**MINISTERUL EDUCAŢIEI, CULTURII ȘI CERCETĂRII 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 APLICAȚIEI MOBILE HIBRIDE**

**TEZĂ DE LICENȚĂ**

**Autor:**

Studenta al grupei IS31Z

**Stanislava VOVOD**

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

(semnătura)

**Conducător științific:**

**Corina NEGARA**

dr., conf., univ.

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

(semnătura)

**Dumitru STOIAN**

asist. univ.

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

(semnătura)

**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

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

(semnătura)

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 MOBILE HIBRIDE

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. | 01.10.20 | 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 |

Studenta \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(semnătura)*

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

(*semnătura*)

*ADNOTARE*

Lucrare autorizată. „ Utilizarea frameworkului JS pentru crearea aplicației mobilă hibridă.”, 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, aplicație web, aplicație mobilă, aplicație mobilă hibridă.

Scopul lucrării este crearea unei aplicatiei mobilă hibridă demo pe baza unui framework JS - 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, веб-приложение, мобильное приложение, гибридное мобильное приложение.

Целью работы является разработка гибридного мобильного демонстрационного приложения, на основе JS фреймворка - Vue.js.

*ANNOTATION*

Licensed work. " Using JS framework for creating hybrid mobile application", 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, web application, mobile application, hybrid mobile application.

The scope of this work is developing a hybrid mobile demo application using Vue.js framework.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ …………………………………………………………………………………….10](#_Toc73099048)

[1. ГИБРИДНЫЕ МОБИЛЬНЫЕ ПРИЛОЖЕНИЯ ………………………………………12](#_Toc73099049)

[1.1. Определение гибридных мобильных приложений ………………………………...12](#_Toc73099050)

[1.2. Сравнение гибридных, нативных и веб приложений 14](#_Toc73099051)

[1.3. Известные примеры гибридных мобильных приложений 19](#_Toc73099052)

[1.4. Дополнительные инструменты для создания гибридного приложения 21](#_Toc73099053)

[2. ФРЕЙМВОРКИ JAVASCRIPT ………………………………………………………...25](#_Toc73099054)

[2.1. Общая характеристика фреймворков JavaScript 25](#_Toc73099055)

[2.2. Сравнение фреймворков для гибридной разработки 27](#_Toc73099056)

[2.3. Обоснование выбора фреймворков JS для разработки демонстрационного приложения …………………………………………………………………………………………34](#_Toc73099057)

[2.3.1. Фреймворк Vue.js 34](#_Toc73099058)

[2.3.2. Фреймворк Ionic 35](#_Toc73099059)

[3. ОПИСАНИЕ ДЕМОНСТРАЦИОННОГО ПРИЛОЖЕНИЯ……………………..……... 39](#_Toc73099060)

[3.1. Описание структуры приложения 39](#_Toc73099061)

[3.2. Наличие гибридных технологий в приложении и обоснование(оправдание) предпочтения их нативным 40](#_Toc73099062)

[3.4. Установка и настройка среды разработки 40](#_Toc73099063)

[3.5. Разработка приложения 41](#_Toc73099064)

[3.5.1. Создание проекта в Firebase 46](#_Toc73099065)

[3.5.2. Интеграция Firebase 46](#_Toc73099066)

[3.5.3. Использование Capacitor для определения гибридной трансляции 50](#_Toc73099067)

[ЗАКЛЮЧЕНИЕ ………………………………………………………………………………..52](#_Toc73099068)

[БИБЛИОГРАФИЯ ……………………………………………………………………………..54](#_Toc73099069)

# ВВЕДЕНИЕ

В наше время прогресс достиг невиданных высот. С появлением интернета технологии развиваются с большой скоростью и проследить за развитием каждой невозможно. Ежегодно появляются все более инновационные платформы, сервисы, приложения, позволяющие делиться информацией, обучаться, работать, веси бизнес и многое другое, не выходя из дома. Растущая популярность приложений повлияла на слияние нативных, веб и мобильных технологий, в следствии чего появились гибридные мобильные приложения.

Опираясь на динамику развития мобильных приложений, мировые тенденции обосновываются повышением потребности в мобильной разработке, возрастает скорость разработки приложений, увеличивается интерес к несложным решениям с бюджетной стоимостью разработки и удобным, понятным интерфейсам. На данный момент огромное количество компаний нуждаются в специалистах по мобильной разработке или менеджерах мобильных проектов. В этой доле компаний нативные разработчики не так востребованы, как фронтенд и мобильные разработчики, в которых молодые стартапы нуждаются больше всего.

*Цель*данной работы состоит в исследовании возможностей гибридных технологий и фреймворков JS и разработка демонстрационного гибридного мобильного приложения, используя фреймворки Ionic и Vue.js.

Для выполнения поставленной цели были спланированы следующие *задачи*:

* Изучение основ гибридных технологий;
* Исследование возможности использования гибридных технологий в качестве замены нативных технологий;
* Сравнительный анализ возможностей разных фреймворков JS;
* Представлены преимущества в пользу выбора фреймворков Ionic и Vue.js для дальнейшей разработки демонстрационного приложения;
* Создание гибридного мобильного приложения, используя гибридные технологии в сочетании с фреймворками Ionic и Vue.js.

*Ценность работы:*Данная работа полезна своей теоретической частью, так как в ней исследуются возможности гибридных технологий и инструментов для создания гибридных приложений. Практическая же часть будет полезна тем, что в ней описано предназначение и принцип работы демонстрационного приложения, а также описано назначение и процесс создания гибридного мобильного приложения в комбмнации двух наиболее оптимальных для этой задачи фреймворков JS - Vue.js и Ionic.

*Актуальность:* данная работа является актуальной, так как гибридные технологии являются универсальным решением как для начинающих предпринимателей, так и для разработчиков, ведь гибридные мобильные приложения - это набор технологий, которые позволяют сочетать в себе основные преимущества мобильной платформы будучи обычным веб приложением, что дает им большое преимущество на ряду с другими технологиями. Для предпринимателей это способ не только сэкономить на ресурсах, но и получить в итоге два приложения - веб приложение и мобильное приложение, а для разработчиков это возможность написания кода на привычном им языке без дублирования кода, низкий порог вхождения непосредственно в разработку, используя гибридные технологии, и кроме того, размещения приложения на нескольких платформах.

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

В первой главе было проведено исследование гибридных технологий, сравнение гибридных, мобильных и веб технологий, а также были приведены примеры популярных приложений и компаний, использующих гибридные технологии.

Во второй главе были описаны фреймворки JavaScript и проведен сравнительный анализ фреймворков, наиболее подходящих для гибридной разработки, таких как Vue.js, React Native, Angular, Ionic, Flutter и Xamarin. На основе данного сравнения была выбрана комбинация фреймворков Vue.js и Ionic для создания гибридного мобильного приложения. Также были исследованы вспомогательные технологии, такие как Capacitor и база данных Cloud Firestore для наделения приложения гибридными свойствами и работой с базой данных.

В третьей главе описывается демонстрационное приложение и процесс его создания, используя вышеперечисленный набор технологий. Также обосновывается наличие гибридных технологий в приложении и описывается причина предпочтения гибридных технологий нативным, после чего описываются установка и настройка среды разработки и, непосредственно, ход разработки приложения.

В ходе данной работы было разработано приложение «Ежедневник Студента», которое включает в себя набор функций для создания, просмотра и управления заданиями, которые делятся на временные промежутки (с истекшим сроком, на сегодня и предстоящие) и на состояния (готовые и неготовые). Данное гибридное приложение будет полезно как студентам, так и преподавателям, так как дает возможность грамотно спланировать свое время и не держать все задачи в голове.

Лицензионная работа изложена на 58 страницах, содержит 1 таблицу, 15 рисунков, 1 приложение и 20 библиографических ссылках.

# ГИБРИДНЫЕ МОБИЛЬНЫЕ ПРИЛОЖЕНИЯ

* 1. Определение гибридных мобильных приложений

Ноутбук, планшет, электронная книга, смартфон - это устройства, которые ежедневно используются человеком на протяжении многих лет и становятся частью их жизни, без которых многие действия становятся затруднительными, если не совсем невыполнимыми. Каждое утро многих людей начинается с будильника, ленты в социальных сетях или чтения любимой книги по дороге на работу, в течение дня устройства напоминают о важных событиях, встречах и т.д., целые системы связанных устройств даже позволяют автоматизировать бытовые процессы, которыми можно управлять всего лишь несколькими касаниями пальца в специальных приложениях. Можно сказать, что девайсы в наше время стали помощниками номер один и неотъемлемой частью повседневной жизни для работы, учебы, общения и отдыха, без которых уже нельзя представить жизнь современного человека. Однако, данные устройства, как правило, управляются своей отдельной операционной системой (iOS, Android, и т.д.). Это создает определенные неудобства для обмена информацией между ними через встроенные приложения.

Инновационным и относительно недавно появившемся решением являются гибридные мобильные приложения. Для их разработки необходимы минимальные опыт в создании встроенных мобильных приложений IOS и Android и опыт в создании веб приложений.

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 и открываются через встроенный браузер, но не установлены на устройстве, таким образом они становятся доступными только с интернет-соединением и без него не работают [1].

Гибридные приложения развертываются в собственном контейнере, который использует мобильный объект WebView (Рис.1.1). При использовании, данный объект отображает веб-контент с помощью веб-технологий (CSS, JavaScript, HTML, HTML5). Веб-контент может отображаться либо сразу после открытия приложения, либо только для определенных частей приложения. Для доступа к аппаратным функциям устройства (акселерометр, камера, контакты) добавляются собственные элементы UI для каждой платформы, чтобы пользователь удобного взаимодействовал с приложением.

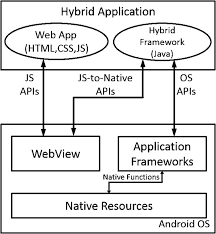


Рис.1.1. Абстрактное представление гибридного приложения.

Преимущества гибридных приложений:

* Стоимость и скорость создания приложения;
* Количество задействованных в разработке людей;
* Кроссплатформенность, позволяющая обратить внимание на продукт большего количества пользователей за счет привлечения людей с различными предпочтениями и возможностями;
* Присутствует опция автономного обновления, что позволяет пользователям не проверять наличие обновлений вручную и значительно улучшает удобство интерфейса.
* Наличие наибольшей функциональности и персонализации для пользователя;
* При разработке гибридного мобильного приложения есть возможность создавать его на нескольких платформах, а не ограничиваться одной, как в случае с нативными приложениями;
* Гибридный подход пользуется успехом у разработчиков, которые создают визуально насыщенные приложения.

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

* 1. Сравнение гибридных, нативных и веб приложений

Еще недавно компаниями были широко востребованы, в основном нативные, разработчики, но сейчас лучшей заменой для них становятся разработчики с опытом веб разработки с перспективой обучения гибридным технологиям, так как для молодых предпринимателей важно производство простых и недорогих приложений за короткое время. В условиях растущей конкуренции в приорителе у компаний - быть первыми и максимально доступными в мобильных решениях для своих потребителей.

Таковыми являются гибридные приложения. Это мобильные приложения с использованием веб-технологий и, скорее, подход к программированию, чем независимое приложение. Гибридные приложения более универсальны и чаще всего подходят для разных систем, чем веб или нативные. Его идея заключается в том, что функционал программируется на стороне сервера, а на стороне клиента остаётся только необходимый минимум. На рис.1.2. представлен принцип работы гибридного приложения.

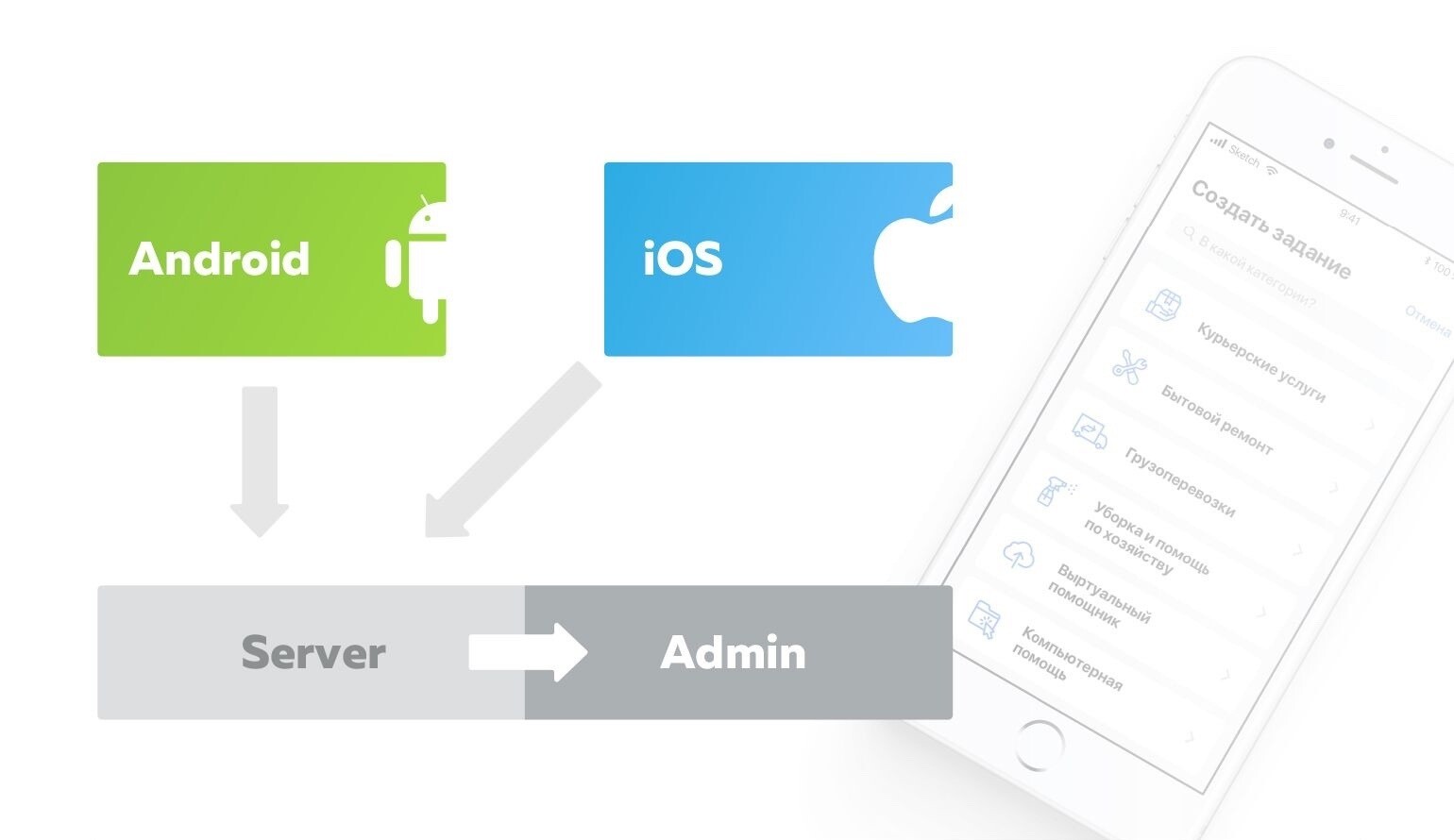


Рис.1.2. Принцип работы гибридного приложения.

Нативные приложения загружаются через магазины приложений (App Store, Google Play) 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. 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. Примером являются одиночные игры, переводчики, приложения для сопряжения с другими устройствами посредством Bluetooth.

Для демонстрации как в гибридном и нативном подходе выглядит распределение обрабатываемого функционала представлены рисунок 3 и рисунок 4. Красным выделено то, что происходит на стороне приложения для телефона, а зелёным - на стороне сервера.

Основным преимуществом гибридного подхода, исходя из рисунков, является то, что перенесенный на сервер функционал пишется один раз для обеих платформ, а красный отдельно под каждую платформу. Как следствие, это выгодно сказывается на потраченном времени, финансах, и количестве привлеченных разработчиков для гибридного подхода, и пагубно на тех же аспектах нативного подхода.

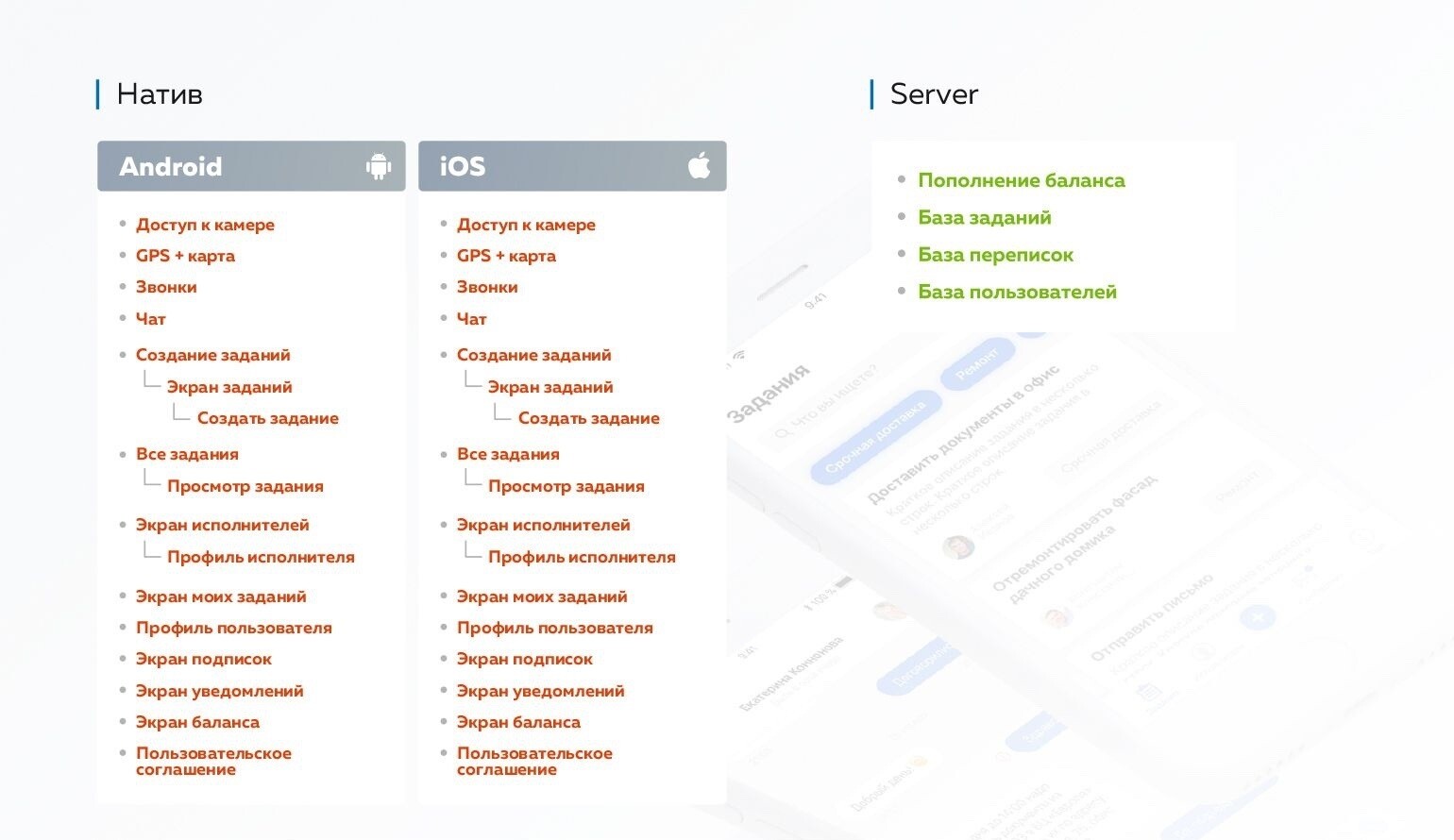


Рис.1.3. Реализация на нативных технологиях.

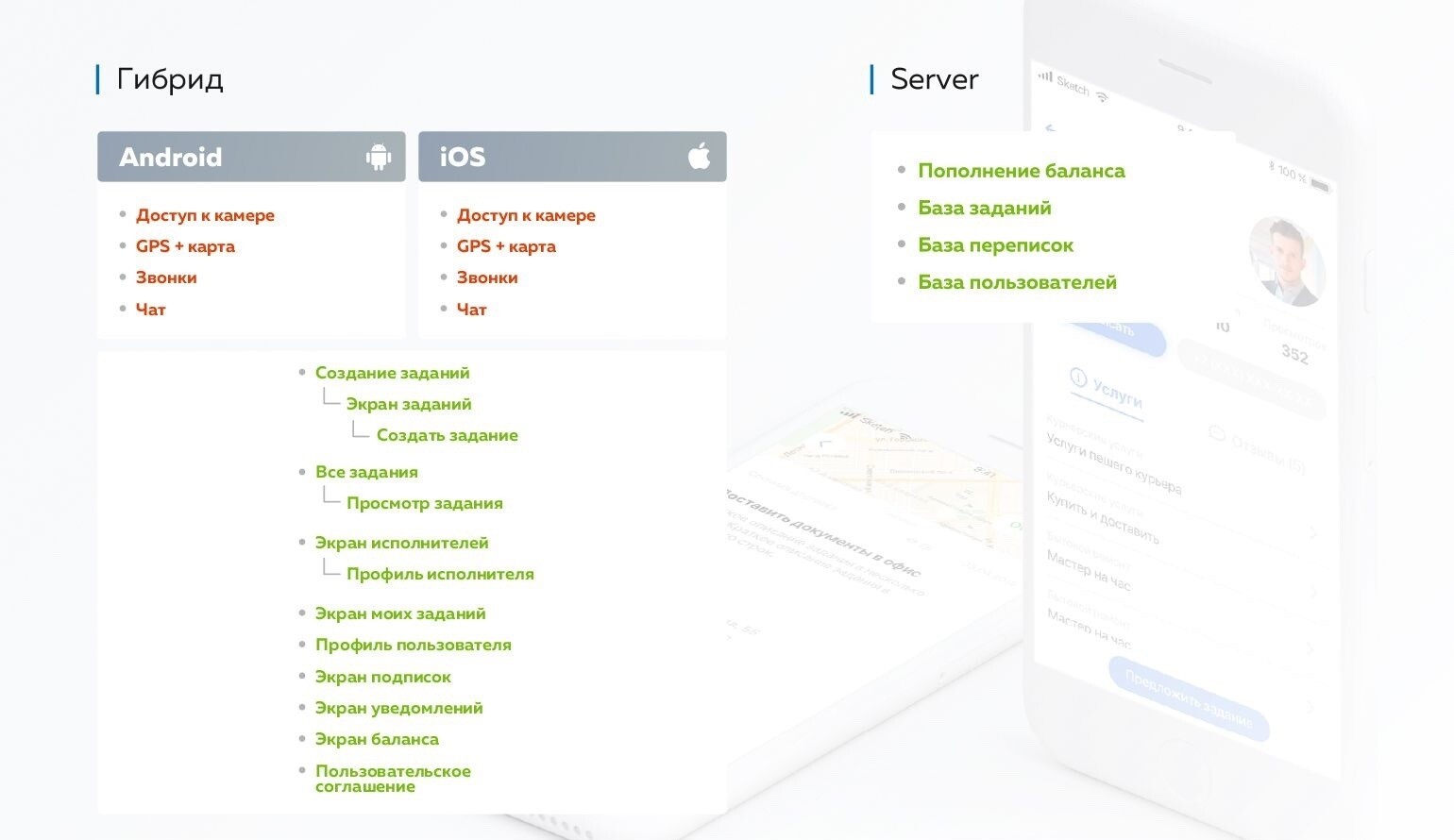


Рис.1.4. Реализация на гибридных технологиях.

Каждый многопользовательский сервис состоит из клиентских приложений и серверной части. При необходимости дополнительно интегрируется сайт, телефония, API сторонних проектов. На стороне клиентской части располагается приложение, а со стороны сервера, серверное ПО и панель администратора для управления данными.

В зависимости от требований к приложению необходимо выбрать правильный подход к разработке, просмотрев (Рис. 1.5.) так как есть риск ошибиться и переписывать проект с нуля, что повлечет за собой огромные последствия в виде потраченного времени, финансовых убытков и претензий заказчиков, которые доверились выбору компании-разработчика.

У бизнес-приложений заниженные требования к вычислительной мощности телефона и отсутствию интернета, он нужен для покупки товара либо отображения новых товаров, поэтому для них лучшим решением являются гибридные приложения. Если речь идет об играх, где в любом случае нужно интернет-соединение, нативные приложения подойдут гораздо лучше.

Еще один подход к программированию, веб-подход, предполагает создание веб-приложений или веб-страниц, адаптированных под мобильные девайсы, которые не загружаются как привычные приложения, в магазинах, но скачиваются в специальных магазинах, которые есть у некоторых современных браузерах, например, у Chrome. Веб-приложения используют для работы браузер телефона. Главным преимуществом данных приложений является их адаптивность к любой платформе и работоспособность на любом устройстве, которое достигается путем добавления некоторых надстроек.



Рис. 1.5. Сравнение нативного, гибридного и веб подхода по основным параметрам.

В таблице 1 проведено сравнение гибридного, веб и нативного подхода и представлены их преимущества и недостатки.

Таблица 1. 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 любой платформе * Запускается не из браузера в отличии от веб приложения * Возможность независимого обновления * Распространение через магазины приложений * Доступная стоимость разработки * Использование одного языка для разработки - JavaScript. * Доступ к основным данным мобильного устройства: GPS, камера, телефонная книга и т.д. * Работа в оффлайн-режиме. | * Загружается из магазина мобильных приложений (необходимо соответствовать требованиям) * Не все нативные возможности доступны * От масштабности приложения зависит его производительность * Возникают сложности при работе со сложной анимацией |

Если для работы приложения необходимо использовать мощности устройства, или для его работы очень важна скорость обработки информации (игры, социальные сети, геолокационные сервисы, сервисы обмена фотографиями и т.д.), лучше обратиться к нативной разработке. Если скорость работы не так важна, лучше выбрать гибридное приложение. А если необходимо разработать страницу-визитку, лучше обратиться к веб разработке. Часто производители делают нативные приложения платными, в то время как гибридные приложения в основном бесплатные, а веб приложения бесплатны и доступны для любой платформы, однако не работоспособны без интернета и установленного на устройстве браузера.

* 1. Известные примеры гибридных мобильных приложений

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

Приложение Wikipedia - общедоступная многоязычная универсальная интернет-энциклопедия со свободным контентом. Загружается из магазина;

* Работает в оффлайн режиме;
* Отображает раннее просмотренные статьи;
* Необходимо подключение к интернету при желании загрузить новые статьи;
* Можно настроить push-уведомления.

Wikipedia сейчас является самым крупным и наиболее популярным справочником в интернете. По объёму сведений и тематическому охвату Wikipedia считается самой полной энциклопедией из когда-либо создававшихся за всю историю человечества. Одним из ее основных достоинств как универсальной энциклопедии является возможность представления информации на родном языке пользователя. Существует как веб версия, так и мобильная.

Приложение HealthTap — онлайн консультация о вопросах здоровья с врачами из США.

* Загружается из магазина;
* Может использовать геолокацию;
* Необходимо подключение к интернету;
* Доступ к камере;
* Доступ к телефонной книге;
* Доступ к звонкам;
* Можно настроить push-уведомления.

Основная услуга, предлагаемая HealthTap, - это возможность бесплатно задавать вопросы о здоровье в веб приложении и с мобильных устройств сети врачей, имеющих лицензию в США. Вместе с этим, предоставляется возможность немедленно или по предварительной записи проконсультироваться с врачом с помощью видеоконференции, телефонного звонка или текстового чата. Медицинская информация на службе предоставляется в интерактивном режиме сетью из почти 140 000 лицензированных врачей с хорошей репутацией (лицензированные врачи с активными дисциплинарными мерами в отношении них не допускаются к сети). HealthTap сообщает, что его сеть врачей представлена в 170 странах.

Приложение BBC News – новостное приложение британской общенациональной общественной телерадиовещательной организации.

* Просмотр подгруженных новостей оффлайн;
* Live стримы;
* Использует геолокацию;
* Работа в фоновом режиме;
* Интеграция с другими сервисами;
* Доступ к смс;
* Можно настроить push-уведомления.

Приложение BBC News предоставляет новости BBC из глобальной сети журналистов. Приложение также предлагает прямую трансляцию BBC World Service Radio, социальные функции и персонализацию. Также предоставляются следующие возможности:

* Запись заметок;
* Возможность поделиться записями;
* Интеграция с приложениями Google Disk, Slack, Gmail, OutLook;
* Доступ к QR;
* Доступ к галерее;
* Доступ к проводнику;
* Доступ к аудиофаилам;
* Можно настроить push-уведомления.

Evernote - веб-сервис и набор ПО для создания и хранения заметок. В качестве заметки может выступать фрагмент форматированного текста, веб страница целиком, фотография, аудиофайл или рукописная запись. Заметки могут также содержать вложения с файлами другого типа. Заметки можно сортировать по блокнотам, присваивать им метки, редактировать и экспортировать.

Evernote поддерживает несколько компьютерных и мобильных платформ, включая macOS, iOS, Chrome OS, Android, Microsoft Windows, а также предлагает онлайн-синхронизацию и резервное копирование.

* 1. Дополнительные инструменты для создания гибридного приложения

Cloud Firestore**.** Разработка серверной части - один из самых трудных этапов создания приложения. Зачастую, при планировании разработки проекта недооценивается необходимый объем ресурсов и время создания бэкенда. 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. Длительная разработка бэкенда даже на простых проектах ведет к увеличению расходов и другим рискам: расход денег без визуально видимого результата.

Решением данной проблемы является использование облачной базы данных Firebase Cloud Firestore. Это размещенная NoSQL база данных, к которой iOS, Android и веб приложения могут получить доступ напрямую через собственные SDK. Cloud Firestore также доступен в собственных пакетах SDK для Node.js, Java, Python, Unity, C ++ и Go в дополнение к REST и RPC API. [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.

Ключевые возможности:

* Гибкость;
* Обновление в реальном времени;
* Офлайн поддержка;
* Масштабирование по нескольким базам данных.

Модель данных Cloud Firestore поддерживает гибкие иерархические структуры данных (Рис.1.6.) и позволяет хранить данные в документах, организованных в коллекции. Cloud Firestore использует синхронизацию данных для обновления данных на любом подключенном устройстве. Он также предназначен для эффективного выполнения простых одноразовых запросов на выборку. Cloud Firestore кэширует данные, которые активно использует приложение, поэтому приложение может записывать, читать, прослушивать и запрашивать данные, даже если устройство находится в автономном режиме. Когда устройство возвращается в сеть, Cloud Firestore синхронизирует любые локальные изменения с Cloud Firestore.

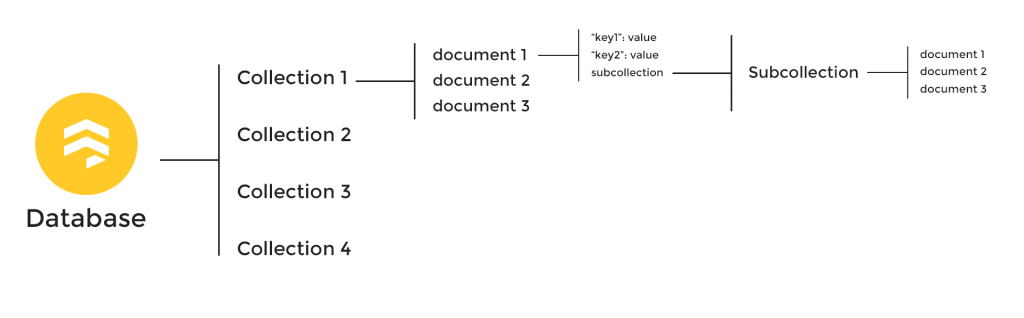


Рис.1.6. Архитектура Cloud Firestore*.*

Firebase Cloud Firestore поддерживает синхронизацию данных между клиентскими приложениями с помощью слушателей в реальном времени и предлагает автономную поддержку для мобильных устройств и интернета для возмжности создавать адаптивные приложения, которые работают независимо от задержки в сети или подключения к интернету.

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

Польза Firebase Cloud Firestore в гибридном мобильном приложении очень весомая, так как это дает возможность легко и быстро добавить полезный функционал в любое место в приложении.

**Capacitor**. В последнее время разработка современных мобильных веб-приложений и гибридных приложений претерпела значительные изменения. Появилась альтернатива Cordova или PhoneGap. Это Capacitor - среда выполнения гибридных приложений с открытым исходным кодом, созданная командой, которая также является создателем фреймворка Ionic.

Capacitor - это проект с открытым исходным кодом, который запускает современные веб-приложения на iOS, Android, Electron и Web (используя Progressive Web App технологию), предоставляя мощный и простой в использовании интерфейс для доступа к собственным SDK и собственным API на каждой платформе. В качестве альтернативы Cordova, Capacitor обеспечивает те же кроссплатформенные преимущества, но с более современным подходом к разработке приложений, использующим новейшие веб-интерфейсы API и собственные возможности платформы.

Capacitor раскрывает все встроенные функции каждой платформы с помощью согласованных межплатформенных API. Capacitor - это проект с открытым исходным кодом, который запускает веб-приложения на нескольких платформах, но без использования Electron и веба как PWA. Используя Capacitor, можно создавать одно приложение и использовать один набор API-интерфейсов независимо от платформы, на которой работает приложение, в отличие от управления несколькими API-интерфейсами для каждой целевой платформы.

Это означает, что, например, для доступа к камере в iOS / Android используется тот же код, и в Electron, и в веб-приложении. Это упрощает создание одного веб-приложения, которое запускается нативно на мобильных устройствах, десктопе и в веб версии как Progressive Web App.

Приложения, созданные на Capacitor могут включать собственные элементы управления пользовательского интерфейса и получать доступ к любому собственному SDK или API, доступным на платформе. Но, в отличие от более традиционных нативных приложений, приложения Capacitor, будут в основном работать во встроенном элементе управления WebView.

Можно выделить следующие преимущества данной среды:

* Поддержка прогрессивного веб-приложения (PWA). Capacitor - единственная собственная среда выполнения, обеспечивающая первоклассную поддержку веб-приложений и прогрессивных веб-приложений.

Это осуществляется за счет обширной коллекции интерфейсов пользовательского интерфейса для таких API-интерфейсов, Camera, а так же создание плагинов, предлагающих веб-функциональность в качестве альтернативы, так же просто, как добавление нескольких файлов в плагин, написанный самим разработчиком Capacitor будет использовать созданную разработчиком веб-реализацию только в том случае, если нативная версия недоступна, поэтому пользователи могут использовать тот же API при работе на iOS, Android, Electron и в веб-приложении.

* Простой, но мощный инструмент CLI, с управляемыми версиями для каждого приложения. Capacitor предоставляет небольшой инструмент CLI, который устанавливается локально для каждого приложения без глобальных зависимостей, которыми нужно управлять, что позволяет легко использовать разные версии Capacitor в каждом созданном приложении.
* Команды нативных разработчиков и веб-разработчиков могут совместно работать над проектами. Поскольку приложения Capacitor являются нативными приложениями, а ключевым аспектом дизайна
* В Capacitor традиционные мобильные разработчики могут использовать свои языки программирования по выбору (Swift / Objective-C на iOS, Java / Kotlin на Android) для создания пользовательского интерфейса или бизнес-логики, а затем предоставлять их веб-уровню через JavaScript-to- Capacitor. собственные API.

Кроме того, разработчики могут быстро добавить собственный код для конкретных платформ и, при необходимости, предоставить этот код веб-уровню, гарантируя, что команда приложений никогда не застрянет в реализации необходимых функций.

Ionic испольнзуется для реализации Capacitor-ом его собственных API-интерфейсов, а также предоставить расширенные корпоративные собственные функциинапример, безопасное управление токенами, зашифрованное автономное хранилище и аутентификация OpenID.

# ФРЕЙМВОРКИ JAVASCRIPT

* 1. Общая характеристика фреймворков JavaScript

Исторически сложилось, что все браузеры, мобильные и десктопные, на всех платформах понимают всего лишь три технологии: HTML, CSS и JavaScript. Каждая технологи, как и браузеры, развиваются независимо и с разной скоростью. Помимо этого, все вышеперечисленные технологии не могут по отдельности решить задачи более высокого уровня. HTML и CSS - это не языки программирования, а языки разметки: один лишь синтаксис, набор команд - конструкций и правил - для представления содержимого страницы. В них нет как таковых классов, объектов, функций, методов, присущих языкам программирования. Например, страница, созданная на чистом HTML не может выглядеть красиво, добавив CSS она будет выглядеть лучше, но не будет функциональной, а с JavaScript появится интерактивность, но, все же, для современного приложения этого мало. Чтобы наделить веб-сайт функциональностью и бизнес-логикой, приходится подключать язык программирования на стороне сервера или на стороне клиента (в браузере), а чаще всего - и там и там. Для расширения возможностей были придуманы библиотеки, которые позволяют пользоваться готовыми решениями и не дублировать часто повторяющийся код, но, так как их взаимодействие друг с другом очень нестабильно, на замену библиотекам пришли фреймворки.

Они, в отличие от библиотек, являются каркасом, на котором строится все приложение, а не используются точечно и только при необходимости, весь проект следует заданной фреймворком структуре. Таким образом контент страницы отображается одинаково во всех браузерах, вместе с этим, снизился порог вхождения разработчиков в проект, уменьшилось время на обучение и увеличилась скорость разработки.

JS фреймворки - это инструменты для построения динамических веб/мобильных/настольных приложений на языке Javascript. Как и к любым другим инструментам, разработчики прибегают к использованию JS фреймворков там, где невозможно/очень сложно/очень долго выполнять задачу обычными средствами. В подавляющем большинстве случаев, фреймворки используются для написания, Single Page Applications, где все, что происходит на сайте, происходит на одной страничке, без прямого перехода с нее. С их помощью можно разрабатывать как полноценные сайты, так и функциональные модули, различные онлайн-инструменты. Фреймворки предоставляют четкую структуру приложения и реализуются с использованием паттернов проектирования. Наиболее широко рапространены следующие паттерны: MVC (Model-View-Controller), MVP (Model-View-Presenter) и MVVM (Model-View-ViewModel).

Суть использования фреймворков заключается в том, чтобы на стороне бэкенда было только хранение, обработка и безопасность данных (извлекает их из хранилища, проверяет наличие прав доступа, передает их на фронтенд), а всё остальное выполнялось на стороне клиента, в браузере. Используя только набор определенных данных и шаблон - набор инструкций по превращению данных в верстку фреймворк будет подставлять данные в шаблон, реализовывать бизнес-логику и манипулировать страницей. Такие фреймворки называются реактивными, потому что в них состояние интерфейса автоматически реагирует на изменение данных. Реактивное программирование - вариант многопоточного, при котором вместо системы «запрос - ожидание ответа - получение ответа - обработка» работает принцип «запрос - ответ - обработка ответа». Примеры: React, Angular, Vue и еще десятки менее известных.

Итак, на бэкенде можно использовать любой язык программирования, добывать им данные, упаковывать их в JSON, XML или что угодно другое машинно-читаемое и отдавать на фронт, а на фронтенде JS фреймворк делает всю чистовую работу: рисует контролы, анимирует их, проверяет данные, представляет их в соответствии с локальными настройками пользователя и реализует бизнес-логику. Между браузером и сервером передаются только данные и иногда - новые части шаблона и правила. Реактивные фреймворки незаменимы для реализации концепции Single Page Application, когда всё происходит на одном экране.

Для реализации полноценного веб приложения также необходимы надстройки. Некоторые надстройки реализованы для нескольких фреймворков, Onsen, например, или тот же KendoUI. Если клиент собирается разрабатывать мобильное приложение современным способом (то есть в виде HTML5 SPA / HybridApp), но пока не знает, будет ли он использовать React, Angular или Vue, то нужна одна из таких универсальных надстроек.

Итак, вот как теперь выглядит веб-проект или мобильное приложение:

* бэкенд на любом языке программирования;
* фронтенд, обычно состоящий из JS фреймворка (React, Angular, Vue) и надстроек.

Преимущества построения приложения на JS фреймворке:

* можно легко реализовать SPA (Single Page Application);
* использование JS фреймворка обязывает нас иметь структуру приложения (скажем решительное «нет» спагетти-коду);
* кода становится заметно меньше, и он чище, что позитивно отражается на скорости разработки, а также поддержке и устранении ошибок в коде приложения;
* наличие структуры подразумевает модульность приложения, а это дает возможность проще работать над приложением нескольким разработчикам одновременно;
* следующее преимущество больше вытекает из использования самого javascript, но значительно усиливается при использовании фреймворка: возможность быстро создать мобильное и/или настольное кроссплатформенное приложение из веб-версии с помощию систем типа PhoneGap или Apache Cordova.

Из существенных недостатков можно выделить только временно неполную поддержку поисковыми системами, но эта задача редко совпадает с задачей по реализации SPA (Single Page Application), тем более, что ведущие поисковые системы (как минимум, Google), уже практически полностью решили эту проблему.

Приложений на JS фреймворках уже выпущенно очень много и этот сегмент только набирает темп.

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 [3].

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

* 1. Сравнение фреймворков для гибридной разработки

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 [7]. Для сравнения приведены следующие фреймворки: Vue.js, React Native, Angular, Ionic, Flutter и Xamarin.

Vue.js – 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.js:

* привязка данных;
* производительный (есть Virtual DOM);
* легкость в изучении;
* адаптивность;
* подробная документация;
* масштабирование;
* оптимизация HTML-блоков;
* миниатюрный размер;
* 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:

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

React Native – прогрессивный фреймворк, разработанный корпорацией Facebook, который обеспечивает традиционные и современные методы разработки гибридных мобильных приложений. 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:

* компонентно-ориентированный;
* декларативный;
* производительный (благодаря React Virtual DOM);
* серверный рендеринг;
* наличие Redux;
* поддержка PWA;
* JSX.

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

Недостатки React:

* Использование шаблонов вместе с JSX для начинающих разработчиков может показаться не понятным и странным;
* Не прямолинейность выбора этапов развития проектов. Для разработки на React нужен опытный руководитель.
* Эта прогрессивная технология обеспечивает традиционные и современные методы разработки гибридных мобильных приложений.

В течение нескольких лет работы фреймворк использовался во всемирно известных приложениях, таких как Skype, Instagram, Airbnb, Walmart, SoundCloud Pulse, UberEats, Tesla, Tencent QQ, Baidu Mobile и многих других.

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.

Особенности фреймворка Angular:

* универсальность;
* нативность;
* Высокая сложность в виду крайне обширного функционала;
* используется с Typescript;
* интеллектуальное автозаполнение HTML-компонента;
* архитектура рассчитана на крупные проекты;
* 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.

Ionic - это кроссплатфоренный фреймворк который позволяет создавать мобильные приложения для телефонов iOS, Windows и Android с использованием одной и той же кодовой базы. Это также кроссплатформенный инструмент для мобильной разработки. Ionic также фреймворк для разработки мобильных приложений HTML5, предназначенная для создания гибридных мобильных приложений. Ionic также фронт-энд фреймворк пользовательского интерфейса, который способен изменять внешний вид приложения и взаимодействовать с пользовательским интерфейсом по аналогии с «Bootstrap for Native», но с поддержкой широкого спектра распространенных нативных мобильных компонентов, красивой анимации и красивого дизайна.

Этот фреймворк позволяет создавать мобильные приложения с помощью веб-приложений и языков, таких как HTML , CSS , Javascript , Angular и Typescript . Ionic имеет набор компонентов, обеспечивающих функциональность мобильной платформы. За исключением функции, Ionic обеспечивает другое поведение. Ionic обладает эффективной производительностью с минимальными манипуляциями с DOM. Он поставляется с интерфейсом командной строки, который позволяет разрабатывать приложение и экономит усилия по написанию кода.

Преимущества Ionic:

* Это обеспечивает быструю разработку по сравнению с нативными приложениями iOS / Android.
* Возможность разрабатывать приложения в браузере.
* Позволяет создавать приложения для разных операционных систем с одной и той же кодовой базой, за исключением поддержки некоторых плагинов.
* Базовые знания Vue.js, HTML, CSS, Javascript и Typescript.
* Множество плагинов и компонентов многократного использования для создания более интерактивных мобильных приложений.
* Упрощенное тестирование модулей и компонентов.
* Быстрый вывод на рынок для различных платформ.
* Предлагает несколько элементов пользовательского интерфейса и возможности для быстрого прототипирования.

Ниже приведен список недостатков, которые могут возникнуть при работе с фреймворком Ionic:

* Наличие некоторых собственных плагинов, которые могут быть нестабильными и конфликтуют друг с другом.
* С помощью Ionic отладка может быть в некоторой степени сложной и требует больше времени для решения проблемы. Это затрудняет определение источника ошибки, а иногда сообщение об ошибке может быть неясным.
* Сборки могут вылетать без какой-либо причины, например, при повреждении исходной папки.
* При создании гибридных приложений, есть риск столкнуться с проблемами безопасности, и код может быть уязвим для хакерских атак.
* Использование плагинов для доступа к встроенным функциям.
* При создании сложных и тяжелых мобильных приложений могут возникнуть проблемы с производительностью.

В отличие от адаптивного фреймворка, Ionic поставляется с элементами мобильного пользовательского интерфейса с нативным стилем и макетами, которые получили бы с собственным SDK на iOS или Android, но на самом деле раньше не существовали в Интернете. Ionic также предлагает несколько смелых, но эффективных способов создания мобильных приложений, которые затмевают существующие фреймворки разработки HTML5.

Flutter - комплект средств разработки и фреймворк с открытым исходным кодом для создания мобильных приложений под Android и iOS, а также веб-приложений с использованием языка программирования Dart, разработанный и развиваемый корпорацией Google. Dart - замечательный язык программирования, который предлагает множество преимуществ и основан на C / C ++ и Java. Несмотря на то, что он является новым, вскоре ожидается, что этот язык покорит отрасль. Dart - одна из причин, почему сейчас многие разработчики предпочитают разработку приложений на Flutter.

При сравнении производительности приложений именно Flutter берет верх над своими конкурентами. Поскольку у него есть преимущества Dart и нет моста JavaScript для запуска взаимодействия с собственными компонентами устройства, скорость, которую он предлагает, потрясающая.

По критерию графического UI, Flutter предоставляет лучший пользовательский интерфейс. Без сомнения, Ionic и Xamarin предоставляют кроссплатформенные приложения, но их эффективность и производительность не могут сравниться с Flutter и React Native. Они застревают и не реагируют на запросы, если приложение тяжелое и используется больше собственных компонентов пользовательского интерфейса.

На мировом рынке Flutter - это относительно новый фреймворк, однако он активно рекламируется Google, что показывает, что они хотят сделать это главным в мобильном мире. Хотя в нем все еще есть некоторые проблемы, пользоваться им и переходить от идеи к прототипу и приложению можно без особого труда.

Xamarin — это фреймворк для кроссплатформенной разработки мобильных приложений. Идея фреймворка состоит в написании кода на привычном языке с применением всех привычных языковых осоенностей, например, LINQ, лямбда-выражений, Generic`ов и async`ов. При этом остается полный доступ ко всем возможностям SDK платформы и родному механизму создания UI, получая на выходе приложение, которое, ничем не отличается от нативных и не уступает им в производительности.

Фреймворк состоит из нескольких основных частей:

* Xamarin.IOS — библиотека классов для C#, предоставляющая разработчику доступ к iOS SDK;
* Xamarin.Android — библиотека классов для C#, предоставляющая разработчику доступ к Android SDK;
* Компиляторы для iOS и Android;
* IDE Xamarin Studio;
* Плагин для Visual Studio.

Xamarin использует C# со средой .net для разработки приложений. Все, что можно сделать с помощью нативных языков, моно написать на C# с помощью Xamarin. Однако разработчики не могут использовать собственные библиотеки с открытым исходным кодом, доступные для iOS и Android с Xamarin, существует множество доступных библиотек .net, которые удовлетворяют эту потребность. Фреймворк поддерживается на таких платформах, как Android 4.0.3+, iOS 8+ и Window 10. На Xamarin разработаны популярные приложения, такие как Olo, Всемирный банк, Storyo.

Производительность Xamarin незначительно отличается от нативной. В Xamarin есть два способа создания мобильных приложений: Xamarin.Android/Xamarin.iOS и Xamarin.Forms.

Приложения Xamarin.Android/iOS работают как нативные, потому что их кроссплатформенные возможности сосредоточены в основном на совместном использовании бизнес-логики, а не кодовой базы. Это помогает достичь нативной производительности, которая невозможна с решениями, интерпретирующими код во время выполнения, однако подход Xamarin.Forms основан на широком совместном использовании кода с менее зависящим от платформы поведением. Это в целом снижает производительность кода при выполнении множества операций по сравнению с различными платформами.

Xamarin позволяет создавать пользовательский интерфейс двумя разными способами: с помощью Xamarin.Android/iOS или Xamarin.Forms. Первый отнимает много времени, но гарантирует естественный внешний вид с точки зрения UX.

С Xamarin.Forms можно значительно ускорить процесс разработки и сэкономить много ресурсов, но за счет локального внешнего вида. Это может быть хорошим решением для внутренних и корпоративных предприятий, где пользовательский интерфейс не так важен, как в общедоступных приложениях.

Xamarin - довольно популярный фреймворк. И все же Microsoft прилагает много усилий для развития сообщества Xamarin. Разработчики, работающие внутри экосистемы Microsoft, могут без особого напряжения начать работу с нововведением, поскольку оно активно поддерживается.

Xamarin не требует переключения между средами разработки: все приложения Xamarin разрабатываются в Visual Studio. Обычно до 96 процентов исходного кода можно повторно использовать с Xamarin.Forms, что ускоряет процесс разработки.

Xamarin требует от разработчиков установки IDE, которая предоставляется по подписке. Однако он также предлагает бесплатную версию для студентов и некоммерческих предприятий с количеством пользователей н более 5 человек.

Изучив особенности выше представленных фреймворков был осуществлен их сравнительный анализ по основным критериям, который представлен в приложении 1.

* 1. Обоснование выбора фреймворков JS для разработки демонстрационного приложения
     1. Фреймворк Vue.js

Vue.js - это современный фреймворк для разработки пользовательских веб интерфейсов с использованием паттерна MVVM (Model-View-ViewModel). Ядро фреймворка хорошо справляется с задачами уровня представления (view), что облегчает интеграцию с различными библиотеками и фреймворками, в число которых входит Ionic. На Vue.js чаще всего создаются сложные одностраничные приложения SPA. Основными конкурентами Vue.js являются React и Angular, которые в совокупности заполняют 100% рынка веб 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 [4].

Vue.js 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.js 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.js также можно создавать фронтенд 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 все же обладает своими особенностями, отличающими его от Angular и React. Одной из неотъемлемых 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.js - 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.

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-приложений, которые используют крупные компании, такие как Red Bull, CNN и Product Hunt. Именно благодаря фреймворку Vue.js 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.js в первую очередь решает задачи уровня представления (view), что упрощает интеграцию с другими библиотеками и существующими проектами.

Vue.js 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 Ionic

Ionic - это платформа SDK с открытым исходным кодом, которая позволяет создавать мобильные приложения для телефонов iOS, Windows и Android с использованием одной и той же кодовой базы. Это также кроссплатформенный инструмент для мобильной разработки.

Этот инструмент позволяет создавать гибридные мобильные приложения. Эта платформа позволяет создавать мобильные приложения с помощью веб-приложений и языков, таких как HTML, CSS, Javascript, Angular и Typescript. Ionic имеет набор компонентов, обеспечивающих функциональность мобильной платформы. За исключением функции, Ionic обеспечивает другое поведение. Ionic обладает эффективной производительностью с минимальными манипуляциями с DOM. Он поставляется с интерфейсом командной строки, который позволяет разрабатывать приложение и экономит усилия по написанию кода.

Цель использования Ionic - создать связь между веб-приложениями Vue.js и гибридными мобильными приложениями. Он позволяет использовать функции Vue.js вместе с библиотекой Cordova для работы с интерактивными приложениями. Он предлагает использовать собственные компоненты пользовательского интерфейса, совместимые с любой операционной системой. Ionic создан для разработки мультиплатформенных приложений.

Фреймворк Ionic использует плагины Cordova и поддерживает множество надстроек для разработки. Он предоставляет плагин Cordova, который помогает дизайнерам предоставлять доступ к различным компонентам. С помощью фреймворка Ionic разработчики смогут создавать надежные и простые конструкции. Ionic предоставляет несколько настраиваемых тем и компонентов, которые обеспечат отличный пользовательский интерфейс и привлекут больше людей к приложению.

Ionic показывает высокую производительность и предоставляет удобную среду для создания приложений, которые будут работать на мобильных устройствах. Вы можете использовать любой компонент для увеличения производительности мобильных устройств.

Ionic использует расширенную веб-технологию, которая соответствует требованиям завтрашнего дня. Это означает, что всякий раз, когда вы хотите внести изменения в свой код или настроить любой используемый компонент, это можно сделать легко и без усилий с использованием веб-технологий.

Есть много веских причин, по которым вы можете предпочесть платформу Ionic для разработки.

Ionic упрощает тестирование мобильных приложений. Это позволяет вам протестировать его в браузере или через тестирование стимулятора для всех ОС. Это позволяет вам выполнять удаленную отладку и тестирование приложений на реальной платформе, чтобы получить фактическую работу приложения.

Преимущества Ionic:

* Это обеспечивает быструю разработку по сравнению с нативными приложениями iOS / Android.
* Возможность разрабатывать приложения в браузере.
* Позволяет создавать приложения для разных операционных систем с одной и той же кодовой базой, за исключением поддержки некоторых плагинов.
* Базовые знания Vue.js, HTML, CSS, Javascript и Typescript.
* Множество плагинов и компонентов многократного использования для создания более интерактивных мобильных приложений.
* Упрощенное тестирование модулей и компонентов.
* Быстрый вывод на рынок для различных платформ.
* Предлагает несколько элементов пользовательского интерфейса и возможности для быстрого прототипирования.

Ниже приведен список недостатков, которые могут возникнуть при работе с фреймворком Ionic:

* Наличие некоторых собственных плагинов, которые могут быть нестабильными и конфликтуют друг с другом.
* С помощью Ionic отладка может быть в некоторой степени сложной и требует больше времени для решения проблемы. Это затрудняет определение источника ошибки, а иногда сообщение об ошибке может быть неясным.
* Сборки могут вылетать без какой-либо причины, например, при повреждении исходной папки.
* При создании гибридных приложений, есть риск столкнуться с проблемами безопасности, и код может быть уязвим для хакерских атак.
* Использование плагинов для доступа к встроенным функциям.
* При создании сложных и тяжелых мобильных приложений могут возникнуть проблемы с производительностью.

Ionic - это мощная технология, которая позволяет создавать веб-приложения и современные мобильные гибридные приложения намного быстрее и без усилий по сравнению с другими нативными методами разработки приложений.

Этот фреймворк предлагает множество функций для разработки мобильных приложений, поддерживающих различные платформы. Это позволяет команде разработчиков сэкономить время, усилия и затраты на разработку.

Чем фреймворк Ionic отличается от других платформ мобильной разработки?

Доступно несколько гибридных платформ, но выбор правильной важен для создания интерактивного приложения. Но Ionic появился и стал популярным для большинства стартапов и даже для крупного бизнеса. Некоторые из доступных альтернатив - Xamarin, Titanium, PhoneGap и многие другие. Каждый из них имеет множество функций, но ионный был предпочтен по следующим причинам:

Ionic обеспечивает высокую скорость, совместимость и поддержку со всеми новейшими устройствами и платформами позволяет создавать высокооптимизированные и надежные приложения.

Пользовательский интерфейс имеет большое значение для привлечения клиентов. Но с помощью структуры Ionic предприятия могут создавать высококачественные интерактивные мобильные приложения с меньшими усилиями. Ionic предлагает множество предопределенных компонентов, которые можно использовать повторно во многих частях для повышения интерактивности. Он предоставляет интерфейс командной строки, который позволит создавать приложения с помощью одной команды с множеством плагинов.

# ОПИСАНИЕ ДЕМОНСТРАЦИОННОГО ПРИЛОЖЕНИЯ

**3.1. Описание структуры приложения**

В ходе данной работы было создано гибридное мобильное приложение «Ежедневник Студента», которое позволяет организовать свой учебный процесс не только студентам, но и преподавателям. Простой пользовательский интерфейс дает возможность с легкостью ориентироваться в приложении, записывать новые задания и не забывать про старые, так как все они отображаются на странице предмета. Данное приложение состоит из списка предметов, в каждом из которых хранятся записи о заданиях, чей срок истек, заданиях, которые необходимо сделать сегодня и заданиях, которые предстоит сделать, а также есть отдельный список, в которых хранятся уже выполненные задания. Для перемещения заданий в последнюю категорию, необходимо просто сдвинуть запись о задании влево. Тогда появится чекбокс, после нажатия на который задание перемещается в категорию «готово». В случае, если задание было записано неверно или оно устарело и больше не нужно, есть возможность сдвинуть его вправо, тогда появится иконка корзины, после нажатия на которую запись удалится. Приложение имеет следующие функциональные возможности:

* Добавление новой записи на страницу предмета;
* Просмотр записей всех предметов на стрнице «Все предметы»;
* Просмотри записей о конкретном предмете;
* Перенос записи в категорию «готово» и обратно при необходимости;
* Удаление неактуальных записей.

Для добавление новой записи необходимо выполнить следующие шаги:

* Нажать на иконку «+» в правой нижней части приложения;
* Ввести заголовок задания (описать его основную суть);
* Нажать на иконку календаря и выбрать желаемую дату исполнения задания;
* Ввести дополнительное описание задания в следующем поле;
* Нажать на следующий селект и выбрать к какому предмету относится данное задание;
* Нажать на кнопку “Добавить”.

Данное приложение будет полезно для тех, кто старается грамотно организовывать учебный процесс и ценит свое время. В демонстрационном приложении «Дневник студента» были имплементированы такие технологии, как Firebase Realtime Database, Ionic, Capacitor, Vue.js.

* 1. Наличие гибридных технологий в приложении и обоснование предпочтения их нативным

Из предыдущих глав было установлено, что в каждом индивидуальном случае, в зависимости от имеющихся ресурсов, необходимо правильно выбрать подход к написанию приложения. В разработке демонстрационного приложения были выбран гибридный подход, так как в этом частном случае необходимо было найти решение, которое позволяет снизить порог вхождения в сам процесс разработки гибридного приложения, использовать как можно больше технологий для увеличения функциональности приложения, а также запускать его как в десктоп версии, так и в мобильной версии на разных платформах.

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 гибридное приложение также полагается на платформу Ionic, 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.

В демонстрационном приложении «Ежедневник Студента» гармонично сочетаются веб и мобильные технологии, так как оно запускается как в браузере, так и на мобильных устройствах как приложение. Приложение выглядит как нативное и ничем не отличается визуально, при этом запускается как веб-приложение в браузере и отображает контент с помощью WebView. При написании приложения использовались фреймворки Vue.js и Ionic, а также технология Capacitor что позволило сделать приложение кроссплатформенным, или иначе, дало возможность запускать приложение как на платформе Android, так и на IOS и других платформах. Одним из самых важных достоинств данного гибридного приложения является возможность изменения кода в редакторе единажды и, после сборки проекта, получить результат одновременно в веб и мобильной версии. Этот аспект позволяет сосредоточиться на расширении возможностей приложения, вместо того, чтобы дублировать код, создавая отдельный проект под каждую плоатформу, 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 размещено в нативной оболочке, 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 на обучение время.

* 1. Установка и настройка среды разработки

**Установка среды разработки**. Для разработки демонстрационного гибридного приложения была выбрана IDE WebStorm от компании JetBrains. Установка данной среды разработки заключается в переходе на официальную страницу JetBrains и загрузки установочного фаила community версии. Данная IDE была выбрала из-за обширный возможностей, таких как автозаполнение, дебага, подсветки кода и возможных ошибок, встроенного терминала, простой интеграции с git, рефактором кода. Вторая IDE для создания мобильной версии приложения является AndroidStudio, который предоставляет возможность использовать Capacitor для сборки приложения и удобный эмулятор для просмотра и проверки работоспособности приложения на любом устройстве, которое также можно выбрать и установить в качестве основного.

**Node Package Manager (NPM).** NPM необходим для подготовки к разработке больших приложений на Vue.js. Эта технология прекрасно работает с инструментамом сборки Webpack и другими. Vue.js также имеет совместимые инструменты для использования однофайловых компонентов. Команда установки npm: npm i позволяет сказать пакетный менеджер на устройство. Команда npm install vue устанавливает последнюю стабильную версию Vue.js. Все пакеты хранятся в папке node modules и, при необходимости, скачиваются дополнительные пакеты командами npm во встроенном терминале IDE.

**Инструменты командной строки (CLI).** Vue.js предоставляет инструкции (CLI) для быстрого создания инфраструктуры одностраничных приложений. Всего за несколько минут собирается рабочая конфигурация с перезагрузкой модулей, анализом ошибок при настройке производственной сборки. CLI - это инструмент, который имплементируется в проект с помощью команды установки: npm install vue-cli.

* 1. Разработка приложения

Первые шаги для создания демонстрационного приложения были: Установка Ionic и создания Ionic-приложения. Для установки Ionic’a глобально была использована команда npm install –g @ionic/cli. После недолгого ожидания установки Ionic’a и его cli был созданн Ionic-проект командой ionic start diplom-app tabs –type vue. Для запуска приложения по аддресу localhost:8800 используется команда ionic serve.

Следующим шагом для разработки было создание страниц (views) для отображения контента (Рис. 1.7.). В папке views были созданы страницы:

* Analizanumerica.vue,
* ArhitecturaCalculatorului.vue,
* ArchitecturaSetei.vue;
* Etica.vue, PADM.vue;
* Home.vue, SGBD.vue;
* SistemeInteligente.vue;
* VsePredmeti.vue.

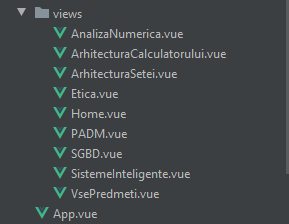


Рис. 1.7. Структура папки views

Для наглядности отображения в каждый компонент была импортирована библиотека IonIcons которая будет отображать иконки по каждому из предметов на каждой странице.

После создания компонентов каждого предмета была разработана структуры страницы (Рис. 1.8.), которая является главной страницей демонстрационного приложения в файле Home.vue. Для этого была использована библиотека tailwindcss. Структура страницы представляет из себя набор карточек для каждого изучаемого предмета, у которого есть: Уникальная для предмета иконка, Название изучаемого предмета и счетчик количества заданий для каждого предмета. Так же была добавленна кнопка с иконкой «+» для вызова модального окна добавления нового задания:

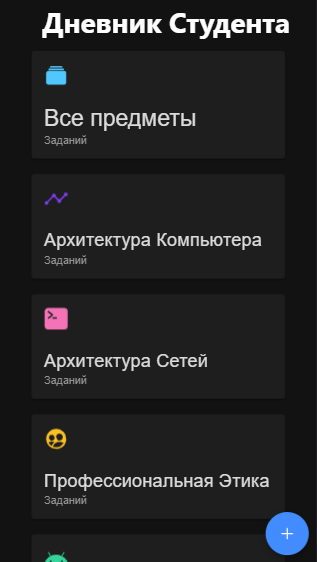


Рис. 1.8. Главная страница приложения.

Далее в папке router в файле index.ts были импортированы все созданные компоненты страниц и был создан массив роутов для создания роутера который бы производил переходы от одной страницы к другой без её перезагрузки.

Далее была создана структура страницы VsePredmeti.vue (Рис. 1.9.), в которой отображаются все задачи по всем предметам. Были созданы следующие компоненты страницы: Уникальная иконка и заголовок «Все предметы», а так же блоки, которые отображают текущее состояние задания: Истек срок: На сегодня:, Предстоящие: и Готово: с счетчиком заданий по каждой категории:

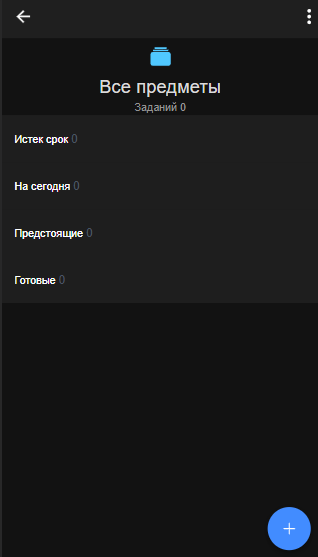


Рис. 1.9. Страница с задачами из всех категорий.

По аналогии со страницей home были стилизированы оставшиеся не стилизированные страницы: Analiza numerica.vue, ArhitecturaCalculatorului.vue, ArchitecturaSetei.vue, Etica.vue, PADM.vue, SGBD.vue, SistemeInteligente.vue и VsePredmeti.vue. У каждой из этих страниц так же присутствуют блоки: Уникальная иконка и заголовок с названием каждого предмета, а так же блоки с названиями Истек срок:, На сегодня:, Предстоящие: и Готово: с счетчиком заданий по каждой категории:

Далее для создания модального окна, в котором добавляется новое задание (Рис.1.10.), была установлена библиотека vee-validate с помощью команды npm install vee-validate@next. Затем создается компонент в папке components – Zadanie.vue, в котором создается структура и форма со следующими полями: Название задания, дата исполнения, описание задания и селект, в котором после нажатия, выбирается предмет, к которому относится данное задание и кнопка добавления задания.

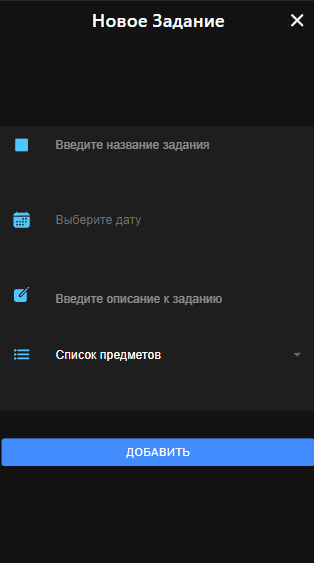


Рис.1.10. Модальное окно добавления нового задания.

После этого прописывается логика срабатывания валидаторов, импортированных с помощью библиотеки vee-validate, которая проверяет наличие данных в поле:

*setup(){*

*const task = ref('');*

*const dueDate = ref('');*

*const note = ref('');*

*const category = ref('');*

*const isRequired = (value) => {*

*if (!value) {*

*return 'Это поле обязательное!';*

*}*

*return true;*

*}*

В случае, если данные в поле отсутствуют, после нажатия на кнопку «добавить», валидатор срабатывает и под полем появляется подсказка о том, что его необходимо заполнить.

* + 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, 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 Cloud Firestore.

* + 1. Интеграция Firebase

После настройки Firebase, следует его интеграция в приложение. Для этого создается фаил firebase.ts в папке src проекта, в который записываются ключи, сгенерированные в Cloud Firestore.

*import firebase from "firebase";*

*const firebaseConfig = {*

*apiKey: "<сгенерированный ключ>",*

*authDomain: "diplomapp-775232.firebaseapp.com",*

*projectId: "diplomapp-775232",*

*storageBucket: "diplomapp-775232.appspot.com",*

*messagingSenderId: "601953471712",*

*appId: "1: 601953471712:web:4fb899236a83341566f4c9"*

*};*

*export default firebase.initializeApp(firebaseConfig);*

Далее в компонент Zadanie.vue, который отвечает за добавление нового задания, импортируем подключение к приложению firebase созданную ранее.

После подключения к firebase необходимо написать логику работы с cloud firestore. Для этого в файле Zadanie.vue необходимо было создать метод записи данных в firebase:

*function addTask() {*

*db.collection('tasks')*

*.add({*

*task: task.value,*

*note: note.value,*

*dueDate: dueDate.value,*

*category: category.value,*

*done: false*

*})*

*.then(() => {*

*task.value = "";*

*dueDate.value = "";*

*note.value = "";*

*category.value = "";*

*this.$emit('close-modal');*

*console.log('Document successfully written !'); })*

*.catch((error) => {console.log("Error writing document: ",error); }) }*

*return{*

*isRequired,task,dueDate,note,category,addTask,*

*close,notifications,document,grid}*

После создания метода он был прописан в форме создания нового задания, а все созданные в методе поля были связанны с полями в html-структуре страницы.

Затем необходимо установить библиотеку Vuex командой npm install vuex@next, после чего в корневой папке проекта создать папку store с фаилом index.ts, в котором описывается логика получения данных из базы данных firebase. В фаиле index.ts импортируется подключение из фаила firebase.ts и создается логика для получения данных из Cloud Firestore. За это отвечает метод gesTasks.

*getTasks: (state) => {*

*state.tasks = [];*

*db.collection('tasks')*

*.onSnapshot((querySnapshot: any ) => {*

*state.tasks = [];*

*querySnapshot.forEach((doc: any) => {*

*state.tasks.push({*

*id: doc.id,*

*task: doc.data().task,*

*dueDate: new Date(doc.data().dueDate).toLocaleDateString('en-US',state.options),*

*category: doc.data().category,*

*note: doc.data().note,*

*done: doc.data().done*

*}) })})}*

Для реализации функционала фильтрации заданий по состояниям: Истек срок, На сегодня, Предстоящие и Готовые были реализованы 4 метода под каждое состояния для управления заданиями и отображения в соответствующих блоках приложения, после чего в представлениях прописвается логика вывода заданий. Помимо функционала для фильтрации заданий по их состоянию, присутствует фильтрация по предметам, к которым они относятся. Вся логика по фильтрации написана в фаиле index.ts. За вывод количества заданий в состониях и предметах отвечает функция:

*lengthTasksByLesson: (state) => {*

*return (category: any) => state.tasks.filter((item: any) => {*

*return item.lesson == lesson;*

*}).length;}*

Этот пример отвечает за вывод количества заданий для предметов. По аналогии с ним были написаны методы для каждого предмета и для каждого состояния индивидуально.

Для реализации функций определения заданий как выполненное, невыполненное или удаленное также были реализованы 3 метода и привязаны к соответствующим кнопкам в пользовательском интерфейсе на каждой странице, где выводятся задания.

На данном этапе веб-приложение готово и для демонстрации заполнены задания для одного из предметов(Рис.1.11.):

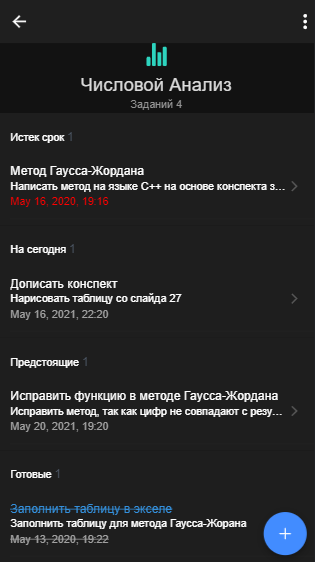


Рис.1.11. Пример отображения заданий в одной из категорий.

Все задания можно увидеть в разделе «Все Предметы» (Рис.1.12).

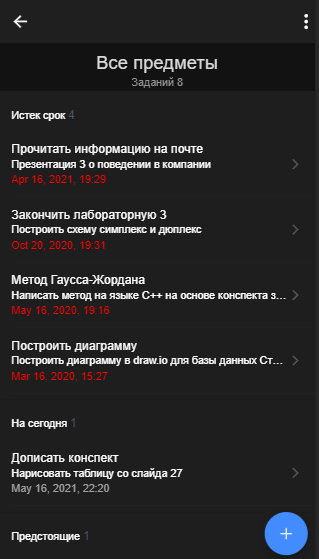


Рис.1.12. Пример отображения всех заданий на странице «Все предметы».

Здесь отображаются задания по всем предметам. Приложение готово к использованию в браузере, но пока еще не готово для установки на мобильные устройства. Для этого выполняются шаги, описанные в следующей главе.

* + 1. Использование Capacitor для определения гибридной трансляции

Средство командной строки Capacitor распространяется как пакет npm в готовом к использованию формате. Нет необходимости для его компиляции из исходного кода. Чтобы установить инструмент командной строки capacitor, необходимо выполнить следующие действия:

* Для сборки приложения под мобильные платформы используется Capacitor и команда для его установки: npm install @capacitor/cli @capacitor/core
* Следующая команда необходима для сборки приложения на Ionic: ionic build prod
* Далее добавляется android платформа в проект командой npx cap add android
* После этого используется команда для открытия IDE Android Studio: npx cap open android
* Следующим шагом необходимо добавить устройство, которое будет отображаться в эмуляторе. В данном приложении это Nexus Pixel 5.
* Затем в верхней панели Android Studio нажать на вкладку Build, в ней найти пункт Build Bundle(s) / APK(s) и нажать на пункт Build APK(s)

На этом моменте apk фаил приложения готов к скачиванию и использованию на мобильном устройстве.

# ЗАКЛЮЧЕНИЕ

В данной работе было произведено исследование гибридных веб приложений были рассмотрены преимущества и недостатки гибридных веб технологий, были сравнены разные JavaScript фреймворки, из которых, на основе сравнения, был выбран самый подходящий 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 Cloud Firestore, Capacitor, Vue.js и другие. Был произведен анализ создания веб приложений применяя соответствующий набор технологий.

В процессе выполнения работы были:

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

В ходе данной работы были рассмотрены преимущества гибридных веб 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 Cloud Firestore, Capacitor, Vue.js и другие.

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

В заключении можно добавить, что в перспективе гибридные веб 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 веб приложений, а также не только для ознакомления с методами и архитектурой веб приложений на основе гибридных веб 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 веб приложениями.

# БИБЛИОГРАФИЯ

1. *Определение гибридных мобильных приложений.* [on-line]. Доступен:

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

1. *Типы мобильных приложений* [on-line]. Доступен:

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

1. *Сравнение гибридных, мобильных и веб приложений* [on-line]. Доступен:

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

1. *Нативные vs. гибридные приложения*. [on-line]. Доступен:

<https://umbrellait.com/ru/blog/native-vs-hybrid-app/> [цитирован 10.05.20];

1. *Разница между нативным и гибридным мобильным приложением*[on-line]. Доступен: <https://wezom.com.ua/blog/chem-otlichajutsja-nativnoe-i-gibridnoe-mobilnye-prilozhenija> [цитирован 10.05.20];
2. *Фреймворки JavaScript* [on-line]. Доступен:

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

1. *Прогрессивный фреймворк Vue.js* [on-line]. Доступен:

<https://ru.vuejs.org/v2/guide/index.html> [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* [on-line]. Доступен:

<https://habr.com/ru/post/348818/> [цитирован 22.03.21];

1. *Сравнительный анализ фреймворков Vue, React и Angular* [on-line]. Доступен:

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

1. *Поверхностный обзор javascript-enabled BaaS платформ* [on-line]. Доступен: <https://habr.com/ru/post/150729/> [цитирован 15.03.20];
2. *Используйте Firebase, единый кроссплатформенный SDK от Google, чтобы улучшить приложения* [on-line]. Доступен:

<https://developer.android.com/distribute/best-practices/develop/build-with-firebase?hl=ru> [цитирован 27.03.20];

1. *Обзор облачных сервисов для разработки бэкенда мобильных приложений* [on-line]. Доступен: <https://habr.com/ru/company/surfstudio/blog/463435/> [цитирован 07.04.20];
2. *Использование сервисов Firebase при разработке мобильных приложений* [on-line]. Доступен: <https://cmsmagazine.ru/journal/cases-4264/> [цитирован 16.04.20];
3. *Vue.js* [on-line]. Доступен: <https://ru.vuejs.org/v2/guide/index.html> [цитирован 10.05.20];

**Приложение 1. Сравнение JS фреймворков.**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **React Native** | **Flutter** | **Ionic** | **Xamarin** | **Vue.js** | **Angular** |
| **Язык** | JavaScript + React | Dart | JavaScript, HTML, CSS + Angular, React, Vue | C# + .NET | JavaScript, TypeScript | JavaScript, HTML, CSS + Angular, React, Vue |
| **Разрабатываемые приложения** | Кроссплатформенные Гибридно-нативные | Кроссплатформенные Гибридно-нативные | Кроссплатформенные Гибридная веб-разработка | Кроссплатформенные Гибридно-нативные | Кроссплатформенные Гибридно-нативные | Кроссплатформенные  Гибридная веб-разработка |
| **Первый релиз** | 2015 | 2017 | 2013 | 2011 | 2015 | 2013 |
| **Разработчик** | Facebook | Google | Drifty Co. | Microsoft | Telerik | Drifty Co. |
| **Платформы** | Android, iOS, UWP | Android, iOS, Google, Fuchsia, Web, Desktop | Android, iOS, Web | Android, iOS, UWP | Android, iOS | Android, iOS, Web |
| **Открытый ресурс** | да | да | да  + платные пакеты | да  + платные пакеты | да | да  + платные пакеты |
| **Инструменты фронтенда** | Native + Declarative UI components | Built-in widgets | HTML, CSS + widgets | Xamarin. iOS/Android or Xamarin.Forms | полностью собственный интерфейс | HTML, CSS + widgets |
| **Производительность** | Высокая, близкая к нативной | Очень высокая | Средняя из-за веб технологий | iOS / Android: высокий, близкая к нативной | Высокая; есть Virtual DOM | Средняя из-за веб технологий |
| **Порог вхождения** | Низкий | Средний | Высокий | Средний | Низкий | Высокий |
| **Рендеринг** | Нативный | Нативный | Webview | Нативный | Нативный | Нативный |
| **Популярность** | высокая | высокая | средняя | низкая | высокая | средняя |

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