

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КПІ»

Кафедра

Обчислювальної техніки

КУРСОВА РОБОТА

з «ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ»

на тему: «Система організації вуличних змагань «WorkOUT» »

Студента 2 курсу групи ІО-32
напряму підготовки
6.050102 «Комп'ютерна інженерія»

Попенко Руслан Леонідович

Керівник

Болдак Андрій Олександрович

(прізвище та ініціали)

Доцент кафедри ОТ

(посада, вчене звання, науковий ступінь)

Національна шкала _____

Кількість балів: _____

Оцінка: ECTS _____

м. Київ - 2015 рік

Інв. № підп.	Підп. і дата	Інв. № дубл.	Взаєм. інв. №	Підп. і дата						Ошибка! Неизвестное имя свойства документа.»	Арк. 25
Зм	Арк.	№ докум.	Підпис	Дат							

РОЗДІЛ 1	3
ЗАПИТИ ЗАЦІКАВЛЕНИХ ОСІБ	3
1.1. Введення	3
1.2. Короткий огляд продукту	3
1.4. Функціональність системи	6
1.5. Надійність.....	7
РОЗДІЛ 2	8
РОЗРОБКА ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ	8
2.1. Загальна схема прецедентів.....	8
2.2. Прецеденти для ролі головного адміністратора.....	8
2.3. Діаграма бізнес-сутностей.....	16
2.4. Реляційна модель бази даних	16
2.5. Специфікація таблиць бази даних	17
РОЗДІЛ 3	23
РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ	23
3.1. Реляційно-об'єктне відображення	23
3.2. Специфікація Service класів	36
3.3. Специфікація DAO-класів	36
3.4. Класи контролерів та їх специфікація.....	37
РОЗДІЛ 4	38
ІЛЮСТРАЦІЯ РОБОТИ ПРОГРАМИ.....	38
4.1. Пошук співробітника та перегляд його дій в системі.....	38
СПИСОК ІНФОРМАЦІЙНИХ ДЖЕРЕЛ	41
ДОДАТОК А.....	42
ДОДАТОК Б	43
ДОДАТОК В.....	58

Підп. і дата																																																																					
Взаєм. інв. №																																																																					
Інв. № дубл.																																																																					
Підп. і дата																																																																					
Інв. № підп.																																																																					
<table border="1" style="width:100%; border-collapse: collapse;"> <tr> <td colspan="5"></td> <td colspan="4" style="text-align: center;">6.050102 «Комп'ютерна інженерія»</td> </tr> <tr> <td>Змн.</td> <td>Арк.</td> <td>№ докум.</td> <td>Підпис</td> <td>Дата</td> <td colspan="4"></td> </tr> <tr> <td>Розроб.</td> <td></td> <td>Міхацький</td> <td></td> <td></td> <td colspan="2" rowspan="2">Система збору та обробки інформації</td> <td>Літ.</td> <td>Арк.</td> <td>Акрушів</td> </tr> <tr> <td>Перевір.</td> <td></td> <td>Болдак</td> <td></td> <td></td> <td></td> <td>2</td> <td>72</td> </tr> <tr> <td>Реценз.</td> <td></td> <td></td> <td></td> <td></td> <td colspan="4" rowspan="2" style="text-align: center;">Кафедра Обчислювальної техніки</td> </tr> <tr> <td>Н. Контр.</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>Затверд.</td> <td></td> <td>Болдак</td> <td></td> <td></td> <td colspan="4" rowspan="2" style="text-align: center;">Ошибки! Неизвестное имя свойства документа</td> </tr> <tr> <td>Ст.</td> <td>Арк.</td> <td>№ докум.</td> <td>Підпис</td> <td>Дата</td> </tr> </table>											6.050102 «Комп'ютерна інженерія»				Змн.	Арк.	№ докум.	Підпис	Дата					Розроб.		Міхацький			Система збору та обробки інформації		Літ.	Арк.	Акрушів	Перевір.		Болдак				2	72	Реценз.					Кафедра Обчислювальної техніки				Н. Контр.					Затверд.		Болдак			Ошибки! Неизвестное имя свойства документа				Ст.	Арк.	№ докум.	Підпис	Дата
					6.050102 «Комп'ютерна інженерія»																																																																
Змн.	Арк.	№ докум.	Підпис	Дата																																																																	
Розроб.		Міхацький			Система збору та обробки інформації		Літ.	Арк.	Акрушів																																																												
Перевір.		Болдак						2	72																																																												
Реценз.					Кафедра Обчислювальної техніки																																																																
Н. Контр.																																																																					
Затверд.		Болдак			Ошибки! Неизвестное имя свойства документа																																																																
Ст.	Арк.	№ докум.	Підпис	Дата																																																																	
<div style="display: flex; justify-content: space-between;"> 25 Арк. </div>																																																																					

РОЗДІЛ 1 ЗАПИТИ ЗАЦІКАВЛЕНИХ ОСІБ

1.1. Введення

У даному документі описуються запити зацікавлених осіб, у якості яких виступає: клієнт - правоохоронні органи України.

1.1.1. Мета

Метою документа є визначення основних вимог щодо функціональності та експлуатаційної придатності, а також визначення бізнес-правил та технологічних обмежень стосовно предмета розробки - «Системи збору та аналізу інформації».

1.1.2. Контекст

Перелік вимог, перерахованих у цьому документі, є основою технічного завдання на розробку «Системи збору та аналізу інформації».

1.2. Короткий огляд продукту

Правоохоронні органи - державні органи, які на підставі Конституції та законів України здійснюють правоохоронну діяльність.

Діяльність правоохоронних органів спрямована на забезпечення законності і правопорядку, захист прав та інтересів громадян, попередження, припинення правопорушень, застосування державного примусу або заходів громадського впливу до осіб, які порушили закон і правопорядок.

1.3. Ділові правила і приписи

1.3.1. Призначення системи

Система призначена для збору, зберігання і показу інформації про підозрюваних, злочинців та інших осіб, які тим чи іншим чином причетні до правоохоронних розглядів.

1.3.2. Політика взаємодії з клієнтом

Клієнтами системи можуть бути прокурори, слідчі, судді, криміналісти та інші співробітники правоохоронних органів України.

Політика взаємин системи з клієнтом складається в наданні клієнту інформації про обличчя, які його цікавлять.

Підп. і дата	
Взаєм. інв. №	
Інв. № дубл.	
Підп. і дата	
Інв. № підп	

Зм	Арк.	№ докум.	Підпис	Дат

1.3.3. Характеристика ділового процесу

Управління системою і її контроль здійснюється головним та регіональними адміністраторами, які взаємодіють з керівництвом правоохоронних «органів.»»»

Співробітники правоохоронних органів мають доступ до запису, зміни, читання або видалення інформації з системи відповідно до свого службового становища.

1.3.4. Сценарій призначення / видалення регіонального адміністратора

Назва: призначення / видалення регіонального адміністратора.

Учасники: головний адміністратор, керівництво.

Попередні умови: заявка від керівництва правоохоронних органів на призначення / видалення регіонального адміністратора.

Результат: призначення / видалення регіонального адміністратора.

Основний сценарій:

1. Головний адміністратор отримує заявку від керівництва правоохоронних органів на призначення / видалення регіонального адміністратора.
2. Головний адміністратор призначає / видаляє регіонального адміністратора.

1.3.5. Сценарій реєстрації користувача

Назва: Реєстрація користувача.

Учасники: головний адміністратор, користувач, керівництво правоохоронних органів.

Попередні умови: заявка від керівництва правоохоронних органів на реєстрацію нового користувача в системі.

Результат: реєстрація користувача.

Основний сценарій:

1. Головний адміністратор / регіональний адміністратор отримує заявку від керівництва правоохоронних органів на реєстрацію користувача в системі.
2. Головний адміністратор реєструє користувача.

Підп. і дата	
Взаєм. інв. №	
Інв. № дубл.	
Підп. і дата	
Інв. № підп	

Зм	Арк.	№ докум.	Підпис	Дат

1.3.6. Сценарій видалення користувача

Назва: Видалення користувача.

Учасники: головний адміністратор, користувач.

Попередні умови: заявка від керівництва правоохоронних органів на видалення користувача.

Результат: видалення користувача.

Основний сценарій:

1. Головний адміністратор отримує заявку від керівництва правоохоронних органів на видалення користувача;
2. Головний адміністратор видаляє користувача.

1.3.7. Сценарій створення / видалення групи

Назва: Створення / видалення групи.

Учасники: головний адміністратор, група користувачів.

Попередні умови: заявка від керівництва правоохоронних органів на створення / видалення групи.

Результат: створення / видалення групи.

Основний сценарій:

1. Головний адміністратор отримує заявку від керівництва правоохоронних органів на створення / видалення групи.
2. Головний адміністратор створює / видаляє групу.
3. У разі створення групи, головний адміністратор призначає права доступу для групи.

1.3.8. Сценарій додавання користувача в групу (видалення користувача з групи)

Назва: Додавання користувача в групу (видалення користувача з групи).

Учасники: головний адміністратор, користувач.

Попередні умови: заявка від керівництва про додавання користувача в групу (видалення користувача з групи).

Результат: додавання користувача в групу (видалення користувача з групи).

Підп. і дата	
Взаєм. інв. №	
Інв. № дубл.	
Підп. і дата	
Інв. № підп	

Зм	Арк.	№ докум.	Підпис	Дат

Основний сценарій:

1. Головний адміністратор отримує заявку від керівництва про додавання користувача в групу (видалення користувача з групи).
2. Головний адміністратор додає користувача в групу (видаляє користувача з групи).

1.3.9. Сценарій тимчасового розширення / отримання прав доступу до даних

Назва: тимчасове розширення / отримання прав доступу до даних.

Учасники: начальство, головний адміністратор;

Попередні умови: потрібне залучення нових співробітників у рамках даного правоохоронного розгляду з видачею часткового / повного доступу до даних.

Результат: розширення / отримання прав доступу співробітника / групи співробітників.

Основний сценарій:

1. Звернення начальства до головного адміністратора з метою розширення / видачі прав доступу до даних співробітнику / групі співробітників.
2. Головний адміністратор розширює / видає права доступу співробітнику / групі співробітників.

1.4. Функціональність системи

Основні вимоги до функціональності, пропоновані зацікавленими особами до предмета розробки, відносяться до трьох категорій:

- Головний адміністратор.
- Регіональний адміністратор.
- Співробітник правоохоронних органів.

1.4.1. Можливості головного адміністратора:

- реєстрація та видалення користувачів в системі;
- створення і видалення груп користувачів із системи;
- призначення групам користувачів прав доступу до даних;
- призначення та зміну прав груп у системі;

- запис, зміна та видалення даних у системі.

1.4.2. Можливості регіонального адміністратора:

- створення та видалення груп користувачів;
- реєстрація та видалення користувачів в системі;
- призначення групам користувачів прав доступу до даних.

1.4.3. Можливості співробітника правоохоронних органів

- реєстрація в системі нового досьє;
- запис, зміна та редагування даних в системі (можливість або неможливість даних дій для даного користувача залежить від групи, до якої він належить);
- звернення до начальства про тимчасове / перманентне розширення прав доступу до даних.

1.4.4. Можливості начальства, як окремої групи співробітників

- звернення до головного / регіонального адміністратора з проханням зареєструвати нового / видалити користувача в системі;
- звернення до головного / регіонального адміністратора з проханням видати права новому співробітнику або розширити права зареєстрованого в системі співробітника правоохоронних органів.

1.5. Надійність.

1.5.1. Резервне копіювання і відновлення даних.

Має здійснюватися резервне копіювання баз даних.

1.5.2. Реєстрація дій користувачів в системі.

В цілях безпеки та для забезпечення контролю використання даних і внесень змін, дії всіх користувачів в системі реєструються.

Іне. № підп	Підп. і дата	Іне. № дубл.	Взаєм. іне. №	Підп. і дата	Іне. № підп						Арк.
Зм	Арк.	№ докум.	Підпис	Дат	6.050102 «Комп'ютерна інженерія»					7	

РОЗДІЛ 2

РОЗРОБКА ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ

2.1. Загальна схема прецедентів

Загальна схема прецедентів для ролі головного адміністратора показує можливі послідовності дій актора. Основним видом діяльності головного адміністратора є створення, видалення та редагування груп користувачів. Також головний адміністратор може здійснювати пошук серед окремих користувачів та переглядати історію дій кожного користувача. Схема прецедентів представлена на рис. 2.1.

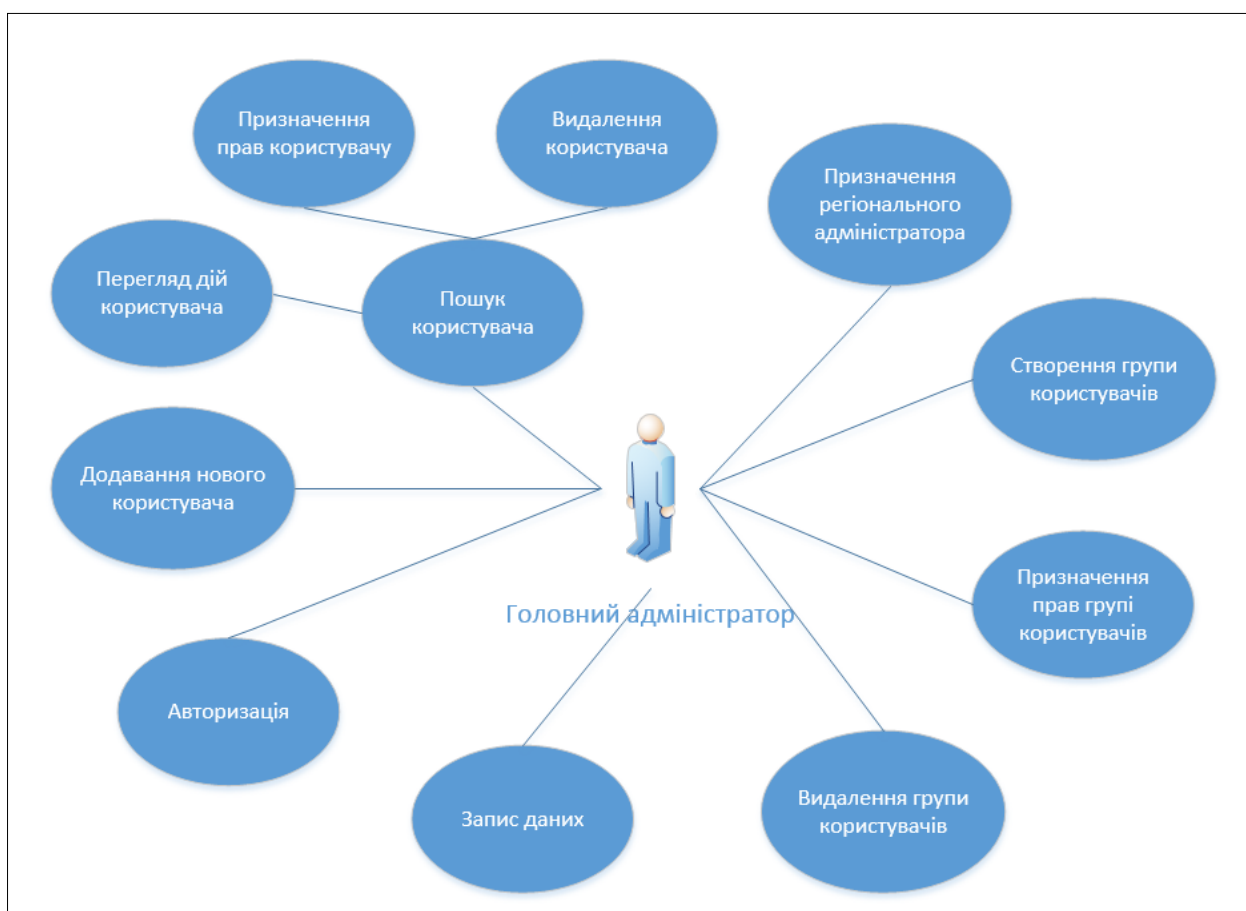


Рис. 2.1 – Загальна схема прецедентів для ролі головного адміністратора

2.2. Прецеденти для ролі головного адміністратора

Нижче описані процедури для ролі користувача з вказаними передумовами, результатом, виключними ситуаціями та детальним описом послідовності дій.

Перецент №1. Призначення/видалення регіонального адміністратора.

Назва: призначення / видалення регіонального адміністратора.

Учасники: Головний адміністратор, система.

Передумови: Головний адміністратор отримав заявку від керівництва на призначення / видалення регіонального адміністратора.

Результат: призначення / видалення регіонального адміністратора.

Основний сценарій:

1. Головний адміністратор подає запит на призначення / видалення регіонального адміністратора.
2. Система реєструє видаляє регіонального адміністратора.
3. Система видає звіт про успішність операції.

Виняткові ситуації:

1. Відсутність даного регіонального адміністратора в системі, при запиті його видалення.

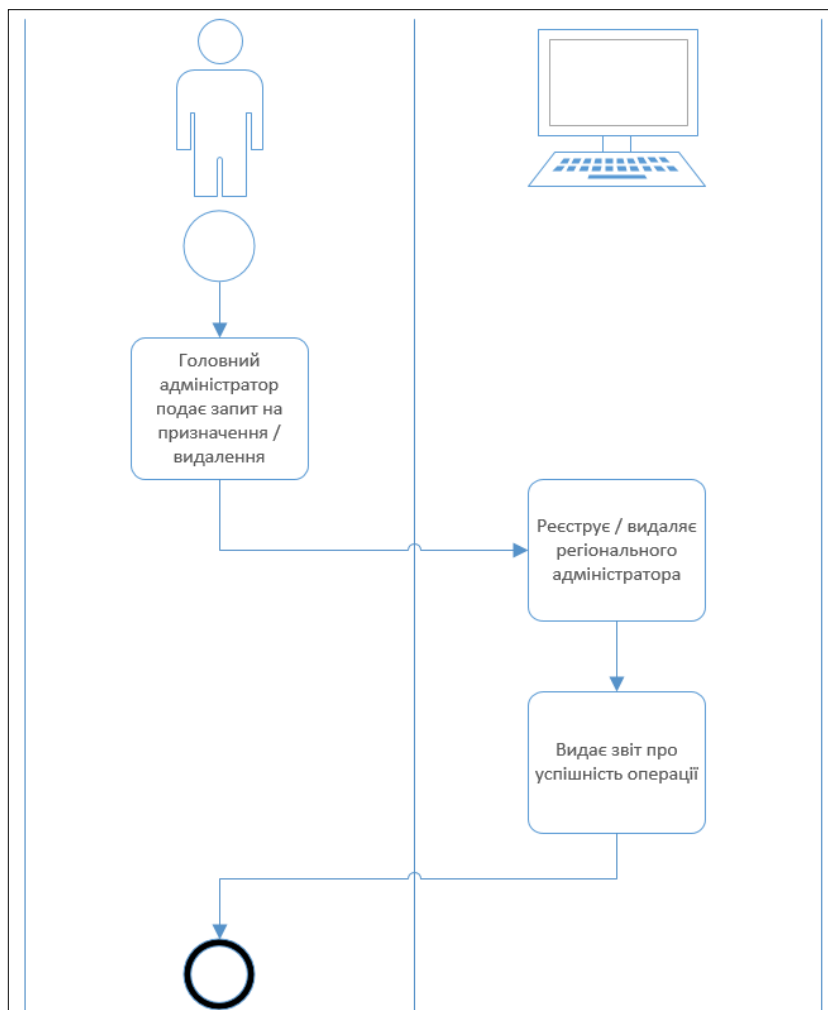


Рис. 2.2 – Схема призначення / видалення регіонального адміністратора

Прецедент №2. Видалення користувача.

Назва: Видалення користувача.

Учасники: головний адміністратор, система.

Попередні умови: заявка від керівництва правоохоронних органів на видалення користувача.

Результат: видалення користувача із системи.

Основний сценарій:

1. Головний адміністратор отримує заявку від керівництва правоохоронних органів на видалення користувача.
2. Головний адміністратор, користувач подає запит на видалення користувача.
3. Система отримує запит, перевіряє права на виконання даної операції.
4. Система видаляє користувача.
5. Система повертає звіт про успішність операції.

Виняткові ситуації:

1. Такого користувача не існує.
2. Не пройдена перевірка прав.

Підп. і дата	
Взаєм. інв. №	
Інв. № дубл.	
Підп. і дата	
Інв. № підп	

Зм	Арк.	№ докум.	Підпис	Дат

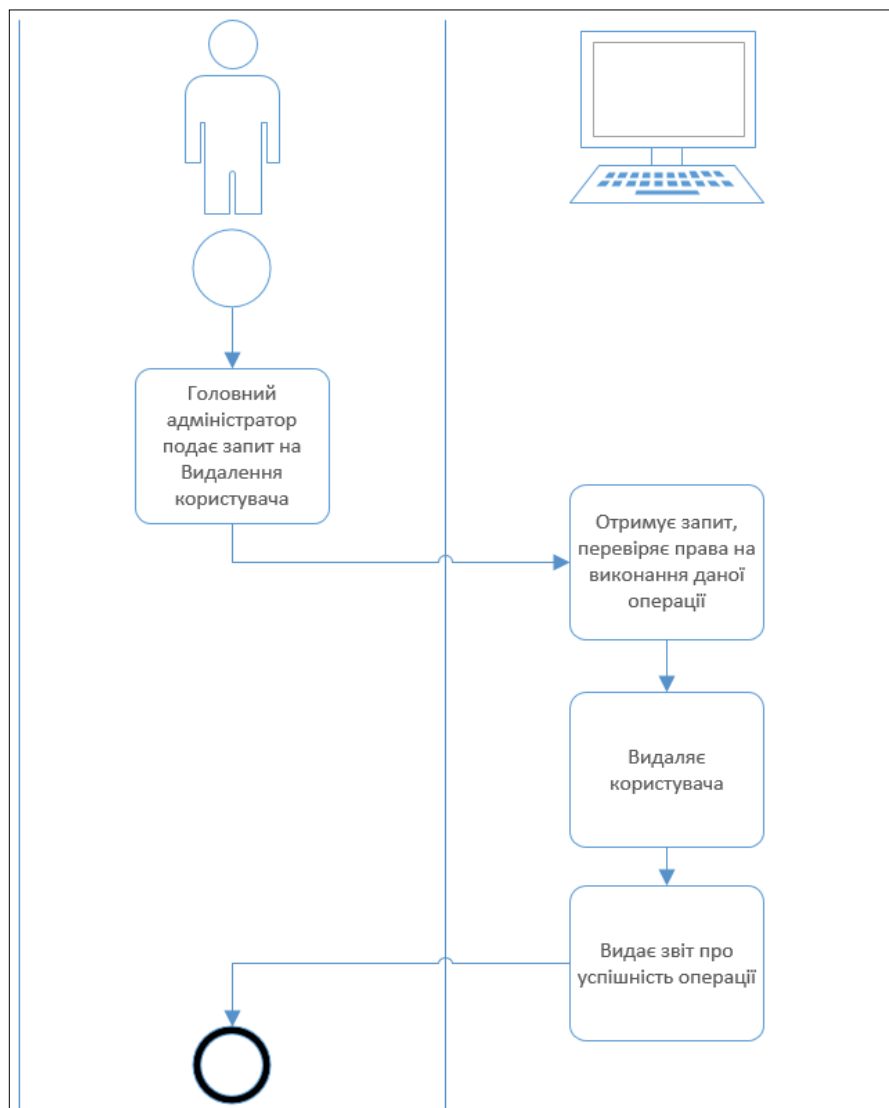


Рис. 2.3 – Схема видалення користувача

Прецедент №3. Створення групи.

Назва: Створення групи.

Учасники: головний адміністратор, система.

Попередні умови: заявка від керівництва правоохоронних органів на створення групи.

Результат: створення групи.

Основний сценарій:

1. Головний адміністратор отримує заявку від керівництва правоохоронних органів на створення групи.
2. Головний адміністратор подає запит на реєстрацію групи користувачів.

3. Система отримує запит, перевіряє права на виконання даної операції.
4. Система надає форму для заповнення інформації про нову групу користувачів.
5. Головний адміністратор заповнює інформацію про нову групу користувачів.
6. Головний адміністратор відправляє форму системі.
7. Система реєструє нову групу користувачів системи.
8. Система повертає звіт про успішність операції.

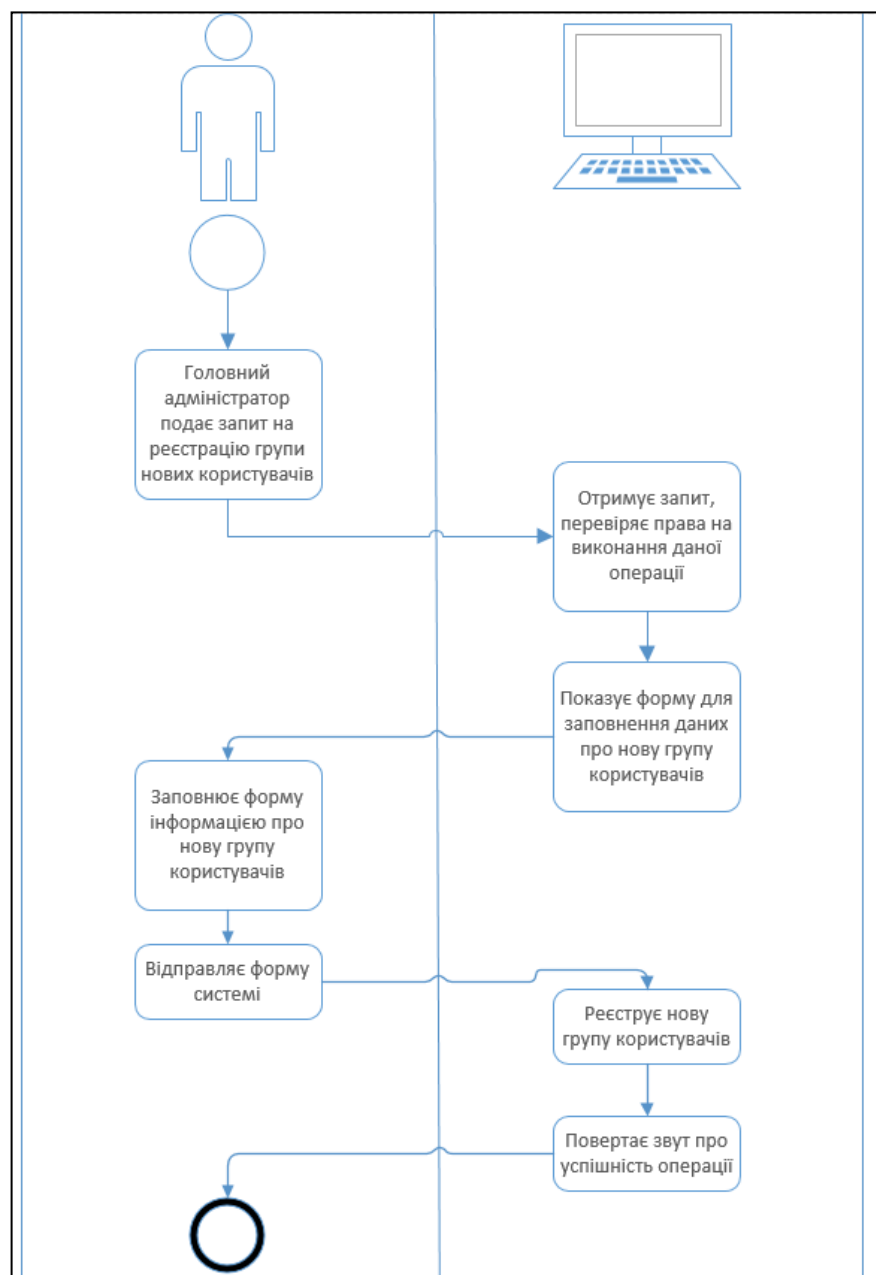


Рис. 2.4 – Схема створення групи

Прецедент №4. Видалення групи.

Назва: Видалення групи.

Учасники: головний адміністратор / регіональний адміністратор, система.

Попередні умови: заявка від керівництва правоохоронних органів на Видалення групи.

Результат: Видалення групи.

Основний сценарій:

1. Головний / регіональний адміністратор отримує заявку від керівництва правоохоронних органів на видалення групи.
2. Головний / регіональний адміністратор подає запит на видалення групи користувачів.
3. Система отримує запит, перевіряє права на виконання даної операції.
4. Система видаляє групу користувачів.
5. Система повертає звіт про успішність операції.

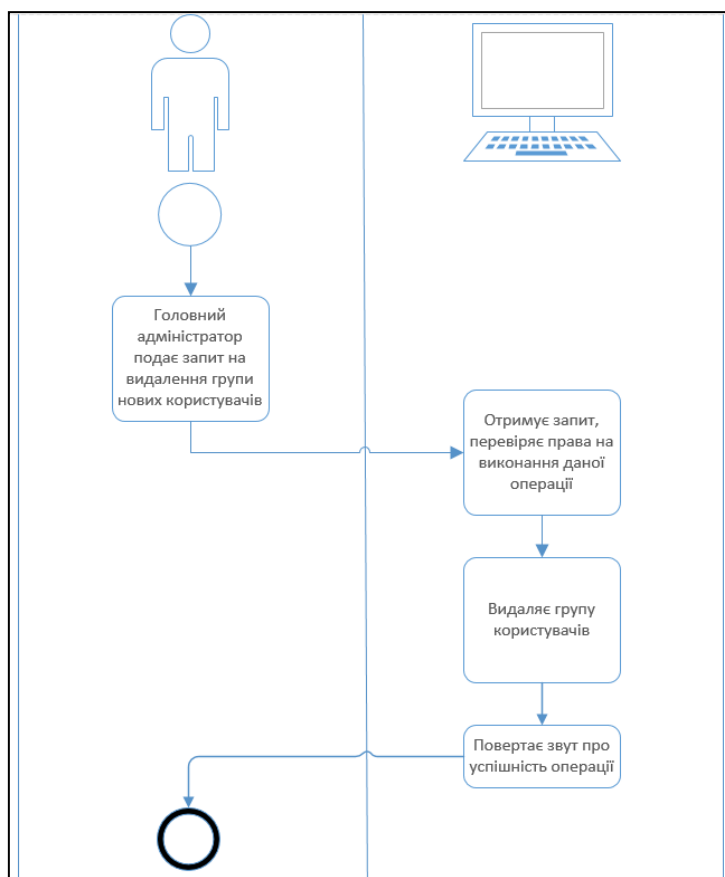


Рис. 2.5 – Схема видалення групи

Прецедент №5. Додавання користувача в групу.

Назва: Додавання користувача в групу.

Учасники: головний адміністратор / регіональний адміністратор, система.

Попередні умови: заявка від керівництва про додавання користувача в групу.

Результат: додавання користувача в групу.

Основний сценарій:

1. Головний / регіональний адміністратор отримує заявку від керівництва про додавання користувача в групу.
2. Головний / регіональний адміністратор подає запит на додавання користувача в групу.
3. Система отримує запит, перевіряє права на виконання даної операції.
4. Система додає користувача в групу.
5. Система повертає звіт про успішність операції.

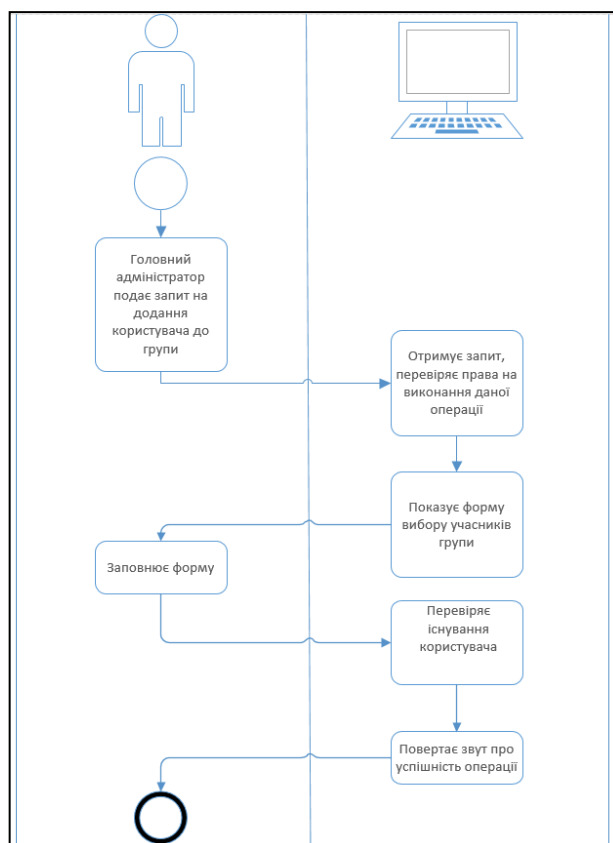


Рис. 2.6 – Схема додавання користувача в групу

Прецедент №6. Видалення користувача з групи.

Назва: Видалення користувача з групи.

Учасники: головний адміністратор / регіональний адміністратор, система.

Попередні умови: заявка від керівництва про видалення користувача з групи.

Результат: видалення користувача з групи.

Основний сценарій:

1. Головний / регіональний адміністратор подає запит на видалення користувача з групи.
2. Система отримує запит, перевіряє права на виконання даної операції.
3. Система видаляє користувача з групи.
4. Система повертає звіт про успішність операції.

Виняткові ситуації:

1. Даний користувач не зареєстрована в системі.

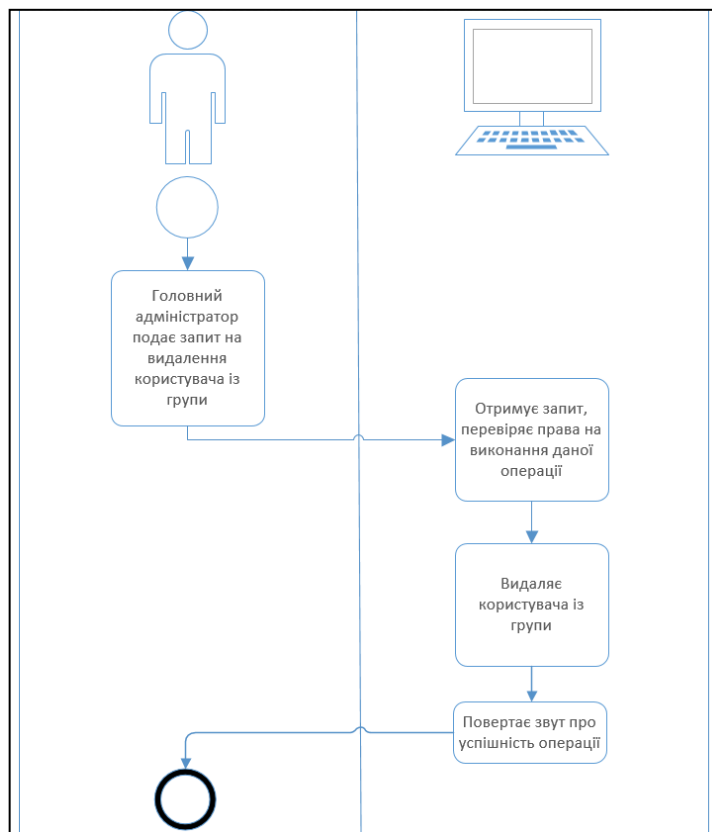


Рис. 2.7 – Схема видалення користувача з групи

2.3. Діаграма бізнес-сутностей

Дана діаграма створюється на етапі бізнес моделювання. Вона відображає основні сутності та взаємозв'язки між ними. В даному випадку основними сутностями є Employee, Action, Artifact та Dossier, які взаємодіють між собою та включають у себе допоміжні бізнес-сутності. Діаграма бізнес-сутностей проекту зображена на рис. 2.8.

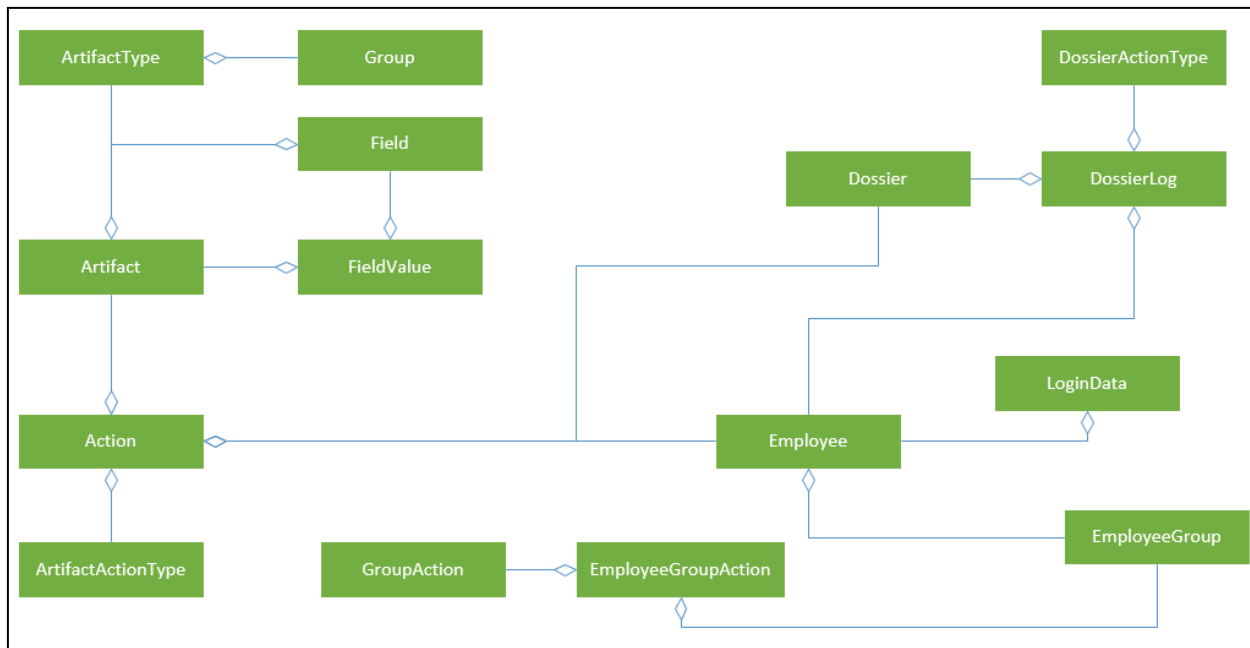


Рис. 2.8 – Діаграма бізнес-сутностей

2.4. Реляційна модель бази даних

Реляційна модель бази даних (рис 2.3) зображує структуру таблиць бази даних, взаємозв'язки між ними та поля кожної з таблиць. Наведена діаграма має багато схожого з діаграмою бізнес-сутностей. Кожній основній бізнес-сутності відповідає таблиця баз даних.

Підп. і дата	
Взаєм. інв. №	
Інв. № дубл.	
Підп. і дата	
Інв. № підп	

Зм	Арк.	№ докум.	Підпис	Дат

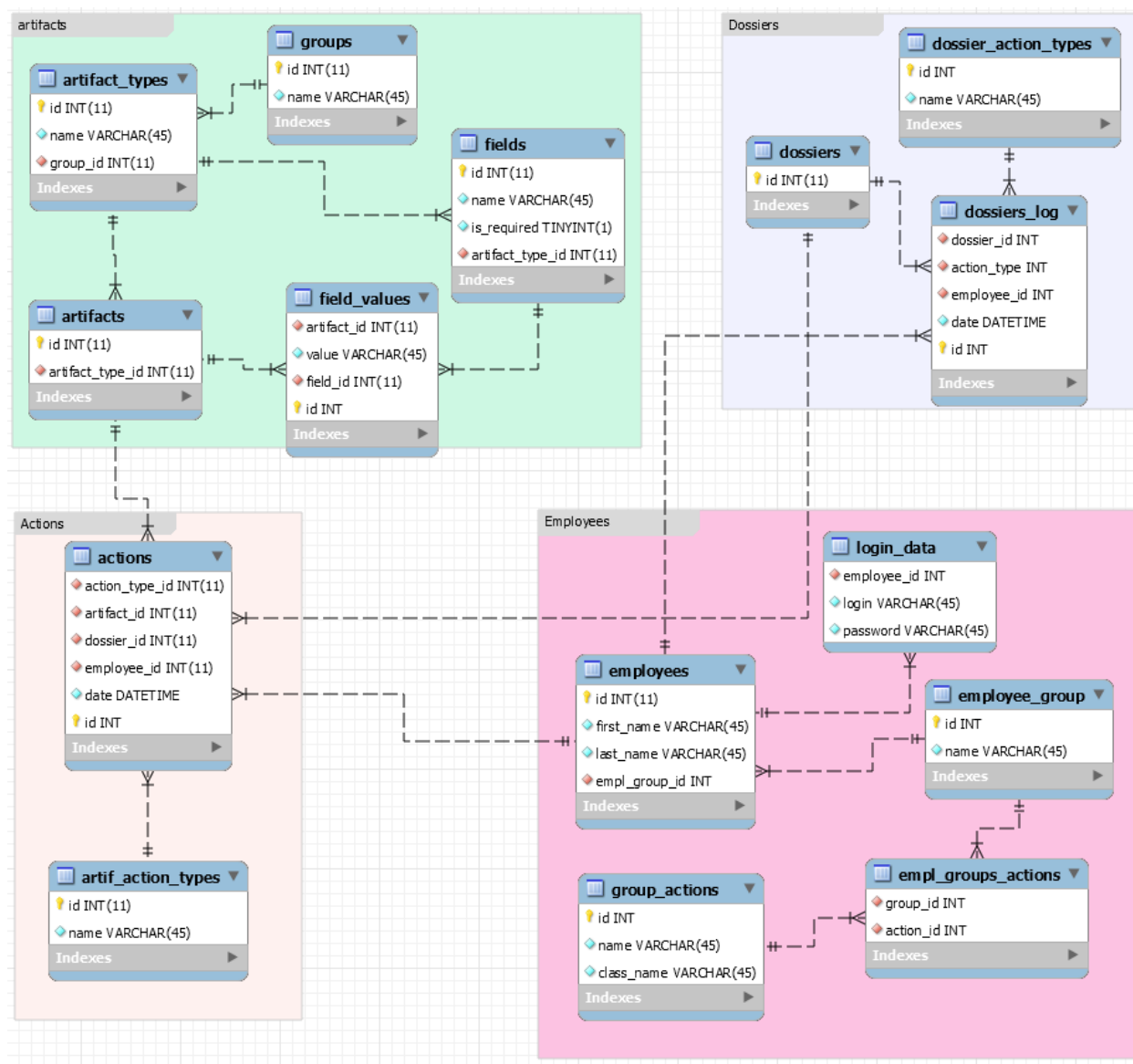


Рис 2.9 – Реляційна модель

2.5. Специфікація таблиць бази даних

Специфікація таблиць бази даних включає в себе інформацію про назви колонок таблиці, їхній тип, інформацію про те, чи є ця колонка первинним ключем, чи поле може бути пустим, чи значення поля автоматично збільшується та коментар щодо призначення колонки. Таблиці зі специфікаціями наведені нижче.

Таблиця 2.1

Таблиця artifacts

Назва	Тип даних	Ключ	Не пуста	Авто-інкремент	Опис
Id	Int	Так	Так	Так	Ідентифікатор

					артефакту
Artifact_type_id	int	Ні	Так	Ні	Ідентифікатор типу артефакту

Таблиця 2.2

Таблиця groups

Назва	Тип даних	Ключ	Не пуста	Авто-інкремент	Опис
Id	INT	Так	Так	Так	Ідентифікатор групи
Name	VARCHAR	Ні	Так	Ні	Назва групи

Таблиця 2.3

Таблиця artifact_types

Назва	Тип даних	Ключ	Не пуста	Авто-інкремент	Опис
id	INT	Так	Так	Так	Ідентифікатор типу артефакта
name	VARCHAR	Ні	Так	Ні	Назва артефакта
group_id	INT	Ні	Так	Ні	Ідентифікатор групи артефакта

Таблиця 2.4

Таблиця fields

Назва	Тип даних	Ключ	Не пуста	Авто-інкремент	Опис
id	INT	Так	Так	Так	Ідентифікатор поля
name	VARCHAR	Ні	Так	Ні	Назва поля
is_required	TINYINT	Ні	Так	Ні	Чи обов'язкове поле
artifact_type_id	INT	Ні	Так	Ні	Ідентифікатор

					типу артефакта
--	--	--	--	--	-------------------

Таблиця 2.5

Таблиця field_values

Назва	Тип даних	Ключ	Не пуста	Авто-інкремент	Опис
id	INT	Так	Так	Так	Ідентифікатор значення поля
artifact_id	INT	Ні	Так	Ні	Ідентифікатор артефакту
value	VARCHAR	Ні	Так	Ні	Значення поля
field_id	INT	Ні	Так	Ні	Ідентифікатор поля

Таблиця 2.6

Таблиця actions

Назва	Тип даних	Ключ	Не пуста	Авто-інкремент	Опис
id	INT	Так	Так	Так	Ідентифікатор дії
action_type_id	INT	Ні	Так	Ні	Ідентифікатор типу дії
artifact_id	INT	Ні	Так	Ні	Ідентифікатор артефакту
dossier_id	INT	Ні	Так	Ні	Ідентифікатор дос'є
employee_id	INT	Ні	Так	Ні	Ідентифікатор співробітника
date	DATETIME	Ні	Так	Ні	Дата створення запису

Підп. і дата	
Взаєм. інв. №	
Інв. № дубл.	
Підп. і дата	
Інв. № підп	

Зм	Арк.	№ докум.	Підпис	Дат

Таблиця 2.7

Таблиця artif_action_types

Назва	Тип даних	Ключ	Не пуста	Авто-інкремент	Опис
id	INT	Так	Так	Так	Ідентифікатор типу дії
name	VARCHAR	Ні	Так	Ні	Назва дії

Таблиця 2.8

Таблиця dossiers

Назва	Тип даних	Ключ	Не пуста	Авто-інкремент	Опис
id	INT	Так	Так	Так	Ідентифікатор досьє

Таблиця 2.9

Таблиця dossier_action_types

Назва	Тип даних	Ключ	Не пуста	Авто-інкремент	Опис
id	INT	Так	Так	Так	Ідентифікатор типу дії з досьє
name	VARCHAR	Ні	Так	Ні	Назва типу дії з досьє

Таблиця 2.10

Таблиця dossiers_log

Назва	Тип даних	Ключ	Не пуста	Авто-інкремент	Опис
id	INT	Так	Так	Так	Ідентифікатор запису логу
dossier_id	INT	Ні	Так	Ні	Ідентифікатор досьє
action_type	INT	Ні	Так	Ні	Ідентифікатор типу дії
employee_id	INT	Ні	Так	Ні	Ідентифікатор

					співробітника
date	DATETIME	Ні	Так	Ні	Дата створення запису логу

Таблиця 2.11

Таблиця employees

Назва	Тип даних	Ключ	Не пуста	Авто-інкремент	Опис
id	INT	Так	Так	Так	Ідентифікатор співробітника
first_name	VARCHAR	Ні	Так	Ні	Ім'я співробітника
last_name	VARCHAR	Ні	Так	Ні	Прізвище співробітника
empl_group_id	INT	Ні	Так	Ні	Ідентифікатор групи співробітника

Таблиця 2.12

Таблиця login_data

Назва	Тип даних	Ключ	Не пуста	Авто-інкремент	Опис
employee_id	INT	Ні	Так	Ні	Ідентифікатор співробітника
login	VARCHAR	Ні	Так	Ні	Логін співробітника
password	VARCHAR	Ні	Так	Ні	Пароль співробітника

Таблиця 2.13

Таблиця employee_group

Назва	Тип даних	Ключ	Не пуста	Авто-інкремент	Опис
id	INT	Так	Так	Так	Ідентифікатор групи співробітника

name	VARCHAR	Ні	Так	Ні	Назва групи співробітника
------	---------	----	-----	----	---------------------------

Таблиця 2.14

Таблиця group_actions

Назва	Тип даних	Ключ	Не пуста	Авто-інкремент	Опис
id	INT	Так	Так	Так	Ідентифікатор дії групи співробітників
name	VARCHAR	Ні	Так	Ні	Назва дії групи співробітників
class_name	VARCHAR	Ні	Так	Ні	Url-адреса для дії групи співробітників

Таблиця 2.15

Таблиця empl_groups_actions

Назва	Тип даних	Ключ	Не пуста	Авто-інкремент	Опис
group_id	INT	Так	Так	Ні	Ідентифікатор групи співробітників
action_id	INT	Так	Так	Ні	Ідентифікатор дії групи співробітників

Підп. і дата	
Взаєм. інв. №	
Інв. № дубл.	
Підп. і дата	
Інв. № підп	

Зм	Арк.	№ докум.	Підпис	Дат
----	------	----------	--------	-----

РОЗДІЛ 3

РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

3.1. Реляційно-об'єктне відображення

Для реляційно-об'єктного відображення в програмі використовується бібліотека Hibernate. Вона надає можливість легко встановити зв'язок з будь-якою базою даних та створити відображення між об'єктно-орієнтованою моделлю та традиційною реляційною моделлю баз даних. На рис. 3.1 зображено діаграму Entity класів. Детальна специфікація (JavaDoc) наведена нижче.

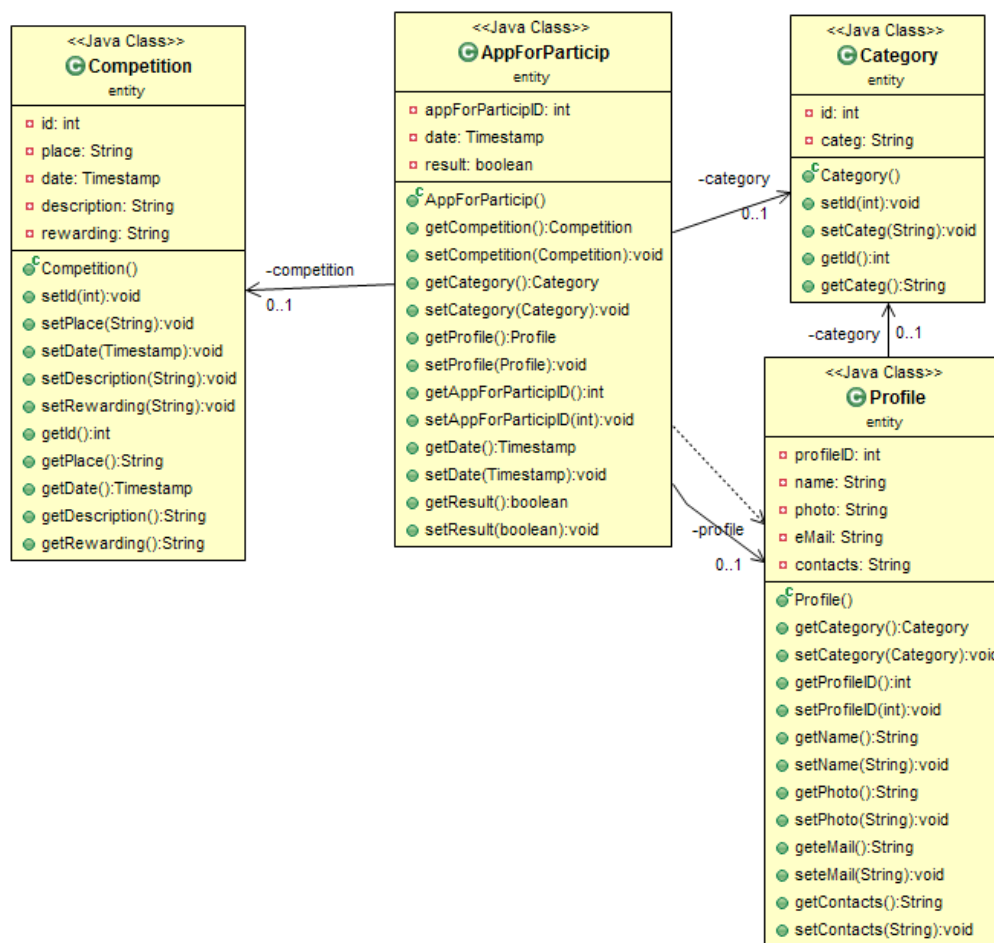


Рис 3.1 – Діаграма entity класів

3.1.1. Клас «AppForParticip»

```

@Entity
public class AppForParticip
extends java.lang.Object

Клас представляє сутність Заявка на участь

Since:
2015-04-14
Version:
1.0
Author:
Руслан Попенко
    
```

Field Summary

Fields

Modifier and Type	Field and Description
private int	appForParticipID Ідентифікатор заявки на участь
private Category	category Ідентифікатор категорії користувача
private Competition	competition Ідентифікатор змагання
private java.sql.Timestamp	date Дата створення заявки
private Profile	profile Ідентифікатор профілю
private boolean	result Результат реєстрації заявки

Constructor Summary

Constructors

Constructor and Description
AppForParticip()

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
int	getAppForParticipID()
Category	getCategory()
Competition	getCompetition()
java.sql.Timestamp	getDate()
Profile	getProfile()
boolean	getResult()
void	setAppForParticipID(int appForParticipID)
void	setCategory(Category category)
void	setCompetition(Competition competition)
void	setDate(java.sql.Timestamp date)
void	setProfile(Profile profile)
void	setResult(boolean result)

3.1.2. Клас «Category»

```
@Entity
public class Category
extends java.lang.Object
```

Клас представляє сутність Категорія

Since:

2015-04-14

Version:

1.0

Author:

Руслан Попенко

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Інв. № підп

Field Summary

Fields

Modifier and Type	Field and Description
private java.lang.String	categ Категорія користувача
private int	id Ідентифікатор категорії

Constructor Summary

Constructors

Constructor and Description
Category()

Method Summary

All MethodsInstance MethodsConcrete Methods

Modifier and Type	Method and Description
java.lang.String	getCateg()
int	getId()
void	setCateg(java.lang.String categ)
void	setId(int id)

@Entity
public class Competition
extends java.lang.Object

Клас представляє сутність Змагання

Since:
2015-04-14

Version:
1.0

Author:
Руслан Попенко

Field Summary

Fields

Modifier and Type	Field and Description
private java.sql.Timestamp	date Дата змагання
private java.lang.String	description Опис змагання
private int	id Ідентифікатор змагання
private java.lang.String	place Місце змагання
private java.lang.String	rewarding Нагородження учасників змагання

Підп. і дата	
Взаєм. інв. №	
Інв. № дубл.	
Підп. і дата	
Інв. № підп	

Зм	Арк.	№ докум.	Підпис	Дат

Field Summary

Fields

Modifier and Type	Field and Description
private java.lang.String	categ Категорія користувача
private int	id Ідентифікатор категорії

Constructor Summary

Constructors

Constructor and Description
Category()

Method Summary

All Methods

Instance Methods

Concrete Methods

Modifier and Type	Method and Description
java.lang.String	getCateg()
int	getId()
void	setCateg(java.lang.String categ)
void	setId(int id)

3.1.1. Клас «Competition»

```
@Entity
public class Competition
extends java.lang.Object
```

Клас представляє сутність Змагання

Since:
2015-04-14

Version:
1.0

Author:
Руслан Попенко

Method Summary

All Methods

Instance Methods

Concrete Methods

Modifier and Type	Method and Description
java.sql.Timestamp	getDate()
java.lang.String	getDescription()
int	getId()
java.lang.String	getPlace()
java.lang.String	getRewarding()
void	setDate(java.sql.Timestamp date)
void	setDescription(java.lang.String description)
void	setId(int id)
void	setPlace(java.lang.String place)
void	setRewarding(java.lang.String rewarding)

3.1.2. Клас «Profile»

```
@Entity
public class Profile
extends java.lang.Object
```

Клас представляє сутність Профіль

Since:
2015-04-14

Version:
1.0

Author:
Руслан Попенко

Fields	
Modifier and Type	Field and Description
private Category	category Ідентифікатор категорії
private java.lang.String	contacts Контактний телефон
private java.lang.String	eMail Пошта
private java.lang.String	name Ім'я
private java.lang.String	photo Фото
private int	profileID Ідентифікатор профіля

Constructor Summary

Constructors

Constructor and Description
Profile()

Підп. і дата	
Взаєм. інв. №	
Інв. № дубл.	
Підп. і дата	
Інв. № підп	

Зм	Арк.	№ докум.	Підпис	Дат

Field Summary

Fields

Modifier and Type	Field and Description
private java.util.Set<GroupActions>	actions Список дій, які дозволені даній групі співробітників
private java.util.List<Employee>	employees Список співробітників, які відносяться до даної групи
private java.lang.Integer	id Ідентифікатор групи співробітника
private java.lang.String	name Назва групи співробітника

Method Summary

Methods

Modifier and Type	Method and Description
java.util.Set<GroupActions>	getActions()
java.util.List<Employee>	getEmployees()
java.lang.Integer	getId()
java.lang.String	getName()
void	setActions (java.util.Set<GroupActions> actions)
void	setEmployees (java.util.List<Employee> employees)
void	setId (java.lang.Integer id)
void	setName (java.lang.String name)

3.1.3. Клас «GroupActions»

```
@Entity
public class GroupActions
extends java.lang.Object
implements java.io.Serializable
```

Клас представляє сутність Дії групи співробітників Created by Andrey on 03.03.2015.

See Also:

[Serialized Form](#)

Field Summary

Fields

Modifier and Type	Field and Description
private java.lang.String	className Url-адреса для дії групи співробітників
private java.util.Set<EmployeeGroup>	groups Список груп співробітників, яким дозволена дана дія
private java.lang.Integer	id Ідентифікатор дії групи співробітників
private java.lang.String	name Назва дії групи співробітників

Constructor Summary

Constructors

Constructor and Description

`GroupActions()`

Method Summary

Methods

Modifier and Type	Method and Description
<code>java.lang.String</code>	<code>getClassName()</code>
<code>java.util.Set<EmployeeGroup></code>	<code>getGroups()</code>
<code>java.lang.Integer</code>	<code>getId()</code>
<code>java.lang.String</code>	<code>getName()</code>
<code>void</code>	<code>setClassName(java.lang.String className)</code>
<code>void</code>	<code>setGroups(java.util.Set<EmployeeGroup> groups)</code>
<code>void</code>	<code>setId(java.lang.Integer id)</code>
<code>void</code>	<code>setName(java.lang.String name)</code>

3.1.4. Клас «LoginData»

```
@Entity
public class LoginData
extends java.lang.Object
implements java.io.Serializable
```

Клас представляє сутність Дані авторизації співробітників Created by Andrey on 28.02.2015.

See Also:

[Serialized Form](#)

Field Summary

Fields

Modifier and Type	Field and Description
<code>private Employee</code>	<code>employee</code> Співробітник, якому належать дані дані авторизації
<code>private java.lang.Integer</code>	<code>employeeId</code> Ідентифікатор співробітника
<code>private java.lang.String</code>	<code>login</code> Логін співробітника
<code>private java.lang.String</code>	<code>password</code> Пароль співробітника

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Інв. № підп

Constructor Summary

Constructors

Constructor and Description

LoginData()

LoginData(java.lang.String login, java.lang.String pass, Employee empl)

Method Summary

Methods

Modifier and Type	Method and Description
Employee	getEmployee()
java.lang.Integer	getEmployeeId()
java.lang.String	getLogin()
java.lang.String	getPassword()
void	setEmployee(Employee employee)
void	setEmployeeId(java.lang.Integer employeeId)
void	setLogin(java.lang.String login)
void	setPassword(java.lang.String password)

3.1.5. Клас «Artifact»

```
@Entity  
public class Artifact  
extends java.lang.Object
```

Клас представляє сутність Артефакти Created by Andrey on 15.03.2015.

Field Summary

Fields

Modifier and Type	Field and Description
(package private) java.util.List<Action>	actions Список дій відносно цього артефакта
(package private) ArtifactType	artifactType Ідентифікатор типу артефакту
(package private) java.util.List<FieldValue>	fieldValues Список значень полів артефакта
(package private) java.lang.Integer	id Ідентифікатор артефакту

Constructor Summary

Constructors

Constructor and Description

Artifact()

Method Summary

Methods

Modifier and Type	Method and Description
java.util.List<Action>	<code>getActions()</code>
ArtifactType	<code>getArtifactType()</code>
java.util.List<FieldValue>	<code>getFieldValues()</code>
java.lang.Integer	<code>getId()</code>
void	<code>setActions(java.util.List<Action> actions)</code>
void	<code>setArtifactType(ArtifactType artifactType)</code>
void	<code>setFieldValues(java.util.List<FieldValue> fieldValues)</code>
void	<code>setId(java.lang.Integer id)</code>

3.1.6. Клас «ArtifactType»

```
@Entity  
public class ArtifactType  
extends java.lang.Object
```

Клас представляє сутність Типи артефактів Created by Andrey on 15.03.2015.

Field Summary

Fields

Modifier and Type	Field and Description
(package private) java.util.List<Artifact>	<code>artifacts</code> Список артефактів, які відносяться до даного типу
(package private) java.util.List<Field>	<code>fields</code> Список полів, які відносяться до даного типу артефактів
(package private) Group	<code>group</code> Ідентифікатор групи артефакта
(package private) java.lang.Integer	<code>id</code> Ідентифікатор типу артефакта
(package private) java.lang.String	<code>name</code> Назва артефакта

Constructor Summary

Constructors

Constructor and Description
<code>ArtifactType()</code>

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Інв. № підп

Зм	Арк.	№ докум.	Підпис	Дат

Method Summary

Methods

Modifier and Type	Method and Description
java.util.List<Artifact>	getArtifacts()
java.util.List<Field>	getFields()
Group	getGroup()
java.lang.Integer	getId()
java.lang.String	getName()
void	setArtifacts (java.util.List<Artifact> artifacts)
void	setFields (java.util.List<Field> fields)
void	setGroup (Group group)
void	setId (java.lang.Integer id)
void	setName (java.lang.String name)

3.1.7. Клас «Group»

```
@Entity
public class Group
extends java.lang.Object
```

Клас представляє сутність Групи артефактів Created by Andrey on 15.03.2015.

Field Summary

Fields

Modifier and Type	Field and Description
(package private) java.util.List<ArtifactType>	artifType Список типів артефактів, які відносяться до даної групи
(package private) java.lang.Integer	id Ідентифікатор групи
(package private) java.lang.String	name Назва групи

Constructor Summary

Constructors

Constructor and Description
Group()

Method Summary

Methods

Modifier and Type	Method and Description
java.util.List<ArtifactType>	<code>getArtifType()</code>
java.lang.Integer	<code>getId()</code>
java.lang.String	<code>getName()</code>
void	<code>setArtifType(java.util.List<ArtifactType> artifType)</code>
void	<code>setId(java.lang.Integer id)</code>
void	<code>setName(java.lang.String name)</code>

3.1.8. Клас «Field»

```
@Entity  
public class Field  
extends java.lang.Object
```

Клас представляє сутність Поля Created by Andrey on 15.03.2015.

Field Summary

Fields

Modifier and Type	Field and Description
(package private) ArtifactType	artifType Ідентифікатор типу артефакта
(package private) java.util.List<FieldValue>	fieldValues Список значень поля
(package private) java.lang.Integer	id Ідентифікатор поля
(package private) java.lang.String	name Назва поля
(package private) boolean	required Чи обов'язкове поле

Constructor Summary

Constructors

Constructor and Description
<code>Field()</code>

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Інв. № підп

Зм	Арк.	№ докум.	Підпис	Дат

Method Summary

Methods

Modifier and Type	Method and Description
ArtifactType	getArtifType()
java.util.List<FieldValue>	getFieldValues()
java.lang.Integer	getId()
java.lang.String	getName()
boolean	isRequired()
void	setArtifType(ArtifactType artifType)
void	setFieldValues(java.util.List<FieldValue> fieldValues)
void	setId(java.lang.Integer id)
void	setName(java.lang.String name)
void	setRequired(boolean required)

3.1.9. Клас «FieldValue»

```
@Entity  
public class FieldValue  
extends java.lang.Object
```

Клас представляє сутність Значення поля Created by Andrey on 15.03.2015.

Field Summary

Fields

Modifier and Type	Field and Description
(package private) Artifact	artifact Ідентифікатор артефакту
(package private) Field	field Ідентифікатор поля
(package private) java.lang.Integer	id Ідентифікатор значення поля
(package private) java.lang.String	value Значення поля

Constructor Summary

Constructors

Constructor and Description
FieldValue()

Method Summary

Methods	
Modifier and Type	Method and Description
Artifact	getArtifact()
Field	getField()
java.lang.Integer	getId()
java.lang.String	getValue()
void	setArtifact(Artifact artifact)
void	setField(Field field)
void	setId(java.lang.Integer id)
void	setValue(java.lang.String value)

Інв. № підп	Підп. і дата	Інв. № дубл.	Взаєм. інв. №	Підп. і дата

Зм	Арк.	№ докум.	Підпис	Дат

3.2. Специфікація Service класів

Класи, що тут представлені, містять service методи програми. В них закладена вся бізнес логіка роботи програми. Діаграму цих класів можна побачити на рис. 3.2. Детальна специфікація наведена в додатку В.

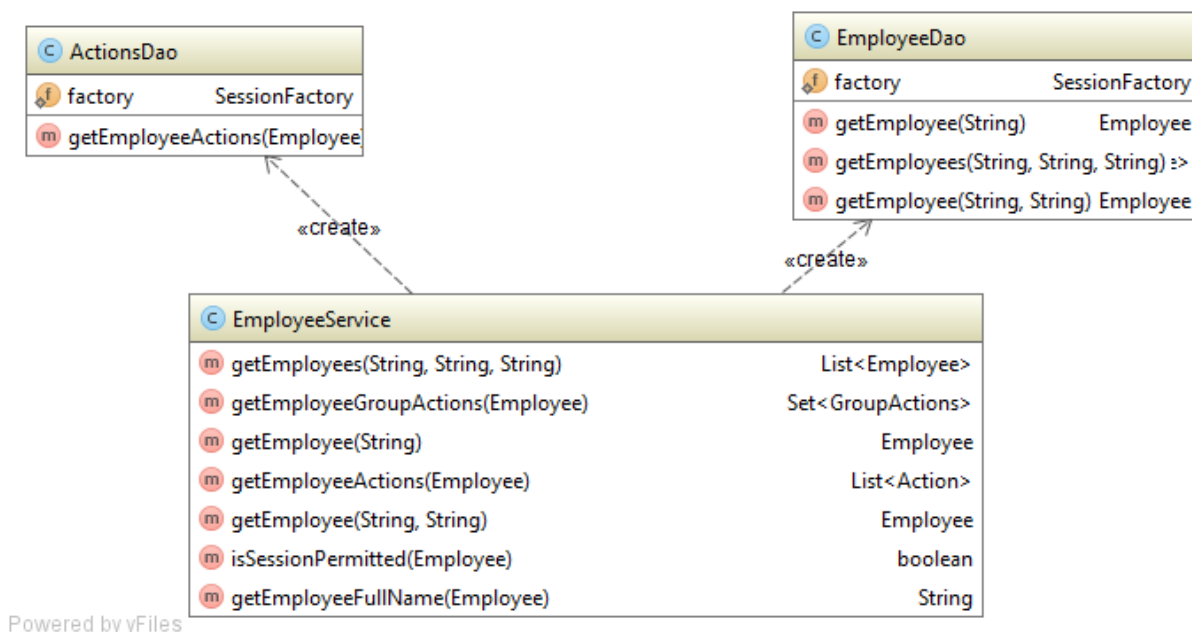


Рис. 3.2 – Діаграма service класів

3.3. Специфікація DAO-класів

Класи, що тут представлені, містять методи для роботи з базою даних. Ці методи використовують бібліотеку Hibernate. Діаграму цих класів можна побачити на рис. 3.3. Детальна специфікація наведена в додатку В.

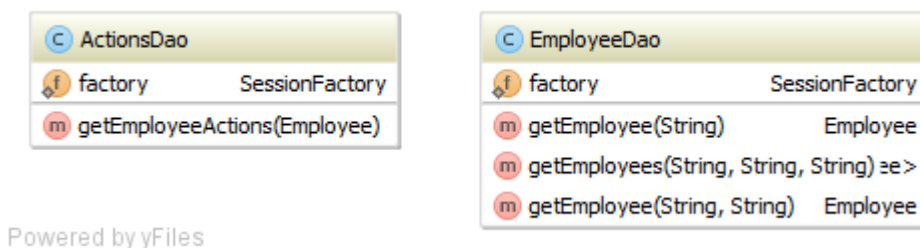


Рис 3.3 – Діаграма DAO класів

3.4. Класи контролерів та їх специфікація

Дані класи призначені для створення зв'язку між сервером та клієнтом. Діаграму цих класів можна побачити на рис. 3.4. Детальна специфікація наведена в додатку В.

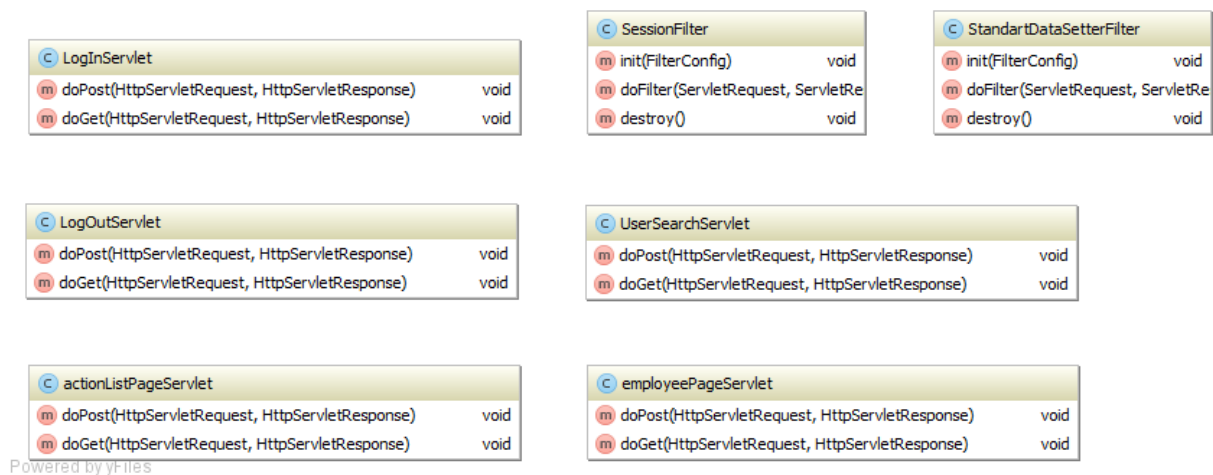


Рис. 3.4 – Діаграма класів контролерів

Інв. № підп	Підп. і дата	Інв. № дубл.	Взаєм. інв. №	Підп. і дата	6.050102 «Комп'ютерна інженерія»					Арк.
					Зм	Арк.	№ докум.	Підпис	Дат	377

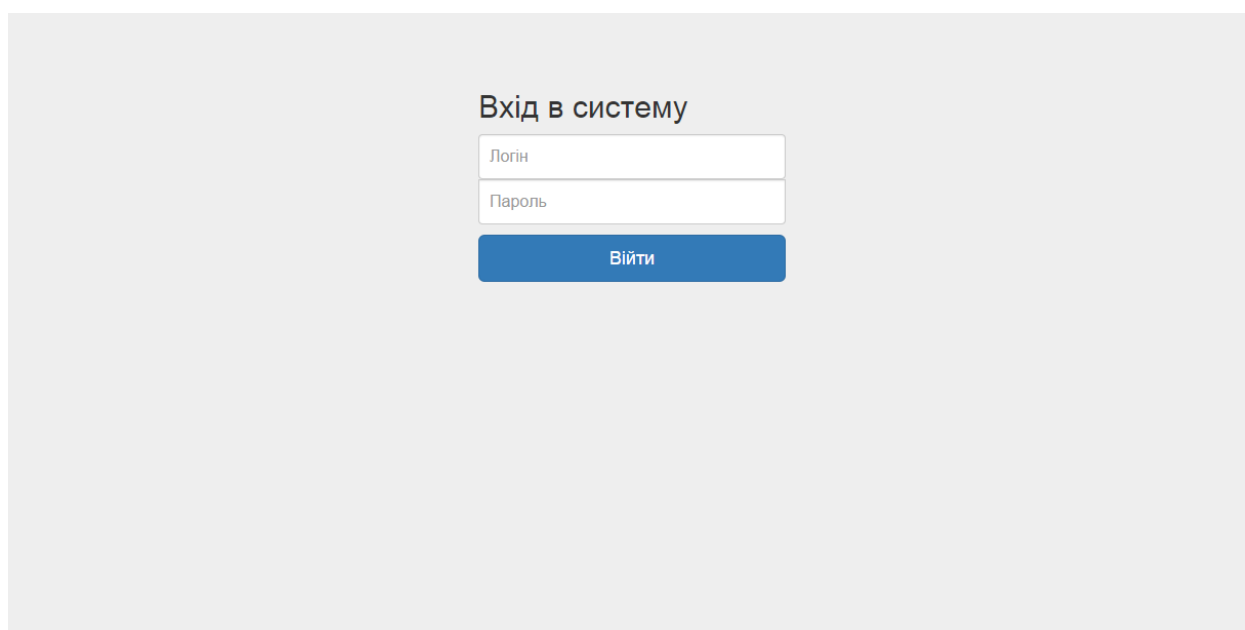
РОЗДІЛ 4

ІЛЮСТРАЦІЯ РОБОТИ ПРОГРАМИ

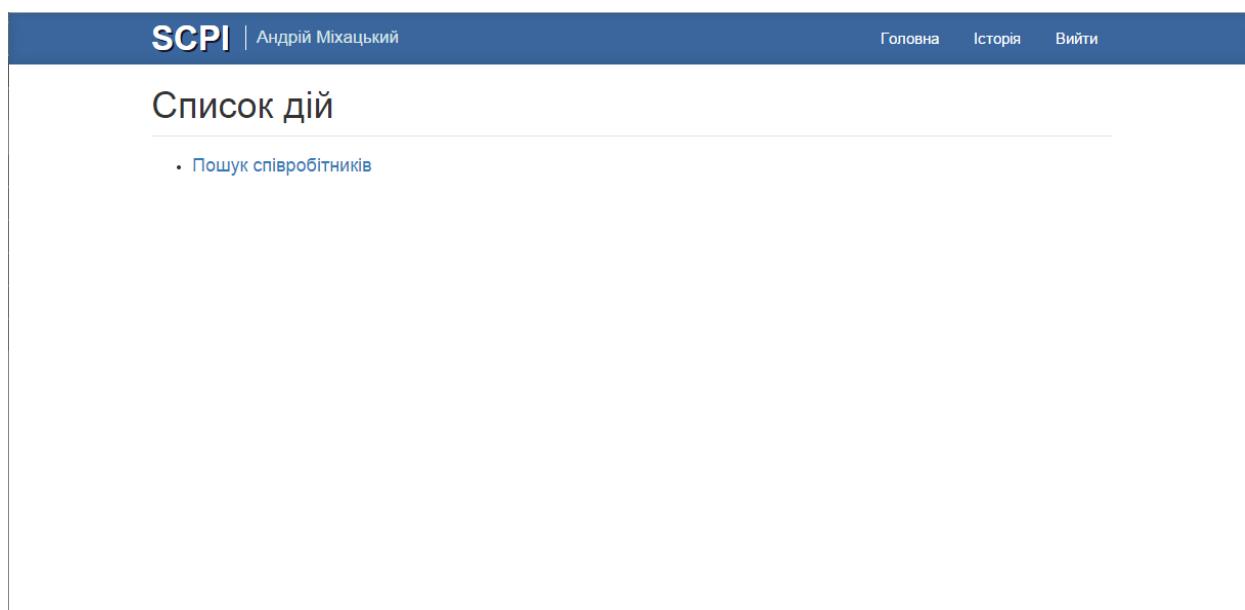
Для ілюстрації роботи програми в цьому розділі наведено графічні сценарії роботи проекту.

4.1. Пошук співробітника та перегляд його дій в системі.

4.1.1. При заході на сайт без попередньої авторизації, система видає користувачеві форму для введення логіну та паролю. Користувач заповнює поля та натискає кнопку Війти.



4.1.2. Після авторизації система показує користувачу сторінку зі списком дозволених користувачеві дій. Користувач обирає необхідну й натискає на неї.



Підп. і дата	
Взаєм. інв. №	
Інв. № дубл.	
Підп. і дата	
Інв. № підп	

Зм	Арк.	№ докум.	Підпис	Дат

4.1.3. Кожна дія має свою власну сторінку. В даному випадку, після вибору дії «Пошук співробітників» користувач потрапляє на сторінку з формою. В цій формі користувач вводить дані, по яким бажає провести пошук. Далі користувач повинен натиснути кнопку Шукати.

SCPI | Андрій Міхацький

ГоловнаІсторіяВийти

Пошук співробітників

Id:

Id

Ім'я:

Ім'я

Прізвище:

Прізвище

Шукати

Заповнювати всі поля не обов'язково. Пошук буде відбуватися по заповненим полям.

4.1.4. Далі користувач потрапляє на сторінку з результатами пошуку. Тут відображається список знайдених співробітників, відповідно до введених вище даних. Кожен елемент цього списку є посиланням на сторінку співробітника. Користувач обирає необхідне посилання й натискає на нього.

SCPI | Андрій Міхацький

ГоловнаІсторіяВийти

Пошук співробітників

#	Имя
1	Андрій Міхацький

Підп. і дата
Взаєм. інв. №
Інв. № дубл.
Підп. і дата
Інв. № підп

Зм	Арк.	№ докум.	Підпис	Дат

4.1.5. На сторінці співробітника відображається список дій, які зробив співробітник в системі.

SCPI | Андрій Міхацький

ГоловнаІсторіяВийти

Андрій Міхацький

Дія	Артефакт	Досьє	Дата
Запис артефакта	1	1	2015-03-26

Інв. № підп	Підп. і дата	Інв. № дубл.	Взаєм. інв. №	Підп. і дата

Зм	Арк.	№ докум.	Підпис	Дат

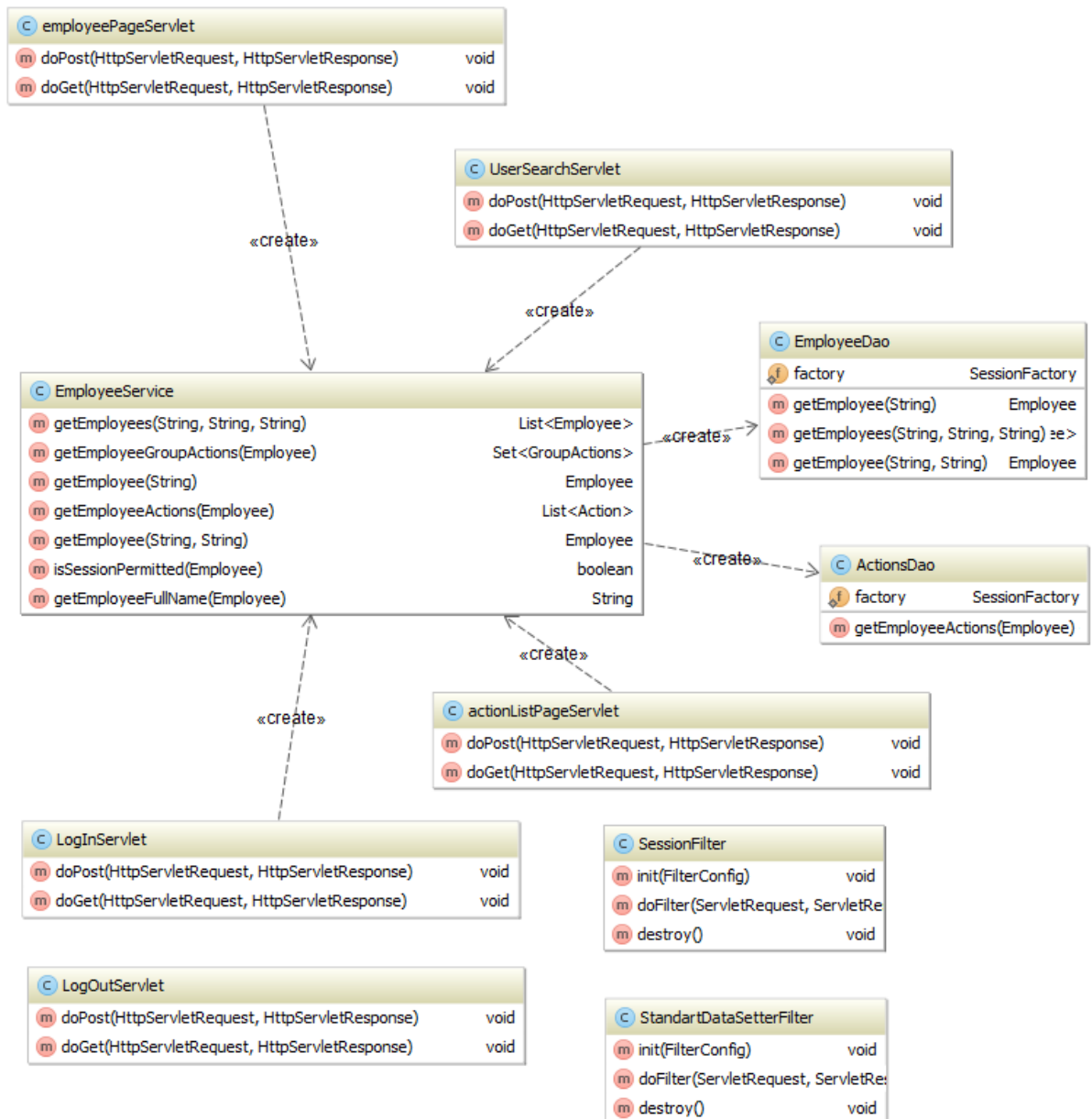
СПИСОК ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. Apache Tomcat. – Посилання: <http://tomcat.apache.org>
2. Hibernate. – Посилання: <http://hibernate.org>
3. Maven. – Посилання: <https://maven.apache.org>
4. Git. – Посилання: <http://git-scm.com>
5. GitHub. – Посилання: <https://github.com>

Інв. № підп	Підп. і дата	Інв. № дубл.	Взаєм. інв. №	Підп. і дата						Арк.
Зм	Арк.	№ докум.	Підпис	Дат	6.050102 «Комп'ютерна інженерія»					41

ДОДАТОК А

Діаграма класів



Powered by yFiles

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Інв. № підп.

Ошибка! Неизвестное имя свойства документа.»

Арк.

25

Зм Арк. № докум. Підпис Дат

ДОДАТОК Б

SQL код для створення таблиць бази даних:

```
CREATE SCHEMA IF NOT EXISTS `informationdb` DEFAULT CHARACTER SET utf8 ;

DROP SCHEMA IF EXISTS `test_informationdb` ;

CREATE SCHEMA IF NOT EXISTS `test_informationdb` DEFAULT CHARACTER SET utf8 ;

USE `informationdb` ;

DROP TABLE IF EXISTS `informationdb`.`artif_action_types` ;

CREATE TABLE IF NOT EXISTS `informationdb`.`artif_action_types` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB
AUTO_INCREMENT = 4
DEFAULT CHARACTER SET = utf8;

DROP TABLE IF EXISTS `informationdb`.`groups` ;

CREATE TABLE IF NOT EXISTS `informationdb`.`groups` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

DROP TABLE IF EXISTS `informationdb`.`artifact_types` ;

CREATE TABLE IF NOT EXISTS `informationdb`.`artifact_types` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(45) NOT NULL,
  `group_id` INT(11) NOT NULL,
  PRIMARY KEY (`id`),
  CONSTRAINT `group_id`
  FOREIGN KEY (`group_id`)
```

Інв. № підп.	Підп. і дата	Інв. № дубл.	Взаєм. інв. №	Підп. і дата	Зм	Арк.	№ докум.	Підпис	Дат	Ошибка! Неизвестное имя свойства документа.»	Арк.
											25

REFERENCES `informationdb`.`groups` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

CREATE INDEX `group_id_idx` ON `informationdb`.`artifact_types` (`group_id` ASC);

DROP TABLE IF EXISTS `informationdb`.`artifacts` ;

CREATE TABLE IF NOT EXISTS `informationdb`.`artifacts` (

`id` INT(11) NOT NULL AUTO_INCREMENT,

`artifact_type_id` INT(11) NOT NULL,

PRIMARY KEY (`id`),

CONSTRAINT `artifact_to_type`

FOREIGN KEY (`artifact_type_id`)

REFERENCES `informationdb`.`artifact_types` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

CREATE INDEX `artifact_to_type_idx` ON `informationdb`.`artifacts` (`artifact_type_id` ASC);

DROP TABLE IF EXISTS `informationdb`.`dossiers` ;

CREATE TABLE IF NOT EXISTS `informationdb`.`dossiers` (

`id` INT(11) NOT NULL AUTO_INCREMENT,

PRIMARY KEY (`id`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

DROP TABLE IF EXISTS `informationdb`.`employee_group` ;

CREATE TABLE IF NOT EXISTS `informationdb`.`employee_group` (

`id` INT NOT NULL AUTO_INCREMENT,

`name` VARCHAR(45) NOT NULL,

PRIMARY KEY (`id`))

Підп. і дата

Взаєм. інс. №

Інс. № дубл.

Підп. і дата

Інс. № підп

Ошибка! Неизвестное имя свойства документа.»

Арк.

25

Зм

Арк.

№ докум.

Підпис

Дат

ENGINE = InnoDB

DROP TABLE IF EXISTS `informationdb`.`employees` ;

CREATE TABLE IF NOT EXISTS `informationdb`.`employees` (

`id` INT(11) NOT NULL AUTO_INCREMENT,

`first_name` VARCHAR(45) NOT NULL,

`last_name` VARCHAR(45) NOT NULL,

`empl_group_id` INT NOT NULL,

PRIMARY KEY (`id`),

CONSTRAINT `to_empl_type`

FOREIGN KEY (`empl_group_id`)

REFERENCES `informationdb`.`employee_group` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

CREATE INDEX `to_empl_type_idx` ON `informationdb`.`employees` (`empl_group_id` ASC);

DROP TABLE IF EXISTS `informationdb`.`actions` ;

CREATE TABLE IF NOT EXISTS `informationdb`.`actions` (

`action_type_id` INT(11) NOT NULL,

`artifact_id` INT(11) NOT NULL,

`dossier_id` INT(11) NOT NULL,

`employee_id` INT(11) NOT NULL,

`date` DATETIME NOT NULL,

`id` INT NOT NULL AUTO_INCREMENT,

PRIMARY KEY (`id`),

CONSTRAINT `to_action_type_id`

FOREIGN KEY (`action_type_id`)

REFERENCES `informationdb`.`artif_action_types` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

Ошибка! Неизвестное имя свойства документа.»

Арк.

25

```

CONSTRAINT `to_artifact_id`

FOREIGN KEY (`artifact_id`)

REFERENCES `informationdb`.`artifacts` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `to_dossier_id`

FOREIGN KEY (`dossier_id`)

REFERENCES `informationdb`.`dossiers` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `to_employee_id`

FOREIGN KEY (`employee_id`)

REFERENCES `informationdb`.`employees` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

```

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

CREATE INDEX `action_type_id_idx` ON `informationdb`.`actions` (`action_type_id` ASC);

CREATE INDEX `artifact_d_idx` ON `informationdb`.`actions` (`artifact_id` ASC);

CREATE INDEX `dossier_id_idx` ON `informationdb`.`actions` (`dossier_id` ASC);

CREATE INDEX `employee_id_idx` ON `informationdb`.`actions` (`employee_id` ASC);

DROP TABLE IF EXISTS `informationdb`.`fields` ;

CREATE TABLE IF NOT EXISTS `informationdb`.`fields` (

`id` INT(11) NOT NULL AUTO_INCREMENT,

`name` VARCHAR(45) NOT NULL,

`is_required` TINYINT(1) NOT NULL,

`artifact_type_id` INT(11) NOT NULL,

PRIMARY KEY (`id`),

CONSTRAINT `artifact_type`

FOREIGN KEY (`artifact_type_id`)

Ошибка! Неизвестное имя свойства документа.»

Арк.

25

Підп. і дата

Взаєм. інс. №

Інс. № дубл.

Підп. і дата

Інс. № підп.

Зм

Арк.

№ докум.

Підпис

Дат

```

REFERENCES `informationdb`.`artifact_types` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

CREATE INDEX `artifact_type_idx` ON `informationdb`.`fields` (`artifact_type_id` ASC);

DROP TABLE IF EXISTS `informationdb`.`field_values` ;

CREATE TABLE IF NOT EXISTS `informationdb`.`field_values` (

`artifact_id` INT(11) NOT NULL,

`value` VARCHAR(45) NOT NULL,

`field_id` INT(11) NOT NULL,

`id` INT NOT NULL AUTO_INCREMENT,

PRIMARY KEY (`id`),

CONSTRAINT `artifact_id`

FOREIGN KEY (`artifact_id`)

REFERENCES `informationdb`.`artifacts` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `field_id`

FOREIGN KEY (`field_id`)

REFERENCES `informationdb`.`fields` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

CREATE INDEX `field_id_idx` ON `informationdb`.`field_values` (`field_id` ASC);

CREATE INDEX `artifact_id_idx` ON `informationdb`.`field_values` (`artifact_id` ASC);

DROP TABLE IF EXISTS `informationdb`.`group_actions` ;

CREATE TABLE IF NOT EXISTS `informationdb`.`group_actions` (

`id` INT NOT NULL AUTO_INCREMENT,

```

<div>Підп. і дата</div> <div>Взаєм. інс. №</div> <div>Інс. № дубл.</div> <div>Підп. і дата</div> <div>Інс. № підп</div>						<div>Ошибка! Неизвестное имя свойства документа.»</div>	Арк.
	Зм	Арк.	№ докум.	Підпис	Дат		25

```

`name` VARCHAR(45) NOT NULL,

`class_name` VARCHAR(45) NOT NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB;

DROP TABLE IF EXISTS `informationdb`.`empl_groups_actions` ;

CREATE TABLE IF NOT EXISTS `informationdb`.`empl_groups_actions` (

`group_id` INT NOT NULL,

`action_id` INT NOT NULL,

PRIMARY KEY (`group_id`, `action_id`),

CONSTRAINT `type_to_id`

FOREIGN KEY (`action_id`)

REFERENCES `informationdb`.`group_actions` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `to_type_id`

FOREIGN KEY (`group_id`)

REFERENCES `informationdb`.`employee_group` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

CREATE INDEX `to_type_id_idx` ON `informationdb`.`empl_groups_actions` (`group_id` ASC);

CREATE INDEX `type_to_id_idx` ON `informationdb`.`empl_groups_actions` (`action_id` ASC);

DROP TABLE IF EXISTS `informationdb`.`login_data` ;

CREATE TABLE IF NOT EXISTS `informationdb`.`login_data` (

`employee_id` INT NOT NULL,

`login` VARCHAR(45) NOT NULL,

`password` VARCHAR(45) NOT NULL,

CONSTRAINT `to_employee`

FOREIGN KEY (`employee_id`)

REFERENCES `informationdb`.`employees` (`id`)

```

<div>Інв. № підп.</div> <div>Підп. і дата</div> <div>Інв. № дубл.</div> <div>Взаєм. інв. №</div> <div>Підп. і дата</div>						<div>Ошибка! Неизвестное имя свойства документа.»</div>	Арк.
	Зм	Арк.	№ докум.	Підпис	Дат		25

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

DROP TABLE IF EXISTS `informationdb`.`dossier_action_types` ;

CREATE TABLE IF NOT EXISTS `informationdb`.`dossier_action_types` (

`id` INT NOT NULL AUTO_INCREMENT,

`name` VARCHAR(45) NOT NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB;

DROP TABLE IF EXISTS `informationdb`.`dossiers_log` ;

CREATE TABLE IF NOT EXISTS `informationdb`.`dossiers_log` (

`dossier_id` INT NOT NULL,

`action_type` INT NOT NULL,

`employee_id` INT NOT NULL,

`date` DATETIME NOT NULL,

`id` INT NOT NULL AUTO_INCREMENT,

PRIMARY KEY (`id`),

CONSTRAINT `to_dossier_table`

FOREIGN KEY (`dossier_id`)

REFERENCES `informationdb`.`dossiers` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `to_employee_table`

FOREIGN KEY (`employee_id`)

REFERENCES `informationdb`.`employees` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `to_action_type_table`

FOREIGN KEY (`action_type`)

REFERENCES `informationdb`.`dossier_action_types` (`id`)

Ошибка! Неизвестное имя свойства документа.»

Арк.

25

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

CREATE INDEX `to_employee_idx` ON `informationdb`.`dossiers_log` (`employee_id` ASC);

CREATE INDEX `to_action_type_idx` ON `informationdb`.`dossiers_log` (`action_type` ASC);

USE `test_informationdb` ;

DROP TABLE IF EXISTS `test_informationdb`.`artif_action_types` ;

CREATE TABLE IF NOT EXISTS `test_informationdb`.`artif_action_types` (

`id` INT(11) NOT NULL AUTO_INCREMENT,

`name` VARCHAR(45) NOT NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB

AUTO_INCREMENT = 4

DEFAULT CHARACTER SET = utf8;

DROP TABLE IF EXISTS `test_informationdb`.`groups` ;

CREATE TABLE IF NOT EXISTS `test_informationdb`.`groups` (

`id` INT(11) NOT NULL AUTO_INCREMENT,

`name` VARCHAR(45) NOT NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

DROP TABLE IF EXISTS `test_informationdb`.`artifact_types` ;

CREATE TABLE IF NOT EXISTS `test_informationdb`.`artifact_types` (

`id` INT(11) NOT NULL AUTO_INCREMENT,

`name` VARCHAR(45) NOT NULL,

`group_id` INT(11) NOT NULL,

PRIMARY KEY (`id`),

CONSTRAINT `group_id`

FOREIGN KEY (`group_id`)

REFERENCES `test_informationdb`.`groups` (`id`)

Ошибка! Неизвестное имя свойства документа.»

Арк.

25

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

CREATE INDEX `group_id_idx` ON `test_informationdb`.`artifact_types` (`group_id` ASC);

DROP TABLE IF EXISTS `test_informationdb`.`artifacts` ;

CREATE TABLE IF NOT EXISTS `test_informationdb`.`artifacts` (

`id` INT(11) NOT NULL AUTO_INCREMENT,

`artifact_type_id` INT(11) NOT NULL,

PRIMARY KEY (`id`),

CONSTRAINT `artifact_to_type`

FOREIGN KEY (`artifact_type_id`)

REFERENCES `test_informationdb`.`artifact_types` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

CREATE INDEX `artifact_to_type_idx` ON `test_informationdb`.`artifacts` (`artifact_type_id` ASC);

DROP TABLE IF EXISTS `test_informationdb`.`dossiers` ;

CREATE TABLE IF NOT EXISTS `test_informationdb`.`dossiers` (

`id` INT(11) NOT NULL AUTO_INCREMENT,

PRIMARY KEY (`id`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

DROP TABLE IF EXISTS `test_informationdb`.`employee_group` ;

CREATE TABLE IF NOT EXISTS `test_informationdb`.`employee_group` (

`id` INT NOT NULL AUTO_INCREMENT,

`name` VARCHAR(45) NOT NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB

Підп. і дата

Взаєм. інс. №

Інс. № дубл.

Підп. і дата

Інс. № підп

Ошибка! Неизвестное имя свойства документа.»

Арк.

25

Зм

Арк.

№ докум.

Підпис

Дат

```
DROP TABLE IF EXISTS `test_informationdb`.`employees` ;

CREATE TABLE IF NOT EXISTS `test_informationdb`.`employees` (

  `id` INT(11) NOT NULL AUTO_INCREMENT,

  `first_name` VARCHAR(45) NOT NULL,

  `last_name` VARCHAR(45) NOT NULL,

  `empl_group_id` INT NOT NULL,

  PRIMARY KEY (`id`),

  CONSTRAINT `to_empl_type`

    FOREIGN KEY (`empl_group_id`)

      REFERENCES `test_informationdb`.`employee_group` (`id`)

    ON DELETE NO ACTION

    ON UPDATE NO ACTION)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

CREATE INDEX `to_empl_type_idx` ON `test_informationdb`.`employees` (`empl_group_id` ASC);

DROP TABLE IF EXISTS `test_informationdb`.`actions` ;

CREATE TABLE IF NOT EXISTS `test_informationdb`.`actions` (

  `action_type_id` INT(11) NOT NULL,

  `artifact_id` INT(11) NOT NULL,

  `dossier_id` INT(11) NOT NULL,

  `employee_id` INT(11) NOT NULL,

  `date` DATETIME NOT NULL,

  `id` INT NOT NULL AUTO_INCREMENT,

  PRIMARY KEY (`id`),

  CONSTRAINT `to_action_type_id`

    FOREIGN KEY (`action_type_id`)

      REFERENCES `test_informationdb`.`artif_action_types` (`id`)

    ON DELETE NO ACTION

    ON UPDATE NO ACTION,

  CONSTRAINT `to_artifact_id`
```

Підп. і дата	
Взаєм. інв. №	
Інв. № дубл.	
Підп. і дата	
Інв. № підп	

Зм	Арк.	№ докум.	Підпис	Дат

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

CREATE INDEX `artifact_type_idx` ON `test_informationdb`.`fields` (`artifact_type_id` ASC);

DROP TABLE IF EXISTS `test_informationdb`.`field_values` ;

CREATE TABLE IF NOT EXISTS `test_informationdb`.`field_values` (

`artifact_id` INT(11) NOT NULL,

`value` VARCHAR(45) NOT NULL,

`field_id` INT(11) NOT NULL,

`id` INT NOT NULL AUTO_INCREMENT,

PRIMARY KEY (`id`),

CONSTRAINT `artifact_id`

FOREIGN KEY (`artifact_id`)

REFERENCES `test_informationdb`.`artifacts` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `field_id`

FOREIGN KEY (`field_id`)

REFERENCES `test_informationdb`.`fields` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

CREATE INDEX `field_id_idx` ON `test_informationdb`.`field_values` (`field_id` ASC);

CREATE INDEX `artifact_id_idx` ON `test_informationdb`.`field_values` (`artifact_id` ASC);

DROP TABLE IF EXISTS `test_informationdb`.`group_actions` ;

CREATE TABLE IF NOT EXISTS `test_informationdb`.`group_actions` (

`id` INT NOT NULL AUTO_INCREMENT,

`name` VARCHAR(45) NOT NULL,

Підп. і дата

Взаєм. інс. №

Інс. № дубл.

Підп. і дата

Інс. № підп

Ошибка! Неизвестное имя свойства документа.»

Арк.

25

Зм

Арк.

№ докум.

Підпис

Дат

```

`class_name` VARCHAR(45) NOT NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB;

DROP TABLE IF EXISTS `test_informationdb`.`empl_groups_actions` ;

CREATE TABLE IF NOT EXISTS `test_informationdb`.`empl_groups_actions` (

`group_id` INT NOT NULL,

`action_id` INT NOT NULL,

PRIMARY KEY (`group_id`, `action_id`),

CONSTRAINT `type_to_id`

FOREIGN KEY (`action_id`)

REFERENCES `test_informationdb`.`group_actions` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `to_type_id`

FOREIGN KEY (`group_id`)

REFERENCES `test_informationdb`.`employee_group` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

CREATE INDEX `to_type_id_idx` ON `test_informationdb`.`empl_groups_actions` (`group_id` ASC);

CREATE INDEX `type_to_id_idx` ON `test_informationdb`.`empl_groups_actions` (`action_id` ASC);

DROP TABLE IF EXISTS `test_informationdb`.`login_data` ;

CREATE TABLE IF NOT EXISTS `test_informationdb`.`login_data` (

`employee_id` INT NOT NULL,

`login` VARCHAR(45) NOT NULL,

`password` VARCHAR(45) NOT NULL,

CONSTRAINT `to_employee`

FOREIGN KEY (`employee_id`)

REFERENCES `test_informationdb`.`employees` (`id`)

```

Підп. і дата	
Взаєм. інв. №	
Інв. № дубл.	
Підп. і дата	
Інв. № підп	

Зм	Арк.	№ докум.	Підпис	Дат

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

DROP TABLE IF EXISTS `test_informationdb`.`dossier_action_types` ;

CREATE TABLE IF NOT EXISTS `test_informationdb`.`dossier_action_types` (

`id` INT NOT NULL AUTO_INCREMENT,

`name` VARCHAR(45) NOT NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB;

DROP TABLE IF EXISTS `test_informationdb`.`dossiers_log` ;

CREATE TABLE IF NOT EXISTS `test_informationdb`.`dossiers_log` (

`dossier_id` INT NOT NULL,

`action_type` INT NOT NULL,

`employee_id` INT NOT NULL,

`date` DATETIME NOT NULL,

`id` INT NOT NULL AUTO_INCREMENT,

PRIMARY KEY (`id`),

CONSTRAINT `to_dossier_table`

FOREIGN KEY (`dossier_id`)

REFERENCES `test_informationdb`.`dossiers` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `to_employee_table`

FOREIGN KEY (`employee_id`)

REFERENCES `test_informationdb`.`employees` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `to_action_type_table`

FOREIGN KEY (`action_type`)

REFERENCES `test_informationdb`.`dossier_action_types` (`id`)

Ошибка! Неизвестное имя свойства документа.»

Арк.

25

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

CREATE INDEX `to_employee_idx` ON `test_informationdb`.`dossiers_log` (`employee_id` ASC);

CREATE INDEX `to_action_type_idx` ON `test_informationdb`.`dossiers_log` (`action_type` ASC);

SET SQL_MODE=@OLD_SQL_MODE;

SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;

SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;

Інв. № підп	Підп. і дата	Інв. № дубл.	Взаєм. інв. №	Підп. і дата						Ошибка! Неизвестное имя свойства документа.»	Арк.
											25
Зм	Арк.	№ докум.	Підпис	Дат							

ДОДАТОК В

HibernateUtil.java

```
package configurations;

import model.entities.artifacts.*;
import model.entities.dossiers.Dossier;
import model.entities.employees.Employee;
import model.entities.employees.EmployeeGroup;
import model.entities.employees.GroupActions;
import model.entities.employees.LoginData;
import org.hibernate.HibernateException;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.AnnotationConfiguration;

/**
 * Created by Andrey on 24.02.2015.
 */
public class HibernateUtil {
    private static final SessionFactory sessionFactory;

    static {
        try {
            sessionFactory = new AnnotationConfiguration()
                .addAnnotatedClass(Employee.class)
                .addAnnotatedClass(EmployeeGroup.class)
                .addAnnotatedClass(LoginData.class)
                .addAnnotatedClass(GroupActions.class)

                .addAnnotatedClass(Action.class)
                .addAnnotatedClass(Artifact.class)
                .addAnnotatedClass(ArtifActionType.class)
                .addAnnotatedClass(ArtifactType.class)
                .addAnnotatedClass(Field.class)
                .addAnnotatedClass(FieldValue.class)
                .addAnnotatedClass(Group.class)

                .addAnnotatedClass(Dossier.class)

                .configure().buildSessionFactory();
        } catch (Throwable ex) {
            // Log exception!
            throw new ExceptionInInitializerError(ex);
        }
    }

    public static Session getSession()
        throws HibernateException {
        return sessionFactory.openSession();
    }
}
```

Підп. і дата		Взаєм. інв. №		Інв. № дубл.		Підп. і дата		Інв. № підп		
					<div style="border: 1px solid black; padding: 5px; display: inline-block;"> Ошибка! Неизвестное имя свойства документа. </div>					Арк. 25
Зм	Арк.	№ докум.	Підпис	Дат						

```

        public static SessionFactory getSessionFactory()
            throws HibernateException {
            return sessionFactory;
        }
    }
}

```

ActionListPageServlet.java

```

package controller;

import model.entities.employees.Employee;
import model.entities.employees.GroupActions;
import model.service.EmployeeService;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;
import java.util.Set;

/**
 * Created by Andrey on 21.02.2015.
 */
@WebServlet("/actions")
public class actionListPageServlet extends HttpServlet {
    @Override
    protected void doPost(HttpServletRequest req,
        HttpServletResponse resp) {

    }
    @Override
    protected void doGet(HttpServletRequest req,
        HttpServletResponse resp) throws ServletException, IOException {
        EmployeeService service = new EmployeeService();

        HttpSession session = req.getSession(false);
        Employee employee = (Employee)
            session.getAttribute("employee");

        Set<GroupActions> groupActions =
            service.getEmployeeGroupActions(employee);

        req.setAttribute("actionsSet", groupActions);
        req.setAttribute("pageName", "Список дій");

        req.getRequestDispatcher("/pages/actions/actionListPage.jsp").forward(req, resp);
    }
}

```

Інв. № підп.	Підп. і дата	Інв. № дубл.	Взам. інв. №	Підп. і дата	Зм	Арк.	№ докум.	Підпис	Дат	Ошибка! Неизвестное имя свойства документа.»	Арк.
											25

EmployeePageServlet.java

```
package controller;

import model.entities.artifacts.Action;
import model.entities.employees.Employee;
import model.service.EmployeeService;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.util.List;

/**
 * Created by Andrey on 15.03.2015.
 */
@WebServlet("/actions/userSearch/employee")
public class employeePageServlet extends HttpServlet {
    @Override
    protected void doPost(HttpServletRequest req,
        HttpServletResponse resp) {

    }
    @Override
    protected void doGet(HttpServletRequest req,
        HttpServletResponse resp) throws ServletException, IOException {
        EmployeeService service = new EmployeeService();

        String id = req.getParameter("id");
        Employee empl = service.getEmployee(id);

        List<Action> actionList =
            service.getEmployeeActions(empl);

        req.setAttribute("employee", empl);
        req.setAttribute("actionList", actionList);
        req.setAttribute("pageName", empl.getFirstName()+"
            "+empl.getLastName());

        req.getRequestDispatcher("/pages/actions/userSearch/employeeInfo
            Page.jsp").forward(req, resp);
    }
}
```

Інв. № підп.	Підп. і дата	Інв. № дубл.	Взаєм. інв. №	Підп. і дата						Арк. 25
Зм	Арк.	№ докум.	Підпис	Дат	Ошибка! Неизвестное имя свойства документа.»					

LogInServlet.java

```
package controller;

import model.entities.employees.Employee;
import model.service.EmployeeService;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;
import java.io.IOException;

/**
 * This class is responsible for guest identification.
 * It shows a login page and creates session if user's login and
 * pass are valid.
 * Created by Andrey on 21.02.2015.
 */
@WebServlet("/login")
public class LogInServlet extends HttpServlet {

    /**
     * Receives a login and password, and if a user with that
     * login data exists -
     * sets a session with with attribute of employee and
     * redirects to the user page.
     * If login and pass are not valid - redirects to the login
     * page.
     * @param req
     * @param resp
     * @throws ServletException
     * @throws IOException
     */
    @Override
    protected void doPost(HttpServletRequest req,
        HttpServletResponse resp) throws ServletException, IOException {
        EmployeeService service = new EmployeeService();

        String login = req.getParameter("inputLogin");
        String password = req.getParameter("inputPassword");

        Employee employee = service.getEmployee(login,
            password);

        if (service.isSessionPermitted(employee)) {

            HttpSession session = req.getSession();
            session.setAttribute("employee", employee);
            session.setAttribute("isAuthorised", true);
            session.setMaxInactiveInterval(30*60);

            resp.sendRedirect("/SCPI/actions");
        }
    }
}
```

Підп. і дата

Взаєм. інс. №

Інс. № дубл.

Підп. і дата

Інс. № підп

Зм

Арк.

№ докум.

Підпис

Дат

Ошибка! Неизвестное имя свойства документа.»

Арк.

25

```

        } else {
//            req.getRequestDispatcher("/login").forward(req,
resp);
            resp.sendRedirect("/SCPI/login");
        }
    }

    /**
     * If session is not valid - redirects to the login page. If
     valid - forwards the request.
     * @param req
     * @param resp
     * @throws ServletException
     * @throws IOException
     */
    @Override
    protected void doGet(HttpServletRequest req,
        HttpServletResponse resp) throws ServletException, IOException {
        if (req.getSession(false) != null) {
            resp.sendRedirect("/SCPI/actions");
            return;
        }

        req.getRequestDispatcher("/pages/authentication/login.jsp").forw
ard(req, resp);
    }
}

```

LogOutServlet.java

```

package controller;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;
import java.io.IOException;

/**
 * Created by Andrey on 21.02.2015.
 */
@WebServlet("/logout")
public class LogOutServlet extends HttpServlet {

    /**
     *
     * @param req
     * @param resp
     * @throws ServletException
     * @throws IOException
     */
    @Override
    protected void doPost(HttpServletRequest req,

```

```

HttpServletResponse resp) throws ServletException, IOException {
    HttpSession session = req.getSession();
    if(session != null){
        session.invalidate();
        resp.sendRedirect("/SCPI");
    } else {

req.getRequestDispatcher("/pages/authentication/login.jsp").forward(req, resp);
    }

    }

@Override
protected void doGet(HttpServletRequest req,
HttpServletResponse resp) throws ServletException, IOException {
    HttpSession session = req.getSession();
    if(session != null){
        session.invalidate();
        resp.sendRedirect("/SCPI");
    } else {

req.getRequestDispatcher("/pages/authentication/login.jsp").forward(req, resp);
    }

    }
}

```

SessionFilter.java

```

package controller;

import javax.servlet.*;
import javax.servlet.annotation.WebFilter;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;

/**
 * This filter calls after each request of a user.
 * It checks a session of the user and passes on the request if
 * session is valid,
 * and forwards to the login page in the other way.
 * Created by Andrey on 20.02.2015.
 */
@WebFilter
public class SessionFilter implements Filter {

    @Override
    public void init(FilterConfig filterConfig) throws
ServletException {

```

```

    }

    /**
     * Checks a request for active session. If there i no
     session or
     * a session is not valid - redirects to login page. In
     other way - forwards the request.
     *
     * @param servletRequest
     * @param servletResponse
     * @param filterChain
     * @throws IOException
     * @throws ServletException
     */
    @Override
    public void doFilter(ServletRequest servletRequest,
        ServletResponse servletResponse, FilterChain filterChain) throws
        IOException, ServletException {
        HttpServletRequest req = (HttpServletRequest)
servletRequest;
        HttpServletResponse res = (HttpServletResponse)
servletResponse;
        HttpSession session = req.getSession(false);

        if (session != null &&
session.getAttribute("isAuthorised")==true) {
            filterChain.doFilter(req, res);
        } else if (req.getRequestURI().equals("/SCPI/login") ||
req.getRequestURI().startsWith("/SCPI/view")) {
            //"/SCPI/view" is the path of view components like
css and js and need to be allowed from all of the pages
            filterChain.doFilter(req, res);
        } else {

req.getRequestDispatcher("/pages/authentication/login.jsp").forw
ard(req, res);
        }
    }

    @Override
    public void destroy() {
    }
}

```

StandartDataSetterFilter.java

```

package controller;

import model.entities.employees.Employee;
import org.hibernate.Session;

```


Інв. № підп	Підп. і дата	Інв. № дубл.	Взаєм. інв. №	Підп. і дата

```
package controller;
```

Зм	Арк.	№ докум.	Підпис	Дат

Арк.
25

```

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.util.List;

/**
 * Created by Andrey on 03.03.2015.
 */
@WebServlet("/actions/userSearch")
public class UserSearchServlet extends HttpServlet {

    @Override
    public void doPost(HttpServletRequest req,
        HttpServletResponse resp) throws ServletException, IOException {
        EmployeeService service = new EmployeeService();

        List<Employee> eList =
            service.getEmployees(req.getParameter("id"),
                req.getParameter("firstName"), req.getParameter("lastName"));

        req.setAttribute("employeeList", eList);
        req.setAttribute("pageName", "Пошук співробітників");

        req.getRequestDispatcher("/pages/actions/userSearch/searchResult
.jsp").forward(req, resp);
    }

    @Override
    public void doGet(HttpServletRequest req,
        HttpServletResponse resp) throws ServletException, IOException {
        req.setAttribute("pageName", "Пошук співробітників");

        req.getRequestDispatcher("/pages/actions/userSearch/searchForm.j
sp").forward(req, resp);
    }
}

```

EmployeeService.java

```

package model.service;

import model.dao.ActionsDao;
import model.dao.EmployeeDao;
import model.entities.artifacts.Action;
import model.entities.employees.Employee;
import model.entities.employees.EmployeeGroup;
import model.entities.employees.GroupActions;

import java.util.List;
import java.util.Set;

```

```

/**
 * Service class for operations with employee.
 * Created by Andrey on 24.03.2015.
 */
public class EmployeeService {

    /**
     * Looking for employees with specified data. Searching
     request includes only that data,
     * that is not equal empty string or null.
     * 

For example, if id==1, firstName==null, lastName=="",
     * than searching request will include only employees with
     id==1.
     * 

If id==null, firstName=="Andrey", lastName=="Portman",
     than the request will
     * consist of employee with specified firstName and
     lastName.
     *
     * @param id of employee
     * @param firstName of employee
     * @param lastName of employee
     * @return a list of Employee with specified data.
     * @throws java.lang.NullPointerException if all parameters
     are null
     */
    public List<Employee> getEmployees(String id, String
    firstName, String lastName) {
        String emptyStr = "";

        if (id == null && firstName == null && lastName == null)
        throw new NullPointerException();

        if (emptyStr.equals(id)) { //marks empty string as null
            id = null;
        }

        if (emptyStr.equals(firstName)) {
            firstName = null;
        }

        if (emptyStr.equals(lastName)) {
            lastName = null;
        }

        return new EmployeeDao().getEmployees(id, firstName,
        lastName);
    }

    /**
     * Returns a set of actions, that employee can do in the
     system. Depends on EmployeeGroup,
     * that employee belongs to.
     * @param employee
     * @return set of GroupActions
     */


```

```

    */
    public Set<GroupActions> getEmployeeGroupActions(Employee
employee) {
        EmployeeGroup emplGroup = employee.getEmployeeGroup();
        return emplGroup.getActions();
    }

    /**
     * Returns an employee by specified id.
     * @param id of employee need to be returned
     * @return a <code>Employee</code> with specified id
     */
    public Employee getEmployee(String id) {
        return new EmployeeDao().getEmployee(id);
    }

    /**
     * Returns actions, that employee made in the system (like
added new artifact)
     * @param empl employee whose actions will be returned
     * @return a list of <code>Action</code>, that employee made
     */
    public List<Action> getEmployeeActions(Employee empl) {
        return new ActionsDao().getEmployeeActions(empl);
    }

    /**
     * Return an employee by specified login and password.
     * @param login of employee
     * @param password of employee
     * @return an <code>Employee</code> with specified login and
password
     */
    public Employee getEmployee(String login, String password) {
        return new EmployeeDao().getEmployee(login, password);
    }

    /**
     * Checks if employee can be authorised
     * @param empl an employee to be checked
     * @return true if employee can be authorised
     */
    public boolean isSessionPermitted(Employee empl) {
        return empl != null;
    }

    /**
     * Return a full name of specified employee.
     * @param empl Employee whose name need to be returned
     * @return a full name of <code>Employee</code>
     */
    public String getEmployeeFullName(Employee empl) {
        if (empl==null) throw new NullPointerException();
    }

```

Підп. і дата	
Взаєм. інв. №	
Інв. № дубл.	
Підп. і дата	
Інв. № підп.	

Зм	Арк.	№ докум.	Підпис	Дат

Ошибка! Неизвестное имя свойства документа.»

Арк.

25

Інв. № пілп	Піпп. і дата	Інв. № дубл.	Взаєм. інв. №	Піпп. і дата

```
package model.dao;
```

```
import java.util.List;
```

```
public class ActionsDao {
    private static SessionFactory factory =
HibernateUtil.getSessionFactory();
```

```
public List<Action> getEmployeeActions(Employee empl) {
    if (empl==null) throw new NullPointerException();

    Session session = factory.getCurrentSession();
    Transaction ta = null;

    session.getTransaction().begin();

    Criteria cr = session.createCriteria(Action.class);

    Criterion criterion = Restrictions.like("employee", empl
);
    List<Action> list = cr.add(criterion).list();
    return list;
}
```

EmployeeDao.java

```
package model.dao;

import model.entities.employees.Employee;
import model.entities.employees.LoginData;
import configurations.HibernateUtil;
import org.hibernate.*;
import org.hibernate.criterion.Criterion;
import org.hibernate.criterion.Restrictions;

import java.util.List;

/**
 * Created by Andrey on 28.02.2015.
 */
public class EmployeeDao {
    private static SessionFactory factory =
        HibernateUtil.getSessionFactory();

    /**
     * Returns an employee by specified id.
     * @param id of employee
     * @return Employee
     */
    public Employee getEmployee(String id) {
        Session s = factory.getCurrentSession();
        s.getTransaction().begin();
        return (Employee) s.get(Employee.class, new
        Integer(id));
    }

    /**
     * Looking for employees by parameters, that are not equal
     * null. At least one parameter must be not null.
     * @param id
     * @param firstName
     * @param lastName
     * @return list of Employee
     */
    public List<Employee> getEmployees(String id, String
    firstName, String lastName) {
        if (id==null && firstName==null && lastName==null) throw
        new IllegalArgumentException();
        List<Employee> employees = null;
        Session session = factory.getCurrentSession();
        Transaction ta = null;

        try {
            ta = session.beginTransaction();
            Criteria cr =
            session.createCriteria(Employee.class);
            Criterion criterion = null;
```

```

        if (id != null) {
            criterion = Restrictions.like("id", new
Integer(id));
        }
        if (firstName != null) {
            if (criterion == null) {
                criterion = Restrictions.like("firstName",
firstName);
            } else {
                criterion =
Restrictions.and(Restrictions.like("firstName", firstName),
criterion);
            }
        }
        if (lastName != null) {
            if (criterion == null) {
                criterion = Restrictions.like("lastName",
lastName);
            } else {
                criterion =
Restrictions.and(Restrictions.like("lastName", lastName),
criterion);
            }
        }
        cr.add(criterion);
        employees = cr.list();

//        ta.commit();
    } catch (HibernateException e) {
        e.printStackTrace();
    } finally {

//        session.close();
    }

    return employees;
}

/**
 * Returns an employee by specified login and password.
 * @param login login of employee
 * @param password password of employee
 * @return <code>Employee</code> by specified login and
password
 */
public Employee getEmployee(String login, String password) {

    Employee employee = null;
    LoginData employeeLD = null;

    Session session = factory.getCurrentSession();
    Transaction ta = null;

```

Підп. і дата	
Взаєм. інв. №	
Інв. № дубл.	
Підп. і дата	
Інв. № підп	

					Ошибка! Неизвестное имя свойства документа.»	Арк.
						25
Зм	Арк.	№ докум.	Підпис	Дат		

```

        try {
            ta = session.beginTransaction();
            Criteria cr =
session.createCriteria(LoginData.class);
            Criterion loginCr = Restrictions.like("login",
login);
            Criterion passwordCr = Restrictions.like("password",
password);
            cr.add(Restrictions.and(loginCr, passwordCr));
            employeeLD = (LoginData) cr.uniqueResult();

//            ta.commit();
        } catch (HibernateException e) {
            e.printStackTrace();
        } finally {

//            session.close();
        }
        if (employeeLD == null) {
            employee = null;
        } else {
            employee = employeeLD.getEmployee();
        }

        return employee;
    }
}

```

Інв. № підп	Підп. і дата	Інв. № дубл.	Взам. інв. №	Підп. і дата						Ошибка! Неизвестное имя свойства документа.»	Арк.
											25
					Зм	Арк.	№ докум.	Підпис	Дат		