

*AREA Project*  
*Year 2024-2025*



# **Nexus Documentation**

## *Web Service*

*The goal of this project is to create an automation platform, using actions and reactions with various APIs.*

*For this project, we needed to create a website and a mobile app.*

*In this documentation, we will tell you about the Web Service that we created to handle actions and reactions.*

# Behaviors

## Actions

This folder contains all the behaviors of the actions for our APIs.

## Reactions

This folder contains all the behaviors of the reactions for our APIs.

## Managers

Select the appropriate action / reaction depending on the name of the entry.

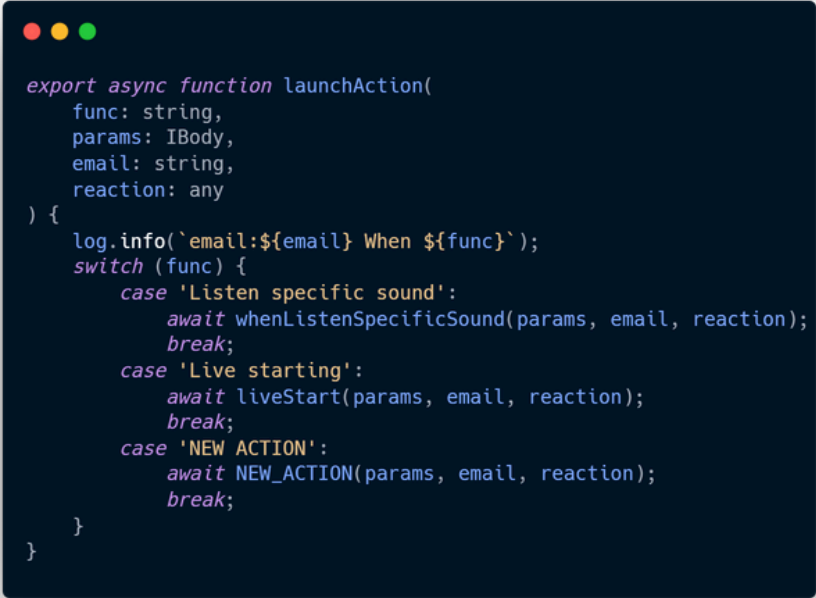
## Web Service

Retrieve the user configurations and launch the actions / reactions.

# Add new actions

“/ws/src/core/action.manager.ts”

To add new actions, modify the **launchAction** function



```
export async function launchAction(  
  func: string,  
  params: IBody,  
  email: string,  
  reaction: any  
) {  
  log.info(`email:${email} When ${func}`);  
  switch (func) {  
    case 'Listen specific sound':  
      await whenListenSpecificSound(params, email, reaction);  
      break;  
    case 'Live starting':  
      await liveStart(params, email, reaction);  
      break;  
    case 'NEW ACTION':  
      await NEW_ACTION(params, email, reaction);  
      break;  
  }  
}
```

The functions are inside “/ws/src/core/actions”, in the different API files.

# Add new reactions

“/ws/src/core/reaction.manager.ts”

To add new reactions, modify the **launchReaction** function

A code editor window with a dark blue background and light blue text. The code is a TypeScript function named `launchReaction` which is exported as an async function. It takes four parameters: `func` (string), `params` (IBody), `actionParam` (IBodySpecific[]), and `email` (string). The function body starts with a conditional check `if (actionParam.length > 0 && params.reaction.length > 0)`. Inside this, there are nested loops: a `for` loop over `params.reaction` and another `for` loop over `actionParam`. The inner loop calls `replaceLabel` with `reaction.value`, `action.name`, and `action.value`, then logs the result with `log.debug`. After the loops, it logs an info message: `log.info('email:${email} Reaction: ${func}')`. A `switch` statement follows, with cases for `'Skip to next'` (calling `skipToNextMusic`), `'Skip to previous'` (calling `skipToPreviousMusic`), and `'NEW REACTION'` (calling `NEW_REACTION`). Each case ends with `break;`. The function ends with a closing brace `}`.

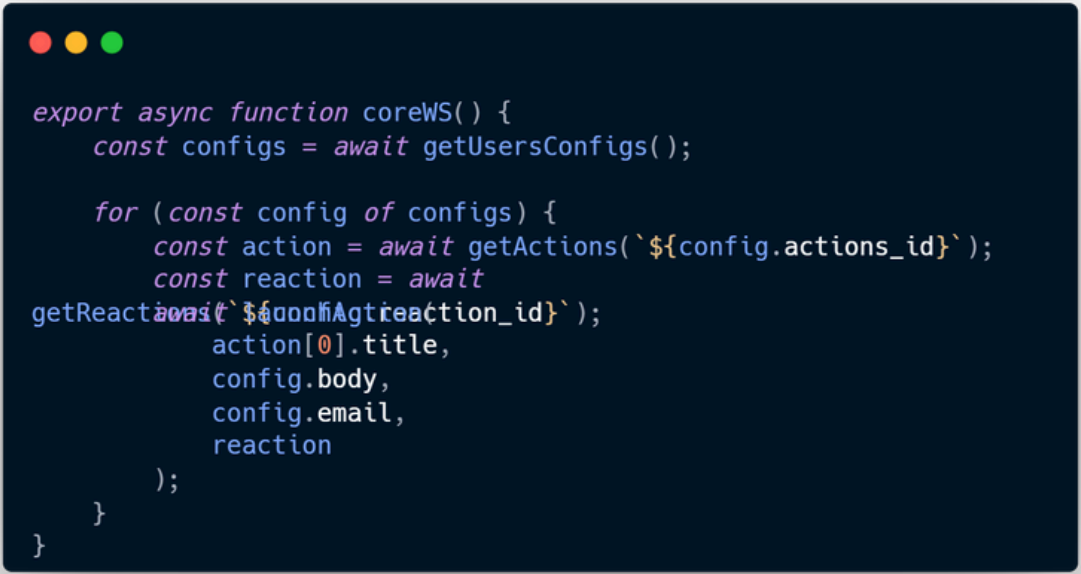
```
export async function launchReaction(  
  func: string,  
  params: IBody,  
  actionParam: IBodySpecific[],  
  email: string  
) {  
  if (actionParam.length > 0 && params.reaction.length > 0) {  
    for (const reaction of params.reaction) {  
      for (const action of actionParam) {  
        reaction.value = replaceLabel(  
          reaction.value,  
          action.name,  
          action.value  
        );  
        log.debug(reaction.value);  
      }  
    }  
  }  
  log.info(`email:${email} Reaction: ${func}`);  
  switch (func) {  
    case 'Skip to next':  
      await skipToNextMusic(email);  
      break;  
    case 'Skip to previous':  
      await skipToPreviousMusic(email);  
      break;  
    case 'NEW REACTION':  
      await NEW_REACTION(email);  
      break;  
  }  
}
```

The functions are inside “/ws/src/core/reactions”, in the different API files.

# Web service

“/ws/src/core/webservice.ts”

Here's how the web service works, to launch the actions and reactions

A code editor window with a dark blue background and three colored window control buttons (red, yellow, green) in the top-left corner. The code is written in TypeScript and defines an asynchronous function named `coreWS()`. It starts by awaiting `getUsersConfigs()` to get a list of configurations. Then, it iterates over each configuration. For each configuration, it awaits `getActions()` to get a list of actions and `getReactions()` to get a list of reactions. It then constructs an array containing the first action's title, the configuration's body, email, and the first reaction's title. The code is as follows:

```
export async function coreWS() {
  const configs = await getUsersConfigs();

  for (const config of configs) {
    const action = await getActions(`${config.actions_id}`);
    const reaction = await
getReactions(`${config.reactions_id}`);
    action[0].title,
    config.body,
    config.email,
    reaction
  );
  }
}
```