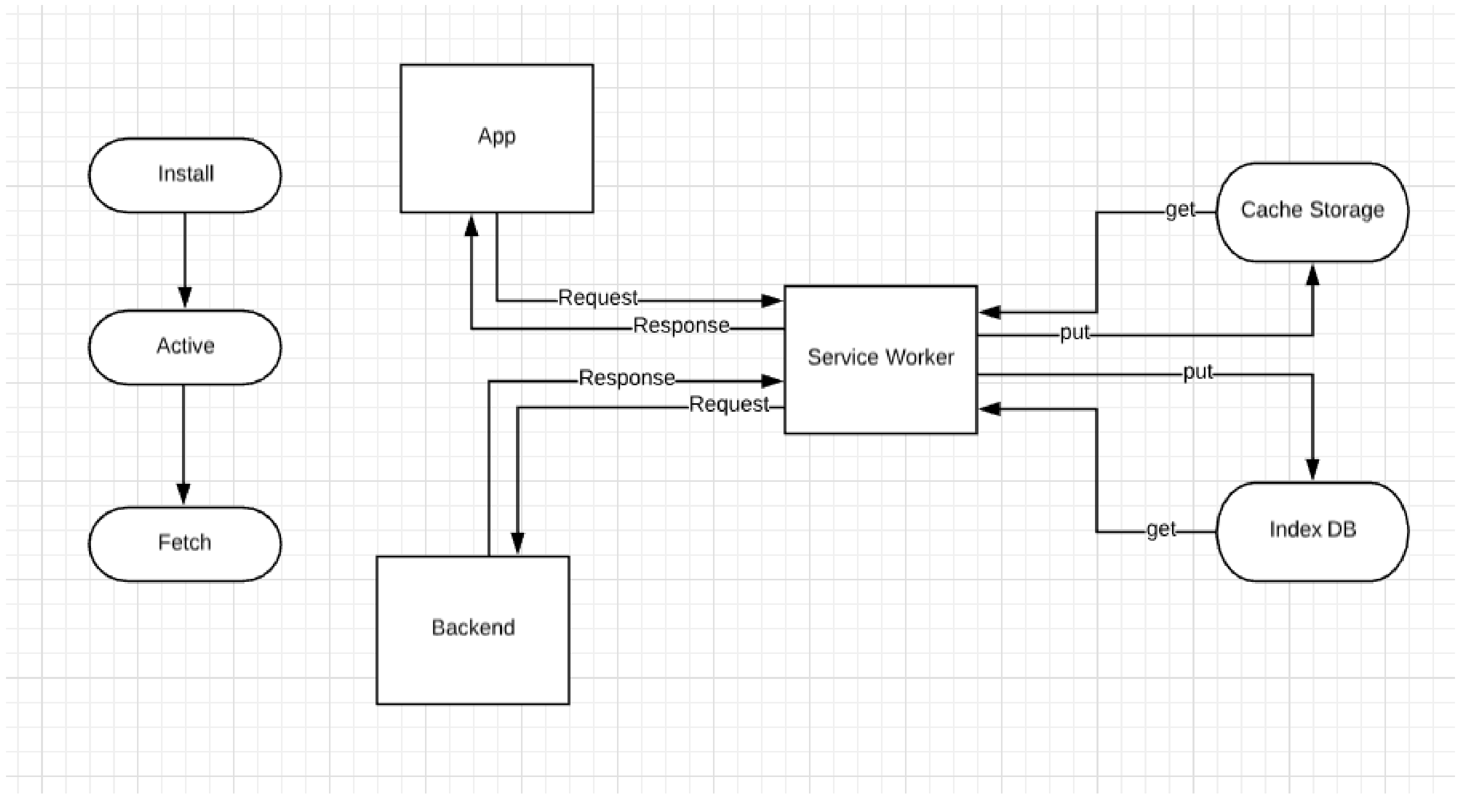
OFFLINE WEB APPS - WITH A SERVICE WORKER

https://przemyslawjanpietrzak.github.io/przemyslawjanpietrzak.github.io/offline-web-apps/dist


Install
Active
Fetch
App
Request
Response
Backend
Response
Request
Service Worker
get
Cache Storage
put
put
Index DB
get

SERVICE WORKER PRECACHE

```
// sw-precache.config.js
module.exports = {
  staticFileGlobs: [
    'static/css/**.css',
    'static/**.html',
    'static/images/**.*',
    'static/js/**.js'
  ],
  stripPrefix: 'static/',
  runtimeCaching: [{
    urlPattern: /this\.is\.a\.regex/,
    handler: 'networkFirst'
  }]
};
```

SERVICE WORKER PRECACHE

```
// sw.js
let precacheConfig = [
  ["background.jpg","eb661a7bfd811daa1ffefe7d527333ab"],
  ['index.js', 'eb661a7bfd811daa1ffefe7d527333ab'],
  ['sw.js', 'eb661a7bfd811daa1ffefe7d527333ab'],
  ['index.html', 'eb661a7bfd811daa1ffefe7d527333ab'],
  ['fonts.css', 'eb661a7bfd811daa1ffefe7d527333ab'],
  ['material.css', 'eb661a7bfd811daa1ffefe7d527333ab'],
  ['material.min.js', 'eb661a7bfd811daa1ffefe7d527333ab'],
  ['material.indigo-pink.min.css', 'eb661a7bfd811daa1ffefe7d527333ab'],
  ['flUhRq6tzZclQEJ-Vdg-IuiaDsNcIhQ8tQ.woff2', 'eb661a7bfd811daa1ffefe7d527333ab'],
];
```

CACHE STATIC FILES

```
self.addEventListener('install', (event) => {
  event.waitUntil(
    caches
      .open(staticFilesCacheName)
      .then(cache => setOfCachedUrls(cache))
      .then(cachedUrls => Promise.all(
          Array.from(urlsToCacheKeys.values()).map((cacheKey) => {
            if (!cachedUrls.has(cacheKey)) {
              const request = new Request(cacheKey, {credentials: 'same-origin'});
              return fetch(request).then((response) => {
                if (!response.ok) {
                  throw new Error(`Request for ${cacheKey} returned a response with status  ${response.status}`);
                }

                return cleanResponse(response)
                  .then(responseToCache => cache.put(cacheKey, responseToCache));
              });
            }
          })
      ))
    .then(() => self.skipWaiting())
  )
});
```

ON FETCH

```
self.addEventListener('fetch', (event) => {
  if (['GET', 'OPTIONS'].includes(event.request.method)) {
      const { url } = event.request;
      const shouldRespond = urlsToCacheKeys.has(url);
      if (shouldRespond) {
          event.respondWith(
            caches
                .open(staticFilesCacheName)
                .then((cache) => cache.match(urlsToCacheKeys.get(url)))
                .then((response) => {
                    if (response) {
                        return response;
                    }
                    throw Error('The cached response that was expected is missing.');
                }).catch(_ => fetch(event.request))
            );
      }

  }
});
```

DEMO

CACHE API

```
const createResponse = obj => new Response(JSON.stringify(obj), {
  headers: {'Content-Type': 'application/json'}
});


if (event.request.url.includes('/api/')) {
  event.respondWith(idbClient
    .keys()
    .then(keys => {
      if (keys.includes(event.request.url)) {
        return idbClient
          .get(event.request.url)
          .then(createResponse);
      }
      return fetch(event.request)
        .then(r => r.json())
        .then(r => idbClient
          .set(event.request.url, r)
          .then(_ => createResponse)
        );
    }));
}
```

DEMO

INVALIDATE CACHE IDB

```
const REQUEST_CACHE_LIFETIME = 60 * 60 * 1000;

const fetchAndSaveAPIRequest = request => fetch(event.request)
    .then(r => r.json())
    .then(response => idbClient
        .set(event.request.url, { response, timestamp: now() + REQUEST_CACHE_LIFETIME })
        .then(_ => createResponse(response))
    );

if (event.request.url.includes('/api/')) {
    event.respondWith(idbClient
      .keys()
      .then(keys => {
        if (keys.includes(event.request.url)) {
          return idbClient
            .get(event.request.url)
            .then(({ response, timestamp }) => {
              if (timestamp > now() || !navigator.onLine) {
                return createResponse(response);
              }
              return fetchAndSaveAPIRequest(event.request);
            });
        }
        return fetchAndSaveAPIRequest(event.request);
      }));
  }
```

REFRESH DATA

```
self.addEventListener('activate', (event) => {
  let setOfExpectedUrls = new Set(urlsToCacheKeys.values());

  event.waitUntil(
    caches
      .open(cacheName)
      .then((cache) => cache.keys())
      .then(existingRequests => Promise.all(
          existingRequests.map((existingRequest) => {
            if (!setOfExpectedUrls.has(existingRequest.url)) {
              return cache.delete(existingRequest);
            }
          })
      ))
      .then(() => self.clients.claim())
  );
});
```

LIE-FI

```
self.addEventListener('fetch', (event) => {
  event.respondWith(Promise(resolve => {
    setTimeout(() => {
      getFromCache.then(resolve)
    }, TIMEOUT)

    fetch(url).then(resolve)
  })

  )
});
```

CHECK NETWORK STATUS

```
if (navigator.onLine) {
  console.log('online');
} else {
  console.log('offline');
}

window.addEventListener('offline', _ => { console.log('offline'); });
window.addEventListener('online', _ => { console.log('online'); });
```

WEBPACK HELPERS

```
// webpack.config.js
{
  loader: 'string-replace-loader',
  options: {
  multiple: [
    {
      search: '__SW_STATIC_HASH__',
      replace: staticHash
    },
    {
      search: '__SW_API_HASH__',
      replace: apiHash
    }
  ]
}
```

CONCLUSION

LINKS

- https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API
- https://github.com/GoogleChromeLabs/sw-precache

THANK YOU ;*