MARCOS OKAMURA RODRIGUES

BANCO DE DADOS

UniPortal

Londrina

2011

1 DESCRIÇÃO

Uniportal é um sistema de portal eletrônico voltado para o registro de informações acadêmicas e administrativas de uma universidade.

O sistema será desenvolvido na linguagem de programação Java com uso do servidor Tomcat e Sistema Gerenciador de Banco de Dados (SGDB) PostgreSQL.

1.1 Descrição dos cenários do sistema

1.1.1 Início do período letivo

O administrador efetuará o cadastramento de alunos ingressantes (oriundos de vestibular ou transferência externa), professores recém-contratados, novos centros, departamentos e cursos de graduação. Os alunos serão identificados pelo número de matrícula, os professores pelo número da chapa e os centros, departamentos e cursos pelos seus respectivos códigos.

Serão adicionadas disciplinas (semestrais ou anuais) para cada curso de graduação com seus respectivos docentes ministrantes. As disciplinas serão identificadas pelos seus respectivos códigos e serão declaradas como essenciais/não essenciais e eletivas/não eletivas.

Os alunos terão seus registros atualizados através de um novo cadastramento de disciplinas cursadas (inclusive dependências) e progressão ou retenção de série de acordo com sua aprovação na série anterior. Os alunos formados, jubilados e desistentes serão descadastrados do portal após 4 anos letivos de ausência.

1.1.2 Durante o ano letivo

O professor efetuará o preenchimento da pauta eletrônica e publicará as notas e faltas dos alunos em cada disciplina ministrada. Além disso, o professor poderá visualizar todas as notas e faltas já publicadas, bem como a lista de suas disciplinas ministradas.

O aluno poderá visualizar apenas informações sobre suas notas e faltas em cada disciplina cursada.

1.1.3 Fim do período letivo

O sistema efetuará o cálculo da média dos alunos nas disciplinas, indicando a necessidade da realização do exame. Após a publicação das notas do exame pelo professor, o sistema aprovará o aluno na disciplina de acordo com sua média final e frequência. Após o fechamento de todas as disciplinas, o sistema aprovará ou não o aluno na série e curso (última série). O aluno poderá visualizar sua média final, nota obtida no exame e aprovação/reprovação em cada disciplina cursada.

1.1.4 Padrão

O administrador poderá efetuar operações de relatório a qualquer momento, solicitando informações sobre os centros, departamentos, cursos, disciplinas, professores, alunos e seus relacionamentos.

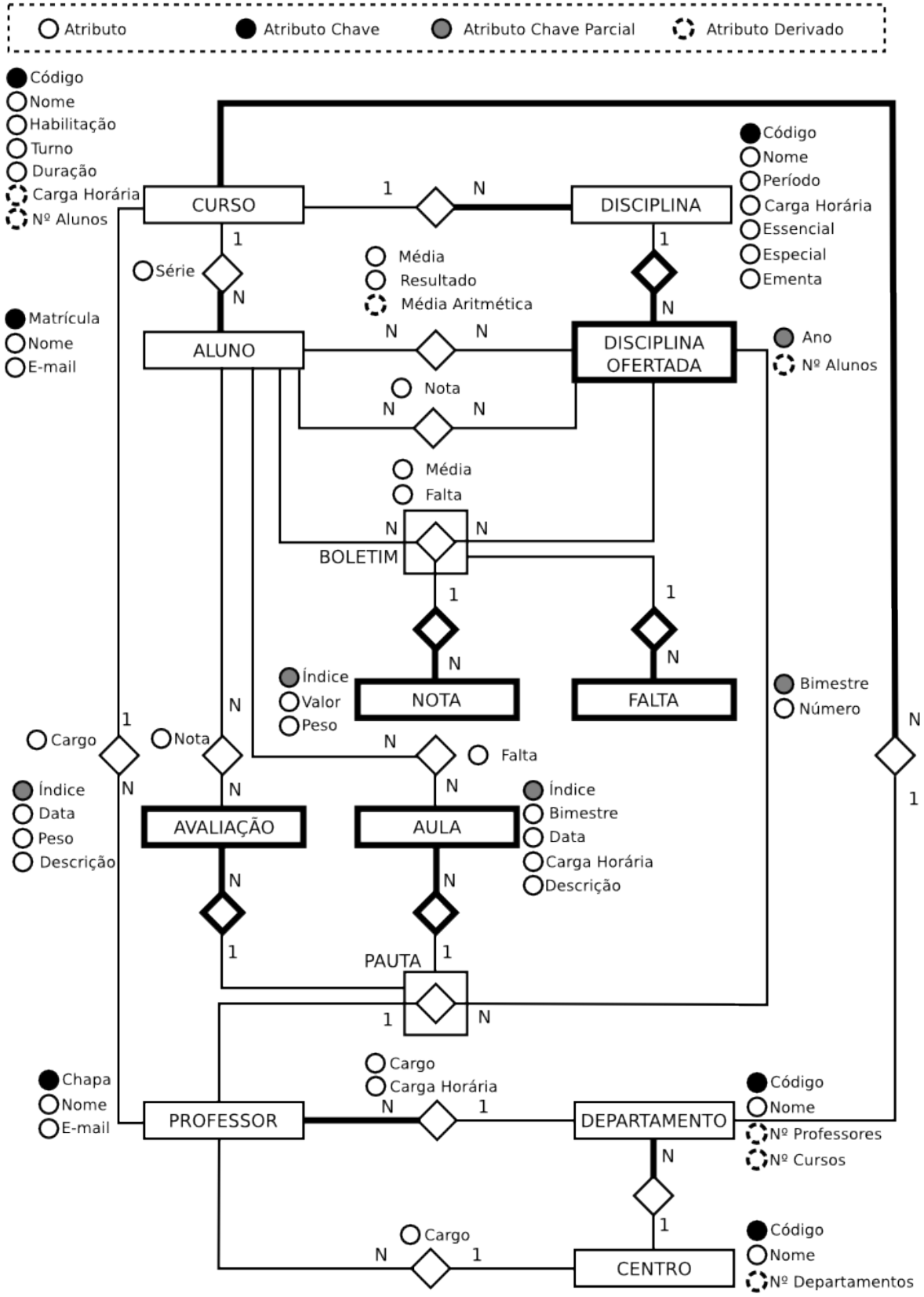
O aluno terá seu histórico escolar e disciplinas do curso sempre disponíveis para impressão.

2 CRONOGRAMA DE DESENVOLVIMENTO

Análise de requisitos	Março
Estudo de ferramentas	Abril
Modelagem	Maio - Junho
Revisão	Setembro - Outubro
Implementação	Setembro - Novembro
Implementação dos pacotes e classes	Setembro - Novembro
Implementação do ambiente web	Setembro - Novembro
Testes	Outubro - Novembro
Teste dos pacotes e classes	Outubro - Novembro
Teste do ambiente web	Outubro - Novembro
Implantação	Novembro

3 Modelagem

3.1 Modelo Entidade Relacionamento (MER)



3.2 Modelo Relacional (MR)

O modelo relacional proposto (modelo 1) está na Forma Normal de Boyce-Codd.

CENTRO (Cent_Código, Cent_Nome)
DEPARTAMENTO (Dep_Código, Dep_Nome, Cent_Código) Cent_Código referencia CENTRO
CURSO (Curs_Código, Curs_Nome, Curs_Habilitação, Curs_Turno, Curs_Duração, Dep_Código) Dep_Código referencia DEPARTAMENTO
DISCIPLINA (Disc_Código, Disc_Nome, Disc_Período, Disc_CargaHorária, Disc_Essencial, Disc_Especial, Disc_Ementa, Curs_Código) Curs_Código referencia CURSO
DISCIPLINA_OFERTADA (Disc_Código, Ofer_Ano) Disc_Código referencia DISCIPLINA
ALUNO (Alun_Matrícula, Alun_Nome, Alun_E-mail, Alun_Série, Curs_Código) Curs_Código referencia CURSO
PROFESSOR (Prof_Chapa, Prof_Nome, Prof_E-mail, Prof_CargaHorária, Prof_CargoCurso, Prof_CargoDepartamento, Prof_CargoCentro, Dep_Código) Dep_Código referencia DEPARTAMENTO
BOLETIM (Alun_Matrícula, Disc _Código, Ofer_Ano, Bol_Média, Bol_Falta) Alun_Matrícula referencia ALUNO Disc_Código, Ofer_Ano referencia DISCIPLINA_OFERTADA
PAUTA (Disc _Código, Disc_Ano, Prof_Chapa) Disc_Código, Ofer_Ano referencia DISCIPLINA_OFERTADA Prof_Chapa referencia PROFESSOR
NOTA (Bol_Matrícula, Bol_Código, Bol_ Ano, Not_Índice, Not_Valor, Not_Peso) Bol_Matrícula, Bol_Código, Bol_Ano referencia BOLETIM
FALTA (Bol_Matrícula, Bol_Código, Bol_Ano, Bimestre, Falt_Número) Bol_Matrícula, Bol_Código, Bol_Ano referencia BOLETIM
AVALIAÇÃO (Paut_Código, Paut_Ano, Índice, Aval_Data, Aval_Peso, Aval_Descrição) Paut_Código, Paut_Ano referencia PAUTA

AULA (Paut_Código, Paut_Ano, Índice, Aul_Bimestre, Aul_Data, Aul_CargaHorária, Aul_Descrição)

Paut_Código, Paut_Ano referencia PAUTA

EXAME (Alun_Matrícula, Disc_Código, Ofer_Ano, Exam_Nota)

Alun_Matrícula referencia ALUNO
Disc_Código, Ofer_Ano referencia DISCIPLINA_OFERTADA

HISTÓRICO (Alun_Matrícula, Disc_Código, Ofer_Ano, Hist_Média, Hist_Resultado)

Alun_Matrícula referencia ALUNO
Disc_Código, Ofer_Ano referencia DISCIPLINA_OFERTADA

RENDIMENTO (Alun_Matrícula, Aval_Código, Aval_Ano, Aval_Índice, Rend_Nota)

Alun_Matrícula referencia ALUNO
Aval_Código, Aval_Ano, Aval_Índice referencia AVALIAÇÃO

FREQUÊNCIA (Alun_Matrícula, Aul_Código, Aul_Ano, Aul_Índice, Freq_Falta)

Alun_Matrícula referencia ALUNO
Aul_Código, Aul_Ano, Aul_Índice referencia AULA

Modelo 1 – Modelo relacional proposto para o sistema UniPortal.

4 Implementação

4.1 Criação de Domínios

4.1.1 D_bimestre

```
CREATE DOMAIN d_bimestre
AS integer
CONSTRAINT d_bimestre_check
CHECK (((VALUE >= 1)
AND (VALUE <= 4)));
```

4.1.2 D_boolean

```
CREATE DOMAIN d_boolean
AS character(1)
CONSTRAINT d_boolean_check
CHECK  (((VALUE = 'S'::bpchar)
OR (VALUE = 'N'::bpchar)));
```

4.1.3 D_indice

```
CREATE DOMAIN d_indice
AS integer
CONSTRAINT d_indice_check
CHECK ((VALUE > 0));
```

4.1.4 D_natural

```
CREATE DOMAIN d_natural
AS integer
CONSTRAINT d_natural_check
CHECK ((VALUE >= 0));
```

4.1.5 D_nota

```
CREATE DOMAIN d_nota
AS real
CONSTRAINT d_nota_check
CHECK (((VALUE >= (0)::double precision)
AND (VALUE <= (10)::double precision)));
```

4.1.6 D_periodo

```
CREATE DOMAIN d_periodo
AS character varying(2)
CONSTRAINT d_periodo_check
CHECK (((((VALUE)::text = 'A'::text)
OR ((VALUE)::text = '1S'::text))
OR ((VALUE)::text = '2S'::text)));
```

4.1.7 D_peso

```
CREATE DOMAIN d_peso
AS integer
CONSTRAINT d_peso_check
CHECK (((VALUE >= 1)
AND (VALUE <= 10)));
```

4.1.8 D_resultado

```
CREATE DOMAIN d_resultado
AS character(1)
CONSTRAINT d_resultado_check
CHECK (((VALUE = 'A'::bpchar)
OR (VALUE = 'R'::bpchar)));
```

4.1.9 D_serie

```
CREATE DOMAIN d_serie
AS integer
CONSTRAINT d_serie_check
CHECK (((VALUE >= 1)
AND (VALUE <= 6)));
```

4.1.10 D_turno

```
CREATE DOMAIN d_turno
AS character(1)
CONSTRAINT d_turno_check
CHECK (((((VALUE = 'M'::bpchar)
OR (VALUE = 'V'::bpchar))
OR (VALUE = 'N'::bpchar))
OR (VALUE = 'I'::bpchar)));
```

4.2 Criação de Tabelas

4.2.1 Centro

```
CREATE TABLE centro(
  cent_id serial NOT NULL,
  cent_codigo character varying(4) NOT NULL,
  cent_nome character varying(100) NOT NULL,

  CONSTRAINT pk_centro PRIMARY KEY (cent_id),
  CONSTRAINT u_centro UNIQUE (cent_codigo)
);
```

4.2.2 Departamento

```
CREATE TABLE departamento(
  dep_id serial NOT NULL,
  dep_codigo character varying(4) NOT NULL,
  dep_nome character varying(100) NOT NULL,
  cent_id integer NOT NULL,

  CONSTRAINT pk_departamento PRIMARY KEY (dep_id),
  CONSTRAINT u_departamento UNIQUE (dep_codigo),
  CONSTRAINT fk_dep_centro FOREIGN KEY (cent_id)
      REFERENCES centro (cent_id) MATCH SIMPLE
      ON UPDATE CASCADE ON DELETE CASCADE
);
```

4.2.3 Curso

```
CREATE TABLE curso(
  curs_id serial NOT NULL,
  curs_codigo character (4) NOT NULL,
  curs_nome character varying(100) NOT NULL,
  curs_habilitacao character varying(100) NOT NULL,
  curs_turno d_turno NOT NULL,
  curs_duracao d_serie NOT NULL,
  dep_id integer NOT NULL,

  CONSTRAINT pk_curso PRIMARY KEY (curs_id),
  CONSTRAINT u_curso UNIQUE (curs_codigo),
  CONSTRAINT fk_curs_departamento FOREIGN KEY (dep_id)
      REFERENCES departamento (dep_id) MATCH SIMPLE
      ON UPDATE CASCADE ON DELETE CASCADE
);
```

4.2.4 Disciplina

```
CREATE TABLE disciplina(
  disc_id serial NOT NULL,
  disc_codigo character(7) NOT NULL,
  disc_nome character varying(100) NOT NULL,
  disc_periodo d_periodo NOT NULL,
  disc_carga_horaria d_natural NOT NULL,
  disc_essencial d_boolean NOT NULL,
  disc_especial d_boolean NOT NULL,
  disc_ementa character varying(1000),
  curs_id integer NOT NULL,

  CONSTRAINT pk_disciplina PRIMARY KEY (disc_id),
  CONSTRAINT u_disciplina UNIQUE (disc_codigo),
  CONSTRAINT fk_disc_curso FOREIGN KEY (curs_id)
      REFERENCES curso (curs_id) MATCH SIMPLE
      ON UPDATE CASCADE ON DELETE CASCADE
);
```

4.2.5 Disciplina Ofertada

```
CREATE TABLE disciplina_ofertada(
  disc_id integer NOT NULL,
  ofer_ano d_natural NOT NULL,

  CONSTRAINT pk_disciplina_ofertada PRIMARY KEY (disc_id, ofer_ano),
  CONSTRAINT fk_ofer_disciplina FOREIGN KEY (disc_id)
      REFERENCES disciplina (disc_id) MATCH SIMPLE
      ON UPDATE CASCADE ON DELETE CASCADE
);
```

4.2.6 Aluno

```
CREATE TABLE aluno (
  alun_id serial NOT NULL,
  alun_matricula character(12) NOT NULL,
  alun_nome character varying(100) NOT NULL,
  alun_email character varying(100) NOT NULL,
  alun_serie d_serie NOT NULL,
  curs_id integer NOT NULL,

  CONSTRAINT pk_aluno PRIMARY KEY (alun_id),
  CONSTRAINT u_aluno UNIQUE (alun_matricula),
  CONSTRAINT fk_alun_curso FOREIGN KEY (curs_id)
      REFERENCES curso (curs_id) MATCH SIMPLE
      ON UPDATE CASCADE ON DELETE CASCADE
);
```

4.2.7 Professor

```
CREATE TABLE professor (
  prof_id serial NOT NULL,
  prof_chapa character (7) NOT NULL,
  prof_nome character varying(100) NOT NULL,
  prof_email character varying(100) NOT NULL,
  prof_carga_horaria d_natural NOT NULL,
  prof_cargo_curso character varying(100),
  prof_cargo_departamento character varying(100),
  prof_cargo_centro character varying(100),
  dep_id integer NOT NULL,

  CONSTRAINT pk_professor PRIMARY KEY (prof_id),
  CONSTRAINT u_professor UNIQUE (prof_chapa),
  CONSTRAINT fk_prof_departamento FOREIGN KEY (dep_id)
      REFERENCES departamento (dep_id) MATCH SIMPLE
      ON UPDATE CASCADE ON DELETE CASCADE
);
```

4.2.8 Boletim

```
CREATE TABLE boletim(
  alun_id integer NOT NULL,
  disc_id integer NOT NULL,
  ofer_ano d_natural NOT NULL,
  bol_media d_nota,
  bol_falta d_natural,

  CONSTRAINT pk_boletim PRIMARY KEY (alun_id, disc_id, ofer_ano),
  CONSTRAINT fk_bol_aluno FOREIGN KEY (alun_id)
      REFERENCES aluno (alun_id) MATCH SIMPLE
      ON UPDATE CASCADE ON DELETE CASCADE,
  CONSTRAINT fk_bol_disciplina_ofertada FOREIGN KEY (disc_id, ofer_ano)
      REFERENCES disciplina_ofertada (disc_id, ofer_ano) MATCH SIMPLE
      ON UPDATE CASCADE ON DELETE CASCADE
);
```

4.2.9 Pauta

```
CREATE TABLE pauta(
  disc_id integer NOT NULL,
  ofer_ano d_natural NOT NULL,
  prof_id integer NOT NULL,

  CONSTRAINT pk_pauta PRIMARY KEY (disc_id, ofer_ano),
  CONSTRAINT fk_paut_disciplina_ofertada FOREIGN KEY (disc_id, ofer_ano)
      REFERENCES disciplina_ofertada (disc_id, ofer_ano) MATCH SIMPLE
      ON UPDATE CASCADE ON DELETE CASCADE,
  CONSTRAINT fk_paut_professor FOREIGN KEY (prof_id)
      REFERENCES professor (prof_id) MATCH SIMPLE
      ON UPDATE CASCADE ON DELETE CASCADE
);
```

4.2.10 Nota

```
CREATE TABLE nota(
  alun_id integer NOT NULL,
  disc_id integer NOT NULL,
  ofer_ano d_natural NOT NULL,
  not_indice d_indice NOT NULL,
  not_valor d_nota NOT NULL,
  not_peso d_peso NOT NULL DEFAULT 1,

  CONSTRAINT pk_nota PRIMARY KEY (alun_id, disc_id, ofer_ano, not_indice),
  CONSTRAINT fk_not_boletim FOREIGN KEY (alun_id, disc_id, ofer_ano)
      REFERENCES boletim (alun_id, disc_id, ofer_ano) MATCH SIMPLE
      ON UPDATE CASCADE ON DELETE CASCADE
);
```

4.2.11 Falta

```
CREATE TABLE falta(
  alun_id integer NOT NULL,
  disc_id integer NOT NULL,
  ofer_ano d_natural NOT NULL,
  falt_bimestre d_bimestre NOT NULL,
  falt_numero d_natural NOT NULL,

  CONSTRAINT pk_falta PRIMARY KEY (alun_id, disc_id, ofer_ano, falt_bimestre),
  CONSTRAINT fk_falt_boletim FOREIGN KEY (alun_id, disc_id, ofer_ano)
      REFERENCES boletim (alun_id, disc_id, ofer_ano) MATCH SIMPLE
      ON UPDATE CASCADE ON DELETE CASCADE
);
```

4.2.12 Avaliação

```
CREATE TABLE avaliacao(
  disc_id integer NOT NULL,
  ofer_ano d_natural NOT NULL,
  aval_indice d_indice NOT NULL,
  aval_data date NOT NULL,
  aval_peso d_natural NOT NULL DEFAULT 1,
  aval_descricao character varying(1000),

  CONSTRAINT pk_avaliacao PRIMARY KEY (disc_id, ofer_ano, aval_indice),
  CONSTRAINT fk_aval_pauta FOREIGN KEY (disc_id, ofer_ano)
      REFERENCES pauta (disc_id, ofer_ano) MATCH SIMPLE
      ON UPDATE CASCADE ON DELETE CASCADE
);
```

4.2.13 Aula

```
CREATE TABLE aula(
  disc_id integer NOT NULL,
  ofer_ano d_natural NOT NULL,
  aul_indice d_indice NOT NULL,
  aul_bimestre d_bimestre NOT NULL,
  aul_data date NOT NULL,
  aul_carga_horaria d_natural NOT NULL,
  aul_descricao character varying(1000),

  CONSTRAINT pk_aula PRIMARY KEY (disc_id, ofer_ano, aul_indice),
  CONSTRAINT fk_aul_pauta FOREIGN KEY (disc_id, ofer_ano)
      REFERENCES pauta (disc_id, ofer_ano) MATCH SIMPLE
      ON UPDATE CASCADE ON DELETE CASCADE
);
```

4.2.14 Exame

```
CREATE TABLE exame(
  alun_id integer NOT NULL,
  disc_id integer NOT NULL,
  ofer_ano d_natural NOT NULL,
  exam_nota d_nota,

  CONSTRAINT pk_exame PRIMARY KEY (alun_id, disc_id, ofer_ano),
  CONSTRAINT fk_exam_aluno FOREIGN KEY (alun_id)
      REFERENCES aluno (alun_id) MATCH SIMPLE
      ON UPDATE CASCADE ON DELETE CASCADE,
  CONSTRAINT fk_exam_disciplina_ofertada FOREIGN KEY (disc_id, ofer_ano)
      REFERENCES disciplina_ofertada (disc_id, ofer_ano) MATCH SIMPLE
      ON UPDATE CASCADE ON DELETE CASCADE
);
```

4.2.15 Histórico

```
CREATE TABLE historico(
  alun_id integer NOT NULL,
  disc_id integer NOT NULL,
  ofer_ano d_natural NOT NULL,
  hist_media d_nota NOT NULL,
  hist_resultado d_resultado NOT NULL,

  CONSTRAINT pk_historico PRIMARY KEY (alun_id, disc_id, ofer_ano),
  CONSTRAINT fk_hist_aluno FOREIGN KEY (alun_id)
      REFERENCES aluno (alun_id) MATCH SIMPLE
      ON UPDATE CASCADE ON DELETE CASCADE,
  CONSTRAINT fk_hist_disciplina_ofertada FOREIGN KEY (disc_id, ofer_ano)
      REFERENCES disciplina_ofertada (disc_id, ofer_ano) MATCH SIMPLE
      ON UPDATE CASCADE ON DELETE CASCADE
);
```

4.2.16 Rendimento

```
CREATE TABLE rendimento(
  alun_id integer NOT NULL,
  disc_id integer NOT NULL,
  ofer_ano d_natural NOT NULL,
  aval_indice d_indice NOT NULL,
  rend_nota d_nota NOT NULL,

  CONSTRAINT pk_rendimento PRIMARY KEY (alun_id, disc_id, ofer_ano,
aval_indice),
  CONSTRAINT fk_rend_aluno FOREIGN KEY (alun_id)
      REFERENCES aluno (alun_id) MATCH SIMPLE
      ON UPDATE CASCADE ON DELETE CASCADE,
  CONSTRAINT fk_rend_avaliacao FOREIGN KEY (disc_id, ofer_ano, aval_indice)
      REFERENCES avaliacao (disc_id, ofer_ano, aval_indice) MATCH SIMPLE
      ON UPDATE CASCADE ON DELETE CASCADE
);
```

4.2.17 Frequência

```
CREATE TABLE frequencia(
  alun_id integer NOT NULL,
  disc_id integer NOT NULL,
  ofer_ano d_natural NOT NULL,
  aul_indice d_indice NOT NULL,
  freq_falta d_natural NOT NULL,

  CONSTRAINT pk_frequencia PRIMARY KEY (alun_id, disc_id, ofer_ano, aul_indice),
  CONSTRAINT fk_freq_aluno FOREIGN KEY (alun_id)
      REFERENCES aluno (alun_id) MATCH SIMPLE
      ON UPDATE CASCADE ON DELETE CASCADE,
  CONSTRAINT fk_freq_aula FOREIGN KEY (disc_id, ofer_ano, aul_indice)
      REFERENCES aula (disc_id, ofer_ano, aul_indice) MATCH SIMPLE
      ON UPDATE CASCADE ON DELETE CASCADE
);
```

4.2.18 Login

```
CREATE TABLE login(
  log_id serial NOT NULL,
  codigo character varying(20) NOT NULL,
  senha character varying(100) NOT NULL,
  tipo character varying(10) NOT NULL,

  CONSTRAINT pk_login PRIMARY KEY (log_id)
);
```

5 Operações disponíveis

Observação: Texto em negrito indica utilização de variável.

5.1 Admin (Universidade)

5.1.1 Cadastrar centro

```
INSERT INTO centro
VALUES (DEFAULT, centCodigo, centNome);
```

5.1.2 Cadastrar departamento

```
INSERT INTO departamento
VALUES (DEFAULT, depCodigo, depNome,
     (SELECT cent_id
     FROM centro
     WHERE cent_codigo = centCodigo));
```

5.1.3 Cadastrar curso

```
INSERT INTO curso
VALUES (DEFAULT, cursCodigo, cursNome, cursHabilitacao, cursTurno,
cursoDuracao,
     (SELECT dep_id
     FROM departamento
     WHERE dep_codigo = depCodigo));
```

5.1.4 Cadastrar disciplina

```
INSERT INTO disciplina
VALUES (DEFAULT, discCodigo, discNome, discPeriodo,
discCargaHoraria, discEssencial, discEspecial, discEmenta,
     (SELECT curs_id
     FROM curso
     WHERE curs_codigo = cursCodigo));
```

5.1.5 Ofertar disciplina

```
INSERT INTO disciplina_ofertada
VALUES (
     (SELECT disc_id
     FROM disciplina
     WHERE disc_codigo = discCodigo), oferAno);
```

5.1.6 Cadastrar aluno

```
INSERT INTO aluno
VALUES (DEFAULT, alunMatricula, alunNome, alunEmail, alunSerie,
     (SELECT curs_id
     FROM curso
     WHERE curs_codigo = cursCodigo));
```

5.1.7 Cadastrar professor

```
INSERT INTO professor
VALUES (DEFAULT, profChapa, profNome, profEmail, profCargaHoraria,
profCargoCurso, profCargoDepartamento, profCargoCentro,
     (SELECT dep_id
     FROM departamento
     WHERE dep_codigo = depCodigo));
```

5.1.8 Cadastrar aluno em disciplina

```
INSERT INTO boletim
VALUES (
     (SELECT alun_id
     FROM aluno
     WHERE alun_matricula = alunMatricula)),
     (SELECT disc_id
     FROM disciplina
     WHERE disc_codigo = discCodigo)), oferAno, NULL, NULL);
```

5.1.9 Cadastrar professor em disciplina

```
INSERT INTO pauta
VALUES (
     (SELECT disc_id
     FROM disciplina
     WHERE disc_codigo = discCodigo), oferAno,
     (SELECT prof_id
     FROM professor
     WHERE prof_chapa = profChapa));
```

5.1.10 Alterar centro

```
UPDATE centro
SET cent_nome = centNome
WHERE cent_codigo = centCodigo;
```

5.1.11 Alterar departamento

```
UPDATE departamento
SET dep_nome = depNome
WHERE dep_codigo = depCodigo;
```

5.1.12 Alterar curso

```
UPDATE curso
SET curs_nome = cursNome
WHERE curs_codigo = cursCodigo;
```

5.1.13 Alterar disciplina

```
UPDATE disciplina
SET disc_nome = discNome
WHERE disc_codigo = discCodigo;
```

5.1.14 Alterar aluno

```
UPDATE aluno
SET alun_nome = alunNome, alun_serie = alunSerie
WHERE alun_matricula = alunMatricula;
```

5.1.15 Alterar professor

```
UPDATE professor
SET prof_nome = profNome, prof_carga_horaria = profCargaHoraria,
     prof_cargo_curso = profCargoCurso,
     prof_cargo_departamento = profCargoDepartamento,
     prof_cargo_centro = profCargoCentro
WHERE cent_codigo = centCodigo;
```

5.1.16 Excluir centro

```
DELETE FROM centro
WHERE cent_codigo = centCodigo;
```

5.1.17 Excluir departamento

```
DELETE FROM departamento
WHERE dep_codigo = depCodigo;
```

5.1.18 Excluir curso

```
DELETE FROM Curso
WHERE curs_codigo = cursCodigo;
```

5.1.19 Excluir disciplina

```
DELETE FROM disciplina
WHERE disc_codigo = discCodigo;
```

5.1.20 Excluir aluno

```
DELETE FROM aluno
WHERE alun_matricula = alunMatricula;
```

5.1.21 Excluir professor

```
DELETE FROM professor
WHERE prof_chapa = profChapa;
```

5.1.22 Calcular média de alunos em disciplina

```
UPDATE boletim
SET bol_media =
     (SELECT SUM (not_valor * not_peso) / SUM (not_peso)
     FROM nota
     WHERE nota.disc_id = boletim.disc_id
     AND nota.ofer_ano = boletim.ofer_ano
     AND nota.alun_id = boletim.alun_id)
WHERE EXISTS(
     SELECT *
     FROM nota
     WHERE nota.disc_id = boletim.disc_id
     AND nota.ofer_ano = boletim.ofer_ano
     AND nota.alun_id = boletim.alun_id
     AND disc_id = (SELECT disc_id
          FROM disciplina
          WHERE disc_codigo = discCodigo)
     AND ofer_ano = oferAno);
```

5.1.23 Calcular faltas de alunos em disciplina

```
UPDATE boletim
SET bol_falta =
     (SELECT SUM (falt_numero)
     FROM falta
     WHERE falta.disc_id = boletim.disc_id
     AND falta.ofer_ano = boletim.ofer_ano
     AND falta.alun_id = boletim.alun_id)
WHERE EXISTS(
     SELECT *
     FROM falta
     WHERE falta.disc_id = boletim.disc_id
     AND falta.ofer_ano = boletim.ofer_ano
     AND falta.alun_id = boletim.alun_id
     AND disc_id = (SELECT disc_id
          FROM disciplina
          WHERE disc_codigo = discCodigo)
     AND ofer_ano = oferAno);
```

5.1.24 Reprovar alunos em disciplina por nota

```
INSERT INTO historico
     (SELECT alun_id, disc_id, ofer_ano, bol_media, 'R'
     FROM boletim
     WHERE bol_media < 3
     AND disc_id =
          (SELECT disc_id
          FROM disciplina
          WHERE disc_codigo = discCodigo)
     AND ofer_ano = oferAno);
```

5.1.25 Cadastrar alunos em exame na disciplina

```
INSERT INTO exame
     (SELECT alun_id, disc_id, ofer_ano, NULL
     FROM boletim
     WHERE bol_media >= 3
     AND bol_media < 6
     AND disc_id =
          (SELECT disc_id
          FROM disciplina
          WHERE disc_codigo = discCodigo)
     AND ofer_ano = oferAno);
```

5.1.26 Aprovar alunos em disciplina por nota

```
INSERT INTO historico
     (SELECT alun_id, disc_id, ofer_ano, bol_media, 'A'
     FROM boletim
     WHERE bol_media >= 6
     AND disc_id =
          (SELECT disc_id
          FROM disciplina
          WHERE disc_codigo = discCodigo)
     AND ofer_ano = oferAno);
```

5.1.27 Reprovar alunos em disciplina após exame

```
INSERT INTO historico
     (SELECT alun_id, disc_id, ofer_ano, (bol_media + exam_nota) /
2, 'R'
     FROM boletim NATURAL JOIN exame
     WHERE ((bol_media + exam_nota) / 2 ) < 6
     AND disc_id =
          (SELECT disc_id
          FROM disciplina
          WHERE disc_codigo = discCodigo)
     AND ofer_ano = oferAno);
```

5.1.28 Aprovar alunos em disciplina após exame

```
INSERT INTO historico
     (SELECT alun_id, disc_id, ofer_ano, (bol_media + exam_nota) /
2, 'A'
     FROM boletim NATURAL JOIN exame
     WHERE ((bol_media + exam_nota) / 2)  >= 6
     AND disc_id =
          (SELECT disc_id
          FROM disciplina
          WHERE disc_codigo = discCodigo)
     AND ofer_ano = oferAno);
```

5.1.29 Reprovar alunos em disciplina por falta

```
UPDATE historico
SET hist_resultado = 'R'
WHERE alun_id =
     (SELECT alun_id
     FROM boletim NATURAL JOIN disciplina
     WHERE bol_falta > 0.25 * disc_carga_horaria
     AND disc_id =
          (SELECT disc_id
          FROM disciplina
          WHERE disc_codigo = discCodigo)
     AND ofer_ano = oferAno);
```

5.1.30 Aprovar aluno no ano letivo

Utiliza as verificações anteriores, aprovando o aluno caso todas as condições abaixo sejam satisfeitas:

1.Não foi reprovado em disciplina essencial;

2.Não foi reprovado em mais de 2 disciplinas;

3.Não foi reprovado por nota e falta.

Obs: Por razões de complexidade, a funcionalidade não foi implementada neste trabalho.

5.1.31 Visualizar todos os centros

```
SELECT *
FROM centro;
```

5.1.32 Visualizar departamentos por centro

```
SELECT *
FROM departamento
WHERE cent_id =
     (SELECT cent_id
     FROM centro
     WHERE cent_codigo = centCodigo);
```

5.1.33 Visualizar cursos por departamento

```
SELECT *
FROM curso
WHERE dep_id =
     (SELECT dep_id
     FROM departamento
     WHERE dep_codigo = depCodigo);
```

5.1.34 Visualizar disciplinas por curso

```
SELECT *
FROM disciplina
WHERE curs_id =
     (SELECT curs_id
     FROM curso
     WHERE curs_codigo = cursCodigo);
```

5.1.35 Visualizar alunos por curso

```
SELECT *
FROM aluno
WHERE curs_id =
     (SELECT curs_id
     FROM curso
     WHERE curs_codigo = cursCodigo);
```

5.1.36 Visualizar professores por departamento

```
SELECT *
FROM professor
WHERE dep_id =
     (SELECT dep_id
     FROM departamento
     WHERE dep_codigo = depCodigo);
```

5.2 Professor

5.2.1 Inserir aula na pauta

```
INSERT INTO aula
VALUES (
     (SELECT disc_id
     FROM disciplina
     WHERE disc_codigo = discCodigo), oferAno, aulIndice,
aulBimestre, aulData, aulCargaHoraria, aulDescricao);
```

5.2.2 Inserir avaliação na pauta

```
INSERT INTO avaliacao
VALUES (
     (SELECT disc_id
     FROM disciplina
     WHERE disc_codigo = discCodigo), oferAno, avalIndice,
avalData, avalPeso, avalDescricao);
```

5.2.3 Inserir frequência de aluno nas aulas

```
INSERT INTO frequencia
VALUES (
     (SELECT alun_id
     FROM aluno
     WHERE alun_matricula = alunMatricula),
     (SELECT disc_id
     FROM disciplina
     WHERE disc_codigo = discCodigo), oferAno, aulIndice,
freqFalta);
```

5.2.4 Inserir rendimento de aluno em uma avaliação

```
INSERT INTO rendimento
VALUES (
     (SELECT alun_id
     FROM aluno
     WHERE alun_matricula = alunMatricula),
     (SELECT disc_id
     FROM disciplina
     WHERE disc_codigo = discCodigo), oferAno, avalIndice,
rendNota);
```

5.2.5 Alterar E-mail

```
UPDATE professor
SET prof_email = profEmail
WHERE prof_chapa = profChapa;
```

5.2.6 Alterar frequência de aluno nas aulas

```
UPDATE frequencia
SET freq_falta = freqFalta
WHERE alun_id =
     (SELECT alun_id
     FROM aluno
     WHERE alun_matricula = alunMatricula)
AND disc_id =
     (SELECT disc_id
     FROM disciplina
     WHERE disc_codigo = discCodigo)
AND ofer_ano = oferAno
AND aul_indice = aulIndice
```

5.2.7 Alterar rendimento de aluno em uma avaliação

```
UPDATE rendimento
SET rend_nota = rendNota
WHERE alun_id =
     (SELECT alun_id
     FROM aluno
     WHERE alun_matricula = alunMatricula)
AND disc_id =
     (SELECT disc_id
     FROM disciplina
     WHERE disc_codigo = discCodigo)
AND ofer_ano = oferAno
AND aval_indice = avalIndice
```

5.2.8 Publicar faltas de alunos

```
INSERT INTO falta
     (SELECT alun_id, disc_id, oferAno, aulBimestre,
SUM(freq_falta)
     FROM frequencia NATURAL JOIN aula
     WHERE disc_id =
          (SELECT disc_id
          FROM disciplina
          WHERE disc_codigo = discCodigo)
     AND ofer_ano = oferAno
     AND aul_bimestre = aulBimestre
     GROUP BY alun_id, disc_id, ofer_ano);
```

5.2.9 Publicar notas de alunos

```
INSERT INTO nota
     (SELECT alun_id, disc_id, oferAno, avalIndice, rend_nota,
aval_peso
     FROM rendimento NATURAL JOIN avaliacao
     WHERE disc_id =
          (SELECT disc_id
          FROM disciplina
          WHERE disc_codigo = discCodigo)
     AND ofer_ano = oferAno
     AND aval_indice = avalIndice);
```

5.2.10 Publicar nota de exame de aluno

```
UPDATE exame
SET exam_nota = examNota
WHERE alun_id =
     (SELECT alun_id
     FROM aluno
     WHERE alun_matricula = alunMatricula)
AND disc_id =
     (SELECT disc_id
     FROM disciplina
     WHERE disc_codigo = discCodigo)
AND ofer_ano = oferAno;
```

5.2.11 Visualizar departamento empregado

```
SELECT *
FROM departamento NATURAL JOIN professor
WHERE prof_id =
     (SELECT prof_id
     FROM professor
     WHERE prof_chapa = profChapa);
```

5.2.12 Visualizar disciplinas ministradas

```
SELECT disc_codigo, disc_nome
FROM disciplina NATURAL JOIN pauta
WHERE prof_id =
     (SELECT prof_id
     FROM professor
     WHERE prof_chapa = profChapa);
```

5.2.13 Visualizar notas de todos os alunos em uma avaliação

```
SELECT alun_matricula, alun_nome, rend_nota
FROM aluno NATURAL JOIN rendimento
WHERE disc_id =
     (SELECT disc_id
     FROM disciplina
     WHERE disc_codigo = discCodigo)
AND ofer_ano = oferAno
AND aval_indice = avalIndice;
```

5.2.14 Visualizar faltas de todos os alunos em um bimestre

```
SELECT alun_matricula, alun_nome, falta FROM
aluno NATURAL JOIN (
     SELECT alun_id, SUM (freq_falta) AS falta
     FROM frequencia NATURAL JOIN aula
     WHERE disc_id =
          (SELECT disc_id
          FROM disciplina
          WHERE disc_codigo = discCodigo)
     AND ofer_ano = oferAno
     AND aul_bimestre = aulBimestre
     GROUP BY alun_id) AS falta_aluno;
```

5.2.15 Visualizar todas as disciplinas ministradas

```
SELECT disc_codigo, nome_disciplina
FROM pauta NATURAL JOIN disciplina
WHERE prof_id =
     (SELECT prof_id
     FROM professor
     WHERE prof_chapa = profChapa);
```

5.3 Aluno

5.3.1 Alterar E-mail

```
UPDATE aluno
SET alun_email = alunEmail
WHERE alun_matricula = alunMatricula;
```

5.3.2 Visualizar curso matriculado

```
SELECT curs_codigo, curs_nome
FROM curso
WHERE alun_id =
     (SELECT alun_id
     FROM aluno
     WHERE alun_matricula = alunMatricula);
```

5.3.3 Visualizar disciplinas cursadas

```
SELECT disc_codigo, disc_nome
FROM disciplina NATURAL JOIN boletim
WHERE alun_id =
     (SELECT alun_id
     FROM aluno
     WHERE alun_matricula = alunMatricula);
```

5.3.4 Visualizar faltas por disciplina

```
SELECT falt_bimestre, falt_numero
FROM falta
WHERE alun_id =
     (SELECT alun_id
     FROM aluno
     WHERE alun_matricula = alunMatricula)
AND disc_id =
     (SELECT disc_id
     FROM disciplina
     WHERE disc_codigo = discCodigo)
AND ofer_ano = oferAno;
```

5.3.5 Visualizar notas por disciplina

```
SELECT not_indice, not_valor, not_peso
FROM nota
WHERE alun_id =
     (SELECT alun_id
     FROM aluno
     WHERE alun_matricula = alunMatricula)
AND disc_id =
     (SELECT disc_id
     FROM disciplina
     WHERE disc_codigo = discCodigo)
AND ofer_ano = oferAno;
```

5.3.6 Visualizar resultado em disciplina

```
SELECT bol_falta, bol_media
FROM boletim
WHERE alun_id =
     (SELECT alun_id
     FROM aluno
     WHERE alun_matricula = alunMatricula)
AND disc_id =
     (SELECT disc_id
     FROM disciplina
     WHERE disc_codigo = discCodigo)
AND ofer_ano = oferAno;
```

5.3.7 Visualizar nota de exame por disciplina

```
SELECT exam_nota
FROM exame
WHERE alun_id =
     (SELECT alun_id
     FROM aluno
     WHERE alun_matricula = alunMatricula)
AND disc_id =
     (SELECT disc_id
     FROM disciplina
     WHERE disc_codigo = discCodigo)
AND ofer_ano = oferAno;
```

5.3.8 Visualizar histórico

```
SELECT disc_codigo, disc_nome, hist_media, hist_resultado
FROM historico NATURAL JOIN disciplina
WHERE alun_id =
     (SELECT alun_id
     FROM aluno
     WHERE alun_matricula = alunMatricula)
```

6 Testes

Os testes serão efetuados para os casos que violem as regras abaixo (com exceção de null):

Campo (s)	Tipo	Exemplo
alun_matricula	sequência de 12 dígitos	'200905600335'
prof_chapa	sequência de 7 dígitos	'0511791'
disc_codigo	sequência de 7 caracteres	'5COP065'
curs_codigo	sequência de 4 caracteres	'056N'
cent_codigo dep_codigo	sequência de 1 até 4 caracteres (a-z/A-Z)	'CCE' 'DC'
cent_nome dep_nome curs_nome curs_habilitacao disc_nome alun_nome prof_nome prof_cargo_curso prof_cargo_departamento prof_cargo_centro	sequência de 1 até 100 caracteres (a-z/A-Z)	'Centro de Ciências Exatas' 'Departamento de Computação' 'Ciência da Computação' 'Bacharelado' 'Banco de Dados' 'Marcos Okamura Rodrigues' 'Evandro Bacarin' 'Coordenador' 'Chefe' 'Diretor'
alun_email prof_email	sequência de 1 até 100 caracteres	'marokamura@gmail.com' 'bacarin@uel.br'
disc_ementa aval_descricao aul_descricao	sequência de 1 até 1000 caracteres	'…' '…' '…'
aval_data aula_data	data	'2011-11-11' '2011-11-11'
disc_periodo	sequência de 1 até 2 caracteres	'A', '1S' e '2S'
curs_turno	um caractere (A-Z)	'M', 'V', N e 'I'
disc_essencial disc_especial	um caractere (A-Z)	'S' e 'N' 'S' e 'N'
hist_resultado	um caractere (A-Z)	'A' e 'R'
falt_bimestre aul_bimestre	um dígito (1-4)	1, 2, 3 e 4 1, 2, 3 e 4
curs_duracao alun_serie	um dígito (1-6)	1, 2, 3, 4, 5 e 6 1, 2, 3, 4, 5 e 6
not_peso aval_peso	número inteiro (1-10)	1, 2, 3, 4, 5, 6, 7, 8, 9 e 10
not_indice aval_indice aul_indice	número inteiro positivo	1 1 1

disc_carga_horaria ofer_ano prof_carga_horaria bol_falta falt_numero aul_carga_horaria freq_falta	número inteiro não negativo	136 2011 40 0 0 4 0
bol_media not_valor exam_nota hist_media rend_nota	número em ponto flutuante não negativo (0.0 – 10.0)	10.0 10.0 10.0 10.0 10.0

7 Manual do Usuário

Será disponibilizado ao usuário o manual em HTML gerado a partir do JAVADOC com a descrição dos métodos das classes da aplicação. Além disso, serão desenvolvidos tutoriais com ilustrações de como utilizar o sistema.

Todas as operações (capítulo 5) e restrições (capítulo 6) do sistema estarão presentes no manual do usuário, em seu respectivo perfil (Admin, Aluno, Professor).