Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБЛЕННЯ ЗАСТОСУНКУ ДЛЯ ПЛАНУВАННЯ ОСОБИСТОГО ЧАСУ
(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-18-1

Тарабукін М. М.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Руденко Д.О.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2022 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Тарабукіну Микиті Максимовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення застосунку для планування особистого часу

затверджена наказом університету від 16 травня 2022 року № 541Ст

2. Термін подання студентом роботи до екзаменаційної комісії 28 травня 2022 р.

3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, CASE-засіб візуального проєктування даних ERwin Data Modeler, система управління реляційними базами даних Microsoft SQL Server, інтегроване середовище розробки програмного забезпечення Microsoft Visual Studio.

4. Перелік питань, що потрібно опрацювати в роботі
1. Огляд основних методів.
2. Проєктування бази даних.
3. Практична реалізація програмного застосунку.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність проблеми планування особистого часу, мета роботи, постановка задачі, етапи виконання роботи, результати роботи, схеми бази даних, форми застосунку, код застосунку.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Консультант з дотримання діючих стандартів та норм	Доцент Бєлова Н.В.		

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	18.04.2022	
2	Аналіз завдання, підбір літератури	18.04.22-21.04.22	
3	Аналіз літератури з досліджуваної проблеми	22.04.22-25.04.22	
4	Аналіз технічних засобів	26.04.22-30.04.22	
5	Проєктування бази даних	01.05.22-14.05.22	
6	Програмна реалізація	15.05.22-23.05.22	
7	Оформлення пояснювальної записки	24.05.22-28.05.22	
8	Перевірка на плагіат	29.05.22	
9	Рецензування	30.05.22	
10	Підготовка презентації та доповіді	31.05.22-05.06.22	
11	Занесення роботи в електронний архів	06.06.22	
12	Попередній захист кваліфікаційної роботи	06.06.22	

Дата видачі завдання 18 квітня 2022 р.

Студент _____

(підпис)

Керівник роботи _____ доц. Руденко Д.О.

(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 44 с., 16 рис., 32 джерела.

ІНФОРМАЦІЙНА СИСТЕМА, СУБД, РЕЛЯЦІЙНА БАЗА ДАНИХ, СУТНІСТЬ, НОРМАЛІЗАЦІЯ ДАНИХ, ER-МОДЕЛЬ, МОДЕЛЬ «СУТНІСТЬ – ЗВ'ЯЗОК», МОВА ЗАПИТІВ SQL, CASE-ЗАСІБ ERWIN, MSSQLSERVER.

Об'єктом роботи є процес розробки інформаційної системи.

Метою роботи є розроблення застосунку для планування особистого часу.

Під час розробки інформаційної системи були використані: теорія проєктування реляційних баз даних на базі будування ER-моделі; основні принципи нормалізації реляційних баз даних; мова запитів SQL; CASE-засіб візуального проєктування даних ERwin, середовище розробки програмних систем MSVisualStudio, мова програмування C#.

У результаті роботи здійснена програмна реалізація застосунку для планування особистого часу.

INFORMATION SYSTEM, DBMS, RELATIVE DATABASE, ENTITY, NORMALIZATION OF DATA, ER-MODEL, «ENTITY – RELATIONSHIP» MODEL, SQL QUERY LANGUAGE, CASE-TOOL ERWIN, MSSQLSERVER.

The object of work is the process of developing an information system.

The aim of the work is to develop an application for personal time planning.

During the development of the information system were used: the theory of relational database design based on the construction of ER-model; basic principles of normalization of relational databases; SQL query language; CASE-tool for visual data design ERwin, MSVisualStudio software development environment, C# programming language.

As a result, the software implementation of the application for personal time planning was implemented.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ПЗ – програмне забезпечення

БД – база даних

СУБД – система управління базами даних

SQL – Structured Query Language (мова структурованих запитів)

ВСТУП

Застосунок для планування особистого часу – програма, що призначена для ведення обліку бази намічуваних заходів (дата, час і тривалість, місце проведення). За допомогою цієї програми людина може дивитися справи за датою, додавати та видаляти їх.

Ця програма виконує такі функції, як перегляд, додавання записів, їх видалення, сортування записів.

Мета роботи зумовлює розв’язання низки завдань, які в результаті зручно впорядкують великі об’єми інформації в одну базу даних, зручний доступ до якої дозволить заощадити час, бо з програмою зможе працювати людина без навичок адміністрування та розробки баз даних.

Актуальність розробки даної програми полягає в тому, що люди потребують зручного впорядкування і обробки даних для підвищення ефективності своєї діяльності.

1 ОГЛЯД ОСНОВНИХ МЕТОДІВ

1.1 Органайзери

Органайзер – діловий (зазвичай) блокнот, який використовується для регулярного запису справ, планів, результатів та інших записів, пов’язаних із щоденною діловою діяльністю особи чи компанії/організації. Найпопулярніший вид – щоденник у вигляді книги, з різними палітурками.

Перша згадка про щоденник відноситься до Італії 1650 року. Існування щоденника з кінця XVII століття до сьогодення свідчить про важливість цієї особи, а також означає, що в щоденнику є про що писати. Це можуть бути боргові зобов’язання, відносини, податкова інформація та розпорядження керівника. Таким чином, сам блокнот виготовляється з усіма регаліями та гербами, властивими прізвищу – за спеціальним замовленням майстра палітурної справи. Здебільшого він вплетений в шкіру з великою позолотою, а в первісному вигляді містить персональні дані власника, вказуючи, кому він належить, куди потрібно повернути щоденник у разі втрати [1–3].

У наші дні щоденник і традиція його використання набув широкого поширення, а сам принцип ведення списку справ та планування часу переріс у напрями управління часом (тайм менеджмент, проєктний менеджмент) [4].

Відомо, що ділові люди та ті, хто намагається зробити якомога більше, дуже ретельно планують своє повсякденне життя. У цьому випадку відмінним помічником стане спеціальний блокнот, виготовлений у вигляді щоденника. Ведення щоденника допомагає оптимізувати ваш день, щоб отримати від нього максимум користі. Особливою популярністю користуються так званий планінг – щоденник, створений для максимальної зручності планування дня. У ньому є багато граф, які допомагають вести структуровані записи у зрозумілому форматі.

Щоденники можна умовно розподілити на дві категорії: датовані та недатовані. Датований оптимально підходить для людей, які звикли вести свої записи щодня. Вони оформлені за календарним принципом, починаючи з першого січня, закінчуючи останнім днем грудня. Покупці, які отримують такі щоденники в середині року, можуть відкласти їх на наступний, або починають вести його з певної сторінки. Недатований щоденник не прив'язаний до дати, тим самим є зручнішим для деяких користувачів. Так, втрачається потреба робити записи у вихідні.

За стилем оформлення щоденники бувають найрізноманітнішими. Наприклад, можна вибрати оригінальні моделі з яскравими та незвичайними обкладинками. Діловим та успішним людям бізнес-етикет диктує правило придбання стриманої друкованої продукції у класичному стилі. Окремою категорією варто виділити щоденники для дітей. Зокрема, ними люблять користуватися маленькі діти та підлітки. Вони виготовляються у яскравому оформленні з улюбленими мультяшними героями чи персонажами з кінофільмів.

За принципом кріплення щоденники розподіляються на пошиті і ті, які кріпляться на кільцях. Більш якісними вважаються саме перші. Вони виготовляються із щільних матеріалів, не бояться надмірно інтенсивної експлуатації. Однак і моделі на кільцях мають деякі вагомі переваги. Наприклад, з них легко можна вирвати зіпсовану сторінку.

В даний час щоденники набувають найрізноманітніших форм виконання, які залежать від можливостей видавця. Зустрічаються навіть щоденники-розмальовки. І значення італійських виробників як законодавців моди в цьому питанні зникло.

Хоча існує багато різновидів, найпопулярнішим є щоденник у вигляді книги в твердій палітурці, виготовленої з м'якої шкіри, яка зазвичай дає власнику додаткову інформацію для комфортної роботи (телефонні коди, таблиці відстаней між обласними центрами, конвертація одиниць виміру, державні та міжнародні свята, тощо), а також (тільки для датованих

щоденників) річні, місячні, тижневі та щоденні розклади. Що, втім, не заважає існуванню недатованих щоденників, дати та події у яких проставляє сам власник.

Найбільш традиційні формати щоденників: А5, А4. У цьому плані кожен визначає собі найзручніші розміри:

– щоденник А4 – 21см*29см – настільний варіант, носити щоденник на зустрічі та ділові переговори некомфортно;

– щоденник А5 – 15см*21см – найпопулярніший формат, дозволяє легко носити щоденник у портфелі чи діловій сумці.

Крім розмірів, виробництво передбачає вибір їхнього внутрішнього наповнення. Папір для них найчастіше беруть щільністю 70 гр/м^2 – 80 гр/м^2, колірна гама найчастіше біла чи кремова.

З розвитком бізнес-середовища в різних країнах, щоденник з персонального продукту, що продається в роздрібних мережах, перетворюється на діловий сувенір, який зазвичай дарують найкращим клієнтам або партнерам від імені компанії. Для цього рекламні агенції освоїли різні методи брендування та ексклюзивного виконання щоденників, з метою підкреслити імідж компанії, яка здійснює такий подарунок, а також цінність подарунка, та шанобливе ставлення до особи, якій такий подарунок було зроблено [1, 4–7].

У наші дні щоденники використовуються все рідше та витісняються іншими більш універсальними засобами для планування особистого та робочого часу, починаючи від схожих по використанню щоденників-планерів, до мобільних додатків, що включають планувальники завдань і доручень [2].

1.2 Програмне забезпечення

Програмне забезпечення – програма або багато програм, що використовуються для керування комп’ютером.

Комп’ютерна програма – це комбінація комп’ютерних інструкцій та даних, що дозволяє апаратному забезпеченню комп’ютерної системи виконувати обчислювальні або керуючі функції. Або синтаксична одиниця, яка відповідає правилам певної мови програмування, що складається з визначень та операторів або інструкцій, необхідних для певної функції, завдання чи вирішення проблеми [8, 9].

1.2.1 Рання історія. Корпоративне ПЗ

Сам термін «програмне забезпечення» широко використовується з початку 1960-х років, коли стало актуальним розмежування між командами, які керують комп’ютером, і його фізичними компонентами – апаратним забезпеченням. У той же час індустрія програмного забезпечення почала формуватися як самостійна галузь. Першу компанію з розробки програмного забезпечення було засновано в 1959 році Роєм Наттом і Флетчером Джонсом Computer Science зі 100 доларів США стартового капіталу. Першими замовниками CSC і наступних компаній-розробників програмного забезпечення були великі корпорації та державні організації, такі як NASA, які продовжували працювати на ринку спеціального програмного забезпечення, як і інші ранні приватні стартапи програмного забезпечення, такі як Computer Use Corporation (CUC).

Першими самостійно випущеними програмними продуктами, що не поставляються в комплекті з комп’ютерним обладнанням, були випущений фірмою Applied Data Research в 1965 році генератор комп’ютерної документації AUTOFLOW, який автоматично малював блок-схеми, і транслятор мови програмування MARK-IV, розроблений в 1960-х роках в Informatics, Inc. Становлення ринку корпоративного програмного забезпечення був із появою сімейства комп’ютерів IBM System/360. Досить масові, відносно недорогі обчислювальні машини, сумісні один з одним на

рівні програмного коду, відкрили дорогу програмному забезпеченню, що тиражується.

Поступове розширення замовників програмного забезпечення стимулювало розробку нових типів програмного забезпечення. Так виникли перші компанії, що спеціалізуються на розробці систем автоматизованого проєктування.

Типові програмні продукти того часу служили для автоматизації спільних для бізнесу завдань, таких як нарахування заробітної плати або автоматизації бізнес-процесів таких підприємств середнього бізнесу, як виробниче підприємство або комерційний банк. Вартість такого ПЗ, як правило, була між п’ятьма та ста тисячами доларів [9–14].

1.2.2 Персональні комп’ютери та програмне забезпечення для масового споживача

Поява у 1970-х роках перших персональних комп’ютерів створила передумови для зародження масового ринку програмного забезпечення (ПЗ). Спочатку програми для персональних комп’ютерів поширювалися в «коробковій» формі через торгові центри або поштою та мали ціну 100 – 500 доларів США.

Важливими для масового ринку програмного забезпечення є такі продукти, як електронна таблиця VisiCalc, яку Деніел Бріклін придумав, коли був випускником MIT і інженером-програмістом в DEC, відвідував курси в Гарвардській школі бізнесу та хотів полегшити свої виснажливі фінансові розрахунки, а також текстовий процесор WordStar, розробку якого розпочав Сеймур Рубінштейн, ретельно вивчивши потреби ринку. Про VisiCalc вперше заговорили, як про killer application, тобто комп’ютерний додаток, який самим фактом свого існування доводить потрібність (і часто необхідність купівлі) платформи, для якої реалізована така програма. Для VisiCalc та WordStar

такою платформою стали персональні комп’ютери, які завдяки ним із багатої іграшки для гіків стали робочим інструментом. З них почалася мікрокомп’ютерна революція, а у цих програм з’явилися конкуренти: електронні таблиці SuperCalc, Lotus 1-2-3, система управління базами даних dBase II, текстовий процесор WordPerfect та ін. Текстові процесори, електронні таблиці, системи управління базами даних, а також графічні редактори стали основними продуктами ринку програмного забезпечення для персональних комп’ютерів.

Масове тиражування дозволило знизити до середини 1990 років вартість програмного забезпечення для персональних комп’ютерів до ста – п’ятисот доларів, при цьому бізнес виробників ПЗ придбав певну схожість з бізнесом звукозаписних компаній [9, 13].

1.2.3 Класифікація ПЗ

Підходи до класифікації програмного забезпечення досить докладно формалізовані в міжнародному стандарті ISO/IEC 12182. Зокрема, перша версія стандарту передбачала 16 критеріїв класифікації програмних засобів: за режимом експлуатації, за масштабом, за стабільністю, за функцією, за вимогою захисту, за вимогою надійності, за необхідними робочими характеристиками, за вихідною мовою, по прикладній області, за обчислювальною системою та середовищем, за класом користувача, за вимогою до обчислювальних ресурсів, за критичністю, готовністю, по представленню даних, використання програмних даних.

Прикладами класів функції ПЗ є наступні: обробка ділових повідомлень, компіляція, наукові обчислення, обробка текстів, медичні системи, системи керування.

Прикладами класів прикладної галузі є наука, побутові пристрої, обладнання, апаратура, управління процесом, підприємництво, система організації мережі.

Прикладами класів масштабу ПЗ є малий, середній, великий.

Прикладами класів критичності є національна безпека, людське життя, соціальний хаос чи паніка, організаційна безпека, приватна власність, таємність.

Прикладами класів користувача є початківець, середній, спеціаліст (експерт), звичайний, випадковий, інша система програмного забезпечення, механічні засоби.

Прикладами класів стабільності є постійне внесення змін, дискретне внесення змін, малоймовірне внесення змін.

За ступенем переносимості програми поділяють на:

– платформозалежні;

– кросплатформні.

За способом поширення та використання програми ділять на пропрієтарні, відкриті, вільні.

За призначенням програми ділять на системні, прикладні.

За видами програми поділяють на:

– компонент – програма, що розглядається як єдине ціле, що виконує закінчену функцію і застосовується самостійно або у складі комплексу;

– комплекс – програма, що складається з двох або більше компонентів та (або) комплексів, що виконують взаємопов’язані функції, та застосовується самостійно або у складі іншого комплексу.

Класифікація програмного забезпечення сектору промисловості включає кілька підходів. У цілому нині програмне забезпечення ділять на замовне, тобто створюване для конкретного замовника, і продуктове, тобто створюване для продажу на ринку. У свою чергу, за типами споживача ПЗ поділяють на Business-to-Business (B2B), тобто для підприємств та організацій, та Business-to-Consumer (B2C), тобто для приватних осіб [14].

Одним із варіантів класифікації по сектору індустрії є розподіл на програмне забезпечення для корпоративного замовника (англ. enterprise software vendors), програмне забезпечення для масового споживача (англ. mass-market software vendors) та IT-сервіси.

Інший підхід полягає у розподілі індустрії ПЗ на три сектори: бізнес-продукти загального призначення (англ. Business Function Software), спеціалізовані бізнес-продукти (англ. Industrial Business Software) та продукти для приватного життя (англ. Consumer Software). Бізнес-продукти загального призначення призначені для підтримки функціонування підприємств та організацій та включають бухгалтерські системи, фінансові системи, системи обліку персоналу тощо. Спеціалізовані бізнес-продукти орієнтовані на завдання конкретного типу бізнесу: геоінформаційні системи, медичні системи, логістичні системи тощо. п. Продукти для приватного життя включають антивірусне ПЗ та системи для інформаційної безпеки, різні корисні утиліти, освітнє ПЗ, мультимедійне ПЗ тощо.

Системне програмне забезпечення – набір програм, які забезпечують управління компонентами комп’ютерної системи, такими як процесор, оперативна пам’ять, пристрої вводу-виводу, мережеве обладнання, виступаючи як «міжшаровий інтерфейс», між апаратним забезпеченням, з одного боку, і програмами користувача з іншого. На відміну від прикладного програмного забезпечення, системне не вирішує конкретних практичних завдань, а лише забезпечує роботу інших програм, надає сервісні функції для них, абстрагує деталі апаратної та мікропрограмної реалізації комп’ютерної системи та керує апаратними ресурсами комп’ютерної системи. Віднесення того чи іншого програмного забезпечення до системного умовно і залежить від угод, що використовуються в конкретному контексті. Як правило, до системного програмного забезпечення відносяться операційні системи, утиліти, системи управління базами даних, широкий клас програмного забезпечення [10, 15, 16].

Прикладна програма або програма – програма, призначена для виконання певних завдань і розрахована на безпосередню взаємодію з користувачем. У більшості операційних систем прикладні програми не можуть звертатися до ресурсів комп'ютера безпосередньо, а взаємодіють з апаратним забезпеченням та іншими програмами за допомогою операційної системи.

До прикладного програмного забезпечення належать комп'ютерні програми, написані для користувачів або користувачами для завдання комп'ютера конкретної роботи. Програми обробки замовлень або створення списків розсилки – приклад прикладного програмного забезпечення [14, 17].

1.3 Існуючі застосунки для планування особистого часу

В Any.do до кожного завдання можна додавати теги, нагадування, підзавдання та вкладення. Функція пріоритетів допоможе сконцентруватися на головному та не забути про важливу справу, а списками можна ділитися з друзями та колегами – та планувати спільні проєкти. У платній версії можна створювати завдання, що повторюються, прикріплювати файли будь-якого розміру, використовувати індивідуальні теми, а також створювати нагадування, прив'язані до певного місця.

Todoist – головний конкурент та відмінна альтернатива Any.do. У додатку немає хіба що дерева, що самознищується. У Todoist можна розбивати завдання щодо проєктів та підпроєктів, додавати теги, змінювати тему, додавати коментарі, прикріплювати файли та вести спільні проєкти. Як додаткова мотивація в сервіс вбудований трекінг продуктивності. Деякі функції, включаючи нагадування та додавання завдань по e-mail, доступні лише у преміум-версії програми.

Wunderlist – ще один застосунок для планування особистого часу. Незважаючи на те, що Microsoft, який купив Wunderlist, заявив про закриття програми, сервіс досі працює. Сам Wunderlist залишається досить зручною

програмою: до кожного завдання можна додати нагадування, повтори, підзадачі, нотатки та файли, створювати кілька різних списків та налаштовувати їх під себе. Wunderlist може похвалитися ще однією особливістю – функцією «Бесіди», завдяки якій можна обговорювати спільні завдання, не виходячи із програми.

Викупивши популярний Wunderlist, Microsoft не стали винаходити колесо і створили додаток Microsoft To-Do, що практично повністю повторює можливості Wunderlist. У Microsoft To-Do додали кілька нових функцій: індивідуальний планувальник завдань, списки з позначеннями кольорів і пропозиції, які дозволяють ефективніше планувати час.

Google Календар – один з найпростіших і найнадійніших планувальників, що синхронізується з усіма розглянутими програмами. Якщо раніше в календарі можна було лише розсортувати завдання по днях та часу, то зараз додалося функцій. Наприклад, у Google Календарі з'явилася можливість ставити цілі: задаєте ціль, і програма сама пропонує вам час для її досягнення.

GTasks – ще один дуже простий сервіс для шанувальників мінімалізму. Програма дозволяє створювати завдання, призначати їм час, ділитися ними з колегами та друзями. У безкоштовному додатку є реклама, яка може неабияк дратувати. Повний аналог програми без реклами – TickTick.

Trello – дуже зручна для роботи над проєктами в невеликих командах, непогано підходить і для особистого планування справ. У Trello можна створити віртуальну дошку з картками, до карток додати учасників, мітки, чек-лист, термін виконання, вкладення та змінити обкладинку.

MyLifeOrganized – програма з легким інтерфейсом, яка підтримує всі функції якісного планувальника завдань: оформлення списків, додавання підзадач, нотаток, тегів та нагадувань. На жаль, створити відразу кілька списків або обговорити завдання з колегами всередині MyLifeOrganized не вдасться. Якщо у вас багато завдань, краще вибрати інше рішення [4, 6, 7, 18].

1.4 Постановка задачі

Таким чином, зручне впорядкування і обробка даних для підвищення ефективності своєї діяльності є актуальним завданням. Тому ставиться завдання розробки застосунку для планування особистого часу.

Об’єктом роботи є процес розробки інформаційної системи.

Метою роботи є розроблення застосунку для планування особистого часу.

Для досягнення мети необхідно вирішити такі завдання:

– ознайомитися з інструментами проєктування та програмування системи, CASE-засобом візуального проєктування даних;

– вивчивши опис предметної області та вимоги до функціональності інформаційної системи, розробити бізнес-правила та глосарій;

– на основі бізнес-правил розробити ER-модель і відобразити її за допомогою ER-діаграми в синтаксисі Чена;

– розробити модель даних за допомогою CASE-засобу візуального проєктування даних;

– сформувати структуру БД в СУБД;

– розробити дизайн бази даних;

– розробити програмний код для забезпечення необхідної функціональності системи;

– провести тестування об’єкту;

– відобразити процес і результати розробки.

2 ПРОЄКТУВАННЯ БАЗИ ДАНИХ

2.1 Принципи проєктування та реалізації реляційних баз даних

Перш ніж перейти до розробки бази даних для предметної області необхідно звернути увагу на основи проєктування та головні теоретичні питання.

2.1.1 Основні етапи життєвого циклу інформаційних систем

Життєвий цикл інформаційної системи – це проміжок часу, який починається з моменту прийняття рішення про необхідність створення інформаційної системи та завершується, коли вона повністю виведена з експлуатації.

Життєвий цикл інформаційної системи поділяється на чотири стадії. Межі кожної стадії визначено деякими моментами часу, коли необхідно прийняти певні ключові рішення і, отже, досягати певних ключових цілей.

Початкова стадія. На початковій стадії встановлюється сфера застосування системи та визначаються граничні умови. Для цього необхідно визначити всі зовнішні об’єкти, з якими має взаємодіяти розроблена система, і визначити характер цієї взаємодії на високому рівні. На початковій стадії ідентифікуються всі функціональні можливості системи і проводиться опис найбільш істотних з них.

Стадія уточнення. На стадії уточнення аналізується область застосування та розробляється архітектурна основа інформаційної системи.

Приймаючи будь-які рішення щодо архітектури системи, необхідно розглядати систему, що розробляється в цілому. Це означає, що необхідно

описати більшість функціональних можливостей системи та врахувати взаємозв’язки між окремими її складовими.

У кінці стадії уточнення проводиться аналіз архітектурних рішень і способів усунення головних факторів ризику у проєкті.

Стадія конструювання. На стадії конструювання розробляється закінчений виріб, готовий до передачі користувачеві.

По закінченні цієї стадії визначається працездатність розробленого програмного забезпечення.

Стадія передачі в експлуатацію. На стадії передачі в експлуатацію розроблене програмне забезпечення передається користувачам. При експлуатації розробленої системи в реальних умовах часто виникають різного роду проблеми, і потрібна додаткова робота по налагодженню розробленого продукту. Зазвичай це відбувається через помилки та недоліки.

У кінці стадії передачі в експлуатацію необхідно визначити, досягнуті цілі розробки чи ні [15, 19–21].

2.1.2 Основні властивості реляційних БД

Реляційна база даних – база даних, заснована на реляційній моделі даних. Для роботи з реляційними БД застосовують реляційні СУБД. Іншими словами, реляційна база даних – це та, яка сприймається користувачами як набір нормалізованих зв’язків різного ступеня.

Реляційна база даних є сукупністю елементів даних, організованих у вигляді набору формально описаних таблиць, з яких дані можуть бути доступними або повторно зібрані багатьма різними способами без необхідності реорганізації таблиць бази даних [22–25].

Реляційні бази даних складаються з таблиць.

Кожна таблиця складається із стовпців (їх називають полями або атрибутами) і рядків (їх називають кортежами). Таблиці в реляційних базах даних мають ряд властивостей:

а) кожен кортежі є унікальним, тобто дублікатів кортежів бути не може;

б) теоретично порядок проходження кортежів у відношенні не має значення (зміна порядку розташування рядків у таблиці не проводить до виникнення нових відносин, але практично цей порядок може істотно вплинути на ефективність доступу до них);

в) порядок проходження атрибутів не має значення (перестановка атрибутів не породжує нової схеми);

г) кожна осередок відносини містить тільки одне атомарне значення, тобто належить одному домену;

д) кожен атрибут має унікальне ім’я;

е) значення атрибута беруться з одного і того ж домена;

є) відношення має ім’я, яке відрізняється від імен всіх інших відносин в реляційної схемі [25–28].

2.1.3 Можливості мови SQL

SQL – декларативна мова програмування, застосовувана для створення, модифікації та управління даними в реляційній базі даних, керованої відповідною системою управління базами даних.

Спочатку SQL був основним способом роботи користувача з базою даних і дозволяв виконувати наступний набір операцій:

- створення в базі даних нової таблиці;
- додавання в таблицю нових записів;
- зміна записів;
- видалення записів;

– вибірка записів з однієї або декількох таблиць (відповідно до заданого умовою);

– зміна структур таблиць.

Оператор SQL складається з зарезервованих слів і призначених для користувача назв.

Зарезервовані слова є постійною частиною мови SQL і мають фіксоване значення. Їх слід записувати в точності так, як це встановлено, і їх можна розбити на частини для переходу від одного рядка до іншого.

Призначені для користувача назви – слова, що визначаються користувачем, задаються користувачем відповідно до синтаксичними правилами і являють собою ідентифікатори або імена різних об’єктів бази даних.

Синтаксичні правила для призначених для користувача назв: Призначені для користувача назви уявляють собою набір символів, який: включає великі та малі літери латинського алфавіту (A-Z, a-z), цифри (0-9) і символ підкреслення (_), назва може мати довжину до 128 символів, починатися з літери, не може містити пробіли. Більшість компонентів мови не чутливі до регістру (виняток – символьна інформація в БД).

Великі літери використовуються для запису зарезервованих слів.

Малі літери вживаються для запису слів, які визначаються користувачем.

Загальний вид запиту виглядає наступним чином:

– SELECT [ALL | DISTINCT] {* | [ім’я_стовпця або вираз [AS нове_і’мя]]} [,... n];

– FROM ім’я_таблиці [[AS] псевдонім] [,... n];

– [WHERE <критерії вибору кортежів>];

– [GROUP BY ім’я_стовпця [,... n]];

– [HAVING <критерії вибору груп>];

– [ORDER BY ім’я_стовпця [,... n]].

Послідовність обробки елементів оператора SELECT визначається так:

– SELECT – встановлюється, які стовпці повинні бути присутніми в вихідних даних;

– FROM – визначаються імена використовуваної таблиці або декількох таблиць;

– WHERE – виконується фільтрація кортежів (рядків) об’єкта відповідно до заданих умов;

– GROUP BY – утворюються групи рядків, що мають одне і те ж значення в зазначеному стовпці;

– HAVING – фільтруються групи рядків об’єкта відповідно до зазначеної умови;

– ORDER BY – визначається впорядкованість результатів виконання оператора.

Порядок конструкцій в операторі SELECT не може бути змінений.

Тільки дві конструкції оператора – SELECT і FROM – є обов’язковими, всі інші конструкції можуть бути опущені.

Операція вибірки за допомогою оператора SELECT є замкнутою, в тому сенсі, що результат запиту до таблиці також є таблицею [29, 30].

2.1.4 Case-засоби моделювання БД

CASE – набір інструментів і методів програмної інженерії для проєктування програмного забезпечення, що допомагає забезпечити високу якість програм, відсутність помилок і простоту в обслуговуванні програмних продуктів.

Появі CASE-технології і CASE-засобів передували дослідження в області методології програмування. Програмування набуло ознак системного підходу з розробкою і впровадженням мов високого рівня, методів структурного і модульного програмування, мов проєктування і засобів їхньої

підтримки, формальних і неформальних мов описів системних вимог і специфікацій і т. д.

Для розробки інформаційної моделі була використана така CASE-програма, як ERwin Data Modeler.

Це комп'ютерна програма для проєктування і документування баз даних. Моделі даних допомагають візуалізувати структуру даних, забезпечуючи ефективний процес організації, управління і адміністрування таких аспектів діяльності підприємства, як рівень складності даних, технологій баз даних та середовища розгортання.

Користувачі можуть використовувати ERwin як засіб створення концептуальної моделі даних і створення логічної моделі даних, яка не залежить від конкретної технології бази даних. Ця схематична модель може бути використана для створення фізичної моделі даних. Потім користувачі можуть направити інженерну мову визначення даних, необхідну для створення схеми для ряду систем управління базами даних. Програмне забезпечення включає в себе функції для графічної модифікації моделі, в тому числі діалогові вікна для вказівки кількості взаємозв'язків сутностей, обмежень бази даних, індексів і унікальності даних.

ERwin створює візуальне уявлення (модель даних) для розв'язуваної задачі. Це уявлення може використовуватися для детального аналізу, уточнення і поширення як частини документації, необхідної в циклі розробки. Логічне моделювання даних: можуть бути створені чисто логічні моделі, з яких можуть бути отримані фізичні моделі. Також підтримуються комбінації логічної та фізичної моделей. Підтримуються типи сутностей і їх атрибути, логічні назви і описи, логічні домени і типи даних, а також назви для зв'язків.

Фізичне моделювання даних: можуть бути створені чисто фізичні моделі, а також комбінації логічних моделей і фізичних моделей. Підтримуються назви та описи таблиць і стовпців, визначені користувачем типи даних, первинні ключі, зовнішні ключі, ключі та альтернативні ключі та

визначення обмежень цілісності. Підтримуються індекси, уявлення, збережені процедури і тригери [31, 32].

2.2 Проєктування бази даних предметної області

Для того, щоб розробники інформаційних систем чітко зрозумів вимоги замовника, їх записуються окремими пунктами – бізнес правилами.

Також, для більшого розуміння системи для замовника пишеться глосарій.

2.2.1 Специфікація вимог

Дана система призначена для ведення обліку бази намічуваних заходів – дата, час і тривалість, місце проведення.

У створюваній системі необхідно забезпечити введення, видалення і перегляд даних в зручній для користувача формі та повинні зберігатися дані.

Інформаційна система повинна виконувати такі дії:

а) додавання справи в базу даних;

б) перегляд справ на день;

в) видалення справи.

2.2.2 Розробка бізнес-правил БД та глосарію БД

Розробленні бізнес-правила БД:

а) в день багато справ;

б) одна справа може бути в багато днів;

в) в справі одне місце проведення;

г) в справі в день один час початку та тривалість.

Розроблений глосарій БД: користувач – особа, яка користується даною програмою.

2.3 Розробка концептуальної моделі БД предметної області

2.3.1 Визначення типів сутностей

Примірник сутності – однозначно ідентифікує об’єкт, який відноситься до суті певного типу.

Кожен тип сутності має унікальний набір атрибутів.

Кожна окрема сутність має свої власні значення для кожного атрибута.

Слабкий тип сутності – тип сутності, існування якого залежить від якогось іншого типу сутності (як первинний ключ або його частини використовується первинний ключ іншої сутності).

Сильний тип сутності – тип сутності, існування якого не залежить від якогось іншого типу сутності (як первинний ключ використовуються тільки власні атрибути даної суті).

В даній концептуальній моделі будуть існувати наступні сутності та їх атрибути(всі вони сильні на цьому етапі розробки):

а) дата:

1) Date (Datetime) – дата;

2) Day (String, VARCHAR (20)) – день тижня;

б) справа:

1) Name (String, VARCHAR (20)) – назва справи;

2) Place (Number) – місце проведення;

в) час:

1) Date (Datetime) – дата;

2) Name (String, VARCHAR (20)) – назва справи;

3) User (String, VARCHAR (20)) – ім’я користувача;

4) Time (String, VARCHAR (20)) – час початку;

5) Length (String, VARCHAR (20)) – тривалість;

г) місце:

1) Id (Number) – номер місця проведення;

2) Name (String, VARCHAR (20)) – назва місця проведення;

д) користувачі:

1) Login (String, VARCHAR (20)) – ім’я користувача;

2) Password (String, VARCHAR (20)) – пароль користувача;

3) Admin (Number) – адміністратор.

2.3.2 Визначення первинних та вторинних ключів, зв’язок між сутностями

Ключ – це стовпець (може бути декілька стовпців), що додається до таблиці і дозволяє встановити зв’язок з іншими таблицями. Існують ключі таких типів: первинні, потенційні, зовнішні.

Потенційний ключ – це мінімальна множина атрибутів, що є підмножиною заголовка даної сутності, складене значення котрих унікально визначає кортеж сутності.

Первинний ключ – це потенційний ключ, котрий обраний для унікальної ідентифікації кортежів всередині сутності.

Зовнішній (вторинний) ключ – це атрибут чи множина атрибутів всередині сутності, котре відповідає потенційному ключу деякою сутності.

Визначивши сутності необхідно зв’язати їх між собою.

Для цього визначимо первинні ключі.

Усі сутності мають ідентифікаційний номер, який визначає унікальність їхніх елементів. Вторинні ключі визначаються зв’язком між сутностями.

Також в сутностях є транзитивні залежності, які будуть розкритті за допомогою додавання нових сутностей.

2.3.3 Побудова логічної моделі даних

Після визначення ключів, типів сутностей та зв’язків можна побудувати логічну модель за допомогою ERwin Data Modeler. Схема бази даних зображена на рисунку 2.1.

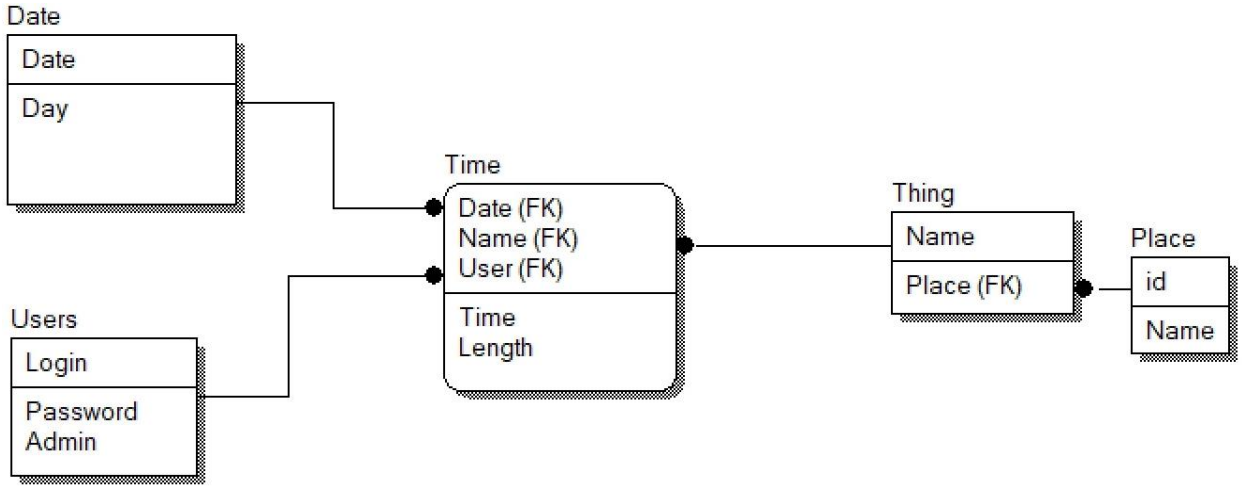

Рисунок 2.1 – Схема бази даних

Розкриття схеми виглядає наступним чином:

– Дата (Date, Day);

– Справа (Name, Place);

– Час (Date, Name, User, Time, Length);

– Місце (Id, Name);

– Користувачі (Login, Password, Admin).

2.3.4 Діаграма системи в SQL

Написання таблиць має результат у вигляді діаграми в системі SQL Server, яка зображена на рисунку 2.2.

Це необхідно для перевірки результатів створення таблиць за допомогою Erwin (перенесення).

Виконуються всі зв’язки, які зображні на логічному рівні моделі, а також що вдосконалює нашу систему ще до переходу до написання програмного коду.

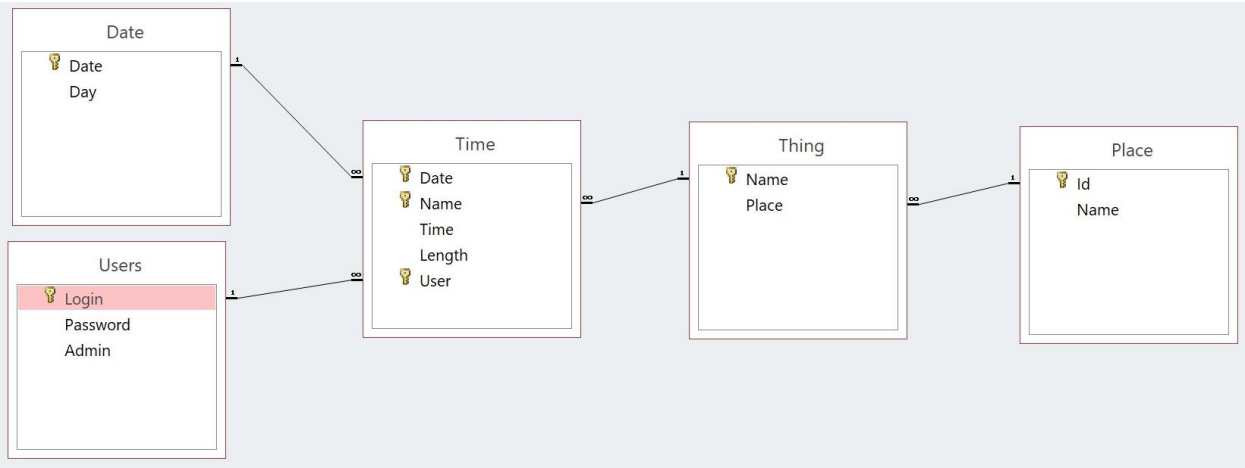

Рисунок 2.2 – Діаграма в системі SQL Server

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАСТОСУНКУ

3.1 Інструменти для розробки програмного застосунку

Для розробки були вибрані наступні інструменти: SQL, Microsoft SQL Server, Erwin Data Modeler, C#, Microsoft Visual Studio.

SQL (англ. structured query language – «мова структурованих запитів») – це декларативна мова програмування, що використовується для створення, зміни та управління даними в реляційній базі даних, що керується відповідною системою управління базами даних.

Є насамперед інформаційно-логічною мовою програмування, призначеною для опису, зміни та вилучення даних, що зберігаються в реляційних базах даних. У загальному випадку SQL (без ряду сучасних розширень) вважається мовою програмування неповною за Тьюрингом, але водночас стандарт мови специфікацією SQL/PSM передбачає можливість його процедурних розширень.

Microsoft SQL Server – система управління реляційними базами даних, розроблена корпорацією Microsoft. Основна мова запитів – Transact-SQL, створений спільно Microsoft і Sybase. Transact-SQL є реалізацією стандарту ANSI/ISO із структурованої мови запитів (SQL) з розширеннями. Використовується для роботи з базами даних усіх розмірів, від персональних до великих баз даних масштабу підприємства; конкурує з іншими СУБД у цьому сегменті ринку.

Erwin Data Modeler – це комп’ютерна програма для проєктування та документування баз даних. Моделі даних допомагають візуалізувати структуру даних, забезпечуючи ефективний процес організації, управління та адміністрування таких аспектів діяльності підприємства, як рівень складності даних, технологій баз даних та середовища розгортання. Спочатку розроблений компанією Logic Works, erwin був придбаний рядом компаній,

перш ніж був виділений приватною інвестиційною компанією Parallax Capital Partners, яка придбала та об’єднала його як окрему організацію, erwin, Inc.

Програмне забезпечення засноване на методі IDEF1X, хоча тепер воно також підтримує діаграми, які відображаються з альтернативною інженерною нотацією для інформаційних технологій, а також нотацію для розмірного моделювання.

C# – об’єктно-орієнтована мова програмування. Розроблена у 2001 році групою інженерів компанії Microsoft як мова розробки програм для платформи Microsoft .NET Framework та .NET Core. Згодом був стандартизований як ECMA-334 та ISO/IEC 23270.

C# належить до сімейства мов із C-подібним синтаксисом, їх синтаксис найбільш близький до C++ і Java. Мова має статичну типізацію, підтримує поліморфізм, перевантаження операторів (у тому числі операторів явного та неявного приведення типу), делегати, атрибути, події, змінні, властивості, узагальнені типи та методи, ітератори, анонімні функції з підтримкою замикань, LINQ, винятки, коментарі у форматі XML.

Перейнявши багато від своїх попередників – мов C++, Delphi, Smalltalk і, особливо, Java – C#, спираючись на практику їх використання, виключає деякі моделі, що зарекомендували себе проблематичними при розробці програмних систем, наприклад, C# на відміну від C++ не підтримує множинне успадкування класів (між тим допускається множинна реалізація інтерфейсів).

Microsoft Visual Studio – лінійка продуктів компанії Microsoft, що включають інтегроване середовище розробки програмного забезпечення та інші інструменти. Дані продукти дозволяють розробляти як консольні програми, так і ігри та програми з графічним інтерфейсом, у тому числі з підтримкою технології Windows Forms, UWP а також веб-сайти, веб-програми, веб-служби як в рідному, так і в керованому кодах всіх платформ, підтримуваних Windows, Windows Mobile, Windows CE, .NET Framework, .NET Core, .NET, MAUI, Xbox, Windows Phone, .NET Compact Framework та

Silverlight. Після придбання компанії Xamarin корпорацією Microsoft з’явилася можливість розробки IOS та Android програм.

3.2 Етапи розроблення застосунку

3.2.1 Створення форм

Першим етапом в розробці застосунку буде створення форм. При запуску застосунку відкриється форма входу. На ній можна увійти або зареєструватися (рис. 3.1).

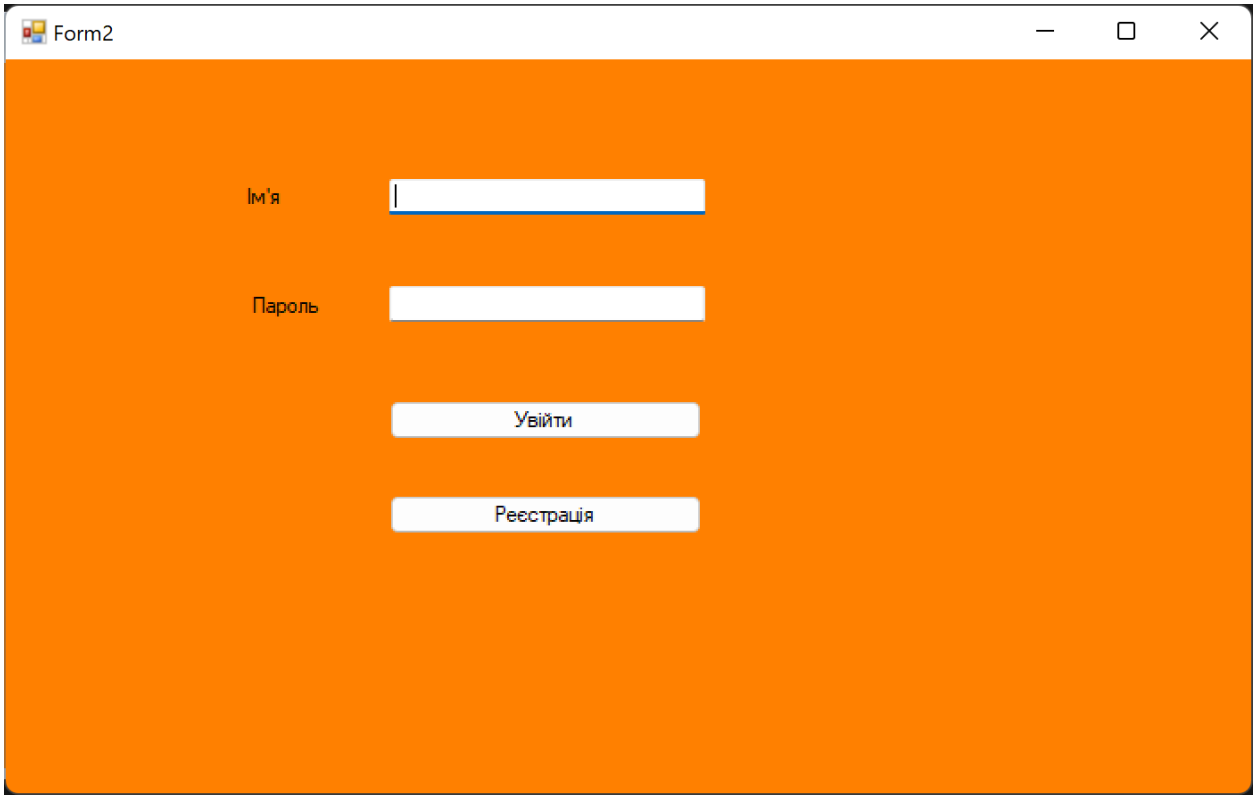

Рисунок 3.1 – Форма входу

Після входу відкриється форма перегляду записів про справи (рис. 3.2). На ній користувач може подивитися всі свої записи, подивитися тільки за одну дату, видалити запис та додати.

Рисунок 3.2 – Форма перегляду

Для додавання записів відкривається нова форма (рис. 3.3).

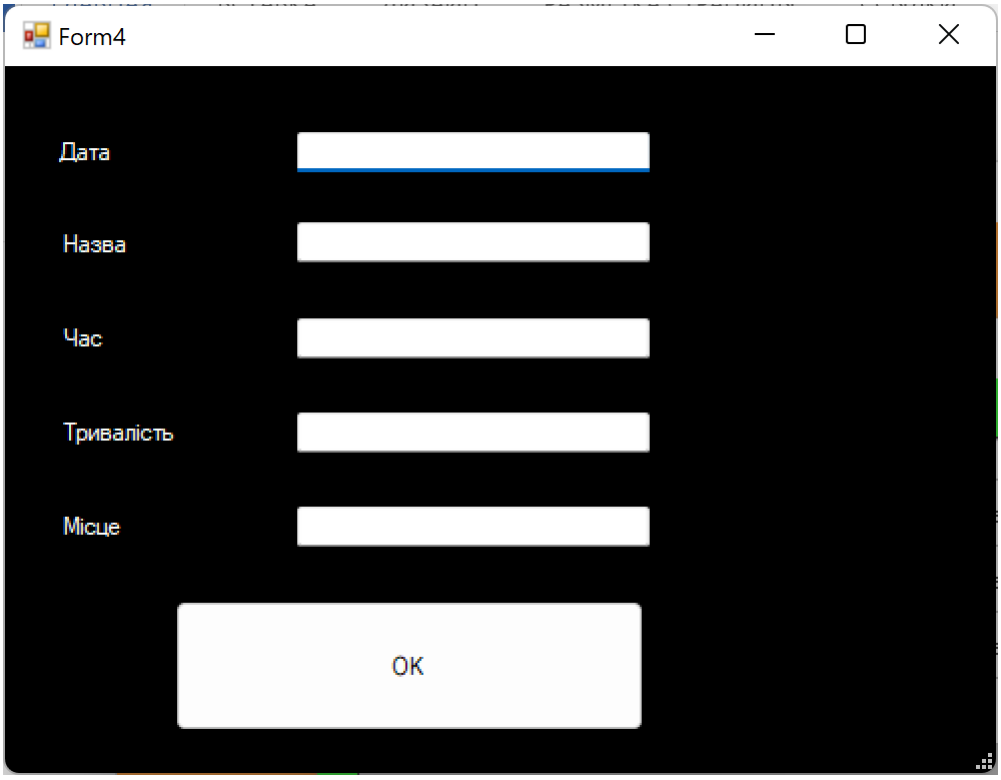

Рисунок 3.3 – Форма додавання записів

3.2.2 Написання коду застосунку

Для роботи з БД створено клас Query з методами Login (вхід), Registration (реєстрація), Update (оновлення записів), Filter (перегляд за одну дату), Delete (видалення), Add (додавання). Використовується набір інтерфейсів OLE DB (рис. 3.4).

```
class Query
{
    OleDbConnection connection;
    OleDbCommand command;
    OleDbDataAdapter dataAdapter;
    DataTable bufferTable;

    Ссылок: 4
    public Query(string Conn)
    {
        connection = new OleDbConnection(Conn);
        bufferTable = new DataTable();
    }
```

Рисунок 3.4 – Частина коду програми

Для входу використовується метод Login. Створюється запит до БД. Якщо користувача з таким ім’ям і паролем не знайдено показується повідомлення про помилку (рис. 3.5). Якщо знайдено, перевіряється, чи цей користувач адміністратор. Код метода приведено на рисунку 3.6.

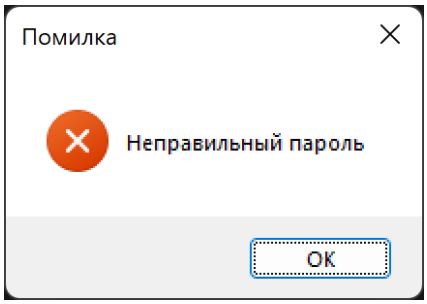

Рисунок 3.5 – Повідомлення про помилку

```
public int Login(string user, string password)
{
    connection.Open();
    dataAdapter = new OleDbDataAdapter("SELECT Login,Password,Admin FROM Users " +
        "WHERE Login="+'"'+user+'"'+" AND Password="+'"'+password + '"', connection);
    bufferTable.Clear();
    dataAdapter.Fill(bufferTable);
    connection.Close();
    if (bufferTable.Rows.Count > 0)
    {
        dataAdapter = new OleDbDataAdapter("SELECT Login,Password,Admin FROM Users WHERE Admin=1 " +
            "AND Login=" + '"' + user + '"' + " AND Password=" + '"' + password + '"', connection);
        bufferTable.Clear();
        dataAdapter.Fill(bufferTable);
        connection.Close();
        if (bufferTable.Rows.Count > 0) return 2;
        else return 1;
    }
    else return 0;
}
```

Рисунок 3.6 – Метод Login

Адміністратор може працювати з записами всіх користувачів. Після цього відкривається форма перегляду з потрібним параметром (рис. 3.7).

```
private void button1_Click(object sender, EventArgs e)
{
    int l = controller.Login(textBox1.Text, textBox2.Text);
    if (l == 0) MessageBox.Show("Неправильный пароль", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
    if (l == 1)
    {
        Form1 newform = new Form1(textBox1.Text);
        newform.Owner = this;
        newform.ShowDialog();
        this.Close();
    }
    if (l == 2)
    {
        Form1 newform = new Form1("Admin");
        newform.Owner = this;
        newform.ShowDialog();
        this.Close();
    }
}
```

Рисунок 3.7 – Частина коду програми

Для реєстрації використовується метод Registration (рис. 3.8). Створюється запит до БД. Якщо користувач з таким ім’ям є в БД або пароль коротше 4 знаків, показується повідомлення про помилку. Якщо ні, користувач додається до бази даних і показується повідомлення про реєстрацію (рис. 3.9).

```
public int Registration(string user, string password)
{
    connection.Open();
    dataAdapter = new OleDbDataAdapter("SELECT Login FROM Users WHERE Login=" + '"' + user + '"', connection);
    bufferTable.Clear();
    dataAdapter.Fill(bufferTable);
    connection.Close();
    if (bufferTable.Rows.Count > 0) return 0;
    else if (password.Length < 4) return 1;
    else
    {
        connection.Open();
        command = new OleDbCommand($"INSERT INTO [Users]([Login], [Password], [Admin]) VALUES(user, password, 0)", connection);
        command.Parameters.AddWithValue("Login", user);
        command.Parameters.AddWithValue("Password", password);
        command.Parameters.AddWithValue("Admin", 0);
        command.ExecuteNonQuery();
        connection.Close();
        return 2;
    }
}
```

Рисунок 3.8 – Метод Registration

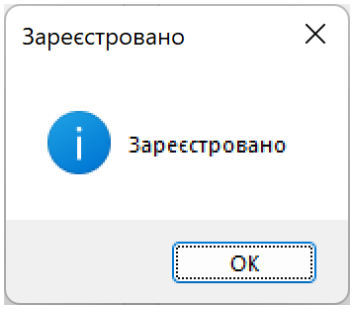

Рисунок 3.9 – Повідомлення про реєстрацію

На формі перегляду є кнопки оновлення, додавання, видалення записів, та перегляд записів тільки за дату, введену в полі.

Для оновлення записів використовується метод Update (рис. 3.10). Створюється запит до БД з вибором всіх записів, доступних цьому користувачу. Вибирається дата, день тижня, назва справи, місце проведення, час початку, тривалість.

```
public DataTable Update(string user)
{
    connection.Open();
    if (user == "Admin") dataAdapter = new OleDbDataAdapter("SELECT Time.Date,Day,Time.Name,Place,Time.Time,Length,User " +
        "FROM [Date] INNER JOIN [Time] ON Date.Date = Time.Date", connection);
    else dataAdapter = new OleDbDataAdapter("SELECT Time.Date,Day,Time.Name,Place,Time.Time,Length FROM [Date] " +
        "INNER JOIN [Time] ON Date.Date = Time.Date WHERE User=" + '"' + user + '"', connection);
    bufferTable.Clear();
    dataAdapter.Fill(bufferTable);
    connection.Close();
    return bufferTable;
}
```

Рисунок 3.10 – Метод Update

На формі додавання записів користувач вводить дату, назву справи, місце проведення, час початку, тривалість. З методом Add запис про справу додається до бази даних (рис. 3.11).

```
public void Add(DateTime Date, string Name, string Time, string Length, string Place, string User)
{
    connection.Open();
    command = new OleDbCommand($"INSERT INTO [Time]([Date], [Name], [Time], [Length], [Place], [User]) " +
        $"VALUES(Date, Name, Time, Length, Place, User)", connection);
    command.Parameters.AddWithValue("Date", Date);
    command.Parameters.AddWithValue("Name", Name);
    command.Parameters.AddWithValue("Time", Time);
    command.Parameters.AddWithValue("Length", Length);
    command.Parameters.AddWithValue("Place", Place);
    command.Parameters.AddWithValue("User", User);
    command.ExecuteNonQuery();
    connection.Close();
}
```

Рисунок 3.11 – Метод Add

Для перегляду записів тільки за дату, введену в полі використовується метод Filter (рис. 3.12). Він працює як метод Update, але з вибором записів тільки за потрібну дату.

```
public DataTable Filter(DateTime Date, string user)
{
    connection.Open();
    string y = Date.ToString("yyyy");
    string m = Date.ToString("MM");
    string d = Date.ToString("dd");
    if (user == "Admin")  dataAdapter = new OleDbDataAdapter("SELECT Time.Date,Day,Time.Name,Place,Time.Time,Length,User " +
        "FROM [Date] INNER JOIN[Time] ON Date.Date = Time.Date " +
        $"WHERE Time.Date=# {m}/{d}/{y}#", connection);
    else dataAdapter = new OleDbDataAdapter("SELECT Time.Date,Day,Time.Name,Place,Time.Time,Length " +
        "FROM [Date] INNER JOIN[Time] ON Date.Date = Time.Date " +
        $"WHERE Time.Date=# {m}/{d}/{y}# AND User=" + '"' + user + '"', connection);
    bufferTable.Clear();
    dataAdapter.Fill(bufferTable);
    connection.Close();
    return bufferTable;
}
```

Рисунок 3.12 – Метод Filter

Для видалення записів використовується метод Delete (рис. 3.13). Користувач вибирає запис на формі перегляду, і з БД видаляється потрібний запис з вибраними датою, назвою та користувачем (рис. 3.14).

```
public void Delete(DateTime Date, string Name, string user)
{
    connection.Open();
    string y = Date.ToString("yyyy");
    string m = Date.ToString("MM");
    string d = Date.ToString("dd");
    if (user == "Admin") command = new OleDbCommand($"DELETE FROM [Time] WHERE Date =# {m}/{d}/{y}# " +
        $"AND Name=" + '"' + Name + '"', connection);
    else command = new OleDbCommand($"DELETE FROM [Time] WHERE Date =# {m}/{d}/{y}# " +
        $"AND Name=" + '"' + Name + '"' + "AND User=" + '"' + user + '"', connection);
    command.ExecuteNonQuery();
    connection.Close();
}
```

Рисунок 3.13 – Метод Delete

```
private void button3_Click(object sender, EventArgs e)
{
    controller.Delete(DateTime.Parse(dataGridView1.Rows[dataGridView1.CurrentRow.Index].Cells["Date"].Value.ToString()),
        dataGridView1.Rows[dataGridView1.CurrentRow.Index].Cells["Name"].Value.ToString(), user);
    dataGridView1.DataSource = controller.Update(user);
}
```

Рисунок 3.14 – Частина коду програми

ВИСНОВКИ

У рамках кваліфікаційної роботи був розроблений і реалізований застосунок для планування особистого часу.

Була спроєктована база даних, розроблені бізнес-правила БД та глосарій БД. Була розроблена концептуальна модель БД предметної області, визначені типи сутностей. Були визначені первинні та вторинні ключі, зв’язок між сутностями. Після цього була побудована логічна модель за допомогою ERwin Data Modeler, та перенесена до системи Microsoft SQL Server. Після проєктування БД був розроблений застосунок, створені форми, та написаний код програми за допомогою Microsoft Visual Studio та мови програмування C#. Були зроблені функції перегляду, додавання записів, їх видалення, сортування записів.

Усі поставлені вимоги було виконано. В результаті був розроблений застосунок для планування особистого часу з яким зможе працювати користувач без навичок адміністрування та розробки баз даних.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Ануфриева, А. А., & Горбунова, Е. С. (2020). Параметры рабочей памяти и внимания (устойчивость, переключаемость) у пользователей оффлайн и онлайн органайзеров. In PSY-Вышка (pp. 139-141).

2. Сорокина, А. А., Каплина, А. А., Чухлов, И., & Падерин, Д. Р. (2020). РАЗРАБОТКА ПРИЛОЖЕНИЯ ОРГАНАЙЗЕРА. In ВНЕДРЕНИЕ РЕЗУЛЬТАТОВ ИННОВАЦИОННЫХ РАЗРАБОТОК: ПРОБЛЕМЫ И ПЕРСПЕКТИВЫ (pp. 48-52).

3. Попков, С. А., & Никитина, С. А. (2020). Разработка календаря-планировщика. In Наука сегодня: опыт, традиции, инновации (pp. 15-16).

4. Григорьева, П. А., & Вусович, О. В. (2020). Инновации в тайм-менеджменте.

5. Штомпель, И. (2017). Популярные органайзеры: какой выбрать руководителю?. БИТ. Бизнес & Информационные технологии, (3), 48-51.

6. Шарикова, Ю. В., Малышева, Е. А., & Кутуев, А. В. (2019). Тайм-менеджмент как инструмент самоорганизации студентов в условиях цифровой экономики. Вестник Самарского муниципального института управления, (1), 128-137.

7. Сулыма, А. И., & Пасечник, О. А. (2021). СОВРЕМЕННЫЕ ТЕХНОЛОГИИ ТАЙМ-МЕНЕДЖМЕНТА ОРГАНИЗАЦИИ.

8. Князева, Л. А., Богомолова, И. П., & Шатохина, Н. М. (2020). ЦИФРОВИЗАЦИЯ УПРАВЛЕНЧЕСКИХ БИЗНЕС-ПРОЦЕССОВ В HR-СФЕРЕ НА ОСНОВЕ КОРПОРАТИВНОГО ПО. In Стратегия и тактика управления предприятием в переходной экономике (pp. 94-98).

9. Рибокене, Е. В., & Леденев, В. А. (2020). РАЗВИТИЕ АВТОМАТИЗАЦИИ ПРОЦЕССОВ ПЛАНИРОВАНИЯ РЕСУРСОВ И ИНФОРМАЦИОННОГО ОБЕСПЕЧЕНИЯ СИСТЕМЫ УПРАВЛЕНИЯ

ПРЕДПРИЯТИЯМИ. In НАУКА И ОБРАЗОВАНИЕ: БУДУЩЕЕ И ЦЕЛИ УСТОЙЧИВОГО РАЗВИТИЯ (pp. 226-231).

10. Танянский, С. С., Тулупов, В. В., & Руденко, Д. А. (2006). Модификация ограничений на ведение данных для обеспечения целостности крупномасштабных информационных систем. Вестник Национального технического университета Харьковский политехнический институт. Серия: Информатика и моделирование, (23), 137-144.

11. Радченко, В. А., & Руденко, Д. А. (2010). Исследование возможности применения подхода реструктуризации гетерогенных источников данных в задачах интеграции информации.

12. Урюпина, Е. В. (2019). История развития технологии проектирования информационных систем. In Наука и образование: прошлое, настоящее и будущее (pp. 16-18).

13. Субботина, Т. В. (2020). ОБЩЕСТВЕННОЕ РАЗВИТИЕ В СВЕТЕ ЦИФРОВЫХ ТЕХНОЛОГИЙ. In Цифровая География (pp. 102-106).

14. Аршинский, В. Л., Аршинский, Л. В., & Бахвалов, С. В. (2019). Принципы необходимости и достаточности в систематизации программного обеспечения. Автоматизация и моделирование в проектировании и управлении, (2 (4)), 18-24.

15. Филатов, В. А., Руденко, Д. А., & Гринева, Е. Е. (2014). Средства интеграции неоднородных данных в корпоративных информационно-телекоммуникационных системах.

16. Казарин, О., & Шубинский, И. (2022). Надежность и безопасность программного обеспечения. Учебное пособие для бакалавриата и магистратуры. Litres.

17. Наумов, А. А., & Айдинян, А. Р. (2018). Надежность программного обеспечения и методы ее повышения. Инженерный вестник Дона, (2 (49)), 98.

18. Тютюгина, Е. В., & Валитова, Н. Л. (2020). Сравнение приложений для передачи фотографий и приложений-уведомлений. StudNet, 3(9), 788-792.

19. Руденко, Д. А., & Филатов, В. А. (2013). Формальный подход к описанию свойств данных в информационных системах. Вестник Херсонского национального технического университета, (1 (46)), 146-149.

20. Руденко, Д. А., & Тулупов, В. В. (2007). Модель и средства поддержки данных в задачах интеграции неоднородных информационных систем. Системи обробки інформації, (4), 110-114.

21. Илюшечкин, В. (2021). Основы использования и проектирования баз данных. Учебник для академического бакалавриата. Litres.

22. Гринева, Е.Е., Танянский, С. С., & Руденко, Д. А. (2012). Однозначность ограниченной целостности при модификации семантики базы данных. Вестник Херсонского национального технического университета, (1), 149-154.

23. Нестеров, С. А. (2018). Базы данных.

24. Илюшечкин, В. (2018). Основы использования и проектирования баз данных. Litres.

25. Стружкин, Н. П., & Годин, В. В. (2018). БАЗЫ ДАННЫХ: ПРОЕКТИРОВАНИЕ.

26. Овчинников, А. С. (2018). Аналитический обзор методов проектирования баз данных для потребностей коммерческой организации. Инженерно-технические решения и инновации, (4), 10-26.

27. Танянский, С. С., & Руденко, Д. А. (2005). Формальная система построения объектов предметной области в задачах интеграции информационных систем. Образование, наука, производство и управление в XXI веке: сб. работ регион. научн. конф (1). 276-280.

28. Radchenko, V. O., Rudenko, D. O., Tkachov, V. M., & Tokarev, V. V. (2017). Мобильная подсистема «Мультикоптер-сенсорная сеть» в компьютерной системе хранения BIG DATA. Системи управління, навігації та зв'язку. Збірник наукових праць, 4(44), 102-105.

29. Моргунов, Е. П., Рогова, Е. В., & Лузанова, П. В. (2018). PostgreSQL. Основы языка SQL. учеб. пособие/ЕП Моргунов.

30. Кондрашов, Ю. Н. (2018). ЯЗЫК SQL.

31. Жалолов, О. И., & Хаятов, Х. У. (2020). Понятие SQL и реляционной базы данных. Universum: технические науки, (6-1 (75)), 26-29.

32. Старушенкова, Е. Е., Шиманова, Е. Н., & Радаев, К. Д. (2020). Язык SQL как средство создания баз данных. E-Scio, (3 (42)), 325-330.