# РЕАЛІЗАЦІЯ ІНТЕРАКТИВНОГО-КОМПОНЕНТНОГО ПЛАНУВАННЯ ЗАВДАНЬ

## Засоби та технології розробки

Вибір технологій для розробки застосунку для мобільних пристроїв залежить від головних критеріїв: вимоги системи та прискорення процесу розробки.

Вибір технологій залежить додатково від факторів:

* Детальна документація технологій та продукту (платформи) для розробки, що дозволяє пришвидшити час розробки програмного продукту.
* Легкість та простота у використанні інструментів продукту (платформи) для розробки.
* Мова програмування, що задовольняє вимогам для створення програмного продукту
* Функціональність, що задовольняє функціональним вимогам програмного продукту.
* Продуктивність, що задовольняє вимогам до програмного продукту з продуктивності, забезпечення мінімальних потреб роботи програмного продукту.

При розробці інформаційної системи інтерактивного-компонентного планувальника завдань використовувались:  
3 мови програмування, 4 фреймворка (бібліотеки), 2 платформи розробки, 1 реляційна база даних та інше. Перелік використованних засобів та платформ наведені у наступній таблиці.

|  |  |
| --- | --- |
| Тип | Назва |
| Мова програмування | Swift, PHP, SQL, |
| Фреймворки | SwiftUI, UIKit, TelegramSDK, Foundation |
| IDE | Xcode, DBeaver |
| RDBMS | MySQL |
| Засіб проектування | Figma |
| Система контроля версій | Git |
| Репозиторій | GitHub |

Таблиця 2 – технології

Git є системою керування версіями, яка забезпечує можливість відстежувати та керувати змінами в файловій системі під час розробки програмного забезпечення. Дозволяє кільком розробникам спільно працювати над проектом одночасно, зберігаючи та контролюючи різні версії файлів зі змінами.

### Платформа та технології додатку до мобільного пристрою iOS

Для розробки мобільного застосунку (додатку) необхідно використовувати інструменти та платформи, що мають можливість проектувати інтерфейс користувача, використовувати бібліотеки та додатки, и мати можливість зборки завантажувального файлу до мобільного пристрою. Платформи з розробки мобільного застосунку поділяються на:

- Інструменти та платформи, що дозволяють розробити застосунок для мобільних платформ iOS і Android одночасно, наприклад, React Native - дозволяє використовувати JavaScript для розробки мобільних додатків. Flutter - використовуючи мову програмування Dart, дозволяє створювати додатки для iOS та Android зі спільним кодом. Xamarin - дозволяє розробляти мобільні додатки для iOS, Android і Windows, використовуючи мову програмування C#.

- Інструменти та платформи, що надаються власниками платформ мобільних пристроїв. Для платформи iOS -Xcode (UIKit та SwiftUI) головні інструменти для розробки мобільних додатків. Використовуються мови Swift і Objective-C, що надають повний доступ до функцій і можливостей, пропонованих платформою Apple. Для платформи Android -Android Studio (Android SDK) головний інструмент для розробки Android застосунку. Використовується мова програмування Java або Kotlin. Забезпечується доступ до функцій та можливостей платформи Android.

Використання при розробки платформ Xcode, Android Studio надає найвищу продуктивність та повний контроль над функціоналом мобільних пристроїв, але потребує використання окремої мови програмування до кожної мобільної платформи.

Переваги розробки на платформах Xcode, Android Studio:

* Максимальна продуктивність та ефективність розробки на платформах.
* Повний доступ до специфічних функцій і можливостей кожної платформи.
* Інтеграція з програмними бібліотеками та технічними засобами кожної платформи та мобільними пристроями.

Переваги розробки на платформах, що дозволяють розробити застосунок до Xcode, Android Studio одночасно:

* Загальний код для різних платформ, що скорочує час розробки.
* Швидка розробка та здатність оновлювати версії застосунків одночасно для обох (кількох) платформ.

Недоліки розробки на платформах, що дозволяють розробити застосунок до Xcode, Android Studio одночасно:

* Менша продуктивність програмних продуктів, порівняно з додатками, розробленими на Xcode, Android Studio.
* Обмеження доступа до специфічних функцій та техінчних засобів на платформах iOS, Android.

Обрано розробку для платформи iOS, використовуючи платформу Xcode (технології Apple), що дозволяють використовувати засоби швидкої розробки інтерфейсу користувача. Розробка програмного продукту має забезпечити сумісність з функціоналом і стандартами платформи iOS.

**Мова програмування.** Swift та Objective-C - основні мови програмування, що використовуються для розробки застосунків на платформі Xcode для iOS. До переваг та недоліків мов платформи Xcode відноситься:

1. Синтаксис і зручність використання:
   * Мова Swift має сучасний, експресивний синтаксис, який робить код більш зрозумілим та зручним для розробника. Мова підтримує безпечну типізацію, функціональне програмування та інші сучасні парадигми програмування.
   * Мова Objective-C використовує класичний синтаксис C з додатковими розширеннями до об'єктно-орієнтованого програмування. Мова має складний синтаксис.
2. Сумісність та наявність кодової бази:
   * Swift є мовою, яка активно розвивається та підтримується Apple. Існуюча кодова база на Objective-C має можливість інтеграції з Swift.
   * Мова Objective-C основна для розробки на платформі iOS для створення: бібліотек, фреймворків та додатків.
3. Продуктивність та швидкодія:
   * Мова Swift забезпечує швидкість розробки та виконання додатків. Мова забезпечує вбудовану оптимізацію кода та має простий доступ до специфічних операцій, що дозволяє розробляти швидкодіючі застосунки.
   * Objective-C має складну структуру завантаження функцій та класів, складні умови використання. Швидкість обумовлено методами та архітектурою розробленого додатку.
4. Підтримка та розвиток:
   * Мова Swift активно розвивається Apple і має велику підтримку спільноти розробників. Мова отримує оновлення та нові функції, що полегшують розробку та покращують продуктивність.
   * Objective-C підтримується Apple.

Порівняльний аналіз надає перевагу розробки додатку до інтерактивного-компонентного планувальника завдань - мові Swift.

**SwiftUI | UIKit.** SwiftUI та UIKit - основні фреймворки (бібліотеки) для розробки інтерфейсу користувача (UI) для мобільних пристроїв та компьютерів Apple, таких як iPhone, iPad та Mac:

1. UIKit:

* офіційний фреймворк для розробки UI під iOS
* Використовує мову програмування Objective-C або Swift.
* Має широкий набір готових елементів інтерфейсу, таких як кнопки, тексти, таблиці, колекції тощо, що надає розробникам велику гнучкість у створенні власних UI-елементів.

1. SwiftUI:

- Новий офіційний фреймворк для розробки UI під iOS

* Використовує мову програмування Swift.
* Легкий в створені інтерфейсу користувача, за допомогою декларативного підходу. Простий процес розробки.
* Має широкий набір вбудованих компонентів та модифікаторів, що дозволяють швидко створювати складні інтерфейси.
* Може бути використаний для розробки UI для iOS, macOS, watchOS та tvOS.

Порівняльний аналіз надає перевагу розробки додатку до інтерактивного-компонентного планувальника завдань - мові Swift на фреймворку (бібліотеки) SwiftUI.

Взаємодія додатка до мобільного пристрою iOS з іншими додатками в системі інтерактивного-компонентного планувальника завдань представлено на **СХЕМІ 1**

### Система компонентів користувача системи для мобільного додатка iOS.

Система компонентів до мобільного додатка iOS включає розробку програмних бібліотек, що реалізують функціонал вимог до реалізації системи інтерактивного-компонентного планувальника завдань:

* Пакети завдань, компонент додатку
* Задачі, що входять до пакету, компонент додатку
* Компоненти, хо входять до задач, компонент додатку

Взаємодія між компонентами додатка до мобільного пристрою iOS в системі інтерактивного-компонентного планувальника завдань представлено на **СХЕМІ 1**

### Платформа бази даних та технології API додатку сервера

Платформою для бази даних використовується реляційна СКБД MySQL за рахунок легкості розгортання, підтримці у хмарах та хостінгах. MySQL - це система керування базами даних (СКБД), яка надає засоби для зберігання, організації та управління великими обсягами даних. Платформа надає можливість використовувати мову запитів SQL (Structured Query Language) для взаємодії з базою даних.

Мовою програмування на сервері для доступа до СКБД MySQL використовується мова PHP за рахунок легкості створення додатків, простого доступа до баз даних та підтримці у хмарах та хостінгах.

Взаємодія між базою даних MySQL та API додатку сервера в системі інтерактивного-компонентного планувальника завдань представлено на **СХЕМІ 1**

### Чат бот системи повідомлень. Додаток Swift для організації повідомлень.

Чат бот системи повідомлень у вигляді додатку Swift для організації повідомлень для користувачів системи інтерактивного-компонентного планувальника завдань використовує систему компонентів користувача системи для мобільного додатка iOS (у вигляді компонент - бібліотек).

Чат бот повідомлень взаємодіє з платформою Telegram API для надсилання інформаційних повідомлень системи до Telegram каналу користувача у вигляді Telegram повідомлень.

**Мова програмування.** Використовується мова програмування Swift для розробки бота та імпорт бібліотек компонентів (розроблених для додатка iOS).

**API для створення бота.** Для створення телеграм-бота на Swift використовується: Telegram Bot API **-** офіційний API, наданий Telegram, який дозволяє взаємодіяти з ботами через HTTP-запити.

Взаємодія з чат ботом системи повідомлень в системі інтерактивного-компонентного планувальника завдань представлено на **СХЕМІ 1**

## Реалізація додатка користувача до мобільного пристрою iOS

Реалізація застосунку забезпечила наступні архітектурні рішення:

* Організація окремих модулів
* Взаємодія модулів та передача даних між ними
* Навігація між модулями
* Впровадження залежності модуля

### Організація окремого модуля

В системі використовується модифікований підхід “MVP” (Model-View-Presenter). Він дозволяє змінити логіку вікна, окремо протестувати, стає більш читабельним, дає гарантії розробнику, що сутність “View” відповідає тільки за інтерфейс, а сутність “Presenter” тільки за логічну її частину. “View” робить три головні дії: верстає весь дизайн, перехвачує і віддає назовні всі інтрефейсі події, реалізує певний набір інструментів, який можна використовувати, щоб керувати інтерфейсом ззовні

“Presenter” реагує на всі події і залежно від вимог, реалізує логіку екрану. За допомогою інструментів що дає “View”, “Presenter” може дати команди змінити інтерфейс в залежності від логіки.

“Model” це будь-який набір даних в модулі. Набір даних може змінюватись з часом, актуальні дані завжди лежать у “Presenter”. “View” може мати свою версію “Model”, для того, щоб реалізовувати інтерфейс

A diagram of a model

Description automatically generated with low confidence

Рисунок 3.1 — Схема MVP

## Шари програми

Програма поділена на шари, для того, щоб завжди знати де що знаходиться

A picture containing text, screenshot, font, line

Description automatically generated

Рисунок 3.2 — iOS шари програми

На вищому рівні в розробці для iOS існує навігація, яка базується на модулях і формує дерево програмного потоку. Одним з ключових елементів цієї навігаційної логіки є координатори, які відповідають за керування модулями та визначають їх послідовність. Використання координаторів є стандартною практикою в iOS.

Головним принципом цієї архітектури є те, що бізнес-логіка залежить лише від інтерфейсів користувача, що означає, що її інтерфейс можна замінити не торкаючись логіки.

Мережевої взаємодії відокремлена в окремий шар, який використовує лише бібліотеки Apple для цієї цілі.

### Навігація

Координатори (Coordinators) є популярним архітектурним підходом в iOS-розробці, який допомагає керувати навігацією та координацією функціональних модулів додатку. Вони дозволяють розділити відповідальність за навігацію між різними екранами та модулями додатку.

Основна ідея полягає в тому, що кожна логічна частина програми має свій власний координатор, який відповідає за керуванням навігацією та роботою з цим конкретним модулем. Координатори спрощують створення, ініціалізацію та взаємодію модулів.

Основні переваги використання координаторів включають:

1. Відокремлення відповідальностей: Координатори дозволяють відокремити відповідальність за навігацію та координацію між модулями. Це полегшує розширення та тестування окремих модулів.
2. Перевикористання та модульніст**ь**: Координатори можуть бути легко перевикористані в інших частинах додатку або навіть в інших проектах. Вони покращують модульність та розширюваність додатку.
3. Спрощення тестування: Координатори полегшують тестування окремих модулів, оскільки їх можна відокремити від решти додатку та незалежно тестувати їх функціональність.

A picture containing text, line, font, diagram

Description automatically generated

Рисунок 3.3 — Схематичне зображення навігаційного дерева iOS застосунку

У рамках архітектури навігації, всі елементи навігаційної структури утворюють дерево, в якому кожен вузол може мати зв'язки з іншими вузлами, що дозволяє переходити до наступного вузла або повертатися назад. Координатори визначають логіку цієї навігаційної структури.

A picture containing text, diagram, plan, line

Description automatically generated

Рисунок 7 - Генеральна схема навігації

На схемі можно помітити два головні типи координторів: ті, що об’єднують тілька вікна і ті, що об’єднують інші координатори, забезпечують повну незалежність і перевикористовуємість окремих координторів. Перший тип у свою чергу забеспечую повну незалежність окремих вікон.

### Впровадження залежностей у модулі

Для забезпечення впровадження залежностей в iOS-розробці існує три основних підходи: Service Locator, Dependency Injection (DI) і фабричний метод.

Dependency Injection (DI) є процесом передачі залежностей об'єкту ззовні, замість внутрішнього створення цих залежностей всередині об'єкта. Цей підхід забезпечує більшу гнучкість, оскільки об'єкт не залежить від конкретної реалізації своїх залежностей і може легко замінювати їх іншими реалізаціями. В проекті також створюється клас з фабричними методами, який збирає окремі модулі, а модулі передаються через інтерфейс збирача, що реалізує DI. Коли створюється модуль, він передає себе як реалізатор контракту.

Service Locator (локатор служб) - це патерн проектування, який використовується в iOS-розробці для централізованого доступу до служб (сервісів) у додатку. Він надає механізм, який дозволяє об'єктам отримувати доступ до потрібних служб без прямого знання про їх реалізацію або деталі їх створення. Service Locator в iOS може бути реалізований за допомогою спеціального об'єкта-локатора, який містить посилання на реалізації служб і надає методи для отримання цих служб.

Фабричний метод (Factory Method) - це метод, який збирає сутність і повертає її екземпляр. Він активно використовується в поєднанні з DI та SL.

У системі використовується фабричний метод, через те, що немає глобальних тестів, але іноді потрібно мати модулі, які потрібно, якимось чином збирати.

## Реалізація бази даних та API додатку сервера

## Реалізація чат боту системи повідомлень

## Аналіз допоміжних програмних засобів та засобів розробки

Xcode - це IDE (інтегроване середовище розробки), яке, можна сказати, є єдиною опцією для розробки під пристрої Apple. Хоча воно не є найкращим IDE у світі, останній конкурент, AppCode від JetBrains, недавно припинив своє існування. Крім того, лише Xcode дозволяє розміщувати додатки в App Store.

DBeaver - це простий безкоштовний редактор для тестування запитів до MySQL та інших баз даних. Він дуже зручний у використанні.

Simulator - це інструмент, що дозволяє симулювати роботу iPhone на ноутбуку. Він розроблений компанією Apple і використовується для тестування програм на різних моделях пристроїв.

## Висновок до розділу

У розділі, я детально пояснив правила та принципи написання коду. Пояснив, які технології і чому буду використовувати. Порівняв їх, один з одним. Розповів про архітектуру та методології в застосунку. Пояснив, де я буду писати системи і чому саме там