



LABORATÓRIO DE SOFTWARE E PROJETOS

DESENVOLVIMENTO DE SOFTWARE

Autor: Me. Rodrigo Ramos Nogueira

Revisor: Feliipe Oviedo Frosi

INICIAR



introdução

Introdução

Prezado(a) aluno(a), você já domina diversos aspectos do processo de criação de um projeto de *software*, bem como da gestão ágil de projetos utilizando *Scrum*. Esta unidade tem como objetivo consolidar o seu conhecimento sobre projeto de *software* e desenvolvimento. Vamos trazer um pouco do conteúdo já visto, juntamente com novos conceitos de uma maneira incremental. Por fim, você verá todas as etapas do desenvolvimento de um sistema, desde a entrevista, passando pelo desenvolvimento e, por último, o teste do sistema.

Preparando o Projeto

Durante a etapa de desenvolvimento de um projeto, deve-se ter em mãos toda a documentação realizada anteriormente. O primeiro documento é justamente o *briefing* que descreve o conteúdo do sistema. Como já fizemos anteriormente, o *briefing* pode organizar uma entrevista. No entanto, vamos ver agora um exemplo de uma entrevista direcionada, ou seja, aquela em que o entrevistador realiza perguntas previamente elaboradas.

De modo geral, o que o sistema irá fazer?

R.: O sistema deverá fazer a gestão acadêmica da instituição.

No contexto da rotina da sua instituição, o que ele não irá fazer?

R.: Não haverá módulo financeiro nem fiscal.

Já existe algum sistema sendo utilizado pela empresa?

R.: Não, mas temos o cadastro dos alunos em planilha excel.

Quais serão os usuários do sistema e suas particularidades?

R.: O administrador será responsável por cadastrar e extrair relatórios do sistema. Os professores poderão lançar frequência e notas, bem como saber quantas aulas ministraram. Os alunos terão acesso apenas aos relatórios da sua nota e frequência.

Dando continuidade, iremos descrever os requisitos para o sistema acadêmico que está sendo desenvolvido. Para complementar o seu conhecimento, nos requisitos, vamos complementar o documento com a prioridade, que pode ser baixa, média ou alta.

Código	Requisito	Prioridade
RF01	O sistema deve permitir Cadastrar Aluno	Alta
RF02	O sistema deve permitir Consultar Aluno	Alta
RF03	O sistema deve permitir Enviar notificações	Média
RF04	O sistema deve permitir Enviar mensagem ao Professor	Baixa
RF05	O sistema deve permitir Cadastrar Professor	Alta
RF06	O sistema deve permitir Consultar Professor	Alta
RF07	O sistema deve permitir Enviar mensagem aos alunos	Baixa
RF08	O sistema deve permitir Fazer relatório de horas	Alta
RF09	O sistema deve permitir fazer relatório de frequência	Alta

Quadro 4.1 - Requisitos Funcionais

Fonte: Elaborado pelo autor.

#PraCegoVer: Neste quadro constam os requisitos funcionais em três colunas que são: Código, Requisito e Prioridade. Esta tabela tem 9 linhas de requisitos passando por sistemas de cadastrar aluno, consulta de professor até relatório de frequência. Na primeira linha código RF01 requisito o sistema deve permitir cadastrar aluno prioridade alta; na segunda linha RF02 o

sistema deve permitir consultar aluno prioridade alta; RF03 o sistema deve permitir enviar notificações prioridade média; RF04 o sistema deve permitir enviar mensagens ao professor prioridade baixa; RF05 o sistema deve permitir cadastrar professor prioridade alta; RF06 o sistema deve permitir consultar professor prioridade alta; RF07 o sistema deve permitir enviar mensagens aos alunos prioridade baixa; RF08 o sistema deve permitir fazer relatório de horas prioridade alta; RF09 o sistema deve permitir fazer relatório de frequência prioridade alta.

Complementarmente, foi realizado o levantamento de alguns requisitos funcionais. Lembrando que os requisitos funcionais são aqueles que dependem dos agentes externos ao sistema, ou seja, não se trata de uma funcionalidade.

Código	Requisito	Prioridade
RNF01	O sistema deve garantir a segurança dos dados	Alta
RNF02	O sistema deve permitir o acesso responsivo às telas do sistema	Baixa
RNF03	O sistema deve permitir integração com os dados já obtidos anteriormente e sistemas existentes	Média

Quadro 4.2 - Requisitos Funcionais

Fonte: Elaborado pelo autor.

#PraCegoVer: Segundo quadro de requisitos funcionais com três colunas que são: Código, Requisito e Prioridade. Esta tabela tem 3 linhas de requisitos passando por sistema que deve garantir a segurança dos dados até os dados obtidos anteriormente e os sistemas existentes. Na primeira linha RNF01 o sistema deve garantir a segurança dos dados prioridade alta; Na segunda

linha RNF02 o sistema deve permitir o acesso responsivo as telas do sistema prioridade baixa; Na terceira linha RNF03 o sistema deve permitir integração com os dados já obtidos anteriormente e sistemas existentes prioridade média.

Uma vez tendo os requisitos em mãos, é realizado o diagrama de casos de uso. O diagrama de casos de uso traz uma visão geral do comportamento dos atores do sistema com as funcionalidades que serão desenvolvidas.

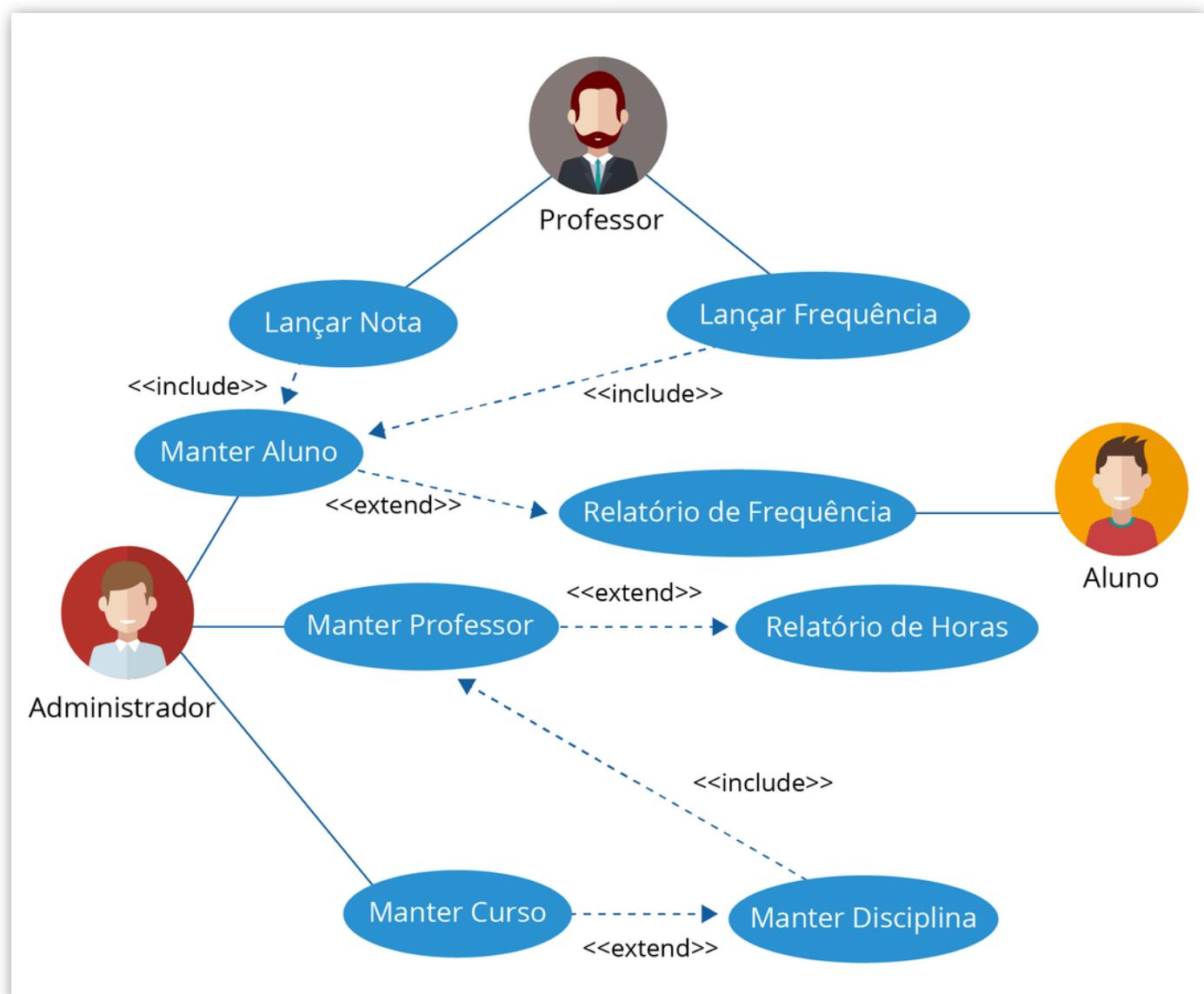


Figura 4.1 - Diagrama de casos de uso

Fonte: Elaborada pelo autor.

#PraCegoVer: Diagrama de caso de uso no qual temos várias relações. Temos a relação de atores como de professor com o Administrador e o Aluno. E de

funcionalidades como: lançar notas, lançar frequências, manter professor, manter disciplinas e outras. O professor pode lançar nota e que vai incluir dentro do manter aluno; o professor pode lançar frequência e também vai incluir dentro do manter aluno; o administrador vai ver manter o aluno, manter o professor e manter o curso; o aluno vai ter o relatório de frequência.

Uma abordagem durante o desenvolvimento de um caso de uso é a utilização do manter. O manter representa as operações principais em cima de determinado dado: inserir, alterar, listar e remover. Desse modo, quando se utiliza o manter, permite-se deixar o diagrama com menos elementos, mantendo o foco nas demais operações que o sistema deve realizar.

reflita

Reflita

A cada etapa desta Unidade, estamos revisando o conteúdo de projeto e desenvolvimento de um sistema de Informação. Um documento de software educacional completo pode chegar a mais de 100 páginas, bem como seus diagramas. Por isso, estamos utilizando trechos menores, para que se torne mais agradável a leitura, bem como a didática do seu aprendizado.

Vamos pegar como exemplo o caso de uso Manter Aluno e, sabendo que representa Inserir aluno, Excluir aluno, Editar aluno e Listar aluno, vamos ver como fica o caso de uso descriptivo do Cadastrar Aluno.

Caso de Uso	Alugar livro.
Autor Principal	Cadastrar Aluno.
Interessados e Interesses	Administrador: usuário responsável por realizar os cadastros do sistema. Deverá utilizar informações dos alunos para alimentar o sistema.
Pré-condições	O Administrador deverá estar logado no sistema.
Pós-condições	Os dados serão armazenados. O usuário será direcionado para a tela de Inserir Aluno em uma disciplina.
Cenário Principal	<ol style="list-style-type: none">1. O Administrador obtém os dados do Aluno.2. O Administrador clica em Menu Alunos >> Cadastrar Novo Aluno.3. O Administrador solicita os documentos para o Aluno.4. O Administrador insere os dados:<ul style="list-style-type: none">• Nome do Aluno;• CPF;• RG;• Data de Nascimento;• Endereço completo;• Telefones para Contato.5. O Administrador clica em Gravar.6. Os dados serão salvos no banco de dados.
Fluxo Alternativo	(5-6). Os campos são todos obrigatórios, caso algum esteja faltando deverá emitir uma mensagem.

Quadro 4.3 - Caso de uso completo

Fonte: Elaborado pelo autor.

#PraCegoVer: Esse quadro tem um caso de uso completo onde tem o nome do caso de uso denominado Manter Aluno. Tem a descrição do Nome do caso de uso, Qual é o ator principal, Quais são os interessados e interesses. Qual as pré-condições e pós-condições, qual o cenário principal e qual é o fluxo de alternativo. Primeira linha: caso de uso alugar o livro; ator principal: cadastrar aluno; interessados e interesses: administrador usuário responsável por realizar os cadastros do sistema. Deverá utilizar informações dos alunos para alimentar o sistema; Pré-condições: o administrador deverá estar logado no sistema; Pós-condições: os dados serão armazenados. o usuário será direcionado para a tela de inserir aluno em uma disciplina; Cenário Principal: 1. O administrador obtém os dados do aluno; 2. O administrador clica em menu alunos cadastrar novo aluno; 3. O administrador solicita os documentos para o aluno; 4. O administrador insere os dados: nome do aluno, cpf, rg, data de nascimento, endereço completo e telefone para contato; 5. O administrador clica em gravar; 6. Os dados serão salvos no banco de dados. Fluxo alternativo: 5 ou 6 os campos são todos obrigatórios caso algo esteja faltando deverá emitir uma mensagem.

Deste modo, o Quadro 4.3 demonstrou o caso de uso completo.

praticar

Vamos Praticar

Seu Ari é proprietário de uma pequena imobiliária de casas de temporada e procurou você para desenvolver um sistema auxiliar para o seu sistema, que, na

opinião dele, é pequeno. No entanto, como será um trabalho que você considera pequeno, você será o único membro da equipe.

Do ponto de vista da engenharia de *software*, selecione a alternativa CORRETA.

- a)** Como é um *software* grande, não é necessário realizar a documentação.
- b)** Como é um *software* pequeno, não é necessário realizar a documentação.
- c)** Como é apenas uma pessoa, não é necessário realizar a documentação.
- d)** Independentemente do caso, é sempre necessário realizar a documentação.
- e)** Independentemente do caso, não é necessário realizar a documentação.

Diagrams e Padrões de Projeto

O objetivo de um projeto de *software* é justamente utilizar recursos da engenharia de *software* para garantir que o *software* esteja atendendo às necessidades, antes mesmo do seu desenvolvimento.

A etapa de projeto e desenvolvimento dos diagramas de banco de dados e diagramas relacionados ao código é considerada um dos pontos cruciais, pois o que for desenvolvido aqui será idêntico na prática.

Desse modo, por mais que se trate de uma etapa de projeto, muitas decisões estão relacionadas ao desenvolvimento do sistema. Tais decisões podem ser relacionadas a algum tipo de *framework* utilizado, a alguma arquitetura específica de um sistema gerenciador de banco de dados e até mesmo a algum padrão de projeto que poderá ser seguido.

Modelagem de Banco de Dados

Você já sabe que o projeto de um banco de dados é composto por modelo conceitual, modelo lógico e modelo físico. No entanto, muitas decisões devem

ser tomadas.

Para chegar até aqui, é essencial um documento de requisitos bem descrito, bem como um diagrama de casos de uso e suas descrições.

No desenvolvimento de um modelo de dados para o mesmo sistema acadêmico, podemos considerar algumas alternativas sobre o modelo conceitual de banco de dados, lembrando que elas irão impactar nos próximos modelos:

- Se alunos e professores têm muitos atributos similares, qual a melhor opção?
 - Criar uma única entidade-pessoa: se essa opção for considerada, todos os registros ficarão armazenados em uma única tabela no banco de dados (o que remete à economia de espaço). No entanto, haverá dificuldades para desenvolver consultas para discernir dados das duas tabelas.
 - Criar duas entidades: nessa opção, haverá uma otimização das consultas e inserções. No entanto, pode implicar redundância e aumento do espaço utilizado.
 - Criar uma entidade-pessoa e generalizar as duas: essa é uma boa opção para garantir a integridade e a baixa redundância dos dados, envolve consultas mais complexas.
- O campo endereço: o campo endereço pode ser modelado no modelo conceitual como um campo composto, mas ficarão opções para o modelo lógico:
 - Inserir todos os campos na tabela como texto: essa opção na prática permitirá com que cada usuário insira seus dados, deste modo além de um grande volume de dados a dificuldade de extrair relatórios. Por exemplo, para o mesmo estado várias pessoas podem inserir dados diferentes: Santa Catarina, SC, S. Catarina.
 - Criar tabela estado ou integrar com base dos correios: esta opção envolve a criação de tabelas e pode auxiliar na diminuição da redundância dos dados.
- O campo Telefone: sabendo que haverá muitos telefones para o usuário, no modelo conceitual basta indicar o campo como

multivvalorado, mas no modelo lógico deve haver as opções:

- Criar campos fixos na tabela: é uma opção que pode limitar a inserção como telefone1, telefone2 etc. Nessa opção, torna-se limitativo e pode desperdiçar espaço de armazenamento.
- Criar uma Tabela-Telefone: essa opção permite um armazenamento dinâmico. No entanto, só faz sentido se o telefone puder pertencer a várias pessoas.
- Criar um campo Telefones: essa opção permite armazenar vários telefones em um único campo, seja utilizando um separador como ";" ou recursos do SGBD.

Avaliando as alternativas acima, juntamente com o objetivo da unidade, foi criado o modelo conceitual a seguir:

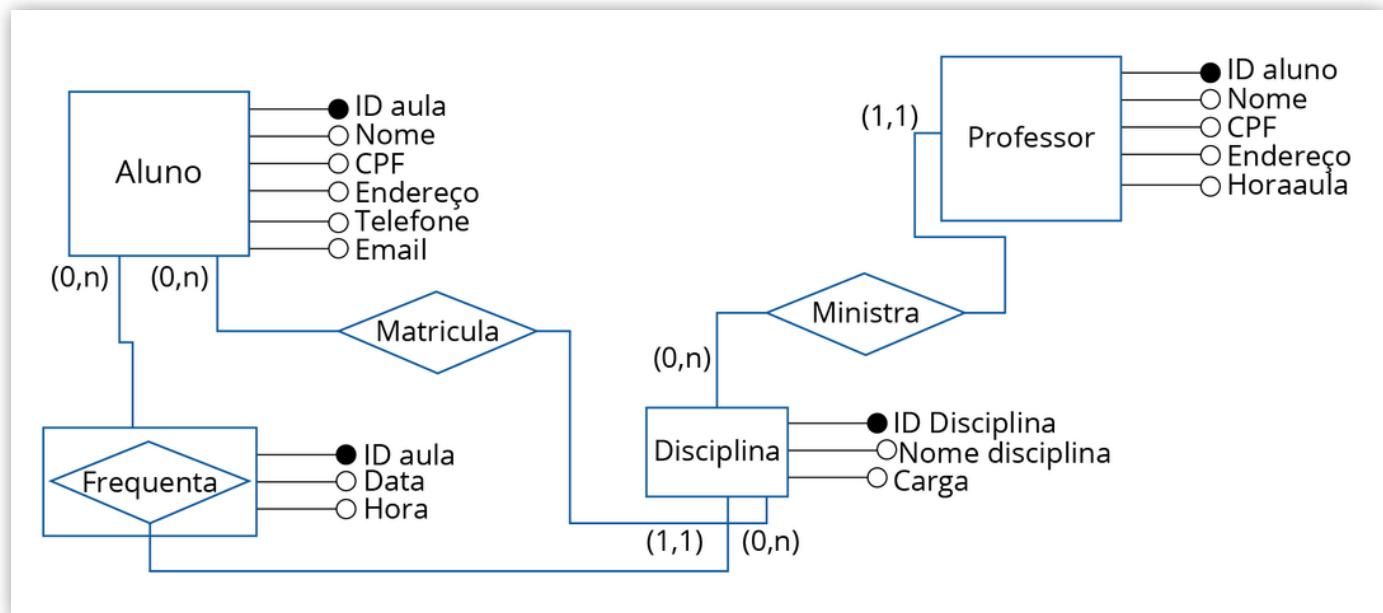


Figura 4.2 - Modelo Conceitual – Diagrama Entidade-Relacionamento

Fonte: Elaborada pelo autor.

#PraCegoVer: Figura com a representação de um diagrama E-R (Entidade Relacionamento). Onde tem as Entidades: Aluno, Professor e Disciplina. Tem como um relacionamento com campos entre aluno e disciplina com um relacionamento chamado de frequência. Tem também, um relacionamento entre aluno e disciplina com nome do relacionamento de matrícula. Tem um relacionamento entre professor e disciplina com o nome do relacionamento de ministra.

A Entidade aluno tem os Campos: identificação da aula, nome, CPF, endereço, telefone e e-mail. A Entidade Professor tem os Campos: identificação do aluno, nome, CPF endereço e hora aula. A entidade Disciplina tem os Campos: identificação da disciplina, nome da disciplina e carga. O relacionamento com o campus chamado Frequentia tem os Campos: identificação da aula, data e hora.

Tendo como base o modelo conceitual desenvolvido e levando em consideração os aspectos para criação dos modelos de dados, foi desenvolvido o modelo de dados mostrado na Figura 4.3.

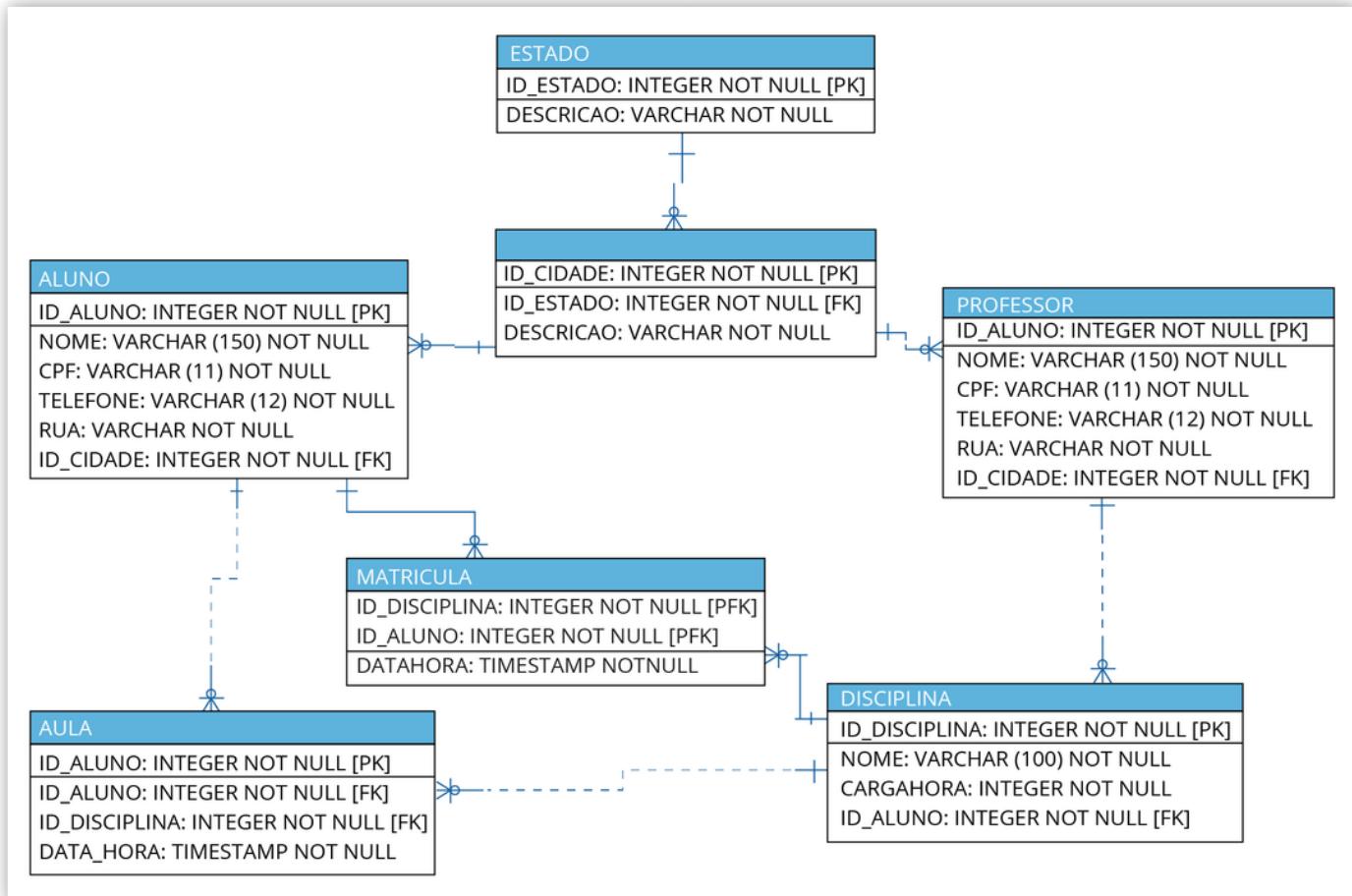


Figura 4.3 - Caso de uso completo

Fonte: Elaborada pelo autor.

#PraCegoVer: Tem-se um caso de uso completo. Tem-se um modelo com as entidades: Estado, Aluno, Professor, Aula, Disciplina e Matrícula. Em cada uma das entidades tem seus campos com suas características.

Descrição da relação entre as tabelas: O Estado pode ter vinculado a ele várias

Cidades; mas uma Cidade pode estar vinculado a somente um estado. Um Aluno pode estar vinculado somente a uma Cidade mas uma cidade pode ter vários alunos vinculados. Um Professor pode ministrar pode morar em uma cidade uma cidade pode ter vários professores o professor pode ministrar várias disciplinas, mas uma disciplina pode estar vinculada a somente um professor. O Aluno faz matrícula em várias disciplinas, mas cada disciplina-aluno pode ter somente um aluno. A matrícula ele tem somente uma disciplina-aluno, mas a Disciplina-aluno pode estar vinculada a várias Matrículas. O Aluno pode estar vinculado à várias salas de aula, mas cada Sala de Aula pode estar vinculada a um aluno.

Perceba que, no modelo desenvolvido, surgiram novas entidades, como é o caso de Cidade e Professor. A decisão de utilizar essas entidades em vez de campos nas tabelas facilitará a criação de combos para que os usuários selecionem de maneira automática, bem como auxiliará a geração de relatórios.

Diagrama de Classes

No desenvolvimento de um projeto de sistemas utilizando diversas metodologias, bem como o desenvolvimento de diagramas UML, cada diagrama tem a sua finalidade. O diagrama de classes mostra as classes de objetos em um sistema e seus relacionamentos.

O diagrama de classes permite desenvolver um projeto de software orientado a objetos, antes mesmo de codificar.

Um diagrama de classes exibe um conjunto de classes, interfaces e colaborações, bem como seus relacionamentos. Esses diagramas são encontrados com maior frequência em sistemas de modelagem orientados a objeto e abrangem uma visão estática da estrutura do sistema. Os diagramas de classes que incluem classes ativas direcionam a perspectiva do processo estático do sistema. Os diagramas de componentes são variantes dos diagramas de classe (SOMMERVILLE, 2011, p. 64).

Para o desenvolvimento do diagrama de classes, foram considerados todos os diagramas desenvolvidos até o momento. Os diagramas de casos de uso, bem como o documento de requisitos e os casos de uso descritivo foram utilizados para compreender a funcionalidade em geral.

Uma vez que o diagrama de banco de dados já foi desenvolvido, auxiliará na compreensão dos principais dados utilizados e no comportamento e relacionamento entre as informações. A Figura 4.4 mostra o diagrama desenvolvido para o nosso projeto.

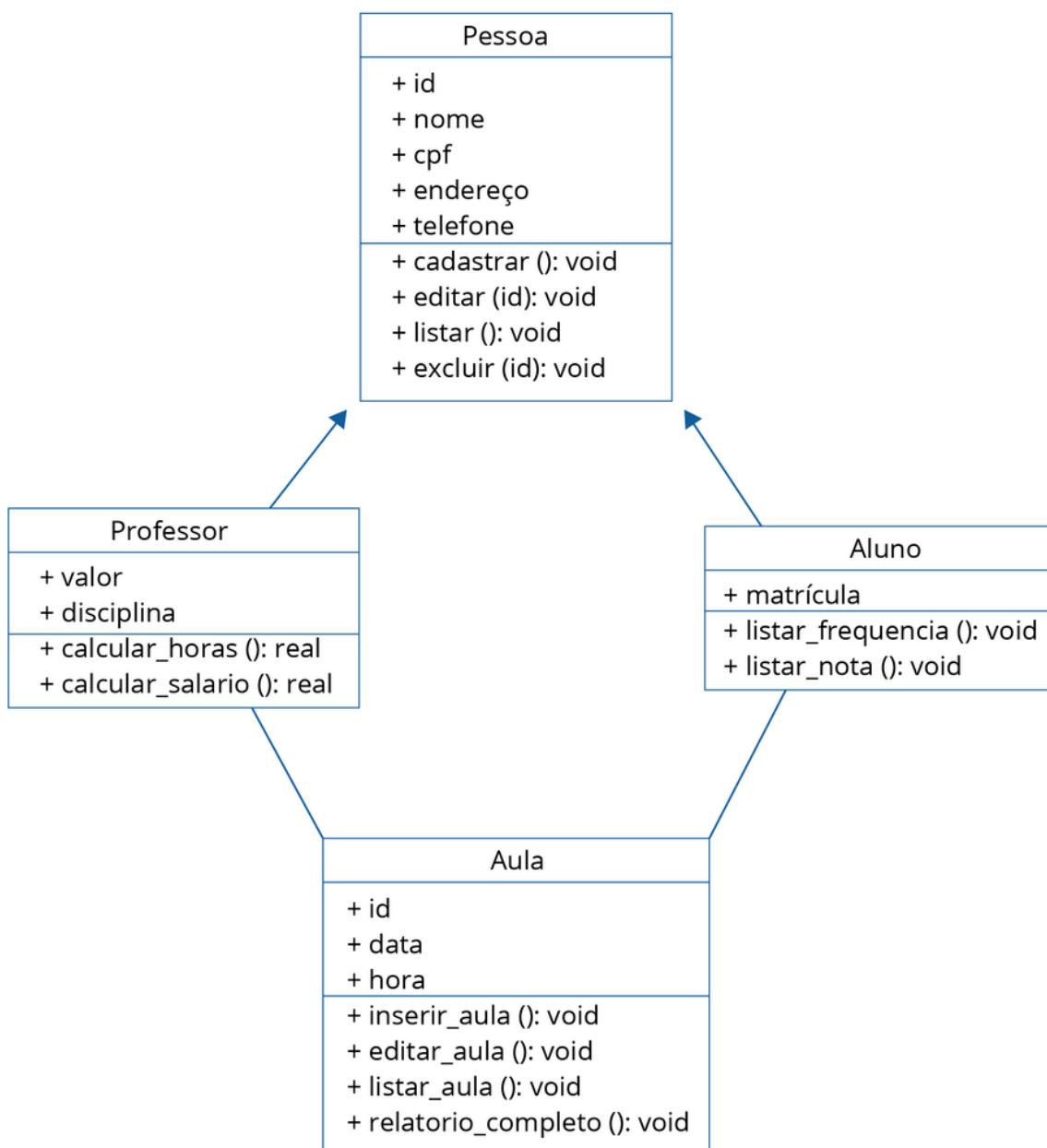


Figura 4.4 - Diagrama de classes

Fonte: Elaborada pelo autor.

#PraCegoVer: Diagrama de classes desenvolvido onde se tem demonstrado: a Pessoa, o Professor, o Aluno e a Aula. Em cada uma destas os seus respectivos Campos identificados. Principais campos de cada um: Pessoa tem identificação, nome, cpf, endereço e telefone; Professor tem: Valor e disciplina; Aula tem identificador, data e hora e no Aluno tem matrícula.

Importante: Deve-se tomar muito cuidado ao desenvolver o diagrama de

classes, tendo o diagrama de banco de dados em mãos. Nem sempre cada tabela terá sua própria classe.

Saiba mais

Padrões de Projeto e MVC

Padrões de projeto são estabelecidos por nome, problema, solução e consequências. O nome deve ser responsável por descrever o problema, a solução e as consequências.

Quando alguém na equipe de um projeto se refere a um padrão pelo nome, os demais membros da equipe devem relacioná-lo com um problema encontrado, a solução e as consequências na utilização de tal padrão.

Encontrar um nome para um novo padrão é considerado uma etapa difícil, já que o nome deve proporcionar a ideia para a qual o padrão foi criado.

Fonte: Introdução... (2020).

ACESSAR

Note que o diagrama de classes utiliza o recurso de herança para realizar os cadastros de Professor e Aluno.

Vamos Praticar

Você assumiu uma equipe de projeto de *software* que recebeu um grande projeto da indústria alimentícia. O projeto está em véspera de sua implementação, que utiliza uma linguagem orientada a objetos. No desenvolvimento do projeto, foram utilizados o diagrama de classes, o diagrama entidade-relacionamento e o diagrama de casos de uso.

Sobre o desenvolvimento de um projeto de *software*, selecione a alternativa correta com a sequência dos diagramas mencionados.

- a)** Diagrama de classes >> Diagrama entidade-relacionamento >> Diagrama de casos de uso.
- b)** Diagrama de casos de uso >> Diagrama de classes >> Diagrama entidade-relacionamento.
- c)** Diagrama entidade-relacionamento >> Diagrama de casos de uso >> Diagrama de classes.
- d)** Diagrama de casos de uso >> Diagrama entidade-relacionamento >> Diagrama de classes.
- e)** Diagrama de classes >> Diagrama de casos de uso >> Diagrama entidade-relacionamento.

Desenvolvimento

O processo de desenvolvimento de *software* é considerado como a hora de “colocar as mãos na massa”. Embora você tenha aprendido até aqui que há muito trabalho até chegar nesta etapa, é na hora do desenvolvimento de um sistema que a codificação acontece e os programadores entram em cena.

Segundo Montoni (2010), o desenvolvimento de *software* é uma atividade complexa e processos de *software* dependem fortemente do comprometimento humano para sua implementação. Desse modo, conforme você já estudou, existe um conjunto de processos e metodologias que impactam diretamente nessa etapa, antes mesmo dela ser inicializada.

Esta é etapa de desenvolvimento prático de um *software*, ou seja, a codificação deve ser inicializada após ter os documentos em mãos. Isso significa que todo o processo de engenharia de *software* e gestão de projetos foi executado e os programadores receberam os *sprints* com suas tarefas.

Assumindo que estamos iniciando um projeto do zero, existem diversas maneiras de se ter um ponto de partida para o desenvolvimento. No entanto, vamos começar preparando o ambiente de criação de um banco de dados.

Criando o Banco de Dados

Se você seguiu as etapas de modelagem de dados criando o modelo lógico e físico, utilizando as mesmas ferramentas, BrModelo e Power Architect, esta etapa pode ser uma etapa simples. Tais ferramentas simplificam a transformação dos modelos de mais alto nível em um modelo de baixo nível, ou seja, geram SQL.

O Power Architect, bem como outras ferramentas de modelagem, permite que você se conecte diretamente com o sistema gerenciador de banco de dados, auxiliando no processo de manutenção desses dados.

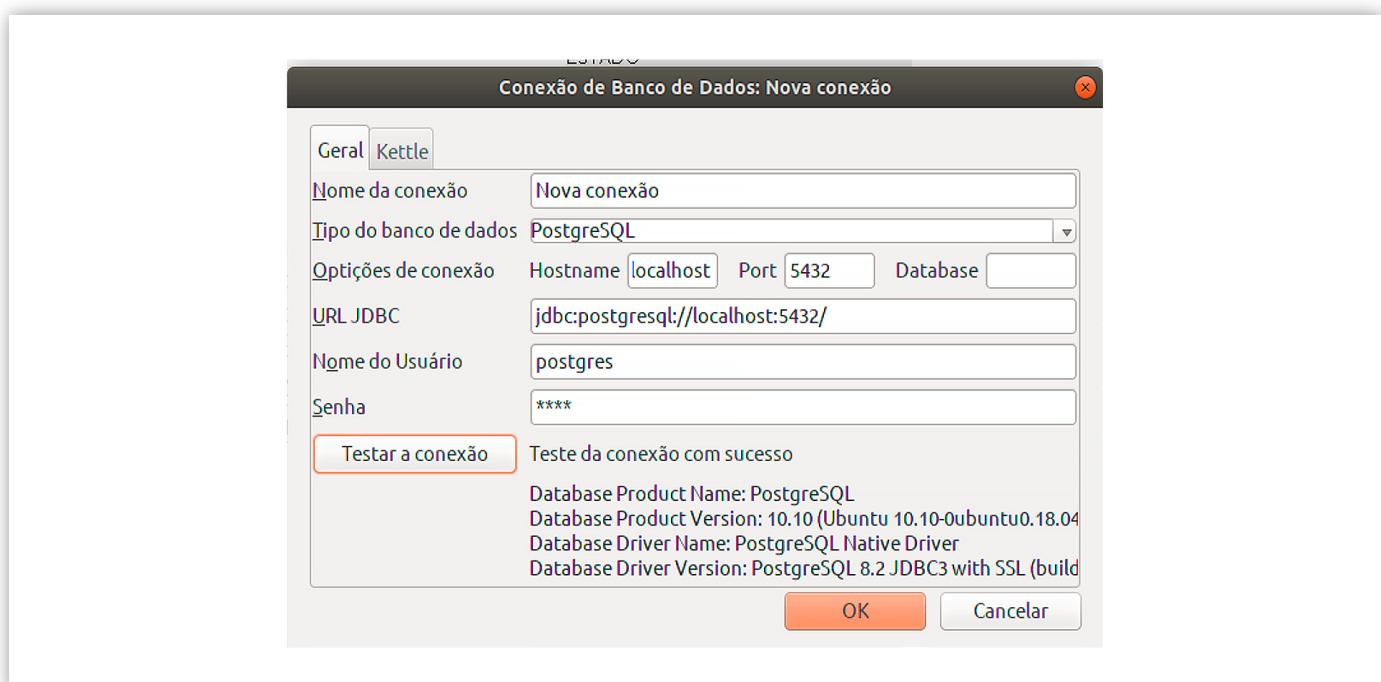


Figura 4.5 - Conectando o Power Archtech ao SGBD

Fonte: Elaborada pelo autor.

#PraCegoVer: Figura tendo bem em cima em destaque o título: Conexão de banco de dados - Nova conexão. Abaixo um Template com os campos: Nome da conexão, Tipo do banco de dados, Opções de conexão, URL JDBC, Nome do usuário e senha. Em um retângulo laranja na parte inferior a esquerda o botão Testar a conexão. Logo abaixo em outro retângulo a opção OK e a Cancelar.

Outra abordagem para criar o seu banco de dados é digitar os comandos SQL.

Você pode fazer isso apenas utilizando o modelo e fazendo à mão. No entanto, para auxiliar no processo, pode utilizar o Power Architect no menu Ferramentas >> Engenharia Reversa. Feito isso, selecione o SGBD de destino, no caso o PostgreSQL, e aperte OK. Depois, a ferramenta irá gerar a DDL para criação da estrutura do banco de dados.

The screenshot shows a window titled "Visualização script SQL". A message at the top says "Your Target Database is not configured.". Below the message is a scrollable text area containing the following SQL code:

```
CREATE TABLE ESTADO (
    ID_ESTADO INTEGER NOT NULL,
    DESCRICAO VARCHAR NOT NULL,
    CONSTRAINT estado_pk PRIMARY KEY (ID_ESTADO)
);

CREATE TABLE CIDADE (
    ID_CIDADE INTEGER NOT NULL,
    ID_ESTADO INTEGER NOT NULL,
    DESCRICAO VARCHAR NOT NULL,
    CONSTRAINT cidade_pk PRIMARY KEY (ID_CIDADE)
);

CREATE TABLE PROFESSOR (
    ID_ALUNO INTEGER NOT NULL,
    NOME VARCHAR(150) NOT NULL,
    CPF VARCHAR(11) NOT NULL,
    TELEFONE VARCHAR(12) NOT NULL,
    RUA VARCHAR NOT NULL,
    ID_CIDADE INTEGER NOT NULL,
    CONSTRAINT professor_pk PRIMARY KEY (ID_ALUNO)
);

CREATE TABLE DISCIPLINA (
    ID_DISCIPLINA INTEGER NOT NULL
);
```

At the bottom of the window are four buttons: "Copiar", "Executar", "Salvar", and "Fechar".

Figura 4.6 - Engenharia reversa com o Power Architect

Fonte: Elaborada pelo autor.

#PraCegoVer: Figura tem como título principal (com fundo preto): Visualização de um script SQL para fazer uma engenharia reversa com Power Architect. Tem escrito um script para a criação da tabela estado, cidade, professor e disciplina. Com seus respectivos campos. Criação de uma tabela chamada Estado com os campos de identificação do Estado inteiro não nulo, campo descrição varchar não nulo onde tem uma chave primária da identificação do Estado. Criação de uma tabela chamada cidade que tem seus campos: Identificação da cidade inteiro não nulo, Identificação do Estado inteiro não nulo, Descrição varchar não nulo e como chave primária código da cidade. Criação da tabela Professor com os campos: Identificação do aluno inteiro não nulo, varchar com 150 posições não nulo, CPF varchar com tamanho 11 não nulo, Telefone varchar com tamanho 12 não nulo, Rua varchar não nulo, Identificação da cidade inteiro não nulo e a chave primária é

a identificação do Aluno.

Uma vez obtido o código DDL para a criação das tabelas, basta executá-lo na interface do SGBD. Em nosso caso, foi executado utilizando o PgAdmin, que é uma ferramenta visual para conectar e utilizar o PostgreSQL.

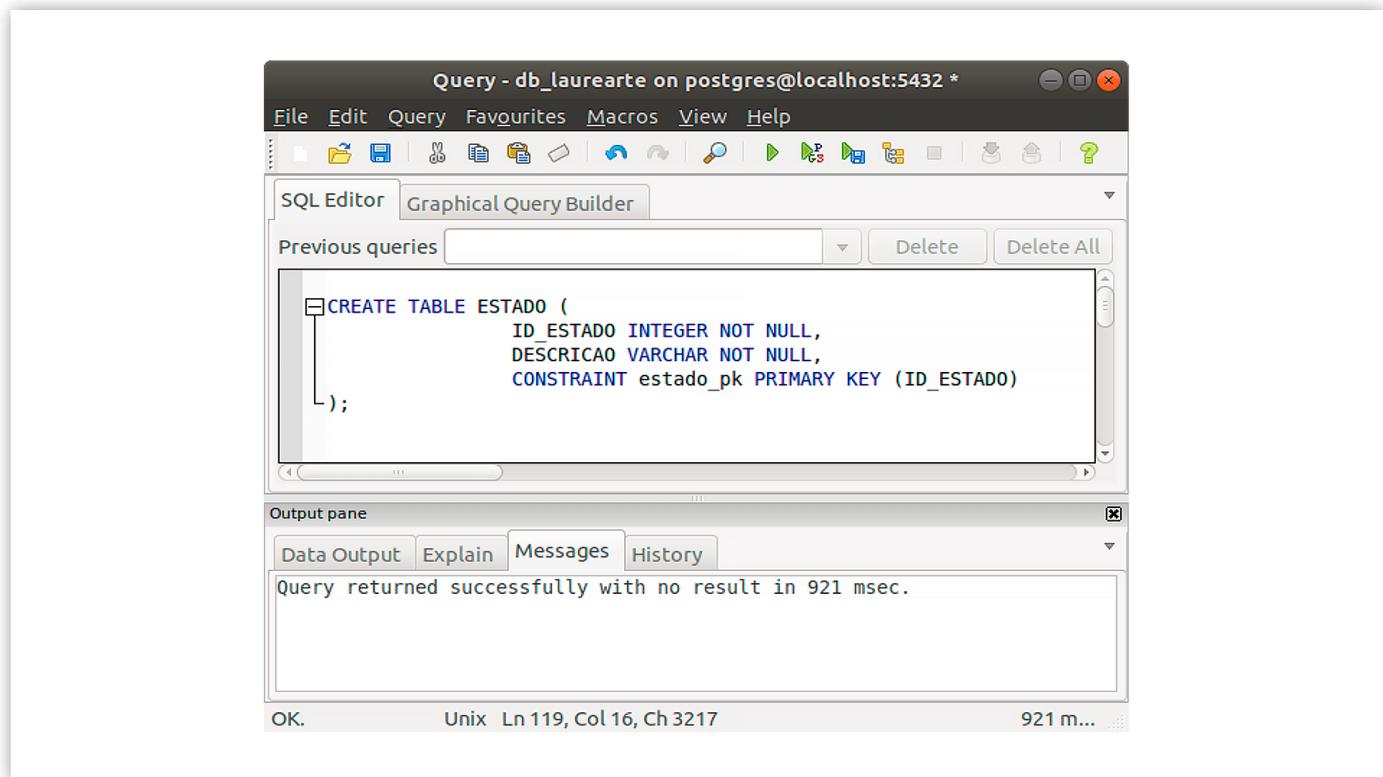


Figura 4.7 - Executando a DDL

Fonte: Elaborada pelo autor.

#PraCegoVer: Na figura aparece um framework de um editor de SQL. Neste editor consta o comando SQL para a criação da tabela Estado e seus campos identificação do estado, descrição do estado e a sua chave primaria. O script é a criação da tabela Estado com os campos Identificação do Estado inteiro não nulo, Descrição varchar não nulo e a chave primária é a Identificação do Estado.

Criando as Telas do Sistema

Os protótipos de tela são desenvolvidos tendo como finalidade dar suporte à

versão final. Nesse caso, vamos desenvolver a tela em si utilizando uma tecnologia específica.

Para isso, utilizaremos a linguagem de marcação HTML (*Hypertext Markup Language* - linguagem de marcação de texto) para a criação das telas, e o CSS (*Cascading Style Sheets* - folha de estilos em cascata) para dar estilo às telas.

Essas duas tecnologias são comumente empregadas em conjunto e em um padrão MVC, compõem a *View*, ou seja, a camada de visualização. Entre vários benefícios, o desenvolvimento com essas tecnologias faz com que a parte operacional (*back-end*) possa ser desenvolvida em qualquer linguagem que opere na *web*.

```
<div class="container">
    <form id="cad_aluno" action="" method="post">
        <h3>Cadastro de Alunos</h3>
        <fieldset>
            <input placeholder="Nome" type="text" tabindex="1" required
autofocus>
        </fieldset>
        <fieldset>
            <input placeholder="CPF" type="email" tabindex="2" required>
        </fieldset>
        <fieldset>
            <input placeholder="Telefone" type="tel" tabindex="3"
required>
        </fieldset>
        <fieldset>
            <input placeholder="email" type="email" tabindex="4" required>
        </fieldset>
        <fieldset>
            Estado:
            <select>
                <option value="-1">--SELEÇÃO--</option>
                <option value="1">--SC--</option>
                <option value="2">--SP--</option>
                <!--A carregar por Ajax-->
            </select>
        </fieldset>
        <fieldset>
            Cidade:
            <select>
                <option value="1">--SELEÇÃO--</option>
                <!--A carregar por Ajax quando selecionar o estado-->
            </select>
        </fieldset>
        <button name="submit" type="submit" id="enviar" data-
submit="...Enviando">Enviar</button>
        <button name="submit" type="submit" id="cancela" data-
submit="...Cancelando">Cancelar</button>
    </form>
</div>
```

Quadro 4.4 - Criando o código

Fonte: Elaborado pelo autor.

#PraCegoVer: Foto da tela do computador com o fundo todo preto. Nesta tela

tem um programa de computador escrito. Este programa vai criar uma classe container. A identificação do um formulário é identificado como cadastro do aluno. Imprime na tela “Cadastro do Aluno” e faz o input (entrada) do nome, input do CPF, input do telefone e e-mail. Para o campo Estado tem com opção de seleção (opção) 1 - SP e 2- SP. Depois o texto “Selecionar”. Depois do input colocar o submit “enviando” ou “cancelando”.

A partir do código acima e sua combinação com CSS, é possível ter a tela do sistema, que é mostrada a seguir. O CSS utilizado pode ser obtido pelo site “codepen”.

The screenshot shows a registration form titled "Cadastro de Alunos". The form is contained within a green-bordered box. Inside, there are four input fields for "Nome", "CPF", "Telefone", and "email". Below these are two dropdown menus for "Estado" and "Cidade", both currently set to "--SELECIONE--". At the bottom of the form are two large green buttons: "CADASTRAR" and "Cancelar".

Figura 4.8 - Tela de Cadastro de Aluno

Fonte: Elaborada pelo autor.

#PraCegoVer: Uma tela de cadastro aluno aonde a borda toda tela é uma borda Verde tem escrito bem em cima, em destaque, Cadastro de Alunos. Abaixo os campos para preencher: O nome do aluno, o CPF, o telefone e o e-mail. Abaixo a opção para selecionar o estado se a cidade. Logo abaixo dois retângulos verdes. No primeiro retângulo verde a opção cadastrar e no outro a opção cancelar.

Codificando o Sistema

Agora você tem praticamente tudo em mãos: os documentos de requisitos, diagramas de caso de uso, descrição dos casos, diagramas de banco de dados e diagrama de classes. Com isso, pode-se dizer que está pronto para codificar.

Saiba mais

Para você compreender um pouco mais sobre a integração da etapa de projeto de *software* com a etapa de codificação, trazemos esta leitura complementar. O objetivo do estudo é realizar, na prática e de forma didática, a construção de um caso de uso que faz parte de um sistema de informação, utilizando a linguagem PHP com padrão de projeto MVC. O texto apresenta o desenvolvimento de uma aplicação pequena; no entanto, completa no que se refere ao ciclo de vida de um *software*, principalmente no que diz respeito à implementação.

Fonte: Rocha (2017).

Nesta etapa, com um projeto de *software* bem definido, a programação torna-se um processo simples, pois grande parte do desenvolvimento de sistemas é gasto na compreensão do que deve ser desenvolvido, bem como em alterações que precisam ser feitas porque a documentação foi construída erroneamente ou até mesmo não foi realizada.

A UML nos fornece o diagrama de classe e nele não definimos uma linguagem

de programação.

O PHP é uma das linguagens mais utilizadas na web. Milhões de sites no mundo inteiro utilizam PHP. A principal diferença em relação às outras linguagens é a capacidade que o PHP tem de interagir com o mundo web, transformando totalmente os websites que possuem páginas estáticas. Imagine, por exemplo, um website que deseja exibir notícias em sua página principal, mostrando a cada dia, ou a cada hora, notícias diferentes. Seria inviável fazer isso utilizando apenas HTML. As páginas seriam estáticas, e a cada notícia nova que aparecesse no site a página deveria ser alterada manualmente, e logo após enviada ao servidor por FTP (File Transfer Protocol) para que as novas notícias fossem mostradas no site. Com o PHP, tudo isso poderia ser feito automaticamente. Bastaria criar um banco de dados onde ficariam armazenadas as notícias e criar uma página que mostrasse essas notícias, "puxando-as" do banco de dados (NIEDERAUER, 2011, p. 2).

Em um primeiro exemplo, vamos fazer o código orientado a objetos da classe Pessoa necessário para executar o exemplo desenvolvido (Cadastro Aluno) utilizando a linguagem PHP.

```
<?php

class Pessoa{

    public $nome;
    public $cpf;
    public $endereco;
    public $telefone;

    function __construct(){
        $conexao = pg_connect("localhost","postgres","");
        if($conexao == false)
            die("Erro na conexão com o banco de dados");
        $banco = pg_select_db("oo");
    }
    function __set($var, $val){
        $this->$var = $val;
    }
    function cadastrar($tabela){

        $inserirPessoa = pg_query("insert into $tabela values('$this->nome', '$this->cpf', '$this->endereco', '$this->telefone')");
        if($inserirPessoa){
            $retorno="Ocorreu tudo bem ao inserir no sistema";
        }else{
            $retorno ="Falha ao realizar inserção na tabela $tabela";
        }

        return $retorno;
    }
}

?>
```

Quadro 4.5 - Implementando uma classe em PHP

Fonte: Elaborado pelo autor.

#PraCegoVer: Foto da tela do computador com o fundo todo preto. Nesta tela tem um programa de computador escrito. Este programa cria a classe como o nome Pessoa que tem como campos: nome, CPF, endereço e telefone. Tem uma função para fazer a conexão entre o localhost sendo o postgres. Seta a @banco = pg_select_db. Abre uma função com nome de Cadastrar na tabela. Insere o nome, cpf, endereço e telefone. Tem um if (teste) na inserção da pessoa . Se sim coloca a mensagem “ocorreu tudfo bem ao inbserir nos sitea” ou e não (else) “Falha ao realizsar inserção na tabela”.

Exemplificando como um mesmo modelo pode dar origens a várias implementações, bem como a mesma tela, agora será desenvolvido um código orientado a objetos da classe Pessoa necessário, utilizando a

linguagem Python.

Python é software livre, ou seja, pode ser utilizada gratuitamente, graças ao trabalho da Python Foundation¹ e de inúmeros colaboradores. Você pode utilizar Python em praticamente qualquer arquitetura de computadores ou sistema operacional, como Linux², FreeBSD³, Microsoft Windows ou Mac OS X⁴. Python vem crescendo em várias áreas da computação, como inteligência artificial, banco de dados, biotecnologia, animação 3D, aplicativos móveis (celulares), jogos e mesmo como plataforma web. Isso explica porque Python é famosa por ter “batteries included”, ou seja, baterias inclusas, fazendo referência a um produto completo que pode ser usado prontamente (quem nunca ganhou um presente de Natal que veio sem pilhas?). Hoje é difícil encontrar uma biblioteca que não tenha bindings (versão) em Python (MENEZES, 2010, p. 22).

O fato de existirem diversas bibliotecas para os mais diversos tipos de problemas em Python faz com que a linguagem seja uma das mais utilizadas no mundo. Devido ao número de bibliotecas, mesmo profissionais de outras áreas, como matemática e estatística, optam pelo emprego dessa linguagem.

```
class Pessoa:
    def __init__(self, id_pessoa, nome, cpf, endereco, telefone):
        self._id_pessoa = id_pessoa
        self._nome = nome
        self._cpf = cpf
        self._endereco = endereco
        self._telefone = telefone

    @property
    def indice(self):
        return self._indice
    @property
    def email(self):
        return self._email
    @property
    def senha(self):
        return self._senha

    @email.setter
    def email(self, email):
        self._email = email
    @senha.setter
    def senha(self, senha):
        self._senha = senha

    try:
        self.con =
            psycopg2.connect(user="user", password="1234", host="127.0.0.1",
                             port="5432",
                             database="postgres")
    except psycopg2.connector.Error as erro:
        print(erro)

    def cadastrar(self, pessoa):
        try:
            cursor = self.con.cursor()
            comando = ("INSERT INTO pessoa (nome, cpf, endereco,
            telefone) VALUES (%s, %s, %s, %s)")
            valores = (pessoa.nome, pessoa.cpf, pessoa.endereco,
            pessoa.telefone)
            cursor.execute(comando, valores)
            self.con.commit()
```

Quadro 4.6 - Implementando classe em Python

Fonte: Elaborado pelo autor.

#PraCegoVer: Foto da tela do computador com o fundo todo preto. Nesta tela tem um programa de computador escrito.

O programa cria a classe Pessoa definindo a identificação da pessoa, o nome da pessoa, CPF, endereço e telefone. Faz um self para cada um destes campos.

Faz um property na def do índice, do e-mail e da senha.

Por fim faz um try o e-mail e a senha faz 13 aonde testa se o usuário for igual e o pessoal foi igual a 123 o host 127.0.0.1 na porta 5432 o banco de dado posto aonde faço o execute do conector se for um erro 30 o erro definir cadastrar onde fazer o cursor igual lugar do cursor o comando insert into

pessoa nome CPF endereço e telefone.

Saiba mais

Orientação a objetos (OO) é uma forma conceitual de estruturar um programa: em vez de definirmos variáveis e criarmos funções que as manipulam, definimos objetos que possuem dados próprios e ações associadas. O programa orientado a objetos é resultado da colaboração entre esses objetos. Em Python, todos os dados podem ser considerados objetos: qualquer variável – mesmo as dos tipos básicos e predefinidos – possui um valor e um conjunto de operações que pode ser realizado sobre este. Por exemplo, toda *string* em Python possui uma operação (ou método) chamada *upper* , que gera uma *string* nova com o seu conteúdo em maiúsculas:

```
>>> a = "Hello"  
>>> a.upper()  
'HELLO'
```

Para saber mais, acesse o link a seguir.

[ACESSAR](#)

Durante a leitura até este ponto, você pode perceber que tanto Python quanto PHP são poderosas linguagens para desenvolver os mais diversos tipos de

aplicações. Além disso, é muito importante compreender que a modelagem de sistemas orientada objetos, por intermédio do diagrama de classes, não tem o seu projeto de desenvolvimento ligado obrigatoriamente a uma linguagem de programação.

praticar

Vamos Praticar

Como analista de um projeto de *software*, você seguiu todos os passos necessários para documentar e planejar um sistema para uma fábrica de chocolates. Entre todos os diagramas necessários, também desenvolveu o diagrama de classes. Sobre o diagrama de classes e sua implementação de código, analise as afirmativas a seguir.

- I) A implementação poderá ser feita em Python.
- II) A implementação poderá ser feita em PHP.
- III) A implementação poderá ser feita em Java.
- IV) A implementação poderá ser feita em C++.

Está correto o que se afirma em:

- a) I, apenas.
- b) II, apenas.
- c) III, apenas.
- d) IV, apenas.
- e) I, II, III e IV.



Finalizando um Projeto de Software: Teste e Validação

Existem muitas perguntas complexas para se responder durante o processo de desenvolvimento de um sistema de informação. Talvez algumas que deem calafrios em desenvolvedores e analistas sejam: o sistema está pronto? Ele está funcionando?

Isso acontece porque cada membro da equipe tem sua definição/expectativa do que é um *software* pronto e seu funcionamento, e o cliente também tem a dele. Por isso, é necessário o desenvolvimento de uma documentação precisa, a cada etapa do desenvolvimento de um *software*, que auxiliará no processo de teste e validação.

O processo de desenvolvimento de software envolve uma série de atividades nas quais, apesar das técnicas, métodos e ferramentas empregados, erros no produto ainda podem ocorrer. Atividades agregadas sob o nome de Garantia de Qualidade de Software têm sido introduzidas ao longo de todo o processo de desenvolvimento,

entre elas atividades de VV&T – Verificação, Validação e Teste, com o objetivo de minimizar a ocorrência de erros e riscos associados. Dentre as técnicas de verificação e validação, a atividade de teste é uma das mais utilizadas, constituindo-se em um dos elementos para fornecer evidências da confiabilidade do software em complemento a outras atividades, como por exemplo o uso de revisões e de técnicas formais e rigorosas de especificação e de verificação (MALDONADO et al., 2004 p. 3).

O processo de teste pode ser composto por etapas e por tipos de testes realizados, tendo o objetivo de garantir que o sistema atenda aos requisitos, bem como indicar que não foram encontrados erros.

No que se refere às etapas de teste, podem-se elencar as seguintes:

- Teste de desenvolvimento: é o teste realizado durante o desenvolvimento do sistema com o objetivo de encontrar bugs e erros. Geralmente essa etapa é realizada pelo próprio programador responsável pela funcionalidade.
- Teste de *release* : é realizado por uma equipe independente, tem como objetivo validar se o sistema atende aos requisitos levantados, bem como procurar erros no sistema.
- Teste de usuário: o teste de usuário é um teste feito por “usuários de sistema”, ou seja, feito por pessoas que não são da área e que têm o perfil do usuário que irá utilizar o sistema.
- Teste de aceitação: é o teste realizado com o usuário final do sistema. Pode-se dizer que é o teste realizado pelo cliente e membros da sua empresa, indicando se o sistema atende à sua necessidade e será aceito.



Saiba mais

Durante o projeto e aplicação de testes de *software*, os termos falha, erro e defeito são muito comuns. Por isso, compreendê-los pode auxiliar a realizar testes mais eficazes.

O defeito é qualquer imperfeição ou inconsistência no produto do *software* ou em seu processo. Um defeito é também uma não conformidade. O erro pode ser o resultado de um defeito ou uma falha, como um retorno esperado, que, por causa de uma falha, teve um valor diferente do esperado. A falha está mais ligada ao *hardware*, como uma rede inacessível, queda de energia. Uma falha pode ocorrer por causa de um erro. Por exemplo, houve um retorno de um valor não esperado, como *null*. Isso é um erro e por causa desse *null* uma falha foi ocasionada no sistema.

Fonte: Silva (2020).

Plano de Testes

O plano de testes é mais um documento que compõe a documentação de *software* e pode ser desenvolvido logo após o desenvolvimento dos casos de uso. Esse documento tem como objetivo guiar a equipe de *software* em relação a como os testes serão realizados.

O principal objetivo desta atividade é planejar os esforços de teste dentro de uma iteração, descrevendo uma estratégia de testes, estimando os requisitos para o esforço de teste, tais como recursos humanos e do sistema, e criando uma escala de testes a ser executada. Assim, constrói-se um plano de testes, com base nos diversos modelos utilizados nas fases de especificação, análise, design e implementação. Assim, desenvolve-se aqui uma estratégia geral de testes, onde determina-se que tipos de testes serão executados, como executá-los, quando executá-los e como avaliar os resultados dos testes. Devemos lembrar que testes custam recursos e tempo. Portanto, deve-se efetuar uma solução de testes que tenha uma boa relação custo-benefício (GUDWIN, 2020 p. 143).

De todos os elementos que um plano de testes contém, pode-se dizer que o mais importante é o caso de testes. Um caso de teste descreve uma rotina que deve ser testada.

Caso N º	CT010 – Realizar o cadastro de aluno
Objetivo do Teste	Verificar se o usuário Atendente consegue realizar o cadastro de um aluno no sistema
Usuário/Ator	Administrador
Passos	<ol style="list-style-type: none">1. O usuário clica em Menu Alunos >> Cadastrar Novo Aluno2. O usuário clica em gravar3. O usuário insere Nome do Aluno “João Silva” e clica em gravar4. O usuário insere o CPF “1111111111”5. O usuário clica em gravar6. O usuário insere RG “5458545854” e clica em gravar7. O usuário insere Data De Nascimento “25/01 /2050” e clica em gravar8. O usuário insere Data De Nascimento “25/01 /1990” e clica em gravar9. O usuário insere o endereço “Rua da Penha” e clica em gravar10. O usuário insere o telefone “2521” e clica em gravar11. O usuário insere o telefone “47952565896” e clica em gravar
Critérios de Êxito	<ol style="list-style-type: none">1. O usuário clica em Menu Alunos >> Cadastrar Novo Aluno<ol style="list-style-type: none">a. A tela de cadastro de alunos é aberta2. O usuário clica em gravar<ol style="list-style-type: none">a. Todos os campos não preenchidos e obrigatórios (*) devem ficar em vermelho3. O usuário insere Nome do Aluno “João Silva” e

clica em gravar

a. Todos os campos não preenchidos e obrigatórios

(*) devem ficar em vermelho

4. O usuário insere o CPF “1111111111”

a. Mensagem de erro “CPF Inválido”

b. Todos os campos não preenchidos e obrigatórios

(*) devem ficar em vermelho

5. O usuário clica em gravar

a. Todos os campos não preenchidos e obrigatórios

(*) devem ficar em vermelho

6. O usuário insere RG “5458545854” e clica em gravar

a. Todos os campos não preenchidos e obrigatórios

(*) devem ficar em vermelho

7. O usuário insere Data De Nascimento “25/01 /2050” e clica em gravar

a. Mensagem de erro “Data de Nascimento maior que Data Atual”

b. Todos os campos não preenchidos e obrigatórios
(*) devem ficar em vermelho

8. O usuário insere Data De Nascimento “25/01 /1990” e clica em gravar

a. Todos os campos não preenchidos e obrigatórios
(*) devem ficar em vermelho

9. O usuário insere o endereço “Rua da Penha” e clica em gravar

a. Todos os campos não preenchidos e obrigatórios
(*) devem ficar em vermelho

10. O usuário insere o telefone “2521” e clica em gravar

a. Mensagem de Erro “Telefone inválido”

b. Todos os campos não preenchidos e obrigatórios
(*) devem ficar em vermelho

11. O usuário insere o telefone “47952565896” e clica em gravar

- a. Mensagem “Aluno cadastrado com sucesso”
- b. Mensagem com opção de seleção “Deseja cadastrar outro aluno?”

Quadro 4.7 - Exemplo de caso de teste

Fonte: Elaborado pelo autor.

#PraCegoVer: O quadro de exemplo de caso de teste com duas colunas. A coluna da esquerda, com o fundo azul escuro, tem a linha com o número do caso, a próxima com o objetivo do teste, a outra com usuário ou ator, a próxima com os passos e a última com os critérios de êxito desse teste. Na coluna da direita tem a descrição de uma destas linhas.

Vamos a descrição de cada uma das linhas.

Caso número: CT010 – Realizar o cadastro de aluno. Objetivo do teste: Verificar se o usuário Atendente consegue realizar o cadastro de um aluno no sistema. Usuário/Ator: Administrador.

Passos: 1. Usuário clica em menu alunos cadastrar novo aluno. 2. Usuário clica em gravar. 3. Usuário insere nome do aluno João Silva e clica em gravar. 4. O usuário insere o CPF “111111111111”. 5. O usuário clica em gravar. 6. O usuário insere o RG “545858545854” e clica em gravar. 7. O usuário insere a data de nascimento “25/01/2020” e clica em gravar. 8. O usuário insere a data de nascimento “25 /01/1990” e clica em gravar. 9. O usuário insere o endereço “Rua da Penha” e clica em gravar. 10. O usuário insere o telefone “2521” e clique em gravar. 11. O usuário insere o telefone “479526345567” e clica em gravar.

A última linha é a de Critérios de êxito: 1. Usuário clica em menu alunos cadastrar novo aluno. A tela de cadastro de alunos é aberta. 2. Usuário clica em gravar. Todos os campos são preenchidos e obrigatórios (*) devem ficar em vermelho. 3. Usuário insere nome do aluno João Silva e clica em gravar. Todos os campos são preenchidos e obrigatórios (*) devem ficar em vermelho. 4. O usuário insere o CPF “111111111111”. a. Mensagem de erro “CPF

inválido". B. Todos os campos são preenchidos e obrigatórios (*) devem ficar em vermelho. 5. O usuário clica em gravar. Todos os campos são preenchidos e obrigatórios (*) devem ficar em vermelho. 6. O usuário insere o RG "545858545854" e clica em gravar. Todos os campos são preenchidos e obrigatórios (*) devem ficar em vermelho. 7. O usuário insere a data de nascimento "25/01/2020" e clica em gravar. a. Mensagem de erro "Data de nascimento maior que a data atual". b. Todos os campos são preenchidos e obrigatórios (*) devem ficar em vermelho. 8. O usuário insere a data de nascimento "25 /01/1990" e clica em gravar. Todos os campos são preenchidos e obrigatórios (*) devem ficar em vermelho. 9. O usuário insere o endereço "Rua da Penha" e clica em gravar. Todos os campos são preenchidos e obrigatórios (*) devem ficar em vermelho. 10. O usuário insere o telefone "2521" e clique em gravar. a. Mensagem de erro "Telefone inválido". b. Todos os campos são preenchidos e obrigatórios (*) devem ficar em vermelho. 11. O usuário insere o telefone "479526345567" e clica em gravar. Todos os campos são preenchidos e obrigatórios (*) devem ficar em vermelho. a. Mensagem "Aluno cadastrado com sucesso". b. Mensagem com opção de seleção "Deseja cadastrar outro aluno?"

O caso de teste tem grande similaridade com o caso de uso, podendo ser desenvolvido com base no diagrama de casos de uso e seu descriptivo. Porém, não se deve limitar à construção de casos de teste, deve-se descrevê-lo buscando refletir a real utilização do usuário.

praticar

Vamos Praticar

Seu Ari é proprietário de uma pequena imobiliária de casas de temporada e

procurou você para desenvolver um sistema para auxiliar o dele, que, na opinião dele, é um pequeno sistema. Tendo o objetivo de validar o sistema e obter um parecer do seu Ari sobre a aplicação desenvolvida, você gera uma versão para que ele teste o sistema.

Nesse sentido, selecione a alternativa correta sobre os tipos de teste que seu Ari realizou.

- a)** Teste de usuário e teste de aceitação.
- b)** Teste de usuário.
- c)** Teste de desenvolvimento.
- d)** Teste de *release*.
- e)** Teste de automação de usuário.

Indicações **Material Complementar**





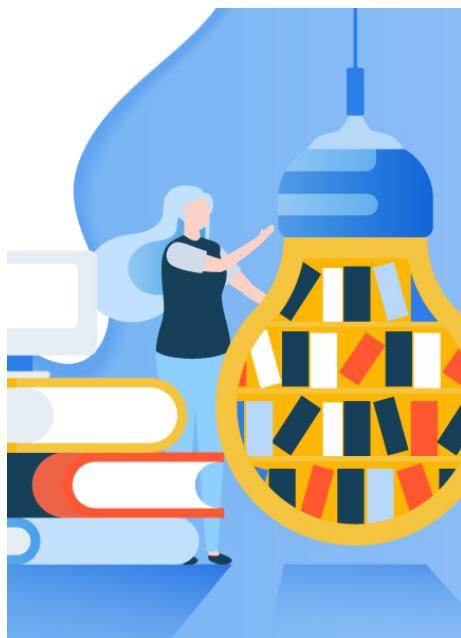
WEB

Inteligência Artificial

Ano: 2018

Comentário: A cada geração, novos paradigmas são criados, novas tecnologias surgem e, como desenvolvedores, temos que nos adaptar. Deixo um documentário exibido pela Discovery Brasil, que aborda a inteligência artificial. Você consegue imaginar o desenvolvimento de *software* aplicado a essas tecnologias?

Para conhecer mais sobre o filme, acesse o *trailer* a seguir.

[ACESSAR](#)

LIVRO

Introdução à programação com Python

Editora: Novatec

Autor: Nilo Ney Coutinho Menezes

ISBN: 8575227181, 9788575227183

Comentário: Deixo como sugestão de leitura um livro de introdução ao Python e não porque é minha linguagem favorita ou porque falamos dele durante o livro. Como você viu na nossa recomendação de vídeo, o mundo está mudando e convergindo para novas

tecnologias. O Python tem sido uma das linguagens mais usadas no mundo, tanto na programação de sistemas comerciais como em sistemas que utilizam inteligência artificial.

.....

conclusão

Conclusão

Prezado(a) aluno(a), com esta unidade, você pôde consolidar os seus conhecimentos sobre desenvolvimento de *software*. Você pôde ligar uma ponta a outra, no sentido de ver um *software* desde uma conversa com o cliente até o momento da sua validação. Durante esse processo, vale ressaltar que cada etapa tem um papel importante, feito em cadeia, e que uma falha no projeto de um *software* pode impactar diretamente na sua versão final.

No decorrer do conteúdo apresentado, vimos diversas abordagens pertencentes ao projeto de *software*. No entanto, não são as únicas. Vale destacar que são abordagens essenciais para compreender o ciclo de vida se *software*, mas que não são únicas.

A cada dia, surgem novas metodologias, abordagens, ferramentas e processos; por isso, você sempre pode buscar conhecimento, em busca de agregar valor ao seu projeto de *software* com entrega mais ágil e visando à qualidade do produto que será desenvolvido.

referências

Referências Bibliográficas

GUDWIN, R. **Engenharia de Software** : uma visão prática. Disponível em: <http://faculty.dca.fee.unicamp.br/gudwin/sites/faculty.dca.fee.unicamp.br.gudwin/files/ea97%20/ESUVP2.pdf>. Acesso em: 01 jan. 2020.

INTRODUÇÃO ao Padrão MVC. **DevMedia** . Disponível em: <https://www.devmedia.com.br/introducao-ao-padrao-mvc/29308> . Acesso em: 01 jan. 2020.

MALDONADO, J. C.; BARBOSA, E. F.; VINCENZI, A. M. R.; DELAMARO, M. E.; SOUZA, S. R. S.; JINO, M. **Introdução ao teste de software** . São Carlos: 23, 2004.

MENEZES, Nilo Ney Coutinho. **Introdução à programação com Python** . São Paulo: Novatec, 2010.

MONTONI, Mariano Angel. **Uma Investigação sobre os Fatores Críticos de Sucesso em Iniciativas de Melhoria de Processos de Software** . 2010. Tese (doutorado) UFRJ/COPPE – UFRJ/ COPPE/Programa de Engenharia de Sistemas e Computação, Rio de Janeiro.

NIEDERAUER, Juliano. **Desenvolvendo Websites com PHP** : aprenda a criar websites dinâmicos e interativos com PHP e bancos de dados. 2. ed. São Paulo: Novatec Editora, 2011.

REIS, Christian Robottom. **Python na prática** : um curso objetivo de programação em Python. Abril de 2004. Disponível em: <https://1filedownload.com/wp-content/uploads/2019/12/Basico-Python.pdf> . Acesso em: 20 dez. 2019

ROCHA, José Gladistone. Construção completa de um Caso de Uso com a utilização da UML, programação PHP e padrão MVC. **Tecnologias em Projeção** , v. 8, n. 1, p. 25-39, 2017. Disponível em: <http://revista.faculdadeprojecao.edu.br/index.php/Projecao4/article/download/798/721> . Acesso em: 22 jan. 2020.

SILVA, Fernando. **Testes de software** : entendendo defeitos, erros e falhas. Disponível em: <https://www.devmedia.com.br/testes-de-software->

[entendendo-defeitos-erros-e-falhas/22280](#). Acesso em: 01 jan. 2020.

SOMMERVILLE, Ian. **Engenharia de Software** . 9. ed. São Paulo, SP: Brasil, 2011.