# АРХІТЕКТУРА СИСТЕМИ

## Формалізоване подання процесу планування завдань

### Вимоги до компонентів системи

##### Вимоги до типів компонентів та їх обробники:

Необхідне створення n типів компонентів + n1 типів обробників + n2 реалізацій типів обробників. Де n = n1 = n2.

##### Вимоги до компонентів:

Необхідне створення m компонентів + m1 вхідних даних до них + m2 реалізацій цих компонентів. Де m = m1 = m2.

### Вимоги до пакетів завдань та алгоритмів роботи з пакетами завдань

##### Вимоги до завдання:

Необхідно, щоб кожне завдання - task[n] мало можливість мати m компонентів, де компонент може мати під собою k типів компонентів, які у свою чергу мають k1….kN обробників, які в свою чергу мають l…lN обробників типів компонентів.

##### Вимоги до пакету завдань:

Необхідне створення та зберігання n пакетів для m користувачів. Кожен пакет може мати k завдань.

##### Вимоги до магазина пакетів:

Магазин повинен зберігати кількість n пакетів. Користувач має мати можливість завантажувати пакети до сховища даних – n.push (newPackage), та видаляти n.deleteAt (packageIndex) пакети зі сховища.

#### Вимоги до програми:

Програма має надавати можливість завантажувати для кожного окремого користувача n[userIndex], k кількість пакетів.

##### Вимоги до обробника показати (Appear) компоненти:

Кожну секунду потрібно робити перевірку чи є завдання y1…yN в пакеті, завдання входить до пакету q[packageIndex] для користувача n[userIndex]. Кожен компонент повертає логічне значення (Boolean) кожну секунди - потрібно показувати (Так/Ні)

##### Вимоги до обробника даних (Data) компонентів:

Для кожного компоненту k, за умови що він дата (Data) компонент в завдані z, за умови, що він повинен активуватись у час T та отримати строку даних. Відобразити масив [k] у завданні.

##### Вимоги до обробника інтерактивності (Interactive) компонентів:

Для кожного обробника (Interactive) компоненту згенерувати тег у форматів «буква» та «цифра» - a1, a2, a3 ….a50…aN. Для кожного компоненту k, за умови що він інтерактивний (Interactive) компонент в завдані z, за умови що він повинен активуватись у час T та отримати строку даних. Додати масив [k] у завдання. Для кожного компонента k надати можливість дати відповідь. Повернути логічне значення (Boolean) як результат відповіді (Так/Ні).

### Вимоги до реалізації компонентів

##### Вимоги до інтервалу подій (Interval) компонентів:

Кожен компонент k перевіряє який тип інтервалу подій – дні тижня (WeekDays) чи дні (Days). Для днів тижня (WeekDays) – чи сьогодні той день d, який зазначено у масиві днів, в яких необхідно активізувати завдання та показати компонент, наприклад масив WeekDays[dayToAppearIndex1 … dayToAppearIndexN]. Компонент перевіряє час події (Interval): чи значення (Interval) актуально та наступив час події.

##### Вимоги до опису (Description) компонентів:

Для кожного компонента k створюється строка або строки з детальним описом призначення компонента.

### Обґрунтування методів та засобів створення програми для плануванні завдань та повідомлень.

Мова програмування Swift є високорівневою мовою програмування, розробленою з урахуванням простоти та ефективності розробки програм. У порівнянні з мовами C або C++, Swift не надає програмісту прямого доступу до структур даних, подібних до “масиву" даних (як для С/С++), які забезпечують швидкий пошук елементів завдяки простим операціям додавання i + index, а також не вимагають виділення (allocate) пам'яті мобільних пристроїв, що може бути складним та процесом, що вимагає час для операційної системи.

Однак, масив у Swift оптимізований та має схожу реалізацію до “вектора”, як для С++. Це означає, що масив у Swift може динамічно збільшуватись, що є критично важливим у випадку, коли не можливо передбачити кількість користувачів або пакетів у системі. Гнучкість є важливою та критичною властивістю в таких випадках. Для досягнення більшої оптимізації, накладаються вимоги належним чином встановити властивість розміру (capacity) для масиву даних, щоб мінімізувати виділення (allocate) нової пам'яті, коли розмір масиву наближається до критичної межі, оскільки в Swift масив (Array) автоматично розширює обсяг виділеної пам'яті.

У випадках, коли послідовність даних не має значення, накладається вимога використовувати множину Swift (Set) для збереження даних в пам'яті. Множина Swift (Set) є більш швидшою, ніж масив Swift (Array) і може використовується для зберігання пакетів у чат боті повідомлень користувача.

Окремо накладається вимога на використання словник даних “хеш-мап” Swift (Dictionary). Для програмного забезпечення бота повідомлень користувача, ключем буде ідентифікатор користувача у системі Telegram, а значенням - сам пакет завдань. Використання словника (Dictionary) є оптимальним засобом при роботі програми для відправлення повидомлень.

## Архітектура

Архітектура процесу планування завдань складається з наступних елементів:

* магазин пакетів (які формує користувачі системи)
* завдання користувача, які формуються в пакеті (які розміщуються в магазині пакетів)
* компоненти завдання, які формуються для кожного завдання та мають опис роботи та набір інших параметрів (Додаток 1).

На Рисуноку 2.1 представлено скорочену схему підпорядкованості архітектури процесу планування завдань.



Рисунок 2.1 — Скорочена схема підпорядкованості

## Постановка задачі і функцій при створенні інтерактивного-компонентного планувальника завдань

### Аналіз вимог

Головними вимогами для створення системи планувальника завдань є:

* Надання можливості користувачу створювати завдання, вказувати опис час, інші парламенти роботи.
* Надання можливості зберегти створене завдання на пристрої та в базі даних.
* Надання можливості отримувати повідомлення (вчасні) про стан або іншу інформацію про завдання.

Вимоги до задач і функцій з планування завдань потребують створення основних функцій програмного забезпечення: налаштування, створення пакетів задач, розміщення або завантаження пакетів задач з магазину задач. Використання основних функцій в програмному забезпеченні клієнта планувальника завдань наведено на Рисунку 2.2

A picture containing text, diagram, screenshot, line

Description automatically generated

Рисунок 2.2 — Використання основних функцій в програмному забезпеченні клієнта планувальника завдань

В наступних таблицях наведено варіанти використання основних функцій програми клієнта планувальника завдань.

Таблиця 2.1 – UC1

|  |  |
| --- | --- |
| Use case name | Відкриття допомоги |
| Goals | Надати користувачу доступ до допомоги в застосунку |
| Actors | Користувач |
| Trigger | Користувач натискає кнопку обирає з меню налаштувань Onboarding |
| Pre-conditions | Зайти до налаштувань |
| Flow of Events | Користувач натискає кнопку налаштування і обирає з меню налаштувань Onboarding |
| Extension | - |
| Post-Condition | Запустився Onboarding з поясненнями |

Таблиця 2.2 – UC2

|  |  |
| --- | --- |
| Use case name | Створити пакет |
| Goals | Надати можливість створювати нові «болванки» пустих пакетів |
| Actors | Користувач |
| Trigger | Користувач натискає кнопку додати пакет |
| Pre-conditions | - |
| Flow of Events | Користувач натискає мнопку налаштування, вводить назву |
| Extension | - |
| Post-Condition | Створюеться пустий пакет |

Таблиця 2.3 – UC3

|  |  |
| --- | --- |
| Use case name | Створити задвання і запонити пакет |
| Goals | Надати можливість наповнювати пакети завданнями |
| Actors | Користувач |
| Trigger | Користувач натискає додати задання |
| Pre-conditions | Бути у пакеті |
| Flow of Events | Користувач заходить до пакету, натискає додати задання, додає компоненти до завдання |
| Extension | - |
| Post-Condition | Створюеться завдання в пакеті |

Таблиця 2.4 – UC4

|  |  |
| --- | --- |
| Use case name | Створити задвання і запонити пакет |
| Goals | Надати можливість наповнювати пакети завданнями |
| Actors | Користувач |
| Trigger | Користувач натискає додати задання |
| Pre-conditions | Бути у пакеті |
| Flow of Events | Користувач заходить до пакету, натискає додати задання, додає компоненти до завдання |
| Extension | - |
| Post-Condition | Створюеться завдання в пакеті |

Таблиця 2.5 – UC5

|  |  |
| --- | --- |
| Use case name | Видаляти пакети |
| Goals | Надати можливість видаляти пакети завданнями |
| Actors | Користувач |
| Trigger | Свайп вліво, натиснути кнопку видалити |
| Pre-conditions | - |
| Flow of Events | Свайп вліво, натиснути кнопку видалити |
| Extension | - |
| Post-Condition | Пакет видаляється і зникає |

Таблиця 2.6 – UC6

|  |  |
| --- | --- |
| Use case name | Видаляти завдання |
| Goals | Надати можливість видаляти завдання |
| Actors | Користувач |
| Trigger | Свайп вліво, натиснути кнопку видалити |
| Pre-conditions | Бути у пакеті |
| Flow of Events | Свайп вліво, натиснути кнопку видалити |
| Extension | - |
| Post-Condition | Завдання видаляється і зникає |

Таблиця 2.7 – UC7

|  |  |
| --- | --- |
| Use case name | Видаляти копомнент |
| Goals | Надати можливість видаляти завдання |
| Actors | Користувач |
| Trigger | Свайп вліво, натиснути кнопку видалити |
| Pre-conditions | Бути у задавнні |
| Flow of Events | Свайп вліво, натиснути кнопку видалити |
| Extension | - |
| Post-Condition | Завдання видаляється і зникає |

Таблиця 2.8 – UC8

|  |  |
| --- | --- |
| Use case name | Авторизація |
| Goals | Надати можливість бути авторизованим і дати доступ до загрузки своїх пакетів до магазину |
| Actors | Користувач |
| Trigger | Викласти пакет |
| Pre-conditions | - |
| Flow of Events | Користувач заходить до магазину і пробує викласти свії пакет до магазину, вилазить авторизація, користувач авторизується |
| Extension | - |
| Post-Condition | Користувач авторизувався, авторизація зникає |

Таблиця 2.9 – UC9

|  |  |
| --- | --- |
| Use case name | Копіювання пакету |
| Goals | Надати можливість вибрати пакет і скопіювати його собі до аккаунту |
| Actors | Користувач |
| Trigger | Свайп вліво, кнопка скопіювати |
| Pre-conditions | - |
| Flow of Events | Користувач заходить до магазину, вибирає пакет, свайп вліво, кнопка скопіювати |
| Extension | - |
| Post-Condition | Користувач скопіював пакет, він з'явився у головному меню, мін може його редагувати |

Таблиця 2.10 – UC10

|  |  |
| --- | --- |
| Use case name | Викладка пакету |
| Goals | Надати можливість викладати пакету до магазину |
| Actors | Користувач |
| Trigger | Нажати + у магазині |
| Pre-conditions | Бути авторизованим |
| Flow of Events | Зайти до магазину, натиснути кнопку +, вибрати пакет |
| Extension | - |
| Post-Condition | Пакет загружається до серверу, кожен інший користувач тепер може його побачити і скопіювати |

Вимоги до задач і функцій з інформаційними повідомленнями про заплановані завдань потребують створення основних функцій програмного забезпечення: надати або видалити пакет завдань зі списку, отримувати наявні пакети завдань, отримувати інформаційні повідомлення про завдання. Використання основних функцій в програмному забезпеченні бота інформаційних повідомлень планувальника завдань наведено на Рисунку 2.3

A diagram of a person

Description automatically generated with low confidence

Рисунок 2.3 — Використання основних функцій в програмному забезпеченні бота інформаційних повідомлень

Таблиця 2.11 – UC11

|  |  |
| --- | --- |
| Use case name | Додати пакет |
| Goals | Надати можливість додавати пакет |
| Actors | Користувач |
| Trigger | Команда add package |
| Pre-conditions | Бути у боті |
| Flow of Events | Закинути в бота пакет, реплаем зробити команду add |
| Extension | - |
| Post-Condition | Пакет додається до боту |

Таблиця 2.12 – UC12

|  |  |
| --- | --- |
| Use case name | Видалити пакет |
| Goals | Надати можливість видаляти пакет |
| Actors | Користувач |
| Trigger | Команда rm package\_index |
| Pre-conditions | Бути у боті |
| Flow of Events | Команда rm package\_index |
| Extension | - |
| Post-Condition | Пакет видаляється з боту |

Таблиця 2.13 – UC13

|  |  |
| --- | --- |
| Use case name | Список пакетів |
| Goals | Надати можливість дивитися список пакетів |
| Actors | Користувач |
| Trigger | Команда pls |
| Pre-conditions | Бути у боті |
| Flow of Events | Команда pls |
| Extension | - |
| Post-Condition | Показує все пакети в боті |

Таблиця 2.14 – UC14

|  |  |
| --- | --- |
| Use case name | Надсилання повідомлення |
| Goals | Надати можливість надсилати повідомлення |
| Actors | Користувач |
| Trigger | Один з «Appear» компонентів повернув True |
| Pre-conditions | Додати до бота пакет |
| Flow of Events | Час приходить для показу задвання, приходить повідомлення на телефон з завданням |
| Extension | - |
| Post-Condition | Повідомлення прийшло |

### Функціональні вимоги

Основними функціональними вимогами до задач і функцій пакетів планувальника завдань є: Допомога, Створення пакетів завдань, Редагування або видалення пакетів завдань, Створення завдань, Редагування або видалення завдань, Створення компонентів завдань, Редагування або видалення компонентів завдань, Завантаження та перегляд пакетів завдань з магазину. Нижче наведено таблиці в переліком основних функціональних вимог для задач і функцій.

Таблиця 2.15 – FR-1

|  |  |
| --- | --- |
| Назва | Надати можливість надавати допомогу |
| Опис | Система повинна надати можливість надавати допомогу |

Таблиця 2.16 – FR-2

|  |  |
| --- | --- |
| Назва | Надати можливість створювати пакети завдань |
| Опис | Система повинна надати можливість створювати пакети завдань |

Таблиця 2.17 – FR-3

|  |  |
| --- | --- |
| Назва | Надати можливість редагувати або видалити пакети |
| Опис | Система повинна надати можливість редагувати пакети, а саме: змінювати назву пакету, додавати нові завдання, видаляти, редагувати завдання |

Таблиця 2.18 – FR-4

|  |  |
| --- | --- |
| Назва | Надати можливість створювати завдання |
| Опис | Система повинна надати можливість створювати завдання, а саме: давати назву завданню, додавати компоненти, видаляти, редагувати компоненти |

Таблиця 2.19 – FR-5

|  |  |
| --- | --- |
| Назва | Надати можливість редагувати або видалити завдання |
| Опис | Система повинна надати можливість редагувати завдання, а саме: давати назву завданню, додавати компоненти, видаляти, редагувати компоненти |

Таблиця 2.20 – FR-6

|  |  |
| --- | --- |
| Назва | Надати можливість створювати компонент завдання |
| Опис | Система повинна надати можливість створювати окремий екземпляр компонента, його налаштування |

Таблиця 2.21 – FR-7

|  |  |
| --- | --- |
| Назва | Надати можливість редагувати або видалити компонент завдання |
| Опис | Система повинна надати можливість редагувати або видаляти окремий екземпляр компонента, його налаштування |

Таблиця 2.22 – FR-8

|  |  |
| --- | --- |
| Назва | Надати можливість переглядати пакети магазину |
| Опис | Система повинна надати можливість переглядати всі пакети магазину |

Таблиця 2.23 – FR-9

|  |  |
| --- | --- |
| Назва | Надати можливість завантажувати пакети магазину |
| Опис | Система повинна надати можливість завантажувати пакети завдань з або до магазину |

Функціональна вимога FR- 9

Основними функціональними вимогами до системи повідомлень пакетів планувальника завдань є: Управління пакетів завдань, Отримувати пакети, Надсилання повідомлень. Нижче наведено таблиці в переліком основних функціональних вимог для задач і функцій системи повідомлень.

Таблиця 2.24 – FR-10

|  |  |
| --- | --- |
| Назва | Надати можливість управління з пакетами в боті |
| Опис | Система повинна надати можливість управління з пакетами в боті, а саме: додавати і видаляти |

Таблиця 2.25 – FR-11

|  |  |
| --- | --- |
| Назва | Надати можливість отримувати активні пакети в боті |
| Опис | Система повинна надати можливість отримувати активні пакети в боті |

Таблиця 2.26 – FR-12

|  |  |
| --- | --- |
| Назва | Надати можливість отримувати повідомлення |
| Опис | Система повинна надати можливість надсилати повідомлення |

Таблиця 2.27 – Матриця трасування вимог

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | FR-1 | FR-2 | FR-3 | FR-4 | FR-5 | FR-6 | FR-7 | FR-8 | FR-9 | FR-10 | FR-11 | FR-12 |
| UC-1 | + |  |  |  |  |  |  |  |  |  |  |  |
| UC-2 |  | + |  |  |  |  |  |  |  |  |  |  |
| UC-3 |  |  | + |  |  |  |  |  |  |  |  |  |
| UC-4 |  |  |  | + |  |  |  |  |  |  |  |  |
| UC-5 |  |  | + |  |  |  |  |  |  |  |  |  |
| UC-6 |  |  |  |  | + |  |  |  |  |  |  |  |
| UC-7 |  |  |  |  |  | + | + |  |  |  |  |  |
| UC-8 |  |  |  |  |  |  |  | + |  |  |  |  |
| UC-9 |  |  |  |  |  |  |  |  | + |  |  |  |
| UC-10 |  |  |  |  |  |  |  |  | + |  |  |  |
| UC-11 |  |  |  |  |  |  |  |  |  | + |  |  |
| UC-12 |  |  |  |  |  |  |  |  |  | + |  |  |
| UC-13 |  |  |  |  |  |  |  |  |  |  | + |  |
| UC-14 |  |  |  |  |  |  |  |  |  |  |  | + |

## Архітектура

Архітектура функціональних вимог до планування завдань складається з наступних елементів:

* Надавати можливість створювати та редагувати пакетів завдань
* Надавати можливість створювати, редагувати та видаляти завдання з пакету
* Надавати можливість створювати та налагоджувати компоненти завдання, з можливістю видалення даних та самого компонента

На малюнку 4 представлено схему архітектури функцій планування завдань.

A picture containing text, screenshot, font, line

Description automatically generated

Рисунок 2.4 — Архітектура функцій планування завдань.

## Вимоги до розробки компонентного планувальника завдань

### Вимоги до коду програмного забезпечення

До розроблення програмного забезпечення накладаються вимоги з дотримання принципів SOLID, DRY з написання програмного коду, з використанням детальних інструкцій, що вимагають ці стандарти. Застосування стандартних підходів допоможуть підтримувати та розвивати систему в наступних версіях:

* Складні програмні засоби, застосунок мобільного пристрою, програмне забезпечення бота інформаційних повідомлень повинні застосовувати стандартну архітектуру, набір правил яких буде мати змогу розвивати або доробляти проект надалі. При невдалому проектуванні частини проекту, виявити проблему і знайти можливість переробити програмний код на частину архітектури. Документувати зміни, коментувати в коді та позначати в описі нових версій, наприклад: <нова архітектура>.х.х.х (1.0.0; 2.2.3)
* Не пов’язувати частини коду (або підсистеми) системи між собою прямим зв’зком. Частини коду або підсистеми повинні бути автономними для легкого оновлення або їх зміни. Код підсистем повинен мати можливість легкого наслідування.
* Кожна зміна в архітектурі та коді системи повинна відповідати стандартним підходам та принципам вказаним вище.
* Накладаються вимоги створення системи з модульною архітектурою, що надає можливості легкого видалення або заміни окремих частин, без переробки системи в цілому.

### Системні вимоги до програмного забезпечення.

Для повноцінного функціювання системи потрібно розробити наступні підсистем:

* Підсистема інформаційних повідомлень. Повинна відповідати наступним вимогам: надсилати повідомлення, збирати і відправляти інформацію разом із завданням.
* Підсистема створення пакетів завдань. Повинна надавати в інструменти для створення гнучких завдань за допомогою компонентної системи. Надавати можливість редагувати компоненти, зберігати і видаляти їх.
* Підсистема відображення пакетів завдань магазину. Повинна надавати можливість завантажувати пакети на сервер або з сервера.
* Гнучка розширювана система компонентів. Повинна надавати можливість додавати нові компоненти, розробити типи компонентів та їх обробники.
* Розробити сховище (або базу даних) пакетів завдань користувачів та авторизацію користувачів.

Системними вимогами до програмного забезпечення, щодо запуску на наступних системах або платформах:

* Сучасна програмна платформа для мобільних операційних систем (Swift, Kotlin).
* Міжплатформена мова для створення програм (Swift).
* Потужна та надійна база даних, яка має давній досвід використання (MySQL).
* Серверна платформа для створення додатку до сервера бази даних, який має давній досвід використання (Linux/Unix).
* Серверна мова додатку до сервера бази даних, яка має давній досвід використання (PHP).