МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**Національний університет «Запорізька політехніка»**

Кафедра програмних засобів

(найменування кафедри)

**КУРСОВИЙ ПРОЕКТ**

з дисципліни «Об’єктно-орієнтоване програмування»

(назва дисципліни)

на тему: «Чат-бот для підтримки міжнародної академічної мобільності»

Студентів 2 курсу групи КНТ-229

спеціальності Комп'ютерні науки

Сепетий О. В.

(прізвище та ініціали)

Колбаса Д. О.

(прізвище та ініціали)

Ващенко М. І.

(прізвище та ініціали)

Момот В. І.

(прізвище та ініціали)

Керівник к.т.н., проф. Табунщик Г.В.

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала

Кількість балів: \_\_\_\_\_\_\_ Оцінка: ECTS

Члени комісії Табунщик Г. В.

(підпис) (прізвище та ініціали)

Миронова Н. О.

(підпис) (прізвище та ініціали)

Каплієнко Т. І.

(підпис) (прізвище та ініціали)

м. Запоріжжя

2020 рік

**ЗМІСТ**

**ЗАВДАННЯ  
на курсовий проект студентів**

Сепетого О.В., Колбаси Д. О., Ващенко М. І., Момота В. І.

**1. Тема проекту:** Чат-бот для підтримки міжнародної академічної мобільності

**2. Термін здачі студентами закінченого проекту:** 14 грудня 2020

**3. Загальні вимоги до курсового проекту:** Розроблений програмний продукт повинний бути новим, розробленим з чистого листа. Для візуалізації поставленого завдання використовувати графічний інтерфейс із використанням інструментальних засобів відповідного середовища розробки. Роботу слід виконати в середовищі розробки Qt Creator, Visual Studio, IntelliJ IDEA. Мови програмування ‒ C++, C#, Java. Для групової роботи використовується GitHub

**Вимоги до інтерфейсу**

* + Необхідно розробити мінімум п'ять форм інтерфейсу, дизайн інтерфейсу.
  + Обов’язково необхідно реалізувати допомогу щодо використання програми, інформацію про розробників.

**Вимоги до класів**

* + Ієрархія класів повинна включати мінімум дванадцять класів, з яких один ‒ абстрактний, три ‒ розроблені класи користувача. Опис класів необхідно оформити у вигляді окремих модулів.

**Вимоги до зберігання даних**

* + Інформація зберігається в звичайних файлах. Інформація завантажується в пам'ять програми при старті, редагується в процесі роботи і зберігається в файли при завершенні. Формат файлу вибирається самостійно.
  + При використанні CSV, JSON та XML будуть додаткові питання на захисті з даних форматів представлення даних.
  + Необхідно взаємодіяти з CSV, JSON або XML (2 з трьох на вибір студентів).
  + За бажанням можна також використовувати базу даних. При використанні СКБД будуть додаткові питання на захисті по завантаженню даних таблиць з бази даних в структуру даних.

**Вимоги до алгоритмів**

* + Необхідно розробити авторських 2-4 алгоритми, наприклад, пошуку або сортування, та порівняти їх за швидкістю виконання зі стандартними реалізаціями.

**Вимоги до оброблення виняткових ситуацій**

* + Для оброблення виняткових ситуацій необхідно розробити авторський клас.
  + Програма повинна обробляти не менш ніж 10 різних помилкових ситуацій.

**4. Зміст розрахунково-пояснювальної записки:**

1. Аналіз предметної області;

2. Аналіз програмних засобів;

3. Основні рішення з реалізації компонентів системи;

4. Керівництво програміста;

5. Керівництво корисутвача;

Висновки

Додаток А. Код програми

Додаток Б. Інтерфейс програми

Додаток В. Слайди презентації

**5. Дата видачі завдання:** 11 вересня 2020 року

**КАЛЕНДАРНИЙ ПЛАН**

**РЕФЕРАТ**

**ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ**

ООП - об’єктно-орієнтоване програмування.

Е.Г.І. - елементи графічного інтерфейсу

АМ – академічна мобільність

ЗВО — заклад вищої освіти

ПЗ — програмне забезпечення

STL – Standart Template Library

MinGW – Minimal GNU for Window

GCC – GNU Compilers Collection

**ВСТУП**

Під час навм легко загубитися або важко знайти потрібний розділ. Також за нашим баченням на сайті не були розкриті деякі ключові питання типу ціни, мови та іншого.

Вирішенням даної проблеми є створення спеціалізованого чат-бота для підтримки міжнародної академічної мобільності. Найголовніша мета нашого чат-бота - донести цікаву, важливу та легку в сприйняттіінформацію до користувачачання в університеті, у студентів є можливість спробувати навчання у закордонних університетах. На жаль про це мало говорять викладачі та куратор. Деяку інформацію можна знайти на офіційному сайті університету, але це не так зручно і зрозуміло, адже наш сайт має дуже багато розділів і часом та та максимально залучити до участі в АМ.

Чат-боти – це універсальні помічники, які можуть бути використані у різноманітних сферах людського життя, в тому числі і наданні інформації про академічну мобільність.

Мета даного курсового проекту є – навчитись використовувати засоби об'єктно-орієнтованого програмування для створення чат-ботів.

Проведено аналіз предметної області, досліджено аналогічне програмне забезпечення, а також існуючі методи та програмні засоби для вирішення завдання.

Для реалізації програмного продукту використовувалася мова програмування С++ та середа розробки Qt Creator з її власними зручними класами.

Здійснено опис прийнятих рішень, реалізованих класів, наведено опис полів та методів реалізованих класів.

**1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ**

В даному розділі розглянуті існуючі методи вирішення завдання підтримки АМ та програми, що реалізують дану потребу, а саме чат-боти що спрямовані на спрощення взаємодії між студентами та університетом, серед можливостей яких, зокрема є підтримка АМ .

1.1 Аналіз академічної мобільності, як процесу що потребує автоматизації

Міжнародна академічна мобільність являє собою короткострокові програми обміну: кредитна мобільність на навчання (3-12 міс.) та на практику (2-12 міс.) для українських студентів: (молодших) бакалаврів, магістрів, аспірантів, докторантів.

Академічна мобільність для викладання або підвищення кваліфікації або стажування (від 5 днів до 2 міс.) для викладачів та працівників закладів вищої освіти.

Організація цієї події проходить між одиним ЗВО з країни-члена Програми та одиним ЗВО країни-партнера Програми Еразмус+.

Заявником - координатором проєктів з мобільності виступають заклад вищої освіти країн-членів програми Еразмус+.  
 Координують напрям діяльності Національні Агентства Еразмус+ країн-членів Програми (децентралізований напрям).

Брати участь у АМ можуть: студенти (молодші бакалаври, бакалаври, магістри, аспіранти, докторанти), викладачі і административний персонал.  
 Щоб взяти участь, необхідно пройти співбесіду з представником університету, в якому особа навчається, або працює і з представником партнерського университету. Але першими і головними кроками є розуміння що таке АМ і реєстрація для проходження співбесіди. Для полегшення цих кроків і створений наш чат-бот з підтримки АМ.

1.2 Огляд існуючих методів вирішення завдання та аналогів програмного забезпечення

***1.2.1 Передмова***

Звісно одним з найочевидніших шляхів донесення студентів (викладачів та інших працівників закладів вищої освіти) є загальна презентація про процес АМ та процес взяття в ньому участі. Але недоліком цього методу витрати часу на підготовку цього заходу і на всласне презентування.

Іншим методом є застосування чат-бота, що доступний цілодобово і має відповіді на всі запитання загального характеру, який швидко і зрозуміло донесе до людини потрібну їй інформацію, а також надасть можливість реєстрації на співбесіду і сформує перелік охочих.

Перед розглядом аналогів, треба зазначити, що програмного забезпечення, яке б задовольняло назві “Програмне забезпечення для підтримки академічної мобільності” або “Чат-бот для підтримки акадеічної мобільності” не знайшлося, що свідчить про те, що конкретних аналогів немає або їх дуже мало або про те що ПЗ з таким функціоналом нині не є дуже затребуваним. Підтримка АМ є частиною декількох знайдених чат-ботів. Але слід зазначити, що ПЗ, яке немає цього функціоналу аналізувалося на предмет загальної організації побудови взаємодії з користувачем.

Програмами для підтримки академічної мобільності є:

* Програма “MoveOn”;
* “Chatbot TARS”;

Чат-бот, що не маює підтримки АМ, але є цікавими прикладами ПЗ для університетів:

* “Virtual spirits chat bot”;

Критерії аналізу аналогів ПЗ:

* точне донесення до користувача інформації щодо АМ, або університет (якщо чат бот не підтримує АМ)
* побудова діалогу з корстувачем;
* наявність безкоштовної ліцензії;
* зручність і простота інтерфейсу.

1.2.2 Програма “MoveOn”

Компанія-видавець: QS UniSolution

Посилання: https://www.qs-unisolution.com/moveon/

Короткий опис: MoveON - це платформа на основі аналітичних даних для міжнародних офісів вищих навчальних закладів, для покращення процесів їхньої мобільності та ефективніше відстеження міжнародних партнерських відносин. Вона надає можливість створити свій шлях обміну студентами та побудувати стратегічне партнерство для досягнення цілей інтернаціоналізації закладу.

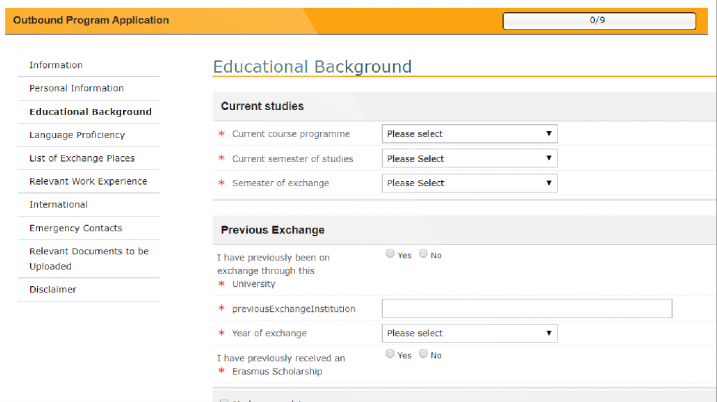
Переваги:

* Розвернення програми проходить швидко і легко
* Має досить постий і зрозумілий інтерфейс
* Дозволяє легко керувати програмами обміну студентів
* Співпрацює з Erasmus

Недоліки

* ПЗ спрямовано на полегшення роботи адміністрації університету, але ні як не на комунікацію зі студентами.
* Має платну ліцензію

Робота програми продемонстрована на рисунку 1.1.

 Рисунок 1.1 – Робота програми «MoveOn»

**1.2.3 Chatbot TARS**

Компанія-видавець: Tars.

Посилання: https://hellotars.com/chatbot-templates/education/SJmmXq/university-search-chatbot-for-studying-abroad

Чат-бот допомагає навчальним закладам автоматизувати процес планування зустрічей, щоб заощадити час та зробити більш привабливими форми реєстрації на курси

Переваги:

* Чат-бот підтримує АМ
* Діалог з користувачем проходить доволі зручно і зрозуміло
* Головним способом комунікації з програмою є панель кнопок, що робіть комунікацією простою
* має безкоштону пробну версію

Недоліки:

* Не надає інформацію про університети.

Робота програми продемонстрована на рисунку 1.2.

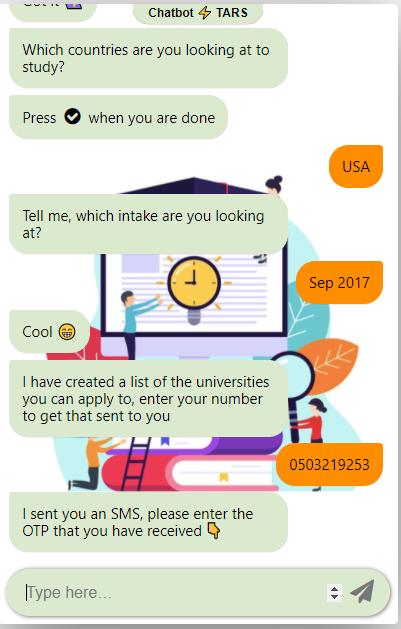


Рисунок 1.2 – Робота з програмою «Chatbot TARS»

1.2.4 «Virtual spirits chat bot»

Компанія-видавець: Virtual spirits.

Посилання: https://www.virtualspirits.com/chatbot-for-university.aspx

Короткий опис: Чат-бот використовується для підтримати студентів та надання миттєвої відповіді на послуги кампусу, реєстрацію, стипендії, навчальні послуги, поради щодо іспитів, ІТ-послуг, усунення несправностей тощо

Переваги:

* Має широкий набір функціоналу.
* Зручний інтерфейс
* Взаємодія виконується за допомоги текстових повідомлень і іноді кнопок (категорій).

Недоліки:

* Нажаль не надає підтримки АМ

Робота програми продемонстрована на рисунку 1.3.

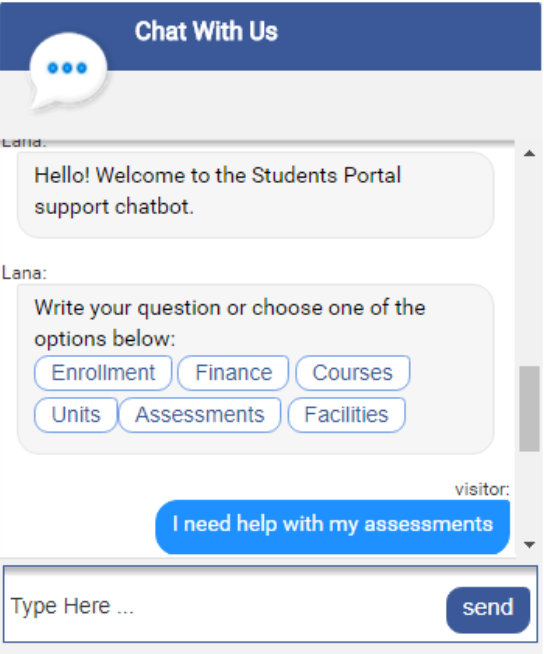


Рисунок 1.3 – Робота з програмою «Virtual spirits chat bot»

**2 АНАЛІЗ ПРОГРАМНИХ ЗАСОБІВ**

В даному розділі розглянуті основні особливості програмних засобів, за допомогою який реалізовано курсовий проект.

2.1 Огляд особливостей мови програмування

С++ - мова програмування загального призначення.

С++ підтримує об’єктно-орієнтовану парадигму, що дозволяє представити програму, як набір обєектів, кожен з яких має свої властивості, поведінку і своє призначення в програмі. Програму, написану з використанням ООП легше модифікути і підтримувати. Однією з особливостей саме мови С++ в тому, що вона підтримує множинне спадкування. С++ підтримує програмування з використанням шаблонів, що спрощує роботу з даними. Бібліотека Qt, що дозволяє розроблювати графічний інтерфейс, має свою реалізацію STL, свою реалізацію потоків вводу-виводу, та інших надбань стандартної бібліотеки з більшою кількістю можливостей, що робить процес розробли ПЗ більш швидким та продуктивним.

2.2 Огляд особливостей обраного компілятору

Компілятор MinGW – версія компілятора GCC, створена для розробки ПЗ мовою С++ на операціїних системах сімейства Windows. Розрорблене нами ПЗ не використовує жодних специфічних речей, що можуть робити Unix-подібні операційні системи, чого не може робити Windows, тому необхідність у використанні компілятора Cygwin відпала. Компілятор MSVC постачається разом з Microsoft Visual Studio, але на наш погляд не має істотних переваг. До того ж робота в Visual Studio потребує додаткових операцій з підключенням бібліотеки Qt.

2.3 Огляд класової ієрархії

QString — Клас динамічного типу даних для роботи зі рядками тексту. Використовується для зберігання інформації (рядки тексту), що зчитується з файлів, вводиться користувачем.

QFile – Клас для введення-виведення для зчитування та запису текстових і двійкових файлів і ресурсів. Використовується для роботи з файлами, зчитування та запису текстової інформації, а також для підключення таблиць стилів.

QList – Контейнерний клас Qt на основі двозв’язного списку. Дозволяє виконувати операції вставки та видалення елементу за постійний час. Використовується для зберігання даних про студентів і університети.

QStringList — Нащадок класу QList, але на відміну від нього не є шаблоним класом, тобто це QList з типом заповнювачем QString. Він також дозволяє виконувати операції вставки та видалення елементу за постійний час. Використовується для зберігання рядків тексу які задіяні у локалізації.

QDomDocument – Клас, що реалізує собою роботу з XML документом. За своєю суттю є коренем дерева документа. Використовується для зчитування та запису інформації в XML документ.

QDateTime — Клас, що надає функції дати та часу, тобто об’єднує класи QDate і QTime. Використовується для визначення часу відпраки повідомлення.

QRegExp - Клас, що надає порівняння зі зразком за домоги регулярних виразів. Використовуєтться для зберігання регулярного виразу.

QRegExpValidator — Клас, використовуються для перевірки відповідності рядка регулярному виразу.

2.4 Висновки з розділу

У даному розділі було описано мову програмування та її особливості. Також було описано класи, що були використані в процесі розробки ПЗ та коротко для чого саме були вони використані.

**3 ОСНОВНІ РІШЕННЯ З РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ**

В даному розділі розглянуті основні рішення з розробки класів, основні розроблені алгоритми, рішення щодо розробки інтерфейсу користувача, рішення щодо збереження даних та обробки виключних ситуацій.

3.1 Основні рішення щодо розроблених класів

Структура програми складається з 19 класів, 13 з яких – користувацькі, 6 – класи форм. Один з користувацьких класів є абстрактним. Діаграми користувацьких класів знаходяться у додатку А.

Класова структура програми:

1. Класи форм:

– MainWindow;

– Bug;

– Developer;

– Manual;

– RateApp;

– Settings;

1. Користувацькі класи:

– TextChat;

– StudentAM;

– Student;

– UnivercityAM;

– University;

– GeneralInfo;

– Localization;

– Exception;

– FileException;

– EmptyFileException;

– ReadingDataException;

– RegisterException;

– Validate.

***3.1.1 Клас MainWindow***

Клас MainWindow є класом головної форми та містить у собі функціонал комуніації з користувачем через графічний інтерфейс (слоти, що оброблюють сигнали від кнопок та інших форм додатку, сигнал для зміни мови інтерфейсу всіх форм, метод для відображення повідомлень в чаті, та покажчики на об’єкти, з яких складається програма). Код файлу визначення класу знаходиться в додатку Б. Дані та методи класу наведені у таблиці 3.1.

Таблиця 3.1 — Опис полів і методів класу MainWindow

|  |  |  |
| --- | --- | --- |
| Поля та методи класу | | Опис |
| **1** | | **2** |
| ***public:*** | | |
| MainWindow(QWidget \*parent = nullptr) | Конструктор, приховує всі слої кнопок окрім головного (того, з я кого покинається діалог з користувачем), встановлює розташування всіх слоів кнопок, створює об’єкти через покажчики Chat та Locale (поля класа), що реалізують роботу чат-бота, задає значення всіх інших покажчиків nullptr (studInfo, bugForm, developerForm, manualForm, rateAppForm, settingsForm) та задає тексти е.г.і задопомоги об’єкта Locale, що має українську як мову за замовчуванням. | |
| ~MainWindow() | Деструктор, видаляє об’єкти Locale та Chat, а такоє, якщо під час роботи програми були створені об’єкти форм, видаляє ці об’єкти. Видаляє поле ui. | |
| ***signals:*** | | |
| void localizateWindows() | | Сигнал, що виникає, після зміни мови інтерфейсу головної форми, для зміни мови інтерфейсу інших форм. |
| ***private slots:*** | | |
| void on\_action\_triggered() | | Слот, що створює або показує форму “Як користуватися?”. |
| void on\_action\_2\_triggered() | | Слот, що створює або показує форму “Оцінити додаток”. |
| void on\_action\_3\_triggered() | | Слот, що створює або показує форму “Детальніше”. |
| void on\_action\_4\_triggered() | | Слот, що створює або показує форму “Налаштування”. |
| void on\_action\_5\_triggered() | | Слот, що створює або показує форму “Знайшли баг?”. |
| void on\_action\_7\_triggered() | | Слот, який зберігає введені данні про студентів у відсортованому вигляді в файл studentsAM.xml. |
| void on\_btn1\_clicked() | | Слот, що відповідає на сигнал кнопки (виводить інформацію в чат) “Що таке АМ?” і показує групу кнопок з назвами btn1\_\* |
| void on\_btn1\_1\_clicked() | | Слот, що відповідає на сигнал кнопки (виводить інформацію в чат) “Для кого це?”. |
| void on\_btn1\_2\_clicked() | | Слот, що відповідає на сигнал кнопки (виводить інформацію в чат) “Мова”. |
| void on\_btn1\_3\_clicked() | | Слот, що відповідає на сигнал кнопки (виводить інформацію в чат) “Коштовність”. |
| void on\_btn1\_4\_clicked() | | Слот, що відповідає на сигнал кнопки (виводить інформацію в чат) “З чого почати?” і показує групу кнопок з назвами btn1\_4\_\* |
| void on\_btn1\_back\_clicked() | | Приховує групу кнопок btn1\_\* і показує групу кнопок btn\* . |
| void on\_btn1\_4\_1\_clicked() | | Слот, що відповідає на сигнал кнопки (виводить інформацію в чат) “Приклад мотиваційного листа”. |
| void on\_btn1\_4\_back\_clicked() | | Приховує групу кнопок btn1\_4\_\* і показує групу кнопок btn1\_\* |
| void on\_btn2\_clicked() | | Слот, що відповідає на сигнал кнопки (виводить інформацію в чат) “Університети-партнери” і показує групу кнопок з назвами btn2\_\* |
| void on\_btn2\_1\_clicked() | | Слот, що відповідає на сигнал кнопки (виводить інформацію в чат) назва якої, залежить від назви першого університету в файлі UniversitiesAM.xml. |
| void on\_btn2\_2\_clicked() | | Слот, що відповідає на сигнал кнопки (виводить інформацію в чат) назва якої, залежить від назви другого університету в файлі UniversitiesAM.xml. |
| void on\_btn2\_3\_clicked() | | Слот, що відповідає на сигнал кнопки (виводить інформацію в чат) назва якої, залежить від назви третього університету в файлі UniversitiesAM.xml. |
| void on\_btn2\_4\_clicked() | | Слот, що відповідає на сигнал кнопки (виводить інформацію в чат) назва якої, залежить від назви четвертого університету в файлі UniversitiesAM.xml. |
| void on\_btn2\_5\_clicked() | | Слот, що відповідає на сигнал кнопки (виводить інформацію в чат) назва якої, залежить від назви п’ятого університету в файлі UniversitiesAM.xml. |
| void on\_btn2\_6\_clicked() | | Слот, що відповідає на сигнал кнопки (виводить інформацію в чат) назва якої, залежить від назви шостого університету в файлі UniversitiesAM.xml. |
| void on\_btn2\_back\_clicked() | | Приховує групу кнопок btn2\_\* і показує групу кнопок btn\* . |
| void on\_btn3\_clicked() | | Приховує групу кнопок btn\* показує рядок введення і кнопку “Enter”. Реалізує початок реєстрації. |
| void on\_btn4\_clicked() | | Слот, що відповідає на сигнал кнопки (виводить інформацію в чат) “Контакти” |
| void on\_btn\_enter\_clicked() | | Реалузує процес реєстрації. Введеня данних з рядку вводу, їх перевірку, запис до об’єкту StudInfo та додавання до списку студентів. |
| void on\_LanguageChanged() | | Слот, що змінює значення мови у об’єкті Locale, реалізує зміну мови інтерфейсу головної форми за сигналом, отриманим від форми “Налаштування” та створює сигнал void localizateWindows(). |
| void on\_FeedbackSent(int feedback) | | Слот, що записує оцінку додатку за сигналом void FeedbackSent(int feedback) форми “Оцінити додаток”. |
| void on\_BugReported(QString Report) | | Слот, що записує повідомлення про баг за сигналом void BugReported(QString Report) форми “Знайшли баг?”. |
| ***private:*** | | |
| void displayChat() | | Метод, що відобращає історію повідомлень полі чату. Визивається при натискані на будь-яку кнопку. |
| Ui::MainWindow \*ui | | Покажчик на елементи форми MainWindow. |
| Student \*studInfo | | Покажчик на тип об’єкт типу Student, в якому зберігається інформація про студента під час реєєстарії. |
| TextChat \*Chat | | Покажчик на об’єкт типу TextChat, що відповідає за реалізацію алгоритмів та роботи з даними. |
| bug\* bugForm | | Покажчик на форму “Занйшли баг?”. |
| developer\* developerForm | | Покажчик на форму “Детальніше”. |
| manual\* manualForm | | Покажчик на форму “Як користуватися?”. |
| rateApp\* rateAppForm | | Покажчик на форму “оцінити додаток”. |
| settings\* settingsForm | | Покажчик на форму “Налаштування”. |
| Localization \*Locale | | Покажчик на об’єкт типу Localization, що містить увесь текст інтерфейсу у відповідній мові. |

***3.1.2 Клас Bug***

Клас Bug є класом форми “Знайшли баг?”, що може реагувати на зміну мови інтерфейсу та дозволяє ввести повідомлення про баг, що буде звережено у файлі, назва якого визначена в методі GeneralInfo::void writeToTxtFile(QString feedBackOrBug). Код файлу визначення класу знаходиться в додатку Б. Дані та методи класу наведені у таблиці 3.2.

Таблиця 3.2 — Опис полів і методів класу Bug

|  |  |
| --- | --- |
| Поля та методи класу | Опис |
| **1** | **2** |
| ***public:*** | |
| explicit bug(Localization \*Locale, QWidget \*parent = nullptr) | Конструктор, встановлює значення поля Locale значення аргументу Locale. |
| ~bug() | Деструктор, видаляє поле ui. |
| ***signals:*** | |
| void BugReported(QString Report) | Сигнал, що передає повидомлення введене в текстове поле форми, головній формі для запису. |
| ***private slots:*** | |
| void on\_btn\_Send\_clicked() | Слот, що створює сигнал void BugReported(QString Report) при натисканні на кнопку “Відправити” |
| void on\_localizateWindows() | Використовує покажчик на об’єкт Locale для взяття тексту елеметів інтерфейсу певної мови. |
| ***private:*** | |
| Ui::bug \*ui | Покажчик на елементи форми Bug. |
| Localization \*Locale | Покажчик на об’єкт типу Localization, що є полем класу MainWindow, який передається конструктору класа Bug при створенні форми. |

***3.1.3 Клас Developer***

Клас Developer є класом форми “Детальніше”, що надає інформацію про розробників та може реагувати на зміну мови інтерфейсу. Код файлу визначення класу знаходиться в додатку Б. Дані та методи класу наведені у таблиці 3.3.

Таблиця 3.3 — Опис полів і методів класу Developer

|  |  |
| --- | --- |
| Поля та методи класу | Опис |
| **1** | **2** |
| ***public:*** | |
| explicit developer(Localization \*Locale, QWidget \*parent = nullptr) | Конструктор, встановлює значення поля Locale значення аргументу Locale. |
| ~developer() | Деструктор, видаляє поле ui. |
| ***private slots:*** | |
| void on\_localizateWindows() | Використовує покажчик на об’єкт Locale для взяття тексту елеметів інтерфейсу певної мови. |
| ***private:*** | |
| Ui::developer \*ui | Покажчик на елементи форми Developer. |
| Localization \*Locale | Покажчик на об’єкт типу Localization, що є полем класу MainWindow, який передається конструктору класа Bug при створенні форми. |

***3.1.4 Клас Manual***

Клас Manual є класом форми “Як користуватися?”, що надає інформацію про те як користуватися додатком. Код файлу визначення класу знаходиться в додатку Б. Дані та методи класу наведені у таблиці 3.4.

Таблиця 3.4 — Опис полів і методів класу Manual

|  |  |
| --- | --- |
| Поля та методи класу | Опис |
| **1** | **2** |
| ***public:*** | |
| explicit manual(Localization \*Locale, QWidget \*parent = nullptr) | Конструктор, встановлює значення поля Locale значення аргументу Locale. |
| ~manual() | Деструктор, видаляє поле ui. |
| ***private slots:*** | |
| void on\_localizateWindows() | Використовує покажчик на об’єкт Locale для взяття тексту елеметів інтерфейсу певної мови. |
| ***private:*** | |
| Ui::manual \*ui | Покажчик на елементи форми Manual. |
| Localization \*Locale | Покажчик на об’єкт типу Localization, що є полем класу MainWindow, який передається конструктору класа Bug при створенні форми. |

***3.1.5 Клас RateApp***

Клас RateApp є класом форми “Як користуватися?”, що надає інформацію про те як користуватися додатком. Код файлу визначення класу знаходиться в додатку Б. Дані та методи класу наведені у таблиці 3.5.

Таблиця 3.5 — Опис полів і методів класу RateApp

|  |  |
| --- | --- |
| Поля та методи класу | Опис |
| **1** | **2** |
| ***public:*** | |
| explicit rateApp(Localization \*Locale, QWidget \*parent = nullptr) | Конструктор, встановлює значення поля Locale значення аргументу Locale. |
| ~rateApp() | Деструктор, видаляє поле ui. |
| ***private slots:*** | |
| void on\_localizateWindows() | Використовує покажчик на об’єкт Locale для взяття тексту елеметів інтерфейсу певної мови. |
| ***private:*** | |
| Ui::rateApp \*ui | Покажчик на елементи форми RateApp. |
| Localization \*Locale | Покажчик на об’єкт типу Localization, що є полем класу MainWindow, який передається конструктору класа Bug при створенні форми. |

***3.1.6 Клас Settings***

Клас Settings є класом форми “Налаштування”, що дозволяє змінювати тему (світла, темна) та мову інтерфейсу (українська, англійська, німецька, російська). Код файлу визначення класу знаходиться в додатку Б. Дані та методи класу наведені у таблиці 3.6.

Таблиця 3.6 — Опис полів і методів класу Settings

|  |  |
| --- | --- |
| Поля та методи класу | Опис |
| **1** | **2** |
| ***public:*** | |
| explicit settings(Localization \*Locale, QWidget \*parent = nullptr) | Конструктор, встановлює значення поля Locale значення аргументу Locale, а такох встановлює індекс комбо-боксу вибору мови значенням Locale::langCode. |
| ~settings() | Деструктор, видаляє поле ui. |
| ***signals:*** | |
| void LanguageChanged() | Сигнал сповіщає головну форму про необхідність зміни мови всіх елементів інтерфейсу. |
| ***private slots:*** | |
| void on\_localizateWindows() | Використовує покажчик на об’єкт Locale для взяття тексту елеметів інтерфейсу певної мови. |
| void on\_comboBox\_currentIndexChanged (int index) | Слот реагує на сигнал комбобоксу при зміні мови. Встановлює певний індекс мови в об’кті Locale та викликає сигнал void LanguageChanged() |
| void on\_btndark\_clicked() | Встановлює таблицю стилів darkstyle.css |
| void on\_btnlight\_clicked() | Встановлює таблицю стилів lightstyle.css |
| ***private:*** | |
| Ui::settings \*ui | Покажчик на елементи форми Settings. |
| Localization \*Locale | Покажчик на об’єкт типу Localization, що є полем класу MainWindow, який передається конструктору класа Bug при створенні форми. |
| QFile styleF | Поле, використовується, для взаємодії з файлом, в якому зберігається стиль інтерфейсу. |

***3.1.7 Клас TextChat***

Клас TextChat є основним класом чат-бота. Він об’єднує в собі инструментарій для роботи с даними (наслідує класи GeneralInfo, StudentAM, UniversityAM) та релізацію алгоритмів швидкого сортування та лінійного пошуку. Також цей клас зберігає історію чату (текст повідомлень, час коли вони були відправлені і ким). Код файлу визначення класу знаходиться в додатку Б. Дані та методи класу наведені у таблиці 3.7.

Таблиця 3.7 — Опис полів і методів класу TextChat

|  |  |
| --- | --- |
| Поля та методи класу | Опис |
| **1** | **2** |
| ***public:*** | |
| TextChat(const QString language) | Конструктор винонує зчитування інформації, що зберігається в файлах generelInfo.csv та universitiesAM.xml певною певної мови (змінна language). |
| void addNewMessage(const bool isUsers, const QString message) | Метод, що додає текст нового повідомлення (message) до списку messages, флаг isUsers до списку isUsersMessage і записує час додавання до списку time. |
| void resetIterators() | Метод занулює iteratorTime та iteratorMessage. Після цього методи getNextMessage() та getNextTime() повертатимуть елементи списків спочатку. |
| QString getNextMessage() | Метод повертає елемент списку messages з індексом iteratorMessage та інкрементує iteratorMessage. |
| QString getNextTime() | Метод повертає елемент списку time з індексом iteratorTime та інкрементує iteratorTime. |
| bool getIsUsersMessage() | Метод повертає елемент зі списку isUsersMessage з індексом iteratorMessage. Важливо викликати його перед методом getNextMessage(). |
| bool isEnd() | Метод повертає false, якщо iteratorMessage < messages.size(). |
| bool searchForTheSame(const Student &info) | Метод, реалізує лінійний пошук об’єкта Student info серед списку StudentAM::QList<Student> studentList. Якщо у списку знаходиться такий об’єкт, що має ті самі значення полів Strudent::fullName, Strudent::group, Strudent::course, то повертає true. |
| void quickSort(QList <Student>\* studList, int left, int right) | Метод, реалізує швидке сортування списку об’ектів типу Student за полем Student::fullName, в алфавітному порядку. |
| int findMinCourse(QList <Student>\* studList) | Метод повертає найменже значення поля Student::cource серед всіх об’ектів списку QList <Student>\* studList. |
| QList<Student>\* sellectSameGroup(QList <Student>\* studList) | Метод повертає покажчик на QList<Student> елемети якого мають однакове поле Student::group, те саме що і перший елемент списку QList <Student>\* studList. Елементи що входять в список покажкик якого повертається — вилчаються зі списку studList. |
| QList<Student>\* sellectMinCourse(QList <Student>\* studList) | Метод повертає покажчик на QList<Student> елемети якого мають однакове поле Student::cource, те саме що і перший елемент списку QList <Student>\* studList. Елементи що входять в список покажкик якого повертається — вилчаються зі списку studList. |
| void sortStudents() | Метод сортує спсок StudentAM::Qlist<Student> таким чином, що інформація про студентів стає відсортованою за курсом (від мінімального до максимального), групою та іменами (в кожній групі студенти розміщені в алфавітному порядку). Групи в межах курсу розміщені в довільному порядку. |
| ***private:*** | |
| QList<QDateTime> time | Список, що зберігає інформацію про час відправки кожного з повідомлень. |
| QList<QString> messages | Список, що зберігає тести повідомлень. |
| QList<bool> isUsersMessage | Список, що зберігає інформацю про те кому (чат-боту фбо користувачеві) належить повідомлення. |
| int iteratorTime | Лічильник, який зберігає номер наступного елемнту списку time. |
| int iteratorMessage | Лічильник, який зберігає номер наступного елемнту списку messages. |

***3.1.8 Клас StudentAM***

Клас StudentAM є класом який зберігає інформацію зареєстрованих студентів, а також дозволяє записати інформацію до \*.xml файлу, назва якого визначена в методі void writeToXml(). Код файлу визначення класу знаходиться в додатку Б. Дані та методи класу наведені у таблиці 3.8.

Таблиця 3.8 — Опис полів і методів класу StudentAM

|  |  |
| --- | --- |
| Поля та методи класу | Опис |
| **1** | **2** |
| ***public:*** | |
| void addStudent(const Student& student) | Метод додає об’єкт типу Student до списку studentList. |
| void deleteAt(int index) | Метод видаляє елемент списку studentList з індексом index. |
| QString getFullName(int index) const | Метод повертає поле Student::fullName елемету списка studentList з індексом index. |
| int getCourse(int index) const | Метод повертає поле Student::course елемету списка studentList з індексом index. |
| QString getGroup(int index) const | Метод повертає поле Student::group елемету списка studentList з індексом index. |
| QString getPhoneNumber(int index) const | Метод повертає поле Student::phoneNumber елемету списка studentList з індексом index. |
| const Student& getStudent(int index) const | Метод повертає елемет списка studentList з індексом index. |
| int getSizeOfStud() const | Метод повертає розмір списку studentList |
| void clearStudentAM() | Метод видаляє все елементи зі списка studentList. |
| void writeToXml() |  |
| ***private:*** | |
| QList<Student> studentList | Список інформації про зареєстрованих студентів. |
| QDomElement formDomElementNode(QDomDocument &domDoc, const Student& student) |  |

***3.1.9 Клас Student***

Клас Student є класом який зберігає інформацію про окремого студента, а також дозволяє перевірити чи всі поля класу заповнені. Код файлу визначення класу знаходиться в додатку Б. Дані та методи класу наведені у таблиці 3.9.

Таблиця 3.9 — Опис полів і методів класу Student.

|  |  |
| --- | --- |
| Поля та методи класу | Опис |
| **1** | **2** |
| ***public:*** | |
| QString fullName | Інформація про прізвище, ім’я, по-батькові. |
| QString group | Інформація про групу. |
| int course | Інформація про курс. |
| QString phoneNumber | Інформація про номер телефону. |
| Student() | Коструктор призначує полю course значення 0. |
| bool isComplete() | Метод повертає true якщо поля типу QString не є пустими, а cource не дорівнює 0. |

***3.1.10 Клас UnivercityAM***

Клас UnivercityAM є класом який зберігає інформацію про університети-партнери, а також дозволяє зчитати інформацію з \*.xml файлу, назва якого визначена в методі void readFromXml(). Код файлу визначення класу знаходиться в додатку Б. Дані та методи класу наведені у таблиці 3.10.

Таблиця 3.10 — Опис полів і методів класу UnivercityAM

|  |  |
| --- | --- |
| Поля та методи класу | Опис |
| **1** | **2** |
| ***public:*** | |
| const University& getUniversity(int index) | Метод поветає елемент списка universityList з індексом index. |
| QString getUnName(int index) const | Метод повертає назву поле University::univName елемету списка UniversityList з індексом index. |
| QString getMainInfo(int index) const | Метод повертає назву поле University::mainInfo елемету списка UniversityList з індексом index. |
| int getSizeOfUniv() const | Метод повертає розмір списку UniversityList. |
| void readFromXml(const QString language) | Метод зчитує інформацію про університети заданою мовою з файлу universitiesAM.xml і зберігає її в списку UniversityList. |
| ***private:*** | |
| QList<University> universityList | Список інформації про університети-партнери. |
| University& formUnivFromXml(const QDomNode& node) |  |

***3.1.11 Клас University***

Клас University є класом який зберігає інформацію про окремий університет (назву та детальну інформацію). Код файлу визначення класу знаходиться в додатку Б. Дані та методи класу наведені у таблиці 3.11.

Таблиця 3.11 — Опис полів і методів класу University.

|  |  |
| --- | --- |
| Поля та методи класу | Опис |
| **1** | **2** |
| ***public:*** | |
| QString univName | Поле, що зберігає інформацію про назву університета. |
| QString mainInfo | Поле, що зберігає детальну інформацію про університет. |

***3.1.12 Клас GeneralInfo***

Клас GeneralInfo є класом який зберігає інформацію про університети-партнери, а також дозволяє зчитати інформацію з \*.xml файлу, назва якого визначена в методі void readFromXml(). Код файлу визначення класу знаходиться в додатку Б. Дані та методи класу наведені у таблиці 3.12.

Таблиця 3.12 — Опис полів і методів класу GeneralInfo.

|  |  |
| --- | --- |
| Поля та методи класу | Опис |
| **1** | **2** |
| ***public:*** | |
| const University& getUniversity(int index) | Метод поветає елемент списка universityList з індексом index. |
| QString getUnName(int index) const | Метод повертає назву поле University::univName елемету списка UniversityList з індексом index. |
| QString getMainInfo(int index) const | Метод повертає назву поле University::mainInfo елемету списка UniversityList з індексом index. |
| int getSizeOfUniv() const | Метод повертає розмір списку UniversityList. |
| void readFromXml(const QString language) | Метод зчитує інформацію про університети заданою мовою з файлу universitiesAM.xml і зберігає її в списку UniversityList. |
| ***private:*** | |
| QList<University> universityList | Список інформації про університети-партнери. |
| University& formUnivFromXml(const QDomNode& node) |  |

***3.1.13 Клас Localization***

Клас Localization є класом який зберігає текст всіх елементів графічного інтерфейсу (кнопок, пунктів і підпунктів меню, назв вікон, лейблів) і регулярні виразі для перевірки даних, що водяться користувачем певною мовою. Всі тексти е.г.і. берутся з .csv файлу назва якого визначена в методі void fillTheLclValue(). Також в файлі Localization.h визначений перелік констант enum languages{UKRAINIAN, ENGLISH, GERMAN, RUSSIAN}. Вони використовуються в методі void setLang(const int langCode) оскільки значення параметру langCode визначається індексом обраної мови в комбо-боксі форми “Налашлування”, для зручності розуміння використовуються не цілочисельні літерали, а костанти, чиї назви дають чітке розуміння про мову, що була обрана користувачем. Код файлу визначення класу знаходиться в додатку Б. Дані та методи класу наведені у таблиці 3.13.

Таблиця 3.13 — Опис полів і методів класу Localization.

|  |  |
| --- | --- |
| Поля та методи класу | Опис |
| **1** | **2** |
| ***public:*** | |
| Localization() | Конструктор викликає метод void setLang(const int langCode) з параметром UKRAINIAN (тобто українська — мова за замовчуванням) і метод void fillTheLclValue(). |
| QString getLclChatbot() | Метод, що повертає текст “Чат-бот” певною мовою. |
| QString getLclSettings() | Метод, що повертає текст “Налаштування” певною мовою. |
| QString getLclHelp() | Метод, що повертає текст “Допомога” певною мовою. |
| QString getLclAboutprog() | Метод, що повертає текст “Про додаток” певною мовою. |
| QString getLclWhatisAM() | Метод, що повертає текст “Що таке АМ?” певною мовою. |
| QString getLclUnivpartn() | Метод, що повертає текст “Університети-партнери” певною мовою. |
| QString getLclNazad() | Метод, що повертає текст “Назад” певною мовою. |
| QString getLclContacts() | Метод, що повертає текст “Контакти” певною мовою. |
| QString getLclWantjoin() | Метод, що повертає текст “Бажаєте долучитися?” певною мовою. |
| QString getLclTheme() | Метод, що повертає текст “Тема” певною мовою. |
| QString getLclThemedark() | Метод, що повертає текст “Темна” певною мовою. |
| QString getLclThemelight() | Метод, що повертає текст “Світла” певною мовою. |
| QString getLclLanguage() | Метод, що повертає текст “Мова” певною мовою. |
| QString getLclHowtouse() | Метод, що повертає текст “Як користуватися?” певною мовою. |
| QString getLclRateapp() | Метод, що повертає текст “Оцінити додаток” певною мовою. |
| QString getLclFindbug() | Метод, що повертає текст “Знайшли баг?” певною мовою. |
| QString getLclMoredetails() | Метод, що повертає текст “Детальніше” певною мовою. |
| QString getLclDlyakogo() | Метод, що повертає текст “Для кого це?” певною мовою. |
| QString getLclKoshtovnist() | Метод, що повертає текст “Ціна” певною мовою. |
| QString getLclDepochaty() | Метод, що повертає текст “З чого почати?” певною мовою. |
| QString getLclSampleSOP() | Метод, що повертає текст “Приклад мотиваційного листа” певною мовою. |
| QString getLclbtnSend() | Метод, що повертає текст “Відправити” певною мовою. |
| QString getLclbtnSave() | Метод, що повертає текст “Зберегти” певною мовою. |
| QString getLclInputFIO() | Метод, що повертає текст “Введіть прізвище, ім’я, по-батькові” певною мовою. |
| QString getLclInputCourse() | Метод, що повертає текст “Введіть курс” певною мовою. |
| QString getLclInputGroup() | Метод, що повертає текст “Введіть групу” певною мовою. |
| QString getLclPhnNumber() | Метод, що повертає текст “Введіть номер телефону” певною мовою. |
| QString getLclErrorOcure() | Метод, що повертає текст “Виникла помилка, ви вже зареєстровані” певною мовою. |
| QString getLclSuccesReg() | Метод, що повертає текст “Дякую, реєстрація завершена” певною мовою. |
| QString getLclDevelLabel() | Метод, що повертає текст “Розробники:” певною мовою. |
| QString getLclDevelText() | Метод, що повертає текст “"Студенти НУ ""ЗП"", групи КНТ-229"” певною мовою. |
| QString getLclVersin( | Метод, що повертає текст “Версія: 0.3” певною мовою. |
| QString getLclFreeChatBot() | Метод, що повертає текст “Безкоштовний чат-бот, призначений для допомоги студентам і викладачам” певною мовою. |
| QString getLclGuideLable() | Метод, що повертає текст “Керівництво користувача” певною мовою. |
| QString getLclGuideText() | Метод, що повертає текст керівництва користувача певною мовою. |
| QString getLclBugReportLable() | Метод, що повертає текст “Знайшли баг або бажаєте щось повідомити розробнику? Напишіть сюди” певною мовою. |
| QString getLclDevMikeName() | Метод, що повертає текст “Ващенко Михайло” певною мовою. |
| QString getLclDevOlegName() | Метод, що повертає текст “Сепетий Олег” певною мовою. |
| QString getLclDevDimaName() | Метод, що повертає текст “Колбаса Дмитро” певною мовою. |
| QString getLclDevVladName() | Метод, що повертає текст “Момот Владислав” певною мовою. |
| void fillTheLclValue() |  |
| void setLang(const int langCode) | Метод, що встановлює поле langCode значення аргументу langCode і в залежності від цього значення встановлює значення полів language, regularExpressionName та regularExpressionGroup. |
| QString getLang() | Метод, що повертає значення поля language. |
| QString getRegExprName() | Метод, що повертає значення поля regularExpressionName. |
| QString getRegExprGroup() | Метод, що повертає значення поля regularExpressionGroup. |
| QString getRegExprCourse() | Метод, що повертає значення поля regularExpressionCourse. |
| QString getRegExprPhone() | Метод, що повертає значення поля regularExpressionPhone. |
| int getLangCode() | Метод, що повертає значення поля langCode. |
| ***private:*** | |
| QString language | Поле, що містить назву обраної мови. |
| int langCode | Поле, що містить індекс обраної мови в комбо-боксі форми “Налаштування”. |
| QString regularExpressionName | Поле, що містить регулярний вираз перевірки даних про прізвище, ім’я, по-батькові студента в залежності від обраної мови. |
| QString regularExpressionGroup | Поле, що містить регулярний вираз перевірки даних про групу студента в залежності від обраної мови. |
| const QString regularExpressionCourse = "[2-6]" | Поле, що містить регулярний вираз перевірки даних про курс студента. Оскільки регулярний вираз не залежить від обраної мови — поле є константним. |
| const QString regularExpressionPhone = "^\\+\\d{1,2}\\(\\d{3,5}\\)\\d{6,7}$" | Поле, що містить регулярний вираз перевірки даних про номер телефону студента, Оскільки регулярний вираз не залежить від обраної мови — поле є константним. |
| QStringList wordList | Поле, що містить тексти всіх е.г.і.. |

***3.1.14 Клас Exception***

***3.1.15 Клас FileException***

***3.1.16 Клас EmptyFileException***

***3.1.17 Клас ReadingDataException***

***3.1.18 Клас RegisterException***

***3.1.19 Клас Validate***

3.2 Основні розроблені алгоритми

3.2.1 Аналіз головного алгоритму

В ході написання курсової роботи був створений власний алгоритм складання розкладу.

Так і розроблений алгоритм, видає прийнятний розклад, але не гарантує, що він є найкращим серед можливих. Це позначає найголовнішу особливість алгоритму, а саме подальшу оптимізацію та можливе розширення виконуваних завдань.

Вхідними даними для алгоритму є:

* …..

Під час підрахунку остаточного результату алгоритм працює з проміжними даними:

* …..

У кінці виконання алгоритм формує вихідні данні:

* …..

Усі значення мають бути коректними та строго формалізованими (це виконується на етапі створення бази даних).

Увесь алгоритм можна поділити на 2 етапи:

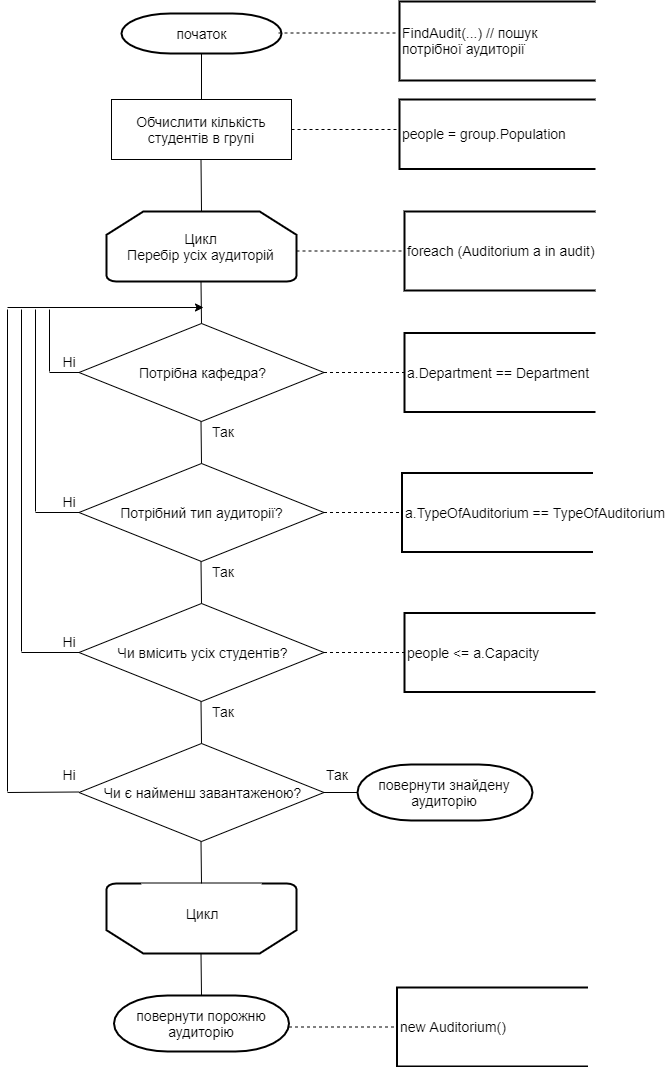
1. Формування списку занять;
2. Пошук часу проведення заняття.

Виконання кожного етапу базується та таких алгоритмах:

* алгоритм лінійного пошуку;
* алгоритм сортування вставками;
* елементи динамічного програмування.

Розглянемо основні типові алгоритми.

Опис розроблених алгоритмів

Рисунок 3.19 – Пошук відповідної аудиторії для заняття

….

Після виконання алгоритму маємо складений розклад для навчальних груп на потоках, аудиторій та викладачів.

3.3 Основні рішення щодо розробки інтерфейсу

3.3.1 Вибір платформи.

Головнним рішенням, щодо розробки інтрерфейсу, було вибрати нову та сучасну платформу для розробки інтерфейсу Qt. Ця платформа відзначаеться кросплатфоменістю та більш зручним інтерфейсом для програміста.

Опис рішень, які використовувались для реалізації інтерфейсу

3.3.2 Основні рішення з впорядкування елементів інтерфейсу

Під час розробки інтерфейсу користувача було використано наступні е.г.і.:

* MainWindow
* VHLayouts
* TextBrowser
* PushButton
* Label
* LineEdit
* MenuBar
* TextEdit
* Frame
* ComboBox

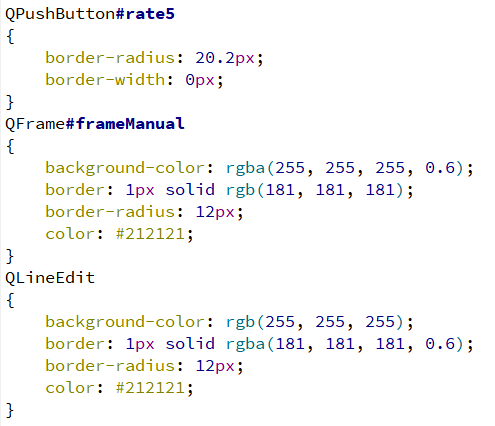
Саме вони були використані тому що, по перше, це одні з найпростіших та найпопулярніших е.г.і. в цілому, по друге, до цих елементів доволі просто застосовувати стилі та інші графічні зміни, по третє, ми вже маємо досвід роботи з цими е.г.і.

3.3.3 Основні рішення з комунікації з користувачем

Для кращої комунікації з користувачем та рівномірного розподілення місця у додатку були використані кнопки(PushButton) для запиту певної стандартної інформації у бота, та строка вводу (TextEdit, LineEdit) для нестандартної інформації такої як: особиста інформація користувача для надання ії керівникам АМ, або повідомлення про помилку\баг у додатку.

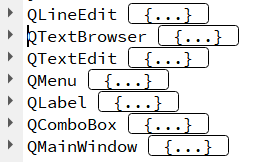
3.3.4 Основні рішення з створенням інтерфейсу

Було активно використано стилі. Тобто відокремлення певної групи елементів і створення для них власного стилю. Приклад данного рішення наведений нижче на рисунку 3.27.1…..

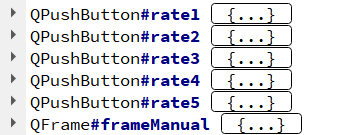


Основним варіантом застосування стилів є використання Qt Style Sheets. Як стає очевидно з назви, це трохи специфічний аналог звичних всім каскадних таблиць стилів CSS(Cascading Style Sheets), без яких немислимий сучасний інтернет. Оскільки в минулому семестрі у нас була дисципліна «WEB-дизайн» і ми активно використовували CSS в своїх роботах, то нам здалося зручним використовувати даний аналог і тут. Також це зручно в тому плані, що всі графічні рішення виводяться в окремий файл, не займають місце в основному коді і довантажуються під час запуску програми, а також потрібний стиль застосовується при виборі теми програми в Налаштуваннях.

Було використано звернення по класу. Це зручно тим, що зміни застосовуються відразу до всіх елементів даного класу. Приклад:



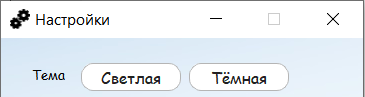
Звернення по імені дає можливість кастомізувати конкретний елемент з усього класу. Приклад:



Псевдо-стан елемента допомагає зробити програму більш динамічною і "живий" для користувача. Приклад:



Також було прийнято рішення додати повністю альтернативну тему для усього додатку, т.з. темний стиль зокрема від світлого. Ця ідея прийшла до нас у процесі створення програми вночі та ми подумали, що користувачеві буде зручно змінювати колірний стиль усього додатку одним кліком у Налаштуваннях.

3.4 Основні рішення щодо розробки бази даних

3.4.1 Основні рішення щодо вибору бази даних

База даних, як один з основних компонентів програми, потребує зваженого вибору середовища та платформи розробки, бо від цього залежатиме якість та комфорт не тільки у користуванні, а й у написанні програмного продукту, отже, саме тому основними критеріями для вибору БД були:

* …..

Отже, проаналізувавши наявні пропозиції у цій сфері, платформа …. вирізняється з-поміж інших варіантів, та відповідає усім заданим критеріям, а саме:…..

3.4.2 Структура збереження даних у БД

У процесі збереження даних використовуються таблиці, як ключові носії інформації, та представлені наступними екземплярами:

«*Auditorium*» - таблиця для збереження даних аудиторій, а саме: номер, літера, кількість місць, тип аудиторії та кафедра, якій вона належить (рис. 3.28).

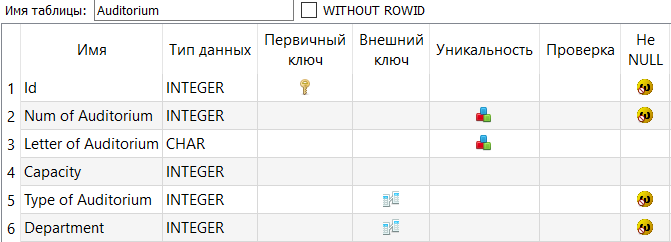


Рисунок 3.28 – Структура таблиці «Auditorium»

Первинним ключем є поле Id, яке ідентифікує кожен запис та допомагає зекономити пам'ять у інших таблицях, адже замість сполучення номеру, літери та кафедри використовується саме Id. Поля «Types of Auditorium» та «Department» пов’язані зовнішнім ключем з допоміжними таблицями для економії пам’яті та збереження цілісності даних, бо зовнішній ключ виступає перешкодою для внесення неіснуючих чи нерелевантних даних.

….

3.4.2 Основні запити SQL, що було використано

Мова для створення запитів SQlite є SQL-подібна, відрізняючись лише синтаксисом (несуттєво) та функціоналом. В ході роботи з базою даних були використані наступні запити для взаємодії:

* ….

4 керівництво програміста

В даному розділі розглянуті призначення, умови застосування, характеристика програми, звертання до програми, початкові та вихідні дані та представлені повідомлення.

4.1 Призначення та умови застосування програми

4.1.1 Призначення програми

Програма “Створення розкладу” призначена для автоматичного стоврeння розкладу на основі даних користувача.

4.1.2 Функції програми

Функції, що виконує програма:

* …..

4.1.3 Умови застосування програми

До комп’ютеру, на якому виконується програма, висуваються наступні вимоги:

* операційна система Windows 7, 8, 10;
* наявність дисплею;
* наявність миші та клавіатури;
* до 300 МБ оперативної пам’яті за умови використання функції створити розклад.

4.2 Характеристика програми

Програма виконана за допомогою мови програмування високого рівня С# в середовищі розробки Visual Studio 2017 , 2019 за допомогою платформи WPF.

4.2.1 Структура програми

Програма складається з 18 класу, 15 з яких – користувацькі, 3 – класи форм. Один з похідних класів інтерфейсу, яких ще 5, є один абстрактний клас. Проект містить наступні класи та їх реалізацію, файли ресурсів, базу даних SQL. Ієрархія представлена на рисунку 4.1.

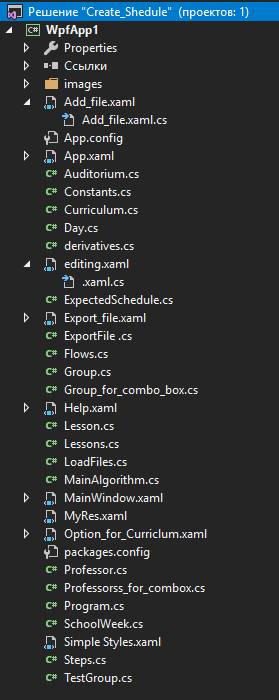


Рисунок 4.1 – Ієрархія класів проекту

4.2.2 Алгоритм роботи програми

….

4.3 Звертання до програми

Для запуску програми потрібно попередньо упевнитися в відповідності характеристик комп’ютера, на який встановлена програма, до системних вимог програми. Потрібно упевнитися у наявності всіх файлів програми.

Звертання до програми передбачене одним способом: через виконуваний файл Create\_Shedule.exe. Звертання до програми через командний рядок не передбачене, адже програма розроблена саме для роботи через інтерфейс для складання розкладу.

4.4 Вхідні та вихідні дані

4.4.1 Вхідні дані

Вхідні дані уявляють собою файли Екселю, з розширення .xlxs та .xlx. Вони повинні буди чотирьох різних типів. У кожному з яких за шаблоном повинні міститися дані в залежності від таблиці.

4.4.2 Вихідні дані

Файли Екселю з розширення .xlxs, .xlx. Вони повинні буди чотирьох різних типів. У кожній з яких за шаблоном повинні міститися дані в залежності від таблиці. Та є можливість зберегти файл з готовим розкладом.

4.5 Повідомлення

* ….

5 керівництво користувача

В даному розділі розглянуто призначення програми, умови її виконання, процес виконання програми та повідомлення для користувача.

5.1 Призначення програми

Програма “Створення розкладу” призначена для автоматичного стоврeння розкладу на основі даних користувача.

5.2 Умови виконання програми

5.2.1 Апаратні вимоги програми

До комп’ютеру, на якому виконується програма, висуваються наступні вимоги:

* наявність дисплею;
* наявність клавіатури та миші;
* до 300 МБ оперативної пам’яті за умови використання функції створення розкладу.

5.2.2 Вимоги до користувача

До користувачів висуваються наступні вимоги:

* …..

5.3 Виконання програми

5.3.1 Запуск програми

Для запуску програми потрібно попередньо упевнитися в відповідності характеристик комп’ютера, на який встановлена програма, до системних вимог програми. Другим кроком потрібно упевнитися в наявності всіх бібліотек для роботи програми та в наявності бази даних для повноцінного функціонування.

Звертання до програми передбачене одним способом: через виконуваний файл. Звертання до програми через командний рядок не передбачене, адже програма розроблена саме для візуалізації роботи алгоритмів пошуку найкоротшого шляху.

Після запуску виконуваного файлу Create\_Shedule.exe з’являється головна форма програми, що свідчить про початок роботи з програмою. Головна форма програми зображена на рисунку 5.1.

Рисунок 5.1 – Головна форма програми

5.3.2 Виконання роботи з програмою

Спершу користувач має ….

5.4 Повідомлення користувачу

….

5.5 Довідка програми

Для зручності користувача та для отримання допомоги українською мовою в будь-який момент часу було створено довідку для програми. Форма довідки зображена на рисунку 5.3.

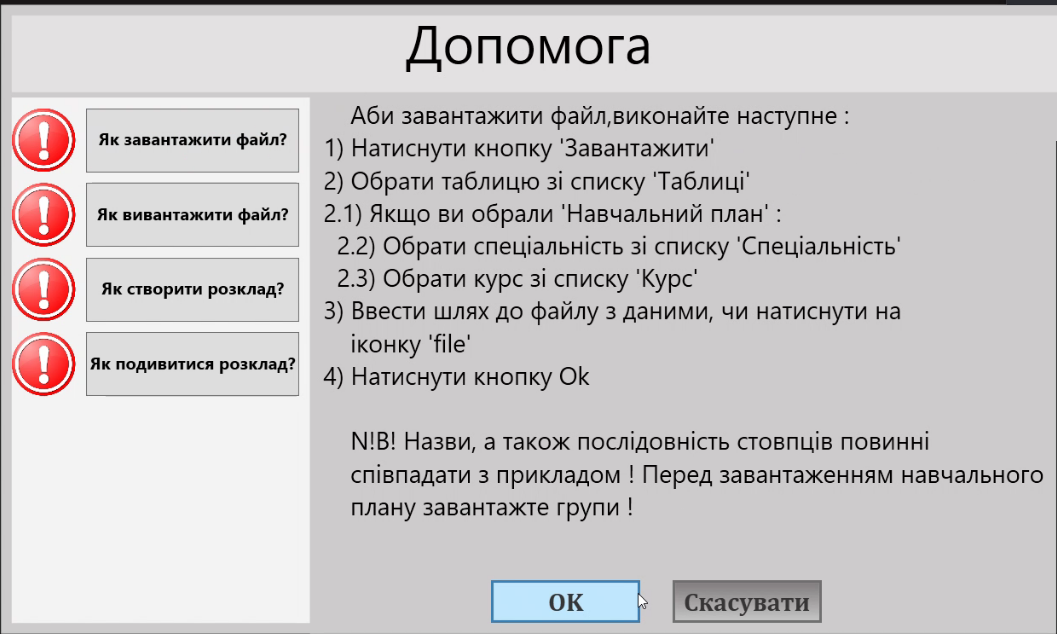


Рисунок 5.3 – Форма довідки

**ВИСНОВКИ**

**ПЕРЕЛІК ПОСИЛАНЬ**

**ДОДАТОК А ДІАГРАМА КЛАСІВ**

**ДОДАТОК Б КОД ПРОГРАМИ**

**ДОДАТОК В ІНТЕРФЕЙС ПРОГРАМИ**