Міністерство освіти і науки України

Національний університет «Запорізька політехніка»

Кафедра програмних засобів

ЗВІТ

Дисципліна «Фреймворки розробки програмного забезпечення»

Робота №2

Тема «Проєктування архітектури програмної системи»

Виконав варіант 19

Студент КНТ-122 Онищенко О. А.

Прийняли

Викладач Зелік О. В.

2024

МЕТА РОБОТИ

Вивчити базові принципи та основи дослідження предметної області, у межах якої розробляється програмне забезпечення, і навчитися виконувати проєктування архітектури системи на базі проведеного дослідження із застосування шаблону проєктування MVVM.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

Ознайомитися з літературою та основними теоретичними відомостями, необхідними для виконання роботи.

Провести аналіз предметної області, у рамках якої буде працювати розроблюване програмне забезпечення: виділити основні об’єкти, що зустрічаються у предметній області; виділити основні компоненти майбутньої системи; проаналізувати функції, які виконують виділені об’єкти предметної області.

На основні проведеного аналізу провести проєктування архітектури розроблюваної програмної системи з обов’язковим використанням шаблону проєктування Model-View-ViewModel (MVVM). В результаті проведеного проєктування має бути отриманий робочий варіант архітектури системи, що має включати: пакети, модулі, класи, бібліотеки тощо з описом призначення та функцій кожного з них у відповідності з принципами MVVM та принципами створення графічного інтерфейсу користувача на основі фреймворку WPF.

Виходячи з отриманої архітектури та описаних у ТЗ вимог до системи, виконати обґрунтований вибір засобів подальної розробки.

Виконати критичний аналіз отриманих результатів. За необхідності внести корективи у розроблену архітектуру або обрані засоби.

РЕЗУЛЬТАТИ ВИКОНАННЯ

Індивідуальне завдання можна розділити на такі кроки:

1. **Аналіз предметної області**: виділити об’єкти та функції
2. **Розробка архітектури**: розписати класи, компоненти, модулі
3. **Аналіз архітектури**: порівняти отриманий результат з питаннями

За кожним пунктом створено підрозділ нижче.

1 Аналіз предметної області

Індивідуальне завдання до проєкту

Програмне забезпечення продажу та придбання нерухомості. Всю нерухомість поділено на приватні будинки, окремі квартири та квартири в новобудовах. Приватні будинки та окремі квартири можуть виставлятися на продаж звичайними користувачами. Квартири в новобудовах можуть виставлятися лише спеціальними менеджерами. Користувачі можуть подавати заявки на перегляд житла, передивлятися стан заявок, а також після перегляду виставляти оцінку. Власники можуть передивлятися заявки на всі їх пропозиції або на обрані, підтверджувати перегляд або скасовувати, знімати пропозиції (кількість кімнат, площа, рік будівництва, зображення, планування, тощо), змінювати.

Ідентифікація об’єктів

Об’єкт нерухомості

Тип будинок / квартира / квартира в новобудові

Власник Користувач

Користувач

Менеджер так / ні

Заявки Заявка на перегляд

Об’єкти Об’єкт нерухомості

Обрані Об’єкт нерухомості

Заявка на перегляд

Статус очікує / переглянута / скасована

Оцінка ціле число

Функції об’єктів

Об’єкт нерухомості

Перегляд деталей

Зміна деталей

Зняття об’єкту (видалення зі списку на ринку)

Додавання об’єкту

Користувач

Перегляд об’єктів

Перегляд деталей

Зміна деталей

Перегляд заявок

Перегляд обраних заявок

Заявка

Додавання заявки

Перегляд деталей

Зміна деталей заявки

Оцінка заявки

2 Розробка архітектури

Після ідентифікації об’єктів предметної області та їх основних функцій можна перейти до розробки архітектури програмної системи використовуючи шаблон проєктування Model View ViewModel. Методи класів було розроблено за принципом CRUD (Create, Read, Update, Delete або Створити, Прочитати, Оновити, Видалити) що дозволяє проводити мінімальні необхідні операції з кожним з об’єктів.

```
class Listing: об'єкт нерухомості
  int id: унікальний ідентифікатор
  User owner: користувач власник
  string name: назва
  int price: вартість
  enum['Private' | 'Flat' | 'New'] kind: тип

  void addListing(string kind, string name, int price): додати до бази
даних
  Listing viewListing(Listing listing): переглянути
  void editListing(Listing listing): змінити
  void deleteListing(Listing listing): видалити
  void makeChosen(Listing listing, User user): додати до обраних

class User: користувач застосунком
  int id: унікальний ідентифікатор
  string name: ім'я
  bool manager: статус
  list[Listing] listings: виставлені об'єкти нерухомості
  list[Listing] chosen: обрані об'єкти нерухомості
  list[Meeting] meetings: зустрічі

  void addUser(string name, bool manager): додати до бази даних
  User viewUser(User user): переглянути
  void editUser(User user): змінити
  void deleteUser(User user): видалити

class Meeting: об'єкт зустрічі
  int id: унікальний ідентифікатор
  Listing listing: об'єкт нерухомості для перегляду
  User viewer: користувач переглядач
  int score: оцінка
  enum['Private' | 'Flat' | 'New'] status: статус

  void addMeeting(Listing listing, User viewer): додати до бази даних
  Meeting viewMeeting(Meeting meeting): переглянути
  void editMeeting(Meeting meeting): змінити
  void deleteMeeting(Meeting meeting): видалити
```

Дані мають бути організовані у відповідні таблиці бази даних MySQL з унікальними ідентифікаторами та полями відповідних назв, як зазначено у архітектурі.

```
TABLE ListingKind: тип нерухомості
  AUTONUM id: унікальний ідентифікатор
  VARCHAR kind: тип нерухомості Private або Flat або New

TABLE Listing: об'єкт нерухомості
  AUTONUM id: унікальний ідентифікатор
  FOREIGN KEY owner: користувач власник
  VARCHAR name: назва
  INT price: вартість
  FOREIGN KEY kind: тип

TABLE User: користувач застосунком
  AUTONUM id: унікальний ідентифікатор
  VARCHAR name: ім'я
  BOOL manager: статус
  FOREIGN KEY listings: виставлені об'єкти нерухомості
  FOREIGN KEY chosen: обрані об'єкти нерухомості
  FOREIGN KEY meetings: зустрічі

TABLE MeetingStatus: статус зустрічі
  AUTONUM id: унікальний ідентифікатор
  VARCHAR status: статус зустрічі Private або Flat або New

TABLE Meeting: об'єкт зустрічі
  AUTONUM id: унікальний ідентифікатор
  FOREIGN KEY listing: об'єкт нерухомості для перегляду
  FOREIGN KEY viewer: користувач переглядач
  INT score: оцінка
  FOREIGN KEY status: статус
```

Система має включати основні бізнес правила для проведення операцій над об’єктами.

```
addListing(string kind, string name, int price): відправити об'єкт до
бази, перевірити змінні
viewListing(Listing listing): відфільтрувати усі об'єкти, отримати
необхідний
editListing(Listing listing): отримати потрібний об'єкт, перевірити
коректність значень
deleteListing(Listing listing): перевірити коректність змінних,
отримати потрібний об'єкт
makeChosen(Listing listing, User user): перевірити коректність змінних,
отримати необхідні об'єкти

addUser(string name, bool manager): відправити користувача до бази,
перевірити коректність змінних
```

```
viewUser(User user): відфільтрувати список користувачів, отримати
потрібного
editUser(User user): отримати потрібного користувача, перевірити
коректність введених значень
deleteUser(User user): отримати необхідного користувача, перевірити
коректність значення

addMeeting(Listing listing, User viewer): перевірити коректність
введеня значень
viewMeeting(Meeting meeting): відфільтрувати об'єкти зустрічей та
отримати необхідний
editMeeting(Meeting meeting): отримати необхідний об'єкт зустрічі
deleteMeeting(Meeting meeting): отримати потрібний об'єкт зустрічі
```

Бізнес-правило `makeChosen` передбачає взаємодію з двома типами об'єктів. Це може нести потенційні безпекові ризики. Усі бізнес-правила, пов'язані з видаленням об'єктів, передбачають безпекові ризики та необхідність створення резервних копій об'єктів для запобігання втрати даних.

Графічний користувацький інтерфейс має проєктуватися за принципами простоти у пріоритеті та доступності. Користувацький інтерфейс має включати світлий фон для убезпечення користувачів. Інтерфейс має бути розроблено окремими компонентами для забезпечення модульності усіх об’єктів.

```
void addListingView: вікно додавання оголошення
void viewListingView: вікно перегляду оголошення
void editListingView: вікно редагування оголошення
void deleteListingView: вікно видалення оголошення

void addUserView: вікно додавання користувача
void viewUserView: вікно перегляду користувача
void editUserView: вікно редагування користувача
void deleteUserView: вікно видалення користувача

void addMeetingView: вікно додавання зустрічі
void viewMeetingView: вікно перегляду зустрічі
void editMeetingView: вікно редагування зустрічі
void deleteMeetingView: вікно видалення зустрічі
```

Стратегія введення-виведення даних передбачає взаємодію кількох шарів застосунку. На початку кожного такого процесу користувач

взаємодіє з графічним інтерфейсом (View) та вводить необхідні дані. Після введення даних користувачем шар бізнес-логіки (ViewModel) отримує та перевіряє і можливо перетворює дані для їх подальшої передачі до шару даних. Після роботи шару бізнес-логіки шар даних (Model) отримує запит на дані або зміну даних якщо було обрано операцію введення. При успішному закінченні роботи шар даних передає повідомлення на шар бізнес-логіки, шар бізнес-логіки опрацьовує запит та надає відповідь на шар представлення. Такий підхід передбачено шаблоном проєктування Model View ViewModel.

Для забезпечення кращої швидкодії системи для роботи з обмеженими ресурсами пам’яті та обчислювальної потужності процесора системи було прийнято рішення використовувати лише один потік процесора та відповідно один запит за раз до системи керування базами даних. Такий підхід обгрунтовується обмеженістю обчислювальних потужностей робочих машин та прагненням забезпечення максимальної швидкодії системи без компромісів безпеці та можливості неперервної роботи.

Для убезпечення архітектури має бути передбачений обмежений доступ до функцій системи в залежності від значення типу авторизованого користувача. Такий підхід необхідний для убезпечення системи від несанкціонованих дій або надання доступу до компонентів користувачам, рівень доступ яких є обмеженим в порівнянні з іншими.

Масштабованість системи має бути забезпечення шляхом виділення усіх необхідних функцій відповідним шарам системи та використання потрібних компонентів. Шляхом використання такого підходу розробки системи можна досягнути можливості створення нових компонентів на основі попередньо розроблених через повторне використання коду програмного продукту.

Обсяг усіх компонентів системи має бути у вищезазнченому порядку, але певно буде адаптуватись під потреби користувачів та час розробників під час процесу розробки програмної системи. Швидкодія усіх компонентів має бути реалізована та протестована у процесі розробки програмного забезпечення.

Для обробки помилкових ситуацій у системі має бути передбачено використання необхідних відповідних вбудованих можливостей системи розробки програмного забезпечення WPF. Через використання необхідних компонентів можна досягти чіткого процесу комунікації помилок, їх змісту та потенційних шляхів вирішення проблемних ситуацій до користувача системи.

Реалізація надлишкової функціональності та її можлива імплементація має бути розглянута по завершенню мінімального робочого продукта (Minimal Viable Product або MVP). Після реалізації такого продукту систему можна вдосконалювати шляхом додавання нових функцій або рефакторингу існуючих.

Усі компоненти системи мають бути створені з нуля для забезпечення адекватного розуміння взаємодії системи.

Повторно використаний код може бути адаптований до інших аспектів системи оскільки уся система побудована на основі методології розробки Create Read Update Delete, що передбачає базові операції створення, перегляду, оновлення та видалення кожного з об’єктів. Виходячи з цього система має працювати приблизно однаково продуктивно у всіх її шарах.

3 Аналіз архітектури

Чи ясно описана загальна організація програми? Так

Чи включає специфікація грамотний огляд архітектури та її обґрунтування? Так

Чи адекватно визначено основні компоненти програми, їх області відповідальності й взаємодія з іншими компонентами? Так

Чи наведено опис найважливіших класів і їх обґрунтування? Так

Чи наведено опис організації даних і її обґрунтування? Так

Чи наведено опис організації й змісту сховища даних? Так

Чи визначені всі важливі бізнес-правила? Так

Чи описано їх вплив на систему? Так

Чи описана стратегія проєктування GUI? Так

Чи зроблено GUI модульним, щоб його зміни не впливали на іншу частину програми? Так

Чи наведено опис стратегії введення-виведення даних та її обґрунтування? Так

Чи вказано оцінки ступеню використання обмежених ресурсів, таких як потоки, з'єднання зі сховищем даних, дескриптори, пропускна спроможність мережі? Так

Чи наведено опис стратегії керування такими ресурсами і її обґрунтування? Так

Чи описані вимоги до захищеності архітектури? Так

Чи визначає архітектура вимоги до обсягу й швидкодії всіх класів, підсистем і функціональних областей? Так

Чи описує архітектура спосіб досягнення масштабованості системи? Так

Чи розглянуто питання взаємодії системи з іншими системами? Ні

Чи описана стратегія інтернаціоналізації або локалізації? Ні

Чи визначена погоджена стратегія обробки помилок? Так

Чи визначений підхід до відмовостійкості системи (якщо потрібно)? Ні

Чи підтверджена можливість технічної реалізації всіх частин системи? Так

Чи визначений підхід до реалізації надлишкової функціональності? Так

Чи прийняті необхідні рішення відносно "придбання або створення" компонентів системи? Так

Чи описано у специфікації як повторно використовуваний код буде адаптований до інших аспектів архітектури? Так

Чи зможе архітектура адаптуватися до ймовірних змін? Так

ВИБІР ПРОГРАМНИХ ЗАСОБІВ

Після отримання архітектури програми, враховуючи вимогу розробки із застосуванням шаблону проєктування MVVM та фреймворку розробки програмного забезпечення WPF, програмні засоби для розробки застосунку такі:

- **Мова програмування** C#
- **Середовище розробки** Microsoft Visual Studio та Visual Studio Code
- **База даних** MySQL