

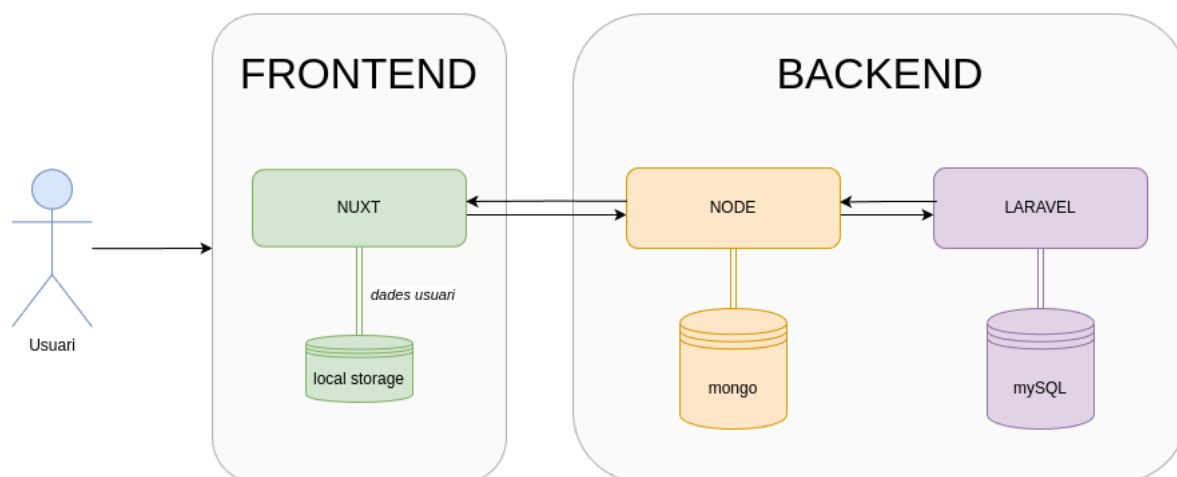
Sound O'Clock

Documentació Tècnica

Índex:

Índex:	2
Arquitectura	3
Rutes	4
Rutes GET	4
Rutes POST	6
Rutes DELETE	9
Sockets	10
Sockets Usuari	10
Sockets Administració	10
Base de Dades	12
Components Frontend	13
Administració	13
Mobile	14
Desktop	14
Documentació de codi frontend	17
Documentació de codi backend	25
Proxy Invers	29
Justificació de l'ús del proxy invers	29
Arxiu de configuració	29
Rutes del proxy invers	31
Disseny	32
Desplegament	36
Worflow de treball	36
Desplegament a la web	1

Arquitectura



Rutes

El frontend només truca a les rutes API del node, i aquest fa les crides necessàries a la BBDD de mongo o al servidor de Laravel. A continuació estan documentades totes les rutes API que implementa el servidor de Node:

Rutes GET

1. GET votacions de l'usuari

Retorna l'historial de cançons proposades i votades per un usuari en el període de votació actual.

```
fetch(`${url}/votingRecords/${id}`)
```

2. GET cançons reportades de l'usuari

Retorna la llista de cançons reportades per un usuari en el període de votació actual.

```
fetch(`${url}/reportSongs/${id}`)
```

3. GET cançons més votades per grup

Retorna la llista de grups amb les cançons més votades per grup en el període de votació actual.

```
fetch(`${url}/sortedVotedSongs`)
```

4. GET cançons proposades

Retorna la llista completa de cançons proposades en el període de votació actual.

```
fetch(`${url}/songs`)
```

5. GET cançons proposades amb els reports dels usuaris

Retorna la llista completa de cançons proposades, juntament amb els reports creats pels usuaris, en el període de votació actual.

```
fetch(`${url}/adminSongs/${token}`)
```

6. GET tots els usuaris

Retorna la llista completa d'usuaris registrats en l'aplicació.

```
fetch(`${url}/users/${token}`)
```

7. GET els grups públics

Retorna la llista dels grups públics.

```
fetch(`${url}/publicGroups`)
```

8. GET les categories públiques

Retorna la llista de les categories públiques.

```
fetch(`${url}/publicCategories`)
```

9. GET tots els grups i categories

Retorna la llista completa de tots els grups i les categories (públic i privat).

```
fetch(`${url}/allGroupsAndCategories`)
```

10. GET franjes horàries

Retorna la llista de franjes horàries.

```
fetch(`${url}/bells/${token}`)
```

11. GET informació de l'usuari

Retorna la informació completa d'un usuari basant-se en el token rebut.

```
fetch(`${url}/userInfo/${token}`)
```

12. GET tots els rols

Retorna una llista de tots els rols d'usuaris.

```
fetch(`${url}/roles/${token}`)
```

13. GET grups d'un usuari

Retorna una llista de tots grups d'un usuari, incloent els reports relacionats.

```
fetch(`${url}/userGroups/${token}`)
```

14. GET cançons seleccionades

Retorna una llista de les cançons seleccionades per aquest període.

```
fetch(`${url}/getSelectedSongs`)
```

15. GET plantilles d'administració

Retorna les plantilles utilitzades per a relacionar franjes horàries amb grups en administració.

```
fetch(`${url}/bellsGroupsTemplate`)
```

16. GET comprovació modal de la temàtica

Retorna si un usuari ha vist ja el modal de condicions que apareix quan una nova temàtica és seleccionada.

```
fetch(`${url}/checkThemeModal/${theme}/${userId}`)
```

17. GET descarregar cançons seleccionades

Descarrega les cançons que han sigut seleccionades per al següent període.

```
fetch(`${url}/selectedSongs`)
```

Rutes POST

1. POST votants d'una cançó

Retorna una llista dels usuaris que han votat per una cançó específica.

```
fetch(`${url}/usersVotes`, {  
  method: "POST",  
  headers: {  
    "Content-Type": "application/json",  
    "Accept": "application/json",  
  },  
},
```

```

    body: JSON.stringify({
      songId,
      token,
    }),
  });

```

2. POST grups d'un usuari

Afegeix els grups seleccionats a un usuari en concret.

```

fetch(`${url}/addGroupsToUser`, {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
    "Accept": "application/json",
  },
  body: JSON.stringify({
    userId,
    groups,
    token,
  }),
});

```

3. POST logout

Tanca la sessió a un usuari en concret.

```

fetch(`${url}/logout`, {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
    "Accept": "application/json",
  },
  body: JSON.stringify({
    token,
  }),
});

```

4. POST guardar cançons seleccionades

Guarda les cançons seleccionades per a que sonin el periode vinent.

```

fetch(`${url}/storeSelectedSongs`, {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
    "Accept": "application/json",
  },

```

```

    },
    body: JSON.stringify({
      token,
      songs,
    }),
  });
};

```

5. POST crear una categoria de grups

Crea una nova categoria que engloba una serie de grups.

```

fetch(`${url}/createGroupCategory`, {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
    "Accept": "application/json",
  },
  body: JSON.stringify({
    token,
    category,
  }),
});

```

6. POST crear un grup

Crea un nou grup.

```

fetch(`${url}/createGroup`, {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
    "Accept": "application/json",
  },
  body: JSON.stringify({
    token,
    group,
  }),
});

```

7. POST guardar una plantilla d'administració

Guarda una plantilla d'administració que s'utilitza per a relacionar grups amb franjes horàries.

```

fetch(`${url}/bellsGroupsTemplate`, {
  method: "POST",
  headers: {
    "Content-Type": "application/json",

```



```

    "Accept": "application/json",
  },
  body: JSON.stringify({
    template,
  }),
});

```

8. POST acceptar termes modal

Accepta els termes quan una nova temàtica ha sigut seleccionada.

```

fetch(`${url}/acceptThemeTerms`, {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
    Accept: "application/json",
  },
  body: JSON.stringify({
    theme,
    userId,
  }),
});

```

Rutes DELETE

1. DELETE plantilla d'administració

Elimina una plantilla d'administració que s'utilitza per a relacionar grups amb franjes horàries.

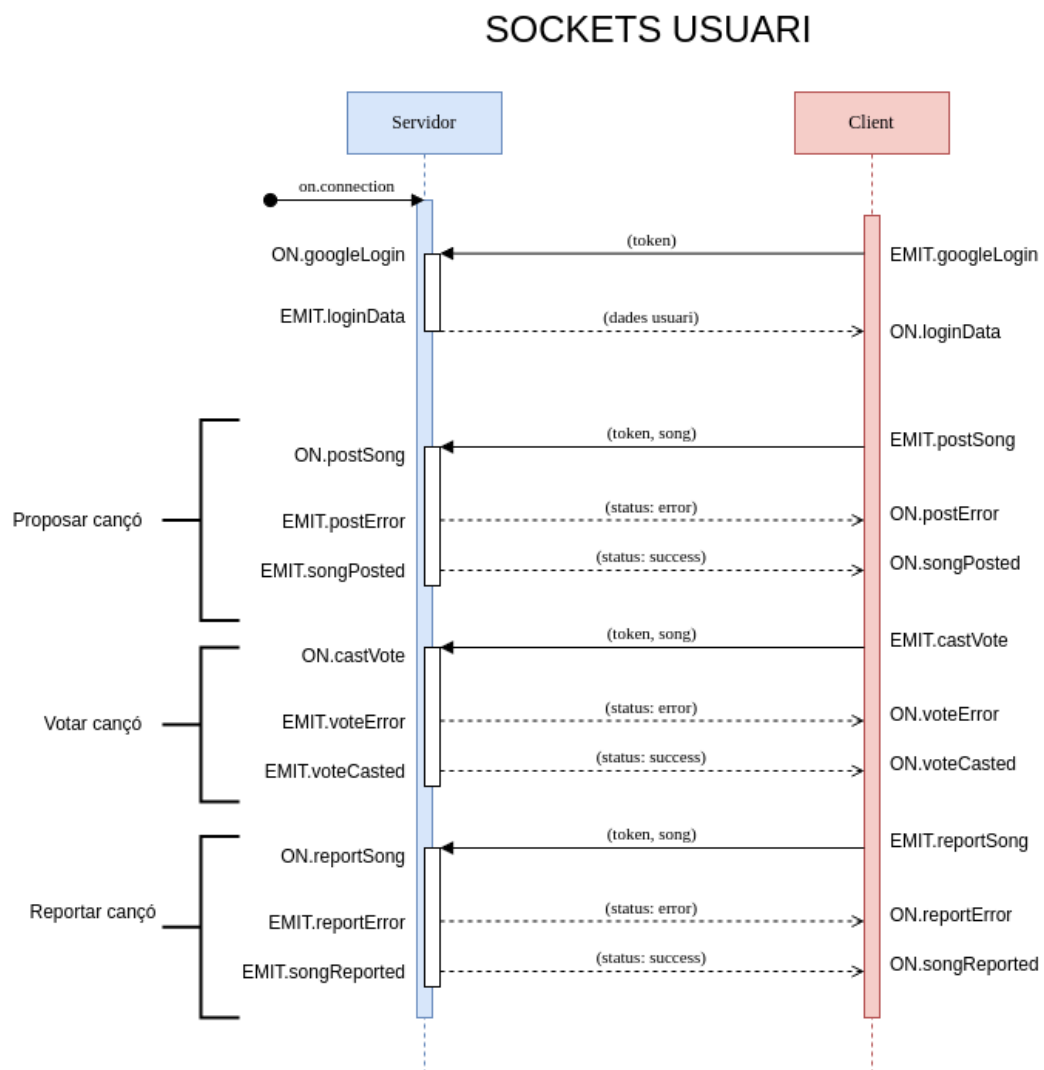
```

fetch(`${url}/bellsGroupsTemplate`, {
  method: 'DELETE',
})

```

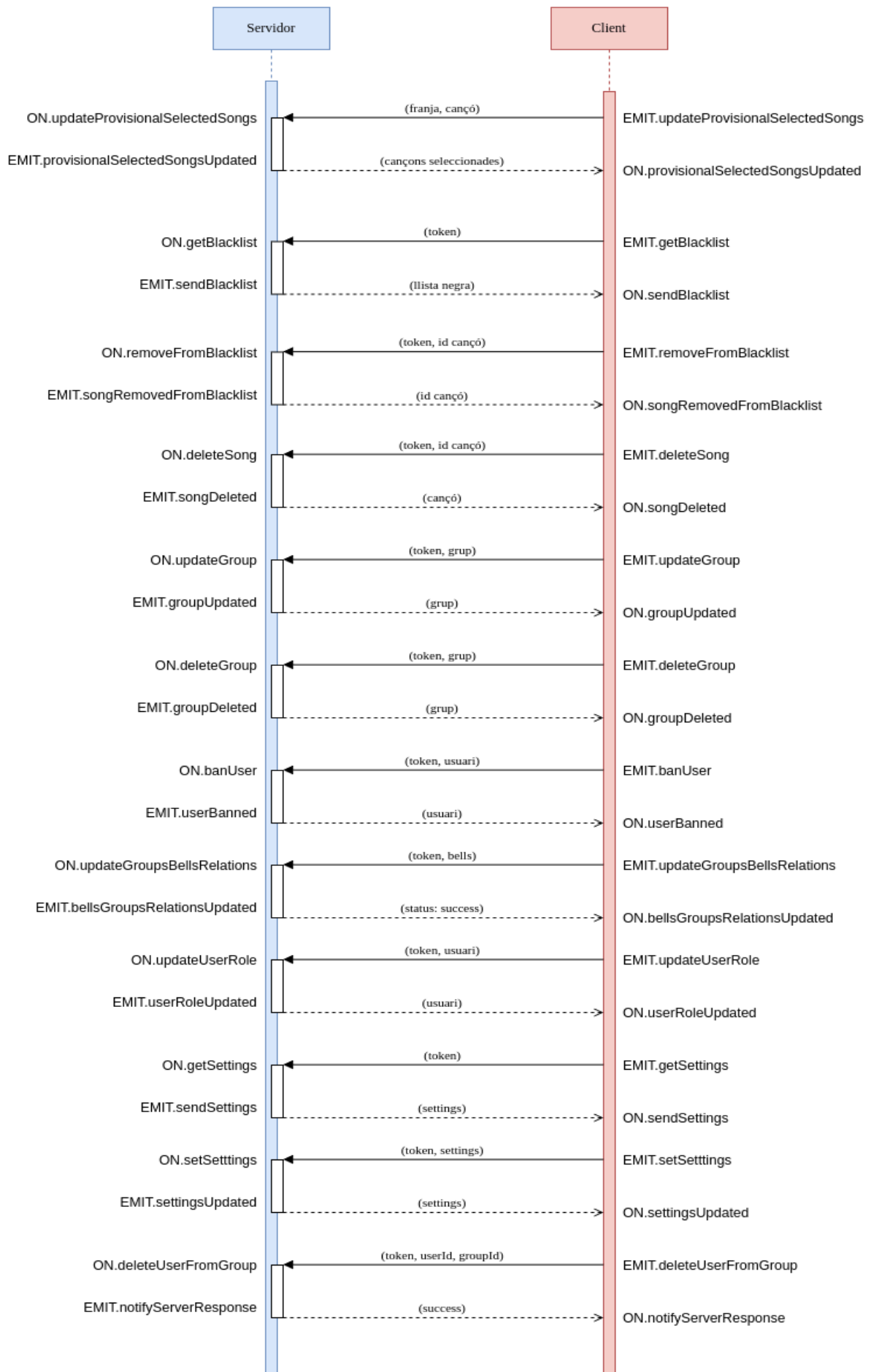
Sockets

Sockets Usuari



Sockets Administració

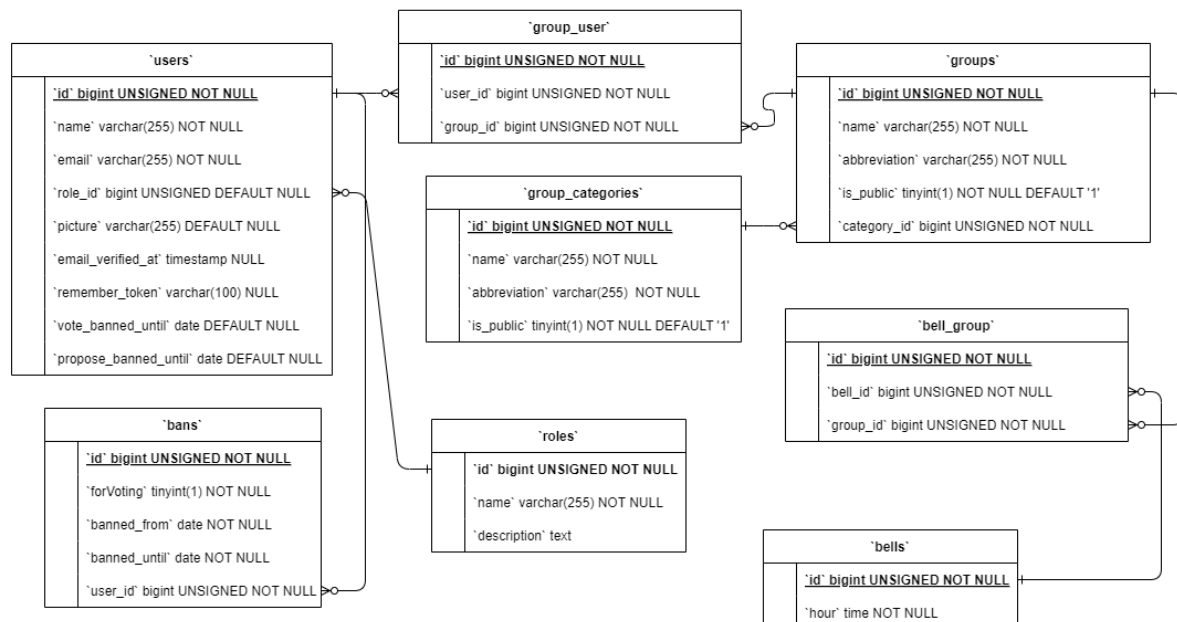
SOCKETS ADMINISTRACIÓ



Base de Dades

`settings`	
<u>`id`</u>	int UNSIGNED NOT NULL
`voteDuration`	int DEFAULT NULL
`start_vote`	date DEFAULT NULL
`end_vote`	date DEFAULT NULL
`moderationDuration`	int DEFAULT NULL
`start_moderation`	date DEFAULT NULL
`end_moderation`	date DEFAULT NULL
`showExplicit`	tinyint(1) NOT NULL DEFAULT '0'
`letProposeExplicit`	tinyint(1) NOT NULL DEFAULT '0'
`alertExplicit`	tinyint(1) NOT NULL DEFAULT '1'
`theme`	varchar(255) DEFAULT NULL
`themeDesc`	varchar(255) DEFAULT NULL
`teacher_email_key`	varchar(255) DEFAULT NULL
`student_email_key`	varchar(255) DEFAULT NULL

`blacklist`	
<u>`id`</u>	bigint UNSIGNED NOT NULL
`spotify_id`	varchar(255) NOT NULL
`name`	varchar(255) NOT NULL
`artists`	json NOT NULL
`img`	varchar(255) DEFAULT NULL
`preview_url`	varchar(255) DEFAULT NULL
`explicit`	tinyint(1) NOT NULL DEFAULT '0'



Components Frontend

Administració

L'estructura de les pàgines d'administració està feta de manera que cada pàgina és un component diferent. De manera que quan l'administrador es mou per les diferents pàgines, la URL no canvia (/admin) sino que diferents components es mostren. Aquests components d'administració son els següents:

1. alarms_crud

Aquesta pàgina és per assignar els grups existents a les franjes horàries del dia, i per a crear plantilles de distribució.

2. banHistory

Aquest component d'administració mostra el historial de bans d'un usuari en concret.

3. BanSong

Aquesta pàgina és per a mirar totes les cançons reportades i afegir cançons a la llista negra.

4. BanUser

Aquesta pàgina és per a banejar a un usuari en concret.

5. black_list_crud

Aquesta pàgina mostra la llista negra de cançons, i dona la opció de treure cançons de la llista negra.

6. grups_crud

Aquesta pàgina mostra tots els grups i les categories que els conformen, i es poden crear, editar i eliminar categories i grups.

7. manage_roles

Aquesta pàgina mostra una llista de tots els usuaris i es poden canviar els rols als usuaris.

8. script

Aquesta pàgina mostra l'estat de la màquina encarregada de posar cançons a l'Institut Pedralbes.

9. set_songs

Aquesta pàgina mostra les cançons que tenen més vots a cada franja horària i permet afegir cançons específiques a franjes, així com descarregar les cançons.

10. settings

Aquesta pàgina permet a l'administrador editar la configuració actual de les votacions i la temàtica.

11. user_roles_details

Aquest component d'administració mostra els detalls dels rols de cada usuari.

Mobile

Tenim 3 components fets especialment per a la versió mòbil, i son els següents:

1. modal

Aquest component mostra un modal per a la versió mòbil.

2. player

Aquest component mostra el player quan una cançó està sonant en versió mòbil.

3. Song

Aquest component mostra els detalls d'una cançó, així com els botons, en versió mòbil.

Desktop

La resta de components que es troben a la carpeta de "components" son per a la versió d'escriptori, i aquests son els següents:

1. modal

Aquest component mostra un modal per a la versió d'escriptori.

2. player

Aquest component mostra el player quan una cançó està sonant en versió d'escriptori.

3. switch

Aquest component mostra un *switch* (botó on/off).

4. toast

Aquest component utilitza NuxtUI per a mostrar un toast cada vegada que rep un socket en concret.

5. Calendar

Aquest component mostra un calendari per a banejar a un usuari durant un temps en concret.

6. Cercador

Aquest component mostra la barra de cerca que els usuaris utilitzen per a buscar cançons en la pàgina principal.

7. FilterButtons

Aquest component mostra uns botons per als filtres de cançons bàsics, per exemple ordenar per títol o autor.

8. header

Aquest component mostra el header de la pàgina web en casi totes les pàgines.

9. loader

Aquest component senzillament és una petita animació per indicar que alguna cosa està carregant.

10. navbar

Aquest component mostra el menú superior dins del header amb links a les diferents pàgines.

11. SearchWithFilters

Aquest component mostra el menú lateral dels filtres detallats que els usuaris utilitzen per a filtrar cançons en la pàgina principal.

12. sideBarMenu

Aquest component mostra el menú lateral de navegació per a la versió de mòbil.

13. Song

Aquest component mostra els detalls d'una canço, així com els botons corresponents, específicament de cançons a les pàgines dels usuaris normals.

14. SongAdmin

Aquest component mostra els detalls d'una canço, així com els botons corresponents, específicament de cançons a les pàgines d'administració.

15. SongDetails

Aquest component mostra els detalls d'una cançó quan ha sigut reportada a la pàgina d'administració, així com els botons per a banejar la cançó.

16. songsPreview

Aquest component mostra les cançons que estan sonant actualment durant el dia.

17. star

Aquest component és simplement una estrella per a la decoració de la landing page.

18. UserDetails

Aquest component mostra els detalls d'un usuari, així com els botons per a banejar al usuari per un temps determinat.

19. vinyl

Aquest component mostra un vinil amb la imatge de la cançó que està sonant, només per a la versió d'escriptori.

Documentació de codi frontend

A continuació adjuntem el codi documentat d'un component de Nuxt.js. En aquest cas, és el component que `songsPreview` que s'encarrega de mostrar les dades de l'última cançó que ha sonat, l'anterior i la següent.

```
<template>
  <div class="w-[85%] h-fit max-h-56 z-[999]">
    <div v-if="selectedSongs">
      <div class="px-2 title">ÚLTIMES CANÇONS SONANT</div>
      <div class="max-h-56 overflow-y-scroll">
        <div v-for="song in selectedSongs">
          <component :is="activeSong" :key="song.id"
:track="song" :currentTrackId="songStatus.currentTrackId"
          :isPlaying="songStatus.isPlaying"
@play="playSong" :type="'selected'" class="w-full"
          :bellId="song.bellId" :isNext="nextBellId ===
song.bellId" />
        </div>
      </div>
    </div>
  </div>
</template>

<script>
import { useAppStore } from '@stores/app';
import comManager from '../communicationManager';

/**
 * Component that displays the data of the last song that has sounded,
the previous one and the following one.
 * @component
 */
export default {
  name: 'songsPreview',
  data() {
    return {
      store: useAppStore(),
    }
  },
  /**
   * An array of song objects. Each song object should have
`hour` and `bellId` properties.
   * @type {Array<Object>}

```

```

        */
        selectedSongs: null,
    /**
     * Id of the next bell that will sound.
     * @type {number}
     */
    nextBellId: null,
    /**
     * Device type. 1 if it is mobile; 0 if it is desktop.
     * @type {number}
     */
    mobileDetector: this.$device.isMobile ? 1 : 0,
    },
    async created() {
        await comManager.getSelectedSongs();
        await comManager.getBells();
    },
    methods: {
        /**
         * Calculate the current time and format it as a string in
"HH:MM" format.
         *
         * This method retrieves the current date and time, extracts
the hour and minute components,
         * formats them to always be two digits (padding with a leading
zero if necessary), and
         * returns the formatted time as a string.
         *
         * @returns {string} - The current time in "HH:MM" format.
         */
        calculateNow() {
            // Get the current time
            const now = new Date();
            const currentHour = now.getHours();
            const currentMinutes = now.getMinutes();

            // Convert current time to a comparable format (HH:MM)
            const currentTime = `${String(currentHour).padStart(2,
'0')}:${String(currentMinutes).padStart(2, '0')}`;
            return currentTime;
        },
    /**

```

```

        * Determine the most recent bell that has rung up to the
current time.
    *
    * This method iterates through a list of bell times and finds
the index of the last bell
    * that has a time less than or equal to the provided current
time.
    *
    * @param {string} now - The current time in "HH:MM" format.
    * @returns {number} - The index of the most recent bell that
has rung. Returns -1 if no bells have rung yet.
    */
    calculateCurrentBell(now) {
        let lastIndex = -1;

        for (let i = 0; i < this.bells.length; i++) {
            if (this.bells[i].hour <= now) {
                lastIndex = i;
            } else {
                break;
            }
        }

        return lastIndex;
    },
    /**
    * Get the IDs of the bell at the given index and its adjacent
bells.
    *
    * This method retrieves the IDs of the bell at the specified
index, as well as the IDs of the
    * preceding and following bells if they exist. It returns an
array containing these IDs.
    *
    * @param {number} i - The index of the current bell in the
`bells` array.
    * @returns {Array<string>} - An array of bell IDs, including
the ID of the bell at index `i`
    * and the IDs of its adjacent bells if they exist.
    */
    getAdjacentBells(i) {

```

```

        let showedBells = [];
        if (i > 0) {
            showedBells.push(this.bells[i - 1].id);
        }

        showedBells.push(this.bells[i].id);

        if (i < this.bells.length - 1) {
            showedBells.push(this.bells[i + 1].id);
        }

        return showedBells;
    },
    /**
     * Retrieve songs associated with the given bell IDs.
     *
     * This method filters the `finalSongsList` array to find songs
that are associated with any
     * of the bell IDs provided in the `bellsId` array. It returns
an array of these songs.
     *
     * @param {Array<string>} bellsId - An array of bell IDs for
which the corresponding songs are to be retrieved.
     * @returns {Array<Object>} - An array of song objects that are
associated with the given bell IDs.
     */
    getShowedSongs(bellsId) {

        let showedSongs = [];
        for (let i = 0; i < this.finalSongsList.length; i++) {
            if (bellsId.includes(this.finalSongsList[i].bellId)) {
                showedSongs.push(this.finalSongsList[i])
            }
        }

        return showedSongs;
    },
    /**
     * Links songs with their corresponding bell hours and sorts
them by hour.
     *
     * @param {Array<Object>} songs - An array of song objects.
Each song object should have a `bellId` property.

```

```

        * @returns {Array<Object>} - The array of song objects with an
added `hour` property, sorted by hour.
    */
    linkSongsWithBells(songs) {

        // Iterate over each song in the provided songs array
        for (let i = 0; i < songs.length; i++) {

            // Iterate over each bell in the bells array of the
component
            for (let j = 0; j < this.bells.length; j++) {

                // Check if the bellId of the current song matches
the id of the current bell
                if (songs[i].bellId == this.bells[j].id) {
                    // If there's a match, assign the hour of the
bell to the hour property of the song
                    songs[i].hour = this.bells[j].hour;
                }

            }

        }

        // Sort the songs array by the hour property in ascending
order
        songs.sort((a, b) => {
            return a.hour.localeCompare(b.hour);
        });

        // Return the sorted songs array
        return songs;
    },
    /**
    * Calculates and updates the showed bells and songs based on
the current time.
    */
    calculateShowedBells() {

        const now = this.calculateNow();
        const currentBell = this.calculateCurrentBell(now);
        const showedBellsId = this.getAdjacentBells(currentBell);

```

```

        const showedSongs = this.getShowedSongs(showedBellsId);
        this.selectedSongs = this.linkSongsWithBells(showedSongs);
        this.nextBellId = this.findNextBell(now,
this.selectedSongs);

    },
    /**
     * Finds the next bell based on the current time and the
provided songs.
     *
     * @param {string} now - The current time in a format comparable
to the song's hour property.
     * @param {Array<Object>} songs - An array of song objects. Each
song object should have `hour` and `bellId` properties.
     * @returns {string|null} - The `bellId` of the next bell, or
`null` if no future bell is found.
     */
    findNextBell(now, songs) {

        let bellId = null

        // Iterate over each song to find the next bell
        for (let i = 0; i < songs.length; i++) {
            if (songs[i].hour > now) {
                bellId = songs[i].bellId;
            }
        }

        return bellId;
    },
    /**
     * Plays a song by triggering the store's playTrack method.
     *
     * @param {Object} track - The track object to be played. The
structure of the track object depends on the store's playTrack method.
     */
    playSong(track) {
        this.store.playTrack(track);
    },

},
watch: {
    'bells'() {

```

```

        this.calculateShowedBells();
    },
},
computed: {
    bells() {
        return this.store.getBells();
    },
    finalSongsList() {
        return this.store.getFinalSongsList();
    },
    songStatus() {
        return this.store.getSongStatus();
    },
    activeSong() {
        return this.songComponent[this.mobileDetector];
    },
}
}
</script>

<style scoped>
.title {
    --text-divider-gap: 0.5rem;
    display: flex;
    align-items: center;
    font-size: 0.9375rem;
    text-transform: uppercase;
    letter-spacing: 0.1em;
}

.title::before,
.title::after {
    content: '';
    height: 1px;
    background-color: silver;
    flex-grow: 1;
}

.title::before {
    margin-right: var(--text-divider-gap);
}

.title::after {

```

```
margin-left: var(--text-divider-gap);  
}  
</style>
```


Documentació de codi backend

El següent és un exemple del codi documentat, en aquest cas és el controlador de Laravel que s'encarrega de administrar els rols en l'aplicació.

```
class RolesController extends Controller
{
    /**
     * Display a listing of all roles.
     *
     * Retrieves all roles from the database and returns them as a collection.
     * This method is typically used to fetch and display all roles in a list view.
     *
     * @return \Illuminate\Database\Eloquent\Collection Returns a collection of Role
models.
     */
    public function index()
    {
        return Role::all();
    }

    /**
     * Store a newly created role in storage.
     *
     * Validates the incoming request to ensure that the user has the necessary admin
permissions
     * and that the required fields are provided. If validation passes, a new role is
created
     * in the database. Otherwise, an error response is returned.
     *
     * @param \Illuminate\Http\Request $request The incoming request object, expected to
contain
     *                                     'name' and 'description' fields for the
role.
     * @return \Illuminate\Http\Response Returns a JSON response with the newly created
role
     *                                     or an error message if the user lacks permissions.
     *
     * @throws \Illuminate\Validation\ValidationException If validation fails on 'name' or
'description'.
     */
    public function store(Request $request)
    {
        // Validate that user is admin
        if (auth()->user()->role_id !== 1 && auth()->user()->role_id !== 2) {
            return response()->json([
                'status' => 'error',
                'message' => 'You do not have admin permissions.'
            ], 404);
        }

        $fields = $request->validate([
```

```

        'name' => 'required|string',
        'description' => 'required|string',
    ]);

    return Role::create($request->all());
}

/**
 * Display the specified role.
 *
 * Retrieves and returns the details of a specific role from the database based on the
provided ID.
 * This method uses the `findOrFail` method, which throws an exception if no role with
the given ID exists,
 * ensuring that the function either returns a valid role or halts with a 404 not found
error.
 *
 * @param string $id The unique identifier of the role to be displayed.
 * @return \Illuminate\Database\Eloquent\Model Returns the Role model instance with the
specified ID.
 *
 * @throws \Illuminate\Database\Eloquent\ModelNotFoundException If no role is found for
the provided id,
 *
 * this exception is
thrown automatically by `findOrFail`.
 */
public function show(string $id)
{
    $role = Role::findOrFail($id);
    return $role;
}

/**
 * Update the specified role in storage.
 *
 * Validates the incoming request for admin permissions and required fields. If the
user has
 * the necessary permissions and the role exists, it updates the role with the new
data.
 * Returns the updated role or an error response if the user lacks permissions or the
role does not exist.
 *
 * @param \Illuminate\Http\Request $request The incoming request object, containing
data to update the role.
 * @param string $id The unique identifier of the role to be updated.
 * @return \Illuminate\Http\Response Returns a JSON response containing the updated
role data
 *
 * or an error message if the user lacks permissions
or the role is not found.
 *
 * @throws \Illuminate\Validation\ValidationException If validation fails on required
fields.

```

```

    * @throws \Illuminate\Database\Eloquent\ModelNotFoundException If no role is found for
the provided id.
    */
    public function update(Request $request, string $id)
    {
        // Validate that user is admin
        if (auth()->user()->role_id !== 1 && auth()->user()->role_id !== 2) {
            return response()->json([
                'status' => 'error',
                'message' => 'You do not have admin permissions.'
            ], 404);
        }

        $fields = $request->validate([
            'name' => 'required|string',
            'description' => 'required|string',
        ]);

        $role = Role::findOrFail($id);
        $role->update($request->all());
        return $role;
    }

    /**
     * Remove the specified role from storage.
     *
     * Validates the incoming request for admin permissions and checks if the specified
role exists.
     * If the user has the necessary permissions and the role exists, it deletes the role
from the database.
     * Returns a success response on deletion or an error response if the user lacks
permissions or the role does not exist.
     *
     * @param string $id The unique identifier of the role to be deleted.
     * @return \Illuminate\Http\Response Returns a JSON response indicating success or
failure of the deletion process.
     *
     * @throws \Illuminate\Database\Eloquent\ModelNotFoundException If no role is found for
the provided id.
     */
    public function destroy(string $id)
    {
        // Validate that user is admin
        if (auth()->user()->role_id !== 1 && auth()->user()->role_id !== 2) {
            return response()->json([
                'status' => 'error',
                'message' => 'You do not have admin permissions.'
            ], 404);
        }

        // Validate that role exists
        $role = Role::find($id);
        if (!$role) {

```

```
        return response()->json([
            'status' => 'error',
            'message' => 'Role not found'
        ], 404);
    }

    $role->delete();
    return response()->json([
        'status' => 'success',
        'message' => 'Role deleted successfully'
    ]);
}
}
```

Proxy Invers

La nostra aplicació web s'ha desplegat amb Docker, per tant, per facilitar la comunicació entre els diferents dockers i poder aplicar un certificat SSL a tots els mòduls que conté aquesta pàgina hem decidit implementar un proxy invers amb Nginx.

Justificació de l'ús del proxy invers

- **Seguretat:** Afegint una capa addicional de seguretat amb SSL.
- **Abstracció:** Oculta la infraestructura backend.

Arxiu de configuració

```
server {
    server_name timbre.inspedralbes.cat;

    location / {
        proxy_pass http://localhost:3000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    location /pma {
        rewrite ^/pma(/.*)$ $1 break;
        proxy_pass http://localhost:9090/;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    location /portainer/ {
        proxy_pass https://localhost:9443/;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    location /mongoexpress {
        rewrite ^/mongoexpress(/.*)$ $1 break;
        proxy_pass http://localhost:8081/;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }
}
```

```

    proxy_set_header X-Forwarded-Proto $scheme;
}

    location /laravel {
proxy_pass http://localhost:8000/public;
proxy_set_header Host $host;
proxy_set_header X-Real-IP $remote_addr;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_set_header X-Forwarded-Proto $scheme;
    }

    location /node/ {
proxy_pass http://localhost:8080/;
proxy_http_version 1.1;
proxy_set_header Host $host;
proxy_set_header Upgrade $http_upgrade;
proxy_set_header Connection "upgrade";
proxy_cache_bypass $http_upgrade;
    }

    location /socket/ {
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_set_header Host $host;

proxy_pass http://localhost:8080;

proxy_http_version 1.1;
proxy_set_header Upgrade $http_upgrade;
proxy_set_header Connection "upgrade";
    }

    listen 443 ssl; # managed by Certbot
    ssl_certificate
/etc/letsencrypt/live/timbre.inspedralbes.cat/fullchain.pem; #
managed by Certbot
    ssl_certificate_key
/etc/letsencrypt/live/timbre.inspedralbes.cat/privkey.pem; #
managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by
Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by
Certbot

```

```
}  
server {  
    if ($host = timbre.inspedralbes.cat) {  
        return 301 https://$host$request_uri;  
    } # managed by Certbot  
  
    server_name timbre.inspedralbes.cat;  
    return 404; # managed by Certbot  
}
```

Rutes del proxy invers

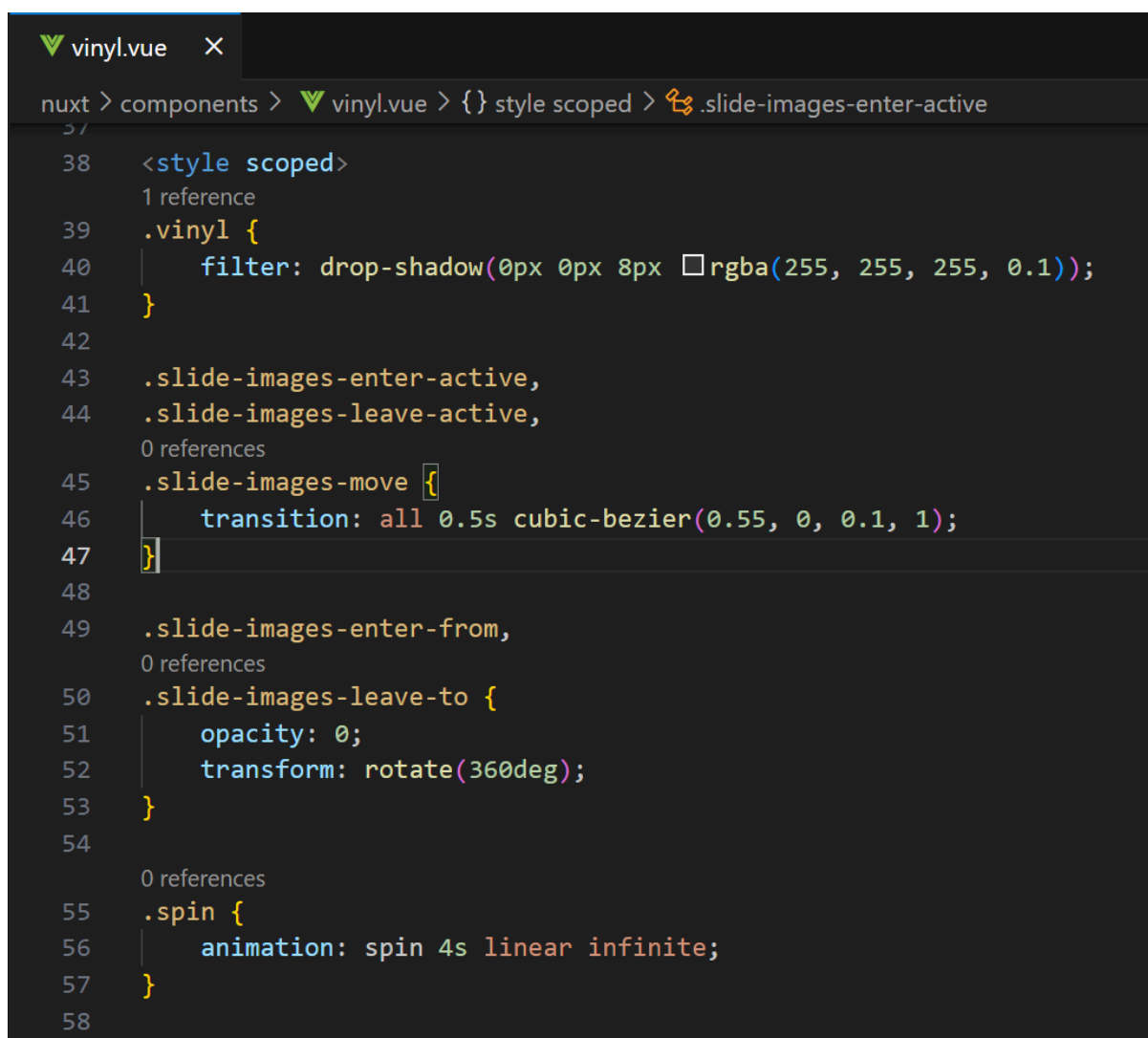
- **/:** Redirigeix al port 3000. Es tracta del frontend de Nuxt montat sobre un servidor de node.
- **/pma:** Redirigeix al port 9091 al phpmyadmin.
- **/portainer/:** Redirigeix al port 9443 al portainer.
- **/laravel:** Redirigeix al port 8000, específicament a `/public`. S'encarrega de servir i dirigir les peticions d'API de Laravel i les rutes de les imatges.
- **/node/` i `/socket/`:** Redirigeixen respectivament als ports 8086 i 8085, utilitzant WebSocket (al port 8086) per a la comunicació en temps real.

Disseny

Pel disseny de l'aplicació utilitzem els estàndards de Nuxt.js: tenim un document amb estils globals al directori `/nuxt/assets/css/main.css` i cada component té els seus estils propis dins del bloc `<style>`.

```
# main.css X
nuxt > assets > css > # main.css > :root
1  :root {
2    --pedralbes-blue: #00ADEF;
3    color-scheme: dark;
4  }
5
6  body {
7    font-family: 'SF Pro', sans-serif;
8    width: 100vw;
9  }
10
11 .material-symbols-rounded {
12   font-variation-settings: 'FILL' 1;
13 }
14
15 .vc-header {
16   margin-bottom: 1em;
17 }
18
19 .vc-weekdays {
20   margin-bottom: 1em;
21 }
22
23 .vc-week {
24   margin-bottom: 1.5em;
25 }
26
```

Estils globals al fitxer `/nuxt/assets/css/main.css`



```
37
38 <style scoped>
39   1 reference
40   .vinyl {
41     filter: drop-shadow(0px 0px 8px rgba(255, 255, 255, 0.1));
42   }
43   .slide-images-enter-active,
44   .slide-images-leave-active,
45   0 references
46   .slide-images-move {
47     transition: all 0.5s cubic-bezier(0.55, 0, 0.1, 1);
48   }
49   .slide-images-enter-from,
50   0 references
51   .slide-images-leave-to {
52     opacity: 0;
53     transform: rotate(360deg);
54   }
55   0 references
56   .spin {
57     animation: spin 4s linear infinite;
58   }
```

Estils del component /nuxt/components/vinyl.css

D'aquesta manera, des del document main.css es poden canviar propietats de totes les pàgines com el color de fons —canviant la propietat css color-scheme (del mode dark pel mode light, per exemple) del selector :root— o la font de la lletra canviant —la propietat css font-family del selector body.

També s'ha integrat Tailwind CSS al projecte com a framework CSS. Per integrar Tailwind CSS s'ha seguit la seva guia d'instal·lació per a Nuxt.js (<https://tailwindcss.com/docs/guides/nuxtjs>). En conseqüència, la majoria dels estils es defineixen aplicant classes preexistents directament al nostre HTML. A mode d'exemple, si es volgués canviar el color de fons del component header s'hauria de canviar la utility class bg-[#1F1F1F].

```

header.vue 2 x
nuxt > components > header.vue > {} template > div.sticky.top-0.left-0.right-0.z-[100].h-fit.bg-[#1F1F1F] > div.flex.flex-row-reverse
1 <template>
2   <div class="sticky top-0 left-0 right-0 z-[100] h-fit bg-[#1F1F1F]"
3     :class="{ 'px-12': $device.isDesktopOrTablet, 'px-4': $device.isMobileOrTablet }">
4     <div class="flex flex-row-reverse md:flex-row justify-between items-center md:h-20">
5
6       <!--MOBILE DESIGN-->
7       <div v-if="$device.isMobileOrTablet" class="flex items-center justify-between gap-3">
8         <!--BURGER BUTTON-->
9         <button class="flex items-center" @click="isSlideoverOpen = !isSlideoverOpen">
10           <span class="material-symbols-outlined text-[2rem]">
11             menu
12           </span>
13         </button>
14         <!--APP NAME-->
15         <NuxtLink :to="/llista_propostes">
16           <div class="brand-name text-3xl md:text-4xl font-bold">sound<span>o'clock</span></div>
17         </NuxtLink>
18         <!--PROFILE IMG -->
19         <div class="flex justify-center rounded-full items-center h-12 w-12">
20           <div class="h-full w-full rounded-full hover:border-2 hover:border-white re">
21             
22             <div class="h-full w-full bg-gray-700 rounded-full absolute z-20">
23               <span
24                 class="material-symbols-outlined text-4xl w-full h-full rounded
25                 person
26               </span>

```

Estils del component /nuxt/components/header.vue. Clases de Tailwind aplicades directament al codi HTML

L'elecció de Tailwind CSS per sobre d'altres frameworks CSS com, per exemple, Bootstrap o Bulma es justifica per l'utilització de la llibreria Nuxt UI (<https://ui.nuxt.com/>). Nuxt UI és una llibreria que simplifica la creació d'aplicacions web responsive oferint una col·lecció de components personalitzables i dissenyats per a Nuxt. Aquest mòdul utilitza Tailwind CSS per construir els seus components. D'aquesta manera s'ha optat per utilitzar Tailwind CSS com a framework CSS del nostre projecte per assegurar una correcta integració amb Nuxt UI.

```

llista_propostes.vue X
nuxt > pages > llista_propostes.vue > {} template > div > div.mb-4 > div.flex.items-center.gap-2 > div.dropdown.relative.z-10 > USlideover.z-[999]
64
65
66 <USlideover v-model="isFiltersSlideOpen" class="z-[9999]">
67   <UCard class="flex flex-col flex-1"
68     :ui="{ body: { base: 'flex-1', background: 'bg-stone-800', }, ring: '', divide: 'divide-
69     <Placeholder class="h-full" />
70     <div class="flex items-center justify-between bg-stone-800 pb-1">
71       <h3 class="text-2xl cursor-default font-semibold leading-6 text-gray-900 dark:text-w
72       FILTERS
73     </h3>
74     <button @click="isFiltersSlideOpen = false"
75       class="h-8 w-8 flex items-center justify-center hover:bg-stone-600 rounded-full
76       <span class="i-heroicons-x-mark-20-solid flex-shrink-0 h-5 w-5"
77       aria-hidden="true"></span>
78     </button>
79   </div>
80   <hr>
81   <div class="pt-3 pb-3">
82     <div class="flex items-center justify-between">
83       <h1 class="text-xl cursor-default pb-2">
84       Hores
85     </h1>
86     <button @click="cleanFilters()" class="hover:font-semibold hover:text-blue-500"
87       :class="{ 'text-blue-500': filterBell == null }">
88       Totes
89     </button>

```

Estils de la pàgina /nuxt/pages/llista_propostes.vue. Modificació dels estils del component proporcionat per Nuxt UI UCard utilitzant Tailwind CSS utility classes

Desplegament

Worflow de treball

Per treballar en local en l'aplicació Spottunes ho fem mitjançant Docker. A continuació s'especifica la configuració del *docker-compose.yml*.

Configuració del docker-compose.yml

```
services:

  db:
    container_name: soundoclock_db
    image: mysql:8.2.0
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: root
      MYSQL_DATABASE: soundoclock
      MYSQL_USER: user
      MYSQL_PASSWORD: user
    ports:
      - 3306:3306
    volumes:
      - ./mysql_data:/var/lib/mysql
      - ./mysql/dades.sql:/docker-entrypoint-initdb.d/dades.sql

  phpmyadmin:
    container_name: soundoclock_phpmyadmin
    image: phpmyadmin/phpmyadmin
    restart: always
    ports:
      - 9090:80
    depends_on:
      - db

  node:
    container_name: soundoclock_node
    image: node:20.11.1-bullseye-slim
    volumes:
      - ./node:/usr/src/app
    working_dir: /usr/src/app
```

```

ports:
  - 8080:8080
command: sh -c "npm install && npm run dev"
depends_on:
  - db

laravel:
  container_name: soundoclock_laravel
  build: ./laravel
  volumes:
    - ./laravel:/var/www/html
  ports:
    - 8000:8000
  environment:
    - APACHE_DOCUMENT_ROOT=/var/www/html/public
  command: /bin/sh -c "composer install && php artisan key:generate
&& chown -R www-data:www-data * && php artisan migrate --force && php
artisan migrate:fresh --seed && php artisan serve --host=0.0.0.0 "
  depends_on:
    - db

nuxt:
  container_name: soundoclock_nuxt
  image: node:20.11.1-bullseye-slim
  working_dir: /usr/src/app
  volumes:
    - ./nuxt:/usr/src/app
  ports:
    - 3000:3000
  environment:
    - WATCHPACK_POLLING=true
    - CHOKIDAR_USEPOLLING=true
  tmpfs:
    - /tmp
  command: sh -c "npm install && npm run dev"

mongodb:
  container_name: soundoclock_mongodb
  image: mongo:latest
  ports:
    - 27017:27017
  restart: always
  environment:

```

```

    MONGO_INITDB_ROOT_USERNAME: mongoadmin
    MONGO_INITDB_ROOT_PASSWORD: mongopassword
  volumes:
    - ./mongodb_data:/data/db

  mongo-express:
    container_name: soundoclock_mongo_express
    image: mongo-express
    restart: always
    ports:
      - 8081:8081
    environment:
      ME_CONFIG_MONGODB_ADMINUSERNAME: mongoadmin
      ME_CONFIG_MONGODB_ADMINPASSWORD: mongopassword
      ME_CONFIG_BASICAUTH_USERNAME: mongoadmin
      ME_CONFIG_BASICAUTH_PASSWORD: mongopassword
      ME_CONFIG_MONGODB_SERVER: mongodb
    depends_on:
      - mongodb

```

Environment de Nuxt.js

```

ENV=DEVELOPMENT

VITE_APP_SOCKET_URI=http://localhost:8080

GOOGLE_CLIENT_ID=Token ID de Google
GOOGLE_CLIENT_SECRET=Token Secret de Google
GOOGLE_REDIRECT_URI=http://localhost:3000/auth/callback/google

```

Environment de Node

```

NODE_ENV=development

DEVELOPMENT_API_URL=http://laravel:8000/api/
PRODUCTION_API_URL=

SPOTIFY_CLIENT_ID=Token ID de Spotify
SPOTIFY_CLIENT_SECRET=Token Secret de Spotify

MONGO_USER=mongoadmin
MONGO_PASSWORD=mongopassword

```

```
PORT=
```

Environment de Laravel

```
DB_CONNECTION=mysql
DB_HOST=db
DB_PORT=3306
DB_DATABASE=soundoclock
DB_USERNAME=root
DB_PASSWORD=root

MAIL_MAILER=smtp
MAIL_HOST=sandbox.smtp.mailtrap.io
MAIL_PORT=2525
MAIL_USERNAME=Username token de Mailtrap
MAIL_PASSWORD=Username password de Mailtrap
MAIL_ENCRYPTION=tls
MAIL_FROM_ADDRESS="timbre@inspedralbes.cat"
MAIL_FROM_NAME="${APP_NAME}"
```

Com aconseguir els tokens

1. Obtenir el token d'accés de Spotify

- 1.1. Visita [Spotify for Developers](#) i inicia sessió o crea un compte.
- 1.2. Crea una nova aplicació al teu panell de control de Spotify Developer.
- 1.3. Obtingues el Client ID i el Client Secret de la teva aplicació.
- 1.4. Configura les redireccions d'URI autoritzades per a la teva aplicació.
- 1.5. Utilitza aquestes credencials per autenticar-te amb l'API de Spotify.

2. Obtenir el token d'accés de Google

- 2.1. Accedeix a [Google Cloud Console](#) i crea un nou projecte.
- 2.2. Habilita les API necessàries per al teu projecte, com ara l'API de Google Maps.
- 2.3. Crea les claus d'API o configura els ID de client i els secrets de client per a l'autenticació d'OAuth, segons les necessitats.
- 2.4. Configura les URL de redirecció autoritzades per a la teva aplicació.
- 2.5. Utilitza les credencials generades per autenticar-te amb les API de Google.

3. Obtenir les credencials de Mailtrap

- 3.1. Visita [Mailtrap](#) i inicia sessió o crea un compte.
- 3.2. Crea un nou inbox des del panell de control.
- 3.3. Accedeix a la secció d'integració del teu inbox per obtenir el username i el password SMTP.

- 3.4. Configura aquests paràmetres SMTP al teu projecte per utilitzar Mailtrap com a servei de correu electrònic de proves.

Desplegament a la web

El desplegament de l'aplicació Spottunes s'ha realitzat utilitzant Oracle Cloud Infrastructure (OCI) i Docker. A continuació les raons per triar OCI:

Justificació de l'ús d'OCI

- **Escalabilitat:** OCI proporciona una infraestructura escalable que permet ajustar els recursos en funció de la demanda de l'aplicació.
- **Rendiment:** OCI ofereix màquines virtuals d'alt rendiment que assegurin una execució fluida i ràpida de l'aplicació.
- **Seguretat:** Les eines i serveis de seguretat d'OCI assegurin la protecció de les dades i la infraestructura.
- **Cost-eficiència:** OCI proporciona opcions de preus competitius que s'ajusten al pressupost del projecte.
- **Integració:** OCI permet la integració amb altres serveis i eines que faciliten el desplegament i la gestió de l'aplicació.

Configuració del Docker Compose

Aquí tens un exemple de com podria ser el fitxer docker-compose.yml. Els espais en blanc s'omplen mitjançant github actions.

```
services:
  phpmyadmin:
    container_name: phpmyadmin
    image: arm64v8/phpmyadmin
    restart: always
    environment:
      PMA_ABSOLUTE_URI: https://timbre.inspedralbes.cat/pma/
    ports:
      - 9090:80
    depends_on:
      - db
  db:
    container_name: soundoclock_db
    image: mysql:8.2.0
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD:
      MYSQL_DATABASE: soundoclock
    ports:
      - 3306:3306
    volumes:
```



```

- ./mysql_data:/var/lib/mysql
- ./mysql/dades.sql:/docker-entrypoint-initdb.d/dades.sql

node:
  container_name: soundoclock_node
  image: node:20.11.1-bullseye-slim
  restart: always
  volumes:
    - ./node:/usr/src/app
  working_dir: /usr/src/app
  ports:
    - 8080:8080
  command: sh -c "npm install && npm run start"
  depends_on:
    - db

laravel:
  container_name: soundoclock_laravel
  build: ./laravel
  ports:
    - 8000:80
  volumes:
    - ./laravel:/var/www/html
  environment:
    - APACHE_DOCUMENT_ROOT=/var/www/html/public
  command: /bin/sh -c "composer install && php artisan key:generate
&& chown -R www-data:www-data * && php artisan migrate --force && php
artisan migrate && apache2-foreground"
  depends_on:
    - db

nuxt:
  container_name: soundoclock_nuxt
  image: node:20.11.1-bullseye-slim
  working_dir: /usr/src/app
  volumes:
    - ./nuxt:/usr/src/app
  ports:
    - 3000:3000
  environment:
    - WATCHPACK_POLLING=true
    - CHOKIDAR_USEPOLLING=true
  tmpfs:

```

```
- /tmp
command: sh -c "node ./server/index.mjs --
--host=http://timbre.inspedralbes.cat:8080"

mongodb:
  container_name: soundclock_mongodb
  image: mongo:latest
  ports:
    - 27017:27017
  restart: always
  environment:
    MONGO_INITDB_ROOT_USERNAME:
    MONGO_INITDB_ROOT_PASSWORD:
  volumes:
    - ./mongodb_data:/data/db
mongo-express:
  container_name: soundclock_mongo_express
  image: mongo-express
  restart: always
  ports:
    - 8081:8081
  environment:
    ME_CONFIG_MONGODB_ADMINUSERNAME:
    ME_CONFIG_MONGODB_ADMINPASSWORD:
    ME_CONFIG_BASICAUTH_USERNAME:
    ME_CONFIG_BASICAUTH_PASSWORD:
    ME_CONFIG_MONGODB_SERVER: mongodb
  depends_on:
    - mongodb
```

Desplegament amb Github Actions

```
name: Sound O'Clock Deploy Actions
run-name: ${{ github.actor }} is deploying Sound O'Clock 🍏
on:
  push:
    branches:
      - main
jobs:
  STOP-DOCKER:
    # needs: [LARAVEL-TEST]
    runs-on: ubuntu-latest
    steps:
      - name: Checkout del código
        uses: actions/checkout@v4

      - name: Conexión al servidor y stop de docker-compose
        run: |
          echo "${{ secrets.SECRET_KEY }}" > ~/prod_key.pem
          chmod 600 ~/prod_key.pem
          ssh -o StrictHostKeyChecking=no -o
UserKnownHostsFile=/dev/null -i ~/prod_key.pem ${{ secrets.PROD_USER
}}@${{ secrets.PROD_HOST }} "docker compose down"
      - run: echo "🍏 This job's status is ${{ job.status }}."

  FILL-DOCKER-COMPOSE:
    needs: [STOP-DOCKER]
    runs-on: ubuntu-latest
    steps:
      - name: Obtaining repository code
        uses: actions/checkout@v4

      - name: Update Docker Compose
        run: |
          echo "Connecting to the server and running docker-compose
commands"
          echo "${{ secrets.SECRET_KEY }}" > ~/prod_key.pem
          chmod 600 ~/prod_key.pem

          ssh -o StrictHostKeyChecking=no -o
UserKnownHostsFile=/dev/null -i ~/prod_key.pem ${{ secrets.PROD_USER
}}@${{ secrets.PROD_HOST }} "rm -f docker-compose.yml"
```

```

        scp -o StrictHostKeyChecking=no -o
UserKnownHostsFile=/dev/null -i "~/prod_key.pem" -r
./docker-compose.yml ${ secrets.PROD_USER }}@${ secrets.PROD_HOST
}}:docker-compose.yml

        sed -i 's#MYSQL_ROOT_PASSWORD:#MYSQL_ROOT_PASSWORD: ${
secrets.DB_PASSWORD }}#g' docker-compose.yml
        sed -i
's#MONGO_INITDB_ROOT_USERNAME:#MONGO_INITDB_ROOT_USERNAME: ${
secrets.MONGO_INITDB_ROOT_USERNAME }}#g' docker-compose.yml
        sed -i
's#MONGO_INITDB_ROOT_PASSWORD:#MONGO_INITDB_ROOT_PASSWORD: ${
secrets.MONGO_INITDB_ROOT_PASSWORD }}#g' docker-compose.yml
        sed -i
's#ME_CONFIG_MONGODB_ADMINUSERNAME:#ME_CONFIG_MONGODB_ADMINUSERNAME:
${ secrets.MONGO_INITDB_ROOT_USERNAME }}#g' docker-compose.yml
        sed -i
's#ME_CONFIG_MONGODB_ADMINPASSWORD:#ME_CONFIG_MONGODB_ADMINPASSWORD:
${ secrets.MONGO_INITDB_ROOT_PASSWORD }}#g' docker-compose.yml
        sed -i
's#ME_CONFIG_BASICAUTH_USERNAME:#ME_CONFIG_BASICAUTH_USERNAME: ${
secrets.MONGO_USER }}#g' docker-compose.yml
        sed -i
's#ME_CONFIG_BASICAUTH_PASSWORD:#ME_CONFIG_BASICAUTH_PASSWORD: ${
secrets.MONGO_PASSWORD }}#g' docker-compose.yml
        scp -o StrictHostKeyChecking=no -o
UserKnownHostsFile=/dev/null -i "~/prod_key.pem" -r
./docker-compose.yml ${ secrets.PROD_USER }}@${ secrets.PROD_HOST
}}:docker-compose.yml
- run: echo "🍏 This job's status is ${ job.status })."

LARAVEL-DEPLOY:
  needs: [FILL-DOCKER-COMPOSE]
  runs-on: ubuntu-latest
  steps:
    - name: Obtaining repository code
      uses: actions/checkout@v4

    - name: Laravel Deploy
      run: |
        echo "${ secrets.SECRET_KEY }}" > ~/prod_key.pem
        chmod 600 ~/prod_key.pem
        cd laravel

```

```

    cp .env.example .env
    sed -i 's/DB_HOST=/DB_HOST=db/g' .env
    sed -i 's/DB_DATABASE=/DB_DATABASE=soundoclock/g' .env
    sed -i 's/DB_USERNAME=/DB_USERNAME=${{ secrets.DB_USERNAME
}}/g' .env
    sed -i 's#DB_PASSWORD=#DB_PASSWORD=${{ secrets.DB_PASSWORD
}}#g' .env
    sed -i 's#MAIL_PASSWORD=#MAIL_PASSWORD=${{
secrets.MAIL_PASSWORD }}#g' .env
    cd ${{ github.workspace }}
    ssh -o StrictHostKeyChecking=no -o
UserKnownHostsFile=/dev/null -i ~/prod_key.pem ${{ secrets.PROD_USER
}}@${{ secrets.PROD_HOST }} "[ -d laravel ] && sudo rm -r laravel"
    ssh -o StrictHostKeyChecking=no -o
UserKnownHostsFile=/dev/null -i ~/prod_key.pem ${{ secrets.PROD_USER
}}@${{ secrets.PROD_HOST }} "mkdir laravel"
    scp -o StrictHostKeyChecking=no -o
UserKnownHostsFile=/dev/null -i "~/prod_key.pem" -r ./laravel/* ${{
secrets.PROD_USER }}@${{ secrets.PROD_HOST }}:laravel
    scp -o StrictHostKeyChecking=no -o
UserKnownHostsFile=/dev/null -i "~/prod_key.pem" -r ./laravel/.env ${{
secrets.PROD_USER }}@${{ secrets.PROD_HOST }}:laravel
    ssh -o StrictHostKeyChecking=no -o
UserKnownHostsFile=/dev/null -i ~/prod_key.pem ${{ secrets.PROD_USER
}}@${{ secrets.PROD_HOST }} "chmod -R 775 laravel/"
    - run: echo "🍏 This job's status is ${{ job.status }}."

NODE-DEPLOY:
  needs: [FILL-DOCKER-COMPOSE]
  runs-on: ubuntu-latest
  steps:
    - name: Obtaining repository code
      uses: actions/checkout@v4

    - name: Node Deploy
      run: |
        echo "${{ secrets.SECRET_KEY }}" > ~/prod_key.pem
        chmod 600 ~/prod_key.pem
        cd node
        cp .env.example .env
        sed -i 's#NODE_ENV=#NODE_ENV=production#g' .env
        sed -i "s#SPOTIFY_CLIENT_ID=#SPOTIFY_CLIENT_ID=$(echo "${{
secrets.SPOTIFY_CLIENT_ID }}" )#g" .env

```

```

    sed -i "s#SPOTIFY_CLIENT_SECRET=#SPOTIFY_CLIENT_SECRET=$(echo
"${ secrets.SPOTIFY_CLIENT_SECRET }")#g" .env
    sed -i "s#PRODUCTION_API_URL=#PRODUCTION_API_URL=$(echo "${ secrets.PRODUCTION_API_URL }")#g" .env
    sed -i "s#MONGO_USER=#MONGO_USER=$(echo "${ secrets.MONGO_INITDB_ROOT_USERNAME }")#g" .env
    sed -i "s#MONGO_PASSWORD=#MONGO_PASSWORD=$(echo "${ secrets.MONGO_INITDB_ROOT_PASSWORD }")#g" .env

    cd ${ github.workspace }
    ssh -o StrictHostKeyChecking=no -o
UserKnownHostsFile=/dev/null -i ~/prod_key.pem ${ secrets.PROD_USER
}}@${ secrets.PROD_HOST } "[ -d node ] && sudo rm -r node"
    ssh -o StrictHostKeyChecking=no -o
UserKnownHostsFile=/dev/null -i ~/prod_key.pem ${ secrets.PROD_USER
}}@${ secrets.PROD_HOST } "mkdir node"
    scp -o StrictHostKeyChecking=no -o
UserKnownHostsFile=/dev/null -i "~/prod_key.pem" -r ./node/* ${ secrets.PROD_USER }}@${ secrets.PROD_HOST }:node
    scp -o StrictHostKeyChecking=no -o
UserKnownHostsFile=/dev/null -i "~/prod_key.pem" -r ./node/.env ${ secrets.PROD_USER }}@${ secrets.PROD_HOST }:node
    ssh -o StrictHostKeyChecking=no -o
UserKnownHostsFile=/dev/null -i ~/prod_key.pem ${ secrets.PROD_USER
}}@${ secrets.PROD_HOST } "chmod -R 775 node/"
    - run: echo "🍏 This job's status is ${ job.status }."

Nuxt-DEPLOY:
  needs: [FILL-DOCKER-COMPOSE]
  runs-on: ubuntu-latest
  steps:
    - name: Obtaining repository code
      uses: actions/checkout@v4

    - name: Front Deploy
      run: |
        echo "${ secrets.SECRET_KEY }" > ~/prod_key.pem
        chmod 600 ~/prod_key.pem
        cd ${ github.workspace }
        cd nuxt
        cp .env.example .env
        npm install
        npm install -D sass

```

```

    sed -i 's/GOOGLE_CLIENT_ID=/GOOGLE_CLIENT_ID=${{
secrets.GOOGLE_CLIENT_ID }}/g' .env
    sed -i 's/GOOGLE_CLIENT_SECRET=/GOOGLE_CLIENT_SECRET=${{
secrets.GOOGLE_CLIENT_SECRET }}/g' .env
    sed -i 's#GOOGLE_REDIRECT_URI=#GOOGLE_REDIRECT_URI=${{
secrets.GOOGLE_REDIRECT_URI }}#g' .env
    sed -i 's#VITE_APP_SOCKET_URI=#VITE_APP_SOCKET_URI=${{
secrets.PRODUCTION_SOCKET_URI }}#g' .env
    npm run build
    ssh -o StrictHostKeyChecking=no -o
UserKnownHostsFile=/dev/null -i ~/prod_key.pem ${{ secrets.PROD_USER
}}@${{ secrets.PROD_HOST }} "[ -d nuxt ] && sudo rm -r nuxt"
    scp -o StrictHostKeyChecking=no -o
UserKnownHostsFile=/dev/null -i "~/prod_key.pem" -r .output ${{
secrets.PROD_USER }}@${{ secrets.PROD_HOST }}:nuxt
    - run: echo "🍏 This job's status is ${{ job.status }}."

START-DOCKER:
  needs: [LARAVEL-DEPLOY, NODE-DEPLOY, NUXT-DEPLOY]
  runs-on: ubuntu-latest
  steps:
    - name: Start Docker
      run: |
        echo "${{ secrets.SECRET_KEY }}" > ~/prod_key.pem
        chmod 600 ~/prod_key.pem
        ssh -o StrictHostKeyChecking=no -o
UserKnownHostsFile=/dev/null -i ~/prod_key.pem ${{ secrets.PROD_USER
}}@${{ secrets.PROD_HOST }} "docker compose up -d"
    - name: Iniciar Portainer
      run: |
        ssh -o StrictHostKeyChecking=no -o
UserKnownHostsFile=/dev/null -i ~/prod_key.pem ${{ secrets.PROD_USER
}}@${{ secrets.PROD_HOST }} "docker start portainer"
    - run: echo "🍏 This job's status is ${{ job.status }}."

```

STOP-DOCKER

- **Checkout del codi:** Fa el checkout del codi del repositori.
- Connexió al servidor i aturada de docker-compose: Es connecta al servidor de producció i atura tots els contenidors Docker utilitzant docker compose down.

FILL-DOCKER-COMPOSE

- **Obtenció del codi del repositori:** Fa el checkout del codi del repositori.
- **Actualització de Docker Compose:** Es connecta al servidor de producció, puja el fitxer docker-compose.yml, i l'actualitza amb els secrets necessaris.

LARAVEL-DEPLOY

- **Obtenció del codi del repositori:** Fa el checkout del codi del repositori.
- **Desplegament de Laravel:** Prepara el fitxer `.env` per a Laravel, puja l'aplicació Laravel al servidor de producció, i ajusta els permisos adequats.

NODE-DEPLOY

- **Obtenció del codi del repositori:** Fa el checkout del codi del repositori.
- **Desplegament de Node:** Prepara el fitxer `.env` per a l'aplicació Node.js, puja l'aplicació Node.js al servidor de producció, i ajusta els permisos adequats.

NUXT-DEPLOY

- **Obtenció del codi del repositori:** Fa el checkout del codi del repositori.
- **Desplegament del Frontend:** Prepara el fitxer `.env` per a l'aplicació Nuxt.js, instal·la les dependències, construeix l'aplicació, i puja els fitxers construïts al servidor de producció.

START-DOCKER

- **Iniciar Docker:** Es connecta al servidor de producció i inicia tots els contenidors Docker utilitzant `docker compose up -d`.
- **Iniciar Portainer:** Inicia el contenidor Portainer per gestionar Docker.

Github Secrets

Els secrets utilitzats en el workflow de desplegament són variables d'entorn que contenen informació sensible necessària per a la configuració i el desplegament de l'aplicació. A continuació es detallen els secrets utilitzats:

1. **SECRET_KEY:** Clau privada utilitzada per establir una connexió segura SSH al servidor de producció.
2. **PROD_USER:** Nom d'usuari del servidor de producció.
3. **PROD_HOST:** Adreça del servidor de producció.
4. **DB_PASSWORD:** Contrasenya de l'usuari root de la base de dades MySQL.
5. **MONGO_INITDB_ROOT_USERNAME:** Nom d'usuari administrador per a MongoDB.
6. **MONGO_INITDB_ROOT_PASSWORD:** Contrasenya de l'usuari administrador per a MongoDB.
7. **MAIL_PASSWORD:** Contrasenya per al servei de correu.
8. **SPOTIFY_CLIENT_ID:** Identificador del client de Spotify.
9. **SPOTIFY_CLIENT_SECRET:** Contrasenya del client de Spotify.
10. **PRODUCTION_API_URL:** URL de l'API en producció.
11. **MONGO_USER:** Nom d'usuari per accedir a MongoDB.

12. **MONGO_PASSWORD:** Contrasenya per accedir a MongoDB.
13. **GOOGLE_CLIENT_ID:** Identificador del client de Google.
14. **GOOGLE_CLIENT_SECRET:** Contrasenya del client de Google.
15. **GOOGLE_REDIRECT_URI:** URI de redirecció de Google.
16. **PRODUCTION_SOCKET_URI:** URI del socket en producció.

Com aconseguir els tokens

4. Obtenir el token d'accés de Spotify

- 4.1. Visita [Spotify for Developers](#) i inicia sessió o crea un compte.
- 4.2. Crea una nova aplicació al teu panell de control de Spotify Developer.
- 4.3. Obtingues el Client ID i el Client Secret de la teva aplicació.
- 4.4. Configura les redireccions d'URI autoritzades per a la teva aplicació.
- 4.5. Utilitza aquestes credencials per autenticar-te amb l'API de Spotify.

5. Obtenir el token d'accés de Google

- 5.1. Accedeix a [Google Cloud Console](#) i crea un nou projecte.
- 5.2. Habilita les API necessàries per al teu projecte, com ara l'API de Google Maps.
- 5.3. Crea les claus d'API o configura els ID de client i els secrets de client per a l'autenticació d'OAuth, segons les necessitats.
- 5.4. Configura les URL de redirecció autoritzades per a la teva aplicació.
- 5.5. Utilitza les credencials generades per autenticar-te amb les API de Google.

6. Obtenir les credencials de Mailtrap

- 6.1. Visita [Mailtrap](#) i inicia sessió o crea un compte.
- 6.2. Crea un nou inbox des del panell de control.
- 6.3. Accedeix a la secció d'integració del teu inbox per obtenir el username i el password SMTP.