Una implementación de authorization en VueJs

Agenda

- Presentación
- ¿Por qué VueJs?
- Cosas a tener en cuenta a la hora de implementar seguridad...
- Arquitectura de nuestra aplicación
- La implementación de los permisos en el Frontend
- Seguridad en las rutas
- Implementación en directivas
- Demo!

Presentación

¿Por qué VueJs?

- Proyecto con código desprolijo y con tecnologías mezcladas.
- Surgió posibilidad de nueva aplicación.

¿Y qué usamos para el Front?

- Casi nula experiencia en Frontend en el equipo.
- Prueba de concepto de Angular, React y Vue.

¿Por qué VueJs?

Angular

- (+) Sponsoreado por Google
- (+) Gran comunidad
- (-) Versiones abandonadas
- (-) Grande y con muchas cosas por aprender

React

- (+) Sponsoreado por Facebook
- (+) Rápido y eficiente
- (-) Sintaxis no standard

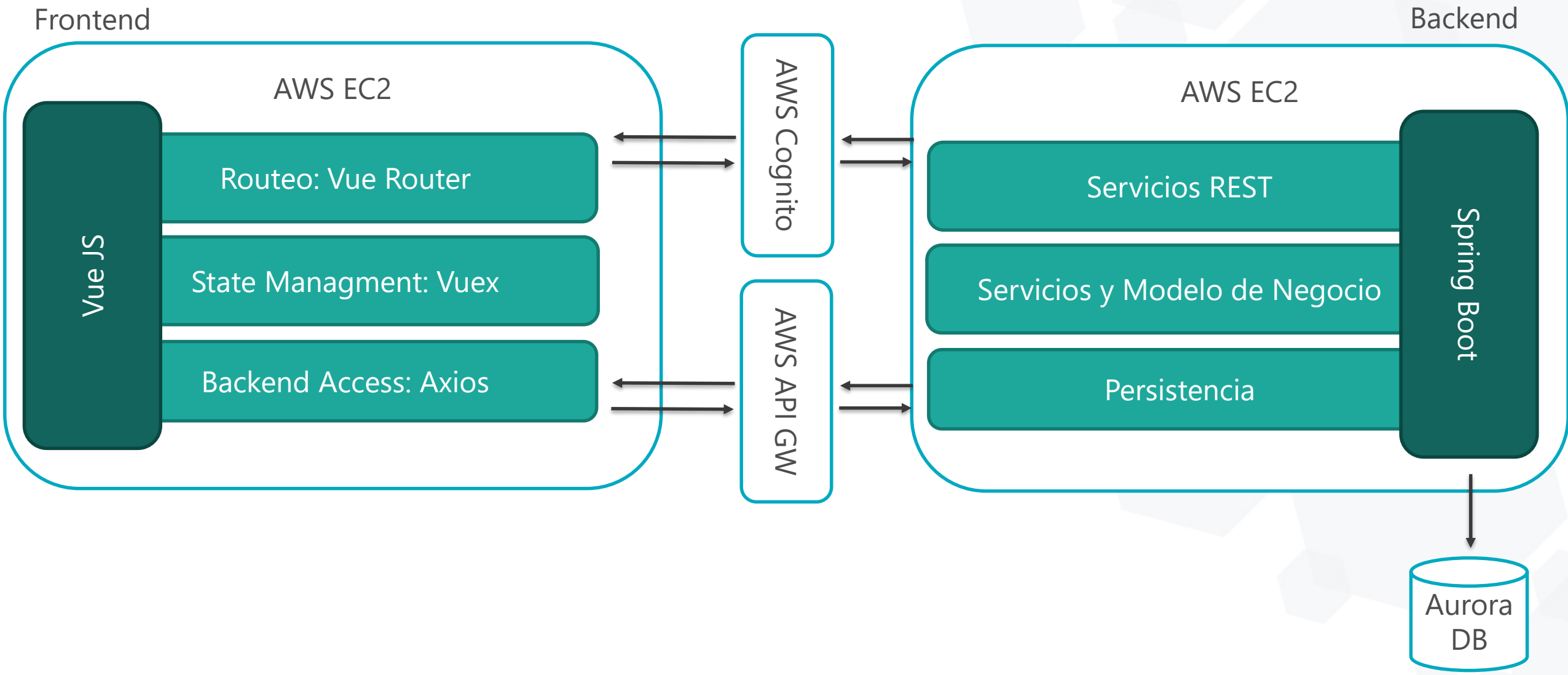
Vue

- (-) Sin grandes sponsors
- (+/-) Comunidad creciente
- (+) Rápido y eficiente
- (+) Fácil de entender y aprender

Cosas a tener en cuenta a la hora de implementar seguridad...

- Autenticar y Autorizar son dos conceptos diferentes.
- Autorizar implica validar que un usuario autenticado pueda realizar ciertas acciones y ciertas no.
- Viejos sistemas => Render en backend => No problema
- Nuevos sistemas => Backend y Frontend separados => Double Problema.
- Seguridad en los servicios del Backend => Deben poseer todas las validaciones y fallar acorde.
- Seguridad en la capa del Frontend => Debemos respetar una UX acorde. No sean como AWS.
- Tener en cuenta que el código JS reside en el browser. Siempre hay alguno que se cree hacker....
- Definir que esperamos que pase cuando un usuario no tiene permisos para una acción o una página; No mostrarla, mostrarla deshabilitada, mostrar un error....

Arquitectura de nuestra aplicación



Arquitectura de nuestra aplicación

Authentication

- AWS Cognito (Backend y Frontend)
- Pool de usuarios en base de datos y cognito.
- El frontend envía usuario y password y obtiene un token.
- Después envía ese token en cada llamada al backend.
- El backend usa el token para validar el usuario y obtener sus datos.

Authorization

- Dos tipos de permisos (Generales y Por Entidad)
- Dos clases de permisos (Aplicación y API)
- Un permiso de "Aplicación" está mapeado a varios permisos de "API"
- Un servicio de backend devuelve todos los permisos para el usuario autenticado.
- El backend utiliza Spring Security para autorizar cada uno de los servicios.
- El frontend....

La implementación de los permisos en el Frontend

- Analizamos varios plugins (vue-kindergarten, vue-auth), pero preferimos hacer la nuestra….
- No parecía tan complicado
- Desafios:
    - Guardar los permisos y permitir su rápida consulta
    - Permitir la consulta de los permisos según la entidad a visualizar
    - Permitir o no ciertas rutas en función de los permisos dados
    - Poder mostrar o deshabilitar botones en función de los permisos dados
    - Tener acceso al framework de seguridad desde cualquier componente

La implementación de los permisos en el Frontend

Guardar los permisos y permitir su rápida consulta

- Obtener los permisos del backend y guardarlos en Vuex

```
class Authorization {
  loadUser () {
    if (this._user()) {
      return Promise.resolve(this._user())
    }

    return new Promise(resolve => {
      users.me()
        .then(response => {
          store.commit('data/user', response.data)
          return resolve(response.data)
        })
    })
  }

  _user () {
    return store.getters['data/user']
  }
}
```

La implementación de los permisos en el Frontend

Guardar los permisos y permitir su rápida consulta

> Estructura del “user”

```
{
  username: String
  id: Integer
  name: String
  lastName: String
  permissions: [
    {
      code: String
      value: String
    }
  ]
  entitiesIds: [Integer]
}
```

La implementación de los permisos en el Frontend

Permitir la consulta de los permisos según la entidad a visualizar

> Generamos funciones para facilitar la consulta

```
class Authorization {
  …
  hasPermission (permissionCode) {
    return this._user().permissions.some(permission => permission.code === permissionCode)
  }

  hasEntityPermission (permissionCode, entityId) {
    const permission = this._user().permissions.find(p => p.code === permissionCode)
    if (!permission) {
      return false
    }
    if (permission.value === 'All') {
      return true
    }
    return this._user().entitiesIds.includes(entityId)
  }
}
```

Seguridad en las rutas

Agregar permisos en las rutas

- Agregamos una propiedad en cada ruta que requiera un permiso

```
{
  path: 'fields',
  name: 'Fields',
  component: Fields,
  meta: {
    permission: 'administration.ff_field'
  }
}
```

- Agregamos un método más para chequear si una ruta es permitida o no

```
class Authorization {
  …
  hasRoutePermission (route) {
    return !route.meta || !route.meta.permission || this.hasPermission(route.meta.permission)
  }
}
```

Seguridad en las rutas

Modificar Vue Router para que chequee permisos

```
Vue.use(Router)

const router = new Router({
  routes: [].concat(publicRoutes).concat(privateRoutes)
})

router.beforeEach((to, from, next) => {
  if (publicRoutes.some(route => route.name === to.name)) {
    next()
  } else if (session.isLogged()) {
    authorization.loadUser()
      .then(() => {
        if (to.matched.every(route => authorization.hasRoutePermission(route))) {
          next()
        } else {
          next(store.getters['data/homePage'])
        }
    })
} else {
    next({ name: 'login-with-redirect', params: { redirect: to.path } })
  }
})
```

Seguridad en las rutas

Resolución del homepage

```
beforeMount () {

  this.items = [obligorBlotterRoute, adminRoute, …]
    .filter(route => this.$auth.hasRoutePermission(route))

  if (this.items.length > 0) {
    this.$store.commit('data/homePage', {name: this.items[0].name})
  }

  if (this.$route.path === '/') {
    this.$router.push({name: this.items[0].name})
  }
}
```

Implementación en Directivas

Poder acceder a los permisos usando directivas

```
vue.directive(‘my-auth', function (el, binding) {
  if (!auth.hasPermission(binding.value)) {
    if (binding.arg === 'disabled') {
      el.setAttribute('disabled', 'disabled')
    } else {
      el.remove()
    }
  }
})
```

```
<button v-my-auth="‘my_permission'"/>

<button v-my-auth:disabled=“’my_permission'"/>
```

Implementación en Directivas

Poder acceder a los permisos usando directivas

```
vue.directive(‘my-entity-auth', function (el, binding) {
  if (!auth.hasEntityPermission(binding.value.permission, binding.value.entityId)) {
    if (binding.arg === 'disabled') {
      el.setAttribute('disabled', 'disabled')
    } else {
     el.remove()
    }
  }
})
```

```
button v-my-entity-auth="{permission: ’my_permission', entityId: entity.id}"/>

<button v-my-auth:disabled=“{permission: ’my_permission', entityId : entity.id}"/>
```

Implementación en Directivas

Tener acceso al framework de seguridad desde cualquier componente

```
class Authorization {
  …
  install (vue) {
    const auth = this

    vue.directive('lum-auth', function (el, binding) {
    })

    vue.directive('lum-entity-auth', function (el, binding) {
    })

    vue.prototype.$auth = auth
  }
}

export default new Authorization()
```

main.js

```
import auth from './js/configuration/authorization.js'

Vue.use(auth)
```

Contacto

emilianodelvalle@gmail.com

@bighead13

emilianodelvalle