MINISTERUL EDUCAŢIEI AL REPUBLICII MOLDOVA

UNIVERSITATEA DE STAT „ALECU RUSSO” DIN BĂLŢI

FACULTATEA DE ŞTIINŢE REALE, ECONOMICE ȘI ALE MEDIULUI

CATEDRA DE MATEMATICĂ ȘI INFORMATICĂ

**UTILIZAREA FRAMEWORKULUI JS PENTRU CREAREA APLICATIEI MOBILA HIBRIDA**

**TEZĂ DE LICENTA**

**Autor:**

Studenta al grupei IS31Z

**Stanislava VOVOD**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(semnatura)

**Conducător științific:**

**Corina NEGARA**

dr., conf., univ.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(semnatura)

**Dumitru STOIAN**

asist. univ.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(semnatura)

**BĂLȚI, 2021**

МИНИСТЕРСТВО ОБРАЗОВАНИЯ, КУЛЬТУРЫ И ИССЕЛЕДОВАНИЯ

РЕСПУБЛИКИ МОЛДОВА

БЕЛЬЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМ. «АЛЕКУ РУССО»

ФАКУЛЬТЕТ ТОЧНЫХ НАУК, ЭКОНОМИКИ И ОКРУЖАЮЩЕЙ СРЕДЫ

КАФЕДРЫ МАТЕМАТИКИ И ИНФОРМАТИКИ

**ИСПОЛЬЗОВАНИЕ ФРЕЙМВОРКА JS ПРИ СОЗДАНИИ ГИБРИДНОГО МОБИЛЬНОГО ПРИЛОЖЕНИЯ**

**ЛИЦЕНЗИОННАЯ РАБОТА**

**Автор:**

Студентка группы IS31Z

**Станислава ВОВОД**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

**Научный руководитель:**

**Корина НЕГАРА**

др., конф., унив.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

**Дмитрий СТОЯН**

асист. унив.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

**БЭЛЦЬ, 2021**

Controlată:

Data „\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_ 2021

Conducător ştiinţific: dr., conf. univ., Corina NEGARA

asist. univ., Dumitru STOIAN

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Aprobată

şi recomandată pentru susţinere

la şedinţa Catedrei de ştiinţe economice

Proces-verbal nr. \_\_\_\_\_\_ din „\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_202\_\_

Şeful catedrei dr., conf. univ. Corina NEGARA

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(semnatura)

*ADNOTARE*

Lucrare autorizată. „ Utilizarea frameworkului JS pentru crearea aplicației mobila hibrida.”, Vovod Stanislava. Specialitate: 0444.1-Informatică (științe exacte). Bălți, 2021. Structura operei. Lucrarea autorizată conține o introducere, două capitole, concluzii, o listă de referințe din 19 de surse, 54 de pagini din textul principal, 10 figuri.

Cuvinte cheie: frameworkuri JS, comparare frameworkurilor, aplicație web, aplicație mobila, aplicație mobile hibrida.

Scopul lucrării este analizarea si compararea frameworkurilor JS; alegerea unui framework la care se dezvolta o aplicație mobila hibrida demo care utilizează cadrul Vue.js.

*АННОТАЦИЯ*

Лицензионная работа. «Использование фреймворкa JS при создании гибридного мобильного приложения.», Вовод Станислава. Специальность: 0444.1-Информатика (точные науки). Бэлць, 2021. Структура работы. Лицензионная работа содержит введение, три главы, выводы, список использованной литературы из 19 источников, 54 страниц основного текста, 10 рисунков, одну таблицу.

Ключевые слова: JS-фреймворки, сравнение фреймворков, веб-приложение, мобильное приложение, гибридное мобильное приложение.

Целью работы является анализ и сравнение JS-фреймворков; выбор фреймворка для разработки гибридного мобильного демонстрационного приложения, использующего фреймворк Vue.js.

*ANNOTATION*

Licensed work. " Using Google Services in hybrid mobile applications JS on Vue.js", Vovod Stanislava. Specialty: 0444.1-Computer science (exact sciences). Balti, 2021. The structure of the work. The licensed work contains an introduction, three chapters, conclusions, a list of references from 19 sources, 54 pages of the main text, 10 figures.

Keywords: JS frameworks, framework comparation, web application, mobile application, hybrid mobile application.

The scope of this work is to analyze and compare JS frameworks; choosing a framework to develop a hybrid mobile demo application that uses the Vue.js framework.

Aprobat:

Șeful catedrei de matematică și informatică

dr., conf. univ. C.Negara

“\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_202\_\_

**Graficul calendaristic de executare a tezei de licență**

Tema tezei de licența: UTILIZAREA FRAMEWORKULUI JS PENTRU CREAREA APLICATIEI MOBILA HIBRIDA

confirmată prin ordinul rectorului USARB nr.\_\_\_ din „\_\_\_\_\_\_\_\_\_\_\_\_\_”

Termenul limită de prezentare a tezei de licență la Catedra de matematică și informatică „\_\_\_\_\_\_\_\_\_\_\_”.

**Etapele executării tezei de licență:**

|  |  |  |
| --- | --- | --- |
| Etapele | Termenul de realizare | Viza de executare |
| 1. Stabilirea temei; fixarea obiectivelor; selectarea surselor de informare. | 11.09.20 | realizat |
| 1. Investigația cadrului teoretic al cercetării; expunerea cadrului teoretic al cercetării. | 03.10.20 | realizat |
| 1. Întocmirea problemei cercetării; stabilirea tipului de cercetare. | 21.10.20 | realizat |
| 1. Specificarea unităților studiate. | 10.11.20 | realizat |
| 1. Alegerea metodelor de cercetare. | 18.11.20 | realizat |
| 1. Culegerea datelor; selectarea modalităților de prelucrare a datelor; stocarea datelor; analiza datelor | 21.01.21 | realizat |
| 1. Elaborarea concluziilor și a recomandărilor practice. | 15.03.21 | realizat |
| 1. Susținerea preventivă a tezei. | 30.04.21 | realizat |

Student (ă) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(semnătura)

Conducători științifici \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(*semnătura*)

СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 9](#_Toc70621902)

[ВВЕДЕНИЕ 11](#_Toc70621903)

[1. ГИБРИДНЫЕ МОБИЛЬНЫЕ ПРИЛОЖЕНИЯ 14](#_Toc70621904)

[1.1. Гибридные мобильные приложения 14](#_Toc70621905)

[1.2. Сравнение гибридных, нативных и веб приложений 16](#_Toc70621906)

[1.3. Примеры 23](#_Toc70621907)

[2. ФРЕЙМВОРКИ JAVASCRIPT 25](#_Toc70621908)

[2.1. Общая характеристика фреймворков JavaScript 25](#_Toc70621909)

[2.2. Фреймворк VUE.js 25](#_Toc70621910)

[2.3. Фреймворк Angular 26](#_Toc70621911)

[2.4. Vuetify - фреймворк для создания гибридных мобильных приложений 27](#_Toc70621912)

[2.5. Фреймворк React 27](#_Toc70621913)

[2.6. Сравнение фреймворков 28](#_Toc70621914)

[2.7. Почему я выбрала вью 31](#_Toc70621915)

[2.8. Платформа для разработки гибридных мобильных приложений – Apache Cordova 32](#_Toc70621916)

[2.9. Google сервисы как неотъемлемая часть гибридного мобильного приложения 35](#_Toc70621917)

[3. ОПИСАНИЕ Демонстрационного ПРИЛОЖЕНИЯ 44](#_Toc70621918)

[3.1. Описание структуры приложения 44](#_Toc70621919)

[3.2. Черты гибридных технологий в новостном приложении 45](#_Toc70621920)

[3.3. Чем обосновано предпочтение гибридных технологий нативным 45](#_Toc70621921)

[3.4. Установка и настройка среды разработки 46](#_Toc70621922)

[3.4.1. Node Package Manager (NPM) 46](#_Toc70621923)

[3.4.2. Инструменты командной строки (CLI) 46](#_Toc70621924)

[3.4.3. Использование Cordova для определения гибридной трансляции 46](#_Toc70621925)

[3.4.4. Создание приложения 47](#_Toc70621926)

[3.4.5. Создание проекта в Firebase 48](#_Toc70621927)

[3.4.6. Интеграция Firebase 48](#_Toc70621928)

[Заключение 53](#_Toc70621929)

[Библиография 55](#_Toc70621930)

# ВВЕДЕНИЕ

В наше время прогресс достиг невиданных высот. С появлением интернета технологии развиваются с такой скоростью, что уследить за всеми практически невозможно. Каждый год появляются все новые и более инновационные площадки, сервисы, приложения, позволяющие делиться информацией, обучаться, работать, веси бизнес и многое другое, не выходя из дома. Однако, все возрастающая популярность приложений привела к появлению их различных вариаций, а именно, появились такие понятия, как нативные, гибридные и веб приложения.

Согласно последнему опросу Red Hat, одного из мировых провайдеров программных решений с открытым исходным кодом, сегодня 50% компаний нуждаются в специалистах по мобильной разработке или менеджерах мобильных проектов. Интересно, что в этой доле компаний нативные разработчики слабо востребованы. Лидирующие позиции занимают front-end специалисты — 32% опрошенных голосуют за front-end.

Данные исследований Gartner также подтверждают повышение интереса со стороны бизнеса к мобильной разработке. По их прогнозам, к концу 2017 года спрос на разработку мобильных приложений будет расти в 5 раз быстрее, чем штат IT компаний, способных удовлетворить этот спрос.

Исходя из динамики развития мобильных приложений, мировые тенденции подтверждаются увеличением спроса на мобильную разработку, растет скорость создания приложений, повышается интерес к простым решениям с доступной стоимостью разработки и удобным, понятным интерфейсам. Всё это открывает возможности для совершенствования гибридных приложений, которые все увереннее закрепляются на мировом рынке и не сдают свои позиции. Отдавая предпочтение тем или другим, важно отслеживать состояние рынка и правильно расставлять приоритеты по скорости запуска проекта, стоимости и производительности.

*Цель*данной работы состоит в исследовании гибридных web технологий и возможностей фреймворков JS для их сравнительного анализа и последующей разработки демонстрационного гибридного мобильного приложения на фреймворке Vue.js.

Для выполнения поставленной цели были спланированы следующие *задачи*:

* Изучение основ гибридных web технологий;
* Исследование возможности использования гибридных web технологий в качестве замены нативных технологий;
* Сравнительный анализ возможностей гибридных технологий в разных фреймворках;
* Аргументы в пользу выбора фреймворка Vue.js для дальнейшей разработки;
* Создание мобильного приложения используя гибридные web технологии фреймворка Vue.js.

*Ценность работы:*Данная работа полезна своей теоретической частью, так как было произведено исследование основных технологий, используемых для создания гибридных web приложений, а также сравнительный анализ фреймворков JS и выбора самого оптимального для разработки гибридного мобильного приложения. Практическая же часть будет полезна тем, что в ней описано назначение и процесс создания гибридного мобильного приложения на выбранном фреймворке JS – Vue.js.

*Актуальность:* данная работа является актуальной и будет таковой еще долгое время, так как мобильные устройства в нынешнее время предпочитает все большее количество потребителей. Они портативные, компактные и позволяют пользователям использовать их в любом месте и в любое время. С этим так же растет и спрос на мобильные приложения, а гибридные мобильные приложения - это набор технологий, которые призваны сочетать в себе основные преимущества мобильной платформы будучи обычным web приложением, что дает им большое преимущество на ряду с другими технологиями.

Основываясь на поставленных целях, работа была разделена на 3 главы:

В первой главе было проведено исследование технологий в составе гибридных мобильных приложений таких как Firebase API, Google OAuth и другие.

Во второй главе описаны гибридные web приложений, и таких технологий так Vue, Vuetify и Cordova для создания мобильного веб приложения.

В третьей главе описывается процесс создания гибридного web приложения используя данный набор технологий:

В ходе данной работы будет разработано приложение, которое является аналогом twitter и включает в себя набор функций для создания и размещения постов. Это будет гибридное web приложение, включающее в себя авторизацию, добавление постов, просмотр карт и геолокацию.

# ГИБРИДНЫЕ МОБИЛЬНЫЕ ПРИЛОЖЕНИЯ

* 1. Гибридные мобильные приложения

Гибридные мобильные приложения - это приложения, которые создаются с единой базой кода, позволяющей запускать их на нескольких операционных платформах и устанавливаются на устройство, как и любое другое приложение. Они являются одновременно нативными и веб-приложениями. По своей структуре такие приложения состоят из нативных элементов собственных приложений, разработанных для конкретной платформы, такой как iOS или Android, но включают в себя элементы веб-приложений, веб-сайтов, которые встроены в приложение, но не установлены на устройстве, таким образом они становятся доступными только с интернет-подключением и без него не работают [9].

Гибридные приложения развертываются в собственном контейнере, который использует мобильный объект WebView. При использовании, данный объект отображает веб-контент с помощью веб-технологий (CSS, JavaScript, HTML, HTML5). В сущности, это адаптированные веб-страницы, отображающиеся с помощью технологии WebView. Веб-контент может отображаться либо сразу после открытия приложения, либо только для определенных частей приложения [10]. Для доступа к аппаратным функциям устройства (акселерометр, камера, контакты ...) добавляются собственные элементы UI для каждой платформы, чтобы пользователь удобного взаимодействовал с приложением. Гибридные приложения также могут полагаться на платформы, которые предлагают API-интерфейсы JavaScript, если эти функции вызываются в WebView. По итогу, смысл гибридной разработки заключается в том, что разработка происходит на WebView мобильного телефона с помощью HTML5/Javascript/СSS, затем просто «пакуем» подобного рода разработку в нативную «обертку».

Родство гибридных приложений с веб-приложениями, позволяет просто и быстро вносить коррективы. Таким образом, разработчикам не приходится повторно размещать приложение в магазине ради устранения ошибок предыдущей версии, как это происходит в случае с нативными приложениями.



Рис.2.1. Гибридное приложение.

Преимущества гибридных приложений:

1. Стоимость и скорость создания приложения;
2. Количество задействованных в разработке людей;
3. Кроссплатформенность, позволяющая обратить внимание на продукт большего количества пользователей за счет привлечения людей с различными предпочтениями и возможностями;
4. Присутствует опция автономного обновления, что позволяет пользователям не проверять наличие обновлений вручную и значительно улучшает показатель UX.
5. Наличие наибольшей функциональности и персонализации для пользователя;
6. При разработке гибридного мобильного приложения есть возможность создавать его на нескольких платформах, а не ограничиваться одной, как в случае с нативными приложениями;
7. Гибриды - хорошая опция для разработчиков, которые создают визуально насыщенные приложения, например, игры (которые не будут хорошо работать в виде веб-приложений).

Выбор подходящей модели мобильного приложения - это очень важный этап в его разработке, на который влияют несколько факторов, таких как:

1. техническая оценка разработчиков;
2. потребность в доступе к информации на устройстве;
3. влияние скорости интернета на приложение;
4. является одноплатформенным или мультиплатформенны приложением.
5. некорректная работа при отсутствии интернет-соединения;
6. средняя скорость работы на фоне нативных;
7. минимализм в отношении визуальных элементов.

Процесс разработки гибридного приложения [11] соответствует процессу разработки веб-сервиса с дополнительным этапом разработки интерфейса обложки мобильного приложения. В результате владелец проекта имеет и сайт веб-сервиса, и мобильное приложение.

* 1. Сравнение гибридных, нативных и веб приложений

Широкая популярность гаджетов вполне закономерно приводит к тому, что на рынок приходят новые производители. А вместе с ними – их собственные разработки, которые находят себя в приложениях. Это так называемые нативные приложения, то есть доступные только на одной определенной платформе или операционной системе. Простой пример: площадка Android на своих использует в качестве стартового браузера сервисы Google, а тот же IOS – браузер под названием Safari. Такой подход касается не только браузера, но и многих других приложений, возможность установить которые ограничивается их «материнской» системой.

Еще недавно компании ориентировались только на опыт в разработке нативных приложений под Android и iOS, но сейчас лучшей заменой для них становятся кросс-платформенные решения, так как для бизнеса в приоритете находится разработка относительно простых и недорогих приложений за короткое время. Компании замотивированы на конкурентную борьбу в своей сфере бизнеса, поэтому они предпочитают приложения тактического, а не стратегического характера. Другими словами, в условиях высокой конкуренции компаниям важно быть первыми и максимально доступными в мобильных решениях для своих потребителей. Следуя этой логике, они расставляют приоритеты в пользу быстрого создания приложений.

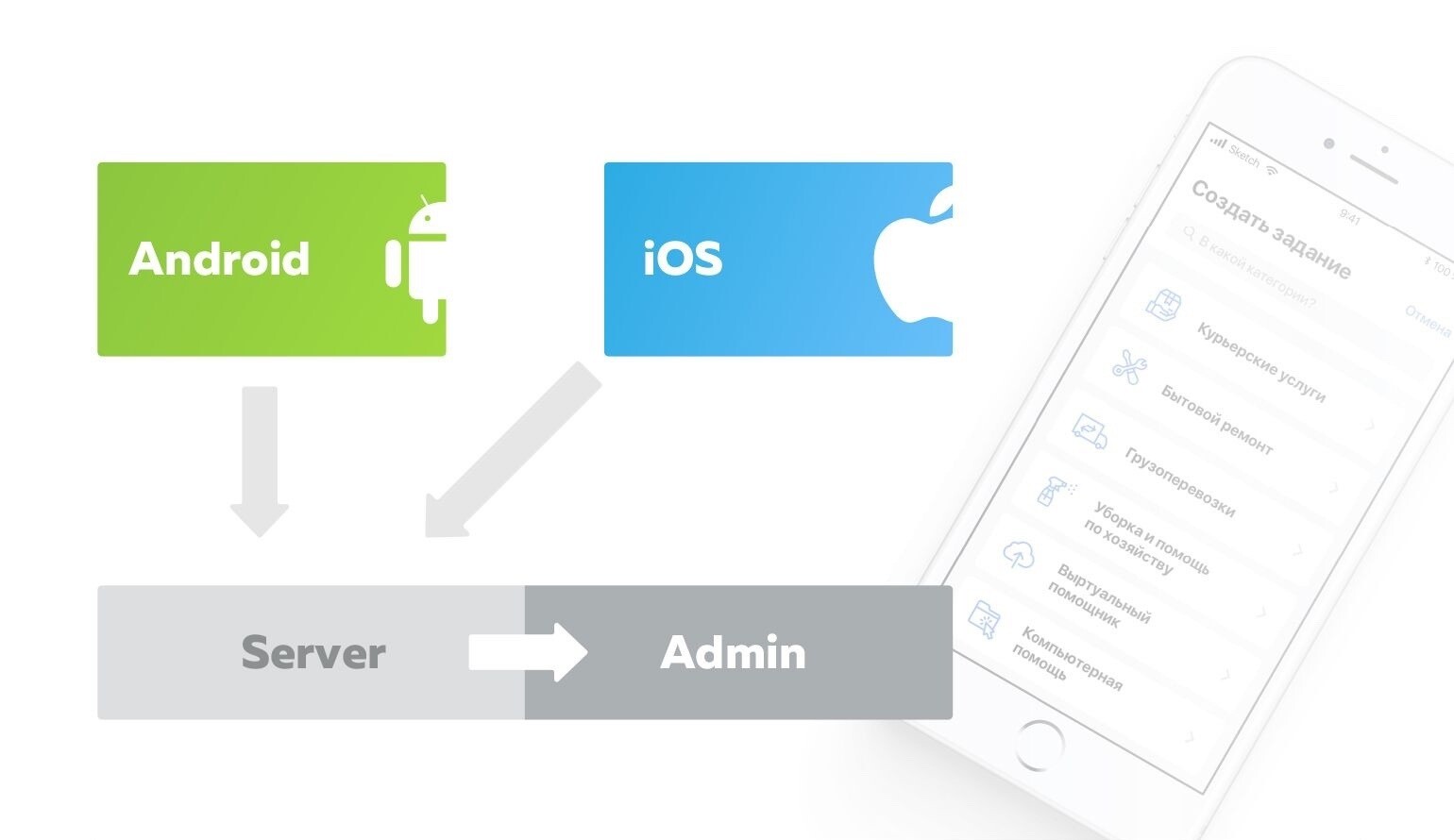
Фактически гибридные мобильные приложения это мобильные приложения с использованием веб-технологий. Обычный пользователь даже не замечает разницы, поскольку гибридное приложение устанавливается на устройство и выглядит в точности, как нативное. Тем не менее, разница существует, и это необходимо учитывать, при выборе подхода.

Гибридные приложения по своей сути более универсальны и чаще всего подходят для разных систем. Во многом, гибридное приложение – это своего рода синтез мобильной направленности и веб-приложений, а именно форм, доступных только для одной системы, но уже браузерной или компьютерной.

Нативные приложения загружаются через магазины приложений (App Store, Google Play или его аффилиаты, магазин приложений Windows и т.д.) и устанавливаются в ПО смартфона. Важным отличием является то, что нативные приложения разрабатываются специально под конкретную платформу (например, под iOS для iPhone, под Android для устройств под управлением ОС Android или под Windows для Windows Phone и т.д.) и требуют от разработчика специальных знаний и умений для работы в конкретной среде разработки (xCode для iPhone, eclipse, Android Studio или IntellijIdea для устройств на Android); более того, используется только «родные» языки программирования для написания таких приложений. Естественно, сам процесс при этом более трудоемкий. Таким образом, нативные приложения всегда «заточены» под конкретную ОС и органично выглядят на смартфоне. Такие приложения с легкостью могут использовать все функции ПО смартфона (камера, микрофон, акселерометр, геолокация, адресная книга, плеер и т.д.), и при этом более бережно расходуют ресурсы телефона (аккумулятор, память). В зависимости от назначения приложения предполагают или не предполагают наличие интернет-соединения

Веб-приложения не случайно называют html5-приложениями. Это, по сути, сайт, оптимизированный под смартфон. Пользовательский интерфейс создается при помощи стандартных веб-технологий. Их не нужно загружать из магазина приложений, но они могут находиться в специальных магазинах веб-приложений, которые есть у некоторых современных браузерах, например у Chrome. Веб-приложения используют для работы браузер телефона. Главной особенностью таких приложений является их кроссппатформенность — возможность работать на всех устройствах, без дополнительной адаптации.

Любой многопользовательский сервис состоит из клиентских приложений и серверной части. Если смотреть совсем широко, то туда ещё подключаются сайт, телефония, api сторонних проектов, но для того, чтобы сравнить натив с гибридом нам это не нужно, поэтому рассмотрим самый простой вариант:



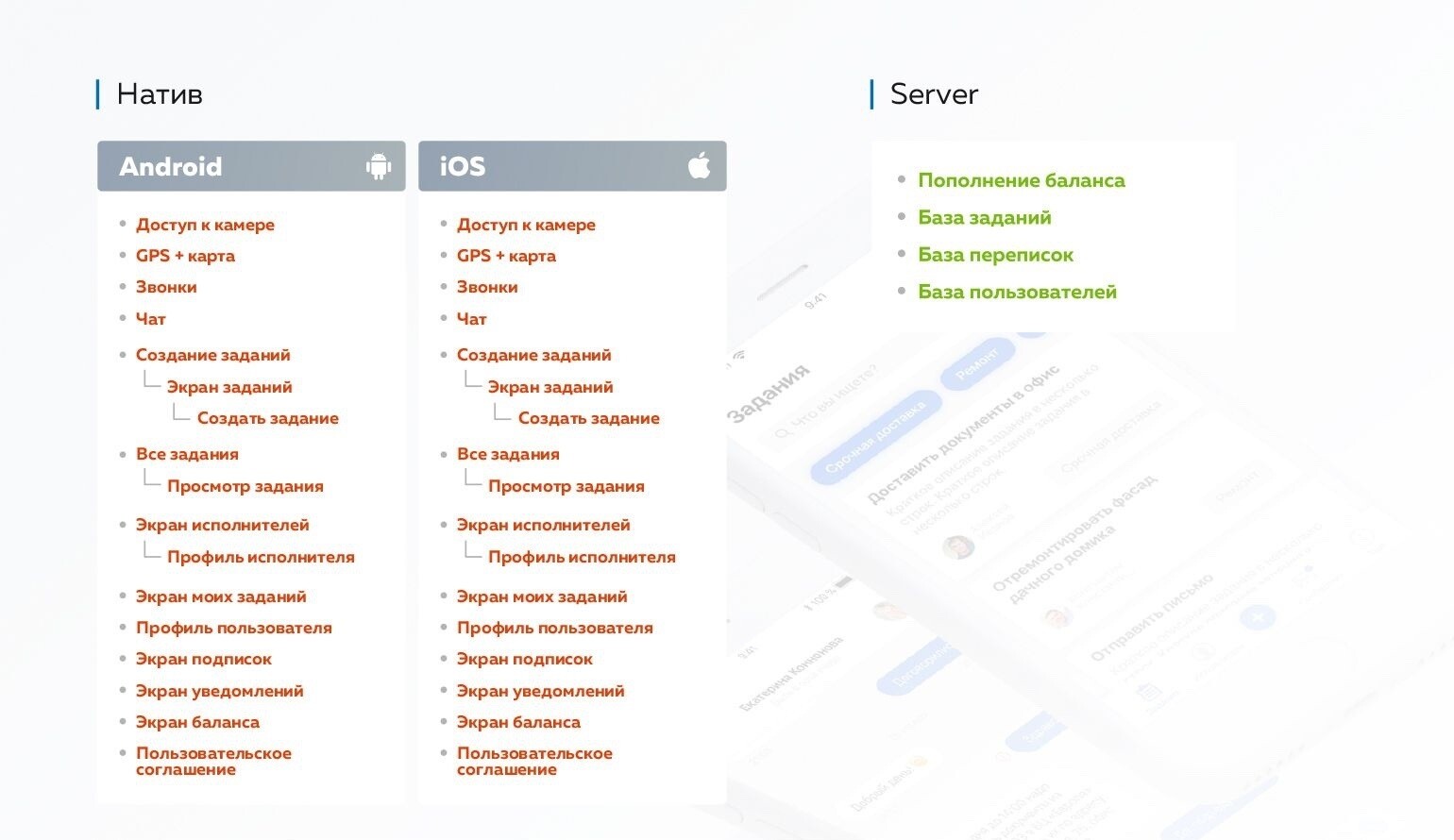
На стороне клиентской части располагается приложение iOS или Android, а со стороны сервера, собственно, серверное ПО и админка для управления данными.

Нативное приложение можно сравнить с приложениями для ПК. Купили MS Word, скачали установщик на компьютер, установили его и пользуетесь. По большому счёту вам уже без разницы есть интернет или нет - программа запускается с вашего ПК. Тут тоже самое - скачали со сторов, установили и весь функционал доступен с вашего телефона. То есть приложение скачивается "от и до" и, если это не заложено логикой приложения, интернет ей не нужен. Самый простой пример - это одиночные игры. Скачал и играешь.

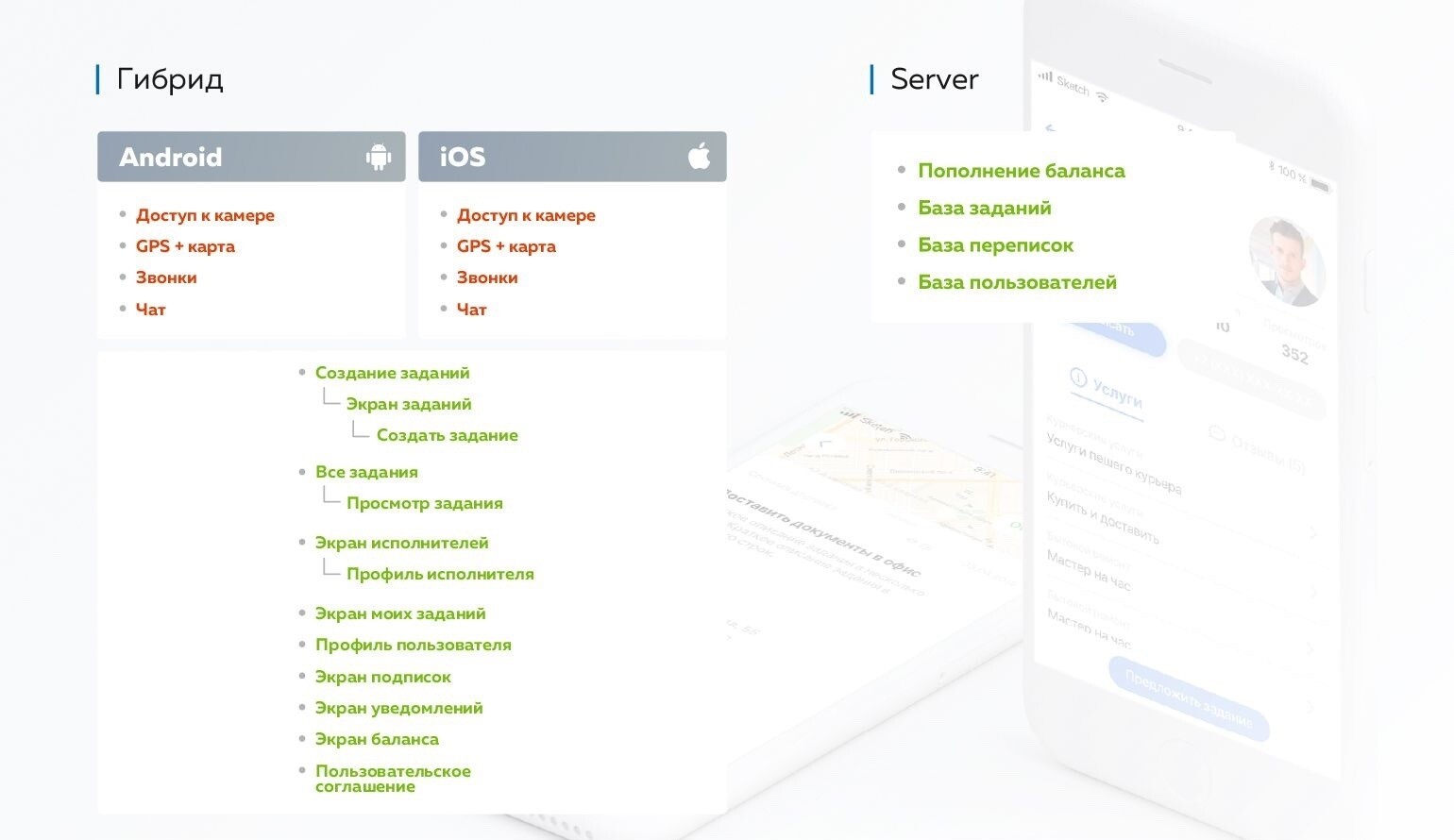
Гибрид - это скорее подход к программированию, чем какой-то особенный вид приложения. Его принцип заключается в том, что всё что можно (читай "все функции не связанные с железом") программируется на стороне сервера, а на стороне клиента остаётся только необходимый минимум. Если рассматривать типовой мобильный магазин, то GPS и функция отображения данных будет на стороне приложения, а каталог, карточки товаров и т.д. будут грузиться с сервера.

Давайте для примера рассмотрим как бы в гибриде и нативе выглядела наша платформа для создания сервиса поиска исполнителей "Сервис ПИ". Красным выделено то, что нужно делать на стороне приложения для телефона, а зелёным - на стороне сервера.

В нативе:



В гибриде:



Отсутствие требований к постоянному наличию интернета (приложения шагомеры, пульсомеры, игры-кликеры, органайзеры и т.д.)

Вообще, как я и писал выше, моя команда занимается разработкой гибридных приложений. Это и не случайно, ведь у бизнес-приложений нет требований ни к вычислительной мощности телефона (нагрузка не выше, чем у браузера), ни к отсутствию интернета (для того, чтобы сделать заказ в мобильном магазине в любом случае нужен интернет). Если же клиент приходит к нам с заказом игры, то, конечно же, мы отправляем его к нативщикам. Самое важное, что я рекомендую - это грамотно выбрать подход к разработке, чтобы потом не было мучительно больно за ошибки, сделанные на старте.

Как видно из картинок единственная разница между нативом и гибридом - это объём функционала, который реализуется на стороне телефона. Но есть большое "НО". Весь "зелёный" функционал, который мы в гибриде перенесли на сервер делается 1 раз для обеих платформ, а красный - для каждой платформы пишется отдельно. Давайте разберём какие из этого можно сделать финансовые выводы.

Таблица 2. Преимущества и недостатки гибридного, веб и нативного приложения

|  |  |  |
| --- | --- | --- |
| **Преимущества** | **Недостатки** | |
| **Нативное** | * Максимальная функциональность и скорость работы * Не требуется интернет- соединение для использования * Имеет доступ к ПО смартфона (GPS, плеер, камера) Распространение через магазины приложений | * Выше стоимость и длиннее сроки разработки * Требует от разработчика знаний определенной среды программирования * Работает только с одной платформой * При косметических изменениях необходимо выпускать обновление |
| **Веб (HTML5)** | * Кроссплатформенность * Не требует загрузки из магазина мобильных приложений * Можно легко адаптировать обычный сайт * Легче найти веб-разработчика нежели разработчика под определенную платформу Простота создания и поддержки | * Требует подключения к интернету * Не имеет доступа к ПО смартфона * Не может отправлять push-уведомления * Должен быть запущен интернет-браузер * При продаже требуется использование своей платежной системы |
| **Гибридное** | * Функциональность нативного приложения на независимой платформе * Запускается не из браузера в отличии от веб приложения * Возможность независимого обновления * Распространение через магазины приложений | * Загружается из магазина мобильных приложений (необходимо соответствовать требованиям) * Разработчик должен быть знаком с разными API |

Таблица 1

Если для работы приложения необходимо использовать мощности устройства, о которых мы писали ранее, или для его работы очень важна скорость обработки информации (игры, социальные сети, геолокационные сервисы, сервисы обмена фотографиями и т.д.), то необходимо делать нативное приложение. Если скорость работы не так важна, то делайте лучше гибридное приложение. А если вам просто важно, чтобы у пользователя была возможность получить от вас необходимую информацию при наличии интернета через мобильный телефон, и его это устраивает, то спокойно делайте веб-приложение!

Часто производители делают нативные приложения платными. И если вы планируете последовать этому плану, помните, что люди в принципе не очень любят платить деньги за какой-либо софт, пускай и качественный. В то время как разработка качественного приложения под конкретную платформу действительно стоит денег. Так что придется хорошо продумать процесс монетизации, а лучше его протестировать перед официальным запуском.

Учитывая свои нюансы разработки нативного приложения, акцентируют внимание на применяемом языке программирование. Например, для платформы IOS актуально и правильно применить Objective-c (реже встречается использование Swift).

Понимая особенность реализованного во время разработки приложения, можно быстро определиться и решить для себя гибридное или нативное приложение будет приносить пользу. При этом учитывается сразу несколько факторов: стоимость самой разработки, преследуемая цель, необходимость работы на нескольких платформах и, конечно же, целевая аудитория.

Плюсы и минусы гибридного подхода

По большему счёту, к основным достоинствам гибридного подхода относятся:

1. Кросс-платформенность. Сделав одно приложение, можно экспортировать его под любую операционную систему – iOS, Android, Windows Phone, BlackBerry.
2. Доступная стоимость разработки – в разы меньше, чем при нативном подходе.
3. Использование одного языка – JavaScript.
4. Доступ к основным данным мобильного устройства: GPS, камера, телефонная книга и т.д.
5. Работа в оффлайн-режиме.
6. Возможность распространять приложение через официальные магазины приложений.
7. Быстрый вывод на рынок: гибридные приложения разрабатываются быстрее, так как в ходе процесса используются стандартные веб-технологии, которые легко обслуживать в долгосрочной перспективе.
8. Дистрибуция для нескольких платформ: этот вид приложений распространяется через оба магазина, что позволяет охватить большее число пользователей.

Однако у гибридной разработки есть свои слабые места:

1. Сниженная скорость работы в отличие от нативных приложений.
2. Меньше возможностей для интеграции с «начинкой» мобильного устройства.
3. Проблематичная верстка адаптивного дизайна. Несмотря на наличие различных веб-фреймворков для построения приложений, мы столкнулись с тем, что все они по-разному ведут себя на различных девайсах.
4. Интеграция сторонних сервисов: Вы не можете разрабатывать гибридное приложение на одном только JavaScript. Вам потребуется интегрировать такие фреймворки для гибридной разработки, как Cordove, Ionic или React Native, каждый из которых требует определенных усилий для освоения.

Появление ряда багов, которые могут приводить к определенным нарушениям нормальной работы приложения. Во время тестирования, исходя из различного поведения приложения на разных устройствах, правка одного бага влечет за собой появление других под новыми версиями операционных систем.

Ограниченное представление визуальных и графических элементов, в частности, анимации.

Таким образом, использовать гибридный подход можно в тех случаях, когда:

1. Стоит задача быстрой разработки приложения под более, чем 2 мобильные платформы;
2. Есть необходимость сэкономить на бюджете;
3. Требуется относительно простое приложение без сложной анимации;
4. Можно исключить поддержку множества параметров нативного функционала.
   1. Примеры

Гибридная разработка пользуется популярностью у крупных мировых компаний. Множество широко известных приложений являются кросс-платформенными. Среди них: Wikipedia, Health tap, BBC Sport App и Evernote. Пользователи скачали каждое из этих приложений более миллиона раз, что говорит о высоком качестве разработанных решений.

Приложение HeartCamera для iOS, позволяющее украсить фотографию рисованными сердцами и т.п.

1. Загружается из магазина;
2. Использует камеру телефона;
3. Необходимо подключение к интернету при желании поделиться результатом своей работы;
4. Можно настроить push-уведомления.

«Heart Camera» — приложение, которое адаптировано для IOS. Его применяют с целью украсить и «оживить» фотоснимки. Здесь есть много фильтров, наклеек и других дополнений, которые пользуются спросом у любителей мобильной обработки. Загрузить программу можно быстро, используя магазин. Для полноценной работы применяется камера мобильного телефона, а в качестве дополнения доступна настройка push-уведомлений. Если есть желание поделиться результатами обработки в социальных сетях или мессенджерах, то можно подключить Интернет для рассылки изображения (остальной функционал работает в офлайн режиме).

Приложение TripCase — органайзер для планирования путешествий.

1. Загружается из магазина;
2. Может использовать геолокацию;
3. Необходимо подключение к интернету;
4. Может использовать сотовую сеть;
5. Можно настроить push-уведомления

«TripCase» — это уникальное гибридные приложение для мобильных устройств, которое предоставляет доступ к планированию своих путешествий. Доступно множество функций для быстрого сбора вещей и даже заказа билетов. Загружается стандартно, используя markets. Для работы нужно проверить стабильное подключение к Интернету. При желании настраиваются push-уведомления. Также стоит выделить важность доступа к таким функциям телефона, как сотовая связь и геолокация.

# ФРЕЙМВОРКИ JAVASCRIPT

* 1. Общая характеристика фреймворков JavaScript

JS-framework’и — это набор методов и функций для построения динамических веб/мобильных/настольных приложений на языке Javascript. Фреймворки созданы для того, чтобы использовать готовые решения на знакомом языке JavaScript. Часто на фреймворках разрабатываются Single Page Applications, все действия на сайте проиходят на одной странице, без прямого перехода с нее.

Из преимуществ фреймворков необходимо выделить то, что данные функции упрощают взаимодействие веб-приложения с сервером и ускоряют манипулирование элементами на веб-странице или в приложении. Они позволяют осуществить мгновенную обратную связь с пользователями в реальном времени. С помощью фреймворков можно создавать как целые сайты, так и отдельные модули. На веб-сайтах без фреймворков все данные и код хранятся на сервере, с каждым изменением происходит загрузка с сервера всей страницы, а при использовании фреймворков перезагружаются только части страницы, где изменился контент. Недостатками можно назвать только временно неполную поддержку поисковыми системами.

Существует достаточно много приложений и, судя по статитсике, их число будет только возрастать.

* 1. Фреймворк VUE.js

Vue - это современный фреймворк для разработки пользовательских Web интерфейсов с использованием паттерна MVVM (Model-View-ViewModel). Его придумал и воплотил в реальность Иван Йу. Ядро фреймворка хорошо справляется с задачами уровня представления (view), что облегчает интеграцию с различными библиотеками и проектами, в число которых входит Vuetify. На Vue чаще всего создаются сложные одностраничные приложения SPA. Основными конкурентами Vue являются React и Angular, которые в совокупности заполняют 100% рынка Web-разработки. Продукты, написанные на этом фреймворке быстро подгружаются на страницу и не нагружают проект за счет небольшого веса библиотеки. Его широко используют такие компании, как Alibaba, Baidu, Xiaomi, GitLab.

Vue хорошо реализует как маленькие проекты, нуждающиеся в реактивности, представлении формы с помощью AJAX запросов, и всего, что касается красивого и плавного отображения на странице, так и большие SPA за счет компонентов, Router и Vuex. На Vue можно использовать как общедоступные API, так и внешние API для обработки данных.

Из наиболее знаменитых продуктов на Vue.js можно отметить следующие:

Chess.com — один из самых посещаемых веб-сайтов, где 19 миллионов пользователей играют друг с другом в режиме реального времени либо в одиночной игре со специально обученными интеллектуальными системами. Кроме игры на сайте есть новости, блог, сообщества, уроки, головоломки.

Codeship — это платформа для облачного хранения веб-приложенийЮ которые используют крупные компании, такие как Red Bull, CNN и Product Hunt. Именно благодаря фреймворку Vue платформе удалось устранить множество багов и ускорить процесс обработки информации.

Livestorm — это веб-приложение для создания вебинаров. Ее используют компании Workable, Pipedrive и Instapage. .Livestorm создавался полностью на Vue JS.

Vue в первую очередь решает задачи уровня представления (view), что упрощает интеграцию с другими библиотеками и существующими проектами.

Vue вобрал в себя лучшие черты Angular и React, скорость, легковесность, возможность поддержки технологий TypeScript и JSX и вместе с тем сохранил стандарты написания кода на HTML и CSS, что облегчает процесс создания и поддержки проекта.

* 1. Фреймворк Angular

Angular — это кроссплатформенный фреймворк (не библиотека), который сделан с использованием MVC-паттерна проектирования, но практически не задействует связь между представлением, данными и логикой компонентов. Angular делает упор на обработку данных на стороне клиента, вместо сервера, как это делают другие фреймворки, чем уменьшает нагрузка на сервер и веб-приложение становится легче. Он написан на TypeScript, что позволяет допускать меньше ошибок при разработке.

Преимущества фреймворка Angular:

1. обширная документация,
2. поддержка компанией Google,
3. обширный комплект инструментов для разработки – Material Design UI, CLI

Из недостатков можно выделить:

1. Наличие TypeScript повышает порог вхождения при изучении фреймворка, требует наличия более квалифицированных и опытных разработчиков;
2. Часты релизы новых версий;
3. Сложная интеграция с другими библиотеками, что усложняет поддержу проектов, написанных на нем продуктов;
4. Большой объем;
5. Отсутствие DOM – напрямую обращается к редставлениям;
6. Меньшая производительность по отношению к другим фреймворкам.

Angular не является библиотекой, как ошибочно можно предположить, так как обладает обширным функционалом и множеством готовых решений. Среди пользователей данного фреймворка отличаются такие компании, как Skyeng, Iponweb, IsSoft, Positive Technology, MTS, Grid Dynamics, Sitecore, Netcraker, Google.

* 1. Vuetify - фреймворк для создания гибридных мобильных приложений

Vuetify является одним из самых важных фреймворков для визуализации контента, написанного при помощи Vue. Он позволяет внедрить в проект любые элементы разных форм и назначений, предложить пользователям все, что нужно для разработки многофункциональных, красивых и интересных веб-приложений, используя Material Design. Это реализуется за счет регулярного цикла обновлений, длительной поддержке предыдущих версий, отзывчивому участию сообщества, обширной экосистеме ресурсов и приверженности качественным компонентам

Библиотека Vuetifyjs получила более 7 тысяч звёзд на GitHub. Она даёт разработчикам возможности по созданию пользовательских интерфейсов с использованием принципов Google Material Design. В её релизе V 1.0 Alpha можно найти более 80 компонентов, подходящих для повторного использования и спроектированных с применением семантических принципов, основанных на простых и чётких именах свойств, которые легко запоминать и использовать.

* 1. Фреймворк React

React — это JavaScript библиотека с открытым исходным кодом для разработки вб-приложений. Впервые React был использован в 2011 году в ленте Facebook, затем в 2012 в ленте Instagram.

И по сей день React разрабатывается корпорацией Facebook. React может использоваться Как для браузерных веб приложений, так и для мобильных. Цель React — предоставить высокую скорость, легкость разработки, а так же производительность и работоспособность приложений на разных платформах.

Изначально React это библиотека, но с добавлением в него сторонних модулей она становится полноценным фреймворком

Сильные стороны React:

1. Приложения на React’е получаются легкими;
2. Кроссплатформенность
3. Огромное сообщество;
4. Скорость работы.

Благодаря React во фронтенд-разработке JavaScript появилась новая архитектура — Flux. Flux — это термин, придуманный для обозначения однонаправленного потока данных с очень специфичными событиями и слушателями.

JSX — это расширение JavaScript, позволяющее декларативно создавать компоненты пользовательского интерфейса. JSX обладает следующими характеристиками и возможностями:

1. Применение простой декларативной разметки;
2. Код разметки расположен там же, где и код компонента.

Если, до появления JSX, нужно было декларативно описывать интерфейсы, то нельзя было обойтись без использования HTML-шаблонов.

. В общем-то, тому, кто описывает интерфейсы средствами JSX, доступны все возможности JavaScript. Полагаю, что в этом заключается огромное преимущество React перед другими фреймворками и библиотеками.

Возможности React, во время работы со стилями:

CSS-файлы, которые можно загружать в заголовочной части страницы могут использоваться для настройки структуры страниц, шрифтов, изображений и прочих подобных элементов. Это — надёжный и работоспособный механизм стилизации.

CSS-модули — это CSS-файлы область применения которых ограничена локальной областью видимости. Их можно имплементироватьс прямо в JavaScript-файлы. Для того чтобы применять CSS-модули, необходимо пользоваться правильно настроенным загрузчиком модулей.

Пакет styled-jsx, который дает возможность объявлять стили прямо в html-коде React-компонентов. Это напоминает использование тега <style> в HTML. Область видимости таких стилей можно назвать «гиперлокальной». Cтили воздействуют только на элементы, к которым они применяются, и на связанный с ними элементы.

* 1. Сравнение фреймворков

Из-за развития веб-разработки и увеличенной востребованности спроса на удобный и отзывчивый дизайн были созданы фреймворки облегчающие и ускоряющие разработку веб-приложений в разы быстрее, чем разработка велась бы на JavaScript. Каждый JavaScript фреймворк по своему уникален и востребован в своей отрасли. Каждый из фреймворков обладает как положительными качествами, так и негативными. Некоторые из фреймворков подойдут для не больших веб приложений, но совершенно не будут подходить для больших нагруженных и на оборот.

На 2021 год в фронт-енд разработке существует огромное множество фреймворков, но по настоящему востребованными, функциональными и надежными являются 3: Vue, React и Angular2+.

Vue – на настоящее время это молодой JavaScript фреймворк с крайне растущей аудиторией. Самый простой для изучения сравнивая с React и Angular. Работу на данном фреймворке можно начать достаточно быстро не затрачивая много времени на подготовку окружения, при этом достаточно мощный для профессиональных разработчиков. Vue не обладает таким количеством встроенных функций, как Angular2+, но их несколько больше, чем у React. Фреймворк Vue это крайне популярный выбор среди начинающих веб-разработчиков. Так же для Vue написана подробная документация как на английском, так и на русском языке.

Особенности Vue:

1. привязка данных;
2. производительный (есть Virtual DOM);
3. Легкость в изучении;
4. Адаптивность
5. Подробная документация;
6. масштабирование;
7. оптимизация HTML-блоков;
8. миниатюрный размер;
9. шаблоны.

На данный момент фреймворк Vue JS используют такие корпорации и компании как: Adobe, Behance, Reuters, Xiaomi, Gitlab, WizzAir, EuroNews, Grammarly, Codeship и Alibaba.

Недостатки Vue JS:

1. Мало обучающих материалов. Хорошая документация – это неоспоримое преимущество, но для некоторых случаев было бы важно иметь больше различных курсов, примеров и готовых решений.
2. Интеграция в проекты может быть крайне затруднительной.
3. Снижение производительности за счет долгой подгрузки кода из фреймворка;
4. Не подходит для маленьких проектов, так как усложняет сам проект за счет большого количества кода, и связей между фаилами, которые невозможно удалить для полноценной работы фреймворка, тем самым проект весит намного больше, чем мог бы без фреймворков.

React – крайне популярный фреймворк разработанный корпорацией Facebook. React позволяет легко создавать крайне функциональные компоненты а также продвигает идею их использования для написания красивого и понятного кода API. Фреймворк крайне популярен, особенно в разных начинающих компаниях и стартапах. Имея большой выбор легкодоступных плагинов с открытым исходным кодом и расширений, можно создать практически любой тип веб-приложения. Шаблоны компонентов создаются с использованием JSX, этот подход немного отличается от аналогов в других фреймворках.

Особенности React:

1. компонентно-ориентированный;
2. декларативный;
3. производительный (благодаря React Virtual DOM);
4. серверный рендеринг;
5. наличие Redux;
6. поддержка PWA;
7. JSX.

Примеры корпораций и компаний, использующих React: Facebook, Instagram, Atlasian, Codepen, Pinterest, Asana, Сбербанк, Revolut, Авито, Discord.

Недостатки React:

1. Использование шаблонов вместе с JSX для начинающих разработчиков может показаться не понятным и странным;
2. Не прямолинейность выбора этапов развития проектов. Для разработки на React нужен опытный руководитель.

Angular - свободная платформа для создания веб-приложений, написанная на TypeScript. Разрабатываемая командой из компании Google. Фреймворк Angular предоставляет огромное количество встроенного функционала, на пример очень удобный CLI, гораздо более удобный чем у React и Vue.

Особенности фреймворка Angular:

1. универсальность;
2. нативность;
3. Высокая сложность в виду крайне обширного функционала;
4. используется с Typescript;
5. интеллектуальное автозаполнение HTML-компонента;
6. архитектура рассчитана на крупные проекты;
7. MVVM-модель.

Фреймворк используют такие корпорации и компании как: Skyeng, Iponweb, IsSoft, Positive Technology, MTS, Grid Dynamics, Sitecore, Netcraker, Google.

Из недостатков фреймворка Angular можно выделить слабую производительность, которую можно решить технологией ChangeDetectionStrategy и наличие огромного колличества разнообразных структур по началу обучения воспринимается не легко и начинающим специалистам нужно понимать что у фреймворка Angular крайне высокий порог входа.

* 1. Почему я выбрала вью

Vue - это современный фреймворк для разработки пользовательских Web интерфейсов с использованием паттерна MVVM (Model-View-ViewModel). Его придумал и воплотил в реальность Иван Йу. Ядро фреймворка хорошо справляется с задачами уровня представления (view), что облегчает интеграцию с различными библиотеками и проектами, в число которых входит Vuetify. На Vue чаще всего создаются сложные одностраничные приложения SPA. Основными конкурентами Vue являются React и Angular, которые в совокупности заполняют 100% рынка Web-разработки. Продукты, написанные на этом фреймворке быстро подгружаются на страницу и не нагружают проект за счет небольшого веса библиотеки. Его широко используют такие компании, как Alibaba, Baidu, Xiaomi, GitLab.

Vue хорошо реализует как маленькие проекты, нуждающиеся в реактивности, представлении формы с помощью AJAX запросов, и всего, что касается красивого и плавного отображения на странице, так и большие SPA за счет компонентов, Router и Vuex. На Vue можно использовать как общедоступные API, так и внешние API для обработки данных.

Функции Vue.js:

1. Реактивные интерфейсы;
2. Декларативный рендеринг;
3. Связывание данных;
4. Директивы (все директивы имеют префикс «V-». В директиву передается значение состояния, а в качестве аргументов используются html атрибуты или Vue JS события);
5. Логика шаблонов;
6. Компоненты;
7. Обработка событий;
8. Свойства;
9. Переходы и анимация CSS;
10. Фильтры.

С помощью Vue также можно создавать frontend блога на популярных CMS. Vue.js отлично подходит и для разработки динамических интерфейсов, которые адаптируются под пользователя.

Фреймворк Vue.js несмотря на сходства с другими фреймворками JavaScript уникален и всеже обладает своими особенностями, отличающими его от Angular и React. Одной из неотьемлемых частей фреймворка являются компоненты. При этом необходимо отметить, что React и Vue хорошо подходят для обработки так называемых немых компонентов — небольших, не имеющих состояния функций, которые получают входные данные и возвращают элементы в качестве вывода.

Vue — это прогрессивный фреймворк для создания пользовательских интерфейсов. Vue создан пригодным для постепенного внедрения, в отличие от Angular или React. Это значит, что внедрять этот фреймворк можно поэтапно начиная с определенных страниц, что значительно упрощает разработку.

* 1. Платформа для разработки гибридных мобильных приложений – Apache Cordova

Apache Cordova — это платформа разработки мобильных приложений с открытым исходным кодом. Она позволяет использовать стандартные веб-технологии, такие как HTML5, CSS3 и JavaScript для кросс платформенной разработки, избегая родного языка разработки для каждой из мобильных платформ. Приложения выполняются внутри обертки нацеленной на каждую платформу и полагаются на стандартные API для доступа к датчикам устройства, данным и состоянию сети.

Apache Cordova закончил инкубационный период в октябре 2012 года как основной проект в рамках Apache Software Foundation (ASF). Через участие в ASF, будущее развитие Cordova, обеспечит открытое руководство проектом. Он всегда будет оставаться бесплатным и с открытым исходным кодом под лицензией Apache версии 2.0. Посетите cordova.apache.org для получения дополнительной информации.

Использование Apache Cordova, универсально и подойдет для таких типов пользователей платформы, как:

Мобильный разработчик и хотите расширить приложение на более чем одну платформу, без необходимости повторно реализовать его для каждого языка разработки платформ и набора инструментов.

Веб-разработчик и хотите, чтобы развернуть веб-приложение, который упаковано для распространения в различных магазинах приложений.

Мобильный разработчик, заинтересованный в смешивания компонентов собственного приложения с WebView (Специальный браузер окно), можно получить доступ к API уровне устройства, или если вы хотите разработать плагин интерфейс между родной и WebView компонентами.

К основным компонентам Cordova относятся следующие. Пиложения Cordova полагаются на общий файл config.xml, который содержит информацию о приложении и определяет параметры, влияющие на то как приложенеи работает, такие как, реагирует ли оно на изменение ориентации устройства. Этот файл соответствует спецификации W3C Упакованные веб-приложения, или widget, .

Само приложение реализована как веб-страницы, по умолчанию локальный файл под названием index.html, который ссылается на любой CSS, JavaScript, изображения, файлы мультимедиа или другие ресурсы необходимы для его запуска. Приложение выполняет как WebView в пределах оболочки приложения, которую вы распространяете в магазины приложений.

WebView с поддержкой Cordova может представлять приложения и полностью его пользовательский интерфейс. На некоторых платформах она также может быть компонентом в больших, гибридные приложения, который объединяют WebView с другими компонентами приложения. (Подробную информацию см. в разделе "Интеграция WebViews".)

Примечание: начиная с версии 3.0, при создании проекта Cordova, она не имеет каких-либо плагинов настоящей. Это новое поведение по умолчанию. Какие плагины вы хотите, даже основные плагины, должен быть явно добавлен.

Cordova не предоставляет каких-либо виджетов пользовательского интерфейса или MV фреймворков. Cordova предоставляет только во время выполнения, в которой те могут выполнять. Если вы хотите использовать UI-виджеты и/или MV фреймворк, вам нужно будет выбрать их и включить их в приложение самостоятельно, как ресурсы третьей стороны.

Начиная с версии 3.0 можно использовать два основных рабочих процесса для создания мобильных приложений. Хотя вы можете использовать любой рабочий процесс для выполнения одной и той же задачи, каждый из этих путей имеет свои преимущества:

Кросс платформенный рабочий процесс: Используйте этот рабочий процесс если вы хотите, чтобы ваше приложение запускалось на максимально возможном количестве мобильных платформ, с минимальными нуждами для платформо-специфической разработки. Этот рабочий процесс формируется возле утилиты cordova, также известном как Cordova CLI, который был введен начиная с Cordova 3.0. CLI это высоко уровневый инструмент который позволяет построить проекты для как можно большего количества платформа одновременно, абстрагируя как можно больше функциональности низко-уровневых скриптов. CLI копирует общий набор web ресурсво в подкаталоги для каждой мобильной платформы, делает любые необходимые изменения в конфигурацию для каждой платформы, запускает сценарии сборки для создания исполняемых файлов приложения.

Платформо-ориентированный процесс разработки: Используйте этот процесс если вы хотите сфокусироваться на построении приложения для одной платформы и вам будет необходимо вносить модификации на низком уровне. Вы должны использовать этот подход, к примеру, если вы хотите чтобы ваше приложение совмещало собственные компоненты с веб-ориентированными компонентами Cordova, как описано в разделе "Интеграция WebViews". Как правило этот рабочий процесс используется, если необходимо изменить проект с помощью SDK.

Примечание: после того, как вы перейти от рабочего процесса на основе CLI к платформо-ориентированному процессу и инструментам командной строки, вы не сможете вернуться назад. CLI поддерживает общий набор кросс платформенный исходного кода, который при каждом построении он использует чтобы изменить платформо-зависимый исходный код. Чтобы сохранить любые изменения, внесенные вами в платформо-зависимые ресурсы, вам нужно перейти к платформо-ориентированным инструментам, которые игнорируют кросс платформенный исходный код и вместо этого полагаются на платформо-зависимый исходный код.

* 1. Google сервисы как неотъемлемая часть гибридного мобильного приложения

Разработка серверной части - один из самых трудных этапов создания приложения. Зачастую, при планировании разработки проекта недооценивается необходимый объем ресурсов и время создания бэкенда (Рис.1.1.). Еще одна проблема - ограниченность доступных команде ресурсов. Чаще всего разрабатывать бэкенд приходится с помощью тех инструментов и технологий, которыми владеют члены команды. Процесс получается длительным, а само приложение — достаточно сложным и дорогим с точки зрения сопровождения. А длительная разработка бэкенда на даже относительно простых проектах ведет к увеличению расходов и другим рискам: расход денег без визуально видимого результата.

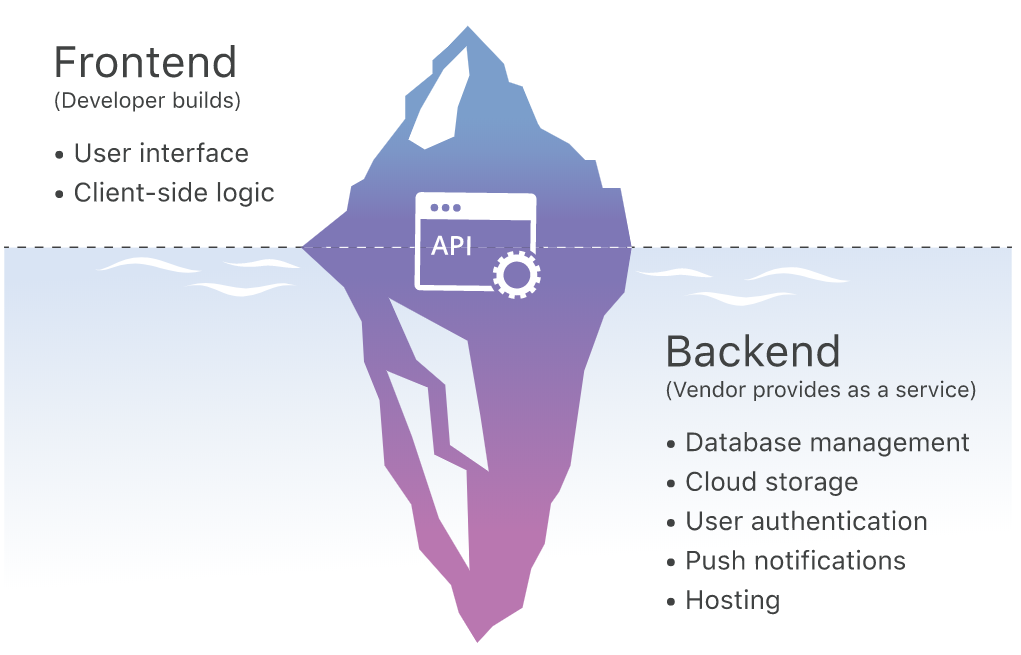


Рис.2.8. Принцип работы современного приложения.

Решить эту проблему возможно с помощью **Backend as a Service (BaaS**) - готовых серверных сервисов, собрав которые вместе, получается необходимый универсальный бэкенд для любого проекта. BaaS является хорошим решением, избавляющем от существенных затрат, а именно написания и поддержки серверного бэкенда. BaaS дает возможность реализовать простой сервис силами одного мобильного разработчика [1]. Такое решение будет особо уместно для небольших стартапов, которые сначала хотят проверить бизнес идею на её работоспособность, не планируя очень сложной логики и взаимодействия со сторонними сервисами — хотя с BaaS такое тоже возможно реализовать.

Преимущества бэкенда на BaaS, в основном, заключаются в том что, используя BaaS, разработчики способны быстро построить необходимый бэкенд и платформу для обработки данных, поступающих из мобильных приложений.

BaaS позволяет избегать разработчикам необходимости иметь дело с:

* физическим сервером приложения;
* базой данных;
* клиент-серверной библиотекой;
* написанием админки;
* дизайном своего API;
* хостингом.

BaaS-функции включают в себя облачное хранилище, push-уведомления, управление пользователями и файлами, службы определения местоположения и др [2]. Все эти сервисы имеют собственный API, чтобы легко встраиваться в приложения.

Один из многофункциональных сервисов гугл – Firebase, помогает быстро создавать качественные приложения, увеличивать аудиторию вовлеченных пользователей и повышать доходы. Платформа содержит множество полезных функций для приложения, в том числе серверный код для мобильных сервисов, статистику, а также инструменты для монетизации и расширения аудитории.

Большинство приложений должны идентифицировать личность пользователя. Знание личности пользователя позволяет приложению безопасно сохранять пользовательские данные в облаке и обеспечивать одинаковую персонализированную работу на всех устройствах пользователя.

Аутентификация Firebase предоставляет бэкэнд-сервисы, простые в использовании SDK и готовые библиотеки пользовательского интерфейса для аутентификации пользователей в приложении. Он поддерживает аутентификацию с использованием паролей, телефонных номеров, популярных провайдеров федеративной идентификации, таких как Google, Facebook и Twitter, и многого другого. Аутентификация Firebase тесно интегрируется с другими сервисами Firebase и использует отраслевые стандарты, такие как OAuth 2.0 и OpenID Connect, поэтому ее можно легко интегрировать с пользовательским бэкэндом.

В приложение с использованием Firebase возможно войти либо с помощью FirebaseUI в качестве полного решения для проверки подлинности, либо с помощью пакета Firebase Authentication SDK для ручной интеграции одного или нескольких методов входа в приложение (Рис.1.2.).

Firebase предоставляет следующие возможности по аутентификации:

* **Аутентификация по электронной почте и паролю;**
* **Интеграция с федеративным поставщиком удостоверений;**
* **Проверка подлинности номера телефона;**
* **Кастомная аутентификация системной интеграции;**
* **Анонимная авторизация.**

Также стоит отметить, что по умолчанию аутентифицированные пользователи могут читать и записывать данные в базу данных Firebase Realtime и облачное хранилище [3].

Еще одним ярким представителем google сервисов является Firebase Hosting. Это хостинг для веб-сайтов промышленного уровня для разработчиков. С помощью одной команды появляется возможность быстро развернуть веб-приложения и предоставлять как статический, так и динамический контент в глобальную CDN (сеть доставки контента). Также можно соединить Firebase Hosting с Cloud Functions или Cloud Run для создания и размещения микросервисов в Firebase.

Firebase Hosting создан для современного веб-разработчика. Веб-сайты и приложения стали более мощными, чем когда-либо, благодаря появлению интерфейсных JavaScript-фреймворков, таких как Angular и статических генераторов, таких как Jekyll. Он предоставляет инфраструктуру, функции и инструменты, адаптированные для развертывания веб-сайтов, приложений и управления ими не зависимо от того, простая ли это целевая страница приложения либо сложное Progressive Web App (PWA).

Ключевые возможности:

* Передача контента через безопасное соединение;
* Статический и динамический контент + микросервисы;
* Быстрая доставка контента;
* Развертывание новых версий одной командой;
* Откат одним кликом.

Firebase Hosting обеспечивает быстрый и безопасный хостинг для веб-приложения, статического и динамического контента, микросервисов.

Не менее важным сервисом является облачное хранилище для Firebase. Это мощный, простой и экономически эффективный сервис хранения объектов, созданный для масштаба Google. Пакеты Firebase SDK для облачного хранилища повышают безопасность Google при загрузке и выгрузке файлов для приложений на основе Firebase, независимо от качества сети [4]. Помимо этого, доступны такие инструменты, как SDK для хранения изображений, аудио, видео или другого пользовательского контента. На сервере существует возможность использовать Google Cloud Storage , чтобы получить доступ к тем же файлам.

Ключевые возможности:

* Надежные операции;
* Высокий уровень безопасности;
* Высокая масштабируемость.

Разработчики используют Firebase SDK для облачного хранилища, чтобы загружать и скачивать файлы непосредственно с клиентов [5]. Если сетевое соединение плохое, клиент может повторить операцию прямо с того места, где он был прерван, что экономит время пользователей и пропускную способность.

Без базы данных Firebase Realtime сложно представить себе хоть одно приложение, работающее с данными пользователей. О представляет собой облачную базу данных [6]. Данные хранятся в формате JSON и синхронизируются в режиме реального времени с каждым подключенным клиентом. При создании кроссплатформенного приложения с помощью набора средств разработки SDK для iOS, Android и JavaScript, все клиенты совместно используют один экземпляр базы данных в реальном времени и автоматически получают обновления с самыми новыми данными.

База данных Firebase Realtime позволяет создавать многофункциональные приложения для совместной работы, обеспечивая безопасный доступ к базе данных непосредственно из кода на стороне клиента (Рис.1.3.).



Рис.1.3. Совместная работа устройств в реальном времени*.*

Данные сохраняются локально, и даже в автономном режиме события в реальном времени продолжают срабатывать, предоставляя конечному пользователю доступ к приложению. Когда устройство восстанавливает соединение, база данных реального времени синхронизирует локальные изменения данных с удаленными обновлениями, которые произошли, когда клиент находился в автономном режиме, автоматически объединяя любые несоответствия.

Ключевые возможности:

* Режим реального времени;
* Доступность в оффайн режиме;
* Доступно с клиентских устройств;
* Масштабирование по нескольким базам данных.

База данных реального времени является базой данных NoSQL и, как таковая, имеет различные оптимизации и функциональность по сравнению с реляционной базой данных.

Google аутентификация, Google OAuth, представляет собой фреймворк для авторизации, позволяющий приложениям осуществлять ограниченный доступ к пользовательским аккаунтам на HTTP сервисах, например, на Facebook, GitHub и DigitalOcean. Он работает по принципу делегирования аутентификации пользователя сервису, на котором находится аккаунт пользователя, позволяя стороннему приложению получать доступ к аккаунту пользователя. OAuth 2 работает на web, desktop и мобильных приложениях [7].

OAuth выделяет четыре роли:

* Владелец ресурса;
* Клиент;
* Сервер ресурсов;
* Авторизационный сервер.

Владельцем ресурса является пользователь, который авторизует приложение для доступа к своему аккаунту. Доступ приложения к пользовательскому аккаунту ограничен “областью видимости” (scope) предоставленных прав авторизации (например, доступ на чтение или запись).

Сервер ресурсов непосредственно хранит защищённые данные аккаунтов пользователей, а авторизационный сервер проверяет подлинность информации, предоставленной пользователем, а затем создаёт авторизационные токены для приложения, с помощью которых приложение будет осуществлять доступ к пользовательским данным.

С точки зрения разработчика приложения API сервиса одновременно выполняет и роль сервера ресурсов и роль сервера авторизации. Далее мы будем считать эти две роли одной, и называть её Сервис или API.

Клиентом является приложение, которое хочет осуществить доступ к аккаунту пользователя. Перед осуществлением доступа приложение должно быть авторизовано пользователем, а авторизация должна быть одобрена со стороны API.

В данной диаграмме (Рис.1.4.) представлены роли, используемые в OAuth, и их взаимодействия друг с другом.



Рис.1.4. Диаграмма Взаимодействия ролей в OAuth.

Последовательность шагов на этой диаграмме:

1. Приложение запрашивает у пользователя авторизацию на доступ к серверу ресурсов;
2. Если пользователь авторизует запрос, приложение получает разрешение на авторизацию (authorization grant);
3. Приложение запрашивает авторизационный токен у сервера авторизации (API) путём предоставления информации о самом себе и разрешении на авторизацию от пользователя;
4. Если подлинность приложения подтверждена и разрешение на авторизацию действительно, сервер авторизации (API) создаёт токен доступа для приложения. Процесс авторизации завершён;
5. Приложение запрашивает ресурс у сервера ресурсов (API), предоставляя при этом токен доступа для аутентификации;
6. Если токен действителен, сервер ресурсов (API) предоставляет запрашиваемый ресурс приложению.

Фактический порядок шагов описанного процесса может отличаться в зависимости от используемого типа разрешения на авторизацию, но в целом процесс будет выглядеть описанным образом.

Перед применением OAuth в приложении, необходимо зарегистрировать его на сервисе. Это делается путём регистрации в разделе “developer” или “API” сайта сервиса, где предоставляется следующая информация:

* Название приложения;
* Сайт приложения;
* Redirect URL или callback URL.

Redirect URL - это URL, на который сервис будет перенаправлять пользователя после авторизации (или отказа в авторизации) вашего приложения. После регистрации приложения сервис создаст учётные данные клиента - идентификатор клиента (client ID) и секрет клиента (client secret). Идентификатор клиента представляет собой публично доступную строку, которая используется API сервиса для идентификации приложения, а также используется для создания авторизационных URL для пользователей. Секрет клиента используется для аутентификации подлинности приложения для API сервиса, когда приложение запрашивает доступ к аккаунту пользователя. Секрет клиента должен быть известен только приложению и API.

Одним из лучших решений для поиска информации на карте с отметками достопримечательностей, организаций и многих других полезных функций является сервис Google Maps[8]. Это комплекс приложений, созданных на базе бесплатного сервиса картографии и технологии, используемой Google.

Среди возможностей отображения контента сервиса Google Maps отмечаются следующие:

1. «Спутник» – показ спутниковых и аэрофотографий (сюда относятся «Имена объектов» – отображение сведений на схемах. До этого режим назывался «Гибрид»);
2. «Карта» – показ данных картографии;
3. «Ландшафт» – схемы, отображающие массивные элементы поверхности, например леса или холмы (ложный 3D-рельеф);
4. «Просмотр улиц» – режим, позволяющий ориентироваться по панорамным фотоснимкам на фоне городских улиц;
5. «Личные схемы» – персональный режим, благодаря которому можно нанести на карту требуемое содержимое.

Благодаря Google Maps пользователь может:

* просматривать интересные локации на карте;
* открывать файлы с расширением KML и KMZ, а также сохранять нужные метки в этих форматах;
* прокладывать маршрут;
* вносить персональный текст, картинки и видеоролики;
* отправлять созданную схему друзьям.

Среди режимов и функции Google Maps отмечаются следующие:

Режим Google Transit дает подробные рекомендации по проезду между двумя указанными точками на доступном общественном транспорте и приблизительно подсчитывает, сколько это займет времени. Данная опция доступна для всей России.

Пользователь может отправить на почтовый ящик ссылку со своим местоположением либо разместить HTML-код карты Google на веб-ресурсе. Также в Google Maps имеется инструмент генерирования карты для последующей распечатки.

Встроены результаты поиска информации по организациям – поиск адресов компаний и их контактных данных в выбранном районе. К примеру, если написать запрос «суши в районе Тверская улица», на локальной системе координат обозначатся точки расположения ближайших заведений, их адреса, режимы работы. Также можно посмотреть дополнительные сведения, такие как формы оплаты, рейтинг, отзывы клиентов и др.

В картах доступна функция Map Maker (ранее это был отдельный сервис), позволяющая изменить содержание собственной информацией.

Сервис Google Maps полностью бесплатный. Но при этом существует ограничение: в локальном поиске отображается только 7 компаний, предоставляющих схожие товары/услуги.

Факторы ранжирования в «Картах Google»:

* полнота информации о компании, представленной при регистрации в «Google Мой Бизнес»;
* упоминание поисковых запросов и категории в имени компании;
* наличие веб-сайта, на котором прописан фактический адрес в тегах <adress> </adress>;
* положительные отзывы;
* интересная информация в заметках;
* подключение контекстной рекламы AdWords;
* присутствие на Google Earth.

Польза google сервисов в гибридном мобильном приложении очень весомая, так как они позволяют легко и быстро добавить полезный функционал в любое место в приложении. Так, например, авторизация с помощью данного гугл сервиса дает большое количество преимуществ разработчикам, так же это простой способ аналога регистрации для пользователя, ведь им всего лишь нужно аутентифицироваться с помощью данных своего гугл аккаунта для первого и последующих входов в приложение, нет необходимости в заполнении какими-либо данными профиля, ведь все необходимое предоставляет гугл. Что касается разработчиков, такой подход авторизации снимает с них ответственность за хранимые данные в своей базе данных, таких как пароль или других данных [9]. Так же этот способ экономит много времени, которое могло быть потрачено на создание, построении защиты данных, которые должны храниться у разработчика, а также, устранять возможности обхода, обмана системы, кражи данных и так далее.

# 3. ОПИСАНИЕ Демонстрационного ПРИЛОЖЕНИЯ

**3.1. Описание структуры приложения**

В ходе данной работы были рассмотрены преимущества и недостатки гибридных web технологий, были сравнены разные JavaScript фреймворки, из которых, на основе сравнения, был выбран самый подхоящий для разработки гибридного мобильного приложения – Vue.js, были изучены основные элементы фреймворка, а также, как работают технологии, имплементированные в данное демонстрационное приложение, такие как Firebase, Google OAuth, Cordova, Vue, автономная работа приложений и другие. Приложение имеет следующие функциональные возможности:

1. Авторизация пользователя - имея логин и пароль, пользователи, зарегистрированные в системе, имеют возможность войти в систему;
2. Просмотр постов других пользователей;
3. Возможность добавления нового поста;
4. Карты для отображения местоположения пользователя.

Регистрация пользователя в системе – это первый шаг, который должны сделать пользователи. Регистрация в данном приложении осуществляется путем добавления емаила любого почтового клиента, где предусматривается наличие существующего аккаунта и его авторизация на устройстве. Чтобы зарегистрироваться, необходимо выполнить следуюшие шаги:

1. Ввести емаил(например [examplemail@yahoo.com](mailto:examplemail@yahoo.com));
2. Ввести свои имя и фамилию;
3. Ввести желаемый пароль либо воспользоваться функцией генерации пароля;
4. Нажать на кнопку “Save”.

Если логин и / или пароль были введены неправильно, пользователю показываются соответствующие сообщения, так что он понимает, какие данные были введены неправильно. Авторизация пользователей осуществляется еще проще, чем в шагах, описанных выше. В случае регистрации через email, пользователь войдет в систему сразу после регистрации. В случае регистрации через google почту, происходит редирект на страницу <https://accounts.google.com/> , где необходимо выбрать учебную запись, через которую пользователь желает авторизоваться либо добавить новую учетную запись и сделать те же шаги.

После авторизации, пользователь перенаправляется на свою главную страницу. Он может содержать различную информацию, такую как:

1. На главной странице - пост, в котором отображаются данные об авторе, текст, описывающий содержание поста, цена и картинка;
2. В сайдбаре - имя пользователя, изображение профиля, соответствующее изображению основной учетной записи(в данном примере это google или yahoo) почты.
3. В сайдбаре – карты, отображающие текущее местоположение пользователя, а так же данные геолокации(широту и долготу), в которй он находится. Так же при закрытии карт приложение запоминает текущее состояние карты и открывает его в том же месте, в которо пользователь просматривал их до закрытия.

Пользователь, в зависимости от уровня доступа, имеет возможность просматривать записи других пользователей и добавлять собственные записи.

* 1. Черты гибридных технологий в новостном приложении

Данное гибридное приложение развертывается в собственном контейнере, который использует мобильный объект WebView. При использовании, данный объект отображает адаптированный веб-контент с помощью веб-технологий (CSS, JavaScript, HTML, HTML5). Веб-контент может отображаться либо сразу после открытия приложения, либо асинхронно, только для определенных частей приложения [10]. Данное гибридно приложение также полагается на платформы, которые предлагают API-интерфейсы JavaScript, если эти функции вызываются в WebView. По итогу, смысл гибридной разработки заключается в том, что разработка происходит на WebView мобильного телефона с помощью HTML5/Javascript/СSS, затем добавляются нативные элементы.

3.3. Чем обосновано предпочтение гибридных технологий нативным

В данном новостном приложении гармонично сочетаются веб и мобильные технологии, так как оно запускается в качестве веб приложения по адресу <https://family-60ba7.web.app/> , и в то же время аналогично работает в качестве мобильного приложения на мобильном устройстве. При написании приложения использовался фреймворк Vue.js и платформа разработки мобильных приложений - Apache Cordova, что позволило сделать приложение мультиплатформенным, или иначе, дало возможность запускать приложение как на платформе Android, так и на IOS, BlackBerry и других, но одним из самых важных достоинств данного гибридного приложения является возможность изменения кода в одном месте и, после деплоя, получить результат одновременно в веб и мобильной версии. Этот аспект позволяет сосредоточиться на расширении возможностей приложения не дублируя их отдельно в мобильном приложении и веб приложении, что так же сокращает время разработки, ускоряет время релиза нового функционала и снижает вероятность устаревших версий (веб или мобильных), пока разрабатывается одна их них. Гибриное мобильное приложение было написано как веб приложение и запаковано в нативную обертку, что ускорило процесс его написания и расширило возможности применения. Для создания приложения был использован один язык – JavaScript, что значительно повлияло на скорость написания приложения, так как в случае с нативным приложением необходимо было бы изучать язык Object C/Swift для платформы IOS и Kоtlin для платформы Android, что значительно повлияло бы на качество работы и затраченное время. В конечном результате получилось универсальное веб и мобильное приложение, удовлетворяющее потребности пользователей, предпочитающих desktop либо пользователей, не имеющих необходимого количества свободной памяти на телефоне, так и пользователей, предпочитающих мобильные приложения.

* 1. Установка и настройка среды разработки
     1. Node Package Manager (NPM)

Рекомендуется использовать NPM при создании больших приложений на Vue. Эта опция прекрасно работает с такими инструментами сборки, как Webpack и Browserify. Vue также имеет совместимые инструменты для использования однофайловых компонентов. Команда установки: npm install vue устанавливает последнюю стабильную версию.

* + 1. Инструменты командной строки (CLI)

Vue.js предоставляет инструкции (CLI) для быстрого создания инфраструктуры для амбициозных одностраничных приложений. Всего за несколько минут вы получите рабочую конфигурацию с перезагрузкой модулей, анализом ошибок при настройке производственной сборки.

CLI - это инструмент для тех, кто знаком с Node.js и соответствующими инструментами сборки. Команда установки: npm install vue-cli.

* + 1. Использование Cordova для определения гибридной трансляции

Средство командной строки Cordova распространяется как пакет npm в готовом к использованию формате. Нет необходимости для его компиляции из исходного кода.

Чтобы установить инструмент командной строки cordova, необходимо выполнить следующие действия:

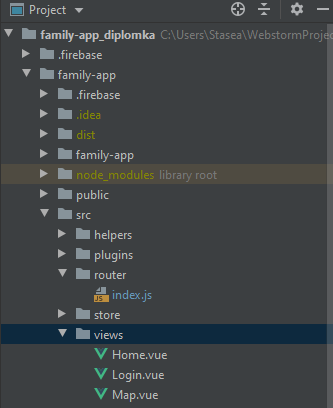
* Выполнить предыдущие 2 пункта;
* Установить модуль cordova с помощью инструментаnpm Node.js.

Модуль cordova будет автоматически загружен npm при вводе команды npm install –g cordova.

* + 1. Создание приложения

Разработка приложений в контексте рабочего пространства VueJs. Рабочая область содержит файлы для одного или нескольких проектов. Чтобы создать новое рабочее пространство и начальный проект для приложения, нужно выполнить команду CLI: vue create vue-app, vue-app в данном случае это название создаваемого проекта, оно может быть произвольным.

Следуя этим командам, все файлы, необходимые для начала создания приложения, а также основные компоненты App.Vue и main.js динамически создаются в выбранной папке (Рис.2.2). Далее нужно установить Vuetify. Для этого нужно ввести команду *vue add vuetify.* После установки Vuetify нужно установить cordova через команду *vue add cordova*.



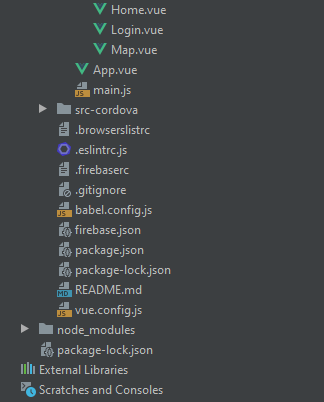


Рис.2.2. Стуктура vue сli проекта.

* + 1. Создание проекта в Firebase

Если учетная запись Google уже существует, можно сразу войти в систему на странице https://firebase.google.com/. В противном случае необходимо зарегистрироваться по адресу http:// accounts.google.com. После входа можно создать свой проект. После нажатия кнопки Add project (Добавить проект) нужно ввести название проекта и выбрать страну. Нажать кнопку Create Project (Создать проект), чтобы перейти в консоль Firebase. Здесь конфигурируется база данных и аутентификация, а также генерируются ключи, которые нужны для начала работы. В разделе Database (База данных) есть два варианта: Realtime Database и Cloud Firestore. В проекте используется Realtime Database.

* + 1. Интеграция Firebase

После настройки Firebase, следует его интеграция в приложение. Для этого файлы из библиотеки Firebase импортируются в файл проекта main.js.

import \* as firebase from "firebase/app";

import "firebase/analytics";

import "firebase/auth";

import "firebase/storage";

import "firebase/firestore";

Далее необходимо авторизовать приложение в Firebase. Для этого приложение инициализируется следующих свойств:

firebase.initializeApp({

apiKey: "AIzaSyDSwwHp51-LlwzIak0WXF8cScPMSS9VBwU",

authDomain: "family-d2146.firebaseapp.com",

databaseURL: "https://family-d2146.firebaseio.com",

projectId: "family-d2146",

storageBucket: "family-d2146.appspot.com",

messagingSenderId: "117478996669",

appId: "1:117478996669:web:e16a09e0f1bcf69ccfd5d4"

});

Для того, чтобы firebase был доступен глобально для всех компонентов приложения, необходимо написать следующее:

Vue.prototype.$firebase = firebase;

Далее происходит создание файла Login.vue, служащего для авторизации пользователя в приложении. Следом происходит создание интерфейса авторизации при загрузке страницы.

this.ui = new this.auth.AuthUI(this.$firebase.auth());

Далее был отрисован интерфейс в таг с id #firebaseui-auth-container.

this.ui.start("#firebaseui-auth-container")

<v-container class="auth-container" fluid>

<div id="firebaseui-auth-container"></div>

</v-container>

Вслед за этим новый объект интерфейса(инстанс) интерфейса передает callback, который вызывается при успешной регистрации.

signInSuccessWithAuthResult: authResult => {

const { user } = authResult;

this.$store.dispatch("setUserInfo", {

displayName: user.displayName,

email: user.email,

photoURL: user.photoURL

});

this.$router.push({ name: "Home" });

Код описанный выше запишет в глобальное хранилище store все данные пользователя для дальнейшего из использования в меню и при сохранении покупок приложения и перенаправит его на главную страницу.

Далее были описаны методы авторизации в приложении

signInOptions: [

this.$firebase.auth.GoogleAuthProvider.PROVIDER\_ID,

this.$firebase.auth.EmailAuthProvider.PROVIDER\_ID

]

Далее был написан метод для проверки авторизации при загрузке страницы и обновления данных в store, если они актуальны, в ином случае переадресации на страницу авторизации

this.$firebase.auth().onAuthStateChanged(user => {

if (user) {

this.$store.dispatch("setUserInfo", {

displayName: user.displayName,

email: user.email,

photoURL: user.photoURL

});

} else {

this.$router.push({ name: "Login" });

}

});

После проверки авторизации пользователь увидит все текущие записи на главной странице.

this.$firebase

.firestore()

collection("goods")

.orderBy("date", "desc")

onSnapshot(querySnapshot => {

this.items = [];

querySnapshot.forEach(doc => {

this.items.push(doc.data());

});

});

Затем из базы данных загружается firestore отсортированная коллекция goods и записывается в массив items для отрисовки на странице

<v-card class="mx-auto" :key="index" v-for="(item, index) of items">

<v-list-item>

<v-list-item-avatar color="grey">

<v-img :src="item.photoURL"></v-img>

</v-list-item-avatar>

<v-list-item-content>

<v-list-item-title class="headline">{{

item.displayName

}}</v-list-item-title>

<v-list-item-subtitle>{{ item.email }}</v-list-item-subtitle>

</v-list-item-content>

</v-list-item>

<v-img v-if="item.image" :src="item.image"></v-img>

<v-card-text>

{{ item.text }}

</v-card-text>

<v-card-actions>

<v-btn text color="deep-purple accent-4"> {{ item.price }} $ </v-btn>

<v-spacer></v-spacer>

<v-btn text color="deep-purple accent-4">

{{ getFormattedDate(item.date) }}

</v-btn>

</v-card-actions>

</v-card>

После отрисовки данных пользователь может просмотреть или добавить новые записи. Ниже приведен пример кода для отображения кнопки добавления

<v-btn @click="dialog = true" color="primary" bottom right>

<v-icon>mdi-plus</v-icon>

</v-btn>

При нажатии на кнопку пользователю будет предложено создать новую запись, написать название, цену и прикрепить картинку. После сохранения покупки данные будут отравлены на сервер.

if (this.image) {

const snapshot = await this.$firebase

.storage()

.ref()

.child(`images/${this.generateUuid()}`)

.put(this.image);

image = await snapshot.ref.getDownloadURL();

}

Код выше проверяет наличие картинки, и загружает ее на сервер. Для того что бы они не повторялись для каждой был сгенерирован уникальный номер. После загрузки картинки на сервер, в переменную записывается ее ссылка для сохранения базе данных. После получения ссылки, все данные сохраняются в базу данных, в коллекцию под названием goods. Ниже приведен пример кода:

try {

await this.$firebase

.firestore()

.collection("goods")

.add({

text: this.text,

price: this.price,

date: new Date(),

...this.user,

image

})

После добавления данных в базу диалоговое окно будет закрыто, а данные полей сброшены.

# Заключение

В данной работе было произведено исследование гибридных web приложений были рассмотрены преимущества и недостатки гибридных web технологий, были сравнены разные JavaScript фреймворки, из которых, на основе сравнения, был выбран самый подхоящий для разработки гибридного мобильного приложения – Vue.js, были изучены основные элементы фреймворка, а также, как работают технологии, имплементированные в данное демонстрационное приложение, такие как Firebase, Google OAuth, Cordova, Vue, автономная работа приложений и другие. Был произведен анализ создания web приложений применяя соответствующий набор технологий.

В процессе выполнения работы были:

* Изучены основы гибридных web приложений;
* Исследованы возможности использования гибридных web технологий в качестве замены нативных технологий;
* Исследованы и сравнены фреймворки языка JavaScript
* Произведен анализ работы современных облачных технологий для интеграции в приложения;
* Разработано приложение с использованием данного набора технологий.

В ходе данной работы были рассмотрены преимущества гибридных web технологий, были изучены их основные элементы , были исследованы фреймворки JavaScript, на которых возможно осуществить написание гибридного веб приложения их основные элементы, а также, как работают ключевые технологии, входящие в данный набор, такие как Firebase, Google OAuth, Cordova, Vue, автономная работа приложений и другие.

На основе всего вышесказанного, можно сформулировать вывод, что гибридные web технологии являются крайне полезными и инновационными инструментами для создания мобильных приложений. Гибридные web приложения включают в себя большинство преимуществ нативных приложений, при этом, технически, оставаясь веб приложением.

В заключении можно добавить, что в перспективе гибридные web технологии позволят убрать проблему кроссплатформенности, так как они позволяют вести разработку одновременно под все платформы сразу. Это приведет к более прогрессивному, прямолинейному и рентабельному пути разработки приложений, ведь теперь компаниям не нужно будет беспокоиться о разработке приложения сразу под несколько разных платформ, что выражается не только в количестве времени, потраченного на разработку, но и количества и специализации самих разработчиков, необходимых для создания продукта.

Данная работа предназначена для исследования гибридных web приложений, а также не только для ознакомления с методами и архитектурой web приложений на основе гибридных web технологии, но исследования и выбора подходящего фреймворка для создания гибридного мобильного приложения, которое является индивидуальным для каждого продукта в соответствии с требованиями и возможностями заказчиков, которая может быть полезна разработчикам, студентам, учителям и всем желающим ознакомиться с гибридными web приложениями.

# Библиография

1. *Определение гибридных мобильных приложений. Доступен:*

<https://www2.stardust-testing.com/en/blog-en/hybrid-apps> [on-line].[цитирован 15.02.21];

1. *Типы мобильных приложений Доступен:*

<https://qastart.by/mainterms/59-tipy-mobilnykh-prilozhenij> [on-line].[цитирован 17.02.21];

1. *Сравнение гибридных, мобильных и веб приложений*

<https://csslike.me/mobil-ny-e-prilozheniya-nativny-e-vs-html5-vs-gibridny-e/> [on-line].[цитирован 26.02.21];

1. *Нативные vs. гибридные приложения*. Доступен: <https://umbrellait.com/ru/blog/native-vs-hybrid-app/> [on-line].[цитирован 10.05.20];
2. *Разница между нативным и гибридным мобильным приложением*. Доступен: <https://wezom.com.ua/blog/chem-otlichajutsja-nativnoe-i-gibridnoe-mobilnye-prilozhenija> [on-line].[цитирован 10.05.20];
3. *Фреймворки JavaScript*

<https://rubygarage.org/blog/best-javascript-frameworks-for-front-end> [on-line].[цитирован 01.03.21];

1. *Прогрессивный фреймворк Vue.js*

<https://ru.vuejs.org/v2/guide/index.html> [on-line].[цитирован 10.03.21];

1. *Callum Macrae. Vue.js: Up and Running, 2017 ISBN 9781491997246;*
2. *Бэнкс Алекс, Порселло Ева. GraphQL: язык запросов для современных веб-приложений, 2019 ISBN 9785446111435;*
3. *Томас Марк Тиленс. React в действии, 2019 ISBN 9785446109999;*
4. *Фреймворк Angular* <https://habr.com/ru/post/348818/> [on-line].[цитирован 22.03.21];
5. *Сравнительный анализ фреймворков Vue, React и Angular*

<https://proglib.io/p/vue-vs-react-vs-angular-kakoy-frontend-freymvork-vybrat-2020-08-03> [on-line].[цитирован 15.03.21];

1. *Поверхностный обзор javascript-enabled BaaS платформ*. Доступен: <https://habr.com/ru/post/150729/> [on-line].[цитирован 15.03.20];
2. *Используйте Firebase, единый кроссплатформенный SDK от Google, чтобы улучшить приложения.* Доступен: <https://developer.android.com/distribute/best-practices/develop/build-with-firebase?hl=ru> [on-line].[цитирован 27.03.20];
3. *Обзор облачных сервисов для разработки бэкенда мобильных приложений*. Доступен: <https://habr.com/ru/company/surfstudio/blog/463435/> [on-line].[цитирован 07.04.20];
4. *OAuth 2.0 простым и понятным языком*. Доступен: <https://habr.com/ru/company/mailru/blog/115163/> [on-line].[цитирован 10.04.20];
5. Описание сервиса Google Maps <https://wiki.rookee.ru/google-maps/> [on-line].[цитирован 10.04.21];
6. *Использование сервисов Firebase при разработке мобильных приложений*. Доступен: <https://cmsmagazine.ru/journal/cases-4264/> [on-line].[цитирован 16.04.20];
7. *Vue.js*. Доступен: <https://ru.vuejs.org/v2/guide/index.html> [on-line].[цитирован 10.05.20];

Declaralie privind asumarea raspunderii

Subsemnata \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, absolventa al Universitdtii de Stat “Alecu Russo” din Balti, Facultatea de \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ program de studii \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, declar pe propria raspundere ca teza de licenta/teza de master cu titlul \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, elaborata sub indrumarea dlui/dnei \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ este rezultatul propriilor cercetdri.

Declar ca nu am plagiat altd lucrare de licenta/master, monografii, articole etc., publicate sau postate pe Internet, toate sursele utilizate in tezi fiind mentionate in cuprinsul acesteia.

De asemenea, declar ca sunt de acord ca teza de licenta/teza de master sa fie verificata pentru confirmarea originalitilii.

Data Semnatura