Logotipo

Descripción generada automáticamenteTexto

Descripción generada automáticamente



NEO4J - Retail Use Case

Version 1.1

Technical Manual

Document/file

|  |  |
| --- | --- |
| Proyect | NEO4J - Retail Use Case |
| Deliverable | Technical manual |
| Author | Javier Plaza Baena |
| Version | v.1.0.0 |

Change register

|  |  |
| --- | --- |
| Version | v.1.1.0 |
| Date | 25/11/2021 |
| Author |  |
| Description | Initial version |

Tabla de contenidos

[1 | Introduction 4](#_Toc89093748)

[2 | Control sheet 4](#_Toc89093749)

[2.1 Document / Archive 4](#_Toc89093750)

[2.2 Change register 4](#_Toc89093751)

[2.3 Glossary of terms 4](#_Toc89093752)

[3 | Api 5](#_Toc89093753)

[3.1 How to use the api? 6](#_Toc89093754)

[3.2 Controllers 6](#_Toc89093755)

[4 | Neo4j Database model 10](#_Toc89093756)

[5 | Technical remarks 12](#_Toc89093757)

[Set neo4jclient connection 12](#_Toc89093758)

[Neo4j project structure 12](#_Toc89093759)

[Dependencies 13](#_Toc89093760)

[Properties 13](#_Toc89093761)

[Controllers 13](#_Toc89093762)

[Helpers 13](#_Toc89093763)

[Mappers 13](#_Toc89093764)

[Model 13](#_Toc89093765)

[Repository 13](#_Toc89093766)

[Scripts 13](#_Toc89093767)

[Services 14](#_Toc89093768)

[Program.cs 14](#_Toc89093769)

[Startup.cs 14](#_Toc89093770)

[6 | FRONT 15](#_Toc89093771)

[Modular Application 15](#_Toc89093772)

[External libraries 16](#_Toc89093773)

[Links 16](#_Toc89093774)

[7 | AWS Subscription 17](#_Toc89093775)

[Front: 17](#_Toc89093776)

[Back 17](#_Toc89093777)

[8 | AWS Architecture Diagram 20](#_Toc89093778)

# Introduction

This document describes the technical characteristics of the Neo4J retail use case project.

# Control sheet

## Document / Archive

|  |  |  |  |
| --- | --- | --- | --- |
| proyect | NEO4J - Retail Use Case | | |
| author | Javier Plaza Baena | | |
| Initial version | 25/11/2021 | **Fecha creada** | 25/11/2021 |

## Change register

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Version | Cause of change | | Responsible for change | Date of change | |
| 1.0 | Initial version | Javier Plaza | | |  |
| 1.1 | Architecture Diagram | Javier Plaza | | | 29/11/2021 |

## Glossary of terms

|  |  |  |  |
| --- | --- | --- | --- |
| nomenclature | description |  |  |
|  |  | | |
|  |  | | |
|  |  | | |
|  |  | | |

# Api

API built using .NET 5.0.

The solution contains two projects:

- SpikeNeo4j: API implementation.

- SpikeNeo4j.Test: unit tests for the controllers

To connect to the Neo4j database, we have installed the NuGet Neo4jClient (community driver) package in the SpikeNeo4j project. The connection parameters are stored in the appsettings.json file.

In addition, the following NuGets has been installed:

* Gmap.Net.Core: connection to google maps API to get truck routes and geolocate addresses.

The Google Maps API key is stored in the appsettings.json file.

* Automapper, this NuGet is used to map model objects to DTO objects.
* Microsoft.AspNetCore.Authentication.JwtBearer, which is used to generate JWT tokens for authentication purposes.
* Swashbuckle.AspNetCore, this package allows developing the API documentation.

URL Swagger: <http://apineo4jha-env.eba-aaavznqz.us-east-1.elasticbeanstalk.com/swagger/index.html>

## 3.1 How to use the api?

Get this API running:

* Neo4j Server 4. x - the easiest way is with Neo4j Desktop, just create a new 4. x version (any will work)
* Populate database - Once you have (1) you need to populate the database with data, to do that:  
  + Start Neo4j (using a command like this: C:\neo4j-community-4.3.5> bin\neo4j console)
  + Open the browser here.
  + Login using your user and password.
  + Go to file Scripts/populateDB.txt placed at SpikeNeo4j project, copy the content to the neo4j line command, and press play.

Now you're ready to use the API!

## 3.2 Controllers

There are two controllers implemented for this API. For more details about how to use their methods, you can have a look at the project swagger interface at /swagger/index.html.

**Neo4jClientController** (api/neo4j)

This is the api main controller, which allow ser to interact with neo4j database. At this controller contains the followings methods:

* Listdistributioncenters ([GET] /api/neo4j/ listdistributioncenters)

Get all the distribution centers in the database.

The Zoomlevel parameter is used to filter distribution centers with a zoomlevel value less than or equal to the parameter value.

* Listdestinationcenters ([GET] /api/neo4j/ listdestinationcenters)

Get all destination centers in the database.

The Zoomlevel parameter is used to filter destination centers with a zoomlevel value less than or equal to the parameter value.

* Listcenters ([GET] /api/neo4j/listcenters)

Get all centers (destination and distribution) in the database.

The Zoomlevel parameter is used to filter centers with a zoomlevel value less than or equal to the parameter value.

* Listcentersandrelations ([GET] /api/neo4j/listcentersandrelations)

Get all centers (destination and distribution) in the database and the relations between them. These relations are used to paint edges connecting nodes at the map.

The Zoomlevel parameter is used to filter centers with a zoomlevel value less than or equal to the parameter value.

* Distributioncenterdetails ([GET] /api/neo4j/distributioncenterdetails)

Get the detailed information for a distribution center with id equal to idDistributionCenter parameter.

* Distributioncentertrucks ([GET] /api/neo4j/distributioncentertrucks)

Gets the trucks that are on route belonging to a distribution center with id equal to idDistributionCenter. This method also returns the relationships between distribution centers and trucks.

* Distributioncentertruckonroute ([GET] /api/neo4j/distributioncentertruckonroute)

Get the route details for a truck with idTruck. The information returned is the starting distribution center, the destination center, the truck, and route details, indicating if there is a problem on route.

* Incidentdetaisandsolutionfortruckonroute

([GET] /api/neo4j/incidentdetaisandsolutionfortruckonroute)

Given a truck with idTruck, get the description of the incident on the route and a solution to resolve it.

* Distributioncentermagicplaces ([GET] /api/neo4j/distributioncentermagicplaces)

Given a distribution center with idDistributionCenter, get a list of magic places according to certain features.

* Magicplacedetails ([GET] /api/neo4j/magicplacedetails)

Given a magic place with idMagicPlace, get the details of the magic place.

* Getmagicplacefeatureslist ([GET] /api/neo4j/getmagicplacefeatureslist)

Get a list of magic place features that are used in the form to display them.

* Getpointsonroutetrucktodestination

([GET] /api/neo4j/getpointsonroutetrucktodestination)

Given a truck with idTruck, get a list of points on route to the destination. This is done using the google maps API (via gmap NuGet).

* Getpointsonroutetruckfromorigin

([GET] /api/neo4j/getpointsonroutetruckfromorigin)

Given a truck with idTruck, get a list of points on route from the starting distribution center.

This is done using the google maps API (via gmap NuGet).

* Getpointsforalternativeroutetrucktodestination

([GET] /api/neo4/getpointsforalternativeroutetrucktodestination)

Given a truck with idTruck, obtain a list of points representing an alternative route to the destination.

To simplify this task, the alternative route is calculated avoiding highways.

This is done using the google maps API (via gmap NuGet).

* Searchlocation ([GET] /api/neo4/searchlocation)

Given some location keywords, get a geolocation point for this city or address.

* Getships ([GET] /api/neo4/ getships)

Get a list of ships to display on the map.

**UsersController** (api/users)

* Login ([POST] /api/users/Login)

Given a username and password, this method attempts to log in a user and return a JWT token if the login is successful.

To simplify the task, the user's credentials are stored in the appsettings.json file.

# Neo4j Database model

URL: <https://64e5d198.databases.neo4j.io/>

User: neo4j

Pwd: LgMasS8xPZo1j7yWTwPCJ0v3ACz7nwkFJUm4DortHpo

Implemented model:

Chart, diagram, schematic

Description automatically generated

Graphical user interface, text, application, chat or text message

Description automatically generated

# Technical remarks

## Set neo4jclient connection

The Neo4j database connection is set at class Startup.cs, calling to this private method:

Pantalla negra con letras blancas

Descripción generada automáticamente

This method reads the neo4j connection parameters from the appsettings.json file.

## Neo4j project structure

The following contains a description of the project Neo4j structure.

Texto

Descripción generada automáticamente

## Dependencies

The Dependencies in the project contains all the installed server-side NuGet packages as mentioned above.

## Properties

The Properties node includes launchSettings.json file, includes Visual Studio profiles of debug settings.

## Controllers

The Controllers folder contains the developed controllers: Neo4jClientController y UsersController. These controllers hand incoming HTTP requests and send the response back to the caller.

## Helpers

The Helpers node includes some reusables classes to help with some repetitive tasks. In this project, MathHelper class is implemented mainly to calculate the middle point between two spatial points.

## Mappers

The Mappers folder contains object-object mappers for transforming an input object of one type into an output object of a different type. These mappers are mainly used to convert model objects to Dto objects.

## Model

The Model node involves all classes of the business model. It also contains the DTO classes.

## Repository

The Repository folder includes all classes which mediate between the domain and data mapping layers. The main repository here is ClientSpikeNeo4jRepository, which implements the queries to the Neo4j database.

## Scripts

This folder stores the populated.txt file. Use this file, as mentioned above, to fill the Neo4j database with data.

## Services

The Services node involves the GmapService, which is used to call Google Maps API to get information about routes, geolocated positions for keywords, etc.

## Program.cs

SpikeNeo4j is a console project which starts executing from the entry point public static void Main() in Program class where we can create a host for the web application.

Startup.cs  
  
ASP.NET Core application must include Startup class. It is like Global.asax in the traditional .NET application. As the name suggests, it is executed first when the application starts.

# FRONT

We have built this application using Angular 12.

Angular is a complete framework that has all the artifacts and requirements to build a SPA (single page application).

We use this framework to give robustness and provide an architecture to modern applications that set a standard in development and allow easy maintenance.

## Modular Application

For future developments, we have modularized as much as possible, thus increasing scalability and maintainability.

**Core**

In this module, we have all the necessary for our application to run correctly, including Components like header and footer.

**Shared**

Although it is empty, we have it prepared for extensions, and here should be everything that is used by various modules/features.

**Maps**

Here we have two main components, one component for the 2D map view and another for the 3D view.

If you want to add new functions later, they must be created in the same way as this module.

## External libraries

We have used different libraries to display maps and graphics.

Maps: in addition to using Leaflet, which is the one that manages everything related to connections and markers, we used a Mapbox for the style of the map.

For the 3D view, the library we chose was Miniature Earth.

**Leaflet**

Open-source JavaScript library for mobile-friendly interactive maps.

We use it to manage the markers and connections between the different distribution centers. It also helps us to trace the route traveled by each truck.

Miniature Earth

A library that uses ThreeJS for rendering the 3D view of the map.

**ChartJS**

ChartJS: detailed charts of distribution centers, which we use to generate sales, planning, and supply statistics.

**Apache ECharts**

ECharts: a powerful interactive charting and visualization library for browsers. We can see this library in the speedometer and the revolution counter in the incident mode of the truck.

### Links

[https://echarts.apache.org/](https://echarts.apache.org/en/index.html)

<https://www.chartjs.org/>

<https://miniature.earth/>

<https://leafletjs.com/>

# AWS Subscription

Graphical user interface, text, application, email

Description automatically generated

Users:

Graphical user interface, text, application

Description automatically generated

## Front:

Amazon S3 static website

s3angularneo4jha



## Back

AWS Elastic Beanstalk

Graphical user interface, text, application

Description automatically generated

|  |
| --- |
|  |
| **Category** | **Options** |
| Software | Enable 32-bit applications: False Log streaming: disabled Rotate logs: disabled Target .NET runtime: 4.0 X-Ray daemon: disabled |
| Instances | EC2 security groups: awseb-e-ksn2yx3cmg-stack-AWSEBSecurityGroup-1TVD5LV141OKU IMDSv1: enabled IOPS: container default Monitoring interval: 5 minute Root volume type: General Purpose (SSD) Size: container default Throughput: container default |
| Capacity | AMI ID: ami-0256c39590c09ee20 Availability Zones: Any Breach duration: 5 Capacity rebalancing: disabled Environment type: load balancing, auto scaling Instance types: t3.medium,t3.large Lower threshold: 2000000 Max: 2 Metric: NetworkOut Min: 1 Period: 5 Placement: Scale down increment: -1 Scale up increment: 1 Scaling cooldown: 360 seconds Statistic: Average Unit: Bytes Upper threshold: 6000000 |
| Load balancer | Listeners: 1 Load balancer type: application Processes: 1 Rules: 0 Shared: false Store logs: disabled |
| Rolling updates and deployments | Batch size: 100% Batch size: 1 Command timeout: 600 Deployment policy: Immutable Healthy threshold: Ok Ignore health check: disabled Minimum capacity: 1 Rolling update type: Rolling based on Health Rolling updates: enabled |
| Security | EC2 key pair: -- IAM instance profile: aws-elasticbeanstalk-ec2-role Service role: arn:aws:iam::316332460124:role/aws-elasticbeanstalk-service-role |
| Monitoring | CloudWatch Custom Metrics-Environment: CloudWatch Custom Metrics-Instance: Health event log streaming: disabled Ignore HTTP 4xx: disabled Ignore load balancer 4xx: disabled System: Enhanced |
| Managed updates | Instance replacement: disabled Managed updates: enabled Update level: Minor and patch Weekly update window: Fri:17:00 |
| Notifications | Email: -- |
| Network | *This environment is not part of a VPC.* |
| Database |  |

# AWS Architecture Diagram

Diagram

Description automatically generated