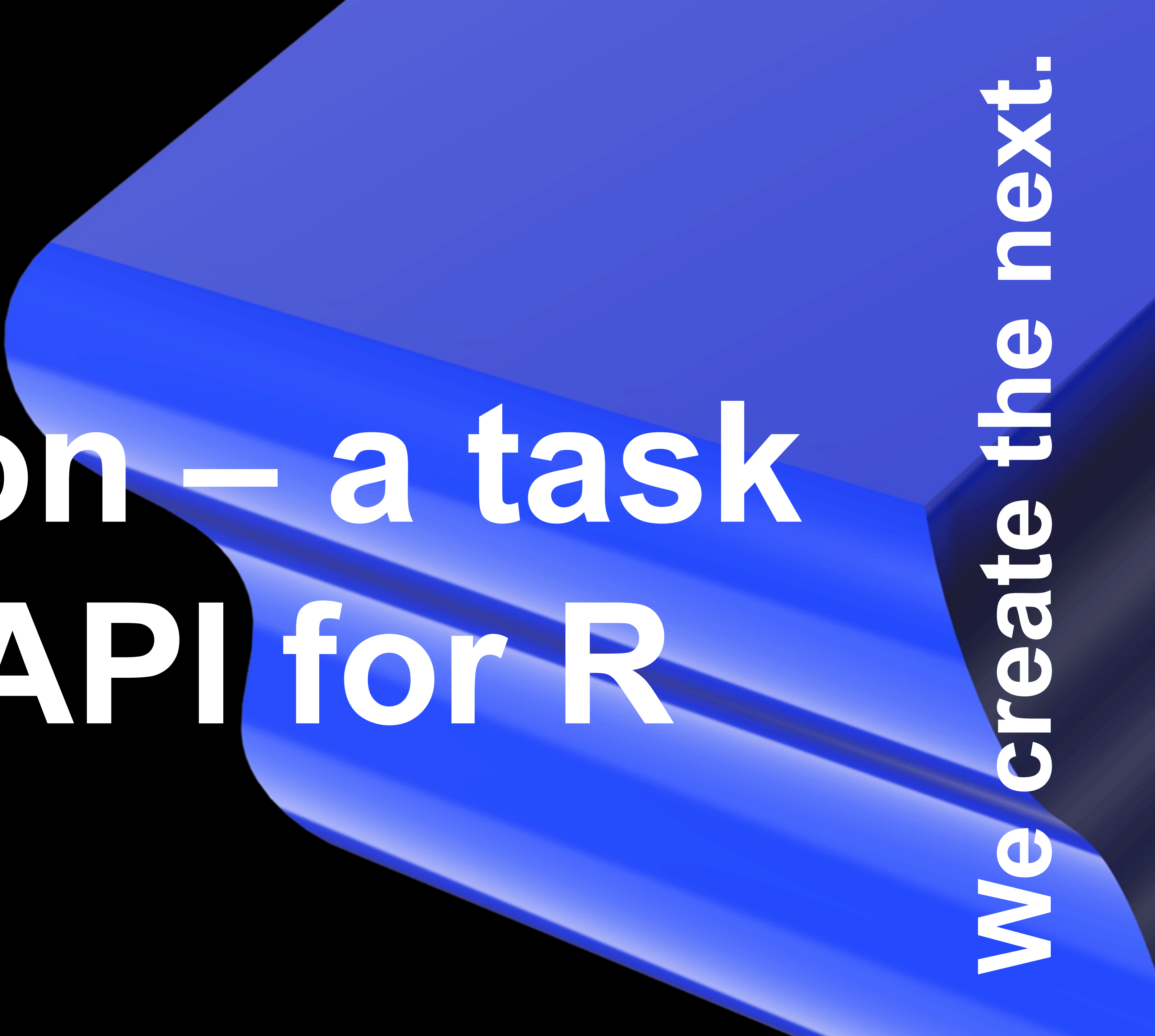
statworx®

LIGHTNING TALK

caRdoon – a task queue API for R



We create the next.

11.07.2024 – useR! 2024

About me

Jakob Gepp

Senior Consultant
statworx

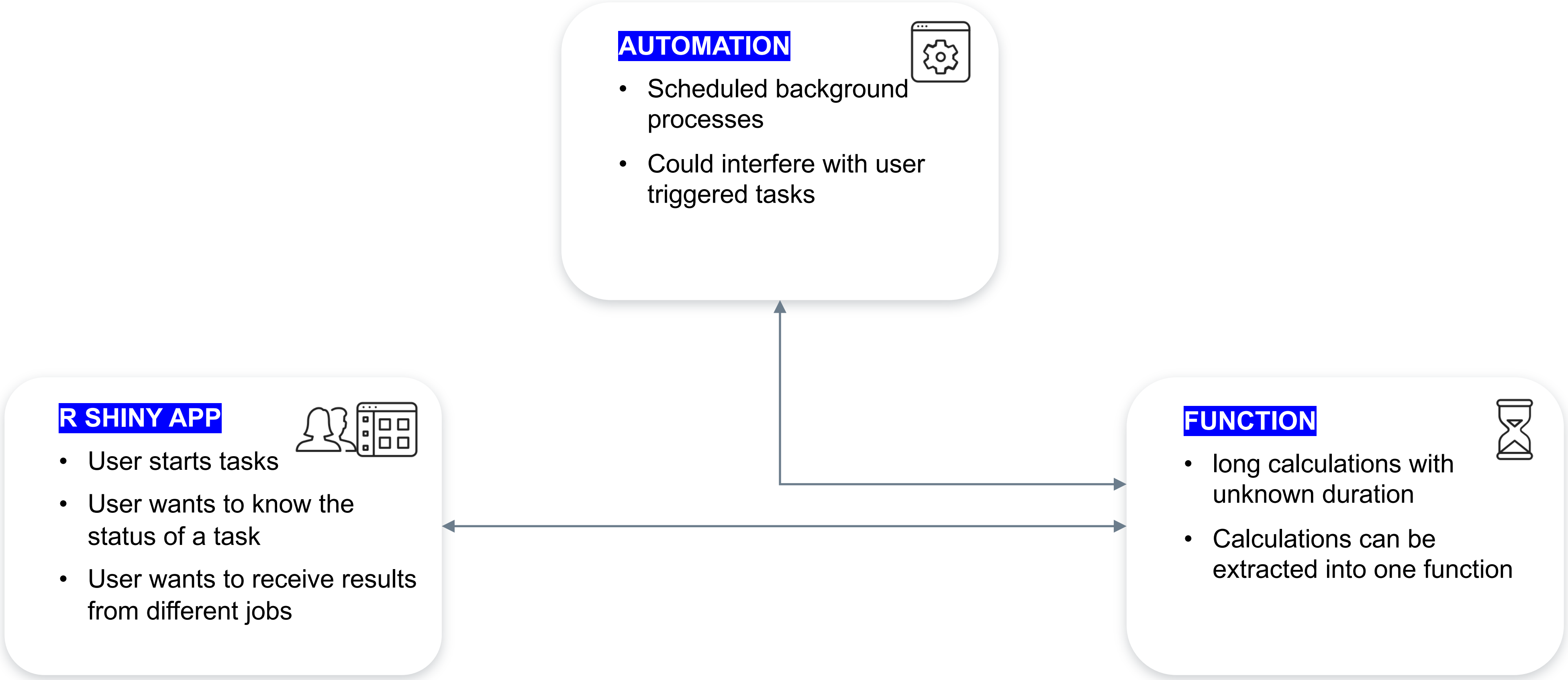
ABOUT ME

- Senior Consultant for Data Science at statworx
- R developer for about ten years
- I like to build frontend solutions and prefer data.table
- Published my first CRAN package last year (newsmd)

MOTIVATION

"I have built quite a few R Shiny apps and most of them had some kind of API connection to run a model in the background (e.g. for some forecasting).
If the model takes a while to compute, one questions arises:
What does the user do in the meantime?"

Multiple tasks with a long duration can reduce the user experience by blocking an app for an unknown time.



AUTOMATION

- Scheduled background processes
- Could interfere with user triggered tasks

R SHINY APP

- User starts tasks
- User wants to know the status of a task
- User wants to receive results from different jobs

FUNCTION

- long calculations with unknown duration
- Calculations can be extracted into one function

Developing a reusable framework that is simple to setup, easy to use and is quick to respond.

AUTOMATION

- Scheduled background processes
- Could interfere with user triggered tasks

R SHINY APP

- User starts tasks
- User wants to know the status of a task
- User wants to receive results from different jobs

CARDOON API

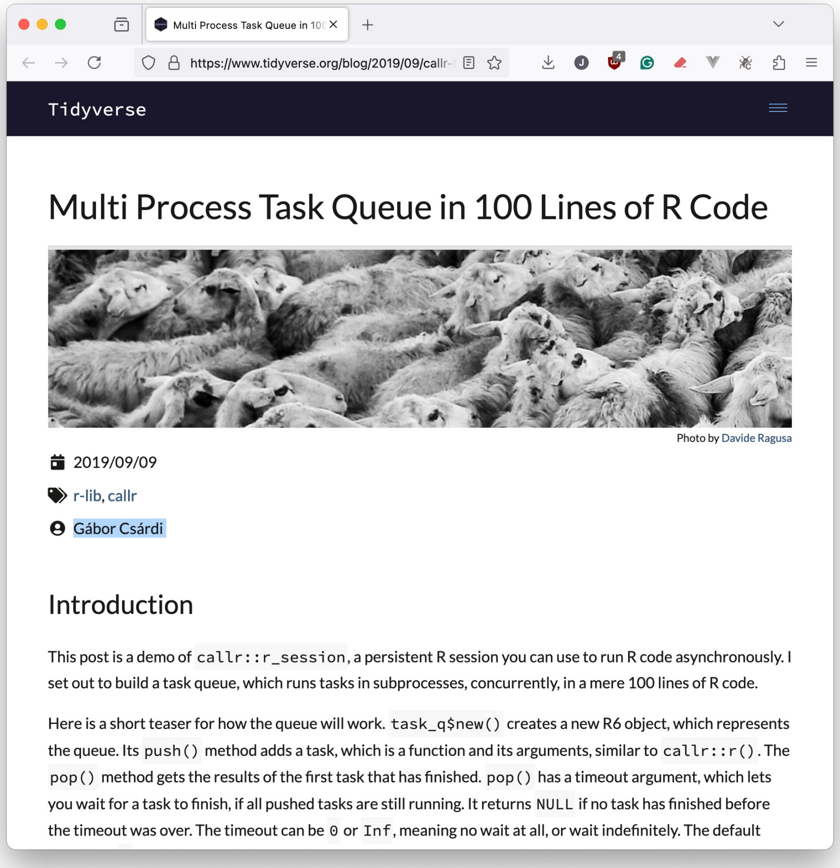
- Manages task queue
- Quick respond time
- Adjustable number of worker nodes

FUNCTION

- long calculations with unknown duration
- Calculations can be extracted into one function

My inspiration for this package was celery and a blog post by Gábor Csárdi from 2019 about callr.

CELERY


Multi Process Task Queue in 10
https://www.tidyverse.org/blog/2019/09/callr-
Tidyverse

Multi Process Task Queue in 100 Lines of R Code

Photo by Davide Ragusa

2019/09/09

r-lib, callr

Gábor Csárdi

Introduction

This post is a demo of `callr::r_session`, a persistent R session you can use to run R code asynchronously. I set out to build a task queue, which runs tasks in subprocesses, concurrently, in a mere 100 lines of R code.

Here is a short teaser for how the queue will work. `task_q$new()` creates a new R6 object, which represents the queue. Its `push()` method adds a task, which is a function and its arguments, similar to `callr::r()`. The `pop()` method gets the results of the first task that has finished. `pop()` has a timeout argument, which lets you wait for a task to finish, if all pushed tasks are still running. It returns `NULL` if no task has finished before the timeout was over. The timeout can be `0` or `Inf`, meaning no wait at all, or wait indefinitely. The default

https://github.com/celery/celery

https://www.tidyverse.org/blog/2019/09/callr-task-q/

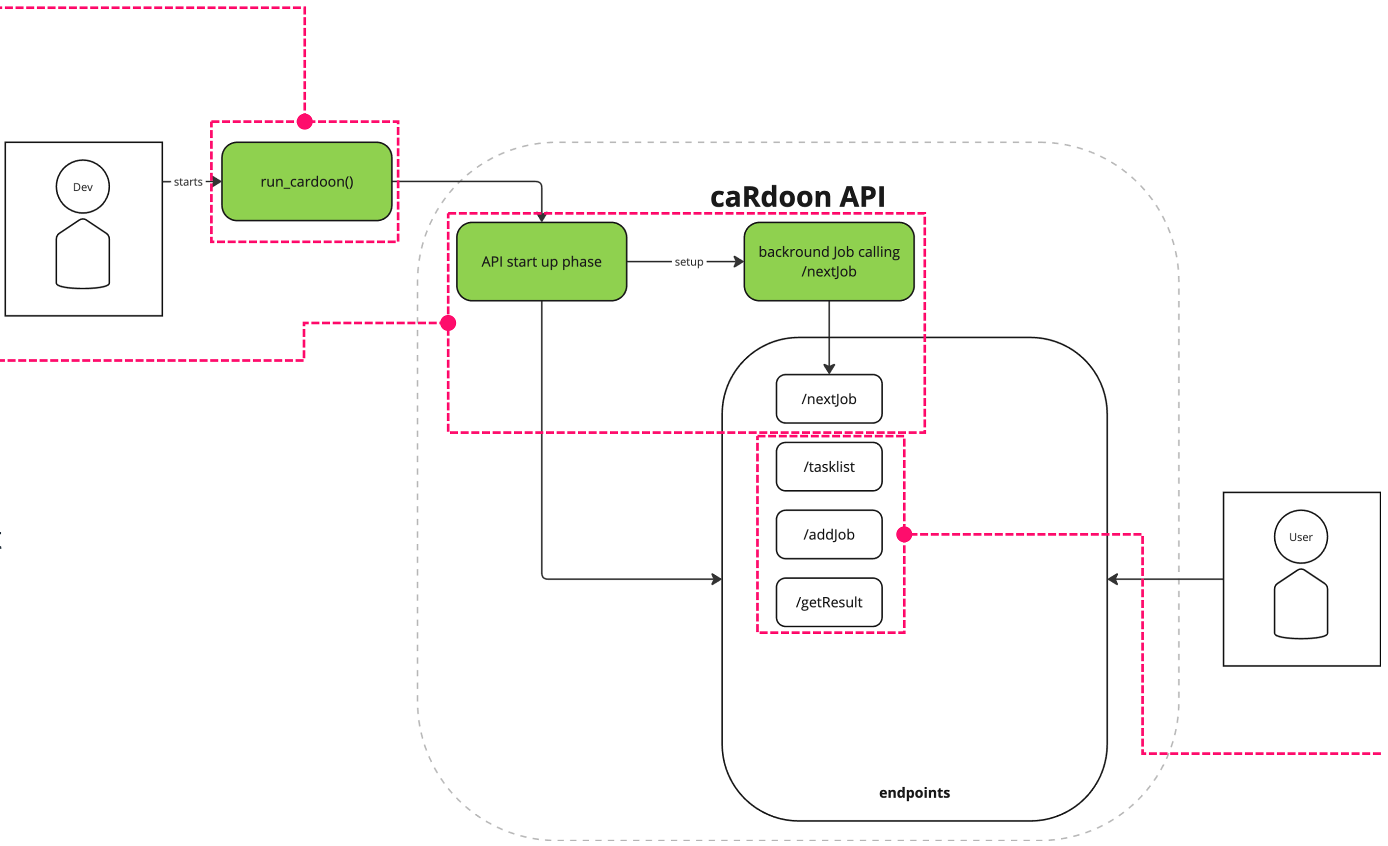
caRdoon separates the starting of tasks and their evaluation into different R processes.

PROVISIONING

- Starting run_cardoon()
- Providing a function to be evaluated at each call later
- API settings like the port, number of workers, etc.

START UP PHASE

- API initializes background processes and sets up worker nodes
- Creates a task-queue object

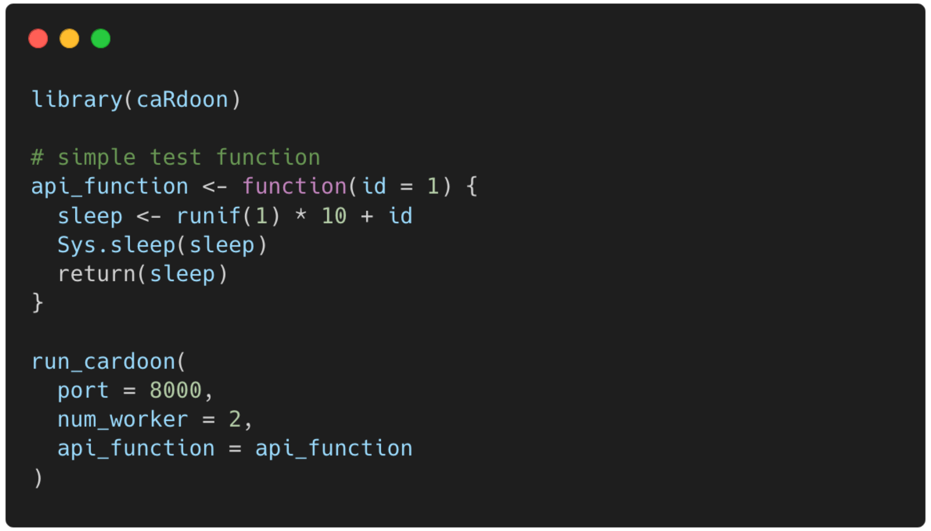


API ENDPOINTS

- Endpoints that should be exposed to the user
- They are all "fast", since there is no calculation needed

Using two R processes to setup the caRdoon API in one and create tasks in the other.

SETUP

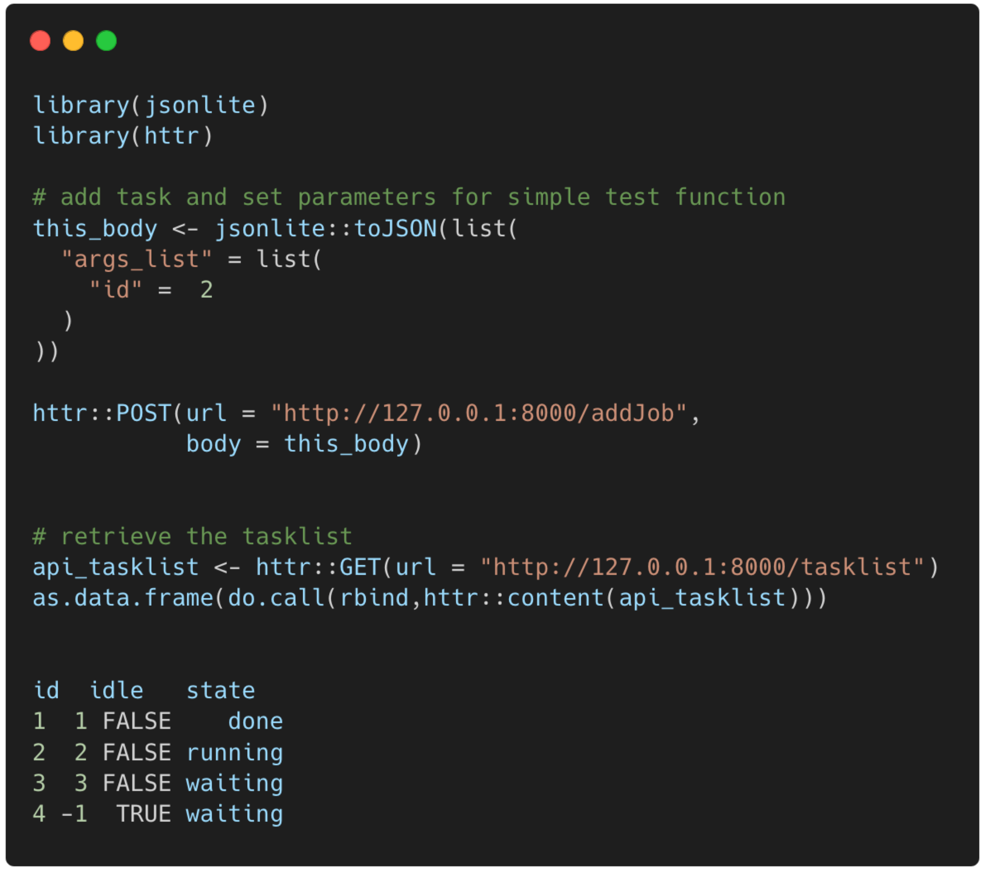


```r
library(caRdoon)

# simple test function
api_function <- function(id = 1) {
  sleep <- runif(1) * 10 + id
  Sys.sleep(sleep)
  return(sleep)
}

run_cardoon(
  port = 8000,
  num_worker = 2,
  api_function = api_function
)
```

CREATE TASKS



```r
library(jsonlite)
library(httr)

# add task and set parameters for simple test function
this_body <- jsonlite::toJSON(list(
  "args_list" = list(
    "id" =  2
  )
))

httr::POST(url = "http://127.0.0.1:8000/addJob",
           body = this_body)


# retrieve the tasklist
api_tasklist <- httr::GET(url = "http://127.0.0.1:8000/tasklist")
as.data.frame(do.call(rbind,httr::content(api_tasklist)))


id  idle   state
1  1 FALSE    done
2  2 FALSE running
3  3 FALSE waiting
4 -1  TRUE waiting
```

The heart of the API is an R6-object in combination with R background processes.

BACKGROUND PROCESS

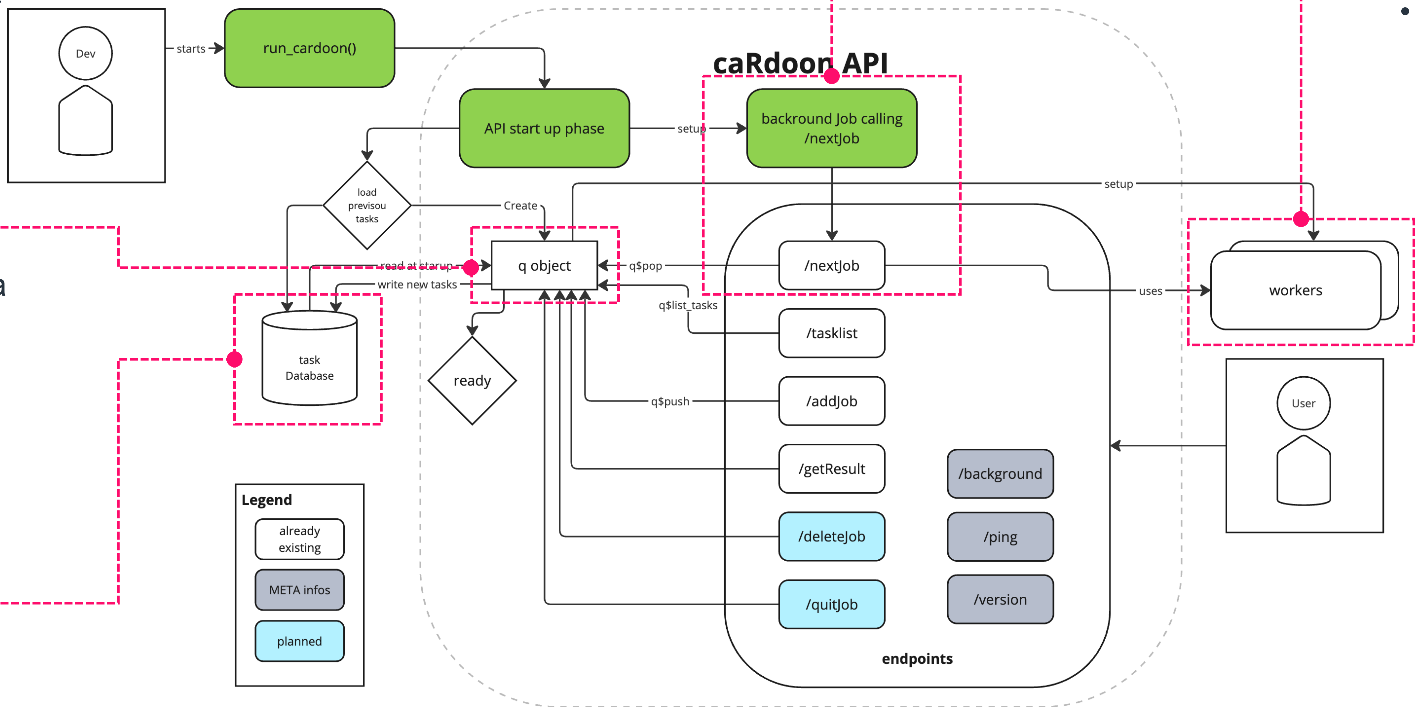
- A background process that calls in intervals /nextJob
- within /nextJob the status of the worker nodes is evaluated and advanced

Q-OBJECT

- An R6-object that contains a list of all task with their status and results
- Functions as the task manager (e.g. which is the next task to run)

DATABASE

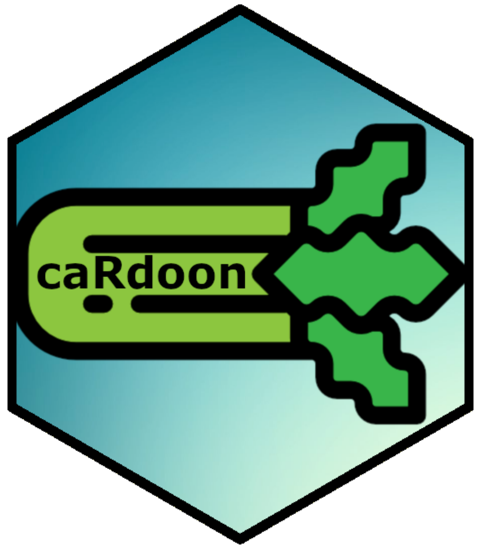
- A “copy” of the q-object
- persist the results and parameters
- can be fetched even after an API restart



WORKER NODES

- Each function is executed in a background process
- Number of nodes can be set with run_cardoon()

Thank you for your attention!



caRdoon

A task queue API for R.

Info **Summary**

An API to run an R function with different sets of parameters whilst keeping track of the current queue status and storing the results for later use.

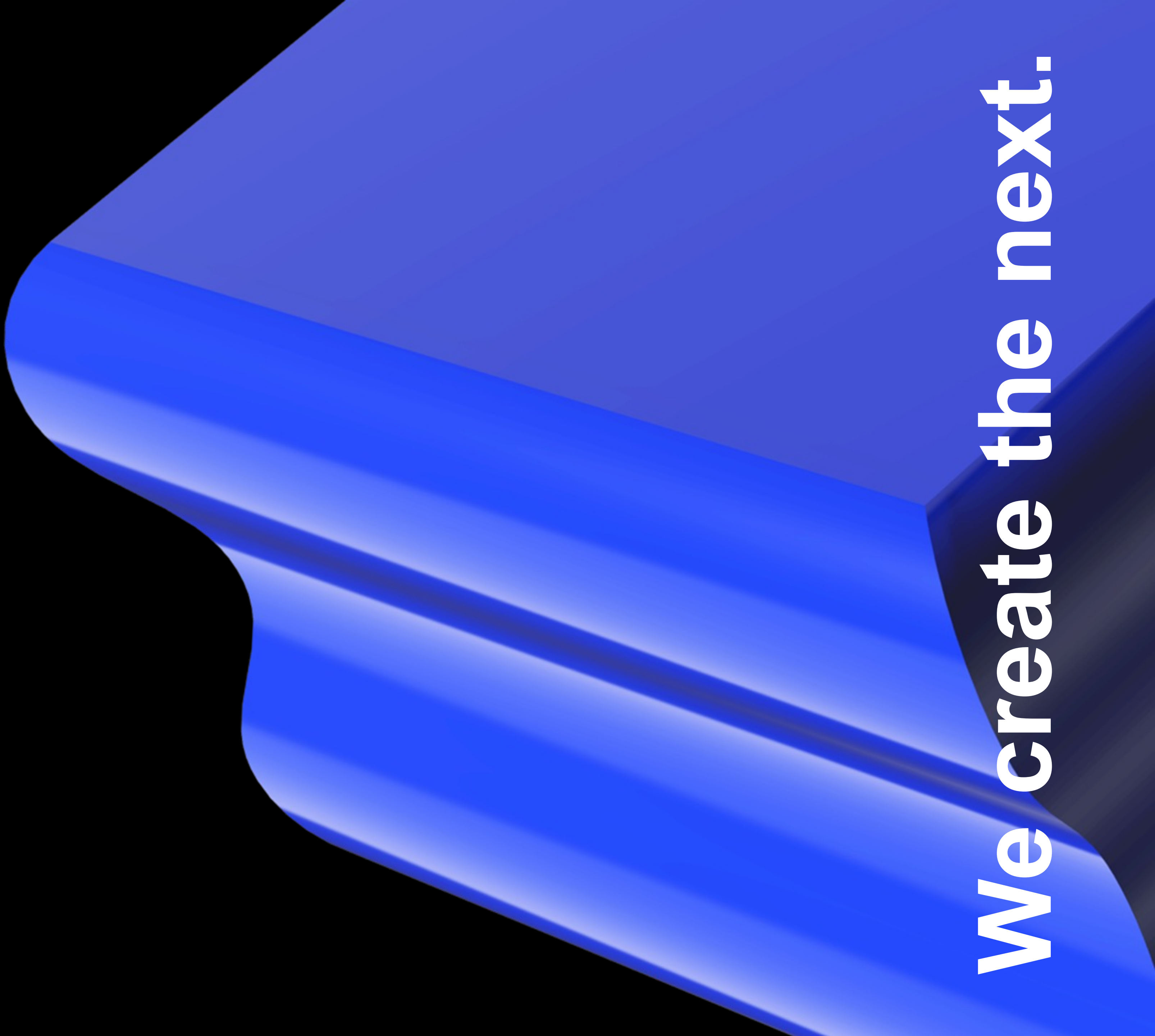
GitHub

https://github.com/Dschaykib/caRdoon

Contact

Jakob.gepp@statworx.com, Senior Consultant @ statworx

statworx®



We create the next.

statworx GmbH
Hanauer Landstr. 150
60314 Frankfurt am Main
+49 (0)69 6783 067 – 51
www.statworx.com