УДК 004.052.32

Техники повышения отказоустойчивости нативных iOS-приложений

Манаков И.М. (ИТМО)
Научный руководитель – доктор экономических наук, профессор Максимова Т.Г. (ИТМО)

Введение. Операционная система iOS является самой защищенной мобильной операционной системой [1]. Данный статус на протяжении многих лет поддерживается за счет защитных механизмов компании Apple на всех платформенных уровнях. Однако такое решение ограничивает и затрудняет разработку нативных приложений. Например, важным аспектом разработки мобильных приложений является обеспечение отказоустойчивости – способности функционировать надежно в различных ситуациях, включая непредвиденные сбои и ошибки. Для решения данной проблемы необходимо внедрять различные техники, снижающие риск сбоя в работе программного продукта. Целью работы является разработка техник повышения отказоустойчивости нативных iOS-приложений.

Основная часть. Приложения на iOS обладают жизненным циклом, который оптимально распределяет ресурсы устройства. Одним из этапов этого цикла является завершение работы приложения (termination), когда система останавливает процесс. Существует множество причин остановки – от пользовательского нажатия до оптимизации системы [2]. Поскольку завершение работы приложения пользователем является явным, требуется определить основные программные причины остановки приложения. В ходе работы было выявлено, что наиболее распространенными являются:

1) ошибка во время выполнения программы (runtime error),
2) системное прерывание (abort),
3) ошибка доступа к памяти,
4) ошибка конфигурации приложения.

Они негативно сказываются на пользовательском опыте, поэтому на этапе разработки их необходимо минимизировать (Apple рекомендует не более 1% сбоев среди всех пользовательских сессий). Для избежания перечисленных сбоев существует множество техник, применяемых при разработке приложений. В работе были предложены техники для устранения перечисленных выше причин. Опишем их.

Все современные нативные приложения на iOS пишутся на высокоуровневом языке Swift. Строгая типизация языка накладывает на разработчика ответственность за исполнение кода. Применение безопасного разворачивания опциональных выражений (optional unwrapping) и использование защитных конструкций (guard statements) позволяют безопасно определить типы во время выполнения программы. Также допускается использовать asserts в случае отсутствия необходимых данных, предупреждая пользователя о имеющейся ошибке и предоставив возможность вернуться в приложении обратно. Повсеместное применение таких конструкций избавляет приложение от ошибок во время выполнения программы.

Чтобы исключить ошибки с зависанием приложения и, вследствие, системное прерывание, необходимо писать архитектурно и синтаксически качественный код. Правильно подобранная архитектура (слабосвязанные зависимости), корректное управление памятью (использование weak и unowned ссылок для освобождения объектов) эффективно расходуют ресурсы устройства.

Для повышения безопасности операционной системы Apple внедрила механизм экстренного завершения процесса при попытке несанкционированного доступа к другим участкам памяти. Чтобы не допустить такую ошибку, разработчику необходимо следить за переполнением переменных и выходом за границы структур данных.

Помимо предупреждения ошибок выполнения при разработке, рекомендуется использовать мониторинговые инструменты для сбора отчетов [3] об ошибках выполнения программы. Существующие решения (например, Crashlytics от Firebase или AppMetrica от

Яндекс) предлагают широкий спектр аналитики для выявления и исправления существующих в приложении уязвимостей.

Выводы. Практическое применение систематизированных техник позволило повысить уровень отказоустойчивости разрабатываемых нативных iOS-приложений до безопасных показателей.

Список использованных источников:

1. Android vs. iOS: Security comparison 2024 [Электронный ресурс]. – URL: https://nordvpn.com/ru/blog/ios-vs-android-security/ (дата обращения 05.02.2024)

2. Reducing terminations in your app [Электронный ресурс]. – URL: https://developer.apple.com/documentation/xcode/reduce-terminations-in-your-app (дата обращения 05.02.2024)

3. Identifying the cause of common crashes [Электронный ресурс]. – URL: https://developer.apple.com/documentation/xcode/identifying-the-cause-of-common-crashes (дата обращения 05.02.2024)