

APLICANDO

DESCRIÇÃO

Aplicação da metodologia de desenvolvimento rápido de software (RAD) com descrição do levantamento de requisitos, modelagem de negócios e de dados, Design de interface com o usuário e uma aplicação prática com o uso da linguagem de programação Python.

PROPÓSITO

Compreender a aplicação da metodologia RAD para o desenvolvimento de um sistema.

PREPARAÇÃO

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

OBJETIVOS

MÓDULO 1

Descrever as etapas para tratamento dos requisitos de um sistema na metodologia de desenvolvimento rápido de software (RAD)

MÓDULO 2

Descrever as modelagens de negócios e de dados da RAD

MÓDULO 3

Definir o design de interface com o usuário na RAD

MÓDULO 4

Esquematizar uma aplicação RAD implementada em Python

INTRODUÇÃO

A metodologia de desenvolvimento rápido de software (RAD) é caracterizada por um processo evolutivo em que usuários e desenvolvedores são engajados através de entregas de versões funcionais do sistema que, apesar de incompletas, permitem que a colaboração seja otimizada.

Os usuários podem operar essas versões e fazer comentários, críticas e sugestões que auxiliam bastante os desenvolvedores a direcionar seus esforços para atender, de fato, às necessidades do cliente. Essas versões parciais são chamadas de protótipos e a ideia é que, ao longo do processo, avancem no sentido de se tornarem a versão final do sistema.

Aspectos como custo-benefício, uso efetivo de tempo, alcançar a satisfação do cliente e controlar riscos são abordados pela RAD através da rápida verificação de inconsistências e alinhamento de entendimento.

Ao longo do texto, abordaremos as etapas de requisitos de um sistema, as modelagens de negócios e de dados e o design da interface com o usuário. No último módulo, apresentaremos uma aplicação RAD simples desenvolvida com o uso da linguagem de programação Python.

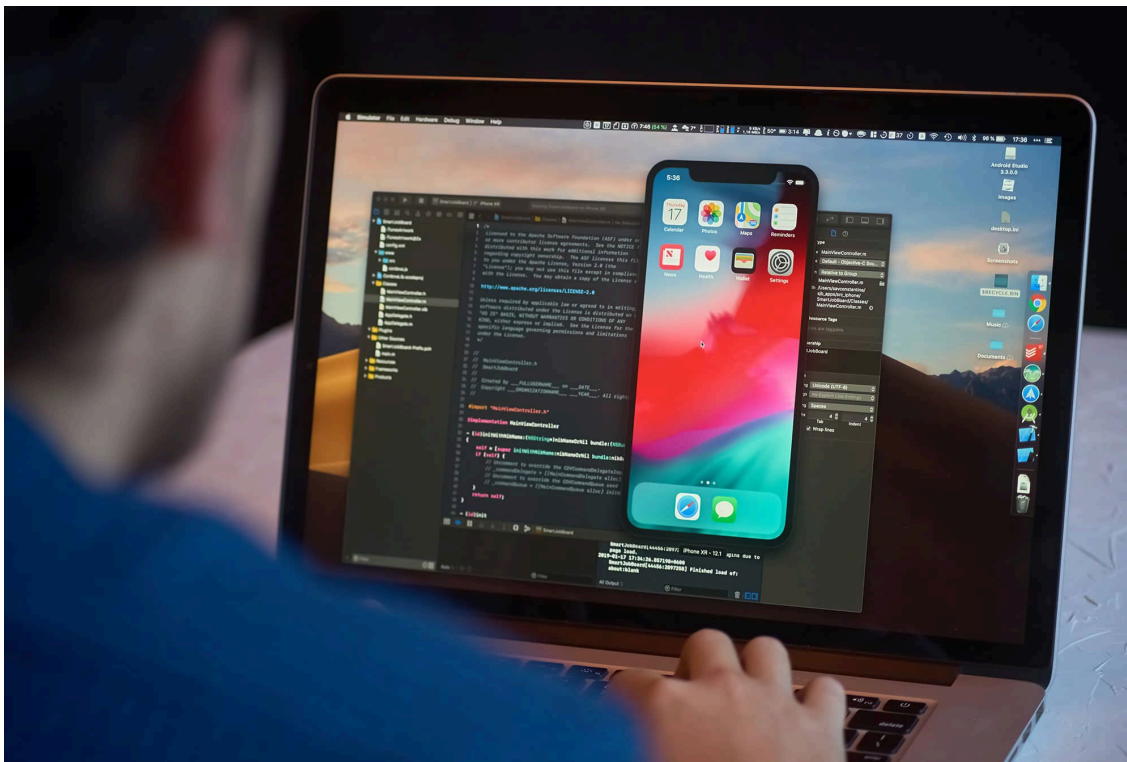
MÓDULO 1

- ⦿ Descrever as etapas para tratamento dos requisitos de um sistema na metodologia de desenvolvimento rápido de software (RAD)

REQUISITOS

CONTEXTUALIZAÇÃO

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.



Fonte: Shutterstock

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

COMENTÁRIO

Além disso, o progresso do desenvolvimento 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: Maior eficiência no uso do tempo das partes envolvidas, comunicação mais produtiva e 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 possa usá-lo de forma mais eficiente para a implementação do sistema. Segundo Martin (1991), a RAD possui quatro fases distintas, que são:

Quatro fases da RAD

Fase de planejamento de requisitos:	Trata das necessidades de negócios, escopo do projeto, restrições e requisitos do sistema.
Fase de design do usuário:	Nessa fase, são desenvolvidos modelos e protótipos para representar todos os processos, entradas e saídas do sistema.
Fase de construção:	É nesta fase que os protótipos são desenvolvidos.
Fase de transição:	Aqui, são feitos processamento de dados, testes, mudança para o novo sistema e treinamento do usuário.

☐ **Atenção!** Para visualização completa da tabela utilize a rolagem horizontal

A seguir, trataremos detalhadamente a fase de planejamento de requisitos.

FASE DE PLANEJAMENTO DE REQUISITOS

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 desse assunto 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.

REQUISITOS

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.

A Engenharia de Requisitos consiste em cinco atividades principais (vide Pressman, 2011):

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: O que foi implementado deve ser o que foi solicitado.

Gerenciamento: Consiste na priorização dos requisitos.

A seguir, cada uma dessas atividades será detalhada.

ELICITAÇÃO DE REQUISITOS

A elicitação de requisitos, ou ainda, 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 se destina e quais os pormenores da operação, a fim de obter uma melhor compreensão do sistema a ser desenvolvido.

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:



Shutterstock

FECHADAS

As partes interessadas respondem um conjunto predefinido de perguntas.





autor/shutterstock

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. Entre essas técnicas, estão:

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.

OS 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 depois de esta ter sido concluída.

OBSERVAÇÃO E ANÁLISE SOCIAL:

Aqui, um analista observa os usuários enquanto trabalham e faz anotações. A principal vantagem desta técnica é mostrar o modo como os usuários realmente trabalham e não uma descrição idealizada de como devem trabalhar.

GRUPOS FOCAIS:

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

BRAINSTORMING: A

Ou tempestade cerebral, é 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 (isto quer dizer que nenhum serviço ou restrição estará ausente) até a viabilidade dos requisitos. Portanto, a análise 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**.

SAIBA MAIS

As sessões JAD — Acrônimo do inglês para Joint Application Development — são oficinas de trabalho em que os desenvolvedores e clientes discutem sobre o sistema. 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. Portanto, é nessa etapa 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.

MODELO TRADICIONAL DE ENGENHARIA DE REQUISITOS APLICADO PARA DESENVOLVIMENTO RÁPIDO DE SOFTWARE (RAD)

Segundo Pressman (2011), o modelo tradicional de engenharia de requisitos para desenvolvimento rápido de software tem as seguintes etapas:



Autor/Shutterstock

Alguns dos problemas para aplicar técnicas de engenharia de requisitos dos modelos tradicionais para a RAD são:

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.

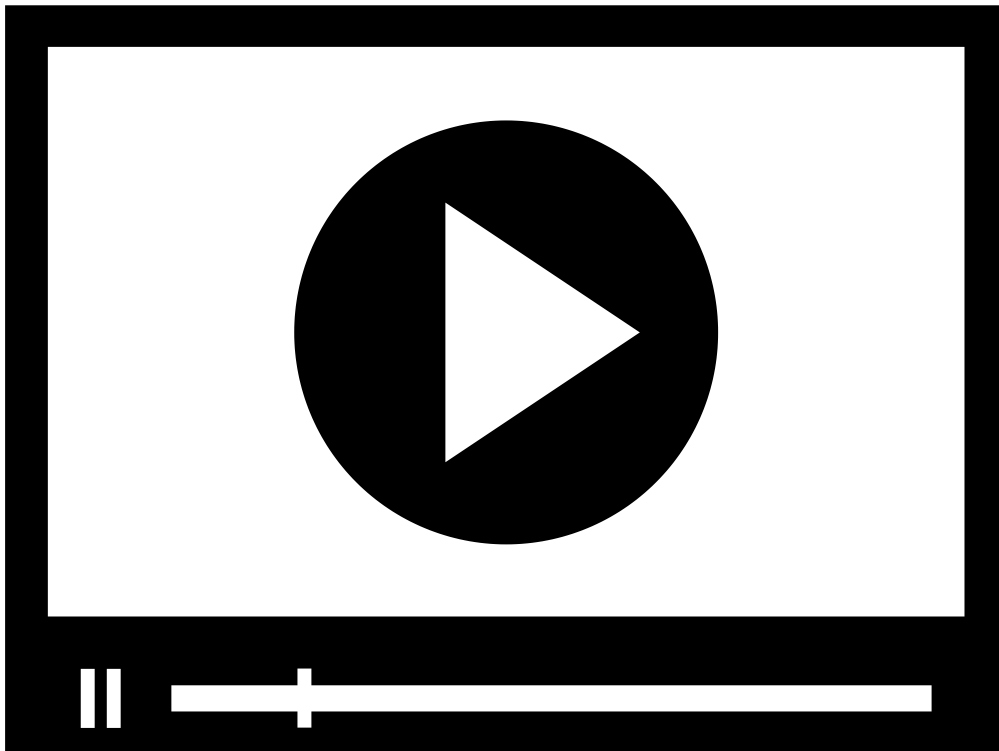
O modelo tradicional não trata de forma específica dos conflitos que podem ocorrer com a implementação dos requisitos ao longo do projeto, como é o caso da RAD em que o projeto vai incorporando sugestões dos usuários ao longo das iterações.

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.

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.

MODELO DE ENGENHARIA DE REQUISITOS PARA RAD



OFICINAS JAD NA METODOLOGIA RAD: CONCEITOS E PRÁTICA

Neste vídeo, desvendamos o poder das Oficinas JAD (Joint Application Development) na Metodologia RAD. Explore o que são, por que são essenciais e como conduzi-las de forma eficaz para acelerar o desenvolvimento de software.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



A elicitação de requisitos é a principal etapa na RAD. Isto ocorre porque o protótipo depende dela. A técnica mais utilizada para levantamento de requisitos na RAD é a JAD (**Joint Application Development**). Nas oficinas JAD, as partes interessadas discutem o sistema e definem seus requisitos. São sessões em que as partes interessadas — usuários e desenvolvedores — debatem sobre as necessidades do negócio. Isto 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.

Ao final da extração dos requisitos pela JAD, é necessário analisá-los. Isso é necessário para garantir sua integridade e consistência. Uma das técnicas comumente utilizadas para análise de requisitos é o **gerenciamento de conflitos**.

MODAL

A gestão de conflitos 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.

Agora, depois da remoção dos conflitos entre os requisitos, é necessário fazer a priorização deles para poder partir para o desenvolvimento do protótipo. Seguindo uma sequência lógica, no protótipo, primeiro desenvolve-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.

ETAPA 1

Na classificação por cartões, todas as funcionalidades do sistema são escritas em cartões individuais. Em seguida, os cartões são dados para todos os participantes das oficinas de trabalho.



ETAPA 2

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. Em seguida, faz-se a análise dos graus de todos os cartões para analisar os dados.



ETAPA 3

Depois de analisados, é feita 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.

Com a priorização dos requisitos definida, 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.

Agora, sim, chega-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 **desenvolvimento** é feito a partir da priorização dos requisitos, que significa que, inicialmente, as principais funcionalidades do sistema devem ser implementadas.



Shutterstock

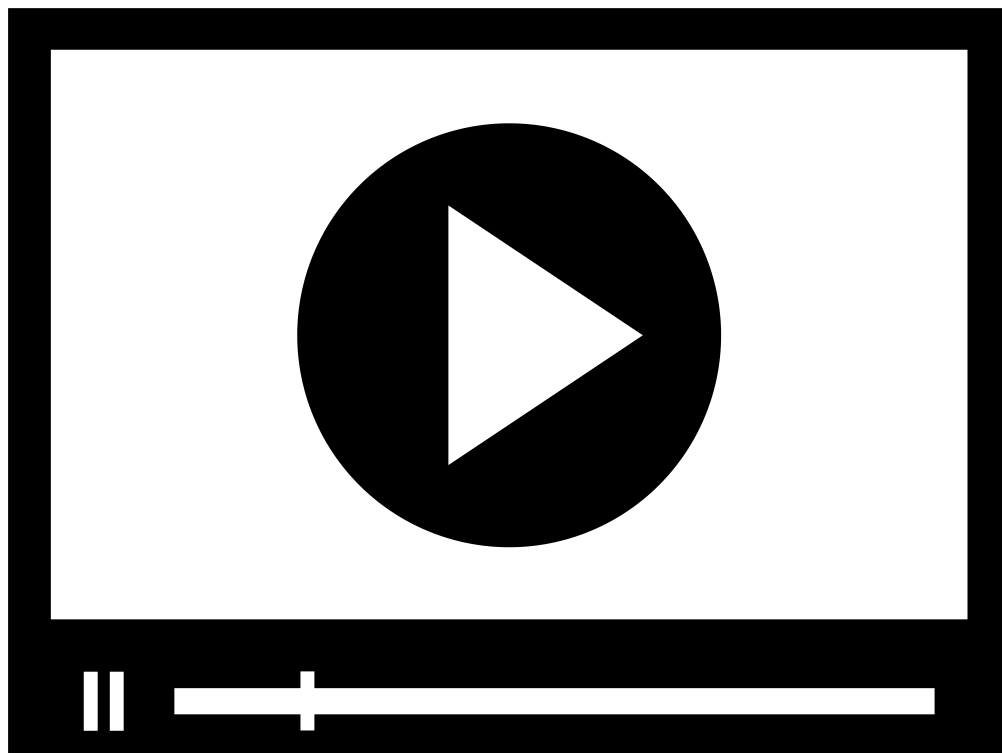
Terminado o protótipo, ele é apresentado para ser validado pelo cliente. Essa é a etapa mais importante em qualquer ciclo de vida de desenvolvimento, pois todo sistema tem um propósito e, quem o avalia, é o usuário. Na RAD, a validação do protótipo é muito importante para direcionar quais serão as ações 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 a RAD, 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

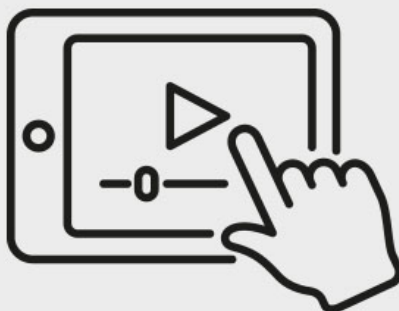
desvantagens. Seu uso depende de uma equipe de trabalho com diversas habilidades e motivada que se adéque rapidamente a mudanças ao longo do projeto.



VÍDEO

Analisando os requisitos de um sistema

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



VERIFICANDO O APRENDIZADO

1. A RAD É UMA METODOLOGIA EVOLUTIVA BASEADA EM ITERAÇÕES INCREMENTAIS COM FASES BEM DEFINIDAS. SOBRE AS FASES DA RAD, SELECIONE A OPÇÃO CORRETA QUE DESCREVE O DESENVOLVIMENTO DO PROTÓTIPO DE DESIGN DE INTERFACE COM O USUÁRIO:

- A) Preocupa-se com o desenvolvimento de componentes interativos que atendam aos requisitos do sistema.
- B) Tem foco na aparência do sistema, pois, desse modo, é mais fácil o usuário operá-lo.
- C) Esta fase não está relacionada, necessariamente, com a confecção de um protótipo com componentes interativos.
- D) Faz parte da primeira fase da RAD.

2. A RAD TEM COMO META ENTREGAR SISTEMAS RAPIDAMENTE, MAS SÓ FAZ SENTIDO ENTREGAR RAPIDAMENTE UM SISTEMA QUE TENHA UMA BOA QUALIDADE, CASO CONTRÁRIO SERÃO NECESSÁRIAS NOVAS ITERAÇÕES PARA FAZER CORREÇÕES E AJUSTES. EM RELAÇÃO À METODOLOGIA RAD, SELECIONE A OPÇÃO CORRETA QUE INDIQUE COMO EVITAR RETRABALHO:

- A) Desenvolvendo protótipos.
- B) Através da combinação de processos bem consolidados das metodologias tradicionais com os métodos ágeis.
- C) Aplicando um processo exaustivo de documentação logo no início do projeto, a fim de minimizar imprevistos ao longo do desenvolvimento.
- D) Através da colaboração contínua das partes interessadas ao longo do projeto.

GABARITO

1. A RAD é uma metodologia evolutiva baseada em iterações incrementais com fases bem definidas. Sobre as fases da RAD, selecione a opção correta que descreve o desenvolvimento do protótipo de design de interface com o usuário:

A alternativa "A " está correta.

O protótipo de design gráfico faz parte da segunda fase da RAD. É muito importante para que o usuário possa interagir com o sistema e verificar se os requisitos são atendidos.

2. A RAD tem como meta entregar sistemas rapidamente, mas só faz sentido entregar rapidamente um sistema que tenha uma boa qualidade, caso contrário serão necessárias novas iterações para fazer correções e ajustes. Em relação à metodologia RAD, selecione a opção **correta que indique como evitar retrabalho:**

A alternativa "D " está correta.

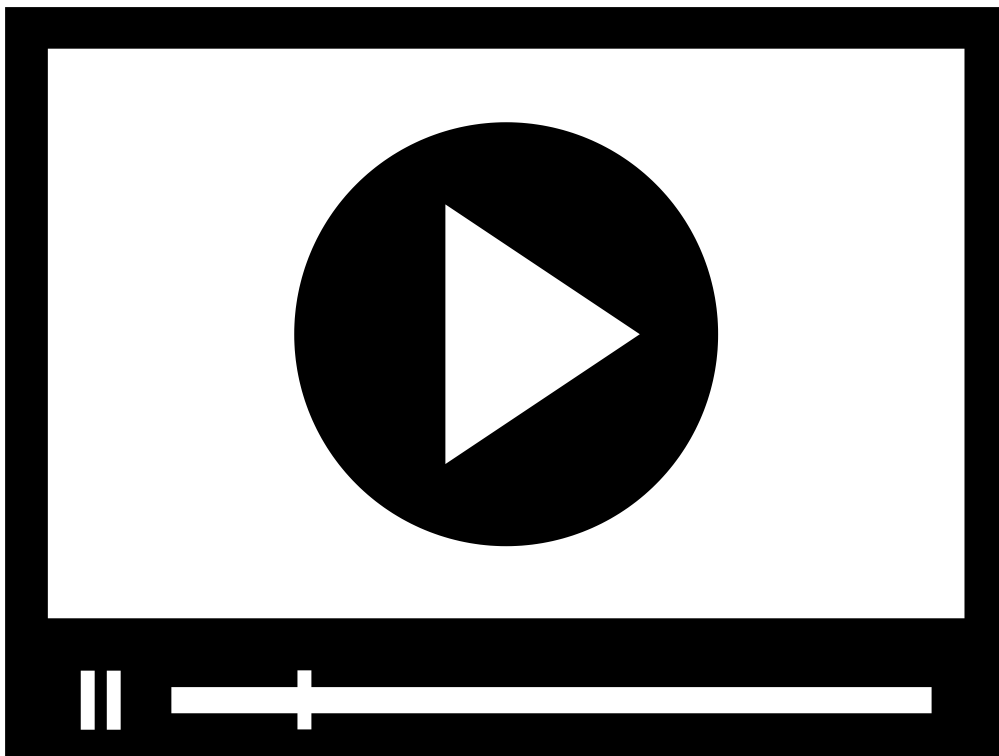
A colaboração entre as partes interessadas é fundamental para o sucesso da RAD. O desenvolvimento dos protótipos é o meio de comunicação para que os usuários deem suas opiniões a respeito do projeto e o validem, para que, assim, os desenvolvedores façam os ajustes necessários de modo que o sistema atenda aos requisitos.

MÓDULO 2

🕒 Descrever as modelagens de negócios e de dados da RAD

MODELAGEM (NEGÓCIOS E DADOS)

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.

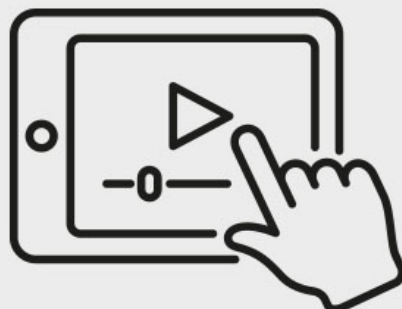


MODELAGEM DE NEGÓCIO NO RAD: TRANSFORMANDO IDEIAS EM RESULTADOS

Neste vídeo, mergulhe na importância da Modelagem de Negócio na Metodologia RAD.

Descubra como traduzir conceitos em sistemas tangíveis por meio de técnicas de modelagem eficazes. Aprenda como identificar requisitos essenciais, definir processos e criar uma base sólida para o desenvolvimento ágil.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



Shutterstock

A RAD é uma metodologia de desenvolvimento de software evolutiva que, além do levantamento de requisitos, visto no Módulo 1, possui outras fases. De modo resumido, segundo Kerr e Hunter (1994), elas podem ser descritas como:

Modelagem de negócios

Nessa fase, é feita a obtenção de informações a respeito dos requisitos funcionais do sistema, reunidas por várias fontes relacionadas aos negócios. 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. Trata-se, portanto, de uma análise com viés comercial.



Modelagem de dados

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. 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. Em seguida, é feito mapeamento que relaciona esses grupos entre si e qual o significado desses relacionamentos, conforme definido na etapa precedente. Avançando um pouco mais, agora 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.

COMENTÁRIO

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

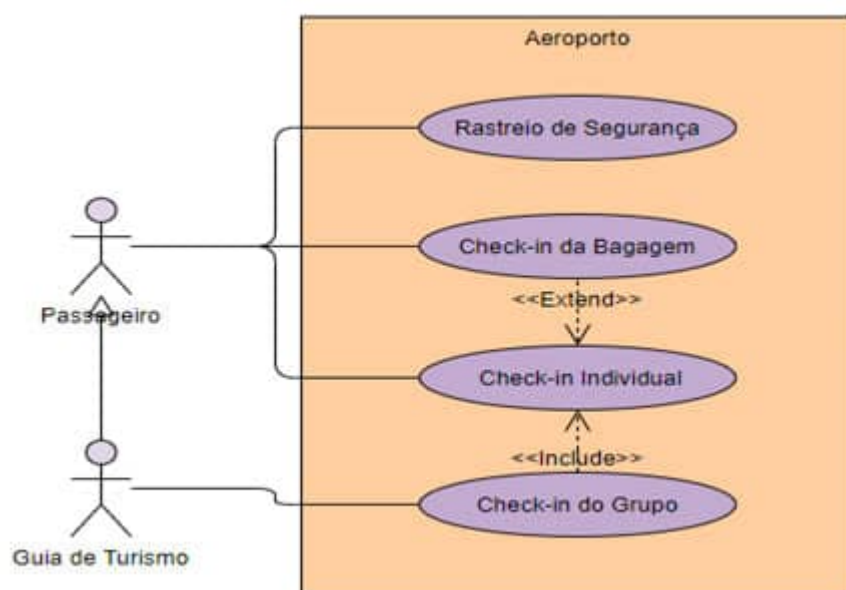
modelagem e, portanto, mais eficiente será a tradução das demandas do cliente para a implementação do sistema. Os modelos e as informações são representados por meio de diagramas conectados.

A seguir, detalharemos tanto a modelagem de negócios como a modelagem de dados.

MODELAGEM DE NEGÓCIOS

Uma das principais técnicas para representar a modelagem de negócios é utilizar o diagrama de Casos de Uso da UML. 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 Figura 1, é apresentado um exemplo de caso de uso de um sistema simplificado de embarque em um aeroporto.



Fonte: Autor

No modelo, 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 Figura 1, 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:

Processos de Negócios Principais
Processos Auxiliares
Processos de Gerenciamento

❑ **Atenção!** Para visualizaçãocompleta da tabela utilize a rolagem horizontal

Os **processos de negócios principais**: Que tratam das principais atividades do negócio.

Os **processos auxiliares**: São atividades que precisam ser feitas de qualquer modo, para que o negócio funcione.

Os **processos de gerenciamento**: Representam os processos que gerenciam outros processos e os relacionam com seus proprietários.

Para ilustrar melhor os conceitos apresentados até agora, será dado o exemplo de um restaurante: Os principais Casos de Uso de negócios são marketing e serviços de almoço e de jantar, e o caso de uso auxiliar é a compra de itens de alimentação e limpeza.

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 serem eliminadas da modelagem.



Fonte: Shutterstock

A modelagem foca apenas em 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 como características:

- 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. Outra coisa a ser considerada é evitar a redundância de Casos de Uso. Caso perceba-se 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.

TEORIA NA PRÁTICA

Como exemplo de generalização, seja um sistema de pagamento que permite que os frequentadores de uma academia paguem de duas formas: Online e por telefone. Certamente, esses dois cenários terão muitas coisas em comum e diferentes, reflita sobre esse contexto e escreva alguns desses aspectos comuns entre esses dois cenários:

RESOLUÇÃO

Alguns aspectos comuns 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 online, ou por telefone.

MODELAGEM DE DADOS

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.

Um exemplo 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.



Fonte:Shutterstock

O **modelo de dados** indica quais são os dados necessários e como devem ser organizados, ao invés de apontar quais operações precisam ser feitas. Os tipos de técnicas mais comuns para representar os modelos de dados são:

Modelo de entidade e relacionamento (E-R).

UML (Unified Modeling Language).

MODELO DE DADOS

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.

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. É um processo trabalhoso, mas investir na construção de um modelo de dados de qualidade 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:

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

Este modelo 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.

Detalhando um pouco mais os modelos, o principal objetivo do **modelo conceitual** é determinar quais são as entidades, seus atributos e como elas se relacionam. Aqui, não há detalhes da estrutura real do banco de dados. Os elementos básicos do modelo conceitual são:

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.

★ EXEMPLO

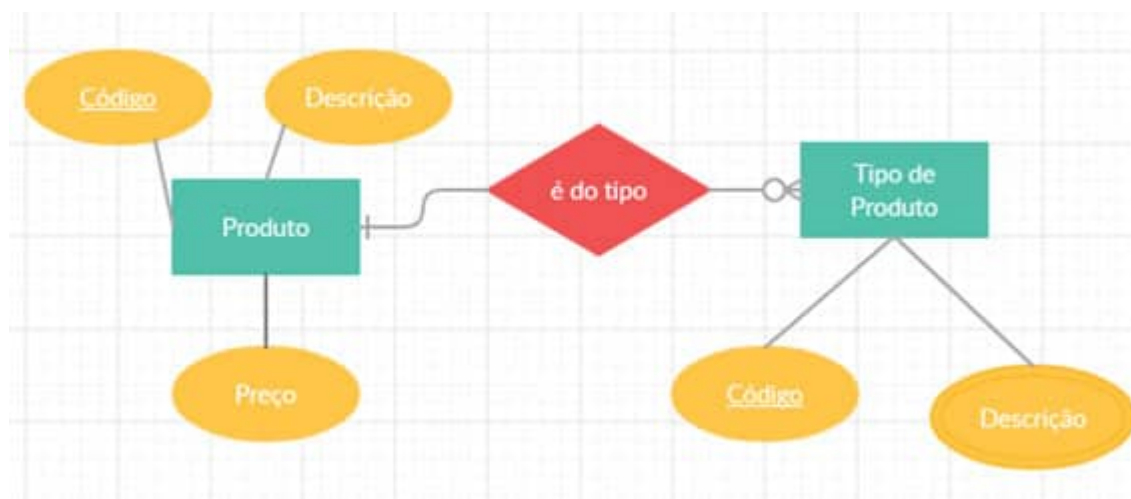
Exemplos de entidades são Escola e Aluno. O nome, matrícula e endereço são exemplos de atributos da entidade Aluno. Um exemplo de relacionamento entre as entidades Escola e Aluno é dado por “Uma Escola pode ter vários Alunos”.

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 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.

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 pode ser visto na Figura 11.



Fonte: O Autor

Nesse diagrama, 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.

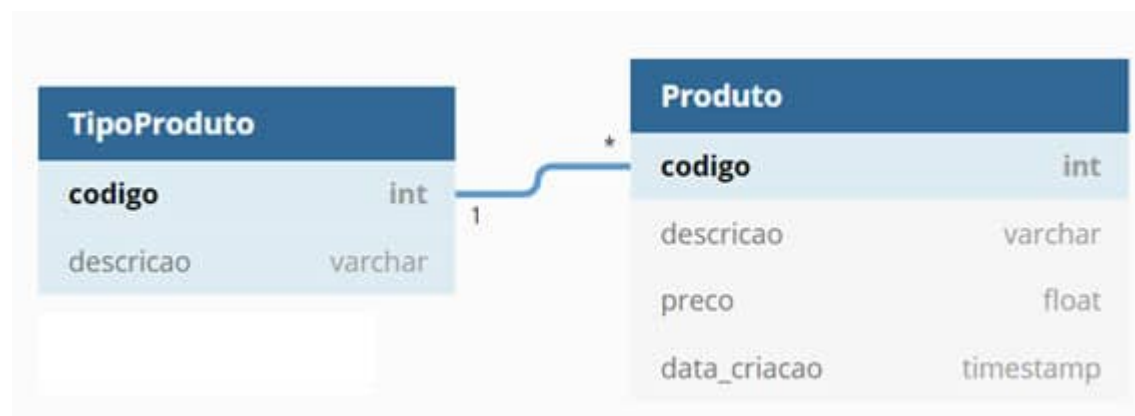
Existe outra etapa importante a respeito do modelo lógico, que é a normalização dos dados. Trata-se de uma técnica de organizar os dados de modo a evitar problemas de inconsistências entre eles.

📢 ATENÇÃO

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

Por fim, o **modelo de dados físicos** descreve detalhadamente como será a implementação no banco de dados. Com esse modelo, 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 figura 12, é apresentado um exemplo de modelo físico de dados.



Shutterstock

Na Figura 12, 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**”.

CADEIA DE CARACTERES

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.

MAIS ALGUMAS CONSIDERAÇÕES SOBRE A MODELAGEM DE DADOS E DE NEGÓCIOS

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.

adaptável

O modelo de dados é uma etapa
essencial no de

ciclo de vida

componentes

diretamente

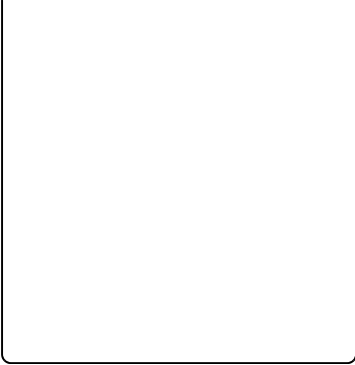
estabilize

foco

protótipos

usuário

qualquer desenvolvimento independentemente da metodologia utilizada. No caso da RAD, na qual a dinâmica do aprendizado do sistema está _____ relacionada às entregas através dos _____, é natural que o modelo de dados também evolua, mas é esperado que ele rapidamente _____ e o _____ esteja em questões relacionadas ao detalhamento de regras de negócio e na interatividade do usuário com o sistema, ou seja, questões relacionadas a _____ interativos e experiência do _____. O perfil esperado de um desenvolvedor da RAD é de um profissional com formação sólida, facilidade de aprendizado e muito _____. Isso, inclusive, é a principal



desvantagem da RAD, pois existem muitas expectativas, especialmente, dos desenvolvedores.

RESPOSTA

A sequência correta é:

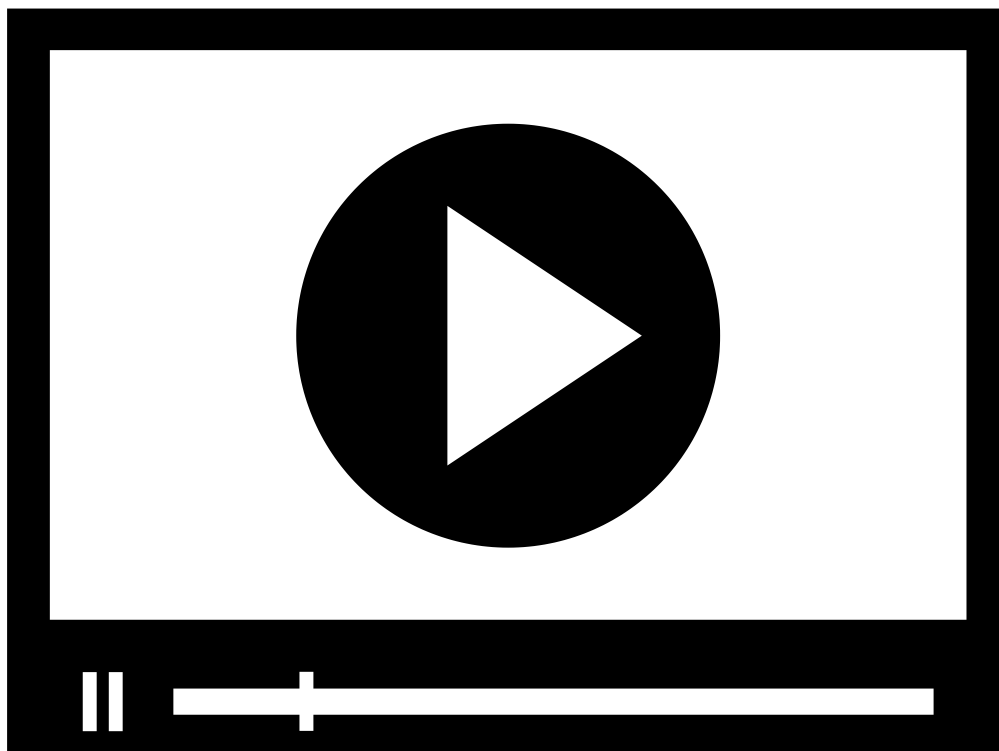
O modelo de dados é uma etapa essencial no **ciclo de vida** de qualquer desenvolvimento independentemente da metodologia utilizada. No caso da RAD, na qual a dinâmica do aprendizado do sistema está **diretamente** relacionada às entregas através dos **protótipos**, é natural que o modelo de dados também 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.



Shutterstock

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.



VÍDEO

Modelando o sistema

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



VERIFICANDO O APRENDIZADO

1. A RAD POSSUI FASES COM PROPÓSITOS ESPECÍFICOS. NESSE SENTIDO, SELECIONE A OPÇÃO **CORRETA QUE TEM COMO OBJETIVO CAPTURAR TODAS AS INFORMAÇÕES RELEVANTES SOBRE O PROPÓSITO DO SISTEMA E COMO ELE SE CONTEXTUALIZA NO NEGÓCIO:**

- A)** Modelagem de dados, pois identifica o fluxo de dados no sistema que dão suporte para o negócio.
- B)** Modelagem de negócios, pois identifica os elementos que caracterizam o negócio.
- C)** Levantamento de requisitos, uma vez que caracteriza as principais necessidades do negócio.
- D)** Modelagem de processos, dado que faz a conexão entre a concepção do sistema até a sua disponibilização para o usuário.

2. A FASE DE MODELAGEM DE DADOS TEM POR OBJETIVO FORMAR CONJUNTOS DE OBJETOS DE DADOS ESSENCIAIS PARA OS NEGÓCIOS. EM RELAÇÃO À FASE DE MODELAGEM DE DADOS, É CORRETO AFIRMAR QUE:

- A) Detalhes sobre como os dados serão implementados são tratados nesta fase.
- B) Por ser uma etapa avançada da RAD, nela são tratados cenários de testes.
- C) Aspectos relacionados à interatividade do sistema são discutidos nesta fase.
- D) Informações sobre o negócio são mapeadas nessa fase.

GABARITO

1. A RAD possui fases com propósitos específicos. Nesse sentido, selecione a opção correta que tem como objetivo capturar todas as informações relevantes sobre o propósito do sistema e como ele se contextualiza no negócio:

A alternativa **"B "** está correta.

A modelagem de negócios levanta aspectos da dinâmica em que o sistema vai operar, ou seja, atividades e pessoas relacionadas aos negócios sob análise.

2. A fase de modelagem de dados tem por objetivo formar conjuntos de objetos de dados essenciais para os negócios. Em relação à fase de modelagem de dados, é correto afirmar que:

A alternativa **"A "** está correta.

Na fase de modelagem de dados, além da identificação das entidades do negócio, elas são detalhadas e é feito o mapeamento de como se relacionam entre si, além de tratar o modo como será feita a implementação do modelo de dados.

MÓDULO 3

DESIGN DA INTERFACE COM USUÁRIO

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.



Shutterstock

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

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.



Fonte:Shutterstock

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 uma 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, tal como botões, janelas e barras de rolagem.

Mas existe outro elemento muito importante que também compõe o projeto de interface de um sistema, que é a experiência do usuário, conhecido pelo acrônimo em inglês UX (User Experience). Ambos os conceitos serão tratados com mais profundidade mais adiante.

PROTÓTIPO DE INTERFACE

O protótipo de interface de um sistema visa à experimentação de conceitos, de modo que evolua para uma representação do sistema no usuário. Trata-se de 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 das necessidades de profissionais bem treinados, é necessário que tenham à disposição ferramentas que facilitem esse processo pelo qual podem tornar a definição da interface em algo concreto.

Desenvolver protótipo de interface do usuário ajuda a refinar os requisitos funcionais logo no início do projeto. A implicação disso tem como consequências positivas a redução dos custos de manutenção, treinamento e melhoria do sistema, além de aumentar as 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, tem-se:

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 um ponto de referência para se comunicarem.

Facilita o engajamento dos usuários, pois podem fazer comentários sobre algo com que podem interagir, facilitando, assim, o trabalho posterior da equipe de desenvolvimento.

Melhora a qualidade das especificações funcionais.

Aumenta as chances de o produto atender às demandas.

Tem o potencial de reduzir o 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, entre os quais, estão:

Ignorar limitações que se aplicam ao produto real no processo de prototipagem.

Criar expectativas irreais sobre o produto.

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.

A NECESSIDADE DA CONTEXTUALIZAÇÃO

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.



Fonte:Shutterstock

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:

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

Aplicação de recursos que destaque um componente no contexto da aplicação. Por exemplo, usar sombras para botões.

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

A organização dos elementos deve ser legível. Para obter isso, é importante que haja uma hierarquia que os relacione, além da preocupação em fazer alinhamentos que priorizem o bem-estar do usuário.

O uso de recursos de cores e contrastes deve ser moderado e sempre com o propósito de guiar o usuário para realização de suas tarefas.

Os tamanhos das fontes, estilo — negrito e itálico, por exemplo —, maiúsculas e distância entre letras também devem ser contextualizados 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.

Por unidade de interação, por exemplo, telas, a quantidade de ações para executar tarefas deve ser a menor possível. O foco sempre deve ser o usuário, portanto a interface deve ser um guia, para que progridam na execução de tarefas complexas.

Os controles devem estar próximos dos objetos que os usuários desejam controlar. Por exemplo, um botão para enviar um formulário deve estar próximo dele.

Fornecer informações para os usuários a respeito de suas ações do sistema, como mensagens de confirmação de envio, por exemplo.

Em algumas situações, é apropriado aplicar padrões de utilização do sistema para facilitar o trabalho do usuário. Por exemplo, preencher formulários previamente. Essa opção, no entanto, deve ser aplicada com bastante ponderação, pois, apesar de ser bastante útil em muitas situações, pode ser utilizada por pessoas mal-intencionadas para obter indevidamente dados dos usuários.

O design dos componentes 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.

A interface deve fornecer os próximos passos 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.

A RAD é adequada para situações em que os projetos não são complexos. Por exemplo, funciona bem para criação, ou renovação de páginas de comércio eletrônico. Portanto, a escolha da RAD para implementar o design de interface de usuário deve satisfazer:

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 podem consumir bastante tempo e esforços que são 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.

Em relação ao último item, parece ser 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 micro serviços e arquitetura de software. Portanto, a escolha da RAD deve levar em consideração todas estas questões. Logo, depois de analisar esses pontos, e a RAD for a melhor opção, o processo deve seguir as seguintes etapas:

ETAPA 1

Escolha de 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.

TÉCNICAS DE PROTOTIPAGEM DE INTERFACE COM O USUÁRIO

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. Nas demais metodologias, o protótipo é usado para provar conceitos e, ao longo do desenvolvimento, o sistema é desenvolvido em características. No caso da RAD, a ideia é que o protótipo evoluirá para a versão final do sistema. Portanto, 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, são elas:

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, entre os itens a serem considerados estão:

Licença	Integração	Fidelidade
Curva de aprendizagem	Facilidade de uso e conforto	Compartilhamento

☐ **Atenção!** Para visualizaçãocompleta da tabela utilize a rolagem horizontal

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 este 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.

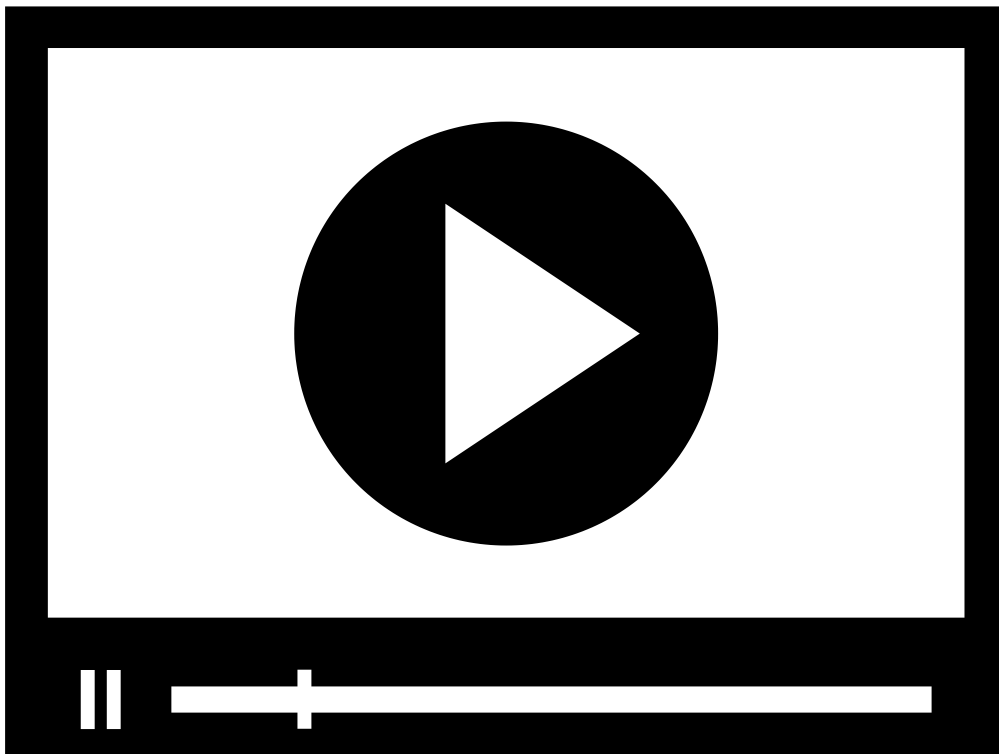
FACILIDADE DE USO E CONFORTO

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.



VÍDEO

Projetando a interface com o usuário

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



VERIFICANDO O APRENDIZADO

1. O DESIGN DE INTERFACE COM O USUÁRIO É UM PONTO FUNDAMENTAL NO DESENVOLVIMENTO DE SOFTWARE. EM RELAÇÃO À METODOLOGIA RAD, SELECIONE A OPÇÃO **CORRETA SOBRE AS VANTAGENS DE DESENVOLVER PROTÓTIPOS DE INTERFACE COM O USUÁRIO:**

- A) Limitar as possibilidades de evolução do sistema.
- B) Criar expectativas no usuário aumentando, assim, seu interesse pelo sistema.
- C) Ter um ambiente que sirva como referência para que as partes envolvidas possam interagir.
- D) Utilizar frameworks que acelerem o desenvolvimento.

2. O DESIGN DE INTERFACE COM O USUÁRIO CONTRIBUI PARA O ENGAJAMENTO DAS PARTES INTERESSADAS NO PROJETO, UMA VEZ QUE PODEM INTERAGIR COM UM SISTEMA REAL. APESAR DISSO, PODEM EXISTIR ALGUMAS DESVANTAGENS NESTE PROCESSO. EM RELAÇÃO À IMPLEMENTAÇÃO DO DESIGN DE INTERFACE COM O USUÁRIO NA METODOLOGIA RAD, SELECIONE A OPÇÃO **CORRETA SOBRE AS DESVANTAGENS DE DESENVOLVER PROTÓTIPOS DE INTERFACE COM O USUÁRIO:**

- A) As funcionalidades do protótipo não funcionam completamente.
 - B) Criar expectativas que dificilmente serão atendidas
 - C) Criar dificuldades no teste do sistema.
 - D) Consumir muito tempo na escolha de uma arquitetura de desenvolvimento adequada.
-

GABARITO

1. O design de interface com o usuário é um ponto fundamental no desenvolvimento de software. Em relação à metodologia RAD, selecione a opção **correta sobre as vantagens de desenvolver protótipos de interface com o usuário:**

A alternativa "**C**" está correta.

A interface gráfica fornece um ponto de referência para que os usuários possam interagir com o sistema e fazer comentários que vão auxiliar os desenvolvedores no direcionamento do projeto.

2. O design de interface com o usuário contribui para o engajamento das partes interessadas no projeto, uma vez que podem interagir com um sistema real. Apesar disso, podem existir algumas desvantagens neste processo. Em relação à implementação do design de interface com o usuário na metodologia RAD, selecione a opção **correta sobre as desvantagens de desenvolver protótipos de interface com o usuário:**

A alternativa "**B**" está correta.

O protótipo de interface com um usuário é um excelente recurso de comunicação entre as partes interessadas, mas é importante entender que o papel do protótipo na RAD é caracterizar a evolução do sistema. Portanto, na apresentação de um protótipo, deve-se ter cuidado de criar expectativas que não serão atendidas depois.

MÓDULO 4

IMPLEMENTAÇÃO EM PYTHON

Neste módulo, será desenvolvida uma aplicação prática de RAD usando o Python (ver Python 3.8.5, 2020). Antes de começar, é necessário pensar sobre alguns pontos sobre a RAD:

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.

É uma metodologia adequada para projetos de baixa complexidade.



Shutterstock

Ser uma metodologia voltada para entregas rápidas não quer dizer que o desenvolvedor não precisa saber programar bem, ao contrário, pois 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. Neste trabalho, o foco está na linguagem Python por se tratar de uma escolha natural por diversos motivos. Entre eles, estão: A simplicidade da sintaxe, disponibilidade de muitos pacotes, ampla

aceitação no mercado, uma 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, as etapas de requisitos, modelagem de negócios e de dados, além da implementação do sistema serão tratadas.

REQUISITOS

Aqui serão apresentados os requisitos do sistema, que são:

PROPÓSITO

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

ESCOPO

Este software será um Sistema de Cadastro de Notas de Aluno. Ele será 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.

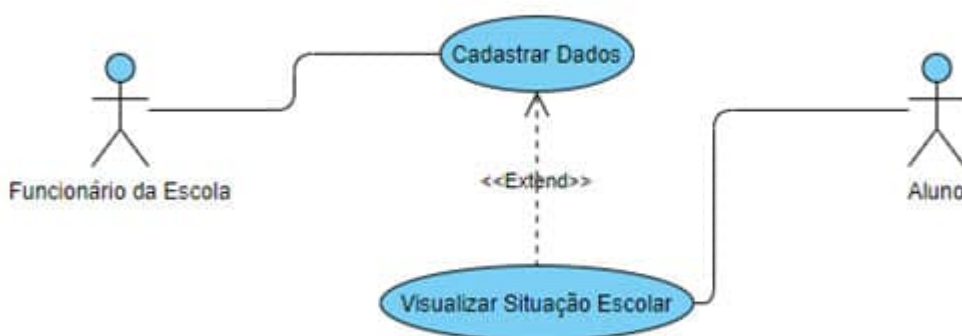
O usuário deve ser capaz de visualizar os dados cadastrados do aluno, sua média e sua situação: Aprovado, ou reprovado, ou em recuperação.

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 Figura 18 é apresentado o diagrama de Casos de Uso do sistema de cadastro de notas e visualização da situação escolar dos alunos.



Fonte: O autor

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 visa a 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.

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

□ **Atenção!** Para visualizaçãocompleta da tabela utilize a rolagem horizontal

DESIGN DA INTERFACE COM USUÁRIO

A interface gráfica do sistema de cadastro e visualização da situação escolar de alunos foi feita com o uso do pacote Tkinter (TKINTER, 2020). Trata-se da biblioteca de interface gráfica padrão da linguagem de programação Python. No ambiente de desenvolvimento Spyder (ver Spyder, 2020), para fazer a instalação do pacote “tkinter”, basta digitar:

```
!pip install tkinter
```

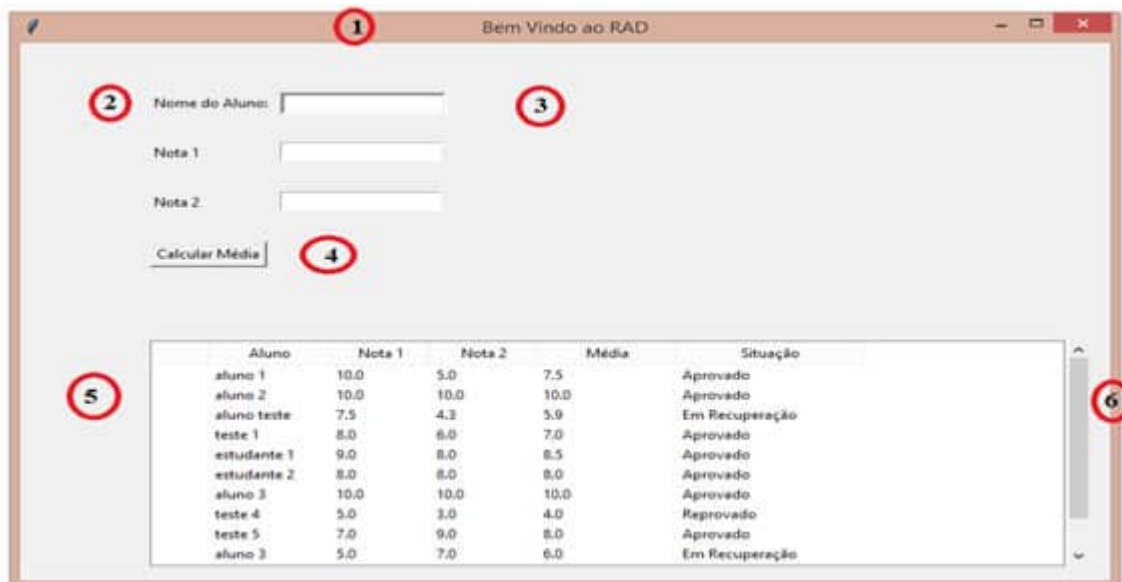
Existem algumas pequenas variações quando se faz a instalação de um pacote no Python. No caso do Spyder, é necessário colocar o símbolo exclamação “!” antes do comando “pip”. Para outras IDEs, é necessário retirar o caractere “!” (símbolo de exclamação) do comando. O pacote Tkinter está disponível na maioria das plataformas Unix, bem como em sistemas Windows. Portanto, é uma boa escolha para o desenvolvimento de interfaces com o usuário.

Para poder usar os componentes gráficos do Tkinter no código é necessário escrever logo no início do sistema as seguintes linhas de código:

```
import tkinter as tk
```

```
from tkinter import ttk
```

A interface gráfica do sistema é apresentada na figura 19.



Fonte: O autor

Nela, são destacados seis tipos de componentes:

Janela: Trata-se do formulário no qual os demais componentes são organizados. No caso da implementação do sistema, esse componente foi usado da seguinte maneira:

```
janela=tk.Tk()
```

Rótulo: Que é usado para textos estáticos. No caso do sistema, foi usado para os textos "Nome do Aluno", "Nota 1" e "Nota 2". Para referenciá-los no código, é necessário escrever: `tk.Label(janela, text='Texto estático')`.

Caixa de texto: São usadas para os usuários entrarem com os dados dos alunos. No caso do sistema, foi usado para entrar com o "nome do aluno", a "nota 1" e a "nota 2". Para referenciá-los no código, é necessário escrever: `tk.Entry()`.

Botão: Usado para iniciar uma ação. No caso do sistema, foi aplicado para cadastrar os dados dos alunos, calcular suas respectivas médias e determinar situação escolar em que o aluno se encontra. Para referenciá-los no código, é necessário escrever: `tk.Button()`;

Visão de árvore: Bastante útil para visualizar os dados. No sistema, foi usado para uma visualização em grade dos dados dos alunos, suas médias e respectivas situações. Para referenciá-lo no código, é necessário escrever: `ttk.Treeview()`.

Barra de rolagem: É um componente útil para interagir com a tabela da “visão de grade”, quando a quantidade de linhas de registros ultrapassa o limite inicial. Para usar no código, é necessário escrever `ttk.Scrollbar()`.

A interface gráfica auxilia no entendimento do sistema. Certamente, é mais fácil fazer sugestões de melhorias a partir de uma experiência real com o sistema do que em um nível abstrato.

DETALHES DE IMPLEMENTAÇÃO

Agora, serão abordados aspectos que se referem à implementação do sistema em Python. Vá até a seção “Explore +” e faça o download do arquivo “listaMediaAlunoFinal-v03.py”, assim será possível acompanhar a implementação.

A primeira parte está na escolha dos pacotes que foram utilizados no desenvolvimento do sistema. No caso, foram escolhidos: Tkinter e Pandas. O Tkinter foi escolhido por ser a biblioteca padrão de componentes gráficos do Python e poder ser utilizada em múltiplas plataformas, apesar de o sistema, inicialmente, ter sido projetado para operar no Windows. O pacote Pandas é útil para manipular conjuntos de dados, mais conhecidos pela expressão em inglês de “datasets”. Para usar as bibliotecas/pacotes, foi necessário escrever no código as seguintes linhas:

```
import tkinter as tk
```

```
from tkinter import ttk
```

```
import pandas as pd
```

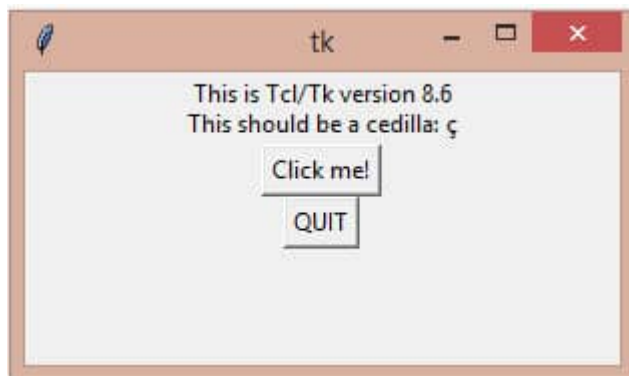
Antes de continuar, é importante verificar se a instalação das bibliotecas está correta. Na linha de comandos da IDE Spyder, digite a seguinte linha:

```
import tkinter
```

Pressione Enter e, em seguida, digite:

```
tkinter._test()
```

Pressione a tecla Enter novamente. Se tudo funcionar corretamente, aparecerá uma janela, conforme a figura 20.



Fonte:Shutterstock

Caso não tenha aparecido a janela, houve algum problema de instalação e será necessário corrigi-lo. Com os problemas de instalação resolvidos, passa-se à criação dos elementos de interface gráfica.

Na parte principal do programa, escreva as seguintes linhas de código, conforme a figura 21:

```
1 janela=tk.Tk()  
2 principal=PrincipalRAD(janela)  
3 janela.title('Bem Vindo ao RAD')  
4 janela.geometry("820x600+10+10")  
5 janela.mainloop()
```

Fonte:Shutterstock

A primeira linha trata da instanciação de um objeto “raiz” do Tkinter onde serão colocados os demais componentes gráficos.

Na linha 2, é feita a instanciação do objeto principal do projeto, onde o ciclo de vida dos componentes será gerenciado – isso inclui a criação, tratamento de ações e liberação para a memória.

Na linha 3, é dada a mensagem “título” para o projeto.

Na linha 4, são estabelecidas as dimensões altura e largura da janela principal.

Na linha 5, por fim, o sistema começa a executar até que o usuário feche a janela principal.

Agora vem a construção do ciclo de vida do programa. Na figura 22, é apresentado o trecho inicial do ciclo de vida dos componentes.

```

1 class PrincipalRAD:
2     def __init__(self, win):
3         #componentes
4         self.lblNome=tk.Label(win, text='Nome do Aluno:')
5         self.lblNota1=tk.Label(win, text='Nota 1')
6         self.lblNota2=tk.Label(win, text='Nota 2')
7         self.lblMedia=tk.Label(win, text='Média')
8         self.txtNome=tk.Entry(bd=3)
9         self.txtNota1=tk.Entry()
10        self.txtNota2=tk.Entry()
11        self.btnCalcular=tk.Button(win, text='Calcular Média', command=self.fCalcularMedia)
12        #----- Componente TreeView -----
13        self.dadosColunas = ("Aluno", "Nota1", "Nota2", "Média", "Situação")
14        self.treeMedias = ttk.Treeview(win,
15                                     columns=self.dadosColunas,
16                                     selectmode='browse')
17        self.verscrlbar = ttk.Scrollbar(win,
18                                     orient="vertical",
19                                     command=self.treeMedias.yview)
20        self.verscrlbar.pack(side='right', fill='x')
21        self.treeMedias.configure(yscrollcommand=self.verscrlbar.set)

```

Shutterstock

Mesmo para um projeto pequeno, existem muitos detalhes a ser observados. Logo na linha 1 está a implementação da classe principal do sistema. É nessa classe que vai estar a lógica do tratamento do ciclo de vida dos componentes gráficos. Das linhas 4 a 21, cada objeto que se refere a um componente gráfico é instanciado. Em especial, merece atenção a linha 11:

self.btnCalcular=tk.Button(win, text='Calcular Média', command=self.fCalcularMedia)

Nessa linha, o objeto '**btnCalcular**' do tipo '**tk.Button**' recebe o texto '**Calcular Média**', que vai aparecer no botão e, além disso, é associado à função "**fCalcularMedia**".

Outro ponto importante para observar no trecho de código da Figura 22 é a linha 2:

def __init__(self, win)

Que é a função "construtora" da classe, ou seja, o primeiro método a ser acionado no ciclo de vida do componente raiz principal.

Ainda na função "init", os componentes são posicionados na janela principal conforme é possível verificar na Figura 23.


```

1  #-----
2  #posicionamento dos componentes na janela
3  #-----
4  self.lblNome.place(x=100, y=50)
5  self.txtNome.place(x=200, y=50)
6
7  self.lblNota1.place(x=100, y=100)
8  self.txtNota1.place(x=200, y=100)
9
10 self.lblNota2.place(x=100, y=150)
11 self.txtNota2.place(x=200, y=150)
12
13 self.btnCalcular.place(x=100, y=200)
14
15 self.treeMedias.place(x=100, y=300)
16 self.verscrlbar.place(x=805, y=300, height=225)
17
18 self.id = 0
19 self.iid = 0
20
21 self.carregarDadosIniciais()
22

```

Shutterstock

Um trecho interessante de ser destacado é a linha 21. Nele, é feito uma chamada para o método “**carregarDadosIniciais**”. Esse método é o responsável por carregar os dados que já estão armazenados na planilha para o componente “**treeview**”, que é a grade que aparece na tela principal. Na Figura 11, é apresentado o código do método “**carregarDadosIniciais**”.

```

1  def carregarDadosIniciais(self):
2      try:
3          fsave = 'planilhaAlunos.xlsx'
4          dados = pd.read_excel(fsave)
5          nn=len(dados['Aluno'])
6          for i in range(nn):
7              nome = dados['Aluno'][i]
8              notal1 = str(dados['Nota1'][i])
9              nota2 = str(dados['Nota2'][i])
10             media=str(dados['Média'][i])
11             situacao=dados['Situação'][i]
12             self.treeMedias.insert('', 'end',
13                                     iid=self.iid,
14                                     values=(nome,
15                                             notal1,
16                                             nota2,
17                                             media,
18                                             situacao))
19             self.iid = self.iid + 1
20             self.id = self.id + 1
21         except:
22             print('Ainda não existem dados para carregar')

```

Shutterstock

Como dito anteriormente, o objetivo desse método é carregar os dados da planilha para o componente “treeview” que é a grade que aparece na janela principal. Um dos problemas que podem ocorrer nessa tentativa de carregar os dados é que ainda não existem dados, ou mesmo, uma planilha de onde carregá-los.

SAIBA MAIS

Caso não seja dado um tratamento explícito para essa situação de exceção, o programa vai apresentar uma mensagem de erro, caso a planilha não exista. O ideal é detectar todas as situações de exceção logo no começo do projeto, pois isso evitará o consumo de horas de testes que podem ser mais bem direcionadas para tentar descobrir problemas mais complexos que podem ser vulnerabilidades para o sistema. O tratamento de exceção foi dado pela aplicação das cláusulas “try-except” (linhas 2 e 21).

Outro ponto interessante de se observar é na linha 4: **“dados = pd.read_excel(fsave)”**.

Com apenas um único comando, os dados da planilha de dados dos alunos são carregados para a variável **“dados”**. Isso ocorre com o uso da função **“read_excel”** da biblioteca **“pandas”**. Com isso, esses dados podem ser facilmente manipulados no código. O nome pelo qual esse tipo de dado é conhecido é **“datasets”**. Das linhas 6 a 20, os dados são obtidos da variável **“dados”** e inseridos no componente **“treeview”**. Caso ocorra algum problema, será enviada a mensagem “Ainda não existem dados para carregar”.

Esse tipo de programação protege o sistema de eventuais problemas através do estabelecimento de um comportamento previsível. Assim, o código fica menos vulnerável contra ataques que tentem ver mensagens padrões não tratadas que revelem características sobre o código, ou o ambiente em que o sistema está operando.

Agora vem o cálculo da média do aluno, identificação de sua situação escolar e armazenamento de dados no sistema.

Na Figura 25 é apresentado o código que é responsável por essas operações.

```

1  def fCalcularMedia(self):
2      try:
3          nome = self.txtNome.get()
4          nota1=float(self.txtNota1.get())
5          nota2=float(self.txtNota2.get())
6          media, situacao = self.fVerificarSituacao(nota1, nota2)
7          self.treeMedias.insert('', 'end',
8                                  iid=self.iid,
9                                  values=(nome,
10                                         str(nota1),
11                                         str(nota2),
12                                         str(media),
13                                         situacao))
14
15         self.iid = self.iid + 1
16         self.id = self.id + 1
17         self.fSalvarDados()
18     except ValueError:
19         print('Entre com valores válidos')
20     finally:
21         self.txtNome.delete(0, 'end')
22         self.txtNota1.delete(0, 'end')
23         self.txtNota2.delete(0, 'end')

```

Fonte: O Autor

Novamente, deve ser destacado o bloco “**try-except**” nas linhas 2 e 17, pois o sistema está esperando um nome — que é um campo texto — e duas notas —, que são campos numéricos. Caso o usuário entre com um dado inválido, ao invés de gerar uma falha o sistema vai exibir a mensagem “Entre com dados válidos”. Além disso, esse método ainda possui a cláusula “**finally**” que vai ser executada de qualquer modo. No caso, os campos de entrada serão “limpos”.

No método “**fCalcularMedia**” são feitas chamadas para os métodos “**fSalvarDados**” e “**fVerificarSituacao**”. O método “**fSalvarDados**” já foi explicado. Em relação ao método “**fVerificarSituacao**”, ele é apresentado na Figura 26.

```

1  def fVerificarSituacao(self, nota1, nota2):
2
3      media=(nota1+nota2)/2
4      if(media>=7.0):
5          situacao = 'Aprovado'
6      elif(media>=5.0):
7          situacao = 'Em Recuperação'
8      else:
9          situacao = 'Reprovado'
10
11     return media, situacao

```

Fonte: O autor

É no método “**fVerificarSituacao**” em que a média é calculada e a situação escolar do aluno é determinada. Das linhas 3 a 8, é encadeada uma série de testes a partir do cálculo da média

na linha 2. No final do método, na linha 10, são retornadas as variáveis “media” e a “situacao” que correspondem, respectivamente, à média e à situação escolar do aluno para o método que fez a chamada. Aqui, está mais uma vantagem da sintaxe do Python: Retorno de múltiplas variáveis. Isso facilita bastante o processo de programação, poupando tempo para o desenvolvedor.

ATENÇÃO

Outro ponto que precisa ser observado, mais no sentido da sintaxe do Python, é o uso da palavra reservada “**self**”. Esse projeto foi programado orientado a objetos (POO). Na POO, a classe é formada por atributos e métodos. Quando são feitas referências aos métodos e atributos na própria classe, é necessário indicar sua origem com o uso da palavra “**self**” seguida de “.” (ponto-final).

EVOLUÇÕES

O sistema “**simplificado de cadastro de notas de alunos**” é funcional. Ou seja, com o que foi construído até agora, é possível ter reuniões com os usuários para apresentar resultados, detectar vulnerabilidades, fazer ajustes e propor avanços.

POR EXEMPLO, O USUÁRIO PERCEBEU A NECESSIDADE DE UMA TELA DE LOGIN EM QUE ALUNOS E FUNCIONÁRIOS TIVESSEM VISÕES ESPECÍFICAS DAS FUNCIONALIDADES DO SISTEMA. OUTRO PONTO PARA EVOLUIR É SOBRE O ARMAZENAMENTO DE DADOS, OU SEJA, PARTIR PARA UMA SOLUÇÃO QUE USE UM SISTEMA GERENCIADOR DE BANCO DE

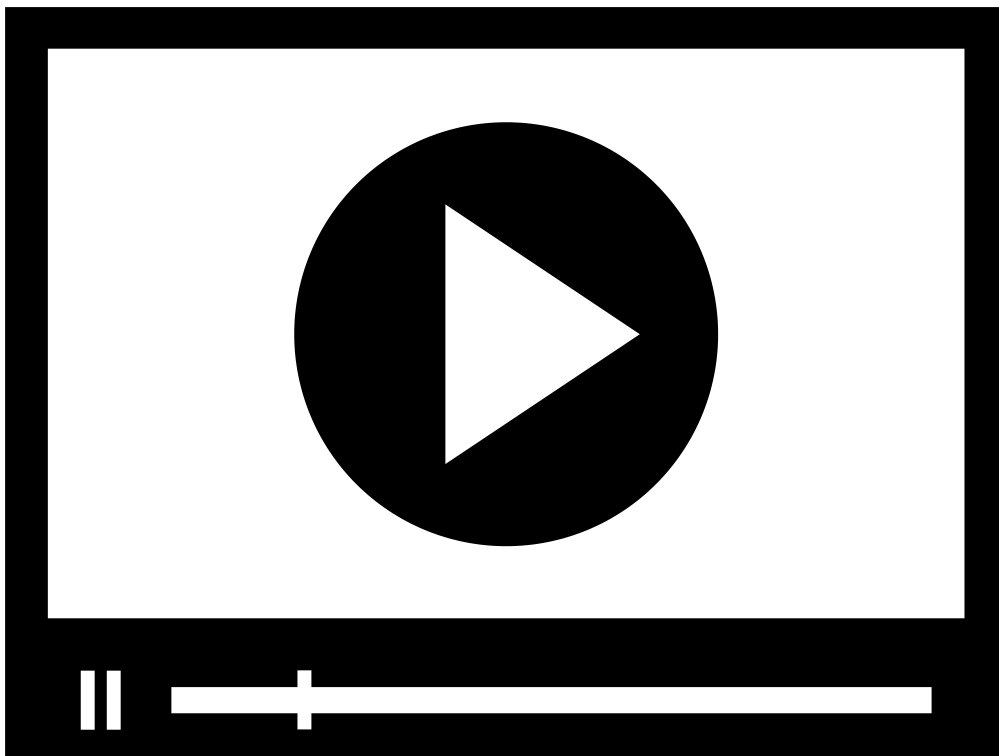
DADOS. DE ACORDO COM ESSE CONTEXTO, QUE TIPOS DE VISUALIZAÇÕES DE INFORMAÇÕES QUE OS USUÁRIOS PODERIAM SENTIR NECESSIDADE?

RESPOSTA 1

RESPOSTA

Os usuários poderiam perceber a necessidade de ter mais algumas visualizações de informações, como, por exemplo, a média e o desvio-padrão da turma, pois essas informações poderiam apoiar o setor pedagógico da escola no desenvolvimento de estratégias para detectar problemas e propor soluções para a turma de um modo geral.

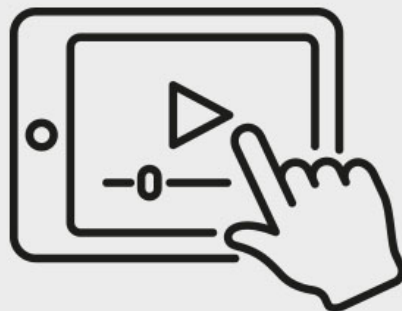
Portanto, tem uma série de questões nesse exemplo que se encaixam em uma metodologia evolutiva como a RAD. Mas é importante que fique claro que a parte mais importante da RAD está na qualidade dos profissionais envolvidos no processo, portanto investir em qualificação técnica e acadêmica é fundamental e no ambiente colaborativo entre as partes interessadas.



VÍDEO

Construindo o protótipo

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



VERIFICANDO O APRENDIZADO

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.

2. A COMPREENSÃO DOS REQUISITOS DE UM SISTEMA É ESSENCIAL PARA INICIAR UM PROJETO DE DESENVOLVIMENTO DE SOFTWARE. NESSE PROCESSO, ALGUMAS INFORMAÇÕES A RESPEITO DO SISTEMA DEVEM SER EXPOSTAS DE MODO A RECEBER O TRATAMENTO ADEQUADO NAS ETAPAS POSTERIORES DO DESENVOLVIMENTO DO SISTEMA. NESSE SENTIDO, SELECIONE A OPÇÃO CORRETA QUE RELACIONA INFORMAÇÕES SOBRE REQUISITOS DE UM SISTEMA E SUA RESPECTIVA JUSTIFICATIVA:

A) Propósito: Descrever as funcionalidades do sistema e como deve ser sua operação no dia a dia.

B) Escopo: Descrição do sistema e delimitação de funcionalidades.

C) Interfaces de software: Descrição dos elementos de interface com o usuário.

D) Especificação de Requisitos Funcionais: Definição dos papéis funcionais dos usuários no sistema e quais requisitos devem ser atendidos para sua correta operação.

GABARITO

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 alternativa "C " está correta.

O propósito de um sistema descreve o que é o sistema e identifica quais serão os seus usuários.

2. A compreensão dos requisitos de um sistema é essencial para iniciar um projeto de desenvolvimento de software. Nesse processo, algumas informações a respeito do sistema devem ser expostas de modo a receber o tratamento adequado nas etapas posteriores do desenvolvimento do sistema. Nesse sentido, selecione a opção **correta que relaciona informações sobre requisitos de um sistema e sua respectiva justificativa:**

A alternativa "B " está correta.

O escopo do sistema é a descrição do que ele se propõe a fazer. Portanto, é necessário apresentar suas funcionalidades e outras informações relevantes que deem suporte para que o usuário tenha uma clara compreensão sobre o sistema.

CONCLUSÃO

CONSIDERAÇÕES FINAIS

No decorrer do texto, apresentamos aspectos práticos das fases da metodologia RAD, tais como as etapas para identificar os requisitos de um sistema, as modelagens de negócios e de dados, o design de interface com o usuário e o desenvolvimento de uma aplicação utilizando a linguagem de programação Python.

A compreensão de como estas etapas devem ser realizadas e como se relacionam é um requisito para poder utilizar a RAD da forma adequada. Essa metodologia não é recomendável para qualquer situação, no entanto, em casos de sistemas com baixa complexidade que,

atualmente, correspondem à maioria das situações do cotidiano, a RAD é uma forte candidata para apoiar o desenvolvimento.

Os benefícios da aplicação da RAD têm impacto desde a melhor utilização do tempo das partes envolvidas até a minimização dos riscos do projeto, através de um processo colaborativo. A escolha da linguagem Python para projetos RAD é natural devido à vasta disponibilidade de documentação e de pacotes de componentes gráficos e de manipulação de dados que facilitam o desenvolvimento.

Para ouvir um *podcast* sobre o assunto, acesse a versão online deste conteúdo.



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.** *In: Information Systems Journal*, 2009, 19(6), 549– 570.

DOCS.PYTHON. **Python 3.8.5 Documentation.** Consultado em meio eletrônico em: 01 out. 2020.

DOCS.PYTHON. **Tkinter** – Python interface to Tcl/Tk. Consultado em meio eletrônico em: 01 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 em meio eletrônico em: 01 out. 2020.

EXPLORE+

Para saber mais sobre os assuntos tratados neste tema, visite os seguintes sites na internet:

O site oficial do Python e leia sobre conceitos da linguagem e de aplicações GUI, além de poder fazer downloads de diversos pacotes para desenvolvimento rápido.

O site oficial de documentação da Microsoft e procure por normalização de tabelas de bancos de dados.

O site oficial do Planalto do Governo Federal e procure pela lei nº 13.709. Veja também mais detalhes sobre a Lei Geral de Proteção de Dados.

CONTEUDISTA

Sérgio Assunção Monteiro

