LARAVEL
REALTIME APP

Panduan Praktis Membuat Realtime App dengan Laravel Websockets

MUHAMMAD AZAMUDDIN

Panduan Praktis Membuat Realtime App dengan Laravel Websockets

Apakah ebook ini gratis?

Ya, dan kamu boleh membagikan ke siapapun, dan yap, silahkan bagikan jika bermanfaat. Tidak diperbolehkan mengubah sebagian atau keseluruhan konten di ebook ini tanpa seizin penulis.

Link source code ada di bagian penutup ya. Saya tahu ada yang pertama dicari adalah source code.

Tentang penulis

Muhammad Azamuddin, Fullstack Developer, lulusan akuntansi bukan IT, penulis ebook Be Fullstack Developer Laravel - Vue (https://buku-laravel-vue.com) bersama Hafid Mukhlasin yang sudah dipercaya oleh 1200+ member. Tech Speaker, meskipun udah lama off karena kesibukkan dengan anak 😄 kalo mau mengundang tetep dipersilahkan.

Senang dengan Laravel, Nodejs, Sails, React tapi tidak menutup diri menggunakan teknologi-teknologi lainnya jika diperlukan. Pernah bekerja di beberapa tempat, freelance writer di media Brazil, bekerja untuk Republik Indonesia, remote di Keller Williams (tapi terpaksa dilepas supaya "fokus" meskipun sempet galau banget), creator produk-produk digital yang sudah terjual lebih dari 5000 items di Indonesia dan Malaysia bersama rekannya, Nofi Bayu Darmawan. Tapi tetep newbie dibandingkan para master, dan selalu butuh waktu buat belajar dan mengembangkan diri. Barakallah.

Facebook: https://facebook.com/script.holic
Email: mas.azamuddin@gmail.com

Setelah membaca ini pembaca diharapkan bisa apa?

Setelah membaca ebook ini pembaca akan memperoleh pengetahuan bagaimana membuat fitur realtime dengan Laravel dan Vue tanpa memanfaatkan package laravel-websockets. Selain itu pembaca akan bisa membuat komponen vue yang bereaksi terhadap event di laravel serta bisa menggunakan chartjs dengan vue.

Kenapa tutorial realtime app?

Oktober 2018 saya dan partner sekaligus guru saya, Hafid Mukhlasin merilis ebook bundle Laravel + Vue yang membahas tuntas dari konsep, study kasus hingga deploy ke server VPS dan hosting.

Selain materi ebook, kami juga menyiapkan grup khusus pembeli (meskipun banyak penyusup masuk, hehe), di grup itu sudah banyak tanya jawab dan diskusi terkait materi secara langsung maupun topik laravel vue secara umum.

Kemudian kami berpikir untuk memberikan tambahan materi terutama untuk para pembeli, lalu kami putuskan untuk melakukan polling di grup, kira-kira materi apa yang paling diinginkan? yang paling banyak vote yang menang. Dan ternyata materi realtime app ini lah yang unggul mengalahkan opsi-opsi lain.

Ebook ini cocok untuk siapa?

Ebook ini bukan untuk yang baru mau belajar Laravel apalagi yang baru mau belajar pemrograman. Meskipun bisa aja kalo mau maksain ngikutin. Tapi saya sarankan belajar dulu sampai bisa dasar Laravel.

Paling cocok ebook ini untuk yang sudah paham Laravel, adapun untuk Vue sebenarnya untuk yang paham dasar-dasarnya sudah cukup, karena tidak akan dibahas topik-topik advanced.

Topik apa saja yang akan kita pelajari di ebook ini?

Kita tidak akan membahas secara detail masing-masing topik di bawah ini. Karena tujuan dari ebook ini adalah untuk memperkenalkan / menunjukkan bagaimana sih setup realtime app dengan Laravel & Vue. Bukan menjelaskan fundamental dari kedua teknologi tersebut. Sesuai dengan poin "Ebook ini cocok untuk siapa?".

Berikut hal-hal yang akan kita pelajari di ebook ini:

- Laravel Websockets Package
- Laravel Echo
- Laravel Mix
- Komponen Vue
- Ajax
- REST endpoints dengan Laravel

System Requirements

- PHP versi 7.1 ke atas
- Node + NPM

Apakah saya perlu Node untuk socket? umumnya saat akan menggunakan websocket kita selalu disuguhkan tutorial-tutorial socket server dengan Node, tapi tidak di ebook ini, kita akan menggunakan `Laravel Web Socket` sebagai socket server untuk mengganti Pusher cloud. Adapun Node + NPM digunakan untuk compile VueJS dengan menggunakan Laravel Mix.

Setup project Laravel baru

1. Buat project laravel baru

```
laravel new realtime-feedback
```

2. Copy `.env-example` ke `.env` (Linux or Mac)

```
cp .env-example .env
```

3. Buka phpmyadmin kamu, lalu buat database baru dengan nama `realtime-feedback`

4. Bukan file `.env` yang tadi lalu sesuaikan konfigurasi database.

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=realtime-feedback
DB_USERNAME=USERNAME_MYSQLMU
DB_PASSWORD=supersecretpasswordKAMU
```

5. jalankan composer dumpautoload

```
composer dumpautoload
```

6. Isi API_KEY

   Buka kembali `.env` lalu ubah API_KEY

```
```

```
API_KEY=
```

menjadi

```
API_KEY=somerandomSECRET
```

7. Nyalakan server, contoh ini menggunakan built in PHP server. Kamu juga bisa menggunakan xampp, Docker atau yang lainnya sesuai setup di komputermu.

```
php -S localhost:9000 -t public
```

Membuat migration untuk tabel feedback

Langkah pertama setelah membuat project laravel baru adalah membuat migration untuk mendefinisikan struktur tabel yang akan kita gunakan.

```
php artisan make:migration create_feedback_table
```

Selanjutnya buka file migration yang baru dibuat di `database/migrations/xxxxxx_create_feedback_table.php`

Lalu jadikan seperti ini:

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateFeedbackTable extends Migration
```

```
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('feedback', function
(Blueprint $table) {
            $table->bigIncrements('id');
            $table->string("word");
            $table->integer("count");
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('feedback');
    }
}
```

Setelah itu jalankan perintah migrate.

```
php artisan migrate
```

Buat model & controller Feedback

Setelah membuat tabel dengan fitur migration, selanjutnya kita buat model Feedback

```
php artisan make:model Feedback
```

kemudian buka model `Feedback` yang baru saja kita *generate*, tambahkan field `word` dan `count` sebagai `$fillable`

```
class Feedback extends Model
{
    protected $fillable = ['word', 'count'];
}
```

Setelah itu buatlah controller `FeedbackController`

```
php artisan make:controller FeedbackController
```

Lalu jadikan seperti ini:

```
<?php

namespace App\Http\Controllers;

use App\Events\FeedbackReceived;
use App\Feedback;
use Illuminate\Http\Request;

class FeedbackController extends Controller
{
```

```php
    /**
     * Store a newly created resource in
storage.
     *
     * @param  \Illuminate\Http\Request
$request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
        $words = $request->get('words');

        // tidak boleh mengandung tanda koma (,)
        if (strpos($words, ",") !== false) {
            return response()->json([
                "message" => "Kata tidak boleh
ada tanda koma (,)",
            ], 400);
        }

        $words = explode(" ", $words);

        // tidak boleh lebih dari 3 kata
        if (count($words) > 3) {
            return response()->json([
                "message" => "Tidak boleh lebih
dari 3 kata",
            ], 400);
        }

        foreach ($words as $key => $word) {
            $this->createOrIncrement($word);
        }

        return response()->json("OK");
    }

    public function dashboard()
    {
        return view('dashboard');
    }
```

```
    public function dashboardData()
    {
        return response()->json($this->getData());
    }

    protected function getData()
    {
        $top_ten = Feedback::orderBy('count', 'DESC')
            ->get()
            ->take(10);

        return $top_ten;
    }

    public function input()
    {
        return view('input');
    }

    protected function createOrIncrement(String $word)
    {
        // jadikan lowercase
        $word = strtolower($word);

        $feedback = Feedback::where('word', $word)->first();

        if ($feedback) {
            $feedback->increment('count');
        } else {
            Feedback::create([
                "word" => $word,
                "count" => 1,
            ]);
        }
    }
```

```
}
```

Routes

Setelah tabel, model dan controller siap, selanjutnya kita definisikan route agar user dapat mengakses data dan fitur aplikasi kita. Tambahkan routes/web.php kode berikut

```
Route::group(['prefix'=> 'api/v1'], function(){
    Route::post('feedback',
'FeedbackController@store');
});
```

Setelah itu kita juga perlu melakukan disable csrfToken middleware untuk route dengan awalan "api/v1". Caranya buka `app/Http/Middleware/VerifyCsrfToken.php` kemudian tambahkan route prefix tadi di variable `$except` seperti ini:

```
protected $except = [
    "api/v1/*",
];
```

Install dan Setup Laravel Websockets Package

- install laravel web socket

  ```
  composer require beyondcode/laravel-
  websockets
  ```

- publish migration dari laravel-websockets

```
php artisan vendor:publish --
provider="BeyondCode\LaravelWebSockets\WebS
ocketsServiceProvider" --tag="migrations"
```

- php artisan migrate
- publish konfigurasi bawaan dari package laravel web socket

```
php artisan vendor:publish --
provider="BeyondCode\LaravelWebSockets\WebS
ocketsServiceProvider" --tag="config"
```

Broadcasting laravel

- install Pusher SDK

```
composer require pusher/pusher-php-server
"~3.0"
```

Kenapa kita menginstall pusher? karena kita akan memanfaatkan laravel websocket sebagai pusher server, bukan menggunakan pusher cloud, jadi ga perlu langganan dan ga ada batasan.

- ganti `BROADCAST_DRIVER` menjadi pusher di `.env`
- ubah `config/broadcasting.php`

```
            'options' => [
             'cluster' =>
env('PUSHER_APP_CLUSTER'),
             'encrypted' => false,
             'host' => '127.0.0.1',
```

```
                'port' => 6001,
                'scheme' => 'http',
            ],
```

- ubah config di `.env`

```
PUSHER_APP_ID=realtime-feedback
PUSHER_APP_KEY=pusherKey
PUSHER_APP_SECRET=pusherSecret
PUSHER_APP_CLUSTER=mt1

MIX_PUSHER_APP_KEY=pusherKey
MIX_PUSHER_APP_CLUSTER=mt1
```

- buka `resources/bootstrap.js` lalu uncomment kode berikut untuk mengaktifkan Laravel Echo .

```
import Echo from "laravel-echo";

window.Pusher = require("pusher-js");

window.Echo = new Echo({
    broadcaster: "pusher",
    key: "your-pusher-key",
    encrypted: false,
    wsHost: window.location.hostname,
    wsPort: 6001,
    disableStats: true
});
```

PENTING: Pastikan broadcaster `pusher` dan key diisi nilai yang sama dengan `MIX_PUSHER_APP_KEY` di `.env`. Selain itu kita perlu menambahkan `encrypted:false,`

- jalankan laravel websocket dengan membuka terminal baru di project kita, lalu ketik perintah berikut

```
php artisan websockets:serve
```

- Uncomment broadcasting provider di `app/config.php`

```
 App\Providers\BroadcastServiceProvider::cl
ass,
```

- buat Event baru dengan nama `FeedbackReceived`, ketik perintah

```
php artisan make:event FeedbackReceived
```

- buka event `FeedbackReceived` lalu sesuaikan kodenya seperti ini:

```
    <?php

    namespace App\Events;

    use Illuminate\Broadcasting\Channel;
    use
Illuminate\Broadcasting\InteractsWithSocket
s;
    use
Illuminate\Foundation\Events\Dispatchable;
    use Illuminate\Queue\SerializesModels;

    class FeedbackReceived
    {
        use Dispatchable,
InteractsWithSockets, SerializesModels;
```

```
        /**
         * Create a new event instance.
         *
         * @return void
         */
        public function __construct($data)
        {
            $this->payload = $data;
        }

        public function broadcastWith()
        {
            return $this->payload;
        }

        /**
         * Get the channels the event should
broadcast on.
         *
         * @return
\Illuminate\Broadcasting\Channel|array
         */
        public function broadcastOn()
        {
            return new Channel('feedback-
received');
        }
    }
```

- tambahkan broadcast di `FeedbackController.store`

```
    // broadcast with new data
    $data = json_decode($this->getData());
    broadcast(new FeedbackReceived($data));
```

sehingga method `store` di `FeedbackController` menjadi seperti ini:

```
public function store(Request $request)
{
    $words = $request->get('words');

    // tidak boleh mengandung tanda koma
(,)
    if (strpos($words, ",") !== false) {
        return response()->json([
            "message" => "Kata tidak boleh
ada tanda koma (,)",
        ], 400);
    }

    $words = explode(" ", $words);

    // tidak boleh lebih dari 3 kata
    if (count($words) > 3) {
        return response()->json([
            "message" => "Tidak boleh lebih
dari 3 kata",
        ], 400);
    }

    foreach ($words as $key => $word) {
        $this->createOrIncrement($word);
    }

    // broadcast with new data
    $data = json_decode($this->getData());
    broadcast(new FeedbackReceived($data));

    return response()->json("OK");
}
```

Penting: pastikan `FeedbackController` memiliki kode ini `use App\Events\FeedbackReceived;` jika belum tambahkan.

Laravel Mix

Laravel sudah menyediakan konfigurasi untuk memudahkan kita melakukan build file-file javascript untuk client side. Pertama yang harus kita lakukan adalah menginstall *dependency* yang tertera di file `package.json`, caranya ketik perintah ini:

npm install

Node dan NPM harus sudah diinstall ya.

Selanjutnya kita akan menginstall dependency yang belum ada di bawaan `package.json` laravel, yaitu `laravel-echo` dan `pusher-js`, install dengan perintah ini:

```
npm install laravel-echo pusher-js --save
```

Jika sudah, buka terminal baru di project kita untuk kepentingan develop, dan jalankan perintah watch

```
npm run watch
```

perintah watch ini akan mengamati file-file javascript kita, jika ada perubahan maka akan otomatis melakukan compile sehingga asset hasil compile akan selalu update dengan perubahan yang kita lakukan. Ini berguna disaat development biar kita tidak manual compile setiap kali mengubah file javascript, termasuk file .vue ya.

User Interface

- Pertama kita harus uncomment line di `resources/js/app.js`

```
const files = require.context("./", true,
/\.vue$/i);
files.keys().map(key =>
    Vue.component(
        key
            .split("/")
            .pop()
            .split(".")[0],
        files(key).default
    )
);
```

Kode di atas berfungsi agar component Vue yang kita buat auto registered dan bisa dipakai.

UI Input feedback

- Buat routes untuk input feedback diluar prefix `api/v1`

```
Route::get('feedback/input',
'FeedbackController@input');
```

- Buat `input` method di `FeedbackController` jika belum ada.

```
public function input(){
    return view('input');
}
```

- Buat view input di `resources/views/input.blade.php`

```
<html>
    <head>
        <title>Feedback Loop</title>
        <link rel="stylesheet"
href="/css/app.css" />
    </head>
    <body>
        <div id="app"></div>
        <script src="/js/app.js"></script>
    </body>
</html>
```

- Buat vue component `feedback-input` di `resources/js/components/FeedbackInput.Vue`

```
<template>
    <div class="container">
        <div v-if="status != 'SUCCESS'">
            <div class="row mt-5">
                <div class="col-md-8
offset-md-2 text-center">
                    <h1>
                        Gambarkan dengan 3
kata, Bagaimana Visi anda terhadap
                        BPN pada tahun
2020.
                    </h1>
                </div>
            </div>

            <div class="row mt-5">
                <div class="col-md-6
offset-md-3">
                    <input
                        type="text"
                        class="form-control
form-control-lg"
```

```
placeholder="contoh: keren top mantab"
                        v-
bind:value="feedback"
                        v-
on:input="feedback = $event.target.value"
                    />
                    <br />
                    <button
                        :disabled="status
== 'PROCESSING'"
                        v-on:click="submit"
                        class="btn btn-
primary btn-block btn-lg"
                    >
                        Submit
                    </button>
                    <br />
                    <div v-if="status ==
'ERROR'" class="alert alert-danger">
                        {{ message }}
                    </div>
                </div>
            </div>
        </div>

        <div class="mt-5" v-if="status ==
'SUCCESS'">
            <div class="row">
                <div class="col-md-6
offset-md-3">
                    <div class="alert
alert-success">{{ message }}</div>
                </div>
            </div>
        </div>
    </div>
</template>

<script>
import axios from "axios";
```

```
export default {
    data() {
        return {
            feedback: "",
            status: "IDLE", // IDLE |
SUBMITTING | ERROR | SUCCESS
            message: ""
        };
    },
    methods: {
        submit: function() {
            this.$data.status =
"SUBMITTING";

            axios
                .post("/api/v1/feedback", {
                    words:
this.$data.feedback
                })
                .then(response => {
                    this.$data.status =
"SUCCESS";
                    this.$data.message =
"Terima kasih atas partisipasi Anda!";
                })
                .catch(error => {
                    this.$data.status =
"ERROR";

                    if
(error.response.data.message) {
                        this.$data.message
= error.response.data.message;
                    } else {
                        this.$data.message
=
                            "Terjadi
kesalahan saat menyimpan feedback";
                    }
                });
```

```
            }
        },
        mounted() {}
};
</script>
```

- Gunakan component feedback-input di view `input.blade.php`

```
<html>
    <head>
        <title>Feedback Loop</title>
    </head>
    <body>
        <div id="app">
            <feedback-input></feedback-
input>
        </div>
        <script src="/js/app.js"></script>
    </body>
</html>
```

UI Dashboard

- Buat routes untuk dashboard di luar group `api/v1`

```
Route::get('feedback/dashboard',
'FeedbackController@dashboard');
```

- buat method `dashboard` di FeedbackController jika belum ada.

```
public function dashboard()
{
    return view('dashboard');
}
```

- buat view di `resources/views/dashboard.blade.php`

```
<html>
    <head>
        <title>Feedback Loop</title>
        <link rel="stylesheet"
href="/css/app.css" />
    </head>
    <body>
        <div id="app">
            <feedback-dashboard></feedback-
dashboard>
        </div>
        <script src="/js/app.js"></script>
    </body>
</html>
```

- buat method di controller feedback jika belum ada

```
public function dashboardData()
{
    return response()->json($this-
>getData());
}
```

- buat routes untuk mengambil data via API di dalam group `api/v1`

```
Route::get('feedback/data',
'FeedbackController@dashboardData');
```

ChartJS

- install vue-chartjs

```
npm install vue-chartjs chart.js --save
```

- buat vue component PieChart di `resources/js/components/PieChart.vue`

```
<script>
import { Pie, mixins } from "vue-chartjs";

export default {
extends: Pie,
props: ["chartData", "options"],
mixins: [mixins.reactiveProp],
mounted() {
    this.renderChart(this.chartData,
this.options);
}
};
</script>
```

- buat feedback-dashboard vue component di `resources/js/components/FeedbackDashboard.vue`
  Sebelumnya install google-palette terlebih dahulu

```
install google-palette
```

Sesuaikan kode `FeedbackDashboard.vue` menjadi seperti ini:

```
<template>
<div class="container">
    <div class="row mt-5">
    <div class="col-md-12">
        <h1 class="text-center">Top
```

```
Word</h1>
        <hr>
    </div>
    </div>
    <div class="row mt-5">
    <div class="col-md-5 offset-md-2">
        <pie-chart :chart-
data="piechartData"></pie-chart>
    </div>
    <div class="col-md-5 text-left">
        <ul style="list-style:
none;padding:0">
        <li v-bind:key="index" v-for="
(feedback, index) in data">
            <h1 v-if="index == 0">
{{feedback.word.toUpperCase()}}
({{feedback.count}})</h1>
            <h2 v-if="index == 1">
{{feedback.word.toUpperCase()}}
({{feedback.count}})</h2>
            <h3 v-if="index == 2">
{{feedback.word.toUpperCase()}}
({{feedback.count}})</h3>
            <h4 v-if="index > 2">
{{feedback.word.toUpperCase()}}
({{feedback.count}})</h4>
        </li>
        </ul>
    </div>
    </div>
</div>
</template>


<script>
import axios from "axios";
import palette from "google-palette";

export default {
data() {
    return {
```

```
    data: [],
    status: "IDLE", // FETCHING | IDLE |
ERROR
    message: ""
    };
},
computed: {
    piechartData: function() {
    return {
        datasets: [
        {
            data: this.$data.data.map(a =>
a.count),
            backgroundColor: palette(
            ["qualitative"],
            this.$data.data.length
            ).map(function(hex) {
            return "#" + hex;
            })
        }
        ],
        labels: this.$data.data.map(a =>
a.word.toUpperCase())
    };
    }
},
mounted() {
    this.fetchInitialData();
    this.listenForChange();
},
methods: {
    fetchInitialData() {
    axios
        .get("/api/v1/feedback/data")
        .then(response => {
        this.$data.data = response.data;
        this.$data.status = "SUCCESS";
        })
        .catch(error => {
        this.$data.status = "ERROR";
        });
```

```
    },
    listenForChange() {
    window.Echo.channel("feedback-
received").listen(
        "FeedbackReceived",
        payload => {
        this.$data.data = payload;
        }
    );
    }
}
};
</script>
```

Penutup

Demikian panduan praktis dan singkat ini saya buat. Semoga dapat bermanfaat untuk pembaca yang budiman. Mohon maaf jika ada error atau kekeliruan atau salah kata, jika ada pertanyaan atau kekeliruan yang ingin disampaikan, hubungi aja via facebook, https://facebook.com/script.holic.

Source code bisa dilihat di github.com/azamuddin/feedback-loop

Terakhir, jika kamu ingin memperdalam Laravel dan Vue, langsung aja cek https://buku-laravel-vue.