проекті SQLite).

Попри активний розвиток альтернативних мережевих, ієрархічних або об'єктно-орієнтованих концепцій, реляційний підхід залишається найбільш оптимальним для структуризації відомостей про РАВ.

Специфіка фізичної організації даних у цільовому середовищі. На відміну від ранніх файлово-орієнтованих систем керування даними, які розподіляли таблиці, форми та індекси за різними ізольованими файлами, обрана для даного проекту СУБД SQLite консолідує повну інформаційну структуру всередині єдиного локального файлового об'єкта.

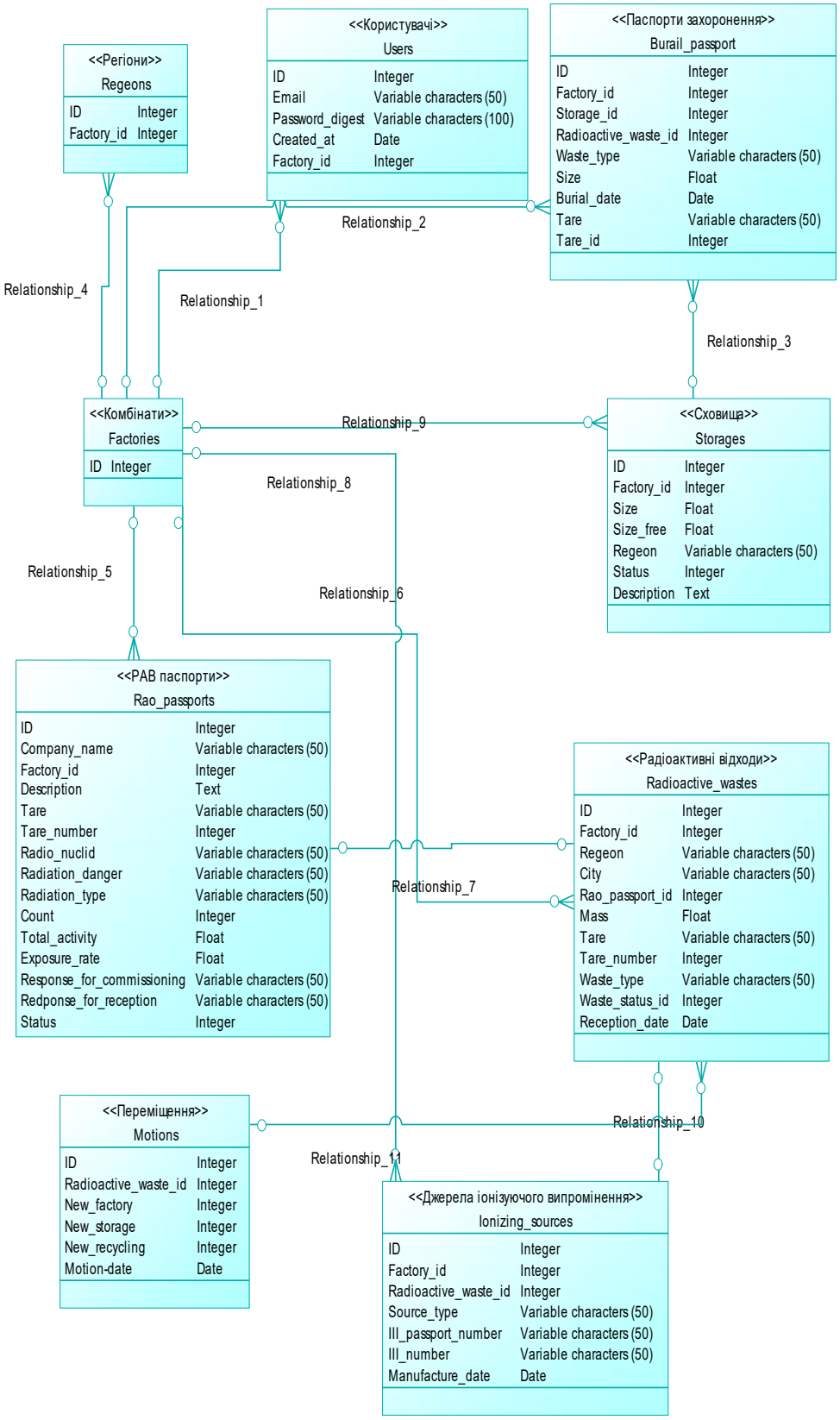


Рисунок 4.4 – Даталогічна модель даних

У цьому монолітному сховищі одночасно перебувають метадані, схеми відношень, індексні масиви та самі облікові записи карт моніторингу РАВ. З огляду на це, етап проектування має враховувати не лише абстрактні правила

побудови реляційних зв'язків, а й архітектурні особливості низькорівневого збереження інформації на фізичному носії [5].

Сюди належать архітектурні ліміти на кількість атрибутів у межах однієї таблиці, граничний розмір кортежу, а також обмеження, що диктуються конкретною файловою системою. Традиційне конструювання реляційного сховища реалізується через механізм послідовних ітерацій, спрямованих на формування оптимальної топології таблиць. Вихідна точка передбачає декомпозицію предметної області до базових універсальних зв'язків. Наступні кроки розробки орієнтовані на створення схем із вищими системними характеристиками. Ця процедура базується на покроковій нормалізації реляційних таблиць, де кожен вищий рівень посилює консистентність даних та нівелює надлишковість інформації. Профільна теорія виокремлює ієрархічну послідовність нормальних форм, яка охоплює рівні від першої до п'ятої нормальної форми (1NF–5NF), включаючи проміжну специфікацію Бойса-Кодда (BCNF). Фіксація відношення на певному рівні нормалізації свідчить про повну відповідність структури математичним критеріям цієї форми. Ключовий архітектурний принцип полягає в тому, що підвищення рівня нормалізації оптимізує схему СУБД, повністю успадковуючи та зберігаючи безпекові властивості попередніх стадій.

4.4.2. Структура таблиць бази даних

Для відображення структури таблиць використовуються рисунки створені в середовищі SQLiteBrowser на основі даталогічної моделі (рисунок 4.3). Усі таблиці мають стандартні поля «created_at» та «updated_at» у яких зберігаються відповідно дати створення та редагування запису в таблиці.

Структура таблиць бази даних представлена у таблицях 4.1 – 4.10.

Таблиця «burial_passports» (таблиця 4.1). Використовується для збереження паспортів захоронення РАВ, що були переміщенні до сховища. При створені запису у цій таблиці, також автоматично створюється запис у таблиці

«Motions», в якій зберігаються усі переміщення РАВ. Первинний ключ id (код паспорту захоронення).

«Id» – порядковий номер запису паспорту захоронення, «factory_id» – номер фабрики до якої відноситься РАВ, «storage_id» – порядковий номер сховища до якого надійде РАВ, «radioactive_waste_id» – порядковий номер РАВ, що буде захоронено, «size» – об'єм ум.од., «burial_date» – ,дата захоронення, «tare» – тип тари у якій захоронено РАВ, «tare_id» – порядковий номер тари.

Таблиця 4.1. Структура таблиці "burial_passport"

ID	Integer
factory_id	Integer
storage_id	Integer
radioactive_waste_id	Integer
waste_type	Integer
size	Integer
burial_date	Date
tare	Varchar(255)
created_at	DateTime
updated_at	DateTime

Таблиця «factories» (таблиця 4.2). У цій таблиці зберігаються усі спеціалізовані комбінати які зареєстровані в системі. Ця таблиця є ключовою в системі так більшість записів з інших таблиць відображають відношення записів до конкретного спеціалізованого комбінату. Первинний ключ id (код фабрики).

«Id» – порядковий номер комбінату, «name» – назва комбінату.

Таблиця 4.2. Структура таблиці «factories»

Id	Integer
name	Varchar(255)

Таблиця «ionizing_sources» (таблиця 4.3). В цій таблиці зберігаються данні, що відносяться до джерел іонізуючого випромінення. Первинний ключ id (код джерела іонізуючого випромінення).

«Id» – порядковий номер запису ДІВ, «radioactive_waste_id» – номер РАВ(необхідно для зв'язку з таблицею «radioactive_wastes»), «factory_id» – порядковий номер комбінату до якого належить ДІВ, «source_type» – тип джерела іонізуючого випромінення, «iii_passport_number» – номер паспорту на ДІВ, «iii_number» – номер ДІВ , «manufacture_date» – дата виготовлення ДІВ.

Таблиця 4.3. Структура таблиці "ionizing_sources"

Id	Integer
factory_id	Integer
radioactive_waste_id	Integer
source_type	Varchar(255)
iii_passport_number	Varchar(255)
iii_number	Varchar(255)
manufacture_date	Date
created_at	DateTime
updated_at	DateTime

В таблиці "radioactive_wastes" (таблиця 4.4) зберігаються дані, що відносяться до всіх РАВ, які відносяться до спеціалізованих комбінатів.

Таблиця 4.4. Структура таблиці «radioactive_wastes»

Id	Integer
factory_id	Integer
regeon	Varchar(255)
city	Varchar(255)
rao_passport_id	Integer
mass	Float
tare	Varchar(255)
tare_number	Integer
waste_type	Integer
waste_status_id	Integer
reception_date	Date
created_at	DateTime
updated_at	DateTime

Первинний ключ id (код радіоактивного відходу) – порядковий номер запису РАВ, «factory_id» – порядковий номер комбінату до якого належить ДІВ, «regeon» – регіон з якого отримано РАВ, «city» – місто з якого отримано РАВ, «rao_passport_id» – номер паспорту прийняття на РАВ, «mass» – маса/об'єм

РАВ, «tare» – тип тари у якій зберігається РАВ на комбінаті, «tare_number» – порядковий номер тари, «waste_type» – тип відходу, якщо значення поля «waste_type» дорівнює 1, це означає, що запис відноситься до ТРВ, значення 2 і 3 відповідно відображають ДІВ та РРВ «waste_status_id» – статус відходу, «reception_date» – дата прийому РАВ.

У таблиці “rao_passports” (таблиця 4.5) зберігаються дані які збираються при прийомі РАВ з підприємства до спеціалізованого комбінату. Первинний ключ id (код РАВ паспорту).

Таблиця 4.5. Структура таблиці «rao_passports»

Id	Integer
factory_id	Integer
company_name	Varchar(255)
description	Text
tare	Varchar(255)
tare_number	Integer
radio_nuclid	Varchar(255)
radiation_danger	Varchar(255)
radiation_type	Varchar(255)
count	Integer
total_activity	Float
exposure_rate	Float
response_for_commissioning	Varchar(255)
response_for_reception	Varchar(255)
created_at	DateTime
status	Integer
updated_at	DateTime

«Id» – порядковий номер запису РАВ паспорту, «factory_id» – порядковий номер комбінату до якого належить ДІВ, «company_name» – назва підприємства з якого було прийнято РАВ, «description» – опис РАВ, «tare» – тип тари у якій зберігається РАВ на комбінаті, «tare_number» – порядковий номер тари, «radio_nuclid» – радіонуклідний склад, «radiation_danger» – вид радіаційної небезпеки, «count» – кількість РАВ, «total_activity» – сумарна активність, «exposure_rate» – експозиційна доза опромінення на відстані 1 м, «response_for_commissioning» відповідальний за прийом, «response_for_reception» – відповідальний за здачу, «status» – статус РАВ.

У таблиці “regeons” (таблиця 4.6) зберігаються регіони обслуговування на території України. Первинний ключ id (код регіону) – порядковий номер запису регіону, «factory_id» – номер фабрики до якої відноситься регіон, «name» – назва регіону.

Таблиця 4.6. Структура таблиці «regeons»

Id	Integer
name	Varchar(255)

Первинний ключ id (код сховища) в таблиці “storages” (таблиця 4.7) – порядковий номер сховища, «factory_id» – порядковий номер комбінату до якого відноситься сховище, «name» – назва сховища, «size» – об’єм сховища ум.од., «size_free» – вільний об’єм, «regeon» – регіон де територіально знаходиться сховище, «status» – статус сховища, «description» – опис.

Таблиця 4.7. Структура таблиці «storages»

Id	Integer
factory_id	Integer
name	Varchar(255)
size	Float
size_free	Float
regeon	Varchar(255)
status	Varchar(255)
description	Text
created_at	DateTime
updated_at	DateTime

Ключ id (код користувача) в таблиці “users” (таблиця 4.8) – порядковий номер користувача, «factory_id» – порядковий номер комбінату на якому працює користувач, «name» – ім’я користувача, «email» адреса електронної скриньки користувача., «password_digest» – поле використовується для аутентифікації користувача, «admin» – відображає статус.

Таблиця 4.8. Структура таблиці «users»

Id	Integer
factory_id	Integer
name	Varchar(255)
email	Varchar(255)

password_digest	Varchar(255)
admin	Boolean
created_at	DateTime
updated_at	DateTime

В таблиці “waste_status” (таблиця 4.9) ключ “Id” – порядковий номер статусу, “status_name” – назва статусу, передбачено два типи статусів “Прийнято на зберігання” та “Захоронено”.

Таблиця 4.9. Структура таблиці “waste_status”

Id	Integer
status_name	Varchar(255)

В таблиці “motions” (таблиця 4.10) зберігаються дані по всім переміщенням які здійснювались над РАВ.

«Id» – порядковий номер переміщення, «radioactive_waste_id» – поле у якому зберігається ідентифікаційний номер РАВ, «new_factory» – поле у якому зберігається ідентифікаційний номер комбінату куди було виконано переміщення, «new_storage» – поле у якому зберігається ідентифікаційний номер сховища до якого було переміщено РАВ, «new_recycling» – опис переміщення РАВ яке було відправлено на переробку, «motion_date» – дата переміщення. При прийнятті РАВ до спеціалізованого комбінату або захороненні до сховища, автоматично створюється запис у цій таблиці.

Таблиця 4.10. Структура таблиці «motions»

Id	Integer
radioactive_waste_id	Integer
new_factory	Varchar(255)
new_storage	Varchar(255)
new_recycling	Varchar(255)
motion_date	date

У четвертому розділі розроблено інформаційна системи обліку та моніторингу переміщення радіоактивних відходів. За допомогою побудови діаграми прецедентів (Use Case Diagram) чітко розмежовано ролі користувачів

системи (оператора та диспетчера). Це дозволило деталізувати бізнес-логіку системи, зокрема процеси реєстрації контейнерів, відстеження логістичних маршрутів та формування звітів екологічної безпеки. Описано структуру класів програми із застосуванням принципів об’єктно-орієнтованого проектування. Визначено ключові класи-сутності (контейнери, ізотопний склад, рейси переміщення, сховища) та зв’язки між ними (агрегація, композиція), що дозволило мінімізувати зв’язність коду та забезпечити інкапсуляцію критичних даних.

5. ОПИС РОБОТИ З СИСТЕМОЮ

Розроблена система інвентаризації радіоактивних відходів являє собою кросс-платформений програмний продукт, який може бути розгорнутий на будь-який операційній системі.

5.1. Інсталяція та системні вимоги

Для того щоб запустити систему на локальному ПК, необхідно в першу чергу встановити Ruby на ваш ПК. При розробці системи інвентаризації була використана версія 2.0, тому для уникнення непередбачених труднощів з запуском системи, ліпше за все буде встановити саме цю версію. За посиланням на головну сторінку Ruby, можна завантажити версію Ruby 2.0, для найпоширеніших операційних систем: Windows, Mac OS або Ubuntu.

Після успішної інсталяції Ruby, необхідно встановити модуль bundler, за допомогою команди «gem install bundler» в терміналі або командній строці. Менеджмент залежностей та модулів у межах обраного технологічного стеку реалізовано за допомогою пакетного менеджера RubyGems. Даний інструментарій задає уніфікований формат дистрибуції програмних компонентів та розширень у вигляді автономних пакетів (гемів). Функціонал системи орієнтований на автоматизацію процесів інсталяції, оновлення та видалення бібліотек коду, а також забезпечує інтеграцію додатка із глобальним репозиторієм для їхнього централізованого пошуку та хостингу. Після того як команда буде запущена, Ruby автоматично встановить усі додаткові модулі які використовуються в проекті і зберігаються у файлі Gemfile.

Коли усі модулі успішно встановлено, необхідно, перебуваючи в корневій директорії проекти, виконати команду «rails server», яка запустить локальний веб-сервер, що постачається у комплекті з Ruby on Rails. Після виконання усіх операцій, проект буде доступний за адресою localhost:3000.

5.2. Сценарії роботи користувача з системою

Після того як користувач перейшов за адресою веб-сервісу, він бачить перед собою форму входу, у яку необхідно ввести корпоративній email та пароль від профілю (рисунок 5.1). Системою не передбачено автоматичну реєстрацію нових користувачів, тому для реєстраціє нового користувача необхідно звернутися до адміністратора ресурсу.

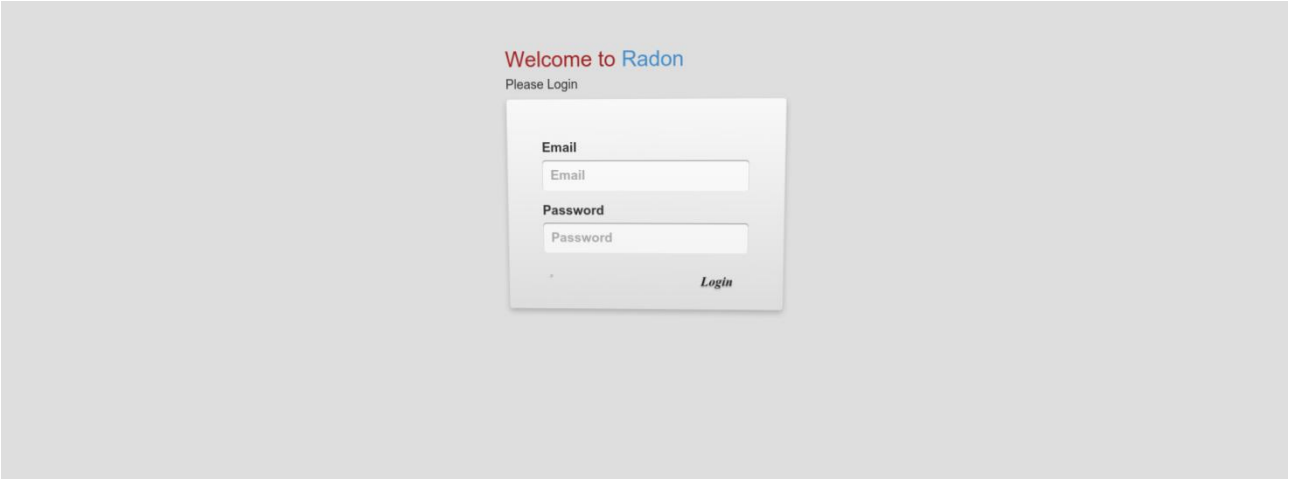


Рисунок 5.1 – Форма входу

У системі використовується хеш-функція, що зветься bcrypt, для незворотнього шифрування пароля у вигляді хеша. Завдяки цьому ми унеможливлюємо доступ до сайту зловмисникам які можуть взломати базу даних.

Після успішного входу до систему користувач потрапляє на сторінку користувача, на якій відображаються дані профілю, зокрема список областей які обслуговує спеціалізований комбінат (рисунок 5.2). У верхній частині сторінки відображається навігаційна панель, що забезпечую навігацію користувача у системі.

Пункт меню «Статистика» включає в себе такі підпункти:

- прийом по областям;
- прийом по сховищам;
- прийом по підприємствам.

Пункт меню «Створити» включає в себе такі підпункти:

- РАВ паспорт;
- ТРВ;
- РРВ;
- джерело іонізуючого випромінення;
- нове сховище;
- паспорт захоронення;
- переміщення РАВ;
- зміна комбінату;
- зміна сховища;
- на переробку.

Пункт меню «Перегляд звітів» включає в себе такі підпункти:

- РАВ паспорт;
- тверді радіоактивні відходи;
- рідкі радіоактивні відходи;
- джерела іонізуючого випромінення;
- сховища;
- паспорта захоронення.

Також у верхній навігаційній панелі присутні посилання «Radon», «Налаштування» та «Вийти»,

Перехід по першому посиланні перенаправляє користувача на домашню сторінку. Перехід по посиланню «Налаштування» направляє користувача на сторінку налаштування профілю, де він може змінити ім’я та аватар у профілю. Перехід по посиланню «Вийти» закриваю поточну сесію роботи користувача, та перенаправляє його до форми входу до системи.

Після того як співробітники підприємства прибувають на підприємство яке бажає здати радіоактивні відходи на переробку або зберігання до спеціалізованого комбінату, повинен бути складений паспорт прийняття РАВ.

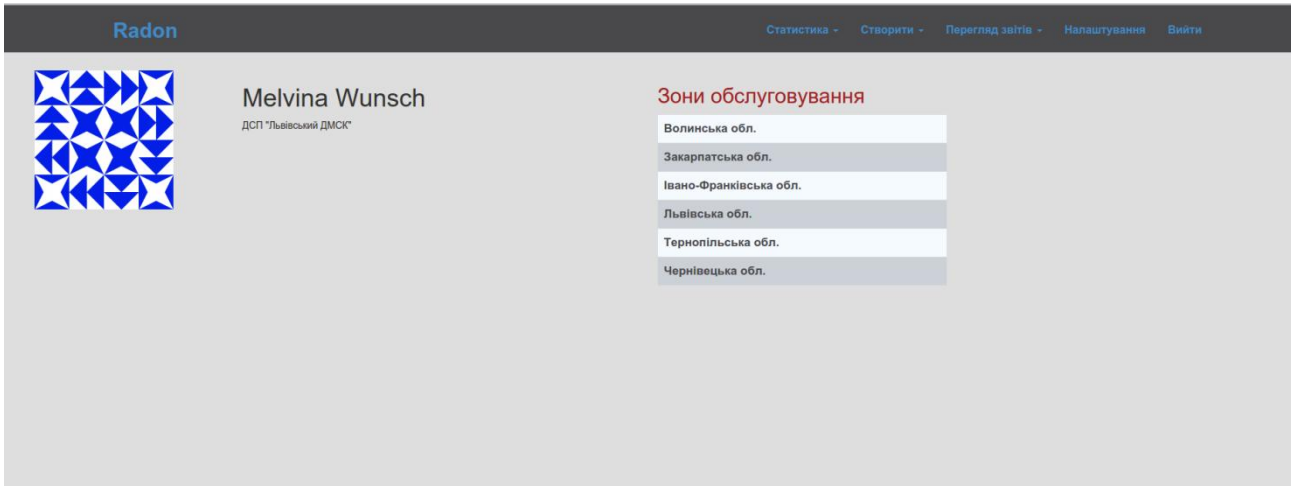


Рисунок 5.2 – Профіль користувача

В паспорт заносяться такі дані: назва компанії яка здає РАВ на зберігання, стислий опис РАВ, вид тари у якій буде зберігатися відходи, порядковий номер контейнеру для зберігання, радіонуклідний склад, група радіаційної небезпеки, тип випромінення, кількість відходів, сумарна активність РАВ, потужність експозиційної дози на відстані 0.1 м. від поверхні, відповідальний за сдачу та відповідальний за прийом РАВ.

Для того щоб створити новий паспорт прийняття РАВ, користувач повинен перейти у меню «Створити->РАВ паспорт», після чого заповнити усі необхідні поля форми і натиснути кнопку «Створити» (рисунок 5.3), після чого користувач буде пере направлений на сторінку з перегляду звіту по усім РАВ паспортам (рисунок 5.9). Якщо якісь з обов’язкових полів не були заповнені або заповнені невірно, користувачу буде виведено повідомлення про помилку з вказанням усіх некоректно заповнених полів форми.

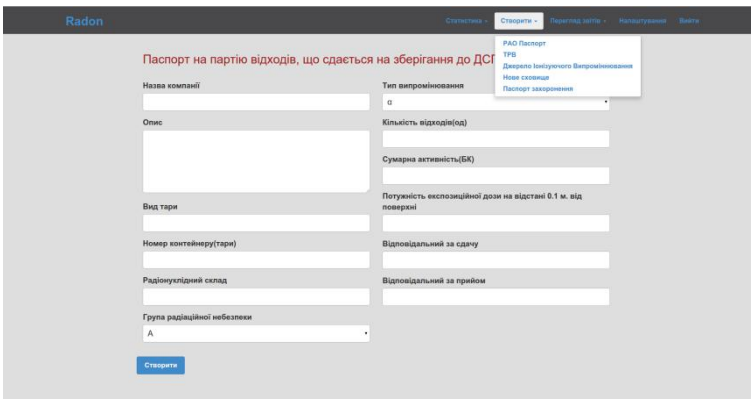


Рисунок 5.3 – Створення нового РАВ паспорту

Для того щоб зареєструвати нові тверді радіоактивні відходи на спеціалізованому комбінаті, користувач повинен перейти у меню «Створити->ТРВ», після чого заповнити усі необхідні поля форми і натиснути кнопку «Створити» (рисунок 5.4).

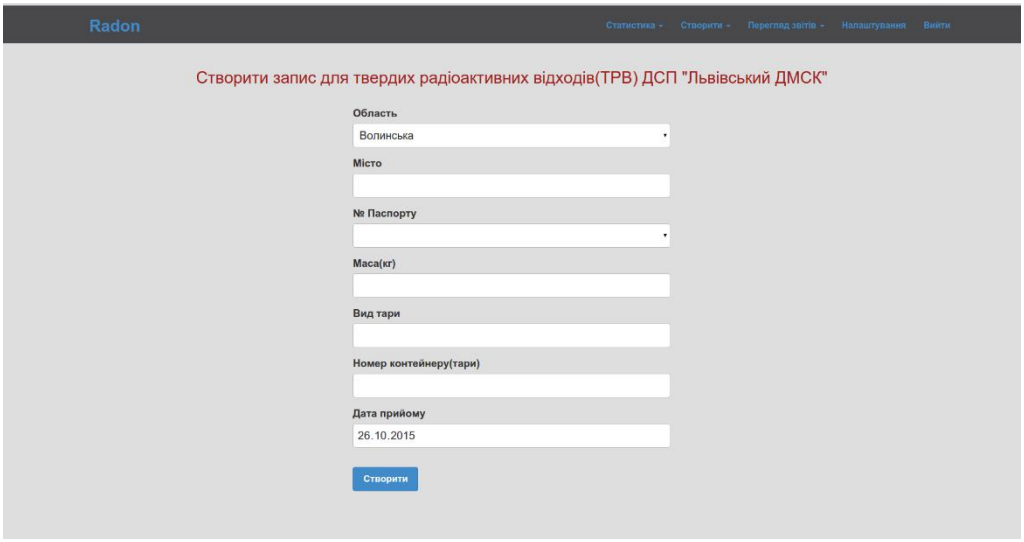


Рисунок 5.4 – Створення нового ТРВ на спецкомбінаті

Для того щоб зареєструвати нове джерело іонізуючого випромінення на спеціалізованому комбінаті, користувач повинен перейти у меню «Створити->Джерело Іонізуючого Випромінення», після чого заповнити усі необхідні поля форми і натиснути кнопку «Створити» (рисунок 5.5). Так я к кожне джерело іонізуючого випромінення має паспорт виробництва та свій унікальний код такі поля як «№ Паспорту на ДІВ», «№ ДІВ», «Дата виробництва», необхідні для зберігання цих даних.

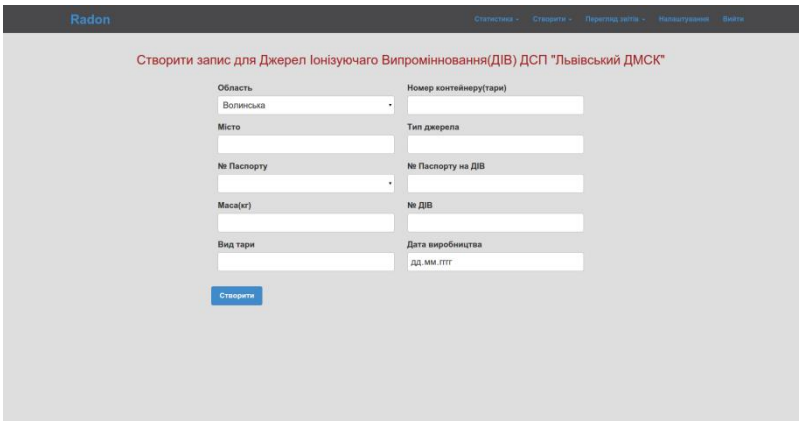


Рисунок 5.5 – Створення нового ДІВ на спецкомбінаті

Якщо заплановано побудову нового сховища і користувачу необхідно внести його до системи, користувач може зробити це перейшовши до пункту меню «Створити->Сховище», після чого заповнити усі необхідні поля форми і натиснути кнопку «Створити» (рисунок 5.6).

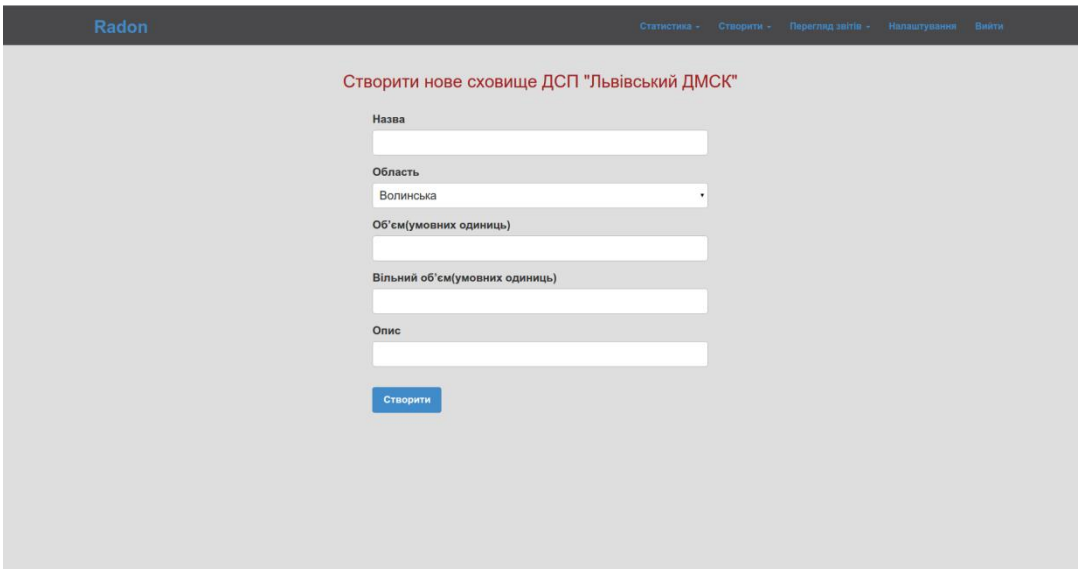


Рисунок 5.6 – Сторінка створення нового сховища

Для того, щоб створити паспорт захоронення радіоактивного відходу, що тимчасово зберігається на спеціалізованому комбінаті, користувач повинен перейти до пункту меню «Створити ->Паспорт захоронення».

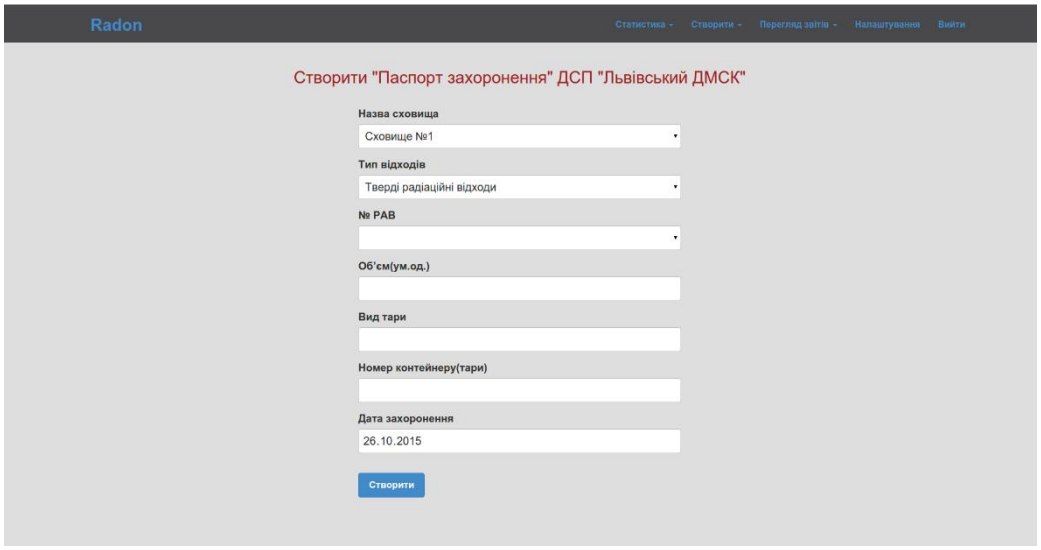


Рисунок 5.7 – Створити новий паспорт захоронення

Для того щоб зареєструвати новий рідкий радіоактивний відхід на спеціалізованому комбінаті, користувач повинен перейти у меню «Створити->РРВ», після чого заповнити усі необхідні поля форми і натиснути кнопку «Створити» (рисунок 5.8).

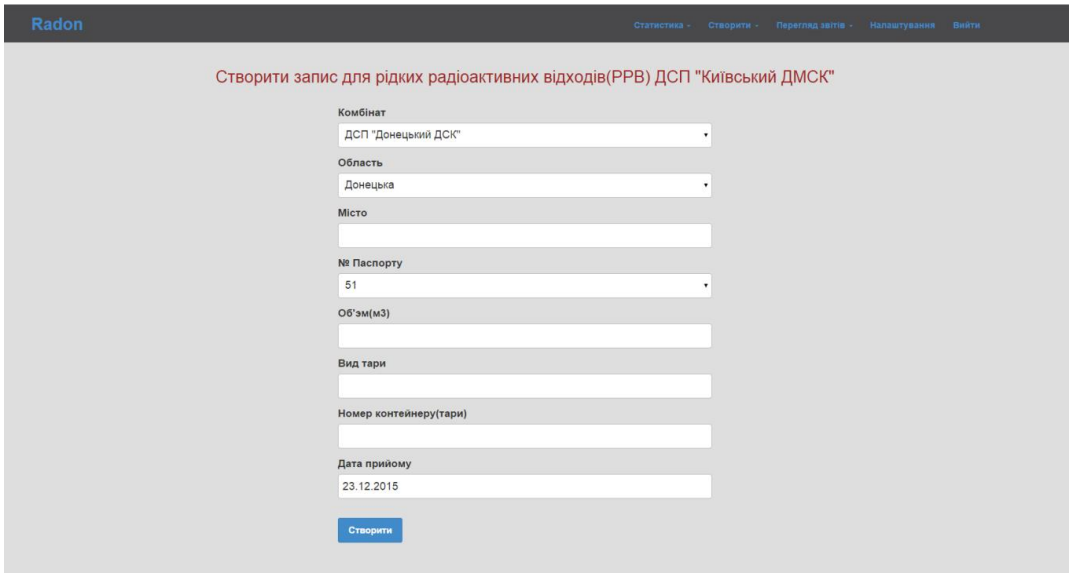


Рисунок 5.8 – Створити новий РРВ

Користувач має можливість переглядати зведені дані в табличному вигляді, що зберігаються у базі даних. Для цього він повинен перейти до пункту меню «Перегляд звітів» і обрати підпункт, який його цікавить рисунок 5.9 – 5.13. Якщо користувач є головним адміністратором він має можливість обрати з нападаючого списку будь-який з семи спеціалізованих комбінатів, щоб продивитися статистику по ньому.

Для того щоб роздрукувати звіт користувач повинен натиснути кнопку «Друк», після чого користувачеві буде запропоновано зберегти звіт у PDF форматі.

Для того щоб продивитися записи які не відображаються на перші сторінці користувач повинен скористатися навігаційної понеллю, що розташована над таблицею з записами.

Натиснувши на посилання “Edit” користувач переходить на сторінку редагування РАВ паспорта (рисунок 5.9).

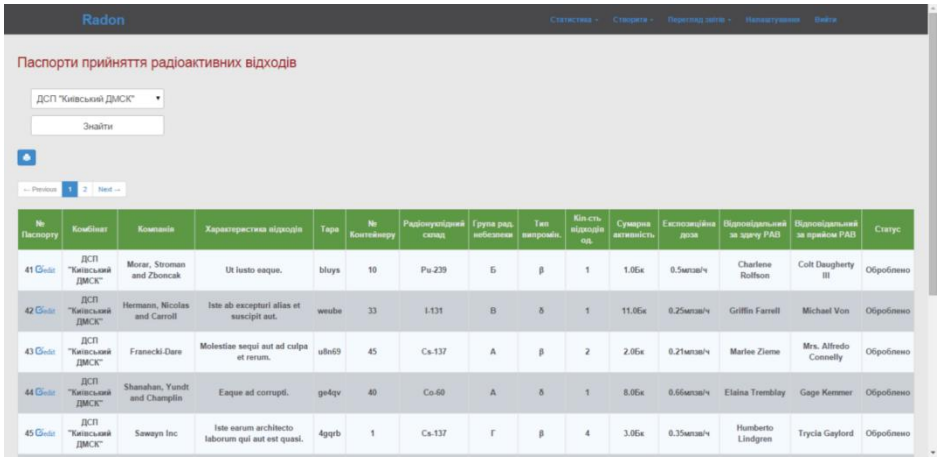


Рисунок 5.9 – Перегляд зведених даних по РАВ паспортам

У системі передбачена можливість редагувати збережені РАВ паспорти, для цього користувач повинен перейти за посиланням «Edit», що знаходиться під порядковим номером паспорта, зробити необхідна правки та натиснути кнопку «Створити». Після чого він буде пере направлений на сторінку з даними по РАВ паспортам.

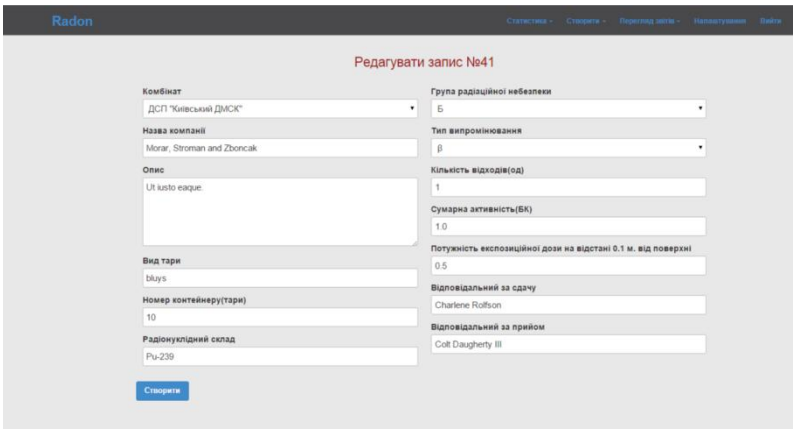


Рисунок 5.10 – Редагування паспорту РАВ

Для перегляду зведених даних по твердих радіоактивних відходів, користувач повинен перйти до пункту меню «Перегляд звітів->Тверді радіоактивні відходи», після чого буде відображена таблиця з даними по ТРВ (рисунок 5.11). Якщо користувач є супер адміністратором він має можливість обрати будь-який комбінат з випадаючого списку, натиснути кнопку «Знайти», після чого будуть відображені дані по обраному комбінату.

Користувач має можливість відфільтрувати записи по даті прийняття РАВ. Для цього він повинен обрати початкову та кінцеву дату пошику і натиснути кнопку «Знайти».

Для того щоб роздрукувати таблицю користувач повинен натиснути кнопку «Друк», та вибрати місце на ПК куди буде збережено файл у PDF форматі.

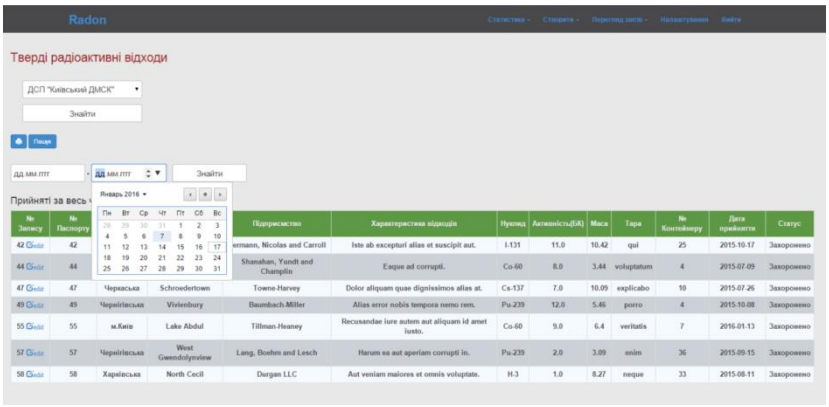


Рисунок 5.11 – Перегляд зведених даних по твердих радіоактивних відходів

Натиснувши на посилання “Edit” користувач переходить на сторінку редагування ТРВ. На сторінці редагування також представлені дані про переміщення конкретного ТРВ. У системі передбачено три типи переміщення: переміщення між комбінатами, переміщення між сховищами та переміщення на переробку (рисунок 5.12). Після виконаних правок користувач повинен натиснути кнопку «Зберегти», для збереження змінених даних.

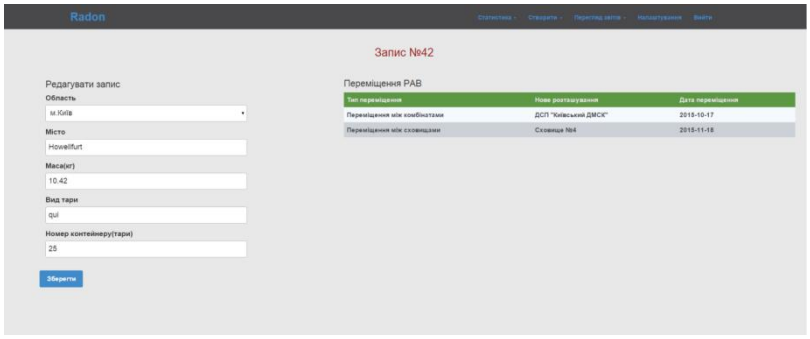


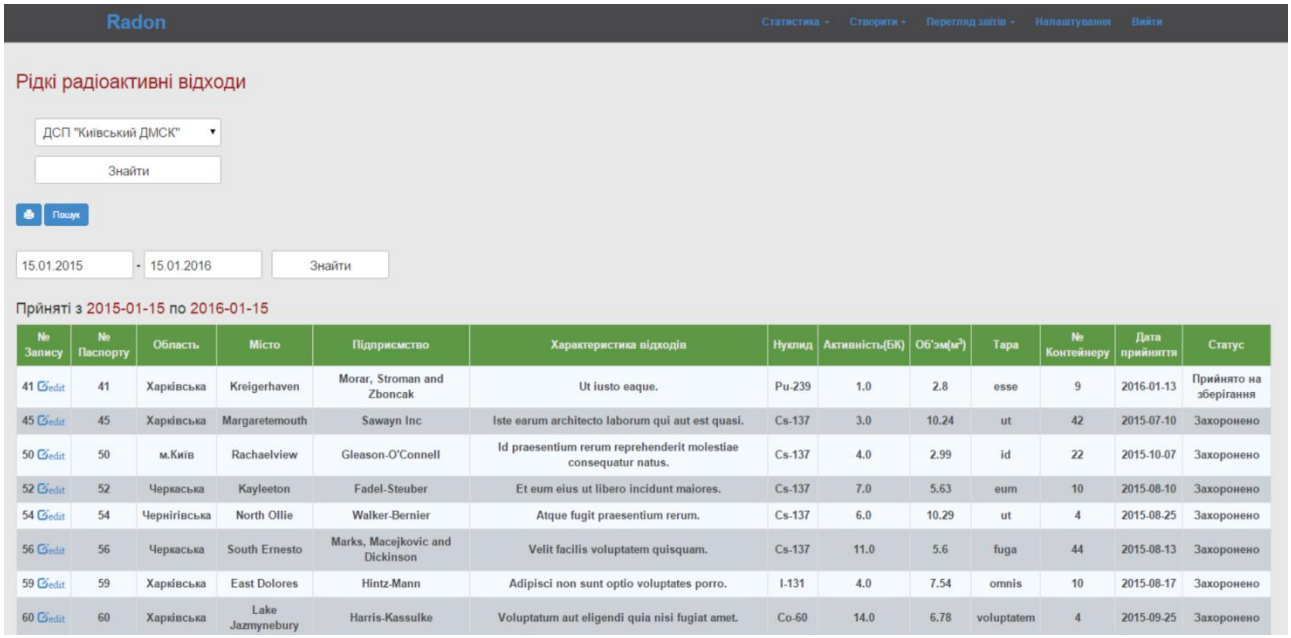
Рисунок 5.12 – Сторінка редагування по твердих радіоактивних відходів

Для перегляду зведених даних по рідким радіоактивним відходам, користувач повинен перйти до пункту меню «Перегляд звітів->Рідкі радіоактивні відходи», після чого буде відображена таблиця з даними по РРВ (рисунок 5.13). Якщо користувач є головним адміністратором він має можливість обрати будь-який комбінат з випадаючого списку, натиснути кнопку «Знайти», після чого будуть відображені дані по обраному комбінату.

Користувач має можливість відфільтрувати записи по даті прийняття РАВ. Для цього він повинен обрати початкову та кінцеву дату пошику і натиснути кнопку «Знайти».

Для того щоб роздрукувати таблицю користувач повинен натиснути кнопку «Друк», та вибрати місце на ПК куди буде збережено файл у PDF форматі.

Натиснувши на посилання “Edit” користувач переходить на сторінку редагування РРВ. На сторінці редагування також представлені дані про переміщення конкретного РРВ. У системі передбачено три типи переміщення: переміщення між комбінатами, переміщення між сховищами та переміщення на переробку. Після виконаних правок користувач повинен натиснути кнопку «Зберегти», для збереження змінених даних (рисунок 5.14).



№ Запису	№ Паспорту	Область	Місто	Підприємство	Характеристика відходів	Нуклід	Активність(БК)	Об'єм(м³)	Тара	№ Контейнеру	Дата прийняття	Статус
41 Edit	41	Харківська	Kreigerhaven	Morar, Stroman and Zboncak	Ut iusto eaque.	Pu-239	1.0	2.8	esse	9	2016-01-13	Прийнято на зберігання
45 Edit	45	Харківська	Margaretemouth	Sawayn Inc	Iste earum architecto laborum qui aut est quasi.	Cs-137	3.0	10.24	ut	42	2015-07-10	Захоронено
50 Edit	50	м.Київ	Rachaelview	Gleason-O'Connell	Id praesentium rerum reprehenderit molestiae consequatur natus.	Cs-137	4.0	2.99	id	22	2015-10-07	Захоронено
52 Edit	52	Черкаська	Kayleeton	Fadel-Steuber	Et eum eius ut libero incidunt maiores.	Cs-137	7.0	5.63	eum	10	2015-08-10	Захоронено
54 Edit	54	Чернігівська	North Ollie	Walker-Bernier	Atque fugit praesentium rerum.	Cs-137	6.0	10.29	ut	4	2015-08-25	Захоронено
56 Edit	56	Черкаська	South Ernesto	Marks, Macejkovic and Dickinson	Velit facilis voluptatem quisquam.	Cs-137	11.0	5.6	fuga	44	2015-08-13	Захоронено
59 Edit	59	Харківська	East Dolores	Hintz-Mann	Adipisci non sunt optio voluptates porro.	I-131	4.0	7.54	omnis	10	2015-08-17	Захоронено
60 Edit	60	Харківська	Lake Jazmynebury	Harris-Kassulke	Voluptatum aut eligendi quia nisi fugiat amet.	Co-60	14.0	6.78	voluptatem	4	2015-09-25	Захоронено

Рисунок 5.13 – Перегляд зведених даних по по рідких радіоактивних відходах

Для перегляду зведених даних по джерелам іонізуючого випромінення, користувач користувач повинен перейти до пункту меню «Перегляд звітів->Джерела іонізуючого випромінення», після чого буде відображена таблиця з даними по ДІВ рисунок 5.15. Якщо користувач є головним адміністратором він має можливість обрати будь-який комбінат з випадаючого списку, натиснути кнопку «Знайти», після чого будуть відображені дані по обраному комбінату.

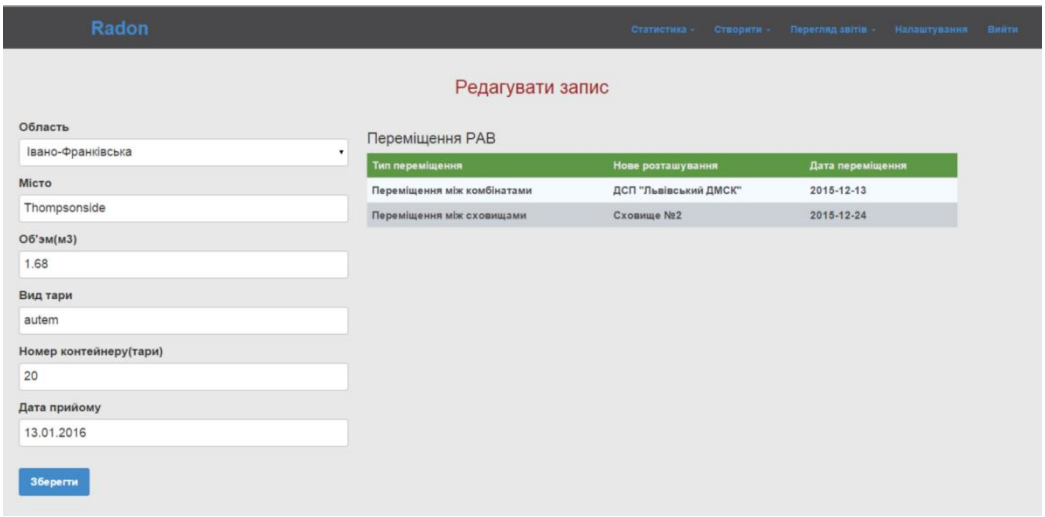


Рисунок 5.14 – Сторінка редагування по рідких радіоактивних відходах

Користувач має можливість відфільтрувати записи по даті прийняття РАВ. Для цього він повинен обрати початкову та кінцеву дату пошику і натиснути кнопку «Знайти».

Для того щоб роздрукувати таблицю користувач повинен натиснути кнопку «Друк», та вибрати місце на ПК куди буде збережено файл у PDF форматі.

Натиснувши на посилання “Edit” користувач переходить на сторінку редагування ДІВ. На сторінці редагування також представлені дані про переміщення конкретного ДІВ. У системі передбачено три типи переміщення: переміщення між комбінатами, переміщення між сховищами та переміщення на переробку. Після виконаних правок користувач повинен натиснути кнопку «Зберегти», для збереження змінених даних (рисунок 5.15).

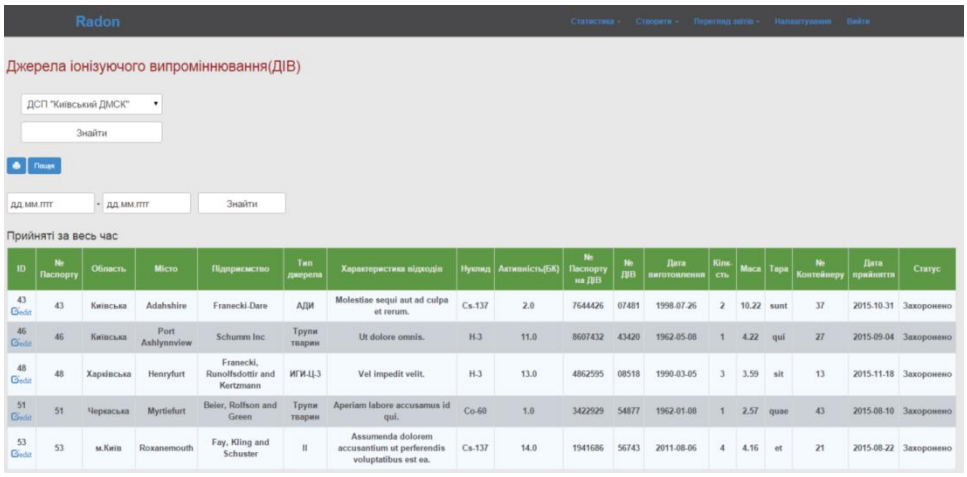


Рисунок 5.15 – Перегляд зведених даних по ДІВ

Для того щоб подивитися інформацію по сховищам, що відносяться до поточного спеціалізованого комбінату, користувач повинен перейти до пункту меню «Перегляд звітів->Сховища», після чого буде пере направлений на сторінку з таблицею даних по сховищам (рисунок 5.16).

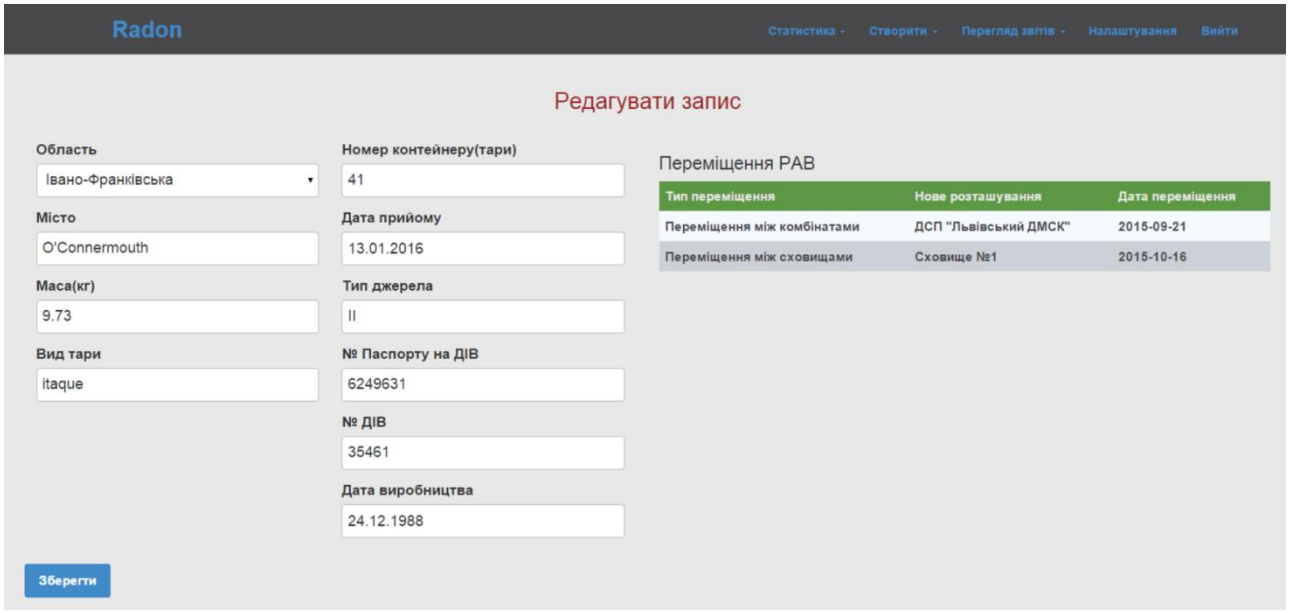


Рисунок 5.16 – Сторінка редагування ДІВ

Якщо користувач є головним адміністратором він має можливість продивитися інформацію по сховищам які належать до будь якого спеціалізованого комбінату.

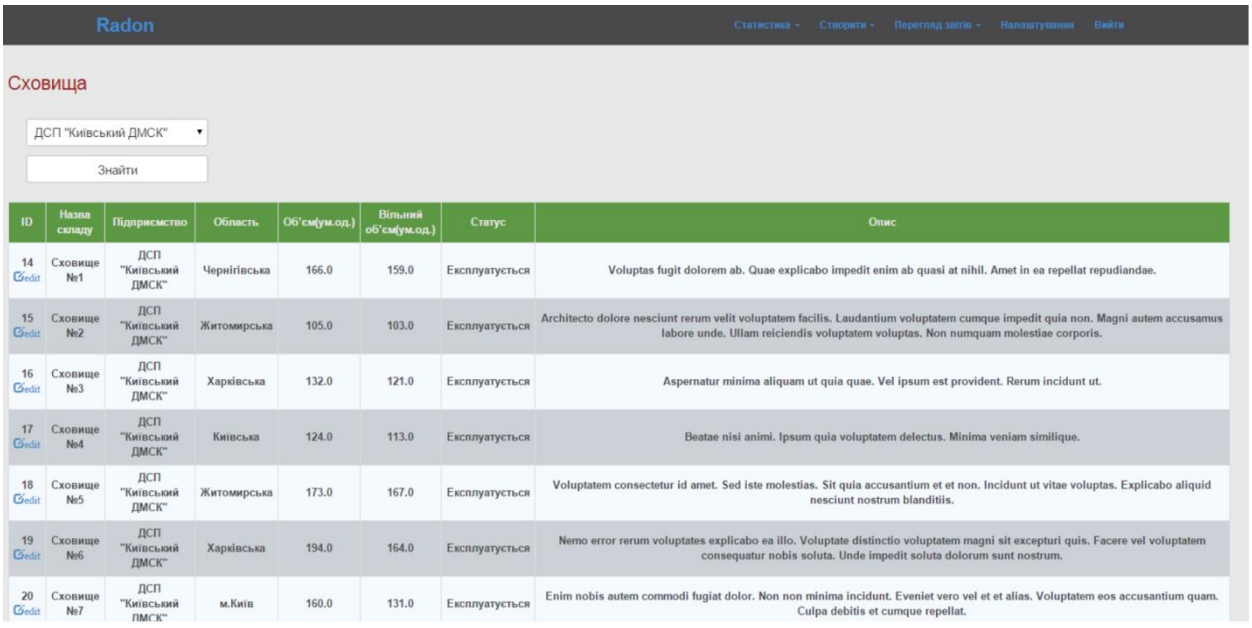


Рисунок 5.17 – Перегляд зведених даних по сховищам

Для перегляду зведених даних по паспортам захоронення, користувач повинен перйти до пункту меню «Перегляд звітів->Паспорта захоронення», після чого буде відображена таблиця з даними по всім паспортам захоронення (рисунок 5.18). Якщо користувач є головним адміністратором він має

можливість обрати будь-який комбінат з випадаючого списку, натиснути кнопку «Знайти», після чого будуть відображені всі дані по обраному комбінату.

Користувач має можливість відфільтрувати записи по даті створення паспорту захоронення. Для цього він повинен обрати початкову та кінцеву дату пошику і натиснути кнопку «Знайти».

Для того щоб роздрукувати таблицю користувач повинен натиснути кнопку «Друк», та вибрати місце на ПК куди буде збережено файл у PDF форматі.

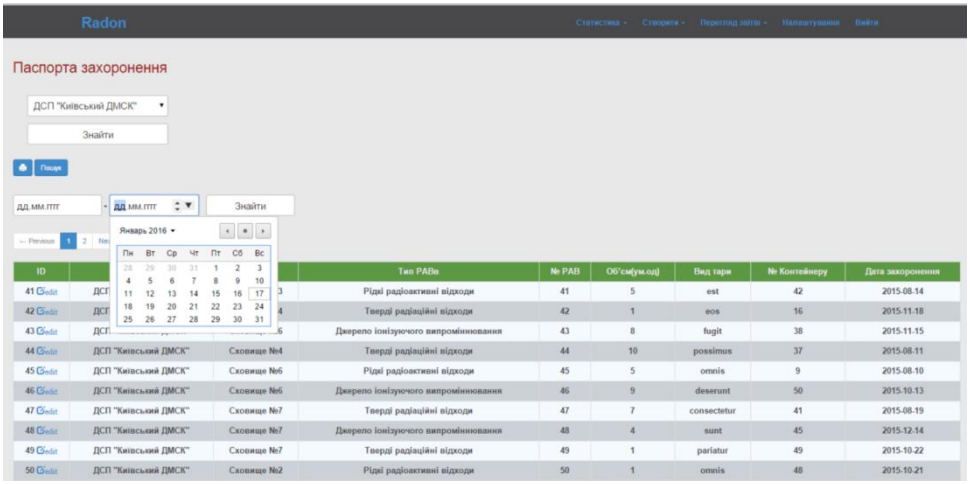


Рисунок 5.18 – Перегляд зведених даних паспортів захоронення

Для того, щоб відредагувати паспорт захоронення, користувач повинен натиснути посилання «Edit», після чого він буде направлений на сторінку редагування (рисунок 5.19). Після того як необхідні дані були відредаговані користувач повинен натиснути кнопку «Створити», для збереження даних.

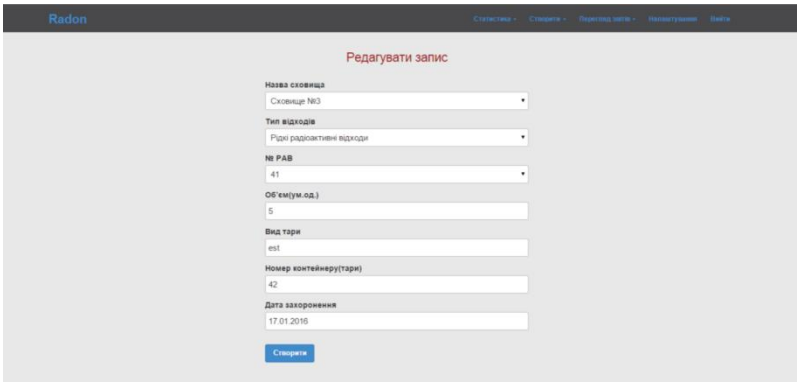


Рисунок 5.19 – Редагування паспорту захоронення

Для того щоб виконати переміщення між спеціалізованими комбінатами, користувач повинен перейти до пункту меню «Створити->Переміщення РАВ->Зміна комбінату», після чого буде відкрито сторінку з формою переміщення, обрати необхідний РАВ, вказати новий комбінат та дату переміщення, після чого нажати кнопку «Створити» (рисунок 5.20).

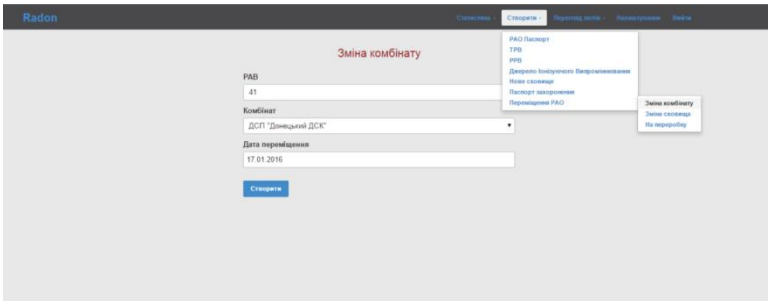


Рисунок 5.20 – Зміна комбінату

Для того щоб виконати переміщення між сховищами, користувач повинен перейти до пункту меню «Створити->Переміщення РАВ->Зміна сховища», після чого буде відкрито сторінку з формою переміщення, обрати необхідний РАВ, вказати нове сховище та дату переміщення, після чого нажати кнопку «Створити» (рисунок 5.21).

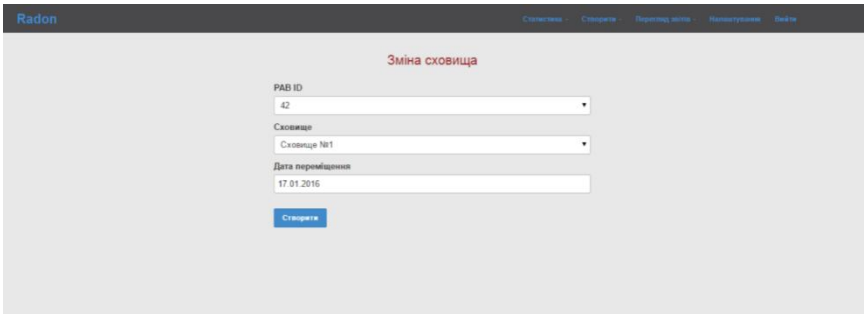


Рисунок 5.21 – Зміна сховища

Для того щоб виконати переміщення на переробку, користувач повинен перейти до пункту меню «Створити->Переміщення РАВ->На переробку», після чого буде відкрито сторінку з формою переміщення, обрати необхідний РАВ, вказати тип переробки та дату переміщення, після чого нажати кнопку «Створити» (рисунок 5.22).

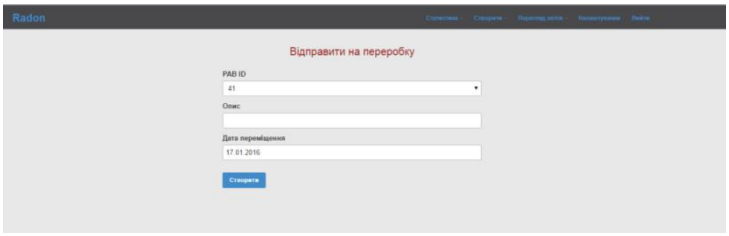


Рисунок 5.22 – Переміщення на переробку

Для зручності сприйняття статистичної інформації і полегшення аналітичних розрахунків, у системі статистичні дані представлені в вигляді таблиць, графіків та діаграм. Для того щоб переглянути статистичні дані, користувач повинен перейти до пункту меню «Статистика», та обрати потрібний розділ, що представлен у системі.

На рисунках 5.23 – 5.27 представлена сторінка на якій відображаються дані по кількості прийнятих РАВ, з кожної області, що відносяться до комбінату. Обравши початкову та кінцеву дату користувач може продивитися статистику за певний період часу.

Для того, щоб продивитися статистику прийому РАВ по місяцях, користувач повинен натиснути кнопку «Детальна статистика», після чого буде відкрита сторінка з графіками по ТРВ, РРВ, ДІВ.

На рисунках 5.23, 5.24 представлено дані за рік по областям, що відносяться до обраного комбінату.

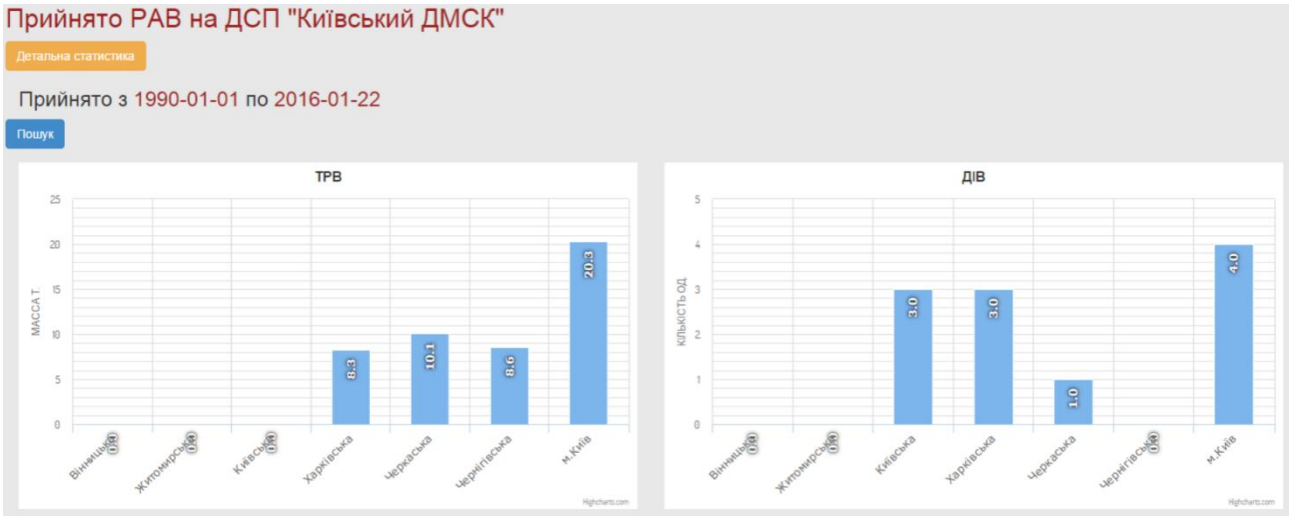


Рисунок 5.23 – Діаграми кількості РАВ з кожної області

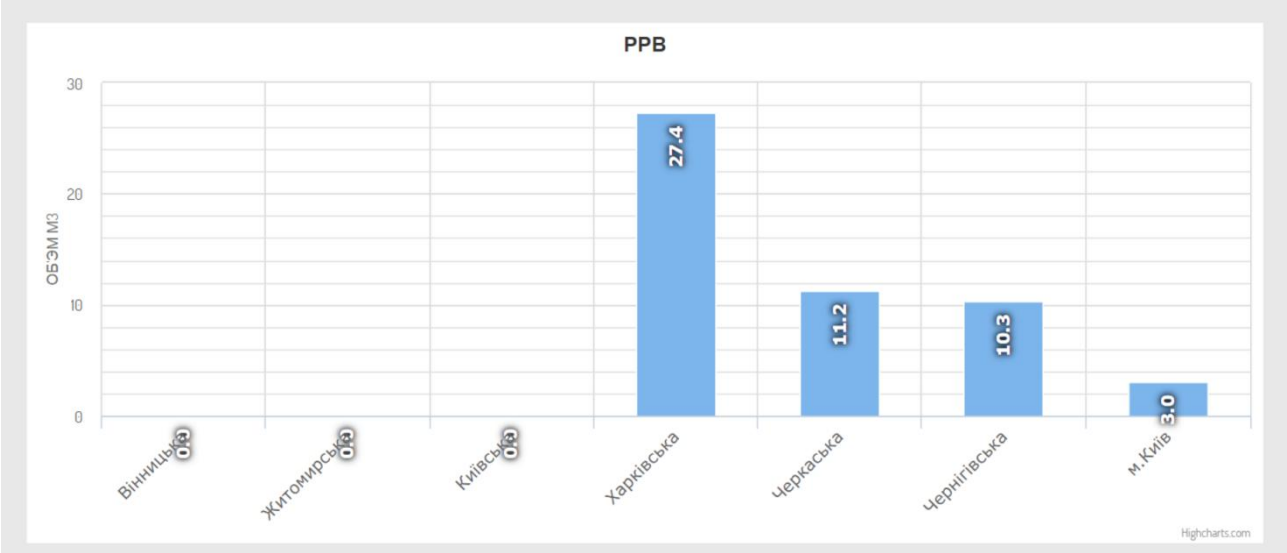


Рисунок 5.24 – Діаграми кількості РРВ з кожної області

Якщо користувач є головним адміністратор він може обрати будь який спеціалізований комбінат для перегляду статистику по ньому, та вибрати будь-який рік по якому необхідно отримати статистику.

На графіку по осі ОХ відображаються 12 місяців, по осі ОУ одиниці виміру РАВ, в залежності від графіку ТРВ, РРВ або ДІВ. По кожному місяцю представлено стовпчасту діаграму, та детальну інформацію при наведенні курсору мищі.

Рисунок 5.25 – Графіки прийому ТРВ за останній рік

Користувач може роздрукувати графік натиснувши кнопку «Print», або зберегти у вигляді картинки у PNG, JPG, SVG форматах.

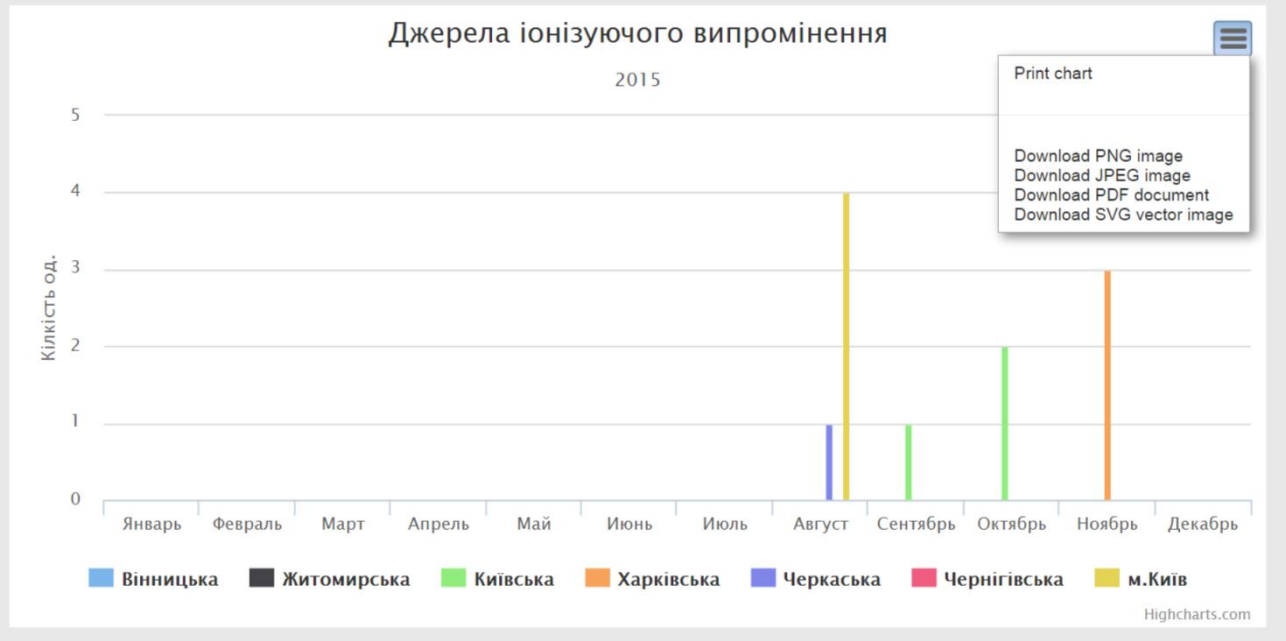


Рисунок 5.26 – Графіки прийому ДІВ за останній рік

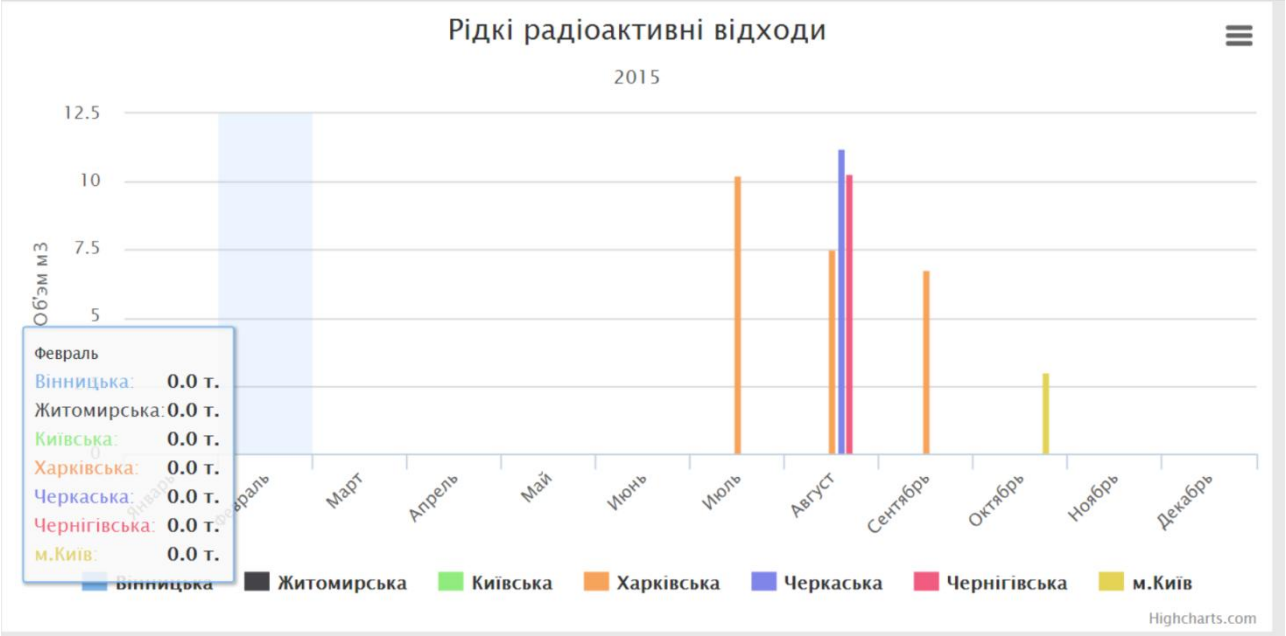


Рисунок 5.27 – Графіки прийому РРВ за останній рік

У системі представлена можливість перегляди статистики безпосередньо по спеціалізованому комбінату (рисунок 5.28). Для цього користувач повинен перейти до пункту меню «Статистика-> Прийом по підприємствам», після чого буде відкрита сторінка з статистикою по поточному підприємством. Якщо користувач є головним адміністратором він може обрати будь-який спеціалізований комбінат для перегляду статистики.

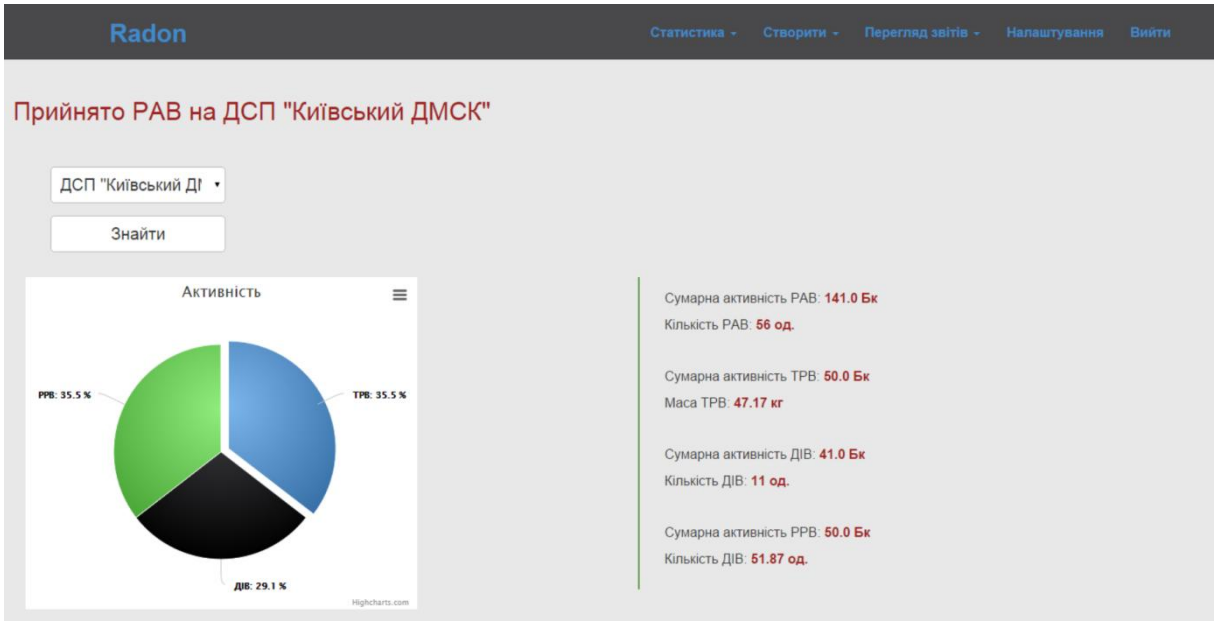


Рисунок 5.28 – Характеристики РАВ на спеціалізованому комбінаті

Однією з найважливіших функцій системи є можливість моніторингу рівня наповненості сховищ захоронення, та перегляд статистики по РАВ, що зберігаються в тому чи іншому сховищі (рисунок 5.29). Для цього користувач повинен перейти до пункту меню “Статистика -> Прийом по сховищам”.

Якщо користувач є головним адміністратором, він може обрати будь-який спеціалізований комбінат до якого відносяться сховища, та продивитися інформацію по ним.

Рисунок 5.29 – Відображення рівня наповненості сховищ

Натиснувши на посилання «Сховище №__», користувач потрапляє на сторінку детального представлення даних по обраному сховищу. На цій сторінці, (рисунок 5.30), представлені графіки та діаграми, що відображають

порівняльну характеристику ТРВ, РРВ та ДІВ, за параметрами займаємого об’єму, рівню радіоактивного випромінення, та кількості відходів. У правій частині сторінки представлені посилання на РАВ, що зберігаються у сховищі.

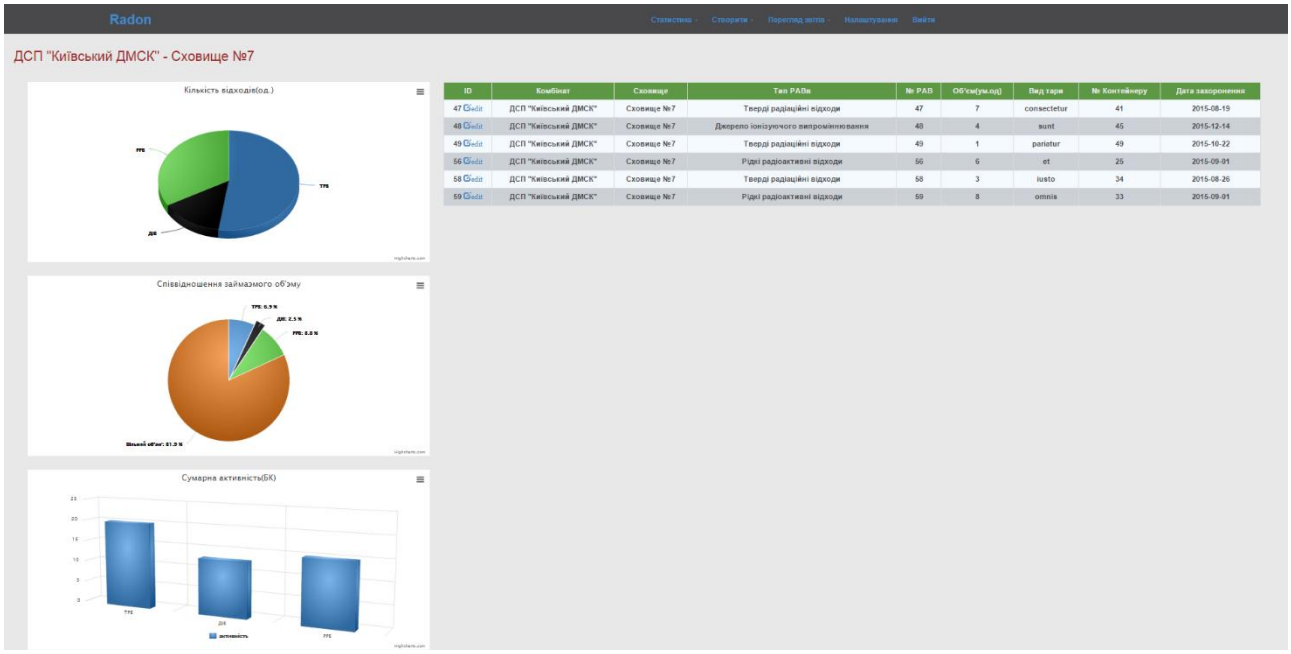


Рисунок 5.30 – Детальна статистика по РАВ, які зберігаються у сховищі

У лівій частині сторінки представлено три види графіків: кількість відходів (од.), співвідношення займаємого об’єму, сумарна активність (бк). Користувач має можливість зберегти кожен з графіків у форматі PDF, JPEG, PNG або SVG, натиснувши на ярлик випадаючого меню у верхньому правому куті.

ВИСНОВКИ

У процесі створення інформаційної системи обліку радіоактивних відходів на території України був проведений аналіз специфіки поводження з відходами не ядерного паливного циклу. Зокрема, досліджено джерела їх утворення в медичній та промисловій сферах, вивчено чинні нормативно-правові вимоги до їх тимчасового зберігання, а також проаналізовано технологічні бар'єри та логістичні виклики, що виникають під час їх переробки, транспортування та остаточного приповерхневого захоронення.

Були опрацьовані напрями діяльності та принципи роботи Державної корпорації Державного спеціалізованого підприємства "Об'єднання "Радон", що дало змогу створити програмний продукт який задовольняє потребам головного інформаційно-аналітичного центру корпорації.

У процесі розробки системи за допомогою сучасних веб-технологій були вирішені проблеми які виникали в результаті взаємодії двох різних мов програмування, а саме Ruby та Java Script. Вирішення цих проблем дало змогу реалізувати систему яка відповідає майже всім вимогам сучасної веб-розробки.

Розроблений програмний модуль є основним компонентом інформаційно-аналітичної системи, що призначений для автоматизації процесів оперативного збору, централізованого оброблення, верифікації та безпечної передачі даних щодо обліку, поточного стану та динаміки руху радіоактивних відходів та джерел іонізуючого випромінювання на території України.

У результаті створення системи інвентаризації радіоактивних відходів було реалізовано наступні пункти:

- розроблена система має зручний інтерфейс користувача;
- створена зручна адміністративна частина що дозволяє легко керувати даними які зберігаються в системі;
- усі статистичні дані представлені у зручному для сприйняття вигляді, а саме у вигляді таблиць та графіків;

- фіксація, супровід та візуалізація етапів транспортування (переміщення) РАВ, а також реєстрація параметрів їхньої технологічної обробки, кондиціювання та утилізації із збереженням історії змін для забезпечення аудиту безпеки;
- інтерактивна візуалізація та подання оперативної інформації в зручному для аналізу вигляді через систему графіків, діаграм та кросс-таблиць, що мінімізує когнітивне навантаження на оператора та пришвидшує оцінку радіоекологічного стану.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Hartl M. Ruby on Rails Tutorial / M. Hartl – 3-d edition – Addison-Wesley Professional, 2015 – 300 p.
2. Фултон Х. Програмування на мові Ruby = The Ruby Way / пер. з англ. – 2-е вид. – М.: ДМК Пресс, 2007. – 688 с.
3. Флэнаган Д., Мацумото Ю. Мова програмування Ruby = The Ruby Programming Language / пер. с англ. – 1-е вид. – СПб.: Питер, 2011. – 496 с.
4. Гайджарги В.І. Основи проектування та використання баз даних : Навч. посібник – 2-ге вид., виправл. і допов / В. І. Гайджарги, О. А. Дацюк. – К.: ІВЦ “Видавництво “Політехніка”, ТОВ “Фірма “Періодика”, 2004. – 256 с.
5. Дакетт Дж. JavaScript і jQuery. Інтерактивна веб-розробка: пер. з англ. Київ : [Видавництво], 2016. – 688 с.
6. Дейт К. Дж. Вступ до систем баз даних / К. Дж. Дейт ; пер. з англ. – 7-е вид. – СПб. : Вильямс, 2001. – 1072 с.
7. Закон України «Про поводження з радіоактивними відходами» від 30.06.1995 № 255/95-ВР // Відомості Верховної Ради України. — 1995. — № 27. — Ст. 198. [Редакція від 23.12.2010. Підстава — № 2856-VI].
8. Закон України «Про використання ядерної енергії та радіаційну безпеку» від 08.02.1995 № 40/95-ВР // Відомості Верховної Ради України. — 1995. — № 12. — Ст. 74. [Редакція від 17.11.2009. Підстава — № 1718-VI].,
9. Про схвалення Стратегії поводження з радіоактивними відходами в Україні : Розпорядження Кабінету Міністрів України від 19 серп. 2009 р. № 990-р. *Урядовий кур'єр*. 2009. № 173.
10. Розпорядження Кабінету Міністрів України «Про схвалення Концепції Програми забезпечення безпечного зберігання відпрацьованих високоактивних джерел іонізуючого випромінювання» від 18.01.2006 № 18-р // Офіційний вісник України. — 2006. — № 4. — Ст. 166..
11. SQLAlchemy Documentation. SQLAlchemy : website. URL: sqlalchemy.org (дата звернення: 18.04.2026).

12. React Native Documentation : website. URL: reactnative.dev (дата звернення: 17.04.2025).

13. PostgreSQL: Documentation. *PostgreSQL*. URL: https://www.postgresql.org/docs/ (date of access: 19.04.2026).

14. Стрихалюк Б. М. Теорія побудови та протоколи інфокомунікаційних мереж: конспект лекцій. Львів: Видавництво Львівської політехніки, 2017. 121 с.

15. Flanagan D. JavaScript: The Definitive Guide. Seventh Edition: довідник. O'Reilly Media, 2020. 1153 с.

16. Kolban, N. (2014). *Kolban's Book on IBM Business Process Manager*. IBM Redbooks.

17. Фрімен Е., Робсон Е. Head First. Патерни проектування: довідник. Харків: Фабула, 2019. 672 с

18. Фрімен Е., Робсон Е. Ізучаем JavaScript: пер. з англ. Львів: О'Рейлі, 2017. – 640 с.

19. Хавербеке М. Виразний JavaScript. Сучасне введення в програмування: пер. з англ. 3-є вид. Електронне видання, 2019. – 416 с.

20. Фленаган Д. JavaScript: Повне керівництво. 7-е вид. : пер. з англ. Київ : Діалектика, 2021. – 720 с.

21. Bibeault B., Katz Y. jQuery in Action. 3rd ed. Shelter Island : Manning Publications, 2015. – 480 p.

22. Chaffer J., Swedberg K. Learning jQuery. 4th ed. Birmingham: Packt Publishing, 2013. – 428 p.

ДОДАТОК А

Система інвентаризації радіоактивних відходів на території України

Програмна реалізація

НТУУ “КПІ ім. Ігоря Сікорського”

Аркушів 5

2026

Програмна частина реалізована за допомогою сучасного веб фреймворку Ruby on Rails, який в свою чергу реалізований на мові програмування Ruby. Середовище розробки − RubyMine.

Даний програмний продукт слідує архітектурній схемі модель-представлення-контролер (MVC), яка реалізує розділення між логікою предметної області від логіки вводу та логіки представлення, зв’язаної з графічним інтерфейсом користувача. Цей шаблон поділяє систему на три частини: модель даних, вигляд даних та керування.

```
<% provide(:title, 'Sign In') %>
<div class="col-sm-8 col-sm-offset-2">
  <div class="login-wrapper">
    <div class="x-wrapper">
      <div class="y-wrapper">
        <div class="title-wrapper">
          <h2>Welcome to <%= link_to 'Radon', root_path, class: 'link'%></h2>
          <h4>Please Login</h4>
        </div>
        <div class="input-box">
          <%= form_for(:session, url: sessions_path) do |f| %>
             <% flash.each do |key, value| %>
                <%= content_tag(:div, value, class: "alert alert-#{key}") %>
             <% end %>
             <%= f.label :email %>
             <%= f.text_field :email, placeholder: "Email" %>
             <%= f.label :password %>
             <%= f.password_field :password, placeholder: "Password" %>
             <div class="button-wrapper">

           <%= f.submit "Login", class:"login-btn" %> <span>&raquo;</span>
             </div>
          <% end %>
        </div>
      </div>
    </div>
  </div>
</div>
//Приклад контролеру для аутентифікації користувача
class SessionsController < ApplicationController
  def new
  end
  def create
    user = User.find_by(email: params[:session][:email].downcase)
    if user && user.authenticate(params[:session][:password])
      session[:current_user_id] = user.id
      redirect_to user
    else
      flash[:danger] = 'Invalid email/password combination'
      redirect_to sign_in_path
    end
  end
```

```
  def destroy
    @current_user = session[:current_user_id] = nil
    redirect_to root_path
  end
end
//Приклад контролеру для роботи з РАВ паспортами
class RaoPassportsController < ApplicationController
  respond_to :html, :js
  def new
    @rao_passport = RaoPassport.new
  end

  def create
    @rao_passport = RaoPassport.new(passport_params)
    if @rao_passport.save
      flash[:success] = 'Паспорт було успішно створено'
      redirect_to rao_passports_url
    else
      render 'new'
    end
  end

  def index
    name = params[:factory_name] || current_user.factory.name
    @rao_passports = Factory.find_by(name: name).rao_passports.paginate(page: params[:page])
    respond_to do |format|
      format.html
      format.pdf
    end
 end

   def update
    @rao_passport = RaoPassport.find(params[:id])
    if @rao_passport.update_attributes(passport_params)
      flash[:success] = "Запис оновлено"
      redirect_to rao_passports_url
    else
      render 'edit'
    end
  end
  def edit
    @rao_passport = RaoPassport.find(params[:id])
  end
end
private
def passport_params

params.require(:rao_passport).permit(:factory_id, :company_name, :description, :tare, :tare_number, :radio_
nuclid, :radiation_type, :radiation_danger,
                                :count, :total_activity, :exposure_rate, :response_for_commissioning, :response_for_
reception,
                                :status)
End
// Приклад контролеру для роботи з ТРВ
class RadioactiveWastesController < ApplicationController
def new
    @solid_waste = RadioactiveWaste.new
  end

  def create
```

```
    @solid_waste = RadioactiveWaste.new(solid_waste_params)
    if @solid_waste.save
      @solid_waste.rao_passport.update(status: 1)
      @solid_waste.motions.create(new_factory: params[:radioactive_waste][:factory_id], motion_date:
params[:radioactive_waste][:reception_date])
      flash[:success] = "Новий запис ТРВ був успішно створений"
      redirect_to radioactive_wastes_url
    else
    render 'new'
    end
  end

  def index
    name = params[:factory_name] || current_user.factory.name
    @solid_wastes = Factory.find_by(name: name).radioactive_wastes.where(waste_type: 1).paginate(page:
params[:page])
  end

  def update
    @solid_waste = RadioactiveWaste.find(params[:id])

if  @solid_waste.update_attributes(solid_waste_params)
      flash[:success] = "Запис оновлено"
      redirect_to radioactive_wastes_path
    else
      render 'edit'
    end
  end

  def edit
    @solid_waste = RadioactiveWaste.find(params[:id])
    @motions = Motion.all.where(radioactive_waste_id: params[:id])
  end

  def search
    @start_date = params[:start_date]
    @end_date = params[:end_date]
    @solid_wastes = current_user.factory.radioactive_wastes.where(reception_date:
"#{@start_date}".."#{@end_date}", waste_type: 1)
  end
end

private

def solid_waste_params

  params.require(:radioactive_waste).permit(:factory_id, :regeon, :mass, :city, :rao_passport_id, :tare,
                                        :tare_number, :waste_type, :waste_status_id, :reception_date)
End
// Приклад контролеру для роботи з РРВ
class FluidWastesController < ApplicationController

  def new
    @fluid_waste = RadioactiveWaste.new
  end

  def index
    name = params[:factory_name] || current_user.factory.name
    @fluid_wastes = Factory.find_by(name: name).radioactive_wastes.where(waste_type: 3).paginate(page:
params[:page])
```

```
  end

  def create
    @fluid_wastes = RadioactiveWaste.new(fluid_waste_params)
    @fluid_wastes.motions.create(new_factory: params[:radioactive_waste][:factory_id], motion_date:
params[:radioactive_waste][:reception_date])
    if @fluid_wastes.save
      @fluid_wastes.rao_passport.update(status: 1)

      flash[:success] = "Новий запис ТРВ був успішно створений"
      redirect_to fluid_wastes_url
    else
      render 'new'
    end
  end

  def search
    @start_date = params[:start_date]
    @end_date = params[:end_date]
    @fluid_wastes = current_user.factory.radioactive_wastes.where(reception_date:
"#{@start_date}".."#{@end_date}", waste_type: 3)
  end

  def edit
    @fluid_waste = RadioactiveWaste.find(params[:id])
    @motions = Motion.all.where(radioactive_waste_id: params[:id])
  end
end

def fluid_waste_params
  params.require(:radioactive_waste).permit(:factory_id, :regeon, :mass, :city, :rao_passport_id, :tare,
                                    :tare_number, :waste_type, :waste_status_id, :reception_date)
end
```

-10-

```
// Приклад контролеру для роботи з ДІВ
class IonizingSourcesController < ApplicationController
  def new
    @ionizing_source = IonizingSource.new
    @ionizing_source.build_radioactive_waste
  end

  def index
    name = params[:factory_name] || current_user.factory.name
    @ionizing_sources = Factory.find_by(name: name).ionizing_sources.paginate(page: params[:page])
  end

  def create
    @ionizing_source = IonizingSource.new(ionizing_source_params)
    if @ionizing_source.save
      @ionizing_source.radioactive_waste.rao_passport.update(status: 1)
      @ionizing_source.radioactive_waste.motions.create(new_factory:
params[:ionizing_source][:radioactive_waste_attributes][:factory_id], motion_date:
params[:ionizing_source][:radioactive_waste_attributes][:reception_date])
      flash[:success] = "Новий запис ДІВ був успішно створений"
      redirect_to ionizing_sources_path
    else
      render 'new'
    end
```

```
end

 def update
   @ionizing_source = IonizingSource.find(params[:id])
   if @ionizing_source.update_attributes(ionizing_source_params)
     flash[:success] = "Запис оновлено"
     redirect_to ionizing_sources_path
   else
     render 'edit'
   end
 end

 def edit
   @ionizing_source = IonizingSource.find(params[:id])
   waste_id = @ionizing_source.radioactive_waste.id
   @motions = Motion.all.where(radioactive_waste_id: waste_id)
 end

 def search
   @start_date = params[:start_date]
   @end_date = params[:end_date]
   @ionizing_sources = current_user.factory.radioactive_wastes.where(reception_date:
"#{@start_date}".."#{@end_date}", waste_type: 2)
 end
end
private

def ionizing_source_params
 params.require(:ionizing_source).permit(:factory_id, :source_type, :iii_passport_number, :iii_number,
                                         :manufacture_date, radioactive_waste_attributes: [:factory_id, :regeon,
                                                                                           :mass, :city, :rao_passport_id, :tare, :tare_numb
er, :waste_type, :waste_status_id, :reception_date])
End

//Приклад контролеру для представлення графіків та діаграм
class StatisticsController < ApplicationController
 respond_to :html, :js

 def factory_statistic
   name = params[:factory_name] || current_user.factory.name
   @factory = Factory.find_by(name: name)
 end

 def regeons_statistic
   @start_date = params[:start_date] ||DateTime.new(1990).strftime('%Y-%m-%d')
    @end_date = params[:end_date] ||DateTime.now.strftime("%Y-%m-%d")

  name = params[:factory_name] || current_user.factory.name
   @factory = Factory.find_by(name: name)
   @solid_wastes = @factory.radioactive_wastes.where(waste_type: 1,
                                                     reception_date: @start_date..@end_date)
   @fluid_wastes = @factory.radioactive_wastes.where(waste_type: 3,
                                                     reception_date: @start_date..@end_date)
   @ion_wastes = @factory.radioactive_wastes.where(waste_type: 2,
                                                   reception_date: @start_date..@end_date)
 end

 def regeons_detail_statistic
   name = params[:factory_name] || current_user.factory.name
   @factory = Factory.find_by(name: name)
   @year = params[:year] || DateTime.current.year
```

```
  end
end
class StatisticsController < ApplicationController
  respond_to :html, :js

  def factory_statistic
    name = params[:factory_name] || current_user.factory.name
    @factory = Factory.find_by(name: name)
  end
  def regeons_statistic
    # if params[:start_date] == ""
    #   @start_date = '0000-01-01'
    #   @end_date = DateTime.now.strftime("%Y-%m-%d")
    # else
      @start_date = params[:start_date] ||DateTime.new(1990).strftime('%Y-%m-%d')
      @end_date = params[:end_date] ||DateTime.now.strftime("%Y-%m-%d")
   #end
    name = params[:factory_name] || current_user.factory.name
    @factory = Factory.find_by(name: name)
    @solid_wastes = @factory.radioactive_wastes.where(waste_type: 1,
                                                      reception_date: @start_date..@end_date)
    @fluid_wastes = @factory.radioactive_wastes.where(waste_type: 3,
                                                reception_date: @start_date..@end_date)
    @ion_wastes = @factory.radioactive_wastes.where(waste_type: 2,
                                                      reception_date: @start_date..@end_date)
  end

  def regeons_detail_statistic
    name = params[:factory_name] || current_user.factory.name
    @factory = Factory.find_by(name: name)
    @year = params[:year] || DateTime.current.year
  end
```