# 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 стандарт розробки програмного забезпечення “MVP” (Model-View-Presenter) Лит. Це дозволяє змінити логіку роботи вікна інтерфейсу користувача, гнучко протестувати та наближає до стандартів створення програмного забезпечення. Сутність компоненти “View” відповідає тільки за інтерфейс користувача, а сутність компоненти “Presenter” тільки за логічну її частину. Компонента “View” забезпечує три головні дії: Компонує дизайн, перехвачує та оброблює назовні події інтерфейсу користувача, реалізує набір інструментів, який використовується для керування інтерфейсом ззовні.

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

Компонента “Model” - це будь-який набір даних в модулі. Набір даних змінюється з часом, актуальні дані лежать у компоненті “Presenter”. Компонента “View” має свою версію “Model”, для того, щоб реалізовувати інтерфейс користувача.

На рисунку 3.1 представлено схему роботи стандарту MVP, що розроблено в програмному забезпеченні додатка користувача до мобільного пристрою iOS

A diagram of a model

Description automatically generated with low confidence

Рисунок 3.1 — Схема стандарту MVP в програмному забезпеченні додатка користувача до мобільного пристрою iOS

* + 1. Шари програми

Програмне забезпечення додатка користувача до мобільно пристрою iOS розділене на шари, щоб забезпечити незалежнне функціюнування окремих частин програми.

На рисунку 3.2 представлено шари програми, що розроблено в програмному забезпеченні додатка користувача до мобільного пристрою iOS

A picture containing text, screenshot, font, line

Description automatically generated

Рисунок 3.2 — Шари програми в програмному забезпеченні додатка користувача до мобільного пристрою iOS

Шар програми навігаціязастосунку знаходиться на вищому рівні, який базується на розроблених модулях і формує дерево програмного потоку данних. Головним елементом навігації є координатори, які відповідають за керування модулями та визначають їх послідовність.

Координатори дозволяють розділити відповідальність за навігацію між екранами користувача та модулями додатку. Кожна логічна частина програми має власний координатор, який відповідає за керування навігацією та роботою з цим конкретним модулем. Координатори спрощують створення, ініціалізацію та взаємодію модулів в програмі.

Основними перевагами використання координаторів в програмі є:

1. Відокремлення відповідальностей: Координатори дозволяють відокремити відповідальність за навігацію та координацію між модулями. Це забезпечує легке розширення та тестування окремих модулів.
2. Перевикористання та модульніст**ь**: Координатори забезпечують легкість перевикористання в інших частинах додатку або в інших проектах. Координатори покращують модульність та розширюваність додатку.
3. Спрощення тестування: Координатори забезпечують легкість тестування окремих модулів, за допомогою властивості відокремлення від решти додатку та незалежного тестування їх функціональності.

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

На схемі представлено два головні типи координторів в програмі: ті, що об’єднують вікна інтерфейса користувача і ті, що об’єднують інші координатори та забезпечують повну незалежність і можливість окремі координатори повторно. Перший тип забезпечує повну незалежність окремих вікон програмного забезпечення користувача.

**Мережевої взаємодії** в програмному забезпеченні відокремлена в окремий шар, який використовує стандартні бібліотеки Apple для системи iOS.

**Впровадження залежностей у модуль.** В програмному забезпеченні використовується спеціалізований метод (fabric method) для того, щоб збирати окремі модулі програми. Цей підхід забезпечує збірку компонентів модулів програмного забезпечення в єдину програму без додаткових реалізацій та перевизначення модулів.

Блок схема алгоритму роботи додатку до мобільного пристрою iOS наведено на **СХЕМІ 2**

Алгоритм роботи програми забезпечує завантаження базового функціоналу:

- завантаження пакетів користувача з диску пристроя з представленням можливостями додавання та корегування данних.

- завантаження меню налаштувань додатка з пунктами про дадаток і допомога.

Алгоритм роботи додатку до мобільного пристрою iOS дозволяє обробляти та налагоджувати пакети користувача, отримувати допомогу та іншу інформацію.

Робота з пакетами користувача в програмі дозволяє завантажити магазин з додатковими пакетами задач для отримання або обмін задачами між окремими користувачами системи. Для роботи в магазині необхідно виконати авторизацію користувача системи.

* 1. Реалізація бази даних та API додатку сервера

Реалізація бази даних на сервері MySQL виконано у хмарі на хостингу за допомогою виконання файлів SQL в середовищі веб застосунка phpAdmin (входить до стандартного набору інструментів cPanel для роботи з базами даних у хмарі). Для розгортання бази даних та завантаження тестових даних виконані наступні кроки:

- Реєстрація в хмарі з автоматичним розгортанням серверу бази даних

- Створення схеми бази даних в середовищі веб застосунка phpAdmin

- Створення користувача бази даних з повноваженнями адміністратора за допомогою файлу SQL (proq\_user.sql) в середовищі веб застосунка phpAdmin

- Створення таблиць бази даних та завантаження тестових даних за допомогою файлу SQL (proq-schema.sql  та proq\_data.sql) в середовищі веб застосунка phpAdmin

Таблиці базі даних системи інтерактивного-компонентного планувальника завдань надають додаткам системи зберігати дані про:

* Користувачів системи
* Магазини для зберігання пакетів задач
* Пакети задач
* Компоненти задач
* Довідкові та інформаційні дані до об’єктів системи

Схема таблиць бази даних системи інтерактивного-компонентного планувальника завдань представлено на **СХЕМІ 2**

Таблиці бази даних системи розподіляються на: основні (для зберігання даних користувачів, магазинів, пакетів, задач та компонент), допоміжні (для організації входження даних пакетів до магазину, задач до пакетів та компонент до задач), службові (для організації додаткових даних системи або основних об’єктів даних) та довідкові.

Перелік основних таблиць, що включає базові об’єкти системи:

* Користувач (User)
* Магазин (Store)
* Пакет (Package)
* Задача (Task)
* Компонент (Component)

Перелік допоміжних таблиць, що допомагає організувати входження базових об’єктів один до одного:

* Пакет магазину (Store package)
* Задача пакету (Package task)
* Компонент задачі (Task component)

Службові та Довідкові таблиці:

* Системні установки (Settings)
* Додаткові дані об’єктів (Options)
* Довідник мов (Language)

Робота з сервером бази даних mySQL, отримання та зберігання даних в таблицях системи виконує додаток API (з використанням стандартних методів та бібліотек доступу до баз даних на мові PHP).

Запити до даних додаток API приймає по протоколу HTTP за методами POST та GET з використанням рекомендацій технології REST API (**ЛІТЕРАТУРА**).

Додаток API складається з наступних модулів на мові PHP:

* Основний завантажувальний файл /service/api/index.php
* Файл доступу до бази даних /service/api/database/repository.php (Реалізація класу Repository)
* Допоміжний файл обробки даних /service/api/includes/functions.php
* Набір спеціалізованих файлів для отримання та зберігання даних в таблицях системи /service/json/modules/json/\*.php, що дозволяє виконати: отримання даних користувача, автентифікацію (або авторизацію) користувача, отримання даних пакетів магазин, тощо

Блок схема алгоритму роботи додатка API сервера наведено на **СХЕМІ 3**

Довжина тимчасового ключа доступу (токену) до бази даних через додаток API дорівнює 64 байта, закодовані в форматі BASE64 (**ЛІТЕРАТУРА**).

Час роботи тимчасового ключа доступу встановлюється з таблиці Системні установки (Settings) за допомогою спеціального параметра - access\_token\_expired\_time.

Часовий пояс для отримання дат з таблиць системи встановлюється з допомогою спеціального параметра з таблиці Системні установки (Settings) - timezone\_offset.

Приймання даних за протоколом HTTP:

* метод GET через перелік набору параметрів ключ = значення
* метод POST через структуру даних за рекомендацій технології JSON

Відправлення відповідей за методами POST та GET: через структуру даних за рекомендацій технології JSON

Перелік помилок з кодами в роботі додатка API наведено нижче:

-1001 Помилка контенту запита

-1002 Не знайдено імені модуля

-1003 Не знайдено файлу модуля

-1004 Не встановлено з'єднання з базою даних

-1005 Авторизацію не виконано

-1006 Користувача не активовано

-1007 Термін дії ключа авторизації минув

-1008 Помилка роботи ключа авторизації

-1009 Помилка ім'я користувача або пароля

-1010 Помилка ключа оновлення

-1011 Невідома помилка при обробці даних

-1100 Помилка бази даних

* 1. Реалізація чат боту системи повідомлень

Реалізація чат боту системи повідомлень виконано у хмарі на Google Cloud за допомогою виконання файлів Swift в середовищі операційної системи Linux (входить до стандартного набору для роботи з додатками у хмарі). Для розгортання додатку чат боту системи повідомлень виконані наступні кроки:

* + Реєстрація в хмарі з автоматичним розгортанням операційної системи Linux
  + Створення середовища для завантаження на виконання додатків Swift

Додаток чат бота системи повідомлень завантажує на виконання наступний реалізований функціонал:

* обробник базових команд чат бота: додати та видалити пакети завдань, допомога, тестування, отримати пакети завдань
* обробник завдань та компонентів з встановленого списку пакетів завдань.
* відправник інформаційних повідомлень до Telegram

Блок схема алгоритму роботи додатка чат бота системи повідомлень наведено на **СХЕМІ 4**

Час активації роботи обробника завдань та компонентів - кожна 1 секунда.

Робота з командами бота та відправлення інформаційних повідомлень за допомогою Telegram Bot API.

* 1. Допоміжні програмні засоби та засоби розробки

Xcode - платформа розробки з IDE (інтегроване середовище розробки). Використання для створення Додатку до мобільного пристрою iOS та Додатку чат бота інформаційних повідомлень системи.

DBeaver - редактор для підключення та завантаження на виконання файлів SQL та окремих запитів до бази даних MySQL. Редагування та видалення даних з таблиць бази даних.

Simulator - інструмент для симуляції роботи мобільного пристрою iPhone на комп'ютері. Використовується для тестування програм моделях мобільних пристроїв.

## Висновок до розділу

В результаті розробки інтерактивного-компонентного планувальника завдань створено повно-функціональну систему вводу, зберігання та обробки даних користувача з наданням простого інтерфейсу для роботи, системою інформаційних повідомлень. Розроблено наступні програмі засоби та компоненти:

* База даних для зберігання та обробки службової та персональної інформації користувача системи
* Створено тестові дані для проведення необхідних тестів в системі
* Створено Додаток API сервера для організації доступу до бази даних системи
* Створено Додаток до мобільного пристрою iOS користувача системи
* Створено Додаток чат бота для інформаційних повідомлень системи

Розроблена система показала можливості розвитку систем, де необхідно спланувати та забезпечити **організацію часу людини.**

Простий і гнучкий інтерфейс користувача Додатків та система може слугувати базовою моделлю для розробки складних на навантажених систем для роботи груп та плануванні часу окремої людини. Використання гнучких чат ботів для отримання інформаційних повідомлень надає напрямок розвитку для систем оповіщення при настанні події в системах, в яких виконуються задачі в режимі реального часу.