УДК 004.4

Н.А. Горбунова, доц.; Н.Ю Фрицлер, студ.
(КарУ, г. Караганда, Казахстан)

АРХИТЕКТУРНЫЕ ПОДХОДЫ К РАЗРАБОТКЕ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ ДЛЯ ANDROID И IOS ПЛАТФОРМ

Разработка мобильных приложений для платформ Android и iOS – одна из ключевых задач современного IT-сектора. Выбор архитектуры играет важнейшую роль в обеспечении успешной реализации проекта, влияя на такие аспекты, как производительность, удобство сопровождения и масштабируемость. Учитывая различия между платформами, разработчики сталкиваются с рядом вызовов, таких как необходимость учитывать особенности каждой системы, потребность в кроссплатформенной совместимости и поддержке долгосрочного жизненного цикла приложения. Кроме того, архитектура определяет, насколько легко можно будет интегрировать новые технологии, реагировать на изменяющиеся требования пользователей и поддерживать высокое качество приложения на протяжении его жизненного цикла. Эффективные архитектурные решения позволяют минимизировать затраты на разработку, оптимизировать процессы тестирования и развертывания, а также обеспечивать надежную и стабильную работу приложения в условиях интенсивного использования.

Современные мобильные приложения часто развиваются и обрастают новыми функциями в процессе эксплуатации. Архитектура должна обеспечивать возможность добавления новых модулей и компонентов без значительных изменений существующего кода. Масштабируемость особенно важна для крупных проектов, где изменения в одной части приложения не должны нарушать работу других модулей. Примером может служить использование многослойных архитектур, таких как Clean Architecture, где каждый слой изолирован и отвечает за строго определенные задачи. Такой подход облегчает внесение изменений и позволяет равномерно распределять рабочую нагрузку между командами разработчиков.

Учитывая высокий спрос на мобильные приложения, работающие на обеих основных платформах (Android и iOS), архитектура должна предусматривать возможность повторного использования кода. Это позволяет сократить время разработки и снизить затраты, не жертвуя качеством продукта. Кроссплатформенные фреймворки, такие как Flutter, React Native или Xamarin, предоставляют инструменты для создания приложений, которые работают на обеих платформах. Однако разработчики сталкиваются с необходимостью учитывать различия в

пользовательских интерфейсах и поведении систем, что требует гибких архитектурных решений, поддерживающих адаптацию к платформенным особенностям [1].

Пользовательский опыт во многом зависит от быстродействия и стабильности приложения. Архитектура должна быть спроектирована таким образом, чтобы эффективно использовать ресурсы устройства (память, процессор, сеть) и минимизировать риски сбоев. Например, использование паттернов управления состоянием, таких как Redux или Bloc, помогает оптимизировать обработку данных в приложении, предотвращая избыточное потребление ресурсов.

Качественное приложение требует регулярного обновления и исправления ошибок, что делает процесс тестирования и поддержки важным аспектом разработки. Хорошо продуманная архитектура способствует модульному тестированию, позволяя изолировать и проверять отдельные части системы. Например, при использовании архитектурного подхода MVVM (Model-View-ViewModel) можно легко тестировать бизнес-логику и взаимодействие с данными, не затрагивая пользовательский интерфейс. Это ускоряет процесс поиска ошибок и внедрения новых функций [2].

Современная разработка мобильных приложений опирается на различные архитектурные подходы, каждый из которых имеет свои особенности, преимущества и области применения. Одним из самых распространенных подходов является Model-View-Controller (MVC). Этот подход предполагает разделение приложения на три ключевых компонента: модель, представление и контроллер. Модель отвечает за управление данными, представление отображает пользовательский интерфейс, а контроллер обеспечивает взаимодействие между ними. Этот подход широко используется благодаря своей простоте и структурированности, однако он может приводить к избыточной связи компонентов в крупных проектах, что затрудняет масштабируемость и поддержку.

Model-View-Presenter (MVP) является эволюцией MVC, где контроллер заменяется презентером, который полностью управляет логикой представления. Такой подход обеспечивает более четкое разделение обязанностей и делает код более тестируемым. MVP часто применяется в проектах с высокими требованиями к тестированию и гибкости пользовательского интерфейса.

Model-View-ViewModel (MVVM) стал популярным благодаря своей способности упрощать привязку данных и взаимодействие между пользовательским интерфейсом и логикой. Этот подход активно используется в мобильной разработке, особенно в среде Android, где фреймворки, такие как Jetpack, поддерживают его реализацию. MVVM

позволяет минимизировать дублирование кода и улучшает тестируемость благодаря отделению логики представления от самой визуальной части [3].

Clean Architecture – это многослойный подход, который обеспечивает модульность и изоляцию различных уровней приложения. Основная идея заключается в разделении приложения на независимые слои, такие как пользовательский интерфейс, бизнес-логика и слой данных. Это не только упрощает тестирование и поддержку, но и делает приложение гибким к изменениям. Clean Architecture подходит для крупных и долгосрочных проектов, где требования могут меняться на протяжении всего цикла разработки.

Другие подходы, такие как Flux, Redux и микросервисная архитектура, также нашли свое применение в мобильной разработке, особенно в приложениях, где требуется сложное управление состоянием или высокая степень масштабируемости.

Разные архитектурные подходы имеют свои сильные и слабые стороны, которые проявляются в зависимости от особенностей проекта. MVC, например, является одним из самых простых для освоения и реализации, что делает его популярным выбором для небольших и средних проектов. Однако в крупных приложениях он может приводить к сильной связности компонентов, усложняя их поддержку и тестирование.

MVP превосходит MVC в вопросах четкого разделения логики и тестируемости, что делает его подходящим для проектов, где бизнес-логика играет ключевую роль. Тем не менее, управление большим количеством презентеров может усложнять разработку, особенно если проект становится более масштабным.

MVVM предлагает удобную модель для обработки данных и их отображения благодаря механизмам привязки, что особенно актуально для платформ с поддержкой таких инструментов. Однако его реализация требует большего времени и ресурсов, что может быть нецелесообразным для небольших приложений.

Clean Architecture, хотя и является наиболее гибким и модульным подходом, имеет значительную сложность в реализации. Его преимущество заключается в возможности изолировать бизнес-логику от пользовательского интерфейса и слоя данных, что упрощает тестирование и адаптацию к изменениям. Однако для малых и краткосрочных проектов его использование может быть избыточным.

В конечном итоге выбор архитектурного подхода зависит от множества факторов: размера команды, масштаба проекта, требований к производительности, бюджета и срока реализации. Для небольших

проектов подходы вроде MVC или MVP могут быть достаточными, тогда как для более сложных и долгосрочных проектов MVVM или Clean Architecture становятся более предпочтительными благодаря своей модульности и гибкости.

Современные подходы к архитектуре мобильных приложений активно развиваются под влиянием новых технологий и меняющихся требований пользователей. Одним из ключевых трендов является использование микросервисной архитектуры, адаптированной для мобильной разработки. Такой подход подразумевает разделение приложения на независимые модули, каждый из которых выполняет свою задачу. Это позволяет улучшить масштабируемость и упростить внедрение новых функций. Асинхронное программирование и управление состоянием также стали важными элементами современных архитектур. Такие инструменты, как Redux, Bloc или MobX, обеспечивают эффективное управление состоянием приложения, минимизируя дублирование данных и обеспечивая согласованность между различными компонентами. Это особенно актуально для приложений с большим количеством динамических данных или сложной бизнес-логикой [4].

Еще одним значительным трендом является интеграция Dependency Injection (DI), которая упрощает управление зависимостями в приложении. Инструменты вроде Dagger или Hilt для Android и SwiftUI для iOS позволяют автоматически создавать и управлять зависимостями, что делает архитектуру более гибкой и удобной для тестирования.

Кроссплатформенные подходы, такие как использование Flutter, React Native или Kotlin Multiplatform, также продолжают набирать популярность. Они позволяют существенно сократить время и ресурсы на разработку, при этом обеспечивая высокую производительность и близкий к нативному пользовательский опыт.

Кроме того, современные архитектурные решения активно адаптируются под требования DevOps, что включает автоматизацию тестирования, интеграции и развертывания. Это упрощает процесс доставки обновлений и улучшает контроль качества приложения.

Архитектура мобильного приложения является фундаментом успешного проекта. Ее выбор влияет на все аспекты разработки: от производительности и масштабируемости до удобства сопровождения и тестирования. Правильно выбранная архитектура позволяет эффективно адаптироваться к изменяющимся требованиям, снижать затраты на разработку и обеспечивать высокий уровень удовлетворенности пользователей.

Рассмотренные подходы, такие как MVC, MVP, MVVM и Clean

Architecture, предлагают широкий спектр возможностей для реализации приложений на платформах Android и iOS. Каждый из них имеет свои сильные стороны и подходит для различных типов проектов. Современные тренды, включая микросервисы, управление состоянием, DI и кроссплатформенные фреймворки, дополняют традиционные подходы, позволяя создавать более гибкие и адаптивные системы.

В условиях стремительно развивающейся мобильной индустрии выбор архитектуры должен основываться на тщательном анализе потребностей проекта, особенностей целевой аудитории и требований бизнеса. Только с учетом всех этих факторов можно создать приложение, которое будет успешно развиваться и оставаться актуальным в долгосрочной перспективе.

ЛИТЕРАТУРА

1. Фаулер М. Архитектура корпоративных программных приложений. – СПб.: Символ-Плюс, 2022.

2. Боб Мартин. Чистая архитектура: Искусство разработки программного обеспечения. – М.: Питер, 2019.

3. Шумаков А. Модели и архитектуры мобильных приложений: от теории к практике. – СПб.: Лань, 2021.

4. Кокошко А. Программная инженерия: подходы, методологии, инструменты. – СПб.: Наука и технологии, 2020.

УДК 004.415.2

О.А. Крайнова, доц. (ТГУ, г. Тольятти, Россия)

МОДЕЛИРОВАНИЕ ДАННЫХ В 1С:ERP: ВЫБОР НОТАЦИИ

ERP-системы играют ключевую роль в цифровой трансформации, обеспечивая единое пространство для управления и планирования. Российские компании всё больше ориентируются на отечественные ERP-решения, среди которых, согласно данным базы TAdviser [1], лидирующую позицию составляет доля вендеров 1С, занимающая более 68 % рынка.

1С:ERP предлагает широкий набор инструментов для управления основными бизнес-процессами предприятия. Система обеспечивает мониторинг ключевых показателей, оптимизирует взаимодействие между подразделениями и позволяет настраивать алгоритмы для оценки эффективности работы на различных уровнях.

Переход на использование такого класса систем процесс трудоемкий и длительный, поэтому в обязательном порядке сопровождается