Aplicando RAD

Você vai estudar a aplicação da metodologia de desenvolvimento rápido de software (RAD), enfatizando o levantamento de requisitos, a modelagem de negócios e de dados, além do design da interface com o usuário. Por meio de uma abordagem prática, utilizando a linguagem de programação Python, será possível compreender como essa metodologia acelera o desenvolvimento de sistemas, garantindo maior flexibilidade e eficiência no processo.

Prof. Kleber de Aguiar

Preparação

Antes de iniciar o conteúdo, você precisa ter instalado o Python versão 3.8.5 e a IDE Spyder.

Objetivos

- Descrever as etapas para tratamento dos requisitos de um sistema na metodologia de desenvolvimento rápido de software (RAD).
- Descrever as modelagens de negócios e de dados da RAD.
- Definir o design de interface com o usuário na RAD.
- Esquematizar uma aplicação RAD implementada em Python.

Introdução

No vídeo a seguir, confira os principais tópicos sobre metodologia RAD, tendo como foco sua aplicação prática. Vamos lá!

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Contextualização do desenvolvimento rápido de software (RAD)

O desenvolvimento rápido de software (RAD) tem se consolidado como uma abordagem para equipes que desejam otimizar processos e entregar soluções de maneira ágil e eficiente. Baseado na prototipagem rápida, colaboração intensiva e adaptação contínua, o RAD reduz as barreiras entre desenvolvedores e usuários, permitindo que o progresso seja avaliado por meio da evolução dos protótipos.

Para os profissionais de desenvolvimento, essa metodologia não só acelera o ciclo de vida do software, mas também aprimora a comunicação entre as partes interessadas, diminuindo o tempo dedicado a planejamentos rígidos e promovendo um desenvolvimento mais flexível e alinhado às demandas do negócio.

Neste vídeo, conheça o desenvolvimento rápido de software (RAD), uma abordagem ágil focada em prototipagem e colaboração contínua. Você entenderá seus princípios, benefícios e impacto na eficiência do desenvolvimento, tornando o processo mais dinâmico e adaptável.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

O desenvolvimento rápido de software (RAD) é uma abordagem iterativa e incremental, portanto, é um modelo para a construção de software caracterizado pela adaptabilidade, prototipagem rápida e colaboração das partes interessadas.

A ênfase da metodologia está na intensificação da colaboração entre as partes interessadas, em vez de estar no planejamento do que será feito. É uma forma de as equipes colaborarem com mais sinergia para desenvolverem aplicações com mais rapidez.

Comentário

O progresso do desenvolvimento da aplicação pode ser medido facilmente através da evolução do protótipo, que é a principal forma de maximizar a comunicação entre desenvolvedores e usuários sobre melhorias, ou mudanças do sistema.

Os objetivos da RAD são bem definidos. Veja!

Tempo

Maior eficiência no uso do tempo das partes envolvidas.

Comunicação

Comunicação mais produtiva.

Desenvolvimento

Processo de desenvolvimento otimizado.

A RAD faz uso de ferramentas de desenvolvimento que padronizam a implementação dos sistemas e apoiam os desenvolvedores na implementação de interfaces gráficas com o usuário, manipulação de dados, desenvolvimentos de APIs e outros serviços que poupam o tempo do desenvolvedor, para que possam usá-lo de forma mais eficiente para a implementação do sistema.

Segundo Martin (1991), a RAD possui quatro fases distintas:

Planejamento de requisitos
Trata das necessidades de negócios, escopo do projeto, restrições e requisitos do sistema.

Design do usuário
Nessa fase, são desenvolvidos modelos e protótipos para representar todos os processos, entradas e saídas do sistema.

Construção
É nesta fase que os protótipos são desenvolvidos.

Transição
Aqui, são feitos processamento de dados, testes, mudança para o novo sistema e treinamento do usuário.

Atividade 1

Qual das seguintes afirmações melhor descreve a abordagem do desenvolvimento rápido de software (RAD)?

A O RAD prioriza um planejamento detalhado e rígido antes do início do desenvolvimento, minimizando alterações durante o processo.

B O RAD é um método exclusivo para desenvolvimento de sistemas grandes e complexos, não sendo aplicável a projetos menores.

C O RAD é um modelo tradicional de desenvolvimento que segue um ciclo linear, sem interação com as partes interessadas.

D A principal característica do RAD é a completa ausência de planejamento e requisitos antes da fase de construção.

E A metodologia RAD enfatiza a prototipagem rápida e a colaboração contínua entre desenvolvedores e usuários para acelerar o desenvolvimento de software.

A alternativa E está correta.

A metodologia RAD enfatiza a prototipagem rápida e a colaboração contínua entre desenvolvedores e usuários para acelerar o processo de desenvolvimento. Diferentemente de abordagens tradicionais, o RAD permite ajustes constantes com base no feedback dos usuários, permitindo maior flexibilidade e adaptação às necessidades do projeto.

As demais alternativas são incorretas porque descrevem características de outros modelos, como o ciclo em cascata (A e C), ou apresentam conceitos equivocados sobre o RAD (B e D).

Fase de planejamento de requisitos

O planejamento de requisitos é a fase inicial e indispensável no desenvolvimento de software, pois estabelece as bases para a construção do sistema. Nessa etapa, são identificadas as necessidades do negócio, o escopo do projeto, as restrições e os requisitos essenciais para o funcionamento adequado da aplicação. Um planejamento bem estruturado garante que os requisitos sejam claros, completos e viáveis, minimizando riscos e retrabalho durante o desenvolvimento.

Além disso, essa fase envolve a colaboração entre as partes interessadas para alinhar expectativas e definir prioridades, assegurando que o sistema atenda aos objetivos propostos de forma eficiente e estruturada.

Confira no vídeo a seguir a importância do planejamento de requisitos no desenvolvimento de software, focando identificação de necessidades do negócio, definição de escopo e estabelecimento de requisitos claros. Veja como um bom planejamento evita falhas, reduz retrabalho e assegura eficiência no processo.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Conceitos

Independentemente da metodologia que seja empregada para desenvolver um software, é um fato que todo sistema é desenvolvido com um propósito. Para chegar a esse objetivo, um conjunto de itens, chamado de requisitos, deve ser implementado.

> A área que trata dos requisitos é a **engenharia de requisitos**. Ela tem como metas a identificação, modelagem, comunicação e documentação dos requisitos de um sistema e em quais situações ele vai operar. O modo como os requisitos serão implementados é uma decisão que deve ser tomada pela equipe de desenvolvimento.

Os requisitos do sistema basicamente descrevem o que deve ser feito. É importante garantir que eles sejam completos, não se contradigam e que sejam relevantes para o desenvolvimento do sistema.

Comentário

Quanto maior o detalhamento dos requisitos logo no começo do projeto, menores serão as chances de haver a necessidade de mudanças no sistema, quando o desenvolvimento já estiver avançado.

De acordo com Pressman (2011), a engenharia de requisitos consiste em cinco atividades principais, veja!

Elicitação

Identifica os requisitos do sistema.

Análise

Elenca os elementos necessários para tratar os requisitos.

Documentação

Comunica as partes envolvidas sobre o entendimento dos requisitos.

Validação

Verifica a implementação do que foi solicitado.

Gerenciamento

Consiste na priorização dos requisitos.

Elicitação de requisitos

A fim de obter melhor compreensão do sistema a ser desenvolvido, a elicitação de requisitos, ou ainda, o levantamento de requisitos, tem por objetivo identificar os requisitos do sistema e em qual contexto será feita a operação deste com o apoio das partes interessadas. Por contexto, entende-se:

- Qual é o domínio da aplicação.
- Quais as necessidades do negócio.

- Quais são as restrições do sistema.

- A quem a aplicação se destina.

- Quais os pormenores da operação.

Entrevistar as partes interessadas é um método para descobrir fatos e opiniões de usuários em potencial e evitar mal-entendidos que podem comprometer o desenvolvimento do sistema.

As entrevistas podem ser de dois tipos:

Fechadas

As partes interessadas respondem a um conjunto predefinido de perguntas.

Abertas

O engenheiro de requisitos e as partes interessadas discutem o que esperam do sistema.

As entrevistas são vantajosas por fornecerem informações úteis para os desenvolvedores.

A desvantagem é que, por se tratar de informações subjetivas, o entrevistador deverá estar atento com a possibilidade de informações conflitantes.

Dada a importância do levantamento de requisitos para todo o projeto, é necessário fazer uso de técnicas que auxiliem extrair a maior quantidade de informações úteis. Confira algumas delas a seguir.

Casos de uso

São usados para descrever as interações entre os usuários e o sistema. Eles especificam uma sequência de interações entre um sistema e um ator externo (por exemplo, uma pessoa, uma peça de hardware, outro produto de software).

Os requisitos funcionais do sistema de software são representados pelos casos de uso, que são, portanto, um importante instrumento para que analistas e usuários possam interagir e validar o entendimento dos requisitos.

Cenários

Têm por objetivo simular situações de interações entre os usuários e o sistema.

Na descrição dos cenários, deve ser informado como o sistema estava antes de o usuário começar a interação e como ficou após esta ter sido concluída.

Observação e análise social

Aqui, um analista observa os usuários enquanto trabalham e faz anotações.

A principal vantagem dessa técnica é mostrar o modo como os usuários realmente trabalham e não uma descrição idealizada de como devem trabalhar.

Grupos focais

São formados por grupos com visão de partes do sistema, os quais ajudam a identificar as necessidades e percepções dos usuários.

Brainstorming ou tempestade cerebral

Trata-se de uma reunião em que os participantes podem desenvolver soluções criativas para problemas específicos.

Esse processo tem duas fases: a de geração, na qual as ideias são coletadas; e a de avaliação, em que as ideias coletadas são discutidas.

Prototipagem

A ideia é desenvolver aplicações rápidas que ajudem a entender melhor os requisitos do sistema como um todo.

Os protótipos podem ser descartáveis e evolutivos.

Análise de requisitos

Todas as questões relacionadas aos requisitos são tratadas pela análise de requisitos, desde a determinação das precedências, da consistência deles (ou seja, não devem ser contraditórios entre si), da integridade dos requisitos (isso quer dizer que nenhum serviço ou restrição estará ausente) até a viabilidade dos requisitos.

> A análise de requisitos preocupa-se em verificar se a implementação dos requisitos é viável com o orçamento e cronograma disponíveis para o desenvolvimento do sistema. A partir da negociação de priorização com as partes interessadas, os conflitos são resolvidos.

A principal técnica usada para análise de requisitos são **sessões JAD**. Conheça um pouco mais sobre essas sessões a seguir.

O que são?

As sessões JAD — acrônimo do inglês para Joint Application Development — são oficinas de trabalho em que os desenvolvedores e clientes discutem o sistema.

Qual o objetivo?

O objetivo da JAD é detalhar a solução, de modo que seja possível extrair elementos passíveis de monitoramento até que o projeto seja concluído.

Depois da extração dos requisitos, é necessário priorizá-los para estabelecer um cronograma de trabalho que contemple o uso eficiente de recursos e atenda às expectativas do cliente.

Documentação de requisitos

Nesta etapa, o objetivo é comunicar os requisitos do sistema para as partes interessadas: clientes e desenvolvedores.

> O documento de requisitos é a referência para avaliar o sistema, ou seja, para construir testes diversos, verificação e validação e para fazer o controle de mudanças. Esse documento é essencial para que haja concordância entre as partes envolvidas e pode, inclusive, fazer parte do contrato.

Gerenciamento de requisitos

A captura, o armazenamento, a disseminação e o gerenciamento de informações compõem o gerenciamento de requisitos.

> É na etapa de gerenciamento que as atividades relacionadas ao controle de mudança e versão, rastreamento de requisitos e dos estados dos requisitos são tratadas.

A rastreabilidade de requisitos mapeia a conexão entre requisitos, design e implementação de um sistema para gerenciar mudanças em um sistema.

Atividade 2

Qual é o principal objetivo do planejamento de requisitos no desenvolvimento de software?

A Criar o código-fonte do sistema antes de definir suas funcionalidades.

B Identificar necessidades do negócio, definir escopo e estabelecer requisitos claros para guiar o desenvolvimento.

C Priorizar exclusivamente a estética da interface do usuário sem considerar as funcionalidades do sistema.

D Desenvolver um protótipo final sem necessidade de validação com as partes interessadas.

E Ignorar restrições e priorizar apenas requisitos sugeridos pelos desenvolvedores.

A alternativa B está correta.

O planejamento de requisitos tem como principal objetivo identificar as necessidades do negócio, definir o escopo e estabelecer requisitos claros que guiem o desenvolvimento do software. Isso garante que o sistema atenda às expectativas dos usuários e funcione conforme o esperado.

As demais alternativas estão incorretas porque desconsideram etapas essenciais do processo: A ignora a necessidade de definir funcionalidades antes da implementação; C foca apenas a estética sem considerar requisitos funcionais; D desvaloriza a validação com stakeholders; E negligencia restrições e requisitos críticos do projeto.

Engenharia de requisitos no RAD

É essencial na adaptação dinâmica dos requisitos ao longo do projeto. Diferentemente dos modelos tradicionais, que buscam definir todos os requisitos no início, o RAD incorpora um processo evolutivo, permitindo que os usuários refinem suas necessidades com base nas entregas parciais.

A elicitação de requisitos, frequentemente realizada por meio de oficinas JAD, possibilita uma colaboração contínua entre desenvolvedores e usuários, fazendo com que as funcionalidades prioritárias sejam identificadas e implementadas em cada iteração. Esse modelo flexível melhora a comunicação, reduz retrabalho e acelera a entrega de soluções eficientes.

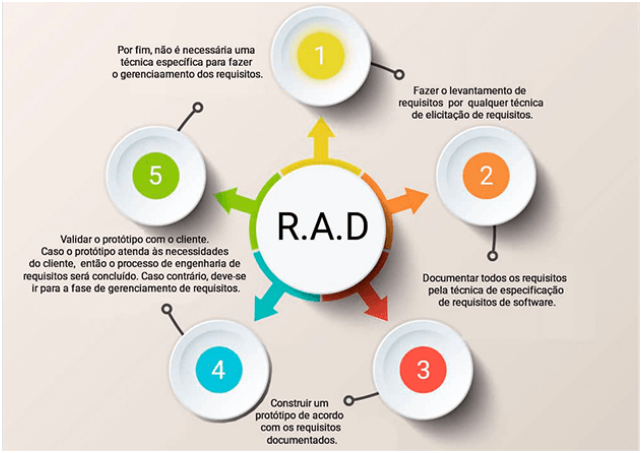
No vídeo, confira como a elicitação, priorização e validação contínua dos requisitos contribuem para um desenvolvimento ágil, além do papel das oficinas JAD e protótipos na colaboração entre usuários e desenvolvedores, garantindo soluções alinhadas às necessidades reais.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Modelo tradicional de engenharia de requisitos aplicado ao RAD

Segundo Pressman (2011), o modelo tradicional de engenharia de requisitos para desenvolvimento rápido de software tem as etapas apresentadas a seguir.



Observe, a seguir, alguns dos problemas para aplicar técnicas de engenharia de requisitos dos modelos tradicionais para a RAD.

Evolução dos requisitos

A RAD se caracteriza por ser um processo evolutivo no qual é incluída uma característica para o projeto em cada ciclo de desenvolvimento. Isso ocorre porque existe a premissa de que os usuários evoluem o seu entendimento do projeto com a análise das entregas, portanto, o modelo tradicional é inadequado quando tenta levantar todos os requisitos logo no início do desenvolvimento.

Conflitos na implementação dos requisitos

O modelo tradicional não trata de forma específica os conflitos que podem ocorrer com a implementação dos requisitos ao longo do projeto, como é o caso da RAD em que o projeto vai incorporando sugestões dos usuários ao longo das iterações.

Priorização dos requisitos

Os mecanismos de priorização de requisitos no modelo tradicional de engenharia de requisitos não se aplicam a projetos baseados na RAD, pois a cada iteração de desenvolvimento do protótipo é necessário estabelecer quais são os requisitos de alta prioridade.

Gerenciamento de prioridades e mudanças

Na RAD, quando o cliente faz sugestões a partir da análise do protótipo, é necessário gerenciar as prioridades do que vai ser implementado para a próxima versão. Os modelos tradicionais não tratam de como deve ser feito o gerenciamento dessas mudanças.

Como visto, então, a RAD precisa de um tratamento específico para fazer o gerenciamento dos requisitos.

Elicitação e validação de requisitos no RAD

A seguir, acompanhe como acontece a elicitação de requisitos e sua validação no RAD.

Elicitação de requisitos

É a principal etapa na RAD, e isso ocorre porque o protótipo depende dela.

A técnica mais utilizada para levantamento de requisitos na RAD é a JAD. Como vimos, nas oficinas JAD, as partes interessadas — usuários e desenvolvedores — discutem o sistema e definem seus requisitos, debatendo sobre as necessidades do negócio.

Tudo isso é muito útil para direcionar o desenvolvimento do projeto e remover todas as ambiguidades a respeito dos requisitos. Assim, ganha-se tempo, que é um recurso caro e limitado para qualquer projeto.

Análise e gerenciamento de conflitos

Após a extração dos requisitos pela JAD, é necessário analisá-los para garantir sua integridade e consistência.

Uma das técnicas comumente utilizadas para análise de requisitos é o gerenciamento de conflitos. Essa técnica vai tratar das possíveis contradições que surgiram na JAD.

Como a RAD tem como característica desenvolver protótipos, é fundamental que os conflitos, se existirem, sejam removidos, pois não faz sentido implementar um protótipo com requisitos conflitantes.

Priorização de requisitos

É necessária antes do desenvolvimento do protótipo. Seguindo uma sequência lógica, no protótipo, primeiro desenvolvem-se as principais funcionalidades do sistema.

Uma técnica usada para isso é a classificação por cartões, que consiste na atribuição de “graus de importância” para cada requisito. Dessa forma, é possível comparar as prioridades das diferentes partes interessadas.

Nessa técnica, **todas as funcionalidades do sistema são escritas em cartões** individuais e os cartões são dados para todos os participantes das oficinas de trabalho; em seguida, **cada um atribui a prioridade que acha ser a adequada para o requisito** e mostra para os demais. Após algumas rodadas, todos os cartões têm algumas prioridades. Por fim, são feitas a **análise dos graus de todos os cartões** e a **priorização das funcionalidades com as quais todas as partes interessadas concordam**. Trata-se de uma técnica colaborativa e orientada para atender às necessidades do cliente.

Documentação dos requisitos

Após a priorização dos requisitos, passa-se para a documentação deles. Trata-se da formalização do acordo sobre o entendimento dos requisitos entre as partes interessadas.

Uma das técnicas para fazer a documentação dos requisitos é através da especificação de requisitos de software.

Essa técnica também é utilizada pelos modelos tradicionais para documentação dos requisitos.

Protótipo

Essa etapa refere-se à **principal característica do ciclo de vida de desenvolvimento da RAD**, que é a **implementação do protótipo**.

O **protótipo** é um programa funcional com um escopo reduzido que reflete para onde o desenvolvimento do sistema está sendo direcionado.

O seu **desenvolvimento** é feito a partir da priorização dos requisitos, que significa que, inicialmente, as principais funcionalidades do sistema devem ser implementadas.

Validação e design

Terminado o protótipo, ele é apresentado para ser validado pelo cliente. Essa é a etapa mais importante em qualquer ciclo de vida de desenvolvimento, pois todo sistema tem um propósito e, quem o avalia, é o usuário.

Na RAD, a validação do protótipo é muito importante para direcionar quais serão as ações a serem tomadas para a próxima iteração.

Atenção: na validação do protótipo, o cliente o verifica com base na documentação dos requisitos. No caso de o protótipo corresponder aos requisitos, ele será válido e a próxima fase é a de design. Mas, se o cliente rejeitar o protótipo e solicitar melhorias, então será necessário ir para o **gerenciamento de requisitos**.

O desenvolvimento rápido de software pode ser um método muito eficaz, desde que sejam observados os pré-requisitos para sua aplicação.

A RAD cria um ambiente colaborativo e criativo em que as partes interessadas podem participar de um projeto muito detalhado. Com ela, pode-se obter resultados rápidos e ideias de sucesso que, dificilmente, seriam implementadas nos modelos tradicionais. É um fato, no entanto, que a RAD também tem suas desvantagens. Seu uso depende de uma equipe de trabalho com diversas habilidades e motivada, capaz de se adequar rapidamente a mudanças ao longo do projeto.

Atividade 3

Qual é o principal objetivo da elicitação de requisitos no desenvolvimento rápido de software (RAD)?

A Definir todos os requisitos do sistema logo no início do projeto, sem possibilidade de mudanças.

B Eliminar a necessidade de validação dos requisitos, reduzindo o tempo de desenvolvimento.

C Criar um protótipo final antes de envolver os usuários no processo de desenvolvimento.

D Identificar e refinar os requisitos do sistema de forma colaborativa, permitindo ajustes ao longo das iterações.

E Priorizar apenas requisitos técnicos, ignorando as necessidades dos usuários e do negócio.

A alternativa D está correta.

No desenvolvimento rápido de software (RAD), a elicitação de requisitos tem como objetivo identificar e refinar as necessidades do sistema de forma colaborativa, permitindo ajustes ao longo das iterações. Esse processo garante que o software atenda às expectativas dos usuários e evolua conforme necessário.

As demais alternativas são incorretas: A sugere rigidez incompatível com o RAD; B ignora a importância da validação dos requisitos; C exclui a participação dos usuários; E desconsidera a relevância das necessidades do negócio.

Prática de elicitação, análise e priorização de requisitos no RAD

A engenharia de requisitos permite o desenvolvimento ágil e eficiente sem comprometer a qualidade. A elicitação de requisitos alinha expectativas do cliente com possibilidades técnicas, enquanto técnicas como a **oficina JAD** facilitam a interação entre clientes e desenvolvedores.

A análise e o gerenciamento de conflitos resolvem inconsistências, e a priorização assegura que as funcionalidades mais importantes sejam desenvolvidas primeiro. Neste estudo, aplicamos esses conceitos no desenvolvimento fictício de um sistema para pedidos em uma lanchonete, cobrindo **elicitação**, **análise** e **priorização**. Essa abordagem estruturada minimiza retrabalho e prepara a base para a próxima fase: a prototipação do sistema.

No vídeo a seguir, confira a engenharia de requisitos no desenvolvimento rápido de software (RAD), destacando a elicitação, análise e priorização de requisitos com um exemplo prático. Veja também técnicas para estruturar sistemas de forma ágil e eficiente.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

A lanchonete Sabor Rápido enfrenta dificuldades no registro manual de pedidos, o que causa erros na cobrança e demora no atendimento. O dono da lanchonete deseja um sistema simples que agilize a anotação dos pedidos e facilite o cálculo do total.

Vamos aplicar a engenharia de requisitos no modelo RAD para desenvolver um sistema de registro de pedidos para a lanchonete Sabor Rápido. A atividade envolverá a elicitação de requisitos por meio de uma oficina JAD, a análise e a resolução de conflitos, além da priorização das funcionalidades essenciais. Ao final, os requisitos

definidos orientarão a prototipação do sistema, garantindo um desenvolvimento ágil e alinhado às necessidades do cliente. Vamos lá!

Passo 1: Elicitação de requisitos (Oficina JAD)

Objetivo: identificar os requisitos essenciais do sistema por meio de uma sessão colaborativa.

Participantes:

- **Cliente:** dono da lanchonete.
- **Desenvolvedor:** responsável pela implementação.
- **Facilitador:** conduz a oficina JAD e documenta as decisões.

Discussão e respostas:

1. Quais funcionalidades são essenciais?

- Registrar pedidos dos clientes de forma rápida.
- Calcular automaticamente o total do pedido.
- Permitir a exclusão de itens antes da finalização.
- Gerar um resumo do pedido antes da cobrança.

2. Como os pedidos devem ser registrados?

- Lista de itens pré-cadastrados no sistema (sem necessidade de digitação).
- Seleção de itens pelo atendente com um clique.

3. Quais informações precisam estar no sistema?

- Nome do produto.

- Quantidade solicitada.

- Preço unitário e total.

4. Há necessidade de integração com outra ferramenta?

- O cliente deseja apenas um sistema local, sem integração com planilhas ou internet.

Passo 2: Análise de requisitos (gerenciamento de conflitos)

Com base nas respostas da sessão JAD, identificamos alguns conflitos:

1. O cliente quer um sistema simples, mas deseja excluir itens do pedido antes de finalizar.

- **Solução:** implementar um botão de remoção rápida para evitar complexidade na interface.

2. O dono quer que os itens sejam cadastrados previamente, mas também quer flexibilidade para adicionar novos produtos futuramente.

- **Solução:** criar uma interface simples para o cadastro de novos itens quando necessário.

Passo 3: Priorização dos requisitos

Técnica: classificação por cartões.

Cada requisito é escrito em um cartão e os participantes os classificam em ordem de prioridade.

Ordem final de prioridade:

1. Registrar pedidos de forma rápida.

2. Calcular automaticamente o total do pedido.

3. Excluir itens antes da finalização.

4. Gerar um resumo do pedido antes da cobrança.

5. Cadastrar novos produtos manualmente.

Com essa priorização, a equipe pode iniciar a próxima etapa: a prototipação do sistema. Esse exercício prático simula como a engenharia de requisitos é aplicada no RAD, garantindo um desenvolvimento ágil e alinhado às necessidades do cliente.

Atividade 4

Durante a aplicação da engenharia de requisitos no desenvolvimento rápido de software (RAD), qual é a sequência correta das etapas abordadas nessa prática?

A Desenvolvimento do código → Elicitação de requisitos → Priorização de funcionalidades → Testes finais.

B Elicitação de requisitos → Análise de requisitos (gerenciamento de conflitos) → Priorização de requisitos.

C Definição do design → Testes de usabilidade → Priorização de requisitos → Codificação.

D Levantamento de dados do mercado → Priorização de funcionalidades → Implementação direta do sistema.

E Codificação do sistema → Ajuste de requisitos → Validação final sem revisão prévia.

A alternativa B está correta.

No desenvolvimento rápido de software (RAD), a engenharia de requisitos segue uma sequência estruturada para garantir que o sistema atenda às necessidades do cliente de forma ágil e eficiente.

A prática apresentada segue as etapas: elicitação de requisitos: as funcionalidades são identificadas por meio da oficina JAD; análise de requisitos (gerenciamento de conflitos): possíveis divergências são resolvidas; priorização de requisitos: garante que os aspectos mais críticos serão desenvolvidos primeiro.

Desenvolvendo um protótipo RAD na prática

É uma abordagem eficiente para criar protótipos interativos, permitindo ajustes contínuos.

Neste estudo, implementamos um sistema de pedidos para uma lanchonete usando **Python** e **Tkinter**. O objetivo é estruturar uma aplicação funcional com interface gráfica intuitiva, permitindo selecionar itens, visualizar pedidos, calcular o total e adicionar novos produtos. Técnicas como manipulação de eventos e interação com listas garantem uma experiência dinâmica.

A prática aborda **design de interfaces**, **tratamento de entradas** e **desenvolvimento iterativo**, conceitos essenciais para programadores. Ao final, o usuário terá um protótipo funcional, facilmente expansível conforme surjam novas necessidades.

No vídeo a seguir, aprenda a criar um protótipo RAD para um sistema de pedidos em uma lanchonete usando Python e Tkinter. Veja como estruturar a interface, adicionar itens, calcular o total e atualizar o menu, permitindo o desenvolvimento rápido e eficiente de aplicações interativas.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Esse roteiro orienta o desenvolvimento de um **protótipo RAD** para um sistema simples de pedidos de uma lanchonete. O objetivo é demonstrar como um software pode ser estruturado de forma rápida e iterativa usando **Python e Tkinter**.

Passo 1: Criando a estrutura inicial do programa.

O primeiro passo é definir a classe principal que gerenciará a interface gráfica do sistema.

````
python

```python
import tkinter as tk
from tkinter import messagebox

class SaborRapidoApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Sabor Rápido - Protótipo")
        self.root.geometry("400x500")

        self.itens_menu = {"Hambúrguer": 10.00, "Batata Frita": 5.00, "Refrigerante":
3.00}
        self.pedido = []
```
````

Explicação: criamos a classe SaborRapidoApp, que inicializa a interface gráfica e define um dicionário contendo os itens do menu e seus preços.

Passo 2: Criando a lista de itens do menu.

Adicionamos uma **Listbox** para exibir os produtos disponíveis e um método para atualizar a lista.

````
python

```python
        tk.Label(root, text="Selecione os itens do pedido:", font=("Arial",
12)).pack(pady=10)

        self.listbox = tk.Listbox(root, selectmode=tk.MULTIPLE, font=("Arial", 10))
        self.atualizar_lista_menu()
        self.listbox.pack()
```
```python
    def atualizar_lista_menu(self):
        self.listbox.delete(0, tk.END)
        for item in self.itens_menu.keys():
            self.listbox.insert(tk.END, item)
```
````

Explicação:

✔ Criamos um **rótulo** informativo.

✔ Adicionamos uma **lista interativa** com os itens do menu.

✔ Implementamos `atualizar_lista_menu()` para garantir que a lista sempre exiba os itens disponíveis.

Passo 3: Criando funções para adicionar e visualizar pedidos.

Adicionamos botões para permitir ao usuário adicionar itens ao pedido e visualizá-lo.

````
python

```python
        tk.Button(root, text="Adicionar ao Pedido",
command=self.adicionar_pedido).pack(pady=5)
        tk.Button(root, text="Visualizar Pedido",
command=self.visualizar_pedido).pack(pady=5)
```



```python
    def adicionar_pedido(self):
        selecionados = self.listbox.curselection()
        for index in selecionados:
            item = self.listbox.get(index)
            self.pedido.append(item)
        messagebox.showinfo("Pedido", "Itens adicionados com sucesso!")

    def visualizar_pedido(self):
        if not self.pedido:
            messagebox.showinfo("Pedido", "Nenhum item no pedido.")
            return
        pedido_texto = "\n".join(self.pedido)
        messagebox.showinfo("Pedido Atual", f"Itens no pedido:\n{pedido_texto}")
```
````

Explicação:

✔ `adicionar_pedido()`: Captura os itens selecionados e os adiciona à lista de pedidos.

✔ `visualizar_pedido()`: Exibe os itens já adicionados ao pedido.

Passo 4: Criando função para finalizar pedido.

Implementamos a **finalização do pedido**, calculando o total da compra.

````
python

```python
        tk.Button(root, text="Finalizar Pedido",
command=self.finalizar_pedido).pack(pady=10)
```
```python
    def finalizar_pedido(self):
        if not self.pedido:
            messagebox.showinfo("Pedido", "Adicione itens antes de finalizar o pedido.")
            return
        total = sum(self.itens_menu[item] for item in self.pedido)
        messagebox.showinfo("Total", f"Total do pedido: R$ {total:.2f}\nPedido
finalizado!")
        self.pedido.clear()
```
````

Explicação:

✔ Se houver itens no pedido, soma os valores e exibe o total.

✔ Após a finalização, a lista de pedidos é **limpa**.

Passo 5: Adicionando novo item ao menu.

Permitir que o **usuário adicione novos itens ao menu**, com nome e preço.

python

```python
        tk.Label(root, text="Adicionar Novo Item ao Menu:", font=("Arial",
12)).pack(pady=10)
        self.entry_item = tk.Entry(root, font=("Arial", 10))
        self.entry_item.pack()
        self.entry_preco = tk.Entry(root, font=("Arial", 10))
        self.entry_preco.pack()
        tk.Button(root, text="Adicionar Item",
command=self.adicionar_item_menu).pack(pady=5)
```

```python
    def adicionar_item_menu(self):
        item = self.entry_item.get().strip()
        preco = self.entry_preco.get().strip()
        if item and preco:
            try:
                self.itens_menu[item] = float(preco)
                self.atualizar_lista_menu()
                self.entry_item.delete(0, tk.END)
                self.entry_preco.delete(0, tk.END)
                messagebox.showinfo("Sucesso", "Item adicionado ao menu com sucesso!")
            except ValueError:
                messagebox.showerror("Erro", "Preço inválido. Digite um valor numérico.")
        else:
            messagebox.showerror("Erro", "Preencha ambos os campos corretamente.")
```

Explicação:

✔ Permite ao usuário adicionar um novo item com nome e preço.

✔ Atualiza automaticamente a lista de itens no menu.

Passo 6: Executando o programa.

Por fim, executamos o programa.

python

```python
if __name__ == "__main__":
    root = tk.Tk()
    app = SaborRapidoApp(root)
    root.mainloop()
```

Explicação:

✔ Inicializa a interface gráfica e executa o programa.

Testando o protótipo:

1. Abrir o programa e verificar se os itens do menu aparecem corretamente.

2. Selecionar itens e adicioná-los ao pedido.

3. Visualizar os itens no pedido.

4. Finalizar o pedido e conferir o total.

5. Adicionar um novo item ao menu e verificar se foi atualizado corretamente.

Atividade 5

Se o desenvolvedor alterar a seguinte linha:

`self.pedido.append(item)`

Na função:

`adicionar_pedido()` para `self.pedido = item`

O que acontecerá?

A O sistema apresentará erro ao tentar visualizar ou finalizar o pedido, pois `self.pedido` deixará de ser uma lista.

B O sistema funcionará normalmente, mas os pedidos serão armazenados de forma mais eficiente.

C O pedido será registrado corretamente, mas apenas o último item selecionado ficará armazenado, substituindo os anteriores.

D A interface gráfica deixará de exibir os itens do menu corretamente.

E Nada mudará no funcionamento do sistema, pois a modificação não afeta a lógica principal.

A alternativa A está correta.

A variável `self.pedido`, que originalmente era uma lista (`[]`), passará a armazenar **apenas uma string** do último item selecionado, fazendo com que a função `sum(self.itens_menu[item] for item in self.pedido)` na finalização do pedido gere um erro, pois tentará iterar sobre uma string ao invés de uma lista.

Conceito de modelagem de sistema no RAD

A modelagem no desenvolvimento rápido de software (RAD) é fundamental na formalização e visualização dos sistemas em construção. Por meio de diagramas e representações gráficas, essa abordagem facilita a compreensão e comunicação das ideias entre as partes interessadas.

No RAD, a modelagem ocorre em duas etapas principais: modelagem de negócios, que identifica e documenta os processos empresariais e requisitos funcionais; e modelagem de dados, que organiza as informações essenciais para garantir a estrutura adequada do sistema. Uma modelagem bem-feita melhora a qualidade do desenvolvimento, permitindo ajustes rápidos e garantindo que o sistema atenda às necessidades do negócio.

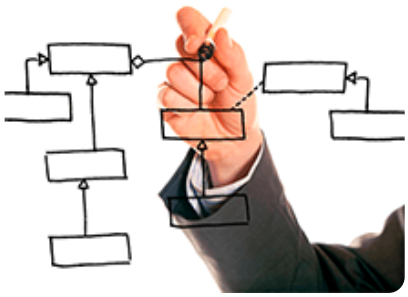
Confira, no vídeo a seguir, como a modelagem de negócios e de dados melhora a estruturação de sistemas, facilitando comunicação e adaptação. Veja também o uso de diagramas e representações visuais para um desenvolvimento ágil e alinhado ao negócio.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

A modelagem de sistema formaliza como os sistemas são definidos. É comum que, durante o desenvolvimento, sejam usadas imagens para ajudar a visualizar alguns aspectos do sistema.

Através de representações por meio de diagramas, a modelagem fornece um meio para compreender e comunicar as ideias associadas ao desenvolvimento do sistema.



A RAD é uma metodologia de desenvolvimento de software evolutiva que, além do levantamento de requisitos, possui outras fases. De modo resumido, segundo Kerr e Hunter (1994), elas podem ser descritas da forma apresentada a seguir.

Vamos entender agora a fase da modelagem de negócio.

Obtenção de informações

Nessa fase, é feita a obtenção de informações a respeito dos requisitos funcionais do sistema, reunidas por várias fontes relacionadas aos negócios.

Combinação de informações e documentação

Essas informações são então combinadas para criar uma documentação que será utilizada para modelar o ciclo de vida dos dados: como os dados podem ser usados, quando são processados e a sua transformação em informações que serão utilizadas por alguma área, ou setor específico do negócio.

Caráter comercial
Trata-se, portanto, de uma análise com viés comercial.

Agora, observe a fase de modelagem de dados.

Análise das informações
Nessa fase, todas as informações que foram obtidas durante a fase anterior são analisadas para formar conjuntos de objetos de dados essenciais para os negócios.

Agrupamento das informações
Através da análise, as informações são agrupadas de modo que sejam úteis para a empresa. Por exemplo, na determinação de quais são as entidades principais que serão tratadas pelo sistema. A qualidade de cada grupo de dados, então, é examinada e recebe uma descrição precisa.

Mapeamento dos grupos de informações
É feito mapeamento que relaciona esses grupos entre si e qual o significado desses relacionamentos, conforme definido na etapa precedente.

Definição dos atributos dos dados e sua relevância
Deve-se identificar e definir os atributos de todos os conjuntos de dados. Também é estabelecida e definida em detalhes de relevância para o modelo de negócios a relação entre esses objetos de dados.

A modelagem não é um método exato e exige muita atenção do analista, pois vai ter impacto para todo o projeto. Quanto melhor for o entendimento do sistema, melhor será a qualidade da modelagem e, portanto, mais eficiente será a tradução das demandas do cliente para a implementação do sistema. Os modelos e as informações são representados por meio de diagramas conectados.

Atividade 1

Qual é a principal função da modelagem de sistema no desenvolvimento rápido de software (RAD)?

A Criar representações visuais para facilitar a compreensão e comunicação das ideias do sistema.

B Substituir completamente o processo de levantamento de requisitos no desenvolvimento do software.

C Focar apenas a interface gráfica do sistema, sem considerar sua estrutura de dados.

D Garantir que todos os requisitos do sistema sejam definidos antes do início do desenvolvimento, sem alterações posteriores.

E Eliminar a necessidade de validação dos requisitos junto aos usuários, acelerando o processo de entrega.

A alternativa A está correta.

No desenvolvimento rápido de software (RAD), a modelagem de sistema tem a função principal de criar representações visuais, como diagramas e protótipos, para facilitar a compreensão e a comunicação das ideias entre desenvolvedores e stakeholders. Isso permite ajustes rápidos e alinhamento com as necessidades do cliente.

As demais alternativas estão incorretas. B sugere a substituição do levantamento de requisitos; C restringe a modelagem apenas à interface; D impõe rigidez incompatível com o RAD; E desconsidera a importância da validação com os usuários.

Modelagem de negócios

Essa etapa ajuda a estruturar e visualizar os processos que serão incorporados ao sistema. Utilizando diagramas de casos de uso da UML, a modelagem de negócios permite mapear como as partes interessadas interagem com o sistema, oferecendo melhor entendimento das demandas do cliente.

A modelagem auxilia na definição dos processos principais, auxiliares e de gerenciamento, permitindo a reutilização de fluxos de trabalho e a identificação de redundâncias. Com um escopo bem definido, evita-se o excesso de informações desnecessárias, assegurando que o desenvolvimento do sistema esteja alinhado às reais necessidades do negócio.

Veja, no vídeo, como a modelagem de negócios no RAD torna o desenvolvimento mais eficiente e alinhado ao cliente. Confira diagramas de casos de uso da UML para mapear processos, otimizar fluxos e garantir um desenvolvimento ágil e estruturado.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

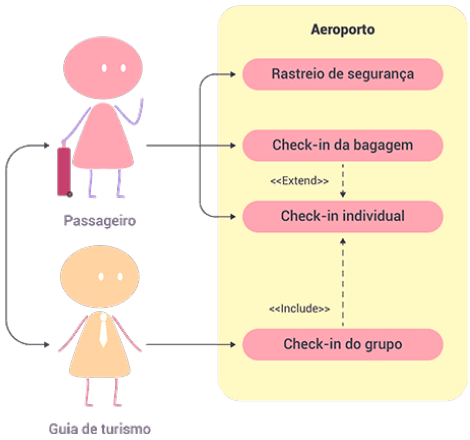
Uma das principais técnicas para representar a modelagem de negócios é utilizar o diagrama de casos de uso da UML.

Comentário

Cabe lembrar que a UML (Unified Modeling Language) - que é traduzida para o português como Linguagem Unificada de Modelagem - é uma linguagem padrão para modelagem orientada a objetos. O objetivo dela é dar suporte para que analistas de negócios e desenvolvedores visualizem os seus projetos em diagramas padronizados. Os diagramas mais comuns são o de casos de uso e o de classes.

A escolha do diagrama de casos de uso para modelagem de negócios ocorre porque esse diagrama incorpora a dinâmica dos negócios sob várias perspectivas, descrevendo, assim, como as partes interessadas — chamadas de atores — interagem com o sistema e entre si.

Na imagem a seguir, é apresentado um exemplo de caso de uso de um sistema simplificado de embarque em um aeroporto.



No modelo apresentado, são destacados os atores “passageiro” e “guia de turismo”. Nas elipses, estão os casos de uso que representam os elementos do sistema com os quais os usuários interagem.

Além disso, o diagrama possui um nome, no exemplo do diagrama de casos de uso da imagem anterior, o nome é “Aeroporto”, que ilustra o escopo do negócio que está sendo modelado.

Outros pontos a serem destacados são como os elementos do negócio interagem entre si representando um fluxo de trabalho que será, posteriormente, implementado no sistema.

Em um negócio, existem, pelo menos, três elementos de casos de uso de negócios. Confira!

Processos de negócios principais

Tratam das principais atividades do negócio.

Processos auxiliares

São atividades que precisam ser feitas de qualquer modo, para que o negócio funcione.

Processos de gerenciamento

Representam os processos que gerenciam outros processos e os relacionam com seus proprietários.

Em um restaurante, por exemplo, **os principais casos de uso de negócios** podem ser **marketing e serviços de almoço e de jantar**, e o **caso de uso auxiliar** pode ser **a compra de itens de alimentação e limpeza**.

Atenção

Os casos de uso principais de negócios devem estar, obrigatoriamente, relacionados a um ator de negócios. Isso ocorre para impor que os negócios estejam vinculados aos serviços que seus usuários solicitam. Uma forma de verificar que a modelagem de negócios está errada é se modelo de caso de uso tiver casos de uso que ninguém solicita.

São os casos de uso principais de negócios que representam a dinâmica do negócio. Eles são acionados por atores que os iniciam e esperam como resposta aos serviços deles. Em outras situações, um ator receberá respostas, ainda que os casos não tenham sido iniciados por ele (ator de negócios).

A modelagem de casos de uso de negócio facilita o entendimento dos principais processos que estão sendo mapeados.

Essa modelagem auxilia na reutilização de partes de fluxos de trabalho que são compartilhadas entre muitos casos de uso de negócios, além de ser um importante instrumento de interação entre as partes interessadas, para evoluir aspectos que precisem de mais detalhamento ou de correção.

Trata-se de uma forma muito flexível de trabalho em que é possível fazer generalizações, detalhamentos e reutilizações de casos de uso dependendo de como ele se contextualiza dentro do negócio.

Comentário

O esforço de modelagem de negócios deve estar circunscrito às necessidades que serão tratadas pelo sistema, ou seja, o escopo da modelagem não deve ultrapassar o que realmente precisa ser modelado, sob o risco de perder o foco do que será tratado e perdendo, assim, a vantagem da modelagem com um excesso de informações que acabam gerando confusão e que, posteriormente, podem ser difíceis de ser eliminadas da modelagem.

A modelagem foca apenas um subconjunto dos processos de negócios, ou seja, nos processos que realmente fazem parte das necessidades demandadas pelos clientes.

Uma modelagem de casos de uso de negócios de boa qualidade deve ter algumas características. Veja quais são elas!

- O mapeamento das partes do negócio que compõem a demanda do cliente.
- Os casos de uso devem estar de acordo com o negócio que descrevem.
- Deve haver um equilíbrio entre o número e o tamanho dos casos de uso.

Ter poucos casos de uso torna o modelo mais fácil de entender, mas não podem deixar de descrever algum fluxo de trabalho que seja relevante para o negócio.

Deve-se ainda evitar a redundância de casos de uso. Caso se perceba a ocorrência de uma situação dessas, deve-se fazer uma generalização dos casos de uso, pois não fazer isso pode provocar inconsistências na modelagem dos negócios.

Exemplo

Como caso de generalização, pode-se ter um sistema de pagamento que permite que os frequentadores de uma academia paguem de duas formas: on-line e por telefone. Certamente, esses dois cenários terão muitas coisas em comum e diferentes. Alguns aspectos comuns entre esses dois cenários são: a especificação das informações pessoais e a especificação do meio de pagamento. Portanto, a melhor maneira de fazer essa modelagem é criar um caso de uso (o pai) que contém o comportamento comum e, em seguida, criar dois casos de uso filho especializados que herdam o comportamento do pai e que contêm as diferenças específicas para fazer o pagamento on-line, ou por telefone.

Atividade 2

Qual é o principal objetivo da modelagem de negócios no desenvolvimento rápido de software (RAD)?

A Criar um modelo genérico de negócios sem considerar as necessidades específicas do cliente.

B Mapear os processos e interações do negócio para que o sistema atenda às demandas reais dos usuários.

C Substituir completamente a necessidade de elicitação de requisitos no desenvolvimento do software.

D Priorizar a implementação do sistema, sem necessidade de representações visuais ou diagramas.

E Definir apenas a interface gráfica do sistema, sem considerar os processos internos do negócio.

A alternativa B está correta.

A modelagem de negócios no desenvolvimento rápido de software (RAD) tem como principal objetivo mapear os processos e interações do negócio, garantindo que o sistema desenvolvido atenda às demandas reais dos usuários. Isso permite um alinhamento eficiente entre o software e as necessidades operacionais da empresa.

As demais alternativas estão incorretas. A sugere um modelo genérico sem personalização; C elimina a importância da elicitação de requisitos; D ignora a necessidade de representações visuais; E limita a modelagem apenas à interface gráfica, sem considerar a estrutura e os fluxos do negócio.

Modelagem de dados

Essa etapa organiza e estrutura as informações que serão manipuladas pelo sistema. A modelagem de dados define como os dados serão armazenados, processados e utilizados, garantindo coerência e eficiência.

A modelagem ocorre em três níveis: **conceitual**, **lógico** e **físico**, permitindo uma transição estruturada do entendimento do negócio para a implementação em banco de dados. Além de facilitar a comunicação entre desenvolvedores e analistas, a modelagem de dados assegura a integridade das informações e melhora o desempenho e a manutenção do sistema.

No vídeo, veja a importância da modelagem de dados no RAD, abordando os níveis conceitual, lógico e físico. Descubra como estruturar informações para garantir eficiência, integridade e conformidade com normas como a LGPD.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

A modelagem de dados é o processo que formaliza o ciclo de vida dos dados de modo que possam ser tratados de um jeito formal, por exemplo, o armazenamento em um banco de dados.

Trata-se, portanto, de uma representação conceitual de objetos de dados, de suas associações e regras de manipulação. Ela auxilia na representação visual dos dados e no modo como as regras de negócios e demais imposições regulatórias devem ser cumpridas.

Exemplo

Um caso dessas imposições é a Lei Geral de Proteção de Dados — mais conhecida pelo acrônimo, LGPD —, que impõe diversas regras sobre a transparência e a proteção do uso de dados. Os modelos de dados aumentam a qualidade dos dados dos sistemas, uma vez que fazem uso de convenções de nomenclatura e de segurança.

Na sua essência, o modelo de dados visa a garantir que todos os objetos de dados — que serão usados pelo sistema — sejam representados com precisão. A ausência de dados pode produzir inconsistências com grande impacto para o negócio do cliente.

Esse modelo indica quais são os dados necessários e como devem ser organizados, em vez de apontar quais operações precisam ser feitas. Confira os tipos de técnicas mais comuns para representar os modelos de dados.

- Modelo de entidade e relacionamento (E-R).
- UML (Unified Modeling Language).

Em relação aos bancos de dados dos sistemas, os modelos de dados ajudam a fazer os projetos nos níveis conceitual, físico e lógico. O modo como o modelo é estruturado ajuda a definir as tabelas relacionais, seus campos, chaves primárias e estrangeiras e o modo como devem ser manipulados para atender às regras dos negócios.

Comentário

A construção de um modelo de dados de qualidade é um processo trabalhoso, mas investir nessa construção vai trazer benefícios no longo prazo tanto sob o ponto de vista tecnológico, com a facilidade de manipulação e manutenção dos dados, como do ponto de vista do negócio.

Basicamente, existem três tipos diferentes de modelos de dados. Veja!

Conceitual

Define o que está contido no sistema. Seu objetivo é a organização, definição do escopo, dos conceitos e das regras de negócios.

Lógico

Define o modo como o sistema deve ser implementado independentemente do banco de dados. O objetivo é desenvolver estruturas de dados e um conjunto de regras técnicas.

Físico

Descreve a forma como o sistema será implementado usando um sistema de banco de dados específico. O objetivo é a implementação do banco de dados que vai, de fato, operar.

A seguir, vamos detalhar um pouco mais cada um desses modelos.

Modelo conceitual

O principal objetivo desse modelo é determinar quais são as entidades, seus atributos e como elas se relacionam. Aqui, não há detalhes da estrutura real do banco de dados.

Observe os elementos básicos do modelo conceitual a seguir.

Entidade

Um elemento que é distinguível no domínio do negócio.

Atributo

Características, ou propriedades de uma entidade.

Relacionamento

Relação de dependência ou associação entre entidades.

Exemplos de **entidades** são **Escola** e **Aluno**. O **nome**, **matrícula** e **endereço** são exemplos de **atributos** da entidade Aluno. Um exemplo de **relacionamento** entre as entidades Escola e Aluno é dado por **"Uma Escola pode ter vários Alunos"**.

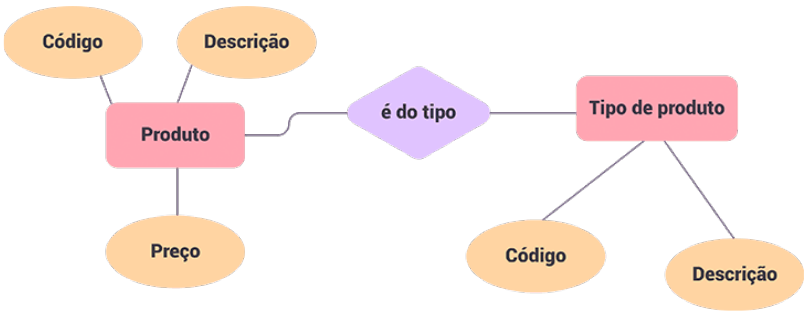
O modelo conceitual organiza os **conceitos de negócios**, portanto são essenciais no início do projeto como um meio de comunicação eficiente entre analistas de negócios e clientes. Ele é desenvolvido independentemente das especificações de hardware, ou de software.

> O objetivo do modelo conceitual é fazer a representação dos dados da forma como um usuário os visualiza no contexto do negócio. Também são conhecidos como modelos de domínio, isso porque criam um vocabulário comum para todas as partes interessadas através de conceitos básicos e escopo.

Modelo lógico

Em relação aos modelos de dados lógicos, o foco está na definição da estrutura dos dados e nas relações entre eles. Ele é iniciado a partir do modelo conceitual e da análise das requisições técnicas e de desempenho das operações que serão realizadas com os dados. Esse modelo cria a base para o modelo físico. Ele não está vinculado a nenhuma tecnologia específica, mas a forma como os dados serão estruturados visa a obter a otimização do desempenho e segurança deles apesar de sua estrutura de modelagem ser genérica.

Uma das técnicas mais comuns para representá-lo é através do **diagrama Entidade-Relacionamento**, conforme ilustrado na imagem a seguir.



No diagrama apresentado anteriormente, são modeladas as entidades Produto e Tipo de Produto. A entidade Produto possui os atributos preço, descrição e código e a entidade Tipo de Produto possui os atributos código e descrição. Além disso, as entidades são relacionadas entre si e os atributos códigos são utilizados para identificar de forma única cada elemento de suas respectivas entidades.

O **modelo lógico**, portanto, descreve as necessidades de dados para um projeto. Ele é projetado independentemente do banco de dados que será utilizado e os atributos das entidades fornecem a base da sua especificação que será implementada no banco de dados.

Existe outra etapa importante a respeito do modelo lógico, que é a normalização dos dados. Trata-se de uma técnica de organizar os dados de modo a evitar problemas de inconsistências entre eles.

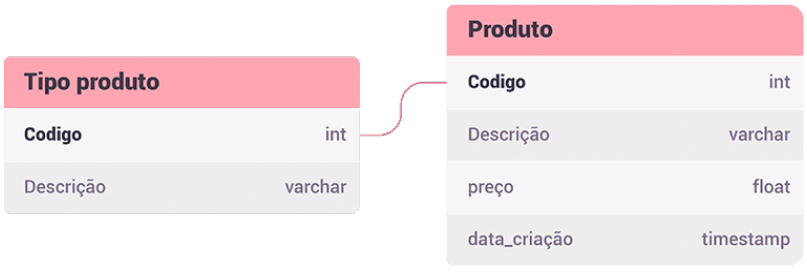
Comentário

Na seção Explore +, são feitas algumas sugestões para você se aprofundar mais sobre o tópico normalização de dados.

Modelo de dados físicos

Por fim, esse modelo descreve detalhadamente como será a implementação no banco de dados. Com ele, o desenvolvedor pode visualizar como os dados serão armazenados no banco de dados e usá-lo para fazer geração do esquema do banco de dados.

Na imagem a seguir, veja um exemplo de modelo físico de dados.



Na imagem apresentada, as entidades são representadas por tabelas com campos e seus respectivos tipos de dados. Por exemplo, a tabela Produto possui os atributos “código”, que é do tipo inteiro, e “descrição,” que é um tipo “cadeia de caracteres” (em muitos bancos de dados, esse tipo de dado é chamado de “VARCHAR”).

Esse modelo de dados ajuda a visualizar a estrutura do banco de dados e modelar os identificadores das tabelas (chaves primárias), as colunas, restrições, índices e outros recursos do gerenciamento do banco de dados. Além disso, ele contém relacionamentos entre tabelas.

Atividade 3

Qual é o principal objetivo da modelagem de dados no desenvolvimento rápido de software (RAD)?

A Criar um design visual para a interface do usuário, sem considerar a estrutura dos dados.

B Definir como os dados serão estruturados, armazenados e manipulados para garantir eficiência e integridade no sistema.

C Eliminar a necessidade de um banco de dados, substituindo-o por arquivos simples de armazenamento.

D Desenvolver o banco de dados apenas na fase final do projeto, sem planejamento prévio.

E Priorizar apenas a segurança dos dados, sem considerar sua organização e relacionamentos.

A alternativa B está correta.

A modelagem de dados no desenvolvimento rápido de software (RAD) tem como principal objetivo definir como os dados serão estruturados, armazenados e manipulados, oferecendo eficiência e integridade ao sistema. Isso permite um desenvolvimento ágil e alinhado às necessidades do negócio.

As demais alternativas estão incorretas. Letra A foca apenas a interface; C elimina a necessidade de banco de dados; D adia o planejamento essencial para a organização dos dados; E prioriza somente a segurança, sem considerar a estrutura e os relacionamentos dos dados.

Reflexões sobre modelagem de dados e de negócios

A modelagem de dados e de negócios ajuda o sistema a atender às necessidades do negócio e a ter uma estrutura eficiente. Enquanto a modelagem de negócios permite compreender e formalizar os processos empresariais, a modelagem de dados organiza e define a forma de armazenamento das informações.

No RAD, esses modelos evoluem continuamente à medida que o sistema se desenvolve, exigindo um equilíbrio entre flexibilidade e estabilidade. Esse processo melhora a comunicação entre as partes interessadas, otimiza a implementação do sistema e contribui para a entrega de soluções eficazes e adaptáveis.

No vídeo a seguir, veja as considerações finais sobre modelagem de dados e negócios no RAD, destacando sua evolução para garantir flexibilidade e alinhamento com o negócio. Confira como a estabilização do modelo de dados melhora a eficiência e a entrega ágil de soluções.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Considerações finais sobre modelagem de negócios e dados

O objetivo principal de um modelo de dados é oferecer meios para que os objetos de dados sejam representados com precisão. Para isso, eles devem ser detalhados o suficiente para ser aplicados na construção do banco de dados físico.

As informações obtidas a partir deles podem ser usadas para definir quais são as chaves primárias e estrangeiras das tabelas, o relacionamento entre elas, além de outras informações que serão úteis para fazer manipulação e manutenção dos dados.

> A **modelagem de dados** é um importante instrumento para a comunicação entre desenvolvedores e clientes, além de ajudar a documentar os mapeamentos de dados, comumente utilizados no processo de extração-transformação-carga de dados (ETL). Por fim, a modelagem de dados identifica fontes corretas de dados para preencher o banco de dados que será utilizado pelo sistema.

O modelo de dados é uma etapa essencial no ciclo de vida de qualquer desenvolvimento independentemente da metodologia utilizada. No caso da RAD, na qual a dinâmica do aprendizado do sistema está diretamente relacionada às entregas através dos protótipos, é natural que o modelo de dados também evolua, mas é esperado que ele rapidamente estabilize e o foco esteja em questões relacionadas ao detalhamento de regras de negócio e na interatividade do usuário com o sistema, ou seja, questões relacionadas a componentes interativos e experiência do usuário.

O perfil esperado de um desenvolvedor da RAD é de um profissional com formação sólida, facilidade de aprendizado e muito adaptável. Isso, inclusive, é a principal desvantagem da RAD, pois existem muitas expectativas, especialmente, dos desenvolvedores.

A evolução do sistema pode ter impacto significativo no que foi feito até então. No sentido de construir um sistema que realmente atenda às expectativas do cliente é excelente, pois as constantes interações fortalecem o entendimento e se concretizam em módulos que atendam às requisições do negócio.

Por outro lado, para o desenvolvedor, torna-se um grande desafio para que tenha tempo de reagir rapidamente e apresentar as melhorias e correções levantadas para as próximas iterações através de incrementos no protótipo.

Tanto a modelagem de dados como a modelagem de negócios são etapas fundamentais na RAD para compreender corretamente como o negócio funciona, como deve ser formalizado e como os dados devem ser

estruturados de modo que sejam adequadamente armazenados e estejam disponíveis para serem usados quando houver necessidade.

Atividade 4

Como ocorre a evolução e estabilização do modelo de dados no desenvolvimento rápido de software (RAD)?

A O modelo de dados é totalmente definido no início do projeto e permanece inalterado durante todo o desenvolvimento.

B O modelo de dados é criado apenas na fase final do projeto, sem relação com as etapas anteriores do desenvolvimento.

C No RAD, o modelo de dados evolui conforme as entregas dos protótipos, ajustando-se às necessidades do negócio até atingir um estado estável. O modelo de dados é criado apenas na fase final do projeto, sem relação com as etapas anteriores do desenvolvimento.

D A estabilização do modelo de dados ocorre apenas após a implantação do sistema, sem influência das interações anteriores.

E A evolução do modelo de dados ocorre sem necessidade de ajustes ao longo do projeto, pois os requisitos são fixos.

A alternativa C está correta.

No desenvolvimento rápido de software (RAD), o modelo de dados evolui conforme as entregas dos protótipos, permitindo ajustes contínuos conforme as necessidades do negócio. Esse processo iterativo garante que o modelo seja refinado até atingir um estado estável.

As demais alternativas estão incorretas. A letra A sugere um modelo fixo desde o início; B e D desconsideram a evolução contínua do modelo durante o desenvolvimento; E assume que os requisitos são imutáveis, o que contradiz a abordagem flexível do RAD.

Design da interface com usuário

O design é um elemento necessário no desenvolvimento de sistemas, pois define como os usuários interagem com o software de forma intuitiva e eficiente. No contexto do desenvolvimento rápido de software (RAD), a criação de interfaces envolve a experimentação contínua por meio de protótipos, permitindo ajustes rápidos conforme as necessidades dos usuários.

Além de fatores visuais, como cores, formas e disposição dos elementos, a experiência do usuário (UX) também é considerada para garantir usabilidade e acessibilidade. Um bom design de interface reduz custos de manutenção, facilita o treinamento e melhora a aceitação do sistema final.

No vídeo, conheça a importância do design de interfaces no RAD, explorando protótipos interativos, usabilidade e boas práticas para criar interfaces intuitivas. Veja como testar e refinar a interface para garantir uma experiência eficiente e alinhada aos usuários.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

A criação de uma interface de usuário tem como meta ser um meio de interação para atender às necessidades de modo prático. Portanto, exige do projetista a capacidade de antecipar comportamentos e um conhecimento das diversas soluções existentes que tornem a construção do projeto viável. Isso significa que as limitações de tempo e recursos financeiros também são impostas para o projeto de design com o usuário.

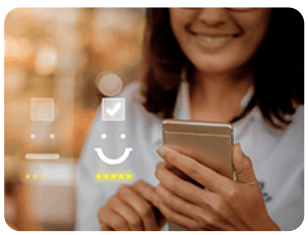
Confira a seguir algumas questões importantes sobre a criação de uma interface de usuário.

Forma, cor e espaço

Questões relacionadas à forma, à cor e ao espaço devem ser tratadas quando se considera um sistema que deve resolver problemas e que será usado por diversos usuários.

Sistema intuitivo

A interatividade do usuário com um sistema é dinâmica, pois trata de eventos que são acionados a partir das ações do usuário para atingir seus objetivos, logo é necessário que o sistema seja intuitivo e que, além disso, leve em consideração restrições que o usuário possa ter tanto físicas, como de ambiente tecnológico que o software vai operar.



Ferramentas e frameworks

Desenvolver um projeto de interface demanda ferramentas e frameworks que facilitem a visualização de protótipos que, após serem testados e aprovados, nortearão os desenvolvedores para a implementação delas. Esses protótipos permitem testar rapidamente conceitos de interação com o usuário, facilitando, assim, a melhor compreensão do sistema, uma vez que são expostos a situações reais, ainda que em um ambiente controlado.

Trabalhar na implementação de protótipos aumenta o engajamento dos usuários e da equipe de desenvolvimento. Isso ocorre porque as ideias relacionadas ao projeto passam a se concretizar em um ambiente interativo em que limitações são evidenciadas e necessidades de melhorias surgem naturalmente.

Quando se fala sobre design de um sistema, o pensamento mais comum é fazer associação com elementos de interface do usuário, normalmente, referenciados pelo acrônimo do inglês UI (User Interface), ou, ainda, GUI (Graphical User Interface), que tratam de componentes visuais com os quais o usuário vai interagir, como botões, janelas e barras de rolagem. Mas outro elemento muito importante que também compõe o projeto de interface de um sistema é a **experiência do usuário**, conhecido pelo acrônimo em inglês UX (User Experience).

Protótipo de interface

Visa à experimentação de conceitos, de modo que evolua para uma representação do sistema no usuário. O protótipo de interface é uma atividade muito importante, uma vez que vai guiar, concretamente, as escolhas dos desenvolvedores quando começarem a fazer a implementação do sistema.

Para cada componente escolhido, existe um conjunto de propriedades e de métodos associados que devem ser explicitamente configurados e tratados. Isso significa que o tempo de pessoas será alocado no estudo da configuração e no uso de componentes, além de como devem ser testados.

Comentário

Em especial, na metodologia RAD, a evolução do projeto de interface pela implementação de protótipos é esperada. Então, além da necessidade de profissionais bem treinados, é necessário ter à disposição ferramentas que facilitem esse processo pelo qual podem tornar a definição da interface algo concreto.

Desenvolver protótipo de interface do usuário ajuda a refinar os requisitos funcionais logo no início do projeto. A implicação disso tem consequências positivas. Veja!

Custo

Redução dos custos de manutenção.

Sistema

Treinamento e melhoria do sistema.

Aceitação

Aumento das chances de o usuário final aceitar a versão final.

De modo resumido, segundo Alavi (1984), entre os diversos benefícios de desenvolver protótipos de interface para o usuário, podemos citar os apresentados a seguir.

- A possibilidade de fazer testes para tratar de questões específicas do produto que dificilmente seriam respondidas por pesquisas.
- Um ambiente real para avaliar os conceitos de UI/UX.
- Um meio pelo qual todos os envolvidos no projeto podem usar como ponto de referência para se comunicarem.
- O engajamento dos usuários, pois eles podem fazer comentários sobre algo com que podem interagir, facilitando, assim, o trabalho posterior da equipe de desenvolvimento.
- Melhora na qualidade das especificações funcionais.
- Aumento das chances de o produto atender às demandas.
- Potencial de redução do custo total de desenvolvimento do produto.

Na RAD, o desenvolvimento de protótipo é um processo natural. No caso de protótipo de interface do usuário, a ideia é que os esforços empregados nos estágios iniciais podem reduzir bastante o custo do software final. Isso ocorre porque, depois de algumas interações, a interface se estabiliza e, portanto, as chances de ter que implementar muitas mudanças mais adiante no projeto diminuem.

Atenção

Apesar de todos os benefícios associados à implementação do protótipo de interface, é necessário ter alguns cuidados, como: ignorar limitações que se aplicam ao produto real no processo de prototipagem; criar expectativas irreais sobre o produto; e estabelecer limites no processo de prototipagem.

Questões relacionadas a licenças de produtos, contexto em que vai operar e viabilidade técnica, além de gerenciamento de recursos de tempo e financeiros sempre devem fazer parte da criação de protótipos. Na RAD, isso é essencial, pois a ideia é que o sistema evolua a partir de iterações e incrementos nos protótipos. Se, desde o início do projeto, houver muitos itens para serem tratados, o processo terá mais custos.

Atividade 1

Qual característica define o design da interface com o usuário no desenvolvimento rápido de software (RAD)?

A Interfaces estáticas que não podem ser alteradas após a primeira versão do sistema.

B Uso exclusivo de elementos gráficos predefinidos, sem possibilidade de adaptação ao usuário.

C Desenvolvimento iterativo com prototipagem contínua para refinar a experiência do usuário.

D Foco apenas na funcionalidade do sistema, sem considerar usabilidade ou acessibilidade.

E Desenvolvimento da interface apenas após a conclusão do backend, sem envolvimento dos usuários.

A alternativa C está correta.

No RAD, o design da interface do usuário é um processo iterativo que utiliza a prototipagem contínua para refinamento. Isso permite ajustes frequentes com base no feedback dos usuários e partes interessadas, garantindo que a experiência seja intuitiva e eficiente. Diferentemente de abordagens tradicionais, em que a interface é definida apenas após a implementação do backend, o RAD prioriza a evolução do design ao longo do desenvolvimento.

Alternativas como interfaces estáticas, uso exclusivo de elementos predefinidos e falta de consideração para usabilidade não condizem com a flexibilidade e adaptabilidade características desse modelo.

A necessidade da contextualização

A contextualização garante que os sistemas atendam às necessidades dos usuários de forma intuitiva e eficiente no desenvolvimento de interfaces. Com a tecnologia presente em diversas áreas, compreender o público-alvo e o ambiente de uso do sistema torna-se um diferencial.

No RAD, a contextualização permite ajustes contínuos na interface por meio de prototipagem e interação com as partes interessadas. Elementos como usabilidade, acessibilidade e alinhamento visual ao propósito do negócio devem ser considerados para criar interfaces estruturadas, proporcionando uma experiência fluida e satisfatória para os usuários.

No vídeo, entenda a contextualização no RAD e como a compreensão do público-alvo, do ambiente de uso e das necessidades do usuário melhora a eficiência do sistema. Por fim, confira a prototipagem iterativa e a adaptação contínua para criar interfaces intuitivas e alinhadas ao projeto.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

A sociedade evoluiu em muitos aspectos, inclusive no acesso à informação. Atualmente, os sistemas são usados para diversas aplicações, desde operações financeiras, educativas e de entretenimento. Além disso, os sistemas operam em diversas plataformas, como aplicações web e móveis.

O perfil do usuário também está mais abrangente, desde as pessoas que têm mais facilidade de usar tecnologias, as que ainda não estão habituadas e aquelas que possuem restrições visuais, por exemplo. Portanto, os desafios de projetar uma interface de usuário ficaram bem mais complexos no sentido de que são muitas perspectivas que devem ser observadas. Por outro lado, existem muitas ferramentas e frameworks disponíveis para apoiar o desenvolvimento de protótipos.

O primeiro ponto a ser considerado no desenvolvimento de um projeto de interface é entender a que público ele se destina. A compreensão de que o sistema será usado por outras pessoas e sobre em que contexto vai operar auxilia na criação de protótipos que atinjam mais rapidamente o objetivo esperado.

De um modo simplificado, um projeto de interface deve levar em consideração alguns pontos, como:

Previsibilidade

Envolve os elementos de interface, tais como botões, barras de rolagem e menus, os quais devem ter um comportamento previsível. Isso significa dar previsibilidade para os usuários, de modo que a interação com os componentes seja intuitiva.

Destaque

Envolve a aplicação de recursos que destaquem um componente no contexto da aplicação. Por exemplo, usar sombras para botões.

Simplicidade

Envolve a interface, que deve ser simples, ou seja, os elementos devem atender aos propósitos dos usuários.

Organização

Envolve organizar os elementos de forma que fiquem legíveis. Para isso, é importante haver uma hierarquia que os relacione, além da preocupação em fazer alinhamentos que priorizem o bem-estar do usuário.

Moderação
Envolve o uso de recursos de cores e contrastes, que deve ser moderado e sempre com o propósito de guiar o usuário para a realização de suas tarefas.

Fontes
Envolve tamanhos, estilo — negrito e itálico, por exemplo —, maiúsculas e distância entre letras. Tudo isso deve ser contextualizado com o objetivo de dar destaque para o usuário sobre o significado de uma mensagem. Por exemplo, um texto em letras maiúsculas pode ser usado para destacar o cuidado que o usuário deve ter ao realizar alguma ação específica no sistema, como excluir um registro.

Ações
Envolve, por exemplo, a quantidade de ações para executar tarefas, que deve ser a menor possível. O foco sempre deve ser o usuário, portanto, a interface deve ser um guia para progredir na execução de tarefas complexas.

Controles
Devem ter proximidade com os objetos que os usuários desejam controlar. Por exemplo, um botão para enviar um formulário deve estar próximo desse formulário.

Fornecinento de informações
Envolve a transparência das ações do sistema aos usuários, como é caso de mensagens de confirmação de envio.

Padrões de utilização
É apropriado, em alguns casos, aplicar padrões de uso do sistema para facilitar o trabalho do usuário. Por exemplo, preencher formulários previamente. Essa opção, no entanto, deve ser aplicada de forma ponderada, pois, apesar de ser bastante útil em muitas situações, pode ser utilizada por pessoas mal-intencionadas para obter indevidamente dados dos usuários.

Contextualização
Evolve o design dos componentes, que deve ser contextualizado com o posicionamento do negócio a que se destina. No caso de sistemas para empresas, é esperado que haja um estilo visual que deve ser mantido.

Próximos passos

Envolve a interface, que deve fornecer os passos seguintes de como os usuários devem proceder para navegar no sistema e atingir seus objetivos em qualquer contexto.

Na RAD, o desenvolvimento rápido de protótipo é enfatizado em detrimento do planejamento. Ele é feito por meio do fluxo de ideias que são exploradas a partir da interação com as partes interessadas e das soluções de problemas que são detectados durante o processo de prototipagem rápida.

Desde o início da RAD, o desenvolvimento de aplicações passou a ser utilizado em diversas situações, como aplicações web e móveis. Lembre-se: a RAD é adequada para situações em que os projetos não são complexos.

Exemplo

A RAD funciona bem para criação ou renovação de páginas de comércio eletrônico.

A escolha da RAD para implementar o design de interface de usuário deve satisfazer aos seguintes requisitos:

Escopo do projeto

Deve ser bem definido, para que seja viável visualizar como será a versão final do sistema e, assim, estimar os esforços e recursos necessários para o desenvolvimento do projeto.

Dados do projeto

Para aplicar a RAD, o projeto já deve ter informações sobre os dados, evitando, assim, a necessidade de aplicar processos de análise de dados que possam consumir bastante tempo e esforços difíceis de dimensionar.

Decisões do projeto

Apesar de as interações com as partes interessadas serem muito importantes, as decisões sobre o projeto também precisam ser rápidas. Em um projeto em que a hierarquia das decisões é muito forte, o uso da RAD é inadequado, especialmente na confecção de um protótipo de interface com o usuário.

Equipe de projeto

O dimensionamento das equipes deve ser pequeno. Parece simples, mas a implicação imediata é que os profissionais dessas equipes devem ter múltiplas habilidades, ou seja, além de trabalhar com programação visual, é esperado que o profissional tenha conhecimento sobre banco de dados, desenvolvimento de microsserviços e arquitetura de software.

Portanto, a escolha da RAD deve considerar todas as questões apresentadas. Logo, após a análise desses pontos, e se a RAD for a melhor opção, o processo deve seguir as seguintes etapas apresentadas:

Etapa 1

Escolher uma equipe com profissionais que serão capazes de desenvolver projetos de interface e que contribuam sobre a escolha dos componentes.

Etapa 2

Analisar as metas do projeto e entender o que precisa ser feito, como se encaixam no projeto e o que é necessário fazer para cumprir as metas.

Etapa 3

Criar um processo iterativo, onde podem ser desenvolvidos protótipos e testes de usabilidade até que seja atingido um ponto que se alinhe com os objetivos do projeto.

Atividade 2

Qual é a relação entre a contextualização e a eficiência e eficácia no desenvolvimento rápido de software (RAD)?

A A contextualização permite criar interfaces alinhadas às necessidades dos usuários, otimizando a usabilidade e reduzindo retrabalho, tornando o desenvolvimento mais eficiente e eficaz.

B A contextualização foca apenas a aparência do sistema, sem impacto na eficiência e eficácia do desenvolvimento.

C No RAD, a contextualização não influencia a eficiência, pois as interfaces são padronizadas e não precisam de adaptações ao público-alvo.

D A contextualização torna o desenvolvimento mais lento e burocrático, reduzindo a eficiência do processo.

E A eficiência e a eficácia no RAD dependem exclusivamente da velocidade do desenvolvimento, sem necessidade de considerar a contextualização.

A alternativa A está correta.

A contextualização no RAD é fundamental porque permite criar interfaces alinhadas às necessidades dos usuários, otimizando a usabilidade e reduzindo retrabalho. Isso torna o desenvolvimento mais eficiente.

As demais alternativas estão incorretas. Letra **B** foca apenas a aparência; **C** ignora a importância da adaptação ao público-alvo; **D** sugere que a contextualização torna o processo mais lento e burocrático; **E** considera apenas a velocidade do desenvolvimento, desconsiderando a importância de compreender o contexto do sistema.

Técnicas de prototipagem de interface com o usuário

A prototipagem de interface com o usuário permite testar e validar conceitos antes da implementação final. Diferentemente de outras metodologias, no RAD, o protótipo evolui continuamente até se tornar a versão definitiva do sistema. Técnicas como sketches, wireframes, mockups e protótipos interativos ajudam a visualizar a experiência do usuário, permitindo interfaces intuitivas e funcionais.

A escolha das ferramentas de prototipagem deve considerar fatores como integração, fidelidade, curva de aprendizado e colaboração. Um bom protótipo reduz erros, melhora a comunicação com clientes e acelera o processo de desenvolvimento.

Confira, no vídeo, as técnicas de prototipagem de interface no RAD, desde sketches e wireframes até mockups e protótipos interativos. Veja como essas abordagens refinam a experiência do usuário e ajudam a criar interfaces intuitivas e funcionais.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Em outras metodologias de desenvolvimento, faz-se o uso de protótipos também. No entanto, elas se diferenciam da RAD em relação ao objetivo do protótipo. Entenda essa diferença:

Demais metodologias	RAD
O protótipo é usado para provar conceitos e, ao longo do desenvolvimento, o sistema é desenvolvido em características.	O protótipo evoluirá para a versão final do sistema. Assim, é importante que a RAD trabalhe com o engajamento do cliente por meio da experiência do usuário.

Existem algumas técnicas para o desenvolvimento de protótipos de interface. Vamos conhecê-las?

Sketches

Trata-se de esboços de um projeto. Eles são fáceis de criar e úteis para verificar se os conceitos e requisitos foram totalmente compreendidos sem consumir muito esforço. O seu uso deve ser limitado ao início do projeto, pois é inadequado testar funcionalidades do sistema.

Wireframes

São muito úteis para fazer uma abordagem mais complexa para mapear a interface do usuário. Eles são mais detalhados do que os sketches, além disso, algumas ferramentas permitem adicionar componentes interativos. Desse modo, o usuário pode fazer navegações no protótipo sem a implementação das funcionalidades.

Mockup

Também conhecida como “maquete”, é usada para refletir as opções de design para a escolha de cores, layouts, tipografia, iconografia e aspectos visuais de navegação do produto.

Protótipos

São modelos funcionais de sistema. São construídos para tratar tanto das funcionalidades do produto, como também da aparência.

Preparar versões, ainda que simplificadas da interface com o usuário, ajuda os desenvolvedores e designers a implementarem suas ideias e, assim, fornece protótipos interativos para os clientes.

A escolha de uma ferramenta para desenvolver protótipos de interface com o usuário depende de diversos fatores, mas, de um modo simples, veja alguns itens a serem considerados.

Licença

A primeira preocupação de qualquer projeto deve ser a respeito dos direitos de uso de um software. O risco de iniciar um projeto em uma versão de testes de uma ferramenta que não poderá ser usada pela equipe posteriormente pode ser grande. Só faz sentido analisar os demais quesitos se esse item for satisfeito.

Integração

Algumas ferramentas de prototipagem geram código que pode ser usado diretamente por algumas linguagens de programação.

Fidelidade

Os protótipos na RAD são essenciais para a versão final do sistema, portanto, devem ser levadas em consideração quais as diferenças entre os elementos de interface do protótipo e o sistema esperado.

Curva de aprendizagem
A metodologia RAD exige ferramentas que apoiem o desenvolvimento rápido, portanto, as ferramentas devem ser intuitivas e fáceis de serem utilizadas e aprendidas pelos membros da equipe.

Facilidade de uso e conforto
As ferramentas de prototipagem devem ajudar a aumentar a produção com a redução de etapas necessárias para criar protótipos.

Compartilhamento
A ferramenta deve fornecer capacidade de colaboração entre várias pessoas.

A tarefa de projetar o design de um sistema é complexa. Portanto, as escolhas que devem ser tomadas logo no início do projeto RAD precisam considerar fatores que apoiem o desenvolvimento.

Atividade 3

Qual das alternativas descreve corretamente a principal diferença entre a prototipagem no RAD e em outras metodologias?

A No RAD, os protótipos são utilizados apenas para testar conceitos iniciais, sem evolução para a versão final.

B Em outras metodologias, os protótipos servem para demonstrar conceitos, enquanto no RAD os protótipos evoluem até a versão final do sistema.

C O RAD descarta completamente o uso de protótipos, pois o desenvolvimento ocorre sem iterações.

D A prototipagem no RAD foca exclusivamente a aparência do sistema, sem preocupação com funcionalidades.

E Em todas as metodologias, os protótipos são idênticos, não havendo diferença entre RAD e outras abordagens.

A alternativa B está correta.

A principal diferença entre a prototipagem no RAD e em outras metodologias é que, enquanto em outras abordagens os protótipos servem para demonstrar conceitos, no RAD eles evoluem até a versão final do sistema, permitindo iterações e melhorias constantes.

As demais alternativas estão incorretas. Letra **A** limita o uso de protótipos a conceitos iniciais. **C** afirma que o RAD descarta protótipos; **D** foca apenas a aparência, ignorando as funcionalidades; **E** sugere que não há diferença entre as metodologias, o que não é verdadeiro.

Considerações gerais sobre a implementação em Python

A implementação de um sistema utilizando o RAD com Python permite a entrega ágil de protótipos funcionais, possibilitando ajustes contínuos com base no feedback dos usuários. No RAD, a simplicidade e flexibilidade do Python facilitam a construção de aplicações iterativas e adaptáveis.

O projeto apresentado envolve um sistema simplificado de cadastro de notas de alunos, permitindo a inserção e consulta de informações acadêmicas, cálculo de médias e exibição da situação escolar. Com interface intuitiva e integração com planilhas Excel, esse exemplo ilustra como o RAD otimiza o desenvolvimento de aplicações de baixa complexidade com eficiência.

No vídeo, confira a implementação de um sistema RAD com Python, criando um cadastro de notas de alunos desde a modelagem até a integração com o Excel. Veja como o RAD permite entregas ágeis, testes iterativos e ajustes contínuos para aplicações eficientes.

> Conteúdo interativo
>
> Acesse a versão digital para assistir ao vídeo.

A entrega de protótipos para os usuários permite que eles possam interagir com o sistema e, portanto, munir os desenvolvedores com comentários que darão suporte para implementarem melhorias e fazer ajustes. Além disso, é uma metodologia adequada para projetos de baixa complexidade.

Ser uma metodologia voltada para entregas rápidas não quer dizer que o desenvolvedor não precisa saber programar bem. Ao contrário, o perfil dos profissionais de desenvolvimento na RAD exige que tenham domínio em linguagens de programação e ferramentas/frameworks, além de serem adaptáveis e motivados. Muitas linguagens de programação podem ser aplicadas para um projeto RAD. Aqui, o foco será na linguagem Python, uma escolha natural por diversos motivos, como simplicidade da sintaxe, disponibilidade de muitos pacotes, ampla aceitação no mercado, grande comunidade engajada na resolução de problemas e licença de software livre.

O exemplo escolhido para apresentar a RAD na prática é de um **sistema simples que faz o cadastro das informações pessoais básicas sobre um estudante e de suas notas**. Ao final do cadastro, o sistema exibe os **dados que foram cadastrados**, a **média das notas** e a **situação do estudante**, ou seja, se foi aprovado, ou reprovado, ou se está em recuperação.

A seguir, veremos as etapas de requisitos, modelagem de negócios e de dados, além da implementação do sistema serão tratadas.

Requisitos

Confira, a seguir, os requisitos do sistema.

Propósito
Apresentar uma descrição do Sistema Simplificado de Cadastro de Notas de Alunos. Essa descrição destina-se tanto aos usuários como aos desenvolvedores do sistema.

Escopo
Este software será um Sistema de Cadastro de Notas de Aluno, projetado para informar se os alunos estão aprovados, em recuperação, ou reprovados.

Além disso, os dados são gravados em um arquivo Excel para facilitar a integração com outros sistemas. O sistema atenderá às necessidades básicas da coordenação da escola para orientar os alunos enquanto permanece fácil de entender e usar.

Interfaces do sistema
O sistema foi projetado para operar no Windows.

Interfaces de usuário
A GUI do sistema possui botões, caixas de texto e grades, facilitando o controle por teclado e mouse.

Interfaces de software
O software permite importar e exportar dados em um documento estruturado do MS Excel via formato de dados XLSX.

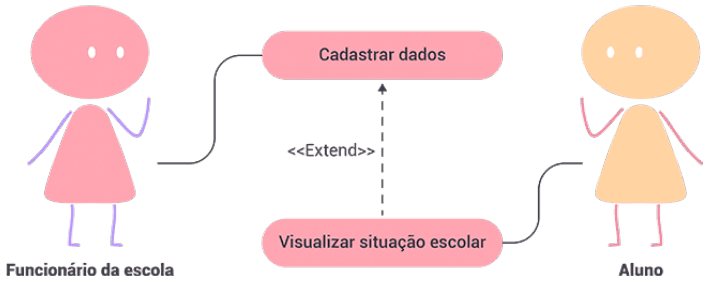
Especificação de requisitos funcionais
O usuário deve ser capaz de cadastrar o nome do aluno e suas notas e de visualizar os dados cadastrados do aluno, sua média e sua situação: aprovado, ou reprovado, ou em recuperação.

Os dados são gravados e lidos de uma planilha do Excel.

Modelagem de negócios

A escola RAD decidiu construir um novo aplicativo para facilitar a visualização da situação escolar dos seus alunos. Dados como nome e notas dos alunos serão cadastrados no sistema. Com isso, serão calculadas automaticamente as médias dos alunos e apresentada a sua situação escolar: aprovado, ou reprovado, ou em recuperação.

Na imagem a seguir, veja o diagrama de casos de uso do sistema de cadastro de notas e visualização da situação escolar dos alunos.



No diagrama, estão destacados os atores do sistema que são os alunos e funcionários da escola. Os funcionários vão cadastrar os dados dos alunos que, por sua vez, poderão visualizar as suas respectivas situações escolares.

Modelagem de dados

O sistema pretende ser apenas um facilitador de uma etapa do gerenciamento das notas dos alunos. Além disso, os dados serão armazenados em uma planilha Excel.

Na tabela, é apresentado o modelo de dados do sistema. Veja!

Campo	Aluno	Nota1	Nota2	Média	Situação
Tipo	Texto	Decimal	Decimal	Decimal	Texto

Tabela: Modelo de dados do sistema.

Kleber de Aguiar

Atividade 1

No início de um projeto, é necessário compreender quais são os requisitos que devem ser tratados para atender às demandas do negócio. Nesse sentido, selecione a opção correta relacionada à descrição dos requisitos de um sistema.

A Escopo: apresenta pormenores da implementação do sistema.

B Interfaces do sistema: descreve os componentes interativos do sistema.

C Propósito: apresenta uma descrição do sistema e para quem ele se destina.

D Interfaces de usuário: identifica com quais sistemas do usuário será feita a integração do software.

E Especificação de requisitos funcionais: define os requisitos técnicos de hardware necessários para a execução do sistema.

A alternativa C está correta.

A descrição do propósito de um sistema inclui uma explicação clara sobre o que o sistema faz e a quem se destina, facilitando a compreensão das necessidades do negócio.

As demais alternativas estão incorretas. Letra A foca os detalhes da implementação; B trata dos componentes interativos sem abranger o sistema como um todo; D confunde interfaces de usuário com integração entre sistemas; E se refere a requisitos técnicos, que não são parte da definição do propósito do sistema.

Design da interface com usuário

Neste estudo, você aprenderá a desenvolver uma interface gráfica em Python utilizando Tkinter para um sistema de gestão escolar. O sistema permitirá cadastrar alunos, inserir notas, calcular a média automaticamente e exibir a situação do aluno (aprovado, reprovado ou em recuperação). Além disso, a tabela de alunos já iniciará com alguns registros pré-carregados.

No vídeo, veja como criar uma interface gráfica em Python com Tkinter para um sistema de gestão escolar que cadastra alunos, insere notas, calcula médias automaticamente e exibe a situação acadêmica, com dados pré-carregados para fácil interação. Acompanhe o passo a passo e desenvolva seu próprio sistema!

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

A prática a seguir será estruturada em etapas progressivas, começando pela configuração do ambiente e criação da janela principal. Em seguida, você adicionará os componentes da interface e configurará a tabela de notas. Por fim, implementará a funcionalidade de cadastro dinâmico de alunos e testará o sistema em execução. Vamos começar!

Passo 1: Configurando o ambiente

1. **Instale o Tkinter** (se necessário):

   - No terminal/powershell, digite o seguinte comando:

   - pip install tkinter

2. **Importe os módulos necessários**:

```
python
```

```python
import tkinter as tk
from tkinter import ttk
```

Passo 2: Criando a janela principal

Inicialize a interface gráfica:

```
python
```

```python
janela = tk.Tk()
janela.title("Sistema de Gestão Escolar")
janela.geometry("600x400")
        ```
```

**Passo 3: Criando os componentes da interface**

1. **Rótulos** (textos explicativos):

```
python
```

```python
tk.Label(janela, text="Nome do Aluno:").pack()
```

2. **Caixas de texto** (para entrada de dados):

```
python
```

```python
entrada_nome = tk.Entry(janela)
entrada_nome.pack()
entrada_nota1 = tk.Entry(janela)
entrada_nota1.pack()
entrada_nota2 = tk.Entry(janela)
entrada_nota2.pack()
```

Passo 4: Criando a tabela e adicionando dados iniciais

1. **Criar uma tabela para visualização das notas**:
```

python

```python
tabela = ttk.Treeview(janela, columns=("Nome", "Nota1", "Nota2", "Média", "Situação"),
show="headings")
tabela.heading("Nome", text="Nome do Aluno")
tabela.heading("Nota1", text="Nota 1")
tabela.heading("Nota2", text="Nota 2")
tabela.heading("Média", text="Média")
tabela.heading("Situação", text="Situação")
tabela.pack()
```

2. **Adicionar barra de rolagem para a tabela:**

python

```python
scrollbar = ttk.Scrollbar(janela, orient="vertical", command=tabela.yview)
tabela.configure(yscrollcommand=scrollbar.set)
scrollbar.pack(side="right", fill="y")
```

3. **Carregar alguns alunos no início**:

python

```python
alunos_iniciais = [
    ("Alice", 8.5, 7.0),
    ("Bruno", 5.0, 6.0),
    ("Carlos", 3.5, 4.0),
    ("Daniela", 9.0, 9.5)
]

for aluno in alunos_iniciais:
    nome, nota1, nota2 = aluno
    media = (nota1 + nota2) / 2
    situacao = "Aprovado" if media >= 7 else "Recuperação" if media >= 5 else "Reprovado"
    tabela.insert("", "end", values=(nome, nota1, nota2, f"{media:.2f}", situacao))
```

Passo 5: Criando a função para cadastrar alunos

Criar um botão para adicionar alunos:

```
python
```python
def cadastrar_aluno():
    nome = entrada_nome.get()
    nota1 = float(entrada_nota1.get())
    nota2 = float(entrada_nota2.get())
    media = (nota1 + nota2) / 2
    situacao = "Aprovado" if media >= 7 else "Recuperação" if media >= 5 else "Reprovado"

    tabela.insert("", "end", values=(nome, nota1, nota2, f"{media:.2f}", situacao))

tk.Button(janela, text="Cadastrar", command=cadastrar_aluno).pack()
        ```
```

Passo 6: Executando o sistema

Execute a aplicação:

```
python
```python
janela.mainloop()
```
```

Atividade 2

Se o desenvolvedor alterar a seguinte linha:

tabela.insert("", "end", values=(nome, nota1, nota2, f"{media:.2f}", situacao))

para:

tabela.insert("", 0, values=(nome, nota1, nota2, f"{media:.2f}", situacao))

O que acontecerá com a exibição dos dados na tabela?

A Os novos alunos serão adicionados no início da tabela em vez de no final.

B O sistema apresentará um erro, pois 0 não é um identificador válido para a inserção de dados.

C A tabela ficará travada e não permitirá mais inserções de novos registros.

D Os novos alunos serão adicionados ao final da tabela, sem alterações visíveis.

E Os dados cadastrados anteriormente serão substituídos pelos novos registros.

A alternativa A está correta.

Alterar "end" para 0 faz com que os novos registros sejam inseridos no início da tabela, empurrando os registros anteriores para baixo. Isso altera a ordem de exibição dos dados, mas o funcionamento geral do sistema permanece inalterado.

Detalhes de implementação

Este estudo guiará você na implementação de um sistema de cadastro e visualização da situação escolar de alunos utilizando Python, Tkinter e Pandas.

O sistema permitirá o cadastro de alunos, entrada de notas, cálculo automático da média e exibição da situação acadêmica. Além disso, os dados serão armazenados e recuperados de uma planilha do Excel.

Acompanhe, no vídeo, um roteiro prático para criar um sistema de cadastro de alunos com Python, Tkinter e Pandas. Aprenda como desenvolver uma interface gráfica interativa, calcular médias automaticamente e salvar os dados no Excel, integrando GUI e manipulação de dados de forma eficiente.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

O processo será dividido em etapas estruturadas, começando pela configuração do ambiente e criação da interface gráfica. Em seguida, você implementará os componentes da interface e programará a lógica para cálculo da média dos alunos. Por fim, adicionará funcionalidades para salvar e carregar os dados, garantindo a continuidade das informações. Vamos começar!

Passo 1: Configurando o ambiente

1. **Instale os pacotes necessários**:

````
python

   ```python
   pip install tkinter pandas openpyxl
   ```
````

2. **Importe as bibliotecas no código**:

```
python
```python
   import tkinter as tk
   from tkinter import ttk
   import pandas as pd
   ```
```

Passo 2: Criando a interface gráfica

1. **Inicialize a janela principal do Tkinter**:

```
python
   ```python
   janela = tk.Tk()
   janela.title("Sistema de Cadastro de Alunos")
   janela.geometry("820x600")
   ```
```

2. **Adicione os componentes gráficos**:

```
python
   ```python
   lblNome = tk.Label(janela, text="Nome do Aluno:")
   lblNota1 = tk.Label(janela, text="Nota 1")
   lblNota2 = tk.Label(janela, text="Nota 2")

   txtNome = tk.Entry(janela, bd=3)
   txtNota1 = tk.Entry(janela)
   txtNota2 = tk.Entry(janela)
   ```
```

Passo 3: Criando a tabela de visualização

1. **Crie a tabela com Treeview**:

```
python
   ```python
   colunas = ("Aluno", "Nota1", "Nota2", "Média", "Situação")
   treeMedias = ttk.Treeview(janela, columns=colunas, show="headings")

   for coluna in colunas:
       treeMedias.heading(coluna, text=coluna)
       treeMedias.column(coluna, width=100)

   treeMedias.pack(padx=10, pady=10)
   ```
```

2. **Adicione uma barra de rolagem**:

```
python
```

```python
scrollbar = ttk.Scrollbar(janela, orient="vertical", command=treeMedias.yview)
treeMedias.configure(yscrollcommand=scrollbar.set)
scrollbar.pack(side="right", fill="y")
```

Passo 4: Criando a função de cadastro e cálculo da média

1. **Defina a lógica de cálculo da média e situação do aluno**:

```
python
```

```python
def verificar_situacao(nota1, nota2):
    media = (nota1 + nota2) / 2
    if media >= 7.0:
        situacao = "Aprovado"
    elif media >= 5.0:
        situacao = "Em Recuperação"
    else:
        situacao = "Reprovado"
    return media, situacao
```

2. **Crie a função para cadastrar alunos**:

```
python
```

```python
def cadastrar_aluno():
    try:
        nome = txtNome.get()
        nota1 = float(txtNota1.get())
        nota2 = float(txtNota2.get())

        media, situacao = verificar_situacao(nota1, nota2)

        treeMedias.insert("", "end", values=(nome, nota1, nota2, f"{media:.2f}",
situacao))
        salvar_dados()
    except ValueError:
        print("Erro: Digite valores numéricos válidos.")
    finally:
        txtNome.delete(0, "end")
        txtNota1.delete(0, "end")
        txtNota2.delete(0, "end")
```

Passo 5: Criando a função para salvar e carregar dados

1. **Salve os dados cadastrados em uma planilha do Excel**:

```
python
```python
def salvar_dados():
    dados = []
    for line in treeMedias.get_children():
        valores = treeMedias.item(line)["values"]
        dados.append(valores)

    df = pd.DataFrame(data=dados, columns=colunas)
    df.to_excel("planilhaAlunos.xlsx", index=False, engine="openpyxl")
    print("Dados salvos com sucesso.")
```
```

2. **Carregue dados existentes no início do programa**:

```
python
```python
def carregar_dados_iniciais():
    try:
        df = pd.read_excel("planilhaAlunos.xlsx")
        for _, row in df.iterrows():
            treeMedias.insert("", "end", values=(row["Aluno"], row["Nota1"],
row["Nota2"], row["Média"], row["Situação"]))
    except FileNotFoundError:
        print("Nenhum dado encontrado.")
```
```

Passo 6: Finalizando a interface e rodando o sistema

1. **Adicione o botão de cadastro**:

```
python
```python
btnCadastrar = tk.Button(janela, text="Cadastrar", command=cadastrar_aluno)
btnCadastrar.pack()
```
```

2. **Chame a função de carregamento de dados no início**:

```
python
```python
carregar_dados_iniciais()
janela.mainloop()
```
```

Atividade 3

Se a linha a seguir no código:

df.to_excel("planilhaAlunos.xlsx", index=False, engine="openpyxl")

fosse alterada para:

df.to_excel("planilhaAlunos.xlsx", index=True, engine="openpyxl")

O que aconteceria?

A O sistema continuaria funcionando normalmente, mas a planilha armazenaria um índice numérico adicional em cada linha.

B O sistema apresentaria um erro ao tentar salvar os dados, pois a opção index=True não é permitida no to_excel().

C A planilha seria sobrescrita com valores incorretos, tornando os dados do aluno ilegíveis.

D O sistema deixaria de salvar os dados na planilha, exigindo uma reconfiguração da função de exportação.

E A função to_excel() perderia compatibilidade com o Pandas, impossibilitando a execução do programa.

A alternativa A está correta.

Alterar index=False para index=True faz com que o Pandas inclua uma coluna adicional com um índice numérico na planilha, sem afetar o funcionamento do código, mas modificando a estrutura do arquivo exportado.

Evoluindo o sistema

A implementação de uma **tela de login** no sistema de notas oferece controle seguro e organizado do acesso aos dados acadêmicos. Cada usuário terá um nível de permissão específico: **alunos** visualizam apenas suas notas, enquanto **professores** têm acesso total para adicionar ou excluir registros.

Utilizamos um **dicionário em Python** para armazenar usuários e senhas, com autenticação ocorrendo antes do acesso à interface principal. O sistema carrega automaticamente as notas de um arquivo Excel, permitindo visualização e manipulação em tempo real. Essa abordagem garante segurança, organização e usabilidade, assegurando o acesso correto às informações e evitando modificações indevidas.

No vídeo, veja um roteiro prático para uma evolução do sistema implementado anteriormente, adicionando a ele uma tela de login, que permite acesso customizado à base de dados, conforme o tipo de usuário do sistema.

> Conteúdo interativo
>
> Acesse a versão digital para assistir ao vídeo.

Este roteiro guiará a implementação de uma **tela de login** para que alunos visualizem apenas suas notas, enquanto professores terão acesso completo aos dados e poderão adicionar/excluir registros. As credenciais serão armazenadas em um dicionário.

Passo 1: Criando a estrutura de login

1. **Armazene as credenciais dos usuários em um dicionário**:

```
python

usuarios = {
    "professor": {"senha": "1234", "tipo": "professor"},
    "aluno1": {"senha": "1234", "tipo": "aluno"},
    "aluno2": {"senha": "1234", "tipo": "aluno"},
}
```

2. **Crie uma nova janela para o login**:

```
python

```python
def abrir_tela_login():
    login_win = tk.Toplevel()
    login_win.title("Login")
    login_win.geometry("300x200")

    tk.Label(login_win, text="Usuário:").pack()
    entry_usuario = tk.Entry(login_win)
    entry_usuario.pack()

    tk.Label(login_win, text="Senha:").pack()
    entry_senha = tk.Entry(login_win, show="*")
    entry_senha.pack()

    def validar_login():
        usuario = entry_usuario.get()
        senha = entry_senha.get()

        if usuario in usuarios and usuarios[usuario]["senha"] == senha:
            tipo_usuario = usuarios[usuario]["tipo"]
            login_win.destroy()
            iniciar_sistema(tipo_usuario, usuario)
        else:
            tk.messagebox.showerror("Erro", "Credenciais inválidas!")

    tk.Button(login_win, text="Entrar", command=validar_login).pack()
```
```

3. **Inicie o sistema ocultando a janela principal até que o login seja validado**:

````
python

```python
janela = tk.Tk()
janela.withdraw()  # Oculta a janela principal
abrir_tela_login()
janela.mainloop()
```
````

Passo 2: Carregando e exibindo dados de forma restrita

1. **Modifique o carregamento de dados para exibir apenas as notas do aluno logado ou todas as notas para o professor**:

````
python

```python
def carregar_dados(usuario, tipo_usuario):
    try:
        df = pd.read_excel("planilhaAlunos.xlsx")

        treeMedias.delete(*treeMedias.get_children())  # Limpa a tabela antes de atualizar

        if tipo_usuario == "professor":
            for _, row in df.iterrows():
                treeMedias.insert("", "end", values=(row["Aluno"], row["Nota1"],
row["Nota2"], row["Média"], row["Situação"]))
        else:
            df_aluno = df[df["Aluno"] == usuario]  # Filtra os dados do aluno logado
            for _, row in df_aluno.iterrows():
                treeMedias.insert("", "end", values=(row["Aluno"], row["Nota1"],
row["Nota2"], row["Média"], row["Situação"]))
    except FileNotFoundError:
        print("Nenhum dado encontrado.")
```
````

2. **Chame essa função ao iniciar o sistema**:

````
python

```python
def iniciar_sistema(tipo_usuario, usuario):
    janela.deiconify()  # Exibe a janela principal
    carregar_dados(usuario, tipo_usuario)
```
````

Passo 3: Restringindo ações para professores

1. **Permitir apenas professores adicionarem alunos**:

python

```python
def cadastrar_aluno():
    if tipo_usuario != "professor":
        tk.messagebox.showwarning("Acesso Negado", "Somente professores podem adicionar
alunos.")
        return

    try:
        nome = txtNome.get()
        nota1 = float(txtNota1.get())
        nota2 = float(txtNota2.get())

        media, situacao = verificar_situacao(nota1, nota2)

        treeMedias.insert("", "end", values=(nome, nota1, nota2, f"{media:.2f}",
situacao))
        salvar_dados()
    except ValueError:
        print("Erro: Digite valores numéricos válidos.")
    finally:
        txtNome.delete(0, "end")
        txtNota1.delete(0, "end")
        txtNota2.delete(0, "end")
```

2. **Permitir apenas professores excluírem registros**:

python

```python
def excluir_aluno():
    if tipo_usuario != "professor":
        tk.messagebox.showwarning("Acesso Negado", "Somente professores podem excluir
registros.")
        return


    selected_item = treeMedias.selection()
    if not selected_item:
        tk.messagebox.showerror("Erro", "Nenhum aluno selecionado para exclusão.")
        return

    treeMedias.delete(selected_item)
salvar_dados()
```

3. **Adicionar um botão de exclusão apenas para professores**:

python

```python
btnExcluir = tk.Button(janela, text="Excluir Aluno", command=excluir_aluno)
if tipo_usuario == "professor":
    btnExcluir.pack()
```

Passo 4: Testando o sistema

1. **Testar logins diferentes**:

- **Usuário: professor / Senha: 1234** → Deve acessar todas as funções.
- **Usuário: aluno1 / Senha: 1234** → Deve visualizar apenas suas notas.
- **Usuário: aluno2 / Senha: 1234** → Deve visualizar apenas suas notas.
- **Usuário ou senha incorretos** → Deve exibir erro de credenciais inválidas.

2. **Verificar se**:

- Alunos conseguem ver **apenas suas próprias notas**.
- Professores podem visualizar **todos os alunos e editar registros**.
- Apenas professores podem **adicionar ou excluir registros**.

Atividade 4

O que aconteceria se a linha:

tree.delete(*tree.get_children())

fosse removida do código?

A O sistema continuará funcionando normalmente, sem nenhuma alteração visível.

B Os dados antigos permanecerão na interface toda vez que novos dados forem carregados, resultando em informações duplicadas na tabela.

C O sistema apresentará um erro ao tentar exibir os dados, pois a remoção dessa linha impede a atualização da interface.

D Apenas os professores serão afetados, pois os alunos continuarão vendo apenas seus próprios dados corretamente.

E A função de carregamento dos dados deixará de funcionar completamente, impossibilitando a exibição das notas.

A alternativa B está correta.

A linha tree.delete(*tree.get_children()) limpa os dados da tabela antes de carregá-los novamente. Se for removida, os registros antigos permanecerão na interface toda vez que novos dados forem adicionados, resultando na exibição duplicada de informações.

Considerações finais

O que você aprendeu neste conteúdo?

- Engenharia de requisitos.
- Modelagem de negócios.
- Modelagem de dados.
- Técnicas de prototipagem.
- Design de interface.

Podcast

Podcast

Acompanhe um bate-papo sobre a aplicação da metodologia de desenvolvimento rápido de software (RAD), abordando os aspectos essenciais de suas fases, como a análise de requisitos, a modelagem e o projeto de interface. Não perca!

Conteúdo interativo

Acesse a versão digital para ouvir o áudio.

Explore +

Para saber mais sobre os assuntos tratados neste conteúdo, visite os seguintes portais na internet:

- O site oficial do Python. Leia sobre conceitos de linguagem e de aplicações GUI e faça downloads de diversos pacotes para desenvolvimento rápido.
- O site oficial de documentação da Microsoft. Procure por normalização de tabelas de bancos de dados.
- O site oficial do Planalto do Governo Federal. Procure pela Lei nº 13.709 e pela Lei Geral de Proteção de Dados.

Referências

ALAVI, M. **An assessment of the prototyping approach to information systems development**. *In:* Communications of the ACM, 27, 556-563, 1984.

BERGER, H.; BEYNON-DAVIES, P. **The utility of rapid application development in large-scale, complex projects**. Information Systems Journal, 2009, 19(6), 549– 570.

DOCS.PYTHON. **Python 3.8.5 Documentation**. Consultado na internet em: 1 out. 2020.

DOCS.PYTHON. **Tkinter** — Python interface to Tcl/Tk. Consultado na internet em: 1 out. 2020.

KERR, J.; HUNTER, R. **Inside RAD**: How to Build Fully Functional Computer Systems in 90 Days or Less. New York: McGraw-Hill, 1994.

MARTIN, J. **Rapid Application Development**. USA: Macmillan, 1991.

PRESSMAN, R. **Engenharia de Software**: Uma abordagem Profissional. 7. ed. Porto Alegre: Bookman, 2011.

SPYDER-IDE. **Spyder**. Consultado na internet em: 1 out. 2020.