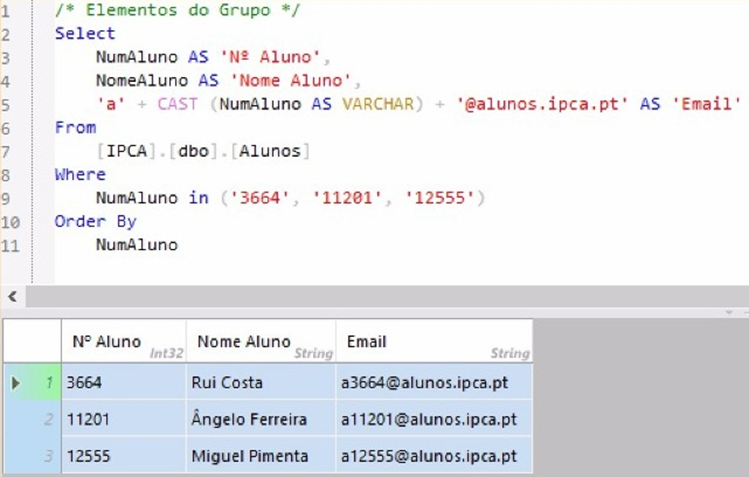




Trabalho Prático

Armazenamento e Acesso a Dados

**00.2.1 Plano de projeto**



Índice

[Objetivo 3](#_Toc479024158)

[Âmbito 3](#_Toc479024159)

[Metodologia 3](#_Toc479024160)

[Análise de Requisitos 3](#_Toc479024161)

[Projeto Concetual 4](#_Toc479024162)

[Projeto lógico 4](#_Toc479024163)

[Projeto físico 4](#_Toc479024164)

[Equipa 5](#_Toc479024165)

[Plano do Projeto 5](#_Toc479024166)

[Fatores Críticos de Sucesso 6](#_Toc479024167)

[Análise de Requisitos 6](#_Toc479024168)

[Regras de Negócio 6](#_Toc479024169)

[Requisitos Funcionais 6](#_Toc479024170)

[Requisitos Técnicos 9](#_Toc479024171)

[Projeto Lógico e Projeto Relacional 11](#_Toc479024172)

[Modelo Lógico 11](#_Toc479024173)

[Ata 02 11](#_Toc479024174)

[Ata 03 12](#_Toc479024175)

[Ata 04 13](#_Toc479024176)

[Ata 05 14](#_Toc479024177)

[Modelo Geral 15](#_Toc479024178)

[Modelo Relacional 16](#_Toc479024179)

[Ata 02 16](#_Toc479024180)

[Ata 03 17](#_Toc479024181)

[Ata 04 18](#_Toc479024182)

[Ata 05 19](#_Toc479024183)

[Modelo Geral 20](#_Toc479024184)

[Opções Conceptuais 21](#_Toc479024185)

# Objetivo

Este projeto visa o desenvolvimento da componente de armazenamento e acesso a dados necessária às operações do cliente Piscinas de Barcacellos.

O novo sistema de Gestão de Bases de Dados deverá possibilitar ao cliente compatibilidade com a solução atualmente em uso ou a possibilidade de migração dos dados para a nova plataforma com um *downtime* baixoou inexistente por forma a não causar impacto nas operações.

A *timeline* do projeto estipula a conclusão da sua implementação até ao dia 02 de Junho de 2017.

# Âmbito

O âmbito do projeto centra-se na componente de armazenamento e acesso a dados das Piscinas de Barcacellos.

Toda a camada aplicacional e interface será da total responsabilidade da equipa de desenvolvimento de software das piscinas.

# Metodologia

## Análise de Requisitos

Primeira fase do projeto, visa levantar os requisitos da informação que deverá estar representada.

## Projeto Concetual

Produção do esquema expresso numa linguagem independente da bases de dados

## Projeto lógico

Conversão do modelo lógico em modelo relacional de dados

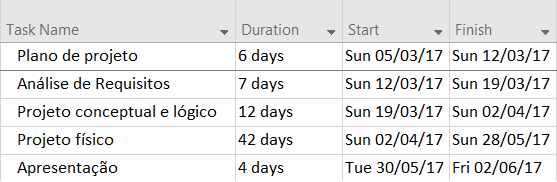
## Projeto físico

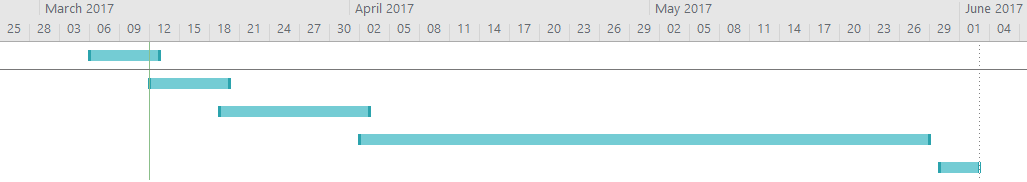
Parametrização do modelo relacional em função do que se pretende da sua utilização

# Equipa

* Gestor do Projeto
  + Rui Castro
* Equipa responsável pelo desenvolvimento da Base de Dados
  + Miguel Pimenta
  + Ângelo Ferreira
  + Rui Costa
* Programador de interfaces
  + João Vítor

# Plano do Projeto





# Fatores Críticos de Sucesso

* Tempo;
* Envolvimento dos responsáveis na fase de levantamento de requisitos;
* Disponibilidade da equipa da Piscina para reuniões adicionais;

# Análise de Requisitos

## Regras de Negócio

* A empresa está organizada por departamentos;
* A cada departamento é atribuído um código, designação, sendo dirigido por um diretor;
* O serviço das piscinas é lecionar aulas de natação e hidroginástica assim como disponibilização, em horário livre, da piscina e jacuzzi.

## Requisitos Funcionais

* Todos os utilizadores deverão ser identificados pelo seu numero de colaborador;
* O acesso às piscinas pode ser feito por utentes e/ou colaboradores;
* A equipa de limpeza e manutenção tem entrada e saídas livres;
* O sistema armazena os dados básicos: categoria e número ;
* O sistema autoriza/nega a entrada/saída com base no estado atual do utente.
* Os utentes e professores apenas podem entrar nos 15 minutos antes do início da aula, e sair apenas nos 30 minutos depois da aula terminar; No caso de incumprimento deverá ser registada uma infração;
* Os nadadores-salvadores obedecem a uma escala de serviço das 9:00-18:00 e 18:00-23:00 tendo tolerância de 15 minutos para entrada e saída;
* O torniquete deverá estar preparado para invocar função com a categoria, número de utente/colaborador e tipo de movimento e conseguir interpretar as respostas;
* O/A rececionista tem a possibilidade de autorizar a saída/entrada sem obedecer aos critérios estabelecidos, mas ficando registada uma infração;
* O responsável de recursos humanos deverá poder consultar a lista de infrações mensal (para efeitos de processamento salarial) e uma listagem dos professores com mais de 20% de picagem com infrações (para efeitos de processo disciplinar); Adicionalmente, como segunda prioridade, pretende-se fazer o controlo de ponto para as outras categorias de colaboradores.
* De acordo com o plano de emergência o/a rececionista deverá ter acesso a listar e contar o número de pessoas que estão na piscina; no caso de utentes com idade inferior a 3 anos, que estão acompanhados por um adulto, devem ser consideradas 2 pessoas para a contagem de utentes;
* O responsável de departamento pretende uma listagem mensal com todas a infrações registadas por categoria e outra listagem com as 50 pessoas com mais infrações no ano corrente;
* As piscinas pretendem melhorar a comunicação instalando um placard eletrónico com várias informações úteis, entre elas, as turmas com vagas; para isso o placard está programado para invocar o pedido desta informação recebendo uma listagem com código da turma (constituído por tipo [I,A,H], nível [1 a 3] e numero sequencial), horário (constituído pela concatenação dos dias da semana e horas) e o número de vagas;
* O responsável pelo departamento pretende saber, entre outras informações, qual a média de turmas por professor (e quais os professores que estão abaixo da média) e quais as turmas com percentagem de vagas acima de 50% (com vista a cativar novos utentes para as aulas ou propor a mudança dos utentes para outro horário).
* O relógio de ponto está preparado para enviar o número de funcionário e data/hora da picagem, interpretando as respostas ‘S’ (sucesso) e ‘E’ (erro) mediante o sucesso do registo da picagem;
* Pretende-se que todas as matérias relevantes para a manutenção da piscina sejam registadas e, para algumas delas, seja definido um nível de stock mínimo;
* A equipa de aprovisionamento dará entrada de novos materiais (identificados por um número sequencial) assim como de movimentos de entrada de stock através de uma compra (com a data da compra, materiais e quantidades);
* A equipa de manutenção, através de um interface tátil desenvolvido pela equipa aplicacional, dará saída de stock por via de uma ordem de manutenção (onde constará a data da manutenção, objetivo de manutenção e todos os materiais envolvidos);
* Assim que o stock mínimo seja atingido, deverá ser gerada uma requisição de compra para a equipa de aprovisionamento (com a quantidade necessária para atingir o stock mínimo);
* A equipa de aprovisionamento pode, contudo, efetuar compras mesmo que não haja requisição de compra e, mesmo que haja, quantidades diferentes das que constam da ordem (por restrições financeiras ou mesmo para tirar partido de promoções do fornecedor);
* A equipa de aprovisionamento terá pois que poder listar todas as requisições de compra (tendo uma vista requisição-material e material-requisição) e, após solicitar a comprar de materiais ao fornecedor, atualizar a requisição como fechada; adicionalmente pretendem saber os produtos em stock sem movimentos há mais de um ano (potenciais monos).

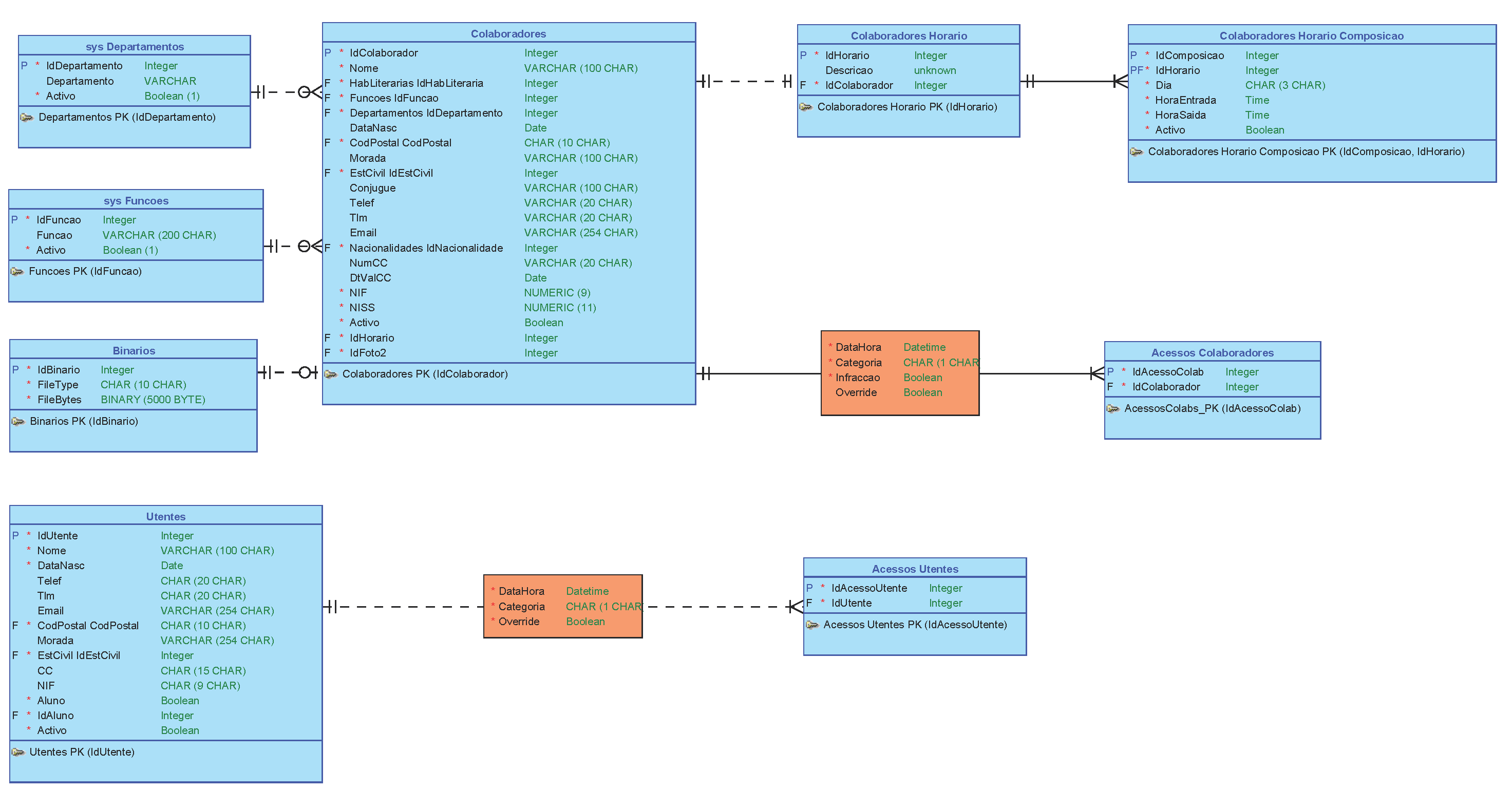
## Requisitos Técnicos

* A base de dados deverá responder às necessidades de armazenamento e exploração dos diversos departamentos envolvidos num único repositório;
* Existirá migração de dados históricos dos sistemas vigentes, nomeadamente os dados mestre de colaboradores (serão exportados os dados em formato .csv);
* Haverão reuniões com cada um dos departamentos para refinar as referidas necessidades;
* Os sistemas a migrar para a nova arquitetura serão, por ordem de prioridade, (1) Sistema de Controlo de Acessos, (2) Sistema de Gestão de Turmas, (3) Sistema de Controlo de Presenças e (4) Sistema de Gestão de Manutenção.
* Deverão ser facultadas funções para invocação remota
* Os servidores estão alojados num fornecedor de hosting responsável pela instalação do servidor, sistema operativo e implementação da política de backups; como tal, a equipa de projeto terá que especificar os requisitos técnicos para a instalação do servidor assim como a política de backups a implementar pelo fornecedor de hosting;
* A camada de base de dados deverá assegurar a autenticação e segurança de utilizadores (os utilizadores deverão ser identificados pelo seu numero de colaborador) e permitir a criação e eliminação de utilizadores (por administradores);
* O Sistema de Gestão de Turmas (SGT) está atualmente implementado por via de formulários em papel onde constam as aulas que a turma tem e respetivos horários (ex.: 4as às 21:00 e Sábados às 9:00), o professor assignado (que poderá mudar durante o ano) e o número de utentes inscritos (cada turma tem um numero variável de vagas consoante seja do tipo natação infantil [limite 10 vagas], natação adulto [limite 20 vagas] ou hidroginástica [limite 15 vagas]);
* O Sistema de Gestão de Manutenção (SGM) visa gerir o processo de manutenção das piscinas nomeadamente a gestão de peças (spares) e consumíveis; Atualmente este processo carece de qualquer sistema, sendo solicitado material (peças ou consumíveis) à medida das necessidades levando à existência de monos (material obsoleto) e de interrupções de serviço por peças em falta (ex.: o jacuzzi esteve encerrado por 4 semanas por falta de um filtro de 5 euros);

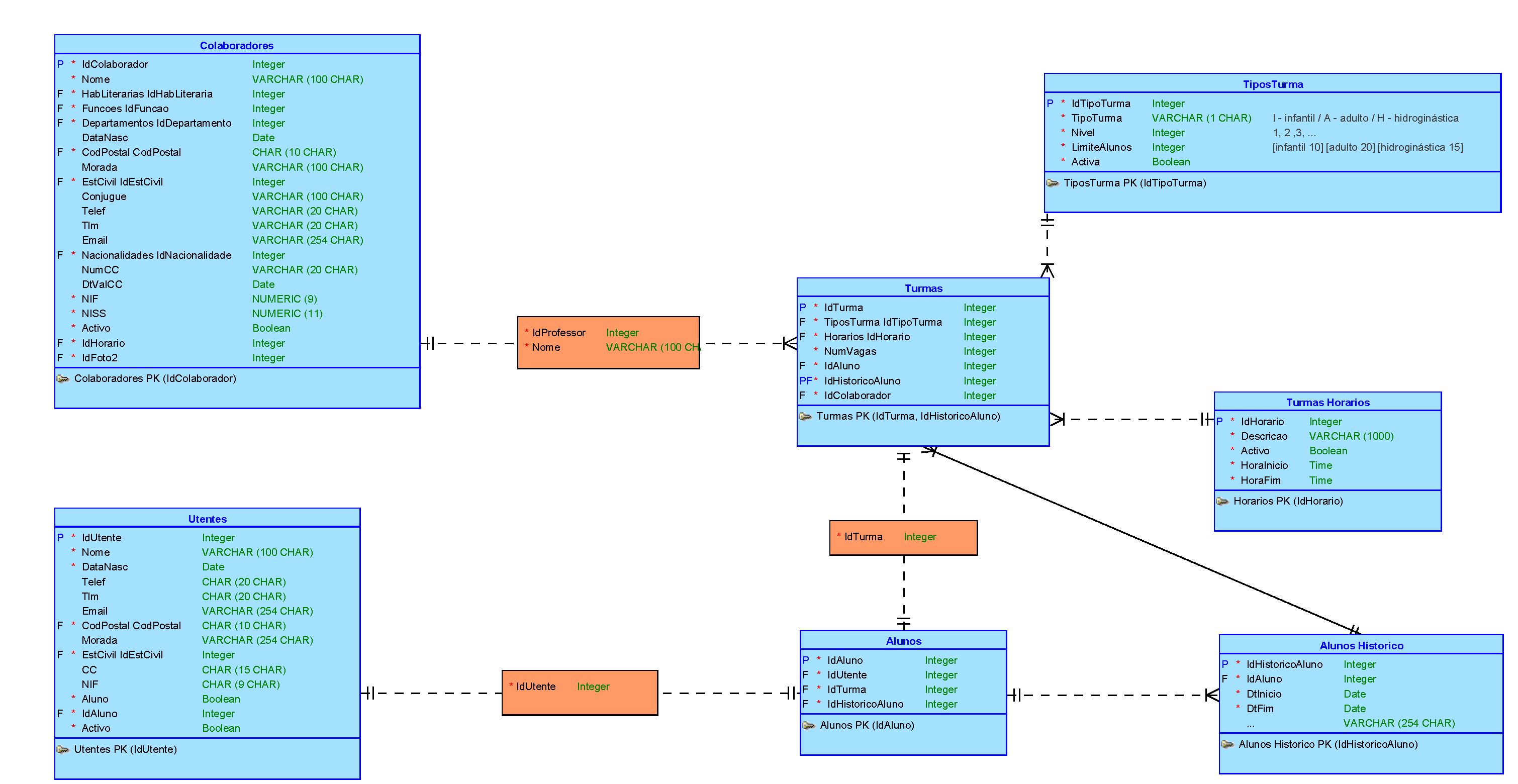
# Projeto Lógico e Projeto Relacional

## Modelo Lógico

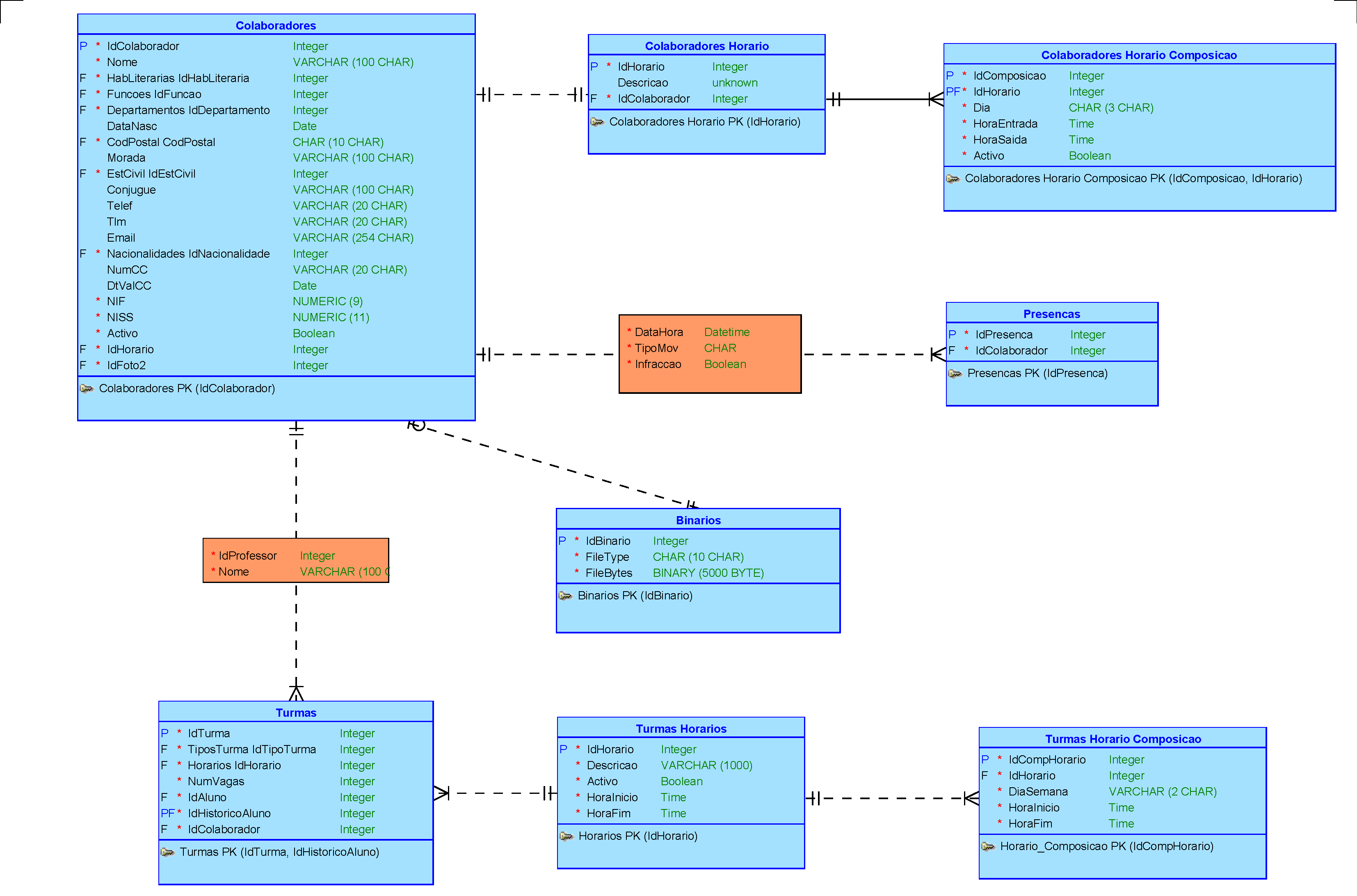
### Ata 02



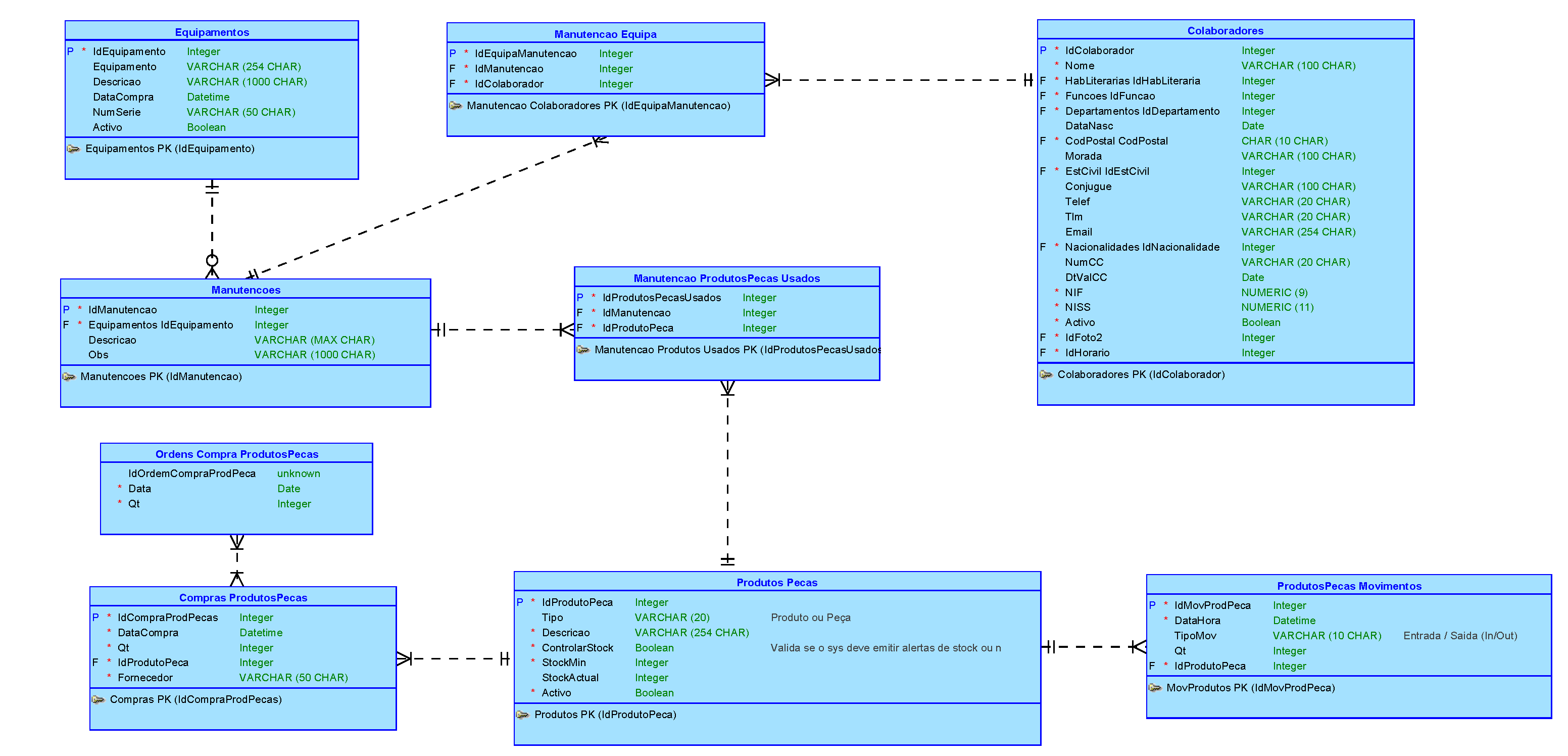
### Ata 03



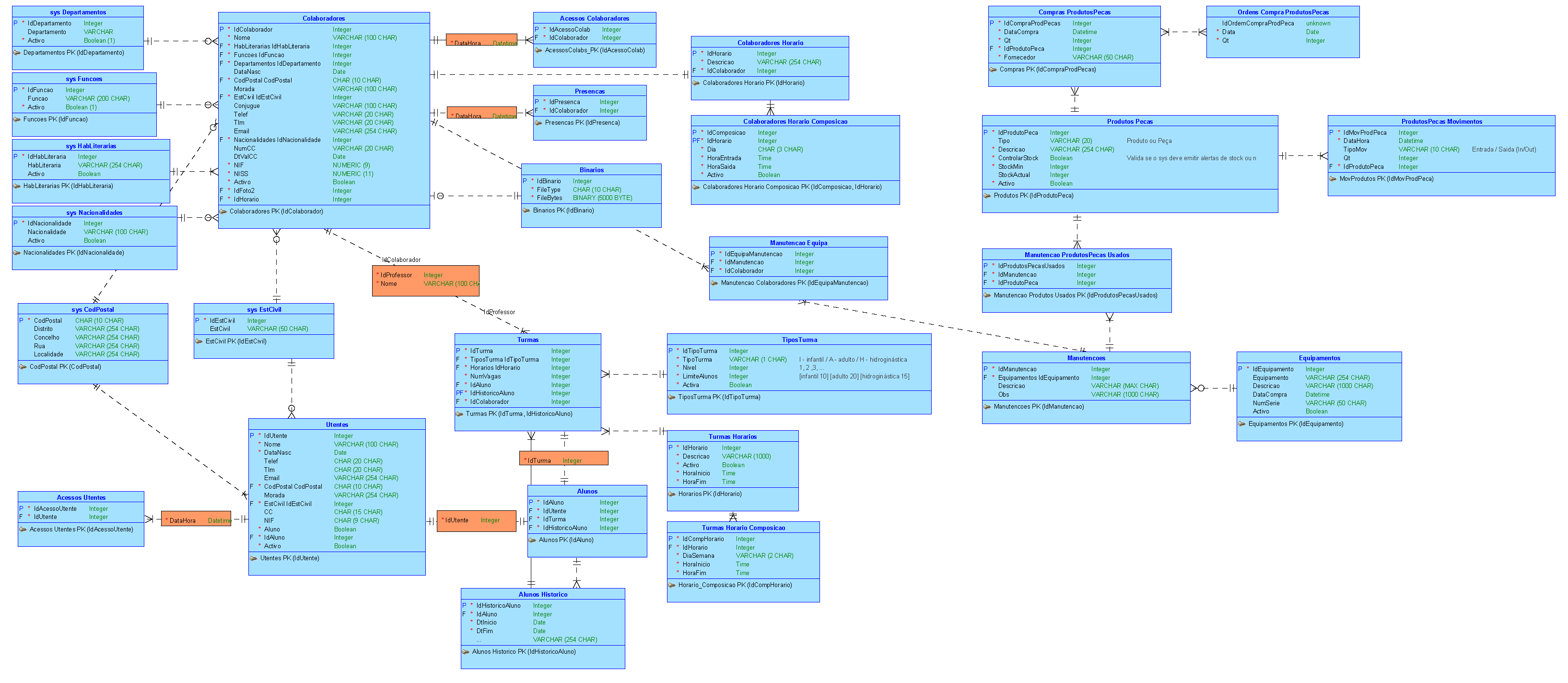
### Ata 04



### Ata 05

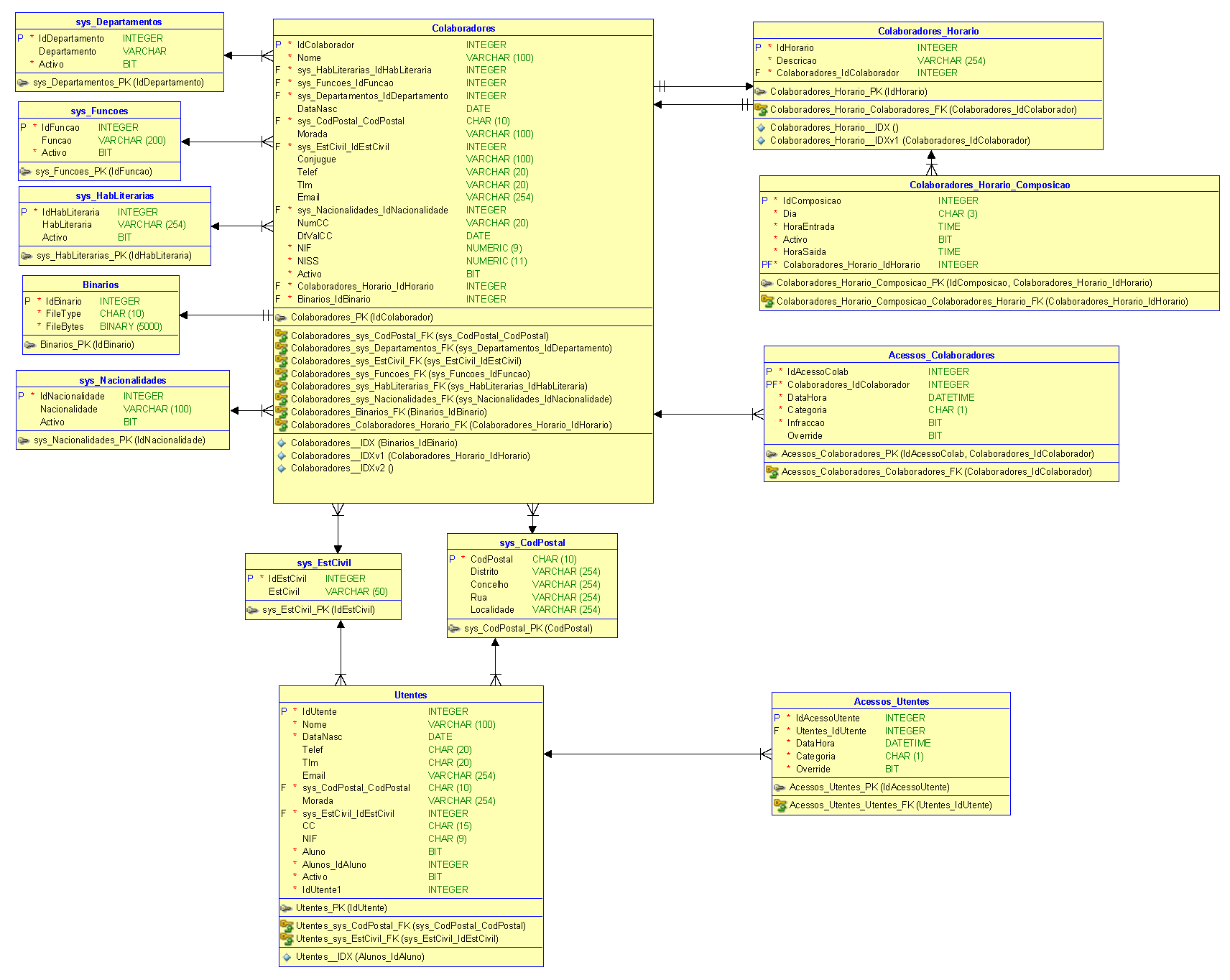


### Modelo Geral

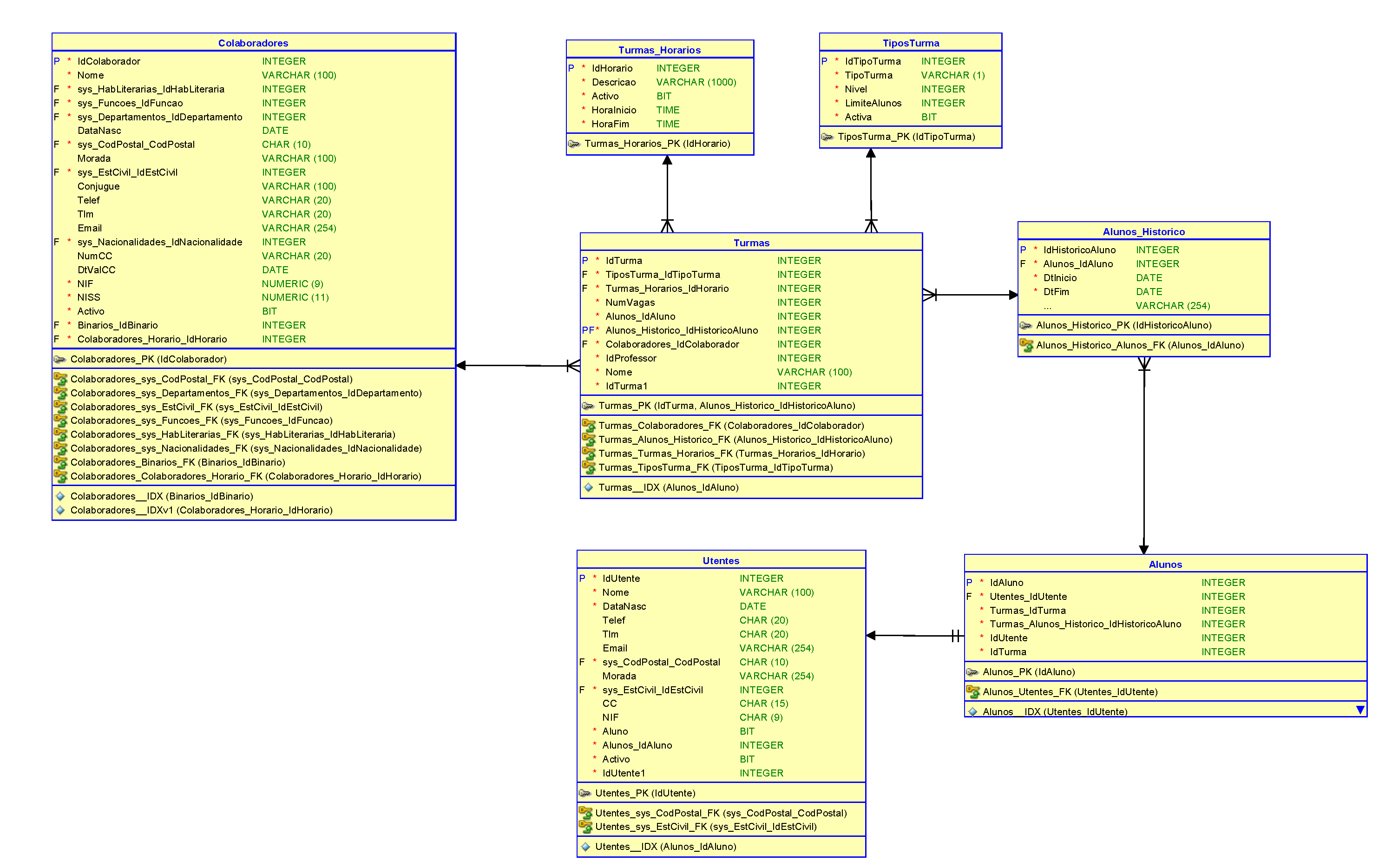


## Modelo Relacional

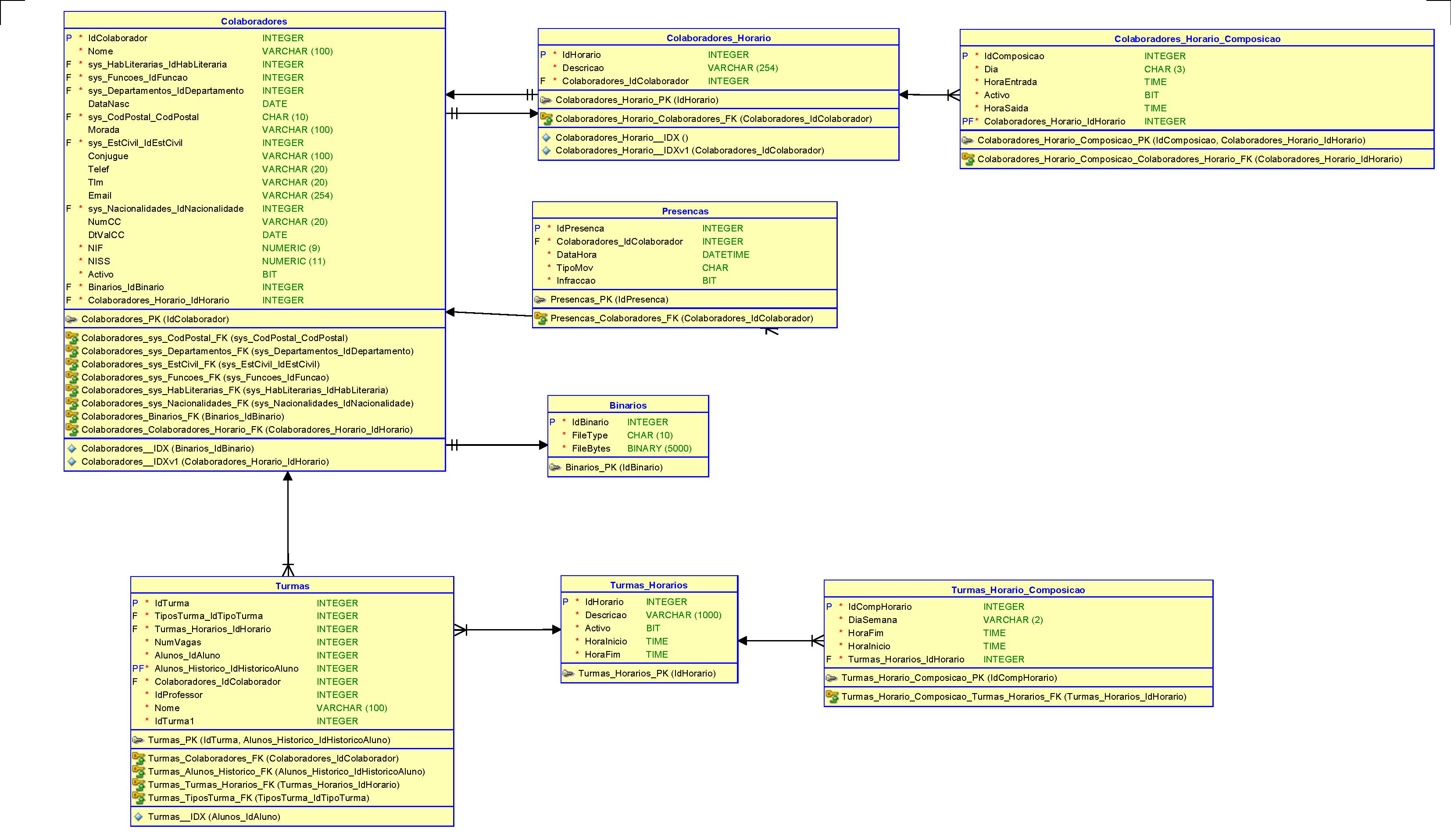
### Ata 02



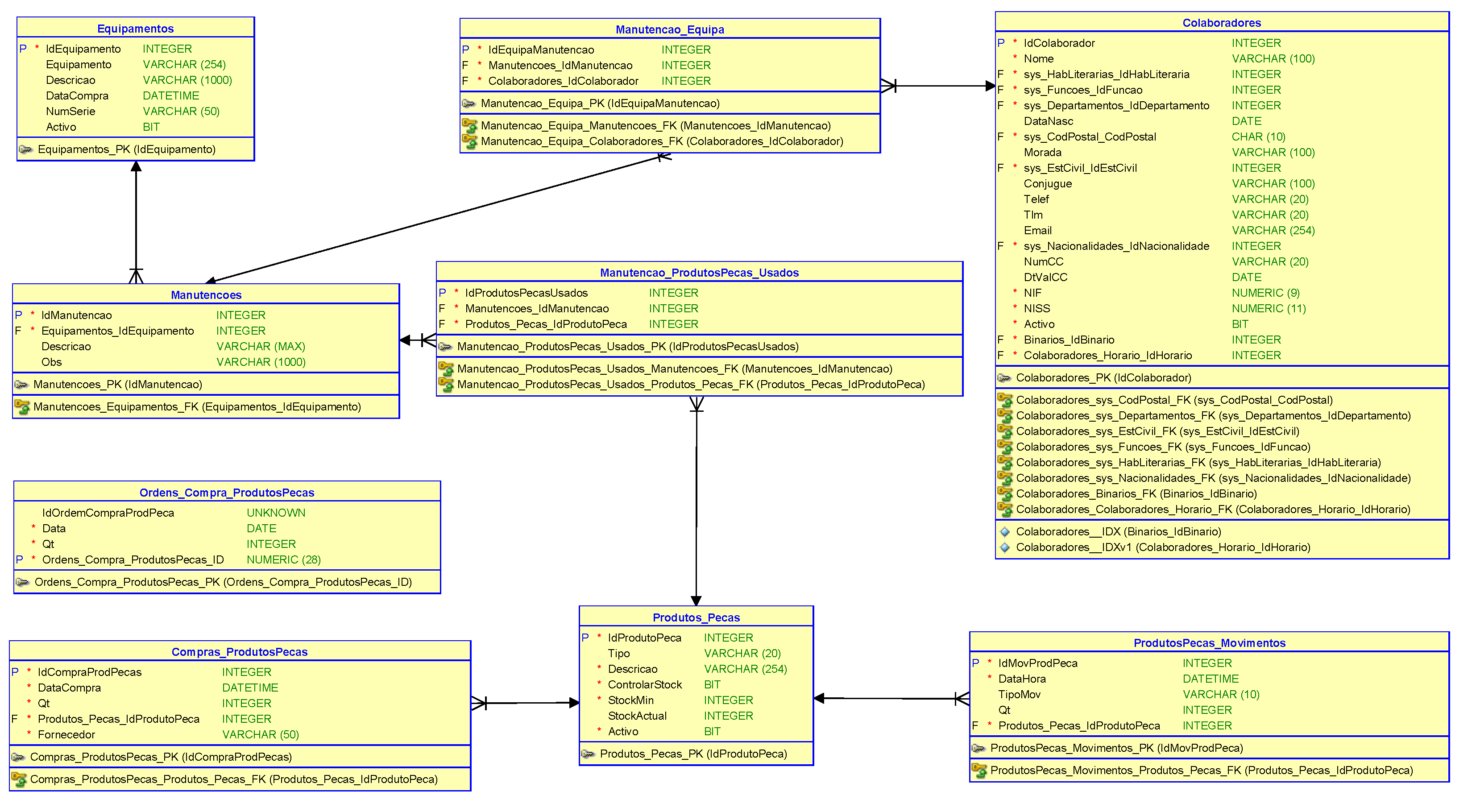
### Ata 03



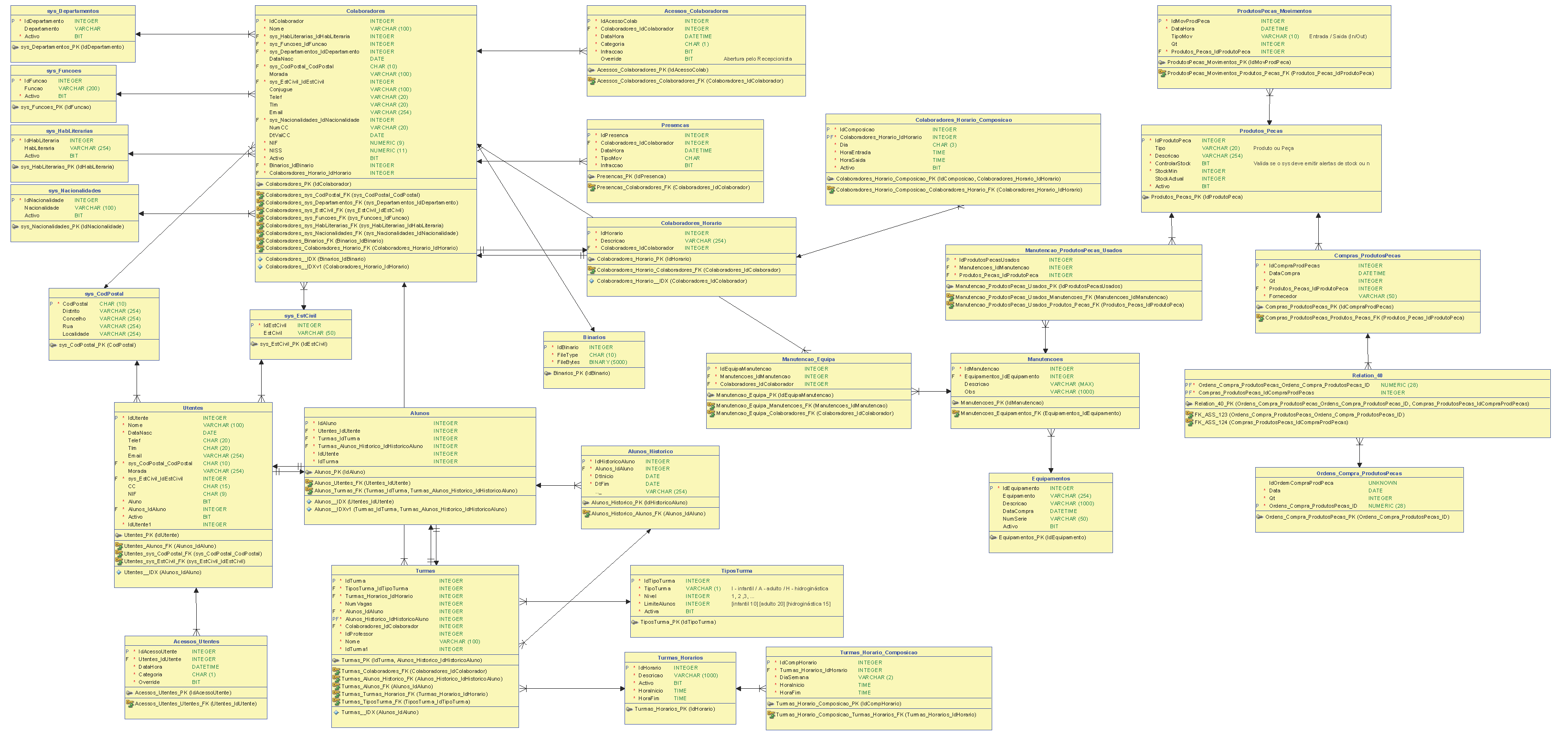
### Ata 04



### Ata 05



### Modelo Geral



## Opções Conceptuais

* A idade dos Colaboradores e Utentes/Alunos é calculada “*on the fly*”;
* A Infração será calculada com base no horário do colaborador;
* Infração é armazenada de forma a não ser necessário calcular sempre que seja pedida uma listagem dos acessos da pessoa (colaborador/utente);
* Os acessos dos Colaboradores e Utentes são registados em tabelas separadas;
* O numero de vagas calculado é calculado com base no numero máximo de inscrições e o numero de alunos da turma;
* Por forma a simplificar o tratamento dos dados postais dos utentes e/ou colaboradores foi criada uma tabela com as possíveis combinações Código Postal + Extensão Postal e as correspondentes moradas.

## Procedimentos (Stored Procedures – SP’s)

Na Base de Dados “BarcaCellos” foram desenvolvidas diversas SP’s por forma a possibilitar á equipa de desenvolvimento todas as ferramentas necessárias para inserção e manipulação de dados.

* **[dbo].[sp\_UtenteCreate]** – Insere um novo Utente na BD;

CREATE PROCEDURE [dbo].[sp\_UtenteCreate]

@Nome VARCHAR(100),

@DataNasc DATE,

@Telef VARCHAR(100),

@Tlm VARCHAR(100),

@Email VARCHAR(100),

@CodPostal VARCHAR(10),

@Morada VARCHAR(100),

@IdEstCivil INT,

@CC NUMERIC,

@NIF NUMERIC

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRY

BEGIN TRANSACTION

INSERT INTO dbo.Utentes

(

[Nome], [DataNasc],[Telef], [Tlm], [Email], [CodPostal], [Morada], [IdEstCivil], [CC], [NIF]

)

VALUES

(

@Nome, @DataNasc, @Telef, @Tlm, @Email, @CodPostal, @Morada, @IdEstCivil, @CC, @NIF

)

COMMIT TRANSACTION

SELECT CAST(1 AS BIT)

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION

SELECT CAST(0 AS BIT)

SELECT ERROR\_MESSAGE() AS 'ErrorMessage';

END CATCH;

END

GO

* **[dbo].[sp\_UtenteRead]** – Retorna todos os dados de um Utente;

(Ver Anexo 1)

* **[dbo].[sp\_UtenteUpdate]** – Actualiza os dados de um Utente;

(Ver Anexo 1)

* **[dbo].[sp\_sp\_UtenteReadListAniverProx15Dias]** – Lista os Aniversários de Utentes nos proximos 15 dias. Pode ser facilmente alterada para retornar a listagem a partir de um valor de entrada.

ALTER PROCEDURE [dbo].[sp\_UtenteReadListAniverProx15Dias]

/\*@Dias INT\*/

AS

BEGIN

SET NOCOUNT ON;

SELECT

[IdUtente] AS 'Nº Utente'

,[Nome] AS 'Utente'

,[DataNasc] AS 'Dt. Nasc.',

'Faz ' + CAST(dbo.CalcularAnos([DataNasc]) AS VARCHAR) + ' Anos' AS 'Obs.'

INTO #TempTable

FROM

dbo.Utentes

WHERE

Activo = 1

AND (DATEADD(Year, DATEPART(Year, GETDATE()) - DATEPART(Year, DataNasc), DataNasc) BETWEEN GETDATE() AND GETDATE() + 15) /\*+ @Dias)\*/

SELECT \*

FROM

#TempTable

DROP TABLE #TempTable

END

* **[dbo].[sp\_ColaboradorCreate]** – Insere um novo Colaborador na BD;

(Ver Anexo 1)

* **[dbo].[sp\_ColaboradorRead]** - Retorna os dados de um Colaborador;

(Ver Anexo 1)

* **[dbo].[sp\_ColaboradorUpdate]** - Actualiza os dados de um Colaborador;

CREATE PROCEDURE [dbo].[sp\_ColaboradorUpdate]

@IdColaborador INT,

@Nome VARCHAR(100),

@IdHabLiteraria INT,

@IdFuncao INT,

@IdDepartamento INT,

@DataNasc DATE,

@Localidade VARCHAR(100),

@CodPostal VARCHAR(50),

@Morada VARCHAR(50),

@Concelho VARCHAR(100),

@Distrito VARCHAR(100),

@IdEstCivil INT,

@Conjugue VARCHAR(100),

@Telef VARCHAR(100),

@Tlm VARCHAR(100),

@Email VARCHAR(100),

@IdNacionalidade INT,

@NumCC NUMERIC,

@DtValCC DATE,

@NIF NUMERIC(9,0),

@NISS NUMERIC(11,0),

@Activo BIT,

@IdBinario UNIQUEIDENTIFIER,

@IdHorario INT

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRY

BEGIN TRANSACTION

UPDATE dbo.Colaboradores

SET

[Nome] = @Nome,

[IdHabLiteraria] = @IdHabLiteraria,

[IdFuncao] = @IdFuncao,

[IdDepartamento] = @IdDepartamento,

[DataNasc] = @DataNasc,

[Localidade] = @Localidade,

[CodPostal] = @CodPostal,

[Morada] = @Morada,

[Concelho] = @Concelho,

[Distrito] = @Distrito,

[IdEstCivil] = @IdEstCivil,

[Conjugue] = @Conjugue,

[Telef] = @Telef,

[Tlm] = @Tlm,

[Email] = @Email,

[IdNacionalidade] = @IdNacionalidade,

[NumCC] = @NumCC,

[DtValCC] = @DtValCC,

[NIF] = @NIF,

[NISS] = @NISS,

[Activo] = @Activo,

[IdBinario] = @IdBinario,

[IdHorario] = @IdHorario

COMMIT TRANSACTION

SELECT CAST(1 AS BIT)

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION

SELECT CAST(0 AS BIT)

SELECT ERROR\_MESSAGE() AS 'ErrorMessage';

END CATCH;

END

GO

* **[dbo].[sp\_AcessosColaboradoresCreate]** - Insere um registo de acesso na BD. A SP verifica se o registo é entrada ou saida com base no ultimo registo do dia do colaborador, caso não tenha nenhum registo no dia, considera como entrada. Ao ser inserido o registo é acionado o “Trigger” [dbo].[trg\_VerificaInfraccao], que vai verificar qual o horário do Colaborador e com base no horário verifica se o registo é infracção ou não;

CREATE PROCEDURE [dbo].[sp\_AcessosColaboradoresCreate]

@IdColaborador AS INT,

@DataHora AS DATETIME,

@Categoria AS CHAR(1),

@Override AS BIT

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRY

BEGIN TRANSACTION

INSERT INTO dbo.AcessosColaboradores

(

[IdColaborador], [DataHora], [Tipo], [Categoria], [Infraccao], [Override]

)

VALUES

(

@IdColaborador,@DataHora,

CASE

WHEN (SELECT

TOP 1 AC.Tipo

FROM dbo.AcessosColaboradores AC

WHERE AC.IdColaborador = @IdColaborador

ORDER BY AC.IdAcessoColab /\*AC.DataHora\*/ DESC) = 'E' THEN 'S' ELSE 'E' END,

@Categoria,

[dbo].[VerificarInfraccaoAcesso](@IdColaborador, @DataHora),

@Override

)

COMMIT TRANSACTION

SELECT CAST('S' AS CHAR) -- Retorna Sucesso

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION

SELECT CAST('E' AS CHAR) -- Retorna Erro

SELECT ERROR\_MESSAGE() AS ErrorMessage;

END CATCH;

END

GO

* **[dbo].[sp\_AlunoCreate]** – Insere um novo Aluno na BD. O Aluno tem obrigatóriamente de ser Utente;

(Ver Anexo 1)

* **[dbo].[sp\_AlunoUpdate]** – Actualiza um Aluno.

(Ver Anexo 1)

* **[dbo].[sp\_ColaboradorReadListDadosEmFalta]** – Devolve uma lista com todos os dados em falta ou incorrectos nos registos dos Colaboradores. Verifica se todos os campos estão preenchidos e verifica se o NIF é válido;

CREATE PROCEDURE [dbo].[sp\_ColaboradorReadListDadosEmFalta]

AS

BEGIN

SET NOCOUNT ON;

DECLARE @TempTable TABLE (IdColaborador INT, Nome VARCHAR(250), Msg VARCHAR(MAX));

DECLARE @\_IdColaborador INT;

DECLARE @\_Nome VARCHAR(250);

DECLARE @\_Msg VARCHAR(MAX);

DECLARE DADOSCOLAB\_CURSOR CURSOR FOR

SELECT

IdColaborador, Nome, NULL

FROM

dbo.Colaboradores

OPEN DADOSCOLAB\_CURSOR;

FETCH NEXT FROM DADOSCOLAB\_CURSOR INTO @\_IdColaborador, @\_Nome, @\_Msg;

WHILE @@FETCH\_STATUS = 0

BEGIN

BEGIN

SET @\_Msg = '';

SET @\_Msg = CONCAT(@\_Msg, (SELECT CASE WHEN (SELECT C.IdDepartamento FROM dbo.Colaboradores C WHERE C.IdColaborador = @\_IdColaborador AND C.Activo = 1) IS NULL THEN 'Falta Departamento. ' END));

SET @\_Msg = CONCAT(@\_Msg, (SELECT CASE WHEN (SELECT C.DataNasc FROM dbo.Colaboradores C WHERE C.IdColaborador = @\_IdColaborador AND C.Activo = 1) IS NULL THEN 'Falta DT. Nasc. ' END));

SET @\_Msg = CONCAT(@\_Msg, (SELECT CASE WHEN (SELECT C.Localidade FROM dbo.Colaboradores C WHERE C.IdColaborador = @\_IdColaborador AND C.Activo = 1) IS NULL THEN 'Falta Localidade. ' END));

SET @\_Msg = CONCAT(@\_Msg, (SELECT CASE WHEN (SELECT C.CodPostal FROM dbo.Colaboradores C WHERE C.IdColaborador = @\_IdColaborador AND C.Activo = 1) IS NULL THEN 'Falta Cod. Postal. ' END));

SET @\_Msg = CONCAT(@\_Msg, (SELECT CASE WHEN (SELECT C.Morada FROM dbo.Colaboradores C WHERE C.IdColaborador = @\_IdColaborador AND C.Activo = 1) IS NULL THEN 'Falta Morada. ' END));

SET @\_Msg = CONCAT(@\_Msg, (SELECT CASE WHEN (SELECT C.Concelho FROM dbo.Colaboradores C WHERE C.IdColaborador = @\_IdColaborador AND C.Activo = 1) IS NULL THEN 'Falta Concelho. ' END));

SET @\_Msg = CONCAT(@\_Msg, (SELECT CASE WHEN (SELECT C.Distrito FROM dbo.Colaboradores C WHERE C.IdColaborador = @\_IdColaborador AND C.Activo = 1) IS NULL THEN 'Falta Distrito. ' END));

SET @\_Msg = CONCAT(@\_Msg, (SELECT CASE WHEN (SELECT C.Telef FROM dbo.Colaboradores C WHERE C.IdColaborador = @\_IdColaborador AND C.Activo = 1) IS NULL THEN 'Falta Telef. ' END));

SET @\_Msg = CONCAT(@\_Msg, (SELECT CASE WHEN (SELECT C.Tlm FROM dbo.Colaboradores C WHERE C.IdColaborador = @\_IdColaborador AND C.Activo = 1) IS NULL THEN 'Falta Tlm. ' END));

SET @\_Msg = CONCAT(@\_Msg, (SELECT CASE WHEN (SELECT C.Conjugue FROM dbo.Colaboradores C WHERE C.IdColaborador = @\_IdColaborador AND C.IdEstCivil = 1 AND C.Activo = 1) IS NULL THEN 'Falta Nome Conjugue.' END));

SET @\_Msg = CONCAT(@\_Msg, (SELECT CASE WHEN (SELECT C.Email FROM dbo.Colaboradores C WHERE C.IdColaborador = @\_IdColaborador AND C.Activo = 1) IS NULL THEN 'Falta Email. ' END));

SET @\_Msg = CONCAT(@\_Msg, (SELECT CASE WHEN (SELECT C.NumCC FROM dbo.Colaboradores C WHERE C.IdColaborador = @\_IdColaborador AND C.Activo = 1) IS NULL THEN 'Falta NumCC. ' END));

SET @\_Msg = CONCAT(@\_Msg, (SELECT CASE WHEN (SELECT C.DtValCC FROM dbo.Colaboradores C WHERE C.IdColaborador = @\_IdColaborador AND C.Activo = 1) IS NULL THEN 'Falta Dt. Val. CC. ' END));

SET @\_Msg = CONCAT(@\_Msg, (SELECT CASE WHEN (SELECT C.NIF FROM dbo.Colaboradores C WHERE C.IdColaborador = @\_IdColaborador AND C.Activo = 1) IS NULL THEN 'Falta NIF. ' END));

SET @\_Msg = CONCAT(@\_Msg, (SELECT CASE WHEN (SELECT dbo.ValidarNIF(C.NIF) FROM dbo.Colaboradores C WHERE C.IdColaborador = @\_IdColaborador AND C.NIF IS NOT NULL AND C.Activo = 1) = 0 THEN 'NIF Inválido ' END));

SET @\_Msg = CONCAT(@\_Msg, (SELECT CASE WHEN (SELECT C.NISS FROM dbo.Colaboradores C WHERE C.IdColaborador = @\_IdColaborador AND C.Activo = 1) IS NULL THEN 'Falta NISS. ' END));

SET @\_Msg = CONCAT(@\_Msg, (SELECT CASE WHEN (SELECT C.IdBinario FROM dbo.Colaboradores C WHERE C.IdColaborador = @\_IdColaborador AND C.Activo = 1) IS NULL THEN 'Falta Foto. ' END));

SET @\_Msg = CONCAT(@\_Msg, (SELECT CASE WHEN (SELECT C.IdHorario FROM dbo.Colaboradores C WHERE C.IdColaborador = @\_IdColaborador AND C.Activo = 1) IS NULL THEN 'Falta Horário. ' END));

SET @\_Msg = CONCAT(@\_Msg, (SELECT CASE WHEN (SELECT C.Conjugue FROM dbo.Colaboradores C WHERE C.IdColaborador = @\_IdColaborador AND C.IdEstCivil = 1) IS NULL THEN 'Falta Nome Conjugue.' END));

IF (@\_Msg != '')

BEGIN

INSERT INTO @TempTable

VALUES (@\_IdColaborador, @\_Nome, @\_Msg)

END

END

FETCH NEXT FROM DADOSCOLAB\_CURSOR INTO @\_IdColaborador, @\_Nome, @\_Msg;

END

CLOSE DADOSCOLAB\_CURSOR;

DEALLOCATE DADOSCOLAB\_CURSOR;

SELECT \* FROM @TempTable;

END

GO

* **[dbo].[sp\_ProdutosPecasCreate]** – Insere um novo registo de Peça/Produto na BD;

(Ver Anexo 1)

* **[dbo].[sp\_ProdutosPecasUpdate]** – Actualiza um registo de Peça/Produto na BD;

(Ver Anexo 1)

* **[dbo].[sp\_ProdutosPecasUpdateStocks]** – Actualiza o stock de uma Peça/Produto, entrada ou saida. Ao ser executada, vai acionar os triggers trg\_MovimentoStock e trg\_OrdensDeCompra;

CREATE PROCEDURE [dbo].[sp\_ProdutosPecasUpdateStocks]

@IdProdutoPeca INT,

@TipoMov VARCHAR(1), -- E - Entrada / S - Saida

@Qt INT

AS

BEGIN

BEGIN TRY

BEGIN TRANSACTION

IF (UPPER(@TipoMov) = 'S' AND ((SELECT [StockActual] FROM [dbo].[ProdutosPecas] WHERE [IdProdutoPeca] = @IdProdutoPeca) - @Qt) < 0 )

BEGIN

RAISERROR('ERRO - Valor de saida de stock não pode ser superior ao valor de stock actual.', 11, 1);

END

ELSE

BEGIN

UPDATE [dbo].[ProdutosPecas]

SET

[StockActual] =

CASE

WHEN UPPER(@TipoMov) = 'S' THEN ([StockActual] - @Qt)

WHEN UPPER(@TipoMov) = 'E' THEN ([StockActual] + @Qt)

END

WHERE

[dbo].[ProdutosPecas].[IdProdutoPeca] = @IdProdutoPeca

END

COMMIT TRANSACTION

SELECT CAST(1 AS BIT)

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION

SELECT CAST(0 AS BIT)

SELECT ERROR\_MESSAGE() AS 'Msg Erro';

END CATCH;

END

## Vistas (Views)

* **vw\_AcessosColabInfracoesUltimos30Dias** – Lista todas as infracções (Entradas/Saidas) dos colaboradores;

CREATE VIEW [dbo].[vw\_AcessosColabInfracoesUltimos30Dias]

AS

SELECT

dbo.Colaboradores.IdColaborador AS [Nº Colab.],

dbo.Colaboradores.Nome,

dbo.sys\_Departamentos.Departamento,

dbo.sys\_Funcoes.Funcao AS Função,

dbo.ColaboradoresHorarios.Descricao AS Horário,

dbo.AcessosColaboradores.DataHora,

dbo.AcessosColaboradores.Tipo

FROM

dbo.AcessosColaboradores INNER JOIN

dbo.Colaboradores ON dbo.AcessosColaboradores.IdColaborador = dbo.Colaboradores.IdColaborador INNER JOIN

dbo.sys\_Departamentos ON dbo.Colaboradores.IdDepartamento = dbo.sys\_Departamentos.IdDepartamento INNER JOIN

dbo.sys\_Funcoes ON dbo.Colaboradores.IdFuncao = dbo.sys\_Funcoes.IdFuncao INNER JOIN

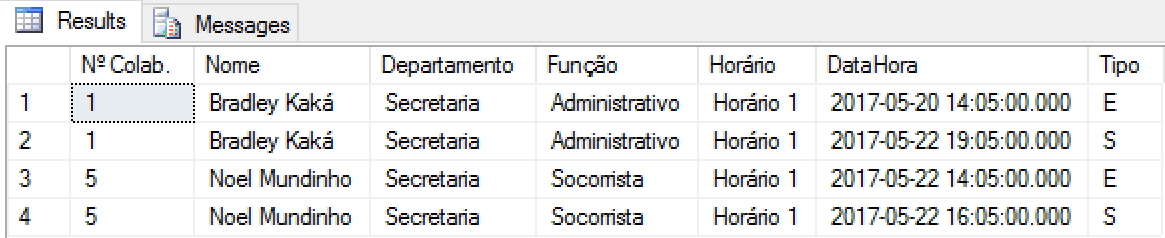
dbo.ColaboradoresHorarios ON dbo.Colaboradores.IdHorario = dbo.ColaboradoresHorarios.IdHorario

WHERE

(dbo.AcessosColaboradores.Infraccao = 1) AND

(dbo.AcessosColaboradores.DataHora BETWEEN DATEADD(DAY, - 30, GETDATE()) AND GETDATE())

GO



* **vw\_Alunos** – Lista todos os alunos activos.

CREATE VIEW [dbo].[vw\_Alunos]

AS

SELECT

dbo.Utentes.Nome, dbo.Utentes.DataNasc AS [Dt. Nasc.],

dbo.Utentes.Telef AS [Telef.], dbo.Utentes.Tlm AS TLM,

dbo.Utentes.Email, dbo.TurmasTipos.TipoTurma AS [Tipo Turma],

dbo.Turmas.Descricao AS Turma,

dbo.Colaboradores.Nome AS Professor

FROM

dbo.Alunos INNER JOIN

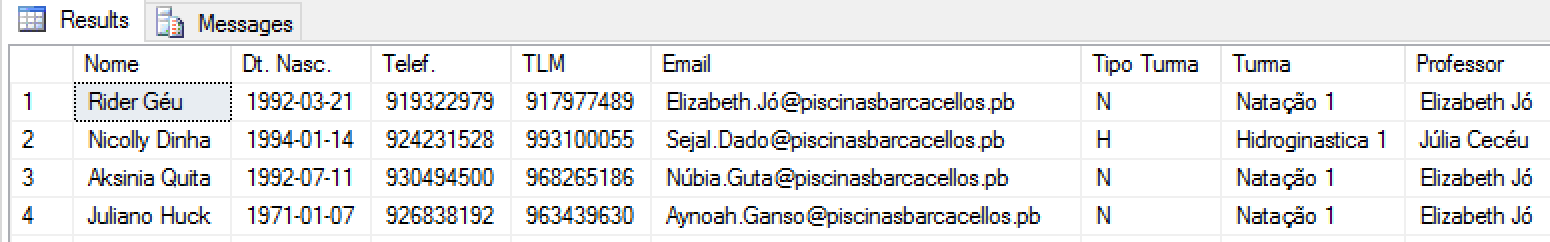
dbo.Utentes ON dbo.Alunos.IdUtente = dbo.Utentes.IdUtente INNER JOIN

dbo.Turmas ON dbo.Alunos.IdTurma = dbo.Turmas.IdTurma INNER JOIN

dbo.TurmasTipos ON dbo.Turmas.IdTipoTurma = dbo.TurmasTipos.IdTipoTurma INNER JOIN

dbo.Colaboradores ON dbo.Turmas.IdProfessor = dbo.Colaboradores.IdColaborador

GO



* **vw\_AniversariosProx7Dias** – Lista os colaboradores que fazem anos nos próximos 7 dias;

CREATE VIEW [dbo].[vw\_AniversariosProx7Dias]

AS

SELECT

TOP (100) dbo.Colaboradores.IdColaborador AS [Nº Colab.],

dbo.TratarNome(dbo.Colaboradores.Nome, 0, 0, 0) AS Nome,

dbo.Colaboradores.DataNasc,

CAST(dbo.CalcularAnos(dbo.Colaboradores.DataNasc) + 1 AS VARCHAR) + ' Anos' AS Faz,

dbo.sys\_Departamentos.Departamento,

dbo.sys\_Funcoes.Funcao

FROM

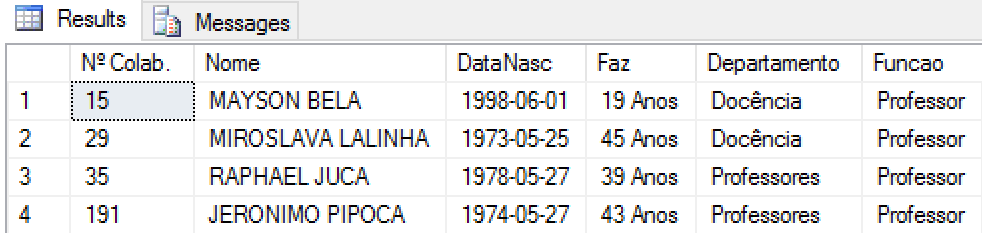
dbo.Colaboradores INNER JOIN

dbo.sys\_Funcoes ON dbo.Colaboradores.IdFuncao = dbo.sys\_Funcoes.IdFuncao INNER JOIN

dbo.sys\_Departamentos ON dbo.Colaboradores.IdDepartamento = dbo.sys\_Departamentos.IdDepartamento

WHERE

(DATEADD(Year, DATEPART(Year, GETDATE()) - DATEPART(Year, dbo.Colaboradores.DataNasc), dbo.Colaboradores.DataNasc) BETWEEN GETDATE() AND GETDATE() + 7)



* **vw\_Colaboradores** – Lista todos os dados relevantes de todos colaboradores activos;

CREATE VIEW [dbo].[vw\_Colaboradores]

AS

SELECT

dbo.Colaboradores.IdColaborador AS [Nº Colab.],

dbo.TratarNome(dbo.Colaboradores.Nome, 0, 0, 1) AS Nome,

dbo.Colaboradores.DataNasc AS [Dt. Nasc.],

dbo.CalcularAnos(dbo.Colaboradores.DataNasc) AS Idade,

dbo.sys\_EstCivil.EstCivil AS [Est. Civil],

dbo.Colaboradores.Conjugue,

dbo.Colaboradores.Morada,

dbo.Colaboradores.Localidade,

dbo.Colaboradores.CodPostal AS [Cod. Postal],

dbo.Colaboradores.Telef, dbo.Colaboradores.Tlm,

LOWER(dbo.TratarNome(dbo.Colaboradores.Email, 0, 0, 1)) AS Email,

dbo.sys\_Nacionalidades.Nacionalidade,

dbo.Colaboradores.NumCC AS [Nº CC],

dbo.Colaboradores.DtValCC AS [Dt. Val. CC.],

dbo.Colaboradores.NIF,

dbo.Colaboradores.NISS,

dbo.sys\_HabLiterarias.HabLiteraria AS [Hab. Lit.],

dbo.sys\_Departamentos.Departamento,

dbo.sys\_Funcoes.Funcao AS Função,

dbo.ColaboradoresHorarios.Descricao AS Horário

FROM

dbo.Colaboradores INNER JOIN

dbo.sys\_Departamentos ON dbo.Colaboradores.IdDepartamento = dbo.sys\_Departamentos.IdDepartamento INNER JOIN

dbo.sys\_EstCivil ON dbo.Colaboradores.IdEstCivil = dbo.sys\_EstCivil.IdEstCivil INNER JOIN

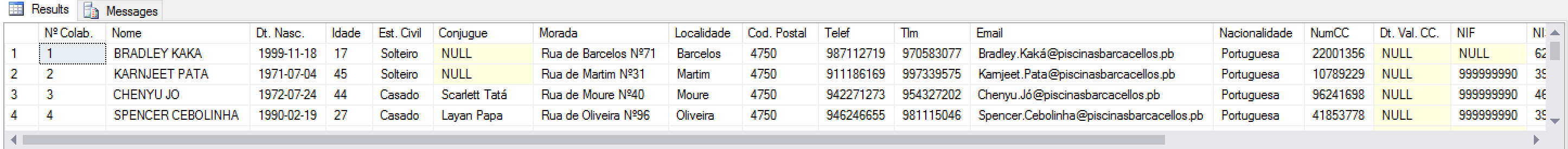
dbo.sys\_Funcoes ON dbo.Colaboradores.IdFuncao = dbo.sys\_Funcoes.IdFuncao INNER JOIN

dbo.sys\_HabLiterarias ON dbo.Colaboradores.IdHabLiteraria = dbo.sys\_HabLiterarias.IdHabLiteraria INNER JOIN

dbo.sys\_Nacionalidades ON dbo.Colaboradores.IdNacionalidade = dbo.sys\_Nacionalidades.IdNacionalidade INNER JOIN

dbo.ColaboradoresHorarios ON dbo.ColaboradoresHorarios.IdHorario = dbo.Colaboradores.IdHorario

GO



* **vw\_Professores** – Lista todos os professores activos;

CREATE VIEW [dbo].[vw\_Professores]

AS

SELECT

dbo.Colaboradores.Nome,

dbo.Colaboradores.Telef,

dbo.Colaboradores.Tlm,

dbo.Colaboradores.Email,

dbo.sys\_Departamentos.Departamento,

dbo.sys\_HabLiterarias.HabLiteraria AS [Hab. Lit.],

dbo.sys\_Funcoes.Funcao AS Função

FROM

dbo.Colaboradores INNER JOIN

dbo.sys\_Departamentos ON dbo.Colaboradores.IdDepartamento = dbo.sys\_Departamentos.IdDepartamento INNER JOIN

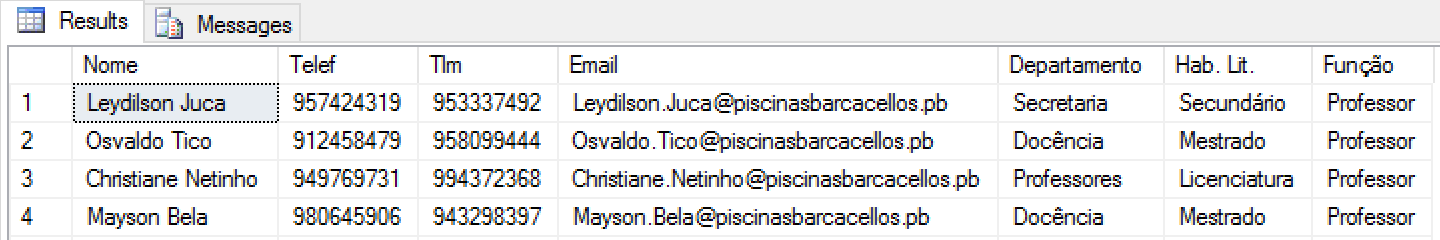
dbo.sys\_HabLiterarias ON dbo.Colaboradores.IdHabLiteraria = dbo.sys\_HabLiterarias.IdHabLiteraria INNER JOIN

dbo.sys\_Funcoes ON dbo.Colaboradores.IdFuncao = dbo.sys\_Funcoes.IdFuncao

WHERE

(dbo.Colaboradores.IdFuncao = 1)

GO



* **vw\_Utentes** – Lista todos os utentes activos;

CREATE VIEW [dbo].[vw\_Utentes]

AS

SELECT

dbo.Utentes.IdUtente AS [Nº Utente],

dbo.Utentes.Nome,

dbo.Utentes.DataNasc AS [Dt. Nasc.],

dbo.Utentes.Telef,

dbo.Utentes.Tlm,

dbo.Utentes.Email,

dbo.Utentes.CodPostal AS [Cod. Postal],

dbo.Utentes.Morada,

dbo.Utentes.CC AS [Nº CC],

dbo.Utentes.NIF,

CASE WHEN dbo.Utentes.Aluno = 1 THEN 'Sim' ELSE 'Não' END AS Aluno

FROM

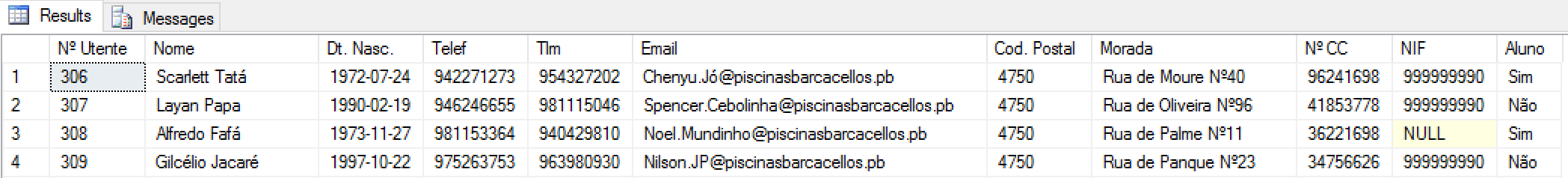
dbo.Utentes INNER JOIN

dbo.sys\_EstCivil ON dbo.Utentes.IdEstCivil = dbo.sys\_EstCivil.IdEstCivil

WHERE

(dbo.Utentes.Activo = 1)

GO



## Funções (Functions)

* **dbo.CalcularAnos** – Efectua o calculo e retorna a diferença de anos entre uma data de entrada e a data actual (calculo de idade, antiguidade na empresa, etc.).

Ex. de uso:

* + - SELECT dbo.CalcularAnos(dbo.Colaboradores.DataNasc);
    - SELECT dbo.CalcularAnos(‘2000-01-01’);

CREATE FUNCTION [dbo].[CalcularAnos]

(

@Data AS DATETIME

)

RETURNS INT

AS

BEGIN

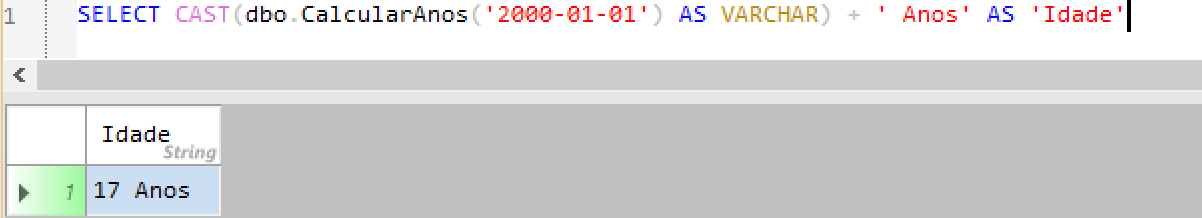
DECLARE @Anos INT;

SET @Anos = ((CAST(CONVERT(VARCHAR(8),GETDATE(),112) AS INT)-(CAST(CONVERT(VARCHAR(8), @Data,112) AS INT)))/10000)

RETURN @Anos

END

GO



* **dbo.TratarNome** – Dada uma string de entrada, pode efectuar as seguintes operações:
  + Capitaliza todas as letras;
  + Capitaliza a primeira letra de cada palavra;
  + Remove os diacriticos de uma string de entrada;

Ex. de uso:

* + - SELECT dbo.TratarNome(dbo.Colaboradores.Nome, 0, 0, 1);
    - SELECT dbo.TratarNome(‘Flávio Láu’, 0, 0, 1);

CREATE FUNCTION [dbo].[TratarNome]

(

@Input VARCHAR(MAX),

@ToUpper BIT,

@ToProperCase BIT,

@RemoveDiacritics BIT

)

RETURNS VARCHAR(MAX)

AS

BEGIN

DECLARE @Output VARCHAR(MAX)

SET @Output = RTRIM(LTRIM(@Input));

IF (@ToUpper = 1 AND @Output != '')

BEGIN

SET @Output = (SELECT UPPER(@Output))

END

IF (@RemoveDiacritics = 1 AND @Output != '')

BEGIN

SET @Output = (SELECT @Output Collate SQL\_Latin1\_General\_CP1253\_CI\_AI)

END

IF (@ToProperCase = 1 AND @Output != '')

BEGIN

DECLARE @Pos1 INT = 1

DECLARE @Pos2 INT

SET @Output = LOWER(@Output)

WHILE (1 = 1)

BEGIN

SET @Output = STUFF(@Output, @Pos1, 1, UPPER(SUBSTRING(@Output, @Pos1, 1)))

SET @Pos2 = PATINDEX('%[- ''.)(]%', SUBSTRING(@Output, @Pos1, 500))

SET @Pos1 += @Pos2

IF (ISNULL(@Pos2, 0) = 0 or @Pos1 > LEN(@Output))

BEGIN

BREAK

END

END

END

RETURN @Output

END

GO



* **dbo.ValidarNIF** – Dado um valor de entrada, verifica se o é um NIF é válido. Devolve 1 se for válido e 0 se inválido.

Ex. de uso:

* + - SELECT dbo.ValidarNIF(dbo.Colaboradores.NIF);
    - SELECT dbo.ValidarNIF(000000000);

CREATE FUNCTION [dbo].[ValidarNIF]

(

@NIF VARCHAR(50)

)

RETURNS bit

AS

BEGIN

DECLARE @Output BIT;

SET @Output = 0;

IF(ISNUMERIC(@NIF) = 0)

BEGIN

SET @Output = 0

RETURN @Output

END

--Tem de ter 9 digitos e o digito inicial tem de ser 1,2,5,6,7,8,9

IF (@NIF < 100000000 OR @NIF > 999999999) OR (@NIF > 299999999 AND @NIF < 500000000)

BEGIN

SET @Output = 0

RETURN @Output

END

DECLARE @NIFString VARCHAR(9)

SET @NIFString = CAST(@NIF AS VARCHAR(9))

DECLARE @Control INT

SET @Control = 9 \* (cast(SUBSTRING(@NIFString , 1, 1) AS INT)) + 8\*(cast(SUBSTRING(@NIFString , 2, 1) AS INT )) +

7 \*(cast(SUBSTRING(@NIFString , 3, 1) AS INT )) + 6\*(CAST(SUBSTRING(@NIFString , 4, 1) AS INT )) +

5 \*(cast(SUBSTRING(@NIFString , 5, 1) AS INT )) + 4\*(CAST(SUBSTRING(@NIFString , 6, 1) AS INT )) +

3 \*(cast(SUBSTRING(@NIFString , 7, 1) AS INT )) + 2\*(CAST(SUBSTRING(@NIFString , 8, 1) AS INT ))

DECLARE @Remainder INT

SET @Remainder = @Control % 11

IF @Remainder < 2

BEGIN

IF CAST(SUBSTRING(@NIFString, 9 , 1) AS INT) = 0

BEGIN

SET @Output = 1

END

ELSE

BEGIN

SET @Output = 0

END

END

ELSE

BEGIN

IF (11 - @Remainder) = CAST(SUBSTRING(@NIFString, 9, 1) AS INT)

BEGIN

SET @Output = 1

END

ELSE

BEGIN

SET @Output = 0

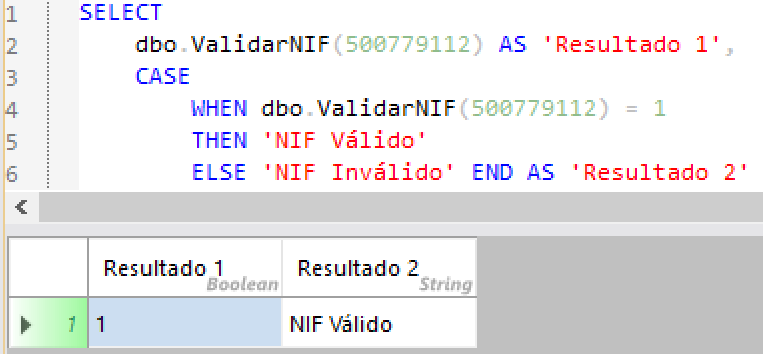
RETURN @Output

END

END

RETURN @Output

END



## Gatilhos (Triggers)

* **[dbo].[trg\_MovimentoStock]** – Ao serem efectuados movimentos nos stocks, regista na tabela “dbo.ProdutosPecasMovimentos” o tipo de movimento (“E” – Entrada ou “S” - Saida), a quantidade e o ID da peça/produto;

CREATE TRIGGER [dbo].[trg\_MovimentoStock] ON [dbo].[ProdutosPecas] AFTER UPDATE

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO dbo.ProdutosPecasMovimentos (TipoMov, Qt, IdProdutoPeca)

SELECT

CASE

WHEN PP.StockActual > D.StockActual THEN 'E' ELSE 'S'

END,

CASE

WHEN PP.StockActual > D.StockActual THEN (SELECT PP.StockActual - D.StockActual) ELSE (SELECT D.StockActual - PP.StockActual)

END,

PP.IdProdutoPeca

FROM

dbo.ProdutosPecas PP INNER JOIN

DELETED D ON PP.IdProdutoPeca = D.IdProdutoPeca

END

GO

ALTER TABLE [dbo].[ProdutosPecas] ENABLE TRIGGER [trg\_MovimentoStock]

GO

* **[dbo].[trg\_OrdensDeCompra]** - Ao serem efectuados movimentos nos stocks, verifica se o stock actual é inferior ao stock minimo e caso o campo "ControlarStock" seja “true”, insere uma ordem de compra na tabela "dbo.ProdutosPecasOrdensCompra".

CREATE TRIGGER [dbo].[trg\_OrdensDeCompra] ON [dbo].[ProdutosPecas] AFTER UPDATE

AS

BEGIN

SET NOCOUNT ON;

IF ((SELECT

CASE WHEN PP.StockActual < PP.StockMin AND PP.ControlarStock = 1 THEN 1 ELSE 0 END

FROM

dbo.ProdutosPecas PP INNER JOIN

DELETED D ON PP.IdProdutoPeca = D.IdProdutoPeca) = 1)

BEGIN

INSERT INTO dbo.ProdutosPecasOrdensCompra

(

IdProdutoPeca, Qt

)

SELECT

PP.IdProdutoPeca,

PP.StockMin

FROM

dbo.ProdutosPecas PP INNER JOIN

DELETED D ON PP.IdProdutoPeca = D.IdProdutoPeca

END

END