sandbox

CÓDIGO | DESIGN | PROCESSOS | HISTÓRIAS

004 : dez 2023

Apple Developer Academy IFCE

Revista “**Sandbox**”. Edição 004. De Dezembro de 2023. “Sandbox” é uma revista de distribuição gratuita.

No momento da escrita das metérias desta revista, as versões mais recentes do Xcode e Swift eram 15 e 5.9 respectivamente. Consultar o histórico de atualizações caso esteja trabalho com uma versão diferente destas.

Para saber mais sobre o
IFCE Apple Developer
Academy, acesso o site:
www.developeracademy.ifce.edu.br

Equipe de Redação: Ana Guimarães, Ayslana Riene, Emily Oliveira, Helena Oliveira, João Victor Ipirajá, Ieda Xavier, Letícia Dutra, Mateus Calisto, Moysés Azevedo, Paulo Henrique, Pedro Muniz, Samantha Eufrásio, Sarah Madalena, Thaynara Andrade, Yara Sampaio.
Equipe de Revisão: Cáren Sousa, David Augusto, Elis Vieira, Lucas Baptista, Marília Oliveira, Milena Maia, Pedro Muniz. **Equipe de Ilustração:** Carolina Parente, David Augusto, Gabriela Souza, Helena Oliveira, Ieda Xavier, Isadora Fontenele, João Victor Ipirajá, Lais Barbosa, Letícia Dutra, Nillia Sousa, Pedro Muniz, Sarah Madalena.

Revisão Técnica de Código: Cáren Sousa, Mateus Rodrigues

Revisão Técnica de Design: David Augusto, Elis Vieira e Lucas Baptista

Capa: Carolina Parente

Projeto gráfico: Lucas Baptista

Diagramação: Lucas Baptista

Editor Geral: Lucas Baptista

EDITORIAL

"Passar pela Apple Developer Academy é uma experiência que mudou minha vida." Esse é provavelmente um dos relatos que escutamos com mais frequência ao ouvirmos pessoas falarem sobre como foi passar pelo programa. E nem é difícil imaginar a verdade desse depoimento, principalmente quando somamos a ele outras histórias e boatos que ouvimos sobre alumni de sucesso ou sobre o que acontece no próprio programa. Aquele marketing orgânico que a Academy tem nos corredores do próprio IFCE e além deles, boca a boca.

Dois anos depois, ainda consigo lembrar do quão impactante foi entrar nos laboratórios pela primeira vez. Cada pequeno momento dessa jornada parece cortar a linearidade do tempo e entrar na eternidade: receber os equipamentos e escrever as primeiras linhas de código; assistir as primeiras apresentações no Campfire; marcar as paredes com nossos brainstormings e nossos planos de projetos revolucionários; dividir o lanche e o café com pessoas que, dia após dia, foram se tornando tão familiares e queridas. São vinte e quatro meses de convivência com pessoas incríveis, com tantas diferentes potencialidades e tanto a compartilhar, crescendo junto. Uma convivência tão intensa que nos faz conseguir ouvir a voz de cada colega ao ler algo escrito por ela.

Vinte e quatro meses de desafios, de (muitas) dúvidas, umas tantas lágrimas derramadas, incontáveis risadas durante os intervalos. São vinte e quatro meses de experiências que, ainda que fosse simplesmente por sua intensidade, não poderiam não deixar marcas. São também vinte e quatro meses de superação, de pequenas e grandes vitórias no código, de testes e pesquisas que se transformaram em excelentes escolhas de design, de fofocas, de piadas, de amizades, de amores, de conquistas inestimáveis. Vinte e quatro meses que não caberiam em vinte e quatro milhões de linhas, pois são fruto de cinquenta seres e mentes radicalmente incomparáveis. E de dez anos de história deste projeto.

Com tudo isso, a sensação é a de que esse é só o início. Com certeza o início de jornadas muito, muito bonitas. Um início no qual já partimos na frente, porque levamos uma bagagem gigantesca: profissionalmente, com todos os conhecimentos técnicos que adquirimos; e pessoalmente, ao conhecer mais e mais profundamente nossas próprias potencialidades. Nessas páginas você vai encontrar pequenas e preciosas porções do que foi acumulado nesses dois anos pela turma de 2023. Conhecimentos dos mais variados tipos, cobrindo os mais variados assuntos, que contribuirão para mudar o mundo para melhor. Que sejamos audazes o suficiente para tal. E que, daqui a um tempo, possamos olhar para trás e também dizer o quanto passar pela Academy mudou nossas vidas, o quanto se deve àqueles vinte e quatro meses.

Pedro Muniz
Aluno da Turma 2022-2023

CONTEÚDO

DESIGN PROCESSO UX

ETNOGRAFIA NO DESIGN DE UX

TEXTO
HELENA OLIVEIRA
ILUSTRAÇÃO
DAVID AUGUSTO

Tem como objetivo entender aspectos culturais, hábitos, convenções e comportamentos de grupos sociais específicos.

O que é etnografia?

A etnografia é uma abordagem oriunda das ciências sociais e antropológicas que tem como objetivo estudar e entender aspectos culturais, hábitos, convenções e comportamentos de grupos sociais específicos. Métodos de pesquisa etnográficos são, em sua maioria, baseados em pesquisa de campo, para que o pesquisador possa mergulhar no ecossistema do grupo que está sendo estudado.

Vendo sob a ótica do design de experiência do usuário, a etnografia é um método de pesquisa flexível que fornece informações de pesquisa em primeira mão sobre os usuários. Ao ficar imerso no ambiente de um grupo, o pesquisador (seja o ux designer ou o ux researcher) obtém acesso a informações mais autênticas, que poderiam não ter sido descobertas simplesmente perguntando. As atividades e comportamentos dos usuários são monitorados em seu "habitat natural". Dessa forma, é possível observar espontaneamente os usuários em seus ambientes reais e identificar pontos problemáticos, frustrações e casos de uso que ajudarão a projetar soluções.

Vantagens da pesquisa etnográfica

A principal vantagem da etnografia é que ela dá aos pesquisadores acesso direto à cultura e às práticas de um determinado grupo, fornecendo contexto de uso. É uma abordagem útil para obter conhecimento verídico sobre o comportamento e as interações das pessoas dentro de um contexto específico. A etnografia permite que os pesquisadores observem não só as situações, modo e circunstâncias em que o produto será usado, como também as reações e opiniões autênticas dos usuários. Imagine que sua empresa produz guardas-chuva. Uma dúvida que pode ser esclarecida por um teste de usabilidade é: Com que rapidez os usuários abrem seu guarda-chuva quando começa a chover? Entretanto, um método de pesquisa etnográfico poderia fornecer mais informações, em outros contextos, como por exemplo: Em que ponto os usuários pegam seu guarda-chuva, depois do primeiro trovão, ou antes dele? Em que ponto entre a névoa e a chuva o guarda-chuva é usado pelos usuários?

O teste de usabilidade está se concentrando no gasto de energia dos participantes e em como é fácil para os usuários interagirem com o produto. A pesquisa de usabilidade pode dizer como foi fácil ou difícil para os usuários interagirem com o design, mas um estudo etnográfico pode explicar por que eles interagiram com o produto de uma determinada forma.

Viéses

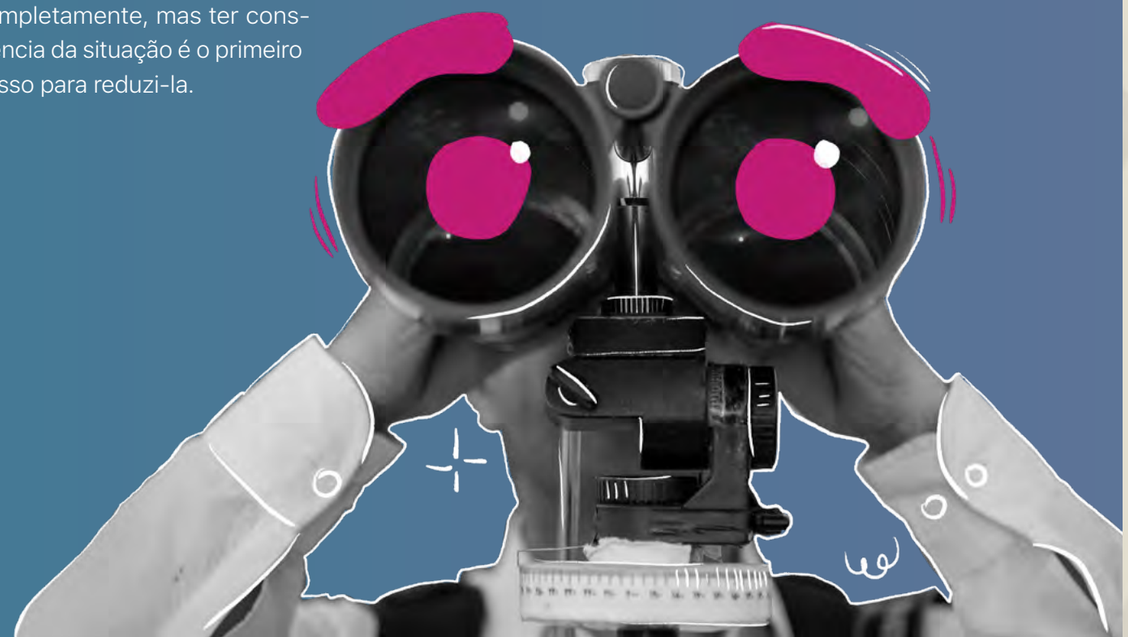
Durante a execução da pesquisa etnográfica, dois tipos de viéses são mais comuns de acontecerem:

Viés observacional acontece quando as pessoas se comportam de forma diferente ao saber que estão sendo estudadas. A melhor maneira de explicar esse viés ao participante é equilibrar os resultados de outros tipos de pesquisa quando aqueles que estão sendo estudados não estão cientes de que estão sendo estudados, ou quando outros dados vêm de fontes mais neutras.

Viés do observador é o mesmo fenômeno, mas do lado do pesquisador - quando ele acredita ou declara que observou algo simplesmente porque esperava ou queria observar. Isso também pode ser impossível de evitar completamente, mas ter consciência da situação é o primeiro passo para reduzi-la.

Limitações

Por outro lado, existem algumas limitações com a pesquisa etnográfica em UX. A principal questão é que a pesquisa etnográfica pode ser demorada, podendo durar até meses, o que representa um custo alto para o projeto. Outra limitação da pesquisa etnográfica é a subjetividade, visto que envolve a interpretação dos pesquisadores sobre o que observam e experimentam. Isso pode levar a viéses, como mostrado anteriormente, e interpretações subjetivas, já que diferentes pesquisadores podem interpretar os mesmos dados de maneira diferente.



CÓDIGO

TUTORIAL

APRENDA OS PRINCÍPIOS SOLID COM OS POWER RANGERS

TEXTO
ANA GUIMARÃES

ILUSTRAÇÃO
CAROLINA PARENTE

REVISÃO
CÁREN SOUSA
LUCAS BAPTISTA

Se você foi uma criança nos anos 90 e 2000, possivelmente ja assistiu algum episódio de Power Rangers tomando Nescau e comendo biscoito no sofá (inclusive sdds dessa época 🥹).

Porém, se, por algum milagre, você NUNCA tiver ouvido falar sobre essa perfeição, hoje é teu dia de sorte.

Power Rangers é uma icônica série de televisão que mistura ação, aventura e elementos de ficção científica. A série possui diversas versões, mas, nesse artigo, vamos falar sobre a primeira que chama "Mighty Morphin Power Rangers".

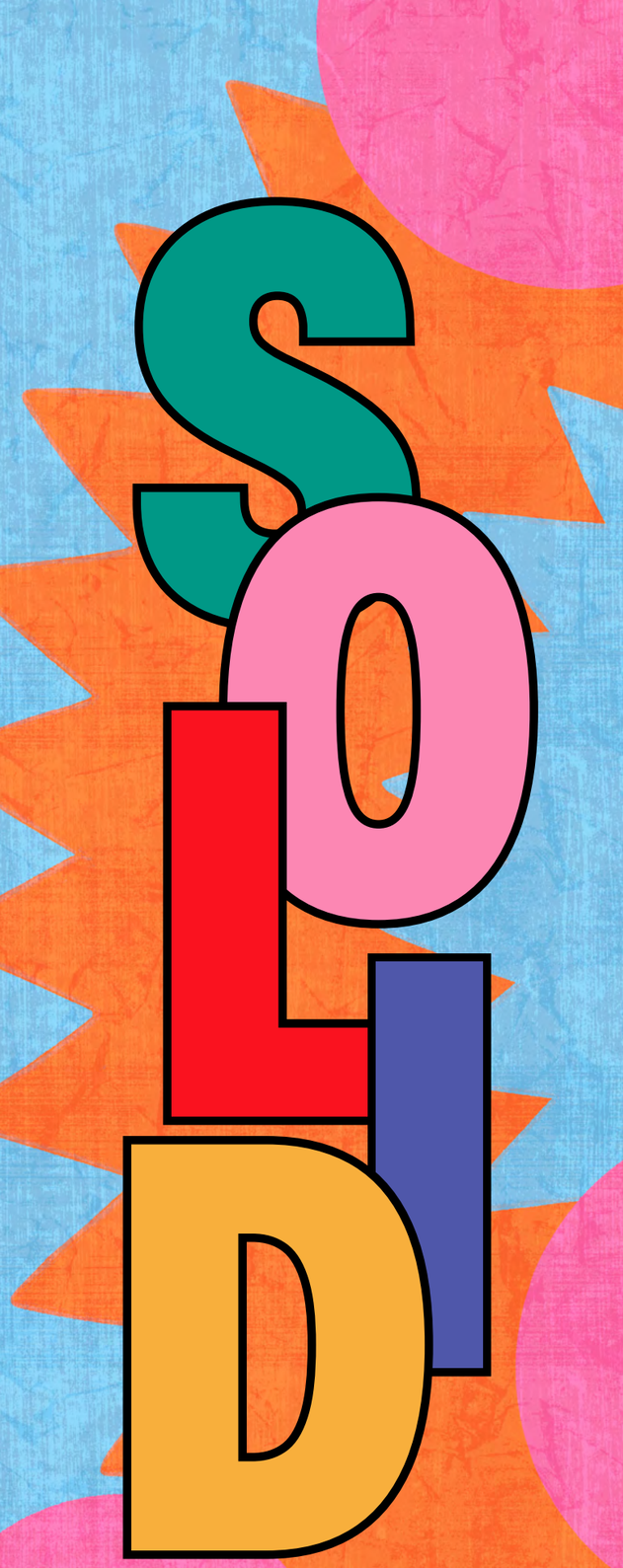
A trama gira em torno de um grupo de cinco jovens adolescentes comuns que são escolhidos para se tornarem os lendários Power Rangers. Eles recebem poderes especiais e são encarregados de proteger a cidade Angel Grove e o mundo de ameaças monstruosas e vilões intergalácticos, liderados pelo maligno Rita Repulsa e, posteriormente, por Lord Zedd.

Cada Ranger é representado por uma cor específica e comanda um Zord, uma espécie de robô gigante que se transforma em um dinossauro ou outro ser poderoso. Quando a ameaça se torna grande demais para ser enfrentada individualmente, os Rangers combinam seus Zords para formar o Megazord, uma poderosa máquina de combate capaz de derrotar os inimigos mais temíveis.

Além das emocionantes batalhas contra monstros e vilões, a série também aborda temas de amizade, trabalho em equipe, respeito e responsabilidade, transmitindo lições valiosas para os jovens espectadores.

"Mighty Morphin Power Rangers" se tornou um fenômeno cultural, gerando uma extensa franquia de séries de TV, filmes, jogos, brinquedos e quadrinhos. Mesmo após décadas de seu lançamento original, a série continua a ser amada e lembrada por fãs de todas as idades, mantendo seu legado como uma das mais queridas produções da cultura pop.

Depois dessa introdução detalhada, vamos entender o que os Rangers podem nos ensinar sobre SOLID.



Princípio S - Responsabilidade Única

(Single Responsibility Principle):

Imagine que cada Power Ranger é especializado em uma habilidade ou conjunto de habilidades específicas. O Ranger Vermelho é especialista em combate corpo a corpo, o Ranger Azul é especialista em tecnologia e ciência, o Ranger Amarelo é especialista em agilidade e acrobacias, e assim por diante. Cada Ranger tem sua responsabilidade única e eles se concentram em executar suas tarefas sem se intrometer nas atribuições dos outros Rangers. Da mesma forma, o princípio da Responsabilidade Única incentiva a dividir as responsabilidades em classes e métodos distintos para que cada um seja responsável por uma única tarefa.

Ao aplicar o Princípio da Responsabilidade Única, nosso código se torna mais legível, reutilizável e flexível. Uma classe que executa apenas uma tarefa é mais fácil de entender, testar e manter. Imagine um Ranger que dominasse todas as habilidades da equipe - a confusão resultante prejudicaria o trabalho em equipe e enfraqueceria a resistência do grupo a ameaças.

A singularidade também fortalece a coesão, um elemento vital para a eficácia tanto dos Power Rangers quanto do código de software. Quando cada classe tem uma única responsabilidade, a interação entre elas se torna mais clara e focada. Em vez de lidar com um código inchado e multifuncional, podemos confiar em classes especializadas, como Rangers bem treinados, para desempenhar seus papéis com excelência.

Princípio O - Aberto para Extensão e Fechado para Modificação

(Open/Closed Principle):

Os Power Rangers sempre enfrentam novos inimigos e desafios que exigem que eles se adaptem e usem novas estratégias para vencer. Apesar de enfrentarem essas mudanças, eles mantêm seus trajes e habilidades básicas (fechados para modificação). No entanto, quando uma nova ameaça mais poderosa aparece, eles podem combinar seus poderes ou receber novos equipamentos (abertos para extensão) para enfrentá-la. Da mesma forma, o princípio Aberto/Fechado sugere que o código deve ser aberto para extensão, permitindo a adição de novas funcionalidades sem modificar o código existente.

Quando nossos sistemas seguem o Princípio O, mudanças em requisitos futuros não precisam resultar em alterações arriscadas em partes funcionais do código. Em vez disso, podemos criar módulos ou componentes que podem ser ampliados sem causar efeitos colaterais indesejados. Isso é especialmente vital em ambientes onde a estabilidade é crucial, assim como a consistência das estratégias de combate dos Power Rangers é um dos principais fatores para derrotar as ameaças em curso.

Imagine que os Power Rangers, ao enfrentar um novo vilão, precisassem redesenhar completamente seus trajes e habilidades existentes para se defender. Isso enfraqueceria a equipe e criaria vulnerabilidades. Da mesma forma, alterações indiscriminadas em nosso código podem introduzir bugs inesperados e tornar a manutenção um pesadelo(e isso é a ultima coisa que você quer).

Princípio L - Substituição de Liskov

(Liskov Substitution Principle):

Imagine que cada Power Ranger representa uma classe base e suas habilidades individuais são subclasses. O Princípio L sugere que, assim como os Power Rangers podem se unir para formar o Megazord sem problemas, as subclasses devem ser capazes de substituir a classe base sem causar disrupções no funcionamento do sistema.

O Ranger Vermelho, por exemplo, pode ser visto como a classe base, e seus diferentes trajes e habilidades especiais, como o Turbo Ranger ou o Zeo Ranger, são as subclasses. Quando os Power Rangers se unem para enfrentar uma ameaça, a equipe continua a funcionar como uma unidade coesa, apesar das mudanças de forma e habilidades. Isso reflete o Princípio L, onde as subclasses são **trocáveis** com a classe base, mantendo a funcionalidade geral do sistema.

Essa coesão permite que nosso código seja mais flexível e adaptável. Da mesma forma que os Power Rangers podem ajustar sua estratégia combinando suas habilidades individuais, podemos criar sistemas que podem incorporar novas funcionalidades ou variações de classes sem causar problemas. Isso facilita a manutenção e a extensibilidade do software, tornando-o mais resistente às mudanças.

Princípio I - Segregação de Interface

(Interface Segregation Principle):

Os Power Rangers têm suas áreas de especialização e não precisam saber detalhes sobre as habilidades específicas de outros Rangers.

Imagine que cada Power Ranger representa uma classe em nosso código e suas habilidades são as interfaces que essas classes implementam. O Ranger Vermelho, especializado em combate, implementaria uma interface de combate, enquanto o Ranger Azul, especializado em tecnologia, implementaria uma interface de tecnologia. Cada classe (Ranger) tem uma interface clara e específica para interagir com o restante da equipe (código).

Aqui está o ponto crucial: assim como o Ranger Preto não precisa conhecer os detalhes das habilidades tecnológicas do Ranger Azul para combater eficazmente o mal, nossas classes não devem ser forçadas a implementar métodos ou funcionalidades que não são relevantes para sua especialização. Isso evita a chamada "poluição de interface", em que uma classe é sobrecarregada com métodos que não fazem sentido para ela.

O resultado é um código mais eficiente e mais fácil de manter. As classes se concentram em fazer o que fazem de melhor, sem desperdiçar recursos em funcionalidades desnecessárias. Isso também promove a reutilização de código, uma vez que as classes podem ser especializadas e, portanto, mais facilmente aplicáveis em diferentes contextos.

Princípio D - Inversão de Dependência

(Dependency Inversion Principle):

Independentemente de quem é o líder ou quais Rangers compõem a equipe em um determinado momento, eles confiam uns nos outros para enfrentar as ameaças intergalácticas. Eles compartilham informações e estratégias, sabendo que a unidade da equipe é mais importante do que qualquer detalhe específico sobre cada membro. Essa dinâmica é muito semelhante ao Princípio D - Inversão de Dependência na programação.

O Princípio de Inversão de Dependência nos ensina a depender de abstrações em vez de implementações concretas. Isso significa que, em vez de depender de detalhes específicos de classes ou módulos, devemos confiar em interfaces ou abstrações que definem um contrato geral. Assim como os Power Rangers não dependem de detalhes específicos sobre as habilidades de cada membro, mas sim da unidade da equipe e do objetivo comum, os desenvolvedores podem confiar em interfaces ou abstrações para alcançar a flexibilidade e a extensibilidade do código.

Vamos explorar essa analogia com mais profundidade

CONFIABILIDADE E FLEXIBILIDADE

Nos Power Rangers, a confiança mútua entre os membros da equipe é o que permite que eles enfrentem desafios aparentemente insuperáveis. Da mesma forma, ao seguir o Princípio de Inversão de Dependência, os desenvolvedores confiam nas abstrações e interfaces, o que torna o código mais confiável e flexível. Eles podem incorporar novas implementações que atendam às mesmas abstrações sem perturbar o funcionamento do sistema.

FOCO NO OBJETO COMUM

Os Power Rangers têm um objetivo claro - proteger o mundo contra ameaças. Eles não se perdem em detalhes sobre as habilidades individuais de cada membro. Analogamente, ao usar abstrações, os desenvolvedores se concentram no objetivo geral do sistema, não em implementações específicas. Isso ajuda a manter a clareza e a coesão no código.

ADAPTAÇÃO A NOVOS DESAFIOS

Assim como os Power Rangers enfrentam novos inimigos com diferentes poderes, os desenvolvedores podem enfrentar novos requisitos ou mudanças no ambiente de software com maior facilidade quando seguem o Princípio de Inversão de Dependência. Eles podem criar novas implementações que aderem às mesmas abstrações existentes, permitindo que o sistema se adapte sem reescrever partes substanciais do código.

MAIOR MANUTENIBILIDADE

A dependência de abstrações em vez de implementações concretas torna o código mais fácil de manter. É semelhante à ideia de que, nos Power Rangers, a equipe é mais importante do que qualquer Ranger individual. Quando um novo membro se junta à equipe, a dinâmica global permanece a mesma. Da mesma forma, ao adicionar novas implementações que seguem as abstrações definidas, o funcionamento geral do sistema permanece consistente

DESACOPLAMENTO

O Princípio de Inversão de Dependência promove o desacoplamento entre módulos ou componentes do código. Isso se assemelha ao desacoplamento entre os Rangers individuais e suas habilidades. Cada Ranger pode operar de forma independente, mas eles se unem para formar uma equipe coesa quando necessário.

RESUMINDO

Assim como os Power Rangers dependem da unidade da equipe e não de detalhes específicos de habilidades individuais, os desenvolvedores que seguem o Princípio de Inversão de Dependência dependem de abstrações e interfaces para criar código flexível, adaptável e fácil de manter. Essa analogia ajuda a destacar a importância da confiança nas abstrações e da colaboração entre partes do código em sistemas complexos de software.

CARREIRA

INTELIGÊNCIA ARTIFICIAL IRÁ ROUBAR SEU EMPREGO?

TEXTO
PAULO HENRIQUE
ILUSTRAÇÃO
NILLIA SOUSA
GABRIELA SOUZA
REVISÃO
MILENA MAIA

Por que estamos pensando isso?

As inteligências artificiais (IA's) têm evoluído bastante nos últimos tempos, principalmente com as novas versões do bastante conhecido ChatGPT, desenvolvido pela OpenAI, ou também a Mirage, IA generativa capaz de produzir imagens, e com elas se tornando comum em nosso dia a dia surge uma pergunta justa: "as IA's irão roubar nossos empregos?" "Elas podem um dia criar elas mesmas e então substituir o ser humano?" "É possível um futuro como o de Exterminador do Futuro?". Tenhamos bastante calma nessa hora, primeiro vamos entender o que são a maioria dessas IA's, já que este termo é bem generalista e acaba por disfarçar a grande área de estudo que existe e seus diversos modelos e algoritmos que realizam tarefas semelhantes a capacidade do ser humano de pensar e decidir.

MAS O QUE SÃO ESSES POSSÍVEIS LADRÕES DE EMPREGO?

Comumente, usamos o termo IA para definir toda e qualquer ferramenta computacional que age por ela mesma. Acontece que esse termo seria mais justo para representar toda a área de estudo que visa espelhar o comportamento humano de forma computacional, e dentro dessa área sub-tópicos surgem representando uma parte desse comportamento.

Por exemplo, Machine Learning, que está dentro da área de IA e faz com que a máquina copie a capacidade de aprender a classificar um dado de entrada a partir de um treinamento feito com outros tipos de dados, como a classificar a espécie de uma flor a partir de fotos dela, e um algoritmo que conhecemos que faz isso é o KNN (K-Nearest-Neighbors).

Porém, esse algoritmo se limita a simplesmente classificar objetos diante de uma referência, enquanto que em Deep Learning, temos um objeto mais complexo e que não busca imitar um comportamento, mas sim um elemento do corpo humano, que seria o próprio neurônio e seu funcionamento de forma matemática, que seria o Perceptron, e que esse assim como o neurônio pode ser aglomerado com muitos outros *Perceptrons* e criando assim uma rede neural computacional, mas ainda assim esse sistema é limitado quando tentamos realizar outras tarefas que não cabem a ele, daí existem outras ferramentas que se encaixam melhor para cada caso de uso. Além disso, o próprio ChatGPT é um LLM (Large Language Model), um modelo de linguagem natural com o propósito de entender e gerar texto, sendo uma rede neural, que adquire essa habilidade a partir do treinamento com uma grande quantidade de dados de forma supervisionada ou semi-supervisionada. E por mais que se pareça com um humano respondendo, não passam de respostas que se baseiam em pesos, que determinam o quão próximo e o quão longe isso está dos dados usados para gerar a resposta e que tem como referência os valores usados no treinamento da IA, e isso pode fazer com que a resposta final seja diferente caso mude algum detalhe do valor de entrada, existindo assim uma taxa de acerto que é previamente pensada e para garantir que a ferramenta possa ser utilizada essa taxa de acerto deve ser próxima de 100%, mas nunca realmente é 100%.

Então notamos que essas IA's ainda são super especificadas para uma determinada tarefa, e que dentro dessa mesma tarefa elas ainda cometem muitos erros e necessitam de ajustes a cada erro para diminuir a taxa de erro. Além de que, elas necessitam de conhecimento prévio para treinamento e funcionam mais como uma busca e junção, apenas alguns modelos realmente geram novos elementos, como a geração de imagens.

Realmente não teremos chances contra as máquinas?

Diante disso, ainda é muito cedo para dizer que a IA é capaz de tomar nosso emprego, mas pelo contrário, elas são ótimas ferramentas para auxiliar no processo de aprendizagem, com a devida curadoria do conteúdo gerado a fim de garantir que erros não sejam replicados. Claro que com a chegada dessas ferramentas, precisamos demonstrar mais ainda que realmente sabemos sobre os conceitos e menos sobre implementações diretas de código, pois é nisso que mora nossa maior habilidade, a criatividade.

Então, não precisamos nos preocupar com a velocidade com que as IA's vão acabar roubando nossos empregos, pois até lá, algo maior e mais rápido vai ter chegado até nós: o colapso ambiental.

CARREIRA ACADEMY

ALÉM DOS CHALLENGES: JORNADA NEURODIVERSA NO APPLE DEVELOPER ACADEMY

TEXTO
JOÃO VITOR IPIRAJÁ
ILUSTRAÇÃO
JOÃO VITOR IPIRAJÁ

Penso, logo existo. Pensar está sempre ligado a experiência viva. Nossa relação com o mundo.

Penso e, assim, existo aqui. Este pensamento está profundamente entrelaçado com nossa vivência e a maneira como nos relacionamos com o mundo. Embora nossos sentimentos sejam poderosos, com frequência recuamos diante da compreensão dessa reação biológica que carregamos em nosso âmago, preferindo contê-la em vez de questioná-la.

O desenvolvimento da inteligência emocional é uma raridade, uma vez que instituições tradicionais muitas vezes optam por educar mediante coação, medo e hierarquia, moldando excessivamente o indivíduo, sem lhe conceder escolha. Em contraste, as configurações ambientais da Apple Developer Academy rompem com essa tradição, possibilitando que as pessoas explorem uma miríade de perspectivas, descobrindo o que melhor funciona para elas.

Outra barreira que enfrentamos é a da linguagem, que nos deixa à mercê de potenciais perdas, especialmente quando tentamos empregar palavras precisas. No entanto, costumamos negligenciar a notável resiliência de nossa língua, que abriga inúmeras maneiras de formular perguntas e respostas. Essas podem ser encontradas até mesmo no que nem parece ser linguagem, como nos sussurros dos acontecimentos sonoros do cotidiano, que muitas vezes carregam consigo significados completos e profundamente impactantes. Formas de comunicação que se manifestam no campo oral, corporal e visual. Infelizmente, o acesso a essa lucidez frequentemente requer a exploração de metodologias complexas, algo reservado às pessoas privilegiadas com um vasto repertório literário-filosófico ou acesso a sessões terapêuticas possivelmente caras. Por consequência, investimento mais que fundamental, embora nem sempre acessível.

Conforme enfrentamos a rotina estressante, nossa percepção da vida gradativamente cede

espaço a mitos. Tentar explicar algo que transcende as fronteiras tanto da razão quanto da irracionalidade pode parecer absurdo. A solução, com frequência, parece ser a adoção de crenças e regras cada vez mais rígidas, que se chocam com nossa natureza fluida. Assim, nos vemos em um dilema: até que ponto estamos dispostos a tolerar o engano e que grau de lucidez estamos dispostos a aguentar?

Para enfrentar esses desafios, as estruturas de gerenciamento de projetos são indispensáveis quando se trata de trabalho em grupo, garantindo a consistência daquilo que está sendo construído. Entretanto, em um ambiente de trabalho, nem tudo pode ser planejado com métodos como Scrum ou Kanban. A fantasia e a idealização frequentemente entram em cena, devido à margem mínima para hipóteses e às nossas necessidades emocionais. Escapar desse ciclo é uma dolorosa desilusão, porém, é crucial para evitar ser engolido pelo abismo da culpa. Possivelmente, a resposta reside na exploração do familiar território do desconhecido, na direção do indivíduo que, em meio às muitas restrições impostas pela pandemia de COVID-19, foi arrancado do nosso repertório, buscando a habilidade de afetar e ser afetado.

> ***A cada entrevista concluída, me emocionava no microcosmo do diálogo.***

Quando comecei a elaborar este artigo, imaginei que se tornaria uma narrativa descritiva com uma consideração motivacional para promover a neurodiversidade. Passei por toda a burocracia e, durante 2 semanas, realizei entrevistas com 10 voluntários. A amostra incluía um grupo teste (com colegas com TOC, TDAH e TEA) e um grupo controle, para não enviesar minhas conclusões. Criei um roteiro de entrevista pensando em todo o tempo e estresse envolvidos no processo, dos relacionamentos às possíveis barreiras de comunicação, das realizações as frustrações.

No entanto, à medida que conduzia entrevistas com pessoas das diversas Apple Developer Academies pelo Brasil, minha abordagem desviou-se organicamente dos aspectos técnicos para explorar questões mais íntimas relacionadas à integridade, ética e vulnerabilidade. Me surpreendi com a força de deixar fluir o diálogo e puxar ganchos pertinentes do que era falado, enquanto anotava freneticamente sem tirar os olhos dos protagonistas. Depois de compilar tudo isso, fui tocado pela importância de compartilhar experiências e pela necessidade de estabelecer laços de confiança com familiares e amigos, a fim de preservar a saúde mental. A esquecida compaixão. A esfera do amor e caridade num sentido imaterial me fez ratificar que a verdadeira felicidade é compartilhada.

No fim das contas, percebi que o caminho começa com um simples “sim”. Aceitar essa jornada implica flutuar no escuro, sem controle ou engano, como ao cair em uma superfície abismal. A flexibilidade, a adaptação aos contextos específicos e o julgamento sensato revelaram-se a dinâmica mais precisa em meio à efervescência. Falo em se permitir ser humilde ao ponto de aceitar que nós somos uma metamorfose ambulante.

Em meio ao luto pelas tradições antigas, surge uma pergunta: “Por que não somos livres? Eu tenho uma reputação a prezar!”, Talvez seja porque simulamos regras e valores para manter nossas expectativas mínimas, preservando a esperança de que as coisas podem mudar. Ao mesmo tempo, somos participantes do jogo político. A ignorância ou inação não nos eximem de responsabilidade. A verdadeira humanidade compartilhada exige diplomacia e empatia, reconhecendo a singularidade dos outros. Não devemos nos esquivar ou tentar invadir o espaço alheio, devemos buscar trilhar o caminho do meio. Entretanto, vem essa fantasia da circunspecção em busca do prestígio e validação uniforme como consequência da prática. Melhor dizendo, praticar humanidade fala muito mais sobre o seu próprio bem estar do que carisma e a aprovação da massa.

Nossa comunicação deve ser assertiva, alinhando nossos valores e desejos com os do grupo. Devemos recuar quando o diálogo falha ou assim que dissipa a reciprocidade. Necessitamos buscar a opinião do destinatário para entender seus incômodos e motivações. Contudo, não devemos esquecer que as dinâmicas de poder e contradição persistem, e é importante estabelecer um nível de confiança que determine o grau de transparência necessário.

Em retrospectiva, ao longo de dois anos, percebemos que não nos tornamos quem somos apenas por meio da influência da Apple Developer Academy. Em vez disso, valorizamos a jornada de crescimento e aprendizado, apesar dos challenges que enfrentamos com as ferramentas únicas que nos foram mostradas. Cada passo à frente, embora repleto de riscos, representa uma vitória, e é essa questão que tem nos moldado ao longo do tempo: o que temos feito uns dos outros?

CARREIRA ACADEMY

COMO GENERALISTAS PODEM SE ENCONTRAR NO DESENVOLVIMENTO DE APLICATIVOS iOS?

TEXTO
AYSLANA RIENE

ILUSTRAÇÃO
DAVID AUGUSTO

REVISÃO
ELIS VIEIRA

Como uma generalista, definir carreira e foco sempre foi um desafio. É natural não se encontrar exatamente em uma área, mas em várias, que podem ser completamente distintas. O foco e o interesse estão em constante mudança, movidos por uma curiosidade contínua, que nunca se esvai, e nunca permanece em um mesmo tópico. Entender um pouco mais sobre o assunto me curou muitas dores, pois antes de me entender como generalista, sempre me forcei a pertencer somente a um local, uma área de atuação. O resultado disso é muito óbvio, falhei no momento em que foi me designado a decisão do curso superior, minhas opções eram: nanotecnologia, design de produto, gastronomia, artes visuais e física. Eu não estava indecisa, todos os cursos citados tinham o meu total interesse de desenvolvimento completo e profissional, e ainda têm. Acabei entrando para Engenharia de Telecomunicações, quando encontrei o desenvolvimento e migrei pra Engenharia de Computação e, finalmente, me encontrei no Academy. Logo nas etapas de seleção da turma, enviei um vídeo falando sobre esse aspecto da minha personalidade, o quanto eu queria me desenvolver em todas as áreas de interesse, e o quanto estava decidida em me tornar uma polímata, um dia.

A partir de agora, chegamos no ponto principal desse artigo, em como o desenvolvimento de aplicativos iOS se mostrou e continua sendo, um oásis de carreira para pessoas generalistas. Especialmente o Academy, lugar onde é possível testar diversas especializações, voltar e se desenvolver sem julgamentos. Aqui dentro pude me desenvolver nas áreas de programação, produto e design. Na minha experiência individual, decidi na minha passagem pelo Academy focar em design, visto que já trabalhava com programação.

Durante os projetos que participei, cheguei a trabalhar com modelagem 3D, UI, UX Research, realidade aumentada, copywriting, branding, ilustrações, artigos... a lista é grande e diversa. Mas o raciocínio é: estar incluída no ecossistema Apple me permite flutuar entre áreas a fim de atingir um mesmo objetivo: a publicação de um app na Apple Store.

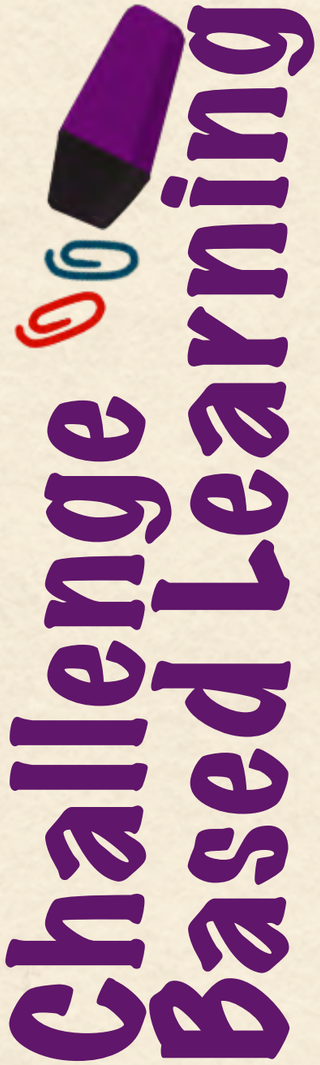
Para mentes criativas, o desenvolvimento de aplicativos iOS contempla todos os aspectos da nossa curiosidade. É preciso ser multipotencial e assumir todos os desafios necessários para concretizar uma ideia em formato de aplicativo, e, após isso, conseguir manter e divulgar o projeto. Ao descobrir esse universo, pude, finalmente, me encontrar como profissional e aspirar pela concepção de novos aplicativos, participando e colaborando ativamente em cada etapa do processo, experiência que eu, pessoalmente, recomendo à todos que se identificam com essa característica que de tão plural, nos torna singular.

METODOLOGIA DESENVOLVIMENTO

FANFICANDO CBL? A IMPORTÂNCIA DO USO DE METODOLOGIAS CBL E SCRUM

TEXTO
SAMANTHA EUFRÁSIO
ILUSTRAÇÃO
SARAH MADALENA
REVISÃO
ELIS VIEIRA

O desenvolvimento de aplicativos tornou-se uma parte essencial da nossa vida cotidiana. Desde aplicativos de mensagens instantâneas até aplicativos de gerenciamento financeiro, eles desempenham um papel crítico na maneira como nos comunicamos, trabalhamos e nos entretemos. Para garantir que esses aplicativos atendam às necessidades dos usuários e estejam sempre evoluindo, é vital adotar metodologias eficazes de desenvolvimento. Duas metodologias que têm sido destaque nesse contexto são o Aprendizado Baseado em Desafios (Challenge Based Learn) e o Scrum. Me segue aqui e vem entender mais sobre elas e suas vantagens.



O CBL é uma abordagem educacional que coloca os alunos no centro do processo de aprendizado, desafiando-os a resolver problemas do mundo real. Não há a hierárquia da sala de aula que somos acostumados a ter. Ao aplicar o CBL ao desenvolvimento de aplicativos, as equipes são incentivadas a compreender os problemas que os usuários enfrentam e a buscar soluções relevantes, você passa a compreender as dores do usuário que se deseja atingir além de permitir que haja um movimento de progresso, sem se basear em um crescimento linear ou cíclico, mas em progresso. Isso se traduz em uma série de benefícios:

RELEVÂNCIA CONTEXTUAL

O CBL permite que as equipes de desenvolvimento se envolvam profundamente nos problemas dos usuários. Eles precisam entender as necessidades e expectativas dos usuários antes de começar a criar um aplicativo. Isso resulta em soluções mais relevantes e centradas no usuário.

APRENDIZADO ATIVO

O CBL promove a aprendizagem ativa, incentivando a pesquisa, a colaboração e a resolução de problemas. As equipes são desafiadas a buscar informações, trabalhar juntas e desenvolver habilidades de solução de problemas - todas essenciais para o desenvolvimento de aplicativos bem-sucedidos.

INTERAÇÃO CONTÍNUA

O CBL incentiva a iteração contínua. À medida que as equipes descobrem novos insights e feedback dos usuários, podem adaptar e melhorar constantemente o aplicativo.

RESOLUÇÃO DE PROBLEMAS COMPLEXOS

O CBL desafia as equipes de desenvolvimento a abordar problemas complexos do mundo real. Isso não apenas impulsiona a criatividade, mas também desenvolve a capacidade de lidar com desafios complexos que podem surgir durante o ciclo de vida do aplicativo.

ENGAJAMENTO SUSTENTÁVEL

O CBL tende a manter as equipes engajadas e motivadas, pois elas estão trabalhando em problemas que consideram relevantes e significativos. Isso é fundamental para evitar a fadiga e o desgaste da equipe, mantendo a qualidade do trabalho ao longo do tempo.

DESENVOLVIMENTO DE HABILIDADES INTERPESSOAIS

O CBL promove a colaboração e a comunicação eficaz entre os membros da equipe, pois eles precisam trabalhar juntos para resolver problemas complexos. Essas habilidades interpessoais são valiosas não apenas para o projeto atual, mas também para o desenvolvimento pessoal de cada membro da equipe.

SCRUM

O Scrum é uma metodologia ágil que se baseia em princípios como iteratividade, transparência e adaptação contínua. Ele é amplamente adotado por equipes de desenvolvimento em todo o mundo devido à sua capacidade de entregar resultados de alta qualidade de forma eficaz e colaborativa. Seja no desenvolvimento de software, na gestão de projetos ou em outros contextos, o Scrum continua a ser uma ferramenta valiosa para alcançar o sucesso em projetos complexos e em constante evolução fazendo com que a gente coloque nossa flexibilidade em jogo. Aqui vai alguns motivos de por que ele é tão valorizado no mundo empresarial moderno.

FLEXIBILIDADE

O Scrum permite que as equipes de desenvolvimento respondam rapidamente às mudanças nas necessidades do usuário ou no mercado. Eles trabalham em ciclos curtos, chamados sprints, onde entregam incrementos funcionais do aplicativo a cada iteração.

TRANSPARÊNCIA

A metodologia Scrum promove a transparência em todas as etapas do desenvolvimento. Isso significa que os stakeholders têm visibilidade constante sobre o progresso do projeto, o que ajuda a evitar surpresas desagradáveis e garante que todos estejam alinhados com os objetivos.

COLABORAÇÃO EFETIVA

O Scrum enfatiza a colaboração entre as diferentes funções envolvidas no desenvolvimento de aplicativos, incluindo desenvolvedores, designers, testadores e gerentes de produto. Isso ajuda a garantir que todas as perspectivas sejam consideradas.

FEEDBACK RÁPIDO

As entregas frequentes durante os sprints permitem que as equipes obtenham feedback rápido dos usuários, permitindo ajustes contínuos no aplicativo para atender às necessidades em evolução.

MAIOR VISIBILIDADE E CONTROLE

O Scrum fornece uma estrutura clara de papéis, responsabilidades e reuniões regulares, como as reuniões diárias (Daily Scrum), reuniões de planejamento de sprint (Sprint Planning), revisões de sprint (Sprint Review) e retrospectivas (Sprint Retrospective). Isso resulta em uma maior visibilidade e controle sobre o progresso do projeto.

REDUÇÃO DE RISCOS

O Scrum ajuda a identificar e mitigar riscos precocemente, uma vez que os problemas são revelados em estágios iniciais e podem ser tratados antes que se tornem mais complexos e caros de resolver.

MAIOR SATISFAÇÃO DO CLIENTE

A entrega regular de incrementos funcionais do aplicativo permite que os clientes vejam o progresso e possam fazer ajustes nas prioridades ou requisitos conforme necessário. Isso leva a uma maior satisfação do cliente, pois eles têm a oportunidade de influenciar diretamente o desenvolvimento.

MELHORIA CONTÍNUA

As retrospectivas do Scrum são uma oportunidade valiosa para as equipes analisarem o que funcionou bem e o que pode ser melhorado no próximo sprint. Isso promove uma cultura de melhoria contínua que pode levar a resultados cada vez melhores ao longo do tempo.

Por fim, ao combinar o Aprendizado Baseado em Desafios (CBL) com o Scrum no desenvolvimento de aplicativos, as empresas podem criar uma abordagem altamente eficaz e centrada no usuário. Isso não apenas resulta em aplicativos de alta qualidade que atendem às necessidades reais dos usuários, mas também fortalece as equipes de desenvolvimento, desenvolve habilidades cruciais e mantém a agilidade necessária para prosperar em um ambiente de constante evolução tecnológica.

Essas metodologias não são apenas para grandes empresas, elas podem ser adaptadas para atender às necessidades de startups, pequenas empresas e projetos individuais (até mesmo no seu dia a dia ou no tão temido TCC). Em última análise, a chave para o sucesso é adotar abordagens que priorizem o aprendizado, a colaboração e a adaptação contínua, o CBL e o Scrum são ferramentas valiosas para alcançar esses objetivos.

DESIGN PROCESSO

UX RESEARCH: PROCURANDO PEÇAS QUE FALTAM EM NARRATIVAS

TEXTO
IEDA XAVIER
ILUSTRAÇÃO
DAVID AUGUSTO
LUCAS BAPTISTA
REVISÃO
MARÍLIA SOUSA
ELIS VIEIRA

Quando alguém diz que é um pesquisador de UX o que te vem à mente? Se eu te falasse que em suma o trabalho do pesquisador de UX é contar uma narrativa do usuário para com um produto e/ou serviço, mas para contar essa narrativa deve-se sanar todas as dúvidas que essa história trás. Aqui eu vou exemplificar algumas ferramentas que utilizamos para sanar essas dúvidas.

Pesquisadores de UX são cruciais componentes para o processo de design centrado no ser humano e para criar soluções que encontrem as expectativas do usuário e entregue valor. Quando queremos desenvolver um produto ou serviço passamos por alguns passos: Descobrir, Explorar, Testar e Monitorar e as metodologias se distinguem durante cada etapa. Na fase de descoberta, você tenta encontrar novas direções e oportunidades, utilizamos métodos de pesquisa generativa, como estudos de campo, Diary Studies, entrevistas, pesquisas, Participatory Design, Concept Testing. O ponto importante aqui é validar e descartar suposições e, em seguida, trazer os dados e insights para a equipe. Algumas dicas durante essa fase seriam conduzir estudos de campo, entrevistar usuários, Diary Studies e fazer benchmark para encontrar pontos fortes e fracos dos produtos de seus concorrentes, descubra o que os usuários mais gostam.

1

FASE EXPLORAÇÃO

Na fase de exploração, entendemos o espaço do problema e planejamos o escopo, atender apropriadamente as necessidades do usuário, utilizamos métodos de pesquisas formativos, como card sorting, tree testing, testes de usabilidade, remotos ou não, moderados ou não. Algumas dicas para essa fase, usar suas pesquisas para construir personas e escrever histórias do usuário, analisar o fluxo do usuário para encontrar maneiras de economizar tempo e esforço das pessoas, obter feedbacks sobre os fluxos do usuário em estágio inicial e usar card sorting para descobrir como as pessoas agrupam informações e adequar o fluxo de navegação e a organização da informação.

2

FASE TESTE

Na fase de teste, mensuramos o desempenho do produto contra si ou sua concorrências, utilizamos métodos de pesquisa sumativos como, Usability benchmarking, teste de usabilidade não moderados, A/B testing, clickstream/analytics e pesquisas. Algumas dicas para essa fase seriam fazer teste de usabilidade cedo e frequentemente, realizar uma avaliação de acessibilidade, pedir às pessoas que relatem suas interações e quaisquer incidentes interessantes ao usar o sistema ao longo do tempo, analisar suas redes sociais, conversar com usuários online e fazer testes de benchmark, se estiver planejando uma grande reformulação ou melhoria de medição, teste para determinar o tempo na tarefa, a conclusão da tarefa e as taxas de erro do seu sistema atual.

3

FASE MODERAÇÃO

Na fase de moderação, ao longo do ciclo de pesquisa e design para ajudar a entender os problemas existentes e procurar novos problemas. Analise os dados coletados e monitore as informações recebidas quanto a padrões e tendências. Boas práticas nessa fase seriam pesquisar clientes e usuários em potencial, monitorar métricas e descobrir tendências e anomalias e avaliar seu progresso, analise os canais de feedback recebidos periodicamente quanto às principais questões de usabilidade e áreas de problemas, procure pistas sobre o que as pessoas não conseguem encontrar, seus mal-entendidos e quaisquer efeitos não intencionais, coletar perguntas frequentes e tente resolver os problemas que eles representam.

Espero ter conseguido passar um pouco como é ser um pesquisador de UX e como ele é fundamental na ideação de um serviço/produto e ajudado com o um pouco das ferramentas que utilizamos para realizar nosso trabalho.

CÓDIGO TUTORIAL

CONSTRUINDO PONTES, NÃO APENAS PASTAS: DICAS SOBRE ARQUITETURA DE SOFTWARE PARA INICIANTES

TEXTO
EMILY MAIA

ILUSTRAÇÃO
HELENA OLIVEIRA

Este não é o relato de uma especialista em arquitetura de software - muito pelo contrário - mas sim de mais uma entre centenas de devs que também se perde na imensidão de informações que é a área da tecnologia.

Este artigo não revelará uma fórmula mágica para aprender arquitetura de software, pois isso levaria tempo e várias páginas. No entanto, posso te dar dicas tão valiosas quanto as que dizem por onde você deve ir. Dicas que dizem: "NÃO vá por esse caminho!".

Quando entramos no mundo do desenvolvimento de software, a vontade de aprender na prática (direto ao código) faz a gente se perder um pouco na maneira que pesquisamos as coisas. É tentador pular direto para a codificação, afinal, é a parte mais emocionante, certo? Porém, entender os princípios subjacentes da arquitetura de software e dos padrões arquiteturais , bem como a diferença entre eles, pode te economizar algumas horas de trabalho e dores de cabeça no futuro.

DICA #1

NÃO CONFUNDA ARQUITETURA DE SOFTWARE COM DESIGN PATTERNS

Arquitetura de Software e Design Patterns foram um dos conceitos que mais me confundiram no início da minha jornada. Por muito tempo, pensei que bastava escolher um Design Pattern (como MVC, por exemplo), criar as pastas correspondentes (Model, pasta View e pasta Controller) e pronto! Porém, qualquer mínima necessidade de integração com outras tecnologias (frameworks, API's, bancos de dados) me surgiam algumas dúvidas do tipo: *Isso se encaixa em Model ou Controller? Se houverem outras pastas separando essas integrações estarei 'saindo' da arquitetura? A arquitetura que estou utilizando é realmente a melhor para minha solução?*

Arquitetura de projeto se refere à estrutura específica de um projeto de software individual, enquanto os padrões arquiteturais são soluções gerais para problemas comuns de design de software. Os padrões arquiteturais, como MVVM(Model-View-ViewModel) ou REST (Representational State Transfer), são diretrizes amplas que podem ser aplicadas a diferentes projetos, independentemente de sua arquitetura específica. Não vou me aprofundar sobre eles pois não são nosso foco hoje.

Em resumo, uma arquitetura define como diferentes componentes do software se relacionam e interagem entre si. Envolve decisões sobre como dividir o código em módulos, como gerenciar a comunicação entre esses módulos, como garantir que o software seja escalável e fácil de manter a longo prazo, como será a documentação e os padrões de codificação, qual modelagem de dados será utilizada, define políticas de acesso e segurança, testabilidade e por aí vai.

A verdade é que a qualidade da arquitetura de software vai muito além da estrutura de pastas. Não importa se você tem dez pastas ou cem pastas em seu projeto, o que realmente importa é como essas pastas são usadas para organizar e conectar seu código.

Embora você não precise seguir à risca o processo de desenvolvimento de arquitetura de software toda vez que iniciar um novo projeto (os primeiros tendem a ser mais simples e tudo bem!), é importante que você saiba quais etapas existem nesse processo e como elas impactam o produto final.

DICA #2

NÃO FIQUE SÓ NA TEORIA

No desenvolvimento de aplicativos, existem algumas arquiteturas que são comumente utilizadas, como ***Arquitetura em Camadas*** (Layered Architecture), ***Arquitetura Limpa*** (Clean Architecture), ***Arquitetura em Cebola*** (Onion Architecture) e muitas outras! Dê uma lida sobre elas, entenda seus objetivos e benefícios e arrisque implementando-as! Esse processo com certeza vai te levar a utilizar alguns Design Patterns e isso será essencial para você identificar as diferenças entre eles.

DICA #3

SOLIDIFIQUE SUAS BASES

Esta provavelmente é a dica mais importante deste artigo. Se você é iniciante, foque no domínio da linguagem de programação, conheça a IDE onde você irá trabalhar, aprenda a tratar erros, Programação Orientada a Objetos, utilizar API's e frameworks...depois de se sentir confortável com esses tópicos, você com certeza estará numa posição melhor para explorar arquitetura de software.

LEMBRE-SE!

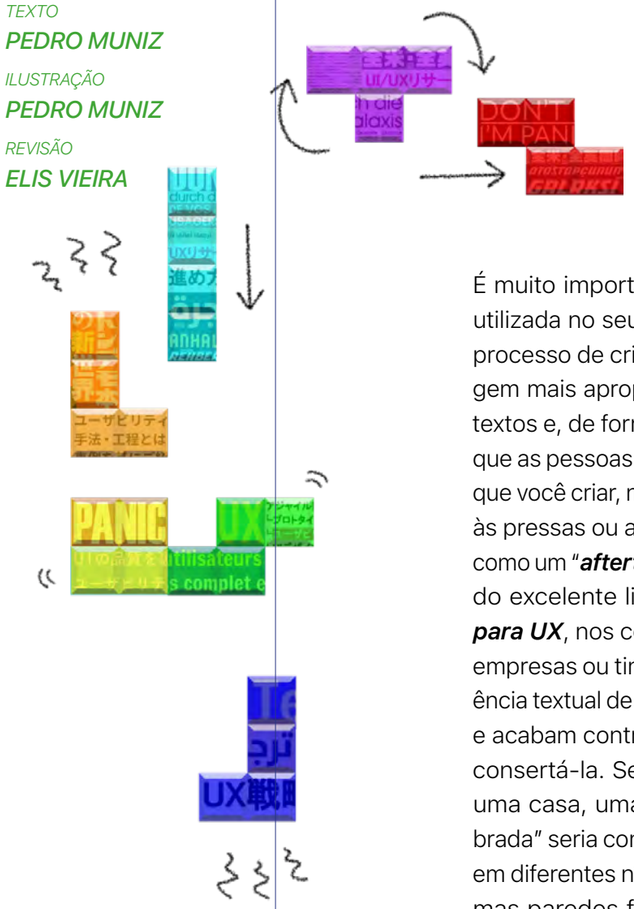
Nada te impede de aprender em paralelo, mas evitar frustrações decorrentes do pensamento de insuficiência ou inaptidão para aprender algo é importante. Tudo tem seu tempo!

Por fim, lembre-se de que o processo de aprendizado é incrivelmente valioso. Cada pequeno passo representa uma conquista, e cada instante dedicado a essa jornada é um investimento no seu desenvolvimento como programador. Ao respeitar e abraçar cada etapa, você tornará sua jornada na programação mais gratificante e menos desgastante. Portanto, prossiga com confiança, mantenha a paciência e tenha em mente que o tempo é um aliado essencial nesse percurso!

DESIGN PROCESSO TUTORIAL

UX WRITING: DICAS PARA CONSTRUIR UMA ARQUITETURA DE CONTEÚDO BÁSICA PARA O SEU APP

TEXTO
PEDRO MUNIZ
ILUSTRAÇÃO
PEDRO MUNIZ
REVISÃO
ELIS VIEIRA



"UX Writing", também conhecido como "Content Design", ou, em bom português, "Design de Conteúdo", é a área do Design de Experiência dedicada ao planejamento e elaboração do conteúdo textual de um produto, serviço ou, de forma bastante geral, da ***experiência*** que você estará propondo para determinadas pessoas. Em tradução livre, "UX Writing" equivaleria a "Escrita para a Experiência de Usuário", apesar de ir muito além do que está escrito, já que toca em todo o conteúdo do produto que é consumido por alguém, incluindo como este conteúdo está disposto em uma tela. Neste texto vamos nos focar no planejamento de conteúdo para aplicativos digitais, mas os princípios do UX Writing ou Design de Conteúdo podem ser adaptados para os mais diferentes projetos que envolvem produção textual. Depois desta curta introdução, você vai ser apresentado a uma ferramenta bem útil para o planejamento dos conteúdos textuais, que podemos chamar de ***Tabela de Voz***; e, logo em seguida, um apanhado de dicas básicas (mas importantíssimas) para desenvolver a ***voz do seu app***. Essas dicas podem ser usadas junto com a Tabela de Voz ou independente dela.

É muito importante pensar na linguagem utilizada no seu app desde o início de seu processo de criação. A definição da linguagem mais apropriada, o planejamento dos textos e, de forma mais geral, do conteúdo que as pessoas poderão consumir nas telas que você criar, não deve ser algo adicionado às pressas ou apenas no final do processo, como um "***afterthought***". Podmajersky, autor do excelente livro ***Redação Estratégica para UX***, nos conta como frequentemente empresas ou times percebem que a experiência textual de seu produto está "quebrada" e acabam contratando um ***UX Writer*** para consertá-la. Se pensarmos no app como uma casa, uma experiência textual "quebrada" seria como ter paredes deterioradas em diferentes níveis. Se há poucos buracos mas paredes foram construídas de uma forma robusta, de forma que os buracos não afetem a encanação ou a instalação elétrica, por exemplo, o reparo será mais fácil. Ou seja, se seu app foi construído com uma terminologia, uma voz e uma arquitetura da informação consistentes, além de formas o trabalho de manter, atualizar e internacionalizar o conteúdo, além de consertar algumas palavras mal colocadas aqui e ali fica naturalmente mais fácil. No entanto, quando esse trabalho de fundamentação não foi feito e os buracos afetam a própria estrutura da experiência e apenas consertar algumas palavras (aquela revisão básica) não vai adiantar: é preciso fazer um verdadeiro (re)trabalho de engenharia.

Um bom planejamento de conteúdo pode determinar o quão acessível é o seu app (e, consequentemente, quais pessoas ficarão de fora da sua experiência), quantas pessoas ficarão com vontade de voltar a usá-lo (ou não), e até quantas pessoas o recomendarão para outras (ou, pelo contrário, farão propaganda negativa). O texto certo em determinado botão, assim como a distribuição dos elementos na tela, podem fazer a diferença no momento em que alguém está finalizando uma compra (ou acaba desistindo dela).

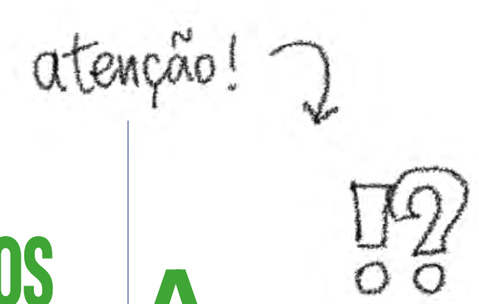
FERRAMENTA PRA QUEM TEM PRESA

Caso seus prazos estejam apertados no momento (uma situação bastante comum hoje em dia, somos obrigados a admitir), mas você ainda quer fazer um planejamento básico do conteúdo textual do seu app, uma das muitas ferramentas interessantes que você pode usar é a "Tabela de Voz", desenvolvida por Podmajersky e adaptada para cá. A tabela de voz contém um conjunto de regras de tomada de decisão e orientação para alinhar o conteúdo de UX às necessidades da organização e do usuário. Quando o conteúdo UX estiver sendo elaborado, a tabela de voz ajudará a identificar o que pode torná-lo melhor.

O que chamamos de ***princípios do produto*** aparecem no topo de uma coluna. Então, para cada princípio, cada um dos seis ***aspectos da voz*** é definido em uma linha diferente: conceitos, vocabulário, verbosidade, gramática, pontuação e letras maiúsculas. As variações entre colunas correspondem à diferença entre a ***voz*** e o ***tom***. A voz é consistente: é possível reconhecer que há uma escolha das palavras usadas ao longo de toda a experiência. O tom varia de acordo com a necessidade, entre diferentes partes da experiência: ele será diferente em uma mensagem de erro, em uma notificação, em um momento de celebração, e assim por diante. A voz do seu app deve ser reconhecível apesar dos diferentes tons, assim como somos capazes de reconhecer a voz de uma pessoa querida quando a ouvimos falar tanto com um amigo quanto com um superior no trabalho.

Os princípios do produto são os fundamentos da tabela de voz. Eles definem o que o produto pretende ser para seus usuários, e podem variar de acordo com o produto e as necessidades da organização.

Pense nos momentos em que haverá comunicação entre seu produto ou sua organização e as pessoas que o utilizarão. Por exemplo: quaisquer momentos de solução de problemas pode gerar um princípio que aparecerá no topo de uma das colunas equivalentes a um tom de voz. As campanhas de ***marketing*** podem ser outro momento de comunicação que gera mais uma coluna e mais uma variação de tom. Os aspectos da voz aparecem à esquerda, iniciando cada linha da tabela.

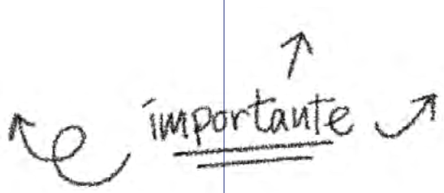


OS CONCEITOS

são as ideias ou assuntos que a organização quer enfatizar a qualquer oportunidade. Pergunte-se qual é o papel que vocês, enquanto organização, querem que o produto desempenhe na vida das pessoas que o utilizarão? Registre as ideias que devem ser transmitidas, independente da terminologia específica que é escolhida.

JÁ O VOCABULÁRIO

especifica os termos que são tão importantes para a experiência que nos permitem identificar sua personalidade. Para um trabalho de planejamento mais completo, o vocabulário que entra na Tabela de Voz não substitui uma lista de terminologias mais robusta e mais completa, como um léxico ou glossário para o produto. Essa lista de terminologias define os termos que têm um significado especial para aquela experiência específica.



A VERBOSIDADE

define a quantidade adequada de texto mostrado. Quando preparamos qualquer texto para um app ou serviço, esse texto não deve de atrapalhar o percurso de quem está usando o aplicativo (pelo contrário), mas geralmente ele também não está lá só para ser "saboreado", por simples prazer. Escrever um texto muito sucinto quando se espera mais informações pode bloquear quem está usando o app, e o contrário também é verdadeiro. Não deixe de levar em consideração também o tipo de tela na qual o texto será inserido: se ela é maior ou menor. E, como sempre, tudo vai depender do contexto. Em um aplicativo cujo objetivo é transmitir um ar de elegância e a importância associada a um serviço ou local, por exemplo, pode-se escolher não poupar palavras em descrições e "enfeitar" o texto com advérbios e adjetivos. Entretanto, nem sempre estaremos lidando com contextos como esse.

A GRAMÁTICA

define as estruturas gramaticas apropriadas, mais simples ou mais complexas, dependendo do contexto. As linguagens humanas nos permitem variados tipos de expressões para dizer coisas similares. Em geral, para maximizar a usabilidade, estruturas gramaticais simples funcionam melhor para a maioria dos propósitos. Em inglês, isso significa frases simples de sujeito-predicado ou comandos tipo verbo-objeto, como "O ônibus aceita troco correto e passes de trânsito" e "Adicione dinheiro ao seu passe de trânsito". Mas não podemos buscar maximizar a usabilidade e acabar criando um tom robótico e impessoal demais.

A PONTUAÇÃO

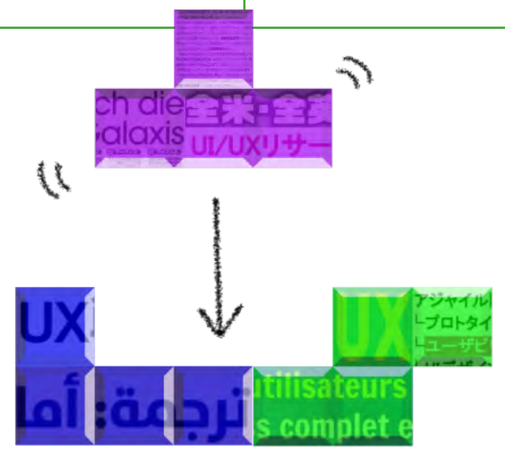
define o tipo de pontuação que deve ser usada ou nunca deve ser usada em todos os textos do app. Você pode, por exemplo, proibir o uso de determinados sinais de pontuação para facilitar o uso de leitores de tela, além de definir quando usar vírgulas, como usar travessões e muito mais.

Princípios de Produtos / Aspectos da Voz	**Descontraído** Momento: Onboarding	**Jovial** Momento: Telas iniciais	**Simples e Direto** Momento: Textos de natureza legal	**Intrigante** Momento: Marketing, redes sociais.
Conceitos *Ideias e assuntos que a organização quer enfatizar a qualquer oportunidade. Reflete o papel que a organização quer que a experiência tenha na vida da pessoa usuária*	Exemplos: Praticidade, clareza, descontração	Escreva aqui as ideias e assuntos que você ache mais importante para a experiência que você quer proporcionar :)	Escreva aqui as ideias e assuntos que você ache mais importante para a experiência que você quer proporcionar :)	Exemplos: Jovialidade, quebra da rotina, despertar a curiosidade...
Vocabulário *Termos que são importantes para a experiência que nos permitem identificar sua personalidade*	Exemplos: Não usar palavras em inglês, exceto quando...	Exemplo: Nunca usar sequências de palavras que possam gerar cacofonia ou confusão para leitores de tela (Ex.: "Há a")	Exemplo: Não usar estrangeirismos, a não ser os amplamentes já adotados, já pertencentes ao jargão usado...	Além de escrever termos e palavras aqui, você pode estabelecer regras. Na verdade, é interessante construir um léxico mais longo para o seu app, separado desta tabela.
Verbosidade *Define a quantidade adequada de texto mostrado*	Exemplos: Evitar frases muito longas; Usar o mínimo de texto possível.	Exemplo: Evitar frases muito longas.	Exemplo: Evitar muitos adjetivos, exceto se reforçar a ideia de segurança...	Exemplos: Todas as frases devem ser curtas ou o mais curta possível.
Gramática *Define as estruturas gramáticas apropriadas, mais simples ou mais complexas, dependendo do contexto*	Exemplo: Preferência por sentenças simples (sujeito-predicado), comandos claros e simples.	Exemplo: Aqui e em todo o app: Usar verbos no infinitivo para botões de ação. Ex.: Criar pedido.	Exemplo: Usar apenas o pronome "nós", com verbos corretamente conjugados – não usar "a gente"	Exemplos: Sentenças simples e completas. Verbos e comandos simples.
Pontuação *Define o tipo de pontuação que deve ser usada ou nunca deve ser usada nos textos do app*	Exemplo: Aqui e em todo o app: Nenhum título usa pontuação, não usar hífens, en ou em dash, parênteses, etc.	Exemplo: Aqui e em todo o app: Sentenças completas que não são títulos incluem pontuação (isso não vale em botões).	Não usar pontos de exclamação. Pode-se usar ponto e vírcula ou dois pontos, se necessário.	Exemplo: Pode-se usar pontos de exclamação, com parcimônia. Estrangeirismos podem ser usados à vontade.
Capitalização Define o uso de letras maiúsculas e/ou caixa alta em todo o app	Exemplos: Aqui e em todo o app: Maiúsculas no início de títulos, em botões e headings.	Igual à coluna anterior	Igual à coluna anterior	Às vezes uma diretriz pode ser a mesma entre uma ou várias colunas! :)

E, por fim, o último aspecto da voz, a ***capitalização*** ou o uso de letras maiúsculas define o uso de letras maiúsculas ou caixa alta em todo o app. Pode-se argumentar que a pontuação e capitalização fazem parte do design visual e tipográfico do produto e, por isso, não seriam responsabilidade do ***UX Writer***, mas eles são essenciais para determinar a qualidade da experiência de uso.

Você pode criar uma versão desta tabela na sua ferramenta favorita (um Notion ou Figma da vida) e, se possível, trabalhar junto com o seu time ou com as pessoas da sua organização para preenchê-la.

O trabalho não precisa necessariamente ser feito em conjunto e em tempo real, mas quanto mais você conversar com as pessoas envolvidas em seu projeto, tirar dúvidas e esclarecer qual é a visão que se tem do app que vocês estão desenvolvendo, melhor será o resultado. Seja como for, separe um tempo com seu time para apresentar a tabela e fazer com que todo mundo esteja alinhado.



ONE MORE THING...

ALGUMAS DICAS PARA QUEM ESTÁ COM MUITA PRESSA MESMO

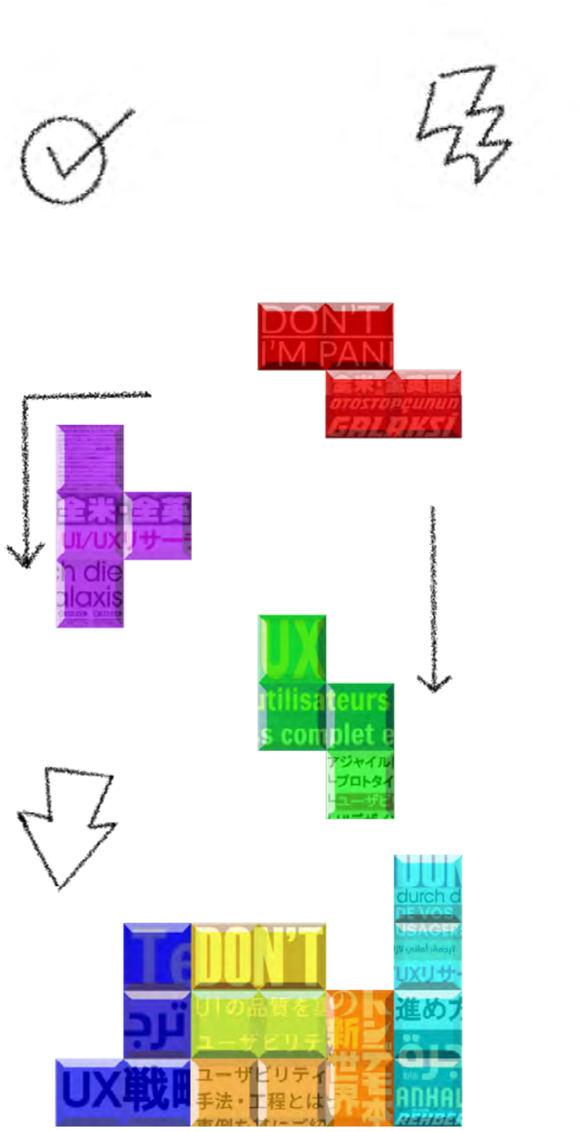
Se o tempo não for suficiente nem mesmo para planejar uma tabela de voz e alinhá-la com sua equipe, reúno logo a seguir um "kit basicão" de dicas, talvez bastante básicas para pessoas mais experientes, mas ainda assim tão importantes que nunca deveríamos perdê-las de vista ao desenvolver qualquer produto ou serviço.

Lembre-se que o conteúdo textual afeta a interface do seu app como um todo, e é crucial para uma boa experiência de navegação, de solução de problemas, de suporte e de ***marketing***, para citar apenas alguns dos contextos mais importantes.

Primeiro de tudo, pense em seu app (ou no serviço que você está oferecendo) como uma ***conversa*** entre você (ou a sua organização) e as pessoas que o utilizarão. É como se apps também contassem histórias, conduzindo quem os utiliza por determinados caminhos, que podem ser claros ou obscuros, prazerosos, frustrantes ou irritantes. A partir disso, desenvolva minimamente uma voz para o seu app (então varie o tom dessa voz de acordo com o contexto ou momento em que a comunicação acontece - provavelmente o tom será diferente no ***onboarding*** e nos termos de uso). Responda questões como: O que o meu app diria? E o que ele não diria? O meu app é excitante e divertido? Ou é um app que precisa transmitir segurança?

Ao desenvolver uma tela específica, é preciso pensar na coisa mais importante que o usuário precisa saber naquele momento. Esse é o propósito daquela tela. Para expressá-lo, devemos levar em conta a hierarquia da informação: como os elementos são ordenados para conduzir quem está tendo a experiência pelo melhor caminho. Ter o propósito da tela claro ajuda a determinar o que manter e o que tirar dela, bem como a ordem do que fica. Com muitos passos em um fluxo, é preciso definir com clareza tanto o propósito do fluxo como um todo, quanto o propósito de cada passo, ou seja,cada tela. Essa clareza evita a criação de passos desnecessários.

Quando você estiver escrevendo alertas, escreva textos que realmente ajudem as pessoas, e que sejam absolutamente claros. Alertas são interrupções no percurso, por isso devem ser extremamente úteis e claros. É bom sempre ser específico nas ações dos botões de alertas (evitar o simples



> ***Pense também fora do seu app: quando as pessoas o utilizarem, elas estarão relaxados na cama ou apressadas no trânsito? O app receberá toda a atenção delas ou ela deverá ser dividida com outras tarefas?***

"sim" e "não"). Os botões devem ser tão claros que, mesmo que o usuário não leia o alerta e leia apenas os botões, ainda assim saberá o que está acontecendo. Não se intimide com o vazio: crie espaços vazios úteis no seu app, eles podem significar muitas coisas, até celebração, como quando você completa uma lista de tarefas. Tudo vai depender do contexto.

Deixei para o final o que talvez seja o mais importante sempre: tente ***praticar a empatia***. Procure se colocar no lugar das pessoas que estarão participando dessa conversa ou ouvindo essa história. O que elas esperam? Com que tipo de linguagem elas se sentiriam melhor? E que tipo de linguagem melhor serviria também a sua organização? Que sentimentos você deseja despertar nessas pessoas a cada etapa do percurso? Pensar nessas coisas ajuda a determinar o vocabulário usado. Com isso você pode fazer uma lista de termos usados com mais frequência (e com termos que não devem ser usados).

Tente escrever para todo mundo: use uma linguagem simplese pense em descrições claras e objetivas para pessoas que usam leitor de tela. Lembre-se que não usar gêneros específicos faz com que mais pessoas se sintam à vontade para usar o seu app. Pense também na responsividade dos elementos da sua interface no caso de necessidades ligadas à localização (às vezes o texto em outra língua pode ser bem maior do que na língua original). Em suma, pense com cuidado em toda a linguagem que você usar. Na dúvida, fica uma última dica: sempre leia seus textos em voz alta e se pergunte se o seu app ou produto estaria ***dizendo aquilo daquela forma naquele momento*** - isso ajuda a encontrar a linguagem adequada.

Aqui você encontra uma versão editável da tabela para baixar e usar como quiser!

REFERÊNCIAS E RECOMENDAÇÕES

artigo!

Livro **Redação Estratégica para UX: Aumente engajamento, conversão e retenção com cada palavra**, de Torrey Podmajersky. São Paulo, Novatec Editora, 2019.

Livro **Nicely Said: Writing for the Web with Style and Purpose**, de Nicole Fenton e Kate Kiefer Lee. Estados Unidos, Peachpit Press, 2014.

Vídeo da WWDC 2022 intitulado "Writing for interfaces":

[Writing for interfaces - WWDC22 - Videos Apple Developer](https://developer.apple.com/videos/play/wwdc2022/10037/)

Episódio 8 do podcast *Design +*, intitulado "Amandine Agic & Camille Promérat - L'UX Writing: quelle valeur ajoutée dans votre process design?"

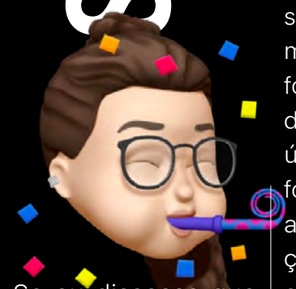
PROCESSO ACADEMY

WWDC: CAMINHO DA IDEIA A IDEAÇÃO

TEXTO
SARAH MADALENA
MATEUS CALISTO
MOYSES AZEVEDO

DIAGRAMAÇÃO
LUCAS BAPTISTA

SARAH



Se eu dissesse que tive plena certeza do que estava fazendo ao longo do processo, tanto na pré escolha da ideia que eu iria submeter como durante o momento da idealização, estaria mentindo. Até hoje tenho escrito a ideia que tive e que coloquei no Notas do iPhone durante um momento de insônia: "Ideia WWDC: um jogo sobre afeto

Tem uma flor e dois botões em formato de coração, mas um é partido. A meta é dar afeto suficiente para a flor viver feliz. Colocar uma introdução sobre afeto pra contextualizar comigo."

Se eu puder resumir meu projeto em algumas palavras seriam: afetividade, tristeza, amor e identidade. Sou uma das milhares de pessoas que sofre com ansiedade e depressão, acabando sempre por externalizar meus sentimentos de forma prática e através de desenhos. Logo, a única certeza que tive foi a de querer mostrar algo que tivesse relação comigo, em que eu pudesse mostrar um pouco do que sou e como me sinto. Então se eu pudesse dar uma dica seria essa, fazer algo que converse com quem você é.

Para por em prática minha ideia, fiz uma espécie de mini versão do CBL, onde coloquei: minha ideia geral, alguns pontos principais que eu gostaria de ter no meu aplicativo, como eu imaginava que ele seria e fiz vários questionamentos (Guiding Questions e Guiding Activities). Formulei e reformulei como ficaria o aplicativo, principalmente após mostrar para várias pessoas como ele estava, fazendo muitos questionamentos e coletando feedbacks de melhoria, bem como de coisas que davam ou não para serem feitas através do código no período de tempo que o aplicativo precisava ser entregue.

Por fim, tentei dar vida aos meus assets colocando tudo que estava sentindo na época do processo de desenvolvimento do aplicativo, onde, coincidentemente, estava tendo crises de ansiedade e refletindo bastante sobre a importância da afetividade na vida das pessoas, principalmente da minha. Outras duas coisas que fizeram total diferença no meu processo foi ter começado a rascunhar como imaginava que seriam os assets antes de sair as datas da submissão e não ter vergonha de pedir ajuda para descobrir como fazer coisas que eu precisava por no meu aplicativo, mas que não sabia fazer.

Mas, no fim de tudo isso, como se resumiria o que é o aplicativo? O Beautiful Scars é um jogo interativo de narrativa que, através do Mandacaru, cacto símbolo do nordeste brasileiro, fala sobre como a afetividade interfere na vida das pessoas. Com o objetivo de mostrar como o amor, principalmente o amor próprio, pode modificar a vida, o jogo usa o Mandacaru (carinhosamente batizado de Mandá, como referencia ao meu segundo nome "Madalena") como uma analogia para ilustrar a forma pela qual lidamos com os sentimentos. Você deve coletar os corações partidos e usá-los para ver as dificuldades pelas quais o Mandá passou em diferentes fases de sua vida, sabendo os efeitos negativos que a falta de afeto causa nas pessoas. Depois você deve coletar os corações cheios para ver o que o amor e o afeto causaram de bom na vida de Mandá, mostrando os benefícios da afetividade.

CALISTO

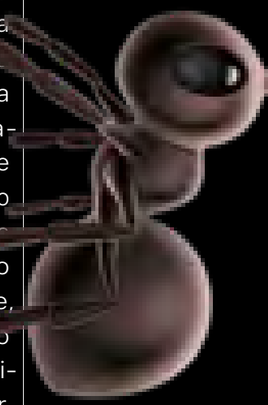
Formiga? Como trazer esse pequeno inseto para um desafio tão grande assim?

Então, ao iniciar as inscrições, passei longos dias pensando: sobre o que vou fazer?

Isso me sobrecarregava muito, pois pensava que, se eu não escrevesse algo considerado vencedor, isso me frustraria. Os dias passavam e nada vinha relacionado a qualquer tema, o que me estressava. Mas tinha uma coisa que me ajudava a relaxar: andar com minha filha de quatro patas, a Malu.

Malu é uma cachorrinha preta, grande, orelhuda, pernuda e talvez pareça um pãozinho. Ela chegou de repente quando me mudei com meu marido para uma nova casa, era filha dele e agora mini enteada. Cachorrinha curiosa, comportada, medrosa, mas que amava passear. Quando podia, eu a levava para passear num campo de areia bem grande, onde moro. Ela, muito serelepe, adora cheirar tudo, inclusive formigueiros, e não seria diferente ao avistar um grande formigueiro que ficava no escanteio da quadra. Um morro bem alto, cheio de formigas que carregavam folhas muito maiores que ela. O resto dessa história é conhecido: elas picaram a Maluzinha por estar cheirando e incomodando onde não devia.

Com isso, pensei: "Como seres tão pequenos podem incomodar seres tão maiores?"



Eu não tinha nada, e de repente, tudo. Graças ao passeio com a Maluzinha, pude notar um grande trabalho em grupo das formigas e consegui refletir no dia a formigas juntas dão passos minúsculos e singulares, porém carregam grandes folhas e animais dezenas de vezes maiores que elas, tudo graças ao trabalho em grupo; por isso chamam-se super organismos. E acho que minha jornada tornou-se muito especial com esse ponto, pois toda a minha história tem se tornado extremamente reflexiva sobre a importância do trabalho em grupo e da razão de dar pequenos passos e se contentar com a conquista deles. O ser é individual, mas o que nos torna humanos é a capacidade de construir comunidades além de desenvolver sentimentos e crescimentos sociais que são extremamente ricos e valorizam aspectos da vivência, da história. Desenhei, escrevi e montei todo um aplicativo para mostrar que somos como as formigas e que só conseguimos construir nossa colônia, carregar nossos pesos e realizar nossas atividades quando estamos em sintonia e parceria com os mesmos. Podemos fazer tudo sozinhos, mas quando temos boas companhias, sejam elas parentesco sanguíneo ou não, eliminamos a solidão e damos entrada ao progresso e a companhia.

MOYSES

Meu app explica a teoria das variáveis aleatórias e sua relação com a função Gaussiana. Ele usa recursos visuais para ajudar os usuários a entender como essa distribuição funciona, incluindo um tabuleiro de Gauss interativo. Além disso, ele relaciona a curva Gaussiana com a falácia da meritocracia.

Em um projeto é necessário. Muitas HardSkills só podem ser adquiridas através de muito estudo, mas para a ideação não necessita de grandes coisas. No meu caso eu só usei empatia, ódio e imersão.

Empatia

A empatia é parte fundamental e mais importante para um projeto, na minha opinião. Ela é responsável pelo significado e pelo ressignificado. O exemplo prático é como eu escrevi este artigo. Fiz-me várias perguntas:

Será que você vai ganhar a próxima WWDC?

Quem vai ler o meu artigo?

Quem eu quero realmente ajudar?

Uma pessoa como eu, só que no passado?

Será que eu realmente consigo escrever o que eu quero dizer?

Pessoas diferentes de mim, que eu quero mostrar o meu ponto de vista?

Como este artigo pode ser realmente útil?

Por que eu quero escrever este artigo?

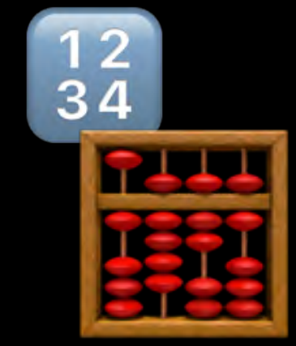
Com base nas minhas perguntas, traço um perfil que desejo atingir. Pretendo, por exemplo, atingir uma pessoa minimamente curiosa para ler o primeiro parágrafo e com marra suficiente para se sentir desafiada pela primeira pergunta. Essa é a pessoa que eu acredito ter capacidade para ganhar a próxima wwdc.

Tento, através das perguntas, realmente me conectar com um público específico, no caso, com você, que chegou até aqui. Imagino, por exemplo, que você ainda está empolgado por ter passado no processo seletivo do Academy e já deve ter ouvido falar do evento da WWDC, das premiações, da possibilidade de visitar a sede oficial da Apple, ganhar o tão sonhado box com o casaco e a carta dizendo parabéns, você foi um dos escolhidos.

Ódio

Normalmente ouvimos que o ódio é o contrário de amor, mas eu acredito que seja a apatia. Tudo que envolve o ódio normalmente possui uma paixão ou amor por trás. Então, quando não acho algo que me importe, busco algo que odeie. Com isso, tenho certeza de que vou me conectar com o ódio em comum de algum público. Você que está lendo recebeu alguma ligação de algum bot que falava alô e desligava?!

Por que existe um bot que liga aleatoriamente para a gente e diz alô e desliga?



Meu amigo, você sente algum ódio disso?! Amor, paixão e ódio unem muitos corações. Eu, por exemplo, trabalharia quase de graça só para resolver esse problema. Eu até pagaria para alguém resolvê-lo para mim.

Eu usei exatamente isso para idealizar meu

programa da WWDC. Eu estava em uma época de utilizar muito transporte público e ouvir algumas pessoas falando sobre meritocracia enquanto andavam com o CARRÃO dado pelos seus pais, me matava por dentro, então tentei expor isso através do meu app. Lá tentei demonstrar através de estatísticas como a meritocracia não faz sentido em países com desigualdade social. Eu basicamente desenhei isso de forma interativa.

Imersão

Quando nos expomos a um novo ambiente, nosso cérebro demora um tempo para criar novas conexões. E é engraçado que ele faz isso fisicamente através de algo chamado neuroplasticidade. Eu me aproprio muito dessa capacidade. No primeiro momento, eu mergulho em um tema sem medo de não aprender, até me sentir saturado dele. Eu passei, por exemplo, 4 dias lendo e assistindo sobre meritocracia e assuntos relacionados. Após isso, parei de pensar, e as ideias mais legais apareciam no meio do meu descanso. Não pense que você não é capaz de alguma coisa; acredite, para fazer qualquer coisa, você só precisa de tempo.

DESIGN TUTORIAL

LEVELS ATÉ O GOOD ENDING NO DESENVOLVIMENTO DE JOGOS

TEXTO
LETICIA DUTRA
ILUSTRAÇÃO
LETICIA DUTRA
ISADORA FONTENELE

Glossário

tasks: tarefas ou atividades que os membros realizam.

game: palavra ‘jogo’ em inglês

levels: palavra ‘níveis’ em inglês (no texto, muito usado para se referir à níveis nos jogos)

O desenvolvimento de jogos é como uma longa e árdua jornada, você precisa enfrentar o frio cortante, o deserto escaldante e a enorme correnteza de tasks que lhe aguarda (metaforicamente, claro). Emocionante, como o próprio game que você pode desenvolver.

Não se assuste com essa introdução dramática, desenvolver jogos é extremamente divertido, e você vai entender melhor a árdua jornada no momento em que tiver que fazer um Sprint Planning.

Mas não tema! Esse texto tem a intenção de guiar você pelos levels até chegar ao seu good ending na criação de um jogo, e utilizarei como exemplo o processo de desenvolvimento do jogo “Totalitaire”.

LEVEL 1

Missão: Organize sua party!

Quando você decide criar um jogo, o primeiro passo é montar sua equipe. Isso vai facilitar bastante a organização do seu projeto, já que cada membro terá suas responsabilidades específicas. Certifique-se de considerar os diferentes “departamentos” necessários para construir o jogo e distribuir os membros de sua equipe de acordo com a carga de trabalho de cada área. Por exemplo, normalmente temos mais programadores, porque há muitas tarefas relacionadas à programação durante todo o projeto. No entanto, isso não significa que haja poucas tarefas para os designers ou artistas; na verdade, há muito a ser feito, especialmente no início do projeto, quando se discutem aspectos como a interface do jogo, que são os elementos que irão persistir durante todo o desenvolvimento. Portanto, é importante não apenas distribuir bem a equipe, mas também garantir uma distribuição equitativa das tarefas durante as sprints para evitar sobrecarregar os membros.

Missão: Organize sua sprint!

Para garantir a conclusão das tarefas, mesmo diante de um grande volume de trabalho e uma equipe com recursos limitados, é fundamental ter um planejamento de sprints bem estruturado com prazos realistas para cada tarefa. Você já ouviu falar do planejamento de sprint? Reúna todos os membros da equipe e coloque em discussão o seguinte: qual é o resultado que precisamos alcançar e quanto podemos realizar em uma semana? Se você desempenha o papel de Product Owner, Project Manager, Scrum Master ou outra função, é importante analisar e dialogar com a equipe. Antes mesmo de iniciar a produção do jogo, é necessário criar a história, definir o tipo de jogo, os personagens, as mecânicas e outros elementos, então pode ser apropriado reservar uma “sprint 0” para se concentrar nessa etapa; por exemplo, quando o entregável envolve atividades relacionadas ao design do jogo. A questão fundamental aqui é estabelecer um diálogo com a equipe responsável por cada tarefa e, de forma colaborativa, analisar o tempo necessário para que cada membro entregue as atividades requeridas. Isso permite que todos tenham uma visão clara dos passos seguintes e saibam quanto podem avançar no desenvolvimento.

Existem muitas ferramentas que você pode utilizar para administrar as sprints, aposto que algumas você até já conhece! Mas mesmo assim, vou listar algumas que são minhas preferidas: Notion, ClickUp e GitHub Projects.

Missão: Escolha a Engine e as Linguagens!

Neste estágio, é crucial escolher a engine que irá impulsionar seu jogo. A missão aqui é avaliar as opções disponíveis, como Unity, Unreal Engine, Godot, e decidir qual se adequa melhor às necessidades do projeto. Além disso, defina as linguagens de programação que serão usadas para criar o jogo. A escolha certa da engine e linguagens pode afetar significativamente a eficiência do desenvolvimento e a qualidade do jogo, tanto que se for uma linguagem/engine nova para o time, o tempo de adaptação deve ser crucial na organização de sprints!

Missão: Documente, Documente, Documente...

A documentação é uma parte essencial do desenvolvimento de jogos. Sua missão é criar documentos que descrevam detalhadamente todos os aspectos do jogo. Isso inclui documentos de design, documentação técnica, documentação de arte e muito mais. A documentação serve como um guia para a equipe e ajuda a evitar problemas de comunicação e retrabalho. Certifique-se de que todos os membros da equipe estejam cientes da importância da documentação.

Missão: Defina Metas Claras

Para manter o projeto no caminho certo, é fundamental definir metas claras. Determine o que você deseja alcançar em cada fase do desenvolvimento. Estabeleça metas realistas para o lançamento do jogo, como datas de entrega de versões beta ou completas. As metas fornecem direção à equipe e ajudam a medir o progresso. Certifique-se de que todas as metas sejam específicas, mensuráveis, alcançáveis, relevantes e temporais (SMART).

LEVEL UP!

Cada uma dessas missões desempenha um papel vital no desenvolvimento do jogo. A escolha da engine e linguagens afeta diretamente a viabilidade técnica do projeto. A documentação ajuda a garantir que todos na equipe estejam na mesma página e evita problemas de comunicação. E a definição de metas mantém o projeto no caminho certo e permite que todos saibam o que é esperado deles. Portanto, não subestime a importância de cumprir essas missões no Level 1!

LEVEL 2

Missão: Forjando Épicas - Qual é a sua história?

Nada cativa os jogadores como uma narrativa envolvente. Comece criando o esqueleto da história, incluindo o enredo principal, personagens-chave e principais pontos de virada. Considere o mundo do jogo e como ele se relaciona com a história. Desenvolva personagens que tenham objetivos e motivações claras, criando conflitos que impulsionem a trama. Pense na estrutura do roteiro, como atos e arcos de personagens. Lembre-se de que uma história bem estruturada não apenas entretém, mas também guia os jogadores através do universo do jogo, mantendo-os intrigados e ansiosos por mais.

Missão: Hora de craftar! (pixel à pixel)

Os sprites são os elementos visuais que darão vida ao seu jogo. Considerando a ambientação (seja ela medieval, cyberpunk ou outra), é fundamental definir a estética visual. Considere as dimensões dos sprites para garantir que sejam proporcionais e funcionais no contexto do jogo. Isso é especialmente importante se você planeja que os sprites interajam uns com os outros ou com o ambiente. Lembre-se de manter a consistência visual para garantir que todos os elementos do jogo se encaixem harmoniosamente.

Missão: Escolha seus heróis!

O design de personagens vai muito além da aparência física. Desenvolva cada personagem com profundidade, criando perfis que incluam sua personalidade, história de fundo e motivações. Pense em como cada personagem contribui para a narrativa e como eles se relacionam entre si. Considere o papel deles no jogo, se são heróis, vilões ou personagens secundários. Um design de personagens bem-sucedido não apenas torna os personagens visualmente atraentes, mas também os torna cativantes e memoráveis para os jogadores.

Missão: Crie imersão através do som!

O som é uma parte essencial da experiência do jogo. Ao criar uma trilha sonora e efeitos sonoros, leve em consideração a atmosfera que deseja criar. A música e os efeitos devem se adequar ao tom do jogo, contribuindo para a imersão dos jogadores. Além disso, pense na acessibilidade sonora, incluindo legendas e opções de áudio para jogadores com deficiência auditiva. O som é uma poderosa ferramenta para evocar emoções e aprofundar a conexão dos jogadores com o jogo.

Missão: Guie através da Game UI!

A interface do usuário é a ponte entre os jogadores e o mundo do jogo. Projete a Game UI de forma intuitiva e atraente, garantindo que os jogadores possam navegar facilmente e compreender as informações apresentadas. Isso inclui a disposição de menus, ícones, botões e barras de status. Lembre-se de que a Game UI deve ser informativa, mas também discreta o suficiente para não distrair os jogadores da ação principal. Uma interface bem projetada melhora a experiência do jogador, tornando o jogo mais acessível e agradável.

Missão: Tijolo a Tijolo, do Tile ao Tileset!

Os cenários são o cenário onde a ação se desenrola. Ao criar tiles e tilesets, pense nos detalhes que tornarão o mundo do jogo vívido e interativo. Considere as texturas, as cores e a coesão visual entre os diferentes elementos do cenário. Certifique-se de que os tiles se encaixem de forma agradável, permitindo a criação de ambientes variados e convincentes. A criação de cenários não é apenas sobre estética, mas também sobre como eles afetam a jogabilidade. Os cenários devem ser projetados de forma a apoiar a narrativa e proporcionar desafios interessantes para os jogadores.

LEVEL UP!

Cada uma dessas missões é fundamental para a construção de um jogo bem-sucedido. Ao explorar o reino dos jogos, você estará moldando o universo que os jogadores explorarão, tornando-o cativante e envolvente. Dedique tempo e criatividade a esses aspectos, e você estará no caminho certo para criar uma experiência de jogo inesquecível.

LEVEL 3

Missão: Seja um contador de histórias!

Para criar uma história de jogo cativante, mergulhe nas profundezas do conhecimento. Estude aspectos políticos, atualidades, referências históricas e científicas que podem enriquecer o arcabouço teórico da sua narrativa. Lembre-se de como a série “The Last of Us” da HBO Max usou uma cena inicial que discutia a probabilidade dos fungos consumirem a mente humana para criar uma imersão arrepiante. Compreender o contexto científico por trás disso é fundamental. Da mesma forma, considere cenas icônicas como a de “Lucy” e “e se nós usássemos 100% do nosso cérebro?” que deixam as pessoas intrigadas até hoje. Criar narrativas convincentes é essencial para a imersão e para se destacar no mercado de jogos.

Missão: O mundo nas suas mãos através do Level Design

O design de níveis é a espinha dorsal da jogabilidade. Sua missão aqui é criar ambientes cativantes e desafiadores. Planeje a progressão do jogo, estabeleça os objetivos dos níveis e pense nos elementos que manterão os jogadores envolvidos. Considere a conexão entre os níveis e como eles se encaixam na narrativa global. Lembre-se de que o level design é onde os jogadores passarão a maior parte do tempo, então torne-o memorável e envolvente.

Missão: Seja um Mestre, pesquise e reflita!

A temática do jogo deve refletir o interesse e o conhecimento da equipe. Explore outras realidades, compreenda experiências e dores para criar personagens mais autênticos e conexões profundas com os jogadores. Pergunte-se: o que apaixona a equipe? Qual é a mensagem que vocês desejam transmitir? Ao incorporar temas que vocês entendem e abraçam, o jogo se tornará mais autêntico e impactante.

Missão: Mantenha as engrenagens girando

A jogabilidade é o coração do seu jogo. Sua missão é definir as mecânicas que tornam a experiência interativa e divertida. Pense no ciclo de jogo: o que os jogadores farão repetidamente e como isso os mantém engajados? Considere os desafios que os jogadores enfrentarão e como as mecânicas afetam a narrativa e o progresso. Equilibre o desafio e a recompensa para criar uma experiência envolvente.

Missão: Aprenda com os melhores!

Para entender a construção de um jogo de sucesso, desmonte-o peça por peça. Analise a mecânica central do jogo, o ciclo de gameplay e o desafio que os jogadores enfrentam. Pergunte-se por que esse jogo é tão atraente e como ele mantém os jogadores voltando. Ao desconstruir um jogo de referência, você aprenderá lições valiosas que podem ser aplicadas ao seu próprio projeto.

Neste nível, você estará se aprofundando na criação da experiência de jogo, explorando a narrativa, o design de níveis, a temática, as mecânicas e a análise de jogos de referência. Cada missão é crucial para criar um jogo que seja envolvente, autêntico e memorável, deixando uma marca duradoura no mundo dos jogos.

LEVEL UP!

Agora que você absorveu os principais fundamentos para o desenvolvimento do seu jogo, está na hora de colocar a teoria em prática. Mãos à obra! E para começar, que tal um tutorial simples sobre como criar um personagem em pixel art usando o Aseprite, uma ferramenta excelente para essa tarefa?

LEVEL 4

1. Instale o software: Se você ainda não possui o Aseprite instalado, faça o download e a instalação da ferramenta em seu computador. Não se preocupe, o programa é Open Source e você pode encontrar facilmente o tutorial de instalação no repositório do GitHub, mas recomendo fortemente que dêem apoio aos devs, sabemos bem como é, né?

2. Crie um Novo Projeto: Abra o Aseprite e crie um novo projeto definindo o tamanho da tela (por exemplo, 32x32 pixels) e a paleta de cores que você deseja utilizar, baseado em todo o estudo que você já tem sobre o seu personagem.

3. Crie um Esboço: Comece com um esboço do seu personagem. Use a ferramenta lápis para desenhar os contornos e as formas básicas do personagem, no entanto, para ajudar à criar formas mais “certinhas” você pode utilizar a ferramenta de pixel-perfect, deixando seus traços mais simétricos.

4. Preenchimento de Cores: Com o esboço em vigor, preencha as áreas do personagem com as cores apropriadas. Lembre-se de que, em pixel art, a precisão e o uso inteligente das cores são essenciais, até mesmo porque os sprites são bem pequenos, então um pontinho aqui e outro ali pode fazer uma diferença considerável!

5. Detalhes e Sombreamento: Adicione detalhes ao seu personagem, como olhos, roupas e acessórios. Use tons mais escuros para criar sombras e dar profundidade ao personagem.

6. Animação Frame a Frame:

a. Frame Inicial (Idle): Crie o primeiro quadro da animação, que será o estado “idle” do seu personagem. Mantenha-o em uma posição neutra, como se o personagem estivesse parado. Certifique-se de que todas as características estejam alinhadas e coerentes com o design.

b. Frame Seguinte: Crie um novo quadro para a animação, levemente deslocado do quadro inicial. No caso da Bellatrix, a protagonista do jogo Totalitaire, deslocamos um pouco a posição dos braços e modificamos as feições dela. Use o “onion skinning” do Aseprite para ver o quadro anterior como referência.

c. Repita o Processo: Continue criando quadros subsequentes, fazendo pequenos ajustes a cada novo quadro para simular uma animação suave. Pode ser uma leve oscilação ou uma animação de respiração, dependendo do personagem.

7. Exportação: Quando estiver satisfeito com a animação, exporte-a em formato GIF. Você pode fazer isso selecionando “File” > “Save for Web” e escolhendo o formato GIF. Isso permitirá que você crie uma animação em loop para o personagem.

8. Crie um Repositório no GitHub: Para manter seu trabalho documentado e compartilhá-lo com sua equipe, crie um repositório no GitHub. Faça o upload dos arquivos, incluindo a animação GIF e quaisquer outros recursos relacionados ao seu jogo. Isso facilitará a colaboração e o controle de versões.

Lembre-se de que a prática é fundamental para aprimorar suas habilidades em pixel art e animação. Continue experimentando e refinando seu personagem e suas animações até alcançar o resultado desejado. Divirta-se e aproveite o processo criativo!

LEVEL BÔNUS

Desenvolver jogos é uma jornada desafiadora, e o sucesso de um time de desenvolvimento de jogos depende principalmente de uma gestão eficiente. Reconheça e enfrente os obstáculos com determinação e saiba como engajar os membros da equipe para manter o ritmo e a motivação elevados. Lembre-se de que o processo de criação de jogos pode ser trabalhoso, mas a recompensa de ver seu projeto ganhar vida é incrível.

Além disso, considere a monetização como uma etapa importante. Existem diversas maneiras de rentabilizar seu trabalho, desde a venda de sprites e outros ativos até o lançamento e comercialização do próprio jogo. Explore oportunidades de negócios adicionais, como a criação de um site dedicado ao seu jogo, para expandir sua presença no mercado e atrair mais jogadores.

Não subestime a importância de avaliar os tipos de investimento necessários para o seu projeto. Isso inclui custos de exportação do jogo e outros recursos. Avalie os riscos e viabilidades para tomar decisões informadas que ajudarão a manter seu projeto dentro do orçamento e prazo.

Lembre-se também de que não há problema em deixar algumas tarefas em standby, desde que a equipe esteja entregando no geral. Ao reconhecer os avanços da equipe e manter uma comunicação transparente, você promove um ambiente de trabalho saudável e produtivo, onde todos podem contribuir para o sucesso do projeto.

E aí? Se sente pronto para embarcar nessa emocionante aventura de desenvolvimento de jogos? Este guia foi apenas o começo, e o mundo dos jogos oferece infinitas possibilidades para sua criatividade florescer. Lembre-se de que o desenvolvimento de jogos é uma jornada repleta de desafios e recompensas. Com os fundamentos, dicas e ferramentas à sua disposição, você está equipado para criar experiências de jogo incríveis. Agarre a oportunidade, mergulhe de cabeça e comece a construir seu próprio mundo interativo. Agora, a próxima fase está em suas mãos! Hasta la vista!

GESTÃO PROCESSO

DA VINCE E MICHELANGELO FALTARAM A SPRINT PLANNING: PORQUE O RENASCIMENTO PRECEDEU A TECNOLOGIA CONTEMPORÂNEA

TEXTO
YARA SAMPAIO

ILUSTRAÇÃO
SARAH MADALENA

O Renascimento foi um movimento de impacto cultural marcado pelo avanço científico, o florescimentos das artes e a exploração do potencial humano que influenciou o desenvolvimento da sociedade europeia e além. O legado do Renascimento é evidente no cotidiano da sociedade contemporânea desde a apreciação pela arte até a ênfase na busca do conhecimento e da inovação.

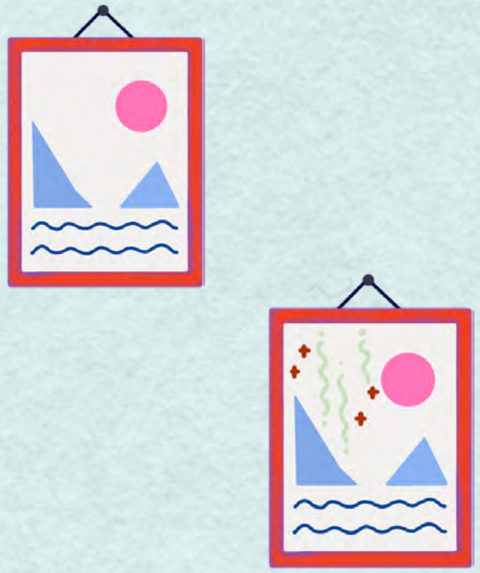
Galeria de arte tech

Paralelo a isso, podemos fazer uma alusão ao avanço da tecnologia, que influenciou e ainda influencia a sociedade a nível global. Ela revolucionou a forma como nos comunicamos, o acesso a informações, os avanços na medicina, sem falar, é claro, da maior façanha dos últimos tempos: a inteligência artificial. Quem acusa dizer que esse feito não estaria nas galerias de artes renascentistas? Leonardo da Vinci, polímata que era, jamais perderia esse feito da engenharia.

O que herdamos dos profissionais renascentistas?

Além dessa abertura para inovação, o renascimento foi marcado também pelo surgimento de profissionais generalista, como Leonardo da Vinci e Michelangelo. Da Vinci, por exemplo, tinha abordagem multifacetada para o conhecimento e influenciou o desenvolvimento de diversas áreas, incluindo a tecnologia. O idealizador de Monalisa não se limitou a ser apenas um artista excepcional, mas também um engenheiro, inventor, anatomista e cientista. Suas habilidades generalistas permitiram que ele explorasse diversas disciplinas e abordasse os problemas de maneira holística, muitas vezes encontrando soluções inovadoras que transcendiam as barreiras convencionais do seu tempo.

Esse contexto nos faz pensar sobre a importância e a necessidade de profissionais multidisciplinares na área de tecnologia. Os desafios tecnológicos atuais são complexos e interdisciplinares, exigindo uma visão ampla e a capacidade de conectar conhecimentos de diferentes campos. Profissionais que possuem habilidades generalistas podem abordar problemas de maneira mais abrangente, identificar soluções criativas e inovadoras, desempenhando um papel vital na resolução dos desafios tecnológicos do século XXI.



Programando novas Capelas Sistinas

Além disso, a sinergia da squad também é de grande relevância, isso porque, todos trabalham com um objetivo comum. Cabeças pensantes de contextos diversos criam produtos esplêndidos, além de se ajudarem durante o processo. Que tal um exemplo exemplo prático? Imagine um cenário em que você está em um time de engenharia e não sabe se utiliza MVVM, MVC ou outra arquitetura para aquele produto. Em um papo com designers, enquanto eles discutem sobre a importância da hierarquia de informações assertiva, você têm um insight de arquitetura perfeita para utilizar no projeto. Duvido que você perderá o café para discutir a fonte do protótipo quando tiver com aquele bug. Um produto tecnológico que provém de um time cooperativo pode ser um obra de arte tão grandiosa quanto à Capela Sistina, obra renascentista famosa que foi projetada por diversos artistas renomados, incluindo Michelangelo, Sandro Botticelli, Pietro Perugino, Pinturicchio, Domenico Ghirlandaio e outros.

CÓDIGO TUTORIAL

DANÇANDO COM SWIFT UI: CONHECENDO ANIMAÇÕES

TEXTO
THAYNARA ANDRADE

ILUSTRAÇÃO
LAIS BARBOSA

REVISÃO
GABRIEL SANTIAGO

As animações desempenham um papel vital na criação de interfaces de usuário mais atraentes. Elas podem ser usadas para transmitir informações, fornecer feedback e criar transições suaves entre telas, assim tornando seu aplicativo mais atraente. Com o SwiftUI, o processo de criação de animações se tornou mais acessível, por possuir uma sintaxe declarativa e orientada a dados. Agora vamos embarcar nessa jornada para que possamos começar a incorporar animações em SwiftUI.

Conceitos Básicos das Animações no SwiftUI

Antes de começarmos a dançar com as animações no SwiftUI, é importante entender alguns conceitos fundamentais:

@STATE E @BINDING

@State é uma propriedade observável que permite que as alterações em seu valor atualizem a exibição automaticamente. Isso é fundamental para criar animações que respondem a alterações de estado. @Binding é uma extensão do @State que permite a passagem do estado entre visualizações.

WITHANIMATION

O withAnimation é um modificador que envolve um bloco de código que realiza mudanças de estado com animações. Isso permite que você controle como as alterações de estado são animadas, definindo a duração e o tipo de animação.

UIVIEWREPRESENTABLE

Para animações mais complexas que não podem ser alcançadas com as ferramentas internas do SwiftUI, você pode usar o UIViewRepresentable para incorporar componentes UIKit personalizados com animações nativas.

HORA DA PRÁTICA

Vamos explorar alguns exemplos de animações comuns que podem ser facilmente implementadas em SwiftUI.

Animação de Transições

Neste exemplo, usamos o @State para rastrear o estado isShowing, que controla a visibilidade do texto. Quando o botão “Toggle” é pressionado, a função withAnimation é usada para suavizar a transição de isShowing. O texto é animado com .transition(.opacity), fazendo-o aparecer e desaparecer com um efeito de fade (opacidade).

```swift
import SwiftUI

struct FadeAnimationExample: View {
    @State private var isShowing = false

    var body: some View {
        VStack {
            Button(“Toggle”) {
                withAnimation {
                    isShowing.toggle()
                }
            }
            if isShowing {
                Text(“Elemento visível”)
                    .transition(.opacity)
            }
        }
    }
}
```

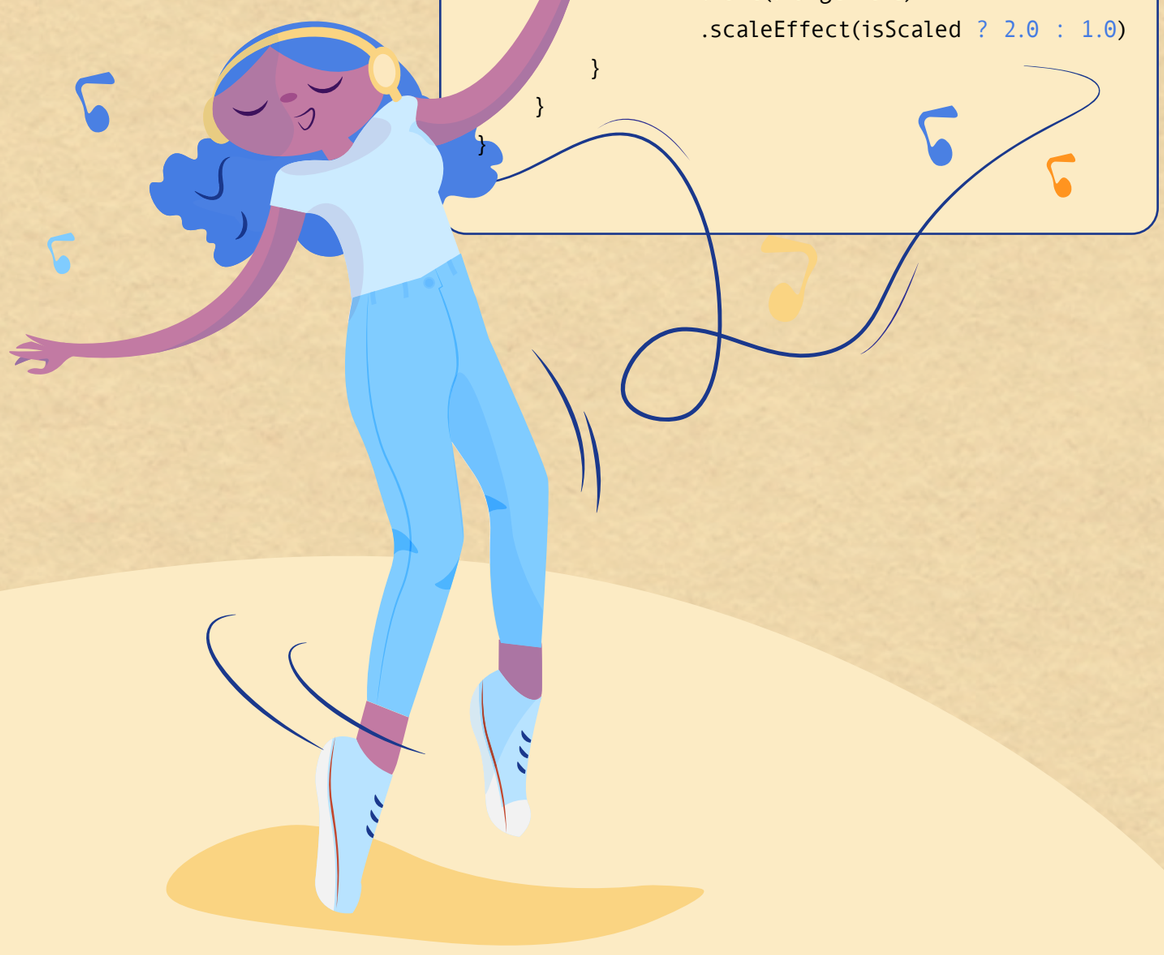
Animação de Escalonamento

Aqui, o @State é usado para rastrear o estado de isScaled, que determina o dimensionamento da imagem. Quando o botão “Toggle” é pressionado, a função withAnimation torna a transição suave. A imagem (o ícone de estrela) é animada com scaleEffect, aumentando seu tamanho para 2x quando o booleano isScaled for true.

```swift
import SwiftUI

struct ScaleAnimationExample: View {
    @State private var isScaled = false

    var body: some View {
        VStack {
            Button(“Toggle”) {
                withAnimation {
                    isScaled.toggle()
                }
            }
            Image(systemName: “star”)
                .font(.largeTitle)
                .scaleEffect(isScaled ? 2.0 : 1.0)
        }
    }
}
```



Animação de Transição

Neste exemplo, o estado de `**isMoving**` é rastreado com `**@State**` para controlar o movimento do círculo. A função `**withAnimation**` é utilizada para criar uma animação suave durante a transição. A posição do círculo é alterada com o modificador `**offset**`, movendo-o horizontalmente em 100 pontos quando `**isMoving**` for `**true**`.

```
swift
import SwiftUI

struct MoveAnimationExample: View {
    @State private var isMoving = false

    var body: some View {
        VStack {
            Button(“Toggle”) {
                withAnimation {
                    isMoving.toggle()
                }
            }
            Circle()
                .frame(width: 50, height: 50)
              .offset(x: isMoving ? 100 : 0, y: 0)
        }
    }
}
```

Embora essas animações tenham sido abordadas nesse artigo de forma isolada, é comum combinar vários tipos de animações para criar experiências de usuário mais complexas e dinâmicas. Além disso, a documentação oficial da Apple sobre SwiftUI é uma fonte valiosa de informações e exemplos detalhados para ajudá-lo a aprimorar suas habilidades de animação.

À medida que você explora, experimenta e implementa mais animações, você descobre que elas podem elevar significativamente a qualidade e a usabilidade do seu aplicativo. Portanto, não hesite em mergulhar mais fundo no mundo das animações em SwiftUI e continuar aprimorando suas habilidades. Seja criativo e inovador em sua abordagem, e suas animações podem se tornar um diferencial significativo em seu projeto.