

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ
ІМЕНІ АДМІРАЛА МАКАРОВА**

Навчально-науковий інститут комп'ютерних наук та управління проектами
(повне найменування інституту, назва факультету)

Інформаційних управляючих систем та технологій
(повна назва кафедри)

"Допущений до захисту"
Завідувач кафедри ІУСТ

к.т.н, доц. Михелєв І.Л.
" ____ " ____ 2022 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти "бакалавр"
(освітньо-кваліфікаційний рівень)

на тему **Розробка інформаційної підсистеми "Розклад студента
НУК" інформаційної системи кафедри ІУСТ НУК**

Виконав: студент 4 курсу, групи 4142
напряму підготовки (спеціальності)
122 – "Комп'ютерні науки"
(шифр і назва напряму підготовки, спеціальності)

Агарін М.А.
(прізвище та ініціали)

Керівник к.т.н., доцент Михелєв І.Л.
(прізвище та ініціали)

Рецензент Беркунський Є.Ю.
(прізвище та ініціали)

м. Миколаїв – 2022 року

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Інститут, факультет Навчально-науковий інститут комп'ютерних наук та управління проектами

Кафедра Інформаційних управляючих систем та технологій

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 – "Комп'ютерні науки"

"ЗАТВЕРДЖУЮ"

Гарант освітньої програми

к.т.н, доц. Гайда А.Ю.

" ____ " _____ 2022 р.

ЗАВДАННЯ НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТУ

Агарін Микита Андрійович

1. Тема проекту (роботи) Розробка інформаційної підсистеми "Розклад студента НУК" інформаційної системи кафедри ІУСТ НУК

керівник проекту (роботи) Михелєв Ігор Леонідович,

затверджені наказом ректора № 269-уч. від "18" травня 2022 року.

2. Строк подання студентом проекту (роботи) 20.06.2020 року.

3. Вихідні дані до проекту (роботи) ДСТУ щодо обробки інформації, літературні джерела, технічна документація на існуючі аналоги систем, матеріали практики.

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

- Титульний аркуш, завдання на дипломну роботу, анотація (українською, російською, англійською), зміст, перелік умовних означень, символів, одиниць та термінів (при необхідності).

- Вступ.
- Аналіз предметної області і постановка задачі.
- Розробка проекту автоматизованої системи.
- Реалізація проекту автоматизованої системи (технічна документація по проекту).

- Розділ з охорони праці.

- Висновки.

- Список використаної літератури.

- Додатки: загальний опис підсистеми, керівництво користувача, керівництво адміністратора, програма і методика випробувань, графічні інтерфейси, код програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень).

- 5.1 Схема роботи інформаційної підсистеми.
- 5.2 Діаграма варіантів використання.
- 5.3 Логічна модель БД.
- 5.4 Діаграма діяльності.
- 5.5 Діаграма класів.
- 5.6 Схема розміщення технічних засобів.
- 5.7 Ескізи інтерфейсу програмного забезпечення

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1.	Михелєв І.Л.	10.03.22	31.04.22
2.	Михелєв І.Л.	10.03.22	29.05.22
3.	Михелєв І.Л.	10.03.22	06.06.22
4.	Михелєв І.Л.	10.03.22	13.06.22

7. Дата видачі завдання 10.03.2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1.	Вивчення і опис предметної області	31.04.22	
2.	Розробка концепції ІС	31.04.22	
3.	Розробка проектних рішень по ІС	29.05.22	
4.	Реалізація проекту, розробка робочої документації	06.06.22	
5.	Вирішення питань з охорони праці	13.06.22	
6.	Виконання графічної частини роботи	15.06.22	
7.	Оформлення пояснювальної записки	15.06.22	
8.	Подання роботи на кафедральний захист	17.06.20	
9.	Подання роботи рецензенту	17.06.20	
10.	Подання роботи на державний захист	20.06.20	

Студент _____ Агарін М.А.

Керівник проекту (роботи) _____ к.т.н., доцент Михелєв І.Л.

Анотація

Пояснювальна записка містить 98 сторінок, на яких розміщені 16 рисунків, 7 таблиць, 6 додатків, 7 джерел літератури.

Метою даного дипломного проектування є розробка автоматизованої системи доступу студентів кафедри ІУСТ до розкладу занять з будь якого місця де є підключення до мережі Інтернет.

Пояснювальна записка містить теоретичне обґрунтування існуючої проблеми, опис предметної області, зокрема схему інформаційних потоків, що підлягають автоматизації, а також всіх стадій розробки інформаційної підсистеми. Крім того, вона містить опис архітектури програмного засобу, функціонального призначення, проектування і розробки бази даних, алгоритмів програмного засобу і тестування.

Впровадження запропонованої в дипломному проекті, інформаційної підсистеми дозволить спростити доступ до розкладу, знизити час на його отримання, а також своєчасно повідомити про внесені зміни студентів, тим самим зменшити кількість можливих запізнень студентів на заняття. Таким чином всі ці параметри ведуть до збільшення продуктивності начального процесу в цілому.

Abstract

The explanatory note contains 98 pages, which contain 16 figures, 7 tables, 6 annexes, 7 sources of literature.

The purpose of this diploma design is to develop an automated access system for the students of the IUST department to the schedule of classes from any place where there is an Internet connection.

The explanatory note contains the theoretical substantiation of the existing problem, the description of the subject area, in particular the scheme of information flows to be automated, as well as all stages of the development of the information subsystem. In addition, it contains a description of the software architecture, functionality, design and development of the database, software algorithms and testing.

The introduction of the information subsystem offered in the diploma project will simplify access to the schedule, reduce the time for its receipt, and also promptly report students about changes, thus reducing the number of possible late delays for students, in turn, all these parameters lead to an increase in the productivity of the initial the whole process.

ЗМІСТ

ВСТУП.....	8
Розділ 1. Розробка концепції автоматизованої підсистеми "Розклад студента НУК"	10
1.1 Аналіз предметної області і формування вимог до проекту.....	10
1.2 Опис концепції системи.....	12
Розділ 2. Розробка проектних рішень по системі.....	14
2.1 Загальносистемні проектні рішення.....	14
2.1.1 Організаційна і функціональна структура.....	14
2.1.2 Специфікація варіантів використання.....	16
2.1.3 Опис функцій, що автоматизуються.....	19
2.1.4 Опис постановки задачі.....	19
2.1.5 Проектна оцінка надійності системи.....	20
2.1.6 Загальний опис системи.....	22
2.1.7 Захист від несанкціонованого доступу.....	23
2.2. Рішення по інформаційному забезпеченню.....	23
2.2.1 Вибір СУБД.....	23
2.2.2 Вибір мови програмування.....	28
2.2.3 Організація інформаційної бази.....	30
2.2.4 Опис інформаційної бази.....	33
2.3 Опис алгоритмів функціонування системи.....	36
2.4 Рішення по програмному забезпеченню.....	39
2.5 Рішення по технічному забезпеченню.....	41
2.5.1 Схема автоматизації.....	41
2.5.2. Структурна схема комплексу обчислювальних засобів.....	42
Розділ 3. Розробка робочої документації.....	44
Розділ 4. Охорона праці у виробничих приміщеннях з ЕОМ.....	45
4.1 Загальні принципи охорони праці.....	45
4.2 Аналіз небезпечних і шкідливих факторів при роботі з ПК у відділі програмного забезпечення.....	46

4.2.1 Вплив електромагнітного та електростатичного випромінювань.....	47
4.2.2 Вплив недостатньої освітленості робочого місця.....	47
4.2.3 Вплив електричного струму.....	48
4.2.4 Вплив шуму.....	48
4.2.5 Вплив несприятливих умов мікроклімату.....	49
4.2.6 Вплив нераціонального розташування устаткування і неправильної організації робочого місця.....	49
4.3 Розрахунок освітленості робочого місця.....	50
4.4 Розробка заходів щодо зменшення впливу небезпечних і шкідливих факторів.....	51
Висновки.....	53
Список літератури.....	54
Додатки.....	55

ВСТУП

Практично щодня і незалежно від курсу студенти нашого університету стикаються з необхідністю дізнаватися розклад занять. На даний момент у НУК імені адмірала Макарова зробити це можна лише відвідавши університет, що вкрай не зручно для більшості учнів. Під час навчального процесу, наприклад перед заняттями, студентам доводиться йти до стенду з розкладом, щоб дізнатися в якій аудиторії воно буде проходити. Такий підхід забирає дорогоцінний час перерв. Також часто перед початком кожного семестру студентам доводиться добиратися до університету і переписувати або фотографувати розклад, що також не зручно. На додаток до всього вищесказаного варто додати, що час від часу в розклад вносяться правки про яких студентів не завжди вчасно сповіщають.

Автоматизація отримання студентами розкладу вирішить відразу кілька проблем: дозволить витратити набагато менше часу на отримання необхідної інформації безпосередньо в університеті, виключить запізнення пов'язані з пошуком потрібної аудиторії, дасть можливість отримувати доступ до розкладу з будь-якого місця де є з'єднання з Інтернетом, а так само повідомить учнів про зміни в ньому. Крім того буде корисною функція розсилки новин усім користувачам даної підсистеми.

Метою дипломного проектування є розробка концепції і її реалізації для створення програмного забезпечення автоматизації доступу до розкладу, що буде володіти такими можливостями:

- Пошук розкладу по групах;
- Завантаження і збереження розкладу на смартфон під управлінням ОС Android;
- Розсилка повідомлень у разі зміни розкладу;
- Додавання, видалення та редагування розкладу на автоматизованому робочому місці;
- Розсилка новин.

Для успішної реалізації проекту необхідно вирішити наступний ряд задач:

- Провести аналіз предметної області для докладного вивчення проблеми і вибору оптимального шляху її вирішення;
- На основі проведеного аналізу визначити вимоги до програмного продукту;
- Виконати аналіз можливих програмних засобів. Визначити мову програмування програми і вибрати середовище виконання програмного коду;
- Скласти робочий і технічні проекти з усіма складовими;
- Скласти документацію;
- Зробити висновки про виконану роботу.

Розробка такого роду підсистеми вимагає врахування специфіки освітнього закладу. Для цього необхідно виявити об'єкт дослідження даного дипломного проекту, ним є інформаційна система кафедри ІУСТ Національного університету імені адмірала Макарова.

Предметом проектування є інформаційна підсистема кафедри ІУСТ для автоматизації доступу до розкладу.

В якості засобів проектування і розробки використовувалися такі програмні продукти: Astah Community, MySQL, Microsoft Visual Studio Community.

Результатом роботи є розроблена автоматизована підсистема «Розклад студента НУК», що представляє швидкий, структурований і зручний доступ студентам до розкладу з докладним відображенням її в табличному вигляді. За допомогою даного ПЗ студенти можуть дізнатися детальну інформацію про кожне заняття. Наприклад таку як: аудиторія, викладач, час початку і кінця. Крім того присутня функція розсилки повідомлень про зміни в розкладі та функція розсилки новин університету.

Розділ 1.

Розробка концепції автоматизованої підсистеми "Розклад студента НУК"

1.1 Аналіз предметної області і формування вимог до проекту

ВНЗ – це динамічна інформаційна система, що об'єднує елементи різної природи, досить складно пов'язані між собою. Одним з таких елементів є розклад, робота над яким ведеться протягом усього навчального року, безперервно, повсякденно.

В процесі навчання у студентів систематично виникає необхідність переглядати свій розклад занять, проте в даний момент зробити це не завжди зручно. Причина незручностей полягає в організації представлення розкладу студентам, що існує на даний момент. Його не можливо переглянути приміром з дому, а добиратися до університету лише заради його перегляду не представляється раціональним і зручним. Так само бувають ситуації коли студенти забувають номери аудиторій, в через що змушені йти до стенду з розкладом, що зазвичай призводить до запізнення на заняття. Крім того, час від часу розклад занять або аудиторії для їх проведення змінюються, а це як правило заплує і так само призводить до незручностей і запізнь. Кожна нова ситуація вимагає прийняття негайних заходів по ліквідації зриву занять або загрози такого зриву. При цьому безперебійність і впорядкованість навчального процесу безпосередньо залежить від оперативного донесення змін у розкладі до учнів.

Дана організаційна проблема зачіпає безліч навчальних закладів завдяки чому вже існують програмні продукти, що вирішують дану задачу. Варто відзначити найбільш популярні рішення, а саме інформаційні системи "КПІ Розклад" та "Універ – розклад ВНЗ".

Інформаційна система "КПІ Розклад" являє собою додаток для смартфонів працює на операційній системі Android. Робота його полягає в підключенні до веб-серверу університету КПІ, зчитування розкладу і подальшого завантаження його в пам'ять пристрою, для подальшого автономного доступу.

Головні недоліки представленої системи:

- Сайт періодично стає недоступний, в слідстві чого додаток стає марним, якщо розклад не було збережено на самому пристрої;
- Оновлення розкладу на сайті здійснюється тільки раз в семестр, що в наслідок змін в ньому, не дозволить додатку надавати студентам актуальну інформацію про заняття.

Друга система позиціонується як універсальний додаток для всіх вищих навчальних закладів і ґрунтується на внесення розкладу самими студентами шляхом звичайного заповнення таблиць.

Це рішення так само має свої мінуси, основні з яких перераховані нижче:

- незручність для користувача при заповненні розкладу допомогою текстового набору;
- можливість зробити помилки при заповненні;
- відсутній своєчасний оновлення розкладу з урахуванням правок.

Вирішити існуючий ряд проблем пов'язаних з організацією розкладу планується шляхом розробки інформаційної системи "Розклад студента НУК". Це дозволить кожному студенту кафедри ІУСТ швидко та зручно отримати доступ до розкладу.

На прикладі вже існуючих додатків буде розроблена власна інформаційна підсистема, при цьому будуть враховані недоліки аналогів для створення якісного програмного продукту.

Вимоги до складу виконуваних функцій

Програма повинна забезпечувати можливість виконання перерахованих нижче можливостей:

1. Розклад повинен розподілятися за двома категоріями – тиждень над рискою і під рисою;
2. Програма повинна надавати доступ до повної та структурованої інформації про заняття в попередньо сформованій базі;
3. Користувач повинен мати можливість переглядати розклад навіть у разі відключення від мережі Інтернет, за умови попереднього збереження розкладу в

пам'ять пристрою;

4. Адміністратор повинен мати можливість редагування, додавання і видалення розкладу окремих груп, а також перегляду розкладу в базі даних;

5. Інтерфейс програми повинен бути інтуїтивно зрозумілим і зручним;

6. Можливість входу на серверний додаток і управління базою даних повинна бути тільки у адміністратора, при цьому доступ повинен надаватися виключно після авторизації.

Вимоги до складу і параметрів технічних засобів

Для роботи серверної частини необхідний ПК який має характеристики не гірше ніж:

- процесор Intel Celeron G1840 2.8 ГГц;
- об'єм ОЗП 1 Гб;
- простір на диску: 50 Мб
- графічний адаптер GeForce 210 512 Мб;
- монітор SVGA 0.26, 15 дюймів;
- мережевий адаптер.

Для работы клиентской части ПО необходимо наличие у пользователя Android устройства имеющее характеристики не хуже чем:

- процесор ARM v7 1000 МГц;
- об'єм ОЗП 256 Мб;
- простір на диску: 50 Мб
- графічний адаптер Adreno 220 / Mali-400 MP;
- дисплей QVGA, 3.5 дюймів;
- мережевий адаптер

1.2 Опис концепції системи

Інформаційна підсистема «Розклад студента НУК» повинна складатися з двох частин-серверної і клієнтської. Серверна частина дозволить вносити, редагувати і видаляти бази даних розклад кожної групи, а так само буде обробляти запити від клієнтської частини інформаційної системи. Клієнтська

частина являє собою вільно розповсюджується серед студентів мобільний додаток надає швидкий доступ до розкладу. Робота системи буде організовано наступним чином: відразу після запуску користувачем клієнтського додатка з'являється вітальне вікно, де потрібно ввести номер групи. Потім після введення йде запит до сервера, на якому здійснюється пошук розкладу по групі в базі даних і в разі якщо результат пошуку позитивний, дані з розкладом відправляються на пристрій. Надійшло розклад виводиться на екран в структурованому та зручному вигляді.

Схема роботи інформаційної підсистеми представлена на рисунку 1.



Рисунок 1 – Схема роботи підсистеми "Розклад студента НУК"

Розділ 2.

Розробка проектних рішень по системі.

2.1. Загальносистемні проектні рішення

2.1.1. Організаційна і функціональна структура

У процесі функціонування підсистема взаємодіє з користувачами різних категорій, в залежності від яких різняться набір можливостей її використання. Схему цієї взаємодії представляє діаграма прецедентів або варіантів використання (Use Case Diagram), що призначена для попереднього визначення найбільш важливих сутностей системи на рівні спілкування з користувачами. Діаграма прецедентів зазвичай не прив'язується до конкретних аспектів реалізації системи. Вона активно застосовується, насамперед, для формалізації вимог до системи. Розробка програмного забезпечення з нуля зазвичай завжди починається з побудови діаграм прецедентів.

Прецедент (Use Case) – це опис послідовності виконуваних системою дій, які виробляють результат, вагомий для певного актора (Actor).

Актор представляє собою зв'язну множину ролей, які користувачі прецедентів виконують під час взаємодії з ними. Зазвичай актор представляє роль, яку в цій системі грає людина, апаратний пристрій або навіть інша система. У розроблювальній системі "Розклад студента НУК" акторами є адміністратор і студент.

В даній системі виділяємо прецеденти: авторизація, редагування даних, перегляд статистики, розсилка новин, пошук розкладу, перегляд розкладу, перегляд новин. Редагування даних в свою чергу включає прецеденти: видалення, додавання, зміна, а розсилка новин прецеденти: створення, публікація. Пошук розкладу розширюється прецедентом: пошук за номером групи.

Елементи діаграми прецедентів (прецеденти і актори) пов'язані відносинами. В даній діаграмі використані відносини асоціації, включення і розширення.

Схема цієї взаємодії, побудована в діаграмі варіантів використання підсистеми (Use Case) і показана на рисунку 2.

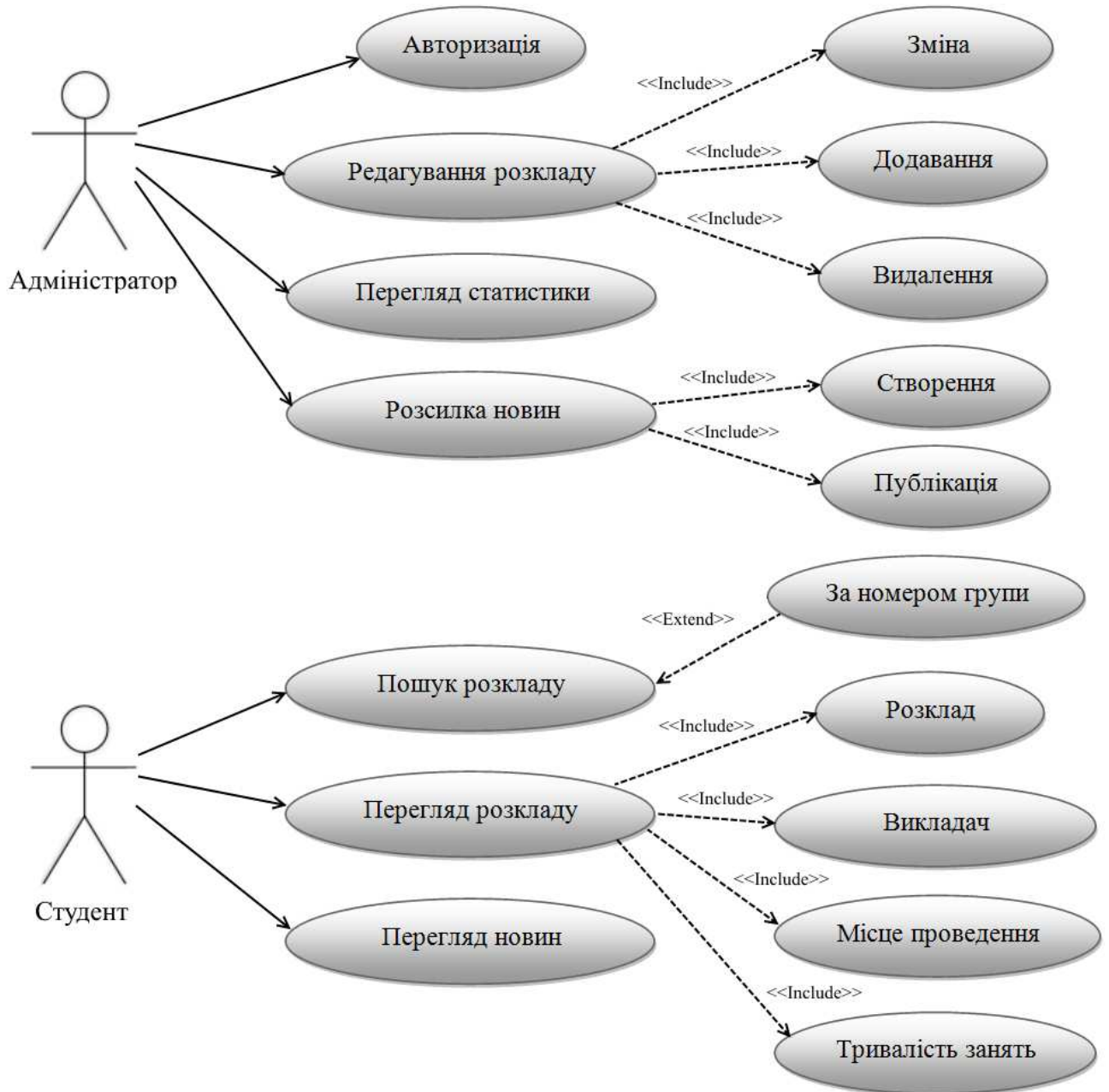


Рисунок 2 – Діаграма варіантів використання інформаційної підсистеми "Розклад студента НУК".

У ході взаємодії з підсистемою, користувачі використовують наступні функції:

Адміністратор – авторизація для отримання доступу до керування підсистемою. Редагування розкладу (зміна, додавання, видалення), перегляд статистики користування підсистемою. Процес розсилки новин, який

складається зі створення новини (оформлення) та її публікації на пристроях студентів.

Студент – пошук та перегляд розкладу, що включає перегляд розкладу самих занять, перегляд викладачів (які проводять заняття), місця проведення занять та їх тривалість.

2.1.2. Специфікація варіантів використання

Варіант використання: "Авторизація"

1. Короткий опис: варіант використання ініціюється суб'єктом "Адміністратор" і пропонує можливості входу в систему.

2. Передумови: повинна бути відкрита панель авторизації.

3. Потоки подій

а. Основний потік: функції варіанту використання починають працювати після відкриття в програмі панелі авторизації. На екрані з'явиться вікно, в якому будуть представлені поля для введення даних адміністратора, такі як логін і пароль. При успішному вході адміністратор отримує доступ до всіх керуючих функцій підсистеми.

б. Альтернативні потоки: Якщо у введенних даних була помилка, з'явиться вікно помилки і буде запропоновано ввести правильний логін і пароль.

4. Постумови: можливе повне використання всіх керуючих функцій підсистеми.

Варіант використання: "Редагування даних"

1. Короткий опис: варіант використання ініціюється суб'єктом "Адміністратор" і пропонує можливості редагування даних.

2. Передумови: успішно виконаний потік варіанту використання "Авторизація".

3. Потоки подій

а. Основний потік: функції варіанту використання починають працювати після вибору адміністратором пункту "Редагування розкладу". На екрані з'явиться панель редагування, в якій буде представлена таблиця розкладу.

Адміністратор може переглянути дані і якщо необхідно, внести зміни. Якщо дані внесені коректно, то при спробі збереження база даних буде оновлена.

б. Альтернативні потоки: Якщо при редагуванні буде введена не повна або не коректна інформація, то буде виведене повідомлення про помилку.

4. Постумови: оновлена база даних розкладу.

Варіант використання: "Перегляд статистики"

1. Короткий опис: варіант використання ініціюється суб'єктом "Адміністратор" і пропонує можливості перегляду статистики використання підсистеми студентами.

2. Передумови: успішно виконаний потік варіанту використання "Авторизація".

3. Потоки подій

а. Основний потік: функції варіанту використання починають працювати після відкриття адміністратором пункту статистики. На екрані з'явиться вікно, в якому будуть представлені структуровані дані про використання підсистеми.

б. Альтернативні потоки: альтернативні потоки не визначені.

4. Постумови: постумови не визначені.

Варіант використання: "Розсилка новин"

1. Короткий опис: варіант використання ініціюється суб'єктом "Адміністратор" і пропонує можливість розсилки новин.

2. Передумови: успішно виконаний потік варіанту використання "Авторизація".

3. Потоки подій

а. Основний потік: функції варіанту використання починають працювати після вибору адміністратором пункту "Розсилка новин". На екрані з'явиться панель, в якій буде представлена форма для вводу новин. Після заповнення форми стає доступна публікація новини і після підтвердження вона виконується.

б. Альтернативні потоки: Якщо при редагуванні буде введена не повна інформація, то буде виведене повідомлення про помилку.

4. Постумови: новина опублікована.

1. Варіант використання: "Пошук розкладу"

2. Короткий опис: варіант використання ініціюється суб'єктом "Студент" і пропонує можливості пошуку розкладу за номером групи.

3. Передумови: повинно бути встановлено з'єднання з Інтернетом та відкрита панель пошуку.

4. Потоки подій

а. Основний потік: функції варіанту використання починають працювати після вибору студентом пункту "Пошук розкладу". На екрані з'явиться панель пошуку, в якій буде представлено поле для введення пошукового запиту, яким повинен бути номер групи студента. У разі успішного пошуку буде завантажено і виведено на екран розклад студента.

б. Альтернативні потоки: Якщо в пошуковому запиті номер групи вказаний невірно, то буде виведене повідомлення про помилку.

5. Постумови: розклад збережений на пристрої, завдяки чому можливий його перегляд оффлайн.

Варіант використання: "Перегляд розкладу"

1. Короткий опис: варіант використання ініціюється суб'єктом "Студент" і пропонує можливості детального перегляду розкладу.

2. Передумови: повинна бути відкрита панель розкладу.

3. Потоки подій

а. Основний потік: функції варіанту використання починають працювати після вибору студентом пункту "Перегляд розкладу". На екрані з'явиться меню, що дозволить виконати детальний перегляд розкладу по таким пунктам як список занять, номер аудиторії, ім'я викладача, час проведення.

б. Альтернативні потоки: альтернативні потоки не визначені.

4. Постумови: постумови не визначені.

Варіант використання: "Перегляд новин"

1. Короткий опис: варіант використання ініціюється суб'єктом "Студент" і пропонує можливість перегляду новин.

2. Передумови: повинна бути відкрита панель новин.

3. Потоки подій

a. Основний потік: функції варіанту використання починають працювати після вибору студентом пункту "Перегляд новин". На екрані з'явиться стрічка новин, що дозволить виконати їх перегляд

b. Альтернативні потоки: альтернативні потоки не визначені.

4. Постумови: постумови не визначені.

2.1.3. Опис функцій, що автоматизуються

У ході розробки підсистеми будуть автоматизовані наступні функції:

1. Введення і контроль розкладу:

- початкове введення семестрового розкладу кожної групи кафедри ІУСТ;
- корегування розкладу та своєчасне повідомлення про зміни учнів;
- введення планового часу початку і закінчення занять;
- введення імен викладачів кожної дисципліни;
- введення даних про місце проведення занять.

2. Формування статистики роботи підсистеми:

- формування даних про кількість запитів до сервера.
- формування даних про кількість студентів, що прочитали новини.

3. Розсилка новин

- оформлення новини;
- публікація;
- повідомлення користувачів про надходження нових новин.

2.1.4. Опис постановки задачі

На підставі аналізу вимог і обмежень, а також попередніх концепцій реалізації системи, сформулюємо опис постановки задачі. Необхідно розробити підсистему для віддаленого доступу до розкладу, що буде задовольняти

перерахованим вище функціональним вимогам. Необхідно також забезпечити надійність збереження і безпеки доступу до інформації системи. Система повинна обслуговувати одночасну безперебійну роботу 100 користувачів, причому зміни, внесені адміністратором повинні бути доступні всім іншим користувачам негайно.

Програмне забезпечення повинне реалізовувати графічний інтуїтивно зрозумілий інтерфейс користувача.

Апаратне забезпечення повинне дозволяти підсистемі максимально продуктивно виконувати поставлені задачі.

2.1.5. Проектна оцінка надійності системи

Для оцінки надійності системи використовується принцип визначення показників надійності за характеристиками надійності комплектуючих елементів, що дозволяє вести розрахунок в процесі проектування системи, виходячи з надійності елементів та вузлів.

Виходячи з важливості серверної станції для функціонування підсистеми загалом, доцільно визначити надійність саме серверної станції. Можна виділити наступні компоненти серверної станції, вихід з ладу яких є найбільш ймовірним:

- блок живлення;
- джерело безперебійного живлення;
- комутатор мережі.

Потрібно розрахувати ймовірність безвідмовної роботи кожного з перелічених вище компонентів, а також надійність підсистеми в цілому.

У розрахунку використовуються дані, унаведені в технічній документації кожного з компонентів групи елементів.

Середній час роботи блоку живлення ПК: $T_{\text{бж.ср.}} = 10000$ годин.

Середній час роботи джерела безперебійного живлення: $T_{\text{дбж.ср.}} = 12000$ годин.

Середній час роботи комутатора: $T_{\text{к.ср.}} = 15000$ годин.

Розрахунок проводимо для визначення ймовірності безвідмовної роботи кожного з перерахованих елементів протягом 1000 годин за формулою:

$$P(t) = \exp\left(-\int_0^t \lambda(t) dt\right) = \exp(-\lambda_0 t),$$

де t – час, протягом якого розраховується надійність;

λ_0 – інтенсивність відмов елемента інформаційної системи протягом середнього часу безвідмовної роботи системи.

Частота відмов елемента інформаційної системи за середній час безвідмовної роботи $T_{\text{ср.}}$ розраховується по формулі:

$$\lambda_0 = 1/T_{\text{ср.}}.$$

Частота відмов для блоку живлення:

$$\lambda_0 = 1/10000 = 0,00001; P_{\text{бж}}(t) = \exp(-0,00001 \cdot 1000) = 0,920.$$

Частота відмов для джерела безперебійного живлення:

$$\lambda_0 = 1/12000 = 0,000083; P_{\text{дж}}(t) = \exp(-0,000083 \cdot 1000) = 0,943.$$

Частота відмов для концентратора:

$$\lambda_0 = 1/15000 = 0,000066; P_{\text{к}}(t) = \exp(-0,000066 \cdot 1000) = 0,961.$$

Під час визначення надійності підсистеми через відомі показники надійності її елементів вводяться два припущення: відмови елементів статично незалежні, відмова будь-якого елемента призводить до відмови підсистеми. Прийняті припущення дозволяють використовувати теорему добутку ймовірностей, яка дозволяє провести обчислення за формулою

$$P_{\text{іс}}(t) = \prod_{i=1}^n P_i(t),$$

де $P_i(t)$ – ймовірність безвідмовної роботи i -го елемента;

n – кількість елементів системи.

Надійність інформаційної системи протягом 1000 годин безперервної роботи визначається добутком ймовірностей безвідмовної роботи блоку живлення, джерела безперебійного живлення та комутатора.

$$P_{IC}(t) = P_{дж}(t) \cdot P_{дж}(t) \cdot P_K(t) = 0,920 \cdot 0,943 \cdot 0,961 = 0,834.$$

Виходячи з проведених розрахунків можна зробити висновок, що ймовірність безвідмовної роботи автоматизованої інформаційної системи створення описів технологічних процесів протягом 1000 годин становить близько 83%.

2.1.6. Загальний опис системи

Інформаційна підсистема представляє собою інструмент віддаленого доступу до розкладу, побудований на основі інформаційної системи кафедри ІУСТ. Структура підсистеми побудована за принципом клієнт-сервер. Серверна частина призначена для обробки запитів від клієнтів, а клієнтська для посилення запиту і обробки його результату. Клієнтська частина являє собою мобільний додаток для смартфонів з ОС Android, який вільно розповсюджується серед студентів.

Робота системи буде організовано наступним чином: відразу після першого запуску користувачем клієнтського додатка з'являється вікно, де потрібно ввести номер групи. Після введення йде запит до сервера, на якому здійснюється пошук розкладу в базі даних за номером групи і в разі якщо результат пошуку позитивний, дані з розкладом відправляються на пристрій. Отриманий розклад виводиться у відповідну таблицю в структурованому та зручному вигляді.

Схема роботи підсистеми представлена на рисунку 3.

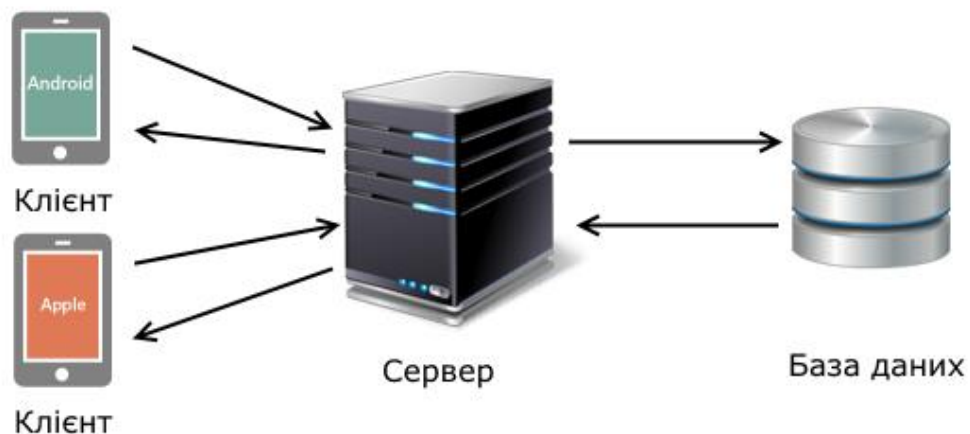


Рисунок 3 – Схема роботи інформаційної підсистеми "Розклад студента НУК"

Основними функціональними характеристиками системи є:

- ведення загальної інформаційної бази з розподіленим доступом;
- авторизований доступ до даних на основі прав доступу адміністратора;
- негайне відображення змін, внесених адміністратором;
- формування необхідних вихідних документів.

Система реалізує наступні функції управління і моніторингу:

- формування і редагування розкладу;
- формування статистики використання підсистеми;
- поширення новин серед користувачів;
- автоматизацію доступу користувачів до розкладу;

Система має звичний графічний інтерфейс користувача, що знижує витрати на навчання персоналу, і побудована з максимальним використанням існуючого на кафедрі у даний час устаткування і програмного забезпечення.

2.1.7. Захист від несанкціонованого доступу

Захист підсистеми від несанкціонованого доступу реалізовано на двох рівнях – системному рівні ОС і рівні розробленої програми. Системний рівень передбачає можливість входу користувача в систему і запуску програм. З достатнім ступенем надійності такий захист реалізований на комп'ютерах кафедри ІУСТ.

При запуску програми відбувається повторна авторизація користувача, причому його ім'я може не збігатися з ім'ям в операційній системі. Доступ до даних і функцій дозволяється тільки при правильному введенні користувальницького імені та пароля.

2.2. Рішення по інформаційному забезпеченню.

2.2.1 Вибір СУБД

Система управління базами даних (СУБД) — це комплекс мовних і програмних засобів, призначений для створення, ведення і сумісного використання БД багатьма користувачами. Зазвичай СУБД розрізняють по

використовуваній моделі даних. Так, СУБД, що базуються на використанні реляційної моделі даних, називають реляційними СУБД.

Основні вимоги до СУБД

- **Надійність:** ніякі дані не повинні пошкодитись. Збої повинні бути мінімізовані і не повинні призводити до втрат даних, бази повинні бути надійно захищені від несанкціонованого доступу.

- **Продуктивність:** СУБД повинна забезпечувати прийнятний рівень продуктивності для вирішення покладених на неї завдань.

Види СУБД

СУБД на даний момент існує безліч і вони класифікуються за різними ознаками. Стандартом в сучасних інформаційних системах є реляційні СУБД, в яких дані зберігаються в табличному вигляді. Розглянемо ключові параметри важливі як для розробників, так і користувачів системи.

Спосіб доступу до БД:

- Клієнт-серверні СУБД
- Файл-серверні СУБД
- Вбудовувані СУБД

У клієнт-серверних СУБД Microsoft SQL Server, Oracle, Firebird, PostgreSQL, InterBase, MySQL та ін) вся обробка даних ведеться в одному місці, на сервері, в тому ж місці, де зберігаються (зазвичай) дані, при цьому до файлів даних має доступ тільки один сервер, одна система – це сама СУБД. Програми-клієнти при цьому посилають запити на обробку та отримання даних з СУБД і отримують відповіді. Програми-клієнти не мають безпосереднього доступу до файлів даних. Всі промислові СУБД на даний момент є саме клієнт-серверними.

У файл-серверних СУБД (Paradox, Microsoft Access, FoxPro, dBase та ін), навпаки, програми мають загальний доступ до всіх файлів бази даних і спільно обробляють ці дані. Кожен додаток самостійно обробляє дані. На даний момент файл-серверна технологія вважається застарілою, а її використання у великих інформаційних системах — недоліком. Проблема в тому, що файл-серверні СУБД не мають багатьох переваг клієнт-серверних, таких як: кешування даних,

паралелізм запитів, висока продуктивність і володіють рядом недоліків (труднощі з підтриманням цілісності бази, відновленням, блокуваннями тощо), що призводить в свою чергу до зниженої надійності і продуктивності. Стан бази у файлових СУБД необхідно постійно відслідковувати і проводити операції з його "лікування" за допомогою вбудованих або сторонніх утиліт.

Вбудовані СУБД (SQLite, FirebirdEmbedded, Microsoft SQL Server Компактний та ін.) поставляються у складі готового програмного продукту, не вимагаючи процедури самостійної установки. Вбудовані СУБД призначені для локального зберігання даних програми і не розраховані на колективне використання в мережі. Наприклад, вбудована безкоштовно СУБД SQLite широко використовується у відомій мобільній ОС Android, розробленої Google, і в багатьох мобільних додатках.

Схема ліцензування:

- Безкоштовні СУБД
- Комерційні промислові СУБД

Файл-серверні і вбудовувані СУБД практично всі є безкоштовними, з безкоштовних клієнт-серверних СУБД найбільш відомі: Firebird, PostgreSQL і MySQL.

Більшість виробників промислових СУБД дають можливість користуватися безкоштовними редакціями своїх продуктів, які є спрощеними по функціоналу і по продуктивності варіантами повнофункціональної версії СУБД.

Плюси вільних СУБД: це безкоштовність, вони менш вимогливі до ресурсів ПК, володіють багатим функціоналом і хорошою продуктивністю при правильному налаштуванню, досить надійні.

Мінуси: існує можливість, що через певний час проект перестане існувати, оскільки його підтримує співтовариство ентузіастів.

Плюси комерційних СУБД: досконале документування, висока продуктивність, масштабованість, надійність, підтримка, наявність вбудованих інструментів для розробки і адміністрування. Ймовірність того, що компанія Oracle, Microsoft і IBM перестануть підтримувати свої системи, прагне до нуля.

Мінуси: вони більш вимогливі до ресурсів, ніж безкоштовні аналоги, коштують чималих грошей.

Враховуючи перераховані аспекти вибір зупинимо на безкоштовних СУБД. Розглянемо найпоширеніші безкоштовні СУБД.

MySQL

СУБД MySQL дуже функціональна, вільно поширювана, яка успішно працює з різними сайтами і додатками. Варто зауважити, що завдяки популярності цієї СУБД, існує величезна кількість різних плагінів і розширень, що полегшують роботу з системою.

Незважаючи на те, що в ній не реалізований весь SQL функціонал, MySQL пропонує досить багато інструментів для розробки додатків.

Переваги MySQL:

- Простота в роботі – встановити MySQL досить просто. Додаткові додатки, наприклад GUI, дозволяє досить легко працювати з БД
- Багатий функціонал – MySQL підтримує більшість функціоналу SQL.
- Безпека – реалізована велика кількість функцій, що забезпечують безпеку
- Масштабованість – MySQL легко працює з великими обсягами даних і легко масштабується
- Швидкість – нехтуючи деякими стандартами MySQL дозволяє значно збільшити продуктивність.

Недоліки MySQL:

- Відомі обмеження – за задумом в MySQL закладені деякі обмеження функціоналу, які необхідні в особливо вимогливих додатках.
- Повільна розробка – Хоча MySQL технічно відкрите ПЗ, існують скарги на процес розробки. Варто зауважити, що існують інші досить успішні СУБД створені на базі MySQL, наприклад MariaDB.

Коли слід використовувати MySQL:

- високий рівень безпеки – система безпеки MySQL включає в себе прості і в той же час гідні способи захисту доступу до даних

Веб сайти і веб-додатки – більшість сайтів і онлайн додатків спокійно працюють з MySQL незважаючи на деякі обмеження. Будучи легкої в налаштуванні і масштабованої системою – MySQL перевірена часом.

Індивідуальні рішення – якщо ви працюєте з яким-небудь специфічним проектом, MySQL легко зможе вам допомогти завдяки широким можливостям у налаштуванні і функціоналом.

Коли краще відмовитися від MySQL:

- Відповідність стандартам – Так як MySQL не ставить для себе мету повністю відповідати стандартам SQL, то ця СУБД не повністю підтримує SQL. Якщо в майбутньому ви плануєте перейти на таку систему, то MySQL – не кращий вибір.

- Багатопоточність – хоча деякі сервери БД досить легко виконують паралельне читання, паралельні операції читання-запису можуть створити проблеми

PostgreSQL

PostgreSQL – це просунута РСУБД, орієнтується в першу чергу на повну відповідність стандартам і розширюваність. PostgreSQL, або Postgres, намагається повністю відповідати SQL-стандартам ANSI/ISO.

Базуючись на потужній технології Postgres відмінно справляється з одночасною обробкою кількох завдань.

Хоча ця РСУБД не так популярна, як MySQL, існує багато сторонніх інструментів і бібліотек для полегшення роботи з PostgreSQL.

Переваги:

- Повна SQL-сумісність.
- Співтовариство: PostgreSQL підтримується досвідченим співтовариством 24/7.

- Підтримка сторонніми організаціями: незважаючи на дуже просунуті функції, PostgreSQL використовується в багатьох інструментах, пов'язаних з РСУБД.

- Розширюваність: PostgreSQL можна програмно розширити за рахунок збережених процедур.

- Об'єктно-орієнтованість: PostgreSQL — не тільки реляційна, але й об'єктно-орієнтована СУБД.

Недоліки:

- Продуктивність: До простих операціях читання PostgreSQL може поступатися своїм суперникам.

- Популярність: завдяки своїй складності інструмент не дуже популярний.

- Хостинг: завдяки перерахованих вище факторів проблематично знайти відповідного провайдера.

Коли варто використовувати PostgreSQL:

- Цілісність даних: якщо пріоритет стоїть на надійність і цілісність даних PostgreSQL — кращий вибір.

- Складні процедури: якщо ваша БД повинна виконувати складні процедури, варто вибрати PostgreSQL в силу її розширюваності.

- Інтеграція: якщо в майбутньому вам належить переміщати всю базу на інше рішення, найменше проблем виникне з PostgreSQL.

Коли не варто використовувати PostgreSQL:

- Швидкість: якщо все, що потрібно — це швидкі операції читання, не варто використовувати PostgreSQL.

- Прості ситуації: якщо не потрібна підвищена надійність, підтримка ACID, використання PostgreSQL лише підвищить вимоги до ЕОМ.

Проаналізувавши недоліки та переваги перерахованих СУБД вибір було зупинено на MySQL.

2.2.2 Вибір мови програмування

Розглянемо дві популярні мови програмування C++ і C#.

Швидкість розробки

C# дозволяє розпочати розробку швидше, а це дозволяє швидше отримати

прототип рішення. Швидкість розробки на C# на початкових етапах проекту значно вище в порівнянні з C++.

Однак, коли інфраструктура проекту створена, основні підходи та бібліотеки обрані, швидкість розробки на C++ і швидкість розробки на C# стають приблизно однаковими.

Таким чином, в коротких малобюджетних проектах C# матиме перевагу в швидкості розробки, але в довгих і відносно дорогих дана перевага буде незначною.

Кросплатформеніст

C++ кросплатформенна мова за фактом, хоча і з деякими нюансами, додатковими витратами, а також бінарною несумісністю між платформами.

C# спроектований бути кросплатформенним. Тому під системами Windows, Android та iOS утворилася досить повна .net інфраструктура, що активно вдосконалюється і підтримується.

І хоча C++ можливо використовувати для побудови додатків під не-Windows платформи, проблеми, викликані використанням в не-Windows оточенні, зводять нанівець переваги вибору C++.

Бібліотеки

Відмінність C++ і C# бібліотек в тому, що C++ бібліотек більше, вони мають велику історію, за яку стали непогано налагоджені та оптимізовані, часто кросплатформені, багато з відкритим кодом. Однак при всіх позитивних сторонах C++ бібліотеки мають дуже різну, часто навіть архаїчну архітектуру, часто не об'єктний, а структурно-процедурний інтерфейс. Пов'язано це з тим, що багато C++ бібліотеки це C бібліотеки.

Інша неприємна особливість C++ бібліотек — це створення і перевизначення своїх базових типів. Багато C++ бібліотек заводять свої типи рядків, контейнерів, зумовлюють деякі базові типи. Цьому є логічні пояснення (краща продуктивність, підтримка кросплатформеності, відсутність підходящих типів на момент написання бібліотеки), однак все це не додає зручності

використання. Базові ж C++ бібліотеки дають не так багато, як дають стандартні бібліотеки C#, тому підбір правильних бібліотек для проекту C++ — це завдання, необхідне навіть в порівняно простих проектах.

В C# величезна кількість бібліотек .net йде в базі, плюс до них безліч вільно доступних бібліотек, це покриває практично всі першорядні завдання розробки під Windows, а якщо до списку додати бібліотеки від Xamarin, то можна повністю охопити основні мобільні платформи. І в силу того, що бібліотеки C# порівняно молоді — інтерфейси бібліотек, як правило, краще вписуються в ті чи інші шаблони проектування, що часто спрощує їх вивчення.

Язык и Синтаксис

З першого погляду код C++ і C# дуже схожий зовні. Але різноманітність коду на C++ більше, адже C++ є одночасно і C і C++ і C++0x і все це можливо використовувати одночасно.

C# ж, це тільки C#, хоча синтаксис постійно розширюється. Код на C#, як правило, виглядає простіше і лаконічніше, ніж код C++. Мовні конструкції C++ і C# дуже схожі, однак суттєві відмінності можна знайти в деталях.

Синтаксис C#, мабуть, можна назвати спрощеною версією C++. Варто сказати, що більш складний код частіше легше пишеться і аналізується, якщо написаний більш простою мовою. З цієї позиції, використовуючи C#, менше шансів допустити помилку в принципово складному коді і більше шансів написати чистий код, володіючи тими ж ресурсами. Це може бути корисно при вирішенні досить складних, але не вимогливих до продуктивності завдань.

Так як розробка підсистеми націлена на кросплатформеність, яка характеризується доступом до розкладу з пристроїв студентів, що як правило мають різні системи, то найкращим рішенням буде мова програмування C#.

2.2.3 Організація інформаційної бази

В даній дипломній роботі розроблена база даних "Розклад" інформаційної підсистеми "Розклад студента НУК" для швидкого і структурованого доступу до розкладу студентів.

Розклад включає в себе наступні таблиці:

Викладачі

1. Аудиторії
2. Групи
3. Заняття
4. Час пари
5. Розклад

Таблиця 1. Викладачі

Викладачі	
Код викладача	ПІБ викладача
1	Викладач А.А.
2	Викладач А.А.
3	Викладач А.А.
4	Викладач А.А.
5	Викладач А.А.
6	Викладач А.А.

Таблиця 2.Аудиторії

Аудиторія			
Код аудиторії	Завдання	Поверх	Номер кабінету
1	Л	5	518
2	П	5	522
3	П	5	513
4	Л	6	625
5	Л	3	319

Таблиця 3.Групи

Групи				
Кодгрупи	Факультет	Спеціальність	Курс	Номер групи
1	Комп'ютерні науки	ІУСТ	1	1141
2	Комп'ютерні науки	ІУСТ	1	1141
3	Комп'ютерні науки	ІУСТ	1	1141
4	Комп'ютерні науки	ІУСТ	1	1141
5	Комп'ютерні науки	ІУСТ	1	1141

Таблиця 4. Розклад дзвінків

Розклад дзвінків		
Номер пари	Початок	Кінець
1	09-00	10-20
2	10-30	11-50
3	12-20	13-40
4	13-50	15-10
5	15-20	16-40
6	16-50	18-10

Таблиця 5. Заняття

Заняття	
Код заняття	Тип заняття
1	Організація баз даних
2	Web-технології
3	Операційні системи
4	Технології розподілених систем
5	Кроссплатформенне програмування

Таблиця 6. Розклад

Розклад						
Код заняття	Дата	№ пари	Група	Тип заняття	Викладач	Аудиторія
4	12.02.2022	2	1142	Програмування	Викладач А.А.	Д-2
5	13.01.2022	5	1142	Менеджмент	Викладач А.А.	Д-2
6	12.02.2022	6	1142	Менеджмент	Викладач А.А.	Д-2
7	13.01.2022	3	1142	Програмування	Викладач В.В.	Д-2
8	13.01.2022	1	1142	Організація баз даних	Викладач В.В.	Д-2
9	14.01.2022	4	1142	Операційні системи	Викладач В.В.	Д-2
10	13.05.2022	2	1142	Програмування	Викладач В.В.	Д-2

Проектування відносин. Після розробки всіх таблиць необхідно встановити зв'язки між ними.

Таблиця 7. Відносини інформаційних об'єктів

Номер	Головний об'єкт	Підлеглий об'єкт	Тип зв'язку
1	Викладачі	Розклад	1:Б
2	Аудиторія	Розклад	1:Б
3	Групи	Розклад	1:Б
4	Заняття	Розклад	1:Б
5	Розклад дзвінків	Розклад	1:Б

2.2.4. Опис інформаційної бази

Логічна модель бази даних

Процес проектування даних можна умовно розділити на два етапи: логічне моделювання та фізичне проектування. Результатом першого з них є так звана логічна (або концептуальну) модель даних, яка виражається зазвичай діаграми «сутність-зв'язок», або ER діаграмою, яка представлена в одній з стандартних позначень, прийнятих для відображення подібних діаграм.

Логічна модель даних інформаційної підсистеми "Розклад студента НУК" представлена на малюнку 4.

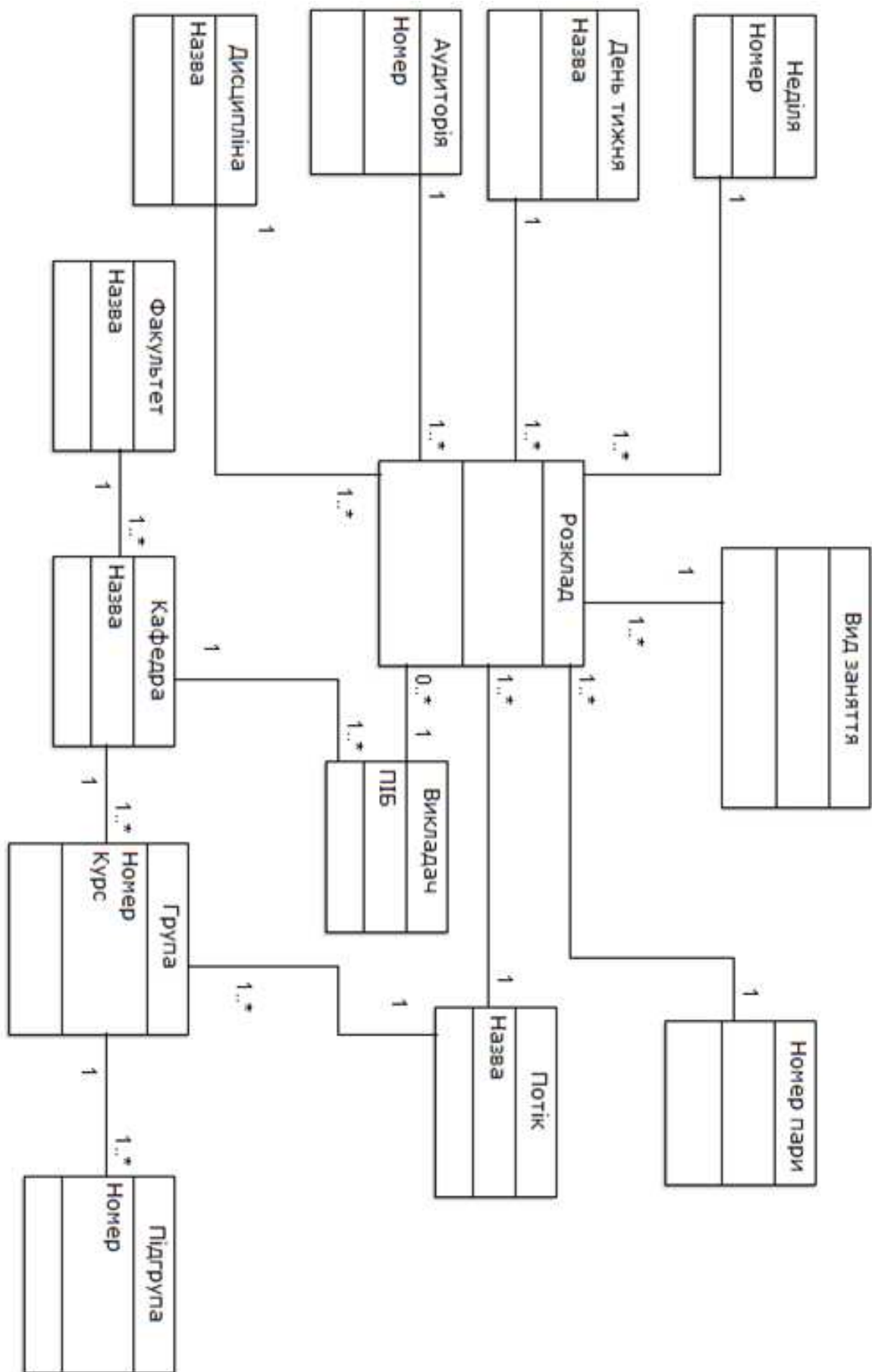


Рисунок 4 – Логічна модель БД інформаційної підсистеми "Розклад студента НУК"

Фізична модель бази даних

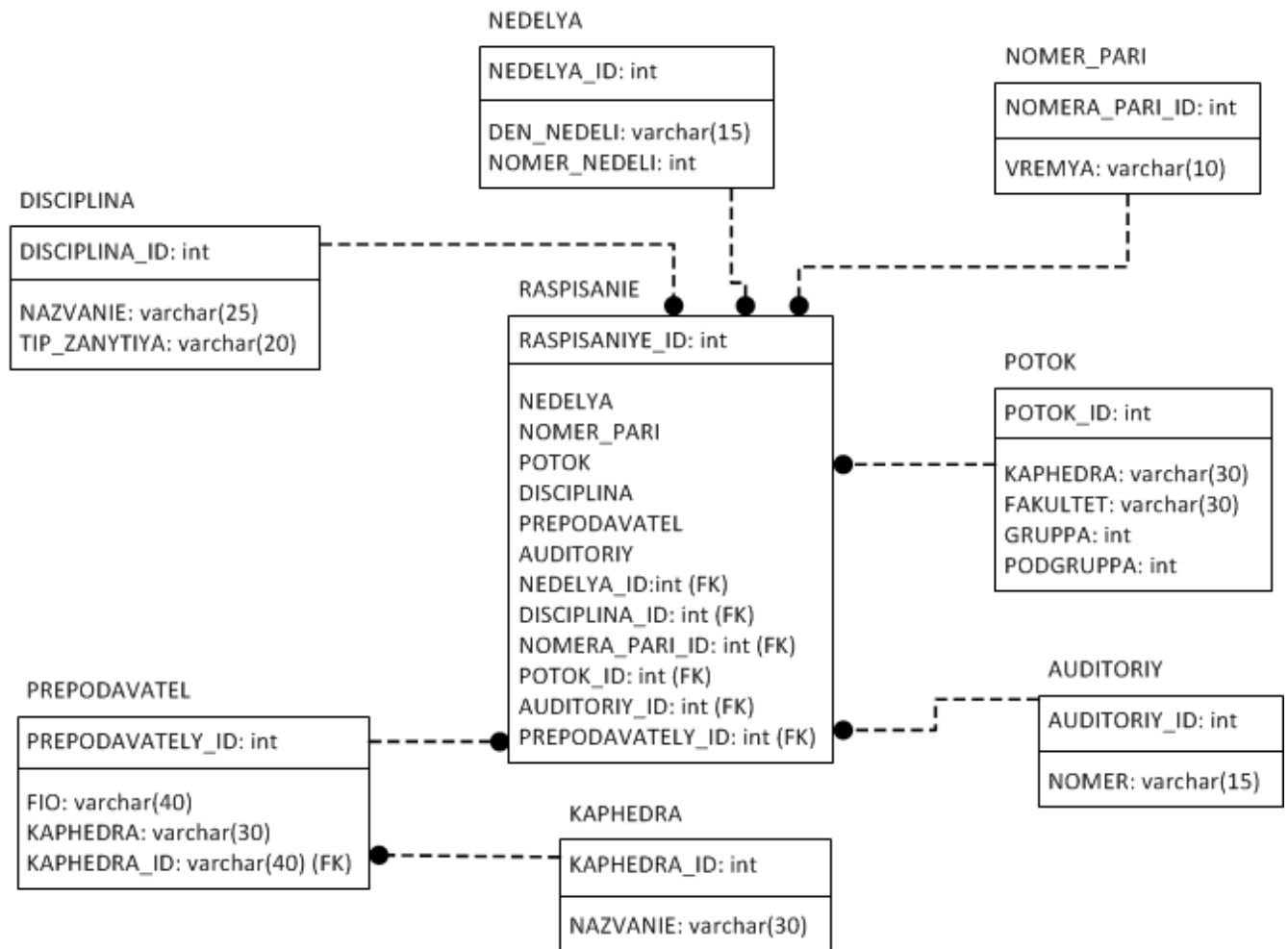


Рисунок 5. Фізична модель БД інформаційної підсистеми "Розклад студента НУК"

Діаграма даних представлена на рисунку 6. Таблиця Група, Аудиторія, Викладачі, Заняття, Час пари пов'язані з таблицею розклад допомогою кодового поля.

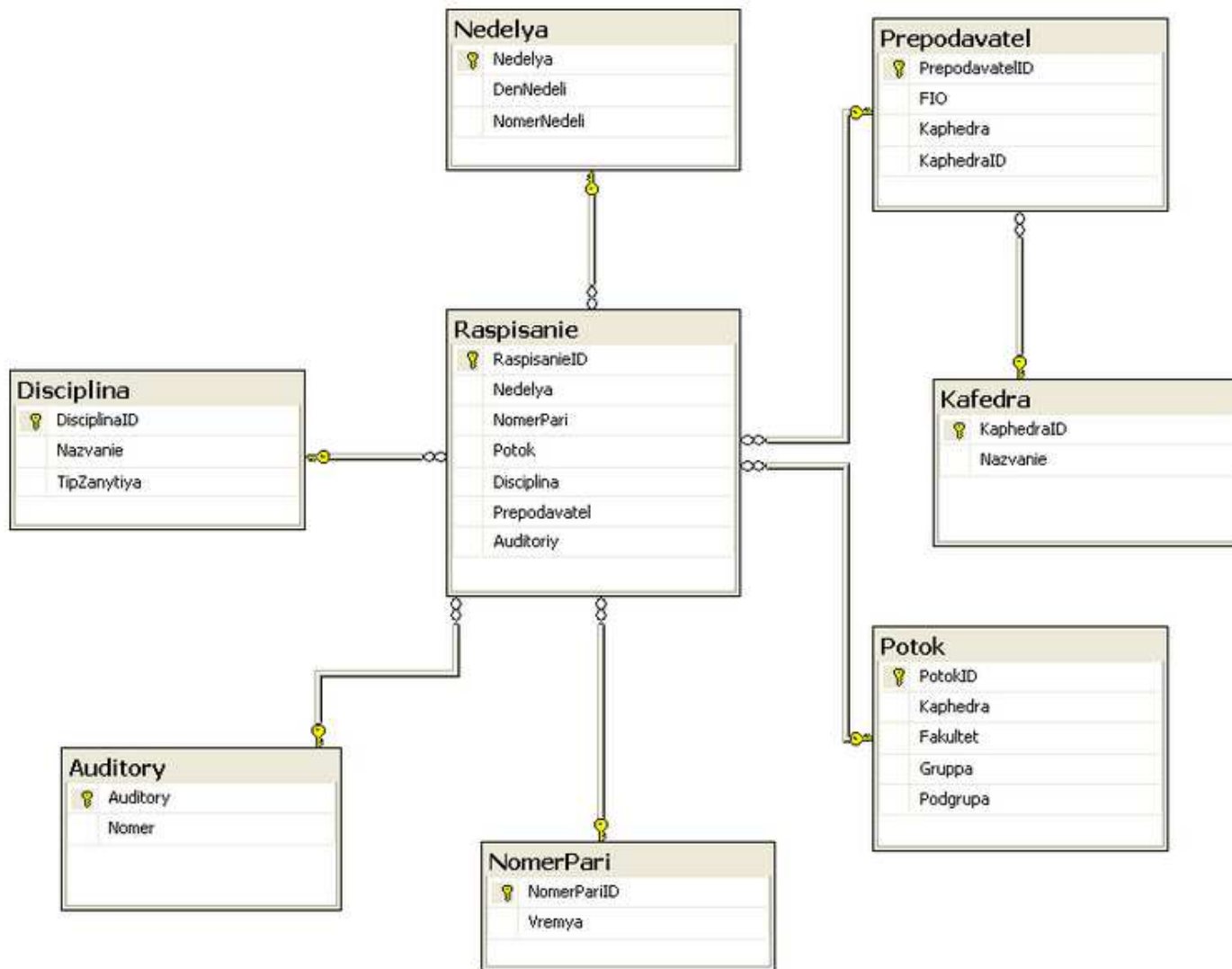


Рисунок 6 – Діаграма БД інформаційної підсистеми "Розклад студента НУК" із MS SQL Server

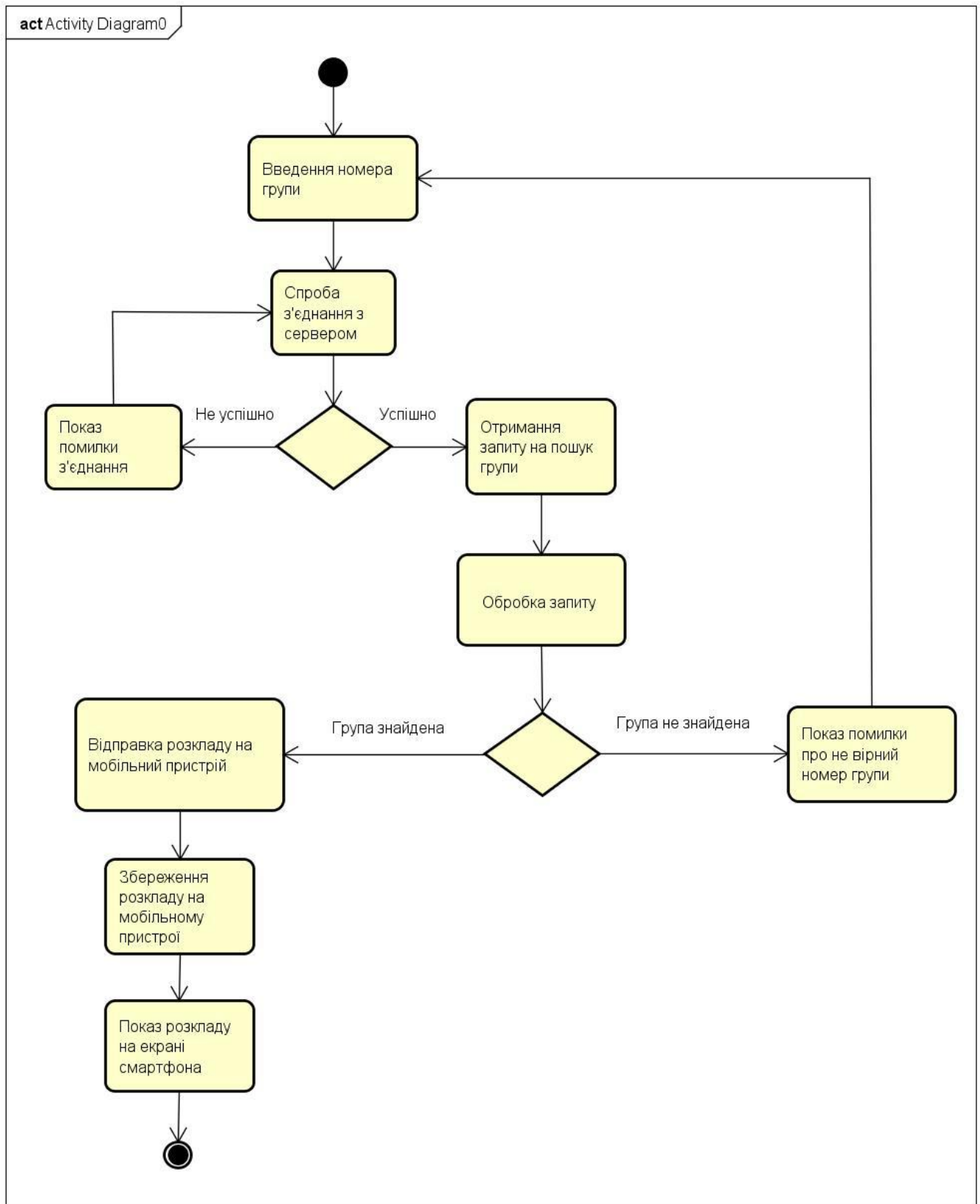
2.3 Опис алгоритмів функціонування системи

Алгоритм роботи підсистеми та її основних модулів представлений за допомогою діаграм діяльності

Діаграма діяльності (англ. Activity Diagram) – UML-діаграма, на якій показано розкладання деякої діяльності на її складові частини. Під діяльністю (англ. activity) розуміється специфікація виконуваного поведінки у вигляді координованого послідовного і паралельного виконання підлеглих елементів – вкладених видів діяльності та окремих дій, з'єднаних між собою потоками, які йдуть від виходів одного вузла до входів іншого.

Діаграми діяльності використовуються при моделюванні бізнес-процесів, технологічних процесів, послідовних і паралельних обчислень.

Діаграма діяльності всієї системи представлена на рисунку 7.



powered by Astah

Рисунок 7 – Діаграма діяльності підсистеми "Розклад студента НУК"

Діаграма діяльності для варіанту використання "Редагування даних" представлена на рисунку 8.

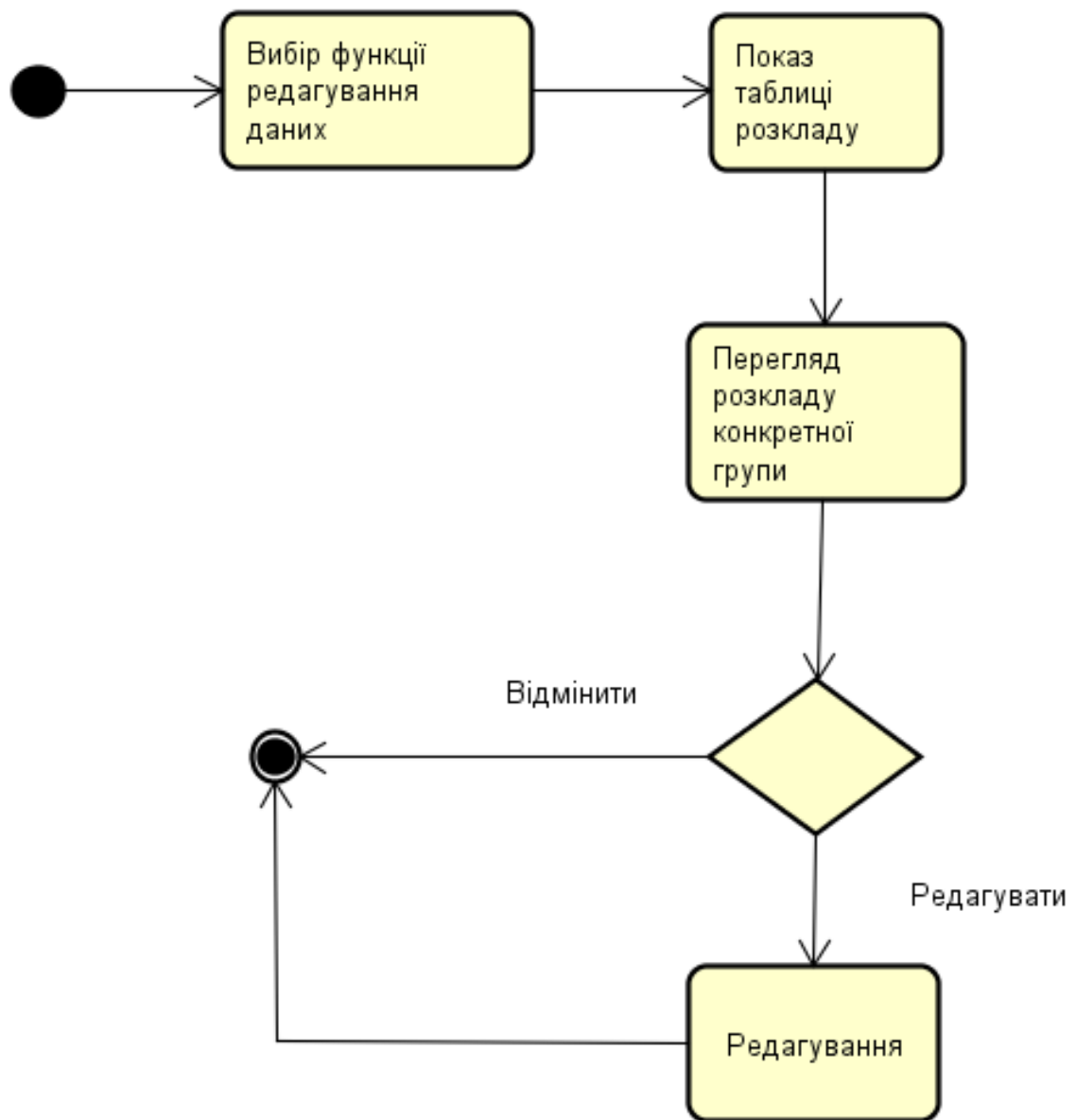


Рисунок 8 – Діаграма діяльності для варіанту використання "Редагування даних"

Діаграма діяльності для варіанту використання "Пошук групи" представлена на рисунку 9.

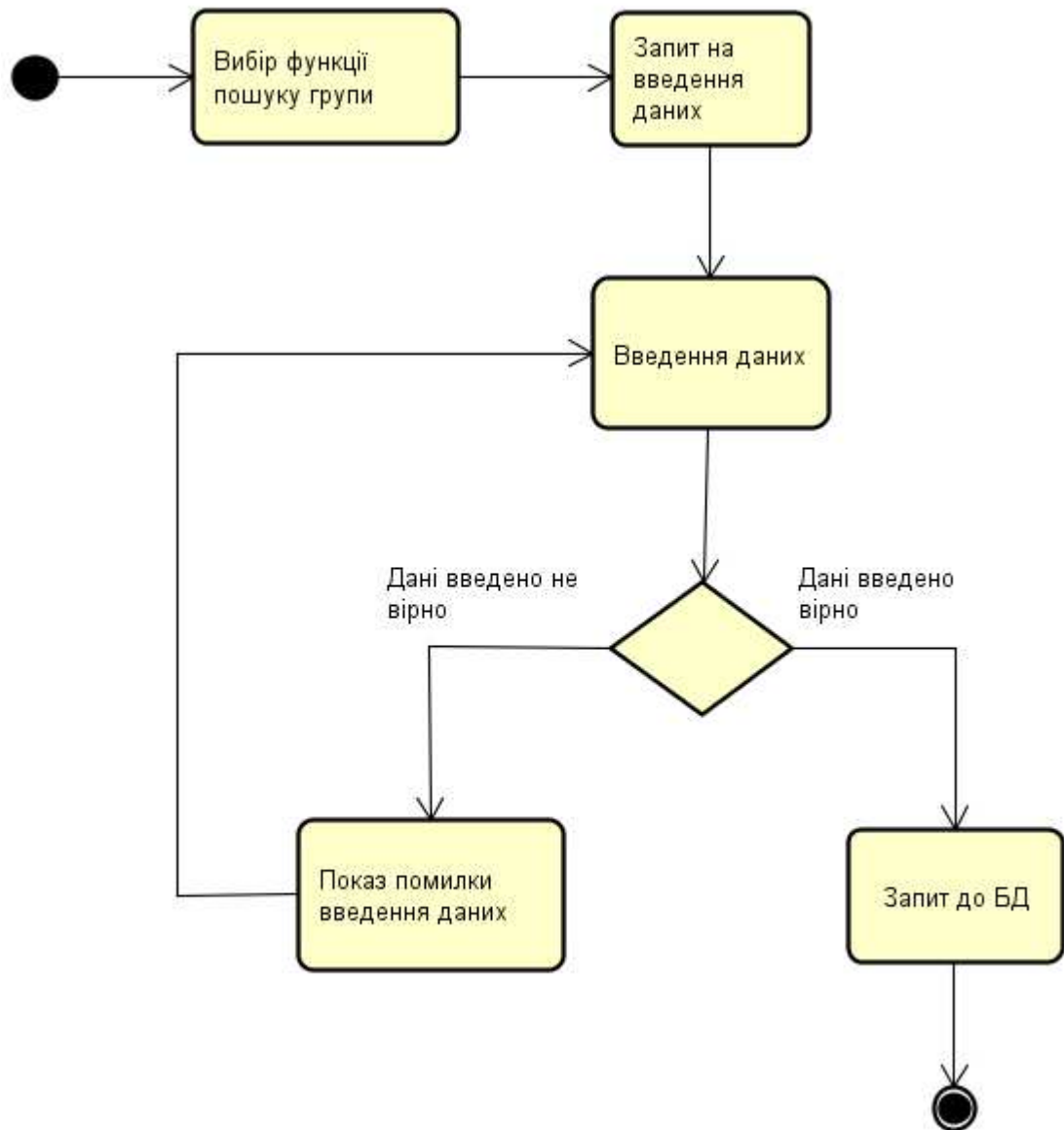


Рисунок 9 – Діаграма діяльності для варіанту використання "Пошук групи"

2.4 Рішення по програмному забезпеченню

Діаграма класів (Class Diagram) служить для представлення статичної структури моделі системи у термінології класів об'єктно-орієнтованого програмування. Діаграма класів може відображати, зокрема, різні взаємозв'язки між окремими сутностями предметної області, такими як об'єкти та підсистеми, а також описувати їхню внутрішню структуру та типи відносин.

Діаграма класів представлена на рисунку 10.

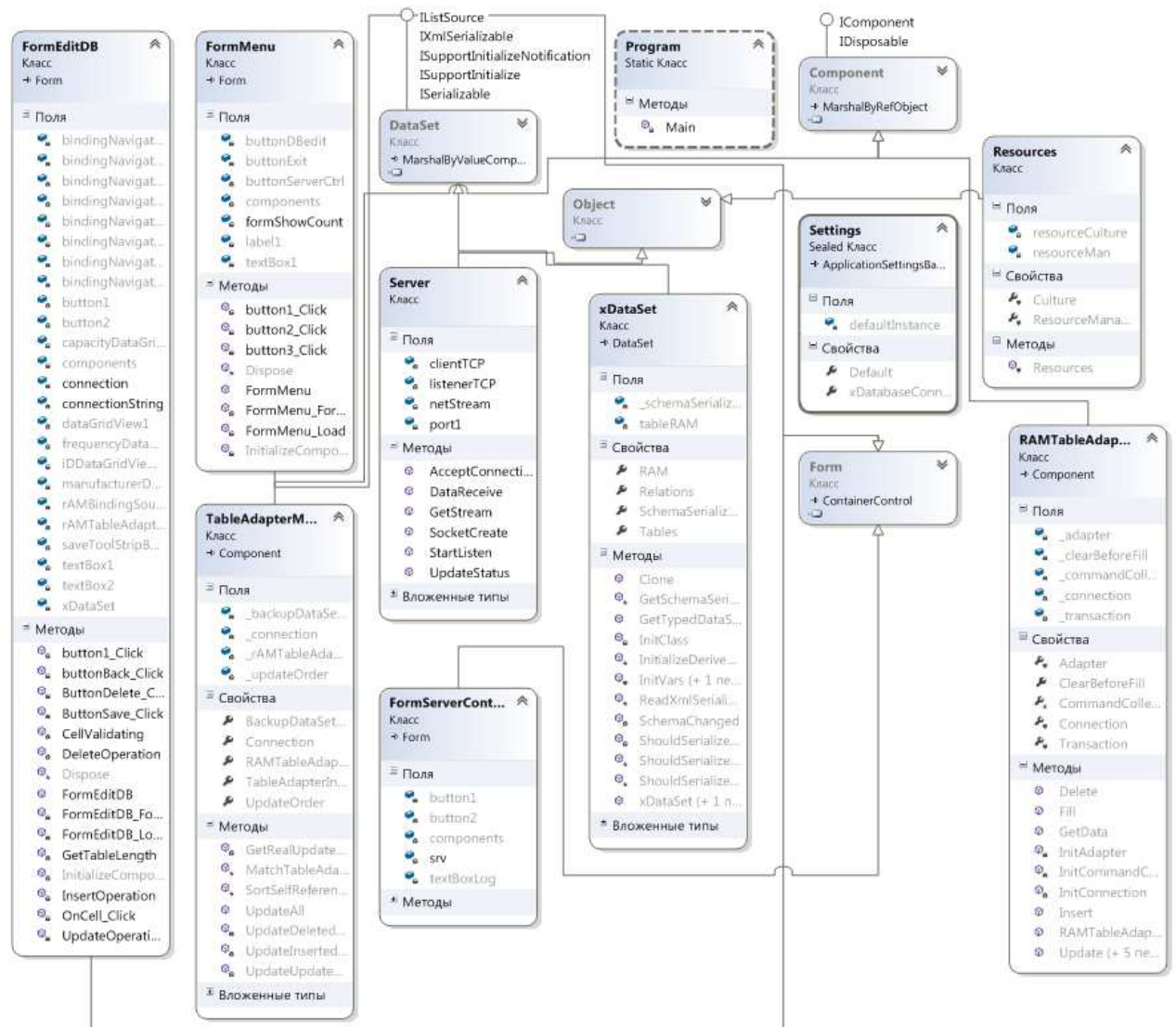


Рисунок 10 – Діаграма класів інформаційної підсистеми "Розклад студента НУК"

У проєкті програми використовуються наступні класи та:

Program.cs – основной клас. Служить для запуску програми.

Main – запускає програму.

FormMenu.cs – клас меню. Служить для запуску головного меню.

buttonDBedit – запускає вікно редагування розкладу.

`buttonServerCtrl` –запускає вікно керування сервером.

`buttonExit` – закриває програму.

buttonAbout – відкриває вікно з інформацією про програму.

FormEditDB.cs – клас що дозволяє редагувати дані таблиць розкладу.

`buttonSave` – виконує збереження таблиць.

buttonBack – виконує повернення до головного меню.

About.cs – служить для виводу інформації про додаток.

About – загрузає інформацію у відведені форми.

Server.cs – служить для виконання мережевих операцій по взаємодії з клієнтами.

netStream – виконує підключення до клієнта і отримання даних про номер групи.

SearchWork.cs – служить для обробки запитів клієнтів на пошук відповідної групи в БД.

search – виконує пошук за запитом в БД.

2.5 Рішення по технічному забезпеченню

2.5.1. Схема автоматизації.

Рациональний вибір технічних засобів, необхідних для впровадження автоматизованої системи буде значною мірою визначати загальний успіх або невдачу проекту. На підставі прийнятих загальносистемних рішень були визначені склад і організація комплексу технічних засобів.

Вимоги до продуктивності.

Обчислювальний комплекс повинний забезпечувати достатню продуктивність для рішення відповідних задач при бажаній мінімізації фінансових витрат на його придбання. Крім того, потрібно забезпечити прийнятний час реакції системи при роботі в інтерактивних режимах, які складають значну частину всіх режимів експлуатації системи. Таким чином буде забезпечена комфортна робота користувачів. Проектована система не буде пред'являти підвищених вимог до обчислювальної потужності. Створювану систему можна віднести до класу облікових задач, у яких загальна продуктивність системи визначається швидкістю операцій введення-виведення при неповному залученні потужності центрального процесора. У такий спосіб будь-яке сучасне апаратне рішення з пропонованих на ринку буде відповідати вимогам до продуктивності. Розподіл програмних компонентів майбутньої

системи дозволить не збільшувати навантаження на існуючі клієнтські робочі місця і не висуватиме вимог до їх модернізації.

Вимоги до надійності.

Розв'язувана задача висуває стандартні вимоги до надійності апаратних засобів без вимоги забезпечення повної стійкості від відмов. Основною вимогою в даній групі буде вимога до збереження даних. Передбачаються наступні заходи для її виконання:

- включення сервера в електромережу через пристрій безперебійного живлення(UPS);
- використання на сервері БД стійких до зносу жорстких дисків;
- використання спеціальних пристроїв резервного копіювання (наявність можливостей використовувати CD-RW привід та USB Flash накопичувач).

Вимоги до експлуатаційних характеристик.

Для мінімізації затрат часу на оперування підсистемою основною вимогою слід виділити простоту експлуатації обчислювальних засобів.

На основі вищесказаного можна зробити висновок про необхідність застосування для сервера БД і клієнтських додатків стандартного апаратного забезпечення. До сервера БД пред'являється традиційний набір вимог для такого рішення.

2.5.2. Структурна схема комплексу обчислювальних засобів

Інформаційна система буде спроектована за схемою "клієнт-сервер", тому апаратне рішення буде складатися з обчислювальної підсистеми і мережі передачі даних.

Схема діаграми розміщення технічних засобів наведена на рисунку 11.

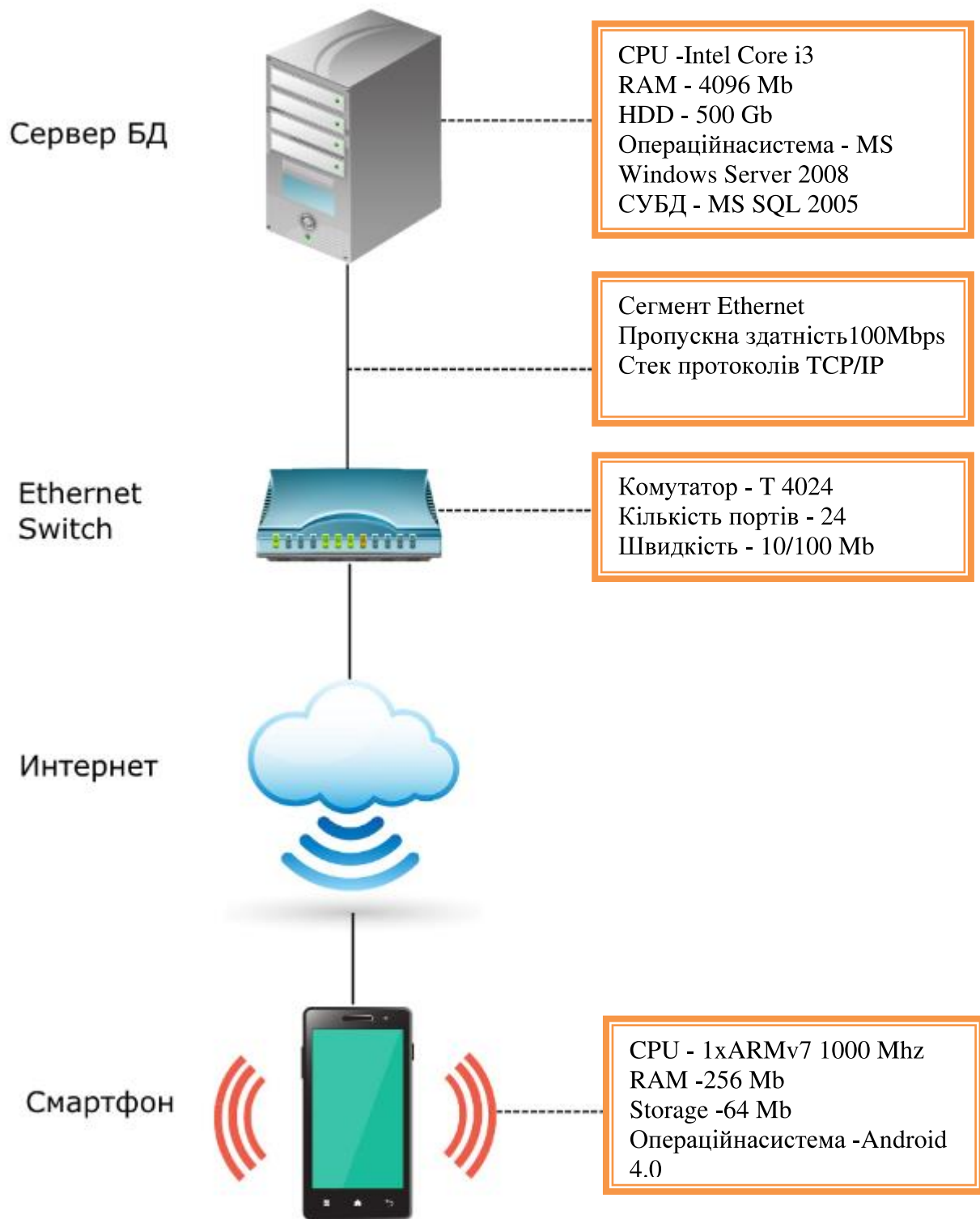


Рисунок 11 – Діаграма розміщення комплексу технічних засобів.

Розділ 3.

Розробка робочої документації

Підсистема віддаленого доступу до розкладу "Розклад студента НУК" призначена для автоматизування процесу отримання студентами розкладу своїх груп і своєчасного сповіщення про будь-які зміни в ньому.

Основними функціями, що автоматизуються є:

- Доступ до розкладу групи за її номером;
- Занесення розкладу до бази даних;
- Редагування розкладу;
- Розсилка сповіщень про зміни в розкладі;
- Розсилка новин;
- Формування статистики використання додатку.

При впровадженні програмного забезпечення підсистеми основними цілями є:

- зменшити витрати часу студентів на пошук і перегляд розкладу, що дозволить їм більш ефективно використовувати час перерв і не запізнюватись на заняття;
- знизити ризики помилок при пошуку аудиторій для проведення;
- дозволити здійснювати перегляд розкладу за межами університету, всюди же є підключення інтернет;
- підвищити рівень організації учбового процесу за допомогою використання інформаційних технологій, що розроблені в даній дипломній роботі.

Для ефективного використання програмного забезпечення важливо впровадити на організаційні і технічні заходи для оптимальної взаємодії з підсистемою. При роботі з програмним забезпеченням підсистеми слід керуватися документами "Загальний опис автоматизованої підсистеми Розклад студента НУК" (Додаток А), "Керівництво користувача" (Додаток Б), "Керівництво системного адміністратора" (Додаток В).

Розділ 4.

Охорона праці у виробничих приміщеннях з ЕОМ

4.1 Загальні принципи охорони праці

Згідно до ст.1 Закону "Про охорону праці", охорона праці є системою правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних та лікувально-профілактичних заходів та засобів, спрямованих на збереження здоров'я і працездатності людей в процесі праці. Як правовий інститут охорона праці визначає цілий комплекс правових норм.

Ст.4 Закону "Про охорону праці" формулює принципи державної політики у сфері охорони праці. Ці принципи в силу вказаного їх визначення, слід кваліфікувати і як принципи інституту охорони праці, які підлягають застосуванню при відсутності відповідних конкретних правових норм, і як такі, що визначають напрямки подальшого розвитку інституту охорони праці, і обов'язки держави, які вона взяла на себе і має додержувати при здійсненні своїй політики у майбутньому. З урахуванням викладеного, принципи, зазначені ст. 4 Закону "Про охорону праці", мають певне регулятивне значення, названій статті закріплені такі принципи:

1) пріоритет життя і здоров'я працівників відносно результатів виробничої діяльності підприємства, повна відповідальність власника за створення безпечних і нешкідливих умов праці;

2) комплексне вирішення завдань охорони праці на засадах національних програм з цих питань, а також з урахуванням інших напрямків економічної і соціальної політики, досягнень в галузі науки, техніки охорони здоров'я;

3) соціальний захист працівників, повне відшкодування шкоди особам потерпілим від нещасних випадків на виробництві та професійних захворювань;

4) установлення єдиних нормативів з охорони праці для всіх підприємств незалежно від форм власності і видів їх діяльності;

5) використання економічних методів управління охороною праці;

6) проведення політики пільгового оподаткування, то сприяє створенню безпечних та нешкідливих умов праці, участь держави в фінансуванні заходів з

охорони праці;

7) здійснення навчання населення, професійної підготовки та підвищення кваліфікації працівників з питань охорони праці;

8) забезпечення координації діяльності державних органів, установ, організацій та об'єднань громадян, що вирішують різні проблеми охорони здоров'я, гігієни та безпеки праці, а також співробітництво.

Умови праці на робочому місці, безпека технологічних процесів, машин, механізмів, устаткування та інших засобів виробництва, стан засобів колективного та індивідуального захисту, що використовуються працівником, а також санітарно-побутові умови повинні відповідати вимогам нормативних актів про охорону праці. Власник або уповноважений ним орган повинен впроваджувати сучасні засоби техніки безпеки, які запобігають виробничому травматизмові, і забезпечувати санітарно-гігієнічні умови, що запобігають виникненню професійних захворювань працівників.

На власника або уповноважений ним орган покладається систематичне проведення інструктажу (навчання) працівників з питань охорони праці, протипожежної охорони.

Охорона праці – один з центральних інститутів трудового права. Він має виключно практичне значення. Недодержання вимог охорони праці створює небезпеку для здоров'я і життя працівників. У свою чергу і ті, кого законодавець називає власником або уповноваженим ним органом, несуть сувору, у тому числі й кримінальну, відповідальність за порушення правил охорони праці.

4.2 Аналіз небезпечних і шкідливих факторів при роботі з ПК у відділі програмного забезпечення

З точки зору впливу небезпечних і шкідливих факторів при роботі з ПК у відділі програмного забезпечення можна виділити наступні:

- 1) вплив електромагнітного та електростатичного випромінювань;
- 2) вплив недостатньої освітленості робочого місця;
- 3) вплив електричного струму;
- 4) вплив шуму;

5) вплив несприятливих умов мікроклімату;

6) вплив нераціонального розташування устаткування і неправильної організації робочого місця.

Тривале перебування людини в зоні комбінованого впливу різних несприятливих факторів може привести до професійного захворювання. Розглянемо більш детально вплив означених факторів на користувача ПК.

4.2.1 Вплив електромагнітного та електростатичного випромінювань

При роботі ПК виникають два типи випромінювань: електростатичне та електромагнітне. Навколо електростатично зарядженого монітора підвищується концентрація пилу. Електромагнітне випромінювання створюється магнітними котушками, що знаходяться біля цокольної частини ЕПТ. Результати медичних досліджень показують, що електризований пил викликає запалення шкіри, а електромагнітне випромінювання приводить до порушення біологічних процесів (порушення сну, зниження кров'яного тиску, уповільнення серцевої діяльності, зміни складу крові).

4.2.2 Вплив недостатньої освітленості робочого місця

Напружена зорова робота внаслідок нераціонального освітлення може бути причиною функціональних порушень у зоровому аналізаторі і привести до розладу зору, а у важких випадках – і до повної втрати. Втома органів зору залежить від ступеня напруженості процесів, що супроводжують зорове сприйняття. При роботі оператора ПК одним з основних навантажень на організм є навантаження на зір, тому дуже важливо приділити особливу увагу освітленості робочого місця.

У залежності від джерела світла виробниче висвітлення може бути трьох видів: природним, штучним і сполученим. При роботі з ПК вдень у ясну погоду використовується природне освітлення, яке здійснюється через світлові віконні прорізи. При нестачі природного світла, а також при роботі у вечірній час використовується штучне освітлення, джерелами якого є газорозрядні лампи.

При такому освітленні величезне значення має правильне розташування джерел світла, їхній тип і кількість.

Правильна освітленість робочого місця сприяє високій працездатності користувача ПК і зводить до мінімуму шкідливий вплив на органи зору. Освітленість робочого місця, що рекомендується для користувача ПК складає 400–750 лк.

4.2.3 Вплив електричного струму

Робота комп'ютерів і периферійних пристроїв можлива завдяки електричному струмові, до джерел якого вони підключаються, тому існує небезпека ураження користувача струмом. Щоб уникнути нещасних випадків варто строго дотримуватися правил електробезпеки.

Небезпека електричного струму на відміну від інших небезпек збільшується тим, що людина не в змозі без спеціальних приладів виявити напругу дистанційно, а також швидкоплинністю поразки – небезпека виявляється, коли людина вже уражена. Ураження струмом приводить до різних порушень в організмі, викликаючи як місцеву поразку тканин і органів, так і загальне ураження організму. Людина відчуває дратівну дію перемінного струму промислової частоти силою 0,6–1,5 мА і постійного струму 5–7 мА. Ці струми не представляють серйозної небезпеки для організму людини, тому що при такій силі струму можливе самостійне звільнення людини від контакту зі струмоведучими частинами. Для перемінного струму промислової частоти сила невідпускаючого струму знаходиться в межах 6-20мА. Постійний струм не викликає невідпускаючого ефекту, але приводить до сильних болючих відчуттів, сила такого струму 15-80мА і більше.

4.2.4 Вплив шуму

Шумом прийнято називати всякий небажаний для людини звук, що заважає сприйняттю корисних сигналів. Шум являє собою безладне сполучення звуків різної інтенсивності і частоти. Він викликає труднощі з розпізнанням колірних сигналів, знижує швидкість сприйняття кольору, гостроту зору, зорову

адаптацію, порушує сприйняття візуальної інформації, зменшує на 5-12% продуктивність праці. Тривалий вплив шуму з рівнем звукового тиску 90 дБ знижує продуктивність праці на 30-60%.

Робота ПК може супроводжуватися шумами, викликаними деякими рухливими пристроями усередині комп'ютера, а також підключеними до нього. До таких пристроїв можна віднести вентилятори, пристрої читання дисків, принтери. Виникаючий шум, у залежності від інтенсивності, може привести до підвищення нервової напруги і зниження працездатності.

4.2.5 Вплив несприятливих умов мікроклімату

Мікрокліматичні параметри (температура, відносна вологість і швидкість руху повітря) впливають на самопочуття і здоров'я людини, а також на надійність роботи засобів обчислювальної техніки.

При підвищеній температурі і вологості в приміщенні відбувається перегрівання тіла, що погіршує самопочуття і працездатність. Варто враховувати, що ПК, які знаходяться в приміщенні, сприяють збільшенню температури.

Для комфортної роботи приймається температура $23 (\pm 2)^{\circ}\text{C}$ й $18 (\pm 2)^{\circ}\text{C}$ в теплий і холодний час року відповідно, при відносній вологості $55 (\pm 5)\%$.

4.2.6 Вплив нераціонального розташування устаткування і неправильної організації робочого місця

Робоче місце – це частина простору, у якій користувач ПК здійснює трудову діяльність і проводить велику частину робочого часу. Важливу роль грає планування робочого місця, яке повинне задовольняти вимогам зручності виконання робіт і економії енергії і часу користувача, зручності обслуговувань ЕОМ, дотримання правил охорони праці.

При плануванні робочого місця необхідно враховувати зони досяжності при розташуванні моніторів, клавіатури, маніпулятора «миша». Ці зони встановлюються на підставі антропологічних даних людського тіла. Під час

роботи користувач ПК не повинен піддаватися впливові сторони подразників, якими можуть бути неадекватне фарбування пристроїв ЕОМ або приміщення.

4.3 Розрахунок освітленості робочого місця

В приміщенні відділу програмного забезпечення необхідно організувати змішане освітлення (комбінування природного і штучного освітлення). Природне освітлення створюється бічним освітленням через вікно. Штучне освітлення використовується при недостатньому природному освітленні. У даному приміщенні використовується загальне штучне освітлення. Розрахунок виконаємо за методом світлового потоку з урахування потоку, віддзеркаленого від стін та стелі.

Необхідно зробити розрахунок загального освітлення виробничої приміщення. Довжина $A = 40$ м, ширина $B = 16$ м, висота $H = 4,5$ м. Висота робочої поверхні $h_p = 1$ м. Для освітлення використовується світильні УТІМ-5. Мінімальна освітленість лампи наколювання по нормах $E_{min}=100$ лк. Коефіцієнт відображення стелі $S_n=50\%$, стін $S_c=30\%$, робочої поверхні $S_p=10\%$. Напруга мережі 220 В.

Відстань від стелі до робочої поверхні:

$$H_0 = H - h_p = 4,5 - 1 = 3,5 \text{ м.}$$

Відстань від стелі до світильника:

$$h_c = 0,2 H_0 = 0,2 * 3,5 = 0,7 \text{ м.}$$

Висота підвісу світильника над освітленою поверхнею:

$$h_c = 0,2 H_0 = 0,2 * 3,5 = 0,7 \text{ м.}$$

Висота підвісу світильника над підлогою:

$$H_n = h + h_p = 2,8 + 1 = 3,8 \text{ м.}$$

Для досягнення найбільшої рівномірності освітлення приймаємо відношення $L/h = 1,5$.

Відстань між центрами світильників:

$$L = 1,5h = 1,5 * 2,8 = 4,2 \text{ м.}$$

Необхідна кількість світильників:

$$N = S / L^2 = (40 * 16) / (4,2)^2 \approx 36.$$

Приймаємо кількість світильників: 36 (3 ряди по 12 світильників).

Індекс приміщення:

$$i = (A \cdot B) / (h \cdot (A + B)) = (40 \cdot 16) / (2,8 \cdot (40 + 16)) = 4,08.$$

Коефіцієнт використання світлового потоку $\eta = 0,66$.

Світловий потік однієї лампи:

$$\Phi = (E_{min} \cdot S \cdot K_z \cdot Z) / (N \cdot \eta) = (100 \cdot 640 \cdot 1,3 \cdot 1,15) / (36 \cdot 0,66) = 4027 \text{ лм.}$$

Вибираємо (у таблиці) лампу, що має світловий потік 4610 лм:

Г215-225-300.

Фактична освітленість:

$$E_{\phi} = E_{min} \cdot \Phi_{\lambda} / \Phi_p = 100 \cdot 4610 / 4027 = 114 \text{ лк.}$$

Загальна потужність:

$$P_{\text{общ}} = P_{\lambda} \cdot N = 300 \cdot 36 = 10800 \text{ Вт} = 10,8 \text{ кВт.}$$

Таким чином, для оптимальної роботи в приміщенні необхідно встановити 3 ряди по 12 світильників типу Г215-225-300. Загальна потужність усіх світильників складе 10,8 кВт.

4.4 Розробка заходів щодо зменшення впливу небезпечних і шкідливих факторів

Оскільки найбільшу загрозу при роботі на ЕОМ встановлює електростатичне та електромагнітне випромінювання. Заходи щодо зниження впливу. Для зниження величини виникаючих зарядів статичної електрики

покриття технологічних підлог варто виконувати з одношарового полівінілхлоридного антистатичного лінолеуму. Іншим методом захисту є нейтралізація заряду статичної електрики іонізованим газом. У промисловості широко застосовуються радіоактивні нейтралізатори. До загальних мір захисту від статичної електрики можна віднести загальне і місцеве зволоження повітря.

Окрім цього необхідно розташовувати монітори ПК таким чином щоб електромагнітне випромінювання від їх електронно-променевих трубок не було направлено на людей. Також слід використовувати захисні екрани, що зменшують електростатичне випромінювання моніторів. Крім цього, на сьогоднішній день існують суміші які при обробці ним екранів також

зменшують електростатичне поле, що направлене безпосередньо на людину. Крім того необхідно дотримуватися загальної техніки безпеки при користуванні електричною технікою.

Висновок

У дипломній роботі була розроблена інформаційна підсистема "Розклад студента НУК" інформаційної системи кафедри ІУСТ.

Результатом роботи є розроблена автоматизована підсистема "Розклад студента НУК", що представляє швидкий, структурований і зручний доступ студентам до розкладу з докладним відображенням її в табличному вигляді. За допомогою даного ПЗ студенти можуть дізнатися детальну інформацію про кожне заняття. Наприклад таку як: аудиторія, викладач, час початку і кінця. Крім того присутня функція розсилки повідомлень про зміни в розкладі та функція розсилки новин університету.

Впровадження запропонованої в дипломному проекті, інформаційної підсистеми дозволить спростити доступ до розкладу, знизити час на його отримання, а також своєчасно повідомити про внесені зміни студентів, тим самим зменшити кількість можливих запізнень студентів на заняття. Таким чином всі ці параметри ведуть до збільшення продуктивності начального процесу в цілому.

Список літератури

1. Фленов М. Библия С# 2-е издание [текст] \ М. Фленов. – СПб.: "БХВ-Петербург", 2013. – 541 с.
2. Культин Н. С# в задачах и примерах. [текст] \ Н. Культин. – СПб.: "БХВ-Петербург", 2007. – 241 с.
3. Зиборов В. В. Visual С# 2012 на примерах. [текст] \ В. В. Зиборов – СПб.: "БХВ-Петербург", 2013. – 475 с.
4. Фролов А. В. Визуальное проектирование приложений С#. [текст] \ А. В. Фролов – СПб.: "Кудиц-образ", 2003. – 512 с.
5. Петцольт Ч. Программирование для Windows на С#. [текст] \ Ч. Петцольт – СПб.: Издательско-торговый дом "Русская Редакция", 2002. – 576 с.
6. Зиборов В. Visual С# 2012 на примерах. #. [текст] \ В. Зиборов – СПб.: "БХВ-Петербург", 2013. – 380 с.
7. Рихтер Дж. Программирование на платформе Microsoft.NET Framework 4.5 [текст] \ Дж. Рихтер 4-е изд. – СПб.: "Питер ", 2013. – 896 с.

Додаток А – Загальний опис автоматизованої підсистеми "Розклад студента НУК"

Повна назва розробленої підсистеми:

"Розклад студента НУК".

Стисла характеристика області використання:

Підсистему планується використовувати для отримання студентами інформації про розклад занять їх груп, сповіщення про зміни в розкладі і для розсилки новин.

Мета розробки: Підвищити рівень організації учбового процесу студентів кафедри ІУСТ та полегшити студентам процес отримання розкладу.

Зменшити витрати часу студентів на пошук і перегляд розкладу, що дозволить більш ефективно використовувати час перерв і не запізнюватись на заняття. Знизити ризики помилок при пошуку аудиторій для проведення занять. Реалізувати можливість здійснювати перегляд розкладу за межами університету, всюди же є підключення Інтернет.

Функціональне призначення

За допомогою цього додатку можна отримати детальну інформацію про розклад кожної групи кафедри ІУСТ. Вчасно отримувати сповіщення про зміни в розкладі занять. А також переглядати новини університету.

Эксплуатационное назначение

Додаток націлений на використання студентами кафедри ІУСТ. Використання даного ПО дозволить студентам бути в курсі останніх новин університету, швидко і зручно отримувати доступ до розкладу занять, а також отримувати сповіщення про зміни в ньому.

Додаток Б – Керівництво користувача

1. Для використання програми необхідно встановити і запустити мобільний додаток "Розклад НУК".
2. Перед користувачем з'явиться головне вікно додатку, в якому треба ввести номер групи, щоб здійснити пошук розкладу. Слід зазначити що для пошуку та завантаження розкладу мобільний пристрій повинен бути підключений до мережі Інтернет.
3. Після введення номера групи необхідно натиснути кнопку "Пошук" для здійснення пошуку розкладу відповідної групи. Через кілька секунд розклад буде завантажений в пам'ять пристрою, завдяки чому стане можливий його перегляд в режимі оффлайн.
4. Відразу після успішного пошуку на екрані відобразиться меню в якому можна подивитися різноманітну інформацію що стосується проведення занять, наприклад назва предмету, час проведення, аудиторія та поверх де буде проходити заняття і т.п.
5. У разі якщо в розкладі конкретної групи були внесені зміни, то при підключенні мобільного пристрою до Інтернету відбудеться розсилка сервером відповідних повідомлень.
6. Серед пунктів меню присутній розділ новини в якому час від часу будуть з'являтися інформація про різні події і заходи університету.
7. Щоб повернутись в головне меню необхідно натиснути кнопку меню у верхньому кутку будь якого вікна додатку.
8. Для виходу з додатку слід швидко два рази натиснути апаратну кнопку "Назад" або згорнути додаток і завершити його як звичайний системний процес.

Додаток В – Керівництво адміністратора

1. Для використання програми необхідно запустити файл "NukTimetable.exe".

2. Перед адміністратором з'явиться панель авторизації що передбачає введення логіну і паролю. Після успішного її проходження з'явиться меню адміністратора, яке дозволяє контролювати роботу сервера, здійснювати редагування розкладу, переглядати статистику використання і здійснювати розсилку новин.

3. Пункт меню "Управління сервером" дозволяє включити і відключити сервер.

4. Пункт меню "Редагування розкладу" дозволяє вносити зміни у розклад груп.

5. Пункт меню "Перегляд статистики" дозволяє переглянути кількість звернень мобільних пристроїв до бази даних.

6. Пункт меню "Розсилка новин" дозволить адміністратору складати короткий опис новин і здійснювати їх розсилку користувачам мобільних додатків.

7. Для повернення до головного меню з будь-якого вікна слід натиснути кнопку "Назад"

Для виходу з програми необхідно натиснути в головному меню кнопку "Вихід". Слід зазначити, що у разі якщо сервер працює виникне вікно вибору між згортанням програми і повним виходом з відключенням серверу. В останньому випадку дію необхідно буде підтвердити введенням паролю адміністратора.

Приложение Б – Програма і методика випробувань

Об'єкт випробувань

Об'єкт випробувань є інформаційна підсистема "Розклад студента НУК"

Мета випробувань

Метою випробувань є перевірка програмного забезпечення на наявність помилок і відповідності реалізованих функцій зазначеним в технічному завданні. Якщо прийнятний рівень відповідності програмного забезпечення відповідає вимогам, а також відсутні помилки і грубі недоліки, то програмне забезпечення приймається може бути прийняте в експлуатацію.

Вимоги до програми

Для проведення випробувань розглядали наступні функції:

- функція авторизації;
- функція пошуку розкладу;

Вимоги до програмної документації

В комплект програмної документації повинні входити наступні документи:

- код програми (ГОСТ 19.401-78);
- програма і методика випробувань (ГОСТ 19.301-79);
- інструкція користувача (ГОСТ 19.505-79).
- інструкція адміністратора (ГОСТ 19.505-79).

Засоби і порядок випробувань

Для випробувань даної програми необхідно, щоб на комп'ютері, де буде встановлено програмне забезпечення "Розклад студента НУК" була встановлена операційна система Windows 7/8/10 і програмне забезпечення .Net Framework 4.5.

Для роботи з програмою необхідний наступний склад технічних засобів з такими мінімальними параметрами:

- процесор Intel Celeron G1840 2.8 ГГц;
- обсяг ОЗП 1 Гб;
- вільне місце на диску: 50 Мб

- графічний адаптер GeForce 210 512 Мб;
- монітор SVGA 0.26, 15 дюймів;
- маніпулятори "миша" і клавіатура;

Випробування програмного забезпечення повинні виконуватися в наступному порядку:

- виконуються інструкції до кожної функції;
- робиться аналіз отриманих результатів і встановлюється відповідність програмного продукту вимогам і системним документам.

При виявленні помилок в роботі системи складається їх перелік і обговорюється термін їх виправлення розробником.

Після цього замовник проводить повторне тестування в повному обсязі (відновлений використання нових або додаткових тестів).

Методи випробувань

Випробування будемо проводити методом "чорного ящика". За рахунок дуже великої кількості функцій, які потрібно відчувати, представимо тільки кілька функцій програми. Усі функції програми будуть проходити випробування.

Складемо програму випробувань:

Функція "Авторизація".

При в ході в програму адміністратор повинен пройти авторизацію. Для цього він повинен мати можливість ввести інформацію в поля "Логін" і "Пароль". Якщо логін і пароль вірні, то має виводитися повідомлення «Успішно!» і користувач потрапить в головне меню програми. Якщо логін буде введений неправильно, то має виводитися повідомлення «Невірний логін» і адміністратору необхідно буде перевірити правильність введення даних в полі "Логін". Якщо пароль введений неправильно, то має виводитися повідомлення "Невірний пароль" і користувачу необхідно буде перевірити правильність введення даних в поле "Пароль".

Функція "Пошук групи"

Після запуску клієнтського додатку користувач може виконати пошук в базі розкладу кафедри ІУСТ. Інформаційне вікно системи містить поле "Пошук групи". Дане поле служить для введення пошукового запиту і при успішному в воді даних виводиться результат пошуку. Якщо залишити поле порожнім, на екран буде виведено повідомлення про порожній запит.

Додаток Г – Графічні інтерфейси

В цьому додатку представлений проект користувацького інтерфейсу програмного забезпечення "Розклад студента НУК», створений в програмі Microsoft Visual Studio.

Панель авторизації представлена на рисунку 12.

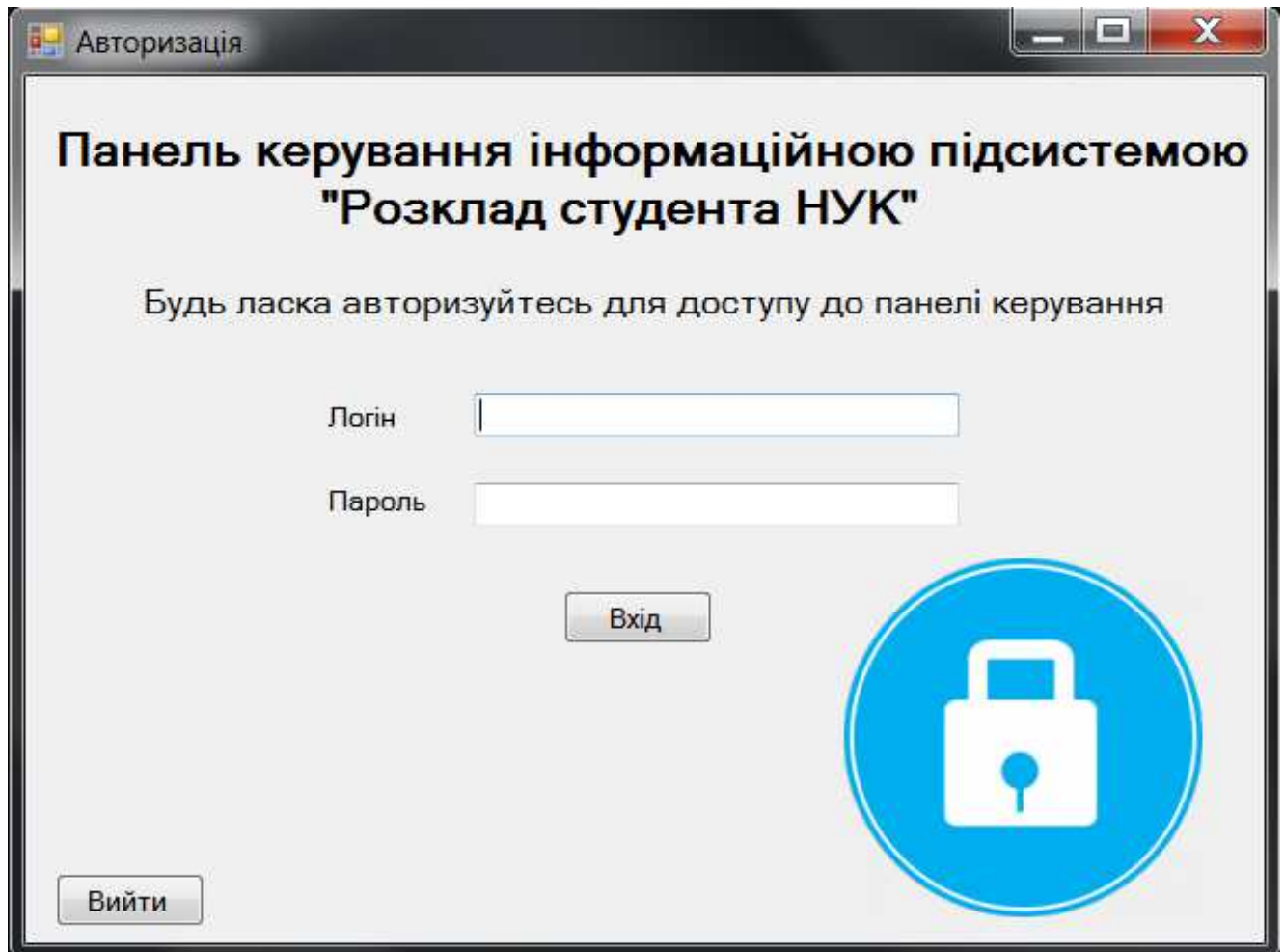


Рисунок 12 – Панель входу адміністратора

Меню адміністратора представлено на рисунку 13.

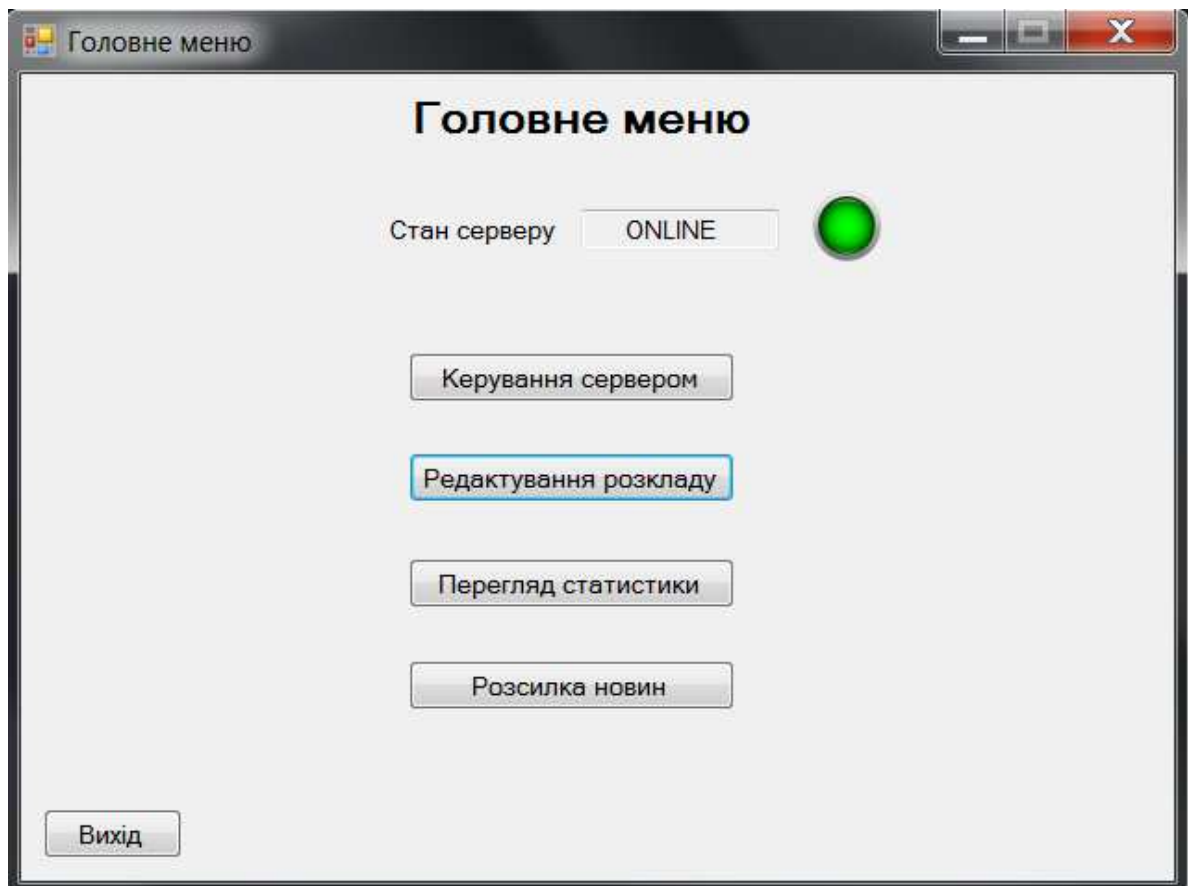


Рисунок 13 – Головне меню адміністратора

Стартове вікно мобільного додатку представлено на рисунку 14.

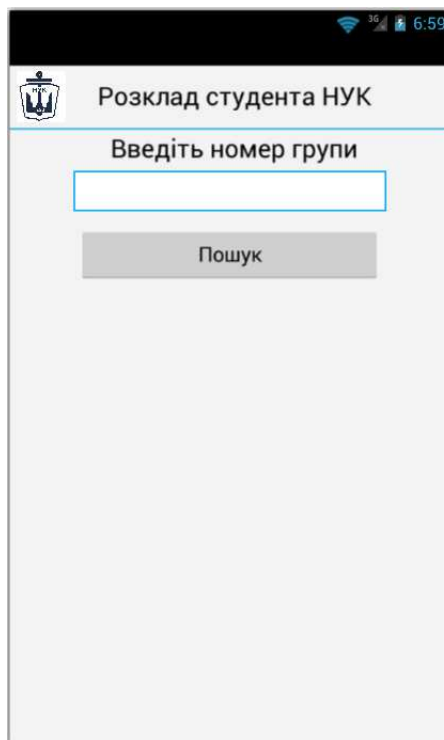


Рисунок 14 – Стартове вікно мобільного додатку

Меню мобільного додатку представлено на рисунку 15.

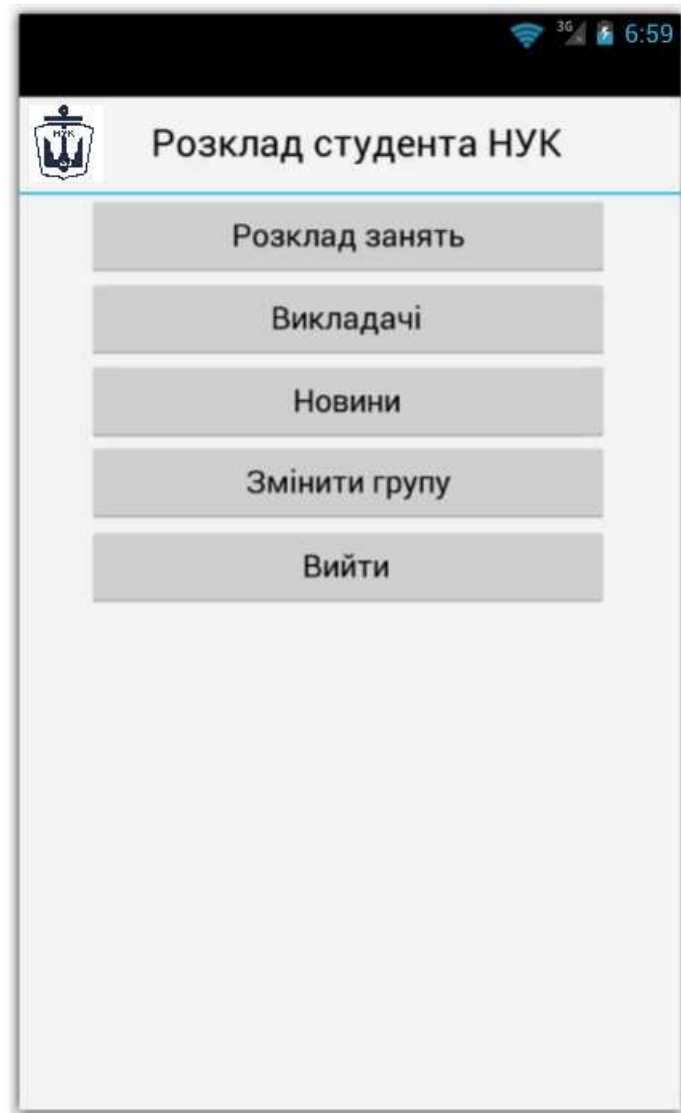


Рисунок 15 – Меню мобільного додатку "Розклад студента НУК"

Розділ "Розклад" мобільного додатку представлено на рисунку 16.



Рисунок 16 – Меню мобільного додатку "Розклад студента НУК"

Додаток Д – код програми

клас Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace СУБД
{
    static class Program
    {
        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new LogIn());
        }
    }
}
```

клас Server.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Net;
using System.Net.Sockets;

namespace СУБД
{
    public class Server
    {
        TcpListener listenerTCP;
        TcpClient clientTCP;
        NetworkStream netStream;
        string port1 = "5461";

        //Делегат для обновления логов
        private delegate void UpdateStatusCallback(string
        StatusMessage);

        public string SocketCreate()
        {
            string hostName = Dns.GetHostName();
            IPAddress ip = null;
```

```

        IPAddress[] ipList =
        Dns.GetHostEntry(hostName).AddressList;
        for (int i = 0; i < ipList.Length; i++)
        {
            if (ipList[i].ToString().Length > 6 &&
                ipList[i].ToString().Length < 16)
            {
                ip = ipList[i];
            }
        }

        int port = int.Parse(port1);
        //создание сокета
        listenerTCP = new TcpListener(ip, port);
        return "Полный адрес сервера: " +
        listenerTCP.LocalEndpoint + "\r\n";
    }

    public static void UpdateStatus()
    {
        FormServerControl frm = new FormServerControl();
        frm.getTextBox().Text = "sdada";
    }

    public string StartListen()
    {
        listenerTCP.Start();
        return "Начата прослушка сокета. Ожидание
        подключений..." + "\r\n";
    }

    public string AcceptConnection()
    {
        string message = "";
        //проверка, на подключения в режиме ожидания
        if (listenerTCP.Pending())
        {
            clientTCP = listenerTCP.AcceptTcpClient();
            message = "Соединение с клиентом установлено" +
            "\r\n";
        }
        else
        {
            message = "Нет подключений" + "\r\n";
        }
        clientTCP = listenerTCP.AcceptTcpClient();
        message = "Соединение с клиентом установлено" + "\r\n";
        return message;
    }

    public string GetStream()
    {

```

```

        string message = "";

        try
        {
            netStream = clientTCP.GetStream();
            message += "Сетевой поток определен" + "\r\n";
        }
        catch (Exception ex)
        {
            message += "Ошибка " + ex.ToString() + "\r\n";
        }
        return message;
    }
    //получение данных
    public string DataReceive()
    {
        string message = "";

        if (netStream != null)
        {
            byte[] bytes = new
byte[clientTCP.ReceiveBufferSize];
            int bytesRead = netStream.Read(bytes, 0,
clientTCP.ReceiveBufferSize);
            // Строка, содержащая ответ от сервера
            string returnData = Encoding.UTF8.GetString(bytes);
            message += returnData;
        }
        else
        {
            message += "Ошибка, поток закрыт " + "\r\n";
        }

        return message;
    }
}

```

клас Menu.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace СУБД
{
    public partial class Menu : Form
    {

```

```

public Menu()
{
    InitializeComponent();
}

private void button2_Click(object sender, EventArgs e)
{
    Application.Exit();
}

private void button1_Click(object sender, EventArgs e)
{
    this.Hide();
    FormEditDB formEditDB = new FormEditDB();
    formEditDB.Show();
}

private void FormMenu_FormClosing(object sender,
FormClosingEventArgs e)
{
    e.Cancel = true;
    this.WindowState = FormWindowState.Minimized;
}

private void button3_Click(object sender, EventArgs e)
{
    this.Hide();
    FormServerControl formServerControl = new
FormServerControl();
    formServerControl.Show();
}
}
}

```

Класс EditDB.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

using System.Data.SqlServer;

namespace СУБД
{
    public partial class FormEditDB : Form
    {
        string connectionString;
        SqlConnection connection;
    }
}

```



```

public FormEditDB()
{
    InitializeComponent();
}

private void FormEditDB_Load(object sender, EventArgs e)
{
    // TODO: данная строка кода позволяет загрузить данные
    в таблицу "xDataSet.RAM". При необходимости она может быть
    перемещена или удалена.
    this.rAMTableAdapter.Fill(this.xDataSet.RAM);

    connectionString = "Data Source=xDatabase.sdf;Persist
Security Info=False";
    //connectionString = "Data Source=\"xDatabase.sdf\"";
    //connectionString = @"Data
Source=\"c:\users\admin\documents\xDatabase.sdf\"";
    // Создание подключения
    connection = new SqlConnection(connectionString);
    try
    {
        //Открытие подключения
        connection.Open();
        //textBox1.Text += "Подключение открыто" + "\r\n";
    }
    catch (SqlCeException ex)
    {
        textBox1.Text += ex.Message;
    }
    finally
    {
        //Закрытие подключения
        connection.Close();
        //textBox1.Text += "Подключение закрыто" + "\r\n";
    }
}

private void FormEditDB_FormClosing(object sender,
FormClosingEventArgs e)
{
    e.Cancel = true;
    this.WindowState = FormWindowState.Minimized;
}

private void OnCell_Click(object sender,
DataGridViewCellEventArgs e)
{
    textBox2.Text = "строка=" +
(dataGridView1.CurrentCell.OwningRow.Index + 1).ToString() +
"\r\n";
    textBox2.Text += "столбец=" +
(dataGridView1.CurrentCell.OwningColumn.Index + 1).ToString() +
"\r\n";
}

```

```

        textBox2.Text += "имя столбца=" +
dataGridView1.CurrentCell.OwningColumn.HeaderText + "\r\n";
        textBox2.Text += "столбцы: ";
        for (int i = 0; i <
dataGridView1.CurrentCell.OwningRow.Cells.Count; i++)
        {
            textBox2.Text +=
dataGridView1.CurrentCell.OwningRow.Cells[i].OwningColumn.HeaderText + ", ";
            if (i ==
dataGridView1.CurrentCell.OwningRow.Cells.Count - 1)
            {
                textBox2.Text += "\r\n";
            }
        }
        textBox2.Text += "данные строки: ";
        for (int i = 0; i <
dataGridView1.CurrentCell.OwningRow.Cells.Count; i++)
        {
            textBox2.Text +=
dataGridView1.CurrentCell.OwningRow.Cells[i].Value + ", ";
            if (i ==
dataGridView1.CurrentCell.OwningRow.Cells.Count - 1)
            {
                textBox2.Text += "\r\n";
            }
        }
    }

private void ButtonSave_Click(object sender, EventArgs e)
{
    int SQLtableLength = GetTableLength();

    //обновление данных
    //перебор всех строк таблицы
    for (int i = 0; i < SQLtableLength; i++)
    {
        ///перебор всех столбцов строки
        //for (int j = 0; j < dataGridView1.ColumnCount;
j++)
        //{
            //
            //textBox2.Text +=
dataGridView1.Rows[i].Cells[j].Value + ", ";
            //
            //если последняя ячейка строки
            //if (j ==
dataGridView1.CurrentCell.OwningRow.Cells.Count - 1)
            //{
                //
                //textBox2.Text += "\r\n";
            //}
        //}

        int id =
int.Parse(dataGridView1.Rows[i].Cells[0].Value.ToString());

```

```

        string task =
dataGridView1.Rows[i].Cells[1].Value.ToString();
        int time =
int.Parse(dataGridView1.Rows[i].Cells[2].Value.ToString());
        int teacher =
int.Parse(dataGridView1.Rows[i].Cells[3].Value.ToString());
        //если достигнута последняя строка в таблице из
базы данных
        if (i == SQLtableLength - 1)
        {
            //вывод последней строки из базы данных
            textBox1.Text += "Данные обновлены\t" + "\r\n";
        }
        UpdateOperation(id, task, time, teacher, id);
    }
    //если в таблице из БД строк меньше чем в
dataGridView1, то выполняется команда добавления
    if (SQLtableLength < dataGridView1.RowCount - 1)
    {
        //textBox2.Text = "prse";
    }
    //добавление данных
    for (int i = SQLtableLength; i < dataGridView1.RowCount
- 1; i++)
    {
        int id =
int.Parse(dataGridView1.Rows[i].Cells[0].Value.ToString());
        string task =
dataGridView1.Rows[i].Cells[1].Value.ToString();
        int time =
int.Parse(dataGridView1.Rows[i].Cells[2].Value.ToString());
        int teacher =
int.Parse(dataGridView1.Rows[i].Cells[3].Value.ToString());
        //если достигнута последняя строка в таблице
dataGridView1
        if (i == dataGridView1.RowCount - 2)
        {
            //вывод последней добавленной строки
            //textBox2.Text += "" + id + task + time +
teacher;
            textBox1.Text += "Данные добавлены\t" + "\r\n";
        }

        InsertOperation(id, task, time, teacher);
    }
}

private void ButtonDelete_Click(object sender, EventArgs e)
{
    //если выделенны несколько строк
    if (dataGridView1.SelectedRows.Count > 0)
    {
        for (int i = 0; i <
dataGridView1.SelectedRows.Count; i++)

```

```

        {
            int id =
int.Parse(dataGridView1.SelectedRows[i].Cells[0].Value.ToString());
            DeleteOperation(id);

dataGridView1.Rows.RemoveAt(dataGridView1.SelectedRows[i].Index);
        }
    }
    else
    {
        int id =
int.Parse(dataGridView1.CurrentCell.OwningRow.Cells[0].Value.ToStri
ng());
        DeleteOperation(id);
    }
}

private void CellValidating(object sender,
DataGridViewCellValidatingEventArgs e)
{
    ////Открытие подключения
    //connection.Open();
    ////обновление строки в
    //if (dataGridView1.IsCurrentCellDirty)
    //{
        //
        string columnToUpdate =
dataGridView1.CurrentCell.OwningColumn.HeaderText;
        //
        string value =
dataGridView1.CurrentCell.EditedFormattedValue.ToString();
        //
        string id =
dataGridView1.CurrentCell.OwningRow.Cells[0].Value.ToString();
        //
        SqlCeCommand command = new SqlCeCommand();
        //
        command.Connection = connection;
        //
        command.CommandText = "UPDATE TimeTable SET " +
columnToUpdate + "=" + "'" + value + "'" + " WHERE ID=" + "'" + id
+ "'";
        //
        command.ExecuteNonQuery();
        //
        textBox1.Text += command.CommandText;
    //}
    ////Закрытие подключения
    //connection.Close();
}

private void UpdateOperation(int id, string task, int time,
int teacher, int rowID)
{
    //Открытие подключения
    connection.Open();
    SqlCeCommand command = new SqlCeCommand();
    command.CommandText = string.Format("UPDATE TimeTable
SET ID={0}, Task='{1}', Time={2}, Teacher={3} WHERE ID={4}", id,
task, time, teacher, rowID);
    command.Connection = connection;
    command.ExecuteNonQuery();
}

```

```

        //Заккрытие подключения
        connection.Close();
    }

    private void InsertOperation(int id, string task, int time,
int teacher)
    {
        //Открытие подключения
        connection.Open();
        //Выполнение команды SQL Добавление
        SqlCeCommand command = new SqlCeCommand();
        command.Connection = connection;
        command.CommandText = string.Format("INSERT INTO
TimeTable (ID, Task, Time, Teacher) VALUES ({0}, '{1}', {2}, {3})",
id, task, time, teacher);
        command.ExecuteNonQuery();
        //Заккрытие подключения
        connection.Close();
    }

    private void DeleteOperation(int id)
    {
        //Открытие подключения
        connection.Open();
        //Выполнение команды SQL Добавление
        SqlCeCommand command = new SqlCeCommand();
        command.Connection = connection;
        command.CommandText = string.Format("DELETE FROM
TimeTable WHERE ID={0}", id);
        command.ExecuteNonQuery();
        textBox1.Text += "Данные удалены\t" + "\r\n";
        //Заккрытие подключения
        connection.Close();
    }

    private int GetTableLength()
    {
        //Открытие подключения
        connection.Open();
        //Выполнение команды SQL на получение числа строк
        SqlCeCommand command = new SqlCeCommand();
        command.Connection = connection;
        command.CommandText = "SELECT COUNT(*) FROM TimeTable
";
        int rowCount =
int.Parse(command.ExecuteScalar().ToString());
        //Заккрытие подключения
        connection.Close();

        tableRowCount = int.Parse(number);
        return rowCount;
    }

    private void button1_Click(object sender, EventArgs e)

```

```

        {
            int id =
int.Parse(dataGridView1.CurrentRow.Cells[0].Value.ToString());
            DeleteOperation(id);

dataGridView1.Rows.RemoveAt(dataGridView1.CurrentRow.Index);
        }

        private void buttonBack_Click(object sender, EventArgs e)
        {
            this.Close();
            FormMenu formMenu = new FormMenu();
            formMenu.Show();
        }
    }
}

клас LogIn.cs
namespace СУБД
{
    partial class LogIn
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources
should be disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            System.ComponentModel.ComponentResourceManager
resources = new
System.ComponentModel.ComponentResourceManager(typeof(LogIn));
            this.labell = new System.Windows.Forms.Label();

```

```

        this.label2 = new System.Windows.Forms.Label();
        this.label3 = new System.Windows.Forms.Label();
        this.pictureBox1 = new
System.Windows.Forms.PictureBox();
        this.textBox1 = new System.Windows.Forms.TextBox();
        this.textBox2 = new System.Windows.Forms.TextBox();
        this.label4 = new System.Windows.Forms.Label();
        this.button1 = new System.Windows.Forms.Button();
        this.button2 = new System.Windows.Forms.Button();

        ((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).BeginInit();
        this.SuspendLayout();
        //
        // label1
        //
        this.label1.AutoSize = true;
        this.label1.Location = new System.Drawing.Point(149,
162);
        this.label1.Name = "label1";
        this.label1.Size = new System.Drawing.Size(42, 17);
        this.label1.TabIndex = 0;
        this.label1.Text = "Логін";
        //
        // label2
        //
        this.label2.AutoSize = true;
        this.label2.Location = new System.Drawing.Point(149,
204);
        this.label2.Name = "label2";
        this.label2.Size = new System.Drawing.Size(57, 17);
        this.label2.TabIndex = 1;
        this.label2.Text = "Пароль";
        //
        // label3
        //
        this.label3.AutoSize = true;
        this.label3.Font = new System.Drawing.Font("Microsoft
Sans Serif", 10F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte) (204)));
        this.label3.Location = new System.Drawing.Point(55,
103);
        this.label3.Name = "label3";
        this.label3.Size = new System.Drawing.Size(522, 20);
        this.label3.TabIndex = 2;
        this.label3.Text = "Будь ласка авторизуйтесь для
доступу до панелі керування";
        //
        // pictureBox1
        //
        this.pictureBox1.Image =
((System.Drawing.Image)(resources.GetObject("pictureBox1.Image")));
        this.pictureBox1.Location = new
System.Drawing.Point(411, 242);

```

```

        this.pictureBox1.Name = "pictureBox1";
        this.pictureBox1.Size = new System.Drawing.Size(180,
181);

        this.pictureBox1.TabIndex = 3;
        this.pictureBox1.TabStop = false;
        //
        // textBox1
        //
        this.textBox1.Location = new System.Drawing.Point(225,
159);

        this.textBox1.Name = "textBox1";
        this.textBox1.Size = new System.Drawing.Size(244, 22);
        this.textBox1.TabIndex = 4;
        //
        // textBox2
        //
        this.textBox2.Location = new System.Drawing.Point(225,
204);

        this.textBox2.Name = "textBox2";
        this.textBox2.Size = new System.Drawing.Size(244, 22);
        this.textBox2.TabIndex = 5;
        //
        // label4
        //
        this.label4.AutoSize = true;
        this.label4.Font = new System.Drawing.Font("Microsoft
Sans Serif", 14F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((byte) (204)));
        this.label4.Location = new System.Drawing.Point(10,
22);

        this.label4.Name = "label4";
        this.label4.Size = new System.Drawing.Size(612, 58);
        this.label4.TabIndex = 6;
        this.label4.Text = "Панель керування інформаційною
підсистемою\r\n                                \r\nРозклад студента " +
        "НУК\>";
        //
        // button1
        //
        this.button1.AutoSize = true;
        this.button1.Location = new System.Drawing.Point(270,
258);

        this.button1.Name = "button1";
        this.button1.Size = new System.Drawing.Size(75, 27);
        this.button1.TabIndex = 7;
        this.button1.Text = "Вхід";
        this.button1.UseVisualStyleBackColor = true;
        //
        // button2
        //
        this.button2.AutoSize = true;
        this.button2.Location = new System.Drawing.Point(15,
400);

        this.button2.Name = "button2";

```



```

        this.button2.Size = new System.Drawing.Size(75, 27);
        this.button2.TabIndex = 8;
        this.button2.Text = "Вийти";
        this.button2.UseVisualStyleBackColor = true;
        //
        // Form1
        //
        this.AutoScaleDimensions = new System.Drawing.SizeF(8F,
16F);
        this.AutoScaleMode =
System.Windows.Forms.AutoScaleMode.Font;
        this.ClientSize = new System.Drawing.Size(622, 435);
        this.Controls.Add(this.button2);
        this.Controls.Add(this.button1);
        this.Controls.Add(this.label4);
        this.Controls.Add(this.textBox2);
        this.Controls.Add(this.textBox1);
        this.Controls.Add(this.pictureBox1);
        this.Controls.Add(this.label3);
        this.Controls.Add(this.label2);
        this.Controls.Add(this.label1);
        this.Name = "Form1";
        this.Text = "Авторизація";

        ((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).EndInit();

        this.ResumeLayout(false);
        this.PerformLayout();

    }

    #endregion

    private System.Windows.Forms.Label label1;
    private System.Windows.Forms.Label label2;
    private System.Windows.Forms.Label label3;
    private System.Windows.Forms.PictureBox pictureBox1;
    private System.Windows.Forms.TextBox textBox1;
    private System.Windows.Forms.TextBox textBox2;
    private System.Windows.Forms.Label label4;
    private System.Windows.Forms.Button button1;
    private System.Windows.Forms.Button button2;
}

```

клас ServerControl

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;

```

```

using System.Threading.Tasks;
using System.Windows.Forms;

namespace CYBD
{
    public partial class FormServerControl : Form
    {
        Server srv = new Server();

        public FormServerControl()
        {
            InitializeComponent();

            private void buttonServerLaunch_Click(object sender,
EventArgs e)
            {
                Server.UpdateStatus();
                textBoxLog.Text += srv.SocketCreate();
                textBoxLog.Text += srv.StartListen();
                textBoxLog.Text += srv.AcceptConnection();
                textBoxLog.Text += srv.GetStream();
                textBoxLog.Text += srv.DataReceive();
                continueReceive();
            }

            public TextBox getTextBox()
            {
                return textBoxLog;
            }
        }
    }
}

```

клас ClientConnect.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

using System.Net;
using System.Net.Sockets;

namespace myClient
{
    public partial class Form1 : Form
    {
        TcpClient myClient;
    }
}

```

```

NetworkStream netStream;
string
    ip = "192.168.1.100", //127.0.0.1
    port = "5461";

public Form1()
{
    InitializeComponent();

    textBoxIP.Text = ip;
    textBoxPort.Text = port;
}

private void buttonCreateClientSocket_Click(object sender,
EventArgs e)
{
    myClient = new TcpClient();
    textBoxLog.Text += "Сокет создан" + "\r\n";
    buttonCreateClientSocket.Enabled = false;
}

private void buttonConnectToServerSocket_Click(object
sender, EventArgs e)
{
    try
    {
        IPAddress serverIP =
IPAddress.Parse(textBoxIP.Text);
        int serverPort = int.Parse(textBoxPort.Text);
        myClient.Connect(serverIP, serverPort);
        textBoxLog.Text += "Подключение к серверу " +
serverIP + ":" + serverPort + "\r\n";
        buttonConnectToServerSocket.Enabled = false;
        textBoxLog.Text += "Подключение к серверу выполнено
успешно" + "\r\n";
        buttonGetStream.Enabled = true;
        buttonDisconnect.Enabled = true;
    }
    catch (Exception ex)
    {
        textBoxLog.Text += "Ошибка " + ex.ToString() +
"\r\n";
        buttonConnectToServerSocket.Enabled = true;
    }
}

private void buttonGetStream_Click(object sender, EventArgs
e)
{
    try
    {
        netStream = myClient.GetStream();
        textBoxLog.Text += "Поток получен" + "\r\n";
        buttonGetStream.Enabled = false;
    }
}

```

```

        buttonSendData.Enabled = true;
    }
    catch (Exception ex)
    {
        textBoxLog.Text += "Ошибка " + ex.ToString() +
"\r\n";
    }
}

private void buttonSendData_Click(object sender, EventArgs
e)
{
    byte[] sendBytes =
Encoding.UTF8.GetBytes(textBoxDataToSend.Text + "\r\n");
    netStream.Write(sendBytes, 0, sendBytes.Length);
}

private void buttonDisconnect_Click(object sender,
EventArgs e)
{
    try
    {
        myClient.Close();
        textBoxLog.Text += "Клиентский сокет закрыт" +
"\r\n";

        buttonCreateClientSocket.Enabled = true;
        buttonConnectToServerSocket.Enabled = false;
        buttonGetStream.Enabled = false;
        buttonSendData.Enabled = false;
        buttonDisconnect.Enabled = false;
    }
    catch (Exception ex)
    {
        textBoxLog.Text += "Ошибка " + ex.ToString() +
"\r\n";
    }
}
}
}

```

```

клас ClientUpdate.cs
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Net;
using System.Net.Sockets;
using System.IO;

```

```

namespace Client
{

```

```

    public partial class Form1 : Form
    {
        // The TCP client will connect to the server using an IP
        and a port
        TcpClient tcpClient;
        // The file stream will read bytes from the local file you
        are sending
        FileStream fstFile;
        // The network stream will send bytes to the server
        application
        NetworkStream strRemote;

        public Form1()
        {
            InitializeComponent();
        }

        private void ConnectToServer(string ServerIP, int
        ServerPort)
        {
            // Create a new instance of a TCP client
            tcpClient = new TcpClient();
            try
            {
                // Connect the TCP client to the specified IP and
                port
                tcpClient.Connect(ServerIP, ServerPort);
                textBoxLog.Text += "Successfully connected to
                server\r\n";
            }
            catch (Exception exMessage)
            {
                // Display any possible error
                textBoxLog.Text += exMessage.Message;
            }
        }

        private void buttonConnect_Click(object sender, EventArgs
        e)
        {
            // Call the ConnectToServer method and pass the
            parameters entered by the user
            ConnectToServer(textBoxSeverIP.Text,
            Convert.ToInt32(textBoxPort.Text));
        }

        private void buttonDisconnect_Click(object sender,
        EventArgs e)
        {
            // Close connections and streams and update the log
            textbox
            tcpClient.Close();
            if (strRemote != null)
            {

```

```

        strRemote.Close();
    }
    if (fstFile != null)
    {
        fstFile.Close();
    }
    textBoxLog.Text += "Disconnected from server.\r\n";
}

private void buttonSendFile_Click(object sender, EventArgs
e)
{
    // If tclClient is not connected, try a connection
    if (tcpClient.Connected == false)
    {
        // Call the ConnectToServer method and pass the
parameters entered by the user
        ConnectToServer(textBoxSeverIP.Text,
Convert.ToInt32(textBoxPort.Text));
    }

    // Prompt the user for opening a file
    if (openFileDialog.ShowDialog() == DialogResult.OK)
    {
        textBoxLog.Text += "Sending file information\r\n";
        // Get a stream connected to the server
        strRemote = tcpClient.GetStream();
        byte[] byteSend = new
byte[tcpClient.ReceiveBufferSize];
        // The file stream will read bytes from the file
that the user has chosen
        fstFile = new FileStream(openFileDialog.FileName,
FileMode.Open, FileAccess.Read);
        // Read the file as binary
        BinaryReader binFile = new BinaryReader(fstFile);

        // Get information about the opened file
        FileInfo fInfo = new
FileInfo(openFileDialog.FileName);

        // Get and store the file name
        string FileName = fInfo.Name;
        // Store the file name as a sequence of bytes
        byte[] ByteFileName = new byte[2048];
        ByteFileName =
System.Text.Encoding.ASCII.GetBytes(FileName.ToCharArray());
        // Write the sequence of bytes (the file name) to
the network stream
        strRemote.Write(ByteFileName, 0,
ByteFileName.Length);

        // Get and store the file size
        long FileSize = fInfo.Length;
        // Store the file size as a sequence of bytes

```

```

        byte[] ByteFileSize = new byte[2048];
        ByteFileSize =
System.Text.Encoding.ASCII.GetBytes(FileSize.ToString().ToCharArray
());
        // Write the sequence of bytes (the file size) to
the network stream
        strRemote.Write(ByteFileSize, 0,
ByteFileSize.Length);

        textBoxLog.Text += "Sending the file " + FileName +
" (" + FileSize + " bytes)\r\n";

        // Reset the number of read bytes
        int bytesSize = 0;
        // Define the buffer size
        byte[] downBuffer = new byte[2048];

        // Loop through the file stream of the local file
        while ((bytesSize = fstFile.Read(downBuffer, 0,
downBuffer.Length)) > 0)
        {
            // Write the data that composes the file to the
network stream
            strRemote.Write(downBuffer, 0, bytesSize);
        }

        // Update the log textbox and close the connections
and streams
        textBoxLog.Text += "File sent. Closing streams and
connections.\r\n";
        tcpClient.Close();
        strRemote.Close();
        fstFile.Close();
        textBoxLog.Text += "Streams and connections are now
closed.\r\n";
    }
}

private void buttonExit_Click(object sender, EventArgs e)
{
    Application.Exit();
}
}

```