УДК 004.42 DOi 10.30837/ bi.2025.1(102).06

О.С. Назаров[1], С.А. Биковська[2], Н.В. Назарова[3]

[1] Харківський національний університет радіоелектроніки, м. Харків, Україна, oleksii.nazarov1@nure.ua, ORCID iD: 0000-0001-8682-5000

[2] Харківський національний університет радіоелектроніки, м. Харків, Україна, sofiia.bykovska@nure.ua

[3] Харківський національний університет радіоелектроніки, м. Харків, Україна, nataliia.nazarova@nure.ua, ORCID iD: 0009-0007-7816-7088

ДОСЛІДЖЕННЯ МЕТОДІВ АНІМАЦІЙ ВИКОРИСТОВУЮЧИ ФРЕЙМВОРК FLUTTER

Стаття присвячена дослідженню методів реалізації анімацій у мобільних застосунках на основі фреймворку Flutter з метою покращення користувацького досвіду. У межах роботи було проаналізовано сучасні підходи до створення анімованих інтерфейсів, зокрема імпліцитні (implicit) та експліцитні (explicit) анімації, а також використання анімованих віджетів і кастомних трансформацій. Особлива увага приділялася не лише технічним аспектам реалізації, а й впливу анімацій на сприйняття інтерфейсу користувачем, емоційне залучення та інклюзивність цифрового продукту. Оцінювання ефективності впроваджених рішень здійснювалося шляхом аналізу продуктивності, плавності відтворення анімацій, відповідності принципам Material Design, а також доступності для користувачів з особливими потребами. У дослідженні застосовувалися як інструменти профілювання Flutter, так і експертне оцінювання взаємодії користувача із застосунком. Результати дослідження показали, що вдумливе використання анімацій не лише покращує естетичне сприйняття інтерфейсу, але й сприяє інтуїтивному розумінню навігації, підвищенню залученості користувача та загальній ефективності мобільного рішення.

DEVTOOLS, FLUTTER, MATERIAL DESIGN, UX, АДАПТИВНІСТЬ, ЕКСПЛІЦИТНІ АНІМАЦІЇ, ІМПЛІЦИТНІ АНІМАЦІЇ, ІНКЛЮЗИВНІСТЬ, ПРОДУКТИВНІСТЬ ІНТЕРФЕЙСУ, ПРОФІЛЮВАННЯ ПРОДУКТИВНОСТІ

O. Nazarov, S. Bykovska, N. Nazarova. Research on animation methods using the Flutter framework. The article focuses on the study of methods for implementing animations in mobile applications using the Flutter framework, aiming to enhance the user experience. The paper analyzes modern approaches to creating animated interfaces, including implicit and explicit animations, as well as the use of animated widgets and custom transformations. Particular attention was paid not only to the technical aspects of implementation but also to the impact of animations on user experience, emotional engagement, and inclusiveness of the digital product. The effectiveness of the implemented solutions was evaluated by analyzing the performance, smoothness of animations, compliance with Material Design principles, and accessibility for users with special needs. The study used both Flutter profiling tools and expert evaluation of user interaction with the application. The results of the study showed that the thoughtful use of animations not only improves the aesthetic perception of the interface but also contributes to an intuitive understanding of navigation, increased user engagement, and overall effectiveness of the mobile solution.

DEVTOOLS, FLUTTER, MATERIAL DESIGN, UX, RESPONSIVENESS, EXPLICIT ANIMATIONS, IMPLICIT ANIMATIONS, INCLUSIVITY, INTERFACE PERFORMANCE, PERFORMANCE PROFILING

Вступ

У сучасну цифрову епоху користувачі очікують не лише функціонального додатку, але й інтуїтивно зрозумілого, приємного та простого у використанні інтерфейсу. Анімація відіграє важливу роль у цьому випадку: вона дозволяє краще зрозуміти логіку навігації, формує ілюзію плавності та знайомства, і навіть допомагає встановити емоційний зв'язок з продуктом. При правильному використанні анімація може зробити інтерфейс «живим» – інтерактивним, інформативним та емоційним. Водночас, надмірна або неякісна анімація може мати негативні наслідки, такі як інформаційне перевантаження, роздратування або погіршення продуктивності додатку. Flutter, відомий крос-платформний набір для розробки від Google, пропонує розробникам потужні інструменти для створення анімованих користувацьких інтерфейсів. Він підтримує легке включення базової неявної анімації, яка автоматично запускається при зміні стану віджету, а також розширювані явні механізми, де розробник зберігає контроль над кожним кроком у процесі анімації. Це відкриває величезні можливості, але водночас і лякає: як досягти анімації, яка буде плавною, але не млявою? Очевидною, але не нав'язливою? Доступної для всіх користувачів, включаючи тих, хто має вади зору або чутливість до руху? У цьому дослідженні розглядаються різні техніки реалізації анімації в середовищі Flutter, їхній вплив на користувацький досвід, а також способи оцінки ефективності анімації з точки зору продуктивності, зручності та доступності. Ми порівнюємо приклади як простих, так і складних технік анімації, використовуємо інструменти профілювання (наприклад, Flutter DevTools) і робимо акцент на дотриманні

принципів інклюзивного дизайну, щоб інтерфейс був не тільки естетично привабливим, але й зручним для якомога ширшої аудиторії. Мета цієї книги – не просто проілюструвати технічні особливості Flutter, а досягти рівноваги між зовнішнім виглядом і функціональністю, привабливістю і продуктивністю, «вау-фактором» і реальною зручністю для користувача.

1. Опис предметної області

У фокусі цього дослідження – методи інтеграції анімації в мобільні додатки з конкретним застосуванням механізмів розробки неявної та явної анімації за допомогою фреймворку Flutter. Матеріал охоплює не тільки вже існуючі анімовані віджети (наприклад, AnimatedContainer, AnimatedOpacity, Hero), але й елементи, які використовуються для ручного управління циклом анімації, тобто AnimationController, Tween, AnimationBuilder тощо [1]. Основна увага в цьому дослідженні зосереджена на вивченні впливу різних форм анімації на досвід користувача при взаємодії з мобільним інтерфейсом, насамперед з точки зору продуктивності, простоти навігації та доступності цифрового середовища. Пріоритетним завданням є пристосування анімації до потреб користувачів з різним сприйняттям, наприклад, чутливістю до руху, при дотриманні стандартів доступності інтерфейсу відповідно до сучасних керівних принципів. Основна мета дослідження – вивчити та проаналізувати ефективність анімаційних рішень, розроблених за допомогою Flutter, з акцентом на поєднанні естетики, покращеної продуктивності та зручності інтерфейсу для широкого кола користувачів. У дослідженні використовувався Flutter DevTools для аналізу продуктивності анімації, а також якісні показники залучення користувачів на основі принципів UI/UX-дизайну [2]. Це дозволило нам поглиблено дослідити вплив анімації на загальний користувацький досвід мобільного додатку.

2. Аналіз задач та методів програмної оптимізації

Метою цього дослідження є вивчення ключових методів реалізації анімації у фреймворку Flutter та їхнього впливу на користувацький досвід. Воно спрямоване на виявлення найкращих практик у досягненні плавного, зручного та доступного інтерфейсу, не впливаючи на рівень продуктивності програми. У сучасному дизайні мобільних додатків все більше уваги приділяється не лише поверхневому графічному дизайну програми, але й емоційному досвіду, який користувачі отримують під час її використання. Анімація в цьому контексті є не просто декоративним елементом, а фактично життєво важливим компонентом, який сприяє покращенню навігації та полегшує розуміння дій користувача. Flutter пропонує два основні методи генерації анімації: неявні анімації, які автоматично відбуваються у відповідь на зміну властивостей віджету, та явні анімації, в яких стан, час та поведінка анімації керуються розробником. Неявна анімація – це простий метод створення швидких візуальних ефектів, включаючи колір, розмір або прозорість, і використовується за допомогою таких віджетів, як AnimatedContainer, AnimatedOpacity або AnimatedAlign. Явна анімація забезпечує повний контроль над анімаційними операціями за допомогою класів AnimationController, Tween, CurvedAnimation та інших. Одним з найважливіших аспектів роботи з анімацією є її вплив на продуктивність інтерфейсу. Некоректне використання анімованих об'єктів може призвести до перевантаження потоків інтерфейсу, фризів і зниження FPS, що особливо критично для пристроїв з обмеженими ресурсами. Тому Flutter надає інструменти профілювання, включаючи Performance Overlay, Timeline і DevTools, які дозволяють виявляти «важкі» віджети, аналізувати використання кадрів і оптимізувати відтворення анімації. Поряд з технічною реалізацією систем важливо звертати увагу на психологічну обробку, пов'язану зі сприйняттям руху. Дослідження підтвердили, що анімація може зменшити когнітивне навантаження, полегшити навігацію між елементами інтерфейсу та підвищити емоційну залученість у додаток. З іншого боку, занадто довгі або кричущі анімації, або ті, що не мають очевидного функціонального виправдання, мають зворотний ефект, викликаючи роздратування або навіть розгубленість. Це особливо важливо з огляду на принципи інклюзивного дизайну, які передбачають створення інтерфейсів, доступних для людей з різною чутливістю до руху або когнітивними порушеннями.

У фреймворку Flutter є конфігурації для підтримки обмежень середовища користувача, такі як MediaQuery.of(context).disableAnimations та AccessibilityFeatures.disableAnimations [1]. Завдяки їх використанню додаток може адаптуватися в реальному часі до потреб користувача, відключаючи складні анімаційні ефекти, що робить його більш інклюзивним і поважає індивідуальні уподобання.

Отже, ефективне використання анімації у Flutter – це делікатний баланс трьох основних факторів: вибір правильного набору технічних прийомів (неявних чи явних), постійна перевірка ефективності за допомогою інструментів профілювання та дотримання рекомендацій щодо доступності. Врахування всіх цих факторів разом дозволяє створювати візуально інтуїтивно зрозумілі мобільні інтерфейси, які водночас є ефективними, простими і доступними для різних груп користувачів.

3. Методи реалізації анімацій у Flutter з урахуванням продуктивності та інклюзивності

Використання хорошої анімації в мобільних додатках є важливим аспектом UI/UX дизайну сьогодні. Вони можуть допомогти користувачеві краще відчути інтерфейс, полегшити навігацію та встановити емоційний зв'язок між користувачем і продуктом. Але якщо ці анімації погано оптимізовані, вони можуть спричинити низьку продуктивність, сповільнити взаємодію та погіршити доступність для деяких груп користувачів. Таким чином, у сучасній практиці мобільної розробки мова йде не лише про створення красивих анімацій, але й про дотримання принципів інклюзивного дизайну та юзабіліті.

Flutter має величезний інструментарій для розробки анімації, який включає деякі з найбільш важливих категорій: неявна анімація, явна анімація, створення кастомної анімації та сумісність з апаратними засобами виконання. У цій доповіді ми розглянемо стандартні підходи до розробки анімації у Flutter, їхній вплив на продуктивність користувацьких інтерфейсів, а також те, як сприяти інклюзивності.

Flutter надає ряд практичних віджетів для неявної анімації, таких як AnimatedContainer, AnimatedOpacity та AnimatedAlign. Ці віджети дозволяють легко реалізувати прості переходи без необхідності створювати спеціалізований контролер анімації. Вони реагують на зміну стану, автоматично анімуючи відповідні властивості.

Такий підхід менш ресурсоємний і зручний для початкової розробки, але має обмеження в гнучкості. Неявна анімація добре підходить для забезпечення плавних переходів у простих елементах інтерфейсу, таких як кнопки, блоки та контейнери. Крім того, завдяки інтеграції з можливостями платформи Flutter, ці анімації забезпечують високий рівень продуктивності навіть на пристроях середнього класу.

Явні анімації з ручним керуванням використовуються для більш складних сценаріїв: AnimationController, Tween, CurvedAnimation, AnimatedBuilder. Такий підхід дозволяє задавати тривалість, криву згладжування, цикл повторення, запускати або зупиняти анімацію тощо [4, 5].

Явні анімації вимагають делікатного поводження з життєвими циклами віджетів і профілюванням, оскільки їх неправильне використання може призвести до витоку пам'яті або низької частоти кадрів в секунду. Однак вони надають більше свободи для створення взаємодій, необхідних для кастомних елементів, анімації переходів між екранами або складних інтерактивних сцен. Вбудований рушій Hero забезпечує плавні переходи між екранами для стандартних елементів інтерфейсу. Ця функція допомагає підтримувати контекст з точки зору користувача та створювати відчуття навігації. Однак важливо регулювати як розмір, так і кількість одночасно анімованих об'єктів, щоб уникнути візуального перевантаження інтерфейсу.

Flutter дотримується системних обмежень щодо анімації, які встановлюються користувачами за допомогою налаштувань доступності. Наприклад, властивість MediaQuery.of(context).disableAnimations посилається на вимогу вимкнути анімацію для людей, чутливих до руху [9]. Ігнорування цього параметра може порушити комфорт користувачів з нейросенсорними особливостями або захворюваннями, пов'язаними з вестибулярним апаратом. Інтеграція цієї практики є ознакою відповідального та інклюзивного дизайну.

Профілювання продуктивності є необхідною умовою для створення складних анімацій. Flutter має ряд інструментів, включаючи DevTools, Performance Overlay та Timeline, які можна використовувати для визначення ділянок коду, які перевантажені. Типовими причинами втрати продуктивності є надто складні збірки в методах build(), відсутність адекватних механізмів кешування або перебудова всього дерева віджетів, а не конкретних цільових компонентів.

Оптимальними практиками є:

– використання RepaintBoundary для ізоляції анімованих елементів;

– анімація лише того, що дійсно змінюється;

– мінімізація кількості викликів setState() під час анімації.

Інклюзивний підхід до анімації охоплює розробку інтерфейсів, які враховують вимоги різних груп користувачів. Це включає в себе:

– контрастність і чіткість візуальних переходів;

– здатність адаптуватися до обмеженого руху;

– анімовані підказки для навігації по складному інтерфейсу;

– врахування принципів універсального дизайну та рекомендацій WAI-ARIA щодо рухомого контенту [9].

Отже, використання анімації в середовищі Flutter передбачає об'єднання креативного дизайну та технічної зручності використання. Балансуючи між продуктивністю, доступністю та гнучкістю, можна розробити інтерфейси, які будуть не лише естетично привабливими, але й зручними, інтуїтивно зрозумілими та стійкими на широкому спектрі пристроїв.

4. Аналіз застосування анімацій у Flutter – застосунках з урахуванням продуктивності та доступності

Використання анімації в мобільних додатках стало поширеною практикою для покращення візуальної взаємодії, мінімізації когнітивного навантаження та забезпечення інтуїтивної взаємодії з користувачем. У фреймворку Flutter анімація може бути реалізована як неявно (наприклад, AnimatedContainer,

AnimatedOpacity та AnimatedAlign), так і явно (наприклад, AnimationController, Tween та AnimatedBuilder). Незважаючи на те, що анімовані інтерфейси красиві, необхідно вивчити їхній вплив на продуктивність системи, швидкість взаємодії та зручність використання для людей з особливими потребами. Щоб оцінити цей вплив, ми розглянули зразок веб-додатку Flutter (на основі PWA), що містить прості неявні та адаптовані анімації. Вимірювання проводилося за допомогою Google Lighthouse – інструменту, що дозволяє оцінити як продуктивність додатку, так і дотримання правил доступності та найкращих практик [7].

На рис. 1 представлено зведену інформацію про показники, зібрані під час процедури оцінювання.

Рис. 1. Загальні метрики продуктивності та доступності

Результати тесту підтверджують, що продуктивність отримала рекордно високий бал 99, що свідчить про відсутність зависань, затримок рендерингу та коректну роботу графічного процесора навіть за умови одночасного використання анімації. Оцінка «Доступність» у 79 балів демонструє певну часткову невідповідність стандартам WCAG 2.1, що потенційно може спричинити перешкоди для людей з порушеннями зору, моторики або нейросенсорних функцій. Оцінка «Найкращі практики» в 75 балів свідчить про наявність невеликих прогалин у конфігурації JavaScript API, але оцінка SEO в 100 балів підкреслює відмінну семантичну структуру веб-сторінок. Щоб краще зрозуміти, як анімація пов'язана з продуктивністю користувацького інтерфейсу, на рис. 2 розглянуто ключові показники Web Vitals – показники, які відображають безпосередній досвід користувача [3].

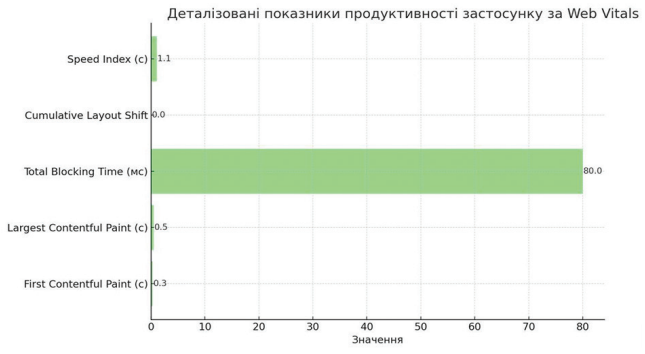

Рис. 2. Деталізовані показники продуктивності застосунку

Результати:

– FCP (First Contentful Paint): 0,3 секунди – початковий рендеринг контенту відбувається миттєво, що дозволяє уникнути відчуття «порожнього екрану»;

– LCP (Largest Contentful Paint): 0,5 секунди – основний контент завантажується швидко, що особливо важливо для веб-сторінок, які містять велику кількість медіа-об'єктів. Загальний час блокування (TBT): 80 мс – цей показник вказує на те, що взаємодія користувача з інтерфейсом не відкладається фоновими процесами або скриптами.

– CLS (Cumulative Layout Shift): 0 – елементи інтерфейсу стабільні під час завантаження, що свідчить про якісну верстку.

– Індекс швидкості: 1.1 секунди – швидкість візуального рендерингу стабільно висока.

Представлені факти (див. рис. 3) демонструють, що використання анімації, коли вона виконується відповідно до визначених найкращих практик, може призвести до дійсно високих результатів продуктивності (див. рис. 4) [8, 10]. Зокрема, використаний додаток використовував RepaintBoundary для ізоляції анімованих областей, мінімізував виклики setState() під час рендерингу та використовував лише легкі криві (Curves.easeInOut, Curves.linear).

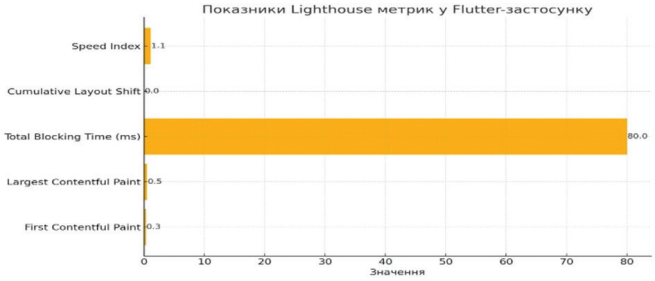

Рис. 3. Показники Lighthouse метрик

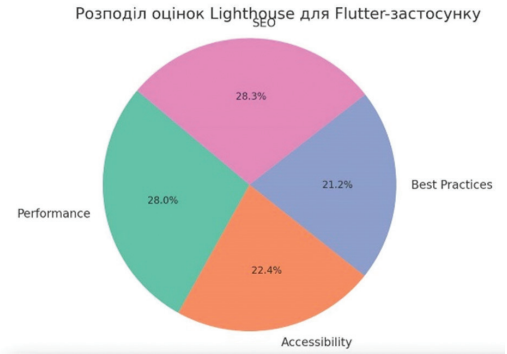

Рис. 4. Розподіл оцінок Lighthouse

Окрім вимірювання ефективності, важливо вимірювати доступність. Найчастіше зустрічаються такі проблеми: Відсутність належних альтернативних описів (alt) для анімованих кнопок і банерів.

Обмеження масштабування, встановлене директивою user-scalable=«no», робить додаток недоступним для людей з порушеннями зору.

Відсутність достатнього контрасту в деяких анімованих елементах (наприклад, текст на кольоровій кнопці), що не відповідає вимогам WCAG.

Ці недоліки характерні для додатків, де анімація досягається за допомогою кастомних віджетів, які не мають належної семантики та достатньої підтримки атрибутів ARIA.

Для сприяння інклюзивності розробникам рекомендується використовувати віджети семантики для визначення анімованого контенту, повністю протестувати додаток з увімкненими функціями системної доступності (TalkBack, VoiceOver), додати логіку для вимкнення анімації, коли MediaQuery.of(context). disableAnimations оцінюється як true.

Отримані результати свідчать про те, що фреймворк Flutter підтримує надійний та універсальний інструментарій для створення анімації з прийнятними технічними показниками продуктивності. Водночас, щоб зробити користувацький досвід справді якісним, необхідно не лише стежити за такими показниками, як FPS або LCP, але й впроваджувати інклюзивні рішення, які задовольняють різноманітні сценарії використання, зокрема, вимоги людей з вадами зору або опорно-рухового апарату.

Тільки завдяки комплексному підходу можна гарантувати ефективність, стабільність і зручність використання цифрового продукту на всіх рівнях взаємодії.

5. Роль інклюзивних анімацій у Flutter-застосунках

Інклюзивність в анімаційному дизайні відіграє надзвичайно важливу роль у розробці сучасних мобільних додатків. Flutter, як кросплатформенний фреймворк, дозволяє створювати візуально привабливі, плавні та інтерактивні інтерфейси. У той же час, надмірне або неправильне використання анімації може стати бар'єром для деяких користувачів, особливо тих, хто має порушення зору, вестибулярні розлади або чутливість до руху.

Люди з такими порушеннями можуть відчувати дискомфорт або втрату контролю при взаємодії з анімованим інтерфейсом. Наприклад, анімація, яка не має логічного контексту або занадто швидка/повільна, може ускладнити навігацію.

Саме тому інклюзивний підхід до проектування анімації вимагає врахування користувацьких налаштувань, таких як MediaQuery.of(context). disableAnimations, які вказують на необхідність мінімізувати або відключити рухомі елементи.

Flutter також підтримує використання віджету «Семантика», який дозволяє озвучувати контент і проговорювати дії, викликані анімацією. Ця функція гарантує, що такі програми для читання з екрану, як TalkBack і VoiceOver, зможуть правильно інтерпретувати динамічні зміни в інтерфейсі.

На особливу увагу заслуговує специфікація ARIA Authoring Practices Guide (APG), адаптована для дизайну Flutter. Хоча Flutter не використовує HTML безпосередньо, принципи APG – такі як передбачуваність, логічна навігація та доступність для альтернативних пристроїв введення – залишаються актуальними.

Розробники можуть слідувати цим принципам, щоб забезпечити комфорт для всіх користувачів незалежно від їхніх фізичних можливостей.

Таким чином, метою анімації доступності є не лише мінімізація когнітивного навантаження, але й забезпечення того, щоб анімаційний шар надавав доступ до функціональності додатку, а не блокував його, а отже, сприяв кращій взаємодії з користувачем.

6. Інструменти для оцінки інклюзивності

Щоб оцінити якість доступності анімації Flutter, розробники можуть використовувати кілька інструментів, які прямо чи опосередковано підтримують тестування доступності:

Flutter DevTools дозволяє профілювати продуктивність анімації, що дає змогу виявити фризи, затримки та перевантаження графічного потоку.

Це важливо для забезпечення плавної анімації, особливо на пристроях з низькою обчислювальною потужністю.

Lighthouse (у веб-версіях додатків Flutter) – дозволяє аналізувати доступність, зокрема виявляти проблеми з контрастністю, нестандартні елементи керування, а також вказує на проблеми зі збільшенням масштабу або відключенням звуку в динамічному контенті.

TalkBack (Android), VoiceOver (iOS) – основні системні інструменти, що використовуються для тестування озвучення елементів, зокрема, після анімаційних переходів або появи нових віджетів [6].

Screen Reader Testing – ручне тестування за участю користувачів з особливими потребами або автоматизоване тестування за допомогою емуляторів, що дозволяє перевірити, наскільки логічною є структура анімованого інтерфейсу для сторонніх програм. Хоча у Flutter ще немає повністю нативного масштабного інструменту тестування доступності анімації, який можна порівняти з axe або WAVE у веб-середовищі, його вбудована інтеграція з Lighthouse, DevTools та відповідність керівним принципам Material Accessibility дає розробникам інструменти, необхідні для виявлення та усунення бар'єрів.

Висновки

Анімація у Flutter – чудовий спосіб підвищити візуальну привабливість, зробити навігацію ефективнішою та зменшити когнітивне навантаження на користувача.

Однак дуже важливо, щоб їх застосування не порушувало принципів доступності, оскільки надмірна або неконтрольована анімація може спричинити проблеми у взаємодії з інтерфейсом для людей з обмеженими можливостями.

Дослідження підтвердило, що за умови дотримання рекомендацій щодо продуктивності (RepaintBoundary, оптимізація setState, профілювання через DevTools) та доступності (Semantics, MediaQuery.disableAnimations, системні зчитувачі екрану) Flutter дозволяє створювати інтерактивні та інклюзивні інтерфейси.

Такі інструменти, як Google Lighthouse, Flutter DevTools, а також рекомендації Material Design Accessibility та ARIA APG надають розробникам засоби для всебічної оцінки та покращення анімованих інтерфейсів.

Таким чином, поєднання продуктивності, естетики та інклюзивності в реалізації анімації Flutter є основою для високоякісного мобільного досвіду, який враховує потреби найширшої аудиторії – незалежно від їхніх фізичних або когнітивних здібностей.

Конфлікт інтересів

Автори заявляють, що не мають конфлікту інтересів.

Список джерел:

[1] Flutter documentation: Animations overview. URL: https://docs.flutter.dev/ui/animations/overview

[2] Flutter Performance profiling with DevTools. URL: https://docs.flutter.dev/tools/devtools

[3] Web Vitals. Google Developers. URL: https://web.dev/articles/vitals

[4] Material Design – Motion principles. Google. URL: https://m3.material.io/styles/motion/overview/how-it-works

[5] Accessibility in Flutter. Flutter documentation. URL: https://docs.flutter.dev/ui/accessibility-and-internationalization/accessibility

[6] TalkBack screen reader for Android. Google Support. URL: https://support.google.com/accessibility/android/answer/6007100?hl=en

[7] Lighthouse. Google Developers. URL: https://developer.chrome.com/docs/lighthouse/overview/

[8] Semantics Widget. Flutter API Reference. URL: https://api.flutter.dev/flutter/widgets/Semantics-class.html

[9] MediaQuery class – Flutter. URL: https://api.flutter.dev/flutter/widgets/MediaQuery-class.html

[10] Animations in Flutter. URL: https://codelabs.developers.google.com/advanced-flutter-animations#0

Надійшла до редколегії 03.04.2025