ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут телекомунікацій

Кафедра системного аналізу

Пояснювальна записка

до бакалаврської роботи
на ступінь вищої освіти бакалавр
на тему: **«Інформаційна система автоматизованого формування розкладу»**

Виконав: студент 4 курсу,
групи САД–41 спеціальності
124 Системного аналізу
(шифр і назва спеціальності)

Гомоляко В. І.
(прізвище та ініціали)

Керівник Резник С. Ю.
(прізвище та ініціали)

Рецензент
(прізвище та ініціали)

Київ – 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут телекомунікацій

Кафедра Системного аналізу_____
Ступінь вищої освіти - «Бакалавр»_____
Спеціальність - 124 Системний аналіз_____

ЗАТВЕРДЖУЮ
Завідувач кафедри
Системного аналізу
_____Гордієнко Т.Б

“____” _____2023 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

_____Гомоляко Валерій Ігорович_____

(прізвище, ім‘я, по батькові)

1.Тема роботи:

Інформаційна система автоматизованого формування розкладу

Керівник роботи____Резник С. Ю._____,

(прізвище, ім‘я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від “ “_____2023 року №__.

2. Строк подання студентом роботи_____

3. Вхідні дані до роботи:

Науково-технічна література з питань складання автоматизованого розкладу,

платформа .NET Framework і мова програмування C++, інструменти розробки програмних засобів.

Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити).

- Аналіз предметної області та постановка завдання
- Проектування і розробка програмного модулю формування розкладу
- Особливості використання програмного модулю

Перелік графічного матеріалу

Презентація_____

Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області та постановка завдання	04.03-09.03	Виконано
2	Проектування і розробка програмного модулю формування розкладу	10.04-20.03	Виконано
3	Особливості використання програмного модулю	21.03-26.03	Виконано
4	Реалізація системи	26.03-06.05	Виконано
5	Висновок	06.05-13.05	Виконано
6	Перевірка на антиплагіат	14.05-01.06	Виконано
7	Захист роботи	07.06	Виконано
8	Випуск		

Студент _____ Гомоляко В. І.

(підпис) (прізвище та ініціали)

Керівник роботи _____ Резник С. Ю.

(підпис) (прізвище та ініціали)

ЗМІСТ

ВСТУП

Актуальність дослідження. В сучасному світі, де швидкість інформаційного обміну набуває все більшого значення, ефективне управління ресурсами та процесами стає невід'ємною частиною успішної діяльності різних організацій. Одним з ключових аспектів такого управління є ефективне планування та керування розкладом роботи.

Крім того, інформаційна система автоматизованого формування розкладу може виявитися корисною для широкого спектру організацій, включаючи навчальні заклади, бізнес-підприємства, медичні установи та інші сфери діяльності, де необхідно регулярно розподіляти ресурси та планувати робочий час.

Система може допомогти вирішити проблеми, пов'язані з конфліктами графіків роботи, недостатньою використаністю ресурсів або неправильним розподілом завдань та обов'язків. Вона може автоматизувати процес розкладування, враховуючи різноманітні фактори, такі як доступність працівників, обсяг роботи, часові обмеження та пріоритети.

Інформаційна система може забезпечити зручний доступ до розкладу для всіх зацікавлених сторін, надаючи їм можливість переглядати розклад, вносити зміни або робити запити на зміну. Це сприятиме забезпеченню прозорості та комунікації між всіма учасниками процесу.

Завдяки використанню інформаційної системи автоматизованого формування розкладу, організації матимуть можливість зберігати, аналізувати та використовувати дані про розклади роботи для підвищення ефективності та прийняття обґрунтованих управлінських рішень. Крім того, система може забезпечити автоматичне сповіщення та нагадування щодо змін у розкладі, що сприятиме покращенню комунікації та уникненню недорозумінь.

Швидкий розвиток інформаційних технологій та зростання їх доступності створюють унікальні можливості для автоматизації та оптимізації процесу формування розкладу. Використання сучасних інструментів, таких як бази даних, алгоритми оптимізації та веб-розробка, дозволяє розробити потужну та гнучку інформаційну систему, яка відповідає потребам сучасного світу.

Збільшення складності та обсягу завдань, які необхідно враховувати при розкладуванні, вимагає використання комп'ютеризованих інструментів для ефективного управління цими процесами. Інформаційна система автоматизованого формування розкладу може враховувати різноманітні фактори, такі як доступність працівників, обсяг роботи, часові обмеження та пріоритети, що допомагає забезпечити оптимальне використання ресурсів та покращити продуктивність.

Отже, розробка інформаційної системи автоматизованого формування розкладу є актуальною задачею, яка може принести значні переваги організаціям у плануванні та управлінні ресурсами, знижуючи затрати часу та зусиль, покращуючи точність та ефективність процесу формування розкладу.

Розробкою даної проблеми займалося багато науковців, представники різних галузей науки: М. Ярку (M. Jarke) і Ю. Коха (J. Koch) включає в себе перелік із понад 253 робіт, М. Манніно (MV Mannino), П. Чу (P. Chu) і Т.Сейджера (T. Sager) - 64 роботи), Я. Іоннідіса (YE Ionnidis)-більше 50 робіт, С. Чаудхари (S. Chaudhari) - 61 робота, С. Д. Кузнєцова - 128 робіт. Огляди методів оптимізації також містяться в тематичних розділах робіт Г. Грейфа (G. Graefe), К. Дейта (C. Date), М. Т. Оцсу (M. Ozsu) і П. Валдуреза (P. Valduriez), Р. Рамакрішнан (R. Ramakrishnan) і Дж. Герке (J. Gehrcke).

Але, незважаючи на це, сьогодні існує потреба у дослідженні, яке б узагальнило, систематизувало існуючі відомості з даної проблеми.

Враховуючи все вищесказане, нами і була обрана тема дипломної роботи:

"Інформаційна система автоматизованого формування розкладу".

Об’єкт дослідження – Сучасний стан вирішення проблеми продуктивності програмних засобів автоматизації формування розкладу .

Предмет – способи та методи автоматизації формування розкладу.

Мета роботи: дослідити основні аспекти автоматизації формування розкладу.

Відповідно до мети були визначені наступні **завдання**:

1) провести дослідження основних проблем предметної області;

2) Обґрунтувати вибір мови програмування та розробки бази данних продукту

3) дослідити особливості розробки програмного продукту;

4) провести аналіз отриманих результатів.

Для розв’язання поставлених завдань нами були використані такі **методи дослідження**: теоретико-критичний аналіз літератури з теми дослідження; зіставлення, узагальнення і синтезування здобутої інформації тощо.

Робота може бути використана студентами ВНЗ для підготовки до семінарських занять, також може бути використана викладачами для проведення лекції, практик тощо.

Структура роботи. Робота складається зі вступу, трьох розділів, висновків, списку використаних джерел, що містить 60 найменувань та додатків. Повний обсяг роботи: 93 сторінки.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Аналіз предметної області

Початковим етапам дипломного проекту, є виявлення предметної області. Вона відіграє більшу роль в аналізі, структуруванні даних і дозволяє класифікувати й формалізувати інформацію про всі процеси даного дослідження.

Предметна область - сукупність зв'язаних між собою функцій, завдань керування, за допомогою яких досягається виконання поставлених цілей, це частина реального миру, що представляє інтерес для конкретного дослідження.

Навчальний відділ здійснює контроль над організацією навчального процесу на факультетах і кафедрах [8, с.73].

Навчальний відділ вчасно представляє факультетам і кафедрам документацію, необхідну для ведення навчального процесу.

Факультети й кафедри зобов'язано представити в навчальний відділ у строки, певні відомості й учбово-методичну документацію, пов'язану з навчальним процесом і роботою професорсько-викладацького складу.

Навчальний відділ дає дозвіл викладачам кафедр на роботу із трудових угод і по сумісництву.

Усі заявки навчального відділу, пов'язані з виконанням робіт із забезпечення навчального процесу, виконуються підрозділами філії в першу чергу [33, с. 54].

І так, ми бачимо активний взаємозв'язок з кафедрами й факультетами навчального закладу, що утворює інформаційні потоки, тобто переміщення інформації від одного суб'єкта освітнього простору до іншого, що сприяють забезпеченню організації усередині установи. Представимо у вигляді моделі (рис. 1.1).

Рис.1.1. Модель інформаційних потоків

Контроль у ВНЗ по організації навчального процесу здійснюється ректоратом, департаментом по учбово-методичній роботі, деканами, завідувачами кафедр.

В обов'язку начальника навчального відділу входить:

Контролювати відповідність розкладу навчальних занять навчальним планам підготовки фахівців, а також нормативним документам організації навчального процесу.

Контролювати рівномірне й своєчасне планування практичних і лабораторних занять протягом семестру [29, с. 68].

Здійснює контроль виконання викладачами запланованого обсягу навчальної, учбово-методичної, організаційно-методичної й виховної робіт.

Здійснює контроль виконання професорсько-викладацьким складом затверджених розкладів навчальних занять, екзаменаційних сесій і іспитів підсумкової державної атестації.

Організовувати роботу методистів відділу.

Підготовляти проекти наказів по організації навчального процесу.

Співробітники цього підрозділу відповідно виконують поставлені завдання.

Дана діяльність займає багато часу й вимагає більших працезатрат. Тому для полегшення роботи навчального відділу, внаслідок дослідження предметної

області, було обрано для автоматизації одне із самих складних і трудомістких справ, виконуваних співробітниками - технологія складання розкладу.

Наше завдання - створити програму по організації навчальних занять. Для цього необхідно розглянути всі аспекти обраної області.

Розклад складається в строгій відповідності з навчальними планами спеціальностей:

- по аудиторному навантаженню;
- по строках початку й закінчення семестру й сесії.

Навчальний розклад повинний забезпечити рівномірне навантаження навчальною роботою студентів протягом семестру, місяців, тижня й дня, виконання дидактичних і методичних вимог і раціональне використання навчальних кабінетів, лекційних аудиторій і викладацького складу.

Один з основних аспектів для комфортної роботи викладачів і студентів є контроль над навчальним розкладом. При його складанні пропонується керуватися наступним:

Навчальний розклад повинний складатися в точній відповідності із затвердженим навчальним планом [41, с. 55].

Навчальний процес може бути організований на основі технологій регулярного або блокового навчання. Звичайно, студенти протягом семестру вивчають близько десяти дисциплін.

При технології регулярного навчання студенти слухають лекції й займаються на практичних заняттях по всіх дисциплінах протягом усього семестру. Заліки ухвалюються наприкінці семестру, а іспити студенти здають під час екзаменаційної сесії. За технологією регулярного навчання займаються студенти очного й очно- заочного відділень.

При технології блокового навчання студенти вивчають протягом усього семестру тільки ті дисципліни, які вимагають великої кількості навчальних годин. В основному дисципліни вивчаються компактно. Іспити й заліки

проводяться під час навчального семестру відразу після завершення лекційних і практичних занять. За технологією блокового навчання займаються студенти заочного відділення.

Навчальні заняття повинні бути організовані по твердому навчальному розкладу, що передбачає безперервність навчального процесу протягом дня й рівномірний розподіл навчальної роботи протягом навчального тижня.

Для всіх видів навчальних занять академічна година встановлюється 45 хвилин, з перервою між заняттями в 10-15 хвилин [18, с. 32].

Навчальні дисципліни слід розміщати по днях так, щоб забезпечувалася правильна постановка викладання й рівномірний розподіл самостійної роботи студентів над навчальним матеріалом. У зв'язку із цим не слід займати навчальний день тільки лекціями, а також проводити в один день лекційні й практичні заняття по тому самому предмету.

При розподілі дисциплін по днях тижня необхідно чергувати дисципліни залежно від труднощів їх засвоєння, а також ураховувати доцільне чергування різних методів роботи.

Для забезпечення рівномірного, протягом семестру, розподіл занять, на які приділяється невелика кількість аудиторних годин, необхідно проводити їх через тиждень.

Лекції є однієї із самих складних навчальних навантажень, тому рекомендується обмежувати лекційне навантаження викладача 4 ч. у день (2 лекції).

Лекційні заняття повинні передувати практичним і лабораторним.

Лекції, як правило, необхідно включати в розклад на початку навчальних занять, причому не рекомендується вводити в розклад 4- х вартові лекції.

Останнє практичне або лабораторне заняття повинне проводиться після завершення читання лекцій по даній дисципліні.

Заняття в лабораторіях, особливо спеціальних, роботу в майстерень доцільно проводити переважно наприкінці дня, після теоретичних занять.

Перерви під час навчальних занять, що подовжують робочий день студента, за винятком обідніх, повинні бути виключені з розкладу.

Навчальний розклад - важливий документ, і складання його є обов'язком деканів факультетів [26, с. 53].

Контроль над виконанням розкладу аудиторних занять, здачі звітності, у тому числі розкладу консультацій, здійснюють декани факультетів, " Навчальне управління університету, проректора по навчальній роботі. Ціль такого контролю - перевірка своєчасності початку й закінчення занять. Про порушення розкладу занять (консультацій) повідомляється у встановленому порядку в Навчальне керування, проректорові по навчальній роботі.

Зміна затвердженого розкладу навчальних занять допускається в окремих випадках за поданням декана факультету з дозволу проректора по навчальній роботі. Зміна розкладу з ініціативи студентської групи здійснюється тільки за узгодженням з деканом факультету (або викладачами), у якого міняється час занять. Усі зміни проводяться протягом місяця з початку навчальних занять.

При складанні розкладу враховуються строки відпусток або відряджень. Як правило, уикладачі мають відпустку влітку, однак деякі з них, що працюють у прийомної комісії, не встигають використовувати свою відпустку до 1- го вересня.

У зв'язку з відпусткою на початку семестру початок проведення занять зрушується на більш пізній строк, а відрядження розбивають заняття на дві частини, при цьому одна частина занять повинна проводитися до відрядження, інша - після.

При тривалій відсутності викладача (відпустка, відрядження або хвороба) завідувачем кафедри складається графік заміни занять, копія якого передається в Навчальне керування. Графік заміни занять узгодиться з деканом факультету.

Разові заміни занять реєструються на кафедрі, у деканаті в журналі реєстрації змін у розкладі, за підписом зав. кафедрою або заст. декана. Заміна викладача на занятті без відповідного оформлення вважається порушенням трудової дисципліни [19, с. 32].

Тривалість екзаменаційних сесій для очно- заочної (вечернею) і заочної форм навчання встановлюється відповідно до додаткових відпусток, які надаються студентам, що навчаються у ВНЗ у відповідності зі ст.17 Федерального закону "Про вищий і послевузовском професійній освіті".

Розкладу курсових іспитів і заліків факультетів затверджуються проректорами по підпорядкованості й доводять до відомості викладачів і студентів не пізніше, чим за місяць до початку зачетно- екзаменаційної сесії.

Розклад для студентів очного й очно- заочного відділень складається з таким розрахунками, щоб на підготовку до екзаменів з кожної дисципліни було відведено не менш 3 днів.

Заліки й іспити слід проводити після завершення всіх лекційних і практичних занять.

Залік і захист курсової роботи повинні передувати екзамену з даної дисципліни.

У дні здачі іспитів студенти звільняються від занять і заліків (на очному й очно- заочному відділеннях).

Перед кожним іспитом проводиться консультація.

Проведення в один день консультацій і іспитів неприпустимо.

Кількість дисциплін, досліджуваних щодня, не повинне перевищувати трьох.

Для кожного курсу необхідно запланувати дні самостійних занять, коли студенти повинні готовити курсові роботи, писати реферати й займатися в бібліотеці. Як правило, такі дні призначаються студентам старших курсів.

У святкові дні заняття зі студентами не проводяться, і в розкладі це повинне бути передбачене [33, .с 76].

Усі ці нюанси необхідно враховувати при складанні розкладу навчальних занять, з метою якісної організації цього процесу.

Отже, у цьому параграфі були розглянуті загальні вистави про предметну область, дане його визначення. Так само був виявлений зв'язок між підрозділами внутрішньої структури університету з навчальним відділом утворююча потоки даних. Описані основні інструкції для складання розкладу. Таким чином, можна переступати до наступного параграфа даної глави, де буде розглянута ИС, класифікація й етапи проектування для подальшого розвитку проекту.

1.2. Огляд існуючих автоматизованих інформаційних систем формування розкладу

На сьогоднішній день існує ряд програм, що реалізують технологію "Розклад". Пропонується розглянути трохи, на прикладах відомих програм.

Система складання розкладів і обліку навантаження викладачів у ВНЗ "1С: Хронограф Розклад".

Програма "1С: Хронограф Розклад" призначена для автоматизації навчального планування й складання розкладу в окремих підрозділах професійних і вищих навчальних закладів, на різних комерційних і некомерційних навчальних курсах (комп'ютерних, вивчення іноземних мов, автошколах і т.д.), в установах додаткового утвору, підвищення кваліфікації й перепідготовки фахівців [24, с. 87].

"1С: Хронограф Розклад" надає можливість:

1. Підготувати дані про період навчання, з урахуванням специфіки організації навчальної діяльності конкретної освітньої установи на основі:

завдання навчального року;

розбивки навчального року на періоди навчального планування;

завдання навчальних періодів у рамках одного періоду навчального планування з метою деталізації строків викладання навчальних курсів, навчання навчальних груп і роботи викладачів;

автоматичного заповнення списку свят;

завдання неробочих тижнів і канікул;

автоматичного формування списку тижнів, що доводяться на обраний період навчального планування із вказівкою ознаки парності/непарності й кількості робочих днів.

2. Планувати навчальну діяльність усього навчального закладу або окремих підрозділів (факультетів, відділень, кафедр і т.п.), використовуючи:

- організацію й завдання структури рівнів навчання з можливістю диференціації їх на подуровни;

- формування списку навчальних курсів (предметів, дисциплін, тем і т.п.) із вказівкою строків їх викладання й можливістю перегляду інформації про навчальні тижні, що входять у заданий відрізок часу;

- створення списку навчальних груп із вказівкою чисельності, спеціалізації й рівня навчання;

- розподіл навчальних груп на необхідну кількість підгруп у рамках обраного навчального курсу;

- завдання ідентичних підгруп у рамках однієї групи по декільком навчальним курсам;

- створення потокових об'єднань груп/підгруп по обраному навчальному курсу;

- завдання навантаження навчальним групам/підгрупам на весь період навчального планування [40 ,с. 64];

- формування списку викладачів;

- розподіл годин навчального курсу в рамках навчальної групи/підгрупи по декільком викладачам на різні відрізки часу в границях заданого періоду навчального планування;

- автоматичне планування навантаження викладача на задану послідовність тижнів з можливістю ручного редагування.

3. Задати графіки роботи викладачів, що навчаються й кабінетів на основі:

призначення конкретних неробочих годин і днів для викладачів, навчальних груп і кабінетів;

- завдання максимально можливої кількості робочих днів викладача для кожного тижня поточного періоду навчального планування;

- автоматичного копіювання графіка роботи обраного викладача/групи/кабінету з одного тижня на задану послідовність тижнів.

4. Створити методично витримане розклад навчальних занять на основі:

- потижневого планування занять конкретного викладача безпосередньо при складанні розкладу;

- призначення занять викладачам у режимі обраної групи на поточний тиждень;

- можливості копіювання розкладу обраного викладача або навчальної групи з поточного тижня на задану послідовність тижнів у рамках навчального періоду;

- використання ефективного алгоритму "Попереднього розрахунків" при складанні "чорнових" варіантів розкладу;

- застосування інтерактивного "Автоматичного розрахунків" для складання "остаточного варіанта" розкладу на поточний тиждень для будь-якої кількості навчальних груп, з можливостями подальшого ручного редагування;

- призначення кабінетів (аудиторій) для проведення занять із обліком їх розташування, місткості й чисельності навчальних груп.

На основі наявної інформації програма "1С: Хронограф Розклад" дозволяє формувати всі необхідні звітні форми з можливістю їх налаштування по численних параметрах.

Програмний продукт (ПП)"1С: Хронограф Розклад" являє собою однокористувацьку конфігурацію системи програм "1С: Підприємство 7.7" і може використовуватися разом з будь-якими (крім базових) версіями програмних продуктів, що використовують компоненти "Бухгалтерський облік", "Розрахунки", "Оперативний облік" системи програм "1С: Підприємство 7.7" (наприклад, "1С: Бухгалтерія 7.7 Стандартна версія") [18, с. 54].

Продукт може функціонувати на базі мережної версії "1С: Підприємства 7.7", але якщо в установі утвору структурні підрозділи (факультети, відділення або кафедри) становлять розклад незалежно, те кожне з них повинне придбати окрему копію продукту.

Проаналізувавши всі сторони даної розробки, можна сказати, що вона передає цілісність такого питання, як складання розкладу. Але основним її недоліком є несумісність із відкритими платформами.1С: Хронограф - закрита система. У ній споконвічно не передбачається інтерфейс для інтеграції продуктів сторонніх розроблювачів, наприклад, неможливо прямо інтегрувати СУБД Хронографа й сервер web- додатків [16]. Так само її можна віднести до дорогих ПП і по деяких статистичних даних, вона складна в обігу.

Програма "Avtor" (АВТОРОЗКЛАД).

Система "АВТОРОЗКЛАД" призначена для швидкого, зручного і якісного складання розкладів занять і супроводу їх протягом усього навчального року.

Є вісім основних модифікацій програми для різних навчальних закладів для середніх загальноосвітніх шкіл, ліцеїв і гімназій, коледжів, технікумів і професійних училищ, училищ мистецтва й культури, для ВНЗ [22, .с 80].

AVTOR допомагає максимально полегшити й автоматизувати складна праця укладачів розкладу. Система допомагає легко будувати, корректировать і роздруковувати у вигляді зручних і наочних документів:

- розкладу занять класів (навчальних груп);
- розкладу викладачів;
- розклад зайнятості аудиторій (кабінетів);
- навчальні навантаження.

Час роботи програми залежить від розмірності навчального закладу й потужності комп'ютера. Повний розрахунки й оптимізація розкладу школи середнього розміру зі складними вихідними даними (40 класів, 80 викладачів, з них більш 10 сумісників; дві зміни; дефіцит аудиторій) іде близько 2-3 хвилин на комп'ютері типу Celeron-2000.

AVTOR дозволяє:

- будувати розклад без "вікон" у класів (навчальних груп);
- оптимізувати в розкладі "вікна" викладачів;
- ураховувати необхідний діапазон днів/годин для класів, для викладачів і для аудиторій;
- ураховувати характер роботи й побажання, як штатних співвикладачів, так і сумісників- погодинників;
- оптимально розміщати заняття по кабінетах (аудиторіям) з урахуванням особливостей класів, предметів, пріоритетів викладачів і місткості кабінетів;
- вводити розклад дзвінків;
- установлювати час переходу (переїзду) між навчальними корпусами;
- оптимізувати кількість переходів з кабінету в кабінет, і з корпуса в корпус;
- легко з'єднувати будь-які класи (навчальні груп) у потоки при проведенні будь-яких занять;

- розділяти класи (навчальні групи) при проведенні занять по іноземній мові, фізичній культурі, праці, інформатиці (і будь-яким іншим предметам) на будь-яка кількість підгруп (до десяти!);
- вводити комбіновані уроки для підгруп (типу "іноземний/інформатика") по будь-яких предметах;
- вводити (крім основних предметів) спецкурси й факультативи;
- оптимізувати рівномірність і трудомісткість розкладу;
- легко й швидко вводити й коректувати вихідні дані;
- мати будь-яка кількість варіантів розкладів;
- автоматично перетворювати розкладу при зміні бази даних;
- легко зберігати в архівах, копіювати й пересилати по E- Mail повні бази даних і варіанти розкладів (обсяг архіву повної бази розкладу середньої школи - 10-30 K, великого ВУЗА - 50-70K);
- швидко вносити будь-які необхідні коректування в розклад;
- знаходити заміни тимчасово відсутніх викладачів;
- автоматично контролювати розклад, крім будь-яких "накладок" і протиріччя;
- виводити розкладу у вигляді зручних і наочних документів: текстових, Word, HTML, а також файлів dbaseи книг Excel;
- виставляти готові розклади в локальній мережі й на Інтернет-сторінках для загального доступу.

Вона призначена для автоматизованого складання розкладу навчальних занять. Програма дозволяє оптимізувати використання аудиторного фонду, ураховувати особливості навчального процесу (лекції, практичні й лабораторні заняття, розподіл на підгрупи й організацію потокових занять, проведення занять по фізичній культурі і т.д.), ураховувати побажання викладачів, курси, розташування навчальних корпусів, тривалість занять і перерв. Крім розкладу очного навчання програма дозволяє становити розклад для заочного факультету,

коледжу й інших навчальних структур вузу, що навчаються по інших графіках навчального процесу.

Багатофункціональність даного ПП залишає гарне враження, розроблювачі віднесли до цієї програми з винною увагою й урахували багато проблем і нюансів при складанні навчального розкладу. Можна із упевненістю сказати про окупність даного ПО у вигляді працезатрат співвикладачів, але, проте , вона залишається дорогим продуктом, не кожний навчальний заклад може дозволити такі витрати [28, с. 64].

Такі програми дуже дорогі, зроблені для великих підприємств або складні в обігу. Так само для ведення таких програм потрібна підтримка, яка вимагає більших матеріальних витрат, висновок спеціальних договорів про супровід і необхідність містити цілий штат співвикладачів, які будуть займатися її підтримкою. Нами було ухвалене рішення створити оригінальну програму більш легку в експлуатації на платформі MS Access.

1.3 Опис технологій для вирішення завдання

Бурхливий розвиток обчислювальної техніки, потреба в ефективних засобах розробки програмного забезпечення привели до появи систем програмування, орієнтованих на "швидку розробку". В основі систем швидкої розробки або Rad- Систем (Rapid Application Development) лежить технологія візуального проектування й событийного програмування. Її суть полягає в тому, що середовище розробки бере на себе більшу рутинної роботи, залишаючи

програмістові роботу з конструювання вікон і створенню функцій обробки подій підвищуючи цим продуктивність роботи програміста.

Розроблювачі додатків баз даних висувають вимоги до інструментів, за допомогою яких такі додатки можна створювати, які в загальному виді можна сформулювати як: "швидкість, простота, ефективність, надійність" [25, с. 72].

Ринок програмних продуктів пропонує безліч середовищ для автоматизації програмування, самими популярними з яких є MS Access, Visual C++, Borland Delphi, Borland З++ Builder. У більшості випадків мови програмування не можна порівнювати між собою без зв'язку з розв'язуваними завданнями. Коротко розглянемо їхній особливості:

1) MS Access - реляційна СУБД корпорації Microsoft. Має широкий спектр функцій, включаючи зв'язані запити, сортування по різних полях, зв'язок із зовнішніми таблицями й базами даних. Завдяки вбудованій мові VBA, у самому Access можна писати додатка, що працюють із базами даних. В MS Access використовується мова програмування Visual Basic for Applications (VBA), істотним недоліком якого є неможливість створення виконуваних файлів (.EXE). У плані підтримки цілісності даних Access відповідає тільки моделям БД невеликої й середньої складності. Відносно захисту інформації й розмежування доступу Access не має надійних стандартних засобів [7 ,с. 35].

2) Visual C++ - хоча в назві й фігурує заявка на "візуальність", не дозволяє прямо створювати візуально-графічні компоненти. Їх доводиться створювати за допомогою класів, набираючи код на клавіатурі, що сприяє здійсненню безлічі помилок і збільшує час, витрачене не на розробку програм, а на створення інтерфейсу. У теж час Visual C++ дозволяє створювати програми мінімального розміру в порівнянні з Rad- Системами [30 ,с. 65].

3) Borland Delphi й Borland З++ Builder - наступний крок у розвитку Rad-Систем. Виглядають зовсім однаково. Основна відмінність полягає в тому, що Delphi базується мовою Pascal, а Builder мовою C++, могутнішому й багатому

по своїх можливостях. Borland C++ Builder досить стійко займає свою нішу, це обумовлене меншою вимогливістю до апаратних ресурсів при розробці додатків, більшою легкістю в освоєнні й застосуванні засобів системи для розробки додатків різному ступеня складності. Borland C++ Builder дозволяє створювати додатка за допомогою інструментальних програмних засобів, візуально підготовляти, а також безпосередньо писати Sql- Запити до баз даних.

Беручи до уваги усе вище викладене й зв'язку з тим, що мова Borland C++ Builder 5 для мене є більш знайомою мовою програмування, у якості середовища розробки вибираємо Borland З++ Builder версії 5.

Для реалізації поставленого завдання як середовища розробки додатки було обрано середовище програмування Embarcadero C++Builder 2010. C++ Builder - програмний продукт, інструмент швидкої розробки додатків (RAD), інтегроване середовище програмування (IDE), система, використовувана програмістами для розробки програмного забезпечення мовою C++. [3]

Програмний продукт C++ Builder споконвічно розроблявся компанією Borland Software, а потім її підрозділом Codegear, яке зараз належить компанії Embarcadero Technologies.

C++ Builder поєднує в собі комплекс об'єктних бібліотек (STL, VCL, CLX, MFC і ін.), компілятор, отладчик, редактор коду й багато інші компоненти.

C++ надзвичайно потужна мова, що містить засоби створення ефективних програм практично будь-якого призначення, від низкоуровневых утиліт і драйверів до складних програмних комплексів всілякого призначення. Зокрема :

- висока сумісність із мовою C, що дозволяє використовувати весь існуючий C- Код (код C може бути з мінімальними переробками скомпільований компілятором C++) [24 ,с. 84];
- підтримуються різні стилі й технології програмування, включаючи традиційне директивне програмування, ООП, узагальнене програмування;

- є можливість роботи на низькому рівні з пам'яттю, адресами, портами;
- можливість створення узагальнених контейнерів і алгоритмів для різних типів даних, їхня спеціалізація й обчислення на етапі компіляції, використовуючи шаблони.

Мова спроектована так, щоб дати програмістові максимальний контроль над усіма аспектами структури й порядку виконання програми. Жодна з язикових можливостей, що приводить до додаткових накладних витрат, не є обов'язковою для використання - при необхідності мова дозволяє забезпечити максимальну ефективність програми. [13, с. 43]

Традиційна технологія програмування складалася в умовах, коли основними споживачами програм були наукові установи, обчислювальні ресурси були обмежені. Криза програмного забезпечення привела до нової технології програмування, яка була названа структурним програмуванням.

Головна вимога, якому повинна задовольняти програма - працювати в повній відповідності з поставленим завданням і адекватно реагувати на будь-які дії користувача. Крім цього, програма повинна бути випущена точно на заявлений термін і допускати оперативне внесення необхідних змін і доповнень. Обсяг займаної пам'яті й ефективність алгоритмів при цьому, на жаль, відходять на другий план. Іншими словами, сучасні критерії якості програми - це, насамперед , надійність, а також можливість точно планувати виробництво програми і її супровід. Для досягнення цих цілей програми повинна мати просту структуру, бути, що добре читається, що й легко модифікується. Структурне програмування - це технологія створення програм, що дозволяє шляхом дотримання певних правил зменшити час розробки й кількість помилок, а також полегшити можливість модифікації програми. Структурний підхід охоплює всі стадії розробки проекту: специфікацію, проектування, властиво програмування й тестування.

Система Builder виробництва корпорації Borland повністю задовольняє цим вимогам. Інтегроване середовище Borland З++ Builder 5 забезпечує швидкість візуальної розробки, продуктивність повторно використовуваних компонентів у комбінації з міццю язикових засобів С++, удосконаленими інструментами й різномасштабними засобами доступу до баз даних.

До безсумнівних гідностей Borland Builder можна віднести наступні особливості:

1) Інтегроване середовище розробки поєднує редактор форм, інспектор об'єктів, палітру компонентів і повністю інтегровані редактор коду й отладчик - інструменти швидкої розробки програмних додатків, що забезпечують повний контроль над кодом і ресурсами [15 ,с. 88].

2) Професійні засоби мови С++ інтегровані у візуальне середовище розробки. Borland Builder надає швидкодіючий компілятор з мови Borland С++, ефективний инкрементальный завантажник і гнучкі засоби налагодження як на рівні вихідних інструкцій, так і на рівні ассемблерных команд - у розрахунках задовольнити високі вимоги програмістів- професіоналів.

3) Конструювання по способу "drag-and-drop " дозволяє створювати додаток простим перетаскуванням захоплених мишею візуальних компонентів з Палітри на форму додатка. Інспектор об'єктів надає можливість оперувати із властивостями й подіями компонент, автоматично створюючи заготовки функцій обробки подій, які наповнюються кодом і редагуються в процесі розробки.

4) Майстер інсталяції керує створенням уніфікованих дистрибутивних пакетів для розроблених додатків.

5) Відриті інструменти API можуть бути безпосередньо інтегровані у візуальне середовище системи. Є можливість підключення звичного текстового редактора або створення власного майстра для автоматизації виконання повторюваних процедур. Усі наведені вище гідності середовища С++ Builder промовляють на користь його використання при написанні пакета програм.

У якості використовуваних користувачем операційних систем були обрані системи сімейства Windows, оскільки саме вони найчастіше використовуються для роботи.

1.4. Платформа .NET Framework і мова програмування C++

Платформа .NET Framework - це технологія, яка підтримує створення та виконання XML-додатків і веб-служб. При розробці .NET Framework основними завданнями були:

- забезпечення узгодженого об'єктно-орієнтованого середовища програмування для локального збереження і виконання об'єктного коду, для виконання коду, розподіленого між різними серверами, пов'язаними один з одним мережею Internet;

- середовище забезпечення, яке мінімізує конфлікти при розгортанні програмного забезпечення та керуванням версіями;

- забезпечення середовища продуктивності, що забезпечує безпечне виконання коду, включаючи код, створений невідомим або ненадійним виробником третьої сторони;

- забезпечення середовища виконання коду, що виключає проблеми з виконанням середовищ виконання сценаріїв або інтерпретацією коду;

- забезпечення уніфікованих принципів розробки для різних типів програм, таких як програми Windows і веб-додатки [29, с. 88];

- розширене співробітництво на основі галузевих стандартів, які інтегрують код .NET Framework з будь-яким іншим кодом.

Платформа .NET Framework складається з загальних середовищ виконання (середовища CLR) і бібліотек класів .NET Framework. Основою для .NET Framework є середовище CLR. Середовище виконання може розглядатися як агент, який керує кодом під час виконання програми і надає основні послуги, такі

як управління пам'яттю, управління асинхронними потоками і віддалену взаємодію. У цьому випадку створюються умови суворої типізації та інших видів контролю правильності коду, які забезпечують безпеку і надійність при роботі. Фактично основним завданням середовища виконання є управління кодом. Код, який звертається до середовища виконання, називається керованим кодом, а код, який не має доступу до середовища виконання, називається некерованим кодом. Бібліотека класів - це повна об'єктно-орієнтована колекція, що дозволяє повторно використовувати типи, що використовуються для розробки програм - від звичайних програм, що виконуються в консолі за допомогою командного рядка або графічного інтерфейсу користувача (GUI), до програм, використовуючи найновішу ASP. Технології NET Core.

Платформа .NET Framework може розміщуватися на некерованих компонентах, які підключають середовище CLR до власних процесів і виконують виконання керованого коду, створюючи тим самим програмну оболонку, яка дозволяє використовувати як керований, так і некерований код. Платформа .NET Framework не тільки забезпечує кілька основних середовищ виконання, але й підтримує розробку основних середовищ, що пропонуються сторонніми постачальниками. Наприклад, ASP.NET Core розміщує середовище виконання й надає масштабоване середовище для керованого коду на стороні сервера. ASP.NET Core працює безпосередньо з середовищем виконання, щоб забезпечити виконання програм [36 ,с. 54].

Браузер Internet Explorer може слугувати прикладом некерованої програми, яка розміщує середовище виконання (як розширення типу MIME). Реалізація середовищ в Internet Explorer дозволяє реалізувати керовані компоненти або елементи керування

Windows Forms в документах HTML. Це розміщення середовища дозволяє виконувати керований мобільний код і скористатися ним, включаючи виконання в умовах неповної довіри та ізоляції файлів.

1.5. Постановка завдання

Метою даного дипломного проекту є підвищення ефективності складання розкладу за рахунок автоматизації його формування. Вирішенням даної проблеми є створення такої системи, яка дозволить полегшити роботу співвикладачів по веденню обліку робочого часу.

В основу конструювання зазначеної системи покладені наступні основні принципи:

- максимальна орієнтація на кінцевого користувача, що досягається створенням інструментальних засобів адаптації системи до рівня підготовки користувача, можливостей його навчання й самонавчання;
- проблемна орієнтація на вирішення певного класу завдань, об'єднаних загальною технологією обробки інформації, єдністю режимів роботи й експлуатації;
- формалізація професійних знань, тобто можливість за допомогою розроблювальної системи самостійно автоматизувати нові функції й вирішувати нові завдання в процесі нагромадження досвіду роботи із системою;
- модульність побудови, що забезпечує сполучення з іншими елементами системи обробки інформації, а також модифікацію й нарощування можливостей без переривання функціонування [24 ,с. 87];
- ергономічність, тобто створення для користувача комфортних умов праці й дружнього інтерфейсу спілкування із системою.

При наявності автоматизованої системи значно зручніше робити багато основних операцій, необхідні для ефективної й безперебійної роботи, (приклад, підсумки по кількості годин, відпрацьованих кожним викладачем, фіксувати інформацію про явки й неявки, так само тих, які занедужали, відсутні по нез'ясованих причинах або пішли у відпустку.) [9, с. 80]

В основі сучасних інформаційних систем лежить використання компонентів, які задовольняють таким властивостям, як відкритість, стандартизація, типізація рішень, масштабованість систем, комплексність підходу й тиражируемость.

Відкритість системи категорія не тільки технічна, але й економічна. Відкритість стандартів на системи автоматизації означає відсутність патентів або авторських прав на специфікацію стандарту і його розширення, відсутність ліцензійної плати за використання стандартів, широку доступність усіх фахівців до розробки, виробництва й використанню продукції в даному стандарті. Відкритість означає застосування відкритих стандартів, визначає гнучкість архітектури системи автоматизації, дружність користувацького інтерфейсу, можливість взаємодії з іншими системами за рахунок сумісності широкого спектра стандартизованих виробів і програм на різних рівнях. [36, с. 14]

Стандартизація припускає використання компонентів систем автоматизації, заснованих на існуючих стандартах для програмних і технічних рішень. Сучасні тенденції в області стандартизації такі, що системи автоматизації, побудовані на основі різних рішень, повинні інтегруватися в єдині системи й комплекси без серйозних додаткових розробок. На місце частнофирменных рішень повинні приходити відкриті міжнародні стандарти. Застосування стандартів при створенні системи автоматизації є показником її якості, гарантує користувачеві сучасний технічний рівень і наступність системи в процесі її подальшого розвитку й модернізації.

Типізація системних рішень повинна бути закладена в основу кожної створюваної системи автоматизації. Проходження цьому принципу дозволити мати доступний для огляду набір як компонентів, так і рішень, що дозволить заощаджувати на закупівлях компонентів і впровадженні систем.

Проходження принципу використання стандартів для компонентів і рішень дозволить створювати масштабовані, нарощувані системи автоматизації,

найбільш відповідні цілям і завданням, що коштують перед ними. Масштабованість дозволяє створювати й модернізувати системи автоматизації з мінімальними засобами, що забезпечують необхідні функції.

Комплексний підхід до систем автоматизації має на увазі не тільки традиційний (об'єктовий) підхід, але й цільовий, орієнтований на використання засобів автоматизації різного рівня для досягнення конкретних цілей (зниження енерговитрат, собівартості продукції, підвищення якості продукції й послуг).

РОЗДІЛ 2. ПРОЕКТУВАННЯ І РОЗРОБКА ПРОГРАМНОГО МОДУЛЮ ФОРМУВАННЯ РОЗКЛАДУ

2.1. Вимоги до програми

За результатами проведѐнного аналізу розроблювальна інформаційна система повинна буде містити в собі наступні основні й додаткові функції.

Основні функції:

1 Складання розкладу в автоматичному режимі;

2 Ручне редагування розкладів;

3 Відомостей вікон між парами до мінімуму;

4 Обмеження денного об’му занять.

Додаткові функції:

1 Урахування зайнятості аудиторії;

2 Урахування переваг викладачів;

3 Висновок інформації в EXEL, XLSX;

4 Урахування максимально припустимого кількості занять у день у студентів і викладачів;

5 Розбивка груп на підгрупи.

Для забезпечення оптимальності, що складається розкладу необхідне формування цільової функції для еѐ подальшого застосування в алгоритмах оптимізації. Дана функція повинна ґрунтуватися на вхідних, у проектовану інформаційну систему, потоках інформації, а також типі застосовуваного алгоритму оптимізації [14, с. 87].

Структурна схема інтерфейсу користувача (рис. 2.1) презентує забезпеченість програмного середовища функціональними можливостями процесу автоматизації.

Рис. 2.1. Структурна схема інтерфейсу користувача

У якості алгоритму загальної оптимізації розкладу передбачається застосування "Жадібного" алгоритму, що полягає в прийнятті локально оптимальних розв'язків на кожному етапі, допускаючи, що кінцевий розв'язок також виявиться оптимальним [29, с. 64].

Локально-оптимальний розв'язок на кожному етапі перебуває состязательным путѐм на основі оцінок, видаваних ИС тому або іншому розв'язку, залежно від ступеня його відповідності вимогам до елементів розкладу. Пошук локально-оптимального розв'язку відбувається через знаходження максимуму для елемента розкладу. При цьому, цільова функція расчѐта оптимального елемента може бути виражена наступною формулою:

$$A * T * D_s * D_t * (1 + P + W_s + W_t) \rightarrow max$$

де:

A - Вільні аудиторії (0 або 1);

T - Вільні викладачі (0 або 1);

D_s – Перехід між корпусами студентів (1, якщо перехід не повинен здійснюватися, 1 якщо це можливо);

D_t – Перехід між корпусами викладачів (1, якщо перехід не повинен здійснюватися, 1 якщо це можливо);

P – Відповідність елемента розкладу побажанням викладача (0 - не відповідає, 1 - відповідає зміна (ранкова / вечірня), 2 - відповідає вимозі до дня відпочинку, 3 - відповідає всім вимогам);

W_s – Виникнення вікна в студентів (-0.5 якщо виникає, 0.5 якщо не виникає);

W_t – Виникнення вікна у викладача (-0.5 якщо виникає, 0.5 якщо не виникає).

Вихідні дані до проекту: розробити програму для автоматизованого складання розкладу занять.

Потрібно реалізувати наступні функції:

- облік персоналу;
- облік викладачів і вихідних днів;
- ведення статистики;
- складання графіка ;
- генерація документів супутніх графіку і експорт їх у документ Excel.

Програма повинна мати можливість зберігати сгенерированные звіти, а також можливість видаляти їх на вимогу користувача [28 ,с. 75].

Переваги програмного засобу:

- зручний і дружній інтерфейс для введення й перегляду даних;
- можливість налаштування зовнішнього вигляду програми;
- інтерактивна допомога.

У цей час завдання поставлені перед програмою виконуються людиною з високим рівнем витрат тимчасових ресурсів.

Періодичність використання програми складе мінімум один раз на місяць, а так само при необхідності скласти розклад і супутні документи. Прямих аналогів для даної програми у вільному доступі знайдене не було.

Кожна система автоматизації повинна мати властивість відтворюваності й тиражируемости, що має прямий вплив на витрати по створенню систем і в остаточному підсумку на собівартість продукції. [37, с. 51]

Розроблювальне автоматизоване ПЗ повинне мати наступні якості:

- прийнятність у настроюванні;
- не повинна бути складної для роботи в ній;
- не маловажним є збереження друкованих форм у форматах Microsoft Excel і HTML;
- можливість розподіленої обробки;
- важливим є така властивість, як використання у вигляді окремих модулів;

- ведення централізованого обліку одночасно на декількох комп'ютерах;
- надійність і отказоустойчивость (наприклад, при відключенні електроенергії).

У програмі вихідна інформація, що й уводиться користувачем, зберігаються в базі даних DB, яка має наступну структуру:

- таблиця Workers містить стовпці:

1) ID - унікальний ідентифікатор;
2) Fam - прізвище працівника;
3) IO - ініціали працівника;
4) Dol - займана працівником посада;
5) D_k -код займаної посади;
6) RVS - кількість годин;
7) Sovm - маркер, що вказує чи є працівник на місці;
8) Dsovm - посада, що сполучається;
9) KDS - код посади, що сполучається;
10) RSVD - кількість годин для посади, що сполучається;
11) Razn - ступінь рівномірності розподілу ;
12) PC - дані про попередні розклади.

- таблиця Ras містить стовпці:

1) Names - коментар;
2) Date - дата.

- таблиця PRZ містить стовпці:

1) Names - назва свята;
2) OPR - рядок ідентифікатор дати проведення свята.

Після введення даних програма використовується для рішення свого основного завдання, а саме генерації розкладу . Результатом роботи програми

будуть складені розклади і супутні документи, а так само інша інформація, яка необхідна для роботи із програмою.

Для заборони безпосереднього доступу в базу даних сторонніми програмними засобами передбачений семизначний цифровий незмінний пароль 7681013. При необхідності пароль може бути змінений розробником у вихідному коді програми [10, с. 83].

Для реалізації системи обліку робочого часу співвикладачів потрібне створення програмного продукту й автоматизованої підсистеми на базі персонального комп'ютери. Однак принципи створення будь-яких автоматизованих систем повинні бути загальними:

- системність;
- гнучкість;
- стійкість;
- ефективність.

Розроблювальну систему слід розглядати як систему, структура якої визначається функціональним призначенням. Система пристосована до можливих перебудов, завдяки модульности побудови всіх підсистем і стандартизації їх елементів.

Принцип стійкості полягає в тому, що система повинна виконувати основні функції незалежно від впливу на неї внутрішніх і зовнішніх факторів, що обурюють. Це значить, що неполадки в окремих її частинах повинні бути, що легко усувається, а працездатність системи швидко відновлювана. Ефективність слід розглядати як інтегральний показник рівня реалізації наведених вище принципів, віднесеного до витрат на створення й експлуатацію системи.

Функціонування може дати бажаний ефект за умови правильного розподілу функцій і навантаження між людиною й машинними засобами обробки інформації, ядром якої є комп'ютер [12, с. 62].

У той же час до будь-якого спеціалізован, що використовує подібну систему можна пред'явити й ряд загальних вимог, які повинні забезпечуватися при його створенні, а саме:

- безпосередня наявність засобів обробки інформації;

- можливість роботи в діалоговому (інтерактивному) режимі;

- виконання основних вимог ергономіки: раціональний розподіл функцій між оператором, елементами комплексу й навколишнім середовищем, створення комфортних умов роботи, зручність конструкцій, облік психологічних факторів людину- оператора, привабливість форм і кольору елементів і ін.;

- досить висока продуктивність і надійність ПК;

- адекватне характеру розв'язуваних завдань програмне забезпечення;

- максимальний ступінь автоматизації рутинних процесів;

- оптимальні умови для самообслуговування фахівців.

Структура автоматизованої системи обліку робочого часу співвикладачів включає сукупність підсистем - технічної, інформаційної, програмної й організаційної.

Виходячи з умов поставленого завдання, можна виділити наступні вузлові пункти:

- проектування структурної схеми системи обліку робочого часу;
- розробка архітектури роботи автоматизованого програмного засобу;
- вибір платформи бази даних;
- вибір мови програмування.

При аналізі даної предметної області були виділені наступні інформаційні класи об'єктів і їх властивості [15 ,с. 83]:

- час роботи (Код роботи, код співробітника, підрозділ, дата, час);

- співробітник (Код співробітника, номер трудового договору, дата трудового договору, прізвище, ім'я, по батькові, дата народження, місце народження, підлога, № страхового свідчення, ідентифікаційний номер,

табельний номер, алфавіт, характер робіт, вид робіт, стан у шлюбі, паспорт серія, паспорт номер, дата видачі, ким виданий, індекс, місто, будинок, вулиця, квартира, дата реєстрації, телефон, додаткові відомості, номер наказу, дата звільнення);

- освіта (Код освіти, код співробітника, диплом, дата закінчення, спеціальність, кваліфікація);

- іноземна мова (Код мови, назва, ступінь знання);

- стаж (Код стажу, код співробітника, дні, місяці, роки);

- атестація (Код атестації, код співробітника, дата атестації, вирішення комісії, номер, дата, підстава);

- підвищення кваліфікації (Код кваліфікації, код співробітника, дата початку, дата закінчення, вид підвищення, найменування освітньої установи, найменування документа, дата документа, підстава);

- перепідготовка (Код перепідготовки, код співробітника, дата початку, дата закінчення, спеціальність, найменування документа, номер документа, дата документа, підстава) [36 ,с. 57];

- нагороди (Код нагороди, код співробітника, назва, найменування документа, номер документа, дата документа);

- відпустка (Код відпустки, код співробітника, назва, робота початок, робота кінець, кількість днів, дата початку, дата закінчення, підстава);

- соціальні пільги (Код пільги, код співробітника, найменування пільги, номер документа, дата документа, підстава).

Інфологична модель є проблемно - орієнтованою й системно - незалежної, тобто не залежної від конкретної СУБД, операційної системи й апаратного забезпечення ЕОМ.

Основною вимогою до инфологической моделі, що випливають із її значення, є вимога адекватного відображення предметної області. Инфологическая модель повинна бути не суперечливої. Вона є єдиним

інтегрованим описом предметної області й відбиває погляди й потреби всіх користувачів системи.

Не можна сказати, що в цей час існує який-небудь стандарт або хоча б загальноприйнятий спосіб побудови инфологической моделі. Для опису инфологической моделі використовуються як мови аналітичного (описового) типу, так і графічні засоби. Ми скористаємося надалі саме описовим способом відображення инфологической моделі. У предметній області в процесі її обстеження й аналізу виділяються класи об'єктів. Класом об'єктів називають сукупність об'єктів, що володіють однаковим набором властивостей. При відображенні в інформаційній системі кожний клас об'єктів представляється своїм ідентифікатором, який відрізняє один клас від іншого. Ідентифікатор повинен бути унікальним.

При описі предметної області відбивають зв'язки між об'єктом его властивостями, що й характеризують. Об'єкт може мати тільки одним значенням якоїсь властивості. Такі властивості називаються одиничними. Для інших властивостей можливе існування одночасне декількох значень в одного об'єкта. Така властивість називається множинним [29 ,с .75].

Крім зв'язку між об'єктом і його властивостями, у логічній моделі фіксуються зв'язки між об'єктами різних класів. Розрізняють зв'язки типу "один до одному" (1:1), "один до багатьом" (1:M), "багато до одному" (M:1), "багато до багатьом" (M:M). Іноді ці типи зв'язків називаються ступенем зв'язку.

2.2. Схеми алгоритмів основних методів

Проектування схеми бази даних повинне вирішувати завдання мінімізації дублювання даних і спрощення процедур їх обробки й відновлення.

У якості методів реалізації обраної тими розглядалися кілька варіантів:

- ієрархічна модель даних;

- мережна модель даних;

- реляційна модель даних.

Ієрархічна модель даних, заснована на понятті дерев, що полягають із вершин і ребер. Вершині дерева ставитися у відповідність сукупності атрибутів даних, що характеризують деякий об'єкт. Вершини й ребра дерева як би утворюють ієрархічну деревоподібну структуру, що полягає з n рівнів. Даний метод реалізації обраної тими не прийнятний, тому що операції в ІМД мають нелогічний позаписный характер. Механізми доступу до даних і переміщення за структурою даних у таких моделях досить складні й істотно опираються на концепцію потокового стану механізму доступу. Відношення " багато- до-багатьом" реалізується дуже складно, дає громіздку структуру й вимагає зберігання надлишкових даних, ієрархічна впорядкованість ускладнює операції видалення й додавання, доступ до будь- якої вершини можливий тільки через кореневу вершину, що збільшує година доступу, складно здійснити пошук і сортування даних [14 ,с. 93].

У мережній моделі даних більше уваги приділяється структуризації даних, чому розвитку її операційних можливостей. Набори відносин і структуру записів необхідно задавати наперед. Зміна структури бази даних звичайно веде до перебудови всієї бази даних. Звичайно, мережна модель має радий переваг: гнучкість, стандартизація, швидкодія.

У даній роботі використана реляційна модель даних. Реляційної називається база даних, у якій усі дані, доступні користувачеві, организованны у вигляді таблиць, усі операції над даними зводяться до операцій над цими таблицями. Дана модель вистави даних має радом незаперечних переваг:

1) доступ до даних вільний від двозначності;

2) автоматично підтримується цілісність даних;

3) перенос бази даних на інший комп'ютер не виявляє впливу на додаток;

4) зручно використовувати для зберігання будь- якої кількості будь- яких даних, особливо якщо дані однорідні;

5) повна незалежність даних;

6) можлива вибірка по заданих умовах;

7) легко здійснити пошук даних.

Програмний засіб складається із шістнадцяти взаємозалежних модулів: Unit1.cpp - модуль головного меню програми;

Unit2.cpp - модуль керування кадрами;

Unit3.cpp - модуль підключення до бази даних;

Unit4.cpp - модуль обліку свят і переносів;

Unit5.cpp - модуль налаштування;

Unit6.cpp - модуль статистики;

Unit7.cpp - модуль завантаження;

Unit8.cpp - модуль введення й редагування даних для позначення робочого дня;

Unit9.cpp - модуль введення й редагування даних для позначення святкового дня;

Unit10.cpp - модуль введення й редагування даних для обліку працівників;

Unit11.cpp - модуль генерації розкладу й переносу розкладу в документи;

Unit12.cpp - модуль вказівки місяця для створення розкладу;

Unit13.cpp - модуль експорту даних в MS Excel;

Unit14.cpp - модуль установки реквізитів;

Unit15.cpp - модуль вивідний відомості про програму;

Unit16.cpp - модуль очищення даних [25, с. 88].

У програмі реалізована система довідки для допомоги користувачеві в роботі із програмою. Вона являє собою CHM файл, і викликається через натискання на відповідну кнопку з написом "Довідка". У довідці розглянуто

десять питань дотичних роботи програми. Кожне питання розглянуте в окремому пункті й містить розгорнута й повна відповідь.

Також для зручності роботи користувача в деяких частинах програми були розташовані підказки, позначені за допомогою виділених областей, які подають коротку інформацію по поточному питанню.

Для обліку святкових днів, а так само днів, які є робітниками внаслідок переносу вихідних днів, у програмі буде розроблена система, призначена для обліку свят. Для вказівки вихідного дня є кілька режимів, які передбачають усі можливі варіанти визначення дати. Переноси робочих днів вказуються за допомогою безпосередньої дати.

Для обліку персоналу буде розроблена система, що дозволяє враховувати: посада, тривалість робочого дня індивідуально для кожного працівника, можливість сполучати дві професії на одному підприємстві [10 ,с. 64].

Вибір архітектури додатка головним чином залежить від реалізованого завдання. Програма для оператора чергової зміни успішно реалізується в невеликій мережі з файловим сервером.

Іншим важливим вирішенням, прийнятим при створенні інформаційних систем, є вибір інструментальних засобів розробки ІС.

До інструментів, орієнтованих на створення систем корпоративних масштабів, представляються наступні вимоги:

- великі інформаційні системи вимагають гнучкості інструмента, з погляду можливості нарощування функціональності повторно використовуваного програмного коду й реалізації нестандартних рішень (користувацький інтерфейс, межпрограммное взаємодія, інтеграція з успадкованими системами, доступ до системних ресурсів і т.п.). Повнота реалізації об'єктної моделі (необмежені можливості розширення ієрархії спадкування об'єктів) плюс можливість зміни функціональності об'єктів без створення нових об'єктних типів - класів (оброблювачі подій) [31 ,с .50];

- нейтральність стосовно використовуваних форматів БД і підтримка специфіки конкретних способів зберігання або доступу до даних, універсальний механізм доступу до даних;

- вимоги до продуктивності: компіляція, у випадку платформо- залежних рішень;

- відкритість середовища розробки, у плані можливостей інтеграції з іншими продуктами.

У якості сервера бази даних був обраний Interbase.Sql- Північ Borland Interbase призначений для зберігання й видачі більших обсягів даних при використанні архітектури клієнт- сервер в умовах одночасної роботи із БД безлічі клієнтських додатків. Для прискорення роботи клієнтських додатків з вилученої БД можуть бути визначені збережені процедури, які являють собою підпрограми, що ухвалюють, що й повертають параметри й здатні виконувати запити до БД, умовні розгалуження й циклічну обробку. Текст процедур зберігається на сервері.

В Interbase підтримується многоверсионная структура записів. При зміні запису якою-небудь транзакцією створюється нова версія запису, куди крім даних записуватися номер транзакції й покажчик на попередню версію запису. Стара версія запису позначається як змінена. Кожна транзакція, що стартує, працює з останньою версією запису, зміни для якої підтверджені. Таким чином, що паралельно працюють із БД транзакції завжди використовують різні версії записів, що дозволяє знімати блокування для клієнтських додатків, що одночасно працюють із тими самими даними в БД [28 ,с. 43].

Технічні характеристики сервера Interbase наведено в таблиці 2.1.

Таблиця 2.1

Технічні характеристики сервера Interbase

Характеристика	Значення
1	2

Максимальний розмір однієї БД	Рекомендується не вище 10Gb
Максимальне число таблиць в одній БД	65536
Максимальне число полів в одній таблиці	1000
Максимальне число записів в одній таблиці	Необмежено
Максимальна довжина запису	64 Kb (не вважаючи полів BLOB)
Максимальна довжина поля	32 Kb (крім полів BLOB)
промаксимальная довжина поля BLOB	Необмежено
Максимальне число індексів у БД	65536
Максимальне число полів в індексі	16
Максимальне число вкладеності Sql- Запиту	16
Максимальний розмір збереженої процедури або тригера	48 Kb

Interbase може посилати повідомлення клієнтським додаткам про настання якої-небудь події.

Модель автоматизованої системи (див. рис. 2.2) відображає свою структурну підпорядкованість всій автоматизованій системі ведення документообігу . Ця модель відображає функціональні зв'язки та можливість узгодженого використання загальних ресурсів всередині системи. Завдяки цій моделі стає зрозумілим, як окремі компоненти системи співпрацюють між собою та як вони спільно використовують загальні ресурси для забезпечення ефективності та гнучкості процесу ведення документообігу [40, с. 76] .

Рис. 2.2. Узагальнена модель автоматизованої системи формування розкладу

Після проведення аналізу предметної області було складено структурну схему даних, яка наочно відображає основні зв'язки між інформаційними таблицями (див. рис. 2.3). Ця схема допомагає уявити структуру та організацію даних в системі, вказує, як таблиці взаємодіють між собою та які відношення існують між різними елементами даних. Вона є важливим інструментом для розуміння та аналізу інформаційних потоків та забезпечує базу для подальшого розроблення та впровадження системи. Основні таблиці схеми даних розкладу

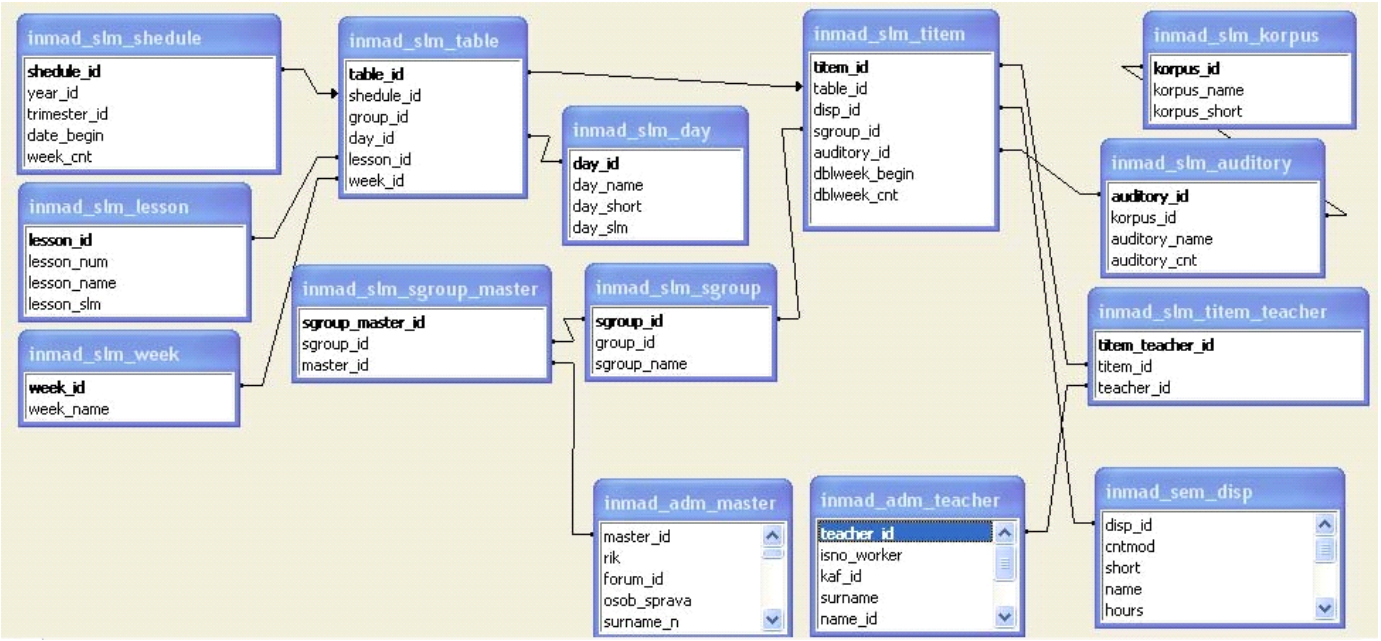

Рис. 2.3. Схема даних та взаємозв’язків між інформативними таблицями

Структурна схема даних (рис. 2.3) є цінним результатом аналізу предметної області. Вона допомагає зрозуміти взаємозв'язки та взаємодію між інформаційними таблицями, що в свою чергу сприяє ефективному проектуванню та реалізації системи. Завдяки цій схемі розробники та користувачі можуть більш чітко уявити структуру даних і їхню організацію, що сприяє кращому розумінню функціональності системи та покращує процес прийняття рішень. Структурна схема даних є потужним інструментом для вдосконалення інформаційних процесів та раціоналізації діяльності відповідно до вимог предметної області.

РОЗДІЛ 3. ОСОБЛИВОСТІ ВИКОРИСТАННЯ ПРОГРАМНОГО МОДУЛЮ

3.1. Розробка основних форм програмного продукту

У різних додатках статистичного аналізу одну із ключових позицій займають завдання кореляційного аналізу. У процесі рішення цих завдань виявляється наявність і характер взаємозв'язки величин, взаємозалежності величин при усуненні впливу сукупності інших або залежності однієї випадкової величини від групи величин, обчислюються оцінки коефіцієнтів і матриць парної, приватної й множинної кореляції, перевіряються різні статистичні гіпотези щодо параметрів багатомірного розподілу й коефіцієнтів кореляції. На підставі результатів кореляційного аналізу може робитися висновок про наявність і характер функціональної залежності або про перевагу для опису досліджуваного об'єкта регресійної моделі того або іншого виду.

В основі класичного апарата кореляційного аналізу лежить припущення про приналежність спостережуваного випадкового вектора багатомірному, нормальному закону. Базуючись на цьому, отримані граничні розподіли статистик, використовуваних у кореляційному аналізі. На практиці передумови класичного кореляційного аналізу виконуються далеко не завжди. Тому виникає питання про справедливість висновків, одержуваних на підставі класичного апарата, при порушенні основного припущення [13 ,с. 86].

Кореляція - це зв'язок, при якому певному значенню факторної ознаки відповідає лише середнє значення результативної ознаки.

Зв'язок між двома ознаками (результативним і факторним, або двома факторними) називається парною кореляцією.

Кореляційний аналіз має своїм завданням кількісне визначення тісноти зв'язки між ознаками (при парному зв'язку) і між результативним і безліччю

факторних ознак (при багатофакторному зв'язку). Тіснота зв'язку кількісно виражається величиною коефіцієнтів кореляції.

При вивчення процесів функціонування складних систем доводиться мати справа із цілим поруч одночасно діючих випадкових величин. У математичному аналізі залежність, наприклад, між двома величинами виражається поняттям функції:

y=f(x),

де кожному значенню однієї змінної відповідає тільки одне значення іншої. Така залежність зветься функціональної.

Набагато складніше обстоит справа з поняттям залежності випадкових величин. Як правило між випадковими величинами (випадковими факторами), що визначають процес функціонування складних систем, звичайно існує такий зв'язок, при якому зі зміною однієї величини, міняється розподіл іншої. Такий зв'язок називається стохастической, або імовірнісної. Існують різні показники, які характеризують ті або інші сторони стохастической зв'язки. Так лінійну залежність між випадковими величинами X і Y визначає коефіцієнт кореляції:

$$r = \frac{M[(X - Ax)(Y - Ay)]}{\delta x * \delta y},$$

де Ах і Ау - математичні очікування випадкових величин X і Y.

$\delta x, \delta y$ - среднеквадратические відхилення випадкових величин.

Лінійна імовірнісна залежність випадкових величин полягає в тому, що при зростання однієї випадкової величини інша має тенденцію зростати (або убувати) за лінійним законом. Якщо випадкові величини X і Y зв'язані строгою лінійною функціональною залежністю:

y = b0+b1x1

те коефіцієнт кореляції буде рівний r= +1 причому знак відповідає знак коефіцієнта b1 , якщо величина X і Y зв'язані довільної стохастической залежністю, то коефіцієнт кореляції буде змінятся в межах:

-1 <r< +1

Слід підкреслити, що для незалежних випадкових величин коефіцієнт кореляції буде дорівнює нулю.

Парний коефіцієнт кореляції й рівняння регресії. При прямолінійній формі зв'язку показник тісноти зв'язки двох ознак визначається по формулі лінійного коефіцієнта кореляції r:

$$r = \frac{\Sigma X * Y - \frac{\Sigma X * \Sigma Y}{n}}{\sqrt{[\Sigma X^2 - \frac{(\Sigma X)^2}{n}] * [\Sigma Y^2 - \frac{(\Sigma Y)^2}{n}]}}$$

де x - значення факторної ознаки;

y - значення результативної ознаки;

n - число пар даних.

r, обчислені по даним порівняно невеликої статистичної сукупності, можуть спотворюватися під дією випадкових причин. Тому необхідна перевірка їх сутності. Для оцінки значимості r застосовується t- критерій Стьюдента. При цьому визначається фактичне значення критерію tr:

$$t_r = r * \sqrt{\frac{n-2}{1-r^2}}.$$

Обчислене tr рівняється із критерієм tк, яке береться з таблиці значень t-стьюдента з урахуванням заданого рівня значення □ і числа ступенів волі k.

Якщо tr>tк, то величина коефіцієнта кореляції зізнається істотної. У рамках кореляційно- регресивного аналізу відбувається й вибір адекватного емпіричним даним рівняння регресії. При цьому недостатньо тільки якісного (логічного) аналізу. Хоча робочі гіпотези про можливу форму зв'язку формулювати можна. Наочне зображення аналізованих даних, тобто застосування графічного методу (шляхом побудови кореляційного поля крапок емпіричної лінії регресії), не дає узагальнену кількісну оцінку адекватності того або іншого рівняння зв'язки. Більш продуктивне використання критерію мінімальної залишкової дисперсії й показника середньої помилки апроксимації [26, .с 97]:

$$\varepsilon = \frac{1}{n} * \sum \frac{|Y_I - Y_{XI}|}{Y_I} * 100 ,$$

де Іуі - ухіі - модуль лінійних відхилень емпіричних і выравненых значень результативної ознаки. Оцінка параметрів рівнянь регресії здійснюється методом найменших квадратів. Сутність методу найменших квадратів полягає в знаходженні параметрів моделі (а0 і а1), при яких мінімізується сума квадратів відхилень емпіричних (фактичних) значень результативної ознаки від теоретичних. Для вираження прямолінійної форми залежності між X і Y застосовується формула:

YX = a0 +a1X,

$$\begin{cases} \mathbf{n a_0} + \mathbf{a_1} \Sigma X = \Sigma Y \\ \mathbf{a_0} \Sigma X + \mathbf{a_1} \Sigma X^2 = \Sigma XY \end{cases},$$

Для визначення параметрів рівняння на основі вимог методу найменших квадратів складається система нормальних рівнянь:

$$a_1 = \frac{n * \Sigma XY - \Sigma X * \Sigma Y}{n * \Sigma X^2 - \Sigma X * \Sigma X} .$$

Для рішення завдання обліку робочого часу системою застосовується спосіб визначників, що дозволяє зводити до мінімуму неточності округлення в розрахунках параметрів рівнянь регресії.

Інтерфейс програми є не тільки інтуїтивно зрозумілим, а так само зручним і візуально привабливим.

На додаток до реалізованих функцій, для підвищення зручності роботи із програмою була розроблена система підказок, яка активується по наведенню курсору миші на спеціальну область, відзначену зображенням

Після активації підказки буде виведене вікно утримуюче текст підказки. Після того як курсор покине відзначену, область підказка зникне [40 ,с. 61].

У програмі існує розвинена система попереджень, що передбачає різноманітні розв'язки, по розсуду користувача програми, більшої частини конфліктних ситуацій виявлених при розробці й тестуванні програми.

3.2. Опис архітектури програми

Після запуску программ, відбувається запит пароля. При введенні неправильного пароля програма завершує свою роботу.

Якщо пароль був уведений вірно, то встановлюється зв'язок з базою даних (БД) і виконується завантаження основного модуля, що організовує інтерфейс. На даному етапі виконання програми можлива обробка основних функцій роботи із БД:

- перегляд;
- введення початкових даних;
- коректування вибіркових даних;
- видалення;
- формування звітів.

Програмний засіб містить у собі програмні модулі (схема алгоритму представлена в додаток В), які організують роботу.

Модулі програми є незалежними, однак, функціонування модулів окремо від програмної системи не має ніякого змісту. Основне призначення модулів - взаємодія з базою даних [24, с. 87].

Ієрархічна структура програми, демонструє порядок взаємодії основних модулів програмної системи. Короткий опис призначення модулів представлене нижче.

Main - містить головну форму клієнтського додатка, що містить меню й визначальну подальші дії користувача.

DM - містить компоненти для взаємодії з базою даних. Усі звертання до бази даних здійснюються тільки через даний модуль.

Parol - даний модуль здійснює перевірку пароля.

U_Res_Copy - даний модуль здійснює резервне копіювання й відновлення бази даних.

Sotr - містить форму для перегляду, додавання нових, редагування й видалення існуючих у базі даних відомостей про облік робочого часу співвикладачів.

L_Cart - містить форму для формування особистої картки співвикладачів.

U_Arhive - містить форму для перегляду, редагування, відновлення й видалення даних про співвикладачів, що перебувають в архівній картотеці.

U_Filtr - здійснює в базі даних пошук співвикладачів за різними критеріями.

U_Grafik - висновок графічної інтерпретації статистичного аналізу.

U_Adress - містить допоміжну форму для заповнення адреси в особистій картці співробітника [30, .с 75].

U_Jazik - містить допоміжну форму для заповнення знання іноземної мови в особистій картці співробітника.

U_Semja - містить допоміжну форму для заповнення складу родини в особистій картці співробітника.

U_Staj - містить допоміжну форму для заповнення даних про стаж в особистій картці співробітника.

U_Obraz - містить допоміжну форму для заповнення даних про освіта в особистій картці співробітника.

Polzreg - модуль для реєстрації нового користувача й визначенні йому прав доступу.

Spravedit - модуль для редагування довідників.

U_Otchet - модуль для формування й підготовки до печатки вихідних документів (накази, особисті картки).

U_Prikaz - модуль для висновку звіту наказу (розпорядження) про приймання на роботу.

U_L_Cart - модуль для висновку звіту по особистій картці співробітника.

U_Sotr - модуль висновку звіту по всіх викладачах.

Help - модуль, що надає інформацію про програму й роботі з нею.

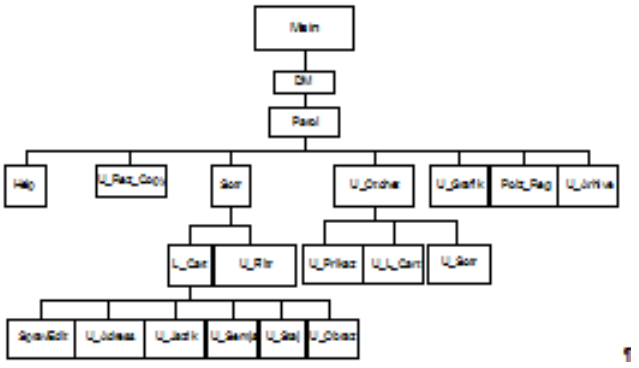

Рис. 3.1. Ієрархічна схема модулів

Більш докладний опис модулів і їх взаємодії.

1. Головний модуль - запускається при завантаженні програмного засобу. Представлений файлом main.pas. У цьому модулі активується головне меню програми, з якого доступні всі інші модулі. Після цього відразу активується модуль parol.pas, який відповідає за авторизацію користувача й визначенню його

прав доступу. Залежно від уведеного користувача й пароля доступними стають тільки певні пункти меню й, відповідно, певні модулі [28 ,с. 65].

2. Введення пароля - викликається з головного модуля. Представлений файлом parol.pas. При ініціалізації модуля на екрані з'являється форма "Пароль", у якій необхідно ввести ім'я користувача й пароль. При введенні невірного пароля програмний засіб завершує роботу.

3. Меню - відображається на головній формі програми. Залежно від того, який користувач працює із програмним засобом, активні ті або інші пункти меню. Через меню можна виконувати всі основні дії по роботі із програмою.

4. Резервне копіювання й відновлення - модуль відповідальний за збереження поточного стану бази даних в окремий файл і відновлення її з копії. При виклику даного модуля на екрані з'являється форма "Резервне копіювання" і форма "Відновлення з копії" при відновлення, у якій користувач може задати параметри збереження й відновлення бази даних на носій інформації. Файл U_Res_Copy.pas.

5. Співробітники - при виклику даного модуля на екрані з'являється форма "Співробітники". Користувач одержує можливість працювати з довідником "Співробітники". Є можливість додавати, видаляти й редагувати дані співвикладачів. Модуль викликається з головного меню "Список співвикладачів" і представлений модулем Sotr.pas. [41 ,с .34].

6. Особиста карта - при активізації команди "Додати", списку співвикладачів, активізується модуль особистої карти співвикладачів. При заповнення особистої карти активізуються допоміжні модулі наведені нижче.

7. "Архів" - при активації даного модуля відкривається форма архівної картотеки, у якій утримується інформація про звільнених співвикладачів. Є можливість пошуку співвикладачів за різними критеріями, відновлення співвикладачів з архіву без втрати інформації про нього, а так само фізичне видалення з бази співвикладачів п'яти літньої давнини.

3.3. Реалізація функціонального призначення програми

Меню запускається при активації програми, через нього виконуються всі інші модулі програмного засобу.

Працювати можна як з даними співвикладачів, що перебувають в основній картотеці, так і зданими співвикладачів з архівної картотеки. Користувач може затягати нових співвикладачів, видаляти дані про співробітника в архів, здійснювати пошук, коректування й перегляд його даних. З архіву користувач може видаляти дані фізично. Як правило, віддаляються дані зі строком зберігання в п'ять років.

У користувача є можливість одержання стандартних документів. Причому документи можна попередньо переглянути на екрані, а так само коректувати й виводити їх на печатку.

Немаловажну роль при роботі відіграють довідники, які дозволяють користувачеві швидко й коректно заповнювати й вибирати ту або іншу інформацію. У програмному засобі передбачена можливість роботи з довідниками. Користувач може при бажанні вводити нові, коректувати й видаляти наявні дані в довідниках [44 ,с. 65].

Тому що в програмному засобі використовується база даних те до вхідних даних відносяться таблиці бази даних, які вибираються з файлу бази даних VOEN_CHAST.gdb. Крім цього до вхідних даних відносяться дані, що вводяться із клавіатури. До вихідних даних відносяться звіти, що формуються в результаті обробки даних з таблиць бази даних. А також до вихідних даних відносяться відредаговані дані, які записуються в таблиці бази даних, у файл VOEN_CHAST.gdb.

Алгоритм програмного засобу розроблявся, виходячи із принципів модульности, ґрунтуючись на методі спадного проектування програм. Визначивши основні функції як визначені процеси, без реалізації їх коду, була

побудована архітектура програмного засобу. Такий спосіб побудови системи дає можливість побачити майбутній додаток у повному обсязі, хоча й не виконуючим ще своїх функцій. Це дозволяє на ранньому етапі проектування продемонструвати замовникові майбутню систему, оцінити й, можливо, скорегувати структуру всієї системи без переписування коду програми.

Загальний принцип побудови алгоритму заснований на самій суті графічного користувацького інтерфейсу, де дії користувача полягають у виборі пункту меню, а програма повинна викликати відповідну процедуру обробки.

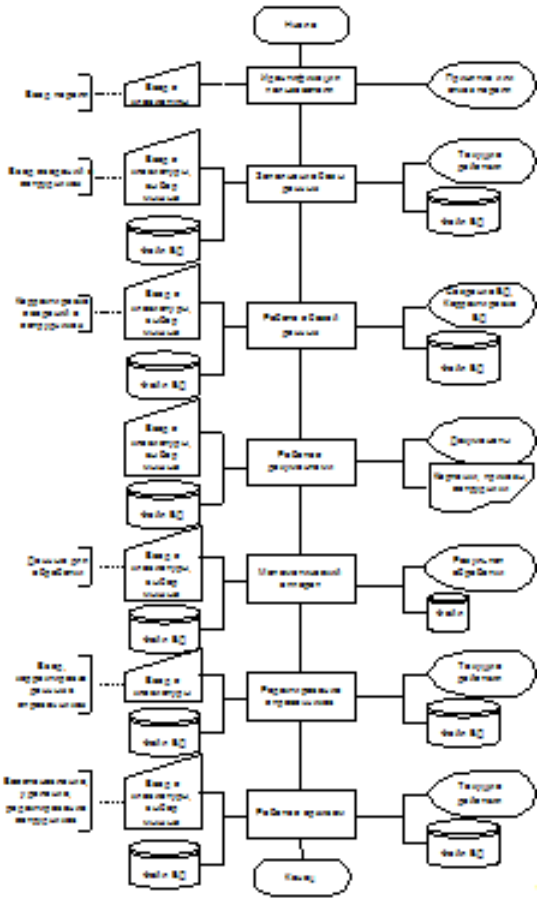

Рис. 3.2. Функціональна схема

Перша необхідна дія з програмним модулем - це введення пароля. При успішній ідентифікації користувача починається робота з програмним модулем. Подальші дії вибираються за допомогою головного меню програми. Основні дії продемонстровані на схемі алгоритму [17 ,с. 54].

До них відносяться:

- формування списку співвикладачів;
- робота з особистою карткою співробітника;
- статистика по обліку робочого часу співвикладачів частини;
- формування вихідних документів;
- редагування довідників;
- налаштування програми;
- виклик довідкової системи.

Введення відомостей про співвикладачів проводиться на основі вхідних документів.

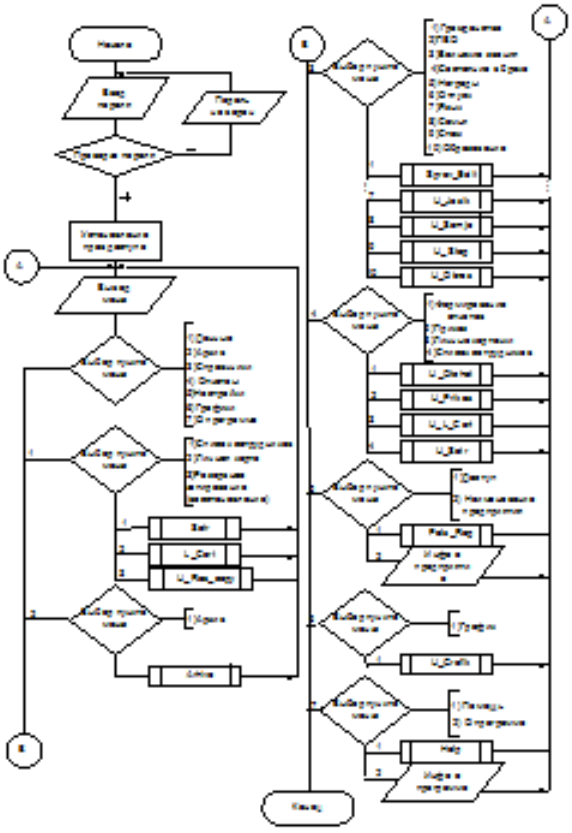

Рис. 3.3. Схема алгоритму

На головній сторінці цього середовища автоматизованої системи формування розкладу (див. рис. 3.4.) ми представляємо розроблені стилі презентації. Це включає в себе вигляд, оформлення та візуальне відображення, які були спеціально розроблені для системи. Ці стилі використовуються для створення зручного та привабливого інтерфейсу, що сприяє зручності

користування та полегшує сприйняття інформації користувачами. Презентовані стилі є результатом досліджень та розробки з метою забезпечити високу якість і візуальну привабливість інтерфейсу системи формування розкладу.

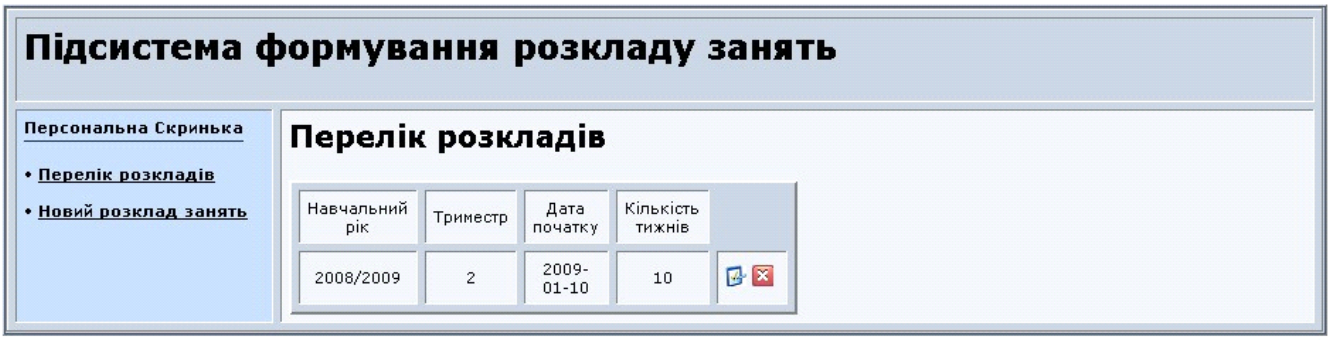

Рис. 3.4. Головна сторінка середовища системи автоматизованого формування розкладу

Після використання автоматизованої системи документообігу для створення розкладу, отримується готовий документ (див. рис. 5). Цей документ може бути подальше оптимізований відповідно до визначених критеріїв, щоб забезпечити оптимальні умови та задовольнити потреби користувачів. В системі також є можливість редагування цього документа, що дозволяє внести необхідні зміни або доповнення. Крім того, є можливість експортувати документ у форматі XLS (Excel) для зручного використання та обміну інформацією з іншими системами чи користувачами. Це забезпечує більшу гнучкість та доступність у роботі з розкладом [25, с. 76].

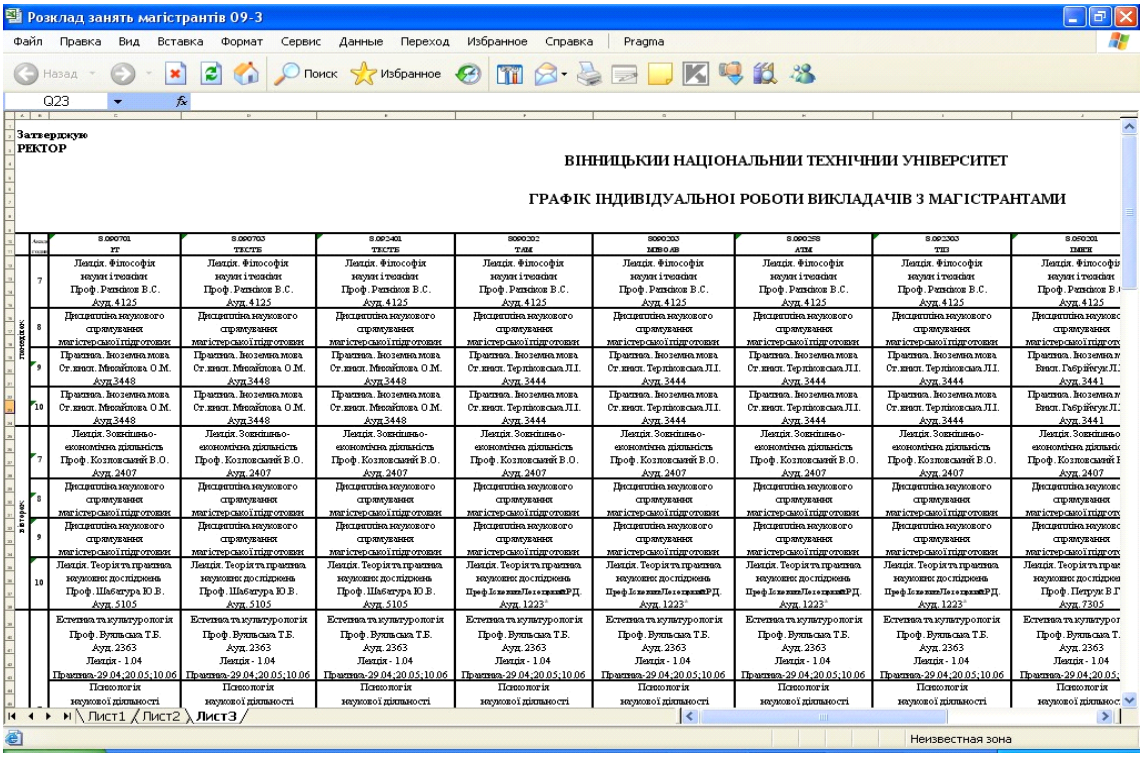

Рис. 3.5. Ілюстрація частини сформованого розкладу занять магістрантів

На рисунку 3.5. наведена блок-схема алгоритму роботи системи, яка ілюструє послідовність кроків для реалізації методу автоматизованого формування розкладу дисциплін магістерської підготовки. Ця блок-схема візуалізує процес обробки та організації даних для створення розкладу. Кожен блок на схемі представляє окрему дію або крок, який потрібно виконати відповідно до алгоритму. З'єднання стрілками вказують напрямок виконання та передачу даних між кроками. Блок-схема допомагає зрозуміти логіку та послідовність виконання операцій в системі формування розкладу та полегшує розуміння алгоритму для розробників та користувачів [19 ,с. 86].

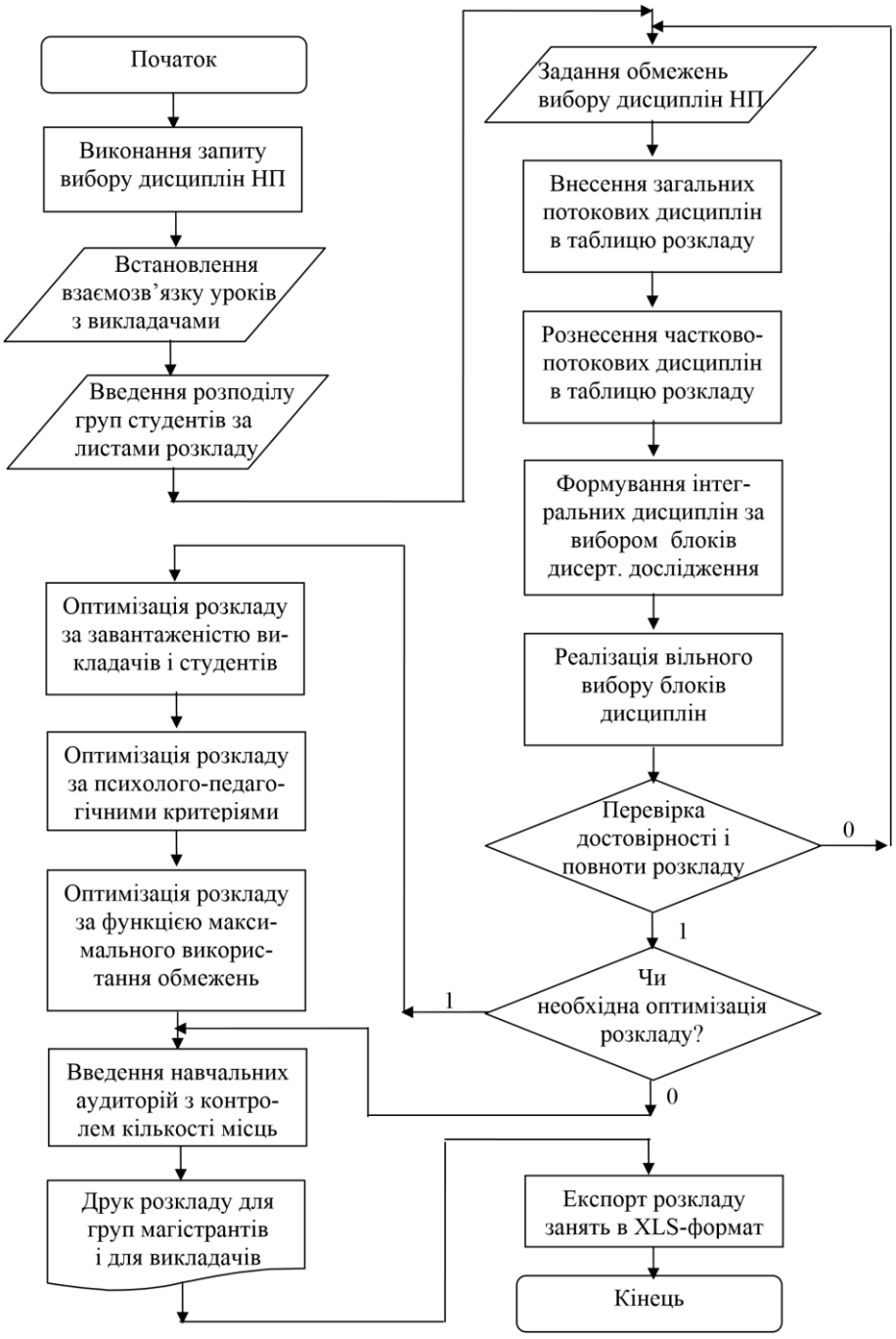

Рис. 3.6. Алгоритм реалізації методу формування розкладу магістратури

Для реалізації розроблених моделей автоматизованої системи та алгоритму формування розкладу ми використали можливості об'єктно-орієнтованої мови програмування PHP. Цей процес включав використання функцій та класів, що пропонує мова програмування PHP, щоб ефективно реалізувати та впровадити необхідний функціонал системи. Використання об'єктно-орієнтованого підходу дозволило нам створити гнучку та зручну систему для формування розкладу.

Користувачі та розробники можуть легко розширювати та змінювати функціонал системи шляхом використання розроблених класів та методів. Цей підхід сприяє зручності у розробці, підтримці та розширенні системи формування розкладу на основі вимог користувачів та змінюються потреб [30, с. 61].

Для формування списку співвикладачів викликається спеціальна форма (модуль Sotr), у якій вказуються критерії формування списку. Роботу з особистою карткою співробітника (модуль Lcart) можна також розділити на кілька важливих дій:

- введення, коректування відомостей;
- сортування даних;
- пошук даних за критеріями;
- збереження відомостей [36, с. 80].

У ході розробки програмного засобу був проведений статистичний аналіз, суть якого полягає у виявлення даних про середній вік співвикладачів.

Рис. 3.7. Вікова структура викладачів

На осі X представлена кількість викладач, а на осі Y середній вік викладачів.

А так само при розробці програмного засобу була реалізована функція кореляційного аналізу.

У цьому випадку розглядалася залежність кількості вступників на роботу співвикладачів від пори року.

У такий спосіб виникає необхідність відшукання двох функцій математичне очікування й дисперсії:

$$M[Y / X = x] = ax / y = \varphi(x),$$

$$D[Y / X = x] = \sigma y / x = \varphi(x)$$.

Коефіцієнт кореляції розраховується по формулі:

$$r = \frac{M[(X - Ax)(Y - Ay)]}{\sigma x * \sigma y}$$,

де: Ах і Ау - математичні очікування випадкових величин X і Y;

- среднеквадратические відхилення випадкових величин.

У такий спосіб коефіцієнт кореляції буде рівний:

r = 0,375562.

Що дозволяє зробити висновок, про те, що тіснота зв'язку даного аналізу помірна

Результат кореляційного аналізу представлено на малюнку 3.6.

Рис. 3.8. Результат кореляційного аналізу

Програмний продукт дає можливість вести особисту картку кожного працівника, становити звіти по працівниках частини, працювати з базою даних.

Установка програми можлива при наявність BDE Administrator ця утиліта, яка необхідна для доступу до баз даних з додатка й Interbase 6.0, необхідний для роботи з локальною або мережною базою даних [34, с. 55].

При наявність на комп'ютері вище перерахованого програмного забезпечення установку програми потрібно робити в наступному порядку:

1) скопіювати каталог Dip з CD-R на жорсткий диск;

2) завантажити утиліту BDE Administrator, далі необхідно створити псевдонім БД із іменем VOEN_CHAST і у властивостях псевдоніма встановити важливі для роботи параметри LANGDRIVER=Pdox_ANSI_Cyrilic,

USER_NAME=SYSDBA, далі в параметрі SERVER_NAME потрібно встановити значення шляху до файлу БД VOEN_CHAST.gdb, який перебуває в каталозі Dip\Base;

3) закрити утиліту BDE Administrator і на запит програми про збереження змін кликнути на кнопці "YES".

Програмний засіб: Автоматизоване робоче місце для формування розкладу занять призначене для функціонування під керуванням операційної системи WINDOWS на комп'ютерах з параметрами:

- Pentium 2, із частотою 133МГЦ і вище;
- обсяг оперативної пам'яті 16 Мб і більш;
- монітор типу SVGA, 800x600 крапок;
- від 50 Мб і більш дискового простору для нормальної конфігурації;
- наявність миші.

Алгоритм функції програми по автоматичній генерації розкладу виконано у вигляді декількох окремих класів, що займаються імпортуванням даних з бази даних, розміщенням предметів і забезпеченню виконання мінімальних умов оптимальності (відсутності накладень занять, як у студентів, так і викладачів; забезпечення використання аудиторного фонду по цільовому призначенню і т.д.) Алгоритм реалізовано у вигляді 5 наступних етапів:

1 Завантаження з бази даних усіх наявних груп і інформації про їхній навчальний план. Складання на їхній основі двовимірного масиву "гнізд" (, щопредставляють собою специфічний тип даних, який має наступні параметри необхідні для складання розклади: id групи, id предмета, id викладача, id аудиторії, номер корпуса, тип необхідної аудиторії для заняття, назва предмета й

тип проведеного заняття (теоретичне або практичне)), де кожний стовпець відповідає за групу, а рядок за відповідне одне заняття [17, с. 54].

2 Розміщення на основі наявного двовимірного масиву аудиторій з урахуванням параметра гнізда "тип необхідної аудиторії для заняття";

3 Нормалізація масиву - додавання порожніх "гнізд" або видалення наявних (з висновком попередження про це користувачеві) для доведення кількості рядків у кожному стовпці до 60 (максимальна кількість занять проведених за два тижні).

4 Розміщення занять (алгоритм представлено на малюнку 32);

5 Збереження сгенерированного, у вигляді двовимірного масиву "гнізд", розкладу в базу даних з можливістю наступної демонстрації й редагування.

У зазначеному вище алгоритмі, для визначення оптимальності вибору конкретного заняття під різні пари (першої, другий, третьої, четвѐртой і п'ятої) використовуються одмінні друг від друга критерії.

Особливо варто відзначити, що розміщення занять починається не з першої пари, а із другий, що дозволяє алгоритму органічно вписувати можливі переходи між корпусами (здійснювані під час великої перерви між другою й третьою парою) і уникнути виникнення "вікон" у розкладі, що виникають, приоритетно на цих парах.

Розглянемо критерії оптимальності вибору для кожної окремо взятої пари.

Критерії оптимальності других пар [22, .с 65]:

1 Відповідна до даного заняття аудиторія повинна бути вільна;

2 Відповідний до даного заняття викладач повинен бути вільний;

3 Відповідність особистим перевагам викладачів; Критерії оптимальності перших пар:

1 Наявність занять на другій парі;

2 Корпус аудиторії другої пари повинен відповідати корпусу аудиторії першої пари;

3 Відповідна до даного заняття аудиторія повинна бути вільна;

4 Відповідний до даного заняття викладач повинен бути вільний;

5 Відповідність особистим перевагам викладачів; Критерії оптимальності третіх пар:

1 Корпус аудиторії третьої пари переважно, але необов'язково повинен відповідати корпусу аудиторії другої пари;

2 Відповідна до даного заняття аудиторія повинна бути вільна;

3 Відповідний до даного заняття викладач повинен бути вільний;

4 Відповідність особистим перевагам викладачів; Критерії оптимальності четвѐртых пара:

1 Наявність занять на третій парі;

2 Корпус аудиторії четвѐртой пари повинен відповідати корпусу аудиторії третьої пари;

3 Відповідна до даного заняття аудиторія повинна бути вільна;

4 Відповідний до даного заняття викладач повинен бути вільний;

5 Відповідність особистим перевагам викладачів; Критерії оптимальності п'ятої пари:

1 Корпус аудиторії п'ятої пари повинен відповідати корпусу аудиторії третьої пари;

2 Відповідна до даного заняття аудиторія повинна бути вільна;

3 Відповідний до даного заняття викладач повинен бути вільний;

4 Відповідність особистим перевагам викладачів;

Для п'ятої пари краще виставляння порожнього гнізда заняття, якщо таке представляється можливим відповідно до існуючого навчального плану.

Перехід між корпусами передбачається виконувати винятково в проміжку між другою й третьою парою, однак кількість переходів, відповідно до критеріїв оптимальності третьої, четвѐртой і п'ятої пари повинне мінімізуватися.

3.4. Реалізації бази даних

Реалізація БД є критично важливим аспектом розробки програмного забезпечення, що вимагає зберігання й обробки інформації. Нижче представлена реалізація таблиць БД, використовуваний у програмі, що автоматизує процес складання розкладу в середовищі розробки із вказівкою прикладу заповнення таблиць.

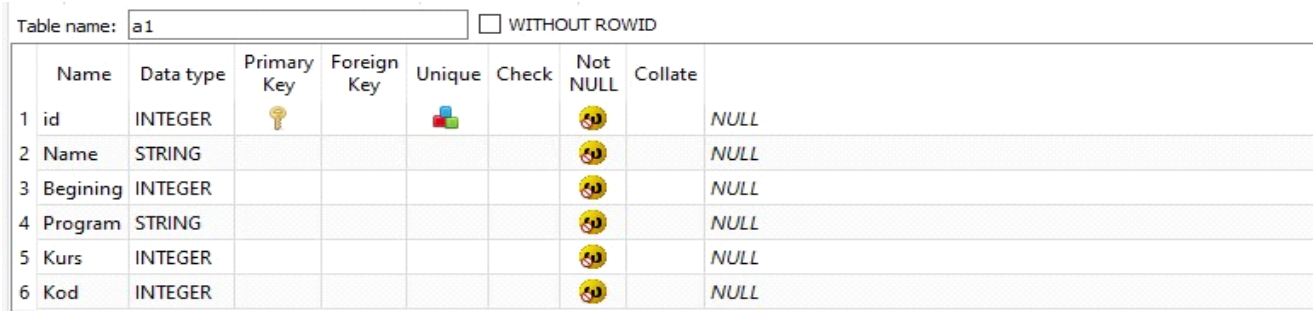
Table name: a1 WITHOUT ROWID

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	
1	id	INTEGER							NULL
2	Name	STRING							NULL
3	Begining	INTEGER							NULL
4	Program	STRING							NULL
5	Kurs	INTEGER							NULL
6	Kod	INTEGER							NULL

Рис. 3.9. Вид реалізованої таблиці "Групи" в Sqlitestudio

На малюнку 3.9. представлена реалізація таблиці БД "Групи". Нижче, у таблиці 3.2. зазначений приклад еѐ заповнення.

Таблиця 3.2.

Таблиця із прикладом заповнення БД (таблиця "Групи")

	Приклад заповнення таблиці "Груп"				
ID	NAME	BEGINING	PROGRAM	KURS	KOD
…	…	…	…	…	…
10	PI	1092016	Бакалаврат	1	90303

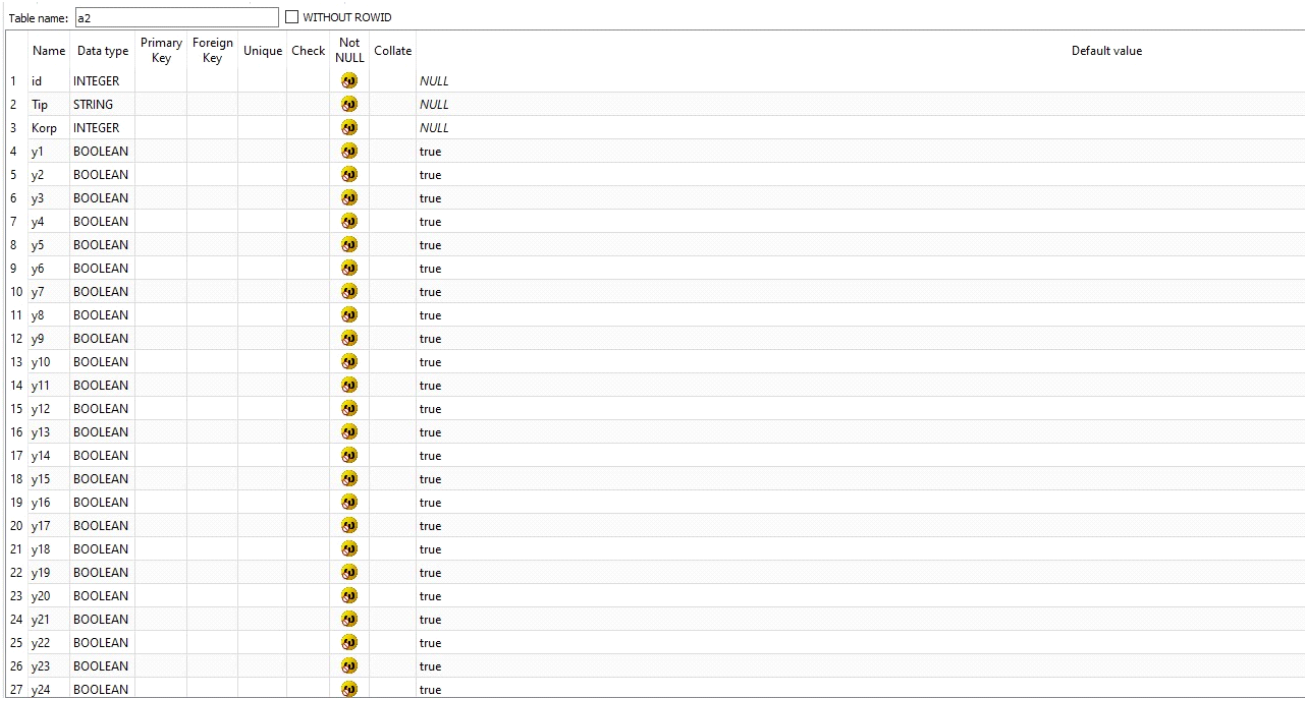

Table name: a2 WITHOUT ROWID

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Default value
1	id	INTEGER							NULL
2	Tip	STRING							NULL
3	Korp	INTEGER							NULL
4	y1	BOOLEAN							true
5	y2	BOOLEAN							true
6	y3	BOOLEAN							true
7	y4	BOOLEAN							true
8	y5	BOOLEAN							true
9	y6	BOOLEAN							true
10	y7	BOOLEAN							true
11	y8	BOOLEAN							true
12	y9	BOOLEAN							true
13	y10	BOOLEAN							true
14	y11	BOOLEAN							true
15	y12	BOOLEAN							true
16	y13	BOOLEAN							true
17	y14	BOOLEAN							true
18	y15	BOOLEAN							true
19	y16	BOOLEAN							true
20	y17	BOOLEAN							true
21	y18	BOOLEAN							true
22	y19	BOOLEAN							true
23	y20	BOOLEAN							true
24	y21	BOOLEAN							true
25	y22	BOOLEAN							true
26	y23	BOOLEAN							true
27	y24	BOOLEAN							true

Рис. 3.10. Вид реалізованої таблиці "Аудиторії" в Sqlitestudio

На малюнку 3.10. представлена реалізація таблиці БД "Аудиторії" [8, .с 9]. Нижче, у таблиці 3.3. зазначений приклад еѐ заповнення.

Таблиця 3.3.

Таблиця із прикладом заповнення БД (таблиця "Групи")

	Приклад заповнення таблиці "Аудитори		и"
ID	Tip	Korp	y1-y60
…	…	…	…
101	n	1	0

Рис. 3.11. Вид реалізованої таблиці "Викладачі" в Sqlitestudio

На малюнку 11 представлена реалізація таблиці БД "Аудиторії".

Нижче, у таблиці 3.4. зазначений приклад еè заповнення.

Таблиця 3.4.

Таблиця із прикладом заповнення БД (таблиця "Викладачі")

		Приклад заповнення таблиці "Викладачі"									
ID	FIO	Smena	FreeDay	Status	p1	p2	p3	p4	p5	Priority	y1-y60
…	…	…	…	…	…	…	…	…	…	…	…
1	Петров Н.Н	0	0	Профессор	1	2	3	4	5	75	0

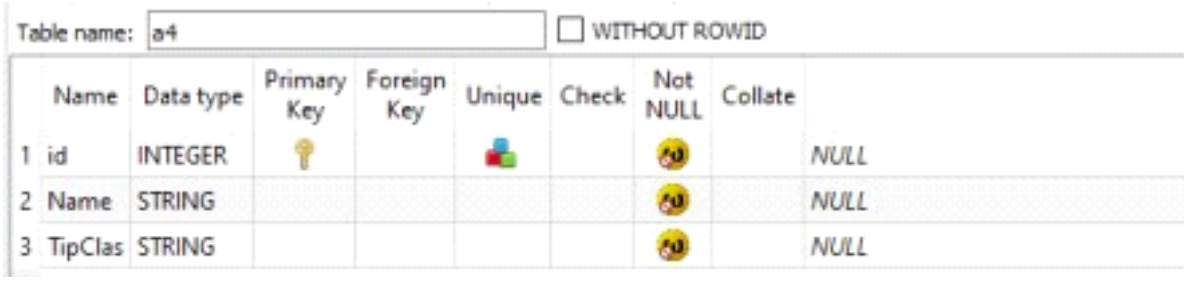

Рис. 3.12. Вид реалізованої таблиці "Предмети" в Sqlitestudio

На малюнку 3.12. представлена реалізація таблиці БД "Предмети". Нижче, у таблиці 3.5. зазначений приклад еѐ заповнення.

Таблиця 3.5.

Таблиця із прикладом заповнення БД (таблиця " Предмети")

Приклад	заповнення таблиці "Предмети"	
ID	NAME	TipClas
…	…	…
0	Математика	n

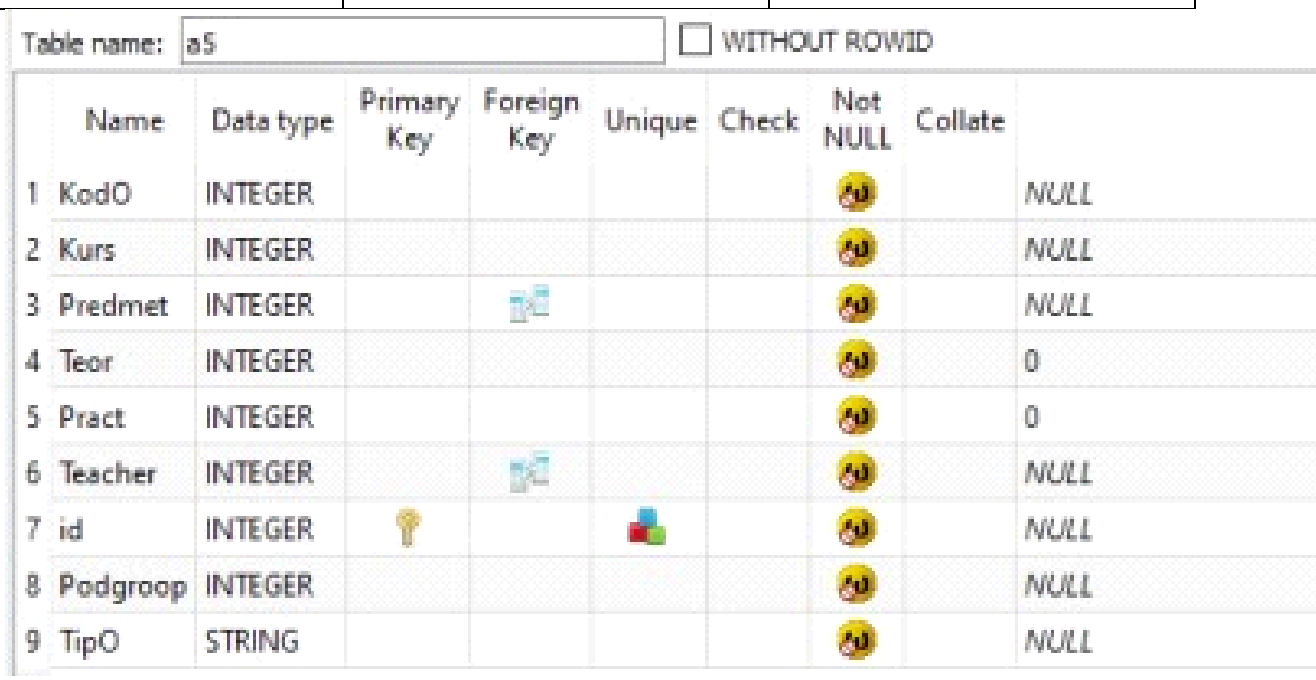

Рис. 3.13 Вид реалізованої таблиці "Навчальний план" в Sqlitestudio

На малюнку 3.13. представлена реалізація таблиці БД "Предмети". Нижче, у таблиці 3.6. зазначений приклад еѐ заповнення.

Таблиця 3.6.

Таблиця із прикладом заповнення БД (таблиця "Навчальний план")

		Приклад заповнення таблиці "Навчальний план"						
KodO	Kurs	Predmet	Teor	Pract	Teacher	id	Podgroop	TipO
…	…	…	…	…	…	…	…	…

90303	4	1	150	0	3	11	2	Магіструра

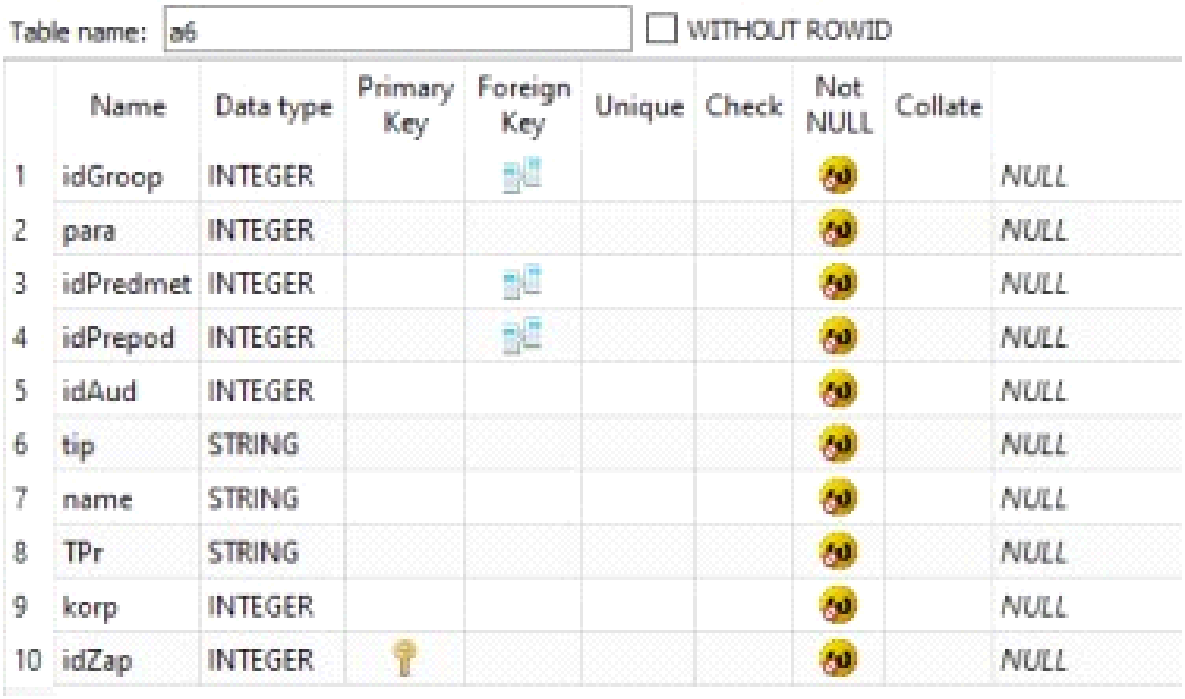

Рис. 3.14. Вид реалізованої таблиці "Розклад" в Sqlitestudio

На малюнку 3.14. представлена реалізація таблиці БД "Предмети". Нижче, у таблиці 3.7. зазначений приклад еѐ заповнення.

Таблиця 3.7.

Таблиця із прикладом заповнення БД (таблиця "Розклад")

		Приклад заповнення таблиці "Розклад"							
idGroop	para	idPredmet	idPrepod	IdAud	tip	name	TPr	korp	idZap
…	…	…	….	…	…	…	…	…	…
1	3	1	2	101			teor	1	100

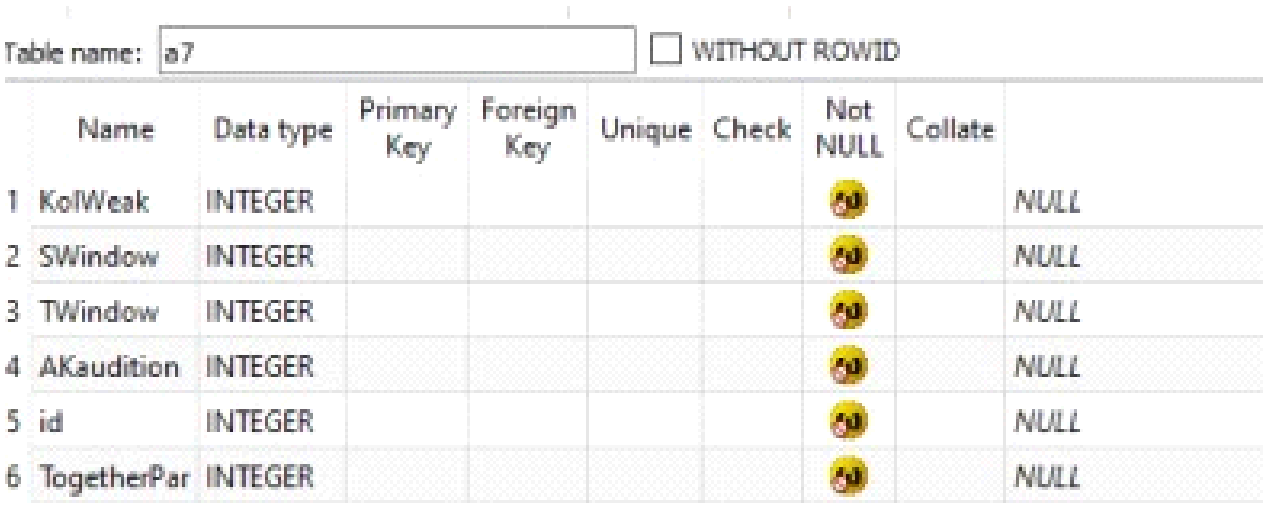
Table name: a7 WITHOUT ROWID

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	
1	KolWeak	INTEGER							NULL
2	SWindow	INTEGER							NULL
3	TWindow	INTEGER							NULL
4	AKaudition	INTEGER							NULL
5	id	INTEGER							NULL
6	TogetherPar	INTEGER							NULL

Рис. 3.15. Вид реалізованої таблиці "Налаштування" в Sqlitestudio

На малюнку 3.15. представлена реалізація таблиці БД "Предмети". Нижче, у таблиці 3.8. зазначений приклад еѐ заповнення.

Таблиця 3.8.

Таблиця із прикладом заповнення БД (таблиця "Налаштування")

	Приклад заповнення таблиці "Налаштування"				
KolWeak	SWindow	TWindow	AKaudition	id	TogetherPar
…	…	…	…	…	…
25	10	10	50	1	15

3.5. Реалізації інтерфейса програми

При запуску програми користувач побачить вікно вітання з індикатором завантаження. При першому запуску програми після установки на екрані з'являється нагадування, що програма запущена в перший раз і вимагає налаштування.

Після того, як завантаження програми буде завершено, користувач побачить головне меню програми за допомогою якого можливий перехід до більшості доступних функцій у програмі [18 ,с. 65].

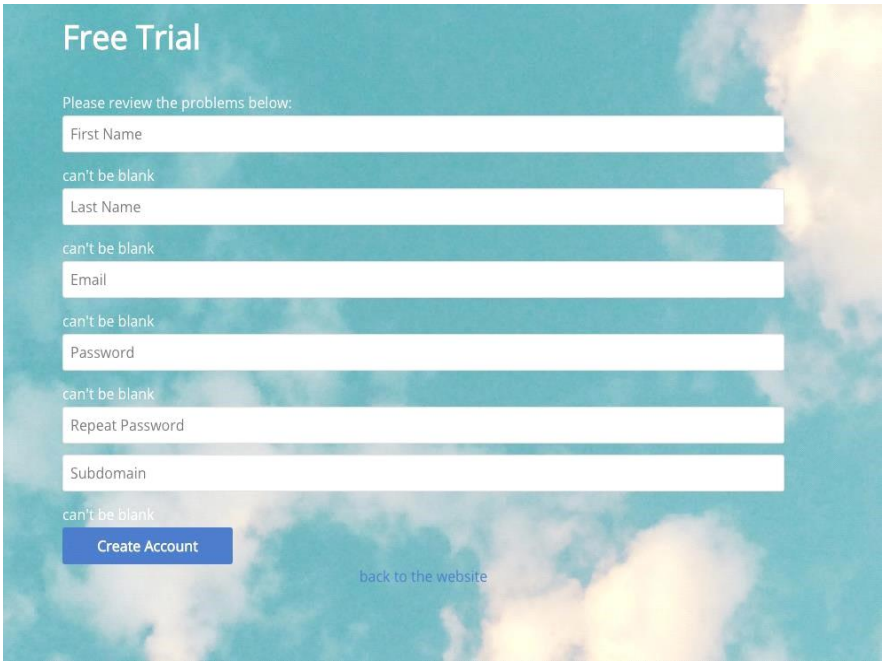

Рис. 3.16. Головне меню програми

Меню містить вісім кнопок:

- кнопка "Вихід" - призначена для завершення роботи із програмою;
- кнопка "Довідка" - призначена для одержання довідки по програмі й методах роботи із програмою;
- кнопка "Про програму" - викликає вікно утримуюче інформацію версії, розроблювачах і іншої інформації про програму;

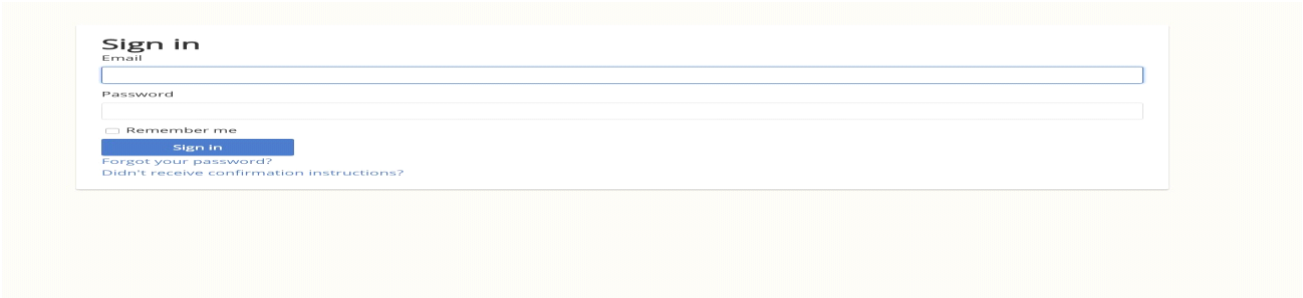

Рис. 3.17. Аутентифікація в програмі

- кнопка "Статистики" - призначена для відображення вікна статистик дотичних різних аспектів роботи програми;
- кнопка "Налаштування" - використовується для відображення вікна настроювань. Вікно настроювань використовується для установок настроювань необхідних для роботи програми. Зокрема це: визначення необхідного кількості

працівників для першої й другої зміни, налаштування зовнішнього вигляд програми, переглянути архів документів (відкриття відбудеться в окремому вікні стандартного провідника Windows, автоматично очистити архів документів і обнулить дані дотичні роботи програми [36, с. 53];

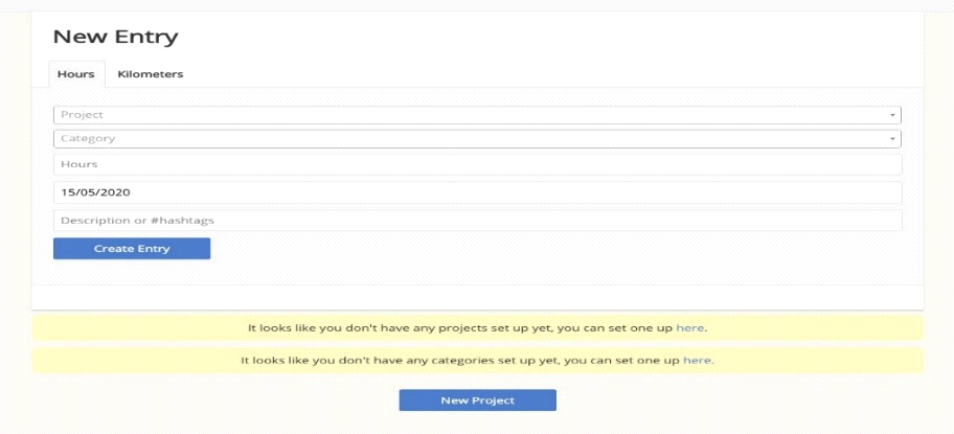

Рис. 3.18. Налаштування

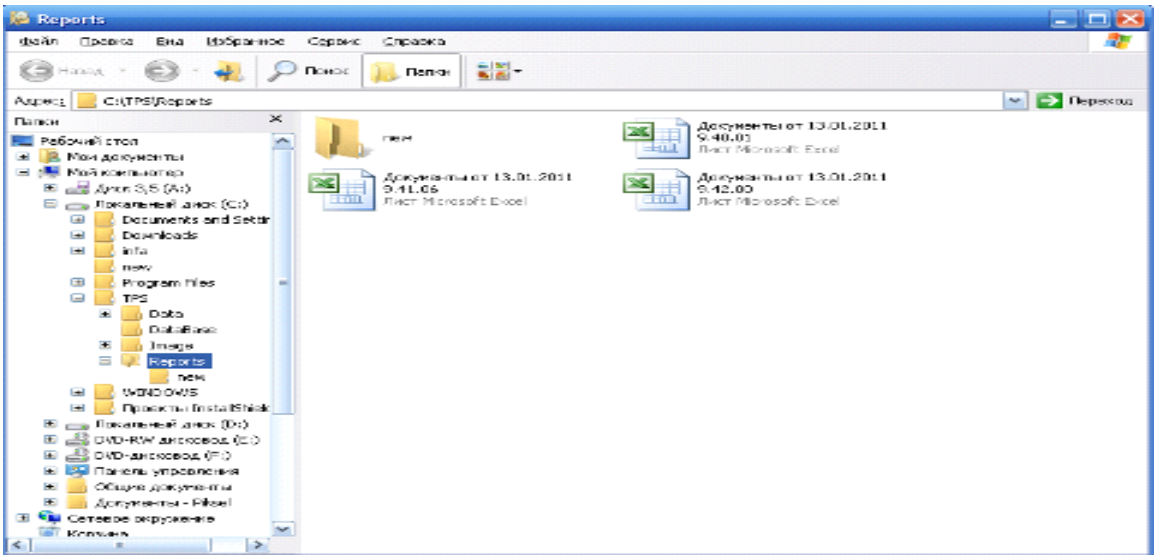

Рис. 3.19 Архів документів

Кнопка "Робочий календар" - призначена для виклику вікна, що відображає виробничий календар, який має режим розширення, при активації якого з'являється додаткова панель, що містить список святкових днів і список робочих днів святкових днів, що з'явилися внаслідок переносів. Так само додаткова частина дозволяє додавати, редагувати й видаляти дати з першого й другого списків. Редагування можна викликати натисканням на відповідну кнопку або за

допомогою подвійного кличу на запис потребуючу зміни. Видалення можна викликати натисканням на відповідну кнопку або за допомогою натискання на клавішу "Delete". Кнопка "Керування кадрами" - призначена для керування записами дотичних персоналу підприємства, записи можна додавати, редагувати, видаляти. Так само як і при роботі з датами виробничого календаря редагування записи можна викликати за допомогою подвійного кличу на ній, а видалення натисканням кнопки "Delete". Додавання й редагування записів відбувається у вікні [27 ,с. 80].

Кнопка "Робота з документами" - призначена для відкриття вікон призначених для генерації розкладу і його експорту в MS Excel.

Після натискання кнопки "Робота з документами" з'явиться вікно із пропозицією вибрати місяць і рік, на підставі яких, буде надалі створений розклад .

Ця підсистема є основною складовою частиною програмної системи. Після авторизації оператора в системі, вікно модуля завантажується негайно. Воно містить чотири вкладки: "Аудиторії", "Викладачі", "Сформувати розклад" та "Редагувати розклад".

На вкладці "Аудиторії" можна переглянути зайнятість конкретної аудиторії або знайти всі вільні аудиторії певного типу з необхідним обладнанням. Інформацію про аудиторії можна переглянути на основі одного дня або на весь тиждень.Вкладка "Викладачі" надає інформацію про графік роботи викладача. Тут також можна фільтрувати інформацію за певною дисципліною, яку викладач веде.Вкладка "Сформувати розклад" призначена безпосередньо для створення розкладу навчального закладу. Перед початком формування розкладу потрібно вибрати факультет, спеціальність, курс та семестр, для яких створюється розклад. Після цього необхідно натиснути кнопку "Застосувати" та перейти до додавання

навчальних занять до розкладу. Також передбачена можливість редагування розкладу [23, с. 76].

Оператор має доступ до вищезгаданих модулів з основного вікна програми. За допомогою меню "Адміністрування" можна викликати всі адміністративні модулі. Це забезпечує зручний та комплексний доступ до функціоналу системи для оператора.

Рис. 3.20. Фрагмент вікна підсистеми формування розкладу

Для поширення розкладу існує два можливих способи. Спосіб 1: Розповсюдження у форматі Excel для внутрішнього використання (розсилка по деканатах, факультетах та кафедрах). Для створення такого файлу був розроблений програмний модуль, який дозволяє згенерувати розклад для студентів або викладачів.

Для складання розкладу студентів (див. рис. 6) використовувалися наступні критерії відбору:

- Факультет;
- Спеціальність (всі спеціальності факультету або конкретна обрана спеціальність);
- Група (всі групи факультету/спеціальності або конкретна обрана група).

Ці критерії дозволяють точно відфільтрувати необхідну інформацію та створити розклад, який відповідає потребам студентів. Після складання розкладу він може бути зручно розповсюджений серед відповідних деканатів, факультетів та кафедр у форматі Excel [19, с. 51].

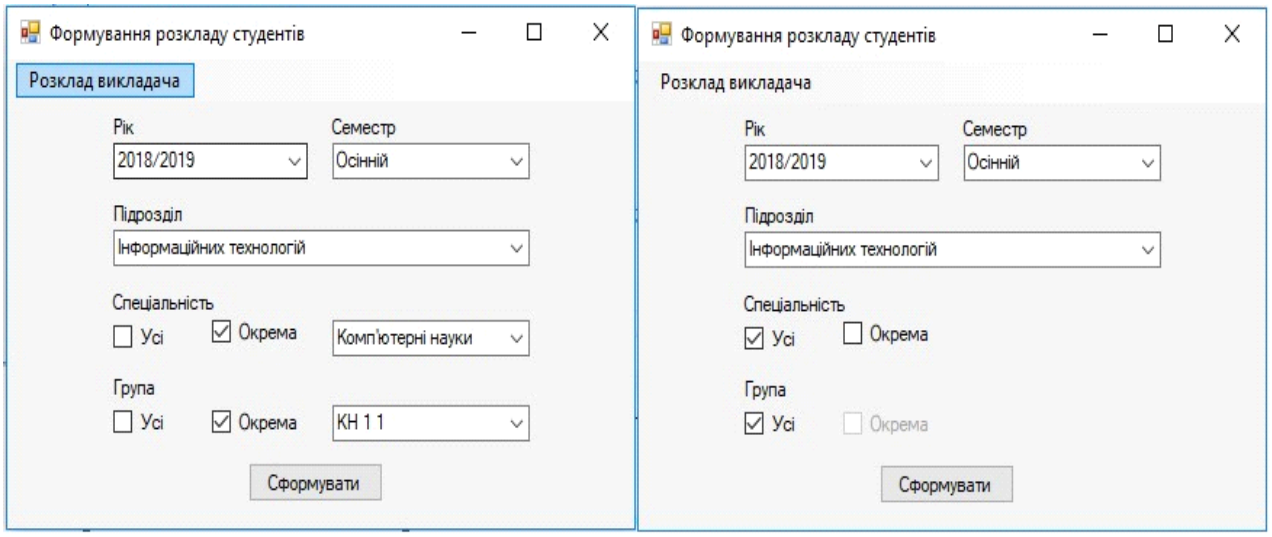

Рис. 3.21. Варіанти вибору критеріїв для формування розкладу студентів

Для складання розкладу викладачів (див. рис. 3.22.) використовувалися такі критерії відбору:

- Кафедра;
- Викладачі (всі викладачі кафедри або конкретний обраний викладач).

Ці критерії дозволяють точно відфільтрувати необхідну інформацію та створити розклад, який відповідає потребам викладачів. Використовуючи ці критерії, система може згенерувати розклад занять для конкретної кафедри та обраного викладача. Це сприяє зручності та ефективності управління розкладом викладачів, дозволяючи точно відображати їхні заняття та графік роботи.

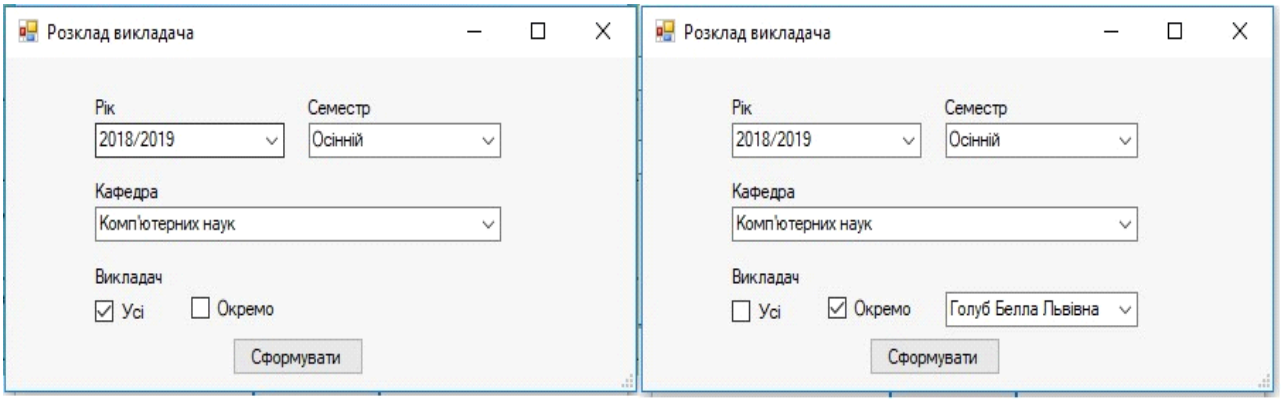

Рис. 3.22. Варіанти вибору критеріїв для формування розкладу викладачів

Сформований файл з розкладом для студентів факультету інформаційних технологій представлений на рисунку 3.23 Цей файл містить інформацію про

розклад занять для студентів цього факультету, що дозволяє їм точно знати час і місце проведення кожного заняття. Розклад може бути представлений у зручному форматі, який дозволяє студентам швидко знайти необхідну інформацію та планувати свій час відповідно [42, с. 67].

Розклад для факультету Інформаційних технологій на 2018/2019 н.р. Весняний семестр

напрям		КН							
курс		1		2		3		4	
група		1	2	1	2	1	2	1	
Понеділок	1	Вища мат. ауд.317 к.11			Числ.мет. ауд.107 к.11	Стат.мет. ауд.231 к.15	Стат.мет. ауд.231 к.15		
	2	Програмування ауд.213 к.15		Орг.баз данн. ауд.213 к.15	Комп.граф. ауд.331 к.11	Техн.ств.прог.про д. ауд.213 к.15	Стат.мет. ауд.222 к.15	Техн.зах.інф. ауд.230 к.15	Пр
	3			Диск.мат. ауд.233 к.15			Техн.ств.прог.про д. ауд.213 к.15	Розп.обр. ауд.213 к.15	
	4		Вища мат. ауд.107 к.11					Крос.прог. ауд.224 к.15	
			Укр.мова						

Рис. 3.23. Фрагмент сформованого файла розкладу студентів факультету

Другий спосіб розповсюдження розкладу передбачає його доступність на загальнодоступному інтернет-сервісі. Фрагмент цього сервісу представлений на рисунку 9. Цей сервіс надає можливість всім учасникам навчального процесу швидко отримати інформацію щодо проведення занять, зокрема: день, номер тижня (парний чи непарний, з початку навчального року), номер пари, тип заняття, викладач, а також корпус і аудиторію, де відбуватимуться заняття. Користувачі можуть налаштувати відображення розкладу залежно від таких параметрів: спеціальність, курс та група. Це дозволяє кожному учаснику швидко знайти необхідну інформацію, а також зручно планувати свій час та організовувати свої навчальні заняття.

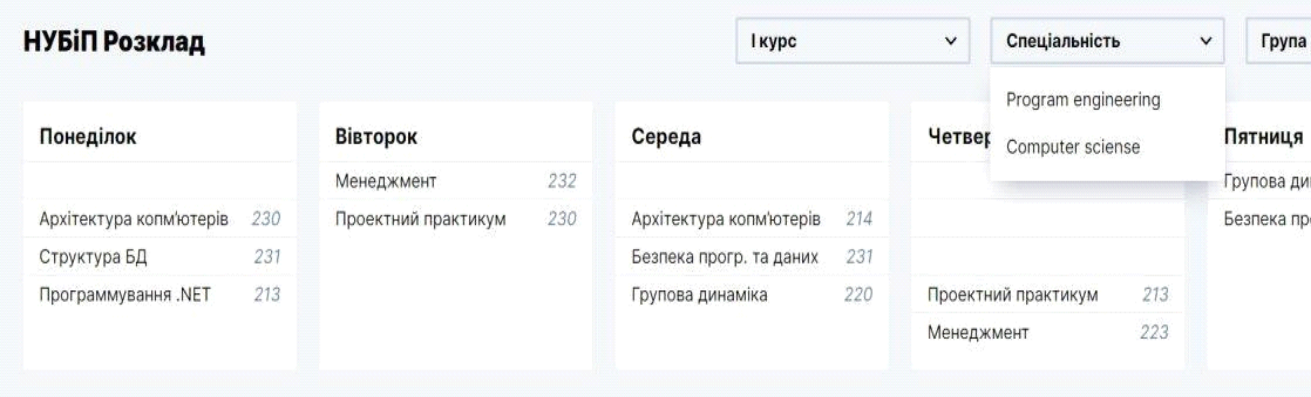

Рис. 3.24. Фрагмент інтернет-сервісу з розповсюдження розкладу

Досліджено проблеми, пов'язані з формуванням розкладу вищих навчальних закладів, а також запропоновано шляхи їх вирішення. У роботі було представлено архітектуру системи, структуру бази даних, алгоритми для розпізнавання елементів документів і готові форми для вибору критеріїв формування розкладу. Також був розроблений і продемонстрований готовий розклад [10, с. 90].

Розроблена система дозволить відмовитися від ручної та паперової роботи, яка зазвичай виконується при створенні та редагуванні розкладу. Автоматизація включає збереження та передачу даних, валідацію, розповсюдження інформації та інші функції. Було проведено попереднє тестування програми для автоматизованого створення розкладу.

Надана інформація може бути корисною для розробки подібних програмних продуктів в умовах невизначеності.

3.6. Тестування програмного модулю

Тестування процес дослідження програмного забезпечення з метою одержання інформації про якість продукту.

Кінцевою метою будь-якого процесу тестування є забезпечення такого ємного поняття як якість, з обліком усіх або найбільш критичних для даного конкретного випадку складових [25, с. 62].

Якість можна визначити як сукупну характеристику досліджуваного ПО з урахуванням наступних складових:

- надійність;
- практичність;
- ефективність;
- мобільність;
- функціональність.

Для реалізації тестування над програмою був виконаний ряд операції. У таблиці 3.9 наведені результати тестування програмного продукту.

Таблиця 3.9.

Результати тестування програмного продукту

Дія	Результат
Запуск програми	Запускається програма. З'являється форма вітання. Відбувається завантаження програми. Зникає форма вітання. З'являється меню програми
Виклик статистики	Відкривається вікно статистик. Статистики виводяться коректно
Виклик настроювань	З'являється вікно настроювань
Установка настроювань	Налаштування встановлюються коректно
Виклик виробничого календаря	Форма відкривається, необхідні дані виводяться
Додавання, редагування й видалення дат	Усі операції виконані успішно

Виклик форми призначеної для роботи з кадрами	Форма відкривається, необхідні дані виводяться
Додавання, редагування й видалення даних персоналу	Усі операції виконані успішно
Дія	Результат
Виклик довідки	Довідка запускається
Натискання на кнопку "Робота з документами"	З'являється вікно, призначене для введення дат, після введення дати з'являється вікно, призначене для генерації розкладу
Установка міток для позначення пріоритетних днів	Мітки встановлюватися й відображаються.
Виконання функцій очищення поля для введення даних	Виконаний успішно
Перевірка алгоритму генерації розкладу	Алгоритм виконаний успішно
Створення документів	Документи створені, форма для їхньої демонстрації показана
Запуск OLE сервера	OLE сервер запущений успішно
Експорт даних в MS Excel	Дані експортовані
Збереження документа	Документ збережений
Закриття OLE сервера	OLE сервер закритий

У результаті проведеного тестування минулого виявлені й виправлені допущені помилки.

Після чого було проведено регресійне тестування, яке проводиться після вдосконалення функцій програми або внесення в неї змін.

Тестування розробленої програми дозволяє зробити висновок, що програма задовольняє вимогам, пропонованим до інтерфейсу взаємодії з користувачем і доступу до даних.

Існуючі на сьогоднішній день методи тестування ПЗ не дозволяють однозначно й повністю виявити всі дефекти й установити коректність функціонування аналізованої програми, тому всі існуючі методи тестування діють у рамках формального процесу перевірки досліджуваного або розроблювального ПЗ [32 ,с. 40].

Такий процес формальної перевірки або верифікації може довести, що дефекти відсутні з погляду використовуваного методу. Тобто , немає ніякої можливості точно встановити або гарантувати відсутність дефектів у програмному продукті з урахуванням людського фактора, що присутствующего на всіх етапах життєвого циклу ПЗ.

Існує кілька класифікацій типів і методів тестування. Відповідно, можна виділити й кілька категорій тестів. Серед найбільш затребуваних на сьогоднішній день: функціональне тестування, навантажувальне, конфігураційне, тестування на зручність експлуатації, а також коректність інсталяції. Залежно від призначення системи випробуванням зазнають різні аспекти її функціональності, відповідно до пріоритетів завдань, які система повинна вирішувати.

Тестування даної програми включило в себе кілька компонентів:

- перевірка правильності взаємодії елементів інтерфейсу;
- тестування методів обробки даних;
- тестування оптимизационных алгоритмів.

При тестуванні був змодельований ряд ситуацій, які досвідчений користувач чи навряд допустить, але їх цілком можна чекати від новачка. Ці випадки, в основному, стосувалися установки вихідних даних.

Програма автоматично генерує розклад служби охорони й автоматично переносить ці відомості в документи, які в наслідку можна експортувати в MS Excel.

Даний програмний продукт підвищить ефективність роботи служби охорони шляхом зменшення тимчасових витрат на складання розкладу і заповнення необхідних документів, так само у випадку втрати документа або приходу його в непридатність документ можна відновити побравши його зі сховища програми. Це дозволить побільшати продуктивність праці служби охорони, що приведе до позитивного виробничого ефекту. Програма дозволить здійснити додатковий контроль над використанням фонду робочого часу, найбільше повно використовувати людські ресурси підприємства [29 ,с. 86].

Періодичність використання програми складе мінімум один раз на місяць, а так само при необхідності скласти розклад і супутні документи.

Мінімальні вимоги, до апаратному й програмному забезпеченню, необхідному для коректної роботи програми є:

- процесор: Pentium III 800МГЦ і вище;
- обсяг оперативної пам'яті не менш 512 Мб;
- не менш 100 MB вільного місця на жорсткому диску;
- операційна система Windows XP SP3;
- наявність маніпулятора "миша";
- клавіатура IBM PC будь-якої модифікації.
- наявність монітора VGA з дозволом не менш 1024x768 крапок;
- наявність Microsoft Office Access 2013 і вище;
- наявність Microsoft Office Excel 2013 і вище;
- наявність Microsoft C++ Redistributable 2015 і вище.

Процес інсталяції відбувається в автоматичному режимі після завдання необхідних параметрів або підтвердження стандартних установок. Після

інсталяції на ПК буде доступна повна версія програми. Додаткове налаштування операційної системи для роботи програми не потрібно.

Для установки програми скористайтеся інсталятором Tpssetup.exe. Після запуску інсталятора додержуйтеся всім вказівок майстра інсталяції. При вказівці шляху для установки програми уникайте в назвах папок спеціальних символів. Після цього ви можете починати роботу із запуску файлу TPS.exe. [35, с. 72]

У процесі реалізації завдання при розробці структури для зберігання даних, першим об'єктом виступають інформація про користувача (працівнику організації) і дані про занесених їм заходах. У нашому випадку БД буде складатися з 5 таблиць.

Надалі по цьому полю ми створимо первинний ключ, щоб СУБД могла швидко знайти потрібний запис для кожного конкретного користувача й заповнити на основі цієї інформації, дати й часу поля на формі додатка. Опис USERS_INDEX презентовано в таблиці 3.10

Таблиця 3.10.

Користувачі (USERS_INDEX)

Назва поля	Тип даних	Тип Firebird	Опис поля
ID	Цілий	INTEGER	Унікальний ідентифікатор користувача.
CAPTION	Строковий	VARCHAR 1000	Логін користувача
USER_PASS	Строковий	VARCHAR 1000	Пароль користувача
FIRSTNAME	Строковий	VARCHAR 1024	Ім'я користувача
LASTNAME	Строковий	VARCHAR 1024	Прізвище користувача

MIDDLENAME	Строковий	VARCHAR 1024	По батькові користувача
BIRTHDAY	Дата	DATE	Дата дня народження користувача.

Таблиця користувачів має дочірню таблицю, у якій зберігаються ідентифікатор користувача й онформация про його статус і приналежності до якої-небудь групи (відділу).

Таблиця 3.11.

Користувачі (USERS_OPTIONS)

Назва поля	Тип даних	Тип Firebird	Опис поля
ID	Цілий	INTEGER	Унікальний ідентифікатор.
USER_ID	Цілий	INTEGER	Унікальний ідентифікатор користувача
VALUE_GROUP	Цілий	INTEGER	Ідентифікатор групи
VALUE_DATA	Цілий	INTEGER	Ідентифікатор рівня доступу

Таблиця PROJECT_INDEX буде створена для зберігання інформації про завдання (таблиця 3.12.).

Таблиця 3.12.

Завдання (PROJECT_INDEX)

Назва поля	Тип даних	Тип Firebird	Опис поля
ID	Цілий	INTEGER	Унікальний ідентифікатор завдання
OWNER_ID	Цілий	INTEGER	Ідентифікатор творця завдання
CAPTION	Строковий	VARCHAR(1024)	Текст завдання

PROJECT_TYPE	Цілий	INTEGER	Ідентифікатор типу завдання
PROJECT_DATE	Цілий	INTEGER	Дата виконання завдання
EXECUTOR	Цілий	INTEGER	Ідентифікатор користувача, призначеного виконавцем
PRIORITY	Цілий	INTEGER	Статус важливості
COMPLETE	Цілий	INTEGER	Оцінка про виконання

У таблиці STATES_INDEX будуть зберігатися всі додаткові потрібні відомості про співвикладачів організації - із вказівкою контактних даних (посада, місце (відділ роботи), телефон внутрішній і мобільний, адреси). Опис STATES_INDEX презентовано в таблиці 3.13.

Таблиця 3.13

Особисті дані (STATES_INDEX)

Назва поля	Тип даних	Тип Firebird	Опис поля
ID	Цілий	INTEGER	Унікальний ідентифікатор (первинний ключ)
GROUP_ID	Цілий	INTEGER	Унікальний ідентифікатор групи
CAPTION	Строковий	VARCHAR(1024)	Повна назва посади співробітника
SHORTCUT	Строковий	VARCHAR(255)	Скорочена назва посади співробітника

WORKPLACE	Строковий	VARCHAR(1024)	Номер кабінету абинет або інше місце роботи
PHONE	Строковий	VARCHAR(255)	Внутрішній номер телефону
MOBILE	Строковий	VARCHAR(255)	Мобільний номер телефону
MAIL	Строковий	VARCHAR(1024)	Адреса електронної пошти
ALLOW_MULTY_USERS	Цілий	INTEGER	Маркер, що дозволяє одночасно кілька підключень під одним. логіном

Третя таблиця SHEDULES_INDEX буде містити відомості про заплановані заходи й справах, ометки про виконання, статус записи й коментарях до неї. Опис SHEDULES_INDEX презентовано в таблиці 3.14.

Таблиця 3.14

Заплановані заходи(SHEDULES_INDEX)

Назва поля	Тип даних	Тип Firebird	Опис поля
ID	Цілий	INTEGER	Унікальний ідентифікатор заходу (первинний ключ)
USER_ID	Цілий	INTEGER	Ідентифікатор відповідального користувача

OWNER_ID	Цілий	INTEGER	Ідентифікатор творця заходу
CAPTION	Цілий	INTEGER	Текст запланованого заходу
DATE_VALUE	Строковий	VARCHAR 1000	Дата занесення заходу
TIME_START	Час	TIME	Час початку
TIME_FINISH	Час	TIME	Час закінчення
RATING	Мале ціле	SMALLINT	Важливість
STATUS	Мале ціле	SMALLINT	Статус заходу
TIME_SUSPEND	Дата	DATE	Час закінчення
COMMENTARY	Строковий	VARCHAR 1000	Коментар
PRIORITY	Цілий	INTEGER	Маркер оповіщення про початок заходу.
SUSPEND_REASON	Цілий	INTEGER	Код, що позначає причину переносу.
ALL_DAY	Цілий	INTEGER	Маркер, що вказує на щоденне виконання заходу
COMPLETE_LEVEL	Цілий	INTEGER	Відсоток завершення
SUSPEND_ID	Цілий	INTEGER	ID відкладеного заходу

Таблиця EVENTS_INDEX буде містити в собі всі події з поділом їх по типу. Таблиця 3.15 - Заплановані завдання (EVENTS_INDEX) буде містити в собі всю інформацію стосовну до подій і роботі з ними [39, с. 51].

Таблиця 3.15

Заплановані заходи(SHEDULES_INDEX)

Назва поля	Тип даних	Тип Firebird	Опис поля
ID	Цілий	INTEGER	Унікальний ідентифікатор події
USER_ID	Цілий	INTEGER	Ідентифікатор польщователя
CAPTION	Строковий	VARCHAR (1000)	Текст події
EVENT_DATE	Дата	DATE	Дата початку події
TIME_START	Час	TIME	Час початку
TIME_FINISH	Час	TIME	Час завершення
TIME_SUSPEND	Дата	DATE	Дата закінчення події
EVENT_TYPE	Мале ціле	SMALLINT	Тип події
STATUS	Мале ціле	SMALLINT	Ідентифікатор стан події
RATING	Мале ціле	SMALLINT	
PRIORITY	Цілий	INTEGER	Важливість події
OPTION_REMIND	Мале ціле	SMALLINT	Маркер включення повідомлення
OPTION_POPUP	Мале ціле	SMALLINT	Маркер включення повідомлення спливаючим повідомленням
OPTION_REFERENCE	Мале ціле	SMALLINT	Опція, що вказує на відображення події в розкладі

COMMENTARY	Строковий	VARCHAR (1000)	Коментар
SUSPEND_REASON	Цілий	INTEGER	Мітка про причину скасування
COMPLETE_LEVEL	Цілий	INTEGER	Прогрес завершення

Таблиця TASKS_INDEX виконує практическу ту ж саму роль, що й SHEDULES_INDEX, з тою лише різницею, що оптимізована для потреб властивим завданням, а не разовим заміткам або завданням на день. Опис TASKS_INDEX презентовано в таблиці 3.16.

Таблиця 3.16

Заплановані завдання (TASKS_INDEX)

Назва поля	Тип даних	Тип Firebird	Опис поля
ID	Цілий	INTEGER	Унікальний ідентифікатор запису
CAPTION	Строковий	VARCHAR (1024)	Повний текст дорученого заходу
TASK_TYPE	Цілий	INTEGER	Тип завдання (на день, на тиждень, на місяць)
SENDER	Цілий	INTEGER	ID поручителя (ФИО користувача)
EXECUTOR	Цілий	INTEGER	ID виконавця (ФИО користувача)
CREATION_DATE	Дата	DATE	Дата створення доручення

DATE_VALUE	Дата	DATE	Крайня дата виконання доручення
STATUS	Цілий	INTEGER	Статус завдання
SENDER_STATUS	Цілий	INTEGER	Оцінка про выполненении поручителем
EXECUTOR_STATUS	Цілий	INTEGER	Оцінка про виконання виконавцем

Для запису регламентних заходів буде заведена своя таблиця. Регламенти відрізняються від звичайних завдань періодичністю повторення. Досить буде лише один раз записати завдання й вибрати з якою періодичністю вона повинна повторюватися в таблиці розкладу дня. Опис REGLAMENT_INDEX презентовано в таблиці 3.17. [27 ,с. 66]

Таблиця 3.17

Регламенти(REGLAMENT_INDEX)

Назва поля	Тип даних	Тип Firebird	Опис поля
ID	Цілий	INTEGER	Уникакльный ідентифікатор запису регламенту.
USER_ID	Цілий	INTEGER	ФИО користувача
CAPTION	Строковий	VARCHAR (1024)	Текст регламентного завдання
ACTION_ALWAYS	Цілий	INTEGER	Маркер "Без тимчасових рамок"
ACTION_START	Дата	DATE	Дата початку
ACTION_FINISH	Дата	DATE	Дата закінчення
ALL_DAY	Цілий	INTEGER	Маркер "Без строку дії"

TIME_START	Час	TIME	Час початку
TIME_FINISH	Час	TIME	Час закінчення
PRIORITY	Цілий	INTEGER	Ступінь важливості(1,2,3 - Звичайна, Важливо, Дуже важливо)
REGLAMENT_ TYPE	Цілий	INTEGER	Тип регламенту (0,1,2 - Щотижневий, Щорічний, Останній у місяці)
FREQUENCY	Цілий	INTEGER	Частота повторення рагламента (дні)
STATE	Цілий	INTEGER	Маркер "Посадовий регламент"
STATUS	Цілий	INTEGER	Статус виконання регламенту

Візуальні засоби розробки запитів забезпечують зручна вистава розв'язуваного завдання, допомагають визначити зв'язки й умови запиту. Але в Firebird немає як таких візуальних засобів розробки запитів. Однак можна звертатися з Firebird за допомогою ibexpert. Це гнучкий і потужний інструмент. У ньому є присутнім можливість розробляти структуру самої БД, зберігати й виконувати скрипты на SQL, використовувати безліч встоенных утиліт, що полегшують роботу, як розроблювача, так і адміністратора.

Використовуючи ibexpert у меню Tools->Query Builder, потрібно перетягнути необхідні таблиці, після чого, стане можливо визначити зв'язку між ними і відзначити поля, необхідні для висновку.

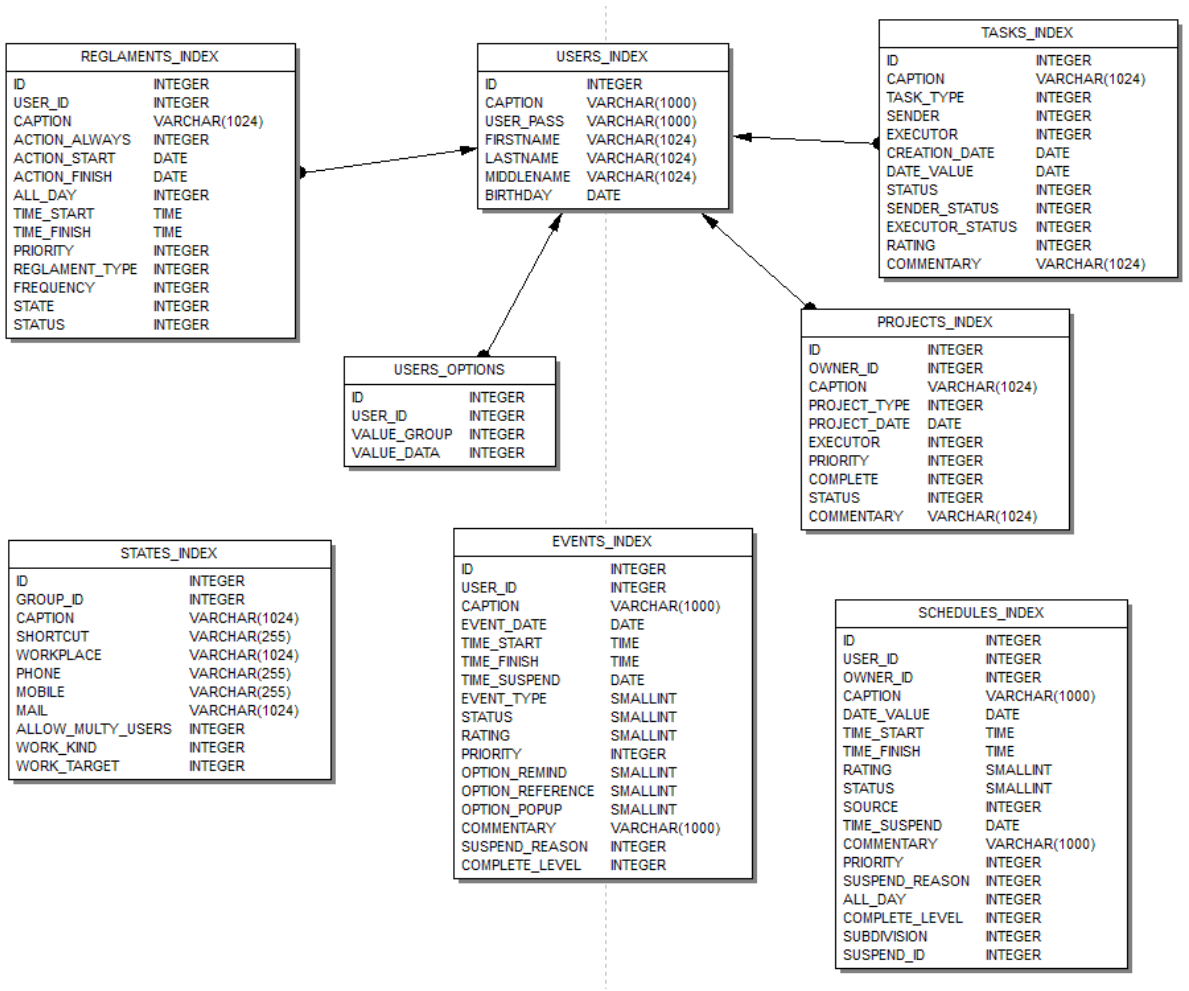

Рис. 3.25. Схема взаємозв'язку таблиць

Центральної (головною таблицею) буде являтся таблиця USERS_INDEX, що містить унікальний ідентифікатор користувача, і іншу основну інформацію (ПІБ, дата народження, пароль). Дані цієї таблиці можна назвати сукупністю ключів для інших таблиць. Ключі зв'язують кожний рядок таблиці даних з асоційованим рядком у дочірніх таблицях [32, с. 54].

Тестування працездатності автоматичної генерації розкладу

Таблиця 3.18.

Тест план перевірки функції автоматичної генерації розкладу

Номер	Номер техничес	Тест кейс	Опис сценарій	Очікуваний результат

	кого Завдання			
АБВ-125	ТЗ-080	Перевірка работоспособности автоматической генерації расписания	Вибрати кнопку "Автоматична генерація" в головному меню	Відкривається інтерфейс автоматичного внесення: списку викладачів, списку аудиторій, списку елементів навчального плану.
			Вибрати кнопку "Згенерувати розклад"	
			Перевірити чи був створений розклад у меню перегляду -> перегляд розкладу -> будь-яка група	У таблиці, повинне бути присутнім розклад, що полягають із 60 записів, що включають у себе:День,Пари, Предмет, Викладача, Аудиторію, Тип (заняття),Корпус, ID

Тестування програмного модулю "Інформаційна система автоматизованого формування розкладу" є важливим етапом у розробці системи, оскільки воно дозволяє виявити та виправити помилки, забезпечити надійність та ефективність роботи модулю. Цей процес забезпечує впевненість у тому, що система буде працювати належним чином та задовольнятиме потреби користувачів.

Інформаційна система автоматизованого формування розкладу спрощує процес створення та редагування розкладу, уникнення помилок та забезпечує швидкий доступ до актуальної інформації для усіх учасників навчального процесу. Такий підхід дозволяє зберегти час і зусилля, які раніше витрачалися на ручну роботу та паперову документацію [25, с. 65].

Отримана інформація та досвід, накопичений під час розробки системи, можуть бути корисними при розробці подібних програмних продуктів у майбутньому. Завдяки цьому, інформаційні системи для автоматизації формування розкладу можуть бути вдосконалені та пристосовані до конкретних потреб різних навчальних закладів.

В цілому, розробка та тестування програмного модулю "Інформаційна система автоматизованого формування розкладу" відкриває нові можливості для ефективного та зручного управління розкладом у вищих навчальних закладах, сприяючи покращенню організації навчального процесу.

ВИСНОВКИ

Таким чином, у ході виконання дипломної роботи було проведено дослідження та розробка інформаційної системи, спрямованої на автоматизацію процесу формування розкладу роботи. Основна мета роботи полягала у визначенні потреб та розробці рішення, яке б забезпечило ефективне планування робочого часу та ресурсів організації.

В ході аналізу поточних методів та практик розкладування було виявлено проблеми, пов'язані з ручним процесом, такі як часові затрати, низька точність та складність управління змінами. Інформаційна система автоматизованого формування розкладу, розроблена в рамках цієї роботи, пропонує ефективний підхід до вирішення цих проблем.

Архітектура системи базується на використанні сучасних інформаційних технологій, таких як бази даних, веб-розробка та алгоритми оптимізації. Це дозволяє системі автоматизовано збирати, обробляти та аналізувати дані про розклади роботи, враховуючи різні фактори, такі як доступність працівників, обсяг роботи та пріоритети.

Застосування інформаційної системи автоматизованого формування розкладу може призвести до таких переваг:

Зменшення часових затрат та зусиль, пов'язаних з процесом розкладування роботи. Система автоматично виконує обчислення та оптимізацію розкладу, що дозволяє звільнити час працівників для виконання інших важливих завдань.

Покращення точності та надійності розкладу. Система враховує різноманітні фактори та обмеження, що дозволяє забезпечити оптимальне використання ресурсів та уникнути конфліктів в графіках роботи.

Забезпечення прозорості та доступності інформації про розклад для всіх зацікавлених сторін. Система може надати зручний інтерфейс для перегляду розкладу, внесення змін та комунікації між учасниками.

Можливість аналізу та використання даних про розклад для прийняття обґрунтованих управлінських рішень. Система зберігає дані про розклади, що дозволяє проводити аналіз ефективності розподілу ресурсів та вносити покращення у процес планування.

Отже, розроблена інформаційна система автоматизованого формування розкладу є потужним інструментом, який може сприяти підвищенню ефективності та продуктивності організацій, забезпечуючи оптимальне використання ресурсів та полегшуючи процес управління робочим часом.

Така інформаційна система може бути особливо корисною для навчальних закладів, де потрібно формувати розклади занять для великої кількості викладачів та студентів. Вона дозволить автоматично враховувати академічні вимоги, наявність ресурсів та пріоритети, спрощуючи процес розкладування та забезпечуючи рівномірний розподіл навантаження на викладачів.

Крім того, інформаційна система автоматизованого формування розкладу може сприяти зменшенню конфліктів та недорозумінь між працівниками, оскільки всі вони матимуть доступ до оновленого розкладу, інформація про зміни буде швидко та точно розповсюджуватися. Це допоможе уникнути зіткнень у графіках роботи та забезпечить кращу комунікацію та співпрацю в колективі.

Застосування інформаційної системи автоматизованого формування розкладу також може мати позитивний вплив на задоволення працівників та студентів. Оптимально розподілені ресурси та збалансований розклад допоможуть уникнути перевантаження та надмірної праці, сприяючи більш ефективному використанню часу та покращенню якості роботи.

Нарешті, інформаційна система може бути розширена та удосконалена з часом, додавши нові функціональні можливості, такі як автоматичне призначення замін, оптимізація розподілу ресурсів або інтеграція з іншими системами управління. Це забезпечить гнучкість та зручність використання системи та підвищить її ефективність в майбутньому.

Отже, інформаційна система автоматизованого формування розкладу є важливим інструментом, який може сприяти поліпшенню організації та управління робочим часом, забезпечуючи ефективне використання ресурсів, зменшення зусиль та покращення якості роботи. Її впровадження може призвести до позитивних змін у роботі організації та сприяти досягненню її цілей та завдань.

Окрім вигод для організацій та навчальних закладів, інформаційна система автоматизованого формування розкладу може також мати позитивний вплив на користувачів, тобто на працівників та студентів. Зручний та легко доступний інтерфейс системи дозволяє користувачам отримати швидкий доступ до своїх індивідуальних розкладів та змін, що робить процес планування та організації робочого часу більш зручним і прозорим.

Крім того, автоматизований процес формування розкладу може допомогти уникнути людських помилок та недоліків, що часто виникають під час ручного планування. Система виконує обчислення, оптимізацію та враховує різні обмеження автоматично, забезпечуючи точність та надійність розкладу.

Крім того, інформаційна система може надати аналітичні звіти та статистику щодо розкладу роботи, які можуть бути корисними для аналізу та прийняття рішень. За допомогою цих даних, організації можуть виявляти тенденції, вдосконалювати процес планування та вирішувати проблеми, що виникають у розкладів.

Загалом, інформаційна система автоматизованого формування розкладу є потужним інструментом, який сприяє ефективному використанню ресурсів, забезпечує точність та надійність розкладу, спрощує процес планування та полегшує комунікацію між учасниками. Її впровадження може мати значний позитивний вплив на організації та їх користувачів, сприяючи покращенню продуктивності, ефективності та задоволення працівників та студентів.

Крім основних переваг, важливо відзначити, що інформаційна система автоматизованого формування розкладу також сприяє економії часу і ресурсів.

Традиційний процес ручного складання розкладу може забирати значну кількість робочого часу і вимагати значних зусиль з боку викладачів та адміністративного персоналу. Автоматизована система дозволяє зменшити цю витрату часу, швидко генерувати розклад та здійснювати необхідні зміни з легкістю. Це дозволяє персоналу організації зосередитися на інших важливих завданнях та забезпечує більш ефективне використання робочого часу.

Крім того, інформаційна система може враховувати різні фактори, які впливають на формування розкладу, такі як наявність необхідних ресурсів, пріоритети, попередження конфліктів у графіках роботи тощо. Це дозволяє досягти оптимального розподілу ресурсів та забезпечити рівномірний навантаження на викладачів та студентів. Такий розподіл сприяє збалансованому навчальному процесу, покращує якість навчання та забезпечує оптимальні умови для розвитку та досягнення успіху учасників навчального процесу.

Крім того, інформаційна система може забезпечити прозорість та доступність інформації про розклад для всіх зацікавлених сторін. Викладачі, студенти та інші учасники можуть швидко та зручно отримувати доступ до свого розкладу через онлайн-інтерфейс. Це дозволяє уникнути недорозумінь, запізнень та конфліктів, пов'язаних з неправильними або неактуальними розкладами. Кожен учасник має можливість завчасно планувати свої заняття та інші зобов'язання, що сприяє ефективному управлінню часом та підвищує загальну організованість.

Отже, інформаційна система автоматизованого формування розкладу є потужним інструментом, який сприяє ефективному використанню ресурсів, забезпечує точність та надійність розкладу, спрощує процес планування та полегшує комунікацію між учасниками. Її впровадження може мати значний позитивний вплив на організації та їх користувачів, сприяючи покращенню продуктивності, ефективності та задоволення працівників та студентів.

Завдяки гнучкості, адаптивності та можливості швидко реагувати на зміни, інформаційна система дозволяє ефективно враховувати потреби та пріоритети учасників навчального процесу. Вона допомагає створити оптимальні умови для навчання, розподілу навантаження та планування занять, що сприяє розвитку і успіху кожного учасника.

Таким чином, використання інформаційної системи автоматизованого формування розкладу є важливим кроком у напрямку модернізації навчальних закладів та організацій. Це дозволяє покращити організацію навчального процесу, забезпечити оптимальне використання ресурсів і підвищити якість навчання. Інформаційна система стає незамінним помічником для всіх учасників, сприяючи раціоналізації робочого процесу та досягненню поставлених цілей.

Інформаційна система забезпечує економію часу та ресурсів, гнучкість та адаптивність у врахуванні змін, прозорість та доступність інформації для всіх учасників. Вона допомагає досягти оптимального розподілу ресурсів та навантаження, сприяє більш ефективному використанню робочого часу та полегшує комунікацію між учасниками. Крім того, вона створює умови для розвитку відкритого та співпрацюючого середовища.

Загальною метою інформаційної системи є поліпшення організації навчального процесу, забезпечення максимального задоволення та досягнення успіху учасників. Її впровадження є актуальним та важливим для сучасних організацій, що прагнуть досягти оптимальної організації та ефективного використання своїх ресурсів.

В цілому, інформаційна система автоматизованого формування розкладу є необхідним інструментом, який допомагає вирішити багато викликів, пов'язаних з плануванням та організацією навчання. Її впровадження веде до покращення організації, ефективності та якості навчального процесу, сприяючи розвитку успішних та організованих навчальних середовищ.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ТА ЛІТЕРАТУРИ

- Абдуліна І. Роблячи ставки на розробку програмного забезпечення [Текст] /І. Абдуліна, В. Березанська// Інтелетуальна власність.–2012.–№9.–С8-12.
- Алгоритми і структура даних: Навчальний посібник / В.М.Ткачук. - ІваноФранківськ : Видавництво Прикарпатського національного університету імені Василя Стефаника, 2016. 286 с.
- Алгоритми та структури даних. Навчальний посібник / Т. О. Коротєєва. Львів : Видавництво Львівської політехніки, 2014. - 280 с.
- Алексенко О. В.. Технології програмування та створення програмних продуктів: конспект лекцій. – Суми : Сумський державний університет, 2013. – 133 с.
- Астахова І. Ф. Створення розкладу навчальних занять на основі генетичного алгоритму / Астахова І. Ф., Фірас А. М. // Вісник воронежского державного університету, серія: «Системний аналіз и інформаційні технології». – 2013. – № 2. – С 93-99.
- Бабкіна Т. С. Задача складання розкладу: рішення на основі багатоагентного підходу / Бабкіна Т. С. // Бізнес-інформатика. – 2008. – №1. – С.23-28.
- Бадд Т. Об'єктно-орієнтоване програмування в дії / Перекл. з англ. - Спб .: Питер, 1997. - 464 с.
- Баркмейер Едвард Дж. (2003). Концепції автоматизації системної інтеграції NIST 2003. – 435 с.
- Бевз С. В. Автоматизація процесу формування розкладу сесії. / Бевз С. В., Войтко В. В., Бурбело С. М., Куба Т. О., Сухоносов О. О. // Принципові концепції та структурування різних рівнів освіти з оптико-електронних інформаційно-енергетичних технологій — 2009 — № 4 — С. 25–36.

- Бевз С. В. Розробка автоматизованої системи формування розкладу магістратури / Бевз С. В., Войтко В. В., Бурбело С. М., Шоботенко А. М. // Інформаційні технології та комп’ютерна техніка — 2009 — № 4 — С. 30–65.
- Буч Г.. Об'єктно-орієнтований аналіз та проектування з прикладами додатків на С ++, 2-е вид. / Пер. з англ. - М .: «Видавництво Бінном», СПб: «Невський діалог», 1998 р - 560 с.
- Верьовкін В. И. Автоматизоване створення розкладів навчальних занять вишу с урахуванням складності дисциплін і втомленості студентів /
- Верьовкін В. И., Ісмагілова О. М., Атавін Т. А. // Доповіді ТУСУР. – 2009. – №1 (19), частина 1. – С. 221-225.
- Винник В. Ю. Основи програмування мовою Сі++ -. Житомир, 2008. – 398 с.
- Войтенко В.В. Морозов А.В. Теорія та практика (мова С++). - Житомир, 2002. – 511 с.
- Вступ до програмування мовою С++. Організація обчислень : навчальний посібник” Ю. А. Бєлов, Т. О. Карнаух, Ю. В. Коваль, А. Б. Ставровський. - К. : Видавничо-поліграфічний центр “Київський університет”, 2012. - 175 с.
- Глоба Л. С. Розробка інформаційних ресурсів та систем [Електронний ресурс] : конспект лекцій / Л. С. Глоба, Т. М. Кот. - Київ : НТУУ "КПІ", 2014. - 318 с.
- Григорович В. Г. Методичні вказівки до виконання лабораторних і практичних завдань з курсу «Алгоритмізація та програмування». Частина 1 : навч. посібник / В. Г. Григорович. – Дрогобич : Редакційно-видавничий відділ Дрогобицького державного педагогічного університету імені Івана Франка, 2016. – 1400 с.

- Грязнова В. О., Єфіменко С. В. Основи методології програмування. - К.: ВПЦ "Київський університет", 2010.
- Девіс Алан М.. Великі дебати щодо програмного забезпечення (8 жовтня 2004 р.), С. 125-128 Wiley-IEEE Computer Society Press\
- Деканова М. В. Математична модель и алгоритм побудови розкладу навчальних занятий університету / Деканова М. В. // Вісник Полоцкого державного університету. Серія С. – 2013. – №12. – С. 24-33.
- Інженерія якості програмного забезпечення: навч. посібник / Г.В Табунщик, Р.К. Кудерметов, Т.І. Брагіна. - Запоріжжя: ЗНТУ, 2013. - 180 с.
- Кравець П. Об'єктно-орієнтоване програмування : навч. посібник / П.О. Кравець. – Львів: Видавництво Львівської політехніки, 2012. – 624 с.
- Кун Д.Л. (1989). "Вибір та ефективне використання засобів автоматизованого програмного забезпечення". Щорічний комп'ютерний симпозіум Westinghouse; 6-7 листопада 1989 р .; Пітсбург, Пенсильванія (США); Проект DOE.
- Лавріщева К. М. Програмна інженерія / К. М. Лавріщева. – К. : Академперіодика, 2008. – 319 с
- Леонов И.В. Введення в методологію розробки програмного забезпечення за допомогою Rational Rose // Ескейп, 2004. – 301 с.
- Мельник А. Архітектура комп’ютерів : підручник / А. Мельник. – Луцьк : Волинська обласна друкарня, 2008. – 470 с.
- Мокін В. Б., Бевз С. В., Бурбело С. М. Розробка та впровадження систем документообігу і менеджменту навчального процесу магістерської підготовки // Оптико-електронні інформаційно-енергетичні технології. — 2006.— № 2. — С. 5–12.
- Мулява І. Я. Вирішення задачі автоматизованого формування розкладу навчального закладу за допомогою генетичних алгоритмів // Міжнародний науковий журнал "Інтернаука". — 2018. — №9.

- Мулява І. Я. Програмна модель формування розкладу навчальних занять // Наука онлайн: Міжнародний електронний науковий журнал — 2018. — №5.
- Николайчук Я. М. Проектування спеціалізованих комп'ютерних систем : навч. посібник / Я. М. Николайчук, Н. Я. Возна, І. Р. Пітух. - Тернопіль : ТзОВ "Терно-граф", 2010. - 392 с.
- Отеро, Карлос. "Виклики дизайну програмного забезпечення". Покращення продуктивності IT. ТОВ "Тейлор і Френсіс". Отримано 19 жовтня 2017. – 280 с.
- Пол Р. Сміт і Річард Сарфаті (1993). Створення стратегічного плану управління конфігурацією за допомогою засобів автоматизованого програмного забезпечення (CASE). Папір для 1993 року Національна група користувачів DOE / Підрядники та об'єкти CAD / CAE.
- Ральф П. та Ванд Ю. Пропозиція щодо офіційного визначення концепції дизайну. У, Lyytinen, K., Loucopoulos, P., Милопулос, Дж., і Робінсон, В., (ред.), Інженерія вимог до проектування: десятирічна перспектива: Springer-Verlag, 2009, с. 103-136
- Робінзон К. (1992). Впровадження програмного забезпечення в CASE. Нью-Йорк: John Wiley and Sons Inc.
- C++. Теорія та практика : навч. посібник / О. Г. Трофименко, Ю. В. Прокоп, І. Г. Швайко, Л. М. Буката та ін. ; за ред. О. Г. Трофименко. – 587 с.
- Сидорова Н. М. Навчання інженерії програмного забезпечення – систематичний огляд літератури/ Н. М. Сидорова // Матеріали міжнародної науково- практичної конференції аспірантів і студентів «Інженерія програмного забезпечення 2011». [Електрон.ресурс].-Спосіб доступу: [http://www.nbuv.gov.ua/portal/ natural/Ipz/2011_2/Sidorova.pdf].

- Системне програмування. Основні поняття [Electronic resource]. – Mode of access:WWW/URL : https://owncloud.kspu.kr.ua/index.php/s/69e091_5776ee3225579f734e986ad017.
- Снитюк В. Є. Аспекти формування цільової функції в задачі складання розкладу занять у вищих навчальних закладах на основі суб’єктивних переваг / Снитюк В. Є., Сіпко Є. Н. // Автоматика. Автоматизація. Електротехнічні комплекси і системи — 2013 — № 2 — С. 98–104.
- Снитюк В. Є. Аспекти формування цільової функції в задачі складання розкладу занять у вищих навчальних закладах на основі суб’єктивних переваг / Снитюк В. Є., Сіпко Є. Н. // Автоматика. Автоматизація. Електротехнічні комплекси і системи - 2013 – №2 – С.98-104.
- Снитюк В. Є. Про особливості формування цільової функції та обмежень в задачі складання розкладу занять / Снитюк В. Є., Сіпко Є. Н. // Математичні машини і системи — 2014 — № 3 — С. 67–76.
- Снитюк В. Є. Про особливості формування цільової функції та обмежень в задачі складання розкладу занять / Снитюк В. Є., Сіпко Є. Н.// Математичні машини і системи – 2014 - №3 – С. 67-76.
- Сяо Хе (2007). "Метамодель для позначення мов графічного моделювання". У: Конференція з комп'ютерного програмного забезпечення та програм, 2007. COMPSAC 2007 - Vol. 1. 31-й щорічний міжнародний, Том 1, випуск, 24–27 липня 2007 р., С. 219-224.
- Тарасенко Р.О. Інформаційні технології: навч. посіб. / Тарасенко Р.О., Гаріна С.М., Робоча Т.П. – К.: Алефа, 2008. – 312 с.
- Технології створення програмних продуктів та інформаційних систем : навч. посібник / М. Ю. Карпенко, Н. О. Манакова, І. О. Гавриленко ; Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. - Харків : ХНУМГ ім. О. М. Бекетова, 2017. - 93 с.

- Функціональне програмування має стати нашим головним пріоритетом в 2015 році [Electronic resource]. – Mode of access : WWW/URL : https://codeguida.com/post/241/.
- Цвіркун Л.І. Глобальні комп'ютерні мережі. Програмування мовою PHP: навч. посібник / Л.І. Цвіркун, Р.В. Липовий, під заг. ред. Л.І. Цвіркуна. – Д.: Національний гірничий університет, 2013. – 239 с.
- Цвіркун Л.І. Розробка програмного забезпечення комп'ютерних систем. Програмування: навч. посіб. [Електронний ресурс] / Л.І. Цвіркун, А.А. Євстігнєєва, Я.В. Панферова ; під заг. ред. Л.І. Цвіркуна. – Електрон. дані та прогр. – М-во освіти і науки України, Нац. техн. ун-т «Дніпровська політехніка». – 1 електрон. опт. диск (CD-ROM) ; 12 см. – Систем. вимоги: Процесор 32-розрядний (x86) 1 ГГц ; MS Windows 7, 8.1, 10 ; CD-ROM drive. – Назва з титул. екрана. – Дніпро: НТУ «ДП», 2019.
- Шаховська Н. Б. Алгоритми та структури даних / Н. Б. Шаховська, Р.О. Голощук. – Львів : Магнолія-2006. – 2009. – 216 с.
- Шевчук І. Б. Інформаційні технології в регіональній економіці: теорія і практика впровадження та використання : монографія. Львів : Видавництво ННВК "АТБ", 2018. 448 с.
- Шеховцов В.А. Операційні системи / В.А. Шеховцов. – К. : Видавнича група BHV, 2005. – 576 с.
- Шпак З. Я. Програмування мовою С / З. Я. Шпак. – Львів : "Оріяна-Нова", 2006. – 431 с.
- Юхимчук С. В., Бевз С. В., Бурбело С. М., Дмитришин О. В., Чернова І. О. Розробка локальної автоматизованої системи розподілу навантаження в процесі ведення документообігу // Вісник Вінницького політехнічного інституту. – Вінниця. – № 1. – 2009. – С. 107-111.
- Юхимчук С. В., Бевз С. В., Бурбело С. М., Крещенецька М. В., Богатчук С. М. Розробка автоматизованої системи формування розподілу

навантаження дисциплін магістерської підготовки // Наукові праці ВНТУ. –2008.–№ 4. С. – 12-17

- Fonseca, C.M.; Fleming, P.J. Genetic algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In Proceedings of the Fifth International Conference on Genetic Algorithms, UrbanaChampaign, IL, USA, 17–22 July 1993; Volume 93, pp. 416–423.
- Guckkenheimer S., Peter J. Software Engineering With Microsoft Visual Studio. Team System. – Adison Wesley, 2006. – 273 p.
- IEEE Standard Glossary of Software Engineering Terminology, Глосарій. IEEE Std 610.12-1990. – (Галузевий стандарт).
- Pinheiro Francisco A. C., Goguen Joseph A.. An Object-Oriented tool for Tracing Requirements // Software.– Mach 1996.– № 3.
- Systems and software engineering – Software Life Cycle Processes. ISO 12207:2008. – [Чинний від 2008-02-01] – II, 122 с.– (Міжнародний стандарт).
- Zave P., Jackson M. Four Dark Corners of Requirements Engineering // ACM Transactions on Software Engineering, January 1997.– № 1.

ДОДАТОК А.

Лістинг програми:

```
        int AlterSet(int index)
{
   int i=current_bestsize;
   if(bestFlags[index] || pool[index]-
>GetFitness()<=(pool[best_pool[current_bestsize-1]]->GetFitness()))        return
0;     while(true)
   {
      if(i<best_pool.size())
      {
if(i==0){
break;
         }
         if(pool[index]->GetFitness()<pool[best_pool[i-1]]->GetFitness())              break;
         best_pool[i]=best_pool[i-1];
       }
      else bestFlags[best_pool[best_pool.size()-1]]=false;        i--;
   }
   if(current_bestsize<best_pool.size())
current_bestsize++;     best_pool[i]=index;
bestFlags[index]=true;     return 0;
}
```

Нижче оператор схрещування в реалізації на мові С ++:

```
Solution* Crossover(Solution* parent1,Solution* parent2)
{
   vector<bool> c_point(all_classes.size());
   for( int i = 5; i > 0; i-- )
   {
      while( 1 )
      {
         int p = rand() % all_classes.size();
```

```
        if( !c_point[ p ] )
        {
           c_point[ p ] = true;
           break;
        }
     }
  }
  Solution* offspring=new Solution(no_of_rooms,no_of_classes);     map<Class*,
int>::const_iterator p1list_iter = (parent1->GetClasses()).begin();     map<Class*,
int>::const_iterator p2list_iter = (parent2->GetClasses()).begin();     int first=rand()%2;
  for(int i=0;i<all_classes.size();i++)
  {       if(c_point[i])
first=1-first;       if(first)
{
        (offspring->GetClasses()).insert(pair<Class*, int>((*p1list_iter).first,
(*p1list_iter).second));
        for(int i=0;i<(*p1list_iter).first->GetDuration();i++)
        {
offspring.GetSlots().at((*p1list_iter).second+i).push_back((*p1list_iter).first);
        }
     }
else
     {
        offspring->GetClasses().insert(pair<Class*, int>((*p2list_iter).first,
(*p2list_iter).second));
        for(int i=0;i<(*p2list_iter).first->GetDuration();i++)
        {
                        offspring-
>GetSlots().at((*p2list_iter).second+i).push_back((*p2list_iter).first);
        }
     }
     // iterating the parent lists
p1list_iter++;       p2list_iter++;
  }
```

```
   CalculateFitness(offspring);     return
offspring;

}
```

Зразок оператора з програми:

```
Solution* Mutation(Solution* ini)

{

   if( rand() % 100 > mutationProbability )        return ini;

   int size = DAYS_NUM*DAY_HOURS*no_of_rooms;

   for( int i = mutationSize; i > 0; i-- )

   {

      int mutation_position = rand() % no_of_classes;         int pos1
= 0;

      map<Class*, int>::iterator it = ini->GetClasses().begin();         for( ;
mutation_position > 0; it++, mutation_position-- );         pos1 = ( *it ).second;
      Class* mutation_class = ( *it ).first;         int rooms =
config.GetNumberOfClassRooms();         int duration_class =
mutation_class->GetDuration();         int day = rand() %
DAYS_NUM;         int room = rand() % rooms;
      int time = rand() % ( DAY_HOURS + 1 - duration_class );         int pos2 = day * rooms *
DAY_HOURS + room * DAY_HOURS + time;         for( int i = duration_class - 1; i >= 0; i-- )

      {

         list<Class*>& slot_iterator = ini->GetSlots()[ pos1 + i ];

         for( list<Class*>::iterator it =slot_iterator.begin(); it != slot_iterator.end(); it++ )

         {

            if( *it == mutation_class )

            {

               slot_iterator.erase( it );
break;

            }

         }

         ini->GetSlots().at( pos2 + i ).push_back( mutation_class );

      }
```

```
        ini->GetClasses()[ mutation_class ] = pos2;

    }

    CalculateFitness(ini);     return ini;

}

void CalculateFitness(Solution* ch)

{

    int criteria_number=0;

    int daySize=DAY_HOURS*no_of_rooms;
vector<bool>  parameter=ch->GetCriteria();     int
score=0;

    for(map<Class*, int>::const_iterator it = ch->GetClasses().begin(); it !=
ch>GetClasses().end(); ++it, criteria_number += 5 )

    {

        int position=(*it).second;        int day =
position / daySize;        int time = position %
daySize;        int room = time / DAY_HOURS;
time = time % DAY_HOURS;
        int duration = ( *it ).first->GetDuration();
bool room_overlap=false;        for(int
i=0;i<duration;i++)
        {

            if(ch->GetSlots()[position+i].size()>1)              room_overlap=true;

        }

        if(!room_overlap)
score++;

        parameter[criteria_number+0]=!room_overlap;

        Class* curr=(*it).first;

        ClassRoom* r=config.GetClassRoomById(room+1);        parameter[criteria_number+1]=r-
>GetStrength()>=curr->GetStrength();        if(parameter[criteria_number+1])          score++;
        parameter[criteria_number+2]=!curr->LabClass() || ( curr->LabClass() && r-

>IsLab());

        if(parameter[criteria_number+2])          score++;

        bool teacher_clash=false,student_clash=false;        int
break_check=0;

        for(int i=0,p=day*daySize+time;i<no_of_rooms;i++,p+=DAY_HOURS)
```

```
    {
        for(int i=0;i<duration;i++)
        {
            list<Class*>& cl = ch->GetSlots()[ p + i ];
            for( list<Class*>::const_iterator it = cl.begin(); it != cl.end(); it++ )
            {
                if( curr != *it )
                {
                    if( !teacher_clash && curr->TeacherClash( **it ) )
teacher_clash = true;
                    if( !student_clash && curr->StudentClash( **it ) )
student_clash = true;
                    if( teacher_clash && student_clash )
break_check=1;
                }
            }
            if(break_check)
break;
        }
        if(break_check)
break;
    }
    if(!teacher_clash)
score++;
    parameter[criteria_number+3]=!teacher_clash;
if(!student_clash)          score++;
    parameter[criteria_number+4]=!student_clash;
  }
  ch->GetFitness()=(float)score/(config.GetNumberOfSubjectClasses()*5);
}
```