The code

I'll try to build a function page for a call to "https://rocket.chat/docs/developer-guides/rest-api/users/create/"

```
library(rvest)
```

```
Loading required package: xml2
```

```
library(magrittr, warn.conflicts = FALSE)
library(purrr, warn.conflicts = FALSE)
page <- read_html("https://rocket.chat/docs/developer-guides/rest-api/users/create/")
```

When on the page, we will find the args and their doc on the second table

```
fun_args <- page %>%
  html_table() %>%
  purrr::pluck(2)
fun_args
```

```
                Argument                  Example                      Required
1                  email      example@example.com                      Required
2                   name            Example User                      Required
3               password               pass@w0rd                      Required
4               username                 example                      Required
5                 active                   false      Optional  Default: true
6                  roles                 ['bot']  Optional  Default: ['user']
7    joinDefaultChannels                   false      Optional  Default: true
8  requirePasswordChange                    true     Optional  Default: false
9       sendWelcomeEmail                    true     Optional  Default: false
10              verified                    true     Optional  Default: false
11          customFields { twitter: '@example' } Optional  Default: undefined
                                                                    Description
1                                          The email address for the user.
2                                            The display name of the user.
3                                               The password for the user.
4                                               The username for the user.
5  Whether the user is active, which determines if they can login or not.
6                    The roles the user has assigned to them on creation.
7         Whether the user should join the default channels when created.
8   Should the user be required to change their password when they login?
9                                     Should the user get a welcome email?
10              Should the user's email address be verified when created?
11               Any custom fields the user should have on their account.
```

Nice! We can start by creating some roxy tags out of this.

```
# Small util functions
tag_app <- function(var) {
  sprintf("#' %s", var)
}

ap_em_rox <- function(val){
  val %<>% c(
    "#' "
  )
  val
```

```
}
```

The title of the function is at the second h2 of the page:

```
page %>%
  html_nodes("h1") %>%
  pluck(2) %>%
  html_text()
```

```
[1] "Create"
```

Let's add it.

```
title <- page %>%
  html_nodes("h1") %>%
  pluck(2) %>%
  html_text()
fun_code <- title %>%
  tag_app() %>%
  ap_em_rox()
cat(fun_code, sep = "\n")
```

```
#' Create
#'
```

And use the first paragraph as a function description.

```
fun_code %<>% c(
  page %>%
    html_nodes("p") %>%
    pluck(1)%>%
    html_text() %>%
    gsub("\n", ". ", .) %>%
    tag_app() %>%
    ap_em_rox()
)

cat(fun_code, sep = "\n")
```

```
#' Create
#'
#' Create a new user. Requires create-user permission.
#'
```

Now, something our API doc doesn't have, the connection token we'll be passing from the auth.

```
fun_code %<>% c("#' @param token The token to connect to the app.")
styler::style_text(fun_code)
```

```
#' Create
#'
#' Create a new user. Requires create-user permission.
#'
#' @param token The token to connect to the app.
```

Ok remember our table of parameters? Let's us it now to create the roxygen tags.

```
fun_code %<>% c(
  purrr::pmap_chr(
```

```
    fun_args,
    ~ {
      sprintf(
        "#' @param %s %s %s",
        thinkr::clean_vec(..1), ..4, ..3
      )
    }
  )
) %<>% ap_em_rox()
cat(fun_code, sep = "\n")

#' Create
#'
#' Create a new user. Requires create-user permission.
#'
#' @param token The token to connect to the app.
#' @param email The email address for the user. Required
#' @param name The display name of the user. Required
#' @param password The password for the user. Required
#' @param username The username for the user. Required
#' @param active Whether the user is active, which determines if they can login
or not. Optional  Default: true
#' @param roles The roles the user has assigned to them on creation. Optional
Default: ['user']
#' @param joindefaultchannels Whether the user should join the default channels
when created. Optional  Default: true
#' @param requirepasswordchange Should the user be required to change their
password when they login? Optional  Default: false
#' @param sendwelcomeemail Should the user get a welcome email? Optional
Default: false
#' @param verified Should the user's email address be verified when created?
Optional  Default: false
#' @param customfields Any custom fields the user should have on their account.
Optional  Default: undefined
#'
```

Of course we'll export the function:

```
fun_code %<>% c(
  tag_app("@export")
)
```

Some deps

```
fun_code %<>% c(
  tag_app("@importFrom httr POST GET add_headers content stop_for_status"),
  tag_app("@importFrom jsonlite toJSON")

)
```

Now, let's create the function and its arguments.

```
fun_code %<>% c(
  sprintf(
    "%s <- function(tok,",
    thinkr::clean_vec(title)
  )
)
```

```
fun_code %<>% c(
  purrr::pmap_chr(
    fun_args,
    ~ {
      if (
        grepl("Optional", ..3)
      ){
        sprintf(
          "  %s = NULL,",
          thinkr::clean_vec(..1)
        )
      } else {
        sprintf(
          "  %s,",
          thinkr::clean_vec(..1)
        )
      }

    }
  )
)

fun_code[
  length(fun_code)
  ] %<>% gsub(",", "", .)
cat(fun_code, sep = "\n")

#' Create
#'
#' Create a new user. Requires create-user permission.
#'
#' @param token The token to connect to the app.
#' @param email The email address for the user. Required
#' @param name The display name of the user. Required
#' @param password The password for the user. Required
#' @param username The username for the user. Required
#' @param active Whether the user is active, which determines if they can login
or not. Optional  Default: true
#' @param roles The roles the user has assigned to them on creation. Optional
Default: ['user']
#' @param joindefaultchannels Whether the user should join the default channels
when created. Optional  Default: true
#' @param requirepasswordchange Should the user be required to change their
password when they login? Optional  Default: false
#' @param sendwelcomeemail Should the user get a welcome email? Optional
Default: false
#' @param verified Should the user's email address be verified when created?
Optional  Default: false
#' @param customfields Any custom fields the user should have on their account.
Optional  Default: undefined
#'
#' @export
#' @importFrom httr POST GET add_headers content stop_for_status
#' @importFrom jsonlite toJSON
create <- function(tok,
  email,
  name,
  password,
```

```
  username,
  active = NULL,
  roles = NULL,
  joindefaultchannels = NULL,
  requirepasswordchange = NULL,
  sendwelcomeemail = NULL,
  verified = NULL,
  customfields = NULL
```

Ok now a little bit of the internals:

```
fun_code %<>% c(
  "){",
  " ",
  "  params <- list("
)

fun_code %<>% c(
  purrr::pmap_chr(
    fun_args,
    ~ {
      sprintf(
        "    %s = %s,",
        thinkr::clean_vec(..1), thinkr::clean_vec(..1)
      )
    }
  )
)

fun_code[
  length(fun_code)
  ] %<>% gsub(",", "", .)

fun_code %<>% c(
  ")",
  "",
  "params <- no_null(params)",
  "",
  "params <- toJSON(params, auto_unbox = TRUE)"
)
cat(fun_code, sep = "\n")

#' Create
#'
#' Create a new user. Requires create-user permission.
#'
#' @param token The token to connect to the app.
#' @param email The email address for the user. Required
#' @param name The display name of the user. Required
#' @param password The password for the user. Required
#' @param username The username for the user. Required
#' @param active Whether the user is active, which determines if they can login
or not. Optional  Default: true
#' @param roles The roles the user has assigned to them on creation. Optional
Default: ['user']
#' @param joindefaultchannels Whether the user should join the default channels
when created. Optional  Default: true
#' @param requirepasswordchange Should the user be required to change their
```

```
password when they login? Optional  Default: false
#' @param sendwelcomeemail Should the user get a welcome email? Optional
Default: false
#' @param verified Should the user's email address be verified when created?
Optional  Default: false
#' @param customfields Any custom fields the user should have on their account.
Optional  Default: undefined
#'
#' @export
#' @importFrom httr POST GET add_headers content stop_for_status
#' @importFrom jsonlite toJSON
create <- function(tok,
  email,
  name,
  password,
  username,
  active = NULL,
  roles = NULL,
  joindefaultchannels = NULL,
  requirepasswordchange = NULL,
  sendwelcomeemail = NULL,
  verified = NULL,
  customfields = NULL
){

  params <- list(
    email = email,
    name = name,
    password = password,
    username = username,
    active = active,
    roles = roles,
    joindefaultchannels = joindefaultchannels,
    requirepasswordchange = requirepasswordchange,
    sendwelcomeemail = sendwelcomeemail,
    verified = verified,
    customfields = customfields
)

params <- no_null(params)

params <- toJSON(params, auto_unbox = TRUE)
```

A piece of the `{httr}` request. Note that our token contains a `$url` that contains the url of the chat, and the `authToken` and `userId`.

The endpoint is on the first table, as is the http method.

```
method <- page %>%
  html_table() %>%
  pluck(1)
method["HTTP Method"]

  HTTP Method
1        POST

method["URL"]

                   URL
```

```
1 /api/v1/users.create
```

```
if ( method["HTTP Method" ] == "POST"){
  fun_code %<>% c(
    sprintf(
      "res <- httr::%s(",
      method["HTTP Method"]
    ),
    "add_headers('Content-type' = 'application/json',",
    "'X-Auth-Token' = tok$data$authToken,",
    "'X-User-Id' = tok$data$userId",
    "),",
    sprintf("url = paste0(tok$url, '%s'),", method["URL"] ),
    "body = params"
  )
} else {
  fun_code %<>% c(
    sprintf(
      "res <- httr::%s(",
      method["HTTP Method" ]
    ),
    "add_headers('Content-type' = 'application/json',",
    "'X-Auth-Token' = tok$data$authToken,",
    "'X-User-Id' = tok$data$userId",
    "),",
    sprintf("url = paste0(tok$url, '%s')", method["URL"] )
  )
}
```

End this:

```
fun_code %<>% c(
  ")",
  " ",
  "stop_for_status(res)",
  "content(res)",
  "}"
)
```

Let's see what this looks like:

```
styler::style_text(fun_code)
```

```
#' Create
#'
#' Create a new user. Requires create-user permission.
#'
#' @param token The token to connect to the app.
#' @param email The email address for the user. Required
#' @param name The display name of the user. Required
#' @param password The password for the user. Required
#' @param username The username for the user. Required
#' @param active Whether the user is active, which determines if they can login
or not. Optional  Default: true
#' @param roles The roles the user has assigned to them on creation. Optional
Default: ['user']
#' @param joindefaultchannels Whether the user should join the default channels
when created. Optional  Default: true
#' @param requirepasswordchange Should the user be required to change their
```

```
password when they login? Optional  Default: false
#' @param sendwelcomeemail Should the user get a welcome email? Optional
Default: false
#' @param verified Should the user's email address be verified when created?
Optional  Default: false
#' @param customfields Any custom fields the user should have on their account.
Optional  Default: undefined
#'
#' @export
#' @importFrom httr POST GET add_headers content stop_for_status
#' @importFrom jsonlite toJSON
create <- function(tok,
                   email,
                   name,
                   password,
                   username,
                   active = NULL,
                   roles = NULL,
                   joindefaultchannels = NULL,
                   requirepasswordchange = NULL,
                   sendwelcomeemail = NULL,
                   verified = NULL,
                   customfields = NULL) {
  params <- list(
    email = email,
    name = name,
    password = password,
    username = username,
    active = active,
    roles = roles,
    joindefaultchannels = joindefaultchannels,
    requirepasswordchange = requirepasswordchange,
    sendwelcomeemail = sendwelcomeemail,
    verified = verified,
    customfields = customfields
  )

  params <- no_null(params)

  params <- toJSON(params, auto_unbox = TRUE)
  res <- httr::POST(
    add_headers(
      "Content-type" = "application/json",
      "X-Auth-Token" = tok$data$authToken,
      "X-User-Id" = tok$data$userId
    ),
    url = paste0(tok$url, "/api/v1/users.create"),
    body = params
  )

  stop_for_status(res)
  content(res)
}
```

And to automate even more:

```
usethis::use_r(title)
write(fun_code, sprintf("R/%s.R", title))
```

Pretty cool! Seems like I have everything… or almost. Let's automate it
even
more:

```
user_methods <- read_html("https://rocket.chat/docs/developer-guides/rest-api/users/")
user_methods %>%
  html_nodes("a") %>%
  html_attr("href") %>%
  grep("rest-api/users/.*/", ., value = TRUE)
```

```
 [1] "https://rocket.chat/docs/developer-guides/rest-api/users/presence/"
 [2] "https://rocket.chat/docs/developer-guides/rest-api/users/create/"
 [3] "https://rocket.chat/docs/developer-guides/rest-api/users/createtoken/"
 [4] "https://rocket.chat/docs/developer-guides/rest-api/users/delete/"
 [5] "https://rocket.chat/docs/developer-guides/rest-api/users/deleteownaccount/"
 [6] "https://rocket.chat/docs/developer-guides/rest-api/users/forgotpassword/"
 [7] "https://rocket.chat/docs/developer-guides/rest-api/users/generatepersonalaccesstoken/"
 [8] "https://rocket.chat/docs/developer-guides/rest-api/users/getavatar/"
 [9] "https://rocket.chat/docs/developer-guides/rest-api/users/getpersonalaccesstokens/"
[10] "https://rocket.chat/docs/developer-guides/rest-api/users/getpresence/"
[11] "https://rocket.chat/docs/developer-guides/rest-api/users/get-preferences/"
[12] "https://rocket.chat/docs/developer-guides/rest-api/users/getusernamesuggestion/"
[13] "https://rocket.chat/docs/developer-guides/rest-api/users/info/"
[14] "https://rocket.chat/docs/developer-guides/rest-api/users/list/"
[15] "https://rocket.chat/docs/developer-guides/rest-api/users/regeneratepersonalaccesstoken/"
[16] "https://rocket.chat/docs/developer-guides/rest-api/users/register/"
[17] "https://rocket.chat/docs/developer-guides/rest-api/users/removepersonalaccesstoken/"
[18] "https://rocket.chat/docs/developer-guides/rest-api/users/requestdatadownload/"
[19] "https://rocket.chat/docs/developer-guides/rest-api/users/resetavatar/"
[20] "https://rocket.chat/docs/developer-guides/rest-api/users/setavatar/"
[21] "https://rocket.chat/docs/developer-guides/rest-api/users/set-preferences/"
[22] "https://rocket.chat/docs/developer-guides/rest-api/users/setactivestatus/"
[23] "https://rocket.chat/docs/developer-guides/rest-api/users/update/"
[24] "https://rocket.chat/docs/developer-guides/rest-api/users/updateownbasicinfo/"    …
```