Building an offline experience with a Rails-powered PWA

Speaker: Alicia Rojas

Railsconf

ATLANTA 2023

Alicia Rojas

- Software Developer at Telos Labs
- Natural resources engineer
- Passionate about technology and sustainability

RailsConf ATLANTA 2023

Overview

1. What is a PWA and why they're great?
2. Case study: Introducing the challenge and the solution
3. Setting up the main files required for PWAs
4. Caching and adding an offline fallback
5. Offline CRU(D) actions
6. Key takeaways and further challenges

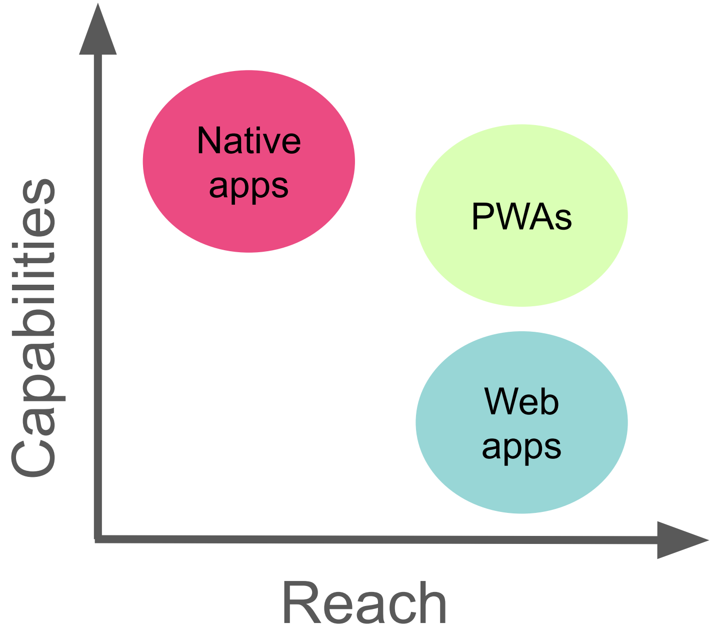
What is a PWA?

Progressive Web Apps (PWAs) are web apps that use ***service workers***, ***manifests****, and other web-platform features in combination with* ***progressive enhancement*** *to give users an experience on par with native apps.*

RailsConf ATLANTA 2023

Railsconf ATLANTA 2023

PWAs mix the best of both the native and web worlds



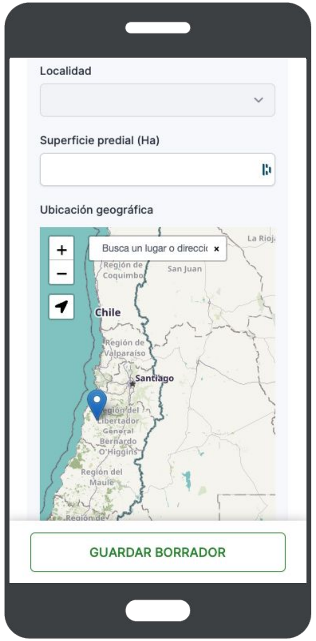
PWA	Native app
Platform independent	Platform specific
Limited functionality	Complex functionality
Search engine visibility	App store visibility
Budget-friendly	Cost-intensive
No installation required	On-device installation

Case study

The challenge

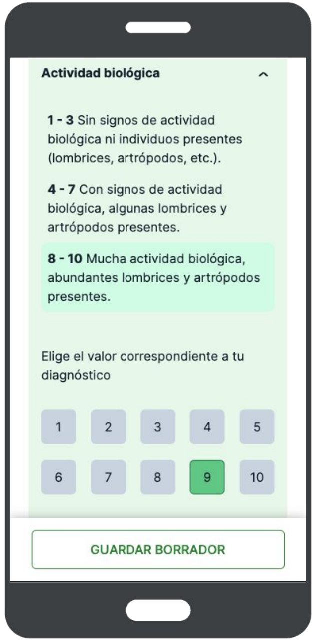
- To create an app that allows to perform CRUD actions in areas with low or no internet connection.
- The app has to be easy to use and share, and must work on mobile devices.

Farm technicians can complete a complex survey in areas where internet is poorly available


Localidad

Superficie predial (Ha)

Ubicación geográfica


Actividad biológica

1 - 3 Sin signos de actividad biológica ni individuos presentes (lombrices, artrópodos, etc.).

4 - 7 Con signos de actividad biológica, algunas lombrices y artrópodos presentes.

8 - 10 Mucha actividad biológica, abundantes lombrices y artrópodos presentes.

Elige el valor correspondiente a tu diagnóstico

1 2 3 4 5

6 7 8 [✓ 9] 10

The solution

An app built with our favorite framework, **enhanced with PWA features**

RAILS + PWA

How to turn your Rails app into a PWA

RailsConf ATLANTA 2023

Railsconf ATLANTA 2023

The ingredients

1. Service worker

Is like a small application that runs in parallel to your Rails app, intercepting requests and providing capabilities like caching assets and background sync.

Available at `/service-worker.js`

The ingredients

2. App manifest

It will tell the browser how your PWA should display within the operating system of your user’s device. This file is key to make your app installable and thus to make it ***look and feel*** like a native app.

Available at `/manifest.json`

Adding the files to our app

1. Create a service worker controller
2. Add routes
3. Add files (service worker and manifest)
4. Write companion JS and place it in the asset pipeline

1. Create a service worker controller

```ruby
class ServiceWorkerController < ApplicationController
  skip_before_action :authenticate_user!

  def service_worker
  end

  def manifest
  end
end
```

Useful if using Devise

2. Add routes

These routes will match the URLs with the actions in our service worker controller.

```ruby
# config/routes.rb

get "/service-worker.js" => "service_worker#service_worker"
get "/manifest.json" => "service_worker#manfiest"
```

3. Add service worker and manifest

Since we created controller actions and routes for the service worker and app manifest, these files can now be served as view templates from your `views/service_worker` directory 😎

```
# app/views/service_worker/manifest.json.erb
{
"short_name": "AwesomeApp",
"name": "AwesomeApp",
"id": "/",
"icons": [
  {
    "src": "<%= image_path "icon.png" %>",
    "sizes": "144x144",
    "type": "image/png"
  }
],
"start_url": "/",
"background_color": "#000000,
"display": "standalone",
"scope": "/",
"theme_color": "#000000"
}
```

json.erb to use `image_path` helper

Home screen icon

Splash screen

How the browser will behave

We then require the manifest in the application layout.

```
<!-- app/views/layout/application.html.erb -->
<link rel="manifest" href="/manifest.json"/>
```

```
// app/views/service_worker/service_worker.js
function onInstall(event) {
console.log('[Serviceworker]', "Installing!", event);
}
function onActivate(event) {
console.log('[Serviceworker]', "Activating!", event);
}
function onFetch(event) {
console.log('[Serviceworker]', "Fetching!", event);
}
self.addEventListener('install', onInstall);
self.addEventListener('activate', onActivate);
self.addEventListener('fetch', onFetch);
```

Callbacks for main SW's lifecycle events

4. Write companion JS

It tells your application when and from where to load your service worker

Railsconf
ATLANTA 2023

Check for SW support

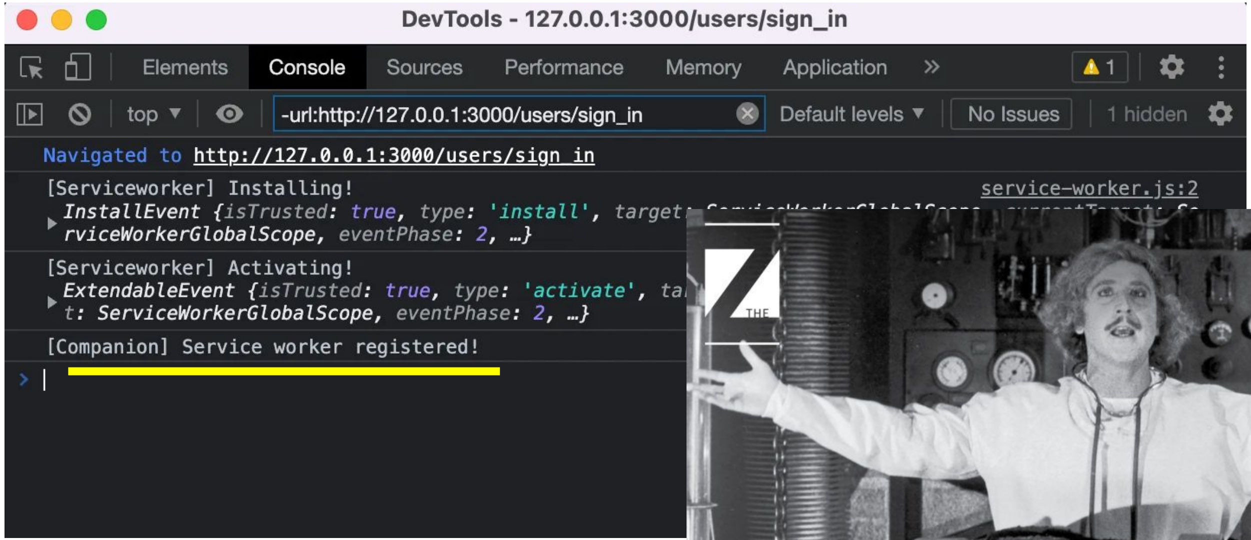
```
// app/javascript/custom/companion.js
if (navigator.serviceWorker) {
// do stuff (register service worker, handle updates, etc.)
}
```

We let Rails see this file by pinning it in the importmap and importing it in `application.js`

```ruby
# config/importmap.rb
pin_all_from "app/javascript/custom", under: "custom"
```

```javascript
// app/javascript/application.js
import "custom/companion"
```

It's alive!



```
Navigated to http://127.0.0.1:3000/users/sign_in
[Serviceworker] Installing!                                        service-worker.js:2
▸ InstallEvent {isTrusted: true, type: 'install', target: ServiceWorkerGlobalScope, currentTarget: Se
  rviceWorkerGlobalScope, eventPhase: 2, …}
[Serviceworker] Activating!
▸ ExtendableEvent {isTrusted: true, type: 'activate', ta
  t: ServiceWorkerGlobalScope, eventPhase: 2, …}
[Companion] Service worker registered!
>
```

Caching assets and adding an offline fallback

WE'RE GONNA BUILD A RAILS PWA
WE'RE WRITING RUBY CODE, RIGHT?
RIGHT?
imgflip.com

RailsConf ATLANTA 2023

Railsconf ATLANTA 2023

```javascript
// app/views/service_worker/service_worker.js
function onInstall(event) {
console.log('[Serviceworker]', "Installing!", event);
}
function onActivate(event) {
console.log('[Serviceworker]', "Activating!", event);
}
function onFetch(event) {
console.log('[Serviceworker]', "Fetching!", event);
}
self.addEventListener('install', onInstall);
self.addEventListener('activate', onActivate);
self.addEventListener('fetch', onFetch);
```

$\}$ Callbacks for main SW's lifecycle events

Workbox

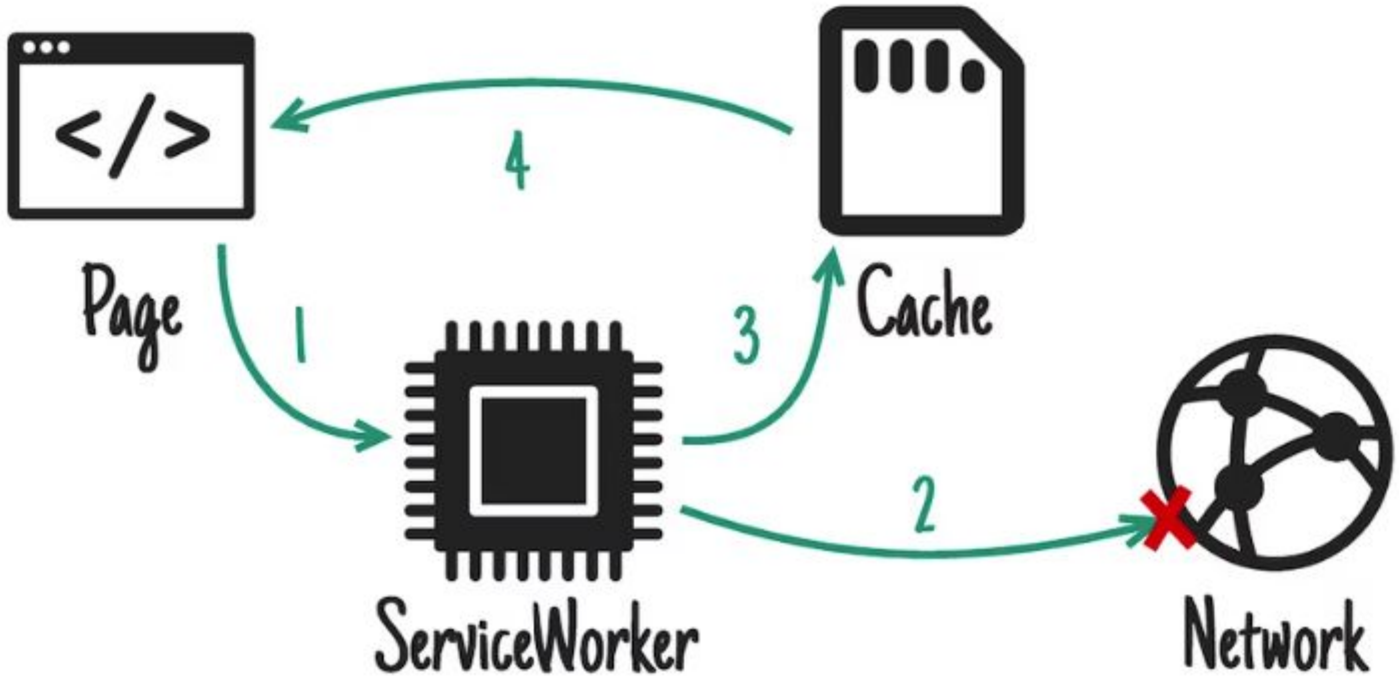
- Set of modules that simplify common service worker routing and caching.
- Workbox makes writing a service worker much easier as if we were to implement these functions ourselves.

Import module via CDN

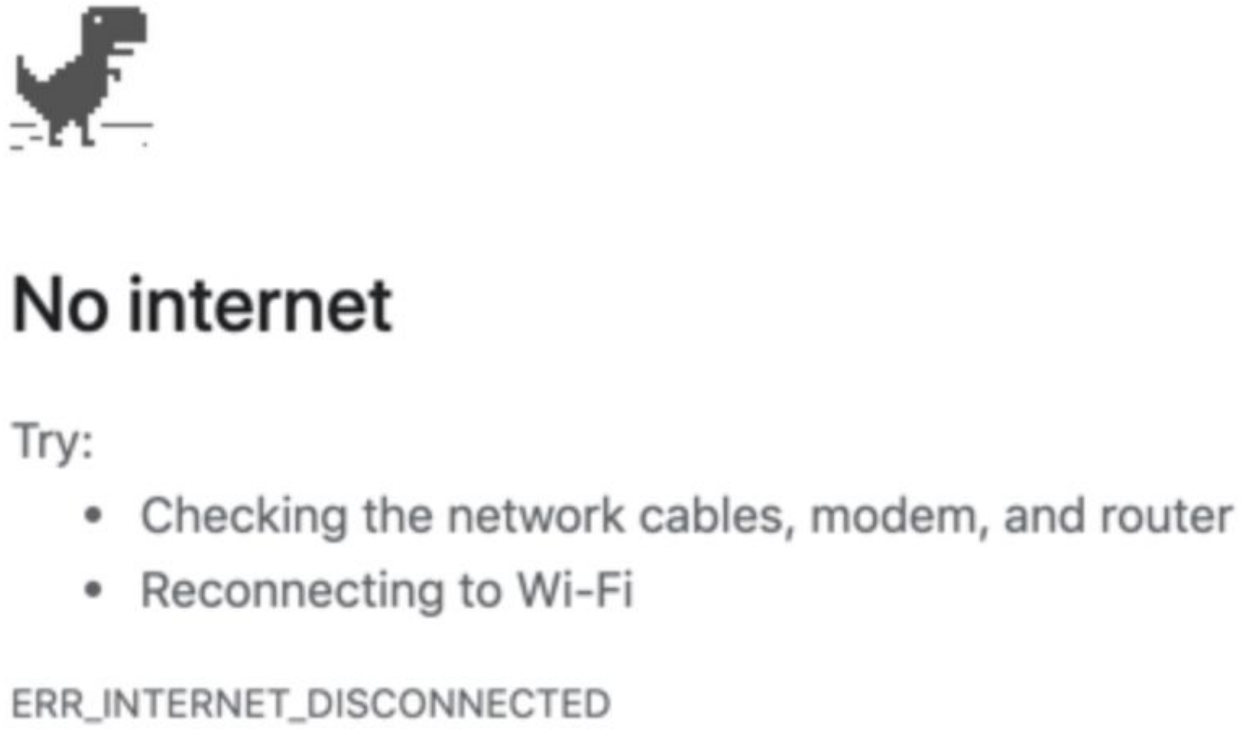
```
// app/views/service_worker/service_worker.js
importScripts(
"https://storage.googleapis.com/workbox-cdn/releases/6.4.1/wor
kbox-sw.js")
```

Workbox Strategies

- **Commonly used patterns** to determine how the service worker will generate a response when a fetch event is received.
- The service worker handles the response by “routing” it to the particular strategy we want to use.


Page
1
ServiceWorker
2
Network
3
Cache
4

What would happen if we’re offline and the service worker intercepts a request that does not match any of the cached responses?



No internet

Try:

- Checking the network cables, modem, and router
- Reconnecting to Wi-Fi

ERR_INTERNET_DISCONNECTED

Offline fallback

- A page that would let our users know that they’re not connected within the UX of the app.
- We cache this page in advance so it will always be available.

Update service worker controller, add route and create view at `views/service_worker/offline.html.erb`

```ruby
# app/controllers/service_worker_controller.rb
class ServiceWorkerController < ApplicationController
    skip_before_action :authenticate_user!

    def service_worker
    end

    def manifest
    end

    def offline
    end
end
```

Set up a catch handler

```
// app/views/service_worker/service_worker.js
const {warmStrategyCache} = workbox.recipes;
const {setCatchHandler} = workbox.routing;
const strategy = new CacheFirst();
const urls = ['/offline.html'];
// Warm the runtime cache with a list of asset URLs
warmStrategyCache({urls, strategy});
// Trigger a 'catch' handler when any of the other routes fail to generate a response
setCatchHandler(async ({event}) => {
     switch (event.request.destination) {
       case 'document':
         return strategy.handle({event, request: urls[0]});
       default:
        return Response.error();
      }
});
```

UNRELIABLE NETWORK
SERVICE WORKER
imgflip.com

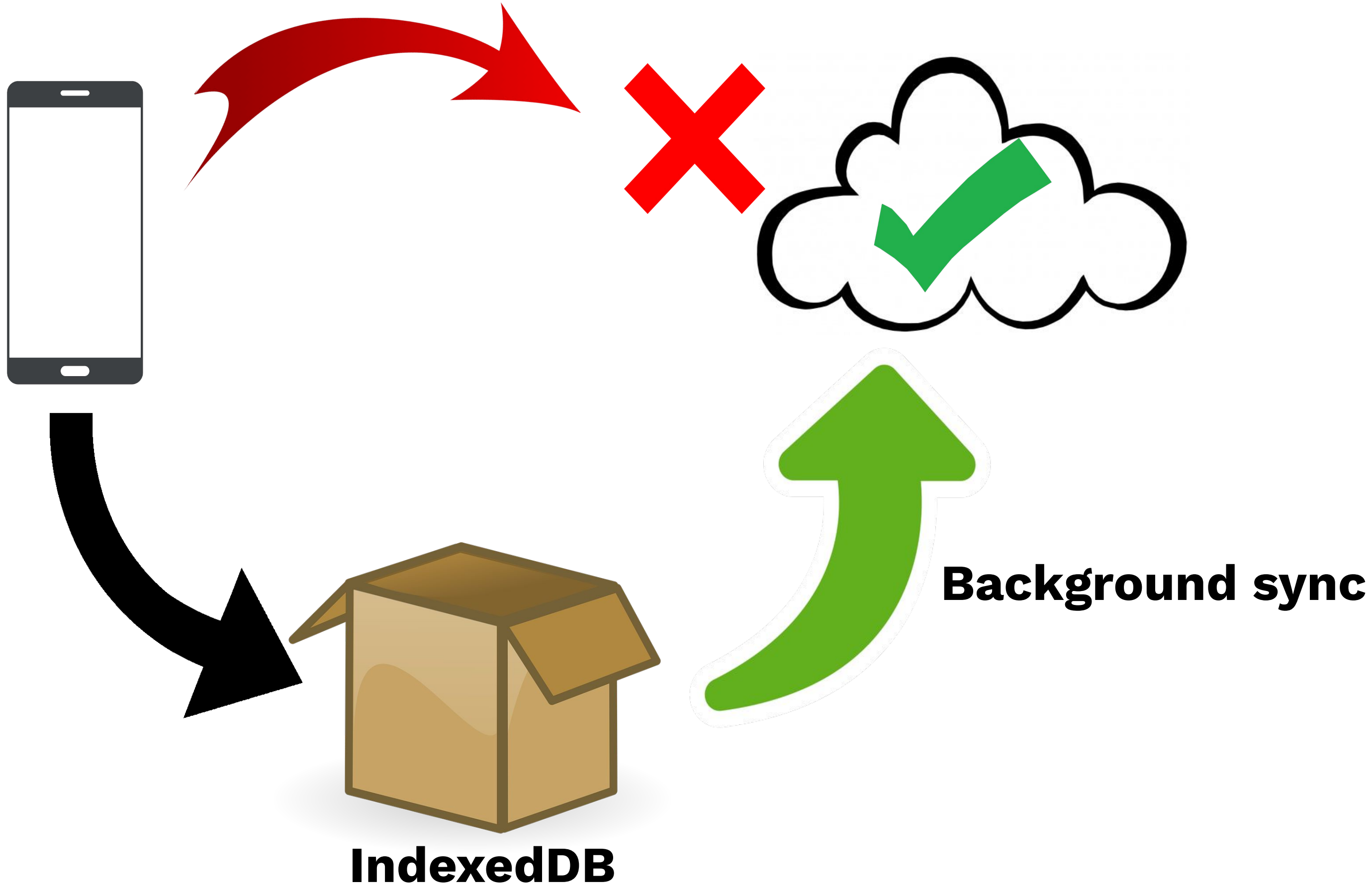
Creating records offline and sync them later

IndexedDB

- JavaScript API for managing a **database of JSON objects** in your browser.
- Relies heavily on **Promises**
- Recommendation: use a wrapper!

Background Sync API

Allows web applications to defer server synchronization work to their service worker to handle at a later time, if the device is offline.


Background sync
IndexedDB

Store records in IndexedDB with Stimulus

1. Check network status
2. Declare database
3. When submitting a form, if no network is available, store record in IndexedDB

Pro Tip: Use mixins!

Setting up background sync

Check for Background Sync support

```
// app/javascript/custom/companion.js
if (navigator.serviceWorker && "SyncManager" in window) {
// do stuff (register service worker, handle updates,
etc.)
}
```

```
// app/views/service_worker/service_worker.js

async function requestBackgroundSync() {
     await self.registration.sync.register('sync-surveys');
 }

self.addEventListener('sync', function(event) {
    if (event.tag == 'sync-surveys') {
        event.waitUntil(syncSurveys());
     }
});
```

```
// app/views/service_worker/service_worker.js

async function syncSurveys() {
const db = findOrCreateDB()

// use your preferred wrapper syntax
if (await.db.surveys.count !=0 ) {
    const surveys = await db.table(‘surveys’).toArray()
    const surveyIdsToRemove = []
     // (...)
    }
}
```

```
// (...)
await Promise.all(surveys.map(survey) => {
  try {
    // make ajax request to your server
if (response.ok) {
     surveyIdsToRemove.push(survey.id)
  }
    } catch (error) {
   // handle error
}
})
```

```
// (...)
// after looping through all surveys, remove synchronized surveys from
IndexedDB

for (let id of surveysIdsToRemove) {
   await db.surveys.delete(id)
}
```

What if we want to enable manual synchronization?

Stimulus!

```
<!-- app/views/surveys/index.html.erb -->
<div data-controller="pwa--sync" data-action="click->pwa--sync#sync" >
 <button type="button" data-pwa--sync-target="button">
   Sync
 </button>
</div>
```

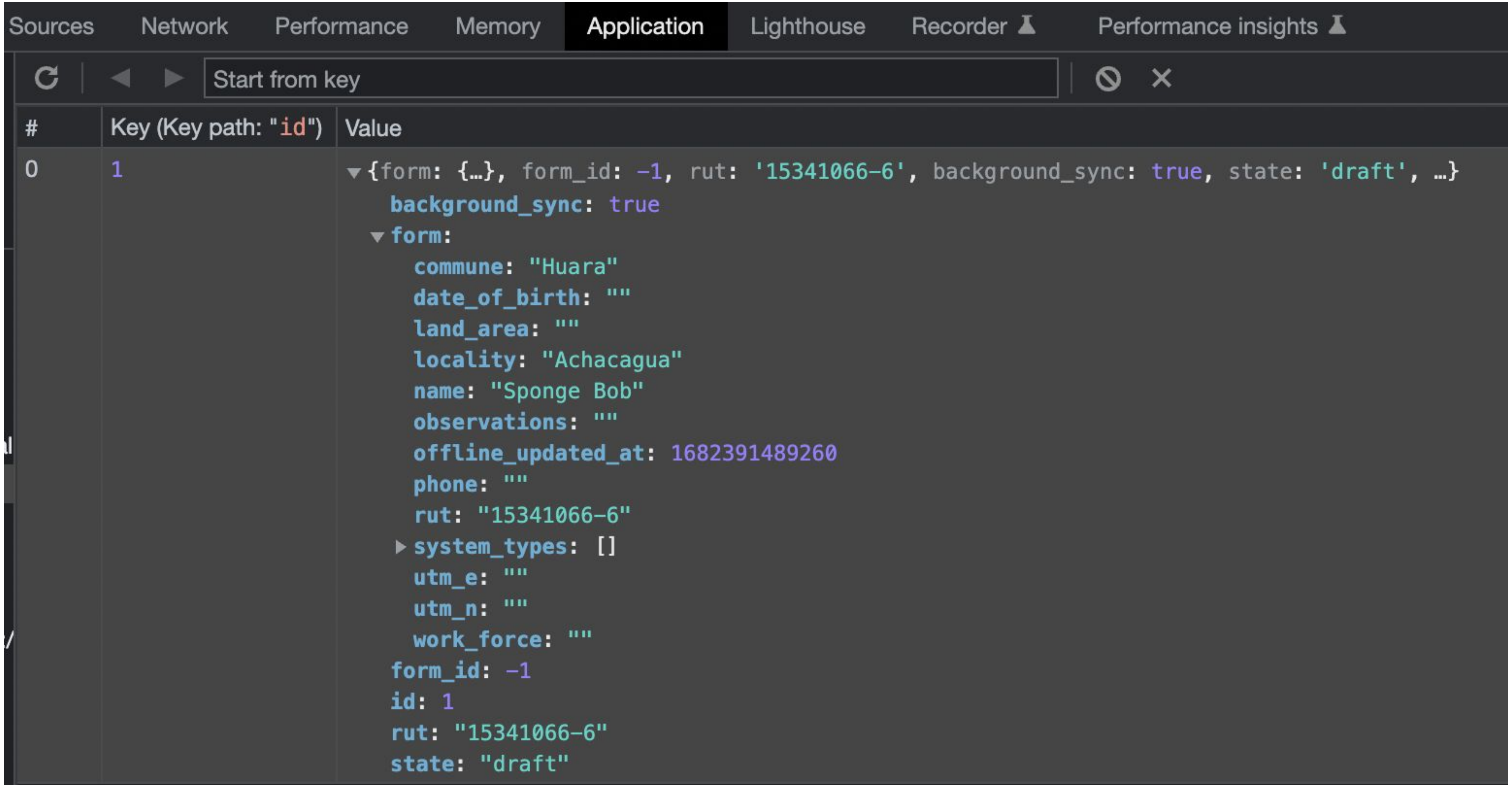
Create

Read

Update

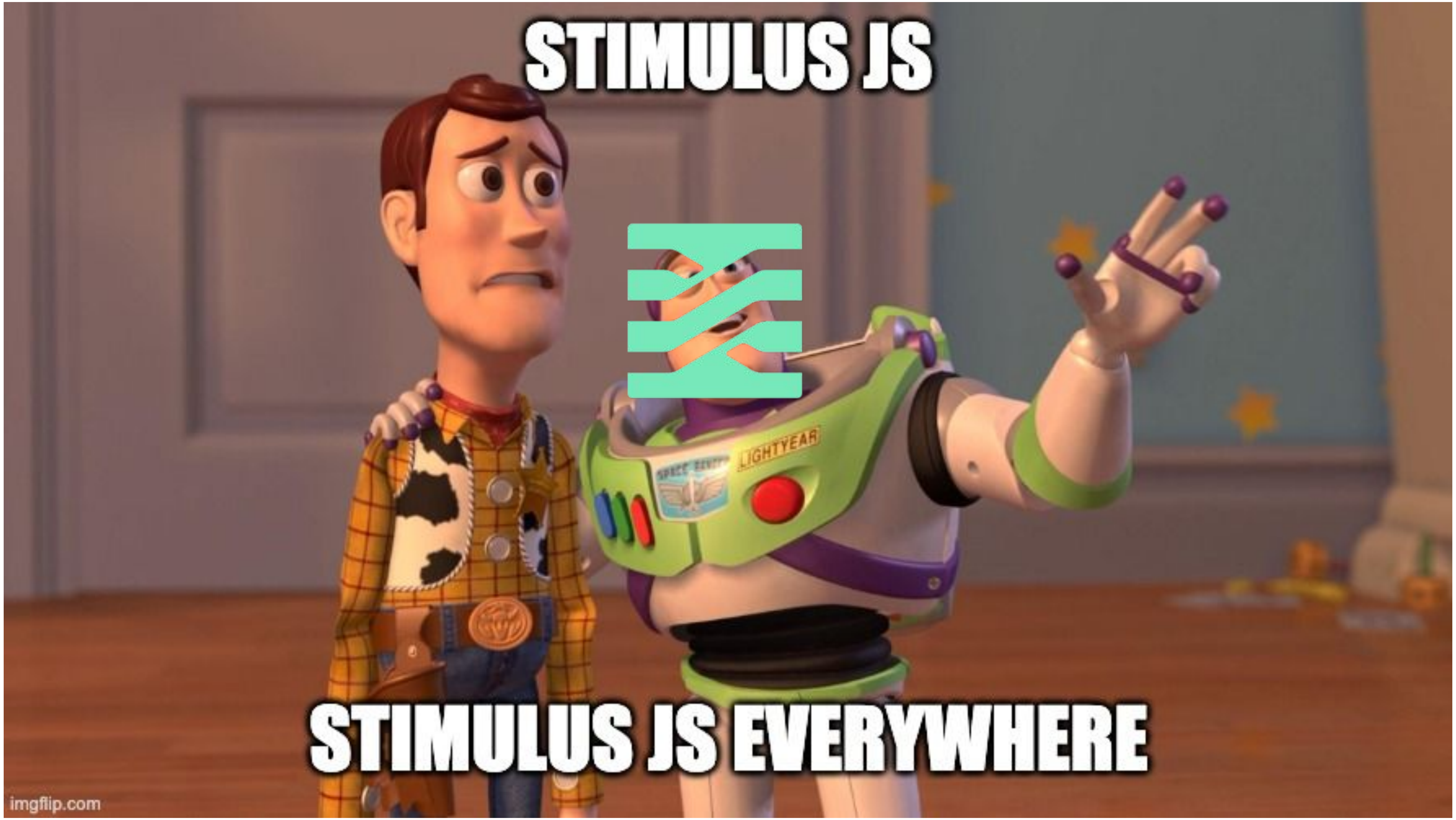
Delete

RailsConf
ATLANTA 2023



```
#   Key (Key path: "id")   Value
0   1                      ▼{form: {…}, form_id: -1, rut: '15341066-6', background_sync: true, state: 'draft', …}
                             background_sync: true
                            ▼form:
                               commune: "Huara"
                               date_of_birth: ""
                               land_area: ""
                               locality: "Achacagua"
                               name: "Sponge Bob"
                               observations: ""
                               offline_updated_at: 1682391489260
                               phone: ""
                               rut: "15341066-6"
                              ▶system_types: []
                               utm_e: ""
                               utm_n: ""
                               work_force: ""
                             form_id: -1
                             id: 1
                             rut: "15341066-6"
                             state: "draft"
```

How do we allow users to interact with app data when no internet connection is available?


STIMULUS JS
STIMULUS JS EVERYWHERE
LIGHTYEAR
imgflip.com

Create

Read

Update

Delete

Read IndexedDB data with Stimulus

```
import { Controller } from "@hotwired/stimulus"
import { useIndexedDB } from "useIndexedDB"
import Mustache from "mustache";

export default class extends Controller {
// (...)
async displayOfflineSurveys() {
  const surveys = await this.db.table('surveys').toArray()
  surveys.forEach(async(survey) => {
      if (!this.listItemExists(survey)) {
        this.listContainerTarget.innerHTML += (this.listItem(survey))
      }
      if (this.formExistsInServer(survey)) {this.removeSyncedItem(survey)}
    })
  }
// (...)
```

HTML templates are your friends :)

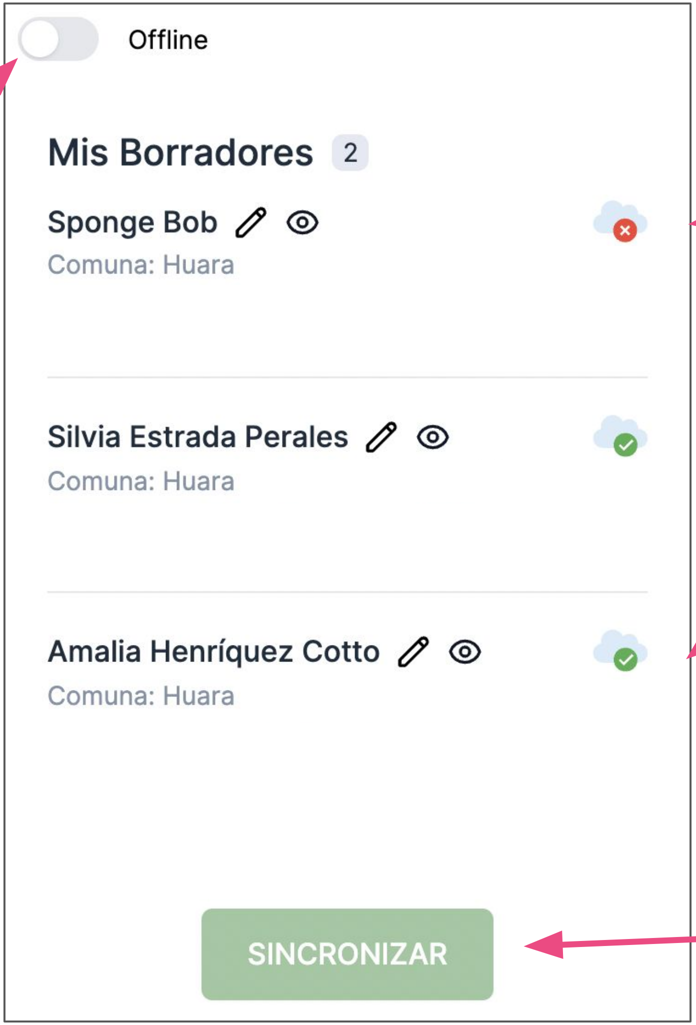
RailsConf
ATLANTA 2023

Railsconf
ATLANTA 2023

```
<template data-pwa--index-target="listItemTemplate">
    <div data-list-item-dom-id="{{ dom_id }}" >
          <a href="">
            <p>{{ name }}</p>
          </a>
         <%= inline_svg_tag 'icons/pencil.svg', size: '20*20' %>
        <p>Comuna: {{ commune }}</p>
      </div>
        <%= inline_svg_tag 'icons/cloud-not-synced.svg', size: '30*23' %>
</template>
```

Use Stimulus + Mustache to populate the template with IndexedDB data and render it

```
  listItem(survey) {
  const template = this.listItemTemplateTarget.innerHTML
  const rendered = Mustache.render(template, {
    name: survey.name,
    commune: survey.commune,
    dom_id: survey.id
  })
  return rendered
}
```


Offline

Mis Borradores 2

Sponge Bob
Comuna: Huara

Silvia Estrada Perales
Comuna: Huara

Amalia Henríquez Cotto
Comuna: Huara

SINCRONIZAR

The toggle informs if the app is online or not and allows the user to force offline behavior.

This survey is not in the server yet

But these are

Sync button is disabled while offline

Update

- Create a Mustache version of the form and put it in your view inside `<template>` tags.
- Use Stimulus + Mustache to populate the form with IndexedDB data.
- Re-use Stimulus controller to save changes in IndexedDB when submitting the form.

RailsConf ATLANTA 2023

Railsconf ATLANTA 2023

Gotchas

- Validations in the front-end and back-end must match, to ensure sync does not fail due to validation errors.
- Understand your audience to assess the importance of browser compatibility for PWA capabilities (especially regarding iOS support).

Railsconf
ATLANTA 2023

Key takeaways

- PWA features can super-charge your app to make it suitable for everyone, everywhere.
- We can make a great impact by reaching unconventional audiences.
- Stimulus is a powerful tool for enhancing our app with offline features, emulating a SPA behavior with minimum JS.

Further oportunities

- Make Rails apps easily PWAble:
  - Add service worker and manifest using a nice command like `rails generate pwa:install`
  - Create offline versions for views using another nice command like `rails generate pwa:views MODEL`

Railsconf ATLANTA 2023

Resources

- Blog posts
  - Part 1
  - Part 2
- WorkBox
- Dexie
- Mustache

RailsConf
ATLANTA 2023

You can find me at @_alicia_paz & alicia@teloslabs.co

Railsconf
ATLANTA 2023