**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ  
імені ІГОРЯ СІКОРСЬКОГО»**

**Факультет інформатики та обчислювальної техніки**

**Кафедра інформаційних систем та технологій**

До захисту допущено:

Завідувач кафедри

\_\_\_\_\_\_\_\_\_\_ Олександр РОЛІК

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2023 р.

**Дипломний проєкт**

**на здобуття ступеня бакалавра**

**за освітньо-професійною програмою «Інформаційне забезпечення робототехнічних систем»**

**спеціальності 126 «Інформаційні системи та технології»**

**на тему: «Інтерактивно-компонентний планувальник організації часу людини»**

Виконав:

студент IV курсу, групи ІК-91

Остапченко Дмитро Олексійович \_\_\_\_\_\_\_\_\_\_

Керівник:

Доцент кафедри ІСТ, канд.техн.наук, доцент

Солдатова М. О. \_\_\_\_\_\_\_\_\_\_

Рецензент:

Зав. Кафедри ІПІ, доктор техн.наук, доцент

Жаріков Е. В. \_\_\_\_\_\_\_\_\_\_

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_

Київ – 2023 року

**Національний технічний університет України**

**«Київський політехнічний інститут імені Ігоря Сікорського»**

**Факультет інформатики та обчислювальної техніки**

**Кафедра інформаційних систем та технологій**

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-професійна програма «Інформаційне забезпечення робототехнічних систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

\_\_\_\_\_\_\_ Олександр РОЛІК

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2023 р.

**ЗАВДАННЯ**

**на дипломний проєкт студенту**

**Остапченку Дмитру Олексійовичу**

1. Тема проєкту «Інтерактивно-компонентний планувальник організації часу людини», керівник проєкту Солдатова Марія Олександрівна, кандидат технічних наук, доцент, затверджені наказом по університету від «11» травня 2021 р. № 1139-с

2. Термін подання студентом проєкту: 12.06.2023р

3. Вихідні дані до проєкту: потік завдань виконуваних колективом людей, вимоги та терміни виконання завдань, технічні засоби планування організації часу, інструментальні програмні засоби розробки на платформах iOS, Telegram Bot, MySQL

4. Зміст пояснювальної записки: назва розділ 1, розділ 2, розділ 3, розділ 4

5. Перелік графічного матеріалу (із зазначенням обов’язкових креслеників, плакатів, презентацій тощо)

6. Консультанти розділів проєкту\*

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Прізвище, ініціали та посада  консультанта | Підпис, дата | |
| завдання  видав | завдання прийняв |
|  |  |  |  |

7. Дата видачі завдання

Календарний план

|  |  |  |  |
| --- | --- | --- | --- |
| № з/п | Назва етапів виконання  дипломного проєкту | Термін виконання  етапів проєкту | Примітка |
| 1 | Аналіз проблем створення програмних систем організації часу людини та існуючих рішень | 24.04.2023р |  |
| 2 | Формулювання завдання на проектування та вимог до системи | 01.05.2023р |  |
| 3 | Розробка архітектури та алгоритмів роботи системи | 5.05.2023р |  |
| 4 | Аналіз та вибір інструментальних засобів проектування | 15.05.2023р |  |
| 5 | Розробка програмних модулів системи | 29.05.2023р |  |
| 6 | Тестування програмних модулів системи | 05.06.2023р |  |
| 7 | Розробка технічною документації | 12.06.2023р |  |

Студент Дмитро ОСТАПЧЕНКО

Керівник Марія СОЛДАТОВА

**АНОТАЦІЯ**

Остапченко Д.О. Інтерактивно-компонентний планувальник організації часу людини. КПІ ім. Ігоря Сікорського, Київ, 2023.

Проєкт містить 63 с. тексту, 7 рисунків, 3 таблиці, посилання на 22 літературні джерела, додатки та 5 конструкторських документів.

Ключові слова: організація часу, планувальник завдань, інформаційне нагадування, пакет програм, застосунок обміну повідомленнями, база даних

Об’єктом розробки є система планування та організації часу.

Мета розробки – підвищення якості і оперативності організації взаємодії колективу людей в інтерактивному середовищі за рахунок автоматизації процесів організації часу і створення програмного інтерактивно-компонентного планувальника завдань.

У дипломному проєкті розроблено комплекс програмних продуктів, які пов’язані між собою і утворюють повну систему, а саме: iOS застосунок, що дозволяє отримувати пакети завдань користувачів та створювати власні; Telegram Bot, що призначений для надсилання повідомлень та інформацію про завдання користувачу; компонентну систему, що дає змогу гнучку конструювати завдання будь-якої, складності може бути розширена і впроваджена в інші програмні засоби; сервіс обслуговування з використанням бази даних, що реалізує авторизацію користувача та зберігання даних пакетів завдань на сервері.

Отримані результати розробки програмного комплексу можуть використовуватись при плануванні робочих завдань як для окремої людини (користувача) так і для груп (колективів) робітників.

**SUMMARY**

Ostapchenko D.O. Interactive component planner for organizing a person's time. KPI named after Igor Sikorskyi, Kyiv, 2023.

The project contains 63 pages. text, 7 figures, 3 tables, references to 22 literary sources, appendices and 5 design documents.

Keywords: time management, task scheduler, information reminder, software package, messaging application, database

The object of development is a system of planning and organizing time.

The purpose of the development is to improve the quality and efficiency of organizing the interaction of a group of people in an interactive environment due to the automation of time management processes and the creation of a software interactive and component task planner.

The diploma project developed a set of software products that are interconnected and form a complete system, namely: an iOS application that allows you to receive packages of user tasks and create your own; Telegram Bot, designed to send messages and task information to the user; the component system, which enables flexible construction of tasks of any complexity, can be expanded and implemented in other software tools; maintenance service using a database that implements user authorization and data storage of task packages on the server.

The obtained results of the development of the software complex can be used in the planning of work tasks both for an individual person (user) and for groups (teams) of workers.

Зм.

Аркуш

№ докум.

Підпис

Дата

Аркуш

1

###### І К91.010БАК.003 ТП

Розроб.

Іванченко І.І.

Керівн.

Петренко П.П.

Затв.

###### Назва проєкту.

Відомість проєкту

Літ.

Аркушів

1

КПІ ім. Ігоря Сікорського

Група ІК-91

Номер рядка

Формат

Позначення

Найменування

Кільк. аркушів

Номер екзем.

Примітка

1

Документація загальна

2

3

Знову розроблена

4

5

А4

ІК91.010БАК.003 ПЗ

Пояснювальна записка

65

6

А3

ІК91.010БАК.003 Э1

Назва проєкту. Схема

1

7

структурна

8

А3

ІК91.010БАК.003 Э2

Назва проєкту. Схема

9

функціональна

1

10

Назва проєкту. Діаграма

11

використання

12

Назва проєкту. Діаграма

13

А3

ІК91.010БАК.003 Д1

діяльності

1

14

15

16

17

А3

ІК91.010БАК.003 Д2

1

18

19

20

21

22

23

24

25

26

27

28

т

**Пояснювальна записка**

**до дипломного проєкту**

**на тему: «** **Інтерактивно-компонентний планувальник організації часу людини»**

Київ – 2023 року

ЗМІСТ

[ВСТУП 11](#_Toc137417447)

[1 АНАЛІЗ ПРОБЛЕМ СТВОРЕННЯ ПРОГРАМНИХ СИСТЕМ ОРГАНІЗАЦІЇ ЛЮДИНИ 13](#_Toc137417448)

[1.1 Огляд предметної області автоматизації процесів організації часу людини 13](#_Toc137417449)

[1.1.1 Сфера 13](#_Toc137417450)

[1.1.2 Ідеологія проекта 14](#_Toc137417451)

[1.1.3 Які проблеми вирішує проект 14](#_Toc137417452)

[1.2 Аналіз існуючих рішень з організації програмних систем 15](#_Toc137417453)

[1.2.1 Огляд Structured 16](#_Toc137417454)

[1.2.2 Огляд Routinery 18](#_Toc137417455)

[1.3 Проблеми створення і організації програмних систем 20](#_Toc137417456)

[1.4 Постановка задачі створення інтерактивного-компонентного планувальника завдань 21](#_Toc137417457)

[Висновок до розділу 21](#_Toc137417458)

[2 АРХІТЕКТУРА СИСТЕМИ ІНТЕРАКТИВНОГО-КОМПОНЕНТНОГО ПЛАНУВАННЯ ЗАВДАНЬ 22](#_Toc137417459)

[2.1 Формалізоване подання процесу планування завдань 22](#_Toc137417460)

[2.1.1 Система компонентів 22](#_Toc137417461)

[2.1.2 Алгоритм роботи з пакетами завдань 22](#_Toc137417462)

[2.1.3 Реалізації компонентів 23](#_Toc137417463)

[2.1.4 Обґрунтування методу розв’язування 24](#_Toc137417464)

[2.2 Архітектура 24](#_Toc137417465)

[2.3 Постановка задачі і функцій при створенні інтерактивного-компонентного планувальника завдань 25](#_Toc137417466)

[2.3.1 Аналіз вимог 25](#_Toc137417467)

[2.3.2 Функціональні вимоги 30](#_Toc137417468)

[2.4 Архітектура 32](#_Toc137417469)

[2.5 Вимоги до розробки компонентного планувальника завдань 32](#_Toc137417470)

[2.5.1 Вимоги до коду 32](#_Toc137417471)

[2.5.2 Системні вимоги 33](#_Toc137417472)

[3 РЕАЛІЗАЦІЯ ІНТЕРАКТИВНОГО-КОМПОНЕНТНОГО ПЛАНУВАННЯ ЗАВДАНЬ 34](#_Toc137417473)

[3.1 Засоби розробки 34](#_Toc137417474)

[3.1.1 Технології iOS застосуноку 35](#_Toc137417475)

[3.1.2 Система компонентів 41](#_Toc137417476)

[3.1.3 Серверна програма і БД 42](#_Toc137417477)

[3.1.4 Система повідомлень 43](#_Toc137417478)

[*3.2* *Реалізація iOS застосунку* 46](#_Toc137417479)

[3.2.1 Організація окремого модуля 46](#_Toc137417480)

[3.2.2 Шари програми 47](#_Toc137417481)

[**3.2.3** Навігація 48](#_Toc137417482)

[3.2.4 Впровадження залежностей у модулі 50](#_Toc137417483)

[3.2.5 Інші частини системи 51](#_Toc137417484)

[*3.3* Аналіз допоміжних програмних засобів та засобів розробки 52](#_Toc137417485)

[Висновок до розділу 52](#_Toc137417486)

[4 ВПРОВАДЖЕННЯ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ІНТЕРАКТИВНО-КОМПОНЕНТНОГО ПЛАНУВАЛЬНИКА ЗАВДАНЬ 53](#_Toc137417487)

[4.1 Тестування 53](#_Toc137417488)

[4.2 Розгортання програмного забезпечення 59](#_Toc137417489)

[4.2.1 iOS застосунок 59](#_Toc137417490)

[4.2.2 Telegram Bot 61](#_Toc137417491)

[4.2.3 Back-end програма 62](#_Toc137417492)

[4.3 Підтримка програмного забезпечення 63](#_Toc137417493)

[4.3.1 iOS застосунок 63](#_Toc137417494)

[4.3.2 Telegram Bot 64](#_Toc137417495)

[4.3.3 Back-end програма 64](#_Toc137417496)

[4.4 Висновок до розділу 64](#_Toc137417497)

[ВИСНОВОК 65](#_Toc137417498)

[ПЕРЕЛІК ПОСИЛАНЬ 66](#_Toc137417499)

# ****ВСТУП****

**Актуальність проблеми**. В сучасному світі, де ритм життя набуває все більшої швидкості, організація часу стає критично важливим фактором для досягнення успіху та збереження рівноваги між роботою, особистими справами та відпочинком. Ефективне управління часом дозволяє максимально використовувати потенціал і досягати поставлених цілей, як для окремої людини, так і для робочих груп (або колективів).

Кожен день людина має два типи завдань для виконання – які надходять протягом доби та ті, які заплановані або повторюються. Якщо перші вирішуються по мірі надходження, то другий тип завдань потрібно виконати за певний час.

Одна з великих проблем людини – це обмежені можливості при утриманні актуального стану (інформації) для кількох складних завдань та часу їх виконання в пам’яті. Протягом робочого дня утримати всю необхідну інформацію про завдання в пам’яті, коли виконати та завершити виконання - достатньо складно, як для окремої людини, так і для колективу робітників (груп розробників), тощо.

Обмеженість пам'яті створює перешкоди в ефективному управлінні робітниками та утриманні необхідних завдань й інформації про завдання у свідомості окремого виконавця (або персонально для людини). Робітники (як окрема людина) та колективи стикаються з ситуаціями, коли частина завдань не виконується або виконується частково та не вчасно. Не ефективне зосередження на другорядних задачах, використання часу більше на окремі завдання призводить до неефективної роботи та невиконання окремих завдань, як частково, так і повністю.

**Об’єктом дослідження** є процеси планування і організації часу людини або колективу при виконанні комплексу взаємопов’язаних завдань або задач проектів. Неправильне розподілення задач робітника або колективу негативно впливає на ~~нашу~~ продуктивність праці та результати роботи в цілому.

**Предмет дослідження**. Для вирішення проблем з розподіленням робочого часу групи (колективу) або часу окремої людини (робітника), створюється проект з організації часу, який спрямований на надання необхідних інструментів та підходів для ефективного управління завданнями, які можуть повторюватись та виконуватись в певний час або з певним проміжком часу (за розкладом).

Важливо підкреслити, розробка проекту не має на меті надати універсальний підхід до організації часу, але ставить за завданням створити інструменти, які дозволять кожному користувачу (або колективам) індивідуалізувати свій підхід до виконання задач залежно від особистих потреб, пріоритетів та стилю роботи (або життя).

**Мета і завдання дослідження**. Підставою для створення проекту стала актуальна проблема відсутності гнучких інструментів організації часу людини на ринку програмних засобів. Головною метою проекту є створення простого та гнучкого програмного засобу для організації роботи (та відпочинку) так, як потребує людина (або управлінець колективом), накопичувати персональний досвід та мати різні можливості як передавати інформацію іншим - в якості завдань або типовим розпорядком дня (шаблоном).

Для досягнення цієї мети, в проекті мають бути виконані наступні завдання:

1. Проведення аналізу предметної області процесів організації часу людини і створення спеціалізованих програмних систем з визначенням проблем у їх застосуванні і напрямків розвитку.
2. Створення концепції інтерактивно-компонентної системи з інтуїтивно зрозумілим інтерфейсом користувача, який дозволить легко планувати свій робочий час та розподіляти завдання.
3. Розробка архітектури та створення функціоналу програмного середовища інтерактивно-компонентної системи, який дозволить користувачам зберігати свій досвід (розпорядок дня) та накопичувати влучні поради, завдання та рекомендації в особистому профілі.
4. Розробка механізму зворотного зв'язку та підтримки, щоб користувачі (або групи) могли обмінюватись інформацією, результатами роботи та досягнутими успіхами.
5. Проведення тестування і експериментального випробування функціоналу програмного середовища інтерактивно-компонентної системи для визначення ефективності використання часу та успішності виконання завдань користувача.

**Практичне значення одержаних результатів**. Проект призначений для вирішення проблеми неефективного використання часу і надання користувачам інструмента, який допоможе раціонально розподіляти час на виконання завдань, отримувати нагадування, повідомлення та іншу необхідну інформацію про роботу або відпочинок. Обмін інформації між користувачами, як керівниками робочих груп або окремими робітниками має призвести до більш швидкого та якісного виконання поставлених задач та розподілення часу між роботою та відпочинком.

Дипломний проєкт складається з наступних розділів: вступ, основні розділи, висновки, список використаних джерел із ... найменувань, ... додатків. Графічна частина включає … креслеників формату … . Загальний обсяг … сторінок

# АНАЛІЗ ПРОБЛЕМ СТВОРЕННЯ ПРОГРАМНИХ СИСТЕМ ОРГАНІЗАЦІЇ ЧАСУ ЛЮДИНИ

## Огляд предметної області автоматизації процесів організації часу людини

Висока інтенсивність процесів організації і проведення різних видів діяльності людини або в виробничій або в побутовій сфері вимагає контролю за відтворенням точної послідовності подій та обробки значного обсягу інформації. Саме тому знаходження ефективних стратегій для керування і обробки інформації стає насущною задачею в області автоматизації процесів організації часу людини.

На даний час існує багато напрямків та програмних систем з організації і планування часу людини, починаючи від простих “списків до виконання" (To-do lists) й закінчуючи складними планувальними системами, такими як Microsoft Project. Основна різниця між ними полягає у масштабі та цілях, які системи намагаються досягти, орієнтуючись на різні фокусні групи людей або колективи. Отже, розвивається дві моделі створення проектів з організації часу людини: спрямовані на бізнес - модель «бізнес-до-бізнесу» (БТБ) і на кінцевого користувача - модель «бізнес-до-споживача» (БТС), тобто "від продукту до клієнта" (Project to Client)).

Проекти, спрямовані на бізнес і організацію бізнес-процесів, створюють свої продукти у вигляді кількох застосунків чи веб-сервісів з навантаженим інтерфейсом користувача та великою кількістю функцій для планування, аналізу ефективності, тощо. Існують окремі випадки програмного забезпечення, з універсальною вимогою для настільних або веб-планувальників з моделлю БТБ (бізнес-до-бізнесу) з великою кількістю функцій, які теж важко використовувати на малих екранах.

Проекти, спрямовані на користувачів, що мають на меті організацію часу для окремої особи (робітника), мають інші підходи. Інтерфейс повинен бути мінімізованим і зрозумілим, оскільки для моделі БТС (бізнес-до-споживача) кожен клієнт є унікальним. Такі системи використовуються "на ходу", тому мобільний телефон чи смарт-годинник мають велике значення для їхнього успішного функціонування.

Прикладами означених проектів є Atlassian Jira та Microsoft Project спрямовані на організацію бізнес-процесів у компаніях, в той час як Microsoft Anydo та Structured націлені на сегмент "від продукту до клієнта" (Product to Customer) - продаж програмних засобів для клієнтів, які зацікавлені у кращому управлінні своїм часом.

В будь якому випадку, таке організаційне програмне забезпечення виконує алгоритмічну роботу з даними та створює умови для ефективного сприйняття завдань та управління часом. Незалежно від цілей і масштабів, системи орієнтуються на найменший елемент обробки подій - завдання.

## Аналіз існуючих рішень з організації програмних систем

1.2.1 Аналіз підходів до організації програмних систем планування і організації часу людини

На ринку офісних застосувань і пакетів програм (в офіційних магазинах розробників) існує велика кількість програмного забезпечення для організації часу. Це означає, що конкуренція серед розробників є дуже високою, і для того, щоб отримати перевагу у користувачів, необхідно створити та запропонувати нові ефективні методи взаємодії та інтерфейси для користувачів.

Деякі засоби програмного забезпечення пропонують інтуїтивно зрозумілий та простий інтерфейс користувача, що дозволяє швидко оволодіти програмою і почати організовувати свій час без складнощів. Інші пропонують інноваційні методи планування, такі як графіки Ганта або “канбан-дошки", які дають змогу візуально відстежувати прогрес виконання та керувати завданнями.

Деякі програми акцентуються на синхронізації та доступності даних. та пропонують синхронізацію розкладів та завдань між різними пристроями та платформами, такими як комп'ютери, смартфони та планшети, щоб мати постійний доступ до оновленої інформації.

Деякі інші програми спеціалізуються на окремих групах користувачів або на виконання специфічних завдань. Наприклад, є програмні засоби, призначені для менеджменту проектів, нагадувань про медичні препарати, отримання часу роботи та багато інших.

Окрім цього, оновлення та підтримка такого програмного забезпечення є одним із важливих аспектів успішності. Розробники надають регулярні оновлення з новими функціями, виправленням помилок та забезпечують гарантовану підтримку користувачів, щоб забезпечити якість та сучасність програмного продукту.

В цілому на даний час утворилося два найпопулярніших підходи (пакети) створення додатків з організації часу на ринку програмного забезпечення, порівняльне оцінювання яких доцільно провести на підставі аналізу таких основних характеристик, як підтримувані пристрої, цілі та ідеології побудови, простоту керування процесами.

1.2.2. Огляд та оцінювання пакету Structured

Пакет Structured - це програмне забезпечення щоденного візуального планувальника, який поєднує календар і список справ для надання можливість планування завдань, відстеження прогресу їх виконання протягом дня та позначення їх як виконані, коли завдання завершені.

Сервіс пакету доступний лише на пристроях від компанії Apple. Може використовуватись на iPhone, iPad та на пристроях, що працюють під управлінням операційної системи macOS. Це означає, що користувачі, які володіють пристроями Apple, можуть використовувати функціональність, яку надає пакет Structured для планування та відстеження своїх завдань та розкладу.

Пакет Structured - це застосунок, який пропонує ряд корисних функцій для організації часу та завдань. Основний функціонал додатку Structured включає:

1. Планування завдань: Застосунок дозволяє вам створювати та планувати завдання на календарі. Надається можливість встановлювати терміни виконання, пріоритети та тривалість кожного завдання.
2. Список справ: Structured надає можливість створювати список справ, який включає окремі завдання або другорядні завдання. Ви можете легко організовувати свої завдання за категоріями або проектами.
3. Візуальне планування: За допомогою Structured є можливість використовувати візуальний планувальник для відстеження розкладу та завдань. Завдання представляються на календарі з можливістю переміщати за допомогою перетягування, змінюючи дату та час.
4. Нагадування та сповіщення: Structured дозволяє встановлювати нагадування про наближення термінів виконання завдань або про важливі події. Отримування сповіщення про важливе завдання.
5. Відстеження прогресу: Застосунок надає можливість відстежувати прогрес виконання завдань. Надається можливість позначати завдання як виконані, встановлювати статуси або використовувати вбудовані мітки для класифікації завдань.
6. Синхронізація даних: Structured забезпечує синхронізацію даних між пристроями, що дозволяє отримувати доступ до завдань та розкладу з різних пристроїв, таких як смартфони або планшети.
7. Функції програмного забезпечення дозволяють більш ефективно організувати час, планувати завдання та відстежувати виконання, забезпечуючи більшу продуктивність та організованість роботи.

Сервіс пакету Structured, на жаль, не має базового функціоналу для організації складних (наприклад, повторюваних) завдань людини, працююча версія програмного забезпечення призначена для планування виконання простих задач. Бізнес схема програми та сервісу в цілому, не може використовуватись для організації робочих процесів для груп користувачів або робочих колективів.

Сервіс програми зосереджений переважно на плануванні завдань та відстеженні прогресу. Це може бути обмеженням для тих користувачів, хто шукає інструмент, який допоможе організувати та регулярно повторювати рутинні дії і ділові звички.

1.2.3. Огляд та оцінювання пакету Routinery

Пакет Routinery - це додаток для організації завдань, який надає широкий функціонал для планування, відстеження та керування складними задачами. Основні функції Routinery включають:

1. Створення задачі: користувач створює власні задачі залежно від потреб і пріоритетів. Наприклад, задачі для ранкових нарад, щоденного тренування, тестування, тощо.
2. Гнучке планування: програмне забезпечення Routinery дозволяє гнучко планувати свої задачі на різні дні тижня або на конкретні дати. Надається можливість встановити повторюваність задачі щоденно, щотижня або вибрати приватний графік.
3. Нагадування і сповіщення: додаток Routinery надсилає нагадування та сповіщення, щоб нагадати користувачеві про важливі завдання. Користувач отримує повідомлення на смартфоні або інших пристроях.
4. Відстеження прогресу: програмне забезпечення Routinery дозволяє відстежувати прогрес у виконанні завдань. Надається можливість позначати завдання як виконані або відмічати їх у разі затримки. Функціонал надає можливість спостерігати за продуктивністю та дотримання графіку виконання задач.
5. Спільний доступ: додаток Routinery дозволяє спільний доступ до задач між користувачами. Надається можливість створювати групи завдань та додавати учасників, що спільно виконують їх. Функціонал надає можливість обмежено використовуватись як програмне забезпечення для робочих груп (колективів).
6. Статистика і звіти: програмне забезпечення Routinery надає статистику та звіти щодо задачі. Надається можливість: переглядати час (проведений на кожному завданні), аналіз продуктивності та отримувати дані виконання задачі.
7. Інтеграція: додаток Routinery надає можливість налаштовувати і персоналізувати завдання згідно з вимогами користувача. Надається можливість додавати зображення, опис, кольори та інші елементи, що допомагають налагодити пакет завдання. Routinery має можливість інтегруватись з іншими додатками або сервісами, такими як календарі та інші (для ефективності і зручності).

Routinery має розширений функціонал, але незадовільний на фундаментальному рівні - складний інтерфейс, відсутність функціоналу обміну завдань, відсутність компонентної системи і інтерактивності, відсутність спільного користування завданнями.

Незважаючи на окремі функціональні можливості, програмне забезпечення Routinery складне в використанні, зі складним дизайном та набором компонентів, які не використовуються при роботі з програмою.

## Проблеми створення і організації програмних систем

Створення і організація програмних систем планування і організації часу людини має проблеми зі складною структурою компонентної системи, залученням новітніх підходів та засобів розробки, в тому числі нових мов програмування.

Переліком основних проблем, з якими доводиться зіткнутися розробниками програмних систем є:

1. Визначення необхідних вимог. Необхідне визначення точних вимог до системи, які можуть змінюватися під час процесу розробки або бути неповними. Неправильне визначення вимог може призвести до некоректної роботи програмної системи.
2. Управління змінами до вимог. Програмні системи потребують постійних змін і оновлень в роботі. Некоректні або недостатні зміни можуть призвести до невірною роботи системи або втрати даних.
3. Розробка та тестування. Розробка програмних систем вимагає достатнього часу, ресурсів та ефективного тестування, щоб виявити помилки та дефекти програми. Недостатнє тестування може призвести до появи помилок у системі.
4. Управління проектом. Управління проектом є ключовим фактором для розробки програмних систем. Недостатнє планування або погане керування призводять до затримок у розробці системи, перевищення бюджету і невиконання необхідних вимог.
5. Безпека. Програмні системи вразливі до різного роду загрозам безпеці: “хакерські” атаки, витоки даних та інше. Забезпечення безпеки програмної системи є важливою вимогою, яке потребує змін та тестування на кожному етапі розробки.
6. Сумісність та інтеграція. Необхідно забезпечити сумісність та інтеграцію програмної системи з вже наявними програмними комплексами та компонентами. Несумісність приводить до проблем з обміном даних та невірною роботою системи.

Отже, можна сформулювати наступні шляхи розв’язання проблем створення програмних систем організації часу людини для підвищення їх ефективного використання і рівня зацікавленості користувачів:

1)удосконалення підходів щодо опису і поданні організаційних завдань за рахунок створення компонентного конструктора для користувачів.

2)удосконалення методів покращення якості виконання завдань за рахунок створення інструментів з поширення напрацювань за окремими видами завдань.

3)удосконалення управлінням робочими групами та окремими робітниками за рахунок створення системи заохочення найкращих рішень при виконанні завдань.

Аналіз показує, що доцільно звернути увагу на подальший розвиток і вдосконалення напряму пакету Structured і розвивати його функціональні можливості за наступними задачами у новому проекті системи організації часу людини:

1. Пакети завдань: проект повинен дозволяти організовувати завдання в пакети або групи, що допомагає структурувати роботу та керувати комплексними проектами.
2. Групи користувачів: проект повинен дозволяти додавати пакети до груп користувачів, що необхідно для спільної роботи над проектами або для делегування завдань між учасниками групи.
3. Відстеження прогресу: проект повинен надавати не тільки можливість відстежувати прогрес виконання завдань, але й інструмент для аналітичної і інтерактивної роботи з даними. Утворити систему інтерактивних компонентів, що надасть можливість нескінченно розширювати можливості для користувачів взаємодіяти з інформацією (завданнями).
4. Компонентна система: проект повинен надати можливість гнучкого середовища для настройки пакетів завдань, оновлення поточного набору компонентів надають набір інструментів для встановлення і налагодження завдань. Проект системи повинен представляти завдання як налаштовані компоненти, що налагоджуються за допомогою простого інтерфейсу користувача.
5. Робота зі списками даних: проект повинен надати можливість працювати зі списками даних та алгоритмічно їх прикріпляти до завдань. Цей функціонал корисний для обробки та аналізу інформації, пов'язаної з конкретним завданням.
6. Запис даних до завдань: проект повинен надати можливість користувачеві зберігати дані до завдань, включати додаткову інформацію, коментарі, примітки або відповіді, які можуть бути оброблені компонентами, пов'язаними з завданням.
7. Інтерфейс користувача: проекті повинен надати користувачеві можливість легко виконувати функції та операції з завданнями та даними.

## Постановка задачі створення інтерактивного-компонентного планувальника завдань

Основна задача проектування інтерактивного-компонентного планувальника завдань полягає в розробці гнучкого інструменту для планування завдань, які повторюються, оптимізації рутинних завдань, інтеграції з групами робітників (колективів) або виконавців. Проект повинен розроблятися за моделлю "від продукту до клієнта" (Project to Client) і бути спрямований на покупців (користувачів).

Основними вимогами до функціоналу інтерактивного-компонентного планувальника завдань є:

* Ефективне розподілення часу людини або робітника при плануванні робочого дня або при плануванні денних подій (які мають вимогу повторення)
* Отримання своєчасних інформаційних повідомлень про розподілення денного часу поточних подій
* Створення нових персональних пакетів завдань. Завантаження пакетів завдань до системи для обміну між всіма користувачами.
* Створення умов для використання пакетів завдань групами користувачів або груп за спільними завданнями.
* Інтуїтивно зрозумілий, гнучкий та простий інтерфейс користувача при роботі з додатками системи

Проект має розроблятися на системах гнучких повідомлень і обробки інформації у платформі iOS, Telegram та враховувати наступні вимоги:

* Застосунок використовує інтерфейс згідно з Apple Guidelines. Кожен елемент повинен мати функціональну роль, дизайн система і ідеологія візуальних компонент Apple повинна використовуватись застосунку. Інтерфейс користувача повинен бути простим та легким в використанні. Інтерфейс користувача адаптований під всі типи екранів смартфонів Apple (iPhone).
* Застосунок повинен заощадливо використовувати ресурси пристрою: швидка та безпечна робота з оперативною пам'яттю, оптимальна робота з потоками системи.
* Інтерфейс взаємодії з ботом повідомлень надається через спеціальні команди.

## Висновок до розділу

Проаналізовано предметну область з розробки програмної системи для автоматизації часу людини, визначені основні проблеми при створені програмних систем, та виконана постановка задачі на створення інтерактивного-компонентного планувальника завдань. За умови ефективного вирішення проблем зі створення програмної системи та за умови виконання поставлених задач, очікується програмне рішення, яке надасть користувачам зручний інструмент для планування денних завдань та подій. Користувачі об'єднані в робочі групи та групи за спільними завданнями отримають гнучке програмне рішення для поліпшення умов праці та результатів роботи.

# Архітектура інтерактивно-компонентного планувальника завдань

## Формалізоване подання процесу планування завдань

### Структура процесу планування завдань

Основним об'єктом у системі організації і плануванні часу виступає завдання користувача. Тому процес планування завдань повинен складатися з наступних елементів:

- магазин пакетів, які формує користувач;

- пакет завдань;

- завдання користувача, які утворюють пакети (які розміщуються в магазині пакетів);

- компоненти завдання, які формуються для кожного завдання, та описують правила обробки і набір інших параметрів (Додаток 1).

На Рисунку 2.1 представлено узагальнену схему підпорядкованості елементів процесу планування завдань.



Рисунок 2.1 — Узагальнена схема підпорядкованості елементів процессу планування завдань

Магазин надає доступ до пакетів. Пакет включає в себе задачі. Задачі мають в собі налаштовані компоненти, аналізуючи котрі досягаються всі вимоги до системи.

### Подання елементів структури процесу планування завдань

Подання елементів структури процесу планування завдань складається з визначення і опису типів компонентів та способів їх обробки. В якості типів компонентів встановлено: магазин, пакет, завдання, компонент. В якості способів обробки розглядаються алгоритми, які подаються як методи обробки. Також в опис елементів включені ознаки компонентів.

1. Подання типів компонентів та їх способів обробки. Необхідне створення n типів компонентів + n1 типів обробників + n2 реалізацій типів обробників. Де n = n1 = n2.

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

2. Подання пакетів, магазинів завдань та алгоритмів їх обробки. Кожне завдання в пакеті task[n] повинно складатися з 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.

Подання пакету завдань вимагає створення та зберігання n пакетів для m користувачів. Кожен пакет може мати k завдань.

Подання магазину вимагає можливості зберігання n пакетів. Користувач має мати можливість завантажувати пакети до сховища даних – n.push (newPackage), та видаляти n.deleteAt (packageIndex) пакети зі сховища.

Повинно надаватися можливість завантажувати для кожного окремого користувача n[userIndex], k кількість пакетів.

3. Подання алгоритмів обробки. До складу алгоритмів включені наступні методи обробки:

1) Appear – слід/не слід показати завдання. 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)

2) Data – обробити дані і повернути у новому виді. 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.

3) Interactive – отримами відповідь від користувача, оборобити, повернути результат. 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).

4. Атрибути опису. Крім методів обробки компоненти визначаються атрибутами опису:

1) Інтервалу подій Interval. 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.

2) призначенням Description. Для кожного компонента k створюється строка або строки з детальним описом призначення компонента.

## Концепція архітектури інтерактивно-компонентного планувальника завдань

Для повноцінного функціювання системи організації часу в складі інтерактивно-компонентного планувальника завдань виділено наступні підсистеми:

* Підсистема інформаційних повідомлень. Повинна надсилати повідомлення, збирати і відправляти інформацію разом із завданням.
* Підсистема створення пакетів завдань. Повинна надавати інструменти для створення гнучких завдань за допомогою компонентної системи з можливістю редагування, зберігання і видалення компонентів.
* Підсистема відображення пакетів завдань магазину. Повинна надавати можливість завантажувати пакети на сервер або з сервера.
* Підсистема налаштування для гнучкого розширювання системи компонентів. Повинна надавати можливість додавати нові компоненти, розробляти типи компонентів та їх обробники.
* Підсистема сховища (або базу даних) пакетів завдань користувачів та авторизації користувачів.

Планувальник завдань повинен мати наступні можливості:

* Створювати та редагувати пакети завдань
* Створювати, редагувати та видаляти завдання з пакету
* Створювати та налагоджувати компоненти завдання, з можливістю видалення даних та самого компонента

На рисунку 2.4 представлено схему взаємозв'язку функцій планувальника завдань.

A picture containing text, screenshot, diagram, circle

Description automatically generated

Рисунок 2.4 — Архітектура функцій планування завдань

## Функції інтерактивного-компонентного планувальника завдань

### Функції клієнта підсистем створення і відображення пакетів завдань

Підсистеми створення і відображення пакетів завдань реалізуються за допомогою інтерактивного клієнта планувальника завдань. В якості головних задач клієнта планувальника завдань встановлені наступні:

* створення користувачем завдання, призначення опису часу та інших параметрів роботи;
* зберігання створеного завдання на пристрої користувача та в базі даних.
* отримування повідомлення (вчасні) про стан або іншу інформацію про завдання.

Наведені задачі дозволити визначити такі основні функції програмного забезпечення клієнта планувальника завдань: 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 таблиці 2.1 … 2.10 наведено опис основних функцій програми клієнта планувальника завдань.

Таблиця 2.1 – Специфікація функцій клієнта планувальника завдань

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Назва функції і її позначення | Елемент опису | | | | | | |
| Goals | Actors | Trigger | Pre-conditions | Flow of Events | Extension | Post-Condition |
| Відкриття допомоги | Надати користувачу доступ до допомоги в застосунку | Користувач | Користувач натискає кнопку обирає з меню налаштувань Onboarding | Зайти до налаштувань | Користувач натискає кнопку налаштування і обирає з меню налаштувань Onboarding | - | Запустився Onboarding з поясненнями |
|  |  |  |  |  |  |  |  |

Таблиця 2.2 – Специфікація функцій клієнта планувальника завдань

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Назва функції і її позначення | Елемент опису | | | | | | |
| Goals | Actors | Trigger | Pre-conditions | Flow of Events | Extension | Post-Condition |
| Створити пакет | Надати можливість створювати нові «болванки» пустих пакетів | Користувач | Користувач натискає кнопку додати пакет | - | Користувач натискає мнопку налаштування, вводить назву | - | Створюеться пустий пакет |
|  |  |  |  |  |  |  |  |

Таблиця 2.3 – Специфікація функцій клієнта планувальника завдань

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

Таблиця 2.4 – Специфікація функцій клієнта планувальника завдань

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

Таблиця 2.5 – Специфікація функцій клієнта планувальника завдань

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Назва функції і її позначення | Елемент опису | | | | | | |
| Goals | Actors | Trigger | Pre-conditions | Flow of Events | Extension | Post-Condition |
| Видаляти пакети | Надати можливість видаляти пакети завданнями | Користувач | Свайп вліво, натиснути кнопку видалити | - | Свайп вліво, натиснути кнопку видалити | - | Пакет видаляється і зникає |
|  |  |  |  |  |  |  |  |

Таблиця 2.6 – Специфікація функцій клієнта планувальника завдань

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Назва функції і її позначення | Елемент опису | | | | | | |
| Goals | Actors | Trigger | Pre-conditions | Flow of Events | Extension | Post-Condition |
| Видаляти завдання | Надати можливість видаляти завдання | Користувач | Свайп вліво, натиснути кнопку видалити | Бути у пакеті | Свайп вліво, натиснути кнопку видалити | - | Завдання видаляється і зникає |
|  |  |  |  |  |  |  |  |

Таблиця 2.7 – Специфікація функцій клієнта планувальника завдань

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Назва функції і її позначення | Елемент опису | | | | | | |
| Goals | Actors | Trigger | Pre-conditions | Flow of Events | Extension | Post-Condition |
| Видаляти копомнент | Надати можливість видаляти завдання | Користувач | Свайп вліво, натиснути кнопку видалити | - | Свайп вліво, натиснути кнопку видалити | - | Завдання видаляється і зникає |
|  |  |  |  |  |  |  |  |

Таблиця 2.8 – Специфікація функцій клієнта планувальника завдань

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

Таблиця 2.9 – Специфікація функцій клієнта планувальника завдань

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

Таблиця 2.10 – Специфікація функцій клієнта планувальника завдань

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

### Функції бота підсистеми інформаційних повідомлень

Підсистема інформаційних повідомлень реалізується за допомогою автоматично діючого бота планувальника завдань. Реалізація задач роботи з інформаційними повідомленнями про заплановані завдання потребують виконання таких основних функцій програмного забезпечення відповідної підсистеми: надати або видалити пакет завдань зі списку, отримувати наявні пакети завдань, отримувати інформаційні повідомлення про завдання. Діаграма застосування 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

В таблиці 2.11...2.14 наведено опис основних функцій програми бота інформаційних повідомлень.

Таблиця 2.11 – Специфікація функцій бота інформаційних повідомлень

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Назва функції і її позначення | Елемент опису | | | | | | |
| Goals | Actors | Trigger | Pre-conditions | Flow of Events | Extension | Post-Condition |
| Додати пакет - UC11 | Надати можливість додавати пакет | Користувач | Команда add package | Бути у боті | Закинути в бота пакет, реплаем зробити команду add | - | Пакет додається до боту |
|  |  |  |  |  |  |  |  |

Таблиця 2.12 – Специфікація функцій бота інформаційних повідомлень

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Назва функції і її позначення | Елемент опису | | | | | | |
| Goals | Actors | Trigger | Pre-conditions | Flow of Events | Extension | Post-Condition |
| Видалити пакет | Надати можливість видаляти пакет | Користувач | Команда rm package\_index | Бути у боті | Команда rm package\_index | - | Пакет видаляється з боту |
|  |  |  |  |  |  |  |  |

Таблиця 2.13 – Специфікація функцій бота інформаційних повідомлень

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

Таблиця 2.14 – Специфікація функцій бота інформаційних повідомлень

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Назва функції і її позначення | Елемент опису | | | | | | |
| Goals | Actors | Trigger | Pre-conditions | Flow of Events | Extension | Post-Condition |
| Список пакетів | Надати можливість дивитися список пакетів | Користувач | Команда pls | Бути у боті | Команда pls | - | Показує все пакети в боті |
|  |  |  |  |  |  |  |  |

## Вимоги до програмного забезпечення

### Вимоги до кодування программного 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

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).

### Обґрунтування методів та засобів створення програмного забезпечення для плануванні завдань та повідомлень

Проект розробляється за моделлю "від продукту до клієнта". В якості платформ для розробки використовуються: мобільна платформу iOS (мобільний додаток, чат бот), платформа Telegram Bot (мобільний додаток Telegram) та база даних MySQL (для зберігання даних користувачів та задач). Структурну схему проекту системи наведено в Додатку 1.

Проект має open-source концепцію, як основу, з використанням новітньої мови програмування до платформ Apple - Swift для клієнтського додатку та чат боту. Для платформи бази даних та розробки серверного додатка для API (доступ до бази даних клієнтами системи) обрано MySQL та мову PHP, які мають найбільше розповсюдження. Структурно проектується конструктор, який надає можливість передавати завдання та іншу інформацію від користувача до користувача. Накопичення завдань призведе до їх оцінювання (з точки зору ефективності та використання ) і має призводити до найкращих управлінських рішень.

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.

# РЕАЛІЗАЦІЯ ІНТЕРАКТИВНОГО-КОМПОНЕНТНОГО ПЛАНУВАННЯ ЗАВДАНЬ

3.1 Аналіз засобів та технологій розробки

Вибір технологій для розробки складових елементів інтерактивно-компонентного планувальника завдань залежить від таких головних критеріїв: вимог системи і забезпечення прискореного 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.

Крім того, вид застосованих засобів і технологій визначається і залежить від об'єкту розробки – підсистем інтерактивно-компонентного планувальника завдань, а саме клієнта підсистем створення і відображення пакетів завдань, бота підсистеми інформаційних повідомлень, бази даних для сховища пакетів завдань користувачів і авторизації користувачів. Тому аналіз і вибір необхідно провести для кожного з них.

3.1.1 Вибір платформ та технологій клієнта підсистем створення і відображення пакетів завдань

Для розробки клієнта підсистем створення і відображення пакетів завдань необхідно використовувати інструменти та платформи розробки мобільного застосунку, що мають можливість проектувати інтерфейс користувача, використовувати бібліотеки і додатки, 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 дозволяє створювати додатки для iOS та Android зі спільним кодом, використовуючи мову програмування Dart, 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 цеXcode (UIKit та SwiftUI), який є головним інструментом для розробки мобільних додатків. Використовуються мови Swift і Objective-C, що надають повний доступ до функцій і можливостей, пропонованих платформою Apple. Для платформи Android цеAndroid Studio (Android SDK), як головний інструмент для розробки Android-застосунку. Використовується мова програмування Java або Kotlin. Забезпечується доступ до функцій та можливостей платформи Android.

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

Перевагами розробки застосунків на спеціалізованих платформах Xcode та Android Studio є:

* 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.

Перевагами розробки застосунків на універсальних платформах, що дозволяють вести розробку до iOS і Android одночасно є:

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

Але слід відзначити, що розробки на універсальних платформах має наступні недоліки, які не дозволяє обрати їх як засоби програмування проекту:

* менша продуктивність програмних продуктів, порівняно з додатками, розробленими на Xcode, Android Studio;
* обмеження доступа до специфічних функцій та технічних засобів на платформах iOS, Android.

Крім того, для розробки клієнта обрано мобільну платформу iOS, використовуючи платформу Xcode (технології Apple), що дозволяють застосовувати засоби швидкої розробки інтерфейсу користувача. Така розробка програмного продукту має забезпечити сумісність з функціоналом і стандартами платформи iOS.

Засобами розробки на платформі Xcode є мови програмування Swift та Objective-C. Це основні мови програмування, що використовуються для розробки застосунків на цій платформі. 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:
   1. Мова Swift активно розвивається Apple і має велику підтримку спільноти розробників. Мова отримує оновлення та нові функції, що полегшують розробку та покращують продуктивність.
   2. Objective-C підтримується Apple.

Результати порівняльного аналізу дозволяє надати перевагу для розробки клієнта інтерактивно-компонентного планувальника завдань - мові Swift.

Вибір цієї мови обумовлює подальший порівняльний аналіз і вибір фреймворку – бібліотек для розробки інтерфейсу користувача (UI) для мобільних пристроїв та компьютерів Apple, таких як iPhone, iPad та Mac. Такими фреймворка слугують SwiftUI та UIKit.

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

Фреймворк SwiftUI є новим офіційним фреймворком для розробки UI під iOS, Використовує мову програмування Swift, легкий в створені інтерфейсу користувача за допомогою декларативного підходу, має простий процес розробки 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 показує перевагу у розробці клієнту інтерактивно-компонентного планувальника завдань - мові Swift на фреймворку (бібліотеки) SwiftUI.

3.1.2 Вибір платформ та технологій чат-бота підсистеми інформаційних повідомлень

Чат бот підсистеми інформаційних повідомлень доцільно розробити у вигляді додатку Swift для організації повідомлень для користувачів системи інтерактивно-компонентного планувальника завдань, який 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.

Мовою програмування є Swift для розробки бота та імпорт бібліотек компонентів, розроблених для додатка iOS. Для створення телеграм-бота на Swift використовується Telegram Bot API **-** офіційний API, наданий Telegram, який дозволяє взаємодіяти з ботами через HTTP-запити.

3.1.3 Вибір платформи бази даних та технології API додатку сервера

Для підсистеми сховища (бази данних) пакетів завдань користувачів та для авторизації користувачів в системі використовується сервер баз даних. Для отримання даних з підсистеми використовується техонологія API в додатку сервера, що використовує мову PHP.

Платформою для бази даних використовується реляційна система керування базами даних (СКБД) MySQL за рахунок легкості розгортання, підтримці у хмарах та хостінгах. MySQL - це 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.

3.2 Реалізація додатка клієнта користувача для мобільного пристрою iOS

Реалізація додатка клієнта забезпечила наступні архітектурні рішення:

* організація окремих модулів;
* взаємодія модулів та передача даних між ними;
* навігація між модулями;
* впровадження залежності модуля.

3.2.1 Система компонентів додатка клієнта користувача

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, що входять до задач, компонент додатку.

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

3.2.2 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

3.2.3 Шари програмного забезпечення додатка клієнта

Програмне забезпечення додатка клієнта користувача до мобільно пристрою iOS розділене на шари, щоб забезпечити незалежне функціонування окремих його частин. На рисунку 3.2 представлено програмні шари, що розроблено в програмному забезпеченні додатка клієнта користувача до мобільного пристрою iOS

A picture containing text, screenshot, line, receipt

Description automatically generated

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

Програмний шар «Навігаціязастосунку» знаходиться на вищому рівні, який базується на розроблених модулях і формує дерево програмного потоку даних. 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.

Мережева взаємодія в програмному забезпеченні відокремлена в окремий програмний 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 клієнта до мобільного пристрою iOS наведено на **СХЕМІ 2**

Алгоритм роботи додатку забезпечує завантаження базового функціоналу, до якого включено:

- завантаження пакетів користувача з диску пристроя додатка з наданням можливостей додавання та корегування даних;

- завантаження меню налаштувань додатка з пунктами відомостей про додаток і допомоги.

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

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

3.3 Реалізація чат боту підсистеми інформаційних повідомлень

Реалізація чат боту підсистеми 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 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.

3.4 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, який 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 інтерактивно-компонентного планувальника завдань надають додаткам користувачів 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 інтерактивно-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 за протоколом HTTP за допомогою методів 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 та інші дії.

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 за форматом «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 в таблиці 3.1

Таблиця 3.1 – Перелік помилок в роботі додатка API

|  |  |
| --- | --- |
| Код | Текстове повідомлення помилки |
| 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 |

3.5 Допоміжні програмні засоби та засоби для розробки

Для ефективного і швидкого розроблення складових системи було використано наступні допоміжні засоби:

1) 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.

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

3) Git - система 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.

В результаті при розробці інформаційної системи інтерактивно-компонентного планувальника завдань було використано 3 мови програмування, 4 фреймворка (бібліотеки), 2 платформи розробки, 1 реляційна база даних та інше. Загальний перелік обраних засобів та платформ для розробки проекту системи наведено у таблиці 3.2.

Таблиця 3.2 – Перелік застосованих засобів і платформ для розробки

|  |  |
| --- | --- |
| 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 |

## 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 та 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.

# ВПРОВАДЖЕННЯ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ІНТЕРАКТИВНО-КОМПОНЕНТНОГО ПЛАНУВАЛЬНИКА ЗАВДАНЬ

## Тестування

Мета тестування якості програмного забезпечення

Виявлення помилок: Головною метою тестування є виявлення помилок, дефектів або некоректного функціонування програмного забезпечення. Це допомагає забезпечити належну якість та надійність продукту перед його випуском на ринок.

Перевірка відповідності вимогам: Тестування допомагає перевірити, чи відповідає програмне забезпечення вимогам і специфікаціям, встановленим для нього. Це дозволяє забезпечити, що програма виконує заплановані функції та задовольняє потребам користувачів.

Підтримка надійності: Тестування допомагає переконатися, що програмне забезпечення працює безперебійно та надійно в різних умовах та сценаріях. Це включає перевірку стабільності, витривалості та відмовостійкості системи.

Підвищення ефективності: Тестування може допомогти виявити та усунути недоліки в роботі програмного забезпечення, що можуть призводити до його низької продуктивності або швидкодії. Це дозволяє поліпшити ефективність роботи системи та забезпечити задоволення користувачів.

Забезпечення високої якості: Тестування сприяє підвищенню загальної якості програмного забезпечення шляхом ідентифікації та усунення проблем. Це включає перевірку функціональності, правильності та зручності використання програми.

Для перевірки якості програмонго забеспечення буде використовуватися Функціональне тестування, Unit tests

Функціональне тестування - це процес перевірки програмного забезпечення з метою визначення, чи працює воно відповідно до очікувань і вимог щодо функціональності. Цей тип тестування спрямований на перевірку основних функцій програми, таких як введення, обробка даних, виведення результатів та взаємодія з користувачем.

Unit tests (одиничні тести) - це процес тестування окремих "одиниць" програмного забезпечення для перевірки, чи працюють вони правильно і відповідають очікуваному результату. Одиниця може бути функцією, методом, класом або іншим незалежним компонентом програми.

Опис процесів тестування

Таблиця 4.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Мета тесту | Елемент опису | | | |
| Початковий стан системи | Схема проведення тесту | Очкуванний результат | Стан системи після тесту |
| Відкриття допомоги | Головний єкран | Натискається кнопка налаштування і обираєтсья з меню налаштувань Onboarding | Onboarding з поясненнями з’явився | -  Запустився Onboarding з поясненнями |

Таблиця 4.2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Мета тесту | Елемент опису | | | |
| Початковий стан системи | Схема проведення тесту | Очкуванний результат | Стан системи після тесту |
| Створити пакет | Пакету немає | Користувач натискає мнопку налаштування, вводить назву | Створюеться пустий пакет | Створився пустий пакет  Запустився Onboarding з поясненнями |

Таблиця 4.3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Мета тесту | Елемент опису | | | |
| Початковий стан системи | Схема проведення тесту | Очкуванний результат | Стан системи після тесту |
| Створити задвання і запонити пакет | Пустий пакет | Тестер заходить до пакету, натискає додати задання, додає компоненти до завдання | Створюеться завдання в пакеті | Створюеться завдання в пакеті |

Таблиця 4.4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Мета тесту | Елемент опису | | | |
| Початковий стан системи | Схема проведення тесту | Очкуванний результат | Стан системи після тесту |
| Видаляти пакети | Пустий пакет | Свайп вліво, натиснути кнопку видалити | Пакет зник | Пакет зник |

Таблиця 4.5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Мета тесту | Елемент опису | | | |
| Початковий стан системи | Схема проведення тесту | Очкуванний результат | Стан системи після тесту |
| Видаляти завдання | Не пустий пакет | Свайп вліво, натиснути кнопку видалити | Завдання видаляється і зникає | Завдання видаляється і зникає |

Таблиця 4.6

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Мета тесту | Елемент опису | | | |
| Початковий стан системи | Схема проведення тесту | Очкуванний результат | Стан системи після тесту |
| Видаляти копомнент | Не пусте завдання | Свайп вліво, натиснути кнопку видалити | Компонент видаляється і зникає | Компонент видаляється і зникає |

Таблиця 4.7

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Мета тесту | Елемент опису | | | |
| Початковий стан системи | Схема проведення тесту | Очкуванний результат | Стан системи після тесту |
| Авторизація | Неавторизована система | Тестер заходить до магазину і пробує викласти свії пакет до магазину, вилазить авторизація, користувач авторизується | Авторизація з’являється і тестер авторизується, стан системи стає авторизованим | Авторизація з’являється і тестер авторизується, стан системи стає авторизованим |

Таблиця 4.8

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Мета тесту | Елемент опису | | | |
| Початковий стан системи | Схема проведення тесту | Очкуванний результат | Стан системи після тесту |
| Копіювання пакету | Авторизована система, тестер у магазині пакетів | Тестер заходить до магазину, вибирає пакет, свайп вліво, кнопка скопіювати користувач авторизується | Користувач скопіював пакет, він з'явився у головному меню | Пакет скопіювався, він з'явився у головному меню |

Таблиця 4.9

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Мета тесту | Елемент опису | | | |
| Початковий стан системи | Схема проведення тесту | Очкуванний результат | Стан системи після тесту |
| Викладка пакету | Авторизована система, тестер у магазині пакетів | Нажати “Share” у магазині | Пакет загружається до серверу, кожен інший користувач тепер може його побачити і скопіювати | Пакет загружається до серверу, кожен інший користувач тепер може його побачити і скопіювати |

Таблиця 4.10

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Мета тесту | Елемент опису | | | |
| Початковий стан системи | Схема проведення тесту | Очкуванний результат | Стан системи після тесту |
| Додати пакет у боті | Пакету немає в боті | Закинути в бота пакет, реплаем зробити команду add | Пакет додається до боту | Пакет додається до боту |

Таблиця 4.11

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Мета тесту | Елемент опису | | | |
| Початковий стан системи | Схема проведення тесту | Очкуванний результат | Стан системи після тесту |
| Видалити пакет у боті | Пакету є в боті | Команда rm package\_index | Пакет видаляється з боту | Пакет видаляється з боту |

Таблиця 4.12

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Мета тесту | Елемент опису | | | |
| Початковий стан системи | Схема проведення тесту | Очкуванний результат | Стан системи після тесту |
| Паказати список активних пакетів у боті | Бот має пакети | Команда pls | Показує всі пакети в боті | Показує всі пакети в боті |

Таблиця 4.13

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Мета тесту | Елемент опису | | | |
| Початковий стан системи | Схема проведення тесту | Очкуванний результат | Стан системи після тесту |
| Надсилання повідомлення у боті | Бот має якісь пакети | Дочекатись на повідомлення, Один з «Appear» компонентів повернув True | Повідомлення прийшло до користувача | Повідомлення прийшло до користувача |

## Розгортання програмного забезпечення

### iOS застосунок

Для того, щоб викласти iOS застосунок потрібно використовувати Xcode і App Store Connect.

App Store Connect є онлайн-платформою, розробленою компанією Apple, яка дозволяє розробникам програм для iOS, iPadOS, macOS, watchOS та tvOS керувати процесом розміщення та поширення своїх програм через офіційний App Store.

App Store Connect надає розробникам можливість вивантажувати свої програми, проводити тестування, встановлювати ціни, керувати версіями програм та здійснювати різні аспекти маркетингу. На цій платформі розробники також можуть відстежувати метрики продажів, аналізувати дані про користувачів та звітувати про доходи.

App Store Connect є важливим інструментом для розробників додатків Apple, оскільки вона надає доступ до широкої аудиторії користувачів Apple та допомагає в управлінні та просуванні їхніх програм у магазині.

A screenshot of a computer

Description automatically generated with medium confidence

Рисунок 4.1 — Архіви iOS застосунку

Загальні кроки, які я виконав, щоб розмістити застосунку в App Store.

* Зареєструвався в Apple Developer Program: Я зареєструвався в Apple Developer Program, щоб мати можливість розміщувати мої застосунки в App Store. Це дозволило мені отримати доступ до необхідних інструментів розробки і можливостей розміщення в App Store.
* Підготував застосунок: Я підготував свій застосунок для розміщення в App Store. Це включало в себе переконання, що мій застосунок відповідає всім вимогам та стандартам, встановленим Apple, і що він працює на платформах iOS, iPadOS, macOS, watchOS або tvOS.
* Зареєстрував застосунок в App Store Connect: Я зареєстрував свій застосунок в App Store Connect, платформі для керування застосунками в App Store. Тут я заповнив всі необхідні метадані, такі як опис, скріншоти, іконки та інші важливі дані про мій застосунок.
* Провів тестування: Я провів тестування мого застосунку, щоб переконатися, що він працює належним чином і не має помилок або проблем. Це включало в себе тестування функціональності, сумісності, безпеки та продуктивності застосунку.
* Встановив ціни та план розповсюдження: Я встановив ціни для свого застосунку, якщо він є платним, або обрав безкоштовний варіант. Також я обрав країни, в яких бажав розповсюджувати свій застосунок.
* Надіслав застосунок на перевірку: Після підготовки і налаштування всіх необхідних даних, я надіслав свій застосунок на перевірку Apple. Apple проводить ретельний технічний та контентний огляд, щоб переконатися, що застосунок відповідає їхнім вимогам та стандартам.
* Отримав схвалення та розмістив застосунок: Після успішного проходження перевірки, мій застосунок отримав схвалення від Apple, і я зміг розмістити його в App Store. Застосунок став доступним для користувачів для завантаження та встановлення.

A screenshot of a phone

Description automatically generated with low confidence

Рисунок 4.2 — вигляд iOS застосунку в AppStore

### Telegram Bot

Для того, щоб захостити бота я використовув Google Cloud

Google Cloud є хмарною платформою, розробленою компанією Google, яка надає широкий спектр послуг інфраструктури, обчислень, зберігання даних, аналітики, штучного інтелекту та інших послуг, що допомагають підприємствам розробляти, розгортати та керувати своїми додатками та сервісами в хмарному середовищі.

Google Cloud пропонує гнучкі та масштабовані рішення для хмарних обчислень, включаючи віртуальні машини, контейнери, сервери без обслуговування (serverless) та інші сервіси. Він також надає можливості зберігання даних, такі як об'єктне сховище, блокове сховище, бази даних та інструменти для аналітики даних.

Google Cloud також має широкий набір інструментів для розробників, включаючи сервіси управління API, інструменти для розробки мобільних додатків, машинного навчання та інтеграції з іншими сервісами Google, такими як Google Maps, Google Analytics та інші.

Ця платформа дозволяє організаціям будувати потужні, масштабовані та безпечні додатки та сервіси, використовуючи потужність хмарних ресурсів та інструменти, надані Google.

Я виконав такі кроки для того, щоб захостити бота на Swift

* 1. Взяв безкоштовних 200 доларів в місяць
  2. Створив server instance з Ubuntu
  3. Настроїв сервер, а саме скачав компілятор Swift, скачав код бота, скопіював
  4. Запустив бота

### Back-end програма

Як встановити серверну частину (Database + API) Встановлюються файли php та запускаються файли sql для створення таблиць та даних схеми бази даних.

Використовую вхід через адміністративну панель cPanel хостингу, підключитись до сервера хостингу https://server148.web-hosting.com/cpanel (або https://proq.cc/cpanel) username: x\*\*\*\*akr password: \*\*\*\*\*\*\*\*

У панелі керування cPanel запустити утиліту phpAdmin для створення схеми бази даних MySQL (найменування схеми: внутрішнє\_ім'я\_хостингу\_proq\_cc)

Створити схему бази даних та користувача бази даних з правами адміністратора командами CREATE DATABASE та CREATE USER, як це наводиться у скрипті /sql/proq-user.sql (або утилітою MySQL Database Wizard)

Запустити скрипт створення таблиць бази даних /sql/proq-schema.sql в утиліті phpAdmin (закладка SQL) або за допомогою програмного забезпечення Dbeaver

Запустити скрипт створення тестових даних для налагодження роботи бази даних /sql/proq-data.sql в утиліті phpAdmin (закладка SQL) або за допомогою програмного забезпечення Dbeaver

Виконати перенесення файлів /php/\*.\* у папку /HOME/service/api/ та зберегти на сервері (хостингу) за допомогою утиліти FileManager з адміністративної панелі cPanel

Виконати налаштування файлу /php/index.php (основний файл для запуску серверної програми для роботи API)

Редагування рядків змінних для підключення до бази даних (схема базхи даних, ім'я користувача та пароль) – вказати налаштування підключення. Для налагодження роботи увімкнути запис у файл протоколювання service.log (вказати значення true для параметра запису у файл лог, у перевійному кроці файлу index.php)

При необхідності налагодити роботу за допомогою запуску API запитів із рядка браузера, наведених у тестовому файлі /doc/readme\_api.txt

## Підтримка програмного забезпечення

### iOS застосунок

Будь-який користвуч, що встановив застосунок з AppStore має можливіть встановлювати оновлення. Оновлення викладуються за такою ж схемою як і просто викладка застосунку.

A screenshot of a phone

Description automatically generated with medium confidence

Рисунок 4.3 — Вигляд оновлення в AppStore

### Telegram Bot

Telegram Bot отримує оновлення не залежно від дій користувача, для цього потрібно просто загрузити нову версію програми до Google Cloud користувачі навіть нічого не помітять.

### Back-end програма

Back-end програма тримує оновлення не залежно від дій користувача, для цього потрібно просто загрузити нову версію програми до Google Cloud користувачі навіть нічого не помітять.

## Інструкція користувача

A screenshot of a phone

Description automatically generated with medium confidence

Рисунок 4.4 — Спливаюче вікно вводу назви для нового пакету

Натиснути New Package. Відкриється вікно для вводи назви пакету. Введіть назву, натисніть Save.

Screens screenshot of a phone

Description automatically generated with low confidence

Рисунок 4.5 — Вікна для налаштування пакету завдань

Натисніть на пакет, налаштуйте його так як потрібно. Додайте завдання, змініть назву, якщо потрібно.

Screens screenshot of a phone

Description automatically generated with medium confidence

Рисунок 4.6 — Вікна налаштування компонентів

Налаштуйте завдання так як Вам потрібно. Додайте потрібну кількість компонентів, налаштовуйте кожний окремий компонент

Screens screenshots of a phone

Description automatically generated with medium confidence

Рисунок 4.7 — Всі вікна налаштувань

В налаштуваннях можна подивитись інформацію про застосунок. Подивитись як підключити пакет до бота, вийти з профілю.

Screens screenshot of a phone

Description automatically generated with medium confidence

Рисунок 4.8 — Магазин і авторизація

Якщо Ви в додатку вперше, заходячі до магазина потрібно зареєструватися. Введіть дані і натисніть Done. Ви отримаєте доступ до користувацьких пакетів, скопіюйте той, який сподобається за допомогою жесту перетаскування пальцем в ліво. Якщо є бажання поділитися пакетом, натисніть кнопку Share і виберіть потрібний.

Screens screenshots of a phone

Description automatically generated with medium confidence

Рисунок 4.9 — Процес підключення пакету до бота

Відправте до бота пакет. Для цього довго натисніть на потрібний пакет і виберіть Connect з контекстного меню. Відправте пакет в телеграм боту. Якщо є бажання використовувати пакет разом з кимось, відправте пакет до групи, де є бот. У боті виберіть пакет і напишіть команду add.

Screens screenshots of a chat

Description automatically generated with medium confidence

Рисунок 4.10 — Команди бота

Додавайте пакетів до бота скільки потрібно. Видаляйте за допомогою команди rm<Index>. Отримуйте повідомлення від боту.

## Висновок до розділу

У розділі було розкрито процес завантаження та оновлення застосунків в App Store - офіційному магазині додатків для пристроїв. Цей магазин є єдиним законним місцем, де користувачі можуть завантажувати та оновлювати застосунки на своїх пристроях. Описав використання системи для користувачів

# ВИСНОВОК

Зроблено огляд предметної області автоматизації процесів часу людини. Проаналізовано існуючі рішення, виявлено їх проблеми і те як їх вирішити. Виявлено проблем при створенні подібних систем і як їх уникнути. Поставлено конкретні вимоги

Поставлена формалізована задача. Описані алгоритми роботи в програмі. Створено архітектуру системи. Поставлені задачі до реалізації систґми.

Описано засоби розробки. Проаналізовані технічні засоби і обґрунтовано їх використання. Описані методолгії, які вискористовуються в системах

Протестовано систему, виявлено та виправлено всі помилки. Розгорнуто всі системні частини, описано їх оновлення для користувача.

# ПЕРЕЛІК ПОСИЛАНЬ

1. Apple Develper Documentation. URL: <https://developer.apple.com/develop/> (дата звернення 15.04.2023р)
2. Jonfir, IoC Article. URL: <https://jonfir.github.io/posts/ioc-ios/> (дата звернення 17.05.2023р)
3. Type-Driven Design in Swift. URL: <https://www.youtube.com/watch?v=pbVjkY9fS8c> (дата звернення 14.04.2023р)
4. PHP Documentation. URL: <https://www.php.net/docs.php> (дата звернення 14.04.2023р)
5. Apple WWDC23. URL: <https://developer.apple.com/wwdc23/> (дата звернення 12.05.2023р)
6. WWDC Notes. URL: <https://www.wwdcnotes.com/> (дата звернення 11.05.2023р)
7. Apple Swift Package Manager Documentation. URL: <https://docs.swift.org/package-manager/> (дата звернення 14.05.2023р)
8. Telegram Bot Documentation. URL: <https://core.telegram.org/bots/api> (дата звернення 05.05.2023р)
9. RxSwift -Как это работает?. URL: <https://www.youtube.com/watch?v=gV22Yy1Hxzw&t=1540s> (Дата звернення: 05.05.2023)
10. YouTube video: "iOS App Development Tutorial for Beginners". URL: <https://www.youtube.com/watch?v=3ReoqZCxg08> (Дата звернення: 29.05.2023)
11. GitHub repository: "Swift Design Patterns". URL: <https://github.com/eleev/swift-design-patterns> (Дата звернення: 05.05.2023)
12. YouTube video: "Introduction to SwiftUI". URL: <https://www.youtube.com/watch?v=RWrDahv8m0I&t=404s> (Дата звернення: 19.03.2023)
13. YouTube video: "Mastering SwiftUI". URL: <https://www.youtube.com/watch?v=iWYpFWQpvkA&t=2398s> (Дата звернення: 01.06.2023)
14. YouTube video: "Advanced iOS Development with SwiftUI". URL: <https://www.youtube.com/watch?v=vJikZvTfrdg&t=1950s> (Дата звернення: 15.05.2023)
15. YouTube video: "Building Interactive iOS Apps with SwiftUI". URL: <https://www.youtube.com/watch?v=AqrMPlzB8iI&t=1493s> (Дата звернення: 07.04.2023)
16. YouTube video: "SwiftUI Masterclass". URL: <https://www.youtube.com/watch?v=OD1uGsM1shg&t=2454s> (Дата звернення: 02.06.2023)
17. YouTube video: "Introduction to iOS App Development". URL: <https://www.youtube.com/watch?v=ewZ85qQIuYQ> (Дата звернення: 01.06.2023)
18. YouTube video: "iOS App Design and Development Tutorial". URL: <https://www.youtube.com/watch?v=B_T8o1vP4H8> (Дата звернення: 28.05.2023)
19. YouTube video: "SwiftUI Crash Course". URL: <https://www.youtube.com/watch?v=aSoWDKonXew&list=PLJfJvphK-POx-wz_e9vcMnY3IhHi9MSwH&index=2> (Дата звернення: 25.04.2023)
20. Article: "iOS Navigation Best Practices". URL: <https://frankrausch.com/ios-navigation> (Дата звернення: 15.03.2023)
21. Website: "Refactoring Guru - Design Patterns". URL: <https://refactoring.guru/design-patterns/> (Дата звернення: 30.05.2023)
22. GitHub repository: "DesignPattern" by artkirillov. URL: <https://github.com/artkirillov/DesignPattern/> (Дата звернення: 17.04.2023)