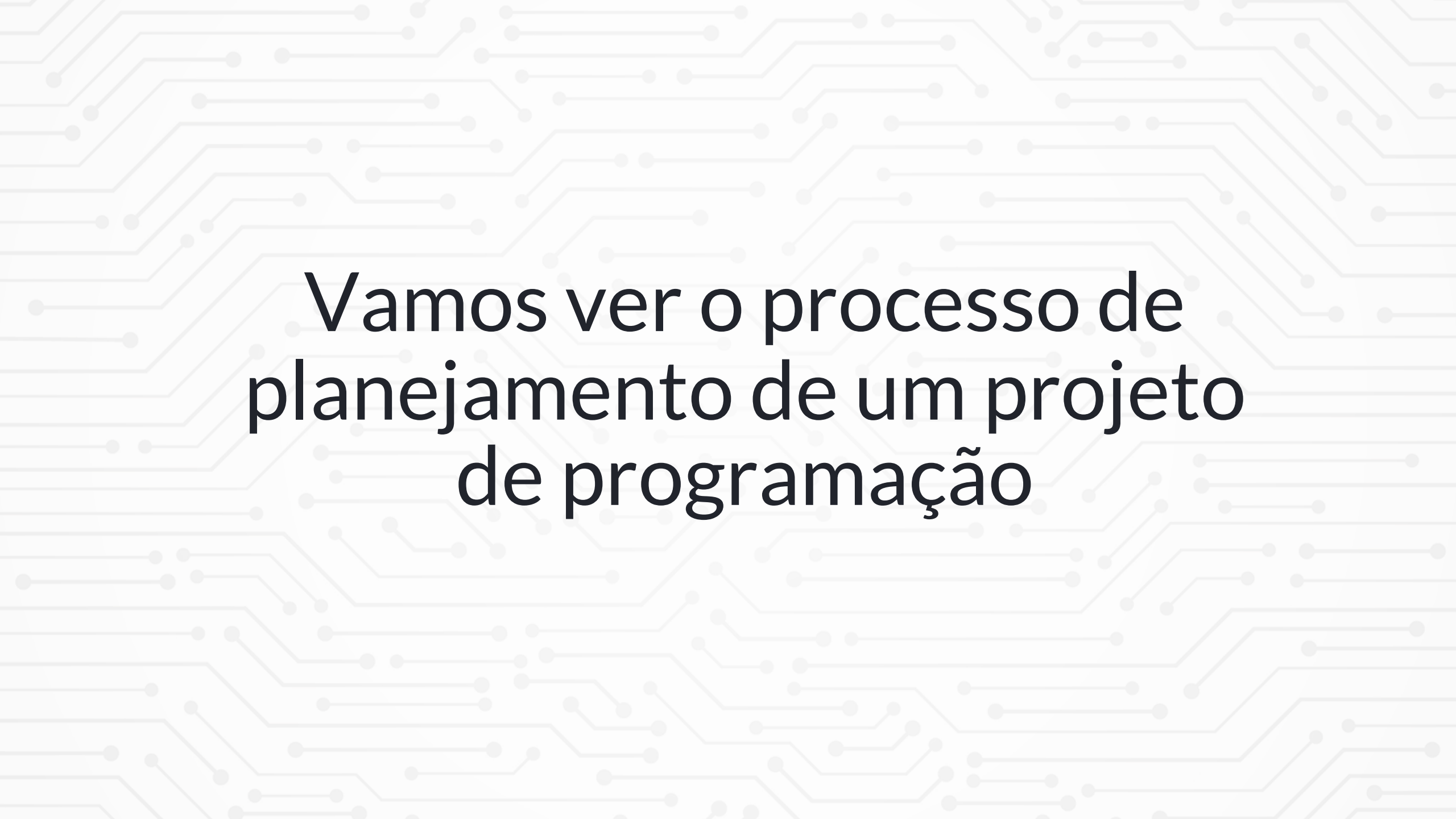




# **Como planejar e construir um projeto de programação**



Vamos ver o processo de  
planejamento de um projeto  
de programação



# **Etapa 1: Definir o projeto**

# O que se quer criar?

Devem começar por responder a estas perguntas:

Qual é o projeto?

O que é o MVP (Produto Viável Mínimo)?

Quais são as coisas boas de se ter o projeto?

Quando o projeto será concluído?

# O que se quer criar?

Se ainda não tem uma ideia, fica aqui algumas questões para ajudar-vos a pensar/ a criar algo:

- Qual é o teu jogo favorito - fliperama, jogo de tabuleiro, jogo de desporto? Vocês poderiam fazer uma versão digital simplificada disso? Vocês poderiam mexer um pouco, como dar-lhe um tema diferente ou outros personagens principais?
- Quais são os outros campos académicos favoritos? Se vocês amam arte, vocês poderiam fazer um programa que produz arte? Se vocês amam história, que tal uma linha do tempo interativa? Se vocês amam ciência, que tal uma simulação científica?
- Qual é o filme ou programa de TV favorito? Vocês poderiam fazer uma versão digital de uma cena ou personagem dele? Talvez fazer um jogo baseado nele?
- Qual é o aparelho da vida real que vocês amam? Vocês poderiam fazer uma simulação dele?

# O que se quer criar?

## Exemplo de definição de projeto

Para dar um exemplo simples, digamos que vamos construir um aplicativo de calculadora, teríamos uma definição básica de projeto como vem a seguir.

## **Projeto de aplicativo de calculadora**

**Qual é o projeto?** – O projeto da calculadora é um projeto para construir uma calculadora que pode ser acessada num navegador da web. Isso permitirá que os utilizadores insiram números e calculem os resultados desses números com base na operação aritmética que escolherem.

**Qual é o MVP?** – O produto mínimo viável é uma calculadora que roda num navegador da web que pode realizar operações de adição, subtração, multiplicação e divisão com base na entrada do utilizador e mostrar ao utilizador o resultado dessa equação.

**Quais são as coisas boas de se ter?** – O que é interessante neste projeto é estilizar a calculadora, usar os pressionamentos do teclado como entrada, não apenas os utilizadores clicarem em botões e adicionar operações de ordem superior como 'à potência de x'.

**Quando o projeto estará concluído?** – O projeto estará concluído assim que todos os recursos do MVP forem implementados e a calculadora tiver sido estilizada.



A definição acima é simples e direta.

Se alguém o encontrasse, entenderia do que se trata o projeto.

Entenderia porque diz a vocês o que o projeto é, os recursos do MVP que vocês deve construir, recursos bons de se inserir e quando estará completo.

Ao definir o projeto, vocês o tornam menos intimidante.

Depois de ter a definição do projeto, vocês podem começar a próxima etapa.



# Qual tecnologia vocês vão usar?

Nessa etapa, vocês precisam considerar quais as tecnologias (linguagens/bibliotecas/ambientes) vocês conhecem ou podem aprender facilmente, e quais delas são mais úteis para o trabalho.

Para muitos de vocês, esta lista pode ter apenas um item, "1. JS + Processing JS", e isso torna a decisão fácil.

Se vocês conhecem outras linguagens/ambientes (como JS+HTML, Python, SCRATCH, Swift, etc) e estão a pensar em fazer algo que não tenha muito sentido com Processing JS, então considerem quais destas tecnologias seria a melhor para o programa.

Se vocês quiserem criar essas coisas mas não conhecem outras tecnologias, vocês devem ter uma nova ideia para o programa.

Vocês *podem* aprender uma nova tecnologia para um novo projeto, mas dado que estão no curso de programação nada melhor que agora! Ou pelo menos aprofundá-la

Neste exemplo suponhamos que o projeto vai ser resolvido usando HTML, CSS e JavaScript.

# Quais recursos serão incluídos ?

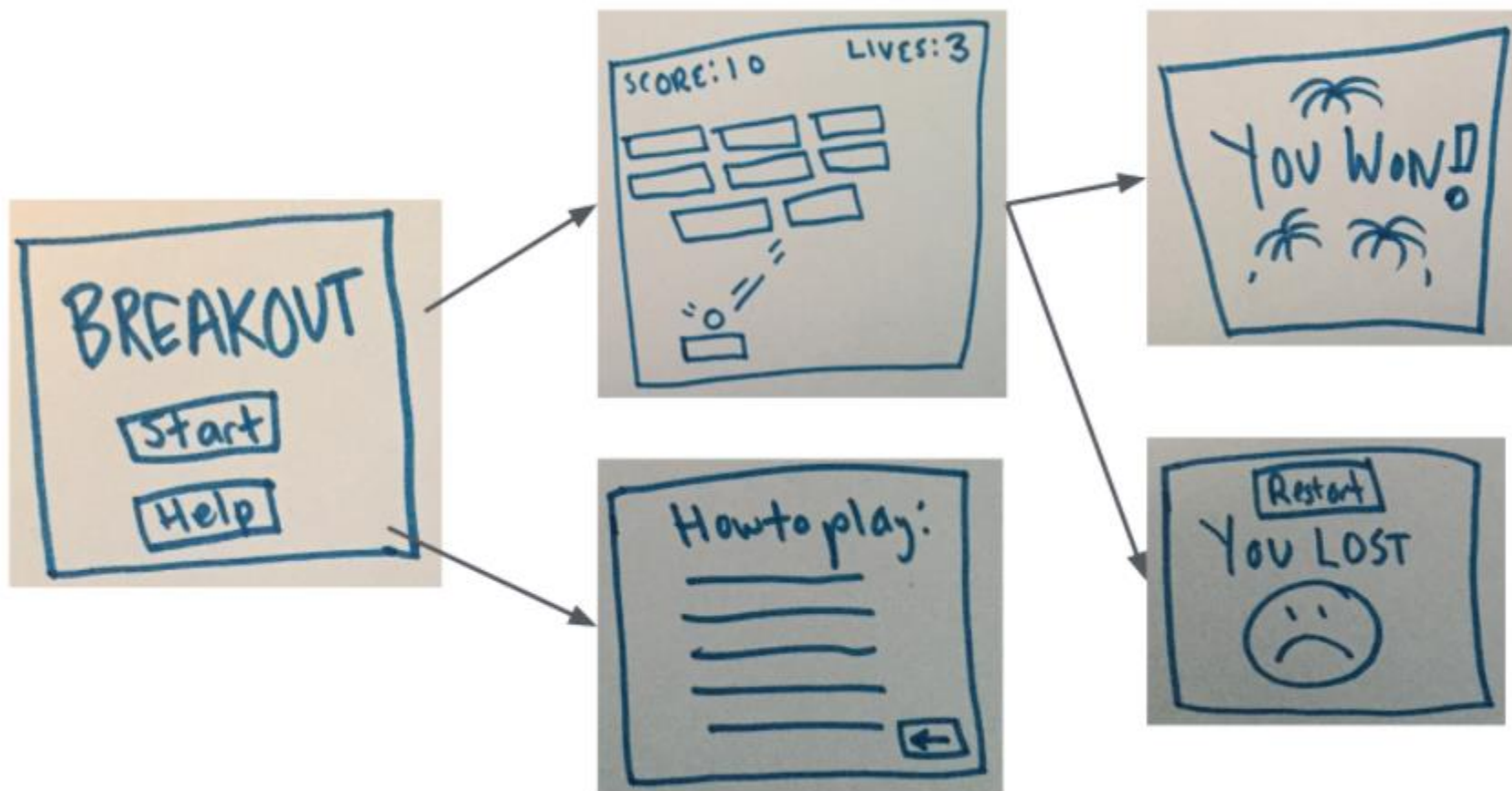
É aqui que iniciamos o verdadeiro planejamento, e onde (eu acho) fica divertido.

O objetivo nesta etapa é descobrir o que vocês realmente vão fazer- qual o visual do programa, quais recursos serão incluídos, quais *não* serão.

A primeira coisa que vocês podem fazer são as "maquetes" - esboços que se parecem com o que vocês estão a fazer, mas sem detalhes como cores ou tamanhos exatos.

Vocês podem fazer maquetes, no papel ou com programas online.

Para te dar uma ideia de como uma maquete se parece.



# Quais recursos serão incluídos ?

Agora vocês podem usar essas maquetes para ajudar a fazer a lista de recursos, onde vocês pensam em todos os recursos do programa, e fazer uma lista deles.

# Mas quais os recursos que devem ser incluídos?

Se tivéssemos tempo infinito para criar todos os programas que temos na cabeça, então todos os recursos deveriam ser inclusos na nossa lista.

Mas não temos, então nesta etapa, vocês devem decidir quais recursos são mais importantes e quais vocês farão se tiverem tempo.

Isso também ajudará a saber em qual ordem os recursos devem ser implementados, do mais para o menos importante.

Para ajudá-lo a descobrir a importância de cada recurso, faça essas perguntas a si mesmo:

- Se eu compartilhar esse programa com um amigo, quais recursos eu quero garantir que estejam funcionando?
- Quais recursos eu estou mais animado para criar?
- Quais recursos são os mais originais para o meu programa?
- Com quais recursos eu vou aprender mais durante a implementação?
- Há algum recurso que parece estar muito além das minhas habilidades atuais?



## **Etapa 2: Criação do fluxo de trabalho**

A próxima etapa é a mais simples. Normalmente, esta etapa pode ser combinada com a etapa 3.

Mas, por enquanto, vamos examiná-la aqui como uma etapa separada para que eu possa mostrar-vos como configurar um fluxo de trabalho muito básico para vossos próprios projetos.

Depois de fazer isso uma vez, pode ser uma etapa padrão para o restante de vossos projetos.



Em seguida, veja sua lista de recursos da etapa anterior, e ou ordene a lista ou adicione uma classificação para cada recurso.

Podíamos, colocar por exemplo, "P1", "P2", e "P3" ao lado dos recursos, significando maior prioridade (P1), prioridade média (P2) e baixa prioridade (P3).

(Agora imaginem que é um jogo ao invés de uma calculadora, fica mais fácil)



### **(P1) Cena do jogo**

- (P1) Paddle controlado pelo utilizador
- (P1) Múltiplos tijolos coloridos
- (P1) Movimento angular da bola
- (P1) Detecção de colisão
- (P2) Mostrador de vidas
- (P2) Mostrador de pontos
- (P3) Efeitos sonoros

### **(P2) Cena Principal**

- (P2) Botão jogar
- (P3) Botão ajuda

### **(P3) Cena de ajuda**

- (P3) Texto
- (P3) Botão voltar

### **(P2) Cena de vitória**

- (P2) Título
- (P3) Animação de fogos de artifício

### **(P2) Cena de derrota**

- (P2) Texto
- (P3) Botão reiniciar

# Como vocês vão fazer a implementação?

Vocês sabem que têm uma ideia de quais recursos fará primeiro no programa - mas se vocês iniciarem agora, iniciará um programa do zero sem nenhum código escrito e isso pode ser intimidador.

Quais as variáveis vocês deve escrever primeiro? Quais as funções?

Uma forma de resolver isso é pensar sobre o "alto nível de arquitetura" do programa - separando-o em categorias como "objetos", "lógica", "interação do usuário", "dados do usuário", e "cenas" - e então pensar em como implementá-las, como tipos de orientação a objetos, funções ou variáveis.

# Como vocês vão fazer a implementação?

## Objetos

- Tijolo(.eAtingido())
- Paddle(.move())
- Bola(.move())

## Cenas

- Início
  - Jogo
  - Final

## Lógica

- Colisão bola-tijolo (função, usar caixa delimitadora)
- Angulação paddle-bola (função, inverter ângulo)

## Interação do usuário

- Movimento teclado-paddle (teclaPressionada)
- Botões para mudanças de cena (mouseClicado)

## Dados do usuário

- Mortes da bola (array)
- Batidas da bola (array)

# Como vocês vão fazer a implementação?

Quando vocês tiverem a arquitetura de alto nível, deve ficar mais claro o que vocês podem implementar primeiro.

Vocês podem decidir escrever todo programa em pseudo-código primeiro, o qual falaremos a respeito depois.

Basicamente, significaria escrever todo o programa na língua nativa dentro de um comentário e depois ir transformando-o em código de verdade.

# Qual é o vosso cronograma?

Quanto tempo vocês tem para desenvolver esse programa?

Quantas semanas e quanto tempo por dia?

Quais recursos vocês escreverão em cada semana?

O objetivo nesta etapa é saber o tempo máximo que demora para a construção do programa - o que é muito importante se vocês tiverem um prazo, mas também muito útil para saber quanto leva para escrever um programa.

# Qual é o vosso cronograma?

Aqui está um exemplo de cronograma, assumindo 2-4 horas de trabalho por semana:

- Semana 1: Design and pseudo-código
- Semana 2: Visual rústico
- Semana 3: movimento da bola/mecânicas de colisão
- Semana 4: Mecanismos de pontuação
- Semana 5: cenas (Início/vitória/derrota)
- Semana 6: poliment, testes manuais (QA), Preparação da demo

Preparar um cronograma para projetos de programação é difícil.

Algumas coisas que parecem fácil levam mais tempo do que vocês esperam (como algum erro em que vocês passam horas a corrigir), algumas coisas que parecem difíceis levam menos tempo do que vocês esperam. Como regra geral, assuma que vai levar mais tempo do que vocês pensam, e ajuste conforme vão fazendo.

# Criação de um cronograma e Orgonograma

PLANILHA EXCEL | TRELLO | ENTRE OUTRAS

Devem conter estados tais como:

1.PENDENTE (“TODO”, em inglês)

2.A FAZER (“DOING”)

3.FEITO (“DONE”)

4.ERROS / NÃO TENHO CERTEZA DE COMO FAZER (“BUGS/NOT SURE”)

# Criação de um cronograma e Orgonograma

PLANILHA EXCEL | TRELLO | ENTRE OUTRAS

Dentro das colunas, vamos adicionar infos.

Conforme trabalhamos nas secções, nós alteramos o estado desse ponto, para A FAZER e, assim que terminarmos com esse ponto, podemos mudar para FEITO. Se vocês tem um bug no qual está travado ou não tem certeza de como fazer algo, podemos colocar com o estado ERROS.

Agora que configuramos nosso fluxo de trabalho, podemos ir para a última etapa.





## **Etapa 3: dividir o projeto em componentes menores**

A chave para construir os próprios projetos começa com a divisão do grande projeto em componentes menores e menos intimidantes.

Esses componentes menores são o que formam os pontos a partir da etapa 2.

Agora, isso parece bastante simples

Um bom desenvolvedor dividirá o projeto em tarefas menores.

No entanto, como podem dividi-lo?

Bem, a primeira coisa que vocês precisam fazer é olhar para a definição do projeto e então dividi-lo em partes menores.

Vamos continuar a usar o exemplo do aplicativo de calculadora para fazer os nosso pontos:

Funções de cálculo – MVP

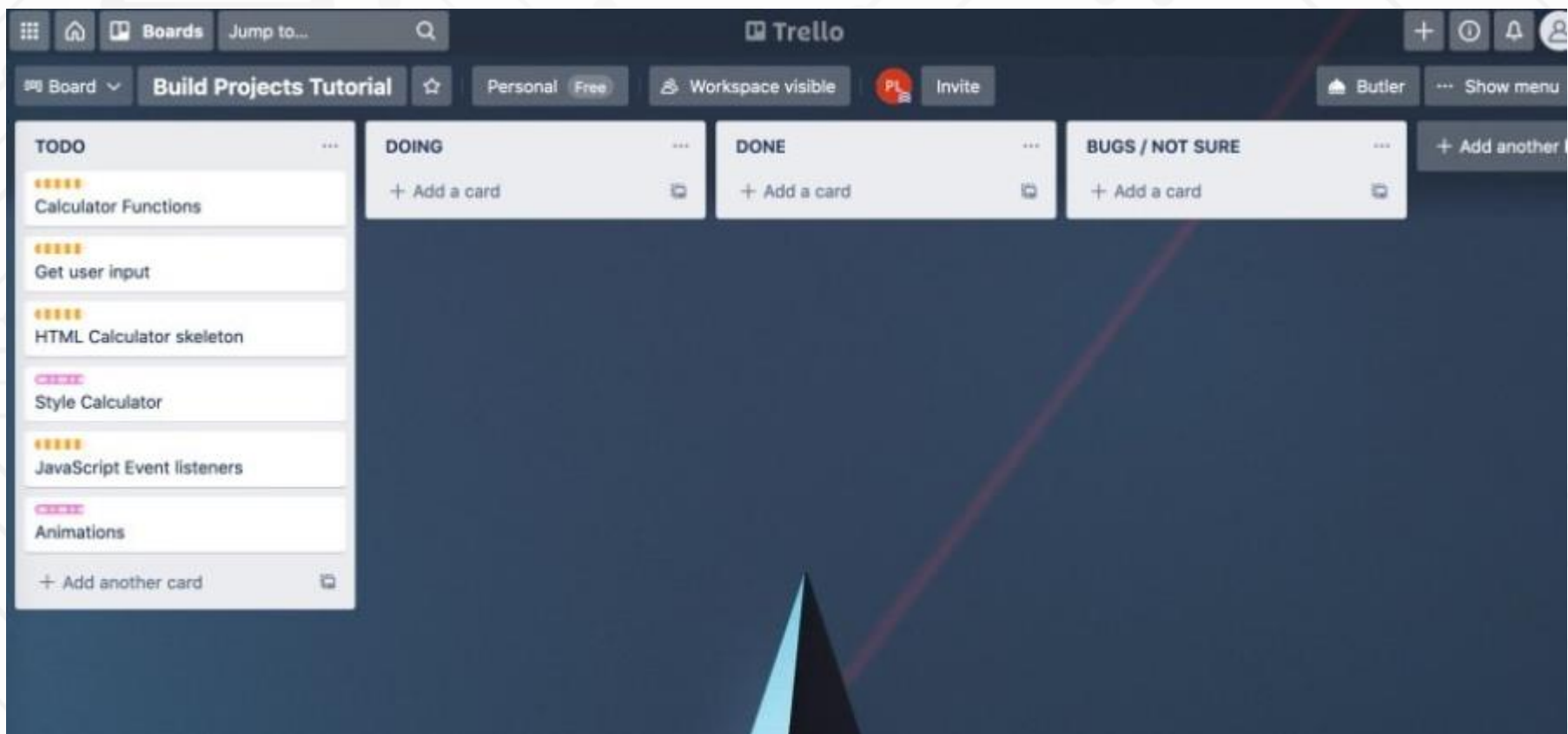
Obtenha a opinião do utilizador – MVP

Interface de usuário HTML – MVP

Interface de utilizador de estilo – Jogada

*Listeners* de eventos JavaScript – MVP

Adicionar animações para cálculos – jogada



Vocês notaram que, para cada tópico, tem um rótulo de MVP ou Jogada para ajudar visualmente a ver quais os pontos que são os mais importantes e, portanto, aqueles com que trabalhar primeiro.

O maior benefício dos pontos/tópicos é que eles simplificaram o que já temos que fazer.

Isso torna os projetos menos intimidantes, pois vocês não estão a fazer um aplicativo de calculadora grande e assustador, mas sim 6 projetos menores que se combinam para criar um grande projeto.

Ao trabalhar em ponto/tópicos, e mudar o seu estado para A FAZER. Levem o tempo que precisarem para fazer o componente funcionar antes de passar para a próxima coluna.

Mas ainda não terminamos, podemos simplificar e melhorar nosso fluxo de trabalho ainda mais para garantir que não fiquemos bloqueados pelo tamanho do projeto durante a construção.

## **Dividam cada componente em listas de verificação menores**

Assim que tivermos os pontos/tópicos de alto nível, podemos dividir esses componentes novamente em tarefas menores, dividindo essas tarefas em listas de verificação para que possamos acompanhar nosso progresso.

O exemplo abaixo mostra como poderá funcionar, então vocês pode dividi-lo em itens menores ou maiores, dependendo do que funciona para vocês.

Vamos usar o tópico de funções de cálculo como um exemplo de como decompor ainda mais um componente.

Como a tarefa é MVP e se definiu o MVP como cálculos básicos de adição, subtração, multiplicação e divisão, precisamos adicionar essas funções à lista de verificação.

## Calculator Functions

in list [TODO](#)

LABELS



### Description

Add a more detailed description...

### Checklist

Delete

0%

- ☐ Addition calculation
- ☐ Subtraction calculation
- ☐ Multiply calculation
- ☐ Divide Calculation

Add an item

Add



[Assign](#)

[Due date](#)



### Activity

Show details



Write a comment...

ADD TO CARD

Members

Labels

Checklist

Due date

Attachment

Cover

POWER-UPS

+ Add Power-Ups

Get unlimited Power-Ups,  
plus much more.

[Upgrade Workspace](#)

BUTLER

+ Add button

ACTIONS

Move

Copy

Make template

Watch

Archive

Share

Agora dividimos o nosso tópico de funções de cálculo em 4 pequenos projetos nos quais podemos trabalhar.

Isto é muito mais fácil do que a tarefa abstrata e assustadora de criar um aplicativo de calculadora ou mesmo escrever as funções da calculadora (tópico).

Agora podemos concentrar-nos e descobrir como fazer cada uma dessas funções.

Ao fazermos isso, verificamos esses itens, o que nos dá uma sensação de realização e progresso.

Então, uma vez que tenhamos feito todos esses quatro itens, podemos mudar o estado do tópico concluído e começar a ler o próximo tópico.

A partir daqui, só precisamos repetir o processo para cada tópico. Então vocês estarão prontos para construir projetos incríveis.



# Comecem a construir o projeto de programação

A chave é definir claramente o projeto, configurar seu fluxo de trabalho e, em seguida, dividir o projeto em componentes menores, todos construídos para a criação de um projeto maior.

Ao fazer isso, o projeto não parece uma montanha enorme para escalar; em vez disso, deve parecer mais uma escada com cada degrau ajudando você a alcançar seu objetivo.

O planejamento é ótimo, mas a chave é então começar e construir.