МІНІСТЕРСТВО НАУКИ ТА ОСВІТИ УКРАЇНИ

ВСП ТАЛЬНІВСЬКИЙ БУДІВЕЛЬНО-ЕКОНОМІЧНИЙ КОЛЕДЖ

УМАНСЬКОГО НУС

**ДИПЛОМНИЙ ПРОЕКТ**

ПРОГРАМНА СИСТЕМА ДЛЯ АВТОМАТИЗАЦІЇ СКЛАДАННЯ РОЗКЛАДУ ЗАНЯТЬ ВСП ТАЛЬНІВСЬКИЙ БУДІВЕЛЬНО-ЕКОНОМІЧНИЙ КОЛЕДЖ УНУС

ТБЕК.188314.001 ПЗ

Пояснювальна записка

Виконавець

Керівник

студент групи ПС-46  
Поліщук В.П.

викладач  
Харченко О.О.

Захист дозволяю:

Голова циклової комісії

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Завірюха В.П.

ЗМІСТ

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

2

ТБЕК.188314.001 ПЗ

Розроб.

Поліщук В.П..

Перевір.

П.І.Б.

П.І.Б.

Н. Контр.

П.І.Б.

Затверд.

П.І.Б.

ПРОГРАМНА СИСТЕМА ДЛЯ АВТОМАТИЗАЦІЇ СКЛАДАННЯ РОЗКЛАДУ ЗАНЯТЬ ВСП ТАЛЬНІВСЬКИЙ БУДІВЕЛЬНО-ЕКОНОМІЧНИЙ КОЛЕДЖ УНУС

Літ.

Акрушів

2

ПС-46

ВСТУП

1 ПОСТАВНОВКА ЗАДАЧІ

2 АКТУАЛЬНІСТЬ ЗАДАЧІ

3 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

3.1 Огляд технічних проблем

3.2 Організація об’єктної моделі

3.3 Організація структури бази даних

3.4 Організація роботи прикладних програм

3.5 Організація програмної системи

3.6 Огляд аналогів

4 ОПИС СИСТЕМИ

4.1 Структура системи

4.2 Розробка системи

4.3 Структура бази даних

4.4 Робота системи

5 ВИБІР ЗАСОБІВ РЕАЛІЗАЦІЇ

5.1 Операційна система

5.2 Мова програмування і середовище розробки

5.3 Система керування базами даних

5.4 Необхідні вимоги до конфігурації ЕОМ

ВИСНОВКИ

СПИСОК ЛІТЕРАТУРИ

Додаток А Специфікація

Додаток Б Текст програми

Додаток В Інструкція користувачеві

**ВСТУП**

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

3

ТБЕК.188314.001 ПЗ

Метою виконання дипломного проекту на тему «Програмна система для автоматизації складання розкладу занять Тальнівського будівельно-економічного коледжу УНУС» має на меті розробку такої програмної системи, що буде володіти набором засобів підвищення поточної швидкості створення розкладу занять та графіка навчального процесу, підвищення рівня точності вказаних даних та звільнення персоналу відповідального за створення розкладу занять від механічної та рутинної роботи, що наддасть можливість врахувати більше побажань трудового колективу. Також дана програмна система має виключати можливість створення помилок або накладок під час розробки розкладу занять.

Актуальність теми дипломного проекту була і є на високому рівні. Поточний стан предметної області дипломного проекту володіє неймовірно низьким рівнем автоматизації. Автоматизація присутня лише на рівні створених персоналом наробок і заготовок, що не здатні задовольнити наростаючу потребу у швидкості роботи. Створення програмної системи дозволить підняти автоматизацію на високий рівень, додати можливості повторного використання уже вирішених задач по створенню розкладу занять. Вирішення задачі дипломного проекту підніме швидкість створення розкладу занять, що зменшить навантаження на навчальну частину на початку року і дозволить вірно планувати навчальний процес.

Перед початку роботи над дипломним проект було проведено огляд ринку на наявність програмних продуктів такої ж та схожої спеціалізації. Після проведення огляду було зроблено висновки, що на ринку існують аналоги даної програмної системи, але всі вони володіють проблемами. Даний дипломний проект зможе вирішити дані проблеми і надати гідну альтернативу. Серед найбільш значних проблем можна виділити високу вартість програмного продукту для навчальних закладів такого рівня фінансування. Придбання ліцензії для використання програмного продукту в навчальному закладу є не рентабельною. Також існують аналоги, що володіють низькою вартістю або безкоштовною ліцензією, але в таких пропозицій існує інша проблема полягає в тому, що вони є спеціалізованими з урахуванням особливостей функціонування того навчального закладу для якого вони були створені. Використання такого продукту створює складності, а іноді навіть не можливе в повній мірі.

Поточний рівень рішення задачі створення розкладу занять знаходиться на низькому рівні автоматизації, що вимагає від адміністрації закладу виділяти трудові ресурси та час на його реалізацію. Також можна виділити проблему низького рівню контролю за точністю даних через що досить часто помилки та помилки набору тексту можна знайти навіть у кінцевому результаті роботи. Це трапляється через те що людина не здатна контролювати одночасно таку велику кількість однотипних даних. Також варто виділити проблему швидкості створення розкладу занять. Через використання ручних методів створення вирішення задачі відбувається досить повільно, що впливає на всі інші процеси навчання. Вирішення за допомогою комп’ютера значно зменшить вплив цієї проблеми.

Виконання дипломного проекту додатково окрім основного завдання вирішує додаткові завдання, що має вплив на інші роботи. Дипломний проекту виконує по при функції розробки і створення розкладу занять володіє набором функції для розробки графіку навчального процесу, а також володіє функція аналізу та обробки інформації, що надає інформативний характер, перевірки розкладу занять і графіку навчання на правильність виконання

Програмний продукт, що розробляється володіє наступний технічними характеристиками:

* Адаптивність - забезпечує найбільшу зручність у використані на різних робочих станціях
* Багатоплатформний – створений програмний продукт може використовувати на різних операційних системах без потреби у додатковому доопрацюванні чи змінах
* Незалежність даних – всі дані не залежать від робочої станції і зберігається дистанційно, що збільшує надійність та безпеку. Також такий формат збереження даних надає можливість переходу на іншу робочу станцію або навіть одночасну роботу декількох робочих станцій у програмі
* Підтримка форматів Excel – можливість збереження всіх даних у найбільш популярному і зручному форматі
* Масштабований – програмний продукт виконує всі вимоги для можливості подальшої розробки та додання нового функціоналу

Розробка програмного продукту, що є темою даного дипломного проекту очікувано має зменшити в декілька разів зменшити затрати часу та ресурсів трудового колективу на виконання завдання створення розкладу занять. Зменшення ресурсів дозволять використати їх в іншому місці, або звільнити кошти, що могли б бути витрачені на їх реалізацію. Виконання робіт вчасно за допомогою даного програмного продукту збільшить ефективність планування і зменшить потреби у внесенні змін до навчального процесу під час семестру.

**1 ПОСТАНОВКА ЗАДАЧІ**

Основне завдання даного дипломного проекту - розробка програмної системи за темою «Програмна система для автоматизації складання розкладу занять ВСП Тальнівський будівельно-економічний коледж УНУС». Тема дипломного проекту вимагає створити такої програмної системи, що має максимально автоматизувати процес створення розкладу занять для навчального закладу, володіти функціонал для автоматизації створення графіку навчального процесу, а також виконувати аналітичні функції та функції контролю введених даних

Під час консультації керівником дипломного проекту було видано список вимог, що є обов’язковими для результуючої програми. Цей список містить наступні пункти:

Дипломний проекту має бути побудований з використанням:

1. Об’єктно-орієнтованої мови програмування
2. Серверної реляційної бази даних

Дипломний проект повинен забезпечити можливість введення та збереження для подальшого використання наступні дані:

1. Аудиторії, що складається з назви та коротких відомостей про кожну
2. Викладачі, що мають володіти ім’я, короткими відомостями та списком вимог до використання в розробці розкладу занять
3. Предмети, що маються містити особливості використання в розкладі занять, а також закріплену аудиторію

Дипломний проект має забезпечувати наступний функціонал у повному обсязі:

1. Створення графіку навчального процесу
2. Виведення даних про кожен день з графіку навчального процесу
3. Врахування виключень в навчальному процесі на випадок святкових днів та загальнодержавних вихідних
4. Створення розкладу занять
5. Контроль розкладу занять за наявністю збігів та помилок
6. Контроль за дотриманням вимог кожного з викладачів
7. Дипломний проект повинен забезпечувати функціонал для можливості виведення розкладу занять та графіку навчання у форматі Excel

До програмної системи ставляться додаткові вимоги по збереженню та захисту інформації:

1. Перевірка правильності збереженої інформації
2. Перевірка цілісності інформації
3. Шифрування збереженої інформації
4. Стиснення збереженої інформації
5. Можливість резервного копіювання

До програмної системи ставляться такі вимоги по реалізації та виконанні:

1. Можливість використання на різних операційних системах
2. Використання на різних дисплеях, без втрати можливостей сприйняття інформації
3. Можливість відновлення системи після пошкодження
4. Вимоги до середовища розробки не висуваються

**2 АКТУАЛЬНІСТЬ ЗАДАЧІ**

В даному дипломному проекту була поставлена задача розробки програмної системи, що володіла б функціоналом для автоматизації створення розкладу занять для навчального закладу, аналізу даних для виведення звітів та контролю за дотриманням поставлених вимог до задачі. Актуальність вирішення задачі знаходилась і знаходиться на дуже високому рівні. Причиною внесення даної теми до списку тем дипломних проекті була саме її висока актуальність в даний момент і особиста зацікавленість в розробці програмної системи адміністрації навчального закладу, що критично потребує для прискорення роботи саме такий продукт.

Тема автоматизації з кожним роком стає все більш актуальною. Її актуальність відображається на як на світовому масштабі, так і на локального рівні, що проявляється особливо проявляється на навчальному закладі для якого розробляється даний дипломний проект. Зростання актуальності автоматизації відбувається з постійно наростаючим темпом через розвиток навчального закладу, що полягає в збільшення кількісті вступників кожного року та відкритті нових напрямків навчання. Такий розвиток збільшує важкість роботи розробки розкладу занять в квадратичні пропорції.

Варто зазначити, що автоматизація стає популярнішою і по іншій причині. Такою причиною є зменшення собівартості вартості, а свою чергу і збільшення доступності, технологій для її реалізації. З кожним роком відбується зменшення собівартості обчислювальної техніки. Те що було не можливим для реалізації декілька років тому назад через високу вартість обладнання для вирахування такої схеми зараз доступне майже кожному. Також збільшується можливості інструментів для розробки і створення нових алгоритмів для роботи, що також не мало важить в процесі оцінки актуальності задачі даного дипломного проекту.

Актуальність вирішення поставленої задачі не обмежується лише навчальним закладом для якого вона розробляється на даний момент. Програмний продукт, що розробляється володію набором функції та можливостей для використання його в інших навчальних закладах без особливих проблем з інтеграцією в процес роботи. Програмний продукт розроблений з набором можливостей для охоплення максимальної аудиторії можливих клієнтів.

Тема дипломного проекту була актуальною раніше, володіє високою актуальністю зараз і буде актуально ще багато років.

**3 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ**

**3.1 Огляд технічних проблем**

Тема дипломного проекту володіє рядом технічних проблем, що потрібно вирішити для виконання програмної системи у повному обсязі. Проблеми відрізняються з складністю та обсягом, але всі вони важливі для повного виконання завдання.

Серед проблем можна виокремити наступні проблеми:

* Відсутність автоматизації як такої на поточному рівні виконання завдання створення розкладу занять
* Відсутність можливості перенесення поточного рішення на рівень автоматизації без повної переробки
* Відсутність розуміння персоналу технічних проблем роботи
* Потреба у перенесенні всіх даних до іншого формату, який доступний для використання кінцевому користувачеві
* Потреба у використанні незалежних даних для можливості стиснення, безпеки та кодування інформації

На даний момент відсутня автоматизація, що могла б використовувати при проектуванні програмної системи і могла б полегшити роботу розробки. Всі поточні наробки доступні лише при виконанні роботи ручним методом і не можуть бути використанні. З цієї причини присутня потреба у створенні нових методів для автоматизації роботи, що вимагає вивчення всієї предметної області.

Всі поточні рішення володію лише паперовим форматом і відсутня будь-яка можливість використовувати дані напряму без додаткових затрат. Для використання даних у програмі потрібно виконувати їх конвертування в електронний формат доступний для використання у розробленій програмній системі. Також відсутня сукупність даних які потрібні у програмі і вони можуть бути зібрані швидко.

Через причину не досконалості розуміння завдання проекту персонал не здатний надавати точні та зрозумілі дані для роботи. Іноді відбувається надання суперечливих даних, що сповільнює процес розробки системи через потребу в аналізі та вирішення, які дані є правильними.

Кінцевий результат повинен бути доступний у тому форматі яким користуються працівники зараз для виконання завдання. Дані формати є протилежними до методів, що використовуються у програмних системах, тому присутня проблема обробки та конвертації даних після виконання завдання в програмній системі.

До дипломного проекту була поставлена вимога використання баз даних, що володіють власним форматом збереження даних, який потребує додаткових сил для перетворення та використання у програмній системі. Більше того на цьому рівні має відбуватися постійний контроль за правильністю збережених даних та виведення застережень у випадку пошкодження або відновлення даних, якщо це можливо.

**3.2 Організація об’єктної моделі**

Для роботи програми використовуються об’єктно-орієнтована мова програмування високого рівня, що вимагає від розробки виконання додаткового процесу створення об’єктної моделі системи.

Об’єктно-орієнтоване програмування – це один з методів програмування, що розглядає програму систему як множину «об’єктів», що володіють зв’язками один з одним і можуть взаємодіяти для досягнення загальної мети. Основу об’єктно-орієнтованого програмування складають основні концепції: інкапсуляція, успадкування, поліморфізм та абстракція.

Об’єкт в об’єктно-орієнтованому програмуванні - основне поняття в технології проектування та програмування. Об’єкт є втіленням абстрактної моделі окремої функції, що володіє чітко вираженим функціональним призначенням в області. Визначення об’єкта тісно пов’язане з визначенням класу та характеризується своїми властивостями та поведінкою. Об’єкт володіє набором атрибутів або полів даних, що визначають його поточний стан у множині станів. Поведінку об’єкта визначається його функціями або методами. Взаємодія відбувається через передання повідомлень між об’єктами.

Клас в об’єктно-орієнтованому програмування – це спеціальна конструкція, що використовується для групування пов’язаних даних та методів. Екземпляром класу називається об’єкт.

Інтерфейс – це спосіб визначення засобів для незв’язних об’єктів виконувати спілкування та обмін даними один з одним. Відбувається визначення методів та даних , з якими погоджуються об’єкти для співпраці.

Інкапсуляція – це один із механізмів об’єктно-орієнтованого програмування, що має на меті заборону доступу до даних напряму, а об’єкт містить не тільки дані, але і правила їх обробки у вигляді методів. Також цей механізм описує керування доступом до стану об’єктів. Застосування механізму інкапсуляцію дозволяє зменшити зв’язність класів.

Успадкування – це механізм утворення нових класів на основі розроблених раніше. Цей принцип дозволяє використовувати властивості та функціональність батьківського класу. Успадкування вводиться поняття породжених класів або підкласів, а також класів на яких базується породжений клас – базовий клас.

Поліморфізм – концепція в програмуванні, що визначає використання спільного інтерфейсу для обробки різних спеціалізованих даних. Також під концепцією поліморфізму є здатність підкласів виконувати роль об’єктів батьківського класу, завдяки чому екземпляр підкласу може використовувати там, де використовується батьківський клас.

Абстракція – це метод керування складністю системи. Цей метод дозволяє використовувати лише ті дані, які потрібні для роботи програми і дозволяє зменшити складністю за допомогою відкидання лишніх даних.

Для створення програмної системи використовували принципи об’єктно-орієнтованого програмування розроблені Робертом Мартіном, що володіють абревіатурою SOLID. Даний принцип володіє складовими: принцип єдиного обв’язку, принцип відкритості або закритості, принцип підстановки Лісков, принцип розділення інтерфейсу та принцип інверсії залежностей.

Принцип єдиного обов’язку – це принцип при якому кожен об’єкт має виконувати лише один доручений його обов’язок.

Принцип відкритості/закритості – принцип при якому програмні сутності повинні бути відкрити для розширення, але в той же час бути закритими для змін. Розширення класу чи інтерфейсу може здійснюватися через його успадкування.

Принцип підстановки Барбари Лісков – принцип при якому об’єкти в програмній системі можуть бути замінені їх нащадками без внесення змін до коду програми.

Принцип розділення інтерфейсу – принцип при якому використання багатьох спеціалізованих інтерфейсів краще за один універсальний. Інтерфейс може бути поділений на спеціалізовані інтерфейси ще на стадії проектування задля майбутньої гнучкості програмних компонентів.

Принцип інверсії залежностей – залежності в середині системи мають будувати на основі абстракцій. Абстракції не повинні бути залежними від деталей, а навпаки, деталі мають залежати від абстракцій. Модулі вищих рівнів не повинні бути залежними від модулів нижчих рівнів, а навпаки, модулі нижчих рівнів мають залежати від модулів вищих рівнів.

Дані принцип володію дуже тісним взаємозв’язком з принципами об’єктно-орієнтованого програмування. Зокрема принцип підстановки уособлює в собі успадкування. Принцип інверсії залежностей володіє розширеним поняттям трактування поліморфізму. Принцип єдиного обов’язку володіє тісним зв’язком з принципом інкапсуляції, що також передбачає відокремлене існування набору методів.

Для проектування програмної системи використовуються шаблони проектування. Одним із шаблоні, що набув найбільшої популярності є шаблон розроблений Еріхом Гамма, Річардом Хелм, Ральфом Джонсоном та Джоном Влісідеса. Даний набір шаблонів володіє назвою «Шаблони проектування», що описані в книзі «Методи об’єктно-орієнтованого програмування. Шаблони проектування». Для проектування програмної системи використовувалися наступні шаблони проектування:

* Адаптер – структурний шаблон проектування, призначений для організації використання функцій об’єкта, недоступного для модифікації через спеціально створений інтерфейс. Дозволяє в процесі проектування не приймати до уваги можливі різності в інтерфейсах уже існуючих класів. Якщо існують класи з потрібними методами і властивостями то завжди можна використовувати шаблон Адаптер
* Міст – структурний шаблон проектування програмного забезпечення, використовуваний щоб розділити абстракцію та реалізацію так щоб вони могли видозмінювати незалежно. Шаблон використовує інкапсуляцію, агрегацію та наслідування, щоб розділити відповідальність між класами. Основною ціллю є реалізація програми при які клас володіє мінімальними знаннями про реалізацію, а потреба в зміні програми велика
* Декоратор – структурний шаблон проектування, призначений для динамічного підключення додаткової поведінки до об’єкта. Надає гнучку альтернативи практичного створення підкласів з метою розширення функціональності. Для тієї ж мети може використовувати поведінковий шаблон Стратегія
* Стратегія – поведінковий шаблон проектування, призначення для визначення сімейств алгоритмів, інкапсуляція кожного з яких і забезпечує їхню взаємозамінність. Дозволяє обрати алгоритм шляхом визначення відповідного класу. Також даний шаблон надає можливість змінювати алгоритм не залежно від об’єктів-клієнтів
* Фасад – структурний шаблон проектування, який дозволяє скрити проблеми реалізації системи шляхом зведення всіх можливих зовнішніх викликів до одного об’єкта, якому делегується відповідні об’єкти системи. Шаблон використовується для застосування деякого виду політики відношення до іншої групи об’єктів. Якщо політика має бути явною використовується Фасад. Для забезпечення скритої і акуратної політики використовується шаблон Замісник.
* Замісник – структурний шаблон проектування, який надає об’єкт, що контролює доступ до іншого об’єкту перехоплюючи всі його виклики. В основному виконує функцію контейнера.
* Посередник – поведінковий шаблон проектування, що забезпечує взаємодію множини об’єктів, формуючи при цьому слабку взаємозалежність і позбавляючи об’єкти необхідності явно посилатися одне на одного.

Для проектування та моделювання об’єктної моделі системи використовувала уніфікована мова моделювання, що має назву UML.

UML (Unified Modeling Language) – уніфікована мова моделювання, що використовується у парадигмі об’єктно-орієнтованого програмування. Мова моделювання використовує графічні позначення для створення абстрактної моделі системи, що має назву UML-модель. Використовується для визначення, візуалізації, проектування й документування в основному програмних систем.

**3.3 Організація структури бази даних**

Для роботи і збереження даних програмної системи дипломного проекту використовувався трансакційна реляційна база даних серверного типу. Для проектування та формування моделі бази даних використовувалися структурована мова запитів.

База даних – це сукупність даних, що організовані відносно моделі, яка описує загальні характеристики цих даних та їх взаємозв’язки. Також базою даних можна вважати впорядкований набір даних, що об’єднані за певною характеристикою.

Реляційна база даних – це база даних, що заснована на реляційній моделі даних. Реляційна база даних є сукупністю елементів даних, організованих у вигляді набору формально описаних таблиць, з яких дані можуть бути доступними або повторно зібрані багатьма різними способами без необхідності реорганізації таблиць бази даних.

Транзакція – це група послідовних операцій з базою даних, яка є логічною одиницею роботи з даними. Транзакція може бути виконана або цілком успішно, дотримуючись цілісності даних і незалежно від інших транзакцій, що йдуть паралельно, або не виконана зовсім, і тоді вона не може справити ніякого ефекту.

SQL (Structured query language – мова структурованих запитів) – декларативна мова програмування для взаємодії розробника з базами даних, що застосовується для формування запитів, оновлення і керування реляційними базами даних, створення схеми бази даних і її модифікації, система контролю за доступом до бази даних. Також SQL можна визначити як діалогову мову програмування для здійснення запиту і внесення змін до бази даних, а також управління базами даних. На основі коду SQL ядро бази даних формує командну мову, що дозволяє здійснювати пошук, вставку, оновлення, і вилучення даних, використовуючи систему управління і адміністративні функції.

Запит – це формулювання інформаційної потреби користувачем деякої бази даних або інформації системи. Для складання запиту використовується мова пошукових запитів. Запити поділяються на інформаційні, навігаційні та трансакційні запити.

Для роботи бази даних використовується ядро бази даних, що називається підсистемою низького рівня, що визначає механізми управління та збереження даних, а також бібліотеки, що підключаються та функції, що можуть визначати функції для системи керування базами даних.

Системи керування базами даних – це сукупність програмних та лінгвістичних механізмів загального та спеціального призначення, що забезпечують управління створенням і використанням баз даних. Основними функціями системи керування базами даних:

* управління даними у зовнішній пам’яті;
* управління даними в оперативній пам’яті з використанням дискового кешу;
* журналювання змін, резервне копіювання та відновлення бази даних після помилок
* підтримка мов баз даних, в особливості мов визначення даних та мови маніпулювання даними

Журналювання – це функція системи керування базами даних, що зберігає інформацію, необхідну для відновлення бази даних з попереднім зіставленням станів у випадку логічних або фізичних відмов у роботі системи. В найпростішому випадку журналювання полягає в послідовному записі у зовнішню пам’ять всіх змін, що виконувалися в базі даних. В кожен запис входить коротка інформація, що містить: порядковий номер, тип і час зміни; ідентифікатор транзакції; об’єкти, що були задіяні в зміні; попередній стан або новий стан об’єкта. Процес журналювання формує журнал змін.

**3.4 Організація роботи прикладних програм**

Для роботи програмної системи даного дипломного проекту використовуються прикладні програми для розширення можливого надаваного функціоналу та можливостей роботи системи. Підключення до прикладних програм виконується через бібліотеки та драйверами, що були створюються розробниками прикладних програм для можливості роботи з іншими програмними системами. Робота драйверів та бібліотеки виконується на мережевих сокетах або за допомогою спеціальних каналів зв’язку, що забезпечуються операційною системою.

Драйвер – це комп’ютерне програмне забезпечення з допомогою якого інше програмне забезпечення, в особливості операційна система, виконує доступ до апаратного забезпечення пристроїв підключених до системи. Додатково драйвер може імітувати існування пристроїв і надавати програмні сервіси для існуючих систем.

Бібліотеки – це набір підпрограм та об’єктів, що використовуються для розробки та функціонування програмного забезпечення. Можуть виконувати роль модулів системи. Бібліотеки можуть розширювати надаваний програмною системою функціонал, що збільшить галузь його призначення. Бібліотеки можуть бути динамічними та статичними. В залежності від типу бібліотеки вона може бути постійно завантажена в пам’ять процесора або завантажувати по мірі необхідності у її використанні.

**3.5 Організація програмної системи**

Суть теми дипломного проекту «Програмна система для автоматизації складання розкладу занять ВСП Тальнівський будівельно-економічний коледж УНУС» є розробка програмної системи, що зможе максимально автоматизувати процес створення розкладу занять і всіх його складових та зможе перекласти всі проблеми роботи на автоматизовану програмну систему. Дана програмна система включає в себе абстрактні поняття, що є представленням реально існуючих сутностей, які потребують пояснення.

«Розклад занять» - це сутність, що присутня в програмній системі і включає в себе майже всі присутні у програмній системі сутності для функціонування. Розклад занять є представленням реального розкладу занять навчального закладу. Розклад занять включає в себе дані про момент створення, час на який даний розклад є актуальним для використання, список спеціальних елементів, що володіють назвою «Елемент розкладу занять».

«Елемент розкладу занять» - це абстрактна сутність програмної системи, що включає в себе дані про групу, а також заняття які призначені для цієї групи. Додатково використовується інформація про порядок тижня для якого інформація є актуальною та час на який призначене заняття.

«Заняття» - абстрактна сутність, що використовується у програмній системі для представлення реального заняття. Кожне заняття володію інформацією про викладача, предмет та аудиторію в якій проводиться заняття.

«Викладач» - сутність, що використовується у програмній системі для представлення реального викладача. Кожен викладач володіє інформацією про власного реального аналога, а в особливості фамілію, ім’я та по батькові, ініціали, короткі відомості та побажання до використання при створенні розкладу занять.

«Предмет» - сутність, що використовується у програмній системі для представлення реального предмету. Володіє інформацією про реальний аналог, що містить в собі назву предмету, особливості використання при створенні розкладу занять та інформацією про аудиторію, що використовується зазвичай при формуванні розкладу.

«Аудиторія» - сутність, що використовується у програмній системі і є сутністю найнижчого рівня. Володіє інформацію про назву аудиторії і короткі відомості до використання.

Програмна система володіє функціоналом для створення графіку навчального процесу для навчального закладу. Графік навчального процесу володіє представлення у програмній системі у вигляді однойменної функції. Сутність включає в себе інформацію про момент створення, автора, коментарі до створеного розкладу занять, період часу на який графік навчання є актуальним і елементи графіку навчання.

«Елемент графіку навчального процесу» - кожен графік навчального складається набору цих елементів. Дані елементи володію інформацією про групу, що приймає участь у графіку навчального процесу, а також набором із 52 тижнів.

«Тиждень» - сутність, що використовується у програмній системі і володіє фіксованим набором значень, які вона може приймати. Значення тижню визначає роботу, що буде проводитися у цей яка до якого він призначений.

Програмна система окрім сутностей, які представляють собою дані програмної системи володіє набором методів та функції.

Додавання, редагування, видалення інформації – ключові функції для заповнення програмної системи даними. Дані функції можуть використовуватися користувач для заповнення програмної системи інформаціє про групи, викладачів, предмети, аудиторії, вихідні дні та не робочі дні які будуть використовуватися при створенні розкладу занять та графіку навчального процесу.

Створення графіку навчального процесу – комплексна функція, що використовує всю надану користувачем інформацію для прискорення та автоматизації процесу створення графіку навчання. Результатом виконання функції може бути програмне представлення графіку або виведення його у інші доступні формати.

Створення розкладу занять – комплексна функція програмної системи, що використовує всю надану користувачем інформацію для прискорення виконання завдання створення розкладу занять. Надає вихідну інформацію у вигляді аналітичної інформації про виконане завдання, розклад занять у представлені програмної системи або в інших доступних форматах документів.

**3.6 Огляд аналогів**

При аналізі предметної області теми дипломного проекту важливо зверну увагу на існування аналогів системи, що розробляється. Огляд аналогів дозволяє дізнатися про стан ринку в даній темі, рівень вирішення проблеми аналогами системи та методи вирішення цієї проблеми. На ринку існує декілька аналогів програмної системи, що розробляється і надалі будуть назватися лише найбільш відомі або найбільш успішні проекту, що володіють повним функціоналом і можуть виступати як аналоги.

Програмна система Ректор-Коледж призначена для створення розкладу занять в коледжах. Володіє набором функції для створення списків, навантажень, розкладу і замін.

Програма поділена на розділи кожен з яких володіє своїм функціоналом і використовується для досягнення загальної мети. Розділ «списки» слугує для введення, редагування і друку списку спеціальностей, груп, предметів, аудиторій, викладачів. Розділ «Навантаження» використовується для введення, редагування навчальних планів по спеціальностям, графіків розділення часу по тижням в межах семестрів. Розділ «Розклад» призначений для створення розкладу по групам, викладачам, аудиторіям і навчальному закладу в цілому. Розділ «Заміни» дозволяє вносити зміни викладачами.

Розклад занять можна створювати в ручному та напівавтоматичному режимі з можливістю переходу од одного режиму до іншого. При створенні розкладу програма аналізу всі вимоги і видає інформацію про найкраще створення розкладу занять. При створені розкладу занять програмна система переглядає вимоги викладачі і на їх основі створює зауваження та підказки.

Готовий розклад занять можна зберегти у форматі Microsoft Word, Excel, HTML.

Вимоги до виконання системи: програмна система працює на будь-якому комп’ютері з операційною системою Windows XP, Windows Vista, Windows 7.

Програма Ректор-Колеж володіє наступним інтерфейсом поданим на рисунку 1.

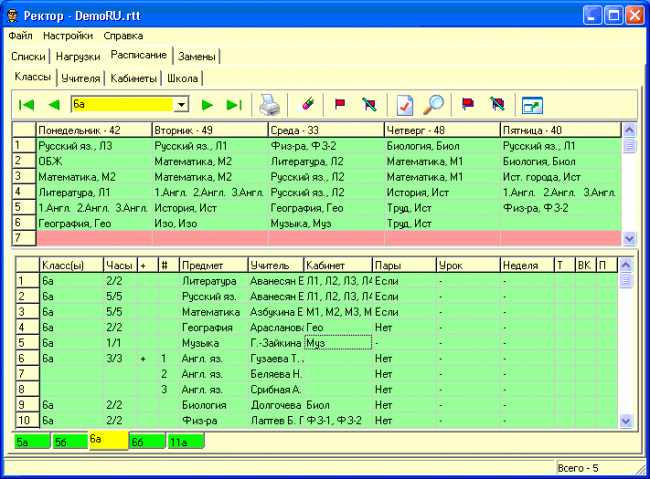


Рисунок 1 — Інтерфейс програми Ректор-Коледж

Програмна система володіє недоліками, що виявляються у відсутності підтримки з боку розробників. Остання версія програми була створена у 2015 році. Відсутність підтримки виявляється у неможливості роботи з новими операційними системи. Ще не мало важливо дана програмна система може працювати лише на деяких версіях операційної системи Windows. Цей фактор складається разом з фактором великої вартості. Мінімальна ліцензія для навчального закладу є 5000 гривень, що в сукупності з вартістю ціни ліцензії на операційну систему Windows, що на момент створення дипломного проекту складає 2500 гривень є захмарною ціною для звичайного навчального закладу.

Програмний продукт, що розробляється може використовуватися на операційних системах Windows, Linux, macOS. Досить розширених рівень підтримки надає можливість використовувати на тій операційній системі, яка вже наявна. Варто зауважити, що операційні системи сімейства Linux є повністю безкоштовними.

Навчальна екосистема MySchool.ua, що додатково володіє набором функції для автоматизації створення розкладу занять школи. В програмну систему потрібно занести дані про предмети, викладачів, аудиторії, класи, заповнити вимоги про графіку навчання і програма буде видавати підказки під час створення розкладу занять.

Дана екосистема призначення для повної автоматизації життя школи. Всі процеси відбуваються через неї. Програмна система особливо зручна, якщо уже існує робочий розклад занять у паперовому або електронному варіанті. Потрібно лише занести всі дані у систему і почнеться контроль та автоматизація роботи.

Система володіє набором шаблонів для створення розкладу занять. Шаблон володіє узагальненою інформацію, яку потрібно конфігурувати для нормального функціонування. Додатково система гарантує високу швидкість складання розкладу занять та реагування на зміни в навколишньому середовищі. Система враховує режими роботи викладачів, режими роботи класів, можливість створення підказок по зменшенню вікон у викладачів, вимоги по проведенню уроків у визначених кабінетах, володіє та використовує інформацію про вміст кабінету, підтримка тижневого навантаження.

Система володію інтерфейсом, що поданий на рисунку 2.

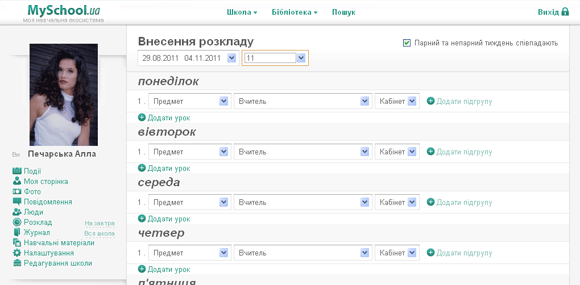


Рисунок 2 - Інтерфейс системи MySchool.ua

Недоліки системи для використання її у навчальному закладі для якого виконується створення дипломного проекту:

* Програмна екосистема встановлюється лише повністю і потребує багато зусиль на її встановлення та перенесення всього життя навчального закладу в неї
* Програмна система створена для шкіл і володіє досконалістю у реалізації для використання її у вищих навчальних закладах
* Висока вартість використання для навчального закладу такого типу

Продукт «Галактика Розклад навчальних занять» представлений корпорацією Галактика призначений для автоматизації процесу створення розкладу занять для студентів у вузах, технікумах, коледжах та ПТУ. Система володіє функціоналом для допомоги співробітникам освітніх установ, що відповідаю за складання розкладу занять навчальних занять.

Продукт «Галактика Розклад занять» включає в себе технології автоматизації формування розкладу занять, що зменшують вплив «людського фактору» на процес створення.

Програмна система контролює роботу працівника, що використовує продукт для створення розкладу занять і не допускає створення накладок і контролює, щоб всі вимоги і чинники були враховані. Також елементарно виключає помилки в розкладі занять. Процес створення проходить у декілька етапів:

* Етап підготовки – програма збирає всі дані
* Етап планування – програмна збирає всі вимоги для результату
* Етап налаштування – виконується налаштування пріоритетів для результату виконання
* Етап формування – відбувається створення розкладу з підказками і дотриманням заданих вимог
* Етап оптимізації – процес доповнення розкладу тією інформацією, що не була врахована раніше.

Програмна система володіє інтерфейсом поданим на рисунку 3.

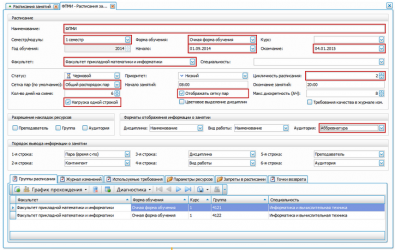


Рисунок 3 - Інтерфейс системи "Галактика Розклад занять"

Недоліками використання є висока вартість придбання системи та її інтеграції в робочий процес.

**4 ОПИС СИСТЕМИ**

**4.1 Структура системи**

Програмна система дипломного проекту побудована з дотриманням об’єктно-орієнтованої схеми проектування систем. Під час проектування використовувалися шаблони об’єктно-орієнтованого програмування, що подані в книзі «Патерни проектування» виданої Gang of Four. Схема роботи програми відповідає схемі поданій на рисунку 3.

**Десь тут рисунок**

Рисунок 3 – Схема роботи програмної системи

Подана схема повністю відповідає тій, за якою відбувається робота з програмною. Варто зазначити, що програма використовує шаблон при якому перехід до компонентів відбувається по «закладках». Такий шаблон був обраний не даремно. Під час розробки були спроби використовувати інша шаблони роботи системи, але використання такого шаблону дозволяє максимально продуктивне її використовувати. Більше того використання саме цієї схеми дозволяє використовувати програму для одночасного виконання декількох задач.

Програмна система побудована на спеціально розробленій об’єктно-орієнтовані структурі. Загальна структура класів для даних подана на UML діаграмі і має вигляд поданий на рисунку 4.

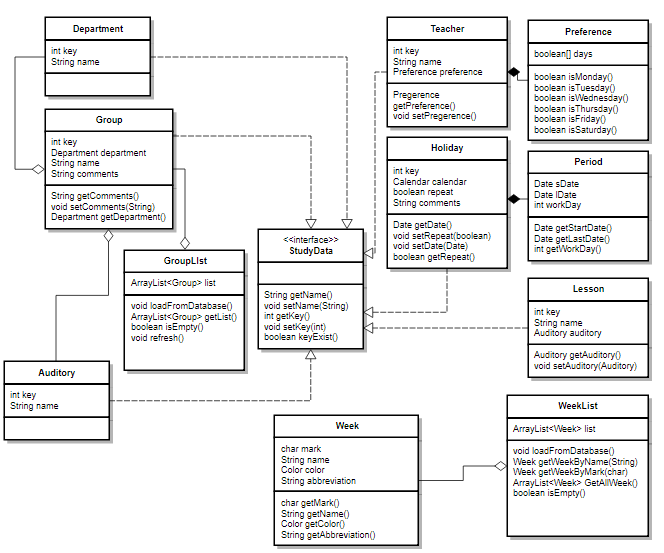


Рисунок 4 – Діаграма класів для роботи з даними

Структура класів потребує розгорнутого пояснення роботи та призначення компонентів, що подані нижче в алфавітному порядку.

«Auditory» - клас, що є абстрактною сутністю і являє собою аналог реальної аудиторії. Володіє інформаціє про себе, що містить назву аудиторії та ключ. Реалізує інтерфейси StudyData та Comparable<Auditory>. Містить методи читання інформації та методи модифікатори. Реалізує методи StudyData, що вимагають наявність ключа в об’єкті та методи Comparable<Auditory>, що вимагають наявність спеціального методу compareTo() для порівняння двох об’єктів одного типу.

«DataModifyInterface» - інтерфейс для роботи діалогів модифікації даних. Об’єкт, що реалізує даний інтерфейс зможе існувати в програмі і володітиме набором функції для додавання, редагування та видалення даних власного типу в програмі. Даних інтерфейс у спеціальні формі реалізації використовують класи Auditory, Teacher, Group та Lesson.

«Department» - клас, що є абстрактною сутністю для реального існуючих відділень навчального закладу. Реалізує інтерфейс StudyData, що дозволяє об’єктам класу використовуватися в програмній системі і працювати з базою даних.

«Group» - клас, що абстрактною сутністю для реальної існуючої групи. Володіє інформацією про групу, яку представляє та об’єктом класу Department, що позначає до якого відділення відноситься група. Реалізує інтерфейси StudyData та Comparable<Group>. Призначення та використання інтерфейсів описано вище. Відмінність у використанні Comparable<Group> покладається у тому, щоб він використовуватися лише з самими собою або іншими представниками цього класу.

«GroupList» - клас-служба, використовується для надання інформації для всіх інших класів. Володіє інформацією про наявні групи в базі даних і може надавати інтерактивну інформацію про групи. Володіє набором методів для вибірки даних за певним критерієм.

«Lesson» - клас-сутність, що є абстракцією від навчального предмету реального світу. Володіє інформацією про самого себе, а також інформацією про аудиторію, що буде використовувати по замовчуванню. Реалізація відбувається через наявність внутрішнього посилання на об’єкт Auditory. Реалізує інтерфейси StudyData та Comparable<Auditory>. Робота інтерфейсів описувалася вище.

«Period» - клас, що використовується як тип даних для роботи програми. Володіє інформацію про дві дати і кількість робочих днів між цими датами. Володіє методи для створення екземплярів класу у правильному форматі та набір функції для роботи з періодами у програмі.

«Preference» - клас, що використовується як тип даних для роботи програми в особливості використовується для збереження даних про вимоги викладачів по використанню під час створення розкладу занять.

«StudyData» - інтерфейс, що забезпечує існування у всіх класів нащадків методів для роботи з ключами та назвами. Використовується для забезпечення поліморфізму у роботі з даними програми. Даний клас реалізують Teacher, Lesson, Auditory та Department.

«Teacher» - клас, що є абстрактною сутністю для представлення у програмі викладача реального світу. Володіє інформацією про викладача, а також вимогами викладача по використанню під час планування розкладу занять. Реалізовує інтерфейси StudyData та Comparable<Teacher> призначення яких описуються вище.

«Week» - клас, що є абстрактною сутністю для представлення у програмній системі тижня навчального процесу. Володіє інформацією про власне існування, використання та призначення. Програмна система володіє чітко фіксованим набором класів даного типу по причині, що дана інформація не змінюється так як часто як інша подана раніше. Додання нових тижнів можливе через спеціальну форму в налаштуваннях системи. Використовується для позначення типу тижня у графіку навчального процесу.

«WeekList» - клас-служба, що використовується іншими об’єктами програмної системи для отримання інформації про тижні в базі даних, або отримання розширеної чи спеціальної інформації про якийсь із тижнів. Даний клас може працювати з базою даних і підтримує автоматичне оновлення інформації у випадку її зміни.

Окрім загальний класів, які зазвичай лише зберігають інформацію в програній системі використовується класи, що мають складну структуру даних і володію функціоналом для роботи з вище зазначеними класами. Такі класи поділяються на два типи: класи для роботи з розкладом занять та класи для роботи з графіком навчання. Додатково існує, ще один клас служба, що не входить в жодну структуру класів і використовується для роботи з базою даних.

Структуру класів, що працюються з розкладом занять подано у вигляді UML діаграми на рисунку 5.

**UML діаграма класів для роботи з розкладом занять**

Рисунок 5 – Діаграма класів для роботи з розкладом занять

Структура класів вимагає пояснення до внутрішнього влаштування та роботи. Далі подано пояснення для кожного зображеного класу в порядку важливості.

«LessonsUnit» - клас, що використовується як контейнер для збереження даних. Використовується розкладом занять для збереження даних про групи та інформацію про розклад для кожної із груп. Даний клас є складним і містить в собі інформацію про групу для якої використовується та набір StudyPair кількість яких залежить від використовуваної схеми розкладу занять, тобто кількості робочих днів у тижні та кількості годин, що є робочими для кожного дня. Даний клас реалізовує інтерфейс Comparable<LessonsUnit> для можливості сортування елементів розкладу занять під час виконання програмної системи.

«StudyPair» - абстрактний клас, що використовується як загальний інтерфейс для всіх можливих складових елементу LessonsUnit. Використовується для можливостей поліморфізму та масштабування системи. Абстрактним даний клас є через потребу у використанні зареєстрованого в програмній системі пустого об’єкта EmptyStudyPair, якого може створювати лише даний клас. Додатково даний клас володіє внутрішніми класами перечисленнями:

* Query – перечислення, що володіє значеннями LESSON, TEACHER, AUDITORY. Використовується для роботи розкладу занять і встановлення вимоги по відображенню інформації
* Forbidden – перечислення, що володіють значення для позначення правил роботи програми під час редагування розкладу занять. Володіють в основному правила заборони на використання в певний час або в певному місці. Перечислення володію багатьма значенням для прикладу приводиться лише декілька з них: заборона по дню використання, заборона використання в один час з поточним елементом, заборона використання по в поточний тиждень та інші.

Клас додатково володіє набором функції для роботи з перечисленнями поданими вище.

«EmptyStudyPair» - константний клас призначений для використання як відсутність даних в розкладі занять. Використовується, щоб уникнути посилання на null, що володіє рядом негативних наслідків. Реалізовує клас StudyPair, хоча і володіє пустою реалізацією всіх методів.

«StudyPairLonely» - клас, що є абстрактною моделлю предмету з одним предметом. Володіє інформацією про предмет, викладача і аудиторію, що використовуються. Інформацію зберігається як посилання на об’єкти Lesson, Teacher, Auditory. Володіє набором методів для обробки інформації зв’язаною з екземплярами цього класу. Являється реалізацію абстрактного класу StudyPair.

«StudyPairDouble» - клас, що використовується рішення задачі з використання змінних предметів, що залежать від порядку тижня в семестрі. Внутрішня структура володіє двома об’єктами StudyPairLonely, володіє набором методів для роботи з екземплярами цього класу. Для програмної системи зазвичай представляє себе як StudyPairLonely і виконує всі ті ж самі функції.

«LessonTableModel» - клас, що є представлення розкладу занять у програмній системі. Володіє всією інформацією для відображення, контролю, аналізу, форматування та збереження даних. Внутрішня структура володіє набором елементів LessonsUnit, що визначають його внутрішню структуру. Являється нащадком AbstractTableModel із внутрішньої структури класів мови програмування. Використовується для можливості роботи із вистроєними компонентами.

Для роботи процесу створення та використання графіку навчального процесу використовується окремо створений набір класів. Структура та робота класів подана на UML діаграмі на рисунку 6.

**Тут має бути діаграма класів графіку навчання**

Рисунок 6 — Діаграма класів графіку навчального процесу

Структура класів потребує пояснення схеми роботи. Пояснення подано далі у по черговості важливості роботи класів.

«ScheduleUnit» - клас, що представляє собою головну одиницю яка використовується у графіку навчального процесу. Володіє інформацією про групу, а також про 52 робочих тижні. Групу представляє об’єкт Group, а для збереження даних про робочі тижні використовуються об’єкти Week. Даний клас володіє набором функцій для роботи з екземплярами класу. Також даний клас реалізовує інтерфейс Comparable<ScheduleUnit>, що дозволяє виконувати співставлення подібних елементів та визначення їх в структурі.

«SchedulerTableModel» - клас, що є представлення графіку навчального процесу в програмній системі. Володіє інформацією про період для якого він є актуальним та набір елементів ScheduleUnit, які володіють всією іншою інформацією. Даний клас володіє набором методів для виведення графіку навчального процесу на екран у вигляді таблиці, для роботи з базою даних, що виявляється у збереженні, оновленні та видаленні інформації, а також для можливості виведення інформації в інші формати даних. Являється нащадком класу AbstractTableModel із внутрішньої структури класів мови програмування.

Для роботи системи використовуються додаткові класів, які не можна визначати до жодної описаних вище категорій. Дані класи подані у вигляді UML діаграми, що подана на рисунку 6.

**Тут мають бути додаткові класи**

Рисунок 7 — Діаграма додаткових класів

«DatabaseData» - клас-служба, використовуються як незалежний від програмної системи клас і володіє прямим зв’язком з базою даних. Виконує роль посередника між базою даних та всіма класами програмної системи. Всі класи в момент потреби отримання доступу до бази даних використовую його для зв’язку. Схема використання на відмінно від всіх інших об’єктів не потребує посилання для його використання.

«DegreeProject» - основний клас програми, виконує роль рушія для ініціалізації всіх елементів і запуску програмної системи. Здійснює постійний контроль за станом роботи системи і у випадку порушення буде намагатися здійснити відновлення або максимально м’яке відключення системи з можливістю подальшого відновлення.

**4.2 Розробка системи**

Переглядаючи структуру програми можна зрозуміти, що вхід у програмну систему відбувається через вікно завантаження даних. Вікно відображається найпершим при запуску програми і покликане створити бренд програмної системи, а також розважити користувача під час виконання процесу завантаження. Процес завантаження залежить від платформи на якій відбується запуск, а також від пропускної здатності каналу з’єднання з інтернетом. В момент успішного завантаження всіх потрібних для роботи даних у пам’ять вікно закривається і відкривається вікно з’єднання з базою даних. Вікно завантаження створювалося не в режимі конструктора, а генеруються в процесі роботи системи.

Вікно завантаження володіє власним класом IntroFrame, що є нащадком від стандартного класу мови програмування JFrame і володіє всіма його методами. Вікно складається з елементів:

* JPanel – використовується як контейнер для всіх інших компонентів вікна і визначає схему розташування внутрішніх компонентів відносно менеджера, який використовується. У даному вікні, як і більшості інших, використовується GridLayoutManager.
* ImagePanel – компонент, що був створений для роботи саме у формі завантаження. Наслідує JPanel. Використовується для відображення графіки і стрічки прогресу завантаження. Володіє набором методів для роботи зі стрічкою завантаження.

Під час роботи клас у внутрішній структурі використовує для роботи наступні компоненти:

Image – абстрактний клас-контейнер для збереження та роботи зі зображеннями. Не володіє реалізацією методів, а лише являє собою інтерфейс для створення реалізацій.

BufferedImage – одна із реалізацій абстрактного класу Image. Дані реалізацію використовується через швидкість роботи і простоту у використанні. Інші реалізації володіють більш складною реалізацію для можливості використання на серверах чи в Інтернет системах

Graphics – клас, що володіє набором інструментів для роботи з графічними об’єктами. Може використовуватися для програмного малювання простих об’єктів

Graphics2D – клас, що володіє розширеним набором інструментів для роботи з графікою відносно Graphics. Може використовувати для створення складних графічних об’єктів.

LinearGradientPaint – клас, що використовується для роботи з градієнтами. Володіє набором функції для створення складних лінійних градієнтів. Використовується в програмній системі для створення полоси завантаження.

Створення полоси завантаження відбується кожну ітерацію оновлення вікна за допомогою наступного методу в програмному коді:

public void paint(Graphics g) {

super.paint(g);

g.drawImage(introImage, 0, 0, null);

Graphics2D g2 = (Graphics2D) g;

gradientPaint = new LinearGradientPaint(

minPosition.x, minPosition.y,

minPosition.x + (int) ((maxPosition.x - minPosition.x) \* (now / ((max - min) \* 1f))),

maxPosition.y,

fractions,

colors

);

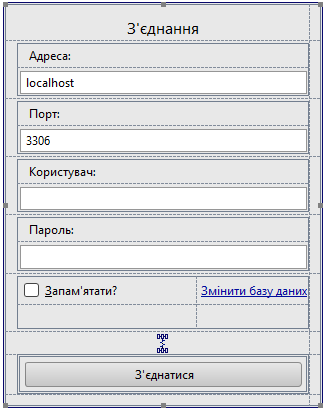
g2.setPaint(gradientPaint);

g2.fillRoundRect(minPosition.x, minPosition.y, (int) ((maxPosition.x - minPosition.x) \* (now / ((max - min) \* 1f))), maxPosition.y - minPosition.y, 4, 2);

}

Де introImage це зображення, що буде відображатися; minPosition, maxPosition, min, max, now – це перемінні, що зберігають поточні налаштування для відображення полоси завантаження; fraction – це масив, що зберігає значення положення ключових кадрів; colors – це масив з кольорів, що використовуються при створені градієнту.

Під час роботи вікна завантаження відбувається створення вікна з’єднання з базою даних, а також відбувається перевірка на наявність раніше збережених даних для підключення. Коли форма готова до відображення і дані або відсутні або за ними не можливо виконати під’єднання відбується відображення форми з’єднання. Форма з’єднання в режимі конструювання володіє наступним виглядом поданим на рисунку 8.



**Рисунок 8 — Форма з’єднання з базою даних в режимі конструктора**

Форма представлена у програмі класом, що має назву ConnectionForm, який є нащадком JFrame. Для роботи використовує наступні компоненти:

JTextField – клас-компонент, який володіє набором методів, щоб використовуватися як поле для введення даних. Використовується для зчитування інформації про підключення до бази даних.

JPasswordField – клас-компонент, що представляє із себе поле для введення як JTextField, але володіє функціоналом для роботи з паролями. Цей функціонал полягає в тому, що при введені дані закриті від користувача і зловмисника, також після введення даних відсутня можливість їх копіювання чи отримання. Також в програмі значення, що було введене передаються у форматі масиву байтів, що унеможливлює їх спотворення, вони перебувають у званому «сирому» вигляді.

JButton – клас-компонент, що володіє функціоналом звичайної кнопки. Використовується для зчитування дії натискання користувачем у певній області екрану.

JCheckBox – клас-компонент, що представляє із себе перемикач між двома станами. Використовується для передачі від користувача інформації по типу «так або ні».

JLabel – клас-компонент, що використовується для виведення статичної текстової інформації користувачеві.

JPanel – клас-контейнер, функціонування якого описувалося при описуванні роботи попередньої форми.

Також для роботи форми використовувалися класи типу Listener (слухачі) та Adapter (адаптери) описані в структурі системи, як шаблони проектування. Використовується ActionListener, MouseListener, KeyListener, що є інтересами для отримання інформації про відповідні події, що сталися в програмній системі. Для обробки отриманої інформації використовуються KeyAdapter, MouseAdapter, що володіють потрібних функціоналом.

Для обробки файлів, що використовуються для зберігання інформації про минулі підключення для бази даних використовуються наступні компоненти:

File – клас, що містить набір інструментів для роботи з файлами та файловою системою платформи на які виконуються робота системи. Використовується для з’єднання файлу і програмної системи і подальшого отримання інформації.

FileInputStream – клас, що створює File для передачі даних і файлової системи у програму. Володіє потоком необроблених байтів у тому вигляді, як вони зберігаються на диску. Для роботи з файлом у програмі використовуються шаблонні класи декоратори, які розширяють функціонал початкового класу.

InputStreamReader – один із класів декораторів, які виконує перетворення FileInputStream у потік даних з яких можливе зчитування даних.

BufferedReader – один із класів декораторів, використовується для можливості масового зчитування даних. Даний клас володіє функціоналом, що прискорює зчитування даних з диску і спрощує роботу програмісту для роботи з даними.

URI – клас, що надає можливості звичайного URI (Uniform Resource Identifier). Використовується для вказання позиції розташування файлу на диску.

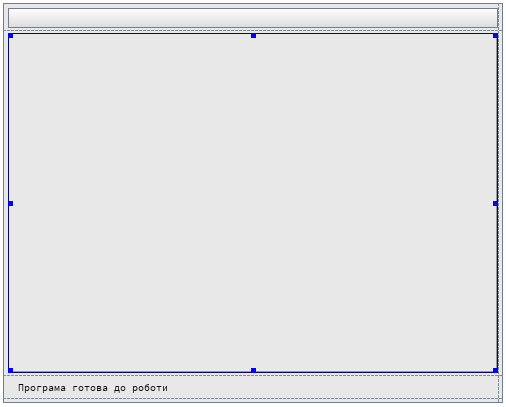
Для підключення і роботи з базою даних використовуються класи із пакету JDBC (Java Database Connection). Даний пакет володіє цілим набором інтерфейсів і класів для роботи з різнотипними базами даних. В даному випадку використовуються наступні компоненти:

Connection – клас, що виконує роль утримувача з’єднання між базою даних і програмною системою. Володіє набором функції для управління з’єднанням. Connection може створювати елементи Statement.

Statement – виконує роль контейнера для зберігання запитів до бази даних та їх виконання. Володіє набором методів для управління запитам. Statement зазвичай використовується для створення простих запитів до бази даних без вхідних параметрів. Після виконання запиту видає в результаті ResultSet.

ResultSet – вихідних результат виконання запиту до бази даних. Володіє множиною записів таблиці в певній структурі. Володіє набором методів для опрацювання результатів запиту, а також для навігації по стрічкам вихідної таблиці.

У випадку під’єднання до бази даних і якщо база даних відповідає структурі, що вимагається для роботи відбувається перехід до основного робочого середовища, відбувається запуск головної форми програми. Головна форма програми є лише контейнером, а вся робота відбувається у спеціальних вкладках. Структуру головної форми в режимі конструктора можна переглянути на рисунку ?.



**Рисунок 9 — Головна форма програми**

В режимі конструювання форма виглядає порожньою по тій причині, що майже всі елементи створюються під час запуску програмної системи. Даний шаблон головної форми володіє наступними компонентами:

JToolBar – компонент, що використовується для створення панелі інструментів програмної системи. Даний компонент використовується як контейнер для JButton, JToolBar.Separator та Box.HorizontalGlue. JButton нічим не відрізняються від звичайної кнопки, а елемент JToolBar.Separator використовується для створення візуального розділення груп елементів JButton. Box.HorizontalGlue використовуються як заповнювач вільного простору. Він використовується для того щоб всі інші елементи займали мінімальний простір в панелі інструментів.

JTabbedPane – компонент-контейнер, що містить набір методів для збереження елементів у вигляді вкладок. Також містить методи для роботи з вкладками. Вкладками можуть бути будь-які об’єкти класу JPanel або його нащадки. Для підпису використовується або назва об’єкту, або спеціальний компонент TabComponent, який буде використовуватися як заголовок для вкладки.

Для роботи головної форми був створений спеціальний клас MainForm, що наслідує JFrame. Даний клас визначає структуру та методи, які може виконувати головна форма. Цей клас включає до себе наступні елементи:

MainFormMenuBar – спеціально створений клас, що представляє собою головне меню для головної форми. Даний клас є нащадком до JMenuBar. У своїй внутрішній структурі використовує елементи JMenu, JMenuItem та JPopupMenu.Separator. JMenu виступає в ролі елементів меню вищого рівня і може містити в собі підменю. Як підменю використовуються елементи JMenuItem. Дані елементи не можуть містити в собі інші елементи, а лише дію яку вони виконують і назву. Для розділення груп елементів використовуються JPopupMenu.Separator. Дані елементи не виконують ніякої дії окрім як створення візуального розділення елементів головного меню.

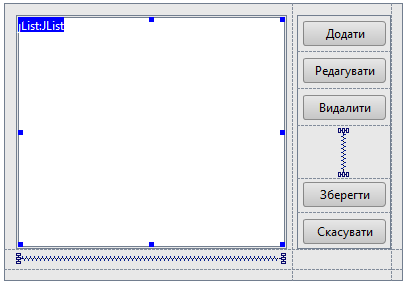
Під час першого запуску головного вікна відбування додавання закріпленої вкладки з короткими відомостями про програму, а також з кнопками швидкого доступу до компонентів. Роль вітальної вкладки виконує клас HelloPanel, що є нащадком JPanel. Внутрішня структура складається з набору кнопок і зображень. В режимі конструктора форма має вигляд поданий на рисунку 10.



**Рисунок 10 – Вітальна форма в режимі конструктора**

Дані форма володіє складається з компонентів JPanel, JButton, JLabel, які описувалися вище.

Далі по порядку зустрічі користувача з формами ідуть форми заповнення даними програмну системи. Для заповнення даних була створена спеціальна універсальна форма, що підходить для використання майже під будь-які типи даних. В режимі конструктора форма наступний вигляд поданий на рисунку ?.



**Рисунок 11 – Форма модифікації даних**

Форма модифікації даних складається з наступних компонентів: JButton, JPanel, JList.

JList – компонент, що виступає в ролі списку з вибором одного елементу. Надає можливість користувачеві виконувати навігацію по списку, якщо кількість елементів більша, ніж може вміститися у виділений області. А також надається можливість переходу до елементу по перший букві за допомогою введення її з клавіатури.

Всі елементи, які використовуються в формі для модифікації даних мають реалізувати інтерфейс, що має наступний вигляд:

public interface DataModifyInterface {

StudyData add();

StudyData edit(StudyData t);

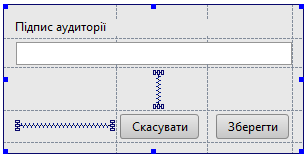
boolean remove(StudyData t);

void exit(StudyData[] t);

}

Методи даного інтерфейсу будуть викликатися у випадку натиснення на відповідні кнопки на формі. У випадку закриття вікна передача всіх елементів здійснюється у метод exit(); для можливості обробки інформації після її зміни. Для реалізації додання та редагування даних більшість даних використовуються спеціальні форми обробки даних.

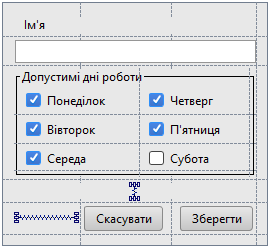
Форма додавання та редагування даних «Аудиторій» володіє наступних виглядом у режимі конструктора поданого на рисунку 12.



**Рисунок 12 – Діалог модифікації даних аудиторій**

Даний діалог викликається при натисненні на кнопки додати або редагування. Відмінності полягають в тому, що при додаванні відкривається пустий діалог, а при редагуванні відкривається уже заповнений даними для редагування. Даний діалог наслідує клас JDialog, що надає можливості блокувати роботу програми до виконання дій саме в цьому діалозі. Використовує компоненти JLabel, JPanel, JTextField, JButton. Робота компонентів описувалася вище.

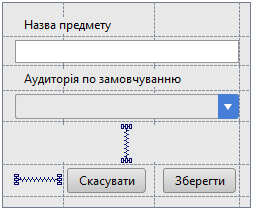
Форма додавання та редагування даних «Викладачів» в режимі конструктора володіє наступним виглядом поданим на рисунку 13.



**Рисунок 13 – Діалог модифікації даних викладачів**

Діалог модифікації даних викликається в момент натиснення користувачем на кнопки додавання або редагування даних при роботі з Викладачами. При додаванні нових даних діалог запускається з пустим іменем, а всі інші параметри вибрані такі, які по замовчуванню будуть використовувати, якщо користувач не встановить інше. Для роботи діалогу був створений клас TeacherDialogModify, що унаслідується від вбудованого класу JDialog. Виклик по замовчуванню відбується в модальному режимі. Для роботи даний клас використовує наступні компоненти: JPanel, JButton, JTextField, JCheckBox, JLabel. Всі використовувані компоненти описували раніше.

Форма для додавання та редагування даних «Предметів» в режимі конструювання володіє наступним виглядом поданим на рисунку 14.

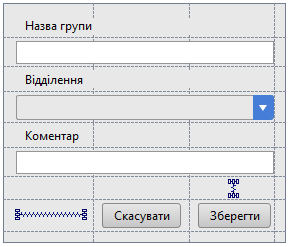


**Рисунок 14 – Діалог роботи з «Предметами»**

Даний діалог викликається у момент натиснення користувачем на одну із кнопок додавання чи редагування даних при роботі з «Предметами». Для роботи даного класу був створений спеціальний клас LessonDialogModify. Даний клас унаслідується він вбудованого класу JDialog, що використовується для надання всі функції діалогу, а в особливості роботи в модальному режимі. Для роботи використовуються наступні компоненти: JPanel, JLabel, JTextField, JButton, JComboBox.

JComboBox – це компонент, що слугує як випадаючий список. Дозволяє збільшити надійність і правильність введених користувачем даних. Використовується в даному випадку для вибору аудиторії по замовчуванню. Для даного діалогу реалізований пошук за введеною з клавіатури маскою можливого тексту.

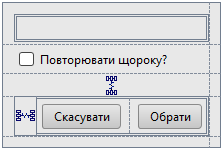
Форма додавання та редагування даних існуючих груп в режимі конструювання володіє наступним виглядом поданим на рисунку 15.



**Рисунок 15 – Діалог модифікування груп**

Діалог використовується для спілкування з користувачем і отримання від нього даних про групи. Користувачеві надається можливість ввести назву групи, обрати відділення із запропонованих та ввести коментар. Відділення володіє чітко фіксованим набором значень, але в програмній системі існує можливість додання або зміни навіть цих даних у випадку потреби. Коментарі можуть використовуватися для збереження службової інформації, що може бути потрібна пізніше при роботі з групою. Даний діалог підкріплюється у програмній системі класом GroupDialogModify, що унаслідується від вбудованого класу JDialog. Володіє наступними компонентами: JPanel, JLabel, JTextField, JButton, JComboBox.

Для модифікування даних про вихідні та не робочі дні в програмній системі був створений спеціальний діалог, що в режимі конструктора володіє наступним виглядом поданим на рисунку 16.



**Рисунок 16 – Форма редагування даних про не робочі дні**

Даний діалог володіє не повною структурою в режимі конструктора. Декілька компонентів створюються програмно через відсутність вбудованих шаблонів. Для роботи діалогу використовується клас HolidayDialogModify, що унаслідується від JDialog. Для роботи використовуються наступні компоненти: JButton, JCheckBox, JPanel, JDatePickerImpl, Calendar.

JDatePickerImpl – це спеціально розроблений клас, що унаслідується від JComboBox і використовується для вибору дати. Даний компонент був розроблений повністю з нуля і володіє унікальним інтерфейсом. Компонент генеруються програмним кодом, тому в режимі конструктора він не відобрається.

Calendar – клас, що володіє функціонал для роботи з часом в середовищі Java. Вхідним параметром для календарю є дата. Використовується для роботи з вибраною користувачем датою та конвертування її до дати sql.Date для подальшого збереження в базу даних.

Після заповнення бази даних необхідними даним для роботи користувач може приступати до створення першого графіку навчального процесу. Для створення графіку навчального процесу використовується панель SchedulesPanel, що в режимі конструювання володіє виглядом поданим на рисунку 17.

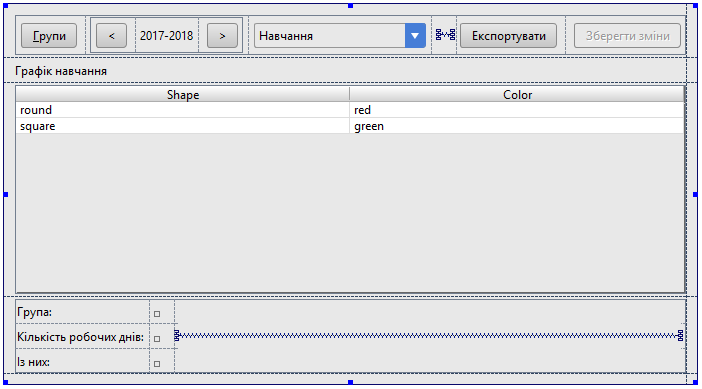


Рисунок 17 – Панель створення та редагування графіку навчального процесу

Дана панель володіє функціонал, що дозволяє створювати та редагувати графік навчального процесу. Роботу даної панелі забезпечує зв’язаний клас SchedulesPanel. Даний клас використовує такі компоненти як: JTable, JPanel, JButton, JLabel, JComboBox, SchedulerTableModel. Панель розширяє внутрішній клас JPanel.

JTable – це компонент, що надає функціонал таблиці в середовищі Java. Використовується для табличного відображення інформації. JTable був створений з використанням парадигми Модель-Вигляд-Контролер. Дана парадигма дозволяє розділити весь код на декілька великих функціональних блоків, код яких максимального відокремлений. Модель реалізується через інтерфейс TableModel, вигляд контролює реалізація класу TableCellRenderer, а контролер – TableCellEditor. Через використання таблиці лише для відображення, а зміна даних відбувається за допомогою зовнішніх контролерів, то реалізація TableCellEditor була замінена на стандартну реалізацію середовища. Для двох інших частин моделі були створені класи SchedulerTableModel та SchedulerTableCellRendererComponent.

SchedulerTableModel – клас, що реалізує модель таблиці, який зберігає в собі дані для відображення і в моменти відображення надає їх в SchedulerTableCellRendererComponent. Внутрішня структура складається з ScheduleUnit.

SchedulerTableCellRendererComponent – клас, що реалізує вигляд таблиці. Даний клас отримує дані від SchedulerTableModel і визначає як вони будуть відображатися на екрані.

Додатково для вибору файлу експорту даних графіку навчання використовується клас JFileChooser. Даний діалог є загальний для відкриття файлів і для вибору місця для збереження файлу.

Для цієї панелі було розроблено ряд діалогів, які використовується. Одним із діалогів є діалог вибору груп MultiChoiceDialog. Даний діалог володіє спеціальною структурою, що дозволяє вибирати не один, а одночасно декілька із запропонованих елементів, а також модифікувати раніше зроблений вибір. В режимі конструювання даний діалог має вигляд поданий на рисунку 18.

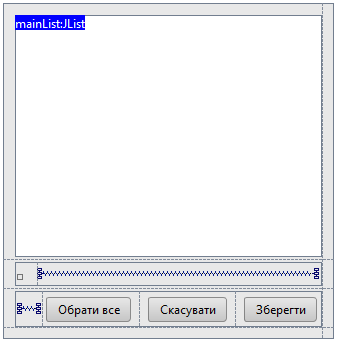


Рисунок 18 –Діалог роботи з групами

Діалог розроблений спеціальним чином, що із списку елементів можна було обирати декілька елементів. Кожен елемент додається до обраних натисканням клавіші миші і залишається обраним поки користувач знову по ньому не натисне. Для прискореного виділення або зняття виділення створена кнопка, що дозволяє виділити всі елементи або у випадку коли всі елементи вже виділені зняти виділення. Даний діалог використовує наступні компоненти для роботи: JList, JButton, JPanel, JLabel, DefaultListModel, DefaultListSelectionModel.

DefaultListModel – спеціальний клас, що використовується для забезпечення збереження та роботу з даними в JList. Використовується в даному діалозі через потребу використання не тільки класу String в MultiChoiceDialog, а і будь-яких інших типів даних.

DefaultListSelectionModel – через потребу у створені спеціальної моделі виділення об’єктів для роботи в цьому діалозі використовується модифікований варіант стандартної моделі виділення об’єктів в JList.

Додатково в даному діалозі використовується можливість дженерифікувати клас, тобто використовується структура при які до створення екземпляру класу не відомо якого типу буде один або декілька об’єктів.

В момент коли потрібно отримати доступ до графіку навчання, що був створений раніше і виконати експортування або редагування використовується діалог відкриття ScheduleChoiceDialog, що має вигляд поданий на рисунку 19.

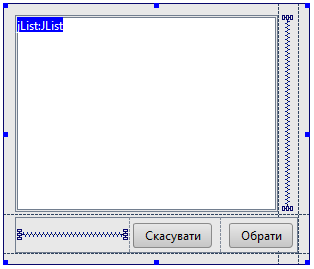


Рисунок 19 – Діалог відкриття графіку навчання

Даний діалог звертається до бази даних, перевіряє наявні графіки навчання і виводить їх список перед користувачем. В момент коли користувач вибере один із графіків навчання і натисне кнопку «Обрати» відбудеться завантаження всіх даних з бази даних і завантажаться до SchedulesPanel, що буде добавлена як нова вкладка на головному вікні. Компоненти, що були використані при створені: JPanel, JList, JButton. Форма унаслідується від класу JDialog.

Для видалення графіків навчання використовується діалог, що поданий в режимі конструювання на рисунку 20.

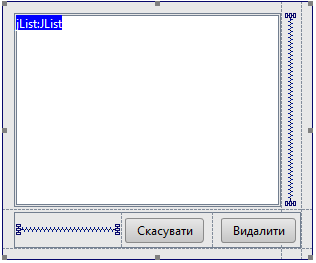


Рисунок 20 – Діалог видалення графіків навчання

Для створення та редагування розкладу занять використовується спеціально розроблена панель LessonsPanel. В режимі конструювання можна переглянути на рисунку 21.

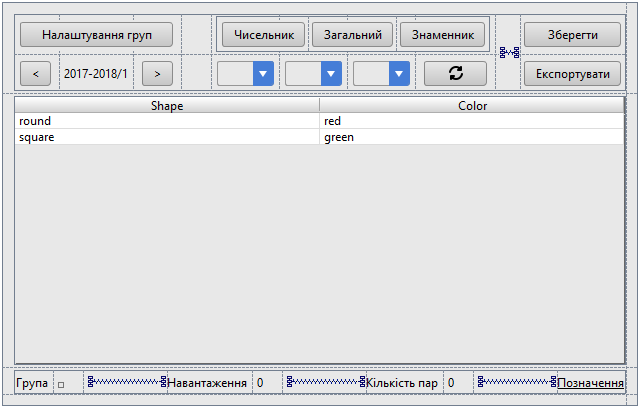


Рисунок 21 – Панель роботи з розкладом занять

Дана панель відкривається в той момент, коли користувач виявив бажання створити розклад занять. Панель володіє всім потрібним фунціоналом для створення та редагування розкладу занять. Роботу панелі забезпечує зв’язаний клас LessonsPanel. Клас панелі унаслідується від JPanel. Для роботи клас використовує наступні компоненти: JTable, JButton, JPanel, JToggleButton, JComboBox, JLabel, ButtonGroup, LessonTableModel, PopupMenu.

JToggleButton – це компонент, що є розширеною версією JButton і використовується для більше спеціальних цілей. Дана кнопка надає функціонал при якому якщо користувач натискає на неї і приймає курсор, то вона залишається в такому положенні. Для того щоб прийняти виділення потрібно ще раз натиснути. Дані кнопки використовуються для позначення і вибору користувачем поточного часу розташування заняття в розкладі занять.

ButtonGroup – компонент-контейнер, що використовується для групування взаємозв’язаних кнопок і розширяє функціонал взаємодії між ними. Використовується для групування трьох кнопок JToggleButton, щоб вибраною могла бути лише одна кнопка.

LessonTableModel – таблична модель, що використовується для збереження даних розкладу занять під час роботи з ним. Забезпечує збереження даних в базу даних та передачу даних на відображення. Для відображення було створено три класи кожен з яких відповідає за власних сегмент роботи. Даними класами є: TableHeaderCellRenderer, TableCellSubjectRenderer та TableCellPairNumberRenderer.

TableHeaderCellRenderer – клас, що визначає вигляд даних заголовка таблиці. Дані отримує від LessonTableModel і виконує їх обробку перед відображенням на екран.

TableCellSubjectRenderer – клас, що визначає зовнішній вигляд кожної комірки, що містить в собі заняття. Так як кожне заняття розділяється на три комірки, то додатково виконує компонування всіх елементів.

TableCellPairNumberRenderer – клас, що визначає зовнішній вигляд всіх інших компонентів, що існують в таблиці, але не використовуються іншими класами.

До відкриття існуючого розкладу занять і для видалення розкладу із бази даних використовуються схожі діалоги, що використовувалися при роботі з графіком навчання.

**Тут ще по суті має бути в режимі роботи**

**4.3 Структура бази даних**

Для роботи програмної системи використовувалася трансакційна реляційна база даних. Створення структури бази даних відбувалося відносно програмної системи та її потреб в реалізації. Структура бази даних була створена за допомогою мови SQL. Структура бази даних має вигляд поданий на рисунку 22.

**Тут має бути структура бази даних**

Рисунок 22 – Структура бази даних