ESCOLA SUPERIOR DE TECNOLOGIA IPCA

TRABALHO PRÁTICO

ARMAZENAMENTO E ACESSO A DADOS

00.2.1 PLANO DE PROJETO

```sql
/* Elementos do Grupo */
Select
    NumAluno AS 'Nº Aluno',
    NomeAluno AS 'Nome Aluno',
    'a' + CAST (NumAluno AS VARCHAR) + '@alunos.ipca.pt' AS 'Email'
From
    [IPCA].[dbo].[Alunos]
Where
    NumAluno in ('3664', '11201', '12555')
Order By
    NumAluno
```

	Nº Aluno (Int32)	Nome Aluno (String)	Email (String)
1	3664	Rui Costa	a3664@alunos.ipca.pt
2	11201	Ângelo Ferreira	a11201@alunos.ipca.pt
3	12555	Miguel Pimenta	a12555@alunos.ipca.pt

Índice

Objetivo 4
Âmbito 4
Metodologia 4
Análise de Requisitos 4
Projeto Concetual 5
Projeto lógico 5
Projeto físico 5
Equipa 6
Plano do Projeto 6
Fatores Críticos de Sucesso 7
Análise de Requisitos 7
Regras de Negócio 7
Requisitos Funcionais 7
Requisitos Técnicos 10
Projeto Lógico e Projeto Relacional 12
Modelo Lógico 12
Ata 02 12
Ata 03 13
Ata 04 14
Ata 05 16
Modelo Geral 17
Modelo Relacional 18
Ata 02 18
Ata 03 19
Ata 04 21
Ata 05 22
Modelo Geral 23
Opções Conceptuais 24

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* baixo ou 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

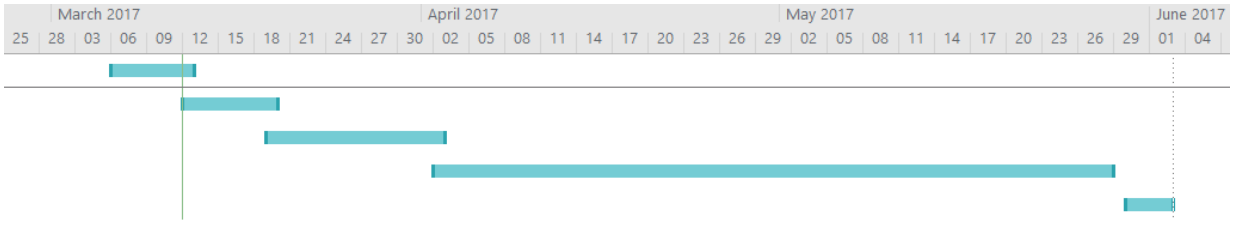
PARAMETRIZAÇÃO DO MODELO RELACIONAL EM FUNÇÃO DO QUE SE PRETENDE DA SUA UTILIZAÇÃO

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

Task Name	Duration	Start	Finish
Plano de projeto	6 days	Sun 05/03/17	Sun 12/03/17
Análise de Requisitos	7 days	Sun 12/03/17	Sun 19/03/17
Projeto conceptual e lógico	12 days	Sun 19/03/17	Sun 02/04/17
Projeto físico	42 days	Sun 02/04/17	Sun 28/05/17
Apresentação	4 days	Tue 30/05/17	Fri 02/06/17



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);

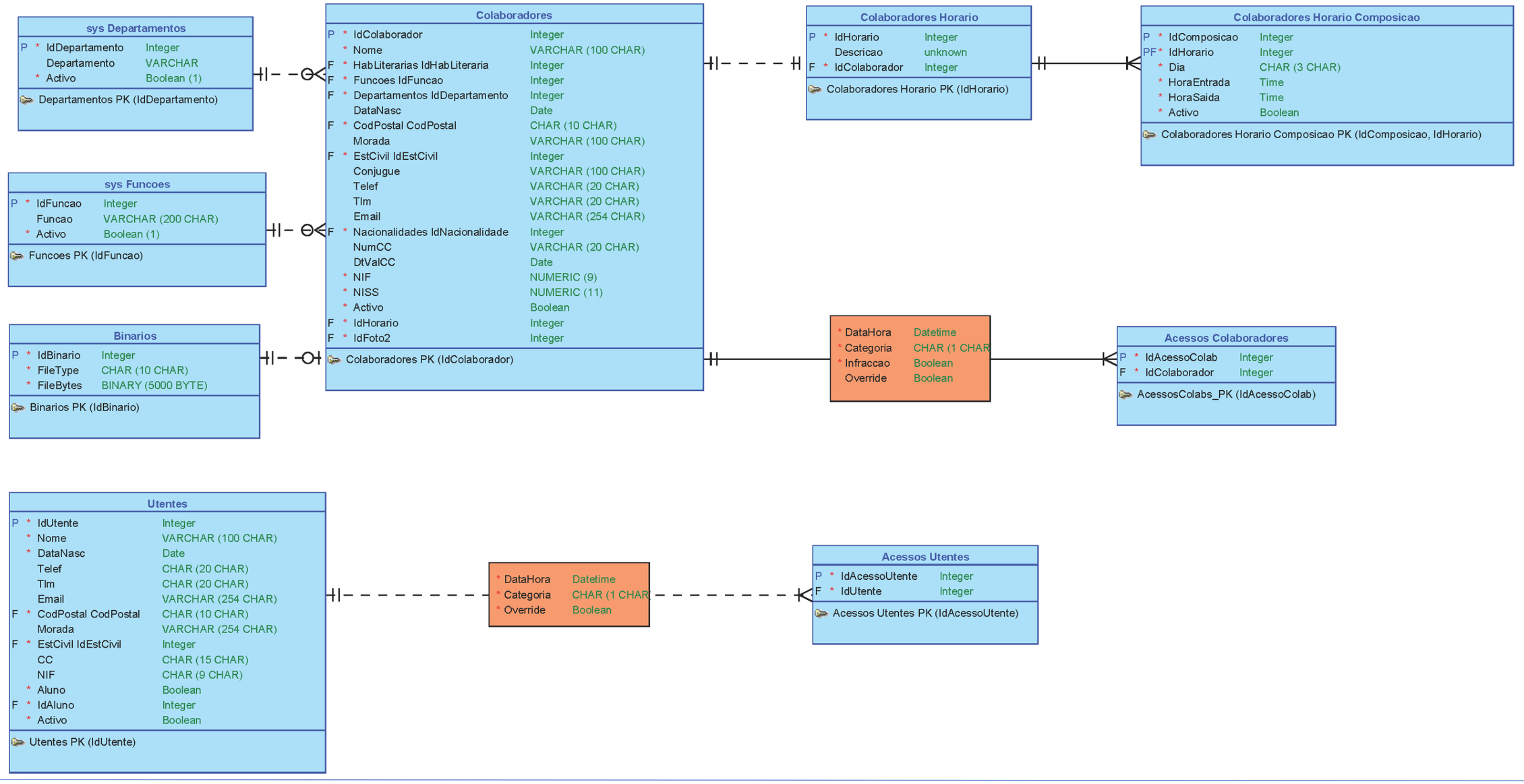
- O SISTEMA DE GESTÃO DE TURMAS (SGT) ESTÁ ATUALMENTE IMPLEMENTADO POR VIA DE FORMULÁRIOS EM PAPEL ONDE CONSTAM AS AULAS QUE A TURMA TEM E RESPETIVOS HORÁRIOS (EX.: 4AS ÀS 21:00 E SÁBADOS ÀS 9:00), O PROFESSOR ASSIGNADO (QUE PODERÁ MUDAR DURANTE O ANO) E O NÚMERO DE UTENTES INSCRITOS (CADA TURMA TEM 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);

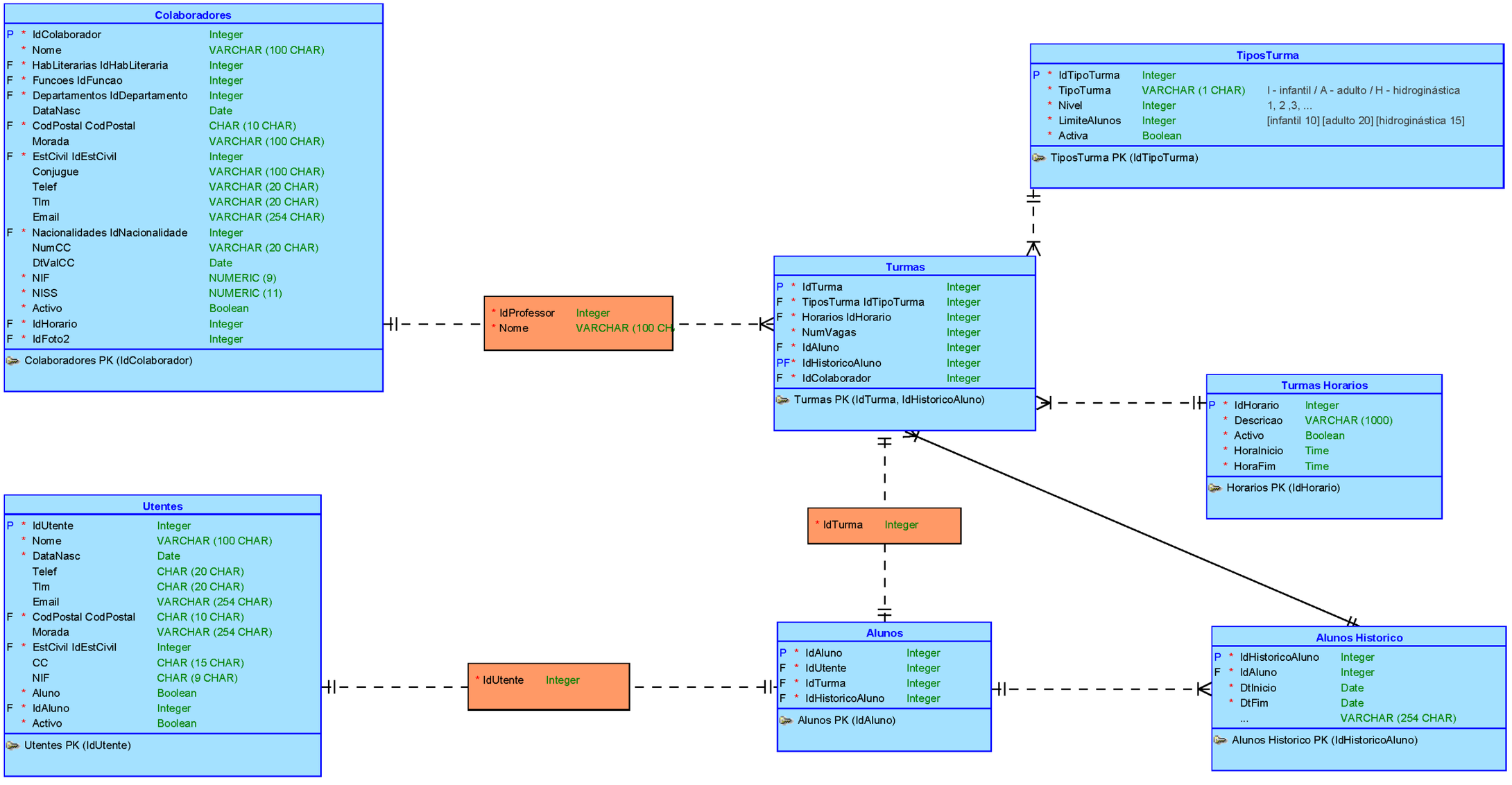
Projeto Lógico e Projeto Relacional

Modelo Lógico

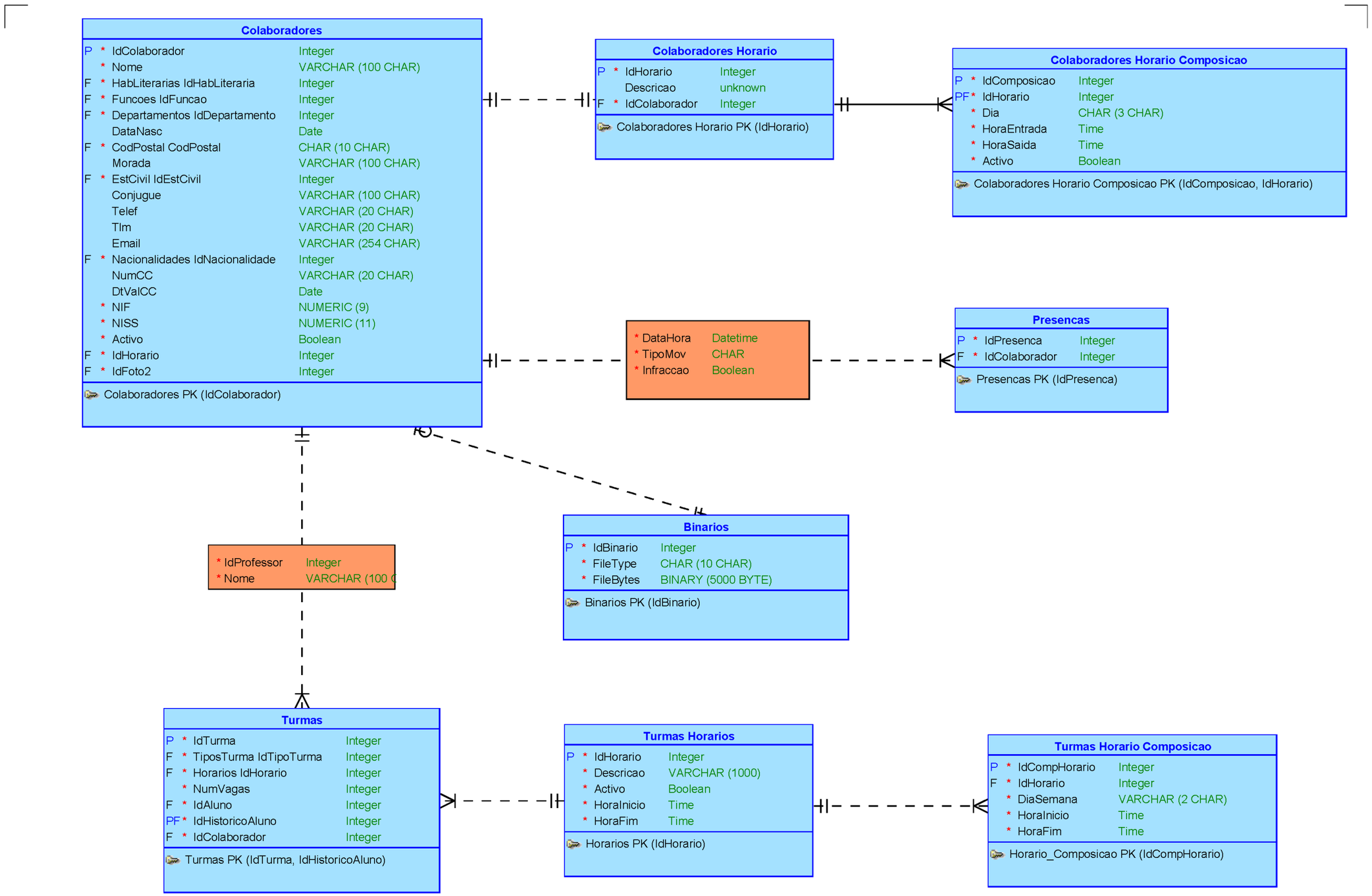
Ata 02



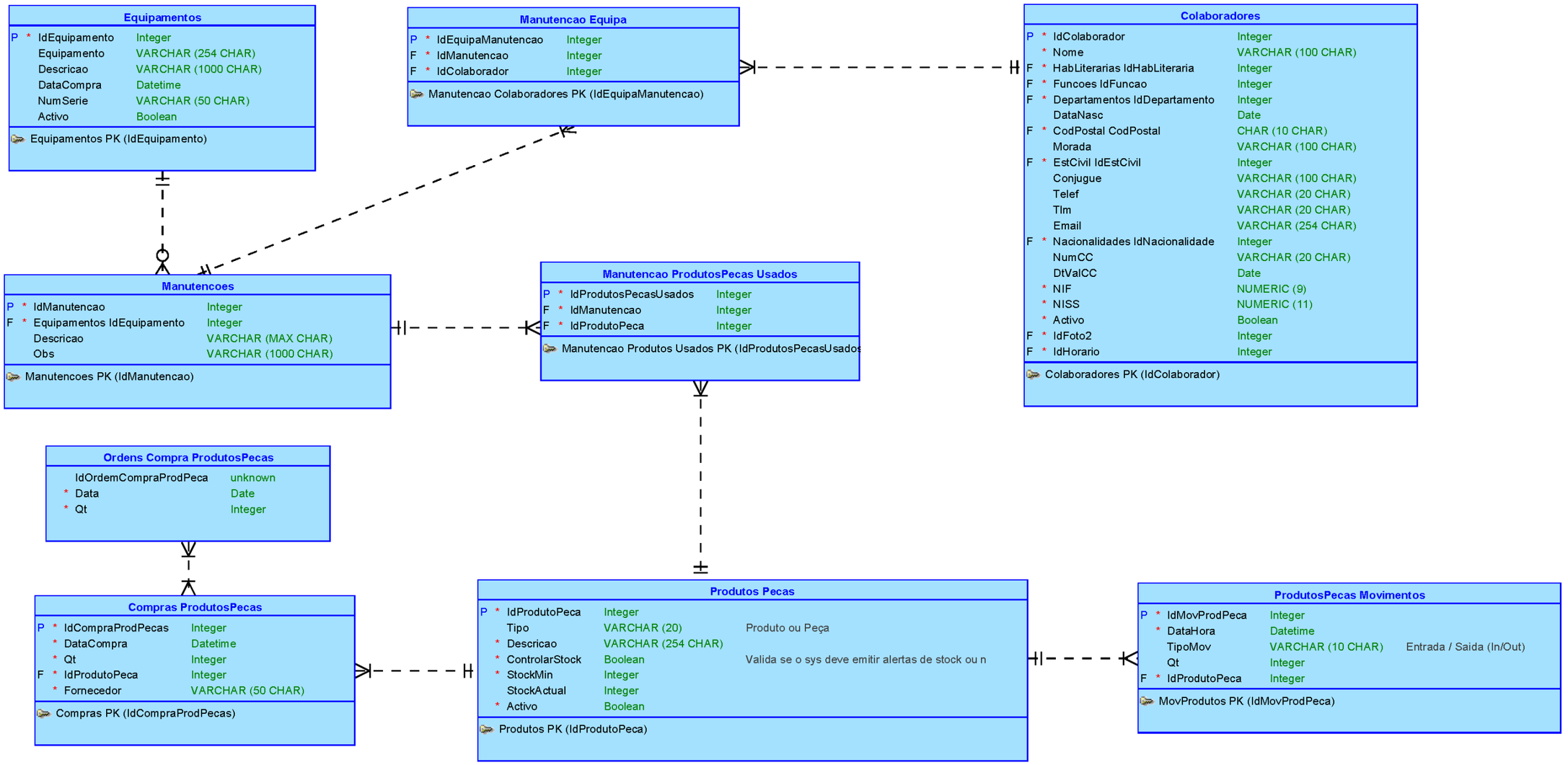
Ata 03


Colaboradores
P * IdColaborador Integer
* Nome VARCHAR (100 CHAR)
F * HabLiterarias IdHabLiteraria Integer
F * Funcoes IdFuncao Integer
F * Departamentos IdDepartamento Integer
DataNasc Date
F * CodPostal CodPostal CHAR (10 CHAR)
Morada VARCHAR (100 CHAR)
F * EstCivil IdEstCivil Integer
Conjugue VARCHAR (100 CHAR)
Telef VARCHAR (20 CHAR)
Tlm VARCHAR (20 CHAR)
Email VARCHAR (254 CHAR)
F * Nacionalidades IdNacionalidade Integer
NumCC VARCHAR (20 CHAR)
DtValCC Date
* NIF NUMERIC (9)
* NISS NUMERIC (11)
* Activo Boolean
F * IdHorario Integer
F * IdFoto2 Integer
Colaboradores PK (IdColaborador)
* IdProfessor Integer
* Nome VARCHAR (100 CH
TiposTurma
P * IdTipoTurma Integer
* TipoTurma VARCHAR (1 CHAR) I - infantil / A - adulto / H - hidroginástica
* Nivel Integer 1, 2 ,3, ...
* LimiteAlunos Integer [infantil 10] [adulto 20] [hidroginástica 15]
* Activa Boolean
TiposTurma PK (IdTipoTurma)
Turmas
P * IdTurma Integer
F * TiposTurma IdTipoTurma Integer
F * Horarios IdHorario Integer
* NumVagas Integer
F * IdAluno Integer
PF * IdHistoricoAluno Integer
F * IdColaborador Integer
Turmas PK (IdTurma, IdHistoricoAluno)
Turmas Horarios
P * IdHorario Integer
* Descricao VARCHAR (1000)
* Activo Boolean
* HoraInicio Time
* HoraFim Time
Horarios PK (IdHorario)
* IdTurma Integer
Utentes
P * IdUtente Integer
* Nome VARCHAR (100 CHAR)
* DataNasc Date
Telef CHAR (20 CHAR)
Tlm CHAR (20 CHAR)
Email VARCHAR (254 CHAR)
F * CodPostal CodPostal CHAR (10 CHAR)
Morada VARCHAR (254 CHAR)
F * EstCivil IdEstCivil Integer
CC CHAR (15 CHAR)
NIF CHAR (9 CHAR)
* Aluno Boolean
F * IdAluno Integer
* Activo Boolean
Utentes PK (IdUtente)
* IdUtente Integer
Alunos
P * IdAluno Integer
F * IdUtente Integer
F * IdTurma Integer
F * IdHistoricoAluno Integer
Alunos PK (IdAluno)
Alunos Historico
P * IdHistoricoAluno Integer
F * IdAluno Integer
* DtInicio Date
* DtFim Date
... VARCHAR (254 CHAR)
Alunos Historico PK (IdHistoricoAluno)

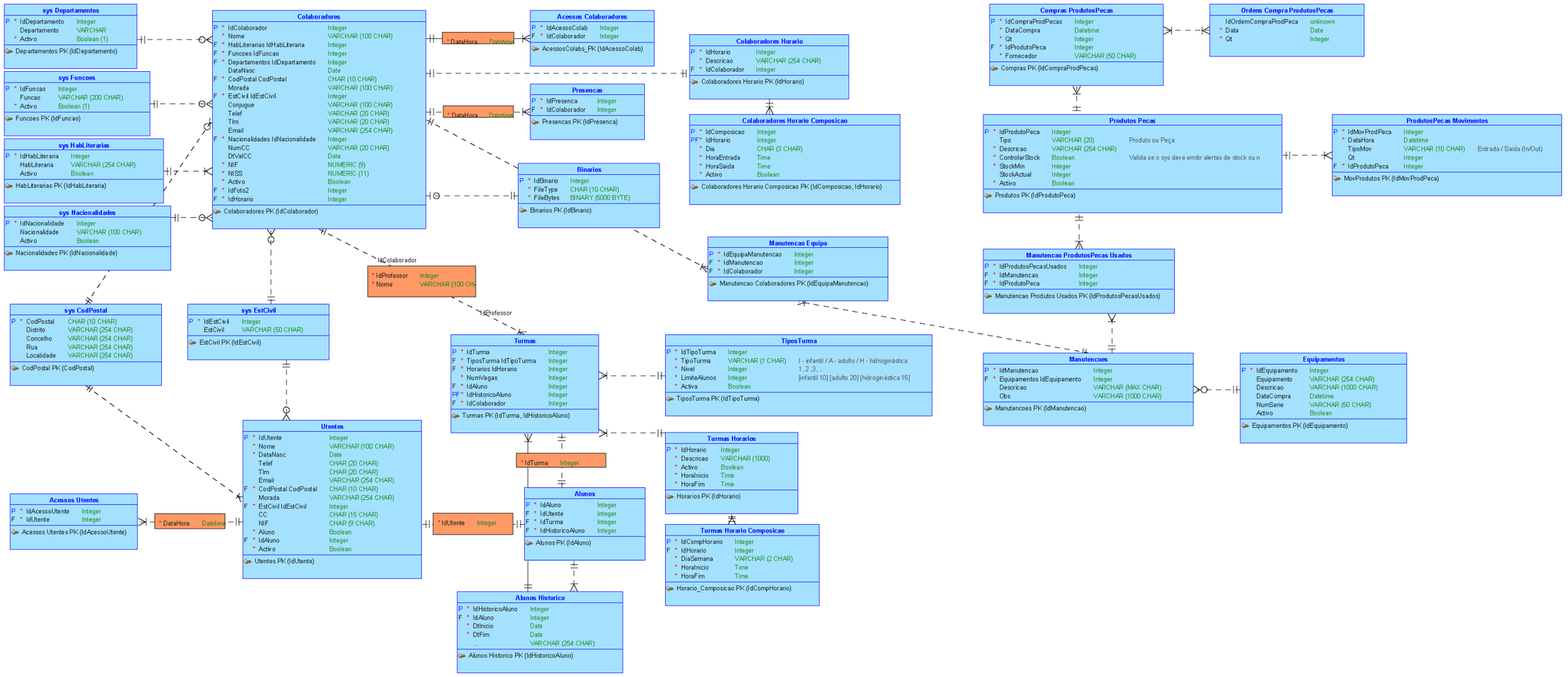
Ata 04


Colaboradores
P * IdColaborador Integer
* Nome VARCHAR (100 CHAR)
F * HabLiterarias IdHabLiteraria Integer
F * Funcoes IdFuncao Integer
F * Departamentos IdDepartamento Integer
DataNasc Date
F * CodPostal CodPostal CHAR (10 CHAR)
Morada VARCHAR (100 CHAR)
F * EstCivil IdEstCivil Integer
Conjugue VARCHAR (100 CHAR)
Telef VARCHAR (20 CHAR)
Tlm VARCHAR (20 CHAR)
Email VARCHAR (254 CHAR)
F * Nacionalidades IdNacionalidade Integer
NumCC VARCHAR (20 CHAR)
DtValCC Date
* NIF NUMERIC (9)
* NISS NUMERIC (11)
* Activo Boolean
F * IdHorario Integer
F * IdFoto2 Integer
Colaboradores PK (IdColaborador)
Colaboradores Horario
P * IdHorario Integer
Descricao unknown
F * IdColaborador Integer
Colaboradores Horario PK (IdHorario)
Colaboradores Horario Composicao
P * IdComposicao Integer
PF * IdHorario Integer
* Dia CHAR (3 CHAR)
* HoraEntrada Time
* HoraSaida Time
* Activo Boolean
Colaboradores Horario Composicao PK (IdComposicao, IdHorario)
* DataHora Datetime
* TipoMov CHAR
* Infraccao Boolean
Presencas
P * IdPresenca Integer
F * IdColaborador Integer
Presencas PK (IdPresenca)
* IdProfessor Integer
* Nome VARCHAR (100 C
Binarios
P * IdBinario Integer
* FileType CHAR (10 CHAR)
* FileBytes BINARY (5000 BYTE)
Binarios PK (IdBinario)
Turmas
P * IdTurma Integer
F * TiposTurma IdTipoTurma Integer
F * Horarios IdHorario Integer
* NumVagas Integer
F * IdAluno Integer
PF * IdHistoricoAluno Integer
F * IdColaborador Integer
Turmas PK (IdTurma, IdHistoricoAluno)
Turmas Horarios
P * IdHorario Integer
* Descricao VARCHAR (1000)
* Activo Boolean
* HoraInicio Time
* HoraFim Time
Horarios PK (IdHorario)
Turmas Horario Composicao
P * IdCompHorario Integer
F * IdHorario Integer
* DiaSemana VARCHAR (2 CHAR)
* HoraInicio Time
* HoraFim Time
Horario_Composicao PK (IdCompHorario)

Ata 05

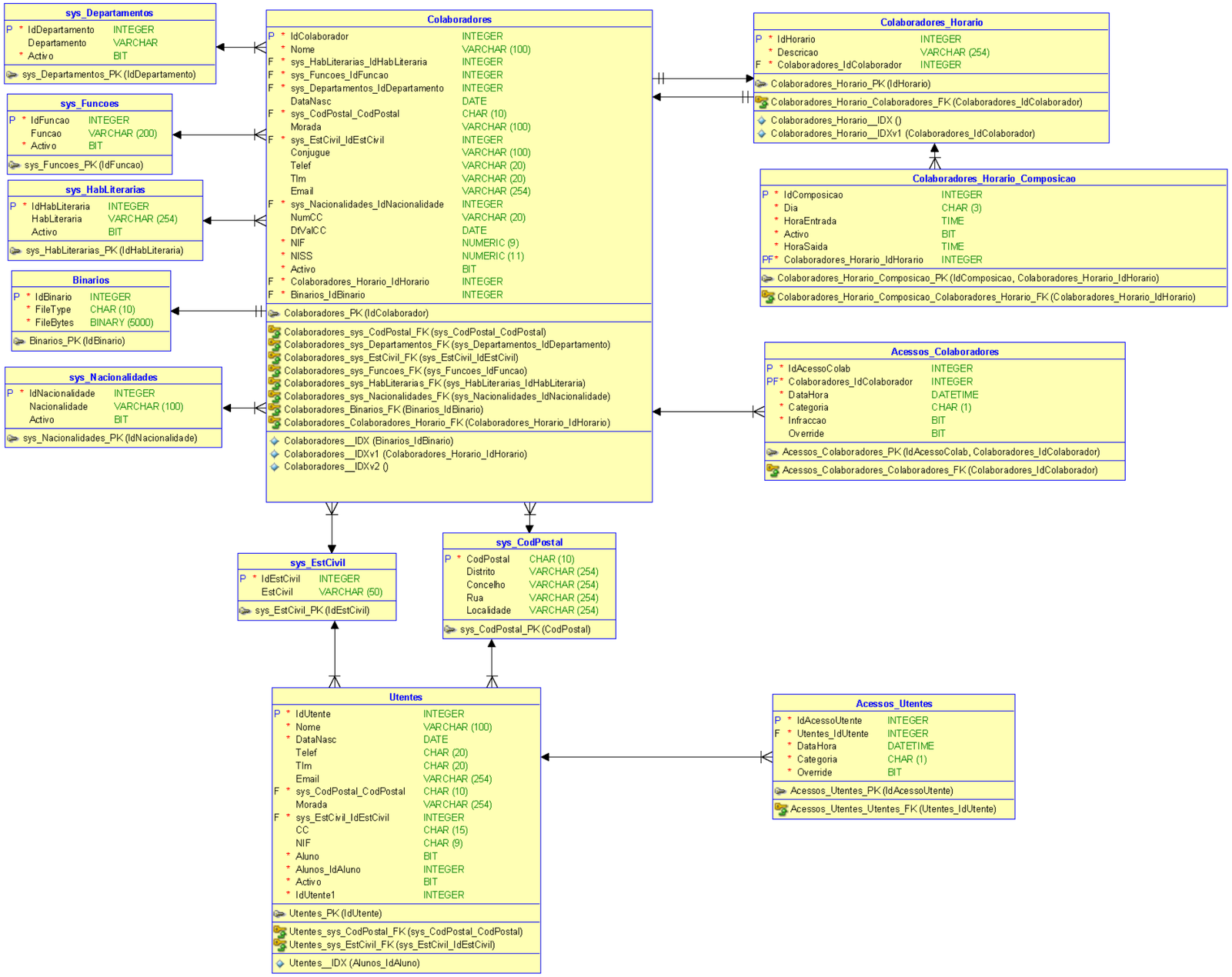

Equipamentos
P * IdEquipamento Integer
Equipamento VARCHAR (254 CHAR)
Descricao VARCHAR (1000 CHAR)
DataCompra Datetime
NumSerie VARCHAR (50 CHAR)
Activo Boolean
Equipamentos PK (IdEquipamento)
Manutencao Equipa
P * IdEquipaManutencao Integer
F * IdManutencao Integer
F * IdColaborador Integer
Manutencao Colaboradores PK (IdEquipaManutencao)
Colaboradores
P * IdColaborador Integer
* Nome VARCHAR (100 CHAR)
F * HabLiterarias IdHabLiteraria Integer
F * Funcoes IdFuncao Integer
F * Departamentos IdDepartamento Integer
DataNasc Date
F * CodPostal CodPostal CHAR (10 CHAR)
Morada VARCHAR (100 CHAR)
F * EstCivil IdEstCivil Integer
Conjugue VARCHAR (100 CHAR)
Telef VARCHAR (20 CHAR)
Tlm VARCHAR (20 CHAR)
Email VARCHAR (254 CHAR)
F * Nacionalidades IdNacionalidade Integer
NumCC VARCHAR (20 CHAR)
DtValCC Date
* NIF NUMERIC (9)
* NISS NUMERIC (11)
* Activo Boolean
F * IdFoto2 Integer
F * IdHorario Integer
Colaboradores PK (IdColaborador)
Manutencoes
P * IdManutencao Integer
F * Equipamentos IdEquipamento Integer
Descricao VARCHAR (MAX CHAR)
Obs VARCHAR (1000 CHAR)
Manutencoes PK (IdManutencao)
Manutencao ProdutosPecas Usados
P * IdProdutosPecasUsados Integer
F * IdManutencao Integer
F * IdProdutoPeca Integer
Manutencao Produtos Usados PK (IdProdutosPecasUsados
Ordens Compra ProdutosPecas
IdOrdemCompraProdPeca unknown
* Data Date
* Qt Integer
Compras ProdutosPecas
P * IdCompraProdPecas Integer
* DataCompra Datetime
* Qt Integer
F * IdProdutoPeca Integer
* Fornecedor VARCHAR (50 CHAR)
Compras PK (IdCompraProdPecas)
Produtos Pecas
P * IdProdutoPeca Integer
Tipo VARCHAR (20) Produto ou Peça
* Descricao VARCHAR (254 CHAR)
* ControlarStock Boolean Valida se o sys deve emitir alertas de stock ou n
* StockMin Integer
StockActual Integer
* Activo Boolean
Produtos PK (IdProdutoPeca)
ProdutosPecas Movimentos
P * IdMovProdPeca Integer
* DataHora Datetime
TipoMov VARCHAR (10 CHAR) Entrada / Saida (In/Out)
Qt Integer
F * IdProdutoPeca Integer
MovProdutos PK (IdMovProdPeca)

Modelo Geral

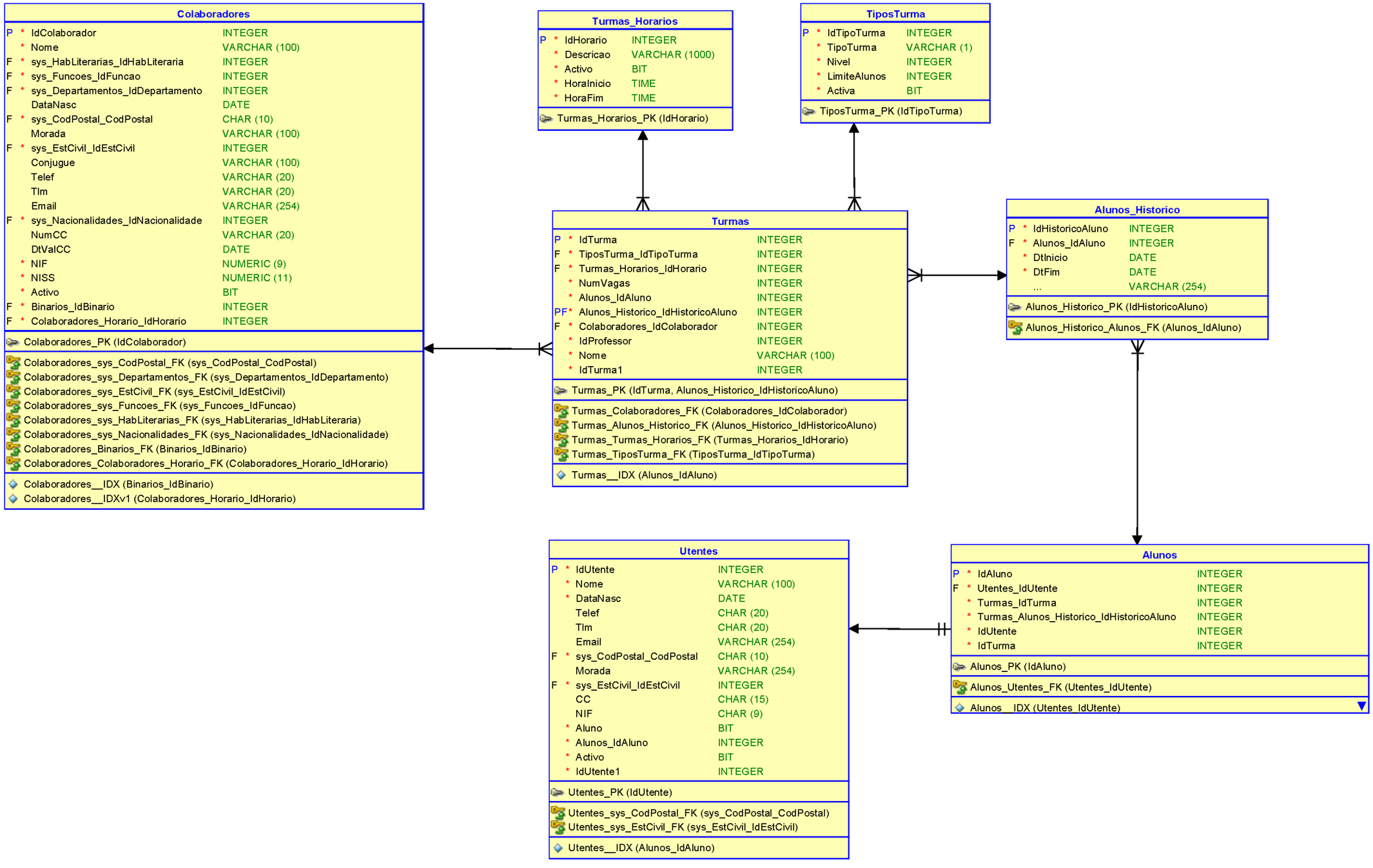


Modelo Relacional

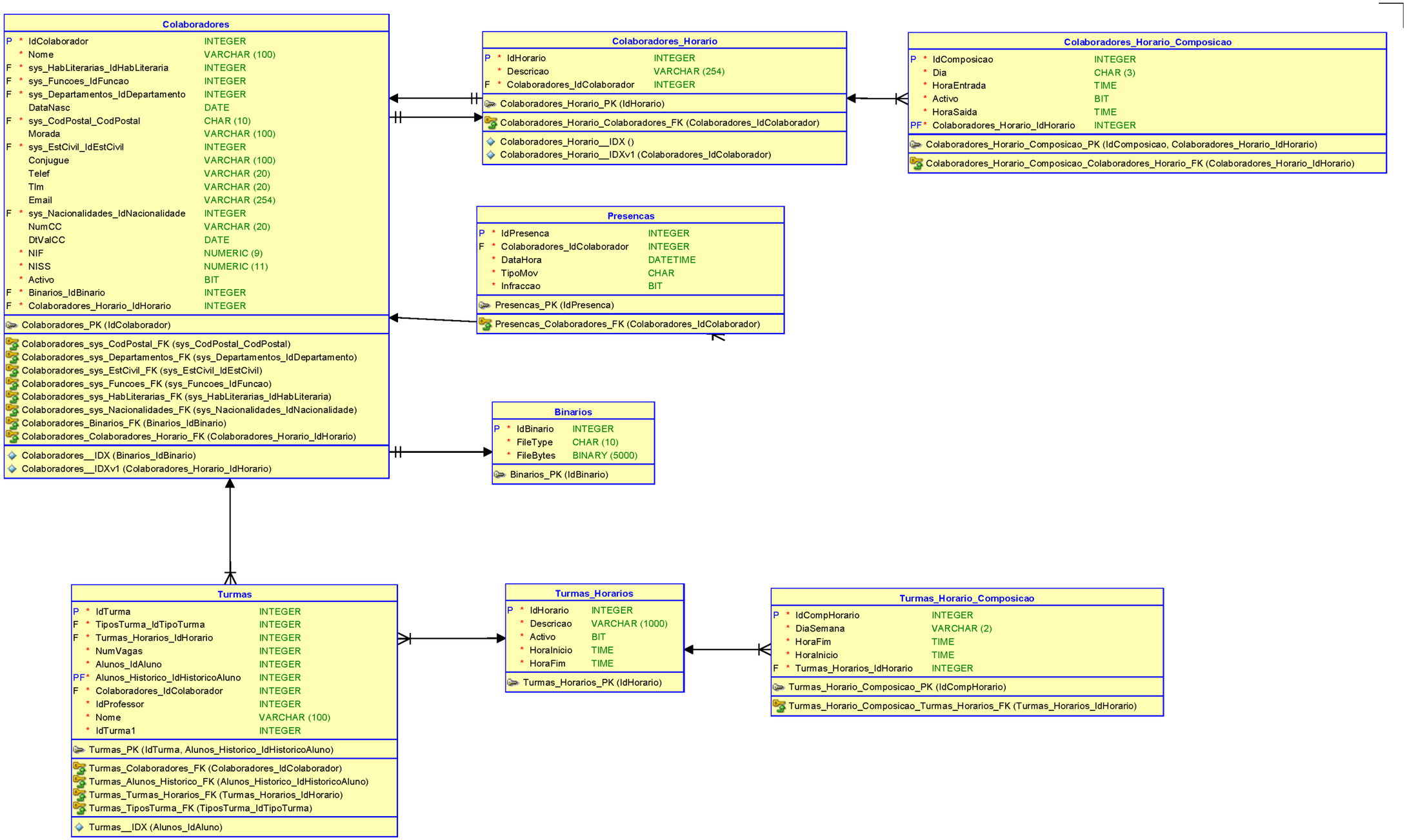
Ata 02



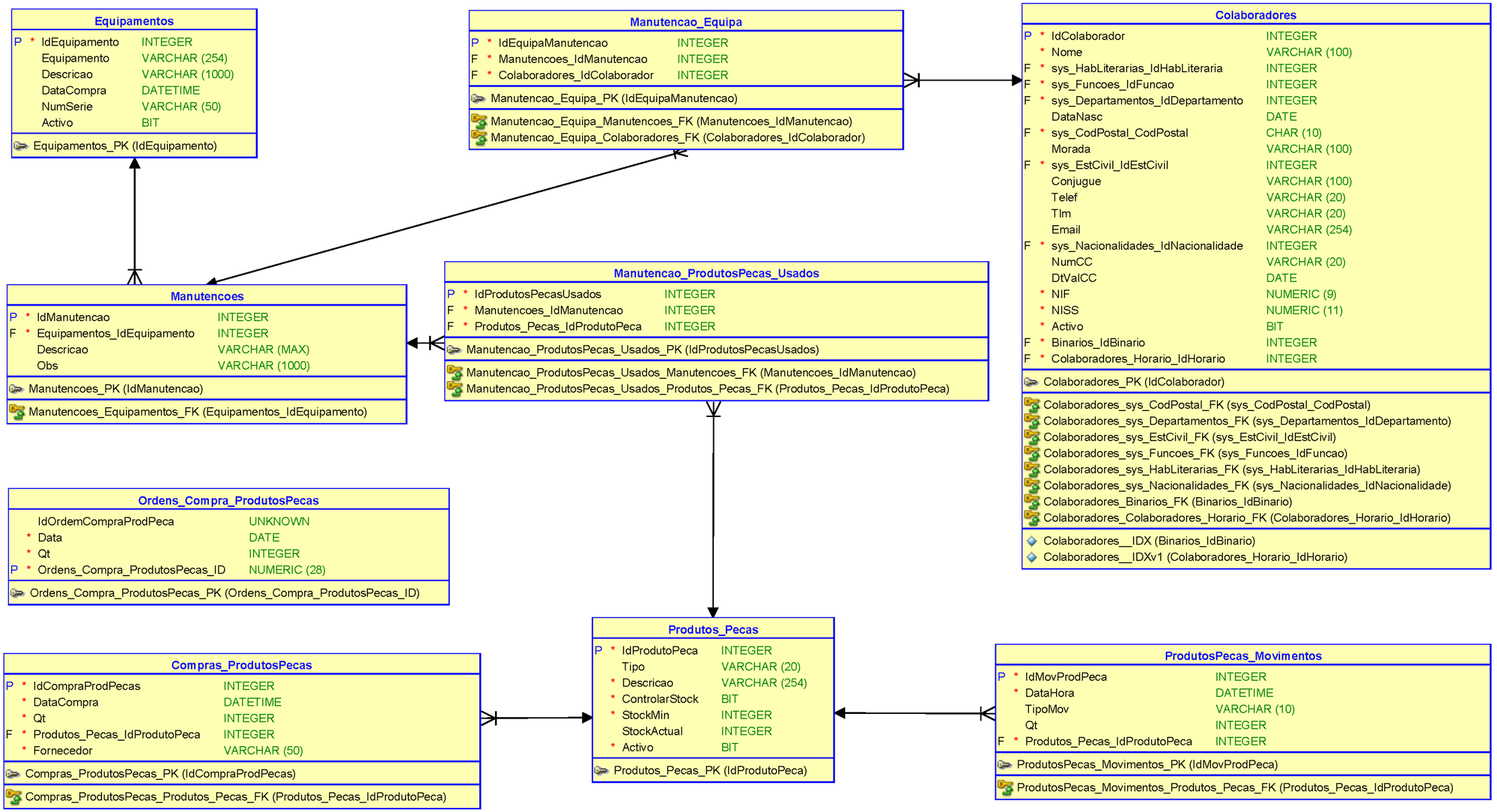
Ata 03


Colaboradores
P * IdColaborador INTEGER
* Nome VARCHAR (100)
F * sys_HabLiterarias_IdHabLiteraria INTEGER
F * sys_Funcoes_IdFuncao INTEGER
F * sys_Departamentos_IdDepartamento INTEGER
DataNasc DATE
F * sys_CodPostal_CodPostal CHAR (10)
Morada VARCHAR (100)
F * sys_EstCivil_IdEstCivil INTEGER
Conjugue VARCHAR (100)
Telef VARCHAR (20)
Tlm VARCHAR (20)
Email VARCHAR (254)
F * sys_Nacionalidades_IdNacionalidade INTEGER
NumCC VARCHAR (20)
DtValCC DATE
* NIF NUMERIC (9)
* NISS NUMERIC (11)
* Activo BIT
F * Binarios_IdBinario INTEGER
F * Colaboradores_Horario_IdHorario INTEGER
Colaboradores_PK (IdColaborador)
Colaboradores_sys_CodPostal_FK (sys_CodPostal_CodPostal)
Colaboradores_sys_Departamentos_FK (sys_Departamentos_IdDepartamento)
Colaboradores_sys_EstCivil_FK (sys_EstCivil_IdEstCivil)
Colaboradores_sys_Funcoes_FK (sys_Funcoes_IdFuncao)
Colaboradores_sys_HabLiterarias_FK (sys_HabLiterarias_IdHabLiteraria)
Colaboradores_sys_Nacionalidades_FK (sys_Nacionalidades_IdNacionalidade)
Colaboradores_Binarios_FK (Binarios_IdBinario)
Colaboradores_Colaboradores_Horario_FK (Colaboradores_Horario_IdHorario)
Colaboradores__IDX (Binarios_IdBinario)
Colaboradores__IDXv1 (Colaboradores_Horario_IdHorario)
Turmas_Horarios
P * IdHorario INTEGER
* Descricao VARCHAR (1000)
* Activo BIT
* HoraInicio TIME
* HoraFim TIME
Turmas_Horarios_PK (IdHorario)
TiposTurma
P * IdTipoTurma INTEGER
* TipoTurma VARCHAR (1)
* Nivel INTEGER
* LimiteAlunos INTEGER
* Activa BIT
TiposTurma_PK (IdTipoTurma)
Turmas
P * IdTurma INTEGER
F * TiposTurma_IdTipoTurma INTEGER
F * Turmas_Horarios_IdHorario INTEGER
* NumVagas INTEGER
* Alunos_IdAluno INTEGER
PF * Alunos_Historico_IdHistoricoAluno INTEGER
F * Colaboradores_IdColaborador INTEGER
* IdProfessor INTEGER
* Nome VARCHAR (100)
* IdTurma1 INTEGER
Turmas_PK (IdTurma, Alunos_Historico_IdHistoricoAluno)
Turmas_Colaboradores_FK (Colaboradores_IdColaborador)
Turmas_Alunos_Historico_FK (Alunos_Historico_IdHistoricoAluno)
Turmas_Turmas_Horarios_FK (Turmas_Horarios_IdHorario)
Turmas_TiposTurma_FK (TiposTurma_IdTipoTurma)
Turmas__IDX (Alunos_IdAluno)
Alunos_Historico
P * IdHistoricoAluno INTEGER
F * Alunos_IdAluno INTEGER
* DtInicio DATE
* DtFim DATE
... VARCHAR (254)
Alunos_Historico_PK (IdHistoricoAluno)
Alunos_Historico_Alunos_FK (Alunos_IdAluno)
Utentes
P * IdUtente INTEGER
* Nome VARCHAR (100)
* DataNasc DATE
Telef CHAR (20)
Tlm CHAR (20)
Email VARCHAR (254)
F * sys_CodPostal_CodPostal CHAR (10)
Morada VARCHAR (254)
F * sys_EstCivil_IdEstCivil INTEGER
CC CHAR (15)
NIF CHAR (9)
* Aluno BIT
* Alunos_IdAluno INTEGER
* Activo BIT
* IdUtente1 INTEGER
Utentes_PK (IdUtente)
Utentes_sys_CodPostal_FK (sys_CodPostal_CodPostal)
Utentes_sys_EstCivil_FK (sys_EstCivil_IdEstCivil)
Utentes__IDX (Alunos_IdAluno)
Alunos
P * IdAluno INTEGER
F * Utentes_IdUtente INTEGER
* Turmas_IdTurma INTEGER
* Turmas_Alunos_Historico_IdHistoricoAluno INTEGER
* IdUtente INTEGER
* IdTurma INTEGER
Alunos_PK (IdAluno)
Alunos_Utentes_FK (Utentes_IdUtente)
Alunos__IDX (Utentes_IdUtente)

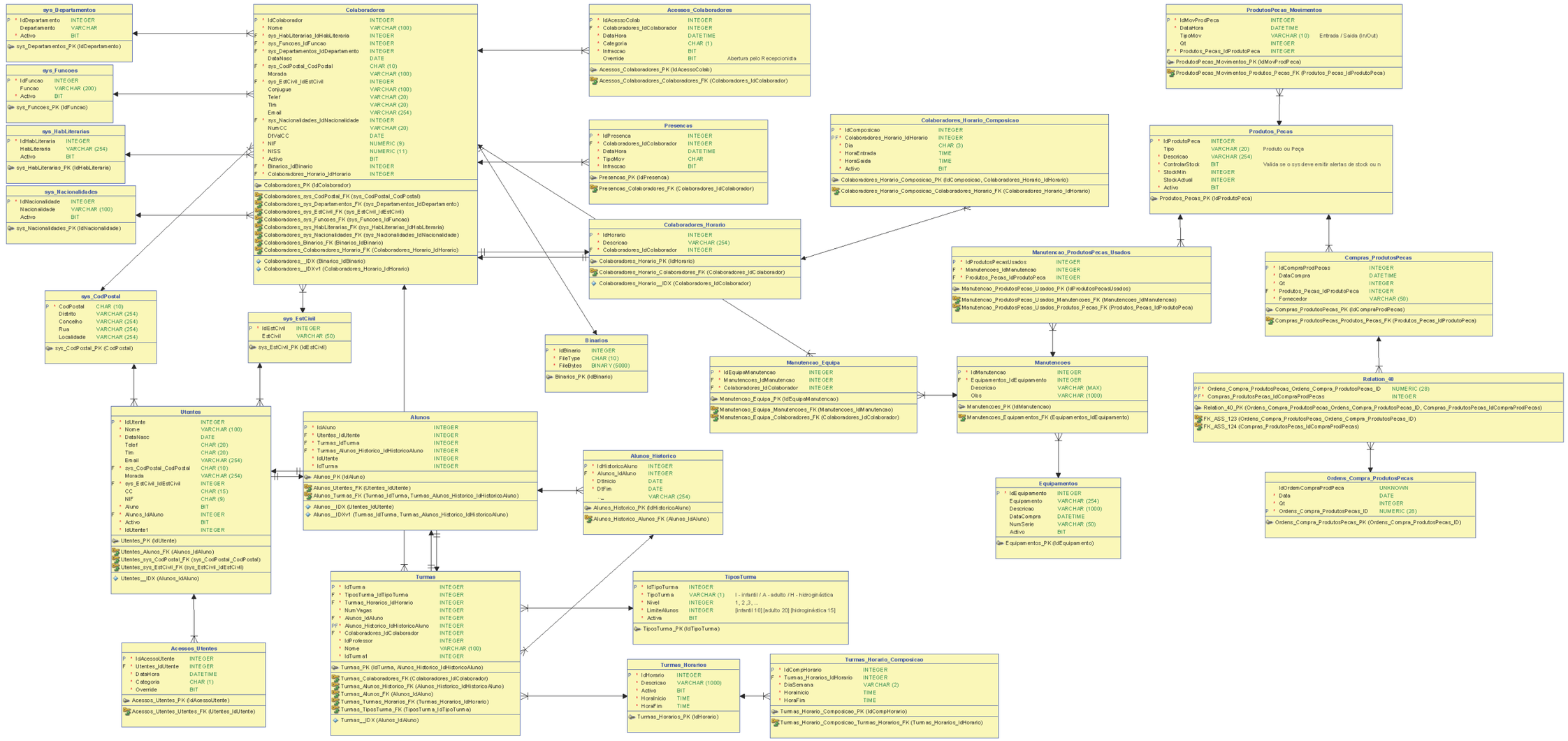
Ata 04


Colaboradores
P * IdColaborador INTEGER
* Nome VARCHAR (100)
F * sys_HabLiterarias_IdHabLiteraria INTEGER
F * sys_Funcoes_IdFuncao INTEGER
F * sys_Departamentos_IdDepartamento INTEGER
DataNasc DATE
F * sys_CodPostal_CodPostal CHAR (10)
Morada VARCHAR (100)
F * sys_EstCivil_IdEstCivil INTEGER
Conjugue VARCHAR (100)
Telef VARCHAR (20)
Tlm VARCHAR (20)
Email VARCHAR (254)
F * sys_Nacionalidades_IdNacionalidade INTEGER
NumCC VARCHAR (20)
DtValCC DATE
* NIF NUMERIC (9)
* NISS NUMERIC (11)
* Activo BIT
F * Binarios_IdBinario INTEGER
F * Colaboradores_Horario_IdHorario INTEGER
Colaboradores_PK (IdColaborador)
Colaboradores_sys_CodPostal_FK (sys_CodPostal_CodPostal)
Colaboradores_sys_Departamentos_FK (sys_Departamentos_IdDepartamento)
Colaboradores_sys_EstCivil_FK (sys_EstCivil_IdEstCivil)
Colaboradores_sys_Funcoes_FK (sys_Funcoes_IdFuncao)
Colaboradores_sys_HabLiterarias_FK (sys_HabLiterarias_IdHabLiteraria)
Colaboradores_sys_Nacionalidades_FK (sys_Nacionalidades_IdNacionalidade)
Colaboradores_Binarios_FK (Binarios_IdBinario)
Colaboradores_Colaboradores_Horario_FK (Colaboradores_Horario_IdHorario)
Colaboradores__IDX (Binarios_IdBinario)
Colaboradores__IDXv1 (Colaboradores_Horario_IdHorario)
Colaboradores_Horario
P * IdHorario INTEGER
* Descricao VARCHAR (254)
F * Colaboradores_IdColaborador INTEGER
Colaboradores_Horario_PK (IdHorario)
Colaboradores_Horario_Colaboradores_FK (Colaboradores_IdColaborador)
Colaboradores_Horario__IDX ()
Colaboradores_Horario__IDXv1 (Colaboradores_IdColaborador)
Colaboradores_Horario_Composicao
P * IdComposicao INTEGER
* Dia CHAR (3)
* HoraEntrada TIME
* Activo BIT
* HoraSaida TIME
PF * Colaboradores_Horario_IdHorario INTEGER
Colaboradores_Horario_Composicao_PK (IdComposicao, Colaboradores_Horario_IdHorario)
Colaboradores_Horario_Composicao_Colaboradores_Horario_FK (Colaboradores_Horario_IdHorario)
Presencas
P * IdPresenca INTEGER
F * Colaboradores_IdColaborador INTEGER
* DataHora DATETIME
* TipoMov CHAR
* Infraccao BIT
Presencas_PK (IdPresenca)
Presencas_Colaboradores_FK (Colaboradores_IdColaborador)
Binarios
P * IdBinario INTEGER
* FileType CHAR (10)
* FileBytes BINARY (5000)
Binarios_PK (IdBinario)
Turmas
P * IdTurma INTEGER
F * TiposTurma_IdTipoTurma INTEGER
F * Turmas_Horarios_IdHorario INTEGER
* NumVagas INTEGER
* Alunos_IdAluno INTEGER
PF * Alunos_Historico_IdHistoricoAluno INTEGER
F * Colaboradores_IdColaborador INTEGER
* IdProfessor INTEGER
* Nome VARCHAR (100)
* IdTurma1 INTEGER
Turmas_PK (IdTurma, Alunos_Historico_IdHistoricoAluno)
Turmas_Colaboradores_FK (Colaboradores_IdColaborador)
Turmas_Alunos_Historico_FK (Alunos_Historico_IdHistoricoAluno)
Turmas_Turmas_Horarios_FK (Turmas_Horarios_IdHorario)
Turmas_TiposTurma_FK (TiposTurma_IdTipoTurma)
Turmas__IDX (Alunos_IdAluno)
Turmas_Horarios
P * IdHorario INTEGER
* Descricao VARCHAR (1000)
* Activo BIT
* HoraInicio TIME
* HoraFim TIME
Turmas_Horarios_PK (IdHorario)
Turmas_Horario_Composicao
P * IdCompHorario INTEGER
* DiaSemana VARCHAR (2)
* HoraFim TIME
* HoraInicio TIME
F * Turmas_Horarios_IdHorario INTEGER
Turmas_Horario_Composicao_PK (IdCompHorario)
Turmas_Horario_Composicao_Turmas_Horarios_FK (Turmas_Horarios_IdHorario)

Ata 05


Equipamentos
P * IdEquipamento INTEGER
Equipamento VARCHAR (254)
Descricao VARCHAR (1000)
DataCompra DATETIME
NumSerie VARCHAR (50)
Activo BIT
Equipamentos_PK (IdEquipamento)
Manutencao_Equipa
P * IdEquipaManutencao INTEGER
F * Manutencoes_IdManutencao INTEGER
F * Colaboradores_IdColaborador INTEGER
Manutencao_Equipa_PK (IdEquipaManutencao)
Manutencao_Equipa_Manutencoes_FK (Manutencoes_IdManutencao)
Manutencao_Equipa_Colaboradores_FK (Colaboradores_IdColaborador)
Colaboradores
P * IdColaborador INTEGER
* Nome VARCHAR (100)
F * sys_HabLiterarias_IdHabLiteraria INTEGER
F * sys_Funcoes_IdFuncao INTEGER
F * sys_Departamentos_IdDepartamento INTEGER
DataNasc DATE
F * sys_CodPostal_CodPostal CHAR (10)
Morada VARCHAR (100)
F * sys_EstCivil_IdEstCivil INTEGER
Conjugue VARCHAR (100)
Telef VARCHAR (20)
Tlm VARCHAR (20)
Email VARCHAR (254)
F * sys_Nacionalidades_IdNacionalidade INTEGER
NumCC VARCHAR (20)
DtValCC DATE
* NIF NUMERIC (9)
* NISS NUMERIC (11)
* Activo BIT
F * Binarios_IdBinario INTEGER
F * Colaboradores_Horario_IdHorario INTEGER
Colaboradores_PK (IdColaborador)
Colaboradores_sys_CodPostal_FK (sys_CodPostal_CodPostal)
Colaboradores_sys_Departamentos_FK (sys_Departamentos_IdDepartamento)
Colaboradores_sys_EstCivil_FK (sys_EstCivil_IdEstCivil)
Colaboradores_sys_Funcoes_FK (sys_Funcoes_IdFuncao)
Colaboradores_sys_HabLiterarias_FK (sys_HabLiterarias_IdHabLiteraria)
Colaboradores_sys_Nacionalidades_FK (sys_Nacionalidades_IdNacionalidade)
Colaboradores_Binarios_FK (Binarios_IdBinario)
Colaboradores_Colaboradores_Horario_FK (Colaboradores_Horario_IdHorario)
Colaboradores__IDX (Binarios_IdBinario)
Colaboradores__IDXv1 (Colaboradores_Horario_IdHorario)
Manutencoes
P * IdManutencao INTEGER
F * Equipamentos_IdEquipamento INTEGER
Descricao VARCHAR (MAX)
Obs VARCHAR (1000)
Manutencoes_PK (IdManutencao)
Manutencoes_Equipamentos_FK (Equipamentos_IdEquipamento)
Manutencao_ProdutosPecas_Usados
P * IdProdutosPecasUsados INTEGER
F * Manutencoes_IdManutencao INTEGER
F * Produtos_Pecas_IdProdutoPeca INTEGER
Manutencao_ProdutosPecas_Usados_PK (IdProdutosPecasUsados)
Manutencao_ProdutosPecas_Usados_Manutencoes_FK (Manutencoes_IdManutencao)
Manutencao_ProdutosPecas_Usados_Produtos_Pecas_FK (Produtos_Pecas_IdProdutoPeca)
Ordens_Compra_ProdutosPecas
IdOrdemCompraProdPeca UNKNOWN
* Data DATE
* Qt INTEGER
P * Ordens_Compra_ProdutosPecas_ID NUMERIC (28)
Ordens_Compra_ProdutosPecas_PK (Ordens_Compra_ProdutosPecas_ID)
Produtos_Pecas
P * IdProdutoPeca INTEGER
Tipo VARCHAR (20)
* Descricao VARCHAR (254)
* ControlarStock BIT
* StockMin INTEGER
StockActual INTEGER
* Activo BIT
Produtos_Pecas_PK (IdProdutoPeca)
Compras_ProdutosPecas
P * IdCompraProdPecas INTEGER
* DataCompra DATETIME
* Qt INTEGER
F * Produtos_Pecas_IdProdutoPeca INTEGER
* Fornecedor VARCHAR (50)
Compras_ProdutosPecas_PK (IdCompraProdPecas)
Compras_ProdutosPecas_Produtos_Pecas_FK (Produtos_Pecas_IdProdutoPeca)
ProdutosPecas_Movimentos
P * IdMovProdPeca INTEGER
* DataHora DATETIME
TipoMov VARCHAR (10)
Qt INTEGER
F * Produtos_Pecas_IdProdutoPeca INTEGER
ProdutosPecas_Movimentos_PK (IdMovProdPeca)
ProdutosPecas_Movimentos_Produtos_Pecas_FK (Produtos_Pecas_IdProdutoPeca)

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
```

Results Messages

	Nº Colab.	Nome	Departamento	Função	Horário	DataHora	Tipo
1	1	Bradley Kaká	Secretaria	Administrativo	Horário 1	2017-05-20 14:05:00.000	E
2	1	Bradley Kaká	Secretaria	Administrativo	Horário 1	2017-05-22 19:05:00.000	S
3	5	Noel Mundinho	Secretaria	Socorrista	Horário 1	2017-05-22 14:05:00.000	E
4	5	Noel Mundinho	Secretaria	Socorrista	Horário 1	2017-05-22 16:05:00.000	S

- **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
```

Results Messages

	Nome	Dt. Nasc.	Telef.	TLM	Email	Tipo Turma	Turma	Professor
1	Rider Géu	1992-03-21	919322979	917977489	Elizabeth.Jó@piscinasbarcacellos.pb	N	Natação 1	Elizabeth Jó
2	Nicolly Dinha	1994-01-14	924231528	993100055	Sejal.Dado@piscinasbarcacellos.pb	H	Hidroginastica 1	Júlia Cecéu
3	Aksinia Quita	1992-07-11	930494500	968265186	Núbia.Guta@piscinasbarcacellos.pb	N	Natação 1	Elizabeth Jó
4	Juliano Huck	1971-01-07	926838192	963439630	Aynoah.Ganso@piscinasbarcacellos.pb	N	Natação 1	Elizabeth Jó

- **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)
```

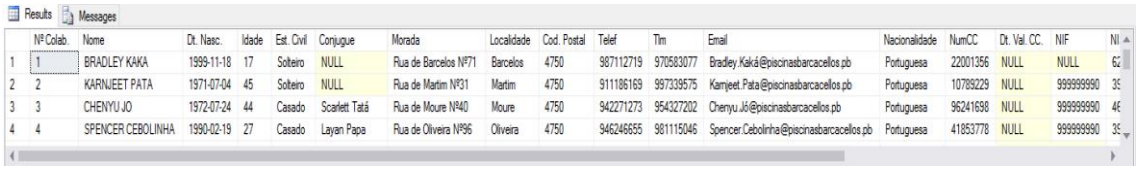
Results Messages

	Nº Colab.	Nome	DataNasc	Faz	Departamento	Funcao
1	15	MAYSON BELA	1998-06-01	19 Anos	Docência	Professor
2	29	MIROSLAVA LALINHA	1973-05-25	45 Anos	Docência	Professor
3	35	RAPHAEL JUCA	1978-05-27	39 Anos	Professores	Professor
4	191	JERONIMO PIPOCA	1974-05-27	43 Anos	Professores	Professor

- **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
```

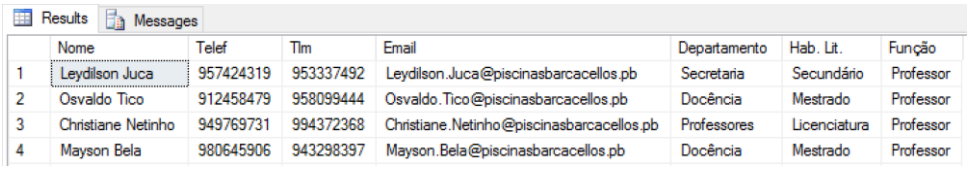
```
GO
```



	Nº Colab.	Nome	Dt. Nasc.	Idade	Est. Civil	Conjugue	Morada	Localidade	Cod. Postal	Telef	Tlm	Email	Nacionalidade	NumCC	Dt. Val. CC.	NIF	NI
1	1	BRADLEY KAKA	1999-11-18	17	Solteiro	NULL	Rua de Barcelos Nº71	Barcelos	4750	987112719	970583077	Bradley.Kaká@piscinasbarcacellos.pb	Portuguesa	22001356	NULL	NULL	62
2	2	KARNJEET PATA	1971-07-04	45	Solteiro	NULL	Rua de Martim Nº31	Martim	4750	911186169	997339575	Karnjeet.Pata@piscinasbarcacellos.pb	Portuguesa	10789229	NULL	999999990	35
3	3	CHENYU JO	1972-07-24	44	Casado	Scarlett Tatá	Rua de Moure Nº40	Moure	4750	942271273	954327202	Chenyu.Jó@piscinasbarcacellos.pb	Portuguesa	96241698	NULL	999999990	4E
4	4	SPENCER CEBOLINHA	1990-02-19	27	Casado	Layan Papa	Rua de Oliveira Nº96	Oliveira	4750	946246655	981115046	Spencer.Cebolinha@piscinasbarcacellos.pb	Portuguesa	41853778	NULL	999999990	35

- **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
```



	Nome	Telef	Tlm	Email	Departamento	Hab. Lit.	Função
1	Leydilson Juca	957424319	953337492	Leydilson.Juca@piscinasbarcacellos.pb	Secretaria	Secundário	Professor
2	Osvaldo Tico	912458479	958099444	Osvaldo.Tico@piscinasbarcacellos.pb	Docência	Mestrado	Professor
3	Christiane Netinho	949769731	994372368	Christiane.Netinho@piscinasbarcacellos.pb	Professores	Licenciatura	Professor
4	Mayson Bela	980645906	943298397	Mayson.Bela@piscinasbarcacellos.pb	Docência	Mestrado	Professor

- **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
```

Results Messages

	Nº Utente	Nome	Dt. Nasc.	Telef	Tlm	Email	Cod. Postal	Morada	Nº CC	NIF	Aluno
1	306	Scarlett Tatá	1972-07-24	942271273	954327202	Chenyu.Jó@piscinasbarcacellos.pb	4750	Rua de Moure Nº40	96241698	999999990	Sim
2	307	Layan Papa	1990-02-19	946246655	981115046	Spencer.Cebolinha@piscinasbarcacellos.pb	4750	Rua de Oliveira Nº96	41853778	999999990	Não
3	308	Alfredo Fafá	1973-11-27	981153364	940429810	Noel.Mundinho@piscinasbarcacellos.pb	4750	Rua de Palme Nº11	36221698	NULL	Sim
4	309	Gilcélio Jacaré	1997-10-22	975263753	963980930	Nilson.JP@piscinasbarcacellos.pb	4750	Rua de Panque Nº23	34756626	999999990	Não

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
```

```
1   SELECT CAST(dbo.CalcularAnos('2000-01-01') AS VARCHAR) + ' Anos' AS 'Idade'
```

	Idade (String)
1	17 Anos

- **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
```

```
1  SELECT dbo.TratarNome('Flávio OLIVEIRA ferreira LÁU', 1, 0, 0) AS 'NOME 1'
2  SELECT dbo.TratarNome('Flávio OLIVEIRA ferreira LÁU', 0, 1, 0) AS 'NOME 2'
3  SELECT dbo.TratarNome('Flávio OLIVEIRA ferreira LÁU', 0, 1, 1) AS 'NOME 3'
4
```

Result 1 - / SQL execution time: 00:00:00.015 / Total time: 00:00:00.101 / (1 row affec

	NOME 1 (String)
1	FLÁVIO OLIVEIRA FERREIRA LÁU

Result 2 - / SQL execution time: 00:00:00.015 / Total time: 00:00:00.106 / (1 row affec

	NOME 2 (String)
1	Flávio Oliveira Ferreira Láu

Result 3 - / SQL execution time: 00:00:00.015 / Total time: 00:00:00.108 / (1 row affec

	NOME 3 (String)
1	Flavio Oliveira Ferreira Lau

- **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
```

```
1  SELECT
2      dbo.ValidarNIF(500779112) AS 'Resultado 1',
3      CASE
4          WHEN dbo.ValidarNIF(500779112) = 1
5          THEN 'NIF Válido'
6          ELSE 'NIF Inválido' END AS 'Resultado 2'
```

	Resultado 1 Boolean	Resultado 2 String
1	1	NIF Válido

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
```