**http status code**

1xx Informational response

100 Continue -everything so far is OK, the client can continue the

request or ignore if it's finished.

101 Switching Protocol - indicates the protocol the server is switching to

102 Processing (WebDAV) - indicates the server received and is

processing the request, but no response available yet.

103 Early Hints

2xx Success

200 OK - The request has succeeded

201 Created - The request has succeeded and new resource was created

202 Accepted - has been received but not yet acted upon

203 Non-Authoritative - meta-information set is not exact set as available

from the origin server, but collected from a local or a third party copy.

204 No Content - Tno content to send for this request

205 Reset

206 Partial

207 Multi-Status (WebDAV) - conveys information

about multiple resources

208 Multi-Status (WebDAV)

226 IM Used (HTTP Delta encoding)

3xx Redirection

300 Multiple Choice

301 Moved Permanently - URI of the requested

resource has been changed permanently

302 Found - has been changed temporarily

303 See Other - direct the client to get the requested

resource at another URI with a GET request.

304 Not Modified - Thas not been modified, so

the client can use cached version

305 Use Proxy - deprecated

306 unused - deprecated

307 Temporary - direct the client to get resource at

another URI with same method that was used in the prior request

308 Permanent - resource is now permanently located at

another URI, specified by the Location: HTTP Response header

4xx Client errors

400 Bad Request - server could not understand the request due to invalid syntax

401 Unauthorized - Asemantically this response means "unauthenticated"

402 Payment Required - This response code is reserved for future use

403 Forbidden - does not have access rights to the content,

unlike 401, the client's identity is known to the server.

404 Not Found - The server can not find requested resource.

405 Method Not Allowed - The request method is known

by the server but has been disabled and cannot be used

406 Not Acceptable

407 Proxy Authentication Required - This is similar to

401 but authentication is needed to be done by a proxy.

408 Request Timeout - sent on an idle connection by some servers

409 Conflict - conflicts with the current state of the server

410 Gone - content has been permanently deleted from server

411 Length Required - Content-Length header field is required

412 Precondition Failed

413 Payload Too Large - payload larger

than limits defined by server

414 URI Too Long - URI longer than server allows

415 Unsupported Media Type - The media format

is not supported by server

416 Requested Range Not Satisfiable

417 Expectation Failed

418 I'm a teapot - joke

421 Misdirected Request

422 Unprocessable Entity (WebDAV)

423 Locked (WebDAV)

424 Failed Dependency (WebDAV)

425 Too Early

426 Upgrade Required

428 Precondition Required

429 Too Many Requests - sent too

many requests ("rate limiting").

431 Request Header Fields Too Large

451 Unavailable For Legal Reasons

5xx Server errors

500 Internal Server Error - error

erver doesn't know how to handle.

501 Not Implemented - The

request method is not supported

502 Bad Gateway

503 Service Unavailable - The server

is not ready to handle the request

504 Gateway Timeout - server is acting as

gateway and cannot get a response in time.

505 HTTP Version Not Supported

506 Variant Also Negotiates

507 Insufficient Storage

508 Loop Detected (WebDAV)

510 Not Extended

511 Network Authentication Required

**http methods**

GET - deve retornar apenas dados.

HEAD - GET sem conter o corpo da resposta.

POST - submeter uma entidade a um recurso específico, causando

mudança de estado do recurso ou efeitos colaterais no servidor.

PUT - substitui todas as atuais representações do recurso d

destino pela carga de dados da requisição.

DELETE - remove um recurso

CONNECT - faz túnel para servidor identificado pelo recurso.

OPTIONS - descrever as opções de comunicação com o recurso.

TRACE - executa um teste loop-back com o caminho para o recurso.

PATCH - aplica modificações parciais a recurso.

**webpack**

The webpack library allows to bundle javascript files into one big file, bundle other static resources like images, handle bar files, css files, etc.

field "module" several loaders are defined, to be able to load static resources like images, css files and handlebars files.

"HtmlWebpackPlugin" plugin was used to simplify the creation of HTML files to serve the webpack bundles.

npm run-script -> arbitrary command from a package’s "scripts", same as npm-build

npm-build: executes build instruction on package.json

array.slice() - copy

arrat.slice(init, end) - sub arrany

end index not included in array

array.find((element, index, array)=>{})

- retorna o objeto se exstir ou undefined

array.findIndex((element, index, array)=>{})

- retorna index da 1ª ocorrencia ou -1

array.indexOf(arg) - igual a findIndex

array.includes(arg, fromIndex) - devolve true se encontrar

array.map((element, index, array)=>{})

array.filter((element, index, array)=>{return boolean for no filtering})

array.reduce((accumulator, element, index, array)

=>{accumulator+element}, valor inicial)

array.sort((a,b)=>{

    0 - iguais

    1 - a > b

    -1 - b < a

})

new Promise(function(resolve, reject) {

    try {

        setTimeout(resolve, 100, 'foo');

    } catch(err) {

        reject(err);

    }

});

Promise.all([promise1, promise2, promise3]).then(function(values) {

  console.log(values);//array of values

});

element = document.querySelector(selectors);

elementList = document.querySelectorAll(selectors);

element = document.getElementById(id);

element.addEventListener(type,listener);

element.className

eventos: change, click, copy, focus, hashchange,

input, keypress, mouseenter, mouseleave

event bubbling - eventos começam do elementos mais

abaixo no DOM e propagam-se para os elementos de cima

<form id="searchGamesForm">

    <input type="text" name="gameName" placeholder="name" value="" />

    <input type="submit" />

</form>

input types: checkbox, email, hidden, image, password, number

**yama-web-api**

async function addMusicToList(req, rsp){

    try {

        /\* validadores de parâmetros \*/

        let playlist = yamaService.addMusicToList(req.params.playlistId, req.params.musicId);

        res.statusCode = 201;

        res.setHeader('Content-type', 'application/json');

        res.end(JSON.stringify(playlist));

    } catch(err) {

        res.statusCode = 500;

        res.setHeader('Content-type', 'application/json');

        res.end(JSON.stringify(err));

    }

}

**yama-db**

async function addMusicToList(playListId, musicId){

    let playlist = await yamaDb.getPlaylistById(playlistId);

    if(!playlist) {

        let err =  new Error("No playlist with that id");

        err.status = 404;

        throw err;

    }

    if(playlist.allowRepeats || !playlist.musics.find(music.id === musicId)) {

        yamaDb.addGameToList(playlistId, musicId);

    } else {

        let err =  new Error("Cannot add repeated music for this playlist");

        err.status = 500;

        throw err;

    }

}

**Yama service**

async function getArtistsNames(nameStart, resultsCount) {

    let options = {

        method: "GET",

        headers: new Headers("Accept", "application/json")

    }

    let response = await fetch(`/artists/${nameStart}?max=${resultsCount}`, options);

    if(response.ok){

        let body = await response.json();

        return body;

    } else { return response.status; }

}

**yama html**

document.querySelector("#artists").addEventListener("change", listener);

async function listener(e) {

    let input = document.querySelector("#artists");

    let div = document.querySelector("#suggestions");

    if(input.value.length < 3) {

        if(input.className != "hiddenAutoComplete") {

            input.className = "hiddenAutoComplete";

        }

        div.innerHtml = "";

    } else {

        if(input.className != "visibleAutoComplete") {

            input.className = "visibleAutoComplete";

        }

        let artists = await getArtistsNames(input.value, 10);

        let htmlList = "";;

        artists.forEach(e => {

            artistList += `<div>${e.name}</div>`;

        });

        div.innerHtml = artistList;

    }

}

**API fetch()**

async function getAllBundlesBookTitles () {

    let response = await fetch ( 'http://localhost:9200/b4/bundle/\_search' )

    let bundles = await response . json ();

    const bookUrlBase = 'http://localhost:9200/books/book/'

    const requests = bundles

        . flatMap ( bundle => bundle . books )

        . map ( b => fetch ( bookUrlBase + b . id ))

    const responses = await Promise . all ( requests );

    const books = await Promise . all ( responses . map ( async resp => await resp . json ()))

    const booksTitles = books . map ( b => b . \_source . title )

    return booksTitles ;

}

async function getAllBundlesBookTitles\_withoutAwaitKeyWord () {

    const url = 'http://localhost:9200/b4/bundle/\_search'

    const bookUrlBase = 'http://localhost:9200/books/book/'

    return fetch (url)

        .then(response => response . json())

        .then(bundles => bundles

            . flatMap ( bundle => bundle . books )

            . map ( b => fetch ( bookUrlBase + b . id ))

        )

        .then(response => response . map (resp => resp . json()))

        .then(books => books . map ( b => b . \_source . title ))

}

module . exports = function( redirectUrl ) {

    return function( req , rsp , next ) {

    if ( req . isAuthenticated ()) {

        return next ();

    }

    rsp . set ({ 'Location' : redirectUrl }). end ()

    }

}

**copia função e adiciona callback**

function nodify(fn) {

    let newFn = function() {

        // last argument its a cb function

        let cb = arguments[arguments.length-1];

        // Use .apply() to have the same access to  fn- arguments in functon -newFn-

        fn.apply(null, arguments)

            .then(rsp => cb(null, rsp)).catch(err => cb(err))

    }

    return newFn;

}

**calcular tempos de execução**

function profile(fn) {

    let newFn = function () {

        newFn.execs = [];

        let start = new Date().getTime();

        let result = fn.apply(null, arguments).then(res => {

            let end = new Date().getTime();

            newFn.execs.push(end - start);

            return res;

        });

        return result;

    }

    newFn.avgDur = function () {

        return (newFn.execs.reduce((accum, curr) => accum + curr, 0) / newFn.execs.length);

    };

    return newFn;

}

**handlebars**

<table>

    <thead>

        <tr>

            {{#each headings as |column|}}

                <th>{{column}}</th>

            {{/each}}

        </tr>

    </thead>

    <tbody>

        {{#each items as |row|}}

            <tr>

                {{#each row as |cell|}}

                    <td>{{cell}}</td>

                {{/each}}

            </tr>

        {{/each}}

    </tbody>

</table>