Svelte

Grupo II

Demo

Routing

```
<script>
  import Router from 'svelte-spa-router';
  import Home from './pages/Home.svelte';
  import CallList from './pages/CallList.svelte';
  import Call from './pages/Call.svelte';

  const routes = {
    '/': Home,
    '/calls': CallList,
    '/calls/:id': Call,
  };
</script>

<Router {routes} />
```

```svelte
<script>
  import Router from './Router.svelte';
</script>

<main>
  <Router />
</main>
```

Manejo de Web Sockets

```
import { io } from 'socket.io-client';
import config from '../config';

const socket = io(config.signalingServerUrl);
export default socket;
```

```
<script>
  import socket from '../services/socket';

  onMount(() => {
    socket.on('custom-event', (data) => {...});
  });

  onDestroy(() => {
    socket.removeAllListeners();
  });
</script>
```

Manejo de sesión


import { writable } from 'svelte/store';
export const username = writable('');

```svelte
<script>
  import { push } from 'svelte-spa-router';
  import { username } from '../stores/session';
</script>

<div>
  <h1>Ingresa tu nombre de usuario</h1>
  <form on:submit|preventDefault={() => push('/calls')}>
    <input
      type="text"
      placeholder="e.g Alex"
      bind:value={$username}
    />
    <input
      type="submit"
      disabled={$username === ''}
      value="Ingresar"
    />
  </form>
</div>
```

Listado de llamadas

```svelte
<script>
  import { onDestroy, onMount } from 'svelte';
  import CallCard from '../components/CallCard.svelte';
  import socket from '../services/socket';

  let calls = [];
  onMount(() => {
    socket.emit('request-calls');
    socket.on('calls-sent', ({ calls: previousCalls }) => {
      calls = previousCalls;
    });
    socket.on('call-created', (call) => {
      calls = [...calls, call];
    });
    socket.on('call-filled', ({ id }) => {
      calls = calls.filter((call) => call.id !== id);
    });
  });
  onDestroy(() => {
    socket.removeAllListeners();
  });
</script>

<div class="container my-5">
  {#if calls.length > 0}
    <div>
      {#each calls as { id, caller } (id)}
        <CallCard
          {caller}
          on:click={() => socket.emit('join-call', { id, username: $username })}
        />
      {/each}
    </div>
  {:else}
    <p>No hay llamadas disponibles</p>
  {/if}
</div>
```

Compatibilidad entre usuarios

```
<script>
  import api from '../services/api';
  import { username } from '../stores/session';

  export let caller;
  const promise = api.get('/calculate', {
    params: {
      personA: $username,
      personB: caller,
    },
  });
</script>

<div>
  <p>{caller}</p>
  <p>
    {#await promise}
      Calculando compatibilidad
    {:then { data: { result } }}
      {result}% de compatibildiad
    {:catch}
      No ha sido posible calcular su compatibilidad
    {/await}
  </p>
  <button on:click>Unirse</button>
</div>
```

Flujo de la llamada

```svelte
<script>
  import { SyncLoader } from 'svelte-loading-spinners';
  import socket from '../services/socket';
  import { id, callee } from '../stores/call';

  let ready = false;
  let localVideo;
  let remoteVideo;
</script>

<div>
  {#if !ready}
    {#if !$callee}
      <p>Esperando a que alguien se una</p>
    {:else}
      <p>Esperando a que empiece la llamada</p>
    {/if}
    <SyncLoader color="#23d160" />
  {/if}
  <video bind:this={localVideo} />
  <video bind:this={remoteVideo} />
  {#if ready}
    <button on:click={() => socket.emit('end-call', { id: $id })}>
      Finalizar
    </button>
  {/if}
</div>
```

```
const pc = new RTCPeerConnection(servers);
let localStream = null;
let remoteStream = null;

localStream = await navigator.mediaDevices.getUserMedia({
  video: true,
  audio: true,
});

remoteStream = new MediaStream();

localStream.getTracks().forEach((track) => {
  pc.addTrack(track, localStream);
});

pc.ontrack = (event) => {
  event.streams[0].getTracks().forEach((track) => {
    remoteStream.addTrack(track);
  });
};

localVideo.srcObject = localStream;
remoteVideo.srcObject = remoteStream;
localVideo.muted = true;
localVideo.play();
remoteVideo.play();
ready = true;
```

Reutilizar lógica

React hooks
Vue composables
Svelte ??

```
import { onMount } from 'svelte';
import { writable } from 'svelte/store';

export default function onServerReady(url) {
  const { subscribe, set } = writable(false);

  onMount(async () => {
    await fetch(url);
    set(true);
  });

  return {
    subscribe,
  };
}
```

```
<script>
  import { SyncLoader } from 'svelte-loading-spinners';
  import onServerReady from './hooks/onServerReady';

  const ready = onServerReady(`${config.signalingServerUrl}/ping`);
</script>

<main>
  {#if $ready}
    <Router />
  {:else}
    <div class="center">
      <SyncLoader color="#23d160" />
    </div>
  {/if}
</main>
```

Muchas gracias