



Universidad Nacional Mayor de San Marcos
Facultad de Ingeniería de Sistemas e Informática

Informe final

Descripción del sistema con enfoque en Base de Datos

Versión 2.1

Base de Datos

Prof. Luis Soto Soto

Equipo 1

Alessandro Jesús Torres Tamariz - 22200049

Ángel Javier Dioses Bellota – 22200209

Felipe Santiago Triveño Daza – 22200051

José Antonio Villanueva Inés - 22200116

Said William Najarro Llacza – 22200031

Semestre 2024-I

Índice

Introducción	3
Objetivos	4
Sobre la aplicación	5
Modelo lógico de la Base de Datos.....	6
Tablas	7
Diagramas	19
Diagrama de Casos de Uso.....	19
Diagramas de secuencia.....	20
Adicionales.....	96
Implementación de la interfaz CRUD.....	96
Otros objetos de la base de datos	97

Introducción

Este documento ha sido creado con la finalidad de documentar y entender cómo funciona la aplicación presentada en el curso sin ser alguien experimentado en la gestión de proyectos y base de datos.

En este proyecto se mostrará cómo funciona la interacción del usuario con la aplicación, así como las consultas e instrucciones que se hacen para la misma con la base de datos en Oracle; además de hacer un aporte de otras funcionalidades y observaciones que hemos encontrado.

Objetivos

- Identificar las relaciones de la base de datos
- Entender cómo funciona la aplicación en la creación de partidas
- Documentar los casos de uso en la creación del presupuesto interno
- Implementar la creación de un presupuesto externo
- Identificar otros elementos de la base de datos como funciones, procedimientos almacenados, triggers, entre otros.

Sobre la aplicación

Esta aplicación es un administrador de proyectos implementado en Java e interactuando con la una base de datos hecha en Oracle.

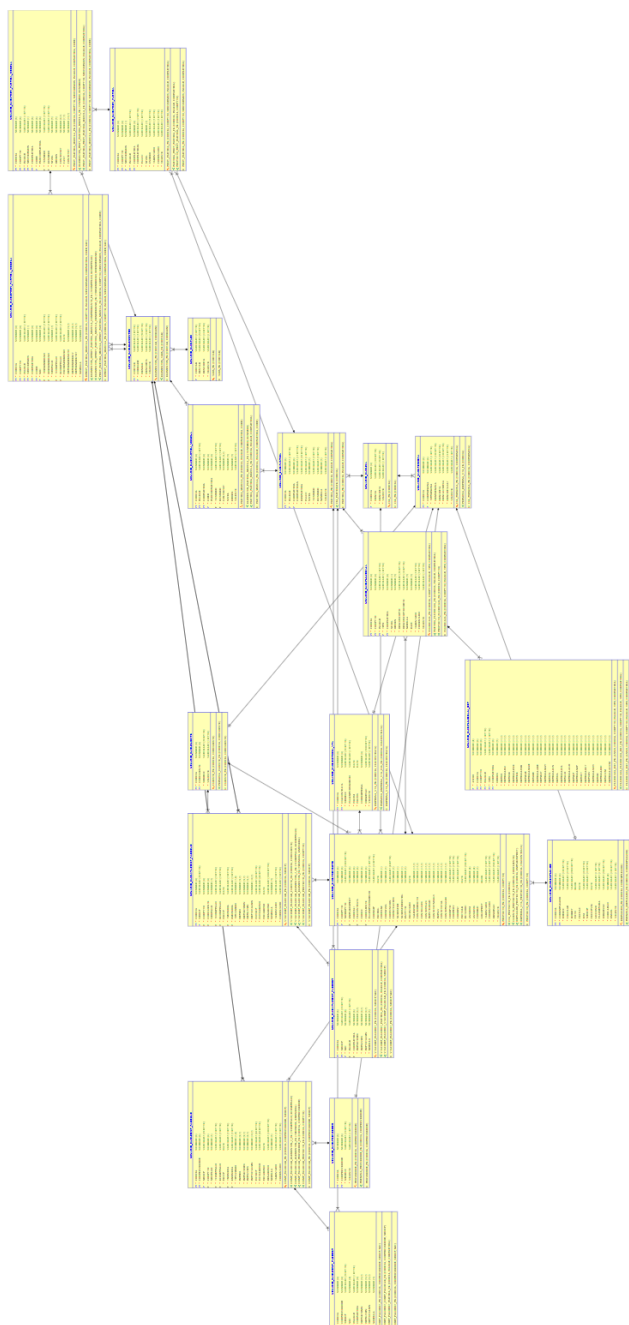
Esta se encarga de crear partidas que representan los elementos de un presupuesto que están divididos en gastos e ingresos.

En esta hemos ahondado este informe en la creación y gestión de las partidas, así también como la conexión a la base de datos para su obtención y manipulación.

Modelo lógico de la Base de Datos

Obtenemos este modelo lógico del gestor de base de datos de Oracle, SQL Developer. Nos da una vista general para poder comprender cómo están relacionados los datos y hacer seguimiento de los métodos que tiene la aplicación para poder acceder a ellos.

Vista general



Tablas

COMP_PAGODET

Esta tabla se usa en la opción Registrar las Compras, la tabla tiene algunos atributos que guardan los datos como Código Cia, Código de Proveedor, Código de Partida, señalar si es Ingreso o Egreso, Impuesto neto, Impuesto IGV y el Impuesto Total.

USUARIO_SAID.COMP_PAGODET	
PF * CODCIA	NUMBER (6)
PF * CODPROVEEDOR	NUMBER (6)
PF * NROCP	VARCHAR2 (20 BYTE)
P * SEC	NUMBER (4)
F * INGEGR	VARCHAR2 (1 BYTE)
F * CODPARTIDA	NUMBER (6)
* IMPNETOMN	NUMBER (9,2)
* IMPIGVMN	NUMBER (9,2)
* IMPTOTALMN	NUMBER (9,2)
* SEMILLA	NUMBER (5)
COMP_PAGODET_PK (CODCIA, CODPROVEEDOR, NROCP, SEC)	
COMP_PAGODET_COMP_PAGOCAB_FK (CODCIA, CODPROVEEDOR, NROCP)	
COMP_PAGODET_PARTIDA_FK (CODCIA, INGEGR, CODPARTIDA)	
U COMP_PAGODET_PK (CODCIA, CODPROVEEDOR, NROCP, SEC)	

COMP_PAGOCAB

Esta tabla se usa en la opción Registrar las Compras, la tabla tiene algunos atributos que guardan los datos como Código Cia, Código de proveedor, Código de Proyecto, Tipo de Pago, Tipo de Moneda, Impuesto neto, Impuesto IGV, Impuesto Total y Fecha de Abono.

USUARIO_SAID.COMP_PAGOCAB		
PF *	CODCIA	NUMBER (6)
PF *	CODPROVEEDOR	NUMBER (6)
P *	NROCP	VARCHAR2 (20 BYTE)
F *	CODPYTO	NUMBER (6)
	* NROPAGO	NUMBER (3)
F *	TCOMPPAGO	VARCHAR2 (3 BYTE)
F *	ECOMPPAGO	VARCHAR2 (3 BYTE)
	* FECCP	DATE
F *	TMONEDA	VARCHAR2 (3 BYTE)
F *	EMONEDA	VARCHAR2 (3 BYTE)
	* TIPCAMBIO	NUMBER (7,4)
	* IMPMO	NUMBER (9,2)
	* IMPNETOMN	NUMBER (9,2)
	* IMPIGVMN	NUMBER (9,2)
	* IMPTOTALMN	NUMBER (10,2)
	* FOTOCP	VARCHAR2 (60 BYTE)
	* FOTOABONO	VARCHAR2 (60 BYTE)
	* FECABONO	DATE
	* DESABONO	VARCHAR2 (1000 BYTE)
	* SEMILLA	NUMBER (5)
	* TABESTADO	VARCHAR2 (3 BYTE)
	* CODESTADO	VARCHAR2 (3 BYTE)
COMP_PAGOCAB_PK (CODCIA, CODPROVEEDOR, NROCP)		
COMP_PAGOCAB_ELEMENTOS_2_FK (TCOMPPAGO, ECOMPPAGO)		
COMP_PAGOCAB_ELEMENTOS_FK (TMONEDA, EMONEDA)		
COMP_PAGOCAB_PROVEEDOR_FK (CODCIA, CODPROVEEDOR)		
COMP_PAGOCAB_PROYECTO_FK (CODCIA, CODPYTO)		
U COMP_PAGOCAB_PK (CODCIA, CODPROVEEDOR, NROCP)		

PROVEEDOR

La tabla Proveedor almacena los datos de los proveedores que se registran en la aplicación, los atributos de la tabla son el Codigo de Cia, Cod de proveedor, Numero de RUC y verificar si está vigente o no.

USUARIO_SAID.PROVEEDOR		
PF *	CODCIA	NUMBER (6)
PF *	CODPROVEEDOR	NUMBER (6)
	* NRORUC	VARCHAR2 (20 BYTE)
	* VIGENTE	VARCHAR2 (1 BYTE)
PROVEEDOR_PK (CODCIA, CODPROVEEDOR)		
PERSONA_PROVEEDOR_FK (CODCIA, CODPROVEEDOR)		
U PROVEEDOR_PK (CODCIA, CODPROVEEDOR)		

VTACOMP_PAGODET

Esta tabla se usa en la opción Registrar las Ventas, algunos de los atributos que tiene esta tabla son Codigo de Cia, Codigo de Partida, señalar si es Ingreso o Egreso, Impuesto neto, Impuesto IGV y el Impuesto Total.

USUARIO_SAID.VTACOMP_PAGODET		
PF *	CODCIA	NUMBER (6)
PF *	NROCP	VARCHAR2 (20 BYTE)
P *	SEC	NUMBER (4)
F *	INGEGR	VARCHAR2 (1 BYTE)
F *	CODPARTIDA	NUMBER (6)
	* IMPNETOMN	NUMBER (9,2)
	* IMPIGVMN	NUMBER (9,2)
	* IMPTOTALMN	NUMBER (9,2)
	* SEMILLA	NUMBER (5)
🔍 VTACOMP_PAGODET_PK (CODCIA, NROCP, SEC)		
🔗 VTACOMP_PAGODET_PARTIDA_FK (CODCIA, INGEGR, CODPARTIDA)		
🔗 VTACOMP_PAGODET_VTACOMP_PAGOCAB_FK (CODCIA, NROCP)		
U VTACOMP_PAGODET_PK (CODCIA, NROCP, SEC)		

VTACOMP_PAGOCAB

Esta tabla se usa en la opción Registrar las Ventas, la tabla tiene algunos atributos que guardan los datos como Código Cia, Código de Cliente, Código de Proyecto, Tipo de Pago, Tipo de Moneda, Impuesto neto, Impuesto IGV, Impuesto Total y Fecha de Abono.

USUARIO_SAID.VTACOMP_PAGOCAB		
PF *	CODCIA	NUMBER (6)
P *	NROCP	VARCHAR2 (20 BYTE)
F *	CODPYTO	NUMBER (6)
F *	CODCLIENTE	NUMBER (6)
	* NROPAGO	NUMBER (3)
F *	TCOMPPAGO	VARCHAR2 (3 BYTE)
F *	ECOMPPAGO	VARCHAR2 (3 BYTE)
	* FECCP	DATE
F *	TMONEDA	VARCHAR2 (3 BYTE)
F *	EMONEDA	VARCHAR2 (3 BYTE)
	* TIPCAMBIO	NUMBER (7,4)
	* IMPMO	NUMBER (9,2)
	* IMPNETOMN	NUMBER (9,2)
	* IMPIGVMN	NUMBER (9,2)
	* IMPTOTALMN	NUMBER (10,2)
	* FOTOCP	VARCHAR2 (60 BYTE)
	* FOTOABONO	VARCHAR2 (60 BYTE)
	* FECABONO	DATE
	* DESABONO	VARCHAR2 (1000 BYTE)
	* SEMILLA	NUMBER (5)
	* TABESTADO	VARCHAR2 (3 BYTE)
	* CODESTADO	VARCHAR2 (3 BYTE)
🔍 VTACOMP_PAGOCAB_PK (CODCIA, NROCP)		
🔗 VTACOMP_PAGOCAB_CLIENTE_FK (CODCIA, CODCLIENTE)		
🔗 VTACOMP_PAGOCAB_ELEMENTOS_2_FK (TCOMPPAGO, ECOMPPAGO)		
🔗 VTACOMP_PAGOCAB_ELEMENTOS_FK (TMONEDA, EMONEDA)		
🔗 VTACOMP_PAGOCAB_PROYECTO_FK (CODCIA, CODPYTO)		
U VTACOMP_PAGOCAB_PK (CODCIA, NROCP)		

PROYECTO

La tabla Proyecto es una de las principales dentro de la base de datos, algunos de los atributos que contiene son el Código de Cia, Código de Proyecto, Nombre de Proyecto, Empleado Jefe del Proyecto, Código de Cliente, Código de Departamento, Código de Provincia, Código de Distrito, Año de inicio, Año de fin y verificar si está vigente o no.

USUARIO_SAID.PROYECTO		
PF *	CODCIA	NUMBER (6)
P *	CODPYTO	NUMBER (6)
	NOMBPYTO	VARCHAR2 (1000 BYTE)
F *	EMPLJEFEPROY	NUMBER (6)
	CODCIA1	NUMBER (6)
F *	CIACONTRATA	NUMBER (6)
	CODCC	NUMBER (6)
F *	CODCLIENTE	NUMBER (6)
	FLGEMPCONSORCIO	VARCHAR2 (1 BYTE)
	CODSNIP	VARCHAR2 (10 BYTE)
	FECREG	DATE
	CODFASE	NUMBER (1)
	CODNIVEL	NUMBER (2)
	CODFUNCION	VARCHAR2 (4 BYTE)
	CODSITUACION	NUMBER (2)
	NUMINFOR	NUMBER (1)
	NUMINFORENTRG	NUMBER (1)
	ESTPYTO	NUMBER (2)
	FECESTADO	DATE
	VALREFER	NUMBER (12,2)
	COSTODIRECTO	NUMBER (12,2)
	COSTOGGEN	NUMBER (12,2)
	COSTOFINAN	NUMBER (12,2)
	IMPUTILIDAD	NUMBER (12,2)
	COSTOTOTSINIGV	NUMBER (12,2)
	IMPIGV	NUMBER (12,2)
	COSTOTOTAL	NUMBER (12,2)
	COSTOPENALID	NUMBER (12,2)
	CODDPTO	VARCHAR2 (2 BYTE)
	CODPROV	VARCHAR2 (2 BYTE)
	CODDIST	VARCHAR2 (2 BYTE)
	FECVIAB	DATE
	RUTADOC	VARCHAR2 (300 BYTE)
	ANNOINI	NUMBER (4)
	ANNOFIN	NUMBER (4)
	CODOBJC	NUMBER (2)
	LOGOPROY	BLOB
	TABESTADO	VARCHAR2 (3 BYTE)
	CODESTADO	VARCHAR2 (3 BYTE)
	OBSERVAC	VARCHAR2 (500 BYTE)
	VIGENTE	VARCHAR2 (1 BYTE)
PROYECTO_PK (CODCIA, CODPYTO)		
CIA_PROYECTO_FK (CODCIA)		
CLIENTE_PROYECTO_FK (CODCIA, CODCLIENTE)		
EMPLEADO_PROYECTO_FK (CODCIA, EEMPLJEFEPROY)		
EMPRESA_VTA_PROYECTO_FK (CODCIA, CIACONTRATA)		
U PROYECTO_PK (CODCIA, CODPYTO)		

EMPLEADO

La tabla Empleado tiene una relación de herencia con la tabla Persona, dentro de esta tabla se definen los atributos de un nuevo empleado, algunos atributos sonCodigo de Cia, Codigo de Empleado, Celular, Hobby, DNI, Email, Foto y verificar si está vigente o no.

USUARIO_SAID.EMPLEADO	
PF * CODCIA	NUMBER (6)
PF * CODEMPLEADO	NUMBER (6)
* DIRECC	VARCHAR2 (100 BYTE)
* CELULAR	VARCHAR2 (33 BYTE)
* HOBBY	VARCHAR2 (2000 BYTE)
FOTO	BLOB
* FECNAC	DATE
* DNI	VARCHAR2 (20 BYTE)
* NROCIP	VARCHAR2 (10 BYTE)
* FECCIPVIG	DATE
* LICCOND	VARCHAR2 (1 BYTE)
* FLGEMPLIEA	VARCHAR2 (1 BYTE)
* OBSERVAC	VARCHAR2 (300 BYTE)
* CODCARGO	NUMBER (4)
* EMAIL	VARCHAR2 (100 BYTE)
* VIGENTE	VARCHAR2 (1 BYTE)
EMPLEADO_PK (CODCIA, CODEMPLEADO)	
PERSONA_EMPLEADO_FK (CODCIA, CODEMPLEADO)	
U EMPLADO_PK (CODCIA, CODEMPLEADO)	

CLIENTE

La tabla Cliente tiene una relación de herencia con la tabla Persona, dentro de esta tabla se definen los atributos de un nuevo cliente, los atributos son Codigo de Cia, Codigo de Cliente, Numero de Ruc y verificar si está vigente o no.

USUARIO_SAID.CLIENTE	
PF * CODCIA	NUMBER (6)
PF * CODCLIENTE	NUMBER (6)
* NRORUC	VARCHAR2 (20 BYTE)
* VIGENTE	VARCHAR2 (1 BYTE)
CLIENTE_PK (CODCIA, CODCLIENTE)	
PERSONA_CLIENTE_FK (CODCIA, CODCLIENTE)	
U CLIENTE_PK (CODCIA, CODCLIENTE)	

EMPRESA_VTA

La tabla Empresa_Vta tiene una relación de herencia con la tabla Persona, dentro de esta tabla se definen los atributos de una nueva empresa, algunos atributos son Codigo de Cia, Cia Contratada, Numero de RUC, Consorcio, Fecho de Inicio, Fecha de Fin, Codigo de Empresa y verificar si está vigente o no.

USUARIO_SAID.EMPRESA_VTA		
PF *	CODCIA	NUMBER (6)
PF *	CIACONTRATA	NUMBER (6)
*	NRORUC	VARCHAR2 (20 BYTE)
*	FLGEMPCONSorcio	VARCHAR2 (1 BYTE)
*	FECINI	DATE
*	FECFIN	DATE
	CODEMPRESA	NUMBER (6)
*	OBSERVAC	VARCHAR2 (2000 BYTE)
*	VIGENTE	VARCHAR2 (1 BYTE)
EMPRESA_VTA_PK (CODCIA, CIACONTRATA)		
PERSONA_EMPRESA_VTA_CIA_FK (CODCIA, CIACONTRATA)		
U EMPRESA_VTA_PK (CODCIA, CIACONTRATA)		

FLUJOCAJA_DET

Esta tabla se usa en la opción Flujo de Caja aquí se almacenan todos los detalles, la tabla tiene algunos atributos que guardan los datos como Año, Codigo de Cia, Codigo de Proyecto, Codigo de Partida, señalar si es Ingreso o Egreso y los detalles de todos los meses del año.

USUARIO_SAID.FLUJOCAJA_DET		
P *	ANNO	NUMBER (4)
PF *	CODCIA	NUMBER (6)
PF *	CODPYTO	NUMBER (6)
PF *	INGEGR	VARCHAR2 (1 BYTE)
PF *	TIPO	VARCHAR2 (1 BYTE)
PF *	CODPARTIDA	NUMBER (6)
*	ORDEN	NUMBER (5)
*	IMPINI	NUMBER (12,2)
*	IMPREALINI	NUMBER (12,2)
*	IMPENE	NUMBER (12,2)
*	IMPREALENE	NUMBER (12,2)
*	IMPFEB	NUMBER (12,2)
*	IMPREALFEB	NUMBER (12,2)
*	IMPMAR	NUMBER (12,2)
*	IMPREALMAR	NUMBER (12,2)
*	IMPABR	NUMBER (12,2)
*	IMPREALABR	NUMBER (12,2)
*	IMPMAY	NUMBER (12,2)
*	IMPREALMAY	NUMBER (12,2)
*	IMPJUN	NUMBER (12,2)
*	IMPREALJUN	NUMBER (12,2)
*	IMPJUL	NUMBER (12,2)
*	IMPREALJUL	NUMBER (12,2)
*	IMPAGO	NUMBER (12,2)
*	IMPREALAGO	NUMBER (12,2)
*	IMPSEP	NUMBER (12,2)
*	IMPREALSEP	NUMBER (12,2)
*	IMPOCT	NUMBER (12,2)
*	IMPREALOCT	NUMBER (12,2)
*	IMPNOV	NUMBER (12,2)
*	IMPREALNOV	NUMBER (12,2)
*	IMPDIC	NUMBER (12,2)
*	IMPREALDIC	NUMBER (12,2)
*	IMPACUM	NUMBER (12,2)
*	IMPREALACUM	NUMBER (12,2)
FLUJOCAJA_DET_PK (ANNO, CODCIA, CODPYTO, INGEGR, TIPO, CODPARTIDA)		
FLUJOCAJA_FLUJOCAJA_DET_FK (CODCIA, CODPYTO, INGEGR, TIPO, CODPARTIDA)		
U FLUJOCAJA_DET_PK (ANNO, CODCIA, CODPYTO, INGEGR, TIPO, CODPARTIDA)		

FLUJOCAJA

Esta tabla se usa en la opción Flujo de Caja, la tabla tiene algunos atributos como Codigo de Cia, Codigo de Proyecto, Codigo de Partida, Descripcion de Concepto, señalar si es Ingreso o Egreso y verificar si está vigente o no.

USUARIO_SAID.FLUJOCAJA		
PF *	CODCIA	NUMBER (6)
PF *	CODPYTO	NUMBER (6)
PF *	INGEGR	VARCHAR2 (1 BYTE)
P *	TIPO	VARCHAR2 (1 BYTE)
PF *	CODPARTIDA	NUMBER (6)
	* NIVEL	NUMBER (1)
	* ORDEN	NUMBER (5)
	* DESCONCEPTO	VARCHAR2 (30 BYTE)
	* DESCONCEPTOCORTO	VARCHAR2 (10 BYTE)
	* SEMILLA	NUMBER (5)
	* RAIZ	NUMBER (5)
	* TABESTADO	VARCHAR2 (3 BYTE)
	* CODESTADO	VARCHAR2 (3 BYTE)
	* VIGENTE	VARCHAR2 (1 BYTE)
FLUJOCAJA_PK (CODCIA, CODPYTO, INGEGR, TIPO, CODPARTIDA)		
PARTIDA_FLUJOCAJA_FK (CODCIA, INGEGR, CODPARTIDA)		
PROYECTO_FLUJOCAJA_FK (CODCIA, CODPYTO)		
U FLUJOCAJA_PK (CODCIA, CODPYTO, INGEGR, TIPO, CODPARTIDA)		

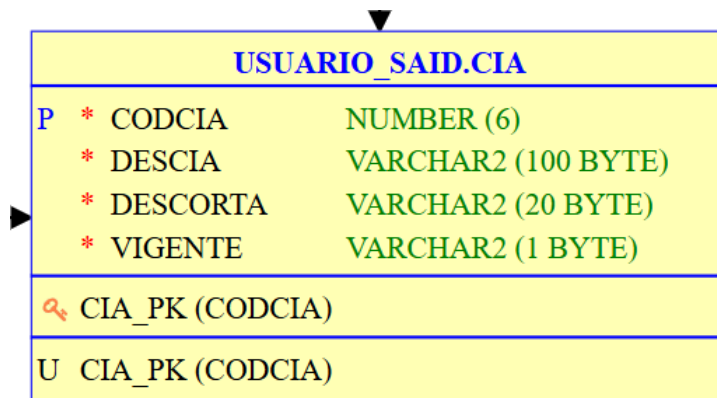
PERSONA

La tabla Persona tiene una relación de herencia de la cual se desprenden varias otras tablas, esta tabla tiene algunos atributos como Codigo de Cia, Codigo de Persona, Tipo de Persona, Descripcion de Persona y verificar si está vigente o no.

USUARIO_SAID.PERSONA		
PF *	CODCIA	NUMBER (6)
P *	CODPERSONA	NUMBER (6)
	* TIPPERSONA	VARCHAR2 (1 BYTE)
	* DESPERSONA	VARCHAR2 (100 BYTE)
	* DESCORTA	VARCHAR2 (30 BYTE)
	* DESCALTERNA	VARCHAR2 (100 BYTE)
	* DESCORTAALT	VARCHAR2 (10 BYTE)
	* VIGENTE	VARCHAR2 (1 BYTE)
CIA_PERSONA_PK (CODCIA, CODPERSONA)		
PERSONA_EMPRESA_VTA_FK (CODCIA)		
U CIA_PERSONA_PK (CODCIA, CODPERSONA)		

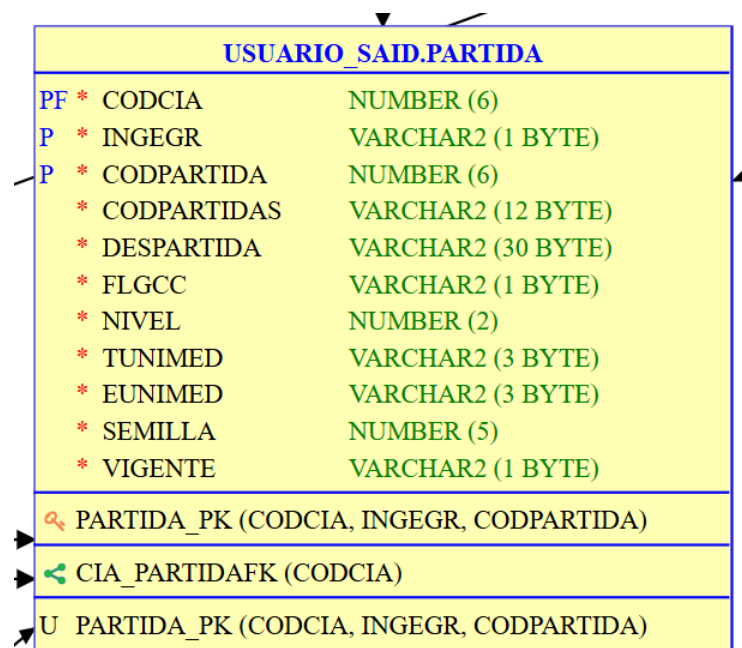
CIA

Esta tabla almacena los atributos de las nuevas compañías, las cuales son Código de Cia, Descripción de la Cia, Descripción Corta y verificar si está vigente o no.



PARTIDA

La tabla Partida tiene algunos atributos como Código de Cia, Código de Partida, Descripción de Partida, Tipo de Unidad de Medida, Elemento de Unidad de Medida y verificar si está vigente o no.



PARTIDA_MEZCLA

Esta tabla almacena atributos de las partidas como Código de Cia, Código de Partida, Código Padre Partida y **verificar** si está vigente o no.

USUARIO_SAID.PARTIDA_MEZCLA		
PF *	CODCIA	NUMBER (6)
PF *	INGEGR	VARCHAR2 (1 BYTE)
PF *	CODPARTIDA	NUMBER (6)
P *	CORR	NUMBER (6)
	* PADCODPARTIDA	NUMBER (6)
F *	TUNIMED	VARCHAR2 (3 BYTE)
F *	EUNIMED	VARCHAR2 (3 BYTE)
	* COSTOUNIT	NUMBER (9,2)
	* NIVEL	NUMBER (5)
	* ORDEN	NUMBER (5)
	* VIGENTE	VARCHAR2 (1 BYTE)
PARTIDA_MEZCLA_PK (CODCIA, INGEGR, CODPARTIDA, CORR)		
ELEMENTOS_PARTIDA_MEZCLA_FK (TUNIMED, EUNIMED)		
PARTIDA_PARTIDA_MEZCLA_FK (CODCIA, INGEGR, CODPARTIDA)		
U PARTIDA_MEZCLA_PK (CODCIA, INGEGR, CODPARTIDA, CORR)		

TABS

La tabla almacena atributos como Codigo de Tab, Denominación de Tab, Denominación Corta y verificar si está vigente o no.

USUARIO_SAID.TABS		
P *	CODTAB	VARCHAR2 (3 BYTE)
	* DENTAB	VARCHAR2 (50 BYTE)
	* DENCORTA	VARCHAR2 (10 BYTE)
	* VIGENTE	VARCHAR2 (1 BYTE)
TABS_PK (CODTAB)		
U TABS_PK (CODTAB)		

ELEMENTOS

La tabla Elementos hereda de la tabla TABS y tiene algunos atributos como Codigo de Tabs, Codigo Elemento, Denominación de Elemento Denominación Corta y verificar si está vigente o no

USUARIO_SAID.ELEMENTOS		
PF *	CODTAB	VARCHAR2 (3 BYTE)
P *	CODELEM	VARCHAR2 (3 BYTE)
	* DENELE	VARCHAR2 (50 BYTE)
	* DENCORTA	VARCHAR2 (10 BYTE)
	* VIGENTE	VARCHAR2 (1 BYTE)
ELEMENTOS_PK (CODTAB, CODELEM)		
ELEMENTOS_TABS_FK (CODTAB)		
U ELEMENTOS_PK (CODTAB, CODELEM)		

DPROY_PARTIDA_MEZCLA

La tabla almacena algunos atributos de la partida comoCodigo de Cia, Codigo de Proyecto, Numero de Version, Codigo de Partida, Tipo de Desembolso, Elementos de Desembolso, Numero de Pago, Tipo Compra Pago, Elemento Compra Pago, Fecha de Desembolso y señalar si es Ingreso o Egreso.

USUARIO_SAID.DPROY_PARTIDA_MEZCLA	
PF * CODCIA	NUMBER (6)
PF * CODPYTO	NUMBER (6)
PF * INGEGR	VARCHAR2 (1 BYTE)
PF * NROVERSION	NUMBER (1)
PF * CODPARTIDA	NUMBER (6)
PF * CORR	NUMBER (6)
P * SEC	NUMBER (4)
F * TDESEMBOLSO	VARCHAR2 (3 BYTE)
F * EDESEMBOLSO	VARCHAR2 (3 BYTE)
* NROPAGO	NUMBER (2)
F * TCOMPAGO	VARCHAR2 (3 BYTE)
F * ECOMPAGO	VARCHAR2 (3 BYTE)
* FECDESEMBOLSO	DATE
* IMPDESEMBNETO	NUMBER (9,2)
* IMPDESEMBIGV	NUMBER (8,2)
* IMPDESEMBTOT	NUMBER (9,2)
* SEMILLA	NUMBER (5)
DPROY_PARTIDA_MEZCLA_PK (CODCIA, CODPYTO, INGEGR, NROVERSION, CODPARTIDA, CORR, SEC)	
ELEMENTOS_DPROY_PARTIDA_MEZCLA_COMPROBANTE_FK (TCOMPAGO, ECOMPAGO)	
ELEMENTOS_DPROY_PARTIDA_MEZCLA_DESEMBOLSO_FK (TDESEMBOLSO, EDESEMBOLSO)	
PROY_PARTIDA_MEZCLA_DPROY_PARTIDA_MEZCLA_FK (CODCIA, CODPYTO, NROVERSION, INGEGR, CODPARTIDA, CORR)	
U DPROY_PARTIDA_MEZCLA_PK (CODCIA, CODPYTO, INGEGR, NROVERSION, CODPARTIDA, CORR, SEC)	




PROY_PARTIDA_MEZCLA

La tabla almacena algunos atributos comoCodigo de Cia, Codigo de Proyecto, Numero de Version, Codigo de Partida, Tipo de Unidad de Medida, Elemento de Unidad de Medida, Nivel, Costo Unitario, Cantidad, Costo Total.

USUARIO_SAID.PROY_PARTIDA_MEZCLA	
PF * CODCIA	NUMBER (6)
PF * CODPYTO	NUMBER (6)
PF * INGEGR	VARCHAR2 (1 BYTE)
PF * NROVERSION	NUMBER (1)
PF * CODPARTIDA	NUMBER (6)
P * CORR	NUMBER (6)
* PADCODPARTIDA	NUMBER (6)
F * TUNIMED	VARCHAR2 (3 BYTE)
F * EUNIMED	VARCHAR2 (3 BYTE)
* NIVEL	NUMBER (5)
* ORDEN	NUMBER (5)
* COSTUNIT	NUMBER (9,2)
* CANT	NUMBER (7,3)
* COSTOTOT	NUMBER (10,2)
PROY_PARTIDA_MEZCLA_PK (CODCIA, CODPYTO, NROVERSION, INGEGR, CODPARTIDA, CORR)	
ELEMENTOS_PROY_PARTIDA_MEZCLA_FK (TUNIMED, EUNIMED)	
PROY_PARTIDA_PROY_PARTIDA_MEZCLA_FK (CODCIA, CODPYTO, NROVERSION, INGEGR, CODPARTIDA)	
U PROY_PARTIDA_MEZCLA_PK (CODCIA, CODPYTO, NROVERSION, INGEGR, CODPARTIDA, CORR)	

PROY_PARTIDA

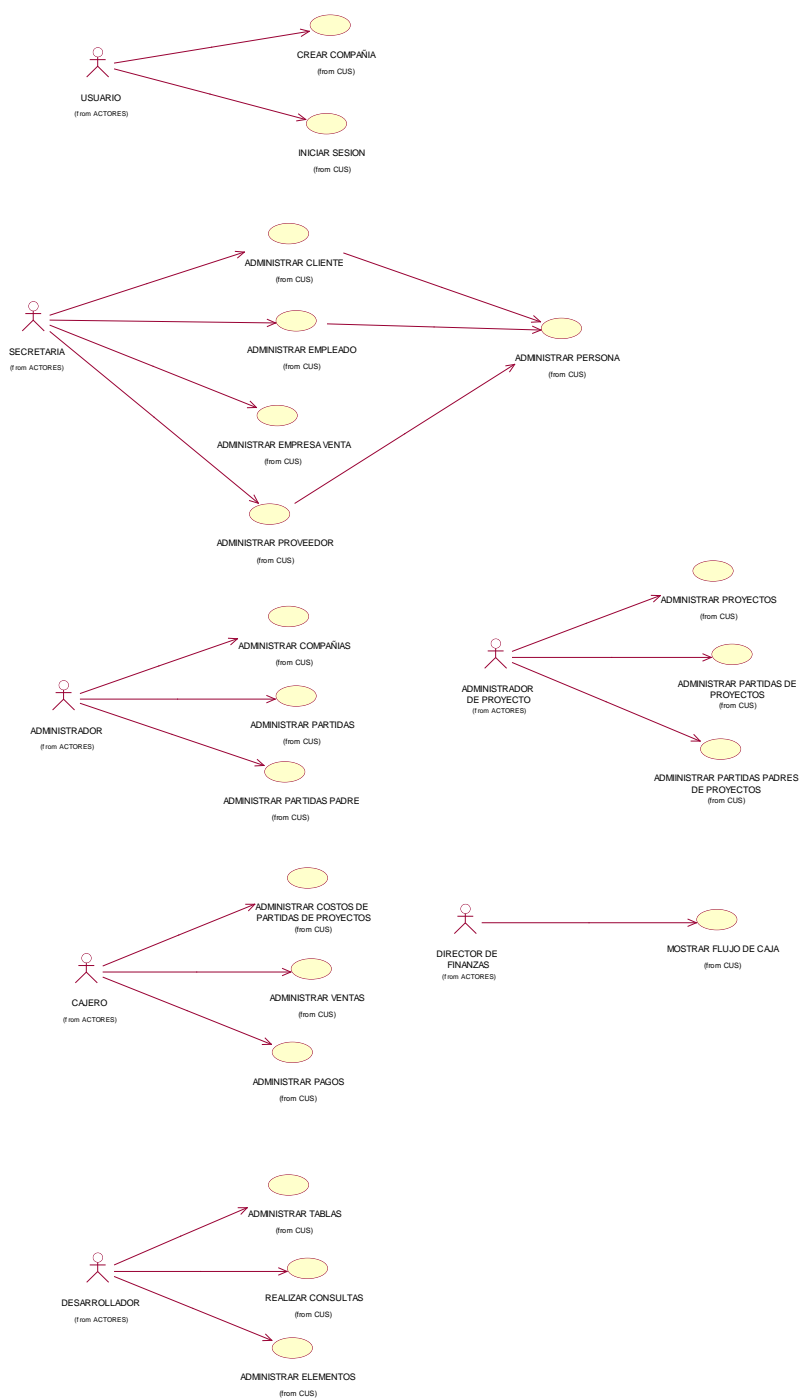
La tabla almacena algunos atributos comoCodigo de Cia, Código de Proyecto, Numero de Version, Codigo de Partida, Codigo de Partidas, Unidad de Medida, señalar si es Ingreso o Egreso y verificar si está vigente o no.

USUARIO_SAID.PROY_PARTIDA	
PF *	CODCIA NUMBER (6)
PF *	CODPYTO NUMBER (6)
P *	NROVERSION NUMBER (1)
PF *	INGEGR VARCHAR2 (1 BYTE)
PF *	CODPARTIDA NUMBER (6)
*	CODPARTIDAS VARCHAR2 (12 BYTE)
*	FLGCC VARCHAR2 (1 BYTE)
*	NIVEL NUMBER (2)
*	UNIMED VARCHAR2 (5 BYTE)
*	TABESTADO VARCHAR2 (3 BYTE)
*	CODESTADO VARCHAR2 (3 BYTE)
*	VIGENTE VARCHAR2 (1 BYTE)
	PROY_PARTIDA_PK (CODCIA, CODPYTO, NROVERSION, INGEGR, CODPARTIDA)
	PARTIDA_PROY_PARTIDA_FK (CODCIA, INGEGR, CODPARTIDA)
	PROYECTO_PROY_PARTIDA_FK (CODCIA, CODPYTO)
U	PROY_PARTIDA_PK (CODCIA, CODPYTO, NROVERSION, INGEGR, CODPARTIDA)

Diagramas

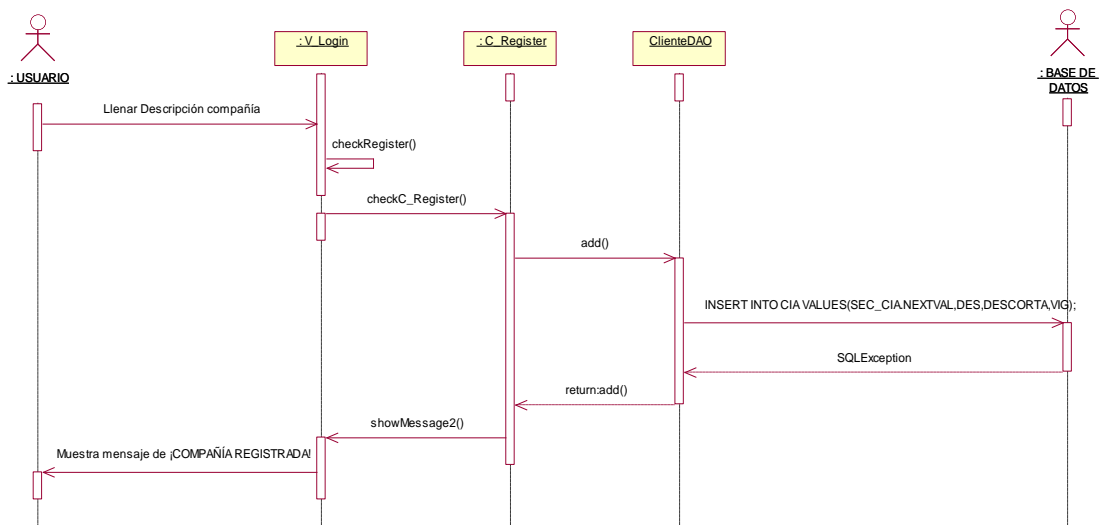
Diagrama de Casos de Uso

Este diagrama muestra los Casos de Uso del Sistema con un posible usuario.



Diagramas de secuencia

Crear compañía



Pasos

- El usuario completa con datos en la interfaz que se le muestra (Descripción Compañía y Descripción corta)
- El usuario hace clic en el botón Registrar
- La clase V_Login muestra la interfaz de usuario para que el Usuario ingrese el nombre de la compañía junto a una descripción corta.
- La clase C_Register obtiene la Descripción de la Compañía y Descripción corta con checkC_Register().
- La clase CiaDAO llama al procedimiento almacenado, a través del método add(), INSERTAR_CIA(?, ?, ?, ?) que en la base de datos está como la consulta `INSERT INTO CIA VALUES (SEC_CIA.NEXTVAL, DES, DESCORTA, VIG)`.
- Luego, según se haya ejecutado el programa, la base de datos vuelve en una excepción, SQLException, lo que se convertirá en un 1 o 0.

- Finalmente, este 1 o 0 representa un mensaje en la vista para el usuario que será ¡COMPañIA REGISTRADA! Si la consulta fue exitosa.

Consultas e instrucciones

Procedimiento almacenado INSERTAR_CIA

```

/*-----*/

create or replace NONEDITIONABLE PROCEDURE INSERTAR_CIA(

    CODC IN CIA.CODCIA%TYPE,

    DES IN CIA.DESCIA%TYPE,

    DESCORTA IN CIA.DESCORTA%TYPE,

    VIG IN CIA.VIGENTE%TYPE)

IS

BEGIN

INSERT INTO CIA (codcia, descia, descorta, vigente)

VALUES (SEC_CIA.NEXTVAL, DES, DESCORTA, VIG);

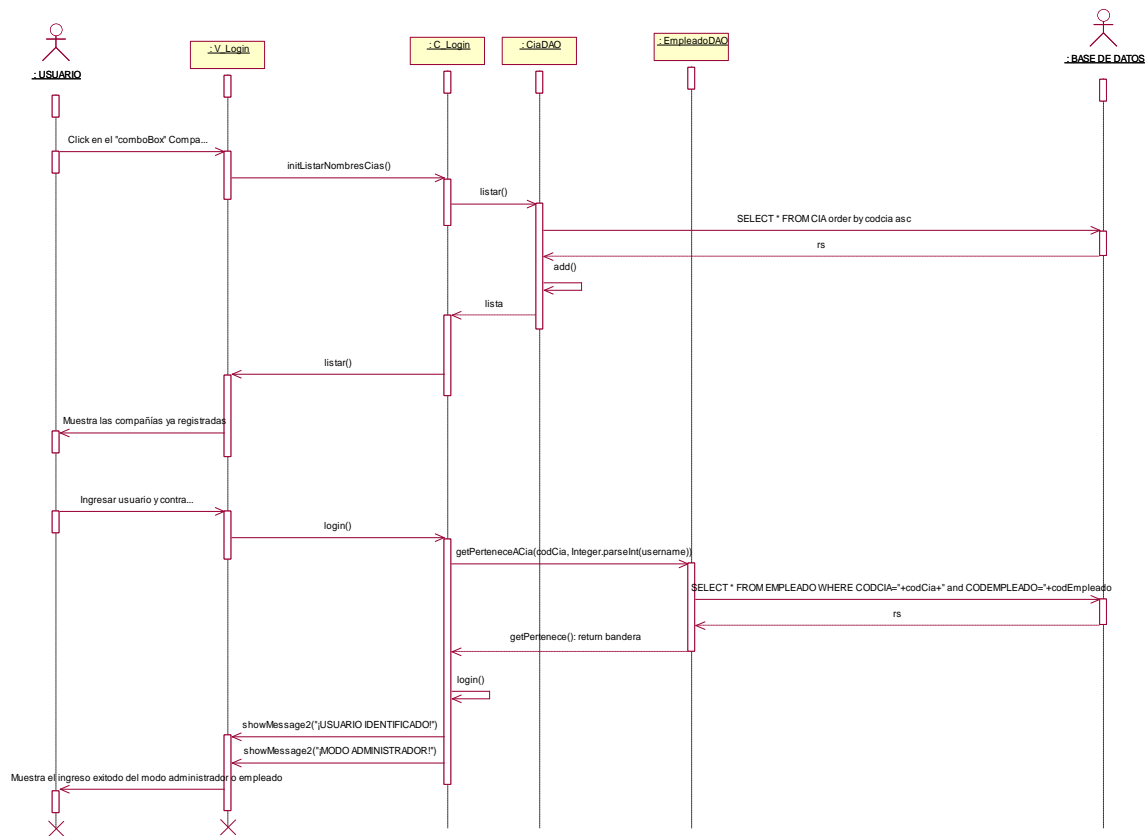
END;

/*-----*/

```

- El procedimiento almacenado empieza creando o reemplazando el procedimiento almacenado llamado INSERTAR_CIA.
- Luego declara los parámetros CODC, DES, DESCORTA y VIG que tendrán el mismo tipo de dato que CODCIA, DESCIA, DESCORTA y VIGENTE respectivamente, que se encuentran en la tabla CIA.
- Finalmente, el procedimiento se encarga de insertar un nuevo registro en la tabla CIA con los parámetros ya mencionados con la instrucción INSERT INTO CIA (codcia, descia, descorta, vigente) VALUES (SEC_CIA.NEXTVAL, DES, DESCORTA, VIG).

Iniciar sesión



Pasos

- El usuario selecciona la compañía con la cuál va ingresar, esto se muestra por la clase V_Login.
- La clase C_Login crear una lista para almacenar los nombres de las compañías a mostrar en el “ComboBox”.
- La clase CiaDao tiene un método listar() que hace que se haga consulta en la base de datos para obtener los datos de las compañías resultadas con “SELECT * FROM CIA order by codcia asc”.
- Se obtienen los datos de las compañías y se añaden con el método add() a la lista de compañías.
- En la interfaz de usuario por la clase V_Login se muestran las compañías a través del “comboBox”.

- Una vez ya seleccionado la compañía, el usuario ingresa su usuario y contraseña, que vendrían a ser el código del empleado o 0, 0 para el administrador.
- En la clase C_Login se utiliza el método login() para identificar al usuario o administrador
- Se utiliza el método “getPerteneceACia(codCia, Integer.parseInt(username))” en la clase EmpleadoDAO para consultar el empleado.
- En la clase EmpleadoDAO se hace una consulta a la base de datos con “SELECT * FROM EMPLEADO WHERE CODCIA="+codCia+" and CODEMPLEADO="+codEmpleado” que nos dará los empleados disponibles en relación con la compañía escogida.
- Se obtiene una bandera que nos indicará si pertenece o no a la compañía.
- Finalmente se muestra un mensaje a través de “showMessage2()” si ingresó como administrador o empleado.

Consultas e Instrucciones

Consultas *SELECT*

```
/*-----*/
```

```
SELECT * FROM CIA order by codcia asc
```

```
/*-----*/
```

- La consulta muestra todos los atributos de la tabla CIA ordenados ascendentemente respecto al CODCIA.

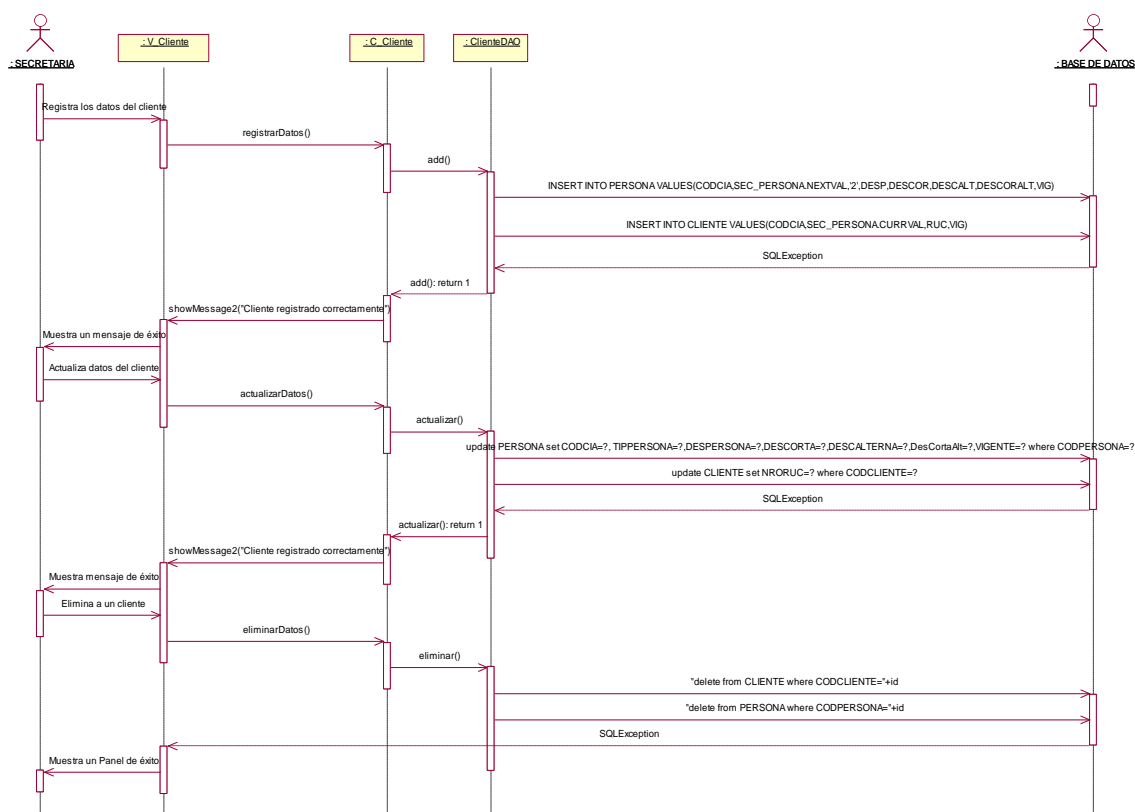
```
/*-----*/
```

```
SELECT * FROM EMPLEADO WHERE CODCIA="+codCia+" and CODEMPLEADO="+codEmpleado
```

```
/*-----*/
```

- Esta consulta muestra todos los empleados con el código cía escogido respecto a la empresa del “comboBox” y el código empleado ingresado en el usuario y contraseña.
- Para el caso de un empleado de código 4 de la empresa Devenco de código 1 se muestra a “Daysi Dr”

Administrar cliente



Pasos

Para registrar

- El usuario llena los datos del cliente a registrar en la clase vista V_Cliente.
- La clase C_Cliente utiliza el método registrarDatos() con el cuál utiliza los datos ingresados en el campo de la vista para hacer un procedimiento en la base de datos.
- Se llama al método add() de la clase ClienteDAO para hacer el procedimiento almacenado “INSERTAR CLIENTE” que consiste en insertar los datos de clientes en la tabla PERSONA y CLIENTE.
- Si el procedimiento se realiza correctamente no existirá alguna excepción y obtendrá un 1 para el siguiente método.

- El método showMessage2() imprimirá un mensaje de éxito.

Para actualizar

- El usuario selecciona alguna fila de datos de un cliente en la tabla de la clase vista V_Cliente, reingresa los datos a cambiar del cliente y hace click en actualizar.
- La clase C_Cliente utiliza el método actualizarDatos() con el cuál utiliza los datos ingresados en el campo de la vista para hacer una instrucción en la base de datos.
- Se llama al método actualizar() de la clase ClienteDAO para hacer la instrucción "update PERSONA set CODCIA=?, TIPPERSONA=?, DESPERSONA=?, DESCORTA=?, DESCALTERNA=?, DesCortaAlt=?, VIGENTE=? where CODPERSONA=?". Y "update CLIENTE set NRORUC=? where CODCLIENTE=?" que actualizaría los datos del cliente con los reingresados.
- Si el procedimiento se realiza correctamente no existirá alguna excepción y obtendrá un 1 para el siguiente método.
- El método showMessage2() imprimirá un mensaje de éxito.

Para eliminar

- El usuario selecciona alguna fila de datos de un cliente en la tabla de la clase vista V_Cliente y hace click en eliminar.
- La clase C_Cliente utiliza el método eliminarDatos() con el cuál eliminará los datos a través de una instrucción en la base de datos.
- Se llama al método eliminar() de la clase ClienteDAO para hacer la instrucción "delete from CLIENTE where CODCLIENTE="+id" y "delete from PERSONA where CODPERSONA="+id"
- Si el procedimiento se realiza correctamente no existirá alguna excepción y se mostrará un mensaje de éxito a través de un JPanel.

Consultas e Instrucciones

Antes de todo resaltar que las instrucciones que se hacen están relacionadas con dos tablas, la tabla PERSONA y CLIENTE. Esto es porque está normalizado y la tabla de PERSONA se relaciona con otras más para aprovechar sus atributos.

En este caso se muestran todas las personas, entre las cuáles encontramos relaciones con las tablas CLIENTE, EMPRESAVENTA, PROYECTO y EMPLEADO.

Procedimiento almacenado INSERTAR_CLIENTE

```

/*-----*/
create or replace NONEDITIONABLE PROCEDURE INSERTAR_CLIENTE(
    CODCIA IN PERSONA.CODCIA%TYPE,
    DESP IN PERSONA.DESPERSONA%TYPE,
    DESCOR IN PERSONA.DESCORTA%TYPE,
    DESCALT IN PERSONA.DESCALTERNA%TYPE,
    DESCORALT IN PERSONA.DESCORTAALT%TYPE,
    VIG IN PERSONA.VIGENTE%TYPE,
    RUC IN CLIENTE.NRORUC%TYPE)
IS
BEGIN
    INSERT INTO PERSONA VALUES (CODCIA, SEC_PERSONA.NEXTVAL, '2', DESP, DESCOR, DESCALT,
    DESCORALT, VIG);
    INSERT INTO CLIENTE VALUES(CODCIA,SEC_PERSONA.CURRVAL,RUC,VIG);
END;
/*-----*/

```

- El procedimiento almacenado empieza creando o reemplazando el procedimiento almacenado llamado INSERTAR_CLIENTE.
- Luego declara los parámetros CODCIA, DESP, DESCOR, DESCALT, DESCORALT, VIG y RUC que tendrán el mismo tipo de dato que CODCIA, DESPERSONA, DESCORTA,

DESCALTERNA, DESCORTALT y VIGENTE de la tabla PERSONA; Y NRORUC de la tabla CLIENTE respectivamente.

- Finalmente, el procedimiento se encarga de insertar un nuevo registro en la tabla PERSONA con los parámetros ya mencionados con la instrucción `INSERT INTO PERSONA VALUES (CODCIA, SEC_PERSONA.NEXTVAL, '2', DESP, DESCOR, DESCALT, DESCORALT, VIG)`. Y también en la tabla CLIENTE con `INSERT INTO CLIENTE VALUES (CODCIA, SEC_PERSONA.CURRVAL, RUC, VIG);`

Instrucciones update para actualizar datos del cliente

```
/*-----*/
update PERSONA set CODCIA=?,
TIPPERSONA=?,DESPERSONA=?,DESCORTA=?,DESCALTERNA=?,DesCortaAlt=?,VIGENTE=? Where
CODPERSONA=?
Update CLIENTE set NRORUC=? Where CODCLIENTE=?
/*-----*/
```

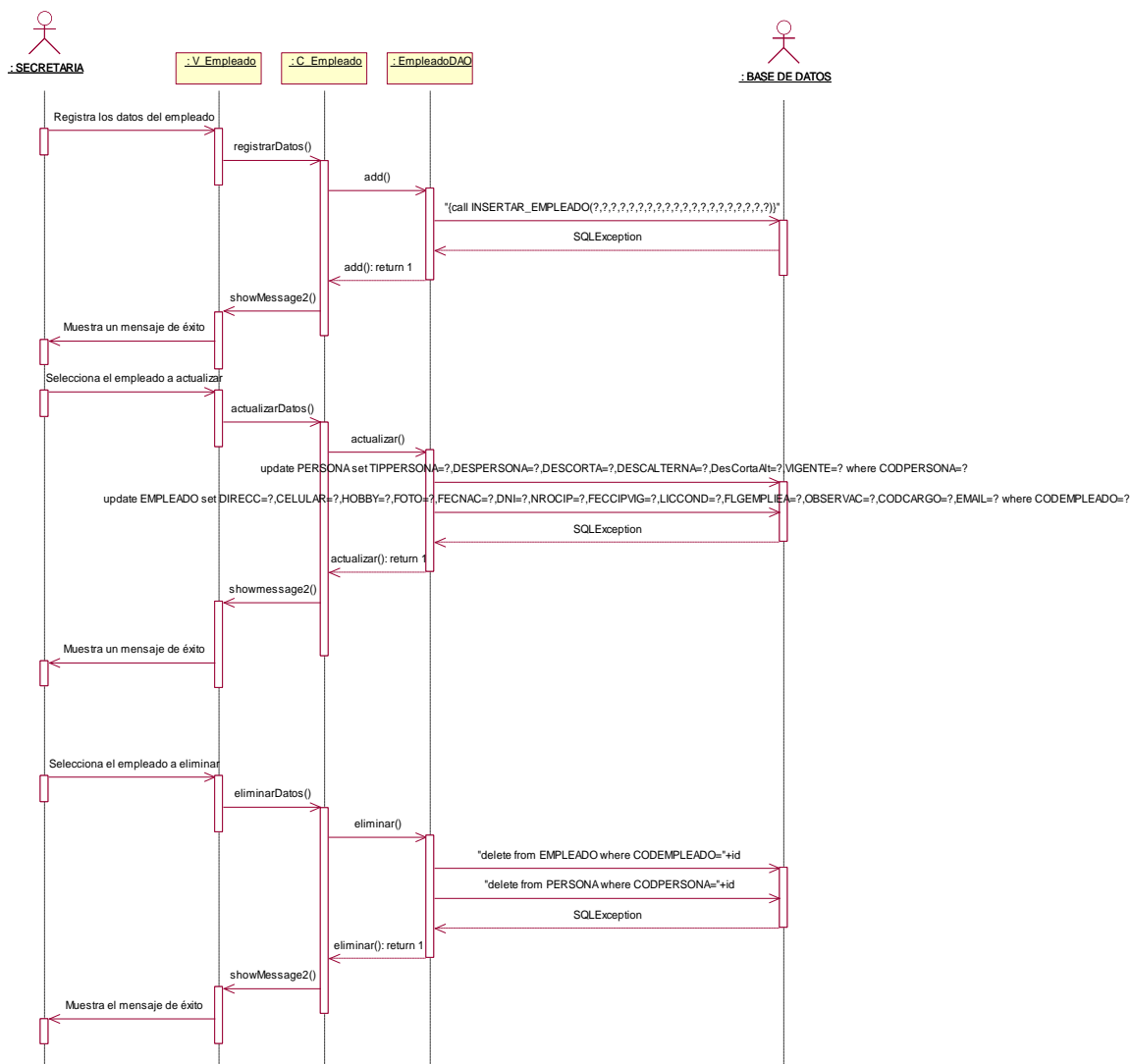
- Esta instrucción actualiza los datos del cliente con los datos de CODCIA, TIPPERSONA, DESPERSONA, DESCORTA, DESCALTERNA, DESCORTAALT Y VIGENTE para la tabla PERSONA y el NRORUC para la tabla CLIENTE.

Instrucciones delete para eliminar al cliente

```
/*-----*/
delete from CLIENTE where CODCLIENTE= id
delete from PERSONA where CODPERSONA=id
/*-----*/
```

- Esta instrucción elimina al cliente según el ID que se obtenga de la selección del usuario, primero elimina el registro de la tabla CLIENTE y luego elimina el registro de la tabla PERSONA.

Administrar empleado



Pasos

Para registrar

- El usuario llena los datos del nuevo empleado en la clase vista V_Empleado.
- La clase C_Empleado utiliza el método registrarDatos() con el cuál utiliza los datos ingresados en el campo de la vista para hacer un procedimiento en la base de datos.

- Se llama al método add() de la clase EmpleadoDAO para hacer el procedimiento almacenado “INSERTAR CLIENTE” que consiste en insertar los datos de clientes en la tabla PERSONA y CLIENTE.
- Si el procedimiento se realiza correctamente no existirá alguna excepción y obtendrá un 1 para el siguiente método.
- El método showMessage2() imprimirá un mensaje de éxito.

Para actualizar

- El usuario selecciona alguna fila de datos de un cliente en la tabla de la clase vista V_Empleado, reingresa los datos a cambiar del cliente y hace clic en actualizar.
- La clase C_Empleado utiliza el método actualizarDatos() con el cuál utiliza los datos ingresados en el campo de la vista para hacer una instrucción en la base de datos.
- Se llama al método actualizar() de la clase EmpleadoDAO para hacer la instrucción update
PERSONA set TIPPERSONA=?, DESPERSONA=?, DESCORTA=?, DESCALTERNA=?,
DesCortaAlt=?, VIGENTE=? where CODPERSONA=? Y update EMPLEADO set DIRECC=?,
CELULAR=?, HOBBY=?, FOTO=?, FECNAC=?, DNI=?, NROCIP=?, FECCIPVIG=?, LICCOND=?, FLGEMPL
IEA=?, OBSERVAC=?, CODCARGO=?, EMAIL=? where CODEMPLEADO=? que actualizaría los datos
del empleado con los reingresados.
- Si el procedimiento se realiza correctamente no existirá alguna excepción y obtendrá un 1 para el siguiente método.
- El método showMessage2() imprimirá un mensaje de éxito.

Para eliminar

- El usuario selecciona alguna fila de datos de un cliente en la tabla de la clase vista V_Empleado y hace clic en eliminar.
- La clase C_Empleado utiliza el método eliminarDatos() con el cuál eliminará los datos a través de una instrucción en la base de datos.

- Se llama al método eliminar() de la clase EmpleadoDAO para hacer la instrucción " "delete from EMPLEADO where CODEMPLEADO="+id y "delete from PERSONA where CODPERSONA="+id".
- Si el procedimiento se realiza correctamente no existirá alguna excepción y se mostrará un mensaje de éxito a través de showMessage2().

Consultas e Instrucciones

Antes de todo resaltar que las instrucciones que se hacen están relacionadas con dos tablas, la tabla PERSONA y EMPLEADO. Esto es porque está normalizado y la tabla de PERSONA se relaciona con otras más para aprovechar sus atributos.

En este caso se muestran todas las personas, entre las cuáles encontramos relaciones con las tablas CLIENTE, EMPRESAVENTA, PROYECTO y EMPLEADO.

Procedimiento almacenado INSERTAR_EMPLEADO

/*-----*/

create or replace NONEDITIONABLE PROCEDURE INSERTAR_EMPLEADO(

```

    CIA IN PERSONA.CODCIA%TYPE,
    TIP IN PERSONA.TIPPERSONA%TYPE,
    DESP IN PERSONA.DESPERSONA%TYPE,
    DESCOR IN PERSONA.DESCORTA%TYPE,
    DESCALT IN PERSONA.DESCALTERNA%TYPE,
    DESCORALT IN PERSONA.DESCORTAALT%TYPE,
    VIG IN PERSONA.VIGENTE%TYPE,
    DIR IN EMPLEADO.DIRECC%TYPE,
    CEL IN EMPLEADO.CELULAR%TYPE,
    HOB IN EMPLEADO.HOBBY%TYPE,
    FOT IN EMPLEADO.FOTO%TYPE,
    NAC IN EMPLEADO.FECNAC%TYPE,
```

```

        IDEN IN EMPLEADO.DNI%TYPE,
        CIP IN EMPLEADO.NROCIP%TYPE,
        CIPVIG IN EMPLEADO.FECCIPVIG%TYPE,
        COND IN EMPLEADO.LICCOND%TYPE,
        FLG IN EMPLEADO.FLGEMPLIEA%TYPE,
        OBS IN EMPLEADO.OBSERVAC%TYPE,
        CODCAR IN EMPLEADO.CODCARGO%TYPE,
        CORREO IN EMPLEADO.EMAIL%TYPE)
IS
BEGIN
    INSERT INTO PERSONA
VALUES(CIA,SEC_PERSONA.NEXTVAL,TIP,DESP,DESCOR,DESCALT,DESCORALT,VIG);

    INSERT INTO EMPLEADO
VALUES(CIA,SEC_PERSONA.CURRVAL,DIR,CEL,HOB,FOT,NAC,IDEN,CIP,CIPVIG,COND,FLG,OBS,CODCA
R,CORREO,VIG);
END;

/*-----*/

```

- El procedimiento almacenado empieza creando o reemplazando el procedimiento almacenado llamado INSERTAR_EMPLEADO.
- Luego declara los parámetros a insertar con el mismo tipo de las tablas que correspondan.
- Quiero señalar que para el dato de la imagen se está utilizando un tipo de dato BLOB que hace referencia a un tipo de dato binario en bruto que se guarda en la base de datos como tal y se recupera en el IDE Netbeans para poder leerlo.
- Finalmente, el procedimiento se encarga de insertar un nuevo registro en la tabla PERSONA con los parámetros ya mencionados con la instrucción INSERT INTO PERSONA VALUES (CIA,SEC_PERSONA.NEXTVAL, TIP,DESP,DESCOR,DESCALT,DESCORALT,VIG); Y también en

la tabla EMPLEADO con INSERT INTO EMPLEADO VALUES (CIA, SEC_PERSONA.CURRVAL,
DIR,CEL,HOB,FOT,NAC,IDEN,CIP,CIPVIG,COND,FLG,OBS,CODCAR,CORREO,VIG);

Instrucciones update para actualizar datos del cliente

```
/*-----*/
update PERSONA set TIPPERSONA=?, DESPERSONA=?, DESCORTA=?, DESCALTERNA=?,
DesCortaAlt=?, VIGENTE=? where CODPERSONA=?Update CLIENTE set NRORUC=? Where
CODCLIENTE=?

update EMPLEADO set DIRECC=?, CELULAR=?, HOBBY=?, FOTO=?, FECNAC=?, DNI=?, NROCIP=?,
FECCIPVIG=?, LICCOND=?, FLGEMPLIEA=?, OBSERVAC=?, CODCARGO=?, EMAIL=? where CODEMPLEADO=?
/*-----*/
```

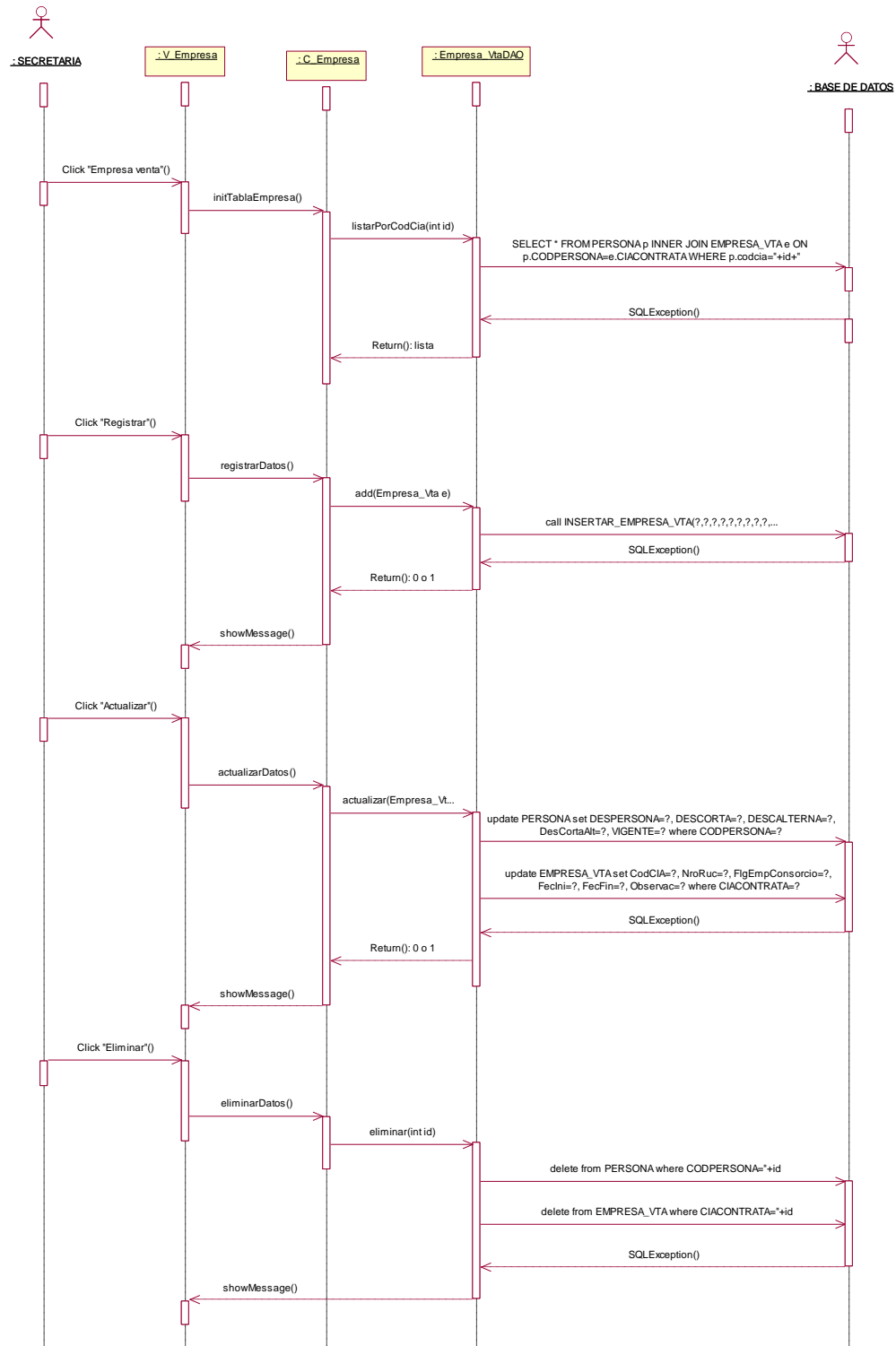
- Esta instrucción actualiza los datos del cliente.

Instrucciones delete para eliminar al cliente

```
/*-----*/
delete from EMPLEADO where CODEMPLEADO="+id
"delete from PERSONA where CODPERSONA="+id
/*-----*/
```

- Esta instrucción elimina al cliente según el ID que se obtenga de la selección del usuario, primero elimina el registro de la tabla EMPLEADO y luego elimina el registro de la tabla PERSONA.

Administrar empresa venta



Pasos

- La secretaria hace click en Empresa venta.
- La clase V_Empresa inicia la interfaz con la función `initTablaEmpresa()` que pertenece a la clase C_Empresa.
- La clase Empresa_VtaDAO ejecuta la función `listarPorCodCia(int id)` que se comunica con la base de datos mediante la consulta `SELECT * FROM PERSONA p INNER JOIN EMPRESA_VTA e ON p.CODPERSONA=e.CIACONTRATA WHERE p.codcia="+id+"`, lo cual mostrara las empresas registradas hasta ese momento.

Registrar

- La V_Empresa tiene un apartado donde la secretaria ingresa los datos de la nueva empresa que se va a registrar, luego hace click en Registrar
- La clase C_Empresa obtiene los datos colocados por la secretaria mediante la función `registrarDatos()` y ejecuta la función `add(Empresa_Vta e)` de la clase Empresa_VtaDAO
- La clase Empresa_VtaDAO llama al procedimiento almacenado `INSERTAR_EMPRESA_VTA(?, ?, ?, ?, ?, ?, ?, ?, ?, ?)` que en la base de datos está como las consultas `INSERT INTO PERSONA VALUES (CODCIA, SEC_PERSONA.NEXTVAL, '2', DESP, DESCOR, DESCALT, DESCORALT, VIG);`

`INSERT INTO EMPRESA_VTA`

`VALUES(CODCIA, SEC_PERSONA.CURRVAL, RUC, CONSORCIO, INI, FIN, NULL, OBS, VIG);`

- Luego, según se haya ejecutado la petición, la base de datos vuelve en una excepción, `SQLException`, lo que se convertirá en un 1 o 0.
- Finalmente, este 1 o 0 representa un mensaje en la vista para la secretaria que será "Empresa registrado correctamente" si la consulta fue exitosa.

Actualizar

- La secretaria selecciona la empresa que desea modificar, una vez modificado los datos hace click en Actualizar.

- La clase C_Empresa ejecuta la función actualizarDatos(), la cual a la vez ejecuta la función actualizar(Empresa_Vta e) en la clase Empresa_VtaDAO.
- La clase Empresa_VtaDAO tiene la conexión a la base de datos y ejecuta la actualización del elemento mediante las peticiones `UPDATE PERSONA set DESPERSONA=?, DESCORTA=?, DESCALTERNA=?, DesCortaAlt=?, VIGENTE=? where CODPERSONA=?` y `UPDATE EMPRESA_VTA set CodCIA=?, NroRuc=?, FlgEmpConsortio=?, FecIni=?, FecFin=?, Observac=? where CIACONTRATA=?`.
- Luego, según se haya ejecutado la petición, la base de datos vuelve en una excepción, `SQLException`, lo que se convertirá en un 1 o 0.
- Finalmente, este 1 o 0 representa un mensaje en la vista para la secretaria que será "Empresa registrado correctamente" si la consulta fue exitosa.

Eliminar

- La secretaria selecciona la empresa que desea eliminar y hace click en Eliminar.
- La clase C_Empresa ejecuta la función eliminarDatos(), la cual a la vez ejecuta la función eliminar(int id) en la clase Empresa_VtaDAO.
- La clase ElementosDAO tiene la conexión a la base de datos y ejecuta la eliminación mediante las peticiones `DELETE from EMPRESA_VTA where CIACONTRATA="+id` y `DELETE from PERSONA where CODPERSONA="+id`
- Finalmente, se muestra el mensaje "Empresa eliminada con éxito" si la consulta fue exitosa.

Consultas e Instrucciones

Procedimiento almacenado INSERTAR_EMPRESA_VTA

/-----*/*

```
create or replace PROCEDURE INSERTAR_EMPRESA_VTA(
    CODCIA IN PERSONA.CODCIA%TYPE,
    DESP IN PERSONA.DESPERSONA%TYPE,
    DESCOR IN PERSONA.DESCORTA%TYPE,
```

```

DESCALT IN PERSONA.DESCALTERNA%TYPE,
DESCORALT IN PERSONA.DESCORTAALT%TYPE,
VIG IN PERSONA.VIGENTE%TYPE,
RUC IN EMPRESA_VTA.NRORUC%TYPE,
CONSORCIO IN EMPRESA_VTA.FLGEMPCONSORCIO%TYPE,
INI IN EMPRESA_VTA.FECINI%TYPE,
FIN IN EMPRESA_VTA.FECFIN%TYPE,
OBS IN EMPRESA_VTA.OBSERVAC%TYPE)
IS
BEGIN
    INSERT INTO PERSONA
VALUES(CODCIA,SEC_PERSONA.NEXTVAL,'2',DESP,DESCOR,DESCALT,DESCORALT,VIG);
    INSERT INTO EMPRESA_VTA
VALUES(CODCIA,SEC_PERSONA.CURRVAL,RUC,CONSORCIO,INI,FIN,NULL,OBS,VIG);
END;
/*-----*/

```

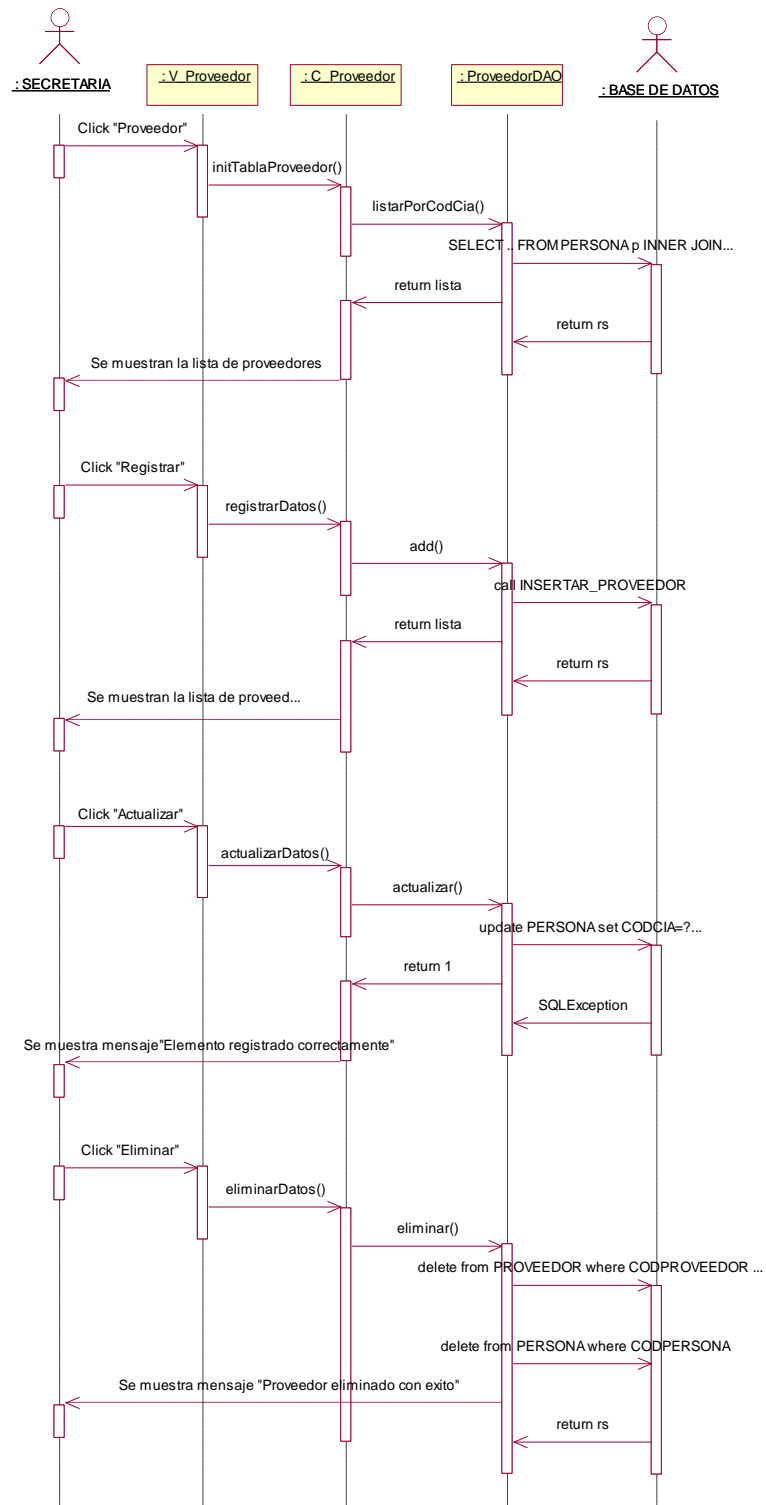
El procedimiento se encarga de insertar un nuevo registro en la tabla PERSONA Y EMPRESA_VTA con los parámetros ya mencionados con las instrucciones:

1. INSERT INTO PERSONA


```
VALUES(CODCIA,SEC_PERSONA.NEXTVAL,'2',DESP,DESCOR,DESCALT,DESCORALT,VIG)
```
2. INSERT INTO EMPRESA_VTA


```
VALUES(CODCIA,SEC_PERSONA.CURRVAL,RUC,CONSORCIO,INI,FIN,NULL,OBS,VIG);
```

Administrar proveedor



Pasos

- La secretaria da click en Proveedores.
- La clase V_Proveedor inicia la interfaz con la función initTablaProveedor() que pertenece a la clase C_Proveedor
- La clase C_Proveedor se comunica con la clase ProveedorDAO mediante el método listarPorCodCia donde se genera la consulta SELECT

p.codcia,p.codpersona,p.tippersona,p.despersona,p.descorta,p.descalterna,p.DesCortaAlt, " +
 "pr.nroruc,p.vigente " + "FROM PERSONA p INNER JOIN PROVEEDOR pr ON
 p.CODPERSONA=pr.CODPROVEEDOR WHERE p.CODCIA="+id+" order by p.codpersona";
- Luego, la consulta se manda a realizar a la base de datos para mostrar todos los proveedores ya registrados.

Registrar

- La V_Proveedor tiene un apartado donde el administrador ingresa los datos de un nuevo proveedor que se va a registrar, luego hace clic en Registrar. La clase C_Proveedor obtiene los datos colocados por el administrador mediante la función registrarDatos() y ejecuta la función add(proveedor) de la clase ProveedorDAO.
- La clase ProveedorDAO llama al procedimiento almacenado call INSERTAR_PROVEEDOR(?,?,?,?,?,?) que en la base de datos está como la consulta: INSERT INTO PROVEEDOR (CODCIA, CODPROV, NOMPROV, DIRECCION, TELEFONO, EMAIL) VALUES (?, ?, ?, ?, ?, ?);
- Luego, según se haya ejecutado la petición, la base de datos devuelve una excepción SQLException, lo que se convertirá en un 1 o 0.
- Finalmente, este 1 o 0 representa un mensaje en la vista para el administrador que será "Proveedor registrado correctamente", si la consulta fue exitosa.

Actualizar

- La clase V_Proveedor obtiene los datos del proveedor seleccionado y permite que el administrador edite esos datos.
- La clase V_Proveedor obtiene los datos editados por el administrador y hace clic en "Actualizar". La clase C_Proveedor recibe los datos editados mediante la función actualizarDatos() y ejecuta la función update(proveedor) de la clase ProveedorDAO.
- La clase ProveedorDAO llama al procedimiento almacenado call ACTUALIZAR_PROVEEDOR(?,?,?,?,?) que en la base de datos está como la consulta:
UPDATE PROVEEDOR SET NOMPROV = ?, DIRECCION = ?, TELEFONO = ?, EMAIL = ?
WHERE CODCIA = ? AND CODPROV = ?;
- Luego, según se haya ejecutado la petición, la base de datos devuelve una excepción SQLException, lo que se convertirá en un 1 o 0.
- Finalmente, este 1 o 0 representa un mensaje en la vista para el administrador que será "Proveedor actualizado correctamente", si la consulta fue exitosa.

Eliminar

- La clase V_Proveedor obtiene el proveedor seleccionado para eliminar.
- La clase C_Proveedor recibe la solicitud de eliminación a través del método eliminarDatos() y ejecuta la función delete(proveedor) de la clase ProveedorDAO.
- La clase ProveedorDAO llama al procedimiento almacenado call ELIMINAR_PROVEEDOR(?,?) que en la base de datos está como la consulta: DELETE FROM PROVEEDOR WHERE CODCIA = ? AND CODPROV = ?;
- Luego, según se haya ejecutado la petición, la base de datos devuelve una excepción SQLException, lo que se convertirá en un 1 o 0.
- Finalmente, este 1 o 0 representa un mensaje en la vista para el administrador que será "Proveedor eliminado correctamente", si la consulta fue exitosa.

Consultas e instrucciones

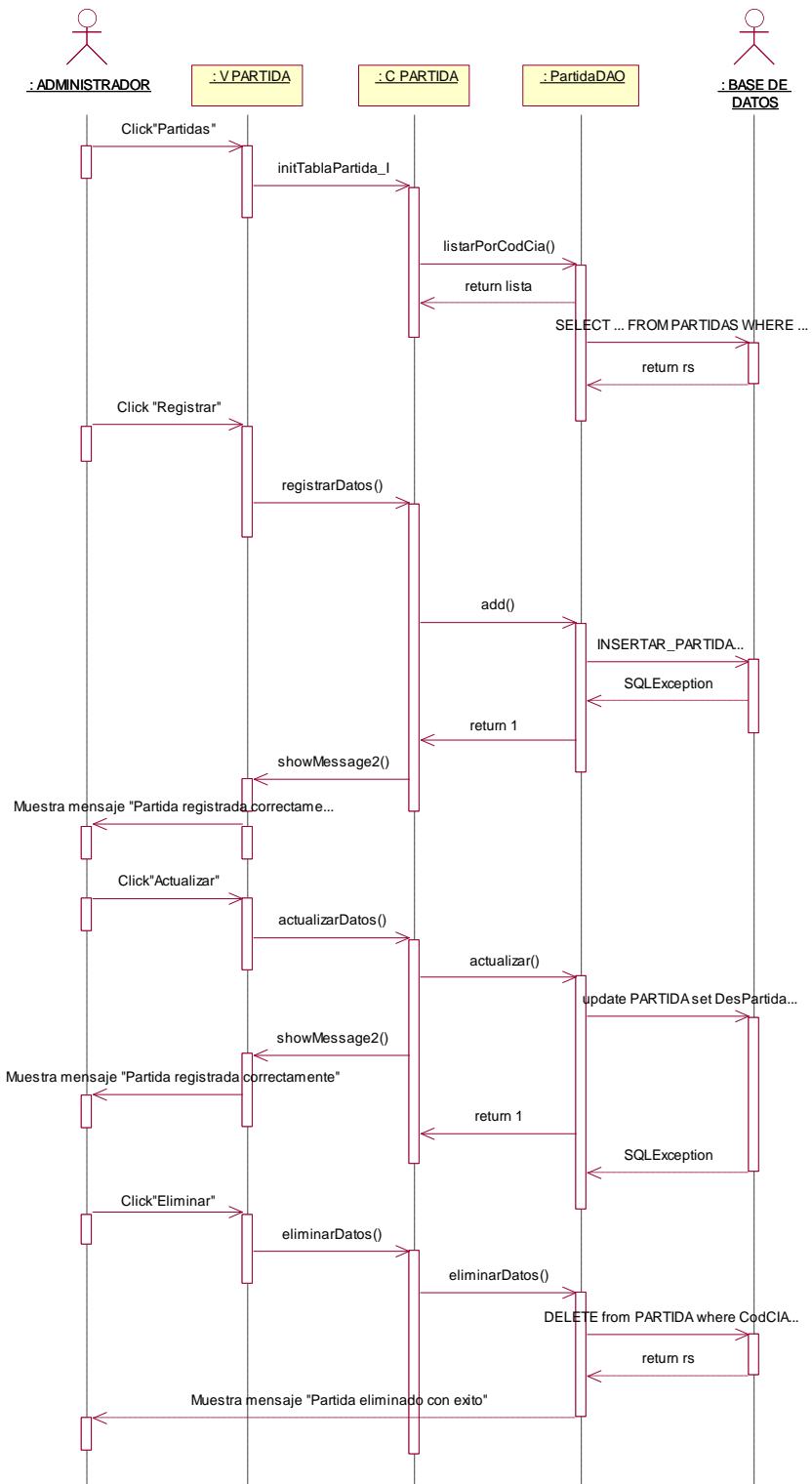
Procedimiento almacenado INSERTAR_PROVEEDOR

```
/*-----*/
```

```
create or replace PROCEDURE INSERTAR_PROVEEDOR(
    CODC IN PERSONA.CODCIA%TYPE,
    TIP IN PERSONA.TIPPERSONA%TYPE,
    DESP IN PERSONA.DESPERSONA%TYPE,
    DESCOR IN PERSONA.DESCORTA%TYPE,
    DESCALT IN PERSONA.DESCALTERNA%TYPE,
    DESCORALT IN PERSONA.DESCORTAALT%TYPE,
    VIG IN PERSONA.VIGENTE%TYPE,
    RUC IN PROVEEDOR.NRORUC%TYPE)
IS
BEGIN
    INSERT INTO PERSONA
VALUES(CODC,SEC_PERSONA.NEXTVAL,TIP,DESP,DESCOR,DESCALT,DESCORALT,VIG);
    INSERT INTO PROVEEDOR VALUES(CODC,SEC_PERSONA.CURRVAL,RUC,VIG);
END;
/*-----*/
```

- El procedimiento almacenado empieza creando o reemplazando el procedimiento almacenado llamado INSERTAR_PROVEEDOR.
- Luego declara los parámetros para registrar al nuevo proveedor.

Administrar partidas



Pasos

- El administrador da click en Partidas.
- La clase V_Partida inicia la interfaz con la función `initTablaPartida_I()` que pertenece a la clase C_Partida.
- La clase C_Partida se comunica con la clase PartidaDAO mediante el método `listarPorCodCia` donde se genera la consulta "SELECT "+
"p.CodPartida,p.CodPartidas,p.DesPartida,p.tUniMed,p.eUniMed,p.Vigente " + "FROM
PARTIDA p WHERE p.CodCIA="+id+" AND p.IngEgr="+tip+" order by p.codPartida";
- Luego, la consulta se manda a realizar a la base de datos para mostrar todas las partidas ya creadas en la compañía.

Registrar

- La V_Partida tiene un apartado donde el administrador ingresa los datos de la nueva partida que se va a registrar, luego hace click en Registrar
- La clase C_Partida obtiene los datos colocados por el administrador mediante la función `registrarDatos()` y ejecuta la función `add(Partida p)` de la clase PartidaDAO
- La clase PartidaDAO llama al procedimiento almacenado `call INSERTAR_PARTIDA(?,?,?,?,?)` que en la base de datos está como la consulta
`INSERT INTO PARTIDA
VALUES(CODCIA,INGEGRE,SEC_PARTIDA_E.NEXTVAL,TO_CHAR(SEC_CODPARTIDA
S.NEXTVAL,'99,999,999'),DESPARTIDA,'1',1,tUniMed,eUniMed,1,VIG);`
- Luego, según se haya ejecutado la petición, la base de datos vuelve en una excepción, `SQLException`, lo que se convertirá en un 1 o 0.
- Finalmente, este 1 o 0 representa un mensaje en la vista para el administrador que será "Partida registrada correctamente", si la consulta fue exitosa.

Actualizar

- La clase V_Partida obtiene los datos de la partida seleccionada y permite que el administrador edite esos datos.

- La clase C_Partida recibe los datos editados mediante la función actualizarDatos() y ejecuta la función update(Partida p) de la clase PartidaDAO.
- La clase PartidaDAO llama al procedimiento almacenado call ACTUALIZAR_PARTIDA(?,?,?,?,?) que en la base de datos está como la consulta: UPDATE PARTIDA SET DESPARTIDA = ?, tUniMed = ?, eUniMed = ?, VIG = ? WHERE CODCIA = ? AND CODPARTIDA = ?;
- Luego, según se haya ejecutado la petición, la base de datos vuelve en una excepción SQLException, lo que se convertirá en un 1 o 0.
- Finalmente, este 1 o 0 representa un mensaje en la vista para el administrador que será "Partida actualizada correctamente", si la consulta fue exitosa.

Eliminar

- La clase V_Partida obtiene la partida seleccionada para eliminar.
- La clase C_Partida recibe la solicitud de eliminación a través del método eliminarDatos() y ejecuta la función delete(Partida p) de la clase PartidaDAO.
- La clase PartidaDAO llama al procedimiento almacenado call ELIMINAR_PARTIDA(?) que en la base de datos está como la consulta:DELETE FROM PARTIDA WHERE CODPARTIDA = ?;
- Luego, según se haya ejecutado la petición, la base de datos vuelve en una excepción SQLException, lo que se convertirá en un 1 o 0.
- Finalmente, este 1 o 0 representa un mensaje en la vista para el administrador que será "Partida eliminada correctamente", si la consulta fue exitosa.

Consultas e instrucciones

Procedimiento almacenado INSERTAR_PARTIDA

```
/*-----*/
create or replace NONEDITIONABLE PROCEDURE INSERTAR_PARTIDA(
    CODCIA IN PARTIDA.CODCia%TYPE,
```

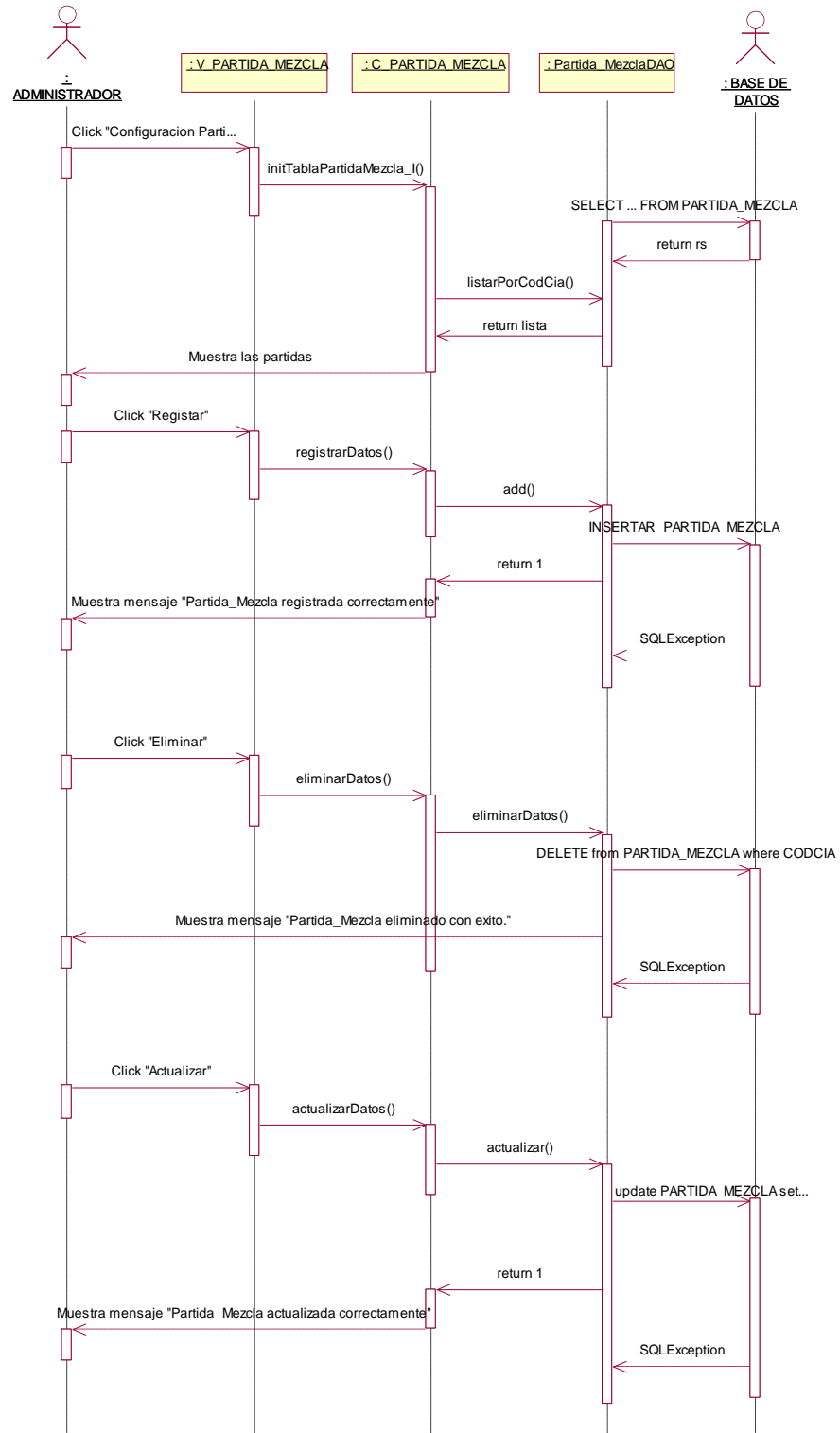
```

    INEGRE IN PARTIDA.IngEgr%TYPE,
    DESPARTIDA IN PARTIDA.DESPARTIDA%TYPE,
    tUniMed IN PARTIDA.TUNIMED%TYPE,
    eUniMed IN PARTIDA.EUNIMED%TYPE,
    VIG IN PARTIDA.VIGENTE%TYPE)
IS
BEGIN
IF(INGEGRE = 'E') THEN
    INSERT INTO PARTIDA VALUES(CODCIA,INGEGRE,SEC_PARTIDA_E.NEXTVAL,
    TO_CHAR(SEC_CODPARTIDAS.NEXTVAL,'99,999,999'),DESPARTIDA,'1',1,tUniMed,eUniMed,1,VIG)
;
END IF;
IF(INGEGRE = 'I') THEN
    INSERT INTO PARTIDA VALUES(CODCIA,INGEGRE,SEC_PARTIDA_I.NEXTVAL,
    TO_CHAR(SEC_CODPARTIDAS.NEXTVAL,'99,999,999'),DESPARTIDA,'1',1,tUniMed,eUniMed,1,VIG)
;
END IF;
END;
/*-----*/

```

- El procedimiento almacenado empieza creando o reemplazando el procedimiento almacenado llamado INSERTAR_PARTIDA.
- Luego declara los parámetros CODC, INEGRE, DESPARTIDA, tUniMed, eUniMed y VIG que almacenaran los valores de la partida registrada.

Administrar partidas padres



Pasos

- El administrador da click en Configuración Partidas.
- La clase V_Partida_Mezcla inicia la interfaz con la función initTablaPartidaMezcla_I() que pertenece a la clase C_Partida_Mezcla.
- La clase C_Partida_Mezcla se comunica con la clase Partida_MezclaDAO mediante el método listarPorCodCia donde se genera la consulta "SELECT "
 +"m.corr,m.codpartida,m.padcodpartida,m.nivel,m.orden,m.tunimed,m.eunimed,m.costounit,m.vi
 gente " + "FROM PARTIDA_MEZCLA m WHERE m.codcia="+id+" AND m.ingegr=""+tip+"
 order by m.corr".
- Luego, la consulta se manda a realizar a la base de datos para mostrar todas las partidas ya creadas en la compañía.

Registrar

- La V_Partida_Mezcla tiene un apartado donde el administrador ingresa los para la partida padre e hijo que se va a registrar, luego hace click en Registrar
- La clase C_Partida_Mezcla obtiene los datos colocados por el administrador mediante la función registrarDatos() y ejecuta la función add(pm) de la clase Partida_MezclaDAO
- La clase Partida_MezclaDAO llama al procedimiento almacenado call
 INSERTAR_PARTIDA_MEZCLA que en la base de datos está como la consulta INSERT INTO
 PARTIDA_MEZCLA
 VALUES(CODCIA,INGEGRE,CODPAR,SEC_PARTIDA_MEZCLA_E.NEXTVAL,
 PADCOD,tUniMed,eUniMed,COSTO,NIVEL,ORDEN,VIG);
- Luego, según se haya ejecutado la petición, la base de datos vuelve en una excepción, SQLException, lo que se convertirá en un 1 o 0.
- Finalmente, este 1 o 0 representa un mensaje en la vista para el administrador que será "Partida_Mecla registrada correctamente", si la consulta fue exitosa.

Actualizar

- La clase V_Partida_Mezcla obtiene los datos de la partida mezcla seleccionada y permite que el administrador edite esos datos.
- La clase C_Partida_Mezcla recibe los datos editados mediante la función actualizarDatos() y ejecuta la función update(pm) de la clase Partida_MezclaDAO.
- La clase Partida_MezclaDAO llama al procedimiento almacenado call ACTUALIZAR_PARTIDA_MEZCLA que en la base de datos está como la consulta: UPDATE PARTIDA_MEZCLA SET PADCOD = ?, tUniMed = ?, eUniMed = ?, COSTO = ?, NIVEL = ?, ORDEN = ?, VIG = ? WHERE CODCIA = ? AND CODPAR = ?;
- Luego, según se haya ejecutado la petición, la base de datos vuelve en una excepción SQLException, lo que se convertirá en un 1 o 0.
- Finalmente, este 1 o 0 representa un mensaje en la vista para el administrador que será "Partida mezcla actualizada correctamente", si la consulta fue exitosa.

Eliminar

- La clase V_Partida_Mezcla obtiene la partida mezcla seleccionada para eliminar.
- La clase C_Partida_Mezcla recibe la solicitud de eliminación a través del método eliminarDatos() y ejecuta la función delete(pm) de la clase Partida_MezclaDAO.
- La clase Partida_MezclaDAO llama al procedimiento almacenado call ELIMINAR_PARTIDA_MEZCLA que en la base de datos está como la consulta: DELETE FROM PARTIDA_MEZCLA WHERE CODPAR = ?;
- Luego, según se haya ejecutado la petición, la base de datos vuelve en una excepción SQLException, lo que se convertirá en un 1 o 0.
- Finalmente, este 1 o 0 representa un mensaje en la vista para el administrador que será "Partida mezcla eliminada correctamente", si la consulta fue exitosa.

Consultas e instrucciones

Procedimiento almacenado INSERTAR_PARTIDA_MEZCLA

```

/*-----*/

create or replace NONEDITIONABLE PROCEDURE INSERTAR_PARTIDA_MEZCLA(

    CODCIA IN PARTIDA_MEZCLA.CODCIA%TYPE,

    INEGRE IN PARTIDA_MEZCLA.IngEgr%TYPE,

    CODPAR IN PARTIDA_MEZCLA.CODPARTIDA%TYPE,

    PADCOD IN PARTIDA_MEZCLA.PADCODPARTIDA%TYPE,

    tUniMed IN PARTIDA_MEZCLA.TUNIMED%TYPE,

    eUniMed IN PARTIDA_MEZCLA.EUNIMED%TYPE,

    COSTO IN PARTIDA_MEZCLA.COSTOUNIT%TYPE,

    NIVEL IN PARTIDA_MEZCLA.NIVEL%TYPE,

    ORDEN IN PARTIDA_MEZCLA.ORDEN%TYPE,

    VIG IN PARTIDA_MEZCLA.VIGENTE%TYPE)

IS

BEGIN

IF(INGEGRE = 'E') THEN

    INSERT INTO PARTIDA_MEZCLA

VALUES(CODCIA,INGEGRE,CODPAR,SEC_PARTIDA_MEZCLA_E.NEXTVAL,

    PADCOD,tUniMed,eUniMed,COSTO,NIVEL,ORDEN,VIG);

END IF;

IF(INGEGRE = 'I') THEN

    INSERT INTO PARTIDA_MEZCLA

VALUES(CODCIA,INGEGRE,CODPAR,SEC_PARTIDA_MEZCLA_I.NEXTVAL,

    PADCOD,tUniMed,eUniMed,COSTO,NIVEL,ORDEN,VIG);

END IF;

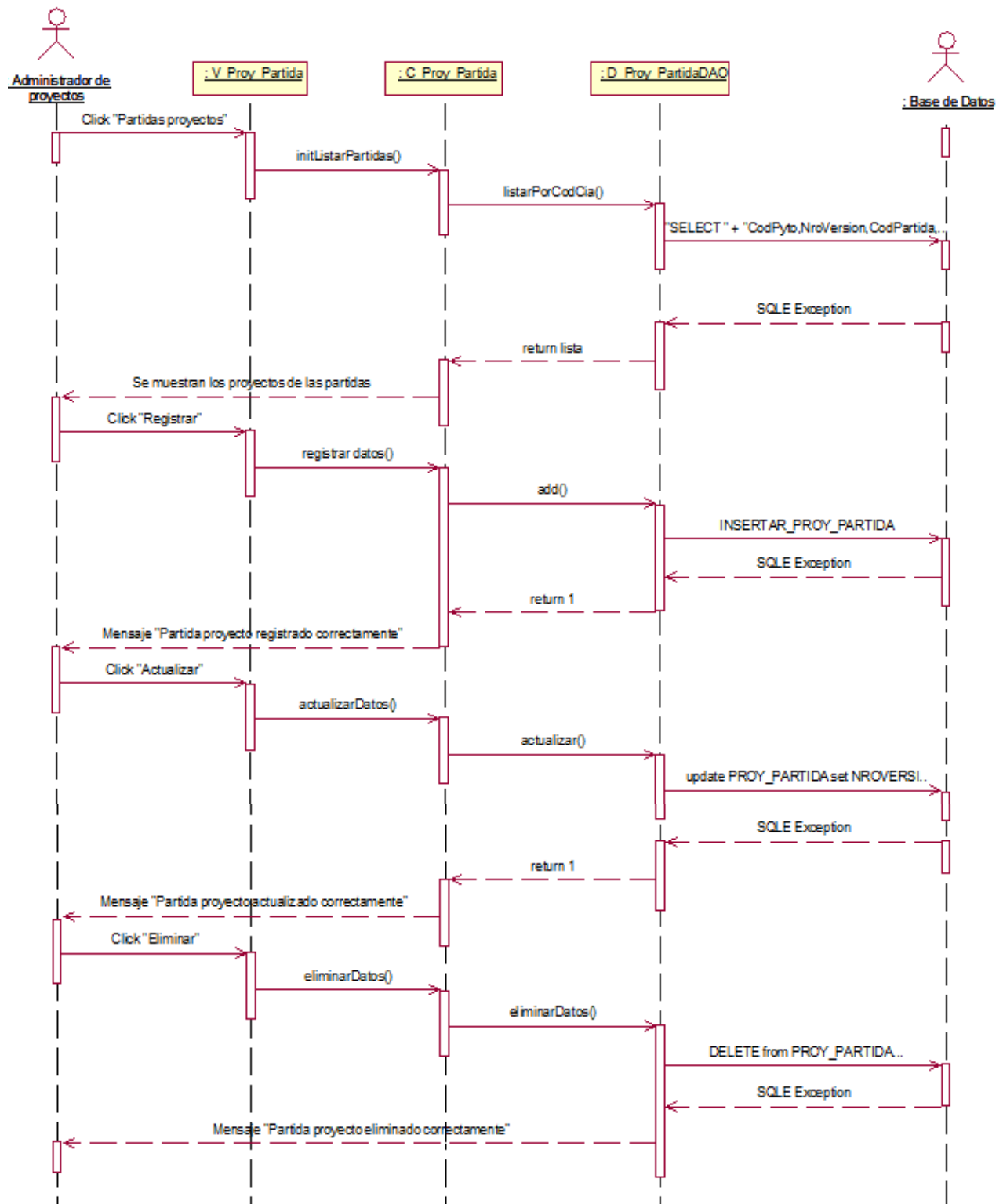
END;

/*-----*/

```


- El procedimiento almacenado empieza creando o reemplazando el procedimiento almacenado llamado INSERTAR_PARTIDA_MEZCLA.
- Luego declara los parámetros
CODCIA, INEGRE, CODPAR, SEC_PARTIDA_MEZCLA_I.NEXTVAL, PADCOD, tUnimed, eUnimed, COSTO, NIVEL, ORDEN, VIG que almacenaran los valores de la partida registrada.

Administrar partidas de proyectos



Pasos

- El administrador de proyectos da click en Proyectos partidas
- La clase V_Proj_Partida inicia la interfaz con la función initListarPartidas() que pertenece a la clase C_Proj_Partida.
- La clase C_Proj_Partida se comunica con la clase Proj_Partida DAO mediante el método listarPorCodCia donde se genera la consulta "SELECT " + "CodPyto,NroVersion,CodPartida,CodEstado,Vigente " + "FROM PROY_PARTIDA WHERE CODCIA="+id+" AND INGEGR="+tip+" order by CODPYTO"; Luego, la consulta se manda a realizar a la base de datos para mostrar todos los proyectos que ya estaban registrados en la compañía

Registrar

- La V_Proj_Partida tiene una opción en donde el administrador de proyectos ingresa los datos de un nuevo proyecto que se va a registrar, luego hace click en Registrar
- La clase C_Proj_Partida obtiene los datos colocados por el administrador mediante la función registrarDatos() y ejecuta la función add() de la clase Proj_Partida DAO
- La clase Proj_Partida DAO llama al procedimiento almacenado call INSERTAR_PROY_PARTIDA(?,?,?,?,?,?,?,?)
- Luego, según se haya ejecutado la petición, la base de datos vuelve en una excepción, SQLException, lo que se convertirá en un 1 o 0.
- Finalmente, este 1 o 0 representa un mensaje en la vista para el administrador que será "Proyecto registrado correctamente", si la consulta fue exitosa.

Actualizar

- La clase V_Proj_Partida obtiene los datos del proyecto seleccionado y permite que el administrador edite esos datos.

- La clase C_Proj_Partida recibe los datos editados mediante la función actualizarDatos() y ejecuta la función update(proyecto) de la clase Proj_Partida DAO.
- La clase Proj_Partida DAO llama a la consulta: "update PROY_PARTIDA set NROVERSION=? where CODCIA=? AND CODPYTO=? AND INGEGR=? and CODPARTIDA=?"; Luego, según se haya ejecutado la petición, la base de datos vuelve en una excepción SQLException, lo que se convertirá en un 1 o 0.
- Finalmente, este 1 o 0 representa un mensaje en la vista para el administrador que será "Proyecto actualizado correctamente", si la consulta fue exitosa.

Eliminar

- La clase V_Proj_Partida obtiene el proyecto seleccionado para eliminar.
- La clase C_Proj_Partida recibe la solicitud de eliminación a través del método eliminarDatos() y ejecuta la función delete(proyecto) de la clase Proj_Partida DAO.
- La clase Proj_PartidaDAO llama a la consulta "DELETE from PROY_PARTIDA where CODCIA="+cia+" AND CODPARTIDA="+cod+" AND INGEGR="+tip+" AND CODPYTO="+pyto+" AND NROVERSION="+ver;
- Luego, según se haya ejecutado la petición, la base de datos vuelve en una excepción SQLException, lo que se convertirá en un 1 o 0.
- Finalmente, este 1 o 0 representa un mensaje en la vista para el administrador que será "Proyecto eliminado correctamente", si la consulta fue exitosa.

Consultas e instrucciones

Procedimiento almacenado INSERTAR_PROY_PARTIDA

```

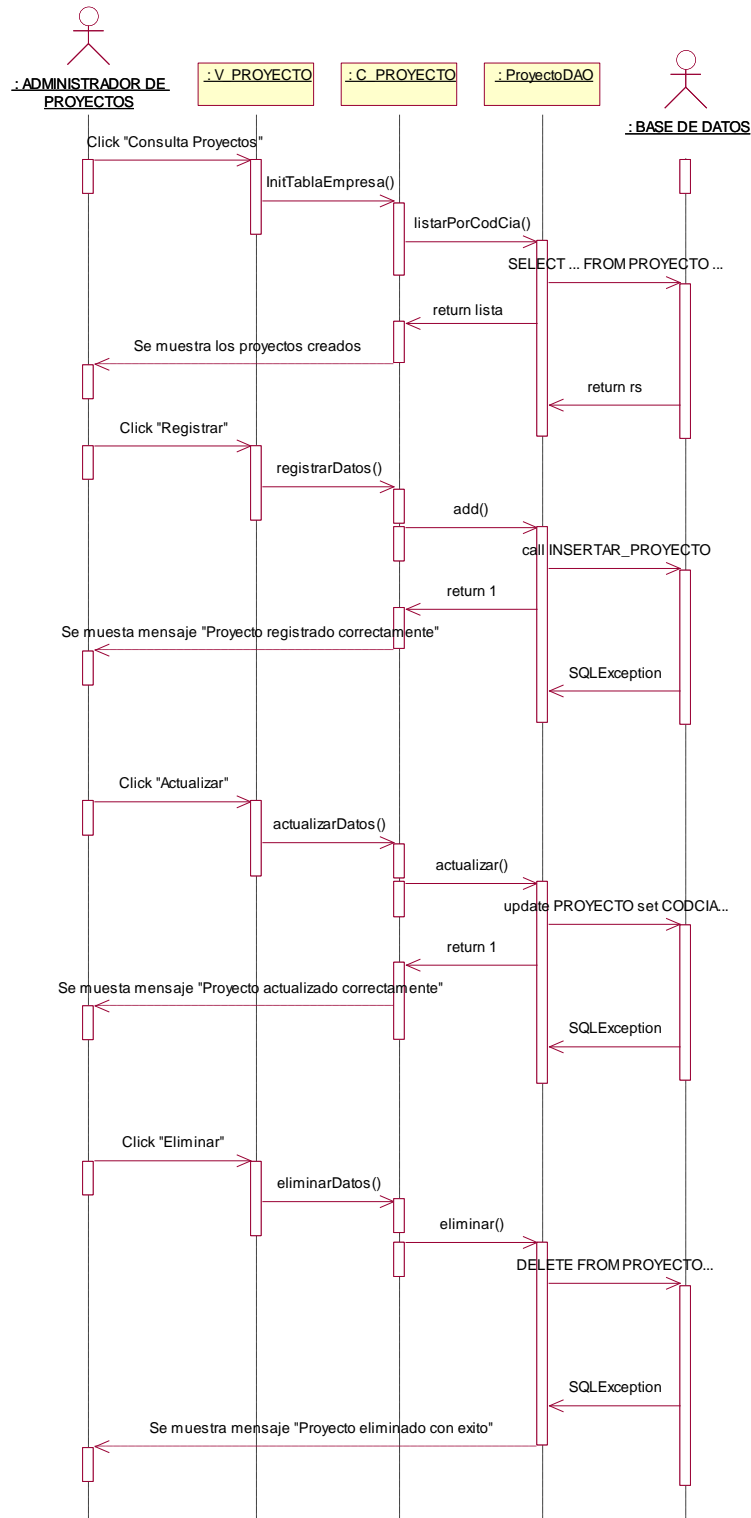
/*-----*/
create or replace NONEDITIONABLE PROCEDURE INSERTAR_PROY_PARTIDA(
    CODPYTO IN PROY_PARTIDA.CODPYTO%TYPE,
    NROVERSION IN PROY_PARTIDA.NROVERSION%TYPE,

```

```
CODCIA IN PROY_PARTIDA.CODCIA%TYPE,  
INGEGRE IN PROY_PARTIDA.IngEgr%TYPE,  
CODP IN PROY_PARTIDA.CODPARTIDA%TYPE,  
CODPAR IN PROY_PARTIDA.CODPARTIDAS%TYPE,  
TABE IN PROY_PARTIDA.TABESTADO%TYPE,  
CODE IN PROY_PARTIDA.CODESTADO%TYPE,  
VIG IN PROY_PARTIDA.VIGENTE%TYPE)  
  
IS  
  
BEGIN  
  
INSERT INTO PROY_PARTIDA  
VALUES(CODCIA,CODPYTO,NROVERSION,INGEGRE,CODP,CODPAR,'1',1,'1',TABE,CODE,VIG);  
  
END;  
  
/
```

- El procedimiento INSERTAR_PROY_PARTIDA se define para aceptar varios parámetros de entrada relacionados con un proyecto y sus partidas, y estará marcado como NONEDITIONABLE para asegurar su consistencia.
- Asimismo, inserta un registro en la tabla PROY_PARTIDA, este es un nuevo registro con los valores proporcionados por los parámetros de entrada.

Administrar proyectos



Pasos

- El administrador de proyectos da click en Consulta Proyectos
- La clase V_Proyecto inicia la interfaz con la función initTablaEmpresa() que pertenece a la clase C_Proyecto.
- La clase C_Proyecto se comunica con la clase ProyectoDAO mediante el método listarPorCodCia donde se genera la consulta "SELECT CODCIA, CODPYTO, NOMBPYTO, EMPLJEFEPROY, CIACONTRATA, CODCLIENTE, FECREG, ESTPYTO, FECESTADO, VALREFER, costoTotSinIGV, impIGV, COSTOTOTAL, observac, ANNOINI, ANNOFIN, LOGOPROY, VIGENTE " + "FROM PROYECTO WHERE CODCIA="+id+" ORDER BY CODCIA";
- Luego, la consulta se manda a realizar a la base de datos para mostrar todos los proyectos ya creadas en la compañía.

Registrar

- La V_Proyecto tiene un apartado donde el administrador de proyectos ingresa los datos de un nuevo proyecto que se va a registrar, luego hace click en Registrar
- La clase C_Proyecto obtiene los datos colocados por el administrador mediante la función registrarDatos() y ejecuta la función add() de la clase ProyectoDAO
- La clase ProyectoDAO llama al procedimiento almacenado call INSERTAR_PROYECTO(?,?,?,?,?,?,?,?,?,?,?,?,?) que en la base de datos está como la consulta INSERT INTO PROYECTO (CodCIA,CodPyto,NombPyto,EmplJefeProy,CodCia1,CiaContrata,CodCC,CodCliente,FlgEmpC onsortio,CodSNIP, FecReg,CodFase,CodNivel,CodFuncion,CodSituacion,NumInfor,NumInforEntrg,EstPyto,FecEsta do,ValRefer,CostoDirecto,CostoGGen,CostoFinan,ImpUtilidad, CostoTotSinIGV,ImpIGV,CostoTotal,CostoPenalid,CodDpto,CodProv,CodDist,FecViab,RutaDoc, AnnoIni,AnnoFin,CodObjC,LogoProy,TabEstado,CodEstado,Observac,Vigente)VALUES

```
(COD_CIA,SEC_PROYECTO.NEXTVAL,NOMPY,JEFE,-999,CIACONT,-999,CODCLI,'-','-
',FECRE,0,0,'-',0,0,0,ESTPYT,FECEST,VALREF,-999,-999,-999,-
999,COSTOTOTSIN,IGV,COSTOT,-999,'-','-',01-01-
2022','RUTA_DOC',ANNOIN,ANNOFI,0,LOGO,'-1','1',OBS,VIGENT);
```

- Luego, según se haya ejecutado la petición, la base de datos vuelve en una excepción, SQLException, lo que se convertirá en un 1 o 0.
- Finalmente, este 1 o 0 representa un mensaje en la vista para el administrador que será "Proyecto registrado correctamente", si la consulta fue exitosa.

Actualizar

- La clase V_Proyecto obtiene los datos del proyecto seleccionado y permite que el administrador edite esos datos.
- La clase C_Proyecto recibe los datos editados mediante la función actualizarDatos() y ejecuta la función update(proyecto) de la clase ProyectoDAO.
- La clase ProyectoDAO llama al procedimiento almacenado call ACTUALIZAR_PROYECTO(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?) que en la base de datos está como la consulta: UPDATE PROYECTO SET NombPyto = ?, EmplJefeProy = ?, CodCia1 = ?, CiaContrata = ?, CodCC = ?, CodCliente = ?, FlgEmpConsortio = ?, CodSNIP = ?, FecReg = ?, CodFase = ?, CodNivel = ?, CodFuncion = ?, CodSituacion = ?, NumInfor = ?, NumInforEntrg = ?, EstPyto = ?, FecEstado = ?, ValRefer = ?, CostoDirecto = ?, CostoGGen = ?, CostoFinan = ?, ImpUtilidad = ?, CostoTotSinIGV = ?, ImpIGV = ?, CostoTotal = ?, CostoPenalid = ?, CodDpto = ?, CodProv = ?, CodDist = ?, FecViab = ?, RutaDoc = ?, AnnoIni = ?, AnnoFin = ?, CodObjC = ?, LogoProy = ?, TabEstado = ?, CodEstado = ?, Observac = ?, Vigente = ? WHERE CodCIA = ? AND CodPyto = ?;
- Luego, según se haya ejecutado la petición, la base de datos vuelve en una excepción SQLException, lo que se convertirá en un 1 o 0.

- Finalmente, este 1 o 0 representa un mensaje en la vista para el administrador que será "Proyecto actualizado correctamente", si la consulta fue exitosa.

Eliminar

- La clase V_Proyecto obtiene el proyecto seleccionado para eliminar.
- La clase C_Proyecto recibe la solicitud de eliminación a través del método eliminarDatos() y ejecuta la función delete(proyecto) de la clase ProyectoDAO.
- La clase ProyectoDAO llama al procedimiento almacenado call ELIMINAR_PROYECTO(?) que en la base de datos está como la consulta: DELETE FROM PROYECTO WHERE CodPyto = ?;
- Luego, según se haya ejecutado la petición, la base de datos vuelve en una excepción SQLException, lo que se convertirá en un 1 o 0.
- Finalmente, este 1 o 0 representa un mensaje en la vista para el administrador que será "Proyecto eliminado correctamente", si la consulta fue exitosa.

Consultas e instrucciones

Procedimiento almacenado INSERTAR_PROYECTO

```

/*-----*/
create or replace PROCEDURE INSERTAR_PROYECTO(
    COD_CIA IN PROYECTO.CODCIA%TYPE,
    NOMPY IN PROYECTO.NOMBPYTO%TYPE,
    JEFE IN PROYECTO.EMPLJEFEPROY%TYPE,
    CIACONT IN PROYECTO.CIACONTRATA%TYPE,
    CODCLI IN PROYECTO.CODCLIENTE%TYPE,
    FECRE IN PROYECTO.FECREG%TYPE,
    ESTPYT IN PROYECTO.ESTPYTO%TYPE,
    FECEST IN PROYECTO.FECESTADO%TYPE,
    VALREF IN PROYECTO.VALREFER%TYPE,
    COSTOTOTSIN IN PROYECTO.COSTOTOTSINIGV%TYPE,

```

```

    IGV IN PROYECTO.IMPIGV%TYPE,
    COSTOT IN PROYECTO.COSTOTOTAL%TYPE,
    OBS IN PROYECTO.OBSERVAC%TYPE,
    ANNOIN IN PROYECTO.ANNOINI%TYPE,
    ANNOFI IN PROYECTO.ANNOFIN%TYPE,
    LOGO IN PROYECTO.LOGOPROY%TYPE,
    VIGENT IN PROYECTO.VIGENTE%TYPE)

IS

BEGIN

    INSERT INTO PROYECTO
(CodCIA,CodPyto,NombPyto,EmplJefeProy,CodCia1,CiaContrata,CodCC,CodCliente,FlgEmpCons
orcio,CodSNIP,FecReg,
CodFase,CodNivel,CodFuncion,CodSituacion,NumInfor,NumInforEntrg,EstPyto,FecEstado,Val
Refer,CostoDirecto,CostoGGen,CostoFinan,ImpUtilidad,
CostoTotSinIGV,ImpIGV,CostoTotal,CostoPenalid,CodDpto,CodProv,CodDist,FecViab,RutaDoc
,AnnoIni,AnnoFin,CodObjC,LogoProy,TabEstado,CodEstado,Observac,Vigente) VALUES
(COD_CIA,SEC_PROYECTO.NEXTVAL,NOMPY,JEFE,-999,CIACONT,-999,CODCLI,'-','-',
FECRE,0,0,'-',0,0,0,ESTPYT,FECEST,VALREF,-999,-999,-999,-999,
COSTOTOTSIN,IGV,COSTOT,-999,'-','-', '01-01-
2022','RUTA_DOC',ANNOIN,ANNOFI,0,LOGO,'-1','1',OBS,VIGENT);

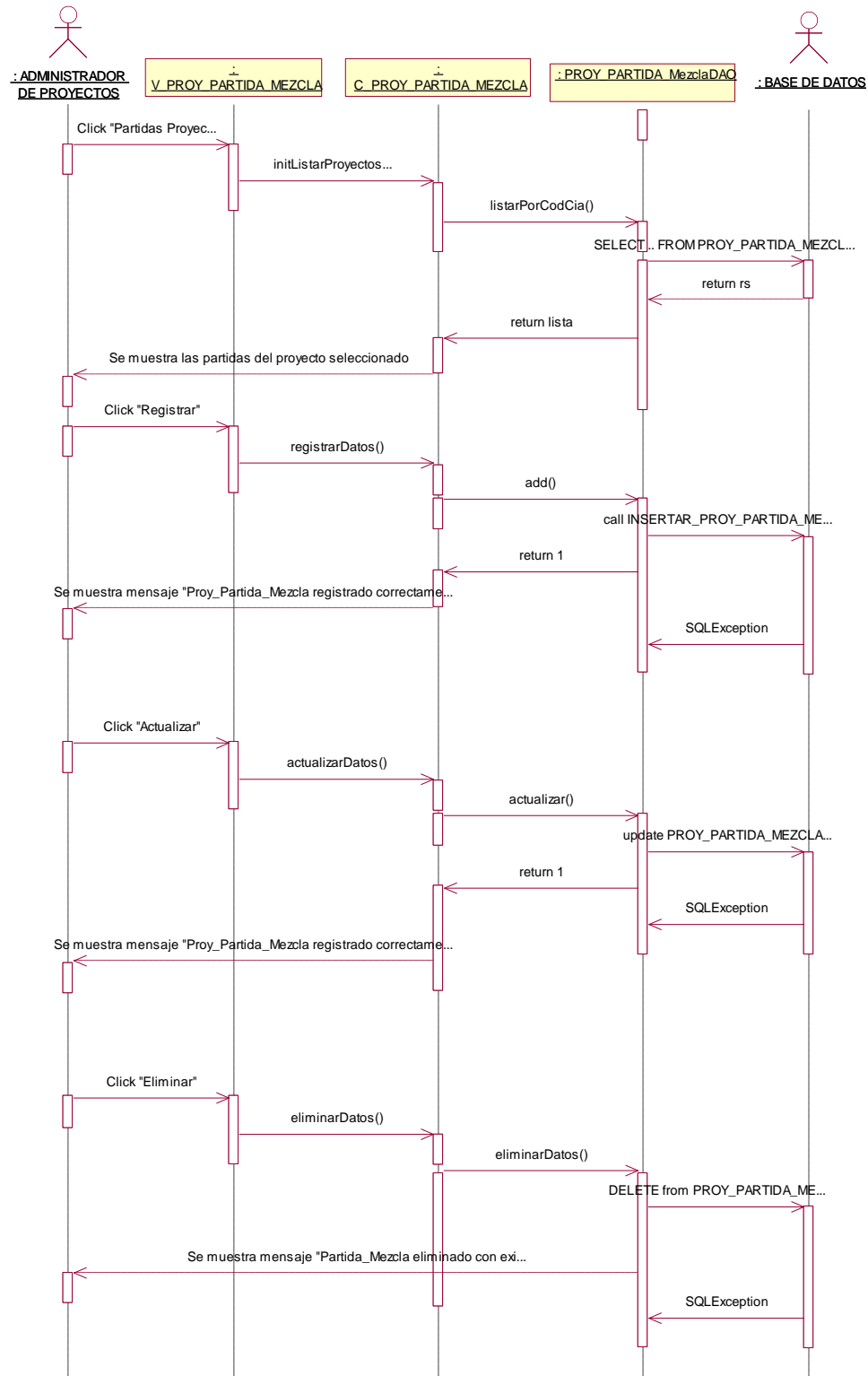
END;

/*-----*/

```

- El procedimiento almacenado empieza creando o reemplazando el procedimiento almacenado llamado INSERTAR_PROYECTO
- Luego declara los parámetros que almacenaran los valores del nuevo proyecto a registrar.

Administrar partidas padres de proyecto



Pasos

- El administrador de proyectos da click en Configuración Partidas Proyectos
- La clase V_Proj_Partida_Mezcla inicia la interfaz con la función `initListarProyectos_I()` que pertenece a la clase C_Proj_Partida_Mezcla.
- La clase C_Proj_Partida_Mezcla se comunica con la clase `Proy_Partida_MezclaDAO` mediante el método `listarPorCodCia` donde se genera la consulta "SELECT "+
"CodPyto,NroVersion,CodPartida,CodEstado,Vigente " + "FROM PROY_PARTIDA WHERE
CODCIA="+id+" AND INGEGR=""+"tip+" order by CODPYTO";
- Luego, la consulta se manda a realizar a la base de datos para mostrar todas las partidas con los proyectos relacionados.

Registrar

- La V_Proj_Partida_Mezcla tiene un apartado donde el administrador de proyectos agrega las partidas al proyecto, luego hace click en Registrar
- La clase C_Proj_Partida_Mezcla obtiene los datos colocados por el administrador mediante la función `registrarDatos()` y ejecuta la función `add()` de la clase `Proy_Partida_MezclaDAO`.
- La clase `Proy_Partida_MezclaDAO` llama al procedimiento almacenado `call`
`INSERTAR_PROY_PARTIDA_MEZCLA(?,?,?,?,?,?)` que en la base de datos está como la
consulta `INSERT INTO PROY_PARTIDA_MEZCLA`
`VALUES(CODC,CODPYTO,INEG,NROVERSION,R1.CODPARTIDA,SEC_PROY_PARTIDA`
`_MEZCLA_E.nextval,R1.PADCODPARTIDA,R1.TUNIMED,R1.EUNIMED,R1.NIVEL,R1.OR`
`DEN,R1.COSTOUNIT,CAN,V_total);`
- Luego, según se haya ejecutado la petición, la base de datos vuelve en una excepción,
`SQLException`, lo que se convertirá en un 1 o 0.
- Finalmente, este 1 o 0 representa un mensaje en la vista para el administrador que será "Proyecto
partida mezcla registrado correctamente", si la consulta fue exitosa.

Actualizar

- La clase V_Proj_Partida_Mezcla obtiene los datos de la partida mezcla seleccionada y permite que el administrador edite esos datos.
- La clase C_Proj_Partida_Mezcla recibe los datos editados mediante la función actualizarDatos() y ejecuta la función update(proy_partida_mezcla) de la clase Proj_Partida_MezclaDAO.
- La clase Proj_Partida_MezclaDAO llama al procedimiento almacenado call ACTUALIZAR_PROY_PARTIDA_MEZCLA(?,?,?,?,?,?,?,?,?) que en la base de datos está como la consulta: UPDATE PROY_PARTIDA_MEZCLA SET PADCODPARTIDA = ?, TUNIMED = ?, EUNIMED = ?, NIVEL = ?, ORDEN = ?, COSTOUNIT = ?, CAN = ?, V_TOTAL = ? WHERE CODC = ? AND CODPYTO = ? AND INEG = ? AND NROVERSION = ? AND CODPARTIDA = ?;
- Luego, según se haya ejecutado la petición, la base de datos vuelve en una excepción SQLException, lo que se convertirá en un 1 o 0.
- Finalmente, este 1 o 0 representa un mensaje en la vista para el administrador que será "Proyecto partida mezcla actualizada correctamente", si la consulta fue exitosa.

Eliminar

- La clase V_Proj_Partida_Mezcla obtiene la partida mezcla seleccionada para eliminar.
- La clase C_Proj_Partida_Mezcla recibe la solicitud de eliminación a través del método eliminarDatos() y ejecuta la función delete(proy_partida_mezcla) de la clase Proj_Partida_MezclaDAO.
- La clase Proj_Partida_MezclaDAO llama al procedimiento almacenado call ELIMINAR_PROY_PARTIDA_MEZCLA(?,?,?,?,?) que en la base de datos está como la consulta: DELETE FROM PROY_PARTIDA_MEZCLA WHERE CODC = ? AND CODPYTO = ? AND INEG = ? AND NROVERSION = ? AND CODPARTIDA = ?;
- Luego, según se haya ejecutado la petición, la base de datos vuelve en una excepción SQLException, lo que se convertirá en un 1 o 0.

- Finalmente, este 1 o 0 representa un mensaje en la vista para el administrador que será "Proyecto partida mezcla eliminada correctamente", si la consulta fue exitosa.

Consultas e instrucciones

Procedimiento almacenado INSERTAR_PROY_PARTIDA_MEZCLA

```

/*-----*/
create or replace NONEDITIONABLE PROCEDURE INSERTAR_PROY_PARTIDA_MEZCLA(
    CODC IN PROY_PARTIDA_MEZCLA.CODCIA%TYPE,
    CODPYTO IN PROY_PARTIDA_MEZCLA.CODPYTO%TYPE,
    NROVERSION IN PROY_PARTIDA_MEZCLA.NROVERSION%TYPE,
    PADCOD IN PROY_PARTIDA_MEZCLA.PADCODPARTIDA%TYPE,
    INEG IN PROY_PARTIDA_MEZCLA.IngEgr%TYPE,
    CAN IN PROY_PARTIDA_MEZCLA.CANT%TYPE)
AS
CURSOR cpm IS
SELECT CODPARTIDA, PADCODPARTIDA, TUNIMED, EUNIMED, COSTOUNIT, NIVEL, ORDEN
FROM PARTIDA_MEZCLA WHERE CODCIA=CODC AND PADCODPARTIDA=PADCOD AND INEGR=INEG;
CURSOR cpm2 IS
SELECT CODPARTIDA, PADCODPARTIDA, TUNIMED, EUNIMED, COSTOUNIT, NIVEL, ORDEN
FROM PARTIDA_MEZCLA WHERE CODCIA=CODC AND CODPARTIDA=PADCOD AND PADCODPARTIDA=0 AND
INEGR=INEG;
V_total PROY_PARTIDA_MEZCLA.COSTOTOT%TYPE;
BEGIN
IF(INEG = 'E') THEN
    FOR R1 IN CPM2 LOOP
        V_total := CAN*R1.COSTOUNIT;
        INSERT INTO PROY_PARTIDA_MEZCLA VALUES(CODC, CODPYTO, INEG, NROVERSION, R1.CODPARTIDA,
        SEC_PROY_PARTIDA_MEZCLA_E.nextval, R1.PADCODPARTIDA, R1.TUNIMED, R1.EUNIMED,

```

```
R1.NIVEL,R1.ORDEN,R1.COSTOUNIT,CAN,V_total);

END LOOP;


FOR R2 IN CPM LOOP

V_total := CAN*R2.COSTOUNIT;

INSERT INTO PROY_PARTIDA_MEZCLA VALUES(CODC,CODPYTO,INEG,NROVERSION,R2.CODPARTIDA,
SEC_PROY_PARTIDA_MEZCLA_E.nextval,R2.PADCODPARTIDA,R2.TUNIMED,R2.EUNIMED,
R2.NIVEL,R2.ORDEN,R2.COSTOUNIT,CAN,V_total);

END LOOP;

END IF;

IF(INEG = 'I') THEN

FOR R3 IN CPM2 LOOP

V_total := CAN*R3.COSTOUNIT;

INSERT INTO PROY_PARTIDA_MEZCLA VALUES(CODC,CODPYTO,INEG,NROVERSION,R3.CODPARTIDA,
SEC_PROY_PARTIDA_MEZCLA_I.nextval,R3.PADCODPARTIDA,R3.TUNIMED,R3.EUNIMED,
R3.NIVEL,R3.ORDEN,R3.COSTOUNIT,CAN,V_total);

END LOOP;


FOR R4 IN CPM LOOP

V_total := CAN*R4.COSTOUNIT;

INSERT INTO PROY_PARTIDA_MEZCLA VALUES(CODC,CODPYTO,INEG,NROVERSION,R4.CODPARTIDA,
SEC_PROY_PARTIDA_MEZCLA_I.nextval,R4.PADCODPARTIDA,R4.TUNIMED,R4.EUNIMED,
R4.NIVEL,R4.ORDEN,R4.COSTOUNIT,CAN,V_total);

END LOOP;

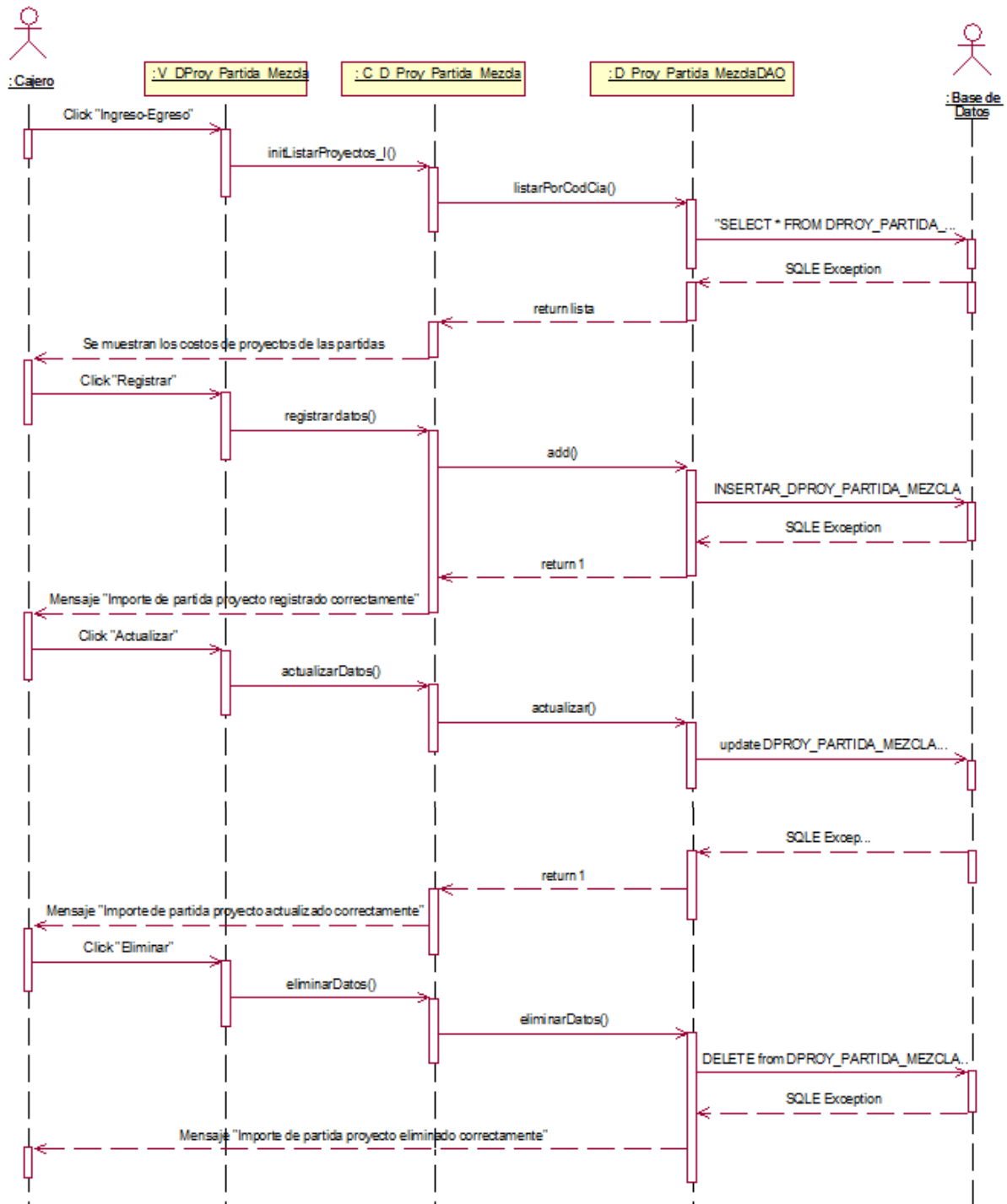
END IF;

END;

/*-----*/
```

- El procedimiento almacenado empieza creando o reemplazando el procedimiento almacenado llamado INSERTAR_PROY_PARTIDA_MEZCLA
- Luego declara los parámetros para registrar la partida relacionada al proyecto seleccionado.

Administrar costos de partidas de proyectos



Pasos

- El administrador de proyectos da click en la ventana “ingreso- egreso”
- La clase V_DProy_Partida_Mezcla inicia la interfaz con la función `initListarProyectos_I()` que pertenece a la clase C_Proj_Partida_Mezcla.
- La clase C_Proj_Partida_Mezcla se comunica con la clase `Proy_Partida_MezclaDAO` mediante el método `listarPorCodCia` donde se genera la consulta `"SELECT * FROM DPROY_PARTIDA_MEZCLA WHERE codcia="+id+" AND ingegr="+tip+" AND CODPYTO="+pyto;`
- Luego, la consulta se manda a realizar a la base de datos para mostrar todas las partidas con los proyectos relacionados.

Registrar

- La clase V_DProy_Partida_Mezcla tiene una opción en donde el cajero agrega los costos de las partidas al proyecto, el usuario hace click en “Registrar”.
- La clase C_DProy_Partida_Mezcla obtiene los datos colocados por el cajero mediante la función `registrarDatos()` y ejecuta la función `add()` de la clase `DProy_Partida_MezclaDAO`.
- La clase `DProy_Partida_MezclaDAO` llama al procedimiento almacenado `call INSERTAR_COMP_PAGODET(?,?,?,?,?,?,?,?,?)`
- Luego, según se haya ejecutado la petición, la base de datos vuelve en una excepción, `SQLException`, lo que se convertirá en un 1 o 0.
- Finalmente, este 1 o 0 representa un mensaje en la vista para el cajero que será "Importe de proyecto partida mezcla registrado correctamente", si la consulta fue exitosa.

Actualizar

- La clase V_DProy_Partida_Mezcla obtiene los datos de los costos de la partida mezcla seleccionada y permite que el cajero pueda editar estos datos.
- La clase C_DProy_Partida_Mezcla recibe los datos editados mediante la función `actualizarDatos()` y ejecuta la función `update()` de la clase `DProy_Partida_MezclaDAO`.

- La clase DProy_Partida_MezclaDAO llama al procedimiento "update DPROY_PARTIDA_MEZCLA set CODCIA=?,CODPYTO=?,INGEGR=?,NROVERSION=?,CODPARTIDA=?,CORR=?,TDESEMBOLSO=?,EDESEMBOLSO=?,NROPAGO=?,TCOMPPAGO=?,ECOMPPAGO=?,FECDESEMBOLSO=?,IMPDESEMBNETO=?,IMPDESEMBIGV=?,IMPDESEMBTOT=?,SEMILLA=? where SEC=?" ;Luego, según se haya ejecutado la petición, la base de datos vuelve en una excepción SQLException, lo que se convertirá en un 1 o 0.
- Finalmente, este 1 o 0 representa un mensaje en la vista para el cajero que será "Importe del proyecto partida mezcla actualizada correctamente", si la consulta fue exitosa.

Eliminar

- La clase V_Proj_Partida_Mezcla obtiene los costos de la partida mezcla seleccionada para eliminar.
- La clase C_Proj_Partida_Mezcla recibe la solicitud de eliminación a través del método eliminarDatos() y ejecuta la función delete() de la clase DProy_Partida_MezclaDAO.
- La clase DProy_Partida_MezclaDAO llama al procedimiento almacenado "DELETE from DPROY_PARTIDA_MEZCLA where SEC="+id
- Finalmente, el sistema mostrará al cajero el siguiente mensaje "Importe del proyecto partida mezcla eliminada correctamente", si la consulta fue exitosa.

Consultas e instrucciones

Procedimiento almacenado INSERTAR_ DPROY_PARTIDA_MEZCLA

/*-----*/

```
create or replace PROCEDURE INSERTAR_DPROY_PARTIDA_MEZCLA(
  CODCIA IN DPROY_PARTIDA_MEZCLA.CODCIA%TYPE,
  CODPYTO IN DPROY_PARTIDA_MEZCLA.CODPYTO%TYPE,
  INGEGR IN DPROY_PARTIDA_MEZCLA.INGEGR%TYPE,
  NROVERSION IN DPROY_PARTIDA_MEZCLA.NROVERSION%TYPE,
```

```

CODPART IN DPROY_PARTIDA_MEZCLA.CODPARTIDA%TYPE,
CORR IN DPROY_PARTIDA_MEZCLA.CORR%TYPE,
EDESEMB IN DPROY_PARTIDA_MEZCLA.EDESEMBOLSO%TYPE,
ECPAGO IN DPROY_PARTIDA_MEZCLA.ECOMPPAGO%TYPE,
FECDESEMB IN DPROY_PARTIDA_MEZCLA.FECDESEMBOLSO%TYPE,
IMPDESEMNETO IN DPROY_PARTIDA_MEZCLA.IMPDESEMBNETO%TYPE,
IMPDESEMIGV IN DPROY_PARTIDA_MEZCLA.IMPDESEMBIGV%TYPE,
IMPDESEMTOT IN DPROY_PARTIDA_MEZCLA.IMPDESEMTOT%TYPE,
SEMI IN DPROY_PARTIDA_MEZCLA.SEMILLA%TYPE,
REPETICION IN DPROY_PARTIDA_MEZCLA.CODCIA%TYPE)
IS
BEGIN
IF(INGEGR = 'E') THEN
IF(REPETICION = 0) THEN
IF(EDESEMB = 1) THEN
INSERT INTO DPROY_PARTIDA_MEZCLA VALUES
(CODCIA,CODPYTO,INGEGR,NROVERSION,CODPART,CORR,
SEC_DPROY_PARTIDA_MEZCLA_E.NEXTVAL,3,EDESEMB,SEC_DPROY_PARTIDA_MEZCLA_ADELANTO.NEXTV
AL,4,
ECPAGO,FECDESEMB,IMPDESEMNETO,IMPDESEMIGV,IMPDESEMTOT,SEC_DPROY_PARTIDA_MEZCLA_SEMILL
A_E.NEXTVAL);
ELSE
INSERT INTO DPROY_PARTIDA_MEZCLA VALUES
(CODCIA,CODPYTO,INGEGR,NROVERSION,CODPART,CORR,
SEC_DPROY_PARTIDA_MEZCLA_E.NEXTVAL,3,EDESEMB,SEC_DPROY_PARTIDA_MEZCLA_PAGO.NEXTVAL,4,
ECPAGO,FECDESEMB,IMPDESEMNETO,IMPDESEMIGV,IMPDESEMTOT,SEC_DPROY_PARTIDA_MEZCLA_SEMILL
A_E.NEXTVAL);
END IF;
ELSE
IF(EDESEMB = 1) THEN
INSERT INTO DPROY_PARTIDA_MEZCLA VALUES

```

```
(CODCIA,CODPYTO,INGEGR,NROVERSION,CODPART,CORR,
SEC_DPROY_PARTIDA_MEZCLA_E.NEXTVAL,3,EDESEMB,SEC_DPROY_PARTIDA_MEZCLA_ADELANTO.NEXTV
AL,4,
ECPAGO,FECDESEMB,IMPDESEMNETO,IMPDESEMIGV,IMPDESEMTOT,SEMI);

ELSE

INSERT INTO DPROY_PARTIDA_MEZCLA VALUES

(CODCIA,CODPYTO,INGEGR,NROVERSION,CODPART,CORR,
SEC_DPROY_PARTIDA_MEZCLA_E.NEXTVAL,3,EDESEMB,SEC_DPROY_PARTIDA_MEZCLA_PAGO.NEXTVAL,4,
ECPAGO,FECDESEMB,IMPDESEMNETO,IMPDESEMIGV,IMPDESEMTOT,SEMI);

END IF;

END IF;

END IF;

IF(INGEGR = 'T') THEN

IF(REPETICION = 0) THEN

IF(EDESEMB = 1) THEN

INSERT INTO DPROY_PARTIDA_MEZCLA VALUES

(CODCIA,CODPYTO,INGEGR,NROVERSION,CODPART,CORR,
SEC_DPROY_PARTIDA_MEZCLA_I.NEXTVAL,3,EDESEMB,SEC_DPROY_PARTIDA_MEZCLA_ADELANTO.NEXTVA
L,4,
ECPAGO,FECDESEMB,IMPDESEMNETO,IMPDESEMIGV,IMPDESEMTOT,SEC_DPROY_PARTIDA_MEZCLA_SEMILL
A_I.NEXTVAL);

ELSE

INSERT INTO DPROY_PARTIDA_MEZCLA VALUES

(CODCIA,CODPYTO,INGEGR,NROVERSION,CODPART,CORR,
SEC_DPROY_PARTIDA_MEZCLA_I.NEXTVAL,3,EDESEMB,SEC_DPROY_PARTIDA_MEZCLA_PAGO.NEXTVAL,4,
ECPAGO,FECDESEMB,IMPDESEMNETO,IMPDESEMIGV,IMPDESEMTOT,SEC_DPROY_PARTIDA_MEZCLA_SEMILL
A_I.NEXTVAL);

END IF;

ELSE

IF(EDESEMB = 1) THEN

INSERT INTO DPROY_PARTIDA_MEZCLA VALUES
```

```

(CODCIA,CODPYTO,INGEGR,NROVERSION,CODPART,CORR,

SEC_DPROY_PARTIDA_MEZCLA_I.NEXTVAL,3,EDESEMB,SEC_DPROY_PARTIDA_MEZCLA_ADELANTO.NEXTVA
L,4,

ECPAGO,FECDESEMB,IMPDESEMNETO,IMPDESEMIGV,IMPDESEMTOT,SEMI);

ELSE

INSERT INTO DPROY_PARTIDA_MEZCLA VALUES

(CODCIA,CODPYTO,INGEGR,NROVERSION,CODPART,CORR,

SEC_DPROY_PARTIDA_MEZCLA_I.NEXTVAL,3,EDESEMB,SEC_DPROY_PARTIDA_MEZCLA_PAGO.NEXTVAL,4,

ECPAGO,FECDESEMB,IMPDESEMNETO,IMPDESEMIGV,IMPDESEMTOT,SEMI);

END IF;

END IF;

END IF;

UPDATE_FLUJOCAJA_DET_UNALINEA_Y_GLOBAL(CODCIA,CODPYTO,INGEGR,CODPART,IMPDESEMTOT,FECDE
SEMB,'P');

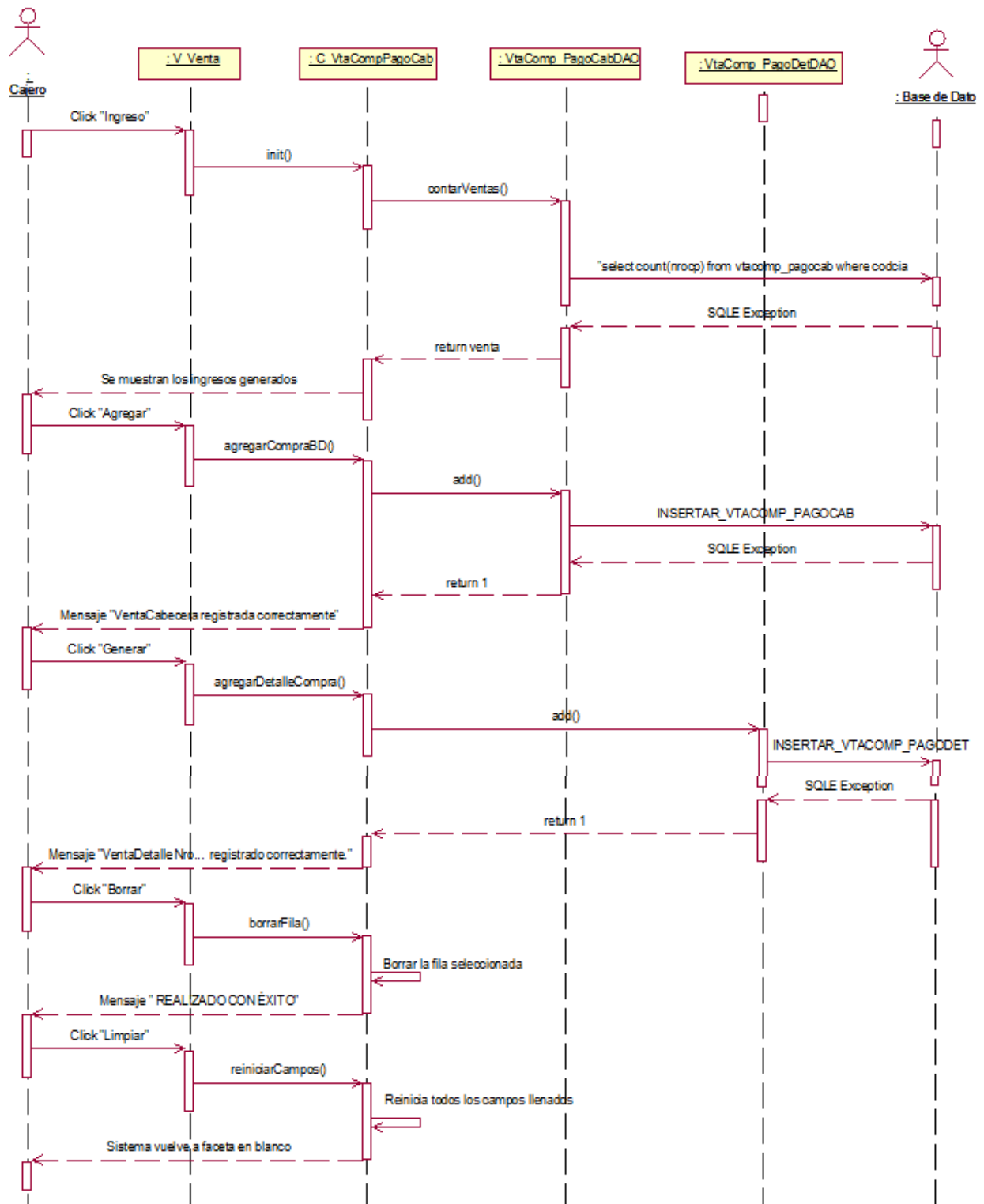
END;

/

```

- Este procedimiento empieza insertando nuevos registros con valores específicos y generados a partir de secuencias y dependiendo de los valores de los parametros de entrada como (INGEGR, EDESEMB, y REPETICION)
- Al final del procedimiento, llama a la función UPDATE_FLUJOCAJA_DET_UNALINEA_Y_GLOBAL para actualizar los detalles del flujo de caja relacionados con el proyecto., utilizando parámetros de entrada y contando con el desembolso total.

Administrar ventas



Pasos

- El cajero se dirige a la opción de “Ventas” en el menú y da click en “ingresos”
- La clase V_ VtaCompPagoCab inicia la interfaz con la función `init()` que pertenece a la clase C_ VtaCompPagoCab.
- La clase C_ VtaCompPagoCab se comunica con la clase VtaCompPagoCabDAO mediante el método `ContarVentas()` donde se genera la consulta `"select count(nrocp) from vtacomp_pagocab where codcia="+varCodCiaGlobalDeLogin;` la consulta se manda a realizar a la base de datos para mostrar todas las ventas que ya estaban registradas en el sistema de la compañía

Agregar

- La clase V_ VtaCompPagoCab tiene una opción en donde el cajero ingresa los datos de principales de una venta realizada y que desea registrarla, a eso, es que hará click en Agregar.
- La clase C_ VtaCompPagoCab obtiene los datos colocados por el cajero mediante la función `agregarCompraBD()` y ejecuta la función `add()` de la clase VtaCompPagoCabDAO
- La clase VtaCompPagoCabDAO llama al procedimiento almacenado `call INSERTAR_VTACOMP_PAGOCAB (?, ?, ?, ?, ?, ?, ?)`
- Luego, la base de datos vuelve en una excepción, `SQLException`, lo que se convertirá en un 1 o 0, según lo que haya ejecutado
- Para finalizar, este 1 o 0 representa un mensaje en la vista para el administrador que será "VentaCabecera registrada correctamente", si la consulta fue exitosa.

Generar

- La clase V_ VtaCompPagoCab también tiene otra opción en donde el cajero puede especificar todos los datos de una venta realizada y que desea registrarla, para esto, hará click en “Generar”.
- La clase C_ VtaCompPagoCab recibe los datos editados mediante la función `agregarDetalleCompra()` y ejecuta la función `add()` de la clase VtaCompPagoCabDetDAO.

- La clase VtaCompPagoCabDetDAO llama al procedimiento almacenado call
INSERTAR_VTACOMP_PAGODET (?, ?, ?, ?, ?, ?, ?); Luego, según se haya ejecutado la
petición, la base de datos vuelve en una excepción SQLException, lo que se convertirá en un 1 o
0.
- Finalmente, este 1 o 0 representa un mensaje en la vista para el administrador que será
"VentaDetalle Nro... registrado correctamente." si la consulta fue exitosa.

Borrar

- El cajero selecciona la venta que desea eliminar y da click en “Borrar”
- La clase V_VtaCompPagoCab obtiene los datos de la venta a eliminar mediante la función
BorrarFila().
- Esta función se ejecuta en la interfaz y luego arroja un mensaje en la ventana para el cajero que
será "REALIZADO CON EXITO", si no hubo algún error.

Limpiar

- Si el cajero desea limpiar todos los datos que ha registrado en la interfaz; dará click en “Limpiar”.
- La clase V_VtaCompPagoCab obtiene los datos de la venta a eliminar mediante la función
reiniciarCampo().
- Esta función se ejecuta en la interfaz y luego el sistema volverá a tener un registro vacío en esta
misma interfaz.

Consultas e instrucciones

Procedimiento almacenado INSERTAR_VTACOMP_PAGOCAB

/*-----*/

create or replace NONEDITIONABLE PROCEDURE INSERTAR_VTACOMP_PAGOCAB(

CODCIA IN VTACOMP_PAGOCAB.codcia%TYPE,

NROCP IN VTACOMP_PAGOCAB.nrocp%TYPE,

CODPYTO IN VTACOMP_PAGOCAB.codpyto%TYPE,

CODCLIENTE IN VTACOMP_PAGOCAB.codcliente%TYPE,

```
--NROPAGO IN VTACOMP_PAGOCAB.nropago%TYPE,
TCOMPPAGO IN VTACOMP_PAGOCAB.tcomppago%TYPE,
ECOMPPAGO IN VTACOMP_PAGOCAB.ecomppago%TYPE,
FECCP IN VTACOMP_PAGOCAB.feccp%TYPE,
TMONEDA IN VTACOMP_PAGOCAB.tmoneda%TYPE,
EMONEDA IN VTACOMP_PAGOCAB.emoneda%TYPE,
TIPCAMBIO IN VTACOMP_PAGOCAB.tipcambio%TYPE,
IMPMO IN VTACOMP_PAGOCAB.impmo%TYPE,
IMPNETOMN IN VTACOMP_PAGOCAB.impnetomn%TYPE,
IMPIGVMN IN VTACOMP_PAGOCAB.impigvmn%TYPE,
IMPTOTALMN IN VTACOMP_PAGOCAB.imptotalmn%TYPE,
FOTOCP IN VTACOMP_PAGOCAB.fotocp%TYPE,
FOTOABONO IN VTACOMP_PAGOCAB.fotoabono%TYPE,
FECABONO IN VTACOMP_PAGOCAB.fecabono%TYPE,
DESABONO IN VTACOMP_PAGOCAB.desabono%TYPE,
SEMILLA IN VTACOMP_PAGOCAB.semilla%TYPE,
TABESTADO IN VTACOMP_PAGOCAB.tabestado%TYPE,
CODESTADO IN VTACOMP_PAGOCAB.codeestado%TYPE)
IS
BEGIN
INSERT INTO VTACOMP_PAGOCAB
VALUES(CODCIA,NROCP,CODPYTO,CODCLIENTE,SEC_NRO_PAGO_VTA.nextval,TCOMPPAGO,ECOMP
PAGO,FECCP,TMONEDA,EMONEDA,TIPCAMBIO,IMPMO,IMPNETOMN,IMPIGVMN,IMPTOTALMN,FOT
OCP,FOTOABONO,FECABONO,DESABONO,SEMILLA,TABESTADO,CODESTADO);
END;

/*-----*/
```

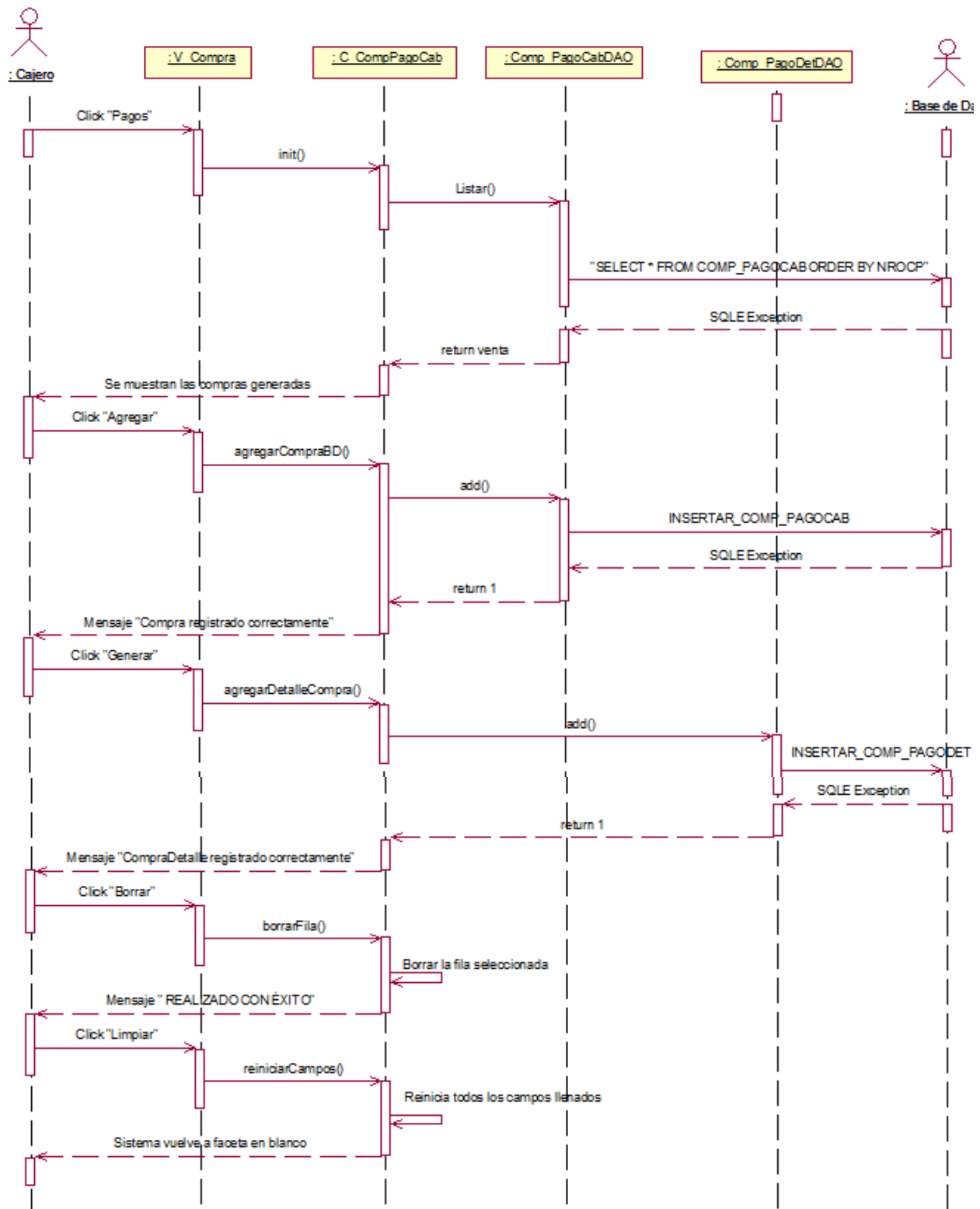
Procedimiento almacenado INSERTAR_VTACOMP_PAGOCABDET

```
/*-----*/
```

```
create or replace NONEDITIONABLE PROCEDURE INSERTAR_VTACOMP_PAGODET(
    CODCIA IN VTACOMP_PAGODET.codcia%TYPE,
    NRO_CP IN VTACOMP_PAGODET.nrocp%TYPE,
    SEC IN VTACOMP_PAGODET.sec%TYPE,
    INGEGR IN VTACOMP_PAGODET.ingegr%TYPE,
    CODPARTIDA IN VTACOMP_PAGODET.codpartida%TYPE,
    IMPNETOMN IN VTACOMP_PAGODET.impnetomn%TYPE,
    IMPIGVMN IN VTACOMP_PAGODET.impigvmn%TYPE,
    IMPTOTALMN IN VTACOMP_PAGODET.imptotalmn%TYPE,
    SEMILL IN VTACOMP_PAGODET.semilla%TYPE,
    COD_PYTO_AUXILIAR IN PROYECTO.codpyto%type)
IS
FECHA_DE_CP DATE;
BEGIN
    INSERT INTO VTACOMP_PAGODET
VALUES(CODCIA,NRO_CP,SEC,INGEGR,CODPARTIDA,IMPNETOMN,IMPIGVMN,IMPTOTALMN,SEMILL);
    SELECT FECCP INTO FECHA_DE_CP FROM VTACOMP_PAGOCAB WHERE NROCP=NRO_CP;

    UPDATE_FLUJOCAJA_DET_UNALINEA_Y_GLOBAL(CODCIA,COD_PYTO_AUXILIAR,INGEGR,CODPARTIDA,IMPTOTALMN,FECHA_DE_CP,'R');
END;
/
```

Administrar egresos



Pasos

- El cajero se dirige a la opción de “Ventas” en el menú y da click en “pagos”
- La clase V_CompPagoCab inicia la interfaz con la función `init()` que pertenece a la clase C_CompPagoCab.
- La clase C_CompPagoCab se comunica con la clase CompPagoCabDAO mediante el método `Listar()` donde se genera la consulta "SELECT * FROM COMP_PAGOCAB ORDER BY NROCP"; la consulta se manda a realizar a la base de datos para mostrar todas las compras que ya estaban registradas en el sistema de la compañía

Agregar

- La clase V_CompPagoCab tiene una opción en donde el cajero ingresa los datos de principales de una compra realizada y que desea registrarla, a eso, es que hará click en Agregar.
- La clase C_CompPagoCab obtiene los datos colocados por el cajero mediante la función `agregarCompraBD()` y ejecuta la función `add()` de la clase CompPagoCabDAO
- La clase CompPagoCabDAO llama al procedimiento almacenado `call INSERTAR_COMP_PAGOCAB (?, ?, ?, ?, ?, ?, ?, ?)`
- Luego, la base de datos vuelve en una excepción, `SQLException`, lo que se convertirá en un 1 o 0, según lo que haya ejecutado
- Para finalizar, este 1 o 0 representa un mensaje en la vista para el administrador que será " Compra registrado correctamente ", si la consulta fue exitosa.

Generar

- La clase V_CompPagoCab también tiene otra opción en donde el cajero puede especificar todos los datos de una compra realizada y que desea registrarla, para esto, hará click en “Generar”.
- La clase C_CompPagoCab recibe los datos editados mediante la función `agregarDetalleCompra()` y ejecuta la función `add()` de la clase CompPagoCabDetDAO.

- La clase CompPagoCabDetDAO llama al procedimiento almacenado call
INSERTAR_COMP_PAGODET (?, ?, ?, ?, ?, ?, ?, ?); Luego, según se haya ejecutado la petición, la base de datos vuelve en una excepción SQLException, lo que se convertirá en un 1 o 0.
- Finalmente, este 1 o 0 representa un mensaje en la vista para el administrador que será " " "CompraDetalle registrado correctamente." si la consulta fue exitosa.

Borrar

- El cajero selecciona la compra que desea eliminar y da click en "Borrar"
- La clase V_CompPagoCab obtiene los datos de la compra a eliminar mediante la función BorrarFila().
- Esta función se ejecuta en la interfaz y luego arroja un mensaje en la ventana para el cajero que será "REALIZADO CON EXITO", si no hubo algún error.

Limpiar

- Si el cajero desea limpiar todos los datos que ha registrado en la interfaz; dará click en "Limpiar".
- La clase V_CompPagoCab obtiene los datos de la compra que se desea eliminar mediante la función reiniciarCampo().
- Esta función se ejecuta en la interfaz y luego el sistema volverá a tener un registro vacío en esta misma interfaz.

Consultas e instrucciones

Procedimiento almacenado INSERTAR_COMP_PAGOCAB

/*-----*/

create or replace NONEDITIONABLE PROCEDURE INSERTAR_COMP_PAGOCAB(

CODCIA IN comp_pagocab.codcia%TYPE,

CODPROVEEDOR IN comp_pagocab.codproveedor%TYPE,

NROCP IN comp_pagocab.nrocp%TYPE,

CODPYTO IN comp_pagocab.codpyto%TYPE,

NROPAGO IN comp_pagocab.nropago%TYPE,

```
TCOMPPAGO IN comp_pagocab.tcomp_pago%TYPE,
ECOMPPAGO IN comp_pagocab.ecomp_pago%TYPE,
FECCP IN comp_pagocab.feccp%TYPE,
TMONEDA IN comp_pagocab.tmoneda%TYPE,
EMONEDA IN comp_pagocab.emoneda%TYPE,
TIPCAMBIO IN comp_pagocab.tipcambio%TYPE,
IMPMO IN comp_pagocab.impmo%TYPE,
IMPNETOMN IN comp_pagocab.impnetomn%TYPE,
IMPIGVMN IN comp_pagocab.impigvmn%TYPE,
IMPTOTALMN IN comp_pagocab.imptotalmn%TYPE,
FOTOCAP IN comp_pagocab.fotocap%TYPE,
FOTOABONO IN comp_pagocab.fotoabono%TYPE,
FECABONO IN comp_pagocab.fecabono%TYPE,
DESABONO IN comp_pagocab.desabono%TYPE,
SEMILLA IN comp_pagocab.semilla%TYPE,
TABESTADO IN comp_pagocab.tabestado%TYPE,
CODESTADO IN comp_pagocab.codeestado%TYPE)

IS
BEGIN
INSERT INTO COMP_PAGOCAB
VALUES(CODCIA,CODPROVEEDOR,NROCP,CODPYTO,NROPAGO,TCOMPPAGO,ECOMPPAGO,FECCP,T
MONEDA,EMONEDA,TIPCAMBIO,IMPMO,IMPNETOMN,IMPIGVMN,IMPTOTALMN,FOTOCAP,FOTOABO
NO,FECABONO,DESABONO,
        SEMILLA,TABESTADO,CODESTADO);

END;

/*-----*/
```

Procedimiento almacenado INSERTAR_COMP_PAGOCABDET

```

/*-----*/

create or replace NONEDITIONABLE PROCEDURE INSERTAR_COMP_PAGODET(

    CODCIA IN comp_pagodet.codcia%TYPE,

    CODPROVEEDOR IN comp_pagodet.codproveedor%TYPE,

    NRO_CP IN comp_pagodet.nrocp%TYPE,

    SEC IN comp_pagodet.sec%TYPE,

    INGEGR IN comp_pagodet.ingegr%TYPE,

    CODPARTIDA IN comp_pagodet.codpartida%TYPE,

    IMPNETOMN IN comp_pagodet.impnetomn%TYPE,

    IMPIGVMN IN comp_pagodet.impigvmn%TYPE,

    IMPTOTALMN IN comp_pagodet.imptotalmn%TYPE,

    SEMILL IN comp_pagocab.semilla%TYPE,

    COD_PYTO_AUXILIAR IN PROYECTO.codpyto%type)

IS

FECHA_DE_CP DATE;

BEGIN

    INSERT INTO COMP_PAGODET

VALUES(CODCIA,CODPROVEEDOR,NRO_CP,SEC,INGEGR,CODPARTIDA,IMPNETOMN,IMPIGVMN,IMPTOTALMN,

SEMILL);

    SELECT FECCP INTO FECHA_DE_CP FROM COMP_PAGOCAB WHERE NROCP=NRO_CP;

UPDATE_FLUJOCAJA_DET_UNALINEA_Y_GLOBAL(CODCIA,COD_PYTO_AUXILIAR,INGEGR,CODPARTIDA,IMPT

OTALMN,FECHA_DE_CP,'R');

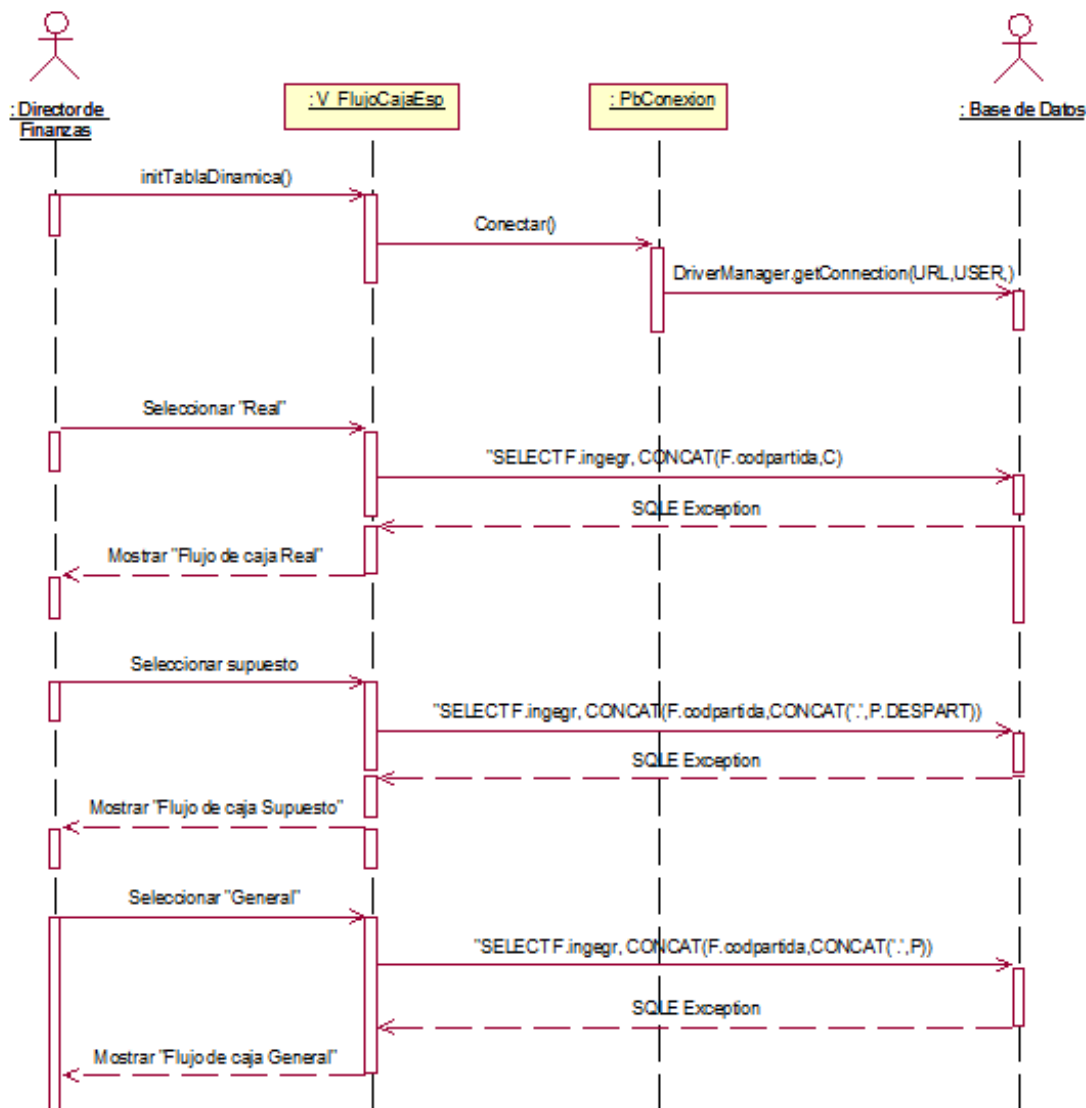
END;

/

/*-----*/

```


Mostrar flujo de caja



Pasos

- El director de finanzas hace click en Flujo de Caja.
- La clase V_FlujoCajaEsp inicia una vista dinámica con la función `initTablaDinamica()` en la cual se mostrará un espacio para buscar los datos relacionados a las ventas y compras registradas por fecha que se quiere hacer en a la base de datos.
- Asimismo, tiene 3 apartados en los cuales puede clasificar las tablas:
 1. Flujo de caja general: Este es llamado por la función `btnGeneralActionPerformed` en la cual

esta hace la siguiente consulta a la base de datos: "SELECT F.ingegr,
 CONCAT(F.codpartida,CONCAT(' ',P.DESPARTIDA)) AS PARTIDA, F.impini AS S_INICIAL,
 F.imprealini AS R_INICIAL, F.impene AS S_ENERO, F.imprealene AS R_ENERO, F.impfeb
 AS S_FEBRERO, F.imprealfeb AS R_FEBRERO, F.impmar AS S_MARZO, F.imprealmar AS
 R_MARZO, F.impabr AS S_ABRIL, F.imprealabr AS R_ABRIL, F.impmay AS S_MAYO,
 F.imprealmay AS R_MAYO, F.impjun AS S_JUNIO, F.imprealjun AS R_JUNIO, F.impjul AS
 S_JULIO, F.imprealjul AS R_JULIO, F.impago AS S_AGOSTO, F.imprealago AS R_AGOSTO,
 F.impsep AS S_SEPTIEMBRE, F.imprealsep AS R_SEPTIEMBRE, F.impoct AS S_OCTUBRE,
 F.imprealoct AS R_OCTUBRE, F.impnov AS S_NOVIEMBRE, F.imprealnov AS
 R_NOVIEMBRE, F.impdic AS S_DICIEMBRE, F.imprealdic AS R_DICIEMBRE, F.impacum
 AS S_ACUMULADO, F.imprealacum AS R_ACUMULADO " + " FROM flujocaja_det F
 INNER JOIN PARTIDA P ON F.CODPARTIDA=P.CODPARTIDA AND F.INGEGR=P.INGEGR
 INNER JOIN FLUJOCAJA FC ON F.CODPARTIDA = FC.CODPARTIDA AND
 F.INGEGR=FC.INGEGR WHERE F.CODCIA="+varCodCiaGlobalDeLogin+" AND
 F.CODPYTO="+Integer.parseInt(codpyto.getSelectedItem().toString())+" AND
 F.ANNO="+Integer.parseInt(anio.getSelectedItem().toString())+" AND
 FC.CODCIA="+varCodCiaGlobalDeLogin+" AND
 FC.CODPYTO="+Integer.parseInt(codpyto.getSelectedItem().toString())+" ORDER BY
 INGEGR DESC, FC.NIVEL ASC,F.ORDEN ASC";

2.Flujos de caja real: Este es llamado por la función btnRealActionPerformed en la cual esta hace la siguiente consulta a la base de datos: "SELECT F.ingegr, CONCAT(F.codpartida,CONCAT(' ',P.DESPARTIDA)) AS PARTIDA, F.imprealini as INICIAL, F.imprealene as ENERO,
 F.imprealfeb AS FEBRERO, F.imprealmar AS MARZO, F.imprealabr AS ABRIL, F.imprealmay
 AS MAYO, F.imprealjun AS JUNIO, F.imprealjul AS JULIO, F.imprealago AS AGOSTO,
 F.imprealsep AS SEPTIEMBRE, F.imprealoct AS OCTUBRE, F.imprealnov AS NOVIEMBRE,
 F.imprealdic AS DICIEMBRE, F.imprealacum AS ACUMULADO "

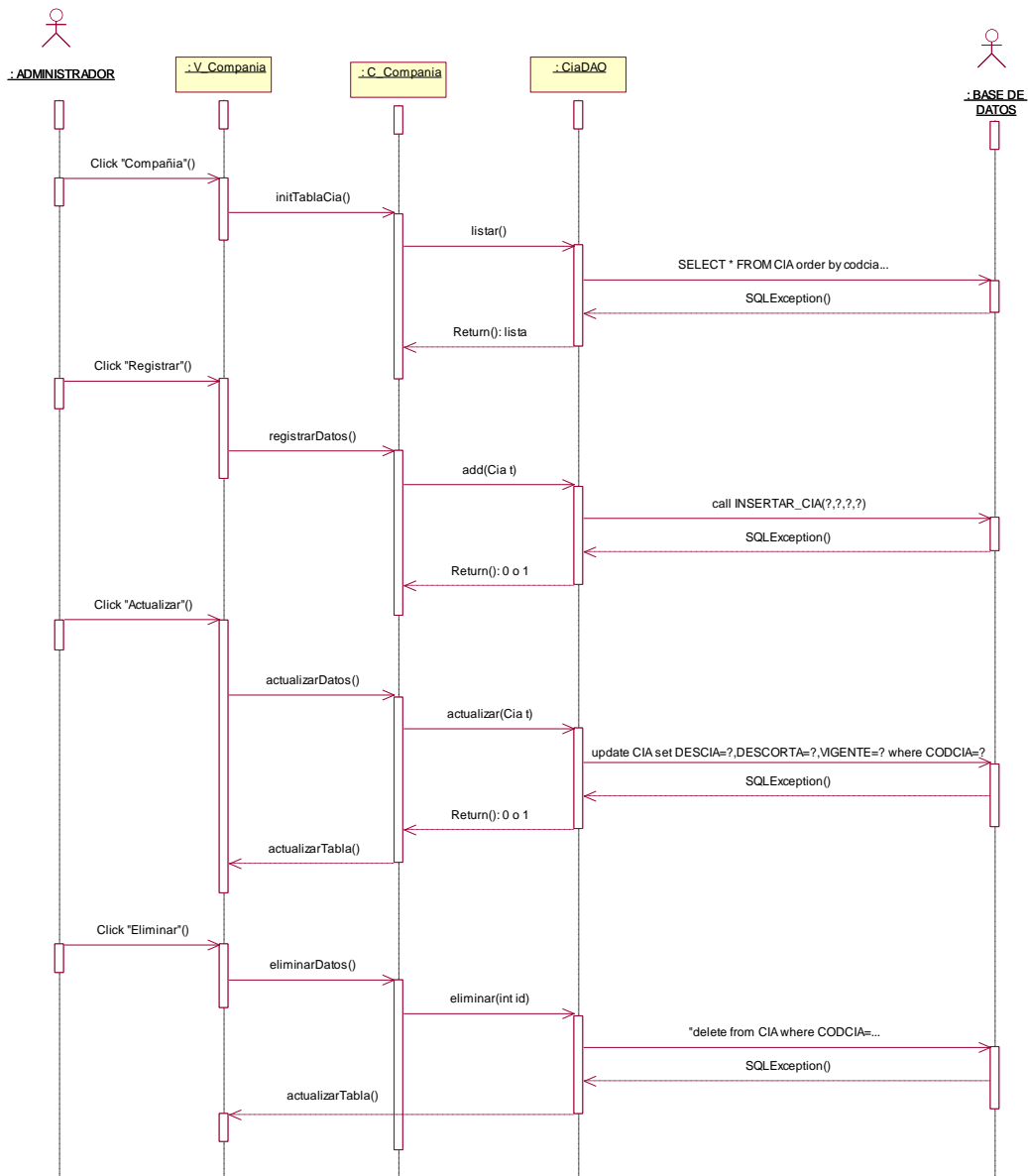
```
+ "FROM flujocaja_det F INNER JOIN PARTIDA P ON F.CODPARTIDA=P.CODPARTIDA
AND F.INGEGR=P.INGEGR INNER JOIN FLUJOCAJA FC ON F.CODPARTIDA =
FC.CODPARTIDA AND F.INGEGR=FC.INGEGR WHERE
F.CODCIA="+varCodCiaGlobalDeLogin+" AND
F.CODPYTO="+Integer.parseInt(codpyto.getSelectedItem().toString())+" AND
F.ANNO="+Integer.parseInt(anio.getSelectedItem().toString())+" AND
FC.CODCIA="+varCodCiaGlobalDeLogin+" AND
FC.CODPYTO="+Integer.parseInt(codpyto.getSelectedItem().toString())+" ORDER BY
INGEGR DESC, FC.NIVEL ASC,F.ORDEN ASC";
```

3.Flujo de caja supuesto: Este es llamado por la función btnSupuestoActionPerformed en la cual esta hace la siguiente consulta a la base de datos: "SELECT F.ingegr,
 CONCAT(F.codpartida,CONCAT(' ',P.DESPARTIDA)) AS PARTIDA, F.impini AS INICIAL,
 F.impene AS ENERO, F.impfeb AS FEBRERO, F.impmar AS MARZO, F.impabr AS ABRIL,
 F.impmay AS MAYO, F.impjun AS JUNIO, F.impjul AS JULIO, F.impago AS AGOSTO,
 F.impsep AS SEPTIEMBRE, F.impoct AS OCTUBRE, F.impnov AS NOVIEMBRE, F.impdic AS
 DICIEMBRE, F.impacum AS ACUMULADO + "FROM flujocaja_det F INNER JOIN
 PARTIDA P ON F.CODPARTIDA=P.CODPARTIDA AND F.INGEGR=P.INGEGR INNER JOIN
 FLUJOCAJA FC ON F.CODPARTIDA FC.CODCIA="+varCodCiaGlobalDeLogin+" AND
 FC.CODPYTO="+Integer.parseInt(codpyto.getSelectedItem().toString())+" ORDER BY
 INGEGR DESC, FC.NIVEL ASC,F.ORDEN ASC";

- Luego se muestra el resultado de la consulta en la parte de inferior de la vista dinámica.

Finalmente, el desarrollador puede limpiar los resultados con el botón Limpiar o cerrar la vista dinámica.

Administrar compañías



Pasos

- El administrador hace click en Compañía.
- La clase V_Compañía inicia la interfaz con la función initTablaCia() que pertenece a la clase C_Compañía.

- La clase CiaDAO ejecuta la función listar() que se comunica con la base de datos mediante la consulta `SELECT * FROM CIA order by codcia asc`, lo cual mostrara las compañías registradas hasta ese momento.

Registrar

- La V_Compañía tiene un apartado donde el administrador ingresa los datos de la nueva compañía que se va a registrar, luego hace click en Registrar
- La clase C_Register obtiene los datos colocados por el administrador mediante la función registrarDatos() y ejecuta la función add(Cia t) de la clase CiaDAO
- La clase CiaDAO llama al procedimiento almacenado INSERTAR_CIA(?,?,?,?) que en la base de datos está como la consulta `INSERT INTO CIA VALUES (SEC_CIA.NEXTVAL,DES,DESCORTA,VIG)`.
- Luego, según se haya ejecutado la petición, la base de datos vuelve en una excepción, SQLException, lo que se convertirá en un 1 o 0.
- Finalmente, este 1 o 0 representa un mensaje en la vista para el administrador que será "Compañía registrada correctamente", si la consulta fue exitosa.

Actualizar

- El administrador selecciona la compañía que desea modificar, una vez modificado los datos de la compañía hace click en Actualizar.
- La clase C_Compañía ejecuta la función actualizarDatos(), la cual a la vez ejecuta la función actualizar(Cia t) en la clase CiaDAO.
- La clase CiaDAO tiene la conexión a la base de datos y ejecuta la actualización de la compañía mediante la petición `UPDATE CIA set DESCIA=?,DESCORTA=?,VIGENTE=? where CODCIA=?`
- Luego, según se haya ejecutado la petición, la base de datos vuelve en una excepción, SQLException, lo que se convertirá en un 1 o 0.

- Finalmente, se actualizan los datos de la compañía y se muestra la tabla con los nuevos datos mediante la función actualizarTabla() de la clase C_Compañía

Eliminar

- El administrador selecciona la compañía que desea eliminar y hace click en Eliminar.
- La clase C_Compañía ejecuta la función eliminarDatos(), la cual a la vez ejecuta la función eliminar(int id) en la clase CiaDAO.
- La clase CiaDAO tiene la conexión a la base de datos y ejecuta la eliminacion de la compañía mediante la petición DELETE FROM CIA WHERE CODCIA="" + id + "
- Finalmente, se actualizan la tabla mediante la función actualizarTabla() de la clase C_Compañía

Consultas e instrucciones

Procedimiento almacenado INSERTAR_CIA

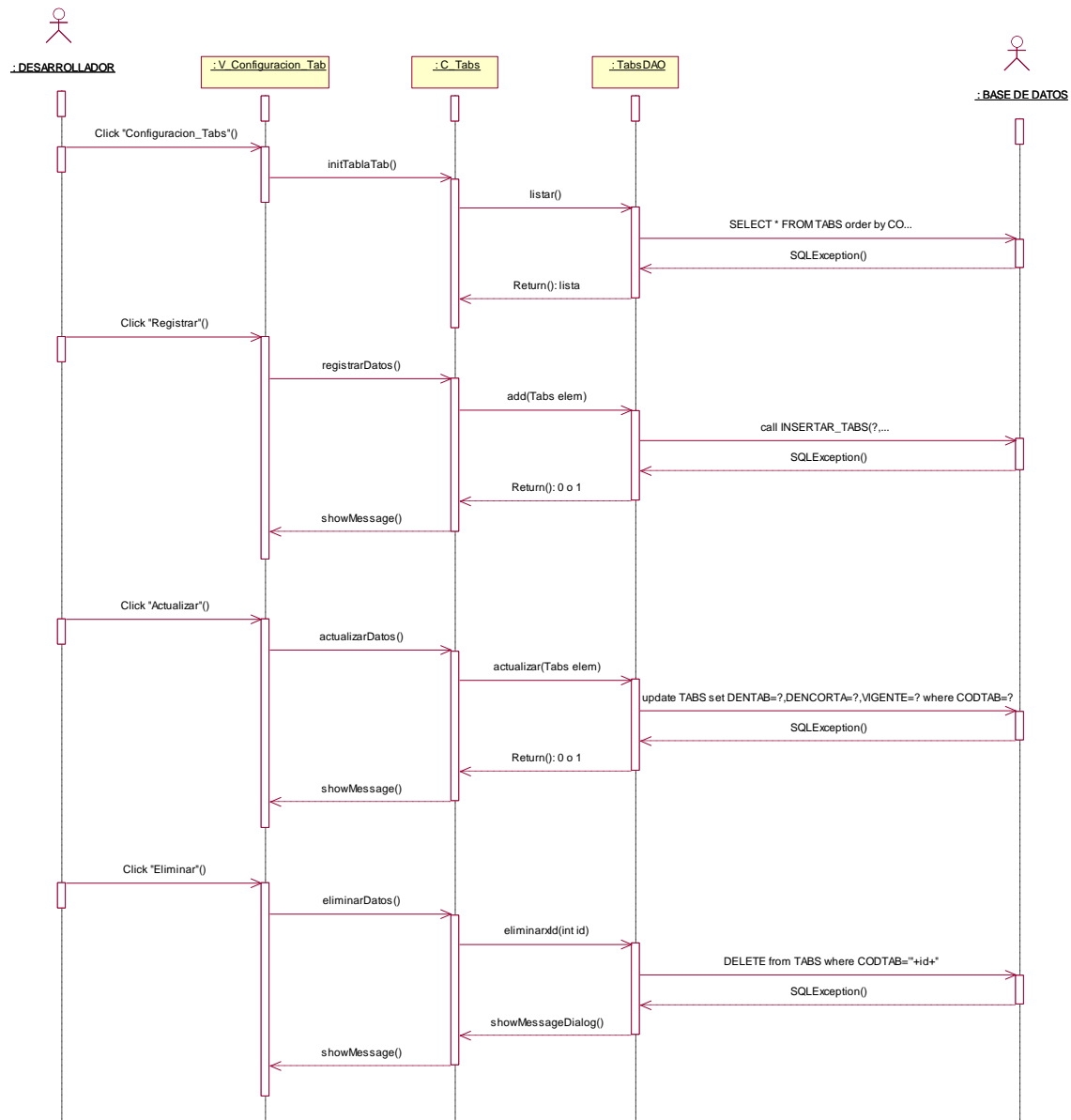
```

/*-----*/
create or replace NONEDITIONABLE PROCEDURE INSERTAR_CIA(
    CODC IN CIA.CODCIA%TYPE,
    DES IN CIA.DESCIA%TYPE,
    DESCORTA IN CIA.DESCORTA%TYPE,
    VIG IN CIA.VIGENTE%TYPE)
IS
BEGIN
    INSERT INTO CIA (codcia, descia, descorta, vigente)
VALUES(SEC_CIA.NEXTVAL,DES,DESCORTA,VIG);
END;
/*-----*/

```

El procedimiento se encarga de insertar un nuevo registro en la tabla CIA con los parámetros ya mencionados con la instrucción `INSERT INTO CIA (codcia, descia, descorta, vigente)`
`VALUES (SEC_CIA.NEXTVAL, DES, DESCORTA, VIG)`.

Administrar tablas



Pasos

- El desarrollador hace click en Tabs.
- La clase V_Configuracion_Tabs inicia la interfaz con la función initTablaTab() que pertenece a la clase C_Tabs.

- La clase TabsDAO ejecuta la función listar() que se comunica con la base de datos mediante la consulta `SELECT * FROM TABS order by CODTAB`, lo cual mostrara los elementos registrados hasta ese momento.

Registrar

- La V_Configuracion_Tabs tiene un apartado donde el desarrollador ingresa los datos del nuevo elemento que se va a registrar, luego hace click en Registrar
- La clase C_Tabs obtiene los datos colocados por el desarrollador mediante la función registrarDatos() y ejecuta la función add(Tabs elem) de la clase TabsDAO
- La clase CiaDAO llama al procedimiento almacenado INSERTAR_TABS(?,?,?) que en la base de datos está como la consulta `INSERT INTO TABS VALUES(SEC_TABS.NEXTVAL,DETAB,DECOR,VIG)`.
- Luego, según se haya ejecutado la petición, la base de datos vuelve en una excepción, SQLException, lo que se convertirá en un 1 o 0.
- Finalmente, este 1 o 0 representa un mensaje en la vista para el desarrollador que será “Tabs registrado correctamente” si la consulta fue exitosa.

Actualizar

- El desarrollador selecciona el elemento que desea modificar, una vez modificado los datos hace click en Actualizar.
- La clase C_Tabs ejecuta la función actualizarDatos(), la cual a la vez ejecuta la función actualizar(Tabs elem) en la clase TabsDAO.
- La clase TabsDAO tiene la conexión a la base de datos y ejecuta la actualización del elemento mediante la petición `UPDATE TABS set DENTAB=?,DENCORTA=?,VIGENTE=? where CODTAB=?`.
- Luego, según se haya ejecutado la petición, la base de datos vuelve en una excepción, SQLException, lo que se convertirá en un 1 o 0.

- Finalmente, este 1 o 0 representa un mensaje en la vista para el desarrollador que será "Tabs actualizado correctamente" si la consulta fue exitosa.

Eliminar

- El desarrollador selecciona el elemento que desea eliminar y hace click en Eliminar.
- La clase C_Tabs ejecuta la función eliminarDatos(), la cual a la vez ejecuta la función eliminarxId(int id) en la clase TabsDAO.
- La clase TabsDAO tiene la conexión a la base de datos y ejecuta la eliminación mediante la petición DELETE from TABS where CODTAB="'+id+'"
- Finalmente, se muestra el mensaje "Empresa eliminada con éxito" si la consulta fue exitosa.

Consultas e instrucciones

Procedimiento almacenado INSERTAR_TABS

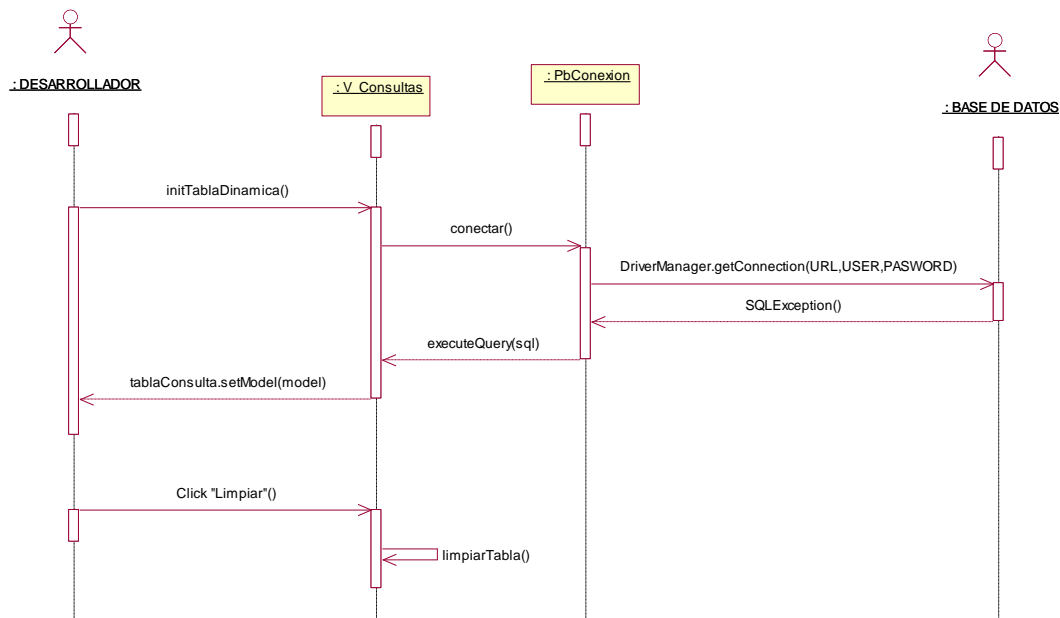
```

/*-----*/
create or replace PROCEDURE INSERTAR_TABS(
    /*CTAB IN TABS.CODTAB%TYPE,*/
    DETAB IN TABS.DENTAB%TYPE,
    DECOR IN TABS.DENCORTA%TYPE,
    VIG IN TABS.VIGENTE%TYPE)
IS
BEGIN
    INSERT INTO TABS VALUES(SEC_TABS.NEXTVAL,DETAB,DECOR,VIG);
END;

El procedimiento se encarga de insertar un nuevo registro en la tabla TABS con los
parámetros ya mencionados con la instrucción INSERT INTO TABS
VALUES(SEC_TABS.NEXTVAL,DETAB,DECOR,VIG).
/*-----*/

```

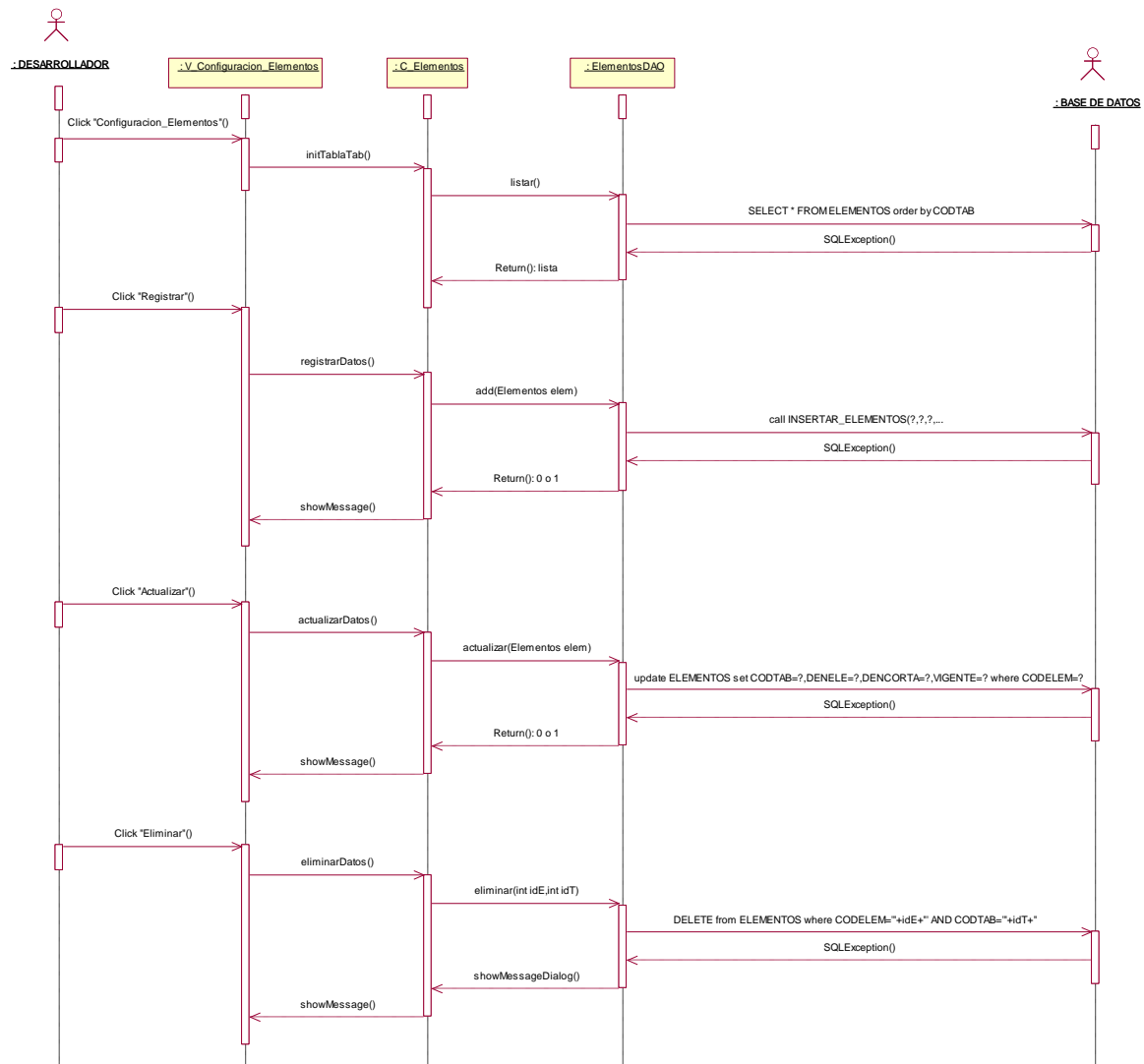
Administrar consultas



Pasos

- El desarrollador hace clic en Consulta.
- La clase `V_Consultas` inicia una vista dinámica con la función `initTablaDinamica()` en la cual se muestra un apartado para escribir la consulta que se quiera hacer a la base de datos.
- Después de escribir la consulta, el desarrollador hace clic en el botón Consultar y la clase `PbConexion` hace la conexión con la base de datos y posteriormente se ejecuta la consulta escrita por el desarrollador.
- Luego se muestra el resultado de la consulta en la parte de inferior de la vista dinámica. Finalmente, el desarrollador puede limpiar los resultados con el botón Limpiar o cerrar la vista dinámica.

Administrar elementos



Pasos

- El desarrollador hace click en Elementos.
- La clase V_Configuracion_Elementos inicia la interfaz con la función initTablaTab() que pertenece a la clase C_Elementos.

- La clase ElementosDAO ejecuta la función listar() que se comunica con la base de datos mediante la consulta `SELECT * FROM ELEMENTOS order by CODTAB`, lo cual mostrara los elementos registrados hasta ese momento.

Registrar

- La V_Configuracion_Elementos tiene un apartado donde el desarrollador ingresa los datos del nuevo elemento que se va a registrar, luego hace click en Registrar
- La clase C_Elementos obtiene los datos colocados por el desarrollador mediante la función registrarDatos() y ejecuta la función add(Elementos elem) de la clase ElementosDAO
- La clase ElementosDAO llama al procedimiento almacenado `INSERTAR_ELEMENTOS(?,?,?,?,?)` que en la base de datos está como la consulta `INSERT INTO ELEMENTOS VALUES (CTAB,CELE,DEELE,DECORTA,VIG);`
- Luego, según se haya ejecutado la petición, la base de datos vuelve en una excepción, `SQLException`, lo que se convertirá en un 1 o 0.
- Finalmente, este 1 o 0 representa un mensaje en la vista para el desarrollador que será "Elemento registrado correctamente" si la consulta fue exitosa.

Actualizar

- El desarrollador selecciona el elemento que desea modificar, una vez modificado los datos hace click en Actualizar.
- La clase C_Elementos ejecuta la función actualizarDatos(), la cual a la vez ejecuta la función actualizar(Elementos elem) en la clase ElementosDAO.
- La clase ElementosDAO tiene la conexión a la base de datos y ejecuta la actualización del elemento mediante la petición `UPDATE ELEMENTOS set CODTAB=?,DENELE=?,DENCORTA=?,VIGENTE=? where CODELEM=?`.
- Luego, según se haya ejecutado la petición, la base de datos vuelve en una excepción, `SQLException`, lo que se convertirá en un 1 o 0.

- Finalmente, este 1 o 0 representa un mensaje en la vista para el desarrollador que será "Elemento registrado correctamente" si la consulta fue exitosa.

Eliminar

- El desarrollador selecciona el elemento que desea eliminar y hace click en Eliminar.
- La clase C_Elementos ejecuta la función eliminarDatos(), la cual a la vez ejecuta la función eliminar(int idE,int idT) en la clase ElementosDAO.
- La clase ElementosDAO tiene la conexión a la base de datos y ejecuta la eliminación mediante la petición DELETE from ELEMENTOS where CODELEM="" + idE + "" AND CODTAB="" + idT + "".
- Finalmente, se muestra el mensaje "Elemento eliminado con éxito" si la consulta fue exitosa.

Consultas e instrucciones

Procedimiento almacenado INSERTAR_ELEMENTOS

```

/*-----*/
create or replace PROCEDURE INSERTAR_ELEMENTOS(
    CTAB IN ELEMENTOS.CODTAB%TYPE,
    CELE IN ELEMENTOS.CODELEM%TYPE,
    DEELE IN ELEMENTOS.DENELE%TYPE,
    DECORTA IN ELEMENTOS.DENCORTA%TYPE,
    VIG IN ELEMENTOS.VIGENTE%TYPE)
IS
BEGIN
    INSERT INTO ELEMENTOS VALUES
    (CTAB,CELE,DEELE,DECORTA,VIG);
END;
/*-----*/

```

El procedimiento se encarga de insertar un nuevo registro en la tabla ELEMENTOS con los parámetros ya mencionados con la instrucción `INSERT INTO ELEMENTOS VALUES (CTAB, CELE, DEELE, DECORTA, VIG);`

Consideraciones

Implementación de la interfaz CRUD

Esta interfaz es implementada en otras clases para hacer uso de los métodos de add(), actualizar() y eliminar(); que hacen referencia a añadir, actualizar y eliminar los registros en la base de datos.

```
1 package Modelo.Interface;
2
3 import java.util.List;
4
5 public interface CRUD<T> {
6     public List listar();
7     public int add(T t);
8     public int actualizar(T t);
9     public void eliminar(int id);
10 }
```

Las clases que implementan esta son las que en la interfaz de usuario se tiene la opción de “registrar”, “actualizar” y “eliminar” a través de botones.

Las clases que la utilizan son del paquete ModeloDAO: Empleado, Empresa, Cliente, Partida, Proveedor, entre otros. Casi todas las clases de este paquete.

La tabla PERSONA

Esta tabla es una tabla interesante, ya que involucra la creación de clientes, empleados, empresas venta y proveedores. Esto debido a que comparten los mismos atributos, se encuentran en la tabla PERSONA. Y para evitar la reiteración de estos se hizo uso de esta nueva tabla.

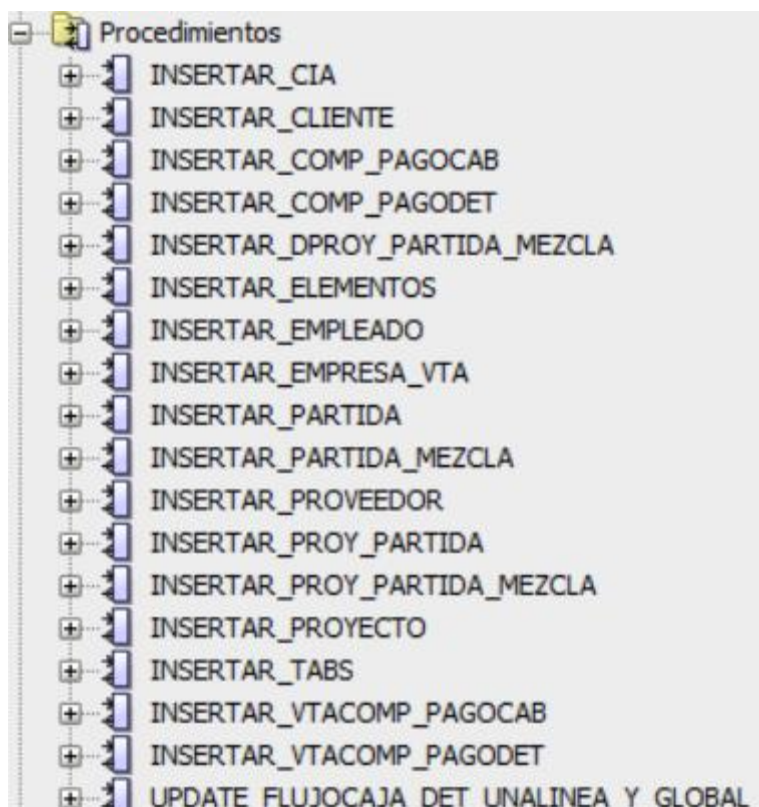
	CODCIA	CODPERSONA	TIPPERSONA	DESPERSONA	DESCORTA	DESCALTERNA	DESCORTAALT	VIGENTE
3	1	21 2		Es una persona de prueba	Si	Estoy probando	No quiero	1
4	1	81 1		ghjf	hgj	jhg	jhg	1
5	1	82 1		ghjf	hgj	jhg	jhg	1
6	1	41 2		ghfdgfh	gfhgfh	fghgfhgfh	hgfgfhfgh	1
7	1	1 1		Devenco	Dev	DevSer	DevSer	1
8	2	2 0		Jorge Vargas Mozz	Jvargas	Jvargas	Jvargas	1
9	3	3 1		PVN	PVN	PVN	PVN	1
10	1	4 1		JPVD	PVD	PVD	PVD	1
11	2	5 1		Trafic Clay	Clay	Clay	Clay	1
12	3	6 1		Lab Pelaez	LabP	LabP	LabP	1
13	1	7 0		Moncada	Mon	Mon	Mon	1
14	2	8 1		Mun. Lima	ML	ML	ML	1
15	3	9 1		Consortio Sierra Sur	ConSS	ConSS	ConSS	1
16	1	10 1		Consortio Mar	ConD	Emape	Ema	1
17	2	11 1		Consortio Sol	ConD	Emape	Ema	1
18	3	12 1		Mun. Pasco	ConD	Emape	Ema	1
19	1	42 1		xzcxzcx	zxczxczxc	zxczxczxc	czxczxczxc	1
20	1	83 1		JOSÉ ROBERTO SERNAQUÉ GUTIERREZ	JOCHE	JOCHEBOT	YOS	1

Con un select a la tabla PERSONA se puede ver las diversas personas como empresas y empleados.

Otros objetos de la base de datos

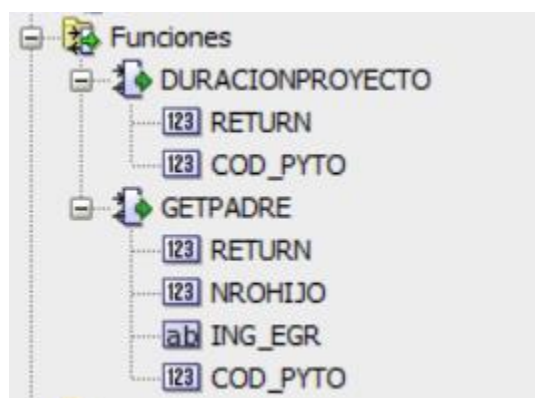
Procedimientos

Los procedimientos son bloques de código predefinidos que se almacenan en la base de datos pueden realizar operaciones complejas de consulta y devolver resultados. Dentro de la base de datos se encuentran todos los siguientes procedimientos los cuales serán explicados en los diagramas de secuencia.



Funciones

Las funciones son similares a los procedimientos, pero una función siempre devuelve un valor. Dentro de la base de datos encontramos dos funciones: DURACIONPROYECTO y GETPADRE, los cuales se explicarán a continuación.



Función DURACIONPROYECTO

/*-----*/

```

create or replace NONEDITIONABLE FUNCTION duracionProyecto
    (COD_PYTO IN PROYECTO.CODPYTO%TYPE)
    return NUMBER
IS
    duracionEnAños NUMBER;
BEGIN
    select count(n) into duracionEnAños from
        (select distinct rownum n from dual connect by level <= (SELECT
        ANNOFIN FROM PROYECTO WHERE CODPYTO=COD_PYTO))
        where n >= (SELECT ANNOINI FROM PROYECTO WHERE CODPYTO=COD_PYTO)
    ORDER BY n;
    RETURN duracionEnAños;
END duracionProyecto;

/*-----*/

```

Explicación

create or replace NONEDITIONABLE FUNCTION: Esta cláusula indica que se va a crear o reemplazar una función en la base de datos que no es editable.

duracionProyecto: Es el nombre de la función. **(COD_PYTO IN PROYECTO.CODPYTO%TYPE):** Es un parámetro de entrada **COD_PYTO** que toma el tipo de datos del campo **CODPYTO** de la tabla **PROYECTO**.

return NUMBER: Indica que la función devolverá un valor de tipo **NUMBER**.

duracionEnAños NUMBER: Declara una variable local **duracionEnAños** de tipo **NUMBER** que almacenará la duración del proyecto en años.

Cuerpo de la función

- Genera una lista de números del 1 al año de finalización (**ANNOFIN**) del proyecto.
- Filtra esta lista para incluir solo los años desde el inicio (**ANNOINI**) hasta el fin (**ANNOFIN**).

- Cuenta los años en este rango y devuelve el resultado.

Función GETPADRE

```

/*-----*/
create or replace NONEDITIONABLE FUNCTION getPadre
(
  NROHIJO IN FLUJOCAJA.CODCIA%TYPE,
  ING_EGR IN FLUJOCAJA.INGEGR%TYPE,
  COD_PYTO IN PROYECTO.CODPYTO%TYPE)
  return NUMBER
IS
  NROPADRE NUMBER;
BEGIN
  SELECT PADCODPARTIDA INTO NROPADRE FROM PROY_PARTIDA_MEZCLA WHERE CODPARTIDA
= NROHIJO AND INGEGR=ING_EGR AND CODPYTO = COD_PYTO;
  RETURN NROPADRE;
END getPadre;
/*-----*/

```

Explicación

- **create or replace NONEDITIONABLE FUNCTION:** Esta cláusula indica que se va a crear o reemplazar una función en la base de datos que no es editionable.
- **getPadre:** Es el nombre de la función.
- **(NROHIJO IN FLUJOCAJA.CODCIA%TYPE, ING_EGR IN FLUJOCAJA.INGEGR%TYPE, COD_PYTO IN PROYECTO.CODPYTO%TYPE):** Son los parámetros de entrada:
 - **NROHIJO:** Un número de hijo que corresponde al tipo de dato CODCIA en la tabla FLUJOCAJA.
 - **ING_EGR:** Un valor que indica ingreso o egreso, correspondiente al tipo de dato INGEGR en la tabla FLUJOCAJA.

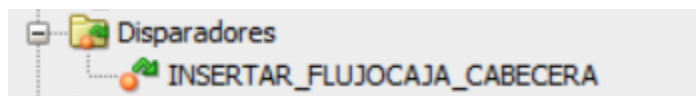
- COD_PYTO: El código del proyecto, correspondiente al tipo de dato CODPYT en la tabla PROYECTO.
- return NUMBER: Indica que la función devolverá un valor de tipo NUMBER.

Cuerpo de la función

- Con los parámetros de entrada la función realiza una consulta en la tabla PROY_PARTIDA_MEZCLA para encontrar el código del padre (PADCODPARTIDA) de la partida especificada. Si encuentra un registro que cumple con los criterios, el valor de PADCODPARTIDA se almacena en NROPADRE y se devuelve.

Trigger

Los triggers son bloques de código que se ejecutan automáticamente en respuesta a ciertos eventos en la base de datos, como INSERT, UPDATE o DELETE en una tabla. En la base de datos encontramos el trigger INSERTAR_FLUJOCAJA_CABECERA



Trigger INSERTAR_FLUJOCAJA_CABECERA

```

/*-----*/
create or replace NONEDITIONABLE TRIGGER INSERTAR_FLUJOCAJA_CABECERA
BEFORE INSERT ON PROY_PARTIDA_MEZCLA FOR EACH ROW
DECLARE
    annoAuxiliar NUMBER(10);
    numeroDeColumnas NUMBER(10);
    CURSOR CURSOR_ANNOS IS
        select n from
            (select distinct rownum n from dual connect by level <= (SELECT ANNOFIN FROM
PROYECTO WHERE CODPYTO=:new.CODPYTO))

```

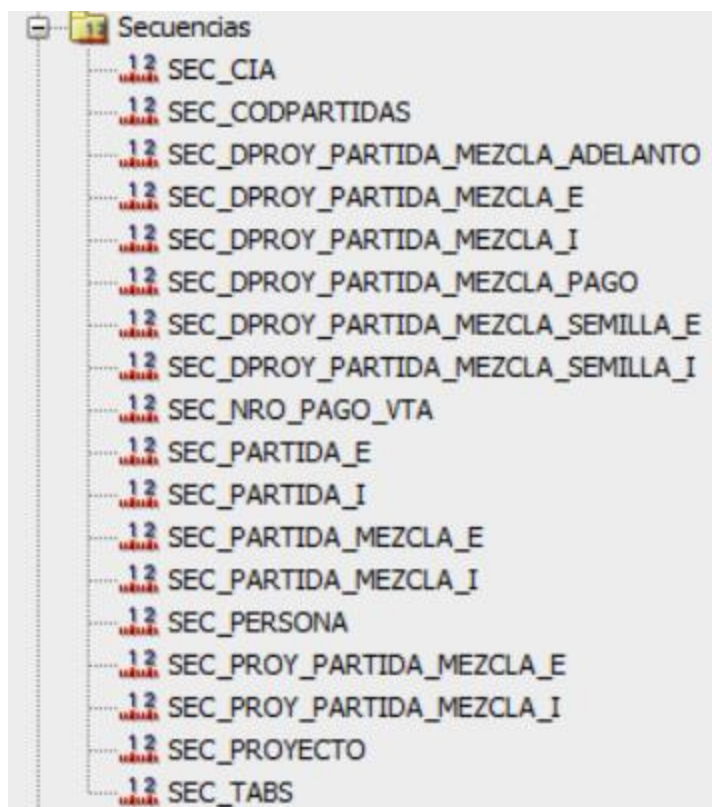

- `INSERTAR_FLUJOCAJA_CABECERA`: Es el nombre del trigger.
- `BEFORE INSERT ON PROY_PARTIDA_MEZCLA FOR EACH ROW`
- `:` Especifica que el trigger se ejecuta antes de que se inserte una fila en la tabla `PROY_PARTIDA_MEZCLA` y que se ejecutará para cada fila individualmente.
- `annoAuxiliar NUMBER`: Declara una variable `annoAuxiliar` de tipo `NUMBER` para almacenar años.
- `numeroDeColumnas NUMBER`: Declara una variable `numeroDeColumnas` de tipo `NUMBER` para contar el número de columnas.
- `CURSOR CURSOR_ANNOS IS`: Declara un cursor llamado `CURSOR_ANNOS` que genera una lista de años desde el año de inicio (`ANNOINI`) hasta el año de fin (`ANNOFIN`) del proyecto especificado por: `new.CODPYTO`.

Cuerpo de la función

- Cuenta los años desde el inicio hasta el fin del proyecto relacionado (`CODPYTO`).
- Insertar un registro en `FLUJOCAJA` con los detalles del nuevo registro de `PROY_PARTIDA_MEZCLA`.
- Insertar varios registros en `FLUJOCAJA_DET`, uno por cada año en el rango determinado, con detalles inicializados a cero o valores predeterminados.

Secuencias

Las secuencias son objetos de base de datos utilizados para generar valores numéricos únicos y secuenciales. Son especialmente útiles para crear valores de clave primaria auto incrementables. En la base de datos encontramos todas las siguientes secuencias, la mayoría utilizada para la generación de los índices o para la secuencia de entradas en los procedimientos de ingreso de datos a la base de datos.



Secuencia SEC_CIA

```
/*-----*/
```

```
create sequence SEC_CIA
```

```
    start with 1
```

```
    increment by 1
```

```
    maxvalue 99999
```

```
    minvalue 1;
```

```
/*-----*/
```

La secuencia SEC_CIA es utilizada en el procedimiento INSERTAR_CIA mediante la consulta INSERT INTO CIA VALUES(SEC_CIA.NEXTVAL,DES,DESCORTA,VIG), en la cual la secuencia automatiza la asignación de la columna CODCIA en la tabla CIA.

Secuencia SEC_PERSONA

```
/*-----*/
```

```
create sequence SEC_PERSONA
```



```
start with 1  
increment by 1  
maxvalue 99999  
minvalue 1;  
  
/*-----*/
```

La secuencia SEC_PERSONA es utilizada en el procedimiento INSERTAR_CLIENTE mediante la consulta INSERT INTO PERSONA VALUES (CODCIA,SEC_PERSONA.NEXTVAL, '2', DESP, DESCOR, DESCALT,DESCORALT,VIG), en la cual la secuencia automatiza la asignación de la columna CODPERSONA en la tabla PERSONA.