

Felicitaciones solo el 10% de los entrevistados llegaron a la etapa final y sos uno de ellos! Ya demostraste tus habilidades a la hora de programar y razonar ejercicios lógicos, también demostraste tus conocimientos teóricos, es hora de demostrar tus habilidades de aprendizaje y de resolución de problemas reales.

Antes que nada, te presentamos a Jhipster <https://www.jhipster.tech/>, que es un autogenerador de código open source. A partir de un JDL (<https://start.jhipster.tech/jdl-studio/>) puedes crear sistemas en Java (u otro lenguaje de Backend) con casi cualquier base de datos relacional y no relacional y puedes elegir casi cualquier framework o biblioteca para realizar el Frontend. Esto permite crear la base de un sistema rápidamente, permitiendo acelerar los tiempos de desarrollo y permitiendo también mostrar un prototipo al cliente de manera casi instantánea.

El equipo de ingeniería de Merlion Techs está compuesto principalmente por tres equipos. En esta oportunidad estás postulando para incorporarte en dos de ellos.

- **Producto:** Este equipo se encarga de enriquecer las funcionalidades del producto que serán utilizadas directamente por el usuario final.
Está más enfocado al desarrollo de tareas de front end (aunque a veces deban realizar alguna tarea sencilla de backend).
- **Analytics:** Este equipo se encarga de enriquecer el producto con estadísticas, recomendaciones, alertas y demás.
Está más enfocados en tareas backend (aunque a veces deban realizar tareas sencillas de front end)

Tendrás la libertad de realizar el test para el equipo que desees.

Parte en común

Apliques para el puesto que apliques, deberás instalar el ambiente para poder desarrollar una aplicación utilizando Jhipster.

Deberás tener instalado

- PostgreSQL 12.x.x <https://www.postgresql.org/download/>
- Java 8 <https://www.java.com/es/download/>
- Node <https://nodejs.org/es/download/>
- Git
<https://git-scm.com/book/es/v2/Inicio---Sobre-el-Control-de-Versiones-Instalaci%C3%B3n-de-Git>

Instalar JHipster (<https://www.jhipster.tech/>): `npm install -g generator-jhipster`

Crear un nuevo directorio con el nombre **merlionTestApp** y dentro correr el comando **jhipster**

Configuración de JHipster: (Pueden utilizar este tutorial para guiarse https://www.youtube.com/watch?v=7_UlnZlalyw. Las configuraciones del tutorial son similares a las que se requieren en este test, pero **NO** son todas las mismas).

Java	8
Tipo de aplicación	Monolítica
Nombre base de la aplicación	Test

Default java package name	merlionTechs
Use Jhipster Registry	No
Auth	JWT
Production database	SQL - > PostgreSQL
Development database	PostgreSQL
Spring cache	Ecache
Hibernate 2 level cache	Si
Maven o Gradle	Maven
Motor de búsqueda	Ninguno
Front end	React
Bootstrap	Default Jhipster
Internationalization support	Si
Native language	Español
Motor de tests avanzado	No
Makertplace	No

- Si no estás muy familiarizado con React y TypeScript, recomendamos ver el siguiente video <https://www.youtube.com/watch?v=lbJFERe9F9w>.
- Si necesitas ayuda con la parte del backend, especialmente con Spring Boot y Hibernate, mira esta serie de videos <https://www.youtube.com/watch?v=UpdkMjunXSk&list=PLclHm18h1i4m1xuhwrL-LjVjf5wuYFRCV>
- Si no estas familiarizado con git <https://learngitbranching.js.org/>

Deberás utilizar componentes de <https://material-ui.com/> y cualquier modificación de la UI deberá realizarse siguiendo los estándares de material ui. Y la paleta de colores deberá seguir la estética de <https://www.merliontechs.com/>.

Si crees que ya has realizado tu ejercicio correctamente, te invitamos a realizar bonus tracks para sumar puntos.

Cada tarea que decidas realizar debe realizarse en un commit separado, en un branch separado partiendo desde master.

Debe entregarse en un repo de **github e imágenes o un video** de la aplicación. Esto debe enviarse al email contrataciones@merliontechs.com con el asunto *Ejercicio Final Entrevista Merlion Techs*.

Test Producto

Siéntete libre de modificar el jdl si lo deseas, siempre y cuando no alteres el objetivo del ejercicio.

Introducir el siguiente JDL en un archivo con el nombre **testProducto-jdl.jh** para crear una aplicación Jhipster

```
entity Product {
    name String
}

entity Sales {
    state State
}

enum State {
    IN_CHARGE, SHIPPED, DELIVERED
}

relationship ManyToOne {
    Product{product} to Sales
}
```

Una venta solo tendrá un producto, pero un producto puede estar en muchas ventas.

Correr en consola: `jhipster import-jdl testProducto-jdl.jh`

Una vez autogenerado el código crear un commit con el nombre “Auto Generated Code” o un nombre similar.

Deben realizar una pantalla que con tres pestañas que permita pasar un producto de IN_CHARGE -> SHIPPED -> DELIVERED

Estos cambios no deben hacerse solo en front end, sino también en backend, el cambio debe verse reflejado sin necesidad de actualizar/refrescar la pantalla.

Tomar esta imagen como EJEMPLO

Development

MerlionTechs DEV

[Home](#) [Entities](#) [Administration](#) [English](#) [Account](#)

ENCARGADO	ENVIADO	ENTREGADO			
Nro	Proveedor	Fecha entrega	Pagado	Pago total	
4551	Merlion Techs	2020-06-17	500	1000	Enviar
4552	Merlion Techs	2020-06-17	0	4000	Enviar
4553	Merlion Techs	2020-06-18	500	1000	Enviar
4901	Merlion Techs	2020-06-19	1000	2000	Enviar
4902	Merlion Techs	2020-06-19	0	2000	Enviar
5151	Merlion Techs	2020-06-19	250	500	Enviar
5152	Merlion Techs	2020-06-19	1000	2000	Enviar
5401	Merlion Techs	2020-06-19	200	4000	Enviar
5402	Merlion Techs	2020-06-19	0	500	Enviar

Bonus track

Dado el siguiente JDL entregar un sistema con las misma configuración de Jhipster que el sistema anterior, de manejos de stock funcional.

```

entity Product {
    name String
}

entity ProductBucket {
    availableToSellQuantity BigDecimal,
    inChargeQuantity BigDecimal,
    brokenQuantity BigDecimal
}

relationship OneToOne {
    ProductBucket{product} to Product
}

```

Una vez autogenerado el código crear un commit con el nombre “Auto Generated Code” o un nombre similar.

El manejo de stock deberá permitir visualizar de manera sencilla el estado del producto en cada uno de sus baldes o buckets (podemos tener 10 unidades de un producto disponibles para la venta, 3 unidades encargadas y 2 unidades rotas).

El sistema debe permitir mover unidades de un balde a otro de manera sencilla.

Test Analytics

Siéntete libre de modificar el jdl si lo deseas, siempre y cuando no alteres el objetivo del ejercicio.

Introducir el siguiente JDL en un archivo con el nombre **testAnalytics-jdl.jh** y crear una aplicación Jhipster

```
entity Product {
    name String
    price BigDecimal
}

entity Sales {
    state State
    date LocalDate
}

enum State {
    IN_CHARGE, SHIPPED, DELIVERED
}

relationship ManyToOne {
    Product{product} to Sales
}
```

Correr en consola: `jhipster import-jdl testAnalytics-jdl.jh`

Una vez autogenerado el código crear un commit con el nombre “Auto Generated Code” o un nombre similar

Realice los siguientes gráficos y coloquelos en el home de la aplicación.

- Cantidad de ventas en estado DELIVERED por dia
- Ventas por dia
- Ranking 5 productos más vendidos
- Ranking 5 productos que dieron más ingresos

Los gráficos deberán realizarse con la siguiente biblioteca:

<https://recharts.org/en-US/examples>. Deberás elegir el tipo de gráfico que consideres acorde a la información que se desea mostrar.

Bonus track

Dado el siguiente JDL entregar un sistema con las misma configuración de Jhipster que el sistema anterior, que permite crear grupos de usuarios con distintos tipos de permisos.

```
entity Product {
    name String
    price BigDecimal
}

entity UserWithPerms {
    user Long
}

entity Permissions {
    perm Perms
}

enum Perms {
    TO_PRODUCT_CREATE_AND_UPDATE, TO_SHOW_PRODUCT_LIST,
    TO_PRODUCT_DETAIL
}

relationship OneToMany {
    UserWithPerms{user} to Permissions
}
```

Una vez autogenerado el código crear un commit con el nombre “Auto Generated Code” o un nombre similar

El administrador de la aplicación debe asignar permisos a un usuarios, y eso restringirá las funcionalidades que pueda realizar. Por ejemplo, un usuario con permiso TO_SHOW_PRODUCT_LIST solo podrá listar los productos, pero no crear o actualizar. Los permisos solamente deben afectar al backend.

Ayuda general

En caso de modificar el modelo (modificando el jdl) después de haberlo generado deberás correr las migraciones, para esto:

1. Correr el comando: **mvn liquibase:diff** para que cree el changelog
El archivo se crea en src/main/resources/config/liquibase/changelog
2. Este archivo modificado/creado, debe guardarse en:
src/main/resources/config/liquibase/master.xml
3. Luego hacer **mvn clean install**
4. **mvn liquibase:update**