BRAZIL

Currículo Technovation
Semana 3

Unidade de marketing 1: Como construir uma marca

Unidade de programação 3: Condicionais, lógica e loops

Continuar Unidade de negócios 2

Marketing (M) Módulo 1. Construir uma marca

Objetivos de aprendizagem:

Neste módulo, você aprenderá a...

- Identificar a voz, os valores e a visão da sua marca
- Integrar sua marca a seus negócios

O que é uma marca? Aprender: Como identificar sua marca

A American Marketing Association, ou, a Associação Americana de Marketing define **marketing** como: "a atividade, instituições e processos de criação, comunicação, entrega e troca de ofertas que agregam valor a clientes, parceiros e à sociedade como um todo." Em outras palavras, é o trabalho feito para mostrar às pessoas que seu negócio e seu produto (o aplicativo) são úteis e atraentes. **Marketing** é a forma como você se comunica com pessoas (clientes, investidores, outros negócios) para falar sobre sua empresa: qual é sua missão, o que ela vende e o que esperar ao trabalhar com você. Outra maneira de pensar sobre as partes importantes do marketing são os quatro Ps: produto, preço, promoção e praça.

"As pessoas têm almas. Os produtos têm marcas." - Jennifer Kinon, Designer e Cofundadora da OCD

Sua **marca** é o que as pessoas pensam sobre você. Se seu aplicativo fosse uma pessoa, sua marca

seria a personalidade dela. É tudo que o público pensa que sabe sobre seu nome, marca e produto: factualmente (p.ex. ele vem em uma caixa roxa e alaranjada) e emocionalmente (p.ex. é romântico). Sua marca existe objetivamente, pois ela é realmente vista pelas pessoas. Ela não muda. Mas sua marca só existe na mente das pessoas.[1] Ela é "quem" o produto/empresa é.

Pense em marcas conhecidas. Você consegue imaginar seus logos? O que elas vendem? Qual seu slogan? Qual sensação você tem ao pensar sobre essas empresas?

Você as conhece bem porque essas empresas possuem um branding forte, memorável e que se destaca na multidão. Por exemplo, o que você pensa quando vê um "swoosh", (ou um ✓ arredondado) em tênis, camisetas ou bonés? A logo do "swoosh" é conhecida mundialmente como o símbolo da Nike, uma das marcas mais conhecidas do mundo. Ela representa velocidade, movimento, excelência atlética e desempenho. Fazendo com que atletas famosos e bem-sucedidos usem seus produtos com a logo, a Nike desenvolveu uma atitude que diz: se você quiser ser atlético, a Nike ajudará você a chegar lá. E seu famoso slogan "Just do it" reforça essa ideia.

NIKE

JUST DO IT.

Assista a [este vídeo sobre branding] e tente reconhecer as marcas!

[1] http://www.forbes.com/sites/jerrymclaughlin/2011/12/21/what-is-a-brand-anyway/#7bcd32202aa4

Branding faz muitas coisas:

- Torna o produto fácil de ser reconhecido por meio de ferramentas visuais (logo, cor tema)
- Dá uma voz e personalidade ao seu produto
- Constrói relacionamentos e memórias com clientes

Identificar sua marca

Para começar com sua marca, consulte seu relatório de pesquisa de consumo feito na unidade de negócios 2. É preciso conhecer muito bem seu público-alvo. São as pessoas para quem você vai vender, seus **clientes**. Isso ajuda você a se focar em uma marca que realmente os atraia e os faça querer comprar seu produto. Tente responder a estas perguntas para começar a ter uma ideia sobre a sua marca:[2]

1. Benefícios para o cliente: Qual problema dos clientes você está tentando resolver? Pense na razão pela qual alguém compraria isso.
2. Clientes-alvos: Quem são eles? Seja o mais específica possível, tente dividir em duas partes:
    - **Demografia** é um conjunto de características de uma determinada população. Isso dirá a você *quem* compra seu produto.
        - Por exemplo: idade, gênero, local, raça, etnia, idioma, educação, religião, renda, etc.
    - **Psicografia** é o estudo de personalidades, valores, opiniões, atitudes, interesses e estilos de vida. Isso mostra a *razão* pela qual alguém compra seu produto.
        - Por exemplo: pessoas focadas na saúde, ocupadas, altamente organizadas, focadas na família, em preservar o ambiente, sociáveis, caseiras, nerds, etc.
3. Personalidade da marca: Se sua marca fosse uma pessoa, quem ela seria? Busque correlacionar sua marca com alguém conhecido. Pode parecer bobo, mas isso ajuda a definir sobre o que é o seu negócio.
    - Por exemplo, Jane Goodall ama animais, ou Michael Jackson é um músico e artista.

Estudo de caso de identidade da marca

Pratique como identificar marcas com a propaganda usada para atualizar a marca de computadores Apple. Assista à propaganda e responda às perguntas acima. Se você precisar de ajuda para responder a essas perguntas, veja o vídeo que tem 15 minutos.

https://www.youtube.com/watch?v=9GMQhOm-Dqo
https://www.youtube.com/watch?v=nmwXdGm89Tk

Em seu discurso, ele buscou responder perguntas dos clientes, como "Quem é a Apple? O que defendemos? Onde nos encaixamos no mundo?" Ele conseguiu passar os valores principais da Apple de forma simples e nem precisou do produto. Com sua campanha "Think Different", ou pense diferente,

[2] Inspirado por Lauren Zirilli, Diretora global de estratégia de Branding da +Vistaprint - 8:05-12:02

ele conseguiu dizer que, se o cliente quer pensar diferente e mudar o mundo de alguma forma, ele precisa de um computador Apple. Essa mensagem chega ao coração do cliente: você é aquilo que você usa. Essa foi uma das campanhas de marketing que mais obtiveram sucesso na história.

Para mais informações sobre branding, veja os seguintes recursos:

- Entrepreneur: Informações básicas sobre branding
- Envato: O que é branding?
- Slideshare: Slides da marca GoPro
- Hubspot: Como criar buyer personas para seu negócio

Atividade nós somos e não somos[3]

[Observação: esta atividade pode ser facilitada por um mentor ou um educador. Vocês também podem colaborar usando um documento online compartilhado, se estiver longe Planilha]

Esta atividade ajudará a descobrir as características principais que descrevem sua marca. Reúna a equipe, as pessoas que melhor conhecem a marca. Escreva adjetivos em post-its ou pedaços de papel que descrevam sua marca e a maneira como gostariam que ela fosse vista pelos clientes. Divida esses papéis em 3 pilhas: "Somos", "Não somos" e "Não se aplica". Ao final, devem haver poucos adjetivos na pilha "Somos".

Prontas para começar?

Vocês vão precisar de:

- Cartões ou papéis pequenos (se estiverem trabalhando online, usem um software colaborativo, como Google Docs ou IdeaBoardz)
- Canetas ou marcadores
- Planilha

Descrição da atividade:

1. Entregue de 10 a 15 cartões para cada membro.
2. Peça que escrevam uma palavra ou frase curta para descrever sua empresa em cada cartão.
3. Em equipe, organizem os cartões nas três categorias: "Somos", "Não somos" e "Não se aplica". Lembre-se de que as palavras devem tentar descrever como vocês querem ser vistas pelos clientes.
4. Reduza as categorias "Somos" e "Não somos" até que tenham 4 a 7 cartões.
5. Terminem os cartões. Certifiquem-se de que todos concordam com a organização. Se não, discutam e concluam em grupo.

O que acontece agora:

Agora vocês têm as palavras mais importantes para sua marca. É hora de aplicar ao negócio! Usem essas palavras para guiar a construção da marca. Esses adjetivos podem ser transmitidos através de muitas coisas durante esta unidade, especialmente pelo nome do negócio e o nome do produto (o

[3] Atividade inspirada em http://branding.cards/

aplicativo).

Eles também serão úteis na hora de fazer a parte visual! Se alguns adjetivos forem "jovem" e "divertido", é possível usar cores mais brilhantes, enquanto para "sério" e "profissional", cores mais simples e sofisticadas podem ser usadas. Ao construir a marca durante este módulo e com a concorrência, pergunte-se se estão consistentes com os adjetivos escolhidos agora.

Como definir o nome do seu negócio

Assim que tiver uma ideia de marca e produto, é hora de nome para sua empresa e seu aplicativo! O nome deve transmitir claramente a sua a marca e o negócio. Lembre-se de que o produto e a empresa podem ter o mesmo nome. Você escolhe.

Quando for enfrentar esse desafio, pense no que faz um nome ser bom e o que faz ele ser ruim. Nomes não precisam dizer exatamente o que a empresa faz, mas podem ficar memoráveis com o tempo, conforme a empresa e sua marca ficarem mais conhecidas. No começo, palavras como Apple, Google e Coca-Cola eram apenas palavras ou frases inventadas. Com o tempo e com a interação das pessoas, elas ficaram cada vez mais conhecidas. Podem ser até acrônimos, sobrenomes ou combinações de palavras. Alguns exemplos seriam BMW, MTN, AT&T e Samsung. Sabia que os fundadores do Google pensaram em chamá-lo de Backrub, que em inglês significa "massagem nas costas"?

Caso não saiba por onde começar, este artigo é um bom lugar!

Depois, faça um brainstorm de nomes possíveis com a equipe e reduza a lista para apenas alguns. Use sua criatividade ou um gerador de nomes (ou experimente este). Geradores de nome podem ajudar unindo as palavras que você quiser e dando mais opções ainda. Tenha em mente que é aconselhável você retomar os adjetivos decididos na atividade "Nós somos e nós não somos". Isso ajudará a guiar conversas e a determinar que nome seria melhor para sua empresa.

Lembre-se de que o nome da sua empresa *não precisa* ser o mesmo que seu aplicativo, mas pode ser, se você preferir. Você escolhe o nome da empresa e do aplicativo!

Após reduzir sua lista de nomes para 5 ou menos, pergunte a possíveis clientes, amigos ou alguém que possa ver a empresa e o aplicativo. Isso pode ajudar você a ver a reação das pessoas ao nome, então você saberá qual nome é mais atraente e faz mas sentido para as pessoas. É possível até perguntar às

pessoas quais adjetivos melhor descrevem o nome da empresa e ver se correspondem com os que vocês pensaram.

Assim que reduzir ainda mais, escolham um! Você e sua equipe precisam concordar sobre o nome final.

Para mais dicas para o processo de definir o nome da empresa ou aplicativo, veja o link abaixo:

- Startupbros: Como escolher o nome perfeito para sua empresa ou start-up
- Entrepreneur: Como definir o nome do seu negócio
- Adweek: Nomes simples são melhores

Criar: Uma declaração de posicionamento

Agora que vocês têm uma noção melhor de qual é a marca de vocês, podem começar a trabalhar na sua declaração de posicionamento. Uma **declaração de posicionamento** é uma descrição concisa do seu cliente e uma imagem cativante sobre como vocês querem que seu cliente veja vocês como marca. Quando bem feita, essa é uma das partes mais importantes do seu plano de marketing: ela dita como o branding, logos e até o atendimento serão feitos. Isso não deve ser confundido com a missão (criada no Módulo 1), que é mais ampla e descreve o que a organização busca resolver ou alcançar. Você pode ver mais detalhes sobre declarações de posicionamento aqui.

[Observação: Os pontos principais foram retirados dos seguintes guias gerais para vocês começar.]

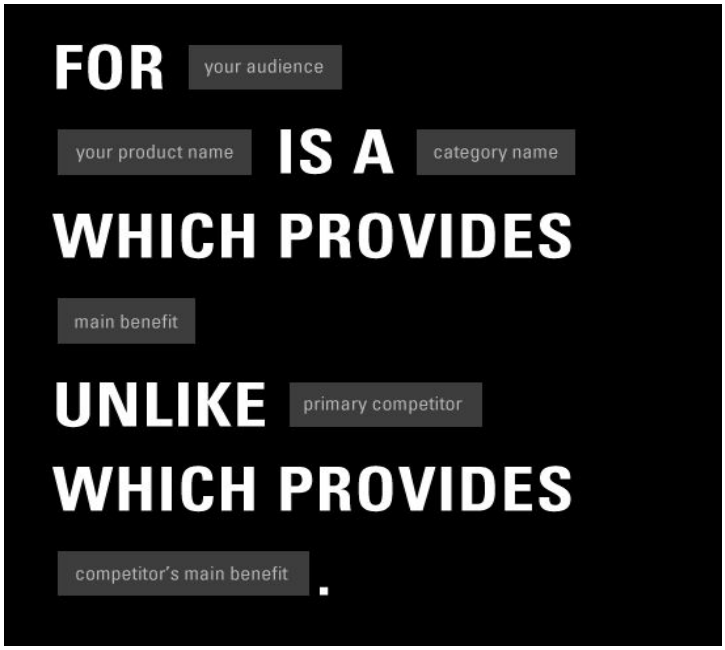
Sua declaração de posicionamento deve cobrir 4 partes principais:[4]

1. Seu produto (e por que ele é especial)
    a. Descrição clara do valor que vocês (ou seu aplicativo) oferecem. Se forem arquitetas, vocês não fazem plantas, vocês projetam edifícios. Se forem um restaurante chique, você não faz comida, vocês oferecem experiências culinárias. Certifique-se de que entendem o que estão oferecendo quando alguém compra seu produto ou serviço.
2. Seu público-alvo (e por que vocês os amam)
    a. A quem vocês são voltadas? Como eles são? Ela deve ser concisa e direta, mas também mostrar aonde querem chegar. Não é suficiente dizer que seu público-alvo são "adolescentes", mas seria extenso demais dizer "Nossa missão é oferecer a homens de 13 a 17 anos de idade que gostam de jogos de tabuleiro semelhantes a Dungeons and Dragons mas NÃO de jogos como Banco Imobiliário…". Uma frase curta e direta, como "Adolescentes que gostam de jogos de tabuleiro" já é o suficiente.
3. Seu cenário competitivo (e por que vocês são melhores)
    a. A declaração de posicionamento da sua marca também deve descrever o valor do seu *produto*. O seu produto realmente oferece uma qualidade maior ou equilibra preço e

[4] Informações retiradas de: https://milesherndon.com/blog/what-is-brand-positioning-statement

qualidade? Sua empresa possui seguidores ou é uma nova estrela? Certifique-se de que entende exatamente onde sua marca está no cenário competitivo. Analise vantagens e desvantagens de comprar sua marca.

4. A promessa da sua marca (e por que devemos acreditar em vocês)
    a. Que promessa sua marca faz para o mercado? Promete confiabilidade? Rapidez? Custo baixo? Certifique-se de que você entende sua promessa e como vai dar conta dela. Como pode prová-la?



[Declaração de posicionamento clássica, popularizada pela Procter & Gamble (Chanpory Rith)]

Bons exemplos de declarações de posicionamento de marca

Como é um bom posicionamento de marca? Eles vêm de várias formas e tamanhos, mas devem sempre englobar os elementos acima. Veja os seguintes exemplos de marcas conhecidas. Como pode ver, alguns elementos estão subentendidos ou implícitos, mas é importante notar como cada um deles é bem pensado.

- Volvo: Volvo é o carro familiar que oferece máxima segurança para as famílias americanas de alto nível.
- Home Depot: A loja de departamento para quem gosta de fazer as coisas por si mesmo.
- Zipcar: Para clientes urbanos, educados e que gostam de tecnologia, o serviço de compartilhamento de carros Zipcar pode ser usado em vez de comprar um carro, para você poupar dinheiro e reduzir a pegada de carbono.

Modelo para escrever a declaração de posicionamento

Agora chegou a hora de escrever sua declaração de posicionamento! Este é um modelo básico (e

outro) para escrever uma declaração de posicionamento:

Para [cliente-alvo], a [marca] é [diferencial] entre [quadro de referência] porque [razão para acreditar].

- O **ponto de diferenciação** (PDD) descreve como sua marca ou produto beneficia clientes de formas que diferenciam vocês de seus concorrentes.
- O **quadro de referência** (QDR) é o segmento ou categoria na qual sua empresa compete.
- A **razão para acreditar** é exatamente isso. Essa declaração fornece evidências e razões convincentes sobre por que os clientes do seu mercado podem confiar nos seus diferenciais.

A maneira como vocês escreverem a declaração de posicionamento não precisa ser exatamente a mesma desse modelo, mas, para ser eficaz, ela deve conter os cinco componentes dentro dos colchetes. Se estiver presa ou só precisar de ajuda, veja este gerador para ajudá-la a escrever. Depois de escrever, vocês podem pedir feedback para seu mentor, ou fazer esta atividade individualmente e dar feedback umas para as outras como uma equipe.

Atividade para o mentor: Revisar a declaração de posicionamento

As alunas entregarão uma declaração de posicionamento, que é uma declaração curta e concisa que aborda sobre o que é a empresa: seus valores, voz e visão. Agora você deve avaliá-la. Certifique-se de dar feedbacks construtivos para o plano, além de fazer perguntas.

Diretrizes para boas declarações de posicionamento

Aqui estão seis itens para ter em mente quando estiver dando feedback para sua equipe:[5]

1. É simples, memorável e feito especialmente para o mercado-alvo.
2. Fornece um retrato claro e inconfundível da marca, que a diferencia dos concorrentes.
3. É digno de confiança e a marca pode cumprir sua promessa.
4. A marca pode ser a única nesta posição particular do mercado. A equipe pode "bancar" a declaração.
5. Ela ajuda a avaliar se as decisões de marketing são consistentes e apóiam a marca.
6. Ela dá espaço para crescimento.

Checkpoint para desenvolvimento de equipe:

Como mentor, essa é uma oportunidade para verificar se as garotas trabalharam como uma equipe para o projeto além de garantir que preocupações e feedbacks sejam abordados. Essa é uma atividade que pode ajudar a construir o espírito de equipe, já que não é complicada técnica ou financeiramente.

[5] Retirado de http://blog.ecornell.com/how-to-write-market-positioning-statements/

Crie um fórum aberto para ver qual foi o papel de cada uma das participantes durante este módulo, como elas tomaram decisões (diplomaticamente, autoritariamente, etc.) e como elas resolveram conflitos que surgiram.

Aplicar: Sua marca ao seu negócio

Vocês têm a marca, agora é hora de aplicá-la ao seu negócio. Isso significa que vocês colocarão a "sensação" da sua marca em todo lugar! Pense em como vocês descreveram seu aplicativo, interações com os clientes e o desenvolvimento do aplicativo. Tudo precisa ser muito bem pensado e, mais importante, consistente. Para garantir que vocês estão prontas, aqui está um quiz de identidade da marca que você e sua equipe podem fazer.

Se for um negócio voltado para crianças, vocês devem ter uma linguagem simples, fácil de entender e com muitas imagens na comunicação. Pode ficar confuso e estranho se houver uma linguagem formal e poucas cores e imagens. Se for um negócio voltado para os amantes de animais, pode haver fotos de animais e fatos engraçados nas descrições. É importante reforçar continuamente os principais conceitos da marca em tudo que for feito para os clientes (desde correspondência até mídia social), para que eles reconheçam e associem isso à sua empresa. Saiba mais sobre padrões repetitivos para reforçar a marca aqui. Lembre-se de que o planejamento de negócios, apresentação, pitch e vídeos de demonstração podem ser mais técnicos, porque serão usados em contextos mais formais.

Dica: Volte ao módulo de pesquisa de mercado e faça entrevistas de solução para testar seus materiais de branding.

Refletir:

Até agora, você identificou a voz, valores e visão do negócio. A partir disso, você criou uma declaração de posicionamento de marca e identificou maneiras de integrar sua marca ao negócio.

Aprendemos MUITO nesta unidade! Vamos recapitular o que aprendemos:

- **Marketing -** É tentar convencer as pessoas a comprarem o que vocês estão vendendo
- **Declaração de posicionamento** - É uma descrição curta do seu cliente que define como querem que a empresa seja vista pelo cliente
    - **Diferencial** - Por que vocês são diferentes de seus concorrentes
    - **Quadro de referência** - Onde seu negócio se encaixa no mercado
- **Clientes -** A pessoa que vocês querem que compre seu produto
- **Demografia** - Características de um grupo de pessoas, neste caso, seus clientes. Isso dá as informações de base sobre quem compra o produto, incluindo idade, gênero, local etc.
- **Psicografia** - O tipo de personalidade que seus clientes podem ter, o que oferecer informações sobre a razão pela qual alguém compraria seu produto. Essas informações podem incluir dados como: se seu cliente gosta de sair de casa, é ocupado, é focado na família etc.

Na próxima unidade, você desenvolverá o visual da sua empresa: isso significa desenvolver sua logo e as cores da marca. Vocês podem começar fazendo um brainstorm para definir quais tipos de componentes visuais vão precisar, como panfletos, boletins informativos, apresentações, cartões de visitas, embalagens etc.

Recursos adicionais:

- Fastco Design: Branding são padrões, não mensagens repetitivas
- Harvard Business Review: Branding na era da mídia social
- TED Talk: Dan Cobley - O que a física me ensinou sobre marketing
- Entrepreneur: Informações básicas sobre branding
  http://www.slideshare.net/nicolasleonard986/gopro-brand-audit?qid=c90bb4e1-bb68-4879-ae25-3f1a31d52d70&v=default&b=&from_search=3
- OwnerNation: Faça o branding do seu negócio
- Google for Small Business: Branding

Unidade de programação 3: Condicionais, lógica e loops

Parabéns pelo seu trabalho até agora! Esta unidade é grande, mas contém muitos conceitos essenciais de programação para seu aplicativo. Ela contém o último desafio de programação e também é a última para ensinar programação básica. Após esta unidade, você aprenderá sobre recursos adicionais e começar a planejar seu aplicativo! Não se prenda a coisas que não entender, porque é sempre possível retornar mais tarde!

Objetivos de aprendizagem:

Nesta unidade, você...

- Aprenderá sobre condicionais e como escrevê-las
- Aprenderá como usar operadores de lógica no seu código
- Aprenderá como usar loops "for" (para) e "while" (enquanto)
- Criar um aplicativo que pesquisa uma base de dados

Revisão

Antes de começar esta unidade, vamos revisar alguns conceitos das unidades 1 e 2 de programação que você usará nesta unidade.

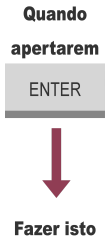
- **Algoritmos** são instruções passo a passo para o computador seguir.
- **Booleanos** são tipos de dados com dois valores: verdadeiro e falso.
- **Event Handlers** dizem ao seu app o que fazer quando algo acontece.
- **Variáveis** - Dados que podem mudar de valor.

Jogo

Antes de começar a unidade, jogue este jogo. Ele ajudará você a se familiarizar com os conceitos que aprenderá. Se você não conseguir resolver os cinco níveis, tente voltar aqui depois que completar esta unidade! [Robô Made With Code]

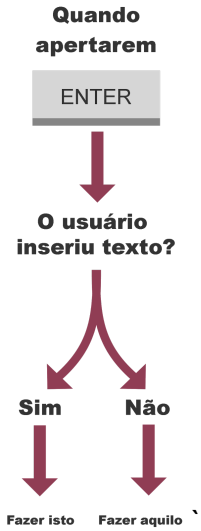
Condicionais

Até agora, você deve ter usado event handlers várias vezes! Nos desafios passados, quando seu usuário apertava um botão, o aplicativo fazia algo. Por exemplo, no aplicativo de lista de tarefas, o botão *enter* adicionava itens à lista. O **evento** era o usuário apertar "enter" e a forma como o aplicativo respondia era adicionando o texto do usuário à lista de tarefas. O código do aplicativo funcionava da seguinte maneira:



É possível ter notado no aplicativo exemplo, ou na sua "lista de tarefas", que o aplicativo permitia adicionar uma caixa em branco para a lista. Por causa disso, o usuário poderia gerar uma lista muito longa sem itens.

Digamos que você não quisesse que ele pudesse adicionar uma caixa em branco para a lista. Então, toda vez que o usuário apertasse "enter", o código perguntaria para o aplicativo: "o usuário escreveu alguma coisa"? Se a resposta for sim, então o aplicativo adicionaria o texto na lista. Se for não, então o aplicativo não faria nada. Agora, seu código deve parecer o seguinte:



Essa é uma maneira inteligente de consertar o problema, mas é importante lembrar que os computadores não pensam por si próprios, então não é possível perguntar coisas como se estivesse perguntando a outra pessoa. Na programação, quando quiser fazer uma pergunta ao aplicativo, você pode programar para ele verificar uma **condição**. Uma condição é algo que um computador determina como **verdadeiro** ou **falso**.

Quando programar seu aplicativo para verificar uma **condição**, devem haver apenas duas possibilidades de resultado. Verdadeiro, ou "true", significa que sim, a condição é verdadeira. Falso, ou "false", significa que não, ou seja, que a condição é falsa. É possível comandar que o aplicativo faça várias coisas dependendo da condição ser verdadeira ou falsa. Essa é uma maneira de perguntar a seu aplicativo, através de uma condicional, se o usuário digitou texto:

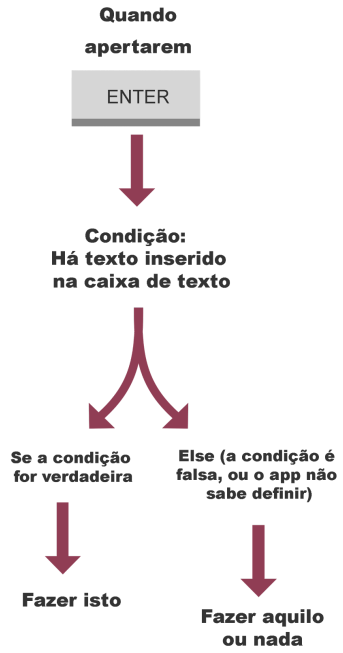
Lembra-se dos dados booleanos vistos na Unidade 2: Dados e variáveis? **Booleanos** são tipos de dados que podem ser verdadeiros ou falsos. Quando seu aplicativo avaliar uma condição, ele dará um valor booleano (output)!

Estruturas condicionais

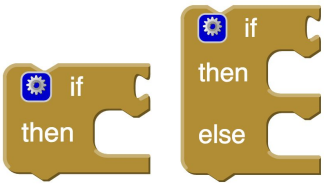
Estruturas If/Else

Agora que sabe o que é uma **condição**, como ela pode ser usada na programação? Na programação, é possível pedir para seu computador avaliar condições escrevendo **estruturas condicionais**. **Estruturas condicionais** são a maneira pela qual computadores tomam decisões. Estruturas condicionais devem sempre ter uma parte **if**, ou **se**, que diz o que o aplicativo deve fazer se a condição for verdadeira. Estruturas condicionais também costumar ter um parte **else,** ou **outro**, que fala para seu aplicativo o que fazer quando a condição é falsa. Se você deixar a parte **else** de fora, seu aplicativo não fará nada quando a condição for falsa. Agora, o código para quando o botão "enter" for apertado deve

se parecer com isto:

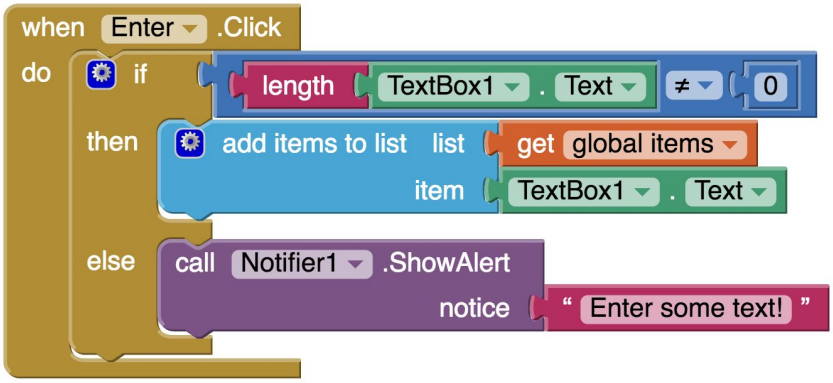


Estruturas condicionais se parecem com isso no App Inventor.



Os blocos funcionam assim. Coloca-se uma condição depois de **if**. Você programa o que seu aplicativo deve fazer se a condição for verdadeira ao lado de **then**, ou **então**, e coloca o que quer que seja feito se a condição for falsa ao lado de **else**. Se a condição for verdadeira, apenas o código para **then** será executado e tudo aquilo do lado de **else** será ignorado. Se a condição for falsa, todo o código para **then** será ignorado e tudo aquilo ao lado de **else** será executado.

Veja o exemplo. O bloco condicional fica assim após adicionar o código.



Aqui, quando o usuário clica em "enter", o aplicativo vai avaliar a seguinte condição: "o comprimento da *string* digitada na caixa de texto não é igual a zero". Se a condição for verdadeira (o comprimento da string não for igual a zero), isso significa que o usuário digitou texto, então o aplicativo adicionará o que foi escrito à lista. Se a condição for falsa (o comprimento da string é igual a zero), então o aplicativo vai ignorar o código escrito para "then" e passará direto para a parte "else", que alerta o usuário para escrever algo.

Atividade

Você consegue pensar em estruturas condicionais que ajudam a tomar decisões todos os dias? Para começar, aqui vão alguns exemplos:

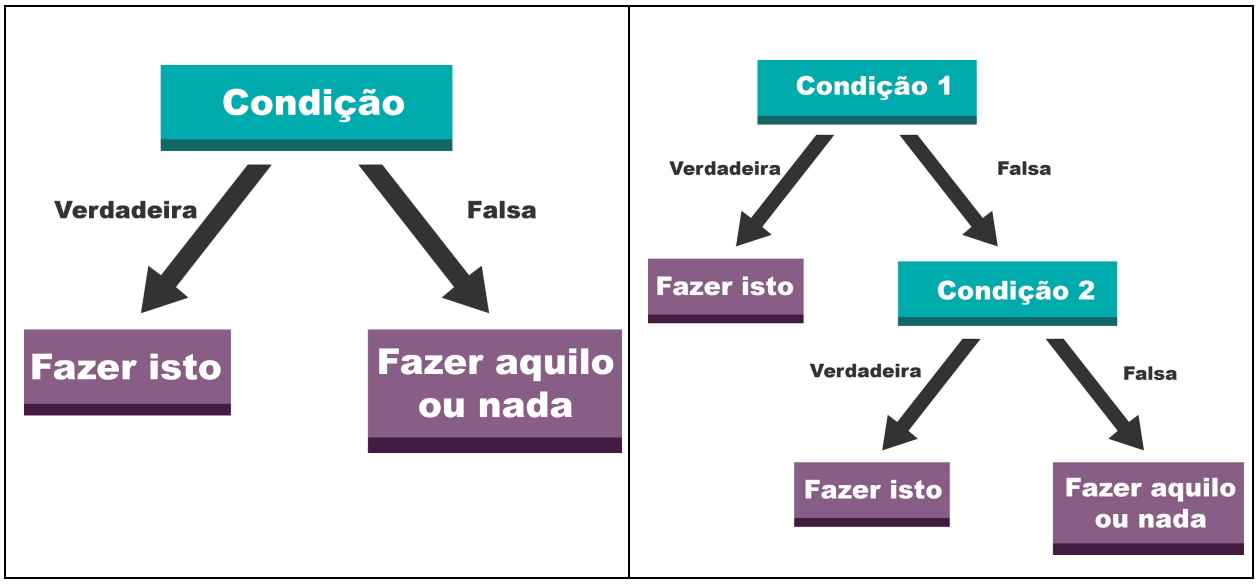
- Se estiver chovendo, leve um guarda-chuva
- Se estiver com fome, faça um lanche, se não, espere até mais tarde
- Se seu cachorro estiver chorando, leve-o para uma caminhada
- Se estiver frio lá fora, leve uma jaqueta
- Se seu cabelo estiver embaraçado, penteie-o
- Se você tiver
- Se você estiver doente, vá ao médico
- Se seu celular estiver sem bateria, carregue-o

Estruturas ElseIf

Até agora você aprendeu como escrever estruturas que avaliam uma condição e têm dois resultados possíveis. Nesta seção, você aprenderá sobre como fazer estruturas condicionais maiores!

Else if (Se não se) permite adicionar outra condição à estrutura condicional. A segunda condição será avaliada após a primeira e só no caso de a primeira ser falsa. Se a segunda também for falsa, então o aplicativo fará o que estiver em **else**, ou fará nada. Aqui está um diagrama para ajudar a entender.

Estrutura condicional com dois resultados possíveis.	Estrutura condicional com três resultados possíveis. A **Condição 2** só é avaliada caso a **Condição 1** seja falsa.



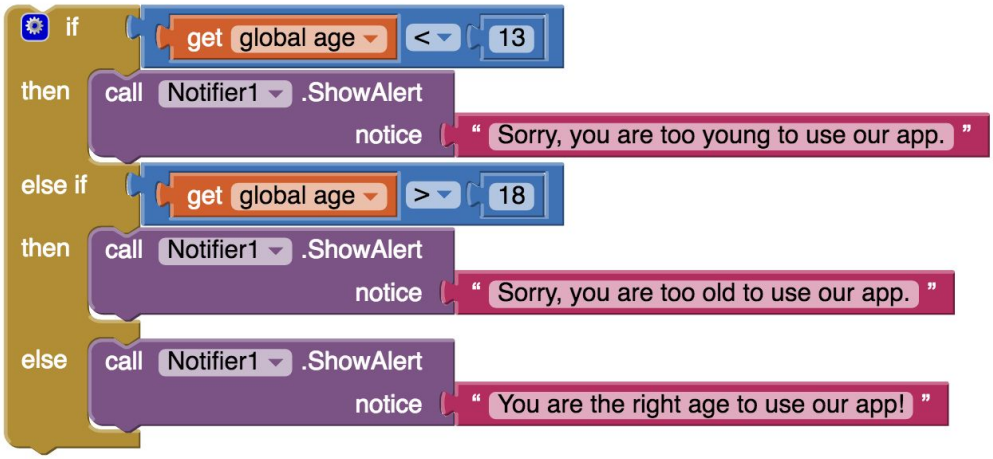
Os blocos se parecem com isto no App Inventor:

A primeira condição vai ao lado de **if** e a segunda condição vai ao lado de **else if**. Vamos ver um exemplo:

Você está desenvolvendo um aplicativo apenas para usuários de 13 a 18 anos e quer que o aplicativo avise os usuários caso sejam novos ou velhos demais para usar o aplicativo. Existem três resultados possíveis:

1. O usuário é novo demais
2. O usuário é velho demais
3. O usuário tem a idade adequada

O usuário digita a idade no aplicativo e ele armazena isso numa variável chamada "age" ou idade. Veja como usar uma estrutura **if/else if/else** para verificar se os usuários têm a idade certa.



A primeira condição testa se a idade for menor que 13. Se for, o aplicativo avisa que o usuário é muito novo. Se não for, o usuário testa se a idade é maior que 18. Se for, o aplicativo avisa que o usuário é muito velho. Se não for mais velho que 18, o aplicativo avisa que a idade é adequada.

É possível adicionar quantas condições quiser usando a estrutura **else if**, mas é importante notar a *ordem* em que o aplicativo avalia essas condições. O código deve sempre começar com a primeira condição e depois continuar até encontrar uma condição verdadeira. Pode-se pensar que o código está procurando por uma "frase verdadeira". Ao encontrar uma condição verdadeira, ele executa o código escrito abaixo disso. Se nenhuma for verdadeira, ele executa o que estiver escrito em **else**.

Aqui estão algumas coisas para se ter em mente quando usar estruturas do tipo **If/ Else/ Else If**

- Você pode testar quantas condições quiser
- A estrutura funciona de cima para baixo, então coloque a condição que você quer testar antes no topo da estrutura
- Apenas uma coisa acontecerá, e ela será aquilo que for definido para a primeira condição que for verdadeira

Para mais informações sobre condicionais, acesse o site do MIT:
http://appinventor.mit.edu/explore/ai2/support/blocks/control.html

Atividade

Identificar quais condições seu aplicativo deve avaliar pode ser complicado! Algumas vezes, você saberá o que quer que o aplicativo faça, mas pode ser difícil saber quais condições verificar. Pratique escrevendo condições que você acha que os aplicativos verificam quando você os usa. Aqui está um exemplo para começar! Veja se você consegue pensar em outros.

Entrar em uma conta de rede social:

- Condições para verificar:
    - O nome do usuário está correto
    - A senha está correta
    - O usuário não tentou logar mais do que cinco vezes

Outra parte complicada de se escrever estruturas condicionais é traduzir a condição para algo que seu aplicativo possa realmente entender. No exemplo anterior, verificamos a idade do usuário ao criar uma variável igual à idade do usuário. Criar condições não é algo intuitivo, leva um tempo de prática! Perguntas para se pensar ao definir condições:

1. A sua condição depende das informações digitadas pelo usuário?
2. Serão necessárias variáveis?
    a. Se sim, de que tipo? Numeral, lista ou string?
3. Você pode usar operadores numéricos, como menor que (<) , maior que (>), igual a (=), diferente de (≠) na sua condição?
4. É preciso comparar o valor de algo com um valor em uma base de dados?
5. É preciso comparar algo com dados de outros locais, como sites? (Isso será discutido com mais profundidade na unidade de programação 4)
6. De quais tipo de dados a condição depende? Você vê alguma operação para esse tipo de dados que ajudará você?
    a. Ex: Comparar duas strings, descobrir o comprimento de uma string, adicionar itens a uma lista, comparar duas listas etc.
    b. Tente ver as funções do App Inventor rapidamente para ver se elas ajudam você.

Hora de praticar! A partir de um ou dois exemplos que você identificou acima, tente traduzi-los para código.

Lógica

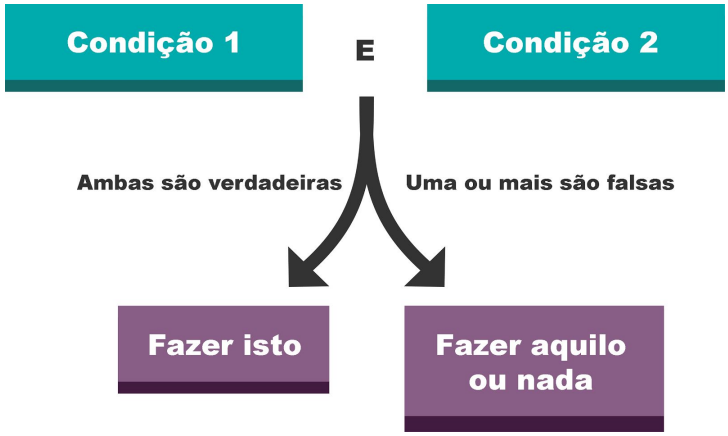
Até agora, você aprendeu a comandar seu aplicativo a fazer coisas diferentes usando **estruturas condicionais**. É possível lidar com muitos novos problemas conhecendo alguns **operadores de lógica**. **Operadores de lógica** usam booleanos tanto como input, entrada, quanto como output, saída. Como o seu aplicativo vê condições como verdadeiras ou falsas, você pode usar as condições como inputs para booleanos. Há três funções principais de **lógica** que veremos nesta seção: **AND**, **OR** e **NOT**

Operador AND

A função **AND** (E) pega dois inputs e ambos devem ser verdadeiros para o output também ser verdadeiro. Caso alguma das condições seja falsa, o código avaliará a estrutura toda como falsa. As possibilidades de resultado usando o operador **AND** (E) são as seguintes:

Verdadeiro E Verdadeiro = Verdadeiro	

Verdadeiro E Falso = Falso	
Falso E Verdadeiro = Falso	
Falso E Falso = Falso	



Observação: Usar o operador **AND** é diferente de usar **else if**, porque ambas as condições serão avaliadas ao mesmo tempo, ao invés de consecutivamente. Ambas precisam ser verdadeiras para o aplicativo ver a condicional como verdadeira.

Exemplos

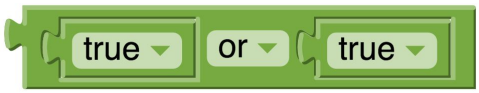
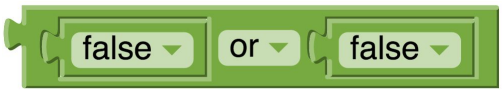
O operador **AND** deve ser usado quando duas condições devem ser verdadeiras para algo acontecer. Exemplos de como você pode usar **AND** no seu aplicativo!

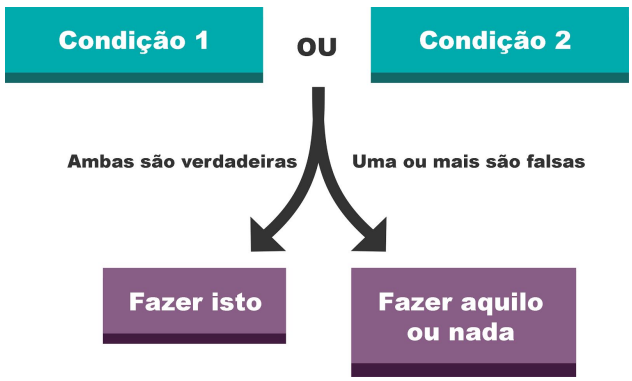
- Fazendo login na mídia social:
  - Se (o nome de usuário estiver certo) **E** (a senha estiver correta) ---> permitir que o usuário faça o login
- Vencendo um jogo
  - Se (o jogador terminar o nível) **E** (o tempo for menor que 3 minutos) ---> o usuário vence
- Encontrando uma loja:
  - Se (a loja estiver aberta) **E** (próxima do usuário) ---> a loja é mostrada nos resultados
- Postando fotos:
  - Se (o usuário selecionou uma foto) **E** (o usuário inseriu um caption) ---> o usuário pode postar a foto

Você consegue pensar em mais exemplos?

Operador OR

A função **OR** (OU) avalia dois inputs e apenas um deles precisa ser verdadeiro para o output ser verdadeiro. Se nenhuma das condições não for verdadeira, seu código avaliará a estrutura como falsa. Os possíveis resultados usando o operador **OR** são os seguintes:

Verdadeiro OU Verdadeiro = Verdadeiro	
Verdadeiro OU Falso = Verdadeiro	
Falso OU Verdadeiro = Verdadeiro	
Falso OU Falso = Falso	



Observação: O operador **OR** pode também ser similar ao **else if**. Usar o operador **OR** é diferente porque ambas as condições serão avaliadas ao mesmo tempo, ao invés de consecutivamente. O operador **OR** é melhor quando você tiver duas condições que devem ter o mesmo resultado, caso sejam verdadeiras.

Exemplos

- Perdendo um jogo
  - (se o tempo acabar) **OU** (o jogador perder todas as vidas) ---> o jogador perde
- Mostrando resultados de pesquisa

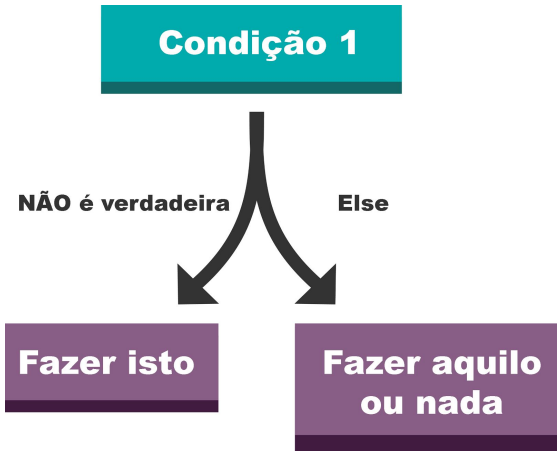
    - (se o título for igual) **OU** (a descrição for igual) ---> o resultado da busca é mostrado
- Recomendando ao usuário vídeos para assistir
    - (Se o usuário já assistiu) **OU** (ele é semelhante a algo que o usuário gostou) --- > o vídeo é recomendado para o usuário

Você consegue pensar em mais exemplos?

Operador NOT

O operador **NOT** (NÃO) é mais fácil! O operador **NOT** troca o valor de um booleano para o valor oposto.

NÃO é verdadeiro = falso	
NÃO é falso = verdadeiro	



Às vezes, é mais fácil verificar uma condição para o oposto daquilo que você quer. Para os seguintes cenários, é possível usar o operador **NOT**.

Exemplos

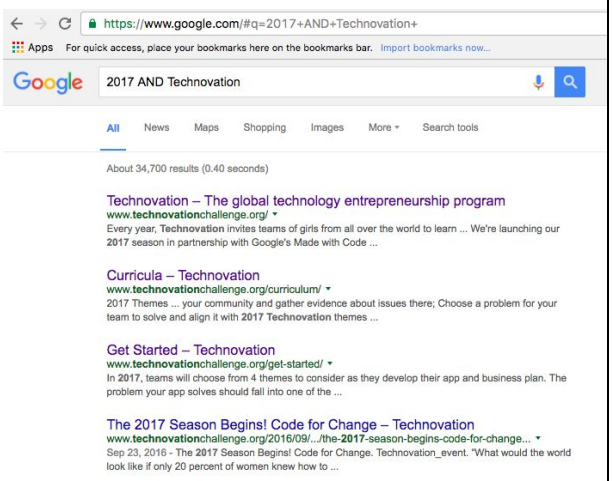
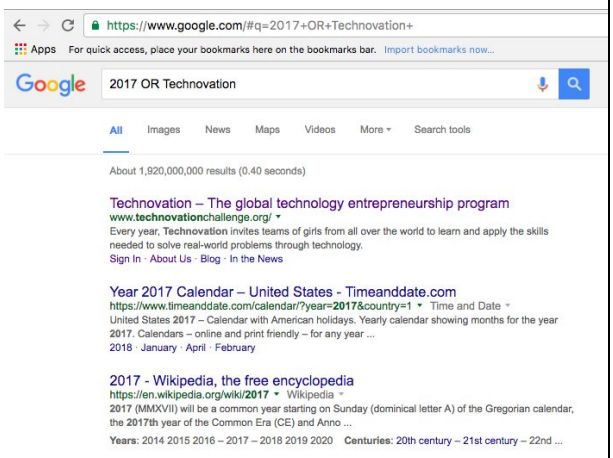
- Você quer verificar se há um texto na caixa de texto, mas é mais fácil verificar se a caixa de texto está vazia, então você usa o operador **NÃO**.
    - Se **NÃO**(textbox = “ ”) ---> fazer isso
- Você quer excluir resultados de pesquisa sobre cachorrinhos, mas é mais fácil mostrar resultados que contém "cachorrinhos", então você usa o operador **NOT**.
    - Se **NÃO**(artigo contém a palavra "cachorrinho") ---> os resultados são mostrados
- Você pode combinar **operadores de lógica** para fazer mais coisas. Para excluir resultados

sobre cachorrinhos, mas incluir resultados sobre gatinhos, a seguinte forma pode ser usada:

- Se **NÃO**(artigo contém a palavra "cachorrinho") **E** (artigo contém a palavra "gatinho") ---> os resultados são mostrados

Atividade

Sabia que é possível usar **AND**, **OR** e **NOT** em uma pesquisa no Google? Essa é uma boa forma de aumentar ou reduzir o número de resultados de uma busca quando estiver pesquisando algo! Tente você mesma! Certifique-se de usar **AND**, **OR** ou **NOT** em letras maiúsculas. Veja os exemplos abaixo:

Pesquisando 2017 **AND** Technovation gera resultados que contenham tanto 2017 quanto Technovation	Pesquisando 2017 **OR** Technovation gera resultados que contenham 2017 ou Technovation
	

Loops

Computadores são ótimos para fazer a mesma coisa repetidas vezes de maneira rápida e sem erros! Você pode aproveitar isso através de **loops**. **Loop** é um bloco de código que se repetirá várias vezes.

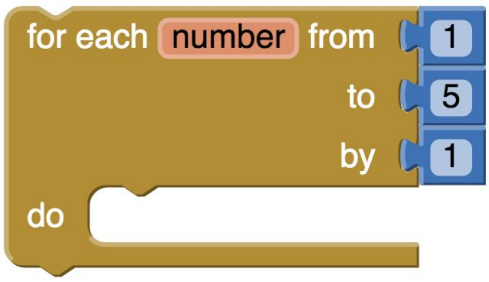
Há dois tipos de loops, **loops while** e **loops for**. **Loops while** continuam fazendo algo até que uma condição deixe de ser verdadeira. **Loops for** fazem algo um número definido de vezes.

Loops For

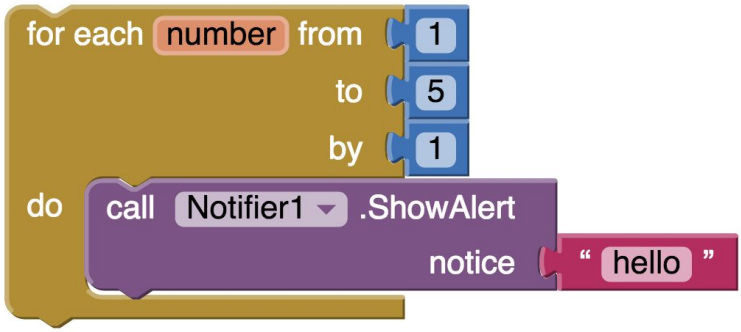
Loops for repetem um bloco de código um número definido de vezes. A razão de serem chamados de **for** (por) é porque você pode dizer para seu aplicativo **por** quantas vezes quer que repita o código. Pense nisso como os loops comandando seu aplicativo "repita isso **por** 15 vezes" ou "repita isso **por** 5 vezes".

Loops for usam uma variável chamada de **counter**, ou contador, para contar quantas vezes o código foi repetido. É possível controlar quantas vezes o loop repete definindo onde começa e onde termina o **counter**. Também é possível definir em quanto o **counter** aumenta cada vez que o código repete. Na maioria dos cenários, o **counter** aumenta em 1 por repetição do loop.

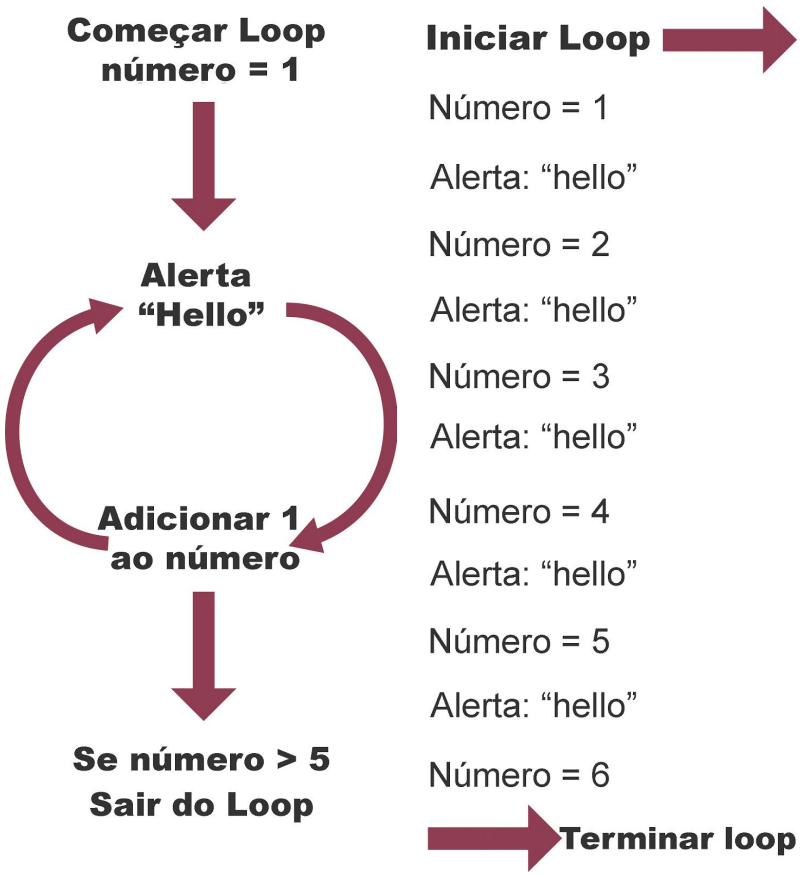
No App Inventor, os **loops for** são assim:



A parte que diz *number* é o **counter**. O nome do **counter** é *number*, mas você pode alterar isso. O *number* começa em 1 e para quando for igual a 5. Cada vez que o código no loop repete, o *number* aumenta em 1. Então, esse loop repetirá o código 5 vezes. Agora, esse loop não faria nada, porque a parte *do*, que diz o que ele deve *fazer*, está vazia. Veja o exemplo.

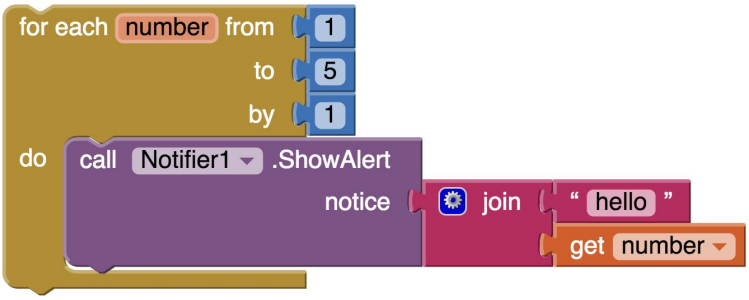


Nada foi alterado na variável *number*, mas um pouco de código foi adicionado à parte *do* do loop. Cada vez que o loop for executado, o aplicativo enviará um alerta "hello" para o usuário, então, o usuário será alertado 5 vezes. O aplicativo executará o loop da seguinte maneira:

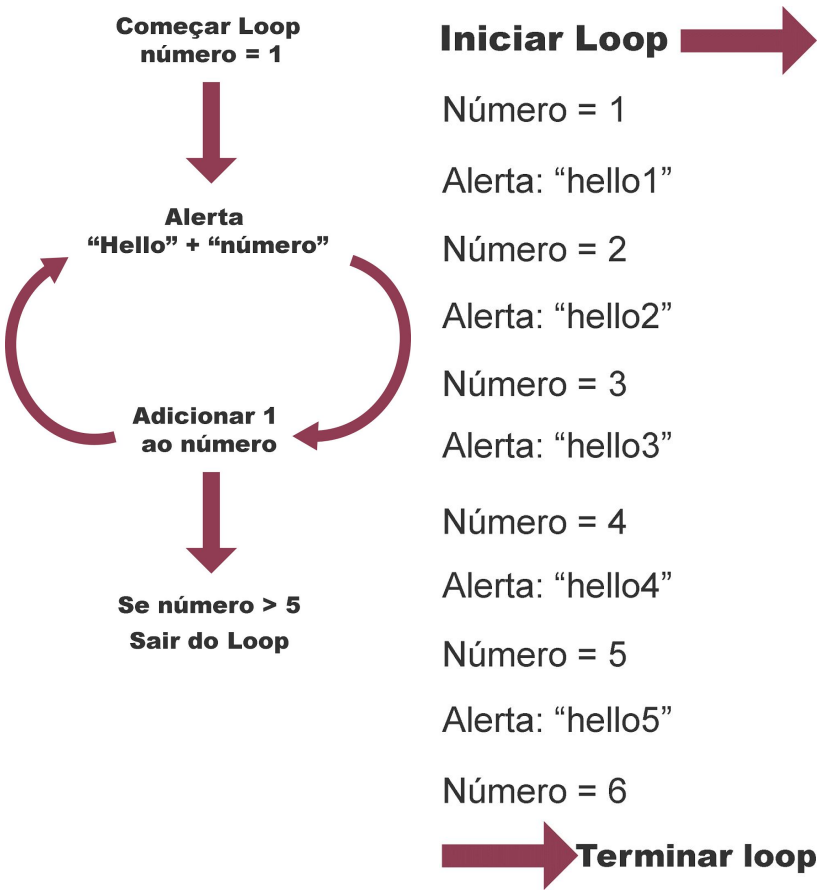


Esse loop seria útil caso quisesse alertar "hello" para o usuário 5 vezes. Isso pode não ser tão útil, já que não é difícil colocar 5 blocos de notificação consecutivos dizendo "hello". Mas e se quisesse alertar "hello" para o usuário 100 vezes? Seria muito mais fácil fazer um loop do que colocar 100 blocos consecutivos!

Outra maneira em que **loops for** podem ser úteis é usando a **variável counter** no código. Cada vez que o loop for executado, a **variável counter** terá um valor diferente, e isso pode ser muito útil. Veja o exemplo.

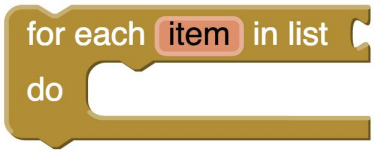


Neste loop *for*, usamos a variável *number* ligada à palavra "hello". *O number* aumenta em 1 a cada vez, então o aplicativo mostrará coisas diferentes toda vez que executar. O aplicativo executará o loop da seguinte maneira agora:



Loops For each

Outra maneira de se usar **loops for** no App Inventor é a seguinte:



Aqui, a **variável counter** é chamada *item* e já vem configurada para repetir o número de itens em uma lista. Esses loops podem ser úteis sempre que precisar fazer algo com uma lista. Veja o exemplo.

Digamos que tenha uma lista de números e queira somar todos os números da lista e armazená-los em uma variável chamada *sum*, ou somatório. Com um **loop for each**, isso seria feito da seguinte maneira.

Cada vez que o loop é executado, a variável *sum* terá um item da lista *numberList* adicionado a ela. O loop automaticamente para depois que todos os números da lista forem adicionados!

Observação: Você pode ter notado que as **variáveis counter** nesta seção se aproximam bastante das **variáveis locais** que aprendeu na unidade de programação 2! Se percebeu isso, você está correta! Assim como variáveis locais, elas só serão utilizadas dentro do loop.

Loops While

Loops while são loops que se repetem até que uma condição deixe de ser verdadeira. A razão de se chamarem **loops while** (**enquanto**) é porque o código se repete **enquanto** uma condição permanece verdadeira. Pense nisso como os **loops while** comandando seu aplicativo a "**enquanto** isso acontece, repita isso" ou "**enquanto** isso não mudar, repita isso".

Exemplo
Você está numa festa e quer que a música toque até que todos os convidados vão embora. Sua festa pode ser descrita com o seguinte loop:

- While (pessoas na festa > 0)
- do: continuar tocando música

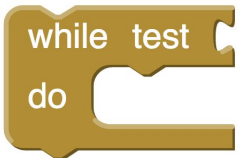
E se quiser que a música pare de tocar quando ficar mais tarde que meia-noite? Através de **lógica**, é possível programar *loops while* para terminarem baseados em várias condições. Agora, sua festa pode ser descrita com o seguinte loop:

- While (pessoas na festa > 0) **OR** (horário < meio-noite)
- do: continuar tocando música

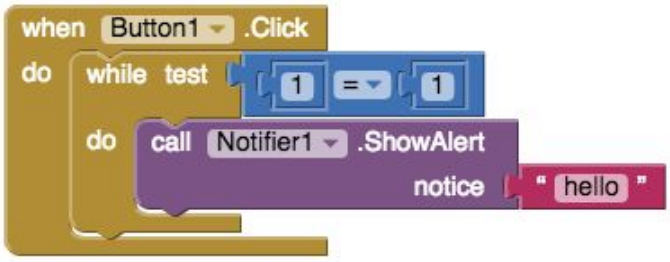
Nesse caso, a música pararia assim que todos forem embora, ou se passar da meia-noite.

Para usar um **loop while,** é necessário definir uma condição que começa como *verdadeira*. Se sua condição começar como *falsa*, o loop nunca será executado. O loop verifica a condição antes de repetir para verificar se a condição ainda é verdadeira.

Os **loops while** são assim no App Inventor:



É que os **loops while** encontrem erros! Se você escolher uma condição que nunca será *falsa*, seu loop nunca terminará. Isso se chama **loop infinito**. Veja o exemplo.



Como 1 é sempre igual a 1, essa condição nunca será falsa! Ao executar esse código no App Inventor, o celular vai parar e não conseguiremos fazer nada. É possível também receber uma mensagem que diz que o App Inventor parou de funcionar.

Reflexão

Parabéns por completar essa aula! Por mais que fosse pesada, ela será muito útil para a criação do seu aplicativo! Não se esqueça de que é sempre possível revisar algo que não tenha entendido.
Palavras-chaves:

- **Booleanos -** tipo de dados que podem ser verdadeiros ou falsos
- **Condições -** algo que um aplicativo avalia como verdadeira ou falsa
- **Estrutura condicional -** diz ao aplicativo o que fazer após avaliar as condições
- **Operador AND** - avalia como verdadeira se todos os inputs foram verdadeiros
- **Operador OR** - avalia como verdadeira se um dos inputs for verdadeiro
- **Operador NOT -** avalia como o oposto do input
- **Loop -** um bloco de código que se repete
- **Loops For** - repete um bloco de código um número definido de vezes.
- **Loops For each** - repete um bloco de código um número definido de vezes em uma lista.
- **Loop While** - repete um bloco de código enquanto uma condição for verdadeira

Desafio de programação #3

Descrição: Crie um aplicativo que permita ao usuário buscar informações em uma base de dados e ver os resultados. A base de dados deve conter nomes e descrições de pessoas famosas, locais, filmes, músicas, eventos ou o que mais você quiser. Certifique-se de ter, no mínimo, três entradas na base de dados.

Tente fazer o desafio sozinha antes de ler as instruções. As instruções são apenas uma maneira de solucionar o desafio. Baixe o aplicativo exemplo na loja Google Play : https://play.google.com/store/apps/details?id=appinventor.ai_alliec.DatabaseSearch
Boa programação!

Para nosso aplicativo, criamos uma base de dados de cientistas mulheres famosas e incríveis. Sinta-se à vontade para copiar estes textos ou criar seus próprios dados!

Nome (tag)	**Descrição:**
Ada Lovelace	Ada foi uma escritora e matemática inglesa que viveu no século XIX e é conhecida principalmente por seu trabalho no computador mecânico para vários propósitos de Charles Baggage, a Máquina Analítica. Ada é muitas vezes considerada a primeira pessoa a programar, por ter escrito o primeiro algoritmo a ser utilizado por uma máquina.
Marie Curie	Marie foi uma física e química nascida em Varsóvia, Polônia, em 1867. Ela descobriu 2 novos elementos radioativos, junto com seu marido Pierre.

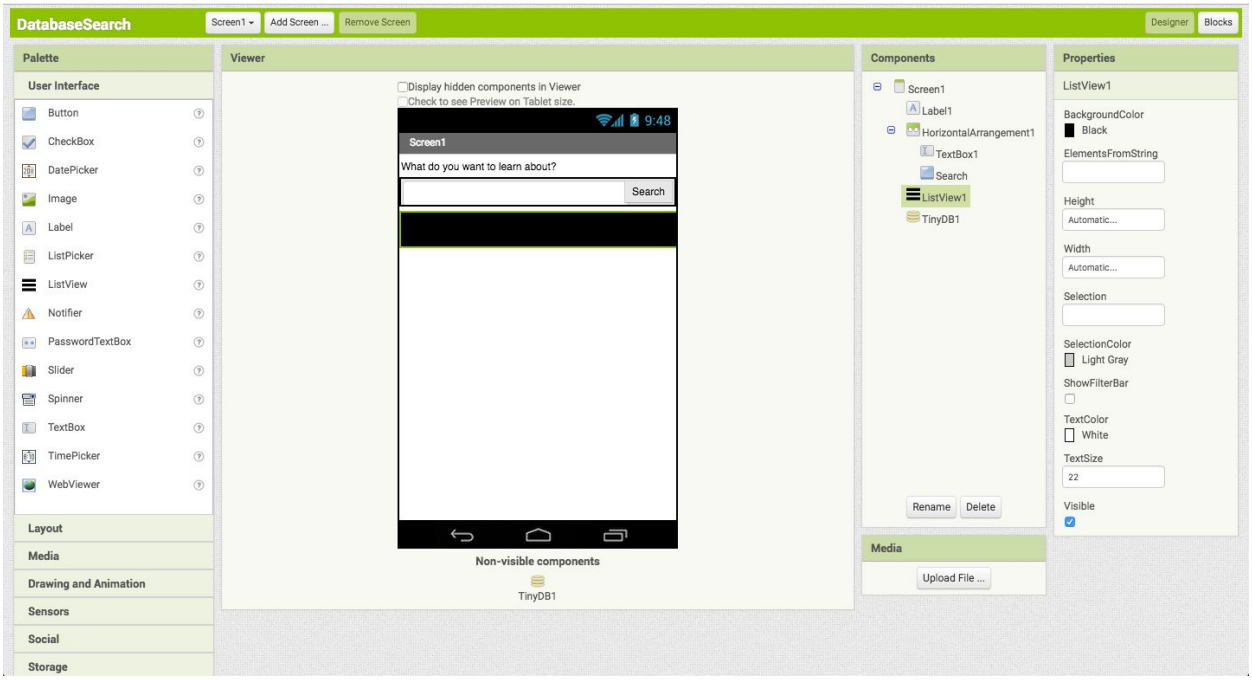
	Marie ganhou o Prêmio Nobel em 1903. Além disso, ela também descobriu o rádio gasoso, que poderia ser usado para o tratamento de câncer.
Alice Ball	Alice foi uma química nascida em Seattle, nos EUA, em 1892. Ela foi a primeira pessoa negra e primeira mulher a se formar na Universidade do Havaí. Com apenas 23 anos, Alice desenvolveu uma cura para a lepra, que era vista como incurável antes de sua pesquisa.
Sau Lan Wu	Sau Lan é uma física de partículas nascida no começo da década de 40, durante a ocupação japonesa de Hong Kong. Ela possui doutorado em Harvard. Liderou a equipe de pesquisa que descobriu o glúon. Ela é uma das mais importantes físicas de partículas na sua área e fez muitas descobertas inovadoras.
Patricia Bath	Patricia é uma oftalmologista e inventora nascida em 1942, no Harlem, na cidade de Nova York. Ela terminou o Ensino Médio em apenas dois anos e meio e já sabia que queria ser médica. Em 1985, ela completou sua invenção que remove cataratas e restaura a visão de pessoas ao redor do mundo.

Plano geral

Decidimos abordar esse desafio armazenando os dados em uma TinyDB. Armazenamos a descrição das cientistas com nomes de tag com seus próprios nomes. Após isso, fizemos duas listas, uma com o nome da pessoa e outra com as descrições. O usuário digitará o texto e o aplicativo buscará corresponder itens em ambas as listas. Se houver correspondência, o nome da mulher aparecerá no ListView. O usuário poderá então selecionar a cientista da qual quer saber mais e uma nova tela se abrirá com a descrição do seu trabalho. Para esse aplicativo, usamos uma TinyDB para armazenar os dados e usá-los em telas separadas.

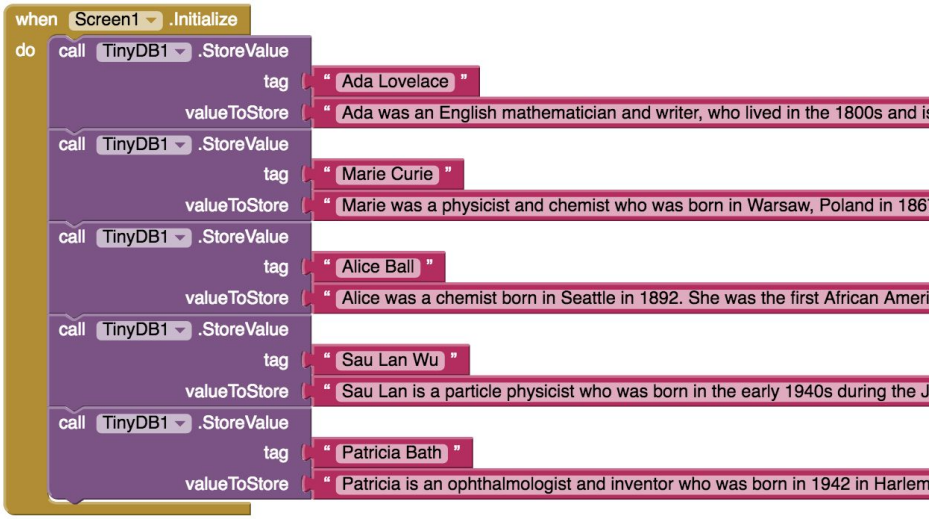
Design da tela

Queremos permitir ao usuário pesquisar em uma base de dados, então adicionaremos uma caixa de texto para o usuário digitar e um botão para clicar quando quiser pesquisar. Já que sabemos que usaremos uma base de dados para armazenar os dados, então arrastamos uma TinyDB para a tela. Como também usaremos listas, adicionamos uma ListView para a tela.



Como adicionar dados à base de dados

Decidimos adicionar o nome da cientista como uma tag e a descrição do seu trabalho como um valor da base de dados. Copiamos e colamos as informações que queremos adicionar à base de dados assim:

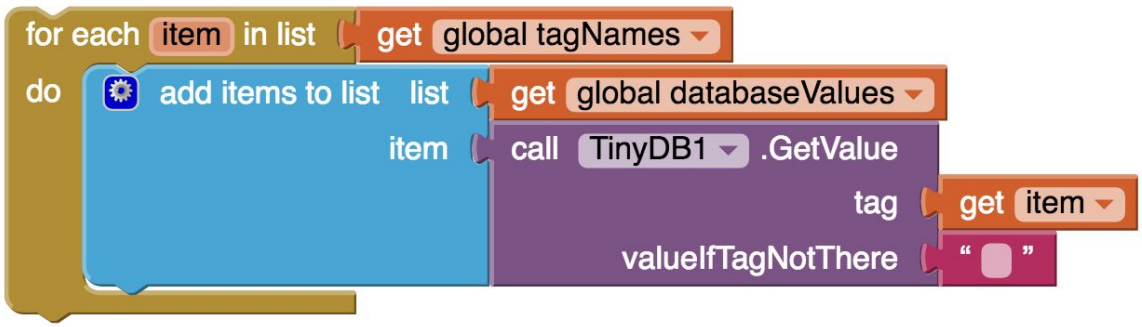


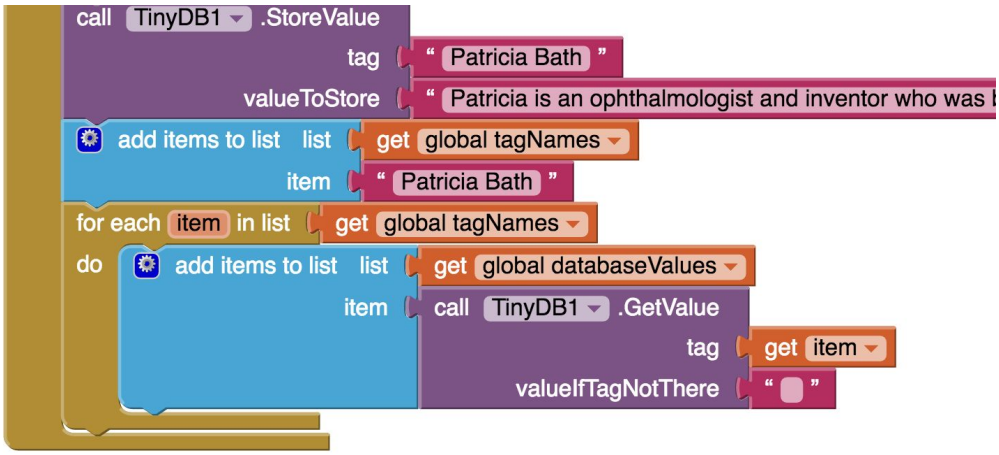
Você deve armazenar as informações na base de dados assim que a tela inicializar, então essa deve ser a primeira coisa que seu aplicativo faz. Será necessária uma base de dados para armazenar essas informações, caso queira usá-la em outras telas.

Como criar listas de nomes de tag e valores de bases de dados

Criamos uma lista para armazenar os nomes de tag. Criamos uma lista vazia chamada "TagNames". Depois, adicionamos blocos para cada um dos nomes de tag na lista. É importante se certificar de que os nomes de tag adicionados à lista correspondam exatamente aos nomes de tag na base de dados, inclusive letras maiúsculas e minúsculas.

Agora que temos uma lista completa com todos os nomes de tag, precisamos criar outra lista com todos os valores da base de dados na mesma ordem que os nomes de tag. Primeiro, criamos uma lista vazia chamada “databaseValues” para guardar todos os valores da base de dados. Depois, adicionamos valores à lista na mesma ordem correspondente às tags, usando um loop for. Para cada item na lista "tagNames", adicionamos um valor da base de dados à lista "databaseValues". Então, adicionamos o loop for ao event handler When Screen1.Initialize, que é quando a Tela 1 for inicializada.





Checkpoint

Neste ponto, você deve ter todos os dados na base de dados, bem como uma lista com todos os nomes de tag e outra com os valores da base de dados. Certifique-se de que cada uma das listas contém os valores que você quer! Como seu aplicativo ainda não faz nada, você pode verificar seu trabalho exibindo suas listas em ListView e confirmando se está tudo lá. Adicione esse bloco ao seu código embaixo do loop for no event handler *when Screen1.Initialize* para confirmar se a lista "tagNames" está correta.

Adicione esse bloco ao seu código embaixo do loop for no event handler *when Screen1.Initialize* para confirmar se a lista "databaseValues" está correta.

Após verificar se as listas estão corretas, certifique-se de deletar os blocos!

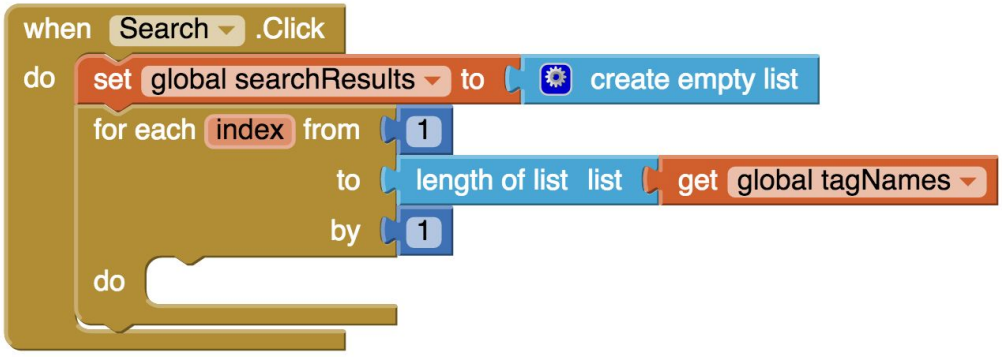
Como programar o botão "Search"

Queremos que o usuário possa pesquisar na base de dados digitando na caixa de texto e apertando o botão "search" para pesquisar. Para fazer isso, colocaremos um event handler no botão que buscará em todos os itens das listas "tagNames" e "databaseValues" por strings que correspondam ao que o usuário digitou na caixa de texto. Se houver correspondência, adicionamos o nome da cientista ao ListView. Veja como fizemos.

Primeiro, criamos uma variável chamada "searchResults", ou resultados da busca, e definimos ela a uma lista vazia:

Depois, adicionamos um event handler ao botão de busca. Dentro dele, adicionamos um bloco para definir os resultados de busca novamente para uma lista vazia. Isso porque, cada vez que o usuário clica em "search", queremos limpar os itens que já tenham sido adicionados à lista "searchResults" na última busca.

Depois, definimos um **loop for** para buscar nos itens das listas "tagNames" e "databaseValues". Não usamos o *loop for para cada item na lista* porque queremos usar uma variável *counter* no *loop for.* Iniciamos o *loop for* em 1 e o terminamos no número de itens na lista. Mudamos o nome da variável para *index*, ou índice.

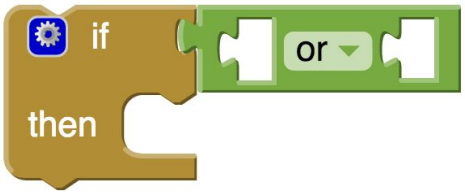


O bloco usado para comparar o texto digitado na caixa para buscar em itens da lista "tagNames" é o

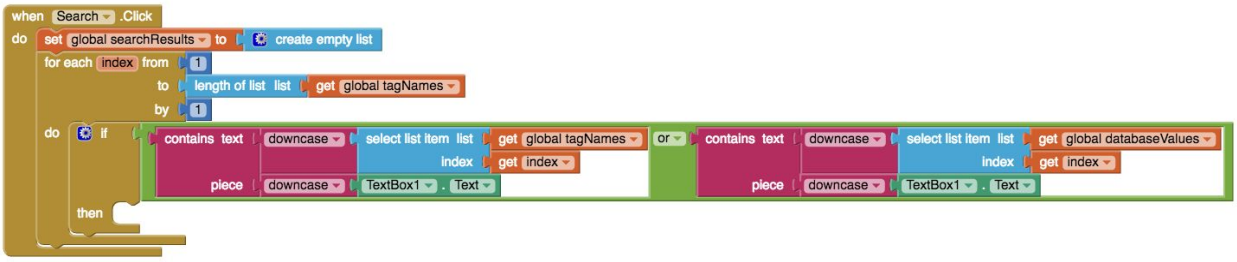
seguinte: A variável "index" aumenta em 1 para cada vez que o loop se repete, então, ela eventualmente vai comparar o texto digitado com todos os itens da lista "tagNames". Escolhemos escrever tudo em letras minúsculas para que letras maiúsculas não importem na busca. O bloco "contains text", ou contém texto, dá um valor verdadeiro se a variável *piece* aparecer no *texto*.

Para comparar a pesquisa do usuário com a lista "databaseValues", usamos o mesmo bloco, mas trocando o nome da lista.

Queremos mostrar ao usuário o nome da cientista cujo nome ou descrição pesquisa corresponde à pesquisa do usuário. Para fazer isso, usamos os dois blocos anteriores com uma condicional e uma operação de lógica OR.

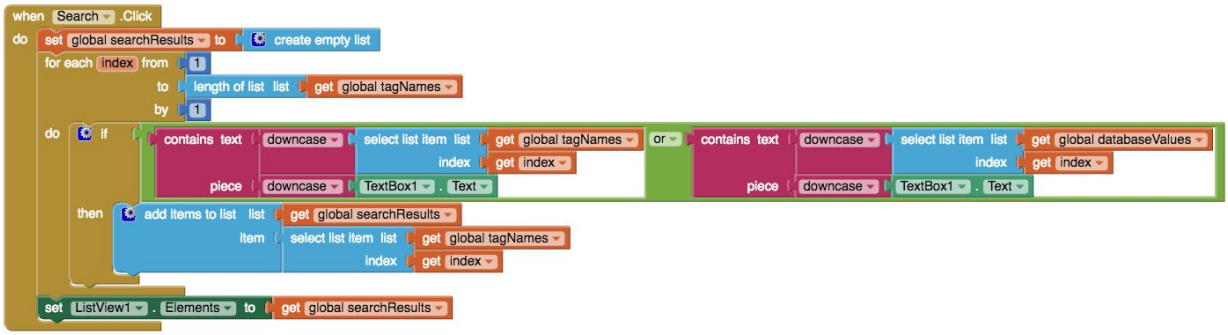


O código deve ficar assim, após preencher a operação OR.



Queremos que nosso código adicione o nome das cientistas à lista "searchResults" se o nome ou a descrição dela corresponder aos termos pesquisados pelo usuário. Para fazer isso, quando houver

correspondência, adicionamos o nome de "tagNames" usando a variável loop "index". Não queremos adicionar *else* à estrutura condicional, porque se o nome ou a descrição não corresponder, queremos que o código faça nada. Então, adicionamos os itens na lista "searchResults" para a ListView. Como esse bloco fica fora do *loop for*, o código só atualiza a ListView depois de o loop terminar.



Checkpoint

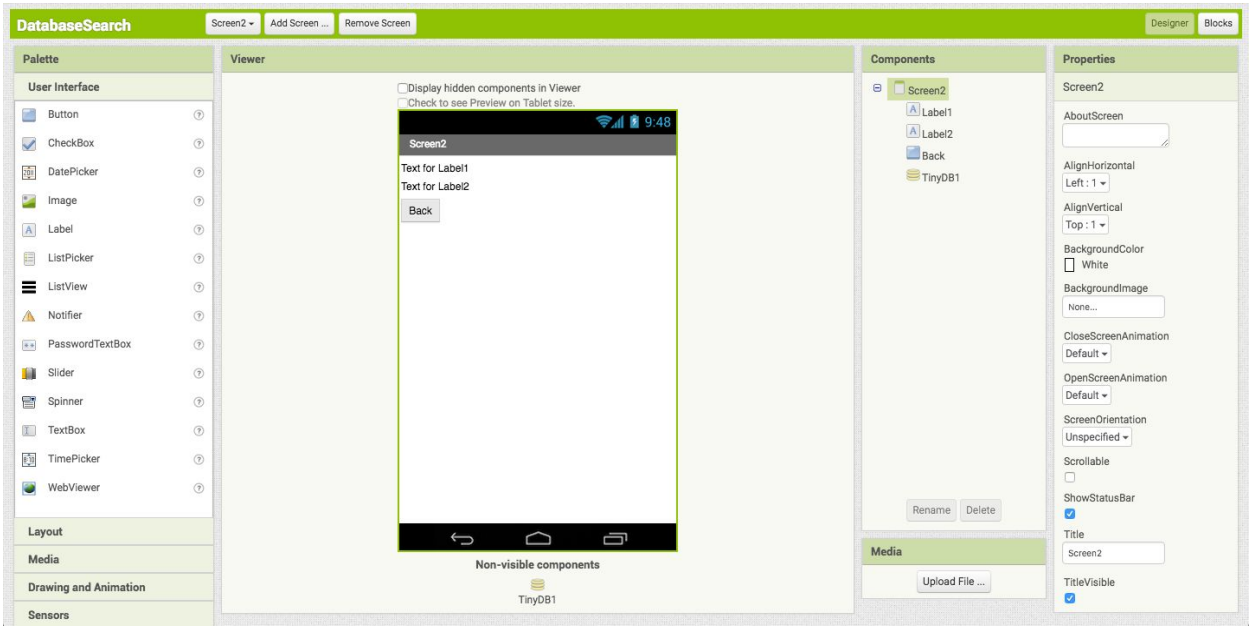
Você acabou de escrever muito código! Verifique se já consegue pesquisar algo. Tente pesquisar um nome, como "Alice" ou uma ocupação, como "química".

Bônus:

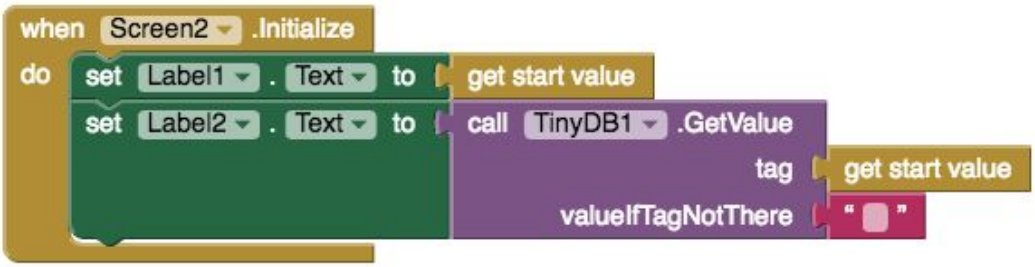
Como exibir resultados em outra tela

Decidimos que o aplicativo pode ficar melhor se o usuário puder escolher a cientista exibida nos resultados da busca para aprender mais sobre ela. Para fazer isso, quando o usuário escolher algo da lista, abrimos uma tela nova com um "startValue", ou valor inicial, com a seleção do usuário. Lembre-se, as duas únicas maneiras de passar dados entre telas é com uma base de dados ou um bloco startValue!

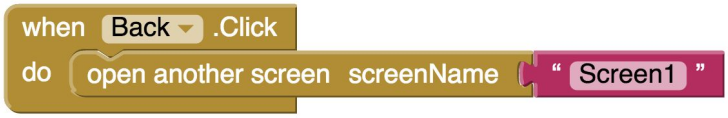
Desenhamos a segunda tela com duas labels. Uma mostra o nome da cientista e outra a descrição. Também adicionamos um botão para retornar à Tela 1. Não se esqueça que também precisamos adicionar uma TinyDB a essa nova tela!



Como o valor inicial é o nome da cientista, mostramos o valor inicial na Label1 e usamos isso para recuperar a descrição da cientista na base de dados e mostrá-la na Label2. Você encontra o bloco *get start value* da categoria *control,* ou controle.



Não se esqueça de programar o botão "back" para voltar para a primeira tela!



Checkpoint final

Certifique-se de que tudo no seu aplicativo está funcionando! Será que se você selecionar um item na ListView, você abrirá uma Tela 2 com o nome da tag e o valor da base de dados?

Ainda está com dificuldades?

Baixe o código-fonte:
https://drive.google.com/file/d/0B9HV58UviqU_YXU1eTBKMDN1ZVE/view?usp=sharing