**Prueba Java Junior y Senior**

La prueba está dividida en dos partes: **Prueba Junior** y **Prueba Senior**. El postulante deberá completar y anexar toda la información solicitada según su caso.

**Consideraciones:**

* **Para Junior**:
  + El postulante **Junior** debe completar únicamente la sección **Prueba Junior**, que consta de **4 actividades**.
  + Si el postulante considera que tiene la capacidad de desarrollar la **Prueba Senior** total o parcialmente: se evaluará con un puntaje positivo adicional.
* **Para Senior**:
  + El postulante **Senior** debe completar ambas partes: **Prueba Junior** y **Prueba Senior**.
  + Las pruebas están estructuradas de la siguiente manera:
    - **Prueba Junior:** 4 actividades
    - **Prueba Senior:** 3 actividades
    - **Total actividades para Senior:** 7 actividades

**PRUEBA JUNIOR**

A continuación, se presenta la estructura de las tablas las cuales servirán de base para desarrollar las cuatro actividades a tratar más adelante para esta prueba:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| company | Datatype |  | version | Datatype |
| **id\_company** | number | **version\_id** | number |
| **codigo\_company** | String | **app\_ id** | number |
| **name\_company** | String | **version** | string |
| **description\_company** | String | **version\_description** | String |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| version\_company | Datatype |  | application | Datatype |
| **version\_company\_id** | number |  | **app\_id** | number |
| **company\_id** | number |  | **app\_code** | String |
| **version\_id** | number |  | **app\_name** | String |
| **version\_company\_description** | string |  | **app\_description** | String |

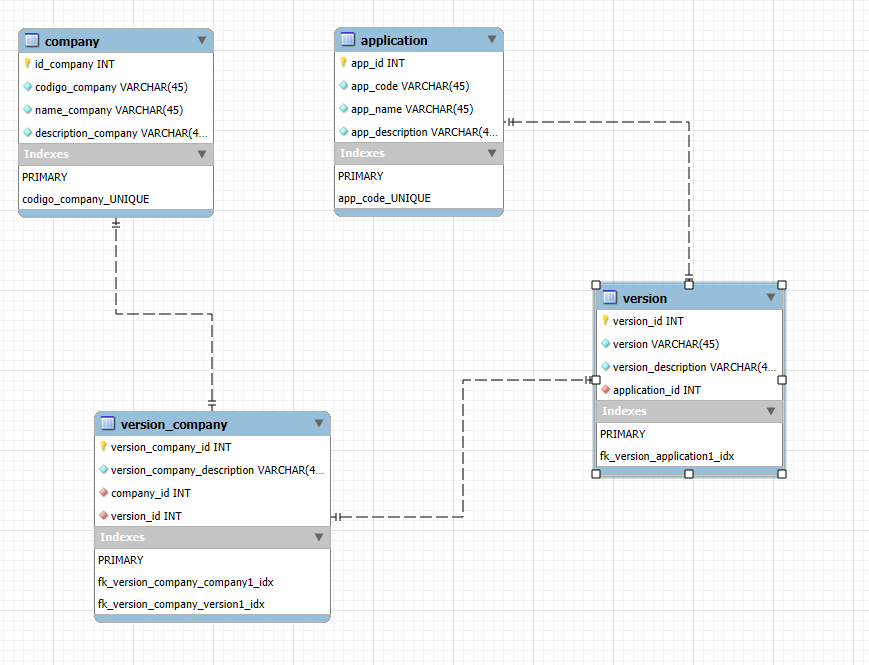
Adicional, se cuenta con las siguientes condiciones asociadas a estas tablas:

* La tabla Company cuenta con un identificador único,que no se deberá repetir (id\_company) y cuenta con un código único para cada empresa.
* La tabla application cuenta con un identificador único,que no se deberá repetir (app\_id) y cuenta con un código único para cada aplicación.
* La tabla version cuenta con un identificador único,que no se deberá repetir (version\_id) ,cabe resaltar que solo se puede tener una version por aplicación.
* La tabla version\_company cuenta con un identificador único,que no se deberá repetir (version\_company\_id) ,cabe resaltar que solo se puede tener una aplicacion por compañía.

**Actividades a desarrollar para el aspirante Junior**

Teniendo en cuenta lo anterior, usted deberá realizar las siguientes actividades:

1. Elabore un **gráfico que ilustre la relación entre las tablas presentadas anteriormente**, tomando en cuenta las reglas expuestas. El gráfico debe representar claramente la relación entre las tablas y especificar: Primary Key, Foreign Key y Unique Key de los atributos que apliquen.

Finalmente de una explicación breve del gráfico realizado.  
  
  
Respuesta:  
  
  
Fig. 1. Modelo Entidad Relación   
  
  
En la Fig. 1 observamos el MER de la aplicación a desarrollar, esta cuenta con 4 tablas, 2 tablas principales que son company y application estas tablas estarán relacionadas 1 a 1 con la tabla versión y versión\_company, evidenciamos que los indixes están definidos de acuerdo a sus requerimientos tanto las llaves primarias como las secundarias y algunos índices de lógica como lo son el UNIQUE Code que nos va a restrigir los valores de esos elementos y no podrán aparecer duplicados.

Genere un **script** para realizar la **creación** de la **tabla ’VERSION\_COMPANY’,** tenga presente **sus respectivas llaves(Primary key,Foreign key,Unique key).**Respuesta: a continuación encontraranel script con el que generaremos todas las tablas y sus respectivas relaciones

CREATE SCHEMA IF NOT EXISTS `company\_db` DEFAULT CHARACTER SET utf8 ;

USE `company\_db` ;

-- -----------------------------------------------------

-- Table `company\_db`.`company`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `company\_db`.`company` (

`id\_company` INT NOT NULL AUTO\_INCREMENT,

`codigo\_company` VARCHAR(45) NOT NULL,

`name\_company` VARCHAR(45) NOT NULL,

`description\_company` VARCHAR(45) NOT NULL,

PRIMARY KEY (`id\_company`),

UNIQUE INDEX `codigo\_company\_UNIQUE` (`codigo\_company` ASC) VISIBLE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `company\_db`.`application`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `company\_db`.`application` (

`app\_id` INT NOT NULL AUTO\_INCREMENT,

`app\_code` VARCHAR(45) NOT NULL,

`app\_name` VARCHAR(45) NOT NULL,

`app\_description` VARCHAR(45) NOT NULL,

PRIMARY KEY (`app\_id`),

UNIQUE INDEX `app\_code\_UNIQUE` (`app\_code` ASC) VISIBLE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `company\_db`.`version`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `company\_db`.`version` (

`version\_id` INT NOT NULL AUTO\_INCREMENT,

`version` VARCHAR(45) NOT NULL,

`version\_description` VARCHAR(45) NOT NULL,

`application\_id` INT NOT NULL,

PRIMARY KEY (`version\_id`),

INDEX `fk\_version\_application1\_idx` (`application\_id` ASC) VISIBLE,

CONSTRAINT `fk\_version\_application1`

FOREIGN KEY (`application\_id`)

REFERENCES `company\_db`.`application` (`app\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `company\_db`.`version\_company`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `company\_db`.`version\_company` (

`version\_company\_id` INT NOT NULL AUTO\_INCREMENT,

`version\_company\_description` VARCHAR(45) NOT NULL,

`company\_id` INT NOT NULL,

`version\_id` INT NOT NULL,

PRIMARY KEY (`version\_company\_id`),

INDEX `fk\_version\_company\_company1\_idx` (`company\_id` ASC) VISIBLE,

INDEX `fk\_version\_company\_version1\_idx` (`version\_id` ASC) VISIBLE,

CONSTRAINT `fk\_version\_company\_company1`

FOREIGN KEY (`company\_id`)

REFERENCES `company\_db`.`company` (`id\_company`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_version\_company\_version1`

FOREIGN KEY (`version\_id`)

REFERENCES `company\_db`.`version` (`version\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

1. Genere un **procedimiento** que realice la **inserción de datos en las 4 tablas anteriormente expuestas**, este procedimiento **obtendrá la información de un cursor que lo deberás llamar ‘CTemporal’**, el cual consultará toda la información de la tabla **TMP\_LLENAR\_CAMPOS.**

**Nota**: Usted deberá llenar la tabla TMP\_LLENAR\_CAMPOS ,respecto a la interpretación que haya tenido del punto 1.

Para su facilidad, se proporciona script que crea la tabla TMP\_LLENAR\_CAMPOS:

CREATE TABLE TMP\_LLENAR\_CAMPOS (id\_company NUMBER ,codigo\_company VARCHAR2,name\_company VARCHAR2,description\_company VARCHAR2,version\_id NUMBER,app\_id NUMBER,version VARCHAR2,version\_description VARCHAR2,version\_company\_id NUMBER,company\_id NUMBER,version\_id NUMBER,version\_company\_description VARCHAR2,app\_id NUMBER,app\_code VARCHAR2,app\_name VARCHAR2,app\_description VARCHAR2);  
  
**Respuesta: Se adjuntara el script correspondiente al procedimiento almacenado que se encargaría de este proceso.**

1. Realice una **API REST** donde implemente un **CRUD** para: consultar, eliminar, actualizar e insertar datos en la tabla ‘COMPANY’.

Adicional, usted deberá exponer un método de consumo ‘GET’ que reciba como parámetro el código de la compañía y devuelva un JSON con los campos: codigo\_company, name\_company, app\_name y version.  
  
**Respuesta: Se adjuntara la aplicación java desarrollada con framework Spring para la solución de este punto.**

**PRUEBA SENIOR**

El **postulante Senior deberá realizar las 4 actividades anteriores (Prueba Junior)**, y **adicional**, debe de **completar la prueba con los siguientes 3 puntos**:

1. Basándose en la información completada en el **punto 3 de la prueba Junior:**

Desarrolle el código que elimine los datos duplicados para garantizar que el listado final contenga sólo registros únicos. Además, adjunte las líneas de código que demuestren cómo se realizó la eliminación y generación de datos (Argumentar la respuesta)  
  
**Respuesta: Desde el modelo entidad relación se utilizó un índice que me restringía de tener datos de aplicaciones o compañías duplicadas.**

1. Cree una aplicación de calculadora con todas sus validaciones y con todas sus operaciones. Esta calculadora debe de permitir operaciones con decimales, y adicional, debe permitir obtener la raíz cuadrada.  
     
   Respuesta: En la aplicación adjunta encontrara un controlador llamado AlgoritmosController y podrá acceder de la siguiente manera   
     
   localhost:8080/ada/algoritmos?operacion=MULTIPLICACION&valor1=64&valor2=81  
   Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

   El contenido generado por IA puede ser incorrecto.  
   también se adjutara la collection de postman
2. Desarrolle un código que genera una o varias palabras del siguiente Array de caracteres:

cadena = [w,e,r,f,b,h,j,i,u,y,t,r,e,d,f,g,y,u,i,o,l,k,m,n,b,v,f,r,e,w,s,x,f,g,y,u,i,k,m,n,b,v,f,r,e,w,w,r,t,y,u,i,o,k,m,n,b,v,w,s,x,c,f,g,h,u,i,o,p,l,k,n,b,v,f,d,e,w,a,z,x,c,g,h,u,i,o,p,u,y,t,r,e,w,q,s,d,f,g,k,j,v,c,x]

El código debe combinar los elementos del array para formar palabras válidas. Explique brevemente el enfoque utilizado para seleccionar y unir los caracteres.  
  
Respuesta: podrá acceder desde la ruta   
  
localhost:8080/ada/algoritmos/cadena

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.