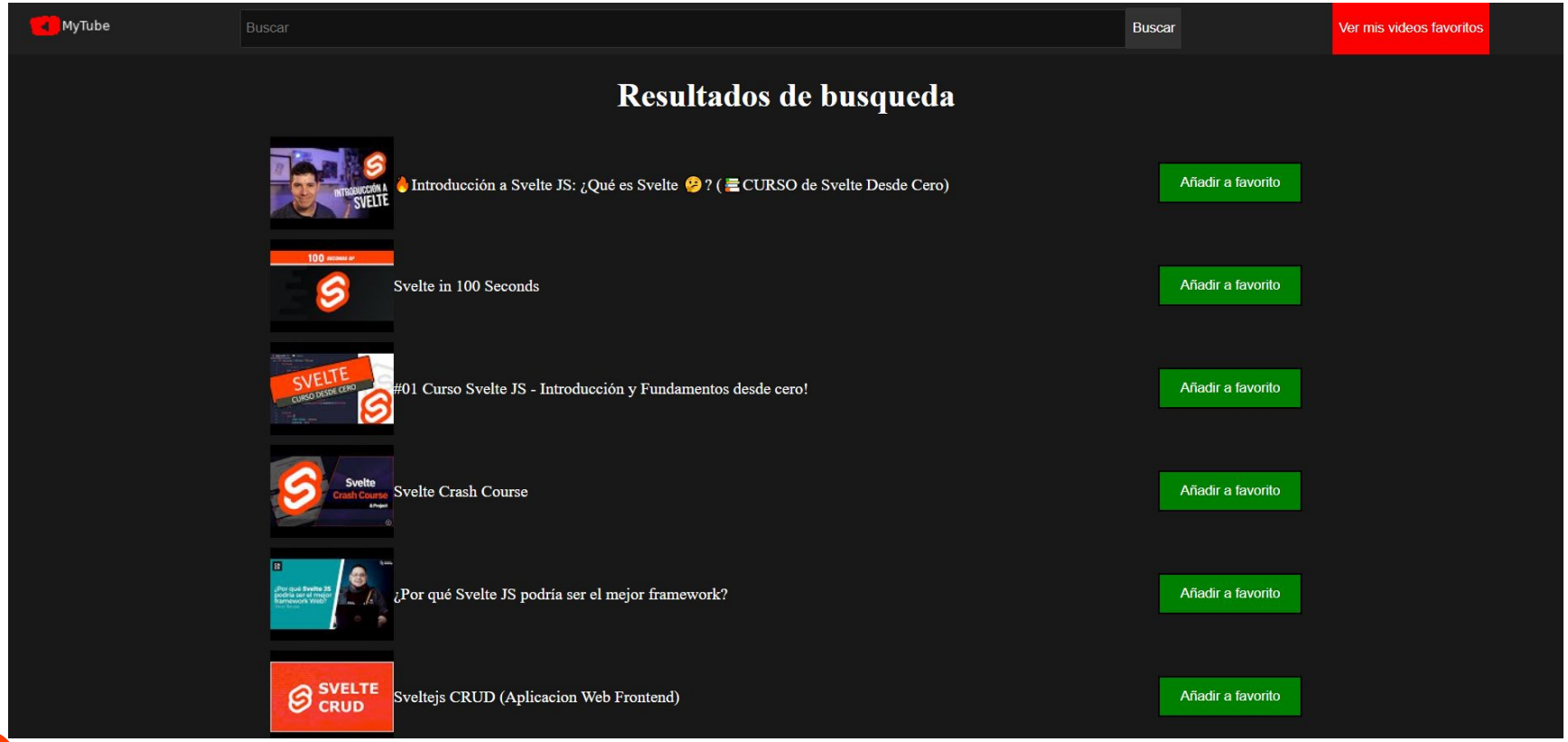
Framework para frontend: Svelte JS

Grupo 1:
Joaquín Cáceres
Julián García
Mathias Valdebenito

Aplicación realizada: MyTube



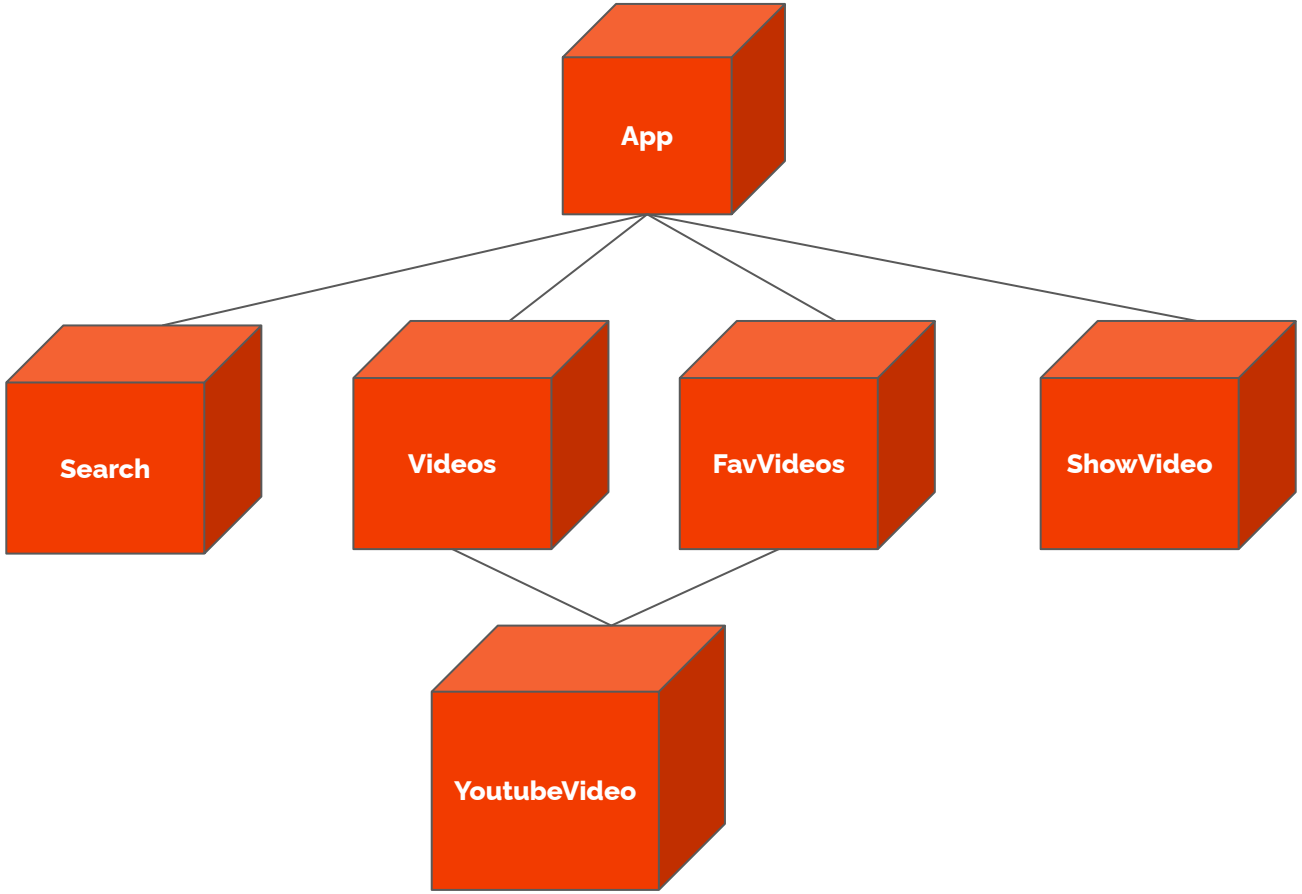
01

Demo: MyTube (versión Svelte)

Estructura aplicación

Utilizamos la misma estructura de componentes que para nuestra aplicación Vue

Estructura aplicación

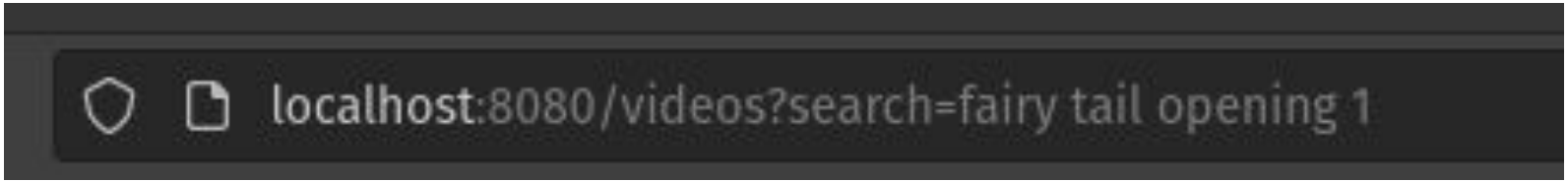


Pero agregamos una librería nueva

Svelte-Routing

```
<Router {url}>
  <Search />
  <div class="centered">
    <Route path="/videos"><Videos /></Route>
    <Route path="/"><FavVideos /></Route>
    <Route path="/watch"><ShowVideo /></Route>
  </div>
</Router>
```

Svelte-Routing



03

Análisis del código

Stores (vue)

```
state: {
  videos: [],
  currentVideo: undefined,
  currentTab: Tab.FAV,
  favVideos: window.localStorage.getItem('favVideos') !== null
    ? JSON.parse(window.localStorage.getItem('favVideos')!) : [],
},
```

Stores (svelte)

```
const { subscribe, update } = writable<Video[]>(
  window.localStorage.getItem("favVideos") !== null
    ? JSON.parse(window.localStorage.getItem("favVideos")!)
    : []
);
```

Stores (vue)

```
mutations: {
  SET_VIDEOS(state, payload: Array<Video>) {
    state.videos = payload;
  },
  CHANGE_VIDEO(state, payload: string) {
    console.log(payload)
    state.currentVideo = payload;
  },
  ADD_FAV_VIDEO(state, payload: Video) {
    state.favVideos.push(payload);
    window.localStorage.setItem('favVideos', JSON.stringify(state.favVideos));
  },
  DELETE_FAV_VIDEO(state, payload: Video) {
    state.favVideos = state.favVideos.filter((video) => video.id.videoId !== payload.id.videoId)
    window.localStorage.setItem('favVideos', JSON.stringify(state.favVideos));
  },
  CHANGE_TAB(state, payload: Tab) {
    state.currentTab = payload
  },
}
```

Stores (vue)

```
update((favVideos) => {
  const newFavVideos = [...favVideos, video];
  window.localStorage.setItem("favVideos", JSON.stringify(newFavVideos));
  return newFavVideos;
});
```

Patrón Observer

Stores (vue)

```
  actions: {
    async searchYoutube({ commit }, query: string) {
      const result = await fetch(`https://www.googleapis.com/youtube/v3/search?q=${query}&
part=snippet&maxResults=50&key=AIzaSyA-tvEokrKF-vdJuqA-MXucQclYYiivAXI`)
      const response = await result.json();
      commit("SET_VIDEOS", response.items);
      commit("CHANGE_TAB", Tab.VIDEOS);
      commit("CHANGE_VIDEO", undefined);
    },
    back({commit}) {
      commit("SET_VIDEOS", []);
      commit("CHANGE_TAB", Tab.FAV);
      commit("CHANGE_VIDEO", undefined);
    },
    showVideo({commit}, videoId: string) {
      commit("CHANGE_VIDEO", videoId);
    },
    addFavVideo({commit}, video: Video) {
      commit("ADD_FAV_VIDEO", video);
    },
    deleteFavVideo({commit}, video: Video) {
      commit("DELETE_FAV_VIDEO", video);
    }
  },
```

Stores (svelte)

```
const addFavVideo = (video: Video) =>
  update((favVideos) => {
    const newFavVideos = [...favVideos, video];
    window.localStorage.setItem("favVideos", JSON.stringify(newFavVideos));
    return newFavVideos;
  });
const deleteFavVideo = (video: Video) =>
  update((favVideos) => {
    const newFavVideos = favVideos.filter(
      (vid) => vid.id.videoId !== video.id.videoId
    );
    window.localStorage.setItem("favVideos", JSON.stringify(newFavVideos));
    return newFavVideos;
  });
```

Operador $

```
{#each $favVideos as video}
  <YoutubeVideo {video} />
{/each}
```

Cualquier cosa que defina el método subscribe puede utilizar ese operador

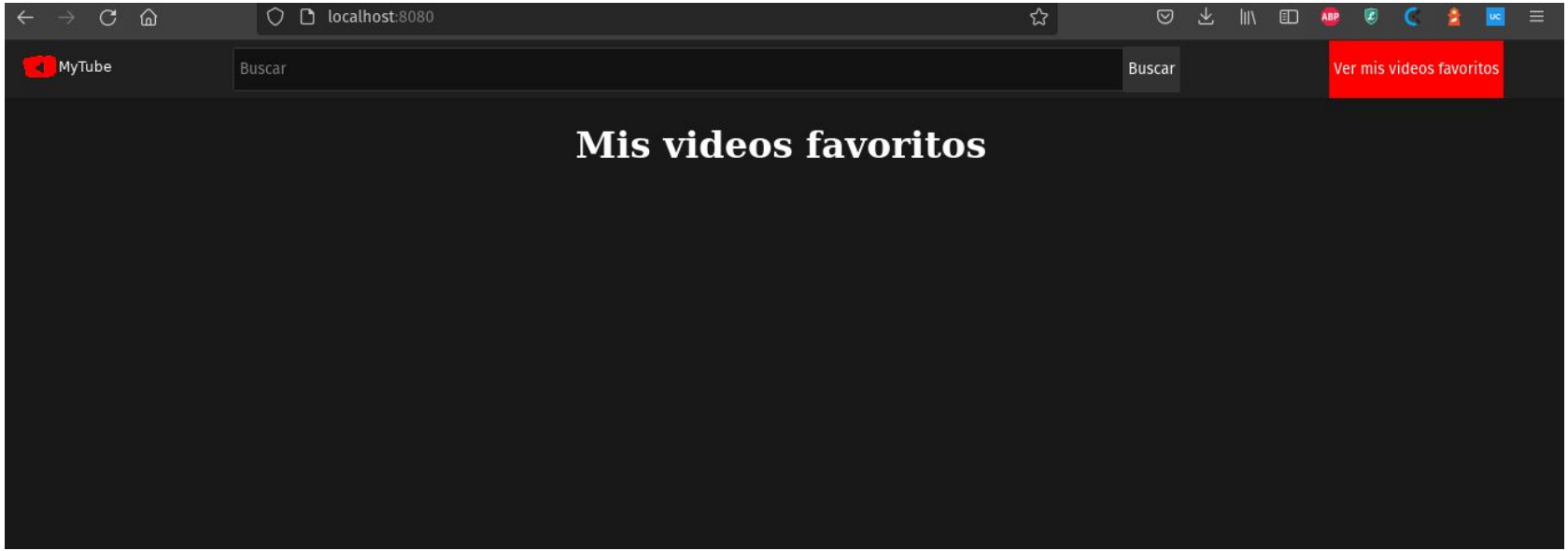
App.svelte - Routing

```
<script lang="ts">
  import { Router, Route} from "svelte-navigator";
  import FavVideos from "./components/FavVideos.svelte";
  import Search from "./components/Search.svelte";
  import ShowVideo from "./components/ShowVideo.svelte";
  import Videos from "./components/Videos.svelte";

  export const url = "";
</script>

<Router {url}>
  <Search />
  <div class="centered">
    <Route path="/videos"><Videos /></Route>
    <Route path="/"><FavVideos /></Route>
    <Route path="/watch"><ShowVideo /></Route>
  </div>
</Router>

<style>
  div.centered {
    width: 70%;
    margin: 0 auto;
  }
</style>
```

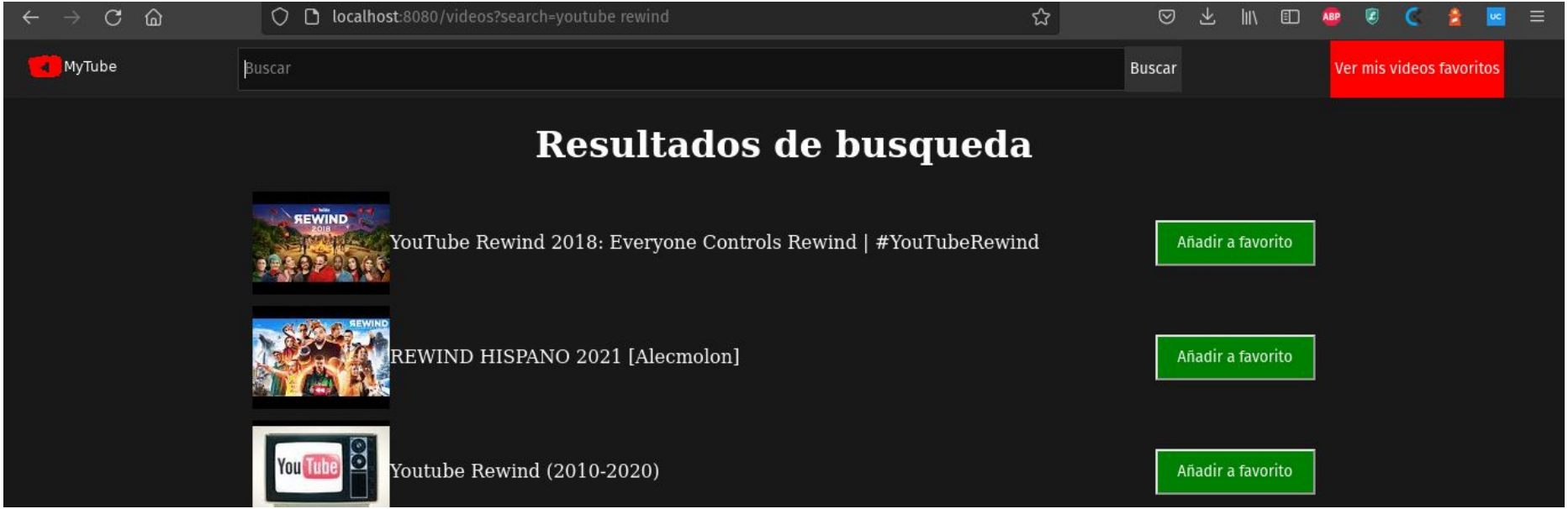
Router: permite agregar rutas y cambiar el output dependiendo de la ruta

Route: vincula un contenido a un path en el navegador

App.svelte - Routing



App.svelte - Routing



Loops y condicionales

```
<div class="w-full">
  <h1 class="title">Mis videos favoritos</h1>
  {#each $favVideos as video}
    <YoutubeVideo {video} />
  {/each}
</div>
```

Uso de each para iterar sobre la variable reactiva favVideos.
Uso de $ para subscribe y unsubscribe

```
<div class="video-title">{video.snippet.title}</div>
  {#if !$favVideos.some((vid) => vid.id.videoId === video.id.videoId)}
    <button
      class="add-fav"
      on:click|stopPropagation={() => {
        favVideos.addFavVideo(video);
      }}>
      Añadir a favorito</button>
  {:else}
    <button
      class="delete-fav"
      on:click|stopPropagation={() => {
        favVideos.deleteFavVideo(video);
      }}>
      Eliminar de favoritos</button>
  {/if}
</div>
```

Uso de los condicionales if/else

uso de on:click para invocar funciones

Loops y condicionales

MyTube

Buscar

Buscar

Ver mis videos favoritos

Resultados de busqueda

Duelo a muerte con cuchillos

Eliminar de favoritos

[comparación de Doblajes] Yu Yu Hakusho - Duelo a muerte con cuchillos

Añadir a favorito

La verdadera historia detrás del "Duelo a muerte con cuchillos" contada por los actores originales

Añadir a favorito

Romane - Duelo a muerte con cuchillos

Añadir a favorito

Uso de bind

```
<script lang="ts">
  import { navigate } from "svelte-navigator";

  let searchQuery;

  async function search(e?: KeyboardEvent) {
    if (searchQuery && ((e && e.key === "Enter") || !e)) {
      navigate(`/videos?search=${searchQuery}`);
      searchQuery = "";
    }
  }
</script>

<div class="search-bar flex">
  <div class="left">
    <button on:click={back} class="left-button">
      <img src="/MyTube.png" class="logo" alt="my-tube logo" />
    </button>
  </div>
  <div class="right flex">
    <div class="search-input">
      <input
        bind:value={searchQuery}
        class="no-display"
        on:keyup={(e) => search(e)}
        placeholder="Buscar"
      />
    </div>
```

Cambios en el input harán update de searchQuery

Bloque await / useLocation

```
<script lang="ts">
  import YoutubeVideo from "./YoutubeVideo.svelte";
  import { useLocation } from "svelte-navigator";
  const location = useLocation();
  async function searchYoutube(searchQuery: string) {
    let videos = [];
    const response = await fetch(
      `https://www.googleapis.com/youtube/v3/search?q=${searchQuery}&part=snippet&maxResults=50&
key=AIzaSyDIq4ZGvS8Z3FXABUHvBFvSAAHXMALeTmA`
    );
    const data = await response.json();
    videos = data.items;
    return videos;
  }
</script>

<div class="w-full">
  <h1 class="title">Resultados de busqueda</h1>
  {#await searchYoutube($location.search)}
    <div class="searching-text">Buscando...</div>
  {:then videos}
    {#each videos as video (video)}
      <YoutubeVideo {video} />
    {/each}
  {/await}
</div>
```

ClientWidth

```
<div class="yt-container" bind:clientWidth={w}>
  <iframe
    width={w}
    height={w / 640 * 360}
    src={`https://www.youtube.com/embed/${currentVideo}?autoplay=1`}
    title="YouTube video player"
    frameborder="0"
    allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture"
    allowfullscreen
  />
</div>
```

04

Aprendizajes

Aprendizajes

- Svelte es limpio y simple, no introduce muchas cosas nuevas, pero sí aprende de su competencia (vue, react, angular, etc) haciendo la curva de aprendizaje menor.
- No es necesario llevar un manejo complejo de los estados con hooks ni utilizar eventListeners o cosas del diaulo. El operador $ facilita mucho el trabajo con observables.
- Realmente reactivo, cualquier nueva asignación realiza una actualización automática sobre el DOM.
- Tiene una comunidad más pequeña que React o Vue.