DOI https://doi.org/10.36059/978-966-397-491-0-26

ЦИКЛ РОЗРОБКИ МОБІЛЬНОГО ЗАСТОСУНКУ

Ілляшенко О. М.
аспірант за спеціальністю 123 – Комп'ютерна інженерія
Міжнародний гуманітарний університет
Науковий керівник: ***Вакарчук А. О.***
кандидат технічних наук,
доцент кафедри комп'ютерної інженерії
та інноваційних технологій факультету кібербезпеки,
програмної інженерії та комп'ютерних наук
Міжнародний гуманітарний університет
м. Одеса, Україна

Виклики що постали перед Україною сьогодні, потребують комплексного підходу для її відновлення і розвитку. IT-технології є одним з ключових рушіїв змін, адже вони дозволяють оптимізувати, автоматизувати та масштабувати різні аспекти державних послуг, освіти, медицини, оборони та інших сфер. Цифрова трансформація, яка триває останніми роками, довела свою важливість і ефективність, значно спростивши взаємодію громадян із державними органами та прискорила бюрократичні процеси. Особливе місце займають мобільні рішення, такі як «Дія» [1] (скорочення від «Держава і я»), котрі дозволяють користувачам отримувати доступ до важливих сервісів у будь-який час і в будь-якому місці. Оскільки це потужний інструмент, який можна застосовувати в багатьох сферах, важливе значення має оптимізація процесів розробки зі збереженням високої якості кінцевих продуктів.

Рис. 1. Схема циклу розробки мобільного застосунку

Дослідження. Цей етап є основою для всього подальшого процесу проектування та допомагає визначити, наскільки необхідне це програмне забезпечення. Бажано залучати експерта з предметної області, для якої планується розробка, він допоможе краще зрозуміти цільову аудиторію, переваги і недоліки конкурентів, основні проблеми, які потрібно вирішити.

Планування. Визначає функціональність, архітектуру, стек технологій, бізнес-логіку та розподіл завдань у команді. Функціонал можна розділити на критичний, важливий, бажаний і непотрібний. Спираючись на цей розподіл, слід обрати, що входитиме до MVP [2], і на основі цього визначити підходящу архітектуру і стек технологій. Неправильно обрана архітектура може призвести до технічних обмежень, нераціонального використання часу, низької якості, що може мати критичні наслідки для продукту.

Сьогодні одними з найпопулярніших підходів у розробці мобільних застосунків є: Native development (Swift/Kotlin), React Native, Flutter та Kotlin Multiplatform [3]. Вибір одного з підходів залежить від кількох факторів, таких як вимоги до продуктивності, часу на розробку, бюджету, технічних навичок команди, а також специфіки самого проєкту.

Дизайн (UX/UI). Добре продуманий UI/UX значно покращує взаємодію користувачів із застосунком і впливає на успіх продукту. На основі результатів попереднього етапу потрібно розробити навігацію і взаємодію з елементами інтерфейсу. Спершу потрібно створити sitemap для розуміння взаємодії між екранами і вайрфрейми для візуалізації інтерфейсу. Вони допоможуть виявити недоліки UX на ранніх етапах розробки. Далі потрібно переходити до інтерактивних прототипів і роботи над UI (кольорова палітра, компоненти, шрифти, анімації). Для виконання описаних задач підходить онлайн-інструмент Figma.

Розробка. Для спрощення процесу розробки, модифікації і підтримки необхідно притримуватись загальних принципів, таких як: S.O.L.I.D, KISS, DRY, YAGNI [4]. Інтегрувати DI [5] для меншої зв'язаності коду, більшої гнучкості у розширенні і спрощення тестування. Використовувати патерни проектування: породжувальні, структурні, поведінкові [6]. Це покращить якість коду і спростить комунікацію між розробниками.

Для спрощення супроводу проєкта і покращення стабільності бажано покривати код unit-тестами і інтеграційними тестами.

Залежно від обраного підходу та складності застосунку, потрібно застосовувати відповідні архітектурні рішення, які є оптимальними для конкретної технології. В Android Native – використовувати MVVM та Clean Architecture. У React Native та Flutter – застосовувати стейт-менеджмент бібліотеки, такі як MobX, Redux (React Native), BLoC (Flutter).

Тестування. Для впевненості в працездатності застосунку потрібно провести серію тестів: від перевірки загального функціоналу (smoke-тестування, функціональне тестування, тестування продуктивності) до

специфічних безпекових тестів для збереження конфіденційних даних (перехоплення та шифрування даних, реверс інжиніринг, безпечність API). Якщо тестування показує незадовільні результати, формується список проблем і застосунок відправляється на доопрацювання. Цей етап повторюється кожен раз при підготовці нової версії до публікації. Для економії часу і спрощення цього етапу можна використовувати фреймворки для автоматизації тестування, це потребує більше часу на початковому етапі, але заощадить його в майбутньому.

Публікація та підтримка. Перед публікацією застосунку в App Store/Google Play необхідно його підготувати для подальшої підтримки. Необхідно інтегрувати інструменти для моніторингу збоїв і збирання логів, такі як: Firebase Crashlytics, Bugfender, Sentry. Реалізувати Force Update для можливості примусового оновлення. Це необхідно для підтримки безпеки, стабільності і сумісності версій. Окрім цього, потрібно налаштувати збирання аналітики за допомогою Google Analytics чи AppsFlyer для відстеження поведінки користувачів.

Ефективна розробка мобільних застосунків вимагає комплексного підходу, де кожен етап впливає на наступний і на кінцеву якість продукту. Важливо правильно обирати технології та архітектуру, бо це вплине на зручність використання, стабільність, безпеку і популярність.

Література:

1. Diia. URL: https://en.wikipedia.org/wiki/Diia
2. MVP: Minimum Viable Product. URL: https://medium.com/@r.b.srikanth/mvp-minimum-viable-product-223e07b92a63
3. Native vs cross-platform mobile app development. URL: https://circleci.com/blog/native-vs-cross-platform-mobile-dev
4. Solid Dry Kiss Yagni – Engineering Principles. URL: https://medium.com/geekculture/solid-dry-kiss-yagni-engineering-principles-c1e73610db4c
5. Dependency injection. URL: https://en.wikipedia.org/wiki/Dependency_injection
6. Рефакторинг.Гуру. URL: https://refactoring.guru/uk