Qual é o número?

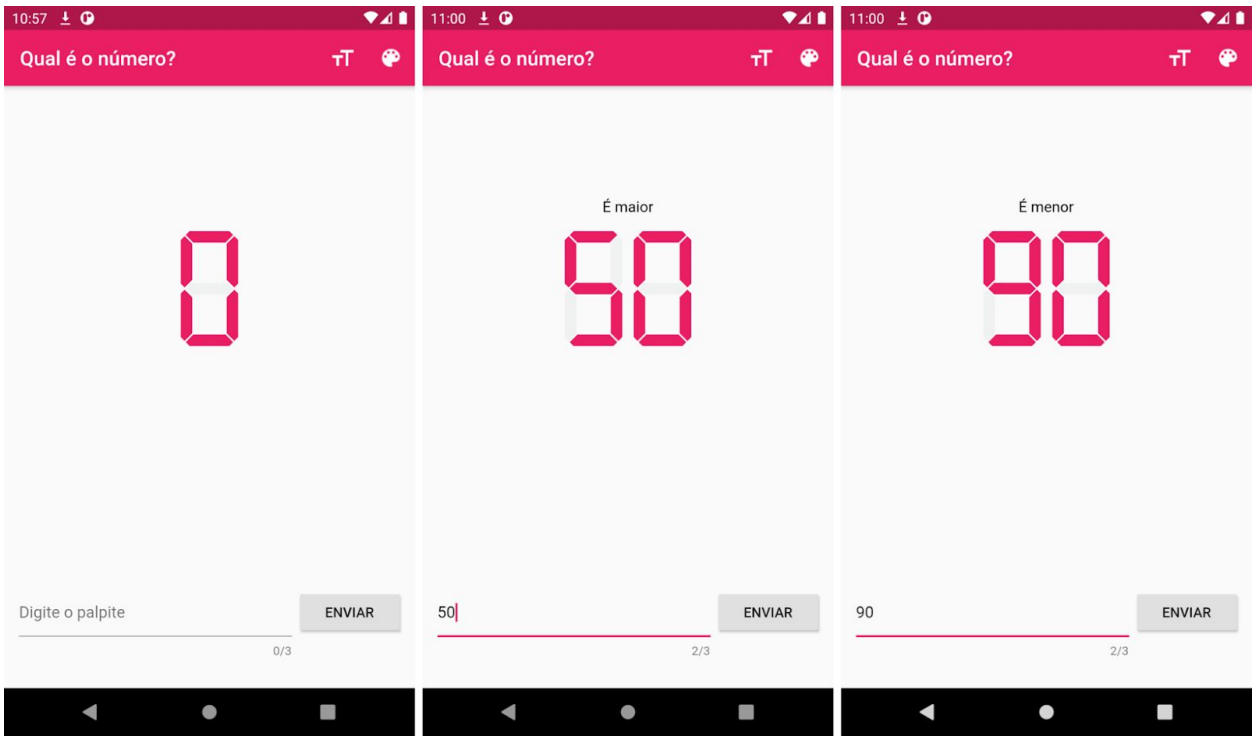
Visão geral

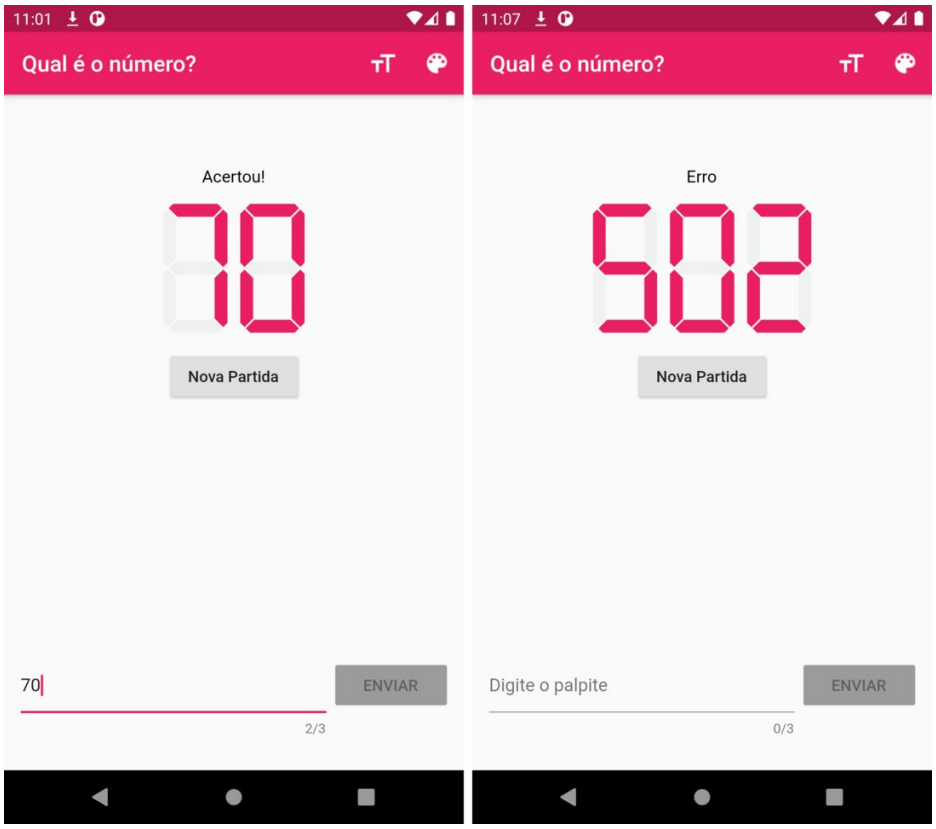
O problema consiste em receber um número através de uma requisição e implementar um jogo para acertar este número através de palpites. Ao errar um palpite, irá ser informado se o número obtido é maior ou menor do que o palpite feito. O palpite realizado ou *status code* de erro de requisição devem ser exibidos na tela no formato de LED de 7 segmentos. O palpite será obtido como entrada em um campo de texto, que deverá ser processado apenas quando o botão ENVIAR for clicado.

Para solucionar o problema proposto, você poderá escolher um dos seguintes caminhos:

1. Implementação nativa de um app para Android ou iOS. Neste caso, você deverá utilizar como linguagem de programação Kotlin, Java ou Swift.
2. Implementação híbrida usando Flutter com linguagem de programação Dart. Não serão aceitas demais frameworks, tais como React Native ou Xamarin.
3. Implementação utilizando o terminal como saída do seu programa. Este caso será melhor explicado em uma seção mais abaixo.

Layout - Android e iOS





Especificação

Você deverá enviar uma requisição para receber um valor aleatório utilizando o endpoint abaixo, com parâmetros especificando o limite inferior e superior. Para o modo padrão de jogo, utilize o limite inferior como 1 e o limite superior como 300, conforme a URL abaixo:

```
https://us-central1-ss-devops.cloudfunctions.net/rand?min=1&max=300
```

A resposta estará no formato JSON com o valor aleatório especificado no campo `value`:

```
{
      "value": 15
}
```

Esteja preparado para eventuais falhas: ausência de parâmetros ou parâmetros incorretos irão causar falha na requisição. Além disso, há uma pequena chance desta requisição retornar com erro mesmo com parâmetros corretos. O *status code* em casos de falha deverá ser mostrado no LED e pode ser obtido em métodos específicos da tecnologia

utilizada ou obtido do JSON respondido em casos de falha, no campo `StatusCode`:

```
{
	"Error": "Bad Gateway",
	"StatusCode": 502
}
```

Você deverá informar em um texto acima do LED os resultados possíveis:

- "Erro": quando houver erro na requisição
- "É menor": quando o palpite enviado é **maior** que o número obtido
- "É maior": quando o palpite enviado é **menor** que o número obtido
- "Acertou!": quando o palpite enviado é igual ao número obtido

Além disso, deverá existir na tela um botão NOVA PARTIDA, cujo clique deverá criar uma nova partida obtendo um novo número aleatório (o que implica em fazer uma nova requisição).

É importante mencionar que o número que o usuário deve tentar adivinhar DEVE impreterivelmente ser obtido por meio da requisição descrita anteriormente. O objetivo do problema envolve testar a sua habilidade de lidar com requisições. Portanto gerar um número aleatório dentro da própria aplicação foge do propósito desta prova.

Implementação Android e iOS

Segmentos:

- O display pode conter números não-negativos de 1 a 3 algarismos.
- O valor numérico exibido nos segmentos deve representar o palpite realizado ou o status code obtido quando a requisição falhar.
- Só deve ser apresentada a quantidade de algarismos necessária (O número 29 precisar ser exibido sem zero à esquerda).
- Você deve implementar sua própria solução para exibir os segmentos. Não serão aceitas bibliotecas de terceiros para esta etapa. Em especial, **NÃO** utilize uma fonte pronta para exibir os segmentos.
- Cada segmento pode ser representado como um retângulo ao invés dos hexágonos mostrados nos layouts de exemplo.

Botão *NOVA PARTIDA*:

- O botão ficará visível apenas quando houve erro ao receber o número ou quando o jogador acertou o palpite

Campo de entrada:

- Mostrar o texto "Digite o palpite" como *hint text* (texto que aparece enquanto o campo de texto estiver vazio).

- O valor deve aparecer nos segmentos assim que o botão *ENVIAR* ao lado direito do campo de texto for clicado.
- O botão de enviar deverá ficar desabilitado quando houve erro ao receber o número ou quando o jogador acertou o palpite. O usuário deve clicar em "NOVA PARTIDA" neste caso.

Implementação via terminal

Alternativamente, você pode criar sua solução sem lidar com Android ou iOS, caso você não esteja familiarizado com a programação para estas plataformas. Neste caso, você pode utilizar as linguagens Kotlin, Java, Swift, C# ou C / C++.

Comentários e documentação

Você DEVE documentar qual a lógica utilizada no funcionamento do programa. Se alguma estrutura de dados especial tiver sido utilizada e for relevante para o funcionamento do algoritmo, citá-la também.

Exemplo:

```
/**
 * A lógica do programa se baseia em (...)
 * Para algarismo do display, existe uma estrutura
 * correspondente (...)
 */
```

Critérios de avaliação

- (1 pt) Compila e executa sem crashes
- (1 pt) Descrição do funcionamento do programa (deve condizer com o que foi implementado)
- (2 pts) Implementação da requisição e tratamento correto dos possíveis status
- (1 pt) Tratamento correto do input do usuário
- (6 pts) Funcionamento correto e robustez da solução
- (1 pt) Controle de estado do botão NOVA PARTIDA
- (4 pts) Layout condizente com os exemplos (Máximo de 2 pts se a implementação tiver sido feita via terminal)
- (4 pts) Organização e clareza do código

Pontos extras

As funcionalidades a seguir **NÃO SÃO OBRIGATÓRIAS** para que seu programa seja avaliado, mas contarão como pontos extras para avaliarmos seu nível técnico de programação. Fique à vontade para escolher um ou mais itens abaixo para usar na sua implementação.

★ **Tamanho do segmento de LED (2 pts)**:

- Você deve dar suporte a no mínimo 5 tamanhos distintos para os números de LED.
- Android e iOS: Exibir um *slider* quando o ícone de tamanho de texto for clicado.
- Via terminal: receba um input do usuário no início do programa definindo o tamanho da fonte.

★ **Cores nos leds (2 pts)**:

- Você pode utilizar qualquer biblioteca disponível na Internet para solução de paleta de cores.
- A paleta de cores precisa aparecer assim que o ícone correspondente for clicado.
- Os segmentos devem ser atualizados assim que uma cor for tocada na paleta de cores.
- O componente da paleta deve ser fechado quando qualquer área fora da paleta for clicada.
- Via terminal: receba um input do usuário no formato RGB definindo a cor da fonte dos LEDs a serem exibidos. Para implementações feitas via terminal, apenas 1 ponto extra poderá ser alcançado.

As funcionalidades devem ser acessadas por ícones dispostos no topo da página, sendo o primeiro relacionado com o tamanho dos segmentos de LED e o segundo com a cor.

Você pode usar os ícones do Material Design para seus ícones:

- Tamanho da fonte
- Paleta de cores

Vale ressaltar: Para a implementação da sua solução você poderá utilizar bibliotecas oficiais do Android, da Apple e da linguagem de programação que você tiver escolhido. Isso inclui bibliotecas que te auxiliem a realizar as requisições HTTP, decodificar a resposta em JSON, lidar com paleta de cores ou lidar com entrada e saída do terminal. Você não deve utilizar qualquer outra biblioteca de terceiros: ao invés disso, construa sua própria solução. Em

especial, implemente você mesmo a lógica relacionada com o LED de sete segmentos, sem utilizar nenhuma biblioteca que porventura o faça.

Total: 24 pontos, sendo 4 pontos extras - Boa prova =)