МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

ЦЕНТР ЗАОЧНОЇ, ДИСТАНЦІЙНОЇ ТА ВЕЧІРНЬОЇ ФОРМ НАВЧАННЯ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ВИПУСКНА РОБОТА

на тему:

«Автоматизована система управління даними для відділу кадрів»

Завідувач випускаючої кафедри — **Довбиш А.С.**

Керівник роботи — **Ободяк В.К.**

Студента групи ІНЗ – 71с — **Захарченко М.О.**

СУМИ 2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

ЦЕНТР ЗАОЧНОЇ, ДИСТАНЦІЙНОЇ ТА ВЕЧІРНЬОЇ ФОРМ НАВЧАННЯ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Затверджую_____

Зав. кафедрою Довбиш А.С.

"_____" _____2021 г.

ЗАВДАННЯ

до випускної роботи

Студента четвертого курсу, групи ІНз-71с спеціальності "Інформатика" заочної форми навчання Захарченка Максима Олександровича.

Тема: " **Автоматизована система управління даними для відділу кадрів** "

Затверджена наказом по СумДУ

№_____ от _____2021 г.

Зміст пояснювальної записки: 1) аналіз існуючих систем управління даними; 2) постановка завдання й формування вимог до програми; 3) вибір інструментів реалізації проектів; 5) розробка бази даних та програмного забезпечання ; 6) тестування працездатності програмного забезпечення; 7) висновки по реалізованій роботі.

Дата видачі завдання "_____"_____2021 г.

Керівник випускної роботи _____ Ободяк В.К.

Завдання приняв до виконання _____ Захарченко М.О.

РЕФЕРАТ

Записка: 85 стор., 22 рис., 16 табл., 3 додатка, 16 джерел.

Об’єкт дослідження — процес управління данними.

Мета роботи — розробка бази даних та програмного забезпечення для управління персоналом для відділу кадрів.

Результати — розроблена база даних та програмний додаток до неї може використовуватись на невеликому підприємстві у відділі кадрів для ведення обліку працівників. Також цей програмний продукт можна використовувати у навчальних цілях, для демонстрації можливостей сучасних мов програмування та потужносте СУБД.

ЗМІСТ

ВСТУП

У минулому столітті, коли цифрові технології стали в реальній мірі входити в життя і побут людей, людство перейшло в нову епоху – епоху інформаційно-комунікаційних технологій. З цього приводу 22 липня 2000 р. на черговому засіданні лідерів восьми найбільш промислово розвинених країн світу була прийнята «Окінавська Хартія Глобального інформаційного суспільства», в якій говорилося: «інформаційно-комунікаційні технології (ІТ) є одним з найбільш важливих факторів, що впливають на формування суспільства двадцять першого століття [1].

Як видно з вище процитованого ІТ стала важливим стимулом розвитку економіки, допомагають вирішувати економічні та соціальні проблеми. Така частина ІТ, як комп'ютерні системи, особливо в області комунікацій і зберігання даних, розвиваються трохи швидше, ніж інші, тому що їм доводиться йти в ногу з технологічними і соціальними змінами в світі.

Оскільки бази даних (БД), поза всякими сумнівами, займають лідируюче положення в області інформаційних технологій, вони стають невід'ємною частиною життя сучасної людини.

Розвиток систем управління процесами йде шляхом інтеграції в єдине ціле, і величезним кроком до цього є використання стандарту обробки даних за допомогою структурованої мови запитів SQL.

БД стали традиційно застосовуватися в наукових, інженерно-технічних, економічних, управлінських завданнях. В управлінських завданнях БД дозволяють ефективно і з легкістю працювати з БД, при цьому, скорочуючи час на обробку інформації і представляти її в зручному вигляді, використовуючи різні запити і засоби впорядкування інформації. Для створення баз даних застосовуються спеціальні програмні продукти такі як, наприклад Microsoft Access, Delphi, FoxPro і багато інших.

В якості теми дипломної роботи було вибрано створення БД «відділ кадрів». На сьогоднішній день дана тема актуальна, так як у відділах кадрів

на кожному підприємстві, організації, фірмі, будь то приватної або державної, ведеться облік персоналу.

Програма призначена полегшити і спростити дану роботу співробітників відділу кадрів.

1 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ВИРІШЕННЯ ПОСТАВЛЕНОЇ ЗАДАЧІ

1.1 Проектування автоматизованої системи управління даними для відділу кадрів.

Система кадрової документації досить специфічна. У роботі з особовим складом створюється великий обсяг однотипних документів, легко піддаються формалізації.

Створення уніфікованих форм цих документів та їх електронних версій, використання спеціальних комп'ютерних програм значно полегшує роботу всіх, хто працює з кадровими документами. Застосування комп'ютера робить можливим навіть силами одного фахівця виконує функції кадровика, вести документацію по особовому складу середніх і навіть великих організацій.

Служба кадрів сьогодні обладнана сучасною оргтехнікою, спеціалізованими програмами, що забезпечують одноразове введення інформації по особовому складу та її використання всіма підрозділами.

Впровадження комп'ютерної техніки дозволяє накопичувати масиви інформації (бази даних) і документи в електронній формі по всім співробітникам організації, кадровому резерву, швидко знаходити і ефективно обробляти всю необхідну інформацію по особовому складу.

Кадрова документація-неодмінна частина документів будь-якої організації. Вона ведеться кадровою службою (відділом, департаментом по роботі з персоналом).

Використання автоматизованих технологій дозволяє істотно підвищити ефективність роботи кадрової служби. Документи по особовому складу – це специфічна документація, що вимагає спеціалізованих програм для роботи з нею. А для складання списку вимог до системи автоматизації служби кадрів,

вибору найбільшою мірою, підходящої для неї системи програмного забезпечення (ПЗ) необхідні зусилля саме фахівців самої кадрової служби. Тому працівники служби кадрів повинні добре орієнтуватися в можливостях сучасних автоматизованих систем, в першу чергу – для визначення потреб свого підрозділу і складанні оптимального завдання для IT-служб на придбання, встановлення, налаштування і подальше обслуговування засобів автоматизації, необхідних в роботі з документацією по особовому складу.

Системи обліку кадрів розроблялися на основі програм призначених для розрахунку заробітної плати. З часом їх функціонал було значно розширено. Пов’язані такі зміни з розумінням керівників необхідності внесення змін у роботу відділу кадрів

Покращення режиму роботи персоналу, підвищення професіоналізму працівників, працюючих з персоналом, призвело до підвищення технічності та якості систем відділу кадрів. Ефективність роботи відділу кадрів безпосередньо впливає на успіх компанії. Завдання фахівців в даній сфері підібрати та завабити необхідних фахівців та розумно розподілити наявні трудові ресурси. Разом з цим збільшуються потреби в наявності ефективних інструментів управління, зберіганні та обробці інформаційних потоків. Саме з цієї причини розробникам автоматизованих систем необхідно приділяти увагу розробці все більш складних систем автоматизації управління трудовими ресурсами, приділяти увагу пошуку шляхів об’єднання необхідних задач в одну інформаційну систему для управління кадрами підприємства.

Програмні продукти даної області дозволяють:

- своєчасно отримувати та обробляти аналітичну інформацію стосовно кадрового складу підприємства та приймати зважені рішення управління;
- срганізовувати бізнес-процеси стосовно управління персоналом, зменшити необхідність повторного введення однакових даних в

системах обліку персоналу та оптимізувати працю співробітників компанії у різних службах;

- салагоджувати облік інформації та робити його більш ефективним, тим самим створюючи основу для подальшого аналізу та планування затрат на персонал;
- сести облік заснований на законодавчих актах та вимогах, мінімізувавши санкції фінансового характеру з боку фіскальних органів;

Правильно продумане програмне забезпечення буде простим і доступним для користувачів. Освоїти автоматизований процес зможе навіть студент, який не має досвіду роботи і необхідної кваліфікації.

Наявність подібних програм доводить перевагу застосування інформаційних технологій з точки зору використання ресурсу часу, економічного і соціального ефекту. Програма повинна містити якомога більше автоматизованих елементів, щоб спростити роботу з нею, різні сортування, фільтри, пошук, звіти, форми, завдяки яким буде забезпечена максимальна ефективність при роботі співробітників.

1.2 Існуючі системи управління даними.

IT-Enterprise.Кадровий облік

«IT-Enterprise.Кадровий облік» - програма масового призначення, що дозволяє в комплексі автоматизувати завдання, пов'язані з розрахунком заробітної плати персоналу і реалізацією кадрової політики, з урахуванням вимог законодавства і реальної практики роботи підприємств [2].

Вона може успішно застосовуватися в службах управління персоналом і бухгалтеріях підприємств, а також в інших підрозділах, зацікавлених в ефективній організації роботи співробітників, для управління людськими ресурсами комерційних підприємств різного масштабу.

В «IT-Enterprise.Кадровий облік» підтримується обчислення ПДФО і страхових внесків.

У програмі підтримуються електронні трудові книжки, звіти та Довідки до державних органів та соціальних фондів.

У програмі реалізована можливість реєстрації подій, пов'язаних з роботою з персональними даними (зокрема, доступу і відмови в доступі до персональних даних), включаючи інформацію про того Користувача, з яким дана подія була пов'язана.

Зручні та гнучкі механізми налаштування звітів дозволяють отримувати повну і достовірну інформацію в самих різних аналітичних розрізах, для різних категорій користувачів: керівництва, бухгалтерії, служби управління персоналом, кадрової служби та інших.

Кадри Плюс

«Кадри Плюс» - потужний професійний додаток для автоматизації роботи відділу кадрів будь-якої організації, яка дозволяє з легкістю створювати будь-які накази, заяви і звіти, моніторити рух персоналу, а також вести облік робочого часу, що дає можливість забезпечити максимальну віддачу від виконання посадових обов'язків фахівців з кадрів [3].

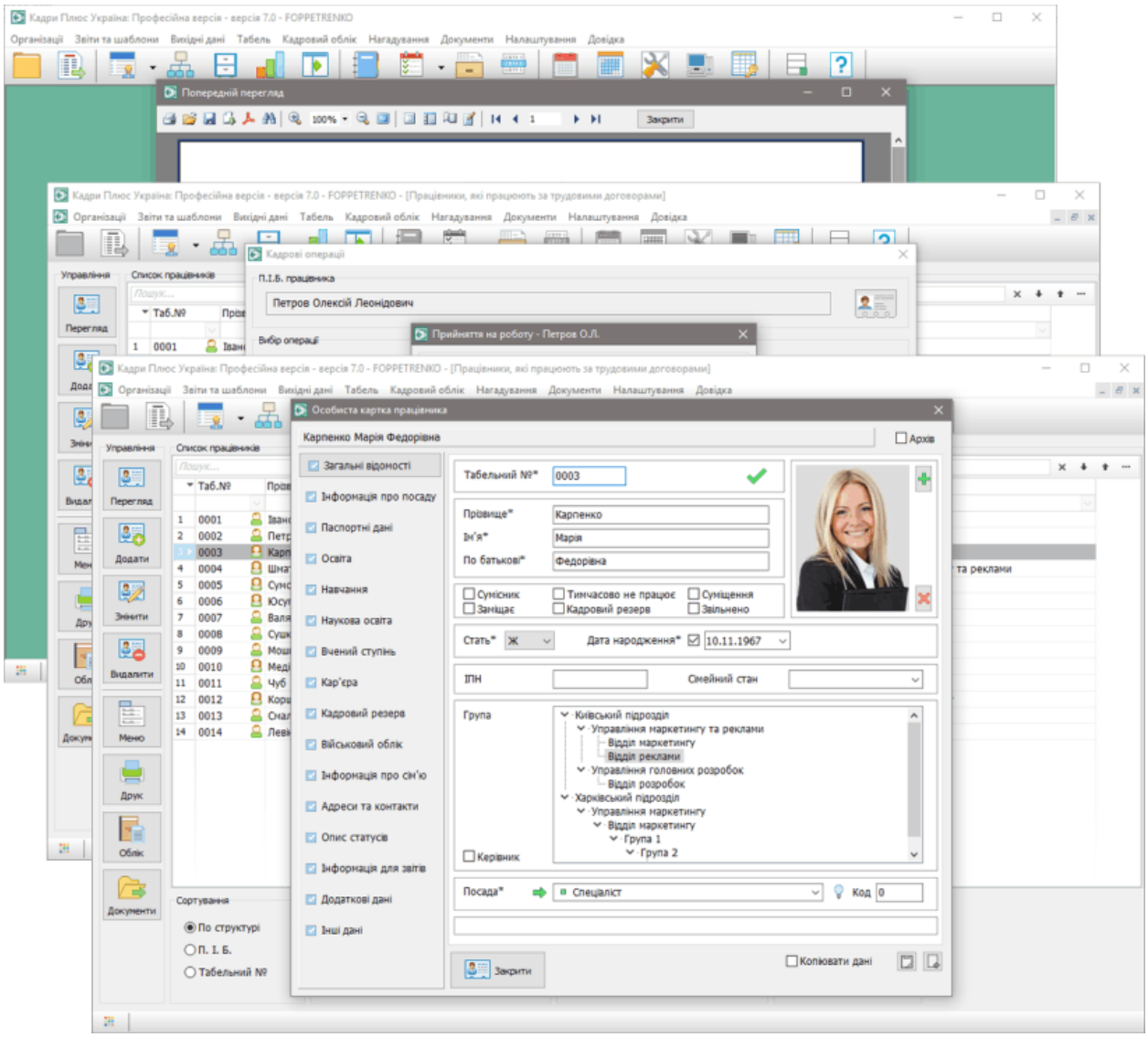


Рисунок 1.1 — Програма «Кадри Плюс»

Ключові особливості програми:

– можливість ведення необмеженої кількості організацій;

– мтруктурована система організації;

– наявність класифікатора посад;

– підтримка бази даних працівників, які влаштовані за трудовими договорами;

– підтримка бази даних працівників, яких наймали за договорами цивільно-правового характеру;

– автоматична підготовка наказів;

– веденя табелю робочого часу працівників;

– підтримка FastReport;

– облік руху співробітників і ведення кадрової статистики;

– журнал документів;

– книга обліку руху трудових книжок і вкладишів;

– розрахунок трудового стажу;

– створення звітів;

– експорт звітів і відомостей про співробітників;

– можливість завантаження з XLS-файлу списку співробітників і структури організації;

– підтримка шаблонів документів;

– шаблони типових бланків;

– підтримка роботи по мережі.

Корс-кадри

Корс-кадри – професійна програма для автоматизації кадрової служби, призначена для автоматизації кадрової діяльності, підвищення зручності роботи кадрової служби. Програми мають зручний і простий інтерфейс і надають можливості, необхідні кожному кадровому співробітнику: особисті картки співробітників, контроль робочого часу, відпусток, хвороби і т.п. Присутній режим «накази», що дозволяє швидко скласти і надрукувати будь-який наказ [4].

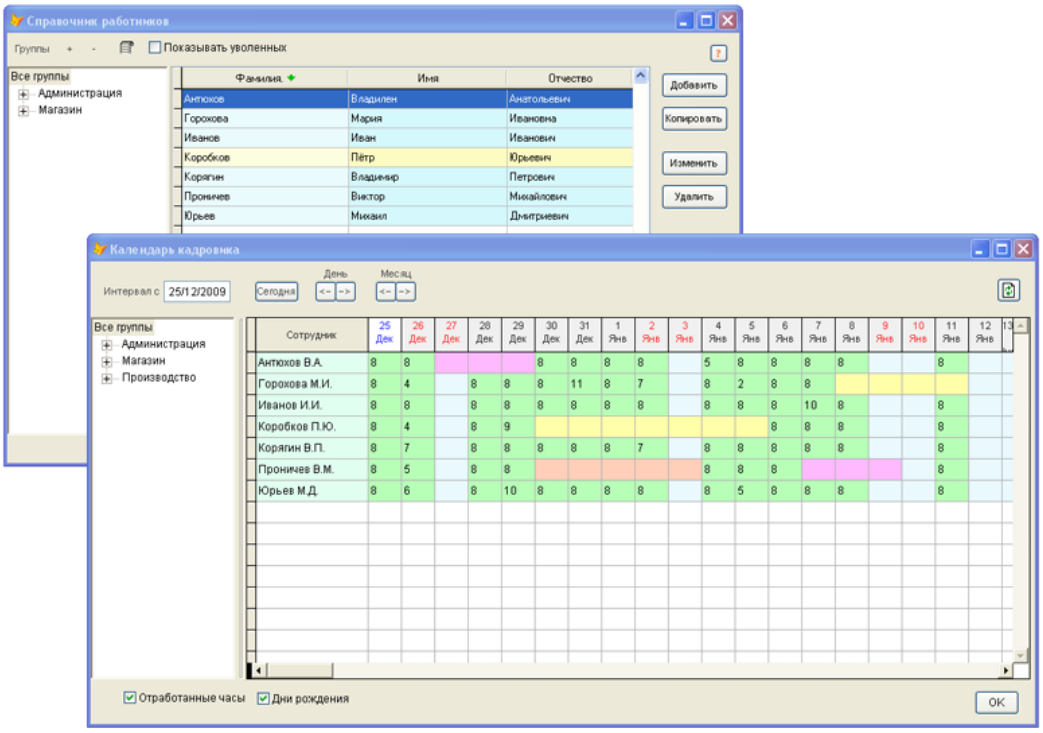


Рисунок 1.2 — Приклад роботи програми Корс-кадри

Основні можливості:

- Повний облік інформації про співробітників:

– основні дані (ПІБ, паспорт, іпн, адреса, телефон і т. п.);

– фотографія;

– освіта (повний список всіх навчальних закладів);

– володіння іноземними мовами;

– повний список професій;

– пані про військовий облік;

– повний склад сім'ї.

- Унікальний режим " календар кадровика":

– рідображення кадрового складу у вигляді " шахти "(календаря по днях);

– розфарбування клітин різними кольорами в залежності від стану співробітника в конкретний день: був на роботі, був відсутній, хворів, у відпустці і т. п.;

– відображення і ведення відпрацьованого часу;

– відображення днів народжень співробітників із зазначенням віку;

– спеціальне виділення ювілеїв;

– відображення «круглих» дат (5, 10,.. років на фірмі).

• Облік наказів:

– про прийом на роботу;

– про звільнення;

– про відпустку;

– про відрядження;

– про заохочення (в Лайтік-Кадрах і Корс-Кадрах);

– про переведення на іншу посаду (в Лайтік-Кадрах і Корс-Кадрах);

– довільна форма наказу (в Корс-Кадрах);

– фіксування всіх основних атрибутів наказів (дата, номер і т. п.);

– можливість налаштування друкованої форми наказу;

– друк наказів.

Співробітники підприємства

«Співробітники підприємства» - це безкоштовний додаток, призначений для кадрової служби. Відмінністю її від більшості подібних програм є те, що вона дозволяє вести документацію одночасно декількох організацій. Для забезпечення простої роботи з програмою розробниками було передбачена можливість взаємодії з додатком з трьох облікових записів: Адміністратор, Користувач і гість. Адміністратор має можливість створювати та редагувати записи. Користувач має можливість лише заповнювати створені раніше бази даних та документи. Облікова запис гостя можуть лише переглядати створену документацію [5].

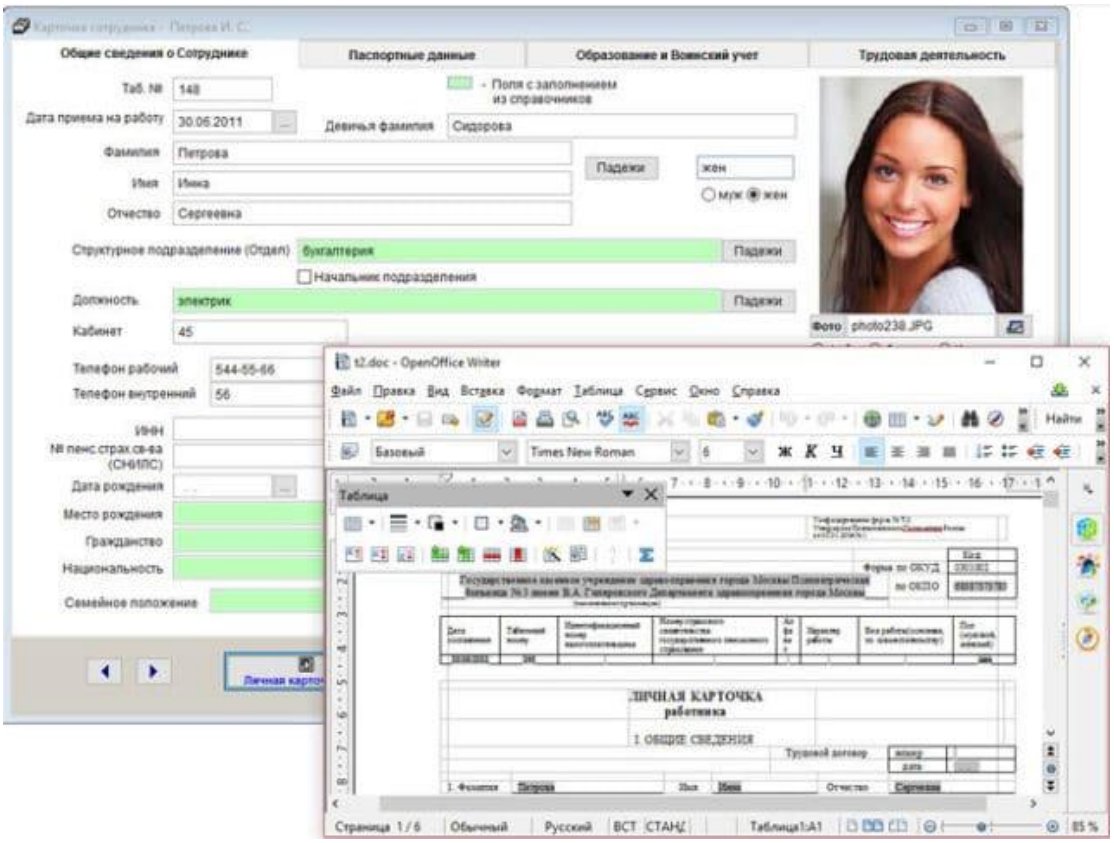


Рисунок 1.3 — Програма «Співробітники підприємства»

До основних можливостей програмки можна віднести:

– використання камери для збереження фотографій працівників;

– підключення сканера;

– створення трудових договорів;

– створення нових шаблонів документів;

– створення наказів;

– наявність можливості розрахунків відпусток та стажу праці;

– ведення табеля обліку робочого часу;

– створювати нагадування необхідності проходження медичного огляду;

– створення та зберігання даних у форматах Word і Excel;

– друк документів принтером.

Програма має доступ до локальної мережі компанії, що дозволяє працювати з системою необмеженої кількістю працівників.

Персонал бізнес

Адаптація професійної програми "Персонал-Про" для непідготовлених користувачів для ведення кадрової документації.

Адаптоване видання професійної кадрової програми "Персонал-Про", для ведення кадрового діловодства непідготовленими користувачами в організаціях, в яких відсутня самостійна кадрова служба. Робота з програмою «Персонал-бізнес» дозволяє користувачеві вести повний облік кадрів у своїй організації, дозволяє проводити друк професійних документів, а також отримання різноманітних довідок і статистичних звітів [6].

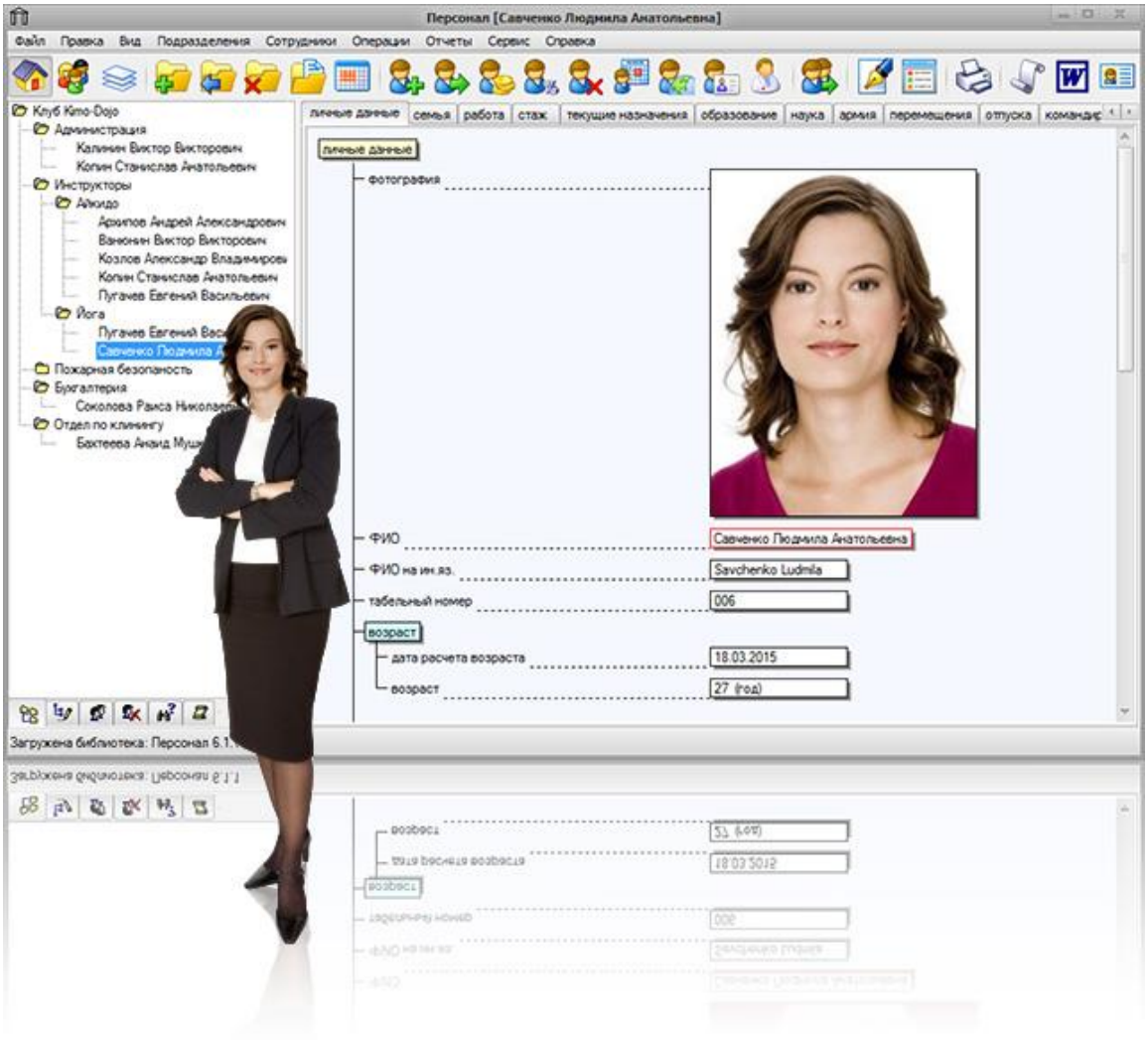


Рисунок 1.4 — Програма «Персонал Бізнес»

Переваги:

- простий інтерфейс, що гарантує створення важливих професійних. Необхідно заповнити поля потрібними атрибутами;
- інформація по всіх працівниках зберігається необмежено довго;
- база кадрових документів, з можливістю друку документів в Microsoft Word. не потрібно друкувати всі документи відразу-можливо надрукувати документи за певний період, що спрощує завдання ведення кадрового обліку. При необхідності ви можете вільно налаштувати друковані форми під ваші умови (наприклад вставити свою емблему компанії в шапку наказу) просто відредагувавши шаблон наказу в Word;
- велика кількість готових звітів і нормальних запитів на різні випадки життя, з можливістю експорту підсумків в Microsoft Excel. Потужний інтегрований механізм запитів дозволить вам самостійно витягнути будь-яку інформацію з системи за будь-який період часу;
- можна організувати на кожну компанію свою базу;
- можна працювати онлайн;
- можливість масштабування системи, тобто при збільшенні підприємства Вам не потрібно купувати нову систему - Ви зможете продовжувати вести облік в цій системі, змінивши інтерфейс системи на професійний, спрямований на обробку великого числа працівників зі складним кадровим обліком;
- безоплатна підтримка.

1.3 Недоліки існуючих системи.

Серед описаних вище програмних продуктів є як платні, так і безкоштовні. У більшості платних програм є демонстраційні приклади. Недоліком таких прикладів є обмежений функціонал, порівняно з повною версією програми.

IT-Enterprise.Кадровий облік є платним програмним додатком. Недоліками такої програми є:

- ціна програми не є остаточною через те, що вона є доповненням до бухгалтерської програми;
- перенасичений функціонал, який включає не тільки функціонал для відділу кадрів, але й функції призначені для бухгалтерського обліку.

Корс-кадри має наступні недоліки:

- програма платна;
- програма призначена лише для малих підприємств та фізичних осіб підприємств;
- не може використовувати дані з хмарного сховища;
- має лише один інтерфейс програми – Російський;
- використовується лише на операційній системі Windows.

Недоліками програм Кадри Плюс є:

- Багато застарілих зразків документів;

- Погана робота сервісної-технічної служби.

Підбиваючи підсумки аналізу існуючих програмних засобів для автоматизації роботи відділу кадрів можна виділити наступні недоліки:

- більшість програм платні;

- ціна платних програм не є остаточною та несе за собою низку додаткових трат;

- перевантаженість функціоналом, через те о програми призначені як для відділу кадрів так і для бухгалтерського обліку;

- не дуже зручний інтерфейс для користувача, через нагромадження функцій та даних;
- обмеженість функціоналу безкоштовних версій програм.

1.4 Постановка задачі.

Метою даної роботи є створення бази даних та програмного додатку для автоматизації управління даними для відділу кадрів.

База даних повинна містити та зберігати наступну інформацію:

- дані про співробітника (прізвище, ім'я, по батькові, дата народження, освіта, військовий обов'язок, адреса, телефон, посада);
- дані про вакансії (посада, назва, дата введення вакансії);
- дані про робочий графік (співробітник, відпустка / лікарняний, дата початку, дата закінчення);
- дані про проходження медогляду;
- дані про проходження курсів підвищення кваліфікації;
- журналювання дій користувачів.

Для підтримки цілісності БД необхідно:

- використання первинного ключа;
- приведення до 3 нормальної форми.

Програмний додаток повинен мати:

- простий та інтуїтивно зрозумілий інтерфейс;
- легкість освоєння (наявність довідки, розбитої на глави з використанням ілюстрацій);
- авторизація користувача повинна здійснюватися за допомогою логіну та паролю;
- організовувати взаємодію користувача з базою даних (додавання, зміна, видалення даних про співробітників, вакансії, робочий графік);
- можливість фільтрації даних по полях в таблиці;
- здійснення пошуку даних за прізвищем, ім'ям і посадою співробітника;

- можливість сортування даних по всіх стовпцях таблиць;
- друкувати анкети співробітників;
- повинна мати перевірку введення даних.

Таким чином аналіз існуючих методів проектування автоматизованої системи управління даними для відділу кадрів показав що для усунення недоліків існуючих систем потрібно вирішити такі задачі.

1. Вибрати інструменти реалізації.
2. Спроектувати автоматизовану систему.
3. Провести тестування розробленої системи.

2 ВИБІР ІНСТРУМЕНТІВ РЕАЛІЗАЦІЇ ПРОЕКТУ ТА ОБҐРУНТУВАННЯ ВИБОРУ

2.1 Transact-SQL.

SQL (Structured Query Language) - це специфічна мова програмування за допомогою якої можна створювати, змінювати, та керувати даними в реляційних базах даних (мова структурованих запитів).

SQL в його вихідному вигляді є інформаційно-логічною мовою, а не мовою програмування, але разом SQL передбачає можливість його процедурних розширень, з урахуванням яких мова вже цілком може розглядатися в якості мови програмування [7].

В даний час широко поширені наступні специфікації SQL,представлені у таблиці 1.

Таблиця 2.1 Специфікації SQL

Бази даних і специфікації SQL	
Тип бази даних	**Специфікація SQL**
Microsoft SQL	Transact-SQL
Microsoft Jet/Access	Jet SQL
MySQL	SQL/PSM (SQL/Persistent Stored Module)
Oracle	PL/SQL (Procedural Language/SQL)
IBM DB2	SQL PL (SQL Procedural Language)
InterBase/Firebird	PSQL (Procedural SQL)

Transact-SQL – це процедурне розширення мови SQL. SQL був розширений можливостями такими як:

- керуючі оператори;
- локальні та глобальні змінні;
- різні додаткові функції для обробки рядків, дат, математики і т. п.;
- підтримка аутентифікації Microsoft Windows.

Мова Transact-SQL є основною мовою при використанні SQL Server. Директиви сценарію - це спеціальні команди мови, які використовуються в MS SQL. Ці команди допомагають серверу визначати правила обробки скриптів і транзакцій. Типовими представниками являються: GO - сигналізує SQL-серверові про завершення сценарію, EXEC (або EXECUTE) – вказує на виконання процедури або скалярної функції.

Коментарі використовуються для створення пояснень до частин коду запитів, а також для тимчасового відключення від виконання команд при від лагоджені скрипта. Коментарі бувають строковими, та блоковими:

-- - строковий коментар виключає з виконання тільки одну строку, перед якою стоять два мінуси.

/* */ - блоковий коментар виключає з виконання цілий блок команд, укладений в зазначену конструкцію.

Як і в інших мовах програмування, SQL має типізовану структуру змінних:

Числа – зберігають числові значення (int, tinyint, smallint, bigint, numeric, decimal, money, smallmoney, float, real).

Дати – зберігають дату та час (datetime, smalldatetime).

Символи – використовуються для збереження символьних та строкових даних (char, nchar, varchar, nvarchar).

Двійкові – слугують для збереження двійкових даних (binary, varbinary, bit).

Велико об’ємні - тип даних використовується для збереження великих двійкових даних таких як зображення (text, n-текст, image).

Спеціальні - покажчики (cursor), 16-байтове шістнадцятирічне число, яке використовується для GUID (uniqueidentifier), штамп зміни рядки (timestamp), версія рядка (rowversion), таблиці (table).

Для використання кирилиці використовують типи даних перед якою є приставка "n" (nchar, nvarchar, n-текст), які дозволяють кодувати символи двома байтами. Тобто щоб працювати з Unicode використовуються типи даних з "n".

Змінні використовуються у сценаріях і для зберігання тимчасових даних. Щоб працювати з змінної, її потрібно оголосити, притому оголошення повинно бути здійснено в тій транзакції, в якій виконується команда, що використовує цю змінну. Інакше кажучи, після завершення транзакції, тобто після команди GO, мінлива знищується.

Оголошення змінної виконується командою DECLARE, завдання значення змінної здійснюється або командою SET, або SELECT.

Оператори - це спеціальні команди, призначені для виконання простих операцій над змінними:

Арифметичні оператори: "*" - множення, "/" - ділення, "%" - модуль від ділення, "+" - сума , "-" - різниця, "()" - дужки.

Оператори порівняння: "=" - дорівнює, ">" - більше, "<" - менше, ">=" - більше або дорівнює, " < = менше або дорівнює, "<>" - не дорівнює.

Оператори з'єднання: "+" - з'єднання рядків.

Логічні оператори: "AND" - і, "OR"- або ,"NOT" - не.

Специфікація Transact-SQl значно розширює стандартні можливості SQL завдяки вбудованим функціям:

Агрегатовані функції - функції, які дозволяють працювати з колекціями значень і повертають одне значення. Представниками таких функцій є: AVG - середнє значення колонки, SUM - сума колонки, MAX - максимальне значення колонки, COUNT - кількість елементів колонки.

Скалярні функції - це функції, на виході яких одне значення, вхідними параметрами можуть бути скалярні дані або без вхідних параметрів.

Представниками є: DATEDIFF - різниця між датами, ABS - модуль числа, DB_NAME - ім'я бази даних, USER_NAME - ім'я поточного користувача, LEFT - частина рядка ліворуч.

Функції-вказівники – функції, які використовуються як посилання на інші дані. Представниками є: OPENXML – вказівник на джерело даних у вигляді XML-структури, OPENQUERY – вказівник на джерело даних у вигляді іншого запиту.

До скалярних функцій можна також віднести і глобальні змінні, які в тексті сценарію викликаються подвійний собакою "@@".

Вираз - це комбінація символів і операторів, яка отримує на вхід скалярну величину, а на виході дає іншу величину або виконує якусь дію. В Transact-SQL вирази поділяються на 3 типи: DDL, DCL і DML.

DDL (Data Definition Language)- використовується для створення нових об'єктів у базі даних. Основні представники даного класу являються: CREATE - створення об'єктів, ALTER - зміна об'єктів, DROP - видалення об'єктів [8].

DCL (Data Control Language) – призначені для призначення прав на об'єкти бази даних. Основні представники цього класу: GRANT - дозвіл на об'єкт, DENY - заборона на об'єкт, REVOKE - скасування дозволів і заборон на об'єкт.

DML (Data Manipulation Language)- використовуються для запитів і зміни даних. Основні представники цього класу: SELECT – вибірка даних, INSERT - вставка даних, UPDATE – зміна даних, DELETE - видалення даних.

В Transact-SQL існують спеціальні команди, які дозволяють керувати потоком виконання сценарію, перериваючи його або направляючи в потрібну логіку.

Блок угруповання - структура, що об'єднує список виразів в один логічний блок (BEGIN ... END).

Блок умови - структура, що перевіряє виконання певної умови (IF ... ELSE).

Блок циклу - структура, що організує повторення виконання логічного блоку (WHILE ... BREAK ... CONTINUE).

Перехід - команда, яка виконує перехід потоку виконання сценарію на зазначену позначку (GOTO).

Затримка - команда, що затримує виконання сценарію (WAITFOR)

Виклик помилки - команда, що генерує помилку сценарію (RAISERROR)

Зазвичай бази даних створюються і заповнюються за допомогою сценаріїв (скриптів) - хоча візуальний редактор простий у використанні, але їм ніколи швидко і без недоліків не створиш велику базу даних і не заповниш її даними.

Сценарій - це одне або більше виразів, об'єднаних в логічний блок, які автоматизують роботу адміністратора.

Зазвичай сценарії пишуться як універсальний засіб для виконання стандартних завдань, тому в них застосовується динамічне конструювання логіки - запити і команди вставляє змінні, а не конкретні назви об'єктів, що дозволяє швидко змінювати параметри скрипту.

SQL дозволяє здійснювати групування даних за певними полями таблиці.

Щоб групувати дані по якому-небудь параметру, в SQL-запиті необхідно написати команду GROUP BY, в якій вказати ім'я колонки, за якою проводиться групування.

Колонки, згадані в команді GROUP BY, повинні бути присутніми в команді SELECT, а так же команда SELECT повинна містити функцію агрегування, яка буде застосована до згрупованих даних [9].

Щоб відфільтрувати рядка в запиті з угрупованням застосовується спеціальна команда HAVING, в якій вказується умова фільтрації. Колонки, за якими здійснюється фільтрація, повинні бути присутніми в команді GROUP

BY. Команда HAVING може використовуватися і без GROUP BY, в цьому випадку вона працює аналогічно команді WHERE, але вона дозволяє застосовувати в умовах фільтрації тільки функції агрегування.

Команда угруповання має можливість доповнюватися оператором WITH CUBE, який у свою чергу дозволяє доповнювати формувати різноманітні комбінації з групування колонок: якщо є N колонок, то вийде 2^N комбінацій.

Ще одна команда угруповання COMPUTE дозволяє групувати дані і виводити з них звіт в різні таблиці. Тобто команда GROUP BY з операторами ROLLUP і CUBE групує дані і дописує в таблицю додаткові рядки зі звітом, а команда COMPUTE групує дані, розриваючи вихідну таблицю на кілька під таблиці, а також формує під таблиці з звітами.

Команда COMPUTE може використовуватися в двох режимах:

1. як проста функція агрегування, виводить результат в окрему таблицю;
2. з параметром BY як команда угруповання, розрізаючи таблицю на кілька під таблиці.

Команда COMPUTE з параметром BY може використовуватися тільки спільно з командою ORDER BY, причому стовпці сортування повинні збігатися зі стовпцями угруповання.

Найбільш важливі та потрібні запити в SQL - це з запити з з'єднанням таблиць, коли вибірка здійснюється відразу з декількох джерел.

Такі запити більш складні в написанні, але і більш зручні в обробці, так як часто видають у програму вже готовий результат, який залишається тільки вивести на екран.

З'єднувати таблиці в SQL можна двома способами: вертикально і горизонтально.

Вертикальне з'єднання здійснюється командою UNION, яка в кінець першої таблиці допише другу таблицю.

При такому з'єднанні кількість колонок з'єднуються таблиць повинно бути однаковим, а самі колонки повинні мати однакові назви і типи даних.

При з'єднанні однакові рядки, що зустрічаються в обох таблицях, будуть видалені, якщо в команді не вказано параметр ALL.

Горизонтальне з'єднання проводиться шляхом зчеплення декількох таблиць за ключовими колонками.

Найпростіше горизонтальне з'єднання виконується за допомогою команди INNER JOIN, яка зв'язує таблиці, вибираючи рядки по ключовому полю, яке зустрічається в обох таблицях.

Щоб виконати зчеплення по всіх полях лівої таблиці, незалежно, чи є такі записи в правій таблиці, необхідно використовувати команду LEFT JOIN. Ця команда з'єднує таблиці, вибираючи всі рядки з лівої таблиці, а відсутні дані правої таблиці заповнюються NULL.

Команда RIGHT JOIN аналогічна попередній, різниця полягає лише в тому, що вона поєднує таблиці, вибираючи всі рядки з правої таблиці, а відсутні дані лівої таблиці заповнюються NULL.

Команда FULL JOIN об'єднує в собі ліве і праве зчеплення, тобто вона з'єднує таблиці, вибираючи рядків з обох таблиць, а відсутні дані заповнюються NULL.

Зчеплення не обмежується тільки двома таблицями, запит може містити декілька команда JOIN, що дуже зручно при формуванні кінцевих звітів.

Діалект Transact-SQL – механізм заснований на транзакції, тобто на послідовності операцій, об'єднаних в один логічний модуль, будь то запит на вибірку даних, зміни даних або структури таблиці.

На час транзакції всі використовувані в сценарії дані блокуються, що дозволяє уникнути не відповідностей даних під час початку роботи з таблицею і завершенням сценарію [10].

За транзакції в Transact-SQL відповідає структура BEGIN TRANSACTION ... COMMIT TRANSACTION. Цю структуру

використовувати необов'язково, але тоді всі команди сценарію є необоротними, тобто не можна зробити "відкат" до попереднього стану.

Для збільшення продуктивності, тобто для швидкого виконання запитів, слід пам'ятати деякі правила складання рядків запитів:

Уникати NOT - команди заперечення виконуються в кілька етапів, що збільшує навантаження на сервер.

Уникати LIKE - цей оператор порівняння застосовує більш м'які шаблони порівняння, ніж оператор =, що збільшує необхідну кількість етапів фільтрації.

Застосовувати точні шаблони пошуку - застосування символів збільшує час виконання запиту, так як для перевірки всіх варіантів підстановки потрібно додаткові ресурси сервера.

Уникати ORDER - команда сортування вимагає упорядкування рядків таблиці висновку, що затримує отримання результату.

2.2 Microsoft SQL Server.

MS SQL Server — система створена для керування реляційними базами даних (СУБД), спроектована компанією Microsoft. Головна мова запитів що використовується -Transact-SQL, створена компаніями Microsoft і Sybase разом. Transact-SQL є реалізацією структурованої мови запитів (SQL) з розширеннями згідно стандарту ANSI / ISO. застосовується при роботі з маленькими і середніми за розміром базами даних до великих баз даних масштабу підприємства; змагається з іншими СУБД в цьому сегменті ринку [11].

SQL є спільним інтерфейсом до баз даних. Всі індустріальні бази-Oracle, Microsoft SQL Server, PostgreSQL, MySQL-працюють на SQL.

До того часу, як вийшла на ринок ОС Windows NT, Sybase і Microsoft розійшлися і слідували власним моделям програмного продукту і маркетингових схем. Microsoft прагнула до виключних прав на всі версії SQL Server для Windows. Пізніше Sybase змінила назву свого продукту на

Adaptive Server Enterprise щоб уникнути плутанини з Microsoft SQL Server. До 1994 року Microsoft отримала від Sybase три повідомлення про авторські права як натяк на походження Microsoft SQL Server.

Після розділення компанії зробили кілька самостійних релізів програм. SQL Server 7.0 був першим сервером баз даних з графічним інтерфейсом користувача та адміністрування.

Запуск SQL Server 2005 відбувався разом з запуском Visual Studio 2005. Існує також версія Microsoft SQL Server Express яка доступна для скачування безкоштовно і поширюється разом із програмним забезпеченням.

Після SQL Server 2000 було здійснено впровадження інтегрованого середовища розробки та кілька додаткових підсистем. Змінилась реалізація технології ETL (перетворення , витягання і завантаження даних), що включені до складу додатка SQL Server Integration Services (SSIS), засобів аналітичного опрацювання багатовимірних моделей даних (OLAP) і збору інформації що до них відносится, сервера оповіщення, а ще деяких служб сповіщень, а саме Notification Services і Service Broker. Значно збільшилась продуктивність.

Функціональність

Microsoft SQL Server використовує версію SQL, що отримала назву Transact-SQL (скорочено T-SQL), що є реалізацією стандарта ISO SQL під номером SQL-92 з багатьма розширеннями. Microsoft SQL Server і Sybase ASE працюють по протоколу додатка під назвою Tabular Data Stream (TDS, протокол передачі табличних даних) для звязку с мережею.

Microsoft SQL Server також підтримує Open Database Connectivity (ODBC) — інтерфейс взаємодії додатків з СУБД. Версія SQL Server 2005 забезпечує підключення юзерів через веб-сервіси за допомогою протокола SOAP. Так клієнтські програми, не призначені для Windows, крос платформено взаємодіють з SQL Server. За дпомогою драйвера JDBC додатки під управлінням Java можуть об'єднуватися Microsoft SQL Server 2000, 2005 [12].

SQL Server має функції віддзеркалення і кластеризації баз даних. Кластер сервера SQL — являє собою деяку кількість однаково конфігурованих серверів міє якими розподіляється робоче навантаження. Дані розподіляються по машинам кластера протягом робочого циклу і всі сервери мають одне віртуальне ім'я. Якщо виникають проблеми на одному з серверів кластеру доступне перенесення навантаження на робочий сервер автоматично.

SQL Server підтримує надлишкове дублювання даних за трьома сценаріями:

1. Знімок: Проводиться «знімок» бази даних потім сервер відправляє його одержувачам.
2. Історія змін: Усі зміни бази даних безперервно передаються користувачам.
3. Синхронізація з іншими серверами. Зміни відбуваються незалежно один від одного на всіх серверах, а при синхронізації відбувається звірка. Даний тип дублювання дає опцію вирішення протиріч між БД.

Завдяки вбудованій пдтримкі .NET Framework у SQL Server 2005 збережені процедури БД можуть бути написані будь-якою мовою програмування платформи .NET, використовуючи набір бібліотек, доступних для .NET Framework, включаючи Common Type System (система поводження з типами даних в Microsoft .NET Framework). Завдяки алгоритмам розподілу ресурсів які спеціально налаштовані для використання в структурах SQL Server підвищується продуктивність в порівнянні з загальними алгоритмами Windows [13].

Microsoft розробив нову технологію in-memory, яка додана в SQL Server. Технологія отримала назву Hekaton. Про це 7 листопада 2012 року повідомив ComputerWorld.

Microsoft в прагненні прискорити процеси оперативної обробки транзакцій (OLTP) додав в SQL Server можливість використовувати реляційні системи управління базами даних.

Вже в наступній версії SQL Server включена можливість розміщення частини таблиць баз даних або навіть всієї бази даних в пам'яті сервера. Так само додані додаткові інструменти для спрощеного запуску технології.

Microsoft стверджує, що сервер буде швидше виконувати операції, якщо необхідні таблиці бази даних в пам'яті, а не записані на диск, до якого необхідно буде звертатися. Гігант впевнений, що технологія дозволить збільшити швидкість обробки даних до 50 разів порівняно з аналогічними системами для SQL Server.

Основними напрямками для використання Hekaton є банківські системи онлайн, ERP, та інші транзакційні системи, яким необхідно оперативно зв'язуватися і використовувати бази даних. Технологія може бути встановлена на одному сервері і далі масштабована на інші сервера, оскільки вона не має жорстких обмежень щодо використовуваної пам'яті.

Вихід Hekaton може стати серйозною головним болем для таких компаній як Oracle з її продуктом Oracle Exadata і для SAP, зокрема SAP HANA. Це обумовлено тим, що технологія значно спрощує IT-архітектуру і знімає необхідність докуповувати компоненти для обробки даних, як це реалізовано у конкурентів.

Дуг Леланд стверджує, що Hekaton – не перший досвід Microsoft в роботі з технологіями in-memory. Оскільки в офісному додатку Microsoft Excel використовуються технології PowerPivot і Power View, що дозволяють оперативно маніпулювати великим об'ємом даних.

Microsoft також оголосила про швидкий вихід наступної версії Data Warehouse Appliance, SQL Server 2012 Parallel Data Warehouse (PDW). А для SQL Server 2012 був випущений пакет оновлень, який, зокрема, включає можливість користувачам Exel 2013 працювати з даними, що зберігаються на SQL Server.

Розробка додатків

Microsoft та інші компанії виробляють велику кількість програмних засобів розробки, що дозволяють розробляти бізнес-додатки з використанням баз даних Microsoft SQL Server. Microsoft SQL Server 2005 включає в себе також Common Language Runtime (CLR) Microsoft .NET, що дозволяє реалізовувати збережені процедури і різні функції додатками, розробленими на мовах платформи .NET (наприклад, VB.NET або C#). Попередні версії засобів розробки Microsoft використовували тільки API для отримання функціонального доступу до Microsoft SQL Server.

SQL Server Express Edition

Microsoft SQL Server Express є безкоштовно поширюваною версією SQL Server, розвитком системи MSDE. Вона використовується в додатках, при проектуванні або для самостійного вивчення. У 2007 році Microsoft випустила окрему утиліту з графічним інтерфейсом для адміністрування цієї версії, яка також доступна для безкоштовного скачування з сайту корпорації.

2.3 Мова програмування C#

Мова програмування C# була розроблена в 1993-2001 роках командою інженерів компанії Microsoft під керівництвом Андерса Хейлсберга і Скотта Вильтаумота як мова розробки додатків для платформи Microsoft.

До 2000 року у Microsoft були готові промислові версії нових технологій і рішень для обміну повідомленнями і даними, а також для створення Internet-додатків. Була випущена нова платформа для розробки під нові рішення — .NET. У ній об'єдналися відразу декілька мов програмування, що було в іноваційно для того часу [14].

Одним з нововведенням платформи .NET була технологія активних серверних сторінок ASP.NET (Active Server Page). З її допомогою можна було відносно швидко розробити веб-додаток, взаємодіючий з базами даних. Спеціально для ASP.NET була створена мова програмування C#. Та й сама ASP.NET була повністю написана на ньому.

C# — це мова з C-подібним синтаксисом. Тут він близький у цьому відношенні до C++ і Java.

Будучи об'єктно-орієнтованою мовою, вона багато перейняла у Java і C++. Як і Java, C# спочатку призначався для веб-розробки, і приблизно 75% його синтаксичних можливостей такі ж, як у Java. C# також називають «очищеної версією Java». Ще 10% наш герой запозичив з C++ та 5% – з Visual Basic. Решта 10% C# — це реалізація власних ідей розробників. Об'єктно-орієнтований підхід дозволяє будувати за допомогою C# великі, але в той же час гнучкі, масштабовані і розширювані програми.

C# вже давно підтримує багато корисних функцій:

– інкапсуляція;

– спадкування;

– поліморфізм;

– перевантаження операторів;

– статична типізація.

При цьому він все ще активно розвивається, і з кожною новою версією з'являється все більше цікавого — наприклад лямбди, динамічне зв'язування, асинхронні методи і т. п.

Коли говорять C#, нерідко мають на увазі технології платформи .NET Windows Forms, WPF, ASP.NET, Xamarin). І навпаки, коли говорять .NET, нерідко мають на увазі C#. Однак, хоча ці поняття пов'язані, ототожнювати їх неправильно. Мова C# була створена спеціально для роботи з фреймворком .NET, проте саме поняття .NET дещо ширше.

Фреймворк .NET являє потужну платформу для створення додатків. Ось його особливості:

Підтримка декількох мов. В основі .NET — загальномовне середовище виконання Common Language Runtime (CLR), завдяки чому платформа підтримує кілька мов: поряд з C# це VB.NET, C++, F#, а також різні діалекти інших мов, прив'язані до .NET, наприклад, Delphi.NET. Код на будь-якому з цих мов компілюється в збірку на загальному мовою CIL (Common

Intermediate Language) — свого роду асемблер платформи .NET. Тому можна зробити окремі модулі однієї програми на різних мовах.

Потужна бібліотека класів. .NET являє єдину для всіх підтримуваних мов бібліотеку класів.

Різноманітність технологій. Загальномовне середовище виконання CLR і базова бібліотека класів — це основа для цілого стека технологій, які розробники можуть задіяти при створенні різних додатків.

.NET Framework і .NET Core

.NET довгий час розвивався під назвою .NET Framework — переважно як платформа для Windows. Але з 2019 вона більше не розвивається — останньою версією цієї платформи стала .NET Framework 4.8.

У 2014 Microsoft почав випускати альтернативну платформу – .NET Core, яка повинна була увібрати в себе всі можливості застарілого .NET Framework і додати нову функціональність. Тому слід розрізняти .NET Framework, який призначений переважно для Windows, і багато платформовий .NET Core.

Переваги і недоліки мови C#

У C# виділяють багато переваг:

- підтримка переважної більшості продуктів Microsoft;
- розповсюдження безкоштовно інструментів для малих компаній;
- типи даних мають фіксований розмір (32-бітний int і 64-бітний long), що підвищує мобільність» мови і спрощує програмування, так як ви завжди точно знаєте, з чим ви маєте справу;
- автоматична «збірка сміття» Це означає, що нам в багатьох випадках не доведеться піклуватися про звільнення місця в пам'яті. Вищезгадана загальномовне середовище CLR сама викличе збирач сміття і очистить пам'ять.
- з допомогою Xamarin на C# можна писати програми для таких операційних систем, як iOS, Android, MacOS і Linux;

– сьогодні в будь-якому регіоні є чимало вакантних місць на посаду C#-програміста.

Але є в C# і деякі недоліки:

– пріоритетна орієнтованість на платформу Windows;

– мова безкоштовна тільки для невеликих фірм, індивідуальних програмістів, стартапів і учнів . Великої компанії купівля ліцензійної версії цієї мови обійдеться в досить велику суму.

C# не представляє складності для новачків, так як його порівняно легко вивчити і зрозуміти. На просторах інтернету можна знайти безліч курсів і онлайн-шкіл по навчанню C#, що пропонують навчання тривалістю від 1 до 6 місяців. Також існують експрес-курси для «чайників», де новачків навчають основам за пару днів. Взагалі, C# набагато простіше буде засвоїти, якщо ви вже знаєте C, C++ або Java[15].

C# на протязі довгого часу впевнено тримає позиції в рейтингу найбільш затребуваних на ринку розробки мов. Спочатку їм цікавилися тільки розробники під Windows, але потім C# навчилася працювати на Mac OS, Linux, iOS і Android. А після того, як код платформи відкрили для всіх бажаючих, були зняті практично усі можливі обмеження в застосуванні C#. В результаті мова активно розвивається і застосовується усе ширше. Його часто рекомендують до вивчення в якості одного з базових для розробників будь-якого профілю.

Набір інструментів вбудованих в C# дозволяє виконувати широке коло завдань, мова дуже потужна і універсальна. На ній часто розробляють:

– веб-додатки;

– ігри;

– мобільні додатки для Android або iOS;

– програми під Windows.

Перелік можливостей розробки практично не має обмежень завдяки найширшому набору інструментів і засобів. Звичайно, все це можна

реалізувати за допомогою інших мов. Але деякі з них вузькоспеціалізовані, а в деяких доведеться використовувати додаткові інструменти сторонніх розробників. В C# вирішити широке коло завдань можливо легше, швидше і з меншими витратами ресурсів і часу.

2.4 Microsoft Visual Studio 2019.

Microsoft Visual Studio-лінійка товарів фірми Microsoft, що включають інтегроване середовище розробки програмного забезпечення та ряд інших приладових засобів [16].

Visual Studio охоплює в собі редактор вихідного коду з підтримкою технології IntelliSense і можливістю простого рефакторингу коду. інтегрований настроювач зможе діяти як відладчик рівня початкового коду, так і відладчик машинного рівня. Інші інтегровані інструменти включають в себе більдредактор конфігурацій для спрощення створення графічного інтерфейсу програми, веб-редактор, конструктор класів і конструктор схеми бази даних.

Visual Studio дозволяє організовувати і підключати сторонні доповнення (плагіни) для розширення функціональності практично на будь-якому рівні, включно приєднання підтримки систем контролю версій початкового коду, приєднання нових комплектів інструментів або інструментів для інших аспектів процесу розробки програмного забезпечення.

Багато варіантів поставки включають:

- Microsoft SQL Server або Microsoft SQL Server Express — засоби для роботи з базами даних.

Висновки до розділу 2

Провівши аналіз мов програмування баз даних те середовищ їх розробки. У даній роботі буде використовуватись середовище Microsoft SQL Server 2019. Та мова програмування запитів Transact-SQL. Через те що, Microsoft SQL Server 2019 Express-потужна і надійна безкоштовна система управління даними, що забезпечує функціональне і надійне сховище даних для веб-сайтів і настільних додатків.

Для створення клієнтського додатку буде використовуватись середовище розробки Microsoft Visual Studio 2019 та мова програмування C#.

Для взаємодії програми та бази даних та спрощення програмного коду буде застосовуватись технологія Entity Framework.

Entity Framework представляє специфічну об'єктно-орієнтовану технологію на базі фреймворку . NET для роботи з даними . Якщо класичні засоби ADO .NET дозволяють організовувати підключення, директиви та інші предмети для взаємодії з базами даних, то Entity Framework являє собою більше високий рівень абстракції, який дозволяє абстрагуватися від самої бази даних і діяти з даними незалежно від типу сховища. Якщо на фізичному рівні ми оперуємо таблицями, індексами, первинними і зовнішніми ключами, але на концептуальному рівні, який нам пропонує Entity Framework, ми вже працює з об'єктами.

Перша версія Entity Framework-1.0 вийшла ще в 2008 році і представляла досить обмежену функціональність, базову підтримку ORM (object - relational mapping - передача даних на реальні об'єкти) і один єдиний підхід до взаємодії з бд-Database First. З виходом версії 4. 0 у 2010 році багато що змінилося - з цього часу Entity Framework став рекомендованою технологією доступу до даних, а сам фреймворк були введені нові можливості взаємодії з БД - підходу Model First і Code First.

3 РЕАЛІЗАЦІЯ ПРОЄКТУ

3.1 Опис вхідної та вихідної інформації.

Вхідною інформацією для розробляємо бази даних буде повна інформація про людину.

Перш за все будуть необхідні паспортні данні :

ПІБ, дата народження, серія/номер паспорту, ким видано, дата видачі, місце реєстрації та фактичного проживання, номер телефону та ІПН.

Далі необхідно буде вводити інформацію про освіту:

Тип освіти, заклад, спеціальність, період навчання, форма навчання, номер отриманого свідоцтва/диплому.

Сімейний стан передбачає таку інформацію як:

Статус, ПІБ членів родини, наявність дітей та їх дати народження.

Для чоловіків обов’язковими є данні про військову службу:

Категорія запасу, звання, придатність, військкомат, відмітка про зняття з реєстру.

Також потрібно буде вводити данні про минуле місце роботи :

Місце, період, спеціальність та причини звільнення.

При прийнятті обов’язково вказується у який відділ та на яку спеціальність буде прийнята людина.

Вихідною інформацією є сформована база даних працівників та можливість її перегляду.

3.2 Проектування та реалізація бази даних.

При розробці структури бази даних інформацію по кадровому складу раціонально представити у вигляді сукупності таблиць.

Таблиця CategoriesReserve містить інформацію про категорії запасу.

Таблиця 3.1 Специфікація таблиці CategoriesReserve

ID	**int**	**Ідентифікатор**
cat_reserve	nvarchar(20)	Категорія запису

Таблиця Контакт містить інформацію про:

- реєстрацію;
- фактичне проживання;
- номер телефону.

Таблиця 3.2 Специфікація таблиці Contacts

ID	**int**	**Ідентифікатор**
Registration	nvarchar(50)	Адреса реєстрації
home_address	nvarchar(50)	Фактична адреса
Phone	nvarchar(50)	телефон
employee_id	int	ІД працівника

Таблиця Відділ містить інформацію про назви відділів.

Таблиця 3.3 Специфікація таблиці Departments

ID	**int**	**Ідентифікатор**
departments	nvarchar(50)	Відділ

Таблиця EducationForm інформацію про тип навчання

Таблиця 3.4 Специфікація таблиці EducationForm

ID	**int**	**Ідентифікатор**
form	nvarchar(50)	Тип навчання

Таблиця Educations містить інформацію про освіту

Таблиця 3.5 Специфікація таблиці Educations

ID	**int**	**Ідентифікатор**
type_id	int	Тип навчання
form_id	int	Форма навчання
Institution	nvarchar(100)	Заклад
Profession	nvarchar(50)	Професія
Datestart	datetime	Дата вступу
Dateend	datetime	Дата випуску
number_document	nvarchar(30)	Номер документу
employee_id	int	ІД працівника

Таблиця EducationType містить інформацію про тип освіти.

Таблиця 3.6 Специфікація таблиці EducationType

ID	**int**	**Ідентифікатор**
form	nvarchar(50)	Тип освіти

Таблиця Employees містить інформацію про працівника.

Таблиця 3.7 Специфікація таблиці Employees

ID	**int**	**Ідентифікатор**
Name	nvarchar(50)	ПІБ
sex	smallint	Стать
birth_date	datetime	Дата народження
employment_date	datetime	Дата прийняття
foto	varbinary(MAX)	Фотографія
Inn	nchar(50)	ІПН
post_id	int	ІД служби
family_status	int	ІД родини

Таблиця FamilyStatus містить інформацію про родинний статус.

Таблиця 3.8 Специфікація таблиці FamilyStatus

ID	**int**	**Ідентифікатор**
family_status	nvarchar(30)	Статус родини

Таблиця FamilyUnits містить інформацію про членів родини.

Таблиця 3.9 Специфікація таблиці FamilyUnits

ID	**int**	**Ідентифікатор**
employee_id	int	ІД працівника
type_id	int	ІД статусу
Name	nvarchar(50)	ПІБ
birth_date	datetime	Дата народження

Таблиця FamilyUnitTypes містить інформацію тип родинних зв’язків.

Таблиця 3.10 Специфікація таблиці FamilyUnitTypes

ID	**int**	**Ідентифікатор**
type	nvarchar(20)	тип родинних зв’язків

Таблиця LastPlaceOfEmployment містить інформацію останнє місце роботи

Таблиця 3.11 Специфікація таблиці LastPlaceOfEmployment

ID	**int**	**Ідентифікатор**
post_id	int	ІД посади
place_of_employment	nvarchar(50)	Місце роботи
Datestart	datetime	Дата прийому
Dateend	datetime	дата звільнення
cause_discharge	nvarchar(50)	Причина звільнення
employee_id	int	ІД працівника

Таблиця Passports містить паспортні данні

Таблиця 3.12 Специфікація таблиці Passports

ID	**int**	**Ідентифікатор**
series	nvarchar(10)	Серія
number	int	Номер
issued	nvarchar(50)	Ким видано
dateissue	datetime	Коли видано
employee_id	int	ІД працівника

Таблиця MilitaryRanks містить інформацію про звання військової служби.

Таблиця 3.13 Специфікація таблиці MilitaryRanks

ID	**int**	**Ідентифікатор**
rank	nvarchar(50)	звання

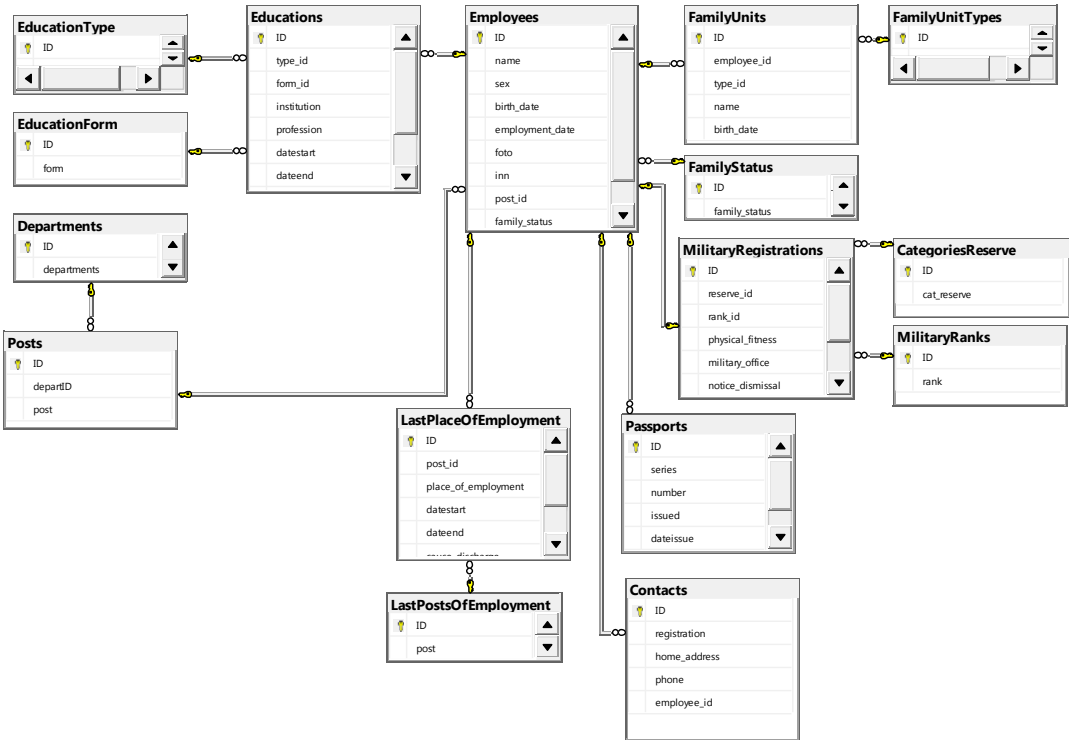


Рисунок 3.1 — Логічна схема БД

На рисунку 3.1 та у додатку А представлена Логічна схема бази даних.

База даних містить у собі 16 таблиць які приведені до 3НФ та мають структурну цілісність.

3.3 Проектування та реалізація програмного забезпечення.

Розроблений програмний клієнтський додаток повинен мати приємним візуальним і інтуїтивно зрозумілим інтерфейсом.

При старті програми відбувається автоматичне підключення до бази даних. Після чого можливо відразу з нею працювати

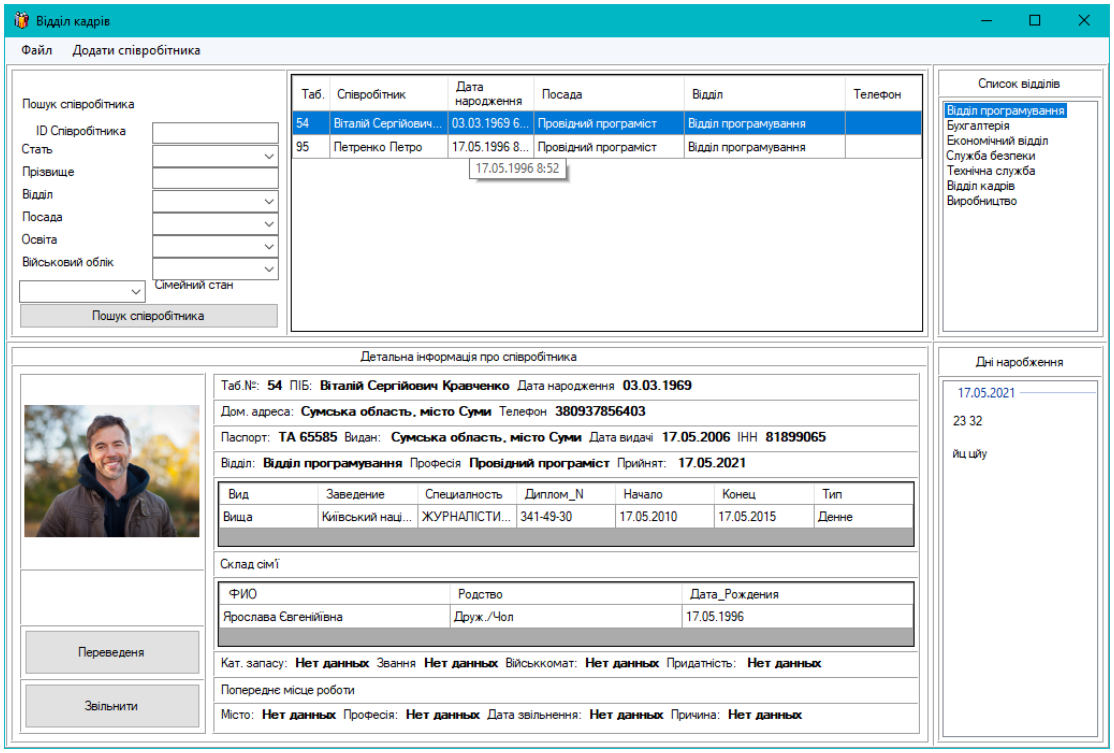


Рисунок 3.2 — Головне вікно програми

На головному вікні розташовані фільтри за якими можна знаходити конкретних працівників та спеціальні поля які виводять про нього інформацію:

– паспортні данні : ПІБ, дата народження, серія/номер паспорту, ким видано, дата видачі, місце реєстрації та фактичного проживання, номер телефону та ІПН;

– інформацію про освіту: Тип освіти, заклад, спеціальність, період навчання, форма навчання, номер отриманого свідоцтва/диплому;

– сімейний стан: Статус, ПІБ членів родини, наявність дітей та їх дати народження;

– данні про військову службу: Категорія запасу, звання, придатність, військкомат, відмітка про зняття з реєстру;

– данні про місце роботи.

Всі працівники розподілені по відділах. Список відділів розташований окремо.

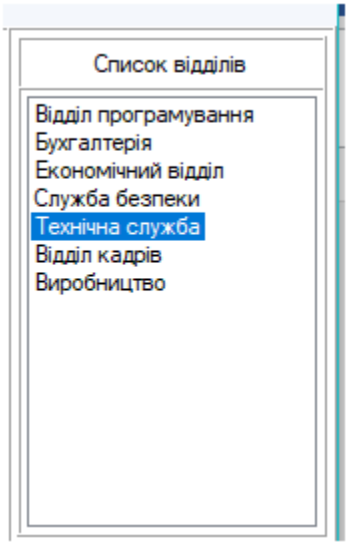


Рисунок 3.3 — Список відділів

Також програма передбачає автоматичний вивід днів народжень на даний день.

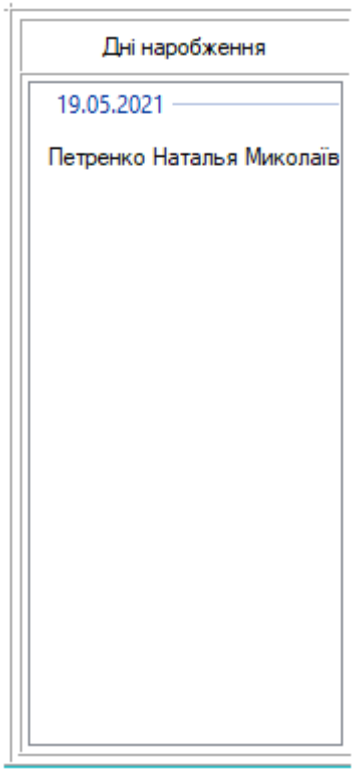


Рисунок 3.4 — Вивід іменинників

Програма також дозволяє здійснювати фільтрацію записів за сімома критеріями

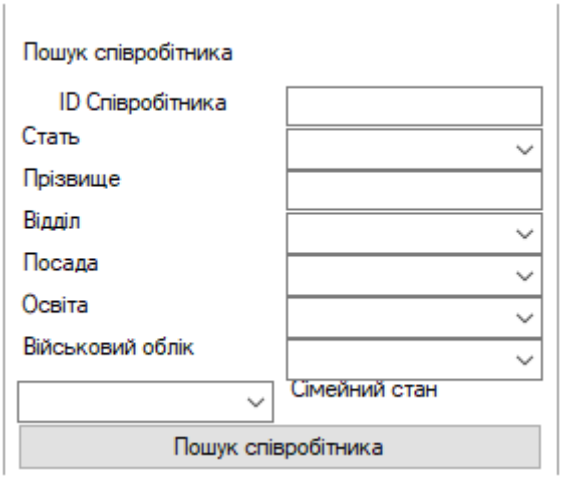


Рисунок 3.5 — Фільтр співробітників

Серед критеріїв для пошуку:

– стать;

– прізвище;

– відділ;

– посада;;

– освіта;

– військовий облік;

– сімейний стан.

Введення інформації відбувається у окремому вікні .

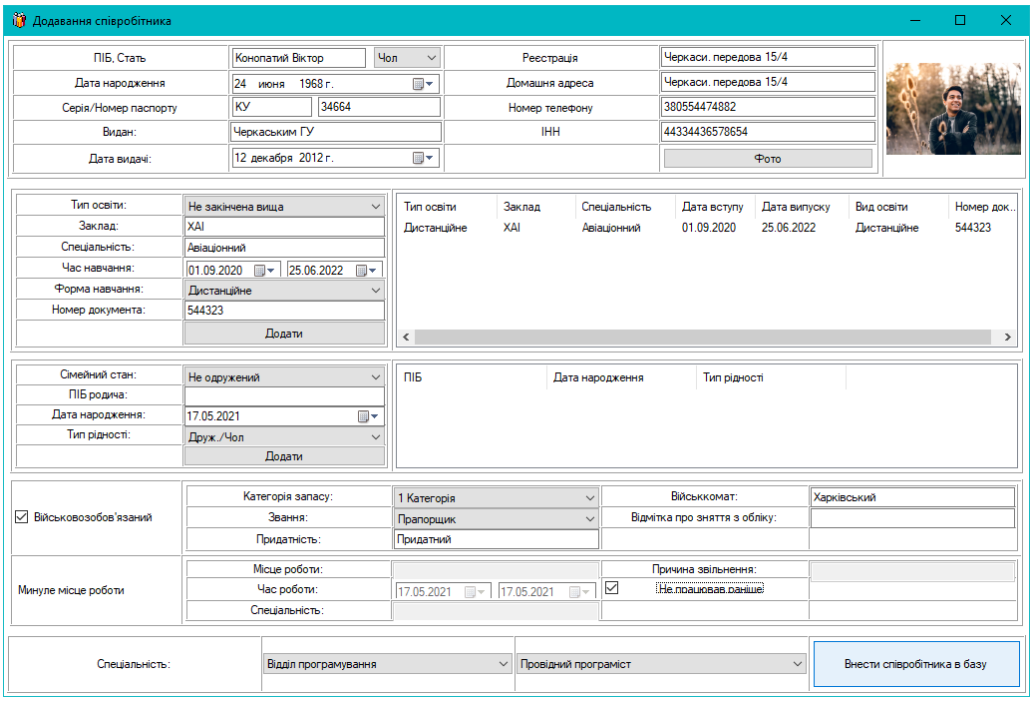


Рисунок 3.6— Введення даних працівника

Перш за все вводиться паспортні данні :

– ПІБ;

– дата народження;

– серія/номер паспорту;

– ким видано;

– дата видачі;

– місце реєстрації;

– місце фактичного проживання;

– номер телефону;

– ІПН.

Далі необхідно ввести інформацію про освіту. Ці поля також є обов’язковими:

– тип освіти;

– заклад;

– спеціальність;

– період навчання;

– форма навчання;

– номер отриманого свідоцтва/диплому.

Сімейний є необов’язковим, але дозволяє ввести:

– статус;

– ПІБ членів родини;

– наявність дітей;

– дати народження.

Для чоловіків обов’язковими є данні про військову службу:

– категорія запасу;

– звання;

– придатність;

– військкомат;

– відмітка про зняття з реєстру.

Також потрібно буде вводити данні про минуле місце роботи:

– місце;

– період;

– спеціальність;

– причини звільнення.

При прийнятті обов’язково вказується у який відділ та на яку спеціальність буде прийнята людина.

Висновки до розділу 3

Під час виконання роботи була розроблена база даних з використанням середовища Microsoft SQL Server 2019. Та мови програмування запитів Transact-SQL.

Для створення клієнтського додатку буде використовуватись середовище розробки 2.4. Microsoft Visual Studio 2019 та мова програмування C#.Код програми розміщений у додатку Б. Логічна схема БД представлена у додатку А.

4 ТЕСТУВАННЯ

4.1 Вимоги до апаратного забезпечення.

Для роботи з базою даних необхідно буде встановити SQL Server 2019 та приєднати до нього базу даних.

Основні вимоги представлені до апаратного забезпечення представлені у таблиці 4.1.

Таблиця 4.1 Апаратні вимоги

Компонент	**Вимога**
Жорсткий диск	мінімум 6 ГБ вільного місця на диску. Вимоги до місця на диску визначаються набором встановлюваних компонентів SQL Server.
Монітор	Super VGA з роздільною здатністю 1366x768 пікселів або вищою.
Інтернет	Для підтримки функціональних засобів Інтернету потрібен доступ до Інтернету
Пам'ять *	**Мінімальні:** 1 ГБ **Рекомендується:** не менше 4 ГБ з подальшим збільшенням в міру зростання розміру бази даних.
Швидкодія процесора	**Мінімум**: x64 процесор з тактовою частотою 1,4 ГГц **Рекомендується:** 2,0 ГГц і вище
Тип процесора	Процесор x64: AMD Opteron, AMD Athlon 64, Intel Xeon з підтримкою Intel EM64T, Intel Pentium IV з підтримкою EM64T.

4.2 Інструкція користувача

Програма може бути використана як АРМ у відділі кадрів невеликого підприємства. Вона дозволить автоматизувати роботу співробітників, що зменшить їх часові затрати а виконання рутинної роботи. Працювати з програмою може спеціаліст з середньої кваліфікації.

Для запуску програми необхідно:

1. Встановити SQL Server 2019.
2. Приєднати до нього розроблену базу даних.

Приєднання бази даних до SQL Server зображено на рисунках 4.1-4.3

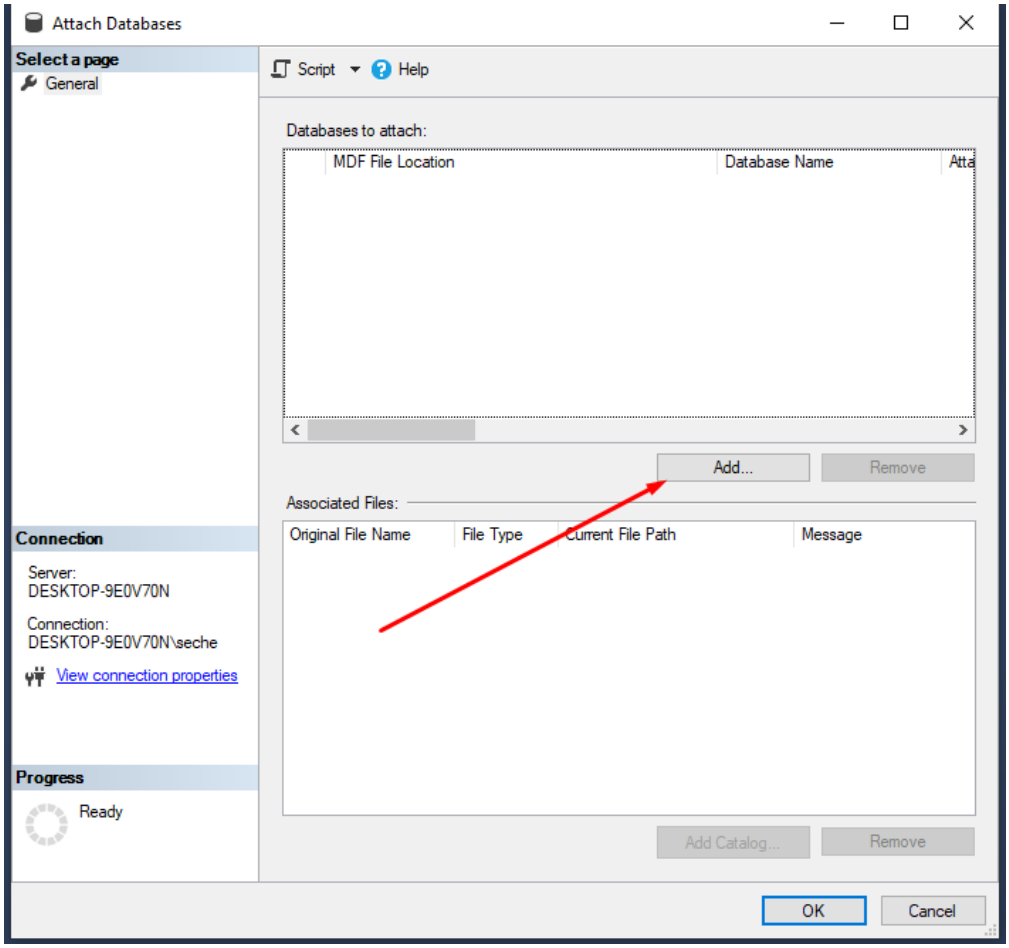


Рисунок 4.1 — Приєднання бази даних

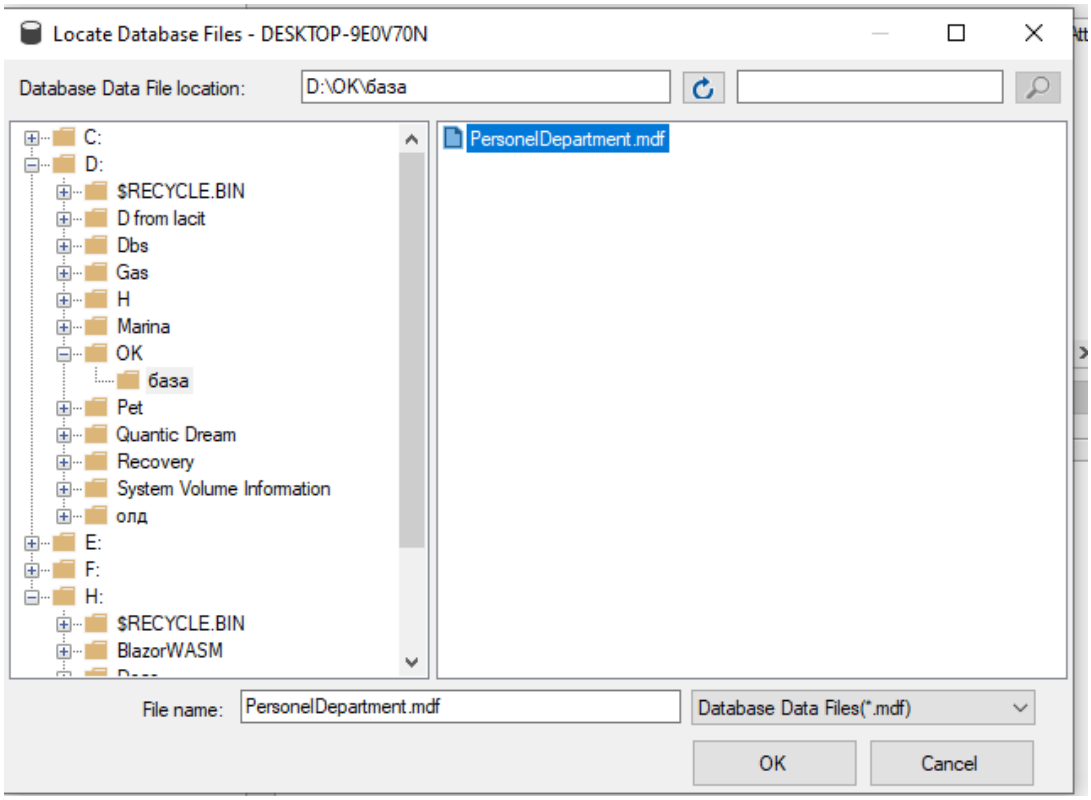


Рисунок 4.2 — Приєднання бази даних

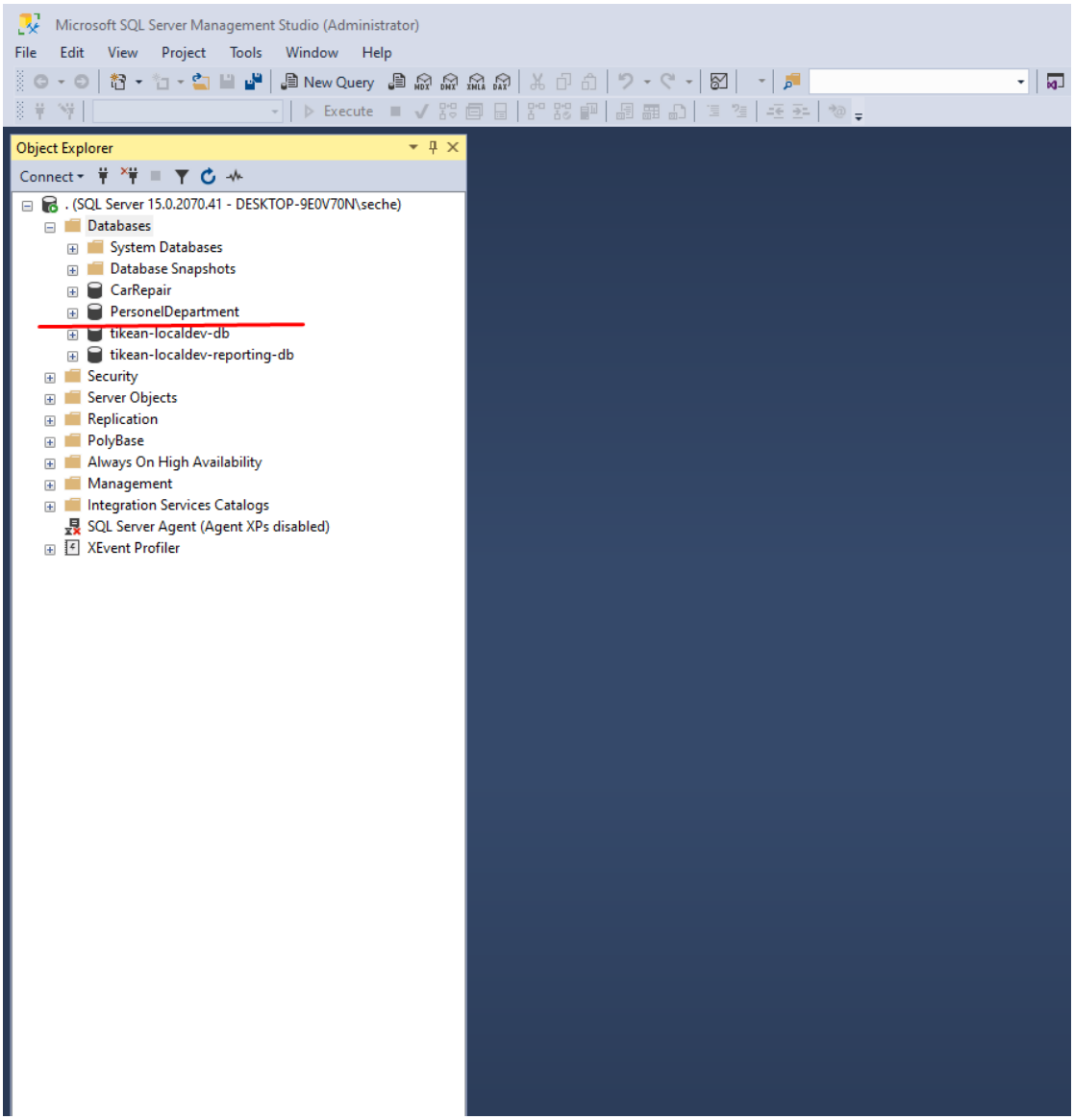


Рисунок 4.3 — Приєднання бази даних

3. запустити програму Відділ кадрів .EXE на виконання.

Для пошуку працівника необхідно ввести відомі про нього данні.

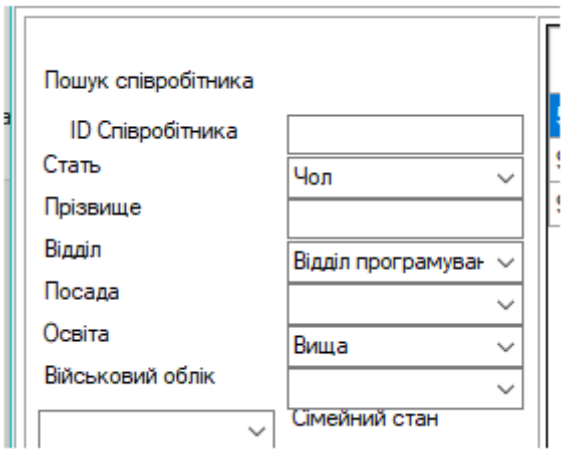


Рисунок 4.4. — Пошук працівника

Після вводу необхідно натиснути на кнопку «Пошук співробітника» Відфільтровані данні з’являться у вигляді списку на головній формі

Таб.	Співробітник	Дата народження	Посада	Відділ	Телефон
54	Віталій Сергійович...	03.03.1969 6...	Провідний програміст	Відділ програмування	
95	Петренко Петро	17.05.1996 8...	Провідний програміст	Відділ програмування	
96	Конопатий Віктор	24.06.1968 9...	Провідний програміст	Відділ програмування	
100	Петренко Наталь...	19.05.2021 1...	Провідний програміст	Відділ програмування	

Рисунок 4.5 — Список співробітників

Для більш детальної інформації про співробітника необхідно вибрати його зі списку. Дані з’являться у нижній частині екрану.

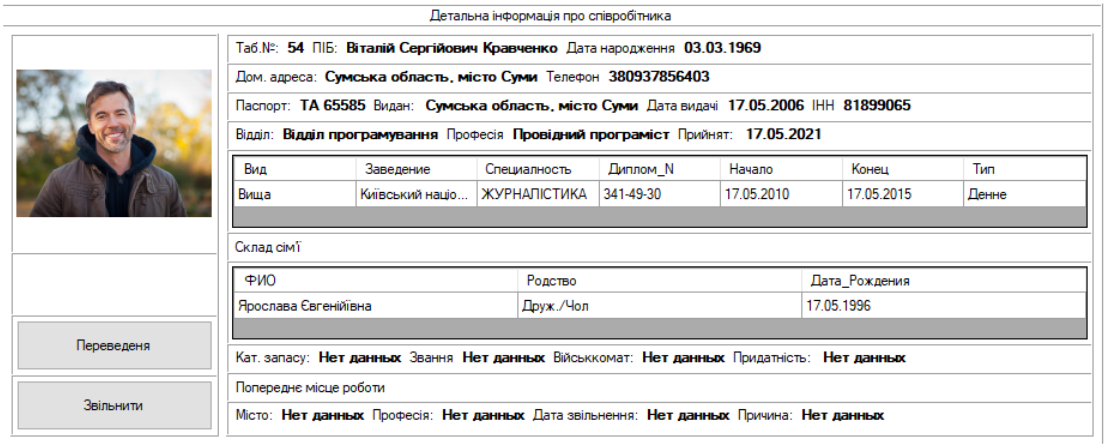


Рисунок 4.6 — Детальна інформація про співробітника

Для того щоб перевести співробітника на іншу посаду та відділ необхідно натиснути однойменну кнопку у нижній частині екрану та вибрати нове місце роботи.

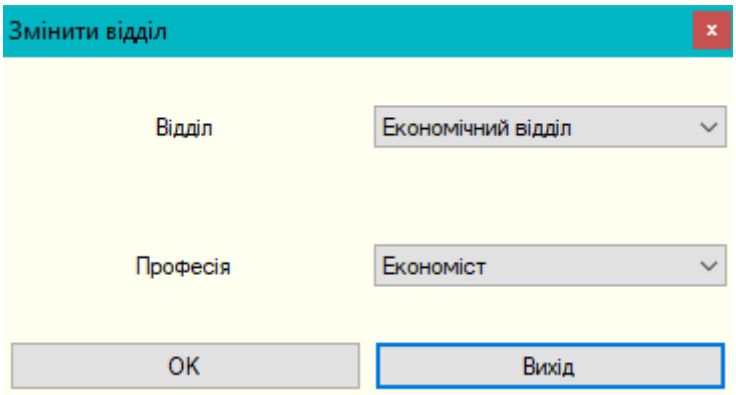


Рисунок 4.7 — Зміна місця роботи

Для звільнення необхідно скористатися відповідну кнопку на формі. Після чого інформація про співробітника буде видалена з бази даних

Для додавання нового працівника слід використовувати меню зверху на головному вікні програми. Після чого відкриється нове вікно програми, яке дозволяє вводити інформацію про співробітника.

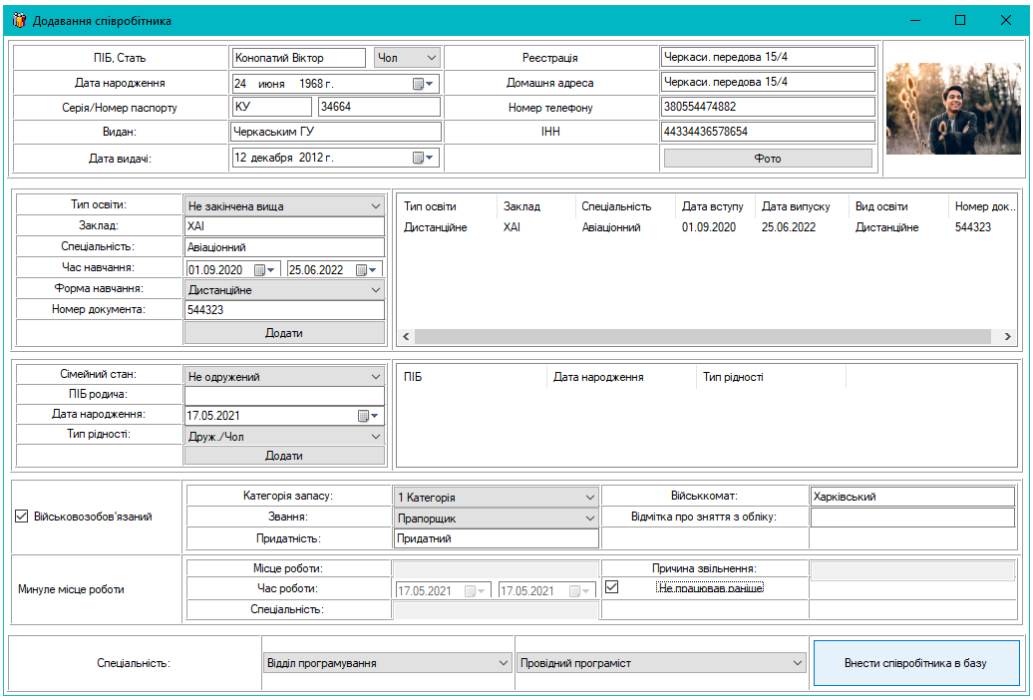


Рисунок 4.8 — Введення даних працівника

У відповідні поля логічних блоків, на які розділене вікно необхідно ввести інформацію:

– паспортні данні;

– освіта;

– родина;

– військова служба (для чоловіків);

– попереднє місце роботи при наявності;

– вибрати відділ та посаду на яку наймається працівник.

Після вводу даних необхідно натиснути на кнопку «ввести співробітника до бази.

4.3 Інструкція адміністратора.

Програма реалізована у середовищі розробки Visual Studio на мові програмування C#.

Програма призначена для керуванням інформацією з бази даних відділу кадрів. Вхідною інформацією для програми слугує особиста інформація про

співробітника. Вихідною інформацією є сформована база даних з даними співробітників підприємства.

Програма розділена на сім класів:

– BL - містить функції для роботи з базою даних;

– ChangeDepartment – зміна місця роботи співробітника;

– EducationPresenter - зберігання даних про освіту, використовується при додаванні співробітника;

– EmployeePresenter - зберігання даних про працівника;

– FamilyUnitPresent - зберігання даних про сімейну одиницю, використовується при додаванні нового співробітника;

– MainForm – головна форма;

– NewEmployeeForm – додавання нового працівника.

Діаграма класів показує взаємодію класів між собою.

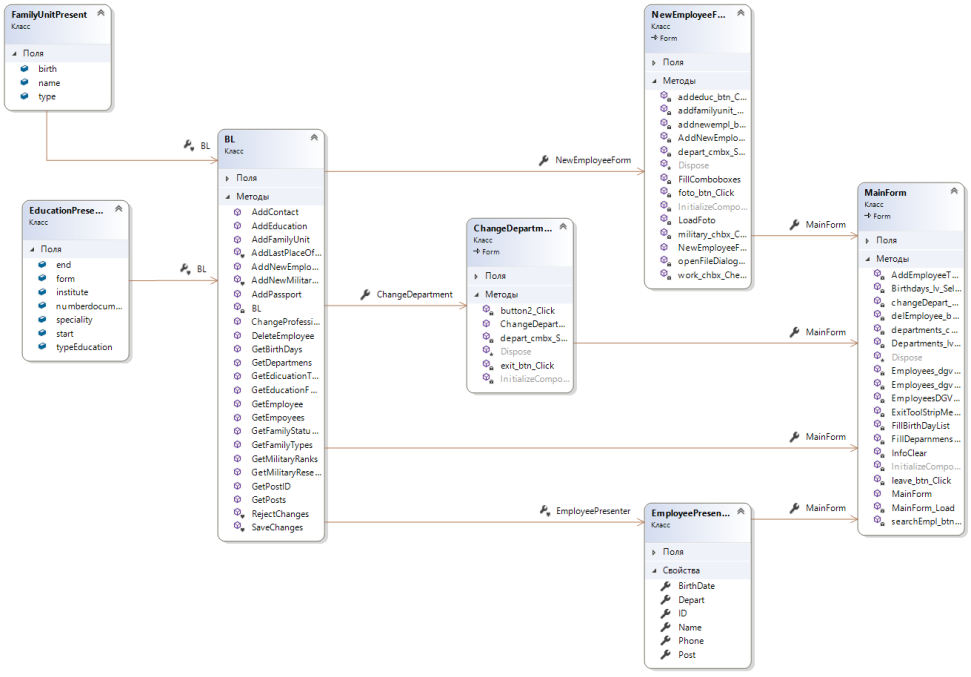


Рисунок 4.9 — Діаграма класів програми

База даних розроблена у середовищі Microsoft SQL Server 2019. База містить у собі 16 таблиць приведених до 3НФ.

Взаємодія між базою даних та програмним забезпеченням здійснюється за допомогою моделі Entity з Entity Framework.

Розроблена база даних та програмний додаток до неї може використовуватись на невеликому підприємстві у відділі кадрів для ведення обліку працівників. Також цей програмний продукт можна використовувати у навчальних цілях, для демонстрації можливостей сучасних мов програмування та потужносте СУБД.

4.4 Працездатність програмного забезпечення.

Тестування програмного забезпечення (Software Testing) – проводиться на кінцевому наборі тестів де перевіряється відповідності реальних і очікуваних результатів поведінки програми.

Мета тестування-перевірка відповідності по пропонованим вимогам, забезпечення впевненості в якості ПЗ, пошук очевидних помилок в програмному забезпеченні, які повинні бути виявлені до того, як їх виявлять користувачі програми.

Основні помилки в програмі будуть викликані при введені даних користувачами в ручному режимі. Для виключення вильоту програмного забезпечення та порушення її працездатності програма повинна обробляти виняткові випадки, помилки користувачів. У програмі повинні бути передбачені фрагменти коду, що відповідають за обробку виняткових ситуацій.

Перевірку працездатності розробленого програмного забезпечення та його тестування проводиться за методом чорного ящика. Метод чорного ящика полягає у тому, що тестер не знає вихідного коду програми, але знає кінцевий результат, який повинен отримати.

При старті програми першою помилкою, що може виникнути це відсутнє підключення до бази даних. У такому випадку програма не запускається та видає повідомлення про виникнення виняткової ситуації.

Після запуску програми відкривається головне вікно. У ньому є можливість здійснювати пошук працівників за різними параметрами. Перши параметр це пошук за табельним номером. Він має числовий формат. Помилка може виникнути тоді, коли у дане поле буде введено нечислові символи. У такому випадку програма видає помилку зображену на рисунку 4.10.

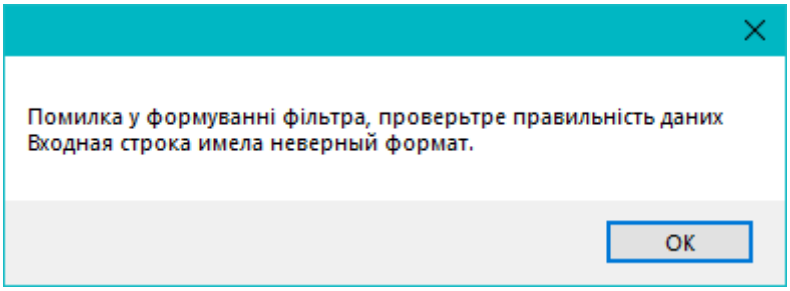


Рисунок 4.10 — Помилка фільтра

Всі інші поля, окрім Прізвища, мають формат випадаючого списку і у разі введення туди некоректних значень помилок не виникає однак фільтр не знаходить, як і повинно бути, жодного запису.

Всі інші дії у головному вікні здійснюються готовими інструментами програми де помилки виключені.

При додавання нового працівника може виникнути ситуація додавання порожнього запису у такому разі користувачеві буде видане повідомлення зображене на рисунку 4.11.

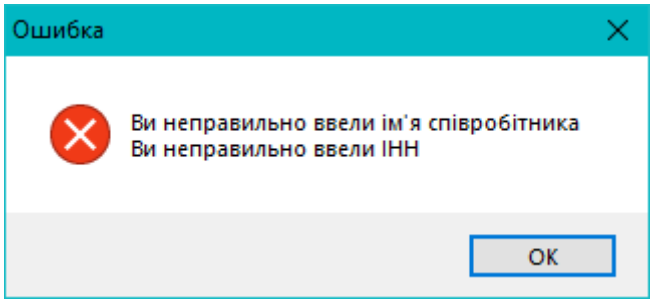


Рисунок 4.11 — Помилка додавання нового робітника

При введені лише паспортних даних виникають помилки заповнення подальшої анкети у тому числі помилка останнього місця роботи, при відсутньому прапорці перша робота.

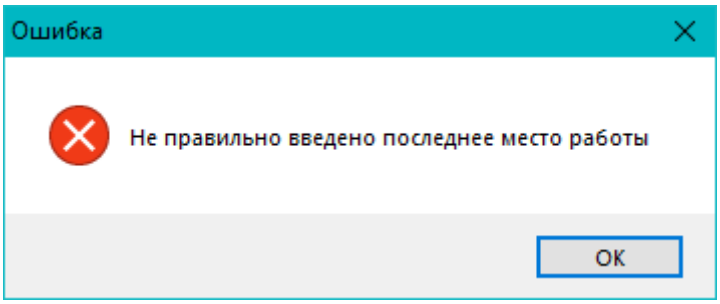


Рисунок 4.12 — Помилка введення останнього місця працевлаштування

При введені некоректних даних у поля, які необхідно заповнити також виникають повідомлення про помилки. Так само при некоректному введені даних військової служби на екрані відображаються помилки введення.

Загальний список проведених тестів зведено до таблиці 16.

Таблиця 4.1 Результати тестування програмного забезпечення

№	**Опис тесту**	**Очікувані результати**	**Отримані результати**
1	Запуск програми без запущеного екземпляру сервера	Вивід повідомлення про відсутність з’єднання	Програма не запускається. Виникає повідомлення про виняткову ситуацію.
2	Введення некоректного номеру працівника	Виведення помилки	Вивід повідомлення помилки фільтру
3	Непередбачене значення фільтру зі списку	Виникнення помилки фільтру	Відсутність знайдених записів з некоректним фільтром

4	Помилка додавання порожнього запису	Вивід помилки додавання запису	Вивід помилки додавання запису
5	Введення лише частини імені	Вивід повідомлення про недостатність даних для збереження	Вивід повідомлення про некоректність введення ім’я співробітника
6	Відсутні введені данні ІПН	Вивід повідомлення помилки	Повідомлення не правильного вводу ІПН
7	Відсутність запису ким видано паспорт	Вивід повідомлення помилки	Повідомлення неправильного вводу поля паспорт видано
8	Введення некоректного значення або взагалі відсутні дані серії та номеру паспорту	Вивід повідомлення помилки	Повідомлення про некоректне введення даних
9	Не задана дата отримання паспорту	Вивід повідомлення помилки	Повідомлення про невірно вказану дату отримання паспорту
10	Відсутність даних про придатність до військової служби	Вивід повідомлення помилки	Вивід повідомлення про невірні данні стосовно придатності до військової служби
11	Відсутність введених даних про військкомат	Вивід повідомлення помилки	Повідомлення про неправильно сказану назву військкомату.

12	Відсутність причини звільнення з попереднього місця роботи	Вивід повідомлення помилки	Повідомлення про неправильно вказану причину звільнення
13	Помилка при введені спеціалізації з попереднього місця роботи	Вивід повідомлення помилки	Повідомлення про невірно введені дані стосовно спеціальності попереднього місця роботи
14	Некоректне введення назви підприємства попереднього місця роботи	Вивід повідомлення помилки	Повідомлення про некоректність введення попереднього місця роботи
15	Введення некоректних даних при додаванні освіти працівника	Вивід повідомлення помилки	Помилка введення назви закладу або спеціальності освіти
16	Помилка при введені номеру документу про освіту	Вивід повідомлення помилки	Помилка не правильного вводу номеру документу
17	При додавання родичів відсутнє ім’я родича	Вивід повідомлення помилки	Помилка при введені імені

Програма реалізує перевірку на правильність введення кожного поля, де користувач вводить данні самостійно. При виникненні передбачуваної помилки програма видає спеціалізоване повідомлення.

При виникненні непередбачених заздалегідь помилок програма виводить помилку з текстом повідомлення згенерованим та передбаченим самою системою та можливостями середовища розробки, у якій було створено даний програмний продукт.

Висновки до розділу 4

Для роботи з базою даних необхідно буде встановити SQL Server 2019 та приєднати до нього базу даних.

Основні вимоги представлені до апаратного забезпечення представлені у таблиці 15.

Програма може бути використана як АРМ у відділі кадрів невеликого підприємства. Вона дозволить автоматизувати роботу співробітників, що зменшить їх часові затрати а виконання рутинної роботи. Працювати з програмою може спеціаліст з середньої кваліфікації.

Програма реалізована у середовищі розробки Visual Studio на мові програмування C#.

База даних розроблена у середовищі Microsoft SQL Server 2019. База містить у собі 16 таблиць приведених до 3НФ.

Взаємодія між базою даних та програмним забезпеченням здійснюється за допомогою моделі Entity з Entity Framework.

Розроблена база даних та програмний додаток до неї може використовуватись на невеликому підприємстві у відділі кадрів для ведення обліку працівників. Також цей програмний продукт можна використовувати у навчальних цілях, для демонстрації можливостей сучасних мов програмування та потужностей СУБД.

ВИСНОВКИ

Служба кадрів сьогодні оснащена сучасною оргтехнікою, спеціалізованими програмами, що забезпечують одноразове введення інформації по особовому складу та її використання всіма підрозділами.

Впровадження комп'ютерної техніки дозволяє накопичувати масиви інформації (бази даних) і документи в електронній формі по всім співробітникам організації, кадровому резерву, швидко знаходити і ефективно обробляти всю необхідну інформацію по особовому складу.

Кадрова документація-неодмінна частина документів будь-якої організації. Вона ведеться кадровою службою (відділом, департаментом по роботі з персоналом).

Використання автоматизованих технологій дозволяє істотно підвищити ефективність роботи кадрової служби. Документи по особовому складу-це специфічна документація, що вимагає спеціалізованих програм для роботи з нею. А для складання списку вимог до системи автоматизації служби кадрів, вибору найбільшою мірою, підходящої для неї системи програмного забезпечення (ПЗ) необхідні зусилля саме фахівців самої кадрової служби. Тому працівники служби кадрів повинні добре орієнтуватися в можливостях сучасних автоматизованих систем, в першу чергу - для визначення потреб свого підрозділу і складанні оптимального завдання для ІТ-служб на придбання, установку, настройку і подальше обслуговування засобів автоматизації, необхідних в роботі з документацією по особовому складу.

Системи обліку кадрів були розроблені на підставі програм розрахунку заробітної плати. Надалі функціонал цих програм значно розширився. Це було пов'язано з усвідомленням керівників підприємств необхідності якісних змін у роботі відділу кадрів.

В результаті дослідження були виконані наступні роботи:

1. Проведено аналіз існуючих методів проектування автоматизованої системи управління даними для відділу кадрів.

2. Вибрані інструменти реалізації проєктування автоматизованої системи.

3. Спроектовано автоматизовану систему.

4. Проведено тестування розробленої системи.

Провівши аналіз мов програмування баз даних те середовищ їх розробки. У даній роботі використалось середовище Microsoft SQL Server 2019. Та мова програмування запитів Transact-SQL. Через те що, Microsoft ® SQL Server ® 2019 Express-потужна і надійна безкоштовна система управління даними, що забезпечує функціональне і надійне сховище даних для веб-сайтів і настільних додатків.

Для створення клієнтського додатку використовувалось середовище розробки. Microsoft Visual Studio 2019 та мова програмування C#.

Для взаємодії програми та бази даних та спрощення програмного коду буде застосовуватись технологія Entity Framework.

Розроблена база даних та програмний додаток до неї може використовуватись на невеликому підприємстві у відділі кадрів для ведення обліку працівників. Також цей програмний продукт можна використовувати у навчальних цілях, для демонстрації можливостей сучасних мов програмування та потужносте СУБД.

СПИСОК ЛІТЕРАТУРИ.

1. Окінавська Хартія Глобального інформаційного суспільства [Електронний ресурс] - https://zakon.rada.gov.ua/laws/show/998_163#Text

2. IT-Enterprise.Кадровий облік [Електронний ресурс] – Режим доступу: https://www.it.ua/products/personal/kadrovyj-uchet

3. AnDeeSoft. Софт для відділу кадрів Кадри Плюс Україна [Електронний ресурс] – Режим доступу: https://andeesoft.com/ua/kpu/

4. КОРС-СОФТ. программы для бизнеса. Корс-Кадры [Електронний ресурс] – Режим доступу: https://www.kors-soft.net/freeware.php?id=41

5. Співробітники підприємства 2.8.1 [Електронний ресурс] – Режим доступу: https://araxgroup.ua/index.php/gallery-categories/second-gallery/28-sotrudniki-predpriyatiya

6. Управління персоналом. Персонал Бізнес. 7.0.01 [Електронний ресурс] – Режим доступу: http://personal.bravosoft.net/

7. TRANSACT-SQL [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Transact-SQL

8. Itzik Ben-Gan. Microsoft SQL Server 2012 T-SQL Fundamentals - Microsoft Press, 1st edition July 15, 2012.- 442 с.

9. What are the SQL database functions [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/en-us/sql/t-sql/functions/functions?view=sql-server-ver15

10. SQL Stored Procedures for SQL Server [Електронний ресурс] – Режим доступу: https://www.w3schools.com/sql/sql_stored_procedures.asp

11. Microsoft SQL Server [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/en-us/sql/relational-databases/sql-server-guides?view=sql-server-ver15

12. Введение в MS SQL Server и T-SQL [Електронний ресурс] – Режим доступу: https://metanit.com/sql/sqlserver/1.1.php

13. Steven Morris. Database Systems: Design, Implementation, & Management. - Cengage Learning, 13th edition January 1, 2018. 816 с.

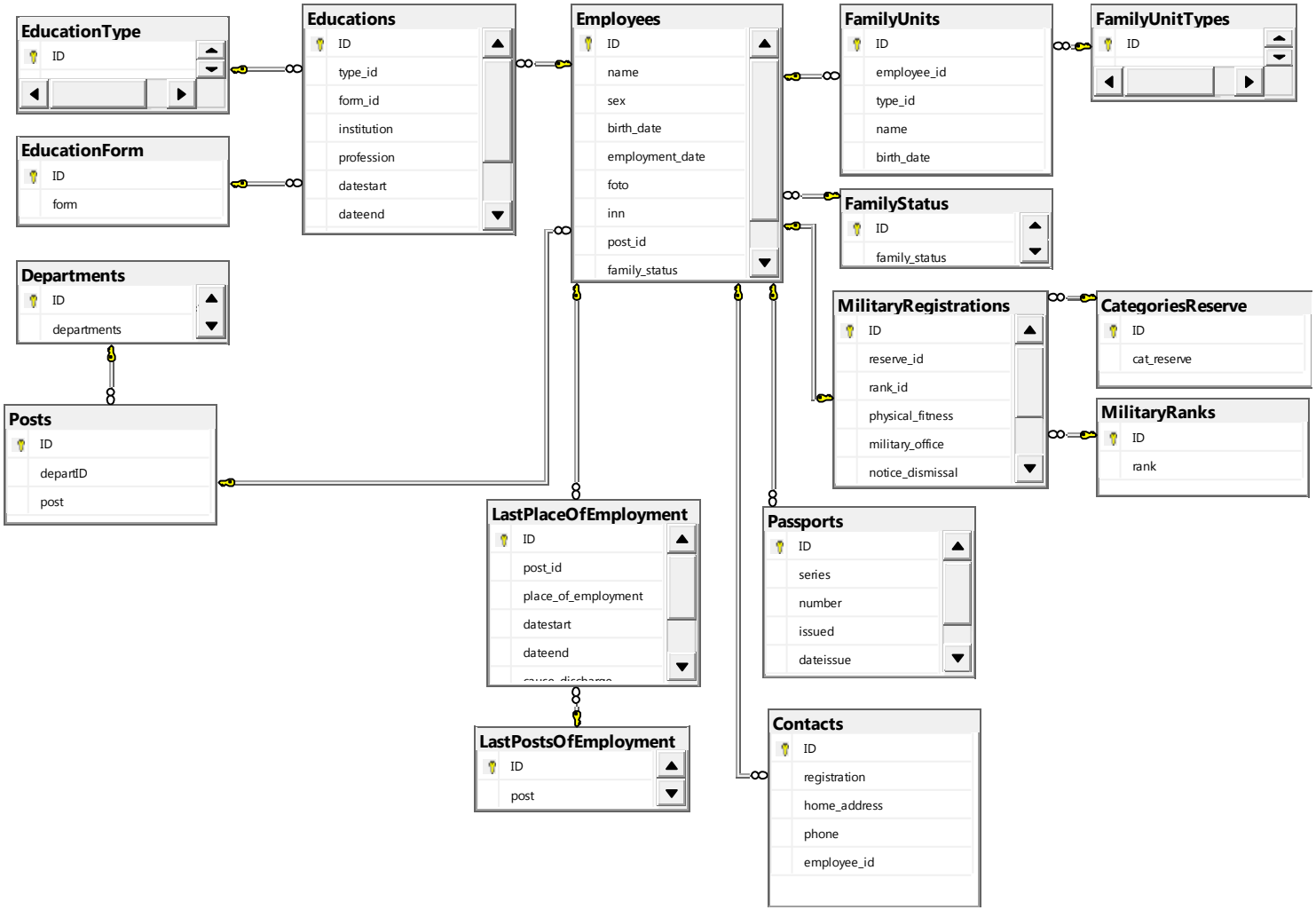
14. Мова програмування C#: [Електронний ресурс] – Режим доступу: http://www.znannya.org/?view=csharp

15. Костриба О.В. Основи програмування. Частина 1.Visual C# Express Edition / О.В. Костриба. – Білогір'я, 2008. – 74 с.

16. Visual Studio 2019 [Електронний ресурс] – Режим доступу: https://visualstudio.microsoft.com/ru/vs/

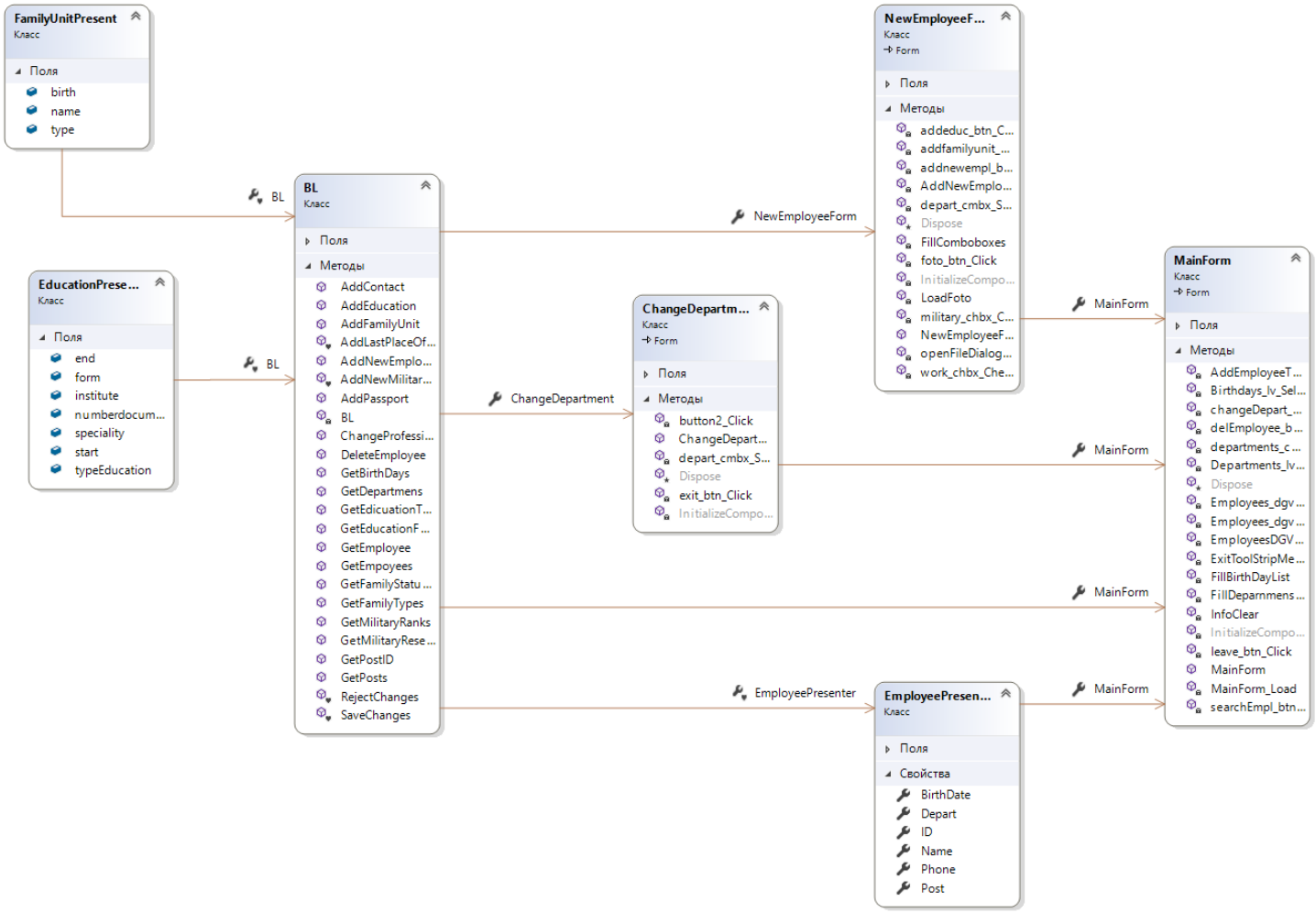
Додаток А.

Логічна схема БД



Додаток Б

Діаграма класів програми



Додаток В.

Лістинг програми

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;


namespace PersonnelDepartment
{
    /// <summary>
    /// Клас BL міститьфункції для роботи з базою даних
    /// </summary>
    class BL
    {
        static  PersonelDepartmentEntities context;

        static BL()
        {
            context = new PersonelDepartmentEntities();
        }

        /// <summary>
        ///Повертає список відділів підприємства
        /// </summary>
        public static List<Departments> GetDepartmens()
        {
            var dep = new List<Departments>();

            var query = from department in context.Departments
                        select department;
            dep = query.ToList();

            return dep;
        }

        /// <summary>
        /// Повертає іменинників на найближчі 30 днів
        /// </summary>
        /// <returns></returns>
        public static List<Employees> GetBirthDays()
        {
            var emp = new List<Employees>();


            // Отримуємо день по який покажемо іменинників, поточний + 30 діб
            var dateShow = DateTime.Now.AddDays(30);

            var query = from empl in context.Employees
                        orderby empl.name
                        select empl;

            foreach (var item in query)
            {
                var dateBirth = new DateTime(DateTime.Now.Year, item.birth_date.Month,
item.birth_date.Day);
                if ((dateShow - dateBirth).TotalDays < 31 && (dateShow - dateBirth).TotalDays
>= 0)
                {
                    emp.Add(item);
                }
            }
            return emp;
        }
```

```
        /// <summary>
        /// Повертає список співробітників у відповідності з фільтром
        /// </summary>
        /// <param name="id">Табельний номер</param>
        /// <param name="sex">Стать</param>
        /// <param name="name">Призвище</param>
        /// <param name="department">Відділ</param>
        /// <param name="post">Посада</param>
        /// <returns></returns>
        public static List<EmployeePresenter> GetEmpoyees(string id, short? sex, string name,
string department,
            string post, string familystat, string education, bool? militaryreg)
        {

            var list = new List<EmployeePresenter>();
            var query = context.Employees.AsQueryable();
            try
            {
                if (id.Length > 0)
                {
                    var _id = int.Parse(id);
                    query = query.Where(x => x.ID == _id );
                }
                if (sex != null)
                    query = query.Where(x => x.sex == sex);
                if (name.Length > 0)
                    query = query.Where(x => x.name.Contains(name));
                if (department.Length > 0)
                    query = query.Where(x => x.Posts.Departments.departments1 == department);
                if (post.Length > 0)
                    query = query.Where(x => x.Posts.post == post);
                if (education.Length > 0)
                    query = query.Where(x => x.Educations.Select(y =>y.EducationType.type ==
education).Count() > 0);
                if (familystat.Length > 0)
                    query = query.Where(x => x.FamilyStatus.family_status == familystat);
                if (militaryreg != null)
                    if (militaryreg == true)
                        query = query.Where(x => x.MilitaryRegistrations.Select(y =>
y).Count() > 0);
                    else query = query.Where(x => x.MilitaryRegistrations.Select(y =>
y).Count() == 0);

                foreach (var item in query)
                {
                    list.Add(new EmployeePresenter() { ID = item.ID, Name = item.name, Post =
item.Posts.post, BirthDate = item.birth_date, Depart = item.Posts.Departments.departments1 });
                }
            }
            catch(Exception exc)
            {
                throw new Exception("Помилка у формуванні фільтра, проверьтре правильність
даних\n" + exc.Message);
                return null;
            }

            return list;
        }

        //    /// <summary>
        //    /// Повертає всі професії людей обраного департаментуа
        //    /// </summary>
        //    /// <param name="departments"></param>
        //    /// <returns></returns>
        public static List<Posts> GetPosts(string departments)
        {
```

```
            var query = from post in context.Posts
                        where post.Departments.departments1 == departments
                        select post;
            var x = query.ToList();
            return x;
        }
        /// <summary>
        /// Возвращает сотрудника с данным ID
        /// </summary>
        /// <param name="ID"></param>
        /// <returns></returns>
        public static Employees GetEmployee(int ID)
        {
            var query = from emp in context.Employees
                        where emp.ID == ID
                        select emp;

            return query.First();
        }

        /// <summary>
        /// Отримує дані про типи освіти
        /// </summary>
        /// <returns></returns>
        public static List<EducationType> GetEdicuationTypes()
        {
            return context.EducationType.Select(x => x).ToList();
        }
        /// <summary>
        /// Повертає типи симейных положень
        /// </summary>
        /// <returns></returns>
        public static List<FamilyStatus> GetFamilyStatuses()
        {
            return context.FamilyStatus.Select(x => x).ToList();
        }

        /// <summary>
        /// видаляє співробітника з бази
        /// </summary>
        /// <param name="emp"></param>
        public static void DeleteEmployee(Employees emp)
        {
            context.Employees.DeleteObject(emp);
            context.SaveChanges();
        }

        /// <summary>
        /// Повертає ID професії
        /// </summary>
        /// <param name="post"></param>
        /// <returns></returns>
        public static int GetPostID(string post, string depart)
        {
            return context.Posts.Where(x => x.post == post && x.Departments.departments1 ==
depart).Select(x => x.ID).First();
        }

        /// <summary>
        /// Переводить співробітника на іншу посаду
        /// </summary>
        /// <param name="emp"></param>
        /// <param name="iDProf"></param>
        public static void ChangeProfession(Employees emp, int iDProf)
        {
            emp.post_id = iDProf;
            context.SaveChanges();
        }
```

```
        /// <summary>
        ///Повертає типи спорідненості
        /// </summary>
        /// <returns></returns>
        public static List<FamilyUnitTypes> GetFamilyTypes()
        {
            return context.FamilyUnitTypes.Distinct().ToList();
        }
        /// <summary>
        /// Повертає види освіти
        /// </summary>
        /// <returns></returns>
        public static List<EducationForm> GetEducationForms()
        {
            return context.EducationForm.Distinct().ToList();
        }

        /// <summary>
        /// Повертає військові звання
        /// </summary>
        /// <returns></returns>
        public static List<MilitaryRanks> GetMilitaryRanks()
        {
            return context.MilitaryRanks.ToList();
        }
        /// <summary>
        ///Повертає військові категорії запасу
        /// </summary>
        /// <returns></returns>
        public static List<CategoriesReserve> GetMilitaryReserveCateg()
        {
            return context.CategoriesReserve.ToList();
        }

        public static Employees AddNewEmployee(string name, short sex, DateTime birthdate,
string inn,
        byte[] foto, string familystatus, string department, string post)
        {
            var emp = new Employees();
            emp.name = name;
            emp.birth_date = birthdate;
            emp.inn = inn;
            emp.foto = foto;
            emp.sex = sex;
            emp.employment_date = DateTime.Now.Date;
            emp.family_status = (from status in context.FamilyStatus
                                 where status.family_status == familystatus
                                 select status.ID).First();

            emp.post_id = BL.GetPostID(post, department);

            context.Employees.AddObject(emp);
            return emp;
        }
        /// <summary>
        /// Додати пасспорт до співробітника
        /// </summary>
        /// <param name="emp"></param>
        /// <param name="series"></param>
        /// <param name="number"></param>
        /// <param name="issue"></param>
        /// <param name="date"></param>
        public static void AddPassport(Employees emp, string series, int number, string issue,
DateTime date)
        {
            Passports passp = new Passports();
            passp.dateissue = date;
            passp.issued = issue;
```

```
            passp.number = number;
            passp.series = series;
            emp.Passports.Add(passp);
        }
        /// <summary>
        /// Додати контакти до співробітника
        /// </summary>
        /// <param name="emp"></param>
        /// <param name="reg"></param>
        /// <param name="adress"></param>
        /// <param name="phone"></param>
        public static void AddContact(Employees emp, string reg, string adress, string phone)
        {
            Contacts contact = new Contacts();
            contact.registration = reg;
            contact.home_address = adress;
            contact.phone = phone;
            emp.Contacts.Add(contact);
        }
        /// <summary>
        /// додати освіту до співробітника
        /// </summary>
        /// <param name="emp"></param>
        /// <param name="educ"></param>
        public static void AddEducation(Employees emp, EducationPresenter educ)
        {
            Educations education = new Educations();
            education.form_id = context.EducationForm.Where(x => x.form == educ.form).Select(x
=> x.ID).First();
            education.type_id = context.EducationType.Where(x => x.type ==
educ.typeEducation).Select(x => x.ID).First();
            education.datestart = educ.start;
            education.dateend = educ.end;
            education.institution = educ.institute;
            education.profession = educ.speciality;
            education.number_document = educ.numberdocument;
            emp.Educations.Add(education);
        }
        /// <summary>
        /// додати сімейну одиницю до співробітника
        /// </summary>
        /// <param name="emp"></param>
        /// <param name="unit"></param>
        public static void AddFamilyUnit(Employees emp, FamilyUnitPresent unit)
        {
            FamilyUnits famunit = new FamilyUnits();
            famunit.name = unit.name;
            famunit.type_id = context.FamilyUnitTypes.Where(x => x.type == unit.type).Select(x
=> x.ID).First();
            famunit.birth_date = unit.birth;

            emp.FamilyUnits.Add(famunit);
        }
        /// <summary>
        /// Додати дані про військову повинністьи
        /// </summary>
        /// <param name="emp"></param>
        /// <param name="militaryoffice"></param>
        /// <param name="fitness"></param>
        /// <param name="notice"></param>
        /// <param name="rank"></param>
        /// <param name="reservecat"></param>
        internal static void AddNewMilitaryRegistration(Employees emp, string militaryoffice,
string fitness, string notice, string rank, string reservecat)
        {
            MilitaryRegistrations reg = new MilitaryRegistrations();
            reg.military_office = militaryoffice;
            reg.physical_fitness = fitness;
```

```
            reg.notice_dismissal = notice;

            reg.rank_id = context.MilitaryRanks.Where(x => x.rank == rank).Select(x =>
x.ID).First();
            reg.reserve_id = context.CategoriesReserve.Where(x => x.cat_reserve ==
reservecat).Select(x => x.ID).First();

            emp.MilitaryRegistrations.Add(reg);
        }
        /// <summary>
        /// Додати дані про останнє місце роботи
        /// </summary>
        /// <param name="emp"></param>
        /// <param name="place"></param>
        /// <param name="post"></param>
        /// <param name="cause"></param>
        /// <param name="start"></param>
        /// <param name="end"></param>
        internal static void AddLastPlaceOfEmployment(Employees emp,string place, string post,
string cause, DateTime start, DateTime end)
        {
            LastPlaceOfEmployment last  = new LastPlaceOfEmployment();
            last.place_of_employment = place;
            last.cause_discharge = cause;
            last.datestart = start;
            last.dateend = end;

            var poslist = context.LastPostsOfEmployment.Where(x => x.post == post).Select(x =>
x.ID).ToList();
            if (poslist.Count > 0)
                last.post_id = poslist.First();
            else {
                LastPostsOfEmployment lp = new LastPostsOfEmployment();
                lp.post = post;
                lp.ID = context.LastPostsOfEmployment.Max(x=>x.ID) + 1;
                context.AddObject("LastPostsOfEmployment", lp);
                last.post_id = lp.ID;
            }
            emp.LastPlaceOfEmployment.Add(last);
        }

        /// <summary>
        /// Зберігає всі зміни в БД
        /// </summary>
        internal static void SaveChanges()
        {

                context.SaveChanges();

        }
        /// <summary>
        /// Скасовує зміни в локальному контексті БД
        /// </summary>
        internal static void RejectChanges()
        {
            context = new PersonelDepartmentEntities();
        }
    }

}
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
```

```
namespace PersonnelDepartment
{
    public partial class ChangeDepartment : Form
    {
        Employees emp;
        public ChangeDepartment(Employees emp)
        {
            InitializeComponent();

            this.emp = emp;
            depart_cmbx.DataSource = BL.GetDepartmens().Select(x => x.departments1).ToList();
        }

        private void exit_btn_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void depart_cmbx_SelectedIndexChanged(object sender, EventArgs e)
        {
            profession_cmbx.DataSource =
BL.GetPosts(depart_cmbx.Text).Select(x=>x.post).Distinct().ToList();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            BL.ChangeProfession(emp, BL.GetPostID(profession_cmbx.Text, depart_cmbx.Text));
            this.Close();
        }
    }
}
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO;

namespace PersonnelDepartment
{
    public partial class MainForm : Form
    {
        public MainForm()
        {
            InitializeComponent();
        }

        /// <summary>
        /// Заповнює список відділів
        /// </summary>
        void FillDeparnmensList()
        {
            foreach (var item in BL.GetDepartmens())
            {
                Departments_lv.Items.Add(item.departments1);
            }
            Departments_lv.Items[0].Selected = true;
        }

        void FillBirthDayList()
        {
            var query = from emp in BL.GetBirthDays()
                        orderby emp.birth_date.Month
```

```
                        orderby emp.birth_date.Day
                        group emp by emp.birth_date.DayOfYear into newGroup
                        select newGroup;

            foreach (var group in query)
            {
                var dateGroup = new ListViewGroup(group.Key.ToString());
                Birthdays_lv.Groups.Add(dateGroup);
                foreach (var emp in group)
                {
                    var item = Birthdays_lv.Items.Add(emp.name);
                    item.Group = dateGroup;
                    dateGroup.Header = new DateTime(DateTime.Now.Year, emp.birth_date.Month,
emp.birth_date.Day).ToShortDateString();
                }
            }
        }

        private void MainForm_Load(object sender, EventArgs e)
        {
            FillDeparnmensList();
            FillBirthDayList();

            Employees_dgv.DataSource = BL.GetEmpoyees(id_txbx.Text, null, name_txbx.Text,
Departments_lv.SelectedItems[0].Text,
                post_cmbx.Text, familystat_cmbx.Text, education_cmbx.Text, null);
            ////Заповнюємо комбобоксы пошуку
            departments_cmbx.DataSource = BL.GetDepartmens().Select(x =>
x.departments1).ToList();
            departments_cmbx.SelectedItem = null;
            education_cmbx.DataSource = BL.GetEdicuationTypes().Select(x => x.type).ToList();
            education_cmbx.SelectedItem = null;
            familystat_cmbx.DataSource = BL.GetFamilyStatuses().Select(x =>
x.family_status).ToList();
            familystat_cmbx.SelectedItem = null;


        }

        // Виводить співробітників згідно фільтру
        private void searchEmpl_btn_Click(object sender, EventArgs e)
        {
            short? sex = null;
            bool? military = null;
            if (sex_txbx.Text != string.Empty)
                sex = sex_txbx.Text == "Чол" ? (short)1 : (short)0;
            if (military_cmbx.Text != string.Empty)
                military = military_cmbx.Text == "Складається" ? true : false;

            try
            {
                Employees_dgv.DataSource = BL.GetEmpoyees(id_txbx.Text, sex, name_txbx.Text,
departments_cmbx.Text, post_cmbx.Text,
                    familystat_cmbx.Text, education_cmbx.Text, military);
            }
            catch (Exception exc) { MessageBox.Show(exc.Message); }
        }

        // Змінює список співробітників при зміні відділу в списку відділів
        private void Departments_lv_SelectedIndexChanged(object sender, EventArgs e)
        {
            if (Departments_lv.SelectedItems.Count > 0)
                Employees_dgv.DataSource = BL.GetEmpoyees("", null, "",
Departments_lv.SelectedItems[0].Text, "", "", "", null);

        }

        // Заповнює комбобокс з професіями при виборі відділу
```

```
        private void departments_cmbx_SelectedIndexChanged(object sender, EventArgs e)
        {
            post_cmbx.DataSource = BL.GetPosts(departments_cmbx.Text).Select(x =>
x.post).Distinct().ToList();
            post_cmbx.SelectedItem = null;
        }

        private void ExitToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void AddEmployeeToolStripMenuItem_Click(object sender, EventArgs e)
        {
            var frm = new NewEmployeeForm();
            frm.ShowDialog();
        }

        private void Employees_dgv_RowEnter(object sender, DataGridViewCellEventArgs e)
        {
            EmployeePresenter emptemp;
            if (e.RowIndex < Employees_dgv.Rows.Count && e.RowIndex >= 0)
                emptemp = Employees_dgv.Rows[e.RowIndex].DataBoundItem as EmployeePresenter;
            else { return; }

            Employees emp;
            if (emptemp != null)
                emp = BL.GetEmployee(emptemp.ID);
            else { return; }

            _tab_lbl.Text = emp.ID.ToString();
            _name_lbl.Text = emp.name;
            _birth_lbl.Text = emp.birth_date.ToShortDateString();
            _inn_lbl.Text = emp.inn.ToString();

            try
            {
                phone_lbl.Text = emp.Contacts.First().phone;
                adress_lbl.Text = emp.Contacts.First().home_address;
            }
            catch (Exception)
            {
                phone_lbl.Text = "Немає даних";
                adress_lbl.Text = "Немає даних";
            }

            if (emp.foto != null)
                foto_pcbx.Image = Image.FromStream(new MemoryStream(emp.foto));


            var passp = emp.Passports.Where(x => x.employee_id == emp.ID).ToList();
            if (passp.Count > 0)
            {
                _passpnum_lbl.Text = passp.First().series + " " + passp.First().number;
                _passpissue_lbl.Text = passp.First().issued;
                _passpdate_lbl.Text = passp.First().dateissue.ToShortDateString();
            }
            else _passpnum_lbl.Text = _passpissue_lbl.Text = _passpdate_lbl.Text = "Нет
данных";
            _depart_lbl.Text = emp.Posts.Departments.departments1;
            _post_lbl.Text = emp.Posts.post;
            _empdat_lbl.Text = emp.employment_date.ToShortDateString();

            educ_dgv.DataSource = emp.Educations.Select(x => new
            {
                Вид = x.EducationType.type,
                Заведение = x.institution,
                Специалность = x.profession,
```

```
                Диплом_N = x.number_document,
                Начало = x.datestart.ToShortDateString(),
                Конец = x.dateend.ToShortDateString(),
                Тип = x.EducationForm.form
            }).ToArray();
            family_dgv.DataSource = emp.FamilyUnits.Select(x => new { ФИО = x.name, Родство =
x.FamilyUnitTypes.type, Дата_Рождения = x.birth_date.ToShortDateString() }).ToArray();

            var military = emp.MilitaryRegistrations;
            if (military.Count > 0)
            {
                _militofice_lbl.Text = military.First().military_office;
                _reservcat_lbl.Text = military.First().CategoriesReserve.cat_reserve;
                _rank_lbl.Text = military.First().MilitaryRanks.rank;
                _fitness_lbl.Text = military.First().physical_fitness;
            }
            else _militofice_lbl.Text = _reservcat_lbl.Text = _rank_lbl.Text =
_fitness_lbl.Text = "Нет данных";
            var lastplace = emp.LastPlaceOfEmployment;
            if (lastplace.Count > 0)
            {
                _placeemp_lbl.Text = lastplace.First().place_of_employment;
                _specempl_lbl.Text = lastplace.First().LastPostsOfEmployment.post;
                _case_lbl.Text = lastplace.First().cause_discharge;
                _dateleave_lbl.Text = lastplace.First().dateend.ToShortDateString();
            }
            else _placeemp_lbl.Text = _specempl_lbl.Text = _case_lbl.Text =
_dateleave_lbl.Text = "Нет данных";
        }

        void InfoClear()
        {
            _tab_lbl.Text = _name_lbl.Text = _birth_lbl.Text = _inn_lbl.Text =
_passpnum_lbl.Text = _passpdate_lbl.Text =
              _depart_lbl.Text = _post_lbl.Text = _empdat_lbl.Text = _militofice_lbl.Text =
_reservcat_lbl.Text =
                _rank_lbl.Text = _fitness_lbl.Text = _passpissue_lbl.Text = _placeemp_lbl.Text
= _specempl_lbl.Text = _case_lbl.Text = _dateleave_lbl.Text = string.Empty;
            foto_pcbx.Image = null;
            educ_dgv.DataSource = null;
            family_dgv.DataSource = null;
        }

        private void Employees_dgv_DataSourceChanged(object sender, EventArgs e)
        {
            EmployeesDGVSettings();
            if (Employees_dgv.Rows.Count == 0)
                InfoClear();
        }
        // Налаштовуємо зовнішній вигляд датагрида з співробітниками
        private void EmployeesDGVSettings()
        {
            if (Employees_dgv.Columns["ID"] != null)
            {
                Employees_dgv.Columns["ID"].HeaderText = "Таб.";
                Employees_dgv.Columns["ID"].Width = 35;
            }
            if (Employees_dgv.Columns["Name"] != null)
            {
                Employees_dgv.Columns["Name"].HeaderText = "Співробітник";
                Employees_dgv.Columns["Name"].Width = 110;
            }
            if (Employees_dgv.Columns["BirthDate"] != null)
            {
                Employees_dgv.Columns["BirthDate"].HeaderText = "Дата народження";
                Employees_dgv.Columns["BirthDate"].Width = 80;
            }
            if (Employees_dgv.Columns["Post"] != null)
```

```
            {
                Employees_dgv.Columns["Post"].HeaderText = "Посада";
                Employees_dgv.Columns["Post"].Width = 140;
            }
            if (Employees_dgv.Columns["Depart"] != null)
            {
                Employees_dgv.Columns["Depart"].HeaderText = "Відділ";
                Employees_dgv.Columns["Depart"].Width = 150;
            }
            if (Employees_dgv.Columns["Phone"] != null)
                Employees_dgv.Columns["Phone"].HeaderText = "Телефон";
        }

        // Показати дані про обраний іменинника
        private void Birthdays_lv_SelectedIndexChanged(object sender, EventArgs e)
        {
            if (Birthdays_lv.SelectedItems.Count > 0)
                Employees_dgv.DataSource = BL.GetEmpoyees("", null,
Birthdays_lv.SelectedItems[0].Text, "", "", "", "", null);
        }

        private void delEmployee_btn_Click(object sender, EventArgs e)
        {

        }

        private void leave_btn_Click(object sender, EventArgs e)
        {
            EmployeePresenter emp;
            if (Employees_dgv.SelectedRows.Count == 1)
            {
                emp = Employees_dgv.SelectedRows[0].DataBoundItem as EmployeePresenter;
                DialogResult dr = MessageBox.Show(string.Format("Ви дійсно хочете звільнити
працівника : {0}?", emp.Name), "Вопрос", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
                if (dr == System.Windows.Forms.DialogResult.Yes)
                {
                    var empl = BL.GetEmployee(emp.ID);
                    BL.DeleteEmployee(empl);
                    Employees_dgv.DataSource = BL.GetEmpoyees("", null, "",
Departments_lv.SelectedItems[0].Text, "", "", "", null);
                }

            }
            else MessageBox.Show("Спочатку потрібно вибрати працівника");
        }

        private void changeDepart_btn_Click(object sender, EventArgs e)
        {
            EmployeePresenter emp = Employees_dgv.SelectedRows[0].DataBoundItem as
EmployeePresenter;
            var empl = BL.GetEmployee(emp.ID);

            if (empl != null)
            {
                ChangeDepartment cd = new ChangeDepartment(empl);
                 cd.ShowDialog();
            }
            Employees_dgv.DataSource = BL.GetEmpoyees("", null, "",
Departments_lv.SelectedItems[0].Text, "", "", "", null);
        }
    }
}
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
```

```
using System.Text;
using System.Windows.Forms;
using System.IO;

namespace PersonnelDepartment
{
    public partial class NewEmployeeForm : Form
    {
        byte[] foto;
        Employees employee;
        List<EducationPresenter> educations = new List<EducationPresenter>();
        List<FamilyUnitPresent> family = new List<FamilyUnitPresent>();

        public NewEmployeeForm()
        {
            InitializeComponent();
            work_chbx.Checked = military_chbx.Checked = false;
            sex_cmbx.SelectedIndex = 0;
            openFileDialog1.FileOk += new CancelEventHandler(openFileDialog1_FileOk);
            FillComboboxes();
        }
        // Відбувається при потдверждении вибору файлу
        void openFileDialog1_FileOk(object sender, CancelEventArgs e)
        {
            LoadFoto();
        }

        // Активуємо чи ні необов'язкові поля (військова повинність і минуле місце роботи)
        private void military_chbx_CheckedChanged(object sender, EventArgs e)
        {
            militarynotuce_txbx.Enabled = militaryoffice_txbx.Enabled = rank_cmbx.Enabled =
reservecat_cmbx.Enabled = fitness_txbx.Enabled = military_chbx.Checked;
        }
        private void work_chbx_CheckedChanged(object sender, EventArgs e)
        {
            lastend_dpcr.Enabled = laststart_dpcr.Enabled = lastspecialyty_txbx.Enabled =
lastplace_txbx.Enabled = case_txbx.Enabled = !work_chbx.Checked;
        }

        // Завантажуємо фото в пам'ять
        void LoadFoto()
        {
            string name = openFileDialog1.FileName;
            FileStream fs = new FileStream(name, FileMode.Open);
            foto = new byte[fs.Length];

            for (int i = 0; i < fs.Length; i++)
                foto[i] = (byte)fs.ReadByte();

            fs.Close();
            foto_pctbx.Image = Image.FromStream(new MemoryStream(foto));
        }
        // Викликає вікно вибору файлу
        private void foto_btn_Click(object sender, EventArgs e)
        {
            openFileDialog1.Filter = "Image Files(*.BMP;*.JPG;)|*.BMP;*.JPG;";
            openFileDialog1.Title = "Выбрать фото";
            openFileDialog1.Multiselect = false;
            openFileDialog1.FileName = "Выбрать файл";
            openFileDialog1.ShowDialog();
        }

        /// <summary>
        /// Заповнює текстбокси даними з БД
        /// </summary>
        void FillComboboxes()
        {
```

```
            depart_cmbx.DataSource = BL.GetDepartmens().Select(x =>
x.departments1).Distinct().ToList();
            familystatus_cmbx.DataSource = BL.GetFamilyStatuses().Select(x =>
x.family_status).Distinct().ToList();
            familytype_cmbx.DataSource = BL.GetFamilyTypes().Select(x =>
x.type).Distinct().ToList();
            formeduc_cmbx.DataSource = BL.GetEducationForms().Select(x =>
x.form).Distinct().ToList();
            rank_cmbx.DataSource = BL.GetMilitaryRanks().Select(x => x.rank).ToList();
            reservecat_cmbx.DataSource = BL.GetMilitaryReserveCateg().Select(x =>
x.cat_reserve).ToList();
            typeeducat_cmbx.DataSource = BL.GetEdicuationTypes().Select(x => x.type).ToList();
        }

        private void depart_cmbx_SelectedIndexChanged(object sender, EventArgs e)
        {
            post_cmbx.DataSource = BL.GetPosts(depart_cmbx.Text).Select(x => x.post).ToList();
        }

        private void addnewempl_btn_Click(object sender, EventArgs e)
        {
            AddNewEmployee();
        }


        void AddNewEmployee()
        {
            string error = "";
            bool validEmployee = false;

            short sex = sex_cmbx.Text == "Чол" ? (short)1 : (short)0;

            if (name_txbx.Text.Length < 3)
                error += "Ви неправильно ввели ім'я співробітника \n";
            if (error.Length == 0)
                validEmployee = true;



            if (inn_txbx.Text.Length > 0)

                    if (validEmployee)
                        employee = BL.AddNewEmployee(name_txbx.Text, sex,
datebirth_datepicker.Value, inn_txbx.Text, foto, familystatus_cmbx.Text, depart_cmbx.Text,
post_cmbx.Text);
                    else error += "Помилка \n";
            else error += "Ви неправильно ввели ІНН \n";

            int number;
            if (employee != null)
            {
                if (seriespassp_txbx.Text.Length > 1)
                    if (int.TryParse(numberpassp_txbx.Text, out number))
                        if (datepassp_cmbx.Value != null)
                            if (passpissue_txbx.Text.Length > 3)
                                BL.AddPassport(employee, seriespassp_txbx.Text, number,
passpissue_txbx.Text, datepassp_cmbx.Value.Date);
                            else error += "Не правильно введено поле Паспорт виданий \n";
                        else error += "Не правильно введена дата отримання паспорту \n";
                    else error += "Не правильно введен номер паспорту \n";
                else error += "Не правильно введена серія пасспорту \n";

                BL.AddContact(employee, reg_txbx.Text, homeadre_txbx.Text, phone_txbx.Text);

                if (family.Count > 0 && error.Length == 0)
                    foreach (var item in family)
                        BL.AddFamilyUnit(employee, item);
                if (educations.Count > 0 && error.Length == 0)
```

```
                    foreach (var item in educations)
                        BL.AddEducation(employee, item);

                if (military_chbx.Checked)
                {
                    if (militaryoffice_txbx.Text.Length > 0)
                        if (fitness_txbx.Text.Length > 0)
                            BL.AddNewMilitaryRegistration(employee, militaryoffice_txbx.Text,
fitness_txbx.Text, militarynotuce_txbx.Text, rank_cmbx.Text, reservecat_cmbx.Text);
                        else error += "Не правильно введена призывная годность \n";
                    else error += "Не правильно введено название военкомата \n";
                }
                if (!work_chbx.Checked)
                {
                    if (lastplace_txbx.Text.Length > 0)
                        if (lastspecialyty_txbx.Text.Length > 0)
                            if (case_txbx.Text.Length > 0)
                                BL.AddLastPlaceOfEmployment(employee, lastplace_txbx.Text,
lastspecialyty_txbx.Text, case_txbx.Text, laststart_dpcr.Value.Date, lastend_dpcr.Value.Date);
                            else error += "Не правильно введена причина увольнения \n";
                        else error += "Не правильно введена специальность пред. места работы
\n";
                    else error += "Не правильно введено последнее место работы \n";
                }

                if (error.Length > 0)
                    MessageBox.Show(error, "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                else { try { BL.SaveChanges(); MessageBox.Show("Сотрудник добавлен");
this.Close(); } catch (Exception ec) { MessageBox.Show(ec.Message); } }
            }
            else MessageBox.Show(error, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }

        private void addeduc_btn_Click(object sender, EventArgs e)
        {
            string error = "";
            int number;
            if (institute_txbx.Text.Length < 2)
                error += "Вы ввели не правильно название заведения \n";
            if (speciality_txbx.Text.Length < 1)
                error += @"Вы ввели не правильно название специальности, если ее нет введите
'Нет данных' ";
            if (educnumber_txbx.Text.Length > 0)
                if (error.Length == 0)
                    educations.Add(new EducationPresenter()
                    {
                        form = formeduc_cmbx.Text,
                        typeEducation = typeeducat_cmbx.Text,
                        institute = institute_txbx.Text,
                        start = datestarteduc_dpcr.Value.Date,
                        end = dateendeduc_dpcr.Value.Date,
                        speciality = speciality_txbx.Text,
                        numberdocument = educnumber_txbx.Text
                    });
                else error += "\nВы ввели не правильно номер документа  ";
            if (error.Length > 0)
                MessageBox.Show(error, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            else
            {
                var item = educ_lv.Items.Add(new ListViewItem(formeduc_cmbx.Text));
                item.SubItems.Add(institute_txbx.Text);
                item.SubItems.Add(speciality_txbx.Text);
                item.SubItems.Add(datestarteduc_dpcr.Value.ToShortDateString());
                item.SubItems.Add(dateendeduc_dpcr.Value.ToShortDateString());
                item.SubItems.Add(formeduc_cmbx.Text);
                item.SubItems.Add(educnumber_txbx.Text);
            }
```

```
        }
        private void addfamilyunit_btn_Click(object sender, EventArgs e)
        {
            string error = "";
            if (familyname_txbx.Text.Length < 3)
                error += "Вы не правильно ввели имя родственника \n";

            if (error.Length == 0)
            {
                family.Add(new FamilyUnitPresent() { name = familyname_txbx.Text, birth =
familydate_dpcr.Value.Date, type = familytype_cmbx.Text });
                var item = family_lv.Items.Add(familyname_txbx.Text);
                item.SubItems.Add(familydate_dpcr.Value.ToShortDateString());
                item.SubItems.Add(familytype_cmbx.Text);
            }
            else MessageBox.Show(error, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }


    }
}
```