**ЗМІСТ**

[ВСТУП 3](#_Toc422262322)

[1. ЗАГАЛЬНА ХАРАКТЕРИСТИКА ІНФОРМАЦІЙНИХ ПРОЦЕСІВ У ПРЕДМЕТНІЇ ОБЛАСТІ 4](#_Toc422262323)

[1.1 Опис та системний аналіз предметної області 4](#_Toc422262324)

[1.2 Обгрунтування мети та визначення основних завдань дипломної роботи 14](#_Toc422262325)

[2. ВИБІР ТА ОБГРУНТУВАННЯ МЕТОДІВ ТА ЗАСОБІВ ДЛЯ РОЗРОБКИ СИСТЕМИ 15](#_Toc422262326)

[2.1 Вибір методів та засобів для реалізації інформаційного забезпечення системи 15](#_Toc422262327)

[2.2 Вибір методів та засобів для реалізації прикладного програмного забезпечення 18](#_Toc422262328)

[3. РОЗРОБКА СИСТЕМИ 22](#_Toc422262329)

[3.1 Розробка структури інформаційного забезпечення системи 22](#_Toc422262330)

[3.2 Розробка та реалізація прикладного програмного забезпечення 29](#_Toc422262331)

[4. РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ 36](#_Toc422262332)

[4.1 Тестування системи 36](#_Toc422262333)

[4.2 Вимоги до апаратного та програмного забезпечення 38](#_Toc422262334)

[4.2.1 Діаграма розміщення 38](#_Toc422262335)

[4.2.2 Вимоги до апаратного забезпечення 39](#_Toc422262336)

[4.2.3 Вимоги до програмного забезпечення 40](#_Toc422262337)

[4.2.4 Склад інсталяційного пакету для встановлення розробленої системи 41](#_Toc422262338)

[ВИСНОВКИ 42](#_Toc422262339)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 43](#_Toc422262340)

[Додаток А 45](#_Toc422262341)

[Додаток Б 51](#_Toc422262342)

# ВСТУП

У час інформаційних технологій майже кожне підприємство використовує в роботі готові інтелектуальні вирішення предметної області, тому напрямок розроблення інформаційних систем є перспективним.

Кожен університет, відповідно до закону України, повинен надавати житло. Поселенням студентів займається деканат та відповідальний за гуртожитки відділ з виховної роботи. За відсутності єдиної інформаційної системи цей процес виконується вручну. Такий підхід займає багато часу працівників університету, а також може призвести до втрати інформації. Виходячи з цих міркувань, розробка даної системи для університету є актуальною.

Метою роботи є створення інформаційної системи поселення студентів у гуртожитки студентського містечка.

Основними завданнями є:

1. розробити централізовану базу даних;
2. нормалізувати базу даних до третього рівня нормалізації;
3. забезпечити захист та цілісність даних;
4. розробити інтерфейсну частину системи, використовуючи обрану мову програмування.

У ході розробки системи треба провести аналіз предметної області, визначити методи та засоби реалізації бази даних та інтерфейсної частини. Вхідними документами системи є заява на поселення студента, вихідними документами є наказ про розподілення місць у студентських гуртожитках та наказ на поселення студентів. Виходячи з того, що вихідним документом є

наказ, потрібно реалізувати засоби формування вихідних документів.

1. **ЗАГАЛЬНА ХАРАКТЕРИСТИКА ІНФОРМАЦІЙНИХ ПРОЦЕСІВ У ПРЕДМЕТНІЙ ОБЛАСТІ**
   1. **Опис та системний аналіз предметної області**

Надання студентам житла є одним із завдань університету відповідно до закону України про вищу освіту. Поселення відбувається відповідно до наказу деканату факультетів університету та на основі заяви, що подається студентом.

Система повинна автоматизувати процес поселення, що дасть можливість виконувати роботу ефективніше, а саме скоротить час виконання, створить єдину базу обліку, формуватиме документи автоматично та розподілятиме студентів по гуртожитках.

Вхідним документом системи є заява на поселення студента до гуртожитку. Отримуючи заяву деканат повинен поселити студента, але перед поселенням потрібно перевірити можливість поселення, а саме відповідність до таких критеріїв:

* рейтинг навчання студента (мінімум 400 балів);
* місце проживання студента (не ближче 30км до місця розташування університету).

Після проходження перевірки визначається номер кімнати та номер гуртожитку де буде проживати студент. Розподілення по кімнатам відбувається на основі наказу про розподіл місць у гуртожитках, який надає відділ з виховної роботи. Під час розподілу враховуються назва факультету, на якому навчається студент, та стать. Після проходження вищезгаданих дій деканат отримує наказ на поселення студентів.

Відділ з виховної роботи визначає призначення кімнат та категорію осіб, які будуть проживати в даній кімнаті. Категорії осіб – це сукупність осіб, які відносяться до певного факультету.

Визначення проблем предметної області відбувається за допомогою CRC-карток, які представлені в таблиці 1 та таблиці 2.

Перша проблема предметної області.

Таблиця 1

|  |  |
| --- | --- |
| Проблема | Ручний розподіл та оформлення вихідних документів |
| Зачіпає | Працівників університету |
| Наслідком цього є | Навантаження, велика трата часу, допущення помилок |
| Успішне рішення | Автоматизований розподіл, та автоматичне оформлення вихідних документів |

Друга проблема предметної області.

Таблиця 2

|  |  |
| --- | --- |
| Проблема | Відсутність єдиної бази даних |
| Зачіпає | Працівників університету |
| Наслідком цього є | Витрата зайвого часу на збір даних, помилки при роботі |
| Успішне рішення | Створення централізованої бази даних |

Проведемо аналіз користувачів системи. У системі існують два основні користувачі: Деканат та Відділ виховної роботи. Деканат редагує дані про студентів, розподіляє студентів по гуртожитках, формує список поселених студентів,оформлює наказ на поселення та звіт. Відділ виховної роботи редагує дані про гуртожитки, розподіляє кімнати серед факультетів університету, оформлює наказ про розподіл місць у гуртожитках.

Профілі користувачів надають дані про тип, відповідальність, критерій успіху та опис користувачів. В табл.3 та табл.4 приведено профіль деканату та відділу виховної роботи .

Профіль користувача деканету.

Таблиця 3

|  |  |
| --- | --- |
| Представник | Деканат |
| Опис | Користувач системи, наділений правами доступу до даних в БД. Може переглядати, редагувати, записувати дані про студентів. Вносить зміни в дані про поселених студентів у гуртожитках |
| Тип | Користувач |
| Відповідальність | Надання студентам житла. |
| Критерій успіху | Полегшення і пришвидшення роботи |

Профіль користувача відділу виховної роботи

Таблиця 4

|  |  |
| --- | --- |
| Представник | Відділ виховної роботи |
| Опис | Користувач системи, наділений можливістю запису даних в БД та перегляду цих даних. |
| Тип | Користувач |
| Відповідальність | Контроль за станом гуртожитків та подання даних про гуртожитки |
| Критерій успіху | Полегшення роботи та пришвидшення звітування. |

Ключові потреби користувачів системи.

* Деканат – змога швидко формувати накази на поселення, полегшення контролю за документообігом, отримувати результати перевірки даних студентів на відповідність критеріям можливості поселення, автоматизований розподіл студентів по місцях .
* Відділ виховної роботи - полегшення роботи та пришвидшення звітування, контроль даних по гуртожитках.

Визначившись с користувачами системи та їх потребами, можна змоделювати взаємодію користувачів з системою за допомогою діаграми прецедентів (usecase).

Діаграма прецедентів – це діаграма, в якій моделюється відношення між акторами (користувачами) та прецедентами (діями). Моделювання системи здійснюється мовою UML.

UML – це мова графічного опису для моделювання систем в розробці програмного забезпечення.

В діаграмі прецедентів є два актори: деканат та відділ виховної роботи. Діаграмі прецедентів зображена на плакаті №3. На ній ми бачимо, що актор «Деканат» має взаємодію з прецедентами: «Авторизація», «Редагування даних», «Заселення», «Переселення», «Виселення» та «Оформлення наказів на поселення». Актор «Деканат» спочатку повинен пройти авторизацію, після чого він має можливість редагувати дані, заселити, переселити та виселити студентів. Коли «Деканат» взаємодіє з вищезгаданими прецедентами, результат дій надходить до прецедента «Оформлення наказів на поселення», що генерує звіт та відповідний наказ. Результат дій цього прецедента надходить до акторів «Деканат» та «Відділ виховної роботи». Прецеденти «Заселення», «Виселення», «Переселення» отримують дані з прецедента «Оформлення наказу про розподіл місць у гуртожитках». З даним прецедентом взаємодіє актор «Відділ виховної

роботи», який, проходячи авторизацію, редагує дані про розподіл місць (якщо у цьому є необхідність) в прецеденті «Редагування даних», після чого процес переходить в прецедент «Оформлення наказу про розподіл місць у гуртожитках», в якому власне і генеруються звіти та накази про розподіл.

Визначивши взаємодію користувачів з системою, потрібно змоделювати послідовність їхніх дій. Послідовність дій зображується на діаграмі послідовності.

Діаграма послідовності – діаграма, на якій показані взаємодії об’єктів системи, які впорядковані за часом їх прояву. Моделювання відбувається мовою UML.

Основними елементами діаграми послідовності є позначення об’єктів (прямокутники) та вертикальні лінії, що відображають плин часу під час діяльності об’єкта і стрілки, що показують виконання дій об’єктами.

На рис.1 показано діаграму послідовностей.



Рис.1 Діаграма послідовностей.

На даній діаграмі ми бачимо, що спочатку «Відділ виховної роботи» оформляє наказ про розподілення місць у гуртожитках, потім на основі даного наказу «Деканат» розподіляє студентів по місцях у гуртожитку, та оформляє наказ на поселення.

Для визначення основних функцій системи була розроблена діаграма функціональної декомпозиції.

Діаграма функціональної декомпозиції – це діаграма, в якій описуються основні функції системи шляхом розбиття системи. Розроблену діаграму можна розглянути на рис.2.



Рис.2 Діаграма функціональної декомпозиції.

Відповідно до діаграми можна виділити перший обов’язковий процес «Авторизація», від якого ідуть інші процеси, а саме для відділу виховної роботи:

* 1. «Редагування даних»;
  2. «Розподіл місць у гуртожитках»;

1.2.1 «Оформлення наказу на розподілення місць у гуртожитках».

Для деканату:

* 1. «Редагування даних»;
  2. «Заселення»;

1.3.1 «Оформлення наказу на поселення»;

1.4 «Переселення»;

1.4.1 «Оформлення наказу на переселення»

1.5 «Виселення»;

1.5.1 «Оформлення наказу на виселення».

Розглянемо дії в процесі «Заселення» на діаграмі діяльності, яка зображена на плакаті №4.

Діаграма діяльності – це UML-діаграма, яка показує складові частини деякої дії.

В процесі «Заселення» на діаграмі діяльності користувачу пропонується вибрати назву групи із запропонованого списку, ввести прізвище студента у відповідне поле для пошуку. Після цього формується список студентів відповідно до попередньо введених даних. Далі користувач має обрати студентів, які претендують на поселення відповідно до заяв. Система перевіряє вибраних студентів за 3 критеріями:

* чи є студент першокурсником або має достатній рівень балів рейтингу для поселення;
* чи не живе студент близько до університету.

Якщо дані студента відповідають всім вимогам, то студент заноситься до списку студентів на поселення, інші заносяться до списку студентів, які не можуть отримати житло у гуртожитках, та вказується причина. Отримуючи дані про студента, про розподіл місць у гуртожитках, система очікує на вибір користувача. Якщо користувач обирає автоматичне поселення, то система автоматично розподіляє студентів по гуртожиткам, або користувач вибирає із запропонованого системою списку номер гуртожитку та номер кімнати, і заселяє студентів у вибрану кімнату. Далі система перевіряє, чи стать студентів підходить призначенню (чоловіча/жіноча) цієї кімнати; якщо ні, то система пропонує вибрати іншу кімнату для заселення, або все-таки заселити студентів у цю кімнату, при чому кімната змінює призначення, та виселяє тих студентів, які там були поселенні. Після проходження цієї дії система перевіряє, чи вистачає місця в кімнаті для студентів. Якщо не вистачає, то система пропонує варіанти інших кімнат, в яких вистачає місць для поселення. По завершенню вищеописаних дій система формує дані про поселення, на основі яких формується наказ на поселення студентів до гуртожитків.

Далі будемо проводити аналіз системи за допомогою SADT методології . Розглянемо контекстну SADT- діаграму (рис.3), діаграму 0-рівня (рис.4), та діаграму 1-рівня процесу «Заселення» (рис.5).

SADT методологія – розроблена Дугласом Россом, відображає функціональну структуру об’єктів, дії що походять від них та зв’язки між діями.



Рис.3 Контекстна SADT-діаграма.

На даній діаграмі вхідними даними у систему є заява на поселення, дані про студента, дані про гуртожитки, логін та пароль. Вихідними даними є накази про розподіл місць у гуртожитках, поселення, переселення, виселення. Керує системою працівник університету, дії відбуваються згідно законів та розпорядження.



Рис.4 SADT-діаграма 0-рівня.

З SADT-діаграми 0-рівня випливає, що спочатку іде процес «Авторизації», до якого входять логін та пароль, і на виході отримуємо дані про користувача, який авторизувався. Якщо авторизувався користувач відділу виховної роботи, то далі розпочинається процес «Розподілення місць у гуртожитках», де на вхід ідуть дані про гуртожитки, та факультети, а на виході отримуємо дані про розподілення, та наказ про розподілення місць у гуртожитках. Коли авторизується користувач деканату, система переходить до процесів: «Заселення», «Переселення», «Виселення». У процесі «Заселення» вхідними даними є дані про розподіл місць у гуртожитках, дані про студентів,

рейтинг, місця проживання та заява на поселення. Вихідними даними є наказ про поселення, звіт та дані про поселення, які ідуть до процесів «Переселення» та «Виселення» . З процесів «Переселення» та «Виселення» вихідними даними є накази, відповідні до процесів, та звіти.

 Рис.5 SADT-діаграма 1-рівня процесу «Заселення».

На рисунку 5 зображена SADT-діаграма 1-рівня. Ця діаграма описує процес «Заселення». Спочатку у вищезгаданому процесі деканат реєструє заяву на поселення: вхідними даними є дані про студента та дані заяви, вихідними даними є дані про місце проживання студента та його рейтинг. Вихідні дані з процесу «Реєстрація заяви» надходять на вхід процесу «Перевірка даних», де відповідно до критеріїв перевірки перевіряються дані про студента. В результаті перевірки створюються два списки:

* студенти, які підлягають поселенню;
* студенти, які не підлягають поселенню.

Відповідно до списку студентів, які підлягають поселенню, у третьому процесі визначаються місця у гуртожитку згідно наказу про розподіл місць у гуртожитках, в які можна заселити студентів. Завершальним етапом є процес «Заселення студентів». Заселення відбувається на основі вихідних даних з процесів «Перевірка даних» та «Визначення місця у гуртожитках». В результаті отримуємо дані про поселення, звіт, та наказ про поселення студентів. Всі вищезгадані дії відбуваються згідно Закону України про освіту та Розпоряджень університету.

## 1.2 Обгрунтування мети та визначення основних завдань дипломної роботи

Метою дипломної роботи є створення «Інформаційної системи поселення студентів у гуртожитки студентського містечка», яка полегшить роботу працівників університету, забезпечить цілісність та захист даних.

Загалом «Інформаційна система» - це комунікаційна система, що забезпечує збирання, пошук, оброблення та пересилання інформаціїдля зaбезпечення інформацiйних потреб кoристувачів.

При аналізі предметної визначено, що в системі є два типи користувачів, які мають виконувати різні функції. Треба створити базу даних для системи, та надати доступ до даних для цих типів користувачів у прикладному програмному додатку.

Система повинна виконувати наступні функції:

* авторизація у системі;
* заселення студентів;
* переселення студентів;
* виселення студентів;
* редагування даних;
* формування звітів;
* оформлення вихідних документів;
* перевірку даних.

1. **ВИБІР ТА ОБГРУНТУВАННЯ МЕТОДІВ ТА ЗАСОБІВ ДЛЯ РОЗРОБКИ СИСТЕМИ**
   1. **Вибір методів та засобів для реалізації інформаційного забезпечення системи**

БД (База даних) - набір логічно взаємопов'язаних даних, що використовується спільно, та призначені для задоволення інформаційних потреб користувачів. Дані записують у таблиці, де стовпці (атрибути) характеризують дані, а рядки саме містять дані. При розробці цієї системи використовується реляційна база даних.

СУБД (Система управління базами даних) - це програмне забезпечення, за допомогою якого користувачі можуть керувати даними у БД та самою БД, а також здійснювати контрольований доступ до них.

Основні характеристики СУБД:

* підвищений рівень безпеки;
* контроль за надлишковістю даних;
* непротирічність даних;
* підтримка цілісності бази даних (коректність та непротирічність);
* спільне використання даних;

Можливості СУБД:

* дозволяє створювати БД (здійснюється за допомогою мови визначення даних DDL (Data Definition Language));
* дозволяє додавання, оновлення, видалення та читання інформації з БД (за допомогою мови маніпулювання даними DML, яку часто називають мовою запитів);
* можна надавати контрольований доступ до БД за допомогою:
  1. системи забезпечення захисту, яка запобігає несанкціонованому доступу до БД;
  2. системи керування паралельною роботою прикладних програм, яка контролює процеси спільного доступу до БД;
  3. система відновлення - дозволяє відновлювати БД до попереднього непротирічного стану, що був порушений в результаті збою апаратного або програмного забезпечення.

Транзакція - це набір дій, що виконуються користувачами або прикладною програмою з метою зміни вмісту БД.

На сьогоднішній день існує багато різних СУБД. Серед лідерів цього ринку можна виділити такі СУБД: ORACLE, DB2, SQL Server, MySQL. Серед вищезгаданих систем управління для розробки бази даних було вибрано Microsoft SQL Server.

Оскільки система, для якої створюється база даних не є web-орієнтованою, повинна працювати з великою кількістю даних та надавати можливість користуванню декільком користувачам, то одразу можна зробити висновок, що Mysql не підходить для виконання таких завдань.

Oracle має перевагу перед іншими СУБД в тому, що він швидко обробляє велику кількість даних, та володіє високим рівнем масштабування.

Однак серед недоліків у нього можна виділити наступне:

* висока ціна;
* потреба в потужному обладнанні для оптимальної роботи;
* потреба у високоспеціалізованому персоналу для підтримки роботи;
* складнощі в навчанні функціоналу;

Отже, дивлячись на наявні недоліків ORACLE не підходить для розробки бази даних.

Серед переваг Sql Server - великий пакет корисних інструментів,

стабільна робота с базами даних, менші затрати на адміністрування. Серед недоліків - Sql Server працює тільки на платформах Windows. Через вищезгадані переваги та той факт, що система розроблюватиметься під платформу Windows, було обрано для розробки бази даних SQL SERVER. Вікно графічного інтерфейсу зображено на рисунку 6.

При створенні бази даних, маніпулювання даними, та надання доступу до них використовувалась мова запитів SQL.

Мова SQL, призначена для взаємодії з базами даних, з'явилася в середині 70-х рр. (перші публікації датуються 1974 р.) і була розроблена у компанії IBM у рамках проекту експериментальної реляційної СУБД System R. Вихідна назва мови SEQUEL (Structured English Query Language) тільки частково відбиває суть цієї мови. Звичайно, мова була орієнтований головним чином на зручну й зрозумілу користувачам формулювання запитів до реляційних БД. Але, у дійсності, вона майже із самого початку була повною мовою БД, що забезпечує крім засобів формулювання запитів і маніпулювання БД наступні можливості:

* засоби визначення й маніпулювання схемою БД;
* засоби визначення обмежень цілісності й тригерів;
* засоби визначення подань БД;
* засоби визначення структур фізичного рівня, що підтримують ефективне виконання запитів;
* засоби авторизації доступу до відносин й їхніх полів;
* засоби визначення крапок збереження транзакції, і виконання фіксації й відкотів транзакцій.

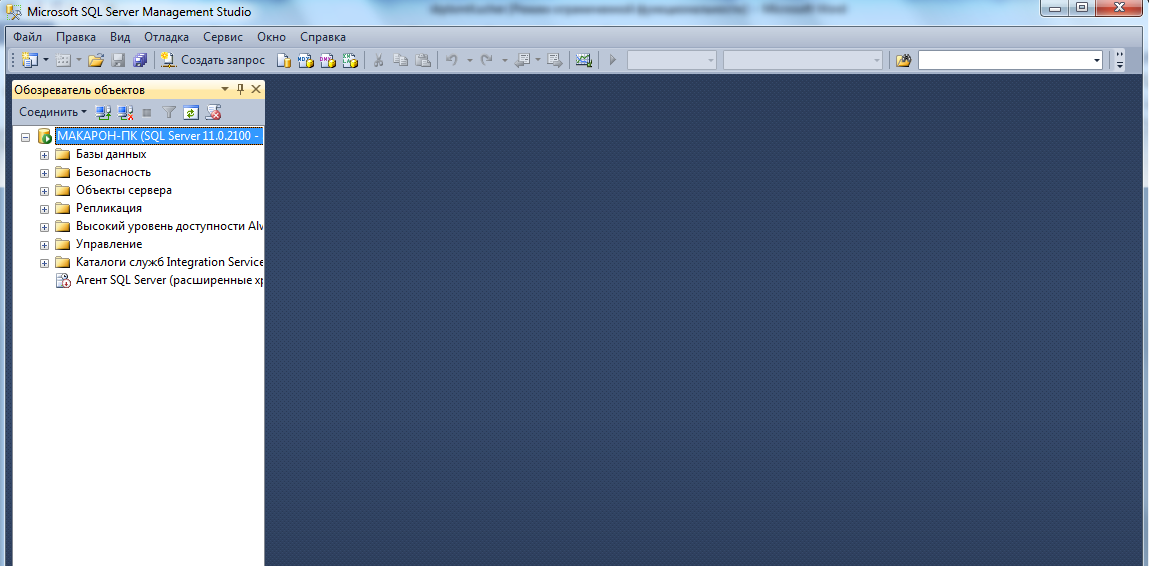


Рис.6 Вікно інтерфейсу SQL Server Management Studio.

* 1. **Вибір методів та засобів для реалізації прикладного програмного забезпечення**

Прикладне програмне забезпечення розроблялось мовою C++, у середовищі Borland C++ Builder 6.

C++ — це універсальна мова програмування високого рівня з підтримкою таких парадигм програмування: об'єктно-орієнтованої, процедурної та узагальненої. Розроблена Б'ярном Страуструпом (Bjarne Stroustrup) у 1979 році та названа «Сі з класами». Страуструп перейменував мову у C++ у 1983 р. Базується на мові Сі. У 1990-х роках С++ стала однією з найуживаніших мов програмування загального призначення.

При створенні С++ прагнули зберегти сумісність з мовою С. Більшість програм на С справно працюватимуть і з компілятором С++. С++ має синтаксис, заснований на синтаксисі С.

Нововведеннями С++ порівняно з С є:

* підтримка об'єктно-орієнтованого програмування через класи;
* підтримка узагальненого програмування через шаблони;
* доповнення до стандартної бібліотеки;
* додаткові типи даних;
* обробка винятків;
* простори імен;
* вбудовані функції;
* перевантаження операторів;
* перевантаження імен функцій;
* посилання і оператори управління вільно розподіленою пам'яттю.

Borland C++ Builder 6 – це програмний продукт, середовище програмування для розробки програмного забезпечення мовою С++. Включає в себе компілятор, об'єктні бібліотеки, редактор коду, зневаджувач.

На рисунку 7 показано інтерфейс Borland C++ Builder .

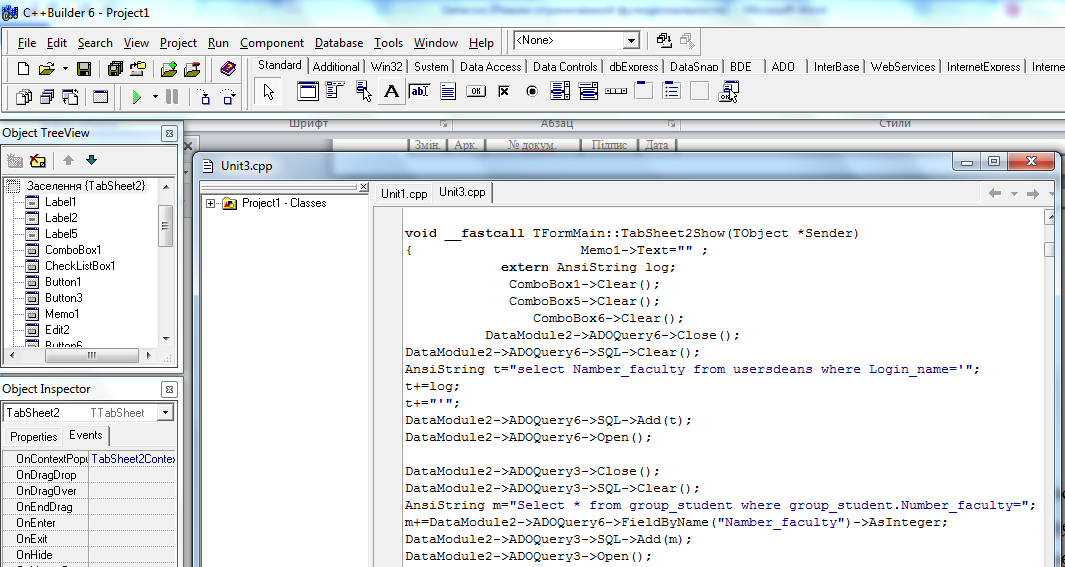


Рис.7 Вікно інтерфейсу Borland C++ builder.

Під час розробки системи використовувались наступні компоненти:

* DBNavigator.
* DBGrid.
* DateTimePicker.
* PageControl.
* Memo.
* Button.
* ListBox.
* CheckListBox.
* ComboBox.
* Label.
* Edit.
* Data Source.
* ADO компоненти.
  + ADO Query.
  + ADO Connection.

Для оформлення наказів у системі використовується компонент FastReport.

FastReport - це компонент для побудови звітів, що являє собою сполучення графічного редактора, генератора і Preview звітів. За можливостями

приблизно відповідає ReportBuilder 5.хх.

На рисунку 8 зображено графічний редактор компонента FastReport.

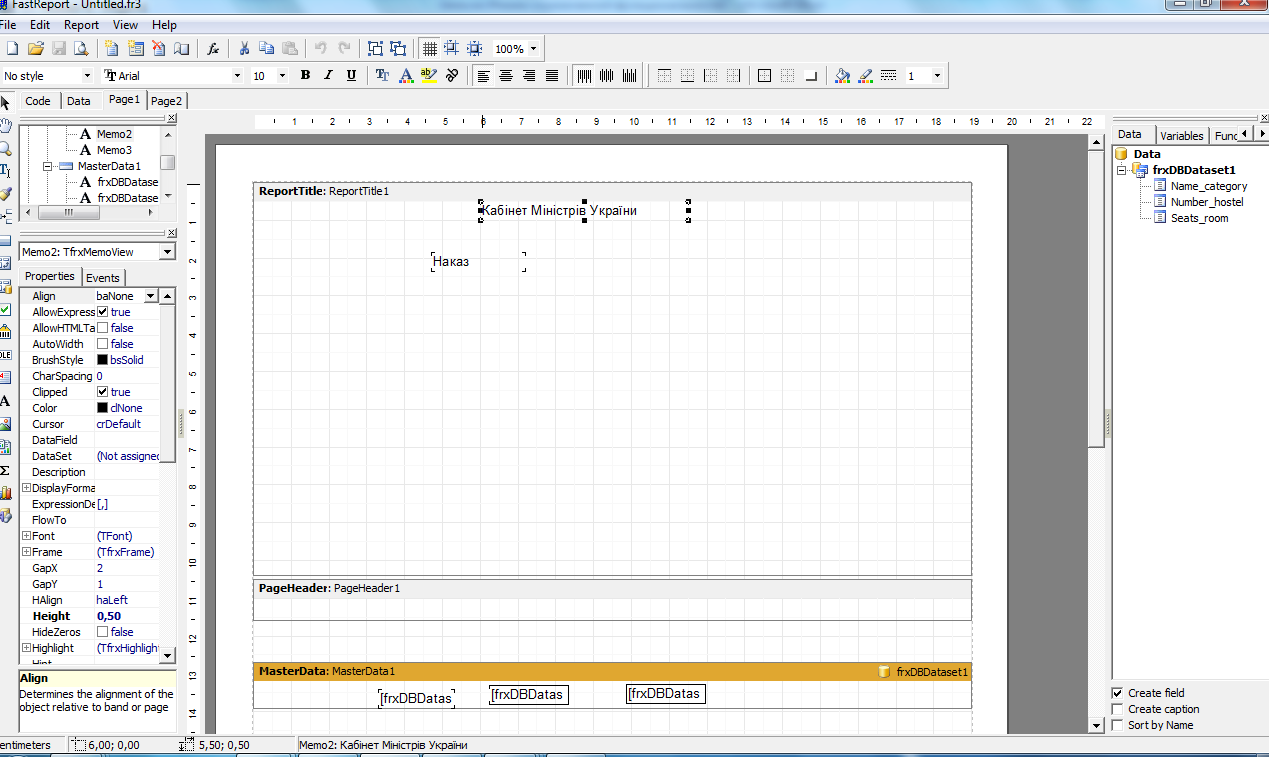
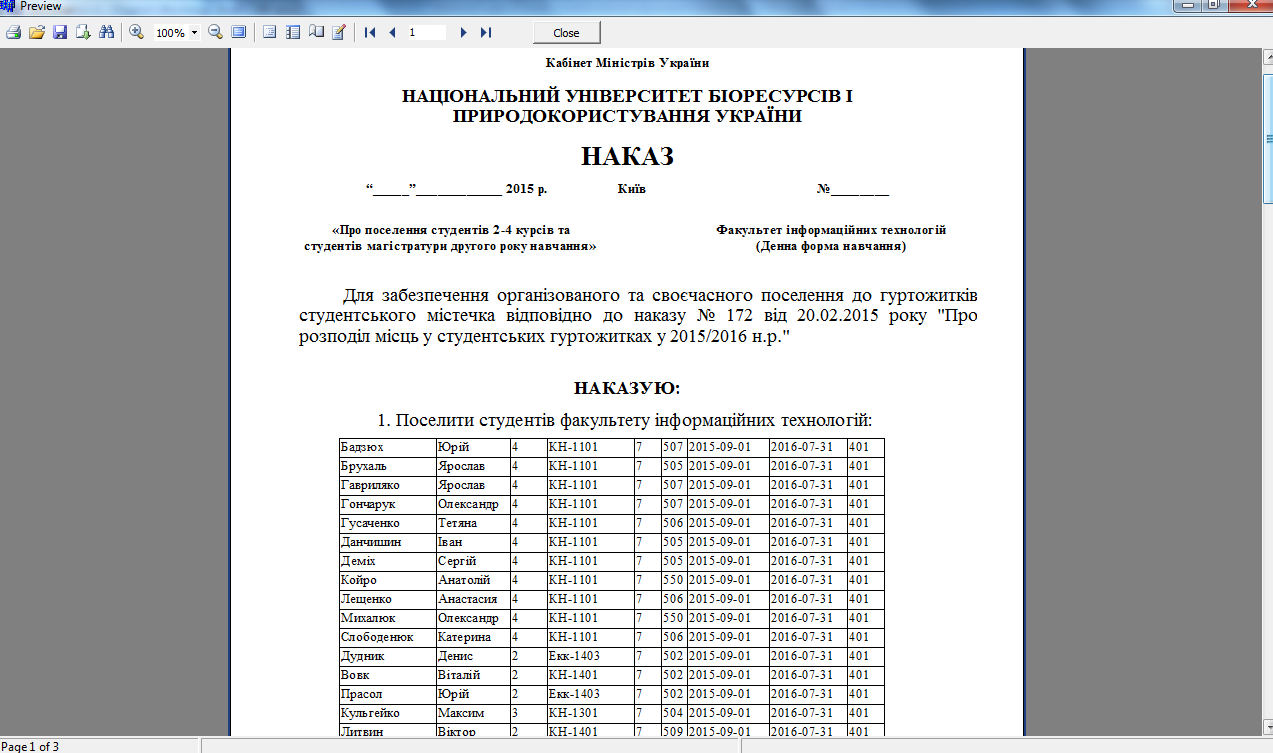


Рис.8 Графічний редактор компонента FastReport.

Приклад наказу про поселення студентів зображено на рисунку 9.

Рис.9 Наказ про поселення студентів у FastReport.



1. **РОЗРОБКА СИСТЕМИ**
   1. **Розробка структури інформаційного забезпечення системи**

Модель "сутність-зв'язок" – це модель даних, яка використовується для проектування різних моделей (баз даних, інформаційних систем, архітектур комп'ютерних додатків та інших систем) і є концептуальною високорівневою моделлю. Модель є графічною нотацією і ґрунтується на певній семантичній важливій інформації про реальний світ; за допомогою неї можна описувати власне об'єкти логічних моделей даних і зв’язки між об'єктами. У даному контексті модель "сутність-зв'язок" є метамоделлю даних, тобто засобом специфікації логічних моделей даних, які є побудовані на основі вихідної концептуальної моделі даних.

Модель "сутність-зв'язок" використовують для отримання концептуальної моделі у разі концептуального моделювання, яку потім транслюють у логічні моделі, зазвичай об'єктно-орієнтовані або реляційні. Модель "сутність-зв'язок" була запропонована П. Ченом для впорядкування задачі проектування моделей, її проект опубліковали у 1976 р.. Дана модель задовольняє дві важливі умови:

* структура різноманітних предметних областей досить адекватно може бути описаною завдяки потужність її засобів;
* не надто великий розрив між можливостями моделі та CASE-засобів (Computer Aided Software Engineering1 ), що її підтримують.

Станом на сьогодні немає єдиного стандарту, загальноприйнятого для моделі "сутність-зв'язок", проте існує набір загальних конструкцій, які знаходяться в основі більшості з її варіантів. Через існування пропозицій різних авторів власних елементів моделі та відповідної термінології сьогодні і виникла така ситуація.

Між типами сутностей, зв'язків та атрибутами немає абсолютної різниці. Атрибут є власне атрибутом лише у тому разі, якщо він пов'язаний із деяким типом сутності або зв'язку, проте в іншому контексті він може існувати як самостійний тип сутності. Через цю неоднозначність робота під час злиття локальних моделей в єдину глобальну (де вказані методи злиття локальних моделей (ідентичність, агрегація, узагальнення) і шляхи подолання неоднозначних ситуацій, що виникають під час злиття) суттєво ускладнена.

Побудови ER-діаграми за допомогою програмного продукту Erwin Data Modeler було побудовано логічну модель даних , що зображено на плакаті №1.

Огляд сутностей логічної моделі бази даних:

1. Сутність «Студент», визначає дані про студента, має три не ідентифікуючі зв’язки за сутностями «Паспортні дані» , «Група» та «Рейтинг», один індетифікуючий зв’язок з сутністю «Список поселених студентів» та первинний ключ «Ідентифікаційний код», атрибути:

* «Ідентифікаційний код» (числового типу, PK);
* «І’мя» (символьного типу);
* «Прізвище» (символьного типу);
* «Дата вступу» (типу дата);
* «Серія та номер паспорта» (символьного типу, FK);
* «Назва групи» (символьного типу, FK).

1. Сутність «Група», визначає дані про групу, що використовується для створення списку студентів групи, первинний ключ «Назва групи», зовнішній ключ «Номер факультету», що має зв’язок з сутністю «Факультет»:

* «Назва групи» (символьного типу, PK);
* «Кількість студентів» (числового типу);
* «Куратор групи» (символьного типу);
* «Староста групи» (символьного типу);
* «Номер факультету» (числового типу,FK).

1. Сутність «Факультет» має неідентифікуючий зв’язок з сутністю «Група» та ідентифікуючий зв'язок з сутністю «Категорія осіб», первинний ключ

«Номер факультету», зовнішніх ключів немає. Визначає дані факультету:

* «Номер факультету» (числового типу, PK);
* «Назва факультету» (символьного типу);
* «Телефон факультету» (числового типу);
* «Декан» (символьного типу).

1. Сутність «Гуртожиток» має ідентифікуючий зв’язок з сутністю «Кімната». Зберігає дані про гуртожиток та визначає факультет до якого відносится гуртожиток:

* «Номер гуртожитку (числового типу, PK);
* «Телефон» (числового типу);
* «Комендант» (символьного типу);
* «Адреса» (символьного типу).

1. Сутність «Кімната» визначає дані про кімнату, має ідентифікуючий зв'язок з сутністю «Гуртожиток» та два неідентифікуючих зв’язки з сутностями «Категорія осіб» та «Призначення». Має композитний ключ, що складається з двох первинних ключів «Номер кімнати» та «Номер гуртожитку»:

* «Номер кімнати» (числового типу, PK);
* «Поверх» (числового типу);
* «Номер гуртожитку (числового типу, PK,FK);
* «Кількість місць» (числового типу);
* Стать (символьного типу);
* Номер призначення (числового типу, FK);
* Номер категорії (числового типу, FK);
* «Кількість вільних місць» (числового типу).

1. Сутність «Тип кімнати» визначає тип кімнати, зокрема стать, тобто якої

статі студенті там проживають:

* «id типу» (числового типу, PK);
* «Тип» (символьного типу);
* «Стан» (символьного типу);
* «Кількість місць» (числового типу).

1. Сутність «Місце в гуртожитку» містить дані про місця в кімнатах:

* «Номер місця» (числового типу, PK);
* «Дата заселення» (типу дата);
* «Дата виселення» (типу дата);
* «Номер кімнати» (числового типу, FK).

1. Сутність «Паспортні дані» зберігає паспортні дані про студента, має не ідентифікуючий зв’язок з сутністю «Студент»:

* «Серія та номер паспорта» (символьного типу, PK);
* «Ким виданий» (символьного типу);
* «Коли виданий» (типу дата);
* «Місце проживання» (символьного типу);
* «Стать» (символьного типу).

1. Сутність «Рейтинг» слугує для визначення рейтингу студента та має не ідентифікуючий зв’язок з сутністю «Студент»:

* «id рейтинга» (числового типу, PK);
* «Курс» (числового типу);
* «Дата оцінювання» (типу дата);
* «Рейтинг» (числового типу);
* «Ідентифікаційний код» ( числового типу, FK).

1. Сутність «Список поселених студентів» збирає дані про поселення у процесі роботи системи. Ця сутність має первинний ключ «Ідентифікаційний код» та неідентифікуючий зв’язок з сутністю

«Студент»:

* «Номер кімнати» (числового типу);
* «Номер гуртожитка» (числового типу);
* «Ідентифікаційний код» (числового типу, PK);
* «Дата заселення» (типу дата);
* «Дата виселення» (типу дата).

1. Сутність «Категорія осіб» містить дані про категорії осіб, до яких належить кімната. Має первинний ключ «Номер категорії» та не ідентифікуючий зв’язок з сутністю «Кімната» :

* «Номер категорії» (числового типу, PK);
* «Назва категорії» (символьного типу).

1. Сутність «Призначення» містить дані про призначення кімнати. Має первинний ключ «Номер призначення» та не ідентифікуючий зв’язок з сутністю «Кімната» :

* «Номер призначення» (числового типу, PK);
* «Призначення» (символьного типу).

1. Сутність «Користувач деканату» ця сутність не має зв’язків, слугує для визначення до якого факультету відноситься користувач деканату:

* «Логін» (символьного типу, PK);
* «Номер факультету» (символьного типу).

Для створення таблиці у середовищі Sql Server Management Studio можна використати графічний інтерфейс (рис.10),

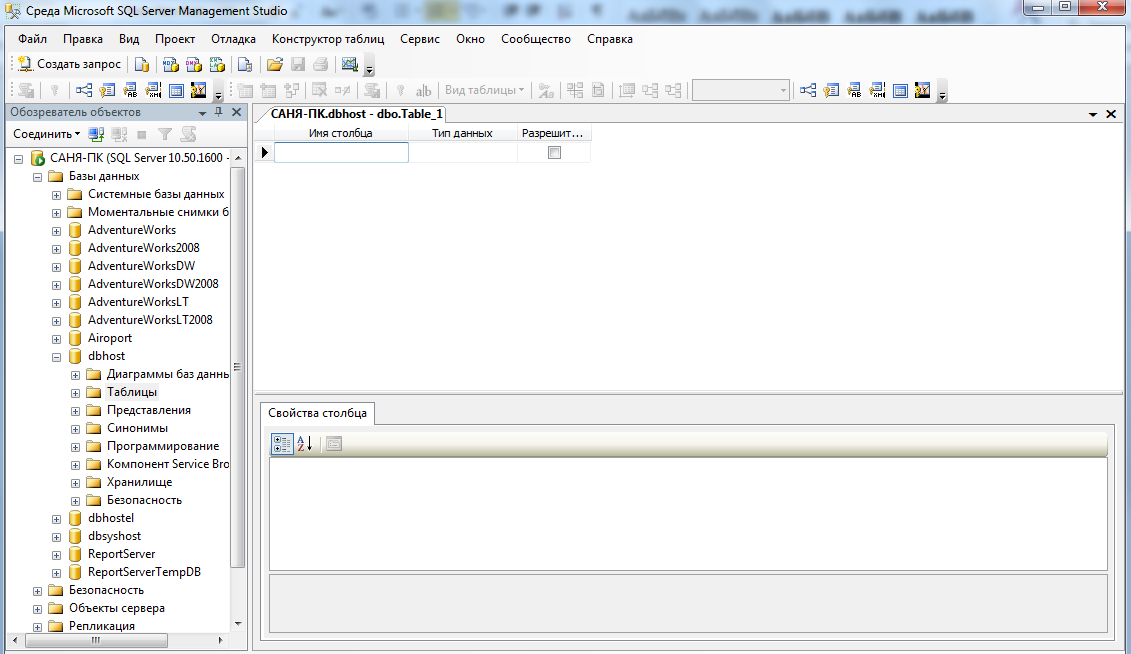


Рис.10 Створення таблиць бази даних в графічному інтерфейсі.

або написати запит на сворення (приклад запиту показаний у додатку А).

Фізична модель бази даних системи показана на рисунку 11.



Рис.11 Фізична модель бази даних.

Уявлення – це тип таблиці, у якій дані обираются з інших різних таблиць.

Уявлення створюється такими самими методами, як при створенні таблиці. В дипломній роботі використовувався метод створення через графічний інтерфейс.

Спочатку треба натиснути пункт «Створити уявлення». Відкривається вікно вибору таблиць (рис.12).

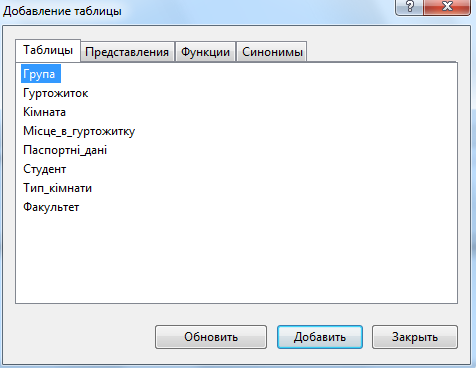


Рис.12. Вікно вибору таблиць

Після вибору таблиць відкривається вікно вибору атрибутів таблиць (рис.13).

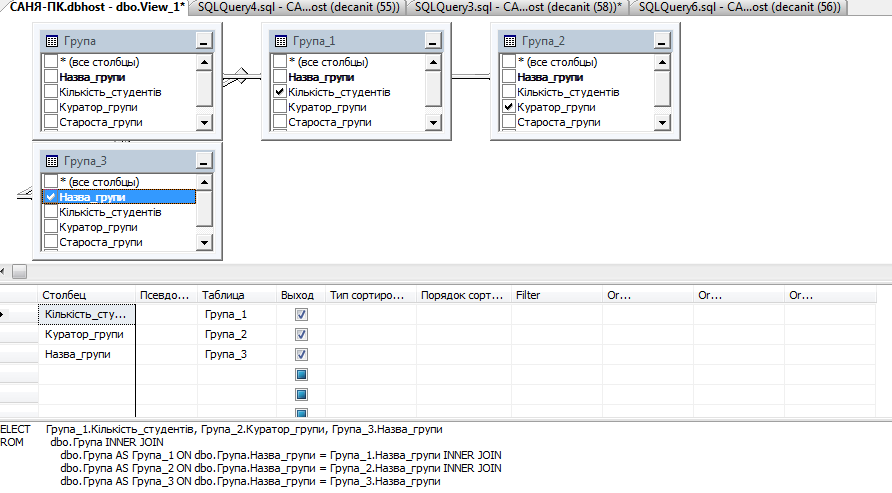


Рис.13 Вікно атрибутів таблиць.

По закінчені натискаємо зберегти, даємо назву уявленню. Потім дане уявлення з’являється в структурі бази даних.

У базі даних створено чотири уявлення:

* «View\_rating»;
* «View\_rating\_change»;
* «View\_settlement»;
* «View\_student».
  1. **Розробка та реалізація прикладного програмного забезпечення**

Після проведення аналізу системи та побудови бази даних розпочинається розробка прикладного програмного забезпечення. Для початку треба встановити з’єднання програмного додатку з базою даних. З’єднання встановлюється за допомогою компонента ADOConnection у формі DataModule. Приклад форми з компонентом ADO зображений на рисунку 14.

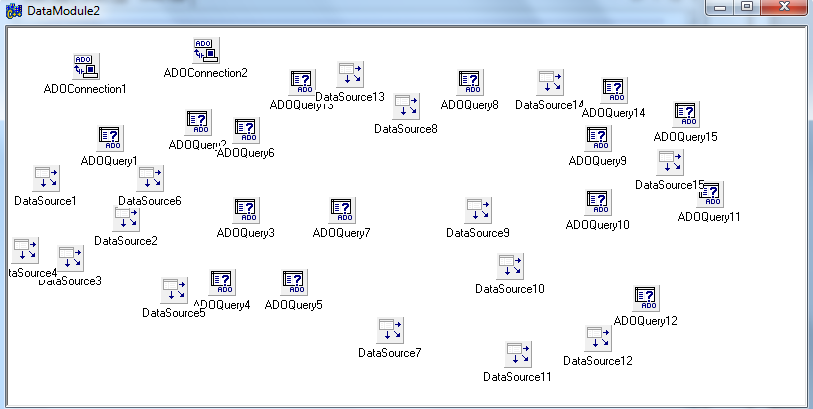


Рис.14 Форма DataModule з компонентами ADO.

У компоненті ADOConnection треба прописати рядок з’єднання з базою даних Сonnection String, приклад:

Provider=SQLOLEDB.1;Password=1234;Persist Security Info=True;User ID=admindean;Initial Catalog=test;Data Source=МАКАРОН-ПК;Use Procedure for Prepare=1;Auto Translate=True;Packet Size=4096;Workstation ID=МАКАРОН-ПК;Use Encryption for Data=False;Tag with column collation when possible=False

Наступним етапом є встановлення компонента ADOQuery для можливості робити запити до бази даних. В ньому треба написати рядок запиту. Приклад такого рядка:

select Name\_category, room.Number\_hostel, Sum(Seats) as Seats\_room from category, room

where category.Number\_category=room.Number\_category

group by Name\_category, room.Number\_hostel order by Name\_category.

Для доступу до ADOQuery необхідно встановити компонент DataSourse, та визначити в ньому DataSet, тобто ADOQuery до якого треба отримати доступ.

Встановивши зв'язок, переходимо до створення вікна авторизації (рис. 15) та пишемо код для події входу у систему.

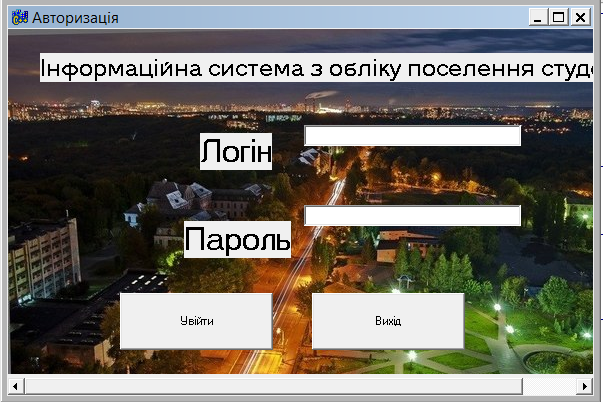


Рис.15 Форма Авторизації.

Приклад коду для авторизації у системі, що дає підключення до бази даних:

void \_\_fastcall TFormEnter::BenterClick(TObject \*Sender)

{

AnsiString conndb;

AnsiString login;

AnsiString pass;

conndb="";

log=Elog->Text;

login=Elog->Text;

pass=Epass->Text;

Підключення до бази даних, якщо входить працівник відділу виховної роботи:

if(login=="admindep") {

conndb+="Provider=SQLOLEDB.1;Password=";

conndb+=pass;

conndb+=";Persist Security Info=True;User ID=";

conndb+=login;

conndb+=";Initial Catalog=test;Data Source=МАКАРОН-ПК;Use Procedure for Prepare=1;Auto Translate=True;Packet Size=4096;Workstation ID=МАКАРОН-ПК;Use Encryption for Data=False;Tag with column collation when possible=False";

DataModule2->ADOConnection2->Connected=false;

DataModule2->ADOConnection2->ConnectionString=conndb;

try{

DataModule2->ADOConnection2->Connected=true;

FormM->ShowModal();

FormEnter->Close();

}

catch(...){ MessageDlg("error", mtError, TMsgDlgButtons() << mbOK, 0);

return;

}}

Підключення до бази даних, якщо входять інші користувачі системи:

else {

conndb+="Provider=SQLOLEDB.1;Password=";

conndb+=pass;

conndb+=";Persist Security Info=True;User ID=";

conndb+=login;

conndb+=";Initial Catalog=test;Data Source=МАКАРОН-ПК;Use Procedure for Prepare=1;Auto Translate=True;Packet Size=4096;Workstation ID=МАКАРОН-ПК;Use Encryption for Data=False;Tag with column collation when possible=False";

DataModule2->ADOConnection1->Connected=false;

DataModule2->ADOConnection1->ConnectionString=conndb;

try{

DataModule2->ADOConnection1->Connected=true;

FormMain->ShowModal();

FormEnter->Close();

}

catch(...){ MessageDlg("error", mtError, TMsgDlgButtons() << mbOK, 0);

return;

}}

}

Навігація по пунктам меню системи зроблена за допомогою компонента PageControl (рис.16).

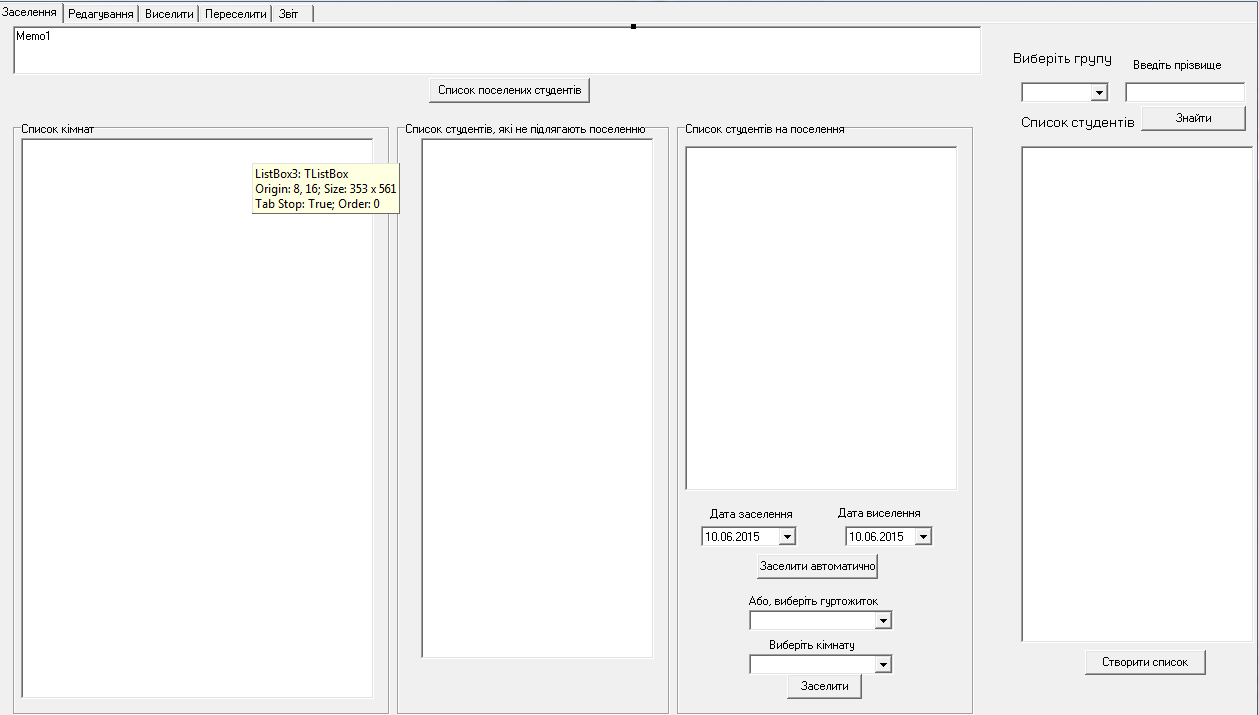
Рис.16 Оформлення меню системи.



Розглянемо розробку процесу заселення. Форма заселення має два компоненти CheckListBox, чотири кнопки (Button), два компоненти ListBox,

три компоненти Combobox та одне поле для введення (Edit). Форма процесу заселення зображена на рисунку 17.

Рис.17 Форма процесу заселення.



Розглянемо код програми, який виводить список груп студентів у ComboBox, відповідно до факультету, від якого зайшов користувач

Код програми:

extern AnsiString log;// Отримання значення глобальної змінної, яка містить дані логіна, під яким увійшов користувач.

ComboBox1->Clear(); // Очищення компонента «ComboBox».

DataModule2->ADOQuery6->Close();// Закриття запиту «ADOQuery6».

DataModule2->ADOQuery6->SQL->Clear();//Очищення запиту «ADOQuery6».

AnsiString t="select Namber\_faculty from usersdeans where Login\_name='";

t+=log;

t+="'";// Створення нової змінної «t» типу AnsiString, та присвоєння їй рядка нового запиту до бази даних, де отримується значення номера факультету з таблиці користувачі деканату, де логін дорівнює значенню глобальної змінної.

DataModule2->ADOQuery6->SQL->Add(t);// Додавання запиту до «ADOQuery6».

DataModule2->ADOQuery6->Open();// Відкриття запиту.

DataModule2->ADOQuery3->Close();//Закриття запиту «ADOQuery3».

DataModule2->ADOQuery3->SQL->Clear();//Очищення запиту «ADOQuery3».

AnsiString m="Select \* from group\_student where group\_student.Number\_faculty=";

m+=DataModule2->ADOQuery6->FieldByName("Namber\_faculty")->AsInteger;// Створення нової змінної «m» типу AnsiString, та присвоєння їй рядка нового запиту до бази даних, де отримується значення з таблиці група, де номер факультету дорівнює значенню номера факультету із запиту «ADOQuery6».

DataModule2->ADOQuery3->SQL->Add(m); );// Додавання запиту до «ADOQuery3».

DataModule2->ADOQuery3->Open();Відкриття запиту.

while(!DataModule2->ADOQuery3->Eof){// Створення циклу, ітерації відбуваються, поки не закінчаться рядки вибірки запиту «ADOQuery3».

ComboBox1->Items->Add(DataModule2->ADOQuery3->FieldByName("Name\_group")->AsString); // присвоєння компоненту «ComboBox1» значень атрибута «Name\_group» із рядків вибірки запиту «ADOQuery3».

DataModule2->ADOQuery3->Next(); // Перехід на наступний рядок вибірки.

}

DataModule2->ADOQuery3->Close();//Закриття запиту «ADOQuery3».

Далі розробляється код виводу списку студентів у компонент CheckListBox, приклад коду:

CheckListBox1->Clear();

AnsiString s="";

s+="select \* from Student1 where Name\_group='";

s+=ComboBox1->Text;

s+="'";

DataModule2->ADOQuery3->Close();

DataModule2->ADOQuery3->SQL->Clear();

DataModule2->ADOQuery3->SQL->Add(s);

DataModule2->ADOQuery3->Open();

while(!DataModule2->ADOQuery3->Eof){

CheckListBox1->Items->Add(DataModule2->ADOQuery3->FieldByName("second\_name")->AsString+" "+DataModule2->ADOQuery3->FieldByName("name")->AsString+" | "+DataModule2->ADOQuery3->FieldByName("Name\_group")->AsString+" ");

DataModule2->ADOQuery3->Next();

}

DataModule2->ADOQuery3->Close();

}

Після цього розробляється код для події натиснення на кнопку «Створити список». Текст коду міститься у додатку Б.

Алгоритм та зображення виконання алгоритму події натиснення на кнопку «Заселити автоматично» показано на плакаті №2.

# РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ

## Тестування системи

Одним з найважливіших етапів є тестування системи, адже під час тестування ми можемо виявити недоліки системи та вчасно їх виправити. Тестування системи проводиться на етапі впровадження. У разі виявлення недоліків (помилок), проводиться робота з виправлення, після чого впровадження продовжується.

Протестуємо процес системи «Переселення». Запускаємо програму. Відкривається вікно «Авторизації» (рис.15), після проходження авторизації відкривається головне вікно програм (рис.18). Переходимо на вкладку переселення, відкривається вікно «Переселення» (рис.18).

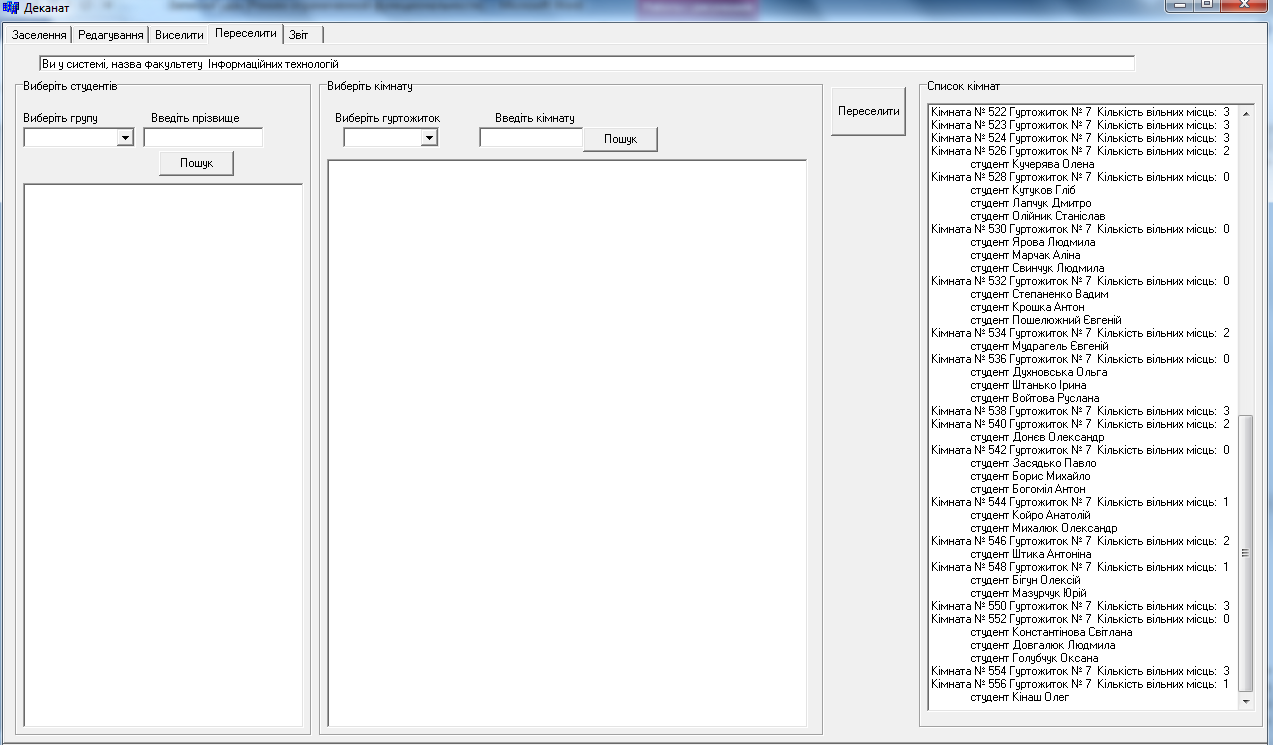


Рис.18 Вікно процесу переселення.

У цьому вікні ми бачимо список кімнат, де в 544 кімнаті поселені студенти Койро Анатолій та Михалюк Олександр, при цьому кількість вільних місць в цій кімнаті - 1. Далі з випадаючого списка обираємо групу в якій навчаються студенти, яких ми хочемо переселити; у нашому випадку з'являються тільки студенти, які поселені в гуртожитку (рис.19).

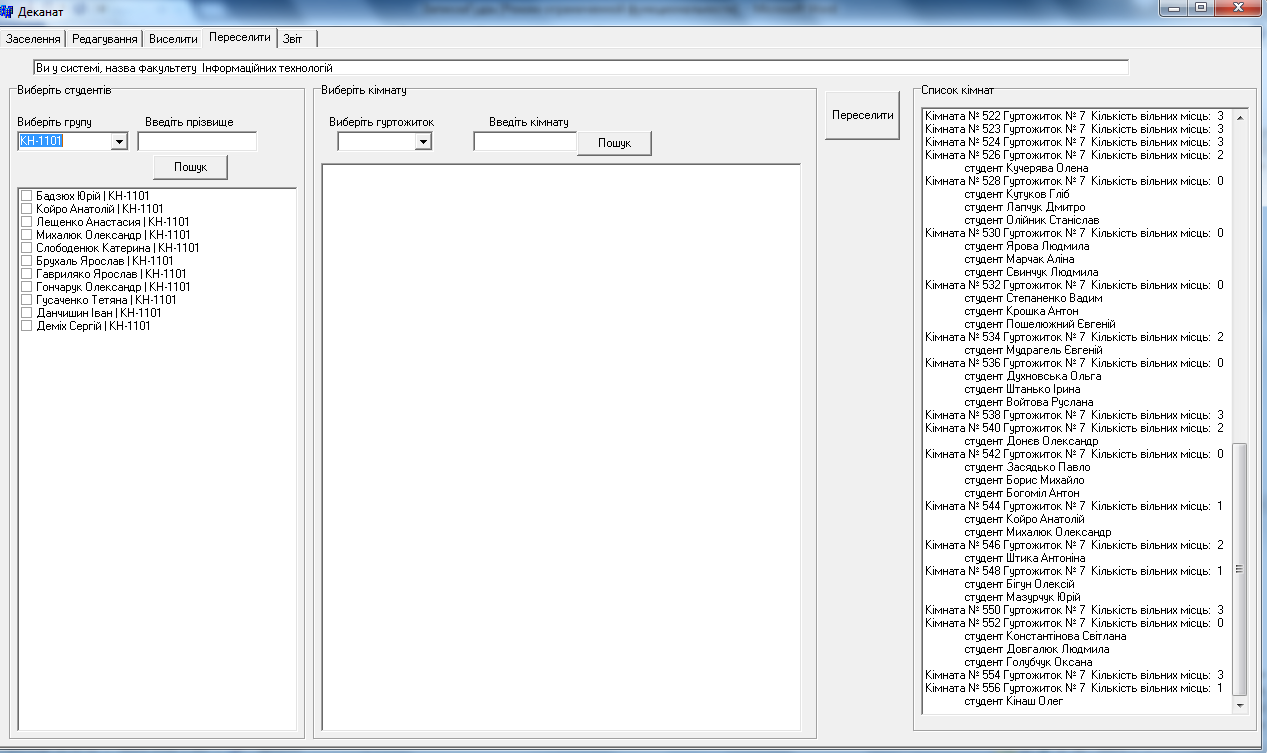


Рис.19 Список поселених студентів.

Обравши студентів, обираємо гуртожиток та кімнату, куди ми хочемо їх переселити. Переселеня відбудеться у 7 гуртожиток, кімната №550 (рис.20). Кількість вільних місць у кімнаті №550 – 3.

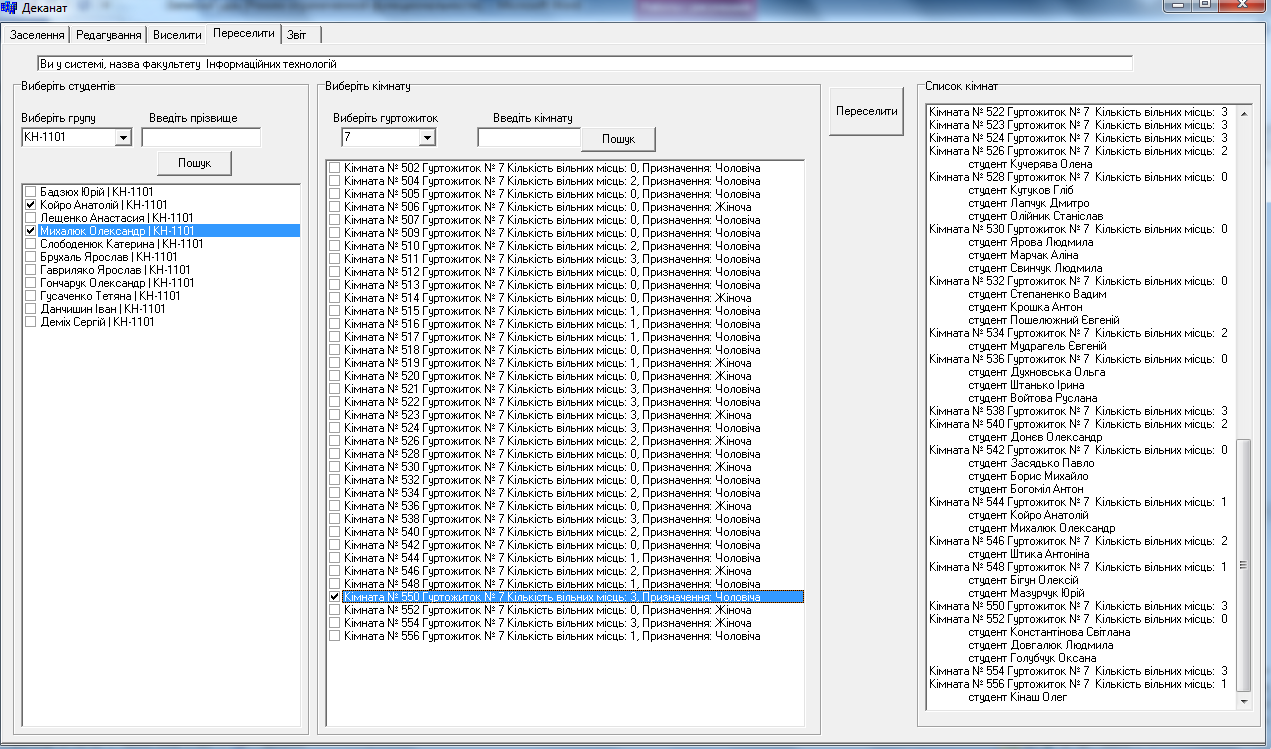


Рис.20. Вибір студентів та місця куди буде переселено студентів.

Після вибору натискаємо на кнопку «Переселити». У результаті ми бачимо, що в кімнаті №544 немає заселених студентів і відповідно вільних місць стало - 3, а в комнаті № 550 заселені вищезгадані студенті та кількість вільних місць стала рівна - 1 (рис.21).

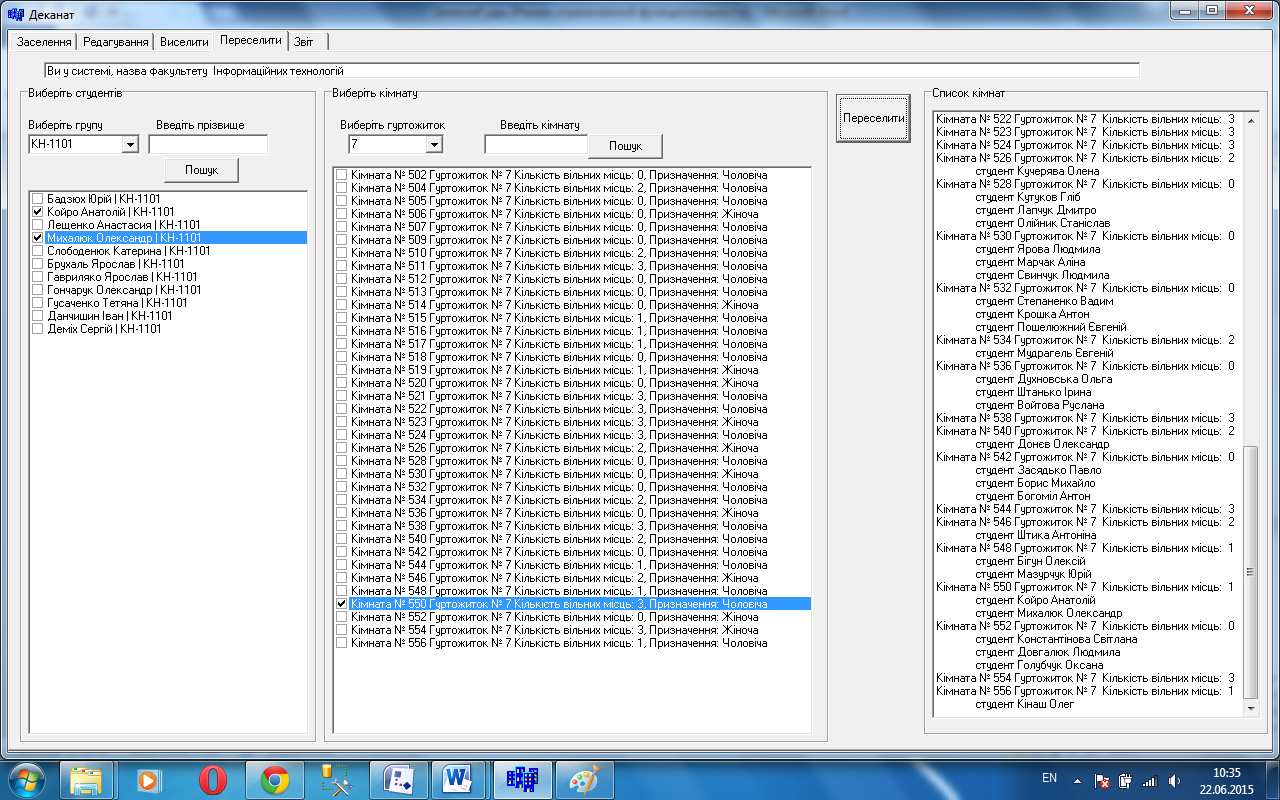


Рис.21 Результат дії переселенння системи.

Тестування процесу «Переселення» пройшло успішно, система виконала цю дію без помилок.

## Вимоги до апаратного та програмного забезпечення

### Діаграма розміщення

Діаграма розміщення — діаграма в UML, на якій зображується фізичний склад ресурсів та вузлів системи, на яких розміщуються компоненти, та відношення між ними. У даній системі є вузол «Сервер», на якому розміщується база даних, та комп’ютери «Деканату» та «Віділу виховної роботи», до яких підключені принтери. Діаграма розміщення зображена на рисунку 22.



Рис.22 Діаграма розміщення.

### 4.2.2 Вимоги до апаратного забезпечення

Для функціонування системи необхідні сервер та клієнтські комп’ютери.

На сервері встановлюється власне сама база даних і комп’ютер, на якому встановлюється прикладне програмне забезпечення. Програма підключається до корпоративної мережі, локальної або до мережі Інтернет.

Клієнтськими називаються комп’ютери, з яких усі користувачі (адміністратори, менеджери, користувачі та гості) отримують доступ до системи.

Сервер

*Системні вимоги*

|  |  |  |
| --- | --- | --- |
| **Ресурс** | **Мінімальний** | **Рекомендований** |
| Процесор | 900 МГц | 2 ГГц і вище |
| Оперативна пам’ять | 512 Мб | 1 Гб і вище |
| Жорсткий диск | 40 Гб | 120 Гб |
| Привод CD-ROM | 12x | 52x і вище |
| Операційна система | Microsoft Windows 2003 Server, 2008 Server | |
| Сервер баз даних | Microsoft SQL Server 2012 | |
| Канал Інтернета, корпоративної або локальної мережі | 1 Мбіт/с | 5 Мбіт/с і вище |

Клієнтський комп’ютер

*Системні вимоги*

|  |  |  |
| --- | --- | --- |
| **Ресурс** | **Мінімальний** | **Рекомендований** |
| Процесор | 600 МГц | 2.5 ГГц і вище |
| Оперативна пам’ять | 512 Мб | 2 Гб і вище |
| Жорсткий диск | 20 Гб | 100 Гб і вище |
| Привод CD-ROM | 4x | 40x і вище |
| Відеокарта | PCI 128 Мб | AGP 512 Мб і вище |
| Канал Інтернета, корпоративної або локальної мережі | 1 Мбіт/с | 5 Мбіт/с і вище |
| Роздільність монітору | 1024х768 |  |

### 4.2.3 Вимоги до програмного забезпечення

З боку клієнта повинна бути встановлена операційна система Microsoft Windows 7 або вище та пакет-встановлювач Windows; для оформлення наказів - програма Microsoft Office 7 або вище. З боку сервера повинно бути встановлено Sql Server 2012.

### 4.2.4 Склад інсталяційного пакету для встановлення розробленої системи

- Інсталяційний пакет, що має файли запитів до бази даних;

- файл з розширенням .exe додатку.

# ВИСНОВКИ

У ході роботи над дипломним проектом на тему «Інформаційна система поселення студентів у гуртожитки студентського містечка» було проведено детальний аналіз предметної області на основі методологій IDEF та SADT та визначено взаємодію користувачів з системою за допомогою діаграми прецедентів на мові UML. Були визначені головні проблеми предметної області та методи їх вирішення.

Було обрано засоби реалізації даної системи, а саме Sql Server як середовище розробки бази даних та Borland C++ builder як середовище розробки прикладного програмного забезпечення на мові C++.

За допомогою Er-діаграми було розроблено логічну модель бази даних, що слугувала для встановлення фізичної моделі бази даних.

Враховуючи вищезгадані фактори та побажання замовників, було розроблено систему, яка отримує вхідні дані у вигляді заяви на поселення та повертає вихідні дані у вигляді відповідних наказів.

Накази оформлюються за допомогою шаблонів, розроблених у компоненті Fast Report. Система працює у вигляді клієнт-серверної архітектури, де база даних знаходиться на стороні сервера, прикладне програмне забезпечення на стороні клієнта. Також визначено системні та програмні вимоги, завдяки яким гарантується якісна робота системи.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бьёрн Страуструп. Язык программирования C++.— М.: Бином-Пресс, 2007. — 1104 с.
2. Буч Г., Якобсон А., Рамбо Дж. UML. Классика CS — 2-е изд.. — СПб.: Питер, 2006. — 736 с.
3. Герберт Шилдт. Теория и практика С++/Шилдт Герберт.— СПб.: BHV — Санкт-Петербург, 1996.
4. Гома Х. UML Проектирование систем реального времени, распределенных и параллельных приложений / Хассан Гома., 2014. – 700 с. – (Объектно-ориентированное программирование)
5. Джаррод Холингворт, Боб Сворт, Марк Кэшмэн, Поль Густавсон. Borland C++ Builder 6. Руководство разработчика. — М.: «Вильямс», 2004. — С. 976.
6. Карвин Б. Программирование баз данных SQL. Типичные ошибки и их устранение / Билл Карвин., 2011. – 336 с.
7. Крэг Ларман. Применение UML 2.0 и шаблонов проектирования. — 3-е изд. — М.: Вильямс, 2006. — 736 с.
8. Полякова Л. Основы SQL / Лариса Полякова., 2007. – 224 с.
9. Послед Б. Borland C++ Builder 6. Разработка приложений баз данных /

Борис Послед., 2003. – 320 с.

1. Прата С. Язык программирования C++. Лекции и упражнения. / Стивен Прата., 2012. – 1248 с.
2. Седжвик Р. Алгоритмы на C++ / Роберт Седжвик., 2011. – 1056 с.
3. Unified Modeling Language (UML) Resource Page [Електронний ресурс] – Режим доступу до ресурсу: http://www.uml.org/.

# Сторінка-1

IF EXISTS(SELECT \* from sys.databases WHERE name='dbsyshost')

BEGIN

DROP DATABASE dbsyshost;

END

create database dbsyshost

use dbsyshost

IF EXISTS(SELECT \* from sys.objects WHERE name='group\_student')

BEGIN

DROP table group\_student;

END

create table group\_student (

Name\_group varchar(25) primary key not null,

number\_of\_students int not null,

curated\_group varchar(25),

group\_leader varchar(25),

Number\_faculty int not null,

)

go

IF EXISTS(SELECT \* from sys.objects WHERE name='faculty')

BEGIN

DROP table faculty;

END

create table faculty(

Number\_faculty int primary key not null,

Name\_faculty varchar(25) not null,

Сторінка-2

telephone\_faculty int,

dean varchar(25),

)

go

IF EXISTS(SELECT \* from sys.objects WHERE name=’passport’)

BEGIN

DROP table passport;

END

create table passport(

series\_number char(10) primary key not null,

issued varchar(40) not null,

when\_issued date not null,

home varchar(50) not null,

sex varchar(10) not null

)

go

IF EXISTS(SELECT \* from sys.objects WHERE name=’Student1’)

BEGIN

DROP table Student1;

END

create table Student1(

identification\_code int primary key not null,

name varchar(40) not null,

second\_name varchar(40) not null,

date\_of\_entry date not null,

Сторінка-3

series\_number char(10) not null,

Name\_group varchar(25) not null,

)

go

IF EXISTS(SELECT \* from sys.objects WHERE name=’rating’)

BEGIN

DROP table rating;

END

create table rating(

id\_rating int primary key not null,

identification\_code int not null,

rating int not null,

date\_of\_assessment date not null,

semester int not null,

)

IF EXISTS(SELECT \* from sys.objects WHERE name=’usersdeans’)

BEGIN

DROP table userdeans;

END

create table usersdeans (

Login\_name varchar(25) primary key not null,

Pussword int not null,

Namber\_faculty int,

)

Go

Сторінка-4

IF EXISTS(SELECT \* from sys.objects WHERE name=’hostel’)

BEGIN

DROP table hostel;

END

create table hostel(

Number\_hostel int primary key not null,

Seats int not null,

Adress varchar(120),

telephone char(10),

commandant varchar(80),

)

go

IF EXISTS(SELECT \* from sys.objects WHERE name=’room’)

BEGIN

DROP table room;

END

create table room(

Number\_room int not null,

Seats int not null,

flor int not null,

id\_assignment int not null,

Number\_hostel int not null,

sex varchar(20) not null,

Number\_category int not null

)

Сторінка-5

go

IF EXISTS(SELECT \* from sys.objects WHERE name=’assignment’)

BEGIN

DROP table assignment;

END

create table assignment(

Number\_assignment int primary key not null,

assignment varchar(250) not null,

)

go

IF EXISTS(SELECT \* from sys.objects WHERE name=’list\_settlement\_student’)

BEGIN

DROP table list\_settlement\_student;

END

create table list\_settlement\_student (

identification\_code int primary key not null,

Number\_hostel int not null,

Number\_room int not null,

settlement\_date date not null,

eviction\_date date not null,

)

Go

IF EXISTS(SELECT \* from sys.objects WHERE name=’assignment’)

BEGIN

Сторінка-6

DROP table assignment;

END

create table assignment (

Number\_assignment int primary key not null,

assignment varchar(250) not null)

go

IF EXISTS(SELECT \* from sys.objects WHERE name='category')

BEGIN

DROP table category;

END

create table category (

Number\_ category int primary key not null,

Name\_ category varchar(250) not null)

go

# Сторінка-1

CheckListBox3->Clear();

ListBox2->Clear();

AnsiString A1[4];

for(int i=0; i < CheckListBox1->Items->Count; i++) {

AnsiString y=»»;

AnsiString d1=»»;

if(CheckListBox1->Checked[i])

{

d1+=CheckListBox1->Items->Strings[i];

int wh1=0;

int len1=d1.Length() ;

String od1,dv1;

for (int i=1;i<=len1;i++){

dv1=d1.SubString(i,1);

if(dv1 != « «) od1+=d1.SubString(i,1);

else {A1[wh1]=od1; od1=»»; wh1++; }

}

y+=»select \* from View\_rating where second\_name=’»;

y+=A1[0];

y+=»’and name=’»;

y+=A1[1] ;

y+=»’»;

DataModule2->ADOQuery3->Close();

DataModule2->ADOQuery3->SQL->Clear();

Сторінка-2

DataModule2->ADOQuery3->SQL->Add(y);

DataModule2->ADOQuery3->Open();

while(!DataModule2->ADOQuery3->Eof){

if((DataModule2->ADOQuery3->FieldByName(«rating»)->AsInteger>300||DataModule2->ADOQuery3->FieldByName(«course»)->AsInteger==1)&&DataModule2->ADOQuery3->FieldByName(«home»)->AsString!=»Київ»)

{

CheckListBox3->Items->Add(DataModule2->ADOQuery3->FieldByName(«second\_name»)->AsString+» «+DataModule2->ADOQuery3->FieldByName(«name»)->AsString+» | «+DataModule2->ADOQuery3->FieldByName(«Name\_group»)->AsString+» «);

}

else{

if(DataModule2->ADOQuery3->FieldByName(«home»)->AsString==»Київ»)

{

ListBox2->Items->Add(DataModule2->ADOQuery3->FieldByName(«second\_name»)->AsString+» «+DataModule2->ADOQuery3->FieldByName(«name»)->AsString+» | «+DataModule2->ADOQuery3->FieldByName(«Name\_group»)->AsString+» Близько живуть «);

}

else{

Сторінка-3

ListBox2->Items->Add(DataModule2->ADOQuery3->FieldByName("second\_name")->AsString+" "+DataModule2->ADOQuery3->FieldByName("name")->AsString+" | "+DataModule2->ADOQuery3->FieldByName("Name\_group")->AsString+" Низький рейтинг ");

}}

DataModule2->ADOQuery3->Next();

}} }

DataModule2->ADOQuery3->Close();

}