УДК 004.42:621.395.721.5

22. ПОТОКОВЫЕ СЕРВИСЫ НА КОМПЛЕКСЕ МОБИЛЬНЫХ УСТРОЙСТВ IOS

Ходарёнок Н.А., магистрант гр. 376741, Петрович Ю.Ю. магистрант 376701,

Белорусский государственный университет информатики и радиоэлектроники[1]
г. Минск, Республика Беларусь

Ефремов А.А. – канд. экон. наук, доцент каф. ЭИ

Аннотация. Целью работы является исследование реализации потоковых сервисов на устройствах iOS, обеспечивая глубокий анализ особенностей разработки и интеграции таких сервисов в экосистему iOS. Работа рассматривает методы оптимизации производительности и управления данными. Исследование охватывает технические аспекты, включая работу с API и инструментами Apple, с целью предоставления пользователям надежных и инновационных потоковых сервисов.

Ключевые слова. Мобильная разработка, архитектура, язык программирования Swift, потоковые сервисы, устройства iOS.

История музыки в мобильных устройствах берет свое начало в 1990-х годах, когда на рынке появились первые мобильные телефоны с поддержкой музыкальных мелодий. Однако, революция в этой сфере произошла с выпуском первого поколения iPhone в 2007 году, который стал первым мобильным устройством, способным воспроизводить музыку в высоком качестве и с возможностью загрузки музыкальных файлов из Интернета. Поток или стриминг является прямым способом воспроизведения аудиозаписей, получаемых в форме серии электронных файлов из Интернета и воспроизводимых на домашних или портативных устройствах без использования промежуточной процедуры скачивания на записывающие устройства, диски или другие локальные накопители [1].

С тех пор, музыкальные сервисы и приложения для мобильных устройств стали становиться все более популярными, предоставляя пользователям широкий выбор музыкальных жанров, альбомов, плейлистов и радиостанций. Такие сервисы, как iTunes, Spotify, Apple Music и другие, позволяют слушать музыку в режиме онлайн, скачивать и сохранять понравившиеся композиции, создавать собственные плейлисты и делиться ими с другими пользователями.

Сегодня, благодаря мощным мобильным устройствам и высокоскоростным сетям Интернета, музыкальные сервисы и приложения стали неотъемлемой частью нашей жизни. Более того, стриминговые сервисы не только обеспечивают доступ к бесконечному музыкальному контенту, но и предоставляют персонализированные рекомендации на основе наших предпочтений и поведения в приложении. Все это сделало мобильные приложения для потоковой передачи музыки незаменимым элементом современной культуры.

Существует несколько классификаций потоковых аудиосистем, которые можно разделить на две основные категории: по типу использования и по методу передачи потока информации. По типу использования подразделяются на следующие части:

- передача музыки;
- передачи речи;
- передачи звуковых эффектов.

По методу передачи потока информации подразделяются на юникаст, мультикаст и на броадкаст. *Юникаст* – метод передачи потока, при котором каждый поток передается отдельно для каждого получателя. Это означает, что каждый получатель получает свой собственный поток данных. *Мультикаст* – метод передачи потока, при котором один поток передается многим получателям одновременно. Этот метод обеспечивает более эффективную передачу потока в группе пользователей. *Броадкаст* – метод передачи потока, при котором поток передается всем получателям, находящимся в сети. Этот метод наиболее эффективен при передаче потоков данных в широковещательных сетях, таких как телевидение и радио [2].

Примеры использования потоковых аудио систем в современном мире:

1. Потоковая передача аудио является важным инструментом в видеоконференциях и вебинарах, которые позволяют участникам общаться в режиме реального времени и работать на расстоянии, не выходя из дома или офиса.

2. Потоковые аудиосервисы, такие как Spotify, Apple Music, Google Play Music, и другие, предоставляют доступ к музыкальным композициям и подкастам через интернет, где аудио передается потоком. Это делает музыку доступной в любое время и место, где есть подключение к Интернету.

3. Онлайн-игры с голосовым чатом используют потоковую передачу аудио для обеспечения связи между игроками. Это позволяет игрокам общаться в режиме реального времени, договариваться о тактике и координировать действия.

4. Вещание в режиме реального времени с использованием потоковых аудиосистем может быть применено для вещания спортивных событий, концертов, новостей и других событий. Это позволяет людям находиться в курсе происходящего и получать информацию мгновенно.

5. Потоковая передача аудио может быть использована в медицинских приложениях, таких как телемедицина, для обеспечения связи между врачами и пациентами. Это позволяет проводить консультации и диагностику на расстоянии, что особенно важно в ситуациях, когда пациент находится в удаленном районе или нуждается в экстренной медицинской помощи.

6. Онлайн-радиостанции используют потоковую передачу аудио для трансляции радиопрограмм через Интернет. Это позволяет слушателям получать доступ к радиостанции и ее программам в режиме реального времени, независимо от местоположения. Кроме того, некоторые радиостанции используют потоковые аудиосистемы для вещания музыкальных композиций в высоком качестве.

Таком образом потоковые аудиосистемы применяются во многих современных сферах жизни человека, упрощая ему использование и потребление большого количества информации.

Для реализации цели и решения задач, которые были поставлены в данном исследовании, было разработано мобильное приложение, предназначенное для управления системой потокового аудио.Необходимо спроектировать удобный пользовательский интерфейс, который должен быть несложен в использовании, а также в состоянии обеспечить эффективную работу пользователя с приложением, не требуя предварительного изучения справочной документации.

Обоснование выбора обеспечивающих технологий включает в себя определение необходимых программных средств.

При создании программного средства следует правильно определить технологии, с помощью которых продукт будет разрабатываться, так как технологии как правило достаточно узкоспециализированы и предназначены для конкретных задач.

Программный продукт представляет из себя мобильное приложение среднего масштаба, работающее с базой данных и имеющее интерактивный пользовательский дизайн.

При разработке мобильного приложения придерживались архитектурного решения VIPER. VIPER является реализацией Clean Architecture для iOS-проектов. Его структура состоит из: View, Interactor, Presenter, Entity, и Router [3]. Это действительно очень распределенная и модульная архитектура, которая позволяет разделить ответственность, очень хорошо покрывается unit-тестами и делает ваш код переиспользуемым.

Основными частями VIPER являются:

1. View – отображает то, что ему говорит докладчик, и ретранслирует пользовательский ввод обратно в Presenter.
2. Interactor – содержит бизнес-логику, указанную в варианте использования.
3. Presenter – содержит логику представления для подготовки контента для отображения (получения от Interactor) и для реагирования на пользовательские входные данные (путем запроса новых данных у Interactor).
4. Entity – содержит основные объекты модели, используемые Interactor.
5. Router – содержит логику навигации для описания того, какие экраны отображаются в каком порядке.

Это разделение также соответствует принципу единой ответственности. Interactor отвечает перед бизнес-аналитиком, ведущий представляет дизайнера взаимодействия, а View – перед визуальным дизайнером.

Ниже, на рисунке 1, приведена схема различных компонентов и того, как они связаны.

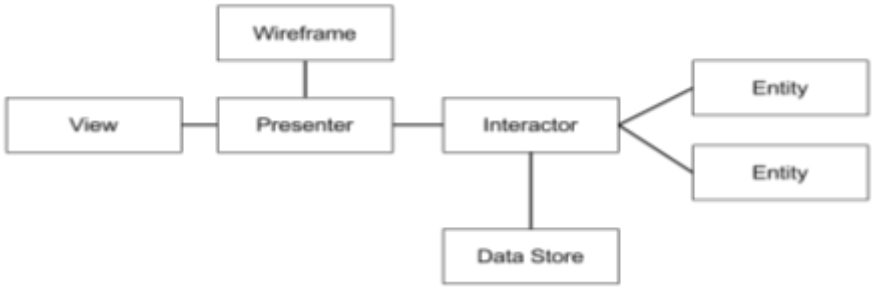

Рисунок 1 – Схема паттерна проектирования VIPER

При разработке приложения возникла потребность в защите данных при использовании сетевых запросов. Такие данные представляют собой пароль и логин пользователя передаваемые в теле запроса и легко перехватываемые системами мониторинга трафика мобильных устройств. Для решения проблемы безопасности был применен алгоритм шифрования и дешифрования. Такой

процесс кодирования сетевых запросов в мобильных устройствах называется обфускация. Данный процесс является процессом шифрования с секретным ключом. Аналогичный ключ находится на серверной стороне приложения и производит расшифровку данных с последующим использованием данных. Для этого применяется фреймворк Obfuscator, который при заданном секретном ключе в виде битов строковых элементов производит шифрование и дешифрование данных запроса спасая тем самым приложение от перехвата трафика и защиты персональных данных пользователя системы потокового аудио на базе устройств iOS.

При разработке использовался язык Objective-C. Objective-C – это язык программирования для разработки iOS и OS X приложений, который сочетает в себе все лучшее от C, но лишен ограничений, накладываемых в угоду совместимости с C. В Objective-C используются паттерны безопасного программирования и добавлены современные функции, превращающие создание приложения в простой, более гибкий и увлекательный процесс [4].

Objective-C разрабатывался несколько лет компанией Apple. Основой языка программирования послужили существующие компилятор, отладчик и фреймворки. Упростили процесс управления памятью с помощью механизма автоматического подсчета ссылок – Automatic Reference Counting (ARC). Фреймворки также подверглись серьезной модернизации. Objective-C начал поддерживать блоки, литералы и модули – все это создало благоприятные условия для внедрения современных технологий. Именно эта подготовительная работа послужила фундаментом для языка программирования, который будет применяться для разработки будущих программных продуктов для Apple.

Для внутренней организации хранения данных в виде базы данных использовался фреймворк Core Data. Core Data – это фреймворк, который используется для управления объектами уровня модели в вашем приложении. Он предоставляет обобщенные и автоматизированные решения общих задач, связанных с жизненным циклом объекта и управлением графом объекта, включая персистентность [5]. Чтобы хранить и извлекать большие объемы данных, вам нужна база данных. Обычно стандартные базы данных требуют своего собственного языка (SQL) для извлечения и сохранения данных. Если вы хотите напрямую использовать базу данных из вашего кода Objective-C, вам нужно будет вручную создать команды SQL, выполнять их, а затем вручную анализировать результат.

На рисунке 2 представлена диаграмма вариантов использования системы.

Актером на данной диаграмме является пользователь системы потокового аудио. Пользователь может зарегистрироваться или авторизоваться при входе в систему. Он может просмотреть все каналы, провести сортировку по любому полю, или найти любую запись в системе, также пользователь может просмотреть тексты песен, произвести перемотку. Пользователю доступно создание каналов, поиск треков, каналов, альбомов, пользователей, текстов песен с последующей сортировкой. Также имеется возможность прослушивание топ каналов, всех каналов или каналов пользователей. Пользователю предоставлена возможность оценки трека, канала, артиста с сохранением в профиле или избранном. Все прослушанные ранее медиа сохраняются и имеется возможность просмотра ранее воспроизведённого контента.

Произведем моделирование пользовательского интерфейса. Пользовательский интерфейс (UI) – это способ, которым вы выполняете какую-либо задачу с помощью какого-либо продукта, а именно совершаемые вами действия и то, что вы получаете в ответ. Программный интерфейс не только решает нашу проблему взаимодействия с приложением, но и делает это взаимодействие максимально комфортным. Нам важно наличие интерфейса, позволяющего при меньшем количестве усилий ознакомиться с возможностями приложения и понять основные принципы работы в нём.

Пользовательский опыт (User Experience) – это взаимодействие пользователя с продуктом (например, с мобильным приложением, веб-сайтом или устройством), включающее в себя все аспекты восприятия, использования и оценки продукта со стороны пользователя. Этот опыт охватывает все этапы взаимодействия пользователя с продуктом, начиная от первого знакомства с продуктом и заканчивая процессом его использования [26].

Основная цель UX – создать продукт, который будет удобен и привлекателен для пользователя, удовлетворяющий его потребности и ожидания, а также обеспечивающий максимально полезный и приятный опыт использования. Проектирование интерфейса (Interface Design) – это процесс создания интерактивного пользовательского интерфейса для продукта, который позволяет пользователям взаимодействовать с ним и выполнять задачи. Хороший интерфейс должен быть простым, понятным, интуитивно понятным и привлекательным для пользователей.

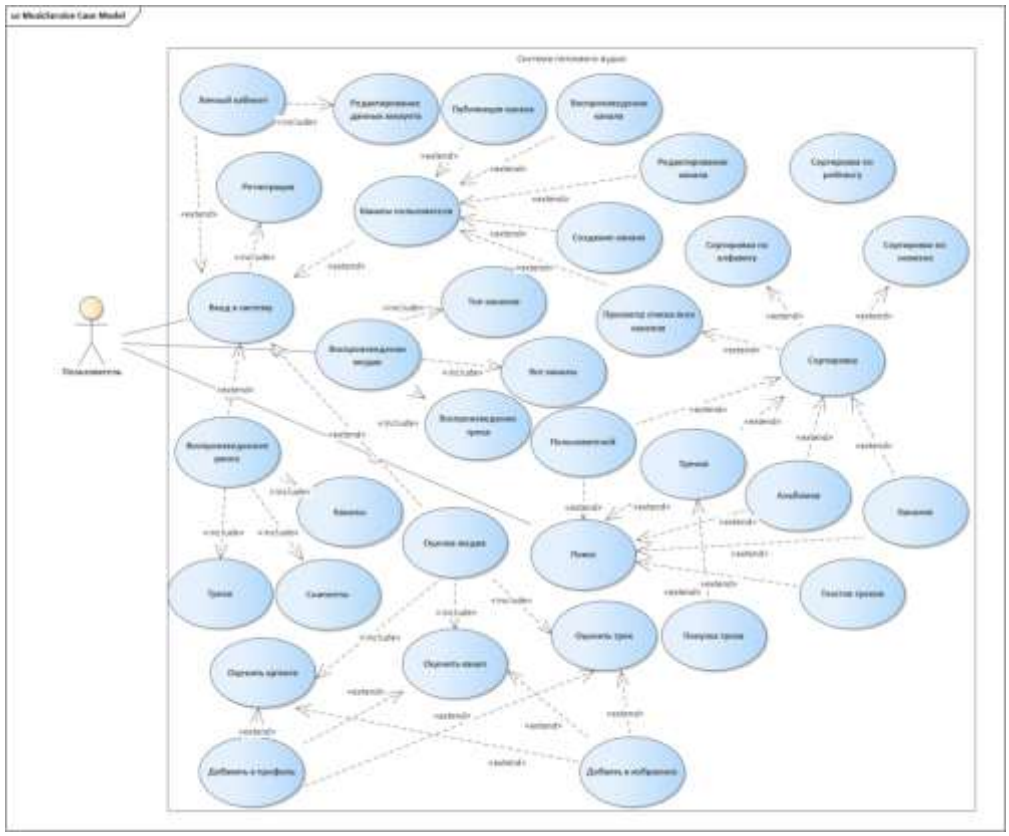

Рисунок 2 – Диаграмма вариантов использования системы потокового аудио

Сначала нам необходимо сориентироваться в элементах будущего интерфейса. Для этого определим задачу, исследовав предметную область, аудиторию и то, как она пользуется продуктом.

Создание мокапа это этап который позволяет быстро понять видение клиента и внести большое количество изменений до начала разработки интерфейса пользователя. Намечаем расположение кнопок, форм и других нужных элементов, а уже после подбираем цветовую палитру, шрифты, изображения, преобразуя всё это в удобный и красивый макет. То есть начинаем с варфрейма (план расположения элементов на странице), а заканчиваем созданием из этого плана красивой картинки. В случае разработки приложений под Android и iOS труд дизайнера облегчается гайдлайнами – правилами оформления и расположения элементов интерфейса, регламентом UX/UI, который был создан непосредственно экспертами по дизайну из Google и Apple.

Создание первичного расположения элементов проведем посредством использования инструмента Figma. Результат приведен на рисунке 3.

Рисунок 3 – Макет главной страницы с плеером

На макете присутствует навигационная панель, кнопка поиска, кнопки каруселей плеера, расположенная фото канала, кнопки для управления текущим и будущим воспроизведением. Опираясь на данный макет, можно приступить к верстке клиентской стороны приложения в специальный файлах storyboard.

В ходе выполнения данного исследования были исследованы процессы функционирования системы потокового аудио, были изучены стратегические цели и направления деятельности, а также организационная структура сервиса.

В результате проделанной работы было разработано мобильное приложение на базе устройств iOS. В процессе разработки учтены возможные ошибки и возможные попытки ввода некорректных значений. Все полученные ошибки были обработаны и созданы специальные уведомления и советы для пользователя приложения. Обработаны ошибки ограничения доступа определенным ролям пользователей системы.

На языке Objective-C была предусмотрена обработка исключительных ситуаций с помощью регулярных выражений, обработки исключений для введенной от пользователя информации, реализован процесс непрерывного поступления аудио информации на устройство.

Для разработки программного средства были смоделированы алгоритмы, реализующие бизнес-логику программного средства. Для улучшения взаимодействия пользователя с программным средством были предоставлены руководства по развёртыванию программного средства и его работе.

Поставленные задачи проекта были выполнены, а цель – повышение качества оказываемых услуг потоковой аудиосистемы, а также разработка приложения для мобильных устройств iOS, которое позволит пользователям слушать музыку и просматривать контент в режиме онлайн – достигнута.

Итогом данной работы является функционирующее приложение, автоматизирующее работу системы потокового аудио. В данном приложении реализованы базовые принципы объектно-ориентированного программирования, использованы стандартные и пользовательские функции, была реализована работа с базами данных, взаимодействие между сервером и клиентом. Для использования данной программы не требуются высококвалифицированные специалисты, так как работа с ней не требует наличия сертифицированной компьютерной программы и имеет интуитивно понятный интерфейс, запускаемый на любом мобильном устройстве под руководством операционной системы iOS.

Список использованных источников:

1. Тесля, П. Цифровая трансформация медиарынка и ее последствия / П. Тесля // ЭКО. – 2020. – № 2. С. 81–101.

2. Червинский В.В. Алгоритм оптимизации сети провайдера услуг IPTV с комбинацией режимов UNICAST и MULTICAST // Труды учебных заведений связи. 2018. Т. 4. No 4. С. 102–110.

3. Дударова, Х. Х. М. Архитектура VIPER / Х. Х. М. Дударова // Студенческий вестник. – 2023. – № 1-10(240). – С. 17–18.

4. Мясоедова, В. А. Сравнение языков программирования Objective-C и Swift для IOS разработки / В. А. Мясоедова // Modern Science. – 2021. – № 3–1. – С. 448–452.

5. CoreData for iOS [Электронный ресурс] Режим доступа: https://www.objc.io/books/core-data/.