Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет Комп'ютерні технології та системи
Кафедра Комп'ютерні інформаційні технології

Пояснювальна записка

до кваліфікаційної роботи
магістра

на тему: «Дослідження часової ефективності Ajax- запитів»
за освітньою програмою **121 Інженерія програмного забезпечення**
зі спеціальності: **121 Інженерія програмного забезпечення**

Виконав: студент групи ПЗ2222: / Сергій ТАРЯНИК /

Керівник: / Вадим АНДРЮЩЕНКО /

Нормоконтролер: /Світлана ВОЛКОВА /

Засвідчую, що у цій роботі немає
запозичень з праць інших авторів
без відповідних посилань.

Студент

Дніпро – 2024 рік

Ministry of Education and Science of Ukraine

Ukrainian State University of Science and Technologies

Faculty Computer technologies and systems
Department Computer information technology

Explanatory Note
to Master's Thesis

on the topic: «Investigating the time efficiency of ajax request»

according to educational curriculum 121 software engineering
in the Speciality: 121 software engineering

Done by the student of the group PZ2222: /Serhii TARIANYK/

Scientific Supervisor: /Vadym ANDRIUSHCHENKO /

Normative controller: /Svitlana VOLKOVA/

Dnipro – 2024

Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет: Комп'ютерних технологій і систем
Кафедра: Комп’ютерні інформаційні технології
Рівень вищої освіти: магістр
Освітня програма: Інженерія програмного забезпечення
Спеціальність: Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ____КІТ
_____ Вадим ГОРЯЧКІН
_____ грудня 2023 р.

ЗАВДАННЯ

На кваліфікаційну роботу ____Магістр_____
студенту Тарянику Сергію Валерійовичу_____.
1. Тема дипломної роботи: «Дослідження часової ефективності Ajax-запитів».
Керівник роботи: Андрющенко Вадим Олександрович
затверджені наказом 1196 ст від 05.12.2022 року
2. Строк подання студентом роботи 25.01.2024 року
3. Вихідні дані до дипломної роботи:
_____.
4. Зміст пояснювальної записки (перелік питань до розробки):
4.1. Аналіз сучасних інструментальних засобів розробки мобільних додатків
4.2. Розробка плану дослідження ефективності засобів розробки
5. Перелік демонстраційного матеріалу:
5.1. презентація;
5.2. демонстраційне відео.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів	Примітка
1	Вступ	03.03.23 – 14.04.23	
2	Аналіз сучасного стану дослідження проблеми за науковими літературними джерелами	15.04.23 – 22.08.23	10%
3	Аналіз сучасного стану програмно-апаратного забезпечення, яке потребує вдосконалення для вирішення проблем дослідження	23.08.23 – 18.10.23	
4	Постановка задачі, технічне завдання	19.10.23 – 20.10.23	30%
5	Розробка інструментальних засобів дослідження	08.11.23 – 13.11.23	
6	Виконання досліджень	14.11.23 – 18.11.23	60%
7	Оформлення статті у фаховий журнал	24.11.23 – 28.11.23	
8	Оформлення пояснювальної записки	29.11.23 – 05.12.23	
9	Розробка демонстраційних матеріалів	02.12.23 – 10.01.24	100%
10	Подання кваліфікаційної роботи до кафедри	12.01.24	
11	Захист кваліфікаційної роботи на засіданні Екзаменаційної комісії	25.01.24	

Студент: _____ Сергій ТАРЯНИК

Керівник роботи: _____ доц. Вадим АНДРЮЩЕНКО

РЕФЕРАТ

Опис магістерської дисертації складається з 122 сторінок, 16 рисунків, 3 таблиць, 5 додатків та 24 джерел.

Предметом дослідження є процес виконання сучасних мобільних додатків.

Об'єктом кваліфікації є створення графічного програмного засобу з використанням Ajax-запитів.

У вступі розглянуто аналіз і сучасний стан проблеми, визначено мету, сферу та застосування кваліфікаційної роботи, обґрунтовано актуальність теми та уточнено постановку задачі.

У першому розділі проведено аналіз теми, визначено актуальність завдання та мету розроблення, поставлено завдання та виявлено потреби в програмній реалізації, технологіях і програмних засобах.

У другому розділі розглядаються наявні рішення, обрані платформи для розроблення ПЗ, проєктування та розроблення, визначається структура застосунку, алгоритми та їхні функції, описується поведінка застосунку, а також його виклик та завантаження, вхідні дані та вихідні дані, а також визначаються межі пристрою.

Розділ 3 описує програмування відповідно до обраного варіанта, а також описує розробку моделі проєкту. У ньому також описується програмування.

У розділі 4 описується етап тестування розробленого програмного продукту, де тестування розділено на кілька етапів, а також представлено використовуване стороннє програмне забезпечення та результати тестування.

Ключові слова: застосунок, веб-ресурс, сайт, конфігурація, система, Ajax, JavaScript, XML, HTML, CSS, операційна система.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАК, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

AJAX - це підхід до створення інтерактивних інтерфейсів веб-додатків, де обмін даними між браузером і веб-сервером відбувається у фоновому режимі.

CSS (Cascading Style Sheets - каскадні таблиці стилів) - це таблиці стилів для оформлення контенту сторінки відповідно до написаних правил. Стиль тексту, організація блоків на сторінці, анімація - все це визначається за допомогою каскадних таблиць стилів;

XML (Extensible Markup Language - розширювана мова розмітки) - стандарт створення мови розмітки для обміну ієрархічно структурованими даними між різними додатками, особливо в Інтернеті [1].

XML-документи складаються з текстових символів і є придатними для читання людиною; HTML - Hypertext Markup Language - є основним будівельним блоком Інтернету. При визначенні змісту і структури веб-контенту часто використовуються технології, відмінні від HTML, для визначення зовнішнього вигляду (CSS), а також функцій і поведінки (JavaScript) веб-сторінок;

JavaScript - багатопарадигмальна мова програмування; JavaScript є багатопарадигмальною мовою програмування і підтримує об'єктно-орієнтоване, імперативне та функціональне програмування.

iOS - мобільна операційна система від Apple.

ВСТУП

Актуальність дослідження. Сучасне розуміння Інтернету істотно змінилося відтоді, як сфера інформаційних технологій почала широко поширюватися. На сьогоднішній день розвиток Інтернету зазнав значних змін, які дозволили побудувати абсолютно новий погляд на інтернет-технології та їх застосування. Сьогодні всі веб-ресурси в Інтернеті ґрунтуються на інноваційних принципах: для розроблення і створення веб-ресурсів (сайтів) використовується безліч веб-технологій. Інноваційним підходом до розроблення веб-додатків є Ajax - технологія асинхронного передавання даних.

Сьогодні смартфони стали незамінним гаджетом для кожної людини. У наші дні часто можна зустріти людей з кількома мобільними пристроями, навіть якщо у них немає комп'ютера. За даними дослідницької компанії Gartner, у 2019 році у світі було продано майже 1,5 мільярда смартфонів (1,4 мільярда попереднього року). Відповідно до цього кількість мобільних застосунків стрімко зростає з кожним днем, що призвело до появи нових інструментів розроблення мобільних застосунків і модифікацій наявних інструментів. Нині у світі існує велика кількість інтегрованих засобів розробки програмного забезпечення.

Популярність мобільних пристроїв продовжує зростати у всьому світі. Сьогодні користувачі проводять все більше часу на своїх смартфонах і використовують їх для різних цілей (соціальні мережі, електронна пошта, карти, новини, відео, комерційні додатки тощо). У зв'язку з цим управління економікою вимагає комплексного використання сучасних інформаційних технологій. Широкі можливості мобільних пристроїв щодо збору, обробки та публікації необхідної інформації дозволяють значно підвищити якість економічних розрахунків та зробити процес обґрунтування економічних рішень більш ефективним.

Саме тому розробка мобільних додатків є актуальною темою в IT-індустрії. Сучасні компанії, такі як Google, Apple та Microsoft, займаються розробкою мобільних платформ, включаючи мобільні операційні системи та інструменти розробки (SDK, software developer kit). Важливими особливостями мобільних пристроїв є обмежене живлення, невеликий розмір екрану та різноманітність сенсорів. Процес розробки мобільних додатків є технічно дуже складним і вимагає специфічних компетенцій, таких як об'єктно-орієнтоване програмування, знання SQL, баз даних та дизайну користувацького інтерфейсу, розуміння роботи в мережі та тестування програмного забезпечення.

Інтерактивність веб-інтерфейсів - важливий елемент дизайну веб-ресурсів. Інтерактивне використання мультимедійних форм.

Принципи взаємодії сучасних шаблонів дизайну з користувацьким інтерфейсом сайту. Наприклад, якщо користувач запитує певні дані з класичного сервера, а на сторінці відображається 16.

Доки вся сторінка не завантажиться, користувач бачить білий екран; за допомогою Ajax можна створити сторінку таким чином, щоб вона записувала, що сталося після кожного перемикання користувача.

IOS є найпопулярнішою мобільною платформою в США - до 2020 року на неї припадатиме понад 82 % усіх пристроїв, що підтверджує її універсальність і потенціал для подальших досліджень. Опанувавши матеріал для магістерської дисертації, студенти можуть набути таких компетенцій: уміння обирати інструменти для розроблення мобільних додатків залежно від умов і мети застосування.

Вивченням цієї проблеми займалася ціла низка вчених різних напрямів: С. Семериков, І. Тепликов, В. Семериков, І. Теплицький, М. Стрюк, С. Шокалюк та інші. Проблема використання різних інструментів для розробки мобільних додатків є предметом дослідження М. Малежика, Р. Горбатюка, П. Малежика та інших. Аналіз можливостей апаратних мобільних платформ та

інструментів для розроблення мобільних застосунків висвітлено в роботах В. Вакалюка, О. Ватоліної, К. Харченка та інших.

Проте, на сьогоднішній день існує потреба в дослідженнях, що об'єднують і систематизують наявну інформацію з цього питання. Мета Визначити залежності часової ефективності для мобільних додатків за допомогою ajax-запитів і засобів розробки.

Відповідно до поставлених цілей було визначено такі завдання

- Проаналізувати новітні засоби розробки мобільних додатків;
- Розробити план дослідження часової ефективності інструментів розробки мобільних застосунків з використанням ajax-запитів; та
- Реалізувати обраний алгоритм за допомогою обраного інструменту;
- Вивчити часову ефективність обраних інструментів для розробки сучасних мобільних додатків;

Об'єктом дослідження є процес виконання сучасного мобільного додатка.

Предметом дослідження є залежність ефективності часу виконання додатків, що використовують Ajax від інструментальних засобів розробки.

Для розв'язання поставлених завдань було використано такі методи дослідження: теоретичний і критичний аналіз літератури за темою дослідження; порівняння, узагальнення та синтез отриманої інформації; статистичний аналіз.

РОЗДІЛ 1. АНАЛІЗ СУЧАСНИХ ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ РАЗРОБКИ МОБІЛЬНИХ ДОДАТКІВ

1.1. Призначення та сфера застосування мобільних додатків

Розробка мобільних додатків - це процес розробки додатків для КПК, смартфонів, мобільних телефонів та інших невеликих портативних пристроїв. Ці додатки можуть бути попередньо встановлені на апаратне забезпечення під час виробництва, завантажені користувачем через різні платформи розповсюдження програмного забезпечення або стати клієнтськими (Javascript) чи серверними веб-додатками. Вони також можуть бути клієнтськими (Javascript) або серверними веб-додатками.

У світі є багато розробників. Це багатомільярдний ринок. Давайте розглянемо основні середовища виконання.

Майже всі веб-сайти сьогодні базуються на Ajax. Це пов'язано з тим, що майже кожен веб-проект має правила асинхронної передачі даних, які визначають і впливають на розробку та використання потужних ресурсів. Ajax розшифровується як Asynchronous JavaScript and XML. На практиці це взаємодія двох веб-технологій, зокрема мови програмування JavaScript для створення динамічних веб-сайтів та XML, розширюваної мови розмітки документів.

Android, iOS, Blackberry, Open Webos, Symbian OS, Samsung Bada і Windows Mobile підтримують стандартні бінарні файли, наприклад, ті, що працюють на процесорах певного форм-фактора (переважно архітектура ARM). Windows Mobile може бути скомпільована для архітектури x86 для налагодження на ПК без емуляції процесора, а формат переносних виконуваних файлів (PE) пов'язаний з NET Framework. Підтримуються Windows Mobile, Android, HP webos і iOS, які надають розробникам безплатні SDK та інтегровані середовища розробки.

Смартфони стають незамінними для всіх людей. У наші дні, навіть якщо

у вас немає комп'ютера, ви, найімовірніше, зустрінете людину з кількома мобільними телефонами - за даними Gartner, до 2020 року у світі буде продано майже 1,5 млрд смартфонів, порівнюючи з 1,4 млрд торік [1], тож кількість мобільних додатків стрімко зростає з кожним днем. Станом на 3 квартал 2017 року було розроблено понад 1 мільйон застосунків для iOS, які було завантажено понад 25 мільярдів разів [8]. Згідно з аналізом, проведеним у 2018 році, понад 67 % мобільних розробників розробили й опублікували свої застосунки з використанням платформи iOS [2]. 2021 Q1 кварталі IOS домінувала на ринку мобільних телефонів із часткою 84,1% [4]. Об'єктом дослідження даної магістерської дисертації є різноманітні мобільні додатки, створені на операційній системі IOS, що розв'язують та виконують певні алгоритми та відправляють тимчасові дані на сервер. У зв'язку з цим тема роботи є актуальною для аналізу та використання засобів розробки на операційній системі IOS.

iOS - це власна операційна система Apple для мобільних пристроїв. Спочатку вона була розроблена для iPhone, а потім оновлена для iPad, iPod Touch і Apple TV. Apple не дозволяє використовувати свою операційну систему на мобільних телефонах сторонніх виробників.

У сучасних умовах більшість програмних продуктів розробляється за допомогою інтегрованих середовищ розробки (IDE).

У IDE є незаперечні переваги: у принципі, процес компіляції, створення і запуску програм автоматизовано. Сучасні IDE для IOS-розробників повинні підтримувати Swift, Objective C, JavaScript і Flutter.

1.2. Постановка задачі

Показники, що характеризують рівень попиту на мобільні пристрої, в останні роки неухильно зростають. Статистика дозволяє дійти невтішного висновку про актуальність і доцільність розробки мобільних додатків [13].

Мобільні додатки - це програмні продукти, призначені для використання на мобільних пристроях з операційною системою. Мобільні додатки можуть бути встановлені на пристрій на заводі від виробника, завантажені з флеш-

накопичувача або завантажені з інтернет-магазину, який пропонує їх на платній або безоплатній основі.

Сьогодні розробка мобільних додатків - це не тільки прибутковий бізнес для веб-студій та фрілансерів, але й не обов'язкова умова для утримання команди програмістів. Обираючи фреймворк або асемблер, що потрібно знати новачкові і якими інструментами користуватися?

Однак, як показує практика, професійне програмне забезпечення сьогодні користується великим попитом. На цих програмах можна заробити чимало грошей, оскільки сучасні компанії не соромляться інвестувати в продукти, які можуть оптимізувати або спростити існуючі бізнес-процеси в тій чи іншій мірі.

Бізнес з розробки додатків величезний, і за даними Statista, очікується, що до 2021 року мобільні додатки принесуть приблизно 188,9 мільярда доларів США доходу через магазини додатків і внутрішню рекламу, зростаючи на 15% на рік.

Розробка додатків, як і розробка будь-якої іншої програми, є абсолютно непередбачуваною і надзвичайно трудомісткою. Перше, що потрібно зробити, це почати з визначення того, що насправді робить додаток і яким він має бути.

Після цього можна щотижня визначати необхідний час. Зазвичай на розробку на стороні сервера йде 10 тижнів, а на стороні клієнта - вісім тижнів, тобто загалом близько 18 тижнів.

Внутрішня розробка включає в себе

- Управління обліковими записами користувачів;
- Паролі;
- Інтеграція даних з додатками Частини 3;.
- Налаштування користувацького досвіду;
- Дизайн інтерфейсу включає:
- Тестування якості
- Оптимізацію;
- Налаштування користувацького інтерфейсу

- обробку даних (наприклад, локальні кеші для підвищення продуктивності); і

- синхронізацію офлайн використання додатку;

Вартість розробки програми тісно пов'язана зі складністю програми і залежить від багатьох факторів, включаючи вибір платформи (веб, андроїд, iOS, комбінація цих двох або всіх трьох).

Щоб додаток працював належним чином, всі елементи повинні взаємодіяти скоординовано. Архівний ресурс - це структура, що складається з кількох внутрішніх елементів. Надсилання інформації, Видалення інформації та Редагування інформації. У цій частині програми використовується технологія AJAX, яка дозволяє веб-сайту працювати у фоновому режимі, поки користувач продовжує переглядати сторінку, надсилаючи дані на веб-сервер без необхідності перезавантаження сторінки, на якій користувач увійшов в систему.

Однак вона не включає API, а розробка стандартних компонентів (основного інтерфейсу та бек-енду) займає близько 400 годин. Розробка плагіна середнього розміру з функціональністю користувацького інтерфейсу, підтримкою планшетів, інтеграцією API та внутрішнім сервером займе від 500 до 800 годин. Для дуже складних додатків, наприклад, тих, що підтримують кілька мов, може знадобитися інтеграція сторонніх додатків з анімацією, що налаштовується. Це найпоширеніший сервер, час розробки якого зазвичай становить від 800 до 1500 годин.

Існує ряд факторів, які визначають вартість розробки плагіна, і було б корисно поговорити більше про спеціалізовані фактори, з якими ви стикаєтесь.

Звичайно, всі люди різні, і у власників малого бізнесу або стартапів їх буде значно менше. Вони не знайдуть компанію, яка спеціалізується на розробці високоякісних додатків. Якщо ви не навчені практичній розробці додатків, є кілька варіантів програмного забезпечення, які можуть зробити всю важку роботу за вас без використання програмування. Це не займе багато часу і дуже легко протестувати.

Це також найпопулярніші мобільні платформи. На жаль, цей продукт і метод не спрацювали. Вони не просунули бренд у простір смартфонів, але сучасні мобільні додатки намагаються швидше проникнути на нові ринки і забезпечити кращий користувацький досвід на мобільних пристроях, щоб збільшити продажі.

Фреймворки - це основа розробки додатків, і використання фреймворків робить процес розробки простішим і приємнішим. Фреймворки містять ряд компонентів, основне призначення яких - допомагати створювати додатки. Без фреймворків розробникам довелося б писати всі необхідні додатки з нуля, витрачаючи на це більше часу.

Flutter - це програмний фреймворк з відкритим вихідним кодом, розроблений компанією Google для створення додатків для Android, iOS та Інтернету. Це основний спосіб створення додатків для Google Fuchsia і основний спосіб створення додатків для Google Fuchsia. Весь графічний інтерфейс Fuchsia побудований на Flutter.

IONIC - надає інструменти та сервіси для розробки гібридних мобільних, десктопних та прогресивних веб-додатків на основі новітніх технологій та практик веб-розробки з використанням таких веб-технологій, як CSS, HTML5 та Sass. Зокрема, мобільні додатки можуть бути створені з використанням цих веб-технологій і поширюватися через спеціалізовані магазини додатків для встановлення на апаратне забезпечення за допомогою Cordova або Capacitor.

Adobe Phonegap - це безкоштовний фреймворк з відкритим вихідним кодом для створення мобільних додатків, створений компанією Adobe Software з використанням JavaScript, HTML5, CSS3 та нативних мов програмування (наприклад, Objective-C), що дозволяє створювати мобільні додатки для всіх мобільних операційних систем (iOS, Android, Bada тощо) без необхідності їх знання. Готовий додаток компілюється як інсталяційний пакет для кожної мобільної операційної системи [15].

React Native - платформа мобільних додатків з відкритим вихідним кодом, створена компанією Facebook, яка використовується для розробки

додатків для Android, iOS, веб та UWP,

Розробники можуть розробляти додатки для веб, UWP та React Native, використовуючи React у поєднанні з можливостями цієї платформи, вони не використовують HTML або CSS. Натомість для керування користувацьким інтерфейсом використовуються потокові повідомлення JavaScript. React Native також дозволяє розробникам писати власний код на таких мовах, як Java для Android та Objective-C або Swift для iOS, що ще більше підвищує гнучкість та універсальність.

Cordova - це фреймворк для розробки мобільних додатків, розробку якого Adobe взяла на себе після передачі платформи PhoneGap до Apache Foundation. Водночас Adobe анонсувала продукт PhoneGap, заснований на спільній кодовій базі, який функціонально ідентичний Apache Cordova.

Xamarin - платформа з відкритим вихідним кодом, призначена для створення сучасних виробничих додатків для iOS, Android і Windows. Це рівень абстракції, який керує взаємодією між кодом, Xamarin працює в керованому середовищі, що реалізує такі функції, як розподіл пам'яті та збір сміття.

Оскільки фреймворки формують ядро майбутніх додатків, вибір правильних ресурсів для використання в розробці вимагає ретельного розгляду і суджень. Ви повинні розуміти, що майже всі фреймворки відрізняються один від одного, витратити багато часу на вивчення документації по використанню API, а потім зупинитися і зробити вибір для свого майбутнього додатку.

Coding-Free Mobile App Builder - це сервіс, створений розробниками, щоб спростити створення додатків і дати можливість навіть недосвідченим користувачам створювати власні програми. Він базується на таких фреймворках, як Ionic і Phonegap, але кінцевому користувачеві не потрібно вчитися кодуванню, оскільки програміст вже налаштував необхідне середовище і готові функції [14].

IBUILDAPP - американський сервіс, який підтримує російську мову. На

нашу думку, це хороший інструмент з великою кількістю шаблонів на різні теми, такі як нерухомість, кафе, ресторани, бізнес тощо. Обирайте готовий шаблон і починайте створювати. Недоліки - не оптимізований під російський ринок і висока ціна.

Appmachine цікавий тим, що розробники не лінуються і автоматизують багато існуючих процесів. Наприклад, веб-сайти можна легко перетворити на мобільні додатки за допомогою однієї URL-адреси, також можлива інтеграція з соціальними мережами, такими як Facebook і Twitter, RSS-каналами і каталогами зображень. Цей конструктор цікавий для створення невеликих додатків, але не підходить для бізнесу або корпоративних рішень.

Biznessapps - ще один американський додаток, який підходить для створення мобільних додатків для МСП, але має обмеження на російському ринку. Засновник має багато функцій, включаючи замовлення їжі, електронний магазин, програми лояльності, інтеграцію сторонніх даних, push-повідомлення та аналітику.

Наразі на ринку існує значна кількість розробників мобільних додатків, але слід зазначити, що 90% цих сервісів орієнтовані на Захід і не завжди є корисними для створення додатків, які використовують персональні дані, онлайн-платежі тощо.

1.3. Пошук найпопулярніших інструментів для розробки сучасних мобільних додатків

Існує так багато інструментів для розробки мобільного програмного забезпечення. Кожен, хто хоче почати розробку мобільних застосунків, задається питанням, які з них будуть найуспішнішими. У цій статті проводиться аналіз ринку найбільш популярних інструментів.

Щоб визначити, які інструменти є найпопулярнішими, я розглядаю такі критерії

1) Кількість запитів на форумі stackoverflow - оскільки stackoverflow є найпопулярнішим форумом серед програмістів, кількість запитів зазвичай

показує, наскільки часто люди використовують той чи інший інструмент. Основний недолік цього критерію полягає в тому, що на кількість запитів на форумі також впливають такі фактори, як документація по інструменту (чим гірша документація, тим більше про неї має бути написано на форумі);

2) Порівняння за Google Trends: цей критерій показує, як часто інструмент "шукають у Google"; один із найважливіших критеріїв у діаграмі Google Trends - побачити, як змінюється популярність інструменту;

3) Порівняння і статистика сторонніх ресурсів - ще один важливий критерій. Багато компаній і вчених, які займаються або цікавляться розробкою мобільних застосунків, досліджують і вивчають інструменти за тими чи іншими критеріями, тому важливо звертати увагу на їхні рейтинги і топ-рейтинги;

Stack Overflow - популярна система запитань і відповідей для професійних програмістів та ентузіастів. Це основний приватний сайт мережі Stack Exchange Network, створеної Джеффом Етвудом і Джоелом Спольскі 2008 року. Він містить запитання та відповіді з широкого кола тем, пов'язаних із програмуванням. Тепер у таблиці 1.1 представлено результати нашого власного дослідження та порівняння інструментів для розробки мобільних додатків.

Таблиця 1.1 – Таблиця порівнянь інструментальних засобів

	Рік випуску	Компанія розробник	Мова програмування	Кількість питань	Вартість
Swift	2015	Apple	Swift	290 тис.	Безкоштовно
ObjectiveC	1986	Apple	Objective-C	280 тис.	Безкоштовно
React Native	2016	Facebook	JS	97 тис.	Безкоштовно
Flutter	2018	Google	Dart	93 тис.	Безкоштовно
Cordova	2014	Adobe	JS	56 тис.	Безкоштовно

Xamarin	2013	Microsoft	C#	54 тис.	Ліцензія. $55
Native Script	2015	Telerik	JS	10 тис.	Безкоштовно

Як бачите, ми порівнюємо два нативні інструменти розробки - Swift і Objective C. Хоча кількість запитів на Stack Overflow майже однакова, Objective C був випущений 1986 року, а Swift - 2015-го. React Native і Flutter - найпопулярніші кросплатформні інструменти.

Далі розглянемо Google Trends, публічний веб-додаток корпорації Google, що базується на Google Search і показує, як часто шукають певний термін стосовно загальної кількості пошукових запитів у різних регіонах світу та різними мовами. Він показує. На рисунку 1.1 показано порівняння нативних інструментів, а на рисунку 1.2 - порівняння кросплатформних інструментів.

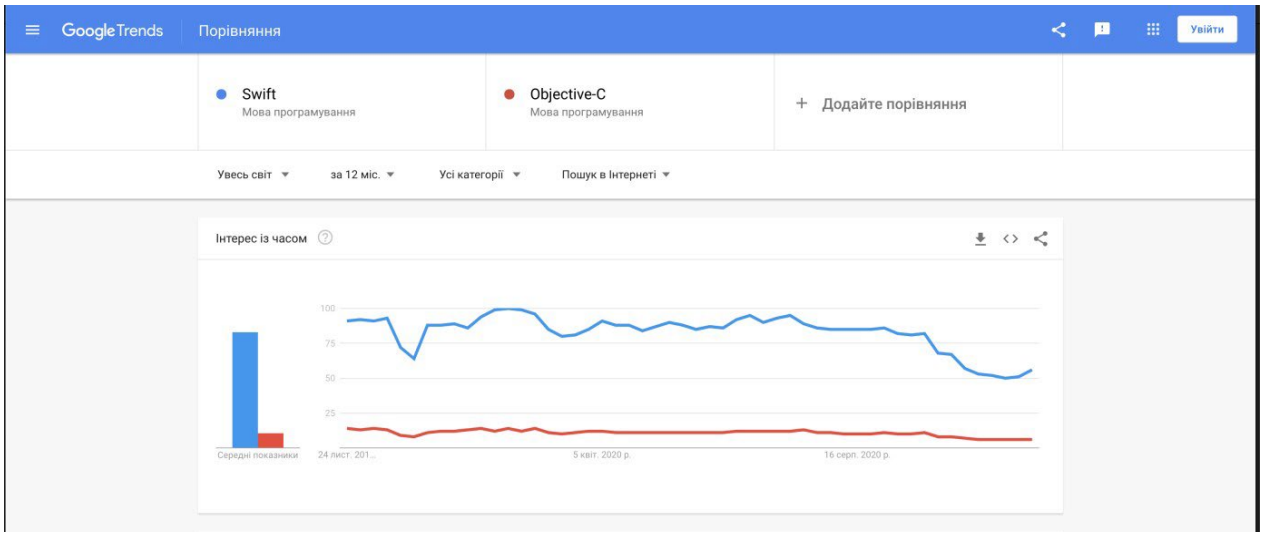


Рисунок 1.1 – Google Trends (Нативні)

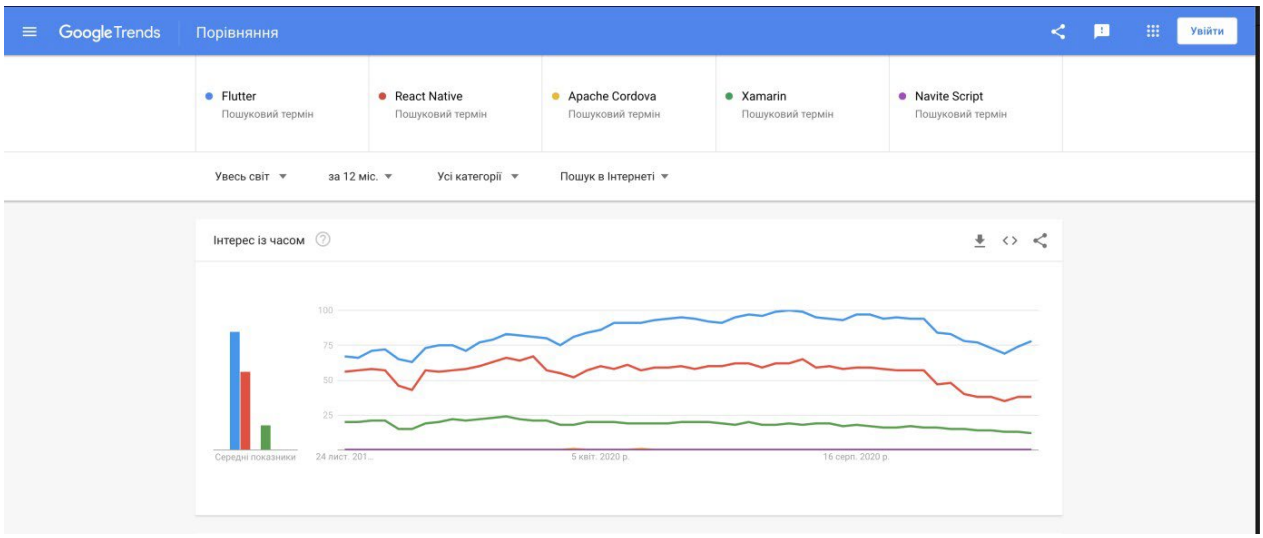


Рисунок 1.2 – Google Trends (Кросплатформенні)

Вісь абсцис показує популярність пошукового терміну відносно найвищої точки графіка для даного регіону та періоду часу; де найвища популярність 100 термінів означає вдвічі більшу популярність 50 термінів .

Вісь x показує часовий діапазон порівняння. Вона дає змогу побачити, як зростає або слабшає тенденція за певний період часу.

Далі розглянемо статистику та порівняння зі сторонніми компаніями, які проводили аналогічне дослідження.

ItCraft - всесвітньо визнана компанія з розробки мобільних додатків, яка успішно реалізувала понад 200 проектів з 2010 року.

Компанія спеціалізується на нативній та кросплатформенній розробці. У 2021 році компанія опублікувала статтю "Головні тренди у розробці мобільних додатків у 2021 році". У статті компанія представляє основні інструменти для розробки мобільних додатків. До топу увійшли наступні:

1. React Native: "З точки зору інтересу ринку, вимоги до розробки на React Native наразі очолюють список. Якщо у вас скромний бюджет на розробку мобільного додатку до 100 000 доларів США, не варто втрачати можливість отримати більше фінансування. Великі проекти також не цураються React Native - Ubereats, Pinterest, Walmart, Bloomberg та багато інших великих брендів використовують React Native і довели, що його

достатньо для їхніх потреб. "

2. Flutter: "Написаний на C++ з бібліотекою Google Skia, Flutter підтримує низькорівневий рендеринг. Тут починається "хороша" частина. За словами виробника, додатки, розроблені на Flutter, можуть відображатися безперервно зі швидкістю 120 кадрів в секунду. Цього більш ніж достатньо для плавного користувацького інтерфейсу".

3. Swift: "За останні п'ять років на зміну традиційній розробці на Objective-C прийшла Swift - мова програмування для iOS. Технології, інструменти та рішення, які підтримують цю мову, проклали шлях до нового покоління розробки додатків для найвимогливіших користувачів". Вибір додатків для iOS обмежений лише можливостями мобільного пристрою. Якщо ви хочете розробляти красиві, якісні, високопродуктивні інновації, нативна iOS - найкращий вибір. Якщо ви хочете підтримувати кількість завантажень додатків, вам, ймовірно, доведеться найняти власних iOS-розробників".

Make It In Ukraine - рекрутингова агенція №1 в Україні, яка залучає першокласних спеціалістів у сфері технологій, дизайну та маркетингу, а також найкращі віддалені вакансії з усього світу. Компанія також публікує список технологій для мобільної розробки. Найпопулярніші з них у рейтингу наступні:

1. react native
2. Fluttering
3. Xamarin
4. Ionic
5. PhoneGap

Statista - німецька статистична компанія, що спеціалізується на ринкових та споживчих даних. За даними компанії, її платформа містить понад 1 000 000 статистичних даних на більш ніж 80 000 тем з більш ніж 22 500 джерел і 170 різних галузей. Статистикою користуються клієнти, студенти та дослідники з усіх галузей. Статистичний портал об'єднує на одній платформі дані та факти на тисячі різних тем з широкого кола джерел. Джерела включають

маркетингові дослідження, галузеві публікації, наукові журнали та урядові бази даних.

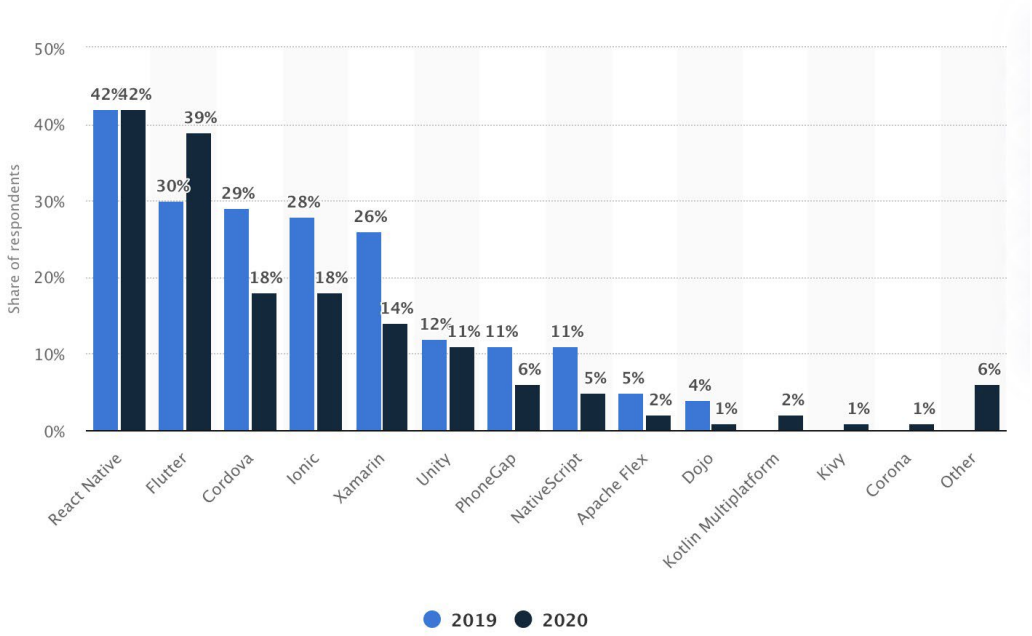


Рисунок 1.3 – Статистика компанії Statista

Як видно з графіка, React Native залишається найпопулярнішим, але порівняно з 2020 роком його обганяє Flutter. Рік від року розробники приділяють дедалі менше уваги Cordova, Ionic та іншим.

Висновки по розділу 1

У цьому розділі подано аналіз та огляд наявних засобів розробки програмного забезпечення. Представлено огляд офіційних інструментів для розробки мобільних додатків. Аналіз показує, що розробка додатків для платформи IOS може здійснюватися різними способами з використанням Swift, JavaScript, Dart та інших мов програмування. Більшість розглянутих інструментів програмування поширюються безкоштовно, але деякі вимагають ліцензії.

Тому можна зробити висновок, що інструменти розробки для платформи IOS і цієї операційної системи стануть дуже популярними в найближчому майбутньому.

У результаті було описано найбільш використовувані сучасні засоби

розробки мобільних додатків. Аналіз показує, що кількість інструментів для розробки програмного забезпечення стрімко зростає, а можливості розробки мобільних додатків стають дедалі доступнішими. За допомогою цих інструментів можна створювати високоякісні додатки високого рівня за відносно короткий час. Крім того, засоби розробки програмного забезпечення, що розглядаються, дають змогу створювати автономні програмні продукти, що встановлюються на операційну систему необхідного комп'ютера. Найбільш популярними кросплатформеними інструментами розробки є React Native і Flutter. Swift, з іншого боку, є лідером серед нативних засобів розробки для операційних систем IOS.

РОЗДІЛ 2. РОЗРОБКА ПЛАНУ ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ ЗАСОБІВ РОЗРОБКИ ДОДАТКІВ

2.1. Основні складові смартфону які впливають на продуктивність програмного забезпечення мобільних пристроїв.

Сучасні смартфони складаються з тисяч компонентів. Кожен з них відіграє свою роль і є надзвичайно важливим аж до найдрібніших деталей. Однак не всі ці компоненти мають відношення до швидкості роботи смартфона.

Продуктивність - це швидкість реакції системи на зовнішні впливи або кількість операцій, що виконуються системою за одиницю часу.

Процесор і оперативна пам'ять мають значний вплив на швидкість роботи смартфона.

Швидкість роботи смартфона сильно залежить від тактової частоти процесора, яка зазвичай вимірюється в мегагерцах (МГц). Коли на кристал кварцу подається напруга, генерується змінний струм, частота якого визначається формою і розміром кристала. Частота цього змінного струму називається тактовою частотою. Типовий чіп смартфона працює на частоті в кілька мільйонів герц. Одиниця швидкості - мегагерц, тобто мільйони циклів за секунду.

Найменшою одиницею часу для процесора як логічного пристрою є такт Кожна операція вимагає щонайменше одного такту.

Основні характеристики процесорів.

- Тактова частота,
- Швидкість передачі даних
- Робоча напруга,
- коефіцієнт множення внутрішньої тактової частоти, обсяг кешу.

Тактова частота визначає кількість основних операцій, які процесор може виконати за одиницю часу. Тактова частота сучасних процесорів

вимірюється в МГц (1 Гц відповідає одній секунді роботи, 1 МГц = 106 Гц). Чим вища тактова частота, тим більше операцій може бути виконано, а отже, тим вища обчислювальна потужність.

Розрядність процесора показує, скільки біт результатів процесор може отримати за цикл і обробити в своїх регістрах. Розрядність процесора визначається розрядністю шини інструкцій (кількістю проводів на шині, що несуть інструкції). Сучасні процесори Intel мають розрядність 32 біта.

Робоча напруга подається на процесор через материнську плату, тому відповідна материнська плата залежить від марки процесора. В даний час робоча напруга процесора ніколи не перевищує 3 В. Зниження робочої напруги не тільки зменшує розмір процесора, але й зменшує тепловиділення, що призводить до кращої продуктивності без ризику перегріву.

Внутрішній коефіцієнт множення тактової частоти - це коефіцієнт, на який необхідно помножити тактову частоту материнської плати, щоб досягти частоти процесора: Процесор отримує тактовий сигнал від материнської плати, але з чисто фізичних причин материнська плата не може працювати на тій же високій частоті, що і процесор. На сьогоднішній день тактова частота материнської плати є фактором, який необхідно помножити на частоту процесора. Сьогодні тактова частота материнської плати є фактором, який необхідно помножити для отримання необхідної частоти. Тактова частота сучасних материнських плат коливається від 100 до 133 МГц. Для отримання більш високих частот процесори можуть бути внутрішньо помножені на 4х, 4,5х, 5х або вище.

Кеш-пам'ять. Обмін даними всередині процесора відбувається набагато швидше, ніж обмін даними з оперативною пам'яттю. Тому процесори оснащені кеш-пам'яттю, щоб зменшити кількість звернень до оперативної пам'яті. Коли процесору потрібні дані, він спочатку шукає їх у кеші, і якщо потрібний результат не знайдено, він звертається до оперативної пам'яті. Чим більший кеш, тим вища ймовірність того, що потрібні дані знаходяться саме там. Саме тому високопродуктивні процесори мають великий кеш. Кеш-

пам'ять поділяється на кеш-пам'ять першого рівня (працює на тій же мікросхемі, що і процесор, ємністю в десятки Кбайт), кеш-пам'ять другого рівня (працює на окремій мікросхемі всередині процесора, ємністю 100 Кбайт і більше), ємністю Кбайт), кеш-пам'ять третього рівня (працює на окремій високошвидкісній мікросхемі на материнській платі, ємністю 100 Кбайт і більше).

Оперативна пам'ять, необхідна для роботи системи в реальному часі, наприклад, браузерів, поштових клієнтів, офісних програм та ігор; це друга за швидкістю оперативна пам'ять в системі: вона розташована одразу після кеш-пам'яті процесора і її ще називають оперативною пам'яттю. Під час роботи оперативна пам'ять постійно споживає енергію, а при вимкненні смартфона всі дані видаляються; оперативна пам'ять містить всю інформацію, до якої потрібен доступ центральному процесору смартфона. Хоча користувачі недооцінюють важливість оперативної пам'яті, вона є найважливішим компонентом смартфона: Розмір і частота оперативної пам'яті визначає загальну продуктивність системи, роботу без збоїв і рівень багатозадачності.

Ключові характеристики оперативної пам'яті.

- Об'єм оперативної пам'яті
- Частота
- Напруга

Об'єм оперативної пам'яті є одним з основних показників і виражається в гігабайтах/мегабайтах.

Він виражається в гігабайтах/мегабайтах. Тут діє просте емпіричне правило: більше - не завжди краще, оскільки все залежить від того, скільки результатів може зберігати оперативна пам'ять.

Частота: відображається в МГц і відображає пропускну здатність модульного каналу. Що вища частота, то швидше обробляється інформація і передається на материнську плату, а потім на процесор або накопичувач.

Напруга: як уже говорилося вище, оперативна пам'ять залежить від напруги. Напруга вказується в специфікаціях. Вона вказує мінімальну

напругу, необхідну для стабільної роботи модуля за базових налаштувань часу і частоти.

У смартфонах оперативна і довготривала пам'ять розділені. При цьому пристрої оперативної пам'яті зазвичай використовуються як в якості робочої пам'яті для додатків, так і в якості середньо- і довгострокової пам'яті. Флеш-пам'ять, зазвичай у вигляді знімних карт пам'яті, дедалі частіше використовується для довготривалого зберігання результатів.

AJAX - це технологія, яка дає змогу веб-додаткам робити запити до сервера й отримувати відповіді без перезавантаження всієї сторінки. Це робить роботу користувача більш плавною та ефективною.

Для реалізації AJAX-запитів використовується бібліотека axios. axios - це бібліотека, заснована на клієнтському HTTP для браузерів і Node.js. Вона надає простий у використанні інтерфейс і містить такі корисні функції, як перехоплювачі запитів і відповідей, що використовуються в проєкті.

Нативні сервіси взаємодії допомагають керувати цими запитами, використовуючи специфічні для застосунку методи та налаштування. Це дає змогу організувати їх належним чином і гнучко адаптуватися до різних вимог і потреб застосунку.

Важливим аспектом реалізації є використання перехоплювачів axios. Перехоплювачі можуть використовуватися для втручання в процес опрацювання запиту або відповіді до того, як його буде опрацьовано, або для усунення помилок до того, як їх буде відхилено. Платформа використовує перехоплювачі для додавання маркерів аутентифікації до запитів аутентифікованих користувачів. Це забезпечує безперервну і безшовну аутентифікацію на всіх рівнях взаємодії між клієнтом і сервером.

Важливо зазначити, що використання перехоплювачів дає змогу реалізувати систему автентифікації, яка є не тільки ефективною, а й надійною. Запити, надіслані авторизованим користувачем, автоматично включають у себе необхідні токени, гарантуючи, що запит буде правильно оброблено сервером. Такий підхід дуже зручний для додатків, що вимагають

постійної взаємодії між клієнтом і сервером.

Такий підхід до AJAX-запитів і використання axios також полегшує обробку помилок. Використовуючи перехоплювачі, платформа може перехоплювати помилки до того, як вони потраплять в основний код програми, і реагувати на них відповідним чином. Це включає в себе збереження помилок для подальшого аналізу та надання корисного зворотного зв'язку користувачам програми.

Розробка серверної частини платформи для створення сайтів є невід'ємною і важливою частиною цього проекту. Одним із важливих аспектів цього етапу є реалізація аутентифікації та авторизації користувачів. Це дає змогу перевіряти особу користувачів і надавати їм відповідні права доступу.

Аутентифікація та авторизація є ключовими компонентами безпеки будь-якого веб-додатку. Аутентифікація визначає, хто використовує систему. Ці два поняття взаємопов'язані і разом створюють безпечну систему, що забезпечує контроль доступу.

Для реалізації цих механізмів використовується підхід на основі токенів JWT: JWT - це відкритий стандарт, що визначає спосіб безпечної передачі інформації у вигляді JSON-об'єктів між двома сторонами. Ця інформація підписується і тому може бути перевірена і довірена; JWT використовується для автентифікації користувачів і може містити інформацію про користувача, необхідну для автентифікації [36].

У контексті Nest.js аутентифікацію та авторизацію за допомогою JWT реалізовано за допомогою модуля @nestjs/jwt. Цей модуль надає набір сервісів і декораторів для спрощення роботи з JWT. Для захисту різних маршрутів використовуються так звані "вартові": вартові в Nest.js - це класи, що реалізують метод canActivate, який визначає, чи може запит продовжити виконання за маршрутом. У контексті JWT вартовий може визначити, чи має запит, що токен присутній і його валідність може бути перевірена. Якщо токен присутній і дійсний, Guardian дозволяє запиту продовжити виконання

за маршрутом.

Одним із самореалізованих Guard Guard є JwtAuthGuard, який автоматично перевіряє наявність JWT-токена в заголовку авторизації запиту. Якщо токен відсутній або недійсний, запит скасовується.

Усі ці механізми забезпечують надійну авторизацію та автентифікацію користувачів на платформі для створення веб-сторінок. Це дає змогу користувачам впевнено користуватися нашими послугами.

Взаємодія з базою даних також важлива. У цьому проєкті ми використовували TypeORM, один із найпопулярніших ORM-інструментів для TypeScript і JavaScript. TypeORM забезпечує абстракцію на рівні даних і дає змогу розробникам використовувати класи TypeScript як моделі. Він дає змогу розробникам використовувати класи TypeScript як моделі.

У контексті Nest.js, TypeORM є природним вибором, оскільки Nest.js також використовує TypeScript як основну мову. Це означає, що всі переваги TypeScript, як-от статична типізація та інтелектуальне завершення коду, можуть бути використані під час роботи з базами даних.

TypeORM також пропонує безліч функцій, як-от транзакції, міграції, вкладені селекти та відносини між сутностями. Це полегшує роботу з базами даних і дає змогу зосередитися на бізнес-логіці програми, а не на деталях реалізації бази даних.

Використання TypeORM у Nest.js має перевагу інтеграції з фреймворком; Nest.js надає модуль TypeORM для полегшення налаштування та використання TypeORM у проекті Nest.js. Nest.js автоматично створює однотонний об'єкт з'єднання TypeORM і може використовувати його як об'єкт з'єднання. тон об'єкт з'єднання TypeORM, який можна використовувати в усьому додатку. Модуль також надає декоратор для легкого вбудовування репозиторіїв TypeORM у сервіси.

Однак, мабуть, найбільшою перевагою використання TypeORM є те, що він дає змогу працювати на вищому рівні абстракції, фокусуючись на об'єктах у коді, а не на SQL-запитах. Це полегшує розуміння і розробку

додатків.

Реалізація процесу публікації сайтів, створених користувачами платформи, була дуже важливим завданням у цьому проєкті. Цей процес мав забезпечити можливість розміщення створених ресурсів у глобальній мережі Інтернет. Для розв'язання цього завдання було орендовано веб-сервер і домен для розміщення самої платформи та всіх веб-сторінок, створених користувачами платформи.

Для забезпечення коректної роботи системи редагування було ухвалено рішення використовувати веб-сервер nginx - високопродуктивний веб-сервер із відкритим вихідним кодом, який використовують для обслуговування веб-сторінок. Він відомий своєю стабільністю, багатим функціоналом, простотою налаштування і низькою вимогливістю до ресурсів [7].

Після встановлення та налаштування на орендованому веб-сервері nginx стежить за папкою 'sites', куди додаються HTML-файли після публікації користувачами своїх сайтів. Це автоматизує процес публікації та забезпечує швидкий доступ до опублікованих сайтів.

Крім того, nginx відповідає за створення сайтів на піддоменах, зазначених користувачами платформи. Це дає змогу користувачам розміщувати свої сайти на власних піддоменах, підвищуючи видимість своїх проектів в інтернеті.

Правильне налаштування та оптимізація nginx не тільки підвищила продуктивність і надійність платформи, а й забезпечила швидке та безперебійне розгортання користувацьких сайтів. Це надзвичайно важливо. Доступність і швидкість завантаження будь-якого веб-сайту - один із ключових чинників успіху, оскільки це безпосередньо впливає на користувацький досвід.

Загалом впровадження nginx у проєкт значно підвищило ефективність розгортання користувацьких сайтів і дало змогу створити масштабовану, гнучку та надійну інфраструктуру для платформи створення веб-сторінок.

2.2. Перший етап проектування. Опис бази даних

Перший етап проектування системи - визначення необхідних цілей. Основна мета - створити систему, яка б відстежувала проєкти і завдання команд розробників з різним набором навичок, з можливістю коментувати завдання для отримання додаткових пояснень, відстежувати час, витрачений на виконання завдань, призначати користувачів на ролі для обмеження доступу, а також можливість призначати користувачів на виконання завдань відповідно до їхніх навичок. Наступний крок - розробка бази даних для цієї системи. Для цієї мети було визначено таку структуру бази даних

а) "користувачі" - ця таблиця містить дані про користувачів. Ця таблиця містить

містить такі поля:

1) "id" - номер користувача. Це поле є первинним ключем;

2) "login" - унікальний ідентифікатор користувача в системі. Це поле є унікальним ключем;

3) "password" - хеш пароля користувача;

4) "email" - адреса електронної пошти користувача; 4) "email" - адреса електронної пошти користувача;

5) "ім'я" - ім'я користувача; 6) "роль" - роль користувача; 7) "роль" - роль користувача

6) "role" - роль користувача. Можливі ролі: адміністратор, менеджер і користувач;

7) "user_pic" - аватар користувача; 8) "user_pic" - аватар користувача

8) "active" - це поле стосується активності користувача; якщо значення дорівнює 1, то користувач може увійти в систему; якщо значення дорівнює 0, то користувач не може увійти в систему;

б) "проекти" - у цій таблиці зберігаються наявні проекти;

1) "id1" - це поле є первинним ключем

2) "name1" - ім'я проекту користувача;

3) "creation_date1" - дата створення проєкту; 4) "status1" - статус проєкту

4) "status1" - поточний статус проєкту; 4) "status1" - поточний статус проєкту

5) "description1" - опис проєкту; 6) "user_id1" - ім'я проєкту користувача

6) "user_id1" - номер користувача, відповідального за проєкт;

в) "sessions1" - ця таблиця містить хеш токенів сесій користувачів, необхідних для авторизації в системі. Поки запис існує, відповідний користувач може увійти в систему. Якщо користувач виходить із системи, запис видаляється;

1) "user_id1" - номер користувача;

2) "user_token1" - хеш-токен користувача; 3) "user_token2" - хеш-токен користувача

3) "remember_me1" - чи була востаннє натиснута кнопка "remember me1". Прапор "remember_me1" перевірявся під час останньої аутентифікації;

4) "creation_time1" - час аутентифікації;

г) "skills1" - у цій таблиці перераховані навички, які існують для того чи іншого користувача.

У ній перераховані навички, що існують для

1) "user_id1" - номер користувача;

2) "skill1" - назва навички; 3) "skill2" - назва навички; 4) "skill2" - назва навички

д) д) "tasks1" - у цій таблиці містяться завдання проєкту;

1) "id1" - це поле є первинним ключем

2) "name1" - назва завдання; д) д) д) д) д) д)

3) "description1" - опис завдання; 4) "project_id1" - це поле є первинним ключем

4) "project_id1" - номер проєкту, до якого належить завдання; 4) "project_id1" - номер проєкту, до якого належить завдання

5) "user_id1" - номер користувача, який виконує це завдання; 5) "user_id1" - номер користувача, який виконує це завдання

6) "creation_date1" - дата створення завдання; 7) "deadline_date1" - крайній термін виконання завдання

7) "deadline_date1" - крайній термін виконання завдання; 8) "status1" - статус завдання; 9) "status2" - статус завдання

8) "status1" - поточний статус завдання; 9) "status2" - поточний статус завдання; 10) "status2" - поточний статус завдання

е) "worklog1" - таблиця, що зберігає дані про виконання завдань користувачем.

У ній зберігається така інформація;

1) "id1" - це поле є первинним ключем

2) "task_id1" - номер виконаного завдання; е) "task_id1" - номер виконаного завдання

3) "user_id1" - номер користувача, який виконав завдання; 4) "user_id1" - номер користувача, який виконав завдання

4) "description1" - опис діяльності, залишений виконавцем; 5) "start_time1" - час початку виконання завдання

5) "start_time1" - час початку виконання завдання; 6) "end_time" - час зупинки виконання завдання; 7) "total_time" - час, витрачений протягом сесії..

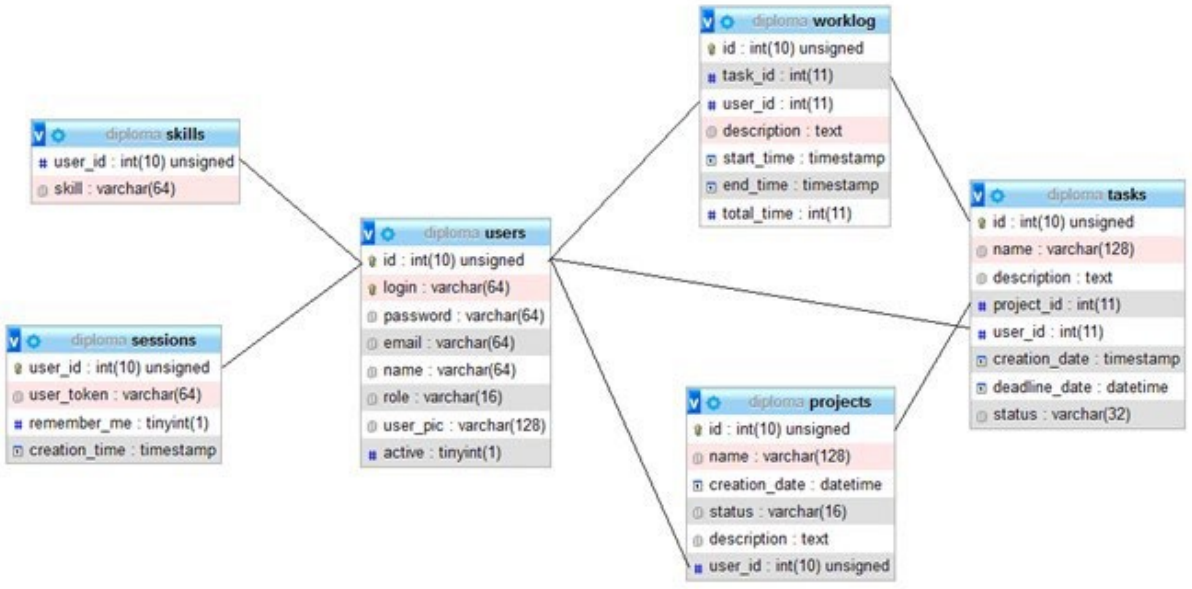


Рисунок 2.1 Структурна схема бази даних

На рисунку 2.1 знаходиться графічне зображення структурної схеми зв`язків таблиць у базі даних.

2.3 Другий етап проектування. Алгоритми роботи системи

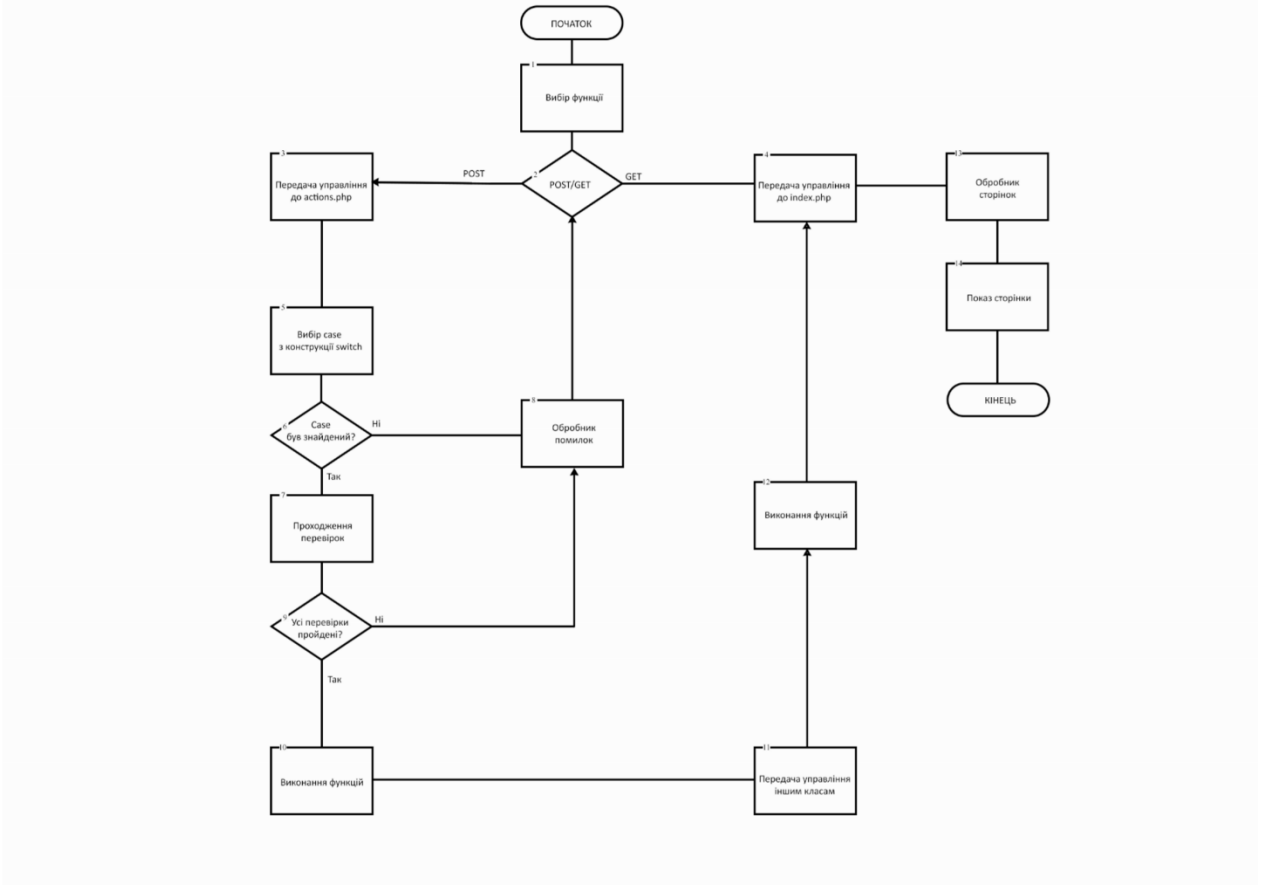
Система розділена на класи PHP, файли шаблонів tpl, файли стилів CSS і скрипти json. Усі функції діляться на два типи: функції для GET-запитів і функції для POST-запитів. Коли сервер отримує POST-запит, він перенаправляє потік даних у файл action.php. Якщо сервер отримує GET-запит, він перенаправляє потік даних у файл index.php.

У системі реалізовано розподіл користувачів за ролями. Ролі користувачів надають доступ до системних функцій для виконання завдань. Роль адміністратора надає доступ до тих самих функцій, що й користувач, але також керує проєктами та завданнями і призначає користувачів на проєкти та завдання. Ролі адміністраторів надають доступ до адміністративних функцій і керують користувачами, наприклад редагують, видаляють, додають і деактивують користувачів. Оскільки ця функціональність відіграє важливу

роль в обмеженні доступу до системи, доступ до функцій ролі є авторизованим і не може бути змінений без модифікації коду. Код також необхідно змінити, якщо потрібно створити нові ролі. Коли систему продають компаніям, заздалегідь визначено, що вони можуть вільно модифікувати та змінювати систему. Однак слід зазначити, що навіть без модифікації або зміни коду систему вже завершено, вона має всю функціональність, необхідну для оптимального функціонування підприємства.

Рисунок 2.2 Алгоритм роботи системи

На рисунку 2.2 показано основний алгоритм, який використовує система. Коли користувач натискає будь-яку кнопку, система починає



працювати відповідно до цього алгоритму. Спочатку функція, що викликається, перевіряє тип запиту. Якщо це звичайний GET-запит, управління передається класу index.php, де контролер сторінки отримує дані, надані користувачем, і відображає обрану сторінку. Якщо функція заснована на POST-запиті, управління передається класу action.php. Контролер action.php

шукає в структурі switch потрібний випадок. Якщо його не знайдено, обробка даних передається в index.php, який виводить повідомлення про помилку або сторінку 404 на головній сторінці. Якщо випадок знайдено, дані користувача перевіряються. Якщо перевірка не пройдена, управління даними передається index.php, а на головній сторінці відображається повідомлення про помилку. Після проходження валідації управління передається відповідному класу, який виконує необхідні функції, зберігає всі нові дані і зберігає всі зміни в даних. Потім управління передається назад в index.php, де відображаються результати роботи потрібної сторінки або функції. Винятком із цього алгоритму є функції, які не потребують перезавантаження сторінки (ajax-запити).

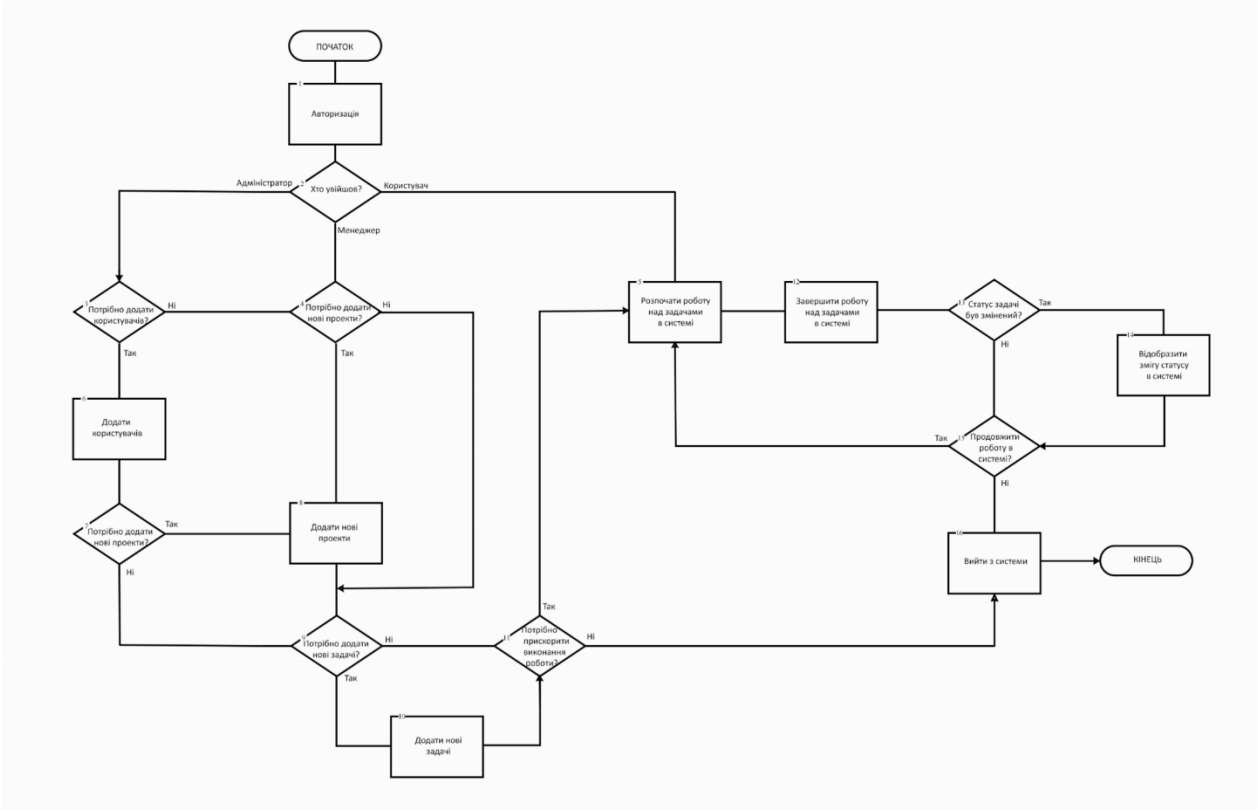


Рисунок 2.3 Алгоритм роботи в системі

На рисунку 2.3 показано алгоритм, якого дотримуються працівники залежно від їхньої ролі в системі. Винятками з цього алгоритму є редагування даних (як користувачів, так і проектів і завдань), а також деактивація або активація користувачів. Це пов'язано з тим, що ці функції не є необхідними для роботи з системою, але важливі для передбачуваного втручання.

Наприклад, відключення користувача - це можливість дозволити співробітнику. Крім того, редагування не є основною функцією, тому що вона призначена для виправлення помилок і оновлення інформації про користувачів, проекти і завдання.

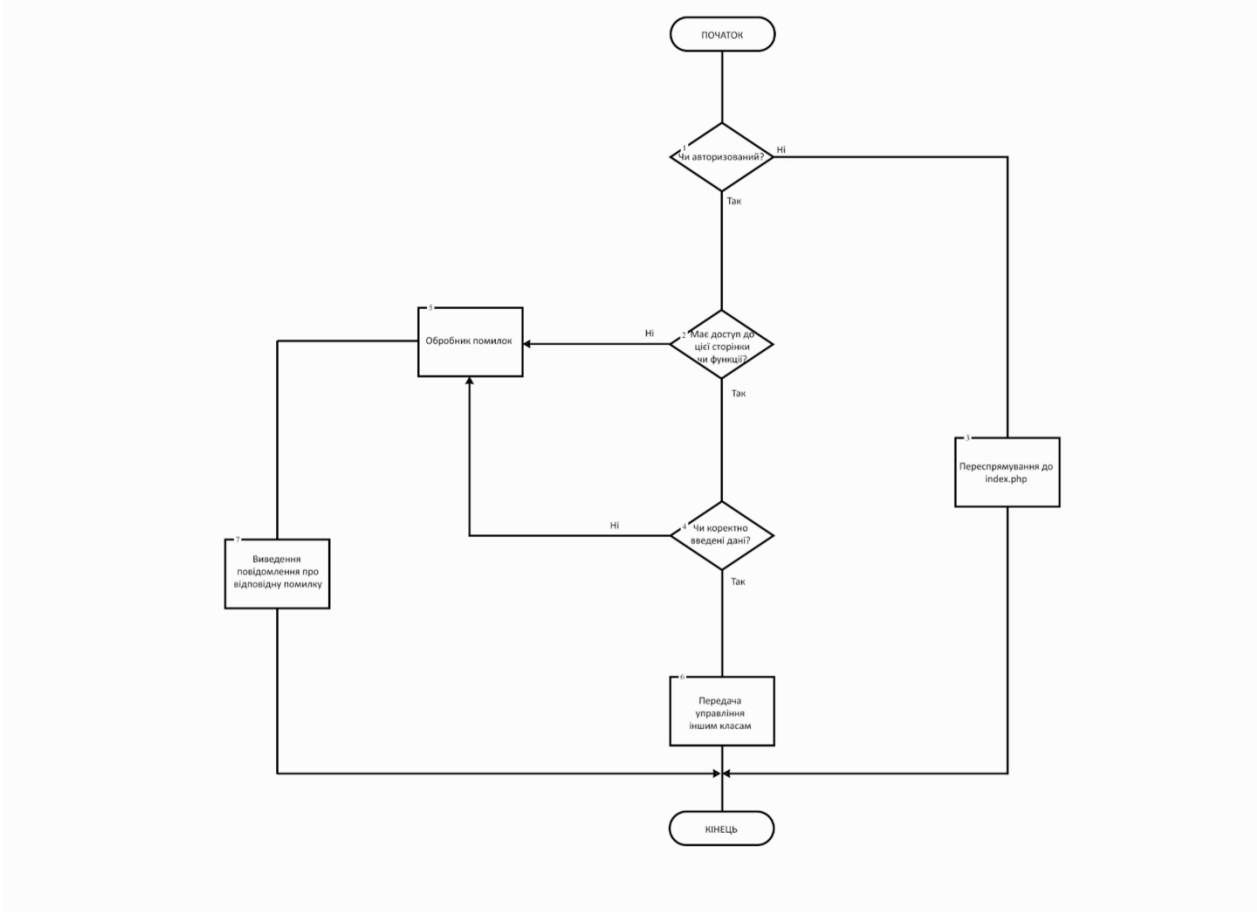


Рисунок 2.4 Основний алгоритм перевірки даних

На рисунку 2.4 показано алгоритм перевірки даних, який використовується контролером actions.php під час отримання даних від користувача. У розділі "Чи правильно введено дані" виконується кілька перевірок. Наприклад, чи існує певний ключ у переданому масиві даних і чи є вміст ключа в масиві даних числовим.

Висновки до розділу 2

У цьому розділі аналізуються новітні мобільні додатки та їхнє використання. Було розглянуто кілька статистичних джерел для виявлення найбільш популярних, якими виявилися React Native і Flutter для

кросплатформенного програмування і Swift для нативного програмування під IOS. IOS було обрано як основну платформу для тестування та експериментів.

У цьому розділі також було розглянуто ключові компоненти сучасних смартфонів, що впливають на їхню продуктивність та енергоспоживання.

Заключним кроком у цьому розділі є вибір і розгляд алгоритмів, які будуть використовуватися для тестування та експериментів.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ АЛГОРИТМІВ ЗА ДОПОМОГОЮ НАЙПОПУЛЯРНІШИХ ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ РОЗРОБКИ

3.1. Організація розробки проекту

Почніть зі створення React Native додатку. Для створення проекту використовуйте стандартний конструктор проектів "create-react-native-app", наданий Facebook. Спочатку встановіть білдер на свій комп'ютер командою "npm install -g react-native-cli". Введіть команду терміналу на комп'ютері, щоб створити проект. Далі вам буде запропоновано ввести назву проекту:

"What is the name of your app, React Native Project".

Таким чином буде створено базовий проект для подальшої розробки додатку. Розглянемо базову структуру React Native проекту, яка показана на рисунку 3.1.

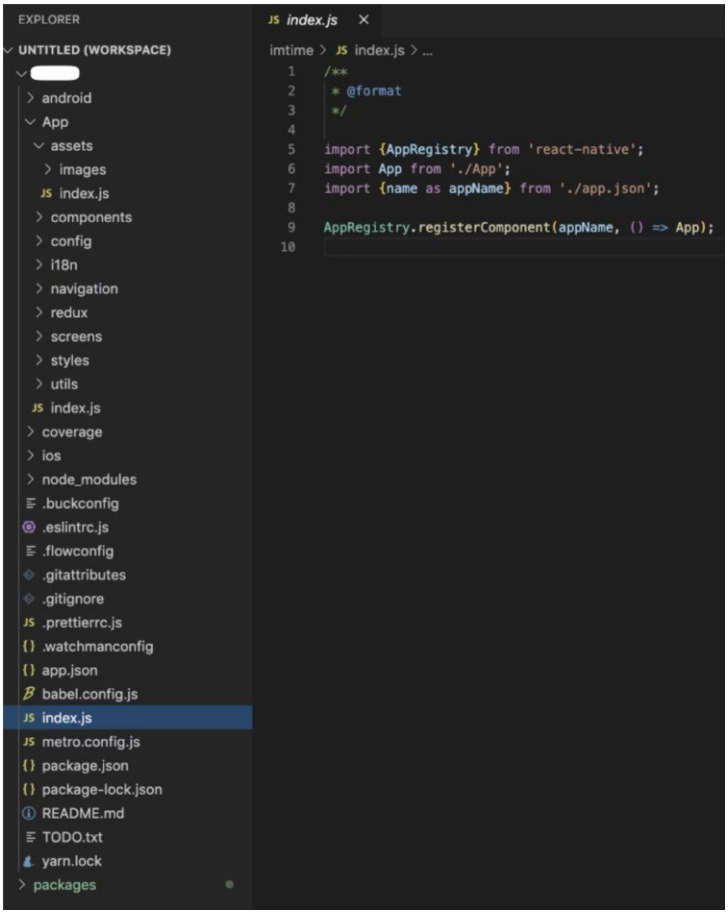


Рисунок 3.1 - Структура React Native проекту

Проект React Native складається з базового набору файлів, кожен з яких має певне призначення. Давайте розглянемо кожен з цих файлів нижче:

1. /index.js1 - це точка входу за замовчуванням для всіх React Native додатків. У цьому випадку цей файл не слід змінювати.

2. /package.json1 - містить інформацію про додаток,

Містить інформацію про додаток, версію та основні залежності.

3. /App1/... - Папка, що містить всі файли, які будуть написані для додатку.

4. /App1/index.js - цей файл є контейнером і точкою входу для додатка.

5. /App1/assets/ - тут розміщуються всі статичні активи. Кожен актив має бути зареєстрований і експортований з /index.js. Тому всі активи будуть доступні та імпортовані з "/assets".

6. App1/components/ - папка, що містить усі візуальні компоненти, з яких складається додаток.

7. App1/config - папка, в якій зберігаються всі файли постійної конфігурації.

8. App1/i18n - папка, в якій знаходяться файли локалізації.

9. App1/navigation - папка, що містить файли, необхідні для навігації по додатку.

10. App1/redux - містить дії, редуктори та саги і є реалізацією патерну Flux у react.

11. App1/screens - сторінки додатка.

12. App1/services - містить усі API-запити до сервера. app1/styles - містить стилі на рівні застосунку. Містить теми інтерфейсу додатка (шрифти, кольори, типографіка) і глобальні визначення стилів.

Щоб запустити додаток на емуляторі або мобільному пристрої, відкрийте термінал і введіть команди react-native start і react-native run-ios.

Щоб створити проект Flutter, виконайте в терміналі команду "flutter create project name". Як і в попередньому випадку, система створить базовий проект, який можна використовувати для подальшої роботи. Розглянемо

структуру проекту Flutter, зображену на рисунку 3.2.

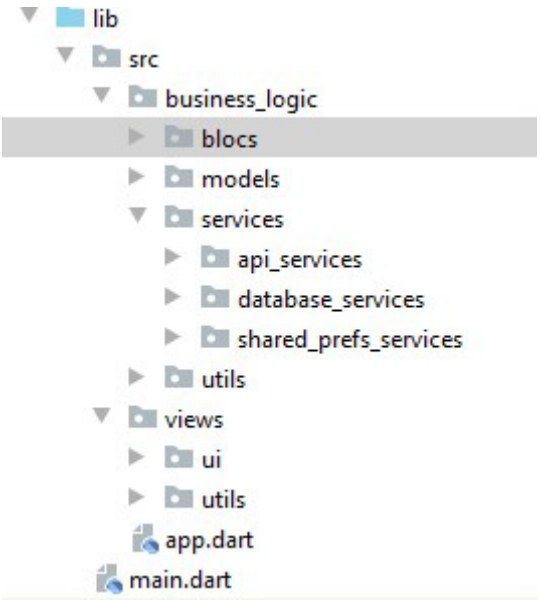


Рисунок 3.2 – Базова структура проекту на Flutter

Як і його попередник React Native, Flutter має фреймворк, який відображає найкращі практики розробників, які обирають цей інструмент. Давайте розглянемо його зараз.

1. /lib1 - весь код знаходиться в цій папці.

2. /src1 - пакет src є основним пакетом для проекту. Таких пакетів може бути декілька, але для нашої задачі цього достатньо.

3. /main.dart1 - точка входу для Dart. Основна роль цього файлу - перенаправлення на /src/app.dart.

4. /src/app.dart1 - головний файл пакета. Збирає та ініціалізує глобальні залежності для подальшої роботи.

5) /src/business_logic1 - назва папки вказує на те, що вона містить основну бізнес-логіку додатку.

6) /src/business_logic/block1s - якщо використовується модель BLoC, всі блоки знаходяться тут.

7) /src/business1_logic/models - тут розміщуються всі моделі, що відображаються в додатку.

8. /src/business_logic1/services - тут зібрані всі api запити до сервера.

9. /src/business_logic1/utils - в цій теці знаходяться всі константи,

конфігураційні файли та утилітні модулі.

10. /views1 - в цій теці знаходяться всі файли, що відповідають за візуальне представлення додатку.

11. /views1/ui - містить весь необхідний код для візуального представлення сторінки.

12. /views1s/utisl - містить всі допоміжні функції, необхідні для візуального представлення.

Щоб запустити проєкт Flutter на смартфоні, підключеному до ПК, або на емуляторі, встановленому на ПК, Google пропонує команду "flutter run". flutter run -d {device id}", Ви також можете вибрати власне місце розташування для запуску програми. Ця команда допоможе вам запустити додаток на пристрої із зазначеним ідентифікатором.

Після розгляду структури Flutter давайте розглянемо Swift, єдиний нативний засіб розробки, що надається Apple Apple розробила чудове середовище розробки під назвою Xcode, і всі операції, пов'язані з проектами Swift, виконуються через це середовище. swift Щоб створити проект, необхідно відкрити середовище розробки Xcode. Це один зі стандартних додатків, встановлених на ноутбуках Apple. Його також можна завантажити з Apple Store. Коли ви відкриваєте Xcode, з'являється вікно привітання Apple (рис. 3.3). Тут можна почати проект, над яким ви вже працювали. Також є низка нових проєктів, які можна створити або доопрацювати за одну ніч у так званому "ігровому майданчику" - місці, де розробники можуть швидко спробувати нові речі, не витрачаючи багато часу на створення та налаштування нового проєкту з нуля.

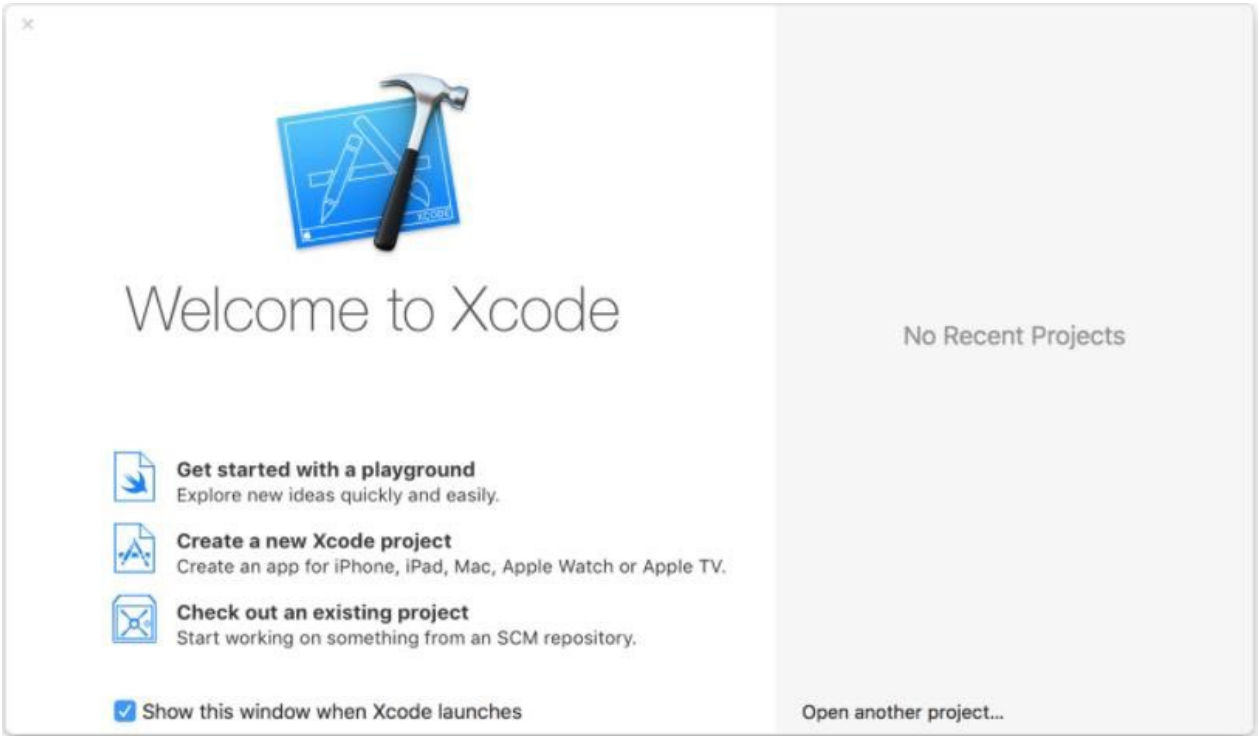


Рисунок 3.3 – Вікно привітання Xcode

Щоб створити новий проєкт, виберіть "Створити новий проєкт Xcode". Після цього ви потрапите в інтерфейс створення проекту (рис. 3.4). У цьому вікні виберіть тип проєкту, який ви хочете створити - Xcode надає кілька готових шаблонів:

1. Додаток з одним видом - це тип додатка, який складається тільки з однієї сторінки. Тут неможливо створити навігацію між сторінками.

2. game1 - шаблон для створення ігор. Його функція полягає в підборі активів, які можуть знадобитися під час створення гри. Потім його налаштовують для оптимізації складання програми.

3. page Based Application1 - найпоширеніший тип проєкту. Тут можна створити безліч сторінок із різним вмістом і налаштувати навігацію між сторінками. У нашому застосунку немає потреби додавати навігацію між сторінками, тому достатньо створити застосунок з одним видом.

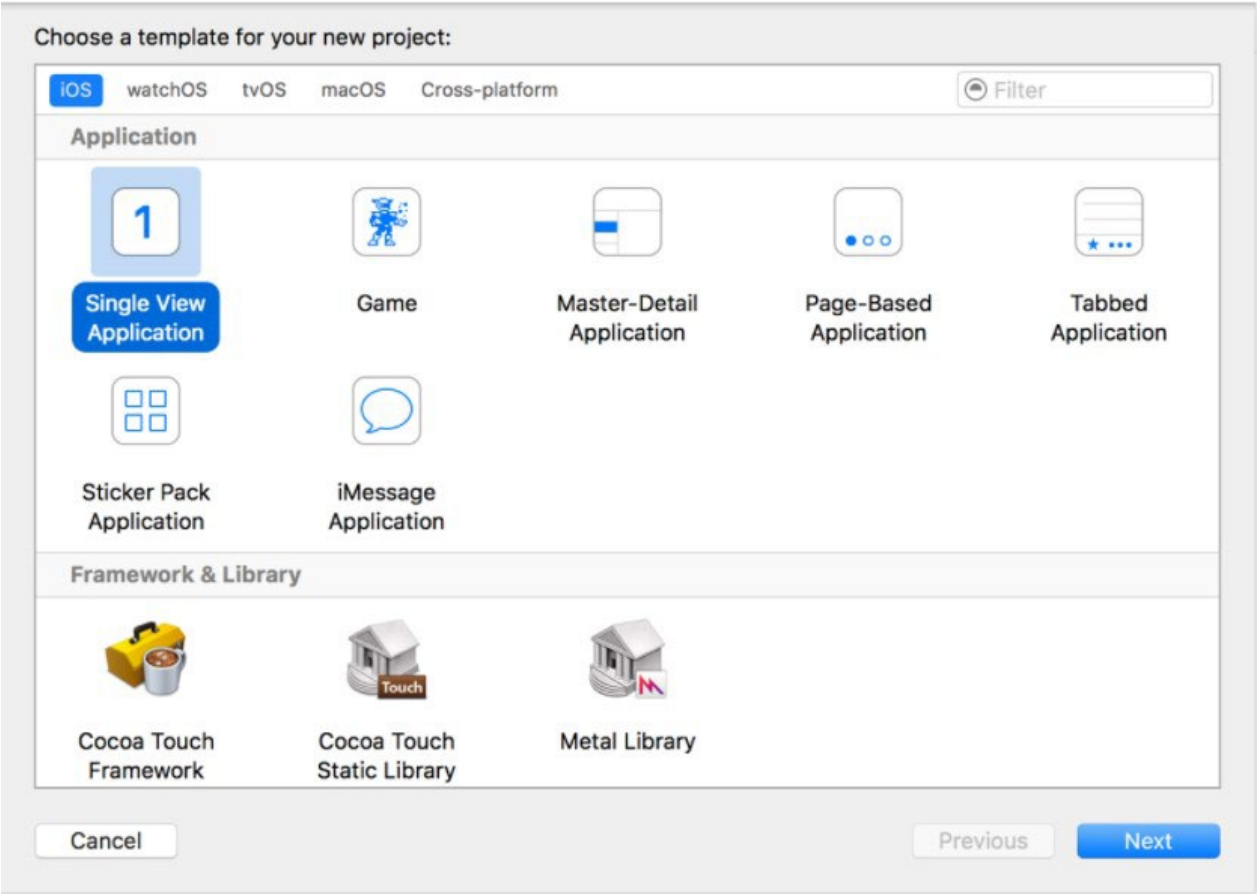


Рисунок 3.4 – Інтерфейс створення проекту Xcode

Далі перейдіть на сторінку налаштувань проекту (рис. 3.5). Тут ви можете налаштувати параметри проекту; Xcode дає змогу задати такі параметри

Ви можете задати такі параметри:

1. product name1 - назва проєкту;

2. team1 - команда, що розробляє проєкт; це необхідно для підпису проєкту під час випуску;

3. назва організації1 - назва організації, що розробляє проєкт;

4. ідентифікатор організації1 - ідентифікатор організації, що має підписати проєкт під час випуску; 4. ідентифікатор організації2 - ідентифікатор організації, що має підписати проєкт під час випуску;

5. мова1 - мова, якою розробляється проєкт. Тут виберіть Swift або Objective-C.

6. devices1 - пристрій (смартфон, планшет або годинник), на якому буде вестися розробка. Можна вибрати універсальний тип, що підходить для всіх

типів пристроїв. Однак у цьому разі слід подбати про розробку інтерфейсу, який можна буде адаптувати під кожен тип пристрою;

7. include unit1 test - створює основу для модульних тестів;

8. include ui1 test - створює основу для UI тестів;.

9. use core datum1 - фреймворк для управління об'єктами рівня моделі додатку; 9. use core datum2 - фреймворк для управління об'єктами рівня моделі додатку;

10. use core datum2 - фреймворк для управління об'єктами рівня моделі додатку;

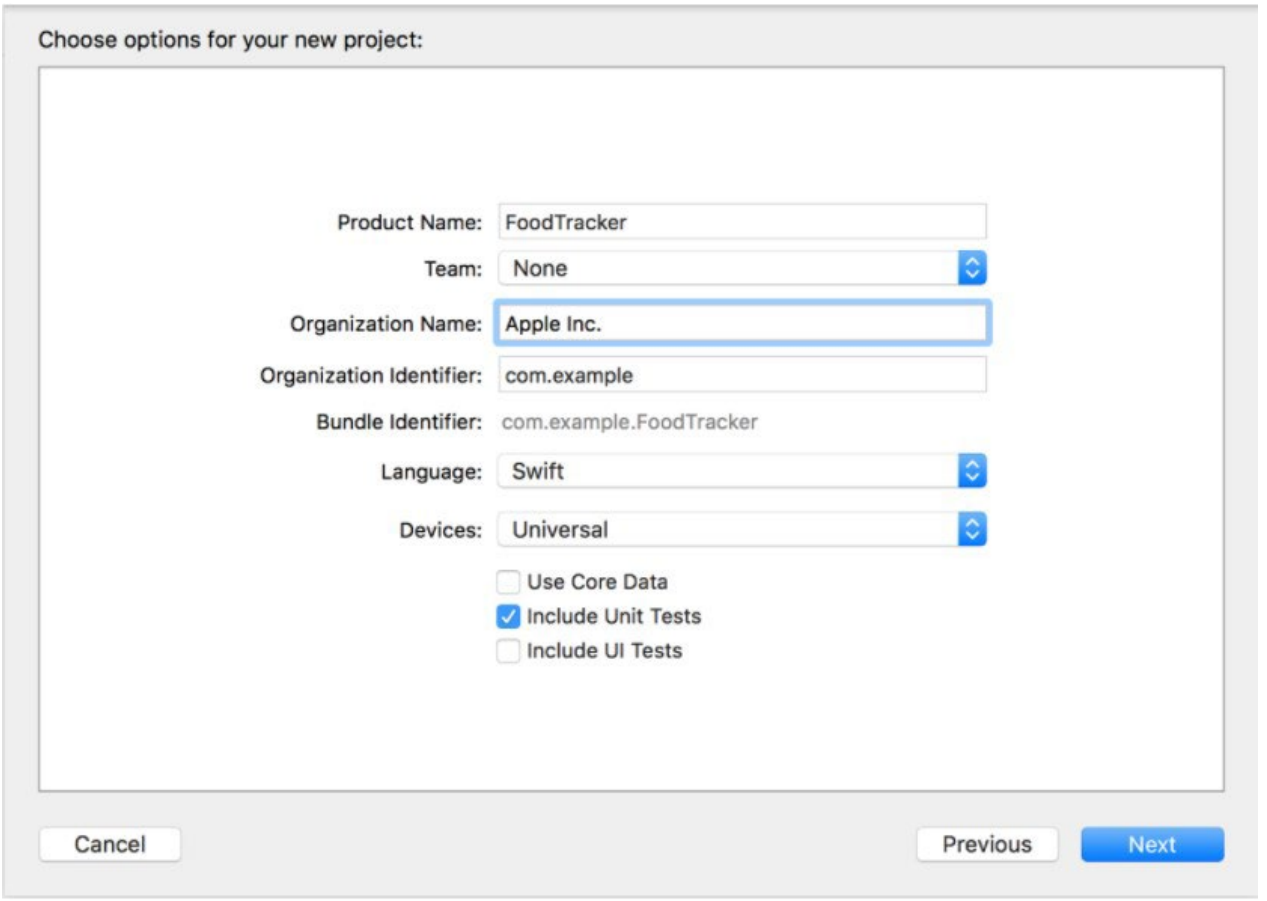


Рисунок 3.5 – Сторінка налаштування проекту Xcode

Це завершальний крок у створенні проекту Xcode. Середовище розробки створює базовий проект, який можна використовувати для подальшої розробки. Як і в попередньому випадку, давайте розглянемо структурування проекту на Swift: Основною моделлю програмування, якої дотримуються розробники Swift, є MVC. Це добре видно в структурі папок проекту.

Рисунок 3.6 – Структура проекту Swift

Я Тут ви можете побачити структуру проєкту, що демонструє використання патерну MVC.

1. /Controllers1 - Ця папка містить компоненти, що відповідають за зв'язок між моделлю та поданням. Код компонента контролера визначає, як додаток реагує на дії користувача;

2./Models1 - компонент, що відповідає за дані, які визначають структуру додатка; 3;

3./Views1 - компонент, що відповідає за взаємодію з користувачем. Інакше кажучи, код компонента виду визначає, який вигляд має і як використовується додаток;

4. /Library1 - місце, де розміщуються додаткові допоміжні файли; 5;

5. /Library1/BaseClasses1 - базові класи, які використовуються в усіх місцях. Це Model, Navigation Controller і View Controller. Можуть бути й інші,

наприклад, контролери колекцій. Усі пов'язані сутності успадковуються від цих класів;

6. /Library1/Helpers1 - всі інші спеціалізовані помічники можна знайти тут. Зазвичай, кожен проект має API, що містить обгортки, такі як AFNetworking. Ці файли розроблені так, щоб бути максимально гнучкими і адаптуватися до вимог будь-якого проекту;

Запустити Swift-проект дуже просто: виберіть емулятор або реальний пристрій у середовищі розробки і натисніть кнопку Run. Статус збірки відображається в області Activity на панелі інструментів.

Якщо збірка пройшла успішно, Xcode запускає додаток і відкриває сеанс налагодження на панелі налагодження. Використовуйте елементи керування на панелі налагодження для переміщення коду, керування змінними та взаємодії з налагоджувачем.

3.2. Розробка ієрархічної моделі мобільного додатку

IOS слідує архітектурному патерну Model-View-Controller1 (MVC) для організації взаємодії між користувацьким інтерфейсом і логікою.

Model-View-Controller1 - це архітектурний патерн, який ділить застосунок на три основні логічні компоненти: модель, подання та контролер. Кожен із цих компонентів призначений для виконання певного аспекту розробки застосунку. MVC є одним із найбільш широко використовуваних галузевих стандартів у веб-розробці для створення масштабованих і розширюваних проєктів.

Патерн розділяє систему на три взаємопов'язані частини: вихідну модель, представлення (користувацький інтерфейс) і модуль управління. Він використовується для відокремлення результатів (моделі) від призначеного для користувача інтерфейсу (представлення), мінімізуючи вплив змін призначеного для користувача інтерфейсу на дані та даючи змогу вносити зміни в модель результатів без зміни призначеного для користувача інтерфейсу. Шаблони MVC можна використовувати для організувати код у

такий спосіб, щоб його частини можна було змінювати, не зачіпаючи решти коду. Це має низку переваг

1. дозволяє писати код в ітеративній та довільній манері

2. полегшує тестування модулів.

3. Більш ефективне використання інструментів проектування. 4. Більш ефективне використання коду.

4. підтримка командної взаємодії.

При використанні шаблонів MVC додатки для IOS можна розділити на наступні частини:

1. інтерфейс, розроблений з використанням технологій Rect Native, Flutter або Swift.

2. Логіка користувацького інтерфейсу реалізована розробником у вигляді компонентів ViewModel.

3. роль контролера полягає у відстеженні певних подій, які відбуваються в результаті дій користувача. Контролери дозволяють структурувати код, групуючи пов'язані дії в окремі класи. Наприклад, типовий MVC-проект може мати контролер користувача, який містить групу методів, пов'язаних з управлінням обліковими записами користувачів, таких як реєстрація користувача, вхід, редагування профілю, зміна пароля. Представлення включають в себе.

Подання - це базовий клас

Інтерфейс програми IOS являє собою дерево екземплярів екземплярів дочірніх класів цього класу.

Клас Experiment і його підкласи містять логіку, що реалізує користувацький інтерфейс. Цей клас відповідає ViewModel в архітектурному шаблоні Model-ViewController1 (MVC); зв'язок між підкласом Experiment і його призначеним для користувача інтерфейсом - один до одного, до того ж кожен підклас Experiment зазвичай має пов'язаний із ним тільки один призначений для користувача інтерфейс і навпаки. У магістерській дисертації мобільний додаток містить трирівневу модель представлення підкласів класу

Experiment (рис. 2.3). Перший рівень - це сторінка завантаження ресурсів. Ця сторінка відображається в момент завантаження застосунку і переходить на наступний рівень після повного завантаження ресурсів застосунку; на другому рівні користувач обирає алгоритм експерименту та кількість ітерацій. Останній рівень - рівень результатів експерименту, де користувач може переглянути результати експерименту та надіслати їх на сервер для подальшого опрацювання.

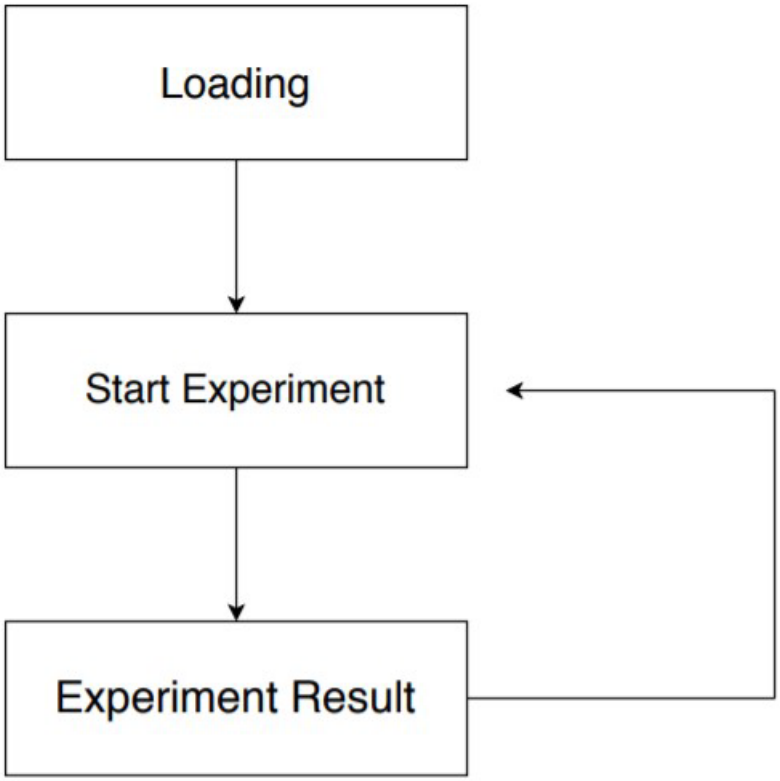


Рисунок 3.6 – Ієрархічна модель мобільного додатку

3.3. Базова архітектура системи

3.3.1 Реалізація логіки програмного продукту з React Native

Під час розробки програми в React Native основним файлом є App.js. Він являє собою точку входу в додаток, визначає глобальні залежності та

ініціалізує їх.

Почніть зі створення маршрутизатора. У веб-браузері, коли користувач натискає на посилання, URL додається до стека історії, а коли користувач натискає кнопку "назад", остання відвідана адреса видаляється зі стека. rn не має цієї концепції за замовчуванням і вимагає наявності залежності react-. залежність від навігаційного пакета. Додаток може мати кілька стеків. Наприклад, кожна вкладка програми може мати свою історію переглядів, але тільки одну.

Створіть папку navigation і створіть у ній RootNavigator.js.

import * as React from 'react';

import {createStackNavigator} from '@react-navigation/stack'; import { ExperimentsScreen} from '../screens/ExperimentsScreen';

konst Stack = createStackNavigator();

export konst RootNavigator = () => {

return (

<Stack.Navigator initialRouteName={'experiments'}>

<Stack.Screen name={'experiments'} component={ExperimentsScreen} />

</Stack.Navigator>

);

};

Далі створіть першу сторінку вашого додатку. Для цього створіть папку screens і файл ExperimentsScreen та додайте код для створення сторінки:

export default class App extends Component { state: { time: string, algorithm: number,

};

konstructor(props: Props) { super(props); this.state = { result: [], algorithm: 'Bubble_Sort',

```
        countfExperiments: NUMBER_OF_EXPERIMENTS,      phoneModel:
'iphone x',
        };
      }


      render() {
       konst {countfExperiments, phoneModel} = this.state;    return (
        <>
         <StatusBar barStyle="dark-content" />
         <View style={styles.container}>
          <RNPickerSelect
           style={pickerSelectStyles}          value={this.state.algorithm}
           onValueChange={(value)   =>   this.setState({algorithm:   value})}
items={EXPERIMENTS}
          />
          <Text style={styles.startTestText}>
           {`Добро пожаловать в иследование алгоритма ${
            EXPERIMENTS_MAPPER[this.state.algorithm]
           } c React
           Native`}
          </Text>
          <Text style={styles.label}>{phoneModel}</Text>
          <Text style={styles.label}>Количество эксперементов</Text>
          <TextInput
           placeholder="Количество                                эксперементов"
style={styles.input}          textAlign="center"
           onChangeText={(text)  =>  this.onChangeExperimentsNumber(text)}
value={countfExperiments.toString()}          keyboardType={'numeric'}
          />
```

```
            <TouchableOpacity
              style={styles.startTestButton}                    onPress={() =>
this.onPressGo()}          underlayColor="#fff">
              <Text style={styles.startTestText}>Начать Эксперемент</Text>
            </TouchableOpacity>
          </View>
        </>
      );
    }
  }
```

Сторінки, як і інші компоненти React Native, мають певний життєвий цикл, який має такий вигляд

- konstructor(): конструктор, у якому ініціалізується компонент;

- componentWillMount(): викликається перед рендерингом компонента; componentWillMount().

- Отрисовує компонент;

- componentWillUnmount(): викликається перед видаленням компонента з DOM;

Обробка класу сторінки показана на рисунку 3.7.

Під час створення нової сторінки, наприклад під час запуску програми, React Native викликає konstructor. Цей метод використовується для

ініціалізації сторінки. Зокрема, створюються такі об'єкти візуального інтерфейсу.

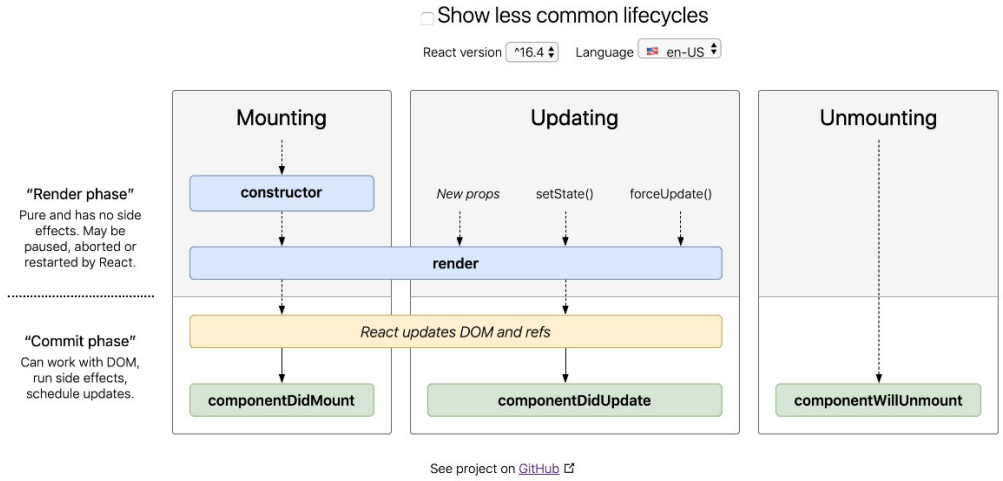


Рисунок 3.7 – життєвий цикл React

Як видно з коду, об'єкт state оголошується в конструкторі. Цей об'єкт описує внутрішній стан компонента, який визначається всередині компонента і може бути доступний тільки йому. Якщо об'єкт не використовується в рендерингу компонента, то немає сенсу зберігати його в state.

У багатьох випадках стан описує візуальну властивість елемента, яка може змінюватися під час взаємодії користувача з цим елементом. Наприклад, припустимо, що кнопка натиснута. Вона може змінити свій стан відповідним чином. Коли кнопку натискають знову, вона може повернутися в початковий стан. Щоб оновити стан, викличте функцію setState(). Зміна стану призведе до повторного рендерингу компонента і відповідного оновлення сторінки.

Під час проєктування та розроблення програмного забезпечення необхідно враховувати його швидкість і ефективність. Дуже важливо приділяти постійну увагу оптимізації програмного забезпечення на всіх етапах розробки.

Ми виконуємо бульбашкове сортування і відправляємо результати на сервер за допомогою Ajax без перезавантаження сторінки.

Далі ми реалізуємо обраний алгоритм. Для цього ми створюємо папку algoritms і описуємо реалізацію в ній.

Бульбашкове сортування.

function bubbleSort(inputArr) {

let len = inputArr.length; for (let i = 0; i < len; i++) { for (let j = 0; j <

```
len; j++) {         if (inputArr[j] > inputArr[j + 1]) {           let tmp = inputArr[j];
inputArr[j] = inputArr[j + 1];        inputArr[j + 1] = tmp;
            }
          }   }   return inputArr;
        }
```

- факторіал

```
function factorial(n) {   var result = 1;   while (n) {     result *= n--;
  }   return result;
}
```

- сортування вставками

```
konst insertionSort = (inputArr) => {   let length = inputArr.length;   for (let i
= 1; i < length; i++) {     let key = inputArr[i];     let j = i - 1;
            while (j >= 0 && inputArr[j] > key) {      inputArr[j + 1] = inputArr[j];      j
= j - 1;     }
            inputArr[j + 1] = key;
          }   return inputArr;
        };
```

Як бачите, кожен алгоритм винесено в окрему функцію і його можна викликати для тестування. Тепер перейдемо до реалізації функції tester.

```
onExperimentStart(algoritm) {
            konst {countfExperiments} = this.state;     konst experimentsResult = [];
            for (let j = 0; j < countfExperiments; j += 1) {           let startTime =
Timing.now();       algoritm ();
              let endTime = Timing.now();         let iterTime = endTime - startTime;
experimentsResult.push(iterTime);
            }
           this.sendDatum(experimentsResult);
```

```
}
```

Функція тестера onExperimentStart бере обраний алгоритм і послідовно виконує його задану кількість разів. На кожній ітерації береться час до і після виконання алгоритму. Потім обчислюється різниця. Це і є час виконання алгоритму.

Після проведення експерименту його результати необхідно передати на сервер. Для цього реалізована функція sendDatum.

```
sendDatum(datum) {
   konst {phoneModel} = this.state;

   fetch(ENDPOINT, {      method: 'POST',      headers: {
     Accept: 'Aplication/json',
     'Content-Type': 'Aplication/json',
    },
    body: JSON.stringify({          phoneModel,          method: METHOD,
platform: PLATFORM,        datum,
    }),
   })
    .then((response) => response.json())
    .then((json) => {        console.log('res', json);
    })
    .catch((error) => {        console.error(error);
    });
 }
```

Метод sendDatum приймає в себе результати експерименту, модель телефону, поточну платформу та алгоритм на якому проводили експеримент.

3.3.2 Реалізація логіки програмного продукту з Flutter

Під час розробки застосунку Flutter для IOS головним файлом є

main.dart. Завдання цього файлу - запустити застосунок і викликати першу сторінку. Давайте перейдемо до реалізації головної сторінки застосунку за допомогою Flutter.

```
class _HomeState extends State<Home> {
 String gaussLegendreTime = ''; String borweinTime = ''; int countfExperiments = 100;
 String experiment = 'Bubble_Sort';
 @override
 Widget build(BuildContext context) {
  return MaterialApp( home: Scaffold( body: Center( child: Column(
        crossAxisAlignment: CrossAxisAlignment.center, mainAxisAlignment: MainAxisAlignment.center, children: <Widget>[ new DropdownButton<String>( value: experiment,
          items: EXPERIMENTS.map((String value) { return new DropdownMenuItem<String>( value: value, child: new Text(value),
           ); }).toList(), onChanged: (value) { setState(() { experiment = value;
          });
         },
        ),
        Text(
          'Добро пожаловать в иследование алгоритма ${experiment} с Flutter'), TextField(
         decoration: InputDecoration( labelText:
            'Введите количество эксперементов. По умолчанию
    ${countfExperiments.toString()}'), keyboardType: TextInputType.number, onChanged: (text) {
```

```
                countfExperiments = int.parse(text);
              },
            ),
            RaisedButton(
              child: Text('Начать эксперемент'), onPressed: onPressed()
          ],
        ),
      ),
    ),
  );
 }
```

Знову ж таки, Flutter не використовує жодних нативних компонентів, тому немає потреби писати шари для взаємодії з компонентами. Натомість, подібно до ігрового рушія (ігри мають дуже динамічні користувацькі інтерфейси), весь інтерфейс сам по собі динамічний. Кнопки, текст, медіаелементи, фони - все це малює основний графічний рушій Flutter. Для побудови користувацького інтерфейсу Flutter використовує декларативний підхід, заснований на віджетах (так званих компонентах в онлайн-світі), натхненний веб-фреймворком ReactJS. Щоб прискорити роботу користувацького інтерфейсу, віджет перемальовується тільки тоді, коли в ньому щось змінюється (аналогічно тому, як це робить Virtual DOM у світі веб-інтерфейсів). Реалізація алгоритму не сильно відрізняється від реалізації React Native. Тут також створюється папка, в яку додається алгоритм. Давайте подивимося на реалізацію алгоритму.

Бульбашкове сортування.

```
runBubble_SortTest(array) {  int lengthOfArray = array.length;  for (int i = 0; i < lengthOfArray - 1; i++) {  for (int j = 0; j < lengthOfArray - i - 1; j++) { if (array[j] > array[j + 1]) {  // Swapping using temporary variable  int temp = array[j];  array[j] = array[j + 1];  array[j + 1] = temp;
```

```
        }
      }   }
    return (array);
  }
```

- факторіал

```
function factorial(n) {   var result = 1;   while (n) {     result *= n--;
  }
  return result;
}
```

- сортування вставками

```
var length = inputArr.length;     for (var i = 1; i < length; i++) {       var key =
inputArr[i];       var j = i - 1;
        while (j >= 0 && inputArr[j] > key) {         inputArr[j + 1] = inputArr[j];
j = j - 1;       }
        inputArr[j + 1] = key;
      }     return inputArr;
};
      for (int i = 0; i < iterations; i++) {       double aNext = (a + b) / 2;       double
bNext = math.sqrt(a * b);
        double tNext = t - p * math.pow(a - aNext, 2.0);       double pNext = 2 * p;
a = aNext;       b = bNext;       t = tNext;       p = pNext;
      }
      return math.pow(a + b, 2) / (4 * t);   }
```

Реалізація функції для проведення експериментів також не сильно відрізняється, використовуються фукції які представленні самим Flutter.

```
pressGaussLegendreButton(algoritm)  async  {                List<String>
resultOfExperiment = [];      for (int i = 0; i < countfExperiments; i += 1) {       var
stopwatch  =  new  Stopwatch();              stopwatch.start();              algoritm  ();
```

```
stopwatch.stop();
            var duration = (stopwatch.elapsedTicks / stopwatch.frequency) * 1000;
resultOfExperiment.add((duration).toString());
          }
          avait sendDatum(resultOfExperiment);     setState(() {
            gaussLegendreTime = resultOfExperiment.toString();
          });
        }
```

Як можна побачити логіка заміру часу аналогічка як і в ваіанті з React Native.

Розглянемо реалізацію методу sendDatum.

```
sendDatum(datum) async {
var deviceInfo = avait getDeviceInfo();

String body = jsonEncode({
'phoneModel': 'iphone x',
'method': METHOD,
'platform': PLATFORM,
'datum': datum
});
avait http.post(ENDPOINT, headers: {
'Accept': 'Aplication/json',
'Content-Type': 'Aplication/json'
}, body: body);
}
```

Метод sendDatum – відпраляє інформацію про експеримент, про середовище експерименту на його результати на сервер.

3.3.3 Реалізація логіки програмного продукту на Swift

Щоб розробити власний додаток на Swift необхідно встановити на комп'ютер і середовище розробки Xcode. Точкою входу звичайного Swift - модуля є файл модуля з ім'ям main.swift. main.swift -це єдиний файл, який може містити оператори і твердження верхнього рівня (всі інші Swift-файли в модулі містять тільки оголошення декларацій).

Cocoa Touch використовує атрибут @UIAplicationMainUIAplicationDelegate замість main.swift для вказівки точки входу; Cocoa викликає тільки мінімальний файл ,NSAplicationMain - тільки @UIAplicationDelegate.Ми звикли використовувати main.swift, але починаючи з Xcode атрибут @NSAplication Main використовується в реалізації NSAplicationDelegate.

```
struct ContentView:view.
State private var time: String = "0.0";
State private var countfExperiments: String = "100";
State private var method: String = "Factorial";
var body:aview{Menu("Дії") {{ { Menu("Дії")
    Button("Bubble_Sort", action: setBubble_Sort)
    Button("Insertion_Sort", action: setInsertion_Sort)
    Button("Binary_search", action: setBinary_search)
    Button("Fibonacci", action: setFibonacci) Button("Factorial", action: setFactorial)
}
Text("Добро пожаловать в иследование алгоритма " + methood + " с Swift")
Text(countfExperiments)
TextField("Введите количество эксперементов", text: $countfExperiments) Button(action:{
    var convertedCount: Int = Int(countfExperiments) ?? 100 if(method == "Bubble_Sort") {
```

```
            Bubble_Sort(n:convertedCount)
        }
        if(method == "Insertion_Sort") {
            Insertion_Sort(n:convertedCount)
        }
        if(method == "Binary_search") {
            Binary_search(n:convertedCount)
        }
        if(method == "Fibonacci") {
            Fibonacci(n:convertedCount)
        }
        if(method == "Factorial") {
            Factorial(n:convertedCount)
        }
    }){
        Text("Начать эксперемент")
    }
    Text(time)

}
```

Спочатку реалізуємо меню, що випадає, зі списком алгоритмів. Потім додамо кнопку для запуску експерименту і приступимо до реалізації алгоритму за допомогою Swift; Swift найбільше відрізняється від свого попередника, і причина цього криється в синтаксисі. Перейдемо до реалізації:

Бульбашкове сортування.

```
func runBubble_Sort(array: Array<Int>) -> Void {  var datumSet = array
let last_positione = datumSet.count - 1  var swap = true  while swap == true
{  swap = false  for i in 0..<last_positione {  if datumSet[i] >
datumSet[i + 1] {  let temp = datumSet [i + 1]  datumSet [i
+ 1] = datumSet[i]  datumSet[i] = temp
```

```
                    swap = true
                }
            }
        }
    }
```

- факторіал

```
func runFactorial(n: Int) -> Double {  if n == 0 {  return 1  }  var a: Double = 1  for i in 1...n {  a *= Double(i)
    }  return a
}
```

- сортування вставками

```
func runInsertion_Sort(a: Array<Int>) -> Void {  guard a.count > 1 else { return }
        var b = a  for i in 1..<b.count {  var y = i  let temp = b[y]  while y > 0 && temp < b[y - 1] {  b[y] = b[y - 1]  y -= 1  }  b[y] = temp  }  return
}
```

Swift - це безпечна за типом мова, що означає, що вона допомагає вам зрозуміти, з яким типом працює ваш код. Якщо фрагмент коду очікує отримати рядок, безпека типів не дозволить вам випадково передати тип Int. Безпека типів також запобігає випадковій передачі необов'язкового рядка у фрагмент коду, який очікує необов'язковий рядок. Безпека типів гарантує, що помилки будуть виявлені і виправлені якомога раніше в процесі розробки. Константи і змінні повинні бути оголошені перед їх використанням. Константи оголошуються за допомогою ключового слова let, а змінні - за допомогою ключового слова var.

Наступним кроком є реалізація функції, яка запускає експеримент і відраховує час.

```
func runExperiment(experiment: Func) -> Void {  var j: Int = 0;
    var convertedCount: Int = 100  var numbers: [Double] = []  var
```

```
numbersString: [String] = []        while((j < n) == true) {
                let startTime =  DispatchTime.now();                experiment (n: convertedCount)          let endTime =  DispatchTime.now();
                let iterTime = Double(endTime.uptimeNanoseconds-startTime.uptimeNanoseconds);          numbers.append(iterTime/1000000)
                numbersString.append(String(iterTime/1000000))          j = j+1
            }
            time = numbersString.joined(separator: ",");
            sendDatum(_datum: numbers, method: method, count: countfExperiments);    }
```

Як бачите, ця функція реалізована так само, як і в попередніх версіях: після отримання результату викликається функція sendDatum, і результат відправляється на сервер.

URLRequest містить дві основні властивості запиту на завантаження: URL, який потрібно завантажити, і політику, використовувану для його завантаження. Крім того, для HTTP- і HTTPS-запитів URLRequest містить HTTP-методи (наприклад, GET, POST) і HTTP-заголовки.

```
func sendDatum(_datum: Array<Double>, method: String, count: String) -> Void {     let Url = String(format: Konstants.API_URL)     guard let serviceUrl = URL(string: Url) else { return }     let parameters: [String: Any] = [
            "method" : method,
            "platform": Konstants.PLATFORM,
            "phoneModel": "iphone x",
            "datum": _datum,
        ]
        var request = URLRequest(url: serviceUrl)    request.httpMethod = "POST"
        request.setValue("Aplication/json", forHTTPHeaderField: "Content-Type")
        guard let httpBody = try? JSONSerialization.datum(withJSONObject: parameters, options: [])
```

```
        else {
            return
          }
          request.httpBody = httpBody     request.timeoutInterval = 20     let session
= URLSession.shared
          session.datumTask(with: request) { (datum, response, error) in          if let
response = response {          print(response)
          }
          if let datum = datum {          do {
                let json = try JSONSerialization.jsonObject(with: datum, options:
[])          print(json)          } catch {          print(error)
            }
          }
        }.resume()
      }
```

Зверніть увагу на додавання примусового розширення ініціалізатора URL (:) string). Створення URL з рядків може бути невдалим через вставляння безглуздих рядків, але тут URL вводяться вручну, тому вони завжди коректні - немає інтерполяції рядків, яка могла б викликати проблеми.

3.4. Внутрішнє проектування

3.4.1 Технологічна платформа

Наразі WebStorm є найпопулярнішим середовищем розробки для мови Node.js.

Ключовими особливостями є.

- Автозавершення, форматування і підсвічування синтаксису;
- Рефакторинг коду;;
- Підтримка MVC-фреймворків;

- Редактор html, css і javascript;

- Інтеграція з системами контролю версій;

- php uml;

- Інструменти для роботи з базами даних результатів;

Методологія - це набір принципів, методів, концепцій, інструментів і методик, які пропонують стиль розробки програмного забезпечення.

Сама методологія впливає на те, як ведеться розробка.

Існує безліч різних методологій розробки програмного забезпечення. Вибір методології залежить від складності проєкту, напряму розробки та індивідуальних особливостей розробника.

Методології agile-розробки - це низка підходів до розроблення програмного забезпечення, що орієнтовані на інтерактивне розроблення з використанням динамічно генерованих вимог [47, с. 35].

З методологіями agile-розробки пов'язано кілька методик, таких як Scrum, DSDM, Extreme Programming і FDD.

Більшість цих методологій спрямовані на зменшення ризику шляхом поділу процесу розробки на низку коротких циклів, які називаються ітераціями. Ітерація зазвичай триває кілька тижнів, і кожна ітерація є мініатюрним програмним проектом, що охоплює всі завдання, необхідні для запуску проекту (планування, дизайн, аналіз вимог, програмування, документування та тестування).

Під час ітерації неможливо випустити нову версію продукту, але передбачається, що листи ПП готові до випуску в кінці кожної ітерації. В кінці ітерації відбувається переоцінка пріоритетів розробки.

Для подальшого розроблення вирішено дотримуватися методології Scrum.

Для управління agile-завданнями і процесами використовували систему управління проектами Youtrack.

3.4.2. Google Sheets як база для результатів експериментів

Електронні таблиці завжди були потужним (хоча й буквальним) аналогом баз даних результатів. База даних результатів містить таблиці, які схожі на одну електронну таблицю. Уявіть собі електронну таблицю для відстеження відгуків на весілля. У верхній частині є заголовки стовпців, як-от "Ім'я", "Прізвище", "Адреса" та "Присутні". Ці заголовки також є стовпцями в основній таблиці результатів. Інакше кажучи, кожна людина в цій таблиці - це буквально рядок, а також рядок в основній таблиці результатів (або запис, елемент або навіть кортеж, якщо ви справді гік).

Дедалі частіше можна зустріти твердження, що це не аналогія. Ми можемо використовувати інтерфейс електронної таблиці, щоб перетворити її на справжню базу результатів. Це важливо, тому що йдеться не просто про відображення результатів бази результатів у вигляді електронної таблиці, а й про створення першокласними громадянами застосунку функції, схожої на електронну таблицю, разом із функцією, схожою на електронну таблицю.

У випадку з електронними таблицями мета може полягати в тому, щоб бачити речі як єдине ціле і розуміти їх як такі. Перегляд, сортування, введення та редагування результатів безпосередньо в користувацькому інтерфейсі для отримання корисних візуальних результатів; Google Sheets часто використовується як редактор електронних таблиць. Водночас його можна використовувати як базову базу даних результатів. Можна створювати прототипи різних моделей і перевіряти гіпотези.

3.4.3. Обробка результатів експериментів

Для опрацювання результатів експерименту було створено сервер із використанням node.js і фреймворку express Node.js - це проміжна версія JavaScript, заснована на русі JavaScript V8 у Chrome Node.js базується на неблокуючих операціях вводу-виводу та базується на моделі спільного управління з простими й ефективними операціями.

Node.js надає можливість писати неймовірно потужний код на стороні сервера, використовуючи JavaScript. Як ідеться в офіційному описі, Node.js -

це проміжне програмне забезпечення, що використовує той самий рушій V8 JavaScript, що й браузер Google Chrome. Однак цього недостатньо для успіху Node.js: він використовує libuv, кросплатформенну бібліотеку підтримки, яка фокусується на асинхронному введенні-виведенні.

Веб-додаток, написаний у цій архітектурі клієнт/сервер, працює так: клієнт запитує ресурси у сервера, а сервер відправляє ресурси у відповідь. У цій серверній схемі відповідь на запит асоціюється зі з'єднанням.

Ця модель справедлива для всіх запитів до сервера, які потребують ресурсів (пам'яті, процесорного часу тощо). Щоб виконати кожен наступний запит від клієнта, сервер має завершити обробку попереднього запиту.

Це означає, що за одне запрошення сервер може виконати тільки один запит.

Коли сервер отримує новий запит, він створює ще один потік для його обробки.

Простіше кажучи, потік - це час і ресурси, які процесор витрачає на виконання невеликого блоку інструкцій. У цьому поданні сервер може виконувати кілька запитів одночасно, але тільки один запит на потік. Ця модель відома як модель потоків на вимогу.

Я використовував Google API для роботи з Google Sheets Робота з Google

Робота починається з авторизації. Найпростіший спосіб авторизації - це замовчування значень аргументів, отриманих функцією gs4_auth().

Недоліком цього методу є те, що зрештою ви, як і 90 % інших користувачів, використовуєте застосунок для вимкнення звуку. Оскільки у кожного додатка є квота на кількість запитів, цей пакет буде збільшуватися в міру зростання числа користувачів, потенційно перевищуючи обмежений ліміт. Однак для нашого завдання цього цілком достатньо.

Основою для зберігання результатів є сторінки. Нижче наведено приклади методів, які створюють нові сторінки або повертають уже створені. Створити нову сторінку async (document, SheetTitle) => { let Sheet = document.ScheetsByTitle[SheetTitle]; if (!Sheet) {

```
    Sheet = avait document.addSheet({ title: SheetTitle }
  } return Scheet;
 }
```

Далі, використовуючи Google Api, ми отримали можливість надати експериментальні результати, отримані з мобільного застосунку. Під час побудови таблиці модуль смартфона враховує кількість ітерацій. Потім він обчислює середній час, витрачений додатком на кожну ітерацію. Для цього ми створили модуль, який інкапсулює цю логіку:

```
addValuesToTable: async (googleScheet, platform, section, datum) => { konst { startLetter, startNumber, endLetter, endNumber,
  } = getParsedWorkingSection(section); console.log("Start of add values");
 if (startLetter && startNumber && endLetter && endNumber) {
  konst letterToWrite = Scheets.getLetterByPlatform(platform, startLetter);

  for (let i = 0; i < datum.length; i = i + 1) { konst numberToWrite = i + 1 + OFFSET;
   konst numberCellA1 = `${letterToWrite}${numberToWrite}`; konst numberCell = avait googleScheet.getCellByA1(numberCellA1); Scheets.createAlignedCell(numberCell, datum[i]); console.log(`Value ${numberCellA1} added`);
  }
  konst average =
   _.sumBy(datum, (item) => { if (_.isNumber(item)) { return item;
    }

   return parseFloat(item);
  }) / datum.length;
```

```
        console.log("Average  value added");
        konst averageCellA1 = `${letterToWrite}${endNumber}`; konst
averageCell = avait googleScheet.getCellByA1(averageCellA1);
Scheets.createAlignedBoldCell(averageCell, average); console.log("End of add
values");
      } else {
        throw Error("Cannot work with this section");
      }
    },
```

Для зберігання даних було розроблено структуру сторінок, що дає змогу розглядати і переглядати результати роботи кожного інструменту розробки і кожного смартфона окремо. Приклад таблиці з результатами показано на рисунку 3.8. У ній чотири основні стовпці. У першому стовпчику вказано модель смартфона, на якому проводили експеримент, і номер експерименту, у наступному - результати React Native, потім - результати Flutter і Swift. Для кожного експерименту і кожної платформи окремо обчислюється середнє значення прогонів алгоритму. Зберігання результатів у такому форматі дає перевагу під час підбиття підсумків. Для кожного експерименту у нас є свої середні значення, які можна просто використовувати для подальшої побудови графіків.

A	B	C	D	E	F	G	H	I	J	K	L
iphone 8				iphone 11				iphone 6s			
	Сортировка пузирьком				Сортировка пузирьком				Сортировка пузирьком		
№	React Native	Flutter	Swift	№	React Native	Flutter	Swift	№	React Native	Flutter	Swift
1	1.379916668	0.095042	31.660083	1	1.196916666	0.443417	21.725708	1	0.4142916799	72583000000000	24.089833
2	0.9510834217	0.039875	15.100959	2	0.8507083319	20208000000000	12.601792	2	0.2764583379	29207999999999	16.711417
3	0.9718750119	0.03975	8.908584	3	0.8322500028	0.220125	8.233958	3	0.2583333403	29167000000000	10.283
4	1.249208331	0.039375	8.498625	4	0.8577083312	20041000000000	7.2715	4	0.2598749995	29207999999999	9.010458
5	1.244333386	0.039667	7.455083	5	0.8604583368	0.22	7.257708	5	0.290625006	0.029166	8.101958
6	1.324666679	0.039875	6.24425	6	0.8448333368	20082999999999	5.661417	6	0.2944166511	0.029375	6.613042
7	1.137208283	0.0395	6.248334	7	0.8355000019	0.22	5.335041	7	0.2942916602	29167000000000	6.626416
8	0.9567500353	0.039792	5.673209	8	0.8317083307	0.22	5.309833	8	0.2942083329	29167000000000	6.015417
9	0.9512916803	0.039625	5.362458	9	0.8316249996	0.220084	4.897208	9	0.2948333323	29332999999999	5.621708
10	0.9507500529	52333000000000	4.894709	10	0.8315833323	0.220042	4.667125	10	0.3076250106	29207999999999	5.16
11	1.041583359	0.039792	4.838375	11	0.8713749982	0.220042	4.255042	11	0.2954999954	29207999999999	5.053834
12	0.9517916441	0.040458	4.755333	12	0.8322916701	20082999999999	4.16475	12	0.2942916602	29167000000000	5.067583
13	0.9608333707	0.039875	4.750458	13	0.8482083306	47791000000000	4.088458	13	0.3086249977	0.029375	5.030208
14	0.9552916884	0.039666	4.757375	14	0.9403333291	0.220416	4.064791	14	0.295541659	29332999999999	5.024459
15	0.9543750286	40958999999999	4.764584	15	0.9395416677	20082999999999	4.057542	15	0.2945833206	29249999999999	5.057584
16	0.7797916532	0.039708	4.74875	16	0.8323333338	0.370875	4.061042	16	0.2945833206	0.029292	5.053875
17	0.7767916918	0.039708	4.765625	17	0.8534583338	0.296125	4.064708	17	0.2951250076	0.029375	5.096541
18	0.7692916393	0.039541	4.759583	18	0.7943333313	0.177125	4.055125	18	0.2954999954	0.029375	5.030917
19	0.7692916989	0.039667	4.778791	19	0.7011666633	0.181917	4.045583	19	0.541625008	0.029166	5.015666
20	0.8506249785	0.03975	4.774625	20	0.6930416673	0.204625	4.049791	20	0.6418333352	0.029792	4.9965
Average	0.9963375151	0.0431979	7.38698965	Average	0.8539687498	0.2391541	6.1934061	Average	0.3271083325	0.03144575	7.4330208

Рисунок 3.8 – структура зберігання результатів експерименту

Коли буде зібрано достатньо результатів, можна побудувати графік, що наочно показує, як змінюються результати залежно від кількості ітерацій. На мій погляд, найкращий спосіб показати цю залежність - побудувати лінійний графік. Лінійний графік відображає розміри показника у вигляді ліній різної довжини, які утворюються в результаті з'єднання точок координатного поля. Одним із видів лінійного графіка є лінійний графік запланованого виконання та опублікований лінійний графік.

```
createChart: async (googleScheet, countfRows) => {     konst requestType = "addChart";     konst AXIS = [
            {
              positione: "BOTTOM_AXIS",
              title: "Count of experiments",
            },      {
              positione: "LEFT_AXIS",        title: "Time",
            },     ];
          konst countfIterationDatum = {          domain: {             sourceRange: { sources: [            {
                    ScheetId:  googleScheet.ScheetId,                  startRowIndex:  0, endRowIndex: countfRows,          startColumnIndex: 0,          endColumnIndex: 1,
                  },
                ],
              },
            },
          };
          konst    {    reactNativeSource,    flutterSource,    swiftSource    }    = Chart.getSources(      countfRows,      googleScheet.ScheetId
          );
```

```
konst rnResponse = avait googleScheet._makeSingleUpdateRequest(requestType, { chart: { spec: {
title: "Count of experiments and React Native", basicChart: { chartType: "LINE", legendPositione: "BOTTOM_LEGEND", axis: AXIS,
domains: [countfIterationDatum], series: [reactNativeSource], headerCount: 1,
}, }, positione: { newScheet: true,
//ScheetId: rnScheetId,
},
},
});
konst flutterResponse = avait googleScheet._makeSingleUpdateRequest( requestType, { chart: { spec: {
title: "Count of experiments and Flutter", basicChart: { chartType: "LINE", legendPositione: "BOTTOM_LEGEND", axis: AXIS,
domains: [countfIterationDatum], series: [flutterSource], headerCount: 1,
}, }, positione: { newScheet: true,
//ScheetId: flutterScheetId,
},
},
}
);
konst swiftResponse = avait googleScheet._makeSingleUpdateRequest( requestType, { chart: { spec: {
title: "Count of experiments and Swift", basicChart: { chartType: "LINE", legendPositione: "BOTTOM_LEGEND", axis: AXIS,
```

```
                domains: [countfIterationDatum],          series: [swiftSource],
headerCount: 1,
                },        },        positione: {        newScheet: true,
                //ScheetId: swiftScheetId,
              },
            },
          }
        );     return {
      rnScheetId:  rnResponse.chart.positione.ScheetId,          flutterScheetId:
flutterResponse.chart.positione.ScheetId,swiftScheetId:
swiftResponse.chart.positione.ScheetId,
        };
      },
```

Ця функція збирає середні результати всіх експериментів і готує структуру для передачі в Google API. Потім можна побудувати графік, що показує залежність кількості експериментів від середнього часу ітерації; приклад результатів цієї функції в порівнянні з результатами алгоритму Гаусса-Лежандра показано на рисунку 3.9.

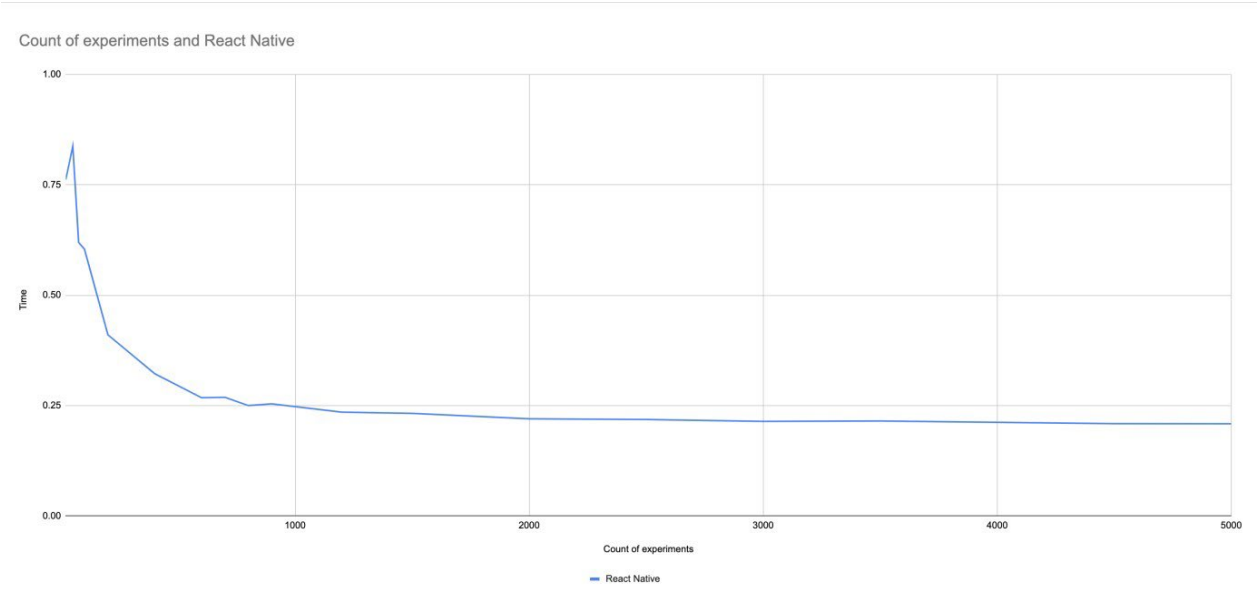


Рисунок 3.9 – створений графіку

Висновки по розділу 3

У цьому розділі розробляється мобільний застосунок, реалізується логіка попередньо обраних алгоритмів і надається сервер для обробки частини коду застосунку та результатів експериментів. Розроблено інтуїтивно зрозумілий користувацький інтерфейс, представлений у вигляді діалогового меню.

У розділі також описано наявні методи тестування програмного забезпечення та проведено порівняльний аналіз найпопулярніших емуляторів для тестування iOS-додатків.

РОЗДІЛ 4. Тестування та налагодження програми

4.1 Аналіз методів тестування та відлагодження

Тестування є завершальним етапом розробки програмного продукту і відіграє важливу роль у створенні високоякісного програмного продукту.

Чим складніший програмний продукт, тим більше часу і ресурсів потрібно для тестування.

Часто розробники пропускають етап тестування, що призводить до збільшення часових і фінансових витрат у майбутньому та необхідних змін у програмному забезпеченні.

Існують спеціально розроблені методики для організації тестування сайтів, в рамках яких виконуються всі тести.

Функціональне тестування - найтриваліший етап, під час якого перевіряється весь функціональний стан:

- Перевірка правильності роботи всіх функцій програми;

- Перевірка правильності роботи всіх функцій сервера; і

Для перевірки основних функцій програми було створено тестову сторінку, засновану на результатах mon_test. Також були створені модульні тести. Юніт-тест, або блоковий тест, модульний тест: процес перевірки коректності окремих модулів вихідного коду програми, набору з одного або

декількох програмних модулів, а також їхнього контролю, використання та правильного поводження в програмному забезпеченні.

Крім того, на цьому етапі перевірялися інші тестові функції програмного забезпечення, працездатність усіх форм, посилань, меню та інших використовуваних елементів.

Залежно від розширення користувацький інтерфейс має на екрані різний вигляд,

Для досягнення найвищої якості тестування для кожного методу або модуля тестованого додатка потрібно застосовувати найвдаліший метод або набір методів, що забезпечують необхідні результати. При цьому необхідно вибрати оптимальне співвідношення між часом, що витрачається на тестування, і типом тестування. Серія тестів має приносити мінімальну користь і водночас охоплювати максимальну функціональність системи.

4.2 Тестування методом чорної шухляди

Основна ідея перевіреної системи, а в даному випадку весь матеріал, доступний для тестування, - це вимоги до системи, що описують її роботу. Усі внутрішні призначені для користувача реалізації системи формуються на основі результатів неточних тестів, тож система являє собою "чорну скриньку", і її коректність може бути перевірена на відповідність вимогам.

З точки зору програмного коду контейнер може являти собою набір класів (або модулів) з відомими зовнішніми інтерфейсами, але без вихідного коду.

Пробірки використовуються для методів, які перевіряються шляхом контролю поведінки системи два рази поспіль. Крім того, в ідеальній ситуації, коли всі варіації критичної ситуації вже розглянуто, вимоги до системи мають бути виконані, і експерту залишається тільки запропонувати конкретні вимоги. Однак у реальності випробування виявили дві системні проблеми Невідповідність вимогам поведінкової системи. Відсутність необхідності перенесення системи в різних ситуаціях, відсутність перенесення трафіку.

Проблеми з перевіркою декількох типів документів або втрачені записи. Щоб вирішити такі проблеми, натисніть кнопку Перейти до кодування або рідше обирайте простий спосіб: змінити вимогу. В іншому випадку вимогу може знадобитися змінити, тому що вона неоднозначна (кілька різних вимог описують різні моделі поведінки системи в одній і тій же ситуації) або некоректна (вимога не є правильною).

Другий тип проблем явно вимагає зміни вимоги через її неповноту: у вимозі явно упущена ситуація, яка призводить до неправильного підключення системи. З цієї причини термін використовується зацікавленою стороною.

Вона перевіряється використовуваним додатком, про що є інформація в плані тестування. Останній тест і тест, що закриває 80% продуктивності програми. Приклад тестового випадку наведено в таблиці 4.1.

Таблиця 4.1 – Приклад тест кейсу

Номер	1
Назва Експерименти з алгоритмами бульбашкового сортування на смартфонах	Експерименти з алгоритмами бульбашкового сортування на смартфонах
Необхідні умови	Додаток встановлено та запущено на смартфоні.
Кроки	Результати експерименту
Виберіть алгоритм	"Сортування бульбашками" у випадаючому списку Алгоритм.
Алгоритм	"Сортування бульбашками" вибрано Експеримент "Сортування бульбашками" вибрано
Встановіть кількість експериментів 30	Встановіть кількість експериментів 30
Перевірики Google Sheets	Результати експерименту додаток до таблиці

4.3 Тестування методом білої шухляди

При тестуванні системи як "білої скриньки" тестувальник має доступ не тільки до вимог, входів і виходів системи, але й до її внутрішньої структури, тобто програмного коду [7].

Можна перевірити, чи відповідає частина програмного коду вимогам, а також визначити, чи присутні вимоги у всьому програмному коді. Програмний код, який не містить вимог, називається кодом без вимог. Такий код може спричинити неналежну поведінку системи. Крім того, більша прозорість системи дає змогу глибше проаналізувати ділянки, які викликають проблеми. При тестуванні білого ящика пишуться модульні тести, що охоплюють 95 відсотків коду. Приклад наведено нижче.

```
describe(Buble Sorting Test, () => {
    it('Array should be sorted with bubble sort, () => {
konst arrayToTest = [5,6,7,3,6,0,3,2] konst expectedArray = [0,2,3,3,5,6,6,7]
konst result = bubbleSort(arrayToTest);
            expect(result.toEqual(expectedArray))
},
it('Array should be sorted with inservion sort, () => { konst arrayToTest =
[5,6,7,3,6,0,3,2] konst expectedArray = [0,2,3,3,5,6,6,7] konst result =
insertionSort(arrayToTest);
            expect(result.toEqual(expectedArray)) } });
```

Висновки по розділу 4

Тестування автоматизованої системи проводиться за допомогою функціональних методів, під час яких текст програми недоступний і розглядається як "чорний ящик".

Результати тестування підтверджують ефективність і коректність роботи розробленої автоматизованої системи тестування алгоритмів.

Загальні висновки

У результаті виконання магістерської дисертації було сформовано вимоги, вивчено та засвоєно інформацію за даним напрямом, обрано методи та прийоми проектування, сформовано та реалізовано макет. Було обрано базу даних, а також саму систему, всі дані зберігаються в хмарі. Програма має графічний інтерфейс і легко доступна для всіх користувачів, незалежно від рівня їхньої підготовки.

На основі аналізу, проведеного в рамках проєкту, було запропоновано критерії оцінювання програмного забезпечення та проведено порівняльний аналіз рішень, уже представлених на ринку програмного забезпечення.

Наступним кроком став огляд літератури та веб-ресурсів для кращого розуміння технологій, необхідних для розробки програмного забезпечення. На підставі всього цього було зроблено висновок про доцільність розробки нової програми, яка дасть змогу усунути всі наявні недоліки.

Основними етапами стали проектування, тестування, налагодження та експерименти. Ці етапи, а також усі методи, використані під час розроблення, детально описано. Детально описано, як розроблялася і тестувалася система.

Ajax - це технологія з величезним потенціалом і можливостями, яка нині посідає важливе місце в глобальній мережі. Програмний продукт, розроблений у цій статті, вирізняється тим, що спрямований на скорочення часу відгуку сервера і поліпшення якості користувацького інтерфейсу.

Крім того, під час експериментів було виявлено закономірність скорочення часу проведення експериментів під час поліпшення основних технічних компонентів смартфона, включно з:

- оперативна пам’ять
- кеш-пам'ять

- тактова частота процесора

ДОДАТОК А

Технічне завдання

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Перший проректор Українського державного університету науки і технологій
Анатолій РАДКЕВИЧ
02.10.23

ДОСЛІДЖЕННЯ ЧАСОВОЇ ЕФЕКТИВНОСТІ AJAX-ЗАПИТІВ

Технічне завдання

ЛИСТ ЗАТВЕРДЖЕННЯ
44165850.01356-01 12 01

Представники підприємства-розробника
Завідувач кафедри КІТ
Вадим ГОРЯЧКІН
02.10.23
Керівник розробки
Вадим АНДРЮЩЕНКО
02.10.23
Виконавець
Сергій ТАРЯНИК
02.10.23
Нормконтролер
Світлана ВОЛКОВА
02.10.23

2024

ЗАТВЕРДЖЕНО

44165850.01356-01 12 01

ДОСЛІДЖЕННЯ ЧАСОВОЇ ЕФЕКТИВНОСТІ AJAX-ЗАПИТІВ

Технічне завдання

44165850.01356-01 12 01

Листів 17

2023

ЗМІСТ

ВСТУП

Часова ефективність AJAX-запитів є ключовою складовою успішної розробки веб-додатків, оскільки впливає на швидкість та відзивчивість веб-сайту або веб-додатка перед користувачами. AJAX (Asynchronous JavaScript and XML) визначає підхід до взаємодії з сервером без перезавантаження сторінки. Цей технічний документ присвячений дослідженню та аналізу часової ефективності AJAX-запитів у веб-розробці.

Мета даного технічного завдання - провести докладне дослідження та оцінку часових показників виконання AJAX-запитів у різних сценаріях веб-додатків. Дослідження покликане визначити, як оптимізувати та покращити продуктивність AJAX-запитів для забезпечення максимальної відзивчивості та задоволення потреб користувачів.

Це дослідження спрямоване на забезпечення максимальної швидкості та відзивчивості веб-додатків, що є важливим аспектом в сучасній веб-розробці. Успішне впровадження рекомендацій цього технічного завдання сприятиме покращенню користувацького досвіду та конкурентоспроможності веб-продукту на ринку.

1. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є наказ 1196 ст від 05.12.2022 року ректора Українського державного університету науки і технологій “Про призначення наукових керівників та затвердження тем магістерських робіт” за спеціальністю 121 “Інженерія програмного забезпечення» факультету “Комп’ютерних технологій і систем” по кафедрі “Комп’ютерні інформаційні технології”.

2. ПРИЗНАЧЕННЯ РОЗРОБКИ

Ця розробка має на меті дослідження та оцінку часової ефективності AJAX-запитів у веб-додатках. Головною метою є вивчення оптимальних підходів до використання AJAX-запитів для забезпечення ефективної взаємодії між клієнтом та сервером.

Розробка цього проекту також спрямована на вдосконалення розуміння принципів та технологій, пов'язаних із веб-розробкою та асинхронним обміном даними. Вивчення можливостей AJAX-запитів дозволить оптимізувати взаємодію веб-додатків, зменшити час завантаження сторінок та покращити користувацький досвід.

Результати цього дослідження можуть бути використані в розробці веб-додатків та впроваджені для покращення продуктивності та реакції додатків на користувацькі запити.

Функціональне призначення розробки:

1. Створення імплементації AJAX-запитів: Розробка програмної імплементації, яка дозволить виконувати асинхронні запити до сервера з використанням AJAX-технології.
2. Збір та аналіз даних про час виконання запитів: Реалізація механізмів для збору даних про час, необхідний для виконання кожного AJAX-запиту та їхню обробку.
3. Визначення часової ефективності: Розробка функцій для обчислення часової ефективності виконання AJAX-запитів і визначення, наскільки швидко вони виконуються.
4. Аналіз факторів, що впливають на час виконання: Вивчення та аналіз чинників, таких як обсяг передачі даних, мережеві умови та оптимізація коду, які можуть впливати на час виконання AJAX-запитів.
5. Подання результатів та рекомендацій: Вивчені та проаналізовані дані про часову ефективність AJAX-запитів подаються у зручному форматі, а також надаються рекомендації щодо покращення часової ефективності виконання запитів.
6. Інтеграція із системами: Забезпечення можливості інтеграції цієї розробки у веб-додатки та системи, які використовують AJAX-запити.

Ця розробка спрямована на дослідження та вдосконалення часової ефективності виконання AJAX-запитів, що відіграють важливу роль у сучасних веб-додатках та взаємодії з сервером.

Експлуатаційне призначення розробки:

1. Визначення оптимальних параметрів AJAX-запитів: Розроблена система може використовуватися для визначення оптимальних параметрів AJAX-запитів, таких як методи передачі даних, розмір пакетів, кількість одночасних запитів тощо.
2. Покращення продуктивності веб-додатків: Результати дослідження

можуть бути використані для оптимізації веб-додатків та підвищення їхньої продуктивності за рахунок зменшення часу виконання AJAX-запитів.

3. Моніторинг та аналіз в реальному часі: Розроблену систему можна використовувати для постійного моніторингу та аналізу часу виконання AJAX-запитів у реальному часі.
4. Вдосконалення користувацького досвіду: Інформація, зібрана системою, може використовуватися для створення веб-додатків, які надають користувачам швидкі та відзивчиві інтерфейси, покращуючи їхній загальний досвід.
5. Підтримка розробників та адміністраторів: Система може бути корисною для веб-розробників та системних адміністраторів, які можуть використовувати її результати для вдосконалення веб-додатків та інфраструктури.
6. Інтеграція з іншими інструментами: Розроблену систему можна інтегрувати з іншими інструментами для моніторингу та аналізу веб-додатків, що дозволить створити комплексний підхід до покращення продуктивності.

Експлуатаційне призначення полягає в застосуванні цієї розробки для покращення часової ефективності AJAX-запитів та оптимізації веб-додатків для досягнення найкращого користувацького досвіду.

3. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

3.1 Вимоги до функціональних характеристик

Збір та моніторинг даних: Система повинна надавати можливість збору та моніторингу даних, пов'язаних з виконанням AJAX-запитів у веб-додатку.

Автоматизований аналіз: Система повинна забезпечувати автоматизований аналіз часу виконання AJAX-запитів, включаючи визначення середнього часу, максимального та мінімального значень.

Підтримка реального часу: Система повинна забезпечувати можливість моніторингу та аналізу в реальному часі для негайного реагування на можливі проблеми з часовою ефективністю.

Графічне відображення результатів: Система повинна надавати можливість графічного відображення результатів аналізу у вигляді діаграм, графіків, або інших візуальних засобів.

Підтримка кількох проектів: Система повинна бути здатною працювати з кількома проектами одночасно, зберігаючи відокремлені дані для кожного проекту.

Можливість налаштування параметрів тестування: Користувач повинен мати можливість налаштовувати параметри тестування, такі як інтервали часу між запитами, кількість повторень, тощо.

Експорт результатів: Система повинна дозволяти користувачам експортувати результати аналізу у різних форматах, таких як CSV, Excel, або PDF.

Безпека даних: Забезпечення безпеки даних та заборону несанкціонованому доступу до них, включаючи права доступу користувачів.

Інтеграція з іншими системами: Можливість інтеграції цієї системи з іншими інструментами моніторингу та аналізу продуктивності.

Вхідні дані:

URL-адреси AJAX-запитів: Список URL-адрес, до яких буде відправлено AJAX-запити для аналізу.

Параметри запитів: Якщо AJAX-запити вимагають передачі параметрів, система повинна отримувати ці параметри для кожного запиту.

Налаштування тестування: Параметри, такі як інтервали часу між запитами, кількість повторень, налаштування тестових умов (наприклад, використання кешу), інші параметри тестування.

Вимоги до тестового середовища: Опис середовища, в якому будуть виконуватись AJAX-запити, включаючи характеристики сервера, мережі та інфраструктури.

Вихідні дані:

Результати аналізу: Це включає середні, мінімальні та максимальні значення часу відповіді на кожен AJAX-запит, а також загальний час виконання всіх запитів.

Графіки та діаграми: Візуальне відображення результатів аналізу у формі графіків, діаграм чи інших візуальних засобів.

Звіти: Повні та зрозумілі звіти, які можна експортувати у різних форматах (наприклад, PDF або Excel).
Рекомендації та висновки: Інформація щодо оптимізації часової ефективності AJAX-запитів та можливих заходів для поліпшення продуктивності веб-додатків.

3.2 Вимоги до надійності

Вимоги до надійності наступні:
Стабільність тестового середовища: Тестове середовище повинно бути стабільним і надійним. Будь-які збої чи відмови на стороні сервера, мережі або інфраструктури можуть спричинити неправильні результати. Забезпечення стабільності тестового середовища є важливим завданням.
Моніторинг і управління помилками: Система повинна вміти виявляти та обробляти помилки під час виконання AJAX-запитів і аналізу даних. Це включає в себе ведення журналу помилок та можливість відновлення роботи в разі виникнення помилок.
Захист даних: Забезпечення конфіденційності даних та їх інтегритету важливо. Всі дані, включаючи результати тестів, повинні бути захищені від несанкціонованого доступу чи модифікації.
Резервне копіювання даних: Проведені тести та результати аналізу повинні регулярно резервуватися. Це допоможе у відновленні даних в разі їх втрати чи пошкодження.
Документація і відстеження: Всі етапи дослідження, включаючи параметри та результати тестів, повинні бути докладно задокументовані. Інформація про використовувані засоби та методи моніторингу також повинна бути доступною для перевірки і відстеження.

3.3 Вимоги експлуатації

Працювати з програмою може людина, що має навички роботи з мобільними пристроями та комп’ютером, а також ознайомлена з керівництвом користувача програмного продукту.

3.4 Вимоги до складу та параметрів технічних засобів

Продукт, що розробляється може використовуватись на мобільних пристроях, що мають наступні характеристики:
діагональ екрану – 5.5;
роздільна здатність дисплею – HD (1280x720);
оперативна пам’ять – 2 ГБ;
вбудована пам’ять – 16 ГБ;
операційна система – IOS;
частота процесора – 1.6 ГГц;
Продукт, що розробляється може використовуватись на комп’ютерах, що мають наступні характеристики:
Операційна система: Комп'ютер повинен працювати під управлінням операційної системи, яка підтримує виконання програмного продукту. Для прикладу, це може

бути операційна система Windows, macOS або Linux.
Діагональ монітора: Монітор комп'ютера повинен мати достатньою діагональ для зручного відображення інтерфейсу продукту та взаємодії з ним. Ідеально, це може бути монітор із середньою діагоналлю, наприклад, 21-24 дюйми.
Роздільна здатність монітора: Монітор повинен мати достатньо високу роздільну здатність для чіткого та якісного відображення інтерфейсу продукту. Рекомендована роздільна здатність - Full HD (1920x1080) або вища.
Оперативна пам'ять (RAM): Комп'ютер повинен мати достатньо оперативної пам'яті для ефективної роботи програмного продукту. Рекомендована кількість оперативної пам'яті - не менше 4 ГБ.
Вбудована пам'ять (жорсткий диск або SSD): Комп'ютер повинен мати достатньо внутрішньої пам'яті для зберігання програмного продукту та його даних. Рекомендована ємність вбудованої пам'яті - не менше 256 ГБ.
Частота процесора: Комп'ютер повинен мати процесор з достатньою частотою для швидкої обробки даних та виконання завдань програмного продукту. Рекомендована частота процесора - не менше 2.5 ГГц.
Порти та комунікації: Комп'ютер повинен мати необхідні порти та комунікаційні можливості для підключення до мережі, зовнішніх пристроїв та інтернету. Доцільно мати USB-порти, Ethernet-порт для мережевого підключення, а також можливість підключення до Wi-Fi.

3.5 Вимоги до інформаційної та програмної сумісності

Підтримка AJAX-технологій: Всі використовувані компоненти та програмне забезпечення повинні підтримувати AJAX-технології для виконання асинхронних запитів та обміну даними.
Сумісність з веб-браузерами: Дослідження повинно бути проведене на різних веб-браузерах, включаючи популярні веб-переглядачі, такі як Google Chrome, Mozilla Firefox, Microsoft Edge, і т. д.
Сумісність з операційними системами: Програмне забезпечення для проведення дослідження має бути сумісним з різними операційними системами, включаючи Windows, macOS та різні дистрибутиви Linux.
Сумісність з серверними системами: Дослідження повинно взаємодіяти з серверними системами, які підтримують AJAX-запити, включаючи сервери даних та сервери додатків.
Відкриті стандарти даних: Для обміну даними та результатами дослідження, використовуйте відкриті стандарти даних, такі як JSON чи XML, для забезпечення сумісності з іншими системами та зручності обробки даних.
Захист від перешкод: Забезпечити високий рівень захисту від перешкод та впливу сторонніх факторів, що можуть впливати на надійність дослідження та його результатів.

4. ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

До складу документації мають входити:
специфікація;
текст програми;
опис програми;
керівництво користувача для користування мобільним додатком.
Вся документація програмних додатків повинна задовольняти вимоги до програмної документації.

5. СТАДІЇ ТА ЕТАПИ РОЗРОБКИ

Таблиця 5.1 – Стадії та етапи розробки

Стадія	Зміст	Строки виконання
Технічне завдання	Постановка задачі, збір інформації, виріб та обґрунтування критеріїв розробки. Попередній вибір методів рішення задач. Визначення вимог до технічних засобів. Узгодження і затвердження технічного завдання.	04.09.23 –15.09.23
Робочий проект	Програмування та відлагодження програми.	18.09.23 – 22.09.23
	Тестування програми	25.09.23 – 27.09.23
	Розробка, узгодження і затвердження програмної документації.	28.09.23 – 29.09.23

6. ПОРЯДОК І КОНТРОЛЬ ПРИЙМАННЯ

Контроль за виконанням роботи здійснює керівник розробки доц. Андрющенко В. О.

7. ТЕХНІКО-ЕКОНОМІЧНІ ПОКАЗНИКИ

Показники та їх розрахунок не були описані у документах бо розробка додатку несе в собі навчальний характер, а не комерційний.

ДОДАТОК Б

Керівництво користувача

ЗАТВЕРДЖУЮ
Проректор
Українського
державного університету
науки і технології
Анатолій РАДКЕВИЧ

ДОСЛІДЖЕННЯ ЧАСОВОЇ ЕФЕКТИВНОСТІ AJAX- ЗАПИТІВ

ЛИСТ ЗАТВЕРДЖЕННЯ
44165850.01356-01 34 01

Завідувач кафедри КІТ
_____Вадим ГОРЯЧКІН
Керівник розробки
_____ Вадим АНДРЮЩЕНКО
Виконавець
_____ Сергій ТАРЯНИК
Нормоконтролер
_____Світлана ВОЛКОВА

2024

ЗАТВЕРДЖЕНО
44165850.01356-01 ІЗ 01

Дослідження часової ефективності Ajax- запитів

Керівництво користувача
44165850.01356-01 ІЗ 01

2024

Анотація

Документ 44165850.01356-01 ІЗ 01 «Дослідження часової фективності Ajax-запитів» входить до складу програмної документації на програму, що займається розрахунком часу та дослідженням обробки Ajax- запитів.

У даному документі представлено керівництво користувача. Програми написані на мові Swift, React Native, Flutter . Об'єм пам'яті, що займають програми комплексу, складає 3 Гб. Конфігурація комп'ютера стандартна.

1.Введення

Сучасне розуміння Інтернету істотно змінилося відтоді, як сфера інформаційних технологій почала широко поширюватися. На сьогоднішній день розвиток Інтернету зазнав значних змін, які дозволили побудувати абсолютно новий погляд на інтернет-технології та їх застосування. Сьогодні всі веб-ресурси в Інтернеті ґрунтуються на інноваційних принципах: для розроблення і створення веб-ресурсів (сайтів) використовується безліч веб-технологій. Інноваційним підходом до розроблення веб-додатків є Ajax - технологія асинхронного передавання даних.

Майже всі веб-сайти сьогодні побудовані на базі Ajax, практично кожен веб-проект має правила асинхронної передачі даних, оскільки це визначає і впливає на розвиток і використання потужних ресурсів. Ajax перекладається на асинхронний JavaScript і XML. Фактично це взаємодія двох веб-технологій, таких як мова програмування JavaScript для складання динаміки веб-сайту та xml – розширена мова розмітки для документів. Суть цього полягає в тому, що користувач продовжує працювати з мережевим ресурсом, надсилаючи дані та запити на веб-сервер без перезавантаження сторінки у вікні браузера, тоді як при класичній моделі обміну архівами без використання технології AJAX сторінка перезавантажується, коли надсилання або замінення даних. отримані в результаті взаємодії веб-сервера і клієнта.

2.Призначення та умови застосування

Функціональне призначення — проведення експерименту з фільтрування числел та відлік часу цієї операції.

Експлуатаційне призначення — дослідження скорочення часу на обробку данних.

Для забезпечення сталого функціонування програми користувачеві і програмісту необхідно дотримуватися таких умов:
- програма повинна використовуватись у приміщеннях, які відповідають умовам роботи ЕОМ [1];
- кваліфікація робочого персоналу з даною програмою повинна бути на рівні досвідченого користувача ЕОМ, операційною системою Windows, IOS. Робітник повинен бути ознайомлений з керівництвом користувача;
- програмний комплекс повинен використовуватись в приміщеннях, з наступними температурними умовами: температура 20-25 °С, відносна вологість повітря 40-60%.

ЕОМ повинен мати такі мінімальні характеристики:
- 2 ГБ оперативної пам’яті;
- 500 ГБ пам’яті на жорсткому диску;
- серверний процесор (кількість ядер 2, базова тактова частота 3.7 GHz, максимальний об’єм пам’яті 32 GB);
- клавіатура;
- миша;
- монітор;
- інтерфейс USB.

Для користувачів програмний продукт повинен використовуватись на комп’ютерах з такими вимогами:
- 2 ГБ оперативної пам’яті;
- клавіатуру;
- мишу;
- монітор.

Програмний продукт розроблений для всіх видів операційних систем Windows 10, а також для IOS. Обов’язковим до встановлення є бібліотека .NET 8 або вище.

3.Аварійні ситуації

Якщо користувач закриє програмний засіб, то при наступному запуску програми, його попередній результат не збережеться.

Якщо програмний засіб під час роботи буде поводити некоректно чи із помилкою, від користувача потребується перезапустити додаток для відновлення коректної роботи програмного засобу.

4. Опис програми

Після запуску програми додаток починає автоматично працювати. Користувач повинен вибрати яким саме методом буде відбуватись сортування (рис 1), після цього користувач задає ту кількість експериментів яку забажає (рис 2) та натиска на розділ «Почати експеримент» (рис 3).

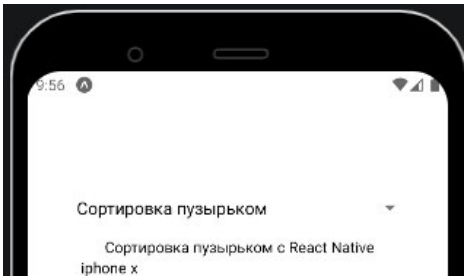


Рисунок 1- Сортування

Рисунок 3 – Вибір кількості експериментів

6. Аварійні ситуації

Якщо програмний засіб буде запускатися з пристрою який не відповідає необхідним технічним характеристикам, буде виведено відповідне повідомлення користувачеві.

Якщо користувач закрив програмний засіб, то при наступному запуску програми, його попередній результат не збережеться.

Якщо програмний засіб під час роботи буде поводити некоректно чи із помилкою, від користувача потребується перезапустити додаток для відновлення коректної роботи програмного засобу.

7.Бібліографічний список

1. HTML, JavaScript, PHP та MySQL. Джентльменський набір Web-майстра, Автор: Микола Прохоренок, Рік: 2017 ISBN: 978-5-9775-0540-6 4. MySQL
2. Бібліотека програміста, Віктор Йосипович Гольцман, рік випуску 2017, ISBN: 978-5-49807-135-0.
3. PHP and MySQL Web Development, Автори: Люк Веллінг, Лаура Томсон, Перекладач: А. Моргунов, Мови: Російська, Видавництво: Вільямс, Серія: Landmark, ISBN 978-5-8459-1574-0, 978-0-672 -32916-6; 2017 р.
4. UX-дизайн. Практичний посібник з проектування досвіду взаємодії. Унгер Р., Чендлер К. (дата звернення 21.05.2021)
5. .Джордж Шлосснейгл. Професійне програмування на PHP, 2016
6. ДСТУ 3008-95. Документація. Звіти у сфері науки та техніки. Структура та правила оформлення / Держстандарт України. - Вид. офіц. – [Чинне від 1995-02-23]. – Київ, 2007. – 86с.
7. Мішель Е. Девіс та Джон А. Філіпс. Вивчаємо PHP та MySQL, 2018

ДОДАТОК В

Текст програми

ЗАТВЕРДЖУЮ
Проректор
Українського державного
університету науки і
технології
Анатолій РАДКЕВИЧ

ДОСЛІДЖЕННЯ ЧАСОВОЇ ЕФЕКТИВНОСТІ AJAX- ЗАПИТІВ

Текст програми
44165850.01356-01 12 01

Завідувач кафедри КІТ
_____Вадим ГОРЯЧКІН
Керівник розробки
_____ Вадим АНДРЮЩЕНКО
Виконавець
_____ Сергій ТАРЯНИК
Нормоконтролер
_____Світлана ВОЛКОВА

2024

Файл RootNavigator.js

```
import * as React from 'react';
import {createStackNavigator} from '@react-navigation/stack'; import { ExperimentsScreen} from '../screens/ExperimentsScreen';

konst Stack = createStackNavigator();

export konst RootNavigator = () => {
    return (
  <Stack.Navigator initialRouteName={'experiments'}>
      <Stack.Screen name={'experiments'} component={ExperimentsScreen} />
    </Stack.Navigator>
      );
};
```

Файл ExperimentsScreen

```
export default class App extends Component {  state: {  time: string, algorithm: number,
  };
  konstructor(props: Props) {  super(props);  this.state = {  result: [],
    algorithm: 'Bubble_Sort',
    countfExperiments: NUMBER_OF_EXPERIMENTS,  phoneModel: 'iphone x',
   };
  }


  render() {
```

```
konst {countfExperiments, phoneModel} = this.state;     return (
  <>
    <StatusBar barStyle="dark-content" />
    <View style={styles.container}>
      <RNPickerSelect
        style={pickerSelectStyles}            value={this.state.algorithm}
        onValueChange={(value)     =>     this.setState({algorithm:     value})}
items={EXPERIMENTS}
      />
      <Text style={styles.startTestText}>
        {`Добро пожаловать в иследование алгоритма ${
          EXPERIMENTS_MAPPER[this.state.algorithm]
        } c React
        Native`}
      </Text>
      <Text style={styles.label}>{phoneModel}</Text>
      <Text style={styles.label}>Количество эксперементов</Text>
      <TextInput
        placeholder="Количество эксперементов"            style={styles.input}
textAlign="center"
        onChangeText={(text)    =>    this.onChangeExperimentsNumber(text)}
value={countfExperiments.toString()}            keyboardType={'numeric'}
      />
      <TouchableOpacity
        style={styles.startTestButton}            onPress={() => this.onPressGo()}
underlayColor="#fff">
        <Text style={styles.startTestText}>Начать Эксперемент</Text>
      </TouchableOpacity>
```

```
        </View>
      </>
    );
  }
}
```

Бульбашкове сортування.

```
function bubbleSort(inputArr) {
  let len = inputArr.length;  for (let i = 0; i < len; i++) {  for (let j = 0; j < len; j++) {  if (inputArr[j] > inputArr[j + 1]) {  let tmp = inputArr[j];  inputArr[j] = inputArr[j + 1];  inputArr[j + 1] = tmp;
      }
    }  }  return inputArr;
}
```

- факторіал

```
function factorial(n) {  var result = 1;  while (n) {  result *= n--;
  }  return result;
}
```

- сортування вставками

```
konst insertionSort = (inputArr) => {  let length = inputArr.length;  for (let i = 1; i < length; i++) {  let key = inputArr[i];  let j = i - 1;
    while (j >= 0 && inputArr[j] > key) {  inputArr[j + 1] = inputArr[j];  j = j - 1;  }
    inputArr[j + 1] = key;
  }  return inputArr;
};
```

 бінарний пошук

```
konst binarySearch = (array, target) => {   let startIndex = 0;   let endIndex = array.length - 1;   while (startIndex <= endIndex) {
    let middleIndex = Math.floor((startIndex + endIndex) / 2);

    if (target === array[middleIndex]) {
      return console.log('Target was found at index ' + middleIndex);
    }
    if (target > array[middleIndex]) {
      console.log('Searching the right side of Array');      startIndex = middleIndex + 1;
    }
    if (target < array[middleIndex]) {
      console.log('Searching the left side of array');      endIndex = middleIndex - 1;
    } else {
      console.log('Not Found this loop iteration. Looping another iteration.');
    }
  }
  console.log('Target value not found in array'); };
```

- послідосність Фібоначі

```
function fibonacci(num) {   var a = 1,     b = 0,     temp;   while (num >= 0) {
temp = a;     a = a + b;     b = temp;     num--;
  }   return b;
}
```

- алгоритм Гауса-Лежандра

```
function gaussLegendre() {   let a = 1.0;   let b = 1.0 / Math.sqrt(2);   let t = 1.0 / 4.0;   let p = 1.0;
```

```
for (let i = 0; i < 100; i++) {     let aNext = (a + b) / 2;     let bNext = Math.sqrt(a * b);     let tNext = t - p * Math.pow(a - aNext, 2);     let pNext = 2 * p;
 a = aNext;     b = bNext;     t = tNext;     p = pNext;
}
return Math.pow(a + b, 2) / (4 * t); }
```

- алгоритм Борейна

```
function borwein() {
 let ak = 6.0 - 4 * Math.sqrt(2);   let yk = Math.sqrt(2) - 1.0;   var ak1;var yk1;
 for (let i = 0; i < 100; i++) {     yk1 =
   (1 - Math.pow(1 - yk * yk * yk * yk, 0.25)) /       (1 + Math.pow(1 - yk * yk * yk * yk, 0.25));     ak1 =
    ak * Math.pow(1 + yk1, 4) -
    Math.pow(2, 2 * i + 3) * yk1 * (1 + yk1 + yk1 * yk1);     yk = yk1;ak = ak1;
 }   return ak;
}
```

Функція tester.

```
onExperimentStart(algoritm) {
   konst {countfExperiments} = this.state;     konst experimentsResult = [];
    for (let j = 0; j < countfExperiments; j += 1) {      let startTime = Timing.now(); algoritm ();
     let endTime = Timing.now();          let iterTime = endTime - startTime; experimentsResult.push(iterTime);
    }
   this.sendDatum(experimentsResult);
  }
```

Функція sendDatum.

```
sendDatum(datum) {
   konst {phoneModel} = this.state;
```

```
fetch(ENDPOINT, {  method: 'POST',  headers: {
  Accept: 'Aplication/json',
  'Content-Type': 'Aplication/json',
 },
 body: JSON.stringify({  phoneModel,  method: METHOD,
platform: PLATFORM,  datum,
 }),
})
 .then((response) => response.json())
 .then((json) => {  console.log('res', json);
 })
 .catch((error) => {  console.error(error);
 });
}
```

Flutter

```
class _HomeState extends State<Home> {
 String gaussLegendreTime = '';  String borweinTime = '';  int
countfExperiments = 100;
 String experiment = 'Bubble_Sort';
 @override
 Widget build(BuildContext context) {
  return MaterialApp(  home: Scaffold(  body: Center(  child:
Column(
      crossAxisAlignment:  CrossAxisAlignment.center,
mainAxisAlignment:  MainAxisAlignment.center,  children: <Widget>[
new DropdownButton<String>(  value: experiment,
      items: EXPERIMENTS.map((String value) {  return new
```

```
DropdownMenuItem<String>( value: value, child: new Text(value),
                    ); }).toList(), onChanged: (value) { setState(() { experiment = value;
                    });
                  },
                ),
                Text(
                  'Добро пожаловать в иследование алгоритма ${experiment} с Flutter'), TextField(
                  decoration: InputDecoration( labelText:
                      'Введите количество эксперементов. По умолчанию
        ${countfExperiments.toString()}'), keyboardType: TextInputType.number, onChanged: (text) {
                    countfExperiments = int.parse(text);
                  },
                ),
                RaisedButton(
                  child: Text('Начать эксперемент'), onPressed: onPressed()
              ],
            ),
          ),
        ),
      );
    }
```

Бульбашкове сортування.

```
runBubble_SortTest(array) {  int lengthOfArray = array.length;  for (int i =
```

```
0; i < lengthOfArray - 1; i++) {       for (int j = 0; j < lengthOfArray - i - 1; j++) {        if (array[j] > array[j + 1]) {          // Swapping using temporary variable          int temp = array[j];          array[j] = array[j + 1];          array[j + 1] = temp;
                }
              }     }
            return (array);
          }
```

- факторіал

```
function factorial(n) {   var result = 1;   while (n) {     result *= n--;
  }
  return result;
}
```

- сортування вставками

```
var length = inputArr.length;      for (var i = 1; i < length; i++) {        var key = inputArr[i];       var j = i - 1;
            while (j >= 0 && inputArr[j] > key) {          inputArr[j + 1] = inputArr[j]; j = j - 1;       }
            inputArr[j + 1] = key;
          }     return inputArr;
        };
```

- Бінарний пошук

```
var startIndex = 0;       var endIndex = array.length - 1;       while (startIndex <= endIndex) {
            var middleIndex = ((startIndex + endIndex) / 2).round();         if (target == array[middleIndex]) {         return middleIndex;
            }
```

```
        if (target > array[middleIndex]) {        startIndex = middleIndex + 1;
        }
        if (target < array[middleIndex]) {        endIndex = middleIndex - 1;
        }
      };
```

- послідосність Фібоначі

```
    runFibonacci(num) {     var a = 1,        b = 0,        temp;
      while (num >= 0) {       temp = a;       a = a + b;       b = temp;       num--;     }
return b;
     }
```

- алгоритм Гауса-Лежандра

```
    double gaussLegendre(int iterations) {       double a = 1.0;       double b = 1.0 /
math.sqrt(2);     double t = 1.0 / 4.0;
      double p = 1.0;

      for (int i = 0; i < iterations; i++) {       double aNext = (a + b) / 2;       double
bNext = math.sqrt(a * b);
        double tNext = t - p * math.pow(a - aNext, 2.0);       double pNext = 2 * p;
a = aNext;       b = bNext;       t = tNext;       p = pNext;
      }
      return math.pow(a + b, 2) / (4 * t);   }
```

- Алгоритм Борейна

```
    double borwein(int k) {
      double ak = 6.0 - 4 * math.sqrt(2);     double yk = math.sqrt(2) - 1.0;     double
ak1 ;     double yk1 ;
      for (int i = 0; i < k; i++) {
        yk1 = (1 - math.pow((1 - yk * yk * yk * yk),(0.25)))/(1 + math.pow((1 - yk *
yk * yk * yk),(0.25)));
```

```
            //ak1 = ak * math.pow((1 + yk1), 4) - math.pow(2, 2 * i + 3) * yk1 * (1 + yk1
+ yk1 * yk1);       ak1 = ak * math.pow((1 + yk1), 4.0) - math.pow(2.0, (2 * i + 3.0)) *
yk1 * (1 + yk1 + yk1 * yk1);       yk = yk1;       ak = ak1;
            }     return ak;
           }
        pressGaussLegendreButton(algoritm)     async     {                    List<String>
resultOfExperiment = [];       for (int i = 0; i < countfExperiments; i += 1) {         var
stopwatch = new Stopwatch();      stopwatch.start();      algoritm ();      stopwatch.stop();
             var duration = (stopwatch.elapsedTicks / stopwatch.frequency) * 1000;
resultOfExperiment.add((duration).toString());
            }
            avait sendDatum(resultOfExperiment);     setState(() {
              gaussLegendreTime = resultOfExperiment.toString();
            });
           }
        NSAplicationDelegate.
        struct ContentView:view.
        State private var time: String = "0.0";
        State private var countfExperiments: String = "100";
        State private var method: String = "Factorial";
        var body:aview{Menu("Дії") {{ { Menu("Дії")
                Button("Bubble_Sort", action: setBubble_Sort)
                Button("Insertion_Sort", action: setInsertion_Sort)
                Button("Binary_search", action: setBinary_search)
                Button("Fibonacci", action: setFibonacci)                Button("Factorial",
action: setFactorial)
              }
              Text("Добро пожаловать в иследование алгоритма " + methood + " с
```

```
Swift")
            Text(countfExperiments)
            TextField("Введите количество эксперементов", text:
$countfExperiments)        Button(action:{
                var convertedCount: Int = Int(countfExperiments) ?? 100        if(method
== "Bubble_Sort") {
                    Bubble_Sort(n:convertedCount)
                }
                if(method == "Insertion_Sort") {
                    Insertion_Sort(n:convertedCount)
                }
                if(method == "Binary_search") {
                    Binary_search(n:convertedCount)
                }
                if(method == "Fibonacci") {
                    Fibonacci(n:convertedCount)
                }
                if(method == "Factorial") {
                    Factorial(n:convertedCount)
                }
            }){
                Text("Начать эксперемент")
            }
            Text(time)

        }
```

Бульбашкове сортування

```
func runBubble_Sort(array: Array<Int>) -> Void {    var datumSet = array       let
```

```
last_positione = datumSet.count - 1 var swap = true while swap == true { swap = false for i in 0..<last_positione { if datumSet[i] > datumSet[i + 1] { let temp = datumSet [i + 1] datumSet [i + 1] = datumSet[i] datumSet[i] = temp
                                swap = true
                }
            }
        }
    }
```

- факторіал

```
func runFactorial(n: Int) -> Double { if n == 0 { return 1 } var a: Double = 1 for i in 1...n { a *= Double(i)
    } return a
}
```

- сортування вставками

```
func runInsertion_Sort(a: Array<Int>) -> Void { guard a.count > 1 else { return }
                var b = a for i in 1..<b.count { var y = i let temp = b[y] while y > 0 && temp < b[y - 1] { b[y] = b[y - 1] y -= 1 } b[y] = temp } return
}
```

Функція runExperiment

```
func runExperiment(experiment: Func) -> Void { var j: Int = 0;
        var convertedCount: Int = 100 var numbers: [Double] = [] var numbersString: [String] = [] while((j < n) == true) {
            let startTime = DispatchTime.now(); experiment (n: convertedCount) let endTime = DispatchTime.now();
            let iterTime = Double(endTime.uptimeNanoseconds-
```

```
startTime.uptimeNanoseconds);          numbers.append(iterTime/1000000)
                numbersString.append(String(iterTime/1000000))          j = j+1
            }
            time = numbersString.joined(separator: ",");
            sendDatum(_datum: numbers, method: method, count: countfExperiments);
```

Функція addValuesToTable

```
addValuesToTable: async (googleScheet, platform, section, datum) => {    konst
{     startLetter,     startNumber,     endLetter,     endNumber,
        } = getParsedWorkingSection(section);    console.log("Start  of add values");
        if (startLetter && startNumber && endLetter && endNumber) {
          konst letterToWrite = Scheets.getLetterByPlatform(platform, startLetter);

          for (let i = 0; i < datum.length; i = i + 1) {       konst numberToWrite = i + 1
+ OFFSET;
            konst numberCellA1 = `${letterToWrite}${numberToWrite}`;          konst
numberCell          =          avait          googleScheet.getCellByA1(numberCellA1);
Scheets.createAlignedCell(numberCell,  datum[i]);                console.log(`Value
${numberCellA1}  added`);
          }
          konst average =
            _.sumBy(datum, (item) => {          if (_.isNumber(item)) {          return
item;
              }

              return parseFloat(item);
            }) / datum.length;
```

```
            console.log("Average  value added");
            konst  averageCellA1  =  `${letterToWrite}${endNumber}`;  konst
averageCell  =  avait  googleScheet.getCellByA1(averageCellA1);
Scheets.createAlignedBoldCell(averageCell, average);  console.log("End  of add
values");
          } else {
            throw Error("Cannot work with this section");
          }
        },
```

Функція describe

```
describe(Buble Sorting Test, () => {
    it('Array should be sorted with bubble sort, () => {
konst arrayToTest = [5,6,7,3,6,0,3,2] konst expectedArray = [0,2,3,3,5,6,6,7]
konst result = bubbleSort(arrayToTest);
            expect(result.toEqual(expectedArray))
},
it('Array should be sorted with inservion sort, () => { konst arrayToTest =
[5,6,7,3,6,0,3,2]  konst  expectedArray  =  [0,2,3,3,5,6,6,7]  konst  result  =
insertionSort(arrayToTest);
          expect(result.toEqual(expectedArray)) } });
```

ДОДАТОК Г

Специфікація

ЗАТВЕРДЖУЮ
Проректор
Українського державного
університету науки і
технології
Анатолій РАДКЕВИЧ

ДОСЛІДЖЕННЯ ЧАСОВОЇ ЕФЕКТИВНОСТІ AJAX- ЗАПИТІВ
Специфікація
ЛИСТ ЗАТВЕРДЖЕННЯ
44165850.01356-01-ЛЗ

Завідувач кафедри КІТ
_____Вадим ГОРЯЧКІН
Керівник розробки
_____Вадим АНДРЮЩЕНКО
Виконавець
_____Сергій ТАРЯНИК
Нормоконтролер
_____Світлана ВОЛКОВА

2024

ЗАТВЕРДЖЕНО
44165850.01356-01

ДОСЛІДЖЕННЯ ЧАСОВОЇ ЕФЕКТИВНОСТІ AJAX- ЗАПИТІВ

Специфікація

Листів 2

2024

Позначення	Найменування	Примітки
	Документація	
44165850.01356-01-ЛЗ	Лист затвердження	
44165850.01356-01 12 01-ЛЗ	Лист затвердження	
44165850.01356-01 12 01	Текст програми	
44165850.01356-01 31 01-ЛЗ	Лист затвердження	
44165850.01356-01 13 01	Опис програми	
44165850.01356-01 І3 01-ЛЗ	Лист затвердження	
44165850.01356-01 І3 01	Керівництво користувача	

ДОДАТОК Д

ЗАТВЕРДЖУЮ
Проректор
Українського державного
університету науки і
технології
Анатолій РАДКЕВИЧ

ДОСЛІДЖЕННЯ ЧАСОВОЇ ЕФЕКТИВНОСТІ AJAX- ЗАПИТІВ
Специфікація
СПИСОК ДЖЕРЕЛ
44165850.01356-90-ЛЗ

Завідувач кафедри КІТ
_____Вадим ГОРЯЧКІН
Керівник розробки
_____Вадим
АНДРЮЩЕНКО
Виконавець
_____Сергій ТАРЯНИК
Нормоконтролер
_____Світлана ВОЛКОВА

2024

Характеристика джерела	
Книги одного, двох або трьох авторів	1. HTML & XHTML: Definitive Guide. Автори: Чак Маскіано, Біл Кеннеді. Перекладач: Сергій Іноземцев, Видавництво: Символ-Плюс, ISBN 978-5-93286-104-2, 5-93286-104-5, 0-596-52732-2; 2017 р. 2. HTML, JavaScript, PHP та MySQL. Джентльменський набір Web-майстра, Автор: Микола Прохоренок, Рік: 2017 ISBN: 978-5-9775-0540-6 4. MySQL 3. Бібліотека програміста, Віктор Йосипович Гольцман, рік випуску 2017, ISBN: 978-5-49807-135-0. 4. PHP and MySQL Web Development, Автори: Люк Веллінг, Лаура Томсон, Перекладач: А. Моргунов, Мови: Російська, Видавництво: Вільямс, Серія: Landmark, ISBN 978-5-8459-1574-0, 978-0-672 -32916-6; 2017 р. 5. UX-дизайн. Практичний посібник з проектування досвіду взаємодії. Унгер Р., Чендлер К. (дата звернення 21.05.2021) 6. Джордж Шлосснейгл. Професійне програмування на PHP, 2016 7. Мішель Е. Девіс та Джон А. Філіпс. Вивчаємо PHP та MySQL, 2018 8. Микола Прохоренок. HTML, JavaScript, PHP та MySQL. Джентльменський набір Web-майстра, 2017 9. Оригінальна назва: Pro PHP and jQuery. Автор: Джейсон Ленгсторф. Видавництво: Вільямс. Рік 2017. ISBN 978-5-8459-1693-8. 10. Сучасний підручник JavaScript - https://learn.javascript.ru/ 11. 17. Теорія систем та системний аналіз в управлінні

	організаціями: Довідник. / За ред. В.М. Волковій та А.А. Ємельянова. - М.: Фінанси та статистика, 2016 - 848 с. 12. Вільям Стейнмець, Браян Вард. 75 готових рішень для вашого сайту на PHP, 2018 13. Бабенко В.В., Гольчевський Ю.В. Вибір мов програмування та засобів проектування для навчання спеціалістів за напрямом «Прикладна інформатика». Прикладна інформатика. 2013. №4 (46). С. 43-45. 14. Ajax у дії Автор: Дейв Крейн, Ерік Паскарелло, Даррен Джеймс
Книги чотирьох авторів	1. Бусигін Б.С., Дивізінюк М.М., Коротенко Г.М., Коротенко Л.М. Введення у сучасну інформатику. Підручник - Севастополь: Видавництво СНУЯЕіП, 2005. - 644 с.
Книги під назвою	1. Бретт Маклафін. Вивчаємо Ajax = Head Rush Ajax. - СПб.: Пітер, 2007 2. Дейв Крейн, Ерік Паскарелло, Даррен Джеймс. AJAX у дії: технологія - Asynchronous JavaScript and XML = Ajax in Action. - М.: Вільямс, 2006. - С. 640
Щорічники	1. HTML та XHTML. Детальний посібник ISBN: 5-93286-104-5 752 сторінки 752, грудень 2012 Символ-Плюс 2. Розробка односторінкових веб-додатків. 2017.
Стандарти	1. ДСТУ 3008-95. Документація. Звіти у сфері науки та техніки. Структура та правила оформлення / Держстандарт України. - Вид. офіц. – [Чинне від 1995-02-23]. – Київ, 2007. – 86с. 2. Офіційна документація LESS – https://lesscss.org

	3. Офіційна документація Node.js – https://nodejs.org/uk/
Статті з журналів та періодичних збірників	1. Інформатика: Базовий курс. С.В. Симонович та ін. СПб.: . 2006