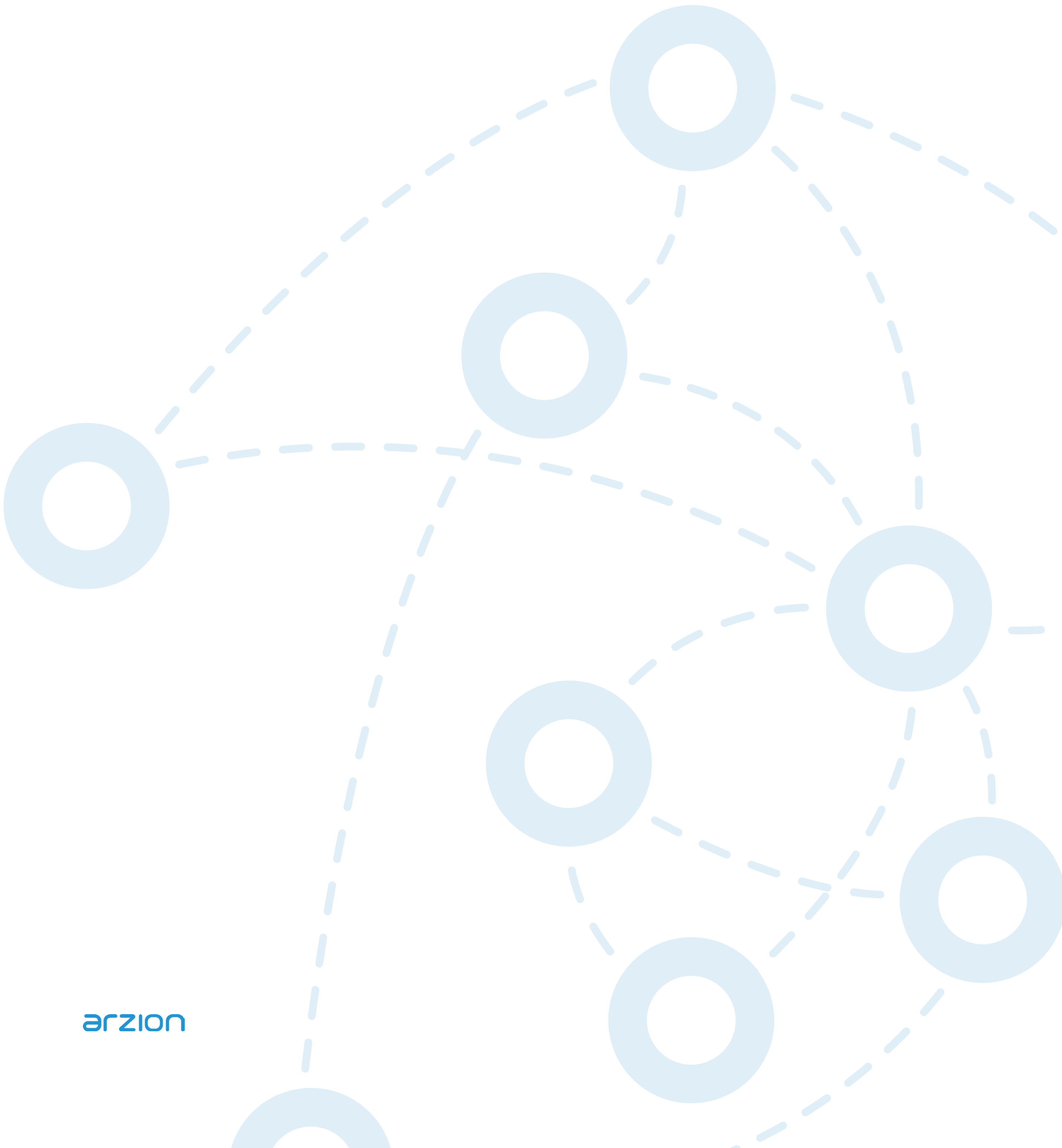
JAVASCRIPT BACKEND TEST

Logistic optimization



arzion

INTRODUCTION

- ARZ is a logistic company that delivers packages coming from all around the world to Argentina.
- The company has a main office that receives, classifies and distributes all the packages between 10 storage points in different cities.
- It warehouses have trucks that delivers the packages to the final customers.
- Depending on their resources, each warehouse has a limit of packages that it can receive and deliver per day:

Warehouse	City	Limit
WH01	Buenos Aires	200
WH02	Rosario	70
WH03	Córdoba	150
WH04	Trelew	140
WH05	Mendoza	150
WH06	La Plata	100
WH07	San Miguel de Tucumán	120
WH08	Mar del Plata	180
WH09	Salta	140
WH10	Santa Fe	70

COMPANY LOGIC

- Deliver a package from a warehouse to the client's house costs 1 USD per 5 km traveled.
- The penalty for late delivery to the final client is USD 70 per day delayed.
- A warehouse can't be overloaded.
- When a warehouse reaches a 95% of it limit, an alert is triggered to the main office. It has to decide if is more convinient start sending the packages to nearby cities or pay the late delivery penalty.
- Transport a package from the main office to a warehouse doesn't have any cost.

YOUR JOB

- Create a data model (DER and classes diagram).
- Develop a RESTful service using Node.js based technologies, implementing the model of the previous step.
- Test your system delivering 1000 packages to random cities across Argentina.

REQUIREMENTS

- Use **Google Maps API** to calculate distances with a **dev API key via http request** or use **any npm library that implement this service**.
  For example: https://github.com/ecteodoro/google-distance-matrix
- Use a **JavaScript framework** like **Sails** or **NestJS**.
- Use an **ORM** and **MySQL** as data base.
- You need to create a **GitHub repository** and send us the link so we can see the progress.
- Please, try to **commit often and use clear and concise commit messages**.
- **Unit testing** and **E2E**. Use **Jest** to test the application.
- Each component must have its **own test file**.
- Every developed test **should be relevant**.
- Try to reach a **good percentage of coverage**.
- Include **ESLint using Airbnb's ESLint Rules** (eslint-config-airbnb).
- Keep the use of **third-party packages to a minimum**.
- Try to use the **latest versions of the packages** that are included.
- Use **design patterns**.
- Complex logic must be **documented**.
- Code and comments must be in **English**.