УДК 004.4

РАЗРАБОТКА ПРОИЗВОДИТЕЛЬНОГО ФРЕЙМВОРКА ДЛЯ ДЕКЛАРАТИВНОГО ПОСТРОЕНИЯ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА В IOS

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

Введение. В связи с появившимися ограничениями в цифровой сфере отечественные технологические компании стали стремительно развивать свои экосистемы. Все открытые для свободного доступа инструменты, библиотеки и фреймворки начали замещаться внутренними решениями для повышения надежности [1]. Такие нововведения затронули и разработку пользовательского интерфейса – основной этап создания мобильных приложений на iOS. Поэтому появилась необходимость разработать внутрикорпоративное решение.

Разрабатываемое программное решение будет полностью соответствовать требованиям проекта и будет совместимо как с нативными компонентами, так и с компонентами корпоративной дизайн-системы. Разрабатываемый DSL-фреймворк сочетает в себе преимущества декларативного подхода SwiftUI и высокую производительность UIKit.

Основная часть. Проект, для которого разрабатывается фреймворк, представляет из себя нативное iOS-приложение с MAU ~23 млн. Кодовая база состоит из ~200 репозиториев на языке программирования Swift. В рамках производственного процесса важно разработать универсальный для всех потребителей инструмент с прозрачной системой версионирования. Для этого было выбрано семантическое версионирование – позволяет однозначно представлять изменения между разными релизами программного продукта или библиотеки.

В процессе проектирования были взяты за основы существующие стандарты компании:

1) Несмотря на подходящую для современного SwiftUI минимальную поддерживаемую приложением версию iOS, фреймворк не прошел проверку на стабильность [2] и не соответствует требованиям платформенной команды – поэтому весь пользовательский интерфейс строится на UIKit.

2) Каждый компонент должен вычислять intrinsicContentSize для корректной расстановки внутри контейнеров, а также соответствовать протоколу Modelable (через свойство model происходит конфигурация).

3) При построении пользовательского используются механизмы frame-based и Auto Layout.

При выборе подхода к построению пользовательского интерфейса был проведен сравнительный анализ двух популярных подходов. Императивный подход дает детальный контроль и понятен в смысле последовательности действий, но требует больше ручного кода и аккуратности при сложной логике. Декларативный подход сокращает код и упрощает описание конечного результата, делая его более читаемым, но иногда скрытые механизмы усложняют отладку и требуют переосмысления принципов разработки. Исходя из рассмотренных критериев было принято решение создать DSL-фреймворк на основе UIKit с декларативными конструкциями, что позволит оптимизировать внутрикорпоративные процессы по разработке приложения и поддержания функционала [3]. Разработанный синтаксис был оценен методом парных сравнений. Также было проведено snapshot-тестирование на полное соответствие построенных интерфейсов.

Для расположения компонентов на экране были изучены нативные механизмы рендеринга, на основе которых был разработан эффективный алгоритм diff. Для группировки объектов и декларативного стиля были созданы контейнеры. Они обладают гибкой настройкой и полностью совместимы с корпоративной системой. Оценка производительности была проведена при помощи бенчмарк-тестов, которые показали, что фреймворк сохранил высокую производительность frame-based механизма.

Выводы. Разработанный DSL-фреймворк демонстрирует высокую производительность и гибкость, что делает его пригодным для создания сложных и отзывчивых пользовательских интерфейсов на iOS. Его использование позволяет сократить время разработки, упростить поддержку кода и обеспечить стабильную работу приложения. В перспективе планируется расширение функциональности фреймворка, включая поддержку реактивных библиотек.

Список использованных источников:

1. Импортозамещение ПО в России: кому обязательно менять зарубежный софт [Электронный ресурс]. – URL: https://kontur.ru/talk/spravka/51205-importozameshchenie_po_v_rossii (дата обращения 15.10.2024).

2. Гухемухова Б., Чомаев Ш. Проблемы внедрения нового фреймворка SwiftUI для разработки приложений на устройства Apple // Тенденции развития науки и образования. – 2020. – № 67. – С. 30 – 33.

3. Gilchrist E., Balasooriya J., Bansal A. SwiftUI vs UIKit: A Case Study on How a Declarative Framework Can Improve Learnability of UI Programming // Skripsi, Program Studi Teknik Informatika. – Tempe: Arizona State University, 2021. – 7 p.