**Міністерство освіти і науки України**

**Національний університет «Запорізька Політехніка»**

Кафедра програмних засобів

**♰ ЗВІТ**

Практична робота №2

Дисципліна “Фреймворки розробки програмного забезпечення”

Тема “Розробка технічного завдання”

**Виконав**

Студент КНТ-122 О. А. Онищенко

**Прийняли**

Викладач О. В. Зелік

2024

**Так бо БОГ полюбив світ, що дав Сина Свого Єдиного, аби кожен, хто вірує в Нього, не згинув, але мав життя вічне** ([Йоан 3:16](https://bolls.life/UTT/43/3/16/))

Завдання 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 тільки спеціальними менеджерами. Користувачі можуть подавати заявки на перегляд житла, передивлятися стан заявок, а також після перегляду виставляти оцінку. Власники можуть передивлятися заяявки на всі їхні пропозиції або на обрані, підтверджувати перегляд або скасовувати, знімати пропозиції (кількість кімнат, площа, рік будівництва, зображення, планування, тощо), змінювати.

Мета роботи

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

Результати

Архітектура програми

Model

**User**: для обробки користувацьких даних та функцій

- string name: ім'я користувача

- string password: пароль користувача

- string type: *user* | *manager* – тип користувача

- List<Listing> listings: список власних оголошень користувача

- string name: назва оголошення

- bool own: *true* якщо оголошення належить користувачеві

- bool selected: *true* якщо оголошення обране

- function loadUsers(): завантажує користувачів

- function saveUsers(): зберігає користувачів

- function getUser(name): шукає користувача за іменем

**Meeting**: для обробки даних зустрічей та функцій

- string client: ім'я користувача клієнта зустрічі

- string seller: ім'я користувача продавця оголошення

- Listing listing: вибране оголошення для перегляду

- string status: pending | approved | canceled – статус зустрічі, змінюється продавцем

- string score: good | great | well – оцінка зустрічі клієнтом

- function loadMeetings(): завантажує зустрічі

- function saveMeetings(): зберігає зустрічі

- function getMeeting(listing): шукає зустріч за назвою оголошення

**Listing**: для обробки даних оголошень та функцій

- string owner: ім'я користувача власника оголошення

- string name: назва оголошення

- string type: *house* | *flat* | *condo* – тип оселі

- int rooms: *1* | *2* | *3* – кількість кімнат

- int area: *12* | *17* | *23* – площа оселі

- int decade: *00* | *10* | *20* – десятиліття побудови оселі

- string style: *Biblical* | *Missionary* | *Monastic* – стиль оселі

- string photo: шлях до зображення оголошення

- function loadListings(): завантажує оголошення

- function saveListings(): зберігає оголошення

- function getListing(name): шукає оголошення за назвою

View

MenuView: головний контейнер для всіх подань

- List<Listing> listings: глобальний список оголошень

- List<Meeting> meetings: глобальний список зустрічей

- List<User> users: глобальний список користувачів

ListingsView: сітка оголошень з описом та зображенням

MeetingsView: сітка зустрічей з деталями

ProfileView: інформація профілю користувача і оголошення

ViewModel

function openListings(listings): відкриває перегляд списків

function openMeetings(meetings): відкриває перегляд зустрічей

function openProfile(users): відкриває перегляд профілю

function checkUser(user): перевіряє правильність облікових даних користувача

function registerUser(user): реєструє нового користувача; перевіряє чи користувач вже існує

function addListing(listing): додає нове оголошення

function deleteListing(listing): видаляє оголошення

function editListing(listing): редагує оголошення

function addMeeting(meeting): додає нову зустріч

function deleteMeeting(meeting): видаляє зустріч

function editMeeting(meeting): редагує зустріч

function rateMeeting(meeting, score): оцінює зустріч

Взаємодія між View та Model відбувається через компоненти ViewModel. Взаємодія між View та Model безпосередньо заборонена.

Робота з даними проводиться через базу даних. Функції View відповідних класів взаємодіють з базою аби створювати, редагувати, чи видаляти потрібні об’єкти. Всі дані мають відповідати вищезазначеним типам. Наразі наявні лише зображення для оголошень, прикріплено їх архівом зі словами “Listing Images” у назві. Дані пов’язані між собою через зовнішні ключі у базі даних. Дані інтегруються у програму через базу даних і зберігаються як копія у файлах XML локально.

Безпека виконується через дотримання типів даних, відокремлення взаємодії з даними та функціями для інтерфейсу.

В результаті, дасть БОГ, матимемо систему окремих компонентів інтерфейсу, компонентів для роботи з даними, та компонентів посередників для взаємодії інтерфейсу з даними. Робота буде виконуватися у середовищі Visual Studio на мові C# з використанням MVVM, WPF, XML. Для розширення системи можна додавати нові компоненти через класи чи подання інтерфейсу. Для розширення кількості користувачів можна забезпечити надійну роботу бази даних. Для розширення обсягу даних треба створити копію даних з бази у файли XML. Такі рішення були прийняті через їх розповсюдженість і підтримку серед спільноти.

Продуктивність буде забезпечена через використання сучасних засобів розробки і чіткого відокремлення турбот. Відмовостійкість може бути забезпечена через обробку виключень.

Через те що компоненти інтерфейсу і даних не залежні один від одного, то можна їх використовувати повторно у різних ситуаціях.

Аналіз архітектури

Загальна організація програми описана ясно. Грамотний огляд архітектури присутній. Також обгрунтування є.

Основні компоненти програми визначені адекватно, також і їх область відповідальності та взаємодія з іншими.

Опис найважливіших класів наведено, обгрунтування є.

Опис організації даних наведено, обгрунтування також.

Чи наведено опис організації й змісту сховища даних?

Чи визначені всі важливі бізнес-правила? Чи описано їх вплив на систему?

Чи описана стратегія проєктування GUI?

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

Чи наведено опис стратегії введення-виведення даних та її обгрунтування?

Чи вказано оцінки ступеню використання обмежених ресурсів: потоки, з’єднання зі сховищем даних, дескриптори, пропускна спроможність мережі? Чи наведено опис стратегії керування такими ресурсами і її обгрунтування?

Чи описані вимоги до захищеності архітектури?

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

Чи описує архітектура спосіб досягнення масштабованості системи?

Чи розглянуті питання взаємодії системи з іншими системами?

Чи описана стратегія інтернаціоналізації/локалізації?

Чи визначена погоджена стратегія обробки помилок?

Чи визначений підхід до відмовостійкості системи, якщо потрібно?

Чи підтверджена можливість технічної реалізації всіх частин системи?

Чи визначений підхід до реалізації надлишкової функціональності?

Чи прийняті необхідні рішення відносно «придбання або створення» компонентів системи?

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

Чи зможе архітектура адаптуватися до ймовірних змін?

Обгрунтування вибору засобів розробки

Висновки

По виконанню цієї роботи було вивчено базові принципи та основи дослідження предметної області, в межах якої розробляється програма, і навчено виконувати проєктування архітектури системи на базі проведеного дослідження із застосування шаблону проєктування MVVM.

Контрольні питання

Що таке архітектура програми?

Загальний *дизайн* системи, регулює як *компоненти* мають взаємодіяти. *Дизайн*: планування вигляду та роботи речей. *Компонент*: частина чогось більшого.

Яке місце проєктування та розробка архітектури програми в життєвому циклі?

Має бути здійснено на початку, після збору й аналізу вимог до програми.

Які основні складові архітектури?

Опис самих компонент системи (класи, модулі, пакети, бібліотеки), як ті компоненти мають взаємодіяти, опис можливих патернів проєктування, обмежень.

Які вимоги до архітектури програми?

Має бути чітко визначеною, повно описувати компоненти, відповідати на всі зазначені питання з методички.

Як визначити якість архітектури?

Якщо воно логічно описано і працює на практиці, то зроблено якісно.

Скільки часу відносно загального розробки має займати розробка архітектури?

Можливо близько 17% від загального часу розробки.

Як можна описати архітектуру?

Через специфікацію архітектури, де описуються всі можливі компоненти і прийняті рішення.

Які є архітектурні шаблони?

Бувають Model-View-ViewModel, Model-View-Controller.

Які є основні принципи проєктування архітектури?

Зрозумілість, масштабованість, повнота.

Які є архітектурні стилі?

Можуть бути клієнт-серверними, пошаровими, мікросервісними, подієвими.

Що таке шаблон проєктування MVVM, які складові у нього?

*Шаблон*: готовий макет для підлаштування і використання. *Шаблон проєктування*: для розробки програм. *Model View ViewModel*: шаблон проєктування для відокремлення *UI* від функціоналу. *UI*: User Interface або користувацький інтерфейс.

*Model*: обробка даних і бізнес-логіки – функціоналу.

*View*: елементи *UI*, виведення даних з *Model*.

*ViewModel*: посередник між *View* і *Model* – функції для *View* аби взаємодіяти з *Model*.

Яка різниця між шаблоном MVVM від інших?

*Model View ViewModel*: відокремлює UI від функціоналу.

*Client-Server*: сервер надає послуги клієнтам через запити.

*Layered*: компоненти розташовані шарами і комунікують між ними.

*Microservice*: система складена з маленьких сервісів, поєднані через *API*. *API*: Application Programming Interface, посередник між двома програмами для надання інформації, наприклад.

*Event-Driven*: компоненти реагують на події, взаємодії користувача наприклад.

Тому різниця між MVVM та іншими підходами велика, вони всі різні і для різних задач.