**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

Основні компоненти: аж до класів.

Всі функції з чітким визначенням, як окремий клас, метод, сукупність методів/класів, чи як інша складова.

Правила взаємодії між складовими

Організація взаємодії і її заборона якщо треба

Формат роботи з даними

Вимоги до даних

Наявні наразі дані

Принципи взаємодії з даними

Шляхи інтеграції даних

Обробка безпеки

Результати

Прийняті рішення

Як розширюватись система може: додавати нові компоненти

Розширення це має бути функціональне, кількості користувачів, обсягу даних.

Вимоги до продуктивності

Забезпечення відмовостійкості: обробка виключень.

Способи повторно ивкористати існуючі речі під час розробки

Аналіз архітектури

Чи ясно описана загальнае організація програми? Чи включає специфікація грамотний огляд архітектури та її обгрунтування?

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

Чи наведено опис найважливіших класів і їх обгрунтування?

Чи наведено опис організації даних і її обгрунтування?

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. Алелуя Слава ІСУС ХРИСТУ.

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.