Laravel Web Push Notifications

Requirements:

1. PHP >= 5.6.4
2. Laravel >= 5.4
3. Enable GMP php extension in the .ini file.

Sources:

1. Youtube Tutorial: <https://www.youtube.com/watch?v=Fh8fpTXnLXg>
2. Git Source: <https://github.com/laravel-notification-channels/webpush>

Steps:

1. Create Laravel Project:
2. Generate authentication (For per user notifications)
3. Create the .js source code for notifications and service-worker.js:

<script>

function sendNotification(){

var data = new FormData();

data.append('title', document.getElementById('title').value);

data.append('body', document.getElementById('body').value);

var xhr = new XMLHttpRequest();

xhr.open('POST', "{{url('/api/send-notification/'.auth()->user()->id)}}", true);

xhr.onload = function () {

// do something to response

console.log(this.responseText);

};

xhr.send(data);

}

var \_registration = null;

function registerServiceWorker() {

return navigator.serviceWorker.register('js/service-worker.js')

.then(function(registration) {

console.log('Service worker successfully registered.');

\_registration = registration;

return registration;

})

.catch(function(err) {

console.error('Unable to register service worker.', err);

});

}

function askPermission() {

return new Promise(function(resolve, reject) {

const permissionResult = Notification.requestPermission(function(result) {

resolve(result);

});

if (permissionResult) {

permissionResult.then(resolve, reject);

}

})

.then(function(permissionResult) {

if (permissionResult !== 'granted') {

throw new Error('We weren\'t granted permission.');

}

else{

subscribeUserToPush();

}

});

}

function urlBase64ToUint8Array(base64String) {

const padding = '='.repeat((4 - base64String.length % 4) % 4);

const base64 = (base64String + padding)

.replace(/\-/g, '+')

.replace(/\_/g, '/');

const rawData = window.atob(base64);

const outputArray = new Uint8Array(rawData.length);

for (let i = 0; i < rawData.length; ++i) {

outputArray[i] = rawData.charCodeAt(i);

}

return outputArray;

}

function getSWRegistration(){

var promise = new Promise(function(resolve, reject) {

// do a thing, possibly async, then…

if (\_registration != null) {

resolve(\_registration);

}

else {

reject(Error("It broke"));

}

});

return promise;

}

function subscribeUserToPush() {

getSWRegistration()

.then(function(registration) {

console.log(registration);

const subscribeOptions = {

userVisibleOnly: true,

applicationServerKey: urlBase64ToUint8Array(

"{{env('VAPID\_PUBLIC\_KEY')}}"

)

};

return registration.pushManager.subscribe(subscribeOptions);

})

.then(function(pushSubscription) {

console.log('Received PushSubscription: ', JSON.stringify(pushSubscription));

sendSubscriptionToBackEnd(pushSubscription);

return pushSubscription;

});

}

function sendSubscriptionToBackEnd(subscription) {

return fetch('/api/save-subscription/{{Auth::user()->id}}', {

method: 'POST',

headers: {

'Content-Type': 'application/json'

},

body: JSON.stringify(subscription)

})

.then(function(response) {

if (!response.ok) {

throw new Error('Bad status code from server.');

}

return response.json();

})

.then(function(responseData) {

if (!(responseData.data && responseData.data.success)) {

throw new Error('Bad response from server.');

}

});

}

function enableNotifications(){

//register service worker

//check permission for notification/ask

askPermission();

}

registerServiceWorker();

</script>

1. Create the service-worker.js file:

self.addEventListener('push', function(event) {

if (event.data) {

var data = event.data.json();

self.registration.showNotification(data.title,{

body: data.body,

icon: data.icon

});

console.log('This push event has data: ', event.data.text());

} else {

console.log('This push event has no data.');

}

});

1. Install helper for laravel web push notifications:

composer require laravel-notification-channels/webpush

1. First you must install the service provider (skip for Laravel>=5.5):

// config/app.php

'providers' => [

...

NotificationChannels\WebPush\WebPushServiceProvider::class,

],

1. Add the NotificationChannels\WebPush\HasPushSubscriptions trait to your User model(For per user notif):

use NotificationChannels\WebPush\HasPushSubscriptions;

class User extends Model

{

use HasPushSubscriptions;

}

1. Next publish the migration with:

php artisan vendor:publish --provider="NotificationChannels\WebPush\WebPushServiceProvider" --tag="migrations"

1. Run the migrate command to create the necessary table:

php artisan migrate

1. Generate the VAPID keys (required for browser authentication) with:

php artisan webpush:vapid

1. Make a new notification by running the command:

Php artisan make:notification GenericNotification

1. Add the following lines in app/Notifications/GenericNotifiaction:

use NotificationChannels\WebPush\WebPushMessage;

use NotificationChannels\WebPush\WebPushChannel;

public function via($notifiable)

{

// return ['mail'];

return [WebPushChannel::class];

}

public function toWebPush($notifiable, $notification)

{

$time = \Carbon\Carbon::now();

return (new WebPushMessage)

// ->id($notification->id)

->title($this->title)

->icon(url('/push.png'))

->body($this->body);

//->action('View account', 'view\_account');

}

1. Add the API route:

Route::post('/save-subscription/{id}',function($id, Request $request){

$user = \App\User::findOrFail($id);

$user->updatePushSubscription($request->input('endpoint'), $request->input('keys.p256dh'), $request->input('keys.auth'));

$user->notify(new \App\Notifications\GenericNotification("Welcome To WebPush", "You will now get all of our push notifications"));

return response()->json([

'success' => true

]);

});

Route::post('/send-notification/{id}', function($id, Request $request){

$user = \App\User::findOrFail($id);

$user->notify(new \App\Notifications\GenericNotification($request->title, $request->body));

return response()->json([

'success' => true

]);

});

1. Add code to GenericNotification to add notifications title and body as properties.

public $title, $body;

/\*\*

\* Create a new notification instance.

\*

\* @return void

\*/

public function \_\_construct($title, $body)

{

//

$this->title = $title;

$this->body = $body;

}