**Tabla**

|  |
| --- |
| use ccjj;  CREATE TABLE ca\_estados\_cargas(  codigo int auto\_increment not null,  nombre varchar(100) not null,  constraint codigo\_pk primary key (codigo)  ) |

**Obtener**

|  |
| --- |
| -- --------------------------------------------------------------------------------  -- Routine DDL  -- Note: comments before and after the routine body will not be stored by the server  -- --------------------------------------------------------------------------------  DELIMITER $$  CREATE DEFINER=`root`@`localhost` PROCEDURE `SP\_OBTENER\_ESTADOS`(  -- Descripción: Obtiene los estados existentes    OUT pcMensajeError VARCHAR(1000) -- Parámetro para los mensajes de error  )  BEGIN  DECLARE vcTempMensajeError VARCHAR(1000); -- Variable para anteponer los posibles mensajes de error  DECLARE EXIT HANDLER FOR SQLEXCEPTION  BEGIN    ROLLBACK;    SET vcTempMensajeError := CONCAT('Error: ', vcTempMensajeError);  SET pcMensajeError := vcTempMensajeError;    END;    -- Obtener estados  SET vcTempMensajeError := 'Error al obtener estados';    SELECT  codigo, nombre  FROM  ca\_estados\_cargas;    END |

**Insertar**

|  |
| --- |
| -- --------------------------------------------------------------------------------  -- Routine DDL  -- Note: comments before and after the routine body will not be stored by the server  -- --------------------------------------------------------------------------------  DELIMITER $$  CREATE DEFINER=`root`@`localhost` PROCEDURE `SP\_INSERTAR\_ESTADOS`(  IN pcnombre VARCHAR(50), -- Almacena el nombre de estados  OUT pcMensajeError VARCHAR(500) -- Mensaje mostrado el sistema  )  SP:BEGIN    DECLARE vcTempMensajeError VARCHAR(500) DEFAULT ''; -- Variable para almacenar posibles errores no controlados de servidor  DECLARE vnContadorEstados INT DEFAULT 0; -- Variable para determinar si el nombre de estado ya esta siendo usado  DECLARE vcMensajeErrorServidor TEXT; -- Variable para almacenar el mensaje de error del servidor    DECLARE EXIT HANDLER FOR SQLEXCEPTION  BEGIN    ROLLBACK;    GET DIAGNOSTICS CONDITION 1 vcMensajeErrorServidor = MESSAGE\_TEXT;  SET vcTempMensajeError := CONCAT('Error: ', vcTempMensajeError, ' Error servidor: ', vcMensajeErrorServidor);  SET pcMensajeError := vcTempMensajeError;  END;    -- Determinar si el nombre de estado ya esta siendo usado  SET vcTempMensajeError := 'Error al determinar la existencia del estado';  SELECT  COUNT(nombre)  INTO  vnContadorEstados  FROM  ca\_estados\_cargas  WHERE  nombre = pcnombre;      -- El nombre de estado ya esta siendo usado  IF vnContadorEstados > 0 then    SET pcMensajeError := 'Ya existe un estado con este nombre, intente con otro nombre.';  LEAVE SP;    END IF;    SET vcTempMensajeError := 'Error al crear el registro, intentelo de nuevo';  INSERT INTO ca\_estados\_cargas (nombre)  VALUES (pcnombre);  END |

**Modificar**

|  |
| --- |
| -- --------------------------------------------------------------------------------  -- Routine DDL  -- Note: comments before and after the routine body will not be stored by the server  -- --------------------------------------------------------------------------------  DELIMITER $$  CREATE DEFINER=`root`@`localhost` PROCEDURE `SP\_MODIFICAR\_ESTADOS`(  IN pnCodigo CHAR(7), -- Almacena el codigo del estado que se va a modificar  IN pcNombre VARCHAR(50), -- Almacena el nombre del estado  OUT pcMensajeError VARCHAR(500) -- Mensaje mostrado el sistema  )  SP: BEGIN  DECLARE vcTempMensajeError VARCHAR(500); -- Variable para anteponer los posibles mensajes de error  DECLARE EXIT HANDLER FOR SQLEXCEPTION  BEGIN    ROLLBACK;    SET vcTempMensajeError := CONCAT('Error: ', vcTempMensajeError);  SET pcMensajeError := vcTempMensajeError;    END;  -- Determinar que el nombre del estado ya esta registrado  SET vcTempMensajeError := 'Error al determinar la existencia del estado';    IF EXISTS  (  SELECT nombre  FROM ca\_estados\_cargas  WHERE nombre = pcNombre  AND codigo != pnCodigo  )  THEN  BEGIN    SET pcMensajeError := 'Ya existe este estado, intente de nuevo con otro nombre.';  LEAVE SP;    END;  END IF;    START TRANSACTION;    -- Actualizar el nombre de estado seleccionado  SET vcTempMensajeError := 'Error al actualizar la información del estado ';  UPDATE ca\_estados\_cargas  SET nombre = pcNombre  WHERE codigo = pnCodigo;    COMMIT;  END |

**Eliminar**

|  |
| --- |
| CREATE DEFINER=`root`@`localhost` PROCEDURE `SP\_ELIMINAR\_ESTADOS`(  IN pnCodigo INT, -- Codigo de estado  OUT pcMensajeError VARCHAR(1000) -- Parámetro para los mensajes de error  )  SP: BEGIN  DECLARE vcTempMensajeError VARCHAR(1000); -- Variable para anteponer los posibles mensajes de error  DECLARE EXIT HANDLER FOR SQLEXCEPTION  BEGIN  ROLLBACK;    SET vcTempMensajeError := CONCAT('Error: ', vcTempMensajeError);  SET pcMensajeError := vcTempMensajeError;  END;  -- Determinar si el estado tiene vinculacion con algun proyecto  SET vcTempMensajeError := 'Error al determinar si el estado tiene vinculación con alguna carga academica';    IF EXISTS  (  SELECT cod\_estado  FROM ca\_cargas\_academicass  WHERE cod\_estado = pnCodigo  )  THEN  BEGIN    SET pcMensajeError := 'Existen cargas academicas vinculadas con este estado, no puede ser borrado.';  LEAVE SP;    END;  END IF;    -- Eliminar el estado  SET vcTempMensajeError := 'Error al eliminar el estado, intentelo de nuevo';    START TRANSACTION;    DELETE FROM ca\_cargas\_academicas  WHERE codigo = pnCodigo;    COMMIT;  end |