A picture containing graphical user interface

Description automatically generated

SPA to PWA

Single Page Application hasta Progressive Web Application

29 marzo 2023

# Descripción del Cloud Academy

Genera tú solución web con SPA (Single Page Application) y SWA (Azure Static Web Application) hasta despliegue como una Progressive Web App integrando herramientas de nivel empresarial sin ningún esfuerzo adicional. Tu solución web contará con ciclo de CI/CD, SSL, distribuido globalmente, seguro, routing, API y más.

# Agenda

|  |  |
| --- | --- |
| **HORARIO** | **ACTIVIDAD** |
| 10:00 a 10:20 | Introducción |
| 10:20 a 10:30 | Single Page Application (SPA) |
| 11:40 a 12:00 | Break |
| 12:00 a 12:10 | Static Web Apps |
| 13:00 a 14:00 | Comida |
| 14:00 a 15:30 | Static Web Apps |
| 15:30 a 16:30 | Progressive Web apps |
| 16:30 a 17:30 | Conclusiones y cierre |

Contenido

[Descripción del Cloud Academy 2](#_Toc130669043)

[Agenda 2](#_Toc130669044)

[1. Herramientas 5](#_Toc130669045)

[Prerequisitos 5](#_Toc130669046)

[Instalación Node.js y NPM 5](#_Toc130669047)

[Visual Studio Code 7](#_Toc130669048)

[Extensiones para VS Code 7](#_Toc130669049)

[Static Web Apps 7](#_Toc130669050)

[Functions 8](#_Toc130669051)

[Vue.js 10](#_Toc130669052)

[Functions Core Tools 11](#_Toc130669053)

[Git 14](#_Toc130669054)

[Vue 15](#_Toc130669055)

[2. Single Page Application (SPA) 17](#_Toc130669056)

[¿Qué es Single Page Application? 17](#_Toc130669057)

[Desplegar el código vía Vue 18](#_Toc130669058)

[Ejecutar la aplicación 26](#_Toc130669059)

[Exploración del SPA 26](#_Toc130669060)

[Solo una página index.html 27](#_Toc130669061)

[Carga de App 28](#_Toc130669062)

[Carga de Route 29](#_Toc130669063)

[Agregar una nueva Vista 30](#_Toc130669064)

[Agregar la ruta a nueva vista 30](#_Toc130669065)

[Prueba de la Vista 31](#_Toc130669066)

[3. Integración Continua y Despliegue Continuo 32](#_Toc130669067)

[Opción 1: Github 33](#_Toc130669068)

[Opcion 2: Azure Dev Ops (ADO) 34](#_Toc130669069)

[4. Azure Static Web Apps (SWA) 39](#_Toc130669070)

[¿Qué es Azure Static Web Apps? 39](#_Toc130669071)

[Despliegue de la primera SWA 41](#_Toc130669072)

[Routing de SWA 47](#_Toc130669073)

[Crear un API en SWA 49](#_Toc130669074)

[Configurar carpeta API para el SWA 51](#_Toc130669075)

[Agregar un elemento de prueba 52](#_Toc130669076)

[Autenticación de SWA 53](#_Toc130669077)

[Acceso de usuario en API 56](#_Toc130669078)

[Roles 57](#_Toc130669079)

[Integración con una Base de datos 59](#_Toc130669080)

[Environments 72](#_Toc130669081)

[Configurar Dominio 74](#_Toc130669082)

[5. Progressive Web App (PWA) 78](#_Toc130669083)

[¿Qué es Progressive Web App? 78](#_Toc130669084)

[Actualización de SPA a PWA 79](#_Toc130669085)

[Revisión de componentes de PWA 80](#_Toc130669086)

[Despliegue con Builder 88](#_Toc130669087)

# Herramientas

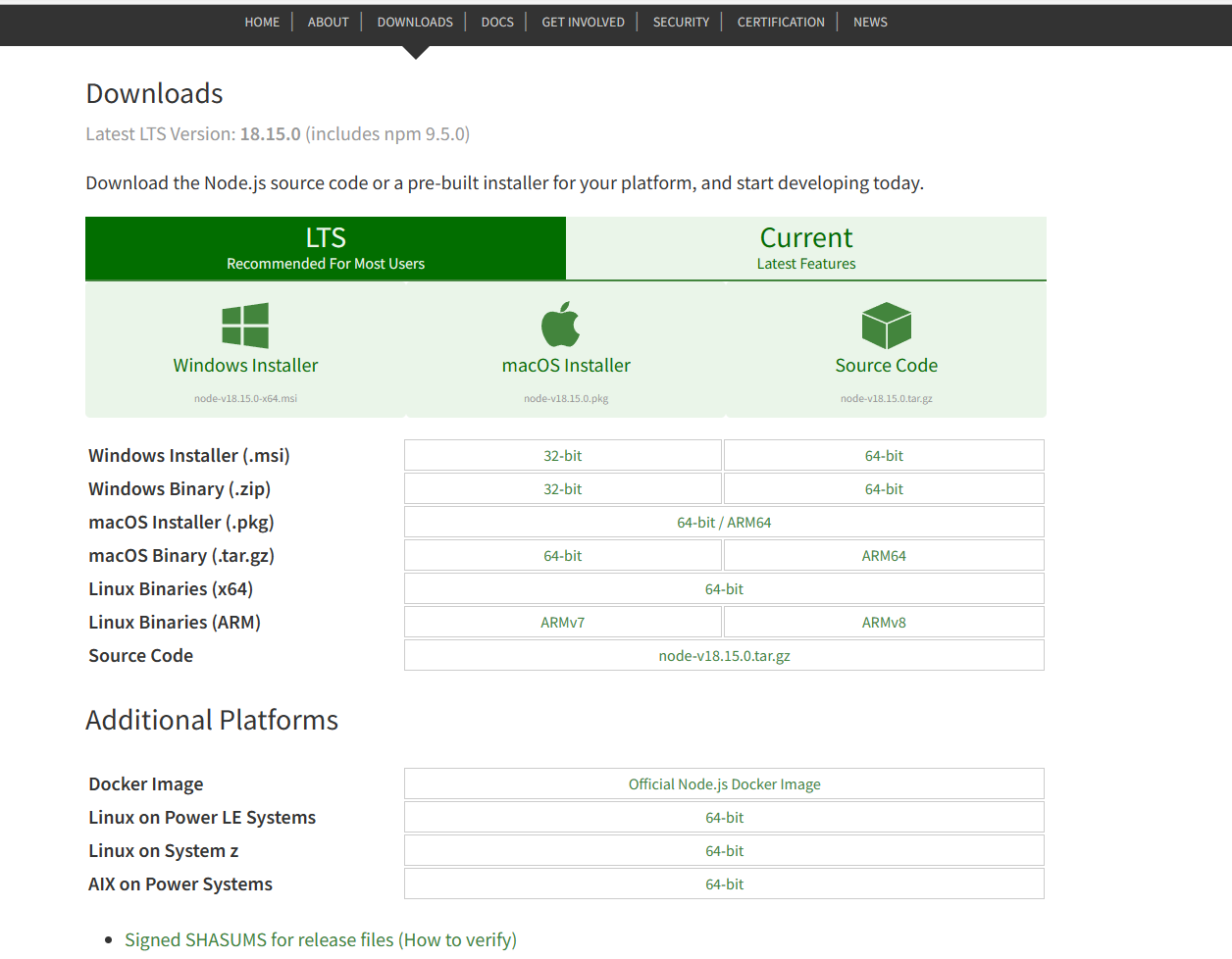
## Prerrequisitos

Las siguientes herramientas se requieren durante este Cloud Academy:

* Node.js
* Visual Studio Code
* Git
* Vue
* Function Core Tools

## Instalación Node.js y NPM

Descargar la versión de Node.js descargando de la URL [Node.js (nodejs.org)](https://nodejs.org/en/download/)



Se ejecuta la aplicación de instalación:Graphical user interface, text, application

Description automatically generated

Se valida la instalación por medio de ejecutar en una línea de comando (cmd) los siguientes comandos:

|  |  |
| --- | --- |
| node -v  npm -v | Text  Description automatically generated |

## Visual Studio Code

VS Code se instalará desde la siguiente URL: <https://code.visualstudio.com/Download>

Graphical user interface, website

Description automatically generated

## Extensiones para VS Code

### Static Web Apps

|  |  |
| --- | --- |
| Ir la opción de Extensiones: | Graphical user interface, application  Description automatically generated |
| Ingresar en el espacio de búsqueda en la parte superior el texto “static web apps”. | Graphical user interface, application  Description automatically generated |
| Se da opción de instalar a la siguiente extensión: |  |

### Functions

|  |  |
| --- | --- |
| Ir la opción de Extensiones: | Graphical user interface, application  Description automatically generated |
| Ingresar en el espacio de búsqueda en la parte superior el texto “functions”. | Graphical user interface, application  Description automatically generated |
| Se da opción de instalar a la siguiente extensión: |  |

### Vue.js

|  |  |
| --- | --- |
| Ir la opción de Extensiones: | Graphical user interface, application  Description automatically generated |
| Ingresar en el espacio de búsqueda en la parte superior el texto “@id:vue.volar”. | Graphical user interface, application  Description automatically generated |
| Se da opción de instalar a la siguiente extensión: | A screenshot of a computer  Description automatically generated with medium confidence |

## Functions Core Tools

|  |  |
| --- | --- |
| Son herramientas que ayudan en el uso de Azure functions en forma local. El primer paso es ir a la URL: <https://learn.microsoft.com/en-us/azure/azure-functions/functions-run-local> | Graphical user interface, text, application, email  Description automatically generated |
| Se descarga la opción de sistema operativo de la maquina local: | Graphical user interface, text, application  Description automatically generated |
| Se ejecuta la solución de instalación: | Graphical user interface, application, Word  Description automatically generated |
| Se continúa registrando los datos que se piden durante la instalación hasta concluir el proceso: | Graphical user interface, text, application, email  Description automatically generated |

Azure Static Web Apps CLI

Es una herramienta para el desarrollo local de Azure SWA. La información de esta herramienta se puede encontrar en la siguiente URL [GitHub - Azure/static-web-apps-cli: Azure Static Web Apps CLI ✨](https://github.com/Azure/static-web-apps-cli).

La instalación se ejecuta la siguiente línea de comando basado en npm:

npm install -g @azure/static-web-apps-cli

Text

Description automatically generated

## Git

Git se usará para el manejo de repositorios. Se puede usar la guía publicada en [Install and set up Git - Azure DevOps | Microsoft Learn](https://learn.microsoft.com/en-us/devops/develop/git/install-and-set-up-git). Se puede validar que se tiene instalado o después de la instalación ejecutando el siguiente comando:

git -v

Graphical user interface, text

Description automatically generated

Si se cuenta con una versión de git, se mostrará la información sobre la versión se tiene instalada.

## Vue

Vue es un framework de desarrollo con backgroud HTML. Se instala via Node Package Manager. Se ejecutarían dos comandos:

npm install vue

npm install -g @vue/cli

Mostrándose la terminal de la siguiente manera:

Text

Description automatically generated

Se valida que vue se encuentre instalado y en que versión.

vue -version

Graphical user interface, text, application

Description automatically generated

En caso de tener este mensaje:

Text

Description automatically generated

Se ejecuta el comando:

Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned

En el clean up puede regresarse con el siguiente comando

Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy Undefined

Se puede tener más información de esta instalación en: [Install Vue.js directly on Windows | Microsoft Learn](https://learn.microsoft.com/en-us/windows/dev-environment/javascript/vue-on-windows)

# Single Page Application (SPA)

## ¿Qué es Single Page Application?

Es una arquitectura que mantiene el procesamiento de los elementos web en un solo documento HTML.

Graphical user interface

Description automatically generated with medium confidence

Esta es la arquitectura básica de un SPAGraphical user interface

Description automatically generated

## Desplegar el código vía Vue

En esta sección se usará el Framework Vue para generar la primera SPA y se revisará las secciones con las que se cuenta esta aplicación base.

Vue puede ser generada por la línea de comando o por interfaz de Vue UI. En este caso se usará Vue UI que facilita visualmente la generación del proyecto.

|  |  |
| --- | --- |
| Para iniciar Vue UI se inicia la interfaz iniciando una línea de comando:  cmd | Graphical user interface, text, application, chat or text message  Description automatically generated |
| Al iniciar la línea de comando se recomienda ubicar en el directorio más cercano a donde se trabajará nuestro código. En mi caso se generó un folder “repos” en ruta local | A picture containing text, electronics  Description automatically generated |
| Ya en esta ubicación se iniciará el comando:  vue ui | A screen shot of a computer  Description automatically generated with medium confidence |
| Esto iniciará en un navegador la generación de proyecto | A picture containing text  Description automatically generated |
| Y muestra la siguiente interfaz | Graphical user interface, text, application  Description automatically generated |
| Se iniciará el generar el proyecto en la opción de Create/Crear | Graphical user interface, text  Description automatically generated |
| Se indica la ruta donde se genera el proyecto y se da clic en el botón inferior de “Create a new Project here” | Graphical user interface, text, application  Description automatically generated |
| Se agrega el nombre del folder del proyecto | Graphical user interface, application  Description automatically generated |
| Se dejarán las siguientes opciones de la siguiente manera  El ligar al repositorio se realizará más adelante, se da Next en este paso. | Graphical user interface, application, website  Description automatically generated |
| En la siguiente pantalla se seleccionará “Manual” para seleccionar nuestras opciones y se dará clic a “Next” | Graphical user interface, text, application  Description automatically generated |
| En la siguiente pantalla se seleccionará “Router” y se da clic a “Next”. | Graphical user interface, text, application  Description automatically generated |
| En la siguiente pantalla se habilita la opción de “history mode for router” y se selecciona la opción de Linter que se quiere manejar y se da clic en Create Project | Graphical user interface, text, application, email  Description automatically generated |
| En caso de querer generar un preset se puede asignar un nombre y guardar o se puede continuar sin guardarlo. | Graphical user interface, text, application  Description automatically generated |
| Al seleccionar una opción se generará el proyecto. | Graphical user interface, text, application  Description automatically generated |
| Al terminar se tendrá a la página de configuración del proyecto en el cual se puede regresar en cualquier momento para configurar. | Graphical user interface, website  Description automatically generated |
| Se puede regresar al cmd y cerrar con combinación de teclas “ctl+c” y cerrando el navegador. | Text  Description automatically generated |

## Ejecutar la aplicación

Posicionados en la ruta que generamos se ingresará al directorio donde está la aplicación.

|  |  |
| --- | --- |
| En el caso de ejemplo se ingresa al directorio “spaini” | Text  Description automatically generated |
| En este punto se ejecutará el comando, esto compilará la solución y nos permite revisar la aplicación localmente.  npm run serve | Text  Description automatically generated |
| Si todo es correcto, contaremos con una SPA con Vue que podemos consultar en el navegador. | Graphical user interface, text, application  Description automatically generated |

## Exploración del SPA

En este submódulo revisaremos los elementos de la SPA en vue, cómo funciona una SPA y el ruteo de los elementos.

|  |  |
| --- | --- |
| Para iniciar se ejecutará el comando desde línea de comando “code .” o abrir el folder en vs code. | A picture containing text, electronics  Description automatically generated |
| Al abrir el Vs Code es posible que pregunte si se confía en los autores se ingresará la opción de “Sí, confío en los autores” | Text  Description automatically generated |
| En VS Code se verá nuestro código de la siguiente forma.  Se expande la aplicación de directorio base y se ingresa al directorio “*public*“. Dentro de este directorio *public* está un archivo llamado *index.html*. En index.html se pude ver una sección en un tag *div* llamada *app*. Aquí es donde todas nuestras vistas serán desplegadas. | A screenshot of a computer  Description automatically generated with medium confidence |

## Solo una página index.html

Lo importante en esta página es validar que en la sección del objeto *html div* llamado app es donde se cargará las vistas que se generen.

|  |  |
| --- | --- |
| Vue es un framework que está basado en uso de *html* para ubicar las vistas. Por lo que durante la ejecución de la solución estará haciendo los cambios visuales en este *div* del contenido que queremos mostrar. | Text  Description automatically generated |

## Carga de App

Las vistas generadas en la SPA se cargan al objeto App como se explica a continuación:

|  |  |
| --- | --- |
| Los objetos se tipo .vue se cargan jerárquicamente sobre el archivo *src/main.js*. Se puede notar la relación del objeto principal *App* y la carga/monta de *#app*. | A screenshot of a computer  Description automatically generated with medium confidence |
| Este objeto se relaciona directamente con nuestro objeto *#app* que es una Vue en *src/App.vue.* | Text  Description automatically generated |

## Carga de Route

Route es un elemento relevante para el SPA. Esto debido a que nos va a ayudar a diferenciar el envío de rutas en el URL y poderlas interpretar como una vista en la SPA. En caso de que no se habilite al manejo de rutas el servidor que está entregando los archivos va a intentar localizar el objeto que se envía como parte de la *url* como un archivo, el cual no existe.

|  |  |
| --- | --- |
| Cada objeto que se requiere manejar en la URL se requiere validar en el ruteo y relacionarlo con un objeto que es una vista.  Se puede ver que se relaciona el objeto de ruteo en la misma línea de la App y *#app.* |  |
| El objeto de Router se encuentra dentro de */src/router/index.js*. Este archivo configura el *path* que se va a usar en la *url*, un nombre y la vista que se cargará en el navegador. |  |
| Un elemento que se debe revisar en este archivo es el manejo del identificador “#” en las *url* en la historia. | Text  Description automatically generated |
| Estos elementos se referencian en ligas en el objeto *App.vue* que permite enviar las vistas configuradas: | A screenshot of a computer  Description automatically generated |

## Agregar una nueva Vista

En este módulo se agrega una nueva vista a la SPA.

|  |  |
| --- | --- |
| Lo primero a realizar es agregar la sección de contacto en la carpeta de *views*. Se agregará el archivo *ContactoView.vue*. Se da clic derecho del mouse sobre el folder de vistas y se selecciona nuevo archivo. | A screenshot of a computer  Description automatically generated |
| Se agregara un nuevo template a mostrar. |  |

## Agregar la ruta a nueva vista

Para agregar la ruta modificaremos dos archivos el *App.vue* y *router/index.js*.

|  |  |
| --- | --- |
| Primero modificaremos el archivo index.js. Se agregará a la colección de rutas nuestra vista y referencias: | A screenshot of a computer  Description automatically generated |
| Se agrega al menú de navegación en *App.vue* la ruta a nuestra vista. Para esto se agrega el *router-link* en *App.vue*. |  |

## Prueba de la Vista

Usaremos para iniciar nuestra ejecución la terminal en VS Code.

|  |  |
| --- | --- |
| Se puede iniciar la terminal por medio de menú superior en Terminal o por shortcut ([us]Ctrl+` o [es]Ctrl+Shift+ñ). |  |
| En nuestra terminal ejecutamos el comando  npm run serve |  |
| Ingresamos a la dirección que nos da en la línea de comando en un navegador. | Text  Description automatically generated with low confidence |
| Aquí se puede probar la navegación y por medio de devtools validar las llamadas que se están realizando. | Graphical user interface, application, Word  Description automatically generated |

# Integración Continua y Despliegue Continuo

El código que se generó se puede integrar a Github o Azure Devops. En caso de que se cuente con una cuenta de Github se puede usar la opción 1. En caso contrario se puede optar por generar un Azure DevOps y continuar la carga del repositorio con la opción 2.

## Opción 1: Github

|  |  |
| --- | --- |
| Se selecciona la opción de “control de código fuente en la barra lateral de VS Code.  En caso de requerir conectar la cuenta a los comandos de git. Se requiere ejecutr:  git config --global user.email "hello@example.com"  git config --global username "hello"  Se da clic en la *Iniciar repositorio* | A screenshot of a computer  Description automatically generated |
| Nos pide iniciar sesión en la cuenta de GitHub. | Graphical user interface, text, application  Description automatically generated |
| Se abrirá un navegador y pedirá la autorización de acceso a recursos. Se da clic en “Autorizar Visual Studio Code” | Graphical user interface, application  Description automatically generated |
| Al terminar la autenticación, Se regresa al VS Code y pregunta el tipo de repositorio se va a usar, público o privado y también pregunta el nombre del repositorio. | A screenshot of a computer  Description automatically generated |
| Se inicia la publicación en GitHub. | A screenshot of a computer  Description automatically generated |
| Al finalizar podemos validar en GitHub que el código está listo para los siguientes pasos. | A screenshot of a computer  Description automatically generated |

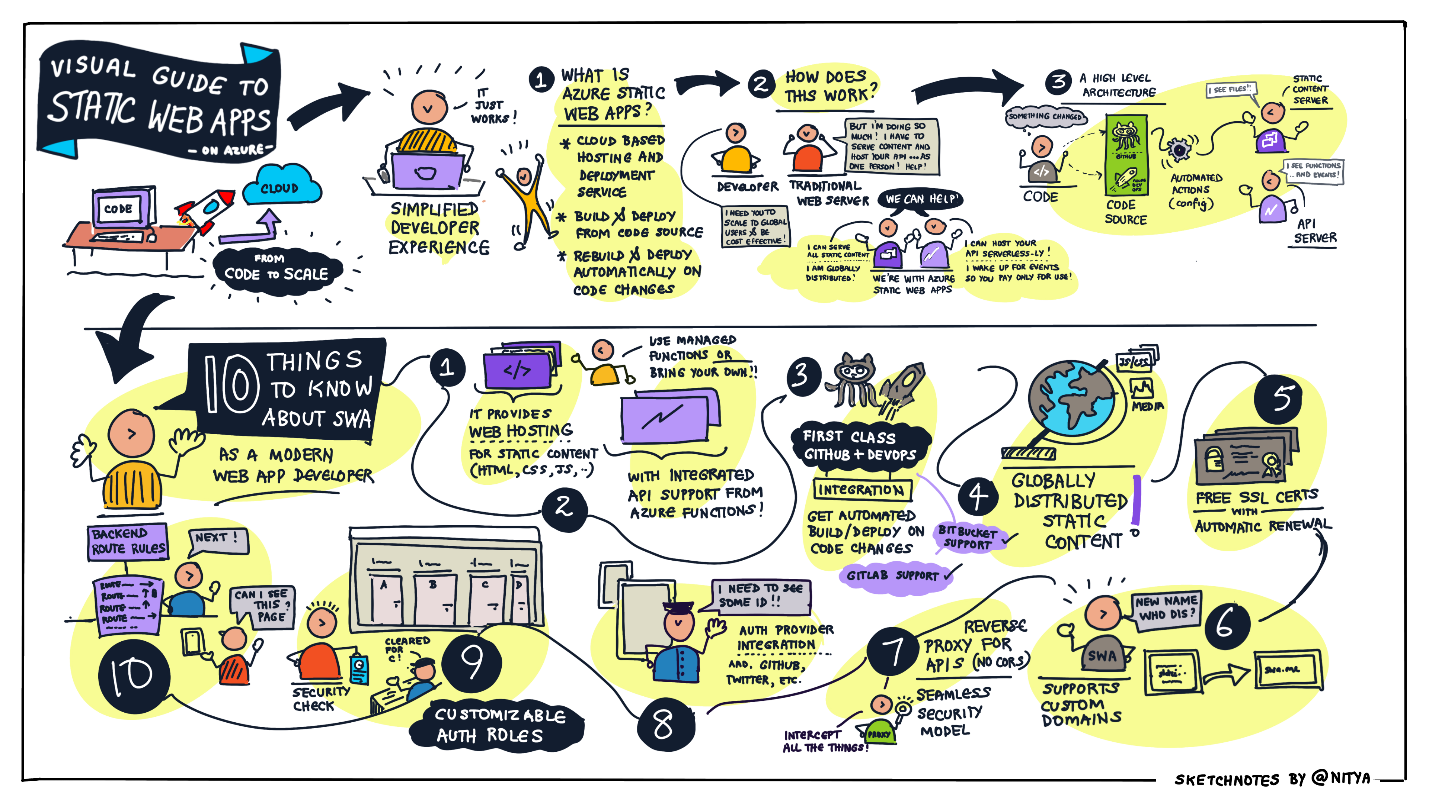
## Opcion 2: Azure Dev Ops (ADO)

|  |  |
| --- | --- |
| Generar Azure DevOps desde el portal de Azure. Se inicia en el portal de Azure y se busca la opción Devops. | Graphical user interface, text, application, email  Description automatically generated |
| Ingresando en esta sección se muestra la siguiente página y se selecciona “My Azure DevOps Organizations”. | Graphical user interface, text, application  Description automatically generated |
| Si no se cuenta con una organización se puede generar para iniciar. | Graphical user interface, text, application, Word  Description automatically generated |
| Se generará una organización en la cual se puede trabajar. | Graphical user interface, text, application, chat or text message  Description automatically generated |
| Al contar con una Organización se inicia un proyecto. Se ingresa un nombre y se da clic en Create Project. | Graphical user interface, application, Word  Description automatically generated |
| Dentro del proyecto se va a la opción de Repos y se copia la segunda opción. | Graphical user interface, text, application, email  Description automatically generated |
| Copiamos nuestro texto para hace *push* en la terminal de VS Code. | A screenshot of a computer  Description automatically generated with medium confidence |
| Pedirá autenticación a la cuenta de Azure. | Text  Description automatically generated |
| Se realiza el sync a nuestro repositorio. | A screenshot of a computer  Description automatically generated with medium confidence |
| Tras de verificar que se generó el Branch y se envió el código podemos validar en el Repos de ADO. | Graphical user interface, application  Description automatically generated |

# Azure Static Web Apps (SWA)

## ¿Qué es Azure Static Web Apps?

Los Static Web Apps es una solución PaaS que se pensó para simplificar al desarrollador el despliegue de soluciones web.



Los Azure Static Web Apps permiten el despliegue y host de aplicaciones integrando elementos de autenticación, autorización, API, Data-API y CI/CD desde su primera implementación.

Graphical user interface, text, application, chat or text message

Description automatically generated

El funcionamiento de un Azure SWA se tiene a partir de un repo hasta el host y distribución segura tipo CDN.

Graphical user interface, application

Description automatically generated

Las soluciones desplegadas soportan el uso de los frameworks como son Angular,Svelte,React, Vue y Blazor. Así como despliegue vía Gatsby como generador de sitios estáticos.



## Despliegue de la primera SWA

En este módulo desplegaremos nuestra SPA en un Azure SWA y se irán integrando los elementos de la plataforma.

|  |  |
| --- | --- |
| Dentro de Azure buscamos la opción de “Static Web apps” y la seleccionamos. | Graphical user interface, text, application, email  Description automatically generated |
| Se da clic en “Crear” | Graphical user interface, text, application, email  Description automatically generated |
| En la siguiente página llenamos el formulario con las siguientes opciones:   |  |  | | --- | --- | | Subscripción | <Selección según el combo> | | Resource Group | Generar un nuevo Resorce Group ej. RG-CA-SPA-001 | | Name | Nombre para nuestro recurso de SWA ej. SPA-CA-APP-001 | | Plan type | Free | | Region | Se selecciona una región ej. Central US | | Github Account | Se selecciona la ubicación del repositorio | | Graphical user interface, text, application  Description automatically generated |
| Se da clic en el botón de *Sign in*. En mi caso será *Github*. Y se pedirá autorización para ingresar a los recursos. | Graphical user interface, text, application  Description automatically generated |
| Al ingresar nos pide los datos del repositorio y Branch estaremos trabajando. | Graphical user interface, text, application, email  Description automatically generated |
| Damos clic a *Review + create* y validará nuestras opciones. Y por último damos *create*. | Graphical user interface, text  Description automatically generated |
| En unos momentos tendremos nuestro recurso en generado con nuestro código. | Graphical user interface, text, application, email  Description automatically generated |
| Se da clic a ir al recurso. | A screenshot of a computer  Description automatically generated with medium confidence |
| Se da clic a la URL del recurso y se valida que contamos con *url*. | Graphical user interface, text, application, Word  Description automatically generated |
| Pero el código aún se está ejecutando en el Despliegue Continuo. Se puede ir a nuestro repositorio y revisar las acciones realizadas para el despliegue. | Graphical user interface, text, application, email  Description automatically generated |
| Se ingresa al workflow que se esta ejecutamos y revisamos el build y despliegue. | Graphical user interface, text, website  Description automatically generated |
| En este caso se ve exitoso el *Action* y si recargamos la página ya se verá nuestro contenido. | Graphical user interface  Description automatically generated with low confidence |
| Regresando a nuestro Repositorio se puede ver la carpeta *.github/workflows*. En donde puede revisar cuando se ejecuta nuestro *build* y que acciones genera. | Graphical user interface, text, application, email, website  Description automatically generated |
| Aquí se puede ver que en una acción en branch *master* con un *pull request* se genera esta acción | Graphical user interface, text, application, email  Description automatically generated |

## Routing de SWA

Se agregará una configuración de *routing* adicional en el SWA.

|  |  |
| --- | --- |
| Este enrutamiento permite el control en el SWA de rutas opcionales o alias. Para esto se agrega un nuevo archivo al proyecto con nombre *staticwebapp.config.json* | A screenshot of a computer  Description automatically generated with medium confidence |
| En este archivo se encuentran dos secciones. La primera permite definir una regla de reserva (fallback) y también sus excepciones. La segundo permite el reescribir alguna ruta a otra. Si se quiere más información ver la siguiente URL [Configuración de Azure Static Web Apps](https://learn.microsoft.com/es-mx/azure/static-web-apps/configuration#fallback-routes)  {      "navigationFallback": {          "rewrite": "/index.html",          "exclude": ["\*.{css,sccs,js,png,gif,ico,jpg,svg}"]      },      "routes": [        {          "route":"/data",          "redirect":"/contact",          "statusCode": 301        }      ]  } | A screenshot of a computer  Description automatically generated |
| Se empuja el cambio a nuestro repositorio de Integración continua. | A screenshot of a computer  Description automatically generated |

## Crear un API en SWA

En VS Code usaremos la paleta de comandos. Esta se encuentra en el menú ver o Ctrl+Shft+P.

|  |  |
| --- | --- |
| Se buscan los comandos de SWA con el texto “azure static web apps” y se selecciona la opción de Create HTTP Function como se muestra a continuación. | A screenshot of a computer  Description automatically generated |
| Se selecciona el lenguaje a usar, en este caso se selecciona Javascript | A screenshot of a computer  Description automatically generated with medium confidence |
| Se selección el modelo V3. | A screenshot of a computer  Description automatically generated with medium confidence |
| Se asigna un nombre en este caso pondremos SPAAPI. | A screenshot of a computer  Description automatically generated with medium confidence |
| Abrimos el archivo que se generó en la carpeta *api/SPAAPI* llamado *index.js* y reescribimos el código con lo siguiente:  module.exports = async function (context, req) {      context.res.json({          text: "Hello from the API"      });  }; | A screenshot of a computer  Description automatically generated with medium confidence |

## Configurar carpeta API para el SWA

Ya que contamos con un API generado requerimos indicarle al Action en que carpeta se encuentran nuestros API.

|  |  |
| --- | --- |
| Se selecciona el archivo *.yml* en la carpeta *github\workflows* y se busca la sección *jobs/steps* el atributo *api\_locations* se pone el valor *api* | A screenshot of a computer  Description automatically generated with medium confidence |
| Se envía la configuración a nuestro repositorio. | A screenshot of a computer  Description automatically generated |

## Agregar un elemento de prueba

Se agrega una llamada de prueba en la vista de contacto para validar que nuestro API funciona correctamente. Agregaremos en el archivo de vista la siguiente sección de script:

<script>

export default {

  name: "App",

  data() {

    return {

      message: ""

    };

  },

  async mounted() {

    const { text } = await (await fetch("/api/info")).json();

    this.message = text;

  }

};

</script>

Viéndose de la siguiente manera dentro de la vista:

A screenshot of a computer

Description automatically generated with medium confidence

Se agrega a function.json un alias en forma opcional como un atributo *“route”*

Text

Description automatically generated

## Autenticación de SWA

Azure SWA permite una integración fácil con los siguientes:

| **Identity provider** | **Login route** |
| --- | --- |
| Azure Active Directory | /.auth/login/aad |
| GitHub | /.auth/login/github |
| Twitter | /.auth/login/twitter |

Esto quiere decir que al agregar una petición al servicio nos genera una integración a la autenticación del usuario, Este es un ejemplo de cómo realizarlo:

<a href="/.auth/login/github">Login</a>

En nuestra aplicación se agregará el *login* y *logout* para ver la integración. Se realizará por el *login* de *github* pero se puede usar cualquiera de los proveedores integrados. Para esta acción se modificará el archivo *App.vue* para agregar a la navegación las siguientes secciones:

|  |  |
| --- | --- |
| Sección en nav después de contacto. | |      <a v-if="!userInfo" :href="`/.auth/login/github?post\_login\_redirect\_uri=${redirect}`"> Login </a>      <template v-if="userInfo">          <a v-if="userInfo" :href="`/.auth/logout`"> Logout </a> |          {{ userInfo.userDetails }}      </template> |
| Sección de Script. | <script>  export default {      data() {          return {              userInfo: {                  type: Object,                  default() { },              },              redirect: window.location.pathname,          };      },      methods: {          async getUserInfo() {              try {                  const response = await fetch("/.auth/me");                  const payload = await response.json();                  const { clientPrincipal } = payload;                  return clientPrincipal;              }              catch (error) {                  console.error("No profile could be found");                  return undefined;              }          },      },      async created() {          this.userInfo = await this.getUserInfo();      }  };  </script> |
| Tras agregar estas secciones se puede ver el archivo de la siguiente forma: | A screenshot of a computer  Description automatically generated |
| Para demostración en el código se agregará en *staticwebapp.config.json* la configuración el uso de las rutas solo a usuarios autenticados. Se asegura nuestra página de contacto las siguientes rutas | ,        {          "route": "/api/\*",          "allowedRoles": ["authenticated"]        },        {          "route": "/contacto",          "allowedRoles": ["authenticated"]        } |
| Viéndose el json de la siguiente manera: | A screenshot of a computer  Description automatically generated with medium confidence |

## Acceso de usuario en API

También se puede el usuario autenticado y enviarlo a nuestra API. Cambiaremos el servicio api/index.js el siguiente código:

module.exports = async function (context, req) {

    const header = req.headers['x-ms-client-principal'];

    const encoded = Buffer.from(header, 'base64');

    const decoded = encoded.toString('ascii');

    context.res.json({

        text: JSON.parse(decoded)

    });

};

El resultado es el siguiente documento:

A screenshot of a computer

Description automatically generated with medium confidence

Y como resultado de la llamada se puede ver lo que se puede decodificar en el lado de API.

Graphical user interface, application, Word

Description automatically generated

## Roles

Azure SWA puede realizar asignación de roles por invitación a usuarios en específico. Esto se realiza en Azure en la sección de *Role Assginment* y se realiza una *invitación*.

Graphical user interface, text, application, email

Description automatically generated

Nos presenta un formulario donde se indica el proveedor de autenticación, usuario, id del rol y tiempo de expiración de invitación. Asignando este usuario con un nuevo role al final

Graphical user interface, text, application, email

Description automatically generated

Se acepta la invitación por parte del usuario y se puede ver como se ve el nuevo rol en la página de prueba.

Graphical user interface, text

Description automatically generated

## Integración con una Base de datos

Static Web Apps tiene una nueva funcionalidad en preview que permite generar un servicio rest y/o graphql automáticamente. Esto facilita el acceso y configuración a las bases de datos.

#### Generar base de datos

|  |  |
| --- | --- |
| El proceso de acceso empieza contando con una base de datos, se genera una base tipo MySQL flex | Graphical user interface, text, application, email  Description automatically generated |
| Se pide el generar una nueva base mysql del menú. | Graphical user interface, text, application, email  Description automatically generated |
| Se selecciona una base de datos tipo flex. | Graphical user interface, text, application  Description automatically generated |
| Se genera una base para Desarrollo, al final del formulario se pide un usuario y contraseña, hay que copiarlos pues más adelante se usarán | Graphical user interface, application  Description automatically generated |
| En la sección de networking es importante permitir los accesos los servicios de Azure.  Se continua con “Review + Create” y después “Create”. | Graphical user interface, text, application, email  Description automatically generated |
| Al terminar se agrega una nueva base de datos llamada *personasdb* | Graphical user interface, text, application, email  Description automatically generated |
| Se salva la nueva base de datos. | Graphical user interface, text, application, email  Description automatically generated |
| Se abre un Azure cli para agregar un par de registros. | Graphical user interface, text, application, email  Description automatically generated |
| Se conecta a la base de datos con el siguiente comando:  mysql -h <<base>>.mysql.database.azure.com -u <<usuario>>-p  Sustituyendo los parámetros de base y usuario. | Graphical user interface, text, application, email  Description automatically generated |
| Se selecciona la base de datos usando el comando “*use personasdb*”. Se genera la tabla de trabajo y un par de registros. | CREATE TABLE PersonasTabla (  Id INT AUTO\_INCREMENT NOT NULL,  Nombre VARCHAR(25) NULL,  PRIMARY KEY (Id)  );  INSERT INTO PersonasTabla (Nombre)  VALUES ('Leonel');  INSERT INTO PersonasTabla (Nombre)  VALUES ('Arturo'); |
| Teniendo una terminal con una vista similar | Graphical user interface, text, application  Description automatically generated |
| Se da “quit” y se puede cerrar el az cli. |  |

#### Configuración del código para acceso a la base

El siguiente paso es configurar al código. El primer paso es ejecutar el siguiente comando que nos genera el template de configuración del data-api.

swa db init --database-type mysql

Se puede observar que se generó un nuevo archivo en archivo *swa-db-connections/satitcwebapp.database.config.json* donde tendremos todas las configuraciones.

A screenshot of a computer

Description automatically generated with medium confidence

Se agregará la siguiente configuración al archivo:

{

"$schema": "https://dataapibuilder.azureedge.net/schemas/latest/dab.draft.schema.json",

"data-source": {

"database-type": "mysql",

"options": {

"set-session-context": false

},

"connection-string": "@env('DATABASE\_CONNECTION\_STRING')"

},

"runtime": {

"rest": {

"enabled": true,

"path": "/rest"

},

"graphql": {

"allow-introspection": true,

"enabled": true,

"path": "/graphql"

},

"host": {

"mode": "production",

"cors": {

"origins": ["http://localhost:4280"],

"allow-credentials": false

},

"authentication": {

"provider": "StaticWebApps"

}

}

},

"entities": {

"Persona": {

"source": "PersonasTabla",

"permissions": [

{

"actions": ["\*"],

"role": "anonymous"

}

]

}

}

}

Quedando de la siguiente forma:

A screenshot of a computer screen

Description automatically generated with medium confidence

Agregaremos un botón en Nuestra vista views/Contactoview.vue:

<button id="list" onclick="list()">List</button>

Teniendo esta vista

A screenshot of a computer

Description automatically generated with medium confidence

Y agregaremos un Script a public/index.html

    <script>

    async function list() {

const query = `

    {

      personas {

        items {

          Id

          Nombre

        }

      }

    }`;

const endpoint = "/data-api/graphql";

const response = await fetch(endpoint, {

    method: "POST",

    headers: { "Content-Type": "application/json" },

    body: JSON.stringify({ query: query })

});

const result = await response.json();

console.table(result.data.personas.items);

    }

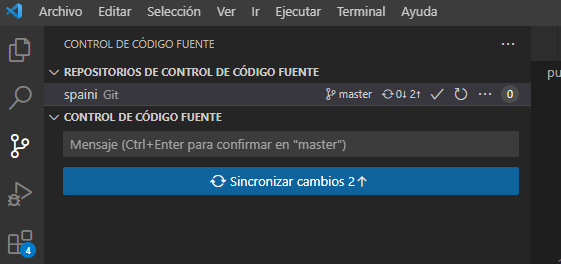
    </script>

Quedando de la siguiente manera

A screenshot of a computer

Description automatically generated with medium confidence

Enviamos nuestros cambios al repositorio.



Entramos al portal y ligamos nuestra base de datos:

Graphical user interface, text, application, email

Description automatically generated

Se da clic en *link existing database* y llenamos el formulario:

Graphical user interface, text, application, email

Description automatically generated

Entramos a nuestra SPA para dar clic en el nuevo botón y observamos en nuestras herramientas del navegador que se hizo la consulta, esta prueba usa graphql:

Graphical user interface

Description automatically generated with medium confidence

Pero se puede hacer la prueba mediante llamada de Rest:

Graphical user interface, text, application, email

Description automatically generated

## Environments

En SWA podemos tener diferentes ambientes de prueba, se pueden configurar por pull request, Branch o named. En este caso agregamos uno tipo Branch de nombre test. Se envía el cambio del archivo de acciones de git a nuestro repositorio. También se tiene que indicar cual es el production\_branch, en este caso master.

A screenshot of a computer

Description automatically generated with medium confidence

Se genera este branch en nuestro repositorio y se tendra un nuevo ambiente. Graphical user interface, text, application, email

Description automatically generated

Se puede verificar en la opcion e ambientes y se puede ver el nuevo branch que s epuede validar independiente de nuestro main branch.

Graphical user interface, text, application

Description automatically generated

Si navegamos a la url vemos que tiempo el prefijo de nuestro Branch.Graphical user interface

Description automatically generated with medium confidence

## Configurar Dominio

Si se cuenta con un dominio se puede registrar. Azure puede llegar a ayudar a adquirir dominios en App services Domain. No todas las cuentas pueden adquirir un Dominio como son el caso del Cloud Academy. La configuración es sencilla. Se entra a la opción de custom domains y se agrega add.

Graphical user interface, text, application, email

Description automatically generated

Se pide agregar un registro:

Graphical user interface, application

Description automatically generated

Agregamos el registro:

Graphical user interface, text, application, chat or text message

Description automatically generated

Y después de un tiempo se puede regresar a hacer el registro y verificarlo.

Graphical user interface, text, application, email

Description automatically generated

Se puede validar nuestro nuevo dominio agregado:

Graphical user interface, text, application, email

Description automatically generated

Y validar que se resuelve la SPA con el dominio nuevo y con ssl.

A picture containing diagram

Description automatically generated

# Progressive Web App (PWA)

## ¿Qué es Progressive Web App?

El último paso es usar nuestro SPA para iniciarlo como una PWA. El PWA busca los siguientes elementos:

Logo

Description automatically generated

Se requiere contar con elementos mínimos para un PWA.

A picture containing diagram

Description automatically generated

## Actualización de SPA a PWA

Agregamos el módulo de pwa con el siguiente comando en una terminal.

vue add pwa

Se vera de la siguiente forma:

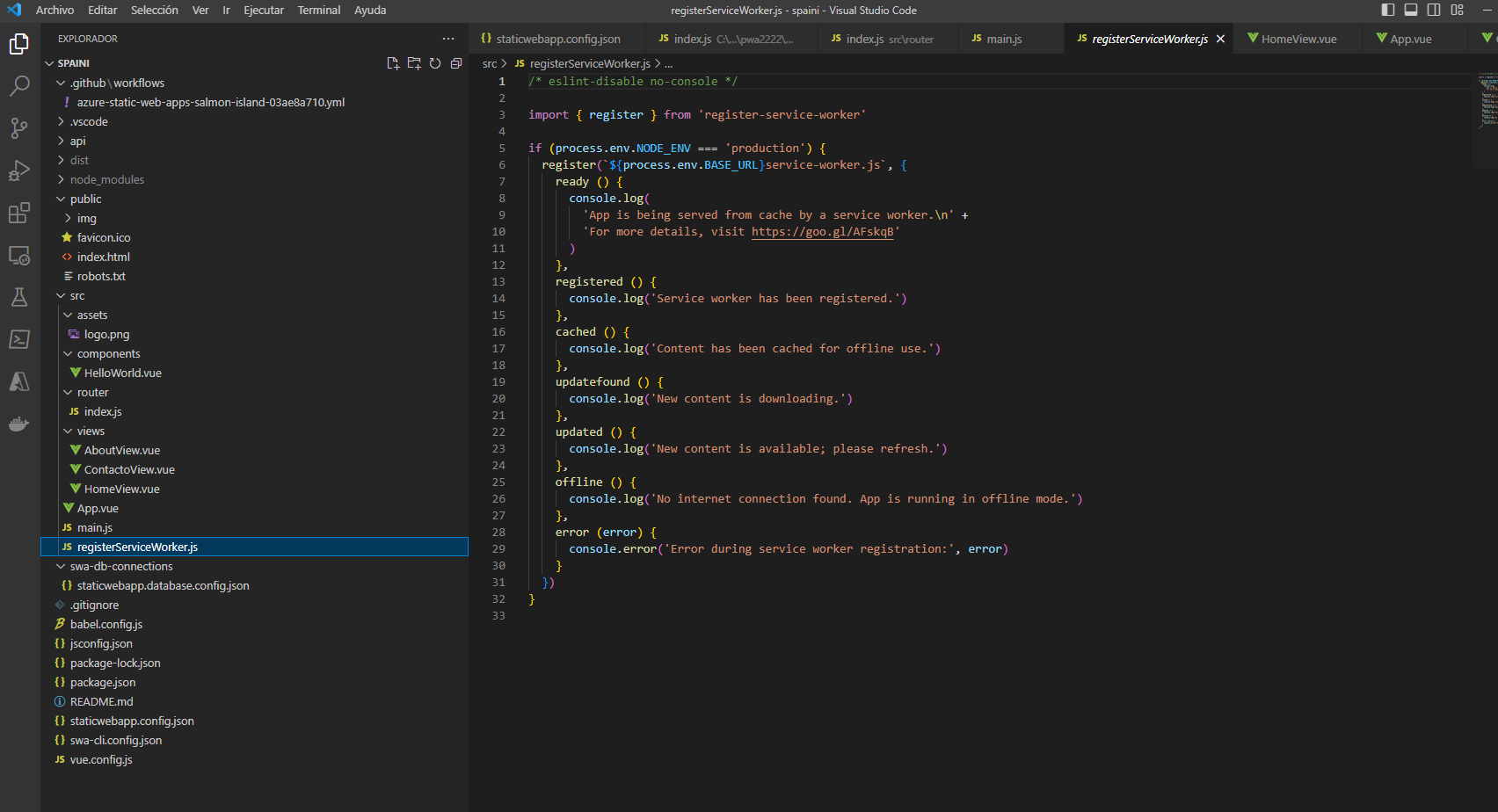
Text

Description automatically generated

### Revisión de componentes de PWA

#### Service Worker

Se agrego un nuevo archivo en src/registerServiceWorker.js donde se puede observar el ciclo de vida del Service worker y realizar acciones.



#### Manifest

El archivo *manifest* se genera hasta que se compila y se encontrará en la carpeta *dist*. Pero los elementos requeridos fueron agregados en la carpeta *public*. Si se quiere cambiar algún elemento del *manifest* file se puede copiar el compilado de *dist* a la carpeta de *public* del código y hacer las adecuaciones.

A screenshot of a computer

Description automatically generated with medium confidence

Para generar el archivo manifest se usará la terminal y se correra el siguiente comando:

npm run build

Teniendo una vista como se muestra a continuación.

Text

Description automatically generated

Al terminar se puede ver en el archivo *dist/manifest.json*

Graphical user interface, text, application

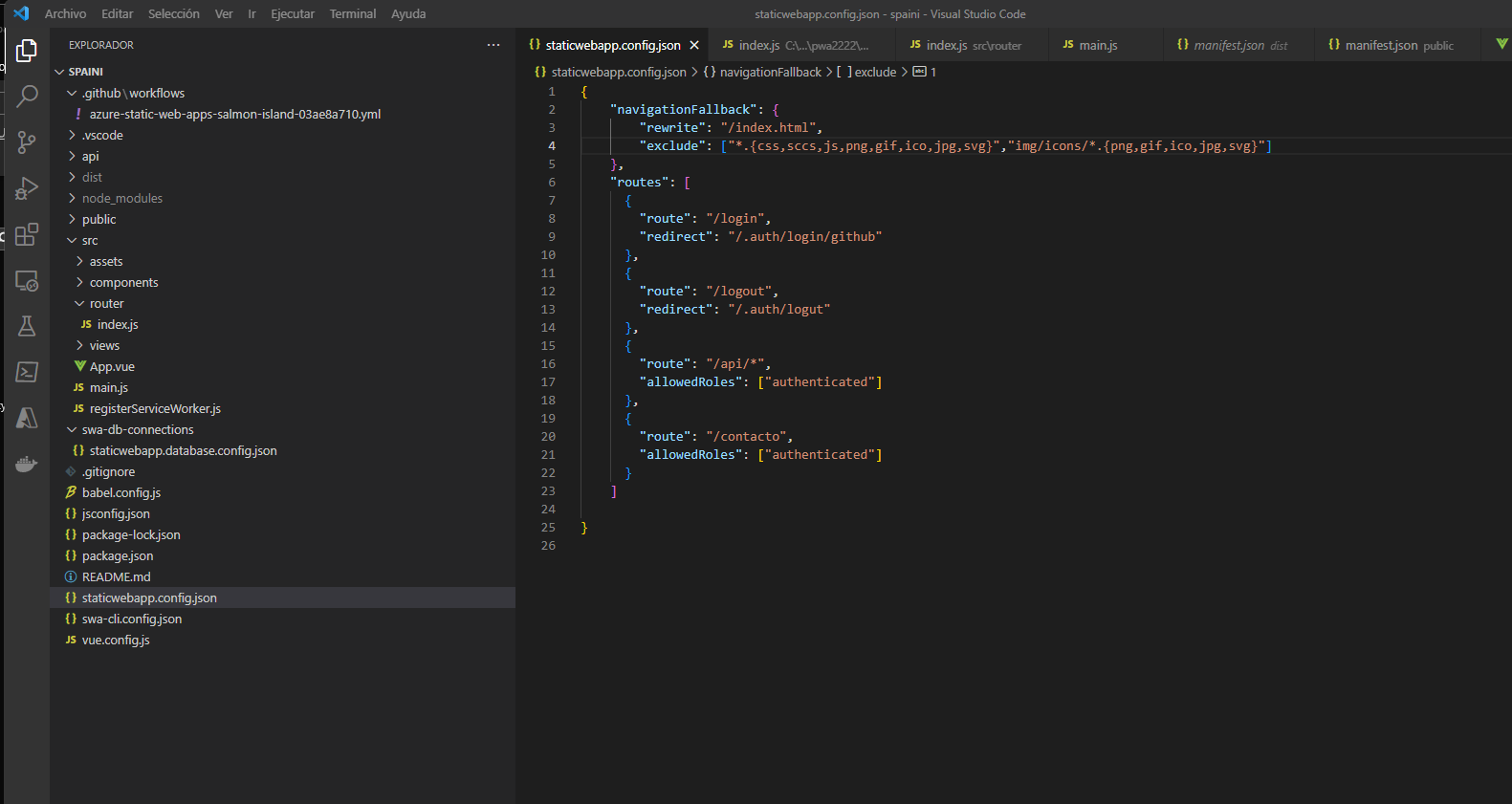
Description automatically generated

Se copia a la carpeta *public* y se pueden realizar adecuaciones si es requerido.

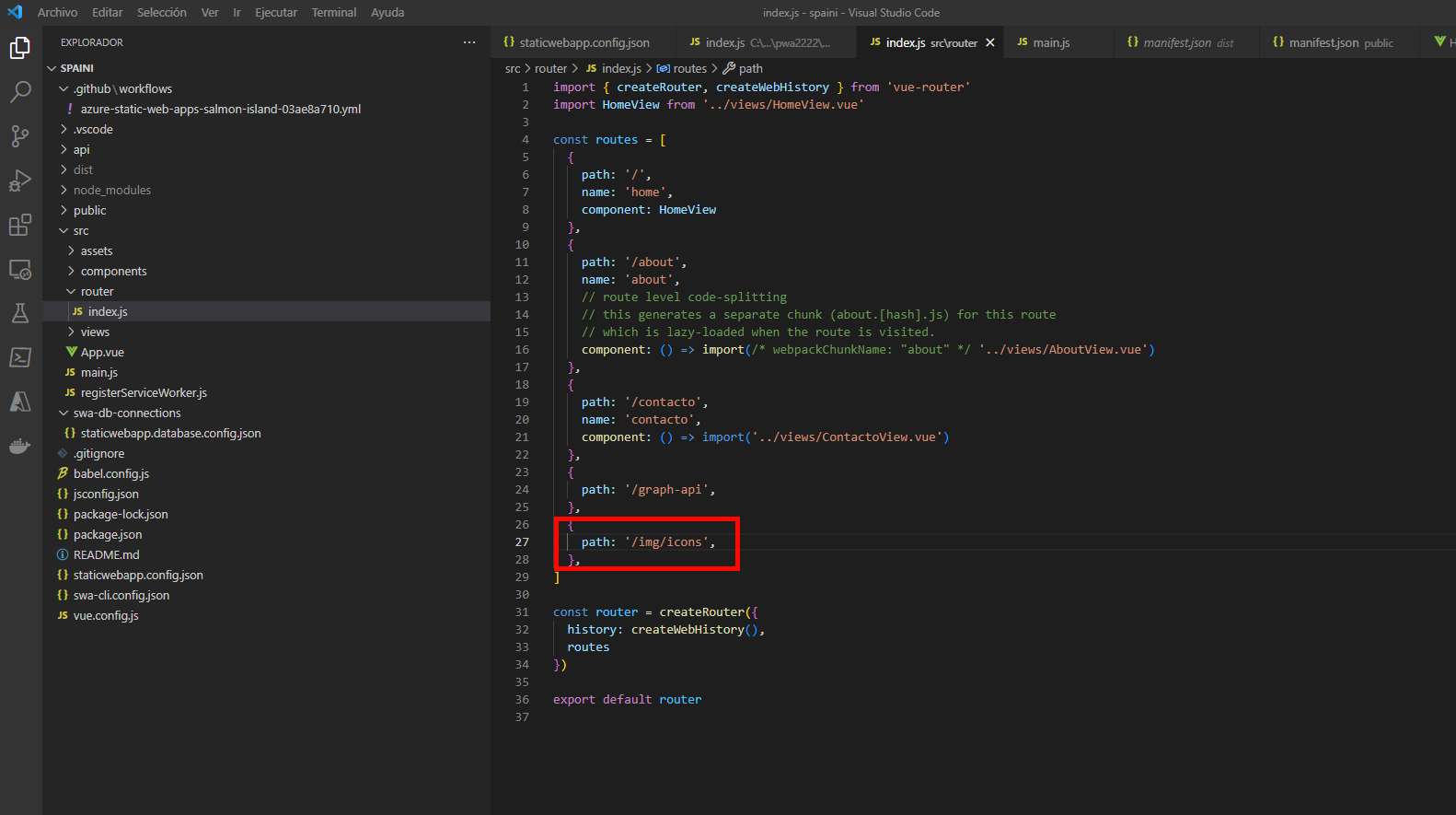
Text

Description automatically generated

También se requiere generar una excepción de rutas para nuestros nuevos elementos. Esto se hace en dos puntos. El primer punto es en nuestro archivo *staticwebapp.config.json*. Agregando en el *exclude* la nueva ruta *img/icons*.



El segundo punto es en nuestro *router* de vue. Se agrega un *path* sin componente de la misma ruta.



Se suben los cambios a nuestro repositorio.

A screenshot of a computer

Description automatically generated with medium confidence

Al desplegarse se pueden ver algunos cambios en nuestra página. Lo primero es que puede instalarse como una aplicación y adicional se puede ver en las herramientas de desarrollo una sección donde se puede examinar tanto el manifest, como el Service worker.

Text

Description automatically generated with medium confidence

## Despliegue con Builder

Contando con una PWA se puede usar la herramienta de [www.pwabuilder.com](http://www.pwabuilder.com) en donde usará nuestra *url* para validar nuestra aplicación y desplegarla en las diferentes plataformas. Ingresamos a la *url* y la pondremos en el espacio de texto y daremos *start*.

Graphical user interface, application

Description automatically generated

Después nos dará recomendaciones para que nuestra PWA cuente con un mejor comportamiento como aplicación en los dispositivos:

Graphical user interface, application, email, website

Description automatically generated

Al momento tenemos un PWA mínimo y se puede hacer pruebas en un dispositivo Windows mediante descargar el paquete de pruebas:

Graphical user interface, application, Teams

Description automatically generated

Descargamos el archivo.

Graphical user interface, text, application

Description automatically generated

El archivo se des compacta.

Graphical user interface, text, application

Description automatically generated

Se ejecuta el comando de PS *install*.

Graphical user interface, application

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated with medium confidence