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 20 de surse, 44 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 20 источников, 40 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, two chapters, conclusions, a list of references from 20 sources, 44 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 2](#_Toc41253564)

[ВВЕДЕНИЕ 3](#_Toc41253565)

[1. ИСПОЛЬЗОВАНИЕ ГУГЛ СЕРВИСОВ В СОВРЕМЕННЫХ ИНФОРМАЦИОННЫХ СИСТЕМАХ 5](#_Toc41253566)

[1.1. Один из многофункциональных сервисов гугл - Firebase 6](#_Toc41253567)

[1.1.1. Аутентификация в Firebase 6](#_Toc41253568)

[1.1.2. Firebase Хостинг 7](#_Toc41253569)

[1.1.3. Cloud Storage 8](#_Toc41253570)

[1.1.4. База данных реального времени Firebase 9](#_Toc41253571)

[1.2. Google OAuth 10](#_Toc41253572)

[1.2.1. Абстрактное описание протокола 11](#_Toc41253573)

[1.2.2. Регистрация приложения 12](#_Toc41253574)

[1.2.3. Идентификатор клиента и секрет клиента 12](#_Toc41253575)

[1.2.4. Итог по первой главе 12](#_Toc41253576)

[2. ГИБРИДНЫЕ ПРИЛОЖЕНИЯ ИСПОЛЬЗУЯ VUEJS И VUETIFY 13](#_Toc41253577)

[2.1. VueJS при создании пользовательских интерфейсов………………………...14](#_Toc41253578)

[2.2. Vuetify - Framework для создания гибридных мобильных приложений 15](#_Toc41253579)

[3. ВНЕДРЕНИЕ ГУГЛ-СЕРВИСОВ В СИСТЕМЫ, ИСПОЛЬЗУЮЩИЕ VUEJS 17](#_Toc41253580)

[3.1. Описание структуры приложения 17](#_Toc41253581)

[3.2. Установка и настройка среды разработки 17](#_Toc41253582)

[3.2.1. Node Package Manager (NPM) 17](#_Toc41253583)

[3.2.2. Инструменты командной строки (CLI) 18](#_Toc41253584)

[3.2.3. Использование Cordova для определения гибридной трансляции 18](#_Toc41253585)

[3.2.3. Создание приложения 18](#_Toc41253586)

[3.3. Создание проекта в Firebase 19](#_Toc41253587)

[3.4. Интеграция Firebase 19](#_Toc41253588)

[Заключение 24](#_Toc41253589)

[Библиография 25](#_Toc41253590)

# 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 WEB ПРИЛОЖЕНИЯ

* 1. Понятие гиб веб прил

Выбор подходящей модели мобильного приложения — это очень важный этап в его разработке, на который влияют несколько факторов, таких как техническая оценка разработчиков; потребность в доступе к информации на устройстве; влияние скорости интернета на приложение; одно- или многоплатформенное ли приложение.

Гибридные приложения являются чем-то средним между нативными и веб-приложениями. Фактически они создаются так, чтобы выглядеть и использоваться как нативные приложения. Их также устанавливают на телефон пользователя и их можно найти в магазинах приложений. Различие заключается в том, что они обязательно должны размещаться в рамках нативного приложения и созданы, чтобы работать через WebView, и таким образом они могут получать доступ к информации на устройстве пользователя для больших возможностей.

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.

Что говорит статистика: тренды в мобильной разработке

Если еще пару лет назад компании делали ставки на конкретный опыт в разработке нативных приложений под Android и iOS, то теперь в качестве альтернативы они рассматривают кросс-платформенные решения. Это можно объяснить потребностью бизнеса в быстрой разработке относительно простых приложений.

Многим компаниям приложения необходимы быстрее, чем они могут быть разработаны. По словам главного аналитика Gartner Эдриана Леова, «быстрая разработка и постоянное поддержание мобильных приложений затруднены из-за растущего спроса, ускоренного развития цифровых технологий и дефицита кадров».

В целом, заказчиков мобильной разработки можно условно разделить на 2 группы:

Первым требуется создание сложных, ресурсоёмких приложений с наличием многослойной анимации и вычислительными функциями.

Для вторых актуальны приложения с адаптивным интерфейсом, без сложной анимации и хранения большого количества данных непосредственно в памяти девайса.

Очевидно, что нативная разработка является лучшим выбором для первых, тогда как гибридный подход больше отвечает потребностям вторых.

В своей практике мы тоже столкнулись с растущим спросом на создание мобильных приложений. При этом, часть обращений направлена на приложения с простым функционалом и возможностью их применения пользователями разных мобильных платформ. Это вполне закономерно. 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.

Отсюда следует вывод, что гибридная разработка становится не менее востребованной, чем нативная.

Фактически, мы создаем 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.

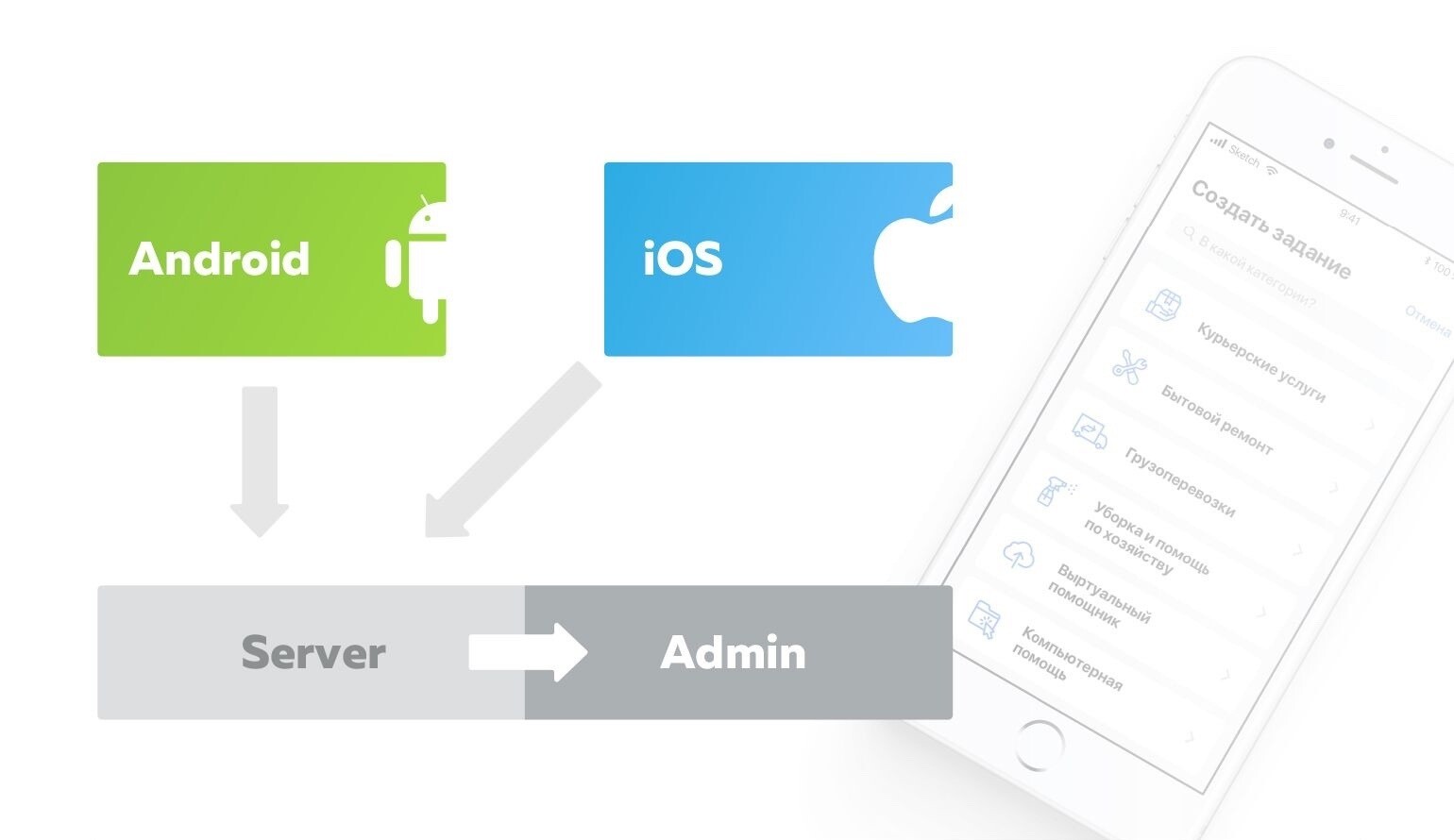
Широкая популярность гаджетов вполне закономерно приводит к тому, что на рынок приходят новые производители. А вместе с ними – их собственные разработки, которые находят себя в приложениях. Это так называемые нативные приложения, то есть доступные только на одной определенной платформе или операционной 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. В свою очередь, к слову, та же операционная система Windows Phone также во многом базируется на собственных сервисах.

В свою очередь, гибридные приложения по своей сути более универсальны и чаще всего подходят для разных систем (в контексте данного материала мы будем отталкиваться от IOS и Android – двух флагманов сегодняшнего мира гаджетов). 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.

**https://medium.com/nuances-of-programming/%D0%BD%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D1%8B%D0%B5-%D0%B3%D0%B8%D0%B1%D1%80%D0%B8%D0%B4%D0%BD%D1%8B%D0%B5-%D0%B8-web-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F-%D0%B2-%D1%81%D1%80%D0%B0%D0%B2%D0%BD%D0%B5%D0%BD%D0%B8%D0%B8-b1360258df2d**

* 1. Сравнение гиб прил и нативных

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.

Вот теперь мы подходим к отличиям. А отличия заключаются в том, какой объём функционала несёт клиентская часть, а какой серверная. Непонятно? Давайте начнём с определений

Что такое натив, а что такое гибрид

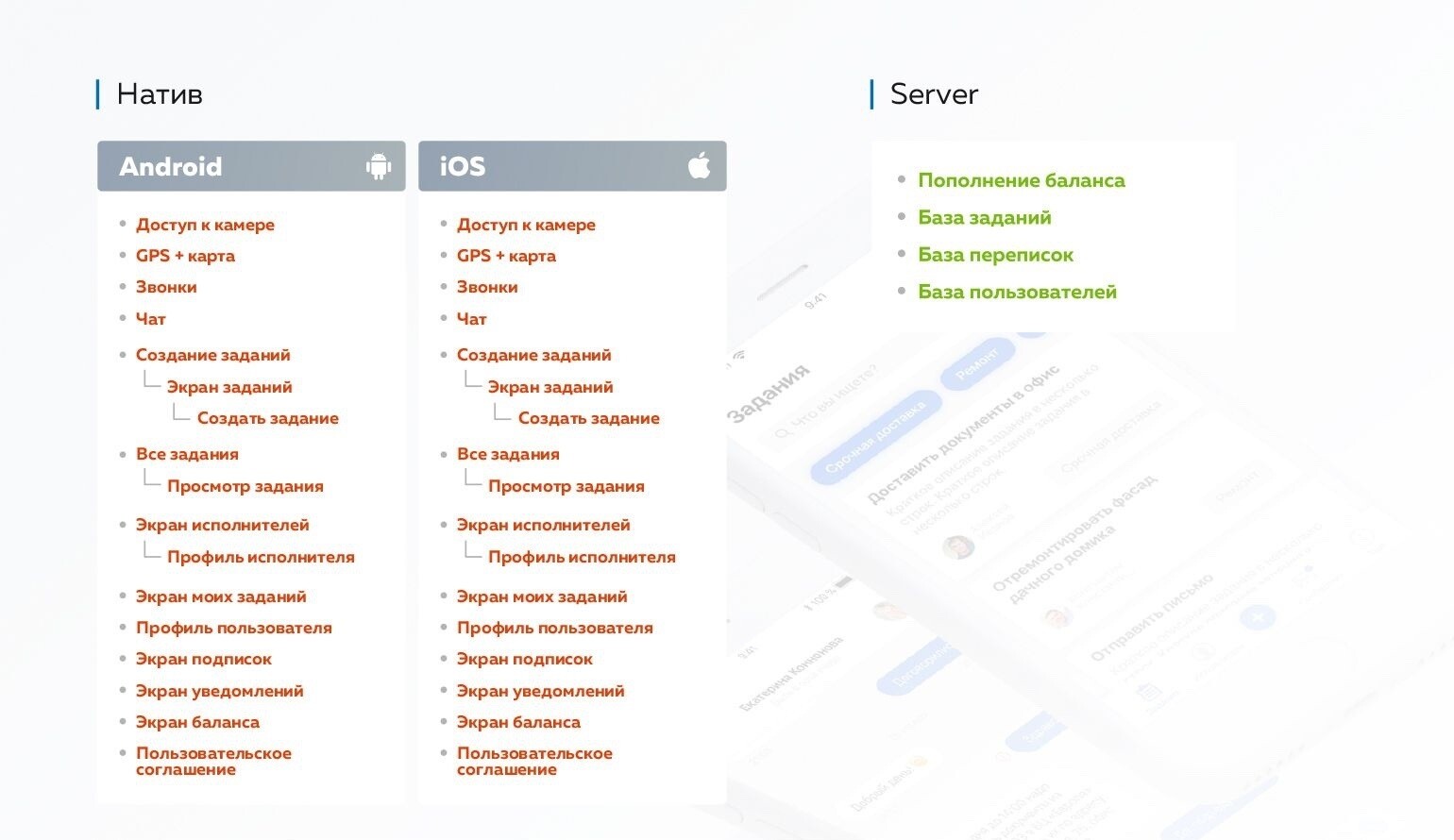
Чтобы было понятно большинству пользователей я буду писать не строгие технические определения (их можно найти в википедии), а расскажу на пальцах, да простят меня программисты.

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.

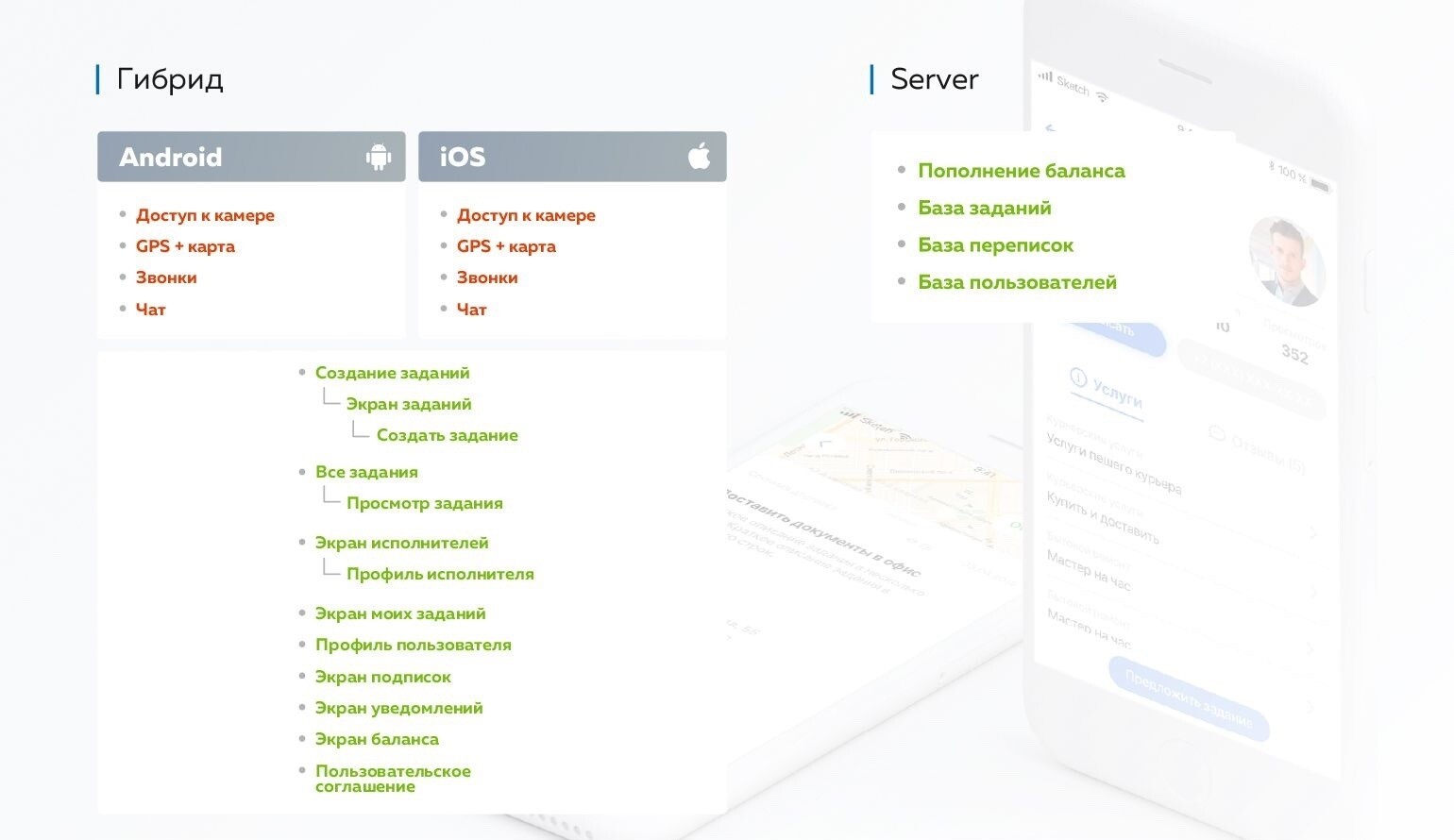
Гибрид - это скорее подход к программированию, чем какой-то особенный вид приложения. Его принцип заключается в том, что всё что можно (читай "все функции не связанные с железом") программируется на стороне сервера, а на стороне клиента остаётся только необходимый минимум. Если рассматривать типовой 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:



Расчёт стоимости разработки приложений

Можно долго спорить о стоимости часа разработки, опыте и т.д. Естественно, на рынке есть большой разбег, но давайте ориентироваться сравнительные метрики. Кто нам потребуется для такого проекта:

Java-программист для андройда

Objective-C/Swift-программист для айоса

php-программист для сервера. Можно, конечно брать python-разработчика, но для чего он конкретно здесь, непонятно, поэтому, ориентируемся на php, т.к. он более дешёвый.

Нужно учитывать, что Андройд и айос программисты в среднем стоят примерно в два раза дороже php.

При полностью нативной разработке мы заплатим программистам приложений за полный функционал, а в случае гибрида только за нативные функции (отмечено красным). Оставшийся же функционал будет реализовывать php-шник. Причём, нативщики будут делать это всё и для iOS и для Android отдельно, а при гибриде php-шник делает эти экраны один раз для обеих систем. Получается, что почти двукратная работа выполняется специалистами, которые в 2 раза дороже. Путём не хитрых вычислений приходим к тому, что полностью нативное приложение будет стоить ~4 раза выше гибридного. Это, конечно, зависит от того какие нативные функции нужны в приложении и как они применяются, но при стандартном бизнес-приложении (GPS, камера, интернет) экономия в 3-4 раза.

Что выбрать для проекта

Конечно же, у натива преимущества вытекают из той же концепции:

"тяжёлые" вычисления, которые нужны на стороне телефона (например, работа с 3D, обработка виртуальной реальности, сложная анимация)

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.

Эра смартфонов

Большинство моих друзей ни на секунду не расстаются со своими смартфонами или планшетами: прочитать новости, посмотреть пробки, узнать погоду и проверить почту — хорошо знакомый многим из нас сценарий обыкновенного утра. Мой муж, например, напевает в душе исключительно под звуки радио-приложения, установленного на его телефоне.

Очевидный рост числа пользователей мобильных устройств неизменно влечет за собой бурное развитие рынка мобильных приложений. Неудивительно, что владельцы бизнеса также стремятся стать «мобильнее».

Если человек уже принял решение о необходимости создания мобильного приложения, один из следующих этапов — определиться, какой тип приложения выбрать. Будет ли это нативное приложение? Или лучше выбрать веб-приложение? Что такое гибридное приложение, и почему его предлагают? Если Вам не понятно, о чем это я, то эта статья для вас. Если понятно, то вы сможете систематизировать свои знания или укрепиться в своем правильном выборе. И, при желании, написать мне о своем мнении или задать вопросы.

Итак, из этой статьи Вы узнаете:

Какие бывают мобильные приложения;

Какие преимущества Вы получаете при выборе определенного типа приложений;

Какие существуют ограничения у разных типов приложений;

Какое приложение подойдет именно вам в конкретной ситуации.

Нативные приложения

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 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.

Первый пример — приложение Shazam, осуществляющее определение и поиск информации об играющей на другом устройстве песне (в App Store, в Google Play):

Устанавливается из магазина приложений;

Для работы необходим доступ в интернет;

Использует диктофон телефона.

Второй пример — не менее известное приложение Instagram (описание и ссылки для скачивания в магазинах):

Устанавливается из магазина приложений;

Для работы также необходим доступ в интернет;

Использует ПО смартфона: камера, геолокация, адресная книга; Можно включить получение push-уведомлений.

Веб-приложения, или приложения на html5

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.

Независимо от установленной ОС такие приложения не могут использовать ПО смартфона. Для обновления информации в приложении необходимо подключение к интернету, скорость работы ограничена возможностями интернет-соединения провайдера услуг. При желании продавать приложение вам потребуется собственная платежная система.

На самом деле, грань между веб-сайтом, оптимизированным под мобильное устройство или с адаптивной версткой, которая способствует адекватному его отображению на любом устройстве, и веб-приложениям очень тонкая. Как разработчики, так и пользователи в некоторых случаях путаются.

Например, last.fm считается веб-приложением, хотя, по сути, это в тоже время и веб- сайт. Или maps.google.com — веб-сайт, но в тоже время это и веб-приложение.

Дело в том, что веб-технологии развиваются так стремительно, что разница начинаетnразмываться, и сайты все более становятся похожими на веб-приложения. Разницу, хотя и спорно, можно по-простому описать так: сайт представляет собой в большей степени статическую информацию (по сути, цифровая брошюра или листовка); а если пользователь с этойnинформацией может взаимодействовать (менять тексты местами, менять оформление, создавать

собственные страницы и т.д.), то это уже веб-приложение. Еще разницу можно объяснить так: то, что принято называть онлайн-сервисами, а также то, что раньше делалось на флешовых сайтах, а сейчас с помощью html5, то веб-приложение.

Здесь осталось только заметить, что не все веб-приложения оптимизированы под мобильные телефоны. Что, конечно, не может быть нашим случаем.

Гибридные приложения

Гибридные приложения сочетают в себе некоторые функции нативных и веб-приложений: кроссплатформенность и возможность использования ПО телефона. Такие приложения могут быть загружены через магазины приложений, и при этом имеют возможность независимого обновления информации. Гибридные приложения требуют подключения к интернету, поскольку веб часть обновляется через интернет. Это, наверное, самый популярный способ построения мобильных приложений, так как у него органическая среда распространения, но разработка происходит быстрее и дешевле, чем в случае с нативными приложениями, так как, хотя оболочка и написана на «родно» языке программирования, «начинка» может быть написана в том или ином объеме на html5. Пользователь же скорее всего не заметит разницу между нативным приложением и гибридным.

ПРИМЕРЫ ГИБРИДНЫХ ПРИЛОЖЕНИЙ:

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

Загружается из магазина;

Использует камеру телефона;

Необходимо подключение к интернету при желании поделиться результатом своей работы;

Можно настроить push-уведомления.

2. Приложение TripCase — органайзер для планирования путешествий. (Ссылки для скачивания в магазинах доступны на сайте )

Загружается из магазина;

Может использовать геолокацию;

Необходимо подключение к интернету;

Может использовать сотовую сеть;

Можно настроить push-уведомления

Таблица преимуществ и недостатков

[**https://ratingruneta.ru/abc-moblile/mobile-apps-native-vs-html5-vs-hybrid/**](https://ratingruneta.ru/abc-moblile/mobile-apps-native-vs-html5-vs-hybrid/) **доделать таблицу**

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. В свою очередь, к слову, та же операционная система Windows Phone также во многом базируется на собственных сервисах.

В свою очередь, гибридные приложения по своей сути более универсальны и чаще всего подходят для разных систем (в контексте данного материала мы будем отталкиваться от IOS и Android – двух флагманов сегодняшнего мира гаджетов). 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.

Пожалуй, единственный минус любого пусть и отличного, но официального контента – это его цена. Да, вам придется разрабатывать отдельное приложение под каждую платформу.

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.

С тем, что это нативное приложение и гибридное мы разобрались. Чтобы поставить решающую точку в выборе нужной модели, необходимо уточнить нюансы разработки. Сначала разберем общий алгоритм разработки мобильных приложений, а после разберем нюансы каждого из них.

Общий алгоритм представляет собой следующую цепочку мероприятий:

Подготовка. Проводится ознакомление с брендом, компанией, организацией или физическим лицом, которые заинтересованы в реализации своего проекта в качестве выпуска мобильного приложения. Структурируется вся необходимая информация о деятельности и формируется общая цель будущей разработки.

Подготовка прототипа. Перед началом осуществления разработки все необходимо зафиксировать в “черновом” варианте не бумаге. Осуществляется подготовка технического задания с детальным описанием каждого рабочего момента (от незначительных функций до анимации и медиа насыщения). Проводится создание прототипа с переносом проекта на мобильное устройство. Фиксируются ошибки и недочеты, если таковые имеются. А также проводятся первые этапы создания дизайна.

Реализация запланированного проекта. После учета всей специфики проводится осуществление выхода нового программного продукта для гаджетов. На данном этапе пишется спецификация API, описываются экраны, проводится программирование (выбирая язык под необходимую операционную систему), проведение тестового периода для оценки эффективности. В результате мобильное приложение выпускается в продакшн.

Нативное приложение: в чем специфика?

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).

Если смотреть процесс разработки со стороны пользователя, то нативное мобильное приложение может выступать в роли профессиональной разработки кроссплатформенного варианта программы. Выделяется более понятный интерфейс. Так называемый «нативный вариант юзабилити».

UI/UX – это та комбинация, которая является максимально эффективной при реализации проекта с целью создания нативного продукта. Дизайнер готов правильно определить, какие подходы UI будут «радовать глаз» пользователя, учитывая конкретную операционную систему.

Так учитываются некоторые фишки для более комфортного использования предложенного функционала. Например, актуально применять кнопку возврата на клавишу Tab Bar, расположенную на гаджетах системы IOS. В результате пользователь может полноценной применять нативное мобильное приложение, которое будет интуитивно взаимодействовать с особенностями интерфейса.

Обратившись на сайт разработки мобильных приложений для Andorid и iOS вы сможете узнать более подробно у поддержки о том, какие есть нюансы разработки тех или иных приложений.

Особенности разработки нативных мобильных приложений

В данном случае применение нативной среды уходит на второй план. А первый в свою очередь будет включать в себя использование веб-технологий. На выходе получается реализация не целой страницы, а автономное приложение. Последнее подается в обновленной «обертке», которая имеет отдельную иконку и подразумевает наличие установочного процесса.

Разработчики кроссплатформенных движков часто сталкиваются с нативными мобильными приложениями. Это связано с функцией веб-браузера, которая одинаково эффективно работает на нескольких мобильных операционных системах. Так можно запускать приложения, используя разные системы. При это не будут возникать трудности.

Заключение

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.

Фактически, мы создаем 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.

Плюсы и минусы гибридного подхода

По большему счёту, к основным достоинствам гибридного подхода относятся:

* Кросс-платформенность. Сделав одно приложение, можно экспортировать его под любую операционную систему – iOS, Android, Windows Phone, BlackBerry.
* Доступная стоимость разработки – в разы меньше, чем при нативном подходе.
* Использование одного языка – JavaScript.
* Доступ к основным данным мобильного устройства: GPS, камера, телефонная книга и т.д.
* Работа в оффлайн-режиме.
* Возможность распространять приложение через официальные магазины приложений.

Однако у гибридной разработки есть свои слабые места:

* Сниженная скорость работы в отличие от нативных приложений.
* Меньше возможностей для интеграции с «начинкой» мобильного устройства.
* 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.

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.

Хотя некоторые недостатки не так существенны, низкая производительность и проблемы с анимацией могут стать ограничениями в работе с кросс-платформенными приложениями. Два популярнейших сервиса Facebook и LinkedIn в своё время работали по гибридной модели, но затем были переписаны на нативную. По отзывам разработчиков, после этого они получили двукратный прирост в производительности решений.

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

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

Судя по динамике развития наших мобильных проектов, мировые тенденции подтверждаются на практике – увеличивается спрос на мобильную разработку, растут скорости создания приложений, повышается интерес к несложным решениям с доступной стоимостью разработки. Всё это открывает возможности для совершенствования гибридных приложений, которые в отдельных случаях могут проявлять себя не хуже нативных. Отдавая предпочтение тем или другим, важно отслеживать состояние рынка и правильно расставлять приоритеты по скорости запуска проекта, стоимости и производительности.

Плюсы нативных приложений

Во-первых, это широкий круг разработчиков, а также своевременная и качественная техническая поддержка клиентов, ведь большие компании могут позволить себе вкладывать серьезные финансовые ресурсы в свой собственный продукт. Да и потом, пользователь никогда, как иногда бывает в случае с гибридными приложениями, не останется наедине со своей проблемой. Хотя бы потому, что тот же IOS или Android, как показывает практика, и правда очень хорошо дают обратную связь своим клиентам: помогут, настроят и исправят неисправимое.

Во-вторых, это повсеместное распространение и высокий процент релевантности устройству. Вряд ли нативное приложение сможет удостоиться такой чести, как отказ в размещении на центральных пользовательских площадках Play Market или App Store. К тому же, скачивая конкретное нативное приложение, пользователь может быть практически на 100% уверен, что оно подойдет для его устройства, ведь оно заведомо соответствует всем параметрам конкретной платформы, а значит – наилучшим образом оптимизировано для нее. А значит, никаких «косяков» и багов ожидаться не должно.

В-третьих, нативное приложение – это не только сервис, подходящий вашему устройству по всем параметрам, но и конкретное качество, которое проявляется в чистом написании кода, широких возможностях функционала, а также в минимальных подтормаживаниях или неисправностях. Крупным компаниям нет смысла делать приложения и софт для своих устройств на скорую руку или «тяп-ляп». Ведь не станешь же ты кое-как клеить обои в квартире, которую рассматриваешь как постоянное место жительства?

В четвёртых, делая выбор в пользу нативного приложения, пользователь имеет огромный шанс получить себе не стандартный и унылый, выполненный в серых тонах интерфейс, а самый что ни на есть яркий и оригинальный дизайн вкупе с удобным и интуитивно понятным управлением. Среднестатистический пользователь с каждым годом становится все избирательнее в своих предпочтениях, так как параллельно с ростом планки требований он все больше и больше пресыщается всевозможным контентом. Вот как удивить человека, который уже видел ну, наверное, ВСЁ? Правильно, подключить все силы, запустить все ресурсы на полную и каждый раз создавать не рядовую блеклую побрякушку, а маленький технологический шедевр.

Минусы нативных приложений

Пожалуй, единственный минус любого пусть и отличного, но официального контента – это его цена. Да, вам придется разрабатывать отдельное приложение под каждую платформу.

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.

Плюсы гибридных приложений

Во-первых, это более широкий охват рынка. Ведь поскольку гибридные приложения выходят сразу на несколько платформ, то и пользователей будет в разы больше. К примеру, существует немало людей, которые пользуются несколькими системами одновременно. И им будет куда выгоднее использовать один привычный для себя софт, нежели разделять сферы собственного пользования.

Во-вторых, это опять-таки цена. Плюс многих гибридных приложений в том, что они позволяют сэкономить и не тратиться на разработку двух отдельных версий одного и того же продукта.

 Минусы гибридных приложений

В отличие от нативных вариаций, у гибридных приложений куда больше минусов.

Для начала, разрабатывая гибридное приложение, держите в уме, что некоторые нативные функции могут работать не до конца, или же их вообще будет невозможно реализовать.

Нужно быть готовым и к тому, что не всегда адаптивность приложения работает идеально, могут возникать баги на разных системах. Гибридные приложения плохо оптимизированы или же страдают сетевыми уязвимостями. Кому нужно приложение, если оно постоянно вылетает или лагает? А еще и имеет возможность с куда большей вероятностью, нежели нативный вариант, быть взломанным мошенниками.

Большая нагрузка ложится на Систему поддержки. Придется столкнуться с проблемами самых разных пользователей, с самыми разными устройствами, а вот пофиксить все быстро может не получиться из-за разнообразности проблем и необходимости постоянных обновлений. Отсюда и потенциальный негатив.

К сожалению, из-за более низкого бюджета и контроля за разработкой гибридные приложения часто весьма ограничены в своем интерфейсе, а также обладают непривлекательным дизайном, что в нынешний век все растущей интерактивности недопустимо для многих пользователей. Даже трекер шагов должен быть удобным, понятным и приятным на вид, что уж говорить про оформление банковских систем и различные редакторы.

Так что же выбрать?

На этот вопрос вам вряд ли кто-то даст конкретный и окончательный ответ. В конце концов, что плохо одному, то хорошо и приятно другому. Куда важнее знать все тонкости и подводные камни обоих вариантов, дабы найти конкретно для себя наиболее удобный и удовлетворяющий пользовательским запросам софт. И неважно, будет ли он нативным или же это окажется гибрид.

* 1. Примеры

Опыт гибридной разработки на примере PhoneGap

В полной мере оценить все особенности гибридной разработки можно только на собственном опыте. Когда число клиентских запросов на мобильную разработку выросло в разы, мы решили подключить к этому процессу нашу команду веб-разработчиков. После изучения и анализа технологических инструментов, основываясь на развернутой документации и перспективности решения, мы остановили свой выбор на PhoneGap.

PhoneGap — это открытый фреймворк, служащий для ускорения процесса разработки кросс-платформенных приложений. Именно PhoneGap позволяет делать приложения на HTML5 + JavaScript, после чего компилировать их в установочные файлы под любую операционную систему: iOS, Android, Windows Phone, BlackBerry.

Общий смысл гибридной разработки заключается в том, что мы работаем с WebView мобильного телефона с помощью HTML5/Javascript/СSS, а затем просто «пакуем» подобного рода разработку в нативную «обертку».

Благодаря использованию PhoneGap, мы:

* создаём мобильную версию сайта с адаптивной вёрсткой;
* получаем взаимодействие с основными функциями мобильного устройства, что обеспечивается за счёт плагинов PhoneGap, написанных на нативном языке и предоставляющих открытые методы JavaScript;
* гарантируем возможность интеграции приложения с любыми мобильными платформами.

Нативных, гибридных и веб-приложений есть достаточно большое количество, доступные в популярных markets. Каждый из них выделяется своим особенным функционалом, который несомненно повлияет на количество потенциальных пользователей вашего приложенияи соответственно на рост продаж. Подробнее о том как это влияет на продажи услуг и товаров вашего бизнеса читайте в статье. Для полной картины и понимания всех отличий, рекомендуем ознакомиться с примерами каждого варианта.

Нативное приложение:

«Shazam» — это первый пример, который пользуется популярностью у любителей применять гаджеты. С таким приложением можно определять и получать информацию о песне, которая играет на другом устройстве. Его можно спокойно установить из магазина, работая в приложении используется доступ в Интернет. А также необходим доступ к дополнительному серверу телефона – диктофон, позволяющий определять музыкальную композицию для поиска актуальной информации.

«Instagram» — одна из самых популярных социальных сетей, которая пользуется спросом у разной возрастной категории пользователей. Ключевые отличия приложения, что характеризуют его, как нативное, представлены возможность установить с AppStore и Play Market. А также необходимостью подключения к Интернету, применения программного обеспечения гаджета (камера, книга номеров, геолокация). При необходимости доступна возможность настроить push-уведомления.

Гибридное:

«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).

«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.

**Это во 2 главу**

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.

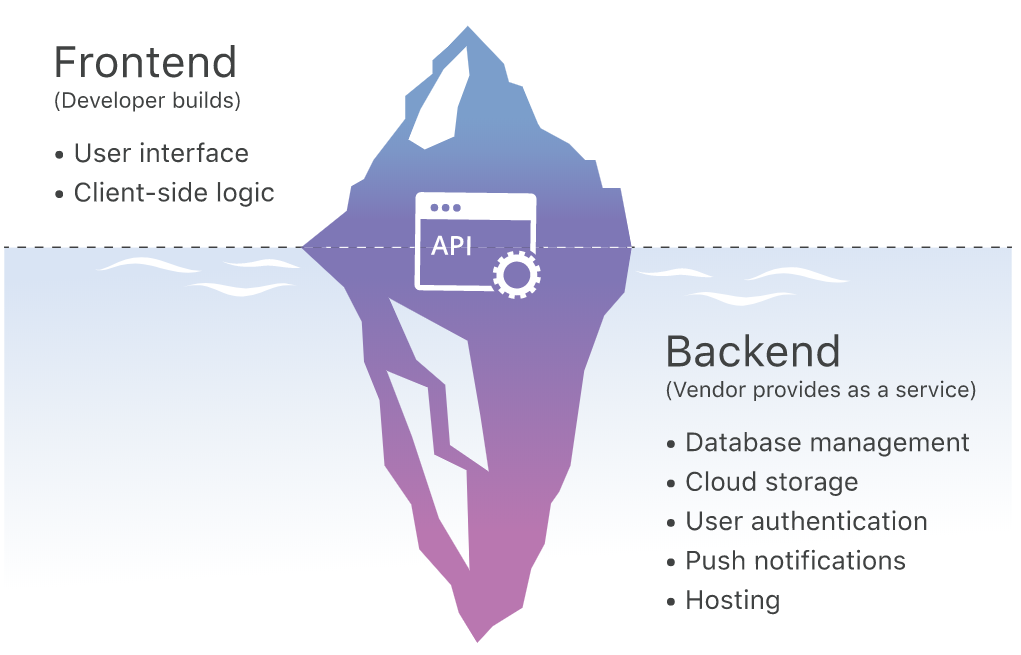


Рис.1.1. Принцип работы современного приложения.

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.

* 1. Один из многофункциональных сервисов гугл - Firebase

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.

* + 1. Аутентификация в Firebase

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.).

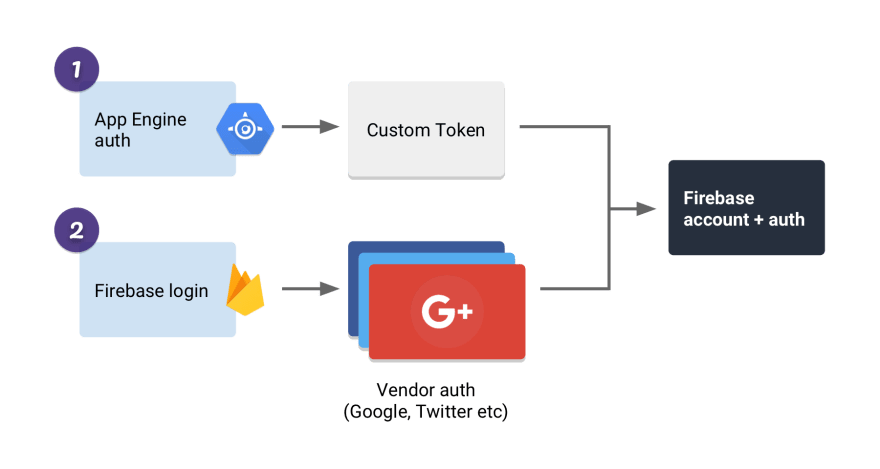


Рис.1.2. Принцип работы аутентификации*.*

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].

* + 1. Firebase Хостинг

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.

* + 1. Cloud Storage

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.

* + 1. База данных реального времени Firebase

База данных Firebase Realtime 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.

* 1. Google OAuth

OAuth 2 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.

1.2.1. Абстрактное описание протокола

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.

1.2.2. Регистрация приложения

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.

1.2.3. Идентификатор клиента и секрет клиента

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.

* 1. Сервис Google Map

Google Maps – это комплекс приложений, созданных на базе бесплатного сервиса картографии и технологии, используемой Google. Данное приложение используется для поиска информации на карте с отметками достопримечательностей, организаций и т.д [8].

Оформление и возможности Google Maps

Сервис предлагает такие режимы отображения, как:

«Спутник» – показ спутниковых и аэрофотографий (сюда относятся «Имена объектов» – отображение сведений на схемах. До этого режим назывался «Гибрид»);

«Карта» – показ данных картографии;

«Ландшафт» – схемы, отображающие массивные элементы поверхности, например леса или холмы (ложный 3D-рельеф);

«Просмотр улиц» – режим, позволяющий ориентироваться по панорамным фотоснимкам на фоне городских улиц;

«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.

Некоторые режимы и функции:

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.

Значение Google Maps для SEO-продвижения

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.
  1. Итог по первой главе

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.

# ГИБРИДНЫЕ ПРИЛОЖЕНИЯ ИСПОЛЬЗУЯ VUEJS И VUETIFY

Гибридные приложения являются чем-то средним между нативными и веб-приложениями (Рис.2.1.). Фактически они создаются так, чтобы выглядеть и использоваться как нативные приложения. Их также устанавливают на телефон пользователя и их можно найти в магазинах приложений [9]. Различие заключается в том, что они обязательно должны размещаться в рамках нативного приложения и созданы, чтобы работать [10] через WebView, и таким образом они могут получать доступ к информации на устройстве пользователя для больших возможностей.

Дополнительные преимущества гибридных приложений:

1. Гибридные приложения обладают наибольшей функциональностью и персонализацией для пользователя;
2. Разработчики не ограничены одной платформой, вместо этого они могут создать гибридное приложение, которое будет работать с несколькими платформами (в случае работы как нативное приложение);
3. Гибриды - хорошая опция для разработчиков, которые создают визуально насыщенные приложения, например, игры (которые не будут хорошо работать в виде веб-приложений).



Рис.2.1. Гибридное приложение.

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

* техническая оценка разработчиков;
* потребность в доступе к информации на устройстве;
* влияние скорости интернета на приложение;
* одно- или многоплатформенное ли приложение.

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.

2.1. VueJS при создании пользовательских интерфейсов

Vue - это прогрессивный фреймворк для создания пользовательских интерфейсов Web с использованием шаблона архитектуры MVVM (Model-View-ViewModel) [12]. В отличие от фреймворков-монолитов, Vue создан пригодным для постепенного внедрения. Его ядро в первую очередь решает задачи уровня представления (view), что упрощает интеграцию с другими библиотеками и существующими проектами. Он работает с уровнем представления. С другой стороны, Vue полностью подходит и для создания сложных одностраничных приложений (SPA, Single-Page Applications), если использовать его совместно с современными инструментами и дополнительными библиотеками. Двумя основными конкурентами Vue являются React и Angular. Вместе с ними Vue закрывает почти 100% рынка Web-разработки.

Поскольку Vue работает только на «уровне представления» и не используется для промежуточного программного обеспечения и бэкэнда, он может легко интегрироваться с другими проектами и библиотеками, в том числе с Vuetify [13]. Vue.js содержит широкую функциональность для уровня представлений и может использоваться для создания мощных одностраничных веб-приложений.

Функции Vue.js:

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

Основная библиотека Vue.js 2 очень маленькая (всего 17 кБ). Это гарантирует, что нагрузка на проект, реализованный с помощью Vue.js, минимальна, а сайт будет быстро загружаться.

Vue подходит для небольших проектов, которым необходимо добавить немного реактивности, представить форму с помощью AJAX, отобразить значения при вводе данных пользователем, авторизацию или другие аналогичные задачи. Vue легко масштабируется и хорошо подходит для объемных проектов, поэтому его называют прогрессивным фреймворком [14].

Vue также отлично подходит для крупных одностраничных приложений благодаря своим основным компонентам, таким как Router и Vuex. С Vue можно как использовать общедоступные API для создания приложений, так и реализовывать выполняемые сервером приложения. Но Vue лучше всего подходит для разработки решений, которые используют внешние API для обработки данных.

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

2.2. Vuetify - Framework для создания гибридных мобильных приложений

Vuetify является библиотекой #1 для Vue.js и активно разрабатывается с 2016 года [15]. Цель проекта - предоставить пользователям все, что необходимо для создания многофункциональных, красивых и интересных веб-приложений, используя спецификацию Material Design. Это возможно благодаря постоянному циклу обновлений, долгосрочной поддержке предыдущих версий, отзывчивому участию сообщества, обширной экосистеме ресурсов и приверженности качественным компонентам (Tаб.1.).

Таб.1. Сравнение UI фреимворков

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Функции | Vuetify | Bootstrap | Buefy | Element UI | Quasar Framework |
| Full Accessibility and Section 508 support | + | + | + |  |  |
| Server Side Rendering support | + | + | + | + | + |
| Long-term Support | + |  |  |  |  |
| Release cadence\*\* | Weekly | Bi-Weekly | Bi-Monthly | Bi-Weekly | Bi-Weekly |
| Treeshaking | Automatic | Manual | Manual | Manual | Automatic |
| RTL support | + | + |  | + | + |
| Premium Themes | + | + |  |  |  |
| Business and Enterprise support | + |  |  |  |  |

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 [16].

# 3. ВНЕДРЕНИЕ ГУГЛ-СЕРВИСОВ В СИСТЕМЫ, ИСПОЛЬЗУЮЩИЕ VUEJS

3.1. **Описание структуры приложения**

В ходе данной работы были рассмотрены преимущества гибридных web технологий, были изучены 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.Приложение имеет следующие функциональные возможности:

* Авторизация пользователя - имея логин и пароль, пользователи, зарегистрированные в системе, имеют возможность войти в систему;
* Список растрат, сделанных членами семьи - пользователи имеют доступ к списку месячных растрат, сделанным ими, и / или тем, которые находятся на их уровне доступа;
* Модуль отчетов, в котором есть список отчетов, доступ к которым так же определяется в зависимости от уровня доступа.

Авторизация пользователей - это первый шаг, который должны сделать пользователи. Это делается путем ввода имени пользователя и пароля, полученных от системного администратора. Если логин и / или пароль были введены неправильно, пользователю показываются соответствующие сообщения, так что он понимает, какие данные были введены неправильно.

После авторизации, пользователь перенаправляется на свою главную страницу. Он может содержать различную информацию, такую как: карточка, содержащая информацию о совершенной покупке, ее цене и, по возможности, картинке; имени пользователя, изображении профиля, почты.

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

3.2. Установка и настройка среды разработки

3.2.1. 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.

3.2.2. 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.

3.2.3. 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 инструмента npm Node.js.

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

3.2.3. 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*.

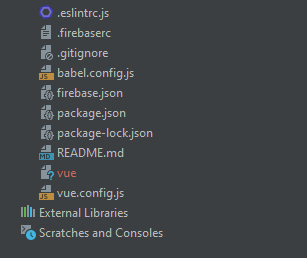
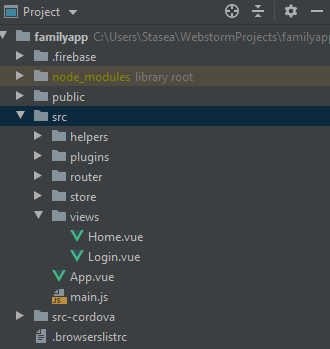


Рис.2.2. Стуктура vue vli проекта.

3.3. 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.

3.4. 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 должен подписаться на обновления всех покупок и получить текущий список.

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, оценить 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. 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;
* 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, 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 - web 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 нужных для разработки.

Данная работа предназначена для изучения гибридных web приложений, а также для ознакомления с методами и архитектурой web приложений на основе гибридных web технологии, также может быть полезна разработчикам, студентам, учителям и всем желающим ознакомиться с гибридными web приложениями.

# Библиография

1. *Поверхностный обзор javascript-enabled BaaS платформ*. Доступен: <https://habr.com/ru/post/150729/> [on-line].[цитирован 15.03.20];
2. *Firebase secures its real-time back-end service.* Доступен: <https://gigaom.com/2012/12/18/firebase-secures-its-real-time-back-end-service/>[on-line].[цитирован 20.03.20];
3. *Gerardus Blokdyk. Firebase The Ultimate Step-By-Step Guide. Emereo Pty Ltd*. ISBN 9780655321477;
4. *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];
5. *Sourav BHOI. Internet of Things: an Application Based Approach Using Arduino Platform and Firebase. Independently Published, 2018* ISBN 1983041378, 9781983041372;
6. *Обзор облачных сервисов для разработки бэкенда мобильных приложений*. Доступен: <https://habr.com/ru/company/surfstudio/blog/463435/> [on-line].[цитирован 07.04.20];
7. *OAuth 2.0 простым и понятным языком*. Доступен: <https://habr.com/ru/company/mailru/blog/115163/> [on-line].[цитирован 10.04.20];
8. Описание сервиса Google Maps <https://wiki.rookee.ru/google-maps/> [on-line].[цитирован 10.04.21];
9. *Использование сервисов Firebase при разработке мобильных приложений*. Доступен: <https://cmsmagazine.ru/journal/cases-4264/> [on-line].[цитирован 16.04.20];
10. *Нативные, веб и гибридные приложения: что выбрать?* Доступен: <https://smartum.pro/ru/blog-ru/native-web-and-hybrid-apps/> [on-line].[цитирован 26.04.20];
11. *Нативные vs. гибридные приложения*. Доступен: <https://umbrellait.com/ru/blog/native-vs-hybrid-app/> [on-line].[цитирован 04.05.20];
12. *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];
13. *Vue.js*. Доступен: <https://ru.vuejs.org/v2/guide/index.html> [on-line].[цитирован 10.05.20];
14. *Callum Macrae. Vue.js: Up and Running., O’Reilly, 2017., 219 с.,* [ISBN 9781491997246](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D1%83%D0%B6%D0%B5%D0%B1%D0%BD%D0%B0%D1%8F:%D0%98%D1%81%D1%82%D0%BE%D1%87%D0%BD%D0%B8%D0%BA%D0%B8_%D0%BA%D0%BD%D0%B8%D0%B3/9781491997246);
15. *Alex Kyriakidis, Kostas Maniatis.*[*The Majesty of Vue.js*](https://books.google.ru/books?id=Xp7cDgAAQBAJ)*., Packt Publishing Ltd, 2016., 230 с.,* [ISBN 9781787125209](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D1%83%D0%B6%D0%B5%D0%B1%D0%BD%D0%B0%D1%8F:%D0%98%D1%81%D1%82%D0%BE%D1%87%D0%BD%D0%B8%D0%BA%D0%B8_%D0%BA%D0%BD%D0%B8%D0%B3/9781787125209);
16. *VuetifyJS — строим свое приложение в стиле Material Design*. Доступен: <https://weatherless.ru/javascript/vuejs/vuetifyjs-stroim-svoe-prilozhenie-v-stile-material-design/> [on-line].[цитирован 20.05.20];
17. *Хэнчеп Эрик, Листуон Бенджамин Х88 Vue.js в действии., СПб.: Питер, 2019., 304 с.: ил., (Серия «Библиотека программиста»). Стр. 257*. ISBN 9785446110988

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