PWA - VUE JS

# PWA Requirements

## Progressive

Work for every user, regardless of browser choice because they’re built with progressive enhancement as a core tenant.

## Responsive

Fit any form factor, desktop, mobile, tablet, or whatever is next.

## App like

Use the app shell model to provide app-style navigations and interactions.

## Connectivity independent

Enhanced with service workers to work offline or on low quality networks.

## Easily installable

Allow users to “keep” apps they find most useful on their home screen without the hassle of an app store.

## Linkable

Easily share via URL and not require complex installation.

## Discoverable

Are identifiable as “applications” thanks to W3C manifests and service worker registration scope allowing search engines to find them.

## Benefit from native features

Notifications, use camera etc…

## Fast & fresh data

Always up-to-date thanks to the service worker update process.

## Safe

Served via TLS to prevent snooping and ensure content hasn’t been tampered with.

## Engaging

Engage the user through notification

# Install VUE Template Project

$ vue init pwa cropchat

$ cd cropchat  
$ npm install  
$ npm run dev

# Run the app

$ npm run dev

## Access from mobile device

1. Install

$ sudo npm install -g ngrok

1. Run dev

$ npm run dev

$ sudo ngrok http 8080

1. Run production

$ serve dist/

$ sudo ngrok http 5000

# Easily Installable

## Manifest

Web applications providing manifest.json are installable. Here is the complete list of manifest options on [Mozilla Developer website](https://developer.mozilla.org/en-US/docs/Web/Manifest) (https://developer.mozilla.org/en-US/docs/Web/Manifest).

### Manifest.json

{  
 "name": "pwa-facturapp",  
 "short\_name": "pwa-facturapp",  
 "icons": [  
 {  
 "src": "/static/img/icons/android-chrome-192x192.png",  
 "sizes": "192x192",  
 "type": "image/png"  
 },  
 {  
 "src": "/static/img/icons/android-chrome-512x512.png",  
 "sizes": "512x512",  
 "type": "image/png"  
 }  
 ],  
 "start\_url": "/index.html",  
 "display": "standalone",  
 "background\_color": "#000000",  
 "theme\_color": "#4DBA87"  
}

### Index.html

<link rel="manifest" href="<%= htmlWebpackPlugin.files.publicPath %>static/manifest.json">

# Progressive & Responsive

## Viewport

The viewport is the user's visible area of a web page.

The viewport varies with the device, and will be smaller on a mobile phone than on a computer screen.

### Index.html

<meta name="viewport" content="width=device-width, initial-scale=1">

# APP Like

Using material design or bootstrap

# Fast & fresh data

## Firebase

Setup database in firebase

### Create Firebase Project / Database

Create new project on <https://console.firebase.google.com>. Allow read and write for testing.

### Install firebase

$ npm install firebase --save

### Setup firebase service

#### src/service/firebase.js

Create a new file src/service/firebase.js and config authentication given by firebase app

import firebase from 'firebase'  
  
var config = {  
 apiKey: "{{ API\_KEY }}",  
 authDomain: "{{ AUTH\_DOMAIN }}",  
 databaseURL: "{{ DB\_URL }}",  
 projectId: "{{ PROJECT }}",  
 storageBucket: "{{ ST\_BUCKET }}",  
 messagingSenderId: "{{ MSG\_ID }}"  
};  
firebase.initializeApp(config)  
  
export default {  
 database: firebase.database()  
}

#### main.js

Setup firebase in main.js

import firebase from './services/firebase'

new Vue({

[…],

firebase: {

cat: firebase.database.ref('cat').orderByChild('created\_at')

},

[…]

})

## Vuefire

### Install Vuefire for database connection

$ npm install vuefire –save

### Setup main.js

import Vuefire from 'vuefire'

Vue.use(Vuefire)

## Vue-resource (HTTP client)

### Install vue-resource for http request

$ npm install vue-resource –save

### Setup main.js

import VueResource from 'vue-resource'

Vue.use(VueResource)

## Xml-parser

### Install xml-parser for easy read XML

$ npm install xml-parser –save

## GET – Getting by id

<template>  
 <div class="mdl-grid">  
 <div class="mdl-cell mdl-cell--8-col">  
 <div class="picture">  
 <img :src="cat.url" />  
 </div>  
 <div class="info">  
 <span>{{ cat.info }}</span>  
 </div>  
 </div>  
 <div class="mdl-cell mdl-cell--4-col mdl-cell--8-col-tablet">  
 <div class="comment">  
 <span>{{ cat.comment }}</span>  
 </div>  
 <div class="actions">  
 <router-link class="mdl-button mdl-js-button mdl-button--raised mdl-button--colored" to="/post">  
 ANSWER  
 </router-link>  
 </div>  
 </div>  
 </div>  
</template>  
<script>  
 **import** { find } **from** 'lodash'  
 **export default** {  
 data () {  
 **return** {  
 cat: **null** }  
 },  
 mounted () {  
 **this**.cat = find(**this**.$root.cat, (cat) => cat['.key'] === **this**.$route.params.id)  
 }  
 }  
</script>

## POST - Posting form Component

<template>  
 <form>  
 <div class="mdl-grid">  
 <div class="mdl-cell mdl-cell--8-col">  
 <div class="card-image\_\_picture">  
 <img :src="this.catUrl"/>  
 </div>  
 </div>  
 <div class="mdl-cell mdl-cell--4-col mdl-cell--8-col-tablet">  
 <div class="mdl-textfield mdl-js-textfield mdl-textfield--floating-label is-upgraded is-dirty">  
 <input id="username" v-model="title" type="text" class="mdl-textfield\_\_input"/>  
 <label for="username" class="mdl-textfield\_\_label">Describe me</label>  
 </div>  
 <div class="actions">  
 <a @click.prevent="postCat" class="mdl-button mdl-js-button mdl-button--raised mdl-button--colored">  
 POST A CAT  
 </a>  
 </div>  
 </div>  
 </div>  
 </form>  
</template>  
  
<script>  
 **import** parse **from** 'xml-parser'  
  
 **export default** {  
 data () {  
 **return** {  
 'catUrl': **null** }  
 },  
 mounted () {  
 **this**.$http.get('https://thecatapi.com/api/images/get?format=xml&results\_per\_page=1').then(response => {  
 **this**.catUrl = parse(response.body).root.children['0'].children['0'].children['0'].children['0'].content  
 })  
 },  
 methods: {  
 postCat () {  
 **this**.$root.$firebaseRefs.cat.push(  
 {  
 'url': **this**.catUrl,  
 'comment': **this**.title,  
 'info': 'Posted by Charles on Tuesday',  
 'created\_at': -1 \* **new** Date().getTime()  
 })  
 .then(**this**.$router.push('/'))  
 }  
 }  
 }  
</script>

# Connectivity independent

<https://developers.google.com/web/ilt/pwa/lab-migrating-to-workbox-from-sw-precache-and-sw-toolbox>

## Service Workers

A Service Worker is a JS script that user’s browser runs in the background, separately from your web application. A Service Worker executes code even when your application is closed or inactive, and opens the way to app-like features such as.

* caching (using Cache Storage API and Fetch API)
* push notifications (using Push API)
* background sync.

caching strategies:

* cache first (useful to cache App shell files)
* network first (useful to cache remote assets or data)

## Cache app shell with sw-precache (Internal assets)

1. Install **sw-precache**

$ npm install [sw-precache-webpack-plugin](https://github.com/goldhand/sw-precache-webpack-plugin) --save

1. Setup **webpack.prod.conf.js**

* service worker id: my-vue-app
* Filename: service-worker.js
* Cache static files in dist: .js, .html and .css
* Minify and strip prefix dist

var SWPrecacheWebpackPlugin = require('sw-precache-webpack-plugin')

...

var webpackConfig = merge(baseWebpackConfig, {

...

plugins: [

...

// service worker caching

new SWPrecacheWebpackPlugin({

cacheId: 'my-vue-app',

filename: 'service-worker.js',

staticFileGlobs: ['dist/\*\*/\*.{js,html,css}'],

minify: true,

stripPrefix: 'dist/'

})

]

})

1. Add to **index.html**

<%= htmlWebpackPlugin.options.serviceWorkerLoader %>

### HTTP Server

Emulate HTTP server for serve static files using **Serve**

1. Install Serve

$ sudo npm install -g serve

1. Execute Serve (service workers on https or localhost only)

$ serve dist/

## Cache external assets with sw-toolbox

**runtimeCaching** cache first strategy for fonts and images:

var SWPrecacheWebpackPlugin = require('sw-precache-webpack-plugin')

...

var webpackConfig = merge(baseWebpackConfig, {

...

plugins: [

...

// service worker caching

new SWPrecacheWebpackPlugin({

cacheId: 'my-vue-app',

filename: 'service-worker.js',

staticFileGlobs: ['dist/\*\*/\*.{js,html,css}'],

minify: true,

stripPrefix: 'dist/',

**runtimeCaching: [**

**{**

**urlPattern: /^https:\/\/thecatapi\.com\/api\/images\/get\.php\?id/,**

**handler: 'cacheFirst'**

**},**

**{**

**urlPattern: /^https:\/\/(\d+)\.media\.tumblr\.com\//,**

**handler: 'cacheFirst'**

**},**

**{**

**urlPattern: /^http:\/\/(\d+)\.media\.tumblr\.com\//,**

**handler: 'cacheFirst'**

**},**

**{**

**urlPattern: /^https:\/\/fonts\.googleapis\.com\//,**

**handler: 'cacheFirst'**

**},**

**{**

**urlPattern: /^https:\/\/fonts\.gstatic\.com\//,**

**handler: 'cacheFirst'**

**},**

**{**

**urlPattern: /^https:\/\/code\.getmdl\.io\//,**

**handler: 'cacheFirst'**

**}]**

})

]

})

## Cache Firebase WebSocket with Local Storage API

Code manually:

1. online: save items to cache (localStorage)
2. offline: get items from cache

<script>  
 **export default** {  
 methods: {  
 displayDetails (id) {  
 **this**.$router.push({name: 'detail', params: { id: id }})  
 },  
 getCats () {  
 **if** (navigator.onLine) {  
 **this**.saveCatsToCache()  
 **return this**.$root.cat  
 } **else** {  
 **return** JSON.parse(localStorage.getItem('cats'))  
 }  
 },  
 saveCatsToCache () {  
 **this**.$root.$firebaseRefs.cat.orderByChild('created\_at').once('value', (snapchot) => {  
 **let** cachedCats = []  
 snapchot.forEach((catSnapchot) => {  
 **let** cachedCat = catSnapchot.val()  
 cachedCat['.key'] = catSnapchot.key  
 cachedCats.push(cachedCat)  
 })  
 localStorage.setItem('cats', JSON.stringify(cachedCats))  
 })  
 }  
 },  
 mounted () {  
 **this**.saveCatsToCache()  
 }  
 }  
</script>

# Benefit from native features

## Camera (mediaDevices API / ImageCapture)

### Get picture

<https://developer.mozilla.org/en-US/docs/Web/API/MediaDevices>

<https://developer.mozilla.org/en-US/docs/Web/API/ImageCapture>

<template>  
 <div class="camera-modal">  
 <video ref="video" class="camera-stream"/>  
 <div class="camera-modal-container">  
 <span @click="capture" class="take-picture-button take-picture-button mdl-button mdl-js-button mdl-button--fab mdl-button--colored">  
 <i class="material-icons">camera</i>  
 </span>  
 </div>  
 </div>  
</template>  
  
<script>  
 **export default** {  
 data () {  
 **return** {  
 mediaStream: **null** }  
 },  
 methods: {  
 capture () {  
 **const** mediaStreamTrack = **this**.mediaStream.getVideoTracks()[0]  
 **const** imageCapture = **new** window.ImageCapture(mediaStreamTrack)  
 **return** imageCapture.takePhoto().then(blob => {  
 console.log(blob)  
 })  
 }  
 },  
 mounted () {  
 navigator.mediaDevices.getUserMedia({ video: **true** })  
 .then(mediaStream => {  
 **this**.mediaStream = mediaStream  
 **this**.$refs.video.srcObject = mediaStream  
 **this**.$refs.video.play()  
 })  
 .catch(error => console.error('getUserMedia() error:', error))  
 },  
 destroyed () {  
 **const** tracks = **this**.mediaStream.getTracks()  
 tracks.map(track => track.stop())  
 }  
 }  
</script>  
  
<style scoped>  
 .camera-modal {  
 width: 100%;  
 height: 100%;  
 top: 0;  
 left: 0;  
 position: absolute;  
 background-color: white;  
 z-index: 10;  
 }  
 .camera-stream {  
 width: 100%;  
 max-height: 100%;  
 }  
  
 .camera-modal-container {  
 position: absolute;  
 bottom: 0;  
 width: 100%;  
 align-items: center;  
 margin-bottom: 24px;  
 }  
 .take-picture-button {  
 display: flex;  
 }  
</style>

### Upload to Firebase:

<https://firebase.google.com/docs/storage/>

1. create storage and make it public

service cloud.firestore {

match /databases/{database}/documents {

match /{document=\*\*} {

allow read, write;

}

}

}

1. services/firebase.js

**const** storage = firebase.storage()  
**const** database = firebase.database()  
  
**export** {  
 database,  
 storage  
}

1. capture method

capture () {  
 **const** mediaStreamTrack = **this**.mediaStream.getVideoTracks()[0]  
 **const** imageCapture = **new** window.ImageCapture(mediaStreamTrack)  
 **return** imageCapture.takePhoto().then(blob => {  
 firebase.storage.ref().child(`images/picture-${**new** Date().getTime()}`).put(blob)  
 .then(res => {  
 **this**.postCat(res.metadata.downloadURLs[0], 'Hello')  
 **this**.$router.go(-1)  
 })  
 })  
}

# SEO

* Meta tags (vue-meta)

Ref: <https://github.com/declandewet/vue-meta>

* HTTPS

A missing HTTPS certification or a broken config could penalize your website

* Page speed

Googlebots themselves are pretty impatient and won't wait for a script longer than 5 seconds

* Sitemap

Sitemaps might not be that useful for smaller websites, but it's still a valuable SEO tool to consider.

You can generate a sitemap.xml by vue-router configuration. (<https://github.com/40818419/vue-router-sitemap>)

* Link building

How does Google know that you've become an authoritative resource? By having other relevant domains linking to yours.

There's no secret formula here, to accomplish this you need to work hard at crafting great content.

## Prerender SPA plugin

Ref: <https://alligator.io/vuejs/vue-prerender-prerenderer/>

Ref: <https://medium.com/@tribex/announcing-prerender-spa-plugin-3-0-stable-238e91fac436>

### Install

$ npm install prerender-spa-plugin --save-dev

### Configure

File webpack.config.js

**const** PrerenderSpaPlugin = require('prerender-spa-plugin')  
// Renders headlessly in a downloaded version of Chromium through puppeteer  
**const** PuppeteerRenderer = PrerenderSpaPlugin.PuppeteerRenderer

[…]

**const** webpackConfig = merge(baseWebpackConfig, {

[…]

plugins: [

[…]

// prerender SPA  
**new** PrerenderSpaPlugin({  
 staticDir: \_\_dirname, // The path to the folder where index.html is.  
 routes: ['/guests', '/invoices', '/expenses', '/graphs'], // List of routes to prerender.  
 renderer: **new** PuppeteerRenderer()  
})

]

})

### Dependencies

sudo apt-get install gconf-service -y

sudo apt-get install libasound2 -y

sudo apt-get install libatk1.0-0 -y

sudo apt-get install libc6 -y

sudo apt-get install libcairo2 -y

sudo apt-get install libcups2 -y

sudo apt-get install libdbus-1-3 -y

sudo apt-get install libexpat1 -y

sudo apt-get install libfontconfig1 -y

sudo apt-get install libgcc1 -y

sudo apt-get install libgconf-2-4 -y

sudo apt-get install libgdk-pixbuf2.0-0 -y

sudo apt-get install libglib2.0-0 -y

sudo apt-get install libgtk-3-0 -y

sudo apt-get install libnspr4 -y

sudo apt-get install libpango-1.0-0 -y

sudo apt-get install libpangocairo-1.0-0 -y

sudo apt-get install libstdc++6 -y

sudo apt-get install libx11-6 -y

sudo apt-get install libx11-xcb1 -y

sudo apt-get install libxcb1 -y

sudo apt-get install libxcomposite1 -y

sudo apt-get install libxcursor1 -y

sudo apt-get install libxdamage1 -y

sudo apt-get install libxext6 -y

sudo apt-get install libxfixes3 -y

sudo apt-get install libxi6 -y

sudo apt-get install libxrandr2 -y

sudo apt-get install libxrender1 -y

sudo apt-get install libxss1 -y

sudo apt-get install libxtst6 -y

sudo apt-get install ca-certificates -y

sudo apt-get install fonts-liberation -y

sudo apt-get install libappindicator1 -y

sudo apt-get install libnss3 -y

sudo apt-get install lsb-release -y

sudo apt-get install xdg-utils -y

sudo apt-get install wget -y