Código

Apostila de estudo: do zero a apps, sistemas e automações

9 trilhas, 28 módulos. Assista às aulas, faça o resumo, a revisão e os desafios por conta própria, e só então confira as soluções.

Trilha 0: Fundamentos

Como tudo funciona por baixo, lógica e as ferramentas do dia a dia.

Módulo 1. Como a internet, apps e sistemas funcionam

Cliente, servidor, banco de dados e como um app chega no celular.

Aulas

- **Como a Internet funciona?** (Curso em Vídeo): Aula do curso de HTML5 e CSS3 do Gustavo Guanabara, bem de iniciante
  https://www.youtube.com/watch?v=nlO5hySqJFA

Resumo

- Cliente (navegador ou app) faz pedidos e o servidor responde.
- O banco de dados guarda as informações de forma organizada.
- O DNS traduz o endereço do site para o endereço do servidor.
- Um app chega às pessoas por uma loja (Play Store) ou pelo navegador (site/PWA).

Revisão

Responda no papel e confira o gabarito no fim da apostila.

1. O que é um servidor?
   ( ) Um tipo de cabo de internet
   ( ) O aparelho que você usa para acessar o app
   ( ) Um computador sempre ligado que responde aos pedidos de outros aparelhos
2. Onde ficam guardados os dados dos usuários de um app, como contas e pedidos?
   ( ) Dentro do arquivo HTML da página
   ( ) No banco de dados
   ( ) Na tela do celular de cada pessoa
3. Qual é a ordem de um pedido na internet?
   ( ) O banco de dados pede ao navegador
   ( ) O servidor pede, o cliente processa e responde
   ( ) O cliente pede, o servidor processa e responde
4. O que é o "cliente" quando falamos de internet e apps?
   ( ) O navegador ou app que faz pedidos e mostra a resposta
   ( ) O computador potente que guarda todos os sites
   ( ) A pessoa que paga pelo serviço do app
5. Para que serve o DNS?
   ( ) Guardar as senhas dos usuários do site
   ( ) Deixar a internet mais rápida em qualquer site
   ( ) Traduzir o nome do site para o endereço do servidor
6. Você digita um endereço no navegador e aperta Enter. Qual é o primeiro passo para o site aparecer?

( ) O navegador descobre o endereço do servidor e faz o pedido
( ) O servidor envia a página antes de qualquer pedido
( ) O banco de dados abre o navegador

7. Qual é a função do banco de dados em um app de pedidos?
   ( ) Guardar de forma organizada informações como clientes e pedidos
   ( ) Desenhar os botões, as cores e o visual de cada tela do app
   ( ) Traduzir o nome do site para o endereço do servidor certo
8. Qual destas é uma forma de um app chegar ao celular das pessoas?
   ( ) Ser enviado pelo DNS para cada celular
   ( ) Ser guardado só dentro do banco de dados
   ( ) Ser publicado em uma loja, como a Play Store
9. Um site que você abre no navegador, sem instalar nada, é um exemplo de quê?
   ( ) Um banco de dados que ficou visível para todo mundo
   ( ) Um servidor instalado dentro do celular da pessoa
   ( ) Um app que roda pelo navegador, sem passar por uma loja
10. Em um app de lista de compras, onde deve ficar guardada a lista de cada usuário para ela aparecer em outro celular?
    ( ) No endereço (DNS) do site
    ( ) No servidor, dentro de um banco de dados
    ( ) Só na tela do celular, enquanto o app está aberto

Desafios

Anotações

Módulo 2. Lógica de programação

Variáveis, condições, laços e funções.

Aulas

- **Curso de Algoritmos e Lógica de Programação** (Curso em Vídeo): 17 aulas do Gustavo Guanabara, começa do zero e sem linguagem
  https://www.youtube.com/playlist?list=PLHz_AreHm4dmSj0MHol_aoNYCSGFqvfXV

Resumo

- Variável é um nome que guarda um valor.
- Condição (se/senão) escolhe um caminho conforme o valor.
- Laço repete um bloco até a condição acabar.

- Função agrupa passos com um nome, recebe valores e devolve um resultado.

Revisão

Responda no papel e confira o gabarito no fim da apostila.

1. Qual estrutura repete um bloco de código várias vezes?
   ( ) Uma variável
   ( ) Um comentário
   ( ) Um laço (for ou while)
2. O que uma variável guarda?
   ( ) O nome do arquivo
   ( ) Só números que nunca mudam
   ( ) Um valor que pode mudar durante o programa
3. Em "se a idade for maior que 17, então é adulto", qual estrutura foi usada?
   ( ) Condição (if)
   ( ) Função
   ( ) Laço (for)
4. Em programação, o que é um algoritmo?
   ( ) Um tipo de computador muito rápido e com muita memória
   ( ) Uma linguagem usada apenas para criar sites e páginas
   ( ) Uma sequência de passos para resolver um problema
5. O que acontece com o valor de x depois de: x = 5 e depois x = x + 1?
   ( ) x passa a valer 51
   ( ) x passa a valer 6
   ( ) x continua valendo 5
6. Em um programa, quando usamos o senão (else)?
   ( ) Para dizer o que fazer quando a condição do se é falsa
   ( ) Para dar um nome a um grupo de passos que se repetem
   ( ) Para repetir o bloco mais uma vez, até o fim do programa
7. Um laço é escrito para repetir 3 vezes. Quantas vezes o bloco dentro dele roda?
   ( ) 1 vez, e as outras 2 são ignoradas
   ( ) Infinitas vezes, sempre
   ( ) 3 vezes
8. O que pode acontecer se a condição de um laço nunca ficar falsa?
   ( ) O programa apaga as variáveis sozinho
   ( ) O laço roda sem parar (laço infinito)
   ( ) O laço roda uma única vez
9. Para que serve uma função?
   ( ) Impedir que o programa use condições e laços em seus passos
   ( ) Agrupar passos sob um nome para poder usá-los várias vezes
   ( ) Guardar um único valor que nunca muda durante o programa
10. Uma função recebe dois números e devolve a soma deles. Como se chamam os números que ela recebe?
    ( ) Laços (ou repetições)

( ) Parâmetros (ou argumentos)
( ) Condições (ou testes)

Desafios

1. Variáveis e texto
Crie duas variáveis: nome (um texto) e idade (um número). Depois mostre no parágrafo #saida a frase: Meu nome é ... e tenho ... anos.
Dica: Para escrever dentro do parágrafo: document.getElementById("saida").textContent = ... Para juntar textos e variáveis use o sinal de +.

2. Condições
Complete a função classificar(idade). Menos de 12 anos: "criança". De 12 a 17: "adolescente". 18 ou mais: "adulto".
Dica: Use if, else if e else, e a palavra return para devolver o resultado.

3. Laços
Complete a função somar(n), que devolve a soma de todos os números de 1 até n. Exemplo: somar(5) devolve 15.
Dica: Crie uma variável total começando em 0 e use um for de 1 até n, somando cada número ao total.

4. O maior de três
Complete a função maior(a, b, c), que devolve o maior dos três números. Tente fazer sem usar Math.max.
Dica: Comece assumindo que o maior é o a. Se b for maior que ele, troque. Depois faça o mesmo com c.

Anotações

Módulo 3. Terminal, Git e GitHub

Comandos básicos, commit, push e repositório.

Aulas

- **Curso de Terminal Linux - Bash** (Diolinux): 9 vídeos. É Linux, mas os comandos básicos (cd, ls, mkdir) valem para o terminal do dia a dia
  https://www.youtube.com/playlist?list=PLZsjaJhVZaxVoF7boPIXE_GGNI6iwt947

- **Curso de Git e GitHub** (Curso em Vídeo): Gustavo Guanabara, prático e sem depender de comandos no começo
  https://www.youtube.com/playlist?list=PLHz_AreHm4dm7ZULPAmadvNhH6vk9oNZA

Falta uma aula em português só sobre terminal no Windows. Se você usa Windows, o PowerShell tem comandos parecidos, mas ainda não achei uma aula boa em português.

Resumo

- O terminal executa comandos escritos, como cd, ls e mkdir.
- git add escolhe os arquivos, git commit grava no histórico e git push envia ao GitHub.
- Branch é uma linha paralela de trabalho. Merge junta de volta.
- Nunca envie segredos (arquivo .env) ao GitHub.

Revisão

Responda no papel e confira o gabarito no fim da apostila.

1. O que faz um commit?
   ( ) Grava no histórico uma foto do estado atual dos arquivos
   ( ) Envia o código para a internet
   ( ) Apaga os arquivos antigos
2. Qual comando envia os commits para o GitHub?
   ( ) git push
   ( ) git add
   ( ) git commit
3. Para que serve o GitHub?
   ( ) Guardar o código na internet e acompanhar o histórico de mudanças
   ( ) Escrever o código no lugar de você
   ( ) Só hospedar imagens
4. O que o comando cd faz no terminal?
   ( ) Cria uma pasta nova
   ( ) Lista os arquivos da pasta
   ( ) Muda para outra pasta
5. Qual comando mostra os arquivos e pastas do lugar onde você está?
   ( ) cd
   ( ) mkdir
   ( ) ls
6. Qual comando cria uma pasta chamada fotos?
   ( ) ls fotos -a
   ( ) rmdir fotos
   ( ) mkdir fotos
7. Qual é a ordem comum para salvar uma mudança e enviá-la ao GitHub?
   ( ) git commit, git push, git add
   ( ) git add, git commit, git push
   ( ) git push, git add, git commit

8. O que faz o comando git add?
    ( ) Escolhe quais arquivos entram no próximo commit
    ( ) Envia os arquivos escolhidos para o GitHub
    ( ) Apaga os arquivos da pasta do projeto no computador
9. O que é uma branch no Git?
    ( ) Uma cópia do projeto que o GitHub apaga sozinho
    ( ) Uma senha para proteger o repositório
    ( ) Uma linha paralela de trabalho, separada da principal
10. Depois de terminar o trabalho em uma branch, o que o merge faz?
    ( ) Envia o projeto para a loja de apps
    ( ) Apaga o histórico de commits
    ( ) Junta as mudanças de volta em outra branch
11. Por que você não deve enviar o arquivo .env para o GitHub?
    ( ) Ele deixa o repositório muito lento e ocupa espaço demais no GitHub
    ( ) Ele guarda segredos, como chaves e senhas, que outras pessoas poderiam ver
    ( ) O GitHub não aceita arquivos que tenham um ponto no começo do nome
12. O que é um repositório?
    ( ) Um projeto com seus arquivos e todo o histórico de mudanças
    ( ) Uma conta de e-mail usada para entrar no site do GitHub
    ( ) Um programa que roda comandos no terminal do computador
13. Qual destas situações mostra a vantagem do Git?
    ( ) O computador passa a funcionar sem precisar de internet
    ( ) Você errou e consegue voltar a uma versão anterior do projeto
    ( ) O código passa a se escrever sozinho, sem precisar de você

Desafios

Anotações

Trilha 1: Web básica

A base de todo site e de todo PWA.

Módulo 4. HTML e CSS

Estrutura e visual das páginas.

Aulas

- **Curso completo e atual de HTML5 e CSS3 - Módulo 1 de 5** (Curso em Vídeo): 41 aulas do Gustavo Guanabara. Os módulos 2 a 5 estão no mesmo canal
  https://www.youtube.com/playlist?list=PLHz_AreHm4dkZ9-atkcmcBaMZdmLHft8n

Resumo

- HTML é a estrutura: títulos, parágrafos, listas, links, imagens e formulários.
- CSS é o visual: cor, fonte, espaço (margin e padding) e borda.
- Flexbox e Grid organizam os elementos em linha ou em colunas.
- Use ids e classes para escolher quem recebe cada estilo.

Revisão

Responda no papel e confira o gabarito no fim da apostila.

1. Qual tag cria o título principal de uma página?
   ( ) <h1>
   ( ) <a>
   ( ) <p>
2. O que o CSS faz?
   ( ) Guarda os dados dos usuários
   ( ) Define o visual da página: cores, tamanhos e espaços
   ( ) Define o que a página faz ao clicar
3. Qual tag cria um link?
   ( )
   ( ) <a href="...">
   ( ) <img src="...">
4. Qual tag HTML cria um parágrafo de texto?
   ( ) <a>
   ( ) <h1>
   ( ) <p>
5. Qual tag mostra uma imagem na página?
   ( ) <pic>
   ( ) <image-link>
   ( ) <img>
6. Qual tag é usada para criar uma lista com marcadores (bolinhas)?
   ( ) sozinha, sem nada em volta

( ) <list>
( )

7. Em HTML, o que é a tag <form>?
   ( ) O bloco usado para criar formulários, como cadastro ou login
   ( ) Uma tag que cria títulos grandes, médios e pequenos na página
   ( ) O bloco que muda a cor do fundo e das letras da página
8. Qual regra CSS deixa os parágrafos com texto vermelho?
   ( ) p { background: <red>; }
   ( ) p { color: red; }
   ( ) red { p: color; }
9. No CSS, qual é a diferença entre margin e padding?
   ( ) Margin é o espaço por fora do elemento, padding é o espaço por dentro
   ( ) Margin é a cor do elemento e padding é o tamanho da fonte
   ( ) Margin vale só para textos e padding só vale para imagens
10. Como se escreve um seletor CSS que escolhe elementos com class="botao"?
    ( ) .botao { }
    ( ) #botao { }
    ( ) botao() { }
11. Qual propriedade CSS ativa o Flexbox em um contêiner?
    ( ) position: flex;
    ( ) flex: container;
    ( ) display: flex;
12. Para que serve o Grid no CSS?
    ( ) Trocar a fonte e as cores de todo o site de uma só vez
    ( ) Enviar os dados de um formulário para o servidor do site
    ( ) Organizar elementos em linhas e colunas, como uma tabela de layout
13. Qual é a diferença entre id e class?
    ( ) O id pode se repetir em todos os elementos da página, mas a class não
    ( ) O id deve ser único na página, e a class pode ser usada em vários elementos
    ( ) A class só funciona em imagens, e o id só funciona em textos

Desafios

1. Estrutura de uma página
Acrescente: dois parágrafos, uma lista com 3 itens e um link para qualquer endereço.
Dica: Parágrafo: <p>. Lista: <ul> com vários <li>. Link: texto.

2. Cores e cantos
No CSS: deixe o título azul e centralizado. Dê ao botão fundo verde, texto branco e cantos arredondados.
Dica: Propriedades: color, text-align, background, border-radius. Os nomes das cores podem ser em inglês: blue, green, white.

3. Flexbox
Faça as 3 caixas ficarem lado a lado usando flexbox em #grupo, com 12px de espaço entre elas (gap).
Dica: No pai (#grupo): display: flex; e gap: 12px;

4. Formulário de login
Crie um formulário com um campo de e-mail (com label), um campo de senha (com label) e um botão de enviar. Ligue cada label ao seu campo pelo id.
Dica: <label for="email">E-mail</label> e <input id="email" type="email">. O for da label precisa ser igual ao id do campo.

Anotações

Módulo 5. JavaScript do básico ao avançado

A linguagem que faz a página funcionar.

Aulas

- **Curso Grátis de JavaScript e ECMAScript para Iniciantes** (Curso em Vídeo): 33 aulas do Gustavo Guanabara, começa do zero
  https://www.youtube.com/playlist?list=PLHz_AreHm4dlsK3Nr9GVvXCbpQyHQl1o1

Resumo

- O DOM é a página vista como objetos que o JavaScript pode mudar.
- Eventos (click, input) fazem o código rodar quando a pessoa age.
- Arrays e objetos guardam listas e dados. map, filter e find ajudam a trabalhar com eles.
- fetch busca dados na internet e async/await espera a resposta.

Revisão

Responda no papel e confira o gabarito no fim da apostila.

1. Qual método busca um elemento da página pelo seu id?
   ( ) document.getElementById
   ( ) window.findText
   ( ) document.makeElement
2. O que significa === em JavaScript?
   ( ) Compara se os dois valores são iguais e do mesmo tipo
   ( ) Atribui um valor à variável

( ) Soma dois valores

3. O que o método map faz com uma lista?
   ( ) Cria uma nova lista aplicando uma função em cada item
   ( ) Apaga os itens repetidos
   ( ) Ordena do maior para o menor
4. Qual comando mostra uma mensagem no console do navegador?
   ( ) console.show("Oi")
   ( ) terminal.log("Oi")
   ( ) console.log("Oi")
5. Qual linha cria uma variável que pode mudar de valor depois?
   ( ) let idade = 20;
   ( ) idade := 20;
   ( ) const idade = 20;
6. O que acontece se você tentar fazer isto: const nome = "Ana"; nome = "Bia";
   ( ) O JavaScript guarda os dois nomes juntos em uma lista
   ( ) O nome passa a ser Bia sem nenhum problema ou aviso
   ( ) Dá erro, porque uma const não pode receber outro valor
7. Qual é o resultado de "5" + 3 em JavaScript?
   ( ) "53"
   ( ) Dá erro
   ( ) 8
8. Qual é o resultado de 10 % 3?
   ( ) 0,3
   ( ) 1
   ( ) 3
9. O que significa DOM?
   ( ) A página vista como objetos que o JavaScript pode ler e mudar
   ( ) Um banco de dados guardado dentro do navegador da pessoa
   ( ) Um tipo de variável usada para guardar números no JavaScript
10. Qual código muda o texto de um elemento guardado na variável titulo?
    ( ) titulo.textContent = "Novo";
    ( ) titulo.changeText("Novo");
    ( ) titulo = text.set("Novo");
11. Qual método faz uma função rodar quando alguém clica em um botão?
    ( ) addEventListener
    ( ) querySelector
    ( ) console.log
12. Em arr = [10, 20, 30], qual é o valor de arr[0]?
    ( ) 30
    ( ) 20
    ( ) 10
13. Qual método adiciona um item no final de um array?

( ) filter
( ) pop
( ) push

14. Qual método devolve uma nova lista só com os itens que passam em um teste?
    ( ) reduce
    ( ) concat
    ( ) filter
15. Qual método devolve o primeiro item de uma lista que passa em um teste?
    ( ) filter
    ( ) find
    ( ) push
16. Dado pessoa = { nome: "Ana", idade: 30 }, como pegar a idade?
    ( ) idade.pessoa
    ( ) pessoa(idade)
    ( ) pessoa.idade
17. O que faz um if (idade >= 18) { ... } em JavaScript?
    ( ) Repete o bloco de código 18 vezes seguidas
    ( ) Executa o bloco só se a idade for 18 ou mais
    ( ) Cria uma variável chamada idade com o valor 18
18. Para que serve o fetch?
    ( ) Buscar dados de um endereço na internet
    ( ) Repetir um bloco de código
    ( ) Mudar a cor de um elemento da página
19. Para que serve o await dentro de uma função async?
    ( ) Esperar a resposta chegar antes de seguir para a próxima linha
    ( ) Apagar da memória o resultado da resposta que chegou
    ( ) Fazer o código inteiro rodar mais rápido que o normal

Desafios

1. Clique e troque o texto
Quando clicar no botão, troque o texto do parágrafo #msg para Olá!
Dica: Pegue o botão com getElementById e use onclick = function () { ... } ou addEventListener("click", ...).

2. Contador
O botão + soma 1 e o botão - tira 1 do número em #valor. O número não pode ficar abaixo de 0.
Dica: Guarde o número em uma variável. A cada clique, mude a variável e depois escreva o valor no #valor. Antes de tirar 1, confira se é maior que 0.

3. Lista de tarefas
Ao clicar em Adicionar, crie um item (li) na lista com o texto do campo e limpe o campo. Se o campo estiver vazio, não adicione nada.
Dica: Leia o texto com campo.value. Crie o item com document.createElement("li") e coloque na lista com appendChild.

4. Listas com map e filter
Complete as funções. dobrar(lista) devolve uma NOVA lista com cada número multiplicado por 2 (use map). pares(lista) devolve só os números pares (use filter).
*Dica: lista.map(function (x) { return x * 2; }) e lista.filter(function (x) { return x % 2 === 0; }). Não esqueça do return.*

5. Evento de digitação
Enquanto a pessoa digita no campo (evento input), mostre em #aviso: Senha forte se tiver 8 ou mais caracteres, senão Senha fraca.
Dica: Use campo.addEventListener("input", function () { ... }) e campo.value.length para saber quantos caracteres tem.

Anotações

Módulo 6. PWA e design responsivo

Site que se comporta como app e serve em qualquer tela.

Aulas

- **Curso completo de flexbox, do básico ao avançado** (Dev. Odair Michael): 15 vídeos sobre flexbox, em português
  https://www.youtube.com/playlist?list=PLa3uqHCcweTggaAGaaImnJEpONkm-7uI9
- **CSS Grid Curso Completo** (dpw): 17 vídeos sobre CSS Grid para layouts modernos
  https://www.youtube.com/playlist?list=PLYgzkrmJnLwpeeGgdADYq3cE2yUwLLTOv
- **Publicando um PWA na Google Play** (Vuejs Expert): 3 vídeos sobre publicar um PWA na Play Store com o PWABuilder
  https://www.youtube.com/playlist?list=PLBjvYfV_TvwJDPHU7VEANIa5vQz45xOJK

Ainda não achei uma playlist em português conferida sobre manifest e service worker. O blog do Fellyph Cintra (blog.fellyph.com.br) tem um curso de PWA em português. Procure pelo curso PWA dele no YouTube.

Resumo

- Design responsivo adapta a página a telas de tamanhos diferentes, com media queries, flexbox e grid.
- PWA é um site instalável, com manifest e service worker.
- O service worker guarda arquivos em cache para abrir sem internet.
- Teste sempre em um celular de verdade.

Revisão

Responda no papel e confira o gabarito no fim da apostila.

1. O que faz uma regra @media (max-width: 600px)?
   ( ) Aplica estilos só quando a tela tem até 600 px de largura
   ( ) Muda o idioma da página
   ( ) Esconde o site em telas grandes para sempre
2. Qual arquivo diz o nome, o ícone e as cores de um PWA?
   ( ) O index.css
   ( ) O package.json
   ( ) O manifest (manifest.json)
3. Para que serve o service worker?
   ( ) Comprimir as fotos da câmera
   ( ) Guardar arquivos em cache para o app abrir mesmo sem internet
   ( ) Criar a conta dos usuários
4. O que a linha display: flex; faz em um elemento do CSS?
   ( ) Transforma o elemento em uma tabela fixa, com várias linhas e colunas.
   ( ) Esconde o elemento inteiro e mostra só o primeiro filho dele na tela.
   ( ) Põe os filhos lado a lado e deixa organizá-los com regras como justify-content.
5. Em um container flex em linha, para que serve justify-content: center?
   ( ) Centralizar o texto de cada item dentro da própria caixa.
   ( ) Deixar todos os itens com o mesmo tamanho de fonte da linha.
   ( ) Centralizar os itens no sentido da linha, horizontalmente.
6. O que o CSS grid-template-columns: 1fr 1fr; cria?
   ( ) Duas linhas com a mesma altura.
   ( ) Uma única coluna com o dobro da largura.
   ( ) Duas colunas com a mesma largura.
7. Para que serve a tag <meta name="viewport" content="width=device-width, initial-scale=1">?
   ( ) Aumentar a velocidade de carregamento da página nos celulares com internet lenta.
   ( ) Fazer a página usar a largura real da tela do celular, para o design responsivo funcionar.
   ( ) Impedir que a página seja aberta em computadores e em tablets grandes.
8. Por que o service worker só funciona em endereços seguros (HTTPS)?
   ( ) Porque o HTTPS deixa o código JavaScript mais rápido para rodar no navegador.
   ( ) Porque ele pode interceptar e guardar o que o site carrega, então exige conexão segura.
   ( ) Porque o HTTP comum não aceita arquivos .js dentro de páginas e aplicativos.
9. Qual é a função do PWABuilder, mostrado nas aulas de publicação?
   ( ) Escrever o CSS responsivo da página no lugar da pessoa, de forma automática.

( ) Hospedar o site de graça para sempre, sem limite de acessos.

( ) Empacotar um PWA para poder publicá-lo em lojas, como a Google Play.

10. O que significa a ideia de "mobile first" no design responsivo?

( ) Começar a estilizar pensando no celular e depois ajustar para telas maiores.

( ) Fazer o site funcionar só no celular e bloquear o acesso por computadores.

( ) Instalar um app no celular antes de abrir o site pela primeira vez.

Desafios

1. Media query

Crie uma regra @media (max-width: 600px) que mude a cor de fundo do body em telas pequenas.

Dica: @media (max-width: 600px) { body { background: lightyellow; } }

2. Grade de cards

Use CSS Grid em #cards para fazer 3 colunas iguais, com 10px de espaço (gap) entre elas.

Dica: display: grid; grid-template-columns: repeat(3, 1fr); gap: 10px;

Anotações

Trilha 2: TypeScript

JavaScript com tipos, usado nos seus apps em Expo.

Módulo 7. TypeScript básico

Tipos, funções e objetos.

Aulas

- **Curso de Typescript** (CFBCursos): 37 aulas, do zero
  https://www.youtube.com/playlist?list=PLx4x_zx8csUhtPMrkiGvFJVE5LX8Qat5s
- **Curso de Typescript Completo #P01: preparando o ambiente**: Série numerada, começa no ambiente
  https://www.youtube.com/watch?v=vcpLrIzIKjI
- **Aprenda TypeScript em 1 hora, na prática**: Visão rápida dos tipos básicos
  https://www.youtube.com/watch?v=lCemyQeSCV8
- **Mini-curso de TypeScript: classes (aula 9)**: Série numerada
  https://www.youtube.com/watch?v=rxpycPZnT6I
- **Curso de TypeScript: interfaces (aula 4)**: Série numerada
  https://www.youtube.com/watch?v=SiZnlpIGWhc

Resumo

- O tipo diz que valor uma variável aceita: string, number, boolean.
- O TypeScript avisa os erros antes de rodar.
- Tipe os parâmetros e o retorno das funções.
- Listas: number[]. Objetos: { nome: string; idade: number }.

Revisão

Responda no papel e confira o gabarito no fim da apostila.

1. O que o TypeScript adiciona ao JavaScript?
   ( ) Uma nova forma de desenhar a tela
   ( ) Um banco de dados embutido
   ( ) Tipos, que avisam erros antes de rodar
2. Como se declara uma variável de texto em TypeScript?
   ( ) let nome: string = "Ana"
   ( ) let nome = string "Ana"
   ( ) string nome := "Ana"
3. O que acontece ao passar um número para uma função que espera texto (string)?
   ( ) Converte sozinho sem avisar
   ( ) O navegador apaga a função
   ( ) O TypeScript mostra um erro antes de você rodar
4. Qual é o tipo certo para a variável em let idade = 30; quando você escreve o tipo de forma explícita?

( ) let idade: boolean = 30;
( ) let idade: string = 30;
( ) let idade: number = 30;

5. O que acontece com let ativo: boolean = "sim";?
   ( ) O TypeScript mostra um erro, porque boolean só aceita true ou false.
   ( ) Funciona normalmente e o valor "sim" vira true de forma automática.
   ( ) O texto "sim" é convertido sozinho para número quando o código roda.
6. Qual é a forma de dizer que uma variável é uma lista de números?
   ( ) numbers()
   ( ) number[]
   ( ) list<number>
7. Em function dobro(n: number): number, o que o segundo ": number" (depois dos parênteses) indica?
   ( ) O tipo do valor que a função devolve.
   ( ) O tipo do primeiro parâmetro.
   ( ) Quantas vezes a função pode ser chamada.
8. O navegador consegue executar um arquivo .ts diretamente?
   ( ) Sim, mas só funciona quando o computador está sem a internet ligada.
   ( ) Sim, todo navegador moderno entende arquivos .ts sem nenhuma conversão.
   ( ) Não, o TypeScript precisa ser convertido para JavaScript antes.
9. Qual comando costuma ser usado para converter um arquivo TypeScript em JavaScript?
   ( ) js build
   ( ) npm play
   ( ) tsc
10. Dado let nome = "Ana"; sem escrever o tipo, o que o TypeScript entende?
    ( ) Que nome é uma string, pelo valor inicial.
    ( ) Que nome pode guardar qualquer valor depois.
    ( ) Que nome é um number, pelo valor inicial.
11. Dado let pessoa: { nome: string; idade: number } = { nome: "Ana" };, o que acontece?
    ( ) Dá erro, porque falta o campo idade.
    ( ) Funciona, e o campo idade fica escondido.
    ( ) Funciona e idade vira 0.
12. Qual destas linhas guarda um valor de texto com tipo explícito?
    ( ) let cidade: string = "Recife";
    ( ) let cidade: text = "Recife";
    ( ) string cidade = "Recife";
13. De forma geral, o que é uma classe (class) em TypeScript?
    ( ) Um molde para criar objetos com os mesmos campos e funções.
    ( ) Um arquivo que guarda só o CSS e as cores do site.
    ( ) Uma lista de números que tem o tipo fixo e não muda.

Desafios

1. Função com tipos
No TypeScript Playground, escreva a função somar(a, b) com os tipos corretos (números). Depois chame somar("1", 2) e veja o erro que o TypeScript mostra.
Dica: function somar(a: number, b: number): number { ... }
https://www.typescriptlang.org/play

2. Tipo de objeto
Crie um tipo Usuario com nome (texto) e idade (número). Crie uma variável desse tipo. Depois apague a idade e veja o erro.
Dica: type Usuario = { nome: string; idade: number };
https://www.typescriptlang.org/play

Anotações

Módulo 8. TypeScript intermediário

Interfaces, generics e módulos.

Aulas

- **Interfaces, Types e Generics explicados de forma simples**: Tudo numa aula prática
  https://www.youtube.com/watch?v=XwtKuAV_fHY
- **Generics em TypeScript (#P29)**: Aula da série completa
  https://www.youtube.com/watch?v=zfLQz2HA4s4
- **Módulos em TypeScript (aula 35, Node)**: Única aula sobre módulos que apareceu
  https://www.youtube.com/watch?v=bPTONsV6q0U
- **Interfaces com TypeScript na prática**
  https://www.youtube.com/watch?v=RkbYUtuRcco
- **Utilizando Generics com TypeScript**
  https://www.youtube.com/watch?v=iGn5iVBk3c8
- **O que é Generics no TypeScript e como usar**
  https://www.youtube.com/watch?v=NsQgpBLmdUY

Resumo

- interface e type descrevem o formato de objetos.
- Generics (<T>) reaproveitam código com tipos diferentes.
- Campos opcionais usam ? e a união de tipos usa |.

- import e export dividem o código em arquivos.

Revisão

Responda no papel e confira o gabarito no fim da apostila.

1. Para que serve uma interface?
   ( ) Descrever o formato (campos e tipos) de um objeto
   ( ) Executar o código mais rápido
   ( ) Conectar ao banco de dados
2. O que são generics, como <T>?
   ( ) Comentários especiais
   ( ) Nomes de arquivos
   ( ) Tipos que ficam abertos para serem definidos no uso, reaproveitando o código
3. O que o export faz em um arquivo?
   ( ) Salva o arquivo na nuvem
   ( ) Deixa a função ou o tipo disponível para outros arquivos importarem
   ( ) Apaga a variável depois do uso
4. Em interface Usuario { nome: string; idade?: number }, o que o ? depois de idade significa?
   ( ) Que idade pode ser qualquer tipo.
   ( ) Que idade é uma pergunta feita ao usuário.
   ( ) Que o campo idade é opcional.
5. O que significa type Id = string | number;?
   ( ) Id pode ser um texto ou um número.
   ( ) Id é uma lista com um texto e um número.
   ( ) Id precisa ser um texto e um número ao mesmo tempo.
6. Em function primeiro<T>(lista: T[]): T, o que é o T?
   ( ) O nome obrigatório que todo parâmetro de função precisa ter.
   ( ) Um tipo que será definido na hora de usar a função.
   ( ) Uma variável chamada T que sempre guarda um valor fixo.
7. Qual é uma vantagem de usar generics em vez de any?
   ( ) O arquivo final fica bem menor quando vai ao navegador.
   ( ) O TypeScript continua sabendo o tipo do resultado.
   ( ) O código fica sem nenhum tipo e roda mais livre.
8. O que faz a linha import { soma } from "./util";?
   ( ) Apaga a função soma de dentro do arquivo util da pasta.
   ( ) Traz a função soma do arquivo util para o arquivo atual.
   ( ) Cria um arquivo novo chamado util na pasta atual do projeto.
9. Para outro arquivo poder importar uma função, o que é preciso fazer no arquivo onde ela foi escrita?
   ( ) Colocar export antes dela.
   ( ) Renomear o arquivo para .json.
   ( ) Escrever a função duas vezes.
10. string[] e Array<string> são parecidos ou diferentes?
    ( ) Array<string> aceita números também, além dos textos da lista.

( ) string[] é uma lista de textos e Array<string> é uma lista de números.
( ) São duas formas de escrever o mesmo tipo: uma lista de textos.

11. Uma interface exige nome: string e você cria um objeto sem o campo nome. O que ocorre?
    ( ) O TypeScript avisa o erro antes de rodar.
    ( ) O código roda e nome vira vazio.
    ( ) O TypeScript cria o campo nome sozinho.
12. O que o modificador readonly faz em um campo, como readonly id: number?
    ( ) Faz o campo aparecer só na tela e nunca ser salvo no arquivo.
    ( ) Esconde o campo para que outros arquivos não consigam enxergá-lo.
    ( ) Impede que o valor seja alterado depois que o objeto foi criado.
13. Em class Carro implements Veiculo, o que a palavra implements pede?
    ( ) Que Veiculo seja apagada do projeto assim que Carro for criada.
    ( ) Que Carro copie todo o CSS que está escrito dentro de Veiculo.
    ( ) Que a classe Carro tenha tudo o que a interface Veiculo descreve.

Desafios

1. Interface e genérico
Crie a interface Produto com id (número), nome (texto), preco (número) e estoque opcional (número). Depois crie a função genérica primeiro<T>(lista: T[]) que devolve o primeiro item da lista, ou undefined se a lista estiver vazia.
Dica: Campo opcional usa ? depois do nome. A função genérica começa assim: function primeiro<T>(lista: T[]): T | undefined
https://www.typescriptlang.org/play

Anotações

Módulo 9. TypeScript avançado

Tipos utilitários e boas práticas.

Aulas

- **Utility Types: Partial, Omit, Pick, Record, keyof**: Idioma a conferir
  https://www.youtube.com/watch?v=E1C00MZXhQ4
- **Utility Types: Partial, Omit e Pick**: Idioma a conferir
  https://www.youtube.com/watch?v=juvaBPMGWD0
- **Utility Types: Pick, Omit, Exclude, Extract (#07)**: Idioma a conferir
  https://www.youtube.com/watch?v=jtNhCPzsTyk

Não achei aula confiável em português sobre tipos utilitários, mapped types e conditional types.

Resumo

- Tipos utilitários: Partial, Required, Pick, Omit e Record.
- Evite any. Prefira unknown e confira o tipo com um if.
- Type guards e narrowing deixam o tipo mais preciso dentro de um if.
- Ligue o strict no tsconfig para pegar mais erros.

Revisão

Responda no papel e confira o gabarito no fim da apostila.

1. O que faz Partial<T>?
   ( ) Apaga o tipo T
   ( ) Torna todos os campos do tipo T opcionais
   ( ) Converte T em texto
2. Por que evitar o tipo any?
   ( ) Porque desliga a checagem de tipos e esconde erros
   ( ) Porque só funciona no celular
   ( ) Porque deixa o código lento
3. O que significa string | number?
   ( ) O valor pode ser texto ou número
   ( ) O valor é a soma dos dois tipos
   ( ) O valor é texto e número ao mesmo tempo
4. O que faz Pick<Usuario, "nome"> ?
   ( ) Apaga o campo nome de Usuario.
   ( ) Torna todos os campos de Usuario opcionais.
   ( ) Cria um tipo só com o campo nome de Usuario.
5. O que faz Omit<Usuario, "senha"> ?
   ( ) Cria um tipo igual a Usuario, mas sem o campo senha.
   ( ) Cria um tipo com apenas o campo senha de Usuario.
   ( ) Muda o campo senha para número, mantendo o resto.
6. O que descreve o tipo Record<string, number>?
   ( ) Um objeto cujas chaves são textos e cujos valores são números.
   ( ) Uma lista que guarda só números, sem chaves e sem nenhum texto.
   ( ) Um texto único que guarda dentro de si um número qualquer.
7. O que faz Required<T> ?
   ( ) Apaga de T todos os campos que eram opcionais, mantendo os demais.
   ( ) Torna opcionais todos os campos de T, inclusive os que eram obrigatórios.
   ( ) Torna obrigatórios todos os campos de T, inclusive os opcionais.
8. Dado interface Pessoa { nome: string; idade: number }, o que keyof Pessoa representa?
   ( ) Os nomes dos campos: "nome" ou "idade".
   ( ) Os valores dos campos.
   ( ) O número de campos da interface.
9. O que o TypeScript exige antes de você usar um valor do tipo unknown?

( ) Nada, funciona igual ao any.
( ) Conferir o tipo dele, por exemplo com um if.
( ) Converter o arquivo para JavaScript.

10. No código if (typeof x === "string") { ... }, o que acontece dentro das chaves?
    ( ) x vira number de forma automática, sem checagem.
    ( ) x deixa de existir e some do código.
    ( ) O TypeScript passa a tratar x como string.
11. Para que serve ligar "strict": true no tsconfig.json?
    ( ) Ativar verificações mais rígidas, que pegam mais erros.
    ( ) Deixar o código compilado mais rápido na hora de rodar.
    ( ) Permitir usar any em todo lugar sem receber nenhum aviso.
12. Qual é o resultado de Exclude<"a" | "b" | "c", "a"> ?
    ( ) "b" | "c"
    ( ) "a" | "b" | "c"
    ( ) "a"
13. O que faz Readonly<T> ?
    ( ) Deixa todos os campos de T só para leitura.
    ( ) Apaga todos os campos de T e deixa o tipo vazio.
    ( ) Transforma T em texto para que possa ser mostrado.

Desafios

Anotações

Trilha 3: Apps nativos

Do React ao app instalado no celular e publicado na loja.

Módulo 10. React e React Native

Componentes, estado e telas no celular.

Aulas

- **Curso de React do zero com TypeScript** (Lucas Souza Dev): 52 vídeos
  https://www.youtube.com/playlist?list=PL29TaWXah3iYzP5FGywXezXm4ZvBibGSk
- **Curso de React com TypeScript** (Lucas Souza Dev): 31 vídeos
  https://www.youtube.com/playlist?list=PL29TaWXah3iZktD5o1IHbc7JDqG_80iOm
- **React do zero: a aula que vai te tornar dev front end**
  https://www.youtube.com/watch?v=hHcaVgoLLQM
- **Curso React completo para iniciantes 2025**
  https://www.youtube.com/watch?v=-GPQNQzeO7M
- **Curso de React completo (Origamid)**: Canal provável: Origamid
  https://www.youtube.com/watch?v=pKP9-SeOLqg
- **Tutorial iniciante de React Native** (Mundo Tech Software School): 16 vídeos
  https://www.youtube.com/playlist?list=PLUbb2i4BuuzBrD7yLahzYlM5WPCDv_vGl
- **Curso de React Native para iniciantes** (Icode Mobile): 20 vídeos
  https://www.youtube.com/playlist?list=PLQlQ8-V7oQHyZGi4bA5qeR3NbELDTym_9
- **Curso React Native** (Desenvolvendo do básico): 21 vídeos
  https://www.youtube.com/playlist?list=PLedVhPP7RyiL0dwk5T2D5_x_lV5KdGP50
- **Curso React Native + TypeScript: app Wallet**: Playlist de projeto
  https://www.youtube.com/playlist?list=PLRGv4LGMjiB9nBoDXNR19CiOft3fW0ald
- **Guia para aprender React Native em 2025**: Roteiro de estudo
  https://www.youtube.com/watch?v=20WH5xB54Hc

Resumo

- Componente é uma função que devolve uma parte da tela (JSX).
- Props passam dados de fora. State guarda dados que mudam (useState).
- useEffect roda código depois de desenhar a tela, como buscar dados.
- React Native usa View, Text e Pressable em vez de div, p e button.

Revisão

Responda no papel e confira o gabarito no fim da apostila.

1. O que é um componente no React?
   ( ) Um servidor
   ( ) Um pedaço reaproveitável da tela, escrito como função
   ( ) Um arquivo de imagem
2. Para que serve o useState?

( ) Buscar dados no banco
( ) Instalar bibliotecas
( ) Guardar um valor que, ao mudar, atualiza a tela

3. Qual a diferença entre props e state?
   ( ) Props vêm de fora do componente; state é guardado dentro dele
   ( ) Props são mais lentas que state
   ( ) São o mesmo nome para a mesma coisa
4. Qual extensão de arquivo costuma ser usada em componentes React escritos com TypeScript?
   ( ) .tsx
   ( ) .html
   ( ) .txt
5. O que é o JSX?
   ( ) Um programa que instala o React e os pacotes no computador
   ( ) Uma sintaxe parecida com HTML escrita dentro do código JavaScript
   ( ) Um banco de dados que guarda as telas do app no navegador
6. Qual destes é um nome válido para um componente React?
   ( ) Botao
   ( ) meu-botao
   ( ) botao
7. Em React, como se usa uma variável dentro do JSX?
   ( ) Colocando entre colchetes, como [nome]
   ( ) Colocando entre chaves, como {nome}
   ( ) Colocando entre aspas, como "nome"
8. Veja o código: const [n, setN] = useState(0); O que o setN faz?
   ( ) Apaga a variável n para sempre
   ( ) Guarda o valor de n no celular mesmo depois de fechar o app
   ( ) Troca o valor de n e faz a tela ser desenhada de novo
9. O que acontece se você mudar o state escrevendo direto n = 5, sem usar setN?
   ( ) A tela não é atualizada, porque o React não percebe a mudança
   ( ) A tela é atualizada na hora, pois o React vigia a variável
   ( ) O valor é salvo no celular e o app reinicia sozinho
10. Para mostrar uma lista de itens na tela, qual método de array costuma ser usado no JSX?
    ( ) push
    ( ) join
    ( ) map
11. Por que cada item de uma lista no React precisa de uma key?
    ( ) Para deixar a lista mais bonita e com cores diferentes
    ( ) Para o React identificar qual item mudou, entrou ou saiu
    ( ) Para proteger cada item da lista com uma senha própria
12. Qual é o jeito certo de um componente filho receber um dado do componente pai?
    ( ) Por props
    ( ) Apagando o componente pai
    ( ) Por um e-mail

13. Em <Botao texto="Salvar" />, o que é texto?
    ( ) Um erro de escrita no código
    ( ) Uma prop do componente Botao
    ( ) Um state do componente Botao
14. Quando um componente mostra um 'Carregando...' até os dados chegarem, qual ideia está sendo usada?
    ( ) Renderização condicional: a tela muda conforme o estado
    ( ) Criptografia, que embaralha os dados guardados
    ( ) Compilação do app, que gera o arquivo APK final
15. O que significa o array vazio em useEffect(() => { ... }, [])?
    ( ) O efeito nunca roda, pois a lista está vazia
    ( ) O efeito roda só uma vez, quando o componente aparece
    ( ) O efeito roda a cada segundo, de forma contínua
16. Qual destes é um uso comum do useEffect?
    ( ) Dar um nome novo para cada componente do app
    ( ) Escolher a cor e o tamanho do texto na tela
    ( ) Buscar dados de uma API quando a tela abre
17. No React Native, qual componente é usado para mostrar um texto na tela?
    ( ) Text
    ( ) span
    ( ) div
18. No React Native, qual componente serve de caixa para organizar outros, como a div da web?
    ( ) View
    ( ) Page
    ( ) Box
19. Em um app React Native, como costumamos definir cores e tamanhos dos componentes?
    ( ) Com objetos de estilo, por exemplo StyleSheet.create
    ( ) Com tags <style> escritas dentro do JSX da tela
    ( ) Com arquivos .css ligados por link no topo da tela

Desafios

Anotações

Módulo 11. Expo: telas, navegação e estado

Expo Router, menus e navegação entre telas.

Aulas

- **Guia completo Expo Router do zero**: Parece estar em português
  https://www.youtube.com/watch?v=UP8wNrFLgtU
- **Navegando com Expo Router (parte 2)**: Curso de React Native 2024
  https://www.youtube.com/watch?v=bOu-B8PGEoQ
- **Navegação baseada em diretórios com Expo Router**
  https://www.youtube.com/watch?v=kJuWCjTIulM
- **Menu lateral passo a passo com Expo Router**: Drawer
  https://www.youtube.com/watch?v=-NkDdPTwOWU
- **Como usar o React Navigation no Expo**
  https://www.youtube.com/watch?v=KOEmcoH-6NE

Não achei playlist completa de Expo em português com estado global (Context, Zustand).

Resumo

- O Expo facilita criar, rodar e publicar apps React Native.
- No Expo Router, cada arquivo dentro de app vira uma tela.
- Abas e pilha de telas são montadas em layouts (_layout.tsx).
- Estado compartilhado pode ficar em um contexto ou em uma biblioteca simples.

Revisão

Responda no papel e confira o gabarito no fim da apostila.

1. O que é o Expo?
   ( ) Um banco de dados
   ( ) Uma loja de apps
   ( ) Um conjunto de ferramentas que facilita criar apps React Native
2. No Expo Router, como se cria uma nova tela?
   ( ) Criando um arquivo dentro da pasta app
   ( ) Editando o AndroidManifest à mão
   ( ) Instalando um plugin pago
3. Para que serve o Expo Go?
   ( ) Abrir e testar seu app no celular durante o desenvolvimento
   ( ) Publicar na Play Store
   ( ) Criar o ícone do app
4. O que o comando npx create-expo-app faz?
   ( ) Cria um projeto novo de app Expo já pronto para uso
   ( ) Instala o Android Studio e o emulador no PC
   ( ) Publica o app pronto na Play Store para todos
5. Qual comando costuma ser usado para iniciar o servidor de desenvolvimento do Expo?
   ( ) npx expo start
   ( ) npm sleep
   ( ) npx expo delete
6. Para que serve o QR code que aparece ao iniciar o Expo?

( ) Pagar a conta de desenvolvedor
( ) Trocar a senha do projeto
( ) Abrir o app no celular pelo Expo Go

7. Como o Expo Router decide qual arquivo é a tela inicial de uma pasta?
   ( ) O arquivo index.tsx
   ( ) O arquivo com o menor tamanho
   ( ) O arquivo mais antigo
8. Se você cria o arquivo app/sobre.tsx, que endereço de tela ele ganha?
   ( ) /app/sobre.tsx
   ( ) /sobre
   ( ) /index/sobre
9. Para que serve o arquivo _layout.tsx dentro de uma pasta do Expo Router?
   ( ) Compila o código e gera o arquivo APK para instalar
   ( ) Guarda as senhas e as chaves secretas de cada usuário
   ( ) Define como as telas da pasta se organizam, em pilha ou abas
10. Qual componente do Expo Router cria um link para ir a outra tela?
    ( ) Anchor
    ( ) Link
    ( ) Goto
11. Qual comando muda de tela por código, sem o usuário tocar em um Link?
    ( ) screen.go('/perfil')
    ( ) router.push('/perfil')
    ( ) window.open('/perfil')
12. Em uma navegação em pilha, o que acontece ao apertar o botão de voltar?
    ( ) O app é desinstalado
    ( ) Todas as telas são apagadas do projeto
    ( ) A tela de cima sai e a anterior aparece de novo
13. Qual nome de arquivo cria uma rota com valor variável, como o id de um produto?
    ( ) (id).tsx
    ( ) [id].tsx
    ( ) id.tsx
14. O que mostra um menu de abas fixo na parte de baixo do app?
    ( ) Um arquivo de imagem
    ( ) O Expo Go
    ( ) Um layout de abas (Tabs)
15. Como o Drawer se parece para o usuário?
    ( ) Um menu lateral que desliza da borda da tela
    ( ) Um aviso de erro que aparece fixo no topo da tela
    ( ) Um teclado que abre sozinho quando o app inicia
16. Qual é o papel do Context no React?
    ( ) Compartilhar um dado entre várias telas sem props em cada nível
    ( ) Enviar notificações de lembrete para o usuário todo dia
    ( ) Guardar fotos e vídeos na memória do celular do usuário

17. Quando é mais indicado guardar um dado em estado compartilhado (contexto) em vez de um useState local?
    ( ) Quando várias telas precisam ler ou mudar o mesmo dado
    ( ) Quando o dado só é usado dentro de um único botão
    ( ) Quando o app não tem nenhuma tela para mostrar
18. Para que serve a pasta assets em um projeto Expo?
    ( ) Guardar as senhas e os dados pessoais dos usuários
    ( ) Guardar imagens, ícones e fontes usados pelo app
    ( ) Guardar o código das telas e a lógica de cada uma
19. O que faz o arquivo app.json de um projeto Expo?
    ( ) Guarda o banco de dados com todos os registros do app
    ( ) Desenha as telas e os botões que o usuário vê no app
    ( ) Guarda as configurações, como nome, ícone e identificador

Desafios

Anotações

Módulo 12. Recursos do celular

Notificações, armazenamento, câmera e voz.

Aulas

- **React Native (Expo) para iniciantes: seu primeiro app** (Coffstack): Nov/2025, cerca de 40 min
  https://www.youtube.com/watch?v=FULK2o5TRiM
- **Curso React Native** (Desenvolvendo do básico): 21 vídeos
  https://www.youtube.com/playlist?list=PLedVhPP7RyiL0dwk5T2D5_x_lV5KdGP50
- **Compreendendo o sistema de notificações do Expo (aula 16)**
  https://www.youtube.com/watch?v=N0sUzwhOMUQ
- **Como implementar notificações com React Native**
  https://www.youtube.com/watch?v=HleOwBhfaac
- **Notificações locais em React Native com Expo**
  https://www.youtube.com/watch?v=Jz9Gqxzun5w
- **Introdução a notificações push (#28)**: Série Meu Primeiro App
  https://www.youtube.com/watch?v=PFwtetE4Tjs
- **Push notifications em produção (#34)**: Mesma série
  https://www.youtube.com/watch?v=Ln4iKbgJtag

- **Como colocar notificações com Expo Notifications**
  https://www.youtube.com/watch?v=li1gFNFFRJY
- **AsyncStorage: armazenamento local (aula 42)**
  https://www.youtube.com/watch?v=pO10tFRs6CA
- **AsyncStorage (aula 32)**
  https://www.youtube.com/watch?v=lJxSzSs1WBQ

Sem aula atual em português para câmera e para áudio/voz no Expo.

Resumo

- Recursos sensíveis pedem permissão antes do uso.
- AsyncStorage guarda dados simples. SecureStore guarda dados sensíveis.
- Câmera, microfone e notificações têm bibliotecas do Expo.
- Notificações push remotas precisam de um build próprio.

Revisão

Responda no papel e confira o gabarito no fim da apostila.

1. Antes de usar a câmera, o app precisa...
   ( ) Pedir permissão ao usuário
   ( ) Ter CNPJ
   ( ) Instalar um segundo app
2. Onde guardar dado sensível, como um token de login, no celular?
   ( ) Em um texto dentro do código
   ( ) Em qualquer arquivo de texto do aparelho
   ( ) No SecureStore
3. Por que testar notificações push remotas exige um build próprio e não o Expo Go?
   ( ) Porque o Expo Go não suporta push remoto nas versões recentes
   ( ) Porque o Expo Go bloqueia todos os apps
   ( ) Porque push precisa de CNPJ
4. Por que um app pede permissão antes de acessar a câmera ou o microfone?
   ( ) Porque o Expo cobra uma taxa por cada permissão usada
   ( ) Para deixar o app mais rápido e gastar menos bateria
   ( ) Para proteger a privacidade: o usuário decide se deixa ou não
5. O que acontece se o usuário negar a permissão da câmera?
   ( ) O app não pode usar a câmera e deve avisar ou dar outra opção
   ( ) O celular reinicia sozinho e apaga os dados do app
   ( ) A câmera funciona normalmente mesmo sem a permissão
6. O AsyncStorage guarda os dados como...
   ( ) Texto, por isso objetos são convertidos com JSON.stringify
   ( ) Imagens, por isso o texto precisa ser convertido em foto
   ( ) Tabelas com relações, como em um banco de dados SQL
7. Os dados guardados no AsyncStorage somem quando o usuário fecha o app?
   ( ) Não, ficam guardados no celular para a próxima vez

( ) Sim, mas só se o celular ficar sem bateria
( ) Sim, somem na hora

8. As funções do AsyncStorage, como getItem, são assíncronas. O que isso exige no código?
   ( ) Chamar a função duas vezes
   ( ) Usar await (ou then) para esperar o resultado
   ( ) Escrever o código em letras maiúsculas
9. Qual destes dados é adequado para guardar no AsyncStorage?
   ( ) A senha da conta do usuário
   ( ) O número do cartão de crédito
   ( ) O tema escolhido, claro ou escuro
10. O que é uma notificação local?
    ( ) Um aviso que só funciona sem internet e sem app instalado
    ( ) Um aviso agendado e disparado pelo próprio app, sem servidor
    ( ) Uma mensagem de e-mail do Google
11. Qual destes é um bom exemplo de notificação local?
    ( ) Uma promoção enviada pela loja para todos
    ( ) Um aviso de que outro usuário te mandou mensagem
    ( ) Um lembrete diário às 8h para beber água
12. Qual destes é um exemplo típico de notificação push?
    ( ) Um aviso de mensagem nova enviado por um servidor
    ( ) Uma vibração curta quando o usuário toca na tela
    ( ) Um alarme agendado dentro do próprio celular
13. Para um app poder receber notificações push, ele precisa de...
    ( ) Um token que identifica aquele aparelho
    ( ) Uma câmera ativa e sempre ligada no aparelho
    ( ) Um cabo ligado ao computador do desenvolvedor

Desafios

Anotações

Módulo 13. Gerar APK/AAB e publicar na Play Store

Build, conta de desenvolvedor e teste fechado.

Aulas

- **Publicando apps na Play Store: faixas de teste**: Interno, fechado, aberto, produção
  https://www.youtube.com/watch?v=hQ0DcFH8W68
- **Restrições do teste fechado da Play Store (clube de testadores)**: Exigência dos testadores
  https://www.youtube.com/watch?v=GwejAaLBj7g
- **Grupo de testadores no teste fechado**
  https://www.youtube.com/watch?v=NzFLQj99E3M
- **Publicar app na Google Play**
  https://www.youtube.com/watch?v=WO1luWdul7c
- **Como publicar um app Android na Google Play: guia completo**: Canal provável: Stack Mobile
  https://www.youtube.com/watch?v=1bhowWSFfKY
- **Gerando APK**
  https://www.youtube.com/watch?v=OE-gilbYCDI
- **Google Play Closed Testing Explained (2026)**: Título em inglês, idioma a conferir
  https://www.youtube.com/watch?v=LugLyL1vmkw

Sem material atualizado em português de EAS Build, AAB e eas submit.

Resumo

- O build gera o APK (para testar) ou o AAB (para a loja).
- O EAS Build compila na nuvem, sem precisar do Android Studio.
- A Play Console exige conta de desenvolvedor e ficha do app.
- Contas pessoais novas pedem teste fechado com 12 testadores por 14 dias.

Revisão

Responda no papel e confira o gabarito no fim da apostila.

1. O que é um AAB?
   ( ) O pacote do app no formato que a Play Store exige para publicar
   ( ) Uma senha do Google
   ( ) Um tipo de vídeo
2. Por que guardar a chave de assinatura (keystore) em lugar seguro?
   ( ) Sem a chave certa, atualizar o app do jeito normal fica difícil ou impossível
   ( ) Porque a Play Store cobra por ela
   ( ) Porque ela contém as senhas dos usuários
3. Em contas pessoais novas, o que a Play Store costuma exigir antes de liberar a produção?
   ( ) 50 testadores por 1 ano
   ( ) Teste fechado com 12 testadores por pelo menos 14 dias
   ( ) 1 testador por 1 dia
4. Qual é a principal diferença entre APK e AAB?
   ( ) O APK instala direto para testar; o AAB é o enviado à loja
   ( ) O APK é o formato da loja; o AAB é só para instalar direto
   ( ) O APK roda só no iPhone e o AAB roda só no Android
5. O que é o EAS Build?

( ) Um serviço que compila o app na nuvem
( ) Um modelo novo de celular da Samsung
( ) Um editor de fotos e vídeos para celular

6. Qual comando costuma ser usado para pedir um build com o EAS?
   ( ) expo cook
   ( ) eas open
   ( ) eas build
7. Para que serve o campo versionCode no Android?
   ( ) Definir a cor do ícone e o tema que o app usa na loja
   ( ) Identificar cada envio; o número precisa subir a cada envio
   ( ) Guardar a senha secreta usada para assinar o app
8. O que é o applicationId (ou package) de um app Android, como com.exemplo.meuapp?
   ( ) O endereço do site que o app usa na internet
   ( ) O identificador único do app na loja e no celular
   ( ) O nome do usuário do desenvolvedor na Play Store
9. O que são as faixas de teste da Play Console, como interno e fechado?
   ( ) Formas de cobrar o usuário pelo uso do app no celular
   ( ) Etapas para liberar o app a grupos pequenos antes da produção
   ( ) Tipos de ícone usados para mostrar o app na loja do Google
10. Qual é a vantagem do teste interno?
    ( ) Liberar o app rápido para um grupo bem pequeno de pessoas
    ( ) Dispensar o envio do app para a loja do Google
    ( ) Publicar o app para o mundo inteiro de uma só vez

Desafios

Anotações

Trilha 4: Backend e dados

Onde ficam login, dados e regras do app.

Módulo 14. Firebase

Login, Firestore e regras de segurança.

Aulas

- **Curso de Firebase: iniciando com o Cloud Firestore (#7)** (Stack Mobile): 2022, série de Android, vale o conceito
  https://www.youtube.com/watch?v=Vy135ddtnzo
- **Curso de Firebase: leitura dos dados (#9)**: Mesma série
  https://www.youtube.com/watch?v=DeX0oiUwNSg
- **Curso de Firebase: como deletar (#11)**: Mesma série
  https://www.youtube.com/watch?v=jMN1LewT-W0
- **React Native e Firebase: autenticação com e-mail e senha**
  https://www.youtube.com/watch?v=Kfp-lfU87ls
- **Autenticação com Firebase, React Native e Redux**: Provavelmente antigo
  https://www.youtube.com/watch?v=scNe3S7QErs
- **Firebase Firestore e Storage: ler coleção (parte 1)**: Parte de um curso maior
  https://www.youtube.com/watch?v=VMthNkkXqLI

Sem aula em português de login com Google no Expo, de regras de segurança do Firestore e de curso atual de Firebase com Expo.

Resumo

- Firebase Auth cuida do login por e-mail ou Google.
- Firestore guarda documentos dentro de coleções.
- As regras de segurança controlam quem lê e escreve.
- Nunca deixe as regras abertas (if true) em produção.

Revisão

Responda no papel e confira o gabarito no fim da apostila.

1. O que é o Firestore?
   ( ) Um servidor de e-mail
   ( ) Um editor de código
   ( ) Um banco de dados na nuvem do Firebase
2. Para que servem as regras de segurança do Firestore?
   ( ) Definir quem pode ler e escrever em cada dado
   ( ) Deixar o app mais bonito
   ( ) Fazer backup automático
3. Qual regra é perigosa em produção?

( ) allow read: if request.auth != null;
( ) allow read, write: if true;
( ) allow write: if request.auth.uid == userId;

4. Quais são os dois jeitos de login citados para o Firebase Auth neste curso?
   ( ) E-mail e senha, e login com o Google
   ( ) Só pelo número do cartão
   ( ) Só impressão digital, sem outra opção
5. Em um cadastro com e-mail e senha, por que pedir a senha duas vezes?
   ( ) Para o Firebase cobrar menos por cada novo usuário
   ( ) Para deixar a senha mais forte e mais difícil de quebrar
   ( ) Para conferir se a pessoa digitou a senha igual nos dois campos
6. Quem guarda e protege as senhas dos usuários quando você usa o Firebase Auth?
   ( ) O Expo Go, que guarda todas as senhas no celular
   ( ) O próprio Firebase, sem que você precise guardar a senha
   ( ) Você mesmo, guardando as senhas em uma planilha
7. Como o Firestore organiza os dados?
   ( ) Em arquivos de texto soltos
   ( ) Em linhas de planilha única
   ( ) Em coleções que contêm documentos
8. O que é um documento no Firestore?
   ( ) Uma senha criptografada guardada no app
   ( ) Um arquivo PDF com os dados do usuário
   ( ) Um registro com campos, como nome e idade
9. Qual operação do Firestore cria um novo documento com id automático?
   ( ) addDoc
   ( ) sleepDoc
   ( ) readDoc
10. Para que serve o getDocs?
    ( ) Apagar todas as coleções do projeto
    ( ) Trocar a senha de um usuário logado
    ( ) Ler os documentos de uma coleção
11. O que faz a função deleteDoc?
    ( ) Apaga um documento específico
    ( ) Cria uma cópia do documento
    ( ) Apaga o projeto inteiro do Firebase
12. Qual regra de segurança do Firestore costuma ser usada para liberar acesso apenas a quem fez login?
    ( ) allow read, write: if false;
    ( ) allow read, write: if request.auth == null;
    ( ) allow read, write: if request.auth != null;
13. Por que não se deve colocar a chave secreta de administrador (service account) dentro do app do celular?
    ( ) Porque o Expo Go não aceita arquivos JSON de nenhum tipo

( ) Porque qualquer um poderia extrair a chave do app e ter acesso total
( ) Porque deixaria o app mais pesado e mais lento para abrir

Desafios

Anotações

Módulo 15. Supabase e SQL

Banco, login, RLS e SQL do básico ao intermediário.

Aulas

- **Curso Supabase gratuito para iniciantes 2026** (NoCode StartUp): Jan/2026, cerca de 1h05
  https://www.youtube.com/watch?v=OztHXOnQLXc
- **Como controlar acesso de usuários com RLS no Supabase** (FlutterFlow Brasil): Jun/2025, exemplos em FlutterFlow
  https://www.youtube.com/watch?v=iFzkI4whgTI
- **Sistema de autenticação com React Native e Supabase**: O mais próximo dos seus apps
  https://www.youtube.com/watch?v=SXjJX9T0QJY
- **Autenticação simplificada no React Native com Supabase**
  https://www.youtube.com/watch?v=IT3ETxxBVkI
- **Banco de dados relacional e SQL com Supabase**
  https://www.youtube.com/watch?v=wOcbetwEJD4
- **Banco de dados grátis na nuvem com Supabase (2025)**
  https://www.youtube.com/watch?v=adMbLA4x3-M
- **Curso Supabase: backend com FlutterFlow**
  https://www.youtube.com/watch?v=EuXKdw8ivUg
- **Usando o banco de dados do Supabase (aula 11)**
  https://www.youtube.com/watch?v=vq_nT1NxwoM
- **Regras de segurança no Supabase (aula 10)**
  https://www.youtube.com/watch?v=wmycZOeIDTU
- **Curso completo de SQL com PostgreSQL, do zero ao avançado** (Stack Mobile): Fev/2026, PostgreSQL é o banco do Supabase
  https://www.youtube.com/watch?v=9cAKQWodpvM
- **Curso completo de SQL com MySQL, nível iniciante**
  https://www.youtube.com/watch?v=BgUxpPKw2LI
- **Curso SQL completo 2023 com desafios**: Muita prática
  https://www.youtube.com/watch?v=G7bMwefn8RQ

- **SQL básico ao avançado (2019)**: Antigo, SQL muda pouco
  https://www.youtube.com/playlist?list=PLnNURxKyyLIInBfeGiJ8L314AD015mHkv
- **Curso SQL (playlist)**
  https://www.youtube.com/playlist?list=PLU8oAlHdN5Bmx-LChV4K3MbHrpZKefNwn

Sem playlist completa de Supabase com Expo em português.

Resumo

- Tabela tem colunas e linhas. A chave primária identifica cada linha.
- SELECT lê, INSERT cria, UPDATE altera, DELETE apaga e WHERE filtra.
- RLS define quem acessa cada linha.
- A chave anon fica no app. A service_role nunca.

Revisão

Responda no papel e confira o gabarito no fim da apostila.

1. O que é RLS (Row Level Security)?
   ( ) Um servidor de vídeo
   ( ) Regras por linha que dizem quem pode ver ou alterar cada linha da tabela
   ( ) Um tipo de gráfico
2. Qual comando SQL busca dados?
   ( ) INSERT
   ( ) SELECT
   ( ) DROP
3. Qual chave pode ficar no app: a anon (publishable) ou a service_role?
   ( ) Nenhuma, nunca
   ( ) A service_role
   ( ) A anon, protegida por RLS
4. Em uma tabela de banco de dados, o que é uma linha?
   ( ) Uma regra de segurança da tabela
   ( ) Um registro, como um cliente guardado
   ( ) O nome de uma coluna da tabela
5. O que é uma coluna em uma tabela?
   ( ) Um usuário que pode acessar o banco
   ( ) Um tipo de informação, como nome ou e-mail
   ( ) Um backup da tabela inteira
6. Para que serve a chave primária de uma tabela?
   ( ) Criptografar os dados da tabela
   ( ) Dar permissão de administrador ao usuário
   ( ) Identificar cada linha de forma única
7. O que este comando faz? INSERT INTO clientes (nome) VALUES ('Ana');
   ( ) Lê todos os clientes da tabela clientes
   ( ) Apaga a linha da cliente Ana na tabela
   ( ) Cria uma nova linha em clientes com o nome Ana

8. Qual comando altera dados que já existem?
   ( ) INSERT
   ( ) SELECT
   ( ) UPDATE
9. O que faz este comando? SELECT * FROM produtos;
   ( ) Cria um produto vazio na tabela produtos
   ( ) Lê todas as colunas e linhas da tabela produtos
   ( ) Apaga todas as linhas da tabela produtos
10. Qual comando apaga uma linha da tabela?
   ( ) CLEAR SELECT
   ( ) DELETE
   ( ) REMOVE TABLE
11. Por que é perigoso rodar DELETE FROM clientes; sem o WHERE?
   ( ) Ele apaga todas as linhas da tabela
   ( ) Ele apaga só a primeira linha
   ( ) Ele não faz nada
12. Para que serve o WHERE em um comando SQL?
   ( ) Criar uma cópia de segurança do banco
   ( ) Ordenar as colunas pelo nome da tabela
   ( ) Filtrar quais linhas o comando vai atingir
13. O que acontece com a chave service_role se ela for colocada no código do app?
   ( ) O app fica mais rápido e bem mais seguro
   ( ) Alguém pode achá-la e ter acesso total ao banco
   ( ) Nada acontece, pois ela é uma chave pública
14. Se você ativa o RLS em uma tabela e não cria nenhuma política, o que normalmente acontece com os acessos feitos pelo app?
   ( ) Eles ficam bloqueados
   ( ) Todos passam a ver tudo
   ( ) Só o dono do projeto vê, e sempre com erro
15. Qual é um bom uso de uma política de RLS?
   ( ) Deixar o site mais bonito e mais leve
   ( ) Trocar a senha do banco de dados todo dia
   ( ) Deixar cada usuário ver só as próprias linhas
16. O que é autenticação em um app com Supabase?
   ( ) Confirmar quem é o usuário, como por e-mail e senha
   ( ) Deixar as telas do app mais leves e rápidas
   ( ) Guardar todas as imagens e vídeos do seu app
17. O Supabase usa qual banco de dados por baixo?
   ( ) Excel
   ( ) Apenas arquivos de texto
   ( ) PostgreSQL
18. Uma tabela pedidos tem a coluna cliente_id apontando para a tabela clientes. Isso é um exemplo de quê?

( ) Relação entre tabelas
( ) Uma política de RLS
( ) Um erro de digitação

19. Qual comando mostra só os clientes da cidade de Itaquera?
    ( ) UPDATE clientes SET cidade = 'Itaquera';
    ( ) SELECT cidade FROM Itaquera;
    ( ) SELECT * FROM clientes WHERE cidade = 'Itaquera';

Desafios

1. Primeira tabela
Crie a tabela produtos (id, nome, preco). Insira 3 produtos. Depois liste só os que custam mais de 10, do mais barato para o mais caro.
Dica: CREATE TABLE, INSERT INTO ... VALUES, SELECT ... WHERE ... ORDER BY.
https://sqliteonline.com/

2. Alterar e apagar
Com a tabela do desafio anterior: aumente em 10% o preço do Caderno e depois apague o produto mais barato.
*Dica: UPDATE ... SET preco = preco * 1.10 WHERE nome = 'Caderno'; e DELETE FROM ... WHERE ... (sempre com WHERE!).*
https://sqliteonline.com/

Anotações

Módulo 16. APIs, Node.js e Cloudflare Workers

Criar a sua própria API.

Aulas

- **Curso de API Rest, Node e TypeScript** (Lucas Souza Dev): 49 vídeos
  https://www.youtube.com/playlist?list=PL29TaWXah3iaaXDFPgTHiFMBF6wQahurP
- **Curso de Node.js 2025 para iniciantes** (Stack Mobile): 12 vídeos
  https://www.youtube.com/playlist?list=PLizN3WA8HR1w14FUaPYsP9q1nBHuTa1Dv
- **API completa com Node e Express, do zero ao pro (aula 01)**
  https://www.youtube.com/watch?v=zLTmbR8xa84
- **Como criar uma API completa com Node e Express**
  https://www.youtube.com/watch?v=Cdu0WJhI-d8

- **Criando uma API do zero com Node.js**
  https://www.youtube.com/watch?v=6Y1cMlTPEaE
- **APIs rápidas e gratuitas com Cloudflare Workers**: Única opção de Workers em português
  https://www.youtube.com/watch?v=zX0g2btna5E
- **Edge computing e Cloudflare Workers**: Conceito, idioma a conferir
  https://www.youtube.com/watch?v=lL_KbnCKt1k

Sem playlist de Cloudflare Workers em português.

Resumo

- API recebe pedidos HTTP (GET, POST) e devolve JSON.
- Códigos: 200 ok, 400 pedido inválido, 401 sem login, 404 não achou, 500 erro no servidor.
- Cloudflare Workers rodam seu código na nuvem sem servidor próprio.
- Valide tudo o que chega antes de usar.

Revisão

Responda no papel e confira o gabarito no fim da apostila.

1. O que é uma API?
   ( ) Um tipo de banco de dados
   ( ) Uma imagem do app
   ( ) Um endereço que recebe pedidos e devolve dados para outros programas
2. O que significa o código HTTP 404?
   ( ) Não encontrado
   ( ) Servidor sobrecarregado
   ( ) Sucesso
3. Qual a vantagem de um Cloudflare Worker?
   ( ) Rodar seu código no servidor sem manter uma máquina própria
   ( ) Guardar fotos
   ( ) Editar o app no celular
4. O que significa JSON, no contexto de uma API?
   ( ) Uma linguagem para desenhar telas
   ( ) Um formato de texto para organizar dados
   ( ) Um tipo de banco de dados na nuvem
5. Qual método HTTP costuma ser usado para enviar dados novos a uma API, como um cadastro?
   ( ) SEND
   ( ) HEAD
   ( ) POST
6. Qual método HTTP costuma ser usado para buscar dados?
   ( ) POST
   ( ) SEND
   ( ) GET
7. O que é o Node.js?
   ( ) Uma forma de rodar JavaScript fora do navegador

( ) Um serviço de banco de dados em nuvem pública
( ) Um editor de imagens e de vídeos online

8. Para que serve o Express no Node.js?
   ( ) Criptografar as senhas do banco sozinho
   ( ) Hospedar o seu site na internet de graça
   ( ) Facilitar a criação de rotas e servidores de API
9. Qual código HTTP indica que deu tudo certo?
   ( ) 500
   ( ) 404
   ( ) 200
10. A API devolveu 401. O que isso quer dizer?
    ( ) O servidor está fora do ar neste momento
    ( ) Você não está autenticado: falta login ou chave
    ( ) A página que você pediu não foi encontrada
11. A API devolveu 500. De quem é o problema, normalmente?
    ( ) Do servidor, que teve um erro interno
    ( ) Da internet do usuário, que está lenta
    ( ) Do navegador do usuário, que está antigo
12. Por que validar os dados que chegam em uma API?
    ( ) Porque qualquer pessoa pode enviar dados maliciosos
    ( ) Porque o JSON só funciona com dados validados
    ( ) Porque assim o servidor fica bem mais barato
13. O que é um Cloudflare Worker?
    ( ) Um navegador alternativo para o seu celular
    ( ) Um programa para desenhar e editar imagens
    ( ) Um pequeno código que roda na nuvem da Cloudflare

Desafios

Anotações

Módulo 17. Segurança para iniciantes

Chaves de API, variáveis de ambiente e senhas.

Aulas

- **Como não enviar chaves de API para o Git**: .gitignore e .env
  https://www.youtube.com/watch?v=Oow_50bPS30
- **Pare de usar senha no GitHub**
  https://www.youtube.com/watch?v=aezsqluwCyo
- **Proteja sua chave de API com variáveis de ambiente (07)**: Aula 7 de uma série
  https://www.youtube.com/watch?v=QMe6OAPubwE
- **Guardar informações confidenciais: .env e variáveis de ambiente**: Exemplo em Python, vale para qualquer linguagem
  https://www.youtube.com/watch?v=orBR5mFYUFE

Sem curso de segurança para iniciantes. Faltam 2FA, o que fazer se uma chave vazar e segredos em Vercel, Cloudflare e GitHub.

Resumo

- Segredos ficam em variáveis de ambiente, nunca no código.
- O .env deve estar no .gitignore.
- Chave vazada deve ser revogada e trocada.
- Use senhas únicas e verificação em duas etapas.

Revisão

Responda no papel e confira o gabarito no fim da apostila.

1. Onde guardar uma chave secreta de API?
   ( ) Em um comentário do GitHub
   ( ) Em variável de ambiente ou secret do serviço
   ( ) Dentro do código do app
2. Se uma chave vazou no GitHub, o que fazer?
   ( ) Revogar e criar outra, porque ela fica no histórico
   ( ) Esperar para ver se alguém usa
   ( ) Só apagar o arquivo
3. Para que serve o arquivo .gitignore?
   ( ) Criar backup
   ( ) Dizer ao Git quais arquivos não devem ser enviados, como o .env
   ( ) Acelerar o código
4. O que é um arquivo .env?
   ( ) Um arquivo com segredos, fora do código
   ( ) Uma imagem usada no layout do site
   ( ) Um backup completo do seu GitHub
5. Por que não se deve escrever a chave de API direto no código?
   ( ) Porque o código fica mais lento ao rodar
   ( ) Porque o código pode ser visto por outras pessoas
   ( ) Porque as chaves não funcionam dentro de código
6. Se o .env está listado no .gitignore, o que acontece com ele no git add?

( ) Ele é enviado ao GitHub, mas criptografado
( ) O git o ignora e ele não é enviado ao GitHub
( ) Ele é apagado do seu computador pelo git

7. Por que apagar a chave do código depois que ela vazou não basta?
   ( ) Porque ela continua no histórico, então é preciso trocá-la
   ( ) Porque chaves nunca expiram nem podem ser trocadas
   ( ) Porque o GitHub troca a chave vazada por conta própria
8. Qual destas é uma boa prática de senha?
   ( ) Usar a mesma senha em tudo para não esquecer
   ( ) Mandar a senha por mensagem para si mesmo
   ( ) Usar uma senha diferente em cada serviço
9. O que é a verificação em duas etapas?
   ( ) Ter duas contas diferentes com o mesmo nome
   ( ) Pedir um segundo passo além da senha, como um código
   ( ) Digitar a mesma senha duas vezes para confirmar
10. Qual destas chaves pode ficar visível em um app, por ser feita para isso?
    ( ) Uma chave secreta de pagamento
    ( ) Uma chave pública, como a anon do Supabase
    ( ) Uma chave de administrador do banco

Desafios

Anotações

Trilha 5: Deploy e domínio

Colocar no ar com endereço próprio.

Módulo 18. GitHub Actions

Builds automáticos sem gastar créditos à toa.

Aulas

- **Master class GitHub Actions**: Aula completa
  https://www.youtube.com/watch?v=GlJAUiQZ79w
- **Como criar pipelines CI/CD com GitHub Actions**
  https://www.youtube.com/watch?v=cmv8XjvfQZ4
- **CI e CD com GitHub Actions**
  https://www.youtube.com/watch?v=-mfSUpLsx6c
- **GitHub Actions: primeiros passos**
  https://www.youtube.com/watch?v=ecXzb6az1rg
- **Apresentando o GitHub Actions (aula 1)**: Mini-curso em aulas
  https://www.youtube.com/watch?v=6bzqVaNls5g
- **GitHub Actions (aula 6)**: Mesmo mini-curso
  https://www.youtube.com/watch?v=c8ekZEHvJ4g
- **Workflow automatizado de CI com GitHub Actions**
  https://www.youtube.com/watch?v=F51HlrEeedw
- **Fundamentos de CI/CD (aula 01)** (WebAcademy): Canal pelo título
  https://www.youtube.com/watch?v=yfRlEJHDHWc

Nada específico sobre build de app Expo com Actions. Vários vídeos podem ser de 2021 a 2023.

Resumo

- Workflow é um arquivo YAML em .github/workflows.
- Os gatilhos podem ser push, pull request ou um horário.
- Secrets guardam as chaves usadas pelo workflow.
- Junte mudanças para gastar menos minutos.

Revisão

Responda no papel e confira o gabarito no fim da apostila.

1. O que é um workflow do GitHub Actions?
   ( ) Um plugin de edição
   ( ) Um conjunto de passos automáticos definido em um arquivo YAML
   ( ) Um tipo de repositório
2. Em qual pasta ficam os workflows?
   ( ) public/ci
   ( ) src/actions

( ) .github/workflows

3. Como gastar menos minutos de Actions?
   ( ) Rodar só quando necessário e juntar várias mudanças antes de enviar
   ( ) Enviar cada letra alterada
   ( ) Deixar rodar em loop
4. O que é o GitHub Actions?
   ( ) Um serviço que roda tarefas automáticas
   ( ) Um editor de código feito para o celular
   ( ) Um banco de dados que roda na nuvem
5. O que significa CI, em CI/CD?
   ( ) Integração contínua, que testa o código a cada mudança
   ( ) Código de interface, que desenha as telas do app
   ( ) Conexão de internet, que liga o app ao servidor
6. O que significa CD, em CI/CD?
   ( ) Criação de design, que desenha as telas do app
   ( ) Entrega contínua, que publica o resultado sozinho
   ( ) Cópia de dados, que guarda um backup do banco
7. Em que formato é escrito o arquivo de um workflow?
   ( ) HTML
   ( ) JPEG
   ( ) YAML
8. O que significa o gatilho 'on: push' em um workflow?
   ( ) O workflow roda só à meia-noite, todos os dias
   ( ) Roda quando alguém envia código ao repositório
   ( ) O workflow nunca roda sozinho, nem com envios
9. Para que servem os Secrets do GitHub?
   ( ) Esconder o repositório dos mecanismos de busca do Google
   ( ) Deixar o workflow mais rápido e mais barato de rodar
   ( ) Guardar chaves e senhas do workflow sem expô-las no código
10. O que é um job em um workflow?
    ( ) Um tipo de arquivo de configuração do GitHub
    ( ) Um conjunto de passos executado em uma máquina
    ( ) Um usuário com acesso total ao repositório
11. O que é um step (passo) em um job?
    ( ) Um arquivo de imagem guardado no repositório
    ( ) Um comentário escrito dentro do código
    ( ) Uma tarefa individual, como instalar pacotes
12. Por que é útil o gatilho de execução manual em um workflow?
    ( ) Porque você escolhe quando gastar minutos e rodar o build
    ( ) Porque deixa o código mais seguro automaticamente
    ( ) Porque apaga os builds antigos
13. Se um passo do workflow falha, o que normalmente aparece no GitHub?

( ) Aparece como falho, com um X vermelho e o log do erro
( ) O GitHub apaga o repositório inteiro na hora
( ) Nada aparece, pois o erro é sempre escondido

Desafios

Anotações

Módulo 19. Vercel e Cloudflare Pages

Hospedagem gratuita de sites e PWAs.

Aulas

- **Deploy de landing page grátis na Vercel**: Passo a passo
  https://www.youtube.com/watch?v=xE_PXDtzbxo
- **Como fazer deploy na Vercel (2026)**
  https://www.youtube.com/watch?v=q8gfD5pz1f8
- **Como hospedar um site na Cloudflare de graça (2026)**
  https://www.youtube.com/watch?v=FRyPM7_ixfU
- **Hospede seu site feito com IA grátis na Cloudflare Pages**: Bem próximo do que você faz
  https://www.youtube.com/watch?v=VkyHVFwSOvQ
- **Hospedagem grátis para site estático e domínio próprio na Cloudflare** (Aylton Inacio): Set/2025
  https://www.youtube.com/watch?v=WcebtNdJDjM

A Cloudflare está migrando Pages para Workers, então confira se o passo a passo ainda vale.

Resumo

- Deploy publica a versão atual do site.
- Com o repositório ligado, cada push gera um novo deploy.
- As variáveis de ambiente do serviço guardam as chaves do deploy.
- Vercel e Cloudflare Pages têm plano gratuito.

Revisão

Responda no papel e confira o gabarito no fim da apostila.

1. O que o deploy faz?
   ( ) Muda o domínio
   ( ) Apaga o repositório

( ) Publica a versão atual do site na internet

2. O que acontece quando você faz push em um repositório ligado ao Cloudflare Pages ou à Vercel?
   ( ) Nada, é preciso avisar por e-mail
   ( ) Um novo deploy é feito automaticamente
   ( ) O domínio expira
3. O que é uma variável de ambiente no deploy?
   ( ) Um tipo de imagem
   ( ) Um comentário
   ( ) Um valor configurado no serviço (como uma chave) que não fica no código
4. O que é um site estático?
   ( ) Um site que nunca muda de endereço nem de conteúdo
   ( ) Um site feito de arquivos prontos, como HTML, CSS e JS
   ( ) Um site que só funciona sem internet, offline
5. Qual a vantagem de usar Vercel ou Cloudflare Pages em vez de manter seu próprio servidor?
   ( ) Você só envia o código, e eles cuidam da hospedagem
   ( ) O site fica imune a erros e a qualquer ataque
   ( ) Eles escrevem e desenham o site inteiro por você
6. O que é uma URL de preview?
   ( ) Um endereço secreto usado para acessar o banco
   ( ) O endereço do seu e-mail para receber avisos
   ( ) Um endereço de teste, antes de ir para o site principal
7. O que acontece se o build do seu site falhar no deploy?
   ( ) O domínio é cancelado e volta a ficar livre
   ( ) O site inteiro é apagado do serviço de hospedagem
   ( ) A nova versão não vai ao ar, e a anterior continua
8. Onde você deve colocar uma chave secreta usada pelo seu site hospedado?
   ( ) No arquivo index.html do seu site público
   ( ) Em um comentário dentro do código do site
   ( ) Nas variáveis de ambiente do serviço de hospedagem
9. O que é um build?
   ( ) Um tipo de domínio usado em sites brasileiros
   ( ) Uma cópia do site feita para o celular do usuário
   ( ) O processo que transforma o código nos arquivos finais
10. Para o deploy automático funcionar, o que você liga ao serviço de hospedagem?
    ( ) O seu número de celular
    ( ) O repositório do GitHub
    ( ) O seu e-mail principal

Desafios

Anotações

Módulo 20. Domínio, DNS e HTTPS

Registrar o domínio e apontar para o site.

Aulas

- **Domínio próprio na Vercel com DNS e Cloudflare**: O mais próximo do seu caso
  https://www.youtube.com/watch?v=gcRMKjyuPhI
- **Domínio do Registro.br na Cloudflare DNS**
  https://www.youtube.com/watch?v=j7M-dMrBo8o
- **Como registrar domínio no Registro.br (2026)**
  https://www.youtube.com/watch?v=zPXlVWTAFGU
- **Como comprar um domínio no Registro.br**
  https://www.youtube.com/watch?v=xdAd56ApgJw
- **Apontando domínio para a Cloudflare (parte 2)**
  https://www.youtube.com/watch?v=gRz5Zl-aZlo
- **Como apontar domínio para hospedagem com Cloudflare**
  https://www.youtube.com/watch?v=35vg431Xj2c

Nenhum vídeo explica DNS (A, CNAME, TTL) e HTTPS de forma didática.

Resumo

- O DNS traduz nomes em endereços.
- Registros comuns: A, CNAME, MX e TXT.
- O HTTPS criptografa a conexão (cadeado).
- Mudanças de DNS podem demorar para valer.

Revisão

Responda no papel e confira o gabarito no fim da apostila.

1. O que o DNS faz?
   ( ) Protege contra vírus
   ( ) Traduz o nome do site para o endereço do servidor
   ( ) Cria o site
2. O cadeado (HTTPS) no navegador significa que...
   ( ) O site é 100% seguro e confiável
   ( ) A conexão com o site é criptografada
   ( ) O site é do governo
3. Qual registro DNS aponta um subdomínio para outro nome?
   ( ) MX

( ) CNAME
( ) TXT

4. O que é um domínio?
   ( ) O arquivo principal do seu site
   ( ) A senha do painel de hospedagem
   ( ) O nome do site, como meusite.com.br
5. Para que serve o Registro.br?
   ( ) Hospedar vídeos e imagens na nuvem
   ( ) Registrar domínios terminados em .br
   ( ) Criar apps de celular para Android
6. O que são nameservers (servidores de nomes) de um domínio?
   ( ) Os servidores de e-mail usados pelo GitHub
   ( ) Os servidores que dizem onde estão os registros DNS
   ( ) Os servidores que guardam o código do site
7. Para que serve o registro MX no DNS?
   ( ) Dizer para onde vão os e-mails do domínio
   ( ) Apontar o site para um endereço de IP
   ( ) Ativar o cadeado do HTTPS no navegador
8. Para que serve o registro A no DNS?
   ( ) Apagar o domínio quando ele vence
   ( ) Guardar a senha do seu domínio
   ( ) Apontar um nome para um endereço IP
9. Para que serve o registro TXT no DNS?
   ( ) Aumentar a velocidade do site para os visitantes
   ( ) Enviar mensagens SMS para os visitantes do site
   ( ) Guardar um texto, usado para provar que o domínio é seu
10. Você mudou o DNS e o site ainda abre do jeito antigo. Qual é a explicação mais comum?
    ( ) O navegador não aceita nenhuma mudança de DNS
    ( ) A mudança pode demorar para se espalhar pela internet
    ( ) O domínio foi apagado da internet sem aviso

Desafios

Anotações

Trilha 6: IA e automação

API da Anthropic, bots, automação e agentes.

Módulo 21. API da Anthropic e prompts

Chamar o Claude pelo seu app.

Aulas

- **Claude, IA da Anthropic** (DR Tech): 13 vídeos. É sobre usar o Claude no chat, não a API por código
  https://www.youtube.com/playlist?list=PLToYQ1snAluIDPC6s1c0F9SvNHMsT9_8O

Não achei aula em português sobre chamar a API por código. A documentação oficial em docs.claude.com explica passo a passo.

Resumo

- A chave da API não fica no app. Chame por um servidor seu.
- A cobrança é por uso.
- O prompt é a instrução e o contexto enviados.
- Limite o uso por pessoa para não ter surpresa na conta.

Revisão

Responda no papel e confira o gabarito no fim da apostila.

1. Por que a chave da API de IA não pode ficar dentro do app?
   ( ) Porque qualquer pessoa poderia copiá-la e gastar seu dinheiro
   ( ) Porque a chave expira em um minuto
   ( ) Porque deixa o app mais lento
2. Como costuma funcionar a cobrança da API?
   ( ) Por uso (quantidade de texto enviado e recebido)
   ( ) Valor fixo, grátis para sempre
   ( ) Só por download do app
3. O que é um prompt?
   ( ) Um tipo de banco de dados
   ( ) O texto de instrução e contexto enviado ao modelo
   ( ) Uma senha
4. No contexto de IA de texto, o que é um "token"?
   ( ) Uma imagem que a IA gera a partir do texto que você escreve
   ( ) Uma senha que o usuário digita no app para entrar na sua conta
   ( ) Um pedaço de texto, como uma palavra ou parte dela, que o modelo lê
5. Por que vale a pena colocar contexto no prompt, como quem vai ler a resposta?
   ( ) Porque deixa a IA mais rápida e barata em qualquer situação
   ( ) Porque assim a IA nunca erra e não precisa de conferência

( ) Porque a IA responde de forma mais útil quando sabe a situação

6. Qual a diferença entre usar o Claude no chat e pela API?
   ( ) Na API só é possível enviar imagens, e nunca é possível enviar texto
   ( ) No chat você paga por uso, e na API o uso nunca é cobrado de ninguém
   ( ) No chat você digita; na API o código envia o pedido e recebe a resposta
7. Seu app manda a pergunta para o seu servidor, e o servidor chama a API da IA. Qual a principal vantagem?
   ( ) A chamada à IA deixa de ser cobrada em qualquer situação
   ( ) A chave da API fica guardada no servidor, longe de quem usa o app
   ( ) A IA passa a responder sem precisar de internet no celular
8. Por que limitar quantas perguntas cada pessoa pode fazer por dia?
   ( ) Para a IA não ficar cansada de responder tantas perguntas
   ( ) Porque a lei proíbe fazer mais de uma pergunta por dia
   ( ) Para uma só pessoa não gastar todo o crédito e gerar uma conta alta
9. Qual destes trechos de prompt pede um formato de resposta?
   ( ) "Preciso de ajuda com uma coisa aqui no meu app"
   ( ) "Responda em no máximo 3 frases, em português simples"
   ( ) "Olá, tudo bem? Espero que você esteja bem hoje"
10. Onde a chave da API deve ficar guardada no servidor?
    ( ) Em uma variável de ambiente, fora do código que vai para o repositório
    ( ) Em um comentário dentro do HTML da página, para não esquecer
    ( ) Escrita direto no arquivo de código do app, para facilitar o uso
11. A aula "Claude, IA da Anthropic" do módulo mostra principalmente o quê?
    ( ) Como programar um servidor do zero usando Node e Express
    ( ) Como publicar um app Android na Play Store passo a passo
    ( ) Como usar o Claude no chat; ela não ensina a chamar a API por código
12. A IA deu uma resposta com um número importante para o seu app. O que fazer?
    ( ) Conferir em uma fonte confiável antes de usar
    ( ) Pedir à IA para repetir até ela concordar com você
    ( ) Usar direto, pois a IA nunca erra números
13. Por que um prompt muito longo pode custar mais?
    ( ) Porque textos longos são sempre proibidos pela API
    ( ) Porque a IA cobra uma taxa fixa por cada letra maiúscula
    ( ) Porque mais texto enviado e recebido significa mais tokens contados

Desafios

Anotações

Módulo 22. Bots de WhatsApp

API oficial da Meta e webhooks.

Aulas

- **Como conectar seu WhatsApp à API Oficial da Meta** (Whatsify Pro): Passo a passo da conta, número e verificação. Tem divulgação da ferramenta e não usa Node
  https://www.youtube.com/watch?v=xvHu8HyaMlg

Não achei aula em português de bot em Node com a API da Meta.

Resumo

- A API oficial da Meta permite criar bots no WhatsApp.
- Webhook é o endereço que recebe as mensagens.
- Confira a assinatura do webhook.
- Respeite as regras de mensagem e o consentimento das pessoas.

Revisão

Responda no papel e confira o gabarito no fim da apostila.

1. O que é um webhook?
   ( ) Um botão do WhatsApp
   ( ) Um endereço seu que recebe avisos automáticos de outro serviço
   ( ) Um tipo de mensagem
2. Para que serve conferir a assinatura do webhook?
   ( ) Traduzir o texto
   ( ) Deixar mais rápido
   ( ) Garantir que a mensagem veio mesmo da Meta
3. Qual API oficial permite automação no WhatsApp?
   ( ) Qualquer app que imita o WhatsApp Web
   ( ) Um plugin do Chrome
   ( ) WhatsApp Business Platform (Cloud API da Meta)
4. Quando alguém manda uma mensagem para o número do bot, o que a Meta faz?
   ( ) Envia os dados da mensagem ao webhook do seu servidor
   ( ) Guarda a mensagem e espera você abrir o celular para ler
   ( ) Responde sozinha à pessoa, sem usar nenhum código seu
5. Por que o webhook precisa de um endereço acessível pela internet?
   ( ) Porque o WhatsApp só funciona com sites que mostram anúncios
   ( ) Porque o endereço precisa ser igual ao do seu e-mail pessoal
   ( ) Porque a Meta precisa conseguir chegar até ele para entregar as mensagens
6. Onde deve ficar guardado o token de acesso da API do WhatsApp?
   ( ) No texto da página do site, para o bot ler
   ( ) No servidor, em variável de ambiente

( ) Em um grupo de WhatsApp, para não perder

7. Antes de mandar mensagens de divulgação por um bot, o que é correto fazer?
   ( ) Ter a permissão (consentimento) da pessoa para receber
   ( ) Comprar uma lista de números e disparar
   ( ) Mandar para todos os contatos que você tem, rápido
8. O webhook recebeu uma mensagem. O que ele deve fazer logo em seguida?
   ( ) Responder rápido com sucesso (código 200) e tratar o resto depois
   ( ) Ficar parado esperando o dono do bot aprovar a mensagem
   ( ) Fechar a conexão com a Meta sem responder coisa nenhuma
9. Uma pessoa escreve "parar" para o bot. O que é o correto?
   ( ) Ignorar o pedido, pois ela já aceitou receber antes
   ( ) Enviar mais mensagens para tentar convencer a pessoa
   ( ) Parar de enviar mensagens a ela e registrar o pedido
10. A aula "Como conectar seu WhatsApp à API Oficial da Meta" cobre principalmente o quê?
    ( ) Como programar o bot inteiro em Node
    ( ) Como criar a arte e o logo do bot
    ( ) Os passos de conta, número e verificação
11. As mensagens chegam ao webhook em JSON. O que é JSON?
    ( ) Um tipo de vírus que infecta o WhatsApp e rouba mensagens
    ( ) Um formato de texto com pares de nome e valor, como {"nome": "Ana"}
    ( ) Uma linguagem usada só para desenhar as telas dos apps
12. Por que preferir a API oficial a programas não oficiais que controlam o WhatsApp?
    ( ) É sempre mais fácil de configurar do que qualquer outra opção
    ( ) Segue as regras da plataforma e reduz o risco de o número ser bloqueado
    ( ) Dispensa qualquer verificação de conta ou de número de telefone
13. O bot não sabe responder uma dúvida. Qual a melhor atitude?
    ( ) Inventar uma resposta que pareça boa
    ( ) Avisar que não sabe e passar a conversa para uma pessoa
    ( ) Repetir a mesma mensagem até a pessoa desistir

Desafios

Anotações

Módulo 23. Automação de navegador e tarefas agendadas

Playwright, n8n e agendamentos.

Não achei playlist em português conferida de Playwright, n8n ou GitHub Actions. Assuntos para buscar: Puppeteer ou Playwright, node-cron e GitHub Actions agendado.

Resumo

- Playwright controla um navegador por código.
- Cron agenda tarefas em horários fixos.
- O n8n liga serviços com fluxos visuais.
- Leia os termos de uso antes de automatizar um site.

Revisão

Responda no papel e confira o gabarito no fim da apostila.

1. Para que serve o Playwright?
   ( ) Controlar um navegador por código para automatizar tarefas e testes
   ( ) Comprimir imagens
   ( ) Criar banco de dados
2. O que é um cron?
   ( ) Um editor de imagem
   ( ) Um agendamento que roda uma tarefa em horários definidos
   ( ) Um tipo de vírus
3. Antes de automatizar um site, o que conferir?
   ( ) Nada, tudo é permitido
   ( ) Só a cor do site
   ( ) Se os termos de uso do site permitem
4. O que faz a linha de código await page.goto("https://exemplo.com") em um script com Playwright?
   ( ) Apaga o site do endereço informado
   ( ) Cria um site novo com esse nome
   ( ) Manda o navegador abrir esse endereço
5. O que significa a expressão de cron 0 9 * * * ?
   ( ) Executar todo dia às 9h
   ( ) Executar a cada 9 minutos
   ( ) Executar somente no dia 9 de cada mês
6. Na expressão de cron, qual é o primeiro dos cinco campos?
   ( ) A semana
   ( ) O segundo
   ( ) O minuto
7. No n8n, o que é um "nó"?
   ( ) O nome do plano pago da ferramenta de automação
   ( ) Um passo do fluxo, como um gatilho ou uma ação
   ( ) Um erro que aparece e trava o fluxo inteiro de repente
8. O que é o gatilho (trigger) de um fluxo de automação?
   ( ) O evento que inicia o fluxo, como um horário ou mensagem
   ( ) A senha usada para poder editar o fluxo de automação
   ( ) O botão que apaga todo o fluxo de automação de uma vez

9. O que significa rodar o navegador em modo "headless"?
    ( ) Rodar sem permissão do usuário
    ( ) Rodar sem mostrar a janela na tela
    ( ) Rodar sem internet
10. Um site mudou o layout e a sua automação parou de funcionar. Por quê?
    ( ) O código procurava elementos pela posição ou nome antigos, que mudaram
    ( ) O Playwright expira todo mês e precisa ser comprado de novo
    ( ) O navegador passou a bloquear qualquer cron
11. Por que não deixar um robô clicando sem pausa em um site?
    ( ) Pode sobrecarregar o site e fazer você ser bloqueado
    ( ) O robô fica cada vez mais lento até parar de funcionar
    ( ) Deixa o computador sem energia depois de algumas horas
12. Quando o serviço oferece uma API oficial para o que você quer, o que costuma ser melhor?
    ( ) Evitar a API, pois ela quebra mais do que cliques na tela
    ( ) Sempre simular cliques na tela, que é mais seguro que a API
    ( ) Usar a API, que é mais estável do que automatizar o navegador
13. Uma tarefa agendada no seu computador precisa de quê para rodar no horário?
    ( ) Que o computador esteja ligado e o programa possa ser executado
    ( ) Que você fique olhando a tela do computador o tempo todo
    ( ) Que o navegador esteja fechado e a internet desligada

Desafios

Anotações

Trilha 7: Segurança e privacidade

Proteger seus apps e sistemas e cumprir o que a lei e as lojas exigem.

Módulo 24. Segurança dos apps e sistemas

Checklist para conferir antes de publicar qualquer app, site ou bot.

Aulas

- **Segurança de Software: curso gratuito OWASP Top 10** (pessonizando): 13 vídeos sobre as ameaças mais comuns, com foco prático
  https://www.youtube.com/playlist?list=PLEqTHftpM91OZzAlOwMcAuQ4ciK1n4_Ll

Ainda faltam aulas em português sobre segredos e variáveis de ambiente, RLS no Supabase e regras do Firestore.

Resumo

- Chaves e segredos nunca vão no código.
- O banco precisa de RLS ou regras ligadas.
- Login com 2FA e conferência de permissão no servidor.
- Confira o checklist antes de cada publicação.

Revisão

Responda no papel e confira o gabarito no fim da apostila.

1. Qual chave pode ficar no site, desde que o banco tenha regras?
   ( ) A chave pública (anon/publishable)
   ( ) A service_role
   ( ) A senha do seu e-mail
2. Esconder um botão na tela protege o dado?
   ( ) Só no celular
   ( ) Não, o servidor também precisa conferir a permissão
   ( ) Sim, sempre
3. Por que ligar a verificação em duas etapas?
   ( ) A conta continua protegida mesmo se a senha vazar
   ( ) Economiza dados do celular
   ( ) O login fica mais rápido
4. O que é o OWASP Top 10?
   ( ) Uma lista das ameaças mais comuns de segurança em aplicações web
   ( ) Um ranking dos 10 sites mais visitados do mundo inteiro
   ( ) Um programa que protege o computador sozinho contra vírus
5. O que é uma injeção de SQL (SQL injection)?
   ( ) Quando o site demora muito para carregar por causa de imagens grandes
   ( ) Quando alguém digita comandos de banco de dados num campo e o sistema os executa

( ) Quando o banco de dados fica cheio e para de salvar os dados novos

6. Um arquivo de código com const chave = "sk-123..." vai para um repositório público. Qual o problema?
   ( ) O arquivo vai ficar mais lento no editor
   ( ) Nenhum, pois o código de um app é só texto comum
   ( ) Qualquer pessoa pode ler a chave e usá-la
7. Uma chave secreta foi publicada sem querer. O que fazer?
   ( ) Apagar o arquivo e continuar usando a mesma chave
   ( ) Revogar a chave e gerar uma nova
   ( ) Esperar para ver se alguém usa
8. Para que serve o RLS (segurança em nível de linha) em um banco de dados?
   ( ) Para deixar a tela do banco de dados mais bonita
   ( ) Para definir quais linhas cada usuário pode ver ou alterar
   ( ) Para fazer uma cópia automática do site inteiro
9. Por que validar os dados também no servidor, e não só no formulário?
   ( ) Porque o formulário do site não aceita números nos campos
   ( ) Porque o servidor é sempre mais lento do que o formulário
   ( ) Porque o usuário pode burlar o formulário e falar direto com o servidor
10. Para que serve o HTTPS em um site?
    ( ) Impedir que o site seja copiado
    ( ) Proteger os dados que viajam entre o usuário e o site
    ( ) Fazer o site abrir mais rápido em qualquer caso

Desafios

Anotações

Módulo 25. Privacidade, LGPD e políticas

Política de privacidade, termos de uso, exclusão de conta e o que a Play Store e o AdSense pedem.

Aulas

- **Adequando sites e lojas virtuais à LGPD** (Hostnet (com Curso em Vídeo)): 14 vídeos sobre consentimento e política de privacidade. O foco é WordPress, mas serve de base https://www.youtube.com/playlist?list=PLtcmVcOuK65ylRTDz7WAkhY9Kw62kyijw
- **LGPD Descomplicada** (SPIC Brasil): 6 vídeos em linguagem simples. É conceito geral, não específico de apps

https://www.youtube.com/playlist?list=PLXHA7gQw8rYGXdIQoCiMnaPUGbdYLnxl1

Não achei aula em português sobre política de privacidade específica para apps. Este módulo é um roteiro geral e não substitui um advogado.

Resumo

- Dado pessoal exige finalidade, necessidade e cuidado.
- Publique política de privacidade e termos de uso.
- Permita ver, corrigir e excluir os dados.
- Preencha a segurança de dados da Play Console de acordo com o que o app faz.

Revisão

Responda no papel e confira o gabarito no fim da apostila.

1. O que é dado sensível pela LGPD?
   ( ) Só senhas
   ( ) Dado como saúde, biometria, religião ou opinião política
   ( ) Qualquer nome
2. Onde deve ficar a Política de Privacidade?
   ( ) Só em inglês
   ( ) Escondida no código
   ( ) Em uma página pública, com link no cadastro e na ficha do app
3. O app cria conta de usuário. O que o Google Play pede?
   ( ) Nada
   ( ) Uma forma de excluir a conta e os dados, no app e por um endereço na web
   ( ) Um CNPJ
4. Na LGPD, quem é o "titular" dos dados?
   ( ) A empresa que guarda e usa os dados pessoais
   ( ) O programador que fez o app que coleta os dados
   ( ) A pessoa a quem os dados pessoais se referem
5. O que significa o princípio da finalidade?
   ( ) Usar os dados só para o propósito que foi informado à pessoa
   ( ) Guardar os dados para sempre, por garantia
   ( ) Usar os dados para o que a empresa quiser depois
6. Para o consentimento ser válido, ele deve ser...
   ( ) Dado por uma caixinha já marcada, sem a pessoa precisar clicar
   ( ) Livre e informado, com a pessoa sabendo para que os dados serão usados
   ( ) Escondido nos termos de uso, em letra pequena e difícil de achar
7. Quais destes são direitos do titular dos dados?
   ( ) Receber uma parte do lucro mensal da empresa
   ( ) Mudar o código-fonte do app que usa os dados
   ( ) Acessar, corrigir e pedir a exclusão dos seus dados
8. Qual a diferença entre política de privacidade e termos de uso?
   ( ) São o mesmo texto, apenas escrito com dois nomes diferentes

( ) A política explica o tratamento dos dados; os termos, as regras de uso
( ) A política é só para empresas e os termos só para pessoas comuns

9. O que as aulas do "LGPD Descomplicada" oferecem?
   ( ) Conceitos gerais da lei, em linguagem simples e sem foco em apps
   ( ) Um passo a passo de como programar o login de um app
   ( ) A lista de preços para vender cursos online na internet
10. Seu app usa a localização do celular. O que fazer na seção de segurança de dados da Play Console?
    ( ) Não declarar esse uso, pois o usuário já sabe disso
    ( ) Declarar esse uso de acordo com o que o app realmente faz
    ( ) Declarar esse uso somente se o app for um app pago

Desafios

Anotações

Trilha 8: Negócio

Ganhar dinheiro com o que você constrói.

Módulo 26. AdSense, afiliados e Kiwify

Monetizar sites e vender produto digital.

Aulas

- **Como monetizar site e blog em 2025 com o Google AdSense**: Playlist atual
  https://www.youtube.com/playlist?list=PLpEvhvupz-uTSbd1Zz9a77o8xCLK35BZW
- **Tudo sobre Google AdSense**: Série ampla
  https://www.youtube.com/playlist?list=PLIzk9IesBa8G5Q05z6weoeAQOhrtNfuin
- **Como monetizar site rápido: curso completo**
  https://www.youtube.com/watch?v=RJs41rHxtxk
- **Como se cadastrar no Google AdSense (2026)**
  https://www.youtube.com/watch?v=BOhX1HPZCMI
- **Curso Monetizando Blogs com o AdSense**: Canal provável: Kildary Oliver
  https://www.youtube.com/watch?v=ETC3wolMVJI
- **Como inserir seu blog no AdSense e ser aprovado**: Foco em aprovação
  https://www.youtube.com/watch?v=abo7avqsNy4
- **Afiliado Mercado Livre: passo a passo (2026)**
  https://www.youtube.com/watch?v=TRrxcd18auU
- **Tudo sobre o programa de afiliados do Mercado Livre**
  https://www.youtube.com/watch?v=XFqZI4fxcQg
- **Mercado Livre afiliados: como começar do zero**
  https://www.youtube.com/watch?v=arUCW_2ysDA
- **Curso completo de afiliado Shopee do zero (2026)**
  https://www.youtube.com/watch?v=SuyOUy6h5IM
- **Curso gratuito para afiliados Shopee**: Playlist
  https://www.youtube.com/playlist?list=PLob6rnOOKlf72FgoKBEsTXyegz1GQmiB0
- **Afiliado Shopee: passo a passo completo (2026)**
  https://www.youtube.com/watch?v=lP2gAXTs7z0
- **Afiliado Shopee na prática**: Playlist
  https://www.youtube.com/playlist?list=PL9HxdmlLGD0iqGZ24TLrMspciAxQ1azgN
- **Mini curso grátis de afiliado digital 2026**
  https://www.youtube.com/watch?v=Wa7pLXO8mvI
- **4 formas de divulgar seu link de afiliado**
  https://www.youtube.com/watch?v=aBCzorRLMu4
- **Kiwify: como fazer sua primeira venda em 20 minutos**
  https://www.youtube.com/watch?v=9a_cbyIJnRo
- **Como cadastrar seu produto na Kiwify e começar a vender**
  https://www.youtube.com/watch?v=Iu9gAYLOVPY

- **Kiwify para iniciantes: cadastrar produtos (2026)**
  https://www.youtube.com/watch?v=51wVUmuR6jo
- **Sem investir nada: como fiz minha primeira venda na Kiwify**
  https://www.youtube.com/watch?v=1idoJgR4K_8

Sem aula específica de afiliado AliExpress e de vender software ou app na Kiwify.

Resumo

- O AdSense exige conteúdo próprio, política de privacidade e páginas básicas.
- Link de afiliado gera comissão por venda.
- A Kiwify vende produto digital e cuida do pagamento.
- Divulgue com transparência.

Revisão

Responda no papel e confira o gabarito no fim da apostila.

1. O que o AdSense costuma exigir para aprovar um site?
   ( ) Só ter muitos anúncios
   ( ) Ter um aplicativo na loja
   ( ) Conteúdo próprio e útil, política de privacidade e páginas básicas
2. O que é um link de afiliado?
   ( ) Um link gratuito para qualquer loja
   ( ) Um link que apaga os cookies
   ( ) Um link com o seu código, que gera comissão nas vendas feitas por ele
3. Para que serve a Kiwify?
   ( ) Hospedar bancos de dados
   ( ) Criar apps nativos
   ( ) Vender produtos digitais e receber os pagamentos
4. Quem paga ao dono do site no AdSense?
   ( ) O Google, com base nos anúncios exibidos no site
   ( ) O provedor de hospedagem onde o site está publicado
   ( ) Os visitantes, cada vez que entram na página do site
5. Por que você não deve clicar nos anúncios do seu próprio site?
   ( ) Porque o anúncio some para sempre depois do clique
   ( ) Porque o site fica mais lento a cada clique feito
   ( ) Isso é cliques inválidos e pode levar ao bloqueio da conta
6. Em geral, quando um afiliado ganha comissão?
   ( ) Quando alguém compra usando o link dele e a venda é concluída
   ( ) Quando o próprio afiliado compra o produto para si
   ( ) Só de alguém abrir o link dele, mesmo sem comprar
7. Por que avisar que o link é de afiliado?
   ( ) Por transparência: o leitor deve saber que você pode ganhar
   ( ) Para o produto ficar mais barato para quem for comprar
   ( ) Para o link ficar mais curto e mais fácil de copiar

8. Qual destes é um produto digital, próprio para vender na Kiwify?
   ( ) Uma camiseta lisa
   ( ) Um e-book em PDF
   ( ) Uma caixa de som
9. O que normalmente se faz em um programa de afiliados, como os do Mercado Livre e da Shopee, para ganhar comissão?
   ( ) Comprar estoque e revender depois
   ( ) Criar a loja virtual do vendedor
   ( ) Gerar o link do produto e divulgá-lo
10. Um site com textos copiados de outros sites pode ser aprovado no AdSense?
    ( ) Sim, se o site tiver muitos anúncios espalhados nele
    ( ) Dificilmente, pois o AdSense valoriza conteúdo próprio e útil
    ( ) Sim, desde que o site tenha muitas imagens bonitas

Desafios

Anotações

Módulo 27. Vender para clientes locais

Prospecção e primeira abordagem.

Aulas

- **Como vender sites todos os meses**
  https://www.youtube.com/watch?v=bVq-iN90N7I
- **Como vender sites e onde encontrar clientes**
  https://www.youtube.com/watch?v=DWR1hCun4Bc
- **Como conseguir clientes sendo freelancer: guia de prospecção**
  https://www.youtube.com/watch?v=JuFj_y2UX2Q
- **Ache clientes para vender sites com o ChatGPT**
  https://www.youtube.com/watch?v=eD53kdGu9Bw
- **3 formas de se conectar com o cliente na primeira abordagem**
  https://www.youtube.com/watch?v=vxkf9awY8oM
- **Como prospectar clientes do zero: guia para iniciantes**
  https://www.youtube.com/watch?v=IGaYty1q5bg

Só vídeos soltos, sem playlist. Nada específico sobre vender app ou PWA.

Resumo

- Comece por um problema real do cliente.
- Mostre um exemplo pronto.
- Combine o que está incluso no serviço.
- Peça indicação aos clientes satisfeitos.

Revisão

Responda no papel e confira o gabarito no fim da apostila.

1. Qual é um bom primeiro passo para abordar um comércio local?
   ( ) Mostrar um exemplo pronto que resolve um problema dele
   ( ) Mandar uma lista de tecnologias
   ( ) Falar só do preço
2. O que perguntar ao cliente antes de orçar?
   ( ) Qual linguagem de programação ele prefere
   ( ) Qual problema ele quer resolver e o que usa hoje
   ( ) Quantos anos o site tem
3. Por que combinar o que está incluso no serviço?
   ( ) Não faz diferença
   ( ) Para cobrar mais caro
   ( ) Evita pedidos extras sem pagamento e mal-entendidos
4. O que é prospecção de clientes?
   ( ) Procurar e abordar possíveis clientes que não conhecem seu trabalho
   ( ) Pedir desconto a fornecedores na compra de materiais para a loja
   ( ) Cobrar um cliente antigo que está devendo um pagamento atrasado
5. Como deve ser uma primeira mensagem para um comércio local?
   ( ) Igual para todos, escrita toda em letras maiúsculas
   ( ) Muito longa, com todo o seu currículo e histórico
   ( ) Curta, educada e falando de algo específico do negócio dele
6. O cliente respondeu "vou pensar". Qual a melhor atitude?
   ( ) Insistir todo dia, por mensagem, até ele finalmente responder
   ( ) Agradecer e combinar quando você pode falar com ele de novo
   ( ) Esquecer o cliente e nunca mais entrar em contato com ele
7. Qual a vantagem de mostrar um exemplo pronto ao cliente?
   ( ) Evita que o cliente faça perguntas sobre o trabalho feito
   ( ) Ele vê o resultado na prática e entende melhor o que vai receber
   ( ) Dispensa de combinar o preço e o prazo do trabalho com ele
8. Como a IA pode ajudar, segundo a aula sobre achar clientes com o ChatGPT?
   ( ) Fechando o contrato sozinha com o cliente, sem precisar de você
   ( ) Garantindo que todo cliente abordado vai aceitar a proposta
   ( ) Dando ideias de tipos de negócio e rascunhando mensagens, que você revisa
9. Onde é possível encontrar negócios locais que talvez precisem de um site?
   ( ) Em mapas e buscas da sua região, vendo quem não tem site ou tem um fraco
   ( ) Só em empresas muito grandes que ficam em outros países distantes

( ) Apenas em quem já pediu um site a você antes de qualquer contato

10. O que significa vender um serviço recorrente, como a manutenção mensal de um site?
    ( ) O serviço é feito de graça no primeiro ano
    ( ) O cliente paga uma vez e nunca mais fala com você
    ( ) O cliente paga todo mês por um serviço contínuo

Desafios

Anotações

Módulo 28. Preço, contrato e cobrança

Quanto cobrar e como receber sem CNPJ.

Aulas

- **Quanto cobrar por um site ou software? Como precificar**
  https://www.youtube.com/watch?v=GsX0ipcZ9ss
- **Quanto cobrar por um site (2025)**
  https://www.youtube.com/watch?v=3L9spiJLqWc
- **Quanto cobrar para criar um site e tabela de preços**
  https://www.youtube.com/watch?v=guQL-UojAKc
- **Quanto cobrar por um projeto freelancer (Ep. 10)** (Programador Br): Canal pelo título
  https://www.youtube.com/watch?v=BoFJHhq1dk0
- **Quanto e como cobrar por projetos web**
  https://www.youtube.com/watch?v=tP3lP9C1hKc
- **MEI precisa de contrato de serviço? Modelo gratuito**
  https://www.youtube.com/watch?v=P0L-KsfwEZI
- **Como emitir um RPA: comprovar serviço sem CNPJ**
  https://www.youtube.com/watch?v=Kd3qSIBwbng
- **Programador freelancer precisa ter CNPJ? Quando abrir MEI**
  https://www.youtube.com/watch?v=Es6S6mzNZ1Y
- **RPA: guia completo para contratar autônomos**
  https://www.youtube.com/watch?v=UKOdtQ3KWqU

Nada sobre atendimento ao cliente, escopo e cobrança de quem não paga. Confira as regras fiscais e de INSS do RPA antes de usar.

Resumo

- Preço é o tempo, mais os custos, mais o valor entregue.
- Um contrato simples tem entrega, prazo, valor e pagamento.
- Cobrar uma parte antes protege os dois lados.
- Sem CNPJ, veja com um contador como declarar e quando formalizar.

Revisão

Responda no papel e confira o gabarito no fim da apostila.

1. O que um contrato simples deve ter?
   ( ) Apenas a cor do site
   ( ) Só o nome do cliente
   ( ) O que será entregue, o prazo, o valor e a forma de pagamento
2. Cobrar uma parte antes de começar ajuda a...
   ( ) Dispensar o contrato
   ( ) Garantir o compromisso do cliente e cobrir o início do trabalho
   ( ) Ficar rico sem trabalhar
3. Sem CNPJ, qual o cuidado ao receber por serviços?
   ( ) É proibido receber dinheiro
   ( ) Nenhum, não precisa de cuidado
   ( ) Entender como declarar a renda e quando vale formalizar (como MEI)
4. Que itens ajudam a montar o preço de um site?
   ( ) Tempo de trabalho, custos e o valor que ele traz ao cliente
   ( ) Só o tamanho em megabytes do arquivo final do site pronto
   ( ) Só o preço que o concorrente mais barato está cobrando hoje
5. O que é o escopo de um serviço?
   ( ) O horário em que você trabalha
   ( ) O que será feito e o que não será incluído
   ( ) A cor do site escolhida pelo cliente
6. O cliente pede algo extra que não estava no combinado. O que é recomendável?
   ( ) Recusar o pedido sem conversar e sumir do contato dele
   ( ) Fazer o pedido sem avisar o cliente e só cobrar bem depois
   ( ) Combinar por escrito um valor e prazo adicionais para esse pedido
7. O que é um RPA?
   ( ) Um robô que automatiza tarefas dentro do navegador web
   ( ) Um recibo de pagamento a autônomo, que registra o serviço feito
   ( ) Um tipo de imposto que é cobrado sobre os sites criados
8. Por que guardar recibos, conversas e comprovantes de pagamento do trabalho?
   ( ) Para o cliente poder cancelar o serviço quando quiser
   ( ) Para registrar o que foi combinado e pago e poder comprovar
   ( ) Porque o banco exige que esses registros sejam apagados
9. Qual a vantagem de ter um contrato por escrito em vez de só combinar por conversa?
   ( ) Garante que o cliente nunca vai reclamar
   ( ) Registra o combinado e reduz mal-entendidos

( ) Dispensa o pagamento

10. Em um projeto grande, uma forma comum de cobrar é por etapas. Como funciona?
    ( ) Tudo só depois que o cliente usar o site por um ano
    ( ) Parte do valor na entrada e o resto em momentos combinados
    ( ) O cliente paga só se gostar, sem combinar nada antes

Desafios

Anotações

Gabaritos e soluções

Só olhe depois de tentar sozinho.

Módulo 1. Como a internet, apps e sistemas funcionam

Gabarito:

1. Um computador sempre ligado que responde aos pedidos de outros aparelhos. O servidor guarda o site ou os dados e responde quando o navegador ou o app (o cliente) faz um pedido.
2. No banco de dados. O banco de dados serve para guardar e consultar informações de forma organizada e segura.
3. O cliente pede, o servidor processa e responde. O cliente sempre começa com um pedido (requisição) e o servidor devolve uma resposta.
4. O navegador ou app que faz pedidos e mostra a resposta. Cliente é o programa (navegador ou app) que pede informações ao servidor e mostra o que volta.
5. Traduzir o nome do site para o endereço do servidor. Você digita um nome fácil de lembrar, e o DNS descobre qual servidor tem aquele site.
6. O navegador descobre o endereço do servidor e faz o pedido. Primeiro o navegador acha o servidor (via DNS) e pede a página. Só depois vem a resposta.
7. Guardar de forma organizada informações como clientes e pedidos. O banco de dados armazena e organiza as informações para que o app possa consultá-las depois.
8. Ser publicado em uma loja, como a Play Store. Apps são distribuídos por lojas, como a Play Store, ou abertos pelo navegador como site/PWA.
9. Um app que roda pelo navegador, sem passar por uma loja. Sites e PWAs rodam no navegador, então a pessoa só precisa do endereço para usar.
10. No servidor, dentro de um banco de dados. Dados guardados no servidor podem ser lidos de qualquer aparelho. O que fica só na tela se perde.

Módulo 2. Lógica de programação

Gabarito:

1. Um laço (for ou while). O laço repete os passos até a condição de parada acontecer.
2. Um valor que pode mudar durante o programa. Variável é um nome ligado a um valor, que pode ser trocado.
3. Condição (if). A condição escolhe um caminho conforme o valor testado.
4. Uma sequência de passos para resolver um problema. Algoritmo é a receita em passos, que pode ser escrita em qualquer linguagem.
5. x passa a valer 6. O computador calcula o lado direito (5 + 1) e guarda o resultado de volta em x.
6. Para dizer o que fazer quando a condição do se é falsa. O senão é o caminho alternativo, usado quando a condição do se não é verdadeira.
7. 3 vezes. O laço repete o bloco até completar a quantidade pedida, aqui 3 vezes.

8. O laço roda sem parar (laço infinito). Sem uma condição que termine a repetição, o laço continua para sempre e pode travar o programa.
9. Agrupar passos sob um nome para poder usá-los várias vezes. Dar nome a um grupo de passos evita repetir código e deixa o programa mais organizado.
10. Parâmetros (ou argumentos). Os valores que a função recebe para trabalhar se chamam parâmetros ou argumentos.

Variáveis e texto

```
<p id="saida"></p>

var nome = "Ana";
var idade = 30;
document.getElementById("saida").textContent = "Meu nome é " + nome + " e tenho " +
idade + " anos";
```

Condições

```
function classificar(idade) {
  if (idade < 12) {
    return "criança";
  } else if (idade < 18) {
    return "adolescente";
  } else {
    return "adulto";
  }
}

console.log(classificar(10));
```

Laços

```
function somar(n) {
  var total = 0;
  for (var i = 1; i <= n; i++) {
    total = total + i;
  }
  return total;
}

console.log(somar(5));
```

O maior de três

```
function maior(a, b, c) {
  var m = a;
  if (b > m) {
    m = b;
  }
  if (c > m) {
    m = c;
  }
  return m;
}

console.log(maior(3, 9, 5));
```

Módulo 3. Terminal, Git e GitHub

Gabarito:

1. Grava no histórico uma foto do estado atual dos arquivos. O commit fica só no seu computador. Quem envia é o push.
2. git push. git add escolhe, git commit grava, git push envia.
3. Guardar o código na internet e acompanhar o histórico de mudanças. O GitHub guarda repositórios Git na nuvem e facilita backup e trabalho em equipe.
4. Muda para outra pasta. cd vem de change directory (mudar de diretório) e serve para entrar em outra pasta.
5. ls. ls lista o conteúdo da pasta atual.
6. mkdir fotos. mkdir (make directory) cria uma pasta nova com o nome indicado.
7. git add, git commit, git push. Primeiro escolhe os arquivos (add), depois grava no histórico (commit) e por fim envia (push).
8. Escolhe quais arquivos entram no próximo commit. git add marca os arquivos que serão gravados no próximo commit.
9. Uma linha paralela de trabalho, separada da principal. Com uma branch você testa mudanças sem mexer na linha principal do projeto.
10. Junta as mudanças de volta em outra branch. Merge é o ato de unir as mudanças de uma branch em outra.
11. Ele guarda segredos, como chaves e senhas, que outras pessoas poderiam ver. Segredos em repositório podem ser vistos ou roubados, principalmente se o repositório for público.
12. Um projeto com seus arquivos e todo o histórico de mudanças. O repositório guarda os arquivos do projeto e cada versão gravada pelos commits.
13. Você errou e consegue voltar a uma versão anterior do projeto. Como cada commit fica no histórico, é possível ver e recuperar versões antigas.

Módulo 4. HTML e CSS

Gabarito:

1. <h1>. h1 é o título mais importante. p é parágrafo e a é link.

2. Define o visual da página: cores, tamanhos e espaços. O HTML é a estrutura, o CSS é o visual e o JavaScript é o comportamento.
3. <a href="..."> . A tag a, com o atributo href, leva para outro endereço.
4. <p>. A tag <p> é usada para parágrafos. <h1> é título e <a> é link.
5. <img>. A tag <img> mostra uma imagem, e o atributo src diz onde está o arquivo.
6. . é a lista sem ordem, e cada item dentro dela usa <li>.
7. O bloco usado para criar formulários, como cadastro ou login. O <form> reúne campos e botões para a pessoa preencher e enviar dados.
8. p { color: red; }. Escrevemos o seletor (p) e, entre chaves, a propriedade color com o valor desejado.
9. Margin é o espaço por fora do elemento, padding é o espaço por dentro. Padding afasta o conteúdo da borda por dentro, e margin afasta o elemento dos vizinhos.
10. .botao { }. O ponto indica classe. O # é usado para id.
11. display: flex;. Com display: flex, os filhos do contêiner passam a se organizar em linha ou coluna.
12. Organizar elementos em linhas e colunas, como uma tabela de layout. O Grid divide o espaço em linhas e colunas para posicionar elementos.
13. O id deve ser único na página, e a class pode ser usada em vários elementos. O id identifica um único elemento. A class serve para agrupar vários elementos com o mesmo estilo.

Estrutura de uma página

```
<h1>Minha página</h1>
<p>Primeiro parágrafo.</p>
<p>Segundo parágrafo.</p>
<ul>
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
</ul>
<a href="https://exemplo.com">Meu link</a>
```

Cores e cantos

```
<h1>Olá</h1>
<button>Entrar</button>

h1 {
  color: blue;
  text-align: center;
}
button {
  background: green;
  color: white;
  border: 0;
  border-radius: 8px;
  padding: 8px 16px;
}
```

Flexbox

```
<div id="grupo">
  <div class="caixa">1</div>
  <div class="caixa">2</div>
  <div class="caixa">3</div>
</div>

.caixa {
  background: #ccd;
  padding: 16px;
}
#grupo {
  display: flex;
  gap: 12px;
}
```

Formulário de login

```
<form>
  <label for="email">E-mail</label>
  <input id="email" type="email">
  <label for="senha">Senha</label>
  <input id="senha" type="password">
  <button type="submit">Entrar</button>
</form>
```

Módulo 5. JavaScript do básico ao avançado

Gabarito:

1. document.getElementById. getElementById devolve o elemento com aquele id, para você ler ou alterar.
2. Compara se os dois valores são iguais e do mesmo tipo. O = atribui. O === compara valor e tipo.
3. Cria uma nova lista aplicando uma função em cada item. map não altera a lista original. Ele devolve uma nova.
4. console.log("Oi"). console.log escreve no console, que ajuda a testar e achar erros.
5. let idade = 20;. let cria variável que pode receber outro valor. const cria uma que não pode ser reatribuída.
6. Dá erro, porque uma const não pode receber outro valor. Uma variável const não aceita nova atribuição.
7. "53". Com um texto envolvido, o + junta os valores como texto em vez de somar.
8. 1. O símbolo % dá o resto da divisão: 10 dividido por 3 sobra 1.
9. A página vista como objetos que o JavaScript pode ler e mudar. Pelo DOM, o JavaScript consegue mudar textos, estilos e elementos da página.
10. titulo.textContent = "Novo";. A propriedade textContent guarda o texto do elemento, e podemos atribuir um novo valor a ela.
11. addEventListener. addEventListener liga um evento (como click) a uma função.

12. 10. A contagem de posições em arrays começa do zero, então arr[0] é o primeiro item.
13. push. push coloca no final. pop faz o contrário e remove o último.
14. filter. filter mantém apenas os itens para os quais o teste é verdadeiro.
15. find. find devolve o primeiro item que atende ao teste, e filter devolve todos.
16. pessoa.idade. Em objetos usamos o ponto seguido do nome da propriedade.
17. Executa o bloco só se a idade for 18 ou mais. O if testa a condição e só roda o bloco quando ela é verdadeira.
18. Buscar dados de um endereço na internet. fetch faz um pedido a um endereço, por exemplo uma API, e devolve a resposta.
19. Esperar a resposta chegar antes de seguir para a próxima linha. Com await, o código aguarda o resultado de algo demorado, como um fetch.

Clique e troque o texto

```
<p id="msg">Ainda não clicou</p>
<button id="botao">Clique</button>

var botao = document.getElementById("botao");
botao.onclick = function () {
  document.getElementById("msg").textContent = "Olá!";
};
```

Contador

```
<button id="menos">-</button>
<span id="valor">0</span>
<button id="mais">+</button>

var n = 0;
var valor = document.getElementById("valor");

document.getElementById("mais").onclick = function () {
  n = n + 1;
  valor.textContent = n;
};

document.getElementById("menos").onclick = function () {
  if (n > 0) {
    n = n - 1;
  }
  valor.textContent = n;
};
```

Lista de tarefas

```
<input id="tarefa">
<button id="add">Adicionar</button>
<ul id="lista"></ul>

var campo = document.getElementById("tarefa");
var lista = document.getElementById("lista");

document.getElementById("add").onclick = function () {
  var texto = campo.value.trim();
  if (texto === "") {
    return;
  }
  var li = document.createElement("li");
  li.textContent = texto;
  lista.appendChild(li);
  campo.value = "";
};
```

Listas com map e filter

```
function dobrar(lista) {
  return lista.map(function (x) {
    return x * 2;
  });
}

function pares(lista) {
  return lista.filter(function (x) {
    return x % 2 === 0;
  });
}

console.log(dobrar([1, 2, 3]));
console.log(pares([1, 2, 3, 4]));
```

Evento de digitação

```
<input id="senha" type="text">
<p id="aviso"></p>

var campo = document.getElementById("senha");
var aviso = document.getElementById("aviso");

campo.addEventListener("input", function () {
  if (campo.value.length >= 8) {
    aviso.textContent = "Senha forte";
  } else {
    aviso.textContent = "Senha fraca";
  }
});
```

Módulo 6. PWA e design responsivo

Gabarito:

1. Aplica estilos só quando a tela tem até 600 px de largura. Media query deixa o visual diferente conforme o tamanho da tela.
2. O manifest (manifest.json). O manifest descreve o app para o navegador poder instalá-lo.
3. Guardar arquivos em cache para o app abrir mesmo sem internet. O service worker fica entre o app e a rede e decide o que vem do cache.
4. Põe os filhos lado a lado e deixa organizá-los com regras como justify-content.. display: flex ativa o flexbox no elemento, que passa a organizar os filhos em uma linha (ou coluna).
5. Centralizar os itens no sentido da linha, horizontalmente.. justify-content distribui os itens ao longo do eixo principal; com center, eles ficam no meio.
6. Duas colunas com a mesma largura.. Cada 1fr é uma fração do espaço livre, então dois valores iguais dão duas colunas iguais.
7. Fazer a página usar a largura real da tela do celular, para o design responsivo funcionar.. Sem ela, muitos celulares mostram a página como se a tela fosse larga e encolhem tudo.
8. Porque ele pode interceptar e guardar o que o site carrega, então exige conexão segura.. O service worker tem muito poder sobre as requisições, por isso os navegadores exigem HTTPS (a exceção é o localhost para testes).
9. Empacotar um PWA para poder publicá-lo em lojas, como a Google Play.. O PWABuilder pega o endereço do seu PWA e gera o pacote para enviar à loja.
10. Começar a estilizar pensando no celular e depois ajustar para telas maiores.. No mobile first, o CSS base é o da tela pequena e as media queries adicionam ajustes para telas maiores.

Media query

```
<h1>Responsivo</h1>
<p>Diminua a largura da janela para ver.</p>

body {
  background: white;
}

@media (max-width: 600px) {
  body {
    background: lightyellow;
  }
}
```

Grade de cards

```
<div id="cards">
  <div class="card">1</div>
  <div class="card">2</div>
  <div class="card">3</div>
</div>

.card {
  background: #ddd;
  padding: 20px;
}
#cards {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  gap: 10px;
}
```

Módulo 7. TypeScript básico

Gabarito:

1. Tipos, que avisam erros antes de rodar. Os tipos ajudam a achar erros enquanto você escreve o código.
2. let nome: string = "Ana". O tipo vem depois do nome, separado por dois-pontos.
3. O TypeScript mostra um erro antes de você rodar. Esse é o grande ganho do TypeScript: o erro aparece cedo.
4. let idade: number = 30;. Números, inteiros ou com vírgula, usam o tipo number.
5. O TypeScript mostra um erro, porque boolean só aceita true ou false.. O tipo boolean só aceita true e false; um texto não serve.
6. number[]. Colocar [] depois do tipo indica uma lista (array) daquele tipo.
7. O tipo do valor que a função devolve.. O tipo depois dos parênteses é o tipo do retorno da função.
8. Não, o TypeScript precisa ser convertido para JavaScript antes.. O TypeScript existe para o programador; o que roda no navegador ou no Node é o JavaScript gerado.
9. tsc. O tsc é o compilador do TypeScript; ele lê arquivos .ts e gera arquivos .js.
10. Que nome é uma string, pelo valor inicial.. Isso se chama inferência de tipo: o TypeScript deduz o tipo pelo valor dado.
11. Dá erro, porque falta o campo idade.. Todo campo declarado no tipo do objeto é obrigatório, a menos que seja marcado como opcional.
12. let cidade: string = "Recife";. Em TypeScript o nome do tipo vem depois do nome da variável, com dois pontos, e o tipo de texto é string.
13. Um molde para criar objetos com os mesmos campos e funções.. Com a classe você descreve como o objeto é, e depois cria várias cópias dele com new.

Função com tipos

Tipo de objeto

Módulo 8. TypeScript intermediário

Gabarito:

1. Descrever o formato (campos e tipos) de um objeto. A interface é um contrato: diz quais campos o objeto precisa ter.
2. Tipos que ficam abertos para serem definidos no uso, reaproveitando o código. Com generics, uma função funciona com vários tipos mantendo a checagem.
3. Deixa a função ou o tipo disponível para outros arquivos importarem. export e import dividem o código em arquivos.
4. Que o campo idade é opcional.. O ? depois do nome do campo permite que ele não exista no objeto.
5. Id pode ser um texto ou um número.. O símbolo | cria uma união: o valor pode ser de um tipo ou de outro.
6. Um tipo que será definido na hora de usar a função.. T é um tipo genérico, como um espaço em branco que é preenchido a cada uso.
7. O TypeScript continua sabendo o tipo do resultado.. Com generics o código serve para vários tipos sem perder as checagens; o any desliga as checagens.
8. Traz a função soma do arquivo util para o arquivo atual.. O import busca algo exportado por outro arquivo, aqui o util na mesma pasta.
9. Colocar export antes dela.. Só o que é exportado fica disponível para ser importado em outros arquivos.
10. São duas formas de escrever o mesmo tipo: uma lista de textos.. Array<string> usa a sintaxe de generics para a mesma lista que string[] descreve.
11. O TypeScript avisa o erro antes de rodar.. Campos sem ? são obrigatórios; o erro aparece já no editor e na compilação.
12. Impede que o valor seja alterado depois que o objeto foi criado.. readonly deixa o campo apenas para leitura; tentar mudar o valor gera erro.
13. Que a classe Carro tenha tudo o que a interface Veiculo descreve.. implements é um contrato: a classe precisa ter os campos e funções da interface.

Interface e genérico

Módulo 9. TypeScript avançado

Gabarito:

1. Torna todos os campos do tipo T opcionais. Partial é útil em atualizações, em que só alguns campos mudam.

2. Porque desliga a checagem de tipos e esconde erros. Com any, o TypeScript deixa de ajudar. Prefira unknown e confira o tipo.
3. O valor pode ser texto ou número. A barra vertical cria uma união de tipos.
4. Cria um tipo só com o campo nome de Usuario.. Pick escolhe, entre os campos de um tipo, apenas os que você indicar.
5. Cria um tipo igual a Usuario, mas sem o campo senha.. Omit é o contrário de Pick: tira os campos indicados e mantém o resto.
6. Um objeto cujas chaves são textos e cujos valores são números.. Record define o tipo das chaves e o tipo dos valores de um objeto.
7. Torna obrigatórios todos os campos de T, inclusive os opcionais.. Required é o oposto de Partial: remove o ? de todos os campos.
8. Os nomes dos campos: "nome" ou "idade".. keyof gera uma união com os nomes das propriedades do tipo.
9. Conferir o tipo dele, por exemplo com um if.. unknown é mais seguro que any porque obriga a checar o tipo antes de usar o valor.
10. O TypeScript passa a tratar x como string.. Isso é narrowing: a checagem estreita o tipo dentro do bloco do if.
11. Ativar verificações mais rígidas, que pegam mais erros.. O modo strict liga um conjunto de checagens que deixa o código mais seguro.
12. "b" | "c". Exclude retira de uma união os tipos indicados; sobram "b" e "c".
13. Deixa todos os campos de T só para leitura.. Com Readonly, tentar mudar qualquer campo do objeto gera erro de tipo.

Módulo 10. React e React Native

Gabarito:

1. Um pedaço reaproveitável da tela, escrito como função. Telas são montadas juntando componentes.
2. Guardar um valor que, ao mudar, atualiza a tela. Quando o state muda, o React desenha o componente de novo.
3. Props vêm de fora do componente; state é guardado dentro dele. Props são parâmetros recebidos. State é a memória do componente.
4. .tsx. Arquivos .tsx misturam TypeScript com JSX, a sintaxe que descreve a tela.
5. Uma sintaxe parecida com HTML escrita dentro do código JavaScript. O JSX permite descrever a interface com marcas parecidas com HTML dentro do JavaScript.
6. Botao. Componentes começam com letra maiúscula; assim o React sabe que não é uma tag comum.
7. Colocando entre chaves, como {nome}. As chaves {} dizem ao JSX para calcular um valor JavaScript naquele ponto.
8. Troca o valor de n e faz a tela ser desenhada de novo. A função de atualização muda o state e o React redesenha o componente com o novo valor.
9. A tela não é atualizada, porque o React não percebe a mudança. Só a função de atualização avisa o React de que precisa redesenhar a tela.
10. map. O map transforma cada item da lista em um pedaço de JSX.

11. Para o React identificar qual item mudou, entrou ou saiu. A key dá uma identidade a cada item e ajuda o React a atualizar a lista de forma correta.
12. Por props. Props são os dados que o pai passa ao filho, como atributos.
13. Uma prop do componente Botao. O que vem como atributo na tag chega ao componente como prop.
14. Renderização condicional: a tela muda conforme o estado. A tela muda conforme uma condição, como estar carregando ou não.
15. O efeito roda só uma vez, quando o componente aparece. Sem itens na lista de dependências, nada muda o efeito depois; ele roda só ao começar.
16. Buscar dados de uma API quando a tela abre. Buscas e outras ações de fora da tela são o uso típico do useEffect.
17. Text. No React Native todo texto precisa estar dentro de um componente Text.
18. View. View é o contêiner básico, parecido com a div.
19. Com objetos de estilo, por exemplo StyleSheet.create. O React Native não usa CSS de página; os estilos são objetos JavaScript.

Módulo 11. Expo: telas, navegação e estado

Gabarito:

1. Um conjunto de ferramentas que facilita criar apps React Native. O Expo cuida de muita configuração para você.
2. Criando um arquivo dentro da pasta app. Cada arquivo dentro de app vira uma rota (tela).
3. Abrir e testar seu app no celular durante o desenvolvimento. Ele roda seu projeto no celular sem precisar gerar um APK a cada teste.
4. Cria um projeto novo de app Expo já pronto para uso. Ele monta a pasta e os arquivos iniciais de um app Expo.
5. npx expo start. O expo start abre o servidor e mostra o QR code para testar no celular.
6. Abrir o app no celular pelo Expo Go. Você aponta a câmera para o código e o app abre no celular.
7. O arquivo index.tsx. O index representa a rota raiz da pasta.
8. /sobre. O nome do arquivo vira a rota, sem a pasta app e sem a extensão.
9. Define como as telas da pasta se organizam, em pilha ou abas. O layout envolve as telas da pasta e define a navegação entre elas.
10. Link. O Link recebe o destino em href e leva o usuário até aquela tela.
11. router.push('/perfil'). O router do Expo Router muda de tela dentro do código.
12. A tela de cima sai e a anterior aparece de novo. A pilha guarda as telas uma sobre a outra e voltar remove a do topo.
13. [id].tsx. Colchetes no nome indicam um trecho dinâmico da rota.
14. Um layout de abas (Tabs). O layout com Tabs coloca os botões das telas principais em uma barra embaixo.
15. Um menu lateral que desliza da borda da tela. O Drawer é o menu lateral escondido que abre ao deslizar ou tocar no ícone.
16. Compartilhar um dado entre várias telas sem props em cada nível. O contexto deixa um valor disponível para todos os componentes dentro dele.

17. Quando várias telas precisam ler ou mudar o mesmo dado. Se só um componente usa o dado, o state local basta; se várias telas usam, compartilhe.
18. Guardar imagens, ícones e fontes usados pelo app. Assets são os arquivos de mídia que o app usa.
19. Guarda as configurações, como nome, ícone e identificador. É o arquivo de configuração principal do app.

Módulo 12. Recursos do celular

Gabarito:

1. Pedir permissão ao usuário. Câmera, microfone e localização exigem permissão explícita.
2. No SecureStore. O SecureStore guarda os dados criptografados.
3. Porque o Expo Go não suporta push remoto nas versões recentes. Para push remoto, gere um build de desenvolvimento.
4. Para proteger a privacidade: o usuário decide se deixa ou não. Recursos sensíveis precisam da autorização do usuário.
5. O app não pode usar a câmera e deve avisar ou dar outra opção. Sem permissão o recurso fica bloqueado; o app precisa lidar com isso.
6. Texto, por isso objetos são convertidos com JSON.stringify. Ele guarda strings; para guardar um objeto, converta para texto antes.
7. Não, ficam guardados no celular para a próxima vez. O armazenamento local é persistente; é o que o torna útil para guardar configurações.
8. Usar await (ou then) para esperar o resultado. A leitura leva um tempo, então é preciso esperar a resposta.
9. O tema escolhido, claro ou escuro. Preferências simples podem ficar no AsyncStorage; dados sensíveis pedem armazenamento seguro.
10. Um aviso agendado e disparado pelo próprio app, sem servidor. A notificação local nasce dentro do celular, por exemplo para um lembrete.
11. Um lembrete diário às 8h para beber água. Lembretes que o próprio app agenda não precisam de servidor.
12. Um aviso de mensagem nova enviado por um servidor. A push vem de fora, enviada por um servidor para o aparelho.
13. Um token que identifica aquele aparelho. O servidor usa o token do aparelho para saber para quem enviar a notificação.

Módulo 13. Gerar APK/AAB e publicar na Play Store

Gabarito:

1. O pacote do app no formato que a Play Store exige para publicar. O APK serve para testar e instalar direto. A loja pede o AAB.
2. Sem a chave certa, atualizar o app do jeito normal fica difícil ou impossível. Faça cópias da chave e das senhas dela em mais de um lugar.
3. Teste fechado com 12 testadores por pelo menos 14 dias. As regras da Google mudam. Confira na Play Console.

4. O APK instala direto para testar; o AAB é o enviado à loja. A Play Store pede o AAB; o APK serve para instalar manualmente.
5. Um serviço que compila o app na nuvem. Ele gera o APK ou AAB nos servidores do Expo, sem precisar montar tudo no seu computador.
6. eas build. O comando eas build inicia a compilação na nuvem.
7. Identificar cada envio; o número precisa subir a cada envio. A loja só aceita um envio se o número de versão for maior que o anterior.
8. O identificador único do app na loja e no celular. Esse identificador não pode se repetir entre apps e não deve mudar depois de publicado.
9. Etapas para liberar o app a grupos pequenos antes da produção. Elas permitem testar com poucas pessoas antes de abrir a todos.
10. Liberar o app rápido para um grupo bem pequeno de pessoas. O teste interno é a faixa mais rápida e simples para poucas pessoas.

Módulo 14. Firebase

Gabarito:

1. Um banco de dados na nuvem do Firebase. Ele guarda documentos dentro de coleções.
2. Definir quem pode ler e escrever em cada dado. Sem regras bem feitas, qualquer pessoa pode mexer nos seus dados.
3. allow read, write: if true;. O if true libera tudo para todo mundo, até para quem não está logado.
4. E-mail e senha, e login com o Google. O Firebase Auth cuida do login por e-mail e senha ou por conta Google.
5. Para conferir se a pessoa digitou a senha igual nos dois campos. Os dois campos precisam coincidir, evitando um erro de digitação.
6. O próprio Firebase, sem que você precise guardar a senha. O Auth cuida do login e do armazenamento seguro das credenciais.
7. Em coleções que contêm documentos. Uma coleção, como clientes, guarda vários documentos.
8. Um registro com campos, como nome e idade. O documento guarda campos e valores de um item, como um cliente.
9. addDoc. O addDoc adiciona um documento novo a uma coleção.
10. Ler os documentos de uma coleção. Ele busca os documentos para você mostrar na tela.
11. Apaga um documento específico. Ela remove só o documento indicado.
12. allow read, write: if request.auth != null;. Quando request.auth é diferente de null, há um usuário logado.
13. Porque qualquer um poderia extrair a chave do app e ter acesso total. Um app no celular pode ser examinado; segredos de administrador ficam só no servidor.

Módulo 15. Supabase e SQL

Gabarito:

1. Regras por linha que dizem quem pode ver ou alterar cada linha da tabela. Com o RLS ligado, o banco confere a permissão em cada acesso.
2. SELECT. INSERT cria, UPDATE altera, DELETE apaga e SELECT lê.

3. A anon, protegida por RLS. A service_role passa por cima das regras e nunca pode ficar em app ou site.
4. Um registro, como um cliente guardado. Cada linha guarda um registro completo, e as colunas dizem quais informações ele tem.
5. Um tipo de informação, como nome ou e-mail. As colunas definem os campos de cada registro, como nome, e-mail ou preço.
6. Identificar cada linha de forma única. A chave primária (muitas vezes um id) nunca se repete, então permite achar exatamente uma linha.
7. Cria uma nova linha em clientes com o nome Ana. INSERT serve para adicionar um novo registro à tabela.
8. UPDATE. UPDATE modifica linhas existentes; SELECT só lê e INSERT cria novas.
9. Lê todas as colunas e linhas da tabela produtos. SELECT lê dados, e o asterisco significa todas as colunas.
10. DELETE. DELETE FROM tabela WHERE ... apaga as linhas escolhidas.
11. Ele apaga todas as linhas da tabela. Sem WHERE não há filtro, então o comando vale para todas as linhas.
12. Filtrar quais linhas o comando vai atingir. WHERE limita o comando às linhas que atendem à condição.
13. Alguém pode achá-la e ter acesso total ao banco. A service_role ignora as regras de segurança, por isso deve ficar só em lugar protegido, nunca no app.
14. Eles ficam bloqueados. Com RLS ativo, sem política nenhuma o acesso comum é negado, e você libera com políticas.
15. Deixar cada usuário ver só as próprias linhas. A política compara o usuário logado com o dono de cada linha e libera só o que for dele.
16. Confirmar quem é o usuário, como por e-mail e senha. Autenticação é o login: saber quem está usando o app.
17. PostgreSQL. O Supabase é construído em cima do PostgreSQL, então o SQL que você aprende vale para ele.
18. Relação entre tabelas. Em banco relacional, uma coluna pode guardar o id de outra tabela para ligar os dados.
19. SELECT * FROM clientes WHERE cidade = 'Itaquera';. SELECT com WHERE lê apenas as linhas que atendem à condição.

Primeira tabela

Alterar e apagar

Módulo 16. APIs, Node.js e Cloudflare Workers

Gabarito:

1. Um endereço que recebe pedidos e devolve dados para outros programas. As APIs ligam o app aos servidores.
2. Não encontrado. 200 é ok, 401 é falta de login, 404 é não encontrado e 500 é erro no servidor.
3. Rodar seu código no servidor sem manter uma máquina própria. O Worker roda perto do usuário e você paga pouco ou nada em uso baixo.
4. Um formato de texto para organizar dados. APIs costumam receber e devolver dados em JSON, que é fácil de ler por pessoas e programas.
5. POST. POST envia dados para criar algo; GET serve para buscar.
6. GET. GET pede informações sem alterar nada no servidor.
7. Uma forma de rodar JavaScript fora do navegador. O Node.js permite usar JavaScript para criar servidores e APIs.
8. Facilitar a criação de rotas e servidores de API. Express é uma biblioteca que simplifica criar rotas como /produtos.
9. 200. 200 significa sucesso; 404 é não encontrado e 500 é erro no servidor.
10. Você não está autenticado: falta login ou chave. 401 indica falta de autenticação válida.
11. Do servidor, que teve um erro interno. 500 indica falha dentro do servidor ao processar o pedido.
12. Porque qualquer pessoa pode enviar dados maliciosos. Nunca se deve confiar no que vem de fora; validar evita erros e ataques.
13. Um pequeno código que roda na nuvem da Cloudflare. Workers rodam seu código na rede da Cloudflare, sem você manter um servidor.

Módulo 17. Segurança para iniciantes

Gabarito:

1. Em variável de ambiente ou secret do serviço. O código vai para o repositório e pode ser lido por muita gente.
2. Revogar e criar outra, porque ela fica no histórico. Apagar o arquivo não apaga o histórico do Git.
3. Dizer ao Git quais arquivos não devem ser enviados, como o .env. Arquivos listados nele nunca entram nos commits.
4. Um arquivo com segredos, fora do código. O .env guarda valores como chaves de API, separados do código.
5. Porque o código pode ser visto por outras pessoas. Se o código vai para o GitHub, a chave vai junto e pode ser roubada.
6. O git o ignora e ele não é enviado ao GitHub. Arquivos no .gitignore não entram no repositório.
7. Porque ela continua no histórico, então é preciso trocá-la. O histórico guarda versões antigas, por isso a chave vazada deve ser revogada e substituída.
8. Usar uma senha diferente em cada serviço. Se uma senha vazar, as outras contas continuam seguras.
9. Pedir um segundo passo além da senha, como um código. Mesmo que roubem a senha, ainda falta o segundo passo para entrar.
10. Uma chave pública, como a anon do Supabase. Chaves públicas são pensadas para o app, enquanto as secretas ficam só no servidor.

Módulo 18. GitHub Actions

Gabarito:

1. Um conjunto de passos automáticos definido em um arquivo YAML. O workflow roda sozinho quando um evento acontece, como um push.
2. .github/workflows. O GitHub só lê os arquivos dessa pasta.
3. Rodar só quando necessário e juntar várias mudanças antes de enviar. Cada execução consome minutos do plano.
4. Um serviço que roda tarefas automáticas. Ele executa tarefas como testes e builds quando algo acontece no repositório.
5. Integração contínua, que testa o código a cada mudança. CI é juntar e testar mudanças com frequência e de forma automática.
6. Entrega contínua, que publica o resultado sozinho. CD leva a versão pronta para o ar sem passos manuais.
7. YAML. Workflows são arquivos YAML, como build.yml, dentro de .github/workflows.
8. Roda quando alguém envia código ao repositório. O gatilho define quando o workflow começa, e push é o envio de commits.
9. Guardar chaves e senhas do workflow sem expô-las no código. Secrets ficam cifrados e o workflow os lê quando precisa.
10. Um conjunto de passos executado em uma máquina. Um workflow tem um ou mais jobs, e cada job tem seus passos.
11. Uma tarefa individual, como instalar pacotes. Os steps rodam em ordem dentro do job.
12. Porque você escolhe quando gastar minutos e rodar o build. Com o gatilho manual, nada roda sozinho, então você controla o uso dos minutos.
13. Aparece como falho, com um X vermelho e o log do erro. O log da execução mostra qual passo falhou e a mensagem do erro.

Módulo 19. Vercel e Cloudflare Pages

Gabarito:

1. Publica a versão atual do site na internet. Deploy é colocar no ar.
2. Um novo deploy é feito automaticamente. Essa ligação com o Git é o jeito mais prático de publicar.
3. Um valor configurado no serviço (como uma chave) que não fica no código. Assim o segredo fica só no painel do serviço.
4. Um site feito de arquivos prontos, como HTML, CSS e JS. Sites estáticos são simples de hospedar em serviços como Vercel e Cloudflare Pages.
5. Você só envia o código, e eles cuidam da hospedagem. O serviço cuida da infraestrutura e você foca no site.
6. Um endereço de teste, antes de ir para o site principal. Previews deixam ver mudanças antes de publicar de verdade.
7. A nova versão não vai ao ar, e a anterior continua. O deploy só vai ao ar se o build der certo, e você vê o erro no log.

8. Nas variáveis de ambiente do serviço de hospedagem. Variáveis de ambiente ficam no painel do serviço, e não no código público.
9. O processo que transforma o código nos arquivos finais. O build gera os arquivos prontos que o serviço publica.
10. O repositório do GitHub. Com o repositório conectado, cada push dispara um novo deploy.

Módulo 20. Domínio, DNS e HTTPS

Gabarito:

1. Traduz o nome do site para o endereço do servidor. As pessoas digitam nomes e os computadores usam números.
2. A conexão com o site é criptografada. O cadeado protege o caminho dos dados, não garante a honestidade do site.
3. CNAME. MX é de e-mail e TXT serve para verificações.
4. O nome do site, como meusite.com.br. O domínio é o nome legível que aponta para o seu site.
5. Registrar domínios terminados em .br. O Registro.br é o órgão responsável pelos domínios .br.
6. Os servidores que dizem onde estão os registros DNS. Ao trocar os nameservers para a Cloudflare, é lá que você passa a gerenciar o DNS.
7. Dizer para onde vão os e-mails do domínio. O MX define qual servidor recebe os e-mails desse domínio.
8. Apontar um nome para um endereço IP. O registro A liga o nome do domínio a um endereço IP.
9. Guardar um texto, usado para provar que o domínio é seu. Serviços pedem um TXT para verificar que você é dono do domínio.
10. A mudança pode demorar para se espalhar pela internet. A propagação do DNS pode levar de minutos a horas, então vale esperar.

Módulo 21. API da Anthropic e prompts

Gabarito:

1. Porque qualquer pessoa poderia copiá-la e gastar seu dinheiro. Chame a API por um servidor seu, que guarda a chave.
2. Por uso (quantidade de texto enviado e recebido). Coloque limites para não ter surpresa na conta.
3. O texto de instrução e contexto enviado ao modelo. A qualidade da resposta depende de uma instrução clara.
4. Um pedaço de texto, como uma palavra ou parte dela, que o modelo lê. Modelos de linguagem dividem o texto em pedaços chamados tokens. A cobrança e os limites costumam ser contados neles.
5. Porque a IA responde de forma mais útil quando sabe a situação. Quanto mais claro o contexto, melhor a IA entende o que você precisa. Isso não elimina erros.
6. No chat você digita; na API o código envia o pedido e recebe a resposta. A API serve para o seu app ou servidor conversar com a IA por código, sem ninguém digitar no chat.
7. A chave da API fica guardada no servidor, longe de quem usa o app. Quem usa o app não vê o servidor, então a chave não vaza. Ela só aparece onde o seu servidor roda.

8. Para uma só pessoa não gastar todo o crédito e gerar uma conta alta. Como a cobrança é por uso, um limite por pessoa evita surpresas na conta.
9. "Responda em no máximo 3 frases, em português simples". Dizer tamanho, idioma e estilo orienta a IA a entregar a resposta no formato que você quer.
10. Em uma variável de ambiente, fora do código que vai para o repositório. Variáveis de ambiente mantêm segredos separados do código, que pode ser compartilhado ou publicado.
11. Como usar o Claude no chat; ela não ensina a chamar a API por código. A aula é sobre o uso do Claude no chat. A parte de API por código é explicada no texto do módulo.
12. Conferir em uma fonte confiável antes de usar. A IA pode errar ou inventar informações. Dados importantes precisam ser verificados.
13. Porque mais texto enviado e recebido significa mais tokens contados. A cobrança costuma acompanhar a quantidade de tokens usados, tanto na entrada quanto na saída.

Módulo 22. Bots de WhatsApp

Gabarito:

1. Um endereço seu que recebe avisos automáticos de outro serviço. Quando chega uma mensagem, a Meta chama o seu webhook.
2. Garantir que a mensagem veio mesmo da Meta. Sem a conferência, qualquer pessoa pode falsificar mensagens.
3. WhatsApp Business Platform (Cloud API da Meta). Soluções não oficiais podem causar bloqueio do número.
4. Envia os dados da mensagem ao webhook do seu servidor. O webhook é a porta por onde a Meta avisa o seu servidor que chegou uma mensagem.
5. Porque a Meta precisa conseguir chegar até ele para entregar as mensagens. Um endereço só do seu computador, como localhost, não é visto pela Meta.
6. No servidor, em variável de ambiente. O token dá acesso ao número do bot. Se vazar, outras pessoas podem enviar mensagens por ele.
7. Ter a permissão (consentimento) da pessoa para receber. As regras da plataforma e a privacidade pedem que a pessoa tenha aceitado receber suas mensagens.
8. Responder rápido com sucesso (código 200) e tratar o resto depois. Se a Meta não receber uma confirmação, pode achar que falhou e tentar enviar de novo.
9. Parar de enviar mensagens a ela e registrar o pedido. Respeitar o pedido de parar faz parte do consentimento e evita bloqueios.
10. Os passos de conta, número e verificação. A aula é o passo a passo da configuração. Ela não ensina a programar o bot.
11. Um formato de texto com pares de nome e valor, como {"nome": "Ana"}. JSON é um jeito simples de organizar dados que o código consegue ler com facilidade.
12. Segue as regras da plataforma e reduz o risco de o número ser bloqueado. Ferramentas fora das regras podem levar ao bloqueio do número. A API oficial é o caminho permitido.
13. Avisar que não sabe e passar a conversa para uma pessoa. Admitir o limite e encaminhar a um humano evita informação errada e cliente frustrado.

Módulo 23. Automação de navegador e tarefas agendadas

Gabarito:

1. Controlar um navegador por código para automatizar tarefas e testes. Ele clica, digita e lê páginas como uma pessoa faria.
2. Um agendamento que roda uma tarefa em horários definidos. Exemplo: todo dia às 9h.
3. Se os termos de uso do site permitem. Automação fora das regras pode bloquear sua conta.
4. Manda o navegador abrir esse endereço. goto significa "ir para": o navegador controlado pelo código navega até a página.
5. Executar todo dia às 9h. Os campos são minuto, hora, dia do mês, mês e dia da semana. Aqui: minuto 0, hora 9, todos os dias.
6. O minuto. A ordem é: minuto, hora, dia do mês, mês, dia da semana.
7. Um passo do fluxo, como um gatilho ou uma ação. O fluxo é feito de nós ligados entre si, cada um fazendo uma parte do trabalho.
8. O evento que inicia o fluxo, como um horário ou mensagem. Sem gatilho o fluxo não sabe quando começar. Ele pode ser um horário, um webhook ou outro evento.
9. Rodar sem mostrar a janela na tela. No modo headless o navegador funciona normalmente, só que sem janela visível. É comum em servidores.
10. O código procurava elementos pela posição ou nome antigos, que mudaram. Automações de navegador dependem da estrutura da página. Se ela muda, é preciso ajustar o código.
11. Pode sobrecarregar o site e fazer você ser bloqueado. Muitos pedidos rápidos parecem abuso. Respeitar limites e pausas evita bloqueios.
12. Usar a API, que é mais estável do que automatizar o navegador. APIs são feitas para código e mudam com aviso. Automação de tela quebra com mudanças visuais.
13. Que o computador esteja ligado e o programa possa ser executado. O agendador só executa se a máquina estiver ligada. Para rodar sempre, use um servidor.

Módulo 24. Segurança dos apps e sistemas

Gabarito:

1. A chave pública (anon/publishable). A chave pública só funciona dentro das regras do banco.
2. Não, o servidor também precisa conferir a permissão. Quem sabe mexer pode chamar o servidor sem passar pelo botão.
3. A conta continua protegida mesmo se a senha vazar. Só a senha não basta.
4. Uma lista das ameaças mais comuns de segurança em aplicações web. É uma referência para quem desenvolve saber quais falhas mais aparecem e como evitá-las.
5. Quando alguém digita comandos de banco de dados num campo e o sistema os executa. Se o sistema monta o comando com o texto do usuário sem cuidado, o atacante pode alterar o que o banco faz.
6. Qualquer pessoa pode ler a chave e usá-la. Segredos no código ficam visíveis a todos que veem o repositório.
7. Revogar a chave e gerar uma nova. Mesmo apagada, a chave pode já ter sido copiada. Só trocar por uma nova resolve.

8. Para definir quais linhas cada usuário pode ver ou alterar. Com RLS ligado, cada pessoa só acessa os dados que as regras permitem.
9. Porque o usuário pode burlar o formulário e falar direto com o servidor. A tela pode ser contornada. Só o que é conferido no servidor é realmente confiável.
10. Proteger os dados que viajam entre o usuário e o site. O HTTPS criptografa a conexão, dificultando que terceiros leiam senhas e dados no caminho.

Módulo 25. Privacidade, LGPD e políticas

Gabarito:

1. Dado como saúde, biometria, religião ou opinião política. Dados sensíveis pedem cuidado e base legal mais rígida.
2. Em uma página pública, com link no cadastro e na ficha do app. As pessoas precisam conseguir ler antes de se cadastrar.
3. Uma forma de excluir a conta e os dados, no app e por um endereço na web. A Play Console tem uma seção para informar isso.
4. A pessoa a quem os dados pessoais se referem. O titular é a pessoa natural dona dos dados. Os direitos da lei são dela.
5. Usar os dados só para o propósito que foi informado à pessoa. Se você coletou o e-mail para criar a conta, não deve usá-lo para outro fim sem avisar.
6. Livre e informado, com a pessoa sabendo para que os dados serão usados. A pessoa precisa entender e escolher aceitar, sem ser induzida.
7. Acessar, corrigir e pedir a exclusão dos seus dados. A lei garante que a pessoa possa consultar, corrigir e, em geral, pedir que seus dados sejam apagados.
8. A política explica o tratamento dos dados; os termos, as regras de uso. Cada documento responde a uma pergunta: o que você faz com meus dados, e quais as regras para usar o serviço.
9. Conceitos gerais da lei, em linguagem simples e sem foco em apps. São aulas de conceito geral. Servem de base para entender a lei antes de aplicar ao seu app.
10. Declarar esse uso de acordo com o que o app realmente faz. As informações precisam refletir o comportamento real do app, e a política de privacidade deve combinar com elas.

Módulo 26. AdSense, afiliados e Kiwify

Gabarito:

1. Conteúdo próprio e útil, política de privacidade e páginas básicas. Sites vazios ou copiados costumam ser recusados.
2. Um link com o seu código, que gera comissão nas vendas feitas por ele. Avise que o link é de afiliado, com transparência.
3. Vender produtos digitais e receber os pagamentos. Ela cuida de checkout, pagamento e entrega do produto digital.
4. O Google, com base nos anúncios exibidos no site. O AdSense mostra anúncios no site e o Google repassa ao dono uma parte do que os anunciantes pagam.
5. Isso é cliques inválidos e pode levar ao bloqueio da conta. Clicar nos próprios anúncios é considerado fraude e pode custar a sua conta.

6. Quando alguém compra usando o link dele e a venda é concluída. A comissão vem da venda feita pelo seu link, e as regras exatas variam por programa.
7. Por transparência: o leitor deve saber que você pode ganhar. Divulgar com honestidade gera confiança e é uma boa prática de mercado.
8. Um e-book em PDF. Produto digital é entregue pela internet, como e-book, curso ou modelo de planilha.
9. Gerar o link do produto e divulgá-lo. O afiliado não vende nem entrega: ele indica o produto com o seu link.
10. Dificilmente, pois o AdSense valoriza conteúdo próprio e útil. Conteúdo copiado não agrega valor e é motivo comum de reprovação.

Módulo 27. Vender para clientes locais

Gabarito:

1. Mostrar um exemplo pronto que resolve um problema dele. O dono quer saber o que muda no dia a dia dele.
2. Qual problema ele quer resolver e o que usa hoje. Entender o problema evita fazer a coisa errada.
3. Evita pedidos extras sem pagamento e mal-entendidos. Escopo claro protege você e o cliente.
4. Procurar e abordar possíveis clientes que não conhecem seu trabalho. Prospectar é ir atrás de novos clientes em vez de esperar que eles apareçam.
5. Curta, educada e falando de algo específico do negócio dele. Uma mensagem curta e personalizada mostra que você olhou o negócio e respeita o tempo da pessoa.
6. Agradecer e combinar quando você pode falar com ele de novo. Um retorno combinado e educado mantém a porta aberta sem pressionar.
7. Ele vê o resultado na prática e entende melhor o que vai receber. Ver um exemplo reduz a dúvida e deixa a conversa mais concreta.
8. Dando ideias de tipos de negócio e rascunhando mensagens, que você revisa. A IA é apoio para pensar e escrever. Revisar e adaptar o texto continua sendo seu trabalho.
9. Em mapas e buscas da sua região, vendo quem não tem site ou tem um fraco. Comércios do seu bairro são fáceis de visitar, e dá para ver pela busca quem tem presença online fraca.
10. O cliente paga todo mês por um serviço contínuo. Serviço recorrente traz renda mais previsível e mantém o contato com o cliente.

Módulo 28. Preço, contrato e cobrança

Gabarito:

1. O que será entregue, o prazo, o valor e a forma de pagamento. Escrever evita discussões depois.
2. Garantir o compromisso do cliente e cobrir o início do trabalho. O pagamento em etapas protege os dois lados.
3. Entender como declarar a renda e quando vale formalizar (como MEI). As regras variam. Converse com um contador.
4. Tempo de trabalho, custos e o valor que ele traz ao cliente. Preço bom cobre o seu tempo e os custos, e leva em conta o quanto o trabalho vale para quem recebe.

5. O que será feito e o que não será incluído. Definir o escopo evita que o cliente espere mais do que foi combinado.
6. Combinar por escrito um valor e prazo adicionais para esse pedido. Mudanças fora do escopo precisam de novo acordo para ninguém ser surpreendido.
7. Um recibo de pagamento a autônomo, que registra o serviço feito. O RPA documenta que um profissional autônomo prestou um serviço e foi pago por ele.
8. Para registrar o que foi combinado e pago e poder comprovar. Ter tudo registrado ajuda em dúvidas com o cliente e na hora de declarar a renda.
9. Registra o combinado e reduz mal-entendidos. Escrito, o acordo fica claro para os dois lados e pode ser consultado depois.
10. Parte do valor na entrada e o resto em momentos combinados. Dividir o pagamento em etapas dá segurança ao profissional e ao cliente.