НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

ННК "Інститут прикладного системного аналізу"
(повна назва інституту/факультету)

Кафедра Системного проектування
(повна назва кафедри)

«До захисту допущено»

Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
"___"_____2016 р.

Дипломна робота

першого (бакалаврського) **рівня вищої освіти**
(першого (бакалаврського), другого (магістерського))

зі спеціальності 7.05010102, 8.05010102 Інформаційні технології проектування
7.05010103, 8.05010103 Системне проектування
(код та назва спеціальності)

на тему: Методи та інструментальні засоби побудови додатків для Apple Watch / iPhone. Аналіз фреймворків як засобів розробки додатків для Apple Watch / iPhone.

Виконав: студент 4 курсу, групи ДА-21
(шифр групи)

Онопрієнко Сергій Володимирович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник к.т.н., доцент Цурін О.П. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант Економіка проф. Семенченко Н.В _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент_____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Нормоконтроль ст. викладач Британов О.А.

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2016 року

Національний технічний університет України
«Київський політехнічний інститут»

Факультет (інститут) ННК “Інститут прикладного системного аналізу”
(повна назва)

Кафедра Системного проектування
(повна назва)

Рівень вищої освіти Перший(Бакалаврський)
(перший (бакалаврський), другий (магістерський) або спеціаліста)

Спеціальність 7.05010102, 8.05010102 Інформаційні технології проектування
7.05010103, 8.05010103 Системне проектування
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
«___»_____2016 р.

ЗАВДАННЯ
на дипломний проект (роботу) студенту
Онопрієнку Сергію Володимировичу
(прізвище, ім’я, по батькові)

1. Тема проекту (роботи) Методи та інструментальні засоби побудови додатків для Apple Watch / iPhone. Аналіз фреймворків як засобів розробки додатків для Apple Watch / iPhone.
керівник проекту (роботи) Цурін Олег Пилипович, к.т.н., доцент,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «__»_____ 2016р. №_____

2. Строк подання студентом проекту (роботи) «___» 2016 р.
3. Вихідні дані до проекту (роботи) _____

1. Мова програмування - swift
2. Системи контролю версій - git
3. Форма реалізації - додаток для iPhone та Apple Watch на базі MVC архітектури.
4. Можливість зчитувати пульс користувача за допомогою Apple Watch

4. Зміст розрахунково-пояснювальної записки (перелік завдань, які потрібно розробити)

1. Огляд apple watch, оперційні системи ios та watchos
2. Аналіз та порівняння мов програмування objective c та swift
3. Аналіз фреймворків та можливостей системи
4. Розробити програмну модель за допомогою мови програмування swift.

5. Виконати тестування роботи системи у комплексі.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслеників, плакатів тощо)

1. Схема роботы пари iPhone-Apple Watch – плакат.
2. Ілюстрація можливостей інтерфейсу Apple Watch – плакат.
3. Ілюстрація роботи створеного додатку – плакат.

6. Консультанти розділів проекту (роботи)*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний розділ	Семенченко Н.В. проф.док.ек.н		

7. Дата видачі завдання 01.02.2016

Календарний план

№ з/п	Назва етапів виконання дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Отримання завдання	01.02.2016	
2	Збір інформації	15.02.2016	
3	Вивчення варіантів реалізації та вибір варіанту для розробки	28.02.2016	
4	Порівняння інструментів програмування	10.03.2016	
5	Розробка плану тестування	15.03.2016	
6	Розробка алгоритму роботи та структури додатку	25.03.2016	
7	Розробка програмної моделі	25.04.2016	
8	Тестування додатку	30.04.2016	
9	Оформлення дипломної роботи	31.05.2016	
10	Отримання допуску до захисту та подача роботи в ДЕК	08.06.2016	

Студент _____ С.В. Онопрієнко
(підпис) (ініціали, прізвище)

Керівник проекту (роботи) _____ О.П. Цурін
(підпис) (ініціали, прізвище)

* Консультантом не може бути зазначено керівника дипломного проекту (роботи).

АНОТАЦІЯ

бакалаврської дипломної роботи Онопрієнка Сергія Володимировича

на тему «Методи та інструментальні засоби побудови додатків для Apple Watch / iPhone. Аналіз фреймворків як засобів розробки додатків для Apple Watch / iPhone»

Дана дипломна робота присвячена вивченню можливостей побудови додатків для розумних годинників Apple Watch та взаємодії їх з iPhohe. В рамках дипломної роботи був проведений аналіз існуючих методів:

1. Можливості системних фреймворків;
2. Інтеграція з БД;
3. Мова програмування.

Результатом роботи є створення програмного продукту для Apple Watch, що являє собою фітнес додаток для користувачів, які займаються бігом. Також було досліджено системи та засоби зчитування таких характеристик користувача, як пульс, кількість витрачених калорій та географічні координати.

Загальний обсяг роботи: 88 сторінок, 22 рисунки, 10 таблиць, 11 бібліографічних найменувань.

Ключові слова: Apple Watch, iPhone, WatchOS, iOS.

АННОТАЦИЯ

бакалаврской дипломной работы Оноприенко Сергея Владимировича

на тему «Методы и инструментальные средства построения приложений для Apple Watch / iPhone. Анализ фреймворков как средств разработки приложений для Apple Watch / iPhone »

Данная дипломная работа посвящена изучению возможностей построения приложений для умных часов Apple Watch и взаимодействия их с iPhohe. В рамках дипломной работы был проведен анализ существующих методов:

1. Возможности системных фреймворков;
2. Интеграция с БД;
3. Язык программирования.

Результатом работы является создание программного продукта для Apple Watch, который представляет собой фитнес приложение для пользователей, занимающихся бегом. Также были исследованы системы и средства считывания таких характеристик пользователя, как пульс, количество потраченных калорий и географические координаты.

Общий объем работы 88 страниц, 22 рисунка, 10 таблиц, 11 библиографических наименований.

Ключевые слова: Apple Watch, iPhone, WatchOS, iOS.

ANOOTATION

to the bachelor thesis of by Onopriienko Serhii

On a theme « Methods and tools for building applications for the Apple Watch /

iPhone. Analysis of the frameworks as a means to develop applications for the Apple Watch / iPhone" »

This thesis is devoted to the study of possibilities to build applications for smart watches Apple Watch, and their interaction with iPhone. analysis of existing methods was carried out in the framework of the thesis:

1. Possibilities of system frameworks;

2. Integration of the database;

3. Programming Language.

The work is to create software for Apple Watch, which is a fitness app for people who want to run. Also, the system and the means for reading such user characteristics have been studied as heart rate, calories burned, and geographic coordinates.

The total amount of work: 88 pages, 22 pictures, 10 tables, 11 bibliographical items.

Keywords: Apple Watch, iPhone, WatchOS, iOS.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ПЗ	програмне забезпечення
GUI	графічний інтерфейс користувача
IDE	Integrated development environment,
IT	Інформаційні технології
ОС	Операційна система
БД	База даних
API	Application Programming Interface
URL	Uniform Resource Locator
CT	Cocoa Touch
HK	Health Kit
MVC	Model-View-Controller

ВСТУП

Світ технологій розвивається безперервно і динамічно. З кожним роком з'являються нові концепції та їх реалізації, а існуючі розширюють свої функціональні характеристики і можливості. Не нова і мета цієї спіралі прогресу - зробити умови життя людини якомога комфортнішими і продуктивними. Однією з причин такого бурхливого зростання є широке поширення Internet і значне збільшення швидкості передачі даних. Як зазначають в Google, до 2008 року був «Інтернет людей», тепер настав «Інтернет речей», так як пристроїв, підключених до світової мережі стало більше ніж жителів планети. Завдяки цьому взаємодію їх з людиною стало можливо винести на зовсім інший рівень.

Розумний годинник – це нове слово у світі високотехнологічних гаджетів. Будучи як би продовженням Вашого смартфону вони можуть допомогти у самих несподіваних ситуаціях – скинути небажаний виклик, подивитися прогноз погоди, дізнатися температуру тіла та багато іншого. Дуже зручним є використання таких гаджетів у якості GPS-навігатора.

Думки про розширення функціональності годинника з’являлися у виробників електроніки дуже давно. Прототипом сучасних Smart-годинників був перший електронний годинник, такий як Pulsar 1972 року випуску. У 80-х -90-х роках функціонал таких пристроїв було розширено. Деякі пам’ятають популярний на території колишнього СРСР годинник Montana з секундомером та будильником. У 2000 році компанія IBM презентувала Linux Watch. Після цього починаються сміливі експерименти у цій області - Smart Personal Object Technology, Fossil Wrist PDA та багато іншого. З виходом операційної системи Андроїд - потужної, полнофункціональної та легковісної ОС, використання її на Smart Watch стало справою часу. Перший годинник з Андроїд на борту - WIMM One вийшов у 2011 році. А вже у 2012 Sony випускє у продаж Sony Smart Watch, які користуються популярністю й зараз. За останні два роки багато різних компаній спробували себе у розробці нових розумних годинників. Всесвітню відомість отримали – Pebble, Cokoo Watch, Samsung Galaxy Gear та

багато інших.

Мета даної роботи - провести аналіз методів та принципів побудови додатків для розумного годинника Apple Watch. Вивчити концепції та принципи побудови додатків, їх переваги та недоліки, а також проектування власного додатку на основі досліджених даних.

Одним з основних гравців у цій сфері є комранія Apple з її рішенням - Apple Watch. У першу чергу, Apple Watch це пристрій для контролю здоров'я та активності людини. Я думаю, хто любить носити годинник, не відмовилися б від них. Великим плюсом для зайнятих людей буде і можливість переглядати повідомлення. Годинник показує нові sms і пошту, повідомлення із соціальних мереж, оповіщення від новинних програм. Усе це виводиться на екран за мить після появи на iPhone.

З появою Apple Watch у багатьох програмістів виникає логічне бажання запрограмувати що-небудь для розумного годинника. А якщо вже програмувати, то краще щось корисне. Керуючись саме такою логікою, в даній роботі було спроектовано фітнес додаток для людей, що займаються бігом. Попри загальне враження, що Apple Watch пропонує широке поле для творчості, на жаль, поточні можливості розробки на емуляторі не збігаються із очікуваннями. Робити щось дійсно потрібне і функціональне для Apple Watch з поточною версією SDK недуже зручно. Тому для тих хто цікавиться даною розробкою рекомендовано мати справжній пристрій Apple Watch і бажано компюрер марки Apple.

На даний момент іде активна розробка і вдосконалення технології «Розумний» годинник. Незважаючи на це, процеси стандартизації та глобалізації почалися порівняно недавно й на даний час існує безліч однотипних рішень, але з різною реалізацією, зав'язаною на виробників та їх технології. Завдяки зацікавленості основних гравців ринку в розвитку системи «Розумний» годинник та інтеграції в неї своїх систем та сервісів, з'явилися зрушення у бік популяризації технології та спільна зацікавленість у ній як покупців, так і виробників.

1. ОГЛЯД APPLE WATCH, ОПЕРЦІЙНІ СИСТЕМИ IOS ТА WATCHOS

1.1. Apple Watch

Apple Watch — наручний годинник із додатковими функціями (розумний годинник) створений корпорацією Apple. Годинник оснащений всіма необхідними датчиками для спорту і здоров'я, які розташовані в нижній частині і взаємодіють безпосередньо з вашим тілом. Це робить Apple Watch незамінним помічником для спортсменів, який позбавляє сенсу існування цілу галузь фітнес-браслетів. Годинники підтримують роботу з цифровим асистентом Siri і можуть взаємодіяти з iPhone через 802.11b / g і Bluetooth 4.0. До речі, підтримуються всі моделі смартфона, починаючи з iPhone 5. Дисплей годинника є сенсорним і розпізнає не лише дотики, але і так звані «посилені торкання» (Force Touch), яким присвоєно різні завдання. Бездротові модулі Apple Watch також дозволяють здійснювати транзакції через нову платіжну систему Apple Pay і використовувати гаджет для дистанційного керування Apple TV.

Apple Watch має датчик натискання на екран і новий жест - посилений дотик. Крім цього, користувач годинника може налаштувати циферблат. Свайпи по циферблату дозволяють швидко отримувати доступ до різних функцій. Доступ до додатків реалізований неймовірно просто - десятки «кульок» чекають своєї черги прямо на екрані.

1.1.1. Історія технології Apple Watch

Коли в Apple лише почали розробку смарт-годин, керівники компанії заговорили про найсучаснішому гаджет для контролю над станом здоров'я. Він повинен був заміряти тиск, серцевий ритм, рівень стресу і робити ще багато чого корисного. Це підтверджували і ті, хто, так або інакше, мав відношення до даного проекту.

Розумні годинник Apple Watch були представлені 9 вересня 2014 року,

вони здатні повноцінно взаємодіяти з iPhone і iPad, а також облаждают широким набором функцій, пов'язаних з охороною здоров'я - вимірювати пульс, кількість пройдених за день кроків і так далі. У продаж надійшли 24 квітня 2015 року у 9 країнах (США, Канада, Великобританія, Австралія, Франція, Німеччина, Гонконг, Китай, Японія). Але до 18 червня 2015 року годинники не були доступні у роздрібній торгівлі в магазинах, їх можна було замовити тільки за попереднім замовленням, зробленим через інтернет-магазин. Ціна становить від 349 дол. США.

1.1.2. Принцип роботи

Дисплей Apple Watch вимкнений, поки ви на нього не дивіться. Як тільки ви повернете руку до очей, екран активується (на відміну від деяких моделей на базі Android, апарат Apple відразу ж знову піде в режим очікування, як тільки ви відвернеться руку з годинником). Якщо потрібно негайно «згасити» дисплей, досить накрити його долонею.

Набір SMS - за допомогою голосової диктування (російський розпізнається), нормальна клавіатура не вміщується на маленькому екранчику. Можна використовувати готовий шаблон.

Вбудований фітнес-трекер (аналог Fitbit) - вимір пульсу, підрахунок кроків і калорій. Taptic Engine дозволяє відправити іншим власникам годин від Apple набір «торкань» або своє серцебиття.

Помічник Siri (викликається натисканням і утриманням коліщатка).

Поштовий клієнт з обмеженою функціональністю (текст деяких листів він показує повністю, а при спробі прочитати інші радить відкрити iPhone і запустити відповідну програму там). Відповісти на лист за допомогою годинника можна, можна лише видалити його, відзначити як непрочитане або встановити прапорець. Браузера в Apple Watch немає, номеронабирателя теж.

В Apple Watch можна завантажувати до 2 ГБ музики. Apple Watch не призначені для ігрового використання. У січні 2015 року низка компаній, включаючи TapSense і InMarket, оголосили про плани по запуску на дисплеях

годин гіперлокальной реклами.

При вмиканні годинник відразу просять приєднатися до свого «родича», телефону iPhone - просять піднести камеру смартфона до зображення на екрані Apple Watch. Потім на пристрій переносяться всі додатки, встановлені на iPhone і адаптовані для годин. Підключення двох пристроїв йде через Bluetooth. Крім того, Apple Watch можуть обмінюватися даними за допомогою Wi-Fi, якщо обидва девайса підключені до однієї бездротової мережі.

1.1.3. Основні характеристики системи

Apple Watch працюють на операційній системі Watch OS. Годинники мають поворотне коліщатко для прокрутки або збільшення. Натискання на нього дозволяє повернутися до домашнього екрану. Дисплей має розмір 38x38 або 42×42 мм, і здатний розрізняти натискання і дотик. Апарат не має роз'ємів,

Рисунок 1.1 - Моделі годинника Apple Watch

підзарядка батареї відбувається за допомогою індуктивного адаптера з магнітною фіксацією. На нижній стороні годинників можуть бути розташовані

світлодіоди і фотодіоди для вимірювання пульсу. Також є в наявності інтерфейс NFC, який дозволить робити безконтактну оплату через систему Apple Pay. Девайс доступний у трьох «колекціях»: Apple Watch Sport, Apple Watch та Apple Watch Edition (виконаний із жовтого і рожевого золота).

Великим плюсом для зайнятих людей буде і можливість переглядати повідомлення. Годинник показує нові sms і пошту, повідомлення із соціальних мереж, оповіщення від новинних програм. Усе це виводиться на екран за мить після появи на iPhone. Очевидно, що обробляти пошту і повідомлення значно швидше на екрані смартфона. Але часом навалюється так багато всього, що діставати щоразу iPhone просто незручно. Лише один погляд на годинник дає можливість зрозуміти: смартфон завібрував у кишені через важливий електронний лист чи знову через коментар на Facebook, вивчення якого можна відкласти.

1.2. Операційна система Watch OS2

WatchOS — операційна система від корпорації Apple, що була спеціально розроблена для Apple Watch. Операційна система підтримує сторонні додатки, має функцію Siri, мапи, календар, нагадування, блокнот, і власне годинник. WatchOS розроблена виключно під апарат Apple Watch і не пристосована для роботи на будь-якому іншому пристрої. Вперше випущена 24 квітня 2015 року. watchOS підтримує застосунки, що розроблені за допомогою WatchKit. Головне меню системи виконане у вигляді Carousel, що надає зручний доступ до любого застосунку.

Додатки для годиника бувають трьох типів: Watch App, Glance Interface і Notifications. Найбільш функціональним є перший. Glance Interface і Notifications призначені для повідомлення користувачів про які-небудь події, Glance дозволяє створити свій кастомний вид оповіщення, а Notifications працює з локальними і push повідомленнями, відображає їх користувачеві і дозволяє додати кілька кнопок для дій.

1.2.1. Загальна інформація про Watch OS2

Ваш додаток на годинник та розширення на iPhone WatchKit працюють в тандемі для повноцінної роботи інтерфейсів. Коли ваш додаток на годиннику, watchOS завантажує призначений для користувача інтерфейс з програми Watch на мобільному додатку. Він також запускає свій додатковий WatchKit, який управляє вмістом інтерфейсу. Для додатку Watch і здійсненних повідомлень, розширення WatchKit обробляє взаємодії користувача з інтерфейсом. Рисунок 1.2.1 показує взаємозв'язок між додатком Watch, розширенням WatchKit і додатком IOS.

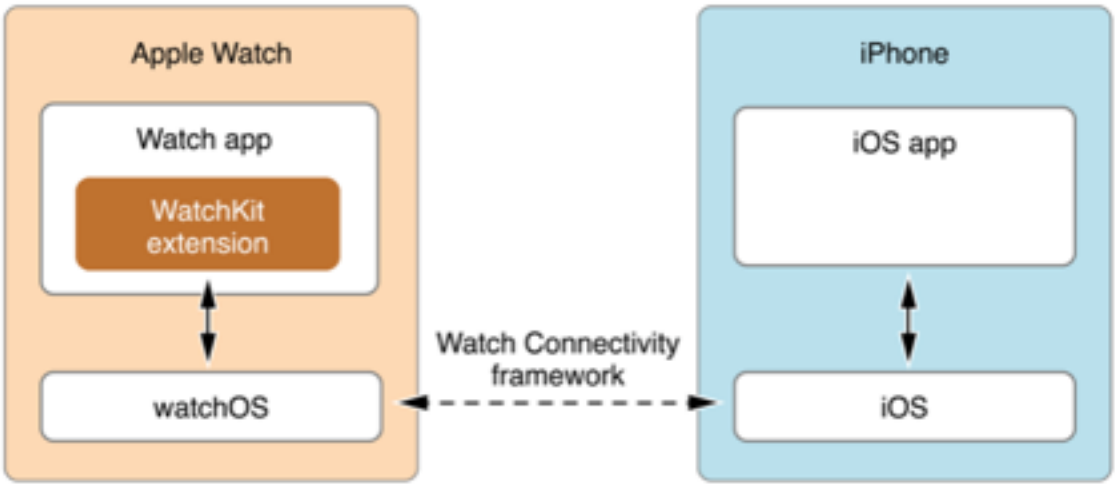

Рисунок 1.2 - Взаємодія додатків iOS та WatchOS

Як додаток IOS, програма “Годинник” складається з одної або більше сцен, де кожна сцена представляє повний екран контенту. Кожна сцена є результатом керування одного об'єкта контролера інтерфейсу в розширенні WatchKit. Контролер інтерфейсу є екземпляром WKInterfaceController класу. Контролер інтерфейсу в watchOS служить для тієї ж мети, що і контролер уявлення в iOS. Він управляє контентом на екрані і реагує на дії користувача з цим вмістом. На відміну від контролера View, контролер інтерфейсу не керує фактичним видом інтерфейсу. Ці View управляються для вас watchOS.

Головний інтерфейс програми, як правило, містить кілька сцен, на кожній з них відображаються різні типи інформації. Тільки одна сцена

відображається на екрані, додаток представляє нові сцени у відповідь на дії користувача. Наігаційний контролер визначає яким чином представлені сцени. Додатки можуть відображати сцени модально. Для отримання інформації про те , як представити нові сцени, см навігаційний інтерфейс .

1.2.2. Архітектурні аспекти

Для того щоб запустити додаток, необхідно обрати його з поміж інших додатків. Запуск додатку призводе до початку роботи життєвого циклу додатку. Є можливість запуску додатку у момент натходження сповіщення чи сигналу з годинника. Під час запуску watchOS автоматичо грузить storyBoard. Після того як сцена завантажена, watchOS дає запит на створення відповідного контролеру взаємодії, яким ми користуємось для підготовки сцени до роботи з користувачем. Розглянемо більш детально процес запуску додатку на інформаційному рисунку, який був опублікований компанію Apple на офіційній сторінці, присвяченій детальному опису роботи додатку на прикладному рівні.

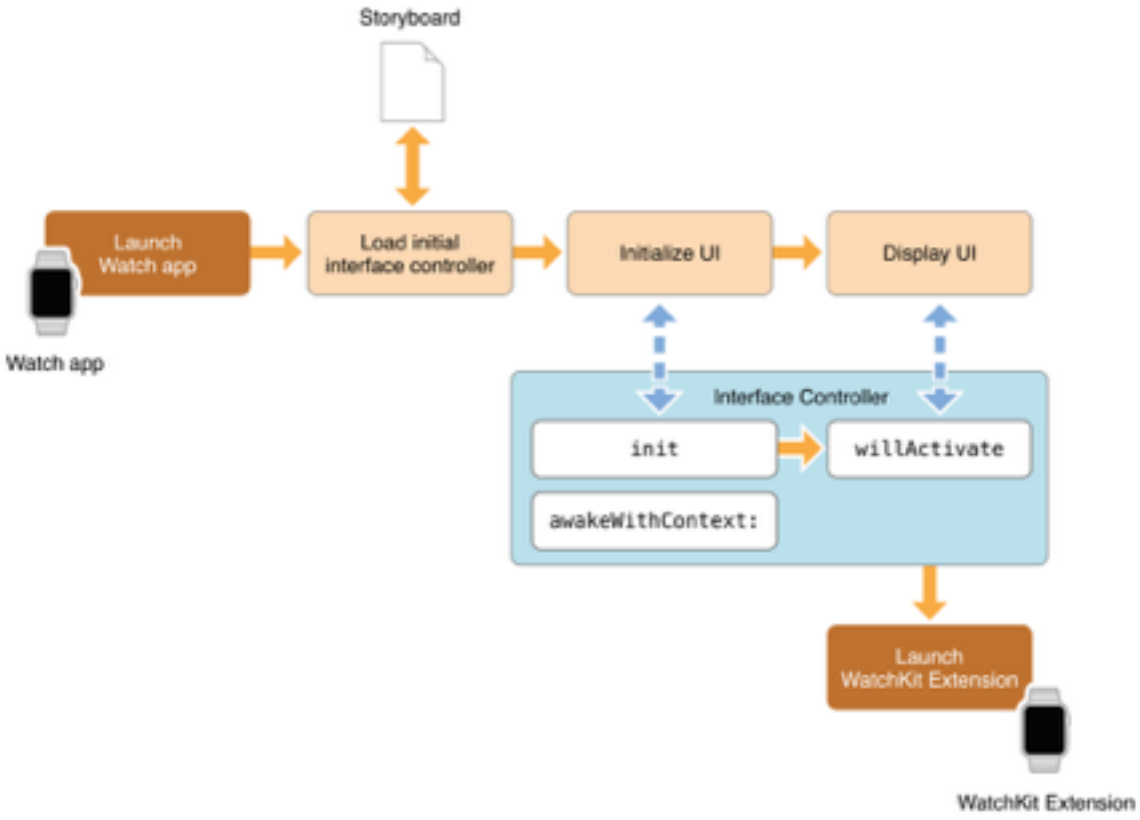

Рисунок 1.3 - Ініціалізація додатку

На рисунку 1.2.3 можна побачити що після запуску додатку відбуважться завантаження інтерфейс контролеру. А вже після, іде ініціалізація UI і безпосередньо відображення сцени на дисплеї Apple Watch.

Перед тим як ініціалізувати UI, контролер повинен бути активований, для цього іде виклик awakeWithContent. Використовуйте willActive тільки для відображення новостворених або оновленної інформаціЇ

Ми розглянули основні єтапи життєвого циклу. Також зверніть увагу на стани додатку. В залежності від вихідних і вхідних умов ці стани відповідно змінюються.

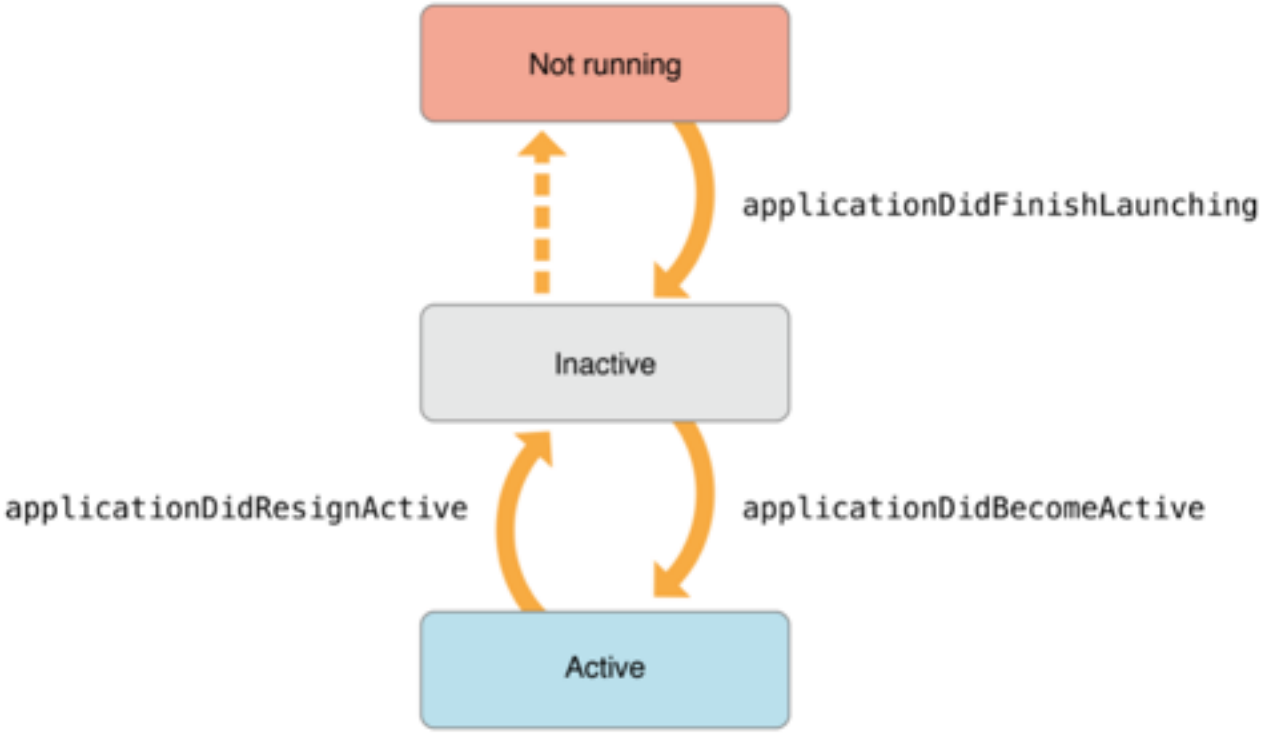

Рисунок 1.4 - Демонстрація станів

1.3. Операційна система iOS

iOS (відома як iPhone OS до червня 2010 року) — це власницька мобільна операційна система від Apple. Розроблена спочатку для iPhone, вона стала операційною системою також для iPod Touch, iPad і Apple TV. Apple не дозволяє роботу ОС на мобільних телефонах інших фірм.

Рисунок 1.5 - Головний екран iOS

iOS є похідною від OS X, отже, є за своєю природою Unix-подібною операційною системою.

Користувацький інтерфейс iOS заснований на концепції прямої маніпуляції з використанням жестів Multi-Touch. Елементи інтерфейсу управління складаються з повзунків, перемикачів і кнопок. Він призначений для безпосереднього контакту користувача з екраном пристрою. Внутрішній акселерометр використовуються деякими програмами для реагування на струшування пристрою, яке є також загальною командою скасування, або обертати пристрій у трьох вимірах, що є загальною командою перемикання між книжковим та альбомним режимами.

Станом на 31 травня 2011 року інтернет-магазин App Store містить понад 500 тисяч застосунків для iOS, які були завантажені понад 15 мільярдів

разів.

Станом на червень 2016 року, iOS становив 23,4 % ринку операційних систем для смартфонів, другий після Android.

1.3.1.Історія

В якості операційної системи iOS була представлена з iPhone на Macworld Conference & Expo 9 січня 2007 року і випущена в червні того ж року. Спершу, Apple не вказувала її ім'я, просто заявивши, що "iPhone використовує OS X". Спочатку, сторонні програми не підтримувалися.

Рисунок 1.6 - Вигляд першої iPhoneOS 1.0

Стів Джобс заявив, що розробники можуть створювати веб-програми, що „будуть вести себе, як рідні програми на iPhone“. 17 жовтня 2007 року Apple оголосила, що рідний SDK знаходиться в стадії розробки, і що вони планують поставити його „в руки розробників у лютому“. 6 березня 2008 року Apple випустила першу бета-версію, а також нове ім'я для операційної системи: iPhone OS. Продажі мобільних пристроїв Apple викликали інтерес до SDK.

Apple також продала більше одного мільйона iPhones під час курортного сезону 2007. У червні 2010 року, Apple перейменувала iPhone OS на iOS. Назва iOS користувалася компанією Cisco вже більше десяти років на маршрутизаторах Cisco. Для того, щоб уникнути будь-якого потенційного позову, Apple ліцензувала торгову марку iOS у Cisco.

1.3.2.Основні можливості iOS

- Фотографії — пошук по місцю і часу, а також нові можливості редагування фотографій.

- Повідомлення — можливість відправляти аудіозаписи і карту з місцезнаходженням в діалог. Тепер можна швидше переслати щойно зняті відеозаписи та фото, а також встановлювати функцію «Не турбувати» на потрібні діалоги.

- Quicktype — передбачає можливі за змістом слова, засновані на розмові, під час друкування пропозицій.

- Family Sharing — можна позначати до шести контактів як членів сім'ї і швидко ділитися з ними фотографіями, покупками додатків і музики, місцеположенням та іншим.

- iCloud Drive — можливість зберігати в хмарі будь-які види файлів з подальшим їх редагуванням на різних пристроях.

- HealthKit — організація відомостей про здоров'я в одному додатку.

- Spotlight — пошук став більш глобальним і тепер можна шукати різну інформацію і за межами телефону / планшета.

- Віджети- тепер в Центр сповіщеннях можна встановлювати віджети від сторонніх розробників.

- Сторонні клавіатури — вперше компанія Apple дозволила стороннім розробникам створювати альтернативні клавіатури, які зможуть замінити стандартну клавіатуру.

1.4. Висновки

Підсумувати враження про Apple Watch найпростіше наступним тижнем, який було проведено вже без нього: класичний механічний годинник виявилися нічим не гіршим за сучасний електронний варіант і так само точно показував час. Його не потрібно було заряджати, а вся метрика за результатами фізичної активності збирається на iPhone. Додатки в цьому варіанті виявилися марними без смартфона, а отже, той же iPhone завжди в руці.

Цілком можливо, що ситуація різко зміниться до другого покоління Apple Watch. Приклад iPad і iPhone доводить це: і перший смартфон, і перший планшет вийшли досить слабкими. А ось про успіх другого покоління можна й не говорити: багато хто використовує iPad 2, випущений 2011 року, і сьогодні. Є підозра, що з годинником буде та ж історія. Вони стануть дешевшими, а отже, і доступнішими. Матимуть потужнішу батарею. Старі проблеми усунуть, а замість них в Apple Watch з'явиться більше можливостей.

Поки що Apple Watch більше нагадує модний аксесуар, який до того ж має функції спортивного браслета і вміє показувати сповіщення. Чи корисний цей ґаджет? Показувати час вміє і годинник.

Попри загальне враження, що Apple Watch пропонує широке поле для творчості, на жаль, поточні можливості розробки на емуляторі не збігаються із очікуваннями. Робити щось дійсно потрібне і функціональне для Apple Watch з поточною версією SDK недуже зручно. Тому для тих хто цікавиться даною розробкою рекомендовано мати справжній пристрій Apple Watch і бажано компюрер марки Apple.

2. АНАЛІЗ ТА ПОРІВНЯННЯ МОВ ПРОГРАМУВАННЯ OBJECTIVE C ТА SWIFT

2.1. Objective C

Рисунок 2.1 - Логотип мови Objective C

Objective-C — рефлективна, високорівнева об'єктно-орієнтована мова програмування загального призначення, розроблена у вигляді набору розширень стандартної C.

Розроблена компанією Apple, використовується в основному у Mac OS X та GNUStep — середовищах, розроблених на основі стандарту OpenStep, та Cocoa — бібліотеки компонентів для розробки програм. Програму на Objective-C що не використовує цих бібліотек можна скомпілювати для будь-якої платформи, яку підтримує gcc компілятор з підтримкою Objective-C.

Objective-C є розширенням C і тому будь-яку програму на C можна скомпілювати компілятором Objective-C.

ООП в Objective-C включає інтерфейси, класи, категорії. Реалізовано одиничне, невіртуальне спадкування. Немає єдиного базового класу для всіх об'єктів. Всі методи в класі — віртуальні. Категорія — парадигма яка дозволяє

описувати інтерфейс з методами які «необов'язково» імплементувати.

Синтакс Objective-C породжений одночасно від C та Smalltalk. Від останньої взято основний семантичний конструкт мови — замість виклику методу об'єктові надсилається повідомлення. Наприклад, якщо клас об'єкта obj імплементує метод doJob то говориться що об'єкт відкликається на повідомлення doJob. Щоб надіслати повідомлення doJob цьому об'єктові потрібно написати:

[obj doJob];

Такий механізм дозволяє надсилати повідомлення навіть до тих об'єктів які не підтримують їх обробки. Такий підхід відрізняється від тих що використовуються в статично типізованих мовах C++ чи Java.

2.2. Swift

Рисунок 2.2 - Логотип мови Swift

Swift — багатопарадигмова компільована мова програмування, розроблена компанією Apple для того, щоб співіснувати з Objective C і бути стійкішою до помилкового коду. Swift була представлена на конференції розробників WWDC 2014. Мова побудована з LLVM компілятором, включеного у Xcode 6 beta. Безкоштовний посібник мови програмування Swift доступний для завантаження у магазині iBooks.

Компілятор Swift побудований з використанням технологій вільного

проекту LLVM. Swift успадковує найкращі елементи мов C і Objective-C, тому синтаксис звичний для знайомих з ними розробників, але водночас відрізняється використанням засобів автоматичного розподілу пам'яті і контролю переповнення змінних і масивів, що значно збільшує надійність і безпеку коду.

При цьому Swift-програми компілюються у машинний код, що дозволяє забезпечити високу швидкодію. За заявою Apple, код Swift виконується в 1.3 рази швидше коду на Objective-C. Замість збирача сміття Objective-C в Swift використовуються засоби підрахунку посилань на об'єкти, а також надані у LLVM оптимізації, такі як автовекторизація.

Мова також пропонує безліч сучасних методів програмування, таких як замикання, узагальнене програмування, лямбда-вирази, кортежі і словникові типи, швидкі операції над колекціями, елементи функційного програмування. Основним застосуванням Swift є розробка користувацьких застосунків для MacOS X і Apple iOS з використанням тулкіта Cocoa і Cocoa Touch. При цьому Swift надає об'єктну модель, сумісну з Objective-C. Сирцевий код мовою Swift може змішуватися з кодом на C і Objective-C в одному проекті.

Swift щільно інтегрований у власницьке середовище розробки Xcode і не може бути використаний відособлено на платформах, відмінних від OS X.

Окремо варто відзначити, що Swift від компанії Apple не варто плутати з досить давно розроблюваною скриптовою мовою Swift, націленої на багатонитеве програмування і поставленого під вільною ліцензією Apache.

2.3. Порівняльна характеристика Objective C та Swift

Мови програмування не вмирають швидко, а студії розробники, які чіпляються за згасаючі парадигми, вмирають. Якщо ви розробляєте програми для мобільних пристроїв і не вивчали Swift, то знайте: Swift не тільки витісняє Objective-C, коли мова йде про розробки додатків під OSX, iOS і WatchOS, але і в недалекому майбутньому замінить C для внутрішнього програмування, для Apple платформ .

Завдяки кільком ключовим особливостям, Swift має потенціал стати єдиною мовою програмування, для створення захоплюючих, гнучких, споживацьки-оріентрірованних додатків або програм, на багато років.

У Apple великі надії на Swift. Компанія настільки ефективно оптимізувала компілятор і саму мову в принципі, що багато можливостей ще тільки потрібно розкрити. Можна сказати, що Swift «призначений для зльоту від “hello, world” до цілої операційної системи.

Таблиця 2.1 - Порівняння мов програмування

	Swift	**Objective-C**
Спеціальні символи	Swift не побудований на C, то він може об'єднати всі ключові слова і видалити численні символи @ перед кожним Objective-C типом або перед пов'язаною з об'єктом ключовим словом.	Для диференціації ключових слів і типів від C типів, Objective-C вводить нові ключові слова, використовуючи символ @.
Наявність крапки з комою в кінці строки	Swift переглядає загальноприйняті умови успадкування. Не повинні ставиити крапку з комою.	Обов’язково прийнятий стандарт.
Дужки після if/else умови	Ні	Так
Читабельність коду	Читабельність коду в Swift полегшує роботу для програмістів JavaScript, Java, Python, C #, C ++	Складна дот сприйняття
Вимога двох файлів .h .m	Свіфт поєднує в собі заголовок Objective-C (.h) і файли реалізації (.m) в одному файлі коду (.swift)	Обов’язкова. У Objective-C ви повинні вручну синхронізувати імена методів і коментарі між файлами.

Таблиця 2.1 (закінчення) - Порівняння мов програмування

	Swift	**Objective-C**
Витрата часу на написання коду	Низька	Велика
Спосіб обробки nil (NULL)	Опціональні типи дають можливість існування в Swift коді nil опционального значення, що говорить про можливість створення помилки компілятора при написанні поганого коду.	У Objective-C нічого не трапиться якщо ви спробуєте викликати метод зі змінною покажчика nil (неініціалізованих). Вираження рядок коду стає нездійсненними і може здатися, що вираз не впаде, але насправді буде повно багів.
Визначення типу	Так, у випадку опціональних типів	Визначення є обов’язковим, у випадку NULL можуливе вінікнення помилок
ARC - Автоматичний підрахунок ссилок	ARC обробляє всі управління пам'яттю під час компіляції, і витрати, які пішли б на управління пам'яттю, тепер можуть бути сфокусовані на ключевій логіці	У Objective-C, ARC підтримується всередині Cocoa API і об'єктно-орієнтованого коду; але він не доступний для C коду і API
Робота ARC рівні	На процесуально і на об'єктно-орієнтованому коді	На рівні Cocoa API
Робота зі строками	Swift приймає особливості сучасної мови, наприклад додавання 2-х рядків разом з оператором «+»	У Objective-C, працюючи з текстовими рядками, вам доводиться бути багатослівним, і щоб об'єднати дві частини інформації, буде потрібно безліч кроків.
Система типів	Визначення типу автоматично	Строготипізована мова

2.4. Порівняння швидкості роботи

Таблиця 2.2 - Перемішування 1000000 інтових обєктів з взяттям AnyObject

Версія	**Objective-C**	**Swift**	**Device (iOS Version)**
Swift 1.1	0.296 seconds 1% STDEV	1.616 seconds 2% STDEV	iPhone 6 (8.3)
Swift 1.2	0.262 seconds 1% STDEV	0.523 seconds 1% STDEV	iPhone 6 (8.3)
Swift 2.0	0.271 seconds 2% STDEV	0.285 seconds 2% STDEV	iPhone 6 (8.3)
Swift 2.1	0.305 seconds 1% STDEV	0.336 seconds 5% STDEV	iPhone 6 (9.2)
Swift 2.2 Snapshot	0.256 seconds 1% STDEV	0.284 seconds 7% STDEV	iPhone 6 (9.2)

Таблиця 2.3 - Цикл видалення 1000000 елементів з масиву Int

Версія	Objective-C	Bridgeable Swift	Swift	Device (iOS Version)
Swift 1.1	0.010 seconds 38% STDEV	-	0.003 seconds 49% STDEV	iPhone 6 (8.3)
Swift 1.2	0.005 seconds 10% STDEV	-	0.001 seconds 20% STDEV	iPhone 6 (8.3)
Swift 2.0	0.005 seconds 23% STDEV	-	0.001 seconds 10% STDEV	iPhone 6 (8.3)
Swift 2.1	0.005 seconds 35% STDEV	-	0.001 seconds 9% STDEV	iPhone 6 (9.2)
Swift 2.2 Snapshot	0.005 seconds 24% STDEV	-	0.000 seconds 106% STDEV	iPhone 6 (9.2)

Розглянемо тепер стандартні операції додавання, видалення та вставку елементів у масив

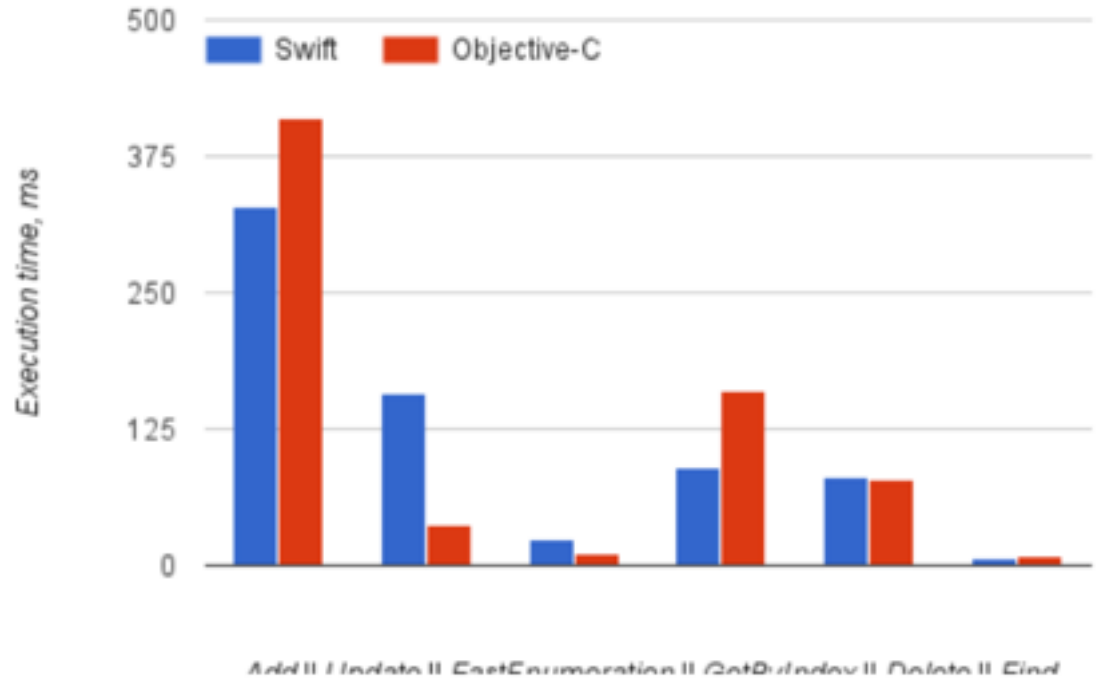

Рисунок 2.3 - Порівняльний графік стандартних операцій. Array/NSArray

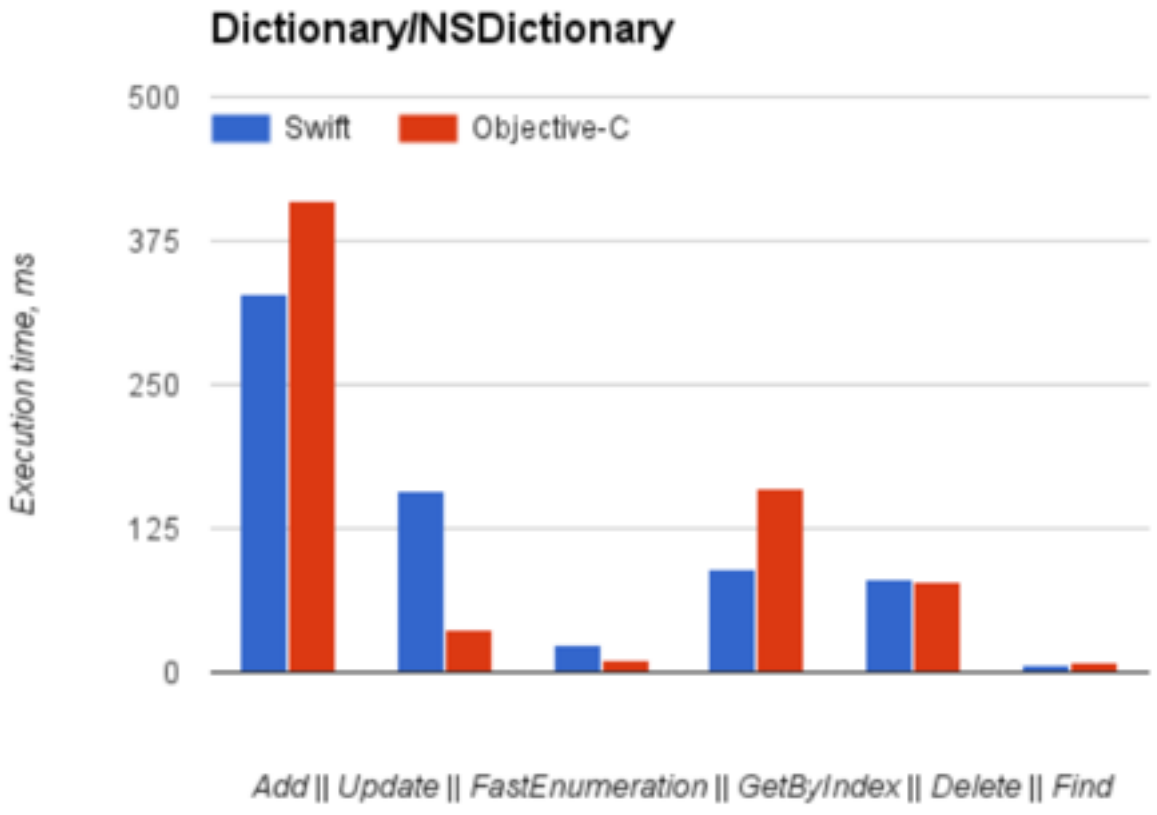

Рисунок 2.4 - Порівняльний графік стандартних операцій. Set/NSSet

- Операції додавання, оновлення та читання в масиві Swift набагато швидше, ніж в NSArray.

- Свіфт не швидший, ніж Objective-C.

- Існує аномалія масиву "видалити індексні" Результати: Видалення даних в Swift в 10 разів повільніше, ніж в Objective-C.

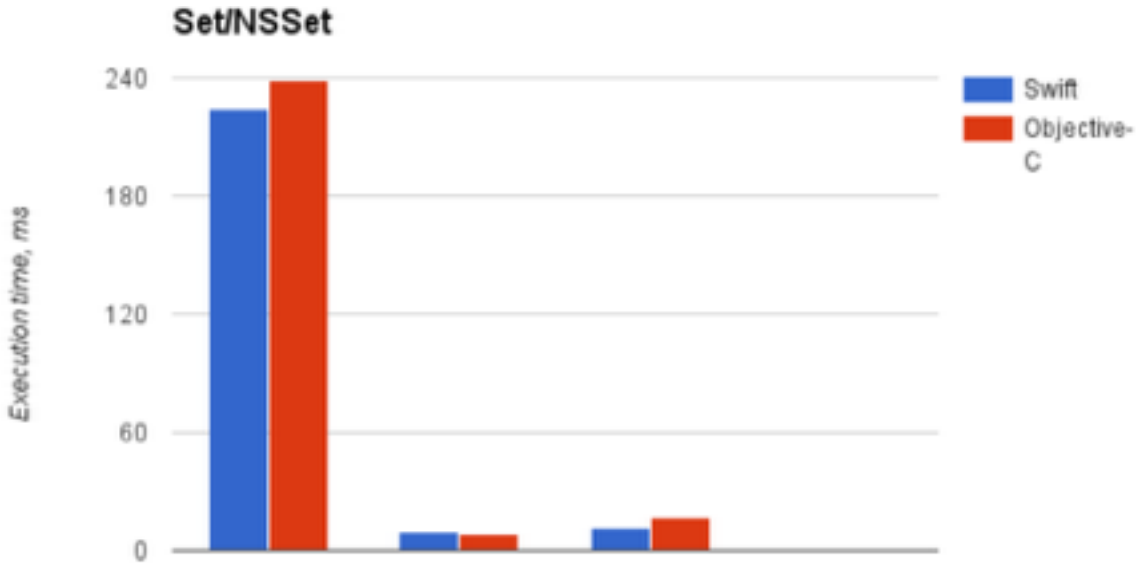

Рисунок 2.5 - Порівняльний графік стандартних операцій. Dictionary/ NSDictionary

Об'єкт NSSet управляє незмінним набором різних об'єктів, тобто після створення набору, ви не можете додавати, видаляти або замінювати об'єкти. Можна, однак, змінити окремі об'єкти самі (якщо вони підтримують зміни). Мінливість колекції не впливає на мінливість об'єктів в колекції. Ви повинні використовувати незмінний набір якщо він імененяются в рідкісних випадках або зміни носять масовий характер.

Ці показники припускають адекватну реалізацію для хеш-методів певних для об'єктів. При поганій реалізації хеш-функцій, доступ і редагування виконується за лінійний час.

	Swift	Objective-C
Class File	One file	Two files (example.h,example.m)
Declaring an int variable	var highScore: Int var playerName: String	int highScore; NSString *playerName;
Assigning to a variable	highScore = 1000 playerName = "Joe Chan"	highScore = 1000; playerName = @"Joe Chan";
Declaring and assigning constants	let skyColor = "Blue" let daysInYear = 365	NSString *const skyColor = @"Blue"; int const daysInYear = 365;
Defining a class	class CustomClass { }	**Header file (.h)** @interface CustomClass : NSObject @end **Implementation file (.m)** @implementation CustomClass @end
Creating a new class instance	var instance = CustomClass()	CustomClass *instance = [[CustomClass alloc] init]; CustomClass *secondInstance = [CustomClass new];

Method	func getOdometerReading() -> Int {
	return 50000
	}
		Header file (.h)
		- (int) getOdometerReading;
		
		Implementation file (.m)
		- (int) getOdometerReading
		{
		return 50000;
		}

Calling a method	var myCar = SpecificCar() myCar.getOdometerReading()	SpecificCar *myCar = [[SpecificCar alloc] init]; [myCar getOdometerReading];
Declaring a method with multiple parameters	func changeEngineOil(oil:Int, transmissionFluid:Int) { }	- (void)changeEngineOil:(int)oil transmissionFluid:(int)fluid { }
Calling a method with multiple parameters	myCar.changeEngineOil(10, transmissionFluid:10)	[myCar changeEngineOil:10 transmissionFluid:10];

Declaring a property	class SpecificCar {
	var transmission: String = "Automatic"
	var numberOfSeats: Int
	
	init() {
	self.numberOfSeats = 4
	super.init()
	}
	}
		Header file (.h)
		@interface SpecificCar ()
		@property NSString *transmission;
		@property int numberOfSeats;
		@end
		
		Implementation file (.m)
		@implementation SpecificCar
		- (id)init {
		self = [super init];
		if (self) {
		self.transmission = @"Automatic";
		self.numberOfSeats = 4;
		}
		return self;
		}
		@end

Рисунок 2.6 - Порівняння сигнарури методів swift та Objective C.

Ще однією з особливостей мови є те, що він message-oriented. Це означає, що в ньому виклики методу інтерпретуються не як виклик функції (хоча до цього зазвичай все зводиться), а саме як посилка повідомлення (з ім'ям і аргументами) об'єкту, подібно до того, як це відбувається в Smalltalk-е. Такий підхід дає цілий ряд плюсів - так будь-якого об'єкта можна послати будь-яке повідомлення. Об'єкт може замість обробки повідомлення просто переслати його іншому об'єкту для обробки (так зване делегування), зокрема саме так можна легко реалізувати розподілені об’єкти.

2.5. Висновки

Очевидно, що впровадження Swift помітно позначиться на всій програмної екосистемі Apple. Оскільки розробка додатків для iOS і OS X стає все простіше і легше, багатьом професіоналам в інших мовах і платформах захочеться спробувати свої сили на новому полі діяльності. Але що це може означати?

Насамперед, це означає, що до мобільних і комп'ютерних платформ Apple буде залучено більше число розробників. Більше розробників - це більше додатків і більше вибору для споживача. А от питання про якісний рівень такого софта доведеться залишити відкритим. Поки що Swift занадто нова технологія і вона ще не освоєна навіть професійними програмістами для пристроїв Apple.

На додаток до UIKit, колекція Cocoa Touch framework-ів включає в себе все необхідне для створення світового класу IOS додатків. Велика частина Cocoa Touch реалізована в Objective-C, об'єктно-орієнтованої мови, який компілюється для запуску з неймовірною швидкістю, а використання дійсно динамічного виконання робить його унікально гнучким. Cocoa Touch включає в себе потужні Objective-C framework-і, які виконують всі завдання лише кількома рядками коду, забезпечуючи при цьому основні C мовної-API, щоб дати прямий доступ до системи, коли це необхідно.

В кінцевому рахунку, Swift найбільш доступний повнофункціональний мова програмування, яка дозволить розробникам не тільки створювати додатки, але і програмувати вбудовані системи, типу нової низько споживаної Apple Watch, на багато років вперед.

3. АНАЛІЗ ФРЕЙМВОРКІВ ТА МОЖЛИВОСТЕЙ СИСТЕМИ

У першій версії розумних годин Apple дозволила стороннім розробникам створювати додатки, які є розширенням основного додатки (тут і далі додаток, що працює на iPhone, будемо називати «основним додатком»). А додаток, що запускається на годиннику - «WatchKit-додатком», або просто «додатком» на iPhone. Розробку виключно під Apple Watch обіцяють зробити в майбутньому. Зараз функціональність годин доступна через фреймворк WatchKit і сильно обмежена в порівнянні з тим, що ви могли бачити на презентації самих годин. Видно, що розробники в Apple і ще в кількох «обраних» компаніях мають більші можливості, ніж інші.

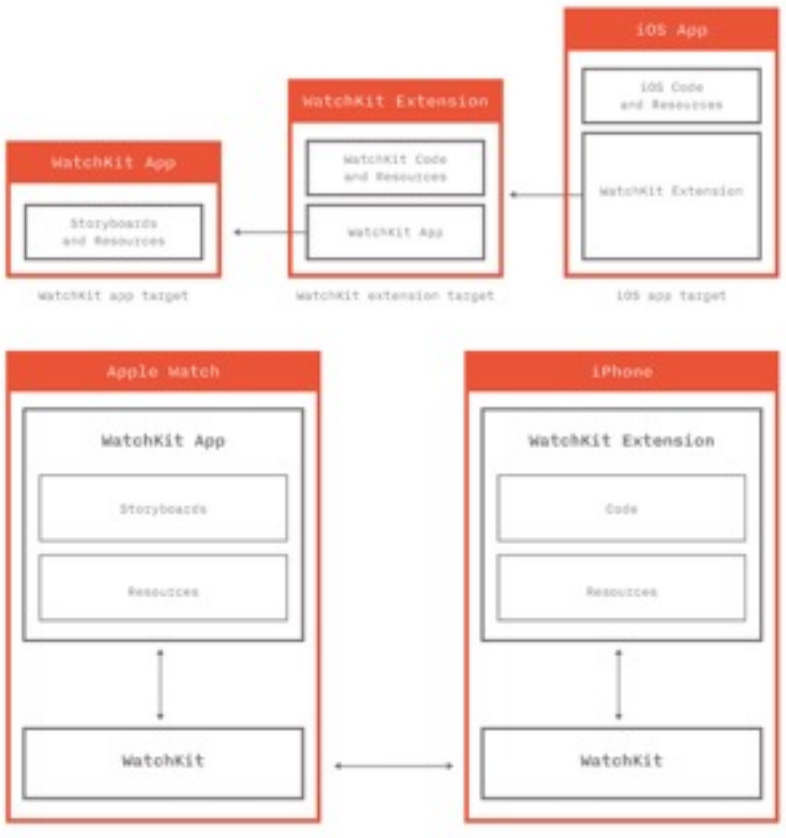

Рисунок 3.1 - Архітектурна модель роботи додатку на Apple Watch

З точки зору архітектури, все влаштовано по-іншому. Файли додатки

поділяються на дві частини: WatchKit App і WatchKit Extension. WatchKit App зберігається безпосередньо на годиннику і складається з сторіборди (опису інтерфейсу) і ресурсів. WatchKit Extension зберігається на iPhone і містить код і ресурси. Таким чином, весь код виконується на телефоні, і без iPhone неподалік ваше додаток на годиннику не запуститься. Взаємодія між телефоном і годинами здійснюється через фреймворк WatchKit і приховано від розробника.

Кожен екран керування управляється об'єктом-контролером класу WKInterfaceController. Всі контролери, включаючи Glance і повідомлення, повинні бути виконані в сторіборде WatchKit App. Коли користувач відкриває додаток на годиннику, WatchKit знаходить початковий контролер в сторіборди і повідомляє на iPhone, що необхідно запустити WatchKit Extension і створити об'єкт потрібного класу контролера. Аналогічні дії відбудуться, якщо користувач відкриє Glance або отримає повідомлення.

Істотна відмінність від розробки основного iOS-додатки - метод ініціалізації контролерів init викликається не розробником, а системою. Параметри і дані передаються від контролера до контролера через контекст. Будь-який перехід від екрану до екрану відбувається або автоматично по сігвеї, або з коду за ідентифікатором контролера в сторіборди. В обох випадках WatchKit знаходить потрібний екран в сторіборди, після чого WatchKit Extension створює об'єкт класу, зазначеного в сторіборди біля екрану з даними ідентифікатором (по даним сігвеї), і викликає у нього метод init. Стек контролерів залишається "під капотом" системи. Розробник не бачить його і працює з кодом контролерів, як з абстрактними і ізольованими сутностями.

3.1. HealthKit

HealthKit - це платформа, яка збирає інформацію про ваш стан не тільки на основі датчиків одного трекера, а звідусіль, звідки це тільки можливо. З додатків, які встановлені на вашому iPhone і iPad, з фітнес-пристроїв. Таким чином HealthKit видаватиме повну картинку про ваш стан. Мало того, в iOS 8 додатка зможуть обмінюватися інформацією з HealthKit, щоб зробити тренування ефективніше або підкоректувати дієту. Наприклад, Nike + зможе

отримати загальну інформацію щодо білоруського режиму сну і харчування з HelthKit, щоб створити унікальний профіль тренувань. Так само повідомляється, що не тільки Найк співпрацюватимуть з Apple в боротьбі за здоров'я нації. Профільних медичних компаній зібралося більше десятка. Зокрема, Health буде стежити за вашим пульсом, тиском, пройденими кілометрами, кількістю спалених калорій, рівнем цукру і холестерину в крові. І якщо щось з цього буде не в порядку, то додаток зможе відправити "тривожний дзвінок" лікареві. Розробники на сайті пишуть, що всі ваші "оздоровчі" додатка сильні в збірці, і ось цей додаток і є той самий “збір”.

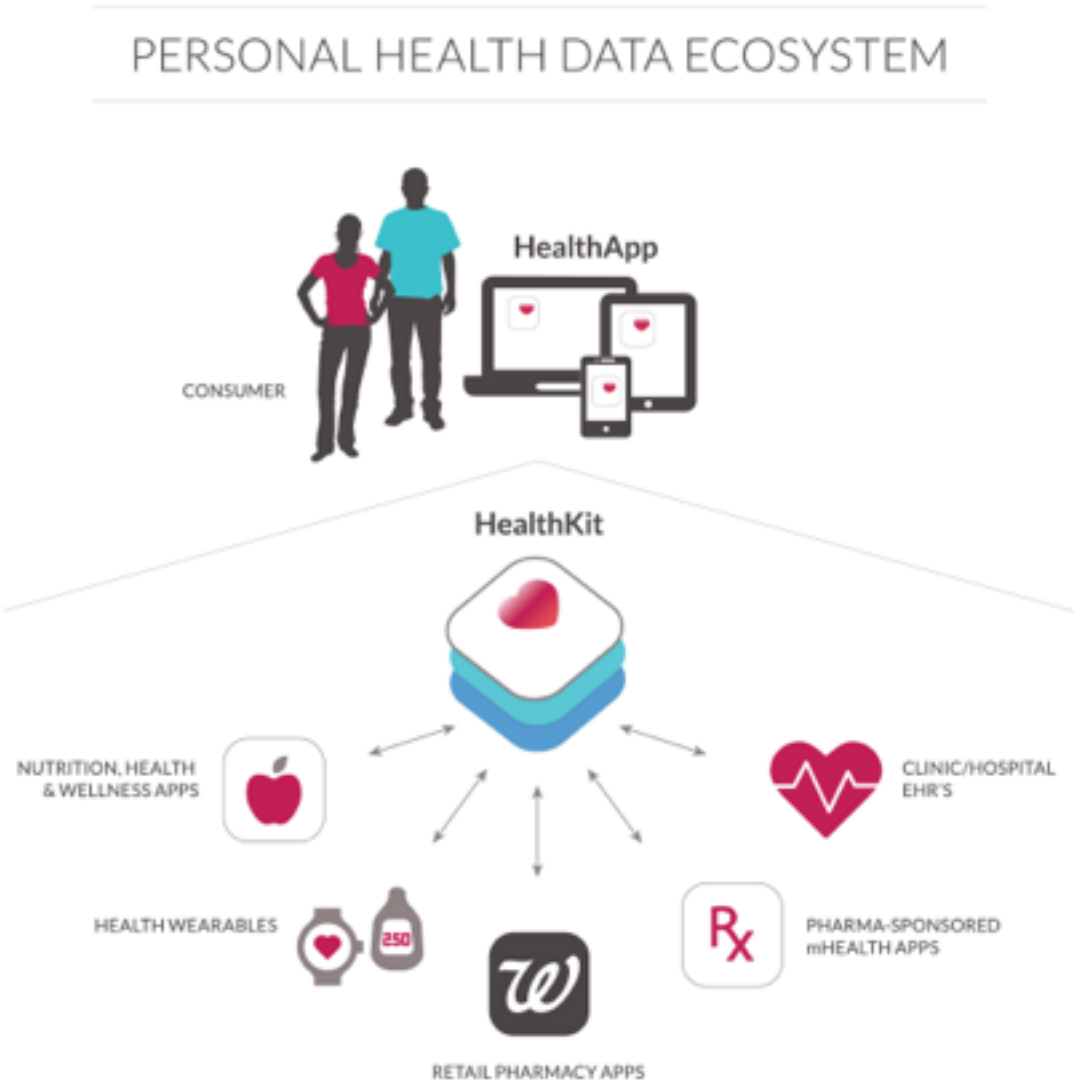

Рисунок 3.2 - Архітектурна модель роботи екосистеми HealthKit

Сторонні розробники теж можуть створювати свої продукти, які взаємодіють з HealthKit. До того ж у розробників безмежна фантазія. Наприклад, можна зробити так, щоб HealthKit мав кілька відкритих профілів і ваш лікар в будь-який момент можу зайти і подивитися дані про ваш стан здоров’я. Завдяки HealthKit розробники зможуть зробити свої додатки ще корисніше, оскільки тепер їм буде доступна інформація про ваше здоров'я. При цьому ви зможете вибрати, які дані надавати. Наприклад, можна автоматично відправляти вашого лікаря дані з програми, що вимірює тиск. Або налаштувати додаток, яке стежить за вашим харчуванням, таким чином, щоб інформація про споживаних щодня калоріях відправлялася в додатки для фітнесу. Спільна робота додатків для здоров'я та фітнесу відкриє їх нові можливості. Так само як і ваші.

В даний момент WatchKit дозволяє сумісним додаткам взаємодіяти з користувачем за допомогою трьох інтерфейсних моделей: Glances, «Активні повідомлення» і «додатки на базі WebKit».

Додатоки можуть зберігати всі дані про ваше здоров'я і фізичні активності в єдиному інтерфейсі і в повному порядку. Інформація про вас належить тільки вам, і тільки вам вирішувати, як її використовувати і кому відправляти. Ви можете вибрати дані, які будуть відображатися в додатку «Здоров'я», і вказати, у яких додатків буде до них доступ. Коли ваш телефон блокується за допомогою пароля або Touch ID, всі дані, що зберігаються в додатку, шифруються. Ви можете створити їх резервну копію в iCloud. Вони будуть шифруватися і при пересиланні, і при зберіганні. Кожна програма, що використовує HealthKit, повинно мати положення про конфіденційність, тому не забудьте ознайомитися з ним, перш ніж відкривати доступ до своїх даних про здоров'я і фізичному стані. Таким чином, завдяки Apple Watch ви отримуєте інформацію, яка допомагає підвищити результативність тренувань і поліпшити здоров'я, а разом з нею - необхідну для цього мотивацію.

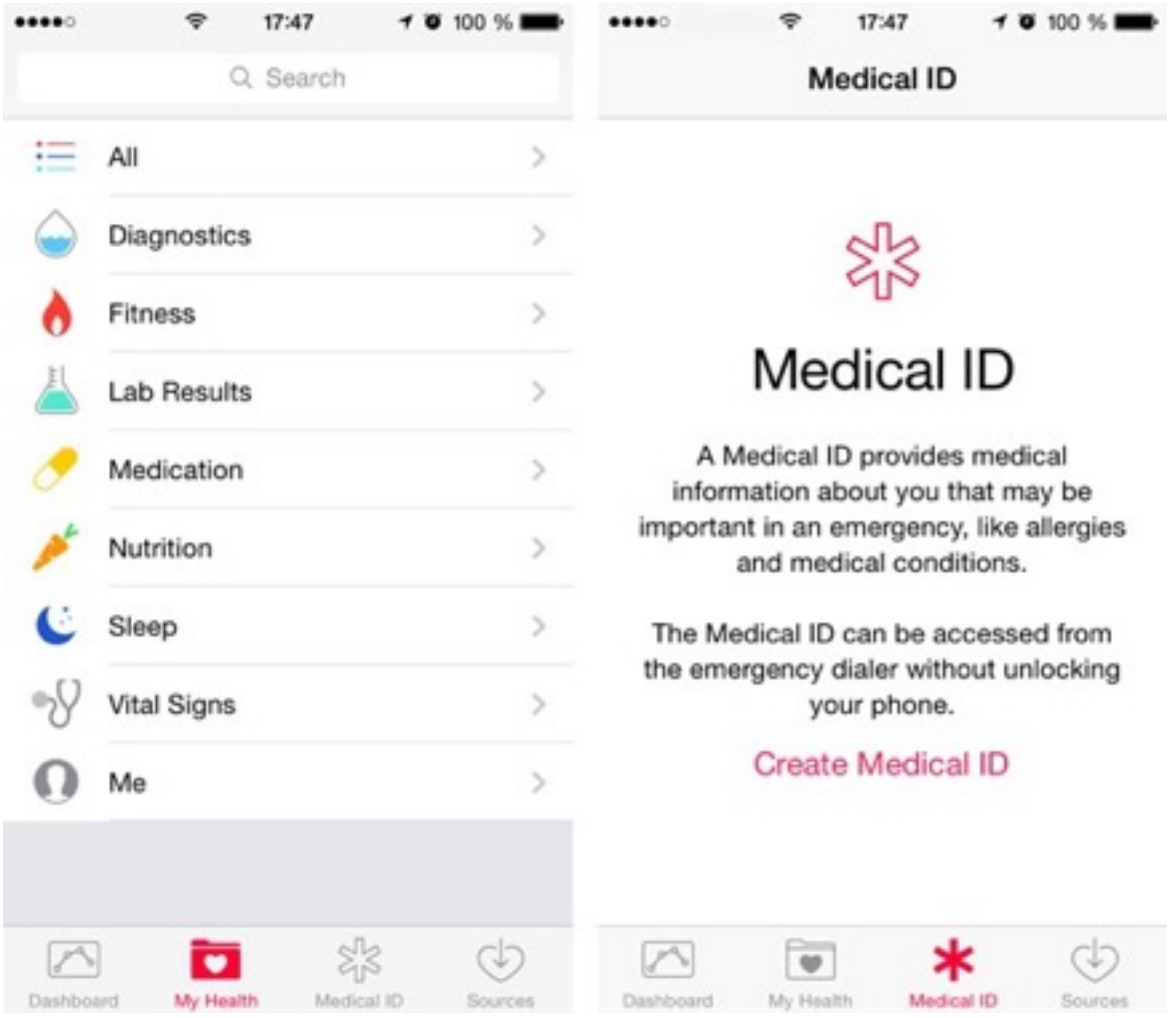

Рисунок 3.3 - Медична карта користувача в додатку “Health”

Більшість звітів посилається через телефон або факс, але HealthKit може збирати дані з медичних апаратів, завантажуючи інформацію в систему і даючи користувачам можливість ділитися їй з лікарем.

Щоб заспокоїти користувачів було заявлено, що Apple не отримає доступу до персональних даних.

Медичні установи повинні будуть пройти сертифікацію Apple для того, щоб використовувати нову платформу, що збільшує конфіденційність даних. Вся інформація буде зберігатися в безпеці на пристроях і не може бути надана іншим особам, наприклад, рекламодавцям.

- Apple працює з декількома виробниками обладнання в США для створення взаємодії між медичними приладами та пристроями на iOS 8, підтримують

HealthKit.

- Пацієнти, які страждають на діабет, зможуть використовувати більш дешеві пристрої, як iPod touch, для відстеження рівня цукру в крові.
- Партнери Apple, такі як Epic Systems або DexCom, створюють програмне забезпечення і різні рішення для HealthKit. Урядова організація з контролю медичного обладнання та лікарських засобів в США бажає інтегрувати HealthKit з новим обладнанням, використовуючи принцип plug-and-play.
- Дані з медичних приладів можуть бути передані в iPhone, iPad або iPod touch і потім спрямовані доктору, віддалено і безпечно.
- Записи про здоров'я і вся зібрана інформація може бути завантажена з HealthKit в додаток від Epic, для перегляду лікарем і збереження в спеціальній електронній бібліотеці Epic.
- Рівень цукру буде вимірюватися невеликими сенсорами, встановленими на шкірі пацієнта, дані з яких будуть надсилатися кожні 5 хвилин на переносний приймач, який може зберігатися в кишені. Далі ця інформація надходить в додаток DexCom, на пристрої користувача.

3.2. Вимірювання пульсу

Ви можете в будь-який час дізнатися частоту свого пульсу в превью Пульс. При використанні програми «Тренування» годинник Apple Watch вимірюють пульс протягом всього тренування. Ця інформація, а також інші зібрані дані допомагають Apple Watch оцінювати кількість спалених калорій. Крім того, годинник Apple Watch вимірюють частоту пульсу протягом усього дня, коли ви не рухаєтесь. Переглянути ці показники частоти пульсу можна в програмі «Здоров'я» на пристрої iPhone. Оскільки годинник Apple Watch зчитують показання в фоновому режимі тільки тоді, коли ви не рухаєтесь, інтервали вимірювань можуть бути нерівномірними.

Рисунок 3.4 - Датчик вимірювання пульсу

Датчик серцебиття Apple Watch працює на основі фотоплетізмографія. В основі цієї технології зі складною назвою лежить один простий факт: червоний колір крові обумовлений тим, що вона відображає червоне світло і поглинає зелений. Завдяки зеленим светодиодам в парі з світлочутливими фотодіодами годинник Apple Watch можуть заміряти обсяг крові, що проходить через зап'ясті в кожен момент часу. При ударі серця обсяг потоку крові в зап'ясті (а з ним і обсяг поглиненого зеленого світла) стає більше, а в періоди між ударами - менше. Частота миготіння світлодіодів Apple Watch - кілька сотень спалахів в секунду, завдяки чому годинник може виміряти кількість ударів серця в хвилину, тобто ваш пульс. Крім того, датчик серцебиття здатний компенсувати слабкий сигнал за рахунок збільшення яскравості світлодіодів і частоти дискретизації.

Датчик серцебиття може також використовувати інфрачервоне випромінювання. Саме так годинник Apple Watch вимірюють ваш пульс у фоновому режимі.

Для отримання оптимальних результатів необхідно носити годинник правильно. Навіть в ідеальних умовах годинник Apple Watch можуть час від часу вимірювати пульс неточно. А в окремих випадках різні фактори роблять

зчитування показників пульсу неможливим. Однак існують способи підвищити точність і акуратність вимірювання пульсу за допомогою Apple Watch.

Що ще може вплинути на показники? На точність роботи датчика серцебиття може впливати безліч факторів. До них відноситься, наприклад, кровопостачання шкіри. Кровопостачання шкіри - це своєрідний спосіб визначити обсяг крові, що надходить до судин в шкірі. Воно сильно залежить від особливостей організму і навколишнього середовища. Наприклад, при тренуваннях в холодну погоду кровопостачання шкіри зап'ястя може бути настільки незначним, що датчик серцебиття не зможе його виміряти.

Ще одним фактором, що впливає на показання датчика серцебиття, є рух. Ритмічні рухи, наприклад при бігу або їзді на велосипеді, забезпечують більш точні результати в порівнянні з заняттями тенісом або боксом, де спортсмени рухаються нерегулярно.

Постійні або тимчасові зміни шкірного покриву (наприклад, татуювання) також можуть впливати на роботу датчика серцебиття. Чорнило, контури і насиченість деяких татуювань можуть блокувати проходження світла до датчика, що знижує ймовірність отримання правильних результатів.

Якщо вам не вдається отримати стабільні показники по будь-якої з цих причин, скористайтеся модулями бездротового зв'язку Apple Watch для підключення до зовнішніх пульсометра, наприклад нагрудним датчикам з підтримкою Bluetooth.

Пульс - це один з багатьох факторів, на підставі яких годинник Apple Watch визначають ступінь активності і інтенсивності виконання фізичних вправ. Для кожного виду тренування вибирається найбільш відповідний їй показник. Наприклад, при бігу в приміщенні задіюється акселерометр, а під час їзди на велосипеді - датчик GPS iPhone. Навіть якщо ви не займаєтеся тренуваннями певного типу, годинник відстежують, скільки ви рухаєтеся протягом дня. Таким чином, завдяки Apple Watch ви отримуєте інформацію, яка допомагає підвищити результативність тренувань і поліпшити здоров'я, а разом з нею - необхідну для цього мотивацію.

3.3. WatchKit

WatchKit - це фреймворк компанії Apple для створення гібридних додатків для годин Apple Watch, до складу якого входить Xcode 6.2. WatchKit дозволяє створювати два варіанти додатків. Перші використовуватимуть ресурс iPhone, який стане транслювати інформацію і зображення на екран «розумних» годин. Це необхідно для економії енергії годин, так як потужний процесор iPhone зможе набагато швидше обробити потрібну інформацію. Другий варіант додатків - працюють на самих Apple Watch. Правда, таких поки мало: це Годинники, Календар і Таймер, а також сторонні розробки з таким же набором функцій.

Одним з найдивовижніших аспектів WatchKit стала сама його архітектура. Ось як працює ваша програма (фактично розділене на дві частини):

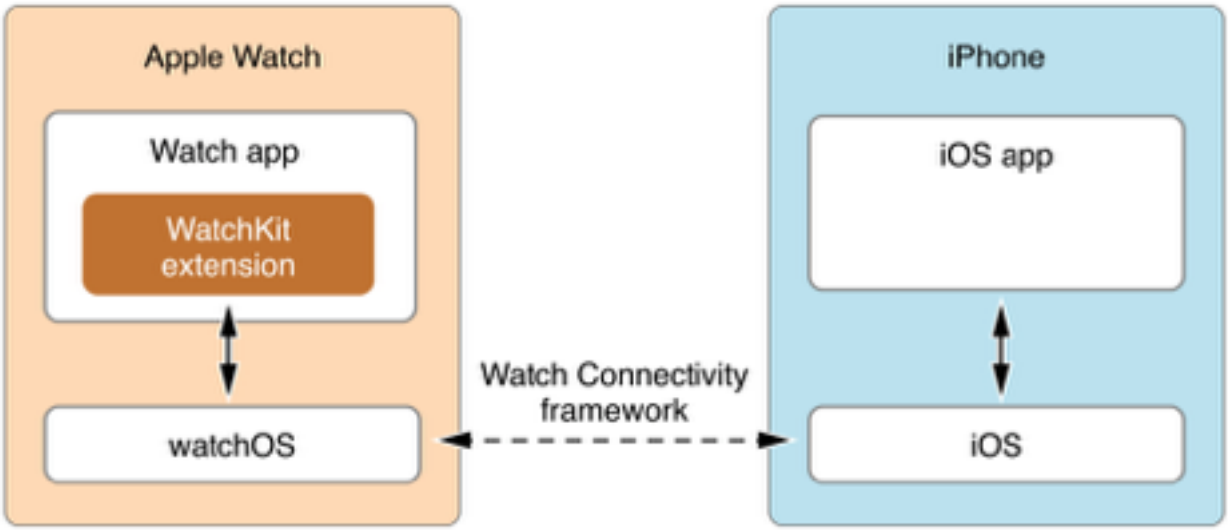

Рисунок 3.5 - Архітектура WatchKit

- Ваш Apple Watch містить ресурси вашого призначеного для користувача інтерфейсу (сторіборди і статичні зображення) і обробляє дані, що вводяться користувачем, але фактично не виконує жодного коду. Іншими словами, ваші Apple Watch є просто відображає вікном (view).

- Ваш iPhone містить код, який відповідає на будь-які події, наприклад на запуск програми від натискання на кнопку або зміни будь-якого значення. Іншими словами, ваш iPhone містить контролер і модель (controller and model).

Взагалі, слід відразу обмовитися: повноцінні і самодостатні сторонні додатки за допомогою WatchKit створювати поки не можна. Принаймні на початку свого шляху Apple Watch з точки зору розробників будуть виглядати як ще один елемент інтерфейсу iPhone, і саме на смартфоні будуть працювати самі додатки. У наступному році компанія обіцяє надати стороннім розробникам можливість створення і повноцінних додатків для Apple Watch, однак відбудеться це, ймовірно, не раніше, ніж будуть визначені параметри вимог, які необхідно пред'являти до таких додатків, і в першу чергу - з точки зору ефективного використання заряду батареї. А це можливо лише після того, як буде проаналізовано інформацію, отримана від власників новинки. Втім, ті кілька місяців, які залишилися до надходження годин в продаж, сторонні розробники можуть провести за «втягуванням» в модель використання годин, запропоновану Apple. Тобто з користю для свого досвіду і для самого пристрою, якому на момент ринкового дебюту зовсім не зашкодить широкий вибір відповідних додатків в онлайновому магазині App Store. Для роботи з WatchKit розробникам необхідно завантажити нову версію середовища розробки Xcode і інтегрувати нові можливості в існуючі проекти додатків для iOS.

В даний момент WatchKit дозволяє сумісним додаткам взаємодіяти з користувачем за допомогою трьох інтерфейсних моделей: Glances, «Активні повідомлення» і «додатки на базі WebKit».

Glances - односторінкові віджети, що відображають будь-яку інформацію, начебто прогнозу погода або новин. Між сусідніми Glances, створеними різними додатками, можна переключатися жестом гортання вправо або вліво. Glances можна порівняти з віджетами з вкладки «Сегодня» з «Центру повідомлень» iOS.

Однак є і суттєва відмінність: Glances неінтерактивному, вони не можуть

взаємодіяти з користувачем. Зовнішній вигляд сторінок Glances визначається пропонованими Apple шаблонами. Натискання на сторінку призводить до відкриття відповідної програми.

Другий варіант - «Активні повідомлення», які, в свою чергу, дозволяють не тільки переглядати повідомлення, але і відповідати на них, виконуючи запропоновані дії. Повідомлення бувають двох видів - короткі та повні (в оригіналі - Short Look і Long Look notifications).

Рисунок 3.6 - Приклад Glances

Короткі повідомлення відображають мінімум інформації: іконку програми, короткий опис події (наприклад, «Нове повідомлення») і назва уведомляющего додатки. Якщо ж підняти руку, на яку надіті годинник, або просто торкнутися екрану, то відобразиться повний текст повідомлення, що містить іконку програми, повну інформацію з повідомлення, а також, в разі необхідності, кнопки для виконання якихось дій у відповідь на повідомлення. Apple вимагає від розробників створювати по два варіанти повних повідомлень: звичайні динамічні і статичні, які використовуються в режимі низького енергоспоживання.

Нарешті, «додатки на базі WatchKit» дозволяють розробникам створювати користувальницький інтерфейс, що працює на Apple Watch, для додатків, запущених на iPhone.

Таблиця 3.1 - Таблиця кореляції класів WatchKit з UIKit

WatchKit	**UIKit**
WKInterfaceController	UIViewController
WKUserNotificationInterfaceControl ler	UIApplicationDelegate
WKInterfaceDevice	UIDevice
WKInterfaceObject	UIView
WKInterfaceButton	UIButton
WKInterfaceDate	UILabel
WKInterfaceGroup	UIScrollView
WKInterfaceImage	UIImageView
WKInterfaceLabel	UILabel
WKInterfaceMap	MKMapView
WKInterfaceSeparator	UITableView.separatorColor
WKInterfaceSlider	UIStepper
WKInterfaceSwitch	UISwitch

Сенсорний Інтерфейс Apple Watch відрізняється від звичного Multi Touch, який став стандартом де-факто завдяки iPhone. Основні жести сенсорного інтерфейсу годин - дотик і огляд, проте звичні багатоточкові жести (на кшталт масштабування розтягуванням) тут відсутні. Як збільшувати або зменшувати використовується коліщатко Digital Crown. Крім того, на відміну від iPhone і IPad, в інтерфейсі Apple Спостерігати за Force сенсорний Присутній - жест натискання одним пальцем, який використовується, наприклад, для виклику контекстних елементів інтерфейсу або меню. Функції звичних жестів зумовлені Apple, і не можуть бути змінені і доповнені: вертикальне гортання активує скролінг, горизонтальне гортання перемикає сторінки, дотиками здійснюється вибір елементів.

Таблиця 3.2 - Відпові UI елементу і виклику його функції

Object	**Action Method**
Button	- (IBAction)doButtonAction
Switch	- (IBAction)doSwitchAction: (BOOL)on
Slider	- (IBAction)doSliderAction: (float)value
Table	- (IBAction)doTableRowTapAction: (NSInteger)rowIndex
Menu Item	- (IBAction)doMenuItemAction

В даний момент компанія Apple обмежує розробників двома варіантами основного інтерфейсу додатків на базі WatchKit. Перший з них - навігаційний стек - ієрархічний, аналогічний інтерфейсу додатку «Налаштування» з ОС IOS: користувач може вибирати один з пунктів меню для проходження на більш низький рівень або ж використовувати кнопку повернення на попередній рівень.

3.4. Core Data

Apple надає еластичний фреймворк для роботи з збереженими на пристрої даними - Core Data. Безумовно ж Core Data не панацея і є інші варіанти зберігання даних, які можуть скасувати підійти при вирішенні певних завдань, але вже дюже відмінно і прекрасно Core Data вписується в Cocoa Touch. Безліч деталей по роботі зі сховищем даних Core Data приховує, дозволяючи вам сконцентруватися на тому, що справді робить ваше додаток веселим, унікальним і комфортним в застосуванні.

Не дивлячись на те, що Core Data може берегти дані в реляційної базі даних як би SQLite, Core Data не є СУБД. По-правді Core Data в якості сховища може взагалі не застосовувати реляційні бази даних. Core Data не є чимось як

би Hibernate, правда і надає деякі ймовірності ORM. Core Data швидше є оболонкою / фреймворком для роботи з даними, що дозволяє працювати з сутностями і їх зв'язками (відносинами до інших об'єктами), ознаками, в тому вигляді, той, що нагадує роботи з об'єктним графом в звичайному об'єктно-орієнтованому програмуванні.

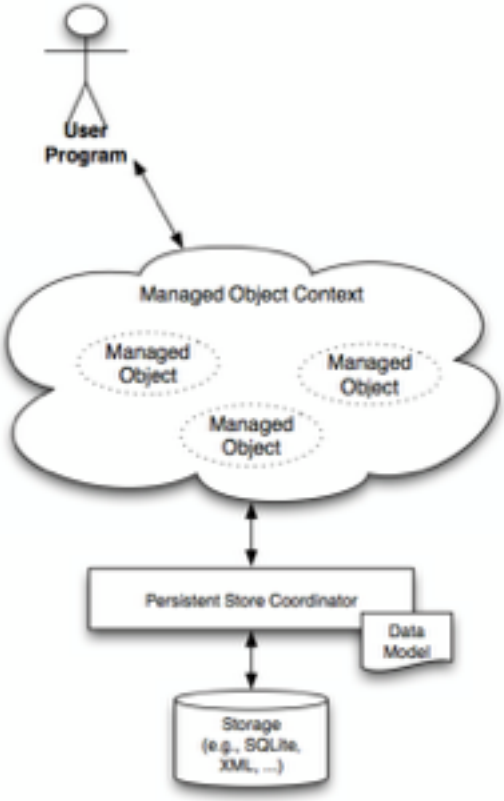

Рисунок 3.7 - Архітектурна модель роботи Core Data

Ми вже бачили багато бібліотек намагаються запропонувати такий же рівень можливостей на вершині SQLite, за рахунок швидкості. На противагу цьому, Realm швидше, ніж навіть сирої SQLite на загальних операцій, зберігаючи при цьому надзвичайно багатий набір функцій.

Не дивлячись на те, що Core Data може берегти дані в реляційної базі даних як би SQLite, Core Data не є СУБД. По-правді Core Data в якості сховища може взагалі не застосовувати реляційні бази даних.

Таблиця 3.3 - Порівняння характеристик SQL та Core Data

SQL	**Core Data**
Первинна функція зберігання і вибірки даних	Основна функція є управління графом даних (хоча читання і запис на диск є важливою підтримкою функція)
Працює на даних, що зберігаються на диску	Працює на об'єктах, що зберігаються в пам'яті
Працює з “сирими” даними	Працює з повноцінними об'єктами, які самостійно управляють своєю поведінкою
Може бути транзакційними, поточно-, розрахований на багато користувачів	Нетранзакціонних, однопотоковий, один користувач (якщо ви не створити цілу абстракції навколо основних даних, яка забезпечує ці речі)
Можна видаляти таблиці і редагувати дані без завантаження в пам'ять	Працює тільки в пам'яті
Завжди зберігає зміни на диску	Потрібен процес збереження
Може бути повільною, щоб створити мільйони нових рядків	Можна створити мільйони нових об'єктів в пам'яті дуже швидко (хоча збереження цих об'єктів буде повільним)
Існує обмеження даних, як "унікальних" ключів	Залишає обмеження даних для бізнес-логіки програми

Структура була оптимізована протягом декількох версій. Вона використовує інформацію, що міститься в моделі і виполненять функції, як правило, не працюють на рівні додатків в коді. Крім того, на додаток до відмінної безпеки і обробці помилок, вона пропонує кращу масштабованість при роботі з пам'яттю, щодо будь-якого конкуруючого рішення. Іншими словами: ви могли б витратити довгий час ретельно обробляючи Ваші власні рішення оптимізації для конкретної предметної області.

Також, з поміж інших рішень, можемо провести порівняльну характеристику:

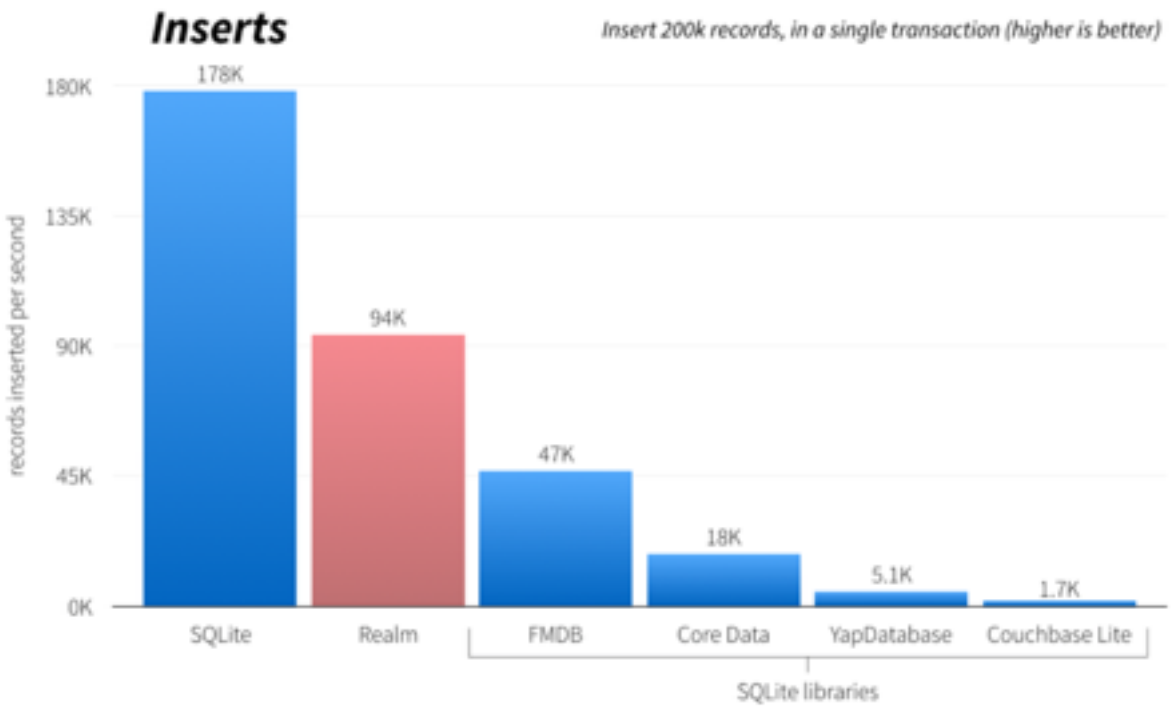

Рисунок 3.8 - Швидкість встаски об’єктів за секунду

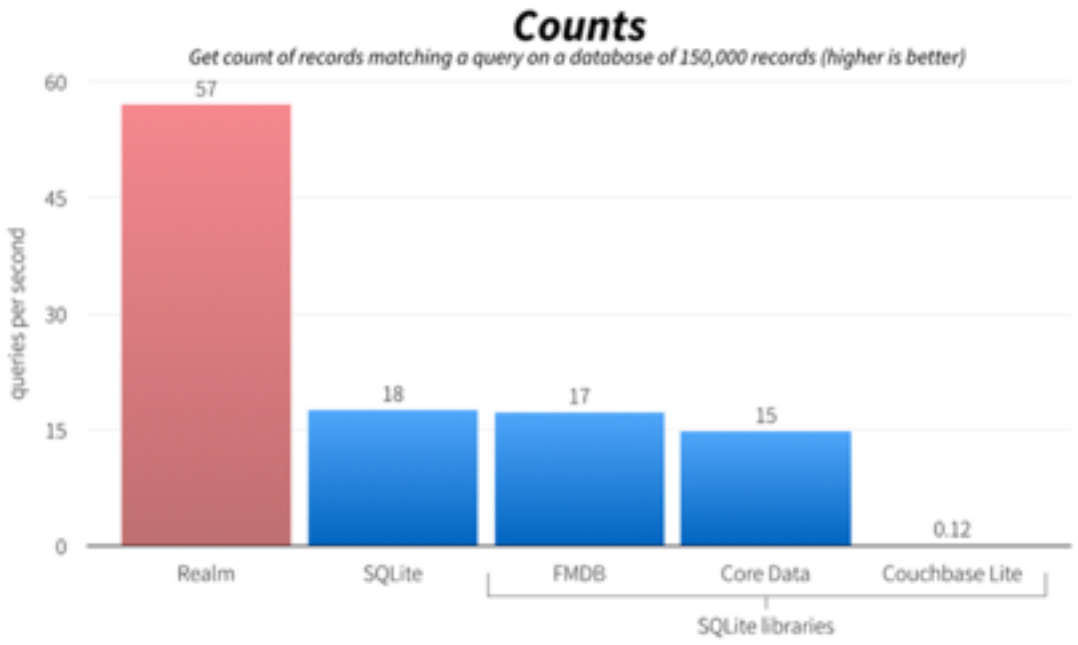

Рисунок 3.9 - Знаходження кількості елементів по критерію

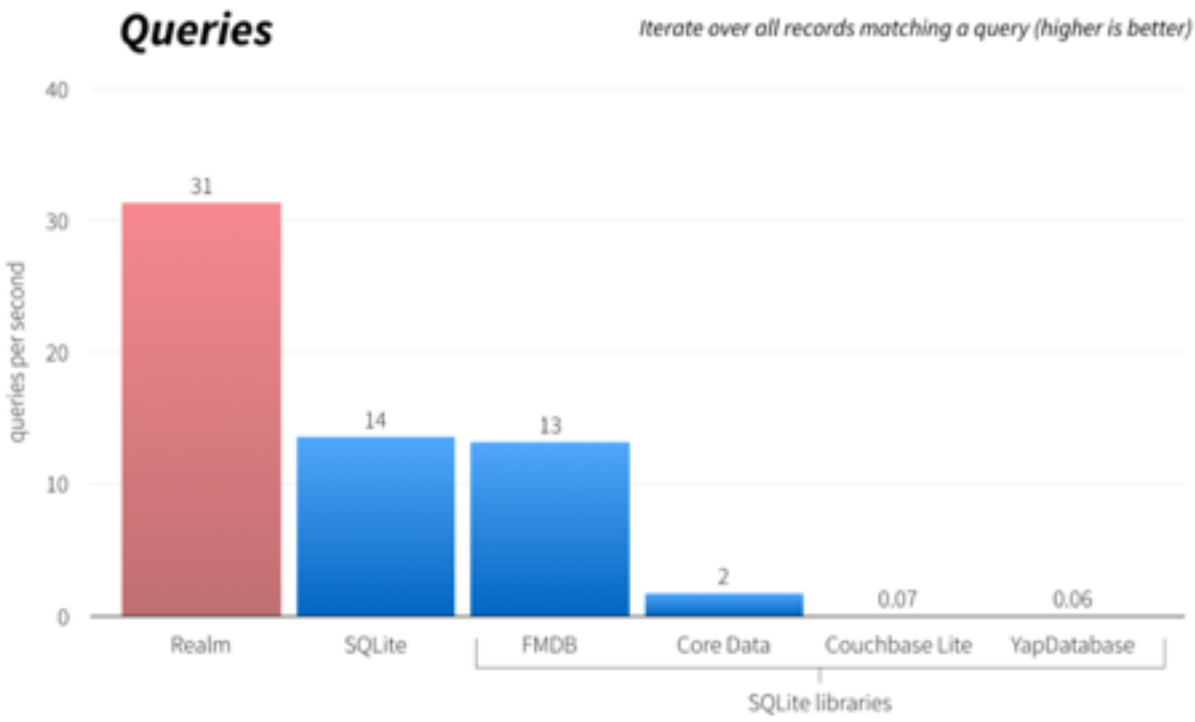

Рисунок 3.10 - Ітерація по всих елементах

Отже, як бачимо, Core Data є зручним і надійним рішенням при побудові додатків з внутрішньою базою даних на девайсі. Контекст несе відповідальність за посередництво між керованими об'єктами і іншою інфраструктурою Core Data. Інфраструктура, в свою чергу відповідальна, наприклад, за, переклад змін до керованих об'єктах в undo дію підтримуване контекстом, а також в операціях, які повинні бути виконані на постійному сховище, з яким керований об'єкт асоційований.

Контекст, по суті також є Вашим пропуском в іншу частину інфраструктури Core Data. Таким чином, очікується, що ви або збережіть посилання на контекст, або у вас є кошти легко витягувати її, наприклад, якщо ви розробляєте документ-додаток, яке використовує NSPersistentDocument, ви можете використовувати managedObjectContext метод класу документа. Керований об'єкт є екземпляром Objective-C класу. З цієї точки зору, він нічим не відрізняється від будь-якого іншого об'єкта, який ви використовуєте, ви можете просто створити екземпляр за допомогою alloc.

3.5. Висновки

Після детально ознайомлення з інструментом для створення додатків для Apple Watch, ми маємо можливість впевненно сказати, що ці інструменти можна вважати невідємною частиною сучасного інструментарію Apple для операційнних систем iOS та WatchOS.

WatchKit дає нам можливість створювати інтерфейси за допомогою GUI в Interface Builder в программному продукті Apple, а також за допомогою коду.

На поведінку інструменту впливають різноманітні чинники, до яких відносяться як внутрішні зміни, так і зовнішні зміни. Різноманітні обмеження, які можливо накласти на позиціонування елементів, є ключовою можливістю для вирішення задач при проектуванні складних, багато-залежних, призначених для користувача інтерфейсів.

Анатомія CoreData є складним елемнтом, який розробники Apple зуміли зробити зрозумілим для розробників. Взагалом ця система є логічним кроком компанії, який призведе до збільшення кількості якісних продуктів у вигляді додатків для iPhone та Apple Watch. Core Data має перевірений код, якість якого забезпечується шляхом юніт-тестів, і використовується щодня мільйонами клієнтів в широкому спектрі додатків. Структура була оптимізована протягом декількох версій. Вона використовує інформацію, що міститься в моделі і виполненять функції, як правило, не працюють на рівні додатків в коді. Крім того, на додаток до відмінної безпеки і обробці помилок, вона пропонує кращу масштабованість при роботі з пам'яттю, щодо будь-якого конкуруючого рішення. Іншими словами: ви могли б витратити довгий час ретельно обробляючи Ваші власні рішення оптимізації для конкретної предметної області, замість того, щоб отримати перевагу в продуктивності, яку Core Data надає безкоштовно для будь-якої програми.

4. ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ДОДАТКУ

4.1. Опис додатку

Під час обговорення ідеї програми для дипломною роботи, наша команда знайшла рішення, яке б задовільняла всі сторони. Було вирішено створити мобільний додаток для iPhone, а також для Apple Watch.

Додаток був розроблений для користувачів, які займаються бігом. Він буде давати змогу користувачеві аналізувати свою показники після бігу, а саме:

- Перегляд пульсу
- Кількість затрачених калогій
- Геолокаційні дані положення

Усього система буде складатись з 2 додатків.

1. Додаток для Apple Watch

Основі функцій:

- старт програми
- початок тренування
- зупинка тренування
- збереження даних під час бігу
- вимір пульсу
- зчитування кількості калорій на кожному проміжку фіксації пульсу
- зчитування гео-локаційних даних на кожному проміжку
- передача даних на iPhone
- Інтеграція з HealthKit
- зупинка тренування

- вимкнення програми

2. Додаток для iPhone

Основні функцій:

- запуск программ
- збереження даних
- відображення всіх тренуваннь
- вибір тренування
- перегляд карти на якій зображені точки
- вибір точки і перегляд інформації пульс і час коли буи зроблений замір
- вихід з перегляду тренування
- видалення тренування
- синхронізація з годиником

Розглянемо основні екрани додатків.

Apple Watch

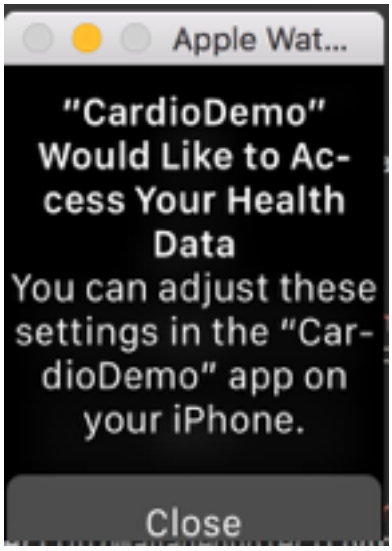

Рисунок 4.1

На рисунку 4.1 зображен запит для користувача на дозвіл користування Health Data на розумному годинику. Операційна систеса рабить такий запит для того щоб користувачі розуміли що саме буде використовувати додаток. У випадку якщо користвачвваже що додаток не повинен мати можливість отримувати дані, то він може натиснути Close.

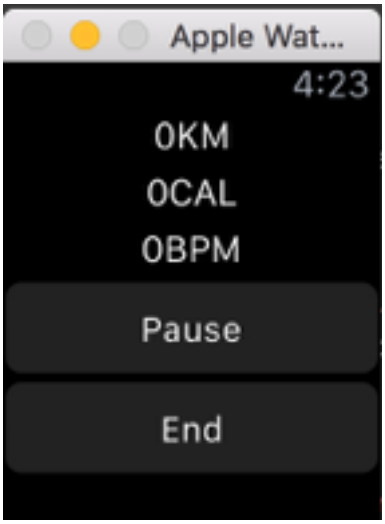

Рисунок 4.2 - головний екран тренування

На рисунку 4.2 зображен головний екран додатку. На ньому користувач має змогу почати тренування та завершіті тренування. Таклж е можливість переглядати дані останього виміру.

Рисунок 4.3 - екран годиника

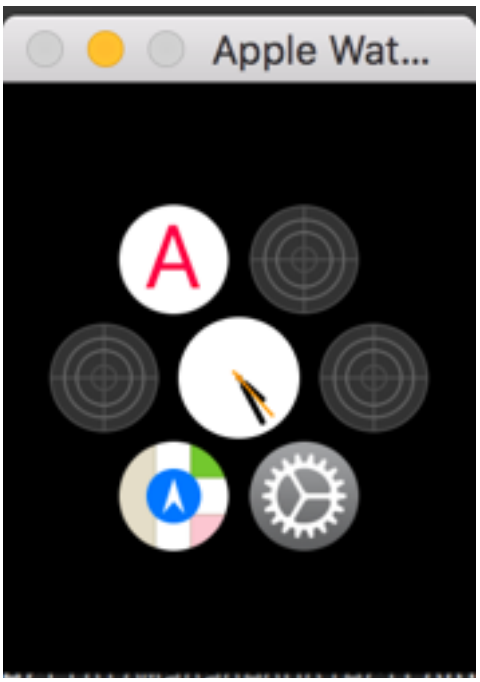

Рисунок 4.4 - Вигляд додатку у меню запуску додатків

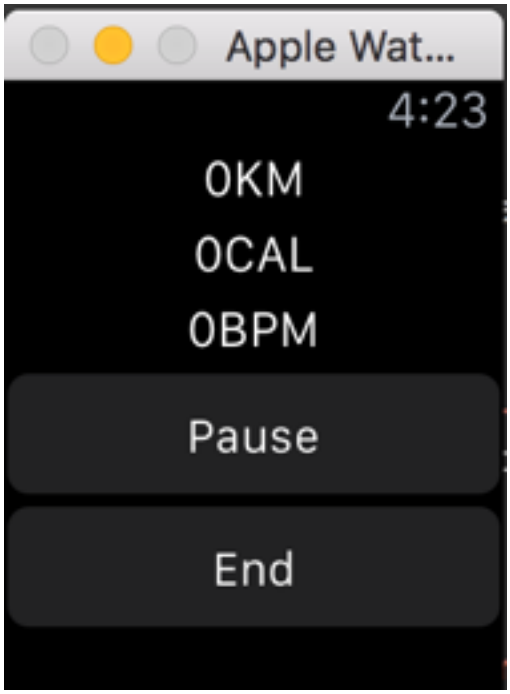

Рисунок 4.5 - Головний екран у робочому режимі з кнопокою Pause

iPhone додаток

Рисунок 4.6 - Головне меню додатку з вибором тренування

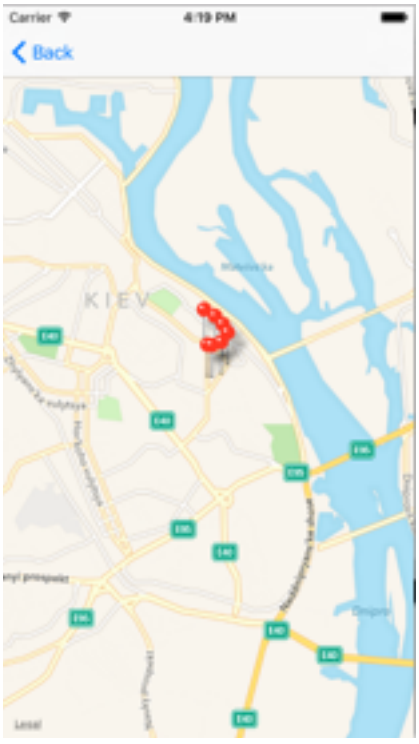

Рисунок 4.7 - Карта після вибору тренування

Рисунок 4.8 - Відображення інформації принатисканні на Pin

4.2. Cocoa Touch як програмний каркас для створення WatchOS та iOS додатків

Cocoa Touch - це фреймворк для створення додатків під iPhone, iPod touch, і iPad.

Бібліотека Cocoa Touch надає рівень абстракції для iOS (операційної системи iPhone, iPad і iPod touch). Cocoa Touch заснована на класах фреймворка Cocoa, використовуваного в Mac OS X, і, аналогічно їй, використовує мову Objective-C. Cocoa Touch слід шаблоном проектування Model-View-Controller.

Інструменти для розробки додатків з використанням Cocoa Touch включені в iOS SDK.

Як і у всіх середовищах додатків, Cocoa має два світи: світ runtime'a і світ розробки. У світі runtime'a, Cocoa додатки представляють призначений для користувача інтерфейс і тісну інтеграцію з іншими компонентами операційної системи в Mac OS X, наприклад, Finder і Dock.

Але в світі розробки Cocoa - інтегрований набір об'єктно-орієнтованих програмних компонентів-класів, які, власне, дозволяють творити ПО під Mac OS X і IOS. Вони дають можливість робити велику кількість речей, від user-interface'a до управління масивами даних.

При розробці Cocoa додатків, насправді, можна використовувати кілька мов програмування, але рідна мова - Objective-C, який є розширенням ANSI C, з деякими синтаксичними і семантичними особливостями (на основі Smalltalk) для підтримки ООП. Крім того, ваш код може викликати функції визначені в non-Cocoa інтерфейсі, такі як бібліотеки BSD в / usr / include. Ви навіть можете змішувати C ++ код з Cocoa кодом і посилатися на цей скомпільований чудо в вашому файлі,.

Найбільш важливі бібліотеки Cocoa упаковані в два основних framework'a для кожної платформи: AppKit для Mac OS X і UIKit для IOS. Як і всі framework'i, вони містять не тільки динамічно доступні бібліотеки (а іноді і кілька версій бібліотек, необхідні для забезпечення зворотної сумісності), але і файли заголовків, API документацію, і пов'язаних з ними ресурси. Framework - це дуже важлива складова для будь-якого проекту під Mac або IOS.

Mac OS X підтримує до всього іншого ще багато корисних бібліотек, таких як: WebKit і Address Book frameworks, але про це пізніше.

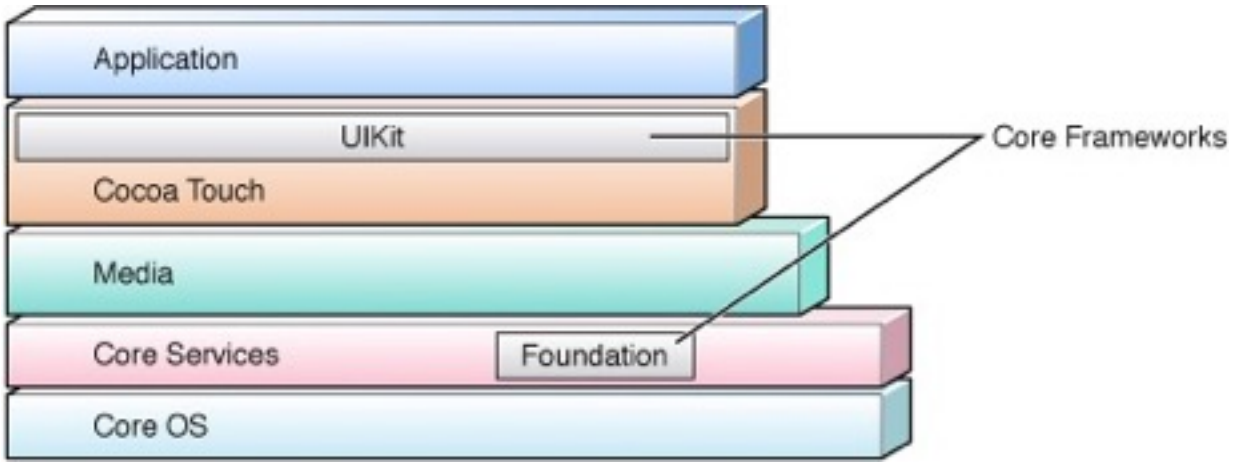

Рисунок 4.9 - Програмні шари iOS

Нижче наводиться короткий виклад деяких з framework'ов кожного шару: Core OS. Цей рівень містить ядро, файлову систему, мережевої інфраструктури, безпеку, управління живленням, а також ряд драйверів пристроїв. У ньому, до всього іншого, міститься бібліотека libSystem, яка підтримує POSIX / BSD 4.4 / C99 API специфікації і включає в себе системи на рівні API для багатьох сервісів.

Core Services. Надає основні сервіси, такі як маніпуляції з рядками, управління колекціями та мережевим взаємодією, управління контактами, настройками. Вони також дають можливість користуватися апаратними особливостями будови (GPS, компас, акселерометр і гіроскоп).

Приклади framework'a цього шару - Core Location, Core Motion, і System Configuration.

Цей шар включає в себе як Foundation так і Core Foundation, які пропонують деякі типи даних, такі як рядки і колекції.

Media. Забезпечує шару Cocoa Touch доступ до мультимедійних можливостей. Включає в себе Core Graphics, Core Text, OpenGLES,

Core Animation, AVFoundation, Core Audio, і сервіси відтворення відео.

Cocoa Touch. Займається безпосередньою підтримкою додатків. Містить такі компоненти як Game Kit, Map Kit і iAd.

Для того щоб інтегрувати Map Kit фреймворк в наш додаток для спортсменів які займаються бігом, нам знадобилось використвти Cocoa Touch. Рисунок 4.1.8.

Безпосередньо за увесь UI відповідає відповідає бібліотека UIKit за допомогою неї ми маємо можливість забезпечити створення графічного інтерфейсу та елементи керування. Він імплементує найнижчі класи які визначають базову поведінку елементів

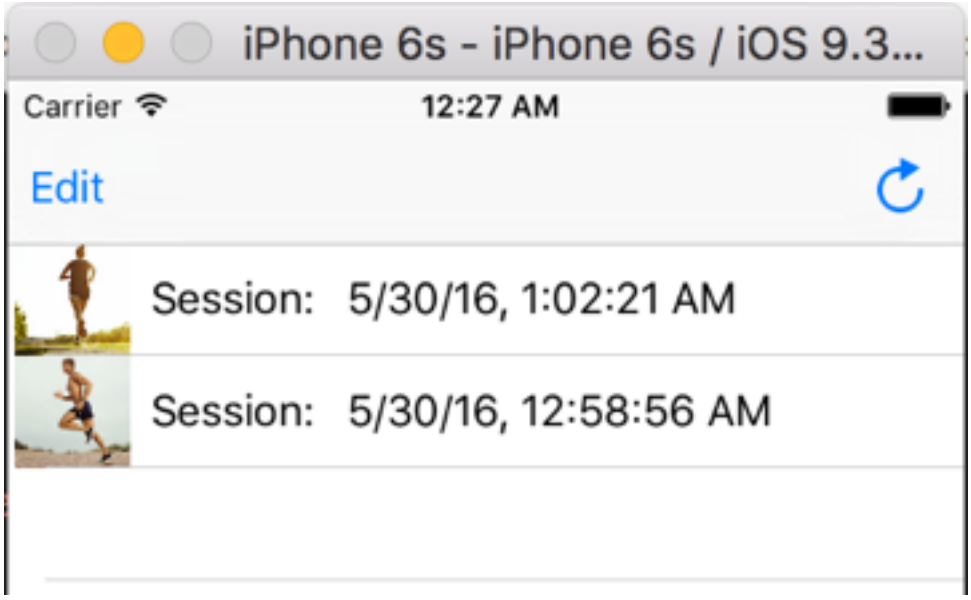

Рисунок 4.10 - Приклад елементів які забезпечує UIKit

На рисунку 4.3.2 ми можмо бачити фрагмент екрану iPhone під операційною системою iOS. На ньму усі елементи підконтрольні батьківскому фреймворку Cocoa Touch

4.3. Розробка структури додатку на базі MVC

Моде́ль-вигляд-контро́лер (або Модель-вид-контролер, англ. Model-view-controller, MVC) — архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення.

Цей шаблон поділяє систему на три частини: модель даних, вигляд даних та керування. Застосовується для відокремлення даних (модель) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача.

Бачення MVC компанією Apple, давайте подивимося на традиційну версію ми можемо розглянути на рисунку 4.3.

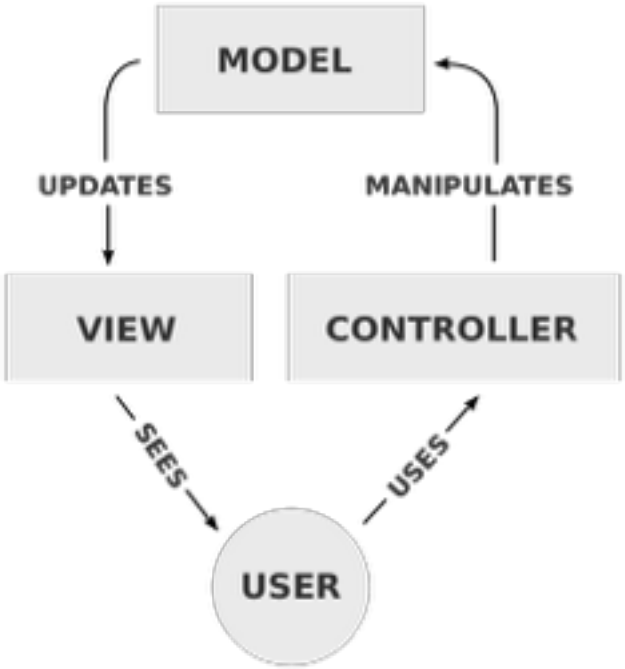

Рисунок 4.11 - Традиційне бачення MVC

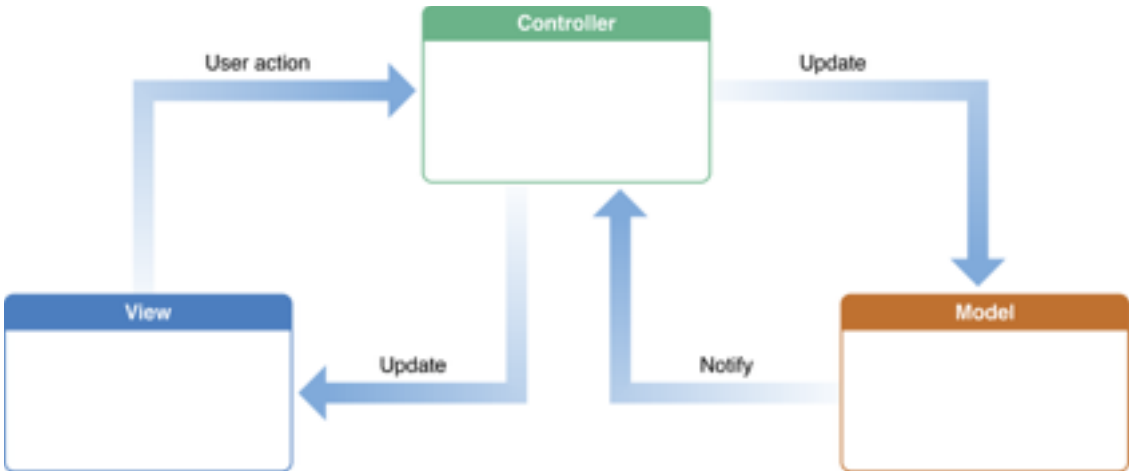

Рисунок 4.12 - Бачення MVC у компанії Apple

У традиційному MVC View не зберігає стану в собі. Controller просто «рендерить» View при змінах Model. Наприклад, веб-сторінка повністю перевантажується після того, як ви натиснете на посилання для переходу в інше місце. Хоча можна реалізувати традиційний MVC в середовищі iOS, це не має

особливого сенсу через архітектурну проблеми: все три сутності тісно пов'язані, кожна сутність знає про двох інших. Це сильно знижує можливість повторного використання кожного з елементів. З цієї причини ми не будемо навіть намагатися написати приклад канонічного MVC.

Традиційний MVC здається непридатним до сучасної iOS розробці.

Розглянемо тепер рисунок 4.3.2. Controller є посередником між View і Model, отже, дві останніх не знають про існування один одного. Тому Controller важко повторно використовувати, але це, в принципі, нас влаштовує, так як ми повинні мати місце для тієї хитрої бізнес-логіки, яка не вписується в Model.

В теорії все виглядає дуже просто, але ви відчуваєте, що щось не так, чи не так? Ви напевно чули, що люди розшифровують MVC як Massive View Controller. Крім того, розвантаження ViewController стала важливою темою для iOS-розробників. Чому це відбувається, якщо в Apple просто взяли традиційний MVC і трохи його поліпшили?

4.4. Структура бази даних

Для того щоб зберігати отриммані дані під час роботи додатку, на основі попереднього аналізу, було прийнято рішення використати системний фреймворк Core Data.

Core Data - один з найбільш потужних фреймворків в iOS. Фундаментально, це спосіб створення Swift (Objective-C) об'єктів як об'єктів, "відображених" в SQL або XML базах даних. Це свого роду "місток" між об'єктно-орієнтованої "територією" і "територією" баз даних. За базу даних домінує SQLite.

Для такого "відображення", як і у всякій іншій базі даних, ми створюємо в Xcode Модель Даних (Data Model) і там працюємо з:

Сутностями (Entities) - в "світі" баз даних це таблиці, в яких будуть "відображатися" наші Swift об'єкти,

Атрибутами (Attributes) - це колонки в таблиці бази даних. У нашому об'єктно-орієнтованому "світі" це відповідає властивостям (properties) об'єктів.

Взаємозв'язками (Relationships) - це також властивості об'єктів, але вони є покажчиками на інші об'єкти в базі даних, або покажчиками на ряд інших об'єктів, це щось типу "joins" між таблицями в базі даних.

Запитами як властивостей (Fetch properties) - це "обчислюється" спосіб отримати покажчик на деякі інші властивості. Якщо Взаємозв'язки посилаються безпосередньо на кінцеві об'єкти, то Запити посилаються на об'єкти, вибрані зазначеним предикатом.

Як нам отримати доступ до всього цього в нашому Swift коді після того, як ми створили Модель Даних (Data Model)? Відповідь полягає в тому, що нам потрібен контекст NSManagedObjectContext (для стислості надалі позначимо його MOC). Цей MOC в коді є "центральним простором" для створення об'єктів в базі даних, встановлення їх атрибутів і запитів до об'єктів. Все це ми будемо робити через MOC. Він же буде автоматично "відображати" наші дії в SQLite.

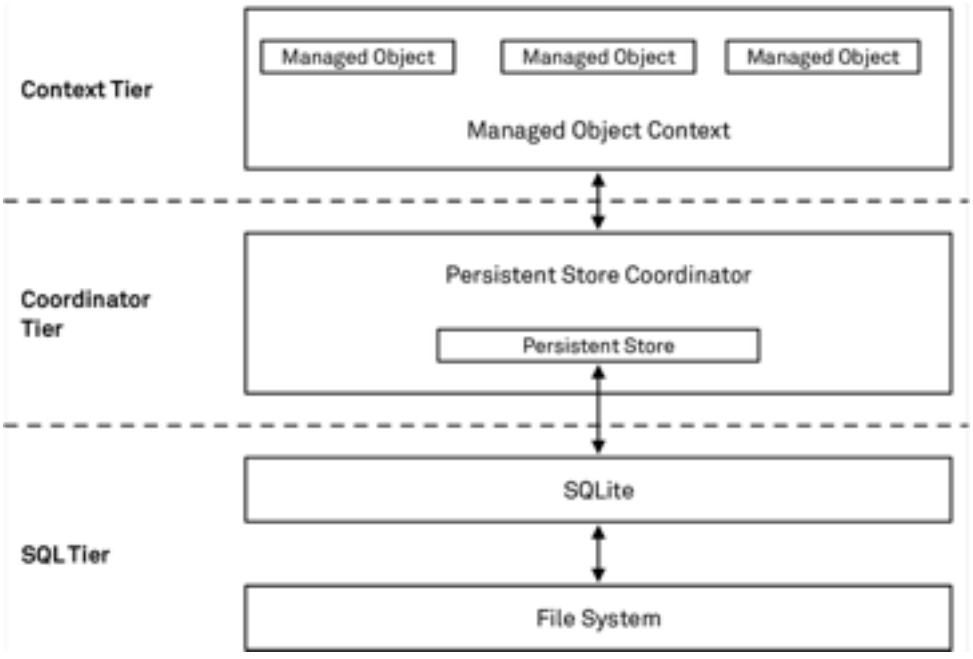

Рисунок 4.13 - Схематичне представлення архітектури Core Data

Звичайно ж Core Data не панацея і є інші варіанти зберігання даних, які можуть краще підійти при вирішенні певних завдань, але вже дуже добре і красиво Core Data вписується в Cocoa Touch. Більшість деталей по роботі зі сховищем даних Core Data приховує, дозволяючи вам сконцентруватися на тому, що дійсно робить ваше додаток, унікальним і зручним у використанні. Також в середовищі розробки xCode присутні зручні інструменти для роботи з фреймворком.

Не дивлячись на те, що Core Data може зберігати дані в реляційної базі даних на зразок SQLite, Core Data не є СУБД. По-правді Core Data в якості сховища може взагалі не використовувати реляційні бази даних. Core Data не є чимось на зразок Hibernate, хоча і надає деякі можливості ORM. Core Data швидше є оболонкою / фреймворком для роботи з даними, яка дозволяє працювати з сутностями і їх зв'язками (відносинами до інших об'єктами), атрибутами, в тому вигляді, який нагадує роботи з об'єктним графом в звичайному об'єктно-орієнтованому програмуванні.

Для того щоб зберігати дані під час початку сесії, було створено дві моделі. Session - описує сесію тренування користувача. Location - описує дані, які додаток збирає під час того, як користувач почав біг

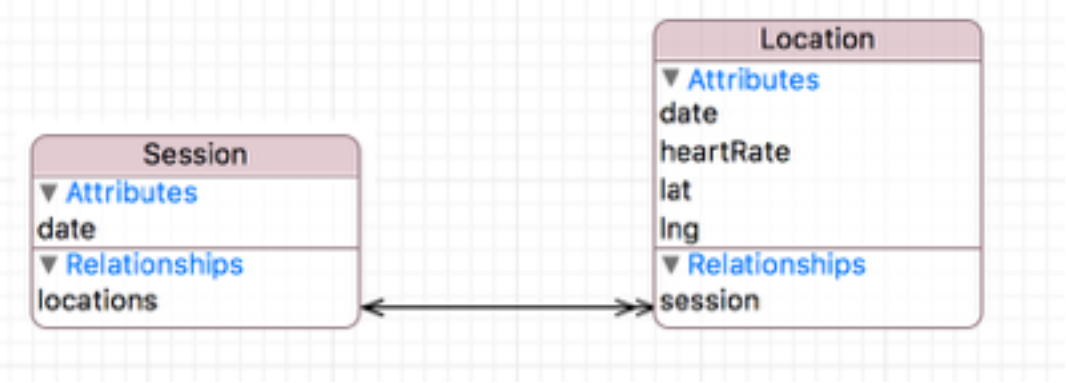

Рисунок 4.14 - Схема бази даних

Після того як користувач починає сесію, тобто “тренування”, створюється обєкт Session з поточною датою. Через кожні 5 секунд програма опитує систему на наявність нової партії обновлених статистичних даних від датчиків пульсу та локації, і зберігає їх в модель Location. Після того як користувач припиняє біг і всі збережені дані Locations додаються в масив locations мождeлі Session і зберігаються в локальне сховище. Тепер ці дані можуть бути використані для грфічного відображення або для подальшого аналізу.

4.5. Програмний інтерфейс додатку на iPhone та Apple Watch

Беручи до уваги результати попереднього аналізу з існуючих інструментів побудови графічного інтерфейсу, було прийнято рішення використовувати фреймворк AutoLayout.

У iOS 6 Apple представили чудову можливість для верстки UI для iOS-додатків - Auto Layout. Але ось що дивно, до цих пір дуже мало хто проекти використовують цю можливість. А адже це дуже сильний інструмент, якщо з розумом підійти до верстці UI, можна заощадити дуже багато часу на підстроювання елементів для 3,5 "і 4" екранів, портретно-ландшафтному розташуванні екрану і навіть на універсальної верстці для iPhone і iPad. І це все не рахуючи того, що скоро представлять iPhone 6 і ніхто досі точно не знає, яке там буде дозвіл і який екран. Краще б заздалегідь підстрахуватися.

В основному, тема Auto Layout досить проста, і вивчити її нескладно. Але особисто я зіткнувся з проблемою при розташуванні елементів в UIScrollView. Я витратив чимало часу і нервів на вивчення того, як же правильно розташувати елементи і вказати розмір контенту для того щоб ScrollView почав перегортувалися.

AutoLayout використовується для побудови динамічних призначених для користувача інтерфейсів, масштабованих і адаптуються до різних форматів і дозволами екранів пристроїв, а також їх орієнтаціям. AutoLayout прийшов на зміну системі «пружин і розтяжок» застосовується в попередніх версіях iOS

SDK. Також AutoLayout робить інтернаціоналізацію більш простим завданням, розміщувати текст змінної довжини на екрані стає простіше, також підтримуються мови з напрямком письма справа наліво, такі як іврит і арабська.

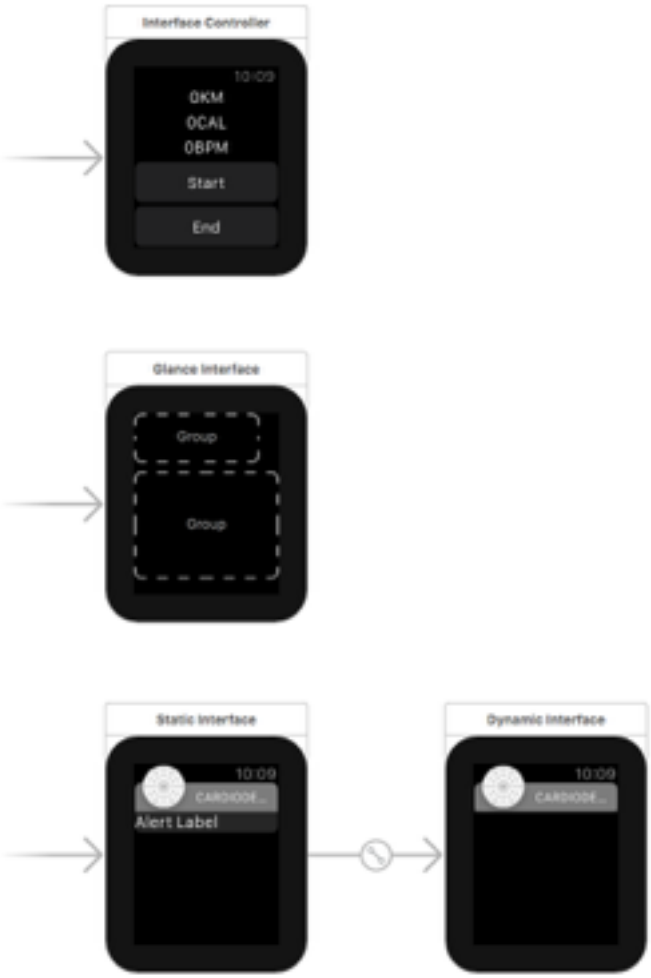

Рисунок 4.15 - Схема інтерфейсу Apple Watch додатку

Робота AutoLayout заснована на зв'язках (constraints), які встановлюють геометричні відносини між уявленнями призначеного для користувача інтерфейсу. Наприклад, ми можемо створити зв'язок яка говорить: «текстова мітка повинна бути закріплена, на деякій відстані від лівого краю батьківського уявлення і з'єднана з лівим краєм кнопки, з додавання між ними проміжку в 10 px».

AutoLayout бере задані зв'язку і математично обчислює ідеальні позиції і розміри для всіх уявлень. Нам не потрібно більше встановлювати розміри уявлень вручну, задавати їх координати розташування - AutoLayout бере цю роботу на себе.

Створювати зв'язку можна як програмно, так і за допомогою Interface Builder.

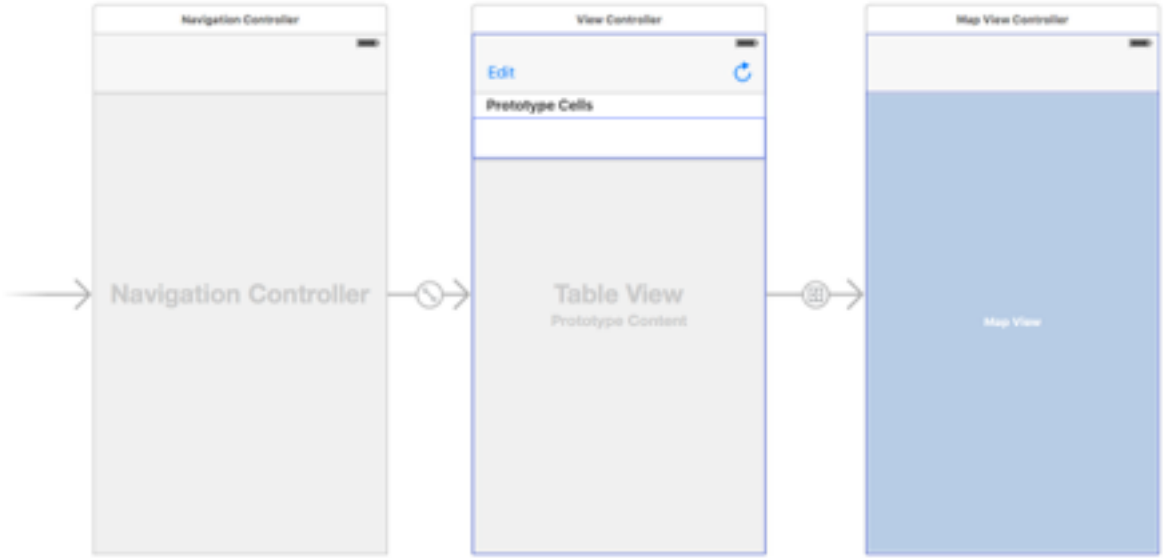

Рисунок 4.16 - Схема інтерфейсу iOS додатку

В основному, тема Auto Layout досить проста, і вивчити її нескладно. Але особисто я зіткнувся з проблемою при розташуванні елементів в UIScrollView. Я витратив чимало часу і нервів на вивчення того, як же правильно розташувати елементи і вказати розмір контенту для того щоб ScrollView почав перегортувалися.

Також AutoLayout робить інтернаціоналізацію більш простим завданням, розміщувати текст змінної довжини на екрані стає простіше, також підтримуються мови з напрямком письма справа наліво, такі як іврит і арабська.

4.6. Тестування та результати роботи програми

Для тестування результатів роботи додатку було використано смартфон iPhone 6 та годинник Apple watch 42мм. У процесі розробки додатку у нас виникали певні труднощі, переважно — під час імплементації дизайну. Ось детальніше про деякі з них.

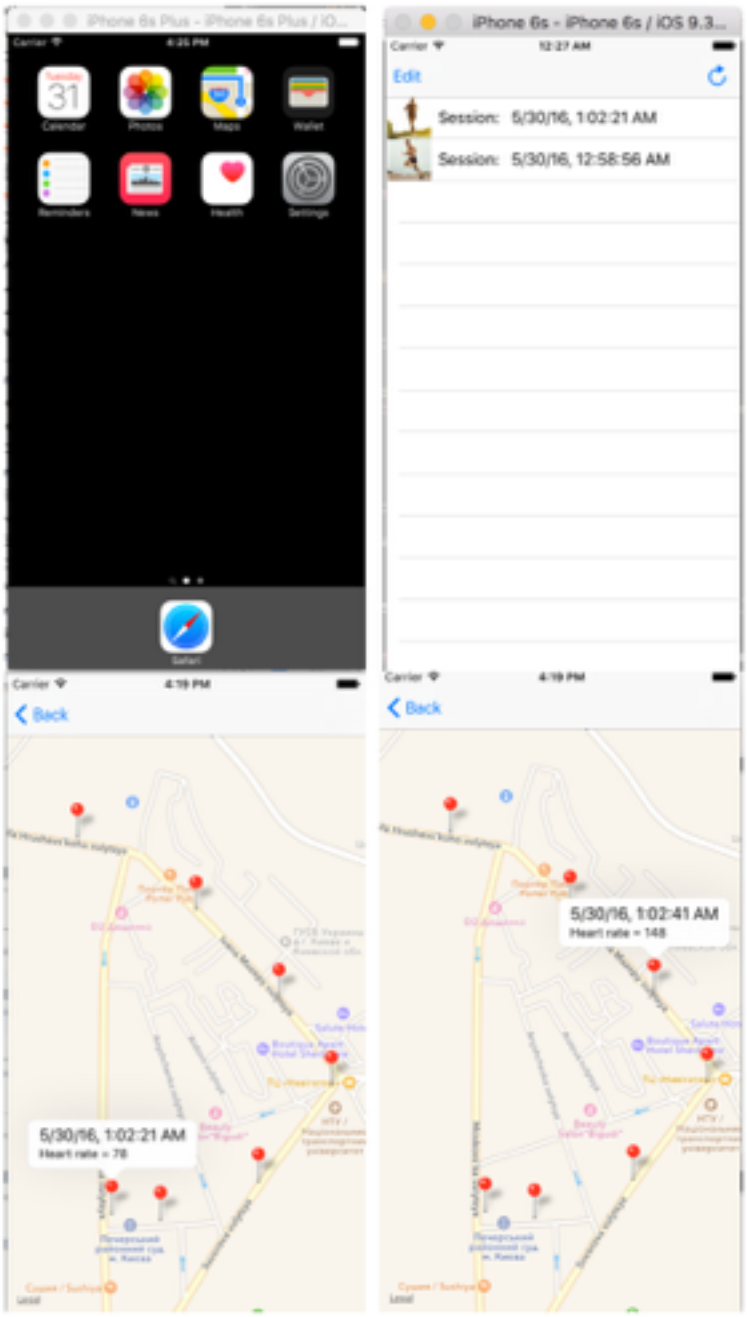

Рисунок 4.17 - Інтерфейс iOS додатку

У першу чергу, Apple Watch це пристрій для контролю здоров’я та активності людини. Я думаю, хто любить носити годинник, не відмовилися б від них

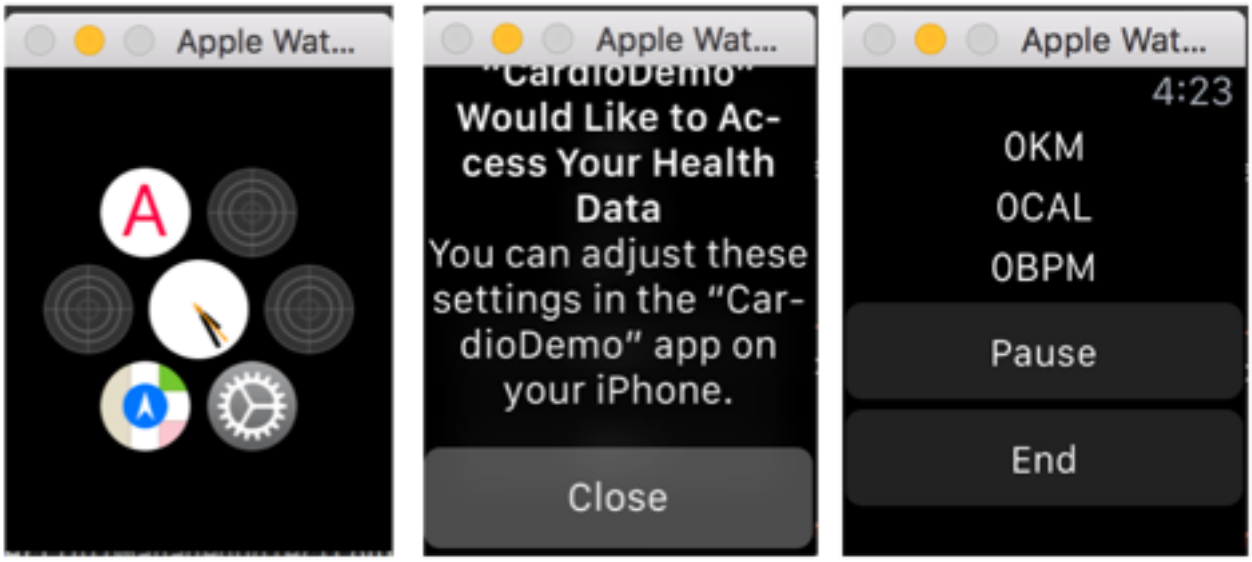

Рисунок 4.18 - Інтерфейс iOS додатку

Тобто, як бачимо, розумний годинник Apple без iPhone —працює корректно. І хоча Apple згадував про WatchKit Apps — повнофункційні додатки, які, теоретично, повинні працювати без iPhone, поки що вони недоступні, і ніхто не знає, коли з’являться на ринку.

З іншого боку цей істотний недолік — як би парадоксально це не звучало — є великим плюсом. Зважаючи на те, що додаток годинника, по суті, працює на iPhone, програміст має у своєму розпорядженні увесь функціонал і дані з самого iPhone (щоправда, Apple вже попередив нас про ймовірні обмеження на енергозатратні операції). Сам же WatchKit — це такий собі «місток», що з’єднує код на iPhone з інтерфейсом на Apple Watch.

4.7. Висновки

Враховуючи описані особливості цільової операційної системи, для створення додатку, що відповідає новим тенденціям в плані дизайну інтерфейсу та функціональності, були виконані наступні кроки: комбіноване використання засобів статичної та динамічної розмітки Auto Layout, робота з базою даних, використані стандартні методи зчитування даних пульсу і розвинута система захисту даних. Статично встановлюється розмітка компонентів, а динамічно - їх угруповання на екрані смартфона та годинника.

Способи інтерракції реалізовані з метою створення найкращого досвіду користувача. Навігація між частинами додатка виконується за допомогою використання шаблону Navigation від Apple. Результат відповідає уявленням про систему та узгоджений в технічній документації.

AutoLayout робить інтернаціоналізацію більш простим завданням, розміщувати текст змінної довжини на екрані стає простіше, також підтримуються мови з напрямком письма справа наліво, такі як іврит і арабська.

Взаємодія між девайсами відбувається за допомогою фреймворку Watch Connectivity і реалізована програмно через клас WCSession з пакету Foundation.

Програмне забезпечення мобільного додатку містить всі обумовлені в технічному завданні функції і знаходиться на ранній стадії відкритого тестування, щоб закрити ті помилки, які не виявили автоматичні тести. Весь код опублікований в публічному репозитарії в Github.

5. ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

У даному розділі проводиться оцінка основних характеристик програмного продукту, призначеного для виміру пульсу на розумних годиниках Apple Watch з відображенням на мобільному додатку на телефоні під керуванням операційної системи iOS. Інтерфейс користувача буде створений з використанням технології Auto-Layout.

Нижче наведено аналіз різних варіантів реалізації модулю з метою вибору оптимальної, з огляду при цьому як на економічні фактори, так і на характеристики продукту, що впливають на продуктивність роботи і на його сумісність з апаратним забезпеченням. Для цього було використано апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) – це технологія, яка дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. Як прямі, так і побічні витрати розподіляються по продуктам та послугам у залежності від потрібних на кожному етапі виробництва обсягів ресурсів. Виконані на цих етапах дії у контексті метода ФВА називаються функціями.

Мета ФВА полягає у забезпеченні правильного розподілу ресурсів, виділених на виробництво продукції або надання послуг, на прямі та непрямі витрати. У даному випадку – аналізу функцій програмного продукту й виявлення усіх витрат на реалізацію цих функцій.

Фактично цей метод працює за таким алгоритмом:

- визначається послідовність функцій, необхідних для виробництва продукту. Спочатку – всі можливі, потім вони розподіляються по двом групам: ті, що впливають на вартість продукту і ті, що не впливають. На цьому ж етапі оптимізується сама послідовність скороченням кроків, що не впливають на цінність і відповідно витрат.

після того, як для кожної функції будуть визначені їх джерела витрат, проводиться кінцевий розрахунок витрат на виробництво продукту.

5.1. Постановка задачі техніко-економічного аналізу

У роботі застосовується метод ФВА для проведення техніко-економічний аналізу розробки.

Відповідно цьому варто обирати і систему показників якості програмного продукту.

Технічні вимоги до продукту наступні:

- програмний продукт повинен функціонувати на розумних годиниках Apple Watch під керуванням WatchOS та додатку на iOS
- забезпечувати візуальний гарний інтерфейс та інтуітивну логіку додатку для користувача
- забезпечувати зручність і простоту взаємодії з користувачем;
- передбачати мінімальні витрати на впровадження програмного продукту.

5.1.1. Обґрунтування функцій програмного продукту

Головна функція F_0– розробка програмного продукту, який аналізує процес за вхідними даними та будує його модель для подальшого прогнозування. Виходячи з конкретної мети, можна виділити наступні основні функції ПП:

F_1 – вибір мови програмування;

F_2 – вибір оптимальної БД;

F_3 – інтерфейс користувача.

Кожна з основних функцій може мати декілька варіантів реалізації.

Функція F_1:

а) мова програмування Swift

б) мова програмування ObjectiveC;

Функція F_2:

а) CoreData;

б) Realm.

Функція F_3:

а) інтерфейс користувача, створений за технологією Auto-Layout

б) інтерфейс користувача, створений без технології Auto-Layout.

5.1.2. Варіанти реалізації основних функцій

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 4.1). На основі цієї карти побудовано позитивно-негативну матрицю варіантів основних функцій (таблиця 4.1).

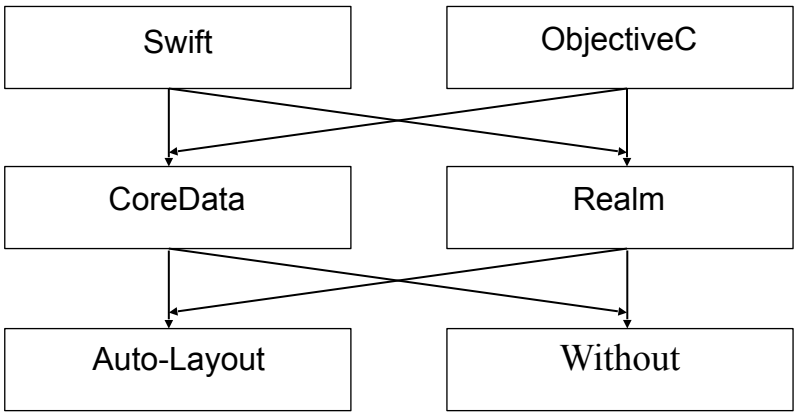

Рисунок 5.1 – Морфологічна карта

Морфологічна карта відображує всі можливі комбінації варіантів реалізації функцій, які складають повну множину варіантів ПП.

Таблиця 5.1 – Позитивно-негативна матриця

Основні функції	**Варіанти реалізації**	**Переваги**	**Недоліки**
F1	*А*	Займає менше коду, швидкість роботи, нова мова	Недостатньо бібліотек пряцює з цією мовою
	Б	Повна документація, більше людей працює з цією мовою	Низька швидкодія
F2	А	Працює по замовчуванню, безкоштовна	Не має можлівості підтримувати великі розміри баз даних
	Б	Нова швидка БД, крос платформена, працює на багатьох OS	Висока вартість, нова технологія
F3	*А*	Під різні діагоналі єкранів	Складна технологія
	Б	Простота створення	Багато коду для різних типів діагоналей єкранів

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Функція *F1:*

Вибір мови відіграє вжливу роль і обидва варіанти буде гарно порівнювати а) та б).

Функція *F2:*

Оскільки ми не будемо мати великіх обсягів даних, а програма буде працювати тільки під ОС iOS та WatchOS то вибір БД буде CoreData а).

Функція *F3:*

Оскільки кількість пристроїв з різними розмірами єкраниів у ОС iOS багото то Auto-Layout потрібно використовувати а).

Таким чином, будемо розглядати такі варіанти реалізації ПП:

1. F1а – F2а – F3а
2. F1б – F2а – F3а

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

5.2. Обґрунтування системи параметрів ПП

5.2.1. Опис параметрів

На підставі даних про основні функції, що повинен реалізувати програмний продукт, вимог до нього, визначаються основні параметри виробу, що будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

−X1 – швидкодія мови програмування;

−X2 – об’єм програмного коду;

−X3 – час компіляції коду;

−X4 – популярність серед програмістіви.

X1: Відображає швидкодію операцій залежно від обраної мови програмування.

X2: Показує розмір програмного коду який необхідно створити безпосередньо розробнику.

X3: Відображає час, який витрачається на дії при компіляції програми.

X4: Показує індекс популярності серед програмістів

5.2.2. Кількісна оцінка параметрів

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника та умов, що характеризують експлуатацію ПП як показано у табл. 5.2

Таблиця 5.2 – Основні параметри ПП

Назва Параметра	**Умовні позначення**	**Одиниці виміру**	**Значення параметра**		
			гірші	**середні**	**кращі**
Швидкодія мови програмування	X1	Оп/мс	2000	11000	19000
Об’єм програмного коду	X2	кількість строк коду	32	16	8
Час компіляції коду	X3	мс	800	420	60
Попуярність серед програмісів	X4	рейтинг інтересу до мови	2000	1500	1000

За даними таблиці 5.2 будуються графічні характеристики параметрів – рис. 5.2 – рис. 5.5.

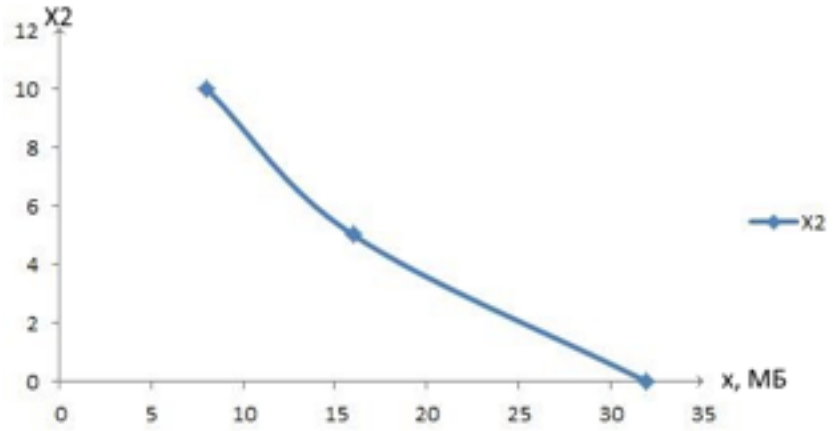

Рисунок 5.2 – X1, швидкодія мови програмування

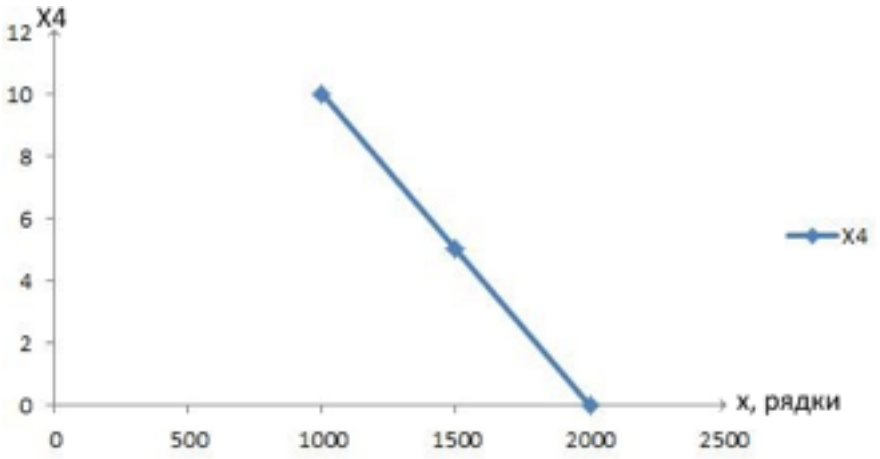

Рисунок 5.3 – X2, Об’єм програмного коду

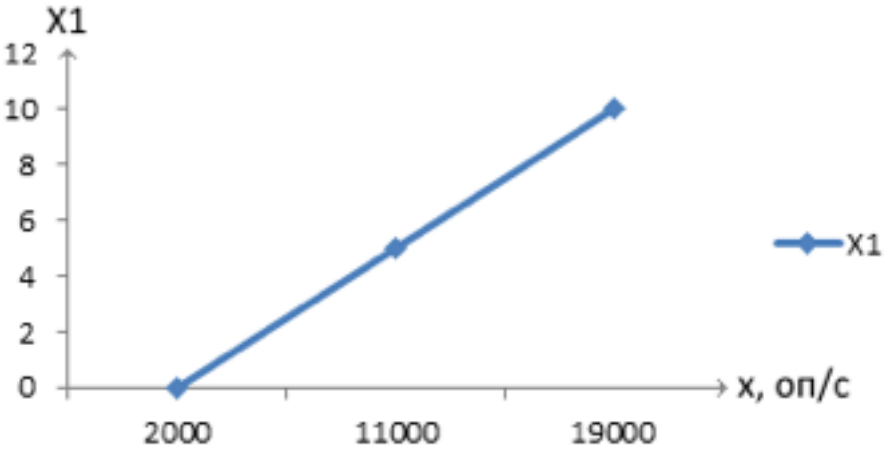

Рисунок 5.4 – X3, Час компіляції коду

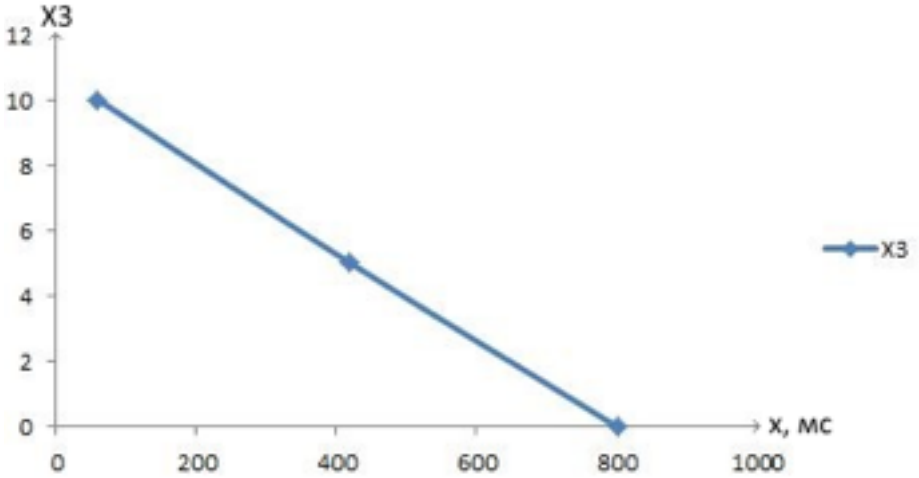

Рисунок 5.5 – X4, Попуярність серед програмісів

5.2.3. Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати при знаходженні параметрів моделей адаптивного прогнозування і обчислення прогнозних значень.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей.

Результати експертного ранжування наведені у таблиці 5.3.

Таблиця 5.3 – Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X1	Швидкодія мови програмування	мс	2	3	3	3	3	2	4	20	5	25
X2	Об’єм програмного коду	кількість строк коду	4	2	2	5	4	4	3	24	1	1
X3	Час компіляції коду	мс	3	4	4	2	3	3	2	21	4	16
X4	Популярність серед програмісів	рейтинг інтересу до язика	5	5	5	5	5	5	5	35	-10	100
	Разом		14	14	14	15	15	14	14	100	0	142
	Порахуємо коефіцієнт узгодженості:	0.579591836734694										

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

де N – число експертів, n – кількість параметрів;

б) середня сума рангів:

в) відхилення суми рангів кожного параметра від середньої суми рангів:

Сума відхилень по всім параметрам повинна дорівнювати 0;

г)загальна сума квадратів відхилення:

Порахуємо коефіцієнт узгодженості:

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0.58.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю5.4.

Таблиця 5.4 – Попарне порівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	>	<	<	>	>	<	<	<	0,5
X1 і X3	>	>	>	<	“="	<	<	<	0,5
X1 і X4	>	>	>	>	>	>	>	>	1,5
X2 і X3	<	>	>	<	<	<	<	<	0,5
X2 і X4	>	>	>	“="	>	>	>	>	1,5
X3 і X4	>	>	>	>	>	>	>	>	1,5

Числове значення, що визначає ступінь переваги i–го параметра над j–тим, a_{ij} визначається по формулі:

$$a_{ij} = \begin{cases} 1{,}5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j \\ 0.5 \text{ при } X_i < X_j \end{cases}$$

З отриманих числових оцінок переваги складемо матрицю $A = \| a_{ij} \|$.

Для кожного параметра зробимо розрахунок вагомості $K_{вi}$ за наступними формулами.

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%).На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

де .

Як видно з таблиці 5.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 5.5 – Розрахунок вагомості параметрів

Параметри	Параметри				Перша ітер.		Друга ітер.		Третя ітер	
	X1	X2	X3	X4						
X1	1,0	0,5	0,5	1,5	3,5	0,219	22,25	0,216	100	0,215
X2	1,5	1,0	0,5	1,5	4,5	0,281	27,25	0,282	124,25	0,283
X3	1,5	1,5	1,0	1,5	5,5	0,344	34,25	0,347	156	0,348
X4	0,5	0,5	0,5	1,0	2,5	0,156	14,25	0,155	64,75	0,154
Всього:					16	1	98	1	445	1

5.3. Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів *X2*(об'єм програмного коду) та *X1* (швидкодія мови програмування) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра *X3 (*Час компіляції коду) обрано не найгіршим (не максимальним), тобто це значення відповідає або варіанту а) 1000 мс або варіанту б) 800мс.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 5.6):

Таблиця 5.6 – Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1(X1)	А	150	3,6	0,215	0,774
F2(X2)	А	310	3,4	0,283	0,962
F3(X3,X4)	А	800	2,4	0,348	0,835
	Б	50	1	0,154	0,154

За даними з таблиці 4.6 за формулою визначаємо рівень якості кожного з варіантів:

$K_{K1} = 0,774 + 0,962 + 0,835 = 2,57$

$K_{K2} = 0,774 + 0,962 + 0,154 = 1,89$

Як видно з розрахунків, кращим є перший варіант, для якого коефіцієнт технічного рівня має найбільше значення.

5.4. Аналіз рівня якості варіантів реалізації функцій

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;

2. Розробка дизайну додатку;

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 1.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує накопичену інформацію та досвід працівника.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань. Загальна трудомісткість обчислюється як

$$T_O = T_P \cdot K_П \cdot K_{СК} \cdot K_М \cdot K_{СТ} \cdot K_{СТ.М}, \quad (5.1)$$

де T_P – трудомісткість розробки ПП;$K_П$ – поправочний коефіцієнт;$K_{СК}$ – коефіцієнт на складність вхідної інформації; $K_М$ – коефіцієнт рівня мови програмування;$K_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм; $K_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: T_P =40 людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $K_П$ = 1.7. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх семи завдань рівний 1: $K_{СК}$ = 1. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $K_{СТ}$ = 0.8. Тоді, за формулою 5.1, загальна трудомісткість програмування першого завдання дорівнює:

$T_1 = 40 \cdot 1.7 \cdot 0.8 = 54.4$ людино-днів.

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм першої групи складності, степінь новизни Б), тобто T_P =20 людино-днів, $K_П$ =0.9,$K_{СК}$ = 1,$K_{СТ}$ =0.8:

$T_2 = 20 \cdot 0.9 \cdot 0.8 = 14.44$ людино-днів.

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$T_I = (54.4 + 14.44 + 4.8 + 14.44) \cdot 8 = 704{,}64$ людино-годин;

$T_{II} = (54.4 + 14.44 + 6.91 + 14.44) \cdot 8 = 706.75$ людино-годин;

Найбільш високу трудомісткість має варіант II.

В розробці беруть участь програміст з окладом 36000 грн., та дизайнер з окладом 30000. Визначимо зарплату за годину за формулою:

*Сч = (36000 + 30000)/(22 * 8 * 2) = 187.5грн*

Тоді, розрахуємо заробітну плату за формулою

$$Сзп = Сч*Ті*Кд$$

де $С_ч$– величина погодинної оплати праці програміста;Ті – трудомісткість відповідного завдання; $К_Д$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

I. $С_{ЗП} = 187.5 \cdot 704{,}64 \cdot 1.2 = 158544$ грн.

II. $С_{ЗП} = 187.5 \cdot 706.75 \cdot 1.2 = 159018.75$ грн.

Відрахування на єдиний соціальний внесок в залежності від групи професійного ризику становить 22%:

I. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 158544*0.22 = 34879.68$ грн.

II. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 159018.75*0.22 = 34984.125$ грн.

Тепер визначимо витрати на оплату однієї машино-години. ($С_М$)

Так як одна ЕОМ обслуговує одного програміста з окладом *36000* грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$С_Г = 12 \cdot М \cdot К_З = 12*36000*0.2 = 86400$ грн.

З урахуванням додаткової заробітної плати:

$С_{ЗП} = С_Г \cdot (1 + К_З) = 86400*(1 + 0.2) = 103680$

Відрахування на єдиний соціальний внесок:

$С_{ВІД} = С_{ЗП} \cdot 0{,}2 = 103680 * 0.22 = 22809.6$ грн.

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 10000 грн.

$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.15 \cdot 0.25 \cdot 10000 = 2875$ грн.,

де $К_{ТМ}$– коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача; $К_А$– річна норма амортизації; $Ц_{ПР}$– договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.15 \cdot 10000 \cdot 0.05 = 575$ грн.,

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$Т_{ЕФ} = (Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = (365 – 104 – 8 – 16) \cdot 8 \cdot 0.9 = 1706.4$ годин,

де $Д_К$ – календарна кількість днів у році; $Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів; $Д_Р$ – кількість днів планових ремонтів устаткування; t –кількість робочих годин в день; $К_В$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 1706{,}4 \cdot 0.5 \cdot 0.2 \cdot 2.0218 = 345{,}00$ грн.,

де $N_С$ – середньо-споживча потужність приладу; $К_З$– коефіцієнтом зайнятості приладу; $Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$С_Н = Ц_{ПР} \cdot 0.67 = 10000 \cdot 0.67 = 6700$ грн.

Тоді, річні експлуатаційні витрати будуть:

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н$$

$С_{ЕКС} = 103680 + 22809.6 + 2875 + 575 + 345 + 6700 = 136984.6$ грн.

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$С_{М\text{-}Г} = С_{ЕКС} / Т_{ЕФ} = 136984.6 / 1706.4 = 80.37$ грн/час.

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$С_М = С_{М\text{-}Г} \cdot Т$$

I. $С_М = 80.37 * 704.64 = 56631.91$ грн.;

II. $С_М = 80.37 * 706.75 = 56801.49$ грн.;

Накладні витрати складають 67% від заробітної плати:

$$С_Н = С_{ЗП} \cdot 0{,}67$$

I. $С_Н = 158544 * 0.67 = 106224.48$ грн.;

II. $С_Н = 159018.75 * 0.67 = 106542.56$ грн.;

Отже, вартість розробки ПП за варіантами становить:

$$С_{ПП} = С_{ЗП} + С_{ВІД} + С_{М} + С_{Н}$$

I. $С_{ПП} = 158544 + 34879.68 + 56631.91 + 106224.48 = 356280.07$ грн.;

II. $С_{ПП} = 159018.75 + 34984.125 + 56801.49 + 106542.56 = 357346.92$ грн.;

5.5. Аналіз рівня якості варіантів реалізації функцій

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$К_{ТЕРj} = К_{Кj} / С_{Фj}$,

$К_{ТЕР1} = 2.57 / 356280.07 = 0.00000721$;

$К_{ТЕР2} = 1.89 / 357346.92 = 0.00000528$;

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР1} = 0,72 \cdot 10^{-5}$.

5.6. Висновки до розділу 5

В даному розділі проведено повний функціонально-вартісний аналіз ПП, який було розроблено в рамках дипломного проекту. Процес аналізу можна умовно розділити на дві частини.

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати при знаходженні параметрів моделей адаптивного прогнозування і обчислення прогнозних значень.

В першій з них проведено дослідження ПП з технічної точки зору: було визначено основні функції ПП та сформовано множину варіантів їх реалізації; на основі обчислених значень параметрів, а також експертних оцінок їх важливості було обчислено коефіцієнт технічного рівня, який і дав змогу визначити оптимальну з технічної точки зору альтернативу реалізації функцій ПП.

Другу частину ФВА присвячено вибору із альтернативних варіантів реалізації найбільш економічно обґрунтованого. Порівняння запропонованих

варіантів реалізації в рамках даної частини виконувалось за коефіцієнтом ефективності, для обчислення якого були обчислені такі допоміжні параметри, як трудомісткість, витрати на заробітну плату, накладні витрати.

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є перший варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості

$К_{ТЕР} = 0,72 \cdot 10^{-5}$.

Цей варіант реалізації програмного продукту має такі параметри:

- мова програмування – Swift;
- БД CoreData;
- інтерфейс користувача, створений за допомогою нового інструменту Auto-Layout.

Даний варіант інструментів дає змогу створити оптимальни і потужний додаток на новітній мові програмування, з натвною швидкою базою даних для оброблення результатів, а також з використанням системи Auto-Layout, для вирішення проблеми різних діагоналей єкранів смартфонів.

ВИСНОВКИ

Світ технологій розвивається безперервно і динамічно. З кожним роком з'являються нові концепції та їх реалізації, а існуючі розширюють свої функціональні характеристики і можливості. Не нова і мета цієї спіралі прогресу - зробити умови життя людини якомога комфортнішими і продуктивними. Однією з причин такого бурхливого зростання є широке поширення Internet і значне збільшення швидкості передачі даних. Як зазначають в Google, до 2008 року був «Інтернет людей», тепер настав «Інтернет речей», так як пристроїв, підключених до світової мережі стало більше ніж жителів планети. Завдяки цьому взаємодію їх з людиною стало можливо винести на зовсім інший рівень.

Розумний годинник – це нове слово у світі високотехнологічних гаджетів. Будучи як би продовженням Вашого смартфону вони можуть допомогти у самих несподіваних ситуаціях – скинути небажаний виклик, подивитися прогноз погоди, дізнатися температуру тіла та багато іншого. Дуже зручним є використання таких гаджетів у якості GPS-навігатора. На даний момент іде активна розробка і вдосконалення технології «Розумний» годинник. Незважаючи на це, процеси стандартизації та глобалізації почалися порівняно недавно й на даний час існує безліч однотипних рішень, але з різною реалізацією, зав'язаною на виробників та їх технології. Завдяки зацікавленості основних гравців ринку в розвитку системи «Розумний» годинник та інтеграції в неї своїх систем та сервісів, з'явилися зрушення у бік популяризації технології та спільна зацікавленість у ній як покупців, так і виробників.

На даний момент іде активна розробка і вдосконалення технології «Розумний» годинник. Незважаючи на це, процеси стандартизації та глобалізації почалися порівняно недавно й на даний час існує безліч однотипних рішень, але з різною реалізацією, зав'язаною на виробників та їх технології. Завдяки зацікавленості основних гравців ринку в розвитку системи «Розумний» годинник та інтеграції в неї своїх систем та сервісів, з'явилися зрушення у бік популяризації технології та спільна зацікавленість у ній як покупців, так і виробників.

Одним з основних гравців у цій сфері є компанія Apple з її рішенням - Apple Watch. У першу чергу, Apple Watch це пристрій для контролю здоров'я та активності людини. Я думаю, хто любить носити годинник, не відмовилися б від них. Великим плюсом для зайнятих людей буде і можливість переглядати повідомлення. Годинник показує нові sms і пошту, повідомлення із соціальних мереж, оповіщення від новинних програм. Усе це виводиться на екран за мить після появи на iPhone.

З появою Apple Watch у багатьох програмістів виникає логічне бажання запрограмувати що-небудь для розумного годинника. А якщо вже програмувати, то краще щось корисне. Керуючись саме такою логікою, в даній роботі було спроектовано фітнес додаток для людей, що займаються бігом. Було досліджено системні можливості платформи і способи збору статистичних даних про користувача, щоб надати користувачам простий і зрозумілий інтерфейс моніторингу за станом свого здотовя під час тренування з фізичними навантаженнями. Проведено аналіз та порівняння мов програмування Objective C та Swift, а також досліджено функціональні можливості системних фреймворків. Попри загальне враження, що Apple Watch пропонує широке поле для творчості, на жаль, поточні можливості розробки на емуляторі не збігаються із очікуваннями. Робити щось дійсно потрібне і функціональне для Apple Watch з поточною версією SDK недуже зручно. Тому для тих хто цікавиться даною розробкою рекомендовано мати справжній пристрій Apple Watch і бажано компюрер марки Apple.

У подальшому планується додати до системи і можливість експорту зібраних статистичних даних для подальшої обробки в інших системах. Наприклад відсилати результати тренувань своєму тренеру або лікарю.

ПЕРЕЛІК ПОСИЛАНЬ

[1]. Apple Inc, Understanding Auto Layout:

[Електронний ресурс] - Режим доступу:

https:/developer.apple.com/library/ios/ documentation/UserExperience/Conceptual/ AutolayoutPG/. – Дата доступу: 13.02.2013

[2]. Matthijs Hollemans, Auto-Layout in iOS7

[Електронний ресурс] - Режим доступу:

https:/www.raywenderlich.com/50317/beginning-auto-layout-tutorial-in-ios-7-part-1.– Дата доступу: 28.02.2015

[3]. Erica Sadun, Mobile Programming Series: iOS Auto Layout Demystified, Second Edition, p.41-44, 2013 – 32. 214

[4]. Разработка мобильных приложений

[Електронний ресурс] - Режим доступу: https://habrahabr.ru/company/mailru/blog/179113/– Дата доступу: 14.02.2015

[5] Ray Wanderlich Developing watchOS apps 2015. № 1. С. 65–77.

[6] Ive John Design Patterns. 2015. № 1. С. 165–176.

[7] Brucker P. Scheduling Algorithms. Springer-Verlag, 2001. 365 p.

[8] Brucker P., Knust S. Complex scheduling Springer-Verlag Berlin, Heidelberg, Germany, 2006.

[9] Queyranne M., Schultz A.S. Polyhedral approaches to machine scheduling // Preprint N 408/1994. Berlin: Technical University of Berlin, Department of Mathematics, 1994.– 61 p.

[10] Sevastianov S.V., Tchernykh I.D. Computer-Aided Way to Prove Theorems in Scheduling // Bilardi G., Italiano G.F., Pietracaptina Литература 222 A., Pucci G. (eds.) Proceedings of Sixth Annual European Symposium on Algorithms ESA'98, 24 – 26 august.– Venice, Italy: 1998.– SpringerVerlag, LNCS, V. 1461, 1998.– P. 502 – 513.

[11] Осадчий Дмитро, Auto-Layout как способ построения графических и мультимедийных интерфейсов для приложений операционной системы iOS / Осадчий Дмитрий. // 18-th International Conference SAIT 2016 Kyiv, Ukraine, May 30 – June 2, 2016 . Proceedings. - "IASA" NTUU "KPI". – 2015. – P. 397