РЕФЕРАТ

Розрахунково-пояснювальна записка складається з 9 розділів:

* вступ – в даному розділі описується сутність розробки, її актуальність. Складається з 2 сторінок;
* призначення, постановка задачі та огляд програмних аналогів і літератури – у цьому розділі описано призначення, постановка задачі, проведений огляд аналогів і літератури, а також аналіз існуючих систем. Складається з 30 сторінок;
* зовнішнє та логічне проектування – у цьому розділі проведений огляд вхідних і вихідних даних, формалізація задачі, розробка структур файлів і структури правил переходу. Складається з 5 сторінок;
* внутрішнє проектування – у цьому розділі приводиться опис об’єктно-орієнтованого проектування, проектування інтерфейсу користувача, ескізи форм, проектування динаміки системи, вибір мови програмування. Складається з 21 сторінки;
* тестування та налагодження – включає в себе вибір об’єкту та методів тестування, результати тестування та налагодження програми. Складається з 5 сторінок;
* безпека праці користувача при роботі на ЕОМ. Складається з 12 сторінок;
* висновки. Складається з 2 сторінок;
* список літератури – включає в себе бібліографічний список використаної літератури. Складається з 1 сторінки;
* додатки – містить технічне завдання і робочий проект.

Кількість таблиць: 23.

Кількість рисунків: 56.

ЗМІСТ

[1 Огляд проблеми 9](#_Toc336324424)

[2 зовнішнє та логічне проектування 19](#_Toc1350072917)

[2.1 Зовнішнє проектування 19](#_Toc2123954721)

[2.1.1 Опис функціональних характеристик 19](#_Toc654706222)

[2.1.2 Вхідні дані 20](#_Toc248894929)

[3 ТЕСтування та налагодження програми 85](#_Toc874105541)

[3.1 Вибір стратегії тестування 85](#_Toc1601119566)

[Для досягнення максимальної якості тестування під кожен метод або модуль програми, що тестується необхідно обрати найбільш вдалий метод або набір методів, що забезпечать необхідний результат. При цьому необхідно підібрати найкраще співвідношення між часом, що буде витрачено на тестування, та якістю тестування. Набір тестів повинен мати мінімальну збитковість, але при цьому максимально охоплювати функціональність системи [16]. 85](#_Toc1718047139)

[4 Охорона праці та безпека у надзвичайних ситуаціях 108](#_Toc1929656551)

[5.1 Аналіз шкідливих та небезпечних виробничих факторів 108](#_Toc1890703606)

[5.2 Організаційні і технічні заходи щодо захисту користувачів від шкідливих і небезпечних факторів. 111](#_Toc295466787)

[5.3 Норми пожежної безпеки на робочому місці користувача 112](#_Toc1854568120)

[5.4 Вимоги безпеки праці 115](#_Toc1821393951)

[5.5 Дії при аварійних ситуаціях 117](#_Toc1967533318)

[5.5.1 Допомога при уражені електричним струмом 117](#_Toc130599929)

[висновки 120](#_Toc369124034)

[Після закінчення розробки системи, була написана уся необхідна програмна документація. Наявність якісного керівництва користувача дозволить методистам учбового відділу, що будуть використовувати систему, швидко вивчити можливості програми та почати її використовувати. Задокументований опис та текст програми спростить подальше супроводження системи. 121](#_Toc2064686966)

[ЛІТЕРАТУРА 122](#_Toc91811583)

ЗМІСТ (old)

[Вступ 7](#_Toc421172491)

[1 Огляд проблеми 8](#_Toc421172492)

[1.1 Призначення та область застосування 8](#_Toc421172493)

[1.2 Постановка задачі 8](#_Toc421172494)

[1.3 Огляд програмних аналогів 9](#_Toc421172495)

[1.3.1 Особливості автоматизованої системи «Розклад занять» 9](#_Toc421172496)

[1.3.2 Особливості системи «Галактика. Управление ВУЗом» 10](#_Toc421172497)

[1.3.3 Особливості автоматизованої системи «Навантаження ВНЗ» 10](#_Toc421172499)

[1.3.4 Особливості АІСУ «Навчальне навантаження» 12](#_Toc421172500)

[1.4 Огляд літературних джерел 13](#_Toc421172500)

[2 Зовнішнє та логічне проектування 17](#_Toc421172500)

[2.1 Зовнішнє проектування 17](#_Toc421172501)

[2.1.1 Опис функціональних характеристик 17](#_Toc421172502)

[2.1.2 Вхідні дані 18](#_Toc421172503)

[2.1.3 Вихідні дані 28](#_Toc421172504)

[2.2 Проектування бази даних 29](#_Toc421172505)

[2.2.1 Специфікація вимог до бази даних 29](#_Toc421172506)

[2.2.2 Визначення типів сутностей та зв’язків між ними 29](#_Toc421172507)

[2.2.3 Логічне проектування бази даних](#_Toc421172508) 57

[2.2.4 Схема бази даних 59](#_Toc421172514)

[2.3 Вибір мови програмування](#_Toc421172515) [64](#_Toc421172515)

[2.3.1 Опис платформи .NET Framework 64](#_Toc421172515)

[2.3.2 Опис мови розробки програмного продукту – C# 68](#_Toc421172515)

[2.4 Об’єктно-орієнтоване проектування 69](#_Toc421172516)

[2.5 Розробка інтерфейсу користувача 78](#_Toc421172516)

[3 Тестування та налагодження програми 80](#_Toc421172500)

[3.1 Вибір стратегії тестування 80](#_Toc421172500)

[3.2 Тестування функціональності 80](#_Toc421172500)

[3.2.1 Тестування методу ReadDocFromExcel класу CurriculumDocService 80](#_Toc421172500)

[3.2.2 Тестування методу FormationDocs класу WorkingCurriculumDocService 96](#_Toc421172500)

[3.2.3 Тестування методу Formation класу LoadDepartmentService 98](#_Toc421172500)

[3.2.4 Результати тестування 99](#_Toc421172500)

[3.3 Налагодження програми 101](#_Toc421172500)

[4 Охорона праці та безпека у надзвичайних ситуаціях 103](#_Toc421172500)

[4.1](#_Toc421172500) [Аналіз шкідливих та небезпечних виробничих факторів 104](#_Toc421172500)

[4.2](#_Toc421172500) [Організаційні та технічні заходи по усуненню виявлених шкідливих та  
 небезпечних виробничих факторів 106](#_Toc421172500)

[4.2.1 Вимоги щодо забезпечення пожежної безпеки 107](#_Toc421172500)

[4.2.2 Вимоги до електробезпеки 108](#_Toc421172500)

[4.2.3 Вимоги до організації робочого місця 109](#_Toc421172500)

[4.3 Правила безпечного виконання робіт методистами навчального відділу  
 при роботі за комп’ютером 110](#_Toc421172500)

[4.4 Дії у разі настання надзвичайної ситуації 111](#_Toc421172500)

[4.4.1 Дії у разі виявлення пожежі 112](#_Toc421172500)

[Висновки 114](#_Toc421172500)

[Література 116](#_Toc421172500)

[Додатки 118](#_Toc421172500)

ВСТУП

Програмний засіб «Система доступу до енциклопедичних знань на природній мові» призначений для отримання енциклопедичних знань у режимі діалогу за допомогою природньої мови.

Причиною виникнення необхідності розробки є відсутність програми, яка б виконувала дану задачу з точністю, достатньою для ефективного повсякденного користування. Традиційно дана задача вирішувалася за допомогою звичайних пошукових систем та формулювання питання у виді ключових слів.

Програмне забезпечення дозволить використовувати енциклопедичну інформацію з навчальною або довідковою метою. Інтерфейс взаємодії з системою реалізований через систему текстових або голосових запитів та відповідей. Це стане у нагоді як звичайним користувачам, так і користувачам з обмеженими можливостями.

**щось про інтеграцію з начальним процесом (“Прометей”)**

Функціональне призначення продукту полягає у наданні користувачу можливості за текстовим або голосовим запитом отримати інформацію, що міститься в енциклопедичних системах.

## 2.2 Експлуатаційне призначення

Розроблюваний програмний продукт призначений для:

* швидкого доступу до енциклопедичних знань у навчанні та роботі;
* надання альтернативного мовного інтерфейсу для людей з проблемами зору;
* збільшення інтересу до знань за рахунок використання перспективних технологій.

**Програмний засіб «Система доступу до енциклопедичних знань на природній мові» призначений призначений для отримання знань, що містяться в системі, у режимі діалогу за допомогою природньої мови.**

**Головною метою розробки є спрощення складності та оптимізація процесу пошуку нової інформації енциклопедичного характеру. Розроблюване програмне забезпечення може використовуватися будь-яким користувачем з навчальною або довідковою метою; оскільки ПЗ планується виконати у форматі веб-додатку, у ролі клієнта може виступати будь-який користувач з доступом до інтернету.**

**Впровадження програмного засобу дасть змогу** **отримати швидкий доступ до енциклопедичних знань у навчанні та роботі, збільшити інтерес до знань за рахунок використання перспективних технологій та надати альтернативний мовний інтерфейс для людей з проблемами зору.**

Програмний засіб «Система доступу до енциклопедичних знань на природній мові» призначений для складання робочих навчальних планів та розподілу навчального навантаження кафедр, адже саме на основі цих документів формується розклад занять.

Причиною виникнення необхідності розробки є відсутність програми, яка б автоматизувала процес формування первинної документації. Традиційно дана задача вирішувалася в ручному режимі.

Складання розкладу навчальних занять безпосередньо пов'язане з робочими навчальними планами, розподілом навантаження на кафедрах і наявним аудиторним фондом. В університеті складанням розкладу занять займається навчальний відділ. За навчальним планом університету складається робочий навчальний план для кожного курсу за фахом і навчальним навантаженням по кафедрі. Використовуючи ці документи, для кожного семестру методист навчального відділу складає документ «Розподіл навчального навантаження по кафедрі» для кожної кафедри. Методист вказує дисципліни, групи, навчальне навантаження по годинах для кожного виду заняття: лекція, практичне заняття чи лабораторна робота. Потім, цей документ відправляється на кафедри університету, де заповнюється викладачем, котрий займається розподілом навчального навантаження викладачів по кафедрі. В документі для кожного виду занять вказуються викладачі, що будуть їх вести.

Після закінчення збору інформації методисти навчального відділу складають розклад.

Для підвищення якості розкладу, і зменшення витрат часу пропонується автоматизувати процес формування первинної документації.

Програмний продукт може використовуватися в вищих навчальних закладах.

1 Огляд проблеми

1.1 Призначення та область застосування

Автоматизована система використовуватиметься у навчальному відділу університету та призначена автоматизувати процеси формування документації для складання розкладу занять університету.

Функціональним призначенням системи є Система доступу до енциклопедичних знань на природній мові, а саме робочих навчальних планів та розподілу навчального навантаження по кафедрам. Також надання необхідної інформації для складання розкладу, а саме перелік: викладачів, дисциплін, аудиторій, груп, вимоги викладачів, тощо.

Експлуатаційним призначенням програмного комплексу є підвищення якості розкладу; автоматизація процесу формування робочих навчальних планів; розподіл навантаження кафедр та складання розкладу занять, що призведе до скорочення часу на виконання роботи; перехід від «паперового» документообігу до електронного.

1.2 Постановка задачі

Необхідно дослідити процеси пов’язані з формуванням документації для складання розкладу занять університету, виявити недоліки та запропонувати методи їх вирішення.

Розробити автоматизовану систему, що використовуватиметься у навчальному відділі для формування первинної документації і її передання в диспетчерську для складання розкладу занять з такими функціями:

* формування робочих навчальних планів на основі навчальних планів в автоматичному режимі в форматі \*.xls;
* складання документів «Очікуване навантаження кафедри», в автоматичному режимі в форматі \*.xls;
* формування документів «Розподіл навчального навантаження кафедр» для надання диспетчерам, в автоматичному режимі в форматі \*.xls;
* формування шаблонів для документів: «Навчальний план», «Робочий навчальний план», «Очікуване навантаження кафедри», «Розподіл навчального навантаження кафедри», «Вимоги викладачів» в форматі \*.xls;
* завантаження готових навчальних планів, робочих навчальних планів, навантаження кафедр, вимог викладачів і на основі цих документів заповнення бази даних;
* адміністрування (створення, редагування, видалення) інформації, яка необхідна для розкладу занять та первинної документації, а саме перелік: викладачів, кафедр, факультетів, спеціальностей, груп, дисциплін, аудиторій, вимоги викладачів, тощо;
* експорту бази даних для системи складання розкладу занять;
* імпорту бази даних системи складання розкладу занять;
* експорту довідкової інформації про викладачів, спеціальності, групи тощо в файли в форматі \*.xls.

1.3 Огляд програмних аналогів

Як правило системи формування навчальної документації університету є частиною багатофункціональних систем управління навчальним процесом, а не окремими системами, якою буде даний програмний комплекс. Ускладнює огляд аналогів те, що більшість знайдених систем є платними, а тому не має достатньої кількості інформації для їх огляду.

1.3.1 Особливості автоматизованої системи «Розклад занять»

Розроблена працівниками Дніпропетровського національного університету залізничного транспорту ім. академіка В. Лазаряна [1].

Система складається з трьох основних підсистем: «Розклад занять», «Розклад модульного контролю», «Прийом обробка та формування документації» [1].

Підсистема «Прийом обробка та формування документації» повинна мати функції щодо формування документації, на жаль дана підсистема закладена, але майже не реалізована. Реалізовані лише функції формування шаблонів для документів «Вимоги викладачів» та «Розподіл навантаження по кафедрі». Введення будь якої інформації відбувається лише в ручному режимі.

Система написана на мові програмування Delphi, з використанням бази даних Borland Interbase. Система не є завершеною [1].

1.3.2 Особливості інформаційної системи «Галактика. Управление ВУЗом»

Комерційна система охоплює багато аспектів роботи вищого навчального закладу [2]. Містить модуль по управлінню навчальним процесом, який надає наступні функції: формування навчальних планів, планування навантаження кафедр, планування навантаження викладачів, підготовка розкладу на семестр.

Всі дані в програму можуть потрапляти лише в ручному режимі.

Також недоліком системи є те, що вона є цілісною, яка надає дуже велику кількість функцій, і не можна замовити лише окремий модуль, наприклад формування документації. А тому дуже багато часу витрачається на навчання персоналу користуватися даною програмою. Також інтерфейс програми є дуже навантаженим, що викликає додаткові проблеми при роботі з програмою.

Система створена в середовищі розробки Microsoft .NET з використанням сервіс-орієнтованої архітектури (SOA), може працювати, як десктоп додаток чи як web сервіс [2].

Оскільки система є платною та не передбачає демонстраційних версій, більше про неї інформації знайдено не було.

1.3.3 Особливості автоматизованої системи «Навантаження ВНЗ»

АС «Навантаження ВНЗ» – комплекс програмного забезпечення для автоматизації адміністративної роботи кафедр ВНЗ, розроблена – лабораторією математичного моделювання інформаційних систем. Це програмне забезпечення передбачає надання комплексного підходу до формування та розподілу навчального навантаження. Система розрахована для роботи в локальній мережі і має три рівні доступу, які визначають функціонал доступний користувачам [3].

Адміністратору системи доступні наступні функції [3]:

* перевірка навчальних планів на спадкоємність;
* формування відомостей про очікуваний контингент студентів;
* створення списку навчальних груп на основі контингенту студентів;
* централізоване перейменування дисциплін та закріплення їх за кафедрами;
* визначення параметрів формування потоків і навчального навантаження;
* формування навчального навантаження кафедр на базі навчальних планів та списку груп;
* розрахунок штатного розкладу кафедр.

Завідувачам кафедрами доступні наступні функції [3]:

* закріплювати навчальне навантаження за викладачем або передавати його іншій кафедрі;
* об'єднувати групи в потоки і розбивати групу на підгрупи з різних видів занять, наприклад, лабораторних, практичних або курсових робіт;
* враховувати дані про розподіл навчального навантаження попереднього навчального року;
* уточнювати навантаження протягом навчального року;
* виконувати перевірку коректності та відповідності розподіленого навантаження.

Викладачам доступні наступні функції [3]:

* отримувати розгорнуту інформацію з навчального навантаження;
* заповнювати необхідну кількість годин на підготовку до занять;
* формувати свої індивідуальні плани, в яких можна конкретизувати зміст робіт, виконуваних у «другій половині дня» (наукова, організаційно-методична, навчально-методична, виховна робота);
* вести облік побажань викладачів за графіком роботи протягом тижня.

АС «Навантаження ВНЗ» має клієнт-серверну архітектуру і припускає наявність сервера баз даних Microsoft SQL Server версії 2000 і вище [3].

Рекомендується використовувати MS SQL Server 2005. Установку необхідно проводити на тому комп'ютері, який передбачається використовувати як сервер баз даних [3].

Головним недоліком даного програмного продукту є те, що вся інформація в систему потрапляє за допомогою ручного введення.

1.3.4 Особливості АІСУ «Навчальне навантаження»

Проект АІСУ «Навчальне навантаження» реалізується у Черкаському національному університеті імені Богдана Хмельницького з 2007 року [4].

АІСУ «Навчальне навантаження» складається з підсистем [4]:

1) «Навчальний план»;

2) «Робочий навчальний план»;

3) «Розрахунок навчального навантаження»;

4) «Навчальне навантаження кафедри»;

5) «Навчальне навантаження викладача».

Підсистема «Навчальний план» відповідає за розробку навчальних планів за напрямами, спеціальностями та спеціалізаціями. Головними складовими підсистеми є: блок генерації графіка навчального процесу, блок формування множини навчальних дисциплін та блок визначення форм та видів державної атестації [4].

На основі даних отриманих з підсистеми «Навчальний план», з урахуванням кількісних характеристик контингенту студентів, підсистема «Робочий навчальний план» у автоматичному режимі формує робочий навчальний план для поточного курсу студентів поточного року навчання [4].

У автоматичному режимі функціонує також підсистема «Розрахунок навчального навантаження». Система аналізує та нормує усі види навчального навантаження, а також розподіляє між його елементами частки ставки, що утворюються контингентом студентів згідно визначених державою нормативів [4].

До функціональних можливостей підсистеми «Навчальне навантаження кафедри» належать: визначення кафедри, яка виконуватиме поточне навчальне навантаження; розподіл будь-яких складових навчального навантаження між декількома кафедрами; розрахунок кількості викладацьких ставок. Дана підсистема працює за рахунок інтегрованих даних отриманих з підсистем «Робочий навчальний план» та «Розрахунок навчального навантаження» [4].

Підсистема «Навчальне навантаження викладача» є завершальною складовою системи планування навчального навантаження. ЇЇ основним призначенням є розподіл навантаження, що отримала кафедра, між викладачами [4].

Система управління навчальним навантаженням створювалась виключно засобами мови програмування PHP та СУБД MySql і має відкритий об’єктно-орієнтований програмний код. Вона реалізує дворівневу модель архітектури клієнт-сервер, в якій клієнт звертається до послуг серверу. Клієнтська частина (діалогові компоненти, засоби візуалізації) генерується на сервері і передається користувачеві у HTML, XLS або PDF форматі. Усі компоненти управління даними: операції з базами даних і файлові операції, бізнес логіка і логіка управління даними розміщуються на сервері [4].

1.4 Огляд літературних джерел

Навчальний план є базовим документом, що визначає зміст професійної підготовки фахівця. В ньому реалізуються основні принципи відбору предметів, їх систематизація, регламентується об’єм навчальних дисциплін, навантаження студента за періодами навчання, форми підсумкового контролю знань. На структуру і зміст навчального плану впливають освітньо-кваліфікаційна характеристика і освітньо-професійна програма за відповідними напрямами і спеціальностями.

В умовах переходу до трирівневої системи вищої освіти та постійного зростання кількості спеціальностей, які запроваджують вищі навчальні заклади, виникає проблема контролю якості розробки навчальних планів та їх подальшого використання у процесі формування робочого навчального плану, розподілу навчального навантаження між кафедрами і викладачами. Використання традиційних способів стало трудомістким і довготривалим, а отже, виникла необхідність в дослідженні можливостей вирішення зазначених проблем шляхом використання інформаційно-комунікаційних технологій [4].

Проблема ефективного використання інформаційно-комунікаційних технологій в освіті в цілому та у вищій зокрема, вже досить широко висвітлена у педагогічній науці. В останні роки активно досліджуються можливості використання інформаційних технологій в управлінні навчальним процесом [5, 6, 7, 8].

Останнім часом у багатьох ВНЗ України почали впроваджувати локалізовані засоби автоматизації окремих напрямів управлінської діяльності (системи обліку документів та кореспонденції, системи обліку кадрів, системи управління бібліотеками, системи бухгалтерського обліку, тощо). Але неможливість синхронізації дій між різними системами, відмінності у представлені інформації, і як наслідок, складність передачі даних між ними, породжують нові проблеми, пов’язані з надлишковою інформацією, неоперативністю отримання потрібних даних, фрагментарністю виконання єдиних процесів. Все це зводить нанівець намагання суб’єктів освітньої діяльності перекласти її рутинні процеси на засоби інформаційно-комунікаційних технологій.

Найбільш вдалим рішенням у цій ситуації є використання сучасних мережевих технологій і єдиного сховища даних, що надає можливість реалізувати простий механізм інтеграції інформації в єдиний інформаційний ресурс вищого навчального закладу і забезпечити можливість сумісного використання інформації з урахуванням механізму розмежування доступу (з метою захисту даних) багатьма користувачами: адміністрацією ВНЗ, викладачами, студентами різних форм навчання, абітурієнтами. Це надає можливість досягти високого рівня цілісності даних і створюються умови для реалізації комплексної автоматизованої системи управління навчальним процесом університету.

Задля спрощення процесів формування похідних від навчального плану документів, пропонується заздалегідь закладати додаткову, неактуальну у поточний момент, інформацію [4].

Так, зокрема, у процесі формування переліку спеціальностей, пропонується додавати нормативний параметр кількості студентів для утворення однієї ставки викладача, який буде використано під час розподілу кількості ставок між формами та елементами навчального навантаження [4].

На етапі формування переліку навчальних дисциплін крім основних складових, зокрема, загальної кількості годин, кількості лекційних, лабораторних та практичних годин, тощо, пропонується вводити кілька додаткових параметрів: статус дисципліни, об’єднання дисциплін у потік та ін. [4].

Усі додаткові параметри дисциплін навчального плану, крім статусу дисципліни, є неактуальними у процесі формування навчального плану, але знадобляться у процесі автоматичного формування робочого навчального плану та під час оптимізації навчального навантаження викладачів [4].

Статус дисципліни є важливим і визначальним параметром для чіткого розмежування дисциплін на державну складову та варіативну складову, це також дає можливість вносити до навчального плану необмежену кількість альтернативних дисциплін (за вибором навчального закладу, за вибором студента). Завдяки цьому параметру існує можливість надання певним дисциплінам унікальних особливостей, зокрема, це наразі стосується дисциплін «Іноземна мова» та «Фізичне виховання» [4].

Для спрощення процесу автоматичного формування робочого навчального плану, дисципліни, читання яких планується здійснювати упродовж декількох семестрів, пропонується заздалегідь розділяти на окремі складові [4].

Пропонується ще на етапі формування дисциплін закладати основи оптимального використання робочого часу викладачів за рахунок об’єднання у потоки однакових або споріднених дисциплін з різних навчальних планів, ще на стадії їх формування [4].

Завдання автоматизації складання розкладу занять у вищому навчальному закладі є багатокритеріальним, традиційно трудомістка і рутинна робота з формування сітки розкладу характеризується відсутністю єдиного джерела вхідної інформації і, як наслідок, з необхідністю ретельної підготовки, структуризації, збору та обробки великого обсягу вхідної інформації з різних структурних підрозділів ВНЗ, таких як навчальний відділ, деканати, кафедри, відділ кадрів, диспетчерська. Весь обсяг документації вести вручну дуже складно. Крім того, цей спосіб ведення документації не виключає «людського фактору», що може призвести до помилок в документах. Тому виникла необхідність створення автоматизованої системи, функціональність якої зможе забезпечити виконання основних задач університету.

У результаті аналізу існуючого програмного забезпечення автоматизації управління навчальним процесом, зокрема формування документації виявлено, що на даний момент, велика частина вузів використовує часткову автоматизацію, тобто комп’ютеризовані лише окремі бізнес процеси. Це веде до дублювання і втрати частини інформації, відсутності даних для оперативного аналізу, втрати часу на обслуговування і підтримку застаріваючих технологій, високого ступеню залежності від людського чинника.

Існують готові програмні рішення задачі формування документації, але більшість з них не дозволяють робити це в автоматичному режимі, без ручного введення даних. Деякі вдалі рішення вирізняють окремі системи від інших: веб-інтерфейс доступу до системи дозволяє працювати з системою на будь-якому комп’ютері без попередньої установки програми [2-3]; різні рівні доступу до системи [2], дозволяє розмежувати права роботи з програмою для різних користувачів, наприклад, надання викладачам часткового доступу до системи, для введення побажань щодо свого навантаження та розкладу, з метою зменшення навантаження на навчальний відділ; введення додаткових параметрів для більш ефективного формування документів на основі навчального плану, які можуть бути використані при формуванні наступних документів [4], що дозволить пришвидшити та покращити процес формування.

Зважаючи на все вищесказане, була розроблена система автоматизації формування первинної документації, а саме робочих навчальних планів та розподіл навчального навантаження по кафедрам. Розробка та впровадження дозволить прискорити формування необхідної документації, поліпшити якість розкладу, перейти від «паперового» документообігу до електронного. Планується використовувати дану програму разом з системою складання розкладу занять, яка на основі отриманих вихідних даних буде формувати розклад занять університету.

2 зовнішнє та логічне проектування

2.1 Зовнішнє проектування

2.1.1 Опис функціональних характеристик

Програмний комплекс призначений для Система доступу до енциклопедичних знань на природній мові, а саме робочих навчальних планів та розподілу навчального навантаження по кафедрам. Також надає необхідну інформацію для складання розкладу, а саме перелік: викладачів, дисциплін, аудиторій, груп, вимоги викладачів, тощо.

Програмний продукт повинен надавати можливість:

* складати робочі навчальні плани на основі навчальних планів в автоматичному режимі в форматі \*.xls;
* формувати очікувані навчальні навантаження кафедр в автоматичному режимі в форматі \*.xls;
* складати документи «Розподіл навчального навантаження кафедр» для надання диспетчерам, в автоматичному режимі в форматі \*.xls;
* формувати шаблони для документів: «Навчальний план», «Робочий навчальний план», «Очікуване навантаження кафедри», «Розподіл навчального навантаження кафедри», «Вимоги викладачів» в форматі \*.xls;
* завантажувати готові навчальні плани, робочі навчальні плани, навантаження кафедр і на основі цих документів заповнювати базу даних;
* адміністрування (створення, редагування, видалення) інформації, яка необхідна для розкладу занять та первинної документації, а саме перелік: викладачів, кафедр, факультетів, спеціальностей, груп, дисциплін, аудиторій, вимоги викладачів, тощо;
* експорт бази даних для системи складання розкладу занять;
* імпорт бази даних системи складання розкладу занять;
* експорт довідкової інформації про викладачів, спеціальності, групи тощо в файли в форматі \*.xls.

Оскільки передбачається використання програми користувачами без професійних навичок роботи з ПК, необхідно створити програму-інсталятор, що допоможе користувачу встановити систему на свій комп’ютер, та зручний інструмент налагодження доступу до бази даних.

2.1.2 Вхідні дані

Вхідними даними програми, що розробляється, є:

* інформація про факультети (табл. 2.1);
* дані про посади (табл. 2.2);
* інформація про форми навчання (табл. 2.3);
* дані про освітньо-кваліфікаційні рівні (табл. 2.4);
* інформація про дисципліни (табл. 2.5);
* відомості про напрями підготовки (табл. 2.6);
* дані про галузі знань (табл. 2.7);
* інформація про типи аудиторій (табл. 2.8);
* відомості про корпуси (табл. 2.9);
* дані про розташування аудиторій (табл. 2.10);
* інформація про кафедри (табл. 2.11);
* відомості про викладачів (табл. 2.12);
* дані про спеціальності (табл. 2.13);
* інформація про типи практик (табл. 2.14);
* дані про тривалість практик (табл. 2.15);
* інформація про групи (табл. 2.16);
* дані про аудиторії (табл. 2.17);
* відомості про те, яка кафедра, які дисципліни яким спеціальностям читає (табл. 2.18);
* вимоги викладачів в форматі \*.xls (рис. 2.1);
* навчальні плани в форматі \*.xls (рис. 2.2, рис. 2.3);
* робочі навчальні плани в форматі \*.xls (рис. 2.4);
* очікувані навчальні навантаження кафедри в форматі \*.xls (рис. 2.5);
* документи «Розподіл навчального навантаження кафедри» в форматі \*.xls (рис 2.6);
* база даних з системи складання розкладу занять університету.

Таблиця 2.1 – Факультети

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Назва | Абревіатура | Номер | Архів | Дата занесення в архів |
|  |  |  |  |  |

Примітки:

Назва – назва факультету, рядок.

Абревіатура – коротка назва факультету, рядок.

Номер – номер факультету, ціле число.

Архів – чи є дана інформація застарілою, так/ні.

Дата занесення в архів – дата занесення інформації в архів, дата.

Таблиця 2.2 – Посади

|  |  |  |  |
| --- | --- | --- | --- |
| Назва | Абревіатура | Архів | Дата занесення в архів |
|  |  |  |  |

Примітки:

Назва – назва посади, рядок.

Абревіатура – коротка назва посади, рядок.

Архів – чи є дана інформація застарілою, так/ні.

Дата занесення в архів – дата занесення інформації в архів, дата.

Таблиця 2.3 – Форми навчання

|  |  |  |
| --- | --- | --- |
| Назва | Архів | Дата занесення в архів |
|  |  |  |

Примітки:

Назва – назва форми навчання, рядок.

Архів – чи є дана інформація застарілою, так/ні.

Дата занесення в архів – дата занесення інформації в архів, дата.

Таблиця 2.4 – Освітньо-кваліфікаційні рівні

|  |  |  |
| --- | --- | --- |
| Назва | Архів | Дата занесення в архів |
|  |  |  |

Примітки:

Назва – назва форми навчання, рядок.

Архів – чи є дана інформація застарілою, так/ні.

Дата занесення в архів – дата занесення інформації в архів, дата.

Таблиця 2.5 – Дисципліни

|  |  |  |  |
| --- | --- | --- | --- |
| Назва | Абревіатура | Архів | Дата занесення в архів |
|  |  |  |  |

Примітки:

Назва – назва дисципліни, рядок.

Абревіатура – коротка назва дисципліни, рядок.

Архів – чи є дана інформація застарілою, так/ні.

Дата занесення в архів – дата занесення інформації в архів, дата.

Таблиця 2.6 – Напрями підготовки

|  |  |  |  |
| --- | --- | --- | --- |
| Назва | Код | Архів | Дата занесення в архів |
|  |  |  |  |

Примітки:

Назва – назва напряму підготовки, рядок.

Код – код напряму підготовки, рядок.

Архів – чи є дана інформація застарілою, так/ні.

Дата занесення в архів – дата занесення інформації в архів, дата.

Таблиця 2.7 – Галузі знань

|  |  |  |  |
| --- | --- | --- | --- |
| Назва | Код | Архів | Дата занесення в архів |
|  |  |  |  |

Примітки:

Назва – назва галузі знань, рядок.

Код – код галузі знань, рядок.

Архів – чи є дана інформація застарілою, так/ні.

Дата занесення в архів – дата занесення інформації в архів, дата.

Таблиця 2.8 – Типи аудиторій

|  |  |  |
| --- | --- | --- |
| Назва | Архів | Дата занесення в архів |
|  |  |  |

Примітки:

Назва – назва типу аудиторії, рядок.

Архів – чи є дана інформація застарілою, так/ні.

Дата занесення в архів – дата занесення інформації в архів, дата.

Таблиця 2.9 – Корпуси

|  |  |  |
| --- | --- | --- |
| Назва | Архів | Дата занесення в архів |
|  |  |  |

Примітки:

Назва – назва корпусу, рядок.

Архів – чи є дана інформація застарілою, так/ні.

Дата занесення в архів – дата занесення інформації в архів, дата.

Таблиця 2.10 – Розташування аудиторій

|  |  |  |
| --- | --- | --- |
| Поверх | Блок | Корпус |
|  |  |  |

Примітки:

Поверх – поверх на якому знаходиться аудиторія, ціле число.

Блок – блок в якому знаходиться аудиторія, ціле число.

Корпус – назва корпусу відповідно до таблиці 2.9.

Таблиця 2.11 – Кафедри

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Назва | Абревіатура | Номер | Факультет | Завідуючий | Архів | Дата  занесення в архів |
|  |  |  |  |  |  |  |

Примітки:

Назва – назва кафедри, рядок.

Абревіатура – коротка назва кафедри, рядок.

Номер – номер кафедри, ціле число.

Факультет – назва факультету до якого належить кафедра, відповідно до таблиці 2.1.

Завідуючий – прізвище, ім’я, по-батькові викладача, який є завідуючим кафедрою, відповідно до таблиці 2.12.

Архів – чи є дана інформація застарілою, так/ні.

Дата занесення в архів – дата занесення інформації в архів, дата.

Таблиця 2.12 – Викладачі

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Прізвище | Ім’я | По-батькові | Кафедра | Посада | Архів | Дата занесення в архів |
|  |  |  |  |  |  |  |

Примітки:

Ім’я – ім’я викладача, рядок.

По-батькові – по-батькові викладача, рядок.

Прізвище – прізвище викладача, рядок.

Кафедра – назва кафедри відповідно до таблиці 2.11.

Посада – назва посади відповідно до таблиці 2.3.

Архів – чи є дана інформація застарілою, так/ні.

Дата занесення в архів – дата занесення інформації в архів, дата.

Таблиця 2.13 – Спеціальності

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Назва | Абревіатура | Код | Напрям  підготовки | Галузь  знань | Кафедра | Архів | Дата  занесення в архів |
|  |  |  |  |  |  |  |  |

Примітки:

Назва – назва спеціальності, рядок.

Абревіатура – коротка назва спеціальності, рядок.

Код – код спеціальності, рядок.

Напрям підготовки – назва напряму підготовки відповідно до таблиці 2.6.

Галузь знань – назва галузі знань відповідно до таблиці 2.7.

Кафедра – назва кафедри відповідно до таблиці 2.11.

Архів – чи є дана інформація застарілою, так/ні.

Дата занесення в архів – дата занесення інформації в архів, дата.

Таблиця 2.14 – Види практики

|  |  |  |
| --- | --- | --- |
| Назва | Архів | Дата занесення в архів |
|  |  |  |

Примітки:

Назва – назва практики, рядок.

Архів – чи є дана інформація застарілою, так/ні.

Дата занесення в архів – дата занесення інформації в архів, дата.

Таблиця 2.15 – Навантаження практик

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Практика | Спеціальність | Семестр | Рік | Кількість годин |
|  |  |  |  |  |

Примітки:

Практика – назва практики відповідно до таблиці 2.14.

Спеціальність – назва спеціальності відповідно до таблиці 2.13.

Семестр – номер семестру, ціле число.

Рік – рік прийому, ціле число.

Кількість годин – кількість годин на практику, ціле число.

Таблиця 2.16 – Групи

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Назва | Кількість студентів | Спеціальність | Форма навчання | Освітньо-кваліфікаційний рівень | Курс | Коледж | Архів | Дата  занесення в архів |
|  |  |  |  |  |  |  |  |  |

Примітки:

Назва – назва групи, рядок.

Кількість студентів – кількість студентів в групі, ціле число.

Спеціальність – назва спеціальності відповідно до таблиці 2.13.

Форма навчання – назва форми навчання відповідно до таблиці 2.3.

Освітньо-кваліфікаційний рівень – назва освітньо-кваліфікаційного рівня відповідно до таблиці 2.4.

Курс – курс на якому навчається група, ціле число.

Архів – чи є дана інформація застарілою, так/ні.

Дата занесення в архів – дата занесення інформації в архів, дата.

Таблиця 2.17 – Аудиторії

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Номер | Вмістимість | Кафедра | Тип | Розташування | Готовність | Архів | Дата занесення в архів | Початок неготовності | Кінець неготовності |
|  |  |  |  |  |  |  |  |  |  |

Примітки:

Номер – номер аудиторії, рядок.

Вмістимість аудиторії – кількість студентів, які можуть одночасно займатися в аудиторії, ціле число.

Кафедра – назва кафедри відповідно до таблиці 2.11.

Тип – назва типу аудиторії відповідно до таблиці 2.8.

Розташування – поверх, блок, корпус аудиторії відповідно до таблиці 2.10.

Готовність – чи є дана аудиторія доступна для проведення занять, так/ні.

Архів – чи є дана інформація застарілою, так/ні.

Дата занесення в архів – дата занесення інформації в архів, дата.

Початок неготовності – дата коли аудиторія стала неготовою до навчального процесу, дата.

Кінець неготовності – дата коли аудиторія стала готовою до навчального процесу, дата.

Таблиця 2.18 – Дисципліни з прив’язкою до кафедри та спеціальності

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Дисципліна | Кафедра | Спеціальність | Архів | Дата занесення в архів |
|  |  |  |  |  |

Примітки:

Дисципліна – назва дисципліни відповідно до таблиці 2.5.

Кафедра – назва кафедри відповідно до таблиці 2.11.

Спеціальність – назва спеціальності відповідно до таблиці 2.13.

Архів – чи є дана інформація застарілою, так/ні.

Дата занесення в архів – дата занесення інформації в архів, дата.

Вхідними документами є наступні:

* вимоги викладачів (рис. 2.1);
* навчальний план (рис. 2.2 – 2.3);
* робочий навчальний план (рис. 2.4);
* очікуване навантаження кафедри (рис. 2.5);
* розподіл навчального навантаження (рис. 2.6).

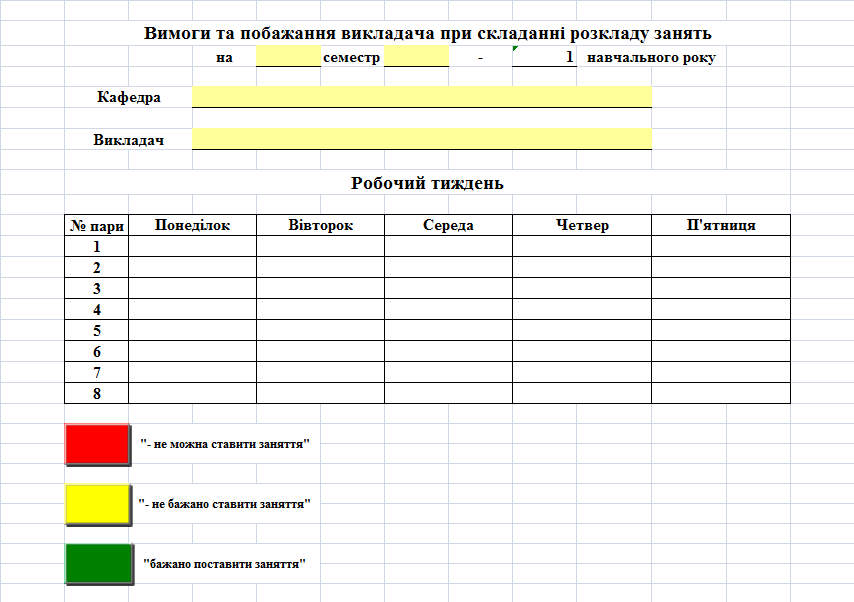


Рисунок 2.1 – Документ «Вимоги викладачів»

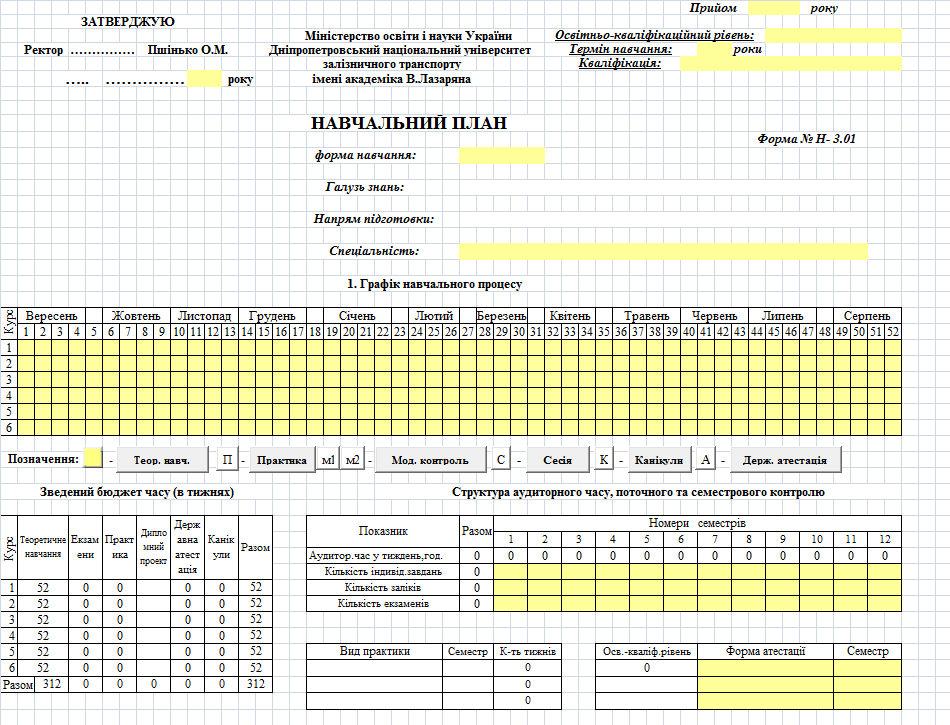


Рисунок 2.2 – Документ «Навчальний план» титульний лист

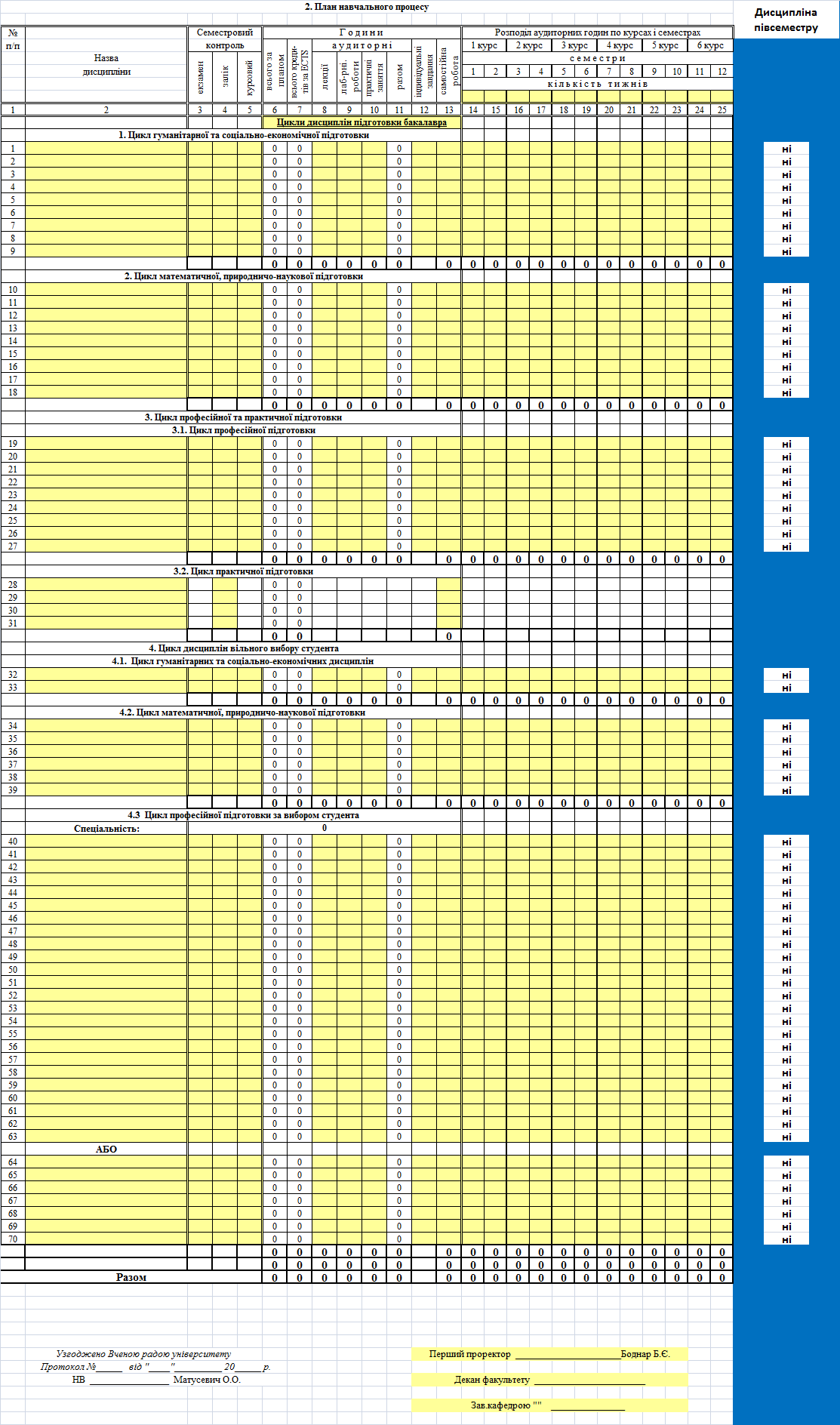


Рисунок 2.3 – Документ «Навчальний план» лист навчального процесу

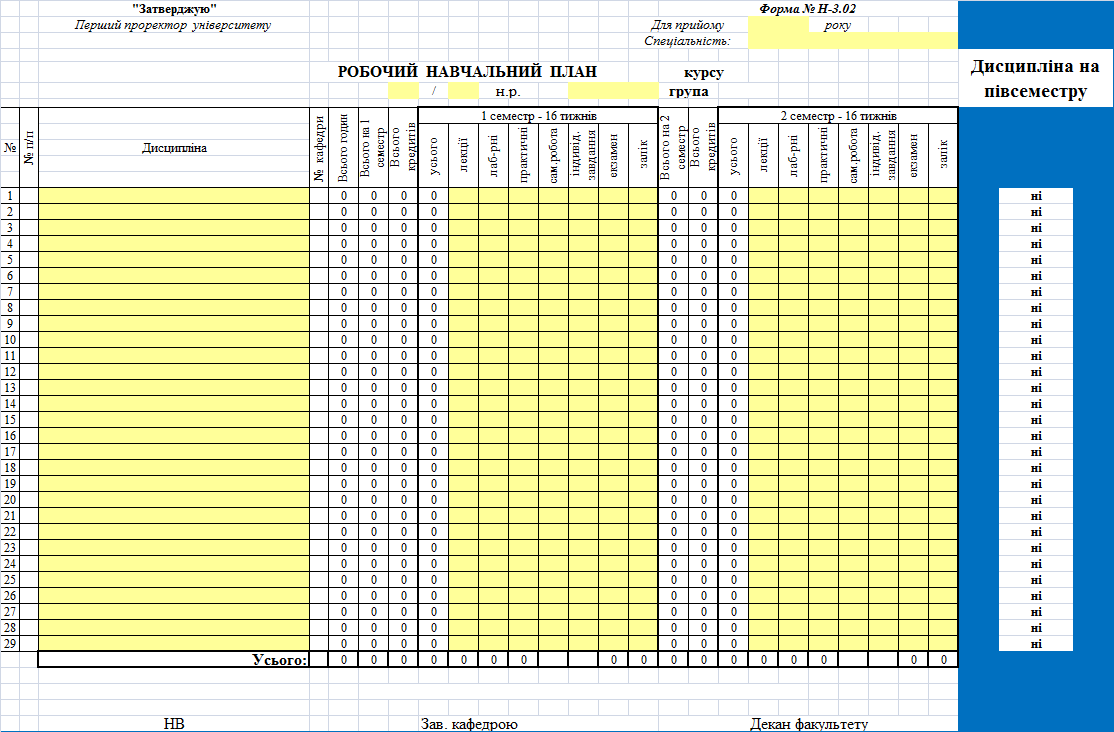


Рисунок 2.4 – Документ «Робочий навчальний план»

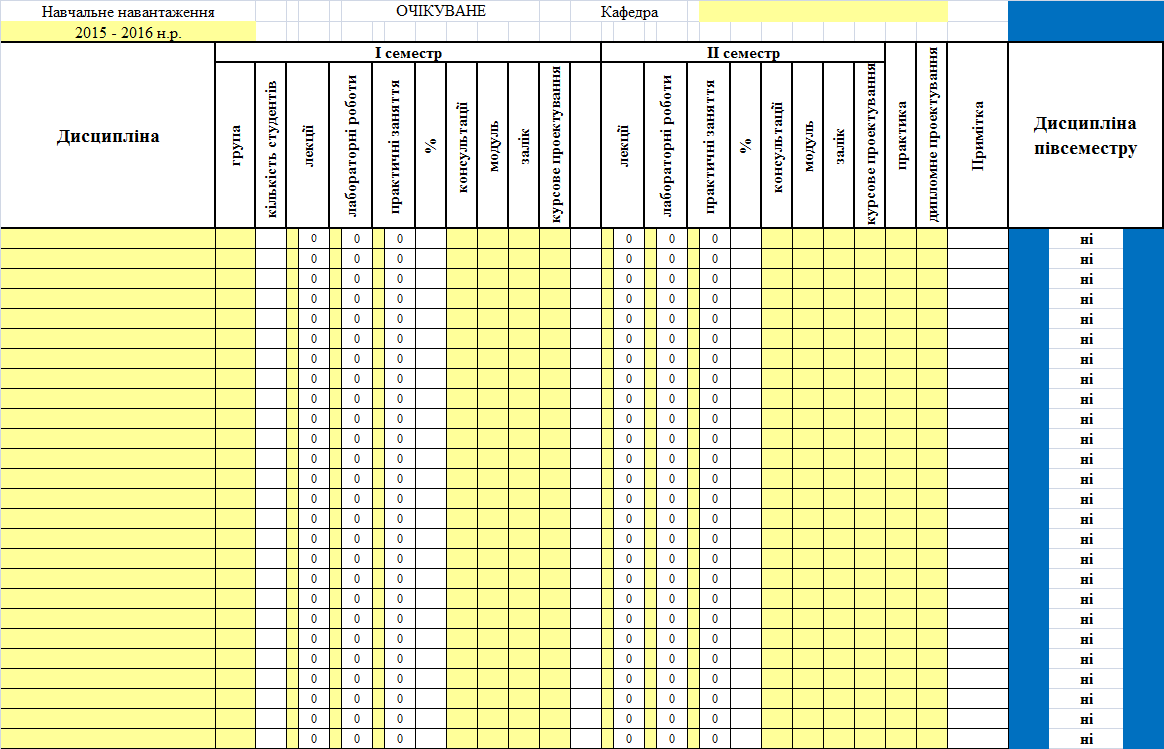


Рисунок 2.5 – Документ «Очікуване навантаження кафедри»

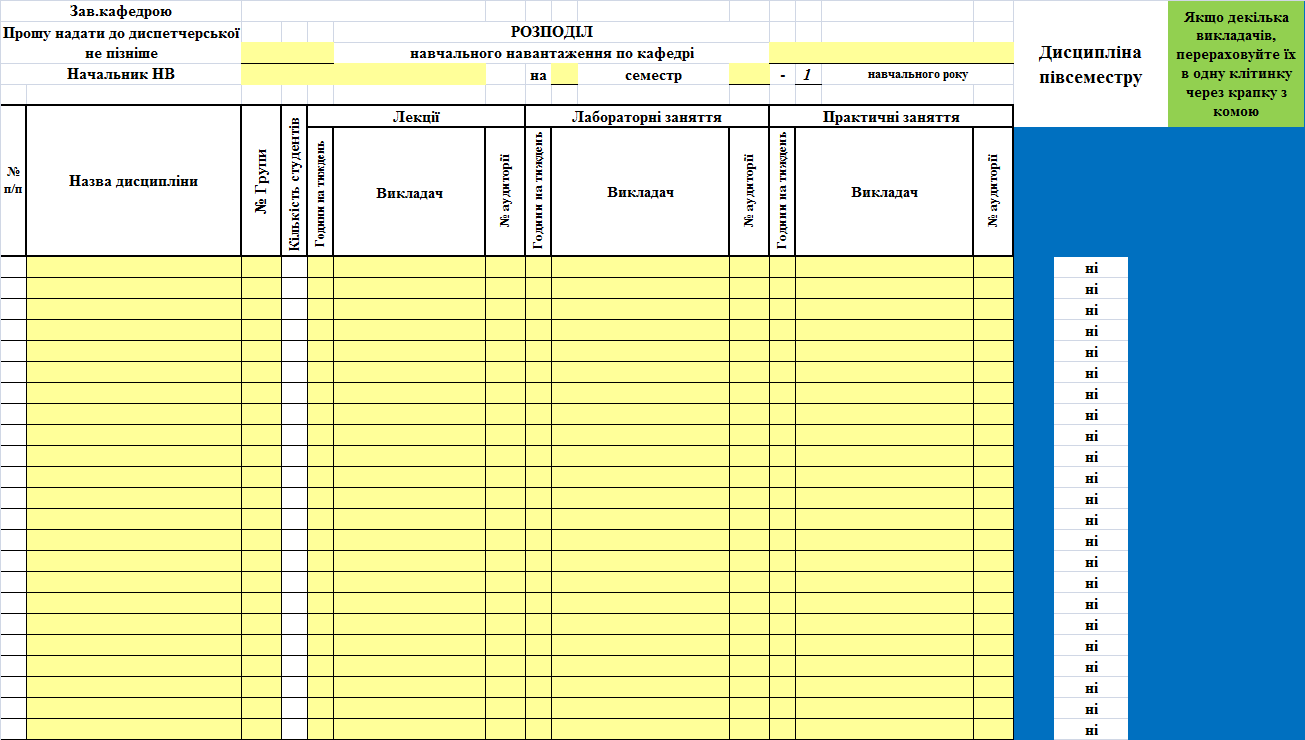


Рисунок 2.6 – Документ «Розподіл навчального навантаження»

2.1.3 Вихідні дані

Результатом роботи програми є наступні вихідні дані:

* «Робочий навчальний план» в форматі \*.xls (рис. 2.4);
* «Очікуване навантаження кафедри» в форматі \*.xls (рис. 2.5);
* «Розподіл навчального навантаження кафедри» в форматі \*.xls (рис. 2.6);
* шаблон документу «Навчальний план» в форматі \*.xls;
* шаблон документу «Робочий навчальний план» в форматі \*.xls;
* шаблон документу «Очікуване навантаження кафедри» в форматі \*.xls;
* шаблон документу «Розподіл навчального навантаження кафедри» в форматі \*.xls;
* шаблон документу «Вимоги викладачів» в форматі \*.xls;
* сформована на основі вхідної інформації база даних;
* потоки груп (табл. 2.19 – 2.20).

Таблиця 2.19 – Потоки

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Об’єм | Рік прийому | Семестр | Дисципліна | Архів | Дата занесення в архів |
|  |  |  |  |  |  |

Примітки:

Об’єм – кількість студентів в потоці, ціле число.

Рік прийому – рік формування потоку, ціле число.

Семестр – семестр на який сформовані потоки, ціле число.

Дисципліна – назва дисципліни відповідно до таблиці 2.5.

Архів – чи є дана інформація застарілою, так/ні.

Дата занесення в архів – дата занесення інформації в архів, дата.

Таблиця 2.20 – Прив’язки групи до потоків

|  |  |
| --- | --- |
| Потік | Група |
|  |  |

Примітки:

Потік – потік відповідно до таблиці 2.19.

Група – назва групи відповідно до таблиці 2.16.

2.2 Проектування бази даних

2.2.1 Специфікація вимог до бази даних

Програма використовує спільну базу даних з системою складання розкладу занять і може імпортувати її дані, тому структура має бути ідентична.

2.2.2 Визначення типів сутностей та зв’язків між ними

Типи сутностей бази даних, що використовуються у системі наведені в табл. 2.21.

Таблиця 2.21 – Типи сутностей

|  |  |
| --- | --- |
| Назва таблиці | Коротка характеристика |
| Faculty | Факультети |
| Position | Посади |
| FormEducation | Форми навчання |
| Qualification | Кваліфікаційні рівні |
| Requirements | Типи вимог |
| Subject | Дисципліни |
| Specialization | Напрями підготовки |
| FieldStudy | Галузі знань |
| TypeAuditory | Типи аудиторій |
| Building | Корпуси |
| Location | Розташування аудиторій |
| Department | Кафедри |
| UniversityTeacher | Викладачі |
| Speciality | Спеціальності |

Продовження таблиці 2.21

|  |  |
| --- | --- |
| Назва таблиці | Коротка характеристика |
| Practic | Типи практик |
| LoadPractic | Навантаження практик |
| GroupStudent | Групи |
| Auditory | Аудиторії |
| SubjectDepartmentSpeciality | Прив’язка дисципліни до кафедри та спеціальності |
| WeekDay | Дні тижня |
| RequirementsUniversityTeacher | Вимоги викладачів |
| CurriculumDoc | Документи «Навчальний план» |
| WorkingCurriculumDoc | Документи «Робочий навчальний план» |
| LoadDepartmentDoc | Документи «Навантаження кафедри» |
| EstimateLoadDepartmentDoc | Документи «Очікуване навантаження кафедри» |
| LoadCurriculum | Навантаження документу «Навчальний план» |
| LoadWorkingCurriculum | Навантаження документу «Робочий навчальний план» |
| LoadDepartment | Навантаження документу «Навантаження кафедри» |
| EstimateLoadDepartment | Навантаження документу «Очікуване навантаження кафедри» |
| TypeWeek | Типи тижнів |
| TypeClass | Типи занять |
| TimeClass | Час проведення занять |
| Stream | Потоки |
| StreamGroup | Групи з прив’язкою до потоків |
| Class | Заняття |
| Schedule | Розклади |
| ScheduleClasses | Заняття з прив’язкою до розкладів |

На основі вимог до даних були виділені наступні зв’язки між сутностями, які наведені у табл. 2.22.

Таблиця 2.22 – Основні зв’язки

|  |  |  |
| --- | --- | --- |
| Тип сутності | Тип зв’язку | Тип сутності |
| Faculty (факультет) | складається з багатьох | Department (кафедра) |
| Department (кафедра) | працює багато | UniversityTeacher (викладач) |
| Position (посада) | працює багато | UniversityTeacher (викладач) |
| Specialization (напрям підготовки) | має багато | Speciality (спеціальність) |
| FieldStudy (галузь знань) | має багато | Speciality (спеціальність) |
| Department (кафедра) | має багато | Speciality (спеціальність) |

Продовження таблиці 2.22

|  |  |  |
| --- | --- | --- |
| Тип сутності | Тип зв’язку | Тип сутності |
| Practic (вид практики) | має багато | LoadPractic (навантаження практики) |
| Speciality (спеціальність) | має багато | LoadPractic (навантаження практики) |
| Speciality (спеціальність) | вчиться багато | GroupStudent (група) |
| FormEducation (форма навчання) | вчиться багато | GroupStudent (група) |
| Qualification (кваліфікаційний рівень) | вчиться багато | GroupStudent (група) |
| Department (кафедра) | має багато | Auditory (аудиторія) |
| TypeAuditory (тип аудиторії) | має багато | Auditory (аудиторія) |
| Location (розташування аудиторії) | має багато | Auditory (аудиторія) |
| Subject (дисципліна) | має багато | SubjectDepartmentSpeciality (прив’язка дисципліни до кафедри та спеціальності) |
| Department (кафедра) | має багато | SubjectDepartmentSpeciality (прив’язка дисципліни до кафедри та спеціальності) |
| Speciality (спеціальність) | має багато | SubjectDepartmentSpeciality (прив’язка дисципліни до кафедри та спеціальності) |
| UniversityTeacher (викладач) | має багато | RequirementsUniversityTeacher (вимоги викладачів) |
| WeekDay (день тижня) | має багато | RequirementsUniversityTeacher (вимоги викладачів) |
| Requirements (тип вимоги) | має багато | RequirementsUniversityTeacher (вимоги викладачів) |
| Qualification (кваліфікаційний рівень) | має багато | CurriculumDoc (документ «Навчальний план») |
| FormEducation (форма навчання) | має багато | CurriculumDoc (документ «Навчальний план») |
| Speciality (спеціальність) | має багато | CurriculumDoc (документ «Навчальний план») |
| GroupStudent (група) | має багато | WorkingCurriculumDoc (документ «Робочий навчальний план») |
| Speciality (спеціальність) | має багато | WorkingCurriculumDoc (документ «Робочий навчальний план») |

Продовження таблиці 2.22

|  |  |  |
| --- | --- | --- |
| Тип сутності | Тип зв’язку | Тип сутності |
| Department (кафедра) | має багато | LoadDepartmentDoc (документ «Навантаження кафедри») |
| Department (кафедра) | має багато | EstimateLoadDepartmentDoc (документ «Очікуване навантаження кафедри») |
| CurriculumDoc (документ «Навчальний план») | має багато | LoadCurriculum (навантаження документу «Навчальний план») |
| Subject (дисципліна) | має багато | LoadCurriculum (навантаження документу «Навчальний план») |
| WorkingCurriculumDoc (документ «Робочий навчальний план») | має багато | LoadWorkingCurriculum (навантаження документу «Робочий навчальний план») |
| SubjectDepartmentSpeciality (прив’язка дисципліни до кафедри та спеціальності) | має багато | LoadWorkingCurriculum (навантаження документу «Робочий навчальний план») |
| LoadDepartmentDoc (документ «Навантаження кафедри») | має багато | LoadDepartment (навантаження документу «Навантаження кафедри») |
| Subject (дисципліна) | має багато | LoadDepartment (навантаження документу «Навантаження кафедри») |
| GroupStudent (група) | має багато | LoadDepartment (навантаження документу «Навантаження кафедри») |
| UniversityTeacher (викладач) | має багато | LoadDepartment (навантаження документу «Навантаження кафедри») |
| Auditory (аудиторія) | має багато | LoadDepartment (навантаження документу «Навантаження кафедри») |
| EstimateLoadDepartmentDoc (документ «Очікуване навантаження кафедри») | має багато | EstimateLoadDepartment (навантаження документу «Очікуване навантаження кафедри») |
| Subject (дисципліна) | має багато | EstimateLoadDepartment (навантаження документу «Очікуване навантаження кафедри») |
| GroupStudent (група) | має багато | EstimateLoadDepartment (навантаження документу «Очікуване навантаження кафедри») |
| TypeWeek (тип тижня) | має багато | TimeClass (час заняття) |
| WeekDay (день тижня) | має багато | TimeClass (час заняття) |
| Subject (дисципліна) | читається багатьом | Stream (Потік) |
| Stream (Потік) | має багато | StreamGroup (група з прив’язкою до Потіку) |
| GroupStudent (група) | має багато | StreamGroup (група з прив’язкою до Потіку) |

Продовження таблиці 2.22

|  |  |  |
| --- | --- | --- |
| Тип сутності | Тип зв’язку | Тип сутності |
| TypeClass (тип заняття) | має багато | Class (заняття) |
| TimeClass (час заняття) | має багато | Class (заняття) |
| Stream (Потік) | має багато | Class (заняття) |
| UniversityTeacher (викладач) | має багато | Class (заняття) |
| Auditory (аудиторія) | має багато | Class (заняття) |
| Schedule (розклад) | має багато | ScheduleClasses (заняття з прив’язкою до розкладу) |
| Class (заняття) | має багато | ScheduleClasses (заняття з прив’язкою до розкладу) |

Кожному факультету може належати багато кафедр, а одна кафедра може належати лише одному факультету, кардинальність зв’язку 1:N (рис. 2.7).

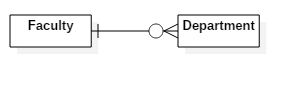


Рисунок 2.7 – Зв’язок факультет має багато кафедр

На кожній кафедрі може працювати багато викладачів, викладач одночасно може працювати лише на одній кафедрі, кардинальність зв’язку 1:N (рис. 2.8).

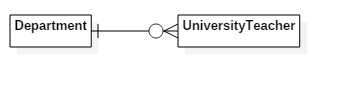


Рисунок 2.8 – Зв’язок на кафедрі працює багато викладачів

На кожній посаді може працювати багато викладачів, викладач одночасно може працювати лише на одній посаді, кардинальність зв’язку 1:N (рис. 2.9).

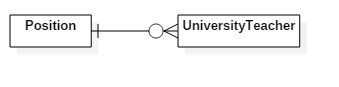


Рисунок 2.9 – Зв’язок на посаді працює багато викладачів

На кожному напрямі підготовки може бути багато спеціальностей, спеціальність може відноситися лише до одного напряму підготовки, кардинальність зв’язку 1:N (рис. 2.10).

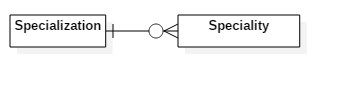


Рисунок 2.10 – Зв’язок напрям підготовки має багато спеціальностей

У кожній галузі знань може бути багато спеціальностей, спеціальність може відноситися лише до однієї галузі знань, кардинальність зв’язку 1:N (рис. 2.11).

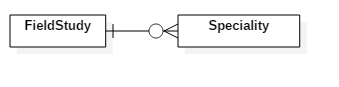


Рисунок 2.11 – Зв’язок галузь знань має багато спеціальностей

На кожній кафедрі може бути багато спеціальностей, спеціальність може відноситися лише до однієї кафедри, кардинальність зв’язку 1:N (рис. 2.12).

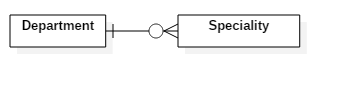


Рисунок 2.12 – Зв’язок кафедра має багато спеціальностей

У кожного виду практики може бути багато навантажень, навантаження може відноситися лише до одного виду практики, кардинальність зв’язку 1:N (рис. 2.13).

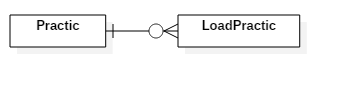


Рисунок 2.13 – Зв’язок вид практики має багато навантажень

У кожної спеціальності може бути багато навантажень, навантаження може відноситися лише до однієї спеціальності, кардинальність зв’язку 1:N (рис. 2.14).

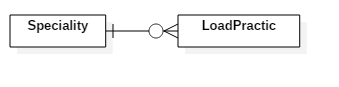


Рисунок 2.14 – Зв’язок спеціальність має багато навантажень практик

На кожній спеціальності може навчатися багато груп, група може навчатися лише на одній спеціальності, кардинальність зв’язку 1:N (рис. 2.15).

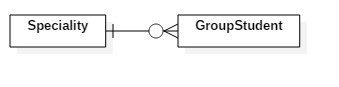


Рисунок 2.15 – Зв’язок на спеціальності навчається багато груп

На кожній формі навчання може навчатися багато груп, група може навчатися лише на одній формі навчання, кардинальність зв’язку 1:N (рис. 2.16).

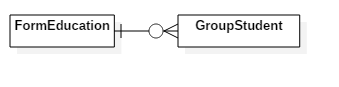


Рисунок 2.16 – Зв’язок на формі навчання навчається багато груп

На кожному кваліфікаційному рівні може навчатися багато груп, група може навчатися лише на одному кваліфікаційному рівні, кардинальність зв’язку 1:N (рис. 2.17).

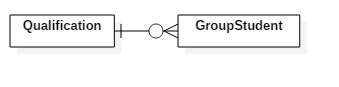


Рисунок 2.17 – Зв’язок на кваліфікаційному рівні навчається багато груп

На кожній кафедрі може бути багато аудиторій, аудиторія може належати лише одній кафедрі, кардинальність зв’язку 1:N (рис. 2.18).

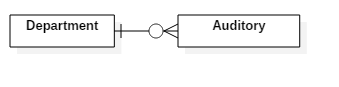


Рисунок 2.18 – Зв’язок на кафедрі багато аудиторій

Одному типу аудиторій може належати багато аудиторій, аудиторія може бути лише одного типу, кардинальність зв’язку 1:N (рис. 2.19).

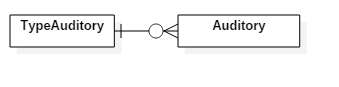


Рисунок 2.19 – Зв’язок типу аудиторії належить багато аудиторій

Кожному розташуванню відповідає багато аудиторій, аудиторія може мати лише одне розташування, кардинальність зв’язку 1:N (рис. 2.20).

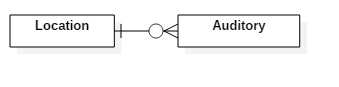


Рисунок 2.20 – Зв’язок розташуванню належить багато аудиторій

Кожна дисципліна може мати багато прив’язок до кафедри та спеціальності, кожна прив’язка відповідає одній дисципліні, кардинальність зв’язку 1:N (рис. 2.21).

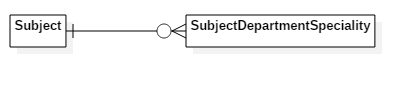


Рисунок 2.21 – Зв’язок дисципліна має багато прив’язок до кафедри та спеціальності

Кожна кафедра може мати багато прив’язок дисципліни до кафедри та спеціальності, кожна прив’язка відповідає одній кафедрі, кардинальність зв’язку 1:N (рис. 2.22).

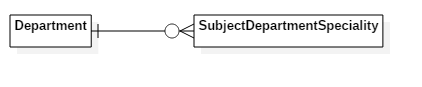


Рисунок 2.22 – Зв’язок кафедра має багато прив’язок дисципліни до кафедри та спеціальності

Кожна спеціальність може мати багато прив’язок дисципліни до кафедри та спеціальності, кожна прив’язка відповідає одній спеціальності, кардинальність зв’язку 1:N (рис. 2.23).

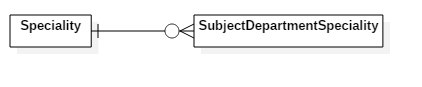


Рисунок 2.23 – Зв’язок спеціальність має багато прив’язок дисципліни до кафедри та спеціальності

Кожен викладач може мати багато вимог, кожна вимога відповідає одному викладачу, кардинальність зв’язку 1:N (рис. 2.24).

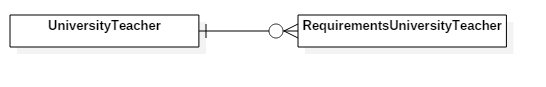


Рисунок 2.24 – Зв’язок викладач має багато вимог

Кожен день тижня може мати багато вимог викладачів, кожна вимога відповідає одному дню тижня, кардинальність зв’язку 1:N (рис. 2.25).

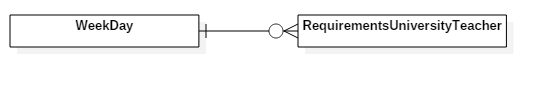


Рисунок 2.25 – Зв’язок день тижня має багато вимог викладачів

Кожен тип вимоги може мати багато вимог, кожна вимога відповідає одному типу, кардинальність зв’язку 1:N (рис. 2.26).

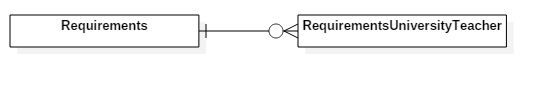


Рисунок 2.26 – Зв’язок тип вимоги має багато вимог викладачів

Кожному кваліфікаційному рівню відповідає багато навчальних планів, кожен навчальний план відповідає одному кваліфікаційному рівню, кардинальність зв’язку 1:N (рис. 2.27).

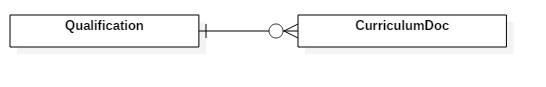


Рисунок 2.27 – Кваліфікаційний рівень має багато документів «Навчальний план»

Кожній формі навчання відповідає багато навчальних планів, кожен навчальний план відповідає одній формі навчання, кардинальність зв’язку 1:N (рис. 2.28).

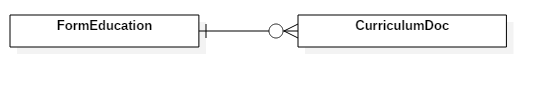


Рисунок 2.28 – Форма навчання має багато документів «Навчальний план»

Кожній спеціальності відповідає багато навчальних планів, кожен навчальний план відповідає одній спеціальності, кардинальність зв’язку 1:N (рис. 2.29).

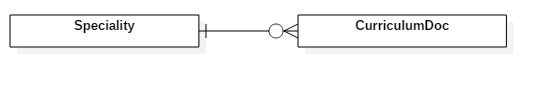


Рисунок 2.29 – Зв’язок спеціальність має багато документів «Навчальний план»

Кожній групі відповідає багато робочих навчальних планів, кожен робочий навчальний план відповідає одній групі, кардинальність зв’язку 1:N (рис. 2.30).

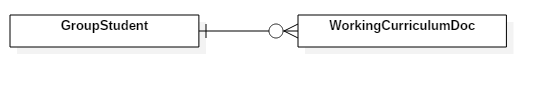


Рисунок 2.30 – Зв’язок група має багато документів «Робочий навчальний план»

Кожній спеціальності відповідає багато робочих навчальних планів, кожен робочий навчальний план відповідає одній спеціальності, кардинальність зв’язку 1:N (рис. 2.31).

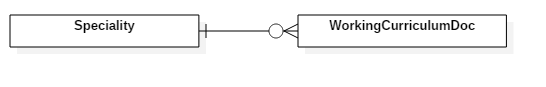


Рисунок 2.31 – Зв’язок спеціальність має багато документів «Робочий навчальний план»

Кожній кафедрі відповідає багато очікуваних навантажень кафедри, кожен документ «Очікуване навантаження кафедри» відповідає одній кафедрі, кардинальність зв’язку 1:N (рис. 2.32).

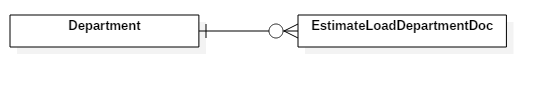


Рисунок 2.32 – Зв’язок кафедра має багато документів «Очікуване навантаження кафедри»

Кожній кафедрі відповідає багато навантажень кафедри, кожен документ «Розподіл навантаження кафедри» відповідає одній кафедрі, кардинальність зв’язку 1:N (рис. 2.33).

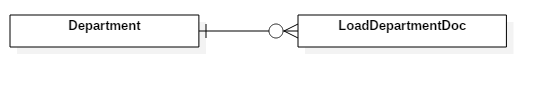


Рисунок 2.33 – Зв’язок кафедра має багато документів «Розподіл навантаження кафедри»

Кожен навчальний план складається з багатьох навантажень, кожне навантаження відповідає одному навчальному плану, кардинальність зв’язку 1:N (рис. 2.34).

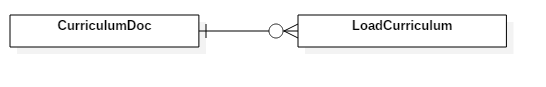


Рисунок 2.34 – Зв’язок документ «Навчальний план» має багато навантажень

Кожен робочий навчальний план складається з багатьох навантажень, кожне навантаження відповідає одному робочому навчальному плану, кардинальність зв’язку 1:N (рис. 2.35).

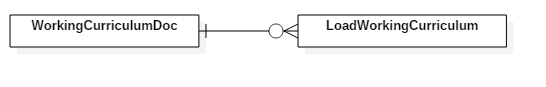


Рисунок 2.35 – Зв’язок документ «Робочий навчальний план» має багато навантажень

Кожен документ «Очікуване навантаження кафедри» складається з багатьох навантажень, кожне навантаження відповідає одному документу «Очікуване навантаження кафедри», кардинальність зв’язку 1:N (рис. 2.36).

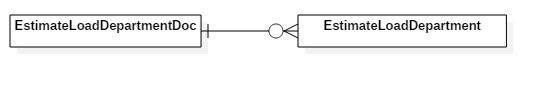


Рисунок 2.36 – Зв’язок документ «Очікуване навантаження кафедри» має багато навантажень

Кожен документ «Розподіл навантаження кафедри» складається з багатьох навантажень, кожне навантаження відповідає одному документу «Розподіл навантаження кафедри», кардинальність зв’язку 1:N (рис. 2.37).

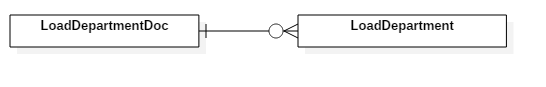


Рисунок 2.37 – Зв’язок документ «Розподіл навантаження кафедри» має багато навантажень

Кожна дисципліна має багато навантажень в документі «Навчальний план», кожне навантаження відповідає одній дисципліні, кардинальність зв’язку 1:N (рис. 2.38).

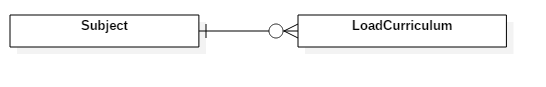


Рисунок 2.38 – Зв’язок дисципліна має багато навантажень в документі «Навчальний план»

Кожна прив’язка дисципліни до кафедри та спеціальності має багато навантажень в документі «Робочий навчальний план», кожне навантаження відповідає одній прив’язці дисципліни до кафедри та спеціальності, кардинальність зв’язку 1:N (рис. 2.39).

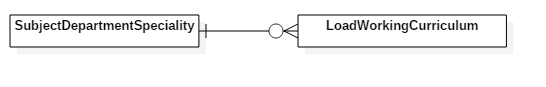


Рисунок 2.39 – Зв’язок прив’язка дисципліни до кафедри та спеціальності має багато навантажень в документі «Робочий навчальний план»

Кожна дисципліна має багато навантажень в документі «Очікуване навантаження кафедри», кожне навантаження відповідає одній дисципліні, кардинальність зв’язку 1:N (рис. 2.40).

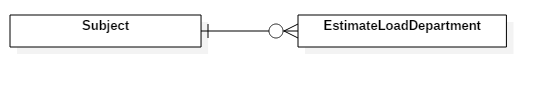


Рисунок 2.40 – Зв’язок дисципліна має багато навантажень в документі «Очікуване навантаження кафедри»

Кожна дисципліна має багато навантажень в документі «Розподіл навантаження кафедри», кожне навантаження відповідає одній дисципліні, кардинальність зв’язку 1:N (рис. 2.41).

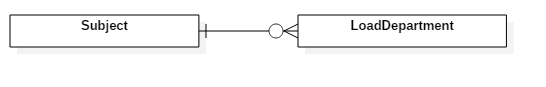


Рисунок 2.41 – Зв’язок дисципліна має багато навантажень в документі «Розподіл навантаження кафедри»

Кожна група має багато навантажень в документі «Розподіл навантаження кафедри», кожне навантаження відповідає одній групі, кардинальність зв’язку 1:N (рис. 2.42).

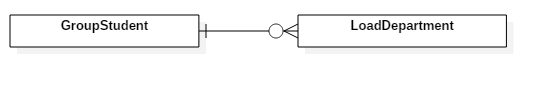


Рисунок 2.42 – Зв’язок група має багато навантажень в документі «Розподіл навантаження кафедри»

Кожна аудиторія має багато навантажень в документі «Розподіл навантаження кафедри», кожне навантаження відповідає одній аудиторії, кардинальність зв’язку 1:N (рис. 2.43).

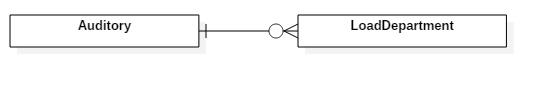


Рисунок 2.43 – Зв’язок аудиторія має багато навантажень в документі «Розподіл навантаження кафедри»

Кожен викладач має багато навантажень в документі «Розподіл навантаження кафедри», кожне навантаження відповідає одному викладачеві, кардинальність зв’язку 1:N (рис. 2.44).

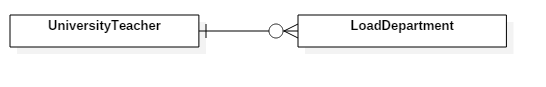


Рисунок 2.44 – Зв’язок викладач має багато навантажень в документі «Розподіл навантаження кафедри»

Кожна група має багато навантажень в документі «Очікуване навантаження кафедри», кожне навантаження відповідає одній групі, кардинальність зв’язку 1:N (рис. 2.45).

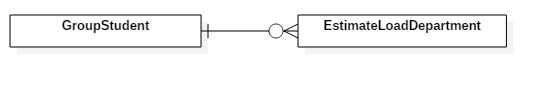


Рисунок 2.45 – Зв’язок група має багато навантажень в документі «Очікуване навантаження кафедри»

Кожен тип тижня має багато часів занять, кожний час занять відповідає одному типу тижня, кардинальність зв’язку 1:N (рис. 2.46).

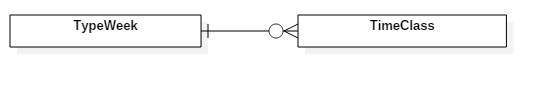


Рисунок 2.46 – Зв’язок тип тижня має багато часів занять

Кожен день тижня має багато часів занять, кожний час занять відповідає одному дню тижня, кардинальність зв’язку 1:N (рис. 2.47).

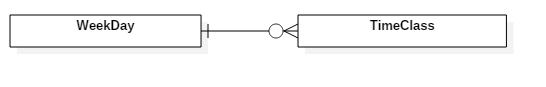


Рисунок 2.47 – Зв’язок день тижня має багато часів занять

Кожна дисципліна може читатися багатьом потокам, кожен потік відповідає одній дисципліні, кардинальність зв’язку 1:N (рис. 2.48).

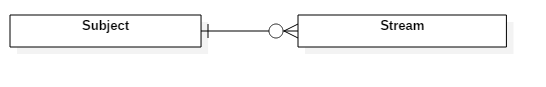


Рисунок 2.48 – Зв’язок дисципліна може читатися багатьом потокам

Кожен потік може мати багато груп з прив’язкою до потоків, кожна прив’язка групи до потоку відповідає одному потоку, кардинальність зв’язку 1:N (рис. 2.49).

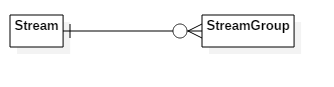


Рисунок 2.49 – Зв’язок потік може мати багато груп з прив’язкою до потоків

Кожна група може мати багато прив’язок групи до потоку, кожна прив’язка групи до потоку відповідає одній групі, кардинальність зв’язку 1:N (рис. 2.50).

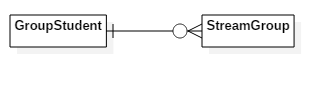


Рисунок 2.50 – Зв’язок група має багато прив’язок групи до потоку

Кожний тип занять має багато занять, кожне заняття відповідає одному типу, кардинальність зв’язку 1:N (рис. 2.51).

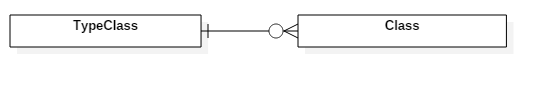


Рисунок 2.51 – Зв’язок тип занять має багато занять

Кожний час занять має багато занять, кожне заняття відповідає одному часу занять, кардинальність зв’язку 1:N (рис. 2.52).

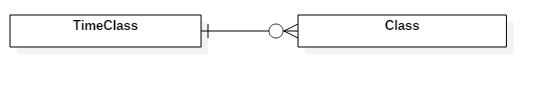


Рисунок 2.52 – Зв’язок час занять має багато занять

Кожен потік має багато занять, кожне заняття відповідає одному потоку, кардинальність зв’язку 1:N (рис. 2.53).

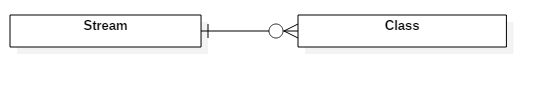


Рисунок 2.53 – Зв’язок потік має багато занять

Кожен викладач має багато занять, кожне заняття відповідає одному викладачу, кардинальність зв’язку 1:N (рис. 2.54).

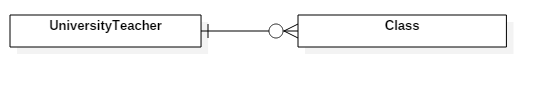


Рисунок 2.54 – Зв’язок викладач має багато занять

Кожна аудиторія має багато занять, кожне заняття відповідає одній аудиторії, кардинальність зв’язку 1:N (рис. 2.55).

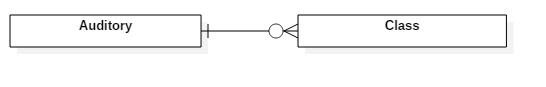


Рисунок 2.55 – Зв’язок аудиторія має багато занять

Кожен розклад має багато занять з прив’язкою до розкладу, кожна прив’язка заняття відповідає одному розкладу, кардинальність зв’язку 1:N (рис. 2.56).

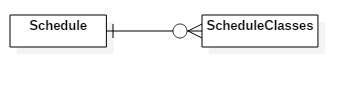


Рисунок 2.56 – Зв’язок розклад має багато занять з прив’язкою до розкладу

Кожне заняття має багато прив’язок занять до розкладу, кожна прив’язка розкладу відповідає одному заняттю, кардинальність зв’язку 1:N (рис. 2.57).

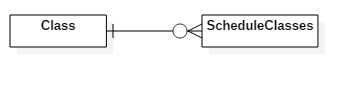


Рисунок 2.57 – Зв’язок заняття має багато прив’язок занять до розкладу

Сутності, що були виділені наведені в таблицях 2.23 – 2.59.

Таблиця 2.23 – Таблиця Faculty

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Опис | Тип даних | Обмеження |
| id | унікальний ідентифікатор факультету | цілий | первинний ключ |
| name | назва факультету | рядок |  |
| abbreviation | коротка назва факультету | рядок |  |
| number | номер факультету | цілий |  |
| isArchive | застаріла інформація | логічний | значення за замовчуванням false |
| archiveDate | дата коли інформація стала застарілою | дата |  |

Таблиця 2.24 – Таблиця Position

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Опис | Тип даних | Обмеження |
| id | унікальний ідентифікатор посади | цілий | первинний ключ |
| name | назва посади | рядок |  |
| abbreviation | коротка назва посади | рядок |  |
| isArchive | застаріла інформація | логічний | значення за замовчуванням false |
| archiveDate | дата коли інформація стала застарілою | дата |  |

Таблиця 2.25 – Таблиця FormEducation

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Опис | Тип даних | Обмеження |
| id | унікальний ідентифікатор форми навчання | цілий | первинний ключ |
| name | назва форми навчання | рядок |  |
| isArchive | застаріла інформація | логічний | значення за замовчуванням false |
| archiveDate | дата коли інформація стала застарілою | дата |  |

Таблиця 2.26 – Таблиця Qualification

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Опис | Тип даних | Обмеження |
| id | унікальний ідентифікатор кваліфікаційного рівня | цілий | первинний ключ |
| name | назва кваліфікаційного рівня | рядок |  |
| isArchive | застаріла інформація | логічний | значення за замовчуванням false |
| archiveDate | дата коли інформація стала застарілою | дата |  |

Таблиця 2.27 – Таблиця Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Опис | Тип даних | Обмеження |
| id | унікальний ідентифікатор типу вимог | цілий | первинний ключ |
| name | назва типу вимог | рядок |  |

Таблиця 2.28 – Таблиця Subject

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Опис | Тип даних | Обмеження |
| id | унікальний ідентифікатор дисципліни | цілий | первинний ключ |
| name | назва дисципліни | рядок |  |
| abbreviation | коротка назва дисципліни | рядок |  |
| isArchive | застаріла інформація | логічний | значення за замовчуванням false |
| archiveDate | дата коли інформація стала застарілою | дата |  |

Таблиця 2.29 – Таблиця Specialization

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Опис | Тип даних | Обмеження |
| id | унікальний ідентифікатор напряму підготовки | цілий | первинний ключ |
| name | назва напряму підготовки | рядок |  |
| code | код напряму підготовки | рядок |  |
| isArchive | застаріла інформація | логічний | значення за замовчуванням false |
| archiveDate | дата коли інформація стала застарілою | дата |  |

Таблиця 2.30 – Таблиця FieldStudy

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Опис | Тип даних | Обмеження |
| id | унікальний ідентифікатор галузі знань | цілий | первинний ключ |
| name | назва галузі знань | рядок |  |
| code | код галузі знань | рядок |  |
| isArchive | застаріла інформація | логічний | значення за замовчуванням false |
| archiveDate | дата коли інформація стала застарілою | дата |  |

Таблиця 2.31 – Таблиця TypeAuditory

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Опис | Тип даних | Обмеження |
| id | унікальний ідентифікатор типу аудиторії | цілий | первинний ключ |
| name | назва типу аудиторії | рядок |  |
| isArchive | застаріла інформація | логічний | значення за замовчуванням false |
| archiveDate | дата коли інформація стала застарілою | дата |  |

Таблиця 2.32 – Таблиця Building

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Опис | Тип даних | Обмеження |
| id | унікальний ідентифікатор корпусу | цілий | первинний ключ |
| name | назва корпусу | рядок |  |
| isArchive | застаріла інформація | логічний | значення за замовчуванням false |
| archiveDate | дата коли інформація стала застарілою | дата |  |

Таблиця 2.33 – Таблиця Location

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Опис | Тип даних | Обмеження |
| id | унікальний ідентифікатор розташування | цілий | первинний ключ |
| floor | поверх | цілий |  |
| block | блок | цілий |  |
| idBuilding | корпус | цілий | зовнішній ключ |

Таблиця 2.34 – Таблиця Department

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Опис | Тип даних | Обмеження |
| id | унікальний ідентифікатор кафедри | цілий | первинний ключ |
| name | назва кафедри | рядок |  |
| abbreviation | коротка назва кафедри | рядок |  |
| number | номер кафедри | цілий |  |
| idFaculty | факультет | цілий | зовнішній ключ |
| idChairHolder | завідуючий | цілий | зовнішній ключ |
| isArchive | застаріла інформація | логічний | значення за замовчуванням false |
| archiveDate | дата коли інформація стала застарілою | дата |  |

Таблиця 2.35 – Таблиця UniversityTeacher

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Опис | Тип даних | Обмеження |
| id | унікальний ідентифікатор викладача | цілий | первинний ключ |
| secondName | прізвище викладача | рядок |  |
| firstName | ім’я викладача | рядок |  |
| middleName | прізвище викладача | рядок |  |
| idDepartment | кафедра | цілий | зовнішній ключ |
| idPosition | посада | цілий | зовнішній ключ |
| isArchive | застаріла інформація | логічний | значення за замовчуванням false |
| archiveDate | дата коли інформація стала застарілою | дата |  |

Таблиця 2.36 – Таблиця Speciality

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Опис | Тип даних | Обмеження |
| id | унікальний ідентифікатор спеціальності | цілий | первинний ключ |
| name | назва спеціальності | рядок |  |
| abbreviation | коротка назва спеціальності | рядок |  |
| code | код спеціальності | рядок |  |

Продовження таблиці 2.36

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Опис | Тип даних | Обмеження |
| idSpecialization | напрям підготовки | цілий | зовнішній ключ |
| idFieldStudy | галузь знань | цілий | зовнішній ключ |
| idDepartment | кафедра | цілий | зовнішній ключ |
| isArchive | застаріла інформація | логічний | значення за замовчуванням false |
| archiveDate | дата коли інформація стала застарілою | дата |  |

Таблиця 2.37 – Таблиця Practic

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Опис | Тип даних | Обмеження |
| id | унікальний ідентифікатор типу практики | цілий | первинний ключ |
| name | назва типу практики | рядок |  |
| isArchive | застаріла інформація | логічний | значення за замовчуванням false |
| archiveDate | дата коли інформація стала застарілою | дата |  |

Таблиця 2.38 – Таблиця LoadPractic

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Опис | Тип даних | Обмеження |
| id | унікальний ідентифікатор навантаження практики | цілий | первинний ключ |
| idPractic | тип практики | цілий | зовнішній ключ |
| idSpeciality | спеціальність | цілий | зовнішній ключ |
| semester | семестр | цілий |  |
| yearReception | рік прийому | цілий |  |
| countHour | кількість годин | цілий |  |

Таблиця 2.39 – Таблиця GroupStudent

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Опис | Тип даних | Обмеження |
| id | унікальний ідентифікатор групи | цілий | первинний ключ |
| name | назва групи | цілий |  |
| countStudent | кількість студентів | цілий |  |
| idSpeciality | спеціальність | цілий | зовнішній ключ |
| idFormEducation | форма навчання | цілий | зовнішній ключ |

Продовження таблиці 2.39

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Опис | Тип даних | Обмеження |
| idQualification | кваліфікаційний рівень | цілий | зовнішній ключ |
| course | курс | цілий |  |
| isCollege | коледж | логічний | значення за замовчуванням false |
| isArchive | застаріла інформація | логічний | значення за замовчуванням false |
| archiveDate | дата коли інформація стала застарілою | дата |  |

Таблиця 2.40 – Таблиця Auditory

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Опис | Тип даних | Обмеження |
| id | унікальний ідентифікатор аудиторії | цілий | первинний ключ |
| number | номер аудиторії | рядок |  |
| capacity | вмістимість | цілий |  |
| idDepartment | кафедра | цілий | зовнішній ключ |
| idType | тип | цілий | зовнішній ключ |
| idLocation | розташування | цілий | зовнішній ключ |
| isAvailability | готовність | логічний | значення за замовчуванням true |
| startUnAvailability | початок неготовності | дата |  |
| endUnAvailability | кінець неготовності | дата |  |
| isArchive | застаріла інформація | логічний | значення за замовчуванням false |
| archiveDate | дата коли інформація стала застарілою | дата |  |

Таблиця 2.41 – Таблиця SubjectDepartmentSpeciality

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Опис | Тип даних | Обмеження |
| id | унікальний ідентифікатор прив’язки дисципліни до кафедри та спеціальності | цілий | первинний ключ |
| idSubject | дисципліна | цілий | зовнішній ключ |
| idDepartment | кафедра | цілий | зовнішній ключ |
| idSpeciality | спеціальність | цілий | зовнішній ключ |
| isArchive | застаріла інформація | логічний | значення за замовчуванням false |
| archiveDate | дата коли інформація стала застарілою | дата |  |

Таблиця 2.42 – Таблиця WeekDay

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Опис | Тип даних | Обмеження |
| id | унікальний ідентифікатор дня тижня | цілий | первинний ключ |
| name | назва дня тижня | рядок |  |

Таблиця 2.43 – Таблиця RequirementsUniversityTeacher

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Опис | Тип даних | Обмеження |
| id | унікальний ідентифікатор вимог викладачів | цілий | первинний ключ |
| idUniversityTeacher | викладач | цілий | зовнішній ключ |
| yearReception | рік прийому | цілий |  |
| semester | семестр | цілий |  |
| idWeekday | день тижня | цілий | зовнішній ключ |
| numberClass | номер заняття | цілий |  |
| idRequirements | тип вимоги | цілий | зовнішній ключ |
| isArchive | застаріла інформація | логічний | значення за замовчуванням false |
| archiveDate | дата коли інформація стала застарілою | дата |  |

Таблиця 2.44 – Таблиця CurriculumDoc

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Опис | Тип даних | Обмеження |
| id | унікальний ідентифікатор навчального плану | цілий | первинний ключ |
| yearReception | рік прийому | цілий |  |
| idQualification | кваліфікаційний рівень | цілий | зовнішній ключ |
| academicTerm | термін навчання | цілий |  |
| idFormEducation | форма навчання | цілий | зовнішній ключ |
| idSpeciality | спеціальність | цілий | зовнішній ключ |

Таблиця 2.45 – Таблиця WorkingCurriculumDoc

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Опис | Тип даних | Обмеження |
| id | унікальний ідентифікатор робочого навчального плану | цілий | первинний ключ |
| idGroupStudent | група | цілий | зовнішній ключ |
| yearReception | рік прийому | цілий |  |
| idSpeciality | спеціальність | цілий | зовнішній ключ |

Таблиця 2.46 – Таблиця LoadDepartmentDoc

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Опис | Тип даних | Обмеження |
| id | унікальний ідентифікатор документу «Розподіл навантаження кафедри» | цілий | первинний ключ |
| idDepartment | кафедра | цілий | зовнішній ключ |
| yearReception | рік прийому | цілий |  |
| semester | семестр | цілий |  |
| isArchive | застаріла інформація | логічний | значення за замовчуванням false |
| archiveDate | дата коли інформація стала застарілою | дата |  |

Таблиця 2.47 – Таблиця EstimateLoadDepartmentDoc

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Опис | Тип даних | Обмеження |
| id | унікальний ідентифікатор документу «Очікуване навантаження кафедри» | цілий | первинний ключ |
| idDepartment | кафедра | цілий | зовнішній ключ |
| yearReception | рік прийому | цілий |  |

Таблиця 2.48 – Таблиця LoadCurriculum

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Опис | Тип даних | Обмеження |
| id | унікальний ідентифікатор навантаження документу «Навчальний план» | цілий | первинний ключ |
| idCurriculumDoc | документ «Навчальний план» | цілий | зовнішній ключ |
| idSubject | дисципліна | цілий | зовнішній ключ |
| isHalfSemester | дисципліна півсеместру | логічний | значення за замовчуванням false |
| numberSubject | номер дисципліни | цілий |  |
| countHourLecture | кількість годин на лекції | цілий |  |
| countHourLabWork | кількість годин на лабораторні | цілий |  |
| countHourPracticWork | кількість годин на практичні | цілий |  |

Продовження таблиці 2.48

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Опис | Тип даних | Обмеження |
| individualTask | індивідуальне завдання | рядок |  |
| isExam | екзамен | логічний | значення за замовчуванням false |
| isTest | залік | логічний | значення за замовчуванням true |
| countHourIndividualWork | кількість годин на самостійну роботу | цілий |  |
| semesterI | навантаження в першому семестрі | цілий |  |
| semesterII | навантаження в другому семестрі | цілий |  |
| semesterIII | навантаження в третьому семестрі | цілий |  |
| semesterIV | навантаження в четвертому семестрі | цілий |  |
| semesterV | навантаження в п’ятому семестрі | цілий |  |
| semesterVI | навантаження в шостому семестрі | цілий |  |
| semesterVII | навантаження в сьомому семестрі | цілий |  |
| semesterVIII | навантаження в восьмому семестрі | цілий |  |
| semesterIX | навантаження в дев’ятому семестрі | цілий |  |
| semesterX | навантаження в десятому семестрі | цілий |  |
| semesterXI | навантаження в одинадцятому семестрі | цілий |  |
| semesterXII | навантаження в дванадцятому семестрі | цілий |  |

Таблиця 2.49 – Таблиця LoadWorkingCurriculum

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Опис | Тип даних | Обмеження |
| id | унікальний ідентифікатор навантаження документу «Робочий навчальний план» | цілий | первинний ключ |

Продовження таблиці 2.49

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Опис | Тип даних | Обмеження |
| isHalfSemester | дисципліна півсеместру | логічний | значення за замовчуванням false |
| idWorkingCurriculumDoc | документ «Робочий навчальний план» | цілий | зовнішній ключ |
| idSubjectDepartmentSpeciality | дисципліна з прив’язкою до кафедри та спеціальності | цілий | зовнішній ключ |
| numberSubject | номер дисципліни | цілий |  |
| countHourLectureSemesterI | кількість годин на лекції в першому семестрі | цілий |  |
| countHourLabWorkSemesterI | кількість годин на лабораторні роботи в першому семестрі | цілий |  |
| countHourPracticWorkSemesterI | кількість годин на практичні роботи в першому семестрі | цілий |  |
| individualTaskSemesterI | індивідуальне завдання в першому семестрі | рядок |  |
| countHourIndividualWorkSemesterI | кількість годин на самостійну роботу в першому семестрі | цілий |  |
| isExamSemesterI | екзамен в першому семестрі | логічний | значення за замовчуванням false |
| isTestSemesterI | залік в першому семестрі | логічний | значення за замовчуванням true |
| countHourLectureSemesterII | кількість годин на лекції в другому семестрі | цілий |  |
| countHourLabWorkSemesterII | кількість годин на лабораторні роботи в другому семестрі | цілий |  |
| countHourPracticWorkSemesterII | кількість годин на практичні роботи в другому семестрі | цілий |  |
| individualTaskSemesterII | індивідуальне завдання в другому семестрі | рядок |  |

Продовження таблиці 2.49

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Опис | Тип даних | Обмеження |
| countHourIndividualWorkSemesterII | кількість годин на самостійну роботу в другому семестрі | цілий |  |
| isExamSemesterII | екзамен в другому семестрі | логічний | значення за замовчуванням false |
| isTestSemesterII | залік в другому семестрі | логічний | значення за замовчуванням true |

Таблиця 2.50 – Таблиця LoadDepartment

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Опис | Тип даних | Обмеження |
| id | унікальний ідентифікатор навантаження документу «Розподіл навантаження кафедри» | цілий | первинний ключ |
| idLoadDepartmentDoc | документ «Розподіл навантаження кафедри» | цілий | зовнішній ключ |
| idSubject | дисципліна | цілий | зовнішній ключ |
| isHalfSemester | дисципліна півсеместру | логічний | значення за замовчуванням false |
| numberSubject | номер дисципліни | цілий |  |
| idGroupStudent | група | цілий | зовнішній ключ |
| countHourLecture | кількість годин на лекції | цілий |  |
| countHourLabWork | кількість годин на лабораторні | цілий |  |
| countHourPracticWork | кількість годин на практичні | цілий |  |
| idUniversityTeacherLecture | викладач лекції | цілий | зовнішній ключ |
| idUniversityTeacherLabWork | викладач лабораторних | цілий | зовнішній ключ |
| idUniversityTeacherPracticWork | викладач практичних | цілий | зовнішній ключ |

Продовження таблиці 2.50

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Опис | Тип даних | Обмеження |
| idUniversityTeacherLabWorkSecond | другий викладач лабораторних | цілий | зовнішній ключ |
| idUniversityTeacherPracticWorkSecond | другий викладач практичних | цілий | зовнішній ключ |
| idAuditoryLecture | аудиторія для лекцій | цілий | зовнішній ключ |
| idAuditoryLabWork | аудиторія для лабораторних | цілий | зовнішній ключ |
| idAuditoryPracticWork | аудиторія для практичних | цілий | зовнішній ключ |

Таблиця 2.51 – Таблиця EstimateLoadDepartment

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Опис | Тип даних | Обмеження |
| id | унікальний ідентифікатор навантаження документу «Очікуване навантаження кафедри» | цілий | первинний ключ |
| idEstimateLoadDepartmentDoc | документ «Очікуване навантаження кафедри» | цілий | зовнішній ключ |
| idSubject | дисципліна | цілий | зовнішній ключ |
| isHalfSemester | дисципліна півсеместру | логічний | значення за замовчуванням false |
| isCourseWorkSemesterI | курсовий в першому семестрі | логічний | значення за замовчуванням false |
| idGroupStudent | група | цілий | зовнішній ключ |
| countHourLectureSemesterI | кількість годин на лекції в першому семестрі | цілий |  |
| countHourLabWorkSemesterI | кількість годин на лабораторні в першому семестрі | цілий |  |
| countHourPracticWorkSemesterI | кількість годин на практичні в першому семестрі | цілий |  |
| consultationSemesterI | кількість годин на консультації в першому семестрі | цілий |  |

Продовження таблиці 2.51

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Опис | Тип даних | Обмеження |
| examSemesterI | кількість годин на модуль в першому семестрі | цілий |  |
| testSemesterI | кількість годин на залік в першому семестрі | цілий |  |
| countHourLectureSemesterII | кількість годин на лекції в другому семестрі | цілий |  |
| countHourLabWorkSemesterII | кількість годин на лабораторні в другому семестрі | цілий |  |
| countHourPracticWorkSemesterII | кількість годин на практичні в другому семестрі | цілий |  |
| consultationSemesterII | кількість годин на консультації в другому семестрі | цілий |  |
| examSemesterII | кількість годин на модуль в другому семестрі | цілий |  |
| testSemesterII | кількість годин на залік в другому семестрі | цілий |  |
| isCourseWorkSemesterII | курсовий в другому семестрі | логічний | значення за замовчуванням false |

Таблиця 2.52 – Таблиця TypeWeek

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Опис | Тип даних | Обмеження |
| id | унікальний ідентифікатор типу тижня | цілий | первинний ключ |
| name | назва типу тижня | рядок |  |

Таблиця 2.53 – Таблиця TypeClass

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Опис | Тип даних | Обмеження |
| id | унікальний ідентифікатор типу заняття | цілий | первинний ключ |
| name | назва типу заняття | рядок |  |

Таблиця 2.54 – Таблиця TimeClass

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Опис | Тип даних | Обмеження |
| id | унікальний ідентифікатор часу заняття | цілий | первинний ключ |
| idTypeWeek | тип тижня | цілий | зовнішній ключ |
| idWeekDay | день тижня | цілий | зовнішній ключ |
| numberClass | номер заняття | цілий |  |

Таблиця 2.55 – Таблиця Stream

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Опис | Тип  даних | Обмеження |
| id | унікальний ідентифікатор потоку | цілий | первинний ключ |
| volume | кількість студентів | цілий |  |
| yearReception | рік прийому | цілий |  |
| semester | семестр | цілий |  |
| idSubject | дисципліна | цілий | зовнішній ключ |
| isArchive | застаріла інформація | логічний | значення за замовчуванням false |
| archiveDate | дата коли інформація стала застарілою | дата |  |

Таблиця 2.56 – Таблиця StreamGroup

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Опис | Тип даних | Обмеження |
| id | унікальний ідентифікатор потоку груп | цілий | первинний ключ |
| idStream | потік | цілий | зовнішній ключ |
| idGroupStudent | група | цілий | зовнішній ключ |

Таблиця 2.57 – Таблиця Class

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Опис | Тип даних | Обмеження |
| id | унікальний ідентифікатор заняття | цілий | первинний ключ |
| idTypeClass | тип заняття | цілий | зовнішній ключ |
| idTimeClass | час заняття | цілий | зовнішній ключ |
| idStream | потік | цілий | зовнішній ключ |
| idUniversityTeacher | викладач | цілий | зовнішній ключ |
| idAuditory | аудиторія | цілий | зовнішній ключ |

Таблиця 2.58 – Таблиця Schedule

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Опис | Тип даних | Обмеження |
| id | унікальний ідентифікатор розкладу | цілий | первинний ключ |
| semester | семестр | цілий |  |
| yearReception | рік прийому | цілий |  |

Таблиця 2.59 – Таблиця ScheduleClasses

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Опис | Тип даних | Обмеження |
| id | унікальний ідентифікатор заняття з прив’язкою до розкладу | цілий | первинний ключ |
| idSchedule | розклад | цілий | зовнішній ключ |
| idClass | заняття | цілий | зовнішній ключ |

2.2.3 Логічне проектування бази даних

Етап логічного проектування бази даних передбачає наступні дії [9]:

1. перетворення концептуальної моделі даних в логічну:
2. видалення зв’язків типу M:N;
3. видалення складних зв’язків;
4. видалення рекурсивних зв’язків;
5. видалення зв’язків, що мають атрибути;
6. видалення множинних атрибутів;
7. перевірка зв’язків типу 1:1;
8. видалення надлишкових зв’язків.
9. визначення набору відношень;
10. перевірка моделі за допомогою правил нормалізації;
11. створення і перевірка логічної моделі даних.

Видалення зв’язків типу M:N **–** проводиться видалення зв’язків, коли багатьом записам таблиці А відповідає багато записів таблиці Б, які є в концептуальній моделі.

В концептуальній моделі даних такі зв’язки відсутні.

Видалення складних зв’язків **–** проводиться видалення не бінарних зв’язків, які є в концептуальній моделі.

В концептуальній моделі даних складні зв’язки відсутні.

Видалення рекурсивних зв’язків **–** проводиться видалення зв’язків між окремими екземплярами сутності одного і того ж типу.

В концептуальній моделі даних рекурсивні зв’язки відсутні.

Видалення множинних атрибутів **–** множинним атрибутом називається атрибут, який має декілька значень для однієї сутності.

В концептуальній моделі даних множинні атрибути відсутні.

В концептуальній моделі даних зв’язки типу 1:1 відсутні.

Нормалізація – це розбивка таблиці на відношення, що не будуть мати аномалії відновлення, включення і видалення даних [9].

Проведемо перевірку розробленої логічної моделі даних на відповідність її до форми НФБК.

НФ1 – відношення знаходиться в НФ1, якщо в ньому на перетині кожного рядка і кожного стовпця міститься тільки одне значення.

У розроблених відношеннях відсутні такі комірки в таблицях, щоб мали більше одного значення на перетині рядка і стовпчику.

НФ2 – відношення знаходиться в НФ2, якщо воно знаходиться в НФ1 і кожен атрибут якого, що не входить до складу первинного ключа, характеризується повною функціональною залежністю від цього первинного ключа.

У розроблених сутностях первинні ключі складаються із одного атрибуту, тому відношення відповідають НФ2.

НФ3 – відношення знаходиться в НФ3, якщо воно знаходиться в НФ1 і НФ2, і не має не вхідних у первинний ключ атрибутів, які б знаходилися в транзитивній функціональній залежності від первинного ключа.

Кожна розроблена сутність має одну функціональну залежність, тому транзитивної залежності між атрибутами сутності не може бути.

Нормальна форма Бойса – Кодда (НФБК) враховує функціональні залежності, у яких беруть участь усі потенційні ключі відносини, а не тільки його первинний ключ.

Для відношення з єдиним потенційним ключем його НФ3 і НФБК є еквівалентними.

Відношення знаходиться в НФБК тоді і тільки тоді, коли кожен його детермінант є потенційним ключем.

Кожна сутність має один потенціальний ключ, який є первинним ключем, тому, в даному випадку, НФ3 і НФБК є еквівалентними.

2.2.4 Схема бази даних

Кожна таблиця належить до групування об’єктів всередині бази даних, котру називають схемою бази даних. Схема бази даних – це іменований контейнер (простір імен) котрий можна використовувати для того щоб згрупувати таблиці і інші об’єкти бази даних.

Для таблиць схема бази даних дозволяє таблицям з однією назвою належати різним схемам.

Схема бази даних (рис. 2.58 – 2.61) відображає таблиці і зв’язки між ними.

Так як кількість відношень бази даних є достатньо великою, то схема представлена чотирма пакетами відношень, для більш зручного аналізу.

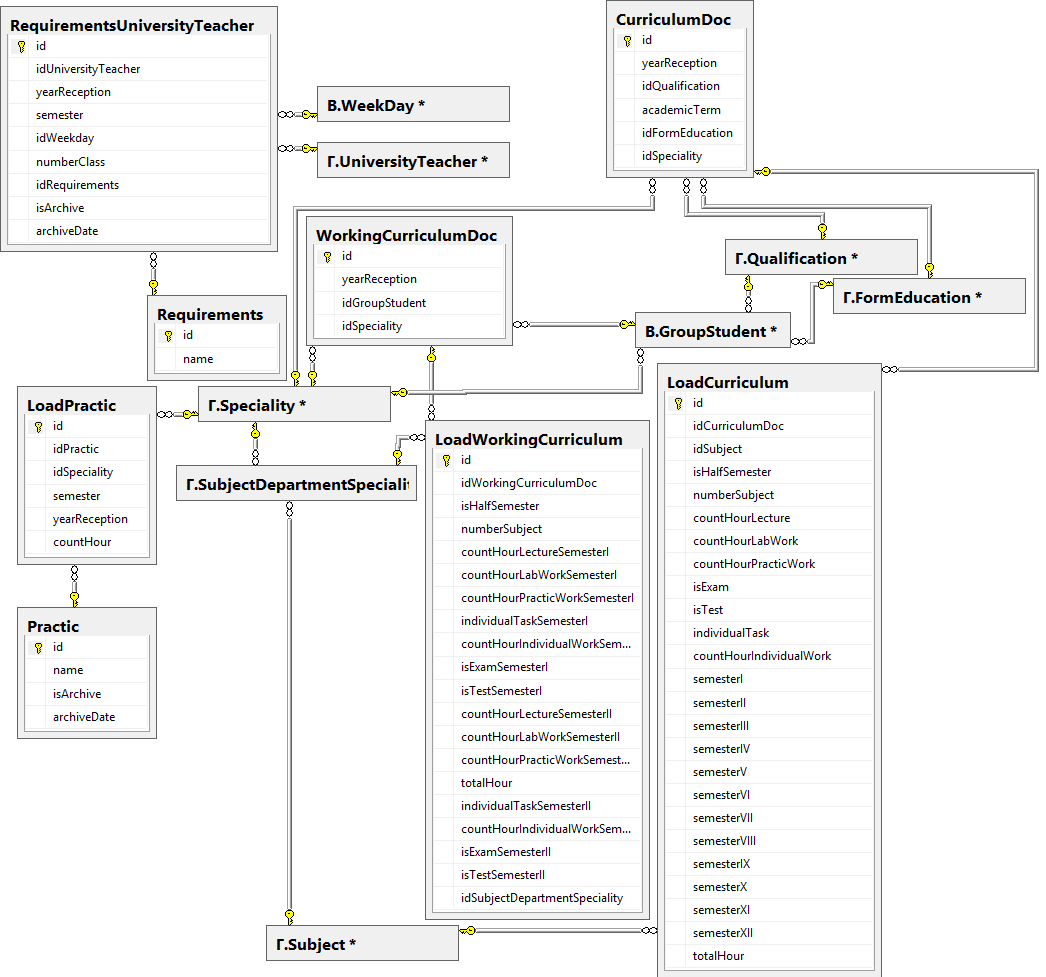


Рисунок 2.58 – Відношення пакету А

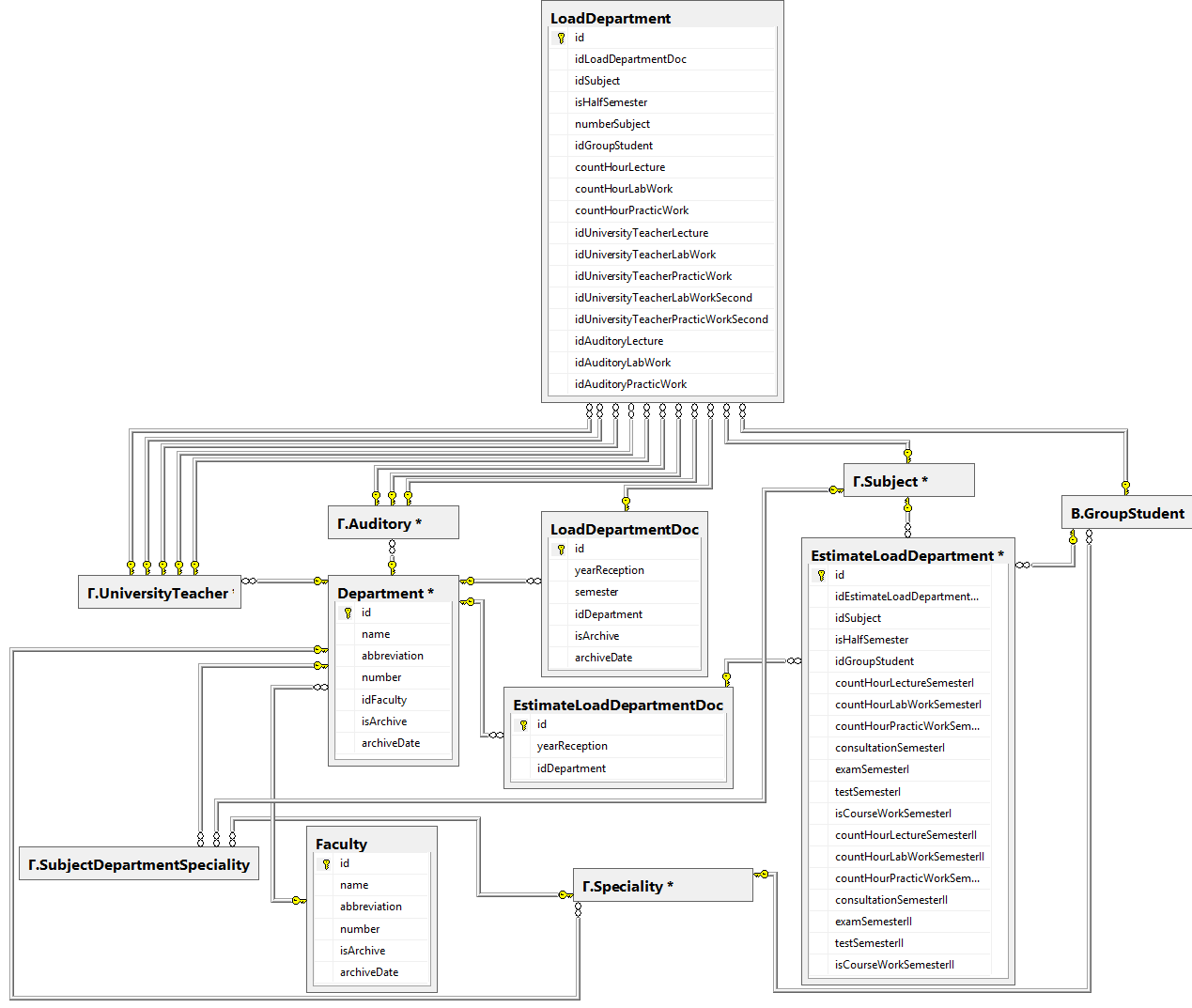


Рисунок 2.59 – Відношення пакету Б

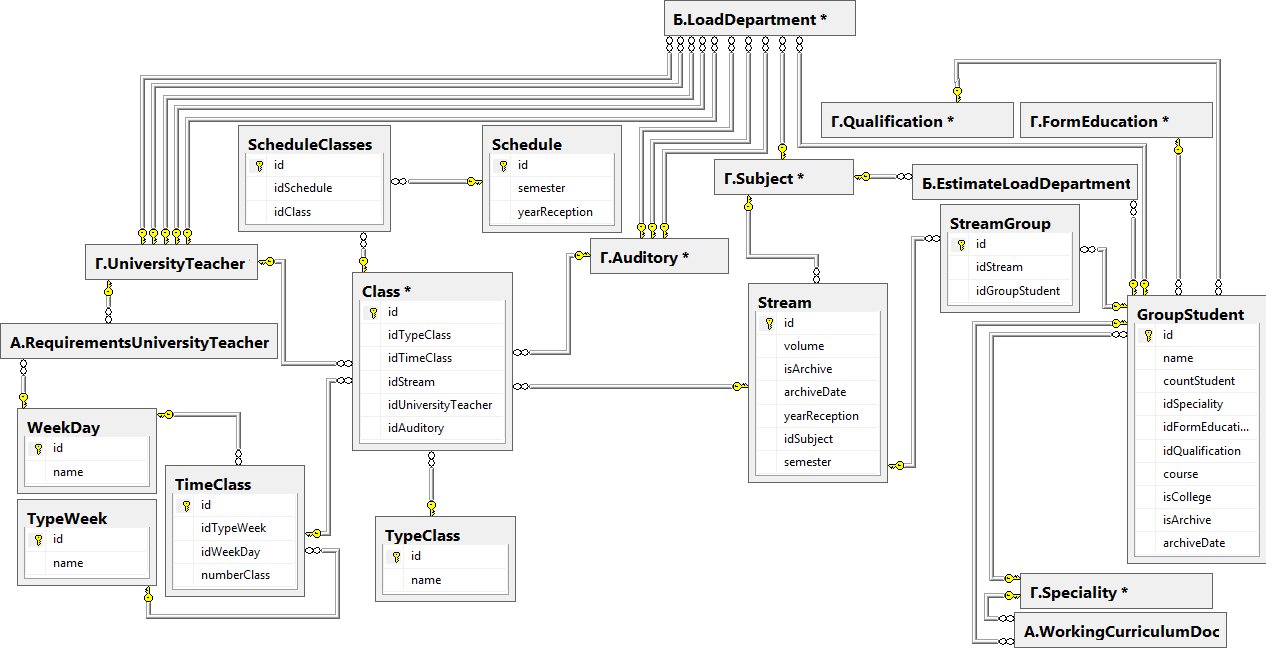


Рисунок 2.60 – Відношення пакету В

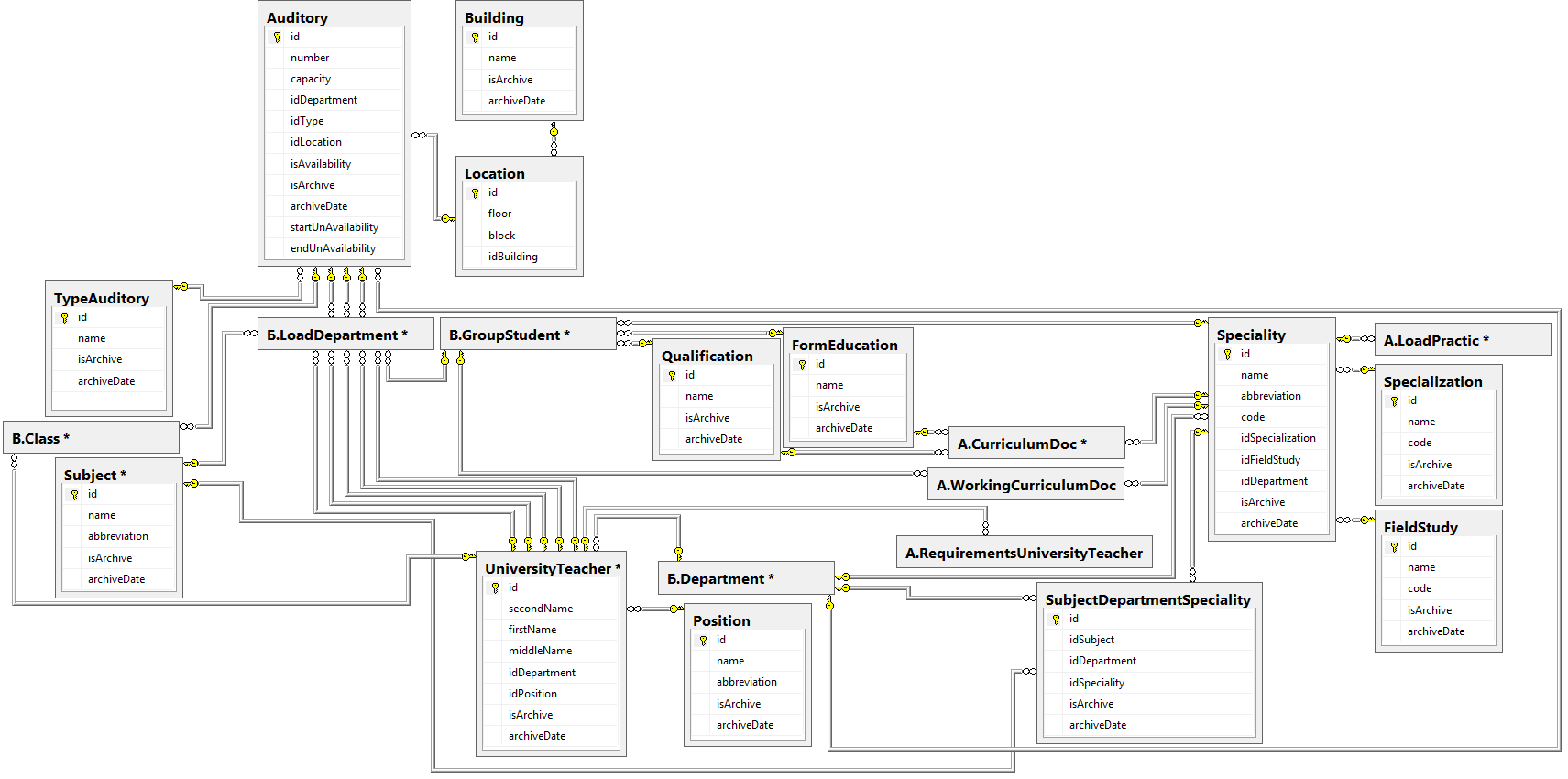


Рисунок 2.61 – Відношення пакету Г

2.3 Вибір мови програмування

Обираючи мову програмування необхідно проаналізувати задачі, що ставляться перед розробником та можливі проблеми, що можуть виникнути під час розробки.

Згідно вимог програма повинна бути настільна. Тому необхідно обрати мову, що дозволяє швидко та ефективно створювати програми даного типу. Окрім того вихідна документація складається у форматі \*.xls тому необхідною буде наявність бібліотек, що дозволять вирішити цю задачу.

Для розробки було обрано мову програмування С# на платформі .NET 4.5, так як 40 % настільних додатків створюється засобами даної мови. Крім того C# має багато бібліотек для роботи з файлами у форматі \*.xls.

В якості СКБД використовується MS SQL Server, оскільки саме ця СКБД найбільш «рідна» обраній мові програмування.

В якості середовища для розробки було обрано Microsoft Visual Studio 2013 Community. Програмний продукт є безкоштовним та має широкі можливості для розробки, рефакторінгу, налагодження та профілювання програмних засобів.

2.3.1 Опис платформи .NET Framework

При проектуванні платформи .NET Framework компанія Microsoft врахувала недоліки існуючих Windows-платформ. .NET Framework складається з двох частин: загальномовного виконуючого середовища (Common Language Runtime, CLR) і бібліотеки класів (Framework Class Library, FCL). CLR надає модель програмування, використовувану у всіх типах програм. У CLR власний завантажувач файлів, диспетчер пам'яті (складальник сміття), система безпеки (безпека доступу до коду), пул потоків і багато що інше. Крім того, CLR надає об'єктно-орієнтовану модель програмування, що визначає, як виглядають і поводяться типи і об'єкти. FCL надає об'єктно-орієнтований API-інтерфейс, використовуваний всіма моделями програм. У ній містяться визначення типів, які дозволяють розробникам виконувати введення/виведення, планування завдань в інших потоках, створювати графічні образи, порівнювати рядки і тому подібне. Природно, що всі ці визначення типів відповідають існуючій моделі програмування в CLR [10].

Повна і абсолютна міжмовна взаємодія. У .NET Framework підтримується міжмовне спадкоємство, міжмовна обробка виключень і міжмовна відладка.

Загальне середовище виконання для будь-яких програм .NET, незалежно від того, на яких мовах вони були створені. Один з важливих моментів при цьому те, що для всіх мов використовується один і той же набір вбудованих типів даних [11].

Єдина програмна модель. На відміну від існуючого підходу, коли одні функції операційної системи доступні через процедури бібліотек (DLL), що динамічно підключаються, а інші – через СОМ-об'єкти, весь прикладний сервіс представлений загальною об'єктно-орієнтованою програмною моделлю [11].

Спрощена модель програмування. Позбавляє від роботи з різними структурами, як це було з Win32 і СОМ. Так, розробникові не потрібно розбиратися з реєстром, глобальними унікальними ідентифікаторами (GUID), IUnknown, AddRef, Release, HRESULT і так далі [11].

Відсутність проблем з версіями. Всі Windows-розробники знають про проблеми сумісності версій, відомих під назвою "DLL hell". Ця проблема виникає, коли компоненти, що встановлюються для нової програми, замінюють компоненти старої програми і у результаті остання починає поводитися дивно або перестає працювати. Архітектура .NET Framework дозволяє ізолювати прикладні компоненти, так що програма завжди завантажує компоненти, з якими вона будувалася і тестувалася. Якщо програма працює після початкової установки, вона працюватиме завжди [11].

Спрощене розгортання. Раніше Windows-програми було дуже важко встановлювати і розгортати: зазвичай потрібно було створити масу файлів, параметрів реєстру і ярликів. До того ж повністю видалити програму практично неможливо. З приходом .NET Framework всі ці проблеми залишаються у минулому. Компоненти .NET Framework не пов'язані з реєстром. Установка програм .NET Framework зводиться лише до копіювання файлів в потрібні каталоги і створення ярликів. Видалення ж програм зводиться до видалення файлів [11].

Робота на багатьох платформах. При компіляції коду для .NET Framework компілятор генерує код на спільній проміжній мові (Common Itermediate Language, CIL), а не традиційний код, що складається з процесорних команд. При виконання CIL транслюється в команди процесора. Оскільки трансляція виконується в період виконання, генеруються команди конкретного процесора. Це означає, що можна розгортати програму .NET Framework на будь-якій машині, де працює версія .NET Framework відповідна стандарту ЕСМА: з архітектурою х86, х64, IA64 [11].

Інтеграція мов програмування. Технологія СОМ підтримує взаємодію різних мов – .NET Framework забезпечує інтеграцію різних мов, тобто одна мова може використовувати типи, створені на інших мовах. Наприклад .NET Framework дозволяє створити на C++ клас, похідний від класу, реалізованого на VisualBasic. У CLR це можливо через наявність загальної системи типів (Common Type System, CTS), яку повинні використовувати всі мови, орієнтовані на CLR. Загальномовна специфікація (Common Language Specification, CLS) визначає правила, яким повинні слідувати розробники компіляторів, щоб їх мови інтегрувалися з іншими. Сама Microsoft пропонує декілька таких мов: C++/CLI (C++ з керованими розширеннями), С#, VisualBasic .NET. Крім того, інші компанії і навчальні заклади створюють компілятори інших мов, сумісних з CLR [11].

Спрощене повторне використання кодів. Всі описані вище механізми дозволяють створювати власні класи, що надають сервіс стороннім програмам. Тепер багатократне використання коду стає виключно простим і створюється великий ринок готових компонентів [11].

Автоматичне управління пам'яттю (збір сміття). Програмування вимагає великої майстерності і дисципліни, особливо коли мова йде про управлінні використанням ресурсів (файлів, пам'яті, простору екрану, мережевих з'єднань, ресурсів баз даних і інших). Одна з найпоширеніших помилок – недбале відношення до звільнення цих ресурсів, що може привести до некоректного виконання програми в непередбачуваний момент. CLR автоматично відстежує використання ресурсів, гарантуючи, що не відбудеться їх витоку [10].

Перевірка безпеки типів CLR може перевіряти безпеку використання типів в коді, що гарантує коректне звернення до існуючих типів. Якщо вхідний параметр методу оголошений як 4-байтове значення, CLR виявить і запобігне передачі 8-байтного значення цього параметра. Безпека типів також означає, що управління може передаватися тільки в певні точки (точки входу методів). Неможливо вказати довільну адресу і змусити програму виконуватися, починаючи з цієї адреси. Сукупність всіх цих захисних заходів позбавляє від багатьох поширених програмних помилок (наприклад, від можливості використання переповнювання буфера для "злому" програми) [10].

Розвинена підтримка відладки. Оскільки CLR використовується для багатьох мов, можна написати окремий фрагмент програми на мові, найбільш відповідній для конкретного завдання, – CLR повністю підтримує відладку багатомовних програм [11].

Єдиний принцип обробки збоїв. Один з найнеприємніших моментів Windows-програмування – неузгоджений стиль повідомлень про збої. Одні функції повертають коди станів Win32, інші – HRESULT, треті генерують виключення. У CLR про всі збої повідомляється через виключення, які дозволяють відокремити код, необхідний для відновлення після збою, від основного алгоритму. Таке розділення полегшує написання, читання і супровід програм. Крім того, виключення працюють в багатомодульних і багатомовних програмах. І на відміну від код станів і HRESULT виключення не можна проігнорувати. CLR також надає вбудовані засоби аналізу стека, що помітно спрощують пошук фрагментів, що викликають збої [11].

Взаємодія з існуючим кодом. У зв'язку з тим, що розробники накопичили величезний об'єм коду і компонентів, переписування всього цього коду, так щоб він задіював всі переваги .NET Framework, значно уповільнило б перехід до цієї платформи. Тому в .NET Framework реалізована повна підтримка доступу до СОМ-компонентів і Win32-функцій в існуючих динамічних бібліотеках DLL [10].

2.3.2 Опис мови розробки програмного продукту – C#

Мова програмування С# була створена в кінці 1990-х років і стала частиною загальної .NET-стратегії Microsoft. Вперше вона з'явилась як бета-версія в середині 2000 року. Головним архітектором С# був Андерс Хейлсберг (Anders Hejlsberg) – один з провідних фахівців в області мов програмування, що отримав визнання у всьому світі. В 1980-х він був автором вельми успішного продукту Turbo Pascal, витончена реалізація якого встановила стандарт для всіх майбутніх компіляторів. С# безпосередньо пов'язаний із С, C++ і Java. Ці три мови – найпопулярніші мови програмування в світі. Більш того, майже всі професійні програмісти сьогодні знають С і C++, і більшість знає Java. Оскільки С# побудований на міцному, зрозумілому фундаменті, то перехід від цих "фундаментальних" мов до "надбудови" відбувається без особливих зусиль з боку програмістів. Оскільки Андерс Хейлсберг не збирався винаходити своє "колесо", він зосередився на введенні удосконалень і нововведень [12].

Першоджерелом С# є мова С. Від С мова С# успадкувала синтаксис, багато ключових слів і оператори. Крім того, С# побудований на об'єктній моделі, визначеній в C++. Якщо програміст знає С або C++, то мову С# програміст освоїть легко.

У C# входить багато корисних особливостей – простота, об'єктна орієнтованість, типова захищеність, "збірка сміття", підтримка сумісності версій і багато що інше. Дані можливості дозволяють швидко і легко розробляти програми, особливо COM+ програми і Web-сервіси. При створенні C#, його автори враховували досягнення багатьох інших мов програмування: C++, C, Java, SmallTalk, Delphi, Visual Basic і так далі. Внаслідок того, що C# розроблявся з чистого листа, у його авторів була можливість (якою вони явно скористалися), залишити у минулому всі незручні і неприємні особливості (що існують, як правило, для зворотної сумісності), будь-якої з попередніх мов. В результаті вийшла дійсно проста, зручна і сучасна мова, яка по потужності не поступається С++, та така, що істотно підвищила продуктивність програм [13].

В C# були зроблені заходи для виключення помилок при оновленні програмного забезпечення. Зміна коду, в такій ситуації, може непередбачувано змінити суть самої програми. Щоб допомогти розробникам боротися з цією проблемою C# включає підтримку сумісності версій. Зокрема, у відмінності від C++ і Java, якщо метод класу був змінений, це повинно бути спеціально обумовлено. Це дозволяє обійти помилки в коді і забезпечити гнучку сумісність версій.

У C# була уніфікована система типів, тепер можна розглядати кожен тип як об'єкт. Не дивлячись на те, використовує програміст клас, структуру, масив або вбудований тип, можна звертатися до нього як до об'єкту. Об'єкти зібрані в простори імен (namespaces), які дозволяють програмно звертатися до чого-небудь. Це означає що замість списку файлів заголовків, що включаються, в програмі необхідно написати які простори імен, для доступу до об'єктів і класів усередині них, потрібно використовувати.

У C# вираз using дозволяє не писати кожного разу назву простору імен, коли використовується клас з нього. Наприклад, простір імен System містить декілька класів, у тому числі і Console. Можна писати або назву простору імен перед кожним зверненням до класу, або використовувати using [12].

Сучасність C# виявляється і в нових кроках до полегшення процесу відладки програми. Традиційним засобом для відладки програм на стадії розробки в C++ є маркування значних частин коди директивами #ifdef і так далі. У C#, використовуючи атрибути, орієнтовані на умовні слова, програміст може набагато швидше писати відладжений код [13].

2.4 Об’єктно-орієнтоване проектування

Для розробки програмного продукту був обраний метод об'єктно-орієнтовного проектування [14].

Підставами до вибору даної парадигми є:

1. швидкість модифікації та розширення програмного коду;
2. наявність якісного середовища проектування;
3. широкі можливості з повторного використання коду;
4. зручність при розробці архітектури системи;
5. наявність загальноприйнятих стратегій щодо вирішення типових проблем.

Для створення об’єктно-орієнтованої моделі системи використовуються діаграми класів UML [15].

Під час об’єктно-орієнтованого проектування для вирішення деяких типових проблем були використані шаблони проектування.

Для роботи з даними використовується технологія ORM (object-relational mapping – відображення даних на реальні об’єкти) Entity Framework.

Entity Framework представляє собою об’єктно-орієнтовану технологію на базі .NET Framework для роботи з даними. Якщо традиційні засоби ADO.NET дозволяють створювати підключення, команди та інші об’єкти для взаємодії з базами даних, то Entity Framework представляє собою більш високий рівень абстракції, котрий дозволяє абстрагуватися від самої бази даних і працювати незалежно від типу сховища. Якщо на фізичному рівні ми оперуємо таблицями, ключами, то на концептуальному рівні, котрий пропонує Entity Framework, ми працюємо з об’єктами.

З виходом версії 4.0 в 2010 році Entity Framework став рекомендованною технологією для доступу до даних, були введені можливості взаємодії з базою даних – підходи Database First, Model First, Code First. В даній розробці використовується підхід Code First.

Центральною концепцією Entity Framework є поняття сутності. Сутність представляє набір даних асоційованих з конкретним об’єктом. Тому дана технологія передбачає роботу не з таблицями, а з об’єктами. Для вибірки з бази даних використовуються запроси LINQ.

Таким чином через класи, об’явлені в програмі можна взаємодіяти з таблицями бази даних.

Для розробки була обрана трирівнева архітектура, яка ділить проект на три рівні:

1. Presentation layer (рівень представлення даних) – рівень з яким безпосередньо взаємодіє користувач. Включає компоненти користувацького інтерфейсу, механізм прийому даних від користувача;
2. Business layer (рівень бізнес логіки) – містить набір компонентів, які відповідають за обробку отриманих від рівня представлення даних, реалізує всю логіку програми, взаємодіє з рівнем доступу до даних і передає йому результат обробки;
3. Data access layer (рівень доступу до даних) – зберігає моделі, які описують використовувані сутності, також тут містяться специфічні класи для роботи з різними технологіями доступу до даних, наприклад клас контексту Entity Framework. Тут також зберігаються репозиторії, через які рівень бізнес логіки взаємодіє з базою даних.

Для роботи з даними був обраний шаблон «Репозиторій» – який дозволяє абстрагуватися від конкретних підключень до даних, з якими працює програма і є проміжною ланкою між класами, які безпосередньо взаємодіють з даними і рештою програми. Таким чином «Репозиторій» додає гнучкості програмі при роботі з різними типами підключень. Через класи репозиторіїв рівень бізнес логіки взаємодіє з базою даних.

Для того щоб спростити роботу з великою кількістю репозиторіїв був обраний шаблон «Одиниця роботи». Даний шаблон дає доступ до репозиторіїв через окремі методи і визначає загальний контекст для всіх репозиторіїв.

Крім того на рівні представлення даних використовується шаблон MVP – який є похідним від MVC. Відмінність заключається в тому що, представлення не зв’язано з модел’ю. Складається з: Model – Presenter – View. Презентер відповідає за переміщення даних, а також за оновлення представлень. «Спілкування» моделей з представленнями відбувається через презентер.

Діаграми UML проекту представлені на рис. 2.62 – 2.76.

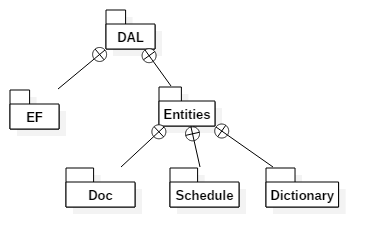


Рисунок 2.62 – Діаграма пакетів рівня доступу до даних

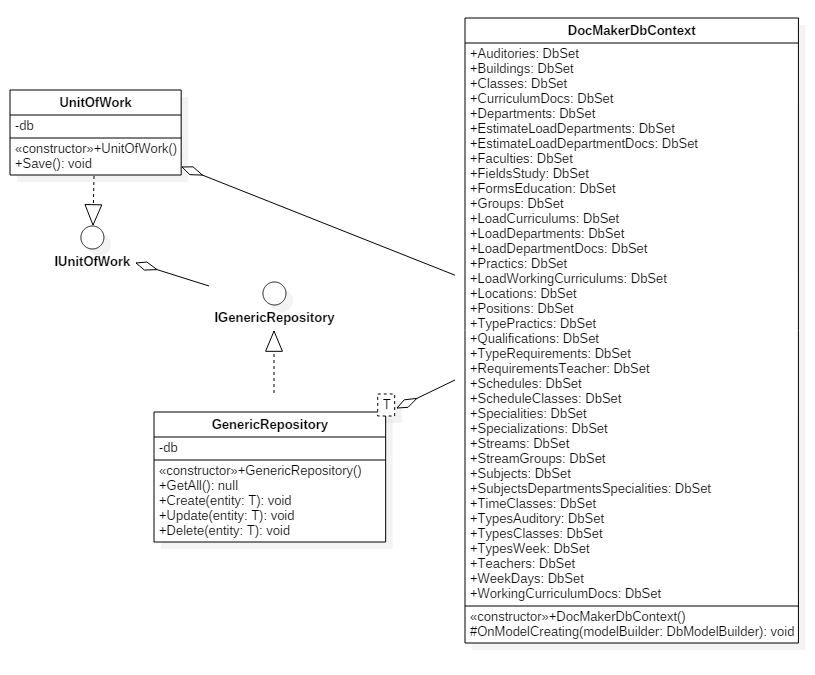


Рисунок 2.63 – Діаграма класів пакету EF

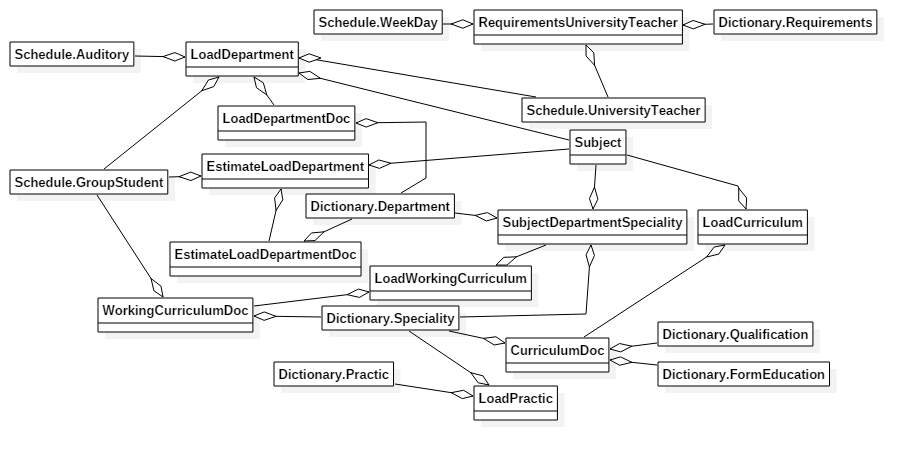


Рисунок 2.64 – Діаграма класів пакету Doc

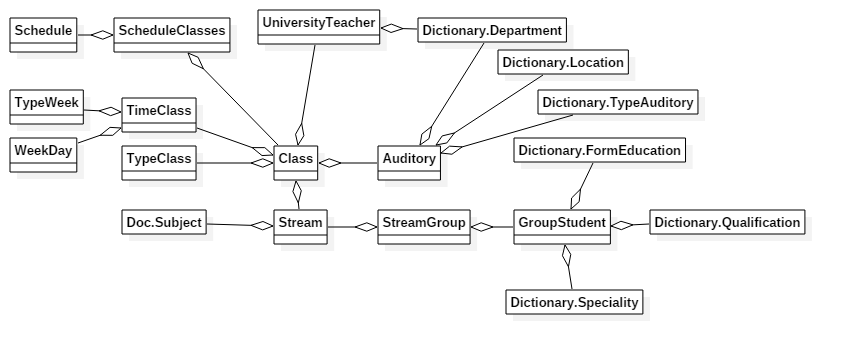


Рисунок 2.65 – Діаграма класів пакету Schedule

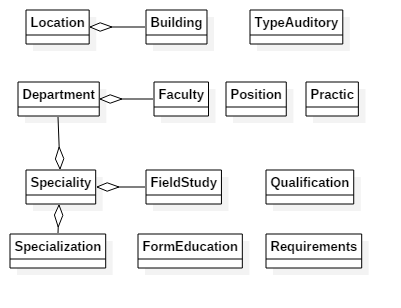


Рисунок 2.66 – Діаграма класів пакету Dictionary

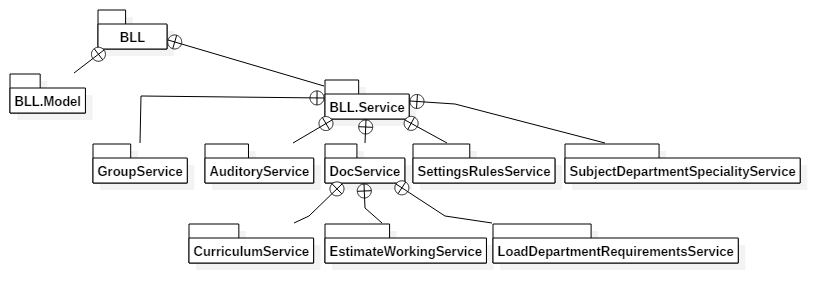


Рисунок 2.67 – Діаграма пакетів рівня бізнес логіки

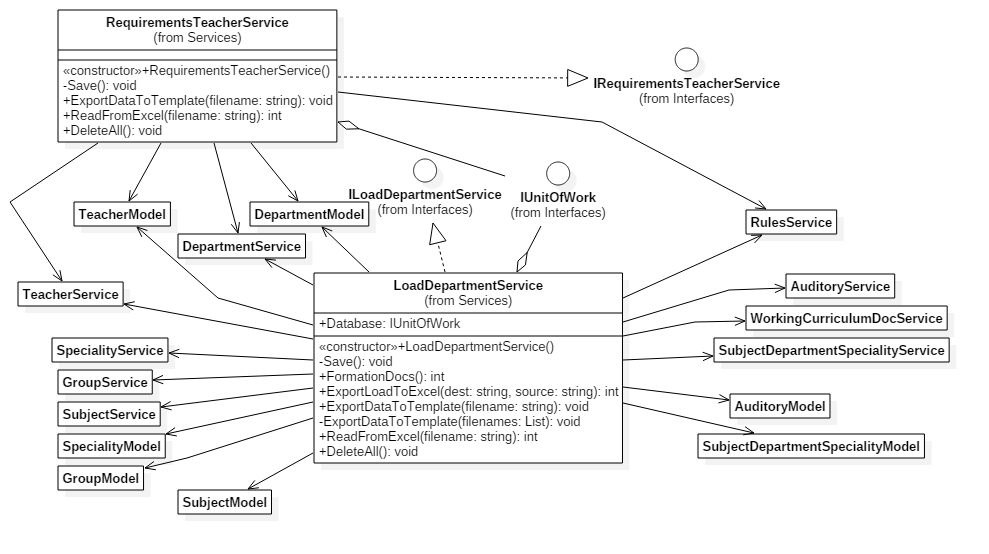


Рисунок 2.68 – Діаграма класів пакету LoadDepartmentRequirementsService

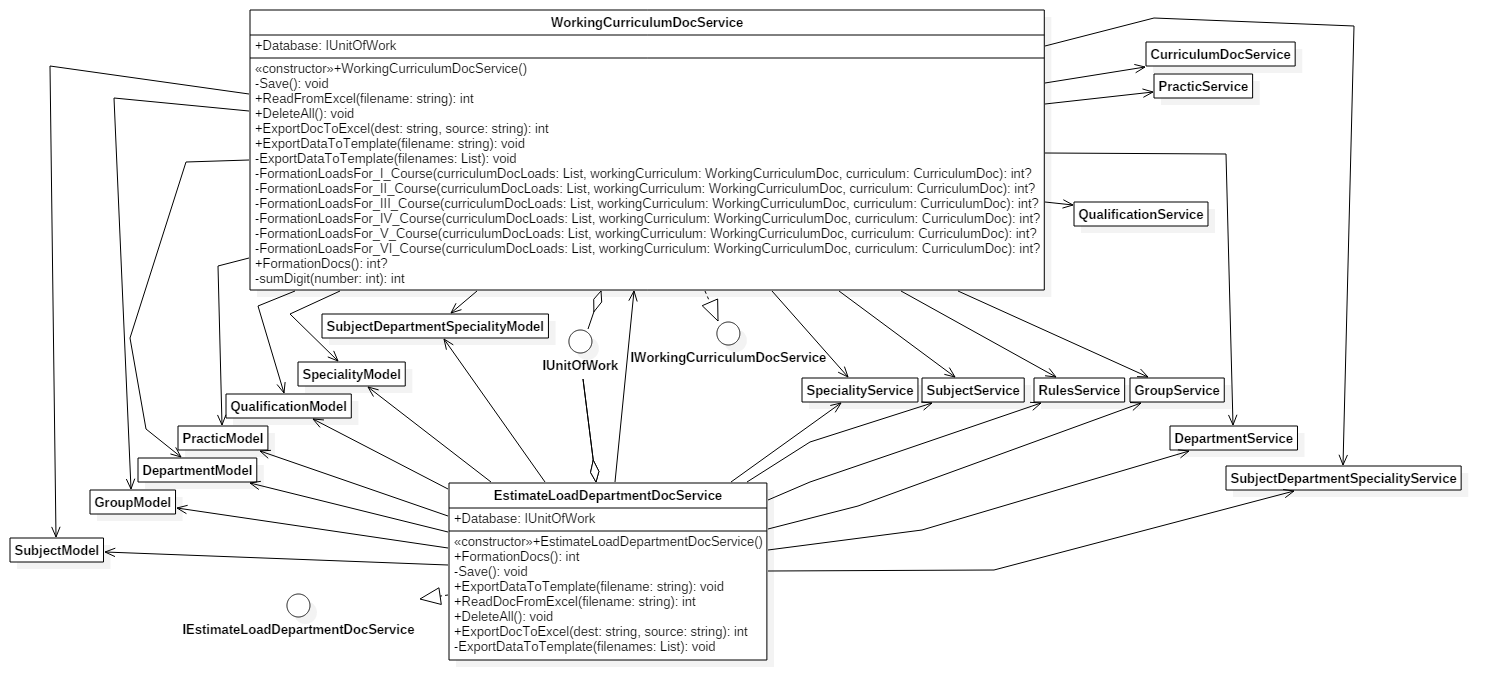


Рисунок 2.69 – Діаграма класів пакету EstimateWorkingService

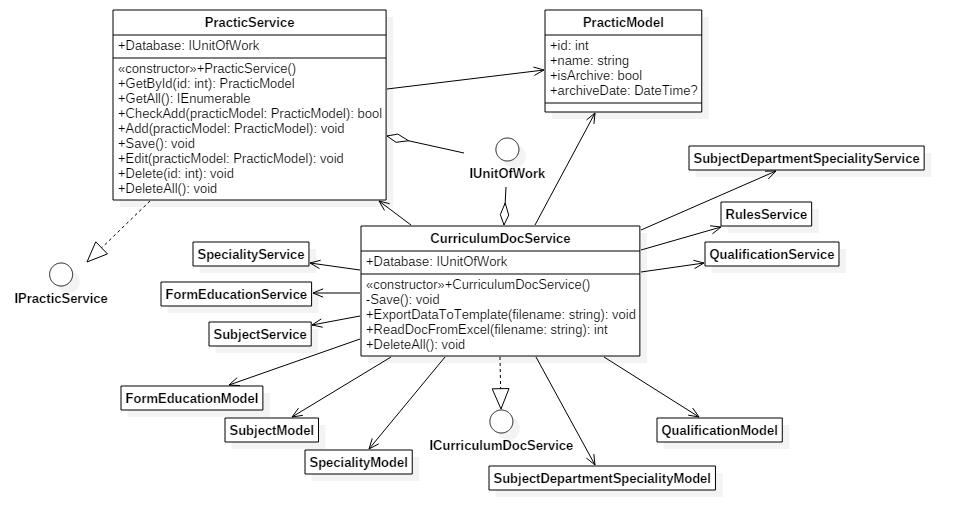


Рисунок 2.70 – Діаграма класів пакету CurriculumService

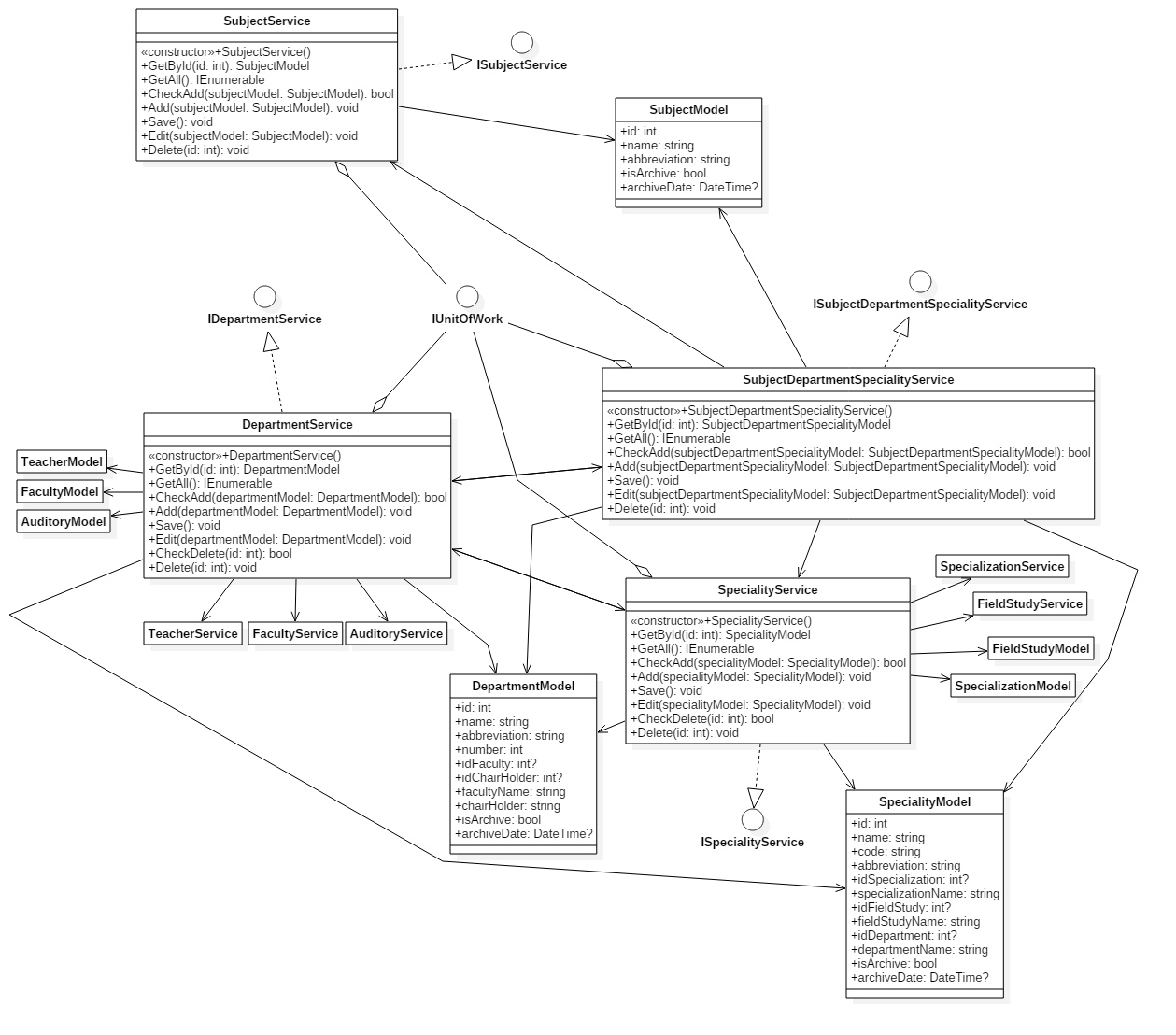


Рисунок 2.71 – Діаграма класів пакету SubjectDepartmentSpecialityService

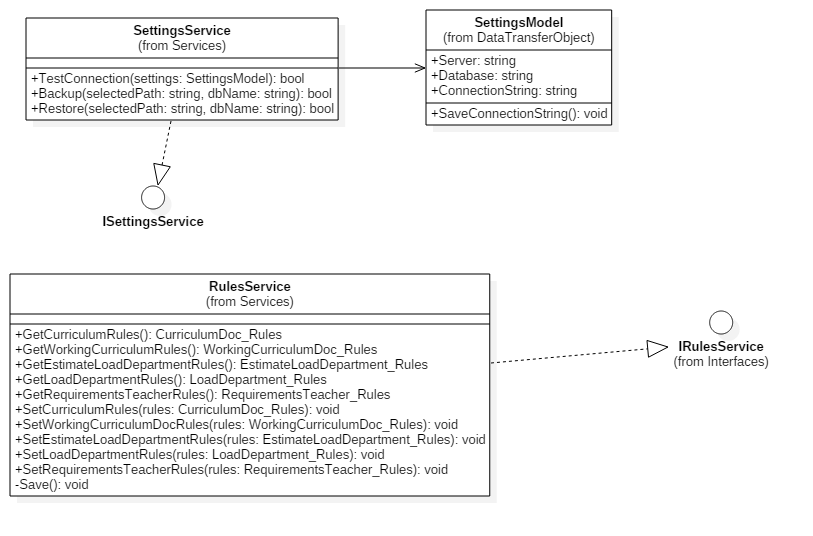


Рисунок 2.72 – Діаграма класів пакету SettingsRulesService

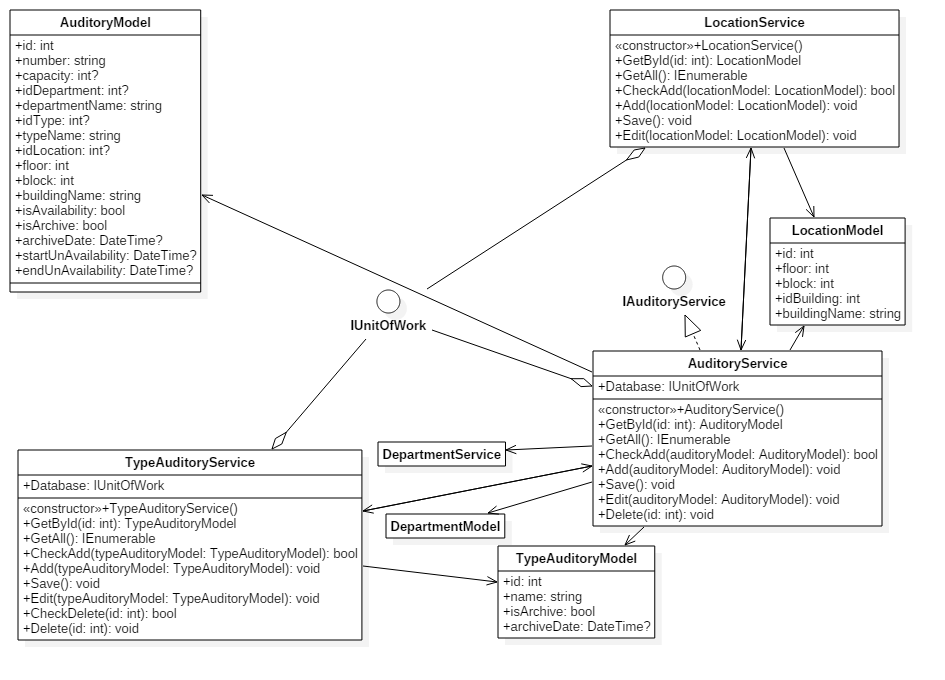


Рисунок 2.73 – Діаграма класів пакету AuditoryService

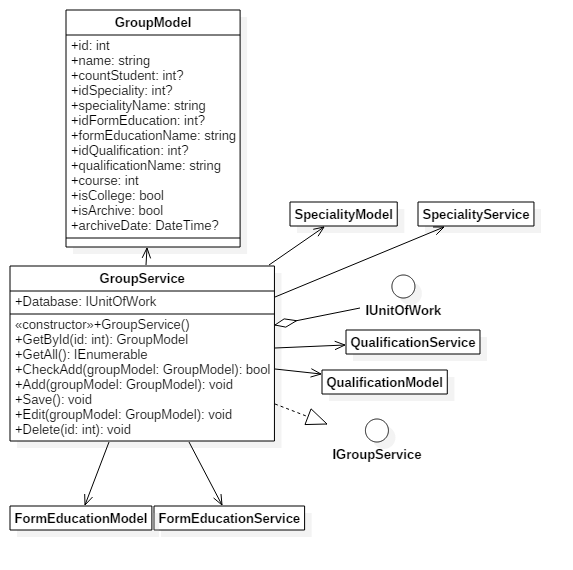


Рисунок 2.74 – Діаграма класів пакету GroupService



Рисунок 2.75 – Діаграма класів пакету BLL.Model

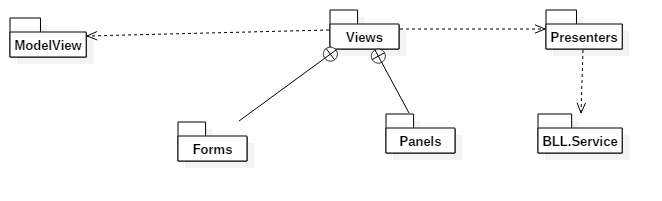


Рисунок 2.76 – Діаграма пакетів рівня представлення даних

2.5 Розробка інтерфейсу користувача

Етап розробки інтерфейсу користувача є дуже важливим етапом в створені даної системи. Це зумовлено тим, що працівникам учбового відділу необхідно оперувати великою кількістю інформації (велика кількість аудиторій, занять, тощо). Тому інтерфейс системи має забезпечувати швидкий пошук інформації, її зручне подання та максимальну кількість інформації, що одночасно відображається користувачу.

Окрім цього інтерфейс програми має бути зручним та інтуїтивно зрозумілим. Ніякі дії користувача не повинні призвести до помилок в системі. У разі виникнення непередбачуваних станів, користувач повинен отримати вичерпне повідомлення про помилку, можливі причині її виникнення та рекомендовані подальші дії.

Колір є одним з важливих атрибутів інтерфейсу і досить суб’єктивним, тому користувач має змогу самостійно його обрати.

Так як методисту доведеться оперувати досить великою кількістю даних, були додані можливості фільтру інформації за допомогою пошуку по будь-якому з полів, що дозволить залишати лише необхідні на даний момент дані.

Всі елементи інтерфейсу для зручності навігації були згруповані в меню з плиток, які відрізняються не лише заголовками, а й зображеннями (рис. 2.77).



Рисунок 2.77 – Головна форма з меню

3 ТЕСтування та налагодження програми

3.1 Вибір стратегії тестування

Для досягнення максимальної якості тестування під кожен метод або модуль програми, що тестується необхідно обрати найбільш вдалий метод або набір методів, що забезпечать необхідний результат. При цьому необхідно підібрати найкраще співвідношення між часом, що буде витрачено на тестування, та якістю тестування. Набір тестів повинен мати мінімальну збитковість, але при цьому максимально охоплювати функціональність системи [16].

Під час тестування програми буде використано декілька різних методів тестування. Умовно ці методи можна поділити на 2 типи: методи білого ящика та методи чорного ящика. Різниця між цими групами методів полягає в тому, що при використанні методів чорного ящика, тестування відбувається без доступу до коду програми, в наявності тестувальника є тільки ті можливості, що матиме користувач програми. Під час використання методів білого ящика, тестувальник використовує код програми для досягнення необхідного результату.

3.2 Тестування функціональності

3.2.1 Тестування методу ReadDocFromExcel класу CurriculumDocService

3.2.1.1 Описання методу

Метод зчитує дані з .xls файлу та завантажує їх в базу даних.

Вхідні дані: файл навчального плану в форматі .xls; правила читання файлів навчальних планів; база даних, а саме таблиці:

* CurriculmDoc;
* LoadCurriculumDoc;
* Qualification;
* FormEducation;
* Speciality;
* Subject;
* SubjectDepartmentSpeciality;
* Practic;
* LoadPractic.

Вихідні дані: результат зчитування та завантаження (-1 – не успішно; 0 – успішно; 1 – такі дані вже є), база даних.

3.2.1.2 Текст методу

public int ReadDocFromExcel(string filename)

{

int result = 0, index = 0;

string codeSpeciality = String.Empty, nameSpeciality = String.Empty;

string nameSubject = String.Empty;

bool isHalfSemester = false;

CurriculumDoc newCurriculumDoc = new CurriculumDoc();

LoadCurriculum curriculumLoad = new LoadCurriculum();

FileInfo fi = new FileInfo(filename);

using (ExcelPackage excel = new ExcelPackage(fi))

{

try

{

RulesService service = new RulesService();

CurriculumDoc\_Rules rules = service.GetCurriculumRules();

ExcelWorksheet sheet = excel.Workbook.Worksheets[rules.titleMainList];

var yearReceptionExcel = sheet.Cells[rules.yearReception\_Cell].Value;

var qualificationNameExcel = sheet.Cells[rules.qualification\_Cell].Value;

var academicTermExcel = sheet.Cells[rules.termEducation\_Cell].Value;

var formEducationNameExcel = sheet.Cells[rules.formEducation\_Cell].Value;

var specialityNameExcel = sheet.Cells[rules.speciality\_Cell].Value;

if (specialityNameExcel == null || formEducationNameExcel == null || qualificationNameExcel == null || yearReceptionExcel == null || academicTermExcel == null) **//1**

{

throw new Exception();

}

while (specialityNameExcel.ToString()[index] != ' ') **//2**

{

codeSpeciality += specialityNameExcel.ToString()[index];

index++;

}

for (index = index + 1; index < specialityNameExcel.ToString().Length; index++)

{

if (Char.IsLetter(specialityNameExcel.ToString()[index]) == true || specialityNameExcel.ToString()[index] == ' ') **//3**

{

nameSpeciality += specialityNameExcel.ToString()[index];

}

}

var qualification = Database.Qualifications.GetAll().Where(\_ => \_.isArchive == false && \_.name.ToLower() == qualificationNameExcel.ToString().ToLower()).FirstOrDefault();

var formEducation = Database.FormsEducation.GetAll().Where(\_ => \_.isArchive == false && \_.name.ToLower() == formEducationNameExcel.ToString().ToLower()).FirstOrDefault();

var speciality = Database.Specialities.GetAll().Where(\_ => \_.isArchive == false && \_.name.ToLower() == nameSpeciality.ToLower()).FirstOrDefault();

if (qualification == null) **//4**

{

Qualification qualificationModel = new Qualification

{

name = qualificationNameExcel.ToString()

};

Database.Qualifications.Create(qualificationModel);

Save();

qualification = qualificationModel;

}

if (formEducation == null) **//5**

{

FormEducation formEducationModel = new FormEducation

{

name = formEducationNameExcel.ToString()

};

Database.FormsEducation.Create(formEducationModel);

Save();

formEducation = formEducationModel;

}

if (speciality == null) **//6**

{

Speciality specialityModel = new Speciality

{

code = codeSpeciality,

name = nameSpeciality

};

Database.Specialities.Create(specialityModel);

Save();

speciality = specialityModel;

}

var curriculumDoc = Database.CurriculumDocs.GetAll().Where(\_ => \_.academicTerm == Convert.ToInt32(academicTermExcel) && \_.idFormEducation == formEducation.id && \_.idQualification == qualification.id && \_.idSpeciality == speciality.id && \_.yearReception == Convert.ToInt32(yearReceptionExcel));

if (curriculumDoc.Count() < 1) **//7**

{

newCurriculumDoc.academicTerm = Convert.ToInt32(academicTermExcel);

newCurriculumDoc.idFormEducation = formEducation.id;

newCurriculumDoc.idQualification = qualification.id;

newCurriculumDoc.idSpeciality = speciality.id;

newCurriculumDoc.yearReception = Convert.ToInt32(yearReceptionExcel);

Database.CurriculumDocs.Create(newCurriculumDoc);

Save();

sheet = excel.Workbook.Worksheets[rules.titleLearningProcessList];

for (int j = rules.startReading\_Row; j < rules.endReading\_Row + 1; j++)

{

if (sheet.Cells[rules.discipline\_Column + j].Value != null && sheet.Cells[rules.individualWork\_Column + j].Value != null && (sheet.Cells[rules.exam\_Column + j].Value != null || sheet.Cells[rules.test\_Column + j].Value != null))**//8**

{

var subjectNameExcel = sheet.Cells[rules.discipline\_Column + j].Value;

var exam = sheet.Cells[rules.exam\_Column + j].Value;

var test = sheet.Cells[rules.test\_Column + j].Value;

var courseWork = sheet.Cells[rules.courseWork\_Column + j].Value;

var lecture = sheet.Cells[rules.lecture\_Column + j].Value;

var labWork = sheet.Cells[rules.labWork\_Column + j].Value;

var practicWork = sheet.Cells[rules.practicWork\_Column + j].Value;

var individualTask = sheet.Cells[rules.individualTask\_Column + j].Value;

var individualWork = sheet.Cells[rules.individualWork\_Column + j].Value;

var numberSubject = sheet.Cells[rules.number\_Column + j].Value;

var totalHour = sheet.Cells[rules.totalHour\_Column + j].Value;

var semester1 = sheet.Cells[rules.semester1\_Column + j].Value;

var semester2 = sheet.Cells[rules.semester2\_Column + j].Value;

var semester3 = sheet.Cells[rules.semester3\_Column + j].Value;

var semester4 = sheet.Cells[rules.semester4\_Column + j].Value;

var semester5 = sheet.Cells[rules.semester5\_Column + j].Value;

var semester6 = sheet.Cells[rules.semester6\_Column + j].Value;

var semester7 = sheet.Cells[rules.semester7\_Column + j].Value;

var semester8 = sheet.Cells[rules.semester8\_Column + j].Value;

var semester9 = sheet.Cells[rules.semester9\_Column + j].Value;

var semester10 = sheet.Cells[rules.semester10\_Column + j].Value;

var semester11 = sheet.Cells[rules.semester11\_Column + j].Value;

var semester12 = sheet.Cells[rules.semester12\_Column + j].Value;

if (subjectNameExcel.ToString()[subjectNameExcel.ToString().Length - 1] == ')' && subjectNameExcel.ToString()[subjectNameExcel.ToString().Length - 2] == '8') **//9**

{

isHalfSemester = true;

nameSubject = subjectNameExcel.ToString().Substring(0, (subjectNameExcel.ToString().Length - 4));

}

else if (subjectNameExcel.ToString()[subjectNameExcel.ToString().Length - 1] == specialSubject) **//10**

{

nameSubject = subjectNameExcel.ToString().Remove(subjectNameExcel.ToString().Length - 1);

if (lecture != null && lecture.ToString()[lecture.ToString().Length - 1] == specialSubject) **//11**

{

lecture = lecture.ToString().Remove(lecture.ToString().Length - 1);

}

if (labWork != null && labWork.ToString()[labWork.ToString().Length - 1] == specialSubject) **//12**

{

labWork = labWork.ToString().Remove(labWork.ToString().Length - 1);

}

if (practicWork != null && practicWork.ToString()[practicWork.ToString().Length - 1] == specialSubject) **//13**

{

practicWork = practicWork.ToString().Remove(practicWork.ToString().Length - 1);

}

if (individualWork != null && individualWork.ToString()[individualWork.ToString().Length - 1] == specialSubject) **//14**

{

individualWork = individualWork.ToString().Remove(individualWork.ToString().Length - 1);

}

if (totalHour != null && totalHour.ToString()[totalHour.ToString().Length - 1] == specialSubject) **//15**

{

totalHour = totalHour.ToString().Remove(totalHour.ToString().Length - 1);

}

if (semester1 != null && semester1.ToString()[semester1.ToString().Length - 1] == specialSubject) **//16**

{

semester1 = semester1.ToString().Remove(semester1.ToString().Length - 1);

}

if (semester2 != null && semester2.ToString()[semester2.ToString().Length - 1] == specialSubject) **//17**

{

semester2 = semester2.ToString().Remove(semester2.ToString().Length - 1);

}

if (semester3 != null && semester3.ToString()[semester3.ToString().Length - 1] == specialSubject) **//18**

{

semester3 = semester3.ToString().Remove(semester3.ToString().Length - 1);

}

if (semester4 != null && semester4.ToString()[semester4.ToString().Length - 1] == specialSubject) **//19**

{

semester4 = semester4.ToString().Remove(semester4.ToString().Length - 1);

}

if (semester5 != null && semester5.ToString()[semester5.ToString().Length - 1] == specialSubject) **//20**

{

semester5 = semester5.ToString().Remove(semester5.ToString().Length - 1);

}

if (semester6 != null && semester6.ToString()[semester6.ToString().Length - 1] == specialSubject) **//21**

{

semester6 = semester6.ToString().Remove(semester6.ToString().Length - 1);

}

if (semester7 != null && semester7.ToString()[semester7.ToString().Length - 1] == specialSubject) **//22**

{

semester7 = semester7.ToString().Remove(semester7.ToString().Length - 1);

}

if (semester8 != null && semester8.ToString()[semester8.ToString().Length - 1] == specialSubject) **//23**

{

semester8 = semester8.ToString().Remove(semester8.ToString().Length - 1);

}

if (semester9 != null && semester9.ToString()[semester9.ToString().Length - 1] == specialSubject) **//24**

{

semester9 = semester9.ToString().Remove(semester9.ToString().Length - 1);

}

if (semester10 != null && semester10.ToString()[semester10.ToString().Length - 1] == specialSubject) **//25**

{

semester10 = semester10.ToString().Remove(semester10.ToString().Length - 1);

}

if (semester11 != null && semester11.ToString()[semester11.ToString().Length - 1] == specialSubject) **//26**

{

semester11 = semester11.ToString().Remove(semester11.ToString().Length - 1);

}

if (semester12 != null && semester12.ToString()[semester12.ToString().Length - 1] == specialSubject) **//27**

{

semester12 = semester12.ToString().Remove(semester12.ToString().Length - 1);

}

}

else**//28**

{

nameSubject = subjectNameExcel.ToString();

}

var subject = Database.Subjects.GetAll().Where(\_ => \_.isArchive == false && \_.name.ToLower() == nameSubject.ToLower()).FirstOrDefault();

if (subject == null && nameSubject.ToLower() != "військова підготовка".ToLower() && nameSubject.ToLower() != "вибіркова дисципліна".ToLower())**//29**

{

if (lecture == null && labWork == null && practicWork == null) **//30**

{

var typePractics = Database.Practics.GetAll().Where(\_ => \_.isArchive == false && \_.name.ToLower() == nameSubject.ToLower()).FirstOrDefault();

if (typePractics == null) **//31**

{

Practic practicModel = new Practic();

practicModel.name = nameSubject;

Database.Practics.Create(practicModel);

Save();

typePractics = practicModel;

}

LoadPractic loadPractic = new LoadPractic

{

countHour = Convert.ToInt32(individualWork),

idPractic = typePractics.id,

idSpeciality = speciality.id,

semester = Convert.ToInt32(test),

yearReception = Convert.ToInt32(yearReceptionExcel)

};

Database.PracticLoads.Create(loadPractic);

Save();

continue;

}

else**//32**

{

Subject subjectModel = new Subject

{

name = nameSubject

};

Database.Subjects.Create(subjectModel);

Save();

subject = subjectModel;

}

}

if (subject != null) **//33**

{

var subjectsDepartmentSpeciality = Database.DepartmentSpecialitySubjects.GetAll().Where(\_ => \_.isArchive == false && \_.idSubject == subject.id && \_.idSpeciality == speciality.id);

if (subjectsDepartmentSpeciality.Count() < 1) **//34**

{

SubjectDepartmentSpeciality subjectDepartmentSpeciality = new SubjectDepartmentSpeciality

{

idSpeciality = speciality.id,

idSubject = subject.id

};

Database.DepartmentSpecialitySubjects.Create(subjectDepartmentSpeciality);

Save();

}

curriculumLoad.countHourIndividualWork = Convert.ToInt32(individualWork);

curriculumLoad.countHourLabWork = Convert.ToInt32(labWork);

curriculumLoad.countHourLecture = Convert.ToInt32(lecture);

curriculumLoad.countHourPracticWork = Convert.ToInt32(practicWork);

curriculumLoad.idCurriculumDoc = curriculumDoc.First().id;

curriculumLoad.idSubject = subject.id;

curriculumLoad.individualTask = Convert.ToString(individualTask);

curriculumLoad.totalHour = Convert.ToInt32(totalHour);

if (exam != null) **//35**

{

curriculumLoad.isExam = true;

}

else**//36**

{

curriculumLoad.isExam = false;

}

if (test != null) **//37**

{

curriculumLoad.isTest = true;

}

else**//38**

{

curriculumLoad.isTest = false;

}

curriculumLoad.isHalfSemester = isHalfSemester;

curriculumLoad.numberSubject = Convert.ToInt32(numberSubject);

curriculumLoad.semesterI = Convert.ToInt32(semester1);

curriculumLoad.semesterII = Convert.ToInt32(semester2);

curriculumLoad.semesterIII = Convert.ToInt32(semester3);

curriculumLoad.semesterIV = Convert.ToInt32(semester4);

curriculumLoad.semesterV = Convert.ToInt32(semester5);

curriculumLoad.semesterVI = Convert.ToInt32(semester6);

curriculumLoad.semesterVII = Convert.ToInt32(semester7);

curriculumLoad.semesterVIII = Convert.ToInt32(semester8);

curriculumLoad.semesterIX = Convert.ToInt32(semester9);

curriculumLoad.semesterX = Convert.ToInt32(semester10);

curriculumLoad.semesterXI = Convert.ToInt32(semester11);

curriculumLoad.semesterXII = Convert.ToInt32(semester12);

Database.CurriculumDocLoads.Create(curriculumLoad);

Save();

}

}

}

}

else**//39**

{

result = 1;

}

}

catch**//40**

{

result = -1;

DeleteAll();

Database.PracticLoads.DeleteAll();

Save();

}

return result;

}

}

3.2.1.3 Вибір методу тестування

Тестування даного методу було вирішено проводити одним із методів білого ящика, оскільки дані методи дозволять досягти кращої якості тестування, ніж методи чорного ящика. Серед методів білого ящика було обрано метод покриття операторів. Його використання забезпечить найкраще відношення якості тестування до витраченого часу. Метод дозволить перевірити досяжність та правильність роботи кожного оператора. Для того, щоб кожен оператор був виконаний, набір тестів має пройти всі гілки коду. Вони помічені коментарями з числами у тексті методу.

3.2.1.4 Формування набору тестів

Вхідні дані тесту 1: файл навчального плану (табл. 3.1) та рис. 3.1, правила читання (табл. 3.2), таблиці бази даних порожні.

Таблиця 3.1 – Склад файлу навчального плану, титульний лист

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Рік прийому | Кваліфікаційний рівень | Термін навчання | Форма навчання | Спеціальність |
| 2015 | бакалавр | 4 | денна |  |

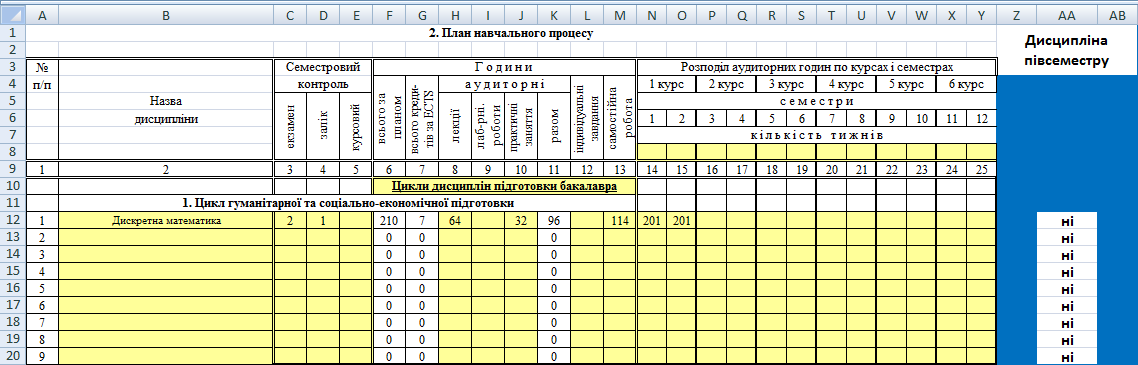


Рисунок 3.1 – Вхідні дані для тесту 1, лист навчальний процес файлу навчальний план

Таблиця 3.2 – Правила читання файлів навчальних планів

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Рік прийому | Кваліфікаційний рівень | Термін навчання | Форма навчання | Спеціальність |
| AS1 | AT3 | AP4 | AB11 | AB17 |

Вихідні дані тесту 1: результат зчитування – -1; таблиці бази даних, які використовуються залишилися порожніми.

Коментар: файл, що передається не містить спеціальності, а отже він не відповідає шаблону і результат виконання -1, а таблиці не змінилися. Тест проходить правильно.

Вхідні дані тесту 2: файл навчального плану (табл. 3.3) та рис. 3.2, правила читання (табл. 3.4), таблиці бази даних порожні.

Таблиця 3.3 – Склад файлу навчального плану, титульний лист

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Рік прийому | Кваліфікаційний рівень | Термін навчання | Форма навчання | Спеціальність |
| 2015 | бакалавр | 4 | денна | 7.05010301 програмне забезпечення систем |

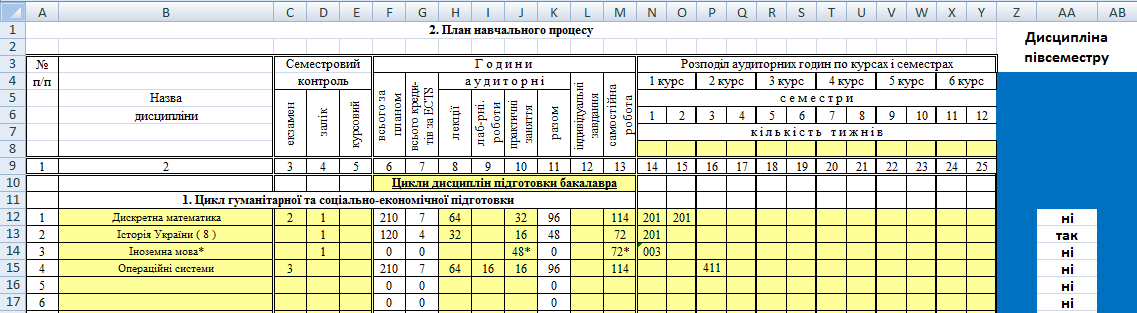


Рисунок 3.2 – Вхідні дані для тесту 2, лист навчальний процес файлу навчальний план

Таблиця 3.4 – Правила читання файлів навчальних планів

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Рік прийому | Кваліфікаційний рівень | Термін навчання | Форма навчання | Спеціальність |
| AS1 | AT3 | AP4 | AB11 | AB17 |

Вихідні дані тесту 2: результат зчитування – 0; таблиці бази даних   
(табл. 3.5 – 3.10) інші залишилися порожніми.

Коментар: база даних порожня, файл, що передається відповідає шаблону. Містить: дисципліни на півсеместру; спеціальні дисципліни, які помічаються \*; дисципліни тільки з екзаменом; дисципліни тільки з заліком; дисципліни з заліком і екзаменом. Тест проходить правильно.

Таблиця 3.5 – Склад таблиці «Документ навчального плану»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Рік прийому | Кваліфікаційний рівень | Термін навчання | Форма навчання | Спеціальність |
| 2015 | бакалавр | 4 | денна | програмне забезпечення систем |

Таблиця 3.6 – Склад таблиці «Кваліфікаційний рівень»

|  |  |  |
| --- | --- | --- |
| Назва | Архів | Дата архіву |
| бакалавр | ні | - |

Таблиця 3.7 – Склад таблиці «Форма навчання»

|  |  |  |
| --- | --- | --- |
| Назва | Архів | Дата архіву |
| денна | ні | - |

Таблиця 3.8 – Склад таблиці «Спеціальність»

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Назва | Коротка назва | Код | Спеціалізація | Галузь знань | Кафедра | Архів | Дата архіву |
| програмне забезпечення систем | - | 7.05010301 | - | - | - | ні | - |

Таблиця 3.9 – Склад таблиці «Дисципліна»

|  |  |  |  |
| --- | --- | --- | --- |
| Назва | Коротка назва | Архів | Дата архіву |
| Дискретна математика | - | ні | - |
| Історія України | - | ні | - |
| Іноземна мова | - | ні | - |
| Операційні системи | - | ні | - |

Таблиця 3.10 – Склад таблиці «Дисципліна\_Кафедра\_Спеціальність»

|  |  |  |
| --- | --- | --- |
| Дисципліна | Кафедра | Спеціальність |
| Дискретна математика | - | програмне забезпечення систем |
| Історія України |  | програмне забезпечення систем |
| Операційні системи |  | програмне забезпечення систем |
| Іноземна мова |  | програмне забезпечення систем |

Склад таблиці «Навантаження документу навчальний план»:

**Запис1:**

Дисципліна: Дискретна математика; Дисципліна на півсеместру: ні;

Номер дисципліни: 1; Кількість годин на лекції: 64;

Кількість годин на лабораторні: 0; Кількість годин на практичні: 32;

Екзамен: 2; Залік: 1; Індивідуальне завдання: -;

Кількість годин на самостійну роботу: 114; Всього годин: 210;

Семестр1: 201; Семестр2: 201; Семестр3: 0; Семестр4: 0; Семестр5: 0;

Семестр6: 0; Семестр7: 0; Семестр8: 0; Семестр9: 0; Семестр10: 0;

Семестр11: 0; Семестр12: 0.

**Запис2:**

Дисципліна: Історія України; Дисципліна на півсеместру: так;

Номер дисципліни: 2; Кількість годин на лекції: 32;

Кількість годин на лабораторні: 0; Кількість годин на практичні: 16;

Екзамен: -; Залік: 1; Індивідуальне завдання: -;

Кількість годин на самостійну роботу: 72; Всього годин: 120;

Семестр1: 201; Семестр2: 0; Семестр3: 0; Семестр4: 0; Семестр5: 0;

Семестр6: 0; Семестр7: 0; Семестр8: 0; Семестр9: 0; Семестр10: 0;

Семестр11: 0; Семестр12: 0.

**Запис3:**

Дисципліна: Іноземна мова; Дисципліна на півсеместру: ні;

Номер дисципліни: 3; Кількість годин на лекції: 0;

Кількість годин на лабораторні: 0; Кількість годин на практичні: 48;

Екзамен: -; Залік: 1; Індивідуальне завдання: -;

Кількість годин на самостійну роботу: 72; Всього годин: 0; Семестр1: 003;

Семестр2: 0; Семестр3: 0; Семестр4: 0; Семестр5: 0; Семестр6: 0;

Семестр7: 0; Семестр8: 0; Семестр9: 0; Семестр10: 0; Семестр11: 0;

Семестр12: 0.

**Запис4:**

Дисципліна: Операційні системи; Дисципліна на півсеместру: ні;

Номер дисципліни: 4; Кількість годин на лекції: 64;

Кількість годин на лабораторні: 16; Кількість годин на практичні: 16;

Екзамен: 3; Залік: - ; Індивідуальне завдання: - ;

Кількість годин на самостійну роботу: 114; Всього годин: 210;

Семестр1: 0; Семестр2: 0; Семестр3: 411; Семестр4: 0; Семестр5: 0;

Семестр6: 0; Семестр7: 0; Семестр8: 0; Семестр9: 0; Семестр10: 0;

Семестр11: 0; Семестр12: 0.

Вхідні дані тесту 3: файл навчального плану (табл. 3.11) та рис. 3.3, правила читання (табл. 3.12), таблиці бази даних порожні, окрім таблиці «Документ навчальний план» (табл. 3.13) і таблиці «Навантаження документу навчального плану».

Таблиця 3.11 – Склад файлу навчального плану, титульний лист

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Рік прийому | Кваліфікаційний рівень | Термін навчання | Форма навчання | Спеціальність |
| 2015 | бакалавр | 4 | денна | 7.05010301 програмне забезпечення систем |

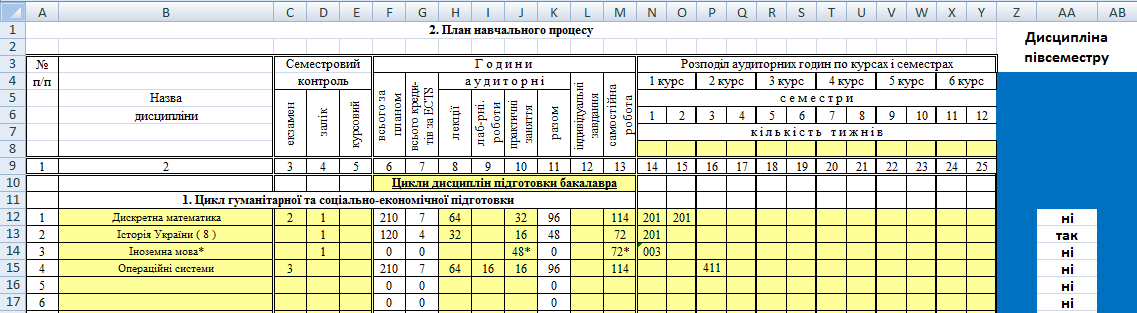


Рисунок 3.3 – Вхідні дані для тесту 3, лист навчальний процес файлу навчальний план

Таблиця 3.12 – Правила читання файлів навчальних планів

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Рік прийому | Кваліфікаційний рівень | Термін навчання | Форма навчання | Спеціальність |
| AS1 | AT3 | AP4 | AB11 | AB17 |

Таблиця 3.13 – Склад таблиці «Документ навчального плану»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Рік прийому | Кваліфікаційний рівень | Термін навчання | Форма навчання | Спеціальність |
| 2015 | бакалавр | 4 | денна | програмне забезпечення систем |

Склад таблиці «Навантаження документу навчальний план»:

**Запис1:**

Дисципліна: Дискретна математика; Дисципліна на півсеместру: ні;

Номер дисципліни: 1; Кількість годин на лекції: 64;

Кількість годин на лабораторні: 0; Кількість годин на практичні: 32;

Екзамен: 2; Залік: 1; Індивідуальне завдання: -;

Кількість годин на самостійну роботу: 114; Всього годин: 210;

Семестр1: 201; Семестр2: 201; Семестр3: 0; Семестр4: 0; Семестр5: 0;

Семестр6: 0; Семестр7: 0; Семестр8: 0; Семестр9: 0; Семестр10: 0;

Семестр11: 0; Семестр12: 0.

**Запис2:**

Дисципліна: Історія України; Дисципліна на півсеместру: так;

Номер дисципліни: 2; Кількість годин на лекції: 32;

Кількість годин на лабораторні: 0; Кількість годин на практичні: 16;

Екзамен: -; Залік: 1; Індивідуальне завдання: -;

Кількість годин на самостійну роботу: 72; Всього годин: 120;

Семестр1: 201; Семестр2: 0; Семестр3: 0; Семестр4: 0; Семестр5: 0;

Семестр6: 0; Семестр7: 0; Семестр8: 0; Семестр9: 0; Семестр10: 0;

Семестр11: 0; Семестр12: 0.

**Запис3:**

Дисципліна: Іноземна мова; Дисципліна на півсеместру: ні;

Номер дисципліни: 3; Кількість годин на лекції: 0;

Кількість годин на лабораторні: 0; Кількість годин на практичні: 48;

Екзамен: -; Залік: 1; Індивідуальне завдання: -;

Кількість годин на самостійну роботу: 72; Всього годин: 0; Семестр1: 003;

Семестр2: 0; Семестр3: 0; Семестр4: 0; Семестр5: 0; Семестр6: 0;

Семестр7: 0; Семестр8: 0; Семестр9: 0; Семестр10: 0; Семестр11: 0;

Семестр12: 0.

**Запис4:**

Дисципліна: Операційні системи; Дисципліна на півсеместру: ні;

Номер дисципліни: 4; Кількість годин на лекції: 64;

Кількість годин на лабораторні: 16; Кількість годин на практичні: 16;

Екзамен: 3; Залік: - ; Індивідуальне завдання: - ;

Кількість годин на самостійну роботу: 114; Всього годин: 210;

Семестр1: 0; Семестр2: 0; Семестр3: 411; Семестр4: 0; Семестр5: 0;

Семестр6: 0; Семестр7: 0; Семестр8: 0; Семестр9: 0; Семестр10: 0;

Семестр11: 0; Семестр12: 0.

Вихідні дані: результат зчитування – 1; таблиці бази даних (табл. 3.14 – 3.17).

Коментар: повторне завантаження файлу, результат 1 отже така інформація вже була завантажена. Тест проходить правильно.

Таблиця 3.14 – Склад таблиці «Документ навчального плану»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Рік прийому | Кваліфікаційний рівень | Термін навчання | Форма навчання | Спеціальність |
| 2015 | бакалавр | 4 | денна | програмне забезпечення систем |

Таблиця 3.15 – Склад таблиці «Кваліфікаційний рівень»

|  |  |  |
| --- | --- | --- |
| Назва | Архів | Дата архіву |
| бакалавр | ні | - |

Таблиця 3.16 – Склад таблиці «Форма навчання»

|  |  |  |
| --- | --- | --- |
| Назва | Архів | Дата архіву |
| денна | ні | - |

Таблиця 3.17 – Склад таблиці «Спеціальність»

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Назва | Коротка назва | Код | Спеціалізація | Галузь знань | Кафедра | Архів | Дата архіву |
| програмне забезпечення систем | - | 7.05010301 | - | - | - | ні | - |

Склад таблиці «Навантаження документу навчальний план»:

**Запис1:**

Дисципліна: Дискретна математика; Дисципліна на півсеместру: ні;

Номер дисципліни: 1; Кількість годин на лекції: 64;

Кількість годин на лабораторні: 0; Кількість годин на практичні: 32;

Екзамен: 2; Залік: 1; Індивідуальне завдання: -;

Кількість годин на самостійну роботу: 114; Всього годин: 210;

Семестр1: 201; Семестр2: 201; Семестр3: 0; Семестр4: 0; Семестр5: 0;

Семестр6: 0; Семестр7: 0; Семестр8: 0; Семестр9: 0; Семестр10: 0;

Семестр11: 0; Семестр12: 0.

**Запис2:**

Дисципліна: Історія України; Дисципліна на півсеместру: так;

Номер дисципліни: 2; Кількість годин на лекції: 32;

Кількість годин на лабораторні: 0; Кількість годин на практичні: 16;

Екзамен: -; Залік: 1; Індивідуальне завдання: -;

Кількість годин на самостійну роботу: 72; Всього годин: 120;

Семестр1: 201; Семестр2: 0; Семестр3: 0; Семестр4: 0; Семестр5: 0;

Семестр6: 0; Семестр7: 0; Семестр8: 0; Семестр9: 0; Семестр10: 0;

Семестр11: 0; Семестр12: 0.

**Запис3:**

Дисципліна: Іноземна мова; Дисципліна на півсеместру: ні;

Номер дисципліни: 3; Кількість годин на лекції: 0;

Кількість годин на лабораторні: 0; Кількість годин на практичні: 48;

Екзамен: -; Залік: 1; Індивідуальне завдання: -;

Кількість годин на самостійну роботу: 72; Всього годин: 0; Семестр1: 003;

Семестр2: 0; Семестр3: 0; Семестр4: 0; Семестр5: 0; Семестр6: 0;

Семестр7: 0; Семестр8: 0; Семестр9: 0; Семестр10: 0; Семестр11: 0;

Семестр12: 0.

**Запис4:**

Дисципліна: Операційні системи; Дисципліна на півсеместру: ні;

Номер дисципліни: 4; Кількість годин на лекції: 64;

Кількість годин на лабораторні: 16; Кількість годин на практичні: 16;

Екзамен: 3; Залік: - ; Індивідуальне завдання: - ;

Кількість годин на самостійну роботу: 114; Всього годин: 210;

Семестр1: 0; Семестр2: 0; Семестр3: 411; Семестр4: 0; Семестр5: 0;

Семестр6: 0; Семестр7: 0; Семестр8: 0; Семестр9: 0; Семестр10: 0;

Семестр11: 0; Семестр12: 0.

3.2.2 Тестування методу FormationDocs класу WorkingCurriculumDocService

3.2.2.1 Опис методу

Метод призначений для формування робочих навчальних планів. Вхідні дані: вміст бази даних, а саме таблиці:

* CurriculmDoc;
* Qualification;
* FormEducation;
* Speciality;
* WorkingCurriculumDoc;
* Group.

Вихідні дані: результат формування, база даних.

3.2.2.2 Вибір методу тестування

Оскільки користувач ніяк не може вплинути на перебіг формування, єдиним методом тестування залишається метод припущення про помилку.

3.2.2.3 Формування набору тестів

Припускається, що при формуванні робочих навчальних планів, зустрінеться такий навчальний план в якому рік прийому буде більше нинішнього або значно менше.

Вхідні дані тесту 1: таблиці бази даних (табл. 3.17 – 3.20) інші таблиці порожні.

Таблиця 3.17 – Склад таблиці «Документ навчального плану»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Рік прийому | Кваліфікаційний рівень | Термін навчання | Форма навчання | Спеціальність |
| 3015 | бакалавр | 4 | денна | програмне забезпечення систем |

Таблиця 3.18 – Склад таблиці «Кваліфікаційний рівень»

|  |  |  |
| --- | --- | --- |
| Назва | Архів | Дата архіву |
| бакалавр | ні | - |

Таблиця 3.19 – Склад таблиці «Форма навчання»

|  |  |  |
| --- | --- | --- |
| Назва | Архів | Дата архіву |
| денна | ні | - |

Таблиця 3.20 – Склад таблиці «Спеціальність»

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Назва | Коротка назва | Код | Спеціалізація | Галузь знань | Кафедра | Архів | Дата архіву |
| програмне забезпечення систем | - | 7.05010301 | - | - | - | ні | - |

Вихідні дані тесту 1: результат формування – 0; таблиці бази даних не змінилися.

Коментар: тест проходить успішно, припущення – невірне.

Вхідні дані тесту 2: таблиці бази даних (табл. 3.21 – 3.24) інші таблиці порожні.

Таблиця 3.21 – Склад таблиці «Документ навчального плану»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Рік прийому | Кваліфікаційний рівень | Термін навчання | Форма навчання | Спеціальність |
| 1015 | бакалавр | 4 | денна | програмне забезпечення систем |

Таблиця 3.22 – Склад таблиці «Кваліфікаційний рівень»

|  |  |  |
| --- | --- | --- |
| Назва | Архів | Дата архіву |
| бакалавр | ні | - |

Таблиця 3.23 – Склад таблиці «Форма навчання»

|  |  |  |
| --- | --- | --- |
| Назва | Архів | Дата архіву |
| денна | ні | - |

Таблиця 3.24 – Склад таблиці «Спеціальність»

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Назва | Коротка назва | Код | Спеціалізація | Галузь знань | Кафедра | Архів | Дата архіву |
| програмне забезпечення систем | - | 7.05010301 | - | - | - | ні | - |

Вихідні дані: результат формування – 0; таблиці бази даних не змінилися.

Коментар: Тест проходить успішно, припущення – невірне.

3.2.3 Тестування методу Formation класу LoadDepartmentService

3.2.3.1 Опис методу

Метод призначений для формування документів «Розподіл навантаження кафедри». Вхідні дані: вміст бази даних, а саме таблиці:

* WorkingCurriculmDoc;
* LoadWorkingCurriculum;
* SubjectDepartmentSpeciality;
* Department;
* LoadDepartmentDoc;
* LoadDepartmentDocLoads.

Вихідні дані: результат формування, база даних.

3.2.3.2 Вибір методу тестування

Так як і в попередньому методі оскільки користувач ніяк не може вплинути на перебіг формування, єдиним методом тестування залишається метод припущення про помилку.

3.2.3.3 Формування набору тестів

Помилка може виникнути лише, якщо некоректні дані в робочих навчальних планах, або якщо їх нема. Тому припускається, що при формуванні документів «Розподіл навантаження кафедри» не буде знайдено робочих навчальних планів.

**Тест1:**

Вхідні дані: таблиці порожні.

Вихідні дані: результат формування – 1, таблиці бази даних залишились порожніми.

Коментар: так як робочі навчальні плани не були знайдені, то результат формування 1, а вміст бази даних не змінився.

3.2.4 Результати тестування

3.2.4.1 Результати тестування методу ReadDocFromExcel класу CurriculumDocService

В ході виконання набору тестів були покриті всі оператори (табл. 3.25).

Таблиця 3.25 – Результати тестування методу ReadDocFromExcel класу CurriculumDocService

|  |  |  |  |
| --- | --- | --- | --- |
| Гілка Тест | 1 | 2 | 3 |
| 1 | + |  |  |
| 2 |  | + | + |
| 3 |  | + | + |
| 4 |  | + | + |
| 5 |  | + | + |
| 6 |  | + | + |
| 7 |  | + |  |
| 8 |  | + |  |
| 9 |  | + |  |
| 10 |  | + |  |
| 11 |  | + |  |
| 12 |  | + |  |

Продовження таблиці 3.25

|  |  |  |  |
| --- | --- | --- | --- |
| Гілка Тест | 1 | 2 | 3 |
| 13 |  | + |  |
| 14 |  | + |  |
| 15 |  | + |  |
| 16 |  | + |  |
| 17 |  | + |  |
| 18 |  | + |  |
| 19 |  | + |  |
| 20 |  | + |  |
| 21 |  | + |  |
| 22 |  | + |  |
| 23 |  | + |  |
| 24 |  | + |  |
| 25 |  | + |  |
| 26 |  | + |  |
| 27 |  | + |  |
| 28 |  | + |  |
| 29 |  | + |  |
| 30 |  | + |  |
| 31 |  | + |  |
| 32 |  | + |  |
| 33 |  | + |  |
| 34 |  | + |  |
| 35 |  | + |  |
| 36 |  | + |  |
| 37 |  | + |  |
| 38 |  | + |  |
| 39 |  |  | + |
| 40 | + |  |  |

3.2.4.2 Результати тестування методу FormationDocs класу WorkingCurriculumDocService

Після автоматичного формування документів робочих навчальних планів аномалій і аномальних даних не виявлено, отже тест завершився вдало.

3.2.4.3 Результати тестування методу Formation класу LoadDepartmentService

Після автоматичного формування документів «Розподіл навантаження кафедри» база залишилась порожньою, отже тест завершився вдало.

3.3 Налагодження програми

Під час копіювання папки з усім вмістимим, було викинуте невідоме виключення. Необхідно провести налагодження програми, для усунення даної помилки [24].

Текст методу, що відлагоджується:

private void CopyDirectory(string from, string to)

{

Directory.CreateDirectory(to);

foreach (string item in Directory.GetFiles(from))

{

File.Copy(item, to + "\\" + Path.GetFileName(item));

}

foreach (string item in Directory.GetDirectories(from))

{

CopyDirectory(item, to + "\\" + Path.GetFileName(item));

}

}

Спершу необхідно знайти оператор, на якому викидається виключення. Для цього в меню підменю «Виключення» меню «Відладка» включимо наступні опції у розділі «Перехоплювати виключення не перехоплені кодом користувача»:«Common Language Runtime Exceptions», «Managed Debugging Assistants», «Native Runtime Checks», «Win32 Exceptions». Після цього повторивши дії, що призводять до виникнення помилки, налагодження зупиниться у місці помилки.

У результаті виконання цих дій було виявлено, що виключення викидається на рядку Directory.CreateDirectory(to).

Є два варіанти можливої причини:

* така папка вже існує;
* некоректний шлях.

Для перевірки першого припущення поставимо точку зупинки на методі, в якому викидається виключення (рис. 3.4). Повторимо дії, що призводять до виникнення помилки. Робота програми зупинилася без викиду виключень у цій точці. Після виконання ще одного кроку програми з’явилася помилка. Це доводить вірність перевіреного припущення.

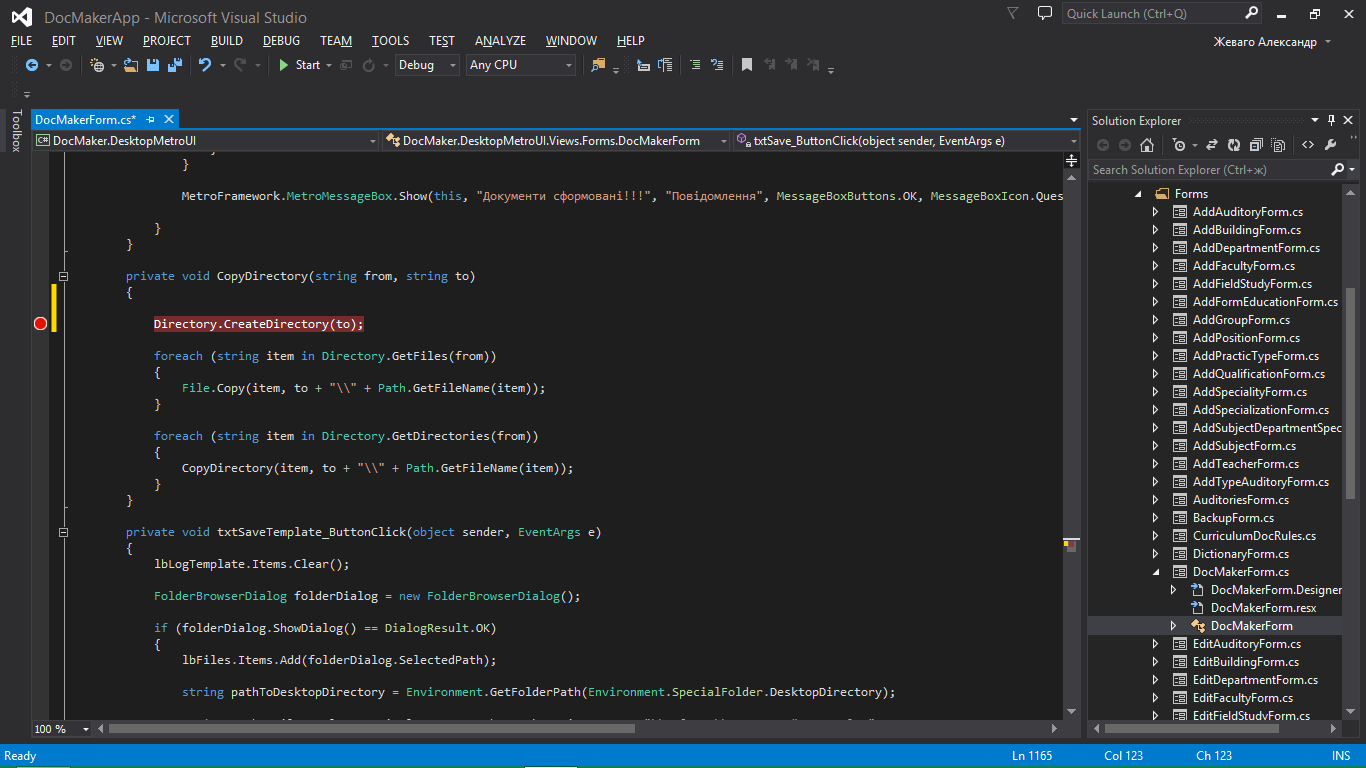


Рисунок 3.4 – Додана точка зупинки

Коли помилка локалізована необхідно визначитись з методом її виправлення. Є 2 найпростіші варіанти вирішення проблеми. Можна «заглушити» переривання порожнім блоком «try-catch». Таке рішення не є вдалим, оскільки механізм перехоплення помилок є доволі затратним по часу і його використання у рекурсії погіршить швидкодію системи. Крім цього можна поставити додаткову умову перевірки на існування папки. Другий варіант є більш прийнятним, його і приймемо.

В результаті код буде виглядати наступним чином:

private void CopyDirectory(string from, string to)

{

if (!Directory.Exists(to))

{

Directory.CreateDirectory(to);

}

foreach (string item in Directory.GetFiles(from))

{

File.Copy(item, to + "\\" + Path.GetFileName(item));

}

foreach (string item in Directory.GetDirectories(from))

{

CopyDirectory(item, to + "\\" + Path.GetFileName(item));

}

}

Після внесення необхідних змін до коду прибираємо точку зупинки та повторюємо дії, що призводили до помилки. Помилка не виникла, можна зробити висновок, що вона була виправлена.

4 Охорона праці та безпека у надзвичайних ситуаціях

«Система доступу до енциклопедичних знань на природній мові» є програмним продуктом, який дозволить використовувати енциклопедичну інформацію з навчальною або довідковою метою. Інтерфейс взаємодії з системою реалізований через систему текстових або голосових запитів та відповідей. Це стане у нагоді як звичайним користувачам, так і користувачам з обмеженими можливостями. Під час роботи з даною системою, користувач буде працювати з персональним комп’ютером, клавіатурою, монітором, або ноутбуком, тому слід зазначити правила користування електронно-обчислювальними машинами та відео-дисплейними терміналами.

5.1 Аналіз шкідливих та небезпечних виробничих факторів

Небезпечні виробничі чинники – це виробничі чинники, вплив яких на користувача в певних умовах призводить до травм, гострого отруєння або іншого раптового різкого погіршення здоров'я або до смерті [8].

До різкого погіршення здоров'я можна віднести отруєння, опромінення, удар електрострумом, тепловий удар та ін.

До небезпечних факторів відносяться електронебезпека та пожежонебезпека.

Шкідливі виробничі фактори – фактори, тривалий вплив яких на працюючого у визначених умовах приведе до захворювання, зниження працездатності i (або) негативного впливу на здоров'я нащадків [8]. У залежності від рівня і тривалості впливу шкідливі фактори можуть класифікуватися і як небезпечні.

Згідно санітарно-гігієнічним вимогам [9] умови праці робітника, який працює з автономною частиною «Системи доступу до енциклопедичних знань на природній мові» на ЕОМ, повинні відповідати I або II класу.

Робота на ЕОМ пов'язана з наступними шкідливими факторами:

* недолік освітлення природним світлом;
* відблиски на екрані монітора;
* електромагнітне поле.

Недостатнє освітлення приводить до швидкої стомленості очей, що у свою чергу, призводить до зниження продуктивності праці і росту кількості прийнятих помилкових рішень.

Відблиски на екрані електронно-променевого монітора, що виникають при неправильному освітленні, приводить до погіршення зору, а у випадку тривалого впливу даного небезпечного фактору, може призвести до повної втрати зору. З метою зниження рівня впливу на робітника даного шкідливого фактору, варто дотримуватись вимог [10] або застосовувати рідкокристалічні монітори, які в силу своєї конструкції і використовуваних матеріалів мають менший коефіцієнт відбиття світла, ніж електронно-променевий монітор, тому відблисків на них практично не буває.

Рівні електромагнітного випромінювання і магнітних полів на робочому місці повинні відповідати вимогам [11].

5.1.1 Небезпечні чинники пов’язані з пожежами в будівлях університету

Пожежі становлять особливу небезпеку для життя людини, і можуть призвести до великих матеріальних втрат. Джерелами загоряння можуть виявитися електронні схеми ЕОМ, елементи, що перегрілися, які здатні викликати загоряння пальних матеріалів.

Для всіх споруджень та приміщень, у яких експлуатується автономна частина «Системи доступу до енциклопедичних знань на природній мові», повинна бути визначена категорія по вибухонебезпечній і пожежній безпеці відповідно до [12]. Відповідні позначення повинні бути нанесені на вхідні двері приміщення.

Споруди та їх приміщення, у яких розміщуються ЕОМ, повинні мати ступінь вогнестійкості не нижче II. Приміщення для обслуговування, ремонту і налагодження ЕОМ повинні відноситися:

* по пожежонебезпеки до категорії В – пожежонебезпечні приміщення, де розташовуються тверді пальні речовини (ТПР);
* по класу приміщення до категорії П ІІ.

Стіни приміщень з ЕОМ виготовляються з не пальних матеріалів. Підходи до засобів пожежогасіння повинні бути вільними.

5.1.2 Електронебезпечні чинники

При роботі з автономною частиною «Системи доступу до енциклопедичних знань на природній мові» на ЕОМ можуть трапитися нещасні випадки, зв'язані з ураженням електричним струмом, викликаним дотиком до оголених місць струмоведучих частин устаткування, або частин, що знаходяться під напругою.

Вплив струму на людину проявляється по-різному, в залежності від його величини:

* струм до 0,6 мА не відчувається людиною;
* струм завбільшки 6 мА призводить до скорочення м'язів тієї частини, тіла, що піддалася його впливу;
* струм 50 мА вражає органи дихання та серцево-судинну систему;
* при досягненні струму порога 100мА настає фібриляція серця і, потім, його зупинка;
* при впливі на тіло людини струму в 3-4 А виникає обвуглювання ділянок тіла.

За способом захисту людини від ураження електричним струмом відео термінали, ЕОМ, периферійні пристосування ЕОМ і оснащення для обслуговування, ремонту і налагодження ЕОМ повинні відповідати I класу захисту. Вимоги електричної і механічної безпеки і методи іспитів чи повинні бути заземлені.

Лінія електромережі для живлення ЕОМ, периферійних пристроїв ЕОМ і устаткування для обслуговування, ремонту і налагодження ЕОМ виконується як окрема групова, трьох провідна мережа, шляхом прокладки фазового, нульового робочого і нульового захисного провідників.

Підключення на розподільному щиті до одного контактного затиску нульового робочого і нульового захисного провідників заборонено.

Площа перетину нульового робочого і нульового захисного провідника в груповій трьох провідній мережі повинна бути не менш площі перетину фазового провідника.

У приміщенні, де одночасно експлуатується чи обслуговується більш п'яти персональних ЕОМ, на видному доступному місці встановлюється аварійний вимикач, за допомогою якого можливо зробити знеструмлення приміщення (за винятком освітлення).

5.2 Організаційні і технічні заходи щодо захисту користувачів від шкідливих і небезпечних факторів.

Для якісної і зручної роботи з автономною частиною «Системи доступу до енциклопедичних знань на природній мові», необхідне проведення проектних заходів: правильна облаштованість, належне дотримання ергономічних характеристик основних елементів робочого місця [14], санітарно-гігієнічних вимог і т.п.

Облаштованість робочого місця користувача при роботі з підсистемою повинне забезпечувати відповідність всіх елементів робочого місця і їхнього розташування.

Площа, яка виділена для одного робочого місця з ЕОМ, повинна складати не менш 6 м2 , а обсяг – не менш 20 м3.

Вимоги до організації робочого місця

При роботі з автономною частиною «Системи доступу до енциклопедичних знань на природній мові» необхідно виконувати вимоги до організації робочого місця, що наведені нижче.

До самостійної роботи допускаються особи, які пройшли медичний огляд, навчання за професією, вступний інструктаж з охорони праці на робочому місці.

Конструкція робочого місця робітника при роботі з ЕОМ (при роботі сидячи) повинна забезпечувати підтримку оптимальної робочої пози з наступними ергономічними характеристиками:

* стопи ніг – на підлозі чи на підставці для ніг;
* стегно – у горизонтальній площині;
* передпліччя – вертикально;
* лікті під кутом 70о-90о до вертикальної площини;
* зап'ястя – зігнуті під кутом не більш 20о щодо горизонтальної площини;
* нахил голови 15о-20о відносно вертикальної площини.

Дисплей та клавіатура мають розташовуватися на оптимальній відстані від очей користувача, але не ближче 600 мм, з урахуванням розміру алфавітно-цифрових знаків та символів.

Розташування екрану відео терміналу має забезпечувати зручність зорового спостереження у вертикальній під кутом ±30° від лінії зору користувача.

Клавіатуру слід розміщувати на поверхні столу або на спеціальній, робочій і такій, яка регулюється за висотою поверхні окремо від столу на відстані 100-300 мм від краю, ближчого до користувача. Кут нахилу клавіатури має бути в межах 5-15°.

5.3 Норми пожежної безпеки на робочому місці користувача

Вимоги безпеки праці перед початком роботи на ЕОМ

Перед початком роботи з автономною частиною «Системи доступу до енциклопедичних знань на природній мові» робітник інструктується первинним інструктажем. Первинний інструктаж завжди проводиться на робочому місці з безпосереднім показом робіт (стажування 1 місяць). Потім, через кожні 6 місяців проводиться повторний інструктаж. Результати інструктажу заносяться в "Журнал реєстрації інструктажів з питань охорони праці". У журналі після проходження інструктажу повинен бути підпис людини, яка інструктувала та робітника. Далі підготовка робочого місця повинна виконується відповідно до нижче приведених пунктів:

* увімкнути систему кондиціонування повітря в приміщенні;
* оглянути робоче місце і привести його в порядок: переконатися, що на ньому немає сторонніх предметів; все устаткування і блоки ЕОМ з'єднані із системним блоком за допомогою сполучних шнурів;
* перевірити надійність установки апаратури на робочому столі. Монітор повинен стояти не на краю столу. Повернути монітор так, щоб було зручно дивитися на екран – під прямим кутом (а не збоку) – та ледь зверху вниз, при цьому екран повинен бути ледь нахилений – нижній його край ближче до робітника;
* перевірити загальний стан апаратури, або перевірити чи справні електропроводка, сполучні шнури, штепсельні вилки, розетки, перевірити заземлення захисного екрана;
* відрегулювати освітлення робочого місця;
* відрегулювати і зафіксувати висоту крісла, зручний для робітника нахил його спинки;
* включити апаратуру комп'ютера перемикачами на корпусі в послідовності: стабілізатор напруги, монітор, системний блок;
* відрегулювати яскравість світіння екрана до яскравості навколишніх його поверхонь у робочій зоні – і не більше 3:1;
* при виявленні будь-яких несправностей роботу не починати, повідомити про це керівника робіт.

Вимоги безпеки праці під час роботи на ЕОМ

Під час роботи з автономною частиною «Системи доступу до енциклопедичних знань на природній мові» необхідно дотримуватися наступних правил:

* при виконанні роботи на комп’ютері слід сидіти прямо, не напружуватися;
* не дозволяються сторонні розмови, подразнюючі шуми;
* постійно слідкувати за функціонуванням систем опалення і кондиціювання;
* для зниження напруженості праці на комп’ютері необхідно рівномірно розподілити і чергувати характер робіт відповідно до їх складності.

Для зменшення негативного впливу на здоров'я працюючих виробничих факторів необхідно застосовувати регламентні перерви. У таблиці 5.1 приведений час регламентованих перерв робітника у залежності від категорії і групи робіт.

Таблиця 5.1 Час регламентованих перерв операторів (користувачів) в залежності від категорії і групи робіт

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Категорія  робіт | Група роботи | | | Час перерви при 8-годинній зміні, хв. |
| А, кількість  знаків | Б, кількість  знаків | В, година |
| Оптимальна-І | до 20 000 | до 15000 | до 2 | 20 |
| Припустима-ІІ | 21 000-40 000 | до 30 000 | до 4 | 40 |
| Важка-ІІІ | понад 40 000  не більш 60 000 | понад 30 000  не більш 45 000 | понад 4  не більш 6 | 60 |

Група А – читання інформації з попередні запитом (діалоговий режим роботи).

Група Б – введення інформації.

Група В – творча робота в режимі діалогу ЕОМ (налагодження програм, переклад і редагування текстів і т, д.).

При восьмигодинній робочій зміні регламентована перерва повинна бути такою:

* для I категорії робіт – через 2 години від початку зміни і через 2 години після обідньої перерви (кожен тривалістю 10 хв.);
* для II категорії робіт – через 2 години від початку зміни (тривалістю 15 хв.), через 1,5 і 2,5 годин після обідньої перерви (15 і 10 хв. відповідно чи тривалістю 5-10 хв. Через щогодини роботи, в залежності від характеру технологічного процесу);
* для III категорії робіт – через 2 години від початку зміни (тривалістю 15 хв.), через 1,5 і 2,5 годин після обідньої перерви (тривалістю 20 хв. Кожний тривалістю 5-15 хв. Щогодини роботи, у залежності від характеру технологічного процесу);
* під час нічної зміни, незалежно від групи і категорії робіт, тривалість регламентованих переривів збільшується на 60 хв.

Під час роботи забороняється:

* використовувати рідинні або аерозольні засоби для чищення поверхонь комп’ютера, крім передбачених для цього засобів (для чищення пластмасових поверхонь використовують чисту бавовняну ганчірку, зволожену мильним розчином; екрани та захисні екрани протирають ватою, змоченою спиртом);
* самостійно ремонтувати апаратуру;
* класти будь-які речі на вузли комп’ютера;
* закривати вентиляційні отвори апаратури, що може привести до її перегрівання і виходу з ладу.

Після закінчення роботи необхідно:

* вийти з програмної оболонки і повернутися в середовище операційної системи;
* вимкнути системний блок, принтер, інші периферійні пристрої, монітор;
* вимкнути стабілізатор живлення, якщо комп'ютер підключений до мережі через нього;
* штепсельні вилки витягнути з розетки;
* накрити клавіатуру кришкою для запобігання потрапляння на неї пилу;
* прибрати робоче місце.

5.4 Вимоги безпеки праці

Перед початком роботи з ПК користувач зобов'язаний: провітрити робоче приміщення; перевірити стійкість положення устаткування на робочому столі; перевірити відсутність видимих пошкоджень обладнання, дискет в дисководі системного блоку; перевірити справність і цілісність живлять і сполучних кабелів, роз'ємів і штепсельних з'єднань, захисного заземлення (занулення); перевірити справність.

Відрегулювати положення столу, стільця (крісла), підставки для ніг, клавіатури, екрана монітора; освітленість на робочому місці. При необхідності включити місцеве освітлення; протерти поверхню екрану монітора, захисного фільтра (за його наявності) сухою м'якою тканинною серветкою.

Переконатися у відсутності відображень на екрані монітора, зустрічного світлового потоку. Включити обладнання ПК в електричну мережу, дотримуючись наступну послідовність: стабілізатор напруги (якщо він використовується), блок безперебійного живлення, периферійні пристрої (принтер, монітор, сканер та інші пристрої), системний блок.

Забороняється приступати до роботи при: вираженому тремтінні зображення на моніторі; виявленні несправності обладнання; наявності пошкоджених кабелів або проводів, роз'ємів, штепсельних з'єднань; відсутності або несправності захисного заземлення (занулення) обладнання.

У період роботи за ПЕОМ необхідно передбачати перерви для відпочинку, які мають бути оптимальної тривалості (надто тривалі ведуть до порушення готовності до дії та розладу динамічного стереотипу). Тому, через кожні 40-45 хвилин необхідно робити перерву на 3-хвилини, а через 2 години – на 15 хвилин. Середня сумарна тривалість роботи за ПЕОМ на день не повинна перевищувати 4 годин, а на тиждень – 20 годин.

При використанні захисного екрана або монітора із зниженим випромінюванням час роботи за ПЕОМ може бути збільшено.

Шкідливою є одна й та сама поза протягом тривалого часу. Тому в положенні сидячі треба час від часу змінювати фіксовані робочі пози, робити короткочасні перерви.

Під час роботи на ПЕОМ напружуються м'язи рук. Для підтримки їх тонусу під час перерви рекомендується проводити гімнастичні вправи.

З метою зниження негативного впливу монотонної діяльності доцільно чергувати операції введення тексту і цифрових даних, редагування тексту.

Забороняється залишати ввімкнені ПЕОМ і пристрої без нагляду. Підключення і відключення роз'ємних кабелів пристроїв ПЕОМ робити тільки при відключенні їх з мережі.

Не можна користуватися біля ПЕОМ аерозолями (дезодорантами, тощо). Не допускати попадання води та інших рідин в середину пристроїв комп'ютера. При наявності електроструму на корпусі припинити роботу, вимкнути ПЕОМ від електромережі, сповістити керівника.

У випадку виникнення у каристувача зорового або дискомфорту інших неприємних суб'єктивних відчуттів, що настають, незважаючи на проходження санітарно-гігієнічних і ергономічних вимог, режимів роботи і відпочинку, варто застосувати індивідуальний підхід в обмеженні часу робіт і корекції тривалості перерв для відпочинку, або проводити заміну іншими видами робіт (не пов'язаних з використанням комп'ютера).

По закінченні роботи користувач зобов'язаний: коректно закрити всі активні завдання; вимкнути живлення системного блоку; вимкнути живлення усіх периферійних пристроїв; відключити блок безперебійного живлення; відключити кабель живлення від мережі; привести в порядок робоче місце; про несправність обладнання та інших зауважень по роботі з ПК повідомити безпосереднього керівника або особам, що здійснюють технічне обслуговування обладнання.

5.5 Дії при аварійних ситуаціях

Після закінчення роботи необхідно:

* при раптовому припиненні подавання електроенергії вимкнути послідовно периферійні пристрої, ВТП, процесор і витягнути штепсель вилки з розетки;
* при виявленні ознак горіння (дим, запах горіння), вимкнути апаратуру, знайти джерело займання і вжити заходів для його ліквідації, повідомити керівництво чи системного адміністратора;
* в разі виникнення пожежі негайно повідомити пожежну частину, вжити необхідних заходів для евакуації людей згідно з планом евакуації по підприємству і приступити до гасіння первинними засобами пожежогасіння.

5.5.1 Допомога при уражені електричним струмом

Перша медична допомога при ураженні людини електричним струмом повинна бути спрямована на звільнення її від дії струму та підтримку або відновлення серцевої діяльності і дихання.

Якщо потерпілий при свідомості, але не спроможний розтиснути руки, йому слід запропонувати підскочити або впасти, щоб розірвати електрострумовий ланцюг.

Коли уражений без свідомості, то потрібно негайно знеструмити його шляхом вимикання напруги в мережі рубильником, вимикачем або запобіжником. При неможливості достатньо швидко вимкнути електромережу слід перерізати чи перерубати електропроводи.

При ситуації, коли потерпілого можна відтягти від ушкоджуючи струмоведучих частин, до яких він торкається, рятівник повинен подбати про власну безпеку: по можливості діяти однією рукою, тягнути тільки за суху одежу, уникати доторкання до оголених частин тіла потерпілого та оточуючих металевих деталей. Електроізолюватися від потерпілого можна за допомогою діелектричних рукавиць або обмотаними тканиною руками, ставши на гумовий килимок, суху дошку або інший підручний предмет, що має діелектричні властивості. Відокремивши ураженого від струмопроводу, потрібно ретельно оглянути його.

Якщо в потерпілого наявні ознаки клінічної смерті, то слід невідкладно розпочати серцево-легеневу реанімацію. При "несправжній смерті" єдиним ефективним заходом першої допомоги є негайне проведення штучного дихання (з рота в рот); інколи його доводиться робити протягом кількох годин поспіль. Штучне дихання при працюючому серці, як правило, швидко покращує стан ураженого: шкіра стає нормального кольору, визначаються пульс і артеріальний тиск. Серцево-легеневу реанімацію або тільки штучне дихання по можливості потрібно поєднувати з підшкірним введенням 2 мл кордіаміну або 1 мл 10% розчину кофеїну.

Непритомну людину в подальшому слід утеплити, напоїти великою кількістю води, чаю чи компоту (давати алкоголь і каву шкідливо) та відправити на транспорті до лікарні. При необхідності реанімаційні заходи чи штучне дихання слід безперервно продовжувати і в транспорті.

Під час транспортування до лікарні потрібно особливо уважно стежити за станом потерпілого, тому що в будь-який час у нього може виникнути зупинка дихання і серцевої діяльності. Слід бути напоготові до реанімаційних заходів у дорозі.

# висновки

Під час виконання роботи, було досліджено та проаналізовано учбові процеси вищого навчального закладу, їх зв’язок та вплив на процес формування документів для складання розкладу занять університету. Були знайдені проблеми та недоліки цих процесів та запропоновані зміни, що допоможуть раціоналізувати ці процеси.

Впровадження запропонованих механізмів дозволить автоматично формувати документацію, що покращить якість розкладу занять університету та зменшить час, що витрачається зараз на ці процеси.

Для задач формування документації була розроблена автоматизована система. Для того, щоб система максимально відповідала поставленим вимогам, була зручною та функціональною, перед її проектуванням та розробкою були розглянуті та проаналізовані існуючі аналоги та літературні джерела. В розглянутих аналогах були виявлені недоліки та переваги. Деякі з рішень, що були знайдені в цих системах, були використанні під час проектування та розробки автоматизованої системи.

Розроблена автоматизована система буде взаємодіяти з системою складання розкладу занять. Ця особливість була врахована під час проектування системи. Проектування та розробка системи проводилась з використанням сучасних методологій та підходів до вирішення цих питань.

Проектування системи відбувалося з використанням об’єктно-орієнтованого підходу. Використання шаблонів проектування та окремі дизайнерські рішення розробника дозволили зробити систему гнучкою, що легко піддається модифікації, зрозумілою та надійною.

Використання сучасних гнучких методологій розробки дозволить і надалі розширювати функціональність системи або вносити зміни до її поточної функціональності.

Інтерфейс програми було спроектовано з урахуванням потреб цільового користувача. Було досягнуто основної мети зробити інтерфейс зручним та зрозумілим. Інтерфейс мав бути спроектований так, щоб будь які дії займали у користувача мінімальну кількість часу, адже планується велика інтенсивність використання системи.

Формат вихідної документації був складений таким чином, щоб він співпадав з форматом, що на даний час використовується учбовим відділом.

Велику увагу було приділено тестуванню та відлагодженню системи. Використання сучасних інструментів тестування та налагодження та використання різних підходів та методів тестування, що найкраще підходять до тестування кожного окремого методу дозволили зробити систему надійною та стійкою до можливих помилок.

Для безпечної роботи з автоматизованою системою, були дослідженні шкідливі та небезпечні фактори, що можуть виникнути під час її функціонування. Були розроблена система вимог, прийомів та інструкцій, що дозволить зменшити ці фактори та зробити роботу з системою максимально безпечною.

Після закінчення розробки системи, була написана уся необхідна програмна документація. Наявність якісного керівництва користувача дозволить методистам учбового відділу, що будуть використовувати систему, швидко вивчити можливості програми та почати її використовувати. Задокументований опис та текст програми спростить подальше супроводження системи.

# ЛІТЕРАТУРА

1. І. М. Вишнякова, С. Ю. Разумов. АРМ «Розклад занять». //Міжнародна науково-практична конференція «Сучасні інформаційні технології на транспорті. В промисловості та освіті». Тези доповідей, м. Дніпропетровськ, 2007. с.90.

2. Автоматизированная система управления вузом Галактика [Електронний ресурс] : <http://vuz.galaktika.ru/>

3. ММИС Лаборатория. Руководство пользователей – [Електронний ресурс] – Режим доступу: <http://www.mmis.ru/Default.aspx?tabid=172>

4. Гриценко В. Г. Інформаційна технологія управління навчальним навантаженням у вищих навчальних закладах [Текст] /В. Г. Гриценко // Збірник наукових праць «Інформаційні технології в освіті» – 2010. – № 8. – С. 61-68.

5. Болюбаш Я.Я. Організація навчального процесу у вищих закладах освіти: Навч. посібник для слухачів закладів підвищення кваліфікації системи вищої освіти / Я.Я.Болюбаш. – К.: ВВП «КОМПАС», 1997. – 64с.

6. Співаковський О.В. Особливості автоматизованих систем управління вищими навчальними закладами. //Вісник Харківського національного університету. № 629. Серія «Математичне моделювання. Інформаційні технології. Автоматизовані системи управління». – Випуск 3. Видавничий центр ХНУ. Айлант – 2004. – С.86-99.

7. Львов М.С., Співаковський О.В., Щедролосьєв Д.Є. Інформаційна система управління вищим навчальним закладом як платформа реалізації управління академічним процесом. //Комп'ютер у школі та сім'ї, №2,3,4, К. – 2007 р.

8. Поліщук В., Тесля Ю., Триус Ю., Левківський К. Дослідження нормативів та управління розрахунками навчального навантаження у вищому закладі освіти // Вища школа. – 2006. – №1. – С.35-52.

9. Крёнке Д. Теория и практика построения баз данных. 8-е изд. / Дэвид Крёнке. – СПб.: Питер, 2003. – 800 с.

10. Нейгел, К. C# 4.0 і платформа .NET Framework 4 для професіоналів: Пер. с англ. [Текст] / К. Нейгел, Б. Івьєн, Д. Глінн, К. Уотсон. – М.: ТОВ "В.Д. Вильямс", 2011. – 1440 с.: іл.

11. Ріхтер, Дж. Програмування на платформі Microsoft .NET Framework, 2-е видання [Текст] / Дж. Рихтер. – М.: Русская редакция, 2003. – 796 с.

12. Ріхтер, Дж. CLRviaC#. Програмування на платформі Microsoft .NET Framework 3.0 на мові C# [Текст] / Дж. Рихтер. – СПб.: Пітер, 2007. – 656 с.

13. Троелсен, Е. C# 2008 и платформа .NET 3.5 [Текст] / Е. Троелсен. – СПб.: Пітер, 2008. – 796 с.

14. Буч Г. Объектно-ориентированное проектирование с примерами / Г. Буч. – М.: Конкорд, 1992. – 519 с.

15. Ларман К. Применение UML и шаблонов проектирования. Введение в объектно-ориентированный анализ и унифицированный процесс UP / Крэг Ларман. – М.: Вильямс, 2004. – 624 с.

16. Макгрегор Дж. Тестирование объектно-ориентированного программного обеспечения / Джон Макгрегор, Девид Сайкс. – М.: ДиаСофт, 2002. – 417 с.

17. Закон України «Про охорону праці», нова редакція, м. Київ, від 21.11.2002 року, №229-IV.

18. ДСанПіН 5.5.6.009-98 «Влаштування і обладнання кабінетів комп'ютерної техніки в навчальних закладах та режим праці учнів на персональних комп'ютерах».

19. ДСТУ 2293-99 «Охорона праці. Терміни та визначення основних понять».

20. ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень».

21. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПіН 3.3.2.007-98.

22. ДСанПіН 3.3.6.096-2002 «Державні санітарні правила і норми при роботі з джерелами електромагнітних полів».

23. НПАОП 0.00-1.31-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин».

24. ДСН 3.3.6.037-99 «Санітарні норми виробничого шуму, ультразвуку та інфразвуку».

25. НАПБ В.01.050-98/920 «Правила пожежної безпеки для закладів, установ і організацій системи освіти України».