УДК 621.3.049.77–048.24:537.2

ОСОБЕННОСТИ ИСПОЛЬЗОВАНИЯ API В МОБИЛЬНЫХ ПРИЛОЖЕНИЯХ

Винцелович Л.В.

Белорусский государственный университет информатики и радиоэлектроники, г. Минск, Республика Беларусь

Научный руководитель: Писарчик А.Ю. – ст. преподаватель кафедры ПИКС

Аннотация. Выяснены особенности использования *API* в мобильных приложениях. Раскрыты проблемы внедрения интерфейсов прикладного программирования при разработке мобильных приложений с их использованием.

Ключевые слова: API, мобильные приложения, проектирование мобильных приложений

Введение. Интерфейсы прикладного программирования *(API)* стали неотъемлемой частью разработки мобильных приложений. *API* – это любой интерфейс, позволяющий двум программным компонентам взаимодействовать. Если сузить определение до современного понимания сетевого *API*, то это любой интерфейс, позволяющий двум программным компонентам взаимодействовать через сеть [1].

В данной статье автором показано, что использование *API* имеет много преимуществ, включая расширение функциональности приложения, увеличение эффективности взаимодействия пользователя с элементами пользовательского интерфейса, а также экономию времени разработчиков и финансовых затрат на создание такого приложения. Однако, это создает такие проблемы, как надежность приложения, его безопасность и совместимость с *API*. Разработчики должны внимание на потенциальные проблемы в ходе разработки такого приложения.

Основная часть. *API*-интерфейсы позволяют разработчикам интегрировать различные функции в свои мобильные приложения. Это означает, что разработчики могут добавлять новые функции в свои приложения, не создавая их с нуля. Например, если разработчик хочет добавить функцию карты в свое приложение, он может просто использовать *API* Карт *Google* для интеграции этой функции.

Первым шагом в интеграции *API* является выбор правильного *API*. Разработчику необходимо тщательно оценить функциональность, надежность, безопасность и совместимость *API* со своим приложением. Также стоит учитывать документацию и поддержку *API*, поскольку эти факторы могут повлиять на процесс интеграции.

При внедрении *API* следует использовать установленные стандарты и протоколы: *RESTful API*, формат данных *JSON* или *XML*, *OAuth* для аутентификации и авторизации и т.д.

Интеграция *API* влечет за собой внедрение механизма обработки ошибок, например, сбои аутентификации, ограничения скорости и ошибки сервера.

В *Java* существует *API*, обеспечивающий простое и быстрое создание *REST*-сервисов, который предлагает модель описания распределенных ресурсов на основе аннотаций, при помощи которых описываются местоположение и представление ресурсов – *JAX-RS*.

Основная суть подхода к обработке ошибок в *JAX-RS* заключается в следующем. Разработчик определяет свой обработчик собственного исключения и регистрирует в окружении *JAX-RS*. После того, как обработчик успешно зарегистрирован, при наступлении исключительной ситуации в коде ресурса достаточно лишь выбросить экземпляр зарегистрированного исключения с помощью ключевого слова *throw*. Все остальное *JAX-RS* сделает автоматически [2].

Интеграцию *API* необходимо тщательно протестировать. Сюда входит тестирование функциональности, производительности и безопасности *API*. Они также должны протести-

ровать интеграцию в различных сценариях, таких как большие объемы трафика и сбои в сети.

Главная цель тестирования *API* – убедиться, что он может достичь стратегической цели, которая была определена в процессе его создания. Вторая цель тестирования *API* – убедиться, что вся проделанная работа была достаточно качественной для того, чтобы поддерживать стратегию. Например, если *API* сложен для использования и изучения, это может повлиять на стратегическую цель «привлечь больше пользователей *API*». Это означает, что необходимо определить конкретные тесты, чтобы добиться высокого качества интерфейса. Нужно также протестировать проделанную работу на внутреннее единообразие. Например, может понадобиться проверить, что разработанная реализация совпадает с интерфейсом [1].

Разработчики должны контролировать и поддерживать интеграцию *API*. Это включает в себя мониторинг производительности и использования *API*, а также решение любых возникающих проблем. Поддержка интеграции *API* заключается и в установке, а также соблюдении совместимости, высокой производительности последних версий и обновлений.

Разработчики *Android*, как правило, используют последние *API*, чтобы разблокировать новые функции и предотвратить сбои приложений в обновленных версиях *Android* из-за изменений в поведении системы. Однако обновление целевого уровня *API* нетривиально. Согласно руководству по миграции *Android API*, чтобы определить места, где может быть затронута программа, разработчики должны проанализировать изменения в поведении системы на уровне *API*, который они хотят обновить. Затем они должны выполнить тесты, сфокусированные на изменении поведения системы используемого *API*, и для этого они должны проверить все рабочие процессы приложения. Когда проблемы обнаруживаются, им нужно найти используемые *API*, которые отвечают за неожиданное поведение, и внести изменения в код, чтобы исправить эти проблемы. В отсутствие автоматизированного анализа воздействия разработчикам, возможно, придется вручную идентифицировать затронутые элементы в коде и выполнить все тесты, чтобы обнаружить неожиданное поведение, вызванное обновлением уровня *API* [3].

В таком случае можно прибегнуть к последовательному совершенствованию. Локальное улучшение компонентов архитектуры приложения должно приводить к совершенствованию всей архитектуры в целом – к росту суммарной полезности компонентов того же уровня, что и изменяемый, равно как и компонентов более низкого и более высокого уровня. Например, известный веб-сервис *Google Translate* постоянно претерпевает изменения. Изначально, он обеспечивал только веб-интерфейс для перевода и ограниченный набор языков. Постепенно увеличивались функциональные возможности сервиса: расширялся набор языков, появилась возможность голосового воспроизведения перевода, при переводе отдельного слова начали выдаваться словарные статьи с несколькими результатами перевода и т.п. При этом *API* и интерфейс менялся настолько незначительно, что клиенты могли использовать новые возможности посредством старого *API* (например, получение словарных статей) или же не менять используемый интерфейс до тех пор, пока не потребуется использование новых возможностей [4].

Связь между *UX* и *API* является важным аспектом разработки мобильных приложений. Важно убедиться, что *UX* и *API* хорошо интегрированы и согласованы. Ключом к разработке эффективной коммуникации *UX* и *API* является учет потребностей и предпочтений пользователя. Пользовательский интерфейс должен быть разработан таким образом, чтобы интеграция *API* была интуитивно понятной и бесшовной, чтобы пользователи могли легко перемещаться по приложению и выполнять нужные действия. Следует учесть тот факт, что интеграция *API* должна быть оптимизирована по скорости и эффективности, чтобы приложение работало гладко и быстро.

Таким образом, интеграция *API* может быть сложной задачей, но следование рекомендациям может помочь разработчикам создавать эффективные, надежные и безопасные интеграции. Выбирая правильный *API*, понимая его требования, планируя процесс интеграции, используя установленные стандарты и протоколы, внедряя механизмы обработки ошибок,

тестируя интеграцию, а также отслеживая и поддерживая интеграцию, разработчики могут создавать успешные интеграции *API*.

Заключение. Выяснены особенности использования *API* в мобильных приложениях. Определены этапы внедрения интерфейсов прикладного программирования в ходе создания приложения. Выявлены потенциальные проблемы и внедрения и предложены способы их решения.

Список литературы

1. *Continuous API Management: Making the Right Decisions in an Evolving Landscape / Mehdi Medjaoui, Erik Wilde, Ronnie Mitra, Mike Amundsen. – 1st ed. – O'Reilly Media, 2019. – 290 p.*

2. *Технологии веб-сервисов : учебно-методическое пособие / Дергачев А. М. [и др.]. – СПб : Университет ИТМО, 2021. – 100 с.*

3. *Analyzing the impact of API changes on Android apps / Tarek Mahmud, Meiru Che, Guowei Yang // Journal of Systems and Software. – 2023. – Vol. 200.*

4. *Распределенные вычислительные системы : учебное пособие / Радченко И. Г. – Челябинск : Фотохудожник, 2012. – 184 с.*

UDC 621.3.049.77–048.24:537.2

FEATURES OF USING API IN MOBILE APPLICATIONS

Vintselovich L.V.

Belarusian State University of Informatics and Radioelectronics, Minsk, Republic of Belarus

Pisarchik A.Y. – senior lecturer of the Department of ICSD

Annotation. Features of API usage in mobile applications have been clarified. The article reveals problems of introduction of interfaces of application programming at development of mobile applications with their use.

Keywords: API, mobile apps, design of mobile applications