НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

Завідувач кафедри

Комп'ютерних наук
(назва кафедри)

_____ Голуб Б.Л.
(підпис) (ПІБ)

"_3" червня 2025 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему

«Програмне забезпечення інформаційно-управляючої системи обліку призовників»

Спеціальність 121 – «Інженерія програмного забезпечення»

Гарант освітньої програми

к.н.т., доцент _____ Вайганг Г.О.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник бакалаврської кваліфікаційної роботи

ст. викладач _____ Міловідов Ю.О.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ Гірченко Андрій Анатолійович
(підпис) (ПІБ студента)

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ

І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ

Завідувач кафедри

Комп'ютерних наук

к.н.т., доцент _____ Голуб Б.Л.

(науковий ступінь, вчене звання) (підпис) (ПІБ)

"_3" червня 2025 р.

ЗАВДАННЯ

на виконання бакалаврської кваліфікаційної роботи студенту

Гірченко Андрій Анатолійович

(прізвище, ім'я, по батькові)

Спеціальність 121 – «Інженерія програмного забезпечення»

Тема бакалаврської кваліфікаційної роботи Програмне забезпечення інформаційно-управляючої системи обліку призовників

Затверджена наказом ректора НУБіП України від "16" грудня 2024 р. №2249 С

Термін подання завершеної роботи на кафедру 2025.05.25

(рік, місяць, число)

Вихідні дані до бакалаврської кваліфікаційної роботи

Опис предмету дослідження, опис програмного забезпечення

Перелік питань, які потрібно розробити:

Системний аналіз предметної області

Аналіз предметної області

Розробка програмного забезпечення

Рекомендації щодо впровадження та експлуатації системи

Дата видачі завдання "16" грудня 2024 р.

Керівник бакалаврської кваліфікаційної роботи _____ Міловідов Ю.О.

(підпис) (прізвище та ініціали)

Завдання прийняв до виконання _____ Гірченко А.А.

(підпис) (прізвище та ініціали студента)

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

C# – Об’єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET

БД – База даних

СУБД – Система управління базами даних

СКБД – Система керування базами даних

SQL – Structured Query Language

ПЗ – Програмне забезпечення

ІТ – Інформаційні технології

CRM – Система управління взаємодіями з клієнтами

IDE – Integrated Development Environment (Інтегроване середовище розробки)

ВСТУП

У сучасному світі інформаційні технології відіграють ключову роль у цифровізації процесів управління в різних сферах суспільного життя. Особливого значення це набуває в державному секторі, де автоматизація облікових та адміністративних процедур дозволяє підвищити ефективність, прозорість і контроль діяльності органів влади. Однією з таких сфер є військовий облік, зокрема облік призовників, який є важливою складовою національної безпеки.

Система обліку призовників має забезпечувати оперативне оновлення даних, зручний доступ до інформації, захист персональних даних та автоматизацію ключових функцій, що виконуються у військкоматах та суміжних структурах. В умовах зростання кількості даних та необхідності взаємодії між різними підрозділами особливо актуальним стає питання створення інформаційно-управляючої системи, яка забезпечить ефективний контроль і управління процесами обліку призовників.

Дана бакалаврська кваліфікаційна робота присвячена розробці програмного забезпечення для інформаційно-управляючої системи обліку призовників.

Актуальність теми зумовлена потребою в модернізації існуючих засобів обліку, які часто є застарілими, не інтегрованими між собою та вимагають значних людських ресурсів для обробки інформації. Запропонована система покликана спростити процеси внесення, оновлення та перегляду даних, автоматизувати формування звітів і забезпечити високий рівень безпеки персональної інформації.

У рамках роботи було проведено аналіз чинних підходів до ведення обліку призовників, вивчено наявні програмні рішення, виявлено їхні переваги та недоліки. На основі цього сформовано вимоги до нової системи, особливу увагу було приділено зручності користувацького інтерфейсу, логіки

доступу до даних, а також можливості масштабування та подальшої інтеграції з іншими державними реєстрами.

Структура роботи охоплює етапи аналізу предметної області, системного проєктування, розробки функціональних модулів, тестування і підготовки до впровадження програмного забезпечення у практичну діяльність.

Таким чином, дана кваліфікаційна робота вирішує важливу задачу цифрової трансформації процесу обліку призовників, що сприятиме підвищенню ефективності управлінських рішень у сфері військового обліку та обороноздатності держави.

Актуальність

У сучасних умовах забезпечення національної безпеки та ефективної організації оборонної спроможності держави особливої ваги набуває процес військового обліку громадян. Однією з ключових категорій, що підлягає обліку є призовники – особи, які, відповідно до законодавства, підлягають призову на строкову службу. У зв’язку зі зростаючими вимогами до точності, своєчасності та повноти облікових даних, постає необхідність у впровадженні сучасних програмних рішень, які дозволятимуть автоматизувати ці процеси.

Більшість існуючих систем обліку призовників є застарілими, побудованими на ручному або частково автоматизованому введенні даних, що призводить до виникнення помилок, затримок та втрати інформації. Така ситуація ускладнює роботу працівників військових комісаріатів, а також уповільнює процеси аналізу, звітності та передачі даних між органами місцевого самоврядування, медичними установами та освітніми закладами.

Актуальність створення інформаційно-управляючої системи обліку призовників обумовлена такими чинниками:

- Потреба в автоматизації процесів введення, зберігання, оновлення та пошуку інформації про призовників.
- Необхідність забезпечення цілісності безпеки персональних даних відповідно до чинного законодавства.
- Оптимізація часу та ресурсів, які витрачаються на підготовку документів, формування звітів, ведення картотек.
- Забезпечення прозорості та контрольованості процесу призову, що є важливим для довіри з боку суспільства та ефективного управління військовим обліком.

Таким чином, впровадження сучасної інформаційної системи для обліку призовників є нагальною потребою, яка дозволить суттєво підвищити ефективність управлінських рішень у сфері оборонного планування та мобілізаційної готовності.

Тема

Дана розроблювана система створювалась з метою автоматизації обліку призовників, що сприятиме підвищенню ефективності військового обліку, зменшенню людських помилок при обробці персональних даних, пришвидшенню формування звітної документації та покращенню координації між різними підрозділами, задіяними у військовому призиві. Особливу увагу приділено зручності роботи з системою для співробітників військкоматів, безпеці персональних даних, можливості масштабування та подальшій інтеграції з іншими державними інформаційними системами.

Зв’язок роботи з науковими програмами, планами, темами

Робота виконувалась згідно плану науково-дослідних робіт кафедри комп’ютерних наук факультету інформаційних технологій Національного університету біоресурсів і природокористування України.

Об’єкт дослідження

Об’єктом дослідження є процеси збору, зберігання, обробки та пошуку персональних даних призовників, а також розробка програмного забезпечення, яке реалізує ці функції з урахуванням вимог безпеки, доступності та масштабованості. Досліджувались алгоритми введення та оновлення записів, генерації звітів, керування доступом до інформації, а також механізми резервного копіювання і відновлення даних.

Завдання дослідження

1. Провести аналіз існуючих систем для обліку призовників та суміжних рішень у сфері державного управління.
2. Сформулювати функціональні та нефункціональні вимоги до майбутньої системи.
3. Розробити архітектуру інформаційно-управляючої системи.
4. Реалізувати модулі введення, редагування та перегляду даних призовників.
5. Розробити систему генерації звітів на основі збережених даних.
6. Створити інтерфейс користувача, орієнтований на працівників військових комісаріатів.
7. Налаштувати систему контролю доступу для захисту персональної інформації.
8. Забезпечити підтримку резервного копіювання та відновлення бази даних.
9. Провести тестування програмного забезпечення в умовах, наближених до реальних.
10. Передбачити можливість масштабування та інтеграції з іншими державними реєстрами.

Методи дослідження

У процесі дослідження використовувалися методи системного аналізу, моделювання, об’єктно-орієнтованого проєктування, структурного та функціонального тестування. Було здійснено порівняльний аналіз існуючих програмних рішень, таких як: «Оберіг», Excel / Google Sheets, 1С:Облік персоналу, кастомні системи військового обліку. Усі ці сервіси мають досить зручний інтерфейс та великий об’єм можливостей. Але кожна з даних систем має певні недоліки у використанні або розгортанні системи під конкретний територіальний центр комплектування (ТЦК). «Оберіг» має все потрібне для обліку призовників, проте його великим недоліком є закритий код, через що його майже неможливо налаштувати під лад конкретного ТЦК, а це є інколи дуже важливим. Excel та Google Sheets є досить зручним, проте застарілим способом ведення обліку, його і досі використовують у деяких сферах, проте автоматизація там на досить низькому рівні. 1С:Облік персоналу – має доволі зручний функціонал, але щоб його налаштувати під роботу конкретного територіального центру комплектування, потрібен програміст 1С, через що даний сервіс може дозволити собі не кожна невелика організація. Кастомні системи військового обліку були розроблені під потреби конкретного ТЦК, і він покриває повністю всі потреби територіального центру комплектування для якого він розроблявся, проте він як правило не має оновлень, адже був розроблений місцевими IT фахівцями під замовлення, та скоріше за все не буде придатний для використання в іншому ТЦК де можуть відрізнятись задачі та потреби.

Маючи перераховані вище обмеження, було прийнято рішення реалізувати нову інформаційну систему для поєднання всіх можливих функцій, які були обмежені або які були відсутні у аналогів. Для розробки даної системи буде використано наступні технічні та дослідницькі підходи:

- Проєктування бази даних

Значну увагу приділено вивченню методів забезпечення безпеки персональних даних та збереженню цілісності інформації у базах даних. Розробка велась з використанням СУБД Microsoft SQL Server.

- Об’єктно-орієнтований підхід

Розробка системи велася з використанням сучасних фреймворків та мов програмування, та з застосування принципів об’єктно-орієнтованого програмування, програмна логіка мого програмного забезпечення була реалізована завдяки мові C# та середовища Windows Forms, це забезпечить модульність, та найголовніше повторне використання коду що дає можливість легкого масштабування.

- Доступ до системи

Під час розробки буде реалізовано систему доступу до функціоналу системи, залежно від правильності введення даних аккаунту користувача, дозволятиметься або буде відмовлено в доступі до програмного забезпечення та його функціоналу

- Оформлення звітів

Користувач (працівник ТЦК) має можливість згенерувати звіт на основі заданої характеристики за потребою ТЦК.

- Проведення тестувань

Після створення програмного забезпечення буде проведено ряд тестувань, який стосуватиметься кожного компоненту програмного забезпечення, що має на меті перевірити стабільність, правильність оброки та цілісність даних.

Теоретична значимість

Теоретична значимість дослідження полягає у систематизації знань щодо проєктування автоматизованих систем обліку, що відповідають державним вимогам до зберігання та обробки персональних даних. У роботі

досліджено принципи побудови інформаційно-управляючих систем, виявлено вузькі місця традиційного обліку призовників та запропоновано шляхи їх усунення за допомогою сучасного ПЗ.

Практична значимість

Практична значимість розробленого програмного забезпечення полягає в можливості його застосування у військових комісаріатах, органах місцевого самоврядування та інших установах, задіяних у процесі призову. Система дозволяє автоматизувати основні функції з обліку призовників, зменшити ризик помилок при роботі з даними, скоротити час на підготовку звітів та забезпечити зручний доступ до актуальної інформації. У перспективі система може бути масштабована для ведення обліку інших категорій населення, що підлягають мобілізаційній підготовці.

1. СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Однією з важливих потреб сучасного суспільства є ефективне функціонування державних структур, зокрема у сфері військового обліку. З огляду на актуальність мобілізаційних питань та необхідність оперативного аналізу призовних ресурсів, постає потреба в створенні сучасного програмного забезпечення для інформаційно-управляючою системи обліку призовників.

Предметна область стосується автоматизації процесів обліку призовників у територіальних центрах комплектування та соціальної підтримки (ТЦК та СП). Традиційні підходи до ведення обліку – на паперових носіях або за допомогою розрізнених програми – не відповідають вимогам сьогодення. Вони призводять до дублювання інформації, затримок в обміні даними між підрозділами та складності у масштабуванні. Враховуючи зростаючий обсяг даних та необхідність у точності й актуальності інформації, зростає потреба у централізованій інформаційній системі.

Інформаційно-управляюча система обліку призовників повинна забезпечувати:

- Облік персональних даних призовників;
- Облік медичних, освітніх, сімейних даних;
- Ведення історії змін статусу;
- Генерація звітів та статистичних довідок;
- Пошук і фільтрація записів;
- Контроль строків та повідомлень.

Одним із базових компонентів такої системи є виконавчий модуль, який координує роботу інших складових, таких як модуль реєстрації призовників,

модуль обліку особових справ, система автентифікації та авторизації, а також допоміжні сервіси – звітність, налаштування тощо.

Розроблюване ПЗ має на меті забезпечити централізоване зберігання, обробку, оновлення й аналіз інформації про призовників – з моменту взяття на військовий облік до моменту їх зарахування на службу чи зняття з обліку. Система має зменшити навантаження на працівників ТЦК, знизити ризик помилок, пришвидшити пошук та обробку інформації.

1.2 Аналіз вимог до програмної системи

Функціональні вимоги – це опис конкретних дій, які має виконувати система, а також обмежень, що стосуються обробки даних і поведінки системи під час виконання цих дій.

Нефункціональні вимоги – це вимоги до умов, в яких функції системи повинні виконуватись. Вони не стосуються безпосередньо функціональності, але відображають очікування користувачів щодо якості, надійності, зручності тощо.

Після проведеного аналізу предметної області можна виокремити такі ключові функціональні та нефункціональні вимоги до системи.

Функціональні вимоги:

- Авторизація;
- Додавання, редагування та видалення даних про призовників;
- Перегляд записів призовників;
- Облік статусів призовника (придатний, не придатний, тимчасово непридатний тощо);
- Реалізувати пошук серед призовників за містом проживання;
- Збереження медичних даних та результатів медкомісій;
- Формування звітів;
- Збереження звітів в PDF форматі.

Нефункціональні вимоги:

- Ефективність – характеристика програмного забезпечення, що відображає його здатність задовольняти вимоги до продуктивності при одночасному мінімальному використанні системних ресурсів.
- Розширюваність – властивість системи, що забезпечує можливість її адаптації та внесення змін відповідно до нових або змінених вимог.
- Продуктивність – сторінка профілю користувача має завантажуватись не довше ніж за 1 хвилину, а час оновлення записів також не повинен перевищувати 1 хвилини.
- Надійність – гарантія того, що додавання нових записів відбуватиметься послідовно й без конфліктів, зберігаючи цілісність даних.
- Підтримка – передбачає періодичне інформування користувача про доступні оновлення системи.
- Інтуїтивно зрозумілий інтерфейс – меню програмного забезпечення має бути інтуїтивно зрозумілим для будь-якого користувача.
- Сумісність з Windows-системами – програмне забезпечення має бути сумісним з системами на операційній системі Windows.
- Стабільність при обробці великої кількості записів.

1.3 Моделювання предметної області

Моделювання є ключовим етапом у процесі розробки програмного забезпечення, оскільки воно дозволяє глибше зрозуміти предметну область та структуру майбутньої системи. Одним із найефективніших засобів для цього є UML (Unified Modeling Language) – уніфікована мова моделювання, що широко використовується в об’єктно-орієнтованому підході до аналізу та проєктування систем. UML забезпечує зручний і потужний інструментарій для візуалізації структури та поведінки програмних систем.

UML-діаграми поділяються на два основних типа:

- Структурні діаграми (наприклад: діаграма класів, компонентів, розгортання, пакетів).
- Діаграми поведінки (зокрема: діаграма прецедентів, діаграма активності, діаграма послідовності).

Для моделювання програмного забезпечення системи обліку призовників доцільно використати діаграми поведінки, які дозволяють детально описати основні дії користувачів та взаємодію з системою.

Діаграма прецедентів є базовим інструментом, який дозволяє відобразити ролі користувачів (акторів) у системі та взаємодію цих ролей з функціоналом програмного забезпечення. Основними елементами діаграми є:

- Актор – зовнішній суб’єкт, що взаємодіє із системою (наприклад, працівник ТЦК або адміністратор);
- Прецедент – дія або функція яку виконує система у відповідь на дії актора (наприклад, перегляд даних призовника, додавання нової інформації, формування звітів тощо);
- Система – межі програмного продукту, що моделюється;
- Зв’язки – асоціації між акторами та прецедентами, що визначають способи взаємодії.

Актори системи:

- Користувач (працівник ТЦК):
    - Авторизація в системі;
    - Внесення нових даних про призовника;
    - Редагування даних;
    - Перегляд списків;
    - Генерація звітів;
    - Пошук призовників;
    - Зміна статусу призовника.

- Адміністратор:
    - Управління користувачами;
    - Резервне копіювання;
    - Відновлення БД.
- Керівник ТЦК:
    - Перегляд звітності;
    - Контроль виконання планів призову;
    - Аналітична обробка статистики.

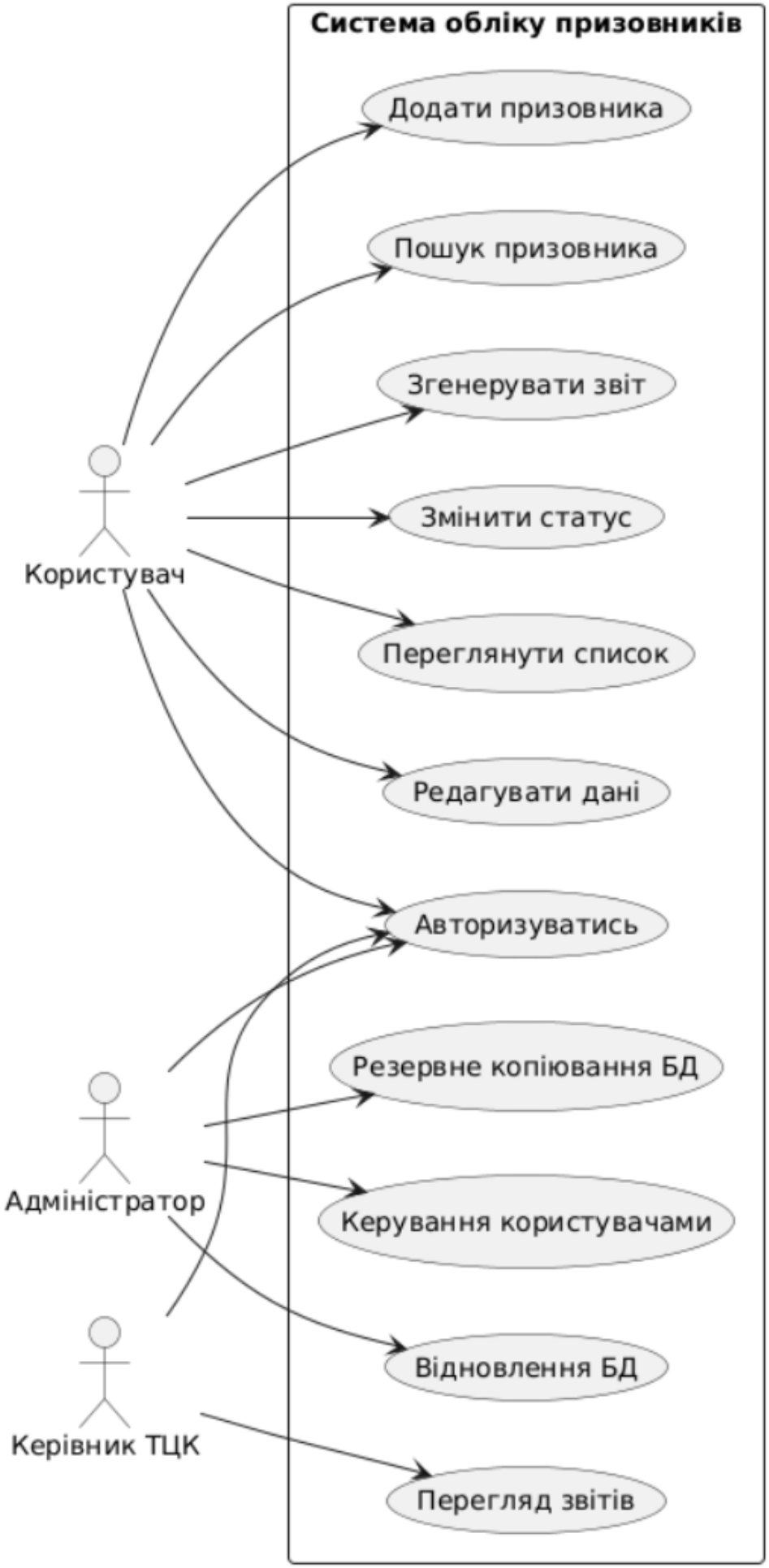

Система обліку призовників
Додати призовника
Пошук призовника
Згенерувати звіт
Змінити статус
Переглянути список
Редагувати дані
Авторизуватись
Резервне копіювання БД
Керування користувачами
Відновлення БД
Перегляд звітів
Користувач
Адміністратор
Керівник ТЦК

Рис. 1. Діаграма прецедентів

1.4 Огляд інформаційних джерел та існуючих рішень

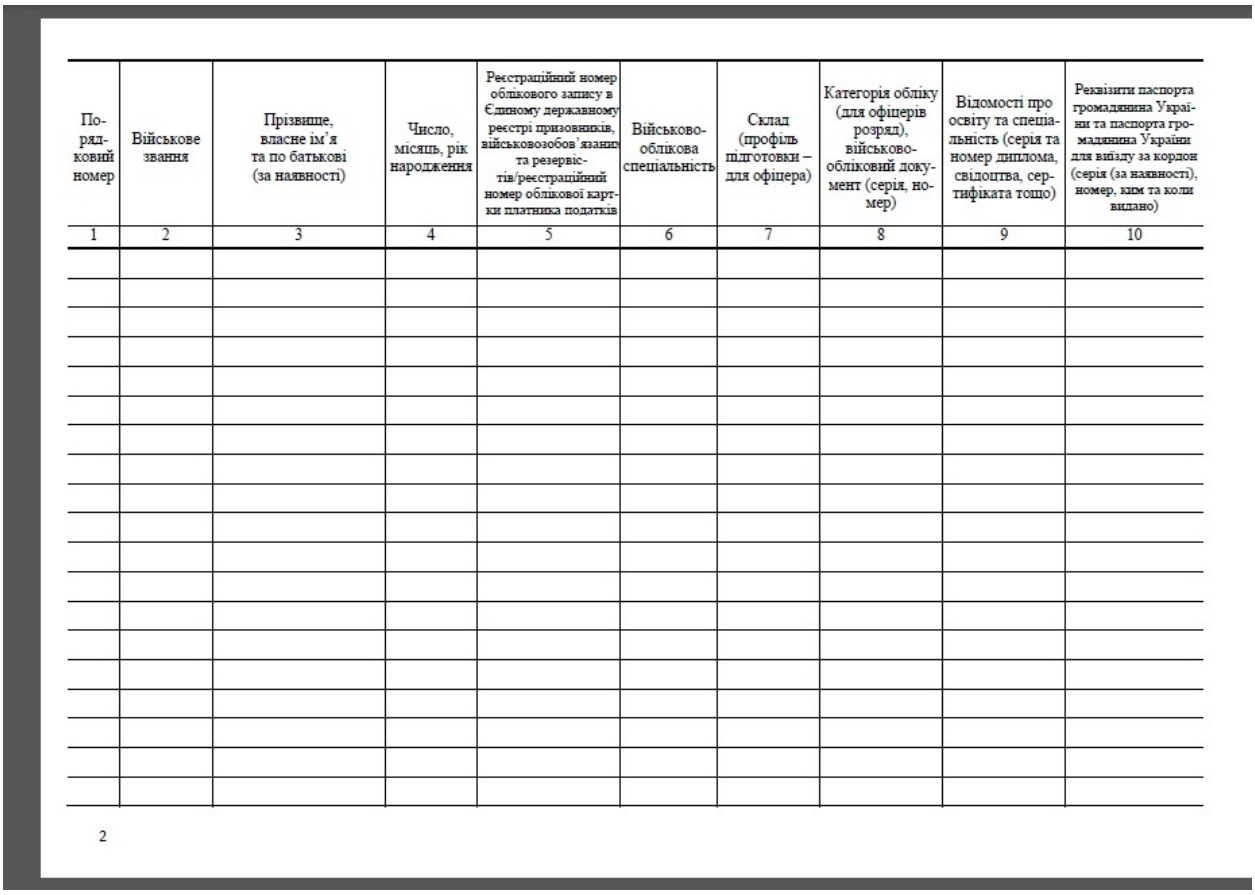
При розробці програмного забезпечення для обліку призовників важливо враховувати вже існуючі рішення, які частково реалізують подібні функціональні можливості. Аналіз аналогів дозволяє виявити типові недоліки та переваги, а також визначити функціональні та технічні характеристики, що потребують реалізації у власному програмному продуктів.

Фізичні існуючі рішення

Історично та в багатьох випадках досі, облік військовозобов’язаних та призовників ведеться за допомогою фізичних, паперових носіїв. Найбільш поширеним і базовим фізичним рішенням можна назвати рукописні журнали обліку, картотеки та блокноти.

Рис. 2. Паперовий журнал обліку

Це можуть бути як звичайні зошити чи блокноти без будь-якої спеціальної розмітки, де відповідальний співробітник сам організовує інформацію (наприклад, прізвища, дати народження, адреси, стан здоров’я), так і спеціалізовані формуляри, книги обліку або картотеки з вже готовими, стандартизованими розділами та полями для заповнення.



Порядковий номер	Військове звання	Прізвище, власне ім’я та по батькові (за наявності)	Число, місяць, рік народження	Реєстраційний номер облікового запису в Єдиному державному реєстрі призовників, військовозобов’язаних та резервістів/реєстраційний номер облікової картки платника податків	Військово-облікова спеціальність	Склад (профіль підготовки – для офіцера)	Категорія обліку (для офіцерів розряд), військово-обліковий документ (серія, номер)	Відомості про освіту та спеціальність (серія та номер диплома, свідоцтва, сертифіката тощо)	Реквізити паспорта громадянина України та паспорта громадянина України для виїзду за кордон (серія (за наявності), номер, ким та коли видано)
1	2	3	4	5	6	7	8	9	10

2

Рис. 3. Стандартизований журнал обліку для заповнення даних про військовозобов’язаних

Переваги обліку у фізичних журналах/картотеках без стандартизованої розмітки:

1) Гнучкість: співробітник що веде облік може повністю підлаштовувати структуру записів, розділи та категорії інформації відповідно до своїх поточних потреб та специфіки роботи конкретного військового комісаріату чи підрозділу.

2) Простота початку: не потребує спеціального програмного забезпечення або обладнання, достатньо ручки та паперу.
3) Автономність: працює незалежно від електроенергії чи доступу до мережі інтернет, що досить актуально на сьогодні на жаль.

Недоліки обліку у фізичних журналах/картотеках без стандартизованої розмітки:

1) Часові витрати: створення та підтримка системи обліку з нуля, а також ручне заповнення даних, займає значно більше часу порівняно з автоматизованими системами. Це включає пошук, верифікацію та занесення кожної одиниці інформації
2) Висока ймовірність помилок: людський фактор відіграє значну роль. Помилки при ручному введенні даних пропуски, нерозбірливий почерк, неправильна класифікація інформації можуть призвести до неточностей у даних, що є критичним для військового обліку.
3) Складність аналізу та пошуку: швидкий пошук конкретного призовника, фільтрація за певними критеріями, наприклад, вік, стан здоров’я, освіта, або формування зведених звітів є вкрай складним або неможливим без значних часових витрат.
4) Проблеми з дублюванням та цілісністю даних: важко контролювати унікальність записів та забезпечувати цілісність даних, що може призвести до дублювання інформації про одного й того ж призовника.
5) Фізичне зберігання та безпека: зберігання паперових носіїв вимагає фізичного простору, схильне до ризиків втрати даних через пожежу, затоплення, фізичне зношення або несанкціонований доступ.

Переваги обліку у фізичних журналах/картотеках з готовою розміткою, в стандартизованих формах:

1) Організація: готові розділи та поля для заповнення допомагають швидко структурувати та організовувати записи, зменшуючи час на вирішення «що і куди писати».
2) Ефективність заповнення: завдяки стандартизованим шаблонам можна зекономити час на розробку структури та одразу переходити до заповнення даних.
3) Зменшення початкових помилок: наявність чітко визначених полів зменшує ймовірність пропуску важливої інформації.

Недоліки обліку у фізичних журналах/картотеках з готовою розміткою:

1) Обмежена гнучкість: готова структура може бути занадто жорсткою і не дозволяти адаптувати формуляри до непередбачених або специфічних потреб обліку, що змінюються з часом.
2) Відсутність особистого стилю: якщо у співробітника є специфічні вподобання щодо візуалізації інформації, хоч такий пункт для військового обліку не є пріоритетом, про те відсутність можливості персоналізації може дещо знизити комфорт роботи у такому випадку.
3) Масштабованість: зі збільшенням кількості призовників або вимог до звітності, паперові системи стають надзвичайно незручними та малоефективними.
4) Складність обміну даними: обмін інформацією між різними підрозділами або інстанціями стає трудомістким і повільним процесом, що потребує фізичного переміщення документів або їх ручного переписування/сканування.

Аналіз цих фізичних рішень чітко демонструє їхні фундаментальні обмеження у сучасному світі, особливо коли йдеться про великі обсяги даних, необхідність швидкого доступу, аналізу та забезпечення високого рівня безпеки та цілісності інформації. Ці недоліки є основною рушійною силою для розробки автоматизованих інформаційно-управляючих систем, таких як пропонована система обліку призовників.

Програмні існуючі рішення

У табл. 1 представлено аналіз найбільш поширених програмних рішень, які використовуються для ведення персонального або військового обліку в державних установах та локальних організаціях. Аналіз проводився за наступними критеріями:

- Доступність
- Відповідність специфіці військового обліку
- Рівень автоматизації
- Безпека даних
- Гнучкість у налаштуванні
- Інтеграція з іншими системами

Табл. 1

Порівняння аналогів:

Сервіс/Система	Переваги	Недоліки
Оберіг (державна система)	Інтеграція з реєстрами, офіційна підтримка	Обмежений доступ, не публічна
Excel / Google Sheets	Простота, гнучкість	Відсутність захисту даних, немає автоматизації, ручне введення
1С:Облік персоналу	Можливість обліку великої кількості людей	Неадаптована під специфіку військового обліку
Кастомні локальні рішення ТЦК	Пристосовані до конкретного офісу	Відсутність єдиних стандартів, труднощі в супроводі

1. Система «Оберіг»

Це централізована державна система, яка призначена для ведення реєстрів військовозобов’язаних. Її головними перевагами є офіційна підтримка та інтеграція з іншими державними реєстрами. Вона повністю відповідає чинному законодавству та нормативним вимогам щодо захисту персональних даних. Проте система «Оберіг» є закритою, її використання обмежене виключно державними структурами. Для сторонніх розробників вона недоступна, що унеможливлює її модифікацію або адаптації до специфіки конкретного територіального центру комплектування (ТЦК).

2. Таблиці Excel / Google Sheets

Найпоширеніше рішення серед невеликих організацій – використання електронних таблиць для ведення обліку. Основні переваги: простота, відсутність необхідності у програмуванні, легкий доступ та редагування. Однак цей підхід не відповідає сучасним вимогам до захисту персональних даних, не забезпечує контроль доступу, не має механізмів журналювання змін, і не здатен забезпечити автоматизацію перевірки строків явки чи генерацію звітів. Також ці таблиці не масштабуються і погано підходять для великих баз даних

Рис. 4. Логотип Microsoft Excel

Рис. 5. Логотип Google Sheets

3. 1С:Облік персоналу

Це потужна система, яка дозволяє вести облік співробітників, формувати звітність, контролювати документообіг. Вона може масштабуватися для обліку великої кількості осіб. Проте програма не передбачена для військового обліку. Необхідна дорога та складна адаптація для урахування специфічних параметрів, таких як військові звання, військовий квиток, категорії придатності, дані про приписку, мобілізаційну готовність тощо. Вартість ліцензій та послуг програмістів 1С також є бар'єром для впровадження в невеликих ТЦК.

Рис. 6. Логотип 1С

4. Локальні кастомні рішення ТЦК

У деяких військкоматах/ТЦК використовуються самописні програми, часто створені ентузіастами або місцевими IT-спеціалістами. Такі рішення можуть враховувати специфіку конкретного підрозділу, мати необхідний функціонал та бути інтегрованими з наявною інфраструктурою. Але ці програми, як правило, мають обмежену документацію, відсутню технічну підтримку, не оновлюються, мають проблеми з безпекою та не можуть масштабуватись або використовуватись в інших регіонах.

Після перегляду всіх переваг та недоліків кожної з систем було створено таблицю порівняння, для більш зручного перегляду даних про аналоги мого сервісу Табл. 1. Порівняння аналогів

1.5 Постановка завдання

Мета системи – створення програмного забезпечення, яке автоматизує процес обліку призовників на рівні районного чи міського ТЦК та СП.

Запропоноване рішення, яке розробляється в рамках кваліфікаційної роботи, дозволяє створити сучасну, безпечну та масштабовану систему обліку призовників. Архітектура MVVM забезпечує чіткий поділ відповідальностей, а використання бази даних PostgreSQL гарантує надійне збереження інформації. Система передбачає підтримку ролей користувачів, фільтрацію, звітність.

Система повинна дозволяти вводити, зберігати, редагувати, сортувати та аналізувати інформацію про призовників, формувати статистичні дані та звіти, забезпечувати контроль.

Очікується що система зменшить кількість помилок, скоротить час на виконання рутинних завдань і підвищить ефективність взаємодії між працівниками.

Основні інструменти реалізації:

- Мова програмування: C#
- База даних: Microsoft SQL Server
- Інтерфейс: Windows Forms
- Інтегроване середовище розробки (IDE): Microsoft Visual Studio

2. ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Побудова логічної моделі даних, у вигляді ER-діаграми

Дуже важлива частина будь-якого програмного забезпечення є база даних, вона відіграє критичну роль у збереженні, управлінні та доступі до даних. База даних це основа функціонування більшості систем не залежно від способу та варіанту реалізації. Саме вона забезпечуватиме надійність, безпеку та цілісність наших даних.

Одним з самих перших етапів проектування є створення логічної моделі даних. Саме завдяки ній ми можемо візуалізувати структуру даних та визначити які взаємозв’язки між ними, тому вона дуже важливий інструмент.

Логічна модель служить не лише для формального опису сутностей, їх властивостей і взаємозв’язків, а й допомагає чіткіше уявити структуру проєкту. Вона дозволяє виявити помилки в проєктуванні даних, уникнути їхнього дублювання або втрат, і закладає основу для наступного етапу – фізичної реалізації бази даних, визначаючи набір необхідної інформації для коректної роботи системи.

У складі логічної моделі виділяють три головні компоненти:

- Сутності (entities): об’єкти, про які зберігається інформація; у термінах реляційної БД відповідають таблицям і їх полям.
- Атрибути (attributes): характеристики сутностей, що визначають, які саме дані потрібно зберігати в кожній таблиці.
- Відношення (relationships): описують, як сутності пов’язані між собою:

- «один-до-одного» - кожна одиниця даних однієї сутності відповідає лише одній одиниці іншої.
- «один-до-багатьох» - один запис може мати зв’язок із багатьма записами іншої сутності.
- «багато-до-багатьох» - численні зв’язки між множинами записів з обох сторін.

Визначено такі основні сутності:

- Users – особа, яка взаємодіє з системою.
  - id: int (PK)
  - username: varchar
  - password: varchar
- Conscripts
  - Id: int (PK)
  - surname: char
  - name: char
  - second_name: char
  - birthday: datetime
  - place: char
  - health: char
  - suitability: char
  - marital_status: char
  - education: char

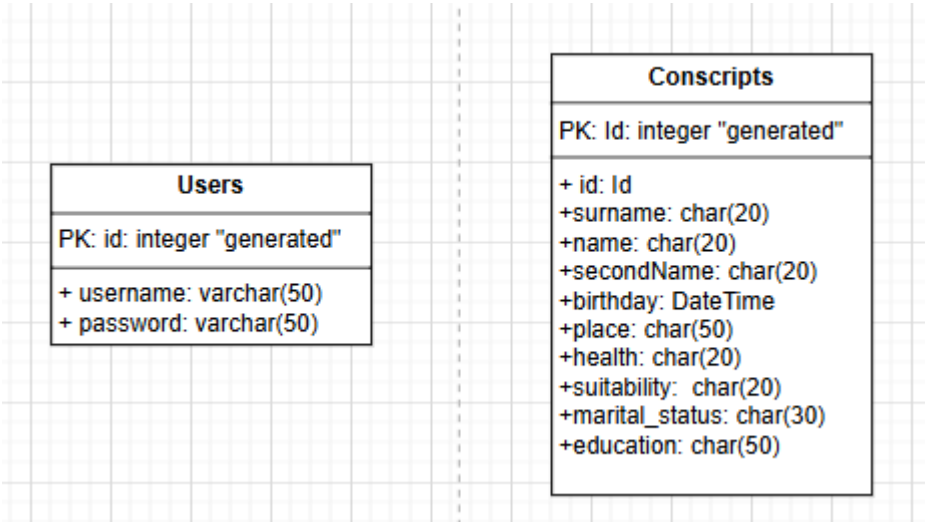


Рис. 7. Модель логічної бази системи

2.2 Діаграма класів

Діаграма класів відображає основні класи системи обліку призовників, їх атрибути, методи та зв’язки між ними. У розробці програмного забезпечення для обліку призовників діаграма класів використовується для моделювання структури об’єктів, які реалізують основну функціональність програми: аутентифікацію користувачів, додавання/редагування/видалення даних призовників, фільтрацію, формування звітів та експорт в PDF.

Основні класи системи:

Login:

Атрибути:

- username: string
- password: string

MainWindow:

Атрибути:

- login: Login

- addConscripts: AddConscripts[]
- filter: Filer
- reportForm: ReportForm
- about: About

Методи:

- login(username, password): void

Conscripts:

Основна сутність системи – призовник.

Була використана в більшості модулів: фільтрація, редагування, звітність.

Атрибути:

- id: Id
- surname, name, secondName: string
- birthday: DateTime
- place, health, suitability, marital_status, education: string

AddConscripts:

Клас для управління даними призовників

Атрибути:

- conscripts: Conscripts[]

Методи:

- addConscript(conscripts): void
- editConscript(conscripts): void
- deleteConscript(conscripts): void

Filter:

Відповідає за пошук і відбір призовників за заданим критерієм (місто)

Атрибути:

- conscript: Conscripts[]

Методи:

- GetFiltered(filter: string): List<ConscriptsFiltered>

ConscriptsFiltered:

Результат фільтрації призовників, має всі ті ж самі поля що Conscripts, і використовується як представлення у звітах.

ReportForm:

Відповідає за формування звітів.

Атрибути:

- Conscripts: Conscripts[]

Методи:

- GetFiltered(criteria: string): List<Conscripts>
- GenerateReport(criteria: string, filePath: string): void

ExporterToPdf:

Використаний для експорту призовників у PDF-звіт.

Атрибути:

- Conscripts: Conscripts[]

Методи:

- ExportToPdf(list: List<Conscripts>): void

Зв’язки між класами:

- Користувач керує списком Призовників та створює Звіти
- Призовник має пов’язані Записи

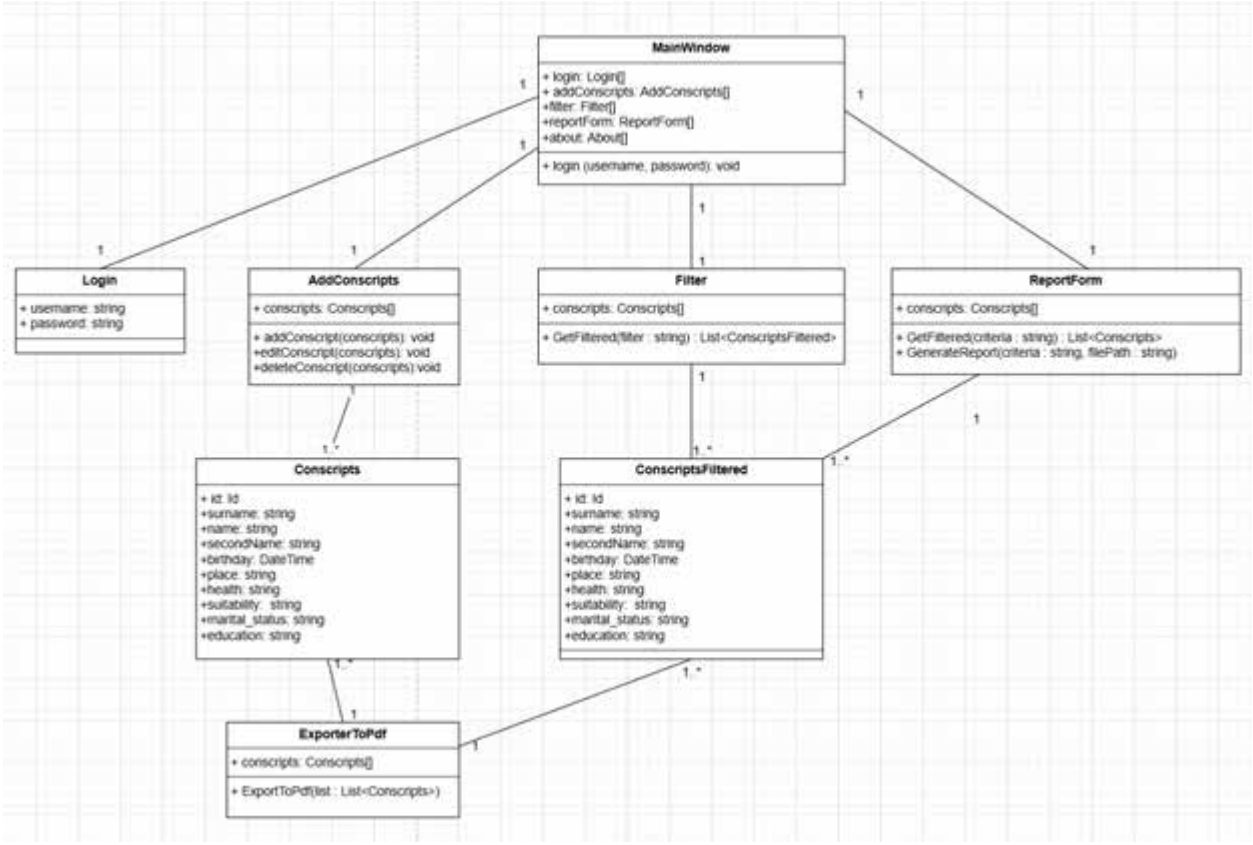


Рис. 8. Діаграма класів

2.3 Діаграма пакетів

Опис діаграми пакетів системи обліку призовників

Діаграма пакетів – це засіб моделювання, який використовується для проєктування структури програмного забезпечення на рівні модулів. Її призначення полягає в логічному об’єднанні класів, інтерфейсів та інших компонентів у пакети для кращої організації та структуризації системи.

Ця діаграма сприяє кращому розумінню архітектури системи та зв’язків між її компонентами, а також допомагає керувати складністю програмного забезпечення. Вона дозволяє застосовувати модульний підхід у розробці, забезпечуючи логічне поділення системи на підсистеми, що значно спрощує подальшу розробку, супровід і тестування.

Архітектурний підхід

Для реалізації системи обрано архітектурний підхід MVVM (Model-View-ViewModel), що забезпечує чіткий поділ між рівнями логіки, представлення і зберігання даних. Це дозволяє легше масштабувати систему та проводити модульне тестування.

Основною перевагою використання даної архітектури є те, що вона спрощує розробку та тестування програмного забезпечення. Графічний інтерфейс зазвичай залежить від платформи, тоді як бізнес-логіка може бути незалежною та повторно використовуваною. Компонент ViewModel виконує роль посередника між Model і View, забезпечуючи обмін даними та взаємодію користувача із системою.

На діаграмі пакетів архітектура системи представлена у вигляді трьох рівнів: Model, ViewModel і View.

Структура діаграми

View (інтерфейс користувача)
Містить усі форми програми, які взаємодіють із користувачем:

- Login – форма авторизації користувача в системі
- MainWindow – головне вікно програми з якого можно перейти на наступне потрібне нам вікно
- AddConscript – вікно додавання/редагування/видалення запису про призовника
- Conscripts – вікно перегляду списку призовників занесених в програму з можливістю збереження данного списку в PDF форматі
- Filter – вікно в якому ми можемо провести пошук по списку за заданим критерієм (місто)
- ReportForm – форма за допомогою якої ми можемо згенерувати звіт обравши потрібну категорію та задавши конкретний критерій
- About – вікно «Про систему»

- Navigation

Особливості:

- Усі форми напряму взаємодіють з Controller, який реалізує бізнес-логіку.
- Форма Login викликає метод Authenticate() через AuthService.
- Клас Navigation реалізує логіку переходів між формами.

ViewModel

Включає:

- AuthService – обробка автентифікації, логіка перевірки користувачів.
- Controller – центральна бізнес-логіка, яка координує взаємодію з Model.
- Interface Repository – інтерфейс доступу до даних (ін’єкція залежностей для зменшення зв’язаності).

Особливості:

- Controller залежить від Interface Repository, що дозволяє легко підмінювати джерела даних.

Model

Містить:

- Repository – реалізація доступу до даних.
- Database – джерело збереження даних.

Особливості:

- Repository забезпечує доступ до Database та реалізує інтерфейс Interface Repository.

Взаємозв’язки

- Forms використовують Controller та AuthService із шару ViewModel для обробки дій користувача.
- Controller залежить від Interface Repository, який реалізується в Repository.
- Repository звертається до Database для зчитування/запису даних.
- Navigation забезпечує переміщення між формами.

Ця діаграма чітко ілюструє модульну побудову системи обліку призовників. Використання MVVM забезпечує:

- Незалежність представлення від логіки,
- Можливість повторного використання компонентів,
- Простоту масштабування та тестування системи.

Система структурована за принципами слабкого зв’язку між шарами, що відповідає сучасним підходам до розробки ПЗ.

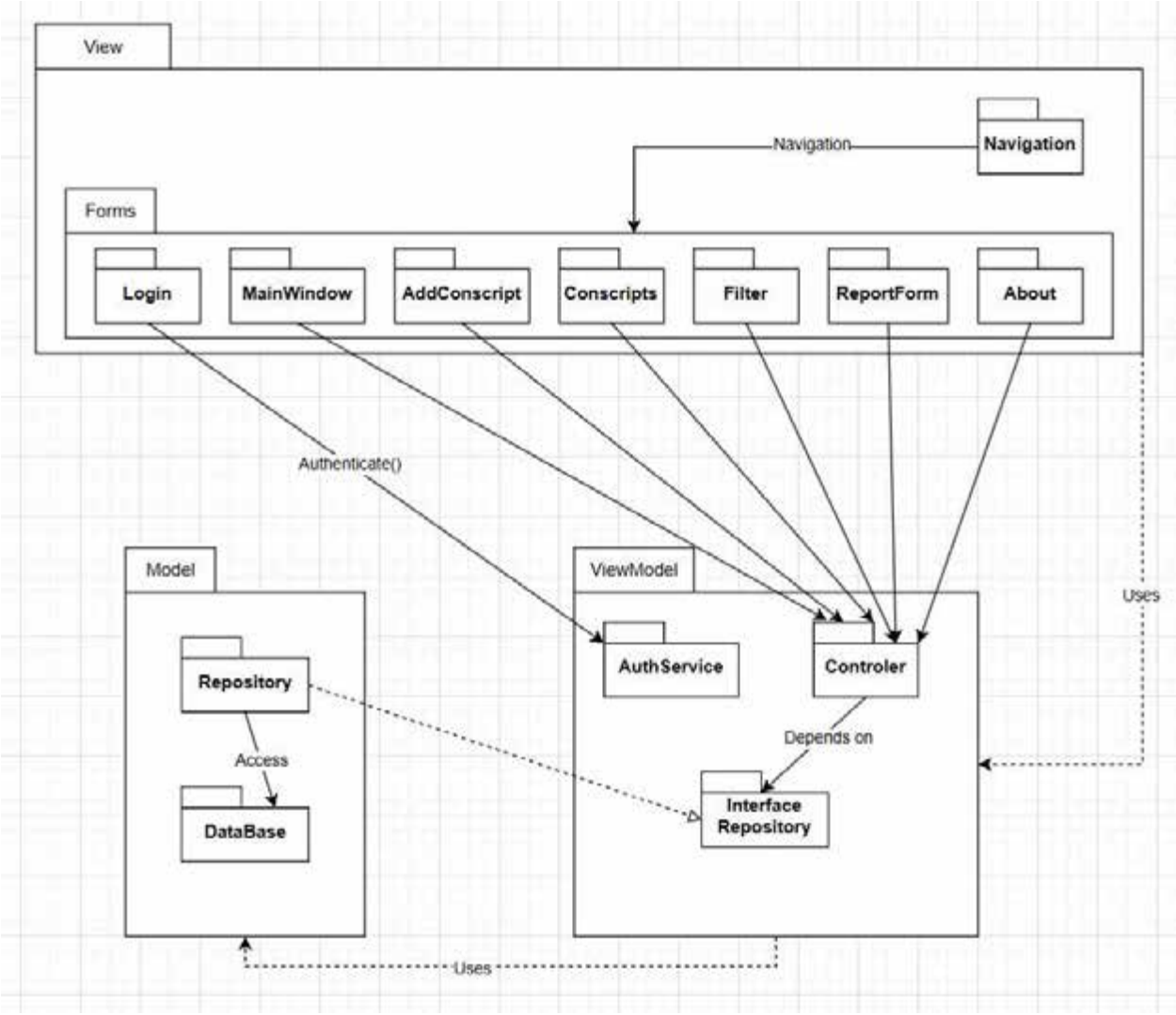


Рис. 9. Діаграма пакетів

2.4 Діаграма компонентів

Діаграма компонентів є одним із типів UML-діаграм, що використовується для моделювання структури програмної системи на рівні компонентів і їхніх взаємозв’язків. Вона дозволяє візуалізувати, як окремі частини системи взаємодіють між собою, які інтерфейси реалізують, та які залежності існують між модулями.

На діаграмі компонентів, представлені нижче, змодельовано програмну структуру системи обліку призовників. Основним виконуваним артефактом є компоненст conscript.exe, який координує роботу всіх функціональних частин

системи та взаємодіє з різними модулями через надані та необхідні інтерфейси. Серед компонентів виділяються:

- Authentication: компонент, який надає функціональність для входу в систему через інтерфейс LoginInterface.
- AddConscript: компонент для додавання нових призовників, за допомогою інтерфейсу AddConscriptInterface.
- Conscripts: компонент, що відповідає за перегляд записів в таблиці призовників, завдяки інтерфейсу ConscriptsInterface.
- Filter: компонент для фільтрація даних в таблиці призовників, за допомогою інтерфейсу FilterInterface.
- ReportForm: компонент, відповідальний за формування звітів, в інтерфейсі ReportInterface.
- About: компонент, що надає інформацію про систему, через інтерфейс AboutInterface

Conscript.exe залежить від цих компонентів, використовуючи їхні інтерфейси для виконання відповідних операцій.

Файли ресурсів, зокрема конфігураційні параметри, налаштування кольорів та базу даних для налагодження представлені у вигляді артефактів. Вони використовуються виконуваним компонентом conscript.exe, що свідчить про його роль як центрального контролера ресурсів та логіки.

Ця діаграма компонентів дає змогу чітко уявити логічну структуру системи обліку призовників, виявити залежності між модулями та визначити межі відповідальності кожного компонента.

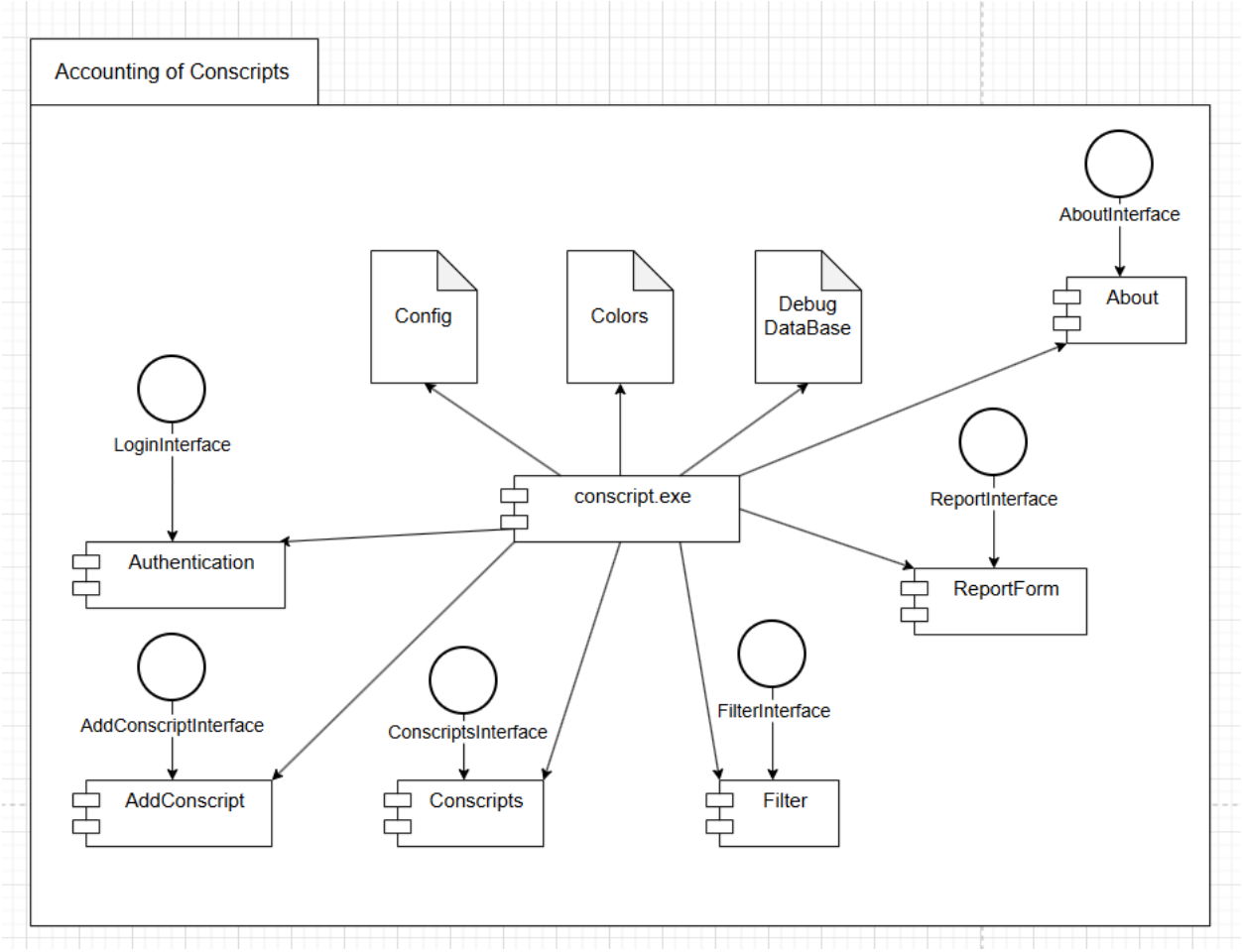


Рис. 10. Діаграма компонентів

3. РОЗРОБКА ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вибір СУБД

Для функціонування програмного забезпечення, розробленого в рамках системи обліку призовників, необхідне надійне зберігання, обробка та ефективне управління даними. Враховуючи вимоги до цілісності, масштабованості та надійності, як основну систему управління базами даних (СУБД) було обраном Microsoft SQL Server – потужну реляційну СУБД, орієнтовану на корпоративне використання.

Реляційна база даних – це тип баз даних, що базується на реляційній моделі, де дані організовано у вигляді взаємопов’язаних таблиць. Кожна таблиця містить записи у формі рядків, а кожен стовпець відповідає певному атрибуту або типу даних. Такий підхід забезпечує логічну організацію даних, їхню цілісність та можливість встановлення зв’язків між таблицями через первинні та зовнішні ключі.

Серед переваг реляційних баз даних можемо виділити:

1. Чітка структура даних, що значно спрощує процес зберігання та обробки інформації.
2. Незалежність даних – зміна структури таблиць може не потребувати зміни прикладного коду.
3. Цілісність інформації, що досягається шляхом визначення ключів, обмежень та зв’язків між таблицями.
4. Гнучкість запитів – мова SQL дозволяє формулювати запити будь-якої складності для аналізу даних.

Перед створенням бази даних особливу увагу було приділено моделювання – були визначені основні сутності, їх атрибути та

взаємозв’язки, що дозволило уникнути проблем у майбутньому при реалізації запитів, фільтрації даних та формуванні звітів.

Основні переваги Microsoft SQL Server у рамках цього проєкту:

1) Інтеграція в екосистему Microsoft – ідеально підходить для проєктів, розроблених у середовищі Visual Studio, особливо на C# із застосуванням Windows Forms або WPF.
2) Підтримка транзакцій – гарантує цілісність даних через реалізацію властивостей ACID (атомарність, узгодженість, ізольованість, довговічність).
3) Потужні засоби безпеки – система забезпечує шифрування данихЮ автентифікацію, авторизацію, журналювання.
4) Масштабованість та продуктивність – підходить як для локальних, так і для розподілених систем із високим навантаженням.
5) Інструменти резервного копіювання та відновлення – вбудовані механізми дозволяють швидко відновити дані у разі збоїв.
6) Зручне адміністрування через SSMS – дозволяє створювати таблиці, збережені процедури, представлення, індекси, тригери тощо без написання великого обсягу коду вручну.
7) Можливість використовувати як LocalDB прямо всередині проєкту в середовищі Microsoft Visual Studio.

Серед недоліків Microsoft SQL Server можна зазначити:

1) Високі системні вимоги – для стабільної роботи потрібно достатньо ресурсів (процесор, пам’ять, дисковий простір).
2) Ліцензійна модель – повна комерційна версія SQL Server потребує оплати, хоча для навчальних цілей доступна безкоштовна версія SQL Server Express з деякими обмеженнями.
3) Складність конфігурації – початкове налаштування SQL Server та безпечного з’єднання може вимагати досвіду.

4) Непридатність для мобільних платформ – СУБД не є вбудованою, як SQLite, тому не може бути використана безпосередньо в мобільних додатках.

Розгляд альтернатив:

Під час вибору бази даних розглядалися й інші варіанти, які були відхилені з таких причин:

1. SQLite – вбудована база даних із мінімальними вимогами до ресурсів, однак має обмежену підтримку транзакцій, не підходить для складних багатокористувацьких систем.
2. PostgreSQL – потужна об'єктно-реляційна СУБД з відкритим кодом, але має складніше адміністрування у Windows-середовищі, порівняно з MS SQL Server, а також вимагає окремих клієнтів для керування (pgAdmin).
3. Firebase Realtime Database – хмарне рішення, не підходить для локального розгортання і не підтримує повноцінну роботу із реляційними зв’язками.
4. MongoDB – документо-орієнтована NoSQL СУБД, що не відповідає реляційній моделі, не має повноцінної підтримки транзакцій, що важливо для критичних систем обліку.
5. Realm – зручна для мобільних додатків, але не підтримує SQL, має обмежену гнучкість у структурі даних.

Таким чином, Microsoft SQL Server було обрано як основну СУБД проєкту на основі ряду переваг – надійність, масштабованість, повна підтримка SQL, зручне адміністрування через SSMS, а також інтеграція з платформою розробки C#. Це рішення ідеально відповідає вимогам інформаційно-управляючої системи обліку призовників, яка потребує високого рівня безпеки, логічної структури та ефективного доступу до даних.

3.2 Розробка об’єктів бази даних

Проєктування таблиць бази даних

Табл. 2.

Таблиця Users:

Поле	Тип	Ключ	Опис
user_id	INT AUTO_INCREMENT	PK	ID користувача
username	VARCHAR(50)		Логін
password	VARCHAR(50)		Хеш пароля

Табл. 3.

Таблиця Order:

Поле	Тип	Ключ	Опис
id	ID AUTO_INCREMENT	PK	ID призовника
surname	CHAR(20)		Прізвище
name	CHAR(20)		Ім’я
second_name	CHAR(20)		По-батькові
birthday	DATETIME		Дата народження
place	CHAR(50)		Адреса проживання
health	CHAR(20)		Стан здоров’я
suitability	CHAR(20)		Придатність до служби
marital_status	CHAR(30)		Сімейний стан
education	CHAR(50)		Освіта

Створення бази даних

Для створення бази даних та її об’єктів було використано мову SQL. Це стандартна мова запитів, яка необхідна для взаємодії з реляційними базами даних. За допомогою неї можна створювати, управляти та модифікувати дані в БД. На рис. 10. Вказано код створення даної бази даних для ПЗ обліку призовників.

```
CREATE DATABASE Database
```

Рис. 10. Код створення бази даних

Код SQL для створення таблиць

```
CREATE TABLE [dbo].[users] (
   [id]       INT          IDENTITY (1, 1) NOT NULL,
   [username] NVARCHAR (50) NOT NULL,
   [password] NVARCHAR (50) NOT NULL,
   PRIMARY KEY CLUSTERED ([id] ASC)
);
CREATE TABLE [dbo].[order] (
   [Id]            INT        NOT NULL,
   [surname]       NCHAR (20) NULL,
   [name]          NCHAR (20) NULL,
   [second_name]    NCHAR (20) NULL,
   [birthday]      DATETIME   NULL,
   [place]         NCHAR (50) NULL,
   [health]        NCHAR (20) NULL,
   [suitability]    NCHAR (20) NULL,
   [marital_status] NCHAR (30) NULL,
   [education]     NCHAR (50) NULL,
   PRIMARY KEY CLUSTERED ([Id] ASC)
);
```

Діаграма структури БД

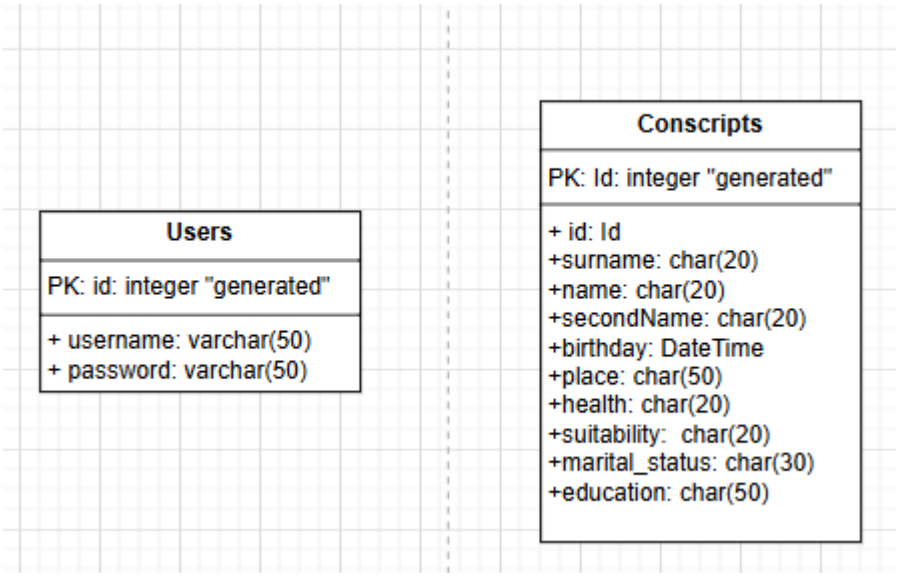


Рис. 12. ER-діаграма

Створення користувачів

Система обліку призовників передбачає механізм автентифікації користувачів для забезпечення контрольованого доступу до функціоналу та даних. Це дозволяє гарантувати, що лише авторизований персонал матиме змогу працювати з конфіденційною інформацією призовників. Процес створення користувачів є початковим та критично важливим етапом налаштування системи після її розгортання.

В даній системі всі користувачі мають однаковий рівень доступу після успішної автентифікації. Тобто не має як такого розмежування на адміністраторів, операторів тощо. Будь-який користувач, що успішно ввійшов до системи, матиме доступ до всіх доступних функцій, таких як додавання, перегляд, редагування даних призовників та генерація звітів.

Створення нового запису користувача в системі обліку призовників відбувається шляхом додавання нового запису до таблиці Users у базі даних Microsoft SQL Server.

3.2 Визначення з інструментами для створення програмного забезпечення системи обліку призовників

Для створення програмного забезпечення інформаційно-управляючої системи обліку призовників було обрано сучасні інструменти розробки, які забезпечують зручність реалізації, масштабованість, підтримку архітектурних патернів, а також ефективну інтеграцію з базою даних. Вибір кожного інструмента обґрунтовано вимогами до функціональності, безпеки, стабільності та зручності у супроводі проєкту.

1. Мова програмування C#

Основною мовою розробки було обрано C# (C-Sharp), оскільки вона є об’єктно-орієнтованою, має високу продуктивність і активно використовується в корпоративних системах. Завдяки платформі .NET C# забезпечує зручну роботу з базами даних, підтримує сучасні архітектурні підходи (MVVM, MVC) та має великий набір бібліотек для реалізації графічного інтерфейсу, обробки даних, звітів, логування тощо.

2. Середовище розробки: Microsoft Visual Studio 2022

Розробка програмного забезпечення здійснювалась у середовищі Visual Studio 2022, яка надає потужні інструменти для проєктування форм, налагодження коду, підключення бази даних, керування залежностями через NuGet, а також підтримує Git для керування версіями. Visual Studio є зручним середовищем для реалізації WinForms-проєктів, які були обрані як основа для клієнтської частини системи

3. Графічний інтерфейс: Windows Forms

Для реалізації користувацького інтерфейсу було обрано Widows Forms (WinForms) – перевірену технологію побудови GUI-додатків на платформі Windows. WinForms дозволяє швидко створювати інтерфейс з використанням

конструкторів форм, що значно пришвидшує розробку та полегшує тестування. Крім того, підтримуються обробка подій, валідація введення, робота з таблицями, діаграмами тощо.

4. Архітектура MVVM (Model-View-ViewModel)

Архітектурна модель MVVM була обрана з метою чіткого розмежування логіки представлення, бізнес-логіки та доступу до даних. Це дозволяє:

- Зробити програму більш масштабованою та зручною для тестування.
- Зменшити зв’язність між компонентами.
- Спростити розширення та підтримку системи.

5. База даних: Microsoft SQL Server

Для збереження та обробки даних була обрана Microsoft SQL Server – реляційна СУБД яка забезпечує високу продуктивність, підтримує транзакції, індекси, збережені процедури, функції, обмеження цілісності тощо. Робота з БД реалізована через підключення SQL Server до застосунку за допомогою бібліотек System.Data.SqlClient або ORM (у разі такої потреби).

6. Інструмент адміністрування бази даних: SQL Server Management Studio (SSMS)

Для моделювання та адміністрування структури бази даних використовується SSMS – офіційне середовище для роботи з Microsoft SQL Server. Воно дозволяє створювати таблиці, встановлювати зв’язки між ними, писати SQL-запити, реалізовувати збережені процедури та перевіряти виконання запитів.

7. Інші додаткові інструменти:

Draw.io / PlantUML – для створення UML-діаграм (діаграми класів, прецедентів, компонентів, пакетів тощо)

Таким чином, використаний мною стек технологій дозволяє ефективно реалізувати програмне забезпечення, що відповідає вимогам до автоматизованої системи обліку призовників, зручно розширюється, підтримується та забезпечує високий рівень надійності і зручності для кінцевого користувача.

3.3 Створення алгоритмів та програмування програмних модулів

Щоб краще зрозуміти алгоритми дій системи було змодельовано основні бізнес процеси у вигляді блок-схеми. Тому що завдяки даної діаграми можна візуалізувати алгоритми роботи модуля програми та виявити помилки та проблеми логіки системи. На Рис. 13. Зображений алгоритм авторизації працівника ТЦК в системі.

Для початку роботи з програмним забезпечення працівник має ввести своє ім’я та пароль, після перевірки актуальності введених даних, система або надає доступ, або видає повідомлення про неуспішну авторизацію.

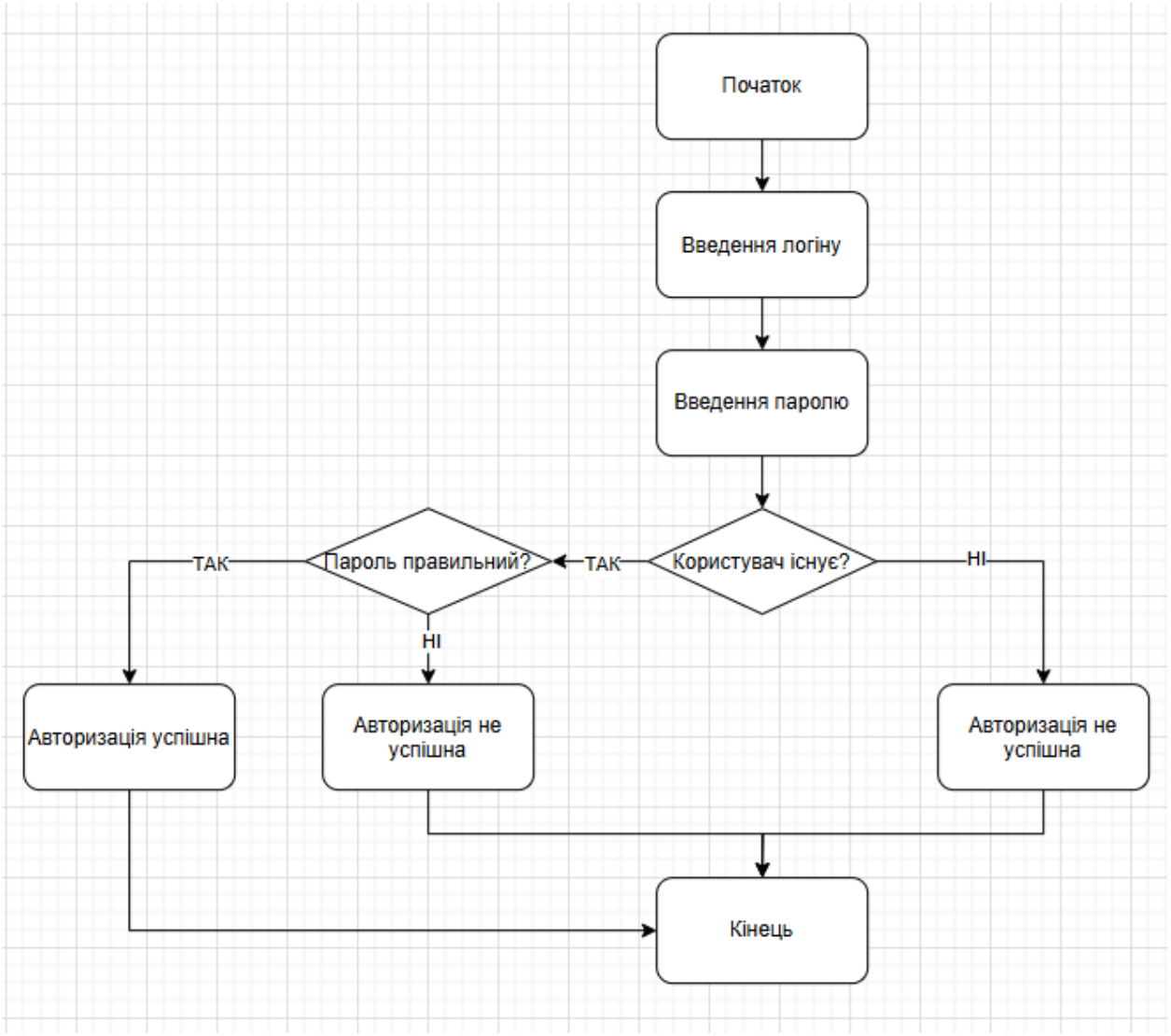


Рис. 13. Авторизація в системі

Переглянувши Рис. 14 можна побачити частину коду для авторизації працівника в системі.

```
string username = textBox1.Text.Trim();
        string password = textBox2.Text.Trim();
        string connectionString = @"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\Datab
ase.mdf;Integrated Security=True";
        using (SqlConnection conn = new SqlConnection(connectionString))
        {
            try
            {
                conn.Open();
                string query = "SELECT COUNT(*) FROM users WHERE
username = @username AND password = @password";
                using (SqlCommand cmd = new SqlCommand(query, conn))
                {
                    cmd.Parameters.AddWithValue("@username", username);
                    cmd.Parameters.AddWithValue("@password", password);
                    int count = (int)cmd.ExecuteScalar();
                    if (count > 0)
                    {
                        MessageBox.Show("Авторизація в системі \"Оберіг\"
успішна!");
                        MainWindow main = new MainWindow(username, password);
                        main.Show();
                        this.Hide();
                    }
                    else
                    {
                        MessageBox.Show("Невірне ім’я користувача або пароль.");
```

Рис. 14. Фрагмент коду перевірки наявності облікового запису та правильності введення даних користувача

Ще одним прикладом є алгоритм що реалізує генерування звіту та подальшому зберіганні його в форматі PDF (рис. 15). Спочатку система отримує доступ до нашої таблиці призовників, користувач задає потрібний йому фільтр, потім система перевіряє чи є записи за заданим фільтром, отримує потрібні дані, створює звіт згідно обраному фільтру та зберігає в форматі PDF.

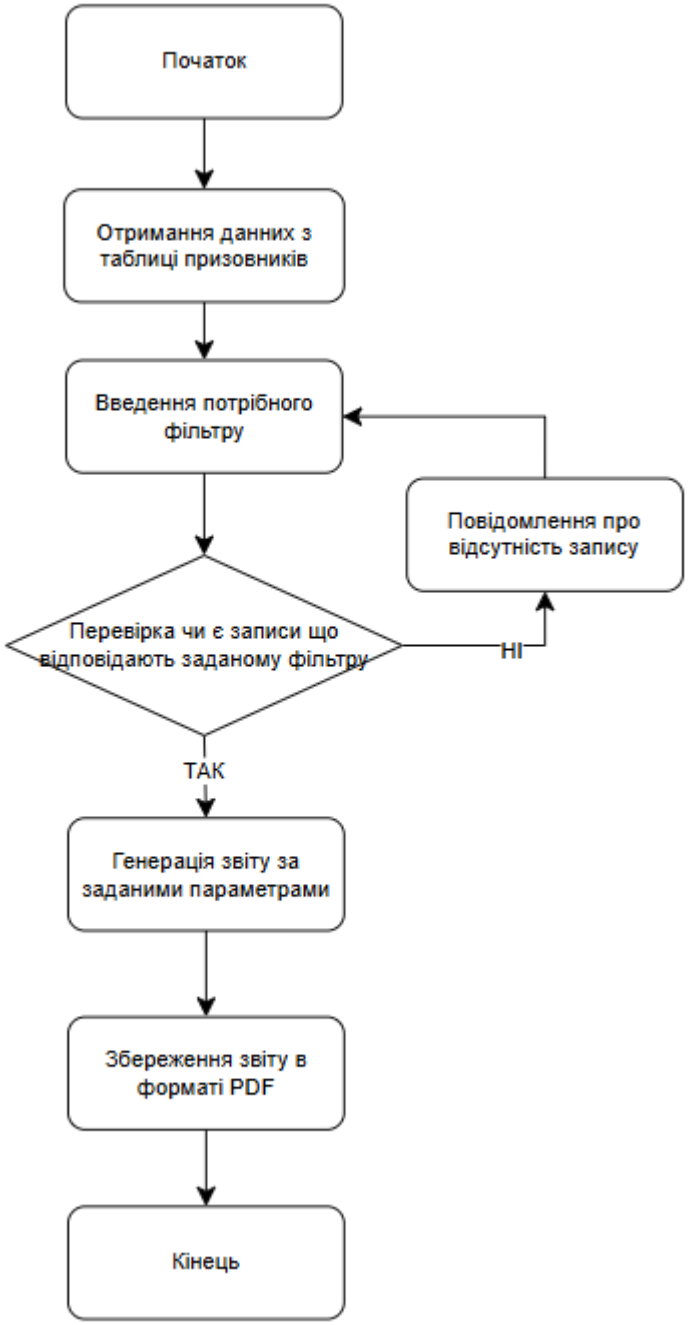


Рис. 15. Алгоритм формування звітів

На рис. 16. можемо ознайомитись з фрагментом коду функції фільтрування запиту на генерацію звіту.

```
string filterType = comboBoxFilterType.SelectedItem.ToString();
        string filterValue = textBoxFilterValue.Text.Trim();
        if (string.IsNullOrEmpty(filterValue))
        {
            MessageBox.Show("Введіть значення для фільтра!");
            return;
        }
        DataRow[] filteredRows;
        try
        {
            switch (filterType)
            {
                case "Місто":
filteredRows = databaseDataSet.order.Select($"place LIKE '%{filterValue}%'");
                    break;
                case "Рік народження":
                    filteredRows = databaseDataSet.order
.Where(r => ((DateTime)r["birthday"]).Year == int.Parse(filterValue))
                        .ToArray();
                    break;
                case "Придатність до служби":
filteredRows = databaseDataSet.order.Select($"suitability = '{filterValue}'");
                    break;
```

```
case "Сімейний стан":
filteredRows = databaseDataSet.order.Select($"marital_status = '{filterValue}'");
                    break;
                case "Освіта":
filteredRows = databaseDataSet.order.Select($"education = '{filterValue}'");
                    break;
                default:
                    filteredRows = databaseDataSet.order.Select();
                 break;
            }
        }
catch (Exception ex)
        {
            MessageBox.Show("Помилка фільтрації: " + ex.Message);
            return;
        }
        if (filteredRows.Length == 0)
        {
            MessageBox.Show("Не знайдено записів за заданим фільтром.");
            return;
        }
```

Рис. 16. Фрагмент коду реалізації фільтрації записів для створення звіту

4. РЕКОМЕНДАЦІЇ ЩОДО ЕКСПЛУАТАЦІЇ ТА ВПРОВАДЖЕННЯ СИСТЕМИ

4.1. Проведення тестувань системи

Тестування є дуже важливою складовою процесу розробки програмного забезпечення. Завдяки тестуванню можливо перевірити коректність функціонування інформаційної системи та забезпечити її відповідність технічному завданню. Існує багато видів тестування, що відрізняються методами реалізації, але всі вони мають на меті перевірку працездатності, надійності, безпеки та якості програмного продукту.

Серед основних видів тестування, які можуть бути застосовані для подібних систем, виділяють:

1) Функціональне тестування – перевірка реалізації функціоналу відповідно до вимог, наприклад, правильність реєстрації призовника, генерації звітів, авторизації тощо.

2) Навантажувальне тестування – оцінка продуктивності при роботі з великою кількістю записів у базі даних (наприклад, при обробці тисяч записів призовників).

3) Перевірка на помилки (debugging) – пошук і виправлення помилок у коді, які можуть викликати некоректну поведінку системи.

4) Регресійне тестування – перевірка після внесення змін у систему (наприклад, після оновлення модуля авторизації або фільтрації даних), щоб переконатися, що інші функції працюють стабільно.

5) Автоматизоване тестування – використання інструментів для

автоматизації перевірок, що зменшує час на тестування повторюваних операцій.

6) Юніт-тестування – перевірка окремих компонентів, наприклад методів роботи з базою даних або логіки авторизації користувача.

7) Верифікація та валідація – підтвердження відповідності розробленого ПЗ вимогам, викладеним у технічному завданні.

8) Тестування безпеки – перевірка стійкості до несанкціонованого доступу, зокрема захисту даних призовників.

9) Тестування з використанням реальних даних – моделювання реальної роботи системи із залученням справжніх (або максимально наближених до них) даних.

10) Тестування користувацького інтерфейсу – перевірка коректного відображення форм, елементів керування, навігації та взаємодії користувача із системою.

11) Тестування сумісності – перевірка роботи на різних версіях Windows, зокрема Windows 10 і 11.

12) Тестування відмовостійкості – перевірка поведінки системи у випадках помилок, наприклад, відсутності підключення до бази даних чи помилки введення даних користувачем.

Для тестування програмного забезпечення системи обліку призовників було обрано два основних типи тестування: юніт-тестування та тестування користувацького інтерфейсу. Вони дозволяють перевірити як внутрішню логіку роботи окремих методів, так і зручність та коректність взаємодії з користувачем.

Переваги юніт-тестування:

1. Виявлення помилок – юніт-тести дозволяють оперативно виявити помилки в логіці роботи окремих методів (наприклад, перевірка автентифікації користувача або створення нового запису призовника).

2. Безпечне внесення змін – наявність тестів дозволяє змінювати код без ризику порушення роботи системи.

3. Документування логіки – тести описують очікувану поведінку компонентів, що виконує роль технічної документації.

4. Покращення архітектури – створення тестів вимагає модульності та чіткого розділення обов’язків між компонентами системи.

Переваги тестування користувацького інтерфейсу:

1. Перевірка інтерфейсу – тестування дозволяє виявити помилки у відображенні форм, розміщенні кнопок, написів тощо.

2. Оцінка користувацького досвіду – виявлення проблем з логікою переходів між формами, зручністю заповнення даних.

3. Підвищення якості системи – усунення недоліків інтерфейсу до впровадження програмного забезпечення.

На рис. 14 представлений приклад юніт-тесту для методу авторизації користувача, логіка якого була додана на рис. 10, та код реалізації показаний на рис. 11. Даний тест перевіряє, чи правильно система оброблює вхідні логін та пароль.

Юніт-тест можна умовно поділити на три етапи:

1. Arrange – ініціалізація тестових даних, таких як ім’я користувача та ї

пароль.

2. Act – виклик тестованого методу з підготовленими даними та збереження результату.

3. Assert – перевірка правильності отриманого результату, тобто чи повертає метод очікуване значення (наприклад, true у разі успішної авторизації)

```
// === Arrange ===
   string username = txtLogin.Text.Trim();
   string password = txtPassword.Text.Trim();
   string connectionString = @"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\Dat
abase.mdf;Integrated Security=True";
   using (SqlConnection conn = new SqlConnection(connectionString))
   {
      try
      {
         conn.Open();
         string query = "SELECT COUNT(*) FROM users WHERE username
= @username AND password = @password";
         using (SqlCommand cmd = new SqlCommand(query, conn))
         {
            // Підготовка параметрів запиту (частина Arrange)
            cmd.Parameters.AddWithValue("@username", username);
            cmd.Parameters.AddWithValue("@password", password);
            // === Act ===
            int count = (int)cmd.ExecuteScalar();
            // === Assert ===
            if (count > 0)
            {
```

```
MessageBox.Show("Авторизація в системі \"Оберіг\" успішна!");
MainWindow main = new MainWindow(username, password);
                main.Show();
                this.Hide();
            }
            else
            {
                MessageBox.Show("Невірне ім’я користувача або пароль.");
        }
}
```

Рис. 17. Юніт-тест методу authenticateUser, який використовується у системі обліку призовників.

Для тестування користувацького інтерфейсу як і під час юніт тесту я обрав форму логіну. На рис. 18. зображено вікно входу в систему

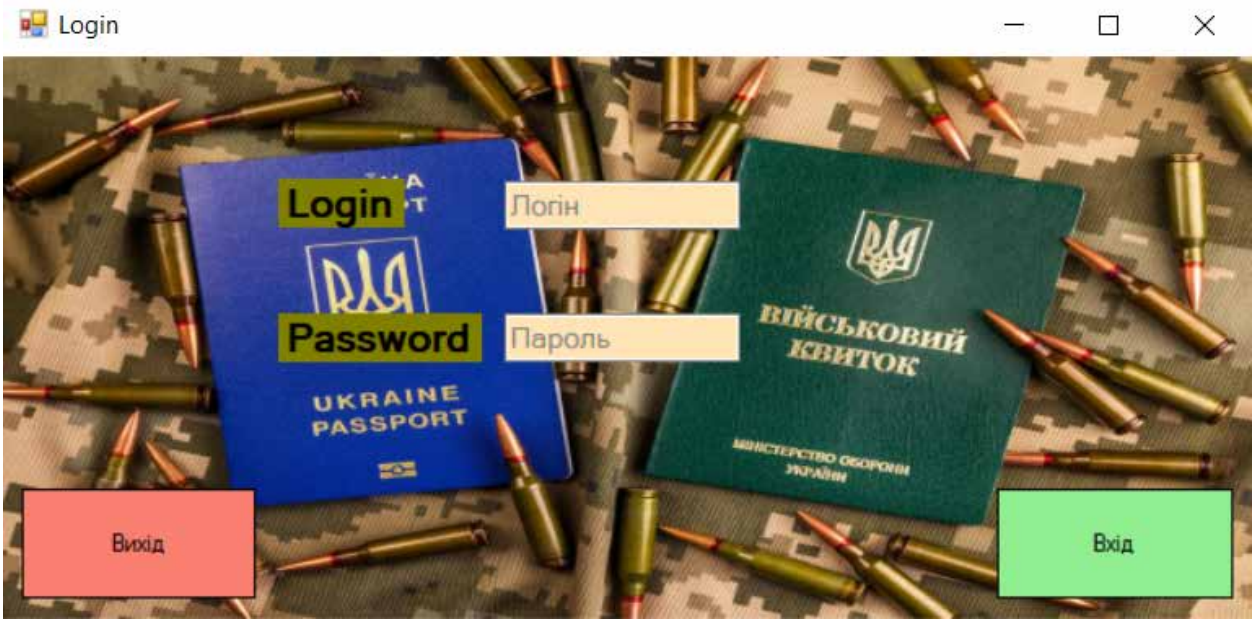


Рис. 18. Вікно авторизації в системі

Під час натискання на поле для введення логіну або пароля зникає фоновий надпис Логін або відповідно Пароль і ми можемо вписувати свої значення логіну та паролю. Рис. 19. вікно з обраним полем введення логін.

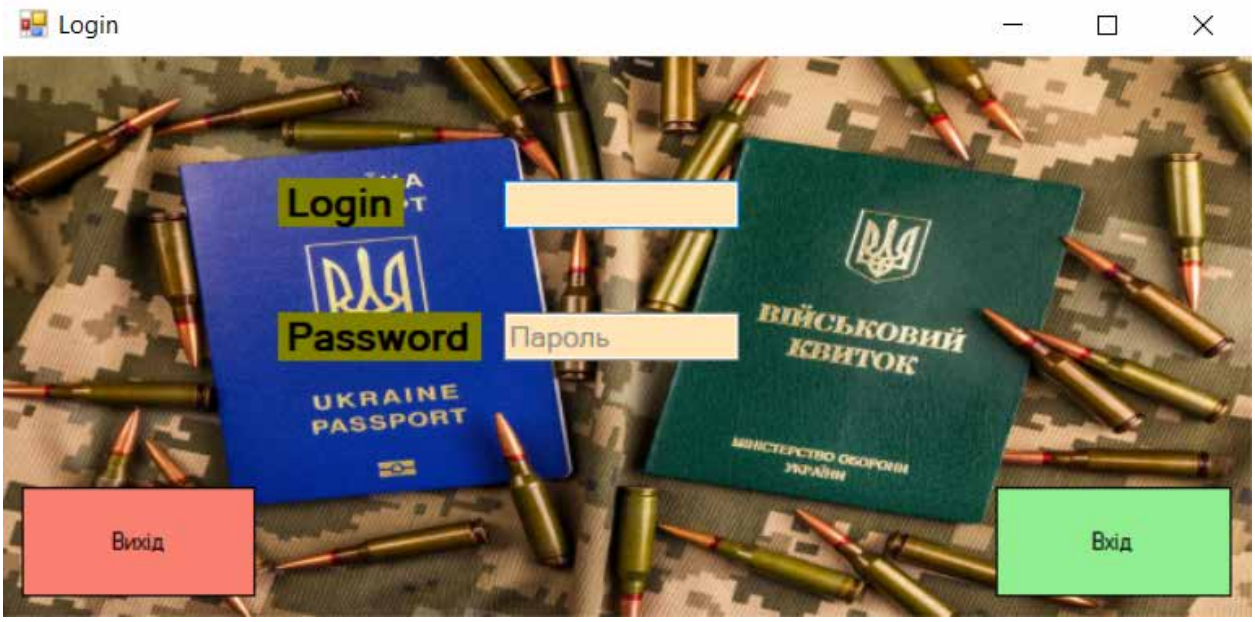


Рис. 19. Вікно входу до пз з обраним полем введення

При введенні паролю створено поверхневу маску що не дає можливості підгледіти ваш пароль в екрані, який ви вводите для авторизації в системі. Рис. 20 вікно авторизації з введеними даними для авторизації.

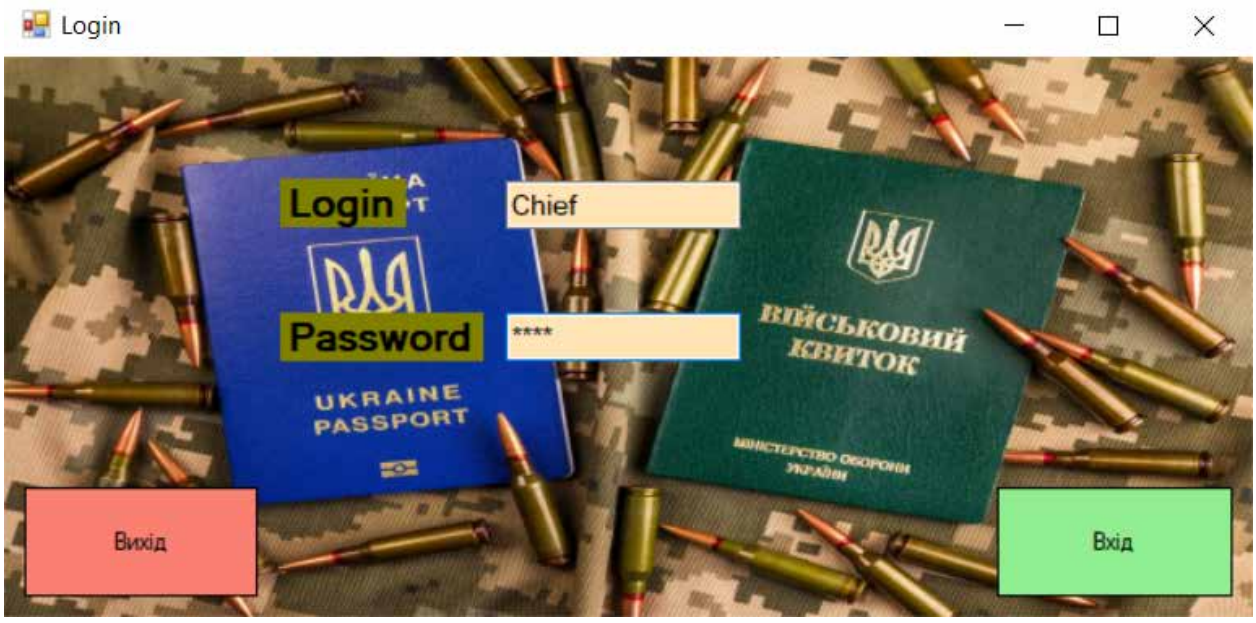


Рис. 20. Вікно авторизації з введеними даними

Якщо всі дані було введено вірно після натискання кнопки Вхід має виводитись повідомлення про успішну авторизацію в системі (рис. 21).

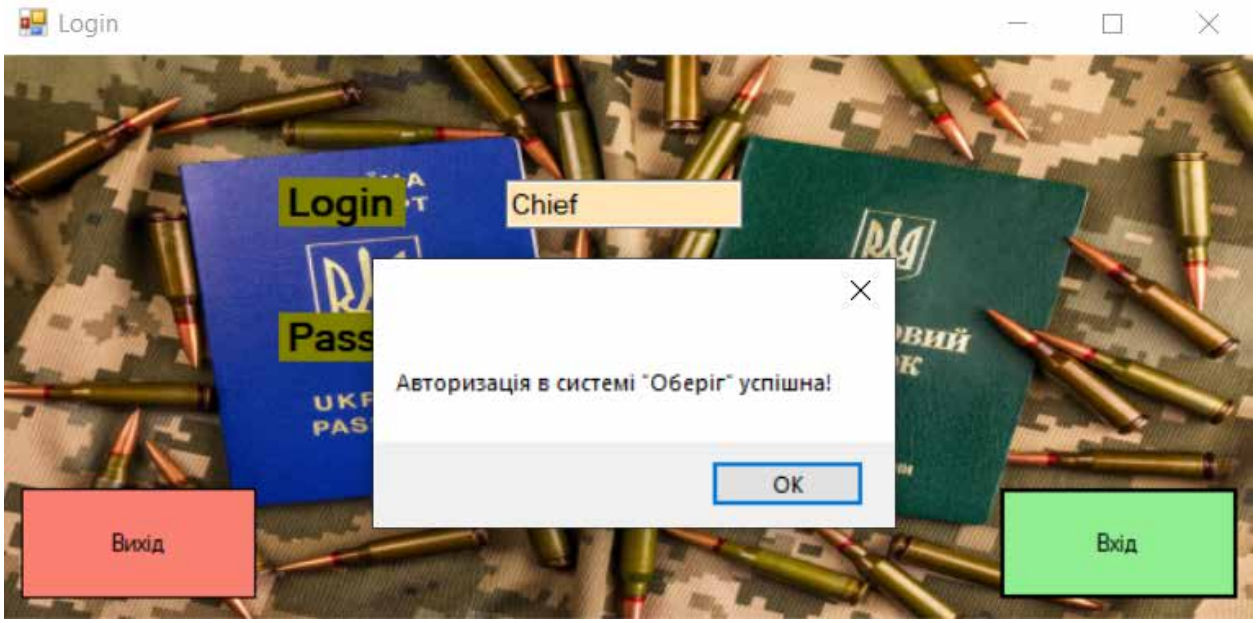


Рис. 21. Повідомлення про успішну авторизацію в системі

Якщо ми ввели невірні дані для входу в систему нам видає відповідне повідомлення про те що було невірно введені дані (рис. 22).

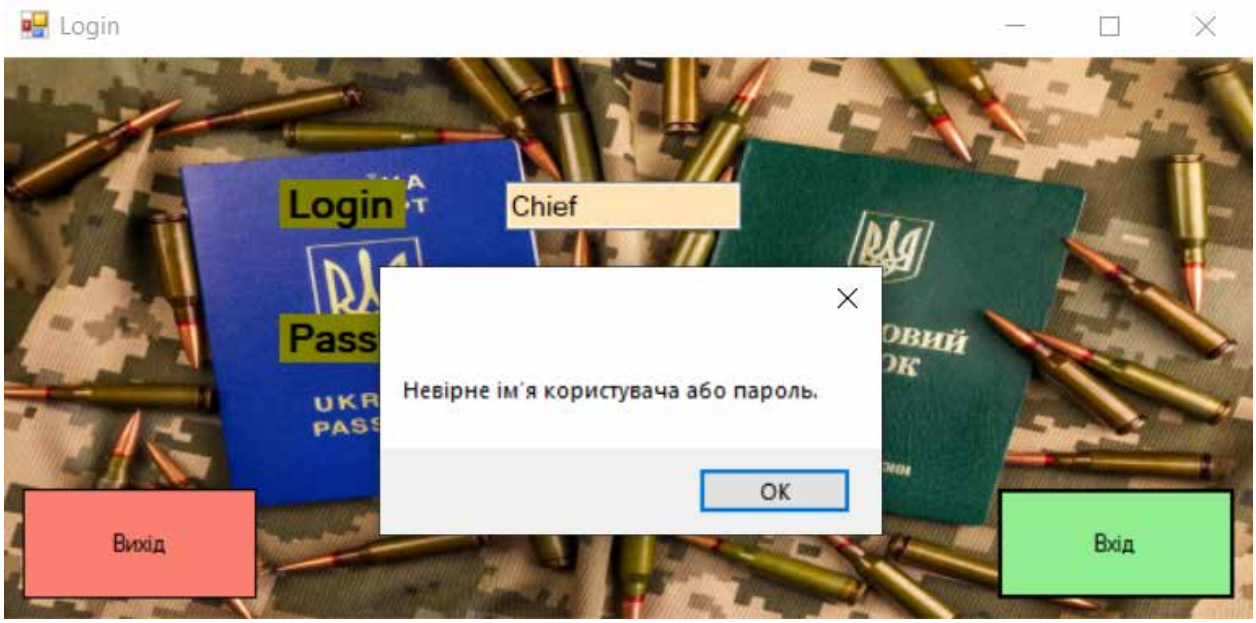


Рис. 22. Повідомлення про невірно введені дані

Якщо у нас всі дані були введені вірно і нам видало повідомлення про успішну авторизацію (рис. 21), після натискання на кнопку «ОК» нас перенесе на домашню сторінку нашого застосунку (рис. 23).

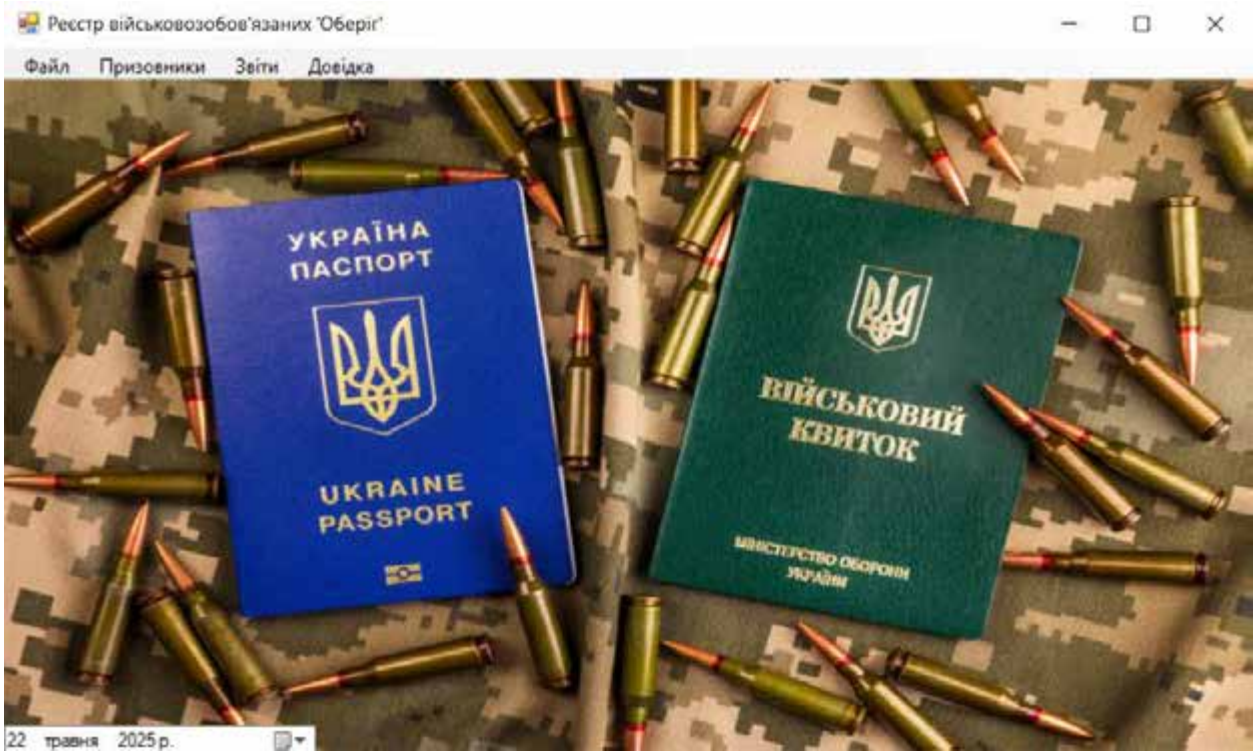


Рис. 23. Домашня сторінка застосунку

При натисканні на текстове меню «Файл» відкривається список доступних функцій в даному меню, в нашому випадку це «покинути сеанс» (рис. 24).

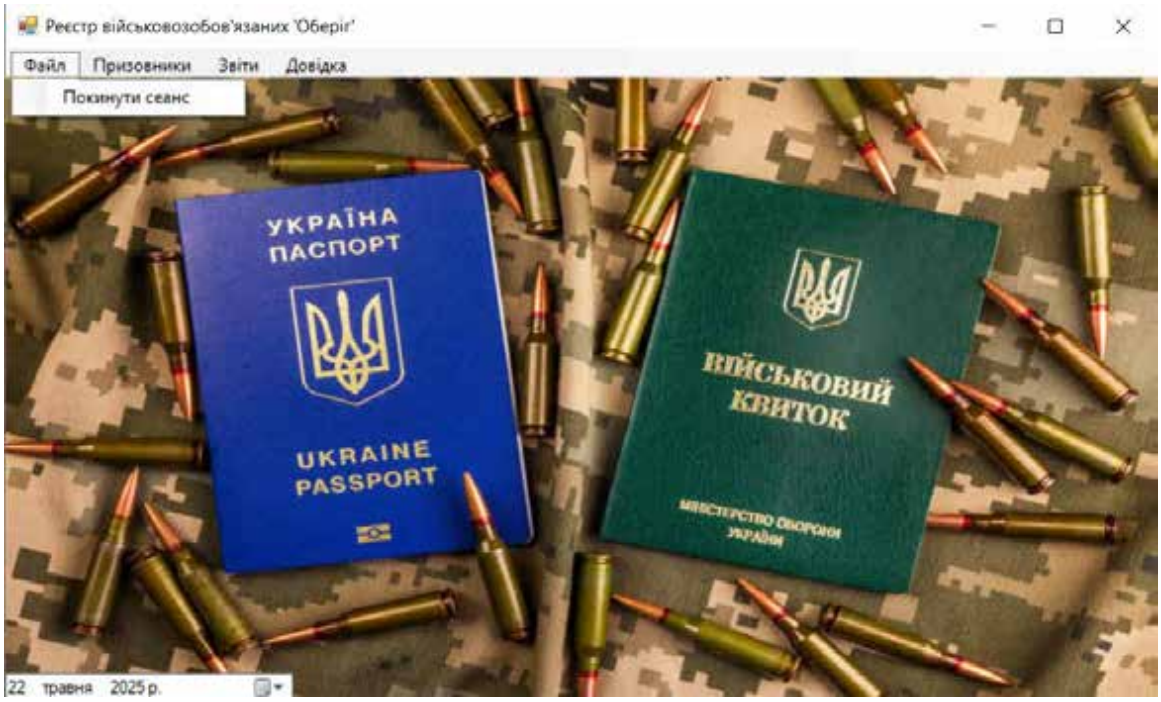


Рис. 24. Текстове меню «Файл»

При натисканні на текстове меню «Призовники» відкривається список доступних функцій в даному меню, в нашому випадку це додавання нових записів призовників, перегляд та пошук серед вже існуючих записів (рис. 25).

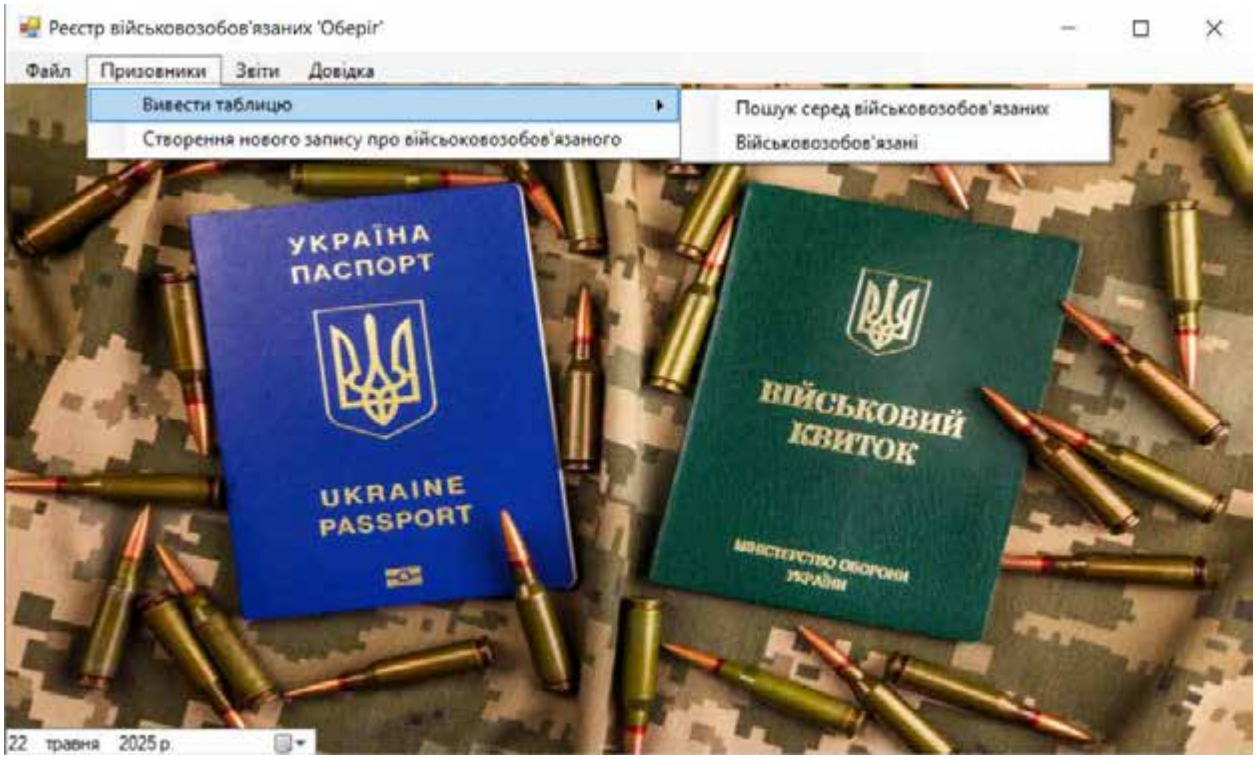


Рис. 25. Текстове меню «Призовники»

Після натискання на меню Звіти ми маємо доступ до функції створення звітів (рис. 26).

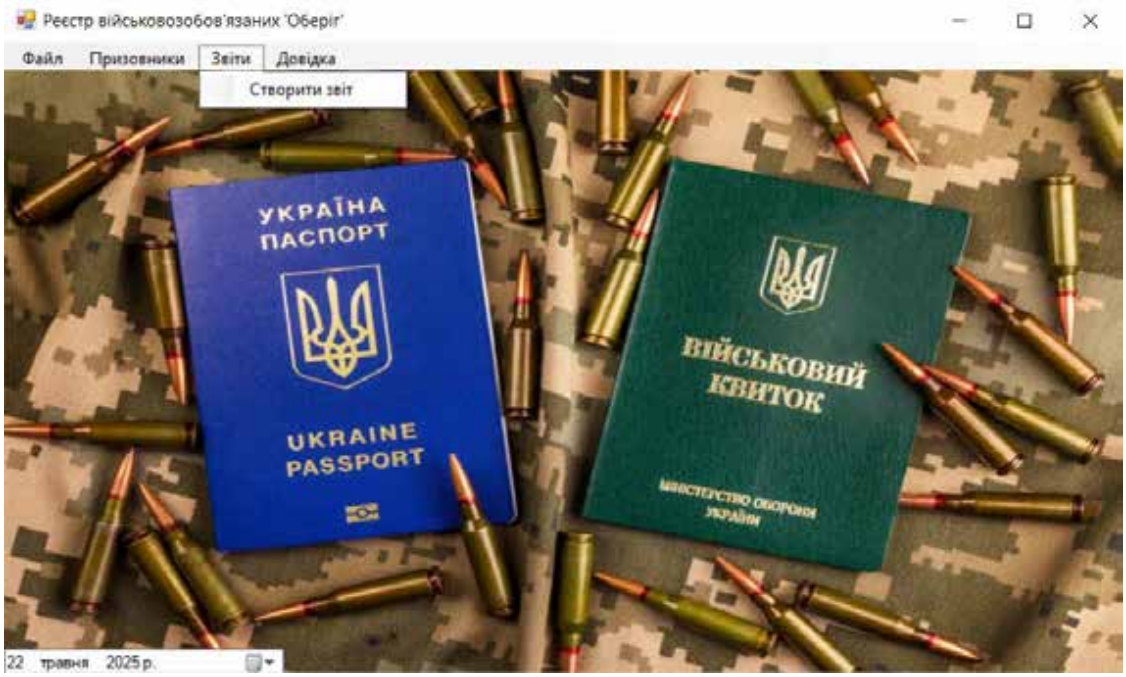


Рис. 26. Текстове меню «Звіти»

Останнім в списку текстових меню на головному екрані є Довідка, натиснувши на яке ми отримаємо доступ до кнопки «Про систему» (рис. 27).

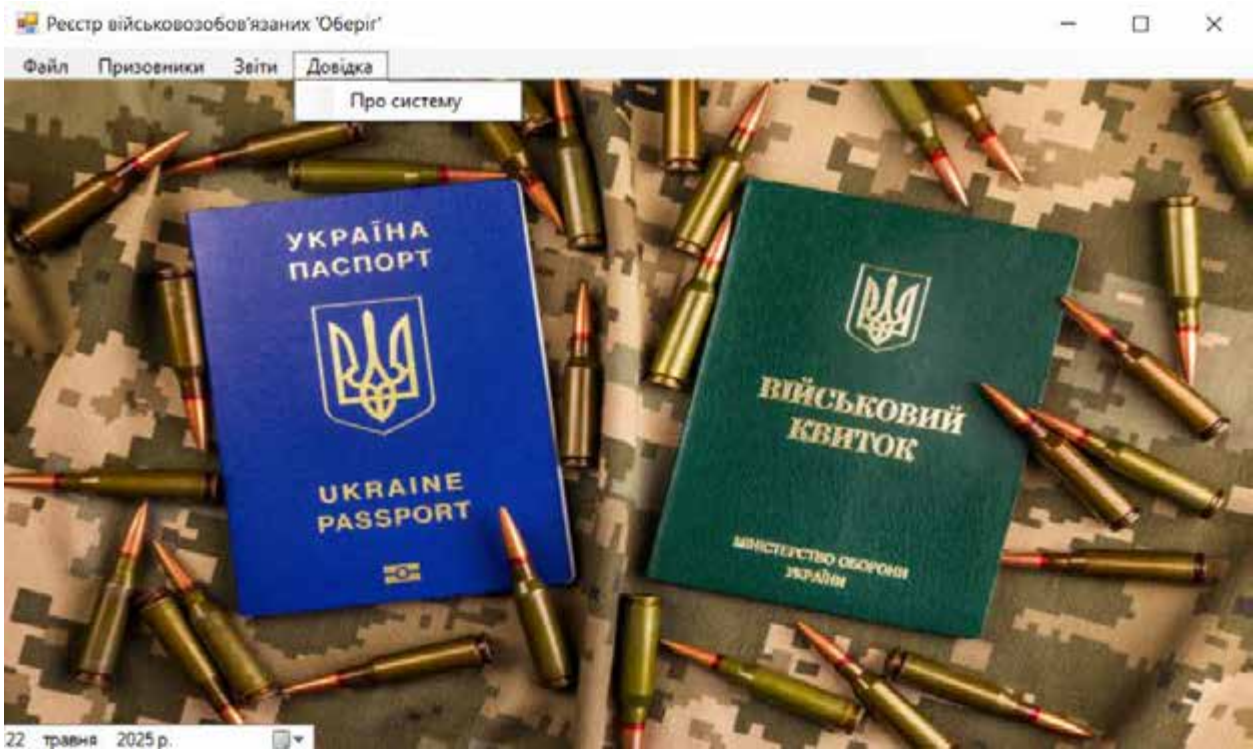


Рис. 27. Текстове меню «Про систему»

Якщо ми натиснемо на кнопку створення нового запису про військовозобов’язаного нам відкриється наступне вікно (рис. 28).

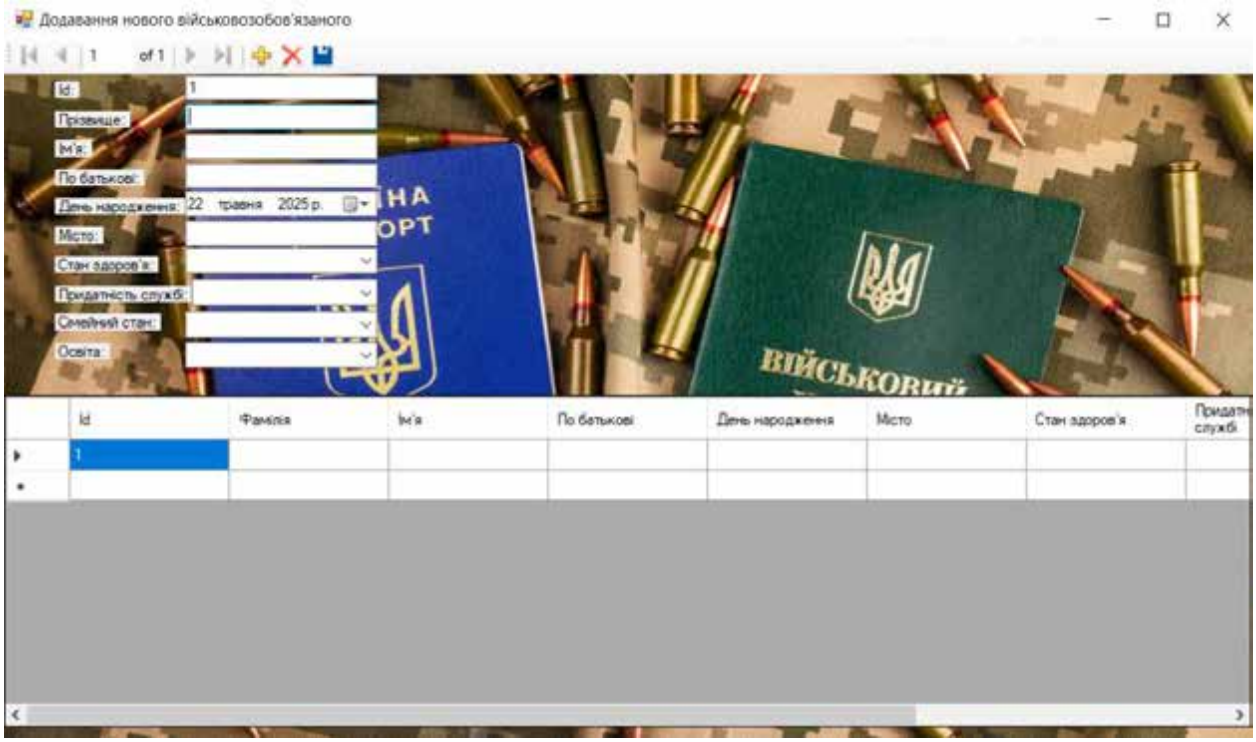


Рис. 28. Форма додавання нового призовника

Також в цьому ж вікні ми можемо відредагувати дані про раніше доданого призовника обравши його за допомогою стрілок зверху або написавши потрібний номер запису (рис. 29).

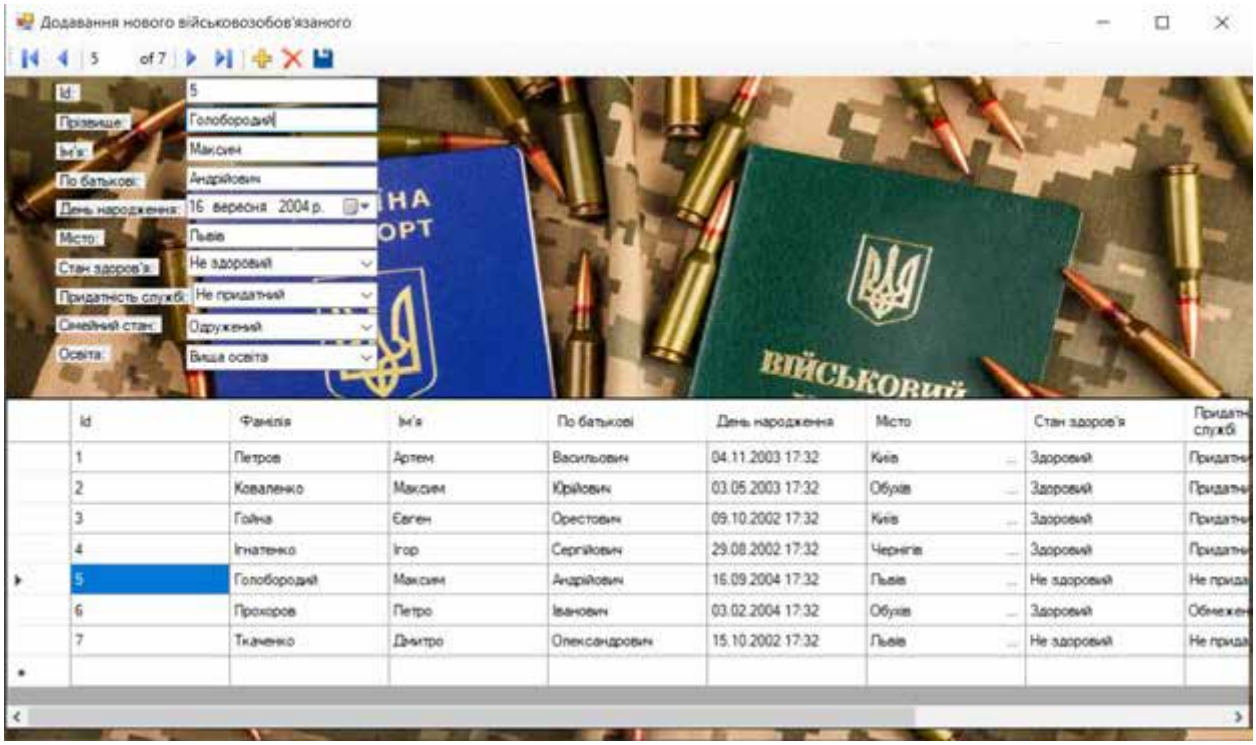


Рис. 29. Форма редагування даних призовників

Натиснувши на кнопку «Військовозобов’язані» нас переносить на форму в якій ми можемо переглянути повний список військовозобов’язаних внесених в нашу систему, також на цій формі можно провести збереження цієї таблиці в форматі PDF (рис. 30).

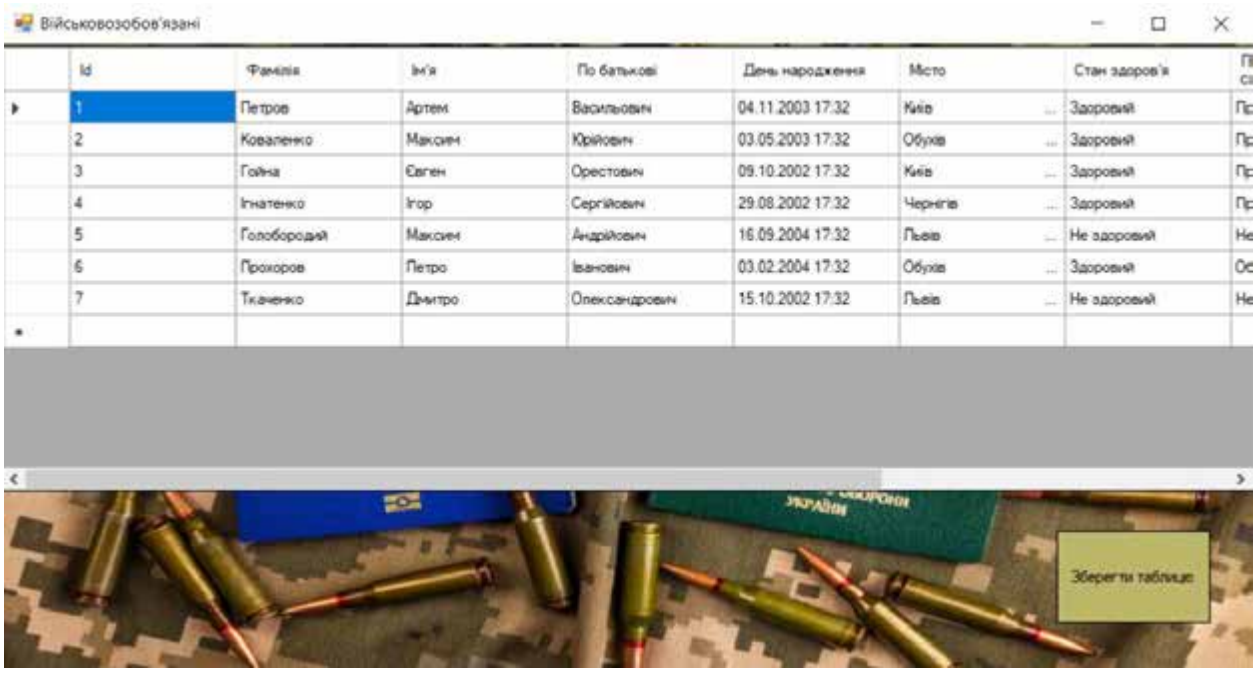


Рис. 30. Форма перегляду військовозобов’язаних

Якщо ми натиснемо на кнопку зберегти таблицю, нам запропонує обрати місце збереження файлу, та після чого ми отримаємо повідомлення про успішне збереження або про помилку створення файлу (рис. 31).

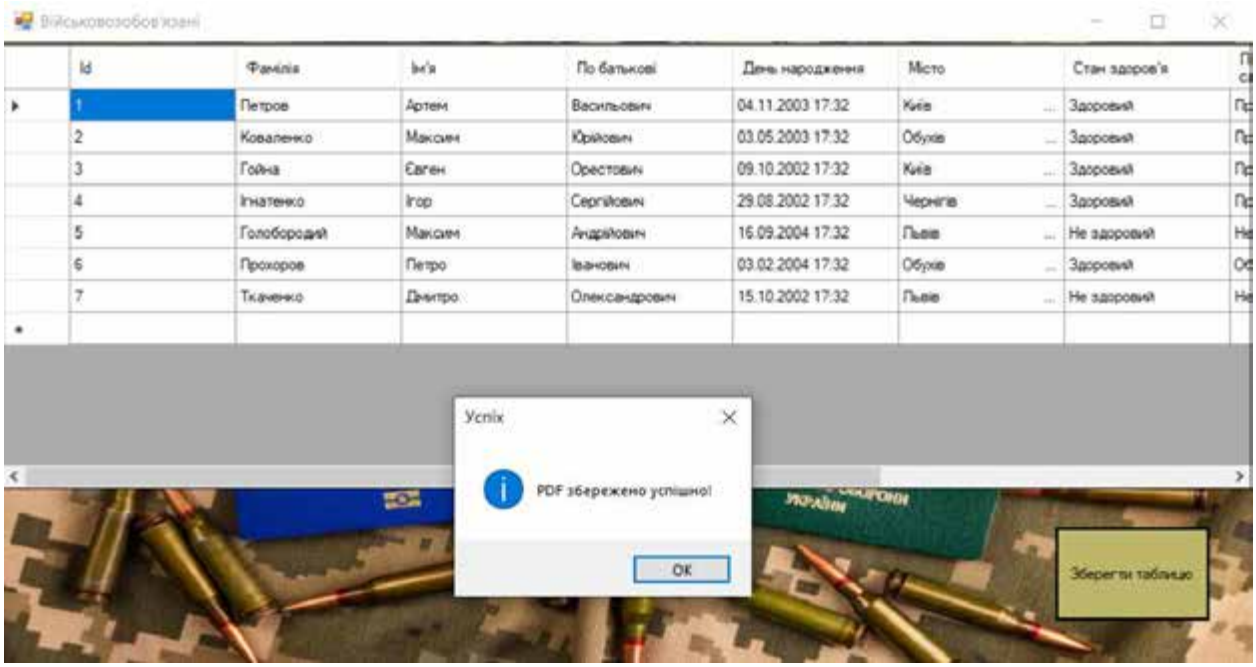


Рис. 31. Вікно інформування про успішне збереження файлу

Скориставшись пунктом пошук серед військовозобов’язаних ми можемо знайти записи про військовозобов’язаних з заданого міста (рис. 32).

Рис. 32. Вікно пошуку серед військовозобов’язаних

Задаємо запит про військовозобов’язаних з міста Львів і отримуємо дані про записи з даного міста (рис. 33).

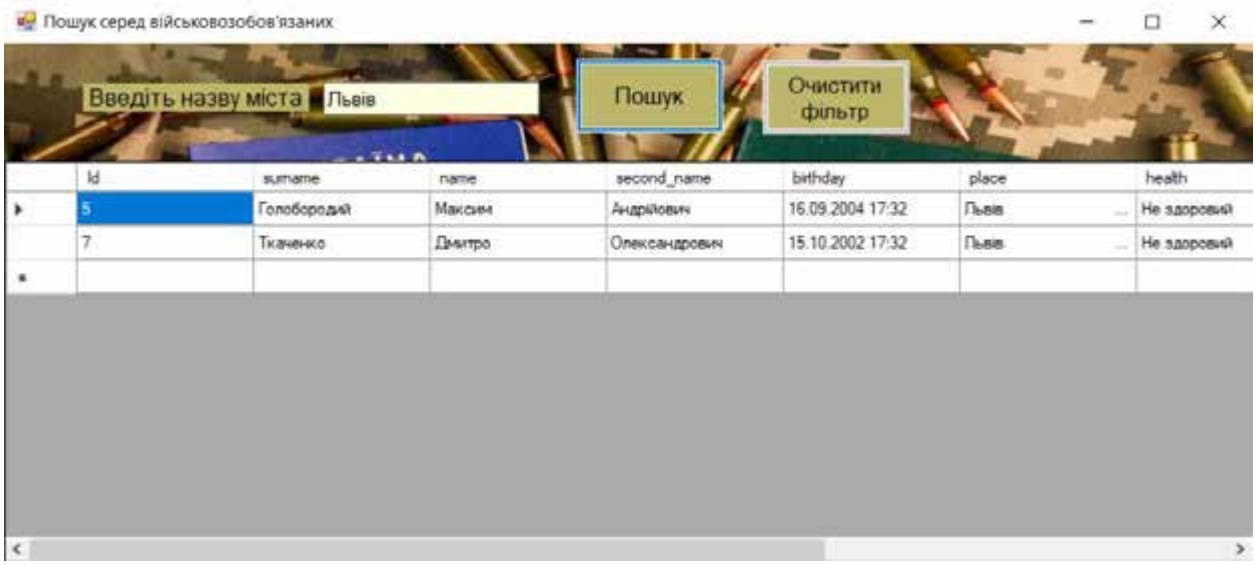
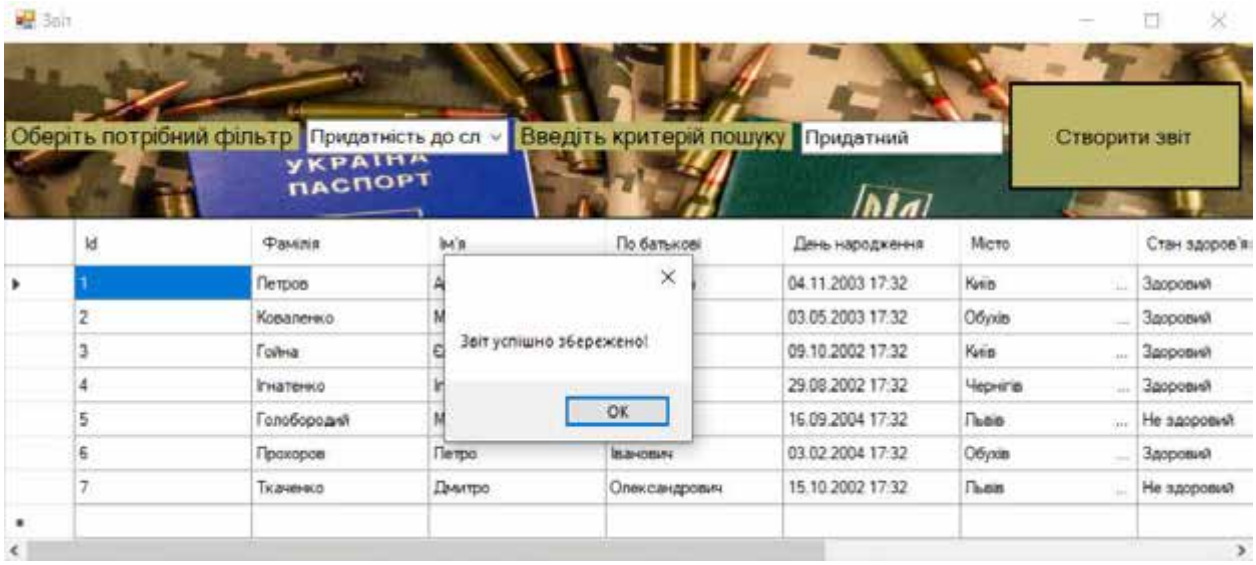


Рис. 33. Пошук серед військовозобов’язаних з міста Львів

Скориставшись пунктом меню «Звіти – Створити звіт» ми отримаємо можливість створити звіт про вибірку призовників за заданим фільтром який буде збережено в PDF форматі в обраному користувачем місцем на комп’ютері (рис. 34 та рис. 35).

Рис. 34. Вікно створення звітів



Id	Фамілія	Ім'я	По батькові	День народження	Місто	Стан здоров'я
1	Петров	А		04.11.2003 17:32	Київ	Здоровий
2	Коваленко	М		03.05.2003 17:32	Обухів	Здоровий
3	Гойна	Є		09.10.2002 17:32	Київ	Здоровий
4	Ігнатенко	І		29.08.2002 17:32	Чернігів	Здоровий
5	Голобородий	М		16.09.2004 17:32	Львів	Не здоровий
6	Прохоров	Петро	Іванович	03.02.2004 17:32	Обухів	Здоровий
7	Ткаченко	Дмитро	Олександрович	15.10.2002 17:32	Львів	Не здоровий

Рис. 35. Створили запит за фільтром придатності до служби, по критерію «придатний»

Результат формування звіту в форматі PDF збережений на комп'ютер (рис. 36).

Звіт про призовників

Фільтр: Придатність до служби = Придатний

Дата: 22.05.2025

ID	Прізвище	Ім'я	По батькові	Дата народження	Місто	Стан здоров'я	Придатність до служби	Сімейний стан	Освіта
1	Петров	Артем	Васильович	04.11.2003	Київ	Здоровий	Придатний	Не одружений	Професійно-технічна освіта
2	Коваленко	Максим	Юрійович	03.05.2003	Обухів	Здоровий	Придатний	Одружений	Вища освіта
3	Гойна	Євген	Орестович	09.10.2002	Київ	Здоровий	Придатний	Не одружений	Вища освіта
4	Ігнатенко	Ігор	Сергійович	29.08.2002	Чернігів	Здоровий	Придатний	Не одружений	Професійно-технічна освіта

Рис. 36. Звіт сформований за запитом по критерію «Придатний»

На останній вкладці ми можемо побачити інформацію про систему, в даній вкладці ми можемо дізнатись версію системи (рис. 37).

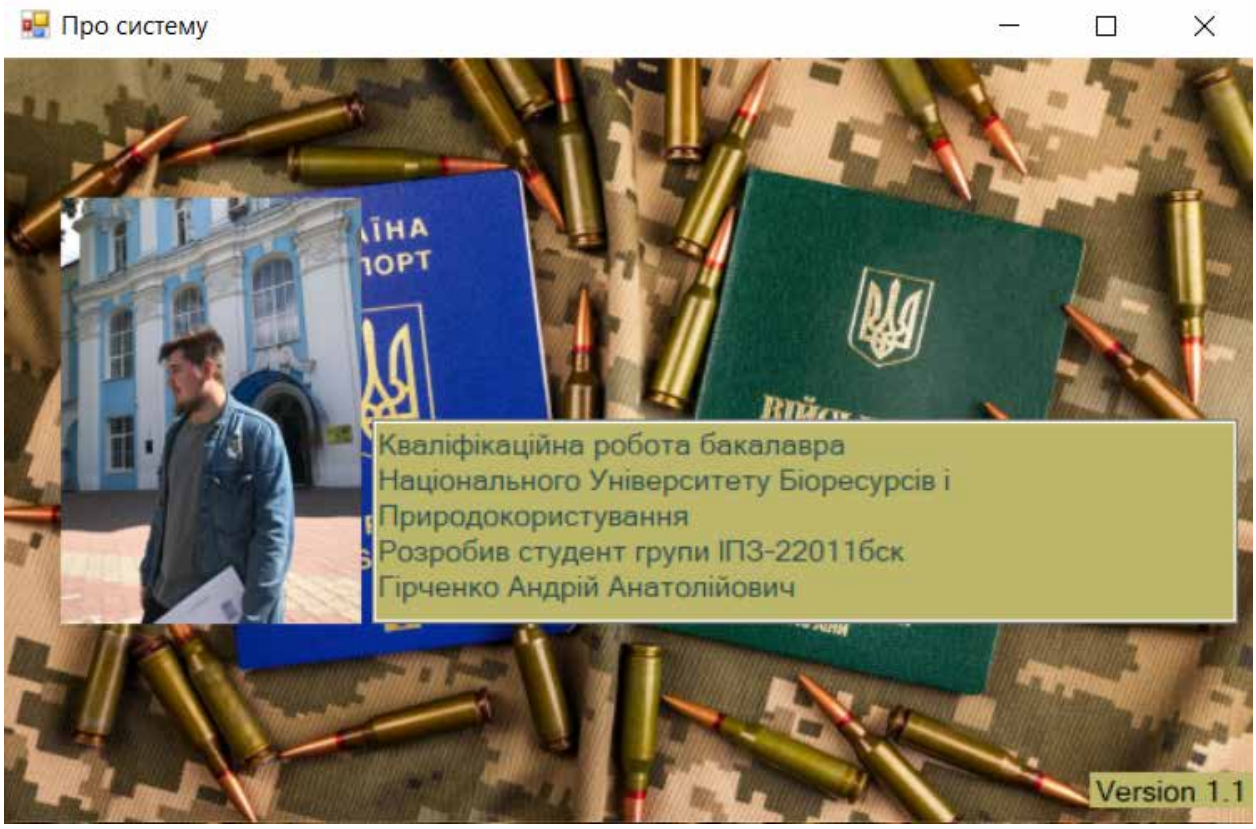


Рис. 37. Форма про систему

Під час перевірки авторизації двома видами тестувань не було виявлено жодних помилок, даний компонент системи працює коректно.

4.2 Вимоги до апаратного та програмного забезпечення

Для системи обліку призовників написаного на мові програмування C# з використанням Windows Forms та бази даних Microsoft SQL Server є такі мінімальні вимоги:

Мінімальні вимоги до програмного забезпечення:

1) Операційна система:
    - Windows 10 (будь-яка сучасна 64-розрядна версія).
2) .NET Framework
    - Актуальна версія .NET Framework, сумісна з версією C#
3) Microsoft SQL Server:

- SQL Server Express Edition: для невеликих додатків та локального розгортання. Мінімальна рекомендована версія: SQL Server 2022 Express

4) PDF-читач:
    - Будь-який встановлений на комп’ютері PDF-читач, для перегляду звітів, що генеруються.

Мінімальні вимоги до апаратного забезпечення (для користувача):

1) Процесор:
    - Двоядерний процесор з тактовою частотою від 1.8 ГГц (x64 архітектура).
2) Оперативна пам’ять:
    - 4 ГБ RAM (для базової роботи системи Windows + додаток)
3) Місце на жорсткому диску:
    - Мінімум 10 ГБ вільного місця на HDD/SSD, для встановлення всього потрібно ПЗ та файлів програми.
4) Дисплей:
    - Монітор з роздільною здатністю 1024x768 пікселів або вище.

4.3 Склад інсталяційного пакету

Під час проєктування для розуміння фізичної організації програмного забезпечення була спроєктована діаграма розгортання.

Діаграма розгортання – це діаграма для моделювання архітектури та фізичного розгортання системи. Вона надає графічне представлення вузлів, на яких виконуються компоненти системи, та зв’язків між ними, що дозволяє візуалізувати фізичну інфраструктуру.

Для даної системи обліку призовників, що базується на C# (Windows Forms) та Microsoft SQL Server, діаграма розгортання передбачає встановлення всіх компонентів на одну робочу станцію. Це означає, що як

клієнтський додаток, так і локальний екземпляр бази даних Microsoft SQL Server функціонуватимуть на одному і тому ж комп’ютері.

Основними компонентами, що розгортаються на цьому єдиному вузлі, є:

- Додаток conscript.exe: Основний виконуваний файл системи, що забезпечує графічний інтерфейс та бізнес-логіку.
- Локальна база даних Microsoft SQL Server: Екземпляр SQL Server, що містить базу даних для зберігання облікових даних призовників.
- Файли ресурсів: Конфігураційні файли, файли стилів та інші необхідні ресурси для роботи додатка.
- PDF-переглядач: Програмне забезпечення для відкриття звітів, які генеруються системою.

Зв’язок між додатком та базою даних відбувається локально, що забезпечує швидкий доступ до даних без необхідності мережевого з’єднання.

Таким чином, інсталяційний пакет програмного забезпечення системного обліку призовників буде включати в себе conscript.exe та інсталятор або компоненти для встановлення локального екземпляра Microsoft SQL Server, а також необхідні файли конфігурації.

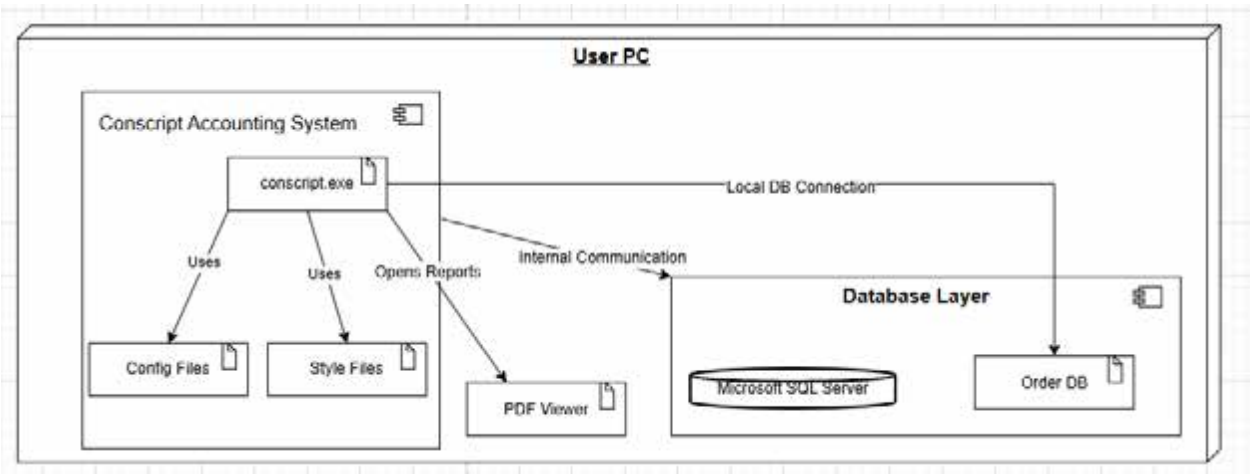


Рис. 38. Діаграма розгортання

ВИСНОВКИ

У рамках даної бакалаврської кваліфікаційної роботи було розроблено програмне забезпечення інформаційно-управляючої системи обліку призовників. Проєкт охопив ключові етапи розробки, починаючи від аналізу предметної області та проєктування архітектури, до реалізації функціональних модулів та розгортання системи.

Спочатку було поставлено такі основні функціональні вимоги до системи:

- Авторизація;
- Додавання, редагування та видалення даних про призовників;
- Перегляд записів призовників;
- Облік статусів призовника (придатний, не придатний, тимчасово непридатний тощо);
- Реалізувати пошук серед призовників за містом проживання;
- Збереження медичних даних та результатів медкомісій;
- Формування звітів;
- Збереження звітів в PDF форматі.

Було успішно вирішено завдання щодо моделювання структури системи за допомогою UML-діаграм. Діаграма компонентів чітко відобразила модульну будову додатка, що складається з шарів представлення, ViewModel, Model, а також компонентів для роботи з базою даних та допоміжніх сервісів. Це забезпечило високу модульність, що спрощує підтримку та потенційне розширення системи.

Розроблена логічна модель бази даних, що включає таблиці users та order, дозволила ефективно зберігати облікові дані призовників та інформацію про користувачів системи. Важливим аспектом реалізації бази

даних є використання механізму автентифікації користувачів, який хоч і не передбачає розмежування ролей на рівні БД, забезпечує контроль доступу та безпеку даних.

Діаграма розгортання відобразила фізичну архітектуру системи, яка передбачає встановлення всіх компонентів на одну робочу станцію, а саме:

1) Додаток C# Windows Forms
2) Локальний екземпляр Microsoft SQL Server
3) Файли ресурсів
4) PDF-переглядач

Такий підхід ідеально підходить для індивідуального використання або для невеликих організацій, де немає потреби у виділеному сервері баз даних, забезпечуючи простоту розгортання та адміністрування.

Програмне забезпечення системи обліку призовників забезпечує основний функціонал для ведення обліку, що включає додавання, перегляд, редагування даних призовників, а також генерацію звітів у форматі PDF. Реалізація цих функцій дозволяє автоматизувати рутинні операції, мінімізувати помилки ручного введення та підвищити ефективність процесу обліку.

Загалом створене програмне забезпечення відповідає поставленим вимогам і може слугувати надійною основною для подальшого розвитку та вдосконалення, включаючи розширення функціоналу, запровадження більш гнучкої системи ролей та покращення користувацького інтерфейсу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. МЕТОДИЧНІ ВКАЗІВКИ з розробки бакалаврської кваліфікаційної роботи для студентів спеціальності 121 [Електронний ресурс] джерело – https://nubip.edu.ua/sites/default/files/u423/met_bak_ipz_2025n.pdf
2. Закон України «Про військовий обов'язок і військову службу» [Електронний ресурс] – https://zakon.rada.gov.ua/laws/show/2232-12
3. Кабінет Міністрів України. Постанова № 563 «Про затвердження Положення про військовий облік призовників і військовозобов'язаних» [Електронний ресурс] – https://zakon.rada.gov.ua/laws/show/563-2022
4. ISO/IEC 25010:2011. Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models.
5. Бойко, В.І. Інформаційні системи та технології в обліку: навчальний посібник. – Київ: Центр учбової літератури, 2020. – 276 с.
6. Milovydov, Y.O. Development of Management Information Systems for State Institutions / Journal of Digital Technologies, 2021. – №3. – С. 52–60.
7. Офіційна документація Microsoft SQL Server Technical Documentation [Електронний ресурс] – https://learn.microsoft.com/uk-ua/sql/sql-server/?view=sql-server-ver16
8. Офіційна специфікація UML [Електронний ресурс] – https://www.omg.org/spec/UML/
9. PostgreSQL Documentation [Електронний ресурс] – Джерело: https://www.postgresql.org/docs/
10. Firebase Realtime Database Documentation [Електронний ресурс] – Джерело: https://firebase.google.com/docs/database/android/start
11. MongoDB Documentation [Електронний ресурс] – Джерело: https://docs.mongodb.com/
12. SQLite. [Електронний ресурс] – Джерело: https://sqlite.org/

13. Microsoft Developer Network (MSDN). (2019). Importance of Databases in Software Development [Електронний ресурс] – Джерело: https://learn.microsoft.com/en-us/
14. 1С управління персоналом [Електронний ресурс] – Джерело: https://bookkeeping.com.ua/1c-enterprise-8-salary-and-personnel-management-ukr.html
15. Google Sheets function list [Електронний ресурс] – Джерело: https://support.google.com/docs/table/25273?hl=en
16. Excel help and learning [Електронний ресурс] – Джерело: https://support.microsoft.com/en-us/excel
17. Реєстр військовозобов'язаних «Оберіг» [Електронний ресурс] – Джерело: https://dou.ua/lenta/articles/electronic-register-oberig/
18. Офіційна документація Visual Studio documentation [Електронний ресурс] – джерело: https://learn.microsoft.com/uk-ua/visualstudio/windows/?view=vs-2022
19. Офіційна документація C# language documentation [Електронний ресурс] – джерело: https://learn.microsoft.com/uk-ua/dotnet/csharp/
20. .NET Framework documentation [Електронний ресурс] – джерело: https://learn.microsoft.com/uk-ua/dotnet/framework/
21. Windows Forms for .NET documentation [Електронний ресурс] – джерело: https://learn.microsoft.com/en-us/dotnet/desktop/winforms/

Додаток А

ОПУБЛІКОВАНІ ТЕЗИ НА VII ВСЕУКРАЇНСЬКІЙ НАУКОВО-ПРАКТИЧНІЙ ІНТЕРНЕТ КОНФЕРЕНЦІЇ СТУДЕНТІВ І АСПІРАНТІВ «ТЕОРЕТИЧНІ ТА ПРИКЛАДНІ АСПЕКТИ РОЗРОБКИ КОМП'ЮТЕРНИХ СИСТЕМ 2025» (24 КВІТНЯ 2025 РОКУ)

Сторінок – 2

2025

УДК 004.51

АКТУАЛЬНІСТЬ РОЗРОБКИ АВТОМАТИЗОВАНОЇ ІНФОРМАЦІЙНО-УПРАВЛЯЮЧОЇ СИСТЕМИ ОБЛІКУ ВІЙСЬКОВОЗОБОВ’ЯЗАНИМИ

Гірченко А.А.
Науковий керівник Міловідов Ю.О.

Сучасна геополітична ситуація в Україні вимагає оперативних, точних і надійних інструментів для ведення обліку військовозобов'язаних громадян. Саме тому автоматизовані інформаційно-управляючі системи обліку призовників мають важливе стратегічне значення. Їх впровадження дозволяє значно покращити організацію роботи військових комісаріатів, зменшити навантаження на персонал та підвищити точність ведення даних.

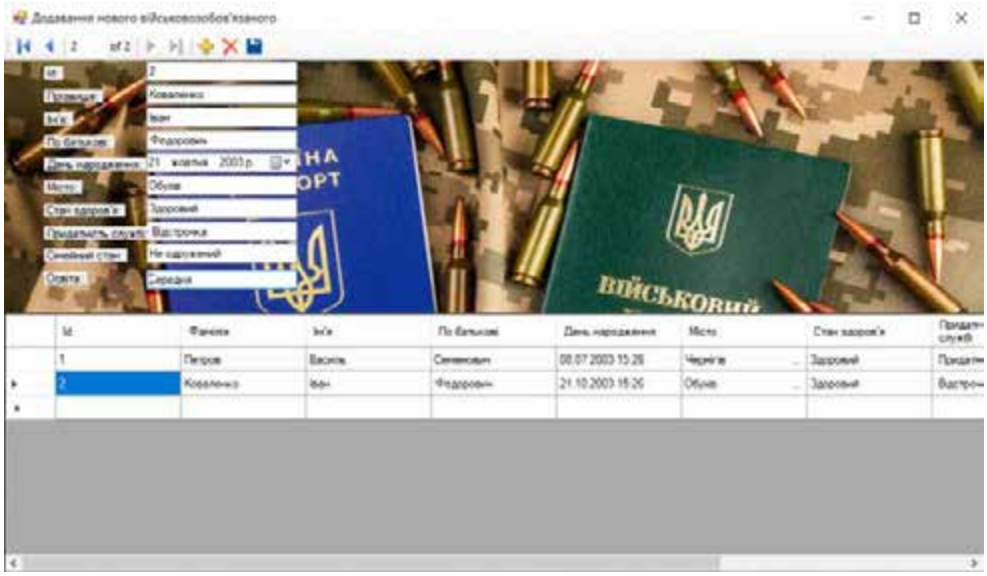


Рис. 1. Процес додавання нового запису військовозобов’язаного

У воєнний і поствоєнний період ефективне функціонування таких систем безпосередньо впливає на мобілізаційну готовність країни. Використання електронних реєстрів дозволяє в реальному часі формувати списки, контролювати наявність документів, надавати відстрочки, а також генерувати необхідні звіти для керівництва. Це сприяє як організаційній дисципліні, так і ефективнішому реагуванню на виклики національної безпеки.

Крім того, цифровізація військового обліку покращує взаємодію з призовниками. Користувачі можуть перевірити свій статус, оновити персональні дані або подати документи онлайн. Це підвищує прозорість процесів і рівень довіри з боку громадян.

Впровадження інтуїтивно зрозумілого інтерфейсу в таких системах робить їх доступними навіть для користувачів з базовими навичками роботи з ПК. Наявність продуманих сценаріїв взаємодії та автоматизованих підказок зменшує кількість помилок і підвищує ефективність облікових дій.

Отже, автоматизація обліку призовників — це не лише про зручність і точність, а про готовність країни швидко мобілізувати ресурс, захистити населення і впевнено реагувати на виклики. Впровадження таких рішень має бути пріоритетом державної цифрової трансформації у сфері безпеки та оборони.

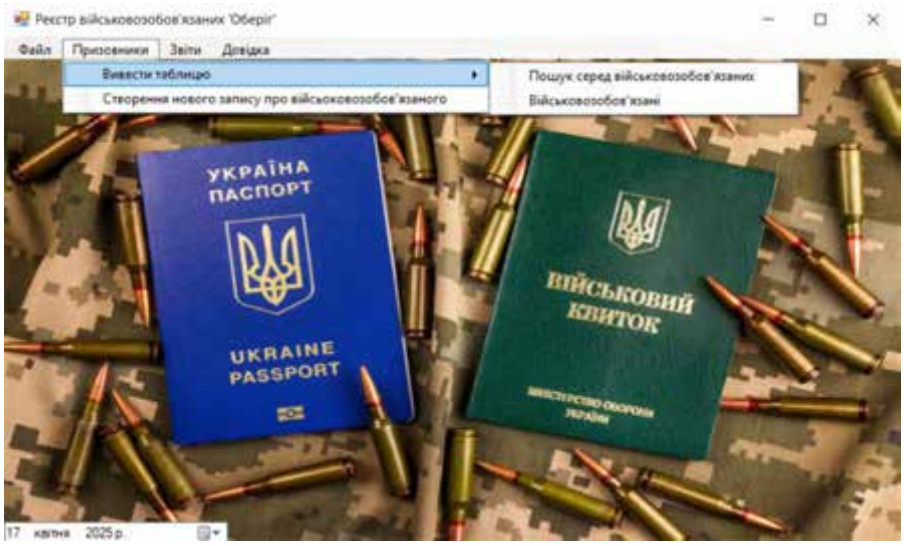


Рис. 2. Головне меню

Переваги мого програмного продукту:

- Інтуїтивність інтерфейсу, завдяки чому з нею зможе розібратись будь-яка людина яка навіть має не великі познання ПК.
- Простота ведення обліку
- Пошук серед військовозобов’язаних які вже внесені в базу даних

Недоліки:

- Малий функціонал, в ідеалі додати ще хоча б декілька функцій які б облегчили ще більше облік військовозобов’язаними
- Доступ до програми є лише у військових комісаріатів, тобто звичайний користувач не може переглянути дані про себе

ВИСНОВКИ

За результатами мого дослідження можна зробити такі висновки, що розробка автоматизованої інформаційно-управляючої системи обліку військовозобов’язаними, дуже актуальна, як на сьогоднішній день, так і на майбутнє, так як вести облік призовників потрібно як у військовий час так і під час мирного життя країни дуже важливо. За допомогою мого продукту можна облегшити процес обліку військовозобов’язаними, та значно спростити час на навчання використання програмою. Зокрема трохи переробивши програму її також можна використовувати як звичайну програму для обліку працівників на роботі, або використовувати її як облік учнів в школі або університеті, тощо…

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Закон України «Про військовий обов’язок і військову службу» [Електронний ресурс]. https://zakon.rada.gov.ua/laws/show/2232-12
2. Кабінет Міністрів України. Постанова № 563 «Про затвердження Положення про військовий облік призовників і військовозобов’язаних» [Електронний ресурс]. https://zakon.rada.gov.ua/laws/show/563-2022
3. ISO/IEC 25010:2011. Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models.
4. Бойко, В.І. Інформаційні системи та технології в обліку: навчальний посібник. – Київ: Центр учбової літератури, 2020. – 276 с.

5. Milovydov, Y.O. Development of Management Information Systems for State Institutions / Journal of Digital Technologies, 2021. – №3. – С. 52–60

Додаток Б

SQL-ЗАПИТИ

Сторінок – 1

2025

```
CREATE DATABASE Database

CREATE TABLE [dbo].[users] (
    [id]       INT          IDENTITY (1, 1) NOT NULL,
    [username] NVARCHAR (50) NOT NULL,
    [password] NVARCHAR (50) NOT NULL,
    PRIMARY KEY CLUSTERED ([id] ASC)
);

CREATE TABLE [dbo].[order] (
    [Id]           INT       NOT NULL,
    [surname]       NCHAR (20) NULL,
    [name]          NCHAR (20) NULL,
    [second_name]    NCHAR (20) NULL,
    [birthday]      DATETIME   NULL,
    [place]         NCHAR (50) NULL,
    [health]        NCHAR (20) NULL,
    [suitability]    NCHAR (20) NULL,
    [marital_status] NCHAR (30) NULL,
    [education]     NCHAR (50) NULL,
    PRIMARY KEY CLUSTERED ([Id] ASC)
);
```

Додаток В

ТАБЛИЦІ

Сторінок – 1

2025

Табл. 1

Порівняння аналогів:

Сервіс/Система	Переваги	Недоліки
Оберіг (державна система)	Інтеграція з реєстрами, офіційна підтримка	Обмежений доступ, не публічна
Excel / Google Sheets	Простота, гнучкість	Відсутність захисту даних, немає автоматизації, ручне введення
1С:Облік персоналу	Можливість обліку великої кількості людей	Неадаптована під специфіку військового обліку
Кастомні локальні рішення ТЦК	Пристосовані до конкретного офісу	Відсутність єдиних стандартів, труднощі в супроводі

Табл. 2.

Таблиця Users:

Поле	Тип	Ключ	Опис
user_id	INT AUTO_INCREMENT	PK	ID користувача
username	VARCHAR(50)		Логін
password	VARCHAR(50)		Хеш пароля

Табл. 3.

Таблиця Order:

Поле	Тип	Ключ	Опис
id	ID AUTO_INCREMENT	PK	ID призовника
surname	CHAR(20)		Прізвище
name	CHAR(20)		Ім’я
second_name	CHAR(20)		По-батькові
birthday	DATETIME		Дата народження
place	CHAR(50)		Адреса проживання
health	CHAR(20)		Стан здоров’я
suitability	CHAR(20)		Придатність до служби
marital_status	CHAR(30)		Сімейний стан
education	CHAR(50)		Освіта

Додаток Г

КОД РЕАЛІЗАЦІЇ ОСНОВНИХ ФУНКЦІЙ

Сторінок – 7

2025

Реалізація зникаючого фонового напису на полях «Логін» та «Пароль»:

```
 public Login()
   {
   InitializeComponent();
   txtLogin.ForeColor = Color.Gray;
   txtLogin.Text = “Логін”;
   txtLogin.GotFocus += RemoveText;
   txtLogin.LostFocus += AddText;

   txtPassword.ForeColor = Color.Gray;
   txtPassword.Text = “Пароль”;
   txtPassword.UseSystemPasswordChar = false;
   txtPassword.GotFocus += RemovePasswordText;
   txtPassword.LostFocus += AddPasswordText;
   label1.Select();
}
private void RemoveText(object sender, EventArgs e)
{
   if (txtLogin.Text == “Логін”)
   {
      txtLogin.Text = “”;
      txtLogin.ForeColor = Color.Black;
   }
}

private void AddText(object sender, EventArgs e)
{
   if (string.IsNullOrWhiteSpace(txtLogin.Text))
   {
      txtLogin.Text = “Логін”;
      txtLogin.ForeColor = Color.Gray;
   }
}

private void RemovePasswordText(object sender, EventArgs e)
{
   if (txtPassword.Text == “Пароль”)
   {
      txtPassword.Text = “”;
      txtPassword.ForeColor = Color.Black;
      txtPassword.PasswordChar = ‘*’;
   }
}

private void AddPasswordText(object sender, EventArgs e)
{
   if (string.IsNullOrWhiteSpace(txtPassword.Text))
   {
      txtPassword.Text = “Пароль”;
      txtPassword.ForeColor = Color.Gray;
   }
```

```
    }
```

Реалізація авторизації:

```
private void button1_Click(object sender, EventArgs e)
{
string username = txtLogin.Text.Trim();
string password = txtPassword.Text.Trim();

string connectionString = @"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\Database.mdf;Integrat
ed Security=True";

using (SqlConnection conn = new SqlConnection(connectionString))
{
    try
    {
        conn.Open();

        string query = "SELECT COUNT(*) FROM users WHERE username = @username AND
password = @password";
        using (SqlCommand cmd = new SqlCommand(query, conn))
        {
            cmd.Parameters.AddWithValue("@username", username);
            cmd.Parameters.AddWithValue("@password", password);

            int count = (int)cmd.ExecuteScalar();

            if (count > 0)
            {
                MessageBox.Show("Авторизація в системі \"Оберіг\" успішна!");
                MainWindow main = new MainWindow(username, password); // ☑
                main.Show();
                this.Hide();
            }
            else
            {
                MessageBox.Show("Невірне ім'я користувача або пароль.");
            }
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Помилка підключення: " + ex.Message);
    }
}

}
```

Код реалізації пошуку даних в таблиці за фільтром «Місто» :

```
private void buttonSearch_Click(object sender, EventArgs e)
 {
    string searchValue = textBoxSearch.Text.Trim();
```

```
    if (!string.IsNullOrEmpty(searchValue))
    {
        orderBindingSource.Filter = $"place LIKE '%{searchValue}%'";
    }
    else
    {
        orderBindingSource.RemoveFilter(); // Якщо поле пусте — скидаємо фільтр
    }
}
```

Реалізація формування звітів за фільтром та вказаному критерію:

```
    private void ReportForm_Load(object sender, EventArgs e)
    {
        comboBoxFilterType.Items.Add("Місто");
        comboBoxFilterType.Items.Add("Рік народження");
        comboBoxFilterType.Items.Add("Придатність до служби");
        comboBoxFilterType.Items.Add("Сімейний стан");
        comboBoxFilterType.Items.Add("Освіта");
        comboBoxFilterType.SelectedIndex = 0;

        // Завантаження даних
        this.orderTableAdapter.Fill(this.databaseDataSet.order);
    }

    private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
    {

    }

    private void buttonGenerateReport_Click(object sender, EventArgs e)
    {
        string filterType = comboBoxFilterType.SelectedItem.ToString();
        string filterValue = textBoxFilterValue.Text.Trim();

        if (string.IsNullOrEmpty(filterValue))
        {
            MessageBox.Show("Введіть значення для фільтра!");
            return;
        }

        DataRow[] filteredRows;

        try
        {
            switch (filterType)
            {
                case "Місто":
                    filteredRows = databaseDataSet.order.Select($"place LIKE '%{filterValue}%'");
                    break;
                case "Рік народження":
```

```
                filteredRows = databaseDataSet.order
                    .Where(r => ((DateTime)r[“birthday”]).Year == int.Parse(filterValue))
                    .ToArray();
                break;
            case “Придатність до служби”:
                filteredRows = databaseDataSet.order.Select($”suitability = ‘{filterValue}’”);
                break;

            case “Сімейний стан”:
                filteredRows = databaseDataSet.order.Select($”marital_status = ‘{filterValue}’”);
                break;


            case “Освіта”:
                filteredRows = databaseDataSet.order.Select($”education = ‘{filterValue}’”);
                break;

            default:
                filteredRows = databaseDataSet.order.Select();
                break;
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(“Помилка фільтрації: “ + ex.Message);
        return;
    }

    if (filteredRows.Length == 0)
    {
        MessageBox.Show(“Не знайдено записів за заданим фільтром.”);
        return;
    }

    SaveFileDialog saveFileDialog = new SaveFileDialog
    {
        Filter = “PDF файл|*.pdf”,
        Title = “Зберегти звіт у PDF”,
        FileName = “Звіт.pdf”
    };

    if (saveFileDialog.ShowDialog() == DialogResult.OK)
    {
        try
        {
            Document doc = new Document(PageSize.A4.Rotate(), 10, 10, 10, 10);
            PdfWriter.GetInstance(doc, new FileStream(saveFileDialog.FileName,
FileMode.Create));
            doc.Open();

            // 1. Реєстрація шрифтів для підтримки кирилиці
```

```
            string fontPath =
Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.Fonts),
“ARIALUNI.TTF”); // Arial Unicode MS
            if (!File.Exists(fontPath))
            {
               fontPath =
Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.Fonts), “arial.ttf”); //
Звичайний Arial
            }
            if (!File.Exists(fontPath))
            {
               fontPath =
Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.Fonts), “times.ttf”); //
Times New Roman
            }
            iTextSharp.text.pdf.BaseFont baseFont; // Явно вказуємо
iTextSharp.text.pdf.BaseFont
            try
            {
               baseFont = iTextSharp.text.pdf.BaseFont.CreateFont(fontPath,
iTextSharp.text.pdf.BaseFont.IDENTITY_H,
iTextSharp.text.pdf.BaseFont.NOT_EMBEDDED);
            }
            catch (Exception fontEx)
            {
               MessageBox.Show($”Не вдалося завантажити шрифт. Перевірте наявність
файлу шрифту: {fontPath}\nПомилка: {fontEx.Message}”, “Помилка шрифту”,
MessageBoxButtons.OK, MessageBoxIcon.Error);
               // Запасний варіант, який може не відображати кирилицю
               baseFont =
iTextSharp.text.pdf.BaseFont.CreateFont(iTextSharp.text.pdf.BaseFont.TIMES_ROMAN,
iTextSharp.text.pdf.BaseFont.CP1252, iTextSharp.text.pdf.BaseFont.NOT_EMBEDDED);
            }

            // Явно вказуємо iTextSharp.text.Font
            iTextSharp.text.Font customFont = new iTextSharp.text.Font(baseFont, 10);
            // Для жирного шрифту використовуємо iTextSharp.text.Font.BOLD
            iTextSharp.text.Font headerFont = new iTextSharp.text.Font(baseFont, 10,
iTextSharp.text.Font.BOLD, iTextSharp.text.BaseColor.BLACK);
            iTextSharp.text.Font titleFont = new iTextSharp.text.Font(baseFont, 14);
            iTextSharp.text.Font filterFont = new iTextSharp.text.Font(baseFont, 12);

            doc.Add(new Paragraph(“Звіт про призовників”, titleFont));
            doc.Add(new Paragraph($”Фільтр: {filterType} = {filterValue}”, filterFont));
            doc.Add(new Paragraph(“Дата: “ + DateTime.Now.ToShortDateString(),
customFont));
            doc.Add(new Paragraph(“ “, customFont));

            // Створюємо таблицю з кількістю колонок з DataTable
            PdfPTable table = new PdfPTable(databaseDataSet.order.Columns.Count);
            table.WidthPercentage = 100;
            // Словник для перекладу назв колонок
```

```
            Dictionary<string, string> columnTranslations = new Dictionary<string, string>
            {
                { “Id”, “ID” },
                { “surname”, “Прізвище” },
                { “name”, “Ім’я” },
                { “second_name”, “По батькові” },
                { “birthday”, “Дата народження” },
                { “place”, “Місто” },
                { “health”, “Стан здоров’я” },
                { “suitability”, “Придатність до служби” },
                { “marital_status”, “Сімейний стан” },
                { “education”, “Освіта” }
            };

            // Додаємо заголовки з перекладом
            foreach (DataColumn column in databaseDataSet.order.Columns)
            {
                string headerText = columnTranslations.ContainsKey(column.ColumnName)
                    ? columnTranslations[column.ColumnName]
                    : column.ColumnName;
                PdfPCell headerCell = new PdfPCell(new Phrase(headerText, headerFont));
                headerCell.BackgroundColor = iTextSharp.text.BaseColor.LIGHT_GRAY;
                table.AddCell(headerCell);
            }

            // Додаємо дані
            foreach (DataRow row in filteredRows)
            {
                foreach (DataColumn column in databaseDataSet.order.Columns)
                {
                    string value;
                    if (row.IsNull(column))
                    {
                        value = “-“;
                    }
                    else if (column.ColumnName == “birthday” && row[column] is DateTime)
                    {
                        // Спеціальне форматування для дати, щоб уникнути часу або зайвого
переносу
                        value = ((DateTime)row[column]).ToShortDateString();
                    }
                    else
                    {
                        value = row[column].ToString();
                    }
                    table.AddCell(new Phrase(value, customFont));
                }

            }

            doc.Add(table);
            doc.Close();
```

```
            MessageBox.Show(“Звіт успішно збережено!”);
        }
        catch (Exception ex)
        {
            MessageBox.Show(“Помилка створення PDF: “ + ex.Message);
        }
    }
}
```

Додаток Ґ

СКРІНШОТИ ПРОГРАМИ

Сторінок – 8

2025

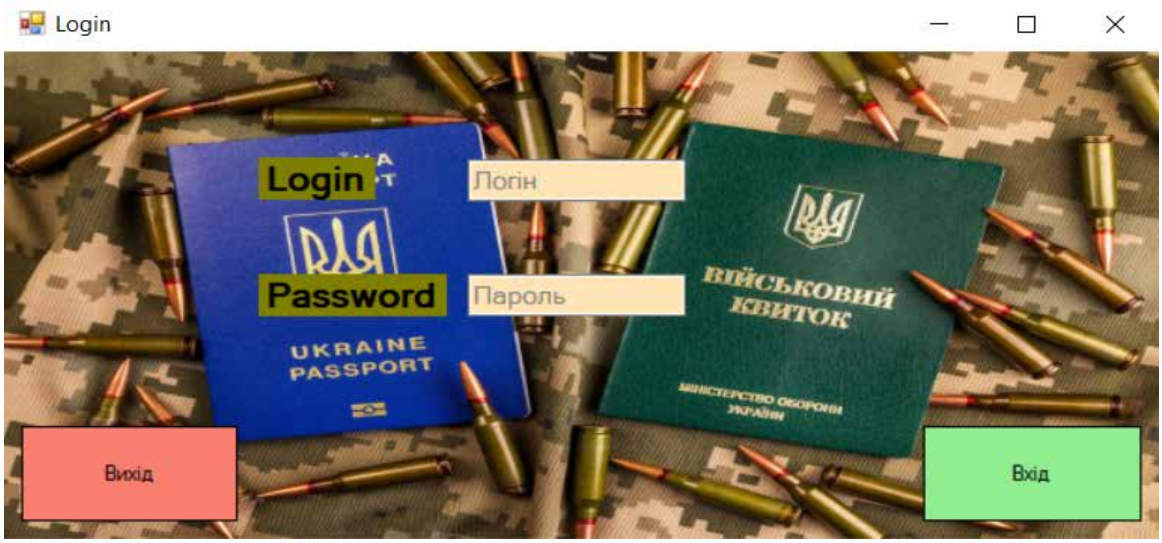


Рис. 18. Вікно авторизації в системі

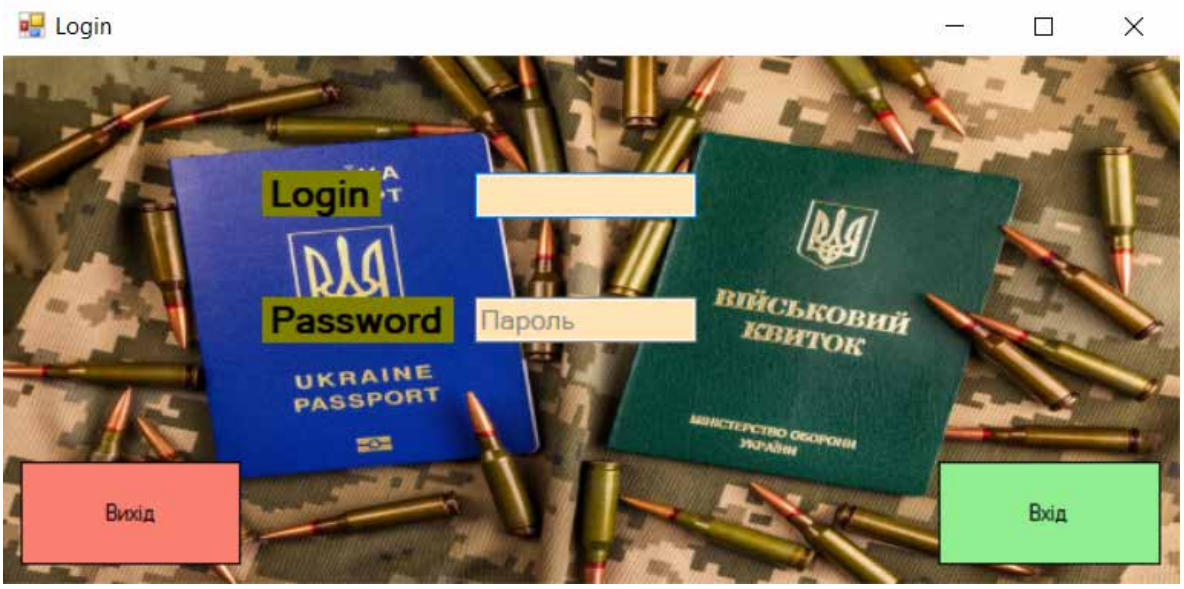


Рис. 19. Вікно входу до пз з обраним полем введення

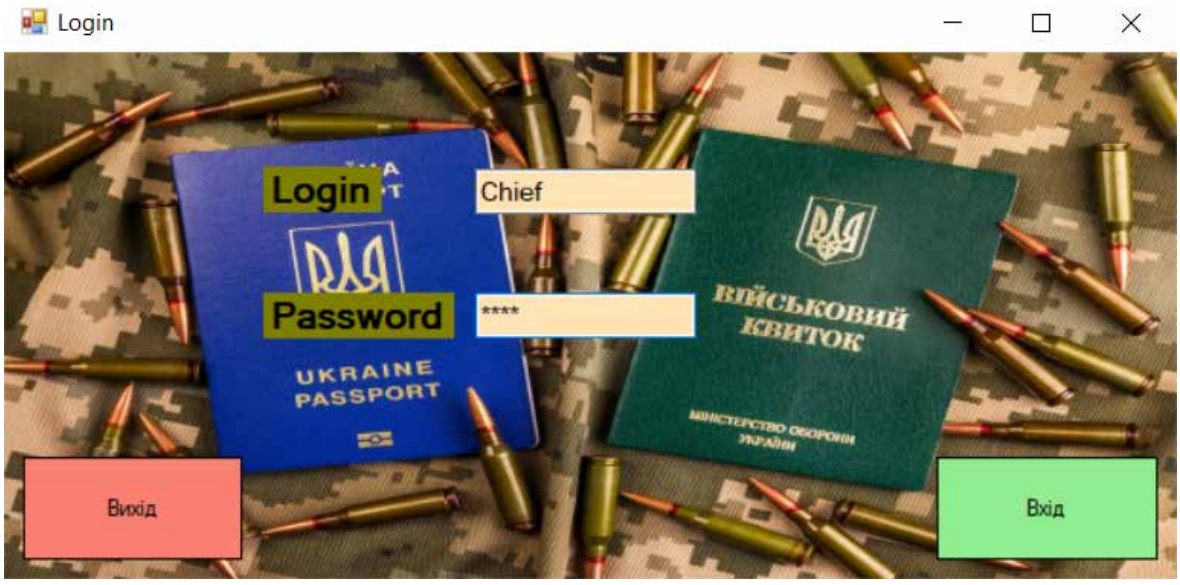


Рис. 20. Вікно авторизації з введеними даними

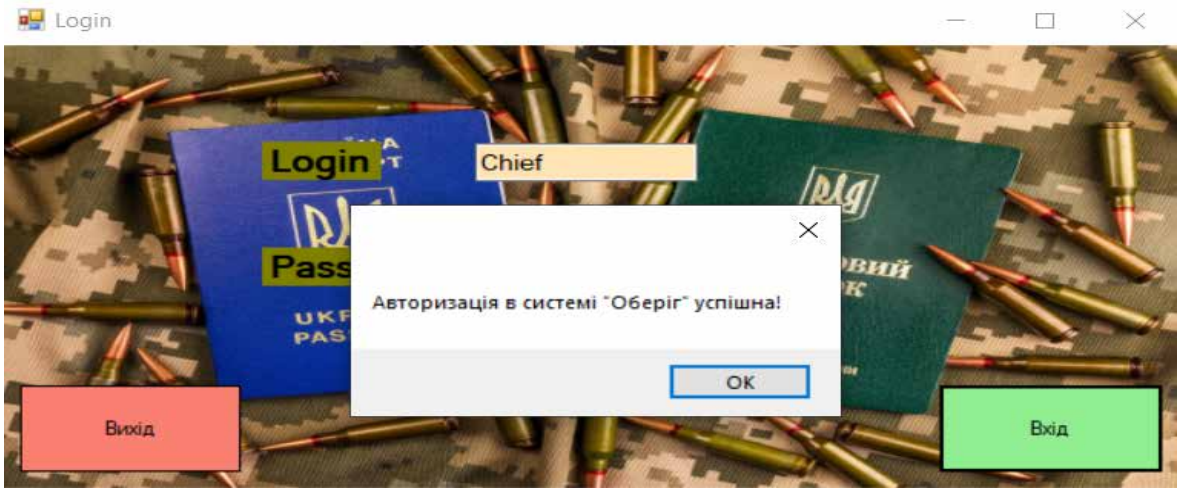


Рис. 21. Повідомлення про успішну авторизацію в системі

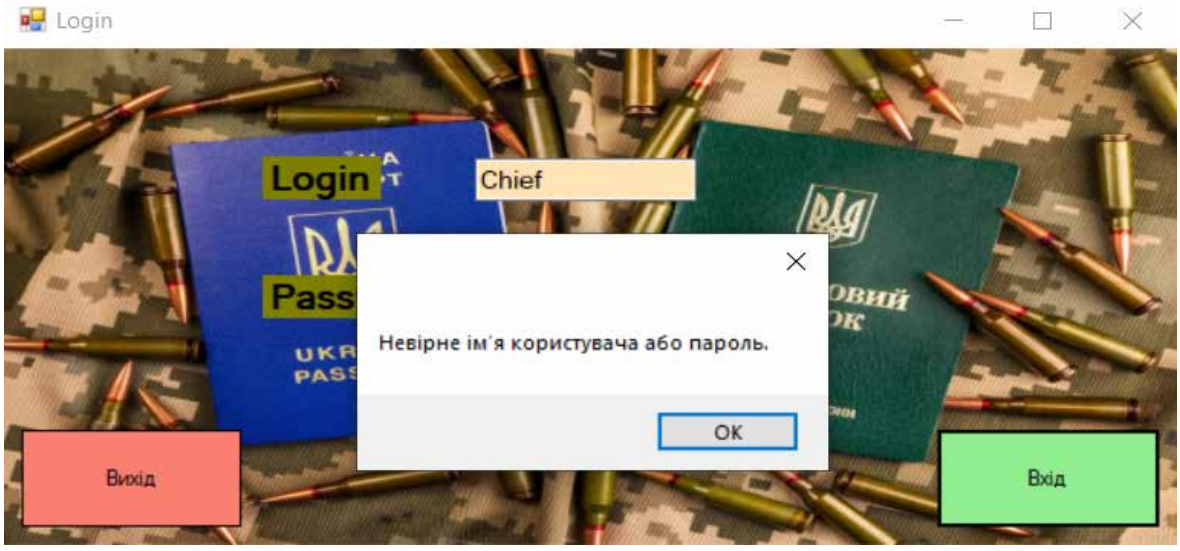


Рис. 22. Повідомлення про невірно введені дані

Рис. 23. Домашня сторінка застосунку

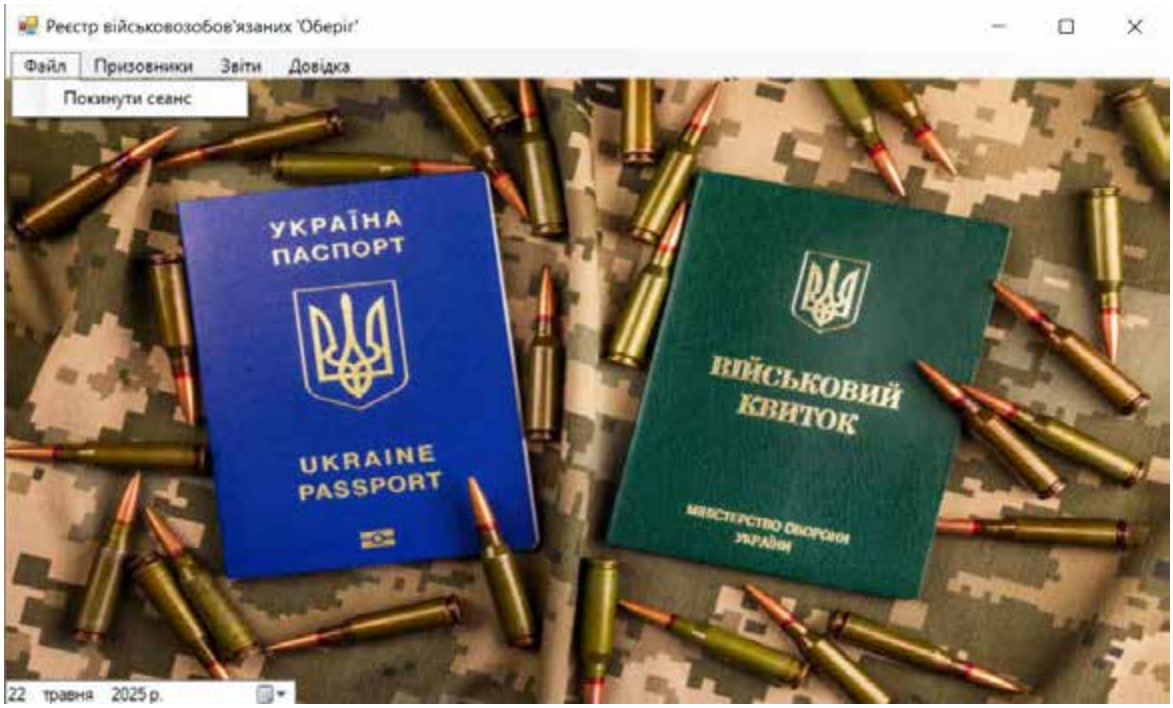


Рис. 24. Текстове меню «Файл»

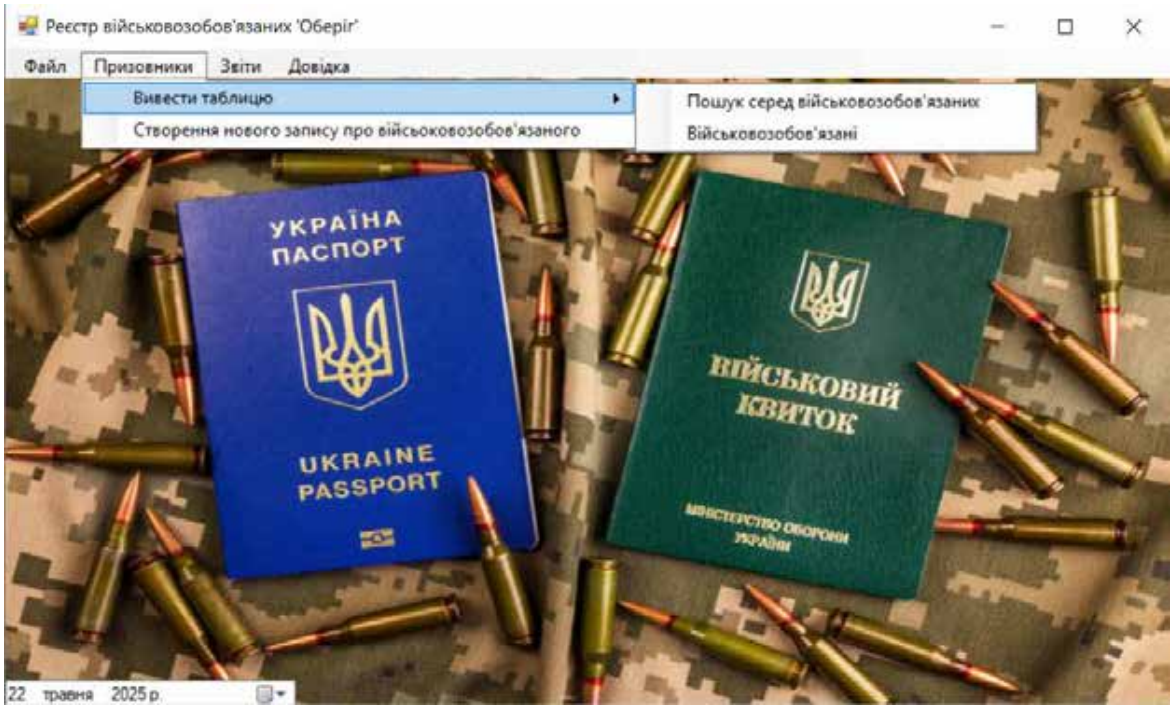


Рис. 25. Текстове меню «Призовники»

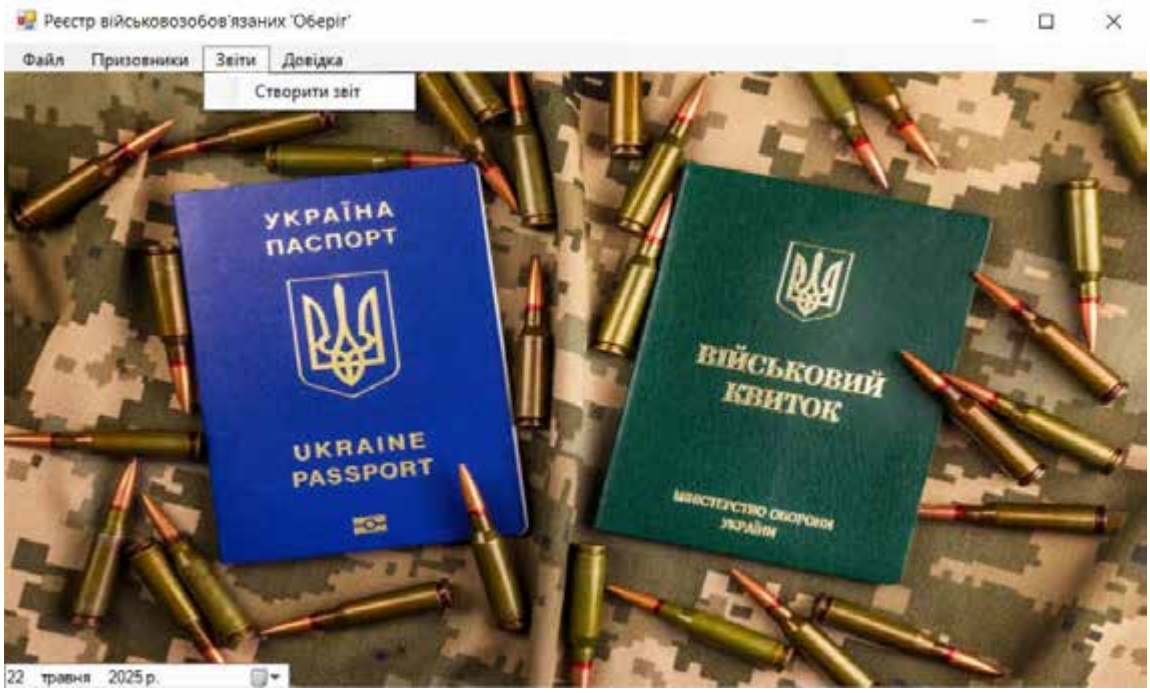


Рис. 26. Текстове меню «Звіти»

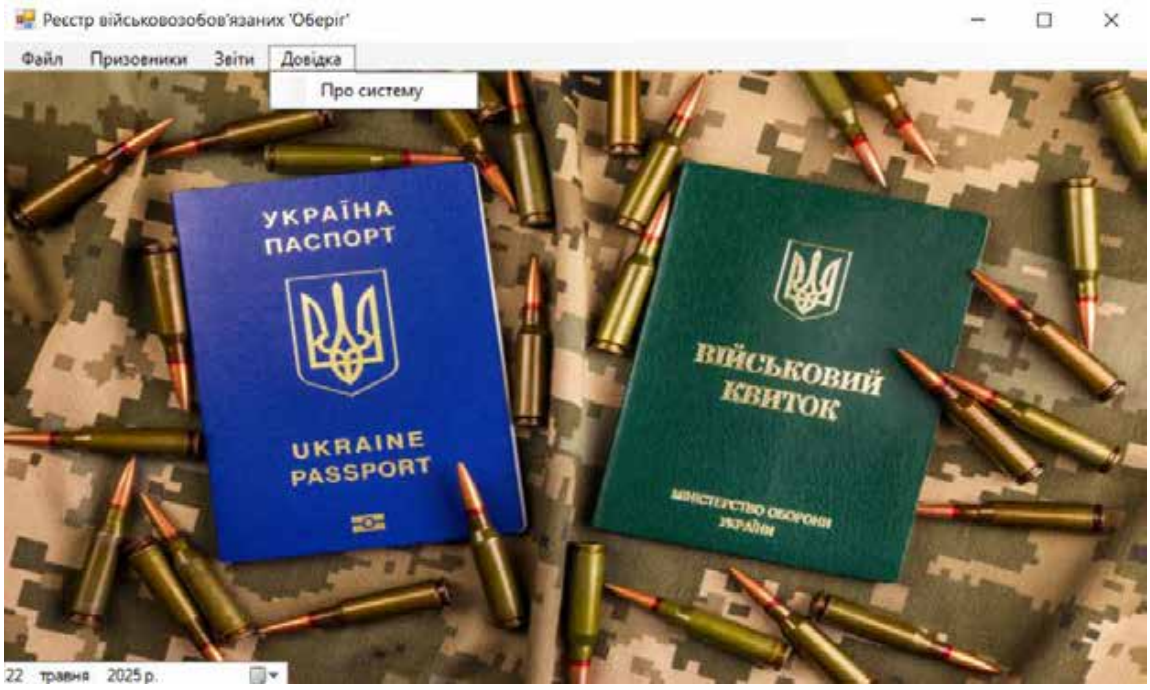


Рис. 27. Текстове меню «Про систему»

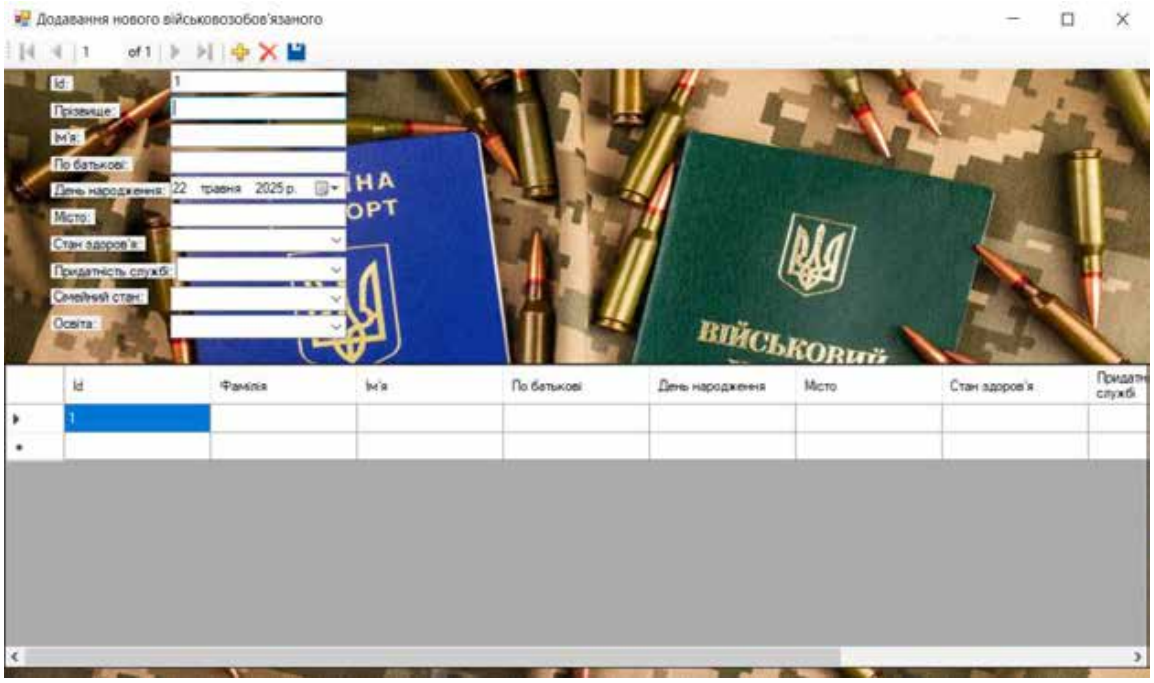


Рис. 28. Форма додавання нового призовника

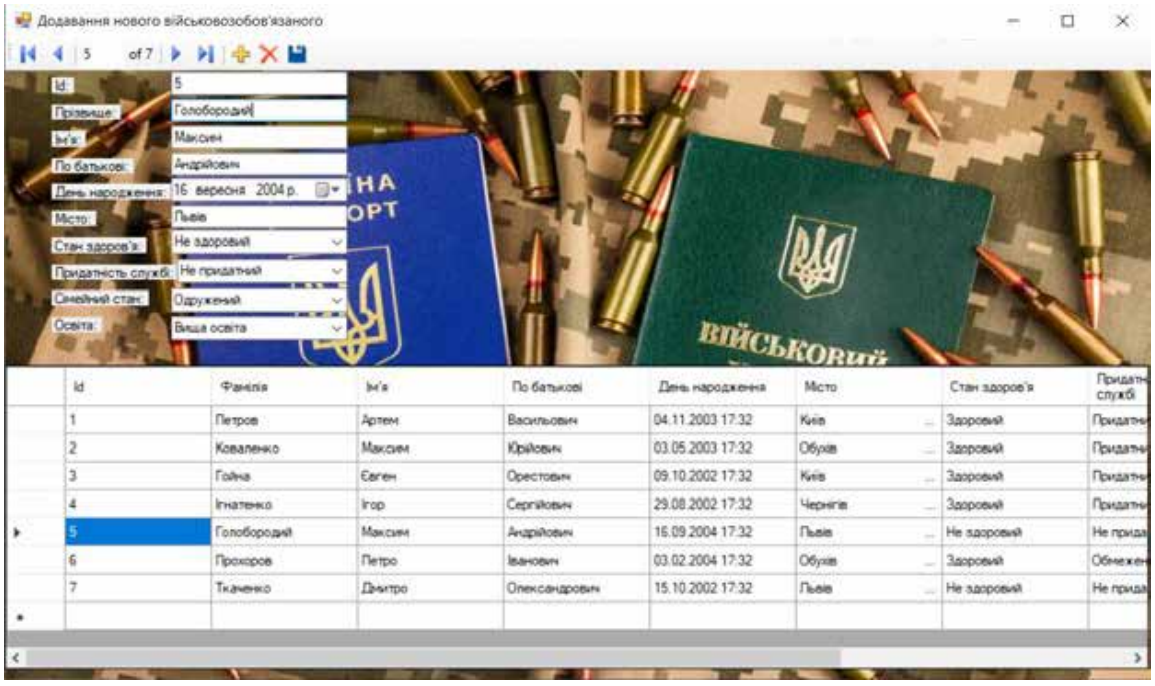


Рис. 29. Форма редагування даних призовників

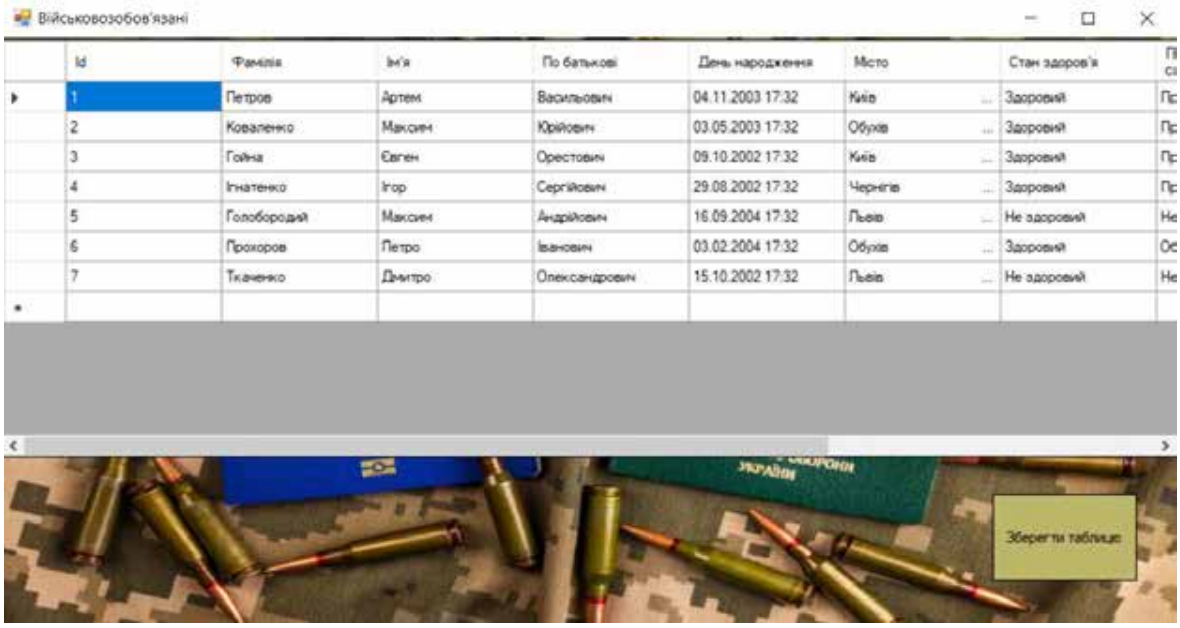


Рис. 30. Форма перегляду військовозобов’язаних

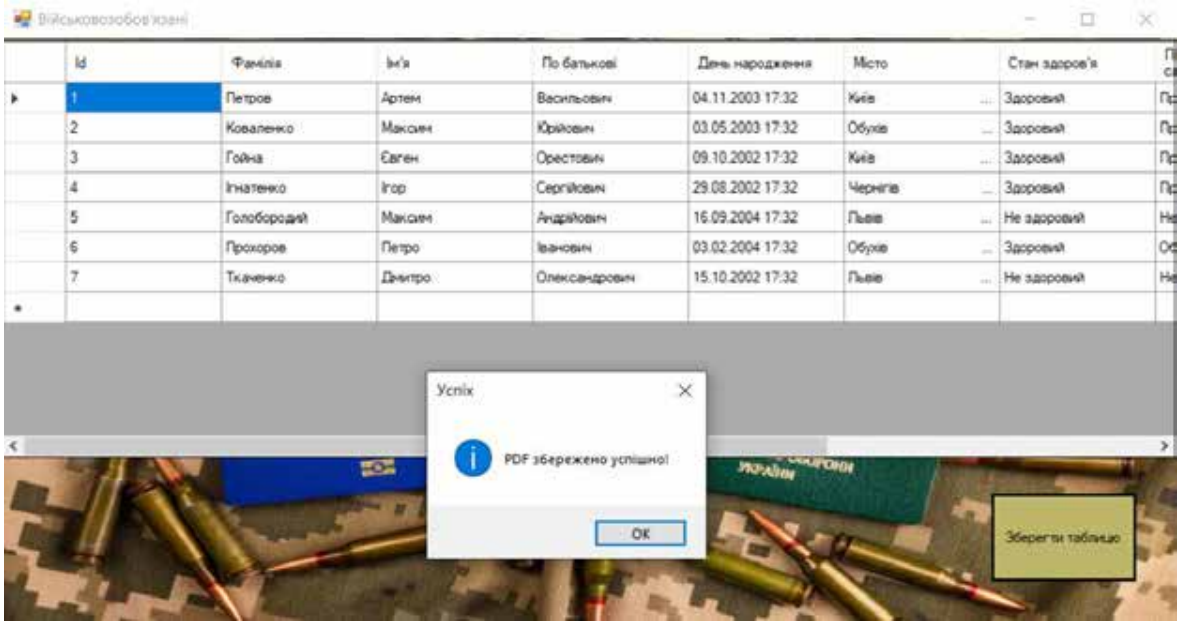


Рис. 31. Вікно інформування про успішне збереження файлу

Рис. 32. Вікно пошуку серед військовозобов’язаних

Рис. 33. Пошук серед військовозобов’язаних з міста Львів

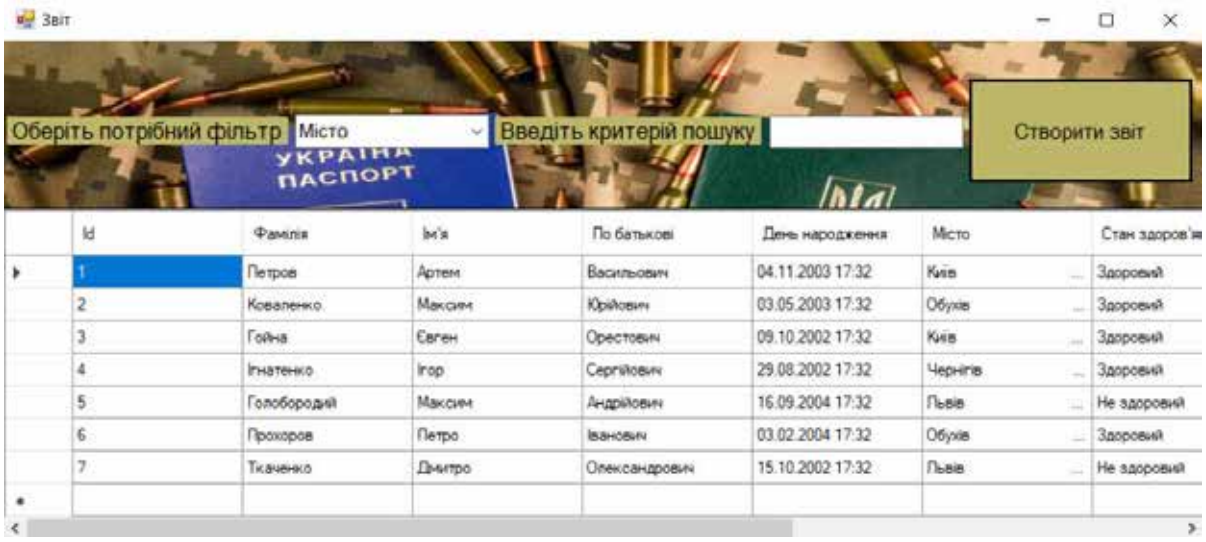
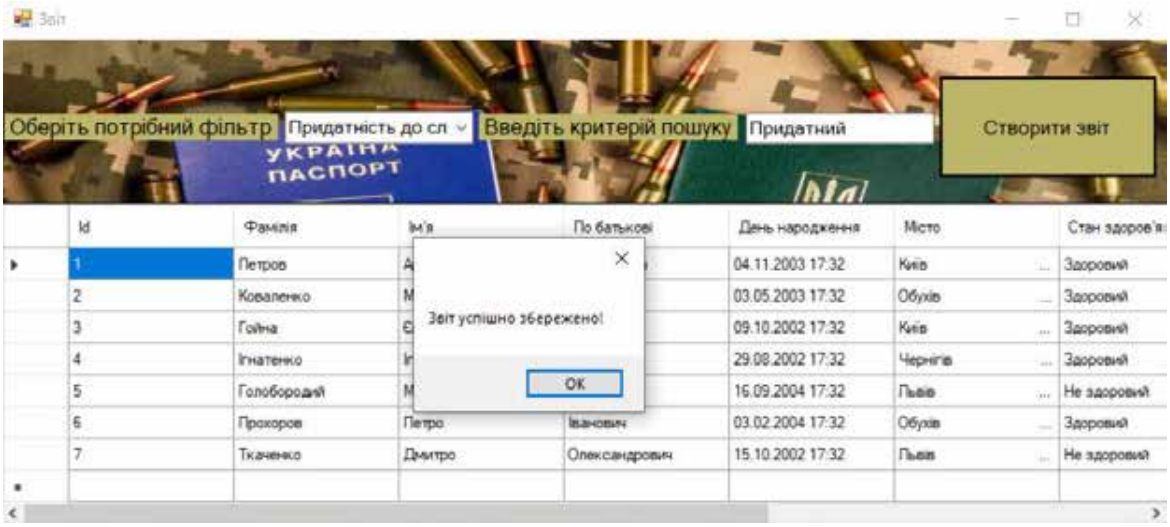


Рис. 34. Вікно створення звітів



Id	Фамілія	Ім'я	По батькові	День народження	Місто	Стан здоров'я
1	Петров	А		04.11.2003 17:32	Київ	Здоровий
2	Коваленко	М		03.05.2003 17:32	Обухів	Здоровий
3	Гойна	Є		09.10.2002 17:32	Київ	Здоровий
4	Ігнатенко	І		29.08.2002 17:32	Чернігів	Здоровий
5	Голобородий	М		16.09.2004 17:32	Львів	Не здоровий
6	Прохоров	Петро	Іванович	03.02.2004 17:32	Обухів	Здоровий
7	Ткаченко	Дмитро	Олександрович	15.10.2002 17:32	Львів	Не здоровий

Рис. 35. Створили запит за фільтром придатності до служби, по критерію «придатний»

Звіт про призовників

Фільтр: Придатність до служби = Придатний

Дата: 22.05.2025

ID	Прізвище	Ім'я	По батькові	Дата народження	Місто	Стан здоров'я	Придатність до служби	Сімейний стан	Освіта
1	Петров	Артем	Васильович	04.11.2003	Київ	Здоровий	Придатний	Не одружений	Професійно-технічна освіта
2	Коваленко	Максим	Юрійович	03.05.2003	Обухів	Здоровий	Придатний	Одружений	Вища освіта
3	Гойна	Євген	Орестович	09.10.2002	Київ	Здоровий	Придатний	Не одружений	Вища освіта
4	Ігнатенко	Ігор	Сергійович	29.08.2002	Чернігів	Здоровий	Придатний	Не одружений	Професійно-технічна освіта

Рис. 36. Звіт сформований за запитом по критерію «Придатний»

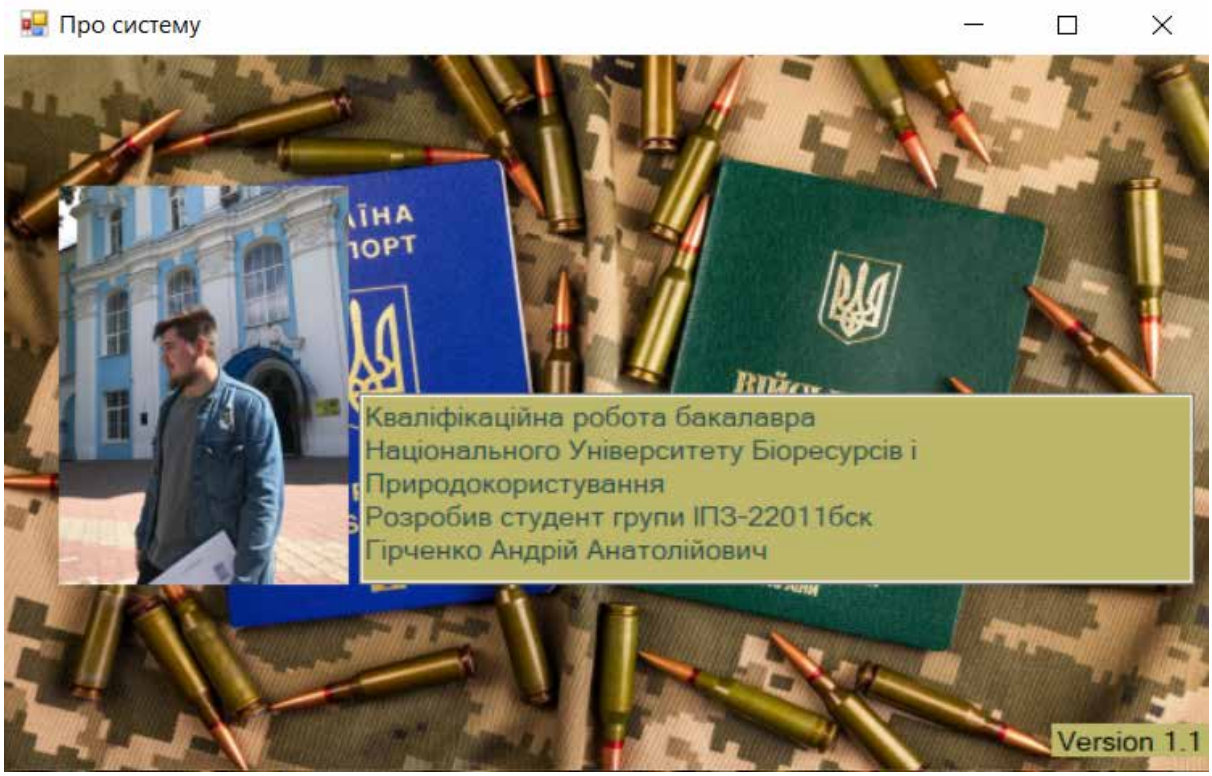


Рис. 37. Форма про систему