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 MOBILĂ HIBRIDĂ**

**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 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.

Aprobat:

Șeful catedrei de matematică și informatică

dr., conf. univ. C.Negara

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

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

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

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

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

**Etapele executării tezei de licență:**

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

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

(semnătura)

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

(*semnătura*)

СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 9](#_Toc70671971)

[ВВЕДЕНИЕ 10](#_Toc70671972)

[1. ГИБРИДНЫЕ МОБИЛЬНЫЕ ПРИЛОЖЕНИЯ 13](#_Toc70671973)

[1.1. Определение гибридных мобильных приложений 13](#_Toc70671974)

[1.2. Сравнение гибридных, нативных и веб приложений 15](#_Toc70671975)

[1.3. Известные примеры гибридных мобильных приложений 21](#_Toc70671976)

[2. ФРЕЙМВОРКИ JAVASCRIPT 23](#_Toc70671977)

[2.1. Общая характеристика фреймворков JavaScript 23](#_Toc70671978)

[2.2. Фреймворк VUE.js 23](#_Toc70671979)

[2.3. Фреймворк Angular 24](#_Toc70671980)

[2.4. Фреймворк React 25](#_Toc70671981)

[2.5. Сравнение фреймворков Vue, React и Angular 26](#_Toc70671982)

[2.6. Выбор фреймворка Vue для разработки демонстрационного приложения 29](#_Toc70671983)

[2.7. Vuetify - библиотека для создания гибридных мобильных приложений 30](#_Toc70671984)

[2.8. Платформа для создания гиб. мобильных приложений – Apache Cordova 30](#_Toc70671985)

[2.9. Google сервисы как неотъемлемая часть гибридного мобильного приложения 32](#_Toc70671986)

[3. ОПИСАНИЕ ДЕМОНСТРАЦИОННОГО ПРИЛОЖЕНИЯ 41](#_Toc70671987)

[3.1. Описание структуры приложения 41](#_Toc70671988)

[3.2. Наличие гибридных технологий в демонстрационном приложении 42](#_Toc70671989)

[3.3. Чем обосновано предпочтение гибридных технологий нативным 42](#_Toc70671990)

[3.4. Установка и настройка среды разработки 43](#_Toc70671991)

[3.4.1. Node Package Manager (NPM) 43](#_Toc70671992)

[3.4.2. Инструменты командной строки (CLI) 43](#_Toc70671993)

[3.4.3. Использование Cordova для определения гибридной трансляции 43](#_Toc70671994)

[3.4.4. Создание приложения 44](#_Toc70671995)

[3.4.5. Создание проекта в Firebase 45](#_Toc70671996)

[3.4.6. Интеграция Firebase 45](#_Toc70671997)

[Заключение 50](#_Toc70671998)

[Библиография 52](#_Toc70671999)

# 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 платформы, сервисы, приложения, позволяющие делиться информацией, обучаться, работать, веси бизнес и многое другое, не выходя из дома. Возрастающая популярность приложений привела к слиянию двух технологий, нативных и веб, в результате чего появились гибридные мобильные приложения.

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. На данный момент более 50% компаний нуждаются в специалистах по мобильной разработке или менеджерах мобильных проектов. В этой доле компаний нативные разработчики менее востребованы, а фроент-енд и мобильные разработчики, наоборот, привлекают в себе повышенный интерес. Данные исследований Gartner аналогично подтверждают повышение заинтересованности к мобильной разработке со стороны бизнеса. По их прогнозам, спрос на создание мобильных приложений будет расти в 5 раз быстрее, чем штат IT компаний, которые смогут покрыть этот спрос. Это открывает возможности для совершенствования гибридных приложений, которые все увереннее закрепляются на мировом рынке. Отдавая предпочтение новым, гибридным технологиям, или привычным за долгие годы существования, нативным, важно отслеживать состояние рынка и правильно расставлять приоритеты по скорости запуска проекта, стоимости и производительности.

*Цель*данной работы состоит в разработке демонстрационного гибридного мобильного приложения используя фреймворк 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 разных фреймворков JS;
* Аргументированный выбор фреймворка Vue.js для дальнейшей разработки демонстрационного приложения;
* Создание гибридного мобильного приложения, используя гибридные технологии на фреймворке 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 описан процесс создания гибридного мобильного приложения на выбранном фреймворке JS – Vue.js.

*Актуальность:* данная работа является актуальной, так как гибридные технологии является 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 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 приведены аргументы в пользу гибридных технологий и приведены примеры самых популярных приложений и компаний, использующих гибридные технологии.

Во второй главе описаны фреймворки JavaScript и проведен сравнительный анализ фреймворков, наиболее подходящих для гибридной раработки, таких как Vue, React и Angular для лучшего понимания того, почему и чем фреймворк Vue лучше подходит для создания гибридного мобильного приложения. Помимо того, были рассмотрены вспомогательные технологии, такие как Vuetify, Cordova и google сервисы для создания мобильного веб приложения.

В третьей главе описывается демонстрационное приложение и процесс его создания, используя выше перечисленный набор технологий, приводятся доводы в пользу наличия гибридных технологий в приложении и чем обосновано предпочтение гибридных технологий нативным.

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

# ГИБРИДНЫЕ МОБИЛЬНЫЕ ПРИЛОЖЕНИЯ

* 1. Определение гибридных мобильных 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, но не установлены на устройстве, таким образом они становятся доступными только с интернет-подключением и без него не работают [1].

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

Родство гибридных приложений с веб-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.



Рис.1.1. Гибридное приложение.

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 показатель UX.
* 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;
* Гибриды - хорошая опция для разработчиков, которые создают визуально насыщенные приложения, например, игры (которые не будут хорошо работать в виде веб-приложений).

Выбор подходящей модели (гибридной или нативной) мобильного приложения - это очень важный этап в его разработке [2], на который влияют несколько факторов, таких как:

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

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

* 1. Сравнение гибридных, нативных и веб приложений

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

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

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

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

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

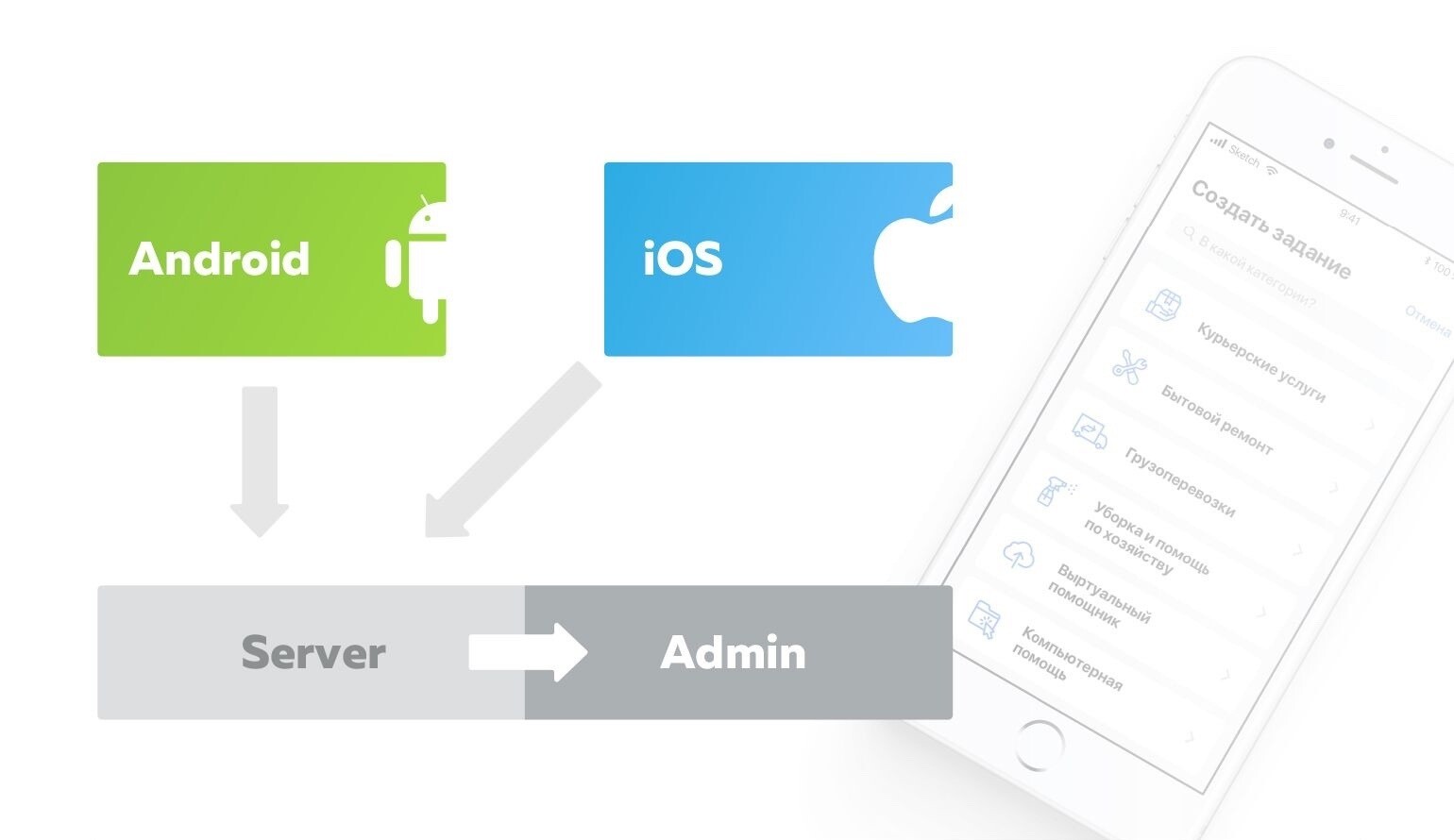


Рис.1.2. Принцип работы гибридного приложения.

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

Любой многопользовательский сервис состоит из клиентских приложений и серверной части. В более широком плане, интегрируется ещё сайт, телефония, api сторонних проектов.

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

Нативное приложение можно сравнить с приложениями для ПК. После покупки с загрузки установщика на компьютер можно сразу приступить к использованию и не беспокоиться об интернет-соединении, так как приложение запускается прямо на ПК. Аналогично с мобильными приложениями, пользоваться функционалом можно сразу после установки, так как он не требует наличие подключенного интернета для дополнительных действий, например, отправки запросов на сервер. Фактически, приложение скачивается "от и до" и, если в логике приложения не рассчитан выход в интернет, он не обязателен. Хорошим примером являются одиночные игры, переводчики, приложения для сопряжения с другими устройствами посредством Bluetooth.

Гибрид - это скорее подход к программированию, чем независимый приложения. Его идея заключается в том, что функционал программируется на стороне сервера, а на стороне клиента остаётся только необходимый минимум. К примеру, если рассматривать обычный мобильный магазин, то GPS и функция отображения данных будет на стороне приложения, а каталог, карточки товаров будут загружаться с сервера.

Для демонстрации как бы в гибриде и нативе выглядела платформа для создания сервиса поиска исполнителей, аналогичной Shazam. Красным выделено то, что происходит на стороне приложения для телефона, а зелёным - на стороне сервера.

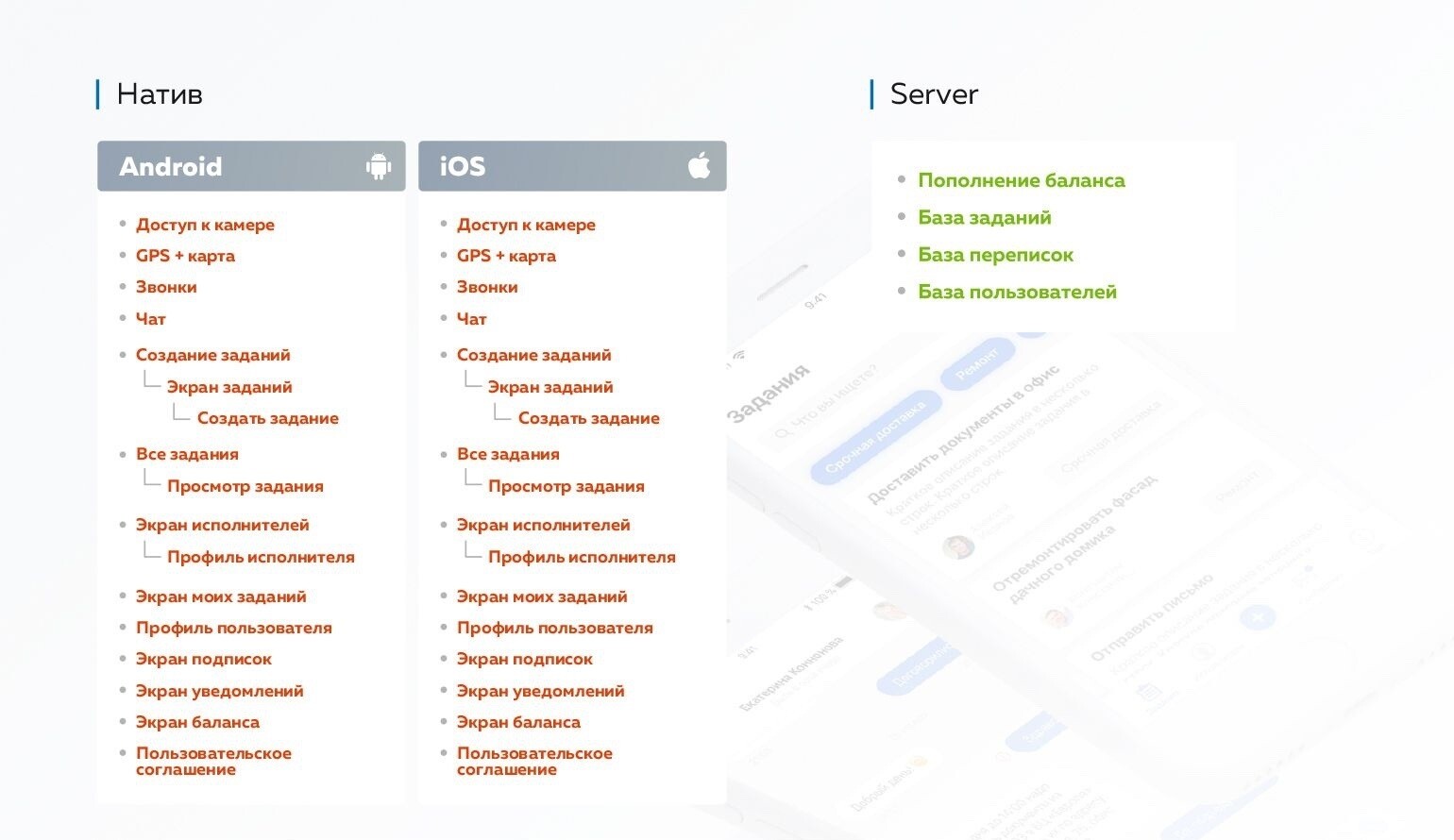


Рис.1.3. Реализация сугубо на нативных технологиях.

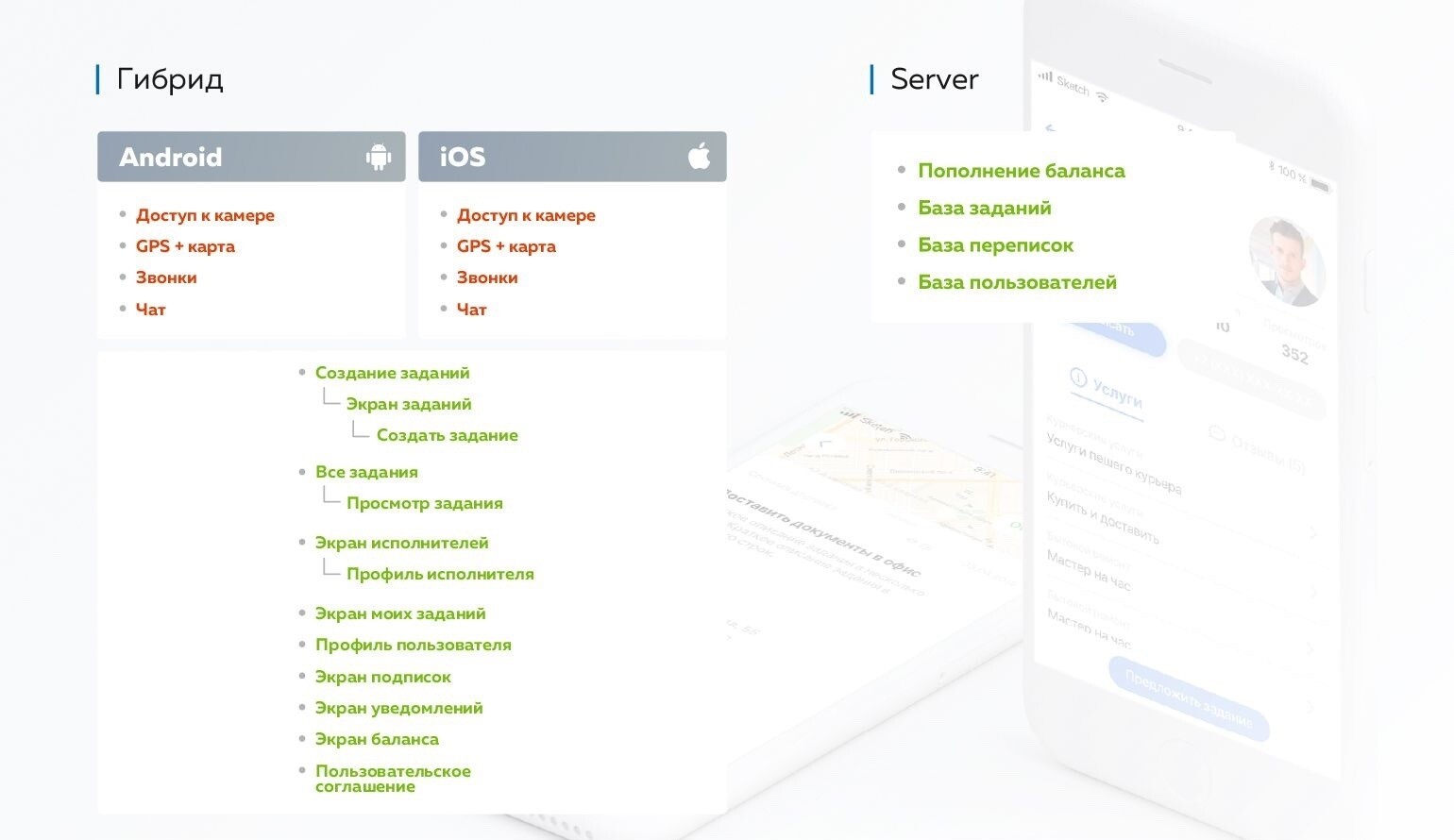


Рис.1.4. Реализация сугубо на гибридных технологиях.

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

Как видно из изображений, единственная разница между нативом и гибридом - это объём функционала, который реализуется на стороне телефона, но самым важным плюсом для гибридного подхода 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.

У бизнес-приложений заниженные требования к вычислительной мощности телефона и отсутствию интернета, он нужен для покупки товара либо отображения новых товаров. Если речь идет об играх, где в любом случае нужно интернет-соединение, нативные приложения подойдут гораздо лучше. В зависимости от требований к приложению необходимо выбрать правильный подход к разработке, 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.

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

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

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.), то необходимо делать нативное приложение. Если скорость работы не так важна, лучше выбрать гибридное приложение. А если необходимо разработать страницу-визитку, лучше разработать веб-приложение.Часто производители делают нативные приложения платными. Гибридные же в основном можно найти без оплаты в Google Play, App Store и других маагазинах.Не менее важен и язык разработки. К примеру, для платформы IOS будет правильнее привлечь разработчиков на Objective-c/ Swift, а на разработку под Android – разработчиков на Kotlin.

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.
* Быстрый вывод на рынок: гибридные приложения разрабатываются быстрее, так как в ходе процесса используются стандартные веб-технологии, которые легко обслуживать в долгосрочной перспективе.
* Дистрибуция для нескольких платформ: этот вид приложений распространяется через оба магазина, что позволяет охватить большее число пользователей.

Однако у гибридной разработки есть свои слабые места:

* 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.
* Интеграция сторонних сервисов: Вы не можете разрабатывать гибридное приложение на одном только JavaScript. Потребуется интегрировать такие фреймворки для гибридной разработки, как Cordove, Ionic или React Native, каждый из которых требует определенных усилий для освоения.

Так же из неприятных сторон гибридной разработки выделяется 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 и ограниченное представление визуальных и графических элементов, в частности, анимации.

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.

* 1. Известные примеры гибридных мобильных приложений

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

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

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

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

Приложение TripCase — органайзер для планирования путешествий.

* Загружается из магазина;
* Может использовать геолокацию;
* Необходимо подключение к интернету;
* Может использовать сотовую сеть;
* Можно настроить push-уведомления

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

* 1. Дополнительные инструменты для создания гибридного приложения
     1. Firebase Realtime

Разработка серверной части - один из самых трудных этапов создания приложения. Зачастую, при планировании разработки проекта недооценивается необходимый объем ресурсов и время создания бэкенда (Рис.2.1.). Еще одна проблема - ограниченность доступных команде ресурсов. Чаще всего разрабатывать бэкенд приходится с помощью тех инструментов и технологий, которыми владеют члены команды. Процесс получается длительным, а само приложение — достаточно сложным и дорогим с точки зрения сопровождения. А длительная разработка бэкенда на даже относительно простых проектах ведет к увеличению расходов и другим рискам: расход денег без визуально видимого результата.

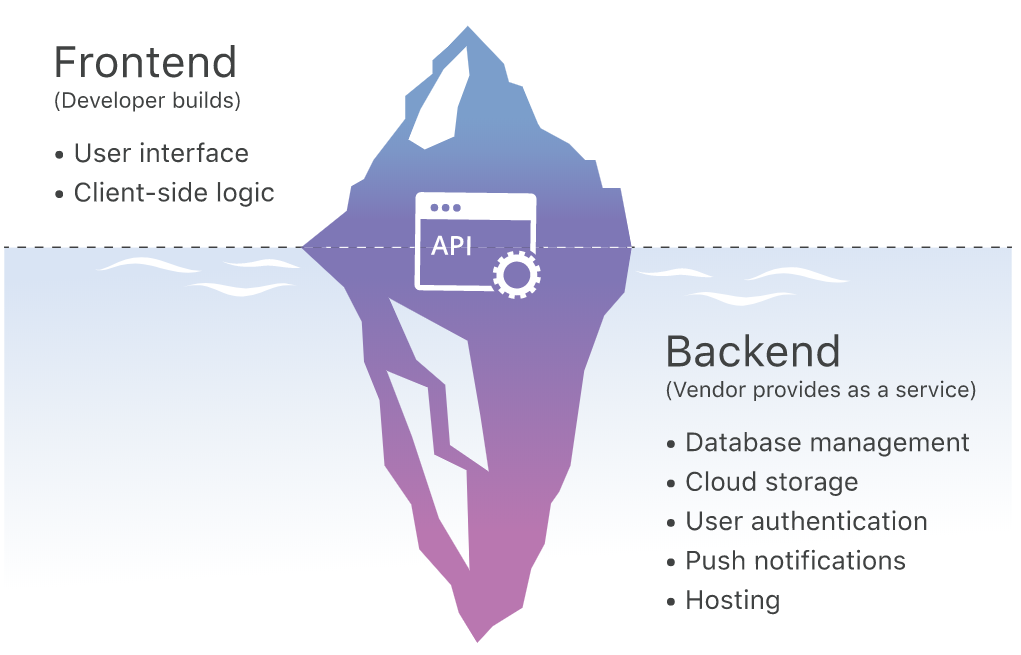


Рис.2.1. Принцип работы современного приложения.

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 [12]. 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. Все эти сервисы имеют собственный API, чтобы легко встраиваться в приложения.

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

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

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

База данных Firebase Realtime позволяет создавать многофункциональные приложения для совместной работы, обеспечивая безопасный доступ к базе данных непосредственно из кода на стороне клиента (Рис.2.2.).



Рис.2.2. 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.

Польза google сервисов в гибридном мобильном приложении очень весомая, так как они позволяют легко и быстро добавить полезный функционал в любое место в приложении.

* + 1. 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 Ionic Framework.

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. 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, который позволяет получить доступ к кросс-платформенным преимуществам и эффективности.

Как и Capacitor, Cordova - это проект с открытым исходным кодом, который запускает веб-приложения на нескольких платформах, но без использования Electron и веба как PWA.

Хотя Cordova и Capacitor имеют некоторое сходство, проекты принимают очень разные решения по нескольким ключевым моментам, так что опыт в обоих случаях сильно различается. Capacitor, выпущенный в 2018 году, использует множество новых современных API-интерфейсов, которые не были доступны, когда Cordova создавалась еще в 2009 году.

* Выбор в пользу Capacitor становится более очевидным при учете таких факторов, как 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 работать бок о бок

Поскольку приложения Capacitor являются нативными приложениями, а ключевым аспектом дизайна Capacitor является использование собственных инструментов, Capacitor позволяет командам, имеющим сочетание традиционных мобильных разработчиков и веб-разработчиков, совместно работать над проектами мобильных приложений.

Cordova, напротив, работает через уровень абстракции, который управляет базовым проектом собственной платформы и исходными файлами самостоятельно. Это затрудняет переход к нативному коду или работу с традиционным нативным процессом мобильной разработки и может привести к потере пользовательских изменений.

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 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

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

Из преимуществ фреймворков необходимо выделить то, что данные функции упрощают взаимодействие веб-приложения с сервером и ускоряют манипулирование элементами на веб-странице или в приложении. Они позволяют осуществить мгновенную обратную связь с пользователями в реальном времени. С помощью фреймворков можно создавать как целые сайты, так и отдельные модули. На веб-сайтах без фреймворков все данные и код хранятся на сервере, с каждым изменением происходит загрузка с сервера всей страницы, а при использовании фреймворков перезагружаются только части страницы, где изменился контент. Недостатками можно назвать только временно неполную поддержку поисковыми системами [3].

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

* 1. Выбор фреймворка Vue для разработки демонстрационного приложения

Vue - это современный фреймворк для разработки пользовательских Web интерфейсов с использованием паттерна MVVM (Model-View-ViewModel). Ядро фреймворка хорошо справляется с задачами уровня представления (view), что облегчает интеграцию с различными библиотеками и проектами, в число которых входит Vuetify. На Vue чаще всего создаются сложные одностраничные приложения SPA. Основными конкурентами Vue являются React и Angular, которые в совокупности заполняют 100% рынка Web-разработки. Продукты, написанные на этом фреймворке быстро подгружаются на страницу и не нагружают проект за счет небольшого веса библиотеки. Его широко используют такие компании, как Alibaba, Baidu, Xiaomi, GitLab.[4]

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

Функции Vue.js:

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

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

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

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

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

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

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

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

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

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

* 1. Фреймворк 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, 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 Ionic Framework 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.

* 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, React Native, Angular, Ionic, Flutter, Intel XDK, Appcelerator Titanium, Sencha Touch, Telerik’s Kendo UI, PhoneGap, Xamarin.

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

Особенности Vue:

* 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 фаилами, 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.



Рис 2.4. Сравнение фреймворков.

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, 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.



Рис 2.4. Сравнение фреймворков.

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.

# 3. ОПИСАНИЕ ДЕМОНСТРАЦИОННОГО ПРИЛОЖЕНИЯ

**3.1. Описание структуры приложения**

В ходе данной работы было создано гибридное мобильное приложение «Дневник 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 сдвигуть 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 демонстрационном приложении

Данное гибридное приложение развертывается в собственном контейнере, который использует мобильный объект WebView. При использовании, данный объект отображает адаптированный веб-контент с помощью веб-технологий (CSS, JavaScript, HTML, HTML5). Веб-контент может отображаться либо сразу после открытия приложения, либо асинхронно, только для определенных частей приложения [10]. Данное 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.

3.3. Чем обосновано предпочтение гибридных технологий нативным

В демонстрационном приложении «Дневник 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 использовался фреймворк Vue.js и Ionic, а также технология Capacitor что позволило сделать приложение кросс-платформенным, 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 одинажды 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 плотформу, 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
     1. 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.

* + 1. 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.

* + 1. 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. В папке views были созданы страницы: Analiza numerica.vue, ArhitecturaCalculatorului.vue, ArchitecturaSetei.vue, Etica.vue, PADM.vue, Home.vue, SGBD.vue, SistemeInteligente.vue и VsePredmeti.vue.

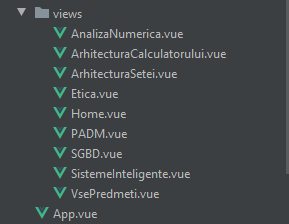
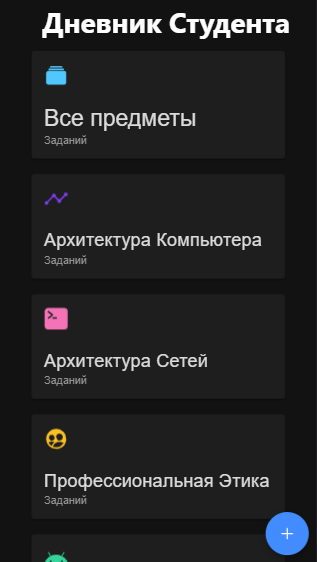


Рис. 3.5. Структура папки views

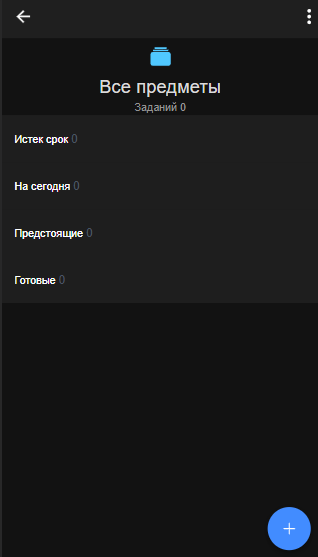
Для наглядности отображения в каждый компонент была импортирована библиотека IonIcons которая будет отображать иконки по каждому из предметов на каждой странице.

После создания компонентов каждого предмета была разработана структура страницы которая бы являлась 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:



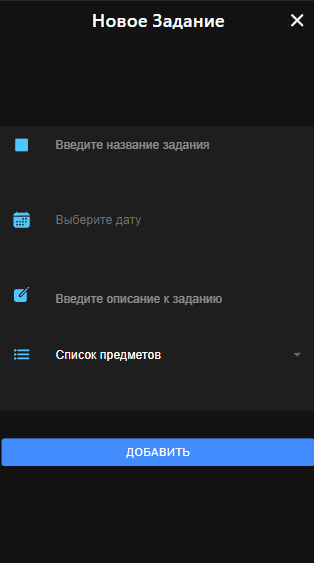
Далее в папке router в файле index.ts были импортированы все созданные компоненты страниц и был создан массив роутов для создания роутера который бы производил переходы от одной страницы к другой без её перезагрузки.

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



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, 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.



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) => {

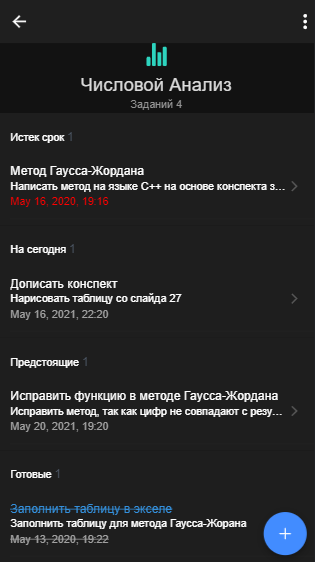
return item.lesson == lesson;

}).length;}

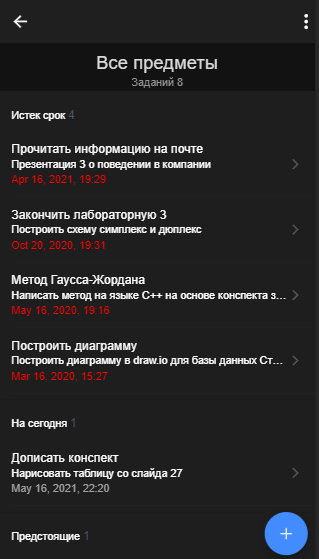
Этот пример отвечает за вывод количества заданий для предметов. По аналогии с ним были написаны методы для каждого предмета и для каждого состояния индивидуально.

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:



Все задания можно увидеть в разделе «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 web приложений были рассмотрены преимущества и недостатки гибридных web технологий, были сравнены разные JavaScript фреймворки, из которых, на основе сравнения, был выбран самый подхоящий для разработки гибридного мобильного приложения – Vue.js, были изучены основные элементы фреймворка, а также, как работают технологии, имплементированные в данное демонстрационное приложение, такие как Firebase Cloud Firestore, Capacitor, Vue.js и другие. Был произведен анализ создания web приложений применяя соответствующий набор технологий.

В процессе выполнения работы были:

1. Изучены основы гибридных web приложений;
2. Исследованы возможности использования гибридных web 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 web технологий, были изучены их основные элементы , были исследованы фреймворки JavaScript, на которых возможно осуществить написание гибридного веб приложения их основные элементы, а также, как работают ключевые технологии, входящие в данный набор, такие как Firebase Cloud Firestore, Capacitor, Vue.js и другие.

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

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

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

# Библиография

1. *Определение гибридных мобильных приложений. Доступен:*

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

1. *Типы мобильных приложений Доступен:*

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

1. *Сравнение гибридных, мобильных и веб приложений*

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

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

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

1. *Прогрессивный фреймворк Vue.js*

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

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

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

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

Declaralie privind asumarea raspunderii

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

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

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

Data Semnatura