# Proceso de Reclutamiento - dridco

## Cuestionario

1. Nombra tecnologías que te hayan gustado (o te gustaría) usar/aprender. ¿Cuáles serían los motivos?

Las tecnologías que utilice que me gustaron hablando de front end fueron los frameworks PrimeFaces, ya que permitía un desarrollo rápido siempre y cuando se usen los módulos que el framework provee, y Java Server Face, ya que permitía un desarrollo de páginas sencillo, intuitivo, y mantenible.

Las tecnologías que me gustaría aprender que ya estuve investigando son Play Framework con Scala junto a AngularJs y Bootstrap, debido a que vi aplicaciones realizadas sobre esta arquitectura y me parece muy interesante la rapidez con la que se puede desarrollar.

1. Nombra tecnologías que no te hayan gustado (o no te gustaría) usar/aprender. ¿Cuáles serían los motivos?

Las tecnologías que no me gustaron que use laboralmente fueron WebLogic, AquaLogic y Oracle Service Bus, debido a que eran herramientas que facilitaban el traspaso de mensajes entre diferentes sistemas pero no presentaban ningún desafío más que el uso de las herramientas.

1. ¿Qué cosas le criticarías a Java, ya sea la plataforma o el lenguaje? Detalla los motivos.

Java al ser un lenguaje orientado a objetos pero a su vez permite estructuras de lenguaje de programación estructurado puede generar confusiones si no se tiene en claro cómo utilizarlo.

He visto porciones de códigos totalmente orientados al lenguaje estructurado y no a objetos, lo cual en una aplicación donde muchas personas trabajan puede generar problemas de mantenibilidad.

Una ventaja que a su vez es una desventaja en Java es el uso de la JVM, ya que al permitir portabilidad (Gran ventaja) sacrifica rendimiento en comparación con otros lenguajes de programación más a bajo nivel como C++ (También con orientación a objetos) o C.

1. ¿Qué otro lenguaje/plataforma manejaste? ¿Lo seguís utilizando? ¿En qué ámbito ¿Que impidió que le dedicases más tiempo?

Utilice C para trabajos prácticos de la facultad, el cual no sigo utilizando debido a que no estoy interesado en ser un experto en este lenguaje.

También utilice SmallTalk, ProLog y Haskell en ámbito facultativo y no los utilice más por ser de puro uso académico.

Adicionalmente estoy aprendiendo Scala, el cual me parece muy interesante debido a que tiene todas las ventajas de un lenguaje tipado orientado a objetos y de los lenguajes funcionales.

1. Si tuvieras que enumerar, ¿cuáles consideras deberían ser las 5 cosas que todo desarrollador Java experimentado debería tener en cuenta o practicar?

* Patrones de diseño (No necesariamente como se llaman, sino saber que existen y que problemas pueden resolver)
* Clean Code y Refactor (Saber codear de forma clara y refactorizar lo que sea necesario para que la mantenibilidad del código sea buena)
* Frameworks de moda como Spring, Hibernate, Junit.
* Conocimiento de repositorios distribuidos como github que facilitan el desarrollo de aplicaciones en equipo
* Buena base de diseño orientado a objetos

1. ¿Qué webs, blogs, feeds revisás regularmente?

Infobae, Ole, Empresores.com.

1. ¿Cuáles fueron los últimos 2 libros que leíste?

“El mago de los millonarios” y “Del colchón a la inversión”

1. ¿Qué aportes pudiste hacer a proyectos de Software Open Source? Bugs, feature requests, design proposals, etc. (Incluir vínculos web)

Ninguno

¿Cómo te imaginas la web de acá a 5 años?

Me imagino la web con más contenidos y más organizados, con mucho énfasis en redes sociales orientadas a determinados rubros y personalizadas para estos.

## Homework

* Utilizar lenguaje Java (5 o superior) – Lenguaje Utilizado Java 7
* No utilizar librerías externas a JRE para el desarrollo de la solución, exceptuando aquellas que sólo sean utilizadas para testing o empaquetado de la misma. Librerías externas utilizadas: Junit4
* Utilizar Maven 3 para construcción.
* Incluir una explicación breve sobre las decisiones de diseño. [Ver: Breve explicación de diseño](#Breve explicacion de diseño)
* Incluir una breve guía sobre la ejecución de la solución. [Ver: Guía de ejecución](#Guia de ejecucion)

Las soluciones desarrolladas deben ser completas, acompañadas con las pruebas automatizados que lo demuestren. El proyecto debe funcionar con una invocación de consola (*mvn test*). No es necesario implementar un frontend de ningún tipo.

La solución podrá ser debatida durante el proceso de entrevistas presenciales.

Enviar las soluciones completas dentro de los 3 días mediante un enlace a un repositorio Git o SVN privado, incluyendo a las cuentas gbermudez@dridco.com y eszechet@dridco.com con acceso de lectura. Ver: Repositorio

Avisar en caso de necesitar más tiempo o por cualquier consulta.

# Problema Ferroviario

Supóngase un sistema ferroviario donde todas sus vías son 'dirección-única', esto es, una ruta desde Villa del Parque hacia Devoto no implica una ruta desde Devoto hacia Villa del Parque. De hecho, si ambas rutas existiesen, serían distintas y no necesariamente recorrerían la misma distancia.

El objetivo de la solución es proveer información de rutas a los usuarios. La solución deberá poder resolver:

* La distancia de una ruta determinada
* La cantidad de rutas existentes entre dos ciudades
* La ruta más corta entre dos ciudades

### ENTRADA

Un grafo dirigido donde cada nodo representa una ciudad, y un vínculo representa una ruta entre dos ciudades. El peso sobre el vínculo representa la distancia entre las dos ciudades vinculadas. Una ruta nunca puede informarse más de una vez, y para una ruta dada, el comienzo y el fin nunca puede ser la misma ciudad.

### SALIDA

Para los datos de entrada 1 a 5, si la ruta no existe, imprimir por salida el mensaje: 'SIN RUTA'.

De lo contrario, seguir la ruta definida, sin hacer paradas extras. Por ejemplo: el ejercicio 1 implica iniciar en la ciudad A, viajar a la ciudad B (una distancia de 5) y luego directo a la ciudad C (distancia de 4).

### CASOS

1. La distancia de la ruta A-B-C
2. La distancia de la ruta A-D
3. La distancia de la ruta A-D-C
4. La distancia de la ruta A-E-B-C-D
5. La distancia de la ruta A-E-D
6. La cantidad de viajes partiendo desde C y terminando en C con un máximo de tres paradas. Con el grafo provisto de ejemplo, pueden encontrarse 2 viajes posibles con dichas características: C-D-C (2 paradas) y C-E-B-C (3 paradas)
7. La cantidad de viajes partiendo desde A y terminando en C con exactamente 4 paradas. Con el grafo provisto de ejemplo, pueden encontrarse 3 viajes posibles con dichas características: A hacia C (vía B-C-D); A hacia C (vía D-C-D); y A hacia C (vía D-E-B).
8. La distancia de la ruta más corta desde A hasta C
9. La distancia de la ruta más corta desde B hasta B
10. La cantidad de rutas distintas desde C hasta C con una distancia menor a 30. Con el grafo provisto de ejemplo pueden encontrarse los siguientes viajes con dichas características: C-D-C, C-E-B-C, C-E-B-C-D-C, C-D-C-E-B-C, C-D-E-B-C, C-E-B-C-E-B-C, C-E-B-C-E-B-C-E-B-C.

### DATOS

Para las pruebas, las ciudades se nombran de la A a la E. Una ruta entre 2 ciudades (A hacia B) con distancia de 5, se representa: **AB5**.

### GRAFO EJEMPLO

**AB5, BC4, CD8, DC8, DE6, AD5, CE2, EB3, AE7**

### RESULTADOS ESPERADOS

1. 9
2. 5
3. 13
4. 22
5. SIN RUTA
6. 2
7. 3
8. 9
9. 9
10. 7

**Breve explicación de diseño:**

Para realizar el ejercicio lo primero que me plantee fue como modelar el grafo, para eso utilicé las clases City y Path siendo estas análogas a Nodo y Arista, luego de esto definí el objeto AreaMap que tendría todas las ciudades y rutas.

Investigando un poco encontré algunos métodos en internet que me facilitaron la carga de objetos para recorrer el grafo y poder implementar mis propios métodos adaptados para la solución. Siendo así, encontré que el algoritmo de dijkstra me serviría para encontrar el camino más corto, con lo cual implemente una versión simple del mismo para ese punto.

Teniendo métodos que me permitían calcular distancia entre dos ciudades y obtener ciudades adyacentes la implementación de métodos propios para resolver los problemas planteados fue bastante intuitiva utilizando métodos recursivos.

Todos estos métodos de los cuales estoy hablando se encuentran dentro de la clase AreaMapService.

Cada ejercicio planteado esta con su respectivo test unitario que corrobora el funcionamiento con el ejemplo presentado en el ejercicio.

**Guía de ejecución:**

Para más información de como ejecutar el programa por consola ver el Readme.md en el proyecto

Repositorio:

<https://github.com/martinlocurciooe/dridco>