**УДК 004.4’236**

**MVP приложения, как дополнительная возможность расширения охвата аудитории для малых предприятий**

Гегия Д., Батанов А.О.

Научный руководитель Волков М.Ю.

РТУ МИРЭА, Институт информационных технологий

***Аннотация:*** *В статье описывается проблема создания мобильных приложений для малых бизнесов и раскрываются возможные методы их решения.*

***Ключевые слова****: малый бизнес, охват аудитории, современные технологии, визуальные конструкторы, графические приложения, Adalo, визуальное программирование, nocode, мобильная разработка.*

Многие предприятия стремятся разработать для клиентов своё собственное мобильное приложение с целью, в частности. расширения потенциального охвата аудитории. При этом, учитываются финансовые возможности предприятия, назначение приложения, а также срок реализации проекта. Понятно, что чем выше ожидания от проекта, чем выше качества приложение в результате предприятие желает получить, тем дороже это будет стоить и тем дольше будет вестись разработка.

К главным задачам, которые необходимо решить, для устранения проблемы разработки мобильных приложений можно отнести следующие:

* уменьшение затрат и времени, требуемых для разработки приложения;
* минимизация рисков и увеличение стойкости приложения после его публикации;
* сохранение изначальной идеи и требуемого уровня качества финального приложения, без потери ключевой функциональности.

Одним из основных решений заданных задач является создание MVP приложения.

MVP – минимально жизнеспособный продукт, обладающий минимальными, но достаточными для удовлетворения первых потребителей функциями [1].

Разработка MVP приложения сводится к следующей последовательности:

- определяются границы необходимого функционала - от каких особенностей финального проекта можно отказаться, а без каких приложение не сможет жить. Определяется предел жизнеспособности продукта.

- выбираются технологии для разработки и происходит составление технического задания.

Приложение может получится более оптимизированным, но в таком случае оно будет более дорогостоящим, и разработка будет вестись намного дольше, либо же менее оптимизированным и стабильным, но в то же время дешёвым, а результат можно будет увидеть намного раньше. И здесь нет универсального решения, выбор делается в зависимости от возможностей и ожиданий конкретного проекта.

Для упрощения разработки часто используют визуальное программирование, повсеместно использующееся в создании графики, соглашаясь с недостатком гибкости во время разработки, но одновременно приобретая уменьшение порога входа или применяя конструкторы сайтов, которые благодаря низкой стоимости, простоты и наличия готовых решений устойчиво занимают своё место на рынке разработки.

Разработка полноценных приложений намного труднее, чем разработка сайтов связанной, например с достижением кроссплатформенности и т.д.

В визуальной разработке мобильных приложений на данный момент можно выделить три различных подхода.

Первый, простой подход – это поместить уже давно работающую систему с конструктором сайтов в оболочку мобильных приложений и в таком виде предоставлять продукт. По итогу мы имеем всё тот же функционал и результат – страницу сайта, которая становится доступна в виде приложения. При открытии приложения, мы открываем браузер, просто на нужной странице. Уведомления также проходят через браузер. Такая технология называется PWA [2].

PWA приложения являются наименее ресурсозатратными и простыми, но в то же время у них наиболее ограниченный функционал, по сравнению с другими визуальными решениями. Основным представителем таких типов конструкторов является Glide [3].

Вторым подходом является создание общей системы, «каркаса» приложения, лишь с базовым функционалом в области визуального программирования, для решения более сложных задач, просто нет нужных инструментов, а разработчик самостоятельно работает одновременно и с кодом, и с визуальными элементами. Минус такого подхода, заключается в том, что порог входа для создания приложений, всё ещё остаётся довольно высоким.

Такие приложения являются наиболее ресурсозатратными в визуальном программировании, и финальная стоимость разработки иногда может доходить по уровню, близкому к созданию полноценного приложения. Представитель такого подхода конструктор Flutter Flow [4].

Третий подход, для решения поставленной задачи пользуется преимуществами визуального программирования и пытается решить его недостатки.

С одной стороны, имеется полноценный визуальный конструктор, с широким ассортиментом элементов, решающих большой спектр задач, при этом на выходе получается стандартное приложение. И тем не менее, всё ещё остаётся возможность внедрить свой собственный компонент с помощью кода.

Приложения с таким подходом, характеризуются умеренной ресурсозатратностью и временем разработки. Однако такой подход для широкого спектра приложений является наиболее оптимальным.

Основным представителем является визуальный конструктор «Adalo» [5].

В визуальном программировании, также, как и в обычном, есть некоторые инструменты и технологии, без которых разработка полноценных приложений была бы невозможна. К таким технологиям можно отнести:

* подключение баз данных к приложению;
* аутентификация пользователей с защитой их данных;
* взаимодействие с бэкендом, наличие API;
* возможность добавления уникального кода.

Полноценное приложение не может существовать без баз данных. В любом случае, нужно где-то хранить пользователей, список категорий и товаров, изображения и так далее.

Все визуальные конструкторы приложений, в том, или ином виде пользуются именно реляционными базами данных. А вот способы реализации таких баз уже отличаются в зависимости от конкретного конструктора.

У Glide, например, с его PWA приложениями нет большой необходимости в наличии сложных и высококонфигурируемых инструментов для баз данных, поэтому там используются простые Google таблицы и точно такие же локальные Glide таблицы.

У FlutterFlow нет своего собственного решения для баз данных, поэтому предлагается интеграция с сервисами Firestore и Xano, который предоставляет собой полноценную визуальную СУБД.

В свою очередь у Adalo есть сразу несколько решений проблемы баз данных, так в Adalo существует своя собственная СУБД, прямо внутри конструктора, которая позволяет добавлять, изменять и удалять данные простыми действиями, которые уже внутри себя содержат запросы к локальному API, которых обычный разработчик не видит. Но в то же время, для более крупных проектов, производительности БД Adalo зачастую может не хватать, именно поэтому есть удобный инструментарий для интеграции с внешними базами данных, начиная от Xano, и заканчивая всё теми же Google таблицами.

Аутентификация и верификация пользователей бывает нужна во многих случаях, а в сфере коммерции, малых и средних бизнесов, которые являются основной аудиторией визуальной разработки она просто необходима.

Glide предлагает отдельные методы для аутентификации пользователей и специальный хешированный тип поля: пароль. Также присутствует встроенная система аутентификации по Google аккаунту.

Adalo поступает похожим образом, однако в этом конструкторе идут ещё дальше: при создании любого приложения уже есть готовый каркас базы данных пользователей с хешированным полем паролей, а также есть структура по регистрации, логину и даже сбросу пароля. Для всего этого есть специальные действия. Во FlutterFlow же, из-за отсутствия собственных баз данных, отсутствует и система аутентификации – эту задачу он перекладывает на внешние сервисы и те базы данных, в которых остальные данные и хранятся.

Весь бэкенд принято выносить за пределы самих конструкторов, оставляя возможности для интеграций путём собственного API и возможности отправки REST-запросов

Glide и Adalo подошли к разработке своего API довольно прямолинейно – так как у них присутствуют базы данных, они просто автоматически генерируют все возможные эндпоинты для полноценного доступа к ним. Очевидным минусом такого решения является полное отсутствие гибкости у эндпоинтов, так как свои собственные эндпоинты разработчик создавать не может. Кроме того, не поддерживается вложенность JSON ответа, то есть от дочерних элементов вернётся только их идентификатор, по которому далее придётся делать отдельный запрос со стороны бэкенда.

У FlutterFlow, в виду отсутствия собственных баз отсутствует и API.

Внешние вызовы, сторонних API выполняются действиями, посредством отправки REST запросов.

Ниша бэкенд визуального программирования также не осталась пустой, на рынке представлено множество уникальных подходов для составления полноценных бэкенд скриптов без единой строчки кода.

В качестве основных визуальных конструкторов можно выделить Integromat и Zappier.

Плюсы их использования заключаются в том, что разработчику нет нужды самостоятельно настраивать входящие и исходящие подключения, вручную обрабатывать запросы и заботиться о защите данных. Сами бэкенд конструкторы разрабатывают специальные модули и функции, уникальные для подключения к фронтенд конструкторам, в которых разработчику остаётся лишь указать секретные ключи от шифрования и выбрать действие из списка, которое будет этот модуль выполнять. А для построения самих скриптов используются уже другие модули и функции, отвечающие за разнообразную логику и подключение к сторонним сервисам.

Принципиальная разница между Integromat и Zappier заключается лишь в предлагаемых уникальных возможностях и разнообразии инструментов. Однако принципы построения скриптов у них одинаковые. Скрипт строится посредством последовательного расположения множества модулей, каждый из которых отвечает за своё, уникальное действие.

Одним из решений проблемы гибкости, является предоставление возможности более продвинутым пользователям, добавлять свой собственный код, для выполнения необходимых им задач, которые невозможно решить в текущей шаблонно ориентированной стратегии визуальных конструкторов.

Политика FlutterFlow заключается именно в перекладывании большей части работы по написанию собственных функций на пользователя, поэтому, свой код пишется и отлаживается напрямую в конструкторе, и именно из-за выбранной политики отхода от шаблонности, разработка собственного кода во FlutterFlow наименее ограничена, по сравнению с другими конструкторами.

Glide предоставляет возможность написания собственных небольших скриптов на Javascript, при этом внутри самого конструктора, необходимый инструментарий для размещения и хранения кода отсутствует, поэтому предлагается хранить его на стороннем хостинге.

Adalo же, делает большой упор именно на разработчиков компонентов, справедливо отмечая их вклад в развитие и популяризацию конструктора в целом, путём расширения его потенциальных возможностей.

Так, конструктором предоставляется свой пакет, реализующийся посредством Node JS, и подготавливающий структуру для работы над полноценным компонентом, или библиотекой, которые пишутся на React native фреймворке. После написания и локальной отладки, посредством всё того же пакета, осуществляется публикация компонента на сервера самого конструктора, для дальнейшего пользования, без необходимости заботиться о хостинге. Разработчик может опубликовать как приватный, так и публичный компонент. Приватный компонент привязывается к аккаунту разработчика, и им можно в индивидуальном порядке делиться с другими. Публичный же компонент, после дополнительной проверки со стороны администрации, загружается в магазин Adalo, где его можно как оставить бесплатным, так и установить на него цену, при этом, не отдавая ни процента с доходов самому конструктору.

Несомненно, направление визуальной разработки мобильных приложений является довольно новым и перспективным. Однако на данный момент существуют неразрешённые проблемы такого подхода, которые не позволяют и вероятно не позволят ему в ближайшее время заменить классическое программирование.

В то же время, свою нишу на рынке разработки мобильных приложений они начинают занимать уже сейчас, и будут лишь укреплять свои позиции в дальнейшем. Причиной тому является высказанный в работе тезис о том, что некоторые заказчики готовы смириться с шаблонностью и отсутствием гибкости в приложении, а также невозможностью реализовать некоторые функции в угоду крайне низкому порогу входа в разработку, быстроте разработки, и стоимости, ниже в несколько раз. Данный вывод, в основном, относится к малому бизнесу, которые не готовы платить большую сумму, ожидая длительно результата по разработке мобильных приложений.

**Список использованных источников:**

1. Ries E. Lessons learned: Minimum Viable Product: a guide [Электронный ресурс]. — URL: <http://www.startuplessonslearned.com/2009/08/minimum-viable-product-guide.html>
2. Vaillant S. Software AG Technology Radar [Электронный ресурс]. — URL: <https://techradar.softwareag.com/>
3. Документация Glide [Электронный ресурс]. — URL: <https://docs.glideapps.com/all/start-here/welcome>
4. Документация FlutterFlow [Электронный ресурс]. — URL: <https://docs.flutterflow.io/>
5. Документация Adalo [Электронный ресурс]. — URL: <https://help.adalo.com/>

© Гегия Д., Батанов А.О., 2022