PROJETO DE DISCIPLINA:
FUNDAMENTOS DE BANCO DE DADOS

HAKKINEN DINIZ SANTOS

SERRA TALHADA, PE
NOVEMBRO / 2019

HAKKINEN DINIZ SANTOS

Projeto desenvolvido para efetivação da 2° Etapa, apresentado para avaliação na Disciplina Projeto de Banco de Dados Relacionais ministrada pelo Prof. Hidelberg Oliveira, período letivo 2019.2

PROJETO DE BANCO DE DADOS

01. DEFINIÇÃO DA PROBLEMÁTICA (DOMÍNIO DA APLICAÇÃO)

O objetivo do Sistema deve prover o acompanhamento do aluno em todas as fases de seu crescimento estudantil, desde o acompanhamento de notas, pagamentos e pedagógico.

02. DEFINIÇÃO DE REQUISITOS

2.1. Requisitos Funcionais:

Identificação:	[RF01] Manter Usuário
Descrição:	O sistema provê meios para cadastrar, atualizar, remover e deletar os usuários, com os seguintes dados (Nome, Data de Nascimento, Naturalidade, Endereço, Login, Senha e CPF).
Prioridade:	(X) Essencial ( ) Importante ( ) Desejável

Identificação:	[RF02] Manter Aluno
Descrição:	O sistema provê meios para cadastrar, remover, deletar e atualizar os alunos, com os seguintes dados (Nome, Data de Nascimento, Naturalidade, Endereço, Pai, Mãe, CPF e responsável financeiro.).
Prioridade:	(X) Essencial ( ) Importante ( ) Desejável

Identificação:	[RF03] Manter Professor
Descrição:	O sistema provê meios para cadastrar, atualizar, remover e deletar os professores, com os seguintes dados (Nome, Data de Nascimento, Naturalidade, Endereço).
Prioridade:	(X) Essencial ( ) Importante ( ) Desejável

Identificação:	[RF04] Manter Disciplina
Descrição:	O sistema provê meios para cadastrar, remover, deletar e atualizar as disciplinas, cadastrando seu código, nome, carga horária e professor.
Prioridade:	(X) Essencial ( ) Importante ( ) Desejável

Identificação:	[RF05] Manter Turma
Descrição:	O sistema provê meios para cadastrar, remover, deletar e atualizar as turmas, cadastrando Alunos, Disciplinas, Notas e situação de cada aluno
Prioridade:	(X) Essencial ( ) Importante ( ) Desejável

Identificação:	[RF06] Gerar Senha
Descrição:	Quando algum usuário é cadastrado no sistema, ele recebe

	automaticamente uma senha padrão.
Prioridade:	(X) Essencial ( ) Importante ( ) Desejável

Identificação:	[RF07] Calcular Situação Bimestral Final do Aluno.
Descrição:	O sistema deverá calcular a Média Parcial a partir do cálculo da média aritmética das avaliações registradas pela secretaria e exibir a situação do aluno.
Prioridade:	(X) Essencial ( ) Importante ( ) Desejável

Identificação:	[RF08] Calcular Situação Trimestral Final do Aluno.
Descrição:	O sistema deverá calcular a Média Parcial a partir do cálculo da média aritmética das avaliações registradas pela secretaria e exibir a situação do aluno.
Prioridade:	(X) Essencial ( ) Importante ( ) Desejável

Identificação:	[RF09] Gerar Boletim Escolar
Descrição:	O Sistema provê meios gerar o boletim escolar, contendo os dados dos alunos nas disciplinas, sejam elas por bimestres ou trimestres.
Prioridade:	(X) Essencial ( ) Importante ( ) Desejável

Identificação:	[RF10] Gerar Histórico Escolar
Descrição:	O Sistema provê meios gerar o histórico escolar, contendo os dados dos alunos nas disciplinas, com todos os dados do aluno, durante o período escolar na instituição.
Prioridade:	(X) Essencial ( ) Importante ( ) Desejável

Identificação:	[RF11] Configuração Acadêmica
Descrição:	O sistema deverá possuir parâmetros nos quais possibilitam a configuração acadêmica de avaliação, seja ela em bimestres ou trimestres.
Prioridade:	(X) Essencial ( ) Importante ( ) Desejável

Identificação:	[RF12] Geração de Currículos
Descrição:	O sistema deverá apresentar um currículo, com código, nome e disciplinas alocadas.
Prioridade:	(X) Essencial ( ) Importante ( ) Desejável

Identificação:	[RF13] Manter Responsável Financeiro
Descrição:	O sistema provê meios para cadastrar, remover, deletar e atualizar o responsável financeiro, com CPF como dado obrigatório.
Prioridade:	(X) Essencial ( ) Importante ( ) Desejável

Identificação:	[RF13] Gerar Pagamento

Descrição:	O sistema deve gerar no momento da matrícula do aluno as parcelas referentes ao período letivo do aluno na instituição
Prioridade:	(X) Essencial ( ) Importante ( ) Desejável

Identificação:	[RF14] Liquidar Mensalidade
Descrição:	O sistema deve oferecer ao usuário financeiro a liquidação da mensalidade
Prioridade:	(X) Essencial ( ) Importante ( ) Desejável

Identificação:	[RF14] Emitir relatório
Descrição:	O sistema deve emitir relatórios na área pedagoga.
Prioridade:	(X) Essencial ( ) Importante ( ) Desejável

Identificação:	[RF19] Gerar LOG
Descrição:	O sistema deverá salvar todas as ações dos usuários a cada ação do mesmo.
Prioridade:	(X) Essencial ( ) Importante ( ) Desejável

2.2. Requisitos Não-Funcionais:

Identificação:	[RNF01] Padrão de projeto MVC DAO
Tipo:	*Tipo do RNF*
RF Relacionado:	Todos
Descrição:	O sistema deverá ser desenvolvido sobre o padrão de projeto MVC DAO
Prioridade:	( ) Essencial ( ) Importante (X) Desejável

Identificação:	[RNF02] Linguagem de Programação Java
Tipo:	*Tipo do RNF*
RF Relacionado:	Todos
Descrição:	O backend. Do sistema deverá ser desenvolvido com a linguagem de programação Java, ficando a critério da equipe do projeto, a plataforma para qual será desenvolvido o fron-end (desktop, web, Android).
Prioridade:	(X) Essencial ( ) Importante (X) Desejável

Identificação:	[RNF03] Backup
Tipo:	*Tipo do RNF*
RF Relacionado:	Todos
Descrição:	O sistema deverá ter sistema de backup do banco de dados diariamente no período programado.
Prioridade:	(X) Essencial ( ) Importante (X) Desejável

2.3. Requisitos de Domínio :

Identificação:	[RD01] Tipos de Usuários
Descrição:	O sistema pode ser usado por quatro tipos de pessoas: Administrador, para o administrador geral do sistema; Coordenação Pedagoga, para os pedagogos da instituição; Direção, para os diretores da instituição e; Secretaria, para os secretários da instituição.
Prioridade:	(X) Essencial ( ) Importante ( ) Desejável

Identificação:	[RD02] Usuário Administrador
Descrição:	Só é permitida uma conta de administrador, que tem acesso total e restrito ao banco de dados e ao gerenciamento de todos os módulos da aplicação, sendo o único capaz de realizar cadastro dos funcionários, dentro de seus respectivos grupos.
Prioridade:	(X) Essencial ( ) Importante ( ) Desejável

Identificação:	[RD03] Usuário Coordenador
Descrição:	Um coordenador, por sua vez Pedagogo, poderá apenas registrar dados de acompanhamento do aluno, sem possuir acesso as demais áreas do sistema.
Prioridade:	(X) Essencial ( ) Importante ( ) Desejável

Identificação:	[RD04] Usuário Direção
Descrição:	Um diretor, não poderá realizar nenhuma alteração, apenas visualizar/acompanhar os módulos do sistema.
Prioridade:	(X) Essencial ( ) Importante ( ) Desejável

Identificação:	[RD05] Usuário Secretária
Descrição:	Um secretário, poderá apenas realizar o cadastro das notas médias dos alunos e cadastro das disciplinas sendo restrito a esses módulos.
Prioridade:	(X) Essencial ( ) Importante ( ) Desejável

Identificação:	[RD06] Alterar/Resetar senha de usuário
Descrição:	Um usuário poderá, ele mesmo, alterar sua senha de acesso ao Delfos. Este procedimento poderá ser feito sem restrições. Caso o usuário perca os dados, a Coordenação fará o reset do acesso do usuário utilizando a própria aplicação, com privilégios administrativos.
Prioridade:	(X) Essencial ( ) Importante ( ) Desejável

Identificação:	[RD07] Nota da Avaliação

Descrição:	A nota da avaliação do aluno deve estar no intervalo "0.0 <= avaliação <= 10.0". Cada disciplina deve possuir, no mínimo, 3 notas e no máximo 4, dependendo da configuração da disciplina
Prioridade:	(X) Essencial ( ) Importante ( ) Desejável

Identificação:	[RD08] Calculo da Situação do Aluno Aprovado
Descrição:	O sistema deverá calcular a Média Parcial a partir do cálculo da média aritmética das avaliações registradas pela Secretaria. Aluno que obtiver média >= 7.0, será considerado "Aprovado" e sua Média Final será igual à Média Parcial.
Prioridade:	(X) Essencial ( ) Importante ( ) Desejável

Identificação:	[RD09] Calculo da Situação do Aluno Aprovado na Final
Descrição:	O aluno que obtiver Média Parcial no intervalo "3.0 <= Media Parcial<= 6.9", fará prova final. O cálculo da Média Final deverá ser feito da seguinte forma: (sendo MF a Média Final, MP a nota da Média Parcial e PF a nota da Prova Final). $$\frac{MF = (MP) + (PF)}{2}$$ Alunos que obtiverem Média Final >= 5.0, serão considerados aprovados, com status "Aprovado na Final".
Prioridade:	(X) Essencial ( ) Importante ( ) Desejável

Identificação:	[RD10] Calculo da Situação do Aluno Reprovado
Descrição:	Caso 1: O aluno que obtiver a Média Final < 5.0, será considerado "Reprovado";
Prioridade:	(X) Essencial ( ) Importante ( ) Desejável

Identificação:	[RD11] Inserir Nota da Prova Final
Descrição:	O sistema deverá dispor de local para a secretaria inserir a nota da prova final. Quando a nota for inserida, deverá se calcular a média final do aluno e mostrar o status.
Prioridade:	(X) Essencial ( ) Importante ( ) Desejável

Identificação:	[RD12] Status do Aluno
Descrição:	O sistema deverá mostrar o status do aluno. Caso 1: O status do aluno pode ser exibido por disciplina, onde terá as seguintes siglas: AM, aprovado por média; AP, aprovado; RP, reprovado. Caso 2: O status do aluno poder ser exibido por currículo, onde terá as seguintes nomenclaturas; APROVADO e REPROVADO.
Prioridade:	(X) Essencial ( ) Importante ( ) Desejável

Identificação:	[RD13] Acompanhamento do Aluno
Descrição:	Para cada aluno deverá existir um local para observações do pedagogo. Essas observações deverão ser impressas.
Prioridade:	(X) Essencial ( ) Importante ( ) Desejável

Identificação:	[RD14] Configuração Acadêmica Bimestral
Descrição:	Para cada aluno deverá existir 4 notas por disciplina, realizando o calculo da média geral e da prova final.
Prioridade:	(X) Essencial ( ) Importante ( ) Desejável

Identificação:	[RD15] Configuração Acadêmica Trimestral
Descrição:	Para cada aluno deverá existir 3 notas () por disciplina, realizando o calculo da média geral e da prova final.
Prioridade:	(X) Essencial ( ) Importante ( ) Desejável

Identificação:	[RD16] Configuração de Currículo
Descrição:	Para cada currículo apresentado o sistema deve dispor da configuração do mesmo em bimestre ou trimestre.
Prioridade:	(X) Essencial ( ) Importante ( ) Desejável

Identificação:	[RD17] Configuração de Currículo
Descrição:	Para cada currículo apresentado o sistema deve dispor da configuração do mesmo em bimestre ou trimestre.
Prioridade:	(X) Essencial ( ) Importante ( ) Desejável

Identificação:	[RD18] Configuração de Pagamento
Descrição:	Para cada aluno cadastrado, o sistema deve emitir parcelas referentes ao período do aluno na instituição, possibilitando a configuração dos valores, mas tendo como valores padrões da instituição, onde: EF - Ensino Fundamental nos anos iniciais o valor é R$ 400, 00 EF - Ensino Fundamental nos anos finais o valor é R$ 600, 00 EM – Ensino Médio o valor é R$ 800,00
Prioridade:	(X) Essencial ( ) Importante ( ) Desejável

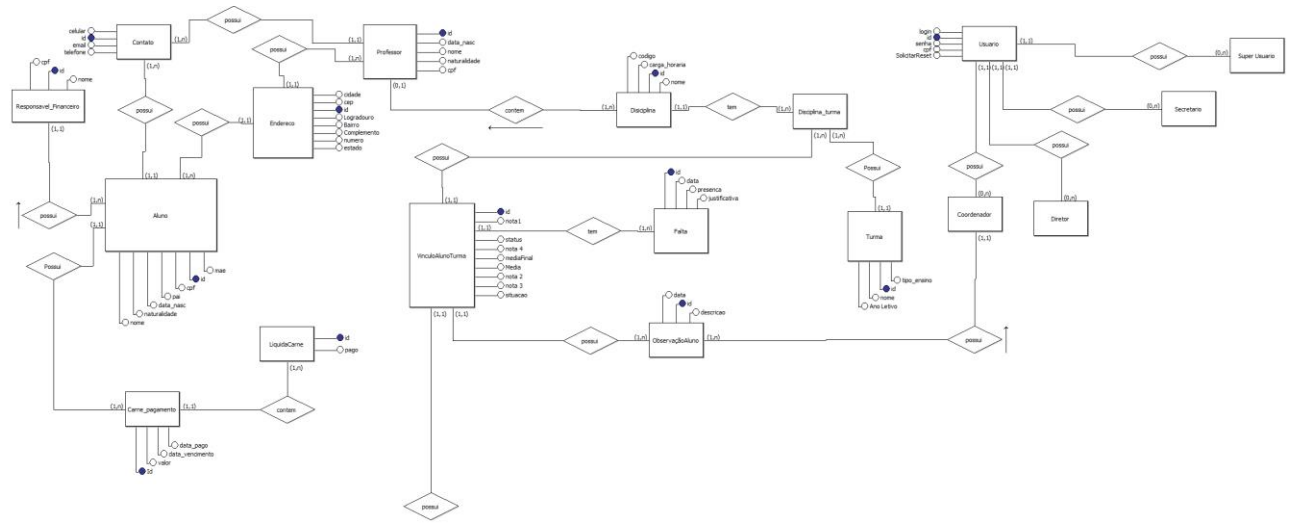
Identificação:	[RD19] Emissão de Relatório por aluno
Descrição:	O sistema deve emitir o relatório de acompanhamentos por aluno, realizados pelos pedagogos da instituição
Prioridade:	(X) Essencial ( ) Importante ( ) Desejável

Identificação:	[RD20] Emissão de Relatório por aluno
Descrição:	O sistema deve emitir o relatório de acompanhamento por aluno, realizados pelos pedagogos da instituição
Prioridade:	(X) Essencial ( ) Importante ( ) Desejável

Identificação:	[RD21] Emissão de Relatório por pedagogo
Descrição:	O sistema deve emitir o relatório de todos os alunos, acompanhados por um usuário, pedagogo, contendo a relação de nomes, e datas de atendimentos dos respectivos alunos, atendido pelo mesmo.
Prioridade:	(X) Essencial ( ) Importante ( ) Desejável

Identificação:	[RD22] Campo senha
Descrição:	O campo senha dos usuários deverá aceitar caracteres alfanuméricos, com tamanho entre 6 e 11, distinguindo caracteres maiúsculos de minúsculos.
Prioridade:	(X) Essencial ( ) Importante ( ) Desejável

03. DIAGRAMA ENTIDADE-RELACIONAMENTO



04. MAPEAMENTO DO ESQUEMA CONCEITUAL PARA O ESQUEMA LÓGICO-RELACIONAL

- ✓ SuperUsuario (**id**, login, senha, resetSenha, cpf);
- ✓ Coordenador Pedagogo (id, login, senha, resetSenha, cpf);
- ✓ Diretor (**id**, login, senha, resetSenha, cpf);
- ✓ Secretario (**id**, login, senha, resetSenha, cpf);
- ✓ Contato (**id**, celular, e-mail, telefone);
- ✓ Endereço (**id**, cidade, cep, logradouro, bairro, complemento, numero, estado);
- ✓ Responsável Financeiro (**id**, nome, cpf);

- ✓ Aluno (**id**, mae, cpf, pai, data_nasc, naturalidade, nome, *responsável_id*, *contato_id*, *endereço_id*);
- ✓ Professor (**id**, nome, cpf, naturalidade, data_nasc, *contato_id*, *endereço_id*);
- ✓ Disciplina (**id**, nome, código, carga horaria, *professor_id*):
- ✓ Turma (**id**, nome, anoLetivo);
- ✓ VinculoAlunoTurma (**id**, nota1, nota2, nota3, nota4, media, mediaFinal, situação, *disciplinaTurma_id*, *aluno_id*):
- ✓ DisciplinaTurma (**id**, *turma_id*, *disciplina_id*);
- ✓ Falta (**id**, data, presença, justificativa, *vinculoAluno_id*);
- ✓ ObservacaoAluno (**id**, data, *vinculoAluno_id*, descrição, *coordenador_id*);
- ✓ Carne_Pagamento (**id**, valor, dataPagamento, dataVencimento, *aluno_id*);
- ✓ LiquidaCarne (**id**, pago, *carne_id*);

05. DICIONÁRIO DE DADOS

Coordenador (Armazena o Coordenador do sistema)			
Atributo	Descrição	Tipo de Dado	Restrições
Id		INT	Chave Primária
Login	Login	VARCHAR(50)	Não nulo
Senha	Senha	VARCHAR(11)	Não nulo
Nome	Nome do Usuario	VARCHAR(100)	Não nulo
ResetSenha	Solicitação de reset senha	VARCHAR	
CPF	Número do CPF	VARCHAR(11)	Não nulo, único

Super Usurio (Armazena o Super Usuario geral do sistema)			
Atributo	Descrição	Tipo de Dado	Restrições
Id		INT	Chave Primária
Login	Login	VARCHAR(50)	Não nulo
Senha	Senha	VARCHAR(11)	Não nulo
ResetSenha	Solicitação de reset senha	VARCHAR	

Nome	Nome do Usuario	VARCHAR(100)	Não nulo
CPF	Número do CPF	VARCHAR(11)	Não nulo, único

Diretor (Armazena o Diretor do sistema)			
Atributo	Descrição	Tipo de Dado	Restrições
Id		INT	Chave Primária
Login	Login	VARCHAR(50)	Não nulo
Senha	Senha	VARCHAR(11)	Não nulo
Nome	Nome do Usuario	VARCHAR(100)	Não nulo
ResetSenha	Solicitação de reset senha	VARCHAR	
CPF	Número do CPF	VARCHAR(11)	Não nulo, único

Secretario (Armazena o secretario do sistema)			
Atributo	Descrição	Tipo de Dado	Restrições
Id		INT	Chave Primária
Login	Login	VARCHAR(50)	Não nulo
Senha	Senha	VARCHAR(11)	Não nulo
Nome	Nome do Usuário	VARCHAR(100)	Não nulo
ResetSenha	Solicitação de reset senha	VARCHAR	
CPF	Número do CPF	VARCHAR(11)	Não nulo, único

Aluno (Armazena todos os alunos e suas respectivas informações)			
Atributo	Descrição	Tipo de Dado	Restrições
Id		INT	Chave Primária
Naturalidade	Local do Aluno Nascimento	VARCHAR(20)	Não nulo
Nome	Nome do Aluno	VARCHAR(100)	Não nulo
CPF	O numero do CPF	VARCHAR(11)	Não nulo e sem mascara
Mãe	Mãe do aluno	VARCHAR(20)	Não nulo
Pai	Pai do aluno	VARCHAR(20)	Não nulo
nascimento	Data_nascimento	VARCHAR(255)	Não nulo
Contato_id	Contato vinculado a aluno	INT	Chave Estrangeira Referencia a contato (id)

Endereço_id	Endereco vinculado a aluno	INT	Chave Estrangeira Referencia a endereço (id)
Responsável Financeiro_id	Responsavel vinculado a aluno	INT	Chave Estrangeira Referencia a financeiro (id)

Professor (Armazena todos os professores e suas respectivas informações)			
Atributo	Descrição	Tipo de Dado	Restrições
Id		INT	Chave Primária
Naturalidade	Local do Professor Nascimento	VARCHAR(20)	Não nulo
Nome	Nome do Professor	VARCHAR(100)	Não nulo
CPF	O numero do CPF	VARCHAR(11)	Não nulo e sem mascara
nascimento	Data_nascimento	VARCHAR(255)	Não nulo
Contato_id	Contato vinculado a professor	INT	Chave Estrangeira Referencia a contato (id)
Endereço_id	Endereco vinculado a professor	INT	Chave Estrangeira Referencia a endereço (id)
Responsável Financeiro_id	Responsavel vinculado a professor	INT	Chave Estrangeira Referencia a financeiro (id)

Endereço (Armazena os endereços e suas respectivas informações)			
Atributo	Descrição	Tipo de Dado	Restrições
Id		INT	Chave Primária
logradouro	Local onde mora	VARCHAR(255)	Não nulo
Bairro	Bairro do local	VARCHAR(255)	Não nulo
Numero	Numero da residência	VARCHAR(255)	Não nulo
Cidade	Cidade onde reside	VARCHAR(255)	Não nulo
Cep	Cep do local	VARCHAR(255)	Não nulo
Uf	Uf do estado onde reside	VARCHAR(255)	Não nulo

Contato (Armazena todos os contatos e suas respectivas informações)

Atributo	Descrição	Tipo de Dado	Restrições
Id		INT	Chave Primária
Telefone	Telefone do contato	VARCHAR(20)	Não nulo
Celular	Celular do contato	VARCHAR(20)	Não nulo
e-mail	e-mail do contato	VARCHAR(50)	Não nulo

Responsável Financeiro (Armazena todos os contatos e suas respectivas informações)			
Atributo	Descrição	Tipo de Dado	Restrições
Id		INT	Chave Primária
Nome	Nome do responsável	VARCHAR(50)	Não nulo
Cpf	CPF do responsável	VARCHAR(11)	Não nulo, único

Turma (Armazena todos os Alunos, disciplinas e suas respectivas informações referente a turma)			
Atributo	Descrição	Tipo de Dado	Restrições
Id		INT	Chave Primária
Nome	Nome da turma	VARCHAR(50)	Não nulo
anoLetivo	Ano letivo da Turma	VARCHAR(20)	Chave Estrangeira Referencia a disciplina (id)

Disciplina (Armazena todos os Professores e suas respectivas informações referente a disciplina)			
Atributo	Descrição	Tipo de Dado	Restrições
Id		INT	Chave Primária
Nome	Nome da turma	VARCHAR(20)	Não nulo
Professor_id	Professor vinculada a disciplina	INT	Chave Estrangeira Referencia a professor (id)
código	Código da disciplina	VARCHAR(20)	Não nulo, único

VinculoAlunoTurma (Armazena todos as respectivas informações referente a aluno)			
Atributo	Descrição	Tipo de Dado	Restrições
Id		INT	Chave Primária
Nota 01	1° nota medial do aluno	VARCHAR(25)	
Nota 02	2° nota medial do aluno	VARCHAR(25)	
Nota 03	3° nota medial do aluno	VARCHAR(25)	
Nota 04	4° nota medial do aluno	VARCHAR(25)	

disciplinaTurma_id	DisciplinaTurma vinculada ao VinculoAlunoTurma	INT	Chave Estrangeira Referencia a DisciplinaTurma (id)
aluno_id	Aluno vinculado ao VinculoAlunoTurma	INT	Chave Estrangeira Referencia a aluno (id)
situação	Situação do aluno	VARCHAR(255)	
média	Média do aluno	VARCHAR(25)	
mediaFinal	Média Final do aluno	VARCHAR(25)	
Status	Status do aluno	BOLEEAN	

Observação Aluno **(Armazena as e suas respectivas informações, de observação)**			
Atributo	Descrição	Tipo de Dado	Restrições
Id		INT	Chave Primária
Data	Data realizada a observação	VARCHAR(20)	Não nulo
Descricao	Descrição das observações	VARCHAR(255)	Não nulo
Vinculo_id	Observação vinculada ao vinculo	INT	Chave estrangeira Referência ao vinculo_Aluno

Carne_Pagamento **(Armazena todos os carne suas respectivas informações)**			
Atributo	Descrição	Tipo de Dado	Restrições
Id		INT	Chave Primária
Valor	Valor do carne	VARCHAR(20)	Não nulo
Data_vencimento	Data de vencimento do carne	DATE	
Data_pagamento	Data de pagamento do carne	DATE	
aluno_id	Aluno vinculado ao Carne	INT	Chave estrangeira Referencia ao Aluno

LiquidaCarne **(Armazena todos os carne liquidados e suas respectivas informações)**			
Atributo	Descrição	Tipo de Dado	Restrições
Id		INT	Chave Primária
Pago	Se foi pago ou não	BOLEEAN	Não nulo
Carne_id	Carne vinculado ao liquida_carne	INT	Chave estrangeira Referencia ao Carne

DisciplinaTurma (Armazena todos os carne liquidados e suas respectivas informações)			
Atributo	Descrição	Tipo de Dado	Restrições
Id		INT	Chave Primária
Turma_id	Turma vinculada a DisciplinaTurma	INT	Chave estrangeira Referente a Turma (id)
Disciplina_id	Disciplina vinculada a DisciplinaTurma	INT	Chave estrangeira Referente a Disciplina (id)

05. SCRIPTS SQL

5.1 DDL

Aluno

(O campo matricula: armezena a matricula do aluno
O campo data: armazena a data de nascimento do aluno
O campo mae: armazena o nome da mae do aluno
O campo naturalizade: armazena a naturaludade do aluno
O campo nome: armazeno o nome do aluno
O campo pai: armazena o nome do pai do aluno
O campo contato: armazena o numero identificador do contato de um aluno
O campo endereço: armazena o numero identificador do endereço de um aluno
O campo responsável_financeiro: armazena o numero identificador de um responsável_financeiro de um aluno**)**

```
CREATE TABLE public.aluno (
    id integer NOT NULL,
    status boolean,
    matricula character varying(50),
    data_nascimento bytea,
    mae character varying(50) NOT NULL,
    naturalidade character varying(50) NOT NULL,
    nome character varying(150) NOT NULL,
    pai character varying(50) NOT NULL,
    contato_id integer,
    endereco_id integer,
    responsavel_financeiro_id integer
);
ALTER TABLE public.aluno
OWNER TO postgres;
```

Carne_Pagemento

(O campo data_vencimento: Armazena a data de vencimento do carne
O campo data_pago: Armazena a data de pagamento do carne
O campo valor: Armazena o valor do carne
O campo aluno_id: Armazena o numero identificador do aluno vinculado ao carne**)**

```
CREATE TABLE public.carne_pagamento (
    id integer NOT NULL,
```

```
   status boolean,
   data_pago bytea NOT NULL,
data_vencimento bytea NOT NULL,
   valor double precision,
   aluno _id integer
);
ALTER TABLE public.carne_pagamento
OWNER TO postgres;
```

Contato

(O campo celular: Armazena o numero do celular com DDD
O campo email: Armazena o email
O campo telefone: Armazena o numero do telefone com DDD**)**

```
CREATE TABLE public.contato (
   id integer NOT NULL,
   status boolean,
   celular character varying(20) NOT NULL,
   email character varying(50) NOT NULL,
   telefone character varying(20) NOT NULL
);
ALTER TABLE public.contato
OWNER TO postgres;
```

Coordenador_Pedagogo

(O campo cpf: Armazena o cpf do usuario
O campo login: Armazena o login do usuario
O campo senha: Armazena a senha do usuario
O campo nome; Armazena o nome do usuário
O campo resetSenha; Armazena a solicitação resetar senha
O campo tipoCargo: Armazena o tipo de acesso do usuario**)**

```
CREATE TABLE public.cord_pedagogo (
   id integer NOT NULL,
   status boolean,
   cpf character varying(15) NOT NULL,
   login character varying(50) NOT NULL,
   nome character varying(100) NOT NULL,
   senha character varying(15) NOT NULL,
   solicitar_reset character varying( (255)
   tipocargo character varying(255)
);

ALTER TABLE public.cord_pedagogo
OWNER TO postgres;
```

Diretor

(O campo cpf: Armazena o cpf do usuario
O campo login: Armazena o login do usuario

O campo senha: Armazena a senha do usuario
O campo nome; Armazena o nome do usuário
O campo resetSenha; Armazena a solicitação resetar senha
O campo tipoCargo: Armazena o tipo de acesso do usuario**)**

```
CREATE TABLE public.diretor (
    id integer NOT NULL,
    status boolean,
    cpf character varying(15) NOT NULL,
    login character varying(50) NOT NULL,
    nome character varying(100) NOT NULL,
    senha character varying(15) NOT NULL,
solicitar_reset character varying( (255)
    tipocargo character varying(255)
);
ALTER TABLE public.diretor
OWNER TO postgres;
```

Disciplina

(O campo carga horaria: Armazena a carga horaria referente a disciplina
O campo codigo: Armazena o codigo referente a disciplina
O campo nome: Armazena o nome referente a disciplina
O campo professor; Armazena o numero identificador do professor referente a esta disciplina**)**

```
CREATE TABLE public.disciplina (
    id integer NOT NULL,
    status boolean,
    carga_horaria character varying(5) NOT NULL,
    codigo character varying(255) NOT NULL,
    nome character varying(20) NOT NULL,
    statusaluno character varying(255),
    professor_id integer
);
ALTER TABLE public.disciplina
OWNER TO postgres;
```

Disciplia_Turma

(O campo disciplina: Armazena o numero identificador da disciplina referente a esta disciplina_turma
O campo turma: Armazena o numero identificador da turma referente a esta disciplina_turma**)**

```
CREATE TABLE public.disciplina_turma (
    id integer NOT NULL,
    status boolean,
    disciplina_id integer,
    turma_id integer
); ALTER TABLE public.disciplina_turma
OWNER TO postgres;
```

Endereco

(O campo bairro: Armazena o bairro
O campo cep: Armazena o cep
O campo cidade: Armazena a cidade
O campo complemento; Armazena o complemento
O campo logradouro: Armazena o logradouro
O campo numero: Armazena o numero da casa
O campo tipo de estado: Armazena o estado**)**

```
CREATE TABLE public.endereco (
    id integer NOT NULL,
    status boolean,
    bairro character varying(120) NOT NULL,
    cep character varying(9) NOT NULL,
    cidade character varying(50) NOT NULL,
    complemento character varying(50) NOT NULL,
    logradouro character varying(120) NOT NULL,
    numero character varying(20) NOT NULL,
    tipoestadouf character varying(255)
);
ALTER TABLE public.endereco
OWNER TO postgres;
```

Falta

(O campo data: Armazena a data da falta
O campo justificativa: Armazena a justificativa da falta
O campo presenca: Armazena a presença
O campo vinculo_aluno_turma; Armazena o numero identificador do vinculo_aluno_turma**)**

```
CREATE TABLE public.falta (
    id integer NOT NULL,
    status boolean,
    data bytea,
    justificativa character varying(255),
    preseca boolean,
    vinculoalunoturma_id integer
);
ALTER TABLE public.falta
OWNER TO postgres;
```

Liquida_Carne

(O campo pago: Armazena se foi pagou ou não
O campo carne_pagamento: Armazena o numero identificador referente ao carne**)**

```
CREATE TABLE public.liquida_carne (
    id integer NOT NULL,
    status boolean,
    pago boolean,
    carne_pagamento_id integer
);
```

```
ALTER TABLE public.liquida_carne
OWNER TO postgres;
```

Observação_Aluno

(O campo data: Armazena a data da observação
O campo descrição: Armazena a descrição do aluno
O campo coordenador: Armazena o numero identificador do aluno referente a esta observação
O campo vinculoalunoturma: Armazena o numero identificador do vinculo_aluno_turma referente a esta observação**)**

```
CREATE TABLE public.observacao_aluno (
    id integer NOT NULL,
    status boolean,
    data bytea,
    descricao character varying(255),
    coordenador_id integer,
    vinculoalunoturma_id integer
);
ALTER TABLE public.observacao_aluno
 OWNER TO postgres;
```

Professor

(O campo cpf: armezena o cpf do professor
O campo data: armazena a data de nascimento do professor
O campo naturalizade: armazena a naturaludade do professor
O campo nome: armazeno o nome do professor
O campo contato: armazena o numero identificador do contato de um professor
O campo endereço: armazena o numero identificador do endereço de um professor**)**

```
CREATE TABLE public.professor (
    id integer NOT NULL,
    status boolean,
    cpf character varying(15) NOT NULL,
    data_nascimento bytea NOT NULL,
    naturalidade character varying(255) NOT NULL,
    nome character varying(255) NOT NULL,
    contato_id integer,
    endereco_id integer
);
ALTER TABLE public.professor
OWNER TO postgres;
```

Responsavel_Financeiro

(O campo cpf: Armazena o cpf do responsável
O campo nome: Armazena o nome do responsável**)**

```
CREATE TABLE public.resp_financeiro (
    id integer NOT NULL,
    status boolean,
```

```
    cpf character varying(15) NOT NULL,
    nome character varying(100) NOT NULL
);
ALTER TABLE public.resp_financeiro
OWNER TO postgres;
```

Secretaria

(O campo cpf: Armazena o cpf do usuario
O campo login: Armazena o login do usuario
O campo senha: Armazena a senha do usuario
O campo nome; Armazena o nome do usuário
O campo resetSenha; Armazena a solicitação resetar senha
O campo tipoCargo: Armazena o tipo de acesso do usuario**)**

```
CREATE TABLE public.secretaria (
    id integer NOT NULL,
    status boolean,
    cpf character varying(15) NOT NULL,
    login character varying(50) NOT NULL,
    nome character varying(100) NOT NULL,
    senha character varying(15) NOT NULL,
solicitar_reset character varying( (255)
    tipocargo character varying(255)
);
ALTER TABLE public.secretaria
OWNER TO postgres;
```

Super Usuario

(O campo cpf: Armazena o cpf do usuario
O campo login: Armazena o login do usuario
O campo senha: Armazena a senha do usuario
O campo nome; Armazena o nome do usuário
O campo resetSenha; Armazena a solicitação resetar senha
O campo tipoCargo: Armazena o tipo de acesso do usuario**)**

```
CREATE TABLE public.super_usuario (
    id integer NOT NULL,
    status boolean,
    cpf character varying(15) NOT NULL,
    login character varying(50) NOT NULL,
    nome character varying(100) NOT NULL,
    senha character varying(15) NOT NULL,
solicitar_reset character varying( (255)
    tipocargo character varying(255)
);
ALTER TABLE public.super_usuario
OWNER TO postgres;
```

Turma

(O campo nome: Armazena o nome da turma
O campo anoLetivo: Armazena o anoLetivo da turma**)**

```
CREATE TABLE public.turma (
    id integer NOT NULL,
    status boolean,
    anoletivo character varying(20),
    nome character varying(20) NOT NULL
);
ALTER TABLE public.turma
OWNER TO postgres;
```

Vinculo_Aluno

(O campo nota1: armazena a nota media 1° da prova de aprendizagem do aluno;
O campo nota2: armazena a nota media 2° da prova de aprendizagem do aluno;
O campo nota3: armazena a nota media 3° da prova de aprendizagem do aluno;
O campo nota4: armazena a nota media 4° da prova de aprendizagem do aluno;

O campo media: armazena a média aritmética das notas;
O campo mediafinal: armazena a média final resultante do cálculo da média;
O campo situaçãoaluno: armazena a situação do aluno referente as suas notas, se ele foi aprovado, aprovado por média, reprovado;
O campo aluno: armazena o número identificador do aluno referente a disciplina que o aluno foi matriculado;
O campo disciplinaTurma: armazena o número identificador da disciplinaTurma do aluno com o curso**)**

```
CREATE TABLE public.vinculo_aluno (
    id integer NOT NULL,
    status boolean,
    media double precision,
    mediafinal double precision,
    nota1 double precision,
    nota2 double precision,
    nota3 double precision,
    nota4 double precision,
    situacaoaluno character varying(255),
    aluno_id integer,
    disciplinaturma_id integer
);
ALTER TABLE public.vinculo_aluno
 OWNER TO postgres;
```

Aluno_contato_view

```
CREATE VIEW public.aluno_contato_view AS
 SELECT a.nome,
    c.celular,
    c.email,
    c.telefone
   FROM (public.aluno a
```

```
    JOIN public.contato c ON ((c.id = a.contato_id)));
```

Aluno_resp_view

```
CREATE VIEW public.aluno_resp_view AS
 SELECT a.nome,
    r.nome AS nome_responsavel
   FROM (public.aluno a
     JOIN public.resp_financeiro r ON ((a.responsavel_financeiro_id = r.id)));
```

Observacao_view

```
CREATE VIEW public.observacao_view AS
 SELECT o.descricao,
    a.nome,
    c.nome AS n_coordenador
   FROM ((public.observacao_aluno o
     JOIN public.aluno a ON ((a.id = o.aluno_id)))
     JOIN public.cord_pedagogo c ON ((c.id = o.coordenador_id)));
```

Calcular_dias

```
CREATE OR REPLACE FUNCTION calcular_dias(data_entrada date, data_saida date)
                        RETURNS integer AS $$
                                DECLARE num_dias int;
                        BEGIN
                                SELECT abs(data_saida - data_entrada) INTO num_dias;
                                RETURN num_dias;
                        END; $$
                        LANGUAGE plpgsql;
```

Calcular_media4

```
CREATE OR REPLACE FUNCTION calcular_media4
    (nota1 DOUBLE PRECISION, nota2 DOUBLE PRECISION, nota3 DOUBLE PRECISION,
                                        nota4 DOUBLE PRECISION)
RETURNS integer AS $$
                                DECLARE media DOUBLE PRECISION;
                                                        BEGIN
                                SELECT abs(nota1 + nota2 + nota3 + nota4)/4 INTO media;
                                                        RETURN media;\n

                                                        END; $$\
                                                LANGUAGE plpgsql;
```

Calcular_media3

```
CREATE OR REPLACE FUNCTION calcular_media3(nota1 DOUBLE PRECISION, nota2
DOUBLE PRECISION,
                                                                nota3 DOUBLE
PRECISION)
                        RETURNS integer AS $$
                                DECLARE media DOUBLE PRECISION;
                        BEGIN
                                SELECT abs(nota1 + nota2 + nota3)/3 INTO media;
        RETURN media;
                        END; $$
LANGUAGE plpgsql;
```

Calcular_mediaFinal

```
CREATE OR REPLACE FUNCTION calcular_mediaFinal(prova DOUBLE PRECISION, media
DOUBLE PRECISION)
                 RETURNS integer AS $$
                DECLARE mediaFinal DOUBLE PRECISION;
                        BEGIN
                                SELECT abs(prova + media)/2 INTO mediaFinal;
                                RETURN mediaFinal;\
                        END; $$
                        LANGUAGE plpgsql;
```

LOG

```
                        CREATE OR REPLACE FUNCTION gera_log()
                        RETURNS TRIGGER
                        AS $$
                        BEGIN
                        INSERT INTO log (data, autor, alteracao, tabela)
                        VALUES (now(), user, TG_OP, TG_RELNAME);
                        RETURN NEW;
                        END;
                        $$ LANGUAGE plpgsql;
                        CREATE TRIGGER tr_gera_log_super_user
                        BEFORE INSERT OR UPDATE OR DELETE
                        ON super_usuario
                        FOR EACH ROW
                        EXECUTE PROCEDURE gera_log();

                        CREATE TRIGGER tr_gera_log_secretaria
                        BEFORE INSERT OR UPDATE OR DELETE
                        ON secretaria
                        FOR EACH ROW
                        EXECUTE PROCEDURE gera_log();
                        CREATE TRIGGER tr_gera_log_coord
                        BEFORE INSERT OR UPDATE OR DELETE
                        ON cord_pedagogo
                        FOR EACH ROW
```

```
                EXECUTE PROCEDURE gera_log();

                CREATE TRIGGER tr_gera_log_diretor
                BEFORE INSERT OR UPDATE OR DELETE
                ON diretor
                FOR EACH ROW
                EXECUTE PROCEDURE gera_log();

CREATE TRIGGER tr_gera_log_endereco
                BEFORE INSERT OR UPDATE OR DELETE
                ON endereco
                FOR EACH ROW
                EXECUTE PROCEDURE gera_log();

CREATE TRIGGER tr_gera_log_contato
                BEFORE INSERT OR UPDATE OR DELETE
                ON contato
                FOR EACH ROW
                EXECUTE PROCEDURE gera_log();

CREATE TRIGGER tr_gera_log_aluno
                BEFORE INSERT OR UPDATE OR DELETE
                ON aluno
                FOR EACH ROW
                EXECUTE PROCEDURE gera_log();

CREATE TRIGGER tr_gera_log_professor
                BEFORE INSERT OR UPDATE OR DELETE
                ON professor
                FOR EACH ROW
                EXECUTE PROCEDURE gera_log();

CREATE TRIGGER tr_gera_log_turma
                BEFORE INSERT OR UPDATE OR DELETE
                ON turma
                FOR EACH ROW
                EXECUTE PROCEDURE gera_log();
```

5.2. DML

BUSCAR_LOGIN **(Buscar todos os usuários, passando login e senha)**
"select u from Usuario u where u.login = :login and u.senha = :senha"

BUSCAR_CPF_PROFESSOR **(Buscar todos os cpfs do professor)**
"SELECT u FROM Professor u WHERE u.cpf = :cpf"

BUSCAR_CPF_RESPONSAVEL

(Buscar todos os cpfs do responsável financeiro)
"SELECT u FROM Resp_Financeiro u WHERE u.cpf = :cpf"

BUSCAR_TIPO_USUARIO **(Buscar o tipo de acesso do usuario)**
"SELECT u FROM Usuario u WHERE u.tipoCargo='"

PESQUISA_RES_FINANCEIRO **(Buscar todos os resp_financeiros a partir da pesquisa)**
"SELECT p FROM Resp_Financeiro p WHERE p.nome LIKE '"

TIPO **Busca o tipo de usuário, retornando a string que o define, a partir da busca de um usuario**
"SELECT";

BUSCAR_CPF_USUARIO **(Buscar todos os cpfs do usuario)**
"SELECT u FROM Usuario u WHERE u.cpf = :cpf";

BUSCAR_TURMA_ORDEM **(descrição funcionalidade)**
SELECT c FROM Turma c order by c.nome

BUSCAR_ALUNO_ORDEM **(descrição funcionalidade)**
SELECT c FROM Aluno c order by c.nome

BUSCAR_DISCIPLINA_ORDEM **(descrição funcionalidade)**
SELECT c FROM Disciplina c order by c.nome

06. ANEXOS (Opcional, válido para a 2ª VA)

DIAGRAMA DE CASO DE USO USUARIO SUPER USUARIO

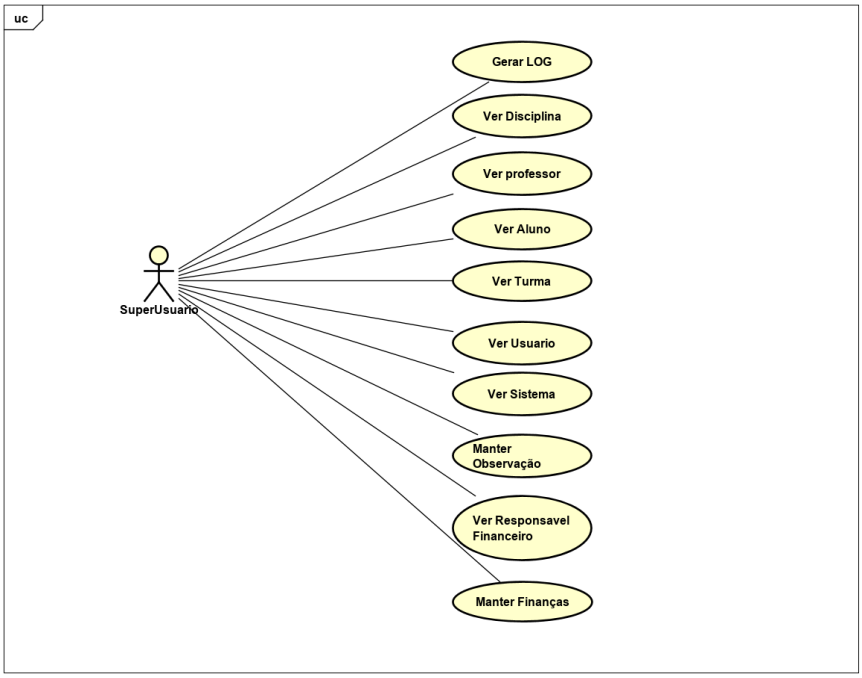


DIAGRAMA DE CASO DE USO USUARIO DIRETOR

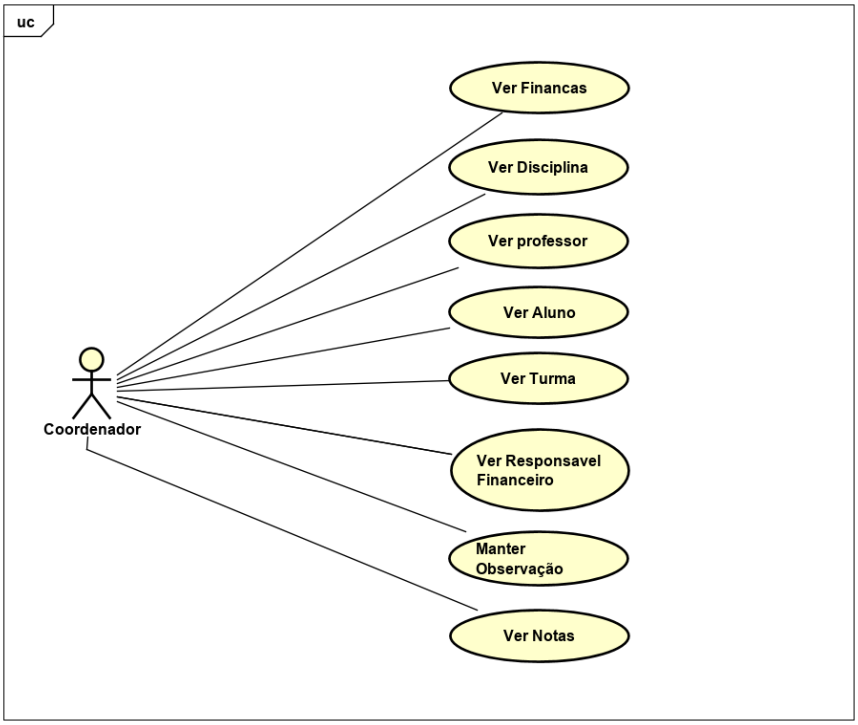


DIAGRAMA DE CASO DE USO USUARIO COORDENADOR PEDAGOGO

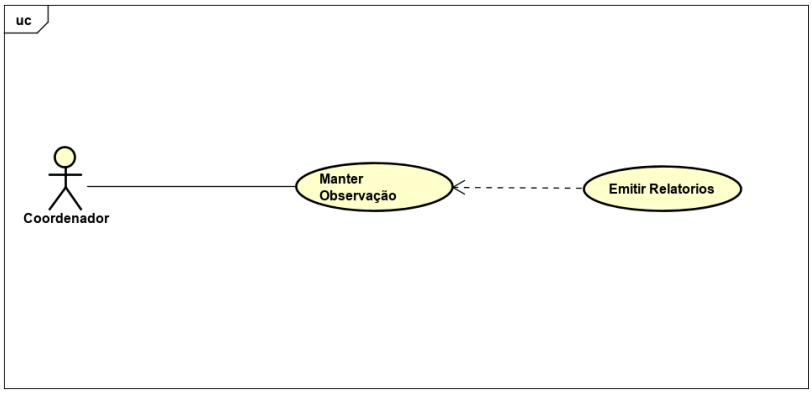


DIAGRAMA DE CASO DE USO USUARIO SECRETARIO