Семинарска работа по предметот

Напреден Веб Дизајн

Тема: Технологии за push известувања

Изработила:

Вероника Огњановска 181045

Ментор:

м-р Бобан Јоксимоски

Февруари 2021

Апстракт

Push известувањата се важен дел во комуникацијата помеѓу развивачите на апликацијата и нејзините корисници. Можноста за ваков тип на известувања, во голем број примери, ја подобрува и задоволноста на корисниците од апликацијата која истите ги имплементира. Како цел на оваа семинарска работа е најпрво низ теориски разгледувања и потоа со пример, да ги објасни технологиите за push известувања.

Вовед

Во време кога сите сме пред нашите мобилни уреди или компјутери, не може да не ги приметиме сите известувања што ни притигнуваат од различни апликации. Овие известувања се со најразлична содржина, од маркетинг промоции, новости од страните што ги сакаме, до известувања за да се стави уредот на полнење. Овие известувања се нарекуваат push известувања[1] (англиски од „бутка“) затоа што „ буткаат “ информации до корисникот без пред тоа корисникот да ги побарал.

Push известувањата се директен пат до корисниците и како такви имаат големо значење во ерата на дигитализација. Тие бараат големо внимание во однос на содржината што се пренесува до читателите за да може да се задржи нивното внимание и со тоа да се исполни и задачата на известувањето. Да извести! Ова треба да се направи додека содржината е сеуште нова, во вистинско време и информативна за корисникот кој ја примил.

Постојат различни технологии за праќање на горенаведените изестувања, како и различни сервиси кои ни овозможуваат полесно користење на овие технологи. Во следните делови, ќе се запознаеме подетално со push известувањата, како и подетален преглед во push технологиите и протоколите кои се користат за испраќање на овие известувања. Исто така ќе споменеме неколку сериси кои имплементираат технологија за push известувањата. За крај ќе разгледаме веб апликација развиена со помош на Vue.js и Firebase како сервис за push известувања.

Push известувања

Постојат различни дефиниции за што се push известувања кои се среќаваат наоколу. Некои ги нарекуваат активни пораки, други ги нарекуваат богати пораки, но се сретнува и терминот персонализирани пораки. Сите наведени описи се добри. Дел од карактеристиките на овие известувања се:

- Испратено од апликација или веб-страница на телефонот или работната површина на корисникот.
- Се користи за испраќање предупредувања и пораки до корисникот во реално време.
- Може да содржи богата медиумска содржина, како што се слики, GIF-карти или видеа.
- Содржината во рамките на известувањето може да биде персонализирана за да го поттикне посакуваното дејство од корисникот[2].

Во кратки црти, push известувањата се кратки пораки што се појавуваат на мобилниот или работната површина на корисникот, потикнувајќи ги да превземат некоја акција.

Кои типови push известувања постојат?

Двата главни типа на push известувања се известувања за веб и мобилни[3].

Веб: овозможува пораките да се појавуваат и на работната површина (дури и ако корисникот не е на вашата веб-страница). Корисниците треба да направат само неколку клика при отворање на веб-страницата за да дозволат известувањата да бидат прикажани на нивните уреди.

Мобилни: испраќа пораки до корисниците дури и ако тие не ги користат своите телефони, но можат да се појавуваат само на мобилни уреди. Корисникот треба да ја инсталира апликацијата со цел да добива известувања. Овие типови сигнали се користат со цел да ги известат корисниците за претстојните настани, новостите, да ги потсетат да извршат одредена задача зависно целите и потребите на апликацијата.

Push технологија

Важен дел од push известувањата е технологијата[4] на која тие се градат и од која го добиваат името и карактеристиката push за приказ на информациите.

Push технологија (или сервер push) е интернет базирана комуникација каде барањето за дадена интеракција е иницирана од издавачот или некој централен сервер. Се користи модел на објавување/претплати (анг. "publish/subscribe"). Клиент „се претплаќа “ на разни информативни „канали/теми “ обезбедени од страна на сервер со цел секогаш кога е достапна нова содржина на еден од тие „канали/теми “ , клиентот да ги добие со тоа што серверот ќе ги турне тие информации до него.

Push известувањата се само кратки пораки кои се пренесуваат со помош на push технологија.

Push технологија наспроти Pull технологија

Во основа, постојат два начина на кои информациите се пренесуваат во светот на Интернетот и технологијата, и тоа се pull (или повлечи) и push (или притисни/бутни). Разликата помеѓу push и pull технологијата е во иницирањето на комуникацијата или поточно кој ја започнува интеракцијата[5]. Доколку клиентот иницира барања тогаш станува збор за користење pull технологија, наспроти push технологијата која ја видовме погоре.

	Pull технологија	**Push технологија**
Дефиниција	Клиент праќа барања за информација од сервер.	Сервер иницира ажурирања на информацијии до клиент.
Пример	Веб прелистувач побарува веб страна.	Email сервер испраќа email до email клиент.

Табела 1. Pull технологијата наспроти Push технологија

Технологии за push известувања

Технологијата за push известувања овозможува испорака на информации од апликација до мобилен уред или десктоп компјутер без конкретно барање од апликацијата т.е корисникот, што значи дека апликацијата не мора да се отвори за да добие push известување.

Како работи технологијата за push известувања?

Како компоненти при праќање на push известувања се вклучуваат[6]:

- Сервис за push известувања(анг. "Push notification service"). Секој оперативен систем (ОС), вклучително и iOS, Android, Windows, имаат свој сервис.
- Издавач на апликација (анг. "App publisher"). Издавачот на апликацијата вклучува сервис за известувања во апликацијата.
- Апликација за клиент. Ова е апликација специфична за ОС, инсталирана на уредот на корисникот и добива дојдовни известувања.

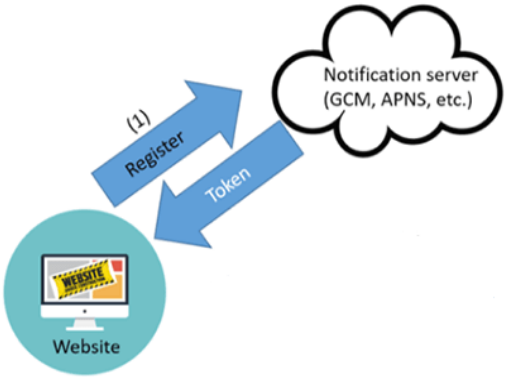
Како почеток, при креирањето на апликацијата, издавачот на апликацијата вклучува сервис за push известувања, при што добива интерфејс за програмирање на апликации (анг."Application programming interface - API"). API е начин за апликацијата да комуницира со сервисот.

Дел од сервисите за известувања се: Apple Push (APNS), Microsoft Push Notification (MPNS), Google Cloud Messaging (GCM), Firebase Cloud Messaging (FCM) како нова верзија на GCM под брендот Firebase.

Накратко можеме да кажеме дека процесот се состои од две фази[7] откако корисникот ќе ја има соодветната апликација на својот уред:

1. Регистрација на уред

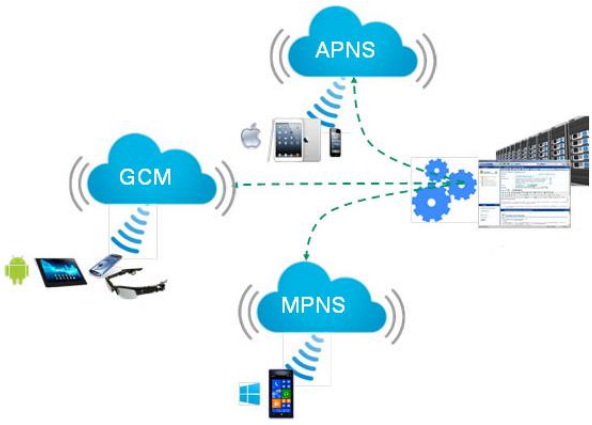
Кога некој корисник ќе ја отвори апликацијата или веб-страната за прв пат, ве прашува дали сакате да прифаќате известувања. Доколку корисникот каже да, апликацијата го регистрира уредот на серверот, и потоа е подготвена да прима пораки од сервисот. Апликацијата мора да е подготвена за оваа функција, и соодветно желбите на корисникот, да го регистрира уредот или во спротивно, да не го регистрира. Регистрирањето се прави кај сервисите за известувања каде истиот обезбедува стандарден ID токен (уникатен идентификатор) за таа веб-страница или апликација (домен) (Слика 1). Сето тоа се случува на ниво на уред.



Слика 1

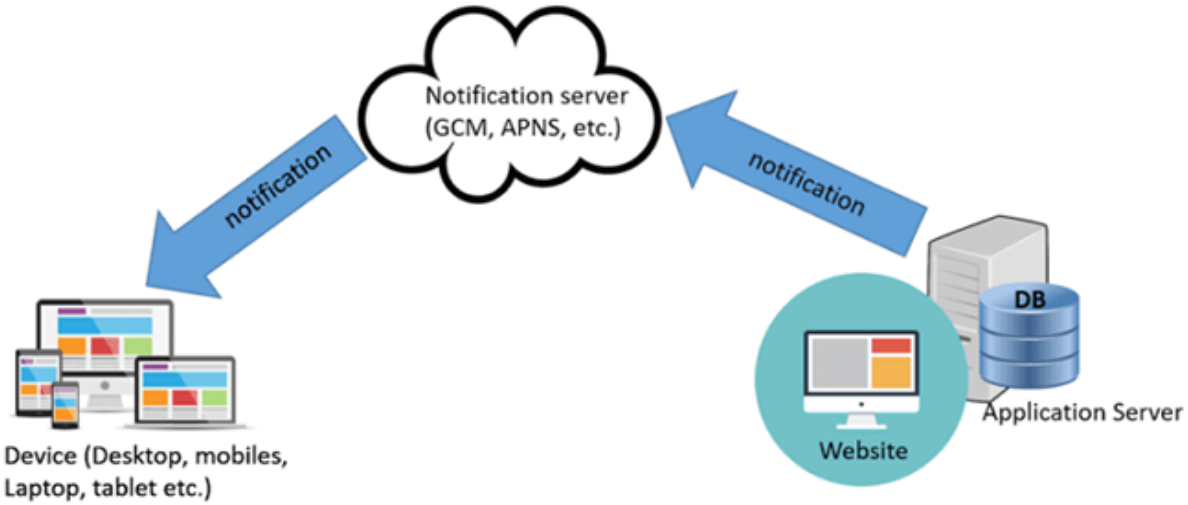
2. Испраќање на известувања

Издавачот на апликации компонира рачна порака преку кориснички интерфејс или издавачот поставува автоматска порака што треба да се испрати преку API до сервисот за известувања. Издавачот исто така ја дефинира публиката на која ќе и биде испратено push известувањето. Може да биде кон сите корисници, групи или индивидуални корисници на апликацијата. Времето на испраќање на порака може да биде закажано од страна на издавачот, или да се испрати во моментот.



Слика 2

Како што сме запознаени, push известувањата стигнуваат до уредот кога апликацијата работи и кога таа не работи. Ова се случува бидејќи пораките не се испраќаат директно до уредот, туку тие се испраќаат до сервисот за известувањат и преку него се доставува пораката до уредот, како што е прикажано на слика 2 и слика 3.



Слика 3

Целта што треба да се запомни тука е: известувањата мора да одат преку сервер за известувања до уредот (работна површина, мобилни телефони, лаптоп, таблет, итн.).

Web Push концепти

Push порака

Push порака[9] е порака испратена од сервер до клиент.

Push известување

Push известување, претходно подетално објаснето, е известување создадено како одговор на push порака.

Notifications API

Notifications API (англиски од „API за известувања“) претставува интерфејс што се користи за конфигурирање и прикажување на известувањата до корисникот.

Push API

Push API[8] е интерфејс кој им дава можност на веб-апликациите да примаат пораки што им се туркаат до нив од страна на сервер, без оглед дали веб-апликацијата е во преден план, па дури и моментално е вчитана, од кориснички агент т.е. уредот.

Application server

Терминот application server (англиски од „сервер за апликација“) се однесува на компонентите од страна на серверот на веб-апликацијата.

Push service

Терминот push service (англиски од „услуга“) се однесува на систем за рутирање на push пораки од сервер до клиент. Секој веб-прелистувач има свој push service.

Web Push Protocol

Web Push Protocol (англиски од „Веб push протокол “) опишува како сервер за апликации или кориснички уред комуницираат со push service.

Service worker

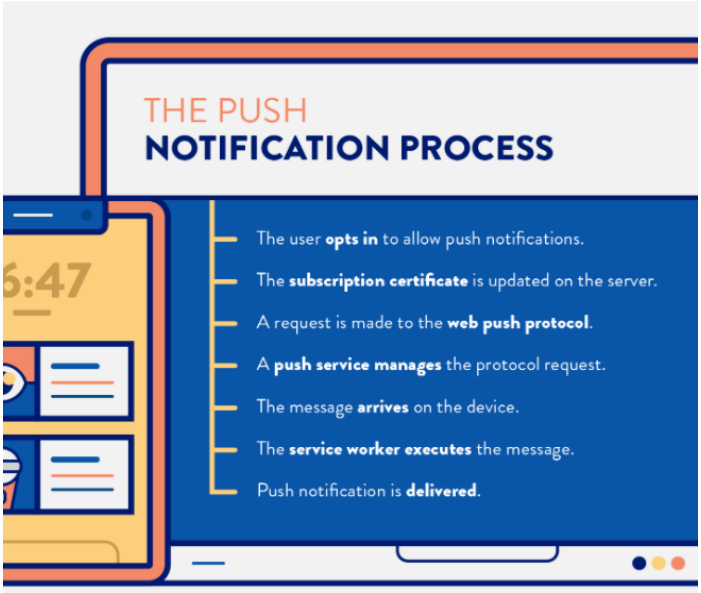
Service worker (англиски од „сервисниот работник“) е вид на event-driven веб-работник (англиски од „воден од настани“). Во суштина, програмабилен мрежен работник што ви овозможува да контролирате како се постапуваат со мрежните барања од и до вашата страница.

Service worker има додатоци, специфирани од Push API, за да се обезбеди влезна точка за користење push пораки. Тие исто така ги следат и реагираат на настаните за push и претплата. Дел од нив се ServiceWorkerGlobalScope и ServiceWorkerRegistration.

Поврзување на web концепти

За веб-push известувања, се користи Web Push Notification Service, и секој веб-прелистувач има свој ваков сервис за испорака на известувања.

За веб-апликацијата да добива push пораки, таа мора да има активен service worker. Кога service worker е активен, тој може да се претплати за да добива push пораки користејќи методата subscribe() од PushManager интерфејсот. Овој интерфејс овозможува начин за примање на известувања од сервери.

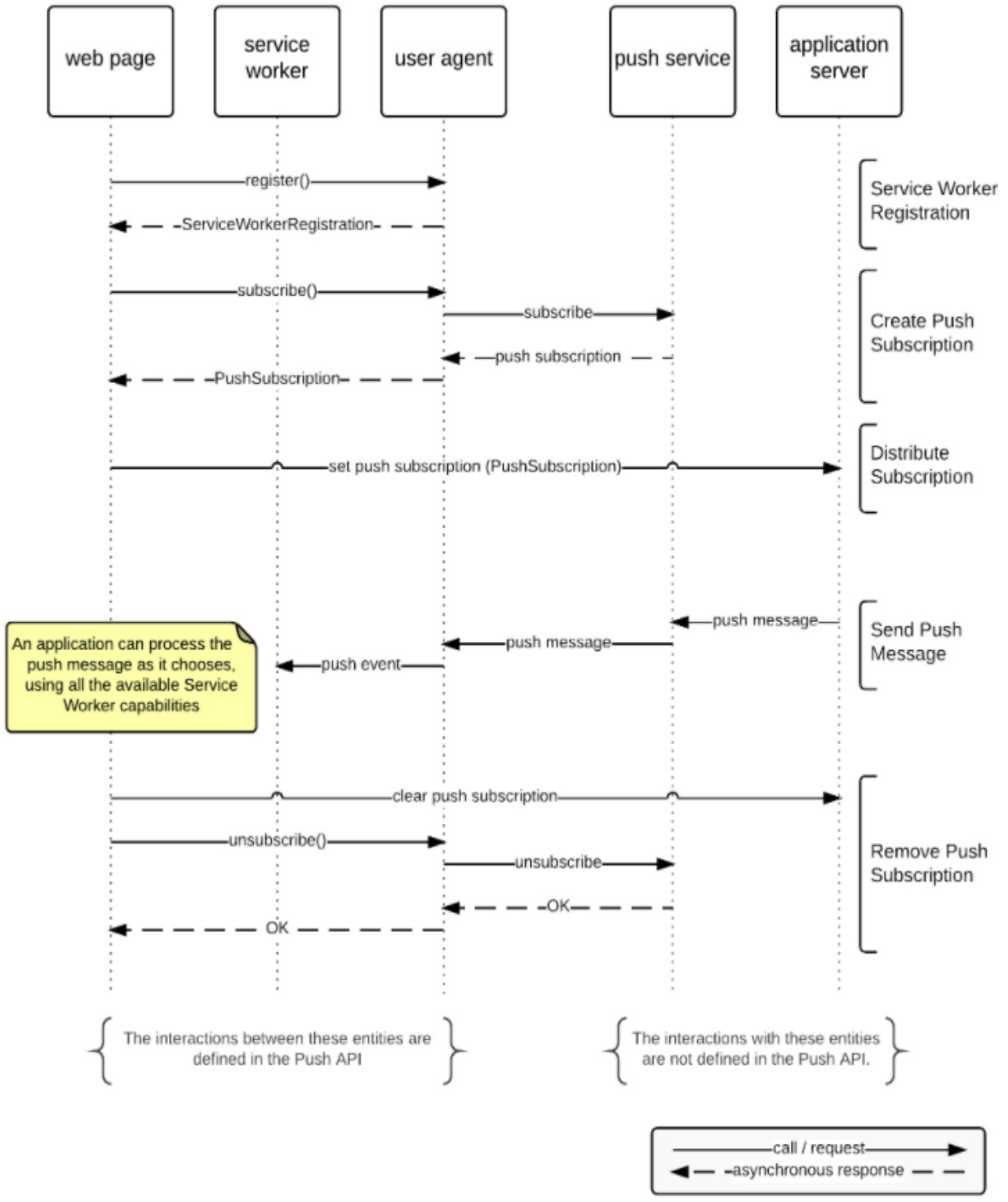


Слика 4

При push настан имаме покренување на управувач на настан ServiceWorkerGlobalScope.onpush, и со помош на service worker кој слуша на вакви настани, соодветно да овозможиме апликациите да реагираат. Користејќи го методот showNotification() на интерфејсот ServiceWorkerRegistration, се создава push известување на активениот service worker кое понатаму се прикажува на корисникот (Слика 4).

Секвенцен дијаграм

Пример проток на настани за претплата, испорака на порака и откажување преплата (Слика 5).



Слика 5

Mobile App Push Notifications

Постојат мали разлики помеѓу различните оперативни сервиси, но потребните чекори за испраќање известувања до мобилна апликација се приближно слични помеѓу различни оперативн сервиси, како и со web push.

Потребно е да се поврзе апликацијата со сервис за известувања. Притоа зачувувајќи ги конекциските параметри. При инсталирање или при користење на апликацијата, да се побара од корисникот да дозволи пристигнување на push изестувања. Со помош на воспоставената конекција до даден сервис за известувања да се пратат и прикажат известувањата до корисникот.

Web Push Notification vs. Mobile App Push Notifications

Важно е да се разбере разликата помеѓу веб и мобилни апликации[10], бидејќи тие можат да претставуваат различни можности за вашата push стратегија.

Мобилни push известувања се пораки примени преку апликација преземена на вашиот мобилен уред или таблет. Тие се особено корисни за ангажирање на корисници дури и кога тие не го користат уредот бидејќи се доставени до заклучениот екран на паметниот телефон.

Известувањата за веб-pushce пораки што доаѓаат од веб-страница и се доставуваат преку прелистувач откако корисникот се одлучил да ги прими. Тие се корисни за достигнување до корисници кои моментално се активни на уредот, но можеби не користат активно вашата веб-страница или веб-апликација. Иако известувањата за веб-push можат да бидат презентирани кога корисникот повторно ќе се вклучи во уредот, тогаш податоците може да бидат застојани или неточни.

Имплементирање на push известувања и протоколи

Протоколи

HTTP

Hypertext Transfer Protocol или HTTP[11] е протокол кој овозможува преземање ресурси, како што се HTML-документи. Тоа е основата на секоја размена на податоци на Интернет и тоа е протокол клиент-сервер, што значи дека барањата се иницирани од примателот. Работи на едноставн начин, барање на клиент - одговор од сервер, додека обратно не е дозволено.

WebSocket

WebSocket[12] е протокол за компјутерски комуникации, кој обезбедува целосен full-duplex т.е. двонасочен комуникациски канал преку единствена TCP врска.

Протоколот WebSocket им овозможува на клиентот (апликацијата) и серверот да испраќаат, едни на други, какви било информации во кој било момент, под услов врската/конекцијата да биде воспоставена. Слично на тоа како разговарате со вашиот пријател преку повик, каде што секој од вас може да зборува и / или да слуша, откако ќе се изврши повикот, во секое време, се додека тој не се прекине.

Push известувања со long polling (долго анкетирање)

Еден едноставен начин за имплементирање[13] на push известувања може да биде со long polling. Ова значи, апликацијата едноставно испраќа HTTP барање до серверот. Серверот, наместо веднаш да одговори на барањето, само го одржува во состојба на чекање. Сега, само кога има push известување што треба да се испрати до апликацијата, серверот ги разгледува HTTP барањата што чекаат, испратени од дадената апликација и тогаш го испраќа известувањето како одговор.

Ова е всушност начинот на кој традиционално се постапуваше со push известувањата. Од повеќе причини, овој метод сега се гледа со презир, најмногу поради тоа што се покажа дека има интензивно влијание на серверската страна.

Push известувања со WebSockets

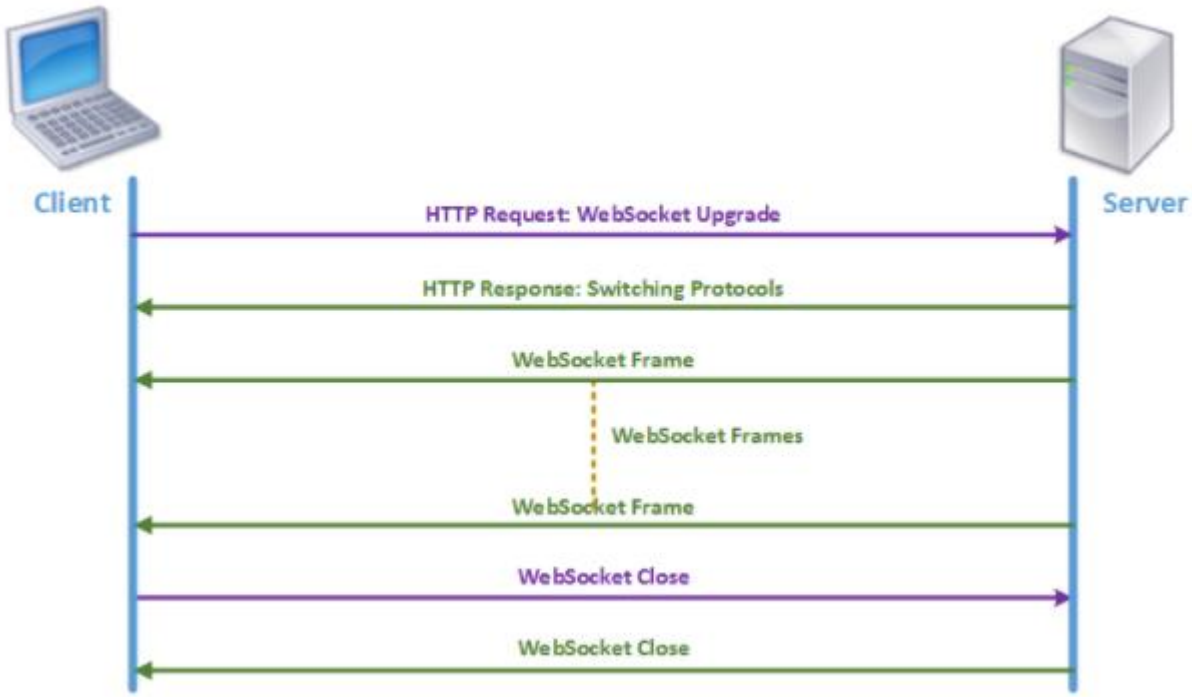
Има подобро решение. Корисничкиот интерфејс, веднаш штом се активира, се воспоставува врска и опстојува со сервисот за push известувања. Апликацијата постојано слуша на оваа врска, доколку има ново известување, и потоа ја обработува пораката соодветно.

Апликацијата и сервисот за push известување можат во ефективни интервали да проверуваат и со тоа да се осигурат дека врската/конекцијата е функционална со помош на анкети т.е. polling. Доколку врската е откриена како прекината, од страна на апликацијата, таа може едноставно да испрати ново барање за конекција. Од друга страна, доколку серверот е тој што открие прекината врска, таа едноставно ги чисти предметите/објекти поврзани со соодветната врска од меморијата. Оваа точна филозофија е собрана за WebSocket протоколот.

Креирање комуникациски канал WebSocket

Слично на тоа, за да се воспостави поврзување на WebSocket, апликацијата прави HTTP-барање до серверот, со нешто што се нарекува „Надградба“ заглавие поставено за да пренесе дека клиентот сака да го надгради протоколот за комуникација во WebSocket. Серверот потоа ги проверува ингеренциите за автентикација и соодветно ја одобрува надградбата / или ја негира (и ги затвора врските). Ако надградбата е одобрена, апликацијата добива потврда по што може постојано да ја слуша оваа врска за нови известувања (Слика 6).

Покрај тоа, има континуирано анкетирање за да се осигура дека врската не поминала „во мирување“ или, не се прекинала.



Слика 6

Push Notification Service

Сервис за известувања[14] (анг. " notification service") обезбедува средства да испратите известување до многу лица одеднаш. Дел од овие сервиси се претходно споменатите Apple Push (APNS), Microsoft Push Notification (MPNS), Google Cloud Messaging (GCM), Firebase Cloud Messaging (FCM) како нова верзија на GCM под брендот Firebase. Овие сервиси за известувања користат имплементација на технологија за push известувања користејќи познати протоколи за праќање на пораки до корисниците.

Повеќето од нив нудат не само сервис за push известувања, туку и многу други корисни алатки.

Amazon Simple Notification Service (Amazon SNS)

Доколку веќе користите какви било услуги на Амазон, тогаш можете лесно да ги користите и неговите AWS push известувања (Amazon Web Services) преку Сервисот за едноставно известување на Амазон или Amazon SNS[15].

При испраќање на известувања за push до Apple, Google, Fire OS, Windows уреди, SNS ви овозможува да испраќате пораки и до еден примач и до голем број примачи истовремено. Таа е наменета да работи во микросервисна архитектура и која може да поддржува event-drive ситуации и да поддржува раздвојување (анг." decoupling") на апликациите, т.е. да не се зависни апликациите едни од други.

Како работи Amazon SNS

Програмерите прво мора да создадат „тема“ што е „пристапна точка“ - идентификувајќи одреден предмет или тип на настан - за објавување пораки и дозволување на клиентите да се претплаќаат заизвестувања. Откако ќе се создаде тема, сопственикот на темата може да постави политики, како што е ограничување на тоа кој може да објавува пораки или да се претплаќа на известувања или да наведе кои протоколи за известување ќе бидат поддржани (т.е. HTTP / HTTPS, е-пошта, СМС).

Претплатници се клиенти заинтересирани да добиваат известувања од теми од интерес, тие можат да се претплаќаат на тема или да бидат претплатени од сопственикот на темата. Кога издавачите имаат информации или ажурирања за да ги известат своите претплатници, тие можат да објават порака на темата - што веднаш го активира Amazon SNS да ја достави пораката до сите претплатници. Amazon SNS има механизми кои помагаат во рутирањето на пораките кои не успеале да стигнат до корисникот, со цел да не бидат изгубени. (Слика 7)

Слика 7

WebSocket во Amazon API Gateway

Почнувајќи од декември 2018 година, веб-сервисите на Амазон (AWS) имплементираа многу барана карактеристика: WebSocket во Amazon API Gateway[16][17]. Со WebSocket достапен во AWS API Gateway, ние можеме да креираме апликации управувани од клиенти, во реално време, кои користат двонасочна / двонасочна комуникација.

Со нив ние можеме да креираме интеракција објавување/претплати слична на Amazon SNS. Додека пак можеме и да направиме интеграција на Amazon SNS со WebSocket и Amazon API Gateway за праќање на push известувања.

AWS Internet of Things

Важен дел од AWS е AWS IoT[18] - Amazon internet of things, која нуди сервиси за поврзување на уред со други уреди и AWS сервиси. Доколку даден уред се поврзе со AWS IoT, AWS IoT може истиот да го поврзе со сервидите што AWS ги нуди. Како можност што ја нуди е реакција на настани. AWS IoT Events ни помага директно да активираме активности во AWS SNS, пример, SNS да испрати push известување.

Сервисите на Amazon, и нивна комбинација, ни нудат различни решенија за push нотификации кои можат да бидат искористени зависно потребите на апликацијата која се развива.

Firebase Cloud Messaging (FCM)

Firebase Cloud Messaging (FCM) е услуга за испраќање единечни и повеќе push известувања до уреди со Android и iOS и до веб-апликации. Оваа услуга е новата верзија на GCM-Google Cloud Messaging. Исто така е бесплатна за кој било број на пораки.

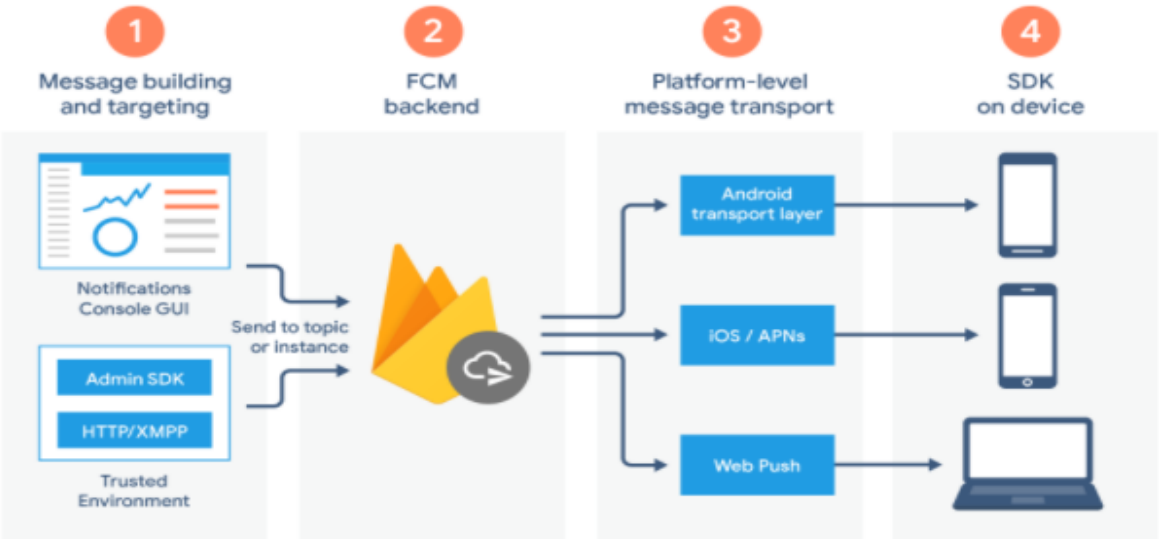
Архитектура на FCM

Архитектурата на FCM [18](Слика 8)се потпира на сет од компоненти за градење, пренос и примање на пораќи. Но најпрво, потребно е регистрирање на клиентската апликација за да прима пораки, со што добиваууникатен регистрациски токен. Компонирањето на пораќи се прави во сигурна околина која подржува FCM протоколи или Firebase SDK (анг. " Software Development Kit ", во превод комплет за развој на софтвер, кој претставува сет од алатки за креирање апликации специфични за платформа или систем).

Пораката потоа се праќа на FCM позадината,која (меѓу другите функции) прифаќа барања за пораки, извршува проследување пораки преку теми и генерира метаподатоци за пораки како што е ID на пораката и ја проследува пораката на следната компонента. Тоа е транспортниот слој на ниво на платформа, кој ја насочува пораката до целниот уред, се справува со испорака на порака и применува конфигурација специфична за платформата каде што е соодветно. Овој транспортен слој е надвор од FCM сервисите и вклучува:

- Андроид транспортен слој (анг. "ATL - Android transport layer") за уреди со Android со услуги на Google Play
- Сервис за известување на Apple Push (анг. "APN - Apple Push Notification service") за уреди со iOS
- Веб-push протокол за веб-апликации

Како последна компонента на која се потпира FCM за испорака на известување, е FCM SDK на уредот на корисникот, каде што се прикажува известувањето или се постапува со пораката според состојбата на апликацијата и некоја релевантна логика на апликацијата.



Слика 8

FCM Протоколи

FCM користи HTTP и XMPP протоколи и работи и со JSON и со обичен текст. Во моментов, FCM ги подржуваследниве сервер протоколи:

- FCM HTTP v1 API
- Наследниот протокол HTTP (анг. "Legacy HTTP protocol")
- Наследниот протокол XMPP (анг. "Legacy XMPP protocol")

Вашиот сервер за апликации може да ги користи овие протоколи одделно или заедно. Бидејќи е најсовремен и најфлексибилен за испраќање пораки до повеќе платформи, се препорачува FCM HTTP v1 API секаде каде е тоа можно.

FCM и WebSocket

Во повеќето модерни прелистувачи, Firebase комуницира со својот позадински дел преку WebSocket. Но, протоколот што го користи не е документиран[19] и може да се промени со текот на времето.

Алтернативно, би можеле да користите Firebase REST API, што е целосно документирано. Вклучува поддршка за праќање податоци во континуиран проток, што ви овозможува многу од предностите на имплементацијата на Web Socket без да се потпирате на недокументиран протокол.

OneSignal

OneSignal[21] е сервис што овозможува push известувања, апстрахирање детали како што е платформата на која работи уредот. Со додатокот OneSignal, апликациите можат да испраќаат и примаат известувања за push.

Интеграција со OneSignal

Потребно е да го конфигурирате OneSignal за секоја од мобилните платформи на кои ќе работи. Откако ќе ја завршите вашата конфигурација, ќе имате ID на апликација и API клуч од OneSignal. Може да го конфигурирате OneSignal за iOS и Android. За вашата апликација да добива известувања, треба да ја имплементирате логиката од страна на клиентот за да го регистрирате уредот во OneSignal. Логиката треба да се додаде на управувачите на настани (анг. "event handlers") и тоа:

- OnNotificationReceived: акција што работи кога апликацијата добива известување.
- OnNotificationOpened: акција што работи кога корисникот отвора известување.

OneSignal поддржува две различни основни технологии за клиенти кои сакаат да интегрираат push известувања на нивните веб-страници. Тоа се HTTPS SDK и HTTP SDK[22], каде OneSignal ја препорачува HTTPS SDK доколку веб-страницата ја подржува.

Изборот на соодветен сервис за push известување ќе ја олесни вашата стратегија за ангажман, бидејќи ќе биде лесно да ги поставите и испратите известувањата до вашите корисници. Некои од сервисите за push изветувања се попознати од другите, но сите имаат добра функционалност.

Важноста на push известувањата

За да се види колку е важна технологијата за push известувања, спроведено е истражување од страна на платформата за мобилен маркетинг Leanplum, специјализирана за ангажман на апликации. При ова истражување, покажано е декаpush известувањата се способни да ги зголемат продажбите до 10 пати. Податоците откриваат и дека во текот на една недела push известувања испратени во сабота можат да го удвојат бројот на купувања што луѓето ги прават со мобилни апликации. Исто така истражувањето покажало дека луѓето ќе купуваат повеќе откако ќе добијат Push известувања во попладневните часови од денот и тоа со 2,7 пати зголемување.

Не сите видови на push известувања предизвикуваат позитивен одговор од корисниците. Како развивачи на апликации треба да бидеме претпазливи во врска со користењето на push известувањата - користете ги за да комуницирате со своите корисници за информации што се корисни за нив, но не ја злоупотребувајте оваа технологија.

Имплементација на проект со FCM и Vue.js

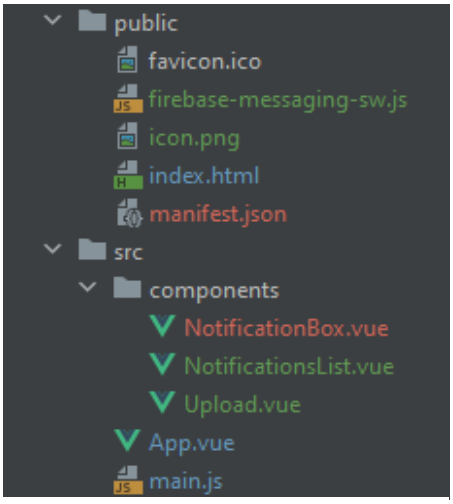
Vue.js е рамка за веб апликации (анг. "framework") што можеме да ја искористиме за развој на интерактивни апликации од предната страна. Во следниот проект, имаме имплементација на push известувања во Vue.js апликација, со помош на Firebase т.е. FCM. Исто така ќе ја разгледаме и употребата на библиотека за известувања во Vue за приказ на овие известувања, како и приказ на истите на наш т.е. custom начин .

Апликацијата треба да обезбеди функционалност за чување на датотеки, нивен преглед и бришење. При додавање и бришење на датотека, апликацијата ќе го извести корисникот преку push известување пратено додека веб-апликацијата е во преден план и кога истата е во позадина.

Структура на апликацијата

За потребите на апликацијата (Слика 9), се користат компонентите App.vue, main.js, icon.png како стандардна слика при приказ на известување, firebase-messaging-sw JavaScrit датотека за постаување на service worker, и креирани се три Vue компоненти и тоа:

- NotificationBox - компонента за custom приказ на push известувањето
- NotificationsList - компонента за custom групирање на NotificationBox компоненти
- Upload - компонента за работа со датотеки (поставување/зачувување, бришење, приказ на сите датотеки).



Слика 9

Соодветно компонентите се регистрирани во главната App.vue компонента.

Потребни библиотеки/пакети

За иработка на дадениот проект, потребни се неколку пакети да се симнат. Најпрво потребно е имаме npm кое претставува помошна алатка за менаџирање со пакети. Може да се симне заедно со Node.js на следниот линк https://www.npmjs.com/get-npm. Сите останати пакети може да се инсталираат преку наредбата npm install <потребниот пакет> и тоа се:

- npm install --global @vue/cli
- npm install --save firebase
- npm install --save bootstrap
- npm install --save vue-notification
- npm install --save axios

Потоа, со наредбата npm run serve во папката со апликацијата, истата ќе се покрене на http://localhost:8080 (бројот на портата може да биде различен зависно други процеси кои се започнати, притоа соодветно ќе биде прикажано во конзолата).

Работа со датотеки

За да можеме да поставуваме, прикажеме и да избришеме избрана датотека[23][24] најпрво потреби се елементи на корисничкиот интерфејс со помош на HTML. Соодветно ќе користиме input од тип file (елемент за внесување датотека), поврзано со копче Browse, копчиња соодветно именувани Upload (постави) и List(листа), како и Delete(избриши) копчиња соодветно за секоја датотека. За приказ на листата од датотеки, ќе користиме table елемент т.е. ќе ги прикажеме со помош на табела.

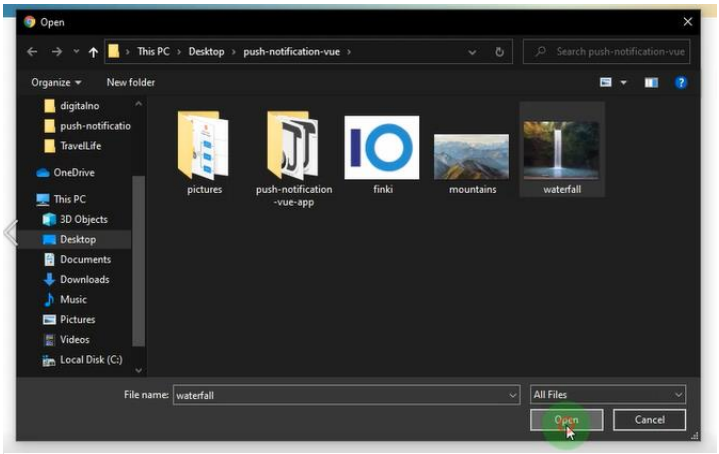
Најпрво, тоа што е прикажано е file input елементот, Upload копчето кое не е активирано и List копчето. (Слика 10)

Слика 10

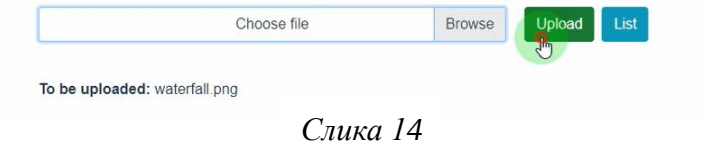
При кликнување на List копчето, доколку немаме прикачено датотеки ќе ни покаже соодветна порака на екранот. (Слика 11)

Слика 11

Слика 12



Слика 13

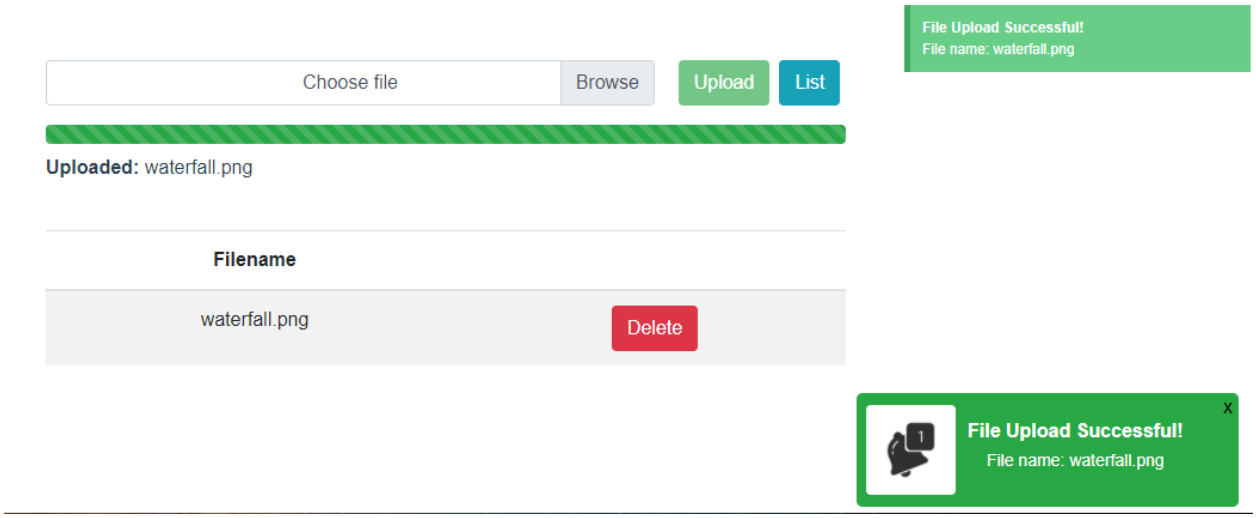


Слика 14

За да прикачиме датотека, најрво треба да избереме слика од нашиот уред и тоа го правиме со помош на file input елементот, со клкнување на копчето Browse (Слика 12). По ова ние имаме можност да одбереме датотека(Слика 13) која соодветно ќе биде подготвена за да се постави на сервер.

Преку кликнување на моментално активното Upload копче (Слика 14), се повикува методата onUpload() од Upload.vue компонентата и избраната датотека ќе се постави на сервер, прикажувајќи визуелено до каде е процесот, и соодветно ќе прикаже известување за дали прикаќувањето е успешно или не.

Во примерот на слика 15, прикачувањето е успешно, бојата на известувањето е зелена и насловот пораката гласи "File Upload Successful!". Доколку истото не беше успешно, бојата на известувањето ќе беше црвена и соодветна порака прикажана, "File Upload Cancelled!".

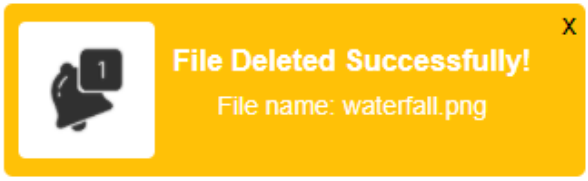


Слика 15

Приказот на известувањето е со 2 компоненти, една со употребата на библиотека за известувања во Vue за приказ на овие известувања т.е. vue-notification во горниот десен агол, додека , како и приказ на истите на custom начин во долниот десен агол.

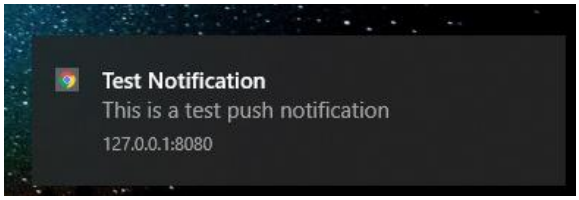
Со ова ние имаме прикачено слика, и доколку кликнеме на List копчето, преку повик на методата onList() од Upload.vue компонентата, во табела ќе бидат прикажани сите датотеки кои сме ги поставиле до тој момент, во овој момент тоа е само една. Во секој ред на табелата соодветно се прикажуваат името на датотеката и копче Delete, кое ја повикува методата onDeleteFile(filename) од Upload.vue компонентата, со аргумент filename за соодветното имена датотеката која сакаме да ја избришеме.

Слика 16



Слика 17

Преку кликнување на некое од Delete копчињата, соодветната датотека ќе биде избришана и ќе добиеме известување за истото. Доколку е успешно, приказот на известувањето ќе биде во жолта боја со наслов "File Deleted Successfully!" (Слика 16 со употреба на vue-notification и Слика 17 на custom начин), во спротивно со црвена боја и наслов "Can Not Delete File!".



Слика 18

Доколку веб-прелистувачот работи во позадина, push известувањата исто така ќе пристигнуваат со соодветна порака. (Слика 18)

Конекција со Firebase

За потребите на методите onUpload, on List и onDeleteFile, потребно е да имаме конекција до сервер каде што ќе ги чуваме датотеките. За ова ќе искотистиме Firebase cloud storage. Најпрво потребно е да иницијализираме Firebase во веб-апликацијата[25]. Соодветно, потребно е да се креира Firebase проект, за потребните информации можете да го посетите https://firebase.google.com/.

Со креиран Firebase проект, можеме да поврземе Firebase со Vue апликацијата со тоа што во main.js датотеката импортираме соодветни компоненти и иницијализираме Firebase проследувајќи ги соодветно параметрите од Firebase проект, и тоа API_KEY, PROJECT_ID, SENDER_ID, APP_ID соодветно. (Слика 19)

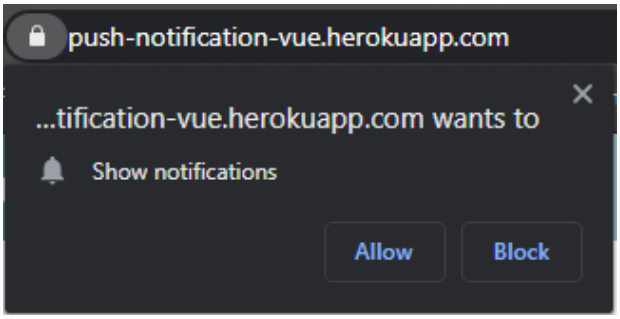
```
import firebase from 'firebase';
import 'firebase/messaging';

// // Initialize Firebase
firebase.initializeApp( options: {
    apiKey: "<API_KEY>",
    authDomain: "<PROJECT_ID>.firebaseapp.com",
    projectId: "<PROJECT_ID>",
    storageBucket: "<PROJECT_ID>.appspot.com",
    messagingSenderId: "<SENDER_ID>",
    appId: "<APP_ID>",
});
```

Слика 19

Push известувања

Регистрација на уред

Кога корисникот ќе ја овтвори веб-апликацијата, апликацијата е задолжена да го праша корисникот дали сака да прима известувања од страна на истата преку firebase.messaging(). Соодветната функционалност е поставена во методата subscribe() во комонентата Upload.vue и истата се повикува по креирањето на компонентата во created() куката (анг. "hook"). Соодветно на корисникот ќе му се појави прашање (Слика 20).



Слика 20

Доколку корисникот прифати, веб-апликацијата ќе добие токен како идентификатор и со истиот испраќа HTTP POST барање за да слуша т.е. да биде известена за промени што настануваат на дадена тема, во овој случај темата е именувана "all". За испраќање на HTTP барања се користи Axios, кој е HTTP клиент базиран на ветувања (анг. "promise"), лесен за употреба и се користи во веб-прелистувачи и JavaScript базирани аликации.

За креирање на соодветното барање потребно е да се постави дополнителен ред во заглавието за авторизација каде се проследува SERVER_KEY, добиен од Firebase проектот. Доколку корисникот не прифати известувања, или апликацијата не може да добие токен, или апликацијата не може да го регистрира уредот за да слуша за известувања, соодветни пораќи ќе бидат прикажани, во овој случај во конзолата. (Слика 21)

```
subscribe(){
  // Getting Notification Permission And Subscribing To A Topic
  try{
    let self=this;
    firebase.messaging().requestPermission().then(()=>{
      console.log("Notification permission granted.");
      return firebase.messaging().getToken()
          // Accessing Token Successfully
          .then((token)=>{
            self.token=token;
            // Subscribing To A Topic
            axios.post( url: "https://iid.googleapis.com/iid/v1/"+token+"/rel/topics/all",
                 data: null,
                 config: {
                  headers:{
                    'Authorization': "key=<SERVER_KEY>"
                  }
                }).then(response => {
              if (response.status < 200 || response.status >= 400) {
                throw 'Error subscribing to topic: '+response.status + ' - ' + response.text();
              }
              console.log('Subscribed to topic: all');
            }).catch(error => {
              // An error occurred
              console.error(error);
            });

          }).catch((err)=>{
            // An error occurred
            console.log("Unable to get token.",err);
          });
    });
  }catch (e){
    // An error occurred
    console.log(e);
  }
},
```

Слика 21

Испраќање на известувања

Во методите onUpload() и onDeleteFile(fileName), како што е споменато претходно, се наоѓа функционалноста за поставување и бришење на датотека соодветно. При комуникација со Firebase, може поставувањео на датотека да е успешно или да се појави грешка. Истото важи и при бришење на датотека. При овие два настани, соодветно сакаме да се прикаже push известување и за таа цел, повторно користејќи Axios, се испраќа HTTP POST барање до FCM за да бидат известени сите уреди кои се преплатени на "all" темата. Во овие барање се поставува повторно авторизациско заглавие, и тип на податоци што се испраќаат во телото на барањето. Во телото на барањето, се испраќа JSON објект со соодветно наведени делови за која тема и кои податоци. За функционалност на апликацијата избрани се наслов, порака, слика за приказ (иста за сите), и тип кој означува понатаму боја на известувањето.

Во продолжени се прикажани onUpload() методот и дополнителниот метод sendAxiosPostToNotify(title,file,status) (Слика 22 и 23). Во методот sendAxiosPostToNotify е поставена функционалноста за испраќање на барања до FCM, и поради повторување на истиот код на повеќе места, истата е соодветно извадена во посебна метода со три влезни параметри.

Методот onDeleteFile(fileName), како и целокупниот код, можете да го видите на https://github.com/veronikaognjanovska/push-notification-vue-public.

```
onUpload(){
  this.urlFileName=null;
  const storageRef = firebase.storage()
      .ref( path: `${this.fileData.name}`)
      .put(this.fileData);
  storageRef.on( event: `state_changed`,
     nextOrObserver: snapshot => {
      this.uploadValue=(snapshot.bytesTransferred/snapshot.totalBytes)*100;
    },
     error: () => {
      // An error occurred
      this.sendAxiosPostToNotify( title: "File Upload Cancelled!",this.urlFileName, status: "error");
    },
     complete: () => {
      // File uploaded successfully
      this.uploadValue=100;
      storageRef.snapshot.ref.getDownloadURL().then(url => {
        this.urlFileName=url.split('?')[0].split('/').slice(-1)[0];
        this.fileData=null;
        this.sendAxiosPostToNotify( title: "File Upload Successful!",this.urlFileName, status: "success");
        this.onList();
      });
    }
  );
  document.getElementById( elementId: "inputGroupFile01").value = "";
},
```

Слика 22

```
sendAxiosPostToNotify(title="error",file=null,status="error"){
  const data = {
    "to": "/topics/all",
    "data" : {
      "title": title,
      "message": "File name: "+file,
      "icon": "icon.png",
      "type":status
    }
  };
  // Sending Request To Notify Subscribers Of Specific Topic
  axios.post( url: "https://fcm.googleapis.com/fcm/send",data, config: {
    headers:{
      'Content-Type': "application/json",
      'Authorization': "key=<SERVER_KEY>"
    }
  });
},
```

Слика 23

Слушање за нови известувања

Кога на темата, на која претходно се имаме регистрирано да слушаме, ќе има нови пораки, серверот, FCM, ќе ни испрати push известување. За слушање на истите има два начина во зависност дали веб-прелистувачот со нашата веб-апликација е во преден или во заден план т.е. foreground или background.

Преден план - foreground

Доколку апликацијата е во преден план, слушањето на методи е преку onMessage методата од firebase messaging. За истото, оваа функционалност е имплементирана во NotificationsList.vue компонентата, во методот receiveMessage(). (Слика 24)

Кога ќе пристигне известување, се креира објект item со информациите кои се проследени, и објектот се поставува во листа notifications кои ги претставуваат моментално активните известувања кои се појавени на уредот на корисникот. Преку соодветни HTML елементи истите се претставени на custom начин во листа на долниот десен агол. Тука е вклучена NotificationBox.vue компонентата која претставува приказ на едно custom известување и соодветно е вклучена во делот components од NotificationsList.vue компонентата.

```
receiveMessage() {
  // Listen For Notifications
  try{
    let self = this;
    const messaging = firebase.messaging();
    messaging.onMessage( nextOrObserver: function(payload) {
      // With Custom Notification
      let item = {
        id:uniqueId( prefix: 'item-'),
        title:payload.data.title,
        message:payload.data.message,
        icon:payload.data.icon,
        type:payload.data.type
      };
      self.notifications.push(item);

      // Using Vue-notifications
      self.$notify( options: {
        group: 'notify',
        title:payload.data.title,
        text:payload.data.message,
        type:payload.data.type
      });

    });
  }catch (e) {
    console.log(e);
  }
}
```

Слика 24

Исто така, известувањето може да го претставиме со помош на библиотеката vue-notification, преку повик на $notify и соодветно испраќање на параметрите за група, наслов, текст и тип. Претходно треба да имаме вклучено Notifications од vue-notification во main.js и поставено ознаката во template делот од некоја компонента зависно структурата на компонентите, тука е во App.vue компонентата (Слика 25 и 26).

```
import Notifications from 'vue-notification';

// using Vue.js notifications
Vue.use(Notifications);
```

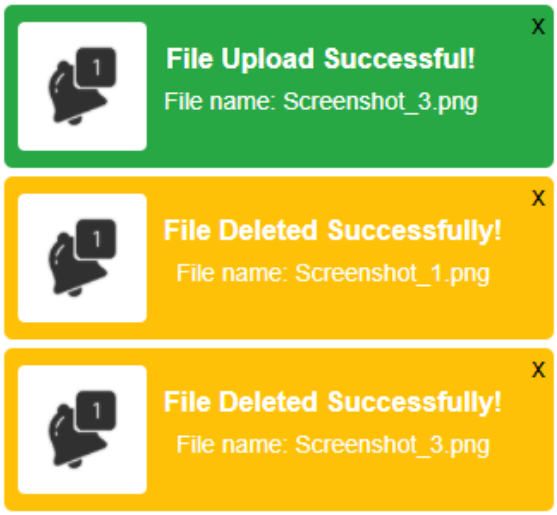
Слика 25

Оваа библиотека има и голем број други опции кои можат да се искористат. (истите може подетално да ги разгледат на https://www.npmjs.com/package/vue-notification).

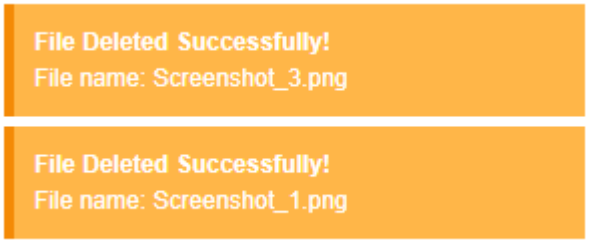
```
<notifications group="notify" style="margin-top: 70px" />
```

Слика 26

Прикажувањето на push известувањето во преден план е прикажано на слика 27 со custom начин и слика 28 со библиотеката vue-notification.



Слика 27

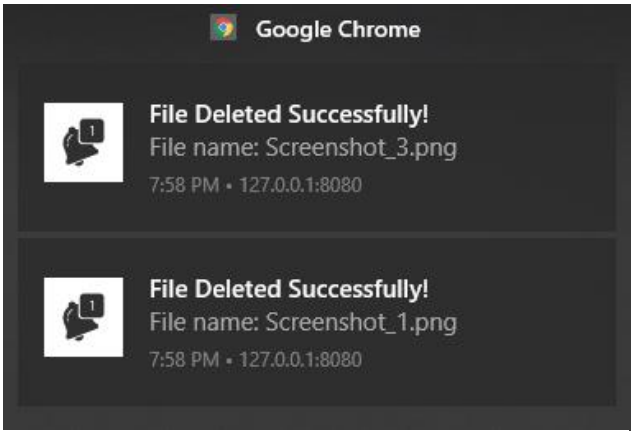


Слика 28

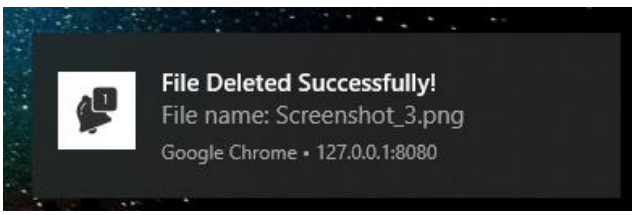
Заден план - background

Кога веб-апликацијата е во заден план, потребено е да имаме дефинирано service worker и да го регистрираме истиот. Со помош на скриптата firebase-messaging-sw.js, се поставуваат сите скрипти потребни за работа со Firebase (верзиите треба да се согласуваат со истите од package.json) и се иницијализира Firebase во service worker. Преку методата setBackgroundMessageHandler од firebase messaging се дефинира логиката при добивање на известување и приказ на истото преку showNotification(title,options) (Слика 29). Регистрацијата на service worker се повикува на navigator интерфејсот кој ја претставува состојбата и идентитетот на корисничкиот прелистувач, и се предава скриптата firebase-messaging-sw.js каде е дефинирана логиката на истиот (Слика 30).

Прикажувањето на push известувањето во заден план е прикажано на слика 31 и слика 32.



Слика 31



Слика 32

```
importScripts( urls: "https://www.gstatic.com/firebasejs/8.2.4/firebase-app.js");
importScripts( urls: "https://www.gstatic.com/firebasejs/8.2.4/firebase-messaging.js");


try{
  // Initialize the Firebase app in the service worker by passing in the
  // messagingSenderId.
  firebase.initializeApp( {
    apiKey: "<API_KEY>",
    projectId: "<PROJECT_ID>",
    appId: "<APP_ID>",
    messagingSenderId: "<SENDER_ID>"
  });
  // Retrieve an instance of Firebase Messaging so that it can handle background
  // messages.
  const messaging = firebase.messaging();
  messaging.setBackgroundMessageHandler(function (payload){
    // Customize notification here
    const title = payload.data.title;
    const options = {
      body:payload.data.message,
      icon:payload.data.icon
    };
    // Show notification
    return self.registration.showNotification(title,options);
  });

}catch (err){
  console.log(err);
}
```

Слика 29

```
navigator.serviceWorker.register( scriptURL: 'firebase-messaging-sw.js',
      options: {scope: "firebase-cloud-messaging-push-scope"})
    .then((registration : ServiceWorkerRegistration ) => {
      const messaging = firebase.messaging();
      messaging.useServiceWorker(registration);

    }).catch(err => {
   console.log(err)
})
```

Слика 30

Заклучок

Технологијата за push известувања ни дозволува на лесен и брз начин да се поврземе со голем број од корисниците и како таква е значаен дел во креирањето на една апликацијата. Можноста со само две три реченици да можеме да го заинтригираме корисникот да ја отвори нашата апликација повторно е важна стратегија за поголема вклученост на корисниците.

Се надевам дека ова истражување ќе ви помогне да добиете подобра слика за што се тоа push известувања, која е технологијата и концептот позади истите како и за нивното користење. Сметам дека во време на дигитализација кога сите сме на нашите мобилни, таблети, компјутери, важно е, особено како развивачи на апликации, па дури и само како корисници на истите, да имаме основни познавања за можностите што push известувањата ни ги нудат.

Користена литература

1. https://buildfire.com/what-is-a-push-notification/
2. https://www.codingtag.com/basics-of-web-push-notifications-for-beginners
3. https://mobileroadie.com/blog/2019/10/what-is-push-notification/
4. https://en.wikipedia.org/wiki/Push_technology
5. https://uxdesign.cc/push-and-poll-f898a910b4aa
6. https://www.airship.com/resources/explainer/push-notifications-explained/#:~:text=For%20app%20publishers%2C%20push%20notifications,app%20is%20open%20or%20not.
7. http://www.mobileui.org/innovaportal/v/181/1/innova.front/push-notifications:-knowing-the-technology.html
8. https://developer.mozilla.org/en-US/docs/Web/API/Push_API
9. https://developers.google.com/web/ilt/pwa/introduction-to-push-notifications
10. https://thenextscoop.com/push-notifications-websites-different-mobile-apps-push-notifications/
11. https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol
12. https://www.youtube.com/watch?v=8ARodQ4Wlf4
13. https://medium.com/swlh/building-a-browser-push-notification-service-http-long-polling-and-the-web-socket-protocol-5e83cd1420c1
14. https://steelkiwi.com/blog/push-notifications-services-how-to-choose/
15. https://aws.amazon.com/sns/?whats-new-cards.sort-by=item.additionalFields.postDateTime&whats-new-cards.sort-order=desc
16. https://iwconnect.com/websocket-apis-in-amazon-api-gateway-push-notifications/
17. https://searchaws.techtarget.com/tip/AWS-WebSocket-support-ushers-in-real-time-serverless-features
18. https://docs.aws.amazon.com/iotevents/latest/developerguide/iotevents-dg.pdf
19. https://firebase.google.com/docs/cloud-messaging/fcm-architecture
20. https://stackoverflow.com/questions/48734290/javascript-websocket-into-firebase
21. https://documentation.onesignal.com/docs/web-push-http-vs-https
22. https://success.outsystems.com/Documentation/How-to_Guides/Integrations/How_to_Use_Push_Notifications_with_OneSignal
23. https://www.youtube.com/watch?v=wLel659TyFE
24. https://awesomeopensource.com/project/invokemedia/vue-push-notification-example
25. https://mabbkhawaja.medium.com/add-firebase-push-notifications-to-vuejs-quasar-app-4ac87683fe37