



Viriciti export assignment

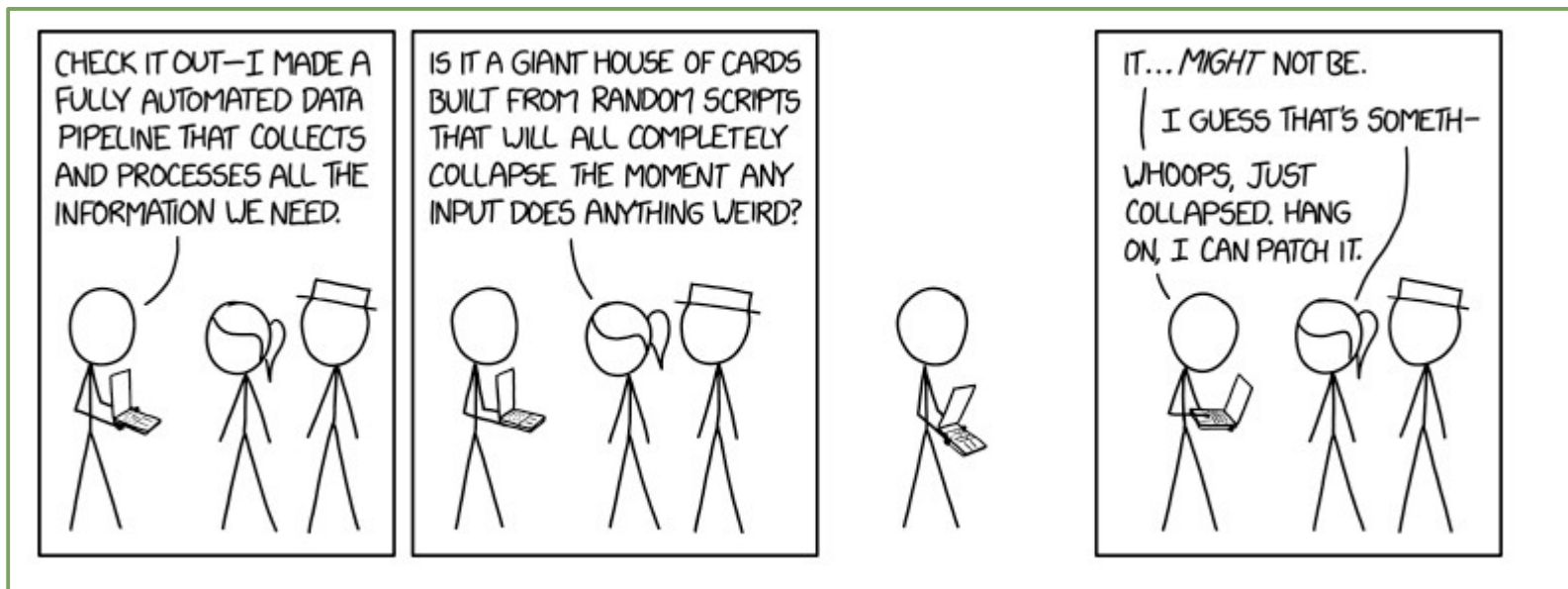
A fun and interesting code challenge.

Kiffin Gish

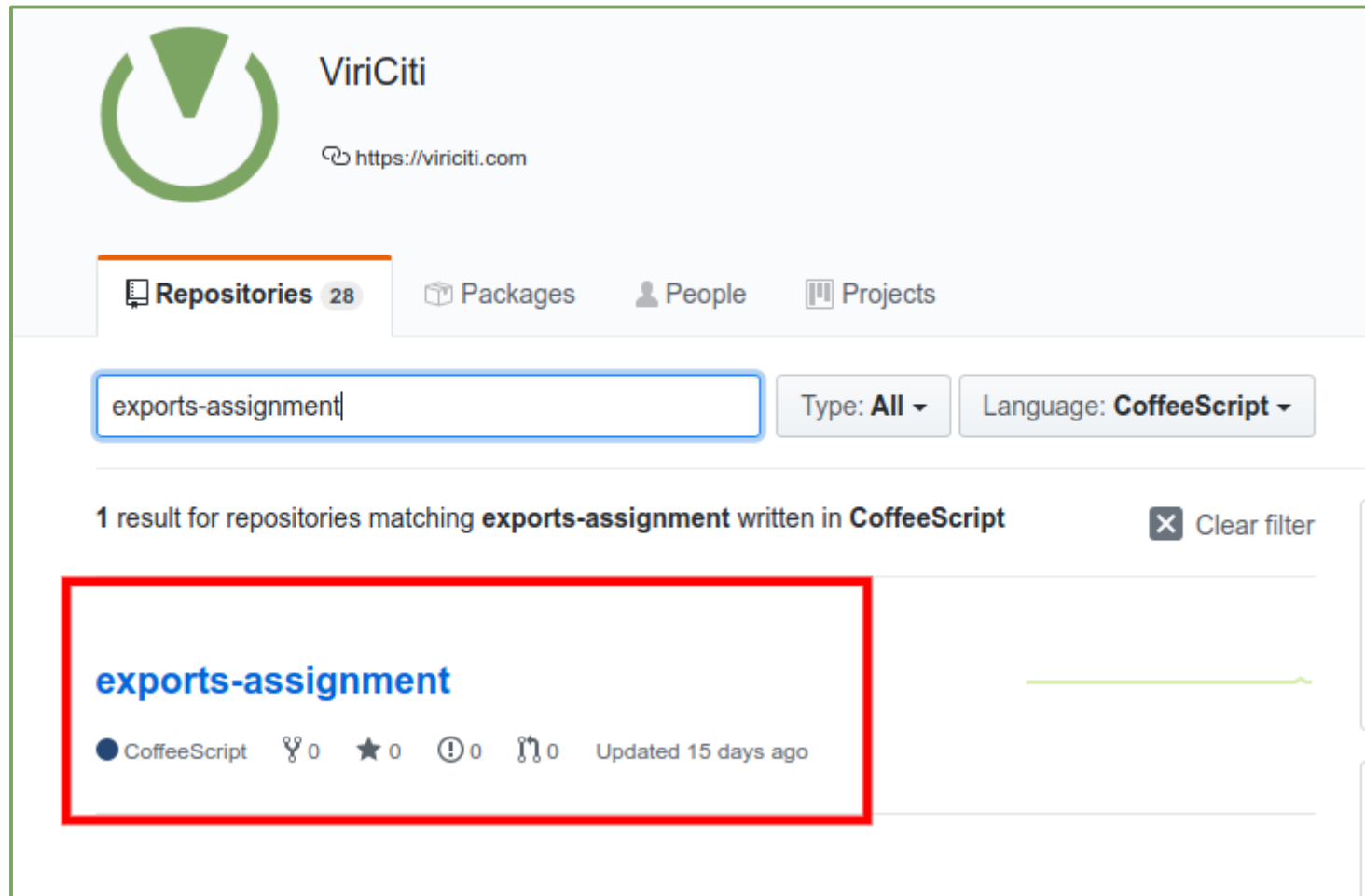
November 28, 2019

Introduction

In order to be able to better assess my technical expertise, the kind folks at Viriciti assigned me the following code challenge.



Repository



The screenshot displays the ViriCiti repository interface. At the top, the ViriCiti logo and name are shown, along with the URL <https://viriciti.com>. Below this, navigation tabs for 'Repositories' (28), 'Packages', 'People', and 'Projects' are visible. A search bar contains the text 'exports-assignment', and filters for 'Type: All' and 'Language: CoffeeScript' are applied. The results section shows '1 result for repositories matching exports-assignment written in CoffeeScript'. A red box highlights the first result, 'exports-assignment', which is a CoffeeScript repository with 0 forks, 0 stars, 0 issues, and 0 pull requests, updated 15 days ago.

ViriCiti
<https://viriciti.com>

Repositories 28 Packages People Projects

exports-assignment Type: All Language: CoffeeScript

1 result for repositories matching **exports-assignment** written in **CoffeeScript** Clear filter

exports-assignment

CoffeeScript 0 0 0 0 Updated 15 days ago

Code challenge

Build an CSV export system exposing an API.

For a given vehicle and range of dates, returns a data set of measured attributes.

Build front-end that accesses a robust back-end.

Must NOT overflow the database system when a lot of export requests are issued.

Plan of attack 1/2

Brush up on MongoDB.

Rewrite the unwind utility and test ES6/TypeScript.

Verify that the rewritten test works with MongoDB.

Create a MongoDB docker container and import the data dump.

Tried to learn Express Gateway but time too short.

Instead used NestJS (more familiar).

Build the back-end API server.

Plan of attack 2/2

Check out Balena (formerly resin.io) but time limited.

Learn Redis quickly.

Create a Redis docker container as cache.

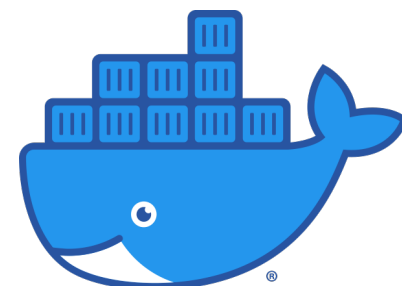
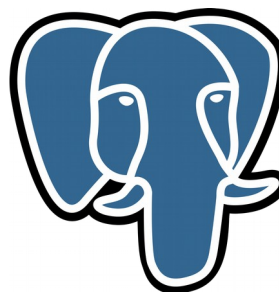
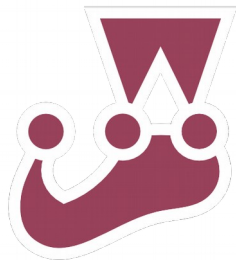
Build a responsive Angular front-end.

Dockerize everything.

Finalize the readme.

Practice this presentation in front of the mirror.

Some pre-requisites



Some pre-requisites



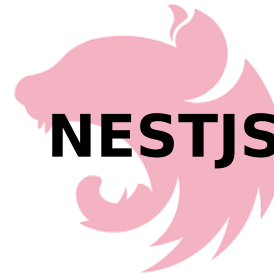
ANGULAR



JEST



REDIS



NESTJS



COCA-COLA



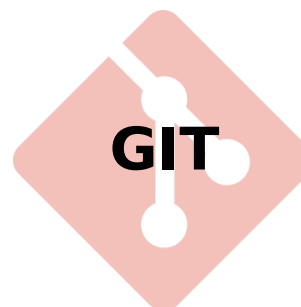
JASMINE



CHAI



CHARTJS



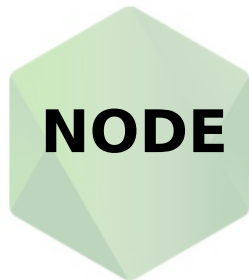
GIT



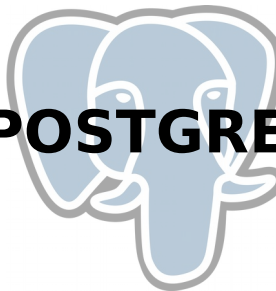
MONGODB



D3JS



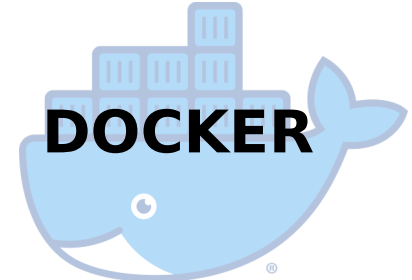
NODE



POSTGRES



MATERIAL



DOCKER

Specifications 1/2

Ensure that the export service will remain robust during periods of high traffic.

Support intensive usage by multiple users exporting large volumes of data simultaneously.

The MongoDB will not become overloaded and that no performance hits arise for those trying to export the data.

After some thought, I came up with the following strategy...

Specifications 2/2

Restrict access to authorized users (JWT).

Use a rate limiter to throttle API calls.

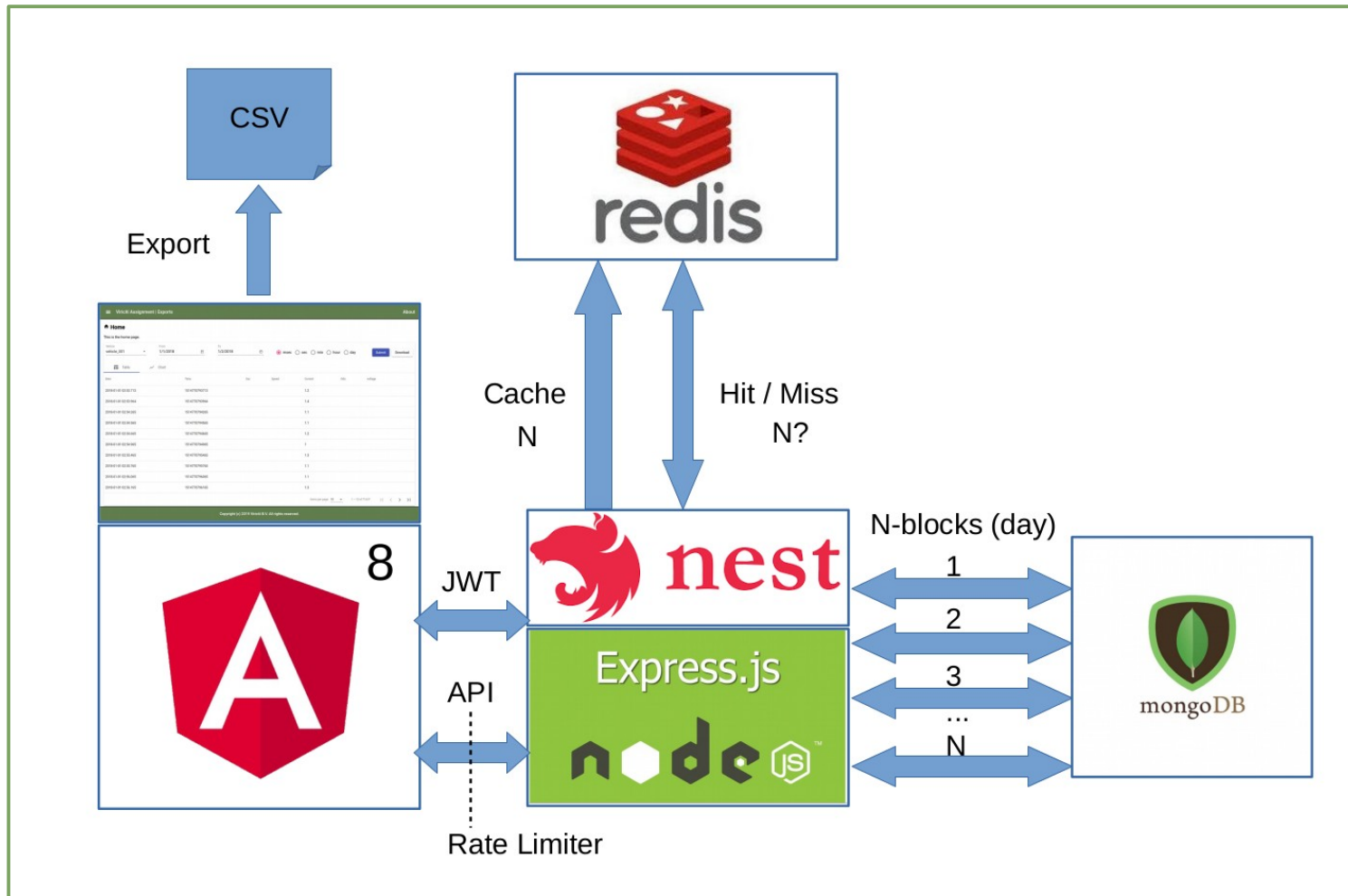
Cache data set results for recurring calls.

Slice date ranges into common blocks (days).

Offload the actual export to the front-end client.

Allow users to interact with the data set using date ranges, filters, sorting, etc.

Architecture



Code walkthrough


```
@Controller( prefix: 'vehicles')
@UseGuards(AuthGuard())
export class VehiclesController {

    private logger = new Logger( context: 'VehiclesController');



    constructor(private vehiclesService: VehiclesService) {
    }


    @Get( path: '/:id/values')
    getVehicleValuesById(
        @Query( property: 'fromDate') fromDate,
        @Query( property: 'toDate') toDate,
        @Param( property: 'id', ParseIntPipe) id: number,
        @GetUser() user: User,
    ): Promise<IValue[]> {
        this.logger.log(`getVehicleValuesById() user='${user}' id='${id}' fromDate='${fromDate}' toDate='${toDate}'`);
        return this.vehiclesService.getVehicleValuesById(id, user, fromDate, toDate);
    }
}
```

Demo

 VIRICITI

Assignment | Exports

 Kiffin 

 Start


Time to do something fun and exciting.

Vehicle

vehicle_001


From

1/1/2018



To

1/2/2018





☐ soc ☐ speed ☐ current ☐ odo ☐ voltage ☒ msec ☐ sec ☐ min ☐ hour ☐ day

Filter

Submit

Download





 Table

 Charts

Date	Time	Soc	Speed	Current	Odo	voltage
2018-01-01 02:39:53.713	1514770793713			1.2		
2018-01-01 02:39:53.964	1514770793964			1.4		
2018-01-01 02:39:54.265	1514770794265			1.1		
2018-01-01 02:39:54.565	1514770794565			1.1		
2018-01-01 02:39:54.665	1514770794665			1.2		
2018-01-01 02:39:54.965	1514770794965			1		
2018-01-01 02:39:55.465	1514770795465			1.3		
2018-01-01 02:39:55.765	1514770795765			1.1		
2018-01-01 02:39:56.065	1514770796065			1.1		
2018-01-01 02:39:56.165	1514770796165			1.3		

Items per page: 10

1 - 10 of 71637

Copyright (c) 2019 Viriciti B.V. All rights reserved.

Todo (rainy day)

Implement BSON to compress data exchange.

Fancier graphs with D3js.

Measure performance and tweak the code accordingly.

Optimize Redis configuration options.

Optimize MongoDB configuration options.

Smaller interval slices, hours or minutes.

Separate microservice for Redis caching.

Aggregate time mean sequences.

End

Thanks for your time and attention.

