

UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

DEPARTAMENTO DE ESTATÍSTICA E INFORMÁTICA

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

BANCO DE DADOS (BC3) – 2015.2

**PROJETO**

**AVA**

Alunos: Guilherme Melo

    João Nascimento

    Paulo Menezes

Recife - 2015

Sumário

[1 Introdução 3](#_Toc437566022)

[2 Descrição do projeto 3](#_Toc437566023)

[3 Requisitos funcionais 3](#_Toc437566024)

[4 Requisitos não funcionais 4](#_Toc437566025)

[5 Consultas 5](#_Toc437566026)

[6 Relatórios 5](#_Toc437566027)

[7 Casos de Uso 6](#_Toc437566028)

[8 Triggers, procedimento e funções 7](#_Toc437566029)

[9 Conclusão 10](#_Toc437566030)

# Introdução

O projeto AVA surgiu após a implementação de sistema homônimo na Universidade Federal Rural de Pernambuco (UFRPE). AVA é um acrônimo para Ambiente Virtual de Aprendizagem e é o ambiente da UFRPE usado por docentes e discentes, sendo útil ao primeiro para gerenciamento de conteúdo aos alunos, administração do curso e do acompanhamento do desempenho de estudantes, enquanto que para os estudantes, facilita a comunicação com o professor e demais alunos e da permanência de todo o conteúdo a ser utilizado por ele em sua vida acadêmica num único local. Sendo assim, o AVA funciona de forma similar a Edmodo ou Edulify. Aproveitando o lançamento oficial do AVA no segundo semestre de 2015, foi-se confirmado a escolha pelos estudantes para desenvolvê-lo.

No entanto, o projeto AVA não tem por objetivo reproduzir o AVA, pois aquele possui objetivos distintos desse, pois (o projeto AVA) não é focado única e exclusivamente em funcionalidade do AVA, mas também em funcionalidades do Sig@ (Sistema de Informações e Gestão Acadêmica). Seria até, talvez, mais adequado o uso de Sig@VA ou qualquer aglutinação desses dois siglemas para representar o projeto. Ademais, o projeto AVA tem como objetivos fornecer uma aplicação capaz de modular um ambiente de gerenciamento de informações pelos atores do sistema (professor, aluno e secretária) e permitir a troca dessas informações, construindo um ambiente integrado e uma ferramenta que busca simplificar processos comuns da vida acadêmica.

# Descrição do projeto

O Ambiente Virtual de Aprendizagem (AVA) é um sistema usado para facilitar a gerência e a comunicação do corpo discente e docente de uma universidade. Os usuários do sistema têm acesso a serviços do ambiente virtual, tais como: se comunicar com usuários online, criar ou participar de fóruns, criar eventos e visualizar o calendário com os próximos eventos (das disciplinas que está matriculado ou que ministra) e ter acesso ao perfil da disciplina. Dentro do sistema também é permitido a disponibilização de atividades pelos professores e submissão da atividade pelo aluno.

# Requisitos funcionais

1. O sistema é composto por usuários que possuem CPF, nome, e-mail, senha e tipo (professor, aluno ou administrador[adm]). Esses usuários podem ser alunos ou professores. Professores podem desempenhar a função de coordenadores dos cursos oferecidos pela universidade.
2. O aluno de uma universidade ou é da graduação (graduando) ou da pós-graduação (mestrando ou doutorando). Para tal condição (a de aluno), deve estar matriculado em um determinado curso da universidade, na qual possui vários alunos.  Na universidade existem vários cursos disponíveis.
3. Um curso tem como atributos: nome, código de identificação, departamento a qual está associado, quantidade de alunos matriculado no curso que ainda estejam vinculados a universidadee seu tipo (graduação ou pós-graduação). Cada curso deve possuir várias disciplinas e um disciplina pode estar vinculada a mais de um curso.
4. Uma disciplina possui: código, nome, carga horária e nº créditos. Uma disciplina pode exigir nenhum ou alguma(s) disciplina(s) como pré-requisito. Só é permitido "pagar" uma disciplina se, e somente se, todos os pré-requisitos forem satisfeitos. Os pré-requisitos da disciplina que o aluno deseja ser matriculado serão comparados com as disciplinas do seu histórico e verificado se os pré-requisitos foram “pagos” e aluno obteve êxito (conseguiu aprovação) neles para serem “satisfeitos”. Cada disciplina deve oferecer uma ou mais ofertas de disciplina para a matrícula de alunos, na qual cada oferta está vinculada a uma disciplina.
5. A oferta de uma disciplina disponibilizará: semestre, ano, dias e horário de realização das aulas da disciplina, bem como quantidade de alunos matriculados. Cada oferta de disciplina pode ter vários alunos matriculados.
6. Uma oferta de disciplina deve ser ministrada por um professor. Um professor pode ministrar mais de uma oferta de disciplina. O professor ficará responsável de organizar todo o calendário dessa oferta da disciplina.
7. É necessário que o aluno esteja matriculado ao menos em uma disciplina e no máximo em seis. Na realização da matrícula do aluno, deve ser guardado a data de realização de matrícula e o número de protocolo.
8. O aluno que está matriculado numa oferta de uma disciplina deve possuir notas que são usadas para verificar o aproveitamento e a aprovação do aluno. A média geral deve estar vinculada a um aluno.
9. A nota de cada aluno, tem como atributos: notas da 1ª VA, 2ª VA, 3ª VA e prova final. Para conseguir aprovação na disciplina, o aluno deve ter uma média igual ou superior a 7 (sete) na oferta da disciplina quando até a realização da terceira VA, e 5 (cinco) ou mais quando realizada a prova final. Conseguindo até o término da oferta da disciplina igual ou acima da média, a situação do aluno naquela disciplina é tida como aprovada. Senão, ele é reprovado. A média é calculada da seguinte forma: (VA1ª Maior Nota + VA 2ª Maior Nota) / 2.  Se não obtiver média suficiente para sua aprovação até a 3ª VA, a sua média final é calculada a seguir: (Média aritmética das duas maiores notas (1ª VA, 2ª VA e 3ª VA) + Nota da prova final) /2.
10. Professores podem criar projetos de pesquisas. Um projeto de pesquisa tem um título, um código de identificação, modalidade (PIBIC, PIBIT, PICME etc.), organização (CNPq, FACEPE, Capes etc.), valor da bolsa dada aos interessados e nº de vagas restantes. Um professor pode coordenar mais de um projeto, e um projeto deve ser coordenado por um ou mais professores.
11. Alunos podem participar de projetos de pesquisa. Um projeto deve ter participação de alunos. Vários alunos podem participar de vários projetos de pesquisa.
12. Com o desenvolvimento de um projeto de pesquisa, um ou mais artigos podem ser gerados. Um artigo está associado ao código de identificação de um projeto. O artigo tem como atributos: nome, tema, objetivo e área científica.

# Requisitos não funcionais

1. O professor atribui as notas (podem ter valores distintos) aos alunos individualmente.
2. O número de protocolo da matricula em um curso será gerado automaticamente e aleatoriamente pelo sistema.
3. Os usuários podem visualizar avisos num quadro (tela) direcionados a ele (por pagar uma oferta de disciplina ou enviado diretamente a ele) ou públicos (todos podem visualizar). Os avisos são organizados de acordo com sua prioridade (prioridade mais alta está acima dos de prioridade mais abaixo). O aviso só pode ser visualizado a determinado conjunto, podendo ser direcionado a: uma pessoa, a uma turma (oferta de uma disciplina) ou a todos os usuários.
4. A nota de um aluno será de 0 (zero) a 10 (dez). O professor poderá publicar e modificar a nota dentro do prazo definido pela universidade.

# Consultas

**Aluno**

Com o professor como ator do sistema, é possível fazer consulta:

* À sua grade de aulas do período vigente e seu histórico escolar. A consulta a grade de aula retorna todas as disciplinas que o aluno está "pagando" no período.  Já histórico escolar de um aluno, tem as disciplinas cursadas por ele durante todo o curso com sua situação (aprovado [por média ou final] ou reprovado), sua média geral e parcial e o período no qual foi cursado.
* O aluno poderá consultar uma tela na qual se pode visualizar todos os avisos a ele direcionado. Ele também pode consultar suas notas das matérias do período que ele está cursando no momento.
* Informações sobre o aluno que estão inseridas no sistema AVA (CPF, por exemplo)

**Professor**

Com o professor como ator do sistema, é possível fazer consulta:

* Dos alunos que estão matriculados na sua disciplina.
* Assim como aluno, o professor pode consultar seus avisos.
* O professor que é um coordenador de curso é capaz também de listar todos os alunos do curso que ele coordena.  O coordenador também pode manipular a grade curricular de um curso, como manipular disciplinas e professores que ensinarão uma disciplina.
* O professor será capaz de visualizar todo o projeto de pesquisa que ele está envolvido, como também alunos que estão participando. Ele poderá também consultar todos artigos produzidos nos projetos de pesquisa.
* Informações sobre o professor que estão inseridas no sistema AVA (CPF, departamento de origem, por exemplo)
* Pode consultar a situação de cada aluno após o fim da oferta da disciplina por ele ministrada

# Relatórios

**Aluno**

* Grade de horário do aluno por semestre
* Histórico escolar do aluno até o período vigente
* Notas em cada oferta paga por ele (podendo ser por período ou "sua vida acadêmica")
* Disciplinas ofertadas pelo curso, na qual o aluno esteja matriculado, por semestre
* Eventos do calendário que estão direcionados ao aluno (da disciplina que está matriculado, por exemplo)

**Professor**

* Alunos matriculados em cada disciplina ofertada que o professor ministra
* Notas de cada aluno matriculado em suas disciplinas
* Alunos que participam de um projeto com ele
* Listar todos os alunos pertencentes ao curso (adm)
* Listar professores que estão no departamento (adm)
* Relatório de atividades por aluno, período, curso e disciplina
* Geração de seus horários de aulas de um semestre em particular
* Boletim de notas (provas e atividades) por aluno e curso (por semestre específico)
* Relatório final de aluno, com sua situação de “aprovado” ou “reprovado”
* Relatório de alunos que ficaram para fazer a prova final, por disciplina e semestre.

# Casos de Uso

**Cadastrar usuário**

1. O administrador seleciona a opção cadastrar usuário
2. O administrador passa os dados necessários para o cadastro
3. O sistema cria e salva um novo usuário no sistema
4. O sistema mostra que o usuário foi cadastrado com sucesso

*Fluxo secundário:* No passo 3, caso o usuário já estiver cadastrado, o sistema notifica que o usuário já possui uma conta e volta ao passo 2.

**Matricular aluno**

1. O aluno escolhe a opção de matrícula no sistema
2. O sistema disponibiliza todas as ofertas que o aluno está apto a pagar
3. O aluno faz escolhe das ofertas desejadas (de 1 a 6 ofertas)
4. O aluno confirma a matrícula nas ofertas
5. O sistema cadastrará as ofertas escolhidas

*Fluxo secundário:* No passo 2, se o aluno escolher menos de 3 ofertas ou mais de 10 ofertas, o sistema o notifica que chegou ao limite de escolhas. No passo 5, caso o aluno já tenha efetuado o cadastro no sistema, o sistema notificará e retornará ao passo 2.

**Participar de um projeto de pesquisa**

1. O aluno escolhe a opção de projeto de pesquisa no menu do sistema
2. O sistema mostra todos os projetos de pesquisa cadastrados com vagas disponíveis
3. O aluno escolhe os projetos que deseja participar e manda uma requisição para participar do(s) projeto(s) escolhido(s)
4. O sistema envia as requisições aos professores
5. O professor aceita o a solicitação
6. O sistema verifica que foi aceita a requisição
7. O aluno é vinculado ao projeto de pesquisa

*Fluxo* *secundário:* No passo 5, caso o professor não aceite a solicitação, o sistema não vincula o aluno ao projeto e o sistema mostra que a solicitação foi recusada.

**Adicionar artigo a projeto de pesquisa**

1. O professor seleciona o projeto que coordena e que deseja inserir o artigo
2. O professor escolhe a opção de criar um artigo
3. O professor passa para o sistema informações sobre o artigo
4. O sistema colhe a informação e cria um novo artigo vinculado a um projeto
5. O artigo fica disponível para manipulação no sistema

# Triggers, procedimento e funções

Uma breve descrição das triggers, procedimentos e funções usadas no projeto AVA.

**Triggers**

* ***salvarAlunoProjeto after update on solicitacaoprojeto*** - após qualquer mudança na tabela de *solicitacaoprojeto* é tomado uma decisão que depende da seguinte condição. Se a variável booleana *estado = 1*, que indica que foi aprovada sua solicitação para a entrada no projeto, insira uma linha na tabela de *participarprojeto* e delete sua linha em *solicitacaoprojeto*. Caso contrário, *estado = 0*, apenas remova a linha da tabela *solicitacaoprojeto*.
* ***inserirAvisoAposMatriculaEInserirNotaNaTabela after insert on matricular*** - após inserida uma linha na tabela matricular, a trigger apenas insere uma linha na tabela nota para que as notas do aluno naquela oferta possam ser inseridas. Além disso, um aviso é criado após a inserção de linha na tabela *matricular*, direcionado ao aluno matriculado para que saiba que houve a confirmação de sua matrícula.
* ***inserirAlunoNaOfertaEmHistorico after insert*** ***on nota***- sua função é apenas inserir uma linha na tabela, porém na tabela *historico*, toda vez que for inserida uma linha em nota.
* ***avisarAlunoNotaECalcularMedia after update on nota*** - após uma modificação de uma nota na tabela nota. O aluno associado àquela nota modificada, receberá um aviso informando que sua nota já se encontra no sistema, e, portanto, pode ser consultada. Além disso sua nota é calculada toda vez que uma nota é atualizada, chamando o procedimento *calcularMediaAluno()[*veja mais detalhes sobre esse procedimento na parte de procedimento do documento].
* ***atualizarNAlunosOferta after update on matricular*** - essa trigger é engatilhada toda a vez que houver uma modificação na tabela matricular. A tabela matricular em sua essência não permite atualização de uma linha, devido ao seu significado. Ou se está matriculado numa oferta de disciplina ou não está. Por isso, quando adicionada uma linha nessa tabela, é feita uma atualização na variável *qtdAlunos* da tabela *ofertadisciplina*, adicionando uma unidade a ela. O contrário, a remoção de uma linha, também é válido, porém, o número de alunos (*qtdAlunos*) é diminuído uma unidade.
* ***atualizarNAlunosCurso after update on aluno*** - assim como a trigger anterior, ela faz a atualização do número de alunos, só que dessa vez, de um curso. Toda vez que um novo aluno for inserido no banco, mais precisamente na tabela aluno, ela atualizará a variável *qtdAlunos* da tabela *curso*, somando uma unidade. Se um aluno for deletado da tabela, será subtraído um da quantidade de alunos (*qtdAlunos*) da tabela *curso*.
* ***criarAvisoProjeto after insert on projetopesquisa*** - após inserido um novo projeto pelo professor no banco de dados, um aviso é criado automaticamente com informações básica sobre o novo projeto a todos os usuários cadastrados do banco.
* ***avisarProfessorDeSolicitacao after insert on solicitacaoprojeto*** - quando o aluno se interessa por uma projeto de pesquisa é possível que solicite ao professor sua entrada nele. Após a solicitação do projeto pelo aluno, um aviso é criado direcionado ao professor que coordena o projeto.
* ***atualizarNVagasProjeto after update on participarprojeto*** - toda vez que ocorrer alguma alteração na tabela *participarprojeto*, uma inserção ou deleção (pela sua natureza não existe atualização, pois ou você está participando de um projeto X ou não está), ele diminui ou aumenta uma unidade a variável *nVagas* na tabela *projetopesquisa*, respectivamente. Além disso, é criado um aviso para informar aos interessados, a depender do que tenha acontecido com a tabela. Se houve uma inserção na tabela *participarprojeto*, um aviso é criado somente para o aluno que conseguiu a confirmação de sua solicitação de participar projeto de pesquisa. Se houve uma deleção, um aluno deixou de participar de um projeto, um aviso é criado direcionados a todos os alunos sobre a oportunidade de uma vaga no projeto.

**Procedimentos**

* ***atualizarNAlunosCursoQuandoNull (in idC int)*** - procedimento usado apenas para atualizar o número de alunos nos cursos já criados no banco de dados com os dados que estão já lá inseridos. É passado como argumento o *idC*, que é o identificador do curso. Usada para evitar a inserção manual dos dados e de um possível erro.
* ***atualizarNAlunosOfertaQuandoNull (in idO int)*** - função semelhante ao procedimento acima, porém atualiza o número de alunos de cada oferta já inclusa no banco de dados. É passado como argumento o *idO*, que é o identificador da oferta. Usada para evitar a inserção manual dos dados e de um possível erro.
* ***atualizarAlunosCurso()*** - procedimento que itera a variável inteira *nCurso* de um a duzentos chamando o procedimento *atualizarNAlunosCursoQuandoNull(in idC int)*, passando *nCurso* como parâmetro.
* ***atualizarAlunosOferta()*** - procedimento que itera a variável inteira *nOferta* de um a duzentos chamando o procedimento *atualizarNAlunosOfertaQuandoNull(in idO int),* passando *nOferta* como parâmetro.
* ***calcularMediaDeCadaAlunoNaOferta(in idO int)*** - procedimento com intuito de atualizar o banco de dados com a média e condição na tabela *historico* de cada aluno nas ofertas já cadastradas.
* ***calcularMediaAluno(idOf int, cpfAl int)*** - procedimento usado para o cálculo da média e condição do aluno (aprovado, aprovado por média ou reprovado). É chamada toda a vez que há inserção de nota (atualização de uma das variáveis de nota, que são: *nota1*, *nota2*, *nota3* e *notaFinal*) de um aluno numa oferta de disciplina. São os seus parâmetros: *in idOf int*, identificador da oferta e *in cpfAl varchar(14)*, CPF do aluno.
* ***atualizarHistorico()*** - procedimento parecido com atualizarAluno[...](), porém itera a variável inteira *nOferta* de um a duzentos passando-a como argumento do procedimento *calcularMediaDeCadaAlunoNaOferta(in idO int*), atualizando todo o histórico já existente no banco de dados.
* ***adicionarAlunoAOferta(in cpfAl varchar(14), in idO int, in dataMatr date, in numProt varchar(20))*** - adiciona um aluno a uma oferta cadastrada no sistema, ou seja, um aluno é matriculado numa oferta na qual solicitou matrícula. Para isso, é inserida uma linha na tabela matricular passando os parâmetros: *in cpfAl varchar(14)*, o CPF do aluno; *in idO int*, o identificador de oferta; *in dataMatr date*, a data de matrícula na oferta e *in numProt varchar(20)*, número de protocolo da matrícula efetuada.
* ***adicionarProfessorAOferta(in cpfProf varchar(14), in idO int)*** - adiciona um professor a uma oferta cadastrada, ou seja, o professor passa a ministrar aquela oferta da disciplina. Para tal, é inserida uma linha na tabela *ministraroferta* passando os parâmetros: *in cpfProfessor varchar(14),* o CPF do professor e *in idO int*, a oferta na qual o professor ministrará suas aulas.
* ***adicionarSolicitacaoDeProjetoDeUmAluno(in cpfAl varchar(14), in idP int)*** - adiciona uma linha na tabela *solicitacaoprojeto* quando o aluno manifesta interesse em participar de um projeto de pesquisa de um professor. São passados os parâmetros: *in cpfAl varchar(14),* CPF do aluno, e *in idP int*, identificador do projeto.
* ***adicionarAlunoAoProjeto (in cpfAl varchar(14), in idP int)*** - adiciona uma linha na tabela *participarprojeto*. Isso ocorre quando o professor aceita a solicitação do aluno de participar no projeto por ele desenvolvido. São passados os parâmetros: *in cpfAl varchar(14)*, CPF do aluno, e *in idP int*, identificador do projeto.
* ***adicionarNotaAAluno (in cpfAl varchar(14), in idOf int, in nota double)*** - é atualizada uma linha na tabela nota. Esse procedimento é usado para inserir uma determinada nota (valor da nota) a um aluno numa oferta de disciplina. São passados os parâmetros: *in cpfAl varchar(14)*, CPF do aluno; *in idOf int*, identificador da oferta de disciplina (turma) e *in nota double*, nota a ser inserida.

**Função**

* ***buscarCpfProfessor (idP int) returns varchar(14) deterministic*** - função que busca o CPF de um determinado professor e o retorna, passando *idP int* (identificador da tabela *projetopesquisa*).
* ***buscarNomeProjeto (idP int) returns varchar(35) deterministic*** - função que retorna o nome de um projeto de pesquisa, sendo passado como parâmetro *idP int*.
* ***buscarNomeUsuario (cpfU varchar(14)) returns varchar(35) deterministic*** - função que retorna o nome de usuário dado como parâmetro o seu CPF.
* ***buscarIdDisciplina (idOf int) returns int deterministic*** - retorna o identificador de uma disciplina (na tabela, *idDisciplina*), sendo passado o parâmetro *idOf int*, que é o identificador de uma oferta de disciplina. Usada basicamente como função secundária para encontrar o nome da disciplina, que é feito pela função logo abaixo.
* ***buscarNomeDisciplina (idDisc int) returns varchar(35) deterministic*** - retorna o nome da disciplina dado *idDisc int* (identificador de disciplina) como seu parâmetro.

# Conclusão

Tantos os objetivos como também as funcionalidades na data de entrega do projeto foram cumpridas. Porém, é notável que ainda é um sistema aquém do AVA. Mas ainda, é válido lembrar que essa não foi a pretensão antes ou no discorrer do projeto para Paradigmas de Programação e Banco de Dados, pois é sabido que um sistema desse porte é necessário um maior tempo ou recurso humano. Com uma base sólida, talvez, possa se expandir o projeto e o desenvolvendo em outras disciplinas do curso de Ciência da Computação e o levando ao mesmo patamar da aplicação original, que deu base ao projeto.

Contudo, o projeto AVA conseguiu implementar todos os objetivos na qual foi focado durante seu desenvolvimento. Logo, chegada a data de sua entrega se conseguiu alcançar as expectativas estabelecidas no início do projeto.