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.

3

ТБЕК.188314.001 ПЗ

Розроб.

Поліщук В.П.

Перевір.

Харченко О.О.

Н. Контр.

Завірюха В.П.

Затверд.

Завірюха В.П.

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

Літ.

Акрушів

103

ПС-46

ВСТУП. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4

1 ПОСТАНОВКА ЗАДАЧІ. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .7

2 АКТУАЛЬНІСТЬ ЗАДАЧІ. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .9

3 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11

3.1 Огляд технічних проблем. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .11

3.2 Організація об’єктної моделі. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .12

3.3 Організація структури бази даних. . . . . . . . . . . . . . . . . . . . . . . . . . . .16

3.4 Організація програмної системи. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19

3.5 Огляд аналогів. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .21

4 ОПИС СИСТЕМИ. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 26

4.1 Структура системи. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 26

4.2 Розробка системи. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 34

4.3 Структура бази даних. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .51

4.4 Робота системи. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 54

5 ВИБІР ЗАСОБІВ РЕАЛІЗАЦІЇ. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 58

5.1 Операційна система. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .58

5.2 Мова програмування і середовище розробки. . . . . . . . . . . . . . . . . . .59

5.3 Система керування базами даних. . . . . . . . . . . . . . . . . . . . . . . . . . . . 61

5.4 Необхідні вимоги до конфігурації ЕОМ. . . . . . . . . . . . . . . . . . . . . . .62

6 ОХОРОНА ПРАЦІ. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .64

ВИСНОВКИ. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .67

СПИСОК ЛІТЕРАТУРИ. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .70

СПИСОК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ. . . . . . . . . . . . . . . . . .71

Додаток А Специфікація. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .72

Додаток Б Текст програми. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 74

Додаток В Інструкція користувачеві. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .90

Реценз.

**ВСТУП**

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

4

ТБЕК.188314.001 ПЗ

Дипломний проект на тему «Програмна система для автоматизації складання розкладу занять Тальнівського будівельно-економічного коледжу УНУС» має на меті розробку такої програмної системи, яка 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 надає 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, яка 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 для 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 даних 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.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

5

ТБЕК.188314.001 ПЗ

Виконання дипломного проекту, окрім основного завдання, вирішує додаткові завдання, що мають вплив на інші роботи. Дипломний проект виконує свої функції розробки. Створення розкладу занять володіє набором функції для розробки графіку навчального процесу, а також володіє функцією аналізу та обробки інформації, що надає інформативний характер, виконує перевірку розкладу занять і графіку навчання на правильність виконання.

Програмний продукт, що розробляється володіє наступними технічними характеристиками:

* Адаптивність – забезпечує найбільшу зручність у використані на різних робочих станціях.
* Багатоплатформність – 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 зберігаються 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.
* Масштабність – 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 затрати часу та ресурсів трудового колективу на виконання завдання по створенню розкладу занять. Зменшення ресурсів дозволять використовувати 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, а також зменшить потреби у внесенні змін до навчального процесу під час семестру.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

6

ТБЕК.188314.001 ПЗ

**1 ПОСТАНОВКА ЗАДАЧІ**

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

7

ТБЕК.188314.001 ПЗ

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 створення такої програмної системи, що має максимально автоматизувати процес створення розкладу занять для навчального закладу, володіти функціоналом 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 з них;
2. Викладачі, що мають володіти ім’ям, 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:

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

8

ТБЕК.188314.001 ПЗ

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**

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

9

ТБЕК.188314.001 ПЗ

В даному дипломному проекті 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 задачі. Актуальність вирішення задачі знаходилась і знаходиться на дуже високому рівні. Причиною внесення даної теми до списку тем дипломних проектів стала 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 як на світовому масштабі, так і на локального рівні, що особливо проявляється в навчальному закладі для якого розробляється даний дипломний проект. Зростання актуальності автоматизації відбувається з постійно зростаючим темпом, через розвиток навчального закладу, що полягає в збільшення кількості вступників кожного року, та відкритті нових напрямків навчання. Такий розвиток збільшує важкість роботи розробки розкладу занять в квадратичній пропорції.

Варто зазначити, що автоматизація стає популярнішою і по іншій причині. Такою причиною є зменшення собівартості, а в свою чергу і збільшення доступності, технологій для її реалізації. З кожним роком відбується зменшення цін на предмети обчислювальної техніки. Те, що було неможливим для реалізації декілька років тому, 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 немало 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, володіє 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, вона володіє високою актуальністю зараз і буде актуальною ще багато років.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

10

ТБЕК.188314.001 ПЗ

**3 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ**

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

11

ТБЕК.188314.001 ПЗ

**3.1 Огляд технічних проблем**

Тема дипломного проекту володіє рядом технічних проблем, які потребують вирішення для виконання програмної системи у повному обсязі. Проблеми відрізняються за складністю та обсягом, але всі вони важливі для повного виконання завдання.

Серед проблем можна відокремити наступні проблеми:

* Відсутність автоматизації 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 використовуватися при проектуванні програмної системи і могла б полегшити роботу розробки. Всі поточні наробки доступні лише при виконанні роботи ручним методом і не можуть бути використаними. З цієї причини з’являється потреба у створенні нових методів для автоматизації роботи, що вимагає вивчення всієї предметної області.

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

Через причину недосконалості у розумінні 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 вирішенні, які ж дані є правильними.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

12

ТБЕК.188314.001 ПЗ

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, які потребують 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.[1]

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 даній області. Визначення об’єкта тісно пов’язане з визначенням класу, що характеризується своїми властивостями та поведінкою. Об’єкт володіє набором атрибутів або полів даних, що визначають його поточний стан у множині станів. Поведінку об’єкта визначають 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.[2]

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

13

ТБЕК.188314.001 ПЗ

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.[3]

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 інкапсуляції дозволяє зменшити зв’язність класів.[4]

Успадкування – це механізм утворення нових класів на основі розроблених раніше. Цей принцип дозволяє використовувати властивості та функціональність батьківського класу. Успадкування вводить поняття породжених класів або підкласів, а також класів на яких базується породжений клас – базовий клас.[5]

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 використовуватися там, де використовується батьківський клас.[6]

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

Для створення програмної системи використовували принципи об’єктно-орієнтованого програмування розроблені Робертом Мартіном, що володіють абревіатурою SOLID[7]. Даний принцип володіє складовими: принцип єдиного обов’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 йому обов’язок.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

14

ТБЕК.188314.001 ПЗ

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.[8]

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 поділеним на спеціалізовані інтерфейси ще на стадії проектування, задля гнучкості програмних компонентів в подальшій роботі.

Принцип інверсії залежностей – залежності в середині системи, які мають будуватися 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 принципи володіють 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 шаблонів, 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:

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

15

ТБЕК.188314.001 ПЗ

* 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 якій клас володіє мінімальними знаннями про реалізацію, а потреба в зміні програми залишається великою.
* Декоратор – структурний шаблон проектування, призначений для динамічного підключення додаткової поведінки до об’єкта. Надає гнучку альтернативу для практичного створення підкласів з метою розширення функціональності. Для тієї ж мети може використовуватися поведінковий шаблон «Стратегія».
* Стратегія – поведінковий шаблон проектування, призначений 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 незалежно від об’єктів-клієнтів.
* Фасад – структурний шаблон проектування, який дозволяє приховати проблеми реалізації системи шляхом зведення всіх можливих зовнішніх викликів до одного об’єкта, якому делегуються 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.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

16

ТБЕК.188314.001 ПЗ

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.[8]

**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 використовувалася 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.[9]

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.[10]

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

17

ТБЕК.188314.001 ПЗ

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.[11]

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 є:

* 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.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

18

ТБЕК.188314.001 ПЗ

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 драйвера, що були створені 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 завантажуватися по мірі необхідності у її використанні.

**3.4 Організація програмної системи**

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

19

ТБЕК.188314.001 ПЗ

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 володіє 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 прізвище, 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.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

20

ТБЕК.188314.001 ПЗ

«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 представленням 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 з набору цих елементів. Дані елементи володіють 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 тижня визначає роботу, що буде проводитися у цей час, до якого він буде 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 користувачем для заповнення програмної системи інформаціє про групи, викладачів, предмети, аудиторії, вихідні дні та неробочі 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 в інші доступні формати.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

21

ТБЕК.188314.001 ПЗ

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.

**3.5 Огляд аналогів**

При аналізі предметної області теми дипломного проекту важливо звернути 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 називатися лише найбільш відомі або найбільш успішні проекти, 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, графіки розділення часу по тижнях 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 від одного режиму до іншого. При створенні розкладу програми, аналіз всіх вимог видає інформацію про найкраще створення розкладу занять. При створені розкладу занять програмна система переглядає вимоги викладачів і на їх основі створює зауваження та підказки.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

22

ТБЕК.188314.001 ПЗ

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-Коледж володіє наступним інтерфейсом поданим на рисунку 1.

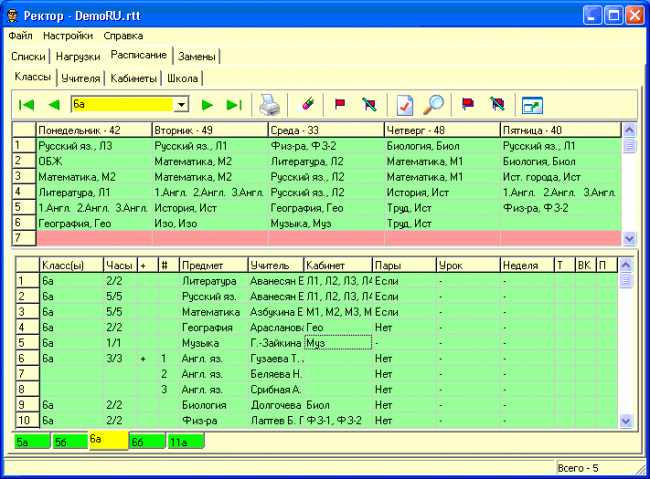


Рисунок 1 – Інтерфейс програми Ректор-Коледж

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 немало важливо, що 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, яка 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 розширений рівень підтримки надає можливість використовувати на тій операційній системі, яка вже наявна. Варто зауважити, що операційні системи сімейства Linux є повністю безкоштовними.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

23

ТБЕК.188314.001 ПЗ

Навчальна екосистема MySchool.ua, що додатково володіє набором функцій для автоматизації створення розкладу занять школи. В програмну систему потрібно занести дані про предмети, викладачів, аудиторії, класи, заповнити вимоги по 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, та підтримку тижневого навантаження.

Система володіє інтерфейсом, що поданий на рисунку 2.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

24

ТБЕК.188314.001 ПЗ

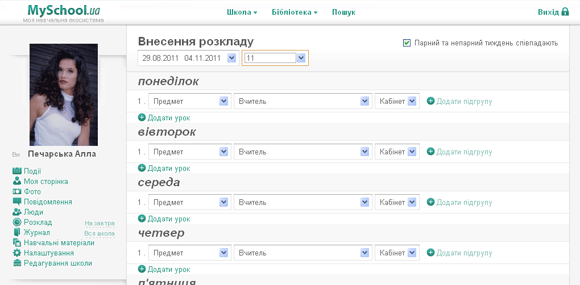


Рисунок 2 – Інтерфейс системи MySchool.ua

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 відповідають 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 – програма 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.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

25

ТБЕК.188314.001 ПЗ

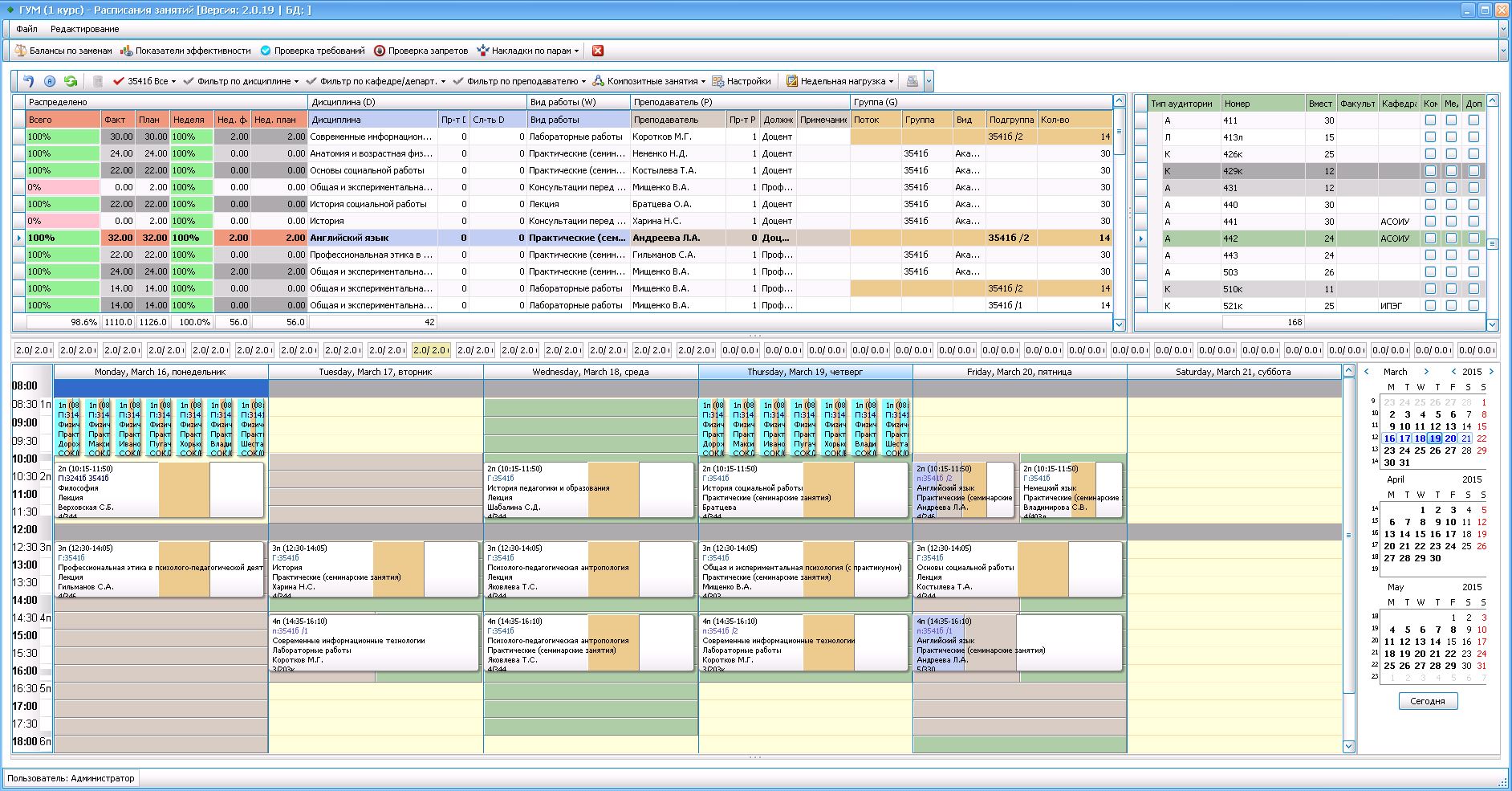


Рисунок 3 – Інтерфейс системи "Галактика Розклад занять"

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

**4 ОПИС СИСТЕМИ**

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

26

ТБЕК.188314.001 ПЗ

**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 4.

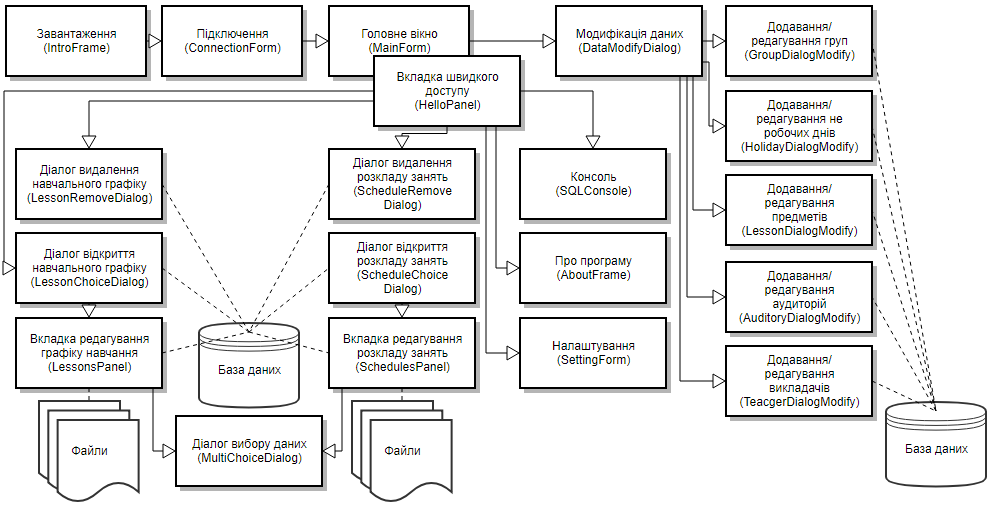
****

Рисунок 4 – 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 недаремно. Під час розробки були спроби використання інших шаблонів роботи системи, але використання такого шаблону дозволяє максимально продуктивно її використовувати. Більше того, використання саме цієї схеми дозволяє використовувати програму для одночасного виконання декількох задач.

Програмна система побудована на спеціально розробленій об’єктно-орієнтованій структурі. Загальна структура класів для даних подана на UML-діаграмі і має вигляд поданий на рисунку 5.

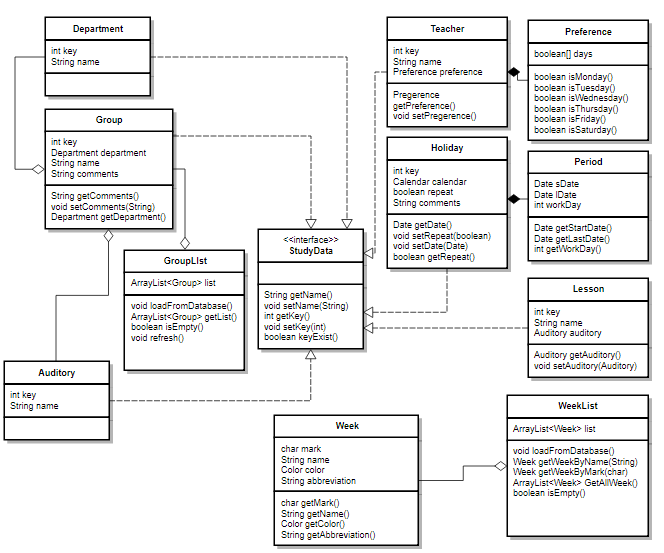


Рисунок 5 – Діаграма класів для роботи з даними

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

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

27

ТБЕК.188314.001 ПЗ

«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. Даний інтерфейс у спеціальні формі реалізації використовують класи Auditory, Teacher, Group та Lesson.

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

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

28

ТБЕК.188314.001 ПЗ

«Group» – клас, що являється абстрактною сутністю для реальної існуючої групи. Володіє інформацією про групу, яку представляє 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 самим 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 використовуватись 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 методами 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.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

29

ТБЕК.188314.001 ПЗ

«StudyData» – інтерфейс, що забезпечує існування у всіх класів нащадків методів для роботи з ключами та назвами. Використовується для забезпечення поліморфізму у роботі з даними програмами. 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 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 загальних класів, які зазвичай лише зберігають інформацію в програній системі, використовується класи, що мають складну структуру даних і володіють 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 6.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

30

ТБЕК.188314.001 ПЗ

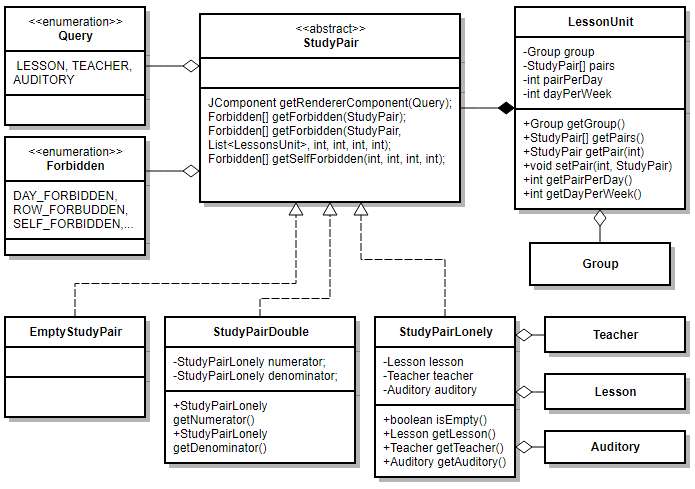
****

Рисунок 6 – 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 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, який 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 значеннями для позначення правил роботи програми під час редагування розкладу занять. Володіють в основному правилами заборони на використання в певний час або в певному місці. Перечислення володіють багатьма значеннями для прикладу приводиться лише декілька з них: заборона по дню використання, заборона використання в один час з поточним елементом, заборона використання в поточний тиждень та інші.

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

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

31

ТБЕК.188314.001 ПЗ

«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. Інформація зберігається як посилання на об’єкти Lesson, Teacher, Auditory. Володіє набором методів для обробки інформації зв’язаною з екземплярами цього класу. Являється реалізацією абстрактного класу StudyPair.

«StudyPairDouble» – клас, що використовує 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 представленням 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.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

32

ТБЕК.188314.001 ПЗ

Для роботи процесу створення та використання графіку навчального процесу використовується окремо створений набір класів. Структура та робота класів подана на UML-діаграмі на рисунку 7.

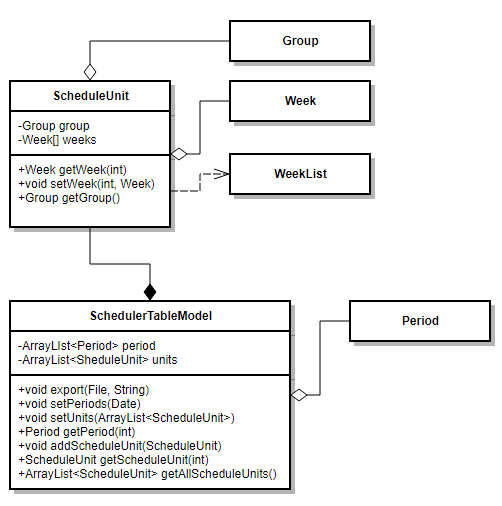
****

Рисунок 7 – 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.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

33

ТБЕК.188314.001 ПЗ

«SchedulerTableModel» – клас, що є представленням 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 класи, які не можна визначати до жодної з описаних вище категорій. Дані класи подані у вигляді UML-діаграми, що подана на рисунку 8.

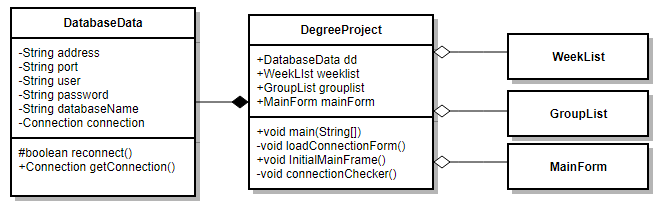
****

Рисунок 8 – Діаграма додаткових класів

«DatabaseData» – клас-служба, використовується 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 використовують 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 в випадку порушення буде намагатися здійснити відновлення або максимально м’яке відключення системи з можливістю подальшого відновлення.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

34

ТБЕК.188314.001 ПЗ

**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 генерується 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 в більшості інших, використовується GridLayoutManager.
* ImagePanel – компонент, що був створений для роботи саме у формі завантаження. Наслідує JPanel. Використовується для відображення графіку і стрічки прогресу завантаження. Володіє набором методів для роботи зі стрічкою завантаження.

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

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

35

ТБЕК.188314.001 ПЗ

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

BufferedImage – одна із реалізацій абстрактного класу Image. Дані реалізації використовуються через швидкість роботи і простоту у використанні. Інші реалізації володіють більш складною реалізацією 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 відбувається кожної ітерації 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.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

36

ТБЕК.188314.001 ПЗ

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 неможливо виконати під’єднання то відбується відображення форми з’єднання. Форма з’єднання в режимі конструювання володіє наступним виглядом поданим на рисунку 9.

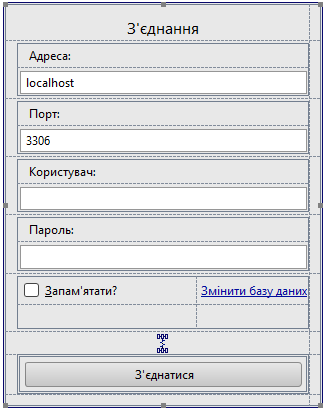


Рисунок 9 – Форма з’єднання з базою даних в режимі конструктора

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

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

37

ТБЕК.188314.001 ПЗ

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 значення в програмі, що було введене, передається 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. Використовуються 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:

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

38

ТБЕК.188314.001 ПЗ

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.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

39

ТБЕК.188314.001 ПЗ

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, 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 10.

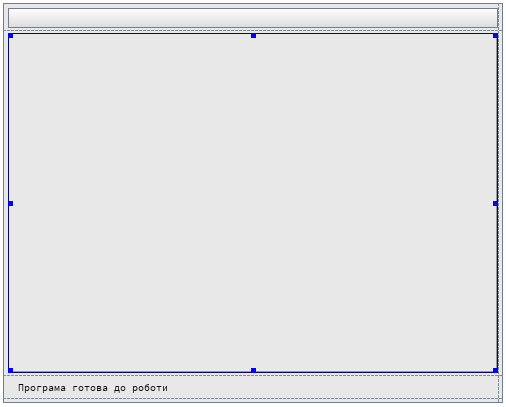


Рисунок 10 – Головна форма програми

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 використовується як заповнювач вільного простору. Він використовується для того, щоб всі інші елементи займали мінімальний простір в панелі інструментів.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

40

ТБЕК.188314.001 ПЗ

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 відбувається 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 11.



Рисунок 11 – Вітальна панель в режимі конструктора

Дана панель складається з компонентів JPanel, JButton, JLabel, які описувалися вище.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

41

ТБЕК.188314.001 ПЗ

Далі по порядку зустрічі користувача з формами ідуть форми заповнення даними програмної 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 12.

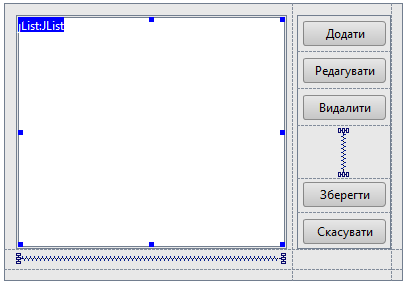


Рисунок 12 – Форма модифікації даних

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

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

42

ТБЕК.188314.001 ПЗ

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 виділеній 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 використовують спеціальні форми обробки даних.

Форма додавання та редагування даних «Аудиторій» володіє наступних виглядом у режимі конструктора поданого на рисунку 13.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

43

ТБЕК.188314.001 ПЗ

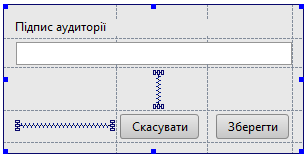


Рисунок 13 – Діалог модифікації даних аудиторій

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 14.

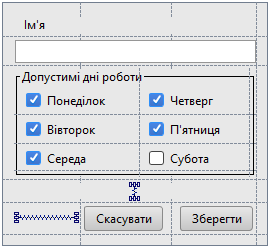


Рисунок 14 – Діалог модифікації даних викладачів

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 15.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

44

ТБЕК.188314.001 ПЗ

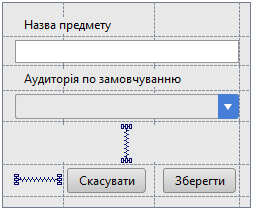


Рисунок 15 – Діалог роботи з «Предметами»

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 від вбудованого класу JDialog, що використовується для надання всіх 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 16.

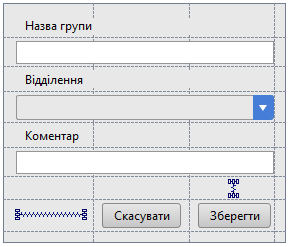


Рисунок 16 – Діалог модифікування груп

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 потрібною 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.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

45

ТБЕК.188314.001 ПЗ

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 17.

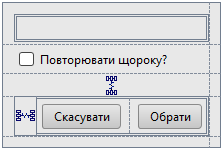


Рисунок 17 – Форма редагування даних про не робочі дні

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.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

46

ТБЕК.188314.001 ПЗ

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. Компоненти генеруються програмним кодом, тому в режимі конструктора він не відобрається.

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

Після заповнення бази даних необхідними даними 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 18.

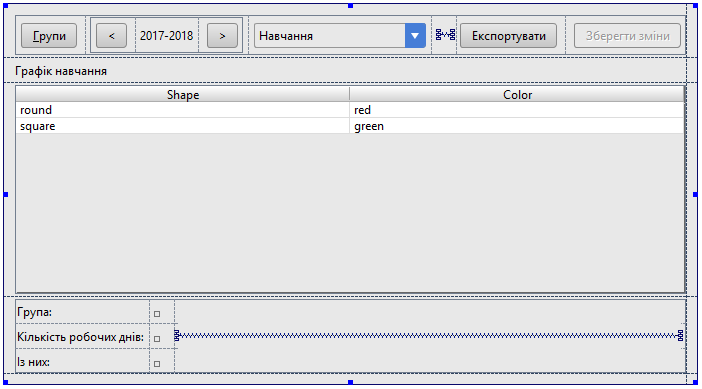


Рисунок 18 – Панель створення та редагування графіку навчального процесу

Дана панель володіє функціоналом, 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.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

47

ТБЕК.188314.001 ПЗ

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 який максимально 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 використовуються. 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 19.

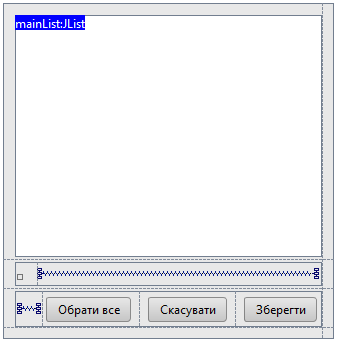


Рисунок 19 –Діалог роботи з групами

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.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

48

ТБЕК.188314.001 ПЗ

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 про 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 20.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

49

ТБЕК.188314.001 ПЗ

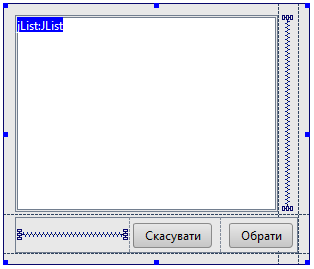


Рисунок 20 – Діалог відкриття графіку навчання

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, яка буде потім додана 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 21.

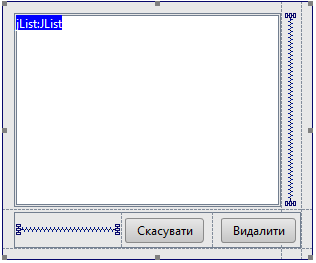


Рисунок 21 – Діалог видалення графіків навчання

Для створення та редагування розкладу занять використовується спеціально розроблена панель LessonsPanel. В режимі конструювання можна переглянути на рисунку 22.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

50

ТБЕК.188314.001 ПЗ

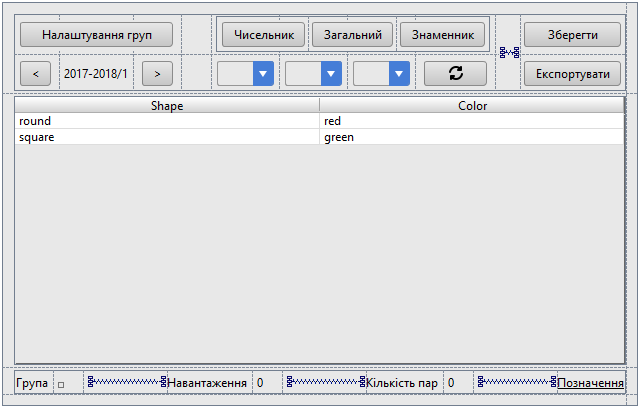


Рисунок 22 – Панель роботи з розкладом занять

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 більш 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.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

51

ТБЕК.188314.001 ПЗ

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.

**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 23.

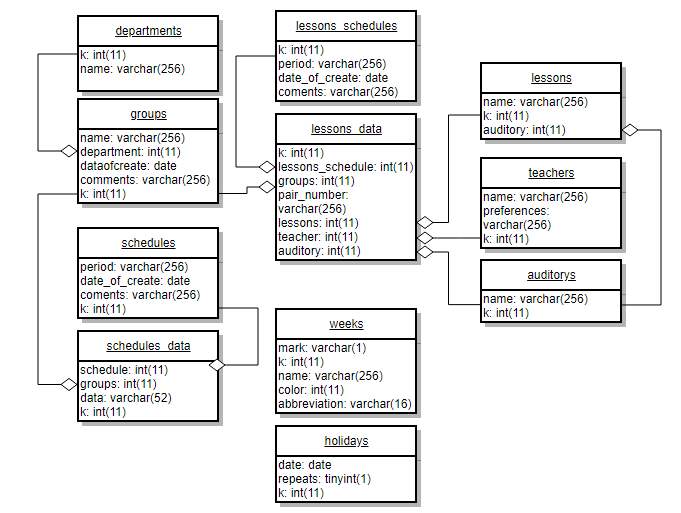
****

Рисунок 23 – Структура бази даних

Опис структури бази даних потрібно почати з таблиць, що не мають зв’язків з іншими таблицями бази даних. Призначення даних таблиць – бути довідниками, програмна система лише звертається до них за даними, які вони вміщують. Такими таблицями є weeks та holidays.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

52

ТБЕК.188314.001 ПЗ

«weeks» – використовується для зберігання можливих варіантів тижнів в графіку навчального процесу. Коли потрібно отримати набір даних, програмна система звертається до цієї таблиці. Збереження інформації для подальшої роботи виконується в іншому форматі.

«holidays» – таблиця-довідник, що використовується для збереження даних про неробочі дні. Дана таблиця може змінюватися користувачем, але програмна система отримує лише довідку від неї.

Дані в базі даних можна розділи на дві групи, одна, що використовується для збереження даних графіку навчального процесу, а інша для збереження даних про розклад занять. Головними в цих групах виступають дві таблиці «schedules\_data» та «lessons\_data». Дані таблиці використовують усі інші таблиці для правильного функціонування системи і забезпечення функціоналу для роботи.

«schedules\_data» – таблиця, що використовується для збереження всіх графіків навчального процесу. Містить в собі здебільшого кодовану і форматовану інформацію. Дані розкладені на багато записів і тому для роботи використовується зв’язана таблиця «schedules», що використовується як довідник по головній таблиці. Містить в собі набір даних для роботи з «schedules\_data», хоча робота можлива і без неї, але використання такої таблиці прискорює роботу системи.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

53

ТБЕК.188314.001 ПЗ

«lessons\_data» – таблиця, що використовується для збереження всіх даних про розклади занять створені у програмній системі. Як і в графіках навчального процесу дані в таблиці зберігаються у розкладеному вигляді в таблиці «lessons\_schedules» і для роботи використовується таблиця, що містить ключову інформацію про дані.

Для роботи також використовують таблиці з інформацією потрібної для роботи програмної системи:

«groups» – таблиця, що містить дані про всі групи наявні в програмній системі. Користувач може вносити зміни до цієї таблиці і вони будуть відображатися на всій програмній системі. Для роботи використовується довідникова таблиці «departments».

«departments» – таблиця, що вміщує в собі дані про наявні відділення наявні в навчально закладі. Використовується як частина таблиці «groups».

«teachers» – таблиця, що використовується для збереження даних про викладачів навчального закладу. Містить всю потрібну інформацію для функціонування викладача як частини програмної системи.

«lessons» – таблиця, що використовується для збереження даних про наявні предмети в навчальному закладі. Кожен запис містить лише ключову інформацію для представлення абстрактної версії об’єкту в програмній системі.

«auditorys» – таблиця, що використовується для збереження даних про аудиторії, які використовують для формування розкладу занять. Інформація використовується в розкладі занять під час роботи.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

54

ТБЕК.188314.001 ПЗ

**4.4 Робота системи**

Робота програмної системи розпочинається із завантаження системи в пам’ять комп’ютера. По причині, що програмна система володіє широким функціоналом і може виконуватися на комп’ютерах минулих поколінь, то було вирішено створити вікно завантаження системи. Відкриття цього вікна відбувається відразу після запуску програмної системи і відображає процес завантаження до його завершення. Вікно завантаження володіє наступним виглядом поданим на рисунку 24.

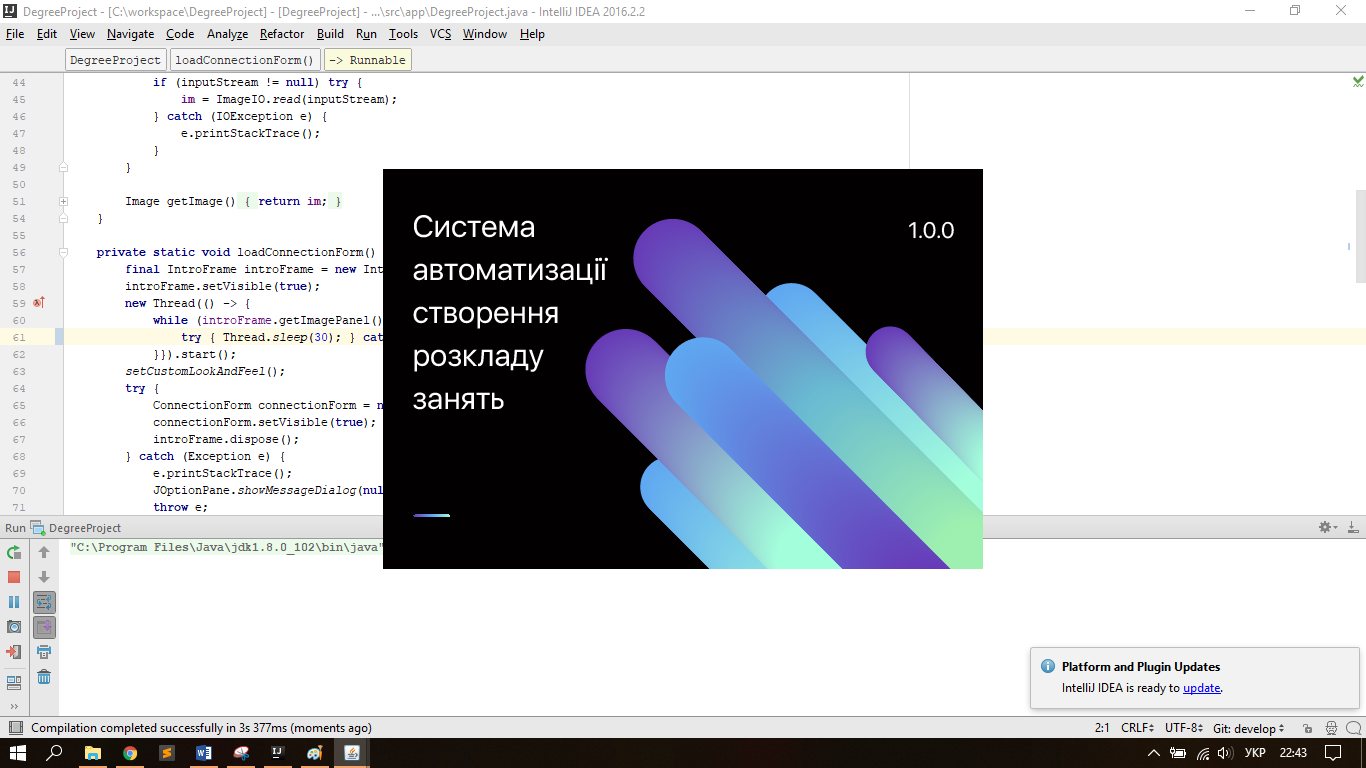


Рисунок 24 – Форма завантаження

Коли програмна система повністю завантажена до пам’яті то відбувається закриття форми завантаження і відкриття форми приєднання до бази даних, яка потрібна для роботи системи. У випадку, якщо система уже завантажувалася і виконувала приєднання до бази даних, а користувач перемкнув прапорець на варіант збереження даних для входу, то відбудеться повторна спроба приєднання. Коли з’єднання буде встановлено, то вікно з’єднання не буде відображеним, а відразу відбудеться перехід до головного вікна системи. Якщо ж цих даних не буде, або з’єднання неможливо встановити, то буде відображено вікно приєднання подане на рисунку 25.

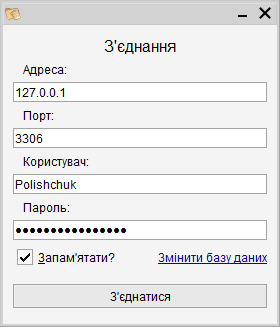


Рисунок 25 – Форма приєднання до сервера

Після вказання даних і натиснення на кнопку «З’єднатися» відбувається спроба встановлення з’єднання. У випадку, якщо процес з’єднання буде успішним то відбувається завантаження даних з бази даних і відображення головної форми програми поданої на рисунку 26.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

55

ТБЕК.188314.001 ПЗ

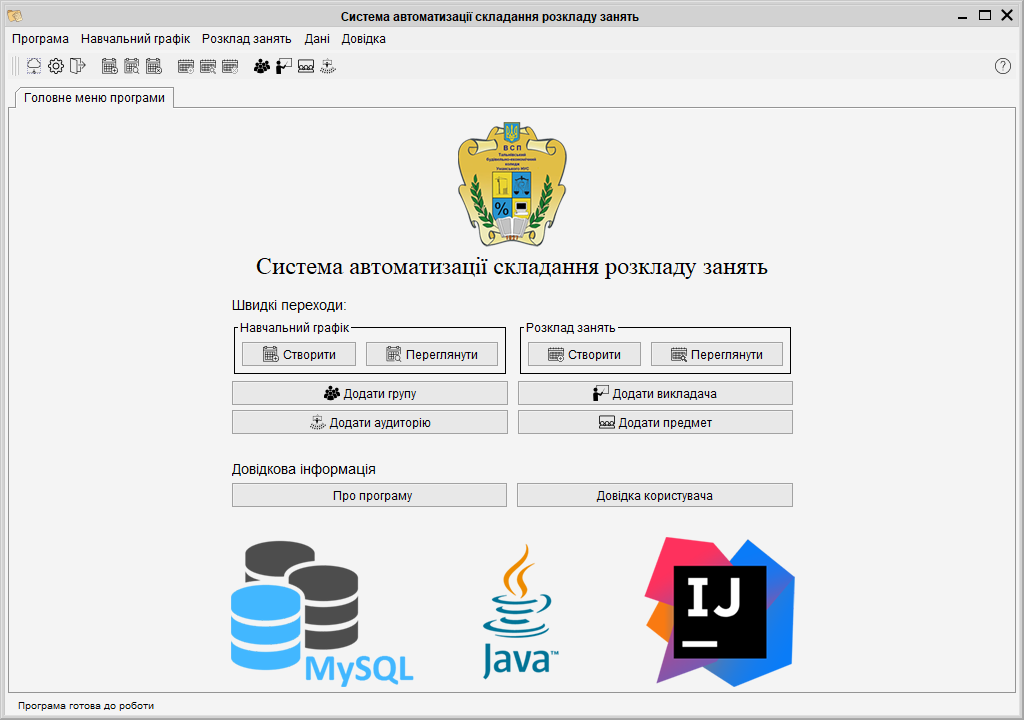


Рисунок 26 – Головна форма програми

Коли головна форма доступна, користувач може виконувати всі потрібні дії. При першому запуску потрібно заповнити базу даних для можливості подальшого її використання. Розглянемо вигляд додавання на прикладі викладачів. Виконуємо перехід до «Дані», а потім «Викладачі». Після натиснення на цей пункт меню відбудеться відкриття форми редагування, що подана на рисунку 27.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

56

ТБЕК.188314.001 ПЗ

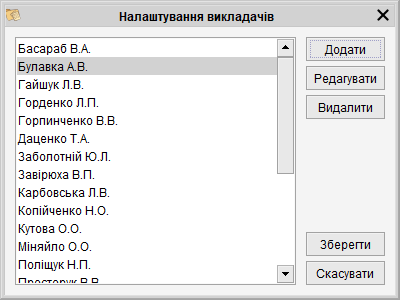


Рисунок 27 – Форма налаштування даних про викладачів

Для виконання дії потрібно натиснути на відповідну до дії кнопку з правої сторони. Після заповнення всіх даних про викладачів та всіх інших даних потрібних для роботи, можна переходити до створення графіку навчального процесу. Для створення графіку навчального процесу потрібно перейти до відповідного пункту меню. Після цього до головної форми буде додана відповідна закладка з інструментом для створення графіку навчального процесу. У випадку, якщо потрібно редагувати уже існуючий графік навчального процесу, то це можливо зробити у відповідному меню. Після натиснення буде відкрита спеціальна закладка, що відповідає існуючому графіку в базі даних. В обох випадках відбудеться відкриття закладки, що має на головній формі вигляд поданий на рисунку 28.

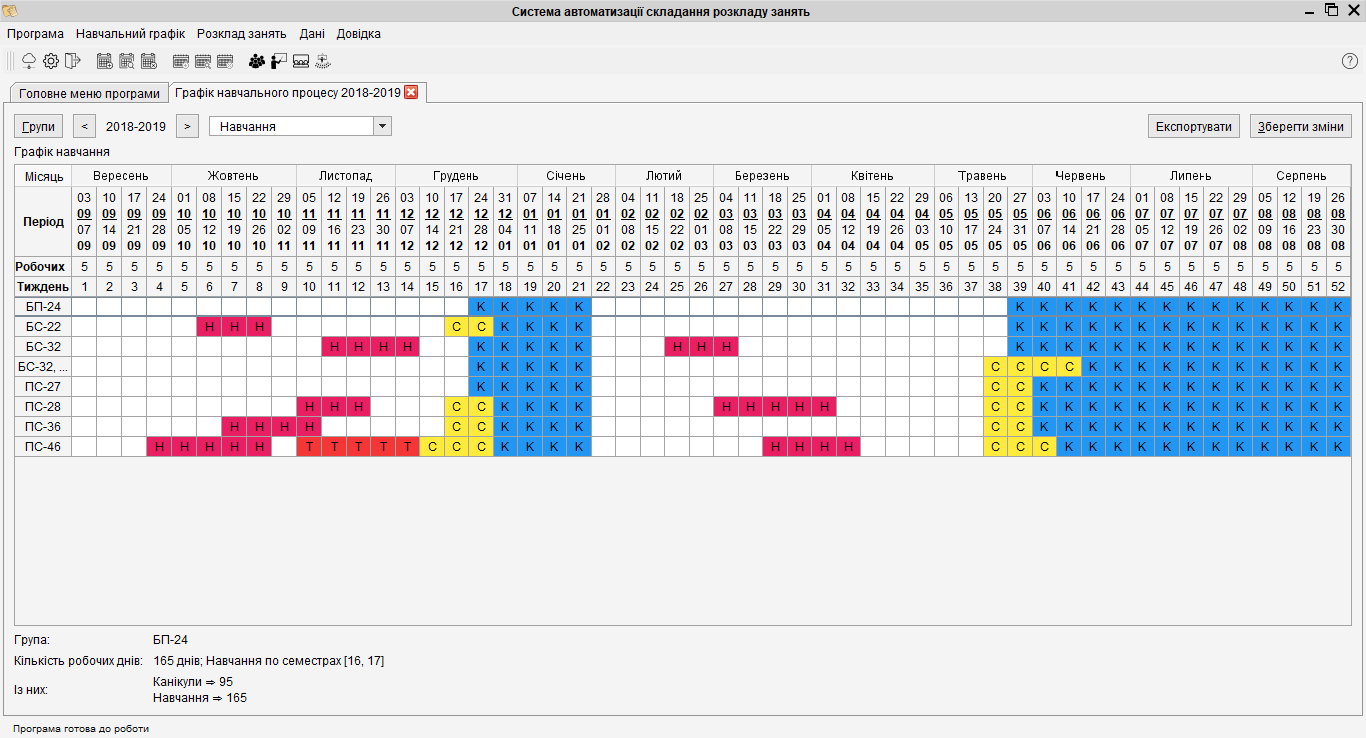


Рисунок 28 – Режим створення графіку навчального процесу

Коли графік навчального процесу створено, можна перейти до створення розкладу занять. Для створення розкладу занять потрібно перейти до відповідного пункту меню програми. Після натиснення на пункт «Створення розкладу занять» відбудеться відкриття закладки для створення розкладу занять. Для прикладу заповнений розклад занять можна переглянути на рисунку 29.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

57

ТБЕК.188314.001 ПЗ

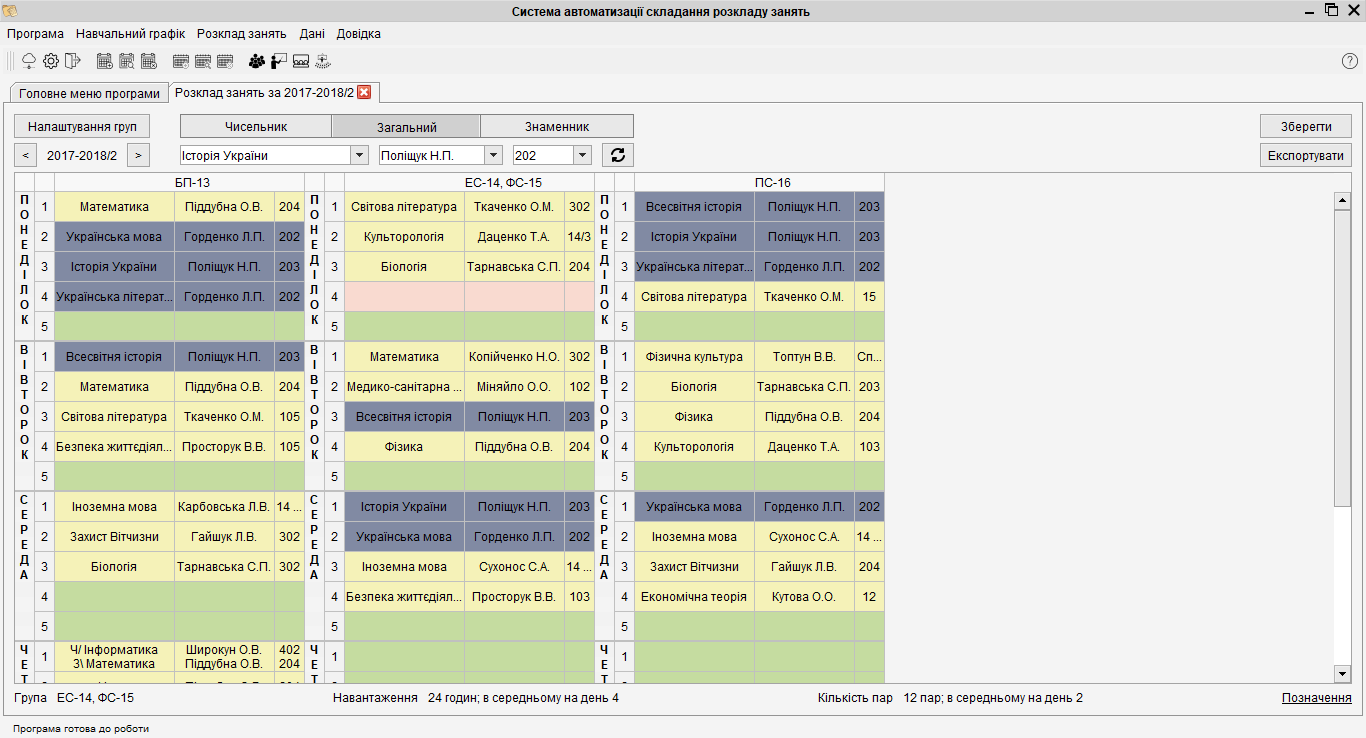


Рисунок 29 – Головна форма під час створення розкладу занять

**5 ВИБІР ЗАСОБІВ РЕАЛІЗАЦІЇ**

**5.1 Операційна система**

Для створення програмної системи завдання дипломного проекту, а також для виконання всіх додаткових задач дипломного проекту використовувалася операційна система Microsoft Windows 10 Pro, що встановлена на робочій машині. Виконання здійснювалося на версії 10.0.16299. Дипломний проект спроектовано і виконано таким чином, що його можливо виконувати і на інших сімействах операційних систем, наприклад Linux або macOS. Додатково всі програмні засоби є багатоплатформними, але так як навчання здійснювалося саме за операційними системи сімейства Windows, то вона була обрана основною.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

58

ТБЕК.188314.001 ПЗ

Термін «операційна система» використовується в значенні системне програмне забезпечення, яке управляє апаратними та програмними ресурсами і надає загальні послуги для комп’ютерних програм. Використовується як оболонка між апаратним забезпеченням та програмними системами, які на ній встановлені. Визначає правила виконання та багатопоточності, що надає можливість прискорення виконання систем.

Для роботи використовується Microsoft Windows 10 Pro тому, що це найновіша версія системи із сімейства Windows. Вона володіє усім найновішим функціоналом, що був розроблений в даний момент. Версія Pro використовується через те, що вона надає всі можливості розробки програмних систем, а не тільки для роботи. В цій версії весь функціонал є доступним для використання.

Програмна система дипломного проекту була створена за допомогою мови програмування Java і тому може бути виконана на будь-якій операційній системі, що володіє відповідною JRE (Java Runtime Environment) для виконання. Тестування роботи здійснювалося на операційних системах Windows XP, Windows 7, Linux Ubuntu 11.03, Linux Ubuntu 14.04 та macOS 10.13 High Sierra. Програмна система виконувалася і працювала на всіх вказаних системах і володіла всім затребуваним у завданні функціоналом. Використання, а також перенесення системи, можливе без будь-якого додаткового втручання зі сторони програміста або користувача.

Завданням дипломного проекту було розробка програмної системи для вищого навчального закладу ВСП Тальнівського будівельно-економічного коледжу. Вищий навчальний заклад володіє ліцензією на декілька систем із сімейства Windows, а саме операційні системи Windows XP та Windows 7, а тому розробка була націленою в першу чергу на ці системи. На них відбувалося додаткове тестування та перевірка функціоналу в робочому режимі.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

59

ТБЕК.188314.001 ПЗ

**5.2 Мова програмування і середовище розробки**

Для виконання завдання дипломного проекту по створенні програмної системи була обрано об’єктно-орієнтована мова програмування високого рівня Java версії 1.8.

Java – сильно типізована об’єктно-орієнтована мова програмування, що була розроблена компанією Sun Microsystems. Застосування написані на Java зазвичай транслюються в спеціальні байт-коди і тому можуть працювати на будь-якій архітектурі комп’ютерів з допомогою віртуальної Java-машини. Переваги мови програмування:

* Незалежність від операційної системи та обладнання за допомогою використання спеціальних байт-кодів.
* Розроблена програмна система є дуже гнучкою в плані безпеки, тому що виконання здійснюється в Java-машині.
* Можливість використання JIT-технології для транслювання часто використовуваних класів в машинний код для прискорення виконання.

XML – розширювана мова розмітки. Використовується для створення шаблонів вікон або компонентів програмної системи. Використовується спеціальна версія, яка підтримується мовою програмування Java.

Для створення програмної системи використовувалося інтегроване середовище розробки програм від компанії JetBrains під назвою IntelliJ IDEA версії 2016 конфігурації 2.2.

IntelliJ IDEA – це інтегроване середовище розробки програмного забезпечення для багатьох мова програмування, в особливості для Java, JavaScript, Python. Володіє широким набором інтегрованих інструментів для рефакторингу, що дозволяє набагато швидше створювати і розробляти програмні системі.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

60

ТБЕК.188314.001 ПЗ

Причиною використання саме цього середовища є присутність безкоштовної ліцензії для студентів, що надає можливість на час навчання отримати повноцінну версію програми без жодних обмежень. Також використання цього середовища надає набагато більший функціонал в порівнянні з всіма іншими доступними для використання IDE. Додатково дане середовище розробки є розширюваним за допомогою сторонніх програм, що надає можливість створити його таким, як потрібно.

Для швидкого редагування документів в моменти коли неможливо запустити IntelliJ IDEA використовувався текстовий редактор Sublime Text. Sublime Text – безкоштовний багатоплатформний текстовий редактор з підтримкою розширень та налаштувань візуальних тем з можливістю завантаження. Здійснюється підтримка основних мов програмування та їх ключових конструкцій для внесення швидких змін. Володіє підтримкою підключення компіляторів, такий як javac або OpenJDK, що надає можливості компіляції прямо під час редагування. Це зменшує необхідність на переключення до інших програмних системи і збільшує швидкість розробки.

Для переведення коду програми в байт-коди використовуються програми компілятори. Для створення програмної системи дипломного проекту використовувався офіційних компілятор створений компанією Oracle. Версія використовуваного компілятора javac 1.8.0\_102. Даний компілятор використовується для можливості виконання програмних систем в Java-машині версії 8 і вище. Версія 8 є найбільш популярною і тому була обрана для використання.

Для розширення можливостей мови програмування використовується набір безкоштовних бібліотек розроблених компанією Oracle під назвою JDK. Даний пакет надає шаблони і заготовки коду для створення власних програмних систем.

При розробці масштабної програмної системи потрібно слідкувати за проектом та змінами в ньому. Для цих цілей використовуються системи контролю версій. При роботі використовувався сервіс GitHub, один з найбільших веб-сервісів для хостингу проектів. Даний сервіс заснованих на технологій версій Git. Використання цієї технології забезпечило повну безпеку та захист проекту, надало можливості створення ліцензії та розміщення проекту в мережі Інтернет.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

61

ТБЕК.188314.001 ПЗ

Однією з вимог в завданні на дипломний проект було виведення результатів у форматі xls. При роботі з цими файлами використовувалася програма Microsoft Excel.

**5.3 Система керування базами даних**

В завданні дипломного проекту було поставлена вимога використання бази даних для збереження даних програмної системи. Для розробки і роботи була обрана база даних MySQL від компанії Oracle. Встановлення і робота бази даних відбувається в консольному режимі, тому для покращення сприйняття використовується MySQL Workbench.

MySQL Workbench – інструмент для візуального проектування бази даних, інтегрувального проектування, моделювання, створення та використання бази даних в єдиному безшовному оточенні для системи бази даних MySQL. Програма дозволяє наглядно представити модель бази даних в графічному вигляді, наглядно встановити зв’язки між таблицями включаючи зв’язок «багато до багатьох».

MySQL Workbench володіє широким функціоналом, але у випадку використання серверної MySQL бази даних, тоді доводиться використовувати попередньо встановлену систему керування базами даних. Зазвичай на серверах використовувалася phpMyAdmin.

phpMyAdmin – система керування базами даних з відкритим кодом, що була написана на мові програмування PHP. Надає інтерфейс для адміністрування базою даних MySQL. Дана система надає функціонал для роботи через браузер і не тільки працювати з базою даних, але і керувати її структурою і виконувати SQL-запити. Робота в цій системі виконувалася з серверами баз даних.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

62

ТБЕК.188314.001 ПЗ

Додатково для роботи з базою даних використовувалася оболонка бази даних MySQL, що представлена у вигляді терміналу. Дана система має назву MySQL Shell. Перевагами використання є мінімальні затрати ресурсів на використання та роботи системи. Її легко запустити, виконати перевірку чи тестування і так само швидко завершити роботу.

**5.4 Необхідні вимоги до конфігурування ЕОМ**

Для правильного та повноцінно виконання програмної системи даного дипломного проекту система виконання потребується конфігурування на достатньому рівні. Загальні вимоги конфігурування:

* Використання IBM сумісного комп’ютера;
* Використання 32-розрядної або 64-розрядної системи з підтримкою 32-розрядної системи;
* Використання кольорового дисплею з мінімальною частотою оновлення 30 Hz та підтримка DirectX 9;
* Використання клавіатури;
* Використання маніпулятора миші для полегшеного доступу до інформації. Можлива заміна за допомогою клавіш швидкого доступу.

До програмної системи ставляться різні вимоги по апаратному забезпеченню, в залежності від можливостей які потрібно реалізовувати. Для мінімального запуску програмної системи вимагається:

* Архітектура – 32-біта;
* Процесор – 233 МГц;
* Оперативна пам’ять – 64 МБ;
* Відеоадаптер – Super VGA (800x600);
* Вільне місце дискового простору – 1,5 ГБ;
* Інші пристрої – звукова карта, колонки, навушники.

Для виконання програмної системи у звичайному режимі радиться використовувати апаратну системи з такими вимогами:

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

63

ТБЕК.188314.001 ПЗ

* Архітектура – 32-біта;
* Процесор – 1 ГГц;
* Оперативна пам’ять – 1 Гб;
* Відеоадаптер – підтримка DirectX 9 і WDDM;
* Вільне місце дискового простору – 16 Гб вільного місця;
* Інші пристрої – звукова карта, колонки, навушники.

Для виконання програмної системи у максимальному режимі радиться використовувати апаратну систему, яка відповідає таким вимогам:

* Архітектура – 64-біти;
* Процесор – 1 ГГц;
* Оперативна пам’ять – 4 Гб;
* Відеоадаптер – підтримка DirectX 9 і WDDM 1.0;
* Вільне місце дискового простору – 20 Гб;
* Пристрій введення – мультисенсорний дисплей.

Програмна система була розроблена на операційній системі Windows 10 і розроблялася з підтримкою всіх операційних систем для яких існує робоча Java-машина версії 8 і вище. Програмна система не потребує попередньої підготовки перед перенесенням на іншу операційну систему і може використовуватися відразу після встановлення.

**6 ОХОРОНА ПРАЦІ**

При роботі з програмною системою і персональним комп’ютером користувач має дотримуватися вимог правил охорони праці викладених в законі України про охорону праці, а також положень про охорону праці при роботі з персональними комп’ютерами та програмними системами, що володіють великим обсягом технічної інформації. Для роботи з програмною системою дипломного проекту було розроблено спеціальні положення з охорони праці, що будуть викладені надалі.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

64

ТБЕК.188314.001 ПЗ

Під час роботи з програмною системою користувач перебуває під впливом наступних фізичних та психофізіологічних чинників:

* підвищений рівень статичної напруги, а також ураження електричним струмом;
* підвищений рівень електромагнітного випромінювання;
* підвищений рівень запиленості повітря робочої зони;
* понижена або підвищена вологість і рухливість повітря робочої зони;
* підвищений або понижений рівень освітленості;
* підвищений рівень прямої і відбитої блискоті, засліпленості.
* напруженість зору та уваги;
* інтелектуальні, емоційні та тривалі статичні навантаження;
* монотонність праці;
* великий обсяг інформації, що обробляється за одиницю часу;
* нераціональна організація робочого місця.

Для зменшення шкідливого впливу на здоров’я користувача персонального комп’ютера необхідно рівномірно розподіляти і чергувати характер робіт відповідно до їх складності, а також застосовувати перерви в роботі, під час яких рекомендується виконувати комплекс вправ виробничої гімнастики:

* для творчої роботи в режимі діалогу з ПК − 15 хв. відпочинку після кожної години роботи;
* для введення інформації − 10 хв. відпочинку після кожної години роботи;
* для читання інформації з попереднім запитом (діалоговий режим роботи) − 15 хв. відпочинку через кожні 2 години роботи.

Тривалість безперервної роботи за персональним комп’ютером без перерви не повинен перевищувати 2-х годин. Тривалість обідньої перерви визначається чинним законодавством про працю та правилами внутрішнього розпорядку навчального закладу.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

65

ТБЕК.188314.001 ПЗ

Щоденно перед початком роботи потрібно проводити вологе прибирання та очищення екрану від пилу та інших забруднень.

В приміщенні, де виконуються роботи на ПК, для нейтралізації зарядів статичного поля, рекомендується збільшувати вологість повітря за допомогою кімнатних зволожувачів. Повинна бути ефективна вентиляція і підтримуватися відносна вологість повітря на рівні 40-60% (можна розмістити поблизу ПК квіти або акваріум).

Вікна в приміщеннях повинні бути обладнані регульованими пристроями типу жалюзі, завіс, зовнішніх козирків та ін. Основний потік природного світла повинен бути зліва, сонячні промені і відблиски не повинні потрапляти в поле зору користувача ПК і на екрани ПК. Користувачі ПК не повинні сидіти обличчям до вікон. При штучному освітленні повинні застосовуватися переважно люмінесцентні лампи типу люмінесцентних. Допускається установка світильників місцевого освітлення для підсвічування документів.

Площа на одне робоче місце користувача ПК повинна бути неменше 4,5 кв. м.

Робочі місця з ПК повинні розміщуватися таким чином, щоб відстань від екрану одного ПК до тилу іншого була не менше 2,0 м, а відстань між бічними поверхнями ПК – не менше 1,2 м.

Екран ПК має бути віддалений від очей користувача на 500-700 мм в залежності від розміру екрана і мати противідблискове покриття. Клавіатуру необхідно розташовувати на робочому столі, не допускаючи її хитання, або на окремому столі на відстані 100-300 мм від ближнього до робочого краю. Положення клавіатури та кут її нахилу (в межах 5°..15°) повинен відповідати вимогам користувача ПК.

Для зменшення несприятливого впливу під час використання пристрою типу «миша» слід забезпечити вільну велику поверхню столу для переміщення «миші» і зручного упору ліктьового суглоба.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

66

ТБЕК.188314.001 ПЗ

Принтер та подібну оргтехніку необхідно розташовувати так, щоб доступ до них користувача ПК і його колег був зручним: максимальна відстань до клавіш управління не повинна перевищувати довжину витягнутої руки (900-1300 мм по висоті, 400-500 мм по глибині).

Не можна ставити ПК впритул до стіни або загороджувати задню стінку системного блоку: це спричиняє порушення його охолодження і перегрів. Необхідно тримати відкритими усі вентиляційні отвори пристроїв.

Кабелі ПК повинні розташовуватися так, щоб їх не можна було пошкодити необережним рухом.

Робоче місце має бути обладнане так, щоб виключити незручні пози і тривалі статичні напруги тіла. Раціональна поза користувача ПК – розміщення тіла, при якому ступні користувача розташовані на рівні підлоги або на підставці для ніг, стегна зорієнтовані у горизонтальній площині, верхні частини рук вертикальні, кут ліктьового суглоба коливається в межах 70°..90°, зап'ястя зігнуті під кутом не більше 20°, нахил голови у межах 15°..20° і виключені часті її повороти. Сидіти потрібно прямо, не напружуючись.

Засоби праці, з якими користувач має тривалий або найбільш частий зоровий контакт, необхідно розташовувати в центрі зони зорового спостереження та моторного поля ближче 500 мм для забезпечення оптимального робочого положення користувача ПК.

**ВИСНОВКИ**

Отже, в результаті виконання дипломного проекту за темою «Створення програмної системи для автоматизації створення розкладу занять ВСП Тальнівський будівельно-економічний коледж УНУС» було розроблено програмну систему, що в повній мірі виконує затребувані вимоги. Програмна система реалізує автоматизацію створення розкладу занять для навчального закладу в повній мірі. Програмна система також реалізує автоматизацію створення графіку навчального процесу. За завданням дипломного проекту програмна система володіє функціоналом для перенесення інформації до інших форматів збереження даних.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

67

ТБЕК.188314.001 ПЗ

В результаті роботи над дипломним проектом була виконана робота з базами даних, що закріпила знання з взаємної роботи бази даних і програмної системи. Підключення до бази даних здійснюється через спеціальні драйвера, а всі дані отримуються і модифікуються за допомогою мови SQL. Вивчення цієї мови є частиною навчання в навчальному закладі, а тому використання її для виконання дипломного проекту закріпило і розширило знання.

Програмна система дипломного проекту була розроблена за всіма вимогами до використання інтерфейсів. Володіє врівноваженим і простим для сприйняття інтерфейсом. Навчання роботі в програмній системі відбувається дуже швидко, що є одною із вимог до створення дипломного проекту. Робота над створенням інтерфейсу програмної системи включена в курс навчання і всі знання були закріпленні і розширенні під час виконання завдання.

Під час виконання дипломного проекту додатково було виконано ознайомлення із предметною областю в особливості з областю створення розкладу занять в навчальному закладі. Завдання створення розкладу включає в себе широку галузь знань, що включають управління і планування. Всі ці знання допоможуть керувати власною організацію і планувати її роботу.

При роботі над дипломний проектом використовувалася об’єктно-орієнтована мова програмування високого рівня Java. Дана мова є одною із найпопулярніших і найбільш затребуваних мов програмування. Виконання завдань дипломного проекту значно розширило знання з мови програмування. Набуті знання є дуже цінними і обов’язково будуть використані в майбутньому. Додатково було виконано проектування програмної системи за допомогою UML, що дуже важливо для роботи над великими проектами. Створення програмної системи такого масштабу надає навиків роботи над проектом, поділення його на частини та використання можливостей об’єктно-орієнтованого програмування.

Для роботи в об’єктно-орієнтованому стилі програмування були вивчене та засвоєне використання шаблонів проектування. Використання шаблонів проектування за GOF є світовою практикою. Їх використання прискорює розробку програмної системи, а також надає їй можливості виконувати розширення та масштабування в майбутньому для створення модифікацій іншими програмістами. Програмна система володіє повністю відкритим кодом і створена таким чином, щоб будь-хто міг продовжити роботу на проектом і додавати нові особливості. Також при проектуванні застосовувалися принципи SOLID, що надають проекту більш розширеного поняття об’єктно-орієнтованого програмування. Використання правил SOLID надає стабільність програмній системі. Ці правила надають простоту і розширенні та створені модифікації.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

68

ТБЕК.188314.001 ПЗ

Для виконання завдання за темою дипломного проекту використовувалася IDE. Довгий час роботи дозволив розширити знання з використання і прискорив швидкість розробки. Всі майбутні проекти, що будуть створюватися на такій же технології будуть створюватися з більшою швидкістю і ефективністю. Знання з ефективного використання IDE цінуються в програмістів на високому рівні.

При тестуванні програмного продукту, що є багатоплатформним, було потрібно використовувати різні операційні системи. Навики роботи з операційними система Windows, Linux, masOS були засвоєні. Інтерфейс роботи з операційними системи володіє деякими особливостями і використання їх для тестування програмного продукту призвело до засвоєння певного досвіду, який буде потрібним в майбутньому. Кожен роботодавець володіє різними системами, які використовує для виробництва і досвід роботи у всіх основних системах прискорить швидкість пристосування до нових умов розробки.

Виконання дипломного проекту вважається успішним. Всі поставлені завдання були виконанні в повній мірі. Деякі завдання були виконані з творчим підходом, що додає їм особливості і ефективності у використанні. Всі програмні розробки володіють відкритим кодом і доступні кожному для використання.

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

69

ТБЕК.188314.001 ПЗ

**ВИКОРИСТАНА ЛІТЕРАТУРА**

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

70

ТБЕК.188314.001 ПЗ

1. Ерих Гама, Ричард Хелм, Ральф Джонсон, Джон Влисдес. Прийоми об’єктно-орієнтованого програмування. Патерни проектування — 1994 — 395 с. : ил.
2. Бертран Мейер. Объектно-ориентированное конструирование программных систем — 2004 — 1200 c
3. Kathy Sierra and Bert Bates. Head First Java, 2nd Edition — 2005 — 688 c.
4. Herbert Schildt. Java, A Beginner's Guide, 5th Edition — 2011 — 640 c.
5. Kent Beck. Test Driven Development: By Example — 2002 — 240 c
6. Joshua Bloch. Effective Java (2nd Edition) — 2008 — 346 c
7. Martin Fowler, Kent Beck, John Brant, William Opdyke, Don Roberts, Erich Gamma. Refactoring: Improving the Design of Existing Code — 1999 — 464 с
8. Robert C. Martin. Clean Code: A Handbook of Agile Software Craftsmanship — 2008 — 464 с
9. Алан Бьюли. Изучаем SQL. — 2016 — 312 c
10. Томас Коннолли, Каролин Бегг. Базы данных. Проектирование, реализация и сопровождение. Теория и практика — 2017 — 1440 c
11. Мартин Грабер. SQL для простых смертных. — 2014 — 378 c
12. Джон Уокенбах. Excel 2016. Библия пользователя. — 2017 — 1040
13. Ильдар Мухутдинов, Самиздат. Революционная десятка. Все секреты и тайны операционной системы Windows 10 — 2016 — 460 с
14. Соловьев Алексей. Разработка Linux — 2012 — 85 c
15. Apple Inc. A start guide on macOS 10.13. — 2016 — 43 c

**СПИСОК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ**

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

71

ТБЕК.188314.001 ПЗ

|  |  |  |
| --- | --- | --- |
| Git-technology | – | Розподілена система керування версіями файлів |
| GOF | – | Gang of Four – команда авторів книг |
| HTML | – | Hypertext Markup Language – мова розмітки. |
| IDE | – | Integrated Development Environment |
| JDBC | – | Java Database Connectivity |
| JDK | – | Java Development Kit |
| JIT-compilation | – | Just-in-time compilation |
| JRE | – | Java Runtime Environment |
| SOLID | – | Single responsibility; Open/close; Liskov substitution; Interface segregation; Dependency inversion – принцип для дизайну та розробки програмних систем. |
| SQL | – | Structured query language – декларативна мова програмування для взаємодії з базою даних. |
| UML | – | Unified Modeling Language – уніфікована мова моделювання, що використовується в парадигмі ООП. |
| XML | – | Extensible Markup Language |
| БД | – | База даних |
| ЕОМ | – | Електронна обчислювана машина |
| ООП | – | Об’єктно-орієнтоване програмування |
| ПК | – | Персональний комп’ютер |
| СКБД | – | Система керування базами даних |
|  |  |  |

**Додаток А**

ЗАТВЕРДЖУЮ

Голова циклової комісії

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2018 р

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

Специфікація

482.ТБЕК.88314-01

Листів 2

Розробник: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Поліщук В.П.

Керівник: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Харченко О.О.

Тальне 2018

|  |  |  |
| --- | --- | --- |
| **Позначення** | **Найменування** | **Примітка** |
| 482.ТБЕК.88314-01 1201 | Додаток Б Текст програми |  |
| 482.ТБЕК.88314-01 3401 | Додаток В Інструкція користувачеві |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Додаток Б**

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

Текст програми

482.ТБЕК.88314-01 1201

Листів 16

Розробник: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Поліщук В.П.

Вхідний метод програмної системи

public static void main(String[] args) throws IOException, SQLException {  
 *loadIcon*(); *// Завантаження ресурсів  
 loadConnectionForm*(); *// Завантаження форми*}

Метод завантаження програмної системи

private static void loadConnectionForm() {  
 final IntroFrame introFrame = new IntroFrame(); // Створення IntroFrame  
 introFrame.setVisible(true); // Відображення на екран  
 new Thread(() -> { // Створення нового потоку задач  
 while (introFrame.getImagePanel().nextStep() < introFrame.getImagePanel().getMax()) { // Виконувати завантаження  
 try { Thread.*sleep*(100); } catch (InterruptedException e) {*/\*\*/*}  
 }}).start(); // Запуск нового процесу  
 *setCustomLookAndFeel*(); // Встановлення режиму відображення  
 try { // Відкриття конструкції для обробки виключень  
 ConnectionForm connectionForm = new ConnectionForm(true);

// Створення екземпляру класу ConnectionForm і передача параметра true, що // означає дозвіл на автоматичне тихе підключення  
 connectionForm.setVisible(true); // Відкриття форми з’єднання  
 introFrame.dispose(); // Закриття форми завантаження  
 } catch (Exception e) { // Обробка всіх виключень, що можуть трапитися  
 e.printStackTrace(); // Виведення повідомлення в консоль  
 JOptionPane.*showMessageDialog*(null, e.getMessage());   
// Відображення повідомлення користувачеві  
 throw e;   
// Прокидання помилки далі для подальшої обробки  
 }  
} // Завершення методу завантаження програмної системи

Метод обробки підключення до бази даних

private void connectionButtonClick(ActionEvent event) {  
 try { // Виконання спроби підключення  
 DegreeProject.*databaseData* = new DatabaseData(addressTextField.getText(),  
 portTextField.getText(), userTextField.getText(),  
 passwordField.getPassword(), DegreeProject.*defaultDB*);  
 ResultSet rs = DegreeProject.*databaseData*.getConnection().createStatement().executeQuery("SHOW TABLES"); // Отримання структури бази даних  
 DatabaseStructureChecker(rs); // Перевірка структури  
 } catch (SQLException e1) {  
 SQLExceptionHandler(e1); // Обробка помилок підключення  
 }  
  
 if (rememberCheckBox.isSelected()) rememberMe(); // Виконання записування даних користувача  
 DegreeProject.*InitialMainFrame*(); // Запуск головної форми  
 dispose(); // Закриття вікна підключення  
}

Метод підключення до бази даних

private void createConnectionWithSchema() throws SQLException {  
// Створення адреси і виконання підключення до бази даних  
 connection = DriverManager.*getConnection*("jdbc:mysql://" + this.address + ":" + this.port + "/" + this.databaseName + "?useSSL=false&useUnicode=true &characterEncoding=utf-8",  
 this.user, this.password);  
}

Створення головного вікна програми

public static void InitialMainFrame() {  
// Створення нового потоку перевірки підключення до бази даних  
 new Thread(DegreeProject::*connectionChecker*).start();  
 *mainForm* = new MainForm(); // Створення головного вікна програми  
 *mainForm*.addWindowListener(new WindowAdapter() {  
 @Override  
// Обробка події закриття вікна  
 public void windowClosing(WindowEvent e) {  
 super.windowClosing(e);  
 try { // Закриття підключення до бази даних для звільнення ресурсів  
 DegreeProject.*databaseData*.getConnection().close();  
 } catch (SQLException e1) {  
// Виведення повідомлень про помилки користувачеві  
 JOptionPane.*showMessageDialog*(null, e1.getMessage());  
 e1.printStackTrace();  
 }  
 }  
 });  
 try { // Завантаження даних до WeekList та GroupList  
 *WEEKLIST* = new WeekList();  
 *GROUPLIST* = new GroupList();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
}

Обробка модифікації даних

public void MenuItemDataTeacher(ActionEvent event) {  
 try (Statement st = DegreeProject.*databaseData*.getConnection().createStatement();  
 ResultSet rs = st.executeQuery("SELECT *\** FROM teachers")) {  
// Отримання з бази даних інформації про викладачів  
 TreeSet<Teacher> teacherTreeSet = new TreeSet<>();  
 while (rs.next()) {  
// Створення дерева, що складається з викладачів  
 teacherTreeSet.add(new Teacher(rs.getInt("k"), rs.getString("name"),  
 Preference.*parsePreference*(rs.getString("preferences"))));}  
// Переведення дерева в масив  
 Teacher[] inputData = new Teacher[teacherTreeSet.size()];  
 int count = 0;  
 for (Teacher teacher : teacherTreeSet) inputData[count++] = teacher;  
// Відкриття нового діалогу модифікації даних  
 StudyData[] outputData = DataModifyDialog.*getInstance*(inputData, new DataModifyInterface() {  
 public StudyData add() {return TeacherDialogModify.*getModify*();}  
 public StudyData edit(StudyData t) {return TeacherDialogModify.*getModify*((Teacher) t);}  
 public boolean remove(StudyData t) {return true;}  
 public void exit(StudyData[] t) {}  
 }, "Налаштування викладачів");  
// Обробка інформації після модифікації  
 boolean b = true;  
 if (outputData.length == inputData.length)  
 for (int i = 0; i < outputData.length; i++) {  
 Teacher in = inputData[i];  
 Teacher out = (Teacher) outputData[i];  
 if (!in.equals(out)) {  
 b = false;  
 break;  
 }  
 }  
 else b = false;  
 if (b) return;  
 for (Teacher in : inputData) {  
 boolean bool = true;  
 for (StudyData out : outputData) {  
 if (in.equals(out)) {  
 bool = false;  
 break;}  
 *//Обробка випадку з Preference* if (out instanceof Teacher) {  
 boolean equals = ((Teacher) (out)).getPreference().equals(in.getPreference());  
 if (out.getKey() == in.getKey() && out.keyExist() && in.keyExist() && !equals) {  
 bool = false;  
 break;  
 }  
 }  
 }  
 if (bool) st.execute("DELETE FROM teachers WHERE k LIKE '" + in.getKey() + "'"); // Внесення змін до бази даних  
 }  
 for (StudyData item : outputData) {  
 if (item.keyExist())  
 st.execute("INSERT INTO teachers(name, preferences, k) VALUE ('" + item.getName() + "', '"  
 + ((Teacher) item).getPreference().getData() + "', " + item.getKey() +  
 ") ON DUPLICATE KEY UPDATE name = '" + item.getName() + "', preferences = '" + ((Teacher) item).getPreference().getData() + "';");  
 else  
 st.execute("INSERT INTO teachers(name, preferences) VALUE ('"  
 + item.getName() + "', '"  
 + ((Teacher) item).getPreference().getData() + "');");  
 }  
 DegreeProject.*mainForm*.setStatusBar("Дані успішно збережено до бази даних");  
 JOptionPane.*showMessageDialog*(null, "Дані успішно змінено");  
 } catch (SQLException e) {  
 JOptionPane.*showMessageDialog*(null, e.getMessage());  
 e.printStackTrace();  
 }  
}

Метод обробки візуального вигляду графіку навчання

public class TableCellDayNameRenderer extends DefaultTableCellRenderer {  
 private final int PAIR\_IN\_DAY;  
 public TableCellDayNameRenderer(int pair\_in\_day) {PAIR\_IN\_DAY = pair\_in\_day;}  
 private final String[] daysName = new String[]{  
 "ПОНЕДІЛОК", "ВІВТОРОК", "СЕРЕДА", "ЧЕТВЕРГ", "ПЯТНИЦЯ", "СУБОТА", "НЕДІЛЯ"  
 };  
 @Override  
 public Component getTableCellRendererComponent(JTable table, Object value, boolean isSelected, boolean hasFocus, int row, int column) {  
 JLabel labelTop = (JLabel) super.getTableCellRendererComponent(table, value, isSelected, hasFocus, row, column);  
 labelTop.setBackground(new Color(242, 242, 242));  
 labelTop.setFont(new Font(labelTop.getFont().getName(), Font.*BOLD*, labelTop.getFont().getSize()));  
 labelTop.setHorizontalAlignment(*CENTER*);  
 if ((row \* 2) % (PAIR\_IN\_DAY \* 2) < daysName[row / PAIR\_IN\_DAY].length()) {  
 labelTop.setText(String.*valueOf*(  
 daysName[row / PAIR\_IN\_DAY].charAt((row \* 2) % (PAIR\_IN\_DAY \* 2))  
 ));  
 }  
 JLabel labelBottom = new JLabel();  
 if (((row \* 2) + 1) % (PAIR\_IN\_DAY \* 2) < daysName[row / PAIR\_IN\_DAY].length()) {  
 labelBottom.setText(String.*valueOf*(  
 daysName[row / PAIR\_IN\_DAY].charAt(((row \* 2) + 1) % (PAIR\_IN\_DAY \* 2))));  
 }  
 labelBottom.setHorizontalAlignment(*CENTER*);  
 labelBottom.setFont(new Font(labelBottom.getFont().getName(), Font.*BOLD*, labelBottom.getFont().getSize()));  
  
 JPanel panel = new JPanel(new GridLayout(2, 1));  
 panel.add(labelTop);  
 panel.add(labelBottom);  
 panel.setBorder(row % PAIR\_IN\_DAY == PAIR\_IN\_DAY - 1 ?  
 BorderFactory.*createMatteBorder*(0, 0, 1, 1, Color.*LIGHT\_GRAY*)  
 :  
 BorderFactory.*createMatteBorder*(0, 0, 0, 1, Color.*LIGHT\_GRAY*)  
 );  
 if (row % PAIR\_IN\_DAY == PAIR\_IN\_DAY - 1)  
 panel.setBorder(BorderFactory.*createMatteBorder*(0, 0, 2, 1, Color.*LIGHT\_GRAY*));  
 return panel;  
 }  
}

Метод обробки візуального вигляду розкладу занять

public Component getTableCellRendererComponent(JTable table, Object value, boolean isSelected, boolean hasFocus, int row, int column) {  
 JLabel label = (JLabel) super.getTableCellRendererComponent(table, value, isSelected, hasFocus, row, column);  
 label.setHorizontalAlignment(*CENTER*);  
 label.setBorder(BorderFactory.*createMatteBorder*(0, 0, 1, 1, *borderColor*));  
 *// Обробка таблиці в місці даних* if (row >= *ROW\_DATA* && column >= *COLUMNS\_DATA*) {  
 label.setBackground(Color.*WHITE*);  
 if (value instanceof Week) {  
 Week w = (Week) value;  
 label.setBackground(w.getColor() == null ? Color.*BLACK* : w.getColor());  
 label.setToolTipText(w.getName());  
 }  
 }  
 *// Обробка таблиці в місці підписів* if ((row == *ROW\_PERIODS* || row == *ROW\_WORK\_DAYS* || row == *ROW\_WEEK\_COUNT*) && column == *COLUMN\_NAMES*) {  
 label.setBackground(UIManager.*getColor*("Panel.background"));  
 if (value instanceof String) {  
 label.setToolTipText((String)value);  
 }  
 }  
 *// Підписи груп* if (row >= *ROW\_DATA* && column == *COLUMN\_NAMES*) {  
 label.setBackground(UIManager.*getColor*("Panel.background"));  
 if (value instanceof Group) {  
 Group g = (Group) value;  
 label.setText(g.getName());  
 label.setToolTipText(g.getComments());  
 }  
 }  
 *// Обробка виділеної стрічки* if ((isSelected || hasFocus) && row >= 3) {  
 label.setBorder(BorderFactory.*createMatteBorder*(1, 0, 2, 1, new Color(122, 138, 153)));}  
 return label;  
}

Метод експортування розкладу занять

public void export(File file, String period) throws IOException {  
 final int trRow = 5;  
 final int trCell = 0;  
  
 HSSFWorkbook workbook = new HSSFWorkbook();  
 Font font = workbook.createFont();  
 font.setFontName("Calibri");  
 HSSFPalette customPalette = workbook.getCustomPalette();  
 for (int i = 0; i < DegreeProject.*WEEKLIST*.GetAllWeek().size(); i++) {  
 java.awt.Color color = DegreeProject.*WEEKLIST*.GetAllWeek().get(i).getColor();  
 customPalette.setColorAtIndex((short) (i + 55), (byte) color.getRed(), (byte) color.getGreen(), (byte) color.getBlue());  
 }  
 customPalette.setColorAtIndex(IndexedColors.*GREY\_25\_PERCENT*.index, (byte)238, (byte)238, (byte)238);  
 customPalette.setColorAtIndex(IndexedColors.*GREY\_50\_PERCENT*.index, (byte)164, (byte)164, (byte)164);  
 HashMap<Week, CellStyle> styleHashMap = new HashMap<>();  
 for (int i = 0; i < DegreeProject.*WEEKLIST*.GetAllWeek().size(); i++) {  
 CellStyle cellStyle = workbook.createCellStyle();  
 cellStyle.setBorderBottom(BorderStyle.*THIN*);  
 cellStyle.setBorderRight(BorderStyle.*THIN*);  
 cellStyle.setFillForegroundColor((short) (i + 55));  
 cellStyle.setFillPattern(FillPatternType.*SOLID\_FOREGROUND*);  
*// cellStyle.setFont(font);* cellStyle.setVerticalAlignment(VerticalAlignment.*CENTER*);  
 cellStyle.setBottomBorderColor(IndexedColors.*GREY\_50\_PERCENT*.getIndex());  
 cellStyle.setRightBorderColor(IndexedColors.*GREY\_50\_PERCENT*.getIndex());  
 cellStyle.setAlignment(HorizontalAlignment.*CENTER\_SELECTION*);  
 styleHashMap.put(DegreeProject.*WEEKLIST*.GetAllWeek().get(i), cellStyle);  
 }  
  
 Sheet sheet = workbook.createSheet("Графік навчання за " + period);  
  
*//Шапка* if (trRow > 3) {  
 CellStyle headerCellStyle = workbook.createCellStyle();  
 headerCellStyle.setVerticalAlignment(VerticalAlignment.*CENTER*);  
 headerCellStyle.setAlignment(HorizontalAlignment.*CENTER\_SELECTION*);  
 HSSFFont headerFont = workbook.createFont();  
 headerFont.setFontHeightInPoints((short)18);  
 headerCellStyle.setFont(headerFont);  
 Row headerRow = sheet.createRow(trRow - 3);  
 Cell headerCell = headerRow.createCell(trCell);  
 headerCell.setCellStyle(headerCellStyle);  
 headerCell.setCellValue("Зведений графік навчального процесу на " + period + " навчальний рік");  
 sheet.addMergedRegion(new CellRangeAddress(trRow - 3, trRow - 2, trCell, 52));  
 }  
*//Місяці* int first = 1 + trCell;  
 int last = 0;  
 int prevMonth;  
 c.setTime(periods.get(0).getStartDate());  
 prevMonth = c.get(Calendar.*MONTH*);  
 for (int i = 0; i < 52 - 1; i++) {  
 c.setTime(periods.get(i).getStartDate());  
 if (c.get(Calendar.*MONTH*) == prevMonth) {  
 last = i + 1;  
 } else {  
 sheet.addMergedRegion(new CellRangeAddress(trRow, trRow, first + trCell, last + trCell));  
 first = i + 1;  
 prevMonth = c.get(Calendar.*MONTH*);  
 }  
 }  
 sheet.addMergedRegion(new CellRangeAddress(trRow, trRow, first + trCell, last + trCell + 1));  
  
 CellStyle monthStyle = workbook.createCellStyle();  
 monthStyle.setFillPattern(FillPatternType.*SOLID\_FOREGROUND*);  
 monthStyle.setFillForegroundColor(IndexedColors.*GREY\_25\_PERCENT*.getIndex());  
 monthStyle.setBorderBottom(BorderStyle.*THIN*);  
 monthStyle.setBorderRight(BorderStyle.*THIN*);  
 monthStyle.setBorderTop(BorderStyle.*THIN*);  
 monthStyle.setFont(font);  
 monthStyle.setAlignment(HorizontalAlignment.*CENTER\_SELECTION*);  
 monthStyle.setVerticalAlignment(VerticalAlignment.*CENTER*);  
 monthStyle.setBottomBorderColor(IndexedColors.*GREY\_50\_PERCENT*.getIndex());  
 monthStyle.setRightBorderColor(IndexedColors.*GREY\_50\_PERCENT*.getIndex());  
 monthStyle.setTopBorderColor(IndexedColors.*GREY\_50\_PERCENT*.getIndex());  
 Row monthRow = sheet.createRow(trRow);  
 Cell nameMonthCell = monthRow.createCell(trCell);  
 nameMonthCell.setCellStyle(monthStyle);  
 nameMonthCell.setCellValue("Місяць");  
 for (int i = 1 + trCell; i < 52 + 1 + trCell; i++) {  
 Cell monthCell = monthRow.createCell(i);  
 monthCell.setCellStyle(monthStyle);  
 c.setTime(periods.get(i - 1).getStartDate());  
 monthCell.setCellValue(MONTHS[c.get(Calendar.*MONTH*)]);  
 }  
*//Дати* CellStyle dateStyle = workbook.createCellStyle();  
 dateStyle.setRotation((short)90);  
 dateStyle.setFillPattern(FillPatternType.*SOLID\_FOREGROUND*);  
 dateStyle.setFillForegroundColor(IndexedColors.*GREY\_25\_PERCENT*.getIndex());  
 dateStyle.setBorderBottom(BorderStyle.*THIN*);  
 dateStyle.setBorderRight(BorderStyle.*THIN*);  
 dateStyle.setBorderTop(BorderStyle.*THIN*);  
 dateStyle.setFont(font);  
 dateStyle.setBottomBorderColor(IndexedColors.*GREY\_50\_PERCENT*.getIndex());  
 dateStyle.setRightBorderColor(IndexedColors.*GREY\_50\_PERCENT*.getIndex());  
 dateStyle.setTopBorderColor(IndexedColors.*GREY\_50\_PERCENT*.getIndex());  
 Row dateRow = sheet.createRow(1 + trRow);  
 Cell nameDateCell = dateRow.createCell(trCell);  
 nameDateCell.setCellStyle(monthStyle);  
 nameDateCell.setCellValue("Період");  
 for (int i = 1 + trCell; i < 52 + 1 + trCell; i++) {  
 Cell dateCell = dateRow.createCell(i);  
 dateCell.setCellStyle(dateStyle);  
 String line = "";  
 c.setTime(periods.get(i - 1).getStartDate());  
 line += AddZeroBefore(c.get(Calendar.*DATE*)) + ".";  
 line += AddZeroBefore(c.get(Calendar.*MONTH*) + 1) + "-";  
 c.setTime(periods.get(i - 1).getLastDate());  
 line += AddZeroBefore(c.get(Calendar.*DATE*)) + ".";  
 line += AddZeroBefore(c.get(Calendar.*MONTH*) + 1);  
 dateCell.setCellValue(line);  
 }  
*//Кількість робочих днів* Row workDayRow = sheet.createRow(2 + trRow);  
 Cell nameWordDayCell = workDayRow.createCell(trCell);  
 nameWordDayCell.setCellValue("Робочих");  
 nameWordDayCell.setCellStyle(monthStyle);  
 for (int i = 1 + trCell; i < 52 + 1 + trCell; i++) {  
 Cell workDayCell = workDayRow.createCell(i);  
 workDayCell.setCellStyle(monthStyle);  
 workDayCell.setCellValue(periods.get(i - 1).getWorkDay());  
 }  
*//Номер тижня* Row weekRow = sheet.createRow(3 + trRow);  
 Cell nameWeekCell = weekRow.createCell(trCell);  
 nameWeekCell.setCellStyle(monthStyle);  
 nameWeekCell.setCellValue("Тиждень");  
 for (int i = 1 + trCell; i < 52 + 1 + trCell; i++) {  
 Cell weekCell = weekRow.createCell(i);  
 weekCell.setCellStyle(monthStyle);  
 weekCell.setCellValue(i - trCell);  
 }  
*//Дані* for (int i = 0; i < units.size(); i++) {  
 Group group = units.get(i).getGroup();  
 Row groupRow = sheet.createRow(4 + trRow + i);  
 Cell nameCell = groupRow.createCell(trCell);  
 nameCell.setCellStyle(monthStyle);  
 nameCell.setCellValue(group.getName());  
 for (int j = 0; j < 52; j++) {  
 Cell dataCell = groupRow.createCell(j + trCell + 1);  
 dataCell.setCellStyle(styleHashMap.getOrDefault(units.get(i).getWeek(j), workbook.createCellStyle()));  
 dataCell.setCellValue(units.get(i).getWeek(j).getAbbreviation());  
 }  
 }  
 for (int i = 0; i < DegreeProject.*WEEKLIST*.GetAllWeek().size(); i++) {  
 Row weekNameRow = sheet.createRow(i + 6 + units.size() + trRow);  
 Cell colorCell = weekNameRow.createCell(1 + trCell);  
 colorCell.setCellStyle(styleHashMap.getOrDefault(DegreeProject.*WEEKLIST*.GetAllWeek().get(i), workbook.createCellStyle()));  
 colorCell.setCellValue(DegreeProject.*WEEKLIST*.GetAllWeek().get(i).getAbbreviation());  
 sheet.addMergedRegion(new CellRangeAddress(i + 6 + units.size() + trRow, i + 6 + units.size() + trRow, 2 + trCell, 2 + 10 + trCell));  
 Cell nameCell = weekNameRow.createCell(2 + trCell);  
 nameCell.setCellStyle(monthStyle);  
 nameCell.setCellValue(DegreeProject.*WEEKLIST*.GetAllWeek().get(i).getName());  
 }  
 for (int i = trCell; i < 53 + trCell; i++) {  
 sheet.autoSizeColumn(i);  
 }  
 workbook.write(new FileOutputStream(file));  
 workbook.close();  
 }

**Додаток В**

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

Інструкція користувачеві

482.ТБЕК.88314-01 3401

Листів 14

Розробник: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Поліщук В.П.

**ЗМІСТ**

1. ВСТАНОВЛЕННЯ ПРОГРАМНОЇ СИСТЕМИ ТА ЇЇ КОМПОНЕНТІВ. . .3
2. РОБОТА З ПРОГРАМНОЮ СИСТЕМОЮ. . . . . . . . . . . . . . . . . . . . . . . . . . .5
   1. Завантаження програмної системи. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .5
   2. Додавання службової інформації. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .7
   3. Робота з графіком навчального процесу. . . . . . . . . . . . . . . . . . . . . . . . . 10
   4. Робота з розкладом занять. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .12

**1 ВСТАНОВЛЕННЯ ПРОГРАМНОЇ СИСТЕМИ ТА ЇЇ КОМПОНЕНТІВ**

У програмної системи існують деякі вимоги до системи у якій вона буде виконуватися. Для виконання програмної системи потрібно встановити потрібні для роботи компоненти. Java-машина є найголовнішим компонентом без якого не може виконуватися програмна система. Для встановлення Java-машини потрібно встановити відповідне програмне забезпечення. Отримати пакет встановлення можна на офіційному веб-сайті компанії Oracle. Після переходу на веб-сайт буде запропоновано завантажити пакет встановлення відповідно до операційної системи, яка встановлена на поточній машині. Завантаживши пакет встановлення, потрібно його запустити. Після запуску буде відображено вікно в якому повідомляється, що встановлення потребує прав адміністратора як це подано на рисунку В.1.

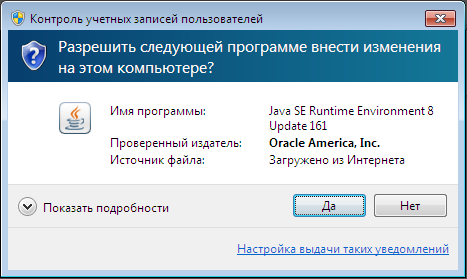


Рисунок B.1 – Вікно отримання прав адміністратора

Після натиснення на кнопку «Да» буде виконано автоматичне встановлення всіх компонентів для роботи Java-машини. Процес встановлення буде відображати спеціальне вікно, як показано на рисунку В.2.

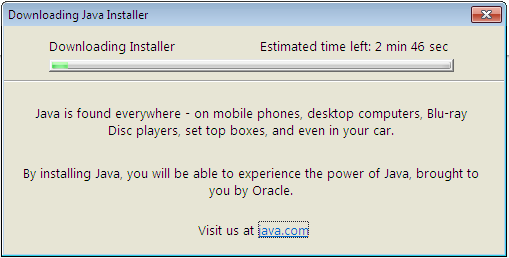


Рисунок В.2 – Вікно відображення процесу встановлення

Після встановлення Java-машини потрібно виконати перезавантаження системи. Коли перезавантаження буде виконано можна перевірити результати виконаної роботи виконавши команду в терміналі «java -version». Якщо результатом виконання буде відображення поточної версії Java встановленої в системі, отже можна переходити до встановлення самої програмної системи.

Встановлення програмної системи спрощене до мінімуму і тому для того, щоб встановити потрібно лише отримати виконуваний файл програми. Виконуваний файл програми зазвичай має назву «DegreeProject\_vX.X.jar» де X.X позначає поточну версії програмної системи. Виконуваний файл програми може мати іншу назву, але обов’язково закінчується розширенням файлу виконуваних програм Java «.jar». Даний файл також можна отримати на офіційній репозиторії проекту на сайт GitHub.com.

Коли виконуваний файл буде отримано, його потрібно розташувати у будь-якому зручному для використання місці. Надалі запуск буде відбуватися саме з цього файлу.

**2 РОБОТА З ПРОГРАМНОЮ СИСТЕМОЮ**

**2.1 Завантаження програмної системи**

Для завантаження програмної системи потрібно знайти виконуваний файл програми і запустити його за допомогою програми Java SE Platform Binary. Запуск здійснюється за допомогою натиснення ПКМ по виконуваному файлу, потім перехід до пункту «Відкрити з допомогою» і серед всіх перерахованих пунктів потрібно обрати «Java SE Platform Binary». Якщо такий пункт відсутній, отже потрібно пройти пункт «Встановлення програмної системи та її компонентів» знову.

Після відкриття виконуваного файлу на екрані буде відображено вікно завантаження, що має наступний вигляд поданий на рисунку В.3.

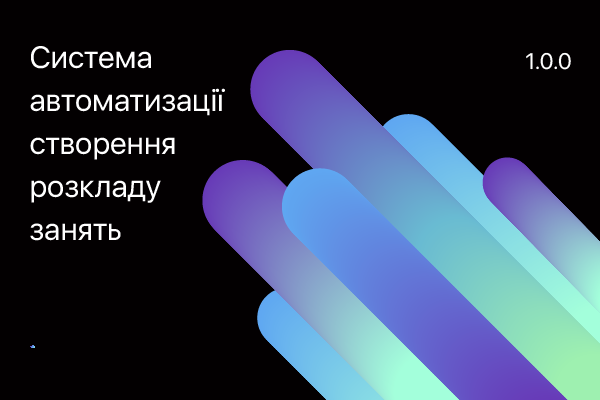


Рисунок В.3 – Вікно завантаження системи

Потрібно дочекатися кінця процесу завантаження. Процес завантаження відображається полосою завантаження в лівому нижньому куті. Коли завантаження буде завершено, відбудеться закриття вікна завантаження і відкриття вікна з’єднання з сервером бази даних. Вікно з’єднання з базою даних можна побачити на рисунку В.4.

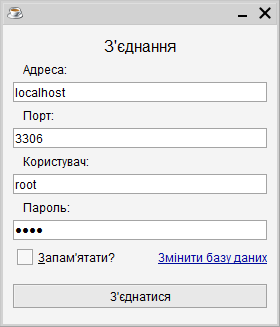


Рисунок В.4 – Вікно з’єднання з сервером бази даних

Для з’єднання потрібно вказати адресу сервера на йому розташовується база даних, потім вказати порт з’єднання з базою даних. Порт видається хостингом, але зазвичай порт відповідає 3306. Далі потрібно вказати ім’я користувача з доступом до адміністрування бази даних і пароль від цього користувача. Для того щоб в подальшому не виконувати процес введення даних, можна перемкнути прапорець «Запам’ятати?». Програмна система автоматично буде встановлювати з’єднання з базою даних «asfsc». Якщо така база даних буде відсутня, то вона буде автоматично створена. Якщо на сервері уже існує база даних з таким ім’ям, то можна встановити інше ім’ям в пункті «Змінити базу даних». Після введення всіх даних потрібно натиснути на кнопку «З’єднатися». Після натиснення буде виконано спробу встановити з’єднання. Якщо на сервері не буде існувати база даних, то вона автоматично створиться про що буде виведено сповіщення у виглядів полоси завантаження. Коли буде встановлено з’єднання буде виконано завантаження службової інформації із сервера та відкриття головної форми програми. Головна форма програми буде мати вигляд як подано на рисунку В.5.

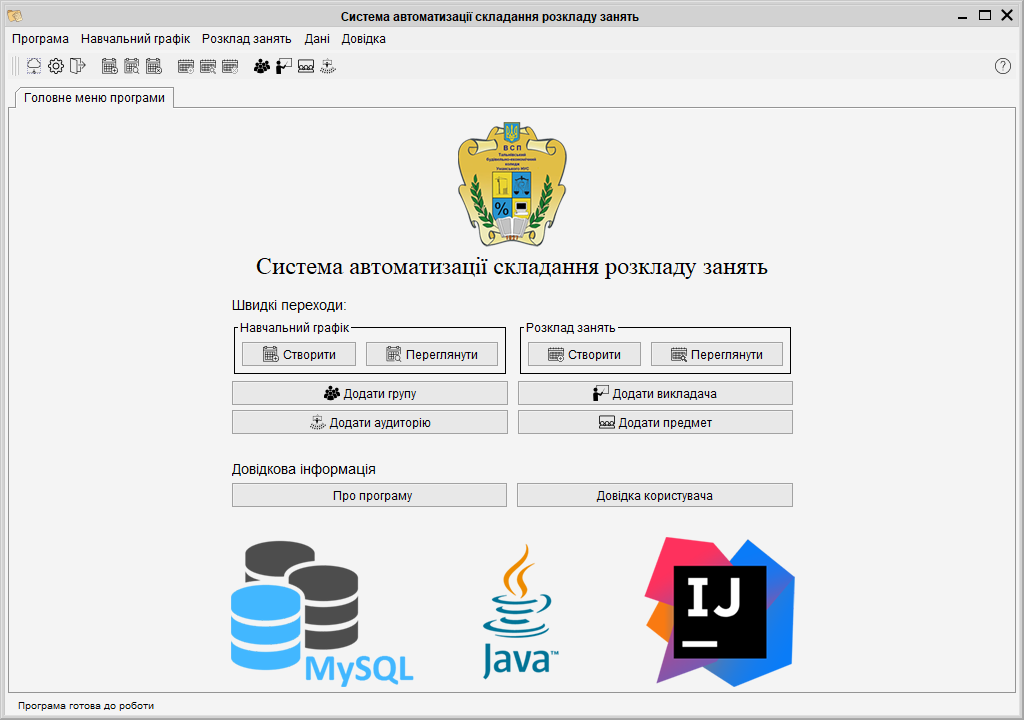


Рисунок В.5 – Головна форма програмної системи

**2.2 Додавання службової інформації**

Перед виконанням будь-яких дій зі створення графіку навчального процесу або створення розкладу занять потрібно додати всю інформацію про навчальний заклад, що складається з викладачів, предметів, аудиторій, груп та днів, які не є робочими. Спочатку розглянемо процес додавання аудиторій до програмної системи. Перейдемо до пункту меню «Дані», потім «Аудиторії» як показано на рисунку В.6.

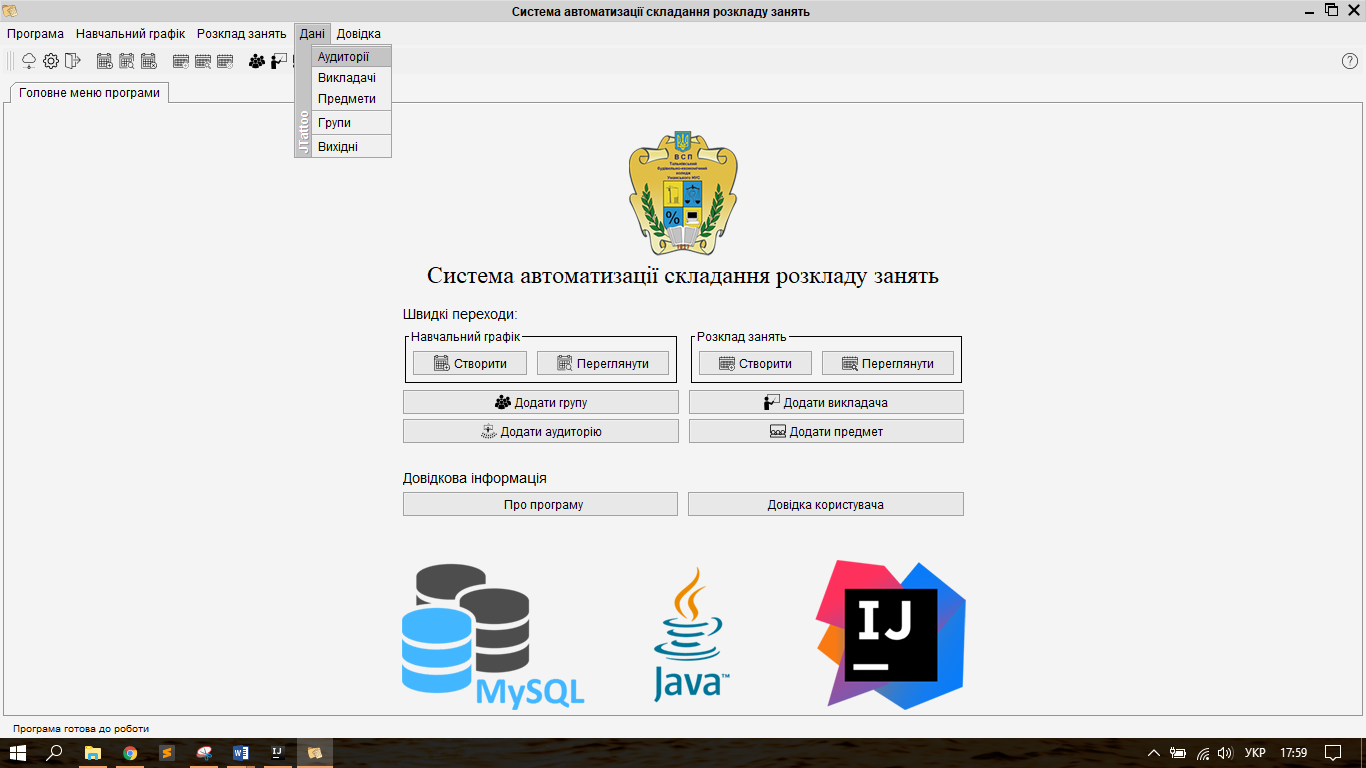


Рисунок В.6 – Вибір пункту в головному меню програми

Після натиснення даного пункту меню буде виконано відкриття діалогу модифікації даних про аудиторії в програмній системі. В діалог будуть додані всі існуючі в базі даних аудиторії. Можна виконувати додавання, редагування чи видалення аудиторій. Діалог модифікації даних відображено на рисунку В.7.

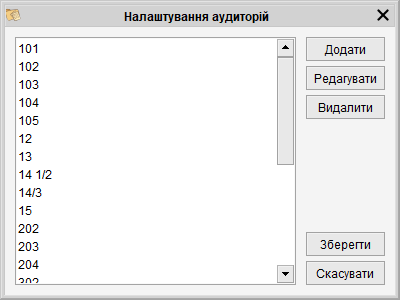


Рисунок В.7 – Діалог модифікації даних

При натисненні на кнопку «Додати» буде виконано відкриття діалогу введення потрібних даних для створення нового об’єкту. Діалог додавання відображається на рисунку В.8.

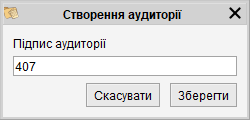


Рисунок В.8 – Діалог додавання аудиторій

В цьому діалозі підпис аудиторії відповідає назві її в дійсності. Далі розглянемо діалоги додавання інших даних. Діалог додавання викладачів відображено на рисунку В.9.

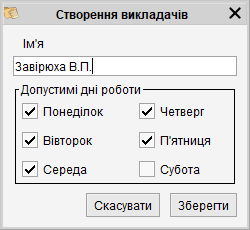


Рисунок В.9 – Діалог додавання викладачів

В діалозі створення викладача, ім’я викладача відповідає тому, як він буде підписуватися надалі у програмній системі, а прапорці на допустимих днях роботи відповідають дням, при яких викладач може працювати. Діалог додавання предметів відображений на рисунку В.10.

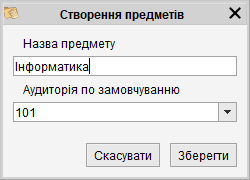


Рисунок В.10 – Діалог додавання предметів

В діалозі створення предметів назва предмету відповідає підпису предмета в програмній системі, а аудиторія по замовчуванню – це та аудиторія, що буде пропонуватися при створенні розкладу занять. Діалог додавання групи до програмної системи відображено на рисунку В.11.

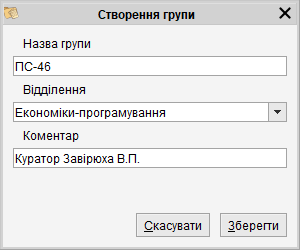


Рисунок В.11 – Діалог додання групи

В діалозі додавання групи назва групи відповідає підпису у програмній системі, відділення обирається із двох запропонованих можливих варіантів. Коментарі вказуються для уточнення використання групи. Коментарі будуть відображатися при розширеному перегляді групи. Діалог додавання неробочих днів виконується в діалозі поданому на рисунку В.12.

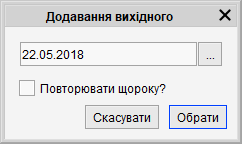


Рисунок В.12 – Діалог додавання вихідних

В даному діалозі відбувається вибір дати, яка буде вихідною. Якщо перемкнути прапорець «Повторювати щороку?», то дата буде використовуватися на всі наступні роки.

**2.3 Робота з графіком навчального процесу**

Коли всі дані заповнено, можна переходити до процесу роботи над графіком навчального процесу. Для створення нового графіку навчального процесу потрібно перейти до головного меню програму, обрати пункт «Навчальний графік», а потім «Створити навчальний графік». Після натиснення на кнопку, до закладок буде додана нова для створення навчального графіку. Головна форма в режимі створення графіка навчального процесу володіє наступним виглядом поданим на рисунку В.13.

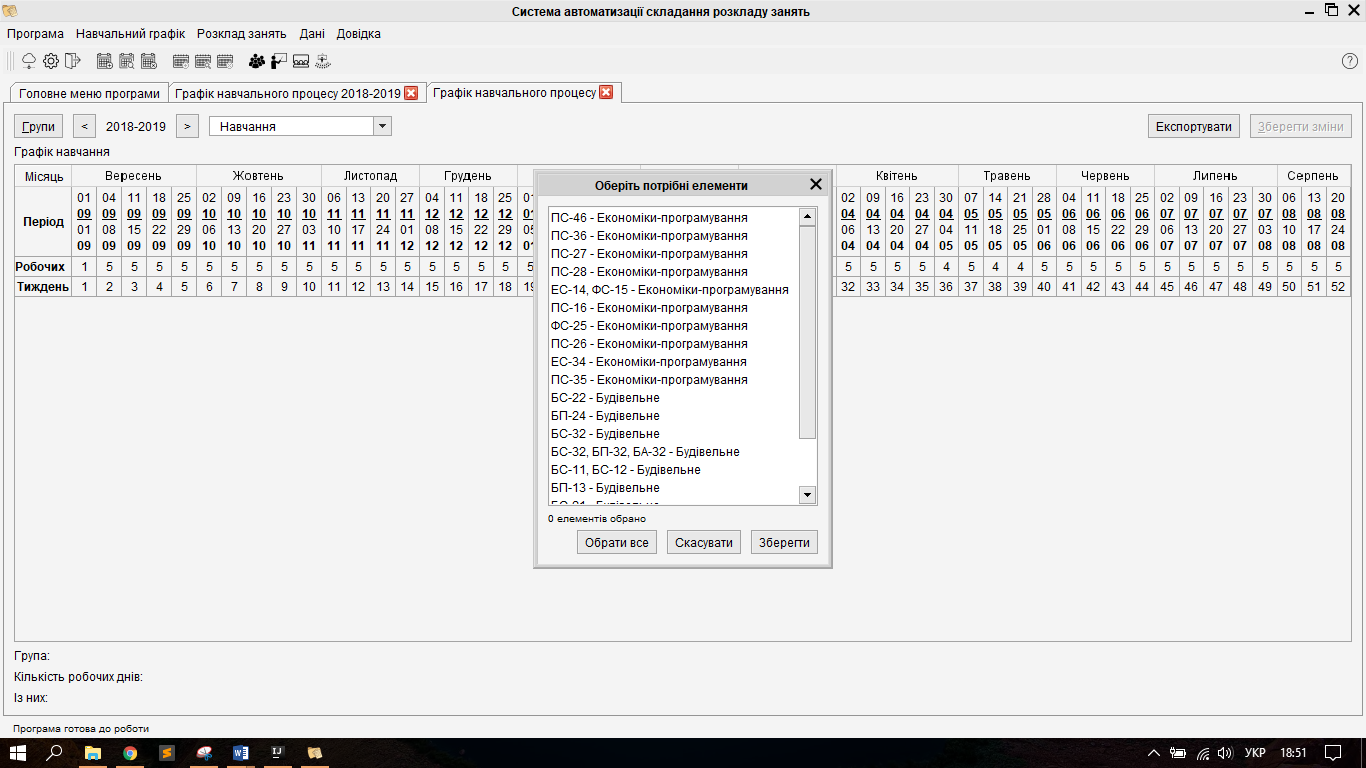


Рисунок В.13 – Головна форма в режимі створення графіку навчального процесу

При відкритті буде запропоновано обрати групи, що будуть включені в графік навчального процесу. Після того як потрібні групи обрані, потрібно натиснути «Зберегти». Далі групи будуть перенесені до графіку, як це показано на рисунку В.14.

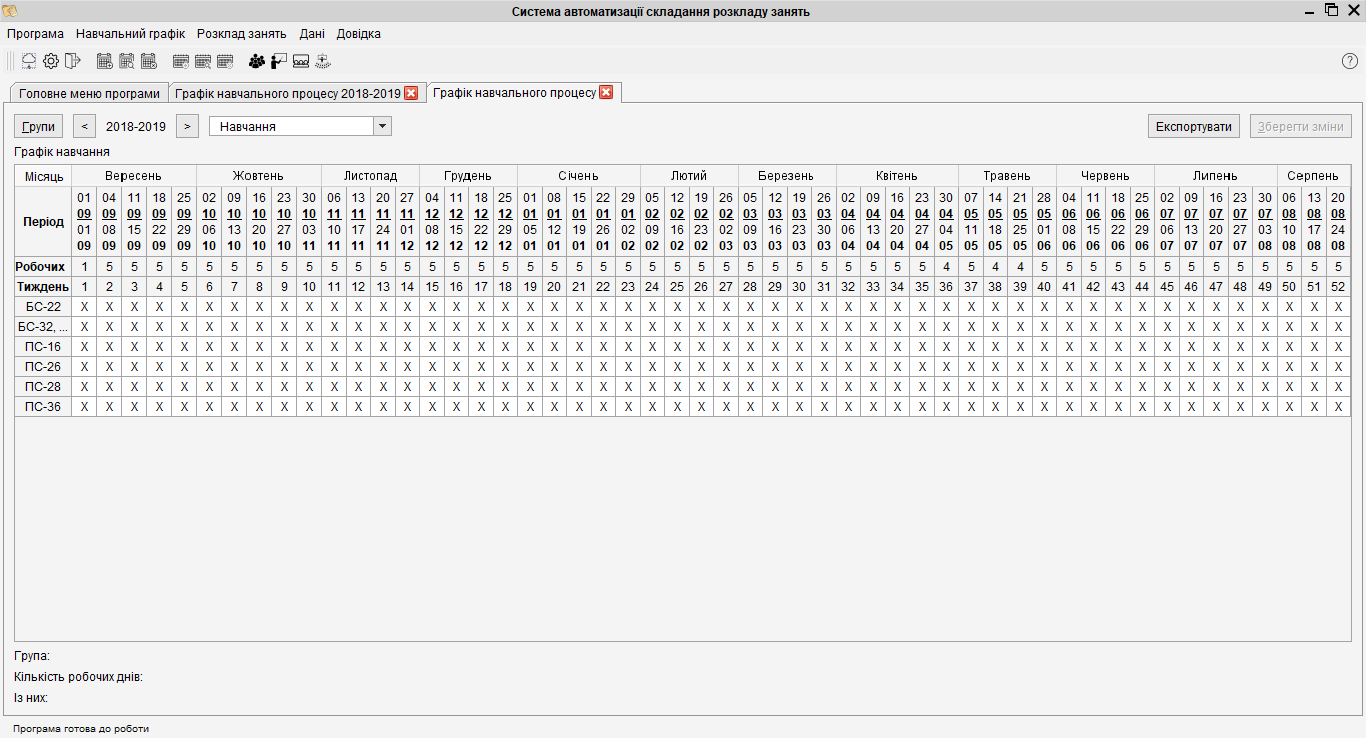


Рисунок В.14 – Групи додані до графіку навчального процесу

Після додання груп можна виконувати створення графіку. Для цього обираємо потрібний період і переносим на відповідне йому місце в графіку. Процес показаний на рисунку В.15.

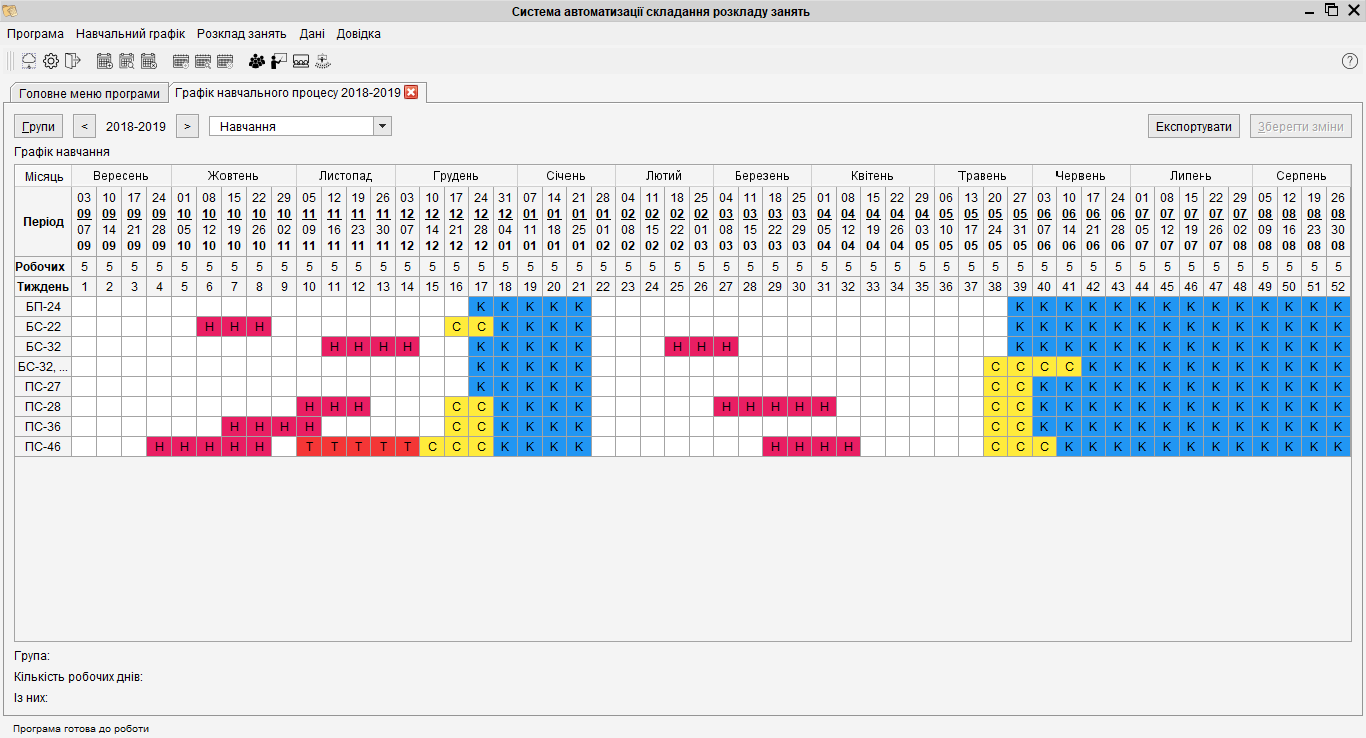


Рисунок В.15 – Процес створення графіку

Коли графік остаточно створений, можна виконати його експортування для подальшого використання. Виведення графіку із програмної системи виконується натисканням на кнопку «Експортувати». Далі буде запропоновано обрати місце для збереження, за допомогою звичайного діалогу вибору файлу. Після того як місце буде обрано і пройде процес збереження графік можна відкрити за допомогою програми Excel. Графік навчального процесу має наступний вигляд поданий на рисунку В.16.

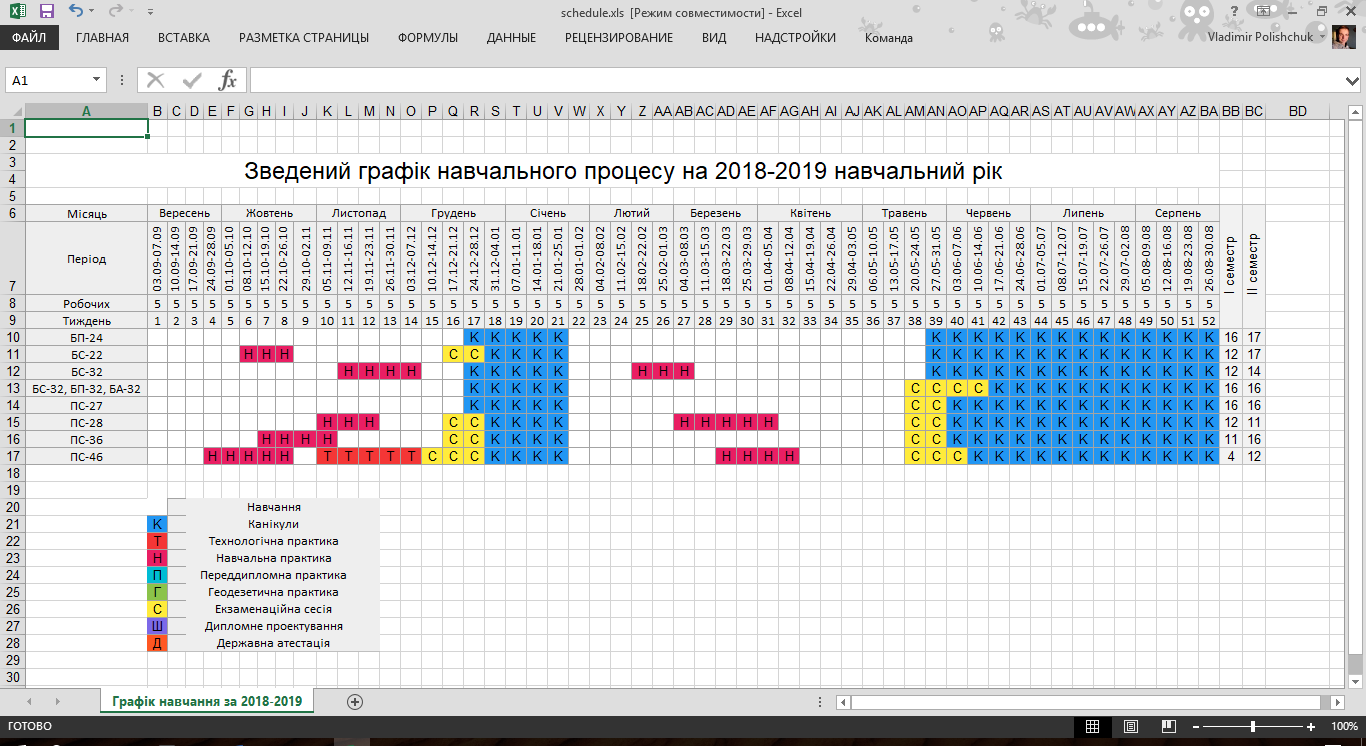


Рисунок В.16 – Графік навчального процесу в Excel

**2.4 Робота з розкладом занять**

Для створення розкладу занять потрібно перейти до головного меню програми і обрати пункт «Розклад занять». Після цього розкриється меню і потрібно обрати пункт «Створення розкладу занять». Процес створення показаний на рисунку В.17.

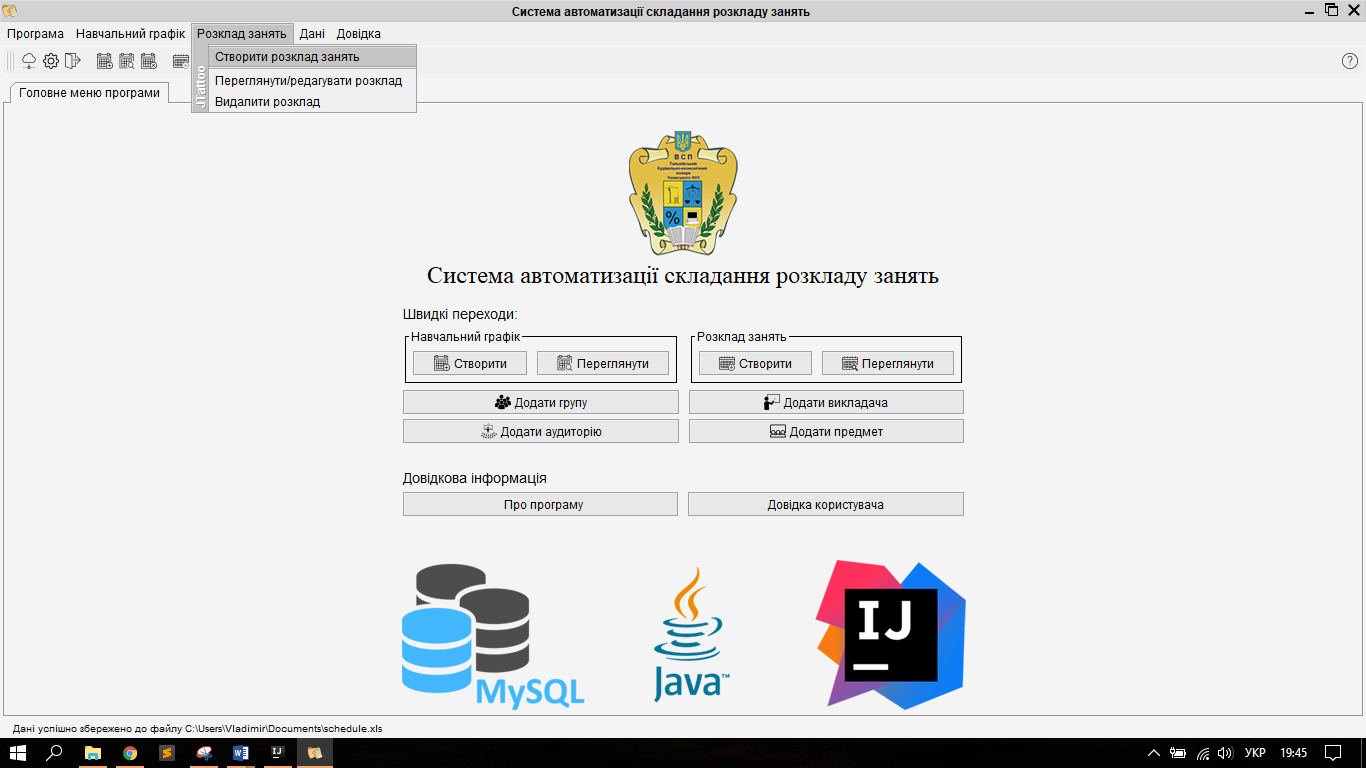


Рисунок В.17 – Створення розкладу занять

Після натиснення відбудеться процес відкриття нової закладки з набором інструментів для створення розкладу занять. Також автоматично відкриється вікно вибору груп, що будуть включені в розклад занять (рисунок В.18).

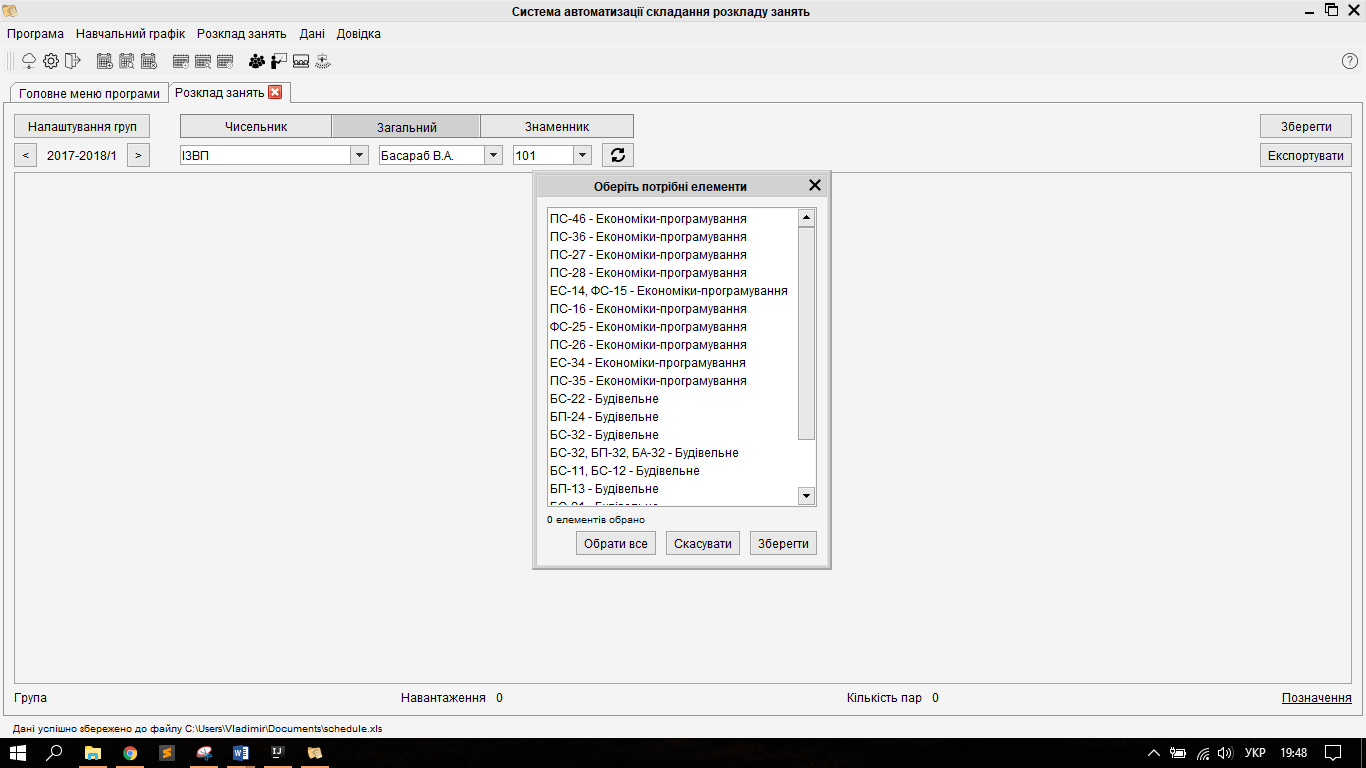


Рисунок В.18 – Пустий розлад занять

Для того щоб продовжити потрібно обрати потрібні групи і натиснути кнопку «Зберегти». Обрані групи будуть додані до розкладу занять. Далі потрібно виконувати заповнення розкладу занять, використовуючи елементи управління. Заповнений для прикладу розклад можна переглянути на рисунку В.19.

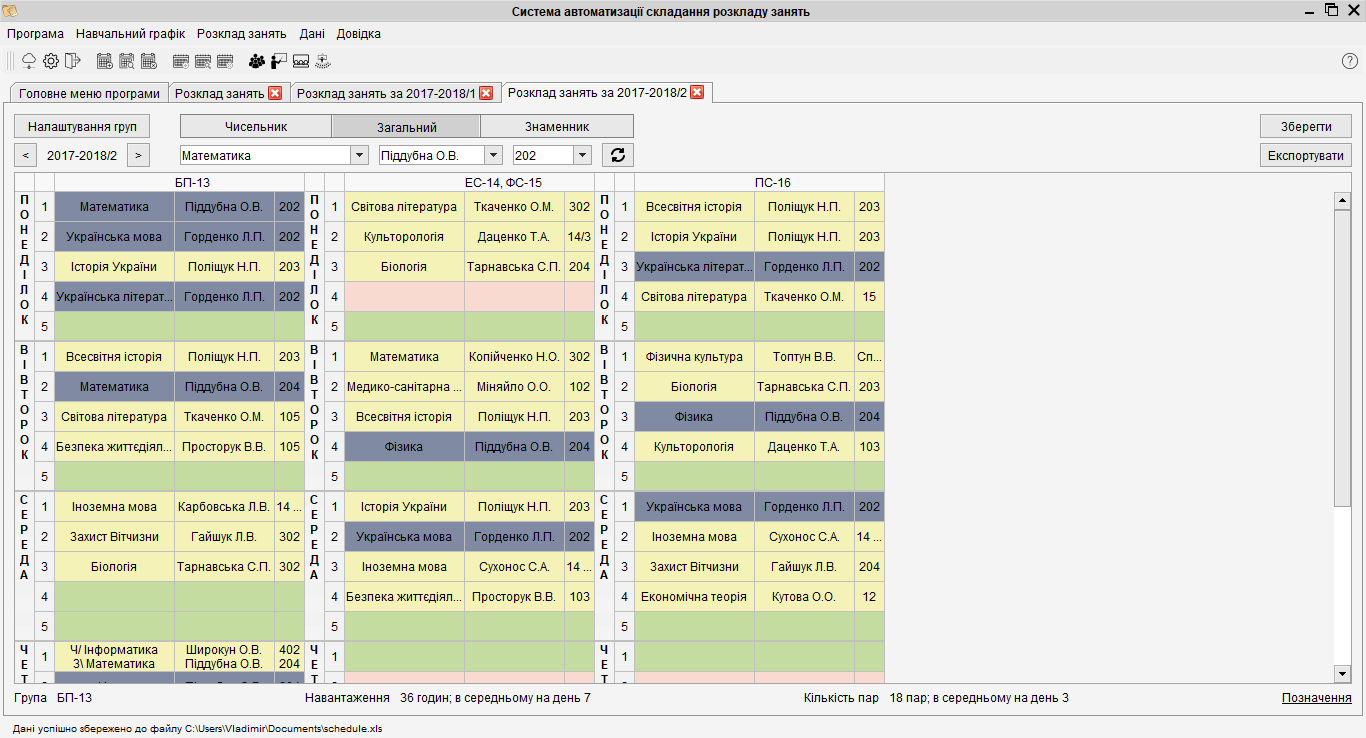


Рисунок В.19 – Створення розкладу занять

Після заповнення розкладу занять можна виконати його експортування до іншого формату збереження. Для цього виконання потрібно натиснути на кнопку «Експортувати». Після натиснення відкриється діалог, в якому буде запропоновано обрати місце для збереження файлу. Після того як файл буде збережено, його можна відкрити за допомогою Excel. Експортований розклад занять в Excel має наступний вигляд поданий на рисунку В.20.

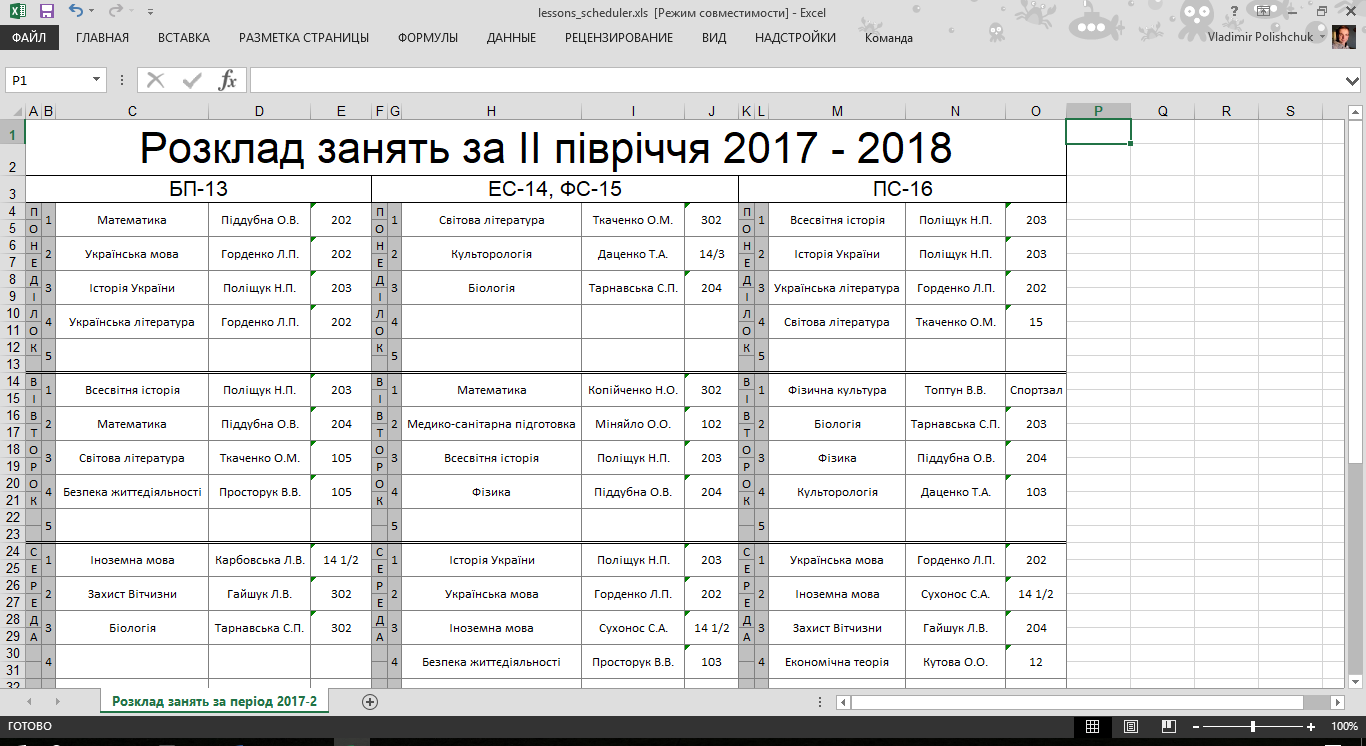


Рисунок В.20 – Розклад занять у форматі Excel