МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний економічний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерних наук

ОЛІЙНИК Олег Ігорович

Програмна система для паспортизації земельних ділянок/ Software system for passportization the plots of land

напрям підготовки: 6.050103 - Програмна інженерія
фахове спрямування - Програмне забезпечення систем

Бакалаврська дипломна робота

Виконав студент групи ПЗС-41
О. І. Олійник

Науковий керівник:
викладач ОЛІЙНИК І.С.

Бакалаврську дипломну роботу
допущено до захисту:

"___" _____ 20___ р.

Завідувач кафедри
_____ **А. В. Пукас**

ТЕРНОПІЛЬ - 2016

РЕЗЮМЕ

Дипломна робота містить 77 сторінки, 21 таблиць, 37 рисунків, список використаних джерел із 14 найменувань, 1 додаток.

Метою дипломної роботи є розробка програми для паспортизації земельної ділянки та електронного обліку документів.

Об'єктом дослідження є процес формування електронного паспорту земельної ділянки для купівлі-продажу земельних ресурсів.

Предметом дослідження є методи та засоби для розробки програмного продукту для формування стандартизованого xml файлу.

Одержані результати полягають у розробці електронного паспорту земельної ділянки.

Ключові слова: обмінний файл; державна організація; паспорт земельної ділянки; програмне забезпечення.

SUMMARY

Thesis contains 77 pages, 21 tables, 37 figures, list of sources with 14 titles, 1 application.

The aim of the thesis is certification program is the development of land for the electronic registration of documents.

Object of research is the formation of e-passports for land purchase and sale of land resources.

The subject of research there are methods and tools for developing software to create standardized xml file.

The resulting is the preparation of receiving e-passport to land.

Keywords: exchange file; State organization; e-passport land; Software.

ЗМІСТ

ВСТУП

В зв’язку із постійним розвитком ринку земельних ресурсів в Україні зростає необхідність в їх обліку. Така інформація є дуже важливою, адже земля часто стає об’єктом правових операцій, зокрема купівлі-продажу, оренди, вирішення спорів. Тому процес документообігу у сфері землевпорядкування є багаторівневим, трудомістким та складним. Згідно вимог законодавства України, основою формування документів для земельної ділянки є створення паспорту. Це складна, багатоетапна юридична процедура, що здійснюється державними реєстраторами. Паспорт земельної ділянки – це документ, яким держава засвідчує приватній особі чи підприємству право власності на земельний ресурс.

Метою роботи є аналіз існуючих спеціалізованих програмних систем, та на його основі визначених переваг й недоліків створення нового програмного засобу – системи для паспортизації земельної ділянки.

Паспортизація земельної ділянки є складним та важливим процесом, кожен етап якого необхідно контролювати. Для отримання паспорту земельної ділянки спочатку необхідно подати заяву до підприємства, що спеціалізується на землевпорядній діяльності. Після опрацювання заяви її реєструють та відправляють на електронну паспортизацію. На цьому етапі відбувається внесення даних про ділянку до системи та створення обмінного файлу у форматі .xml. На сьогодні у більшості землевпорядних підприємств наповнення та створення обмінного файлу відбувається в напівавтоматичному режимі, коли більшість даних необхідно вносити вручну, тому актуальною є розробка системи для паспортизації земельних ділянок, що автоматизує створення та формування обмінного файлу.

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Коротка характеристика об’єкту управління

Паспортизація земельної ділянки – це процес, що здійснює облік та контроль за земельним ресурсом. Паспорт земельної ділянки – це обмінний файл, який містить головні дані про ділянку, зокрема, кадастровий номер, розміри, інформацію про власника, місце розташування ділянки та її вартість.

Процес паспортизації земельної ділянки включає:

- ведення обліку земельних ділянок, їх розмірів, опису та площі;
- визначення видів дозволеного використання, обмеження та обтяження;
- ведення обліку нерухомості на земельних ділянках;
- ведення правовстановлюючих документів і документів, що підтверджують припинення права власності на земельні ділянки;
- ведення картотеки правовласників: як фізичних осіб з їх паспортними даними, так і юридичних осіб з повним складом реквізитів;
- ведення повного переліку власників для кожної земельної ділянки;
- ведення журналу реєстрації договорів оренди та договорів купівлі-продажу земельних ділянок;
- створення електронної картотеки для зберігання договорів оренди та купівлі-продажу;
- забезпечення миттєвого пошуку потрібного договору в електронній картотеці;
- перегляд та друк будь-якого договору з електронної картотеки, незалежно від року його створення;

- формування договорів в автоматичному режимі з найменшим заповненням «вручну» або взагалі без заповнення;
- ведення розрахунку орендної плати на земельній ділянці, що перебуває у державній власності;
- ведення розрахунку нарахування земельного податку;
- формування відомостей про земельні ділянки в податковий орган за звітний період;
- перегляд відомостей про земельні ділянки та їх правовласників за попередні роки.

Важливою умовою оптимізації використання та охорони земель є інформаційне висвітлення питань, які дають змогу конкретизувати ситуацію стосовно певних природніх і господарських показників, притаманних певній ділянці. Відсутність тих чи інакших характеристик може кардинально вплинути на такі критерії як результати розрахунків і вартість, навіть якщо роботи виконані за коректними алгоритмами. Це стосується і випадків розробки землевпорядної проектної документації щодо використання і охорони земель і різних видів оцінки земель.

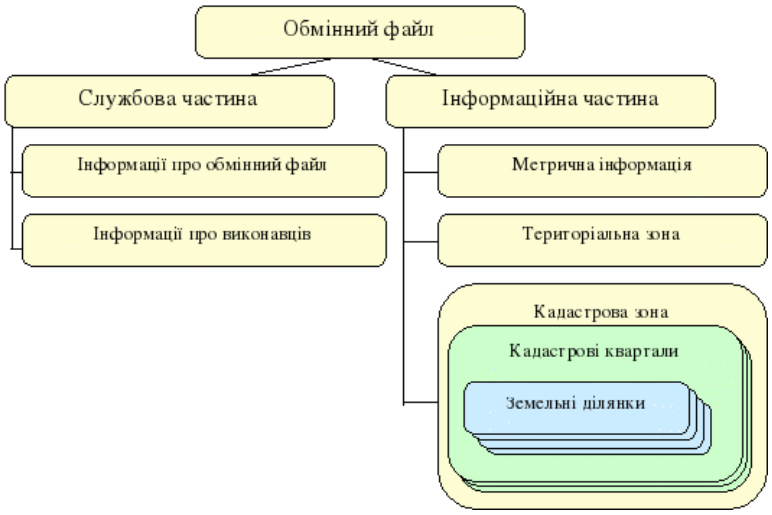


Рисунок 1.1 – Структура обмінного файлу

Обмінний файл земельної ділянки – це файл формату .xml, що використовується для збереження даних і внесення змін про земельну ділянку в базах земельних ресурсів. Приклад структури обмінного файлу наведений на рисунку 1.1.

Для автоматизації процесу паспортизації необхідний програмний продукт, простий у використанні, але достатньо функціональний, щоб оптимізувати процес створення обмінного файлу.

1.2 Опис предметної області

Аналіз предметної області показав, що можна виділити п’ять основних бізнес-процесів, які відповідають функціоналу роботи системи. Зокрема:

- авторизація користувача в системі;
- перегляд зібраних даних;
- формування документу;
- редагування документу;
- пошук ділянки.

На рисунку 1.2. зображено діаграму ієрархії даних процесів.

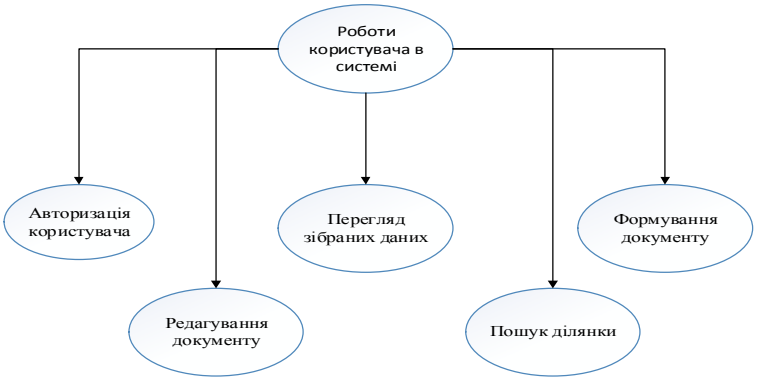


Рисунок 1.2 – Діаграма ієрархії бізнес-процесів для роботи користувача із системою паспортизації земельних ділянок

На основі аналізу бізнес-процесу авторизації користувача можна відобразити діаграму функцій цього процесу (рис. 1.3).

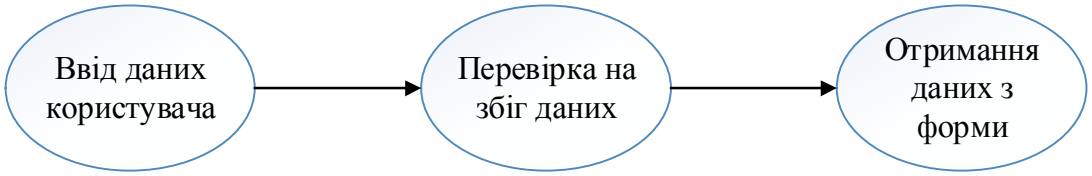


Рисунок 1.3 – Діаграма бізнес-процесу авторизації користувача

Для авторизації користувача в системі, йому необхідно при старті програми заповнити форму з полями «Логін» та «Пароль». Після чого дані відправляються на перевірку і якщо такі дані знайдені, то відбувається отримання доступу до робочої зони користувача. При помилці авторизації користувач зможе повторити спробу.

Таблиця 1.1

Характеристика бізнес-процесу авторизації користувача в системі

Назва характеристики	Значення характеристики
Ім'я бізнес-процесу	Авторизація користувача
Основні учасники	Адміністратор ,користувач
Вхідна подія	Перехід на сторінку авторизації
Вхідні документи	Дані з форми
Вихідна подія	Вхід в робочий кабінет
Вихідні документи	-
Клієнт бізнес-процесу	Адміністратор

На рисунку 1.4 зображено діаграму функцій бізнес-процесу редагування документу.

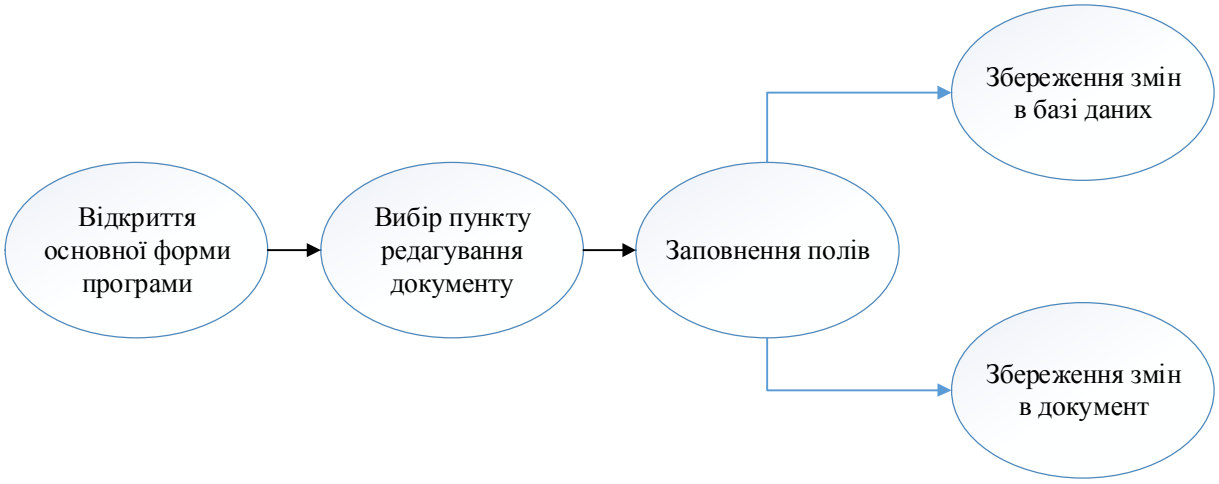


Рисунок 1.4 – Діаграма бізнес-процесу редагування документу

Для роботи з процесом редагування документу користувачу необхідно обрати на формі розділ «Редагувати документ». Після цього йому відкриється форма з полями, потім необхідно завантажити файл, з якого зчитуються дані і встановлюються в поля. Після зміни значень полів необхідно зберегти усі зміни.

Таблиця 1.2

Характеристика бізнес-процесу редагування документу

Назва характеристики	Значення характеристики
Ім'я бізнес-процесу	Редагування документу
Основні учасники	Адміністратор
Вхідна подія	Відкриття форми; завантаження значень;
Вхідні документи	Дані з форми
Вихідна подія	Збереження змін
Вихідні документи	Змінений вміст документу
Клієнт бізнес-процесу	Адміністратор

На рисунку 1.5 – зображено діаграму функцій бізнес-процесу перегляду даних земельних ділянок.

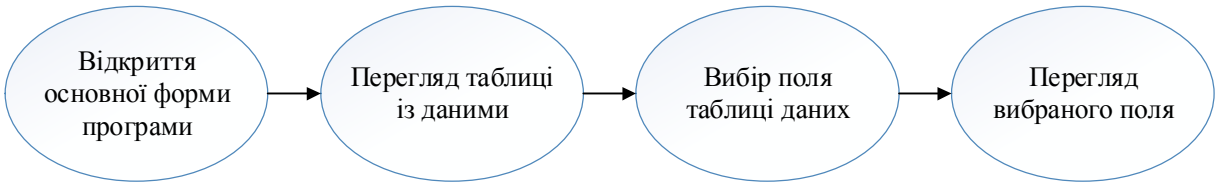


Рисунок 1.5 – Діаграма бізнес-процесу перегляду даних земельних ділянок

Для того, щоб здійснювати перегляд даних користувачу необхідно відкрити головну форму, на якій відображена таблиця, вміст якої завантажується з бази даних.

Таблиця 1.3

Характеристика бізнес-процесу перегляду даних

Назва характеристики	Значення характеристики
Ім'я бізнес-процесу	Перегляд даних
Основні учасники	Адміністратор, користувач
Вхідна подія	Відкриття форми завантаження значень
Вхідні документи	Таблиця з даними
Вихідна подія	-
Вихідні документи	-
Клієнт бізнес-процесу	Користувач

На рисунку 1.6 зображено діаграму бізнес-процесу перевірки коректності даних.

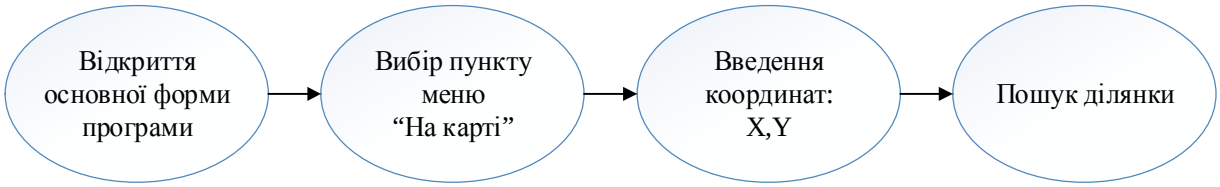


Рисунок 1.6 – Діаграма бізнес-процесу пошуку ділянки

Цей процес дозволяє користувачу переглянути на карті точні дані, які він використовує. Для того, щоб дана функція була доступна користувачу необхідно вибрати пункт «Показати на карті», після чого з’явиться вікно для введення координат. Після введення координат необхідно використати функцію пошуку, яка дозволить відкрити у новому вікні карту, що відповідає введеним даним.

Таблиця 1.4

Характеристика бізнес-процесу пошуку ділянки

Назва характеристики	Значення характеристики
Ім'я бізнес-процесу	Пошук ділянки
Основні учасники	Користувач
Вхідна подія	Відкриття форми з полями для введення координат
Вхідні документи	Координати
Вихідна подія	Форма із зображенням
Вихідні документи	-
Клієнт бізнес-процесу	Користувач

На рисунку 1.7 зображено діаграму бізнес-процесу формування документу.

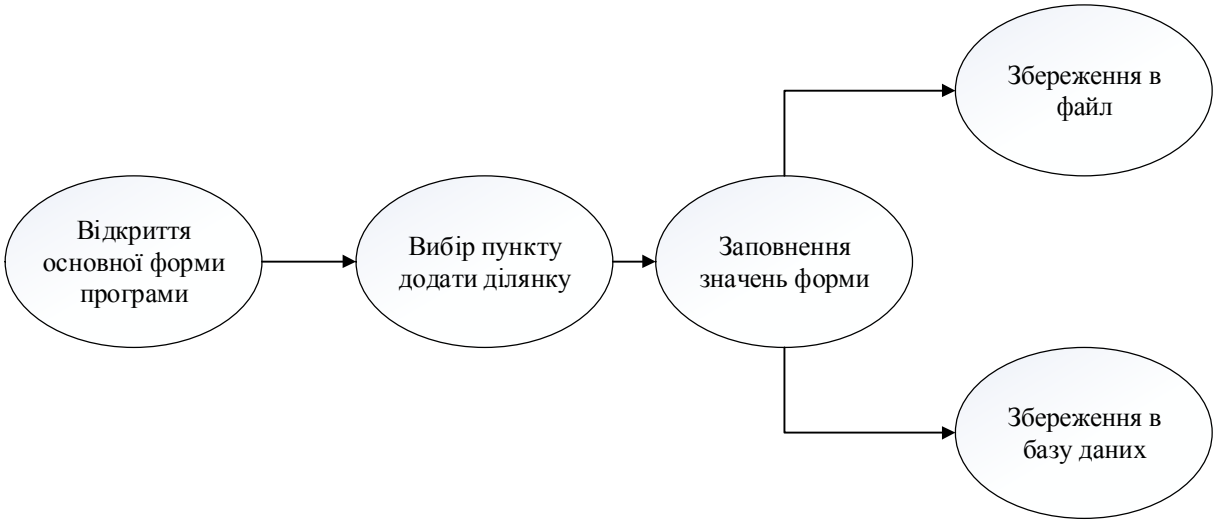


Рисунок 1.7 – Діаграма бізнес-процесу формування документу

Для роботи з даною функцією користувачу потрібно вибрати розділ «Додати ділянку». Далі необхідно коректно заповнити усі поля. На наступному етапі користувач повинен вибрати - додати дані в базу даних чи сформувати в обмінний файл, або ж виконати одночасно обидві дії.

Таблиця 1.5

Характеристика бізнес-процесу формування документу

Назва характеристики	Значення характеристики
Ім'я бізнес-процесу	Формування документу
Основні учасники	Адміністратор
Вхідна подія	Відкриття форми з полями для введення даних
Вхідні документи	Дані з форми
Вихідна подія	Збереження
Вихідні документи	Сформований обмінний файл
Клієнт бізнес-процесу	Адміністратор

1.3 Огляд і аналіз існуючих аналогів

Найпопулярнішими серед програмних продуктів для паспортизації земельних ділянок є Digitals, Topocad та AutoCAD.

Digitals – програмний засіб для землевпорядкування, геодезії і картографії. Створений багато років тому, як пакет для цифрової картографії, Digitals вже більше десяти років активно використовується в землеустрої. Його масове застосування почалося в епоху розподілу колгоспних земель на паї в кінці 90-х років. З тієї пори програма активно розвивалася і доповнювалася новими можливостями, перетворившись в підсумку на повноцінну землевпорядну ГІС. Потужне картографічне ядро, що дозволяє використовувати в одній карті тисячі растрових зображень і сотні тисяч векторних об'єктів в умовних знаках. Підтримка повного технологічного ланцюжка від обробки геодезичних вимірювань до роздруківки техдокументації. Запис і читання файлів у форматах популярних ГІС. Ось основні чинники, що забезпечили поширення програми в Україні та за її межами.

На рисунку 1.8 зображено вигляд програми Digitals.

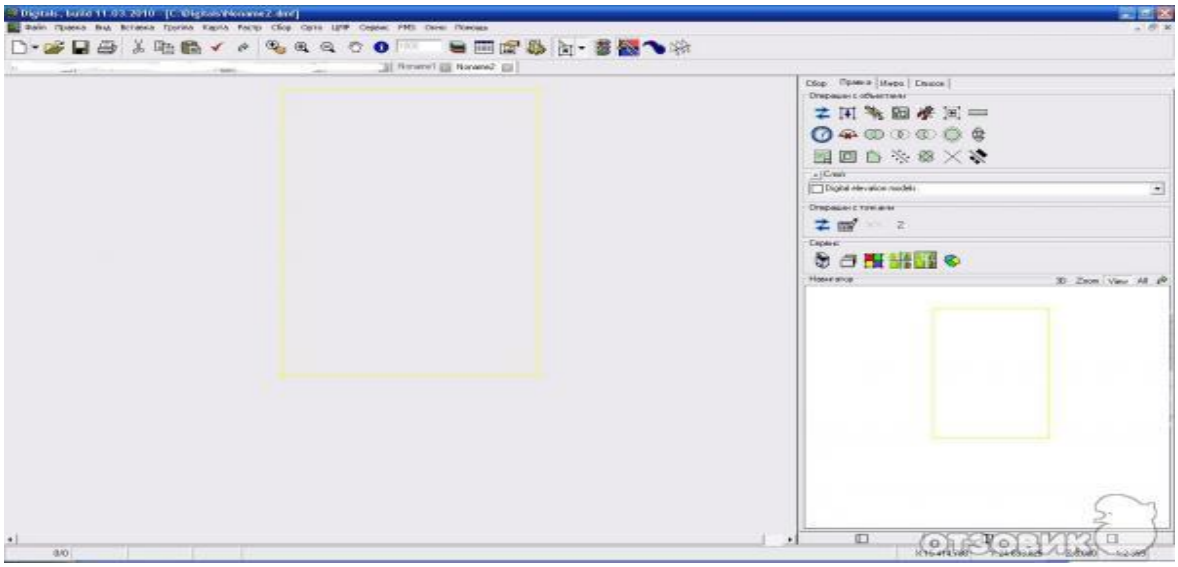


Рисунок 1.8 – Головне вікно програми Digitals

Topocad - це система автоматизованого проектування (САПР), створена спеціально для обробки результатів площинних і лінійних пошуків, створення ЦММ, підготовки топографічних креслень, геодезичного забезпечення будівництва, маркшейдерського забезпечення розробки родовищ корисних копалин, збору та оновлення даних ГІС.

На рисунку 1.9 зображено вигляд програми Topocad.

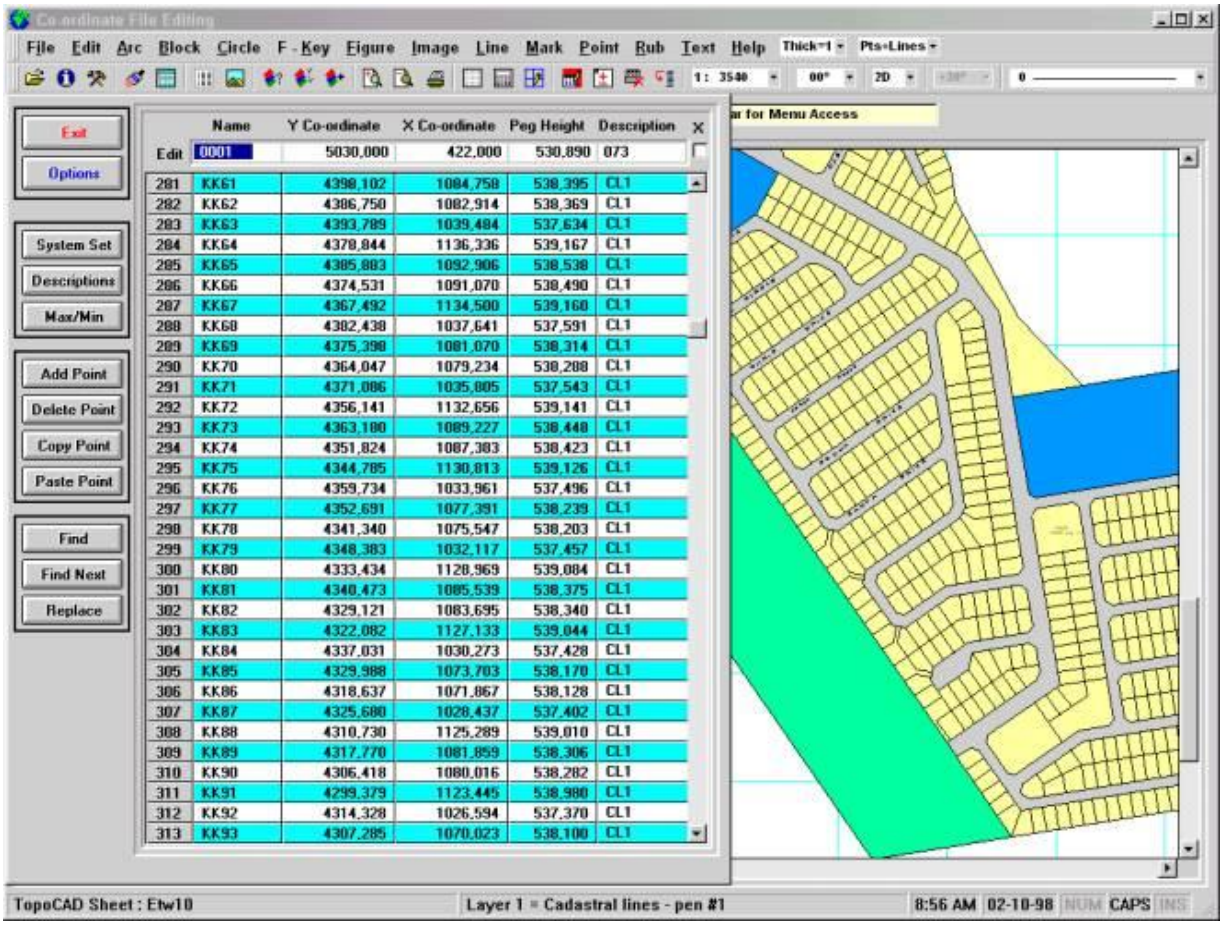


Рисунок 1.9 – Головне вікно програми Topocad

AutoCAD — дво- і тривимірна система автоматизованого проектування і креслення, розроблена компанією Autodesk. Перша версія була випущена в 1982 році. AutoCAD і спеціалізовані додатки на його основі знайшли широке

застосування в галузях геодезії. Вперше випущений в грудні 1982 року AutoCAD був однією з перших програм САПР для роботи на персональних комп'ютерах, зокрема, IBM PC.

На рисунку 1.10 зображено вигляд програми AutoCAD.

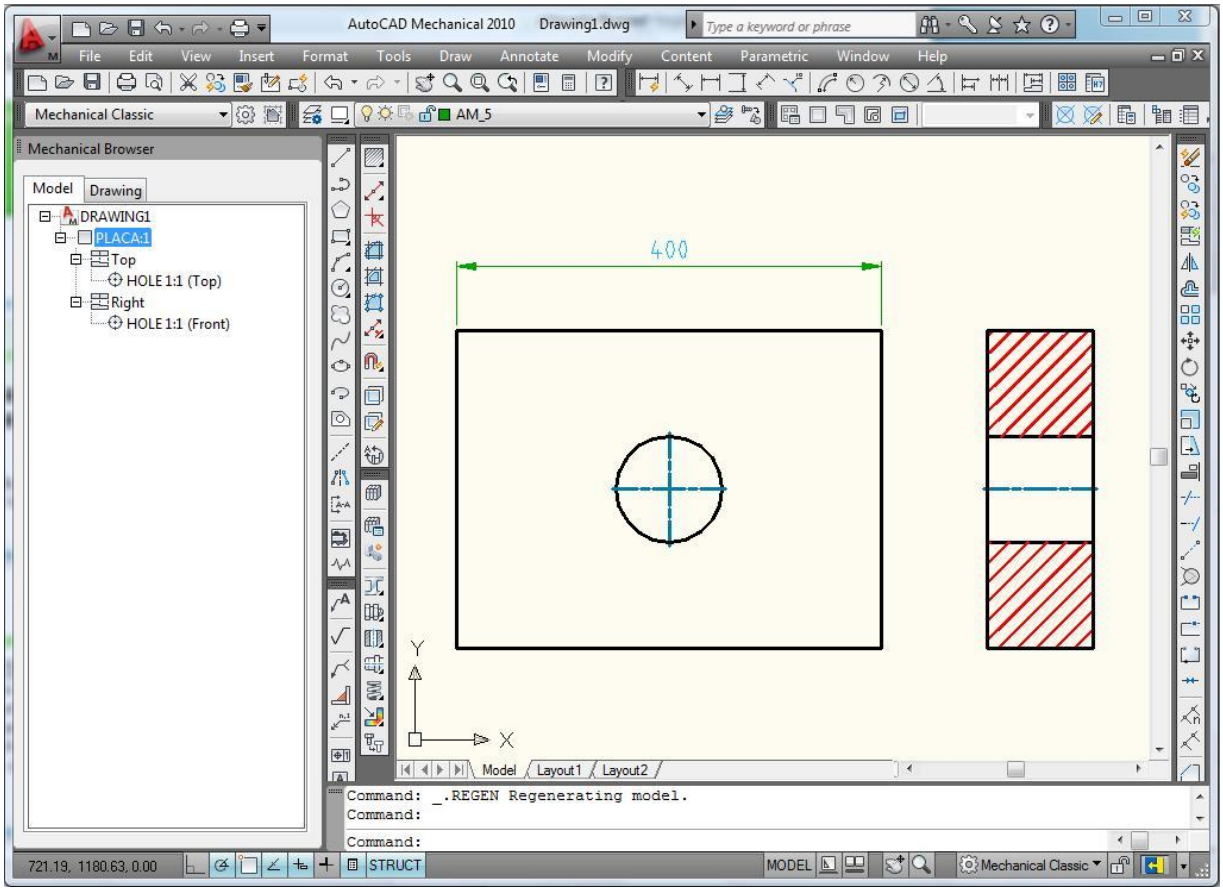


Рисунок 1.10 – Головне вікно програми AutoCAD

Таблиця 1.6

Порівняльна характеристика програмних продуктів

Фірма-розробник	Geosystem	Adtollo AB.	Autodesk
Назва програмного продукту	Digital	Topocad	AutoCAD
Версії продукту	Standard	14.2.2	20.0
Ціна	-	-	-
Доступність	-	+	+

Потреби у встановлені додаткового ПЗ	-	-	+
Звітність	+	-	-

1.4 Специфікація вимог до модуля

Специфікація вимог до системи – це опис поведінки системи, що розробляється. Включає прецеденти, які представляють собою опис функціональних і нефункціональних вимог. Визначає набір необхідних обмежень на реалізацію, що в свою чергу накладаються на програмний продукт.

У таблиці 7 наведено глосарій основних термінів, що використовуються в процесі проектування та розробки програмної системи для паспортизації земельних ділянок.

Таблиця 1.7

Глосарій основних термінів

Термін	Опис терміну
геодезія	наука, яка працює над зображення земної поверхні на планах
земельна ділянка	частина земної поверхні з установленими межами, певним місцем розташування, з визначеними щодо неї правами (стаття 79 Земельного кодексу України).
паспортизація	процес послідовного збору, узагальнення та зберігання відомостей про земельну ділянку.
обмінний файл	електронний файл, який призначений для внесення змін про земельні ділянки до баз даних управлінь земельних ресурсів та центрів державного реєстрування
кадастр	упорядкована геоінформаційна система про правове, природне, господарське, економічне та просторове розміщення об'єктів, що підлягають обліку в системі відповідного рівня управління
автоматизована система	система, що ґрунтується на комплексному

	використанні технічних, математичних, інформаційних та організаційних засобів для управління складними технічними й економічними об'єктами

Діаграму варіантів використання для користувача (див. рис. 1.11)

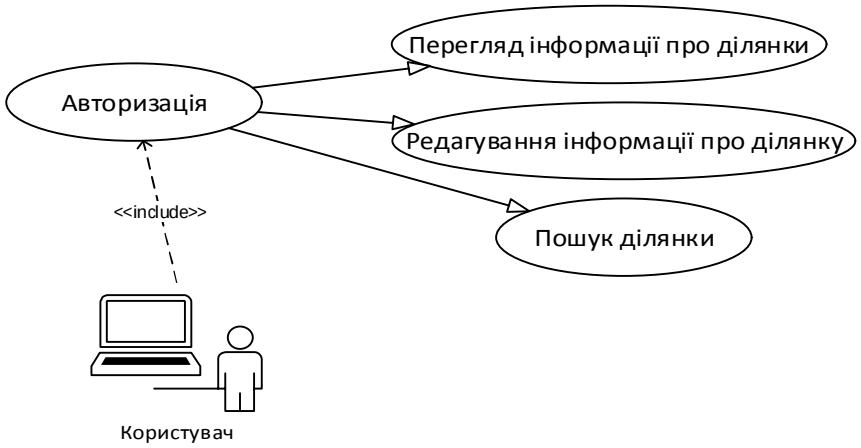


Рисунок 1.11 – Діаграма варіантів використання для геодезиста

Діаграму варіантів використання для адміністратора зображена нижче:

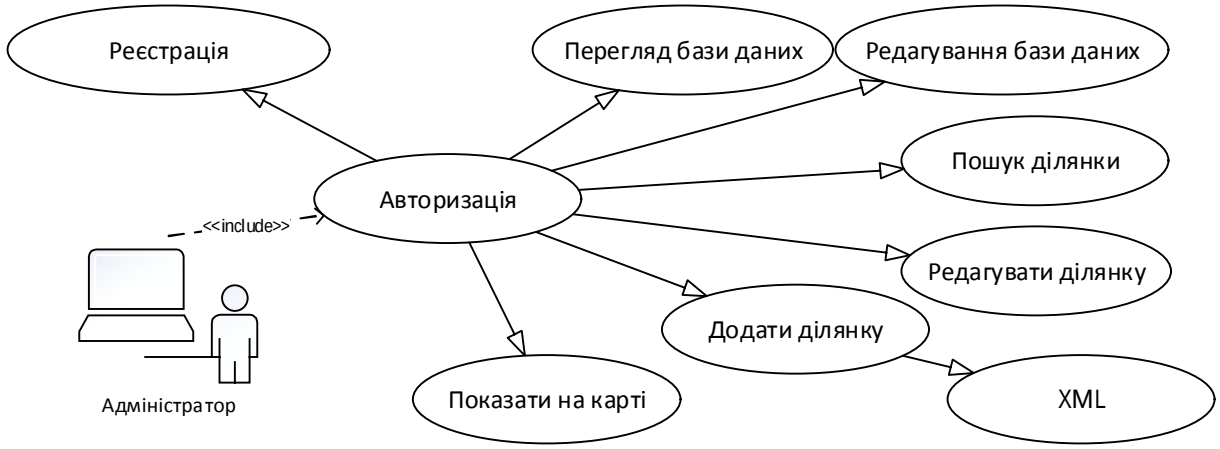


Рисунок 1.12 – Діаграма варіантів використання для адміністратора

Детальний опис усіх найважливіших характеристик варіантів використання для адміністратора та користувача подано у вигляді таблиць 1.8 – 1.15.

Таблиця 1.8

Варіант використання «Авторизація»

Контекст використання	Вхід в систему
Дійові особи	Користувач, Адміністратор
Передумова	Немає
Тригер	Немає
Сценарій	1. Запустити програму 2. Ввести логін та пароль 3. Натиснути “Ввійти”
Постумова	Немає

Таблиця 1.9

Варіант використання «Реєстрація користувача»

Контекст використання	Реєстрація
Дійові особи	Користувач, Адміністратор
Передумова	Немає
Тригер	Немає
Сценарій	1. Запустити програму 2. Ввести логін та пароль 3. Натиснути “Зареєструватися”
Постумова	Немає

Таблиця 1.10

Варіант використання «Перегляд бази даних»

Контекст використання	Перегляд бази даних

Дійові особи	Адміністратор
Передумова	Авторизація
Тригер	Немає
Сценарій	1. Запустити програму 2. Ввести логін та пароль 3. Натиснути “Ввійти”
Постумова	Рівень доступу повинен бути >2

Таблиця 1.11

Варіант використання «Редагувати базу даних»

Контекст використання	Редагування
Дійові особи	Адміністратор
Передумова	Авторизація
Тригер	Немає
Сценарій	1. Запустити програму 2. Ввести логін та пароль 3. Натиснути “Ввійти” 4. Натиснути на поле редагування
Постумова	Рівень доступу повинен бути >1

Таблиця 1.12

Варіант використання «Пошук»

Контекст використання	Пошук
Дійові особи	Адміністратор, Користувач
Передумова	Авторизація
Тригер	Немає
Сценарій	1. Запустити програму 2. Ввести логін та пароль 3. Натиснути “Ввійти” 4. Ввести дані в поле пошук 5. Натиснути клавішу “Пошук”

Постумова	Рівень доступу повинен бути >1

Таблиця 1.13

Варіант використання «Редагувати ділянку»

Контекст використання	Редагування файлу
Дійові особи	Адміністратор
Передумова	Авторизація
Тригер	Немає
Сценарій	1. Запустити програму 2. Ввести логін та пароль 3. Натиснути “Ввійти” 4. Відкрити пункт меню “Редагувати ділянку” 5. Натиснути клавішу “Завантажити файл” 6. Змінити значення полів 7. Натиснути клавішу “Зберегти зміни”
Постумова	Рівень доступу повинен бути >2

Таблиця 1.14

Варіант використання «Додати ділянку»

Контекст використання	Формування файлу, збереження в базі даних
Дійові особи	Адміністратор, користувач
Передумова	Авторизація
Тригер	Немає
Сценарій	1. Запустити програму 2. Ввести логін та пароль 3. Натиснути “Ввійти” 4. Відкрити пункт меню “Додати ділянку” 5. Ввести значення в поля 6. Натиснути клавішу “Записати в базу даних” 7. Натиснути клавішу “Створити файл” 8. Ввести ім’я файлу 9. Натиснути клавішу “Зберегти”

Постумова	Рівень доступу повинен бути >2

Таблиця 1.15

Варіант використання «Показати на карті»

Контекст використання	Показати на карті
Дійові особи	Адміністратор, користувач
Передумова	Авторизація
Тригер	Немає
Сценарій	1. Запустити програму 2. Ввести логін та пароль 3. Натиснути “Ввійти” 4. Відкрити пункт меню “Показати на карті” 5. Ввести значення в поля: X,Y 6. Натиснути клавішу “Показати”
Постумова	Рівень доступу повинен бути >2

Далі необхідно визначити, які вимоги системи є функціональними. Розглянемо їх детальніше. Специфікація даних вимог наведена у таблицях 1.16.-1.17.

Таблиця 1.16

Специфікація функціональних вимог

Ідентифікатори вимоги	Вимоги	Атрибут вимог	
		Пріоритет	Виконавець
1	Авторизація	Обов’язковий	Адміністратор Користувач
2	Реєстрація	Обов’язковий	Адміністратор
3	Перегляд бази даних	Обов’язковий	Адміністратор
4	Редагування бази даних	Обов’язковий	Адміністратор

5	Редагування ділянки	Обов’язковий	Адміністратор
6	Додавання ділянки	Обов’язковий	Адміністратор Користувач
7	Пошук по карті	Не обов’язковий	Користувач

Таблиця 1.17

Специфікація нефункціональних вимог

Ідентифікатор вимоги	Вимоги	Атрибути вимог	
		Пріоритет	Виконавець
1	Застосованість нового ПЗ відносно аналогів	Рекомендований	Адміністратор
2	Вимоги до відповідальності стандартам	Обов’язковий	Адміністратор Користувач
3	Точність роботи	Обов’язковий	Адміністратор Користувач
4	Час безвідмовної роботи	Обов’язковий	Адміністратор
5	Вимоги до технології проектування	Обов’язковий	Адміністратор
6	Доступність	Рекомендований	Адміністратор Користувач

Висновки до першого розділу:

1. Визначено специфікацію вимог до програмної системи паспортизації земельних ділянок.
2. Проведено порівняльну характеристику аналогів, визначено недоліки існуючих програмних продуктів.
3. Розроблено діаграми варіантів використання для системи.

РОЗДІЛ 2 ПРОЕКТУВАННЯ

2.1 Розробка архітектури програмної системи

Архітектура програмного забезпечення відіграє важливу роль у ефективності, зручності, безпеці та експлуатації. При розробці програмного продукту не можна ігнорувати дані критерії. Тому оптимальним варіантом архітектури в даному випадку буде «клієнт-сервер».

Архітектура “клієнт-сервер” цікава й актуальна, бо забезпечує просте й відносно дешеве рішення проблеми колективного доступу до баз даних у локальній мережі.

Використання архітектури “клієнт-сервер” можливе у випадку, коли при розробці програми необхідно постійно звертатися до даних у локальній мережі або до файлового сервера.

JDBC (Java DataBase Connectivity) - набір класів і методів, які використовуються у мові програмування Java для роботи з базами даних. JDBC забезпечує прості, універсальні й добре адаптовані засоби взаємодії з різними СУБД.

Інтерфейси JDBC, розроблені корпорацією Sun, забезпечують виконання всіх стандартних операцій з базами даних SQL, а розробники MySQL надають конкретну реалізацію цих інтерфейсів. Реалізація робить всю взаємодію з базою даних: підключення, реєстрацію, виклик збережених процедур і т.д. Інтерфейси спроектовані таким чином, що програма, що використовує JDBC, може підключитися до будь-якої JDBC-сумісної бази даних без модифікації коду. Втім, при цьому все-таки необхідно враховувати деякі обставини.

По-перше, JDBC не виконує лексичного аналізу або перевірки синтаксису SQL на стороні клієнта. Команди просто передаються базі даних, незалежно від їхньої правильності. Таким чином, якщо код SQL працює в одній СУБД, але не підходить для іншої, проблем з реалізацію не буде до моменту фактичного встановлення з'єднання й пересилання коду SQL.

По-друге, у реалізацію включаються додаткові класи, специфічні для конкретного продукту. Наприклад, в MySQL є розширення для геометричних типів даних. Вони існують тільки в MySQL і не підтримуються іншими фірмами. Якщо використати ці спеціалізовані класи, програма не буде працювати в інших JDBC-сумісних базах даних.

Одна з переваг драйвера JDBC для MySQL полягає в тому, що він є драйвером «четвертого типу». Це означає, що він написаний на «чистій» мові Java, що дозволяє перенести його куди завгодно й використати на будь-якій платформі з підтримкою TCP/IP, оскільки драйвер підключається тільки через TCP/IP.

Діаграму розміщення для системи зображено на рисунку 2.1.

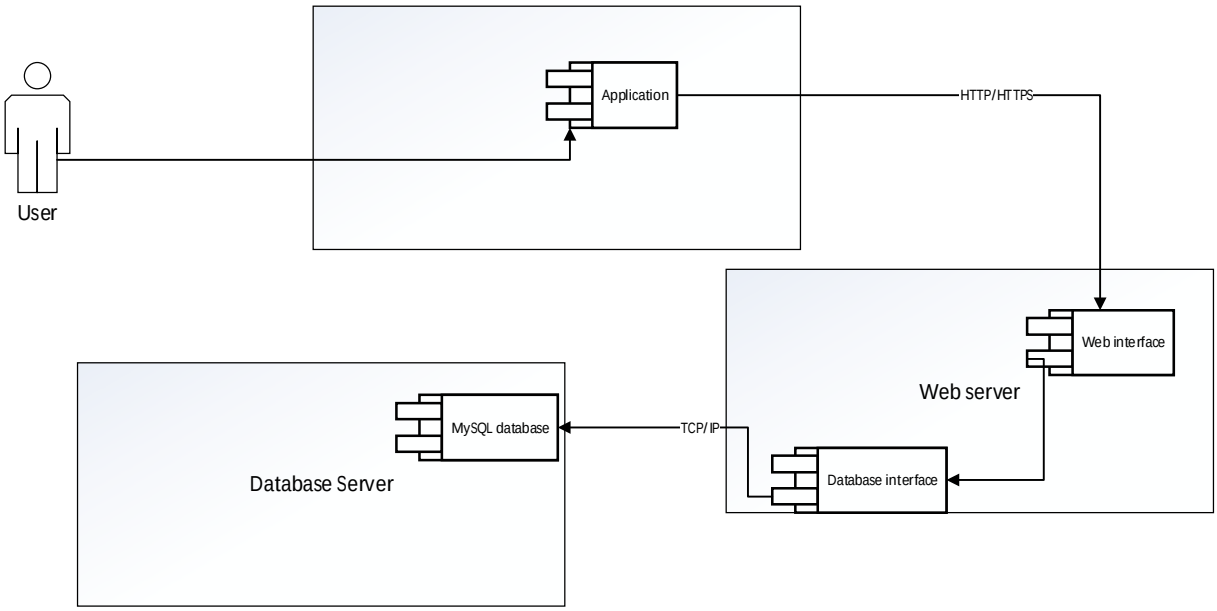


Рисунок 2.1 – Діаграма розміщення

На рисунку 2.1 зображено три компоненти такі як Worksation,Web server, Database Server. Кожен з цих компонентів реалізує інтерфейс для роботи з об’єктами. Зображено, що користувач динамічно працює з Application протоколу HTTP чи захищеного протоколу HTTPS, додаток зв’язується із веб-інтерфейсом, який в свою чергу ініціалізує роботу із Database interface, що за допомогою протоколів TCP/IP передає дані безпосередньо в database MySQL.

Наступним етапом необхідно розробити діаграму станів, на якій буде зображено поведінку системи.

Діаграма станів зображена на рисунку 2.2

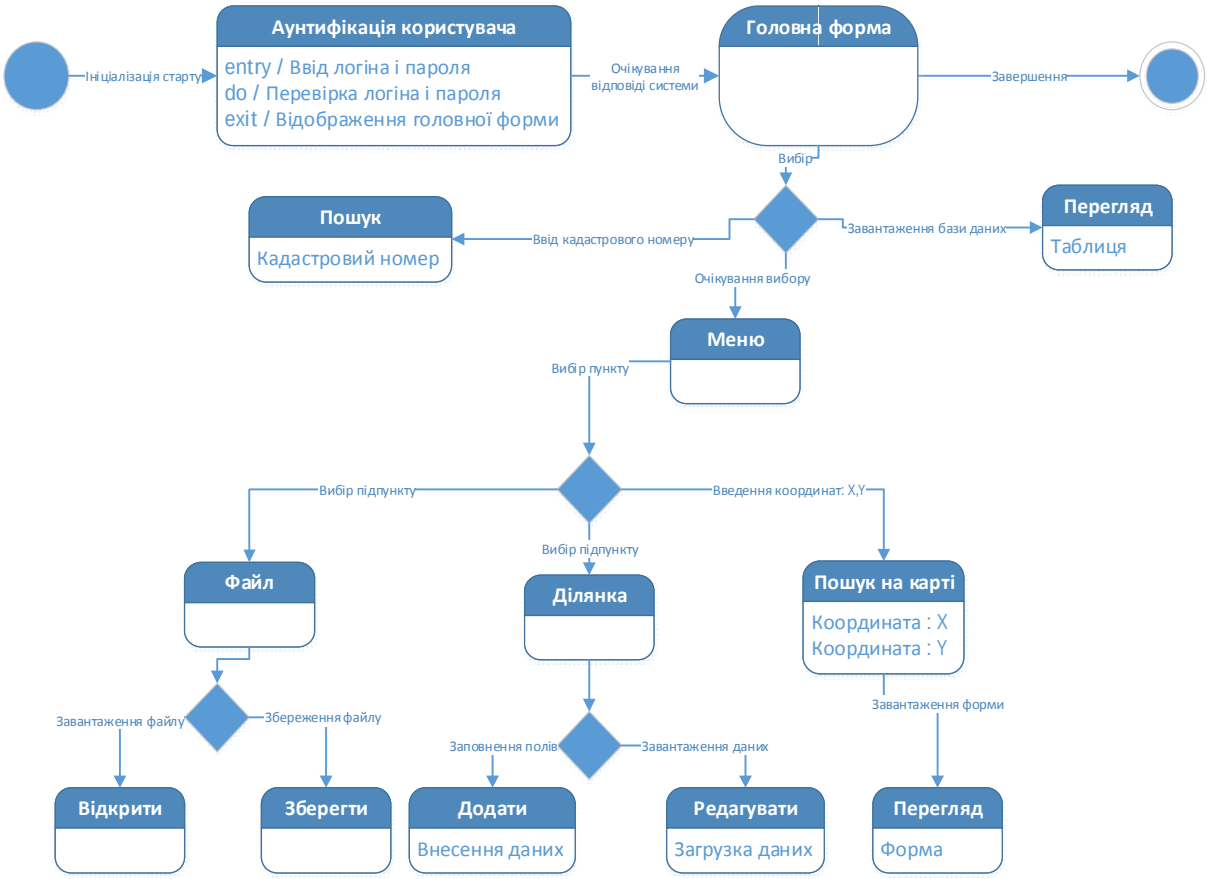


Рисунок 2.2 – Діаграма станів системи

Діаграма класів системи зображена на рисунку 2.3

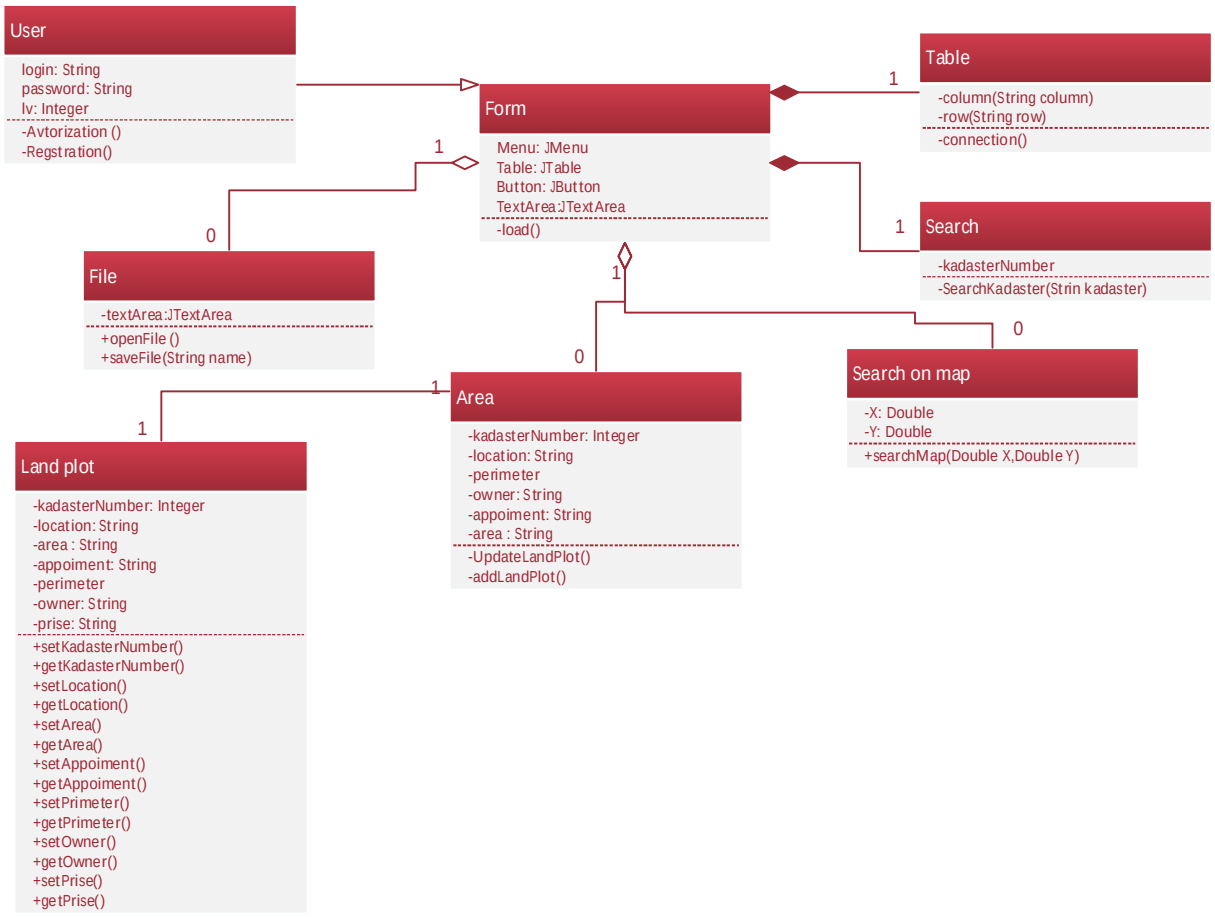


Рисунок 2.3 – Діаграма класів

На рисунку 2.3 зображено діаграму класів. Кожен клас з діаграми – фунціонально незмінна одиниця системи. Наприклад, клас Form містить змінні Menu, Table,JButton, TexArea і функцію load(). Функція load() завантажує елементи в об’єкт TextArea. Клас Search приймає на вхід змінну kadasterNumber, яка є входом функції SearchKadaster(). Дана функція повертає діалогове вікно. Класи LandPlot і Area зв’язані асоціативним зв’язком. Клас Area використовує значення, отримані від функцій LandPlot і застосовує у власних функціях: updateLandPlot() і addLandPloat().

2.2 Проектування структури бази даних

Розробка структури бази даних – великий і необхідний етап створення якісного програмного продукту, який б швидко і безпечно працював з даними. Оскільки архітектура системи «клієнт-сервер», то необхідно визначити будову бази даних, її таблиці і зв'язки між ними.

Управління даних забезпечує система управління базами даних (СУБД) MySQL v 5.5, оскільки дане програмне забезпечення є безкоштовним і популярним.

Діаграма бази даних, яка описує роботу таблиць coordinates, geodesy, location, owner, placing зображена на рисунку 2.4.

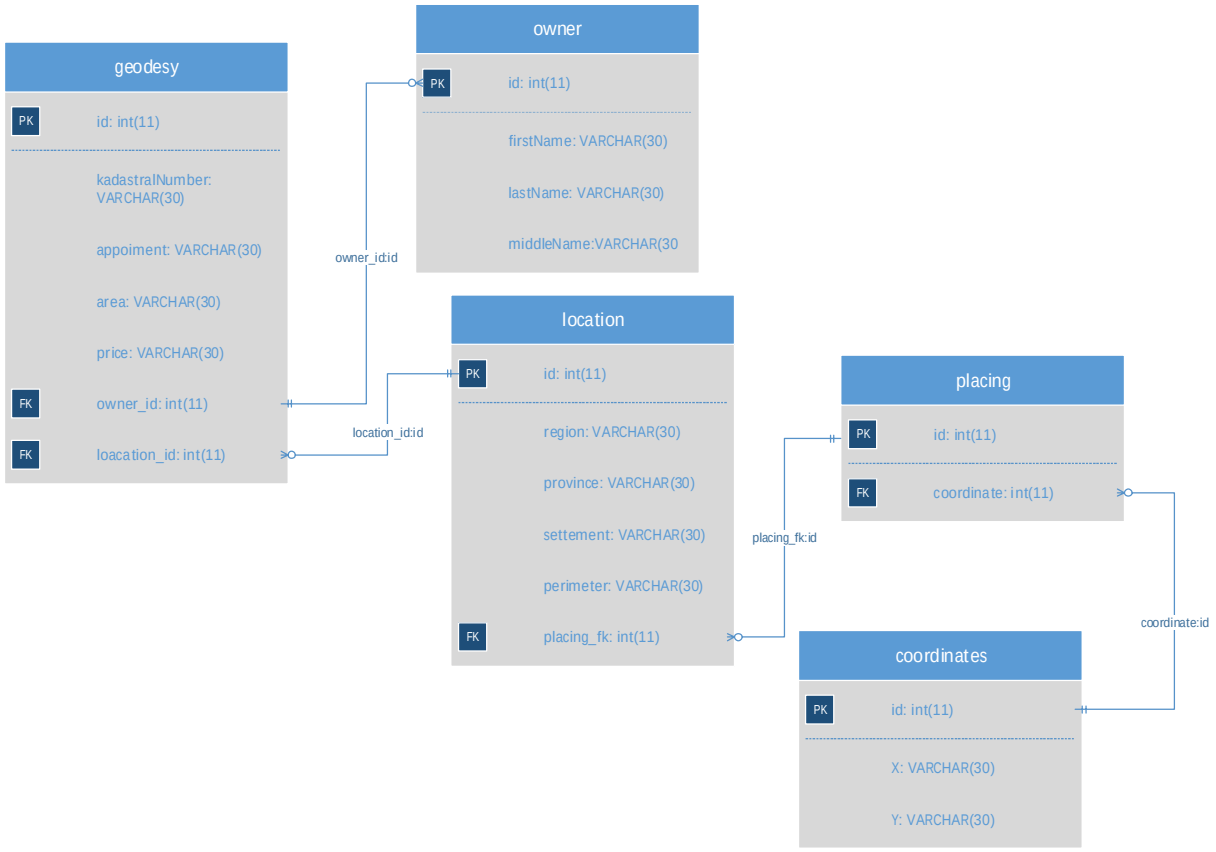


Рисунок 2.4 – ER діаграма

На основі діаграми з рисунку 2.4 можна сказати, що таблиця geodesy містить в собі дані з таблиць owner і location за допомогою поля owner_id, яке посилається на таблицю owner із унікальним ідентифікатором id. В свою чергу, поле location_id вказує на поле id в таблиці location, що містить поле placing_fk, що бере своє значення з таблиці coordination, які зв'язані проміжною таблиці placing. Завдяки цій структурі збільшується швидкість обробки даних і зменшуться навантаження на сервер.

Проектування бази даних слід розпочинати зі створення DFD процесу створення документа права власності на земельну ділянку (див. рис. 2.5).

Рисунок 2.5 – DFD процесу створення паспорта земельної ділянки

На рисунку 2.5 показано взаємодію основних функцій системи, що проектуються.

Далі на рисунку 2.6 зображена діаграма елементів і зв'язків.

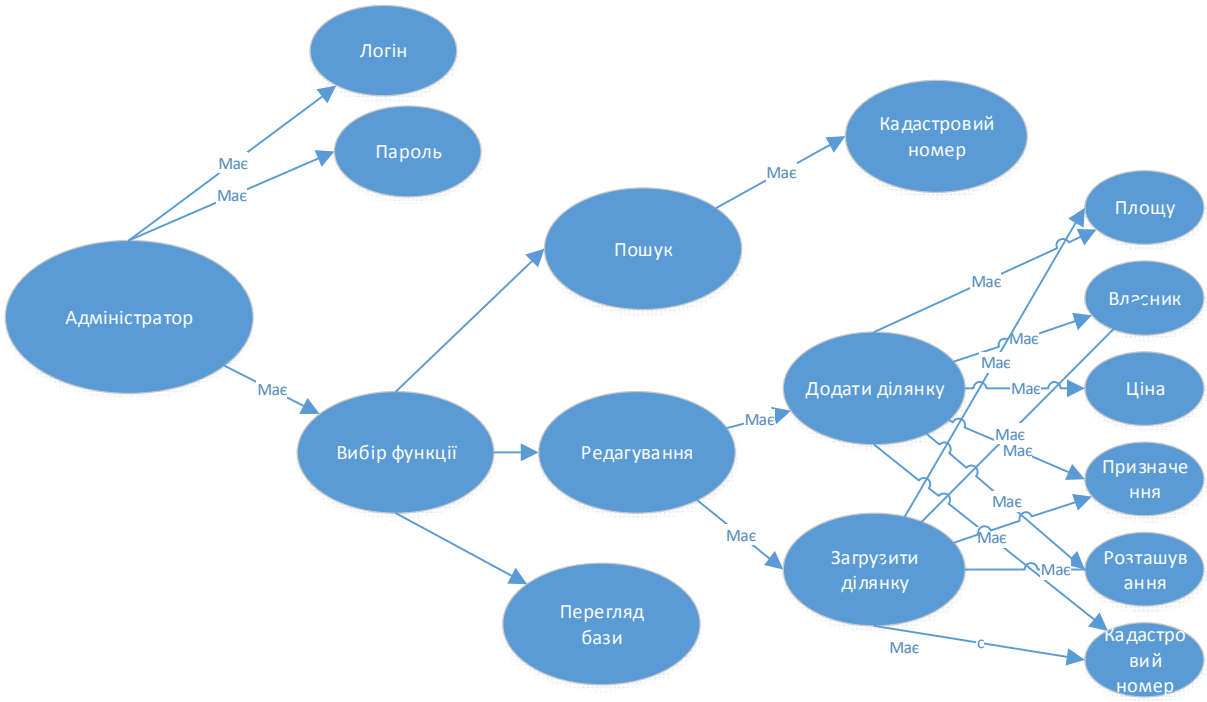


Рисунок 2.6 – Діаграма елементів і зв’язків

У таблиці 2.1наведено ідентифікатори бази даних.

Таблиця 2.1.

Об’єкт	Поле	Тип	Розмірність
Geodesy	Id cadastralNumber owner_id appoiment areas location_id price	int VARCHAR int VARCHAR VARCHAR int VARCHAR	11 30 11 30 30 11 30
Owner	Id firstName lastName middleName	int VARCHAR VARCHAR VARCHAR	30 30 30 30
Location	Id Region Province Settment Placing_id Perimeter	Int VARCHAR VARCHAR VARCHAR VARCHAR VARCHAR	11 30 30 30 30 30
Placing	Id	Int	11

	Coordinates	VARCHAR	30
Coordinates	Id X Y	Int VARCHAR VARCHAR	11 30 30

Висновки до другого розділу:

1. Обгрунтовано вибір архітектури, розроблено діаграму розміщення та діаграму станів системи.
2. Спроектовано структуру бази даних системи, створено ER діаграму та діаграму елементів і зв’язків.

РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Програмна реалізація проекту

Для програмної реалізації системи паспортизації земельних ділянок було обрано мову програмування Java, середовище IntellijIdea v15.

Перевагою Java є те, що вона інтерпретується, а не компілюється. Тим самим полегшується робота з різноманітними платформами. Достатньо створити Java систему після чого Java-програми можуть виконуватись в даному середовищі без додаткової компіляції на цій платформі. Проте, Java не є інтерпретованою мовою в частковому розумінні. Програма на Java компілюється. Результатом роботи компілятора Java є байт код. Байт код - це оптимізований набір команд, призначений для виконання уявним пристроєм - віртуальною Java-машиною. У такий спосіб витрати на інтерпретацію зводяться до мінімуму, оскільки байт код вже є оптимізованим, і досягається достатньо висока продуктивність Java-програм.

Завдяки C++ Java унаслідувала потужний механізм об’єктно-орієнтованого програмування. В результаті цього був сформований прагматичний підхід до об’єктів.

Організацію з користувачем організовано за допомогою бібліотеки Swing, яка вбудована в Java в образі діалогових вікон, що викликаються за допомогою класу JFrame.

Swing дає можливість працювати і під Windows, і під Linux, і на будь-якій іншій платформі, завдяки API (Application Program Interface).

Переваги Swing в порівнянні AWT:

- багатий набір інтерфейсних об’єктів;
- налаштовується зовнішній вигляд на різних платформах;

- роздільна архітектура модель-вид;
- вбудована підтримка HTML.

У таблиці 3.1 подано вимоги до апаратного забезпечення.

Таблиця 3.1

Апаратні вимоги	
Параметр	Значення
Процесор	Intel Core i3
Тактова частота процесора	1GHz;
Об’єм оперативної пам’яті	1024 Mb.
Жорсткий диск	20 G
Дисковод для компакт-дисків,	Наявний
JDK	1.7
Операційна система	
Windows, Linux, OS X	

Робота з програмним забезпеченням розпочинається із виклику екземпляру конструктора під назвою StartWindow() в методі public static void main, яка ініціалізує вивід компонентів.

```
public static void main(String[] args) {
   new StartWindow();
}
```

Компоненти конструктора підвантажуються з класу public class StartWindow extends JFrame, який наслідує об’єкти JFrame. Результатом виклику буде вивід, зображений на рисунку 3.1.

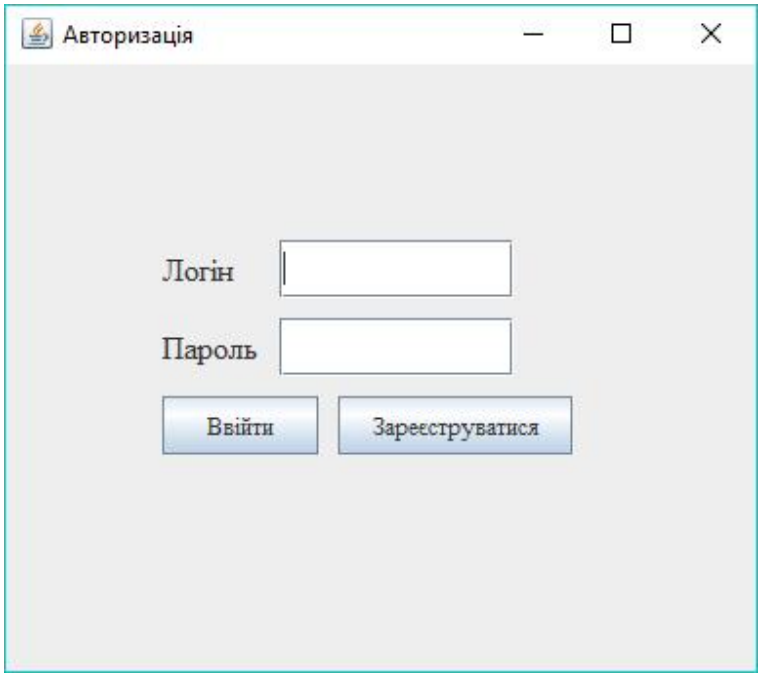


Рисунок 3.1 – Авторизація

На формі зображено два поля: “логін” і “пароль”, необхідні для авторизації користувача і два елемента JButton, один з яких - вхід, що викликає функцію зчитування значень полів “логін” і “пароль” і запис значень в базу даних user, зображену на рисунку 3.2

	id	login	password	lv
1	2	Oleh	oleh	4
2	3	Taras	Taras	1
3	4	Vadim	Vadim	2

Рисунок 3.2 – Таблиця user

Поля id, login, password, lv, зображені на рисунку 3.2, унікальні, оскільки поле id - унікальний атрибут користувача, поле логін відповідає значенню, яке ввів користувач в полі логін на рисунку 3.1, а поле password відповідає значенню, яке ввів користувач в полі пароль на рисунку 3.1. Поле lv вказує рівень доступу користувача по замовчуванню, він рівний одиниці.

Якщо поля ввели з дозволеними символами, перевіряють за допомогою функції filter(), яка наведена нижче:

```
public static boolean filter(String filterFix) {
   boolean b = true;
   char[] CharArray = null;
   CharArray = filterFix.toCharArray();
   char[] filterReg = {'1', '2', '3', '4', '5', '6', '7', '8', '9', '0', '|', ' '};
   for (int j = 0; j < CharArray.length; j++) {
      for (int i = 0; i < filterReg.length; i++) {
         if ((CharArray[j] == filterReg[i])) {
            b = false;
            break;}}}
   return b;}
```

Якщо заборонених символів не знайдено, виводиться повідомлення про успішну реєстрацію. Якщо заборонені символи присутні, то виводиться повідомлення про наявність заборонених символів.

При натисканні на елемент “Ввійти”, записи з полів відправляються на перевірку до бази даних з полями “логін” і “пароль”. Якщо еквівалентні збіги знайдені, то відбувається виклик конструктора Forma(), зображений на рисунку 3.3

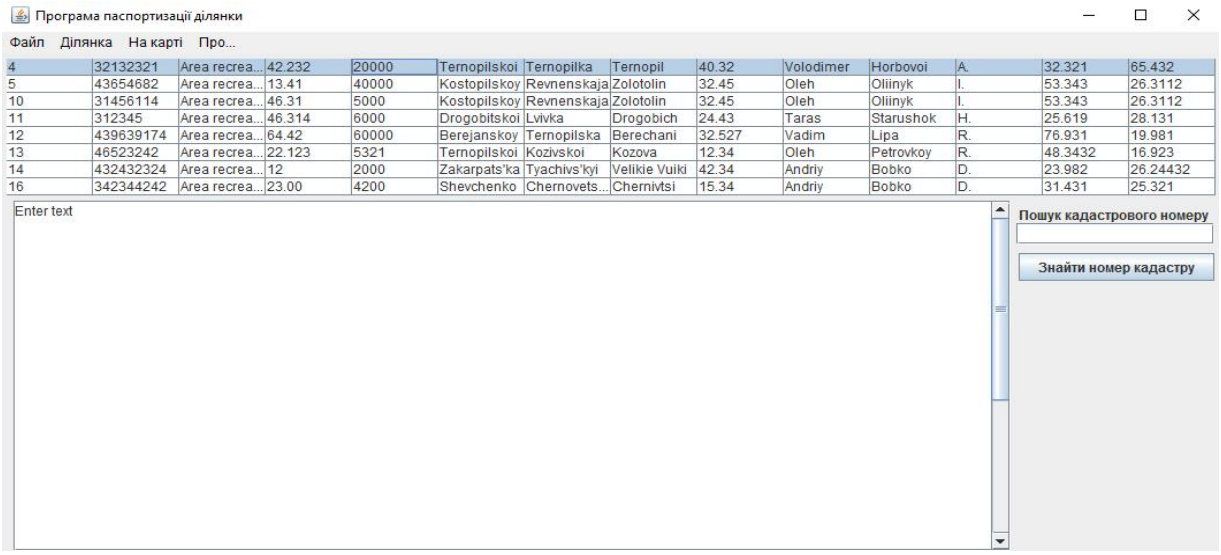


Рисунок 3.3 – Діалогове вікно Form

Якщо контактні дані невірні, то викликається діалогове вікно, зображене на рисунку 3.4

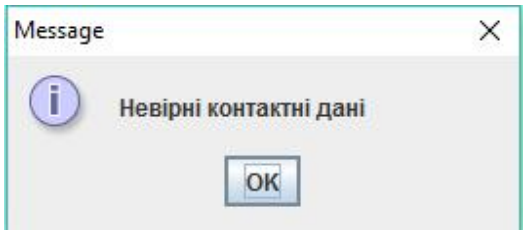


Рисунок 3.4 – Вікно повідомлення про некоректність введених даних

Вікно, зображене на рисунку 3.3, містить елементи JTable, JTextArea, MenuItem, JButton.

Завдяки елементу jButton можна здійснити пошук по базі даних по такому критерію як кадастровий номер. Результат виводу зображений на рисунку 3.5.

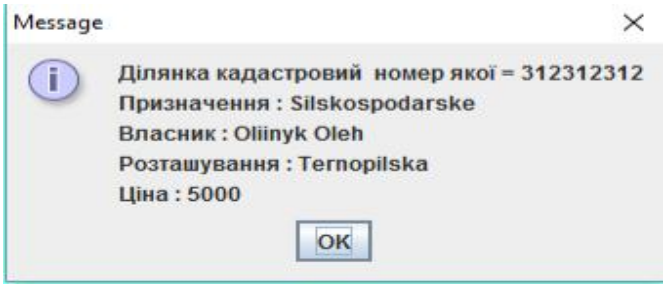


Рисунок 3.5 – Результат роботи функції пошуку

JTable елемент, що містить дані з таблиці geodesy з полями id, cadastralNumber, appoiment, owner_id, location_id, areas, price.

	id	cadastralNumber	owner_id	appoiment	areas	location_id	price
1	4	32132321	1	Area recreation appoiment	42.232	1	20000
2	5	43654682	2	Area recreation appoiment	13.41	2	40000
3	10	31456114	2	Area recreation appoiment	46.31	2	5000
4	11	312345	5	Area recreation appoiment	46.314	5	6000
5	12	439639174	6	Area recreation appoiment	64.42	6	60000
6	13	46523242	7	Area recreation appoiment	22.123	7	5321
7	14	432432324	8	Area recreation appoiment	12	8	2000
8	16	342344242	8	Area recreation appoiment	23.00	9	4200

Рисунок 3.6 – База даних geodesy

База даних ProgramPasportisation містить основні дані, які програма повинна обробляти.

JTextArea елемент текстового поля, який використовується для створення і редагування існуючих обмінних файлів типу .xml, як це показано на рисунку 3.6

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<UkrainianCadastralExchangeFile>
  <AdditionalPart>
    <ServiceInfo>
      <FileID>
        <FileDate>2015-03-04</FileDate>
        <FileGUID>4fa8af92-9f4a-4401-9376-de6305b07726</FileGUID>
      </FileID>
      <FormatVersion>0.7</FormatVersion>
      <ReceiverName></ReceiverName>
      <Software>Менеджер Обмінних Файлів</Software>
      <SoftwareVersion>1.01.0</SoftwareVersion>
```

Рисунок 3.7 – Текстове поле програми

JMenu елемент містить чотири основні пункти: “файл”, “ділянка”, “на карті” і “про”.

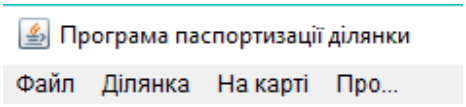


Рисунок 3.8 – Вигляд основних пунктів меню

Лістинг програмного коду для формування меню наведений нижче.

```
mb = new MenuBar();
Menu fileMenu = new Menu("Файл");
Menu areaMenu = new Menu("Ділянка");
Menu printMenu = new Menu("На карті");
Menu aboutMenu = new Menu("Про...");
mb.add(fileMenu);
mb.add(areaMenu);
mb.add(printMenu);
mb.add(aboutMenu);
```

Пункт файл містить підпункти: “відкрити файл”, “зберегти” і “вийти”. Відповідно до даних функція “відкриття файлу” дозволяє завантажити обраний файл в текстовий об’єкт text_area. Дана функція реалізована в класі FileOperation і лістинг функції open(), наведений нижче:

```
public void open() throws IOException {
        stream = FileTemplate.class.getResourceAsStream("6125285700010020343.xml");
        BufferedReader reader = new BufferedReader(new InputStreamReader(stream));
        String line = reader.readLine();
        text_area.append(line + "\n");
        while (line != null) {
            text_area.append(line + "\n");
        }
    }
```

Відповідно до функції “зберегти” дозволяє зберегти сформований файл з текстового об’єкту text_area у відповідному до потреб користувача форматі. Дана функція реалізована в класі FileOperation і міститься в функції saveAs(). Лістинг до даної функції наведений нижче:

```
public void saveAs() {
    JFileChooser chooser;
    chooser = new JFileChooser();
    int r = chooser.showSaveDialog(form);
    if (r == JFileChooser.APPROVE_OPTION) {
        file = chooser.getSelectedFile();
        try {
            marshaller.marshal(cadastralExchangeFile, file);
        } catch (Exception e1) {
            e1.printStackTrace();
        }
    }
}
```

Пункт меню ділянка містить підпункти такі як “додати ділянку” і “редагувати ділянку”. Реалізовані дані функції в класі Area.

Функція “додати ділянку” дозволяє користувачу відкрити зручну форму, яка дає можливість записати файл в базу даних. Реалізований даний метод в функції area().

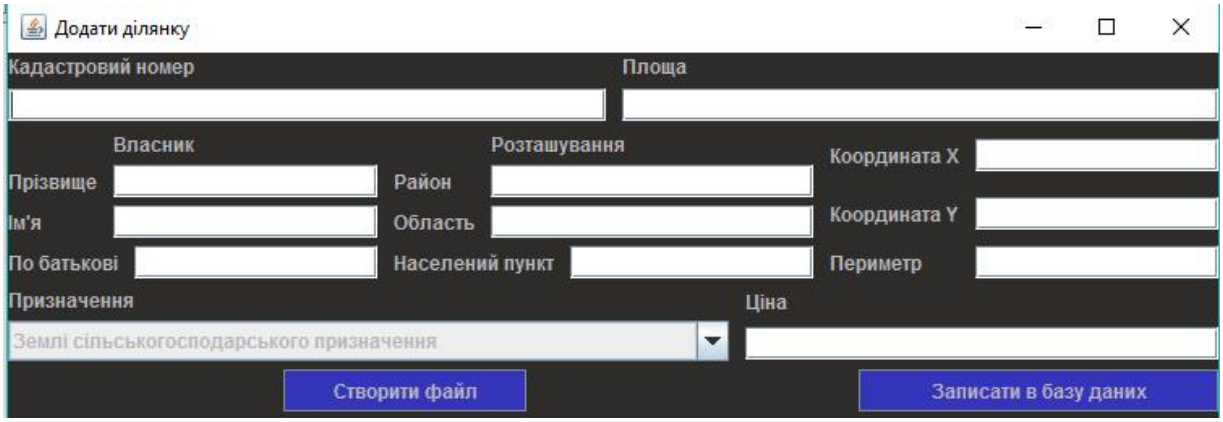


Рисунок 3.9 – Вікно додати ділянку

Функція “завантажити файл” дозволяє завантажити дані в форму. А функція “зберегти зміни” зчитує дані з форми і записує їх зворотньо в файл. Форма даного методу міститься в функції LoadAndUpdateFile і зображена на рисунку 3.10.

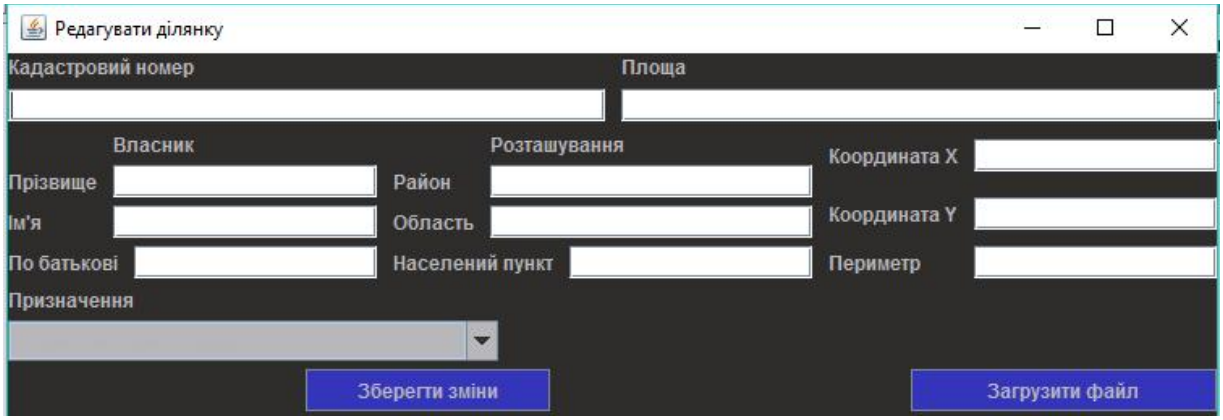


Рисунок 3.10 – Форма Редагувати ділянку.

Розділ меню “Показати на карті ” дозволяє викликати форму, в якій вводяться координати. Це розроблено з метою, щоб користувач зміг

пересвідчитися в достовірності розташування земельної ділянки. Вікно для вводу координат зображене на рисунку 3.11.

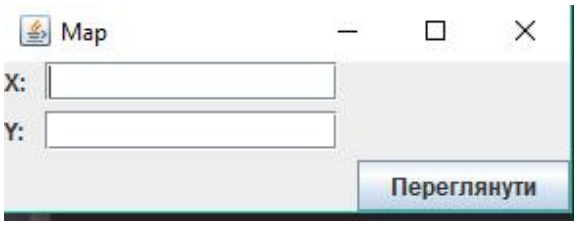


Рисунок 3.11 – Вікно вводу координат.

Результатом даної операції буде зображена на рисунку 3.12 форма.

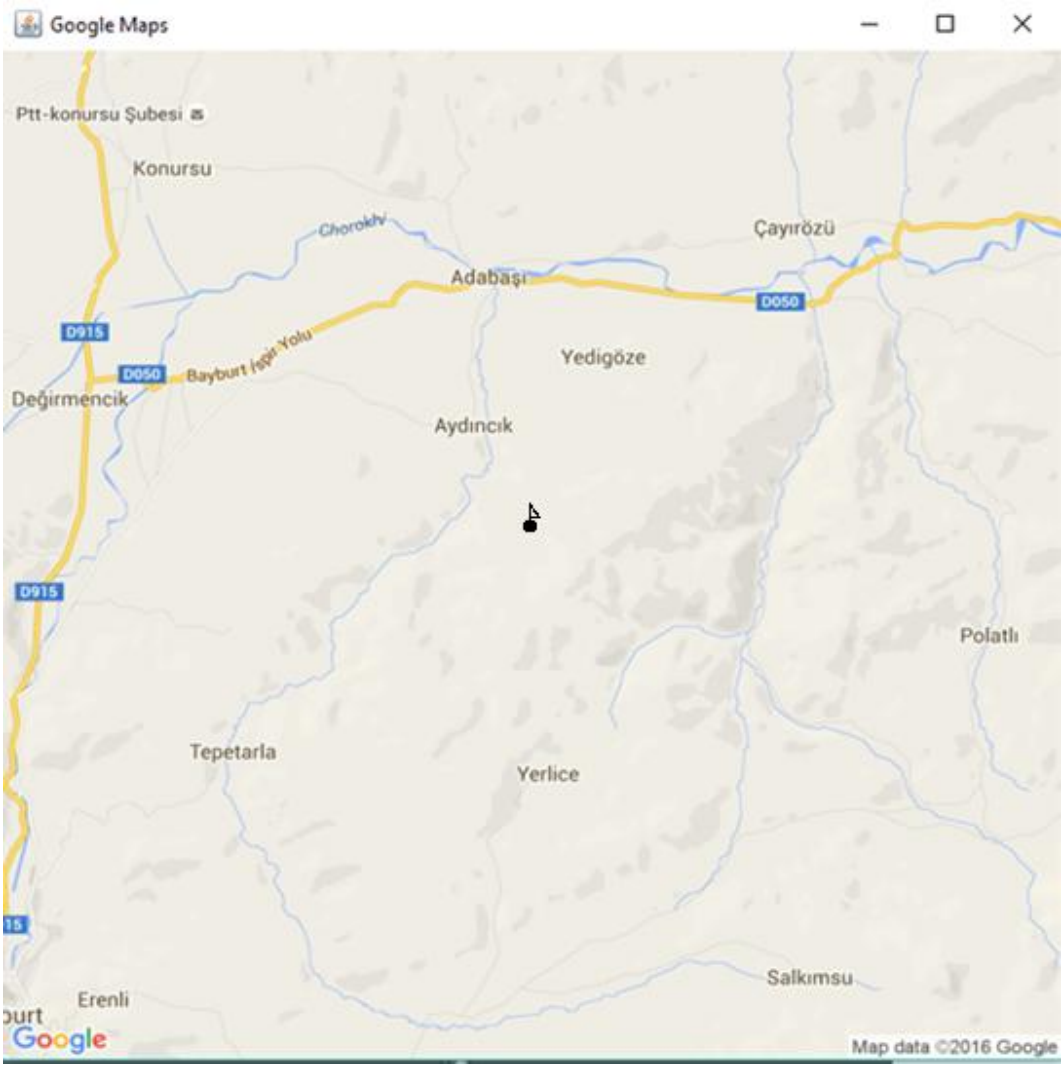


Рисунок 3.12 – Форма функції “Показати на карті”

Пункт “Про автора” - форма містить дані про розробника і контактні дані, завдяки яким можна зв’язатися з ним, а пункт “Про програму” - форма, на якій розміщені короткі теоретичні відомості безпосередньо про програму і її призначення.

Лістинг коду основних модулів системи подано у додатку А.

3.2 Програмна реалізація бази даних

Програмна реалізація бази даних додатку проводилася в СУБД MySQL за допомогою інтеграції в інструмент розробки IntellijIdea v.15. Основною причиною вибору цієї СУБД - це її доступність і надійність. SQL-сервер - програма, яка приймає запити, написані на SQL, і відсилає назад певні відповіді. Відповідями можуть бути: дані, кількість рядків, задіяних в запиті або просто рядок.

SQL - це мова структурованих запитів (Structured Query Language), міжнародний стандарт мови для доступу до баз даних.

Приклад SQL запиту створення таблиці geodesy наведений нижче:

```
CREATE TABLE `geodesy` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `cadastralNumber` longtext,
  `owner_id` int(11) DEFAULT NULL,
  `appoiment` varchar(70) DEFAULT NULL,
  `areas` mediumtext,
  `location_id` int(11) DEFAULT NULL,
  `price` double DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `geodesy_owner_id_index` (`owner_id`),
  KEY `geodesy_location_id_index` (`location_id`),
  CONSTRAINT `geodesy_location_id_fk` FOREIGN KEY (`location_id`) REFERENCES `location` (`id`),
  CONSTRAINT `geodesy_owner_id_fk` FOREIGN KEY (`owner_id`) REFERENCES `owner` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=17 DEFAULT CHARSET=latin1
```

Створення таблиці відбувається, завдяки вбудованому інстументу в середовищі IntelijIdea. Він надає графічну побудову бази даних. Приклад даного вікна наведено на рисунку 3.13.

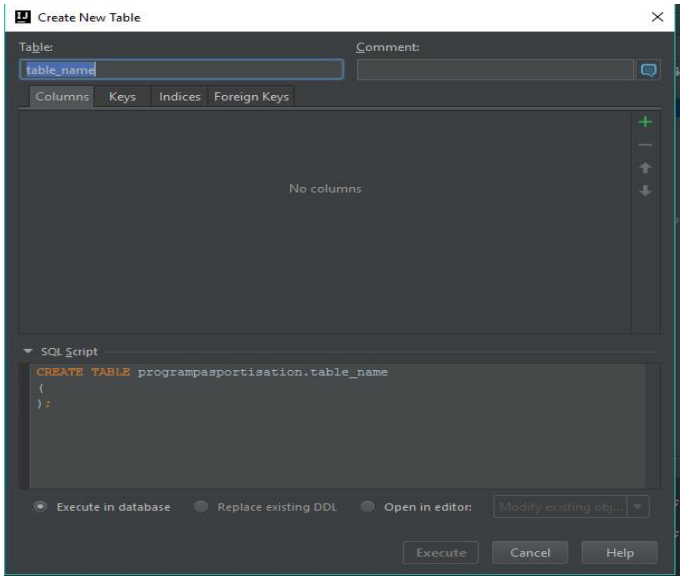


Рисунок 3.13 – Інструмент для створення таблиці.

Звичайно, нам потрібно якось забезпечувати доступ до MySQL. Це можна зробити через клієнт або через мову програмування Java.

Основні цілі MySQL: швидкодія і стійкість до помилок. Ця СУБД прекрасно справляється з обробкою великих масивів даних Крім того, MySQL в запитах у великих таблицях перевершує багато інших систем.

У поєднанні із середовищем розробки IntelliJ Idea зручний і ефективний засіб для роботи з проектами різного масштабу. Інтегрована панель роботи із СУБД зображена на рисунку 3.14.

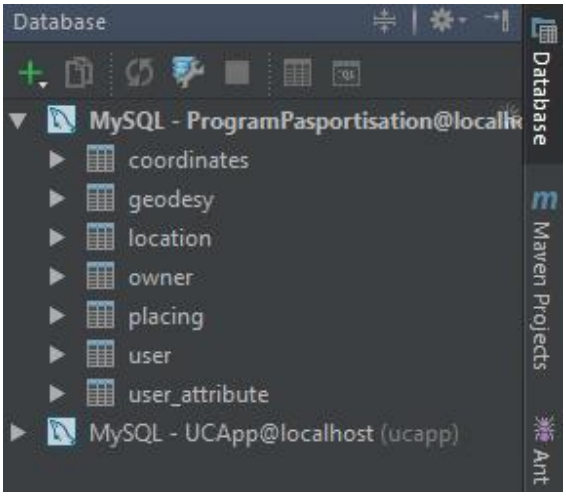


Рисунок 3.14 – Панель роботи із СУБД

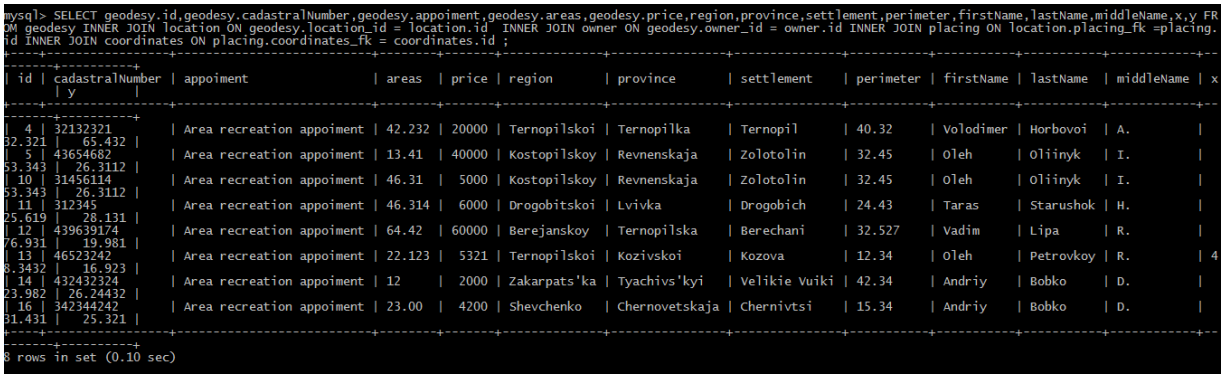
Дана панель дозволяє повністю працювати із СУБД MySQL. На даній панелі розміщена база даних під назвою ProgramPasportisation. Вона містить таблиці, які зберігають дані, що подає користувач.

З метою оптимізації роботи бази даних, дані розподілені між такими таблицями: coordinates, geodesy, location, owner, placing, user, user_attribute. Дані об’єднуються за допомогою SQL запиту.

Лістинг SQL зображений нижче:

```
String sql = "SELECT geodesy.id,geodesy.cadastralNumber,geodesy.appoiment,geodesy.areas,geodesy.price," +
    "region,province,settlement,perimeter,firstName,lastName,middleName,x,y " +
    "FROM geodesy INNER JOIN location " +
    "ON geodesy.location_id = location.id  INNER JOIN owner ON geodesy.owner_id = owner.id " +
    "INNER JOIN placing ON location.placing_fk =placing.id INNER JOIN coordinates " +
    "ON placing.coordinates_fk = coordinates.id";
```

Приклад роботи, який демонструє роботу в середовищі MySQL 5.5 Command Line Client зображено на рисунку 3.15.



```
mysql> SELECT geodesy.id,geodesy.cadastralNumber,geodesy.appoiment,geodesy.areas,geodesy.price,region,province,settlement,perimeter,firstName,lastName,middleName,x,y FR
OM geodesy INNER JOIN location ON geodesy.location_id = location.id  INNER JOIN owner ON geodesy.owner_id = owner.id INNER JOIN placing ON location.placing_fk =placing.
id INNER JOIN coordinates ON placing.coordinates_fk = coordinates.id ;
+----+---------------+--------------------------+-------+-------+-------------+---------------+--------------+-----------+-----------+-----------+-----------+--
------+----------+
| id | cadastralNumber | appoiment                | areas | price | region      | province      | settlement   | perimeter | firstName | lastName  | middleName | x
      | y        |
+----+---------------+--------------------------+-------+-------+-------------+---------------+--------------+-----------+-----------+-----------+-----------+--
------+----------+
|  4 | 32132321      | Area recreation appoiment | 42.232 | 20000 | Ternopilskoi | Ternopilka    | Ternopil     | 40.32     | Volodimer | Horbovoi  | A.         |
32.321 |   65.432 |
|  5 | 4365468       | Area recreation appoiment | 13.41  | 40000 | Kostopilskoy | Revnenskaja   | Zolotolin    | 32.45     | Oleh      | Oliinyk   | I.         |
53.343 |  26.3112 |
| 10 | 31456114      | Area recreation appoiment | 46.31  |  5000 | Kostopilskoy | Revnenskaja   | Zolotolin    | 32.45     | Oleh      | Oliinyk   | I.         |
53.343 |  26.3112 |
| 11 | 312345        | Area recreation appoiment | 46.314 |  6000 | Drogobitskoi | Lvivka        | Drogobich    | 24.43     | Taras     | Starushok | H.         |
25.619 |   28.131 |
| 12 | 439639174     | Area recreation appoiment | 64.42  | 60000 | Berejanskoy  | Ternopilska   | Berechani    | 32.527    | Vadim     | Lipa      | R.         |
76.931 |   19.981 |
| 13 | 46523242      | Area recreation appoiment | 22.123 |  5321 | Ternopilskoi | Kozivskoi     | Kozova       | 12.34     | Oleh      | Petrovkoy | R.         | 4
8.3432 |   16.923 |
| 14 | 432432324     | Area recreation appoiment | 12     |  2000 | Zakarpats'ka | Tyachivs'kyi  | Velikie Vuiki | 42.34     | Andriy    | Bobko     | D.         |
23.982 | 26.24432 |
| 16 | 34234242      | Area recreation appoiment | 23.00  |  4200 | Shevchenko   | Chernovetskaja | Chernivtsi   | 15.34     | Andriy    | Bobko     | D.         |
31.431 |   25.321 |
+----+---------------+--------------------------+-------+-------+-------------+---------------+--------------+-----------+-----------+-----------+-----------+--
------+----------+
8 rows in set (0.10 sec)
```

Рисунок 3.15 – Command Line Client

На рисунку 3.15 зображено виконання SQL запиту, який об’єднує поля з усіх таблиць бази даних ProgramPasportisation.

Висновки до третього розділу:

1. Реалізовано функціонал програмного продукту, інтерфейс зв'язку з користувачем і представлено його.
2. Реалізовано таблиці і зв'язки між ними за допомогою інтегрованого середовища управління СУБД у IntelliJ Idea v15.

РОЗДІЛ 4 ТЕСТУВАННЯ ТА ДОСЛІДНА ЕКСПЛУАТАЦІЯ

4.1 Тестування

Коли здійснюється вибір стратегії тестування важливо зробити оптимальний спосіб тестування програмної системи, зважаючи на особливості програмного продукту. Необхідно здійснити такі види тестування як модульне, інтеграційне і системне.

Тестування це необхідний етап розробки усіх проектів, оскільки він дозволяє зекономити багато часу, ресурсів, можливих втрат даних, перевірити розроблений програмний продукт на ефективність і доцільність.

Модульне – це тестування кожної функціональності системи, незалежно від програмного продукту, що розробляється.

Інтеграційне – це такий вид тестування, при якому комбінуються окремі модулі, об’єднуються і тоді тестуються разом.

Системне – це такий вид тестування, при якому тестуються як функціональні так і нефункціональні вимоги.

Таблиця 4.1

№	Дія	Очікуваний результат	Результат
1	Натискання кнопки “Зареєструватися”	Реєстрація користувача	True
2	Натискання кнопки “Ввійти”	Відчинення головної форми	True
3	Натискання кнопки “Знайти номер кадастру”	Вікно з результатом пошуку	True
4	Натискання кнопки “Створити файл”	Створення файлу	True
5	Натискання кнопки “Записати в базу даних”	Запис в базу даних	True

6	Натискання кнопки “Зберегти зміни”	Збереження в файл	True
7	Натискання кнопки “Загрузити файл”	Заванаження даних з файлу	True
8	Натискання кнопки “Відкрити”	Завантаження файлу	True
9	Натискання кнопки “Зберегти”	Збереження в файл	True
10	Натискання кнопки “Додати ділянку”	Запуск форми “Додати ділянку”	True
11	Натискання кнопки “Відкрити ділянку”	Запуск форми “Відкрити ділянку”	True
12	Натискання кнопки “Побачити на карті”	Відкриття полів для вводу	True
13	Натискання кнопки “Про автора”	Відкриття форми “Про автора”	True
14	Натискання кнопки “Про програму”	Відкриття форми “Про програму”	True

В таблиці 4.1 наведено test-case до варіантів можливих сценаріїв при роботі із програмою. В ній показано, що при усіх сценаріях роботи із програмою, завжди очікується позитивний результат, а отже, усі функції програми працюють і програма готова до застосування.

Тестування безпеки ‘розпочинається при старті програми, коли з’являється вікно “Авторизація”, що ускладнює можливість несанкціонованого доступу користувачів до конфіденційної інформації.

Такий захист забезпечує функція filter класу WordFilter. Дана функція перевіряє на наявність такі символи як '1', '2', '3', '4', '5', '6', '7', '8', '9', '0', '|', ' '. Якщо дані відсутні в полях “логін” і “пароль”, тоді програма дозволяє вхід. Також функція validate() перевіряє на кількість введених символів. Якщо кількість символів становить менше 4 символів програма видає помилку, що зображено на рисунку 4.1.

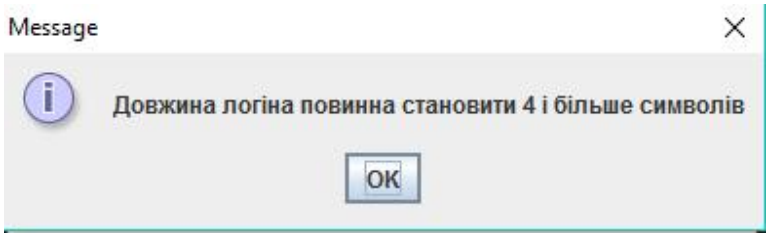


Рисунок 4.1 – Повідомлення про недостатню кількість символів в полі ”логін”

При перевірці в базі даних полів з кількістю символів більше 4 і відсутності заборонених символів відбувається пошук в базі даних UCApp в таблиці user користувача з введеними даними. Якщо збігів не знайдено програма видає вікно зображене на рисунку 4.2.

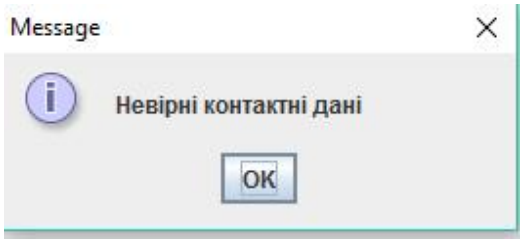


Рисунок 4.2 – Помилка вводу даних користувача.

4.2 Розгортання програмного продукту

Інструкція по встановленню JDK 8 наведена нижче.

1) JDK 8 можна скачати з офіційної сторінки http://www.oracle.com.

2) Далі потрібно вибрати розділ JDK і в ньому вибрати версію 8.

3) Після чого потрібно натиснути “Accept License Agreement”

3) Натиснути клавішу “Download”

На рисунку 4.3 зображено скріншот розділу JDK на сайті oracle.com.

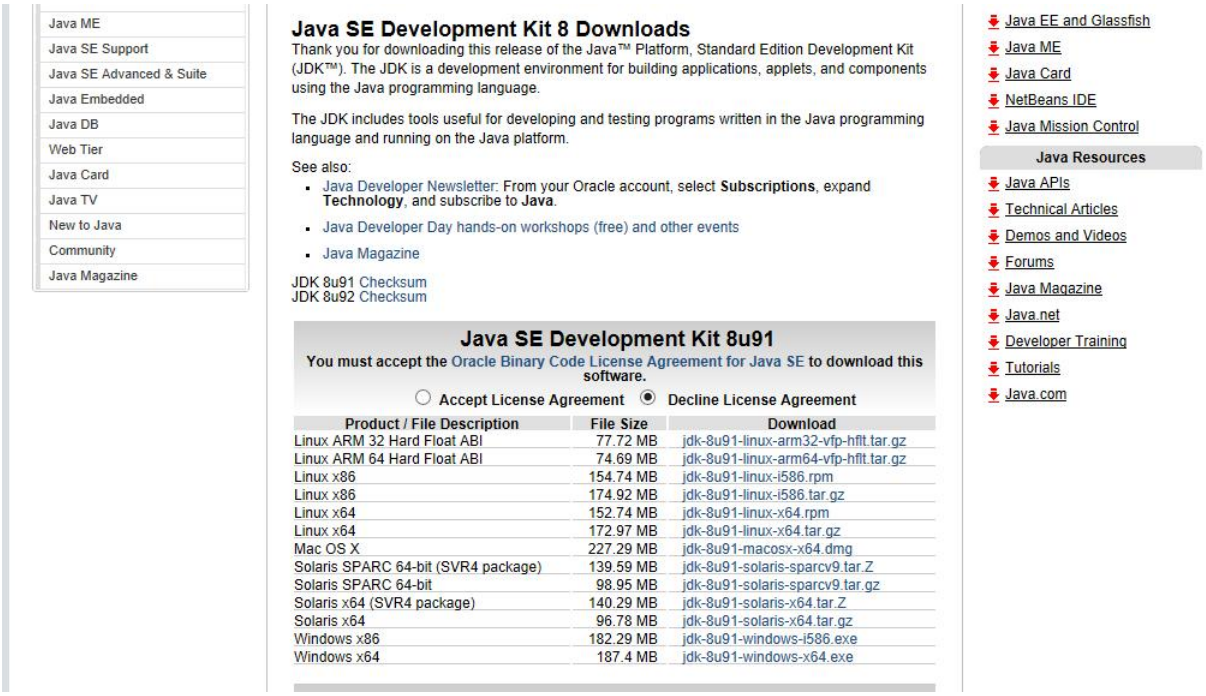


Рисунок 4.3 – Скріншот розділу JDK

При встановленні JDK 8 програма автоматично конфігуруватиметься із JAVA машиною і буде готова до повноцінного запуску.

Інструкція щодо встановлення СУБД MySQL наведена нижче:

1. СУБД MySQL з офіційної сторінки http://www.oracle.com.
2. Далі потрібно знайти і вибрати розділ MySQL.
3. Наступним кроком необхідно натиснути на силку Downloads
4. Тоді необхідно обрати версію СУБД Enterprise Edition
5. Потім натиснути Trial Download

На рисунку 4.4 зображено скріншот розділу MySQL на сайті oracle.com.

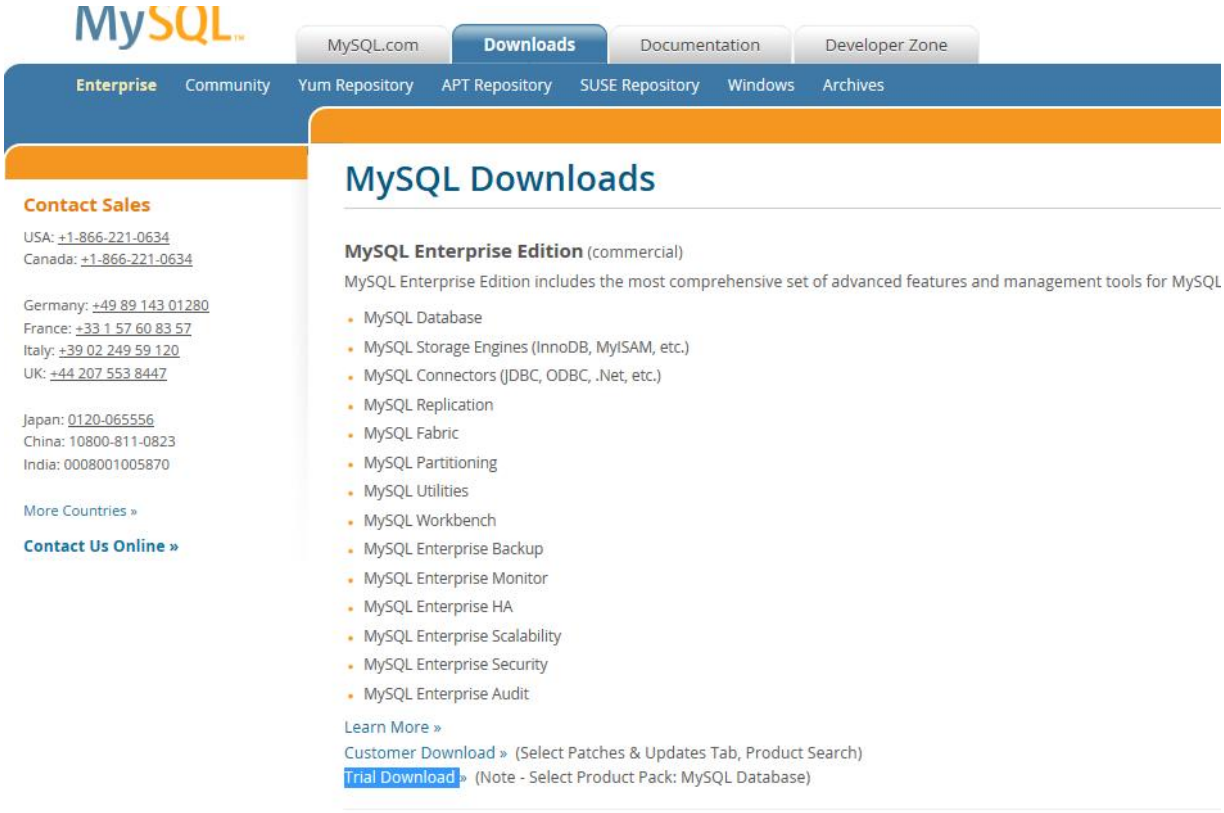


Рисунок 4.4 – Скріншот розділу MySQL Enterprise Edition

MySQL необхідна для збереження даних з метою подальшого їх використання.

4.3 Інструкція користувача

Інструкція користувача – це документ, який містить інформацію про встановлення і використання програмного продукту.

Компоненти ПЗ. Дане програмне забезпечення розроблялося за допомогою мови програмування Java version 8, відлагодження відбувається за допомогою віртуальної машини JAVA, яка викликається у середовищі розробки IntelliJ Idea v15. Завдяки цьому програма може експлуатуватися в операційних системах як OS, Windows і Linux.

Також було застосовано об’єктно-орієнтовану парадигму програмування що дозволило позбутися надлишковості і повторювання коду, збільшивши таким чином швидкість обробки інформації. Для повноцінної роботи системи необхідно процесор з частотою не менше 800 MHz, оперативною пам’яттю 512 Mb. Для експлуатації програми необхідно встановлення JDK 8. Цим самим програма може інтерпретуватися під обрану операційну систему.

Програмний продукт призначений для фірм і юридичних осіб, які займаються видачею паспорта на земельну ділянку для купівлі-продажу.

Робота з програмним продуктом розпочинається із запуску вікна авторизації, але для того, щоб користувач зміг авторизуватися необхідно спочатку зареєструватися. Для реєстрації користувача необхідно заповнити поля “логін”,”пароль” і натиснути клавішу “Зареєструватися”. Після чого з’явится вікно, зображене на рисунку 4.5.

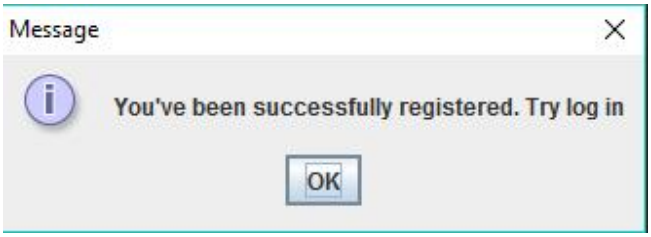


Рисунок 4.5 – Повідомлення про успішну реєстрацію

Після чого необхідно ввести зареєстровані дані і натиснути клавішу “Ввійти”. В результаті чого запуститься головна форма програми.

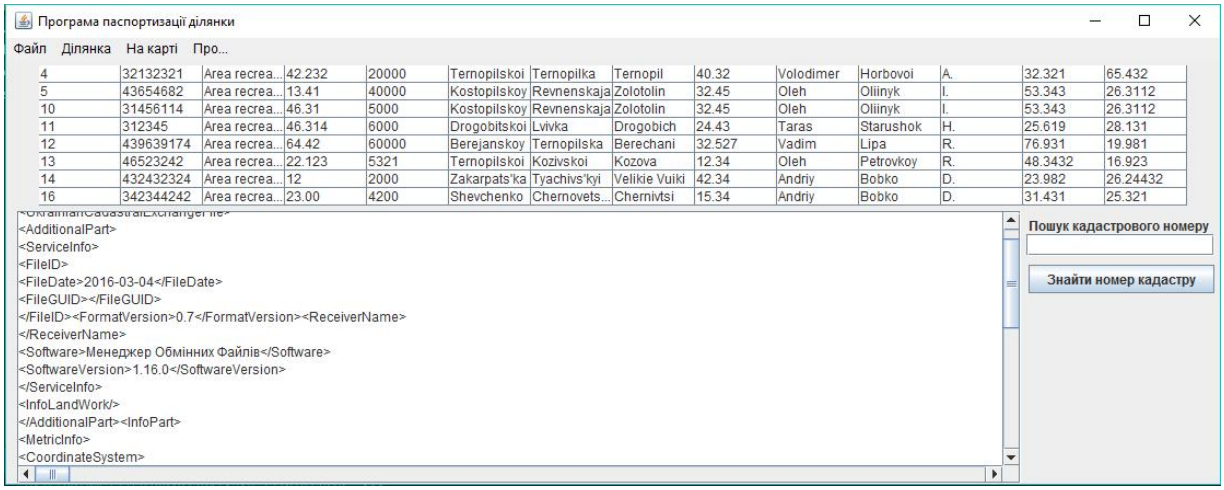


Рисунок 4.6 – Головне вікно програми

Після запуску форми на панелі меню містить такі пункти як “Файл”, ”Ділянка”, ”На карті”, “Про”.

Головне вікно містить такі компоненти: таблиця - містить дані із бази даних; поле – призначене для завантаження файлів в поле і редагування їх, або написанні файлу з метою подальшого збереження; кнопка “Знайти номер кадастру” зчитує значення з поля “Пошук кадастрового номеру” і перевіряє чи існують дані в базі даних. Якщо пошук вдалий відкривається вікно з даними, які відповідають кадастровому номеру. Якщо дані відсутні відкривається вікно зображене на рисунку 4.7.

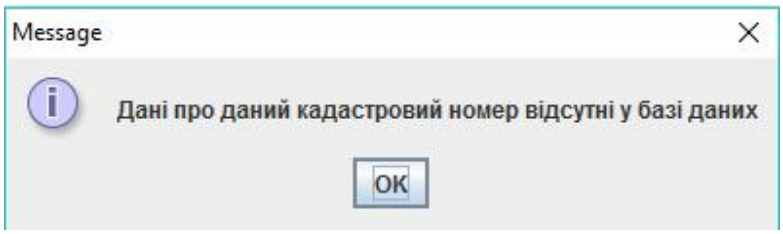


Рисунок 4.7 – Повідомлення про відсутність кадастрового номеру

Пункт меню “Файл” містить підпункти “Відкрити”, “Зберегти” і “Вихід”. Зображені на рисунку 4.8.

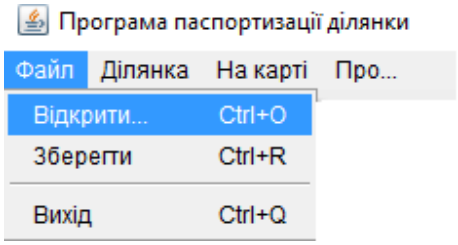


Рисунок 4.8 – Пункту “Файл”

Підпункт “Відкрити” дозволяє викликати діалогове вікно “Open”. Яке дозволяє завантажити файл в поле форми TextArea.

Підпункт “Зберегти” дозволяє викликати діалогове вікно “Save”. Яке дозволяє зберегти вміст поля TextArea в файл.

Підпункт “Вихід” призначений для зупинки роботи програми. Натисунувши на цей підпункт викликається вікно зображене на рисунку 4.8.

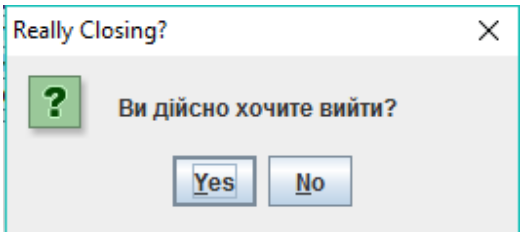


Рисунок 4.9 – Вікно “Вихід”

При натисканні на клавішу “Yes” програма припинить роботу. А при натисканні на клавішу “No” програма продовжить працювати.

Пункт меню “Ділянка” містить підпункти “Додати ділянку”, “Відкрити ділянку”, зображені на рисунку 4.10.

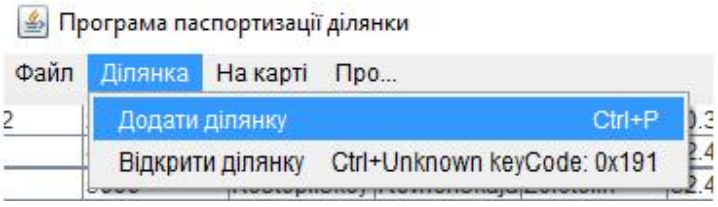
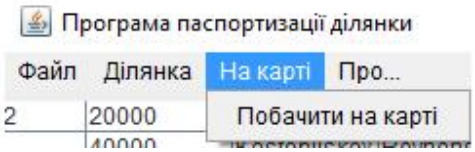


Рисунок 4.10 – Пункт “Ділянка”

Підпункт “Додати ділянку” відкриває вікно з полями, які небхідно заповнити після чого натиснути клавішу “Створити файл”, “Записати в базу даних”. Після даних операцій формується обмінний файл і дані записуються у базу даних.

Підпункт “Відкрити ділянку” відкриває вікно з полями, які заповняються при натисканні на клавішу “Загрузити файл” і обрати файл після чого поля заповняться даними з файлу, а при натисканні на клавішу “Зберегти зміни” відбувається перезаписування існуючих даних у файлі та базі даних, а якщо вони не існували, то відбувається запис їх в базу даних.

Пункт меню “На карті” містить підпункти “Побачити на карті”, зображені на рисунку 4.11.



4.11 – Пункт “На карті”

Підпункт “Побачити на карті” відкриває вікно з полями: X, Y. При заповненні полів необхідно натиснути на клавішу “Переглянути” запуститься карта. На карті червоною точкою відзначено місце розташування запиту.

Пункт меню “Про ” містить підпункти “Про автора”, “Про програму” і “Вихід”, зображені на рисунку 4.12.

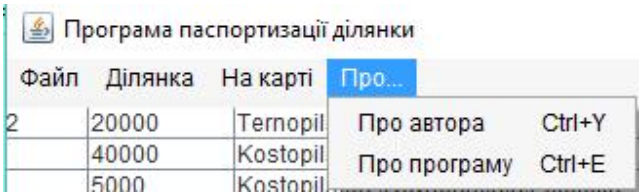


Рисунок 4.12 – Пункт “Про”

У підпункті “Про автора” відкривається форма, на якій зображено дані про автора і його контакти (рис. 4.13).

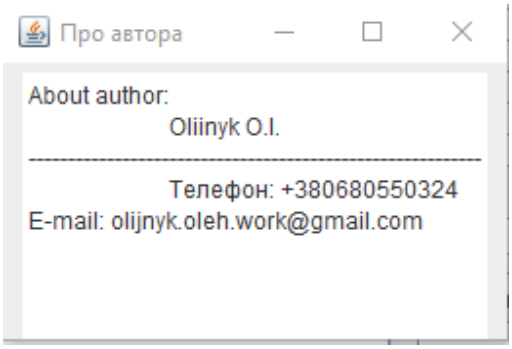


Рисунок 4.13 – Вікно “Про автора”

Висновки до четвертого розділу:

1. Протестовано програму паспортизації земельної ділянки та отримано дані про повну функціональність системи.
2. Розроблена інструкція користувача з описом усіх функцій системи і інструкцією для роботи із програмним продуктом.

ВИСНОВКИ

В результаті виконання дипломної роботи розроблено програмний продукт, який дозволяє автоматизувати процес паспортизації земельних ділянок. Результатом роботи такої програми є формування файлу типу xml, який використовують при купівлі-продажі об’єктів правових маніпуляцій із земельними ресурсами.

В процесі дослідження виникла потреба у зручному редагуванні таких файлів у випадку, наприклад, можливої зміни власника чи інших обставин. Також вдалося досягти значного прогресу у роботі із xml файлами за допомогою мови програмування Java.

Розробляючи даний програмний продукт:

На першому етапі розробки проекту було досліджено проблеми даної галузі.

Другим етапом проекту було проектування архітектури, яка б забезпечувала потреби системи і проектування сховища даних.

Третім етапом було розробка програмного продукту його інтерфейсу і розробка безпосередньо сховища даних.

Четвертим етапом було зроблено тестування програмного продукту і побудована інструкції для користувача.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бенджамин Эванс/ Мартин Вербург. Java. Новое поколение разработки. 2014 р Питер.
2. Инженерная геодезия / [Багратуни Г. В., Ганьшин В. Н., Данилевич Б. Д. и др.] – М.: Недра, 1984. – 344 с.
3. Справочник по инженерной геодезии / [Баран П. И., Войтенко С. П., Полищук Ю. В. и др.]. – К.: Высшая школа, 1978. – 376 с.
4. Земельний кодекс України. 25.10.2001, №2768-III.
5. Про землеустрій. Закон України від 22. 05. 2003, № 858-IV
6. Про затвердження Положення про технічний паспорт земельної ділянки, яка виставляється на земельні торги. Постанова Кабінету Міністрів України від 16.05.2002р. № 648.
7. Про затвердження Інструкції про загальні вимоги до оформлення технічного паспорта земельної ділянки, яка виставляється на земель-ні торги. Держкомзем України. Наказ № 114 від 10.07.2002.
8. Законодавство України про землю. - К.: Юрінком Інтер, 2002. - 353 с.
9. Земельне право. Академічний курс. Підручник /за ред. В.І.Семчика і П.Ф.Кулинича. -К.: Видавничий дім «Ін Юре», 2001, -242 с.
10. http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
11. http://dev.mysql.com/downloads/
12. Герберт Шілтд, Джеймс Холмс, Мистецтво програмування на Java, М: Вільямс, 331с.
13. Змитрович А.І. Бази данних.-Мінськ.: Університетське, 1981.
14. James Gosling, Bill Joy, Guy Steele, Gilad Bracha - The Java Language Specification, Second Edition.

ДОДАТОК А

```
public class main {
   public static void main(String[] args) {
     new Form. ();
   }

}
package com.oleg.oliinyk.cadastr.ui.Diplomna;

import com.oleg.oliinyk.cadastr.modelForm.*;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.io.IOException;
import java.sql.*;
import java.util.Vector;

public class Form extends JFrame {

   private static final int DEFAULT_WIDTH = 550;
   private static final int DEFAULT_HEIGHT = 380;
   private final FileOperations fileOperations;
   private final MenuItem mapSee;
   private final JButton seartchKadast;
   private final JLabel seartchKadastNumber;
   private JTextField textSearch;
   public JTextField textField1;
   public JTextField textField2;
   public JTextField textField3;
   public JTextField textField4;
   public JTextField textField5;
   public JTextField textField6;
   public JTextField textField7;
   public JTextField textField8;


   public MenuBar mb;
   public JTextArea text_area;
   public MenuItem openItem;
   public MenuItem saveAsItem;
   public MenuItem exitItem;
   public MenuItem updateArea;
   public MenuItem avtor;
   public MenuItem aboutProgram;
   public MenuItem addNewArea;
   public JButton updateTable;
   public JTable tableArea;
   protected Connection connection = null;
   protected Statement statement = null;
   protected ResultSet resultSet = null;
```

```
    public String[] args;
    private String searcha;


    public Form() throws ClassNotFoundException, SQLException, IOException {

        Class.forName("com.mysql.jdbc.Driver");
        connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/ProgramPasportisation", "root",
"root");
        statement = connection.createStatement();
        String sql = "SELECT geodesy.id,geodesy.cadastralNumber,geodesy.appoiment,geodesy.areas,geodesy.price,"
+
                "region,province,settlement,perimeter,firstName,lastName,middleName,x,y " +
                "FROM geodesy INNER JOIN location " +
                "ON geodesy.location_id = location.id  INNER JOIN owner ON geodesy.owner_id = owner.id " +
                "INNER JOIN placing ON location.placing_fk =placing.id INNER JOIN coordinates " +
                "ON placing.coordinates_fk = coordinates.id";
        resultSet = statement.executeQuery(sql);

        ResultSetMetaData resultSetMetaData = resultSet.getMetaData();

        int c = resultSetMetaData.getColumnCount();
        Vector column = new Vector(c);
        for (int i = 1; i <= c; i++) {

            column.add(resultSetMetaData.getCatalogName(i));


        }
        Vector data = new Vector();
        Vector row = new Vector();
        while (resultSet.next()) {
            row = new Vector(c);
            for (int i = 1; i <= c; i++) {
                row.add(resultSet.getString(i));

            }
            data.add(row);
        }
        fileOperations = new FileOperations(this);
        fileOperations.create();

        setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);
        setBackground(Color.gray);

        seartchKadastNumber = new JLabel("Пошук кадастрового номеру");
        seartchKadast = new JButton("Знайти номер кадастру");
        seartchKadast.setBounds(955, 255, 40, 100);

        textSearch = new JTextField();
        textSearch.setBounds(955, 290, 40, 90);


        updateTable = new JButton("Оновити таблицю");
        updateTable.setBounds(955, 290, 40, 90);
```

```
        updateTable.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                //scrollPaneTable.updateUI();
            }
        });

        //add(updateTable);
        seartchKadast.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                searcha = textSearch.getText();
                try {
                    Class.forName("com.mysql.jdbc.Driver");
                    connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/UCApp", "root", "root");
                    statement = connection.createStatement();
                    String sql = "SELECT * FROM list_area WHERE kadastrovoi_nomer = " + searcha + "";
                    resultSet = statement.executeQuery(sql);
                    System.out.println(resultSet.toString());
                    boolean hasRows = false;


                        while (resultSet.next()) {
                            hasRows = true;
                            JOptionPane.showMessageDialog(null, "Ділянка кадастровий  номер якої = " +
resultSet.getString(2) +
                                    "\nПризначення : " + resultSet.getString(3) + "\nВласник : " + resultSet.getString(4) +
                                    "\nРозташування : " + resultSet.getString(6) + "\nЦіна : " + resultSet.getString(7));

                            break;
                        }
                    if (hasRows==false) {
                        JOptionPane.showMessageDialog(null, "Дані про даний кадастровий номер відсутні у базі
даних");
                    }



                } catch (ClassNotFoundException e1) {
                    e1.printStackTrace();
                } catch (SQLException e1) {
                    e1.printStackTrace();
                }
            }
        });
        updateTable.addActionListener(e -> {
            fileOperations.create();
        });


        {
            mb = new MenuBar();
            Menu fileMenu = new Menu("Файл");
            Menu areaMenu = new Menu("Ділянка");
            Menu mapMenu = new Menu("На карті");
            Menu aboutMenu = new Menu("Про...");
```

```
    mb.add(fileMenu);
    mb.add(areaMenu);
    mb.add(mapMenu);
    mb.add(aboutMenu);

    openItem = new MenuItem("Відкрити...", new MenuShortcut(KeyEvent.VK_O));
    saveAsItem = new MenuItem("Зберегти", new MenuShortcut(KeyEvent.VK_R));
    exitItem = new MenuItem("Вихід", new MenuShortcut(KeyEvent.VK_Q));
    fileMenu.add(openItem);
    fileMenu.add(saveAsItem);
    fileMenu.addSeparator();
    fileMenu.add(exitItem);

    addNewArea = new MenuItem("Додати ділянку", new MenuShortcut(KeyEvent.VK_P));
    updateArea = new MenuItem("Відкрити ділянку", new MenuShortcut(KeyEvent.KEY_PRESSED));
    areaMenu.add(addNewArea);
    areaMenu.add(updateArea);

    mapSee = new MenuItem("Побачити на карті");
    mapMenu.add(mapSee);

    avtor = new MenuItem("Про автора", new MenuShortcut(KeyEvent.VK_Y));
    aboutProgram = new MenuItem("Про програму", new MenuShortcut(KeyEvent.VK_E));
    aboutMenu.add(avtor);
    aboutMenu.add(aboutProgram);
}

// Відкрити файл
openItem.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        try {
            fileOperations.open(text_area);
        } catch (IOException e1) {
            e1.printStackTrace();
        }
    }
});

//створити новий файл


/*saveItem.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        setupValues();
        fileOperations.save(text_area);

    }
});*/
//Сохранити як
saveAsItem.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        setupValues();
        fileOperations.saveAs(text_area);
```

```
    }
});

// Вихід  з програми
this.exitItem.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        FileOperations.exit(Form.this);
    }
});

//додати нову ділянку
addNewArea.addActionListener(new ActionListener() {


    @Override
    public void actionPerformed(ActionEvent e) {

        try {
            CreateAreaForm.main(args);

        } catch (Exception e1) {

        }
    }


});
//Змінити файл
updateArea.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {

        UpdateFile.main(args);


    }
});


mapSee.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        ShowMap.main(args);
    }
});

// Про автора
avtor.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        AboutTheAuthor.main(args);
    }
});

aboutProgram.addActionListener(new ActionListener() {
```

```
    @Override
    public void actionPerformed(ActionEvent e) {

        AboutTheProgram.main(args);
    }
});
{

    tableArea = new JTable(data, column);
    tableArea.setSize(500, 250);
    text_area = new JTextArea("Enter text", 35, 35);


    //text_area.setCaretPosition(0);
}
{



    //tableArea.setBackground(Color.darkGray);

}
{
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setTitle("Програма паспортизації ділянки");

    Box box = Box.createVerticalBox();
    box.add(seartchKadastNumber);
    box.add(textSearch);
    box.add(Box.createVerticalStrut(8));
    box.add(seartchKadast);
    box.add(Box.createVerticalStrut(10));
    // box.add(updateTable);


    setLocationRelativeTo(null);
    setMenuBar(mb);

    Box bv = Box.createVerticalBox();
    JPanel panelTable = new JPanel();
    JPanel panelText = new JPanel();
    JPanel panelButton = new JPanel();


    //tableArea.setBounds(0, 0, getWidth(), 300);

    //getContentPane().add(new JScrollPane(tableArea));
    panelTable.add(tableArea);

    //panelTable.add(new JScrollPane(tableArea));


    tableArea.setVisible(true);
    JScrollPane ScrollPaneText = new JScrollPane(text_area);
    ScrollPaneText.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS);

    getContentPane().add(ScrollPaneText);
```

```
            panelButton.add(box);


            add(panelTable, BorderLayout.NORTH);
            add(panelText, BorderLayout.WEST);
            add(panelButton, BorderLayout.EAST);

            setVisible(true);

            //add(updateTable);
        }

    }
package com.oleg.oliinyk.cadastr.ui.Diplomna;

import com.oleg.oliinyk.cadastr.Securety.WordFilter;
import utils.DbUtils;

import javax.swing.*;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;


public class Avtorization {
    protected static String login;
    protected static String password;

    private static Connection connection = null;
    private ResultSet resultSet;
    private String lv = "1";
/*SELECT user.login.password.level,
    user_attribute.user_attribute.user_attribute.fistName.user_attribute.lastName.middleName
    FROM user
    INNER JOIN user_attribute ON user.id = user_attribute.id_fk;*/

    public Avtorization(JTextField loginItem, JPasswordField passwordItem) {
        this.login = loginItem.getText();
        this.password = passwordItem.getText();
    }
    public static boolean validate() throws SQLException {
        if (login.length() < 4) {
            Message.showError("Довжина логіна повинна становити 4 і більше символів");
            return false;
        }
        if (password.length() < 4) {
            Message.showError("Довжина пароля повинна становити 4 і більше символів");
            return false;
        }
        if (!WordFilter.filter(login)) {
            Message.showError("Логін містить недоступні символи");
            return false;
        }
```

```
        if (!WordFilter.filter(password)) {
            Message.showError("Пароль містить недоступні символи");
            return false;
        }
        return true;
    }

    public static boolean connect() {
        connection = DbUtils.getDbConnection();
        if (connection == null) {
            Message.showError("Cannot connect to db");
            return true;
        }
        return false;
    }

    public boolean authorize() throws SQLException {
        if (!validate()) {
            return false;
        }
        try {
            if (connect()) return false;
            //String sql = "SELECT * FROM user WHERE login LIKE ('" + loginItem.getText() + "%')AND password
LIKE ('" + passwordItem.getText() + "%')";
            String sql = "SELECT * FROM  user where login=? and password=?";
            PreparedStatement statement = connection.prepareStatement(sql);
            statement.setString(1, login);
            statement.setString(2, password);
            resultSet = statement.executeQuery();
            if (resultSet.next()) {
                for (int j=1; j<=3; j++) {
                    System.out.printf("column %d = %s\n", j, resultSet.getString(j));
                }
            } else {
                Message.showError("Невірні контактні дані");
                return false;
            }
        } catch (Exception e1) {
            e1.printStackTrace();
        } finally {
            connection.close();
        }
        return true;
    }

    public boolean register() throws SQLException {
        if (!validate()) {
            return false;
        }
        try {
            if (connect()) return false;
            System.out.println("4");
            String sql = "INSERT INTO user (login, password,lv) VALUES (?, ?, ?)";
            PreparedStatement statement = connection.prepareStatement(sql);
            statement.setString(1, login);
            statement.setString(2, password);
```

```
            statement.setString(3, lv);
            int updateRes = statement.executeUpdate();
            System.out.println("updateRes = " + updateRes);
        } catch (Exception e1) {
            e1.printStackTrace();
            Message.showError("Cannot register");
            return false;
        } finally {
            connection.close();
        }
        Message.showMessage("You've been successfully registered. Try log in");
        return true;
    }
}
package com.oleg.oliinyk.cadastr.ui.Diplomna;

import assets.FileTemplate;
import com.oleg.oliinyk.cadastr.model.UkrainianCadastralExchangeFile;

import javax.swing.*;
import javax.swing.filechooser.FileNameExtensionFilter;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Marshaller;
import javax.xml.bind.Unmarshaller;
import java.io.*;

/**
 * Created by Oleh on 2/4/2016.
 */
public class FileOperations {
    private final Form form;

    public UkrainianCadastralExchangeFile cadastralExchangeFile;
    public JTextArea text_area;
    private Unmarshaller unmarshaller;
    private Marshaller marshaller;
    private File file;
    private InputStream stream;

    public FileOperations(Form form) throws IOException {
        this.form = form;

        try {
            JAXBContext jaxbContext = JAXBContext.newInstance("com.oleg.oliinyk.cadastr.model");
            unmarshaller = jaxbContext.createUnmarshaller();
            marshaller = jaxbContext.createMarshaller();
        } catch (Exception e) {
            System.err.println(e);
        }
    }

    public static void exit(JFrame frame) {
        if (JOptionPane.showConfirmDialog(frame,
                "Ви дійсно хочите вийти?", "Really Closing?",
                JOptionPane.YES_NO_OPTION,
```

```
            JOptionPane.QUESTION_MESSAGE) == JOptionPane.YES_OPTION) {
        System.exit(0);
    }
}

public void open(JTextArea text_area) throws IOException {
    this.text_area = text_area;
    JFileChooser fileChooser = new JFileChooser();
    FileNameExtensionFilter filter = new FileNameExtensionFilter(
            ".xml","xml");
    fileChooser.setFileFilter(filter);
    if (fileChooser.showOpenDialog(form) == JFileChooser.APPROVE_OPTION) {
        fileChooser.setCurrentDirectory(new File("/home/me/Documents"));



        File file = fileChooser.getSelectedFile();
        try {
            BufferedReader br = new BufferedReader(new FileReader(file));
            try {
                StringBuilder sb = new StringBuilder();
                String line;

                while ((line = br.readLine()) != null) {
                    sb.append(line + "\n");
                }
                text_area.setText(sb.toString());
            } finally {
                br.close();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
   /* stream = FileTemplate.class.getResourceAsStream("6125285700010020343.xml");
    BufferedReader reader = new BufferedReader(new InputStreamReader(stream));
    String line = reader.readLine();
    text_area.append(line + "\n");

    while (line != null) {
        text_area.append(line + "\n");


    }*/
    }
}


public void save(JTextArea text_area) {
    this.text_area = text_area;
    try {
        StringBuilder sb = new StringBuilder(text_area.getText());
        BufferedWriter bufferedReader = new BufferedWriter(new FileWriter("test.txt"));
        bufferedReader.write(String.valueOf(sb));
        bufferedReader.close();
        //marshaller.marshal(cadastralExchangeFile, file);
    } catch (Exception e1) {
```

```
            e1.printStackTrace();
        }
    }

    public void saveAs(JTextArea text_area) {
        this.text_area = text_area;
        JFileChooser chooser = null;
        chooser = new JFileChooser();
        chooser.setSelectedFile(new File(".xml"));
        FileNameExtensionFilter filter = new FileNameExtensionFilter(
                ".xml" , "xml");
        chooser.setFileFilter(filter);
        int r = chooser.showSaveDialog(form);
        if (r == JFileChooser.APPROVE_OPTION) {
            file = chooser.getSelectedFile();
            try {
                marshaller.marshal(cadastralExchangeFile, file);
            } catch (Exception e1) {
                e1.printStackTrace();
            }
        }
    }

    public void create() {
        stream = FileTemplate.class.getResourceAsStream("6125285700010020343.xml");

        try {
            cadastralExchangeFile = (UkrainianCadastralExchangeFile)
                    unmarshaller.unmarshal(stream);
        } catch (JAXBException e) {
            e.printStackTrace();
        }
    }

}
package UkrainianCadastral;

import assets.FileTemplate;
import com.oleg.oliinyk.cadastr.model.UkrainianCadastralExchangeFile;
import com.oleg.oliinyk.cadastr.modelForm.UpdateFile;

import javax.swing.*;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Marshaller;
import javax.xml.bind.Unmarshaller;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;

/**
 * Created by Oleh on 5/12/2016.
 */
public class LoadAndUpdateFile {
    //private final Experement experement;
```

```
    private final UpdateFile updateFile;
    public UkrainianCadastralExchangeFile cadastralExchangeFile;
    public JTextArea text_area;
    private Unmarshaller unmarshaller;
    private Marshaller marshaller;
    private File file;
    private InputStream stream1;
    private static InputStream stream;
    private JTextField textField1;
    private JTextField textField2;
    private JTextField textField3;
    private JTextField textField4;
    private JTextField textField5;

    private JTextField textField7;
    private JTextField textField8;
    private JTextField textField9;
    private JTextField textField10;
    private JTextField textField11;
    private JTextField textField12;


    public LoadAndUpdateFile(UpdateFile updateFile) throws IOException {
        this.updateFile = updateFile;

        try {
            JAXBContext jaxbContext = JAXBContext.newInstance("com.oleg.oliinyk.cadastr.model");
            unmarshaller = jaxbContext.createUnmarshaller();
            marshaller = jaxbContext.createMarshaller();
        } catch (Exception e) {
            System.err.println(e);
        }
    }


    public void openFileIeInDocument(JTextField textField1, JTextField textField2, JTextField textField3,
JTextField textField4, JTextField textField5, JTextField textField7, JTextField textField8, JTextField textField9,
JTextField textField10, JTextField textField11, JTextField textField12) {
        this.textField1 = textField1;
        this.textField2 = textField2;
        this.textField3 = textField3;
        this.textField4 = textField4;
        this.textField5 = textField5;
        this.textField7 = textField7;
        this.textField8 = textField8;
        this.textField9 = textField9;
        this.textField10 = textField10;
        this.textField11 = textField11;
        this.textField12 = textField12;


        JFileChooser fileChooser = new JFileChooser();
        if (fileChooser.showOpenDialog(updateFile) == JFileChooser.APPROVE_OPTION) {
            File file = fileChooser.getSelectedFile();
            try {
                cadastralExchangeFile = (UkrainianCadastralExchangeFile)
```

```
                unmarshaller.unmarshal(new FileInputStream(file));
            //String alfa = cadastralExchangeFile.getInfoPart().getCadastralZoneInfo().getCadastralZoneNumber();
            String KOATUU = cadastralExchangeFile.getInfoPart().getCadastralZoneInfo().getKOATUU();
            String appoimentArea =
cadastralExchangeFile.getInfoPart().getCadastralZoneInfo().getCadastralQuarters().getCadastralQuarterInfo().getPa
rcels().getParcelInfo().getCategoryPurposeInfo().getPurpose();
            String region =
cadastralExchangeFile.getInfoPart().getCadastralZoneInfo().getCadastralQuarters().getCadastralQuarterInfo().getPa
rcels().getParcelInfo().getParcelLocationInfo().getRegion();
            String settlement =
cadastralExchangeFile.getInfoPart().getCadastralZoneInfo().getCadastralQuarters().getCadastralQuarterInfo().getPa
rcels().getParcelInfo().getParcelLocationInfo().getSettlement();
            String district =
cadastralExchangeFile.getInfoPart().getCadastralZoneInfo().getCadastralQuarters().getCadastralQuarterInfo().getPa
rcels().getParcelInfo().getParcelLocationInfo().getDistrict();
            String coordinatesX =
cadastralExchangeFile.getInfoPart().getMetricInfo().getPointInfo().getPoint().getX();
            String coordinatesY =
cadastralExchangeFile.getInfoPart().getMetricInfo().getPointInfo().getPoint().getY();
            String ownerFirstName =
cadastralExchangeFile.getInfoPart().getCadastralZoneInfo().getCadastralQuarters().getCadastralQuarterInfo().getPa
rcels().getParcelInfo().getProprietors().getProprietorInfo().getAuthentication().getNaturalPerson().getFullName().ge
tFirstName();
            String ownerLastName =
cadastralExchangeFile.getInfoPart().getCadastralZoneInfo().getCadastralQuarters().getCadastralQuarterInfo().getPa
rcels().getParcelInfo().getProprietors().getProprietorInfo().getAuthentication().getNaturalPerson().getFullName().ge
tLastName();
            String ownerMiddleName =
cadastralExchangeFile.getInfoPart().getCadastralZoneInfo().getCadastralQuarters().getCadastralQuarterInfo().getPa
rcels().getParcelInfo().getProprietors().getProprietorInfo().getAuthentication().getNaturalPerson().getFullName().ge
tMiddleName();
            String areaSize =
cadastralExchangeFile.getInfoPart().getCadastralZoneInfo().getCadastralQuarters().getCadastralQuarterInfo().getPa
rcels().getParcelInfo().getParcelMetricInfo().getArea().getSize();
            String perimeter =
cadastralExchangeFile.getInfoPart().getCadastralZoneInfo().getCadastralQuarters().getCadastralQuarterInfo().getPa
rcels().getParcelInfo().getLandsParcel().getLandParcelInfo().getMetricInfo().getPerimeter();
            textField1.setText(KOATUU);
            textField2.setText(areaSize);
            textField3.setText(ownerLastName);
            textField4.setText(region);
            textField5.setText(district);
            textField7.setText(ownerFirstName);
            textField8.setText(ownerMiddleName);
            textField9.setText(settlement);
            textField10.setText(coordinatesX);
            textField11.setText(coordinatesY);
            textField12.setText(perimeter);

//System.out.println(cadastralExchangeFile.getInfoPart().getCadastralZoneInfo().getCadastralZoneNumber());
        } catch (Exception e) {

        }
    }

}
```

```
    public void loadAndUpdateFile(JTextField textField1, JTextField textField2, JTextField textField3, JTextField
textField4, JTextField textField5, JTextField textField6, JTextField textField7) throws IOException {
        this.textField1 = textField1;
        this.textField2 = textField2;
        this.textField3 = textField3;
        this.textField4 = textField4;
        this.textField5 = textField5;

        this.textField7 = textField7;

        JFileChooser fileChooser = new JFileChooser();
        if (fileChooser.showOpenDialog(null) == JFileChooser.APPROVE_OPTION) {
            File file = fileChooser.getSelectedFile();
            try {
                cadastralExchangeFile = (UkrainianCadastralExchangeFile)
                        unmarshaller.unmarshal(new FileInputStream(file));
                String alfa = cadastralExchangeFile.getInfoPart().getCadastralZoneInfo().getCadastralZoneNumber();
                textField1.setText(alfa);
                System.out.println("Zone number " +
cadastralExchangeFile.getInfoPart().getCadastralZoneInfo().getCadastralZoneNumber());

            } catch (Exception e) {

            }

        }
    }

    public void create() {
        stream1 = FileTemplate.class.getResourceAsStream("6125285700010020343.xml");

        try {
            cadastralExchangeFile = (UkrainianCadastralExchangeFile)
                    unmarshaller.unmarshal(stream1);
        } catch (JAXBException e) {
            e.printStackTrace();
        }
    }

}

package com.oleg.oliinyk.cadastr.model;

import java.io.Serializable;

/**
 * Created by Oleh on 2/29/2016.
 */
public class Area implements Serializable {
    float area;
    String address;
    String purpose;
    String cadasterNumber;
    String location;
    String owner;
```

```
public String getAddress() {
    return address;
}

public void setAddress(String address) {
    this.address = address;
}

public float getArea() {
    return area;
}

public void setArea(float area) {
    this.area = area;
}

public String getCadasterNumber() {
    return cadasterNumber;
}

public void setCadasterNumber(String cadasterNumber) {
    this.cadasterNumber = cadasterNumber;
}

public String getLocation() {
    return location;
}

public void setLocation(String location) {
    this.location = location;
}

public String getOwner() {
    return owner;
}

public void setOwner(String owner) {
    this.owner = owner;
}

public String getPurpose() {
    return purpose;
}

public void setPurpose(String purpose) {
    this.purpose = purpose;
}}
```