

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЦЕНТРАЛЬНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ
УНІВЕРСИТЕТ
ЕКОНОМІЧНИЙ ФАКУЛЬТЕТ
КАФЕДРА ЕКОНОМІЧНОЇ ТЕОРІЇ, МАРКЕТИНГУ ТА
ЕКОНОМІЧНОЇ КІБЕРНЕТИКИ

На правах рукопису

Допущено до захисту
Завідувач кафедри ЕТМЕК
д.е.н., проф. Жовновач Р.І.

«_____» _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

за ступенем вищої освіти «Магістр»

на тему:

***«Розробка застосунку для вимірювання
показників фізичної активності
спортсменів»***

Виконав здобувач II курсу, групи МК-22м
спеціальності 051 — Економіка
(спеціалізація Економічна кібернетика)

Бойко Андрій Юрійович

Керівник: _____ д.е.н., доц. Вишневська В.А.
(вчений ступінь, звання, ПІБ)

Рецензент: _____ д.е.н, проф. Сибірцев В.В.
(вчений ступінь, звання, ПІБ)

Кропивницький — 2023

ПЛАН

	ВСТУП	7
РОЗДІЛ І	АНАЛІЗ ІНСТРУМЕНТІВ, МЕТОДІВ ТА ЗАСОБІВ РОЗРОБКИ ЗАСТОСУНКІВ ДЛЯ МОБІЛЬНИХ ПРИСТРОЇВ	12
1.1.	Аналіз поточної ситуації в розробці мобільних застосунків	12
1.2.	Підходи до організації розробки мобільних застосунків	22
1.3.	Основні обмеження та ризики при розробці мобільних застосунків	35
1.4.	Огляд існуючих рішень	39
РОЗДІЛ 2.	АНАЛІЗ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ ПРОГРАМНОГО ЗАСТОСУНКУ	43
2.1.	Використання аутсорсингу для розробки мобільного додатку. Аналіз та основні напрямки діяльності ОНІКС-СИСТЕМЗ	43
2.2.	Огляд технологій для розробки мобільного застосунку	50
2.3.	Огляд технологій для відстеження показників фізичної активності	57
2.4.	Постановка задачі	70
РОЗДІЛ 3	ПРАКТИЧНА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ	73
3.1.	Архітектура програмного застосунку та аналіз тенденцій у сфері	73
3.2.	Вибір технологій для реалізації програмного застосунку.	78
3.2.1.	Опис загальної архітектури програмного застосунку.	78
3.2.2.	Вибір засобів для розробки серверної частини.	81
3.2.3.	Вибір засобів для розробки клієнтської частини.	85
3.2.4.	Вибір бази даних та патернів її проектування.	90
3.3.	Реалізація програмного застосунку.	93

3.3.1.	Опис функціональної частини застосунку.	93
3.3.2.	Опис організації процесу роботи над застосунком.	95
3.4.	Оцінка ефективності нового рішення.	103
3.4.1.	Розрахунок витрат на розробку та подальшу підтримку мобільного застосунку.	103
3.4.2.	Розрахунок потенційного ефекту для користувача від використання мобільного застосунку.	109
3.4.3.	Розрахунок очікуваного прибутку від реалізації програмного застосунку.	111
	ВИСНОВКИ	116
	ЛІТЕРАТУРА	123
	ДОДАТКИ	138

ВСТУП

Розробка застосунку для вимірювання показників фізичної активності спортсменів є актуальною темою в сучасному спортивному світі. Пандемія COVID-19 підкреслила необхідність використання технологій для моніторингу та підтримки фізичного здоров'я на відстані. Застосунок може стати ефективним інструментом для тренерів та спортсменів у веденні контролю за тренуваннями та досягненням цілей. Зокрема, врахування індивідуальних характеристик атлетів дозволить персоналізувати підходи до тренувань, оптимізувати навантаження та підвищити ефективність тренувального процесу. Розвиток подібних застосунків також сприятиме взаємодії між тренерами та спортсменами, підвищуючи якість комунікації та сприяючи спільному досягненню спортивних цілей.

Застосунок для вимірювання фізичної активності також може вплинути на популяризацію здорового способу життя серед широкого загалу. З огляду на зростання інтересу до фітнесу та активного відпочинку, створення зручного та доступного інструменту для відстеження фізичної активності стає ключовим елементом популяризації здорового способу життя. Технологічні рішення такого роду також можуть виявитися корисними для медичної діагностики та реабілітації, дозволяючи лікарям моніторити пацієнтів під час відновлювальних процесів.

Наприклад, інтеграція різноманітних датчиків та алгоритмів аналізу даних в застосунок може допомогти вченим та науковцям збирати інформацію для досліджень у галузі спортивної науки та фізіології. Такий підхід дозволяє не лише оптимізувати тренувальні програми, але і сприяє розвитку нових наукових підходів до підтримки фізичної форми та здоров'я спортсменів. Усе це робить тему розробки застосунку для вимірювання показників фізичної активності спортсменів не лише актуальною, але і перспективною для подальших досліджень та реалізації в практиці.

Мета даної роботи - розробка застосунку для вимірювання показників фізичної активності спортсменів - полягає в створенні зручного та ефективного інструменту, який дозволить тренерам та спортсменам відстежувати та аналізувати рівень фізичної активності.

Основні завдання роботи включають:

- Аналіз інструментів, методів та засобів розробки застосунків для мобільних пристроїв;
- А
аналіз технологій для реалізації програмного застосунку;
- С
творення плану для практичної реалізації застосунку.

Об'єктом дослідження є процес розробки мобільного застосунку для вимірювання фізичної активності спортсменів.

Предметом дослідження є методи розробки мобільного програмного забезпечення та методології організації процесу розробки.

Найбільш вагомими результатами магістерської роботи, що містять наукову новизну та практичну цінність, полягають у розробці мобільного застосунку для вимірювання показників фізичної активності спортсменів, який на відміну від існуючих рішень дозволяє використовувати саме SLeSS - гнучкий підхід, який інтегрує Scrum і Lean Six Sigma (LSS) як основу для організації власного процесу. В процесі дослідження нами було запропоновано загальний опис організації процесу роботи над продуктом для його ефективної реалізації.

Перший розділ роботи присвячений аналізу інструментів, методів та засобів розробки застосунків для мобільних пристроїв, який пропонуємо провести в наступному порядку: здійснити аналіз поточної ситуації в розробці мобільних застосунків; розглянути існуючі підходи до організації розробки мобільних застосунків; проаналізувати основні обмеження та ризики при розробці мобільних застосунків; здійснити огляд існуючих рішень.

Другий розділ роботи - це аналіз технологій для реалізації програмного застосунку, який ми пропонуємо провести наступним чином: вивчити та

проаналізувати використання аутсорсингу для розробки мобільного додатку, а також аналіз та основні напрямки діяльності ОНІКС-СИСТЕМЗ; провести огляд технологій для розробки мобільного застосунку; та власне здійснити постановку задачі кваліфікаційної роботи.

Третій розділ роботи присвячений практичній реалізації застосунку. Він складається з чотирьох логічних частин. Спочатку розберемо архітектуру програмного застосунку та проведемо аналіз тенденцій у сфері. Далі розглянемо вибір технологій для реалізації програмного застосунку. Ця частина роботи складається з опису загальної архітектури програмного застосунку, вибору засобів для розробки серверної частини, вибору засобів для розробки клієнтської частини та вибору бази даних та патернів її проєктування. Наступним кроком ми детально розглянемо питання реалізації програмного застосунку, де зупинимось на описі функціональної частини застосунку та описі організації процесу роботи над застосунком. І заключна частина третього розділу роботи полягає в оцінці ефективності нового рішення. Тут ми робимо розрахунок витрат на розробку та подальшу підтримку мобільного застосунку, розрахунок потенційного ефекту для користувача від використання мобільного застосунку та розрахунок очікуваного прибутку від реалізації програмного застосунку.

Вважаємо такий підхід до розкриття теми дослідження дозволить отримати наступні практичні результати:

- О
 оптимізація тренувального процесу: Застосунок має спростити та покращити процес планування та моніторингу тренувань, дозволяючи тренерам ефективно керувати та адаптувати навантаження відповідно до показників фізичної активності спортсменів.

- П
 персоналізація тренувальних програм: Застосунок зможе враховувати індивідуальні характеристики кожного спортсмена, щоб надати персоналізовані рекомендації та оптимальні тренувальні плани для досягнення поставлених цілей.

●
моніторинг здоров'я та уникнення перенавантажень: Застосунок допоможе у вчасному виявленні ознак перенавантаження або травм, сприяючи збереженню фізичного здоров'я спортсменів.

- Поліпшення комунікації між тренерами та спортсменами: Застосунок може служити як зручний засіб для обміну інформацією між тренерами та спортсменами, сприяючи ефективній комунікації та взаєморозумінню в плані тренувань.
- Внесок у наукові дослідження: Застосунок може забезпечити додаткові дані для наукових досліджень у галузі спортивної науки, допомагаючи розширити знання про фізичну активність та оптимізацію тренувань.

Отже, розробка такого застосунку вирішує конкретні завдання та сприяє подальшому розвитку спортивного тренування та наукових досліджень у цій області. Методологічною основою дослідження є детальне вивчення літератури про існуючі дослідження та розробки в області моніторингу фізичної активності, технологій вимірювання, інформаційних систем у спорті та суміжних галузях; на етапі проєктування - розробка методів та критеріїв для етапу проєктування застосунку, включаючи визначення основних функціональностей, вибір технічних засобів (датчики, платформи), та визначення структури та інтерфейсу застосунку.

Основними джерелами для виконання роботи є актуальні наукові статті в журналах, що спеціалізуються на інформаційних технологіях, менеджменті, фізіології, спортивних науках, технологіях в спорті та розробці застосунків для спорту; книги та підручники з області спортивної науки, програмування мобільних додатків, технологій вимірювання фізичної активності; оглядові статті та дослідження, які розглядають тенденції в розробці застосунків для фізичної активності та спорту; матеріали та доповіді, представлені на конференціях, присвячених спортивним технологіям та розробці застосунків; опитування та інтерв'ю зі спортсменами, тренерами, технічними експертами;

відгуки користувачів аналогічних застосунків та технічних рішень; електронні ресурси та інтернет-публікації. Ці джерела дають змогу отримати ширший розгляд та глибше розуміння теми кваліфікаційної роботи, враховувати актуальність та авторитетність джерел, щоб забезпечити науковий та обґрунтований характер дослідження.

РОЗДІЛ І

АНАЛІЗ ІНСТРУМЕНТІВ, МЕТОДІВ ТА ЗАСОБІВ РОЗРОБКИ ЗАСТОСУНКІВ ДЛЯ МОБІЛЬНИХ ПРИСТРОЇВ

1.1. Аналіз поточної ситуації в розробці мобільних застосунків

В індустрії інформаційних технологій різноманітне програмне забезпечення змінювало наш погляд на речі з самого початку своєї ери. Ми бачимо, що майже всі процеси, які ми виконували вручну, тепер виконуються та контролюються системами. Розробка програмного забезпечення – це галузь, яка входить до цієї материнської групи, і є загальним терміном, який використовується для розробки та розгортання програмного забезпечення, яке належить до категорії веб-сайтів і автономних програм. Під час створення цих програм мінімальні вимоги до апаратного забезпечення зазвичай стосувалися персонального комп'ютера з відповідними специфікаціями. Зараз, попри те, що багато хто має доступ до ноутбуків і комп'ютерів, кількість залишається значно нижчою порівняно з населенням. Після появи мобільних телефонів, які зазвичай являли собою портативні комп'ютери, кількість людей, які почали купувати мобільні телефони, значно зросла, а отже зріс і потенційний ринок користувачів різноманітних програм. Сьогодні ми можемо сказати, що незалежно від того, до якої вікової групи вони належать, люди вчаться використовувати програми як такі у вигляді мобільного додатку. Тому дана індустрія наразі й розвивається шаленими темпами.

Мобільні додатки були визначені як найкращий маркетинговий інструмент [1] і основна рекламна тактика [2] для залучення бізнесу. Вони дають великий потенціал для залучення клієнтів завдяки специфічним характеристикам (наприклад, яскравості, новизні та вбудованим функціям) [3], підтримці взаємодії один-до-одного та один-до-багатьох та полегшення обміну без обмежень за часом або місцем [4]. По суті, програми перетворюють комунікаційні зусилля на інтерактивний досвід клієнтів, посилюючи когнітивні,

емоційні та поведінкові реакції [3]. Наприклад, додатки підтримують діяльність, що створює цінність, як-от здійснення покупок і доступ до інформації [5]. Відповідно, додатки пропонують компаніям численні можливості для досягнення маркетингових цілей, впливаючи на шлях клієнта та формуючи його [6]. Загалом, додатки також дозволяють компаніям реалізувати цифрову орієнтацію на клієнта та досягти конкурентних переваг завдяки забезпеченню чудового досвіду для клієнтів [7].

Протягом останнього десятиліття популярність додатків продовжувала зростати (наразі доступно понад 2,87 мільйона додатків, [8]), і, хоча зростання додатків поступово сповільнилося, вони залишаються в центрі стратегій цифрового маркетингу, впливаючи на економіку у всьому світі [9]. Наприклад, у США на додатки припадає близько 60% споживання цифрових медіа [10], а 90% із 100 найкращих світових брендів пропонують один або кілька додатків [11]. Додатки також дають значні економічні результати завдяки досягненню тривалої експозиції в ЗМІ та споживчих витрат. Наприклад, додаток TikTok генерує понад один мільярд переглядів відео щодня [12] і залучив 50 мільйонів доларів споживчих витрат минулого року на додаток до доходів від реклами [11]. Глобальна медична та фінансова криза, спричинена пандемією COVID-19, додатково ілюструє ключову роль, яку відіграють додатки у сприянні виживанню бізнесу та відновленню клієнтського досвіду. Яскравий приклад додатка Zoom, який у 2020 році приніс дохід у 2,6 мільярда доларів США (Sensortower 2020) [13].

Розробка програм для мобільних пристроїв — це процес розробки програмного забезпечення, яке в основному працює на портативному цифровому пристрої, наприклад мобільному телефоні. Це галузь, що постійно розвивається, і тепер вона є невіддільною частиною будь-якого бізнесу чи індустрії, про які ми можемо подумати. Незважаючи на те, що на ринку доступні різноманітні мобільні телефони, Android та iOS займають основну частину. У таблиці 1.1. наведено розподіл світового ринку мобільних пристроїв за операційними системами (ОС) встановленими на них:

Таблиця 1.1.

Розподіл світового ринку мобільних пристроїв за операційними системами (ОС)

№	Мобільна операційна система	Частка ринку
1	Android	71.81%
2	iOS	27.43%
3	Samsung	0.38%
4	KaiOS	0.14%
5	Інші	0.14%
6	Linux	0.02%

На такому ринку люди зазвичай шукають розробки мобільних додатків для Android та iOS, що зараз становить 99,24% світового ринку. Незважаючи на те, що Android з великим відривом має перевагу по всьому світу, iOS лідирує в США з 59,97 % [14]. А зважаючи на велику привабливість американського ринку мобільних додатків (висока платоспроможність та низький поріг здійснення покупки у клієнтів) мобільні додатки на iOS продовжують бути не менш популярними, ніж на Android (зі сторони привабливості інвестування в них). Таким чином, галузь розробки додатків в основному зосереджена на цих двох операційних системах. Розробка “традиційного” програмного забезпечення все ще життєздатна, проте розробка додатків доповнює їх, надаючи функції, які типовий настільний або веб-додаток може не забезпечити. Наприклад, наявність GPS, компаса, акселерометра, датчиків, Bluetooth тощо робить мобільний телефон набагато потужнішим пристроєм для створення програм, ніж ПК.

У 2018 році цей ринок оцінювався в 106,27 мільярда доларів, а до 2026 року очікується, що він досягне 407,31 мільярда доларів [15]. Окрім прибутку, отриманого через програму, також потрібно говорити про галузь розробки. Плата за розробку типової програми залежить від багатьох факторів, і не може

мати фіксованої ціни. Однак ми можемо приблизно оцінювати вартість розробки для типового простого додатка в діапазоні від 15 до 25 тисяч доларів США, а для середнього складного додатка – від 40 до 100 тисяч доларів США.

Кількість мобільних додатків, які зараз представлені на ринку, зростає з кожним днем, і кількість людей, які завантажують додатки на різні пристрої, також зростає експоненціально [16]. Це тому, що зараз середньостатистичний дорослий настільки сильно прив'язаний до мобільного телефону, яким володіє. Дослідження ринків вказують на те, що індустрія мобільних додатків продовжуватиме зростати, і в осяжному майбутньому цей показник лише збільшуватиметься [17]. Нижче наведено графік, який показує завантаження мобільних додатків у всьому світі з 2016 по 2020 рік. Цей графік показує, що цифра в мільярдах просто зростає з року в рік [18].

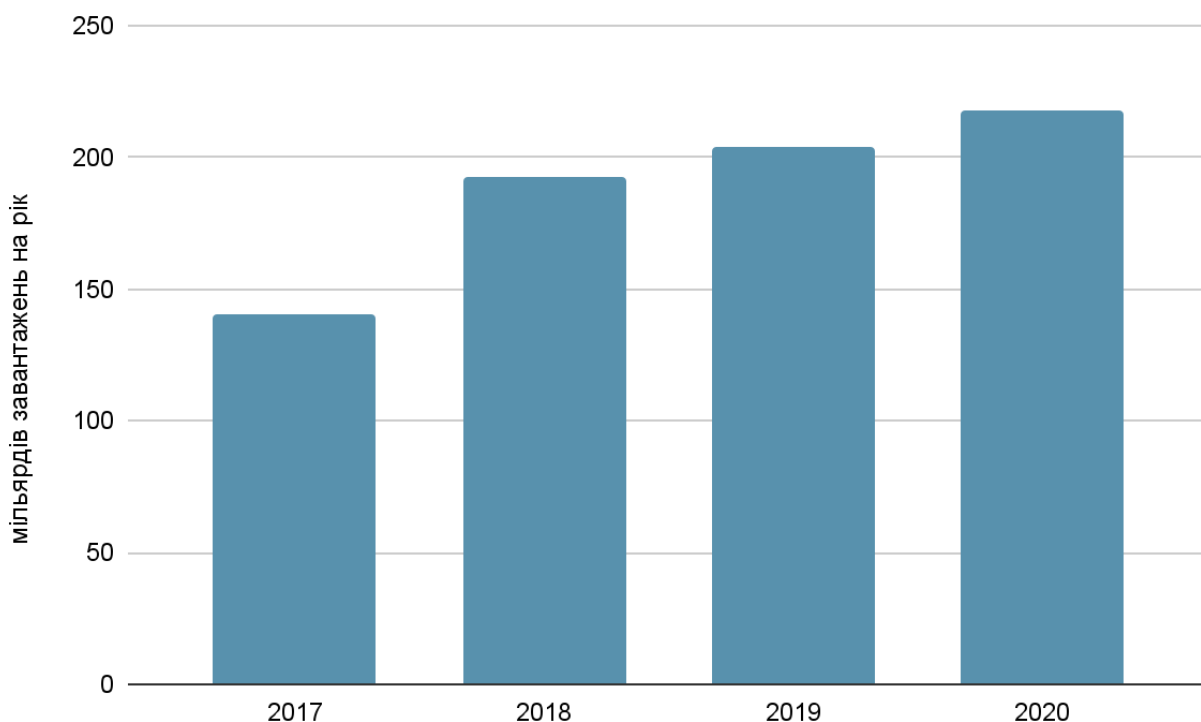


Рис. 1.1. Завантаження мобільних додатків у всьому світі з 2016 по 2020 рік.

Переважає більшість додатків доступна для безкоштовного завантаження в обох основних магазинах (App Store та Play Store), що змушує користувачів вагатися платити за завантаження додатків [19]. Використання мобільних

програм почало набирати популярності в усьому світі завдяки додаткам для спілкування в соціальних мережах і чатах.

Середній користувач смартфона зазвичай проводить близько трьох годин за телефоном, з яких дев'яносто відсотків часу припадає на програми телефону. Незважаючи на те, що на середньостатистичному телефоні встановлено понад 50 додатків, небагато з них часто використовуються. Тому те, що хтось завантажив програму, не означає, що користувач буде її використовувати. Щоб переконатися, що програма має вищий рівень утримання, розробник повинен мати можливість змусити користувача повернутися до програми та продовжувати використовувати її протягом значного періоду часу. Тому те, що люди завантажують програму, не означає, що вона чудова й існуватиме довго. Додаток має бути на такому рівні, щоб користувач продовжував використовувати його та надавати зворотний зв'язок розробникам, аби додаток міг продовжувати розвиватися та використовуватися.

Проте також слід відзначити, що розробка мобільних додатків в більшості своїй це все-таки бізнес складова. Тому основною метою розробки додатків та інвестування в них є або отримання прибутку з цього, або обслуговування певного суспільства, організації чи особи. У більшості випадків ціллю стає найперше - отримання доходу. Існують різні підходи до того, як організувати процес отримання прибутку з мобільного додатку (монетизація мобільного додатку). Для того, щоб визначити, яка модель монетизації буде оптимальною, дуже важливо знати ринок користувачів, адже так можна застосувати відповідні маркетингові зусилля, а також легко спрогнозувати дохід, який можна отримати [20]. Пропонуємо коротко розглянути основні моделі монетизації для подальшого розуміння контексту.

1. Реклама в додатку. Це модель, яка найчастіше використовуються майже в усіх безкоштовних програмах на ринку. Це спосіб, у який розробник отримує гроші за вбудовування реклами в додаток. Для цього розробнику потрібно використати SDK, наданий постачальником реклами, і зазвичай це дуже простий процес. Однак для того, щоб ця модель працювала,

першочерговою вимогою є наявність величезної кількості баз користувачів. В іншому випадку дохід, отриманий за допомогою цієї моделі, може бути мізерним. Перевага збереження реклами в програмі полягає в тому, що програма назавжди залишається безкоштовною для всіх. Проте, якщо реклама переважає функціональність програми, незабаром застосунок в цілому може втратити свою цінність для користувача, або ж він може почати шукати аналоги йому.

2. Покупки в програмі. Це спосіб, за допомогою якого більшість безкоштовних ігор приносять дохід за поточним сценарієм. Ігри надаються безкоштовно на ринку, і коли користувачі занурюються в гру, настає важкий поворот, який користувачам буде важко подолати, якщо вони не купують деякі витратні матеріали, як-от монети чи бали, щоб продовжити далі. Подібним чином розробники пропонують уявлення про те, як працюватиме їхній додаток, якщо ви готові платити за продемонстровані функції. Покупки в програмі бувають трьох основних типів: придбання витратних матеріалів, безкоштовні моделі та послуги за підпискою.

3. Монетизація даних. Більшість технологічних гігантів зараз отримують дохід від монетизації даних. Іншими словами - використання даних користувачів. Незважаючи на те, що це не звучить чудово чи етично, це справді законний спосіб отримання доходу. Можливо, програма збирає деяку інформацію, яка може мати велике значення для установ, що займаються дослідженнями чи маркетингом. Зібрані дані можуть бути використані для надання кращих послуг користувачам у майбутньому тощо. Зазвичай розробники збирають таку інформацію, як пристрій, на якому використовується програма, місцезнаходження, мережа, запитання, IP-адреса тощо.

4. Краудфандинг. Дуже рідко додаток фінансується за рахунок пожертвувань. Є кілька веб-сайтів краудфандингу, таких як Kickstarter, Indiegogo тощо, на яких розробники можуть висловити свою ідею щодо проєкту та почати кампанію з краудфандингу цього проєкту. Люди, які цікавляться новітніми технологіями та вважають ваш проєкт інноваційним, можуть об'єднатися, щоб підтримати. Рекомендується, лише якщо у вас є чудова ідея,

яку, на вашу думку, люди підтримують, і у вашому розпорядженні немає інших засобів отримання інвестицій чи стартового капіталу.

5. Партнерський маркетинг. Це процес розміщення реклами іншої програми у вашій та отримання комісії за кожен клік по цій рекламі, що перетворюється на продажі. В такому разі фіксується контактна інформація зацікавлених користувачів, які погоджуються поділитися нею. Щоб це працювало, обидві програми мають бути націлені на одну й ту саму аудиторію. Не обов'язково рекламувати лише одну конкретну програму, це може бути кілька програм, які належать до однієї категорії. Так само, як і у випадку з рекламою в додатку, неможливо передбачити дохід, який можна отримати таким чином.

6. Спонсорство. Подібно до того, як деякі події спонсоруються певними компаніями, додатки також можуть бути спонсорованими, щоб мати можливість розвиватися та в цілому існувати. У відповідь на подібне фінансування нерідко певні пропозиції чи інформація, пов'язані зі спонсором, мають бути інтегровані в додаток, щоб вигода була обопільною. У більшості випадків це буде компанія, яка тісно займається цією категорією програм чи має бізнес в тій самій дотичній ніші.

7. Платні програми. Незважаючи на те, що переважну більшість додатків на ринку можна завантажити безкоштовно, деякі додатки є програмами, які оплачуються заздалегідь, і вам потрібно сплатити суму, перш ніж навіть встановити програму. В такому разі всі функції пропонуються користувачеві без будь-якої реклами чи подальших перерв. Незважаючи на те, що це дуже простий і зрозумілий спосіб отримання прибутку, користувач повинен переконатися, що сума, яку він платить, щоб отримати певну функцію, того варта.

Отже, ми розглянули базові моделі отримання прибутку з готового мобільного додатку. Також слід розглянути й зворотну сторону - вартість створення подібних мобільних додатків. Розрахунок вартості розробки програми є однією з найважливіших частин планування проєкту в цілому. Він залежить від багатьох факторів. Однак існує явна різниця у вартості, коли окрема особа або фрілансер створює програму, порівняно з використанням для цього великих

компаній (наприклад, через модель аутсорсингу) [21]. Обидва мають свої переваги та недоліки, і вибір слід робити на основі контексту самої програми та клієнта.

Неможливо розрахувати точну суму вартості розробки програми. Скоріше можна скласти лише оцінку на основі наданих вимог. Існує кілька основних факторів, які необхідно враховувати перед тим, як визначати вартість [22]. Ці характеристики визначають цінність програми [23]:

- функціональність/призначення програми;
- нативна або крос-платформна розробка програм;
- сторонні інтеграції;
- складність візуального інтерфейсу та коду;
- використання апаратних функцій, таких як камера, Bluetooth, NFC тощо;
- план подальшого технічного обслуговування програми.

На веб-сайті Business of Apps зазначено, що просту програму можна створити маючи бюджет від 40 000 до 60 000 доларів США, за програму середньої складності можна заплатити від 61 000 до 120 000 доларів США, а розробку складної програми - від 120 000 і більше доларів США. У ньому також зазначено, що мати розробника додатків у штаті компанії коштуватиме приблизно 90 000 доларів США на рік [22].

Додатково варто також зазначити, що існують різні ролі в команді розробки мобільних додатків, серед яких можна виділити: проєктний менеджер, менеджер команди, розробник клієнтської частини, розробник серверної частини, дизайнер і тестувальник. Цих ролей може бути значно більше і залежить це від складності та особливостей самої програми. Так само часто одна людина може закривати декілька подібних ролей в процесі розробки. Залежно від складності програми змінюватиметься кількість людей і навантаження на кожного, а також вартість програми.

Підсумовуючи, можемо сказати, що не існує якогось єдиного інструменту, щоб можна було точно порахувати вартість створення мобільного додатку, а

також точно передбачити майбутні доходи, що він принесе. На подібні речі впливає занадто багато факторів, тому ми можемо лише робити прогнози. А їх точність буде базуватись на тому, наскільки глибоко ми розуміємо ринок, на якому збираємось працювати, користувачів, технології та методології створення мобільних додатків як таких та інші фактори.

Отже, проаналізувавши сферу розробки мобільних додатків та ринок мобільних програм в цілому, ми можемо зробити SWOT-аналіз цієї сфери, розглянувши її з чотирьох сторін (Додаток А):

- сильні сторони;
- слабкі сторони;
- можливості;
- виклики.

Як вже відзначалося вище, цілком очевидним є той факт, що в майбутньому ринок мобільних додатків буде продовжувати стрімко зростати. Щоб залишити задоволених клієнтів і зберегти тенденцію, яка набирає обертів у галузі, слід мати на увазі наявні нові технології, що вже починають активно застосовуватись в сучасних мобільних додатках. Коротко розглянемо деякі з них.

1. Штучний інтелект (ШІ). Штучний інтелект не є новим у світі мобільних додатків, але він почав робити собі ім'я, коли його почали впроваджувати в такі послуги, як банкінг, онлайн-покупки, музика, фінанси тощо. Зараз люди добре знайомі з такими помічниками на основі ШІ, як Siri, Google Assistant, Alexa тощо. Вони звикають до цих помічників, як до частини свого життя. Чат-боти на різних веб-сайтах використовують як засіб отримання негайного спілкування та оперативних відповідей на поширені запитання. У поєднанні штучного інтелекту з Інтернетом речей, хмарними обчисленнями та блокчейном може виникнути безліч нових механізмів та підходів до взаємодії з користувачем.

2. AR/VR (доповнена реальність). Мобільні програми можуть використовувати концепцію AR і VR, щоб зробити ігри більш реалістичними та приємними. Додатки для роздрібної торгівлі можуть використовувати AR/VR,

щоб надавати користувачам індивідуальний досвід, як-от оптика, одяг, дизайн інтер'єру, меблі. Це може бути дуже захоплюючим і цікавим, якщо використовувати його в галузі освіти для проведення онлайн-занять. У сфері охорони здоров'я AR/VR може відігравати важливу роль у діагностиці, лікуванні та розширенні можливостей медичних працівників. Цифрова взаємодія на основі AR/VR може підняти клієнтський досвід роботи з програмою на новий рівень.

3. Технологія Beacon. Технологія Beacon стає все більш популярною серед останніх тенденцій, які застосовуються в галузі охорони здоров'я, магазинах і готелях тощо. Клієнти зазвичай хочуть отримувати персоналізовані пропозиції та знижки, бачити, на що орієнтуються люди, які знаходяться поблизу. У готельній сфері додатки можуть використовувати Beacon для покращення досвіду реєстрації та виїзду, перевірок безпеки та отримання важливої інформації про наявність вільних номерів. Програми, засновані на даній технології, також можуть автоматизувати реєстрацію та виїзд, навігацію тощо. Програми, які використовуються на спортивних змаганнях та в спортивній індустрії в цілому допоможуть керувати враженнями аудиторії, визначати місцеположення, розміщувати кіоски, продавати квитки з більшою ціною, акції, давати інформацію в реальному часі тощо.

Отже, якщо підсумовувати наш аналіз поточної ситуації в сфері розробки мобільних застосунків, то можна сказати, що дана сфера має величезний потенціал. І головною перевагою є те, що наразі всі тенденції та прогнози вказують на те, що ринок мобільних застосунків буде тільки продовжувати рости. Але при цьому слід відзначити, що дана сфера є також достатньо конкурентною, а самі користувачі через велику кількість пропозицій стають все більш вибагливими. Існує значна кількість можливостей до монетизації мобільних застосунків, вибір яких залежить від особливостей самого застосунку, маркетингової стратегії, цільової аудиторії тощо. Але при цьому і розробка мобільних програм являє собою не тривіальну задачу і вимагає як висококваліфікованих спеціалістів, так і значні бюджети.

1.2. Підходи до організації розробки мобільних застосунків

Методології розробки програмного забезпечення починають свій розвиток з 1970-х років. Гнучкі методології з'явилися, щоб відповідати мінливим вимогам бізнесу та краще керувати життєвим циклом розробки програмного забезпечення. Вони надають методи, які полегшують спілкування між розробником і замовником і проходять цикли «розробка - доставлення - зворотний зв'язок», щоб мати більш точне уявлення про вимоги та бути готовим до будь-яких змін у будь-який час. Гнучкі методології розробки допомагають компаніям створювати правильний продукт і дозволяють командам постійно переробляти свої застосунки, щоб оптимізувати їхню цінність протягом усього процесу розробки, дозволяючи їм бути максимально конкурентоспроможними на ринку. Таким чином, основна мета методології полягає в тому, щоб надати те, що потрібно, у відповідний час протягом циклу розробки. Гнучкі методології є одними з найкращих підходів до розробки програмного забезпечення, які можна застосовувати в той час, коли вимоги клієнта не є точними або коли терміни та бюджет обмежені. Однак, окрім переваг, пов'язаних із застосуванням гнучких методологій у розробці програмного забезпечення для мобільних пристроїв, існують проблеми, які викликають занепокоєння у звітах про підвищення якості, продуктивності та задоволеності бізнесом різними групами показників. Повідомлялося, що гнучкі методології були успішними в більшості випадків, тоді як існують суперечливі звіти, які стверджують, що методологія ще занадто молода, щоб вимагати широких академічних доказів їх успіху.

Agile Маніфест, який також називають Маніфестом гнучкої розробки програмного забезпечення, є офіційним документом, в якому висвітлюються чотири ключові цінності й 12 принципів, якими слід керуватися при розробці програмного забезпечення. Маніфест був опублікований групою розробників програмного забезпечення, яка обговорювала різні засоби для полегшених

методів розробки порівняно з екстремальним програмуванням, SCRUM, DSDM, адаптивною розробкою програмного забезпечення, Crystal, розробкою, керованою функціями, і практичним програмуванням [28]. Простіше кажучи, маніфест Agile виділяє процес розробки, який став керівною силою для практиків Agile. Гнучка методологія зосереджується більше на людських аспектах розробки програмного забезпечення, ніж на перспективних процесах, таким чином використовуючи людську взаємодію над інструментами та процесами. Agile Маніфест встановив загальний набір цінностей і принципів для всіх окремих гнучких методологій. Він детально описує чотири основні цінності для забезпечення високої продуктивності, ефективності та результатів:

1. Люди та співпраця важливіші за процеси та інструменти.
2. Продукт, що працює важливіший за вичерпну документацію.
3. Співпраця з замовником важливіша за обговорення умов контракту.
4. Готовність до змін важливіша за дотримання плану.

Ці основні цінності додатково підтримуються 12 принципами, які лежать в основі Agile Маніфесту та включають наступні:

1. Найвищим пріоритетом для нас є задоволення потреб замовника, шляхом завчасного та регулярного постачання програмного забезпечення.
2. Схвальне ставлення до змін, навіть на заключних стадіях розробки. Agile-процеси надають можливість використовувати зміни задля забезпечення конкурентоспроможності замовника.
3. Продукт, що працює слід випускати якомога частіше, з періодичністю від пари тижнів до пари місяців.
4. Впродовж усього проєкту розробки й представники бізнесу повинні працювати разом щодня.
5. Над проєктом повинні працювати вмотивовані професіонали. Щоб робота була виконана, потрібно створити їм умови, надати підтримку і повністю на них покластися.
6. Особиста комунікація – найефективніший та найпрактичніший метод донесення інформації до команди, та поширення її всередині.

7. Продукт, що працює – головний показник прогресу.
8. Інвестори, розробники й користувачі повинні мати можливість підтримувати постійний ритм як завгодно довго. Agile допомагає налагодити такий сталий процес розробки.
9. Постійна увага до технічної досконалості і якості проєктування підвищує гнучкість проєкту.
10. Простота, або ж мистецтво мінімізації зайвої роботи, є вкрай необхідною.
11. Найкращі вимоги, архітектурні та технічні рішення виникають у командах, що здатні самоорганізовуватись.
12. Команда регулярно намагається знайти способи підвищення ефективності та відповідно корегує свою роботу.

Швидке зростання мобільних обчислювальних платформ перевершило процеси розробки програмного забезпечення, призначені для розробки мобільних додатків. Групи розробників програмного забезпечення для мобільних пристроїв стикаються з проблемою динамічного середовища з різними розробками та технічними вимогами, з частими змінами потреб і очікувань клієнтів. Змінні потреби та очікування роблять системи складнішими. У цьому безперервно мінливому середовищі гнучкі процеси вважаються найбільш придатними для ринків, що швидко розвиваються, де задоволеність клієнтів регулюється ранньою та частою доставкою цінності, можливостями для змін навіть на пізніх етапах проєкту, скороченим циклом доставки, співпрацею між компаніями та розробниками. Потреба в безперервній технічній досконалості, гарному дизайні, простоті та тому, що робоче програмне забезпечення є основним мірилом прогресу.

Індустрія мобільного зв'язку складається з висококонкурентного, динамічного та невизначеного середовища. Гнучкий підхід розглядається як природний варіант для розробки мобільних додатків. Дослідження, проведені для застосування підходу гнучкої розробки до розробки мобільних додатків, вказують на необхідність процесів розробки програмного забезпечення,

адаптованих до вимог до мобільних застосунків [30]. Було рекомендовано, що гнучкі практики є найкращим вибором, який забезпечує різні фази життєвого циклу розробки програмного забезпечення та більш ефективно розв'язує питання розробки [31]. Вважається, що гнучкі інновації можуть запропонувати різноманітні рішення для мобільних додатків і допомогти розробникам послуг, які потребують високоякісних процесів розробки [29].

Абрахамссон та ін. [32] продемонстрували риси, чому гнучкі методології найкраще підходять для розробки мобільного програмного забезпечення. Каннан [33] також підкреслює придатність гнучкої розробки програмного забезпечення для розробки мобільних додатків через невеликі команди, короткі терміни, важливість зручності використання, швидку доставку та меншу складність. Автори запропонували сім методів, за допомогою яких методи Agile-розробки покращують розробку мобільних додатків, включаючи:

- експерименти та адаптацію мобільних додатків;
- надійність, що забезпечує безперервне використання додатків;
- розширення Agile спринтів у моделі мобільних додатків;
- реагування на зміни технологій;
- швидке реагування на відгуки клієнтів;
- більш продуманий досвід користувача;
- поетапне розгортання наборів функцій.

Холлер [30] припустив, що гнучка розробка програмного забезпечення пропонує величезні можливості та цінність для команд мобільних розробників, які працюють над впровадженням легкого процесу розробки або скороченням бюрократичних процесів. Автор наголосив на прогресі мобільних комп'ютерних технологій і швидкому зростанні якості та кількості бездротових мереж, що принесло нові програми та проблеми в цьому динамічному середовищі. Він також підкреслив швидкість, з якою промисловість повинна адаптуватися та змінювати себе від звичайних методів розробки до систем, що задовольняють особливі потреби цієї галузі.

Крім того, гнучкі методології також піддавалися критиці через їх неефективність при використанні у великих організаціях і певних типах проєктів, де вони були зараховані як область, що потребує подальших досліджень. Було припущено, що гнучкі методи здаються найкращими для розробки непослідовних проєктів, і багато організацій вважають, що методологія є надто екстремальною при великій кількості залучених підсистем до процесу. Барлоу та ін. [34] запропонував цим організаціям віддати перевагу гібридному підходу, який поєднує елементи гнучкого та планового підходів для задоволення своїх потреб. Боем і Тернер [35] і Бек [36] припустили, що гнучка розробка виявилася менш надійною та придатною для певних типів середовища та команд, які включають невелику кількість експертів.

Технологія також була оцінена численними спостерігачами як «примха управління» та твердження про вимірне покращення бізнесу через вимірювання метрик, визначених нею самою (наприклад, швидкість) [37-39]. Технологія має свої обмеження щодо розподілених зусиль розробки, використання гнучкого процесу розробки програмного забезпечення з використанням аутсорсингу розробки критично важливих систем, де збій не є варіантом за будь-яку ціну (наприклад, програмне забезпечення для управління повітряним рухом) (Fowler, 2010) [40].

Цю методологію також критикували з інших причин, зокрема відсутність структури та необхідної документації, проблеми в роботі з розробниками старшого рівня, недостатній дизайн програмного забезпечення, потреба в прийнятті занадто значних культурних змін, що може призвести до складніших переговорів щодо укладання контрактів, залежність від функціональних вимог, оскільки нефункціональні атрибути якості важко розмістити як історії користувачів. [41].

У сфері розробки програмного забезпечення для мобільних пристроїв Корал та ін. представив опитування, яке демонструє відсутність доказів, які вказують на чіткий зв'язок між запропонованими гнучкими методологіями та їх використанням у реальних умовах [42], що свідчить про тенденцію, згадану в

попередньому аналізі Джейнсом та ін. [43], що відображає виявлений спад у розгляді методів Agile як «срібної кулі».

За останнє десятиліття різними вченими були запропоновані наступні гнучкі методології, які використовують комбінацію гнучких і негнучких методів для розробки мобільних додатків.

Одне з піонерських досліджень гнучкого підходу було проведено Абрахамсоном та ін. [44], де було оцінено, що рішення гнучкої розробки добре підходить для середовища розробки мобільних додатків, і запропоновано новий підхід під назвою Mobile-D. Дослідження надає огляд процесу розробки мобільних додатків, і як їх особливі характеристики і обмеження впливають на процес розробки програмного забезпечення. Дослідження також представляє підхід до розробки програмного забезпечення, взятий із галузі гнучкої інженерії програмного забезпечення, яка розроблена для задоволення конкретних вимог надзвичайно нестабільного мобільного середовища. Також була представлена нова концепція архітектури під назвою Architecture Line, яка спрямована на створення інфраструктури додатків, яка керує розробкою майбутніх програм.

Підхід Mobile-D базується на Rational Unified Process RUP (ітеративний процес розробки програмного забезпечення від IBM), eXtreme Programming XP (набір практик для розробки програмного забезпечення) і методології Crystal (одна з гнучких методологій до розробки програмного забезпечення). Коротко, методологія Mobile-D складається з п'яти етапів, кожен з яких має ряд пов'язаних етапів, завдань і практик:

- дослідження;
- ініціалізація;
- виробництво;
- стабілізація;
- тестування системи та виправлення.

Проект розробки програмного забезпечення, який слідує підходу Mobile-D, розділений на п'ять ітерацій. Ці фази:

- налаштування;

- базова;
- основна;
- стабілізація;
- завершення.

Кожна з цих фаз складається з трьох різних типів днів розробки:

- планування;
- роботи;
- випуск продукту.

Mobile-D організований у структуру, яка об'єднує основні процеси (планування, проєктування, впровадження, тестування, випуск) із допоміжними процесами (керування проєктом, керування конфігурацією програмного забезпечення, вдосконалення процесу програмного забезпечення).

Також існує 9 основних елементів методології Mobile-D:

- вимоги;
- планування;
- моделювання;
- архітектура;
- час;
- метрики;
- документація;
- вдосконалення;
- кінцеві користувачі;
- тестування).

Підхід Mobile-D було емпірично перевірено в проєктах розробки та успішно оцінено відповідно до сертифікації рівня 2 CMMI (Capability Maturity Model Integration - підхід для вдосконалення процесів, який забезпечує організації суттєвими елементами ефективних процесів). Рекомендується використовувати Mobile-D невеликою групою спільно розташованих розробників, щонайбільше десять, які працюють у короткому циклі розробки для

доставки продукту протягом 8–10 тижнів календарного часу. Попри те, що ця методологія є піонерським дослідженням у цій галузі, здається дуже перспективною та відіграє важливу роль у теорії. Важливо зазначити, що цей підхід є поверхневим і не повністю визначеним, щоб буквально використовувати його на практиці. Крім того, інші автори запропонували її подальші вдосконалення, тож модель може бути удосконалена за допомогою гібридних гнучких методологій.

Спатару [45] оцінив придатність Agile-методів для проєктів розробки мобільних додатків, внісши набір покращень у Mobile-D методологію та забезпечивши інструментальну підтримку, щоб впровадити ці покращення, полегшивши тестування продуктивності, журнал використання та автоматичне тестування. Дослідник виявив важливість пост-релізу, який не описано в Mobile-D, і вирішив проблему, розширивши методологію з точки зору охоплення життєвого циклу та додавши нову фазу, яка стосується постійної інтеграції відгуків кінцевих користувачів про наданий продукт у майбутні версії. Удосконалення Mobile-D, запропоновані Spataru, повинні пройти експериментальну перевірку в реальній організації. Такий експеримент вимагатиме великої роботи у співпраці з командами розробників, які працюють із різними методологіями, і аналізу того, чи досягнуто очікуваних переваг, описаних удосконалень при застосуванні в реальних проєктах.

Думс та ін. запропонував метод під назвою «RaPiD7» (швидке виробництво документації, 7 кроків), який покращує документаційну роботу без шкоди для кількості та якості документації [36]. RaPiD7 описує, як планується взаємодія людей у проєктах програмного забезпечення та як створюються документи на фасилітованих семінарах.

RaPiD7 підтримує всі проєкти розробки програмного забезпечення, незалежно від того, пов'язані чи не пов'язані з розробкою мобільних додатків, але експерименти з тематичних досліджень проводилися у великих мобільних компаніях, таких як Nokia та Philips. Цей метод був експериментально випробуваний у лабораторії цифрових систем Philips і розроблений у компанії

Nokia у 2002-2003 роках. Мотивація полягає в тому, щоб створювати документи, які відповідають дійсності, і створювати їх з якомога меншими зусиллями. RaPiD7 удосконалює традиційний підхід роботи зі специфікаціями, пропонуючи спосіб планування людської взаємодії на ранніх етапах проєктів програмного забезпечення та надаючи засоби для прийняття рішень і документування разом із вбудованою системою контролю якості.

Рахіміан і Рамзін [47] представляють інший підхід до Mobile-D. Вони запропонували гібридну гнучку та ризикоорієнтовану методологію, яка генерує метод, придатний для розробки мобільних додатків, розроблений на основі методологічної техніки. Це дисципліна, яка займається створенням методологій, придатних для різних сценаріїв розвитку, мотивованих переконанням, що жоден процес не підходить для всіх ситуацій. Він заснований на поєднанні гнучких методологій, адаптивної розробки програмного забезпечення (ASD) і розробки нових продуктів (NPD).

Ідеальними характеристиками розробки програмного забезпечення для мобільних пристроїв, на яких базується методологія гібридного проєктування, є:

- гнучкість;
- ринкова свідомість;
- підтримка лінійки програмних продуктів;
- розробка на основі архітектури;
- підтримка багаторазового використання;
- включення сеансів перегляду та навчання;
- рання специфікація фізичної архітектури.

Процес проєктування гібридної методології був розроблений як низхідний, ітераційно-поступовий процес, який складається з наступних завдань:

- визначення пріоритетів вимог;
- вибір підходів до проєктування, які будуть використовуватися в поточній ітерації;
- застосування вибраних підходів до проєктування;

- перегляд, уточнення та реструктуризація розробленої на цей момент методології;
- визначення рівня абстракції для наступної ітерації;
- перегляд і уточнення вимог, визначаючи їх пріоритетність для наступної ітерації.

Коротко кажучи, запропонована методологія мобільної розробки була створена в чотири ітерації, починаючи з загального життєвого циклу розробки програмного забезпечення. У першій ітерації методологія була деталізована шляхом додавання практик, які зазвичай зустрічаються в гнучких методах. Беручи до уваги ринкові міркування, друга ітерація включала дії з розробки нового продукту, процесу, пов'язаного з виведенням на ринок нового продукту чи послуги. У третій ітерації ідеї адаптивної розробки програмного забезпечення (ASD) були інтегровані в методологію, тоді як у четвертій і останній ітерації було додано прототипування, щоб зменшити ймовірні ризики, пов'язані з новою методологією.

Структура розробки гібридного методу враховує більшість питань та проблем, виявлених у роботі за подібними методологіями в цій галузі. Проте методологія поки що на високому рівні, і конкретних практик чи визначених етапів не передбачено. Опублікований матеріал про гібридну інженерну методологію не містить прикладів та не показує, що методологія була емпірично перевірена на розробці реального мобільного програмного продукту.

Йонг та ін. [48] запропонував гнучку методологію програмного забезпечення мобільних додатків (MASAM), яка забезпечує процес швидкої розробки мобільних додатків за допомогою гнучкого підходу. Він базується на екстремальному програмуванні (XP), гнучкому уніфікованому процесі, RUP і метамоделі розробки програмного забезпечення та систем (SPEM). Це графічний інтерфейс, орієнтований на архітектуру, який використовує гнучкі підходи для швидкої розробки та використовує знання домену. MASAM демонструє тісний зв'язок із методологією Mobile-D і вводить невеликі варіації, такі як інструмент

управління проєктом і подальшої роботи в поєднанні з Eclipse Process Framework.

MASAM пропонує цикл розробки мобільних додатків, що складається з чотирьох етапів. На підготовчому етапі визначається короткий зміст і перше уявлення про продукт, а також розподіляються ролі та обов'язки. Етап втілення зосереджується на розумінні потреб користувача та визначенні архітектури програмного продукту. Останній етап включає фазу розробки продукту, яка використовує традиційні гнучкі принципи для забезпечення ітеративної послідовності розробки та екстремальне програмування. Впровадження програмного продукту здійснюється за допомогою тестової розробки, парного програмування, рефакторингу та безперервної інтеграції з тісним зв'язком із діяльністю ітераційного тестування. Нарешті, фаза комерціалізації зосереджена на запуску продукту та діяльності з продажу продукту.

Рекомендується використовувати методологію MASAM невеликим компаніям, які орієнтовані на розробку мобільних програмних додатків. Проте автори не представили приклад фактичного впровадження цієї методології в реальному середовищі.

Куна та ін. [49] запропонований SLeSS - гнучкий підхід, який інтегрує Scrum і Lean Six Sigma (LSS), які зосереджуються на управлінні проєктами та удосконаленні процесів відповідно. У цьому підході використовуються два типи пріоритезованих списків задач (беклогів):

- Customization Product Backlog - фокус саме на продукті, який створюється. Містить в основному функціонал продукту, над яким працює команда;
- LSS Product Backlog - набір задач для вдосконалення процесів роботи.

Використання SLeSS сприяє легкій адаптації до змін вимог на пізніх стадіях проєкту з меншим загальним впливом, ніж традиційний підхід, допомагає дотримуватись термінів, скорочує понаднормові години та швидше забезпечує постачання нових версій, скорочує цикл розробки. Крім того,

використання підходу дозволяє досягти цільових показників продуктивності та якості реального проєкту розробки програмного забезпечення, підвищує продуктивність, покращує якість процесу, допомагає зменшити витрати, поступово покращує процес розробки, процес управління та результати проєктів із меншою кількістю дефектів і збоїв.

Scrum — це гнучка методологія для управління проєктами та розробки програмного забезпечення, яка використовує емпіричний підхід, а не директивний, і тому її можна використовувати в складних проєктах. З іншого боку, Lean Six Sigma (LSS) — це методологія для визначення та вдосконалення продуктів, процесів і послуг із зосередженням на зменшенні дефектів і збоїв, на варіаціях і усуненні відходів, розставляючи пріоритети, запланованим і об'єктивним чином, на досягнення якості та фінансових результатів.

Scrum широко використовується в розробці програмного забезпечення, і це спостерігалось та задокументовано у сфері розробки мобільного програмного забезпечення [50]. Виконання SLeSS передбачає поетапний підхід, спочатку впроваджуючи лише Scrum, а після того, як Scrum добре врегульовано в організації, LSS слід впроваджувати як структуру якості. Рекомендації щодо Scrum у SLeSS:

- розмір спринту: має тривати один або два тижні;
- розмір команди: у групі має бути від чотирьох до дев'яти осіб;
- список задач спринту: містить дії направлені вдосконалення процесу;
- команда розробників і клієнт разом визначають проблеми та додають їх в спринт;
- Lean Six Sigma (LSS): всі члени команди та клієнт/замовник мають належне розуміння методів LSS, процесів розробки та управління.

Після того, як Scrum добре влаштувався в організації, Lean Six Sigma (LSS) застосовується як структура якості, яка доповнює Scrum як методологію розвитку. Модель представлена 5 фазами DMAIC:

- визначення;

- вимірювання;
- аналіз;
- покращення;
- контроль.

Підхід SLeSS використовувався в реальних проєктах розробки вбудованого програмного забезпечення для мобільних телефонів. Цей підхід експериментували в лабораторії, за участю виробника мобільних телефонів як клієнта з командою з 7-12 розробників, протягом 4-6 місяців із середнім розміром 529 LOC (рядків коду), розробленого мовою програмування ANSI C.

Практика SLeSS сприяє досягненню кращих результатів, таких як краща адаптація до змін у вимогах, виконання крайніх термінів, зменшення кількості незапланованих понаднормових робочих днів і швидке надання більшої кількості версій продукту, а також з меншою кількістю дефектів або збоїв. Це демонструє збільшення продуктивності, покращення якості процесу та зниження вартості. Крім того, цей підхід дозволив удосконалити процеси розробки та управління, започаткувавши їх статистичний контроль і узгодивши їх із цілями та очікуваннями клієнтів.

Підсумовуючи, варто зазначити, очевидно, що відбулося значне збільшення проєктів розробки мобільних додатків. Попит на мобільні додатки зростає, а потенційна їх кількість практично необмежена. Відповідний метод Agile можна вибрати для певного проєкту та адаптувати до конкретних вимог на основі складності проєкту та розміру команди. Незважаючи на виявлені можливості для бізнесу, можна знайти дуже мало наукових публікацій, які стосуються конкретних проблем, з якими стикаються організації-розробники під час розробки програмного забезпечення для мобільних пристроїв. Хоча запропоновані моделі розробки мобільних додатків, які обговорюються та розглядаються в цьому розділі, є обнадійливими, але все ще є теоретичними і на практиці вимагають значної адаптації під контекст команди, проєкту, клієнта тощо.

1.3. Основні обмеження та ризики при розробці мобільних застосунків

Лі Вільямсон перерахував унікальні проблеми для розробки мобільних додатків, такі як форм-фактори, технологія введення користувача, зручність використання та дизайн взаємодії з користувачем [51]. Васерман та ін. визначив проблеми, пов'язані з розробкою мобільних додатків на основі процесу розробки, інструментів, дизайну інтерфейсу користувача, переносимості додатків, якості та безпеки [29]. Подібним чином Делінджер та ін. у своєму дослідженні визначили чотири основні виклики для розробки мобільних додатків. Вони помітили ці проблеми у необхідності створення універсального інтерфейсу користувача, під час повторного використання програмного забезпечення на мобільних платформах, під час розробки контекстно-залежних мобільних додатків і під час балансування гнучкості та невизначеності у вимогах [52]. Дай та ін. зазначають, що у зв'язку з повсюдним використанням і розгортанням мобільних додатків проблеми з безпекою зростають, і їх вирішення стало головною проблемою [53]. Слід також зазначити, що з постійним збільшенням кількості апаратного та програмного забезпечення кількість проблем, з якими стикаються при розробці та використанні мобільних додатків також зростає. Зростаюча складність пристроїв, зростаючий ринок додатків і зростаюча перевага бездротових мереж разом роблять розробку мобільних додатків галуззю з великим потенціалом, але в той самий час - з великими викликами. Тож нижче пропонуємо розглянути основні виклики сфери розробки мобільних застосунків.

1. Взаємодія з користувачем. Використання мобільного пристрою відрізняється від роботи з настільним комп'ютером чи ноутбуком. Менший дисплей і різні стилі взаємодії з користувачем також сильно впливають на дизайн взаємодії при розробці додатків. Мобільний інтерфейс користувача базується на віджетах, дотику, фізичних рухах і клавіатурах, а не на звичних меню піктограм, що є

звичним для ПК. При використанні мобільних програм доводиться враховувати велику кількість потенційних сценаріїв її використання з боку користувача, різні умови, в яких він може перебувати, взаємодіючи з нею. Розробник повинен переконатися, що взаємодія адаптована до поточної ситуації користувача, наприклад, що існує можливість перемикатися між візуальним і акустичним способом подачі інформації, якщо це доцільно.

2. Інтерфейс користувача (UI). Користувачі мобільних додатків часто використовують свої пристрої на ходу, що означає, що їм потрібен дуже простий інтерфейс користувача.. Їм необхідно отримати конкретну інформацію в ефективний спосіб. Але оскільки все більше людей починає займатися розробкою додатків, кілька додатків іноді пропонують однакові функції. А це означає, що додатку з привабливішим інтерфейсом користувача віддається перевага над конкурентами. Тому розробнику необхідно знайти правильний баланс між простотою та привабливістю. Спроба збільшити привабливість за допомогою кращої графіки та інших тривіальних функцій також негативно впливає на ресурси пристрою, що сповільнює роботу програми та робить її громіздкою. Мобільні пристрої дуже неоднорідні щодо можливостей відображення та введення даних, а також використовуваних конфігурацій програмної платформи. Це призводить до складного процесу розробки додатків, які мають націлюватися на широкий спектр пристроїв [54]. Розробникам необхідно звернути увагу на потребу програми адаптуватися до різних розмірів екрана та можливість перемикавання між портретним і альбомним режимами відображення. Грубий спосіб перевірити зручність використання різних пристроїв — це емулятори, які є віртуальною симуляцією реальних пристроїв. Це недорогі інструменти тестування, які доступні разом із наборами для розробки для кожної платформи, але часто вони не відповідають

реальним пристроям [55]. Остаточний тестовий запуск на реальних пристроях все ще необхідний, щоб переконатися в дотриманні всіх нефункціональних вимог. Крім того, парадигма інтерфейсу користувача не повинна сильно відхилятися від поточних тенденцій. Це позбавить користувача від клопоту звикання до нової програми.

3. Технологія введення користувача. методи введення, які пропонуються для мобільних пристроїв, є складними та вимагають певного рівня кваліфікації. Фізична відмінність мобільних додатків полягає в тому, що механізм введення користувачами інший. Мобільні пристрої стали першими у використанні жестів без клавіатури, наприклад, дотиків, форм і щипків, як ефективних і популярних методів введення інформації та взаємодії з програмами. Жести потрібно спланувати та підтримувати для задоволення користувачів мобільних додатків.

4. Управління даними. Де зберігати дані користувача – це питання, яке розробники додатків повинні вирішити перед початком етапу проєктування. Можливими варіантами є власна файлова система пристрою користувача та онлайн-сервер. Обидва мають свої підводні камені та переваги. У той час, як простір для зберігання в пристрої користувача складається з масиву незалежних жорстких дисків малого форм-фактора, підключених до хоста через інтерфейс зберігання в мобільному обчислювальному середовищі [56], це вигідно через його постійну доступність і можливу економію часу. В той самий час сховище на сервері створює проблему, коли користувач не може під'єднатися до мережі. Час, витрачений на розподілену передачу даних також є джерелом затримки. В ідеалі програма повинна розглянути гібридне рішення, у якому програма спілкується з онлайн-сервером через часті проміжки часу для синхронізації та оновлення даних, які кешуються на пристрої користувача. Ця архітектура вимагає вбудованих можливостей для

фіксації транзакцій і відкату, щоб запобігти виникненню неузгодженості даних. Формат зберігання даних слід вибирати відповідно до вимог програми, щоб оптимізувати продуктивність програми. Головною метою є скорочення загального часу пошуку, перетворення, перекладу та зберігання даних.

5. Безпека. бездротова передача в мобільних пристроях вразлива для атак деяких неавторизованих користувачів, які завантажують дані бездротовим способом у потенційно небезпечних місцях. Розробник мобільних додатків відповідає за збереження даних, захист даних і безпеку мобільного додатка від кінця до кінця, включаючи програмне забезпечення, а також передачу даних і сервер.

6. Технічне обслуговування та підтримка програми. Це значною мірою залежить від потреб користувачів і змін тенденцій. Це також спосіб виправлення помилок у програмах. Додатки суттєво відрізняються від традиційних програмних систем у багатьох аспектах, що вимагає розробки нових підходів до їх підтримки та розуміння [57]. Це вимагає механізму для розробників, щоб бути в курсі використання програми та задоволеності користувачів. Цей механізм можна автоматизувати за допомогою аналітичних інструментів або запитати як відгук користувача. Аналітика може відстежувати, які функції додатків частіше використовують користувачі, і визначати моделі використання ними додатків. Flurry Analytics, наприклад, галузевий стандарт у сфері мобільних пристроїв, надає понад 125 000 розробникам бізнес-дані, необхідні їм для розуміння своєї аудиторії, використання та ефективності [58]. Розробляючи аналітичні рішення, розробник повинен включити механізми, які реєструють офлайн-активність і повідомляють про це, коли користувач виходить в Інтернет. Аналітика також може бути

корисною функцією для розв'язання технічних проблем шляхом зберігання журналів збоїв та інших проблем.

7. Нефункціональні вимоги. Існує низка нефункціональних вимог, які відіграють ключову роль у забезпеченні того, щоб мобільна програма надавала послуги з прийнятними рівнями стандартів якості послуг. Основними та фундаментальними функціональними параметрами для мобільних додатків є надійність, якість, безпека тощо.

Мобільне середовище відрізняється від традиційних середовищ, і це викликає деякі нові дослідницькі питання, такі як:

- Чи поведуться мобільні програми інакше під час підключення до телефонної мережі (3G, 4G), ніж під час використання з'єднання Wi-Fi?
- Чи потрібні нові методи для забезпечення цілісності даних? Чи становить втрата з'єднання або заряду батареї ризик для цілісності програми чи даних?
- Як створити мобільну програму, яка максимізує час автономної роботи та використання ресурсів?

Отже, підсумовуючи, можна сказати, що незважаючи на багато схожих рис між розробкою програм для ПК та мобільних пристроїв, існують доволі суттєві особливості, характерні саме для мобільної розробки та в цілому ринку мобільних застосунків. Ці фактори неодмінно слід враховувати як під час планування створення мобільного застосунку, так і в процесі його вже життєвого циклу.

1.4. Огляд існуючих рішень

Якщо оцінювати потенційних конкурентів, то, на нашу думку, слід в першу чергу відштовхуватись наразі від сервісів, які надають можливість не тільки оцінювати фізичні якості спортсменів, а й надавати можливість робити це

на рівні всієї команди, залучаючи до процесу тренерів та інших зацікавлених представників команди (фізіотерапевтів, аналітиків і т.п.). Аналізуючи ринок подібних рішень, ми виділили наступні продукти:

-	T
eamSnap;	
-	S
portsmanGen;	
-	Tr
ainHeroic;	
-	M
ySked;	
-	C
oachMePlus.	

Для порівняльного аналізу нами були обрані наступні критерії: приблизна кількість активних користувачів, наявність мобільних додатків для операційних систем iOS та Android, наявність безкоштовної версії, вартість платної версії. Таблиця з порівняльним аналізом популярних сервісів для менеджменту спортивних команд наведена нижче (Таблиця 1.2.):

Таблиця 1.2.

Порівняльний аналіз популярних сервісів для менеджменту спортивних команд

Сервіс	Кількість користувачів	Додаток для iOS	Додаток для Android	Наявність безкоштовної версії	Вартість підписки за місяць
TeamSnap	20+ млн.	Так	Так	Так	\$10 - \$25
SportsmanGen	100+ тис.	Так	Так	Так	\$10 - \$50
TrainHeroic	1+ млн.	Так	Так	Так	\$40 - \$100
MySked	500+ тис.	Так	Так	Так	\$30 - \$50
CoachMePlus	250+ тис.	Так	Так	Так	\$20 - \$50

Аналізуючи представлені вище дані, можемо відзначити декілька важливих моментів:

- Кожен з популярних сервісів має додатки як для платформи iOS, так і для платформи Android;

- В кожного з сервісів є можливість безкоштовно користування, при цьому функціонал сервісу буде обмежений. Проте такий механізм спрощує залучення користувачів, адже дає можливість спочатку оцінити потенційну цінність, яку їм надає продукт, а вже потім оплачувати підписку, отримуючи доступ до додаткового функціоналу;

- Ціна платної підписки досить сильно варіюється (від 10 до 100 доларів), та залежить здебільшого від основної цільової аудиторії;

- Подібні сервіси мають доволі велику аудиторію, що свідчить про наявність необхідного ринку користувачів.

Також додатково ми вирішили розглянути наступні популярні сервіси, які не мають функціоналу, пов'язаного з менеджментом спортивних команд, але при цьому є досить популярними для вимірювання та оцінювання специфічної фізичної активності спортсменів:

- Nike Training Club; Ni

- Strava; St

- MyLift; M

- MyJump. M

Незважаючи на те, що ці сервіси не є нашими прямими конкурентами, вони мають функціонал, який ми очікуємо також пропонувати нашим потенційним користувачам у майбутньому. Тому доцільно розуміти зацікавленість користувачів у такого роду застосунках. Нижче наведено порівняльну таблицю популярних сервісів для вимірювання та оцінювання специфічної фізичної активності спортсменів (Таблиця 1.3.).

Таблиця 1.3.

Порівняльна таблиця популярних сервісів для вимірювання та оцінювання специфічної фізичної активності спортсменів

Сервіс	Кількість користувачів	Додаток для iOS	Додаток для Android	Наявність безкоштовної версії	Вартість підписки за місяць
Nike Training Club	150+ млн.	Так	Так	Так	\$10 - \$25
Strava	100+ млн.	Так	Так	Так	\$10
MyLift	1+ млн.	Так	Так	Так	\$10
MyJump	100+ тис.	Так	Так	Так	\$10

Порівнюючи дані подані у таблиці вище з аналогічними даними щодо найпопулярніших сервісів для менеджменту команд, можемо тільки підтвердити важливість наявності застосунків одразу для обох операційних систем (iOS та Android), а також можливості безкоштовного використання застосунку (навіть якщо це буде версія з обмеженим функціоналом).

Важливим моментом також є висока популярність подібних застосунків. Це надає доволі багато можливостей для потенційних змін у майбутньому. У випадку, якщо в ході роботи над нашим випадком виявиться, що гіпотези з приводу цінності нашого рішення, чи зацікавленості в ньому потенційних користувачів виявляться хибними, існує багато можливостей внесення не радикальних змін для покращення шансів проєкту на успіх: змінити цільову аудиторію, змінити ключовий функціонал сервісу, змінити бізнес-модель чи спосіб монетизації і т.п.

Отже, аналізуючи існуючі рішення як потенційних конкурентів, можемо дійти висновку, що існує доволі великий ринок користувачів, зацікавлених у подібного роду сервісах. При цьому паралельно існують продукти, які не надають можливості менеджменту спортивних команд, але при цьому мають функціонал оцінки фізичних активностей. І дані продукти так само користуються високою популярністю. Також важливим фактом є те, що всі популярні сервіси, розглянуті вище, мають як безкоштовні версії з обмеженим функціоналом або часом використання, так і платні версії.

РОЗДІЛ 2

АНАЛІЗ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ ПРОГРАМНОГО ЗАСТОСУНКУ

2.1. Використання аутсорсингу для розробки мобільного додатку. Аналіз та основні напрямки діяльності ОНІКС-СИСТЕМЗ

Для роботи над застосунком як потенційного партнера було обрано українську ІТ компанію “ОНІКС-СИСТЕМЗ”.

Компанія ТОВ “ОНІКС-СИСТЕМЗ” - українська ІТ компанія, заснована у 2001 році в місті Кропивницький. На сьогодні в компанії працює понад 350 спеціалістів. Компанія має офіси в чотирьох країнах: Україна, Польща, Словаччина та США. ТОВ “ОНІКС-СИСТЕМЗ” є ІТ компанією аутсорсингового типу та надає послуги повного циклу розробки для клієнтів по всьому світу. Основні цільові ринки - США, Канада, Західна Європа. Наразі компанія інформує, що за свою більш ніж 20-річну історію нею реалізовано вже понад 2500 проєктів.

Компанія надає послуги в наступних напрямках:

- | | |
|---|---|
| - | Р |
| розробка Web проєктів (веб-сайти та веб-сервіси); | |
| - | Р |
| розробка мобільних застосунків: | |
| - | Р |
| розробка нативних Android додатків; | |
| - | Р |
| розробка нативних iOS додатків; | |
| - | Р |
| розробка крос-платформних мобільних додатків; | |
| - | Р |
| розробка гібридних мобільних додатків; | |

-
озробка VR/AR проєктів;

Р

-
изайн експертиза (UI/UX);

Д

-
енеджмент ІТ проєктів та інше.

М

Компанія спеціалізується в наступних доменах (напрямах діяльності):

-

п

роєкти пов'язані з навчанням (EdTech домен):

-

Д

одатки для онлайн навчання;

-

Н

авчальні діджитал платформи;

-

V

R/AR рішення;

-

Рі

зноманітні рішення для корпоративного навчання.

-

О

хорона здоров'я (HealthCare домен):

-

П

рограми для моніторингу здоров'я;

-

С

RM системи для медичних закладів;

-

М

обільні застосунки для здоров'я;

-

С

истеми менеджменту медичних даних;

-

С

истеми менеджменту даних пацієнтів медичних закладів.

-	Д
омен криптовалют;	
-	П
рограмне забезпечення для страхових компаній	
-	За
стосунки для фітнесу та спортивних активностей	
-	П
рограмне забезпечення для фінансового сектору;	
-	П
рограмне забезпечення для бізнесів пов'язаних з нерухомістю та інші.	

Отже, дивлячись на представлені напрямки діяльності компанії, вона виглядає як гарний потенційний партнер для розробки мобільного застосунку, адже має достатню експертизу у потрібній та в дотичних областях. Також компанія має гарну репутацію, багато різноманітних нагород. А доволі великий розмір компанії свідчить про стабільність бізнес-моделі та наявність ресурсів для забезпечення достатньої гнучкості враховуючи потенційні виклики сьогоденного ринку.

Наразі на ринку розробки мобільних застосунків використовується три основні моделі організації роботи плюс їх різноманітні поєднання:

-	Р
розробка внутрішніми силами компанії, наймання команди в штат (in-house development);	
-	Н
наймання команди за принципами аутстафу;	
-	П
передача розробки іншій компанії за принципами аутсорсингу.	

Враховуючи відсутність штату компанії як такого на етапі планування нашого продукту, варіант аутсорсингу виглядає наразі як найбільш прийнятний.

Аутсорсинг у багатьох випадках здійснюється через відсутність досвіду в якомусь конкретному процесі розробки програмного забезпечення [59]. У той

час як в інших випадках аутсорсинг розглядається через економію коштів і технологічні фактори [60].

Найважливішим фактором, який враховується при виборі цієї моделі роботи, є фактор вартості [61]. Іноді найняти штатного експерта дуже дорого, особливо якщо проєкт має бути завершено в дуже короткий термін. Крім того, у деяких країнах витрати на розробку програмного забезпечення занадто високі, тому організації, що займаються програмним забезпеченням, вважають за краще передати свій проєкт програмного забезпечення якійсь зовнішній офшорній компанії з програмного забезпечення, яка виконає це завдання з відносно меншими витратами [62].

Також у багатьох випадках, коли компанія, що займається розробкою програмного забезпечення, отримує такий тип програмного забезпечення, у якому вона не має достатнього досвіду, тоді виконання цього процесу власними силами призведе до ризику компромісу стосовно якості [63]. Повний провал проєкту також може статися, якщо проєкт виконується власними силами, попри відсутність досвіду у відповідній галузі. У таких випадках компанії віддають перевагу передачі такого типу проєктів сторонній програмній компанії, яка має достатні навички та досвід у цьому конкретному виді програмного проєкту.

Аутсорсинг підвищує коефіцієнт успіху програмного забезпечення завдяки розгортанню висококваліфікованої команди експертів над проєктом, яка інакше була недоступна в домашній організації. Якщо компанія, що займається програмним забезпеченням, не має досвіду в якомусь проєкті, тоді вона передає цей проєкт іншій компанії, яка має високий досвід у цьому конкретному виді проєкту [56,61]. Це призведе до створення якісного програмного проєкту.

Компанії, що займаються програмним забезпеченням, вважають аутсорсинг надзвичайно вигідним, коли проєкти мають бути завершені в дуже короткий термін. Час є важливим фактором, який слід враховувати в будь-якому проєкті. Якщо немає впевненості, що компанія досягне поставленої мети за визначений проміжок часу, вони, як правило, передають проєкт програмного забезпечення зовнішній експертній організації з програмного забезпечення [64].

До того ж аутсорсинг дозволяє фірмам проникати на зовнішні ринки. Отримання послуг від місцевих працівників може допомогти фірмам адаптуватися до звичаїв і традицій зовнішнього ринку [61].

Також аутсорсинг дозволяє фірмам працювати 24 години на добу, коли домашні та аутсорсингові організації знаходяться в різних географічних регіонах або континентах. Якщо є велика різниця в часі, то стороннім організаціям і домашнім організаціям доведеться працювати більше, ніж просто робочий час, а в деяких випадках вони працюватимуть 24 години на добу. Таким чином вони зможуть швидше завершити проєкт або ж надавати цілодобову підтримку клієнтам застосунку вже в процесі його використання [61].

Проєкти програмного забезпечення здебільшого передаються на аутсорсинг, коли вартість власної розробки є вищою, ніж в інших регіонах, через вищі ставки та зарплати. У таких ситуаціях домашня організація вирішить передати проєкт програмного забезпечення якійсь організації в інших регіонах, де вартість виробництва або розробки нижча.

Аутсорсинг програмного забезпечення дуже вигідний як для домашніх, так і для аутсорсингових організацій. Домашня організація отримує експертні знання зовнішніх висококваліфікованих працівників, які дадуть кращу якість програмного забезпечення. Крім того, домашня організація знижує вартість програмного проєкту, а також здатна перекласти ризик.

На додачу до всього аутсорсинг дозволяє отримувати послуги від найкращих кадрів у всьому світі, отже створюючи високоякісні проєкти з меншими зусиллями та часом. Або ж отримувати кваліфіковану підтримку від високоякісних спеціалістів де це потрібно, що може позитивно відзначитись як на якості продукту в цілому, так і на швидкості його розробки [61, 65].

Проте існують і певні недоліки аутсорсингу, як моделі організації роботи для розробки мобільного продукту. Коли домашня організація та аутсорсингова організація знаходяться в різних географічних регіонах, можуть існувати мовні та комунікаційні бар'єри, які можуть спричинити неправильне тлумачення

деяких бізнес-процесів, особливо що потребують розуміння [66]. Це зрештою призведе до поганого програмного проєкту.

Коли організація передає будь-який проєкт іншій організації, конфіденційність даних і конфіденційність клієнта також поставлені під загрозу, оскільки багато конфіденційних даних потрібно надати зовнішній організації [67]. Іноді через секретність, безпеку або з метою мінімізації ризику інформація не надається в повному вигляді, тому результати можуть не відповідати максимально можливим або бажаним. Це нерідко може призвести до втрати чи шкоди репутації головної організації.

Також передача проєктів комусь за межами організації означає втрату контролю над проєктом. Коли проєкт передається аутсорсинговій організації, контроль зрештою переходить до рук сторонніх організацій, що означає, що домашня організація не може керувати та контролювати речі чи бізнес-процеси відповідно до власного бажання чи бажання клієнта [66, 67]. Якщо стороння організація не зможе забезпечити очікувану якість або своєчасну поставку проєкту, це завдасть шкоди довірі та репутації головної організації, що зрештою стане причиною втрати цінних клієнтів чи репутації.

Слід також пам'ятати, що у разі будь-якого технічного обслуговування або оновлення, яке вимагає клієнт, домашня організація повинна знову отримати послуги від аутсорсингової організації. Якщо аутсорсингова організація не має достатньо часу або не бажає надавати необхідні послуги через якісь фінансові чи договірні питання, домашня організація сильно постраждає.

Далі буде проведено порівняльний аналіз власної розробки програмного забезпечення та розробки програмного забезпечення на сторонніх підприємствах, як основних потенційних моделей до організації процесу розробки нашого мобільного застосунку.

У таблиці “Порівняння основних аспектів внутрішньої та аутсорсингової розробки програмного забезпечення” (Додаток Б) розглянуто різні функції та аспекти як внутрішньої, так і сторонньої розробки програмного забезпечення. Як внутрішні, так і зовнішні проєкти програмного забезпечення мають перевагу

один над одним у різних аспектах. У деяких аспектах власна розробка програмного забезпечення є кращою, тоді як в інших аспектах аутсорсинг має явну перевагу над власною розробкою.

Також можна виділити наступні переваги аутсорсингу перед власною розробкою в наступних аспектах організації роботи:

1. П
прорахунок потенційного бюджету. Для власної розробки досить характерною рисою є перевикористання бюджетів для виконання проєктів. Причина - всі потенційні ризики пов'язані з проєктною діяльністю (розробкою мобільного застосунку) буде нести сама організація. В той самий час правильно налагоджена співпраця у вигляді аутсорсингу даної розробки може значно полегшити планування необхідного бюджету на виконання поставлених задач, тому що в такому разі ризики діляться між замовником та виконавцем. Сам розподіл є доволі варіативним і залежить від умов контракту.

2. Н
аймання та навчання персоналу. Для того, щоб зібрати необхідну команду і впевнитись в тому, що вона достатньо кваліфікована для виконання необхідних завдань, плюс для того, щоб вона вийшла на стадію ефективної роботи, треба витратити доволі багато часу та ресурсів. При всьому цьому, завдяки аутсорсингу потенційно цілком можливо знайти доволі кваліфіковано команду з необхідною експертизою та яка вже працювала разом і над подібними проєктами. Це значно зменшує і час, і вартість старту роботи.

3. Пі
дтримка проєкту після його виконання. Підтримка проєкту після релізу є також доволі специфічною активністю і часто потребує спеціалістів, які мають досвід в цьому. І часто це не ті самі спеціалісти, які виконують першочергову розробку самого мобільного застосунку. Тому в такому разі також передача подібних активностей на аутсорсинг може бути цілком доцільною і зі сторони якості, і зі сторони витрат.

оступ до новітніх технологій. Нерідко маючи власну розробку компанія вимушена спеціалізуватись на визначених технологіях, що приносять результат тут і зараз. Тож на вивчення та апробацію найбільш сучасних варіацій просто не вистачає бюджетів та часу. На противагу, компанії, що надають послуги аутсорсингу часто витрачають доволі багато часу та ресурсів для того, щоб мати експертизу у найновіших та найефективніших технологіях. Тому долучення подібних спеціалістів може бути дуже доцільним і на етапі планування розробки, і під час самої розробки мобільних застосунків.

Але також важливо зазначити, що коли існує величезна культурна та мовна різниця між домашніми та аутсорсинговими організаціями, стиль спілкування та роботи може сильно відрізнятись.

Іноземці не можуть стати місцевими жителями, тому до них не можна ставитися так, як до організаційних працівників і мотивувати їх. Фактор мотивації працівників часто відсутній в аутсорсингу, на відміну від внутрішніх програмних проєктів, де кожен окремий співробітник працює за підвищення та винагороду, показуючи найкращу продуктивність.

Як внутрішні, так і аутсорсингові практики використовуються в усьому світі з останніх кількох десятиліть з усіма їхніми перевагами, недоліками та ризиками, пов'язаними з ними. У середовищах, де проєкти дуже складні, кошти занадто високі для їх створення, технології недостатньо розвинені, експертиза недостатня, аутсорсинг відіграє життєво важливу роль для задоволення очікувань. Це також відіграє ключову роль у виконанні завдання вчасно, завдяки цьому організації можуть заощадити витрати на виробництво та зосередитися на своїй основній діяльності. Це хороший інструмент для передачі ризику. Хоча, з іншого боку, це також пов'язано з деякими ризиками, такими як проблема конфіденційності даних, втрата потенційних співробітників і потенційних клієнтів.

Тож, оцінюючи переваги та недоліки основних моделей організації роботи над розробкою мобільного застосунку, ми схилиємося до використання

моделі аутсорсингу як основного інструмента для розробки. При цьому певні варіації з іншими моделями також можливі для того, щоб закривати певні ключові процеси.

2.2. Огляд технологій для розробки мобільного застосунку

Стек мобільної розробки поєднує всі технології, які використовуються для розробки функціонального мобільного додатку. Є чотири основні компоненти стека технологій для розробки мобільних застосунків:

- Платформа: основа програмного забезпечення або середовище, де розробляється та виконується мобільний додаток;

- Фронтенд або користувацький інтерфейс: мови програмування (наприклад, HTML5/CSS, JavaScript тощо), фреймворки (наприклад, Flutter, React Native, Xamarin тощо) і бібліотеки для реалізації дизайну інтерфейсу користувача та UX;

- Бекенд або серверна частина: компоненти та інфраструктура на стороні сервера, які забезпечують роботу мобільного додатка. Він керує зберіганням даних, обробкою, пошуком і реалізацією бізнес-логіки програми. Серверний рівень включає сервери, бази даних, API, бізнес-логіку, безпеку та автентифікацію, а також хмарні служби для використання масштабованої та надійної інфраструктури;

- Хостинг: необхідне програмне та апаратне середовище для запуску серверних компонентів і обслуговування функцій мобільного додатка.

Крім цих критичних компонентів, стек технологій для розробки мобільних додатків може включати веб-сервіси та API для мобільного додатка для взаємодії із зовнішніми службами шляхом отримання даних і виконання різноманітних операцій, DevOps-інструменти та практики для безперервної інтеграції, безперервної доставки (CI/CD), керування версіями та функціональність автоматичного розгортання.

Стек технологій мобільних додатків безпосередньо залежить від типу додатків, які обирає користувач. Існує чотири основні підходи до розробки мобільних додатків, і кожен із них передбачає власний набір потенційних технологій.

Розробка нативних мобільних додатків. Нативні мобільні програми розроблено спеціально для конкретної мобільної операційної системи з використанням власних мов програмування та фреймворків платформи. Це може бути Java або Kotlin для Android і Objective-C або Swift для iOS. Такі програми використовують унікальні (власні) функції та можливості конкретної платформи, для якої вони розроблені, як правило, Android або iOS.

У випадку нативного підходу, якщо потрібно, щоб програма могла працювати на кількох платформах, то потрібно створити окремі кодові бази для кожної платформи.

Популярні мови програмування для нативної розробки відповідно до платформи (ОС):

-	A
ndroid: Java, Kotlin;	
-	iO
S: Objective-C, Swift.	

Популярні фреймворки для розробки, що використовуються для нативних програм відповідно до платформи (ОС):

-	A
ndroid: Android Studio, Android SDK;	
-	iO

S: Apple Xcode, AppCode, iOS SDK.

Популярні фреймворки користувацького інтерфейсу, що використовуються в розробці нативних програм відповідно до платформи (ОС):

- Android

Android: Android UI, Jetpack Compose;

- iOS

S: Swift UI, UIKit.

Наразі 95% із 1000 найпопулярніших нативних програм для Android використовують код Kotlin, що робить його більш популярним вибором мови, ніж Java. Для платформи iOS схожа ситуація зі Swift - наразі це більш популярний варіант.

Типові випадки в яких доцільно використовувати саме нативні методи розробки мобільних додатків:

- П

рограми, які мають критичне значення для продуктивності: складні 3D-ігри, мультимедійні програми або програми, які значною мірою залежать від обробки даних у реальному часі;

- сп

еціальний користувацький досвід платформи: продукти, спеціально адаптовані до користувацького досвіду платформи, які вимагають власного підходу до розробки;

- П

рограми в яких ключовими є функціональність і продуктивність в автономному режимі: нативні мобільні програми, які забезпечують надійну функціональність в автономному режимі, повинні контролювати локальне зберігання даних, механізми кешування та фонову обробку.

Розробка гібридних мобільних додатків. Гібридний підхід передбачає створення однієї кодової бази, що працює на кількох мобільних платформах. Кодову базу створено за допомогою стандартних веб-технологій, таких як HTML, CSS і JavaScript. Код загорнутий у WebView – нативний контейнер. Вміст

у WebView відображається як звичайний веб-сайт. Такий підхід відносно обмежений у використанні функціональності апаратного забезпечення (GPS, камера тощо).

-

М

ови програмування: HTML, CSS, JS

-

Ф

реймворки розробки: Ionic, Cordova, Visual Studio

Типові додатки, в яких доцільно розробляти саме гібридні мобільні додатки:

-

СЛ

ужбові програми: програми без інтеграції з пристроєм або складних функцій (програми погоди, калькулятори, програми зі списками справ тощо);

-

В

нутрішні бізнес-програми: такі функції, як відображення даних, надсилання форм і доступ до внутрішніх систем можна повністю охопити за допомогою гібридного підходу до розробки.

-

програми для публікації контенту: блоги, журнали, програми новин та інші продукти, які вимагають узгодженого інтерфейсу користувача на різних платформах і частого оновлення вмісту.

Розробка кросплатформних мобільних додатків. Це ще один підхід до створення мобільного додатку, який може працювати на кількох платформах в одній кодовій базі. Він використовує фреймворки, що не залежать від платформи, і власні компоненти інтерфейсу користувача, щоб досягти нативної взаємодії з користувачем при спільному використанні тієї самої кодової бази. Кодова база написана на JavaScript і з'єднує нативні компоненти за допомогою так званих мостів.

-

М

ови програмування: JS, JSX, C#.

-
реймворки розробки: React Native, Xamarin, Flutter.

Типові додатки, для яких доцільно використовувати кросплатформову розробку додатків:

-
П
рограми для продуктивності: вони спрямовані на підвищення особистої або командної продуктивності, що не потребує залучення власного апаратного забезпечення та функцій. До таких програм належать списки справ, календарі, програми для керування проєктами та інші;

-
В
нутрішні бізнес-програми: продукти, які надають перевагу функціональності та швидким циклом розробки, а не вбудованим апаратним забезпеченням і безперебійній роботі користувачів на різних платформах;

-
КО
нтентні програми: агрегатори новин, платформи електронної комерції, блоги тощо.

Загалом кросплатформовий підхід може принести відчутну економію коштів для розробки корпоративних додатків.

Прогресивні веб-додатки. Прогресивні веб-програми або PWA нагадують гібридні. Вони створені з використанням веб-технологій і забезпечують нативний досвід для всіх підтримуваних пристроїв. PWA можна встановлювати та запускати на всіх пристроях за допомогою однієї кодової бази. Вони можуть запускатися з веб-браузера, як і веб-сайти, і можуть продовжувати працювати в автономному режимі. Цей тип програми адаптується до апаратних можливостей пристрою, на якому вони працюють, підтримуючи GPS, push-сповіщення, доступ до камери та інші функції. Однак технологія передбачає більше обмежень для платформи iOS: відсутність доступу до камери, відсутність можливості API встановлення тощо.

В додатку В “Порівняльна таблиця методів розробки мобільних додатків” наведено зведену таблицю для порівняння методів розробки мобільних додатків (розроблено автором).

Для вибору технологічного підходу до розробки мобільних додатків слід враховувати багато важливих факторів, серед яких:

- Аналіз цільової аудиторії: місцезнаходження, сферу використання та сегмент ринку.
- Вивчення каналів просування, щоб залучити клієнтів і змусити їх віддавати перевагу продукту над конкурентами.
- Охоплення ринку.

Виходячи з цільової аудиторії та специфіки сервісу, може знадобитися додаток лише для певної мобільної платформи. Якщо цільова аудиторія є мультиплатформовою, тоді слід розглядати нативну версію для Android та iOS або кросплатформову. Розташування також має значення. Наприклад, майже 62% європейських користувачів віддають перевагу Android, тоді як ринок США становить 55,4% iOS і 44,2% Android.

● М
ета програми: яку проблему цільової аудиторії вирішить продукт? Важливо розуміти, як саме конкуренти вирішують ту саму або подібну проблему, а також плюси і мінуси їх рішень. Підключення цього аналізу до стеку технологій мобільних додатків також може допомогти визначитися з відповідними технологіями.

● Ч
ас виходу на ринок: нативна розробка займає більше часу, ніж гібриди або PWA. Однак, прагнучи швидкого виходу на ринок, слід знизити очікування щодо продуктивності та UX.

● Б
юджет: у більшості випадків розробка кросплатформової програми дешевша, ніж розробка нативних програм. Якщо бюджет є вирішальним фактором бізнес-стратегії, то найкращий стек технологій для мобільних додатків —

кроссплатформовий або гібридний.



U

I/UX: рівень утримання мобільних додатків у день 1 становить 25,6% для iOS і 22,6% для Android. Враження від першого користувача має вирішальне значення для того, щоб клієнт продовжував користуватися додатком. Нативна розробка є кращою з точки зору інтерфейсу користувача. Кроссплатформовий підхід надає обмежені можливості інтерфейсу користувача, але все ж ширші, ніж гібридний.



П

продуктивність: ділові та функціональні вимоги визначають необхідну швидкість продуктивності та потужність обробки даних, які також слід враховувати при виборі технологічного стеку мобільних додатків.



Ф

ункціональність: залежно від того, як функціональність покладається на такі апаратні можливості, як датчики, камери, GPS та інші, гібридні або кроссплатформові підходи можуть не задовольнити всі потреби продукту.



Б

езпека: у певних галузях, враховуючи банківські та фінансові послуги, безпека має надважливе значення для безпечного обслуговування клієнтів і вдосконаленого захисту фінансових операцій, в тому числі біометричні дані, які дозволяють використання датчиків мобільного обладнання. Ось чому банки надають перевагу нативному стеку технологій для мобільних додатків.



М

асштабованість: вимоги до масштабованості програми впливають на стек технологій для мобільної програми, який може ефективно масштабувати серверну частину, обробляти збільшений трафік, оптимізувати продуктивність та інтегруватися із зовнішніми службами для підтримки довгострокового розвитку та успіху мобільної програми.

В результаті аналізу нами запропоновано наступний алгоритм для вибору підходу до розробки мобільного програмного забезпечення (рис. 2.1.). Слід

розуміти, що даний алгоритм є спрощеним і не враховує всі потенційні особливості кожного конкретного проєкту.

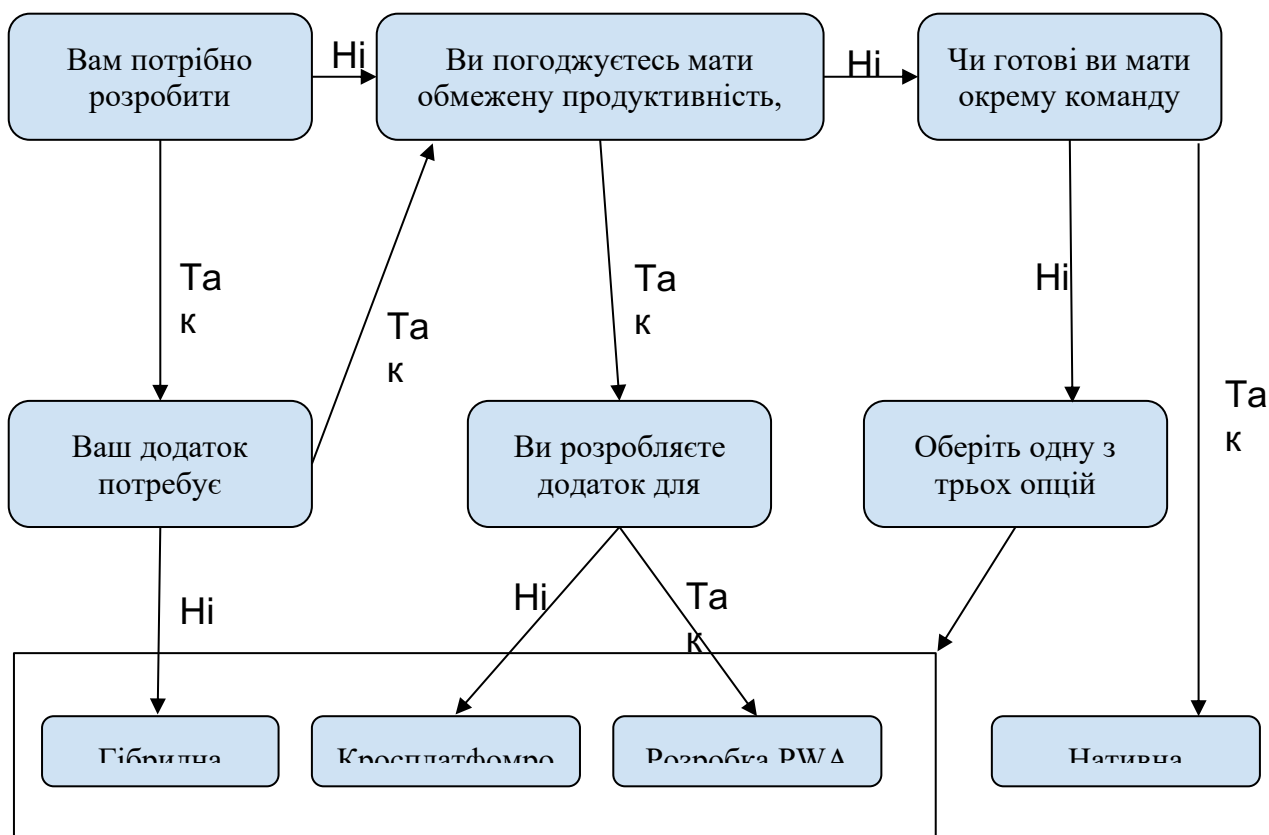


Рис. 2.1. Алгоритм вибору підходу до розробки мобільного застосунку.

Отже, щоб визначити найкращий стек технологій для мобільних додатків у межах вибраного підходу до розробки, спочатку слід вивчити усі вимоги на основі міркувань, наведених вище. Це допоможе нам зрозуміти, які засоби розробки та мови програмування в рамках обраного підходу краще підійдуть для майбутнього продукту.

Слід зазначити, що використання найпопулярніших технологій не завжди є найкращим вибором через можливу відсутність підтримки спільноти та доступних ресурсів для використання в процесі розробки. Іншим важливим фактором є розмір команди, який також залежить від обмежень бюджету. Одна кросплатформова команда буде коштувати менше, ніж дві команди для нативної розробки (за умови розробки для двох операційних систем). Тому це ще один аргумент обрати кросплатформову розробку перед нативною, якщо ресурси є обмеженими, а функціональні вимоги задовольняються повністю без

необхідності розробки повністю нативними методами.

Запуск нового продукту часто починається з розробки MVP (мінімально дієздатний продукт). Це необхідно щоб перевірити життєздатність ідеї продукту перед тим, як інвестувати в розробку повної версії продукту. На цьому етапі слід обрати економічно ефективний стек технологій для мобільного додатка, щоб реалізувати основні функції та базовий інтерфейс користувача. Крім того, якщо проєкт виявиться життєздатним, можна перейти на нативний стек або продовжити розширення функцій продукту за допомогою обраної гібридної або кросплатформової технології.

Отже, нами було проаналізовано найбільш популярні і ефективні методи розробки мобільного програмного забезпечення. На основі даного аналізу також було розроблено алгоритм для вибору методу розробки мобільного програмного забезпечення виходячи з верхньорівневих характеристик самої програми та бізнес контексту замовника.

2.3. Огляд технологій для відстеження показників фізичної активності

Фізіологічні та антропометричні вимірювання є важливою частиною науки про спорт і фізичні вправи, оскільки їх можна використовувати для моніторингу, оцінки та розробки програм тренувань. Умови випробувань можна жорстко контролювати в лабораторних умовах, які можна відтворити з відносно відомим рівнем точності за допомогою документації для перевірки надійності. Можливим обмеженням цих тестів є відсутність валідності. Практики часто покладаються на польові тести для вимірювання та оцінки продуктивності, щоб підвищити обґрунтованість для спортсмена. Максимальна мобільність обладнання, необхідного в польових умовах, допоможе фахівцю, а прогрес у технології означає, що менші технології здатні на набагато більше. Опублікована стаття Кардінале та Варлі [68] розглядає носимі технології для моніторингу тренувань, такі як пристрої глобальної системи позиціонування (GPS), монітори

серцевого ритму та акселерометри. Однак для деяких технологій не потрібні переносні пристрої, а лише сам мобільний пристрій для збору даних через завантажувані програми (додатки). Завдяки деяким останнім досягненням, тренери можуть збирати більшість своїх даних лише за допомогою мобільного пристрою.

З точки зору збору фізіологічних даних мобільні пристрої можна використовувати двома основними способами:

- діючи як реєстратор даних та інтерфейс для периферійного підключення;
- використовуючи зовнішні датчики (наприклад, мікрофон, камеру) і внутрішні процесори самого пристрою для збору та інтерпретації сигналів.

Частота серцевих скорочень є основним фізіологічним вимірюванням у спорті, здоров'ї та фізичних вправах. Критерієм, або «золотим стандартом», залишається електрокардіограма (ЕКГ), яка може бути непрактичною в польових умовах. Ряд телеметричних пристроїв перевірено на ЕКГ для використання в більш практичних ситуаціях [69], однак ці пристрої також пов'язані з витратами на самі пристрої. При цьому розміщення нагрудного ремня може вважатися нав'язливим для спортсмена. Крім того, вимога щодо додаткового обладнання може обмежити широке використання [70]. Особливо це може мати місце в середовищах, пов'язаних зі здоров'ям, наприклад у фітнес-центрах і реабілітаційних відділеннях. Практикуючі лікарі в цих областях можуть мати лише ручні методи пальпації, які, як було доведено, є неточними [71,72]. Саме в таких випадках технологія повсюдно доступних мобільних пристроїв може бути корисною. Найпростіша програма для полегшення вимірювання частоти серцевих скорочень діє подібно до метронома, коли на екрані стукають щоразу, коли пальпують пульс. Імовірно, цей метод призначений для зменшення помилок шляхом розділення завдань пальпації та підрахунку. Однак Перт та ін. [73] виявили, що одна така програма на iOS iPad («Tap the Pulse» від Orangesoft LLC) мала більшу розбіжність із телеметрією.

Більш новітні вимірювання використовують технологію, відому як фотоплетизмографія (PPG). PPG — це технологія, яка наразі використовується в пульсоксиметрах на кінчиках пальців, і працює на основі того, що коли капіляри наповнені кров'ю, світло перекривається, і більше світла може проходити крізь нього, коли кров відходить. Пелегрін та інші дослідники [74] пояснюють, що саме ця зміна середньої яскравості діє як сигнал для пристрою, який інтерпретує та витягує показники пульсу. Ті ж автори намагалися перевірити свою технологію, яка обчислювала частоту серцевих скорочень, взяту з потоку кадрів зображень, коли палець притискав об'єктив камери та спалах мобільного телефону, у порівнянні з пульсоксиметром. Необроблені дані надані в статті, і кореляція між методами була розрахована, як помірна, із середньою різницею між методами в чотири удари на хвилину. Попеску та ін. [75] і Лоза-Іглесіас та ін. [76] оцінили можливості двох комерційно доступних додатків, які працювали на одній передумові застосування кінчика пальця до камери пристрою та спалаху. Попеску ін. [75] порівняли «Cardiowatch» Радю Іонеску на iPhone з апаратом ЕКГ, а Лоза-Іглесіас та інші [76] порівняли «Heart Rate Plus» від AVDApps на телефоні Samsung Galaxy Note з пульсоксиметром. При цьому обидва дослідження показали типову різницю $\pm 3-4$ уд/хв між методами вимірювання.

Хоча контактна технологія PPG, здається, здатна відносно точно вимірювати частоту серцевих скорочень у стані спокою, дані Вакель та ін. [109] свідчать про те, що похибка може зростати зі збільшенням частоти серцевих скорочень. Ці автори повідомили, що показники спокою, виміряні за допомогою програми Azumio та Bio2imaging на iPhone, були в межах ± 4 ударів на хвилину від вимірювання ЕКГ у педіатричних пацієнтів [74-76]. Однак, коли додатки використовувалися під час періоду тахікардії (156–272 ударів за хвилину), середня різниця порівняно з ЕКГ зросла до 18 ударів за хвилину, а кореляція зменшилася. Це має очевидні наслідки для спорту та фізичних вправ, оскільки вимірювання серцевого ритму, ймовірно, відбуватимуться після тренування.

У той час як тахікардичний діапазон, засвідчений Вакелем та

співавторами [109], становив від 156 ударів за хвилину, більшість із них перевищувала 200 ударів за хвилину. Хо та інші [77] вимірювали частоту серцевих скорочень у 126 дітей, які потрапили до лікарні, за допомогою чотирьох різних додатків на iPhone на мочці вуха та кінчику пальця разом із ЕКГ. Частота серцевих скорочень із додатків більш тісно корелювала з ЕКГ на мочці вуха, а не на пальці. Додаток A значно перевершив три інші додатки з аномальними результатами, які починалися приблизно з 160 ударів на хвилину. Єдине відоме дослідження для перевірки контактної технології PPG на мобільних пристроях після фізичних вправ було проведено Мітчеллом та ін. [78]. Учасникам вимірювали частоту серцевих скорочень у стані спокою та після 1-хвилинного покрокового тесту, таким чином відтворюючи умови, за яких технологія, ймовірно, найімовірніше буде використовуватися. Вимірювання проводилися за допомогою програми від Azumio, яку використовували Вакел та інші [109] на телефоні iOS і Android, а також телеметричний нагрудний ремінь Polar. Автори дійшли висновку, що обидві платформи можна використовувати з упевненістю.

Конг та інші [79] припустили, що PPG можна зробити більш точним за допомогою безконтактних методів, оскільки контактна сила на датчику може впливати на форму сигналів. Безконтактний PPG за допомогою веб-камери на ноутбучі був описаний Похом [80]. Ця технологія працює за принципом, схожим з контактним PPG, але замість цього спостерігає відеозапис обличчя. Кілька безкоштовних додатків використовують цей безконтактний метод PPG і вказують користувачам тримати камеру пристрою перед обличчям, доки не буде знято показання. Пірт та інші [73] досліджували дві безконтактні програми PPG у спокої на iPad mini від ViTrox Technologies і Cardio Inc, повідомляючи про середню різницю в один і два удари в порівнянні з телеметричним монітором Polar за хвилину. Частина серцевих скорочень сама по собі може представляти обмежений інтерес для деяких практикуючих, і натомість багатьох може більше цікавити регулярність серцевих скорочень. Анотація з обмеженою інформацією від Сардана та ін. [70] повідомляє про високу чутливість і надійність програми iPhone для визначення фібриляції передсердь (ФП). МакМанус і його колеги

описують програми, які можуть ідентифікувати ФП, а також передчасні скорочення передсердь (ПСП) і передчасні скорочення шлуночків (ПСШ) [81,82]. Хоча такі вимірювання можуть не представляти широкого інтересу для вчених у галузі спорту та фізичних вправ, здатність визначати регулярність буде корисною для моніторингу рівня адаптації до тренувань [80]. Наразі існує обмежена кількість засобів для вимірювання подібних показників лише за допомогою мобільного пристрою, хоча в деяких дослідженнях описується дійсне вимірювання за допомогою нагрудного ремня або периферійних пристроїв *ithlete* (HRV Fit Ltd), які приєднуються до мобільного телефону [83,84], достатньо чутливі, щоб відстежувати зміни протягом трьох тижнів [85].

Також на сьогодні подібні технології активно використовуються у наручних смарт-годинниках та фітнес браслетах. Це є найбільш популярним та доступним варіантом для спортсменів різного рівня. При цьому, враховуючи вищенаведені дослідження, ми можемо говорити про достатньо непогані результати подібного методу.

Вимірювання дихання. Фолке та ін. [86] припускають, що дихальний об'єм і частота дихання є двома основними життєво важливими ознаками, які слід забезпечувати моніторингом дихання. Методи реєстрації зазвичай включають використання спірометра, який може бути або портативним (наприклад, ручним). Показники можна отримати шляхом простого спостереження за людиною або за допомогою більш складних процедур, таких як газовий аналіз дихання або трансторакальний імпеданс. У той час як Реес та ін. [87] визнають існування клінічних вимірювань об'єму та частоти дихання, вони також підкреслюють обмеження та недоліки існуючого обладнання, зокрема обмежений доступ за межами клінічних та/або дослідницьких установ. Інші обмеження існуючих методів включають високу вартість, спеціальний персонал і відсутність мобільності [87,88]. Дихальну функцію можна оцінити різними способами за допомогою різного апаратного забезпечення смартфона, включаючи камеру, мікрофон і акселерометр. Реес та ін. [87] використовували фронтальну камеру смартфона з операційною системою Android, щоб отримати

сигнал руху грудної клітки, який продемонстрував сильний зв'язок зі спірометром під час запису. Нам та ін. [88] продемонстрували подібні висновки, завершивши точну оцінку частоти дихання на тому самому пристрої. Однак, хоча Реєс та ін. [87] не виявили статистично значущого зміщення в записі показників, автори поставили під сумнів, чи оцінка помилки була прийнятною для домашнього використання. Попри те, що дослідження продемонструвало надійність і обґрунтованість оцінки показників дихання, існували обмеження, притаманні безконтактним оптичним процедурам. Крім того, Нам та ін. [88] припустили, що одяг впливав на відеосигнал, наприклад, простий дизайн порівняно зі смугастим або неоднорідним дизайном викликав менші відносні зміни в записаних рухах грудної клітки та живота. Крім обмежень збору та обробки даних, безконтактні оптичні процедури для оцінки дихальних параметрів не мають практичного застосування для більш загального використання. Обидва Реєс [87] і Нам [88] продемонстрували правильне та надійне використання апаратного забезпечення смартфона для запису параметрів функції легень. Однак, жоден автор не досліджував валідність і надійність конкретного програмного забезпечення для смартфонів, яке є комерційно доступним для загального використання.

Антропометрія та діапазон рухів. Склад тіла оцінювали кількома способами, включаючи гідростатичне зважування [89] і рентгенівську абсорбціометрію з подвійною енергією [90], з деякими розбіжностями щодо золотого стандарту. Існує, однак, згода, що ці методи представляють труднощі, такі як витрати, споживання часу, доступ і портативність [91,92]. Таке обладнання, як правило, обмежується університетськими лабораторіями та дослідницькими установами, і тому до нього важко отримати доступ для деяких практиків, таких як працівники первинної медичної допомоги, дієтологи, інструктори з фітнесу та персональні тренери. З розвитком технологій з'являється потенціал для більш економічно ефективних рішень у вимірюванні та оцінці складу тіла. Фаріна та ін. [93] розглядають 2D-зображення, використовуючи фронтальні та бічні зображення, отримані зі стандартної

цифрової камери, як альтернативу дорогим 3D-системам. Використання двовимірних зображень для отримання точних антропометричних даних не є новою розробкою [94]. Більш пізні застосування оцифровки двовимірних зображень для забезпечення антропометрії включають вимірювання рук для виробництва робочих рукавичок [95]. Однак ці застосування двовимірних зображень забезпечують лише вимірювання поверхні й не роблять висновків щодо складу тканини. Фаріна та ін. [93] досліджували використання камери, вбудованої в смартфон, для отримання цифрових зображень всього тіла для оцінки складу тіла людини, виявивши незначну різницю в 0,02 кг і 0,07 кг в оцінюваній масі жиру між додатком і клінічними дослідженнями в жінки та чоловіки відповідно. Дослідження використовувало програмне забезпечення, яке використовує фотографії, зроблені за допомогою смартфона або планшета, щоб оцінити відсоток жиру в організмі шляхом оцифрування серії обхватів. Шоу та інші [102] оцінили надійність цього програмного додатка на iPad mini щодо вимірювань шкірної складки та біоелектричного опору. Не було суттєвих відмінностей між методами для розрахункового відсотка жиру в організмі, а коефіцієнти кореляції всередині класу показали, що надійність кожного методу є хорошою. Попередні дослідження вивчали використання смартфонів у більш прикладних контекстах антропометрії, таких як оцінка постави. PostureScreen Mobile - це програма для смартфонів від тієї ж компанії, яка створила LeanScreen (PostureCo Inc., Трініті, штат Флорида, США), яка оцінює поставу за допомогою двовимірних фотографій, зроблених смартфоном або планшетом. Боланд та ін. [96] вивчили PostureScreen та дійшли висновку, що отримали прийнятні результати для трьох різних екзаменаторів з різним досвідом. З 13 постуральних показників, які забезпечує PostureScreen Mobile (зміщення голови вбік, зсув голови вбік, нахил голови, зміщення плечей вбік, поздовжнє зміщення плечей, нахил плечей, зміщення грудної клітки, зміщення стегна вбік, поздовжнє зміщення стегна, нахил стегна, вага голови, ефективна вага голови та зміщення колін). Боланд та ін. [96] раціоналізували своє дослідження, припустивши, що вимірювання з додатка будуть мати цінність, лише якщо вони можуть бути

надійними в багатьох випробуваннях. Враховуючи, що PostureScreen Mobile комерційно доступний для громадськості, надійність цього додатка може бути поставлена під сумнів на основі дослідження Боланда та ін. (2016). Додаток «Scolioscreen» є доповненням до фактичного Scolioscreen, який є конструкцією сколіометра для розміщення будь-якого смартфона, що містить обладнання для вимірювання нахилу. Виробники заявляють, що Scolioscreen можна поєднувати з будь-яким додатком, який вимірює нахили.

Програми на основі акселерометра можуть стати ефективною альтернативою звичайному гоніометру з довгою рукою. У цих програмах використовується апаратне забезпечення акселерометра, вбудованого в смартфони, яке традиційно служить датчиком положення для використання у відеоіграх шляхом вимірювання нахилу смартфона [97]. Окендон і Гілберт [97] продемонстрували високу надійність і валідність програми на основі акселерометра для смартфона iPhone. Крім того, автори також виявили більшу надійність між вимірювачами порівняно з традиційним гоніометром. Враховуючи те, що більшість практиків, які зазвичай оцінюють діапазон рухів (наприклад, фізіотерапевти, тренери з силової підготовки), робили б це незалежно, можна стверджувати, що надійність між оцінювачами не має відношення до цього контексту. Проте те саме дослідження продемонструвало вищу надійність порівняно з традиційним методом, пропонуючи підтримку додатків на основі акселерометра як життєздатну альтернативу традиційним методам гоніометрії. Мілані та ін. [98] стверджують, що програми на основі акселерометра, фотографії та магнітометра мають однакові обмеження в тому, що вони можуть вимірювати діапазон руху лише в статичних умовах. Тому, щоб додатки для смартфонів розглядалися як ефективна альтернатива, вони повинні бути здатними вірогідно та надійно вимірювати кутовий рух у динамічних умовах, наприклад активні оберти. Нещодавно Бітель та ін. [99] використовували акселерометр iPhone для вимірювання рухів розгинання та згинання одночасно з ізокінетичним динамометром у діапазоні різних швидкостей (30, 60, 90, 120 та 150° на секунду). Автори продемонстрували межі збігу 2° між смартфоном і

динамометром.

Підбиваючи підсумки, попередні дослідження продемонстрували надійність, а також валідність додатків для гоніометрів на основі фотографії, акселерометра та магнітометра.

Однією з головних проблем, з якою стикаються тренери з силової підготовки та підготовки, є те, як об'єктивно оцінити фізичні можливості своїх спортсменів [100,101]. Вимірювання фізичної працездатності дійсно є ключовою частиною будь-якої програми тренувань, оскільки воно дозволяє контролювати та коригувати навантаження [102,103], аналізувати втому [104], виявляти сильні сторони [105,106], визначати слабкі сторони [(97).), або запобігти травмам [108-110]. Таким чином, поширеною практикою при розробці силових і кондиційних програм є вимірювання конкретних змінних, що представляють інтерес, які можуть допомогти у призначенні тренувального стимулу [111]. Однак технологія та досвід, необхідні для цього, часто є дорогими та незручними, особливо для тренерів або команд за межами великих організацій чи університетів. Для цього зростання кількості смартфонів, які наразі містять кілька датчиків, спеціально розроблених для вимірювання фізичної працездатності (наприклад, моніторів серцевого ритму, GPS або акселерометрів), набирають популярності у спільноті фітнесу та здоров'я [112]. Наприклад, додатки для фітнесу та здоров'я є одними з найкращих фітнес-трендів у списку, складеному Американським коледжем спортивної медицини [112]. Однак лише деякі з тисяч доступних фітнес-додатків підтверджені науково [113].

Максимальна сила. Рекомендації щодо силових тренувань базуються на добре відомій парадигмі максимальної ваги на 1 повторення, згідно з якою інтенсивність розраховується як відсоток від максимального навантаження, яке спортсмен може підняти лише один раз [101,114]. Однак вимірювання максимуму 1 повторення вимагає виконання максимального підйому, що може не підходити для всіх груп населення, особливо для тих, хто має невеликий досвід у піднятті важкої ваги, оскільки це може призвести до неточних

результатів і збільшити ризик травм [111]. Тому для прогнозування максимуму 1 повторення із субмаксимальними навантаженнями використовували кілька альтернатив, наприклад виконання повторень до відмови або використання швидкості сприйманого навантаження [115,116]. Проте вважалося, що найточніші методології складаються з вимірювання швидкості штанги. Це пов'язано з тим фактом, що було широко продемонстровано, що існує дуже сильний зв'язок між максимальною вагою на 1 раз і швидкістю, з якою кожна вага підіймається [111]. Таким чином, нова парадигма тренування з опором, яку часто описують як тренування на основі швидкості, з'явилася на основі систематичних вимірювань швидкості штанги для коригування та призначення інтенсивності тренування, оскільки кожен показник максимального повторення на 1 раз має певний діапазон швидкості [111]. Золотим стандартом для вимірювання швидкості штанги є високочастотні лінійні датчики [111]. Однак його вартість, яка в більшості випадків перевищує 2000 доларів, не дозволяє використовувати його в невеликих організаціях або клубах з невеликими ресурсами.

Намагаючись усунути це обмеження, додаток для iOS під назвою «PowerLift» нещодавно було підтверджено для вимірювання швидкості штанги у вправі жиму лежачи у чоловіків, які тренують силу [117]. Для цього автори одночасно виміряли кілька повторень у групі пауерліфтерів за допомогою лінійного датчика (працює на частоті 1 кГц) і програми «PowerLift» на iPhone. А потім порівняли результати. «PowerLift», який складається із запису та остаточного аналізу уповільненого відео підйому завдяки високошвидкісній камері на найновіших пристроях iOS, суттєво корелював із лінійним датчиком і показав достатньо незначну стандартну помилку оцінки при вимірюванні швидкості штанги.

М'язова сила або імпульс: висота вертикального стрибка. Вимірювання висоти вертикального стрибка широко використовується в літературі для оцінки м'язової сили або аналізу нервово-м'язової втоми [117,118]. Враховуючи, що вертикальні стрибки є важливою здатністю в багатьох видах спорту [112], їх

вимірювання часто є ключовою частиною будь-якого аналізу здібностей та фізичного рівня. Було запропоновано кілька підходів для вимірювання висоти, яку спортсмен може досягти під час вертикального стрибка [119], хоча найбільш точні зазвичай включають вимірювання швидкості відриву або часу польоту під час стрибка. Це тому, що ці параметри можуть обчислити вертикальне зміщення центру маси за допомогою добре відомих рівнянь Ньютона [117]. Хоча силові платформи часто вважаються золотим стандартом для вимірювання вертикальних стрибків шляхом вимірювання швидкості відриву спортсмена [118], кілька систем, заснованих на визначенні часу польоту (таких як інфрачервоні платформи), стали популярними у спільноті спортсменів, оскільки вони дешевші, більш портативні та все ще можуть забезпечити дуже точні вимірювання висоти стрибка [112,119]. Однією з таких систем є програма для iPhone («My Jump»), яка вимірює час польоту під час стрибка завдяки можливостям уповільненого запису на iPhone [112]. За допомогою простого відеоаналізу, під час якого користувач візуально визначає зліт і приземлення під час стрибка в додатку, «My Jump» обчислює час польоту під час стрибка точним і надійним способом. Продуктивність програми була широко підтверджена в літературі протягом останніх років, демонструючи рівні кореляції вище 0,96 і систематичне зміщення менш як 10 мм у порівнянні з еталонними системами [112].

Пересування людини: біг і спринт. Аналіз локомоції людини представляє великий інтерес як для працездатності, так і для запобігання травмам [110,121]. Дослідження показали, що асиметрія між ногами може бути використана як релевантний показник, пов'язаний з ризиком травми [122]. Для вимірювання механіки бігу та спринту зазвичай потрібні передові системи вимірювання, такі як інструментальні бігові доріжки, силові платформи, хронометражні ворота або радарні пристрої [123]. Дорогі технології, до яких більшість тренерів не мають доступу. Використовуючи той самий підхід, що й у згаданих вище додатках для стрибків і опору, дві нові програми, також засновані на високошвидкісному відеоаналізі, нещодавно перевірили для вимірювання механіки бігу та спринту

за допомогою iPhone [123]. Перший, «Runmatic», був протестований на інфрачервоній платформі для визначення часу контакту та польоту під час бігу на декількох швидкостях у діапазоні 10-20 км/год у чоловіків-бігунів [124]. Крім того, програма використовувала деякі підтверджені рівняння моделі пружинної маси, які дозволяють розрахувати різні механічні змінні на основі часу контакту, часу польоту та простих антропометричних показників [121]. Було показано, що додаток є надійним для вимірювання жорсткості ніг, вертикальних коливань центру маси, максимальної сили, прикладеної до землі, і частоти кроків. Другий, «My Sprint», також продемонстрував свою вірогідність і надійність для вимірювання часу спринту на 30 метрів, горизонтальної сили, швидкості та потужності у спринтерів-чоловіків у порівнянні з хронометражами та радарним пістолетом, без істотних відмінностей між пристроями [123]. Таким чином, ці програми дозволяють практикуючому вимірювати важливі змінні, пов'язані з бігом і максимальним спринтом, без потреби в будь-яких передових інструментах.

Відстеження відстані за допомогою GPS і датчиків акселерометра. Коли говорити про біг, мабуть, найпопулярнішою змінною в індустрії спортивних технологій є відстань, пройдена за допомогою сигналів GPS і, відповідно, темп бігу [125]. Кілька переносних пристроїв (переважно годинників) використовувалися як на практиці, так і в дослідженнях для вимірювання дистанцій бігу [126,127], хоча включення GPS-датчиків у більшість смартфонів за останні роки стало каталізатором створення додатків, які використовують переваги цієї технології, щоб відстежувати дистанції та темп бігу (24). Нещодавнє дослідження проаналізувало валідність і надійність додатка для iOS, призначеного для вимірювання дистанцій під час бігу за допомогою GPS, який входить до складу смартфонів iPhone [129]. Для цього дослідники змусили піддослідних пробігти 400-метрову доріжку загалом 2400 метрів з iPhone на пов'язці, а потім порівняли значення відстані та швидкості, отримані додатком, із фактичними значеннями. Додаток занижував і відстань, і швидкість на 3-4%, що означає абсолютну різницю приблизно в 100 м або 0,7 км/год. Проте

спостережена хороша надійність тестування та повторного тестування (тобто порівняння значень у двох окремих дослідженнях) і відносно низьке зміщення між додатком і фактичною відстанню змусили авторів зробити висновок, що додаток може бути доречним для відстеження бігу в загальній популяції, хоча він може бути недостатнім для тренуваних спортсменів.

Ще одна широко поширена змінна, пов'язана з ходьбою або бігом, — кількість кроків [130,131]. Зокрема, було запропоновано, що мінімальна кількість кроків у 10 000 на день пов'язана з хорошими рівнями щоденної фізичної активності та стану здоров'я [130,131]. Для цього багато найпопулярніших на ринку пристроїв, які можна носити, зосереджені на відстеженні кроків, використовуючи дані про прискорення, щоб надати користувачам інформацію про кількість кроків [132]. Звичайно, оскільки смартфони містять акселерометри, були розроблені буквально тисячі додатків для відстеження кроків, які підраховують кроки користувачів без використання зовнішніх пристроїв. Однак нещодавнє дослідження показало, що цим програмам бракує точності порівняно з професійним крокоміром, ймовірно, через низьку якість акселерометрів, які входять до складу більшості смартфонів [133]. У цьому дослідженні порівняли еталонний крокомір із трьома додатками для відстеження кроків на базі Android («Runtastic», «Pacer Works», «Tayutau») у лабораторних умовах, а також власний смартфон кожного учасника в умовах вільного життя. Три додатки значно занижували або перевищували підрахунок кроків на 16-50% і показали низький рівень узгодженості з еталонним методом, тому дослідники дійшли висновку, що цей тип додатків не можна рекомендувати для відстеження кроків у їхньому поточному стані.

Отже, підсумовуючи, можна сказати, що наявні технологічні можливості в смартфонах та смарт-годинниках чи фітнес-браслетах надають достатньо багато можливостей для отримання даних про людину. Більшість цих даних може бути достатньої точності для їх використання у розрахунках необхідних показників та аналізі фізичного стану піддослідного.

2.4. Постановка задачі

Оцінюючи наведену вище інформацію, ми абсолютно точно можемо зробити висновок, що розробка мобільних застосунків має величезний ринок, а запит користувачів все продовжує рости. При цьому технології мобільних телефонів, що також постійно покращуються, створюють нові можливості на даному ринку та дозволяють випускати застосунки зі все прогресуючим функціоналом, який може покривати більше проблематик, або робити це якісніше ніж до цього. Саме тому темою даної роботи й було вибрано розробку детального плану створення мобільного застосунку.

Обираючи сферу для розробки мобільного застосунку враховувалися наступні факти:

- на явна експертиза;
- о б'єм ринку;
- п латоспроможність та готовність платити за сервіс потенційних користувачів.

Виходячи з цього, було обрано розробку мобільного застосунку для школярів та студентів, які займаються спортом на регулярній основі. Основним цільовим ринком обрано Сполучені Штати. Основна причина - серйозне відношення до спортивних активностей серед американських школярів та студентів, відносна платоспроможність та наявність великого ринку: на сьогодні школярів, які займаються регулярно спортом (беруть участь у змаганнях) - понад 8 млн, студентів вищих навчальних закладів - понад 550 тисяч.

Проте, як зазначалось у попередніх розділах, власне розробка програмного застосунку для мобільного пристрою - це лише незначна частина

роботи. Багато уваги та зусиль слід приділити також плануванню цього процесу. Тому для досягнення мети, поставленої в даній роботі, необхідно наступне:

- А
аналізу потенційних конкурентних переваг: це буде досягнуто через формування верхньорівневого списку функціонала сервісу та виділення його конкурентних особливостей порівняно з наявними аналогами на ринку.

- П
лануванню тестування гіпотез: створення алгоритму та етапів оцінки привабливості нашої ідеї для потенційних користувачів сервісу.

- Р
озрахунок економічної моделі отримання прибутку від застосунку: обрання моделі монетизації застосунку та обґрунтування даного вибору, розрахунок цільового рівня доходу, а також розрахунок необхідної кількості клієнтів для досягнення цього цільового рівня;

- Ф
ормування структури команди для створення продукту: оцінка необхідних людських ресурсів, їх кваліфікації та технічних можливостей, спеціалізації та рівня залучення на відповідних етапах створення застосунку;

- Р
озрахунок потенційних витрат на створення подібного продукту: дана частина тісно пов'язана з розрахунком цільового рівня прибутку та включатиме розрахунок витрат на використання необхідних ресурсів як в процесі розробки застосунку, так і на його підтримку після випуску та на просування;

- Р
озробка механізму оцінки ефективності реалізованого проєкту: визначення ключових метрик успішності проєкту та їх показників, а також їх обґрунтування;

-

В

визначення верхньорівневої архітектури проєкту: виділення ключових складових системи та створення схеми їх взаємодії між собою;

-

О

брати технології для реалізації продукту: має включати засоби та технології для розробки клієнтської частини продукту, серверної частини, бази даних, інтерфейсу адміністратора системи;

-

О

брати методологію для організації роботи над застосунком: ця частина тісно пов'язана з визначенням структури команди для роботи над проєктом та має включати визначення необхідних ролей в цій команді та моделей їх взаємодії між собою для досягнення необхідного результату в найбільш ефективний спосіб.

За умови досягнення поставленої задачі й наявності усіх вищеперелічених складових ми очікуємо отримати дієздатний план для створення конкурентного сервісу, що цілком спроможний вирости в успішний бізнес-проєкт в майбутньому. При цьому даний план має бути достатньо гнучкий та адаптивний для того, щоб відповідати умовам ринку, мінливості зовнішніх факторів та загального контексту.

РОЗДІЛ 3

ПРАКТИЧНА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ

3.1. Архітектура програмного застосунку та аналіз тенденцій у сфері

Архітектура програмного забезпечення описує та демонструє зміст створеного програмного забезпечення. Форма опису може бути різною. Різні потреби в інформації та вимоги до архітектури спричинили різноманітність у визначеннях архітектури як такої. Одним із визначень архітектури програмного забезпечення є опис зв'язку між статичними та динамічними частинами програмного забезпечення. Іншими словами, це натякає на дві важливі характеристики комп'ютерної програми: ієрархічну структуру процедурних компонентів і структуру даних. У 90-х роках архітектура програмного забезпечення стала однією з найважливіших частин розробки програмного забезпечення. Різні технології, які обробляються особливо з точки зору архітектури, поширювалися на промислове використання. Крім того, постійно розробляються складніші програми, які включають тисячі рядків коду, які складно підтримувати та повторно використовувати. Це реальні проблеми, які спричинили появу різних архітектур програмного забезпечення. Таким чином, визначення архітектури програмного забезпечення сьогодні можна розглядати як критичну частину розробки програмного додатка. Кордон між архітектурою та структурою найнижчої частини програмного забезпечення є тонким, що може вплинути на збіги в проєктуванні викликаних різними підходами до описання цих частин. Одне з визначень може стверджувати, що архітектури програмного забезпечення включають ті функції програмного забезпечення, які не змінюються під час процесу розробки як такої.

Моделювання та документування архітектури є важливим з багатьох причин. Зацікавлені сторони можуть дивитися на структурні особливості програмного забезпечення за допомогою моделювання та документації. Використовуючи архітектурну модель, розробники можуть аналізувати

програмне забезпечення на дуже ранній стадії процесу розробки. Архітектура є першим кроком у розробці самого проєкту, і вона визначає стабільну основу для подальших етапів. Різні моделі архітектури можна стандартизувати та називати, а також повторно використовувати в багатьох наступних програмах. Це призводить до ідентифікації та документування загальних моделей архітектури, які не залежать від додатків чи кінцевих застосунків. Архітектурні моделі називаються архітектурними стилями. Прикладами архітектурних стилів є багатошарова архітектура та архітектура сховища. Рівнева архітектура програмного забезпечення означає структуру програмного забезпечення, яка складається з рівнів, логічно подібних на різних рівнях абстракції. Функція багаторівневої архітектури програмного забезпечення полягає в тому, що вищий рівень може використовувати послуги, які виробляє нижній рівень. Зазвичай це призводить до ситуації, коли нижні рівні виробляють найбільш поширені послуги. З іншого боку, це означає, що чим вищий рівень описується, тим більш незалежним є рівень від самої програми. Багаторівневу архітектуру програмного забезпечення можна описати в трирівневій архітектурній моделі, яка складається з рівня програми, рівня проміжного програмного забезпечення та рівня платформи. Проміжне програмне забезпечення пропонує незалежні від платформи загальні служби, такі як підтримка інтерфейсу користувача. Рівень платформи пропонує, наприклад, операційну систему, комунікаційне програмне забезпечення та інше програмне забезпечення, що залежить від апаратного забезпечення (драйвери тощо).

Розробка архітектури мобільного додатку має бути обов'язковим кроком у всіх проєктах, спрямованих на створення цифрових продуктів Android та iOS. Але в більшості випадків цьому не приділяють належної уваги. Ось лише деякі проблеми, які можуть виникнути, якщо ігнорується архітектура мобільного додатка:

- системні помилки;
- складність обслуговування в майбутньому;
- нижча читабельність коду;

- складнощі в тестуванні та рефакторингу;
- недостатній потенціал масштабованості/

Розробка мобільного додатку без попереднього створення архітектури — це все одно, що будівництво будинку без технічного креслення. Так, це може заощадити вам багато часу.

Хоча якість архітектури залежить від багатьох факторів, є деякі речі, які можуть значно покращити результати в більшості випадків. Нижче наведено контрольний список пунктів, які ви повинні враховуватись:

- Тип пристрою. Параметри мобільних пристроїв впливають на роботу додатка. Ось чому команда слід передбачити, як цифровий продукт виглядатиме та працюватиме на кожному цільовому екрані. Програмісти також повинні брати до уваги характеристики процесора, обсяг доступної пам'яті, місткість пам'яті та інші важливі аспекти всіх або принаймні найпопулярніших моделей телефонів, на яких планується використання додатка.

- Сценарії пропускної здатності. Якість підключення до Інтернету є ще одним важливим аспектом, який слід враховувати при створенні архітектури мобільного додатку. Хоча ніхто не може передбачити доступність мережі Інтернет у кожному цільовому регіоні, можна підготувати своє рішення до найгіршого можливого сценарію. Якщо це зробити, взаємодія з користувачем не постраждає, навіть якщо швидкість Інтернету обмежена.

- Метод навігації. Вибір відповідного методу навігації є важливим, оскільки він впливає на взаємодію з користувачем і пов'язує два будівельних блоки мобільного рішення: сторону клієнта та сторону сервера. Не завжди є гарною ідеєю вибирати щось абсолютно незвичайне, коли йдеться про навігацію, оскільки користувачі можуть просто загубитися.

- Інтерфейс користувача. Для кінцевих користувачів інтерфейс користувача є

ключовим компонентом рівня презентації та мобільної архітектури загалом, оскільки він дозволяє їм взаємодіяти з продуктом. Ось чому дизайн інтерфейсу користувача має бути зрозумілим і узгодженим у всій програмі. Навіть якщо рішення має складну функціональність, не повинно бути зайвого шуму. Розробники додатків мають поставити себе на місце користувачів і подумати, як зробити продукт привабливим і простим у використанні.

Залежно від критеріїв шаблони архітектури мобільного додатку можна розділити на кілька типів. Опишемо найбільш фундаментальні з них: монолітну архітектуру та мікросервісну архітектуру.

Монолітна архітектура вважається більш традиційним підходом до розробки мобільних додатків. Однак сьогодні він використовується рідко, оскільки монолітна конструкція має масу серйозних недоліків. Зокрема, ця концепція вимагає від програмістів створення мобільного рішення як єдиного цілого. Це означає, що всі функції керуються в одному місці. У результаті, якщо щось піде не так в одному компоненті, це вплине на роботу всієї програми. Крім того, продукти, які базуються на монолітній архітектурі, набагато важче підтримувати та розширювати.

Архітектура мікросервісу — це сучасний спосіб створення додатків для мобільних пристроїв. Відповідно до цього підходу, кожна функція продукту представляє мікросервіс. Усі мікросервіси в додатку є автономними та працюють незалежно. Однак різні компоненти продукту можуть спілкуватися один з одним за допомогою API. Це дозволяє програмістам змінювати будь-який елемент програми, не впливаючи на інші функції. Завдяки цьому мобільні програми, які використовують шаблон архітектури мікросервісу, є більш гнучкими. Розробники можуть легко оновити будь-яку свою частину, і масштабованість також не є проблемою. Нижче наведено порівняльну діаграму монолітної та мікросервісної архітектури (Рис. 3.1.):

Також на сьогодні виділяють наступні основні принципи розробки високоякісної архітектури програми:

- Зрозумілість. Архітектура програми має відображати специфіку цільової

сфери діяльності (наприклад, служби таксі, індустрія доставлення їжі, електронне навчання).

- Масштабованість. Як уже згадувалося, хороша архітектура мобільного додатка дозволяє додавати нові функції та легко розширювати продукт.

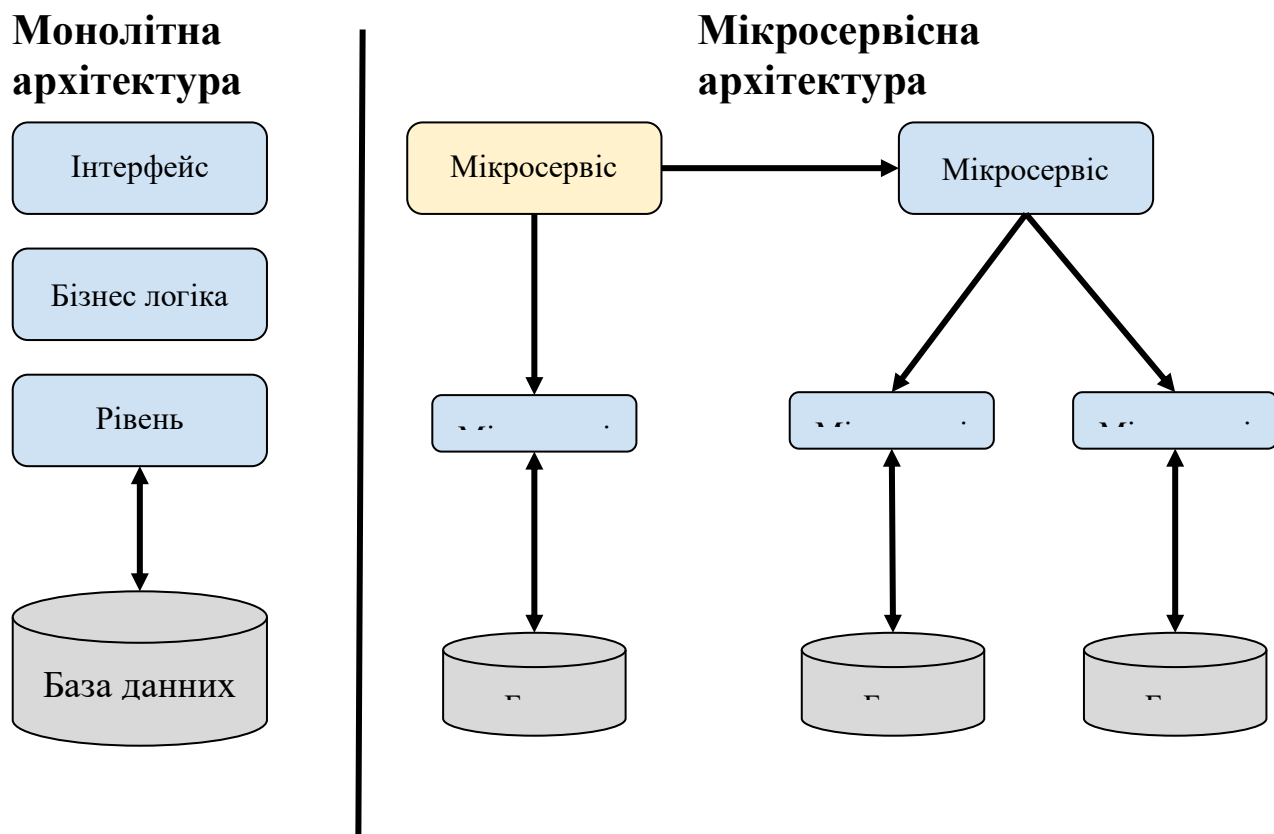


Рис. 3.1. Порівняльна діаграма монолітної та мікросервісної архітектури

- **Читабельність.** Код повинен легко читатися іншим розробникам — навіть тим, хто не брав участі в проєкті.
- **Можливість проведення тестування.** Якщо архітектура мобільного додатку побудована правильно, тестування продукту займе менше часу та ресурсів, а ефективність роботи з контролю якості залишиться на високому рівні.
- **Ефективність.** Мобільний додаток, який базується на надійній архітектурі, забезпечує плавну роботу та може впоратися з великими навантаженнями.
- **Виправлення недоліків.** Добре розроблена архітектура мобільного додатка гарантує, що продукт буде зручним для обслуговування, а витрати, необхідні для його підтримки, не будуть надто високими.

- Оптимізація. Якщо ви створите хорошу архітектуру для проєкту мобільної розробки, це заощадить час і гроші, необхідні для втілення продукту в життя.
- Безпека. Архітектура мобільного додатку повинна передбачати шифрування даних і створювати безпечне середовище для передачі даних.

3.2. Вибір технологій для реалізації програмного застосунку

3.2.1. Опис загальної архітектури програмного застосунку

Нижче наведено загальну рівневу архітектуру запропоновану для розробки нашого проєкту (рис. 3.2).

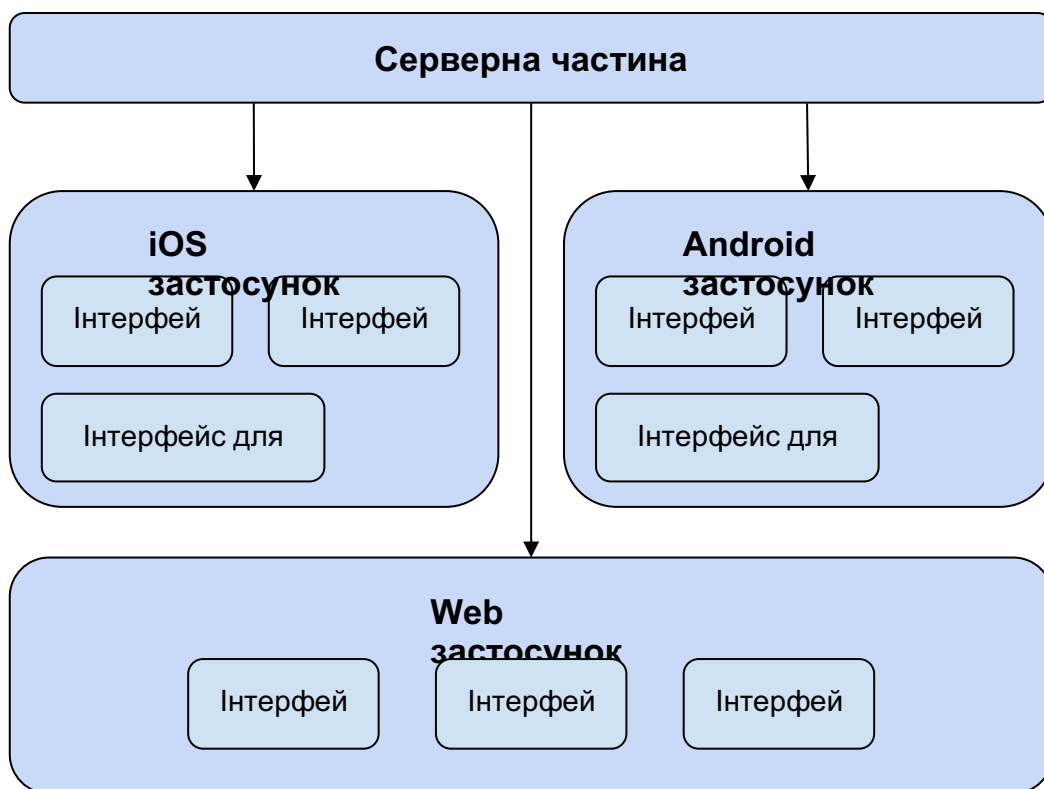


Рис. 3.2. Загальна рівнева архітектура проєкту

Варто відзначити, що дана архітектура є досить верхньорівнева і складена в такому вигляді для загального розуміння всіх учасників процесу потенційних складових всього продукту. При цьому це не означає, що на етапі першої версії

продукт обов'язково має включати всі наведені вище складові. Проте він обов'язково має враховувати, що в майбутньому саме така структура є бажаною або ж планується на даному етапі. Це робиться для того, щоб на ранніх етапах одразу закладати необхідні аспекти для майбутніх інтеграцій функціональних складових системи.

Також важливо розуміти як верхньорівнево архітектурні складові системи будуть взаємодіяти один з одним та обмінюватись даними. Подібну логіку зазвичай треба закладати на початковому етапі планування дизайну архітектури, адже від цього буде залежати багато подальших факторів, таких яких вибір найбільш доцільних технологій, бібліотек та фреймворків для реалізації проєкту, а також від цього можливо доведеться відштовхуватись і при плануванні функціональних вимог до продукту.

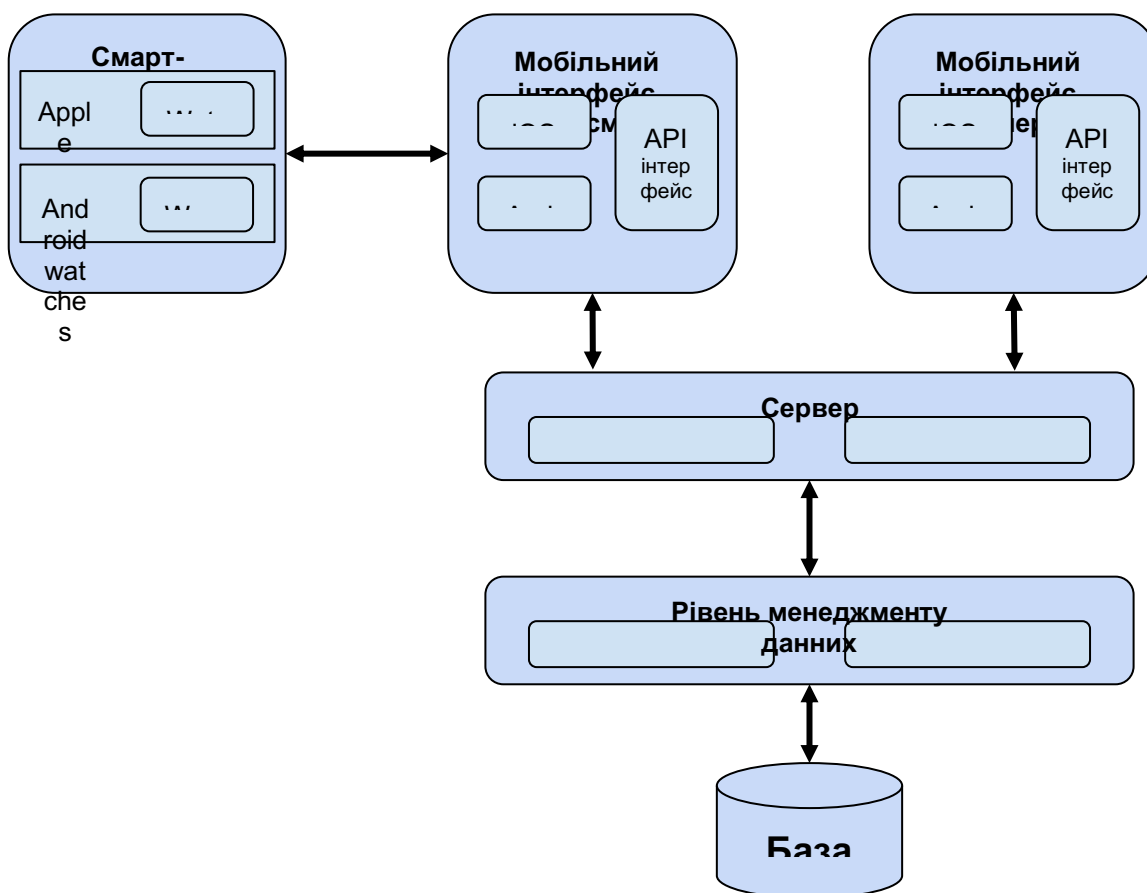


Рис. 3.3. Архітектура взаємодії компонентів системи

Відповідно до алгоритму (рис. 3.3.), наведеного вище, нами запропоновано використовувати наступні рішення для реалізації взаємодії смарт-годинників з іншими складовими продукту:

-

W

atchKit framework від Apple для Apple Watch девайсів

-

W

ear OS Jetpack від Google для Android девайсів.

Вибір даних технологій обґрунтовується наступними факторами:

-

В

они наразі є стандартами в індустрії розробки мобільних застосунків з використанням смарт-годинників;

-

К

омпанії, що розробили та підтримують їх є лідерами на своїх ринках, тому порівняно з іншими альтернативами є найбільш надійними опціями;

-

Кі

лькість спеціалістів, що мають достатній рівень компетенції та досвіду в роботі з подібними технологіями вищий, ніж в будь-яких інших аналогічних варіантах, тому це мінімізує ризики як при пошуку спеціалістів для реалізації проєкту, так і для подальшої його підтримки у майбутньому;

-

Д

ані технології мають гарно описану та доступну документацію, що також спрощує процес реалізації та мінімізує ризики.

Також, якщо звернути увагу на представлену вище діаграму, то можна помітити, що ми розділили на два окремі архітектурні компоненти інтерфейс для спортсмена та інтерфейс для тренера. Таке рішення вважаємо оптимальним з огляду на те, що кожна з цих ролей має окремі унікальні потреби та умови використання застосунку. До того ж, маючи два окремі компоненти, їх буде легше підтримувати у майбутньому, робити оновлення та додавати додатковий функціонал.

Вся бізнес-логіка у нашому випадку виведена на сторону сервера. По-перше, це є загальною практикою та дозволяє робити інтерфейси клієнта легшими, що особливо може бути важливим при роботі з менш потужними мобільними телефонами (зокрема це може бути доречно для Android смартфонів,

а також для старих моделей iPhone). По-друге, це спрощує синхронізацію даних між інтерфейсом тренера та інтерфейсом спортсмена, особливо враховуючи потенційну можливість необхідності робити різні маніпуляції з цими даними, які можуть бути унікальними для кожної з ролей. По-третє, планується використовувати певні алгоритмічні розрахунки, що більш доцільно, зі сторони продуктивності, виконувати саме на стороні сервера.

Проте подібний підхід може мати й ряд ризиків, зокрема, буде необхідно правильно налаштувати передачу даних, враховуючи можливі перебої в роботі мережі, можливі проблеми у доступі між смарт-годинником та смартфоном і т.п. Дані нюанси є доволі серйозними й неправильне розв'язання подібних проблем може призвести до серйозних наслідків при роботі з продуктом (отримання неправильних даних або проблеми з доступом в цілому до продукту). Але в цей самий час, ці проблеми є відомими й рішення давно існують на рівні загальноприйнятих в індустрії правил та стандартів.

Додатково в окремий архітектурний компонент виведено рівень менеджменту даних. Причиною для цього є особливість того, як саме дані будуть збиратися в нашому продукту - за допомогою двох девайсів (смарт-годинник та смартфон).

3.2.2. Вибір засобів для розробки серверної частини

Маючи вже верхньорівневу архітектуру проєкту ми можемо спуститися на рівень нижче до вибору власне технологій, які будуть використовуватись в процесі розробки. Це важливий етап, адже він напряду впливає на команду, яка потім буде працювати над проєктом (тому варто враховувати стан ринку спеціалістів, їх доступність та середній рівень знань і досвіду).

Для реалізації серверної частини основною мовою програмування нами було обрано Python. Таке рішення було прийнято виходячи з наступних факторів:

ана мова програмування є легкою у використанні та читанні. Python схожий на англійську мову, яку ми використовуємо в повсякденному житті. Простота синтаксису дозволяє мати справу зі складними системами та гарантувати, що всі елементи мають чіткий зв'язок один з одним. Представлення даних у форматі, який можна легко зрозуміти, досягається за допомогою різних графіків і діаграм. Вони є ефективним способом візуального представлення та розуміння даних.

-

М

ожливість асинхронного кодування. Щоб написати та підтримувати асинхронний код за допомогою Python, не потрібно багато зусиль, оскільки немає взаємоблокувань, суперечок у дослідженнях або будь-яких інших заплутаних проблем. Кожен блок такого коду працює окремо, що дозволяє швидко вирішувати різні ситуації та проблеми.

-

М

енш обмежений підхід до програмування. У порівнянні з іншими мовами програмування, такими як Java, Python має менш обмежений підхід до програмування. Він має кілька парадигм і може підтримувати безліч стилів програмування, включаючи процедурний, об'єктно-орієнтований і функціональний. Це робить Python чудовою мовою для нашого проєкту, оскільки може знадобитися змінити підхід у будь-який момент, і це не стане проблемою. На додачу, бібліотеки Python містять величезну кількість попередньо написаного коду. Отже, розробникам не потрібно витрачати час на створення основних елементів.

-

Інтеграція існуючих додатків та елементів з інших мов програмування. Python також є популярним вибором для інтеграції з елементами, модулями чи бібліотеками, як написані іншими мовами, такими як Java, PHP і .NET. Виклики Python з і до коду Java, C++ або C напряду дозволяють значно контролювати процес і реалізувати найпоширеніші протоколи та формати даних. Крім того, його можна застосовувати для збирання нових і старих фрагментів інфраструктури, що є типовим випадком для складних мобільних додатків.

є одна хороша риса Python полягає в тому, що вона має багато фреймворків, які спрощують процес розробки. Ці фреймворки своєю чергою загалом мають доволі активні ком'юніті, детальну документацію та постійно розвиваються.

Також важливим моментом є те, що загальною практикою сьогодні є саме використання фреймворків в розробці, а не чистої мови програмування. Це пришвидшує процес реалізації продукту, а також допомагає знизити ризики. Але при цьому використання фреймворків вносить певні обмеження, адже доводиться дотримуватись певних правил, встановлених конкретним фреймворком при написанні коду. Зважаючи на це, для реалізації серверної частини нами було обрано фреймворк, який будується на основі Python мови програмування - Django.

Django — це платформа з відкритим вихідним кодом для серверних веб-додатків. Його головні цілі - простота, гнучкість, надійність і масштабованість.

Django має власну систему іменування для всіх функцій і компонентів (наприклад, HTTP-відповіді називаються «переглядами»). Він також має панель адміністратора, з якою легше працювати, ніж у Lavarel або Yii, та інші технічні функції Django, зокрема:

простий синтаксис - це та перевага, яка фактично наслідується від самого Python та пришвидшує процес роботи. Також це впливає на подальшу підтримку продукту, адже простий синтаксис допомагає краще сприймати код, особливо, якщо він написаний іншим розробником та може мати певні особливості;

ласний веб-сервер, що також спрощує процес, адже не потрібно додатково використовувати сторонні ресурси для того, аби підняти власний сервер в процесі реалізації;

ожливість використання архітектури ядра MVC (Model View Controller) - це пришвидшує розробку та допомагає в цілому зробити процес більш

структурованим та стандартизованим;

-

В

становлені внутрішні бібліотеки HTTP, що, враховуючи наявність веб інтерфейсів у архітектурі проєкту, також є бажаним;

-

Д

оступ до платформи модульного тестування Python, яку ми плануємо використовувати для покриття ключових компонентів продукту автоматизованими тестами для того, щоб бути впевненими у якості його роботи.

На додачу до поданих вище технічних особливостей фреймворку Django, ми також можемо говорити про наступні сильні сторони, які вплинули на наш вибір, порівнюючи його як з іншими фреймворками від Python (наприклад, Flask), так і з іншими альтернативними мовами програмування (Ruby, Rust, Java, Node.js і т.п.):

-

В

икористання принципу швидкої розробки (RAPID), який дозволяє розробникам робити більше однієї ітерації за раз, не починаючи з нуля, тим самим зменшуючи час на імплементацію функціонала

-

М

ожливість успішно інтегрувати філософію DRY (“не повторюватись”) - розробники можуть повторно використовувати існуючий код та зосередитись на імплементації саме унікальних його частин

-

Б

езпека також є пріоритетом для Django. Він має одну з найкращих готових систем безпеки, яка допомагає розробникам уникати поширених проблем безпеки, зокрема: клікджекінг, міжсайтове скриптування, SQL-ін'єкція.

-

Dj

ango, не дивлячись на те, що це доволі молодий фреймворк, вже можна вважати перевірений часом. Він має велику спільноту підтримки, до якої можна отримати доступ через численні форуми, канали та спеціальні веб-сайти. Легко знайти

допомогу, якщо в коді є проблемна функція, і знайти розробників, якщо доведеться розширювати команду, чи під час подальшої підтримки продукту.

-

Dj

ango почав із чудової документації, найкращої з усіх інших фреймворків з відкритим кодом. І він все ще підтримується на високому рівні, оновлюється разом із новими функціями та виправленнями, тож за потреби можна буде легко адаптуватися до змін. Розробнику не складе труднощів розібратися, як використовувати Django в конкретному випадку. Також можна бути впевненим, що будь-які проблеми з фреймворком будуть вирішені, як тільки вони виникнуть. Програмне забезпечення постійно оновлюється та випускаються нові пакети, щоб зробити роботу з Django зручнішою, ніж вона є.

Додатковим важливим фактором було визнання і використання фреймворку успішними компаніями індустрії. Ось деякі з популярних програм, які користуються швидкістю, гнучкістю та простотою використання Django:

-

In

stagram: популярна соціальна мережа, яка обробляє велику кількість медіа-даних і взаємодії користувачів. Django забезпечує функціональність, завдяки якій додаток працює безперебійно, додає нові функції та миттєво розв'язує проблеми.

-

S

potify: велика медіа-бібліотека з величезним обсягом інформації, яка дозволяє користувачам слухати музику безкоштовно або за підпискою без реклами. З технічної сторони Spotify також використовує машинне навчання, де Python є одним із найкращих варіантів. Творці вирішили поєднати його з фреймворком Django.

-

T

he Washington Post: Не дивно, що The Washington Post використовувала Django для обробки свого великого трафіку, оскільки сам фреймворк був створений для роботи онлайн-газети. Кілька інших періодичних онлайн-видань також використовують Django.

-

D

Dropbox: хмарна технологія для зберігання файлів вимагає високопродуктивної функціональності. Django надає Dropbox інструменти, необхідні для спільного використання та синхронізації, а також масштабованості.

-

N

ASA: вони використовують Django для деяких функціональних елементів, які потребують найбільшої надійності. Їх веб-сайт не такий популярний, як Instagram, наприклад, але він обробляє зображення високої роздільної здатності та в середньому має 2 мільйони переглядів на місяць.

-

3.2.3. Вибір засобів для розробки клієнтської частини

Якщо ми говоримо про клієнтську частину (клієнтський інтерфейс), то в першу чергу варто зазначити, що ми маємо не один такий інтерфейс:

-

iOS

S мобільний застосунок (інтерфейс тренера та спортсмена);

-

A

ndroid мобільний застосунок (інтерфейс тренера та спортсмена);

-

За

стосунок для смарт-годинника Apple Watch (інтерфейс спортсмена);

-

За

стосунок для смарт-годинника на базі Android (інтерфейс спортсмена);

-

B

еб-додаток (інтерфейс тренера, спортсмена та адміністратора).

Проте якщо дивитись зі сторони саме технологій, то нам потрібно сформулювати список технологій, які ми будемо використовувати для розробки:

-

M

обільного застосунку (для iOS та Android);

-

За

стосунку для смарт-годиннику (Apple Watch та на базі Android);

-

В

еб-застосунку.

У попередніх розділах ми зазначали, що є декілька підходів до розробки мобільних застосунків і обирати який саме використовувати треба, виходячи з багатьох факторів, основні з яких це функціональні обмеження, обмеження по бюджету, наявність ресурсів та додаткові нефункціональні вимоги. Враховуючи все це, ми обрали розробку мобільних застосунків саме нативними методами. Такий вибір в першу чергу було зроблено, тому що нам необхідно буде активно використовувати апаратну складову смартфонів, а також доволі значний функціонал продукту базується на отриманні даних зі смарт-годинників. Все це значно краще реалізується завдяки нативним методам, тому прийнято рішення не намагатися економити на основному функціоналі продукту. Адже проблеми з якістю саме в цьому функціональні можуть дуже негативно вплинути на сприйняття продукту юзером, а, отже, і на успіх продукту в цілому.

Виходячи з вище перерахованого, для розробки мобільних застосунків (інтерфейсів спортсмена та тренера) було обрано наступні технології:

-

Д

ля застосунку під операційну систему iOS: Swift;

-

Д

ля застосунку під операційну систему Android: Kotlin.

Додатково планується використовувати ряд сторонніх модулів та бібліотек, які направлені на спрощення та пришвидшення розробки. Нижче наведений список потенційно корисних ресурсів, які ми наразі бачимо корисними для розробки клієнтського інтерфейсу iOS застосунку:

-

С

oreMotio: надає доступ до акселерометра, гіроскопа та інших датчиків руху на пристрої, що має вирішальне значення для відстеження даних про діяльність.

-

Н

ealthKit: інтегрується з додатком «Здоров'я» на iOS для читання та запису даних

про здоров'я та фізичну форму, як-от частота серцевих скорочень, відстань і спалені калорії.

-

M

arKit: дозволяє відображати маршрути та місця тренувань на картах.

-

A

VFoundation: для запису та аналізу відео для оцінки ефективності.

-

R

ealm: гнучка та потужна база даних із відкритим вихідним кодом для ефективного зберігання та запиту даних про діяльність.

-

S

wiftyJSON: спрощує синтаксичний аналіз і роботу з даними JSON, які часто використовуються для зв'язку через API.

-

C

ore ML і Create ML: для створення спеціальних моделей машинного навчання для аналізу даних про ефективність і пропонування персоналізованих планів навчання.

-

Cl

oudKit: забезпечує безпечну платформу для зберігання та обміну програмами тренувань і даними тренувань між спортсменами та тренерами.

-

P

ushKit: дозволяє надсилати тренерам оновлення прогресу тренування та сповіщення в реальному часі.

Так само ми склали подібний список потенційно корисних ресурсів для розробки клієнтського інтерфейсу Android застосунку. Слід зазначити, що дані платформи є незалежними й тому більшість приведених ресурсів є унікальними для кожної з платформ. Проте майже у кожного з ресурсів (бібліотека, фреймворк, модель і т.п.) існує якщо не аналогічна, то дуже подібна альтернатива на протилежній платформі. Нижче приведено список для Android:

- G
 oogle Fit API: надає доступ до датчиків і даних про стан здоров'я, подібних до HealthKit, що має вирішальне значення для відстеження даних про активність.

- W
 ear OS API: дозволяє створювати супутню програму годинника для відображення даних тренування в реальному часі, керування музикою та надсилання сповіщень.

- M
 aps SDK: дозволяє відображати маршрути тренувань і місця на картах, подібно до MapKit.

- M
 ediaRecorder: для запису та аналізу відео для отримання відгуків про продуктивність.

- G
 son або Moshi: бібліотеки для спрощення обробки даних і серіалізації в JSON, подібні до SwiftyJSON.

- M
 PAndroidChart: універсальна бібліотека для створення різноманітних діаграм і графіків для візуалізації даних тренувань.

- A
 pache Spark: для розширеного аналізу даних і завдань машинного навчання, подібних до Core ML.

- T
 ensorFlow Lite: дозволяє запускати попередньо підготовлені моделі машинного навчання для персоналізованих планів навчання, подібно до Create ML.

- Б
 аза даних у реальному часі Firebase: хмарна платформа для зберігання та обміну програмами тренувань і даними тренувань, подібна до CloudKit.

- F
 CM (Firebase Cloud Messaging): забезпечує платформу для надсилання тренерам

оновлень і повідомлень про прогрес тренування в реальному часі, подібно до PushKit.

Що ж до застосунків для смарт-годинників, то загалом, в процесі їх розробки буде використовуватись майже аналогічний технічний стек та допоміжні ресурси, що й наведено вище для мобільних застосунків для операційних систем iOS та Android. Майже єдині додаткові технології, що будуть застосовуватись (якщо ми говоримо про основні технології):

-

W

atchKit: фреймворк Apple для створення додатків watchOS, надання доступу до датчиків, елементів інтерфейсу користувача та зв'язку з підключеним iPhone. Фреймворк створений на основі та використовує Swift як мову програмування;

-

W

ear OS API: фреймворк Google для створення додатків Wear OS, надання доступу до датчиків, елементів інтерфейсу користувача та зв'язку зі сполученим пристроєм Android. Фреймворк створений на основі та використовує Kotlin як мову програмування.

Залишилось розглянути технології які ми плануємо використовувати для розробки клієнтської частини веб-застосунків. Враховуючи, що згідно з обраною архітектурною структурою, веб-інтерфейси будуть виведені окремо від мобільних, а основна бізнес-логіка зосереджена на серверній частині, ми маємо ситуацію, коли наші обмеження до вибору технології для розробки є мінімальними. В такому разі, обрати пропонується JavaScript як основну мову програмування, а React, як обраний фреймворк для роботи. Говорячи про причини такого вибору, окремо хочу розглянути більш бізнесові:

-

В

елика кількість доступних спеціалістів на ринку. В такому разі буде можливість обрати спеціалістів необхідної кваліфікації при мінімальних затратах ресурсів на це;

-

В

елика та активна професійна спільнота, що дозволить значно легше розв'язувати потенційні проблеми в ході реалізації;

-

Д

аний фреймворк розроблений компанією Facebook, що говорить про його високу стабільність та надійність (компанія продовжить підтримувати продукт і надалі).

До того ж є ряд технічних причин, які роблять вибір React фреймворку доцільним для використання в нашому продукті:

-

R

east поставляється з JSX, необов'язковим розширенням синтаксису, яке дає змогу писати власні компоненти. Ці компоненти в основному приймають HTML-цитування, а також роблять відтворення всіх субкомпонентів простим та таким, що не вимагає багато зусиль від розробників;

-

В

икористання React забезпечує гарні показники веб-застосунків у видачі Google (SEO). Чим нижчий час завантаження сторінки та швидша швидкість візуалізації, тим вищий рейтинг програми в Google. Завдяки швидкому рендерингу React, порівняно з іншими фреймворками, значно скорочує час завантаження сторінки, що допомагає компаніям забезпечити високе місце на сторінці результатів пошукової системи Google.

-

У

нікальні хуки React - це функція, яка дозволяє розробникам додавати стани та інші функції до функціональних компонентів.

-

Ін

струменти розробника React разом із інструментами розробки Redux пропонують надзвичайно зручні можливості, хоча їх легко встановити та використовувати як звичайні розширення Chrome. Зокрема, з їх допомогою можна ефективно перевіряти компоненти ієрархії на основі React (включно з пов'язаними атрибутами та станами), перевірити дії надсилення та переглядати зміни стану негайно в розширенні.

ртуалізуючи та зберігаючи DOM (Об'єктна модель документа) у пам'яті, React надає неймовірно швидко здатність рендерингу з усіма змінами перегляду, які легко відображаються у віртуальному DOM. Спеціалізований алгоритм відмінностей зіставляє колишні та існуючі віртуальні стани DOM, обчислюючи найефективніший спосіб застосування нових змін, не вимагаючи занадто багато оновлень. Потім вводиться мінімальна кількість оновлень для досягнення найшвидшого часу читання/запису, що призводить до загального підвищення продуктивності.

Саме тому серед інших потенційних варіантів, які фактично також можна було б використовувати для розробки інтерфейсу клієнта веб додатку, вибір зроблено саме на користь React.

3.2.4. Вибір бази даних та патернів її проєктування

Те, як відбувається керує даними у програмі, відіграє вирішальну роль у забезпеченні позитивної взаємодії з користувачем. Зрештою, не має значення, наскільки добре розроблено інтерфейс програми та наскільки чистий код, якщо програма не здатна швидко отримувати, обробляти та доставляти дані. Крім того, всі ці дані повинні бути захищені, щоб злоумисники не могли отримати до них доступ. База даних — це місце, де зберігаються й упорядковуються всі дані, які збираються за допомогою продукту, а система керування базами даних (СУБД) — це програмне забезпечення для зручного керування цією базою даних.

На ринку представлено понад 300 баз даних. Коли справа доходить до вибору найкращого рішення, однією з найбільших проблем є вибір між структурою даних SQL (реляційна) і NoSQL (нереляційна). Хоча обидва мають хорошу продуктивність, є ключові відмінності, про які слід пам'ятати.

Реляційна база даних — це набір таблиць, між якими є попередньо

визначені зв'язки. Це найбільш використовуваний тип бази даних. Для підтримки та запитів до реляційної бази даних система керування базою даних використовує мову структурованих запитів (SQL). Реляційні бази даних складаються з рядків, які називаються кортежами, і стовпців, які називаються атрибутами. Кортежі в таблиці мають однакові атрибути.

Реляційна база даних ідеально підходить для зберігання структурованих даних (поштових індексів, номерів кредитних карток, дат, ідентифікаційних номерів). SQL — це зріла технологія, яка:

- Д
обре задокументована;
- М
ожливо похвалитися великою підтримкою;
- Д
обре працює з більшістю сучасних фреймворків і бібліотек.

Найкращими базами даних SQL є PostgreSQL і MySQL. Обидві довели свою стабільність та безпечність. Реляційні бази даних підтримують права доступу, які визначають, кому дозволено читати та редагувати дані. Адміністратор бази даних може надати певні привілеї користувача для доступу, вибору, вставки або видалення даних. Це не дає третім сторонам можливості викрасти інформацію. Використання найкращої системи керування реляційною базою даних (RDBMS) захищає від втрати та пошкодження даних завдяки відповідності властивостям ACID: атомарності, узгодженості, ізоляції та довговічності.

Але реляційні бази даних також мають недоліки:

- ВО
ни неефективно працюють із напівструктурованими або неструктурованими даними, тому вони не підходять для великих навантажень і аналітики;
- К
коли структура даних стає складною, стає важче обмінюватися інформацією з одного великого програмного рішення, керованого даними, до іншого. У великих

системах реляційні бази даних часто розростаються незалежно в окремих підсистемах;

-

Ре

ляційні бази даних працюють лише на одному сервері, а це означає, що якщо ви хочете, щоб ваша СУБД справлялася з більшим обсягом даних, вам потрібно інвестувати у дороге фізичне обладнання;

Ці недоліки змусили розробників шукати альтернативи реляційним базам даних.

В результаті з'явилися бази даних NoSQL, також звані нереляційними або розподіленими базами даних, служать альтернативою реляційним базам даних. Відносно реляційних баз даних, вони мають наступні переваги:

-

В

они можуть зберігати й обробляти неструктуровані дані (дані із соціальних мереж, фотографії, MP3-файли тощо), пропонуючи розробникам більшу гнучкість і масштабованість;

-

Д

ані в нереляційних базах даних можна змінювати на льоту, не впливаючи на наявні дані. Крім того, бази даних NoSQL можна запускати на кількох серверах, тому їх масштабування дешевше та легше, ніж масштабування баз даних SQL.

-

О

скільки бази даних NoSQL не покладаються на один сервер, вони більш відмовостійкі. Це означає, що якщо один компонент виходить з ладу, база даних може продовжувати роботу.

Але бази даних NoSQL є менш зрілими, ніж бази даних SQL, і спільнота NoSQL не так добре визначена. Крім того, бази даних NoSQL часто жертвують відповідністю ACID заради доступності та гнучкості. Проте подібні недоліки значно перекриваються приведеними вище аргументами щодо переваг використання нереляційної бази даних в розробці продукту, тому саме такий вибір нами і пропонується. Системою для управління нереляційною базою даних обрано MongoDB, як одне з найбільш популярних рішень на сьогодні на ринку.

MongoDB — це база даних NoSQL, яка зберігає всі дані в документах BSON (Binary JSON). Завдяки цьому дані можна легко передавати між веб-додатками та серверами в зручному для читання форматі.

MongoDB має вбудовану реплікацію, що забезпечує високу масштабованість і доступність. Автоматичне шардинг означає, що ви можете легко поширювати дані на сервери, підключені до вашої програми. Загалом MongoDB є найкращою веб-базою даних для роботи з масивними неструктурованими наборами даних. Вона може слугувати основою більшості систем великих даних не лише як сховище оперативних даних у реальному часі, але й в автономному режимі.

3.3. Реалізація програмного застосунку

3.3.1. Опис функціональної частини застосунку

Отже, наразі ми дійшли до однієї з ключових частин планування створення нашого сервісу - опис функціональної складової системи. Слід одразу зазначити, що враховуючи той факт, що обраною моделлю до організації роботи над застосунком є доволі гнучка модель яка базується на Scrum методології, то і запланований функціонал застосунку може (і має) змінюватись в процесі виконання проєкту, орієнтуючись на відгуки користувачів та інших стейкхолдерів при тестуванні проміжних результатів (інкрементів продукту). Також важливим моментом є те, що планувати ми будемо лише функціонал системи для запуску першої версії продукту (MVP - мінімально життєздатного продукту), а функціонал для наступних версій на разі має залишитись лише на верхньорівневому рівні та в статусі більш “потенційного” функціоналу. Прийняття рішення про те, додавати його до продукту, чи, можливо, він потребуватиме певних змін і переосмислення, має бути відкладено до подальших фаз розвитку продукту і буде залежати від результатів запуску першої версії

продукту та відгуків користувачів.

Також, враховуючи наявність різних інтерфейсів, опис функціональної частини застосунку пропонуємо розділити відповідно до наступних ролей:

- Користувач-спортсмен;
- Користувач-тренер;
- Адміністратор.

Опис функціонала для ролі користувач - спортсмен подано у відповідній таблиці додатку Г. Як видно з таблиці додатку Г, сервіс навіть у першій своїй версії повинен мати доволі розлогий функціонал, аби його було доцільно застосовувати спортсмену. При всьому цьому, даний функціонал не є спеціалізованим і покриває лише загальні потреби спортсмена. Саме тому великий акцент при випуску першої версії має бути на комунікацію з користувачами, аби зібрати список їх запитів та потенційних проблем, щоб сформуванати список максимально бажаного та корисного функціоналу для наступних версій сервісу.

Опис функціонала для ролі користувач - тренери зроблено у відповідній таблиці додатку Д.

Також слід додати, що в першій версії пропонується на разі додавання лише одного виду спорту - бейсболу. Вибір бейсболу обумовлений наступними критеріями:

- Д
для даного виду спорту ринок сервісів, що пропонують подібний функціонал не такий широкий, враховуючи, що його глобальна популярність наразі обмежена кількома країнами (США, Канада, Японія - основні з них);

- П
ри цьому потенційний ринок користувачів доволі великий - понад 500 тисяч активних спортсменів в університетах, коледжах та старших школах.

Надалі список доступних видів спорту планується розширювати.

Ще одним важливим елементом для адекватної роботи системи та можливості надавати необхідну підтримку користувачам є необхідність

створення ролі адміністратора з відповідним функціоналом. Таблицю “Опис функціонала для ролі “адміністратор” можна побачити в додатку Е.

Нагадаємо, що в додатках Г, Д, Е наведені списки функціонала, який планується включити в першу версію продукту. Тому варто додатково зазначити список того, що не входить до першої версії, проте майже точно буде включено у наступних найближчих оновленнях:

- В
еб-версія сервісу (для тренерів та для спортсменів);

- За
стосунок для смарт-годинників під операційну систему iOS та під операційну систему Android (для спортсменів);

- Д
одаткові види спорту - американський футбол, баскетбол, плавання, легка атлетика плануються як наступні види спорту для додавання у продукт в першу чергу;

- А
лгоритми машинного навчання (ML) та функціонал штучного інтелекту (AI) для оцінки наявних даних і формування рекомендацій для максимізації спортивних результатів;

- Р
озділ харчування;

- Р
озділ контролю якості сну.

Отже, нами було запропоновано список функціонала для першої версії, який, на нашу думку, буде достатньо для виведення продукту на ринок та для того, щоб почати залучати перших користувачів і отримувати критично важливий зворотній зв'язок від них. Надалі буде формуватись і список функціонала для наступних версій, який допоможе не тільки втримати вже існуючих користувачів продукту, а й масштабувати його, залучити нові аудиторії.

3.3.2. Опис організації процесу роботи над застосунком

У першому розділі даної роботи нами були розглянуті декілька підходів до організації процесу роботи на подібних до нашого проєктах. Враховуючи наведені там переваги та недоліки методів, та з огляду на наявні особливості нашого продукту, ми пропонуємо використовувати саме SLeSS - гнучкий підхід, який інтегрує Scrum і Lean Six Sigma (LSS) як основу для організації власного процесу. Основними перевагами даного підходу в нашому конкретному випадку є:

- Додаткова гнучкість та адаптивність, що характерна як для Scrum, так і для Lean Six Sigma;

- Scrum на сьогодні фактично являється стандартом індустрії, тому більшість спеціалістів мають досвід роботи в такому середовищі, що зменшить час на навчання та адаптацію команди;

- Використання коротких ітерацій (характерних для Scrum) дозволить швидко отримувати робочий інкремент продукту та проводити його періодичні тестування і валідацію з потенційними користувачами.

Починаючи описувати процес для організації роботи над продуктом, першочергово пропонуємо визначити ролі в команді для роботи над проєктом:

- Інженери - це всі спеціалісти, які напряду відповідають за створення продукту - розробники серверної та клієнтської частини, інженери з якості (тестувальники), дизайнери;

- Власник продукту (Product owner) - пояснює команді бачення та мету продукту.

Визначає пріоритети на основі цілі продукту, щоб забезпечити цінність для всіх зацікавлених сторін, включаючи тих, хто входить до організації, і всіх користувачів як всередині, так і ззовні. Власник продукту визначає, вимірює і максимізує цінність протягом усього життєвого циклу продукту.

-

Sc

rum-майстер (Scrum master) - несе відповідальність за організацію процесу Scrum. Він робить це, допомагаючи кожному зрозуміти теорію та практику Scrum, як у команді, так і в організації, обслуговуючи команду Scrum, а також більшу організацію.

Враховуючи, що Scrum є основою до організації процесу відповідно до обраного нами підходу, потрібно визначити стандартну тривалість спринту/ітерації. В майбутньому ця тривалість може бути змінена командою у випадку, якщо це буде необхідно. Ми рекомендуємо обрати стандартну тривалість, яка використовується на більшості подібних проєктах - 2 тижні.

Якщо дивитися зі сторони зустрічей та регулярних подій, нами запропоновано включити до процесу наступні:

-

Щ

оденна Scrum зустріч (Daily Scrum) - це 15-хвилинна подія для розробників команди Scrum. Щоб зменшити складність, вона проводиться в один і той же час і в одному місці кожного робочого дня ітерації/спринту. Якщо власник продукту або Scrum-майстер активно працюють над задачами, то вони також беруть участь як розробники. Розробники можуть вибрати будь-яку структуру та методи, які вони хочуть, якщо їх Scrum зустріч зосереджується на досягненні цілі спринту та створює дієвий план на наступний день роботи. Це створює концентрацію та покращує самоконтроль. Щоденні зустрічі покращують комунікацію, виявляють перешкоди, сприяють швидкому прийняттю рішень і, отже, мінімізують потребу в інших зустрічах.

-

П

ланування спринту (Sprint planning) - це подія в Scrum, яка починає спринт. Метою планування спринту є визначення того, що може бути досягнуто під час

спринту та як ця робота буде досягнута. Планування спринту здійснюється у співпраці з усією Scrum-командою.

-

О

гляд спринту (Sprint review) - це фактично демонстрація роботи всієї команди протягом останньої ітерації. Це зустріч, на якій зацікавлені особи можуть поставити запитання, команда демонструє їм новий інкремент продукту або конкретний новий функціонал та отримує відгук про нього.

-

Рет

троспектива спринту (Sprint retrospective) - це тип зустрічі в рамках Agile, під час якої команди обмірковують, що пройшло добре, а що можна було б покращити для наступного спринту. Ретроспективи є важливими для постійного вдосконалення процесу спринту та забезпечення врахування ключових знань для наступного разу. Насправді, це один з основних елементів для реалізації принципу Lean Six Sigma, який є важливою складовою обраного нами підходу для організації процесу роботи.

-

У

точнення невиконаних завдань (Sprint refinement) - це зустріч, де власник продукту та деякі або всі інші члени команди переглядають елементи в списку не виконаної роботи (Backlog), щоб переконатися, що в ньому задачі мають відповідні елементи, пріоритет і що задачі у верхній його частині готові до того, щоб їх брати в роботу. Ми пропонуємо зробити дану зустріч регулярною, враховуючи, що ми плануємо отримувати регулярний зворотний зв'язок від потенційних користувачів продукту, що цілком може змінювати пріоритети та в цілому функціональні вимоги й задачі продукту.

В таблиці 3.1., наведеній нижче підсумовано перелік стандартних зустрічей в розробленому процесі, а також запропоновані їх характеристики:

Таблиця 3.1.

Перелік стандартних зустрічей в розробленому процесі та їх характеристики

Зустріч	Тривалість	Регулярність	Учасники
Scrum зустріч	> 15 хвилин	Кожен робочий день	Інженери - обов'язково, Власник продукту та Scrum-майстер - за потреби
Планування спринту	> 2 годин	Старт кожного спринту	Інженери, Власник продукту, Scrum-майстер - обов'язково
Огляд спринту	> 1 години	Кінець кожного спринту	Інженери, Власник продукту, Scrum-майстер - обов'язково
Ретроспектива спринту	> 1 години	Кінець кожного спринту + при потребі	Інженери, Scrum-майстер - обов'язково
Уточнення невиконаних завдань	1-2 години	1 раз в спринт + за потреби	Інженери - обов'язково, Власник продукту та Scrum-майстер - за потреби

Також важливим елементом організації процесу є різноманітні сервіси для спрощення моніторингу прогресу та синхронізації діяльності команди в цілому. Нами запропоновано наступні сервіси для цього:

- **GitLab** - сервіс контролю версій, зберігання кодової бази, автоматизації процесів деплою. Даний сервіс допомагає команді в цілому ефективніше працювати в процесі реалізації застосунку, ефективніше оновлювати його, впроваджувати найкращі інженерні практики для забезпечення відповідної якості продукту та швидкості його розробки;

- **Figma** - це інструмент для спільного проєктування інтерфейсу користувача на основі браузера, який дозволяє користувачам працювати разом, щоб створювати яскраві та інтерактивні прототипи;

- **Jira** - це сервіс для менеджменту задач, організації процесу роботи над ними, планування проєктних активностей. На сьогодні це один з найбільш популярних сервісів менеджменту задач у світі й, в цілому, є золотим стандартом ІТ індустрії;

-

С

onfluence - це інструмент для спільної роботи, використання та зберігання різного роду інформації. Даний продукт розроблений тією ж компанією, що й Jira (Atlassian), тому вони гарно інтегруються і доповнюють один одного;

-

А

mplitude - це потужна платформа аналітики, яка дає змогу створювати кращі продукти, відстежуючи та розуміючи поведінку користувачів.

Отже, нами було запропоновано загальний опис організації процесу роботи над продуктом для його ефективної реалізації. Проте, важливо прокоментувати, що даний процес є лише правилами, за якими команді рекомендується почати працювати. Але важливою характеристикою обраного нами підходу до роботи є адаптивність та необхідність постійного покращення. Тому, скоріш за все, дані правила будуть видозмінюватись та доповнюватись у процесі, що є не тільки нормальним явищем, а й бажаним.

Маючи вже загальне бачення архітектури нашого продукту, верхньорівневий функціонал та опис процесу організації роботи, ми можемо перейти до опису потенційного складу команди для роботи над продуктом.

Виходячи з технічних вимог, ми можемо виділити наступні компетенції, які мають закриватися командою:

-

Д

изайн інтерфейсів та досвіду користувача (UI/UX design);

-

М

енеджмент продукту;

-

Р

озробка серверної частини

-

Р

озробка клієнтської частини (мобільні застосунки);

-

Р

озробка клієнтської частини (веб-застосунки);

-

К

онтроль якості;

-

М

енеджмент процесів (Scrum-майстер).

Також треба враховувати, що при формуванні команди треба дотримуватись балансу професійного рівня спеціалістів та бюджету, який може бути алокований на оплату команди. Цього можна досягти, якщо формувати команду, залучаючи також менш досвідчених виконавців, які можуть закривати необхідні задачі на достатньому рівні. Адже, логічно, як і в будь-якій іншій сфері, чим вищу кваліфікацію має певний спеціаліст, тим дорожче він буде коштувати.

Для подальшого спрощення розрахунків вартості команди, а також для формування складу команди з урахуванням рівня кваліфікації спеціалістів, ми будемо використовувати прийняті в індустрії рівні кваліфікації:

-

М

олодший рівень (junior level);

-

С

ередній рівень (middle level);

-

С

тарший рівень (senior level).

Слід розуміти, що таке розділення є доволі умовним. Але при цьому це дозволить нам розрахувати приблизний бюджет витрат на команду далі, а також скласти необхідну структуру, за якою команда надалі може бути найнята.

Нижче наведена таблиця 3.2. із запропонованим складом команди для реалізації подібного проєкту, необхідна спеціалізація членів команди, а також бажаний рівень кваліфікації:

Таблиця 3.2.

Якісний склад команди для реалізації проєкту

Напрямок	Спеціалізація	Кваліфікація	Кількість
Менеджмент	Власник продукту	senior	1

	Scrum-майстер	middle	1
Дизайн	Дизайнер продукту	senior	1
	Дизайнер інтерфейсів	junior	1
Мобільна розробка	iOS інженер	senior	1
	iOS інженер	middle	1
	Android інженер	senior	1
	Android інженер	middle	1
Розробка веб інтерфейсу	React інженер	senior	1
	React інженер	middle	1
Розробка серверної частини	Python інженер	senior	1
	Python інженер	middle	1
	Python інженер	junior	1
Контроль якості	Інженер з контролю якості (QA)	senior	1
	Інженер з контролю якості (QA)	middle	1
	Інженер з контролю якості (QA)	junior	1

Слід також враховувати, що подібний склад команди необхідний не на кожному етапі реалізації проєкту і не враховує наразі особливості етапів життєвого циклу продукту. Це максимальний пропонований об'єм команди в період її максимальної продуктивності. Пропонуємо розглянути більш детально складові запропонованої структури.

На рівні менеджменту ми виділяємо двох членів команди - Власника продукту (фокусується на продукти) та Scrum-майстра (фокусується на процесі). На нашу думку, критично необхідним є високий рівень кваліфікації саме

Власника продукту, адже на етапі створення нового продукту його подальший успіх надзвичайно сильно залежить від вміння визначити справжні потреби ключових груп користувачів, для чого треба провести необхідні дослідження, сформулювати гіпотези, які потім перевірити. Це доволі складний процес, який потребує досвіду. А враховуючи вплив результатів роботи Власника продукту на роботу всієї іншої команди (адже саме він займається постановкою задач для команди), ця роль виходить занадто важливою і ризикованою, щоб на ній економити. Scrum-майстра пропонується наймати рівня middle, враховуючи, що команда буде одна і значних взаємодій з додатковими системами на першому етапі не передбачається.

На рівні дизайну нами пропонується розділити активності на 2 напрямки:

- Д
дизайн продукту (product design) - створення загального бачення взаємодії користувачів з майбутнім продуктом, розробка дизайн системи та концепції, задачі більш вищого рівня, а також постановка задач для дизайнера інтерфейсів;

- Д
дизайн інтерфейсів (UI/UX design) - реалізація поставлених задач, шляхів взаємодії користувачів з продуктом, створення дизайнів та матеріалів, з якими далі будуть працювати розробники.

Для першої активності запропоновано наймання досвідченого спеціаліста senior рівня, тому як вимоги до якості дизайну та вражень від взаємодії з продуктом у користувачів на сьогоднішньому ринку доволі високі. А приймаючи на увазі комплексність нашого продукту та відсутність повністю ідентичних рішень на ринку, дана позиція також має ключове значення в процесі реалізації проєкту. Проте на другу активність (дизайн інтерфейсів) на нашу думку цілком логічно було б найняти спеціаліста з незначним досвідом. Мотивація до цього полягає в тому, що при створенні проєкту з нуля існує доволі багато рутинних активностей, які дозволять не настільки досвідченому спеціалісту, під керівництвом досвідченого продуктового дизайнера отримати необхідний досвід та знання про конкретний продукт, що значно збільшить з часом його

ефективність та якість роботи.

На рівні розробки (мобільна, веб-клієнта та серверної частини) нами запропоновано однакову стратегію - мати по кожній з компетенцій одного члена команди senior рівня, який буде відповідати в тому числі за дотримання стандартів та рівня якості реалізації технічних рішень. Плюс в пару до нього спеціаліста middle рівня, тому як подібного рівня навичок має бути цілком достатньо для реалізації більшості завдань на проєкті, особливо під контролем та з допомогою більш досвідченого партнера у вигляді senior інженера такої ж спеціалізації.

Майже аналогічна ситуація і з рівнем контролю якості (QA). Нами пропонується ті самі один інженер рівня senior та один інженер рівня middle. Але при цьому вважаємо за доцільне додати ще одного інженера рівня junior. Це необхідно для того, щоб збільшити пропускну спроможність інженерів контролю якості, при цьому мінімально збільшуючи бюджет на цей напрямок. У випадку, якщо такої кількості інженерів по цьому напрямку все одно не буде вистачати - рекомендується додатково найняти ще одного інженера рівня junior.

Отже, нами запропонована орієнтовна структура команди, необхідна для реалізації подібного проєкту. Слід враховувати, що така її композиція не є єдиною можливою і може дуже сильно варіюватись при зміні контексту, зовнішніх чи внутрішніх обставин. Наприклад, необхідність реалізувати проєкт в найкоротші терміни з можливих призведе до необхідності збільшити розмір команди, додавши спеціалістів за різними напрямками. При цьому бюджетні обмеження можуть спричинити значне зменшення команди або навіть реалізацію проєкту поетапно, не покриваючи одразу всі можливі інтерфейси.

3.4. Оцінка ефективності нового рішення

3.4.1. Розрахунок витрат на розробку та подальшу підтримку мобільного застосунку

Раніше у цьому розділі ми описали верхньорівневий список задач для реалізації першої версії нашого продукту. Також після цього ми запропонували потенційний склад команди, яка може провести подібну реалізацію. Маючи необхідні дані, нам треба лише вартість відповідних спеціалістів, для того, аби скласти орієнтовний план реалізації проєкту, а також потенційний бюджет, який для цього буде потрібен.

Нижче в таблиці 3.3 наведено спеціалістів, яких нам необхідно буде наймати в команду виходячи з тих технічних завдань, що перед нею стоять, разом з витратами на таких спеціалістів в місяць залежно від рівня їх кваліфікації.

Таблиця 3.3.

Витрати на спеціалістів для реалізації проєкту в залежності від рівня кваліфікації

Спеціалізація	Junior рівень	Middle рівень	Senior рівень
Власник продукту	\$2700	\$3700	\$4000
Scrum-майстер	\$2250	\$3300	\$4000
Python інженер	\$1000	\$2700	\$5200
React інженер	\$900	\$2500	\$4750
iOS інженер	\$1000	\$2800	\$5000
Android інженер	\$850	\$2800	\$5200
Інженер з контролю якості (QA)	\$800	\$1800	\$3300
Дизайнер продукту (Product Designer)	\$1500	\$2500	\$3400
Дизайнер інтерфейсів (UI/UX Designer)	\$1000	\$1500	\$3000

Знову ж таки, наведені вище дані є орієнтовними. На сьогодні на ринку можна знайти як значно дешевших спеціалістів, так і значно дорожчих. Але у наших розрахунках на цьому етапі ми не будемо це враховувати.

Розраховуючи вартість витрат на розробку першої версії застосунку слід враховувати декілька речей - склад команди, яку необхідно буде мати при розробці першої версії, а також функціонал продукту та необхідні спеціалізації членів команди. Виходячи з запропонованого списку функціонала до першої версії продукту, нижче наведено верхньорівневі групи функціональностей разом з оцінкою приблизних витрат часу з розбивкою на такі етапи: опис вимог, дизайн, власне розробка та тестування (Таблиця 3.4.):

Таблиця 3.4.

Поетапний аналіз витрат часу на розробку застосунку

Функціональність	Приблизна оцінка витрат часу, дні			
	Опис вимог	Дизайн	Розробка	Тестування
Функціонал авторизації	3	5	15	3
Профіль користувача	10	10	30	10
Функціонал тренувань	20	15	30	10
Функціонал розкладу тренувань спортсмена	10	10	25	7
Функціонал управління доступом тренера до інформації про спортсмена	5	10	15	5
Функціонал статистики	10	15	15	5
Функціонал чату	5	5	10	5
Функціонал сповіщень	2	2	5	2
Функціонал підписок та платежів	3	3	5	5
Функціонал	5	3	10	5

адміністратора				
Всього	73	78	160	57

На кожному з цих етапів необхідне залучення різних спеціалістів. Але в цілому, можемо відмітити, що в першій версії продукту не буде потреби в розробці веб-інтерфейсів для тренера та спортсмена, а також не буде необхідності розробляти застосунки для смарт-годинників. Тому для розробки першої версії застосунку пропонується дещо скорочена команда, відносно повноцінної, що ми пропонували вище. Склад команди та кваліфікація спеціалістів наведені у таблиці 3.5. нижче:

Таблиця 3.5.

Скорочений варіант витрат на розробку застосунку

Напрямок	Спеціалізація	Кваліфікація	Вартість за місяць
Менеджмент	Власник продукту	senior	\$4000
	Scrum-майстер	middle	\$4000
Дизайн	Дизайнер продукту	senior	\$3400
Мобільна розробка	iOS інженер	senior	\$5000
	iOS інженер	middle	\$2800
	Android інженер	senior	\$5200
	Android інженер	middle	\$2800
Розробка серверної частини	Python інженер	senior	\$5200
	Python інженер	middle	\$2700
Контроль	Інженер з контролю якості (QA)	senior	\$3300

якості	Інженер з контролю якості (QA)	middle	\$1800
	Всього за місяць		\$40200

Відповідно до розрахунків, показаних вище, ми вбачаємо за доцільне мати на етапі розробки першої версії продукту команду з 11 спеціалістів. Вартість такої команди буде складати \$40200 за місяць. Знову ж таки, дана вартість є орієнтовною, проте її можна використовувати аби провести приблизні розрахунки.

Наступним кроком є визначення орієнтовного часу, необхідного на розробку подібного продукту. Найбільші витрати ресурсів, за оцінками наведеними вище, будуть йти саме на власне розробку застосунку. Як бачимо із запропонованого складу команди, ми плануємо мати по 2 спеціалісти на кожен напрямок розробки (iOS застосунок, Android застосунок, серверна частина). Проте, враховуючи, що це спеціалісти різного рівня кваліфікації та також беручи до уваги витрати часу на комунікацію, ми не можемо казати, що це зменшить час розробки вдвічі. Стандартним коефіцієнтом для розрахунку необхідного часу на створення продукту для таких команд в індустрії вважається 1.6. Тобто, якщо ми збільшуємо команду на 1 людину, то швидкість розробки зростає на 60 відсотків. Провівши простий розрахунок, та користуючись методом критичного шляху, можемо сказати, що власне на розробку необхідно буде виділити орієнтовно 100 днів.

Також слід враховувати, що етап дизайну має стартувати хоча б на 1 тиждень раніше, ніж розробка. І так само етап опису вимог бажано стартувати хоча б на 1, а краще на 2 тижні раніше, ніж етап дизайну. І плюс етап тестування зазвичай закінчується дещо пізніше після власне розробки, тому краще додати ще мінімум 1 тиждень. В такому разі, враховуючи додаткові умови, можемо сказати, що нам необхідно 100 днів на власне розробку, а також 15 днів до її початку та 5 днів після початку розробки на те, щоб створити першу версію продукту.

Після цього варто буде провести внутрішнє тестування першої версії продукту (альфа-тестування). А вже за цим - продукт буде готовий для того, щоб надавати його першим зовнішнім користувачам, та проводити бета-тестування.

Для спрощення розрахунків, ми рекомендуємо умовно вважати, що місяць складається з чотирьох тижнів. Таким чином, за тиждень команда буде вартувати $\$40200 / 4 = \10050 .

Нижче наведено таблицю 3.6. з розрахунком необхідного бюджету на створення першої версії продукту, відповідно до інформації, наведеної вище в даній роботі:

Таблиця 3.6.

Розрахунок бюджету на створення першої версії продукту

Етап роботи	Днів	Тижднів
Резерв часу на етап опису вимог до старту етапу дизайну	10	2
Резерв часу на етап дизайну до старту розробки	5	1
Резерв часу на розробку	100	20
Резерв часу на тестування після етапу розробки	5	1
Всього	120	24

Отже, ми маємо необхідні дані для розрахунку бюджету, необхідного для створення першої версії продукту. Вартість команди - \$10'050 на місяць. При цьому орієнтовно нам необхідно виділити 24 тижні для створення цієї версії.

$$\text{Бюджет} = \text{Вартість команди} * \text{Час створення продукту}$$

$$\text{Бюджет} = \$10'050 * 24 \text{ тижні} = \$241'200$$

Отже, виходячи з цього всього, для створення першої версії продукту орієнтовно необхідно виділити бюджет \$241'200. Слід також враховувати, що подібний бюджет враховує ризики, закладені в оцінки. Але при цьому також

варто враховувати додаткові витрати на необхідне обладнання, платні сервіси та непередбачувані витрати. Оцінити їх зараз дуже складно, тому, враховуючи, що вони не мають бути дуже суттєвими, ми пропонуємо їх не додавати до загального розрахунку необхідного бюджету на створення першої версії продукту.

3.4.2. Розрахунок потенційного ефекту для користувача від використання мобільного застосунку

Оцінка потенційного ефекту для користувачів доволі складно, враховуючи що мотивацією для використання застосунку є не стільки отримання додаткової економічної вигоди, скільки краща організація процесу тренувань спортсменів та його моніторингу зі сторони тренерів. Слід також зважати на той факт, що більшість потенційних користувачів мають різний рівень та специфіку даного процесу сьогодні, не використовуючи подібні продукти, або використовуючи аналоги від конкурентів. Тому виміряти такий ефект буде проблематично.

Зважаючи на це, для оцінки впливу нашого продукту на користувачів, ми пропонуємо використовувати оцінку рівня задоволення від користування продуктом самими користувачами після певного часу користування продуктом. Подібну оцінку пропонуємо провести декількома способами:

- П
роведення періодичних опитувань в самому застосунку;
- П
роведення анкетування через розсилку анкет на електронну пошту користувачів;
- П
роведення вибіркового інтерв'ю з користувачами.

На додачу до цього необхідним вважаємо покриття застосунків системою аналітики для відслідковування патернів використання продукту користувачами. Для цього вважаємо доцільним покриття функціоналу застосунків аналітичними тригерами на наступні події:

- П

ерша реєстрація у застосунку;

-

В

хід у застосунок;

-

О

плата підписки;

-

За

повнення профілю;

-

За

повнення специфічної інформації по виду спорту;

-

Д

одавання/видалення тренера прив'язаного до спортсмена;

-

П

ерегляд списку тренувань;

-

Д

одавання тренування до плану тренувань;

-

С

творення власного тренування;

-

З

міна існуючого тренування;

-

Д

одавання власної вправи;

-

П

ерегляд списку спортсменів;

-

В

иконання тренування та внесення інформації про нього;

-

Д

одавання відео до тренування;

-

П

ерегляд тренування тренером;

-	Д
одавання коментаря тренером до тренування;	
-	Д
одавання коментаря тренером до конкретної вправи в тренуванні;	
-	С
творення тренером тренування;	
-	С
творення тренером вправи;	
-	П
ерегляд статистики;	
-	С
творення звіту в розділі статистика;	
-	Ві
дправлення повідомлення в чаті;	
-	Ві
дправлення медіа в чаті.	

Покриття функціонала застосунку подібною розгалуженою системою аналітичних тригерів є критичним для оцінки патернів використання продукту. Адже відгук користувача може бути суб'єктивним, і планувати розробку наступних версій лише на основі відгуків ми вважаємо недоцільним. Необхідно також мати дані для того, аби наші рішення були більш обґрунтованими. Завдяки тригерам, що подані вище, ми матимемо змогу відповісти на наступні запитання:

-	Я
ка кількість/доля клієнтів заходить в застосунок більш як 1/3/5/10 разів?	
-	С
кільки разів користувачі заходять в застосунок за досліджуваний період?	
-	Я
к часто користувачі створюють власні тренування, а не використовують наявні?	
-	Я
к часто користувачі використовують наявні в системі тренування, змінюючи їх	

для створення нових/власних?

-

Я

кий відсоток тренерів переглядають тренування спортсменів?

-

Я

кий відсоток тренерів коментує тренування спортсменів?

-

Я

кий відсоток тренерів додає коментарі до конкретних вправ в тренуваннях?

-

Я

к швидко тренери переглядають тренування спортсмена, після того, як воно з'являється у системі?

-

Н

аскільки активно користувачі використовують чат у застосунках?

-

Н

аскільки часто користувачі використовують чат для відправлення меді файлів?

-

Я

кий відсоток користувачів користується статистикою?

-

Я

кий відсоток користувачів завантажує звіти у статистиці? Які саме та за який саме період це робиться найчастіше?

Маючи відповіді на дані питання ми можемо зрозуміти, який функціонал є найбільш популярним у нашому продукті, а, отже, потенційно саме він надає найбільшу цінність. Також ми бачитиме, з яким функціоналом є проблеми (якщо його користувач використовує значно менше, ніж інший).

Після цього ми зможемо обрати найбільш “цікавих” для нас користувачів для проведення користувацьких інтерв'ю з ними. Так ми зможемо сформулювати основні користувацькі патерни та моделі використання продукту, що допоможе значно краще сформулювати план для розвитку продукту у майбутньому.

3.4.3. Розрахунок очікуваного прибутку від реалізації програмного застосунку

Розрахунок очікуваного прибутку є досить складним завданням, адже має чисто прогностичний характер і ґрунтується здебільшого на різного роду гіпотезах та середніх показниках в індустрії. Як ми акцентували раніше, перша версія продукту буде включати лише один вид спорту - бейсбол. Для спрощення розрахунків, ми будемо орієнтуватися лише на ринок США та на спортсменів, які зараз тренуються у бейсбольних командах старших шкіл, коледжів та університетів. Це дає нам загальний ринок у понад 500'000 користувачів. При цьому тренерів ми до цього списку не будемо включати, також щоб спростити розрахунки.

Аналізуючи індустрію, в якій ми плануємо запускати наш продукт, можемо сказати, що для платформи iOS фактично не існує безкоштовних сервісів достатнього рівня, більшість додатків є платними. Це доволі важливий момент, адже в такому випадку потенційні користувачі будуть мати значно нижчий бар'єр до купівлі сервісу. Для того, аби розрахувати потенційний прибуток, припустимо, що ми зможемо отримати близько 5% ринку, тобто 25'000 користувачів.

Також для того, аби розрахувати потенційний прибуток, нам важливо визначитися з моделлю монетизації, яку ми будемо використовувати для нашого продукту. На сьогодні виділяють наступні найбільш популярні моделі монетизації для мобільних продуктів:

- Ре
клама в самому застосунку: модель, за якою використання застосунку є безкоштовним. Але при цьому в самому застосунку продаються місця, в яких рекламодавці можуть додавати свою рекламу;

- П
окупки через застосунок: дана модель також передбачає безкоштовне використання застосунку, але частина функціоналу може бути платною, з покупок якого власники продукту і отримують дохід;

- М

онетизація даних: в даному випадку власники продають дані про користувачів іншим бізнесам;

-

С

понсорська допомога: треті сторони, які зацікавлені в існуванні подібного продукту надають фінансові ресурси для його розробки та роботи, з чого продукт і отримує дохід;

-

П

латні програми: в цьому випадку саме використання застосунку є платним.

Для обраної нами сфери найбільш підходящим та популярним вибором є стягнення плати за використання самого застосунку. При цьому, ми вважаємо за доцільне, мати свого роду комбінацію декількох форм монетизації у майбутньому, додавши надалі можливість купувати певні послуги в самому застосунку. Але подібний функціонал буде мати сенс додавати вже після того, як продукт буде мати достатню активну аудиторію, для того, щоб інвестиції у створення додаткового функціоналу могли окупитися.

Якщо оцінювати вартість подібних сервісів на ринку, то можна сказати, що вона починається від \$5 і доходить до \$20. Враховуючи, що ми заходимо на вже існуючий ринок і основним завданням є отримання великої активної аудиторії (наша перша ціль - отримати 5% ринку), то за доцільне вважаємо вибір ціни місячної підписки на наш продукт \$5, для того, щоб ціна не ставала додатковим аргументом для потенційних клієнтів аби відмовитись від продукту.

Тож, маючи необхідні дані, ми можемо розрахувати потенційний дохід нашого сервісу при досягненні необхідної кількості клієнтів - 25000 (Таблиця 3.7.).

Таблиця 3.7.

Потенційний дохід нашого сервісу при досягненні необхідної кількості клієнтів

Показник	Значення
Цільова кількість активних клієнтів продукту	25'000

Вартість місячної підписки на продукт	\$5
Орієнтовна дохідність продукту від підписок на місяць	\$125'000

Додатково також слід враховувати витрати на маркетинг для залучення цієї кількості користувачів. Для цього можна використовувати різні методи, та ми не будемо деталізувати це питання, зважаючи на те, що це не є метою даної роботи. Відповідно до середніх показників в індустрії, вартість залучення нових користувачів до використання мобільних додатків в США буде складати орієнтовно від \$5 до \$10 за одного користувача. Маючи ці дані, можемо зробити наступні розрахунки (Таблиця 3.8.):

Таблиця 3.8.

Розрахунок орієнтовної вартості залучення цільової кількості користувачів

Показник	Значення
Цільова кількість активних клієнтів продукту	25'000
Вартість залучення нового користувача	\$10
Орієнтовна вартість залучення цільової кількості користувачів	\$250'000

Вище, оцінюючи вартість розробки першої версії продукту, ми вираховували, що вона складає приблизно \$250'000. Тому склавши вартість розробки застосунку, та вартість залучення цільової кількості активних користувачів, ми матимемо суму приблизно \$500'000.

Для розрахунку потенційної місячної дохідності продукту після виходу на необхідну кількість активних користувачів, також необхідно враховувати, що зберігаються витрати на утримання команди, яка має як надавати необхідну технічну підтримку існуючій версії продукту, так і займатися розробкою нової версії. Як зазначалося раніше, вартість команди складає \$10'050 на тиждень. Як і раніше, для спрощення розрахунків, ми вважаємо, що це складає \$40'200 на місяць. В такому разі ми можемо розрахувати орієнтовну місячну прибутковість

нашого продукту (Таблиця 3.9.):

Таблиця 3.9.

Розрахунок орієнтовної місячної прибутковості застосунку

Показник	Значення
Кількість активних клієнтів продукту	25'000
Вартість місячної підписки на продукт	\$5
Орієнтовна дохідність продукту від підписок на місяць	\$125'000
Витрати на утримання команди розробки на місяць	\$40'200
Орієнтовна прибутковість продукту на місяць	\$84'800

Звісно слід враховувати, що частина даного прибутку може бути направлена на пришвидшення темпів розробки наступної версії продукту, або ж на залучення додаткових користувачів (в наших розрахунках ми покриваємо лише 5 відсотків потенційного ринку). Проте якщо не враховувати дані статті витрат, ми можемо розрахувати терміни повернення інвестицій на створення подібного продукту (Таблиця 3.10.):

Таблиця 3.10.

Розрахунок термінів повернення інвестицій на створення подібного продукту

Показник	Значення
Сумарні витрати на розробку продукту та залучення цільової кількості користувачів	\$491'200
Орієнтовна прибутковість продукту на місяць	\$84'800
Орієнтовні терміни повернення інвестицій	5.79 місяця

Отже, виходячи з поданих вище розрахунків, ми можемо сказати, що даний продукт вимагає стартових інвестицій приблизно \$500'000. В такому разі, якщо вдасться залучити очікувану кількість активних клієнтів (25'000),

використовуючи модель монетизації, за якої користувачі будуть платити за використання нашого продукту напряду, очікувана прибутковість продукту буде складати \$84'800 на місяць та орієнтовні терміни повернення інвестицій - до 6 місяців після виходу на цільову кількість активних користувачів.

ВИСНОВКИ

Метою даної роботи було розробити детальний план створення сервісу для школярів та студентів, які займаються спортом на регулярній основі. Для цього було додатково проведено аналіз поточної ситуації сфери розробки мобільних застосунків, основних підходів до створення подібних продуктів, а також розглянуто основні технології, що можна для цього використовувати.

Якщо підсумовувати наш аналіз поточної ситуації у сфері розробки мобільних застосунків, то можна сказати, що дана сфера має величезний потенціал. І головною перевагою є те, що наразі всі тенденції та прогнози вказують на те, що ринок мобільних застосунків буде тільки продовжувати рости. Але при цьому слід відзначити, що дана сфера є також достатньо конкурентною, а самі користувачі через велику кількість пропозицій стають все більш вибагливими. Існує значна кількість можливостей до монетизації мобільних застосунків, вибір яких залежить від особливостей самого застосунку, маркетингової стратегії, цільової аудиторії тощо. Але при цьому і розробка мобільних програм являє собою не тривіальну задачу і вимагає як висококваліфікованих спеціалістів, так і значні бюджети.

Якщо говорити про методології до організації процесу розробки, то варто зазначити, очевидно, що відбулося значне збільшення проєктів розробки мобільних додатків. Попит на мобільні додатки зростає, а потенційна їх кількість практично необмежена. Відповідний метод Agile можна вибрати для певного проєкту та адаптувати до конкретних вимог на основі складності проєкту та розміру команди. Попри виявлені можливості для бізнесу, можна знайти дуже мало наукових публікацій, які стосуються конкретних проблем, з якими стикаються організації-розробники під час розробки програмного забезпечення для мобільних пристроїв. Хоча запропоновані моделі розробки мобільних додатків, які обговорюються та розглядаються в цій роботі, є обнадійливими, але все ще є теоретичними й на практиці вимагають значної адаптації під контекст команди, проєкту, клієнта тощо.

Незважаючи на багато схожих рис між розробкою програм для ПК та мобільних пристроїв, існують доволі суттєві особливості, характерні саме для мобільної розробки та в цілому ринку мобільних застосунків. Ці фактори неодмінно слід враховувати як під час планування створення мобільного застосунку, так і в процесі його вже життєвого циклу.

Нами також було проаналізовано компанію “ОНІКС-СИСТЕМЗ”, як потенційну базу для створення продукту. Дивлячись на представлені напрямки діяльності компанії, вона виглядає як гарний потенційний партнер для розробки мобільного застосунку, адже має достатню експертизу у потрібній та в дотичних областях. Також компанія має гарну репутацію, багато різноманітних нагород. А доволі великий розмір компанії свідчить про стабільність бізнес-моделі та наявність ресурсів для забезпечення достатньої гнучкості враховуючи потенційні виклики сьогоденного ринку.

В роботі було проаналізовано найбільш популярні та ефективні методи розробки мобільного програмного забезпечення. На основі даного аналізу також було розроблено алгоритм для вибору методу розробки мобільного програмного забезпечення виходячи з верхньорівневих характеристик самої програми та бізнес контексту замовника.

Наявні технологічні можливості в смартфонах та смарт-годинниках чи фітнес-браслетах надають достатньо багато можливостей для отримання даних про людину. Більшість цих даних може бути достатньої точності для їх використання у розрахунках необхідних показників та аналізі фізичного стану піддослідного.

Аналіз джерел свідчить, що розробка мобільних застосунків має величезний ринок, а запит користувачів все продовжує рости. При цьому технології мобільних телефонів, що також постійно покращуються, створюють нові можливості на даному ринку та дозволяють випускати застосунки зі все прогресуючим функціоналом, який може покривати більше проблематик, або робити це якісніше ніж до цього. Саме тому темою даної роботи й було вибрано розробку детального плану створення мобільного застосунку.

Обираючи сферу для розробки мобільного застосунку враховувалися наступні факти:

- наявна експертиза;
- об'єм ринку;
- платоспроможність та готовність платити за сервіс потенційних користувачів.

Виходячи з цього, було обрано розробку мобільного застосунку для школярів та студентів, які займаються спортом на регулярній основі. Основним цільовим ринком обрано Сполучені Штати.

Безпосередня розробка програмного застосунку для мобільного пристрою - це лише незначна частина роботи над готовим продуктом. Багато уваги та зусиль було приділено також плануванню цього процесу. Тому для досягнення мети, поставленої в даній роботі, було здійснено:

- Аналіз потенційних конкурентних переваг (шляхом формування верхньорівневого списку функціонала сервісу та виділення його конкурентних особливостей порівняно з наявними аналогами на ринку).
- Планування тестування гіпотез (створення алгоритму та етапів оцінки привабливості нашої ідеї для потенційних користувачів сервісу).
- Розрахунок економічної моделі отримання прибутку від застосунку (обрання моделі монетизації застосунку та обґрунтування даного вибору, розрахунок цільового рівня доходу, а також розрахунок необхідної кількості клієнтів для досягнення цього цільового рівня).
- Формування структури команди для створення продукту (оцінка необхідних

людських ресурсів, їх кваліфікації та технічних можливостей, спеціалізації та рівня залучення на відповідних етапах створення застосунку).

-

Р

озрахунок потенційних витрат на створення подібного продукту.

-

Р

розробка механізму оцінки ефективності реалізованого проєкту (визначення ключових метрик успішності проєкту та їх показників, а також їх обґрунтування).

-

В

изначення верхньорівневої архітектури проєкту (виділення ключових складових системи та створення схеми їх взаємодії між собою).

-

О

брано технології для реалізації продукту (з урахуванням засобів та технології для розробки клієнтської частини продукту, серверної частини, бази даних, інтерфейсу адміністратора системи).

-

О

брано методологію для організації роботи над застосунком (ця частина тісно пов'язана з визначенням структури команди для роботи над проєктом та включає визначення необхідних ролей в цій команді та моделей їх взаємодії між собою для досягнення необхідного результату в найбільш ефективний спосіб).

В результаті дослідження ми отримали дієздатний план для створення конкурентного сервісу, що цілком спроможний вирости в успішний бізнес-проєкт в майбутньому. При цьому даний план залишається достатньо гнучким та адаптивним для того, щоб відповідати умовам ринку, мінливості зовнішніх факторів та загального контексту.

У роботі також зазначали, що є декілька підходів до розробки мобільних застосунків і тому вибір який саме використовувати робився, виходячи з багатьох факторів, основні з яких це функціональні обмеження, обмеження по бюджету, наявність ресурсів та додаткові нефункціональні вимоги. Враховуючи

все це, ми обрали розробку мобільних застосунків саме нативними методами. Такий вибір в першу чергу було зроблено, тому що нам необхідно буде активно використовувати апаратну складову смартфонів, а також доволі значний функціонал продукту базується на отриманні даних зі смарт-годинників. Все це значно краще реалізується завдяки нативним методам, тому прийнято рішення не намагатися економити на основному функціоналі продукту. Адже проблеми з якістю саме в цьому функціональні можуть дуже негативно вплинути на сприйняття продукту юзером, а, отже, і на успіх продукту в цілому.

Виходячи з вище перерахованого, для розробки мобільних застосунків (інтерфейсів спортсмена та тренера) було обрано наступні технології:

- Д
для застосунку під операційну систему iOS: Swift;
- Д
для застосунку під операційну систему Android: Kotlin.

Додатково планується використовувати ряд сторонніх модулів та бібліотек, які направлені на спрощення та пришвидшення розробки.

Основною мовою програмування для розробки веб-частини було обрано JavaScript, а React обраний як фреймворк для роботи. Для реалізації серверної частини нами було обрано фреймворк, який будується на основі Python мови програмування - Django.

Обраною моделлю до організації роботи над застосунком є доволі гнучка модель яка базується на Scrum методології, то ж і запланований функціонал застосунку може (і має) змінюватись в процесі виконання проєкту, орієнтуючись на відгуки користувачів та інших стейкхолдерів при тестуванні проміжних результатів (інкрементів продукту). Також важливим моментом є те, що було заплановано лише функціонал системи для запуску першої версії продукту (MVP - мінімально життєздатного продукту), а функціонал для наступних версій на разі має залишитись лише на верхньорівневому рівні та в статусі більш “потенційного” функціоналу. Прийняття рішення про те, додавати його до продукту, чи, можливо, він потребуватиме певних змін і переосмислення, має

бути відкладено до подальших фаз розвитку продукту і буде залежати від результатів запуску першої версії продукту та відгуків користувачів.

Також, враховуючи наявність різних інтерфейсів, опис функціональної частини застосунку було запропоновано розділити відповідно до наступних ролей:

- Користувач-спортсмен;
- Користувач-тренер;
- Адміністратор.

Сервіс навіть у першій своїй версії повинен мати доволі розлогий функціонал, аби його було доцільно застосовувати спортсмену. При всьому цьому, даний функціонал не є спеціалізованим і покриває лише загальні потреби спортсмена. Саме тому великий акцент при випуску першої версії має бути на комунікацію з користувачами, аби зібрати список їх запитів та потенційних проблем, щоб сформувати список максимально бажаного та корисного функціоналу для наступних версій сервісу.

Також слід додати, що в першій версії пропонується на разі додавання лише одного виду спорту - бейсболу.

Нами було запропоновано список функціонала для першої версії, який, на нашу думку, буде достатньо для виведення продукту на ринок та для того, щоб почати залучати перших користувачів і отримувати критично важливий зворотній зв'язок від них. Надалі буде формуватись і список функціонала для наступних версій, який допоможе не тільки втримати вже існуючих користувачів продукту, а й масштабувати його, залучити нові аудиторії.

У першому розділі даної роботи нами були розглянуті декілька підходів до організації процесу роботи на подібних до нашого проєктах. Враховуючи наведені там переваги та недоліки методів, та з огляду на наявні особливості нашого продукту, ми пропонуємо використовувати саме SLeSS - гнучкий підхід, який інтегрує Scrum і Lean Six Sigma (LSS) як основу для організації власного процесу.

Враховуючи, що Scrum є основою до організації процесу відповідно до

обраного нами підходу, було визначено стандартну тривалість спринту/ітерації (2 тижні).

В роботі нами було запропоновано загальний опис організації процесу роботи над продуктом для його ефективної реалізації. Проте, важливо прокоментувати, що даний процес є лише правилами, за якими команді рекомендується почати працювати. Але важливою характеристикою обраного нами підходу до роботи є адаптивність та необхідність постійного покращення. Тому, скоріш за все, дані правила будуть видозмінюватись та доповнюватись у процесі, що є не тільки нормальним явищем, а й бажаним.

Також було визначено, що для створення першої версії продукту орієнтовно необхідно виділити бюджет \$241'200. Слід також враховувати, що подібний бюджет враховує ризики, закладені в оцінки. Але при цьому також варто враховувати додаткові витрати на необхідне обладнання, платні сервіси та непередбачувані витрати. Оцінити їх зараз дуже складно, тому, враховуючи, що вони не мають бути дуже суттєвими, ми пропонуємо їх не додавати до загального розрахунку необхідного бюджету на створення першої версії продукту.

Також нами було поставлено цільову кількість активних користувачів, яких ми хочемо залучити до використання першої версії продукту - 25 тисяч, що складає 5 відсотків від ринку потенційних користувачів у США. Провівши розрахунки ми визначили, що це буде коштувати приблизно \$250'000.

Виходячи з розрахунків, ми можемо сказати, що даний продукт вимагає стартових інвестицій приблизно \$500'000. В такому разі, якщо вдасться залучити очікувану кількість активних клієнтів (25'000), використовуючи модель монетизації, за якої користувачі будуть платити за використання нашого продукту напряду, очікувана прибутковість продукту буде складати \$84'800 на місяць та орієнтовні терміни повернення інвестицій - до 6 місяців після виходу на цільову кількість активних користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. C.Watson, J.McCarthy and J.Rowley. Consumer attitudes towards mobile marketing in the smartphone era/International Journal of Information Management, Volume 33, Issue 5, October 2013, Pages 840-849. URL: <https://doi.org/10.1016/j.ijinfomgt.2013.06.004>
2. A.Rohm, T.Gao, F.Sultan and M.Pagani. Brand in the hand: A cross-market investigation of consumer acceptance of mobile marketing/ Business Horizons September 2012/ 55(5): Pages 485–493. URL: <http://surl.li/orvsq>
3. E.Kim, J.Lin and Y.Sung. To App or Not to App: Engaging Consumers via Branded Mobile Apps Journal of Interactive Advertising/ January 2013/13(1): Pages 53-65. URL: <http://surl.li/orvsi>
4. I. Alnawas, F. Aburub. The effect of benefits generated from interacting with branded mobile apps on consumer satisfaction and purchase intentions Journal of Retailing and Consumer Services/ July 2016,31(1): Pages 313-322. URL: <http://surl.li/orvrw>
5. T.Natarajan, B. Senthil Arasu, D.Kasilingam. Understanding the intention to use mobile shopping applications and its influence on price sensitivity. Journal of Retailing and Consumer Services/ July 2017/37: Pages 8-22. URL: <http://surl.li/orvxy>
6. R.Wang, S.Kim, E.Malthouse. Branded apps and mobile platforms as new tools for advertising. The new advertising: Branding, content, and consumer relationships in the data-driven social media era/ January 2016, 2; Pages 123-156. URL: <http://surl.li/orwbc>
7. Kopalle, Kumar and Subramaniam. How legacy firms can embrace the digital ecosystem via digital customer orientation Journal of the Academy of Marketing Science November 2019, 48(3). URL: <http://surl.li/orwch>
8. <https://buildfire.com/g2-summer-2021/>

9. S.Arora, F.Hofstede, V.Mahajan. To free, or not to free: the implications of offering free versions for the performance of paid mobile apps. Journal of Marketing July 2017, 81(6)/ URL: <http://surl.li/orwfi>
10. Z. Fang, A. Santos. Inference on Directionally Differentiable Functions/ The Review of Economic Studies, Volume 86, Issue 1, January 2019, Pages 377–412, URL:<https://academic.oup.com/restud/article/86/1/377/5094886>
- 11.Tseng, S.M. and Lee, P.S. The Effect of Knowledge Management Capability and Dynamic Capability on Organizational Performance. Journal of Enterprise Information Management, (2014), 27, 158-179.URL: <http://surl.li/orwjf>
12. The State of Influencer Marketing 2019 : Benchmark Report [+Infographic] URL: <http://surl.li/orwkc> Last accessed: 28.11.2023).
13. The Top Mobile Apps, Games, and Publishers of 2020: Sensor Tower's Data Digest. URL: <https://sensortower.com/blog/q4-2020-data-digest> Last accessed: 28.11.2023).
14. Mobile Operating System Market Share United States of America. (2021, March). URL: <http://surl.li/orwnm> (Last accessed: 02.11.2023).
15. Mobile Application Market Size, Share & Growth| Industry Analysis, 2026. (n.d.). URL: <https://www.alliedmarketresearch.com/mobile-application-market> (Last accessed: 02.12.2023).
- 16.Holzer, A., & Ondrus, J. (2009). Trends in Mobile Application Development. Lecture Notes of the Institute for Computer Sciences. Social Informatics and Telecommunications Engineering, 55–64. URL: https://doi.org/10.1007/978-3-642-03569-2_6. (Last accessed: 28.11.2023).
17. Enabling the new normal - mobile app development in the COVID-19-Era. (2020, July 14). URL: <http://surl.li/orwmx> Last accessed: 28.11.2023).
18. Statista. (2021, February 4). Annual number of global mobile app downloads 2016–2020. URL: <http://surl.li/orwnq> (Last accessed: 28.11.2023).
- 19.Holzer, A., & Ondrus, J. (2010). Mobile Application Market: A Mobile Network Operators' Perspective. Lecture Notes in Business Information Processing, 186–191. URL: https://doi.org/10.1007/978-3-642-17449-0_19

20. Wilcox, M. (2013, March 1). How Much Is an Active User Worth? Retrieved May 11, 2021. URL: <http://surl.li/orwow>
21. Top App Development Companies (2021). (2021a, May 10). Retrieved May 11, 2021. URL: <https://www.businessofapps.com/app-developers/>
22. App Development Cost (2021). (2021, March 26). Retrieved May 11, 2021. URL: <http://surl.li/orwpt>
23. K. Flora, H., Wang, X., & V. Chande, S. (2014). An Investigation into Mobile Application Development Processes: Challenges and Best Practices. International Journal of Modern Education and Computer Science, 6(6), 1–9. URL: <http://surl.li/orwqg> (Last accessed: 28.11.2023).
24. Jabangwe, R., Edison, H., & Duc, A. N. (2018). Software engineering process models for mobile app development: A systematic literature review. Journal of Systems and Software, 145, 98–111. URL: <https://doi.org/10.1016/j.jss.2018.08.028> (Last accessed: 28.11.2023).
25. Joorabchi, M. E., Mesbah, A., & Kruchten, P. (2013). Real Challenges in Mobile App Development. 2013 ACM / IEEE International Symposium on Empirical Software Engineering and Measurement. Published. URL: <https://doi.org/10.1109/esem.2013.9> (Last accessed: 02.12.2023).
26. Francese, R., Gravino, C., Risi, M., Scanniello, G., & Tortora, G. (2017). Mobile App Development and Management: Results from a Qualitative Investigation. 2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems (MOBILESoft). Published. URL: <https://doi.org/10.1109/mobilesoft.2017.33> (Last accessed: 28.11.2023).
27. Kornilova, V. (2016, September 15). Perfect SWOT Analysis for Mobile App. How to Validate Your Ideas. URL: <http://surl.li/orwsc> (Last accessed: 08.12.2023).
28. “Agile-маніфест розробки програмного забезпечення” Agile Alliance, 2001. URL: <https://agilemanifesto.org/iso/uk/manifesto.html>
29. A.I. Wasserman, “Software Engineering Issues or Mobile Application Development, 2010. URL: <http://surl.li/orwtv>

30. Harleen K Flora, Swati V Chande and Xiaofeng Wang. Article: Adopting an Agile Approach for the Development of Mobile Applications. International Journal of Computer Applications 94(17):43-50, May 2014. URL: <https://www.ijcaonline.org/archives/volume94/number17/16454-6199>
31. Nelsen A. Rahul, Thierry S. Nouidui, Paul L. Ulaya, David D. Kiwia, The Impact of Agile Methods on the Software Projects Implementation and Management. American Journal of Industrial and Business Management Vol.13 No.4, April 2023. URL: <https://www.scirp.org/journal/paperinformation?paperid=124307>
32. Abrahamsson, P. (2005). Keynote: Mobile software development – the business opportunity of today, Proceedings of the International Conference on Software Development, Reykjavik, Iceland, pp. 20-23. URL: <http://surl.li/orwul>
33. Kannan N. (2011). “Mobile development: Why using an agile methodology makes sense”, September 2011. [Online]. <http://surl.li/orwxz>
34. Jordan B. Barlow, Justin Scott Giboney, Mark Jeffery Keith, David W. Wilson, Ryan M. Schuetzler, Paul Benjamin Lowry, Anthony Vance. “Overview and Guidance on Agile Development in Large Organizations”, Communications of the Association for Information Systems Vol. 29, Article 2, pp.25–44, 2011. URL: <http://surl.li/orxbs>
35. B. Boehm, R. Turner, “Balancing Agility and Discipline: A Guide for the Perplexed”, Boston, MA: Addison-Wesley. pp. 55–57, ISBN 0-321-18612-5, 2004.
36. K. Beck, “Extreme Programming Explained: Embrace Change”, Boston, MA: Addison-Wesley. ISBN 0-321- 27865-8, 1999. URL: <http://surl.li/orxcy>
37. Kupe Kupersmith, “Agile is a Fad”/ URL: <http://www.batimes.com/kupe-kupersmith/agile-is-a-Fad.html>
38. Dag Øivind Madsen. The Evolutionary Trajectory of the Agile Concept. “Viewed from a Management Fashion Perspective”. Social Sciences 9(5):69, 2020. URL: <http://surl.li/orxdf>

39. Luke Halliwell, “The Agile Disease”, URL: <http://lukehalliwell.wordpress.com/2008/11/16/the-agile-disease/>
40. Martin Fowler, “Using an Agile Software Process with Offshore Development”. URL: <http://martinfowler.com/articles/agileOffshore.html> July 2006.
41. What are pitfalls for agile development methodologies?, [Online] <http://stackoverflow.com/questions/359790/what-are-pitfalls-for-agile-development-methodologies> .
42. Luis Corral, Alberto Sillitti, Giancarlo Succi, “Software Development Processes for Mobile Systems: Is Agile Really Taking Over the Business?” 1st International Workshop on Mobile-Enabled Systems (MOBS 2013) in connection with ICSE 2013, pp. 19-24. IEEE. 2013. URL: <https://ieeexplore.ieee.org/abstract/document/6614218>
43. Andrea A. Janes, Giancarlo Succi, “The dark side of Agile software development”, Proceedings of Onward, 2012. pp. 215-228, 2012. URL: <https://dl.acm.org/doi/10.1145/2384592.2384612>
44. Pekka Abrahamsson, Tuomas Ihme, Kari Kolehmainen, Pekka Kyllönen, Outi Salo, “Mobile-D: An Agile Approach for Mobile Application Development”, OOPSLA 2004. URL: <https://dblp.org/pid/59/627.html>
45. Spataru A. C., “Agile Development Methods for Mobile Applications”, Master of Science Thesis submitted to Computer Science School of Informatics, University of Edinburgh, 2010. URL: <http://surl.li/orxfh>
46. Ko Dooms, Roope Kylmäkoski, “Comprehensive documentation made agile – experiments with RaPiD7 in Philips”, In Proceedings of the 6th International Conference on Product Focused Software Process Improvement - PROFES 2005, pp 224-233. URL: https://dl.acm.org/doi/abs/10.1007/11497455_19
47. Rahimian V ., Ramsin R., “Designing an Agile Methodology for Mobile Software Development: A Hybrid Method Engineering Approach”, Second International Conference on Research Challenges in Information Science, RCIS 2008, pp. 337-342. doi: 10.1109/RCIS.2008.4632123

48. Yang-Jae Jeong, Ji-Hyeon Lee, Gyu-Sang Shin, “Development Process of Mobile Application SW Based on Agile Methodology”, 10th International Conference on Advanced Communication Technology, ICACT 2008, vol.1, pp. 362-366. doi: 10.1109/ICACTION.2008.4493779
49. T.F.V. da Cunha, Valeria L. L. Dantas, Rossana M. C. Andrade, “SLeSS: A Scrum and Lean Six Sigma Integration Approach for the Development of Software Customization for Mobile Phones”, 25th Brazilian Symposium on Software Engineering, pp. 283-292, 2011. doi: 10.1109/SBES.2011.38
50. Christelle Scharff, Ravi Verma, “Scrum to support mobile application development projects in a just-in-time learning context”, Proceedings of the 2010 Workshop on Cooperative and Human Aspects of Software Engineering - CHASE 2010, pp. 25-31. doi: 10.1145/1833310.1833315
51. L. Williamson. Enterprise Class Mobile Application Development: A Complete Lifecycle Approach for Producing Mobile Apps (Developerworks) 1st Edition, IBM Press: 2012/ URL: <http://surl.li/orxje>
52. J. Dixon, J. Dehlinger, S. Dixon. Mobile Application Software Engineering: Challenges and Research Directions. URL: https://link.springer.com/chapter/10.1007/978-3-642-39473-7_21
53. S. M. Dye, K. Scarfone. A standard for developing secure mobile applications. URL: <https://doi.org/10.1016/j.csi.2013.09.005>
54. T. Butter, M. Aleksy, P. Bostan, M. Schader, “Context-aware User Interface Framework for Mobile Applications”, Proceedings of 27th International Conference on Distributed Computing Systems Workshops, ICCDCSW, 2007, pp. 39/ URL: <http://surl.li/orxlx>
55. Biraj Das, “How to test Mobile Applications – Emulators or Real Devices”, Congruent Blog, URL: <http://surl.li/orxmf>
56. M. Nijim, Lee Young, L. Bellam, “Hybum: Hybrid Energy Efficient Architecture for Mobile Storage Systems”, Proceedings of 9th International Conference on Information Technology: New Generations, 2012, pp. 214-220

57. R. Minelli, M. Lanza, “Software Analytics for Mobile Applications – Insights & Lessons Learned”, Proceedings of 17h European Conference on Software Maintenance and Reengineering, 2013, pp. 144-153
58. Flurry Analytics, About Us, Retrieved March 15, 2013 from <http://www.flurry.com/about-flurry>
59. Shahnam Taheri, “Advantages and Disadvantages of outsourcing”/ August 2013. URL: <http://surl.li/oshqf>
60. Rob. Aalders, “IT OUTSOURCING : Making it work,” 2002/Fujitsu Aust. Ltd. URL: <http://surl.li/oshtf>
61. S. Haider, G. Samdani, M. Ali, M. Kamran A Comparative Analysis of In-house and Outsourced Development in Software Industry. Foundation of Computer Science (FCS), NY, USA URL: <http://surl.li/oshxi>
62. M. Ali. A Comparative Analysis of In-house and Outsourced Development in Software Industry. URL: <http://surl.li/osifq>
63. C. Doyle, D. Tapper. “Evaluating the Benefits of IT Outsourcing,” 2007, no. 508. URL: <http://surl.li/osigk>
64. T. Kremic, O. I. Tukul, and W. O. Rom. “Outsourcing decision support: a survey of benefits, risks, and decision factors,” Supply Chain Manag. An Int. J., , 2006, vol. 11, no. 6, pp. 467–482. URL: <http://surl.li/osijs>
65. Satwik Seshasai, Amar Gupta ,2004 “Global Outsourcing of Professional Services,” p. p 1–12. URL: <http://surl.li/osinq>
66. R. A. Khan and S. U. Khan, 2014 “Communication and Coordination Challenges in Offshore Software Development Outsourcing Relationship From Vendors ’ Perspective :,” vol. 2014, no. October, pp. 1425–1429. URL: <http://surl.li/osioj>
67. B. Rowe. “Will Outsourcing IT Security Lead to a Higher Social Level of Security?,” , 2008, pp. 1–22. URL: <http://surl.li/osisg>
68. Cardinale M and Varley MC. Wearable training monitoring technology: Applications, challenges and opportunities. Int J Sport Physiol Perf: 1-23, 2016. URL: <https://pubmed.ncbi.nlm.nih.gov/27834559/>

69. Poh M-Z, McDuff DJ, and Picard RW. Non-contact, automated cardiac pulse measurements using video imaging and blind source separation. *Optics express* 18: 10762-10774, 2010. URL: <http://surl.li/osiwp>
70. Taylor JL, Amann M, Duchateau J, Meeusen R, Rice CL, and Taylor J. Neural Contributions to Muscle Fatigue: From the Brain to the Muscle and Back Again. *Med Sci Sport Exer*, 2016. URL: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5033663/>
71. Heathers JA. Smartphone-enabled pulse rate variability: An alternative methodology for the collection of heart rate variability in psychophysiological research. *Int J Psychophysiol* 89: 297-304, 2013. URL: <https://pubmed.ncbi.nlm.nih.gov/23751411/>
72. Mendiguchia J, Martinez-Ruiz E, Edouard P, Morin JB, Martinez-Martinez F, Idoate F, and Mendez-Villanueva A. A multifactorial, criteria-based progressive algorithm for hamstring injury treatment. *Med Sci Sport Exer*: 1-1, 2017. URL: <https://pubmed.ncbi.nlm.nih.gov/28277402/>
73. Reyes B, Reljin N, Kong Y, Nam Y, and Chon K. Tidal volume and instantaneous respiration rate estimation using a smartphone camera. *IEEE J Biomed Health Informatics*, 2016. URL: <https://pubmed.ncbi.nlm.nih.gov/26915142/>
74. Rogers SA, Whatman CS, Pearson SN, and Kilding AE. Assessments of mechanical stiffness and relationships to performance determinants in middle-distance runners. *Int J Sport Physiol Perf*: 1-23, 2017. URL: <http://surl.li/osjbd>
75. Sardana M, Saczynski J, Esa N, Floyd K, Chon K, Chong JW, and McManus D. Performance and usability of a novel smartphone application for atrial fibrillation detection in an ambulatory population referred for cardiac monitoring. *J Am Coll Cardiol* 67: 844, 2016. URL: <http://surl.li/osjct>
76. Mitchell K, Graff M, Hedt C, and Simmons J. Reliability and validity of a smartphone pulse rate application for the assessment of resting and elevated pulse rate. *Physiother Theory Pract* 32: 494-499, 2016. URL: <https://pubmed.ncbi.nlm.nih.gov/27459148/>

- 77.Laukkanen RM and Virtanen PK. Heart rate monitors: State of the art. J Sports Sci 16: 3-7, 1998. URL: <https://pubmed.ncbi.nlm.nih.gov/22587712/>
- 78.Nagahara R, Botter A, Rejc E, Koido M, Shimizu T, Samozino P, and Morin JB. Concurrent validity of GPS for deriving mechanical properties of sprint acceleration. Int J Sport Physiol Perf 12: 129-132, 2016. URL: <http://surl.li/osjpm>
- 79.Mantovani AM, Duncan S, Codogno JS, Lima MCS, and Fernandes RA. Different amounts of physical activity measured by pedometer and the associations with health outcomes in adults. J Phys Act Health 13: 1183-1191, 2016. URL: <https://pubmed.ncbi.nlm.nih.gov/27334678/>
- 80.Samozino P, Rejc E, Di Prampero PE, Belli A, and Morin JB. Optimal force-velocity profile in ballistic movements--altius: citius or fortius? Med Sci Sports Exerc 44: 313-322, 2012.URL: <http://surl.li/osjqn>
- 81.Moore IS. Is there an economical running technique? A review of modifiable biomechanical factors affecting running economy. Sports Med 46: 793-807, 2016. URL: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4887549/>
- 82.Morin JB, Dalleau G, Kyröläinen H, Jeannin T, and Belli A. A simple method for measuring stiffness during running. J Appl Biomech 21: 167-180, 2005. URL: <https://journals.humankinetics.com/view/journals/jab/21/2/article-p167.xml>
- 83.Giroux C, Rabita G, Chollet D, and Guilhem G. What is the best method for assessing lower limb force-velocity relationship? Int J Sports Med, 2014. URL: <https://pubmed.ncbi.nlm.nih.gov/25259590/>
- 84.Kraemer WJ and Ratamess NA. Fundamentals of resistance training: Progression and exercise prescription. Med Sci Sport Exer 36: 574-688, 2004. URL: <http://surl.li/osjwa>
- 85.Glatthorn JF, Gouge S, Nussbaumer S, Stauffacher S, Impellizzeri FM, and Maffiuletti NA. Validity and reliability of Optojump photoelectric cells for estimating vertical jump height. J Strength Cond Res 25: 556-560, 2011. URL: <https://europepmc.org/article/med/20647944>

86. Gonzalez-Badillo JJ and Sánchez-Medina L. Movement velocity as a measure of loading intensity in resistance training. *Int J Sports Med* 31: 347-352, 2010. URL: <http://surl.li/osjxe>
87. Schoenfeld BJ. Is there a minimum intensity threshold for resistance training-induced hypertrophic adaptations? *Sports Med* 43: 1279-1288, 2013. URL: <http://surl.li/oskap>
88. Plews DJ, Laursen PB, Stanley J, Kilding AE, and Buchheit M. Training adaptation and heart rate variability in elite endurance athletes: opening the door to effective monitoring. *Sports Med* 43: 773-781, 2013. URL: <https://link.springer.com/article/10.1007/s40279-013-0071-8>
89. Farina GL, Spataro F, De Lorenzo A, and Lukaski H. A smartphone application for personal assessments of body composition and phenotyping. *Sensors* 16: 2163, 2016. URL: <https://www.mdpi.com/1424-8220/16/12/2163>
90. Leong JY and Wong JE. Accuracy of three Android-based pedometer applications in laboratory and free-living settings. *J Sports Sci* 35: 14-21, 2016. URL: <http://surl.li/oskbz>
91. Losa-Iglesias ME, Becerro-de-Bengoa-Vallejo R, and Becerro-de-Bengoa-Losa KR. Reliability and concurrent validity of a peripheral pulse oximeter and health-app system for the quantification of heart rate in healthy adults. *Health Informatics J* 22: 151-159, 2016. URL: <https://pubmed.ncbi.nlm.nih.gov/25038201/>
92. Mitchell K, Gutierrez SB, Sutton S, Morton S, and Morgenthaler A. Reliability and validity of goniometric iPhone applications for the assessment of active shoulder external rotation. *Physiother Theory Pract* 30: 521-525, 2014. URL: <https://journals.sagepub.com/doi/full/10.1177/23259671231187297>
93. Folland JP and Williams AG. The adaptations to strength training. *Sports Med* 37: 145-168, 2007. URL: <https://ouci.dntb.gov.ua/en/works/7WRV8bO9/>
94. Lee JM, Kim Y, and Welk GJ. Validity of consumer-based physical activity monitors. *Med Sci Sports Exerc* 46: 1840-1848, 2014. URL: <https://pubmed.ncbi.nlm.nih.gov/24777201/>

- 95.Kälvesten J, Lui L-Y, Brismar T, and Cummings S. Digital X-ray radiogrammetry in the study of osteoporotic fractures: Comparison to dual energy X-ray absorptiometry and FRAX. Bone 86: 30-35, 2016. URL: <https://pubmed.ncbi.nlm.nih.gov/26921822/>
- 96.Chapman M, Larumbe-Zabala E, Gosss-Sampson M, Colpus M, Triplett NT, and Naclerio F. Perceptual, mechanical and electromyographic responses to different relative loads in the parallel squat. J Strength Cond Res: 1-1, 2017. URL: <https://pubmed.ncbi.nlm.nih.gov/28338528/>
- 97.Popescu AL, Ionescu RT, and Popescu D. Cardiowatch: A solution for monitoring the heart rate on a mobile device. UPB Scientific Bulletin 78: 63-74, 2016. URL: <http://surl.li/oskqy>
- 98.Naclerio F and Larumbe-Zabala E. Relative load prediction by velocity and the omni-res 0-10 scale in parallel squat. J Strength Cond Res in press, 2017. URL: <https://pubmed.ncbi.nlm.nih.gov/28538309/>
- 99.Cardinale M and Varley MC. Wearable training monitoring technology: Applications, challenges and opportunities. Int J Sport Physiol Perf: 1-23, 2016. URL: <https://pubmed.ncbi.nlm.nih.gov/27834559/>
100. Guede-Fernández F, Ferrer-Mileo V, Ramos-Castro J, Fernández-Chimeno M, and García- González MA. Real time heart rate variability assessment from Android smartphone camera photoplethysmography: Postural and device influences. Presented at Engineering in Medicine and Biology Society (EMBC), 2015 37th Annual International Conference of the IEEE, 2015. URL: <https://pubmed.ncbi.nlm.nih.gov/26737985/>
101. M cManus DD, Chong JW, Soni A, Saczynski JS, Esa N, Napolitano C, Darling CE, Boyer E, Rosen RK, and Floyd KC. PULSE-SMART: Pulse-based arrhythmia discrimination using a novel smartphone application. J Cardiovasc Electrophysiol 27: 51-57, 2016. URL: <https://pubmed.ncbi.nlm.nih.gov/26391728/>

102. H
ung PC-Y, Witana CP, and Goonetilleke RS. Anthropometric measurements from photographic images. *Computing Systems* 29: 764-769, 2004. URL: <http://surl.li/oskti>
103. Pe
art DJ, Shaw MP, and Rowley CG. An investigation into a contactless photoplethysmographic mobile application to record heart rate post-exercise: Implications for field testing. *Biomed Human Kinet* 7: 95-99, 2015. URL: <https://sciendo.com/article/10.1515/bhk-2015-0015>
104. K
arlen W, Gan H, Chiu M, Dunsmuir D, Zhou G, Dumont GA, and Ansermino JM. Improving the accuracy and efficiency of respiratory rate measurements in children using mobile devices. *PLOS ONE* 9: e99266, 2014. URL: <http://surl.li/oskuv>
105. H
abibi E, Soury S, and Zadeh AH. Precise evaluation of anthropometric 2D software processing of hand in comparison with direct method. *J Med Signals Sensors* 3: 256-261, 2013. URL: <http://surl.li/oskvp>
106. Ni
elsen RO, Buist I, Parner ET, Nohr EA, Sørensen H, Lind M, and Rasmussen S. Foot pronation is not associated with increased injury risk in novice runners wearing a neutral shoe: A 1-year prospective cohort study. *Brit J Sport Med* 48: 440-447, 2014. URL: <http://surl.li/oskwd>
107. S
un Y, Hu S, Azorin-Peris V, Greenwald S, Chambers J, and Zhu Y. Motion-compensated noncontact imaging photoplethysmography to monitor cardiorespiratory status during exercise. *J Biomed Opt* 16: 077010-077010-077019, 2011. URL: <https://pubmed.ncbi.nlm.nih.gov/21806290/>
108. D
el Rosario MB, Redmond SJ, and Lovell NH. Tracking the evolution of

- smartphone sensing for monitoring human movement. *Sensors* 15: 18901-18933, 2015. URL: <http://surl.li/oskwu>
109. M
 orin JB, Slawinski J, Dorel S, de villareal ES, Couturier A, Samozino P, Brughelli M, and Rabita G. Acceleration capability in elite sprinters and ground impulse: Push more, brake less? *J Biomech* 48: 3149-3154, 2015. URL: <http://surl.li/oskxg>
110. M
 uñoz-López M, Marchante D, Cano-Ruiz MA, Chicharro JL, and Balsalobre-Fernández C. Load, force and power-velocity relationships in the prone pull-up exercise. *Int J Sport Physiol Perf*: 1-22, 2017. URL: <http://surl.li/oskxr>
111. H
 ewit J, Cronin J, and Hume P. Multidirectional leg asymmetry assessment in sport. *Strength Cond J* 34: 82-86, 2012. URL: <http://surl.li/oslac>
112. Br
 own SR, Feldman ER, Cross MR, Helms ER, Marrier B, Samozino P, and Morin J-B. The potential for a targeted strength training programme to decrease asymmetry and increase performance: A proof-of-concept in sprinting. *Int J Sport Physiol Perf* in press, 2017. URL: <https://pubmed.ncbi.nlm.nih.gov/28338355/>
113. C
 howdhury EA, Western MJ, Nightingale TE, Peacock OJ, and Thompson D. Assessment of laboratory and daily energy expenditure estimates from consumer multi-sensor physical activity monitors. *PLOS ONE* 12: e0171720-e0171720, 2017. URL: <http://surl.li/oslax>
114. T
 hompson WR. Worldwide Survey of Fitness Trends for 2017. *ACSM's Health & Fitness Trends*, 2016. URL: <http://surl.li/oslbi>
115. Fl
 att AA and Esco MR. Validity of the ithlete TM smart phone application for

- determining ultra-short-term heart rate variability. *J Human Kinetics* 39: 85-92, 2013. URL: <https://pubmed.ncbi.nlm.nih.gov/24511344/>
116. Pelegris P, Banitsas K, Orbach T, and Marias K. A novel method to detect heart beat rate using a mobile phone. Presented at Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE, 2010. URL: <https://ieeexplore.ieee.org/document/5626580>
117. Buchheit M, Gray A, and Morin J-B. Assessing stride variables and vertical stiffness with GPS- embedded accelerometers: Preliminary insights for the monitoring of neuromuscular fatigue on the field. *J Sport Sci Med* 14: 698-701, 2015. URL: <https://pubmed.ncbi.nlm.nih.gov/26664264/>
118. Ferriero G, Sartorio F, Foti C, Primavera D, Brigatti E, and Vercelli S. Reliability of a new application for smartphones (DrGoniometer) for elbow angle measurement. *PM&R* 3: 1153- 1154, 2011. URL: <https://pubmed.ncbi.nlm.nih.gov/22192326/>
119. Bunde MW, Powell MO, and Ryan LJ. Design and testing of a high-speed treadmill to measure ground reaction forces at the limit of human gait. *Med Eng Physics* 37: 892-897, 2015. URL: <https://pubmed.ncbi.nlm.nih.gov/26143150/>
120. Haldson SL. Monitoring Training Load to Understand Fatigue in Athletes. *Sports Med* 44: 139- 147, 2014. URL: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4213373/>
121. Ockendon M and Gilbert RE. Validation of a novel smartphone accelerometer-based knee goniometer. *J Knee Surgery* 25: 341-346, 2012. URL: <https://pubmed.ncbi.nlm.nih.gov/23150162/>
122. Laffaye G, Wagner PP, and Tombleson TIL. Countermovement jump height:

- Gender and sport-specific differences in the force-time variables. *J Strength Cond Res* 28: 1096-1105, 2014. URL: <https://pubmed.ncbi.nlm.nih.gov/23838969/>
123. S
haw MP, Robinson J, and Peart DJ. Comparison of a mobile application to estimate percentage body fat to other non-laboratory based measurements. *Biomed Human Kinet* 9: 94-98, 2017. URL: <https://sciendo.com/article/10.1515/bhk-2017-0014>
124. B
ort-Roig J, Gilson ND, Puig-Ribera A, Contreras RS, and Trost SG. Measuring and influencing physical activity with smartphone technology: A systematic review. *Sports Med* 44: 671-686, 2014. URL: <https://pubmed.ncbi.nlm.nih.gov/24497157/>
125. F
olke M, Cernerud L, Ekström M, and Hök B. Critical review of non-invasive respiratory monitoring in medical care. *Med Biol Eng Comput* 41: 377-383, 2003. URL: <https://pubmed.ncbi.nlm.nih.gov/12892358/>
126. Colantonio E, Dâmaso AR, Caranti DA, Pinheiro MM, Tufik S, and Mello MTd. Clinical performance of 3-body fat measurements in obese adolescents 15 to 18 years-old. *Rev Bras Med* 72: 77-82, 2015. URL: <http://surl.li/oslyb>
127. Pi
cerno P, Iannetta D, Comotto S, Donati M, Pecoraro F, Zok M, Tollis G, Figura M, Varalda C, Di Muzio D, Patrizio F, and Piacentini MF. 1RM prediction: A novel methodology based on the force–velocity and load–velocity relationships. *Eur J Appl Physiol*: 1-9, 2016. URL: <http://surl.li/oslzp>
128. Fe
rriero G, Vercelli S, Sartorio F, and Foti C. Accelerometer-and photographic-based smartphone applications for measuring joint angle: are they reliable. *J Arthroplasty* 29: 448- 449, 2014. URL: <http://surl.li/osmae>

129. B
utler RJ, Crowell HP, and Davis IM. Lower extremity stiffness: Implications for performance and injury. Clin Biomech 18: 511-517, 2003. URL: <https://pubmed.ncbi.nlm.nih.gov/12828900/>
130. E
hrmann FE, Duncan CS, Sindhusake D, Franzsen WN, and Greene DA. GPS and injury prevention in professional soccer. J Strength Cond Res 30: 360-367, 2015. URL: <https://pubmed.ncbi.nlm.nih.gov/26200191/>
131. M
ohamed H, Vaeyens R, Matthys S, Multael M, Lefevre J, Lenoir M, and Philppaerts R. Anthropometric and performance measures for the development of a talent detection and identification model in youth handball. J Sports Sci 27: 257-266, 2009. URL: <https://pubmed.ncbi.nlm.nih.gov/19153859/>
132. C
howdhury EA, Western MJ, Nightingale TE, Peacock OJ, and Thompson D. Assessment of laboratory and daily energy expenditure estimates from consumer multi-sensor physical activity monitors. PLOS ONE 12: e0171720-e0171720, 2017. URL: <http://surl.li/oslax>
133. M
ilani P, Coccetta CA, Rabini A, Sciarra T, Massazza G, and Ferriero G. Mobile smartphone applications for body position measurement in rehabilitation: A review of goniometric tools. PM&R 6: 1038-1043, 2014. URL: <https://pubmed.ncbi.nlm.nih.gov/24844445/>

ДОДАТКИ

SWOT-аналіз сфери розробки мобільних додатків

Сильні сторони	- індустрія розробки мобільних додатків – це галузь, що постійно зростає, і щороку до неї додаються мільярди користувачів; - даній галузі часто надається перевага над веб-сайтами та іншим програмним забезпеченням через простоту використання та доступність у портативних пристроях; - охоплюється величезна база користувачів завдяки широкій доступності;
Слабкі сторони	- доступність різноманітних апаратних конфігурації мобільних пристроїв вимагає підтримку від програми як різних платформ, так і різних технічних можливостей цих пристроїв для адекватного використання програми користувачем; - часто витрати на розробку мобільних програм вищі, ніж витрати на традиційну розробку веб-сайту; - продуктивність роботи програми напряму залежить від цільового пристрою; - у користувачів постійно існує занепокоєння щодо безпеки пристроїв та особистих даних; - отримання доходу може бути проблематичним для програм із меншою базою користувачів.
Можливості	- величезний попит на програми постійно зростає на поточному ринку [24]; - існує можливість залучати більшу та ширшу аудиторію за короткий проміжок часу; - зрозумілий механізм розширення програми для надання все більшої кількості функцій; - персоналізація програм робить кожен програму унікальною для її користувача;
Виклики	- розробка, що виконується під різні платформи, ускладнює надання індивідуального досвіду [25]. - встановлення програм за допомогою будь-якого іншого режиму, окрім офіційних магазинів програм є проблематичним, а часто - неможливим; - захист програм і даних користувача може бути проблемою, оскільки більшість телефонів мають відбитки пальців і засоби розпізнавання обличчя виявлення як способи аутентифікації [26]; - значні витрати на маркетинг і технічне обслуговування через високу конкуренцію, розвинений ринок, а тому і вимоги користувачів; - на сьогоднішній момент існує дуже конкурентний ринок і цілком очевидно, що конкуренція буде тільки збільшуватись [27].

Додаток Б

**Порівняння основних аспектів внутрішньої та аутсорсингової розробки
програмного забезпечення**

Аспект	Власна розробка	Аутсорсинг
Ресурси	Обмежені та відносно сталі	Більш варіативні через можливу спеціалізацію
Технології	Обмежені технологічні можливості зі спеціалізацією на визначених проектах	Більші технологічні можливості через можливу спеціалізацію на тих проектах, які потрібні
Вартість	Більші витрати для залучення висококваліфікованих спеціалістів	Менші витрати для залучення експертів через їх доступність
Експертиза	Вузька експертиза	Широка експертиза
Швидкість	Повільна і поступова	Швидка
Якість	Нижча якість до моменту стабілізації команди	Потенційна вища якість та більші можливості до керування якістю
Гнучкість	Менше гнучкості	Більше гнучкості
Законодавство	Менше законодавчих обмежень	Більше обмежень через потенційну співпрацю з компаніями з інших країн
Впровадження змін	Зміни легше впровадити та відстежувати	Зміни складніше впровадити та відстежувати
Комунікація	Комунікаційні процеси більш керовані через те, що вони локальні й замкнені більшою мірою в межах однієї організації.	Комунікаційні процеси нерідко значно складніше налаштовувати та управляти ними, тому що вони є міжорганізаційними.
Менеджмент	Управління є централізованим і зробити його ефективним значно легше	Менеджмент міжорганізаційних команд та процесів є значно складнішим, тож добитися його ефективності також складніше
Контроль	Більше можливостей контролювати бізнес-процеси	Менше можливостей контролювати бізнес-процеси
Безпека	Більше захищеності конфіденційної інформації	Менше захищеності конфіденційної інформації

Порівняльна таблиця методів розробки мобільних додатків

Показник	Гібридна розробка	Нативна розробка	Кросплатформ на розробка	Прогресивні веб-додатки
Швидкість виведення на ринок	Швидка	Повільна, якщо говорити про дві платформи	Середня	Швидка
Дизайн (UI/UX)	Складно досягти однакового нативного дизайну для обох платформ	Нативний вигляд та відчуття від взаємодії з програмою	Нативний вигляд та відчуття від взаємодії з програмою	Нативний вигляд та відчуття від взаємодії з програмою
Продуктивність	Низька	Висока	Майже як в нативній	Низька
Мови програмування	HTML, CSS, JS	Java/Kotlin для Android; Swift/Objective-C для iOS	Javascript, JSX, C#	HTML, CSS, JS
Функціональність додатків	Залежить від сторонніх бібліотек і має обмежені можливості	Є доступ до всіх нативних API та унікальних для платформи функціональностей	Залежить від сторонніх бібліотек, але має доволі широкі можливості	Залежить від сторонніх бібліотек і має обмежені можливості
Вартість розробки	Найнижча	Висока	Середня	Низька
Спосіб візуалізації	Браузер	Нативні ресурси платформи	Нативні ресурси платформи	Браузер
Інструменти	Ionic, Apache, Cordova, Visual Studio	Xcode, AppCode, Android Studio	React Native, Xamarin, Flutter	Polymer, LightHouse, Magento PWA Studio, Webpack Module Bundler
Кодова база	Єдина кодова база зі специфічними частинами під кожен з платформ	Окремі кодові бази під кожну з платформ	Єдина кодова база зі специфічними частинами під кожен з платформ	Єдина кодова база зі специфічними частинами під кожен з платформ

Додаток Г
Таблиця

Опис функціонала для ролі “користувач-спортсмен”

Функціональна група	Функціонал	Опис та коментарі
Авторизація	Авторизація за допомогою Google	Можливість створити акаунт у системі та увійти в систему за допомогою облікового запису Google
Авторизація	Авторизація за допомогою мобільного телефону	Можливість створити акаунт у системі та увійти в систему за допомогою мобільного телефону
Авторизація	Авторизація за допомогою електронної пошти	Можливість створити акаунт у системі та увійти в систему за допомогою електронної пошти
Профіль спортсмена	Додати основу інформацію про спортсмена	Можливість додати та змінити основну інформацію: ім'я, вік, навчальний заклад, фотографія, електронна пошта, мобільний телефон
Профіль спортсмена	Додати вид спорту	Можливість додати або змінити список видів спорту, які будуть відображатись у профілі спортсмена. Від цього залежатиме список іншої інформації, які буде заповнювати гравець, а також список доступних тренувань.
Профіль спортсмена	Додати тренера	Можливість додати або видалити тренера, який матиме доступ до інформації спортсмена, що стосується конкретного виду спорту, що є в системі, а також загальної інформації про спортсмена. До одного виду спорту буде можливість додавати до трьох тренерів одночасно.
Профіль спортсмена	Додати загальну спортивну інформацію про спортсмена	Можливість додати або змінити загальну спортивну інформацію, що буде доступна всім тренерам незалежно від виду спорту (зріст, вага, силові показники і т.п.)
Профіль спортсмена	Додати інформацію про конкретний вид спорту	Можливість додати або змінити інформацію, що буде стосуватись конкретного виду спорту та буде доступною лише для тренера з цього виду спорту

Тренування	Список доступних тренувань	Можливість переглянути список доступних тренувань у додатку та додати їх до свого тренувального плану.
Тренування	Створення та редагування тренувань	Можливість редагувати існуючі тренування та створювати нові: додавати чи змінювати вправи, їх кількість, інтенсивність та інші характеристики, додавати фото, відео та іншу інформацію для опису вправ, створювати власні вправи, змінювати дати, час та тривалість тренувань.
Тренування	Внесення інформації про тренування	Можливість запустити тренування та внести необхідну інформацію про виконання вправ, завантажити відео. Дана інформація потім буде доступна для тренерів того виду спорту, з якого було дане тренування.
Статистика	Перегляд статистики	Можливість переглянути зведену статистику з функціоналом фільтрації по видах спорту та вибором періоду за який виводити статистику (день, тиждень, місяць, рік).
Статистика	Сформувати звіт	Можливість сформувати звіт у форматі PDF для того, щоб поділитися ним із зацікавленими сторонами.
Чат	Текстові повідомлення	Можливість відправляти та отримувати текстові повідомлення у застосунку
Чат	Медіа повідомлення	Можливість відправляти та отримувати різного роду медіа файли у застосунку (відео, фото, аудіо, документи).
Загальний функціонал	Пуш-сповіщення	Можливість отримувати від застосунку пуш-сповіщення про активності, оновлення, повідомлення, необхідність оновити інформацію та інші.
Загальний функціонал	Емейл-сповіщення	Можливість отримувати від застосунку емейл повідомлення із загальною інформацією (статистика за період, звіт про тренування та інше)
Загальний функціонал	Налаштування	Можливість налаштовувати взаємодію із сервісом: сповіщення, інформація, яка доступна тренерам, візуальний вигляд застосунку.
Загальний функціонал	Зворотний зв'язок	Можливість комунікації зі службою підтримки для того, щоб відправити відгук про роботу застосунку, сформувати запит на додавання нового функціоналу чи сповістити про наявні

		проблеми.
--	--	-----------

Опис функціонала для ролі “користувач - тренери”

Функціональна група	Функціонал	Опис та коментарі
Авторизація	Авторизація за допомогою Google	Можливість створити акаунт у системі та увійти в систему за допомогою облікового запису Google
Авторизація	Авторизація за допомогою мобільного телефону	Можливість створити акаунт у системі та увійти в систему за допомогою мобільного телефону
Авторизація	Авторизація за допомогою електронної пошти	Можливість створити акаунт у системі та увійти в систему за допомогою електронної пошти
Профіль тренера	Додати основну інформацію про тренера	Можливість додати та змінити основну інформацію: ім'я, вік, навчальний заклад, фотографія, електронна пошта, мобільний телефон
Профіль тренера	Додати вид спорту	Можливість додати або змінити список видів спорту, які будуть відображатись у профілі тренера. Від цього залежатиме до яких видів спорту буде можливість у спортсменів додавати тренера
Спортсмени	Додати або видалити спортсмена	Можливість додати або видалити спортсмена, до інформації якого потім буде доступ. Для того, щоб додати спортсмена, тренер має отримати запрошення від цього спортсмена і підтвердити його у застосунку. Також тренер матиме можливість видалити спортсменів зі свого списку
Спортсмени	Переглянути інформацію	Переглянути загальну інформацію про конкретного спортсмена
Тренування	Список доступних тренувань	Можливість переглянути список доступних тренувань у додатку
Тренування	Створення та редагування тренувань	Можливість редагувати існуючі тренування та створювати нові: додавати чи змінювати вправи, їх кількість, інтенсивність та інші характеристики, додавати фото, відео та іншу інформацію для опису вправ, створювати власні вправи, змінювати дати,

		час та тривалість тренувань.
Тренування	Перегляд інформації про тренування	Можливість переглянути інформацію про виконання вправ, відео. Також можливість додати коментар про виконання конкретної вправи чи тренування загалом
Статистика	Перегляд статистики	Можливість переглянути зведену статистику з функціоналом фільтрації по видах спорту та вибором періоду за який виводити статистику (день, тиждень, місяць, рік).
Статистика	Сформувати звіт	Можливість сформувати звіт у форматі PDF для того, щоб поділитися ним із зацікавленими сторонами.
Чат	Текстові повідомлення	Можливість відправляти та отримувати текстові повідомлення у застосунку
Чат	Медіа повідомлення	Можливість відправляти та отримувати різного роду медіа файли у застосунку (відео, фото, аудіо, документи).
Загальний функціонал	Пуш-сповіщення	Можливість отримувати від застосунку пуш-сповіщення про активності, оновлення, повідомлення, необхідність оновити інформацію та інші.
Загальний функціонал	Емейл-сповіщення	Можливість отримувати від застосунку емейл повідомлення із загальною інформацією (статистика за період, звіт про тренування та інше)
Загальний функціонал	Налаштування	Можливість налаштовувати взаємодію із сервісом: сповіщення, інформація, яка доступна тренерам, візуальний вигляд застосунку.
Загальний функціонал	Зворотний зв'язок	Можливість комунікації зі службою підтримки для того, щоб відправити відгук про роботу застосунку, сформувати запит на додавання нового функціоналу чи сповістити про наявні проблеми.

Опис функціонала для ролі “адміністратор”

Функціональна група	Функціонал	Опис та коментарі
Авторизація	Авторизація за допомогою електронної пошти	Можливість увійти в систему за допомогою електронної пошти
Користувачі	Перегляд та редагування списку спортсменів	Можливість переглянути список всіх користувачів та детальну інформацію по кожному з них. Також має бути можливість перегляду та редагування статусу користувача в системі (активний чи неактивний). Можливість заблокувати користувача на певний час (наприклад, за порушення правил та норм користування сервісом).
Користувачі	Перегляд та редагування списку тренерів	Можливість переглянути список всіх користувачів та детальну інформацію по кожному з них. Також має бути можливість перегляду та редагування статусу користувача в системі (активний чи неактивний). Можливість заблокувати користувача на певний час (наприклад, за порушення правил та норм користування сервісом).
Тренування	Перегляд та редагування списку тренувань	Можливість переглядати та редагувати тренування наявні в системі, а також плани тренувань користувачів.
Тренування	Перегляд та редагування списку вправ	Можливість перегляду та редагування списку вправ, включно з медіа файлами, прикріпленими як опис до вправ.
Платежі	Перегляд історії платежів за підписку	Можливість перегляду списку платежів, а також інформації про кожен конкретний платіж.
Зворотній зв'язок	Перегляд запитів від користувачів	Перегляд списку запитів від користувачів, а також можливість перегляду та зміни статусів по цим запитам.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЦЕНТРАЛЬНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ
УНІВЕРСИТЕТ (повне найменування вищого навчального закладу)

Факультет економічний
Кафедра економічної теорії, маркетингу та економічної кібернетики

Освітній рівень магістр

Галузь знань 05 Соціальні та поведінкові науки

Спеціальність 051 Економіка (спеціалізація Економічна кібернетика)
(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри ЕТМЕК
д.е.н, проф. Жовновач Р.І

«__» _____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА
Бойка Андрія Юрійовича

1. Тема роботи Розробка застосунку для вимірювання показників фізичної активності спортсменів. Затверджена наказом по університету № _____

2. Термін здачі студентом закінченої роботи _____ 3.
Вихідні дані до роботи _____ праці вітчизняних і зарубіжних науковців з проблеми дослідження розробки застосунків, в т.ч. Для вимірювання фізичної активності спортсменів; періодичні фахові видання; первинні документи, а також внутрішня інформація "ОНИКС-СИСТЕМЗ", ресурси мережі Internet, наукові розробки в галузі створення мобільних додатків, результати власних досліджень. 4.
Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) _____ 1.
Аналіз інструментів, методів та засобів розробки застосунків для мобільних пристроїв

2. Аналіз технологій для реалізації програмного застосунку _____

3. Практична реалізація застосунку _____

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

таблиць- 13, рисунків - 5, додатків - 6

2

6. Консультанти розділів проєкту (роботи)

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

7. Дата видачі завдання 03 вересня 2023 р _____

Підпис керівника _____ к.е.н., доц. Вишневська В.А.

Завдання прийнято до виконання 03 вересня 2023 р

Підпис студента _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)		Додаткові строки
		По плану	Фактично	
1	Вибір теми дипломної роботи	11.06-22.06.23	11.06-22.06.23	
2	Одержання індивідуального завдання	03.09.23	03.09.23	
3	Складання календарного плану-графіка написання дипломної роботи	03.09-07.09.23	06.09.23	
4	Підготовка до виконання дипломної роботи: вивчення літератури, складання плану тощо	08.09-14.09.23	08.09-14.09.23	
5	Уточнення плану дипломної роботи та календарного плану-графіка, виходячи зі специфіки базового підприємства, установи	15.09-21.09.23	17.09.23	
6	Підготовка: - першого розділу; - другого розділу; - третього розділу; - висновки; - здача науковому керівникові	21.09-02.10.23 03.10-17.10.23 18.10-24.10.23 25.10-27.10.23 27.10-29.10.23	21.09-02.10.23 03.10-17.10.23 18.10-24.10.23 25.10-27.10.23 27.10-29.10.23	
7	Доопрацювання дипломної роботи з урахуванням зауважень наукового керівника	29.10-02.11.23	29.10-02.12.23	
8	Написання та оформлення роботи в остаточному варіанті	03.11-05.11.23	03.12-05.12.23	

9	Попередній захист дипломної роботи на кафедрі	06.11-07.11.23	17.12.23	
10	Одержання відгуку наукового керівника	07.11-11.11.23	20.12.23	
11	Одержання рецензії зовнішнього рецензента	12.11-19.11.23	22.12.23	
12	Висновок завідувача кафедри	20.11-10.12.23	25.12.23	
13	Захист дипломної роботи	02.12-22.12.23	17.01.24	

Здобувач _____ Бойко А.Ю.
(підпис) (прізвище та ініціали)

Керівник проєкту (роботи) _____ к.є.н, доц. Вишневська В.А.
(підпис) (прізвище та ініціали)

