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	Затвердження теми	01.04.2023р	
2	Аналіз проблем створення програмних систем організації часу людини та існуючих рішень	24.04.2023р	
3	Формулювання завдання на проектування та вимог до системи	01.05.2023р	
4	Розробка архітектури та алгоритмів роботи системи	05.05.2023р	
5	Аналіз та вибір інструментальних засобів проектування	15.05.2023р	
6	Розробка програмних модулів системи	29.05.2023р	
7	Тестування програмних модулів системи	05.06.2023р	
8	Розробка технічною документації	10.06.2023р	

Студент Дмитро ОСТАПЧЕНКО

Керівник Марія СОЛДАТОВА

АНОТАЦІЯ

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

Проєкт містить 68 с. тексту, 16 рисунків, 17 таблиць, посилання на 24 літературні джерела, 1 дадаток та 6 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 68 pages. text, 16 figures, 17 tables, references to 24 literary sources, 1 appendix and 6 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.

r

Пояснювальна записка
до дипломного проєкту
на тему: « Інтерактивно-компонентний планувальник організації часу людини»

Київ – 2023 року

Но	Ф	Позначення	Найменування	Кі		Примітка
		ІК91.050БАК.005 ТЗ	Пояснювальна записка	69		
		ІК91.050БАК.005 Д1	**Інтерактивно-компонентний**	1		
			планувальник організації часу			
			людини			
			Схема БД			
		ІК91.050БАК.005 Д2	**Інтерактивно-компонентний**	1		
			планувальник організації часу			
			людини			
			Схема навігації застосунку			
		ІК91.050БАК.005 Д3	**Інтерактивно-компонентний**	1		
			планувальник організації часу			
			людини			
			Схема архітектури системи			
		ІК91.050БАК.005 Д4	**Інтерактивно-компонентний**	1		
			планувальник організації часу			
			людини			
			Блок-схема логіки бота			
		ІК91.050БАК.005 Д5	**Інтерактивно-компонентний**	1		
			планувальник організації часу			
			людини			
			Блок-схема роботи iOS			
			застосунку			
		ІК91.050БАК.005 Д6	**Інтерактивно-компонентний**	1		
			планувальник організації часу			
			людини			
			Блок-схема роботи додатку API			
			серверу			

			1.1.1.1.1.1 ІК91.031БАК.005 ТП		
		№			
	Остапченко Д.О.		1.1.1.1.1.2 Інтерактивно-компонентний планувальник організації часу людини. Відомість проєкту.	Літ.	Аркуш
	Колеснік			1	1
				КПІ ім. Ігоря Сікорського	

ЗМІСТ

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 24 найменувань та 1 додатку. Графічна частина включає 6 креслеників формату А3. Загальний обсяг 68 сторінок.

1 АНАЛІЗ ПРОБЛЕМ СТВОРЕННЯ ПРОГРАМНИХ СИСТЕМ ОРГАНІЗАЦІЇ ЧАСУ ЛЮДИНИ

1.1 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.

1.2 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, 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.

1.3 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.

1.4 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.

2 АРХІТЕКТУРА ІНТЕРАКТИВНО-КОМПОНЕНТНОГО ПЛАНУВАЛЬНИКА ЗАВДАНЬ

2.1 Формалізоване подання процесу планування завдань

2.1.1 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.

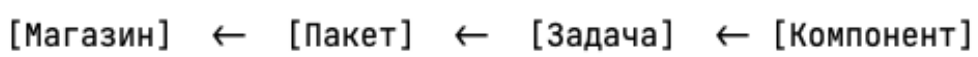


Рисунок 2.1 — Узагальнена схема підпорядкованості елементів процессу планування завдань

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

2.1.2 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.

2.2 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

ІК91.031БАК.005 Д3 детально показує архітектуру системи, візуально представляє всі компоненти описані вище.

На рисунку 2.2 представлено схему взаємозв'язку функцій планувальника завдань.

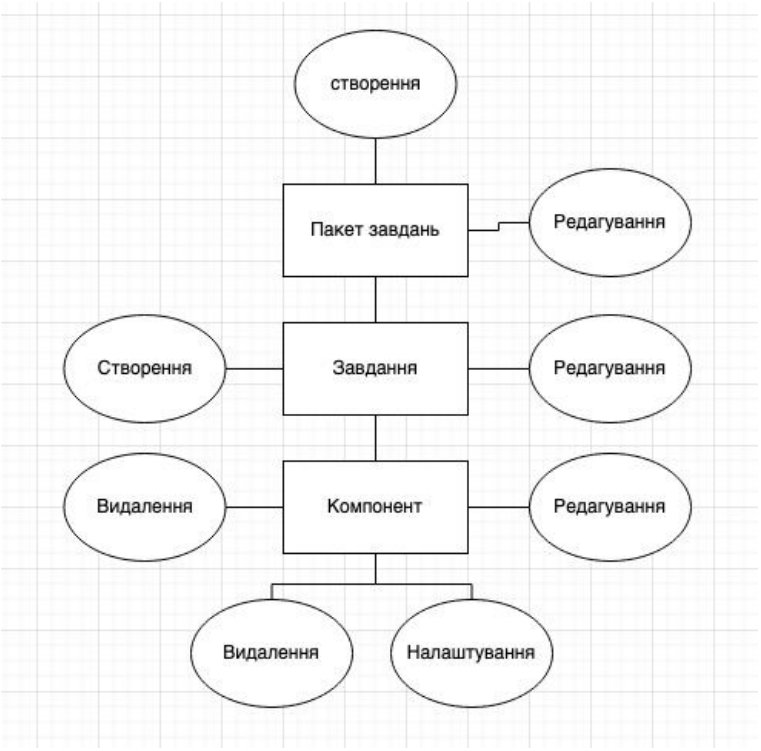


Рисунок 2.2 — Архітектура функцій планування завдань

2.3 Функції інтерактивного-компонентного планувальника завдань

2.3.1 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 2.3

Рисунок 2.3 — 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 налаштування Onboarding	Зайти до налаштувань	Користувач натискає кнопку налаштування і обирає з меню налаштувань	-	Запустився Onboarding з поясненнями

					Onboarding		

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	-	Створюється 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	Користувач	Користувач натискає додати задання	Бути у пакеті	Користувач заходить до пакету, натискає	-	Створюється завдання в пакеті

	завдання ми				додати задання, додає компоне нти до завдання		

Таблиця 2.4 – Специфікація функцій клієнта планувальника завдань

Назва функції і її позначе ння	Елемент опису						
	Goals	Actors	Trigger	Pre-conditi ons	Flow of Events	Extensi on	Post-Condition
Створит и задванн я і запонит и пакет	Надати можливіс ть наповнюв ати пакети завданн ями	Користу вач	Користу вач натискає додати задання	Бути у пакеті	Користу вач заходить до пакету, натискає додати задання, додає компоне нти до завдання	-	Створюет ься завдання в пакеті

Таблиця 2.5 – Специфікація функцій клієнта планувальника завдань

Назва функції і її	Елемент опису						
	Goals	Actors	Trigger	Pre-conditio ns	Flow of Events	Extensi on	Post-Condition

позначення							
Видалити 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
Видалити 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

Видалити 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 свій 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	Користувач	Нажати + у магазині	Бути авторизованим	Зайти до магазину, натисну	-	Пакет загружається до серверу, кожен

	до магазину				ти кнопку +, вибрати пакет		інший користувач тепер може його побачити і скопіювати

2.3.2 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 2.4

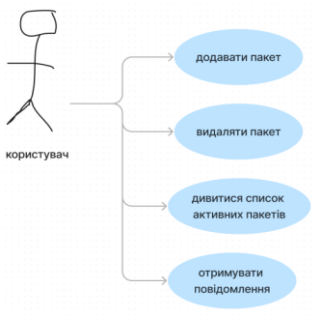


Рисунок 2.4 — Використання основних функцій в програмному забезпеченні бота інформаційних повідомлень

В таблицях 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, реплаєм 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 видалят и 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 поверну в 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	Користувач	Команда pls	Бути у боті	Команда pls	-	Показує все пакети в боті

	список пакетів						

2.4 Вимоги до програмного забезпечення

2.4.1 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>.x.x.x (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.

2.4.2 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).

2.4.3 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 C/C++), 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 C++. 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.

3 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 це Xcode (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 це Android 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.

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. [16] [5]

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. [12] [14] [15]

Фреймворк SwiftUI є новим офіційним фреймворком для розробки UI під iOS. [13] 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. [5]

3.1.1 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.

3.1.2 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

[1] 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 [3]

3.2.2 Організація окремого модуля

В системі використовується стандарт розробки програмного забезпечення “MVP” (Model-View-Presenter) [17]. 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.

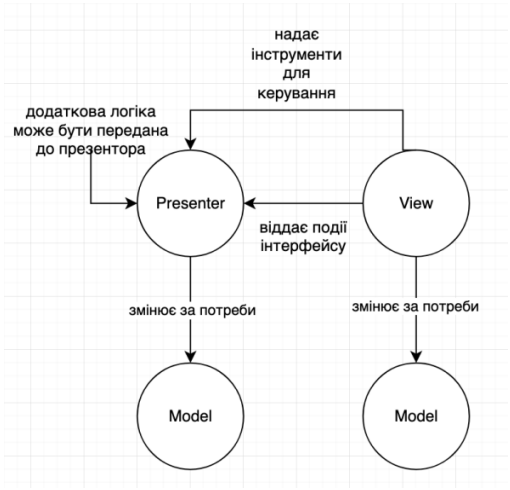


Рисунок 3.1 — Схема стандарту MVP в програмному забезпеченні додатка користувача до мобільного пристрою iOS

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

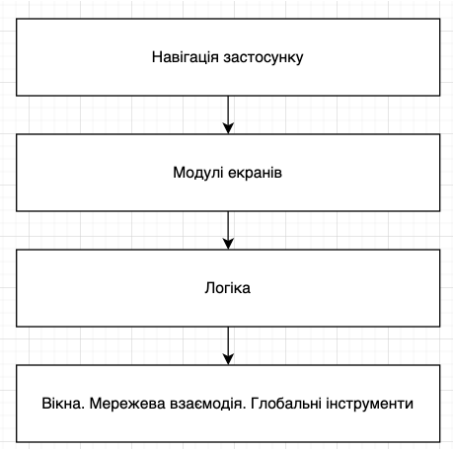


Рисунок 3.2 — Шари програми в програмному забезпеченні додатка користувача до мобільного пристрою iOS

Програмний шар «Навігація застосунку» знаходиться на вищому рівні, який базується на розроблених модулях і формує дерево програмного потоку даних. [23] [1] 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 модульність. 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.

ІК91.031БАК.005 Д2 представляє 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.

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 ІК91.031БАК.005 Д5

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). [8] [7]

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 ІК91.031БАК.005 Д4

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 ІК91.031БАК.005 Д1

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. [4]

Запити до даних додаток API приймає за протоколом HTTP за допомогою методів POST та GET з використанням рекомендацій технології REST API [24]

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 ІК91.031БАК.005 Д6

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

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

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.

4 ВПРОВАДЖЕННЯ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ІНТЕРАКТИВНО-КОМПОНЕНТНОГО ПЛАНУВАЛЬНИКА ЗАВДАНЬ

4.1 Організація тестування програмного забезпечення

Метою проведення тестування програмного забезпечення є визначення рівня готовності і придатності інтерактивно-компонентного планувальника завдань до впровадження і практичного застосування. В процесі тестування оцінюються наступні показники якості програмного забезпечення:

1)Виявлення наявних помилок, дефектів або некоректного функціонування програмного забезпечення. Допомагає забезпечити належну готовність продукту перед його випуском на ринок.

2)Перевірка відповідності функціональним вимогам. Дозволяє забезпечити відповідність вхідним специфікаціям на систему, гарантування виконання запланованих функції і задоволення потреб користувача.

3)Підтримка надійності. Допомагає переконатися, що програмне забезпечення працює безперебійно і надійно в різних умовах та сценаріях за допомогою перевірки стабільності, витривалості та відмовостійкості системи.

4)Перевірка рівня ефективності. Допомагає виявити та усунути недоліки в роботі програмного забезпечення, що можуть призводити до його низької продуктивності або швидкодії.

5)Перевірка якості. Це включає перевірку функціональності, правильності та зручності використання програми шляхом ідентифікації та усунення проблем.

Для виявлення визначених показників програмного забезпечення використовується функціональне та модульне тестування.

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.

Модульне тестування - це процес тестування окремих модулів програмного забезпечення для перевірки, чи працюють вони правильно і відповідають очікуваному результату. Модулем може бути функція, метод, клас або інший незалежний компонент програмного забезпечення.

Основним об’єктом тестування програмного забезпечення виступає додаток користувача до мобільного пристрою iOS. Здійснюється перевірка інтерфейсу користувача, результати якої підтверджують узгоджену роботу підсистем програмного комплексу інтерактивно-компонентного планувальника завдань, а також очікуваний результат згідно з вимогами, які ставилися до розробки програмної системи. Перелік проведених тестових операцій над додатком користувача до мобільного пристрою iOS наведено в таблиці 4.1.

Таблиця 4.1 – Склад тестових операцій перевірки додатку користувача

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

Створити пакет	Пакету немає	Користувач натискає мнопку налаштування, вводить назву	Створюється пустий пакет	Створився пустий пакет Запустився Onboarding з поясненнями
Створити задвання і запонити пакет	Пустий пакет	Тестер заходить до пакету, натискає додати задання, додає компоненти до завдання	Створюється завдання в пакеті	Створюється завдання в пакеті
Видаляти пакети	Пустий пакет	Свайп вліво, натиснути кнопку видалити	Пакет зник	Пакет зник
Видаляти завдання	Не пустий пакет	Свайп вліво, натиснути кнопку видалити	Завдання видаляється і зникає	Завдання видаляється і зникає
Видаляти копомнент	Не пусте завдання	Свайп вліво, натиснути кнопку видалити	Компонент видаляється і зникає	Компонент видаляється і зникає
Авторизація	Неавторизована система	Тестер заходить до магазину і пробує викласти свії пакет до магазину, вилазить	Авторизація з'являється і тестер авторизується, стан системи стає авторизованим	Авторизація з'являється і тестер авторизується, стан системи стає авторизованим

		авторизація, користувач авторизується		
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
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
Додати пакет у боті	Пакету немає в боті	Закинути в бота пакет, реплаєм зробити команду add	Пакет додається до боту	Пакет додається до боту
Видалити пакет у боті	Пакету є в боті	Команда rm package_index	Пакет видаляється з боту	Пакет видаляється з боту
Паказати список активних пакетів у боті	Бот має пакети	Команда pls	Показує всі пакети в боті	Показує всі пакети в боті

Отримати повідомлпення	Бот вирішив відправити повідомлення	Почекати до часу повідомлення	Повідомлення прийшло на телефон	Повідомлення прийшло на телефон

4.2 Розгортання програмного забезпечення системи

4.2.1 Розгортання додатку користувача на мобільного пристрою iOS

Розгортання iOS додатка потребує використання платформ Xcode та App Store Connect.

App Store Connect – це розроблена компанією Apple платформа, яка дозволяє розробникам програм для iOS, iPadOS, macOS, watchOS та tvOS керувати процесом розміщення та поширення своїх програм через офіційний магазин застосунків App Store. Вона надає розробникам можливість вивантажувати програми, проводити тестування, встановлювати ціни, керувати версіями програм та здійснювати операції маркетингу. Ця платформа здійснює відстежування метрик, аналіз даних та звітів. Фактично App Store Connect слугує інструментом для управління та просування програмного забезпечення розробників.

На рисунку 4.1 представлено розроблені версії мобільного додатку користувача інтерактивно-компонентного планувальника завдань, що вивантажені на App Store Connect.

Рисунок 4.1 — Архіви iOS застосунку

Для розгортання програмного забезпечення додатку користувача на мобільному пристрої iOS в App Store виконані наступні дії:

1. реєстрація в Apple Developer Program для можливості розміщувати додатки в App Store;
2. підготовка додатка до розміщення в App Store згідно вимогам та стандартам, встановленим Apple, перевірка можливості працювати на платформах iOS, iPadOS, macOS, watchOS або tvOS;
3. реєстрація додатка в App Store Connect та введення даних: опису, зображень, іконки та іншої інформації;
4. тестування додатка, що включає тестування функціональності, сумісності, безпеки та продуктивності за стосунку;
5. встановлення ціни додатку та плана розповсюдження на безкоштовних умовах;
6. відправлення додатку на перевірку для ретельного технічного та зовнішнього огляду, підтвердження, що застосунок відповідає вимогам та стандартам;
7. отримання схвалення та розміщення додатка, після чого застосунок доступний для користувачів для завантаження і встановлення.

На рисунку 4.2 приведено зображення розміщеного програмного забезпечення додатку користувача до мобільного пристрою iOS в App Store.

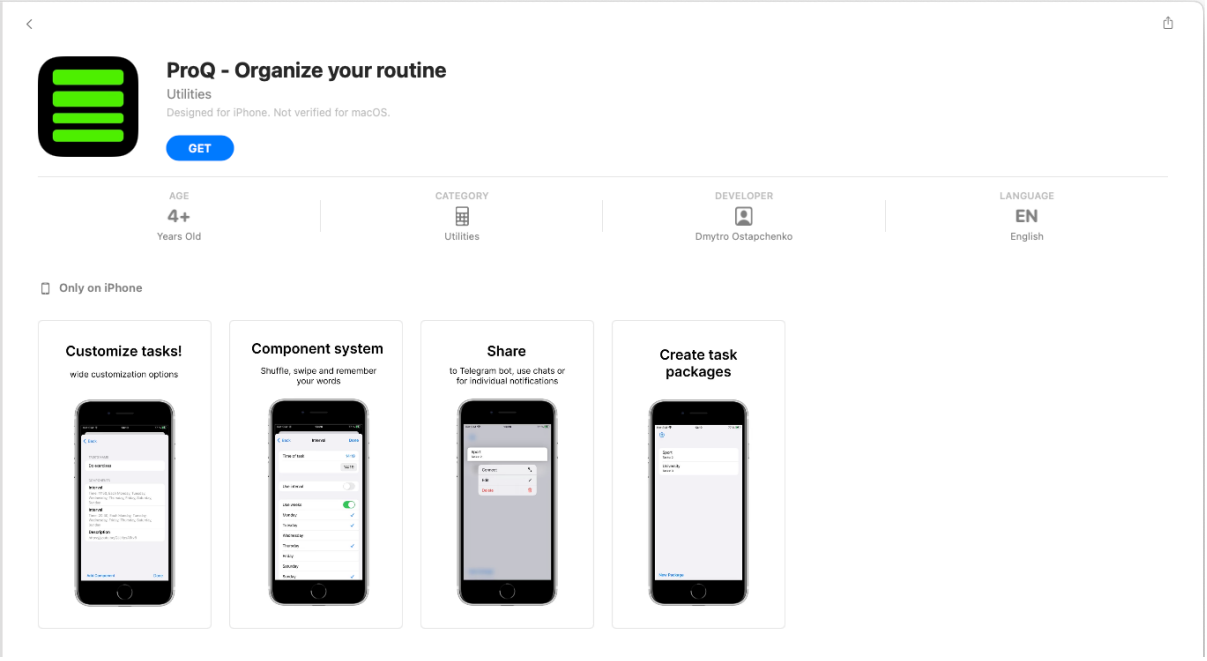


Рисунок 4.2 — Розміщення додатку iOS в AppStore

4.2.2 Розгортання чат боту підсистеми інформаційних повідомлень

Розгортання та встановлення чат боту підсистеми інформаційних повідомлень виконується в хмарі через мережу Інтернет. Після реєстрації виконується завантаження файлів додатку Swift в хмарне середовище Google Cloud.

Google Cloud - хмарна платформа розроблена компанією Google, яка надає широкий спектр послуг інфраструктури, обчислень, зберігання даних, аналітики, штучного інтелекту та інших послуг, що допомагають розробляти, розгортати та керувати додатками і сервісами.

Розгортання чат боту системи повідомлень виконується за допомогою адміністративної утиліти в хмарі Google Cloud через виконання наступних дій:

- створення серверного середовища на платформі оперативної системи Ubuntu;
- налагодження сервера та завантаження компілятора файлів Swift;
- завантаження Swift файлів чат боту системи повідомлень;
- завантаження файлів на виконання.

4.2.3 Розгортання бази даних та API додатку сервера

Розгортання та встановлення серверу бази даних та API додатку виконується в хмарі через мережу Інтернет та за допомогою завантаження файлів: додатку php та завантажуються файли sql для створення схеми бази даних, таблиць та тестових даних.

Розгортання виконується за допомогою адміністративної утиліти - панелі cPanel хостингу, доступ до якої надається після реєстрації на хостингу. Для розгортання бази даних та додатку API виконуються наступні дії:

- У панелі керування cPanel завантажено утиліту phpAdmin для створення схеми бази даних MySQL (найменування схеми: внутрішнє_ім'я_хостингу_proq_cc);

- Створено 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 (виконано утилітою MySQL Database Wizard);

- Завантажено файл створення таблиць бази даних /sql/proq-schema.sql в утиліті phpAdmin (закладка SQL). Альтернативно створення таблиць можливе за допомогою програмного забезпечення Dbeaver в режимі тестування;

- Завантажено файл створення тестових даних для налагодження роботи бази даних /sql/proq-data.sql в утиліті phpAdmin (закладка SQL). . Альтернативно створення даних можливо за допомогою програмного забезпечення Dbeaver в режимі тестування;

- Виконано перенесення файлів /php/*.* у папку /HOME/service/api/ та зберегжено на сервері хостингу за допомогою утиліти FileManager адміністративної панелі cPanel;

- Виконано налаштування файлу /php/index.php (основний файл для запуску програми API сервера);

- Встановлено дані для підключення до бази даних (схема бази даних, ім'я користувача та пароль). Для налагодження роботи або тестування використовувався режим запуску до файла даних роботи в режимі тестування (встановлення значення true для параметра запису у файл у файлі index.php);

- Налагодження роботи за допомогою запуску API запитів із рядка браузера, наведених у тестовому файлі /doc/readme_api.txt.

4.3 Підтримка версій програмного забезпечення системи

Підтримка версії програмного забезпечення системи виконується через процедуру оновлення на платформах його розташування.

Оновлення додатку користувача до мобільного пристрою iOS виконується автоматично в операційній системі iOS за розкладом, встановленим користувачем. Оновлення завантажуються з платформи App Store. На рисунку 4.3 представлено приклад оновлення додатка.

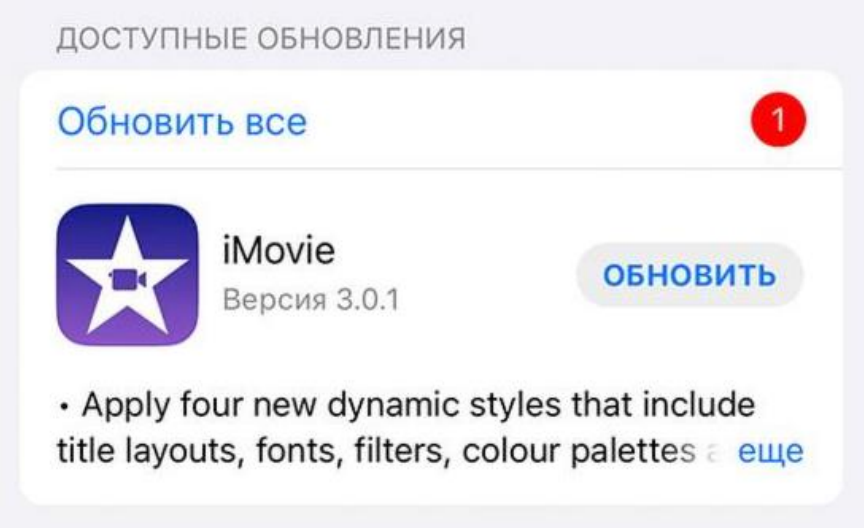


Рисунок 4.3 — Приклад оновлення додатка в AppStore

Чат бот системи повідомлень отримує оновлення в автоматичному режимі, нові версії програми завантажуються у вигляді файлів до Google Cloud.

База даних та API додатку сервера отримує оновлення в автоматичному режимі, нові версії програми завантажуються у вигляді файлів до хостингу.

4.4 Інструкція користувача додатку мобільного пристрою iOS

Інструкція користувача додатку мобільного пристрою відтворює виконання основних його функцій при роботі із пакетами завдань, налаштування компонент, входу до магазину пакетів, роботи з ботом інформаційних повідомлень.

Створення нового пакету завдань представлено на рисунку 4.4. Натиснути кнопку New Package, введіть дані назви пакету, для збереження нового пакету натисніть Save.

Рисунок 4.4 — Створення нового пакету завдань

Налаштування пакету завдань представлено на рисунку 4.5. Натисніть на пакет, налаштуйте: додайте завдання, введіть назву.

Рисунок 4.5 — Налаштування пакету завдань

Налаштування компонентів завдань представлено на рисунку 4.6. Налаштування компонентів завдання: додайте компоненти, налаштуйте окремий компонент.

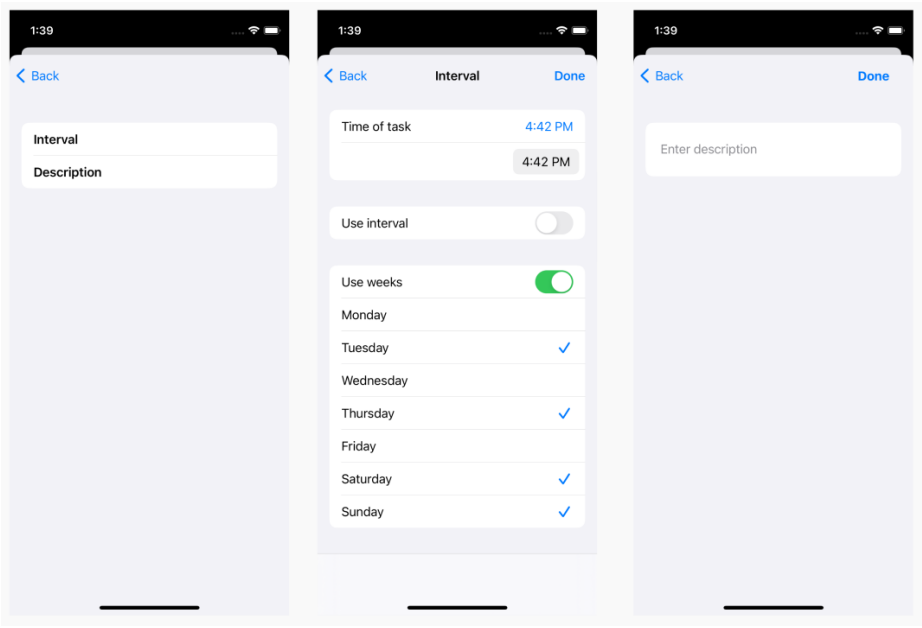


Рисунок 4.6 — Налаштування компонентів завдання

Виконання налаштування пакетів завдань представлено на рисунку 4.7. Налаштування завдання дозволяє додати необмежену кількість компонентів, що вимагає окрема задача, налаштування дозволяється для кожного окремого компонента.

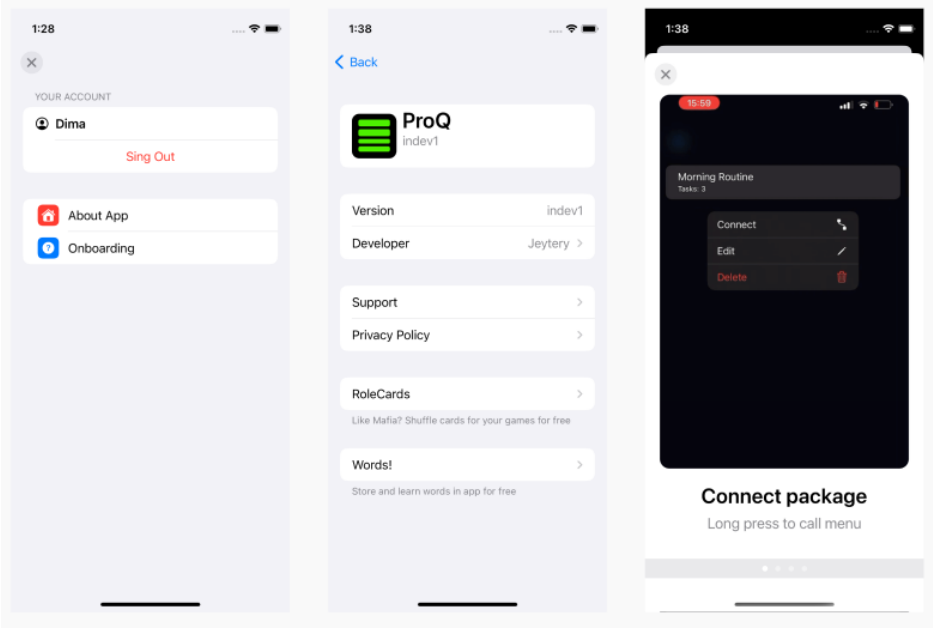


Рисунок 4.7 — Налаштування задачі

Виконання входу до магазину пакетів завдань представлено на рисунку 4.8. При першому вході до магазину пакетів завдань необхідно зареєструватися або ввести дані користувача для входу. Введіть дані

користувача та натисніть Done. Отримавши доступ до пакетів завдань, з'являється можливість отримати пакет (в режимі перетягнути Drag and Drop) або відправити до магазину, натиснувши на кнопку Share.

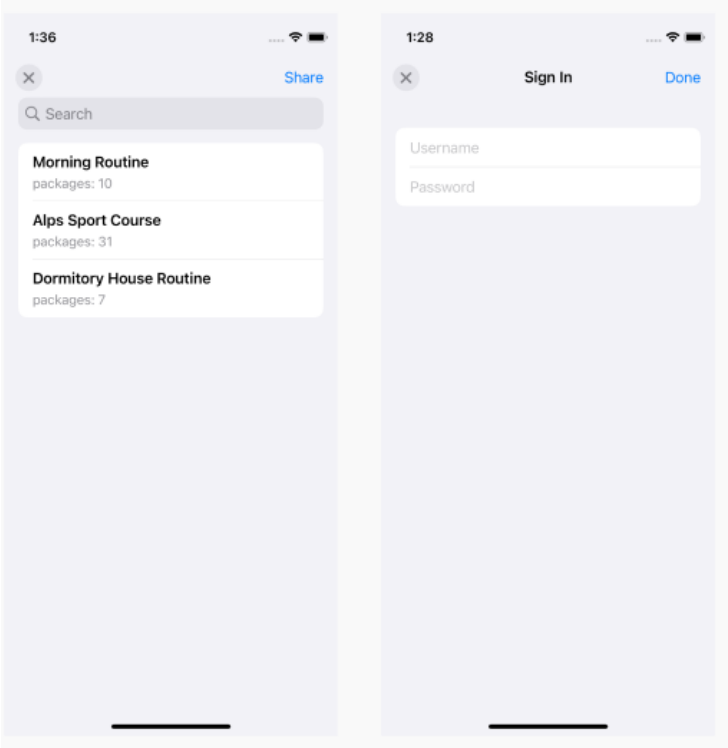


Рисунок 4.8 — Вхід до магазину пакетів завдань

Робота додатка з ботом інформаційних повідомлень представлено на рисунку 4.8. Для завантаження пакета до бота натисніть на потрібний пакет та виберіть Connect з контекстного меню. Відправте пакет до бота повідомлень для початку роботи. Для використання пакету завдань разом з групою користувачів, відправте пакет завдань до групи, де бот інформаційних повідомлень встановлено. В інформаційному вікні бота виберіть пакет завдань та введіть команду add.

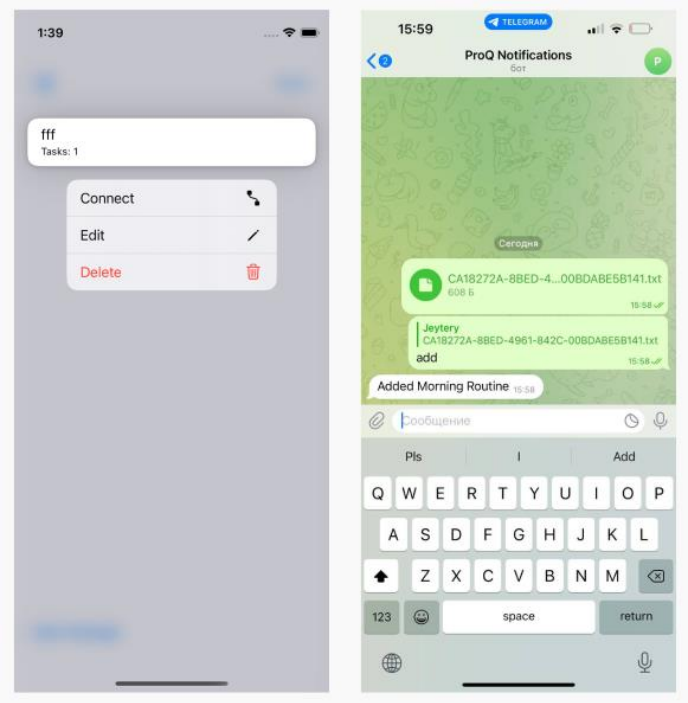


Рисунок 4.9 — Робота додатка з ботом повідомлень

Виконання команд ботом інформаційних повідомлень представлено на рисунку 4.10. Видалення пакетів завдань виконується за допомогою команди бота rm<Номер пакету завдань>. В інформаційному вікні бота відображаються повідомлення щодо роботи бота та інші повідомлення про завдання користувачів.

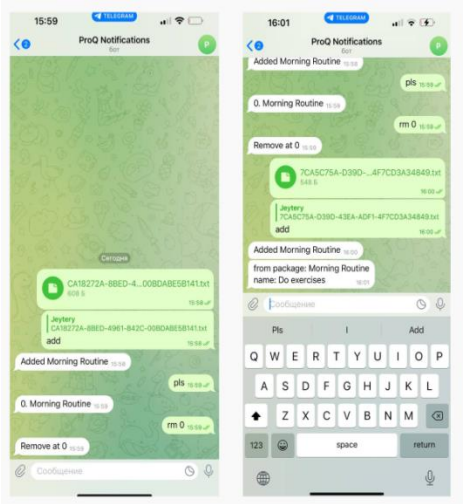


Рисунок 4.10 — Виконання команд ботом повідомлень

Висновок до розділу

У розділі розглянуто комплекс питань організації процесу тестування, завантаження і оновлення складових програмного комплексу інтерактивно-компонентного планувальника завдань - додатку користувача до мобільного пристрою iOS, чат боту системи повідомлень, бази даних та API додатку сервера, які є завершальними у проектуванні. Повне виконання умов тестування, додержання вимог до завантаження дозволило отримати працездатну систему згідно з вимогами, встановленими при проектуванні.

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. MVP in iOS. URL: https://www.javatpoint.com/ios-model-view-presenter (Дата звернення: 06.04.2023)
18. YouTube video: "Introduction to iOS App Development". URL: https://www.youtube.com/watch?v=ewZ85qQIuYQ (Дата звернення: 01.06.2023)
19. YouTube video: "iOS App Design and Development Tutorial". URL: https://www.youtube.com/watch?v=B_T8o1vP4H8 (Дата звернення: 28.05.2023)
20. 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)
21. Article: "iOS Navigation Best Practices". URL: https://frankrausch.com/ios-navigation (Дата звернення: 15.03.2023)
22. Website: "Refactoring Guru - Design Patterns". URL: https://refactoring.guru/design-patterns/ (Дата звернення: 30.05.2023)
23. GitHub repository: "DesignPattern" by artkirillov. URL: https://github.com/artkirillov/DesignPattern/ (Дата звернення: 17.04.2023)
24. Введение в REST API. URL: https://habr.com/ru/articles/483202/ (Дата звернення: 27.05.2023)

ДОДАТОК А

iOS застосунок source code: https://github.com/swiftonica/ProjectQ-iOS

Telegram Bot source code: https://github.com/Jeytery/ProjectQ-NotificationBot

Component Library source code: https://github.com/swiftonica/ProjectQ-Components2

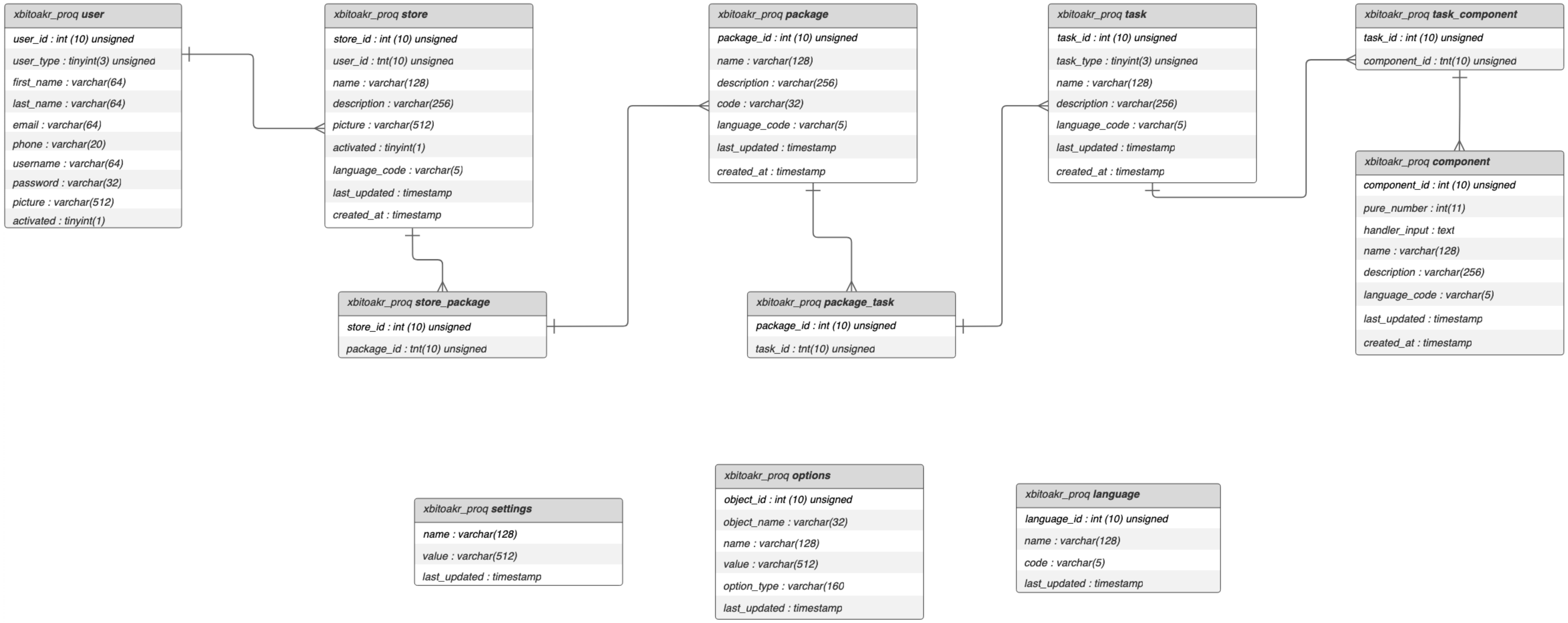
Backend source code: https://github.com/Jeytery/ProjectQ-backend

Графічний матеріал

до дипломного проєкту

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

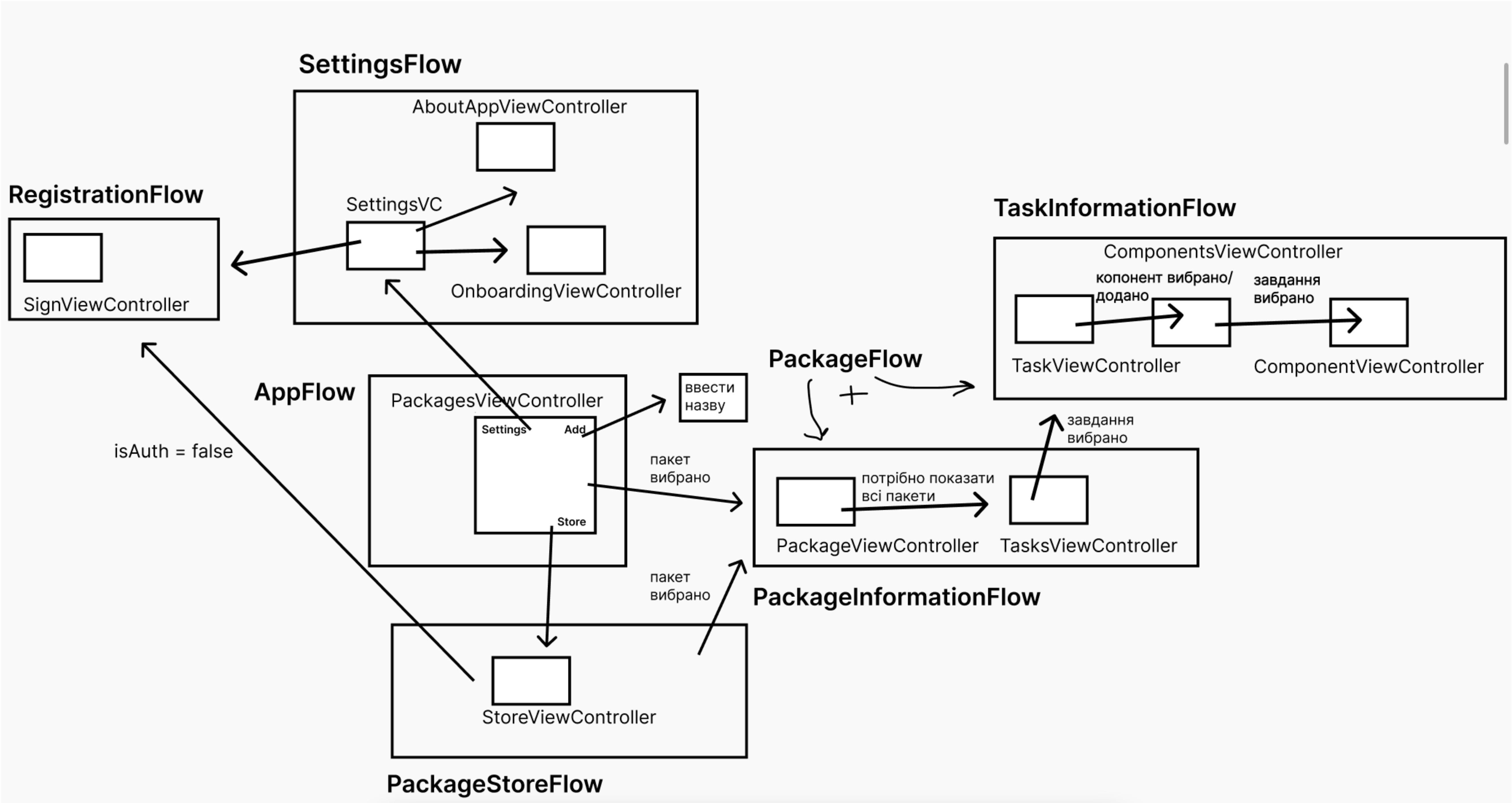
ІК91.031БАК.005 Д1



Підпис і дата | Інв. дубл. | Взам. інв. | Підпис і дата | Інв. ориг.

Зм.	Лист	докум	Підпис	Дата	ІК91.031БАК.005 Д1		
Розроб.		Остапченко Д.О.			Схема БД	Літ. / Маса / Мірило	
Перев.						Лист 1	Листів 1
Н. контр		Шинкевич М. К.				КПІ ім. Ігоря Сікорського ФІОТ, ІК-91	
Затв.		Ролік О. І.					

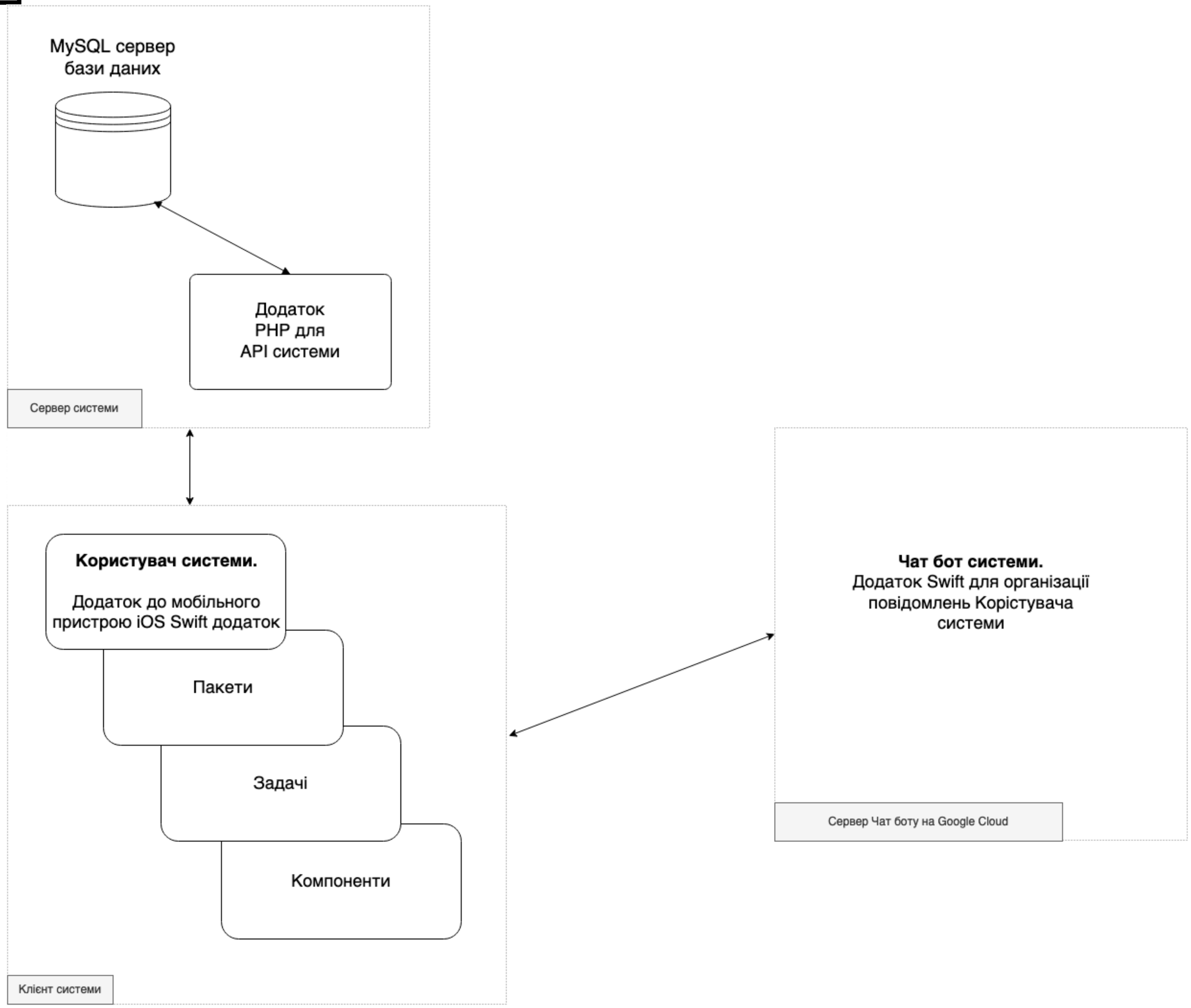
ІК91.031БАК.005 Д2



Підпис і дата | Інв. дубл. | Взам. інв. | Підпис і дата | Інв. ориг.

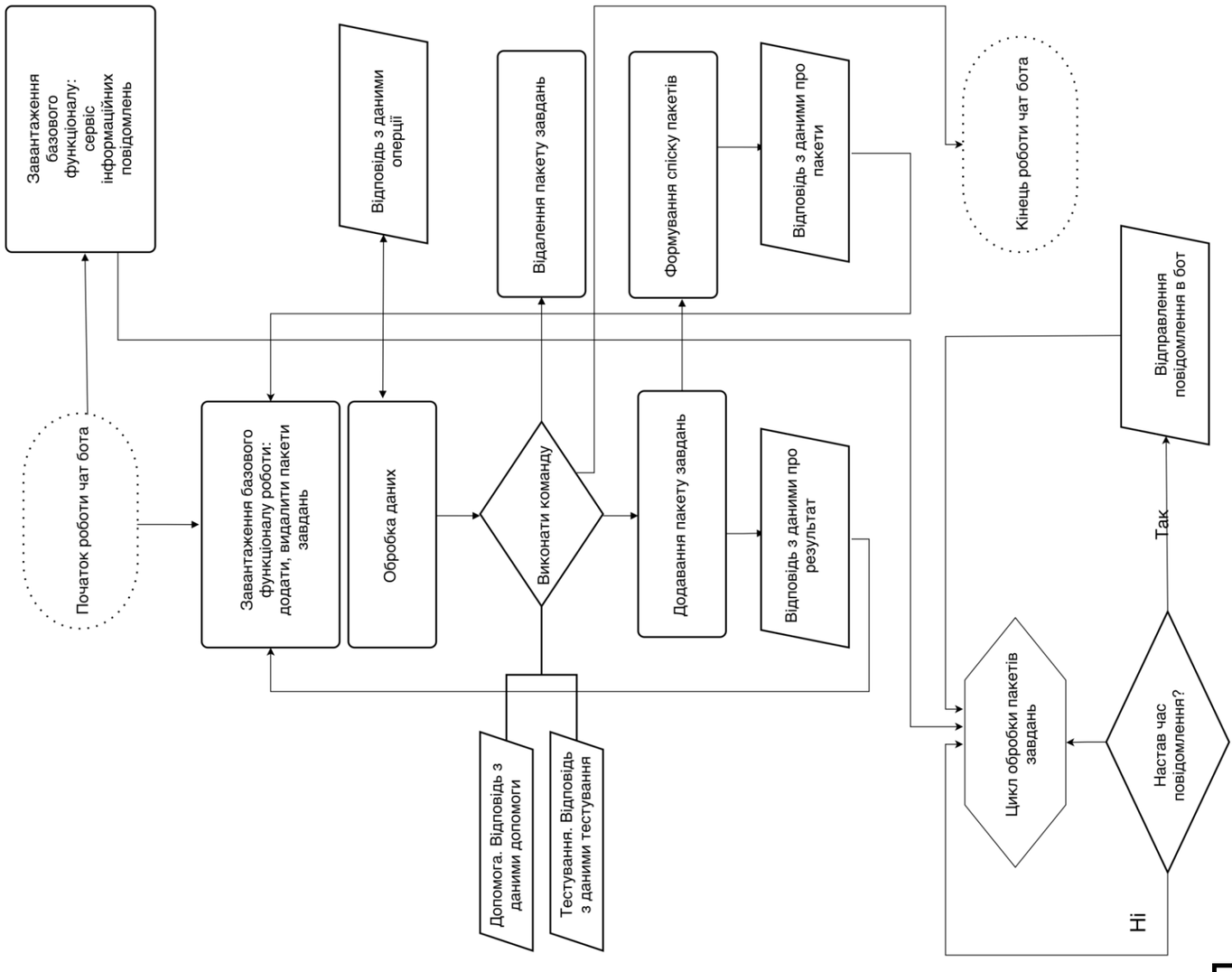
Зм.	Лист	докум	Підпис	Дата	ІК91.031БАК.005 Д2		
Розроб.		Остапченко Д.О			Схема навігації застосунку	Літ.	Маса	Мірило
Перев.		К				т		
						Лист 1	Листів 1	
Н. контр		Шинкевич М. К.				КПІ ім. Ігоря Сікорського ФІОТ, ІК-91		
Затв.		Ролік О. І.						

ІК91.031БАК.005 ДЗ



Підпис і дата | Інв. дубл. | Взам. інв. | Підпис і дата | Інв. ориг.

					ІК91.031БАК.005 ДЗ			
Зм.	Лист	докум	Підпис	Дата	Схема архітектури системи	Літ.	Маса	Мірило
Розроб.		Остапченко Д.О				т		
Перев.						Лист 1	Листів 1	
Н. контр		Шинкевич М. К.				КПІ ім. Ігоря Сікорського ФІОТ, ІК-91		
Затв.		Ролік О. І.						



Зм.	Лист	докум	Підпис	Дата
Розроб.		Остапченко Д.О		
Перев.				
Н. контр		Шинкевич М. К.		
Затв.		Роїк О. І.		

ІК91.031БАК.005 Д4

Блок-схема логіки бота

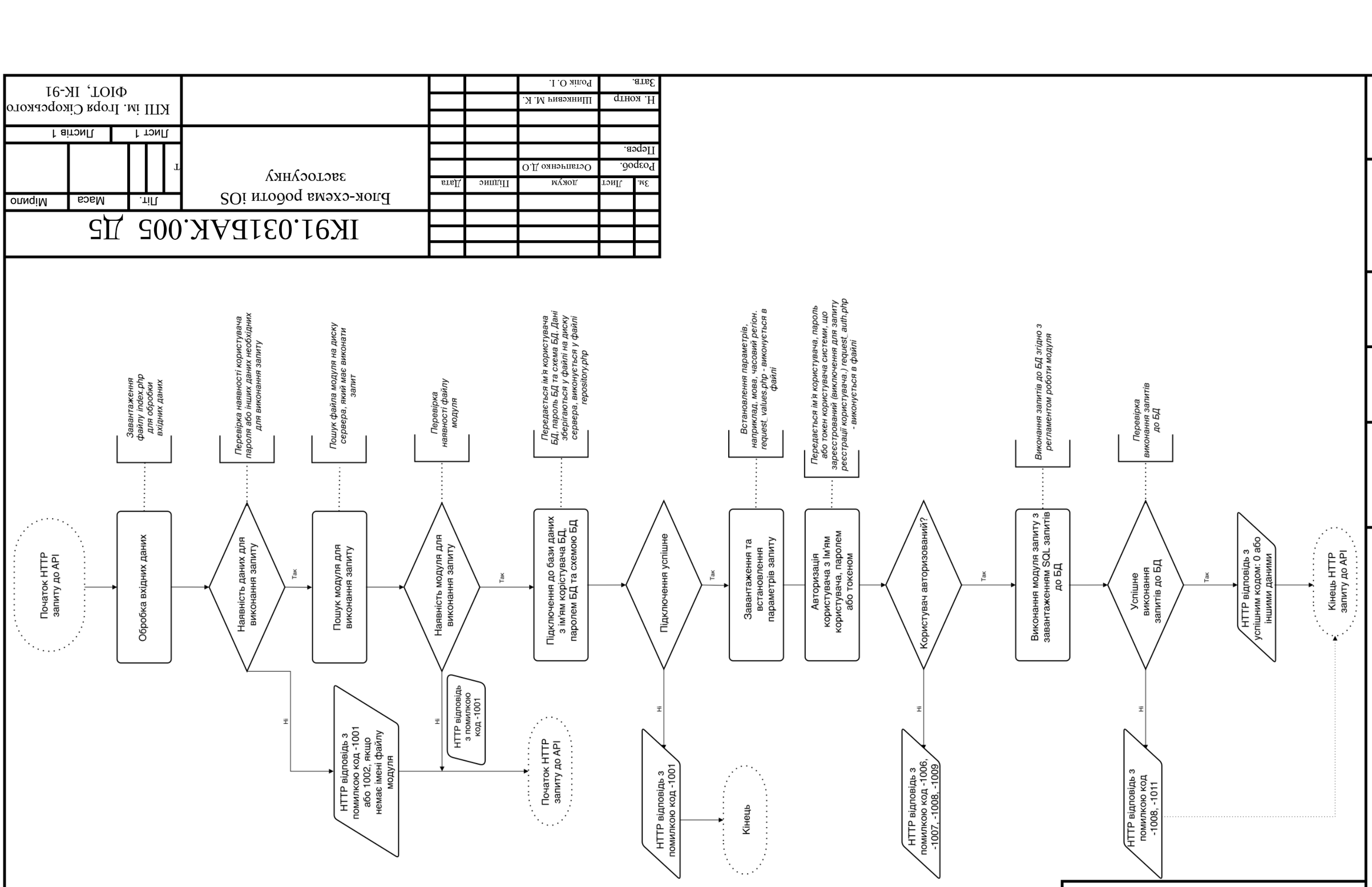
Літ.	Маса	Масштаб
Т		

Лист 1	Листів 1

КПІ ім. Ігоря Сікорського
ФІОТ, ІК-91

Інв. ориг. | Підпис і дата | Взам. інв. | Інв. | Підпис і дата

ІК91.031БАК.005 Д4



IK91.031БАК.005 Д5

Зм.	Лист	№ докум.	Підпис	Дата
Розроб.		Остапченко Д.О		
Перев.				
Н. контр.		Шинкевич М. К.		
Затв.		Рогік О. І.		

IK91.031БАК.005 Д5

Блок-схема роботи iOS застосунку

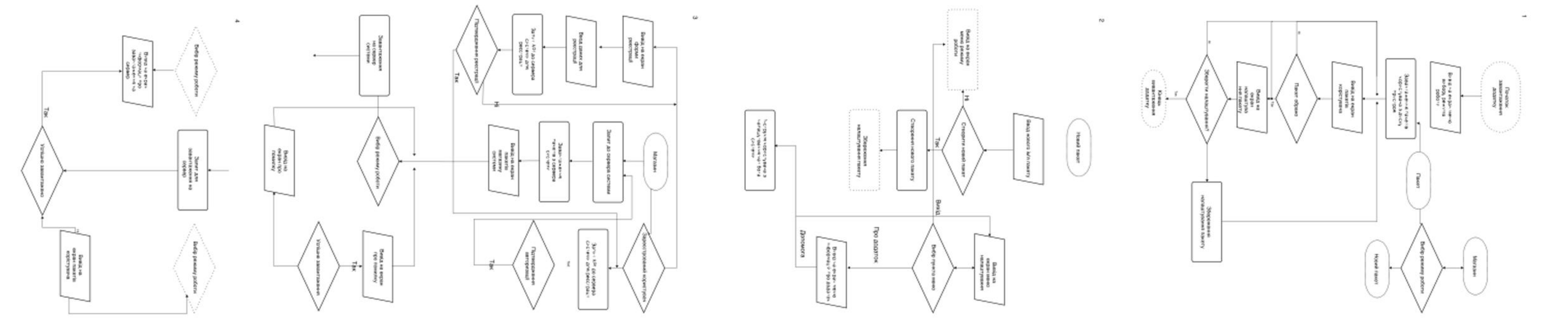
Літ.	Маса	Масштаб
Т		

Лист 1 Листів 1

КПІ ім. Ігоря Сікорського ФІОТ, ІК-91

Інв. № орит.	Підпис і дата	Взам. інв. №	Інв. № дубл.	Підпис і дата

ІК91.031БАК.005 Д6



Зм.	Лист	докум	Підпис	Дата
Розроб.		Остапченко Д.О		
Перев.				
Н. контр		Шинкевич М. К.		
Затв.		Ролік О. І.		

ІК91.031БАК.005 Д6

Блок-схема роботи додатку API серверу

Літ.	Маса	Мірило
т		

Лист 1 | Листів 1

КПІ ім. Ігоря Сікорського
ФІОТ, ІК-91

Підпис і дата | Інв. | Взам. інв. | Підпис і дата | Інв. ориг.