



# MANUAL QA → AUTOMATED QA

Hacer más trabajo  
trabajando menos

# ¿De qué vamos a hablar?

1. Contexto: bases de QA
2. Guía para el cambio a más automatización
3. Hojas de ruta
  - a. QA
  - b. Programming
  - c. Designer/Artist
  - d. Publisher/Project Lead



# ¿Quién soy yo?

- Nepo (elle/they)
- QA (8 años), freelance (1 año 🧐)
- "Configuración de CI para jams"
- Nokorpo: consultoría, juegos por encargo



**Parte 1**

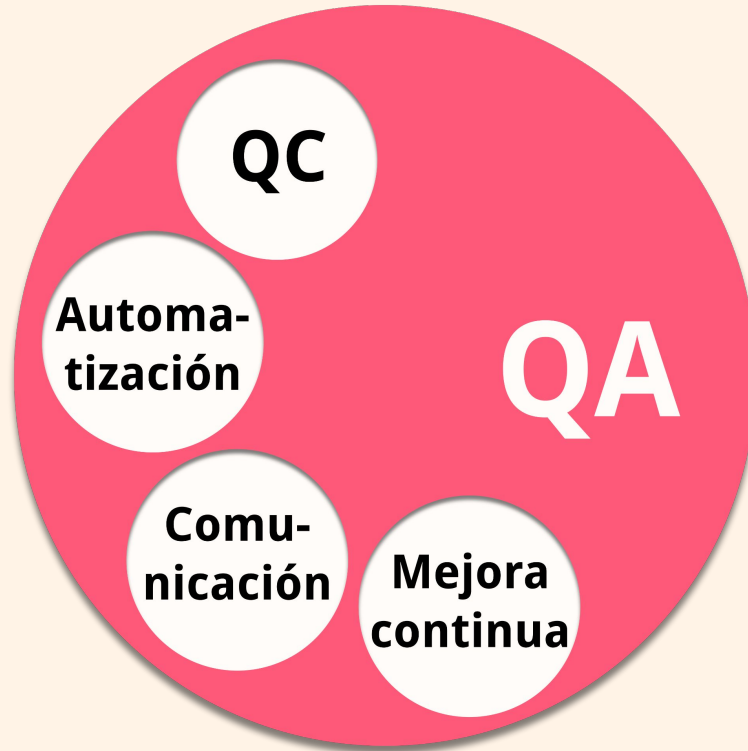
**QA resumido en 5 mins**



**QA  $\neq$  QC**



**QA ≠ QC**



# ¿Qué es QA?

**Ideación**

**Preproducción**

**Producción**

**Postproducción**

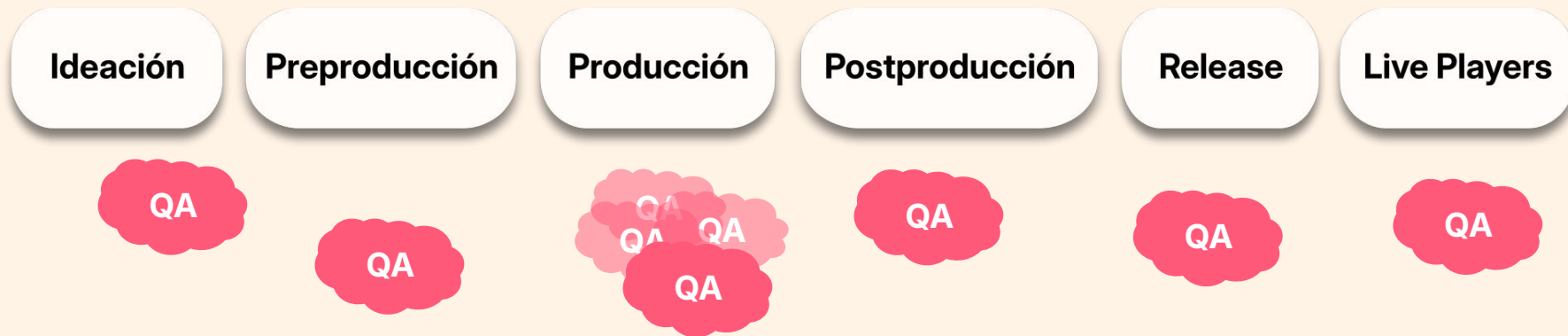
**Release**

**Live Players**

**QA**



# ¿Qué es QA?

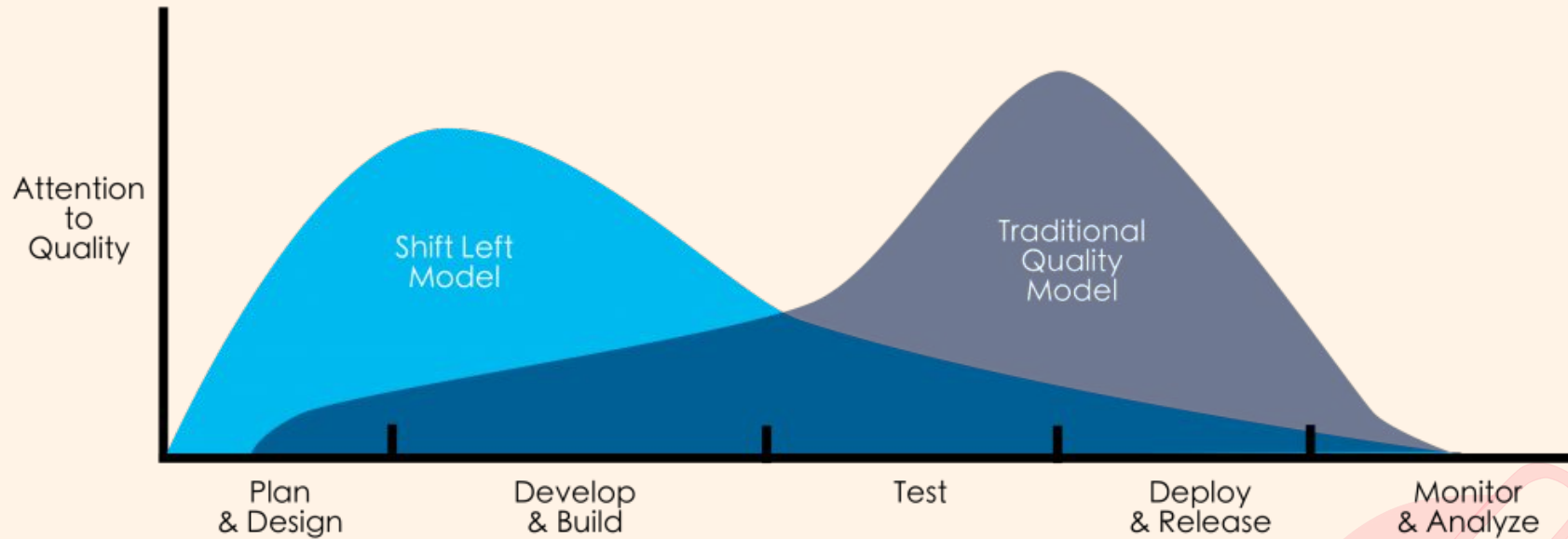


QA trabaja sobre **procesos** del equipo que se dan a lo largo del desarrollo



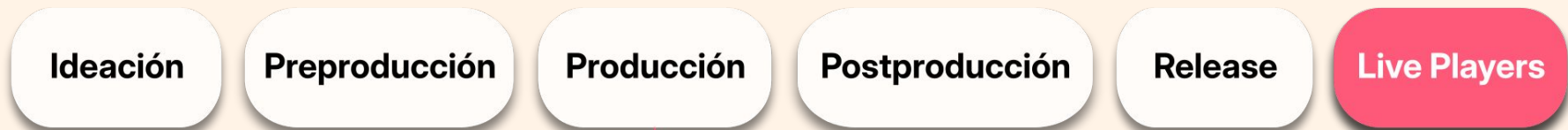


# Shift left testing



*"It's 2017: Test automation is not optional when building mobile apps!"* — Geert van der Cruisen

# Shift left testing



- Investigar
- Arreglar

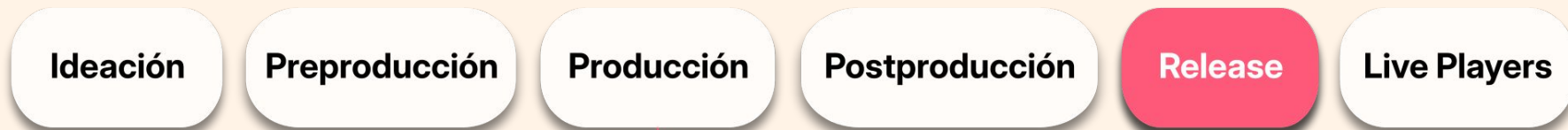
- Verificar que funciona

- Build
- Subir nueva versión

- Descargar
- Jugar
- Reportar error



# Estrategia de QA: objetivo



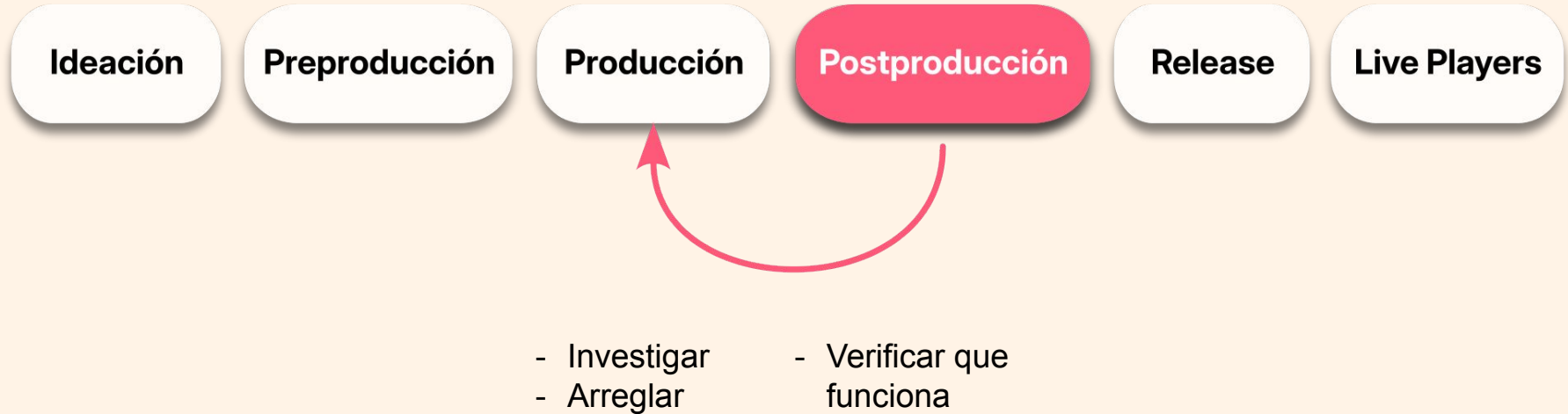
- Investigar
- Arreglar

- Verificar que funciona

- Build
- Subir nueva versión



# Estrategia de QA: objetivo



# Estrategia de QA: objetivo



- Investigar
- Arreglar



# Estrategia de QA: objetivo

**Ideación**

**Preproducción**

**Producción**

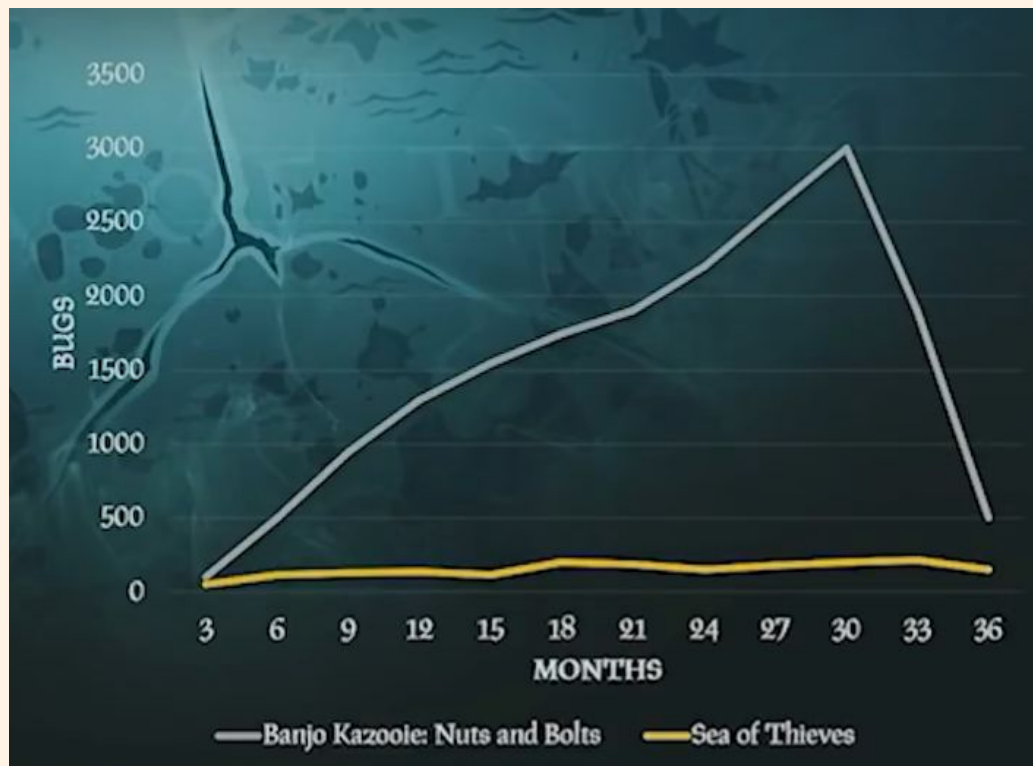
**Postproducción**

**Release**

**Live Players**



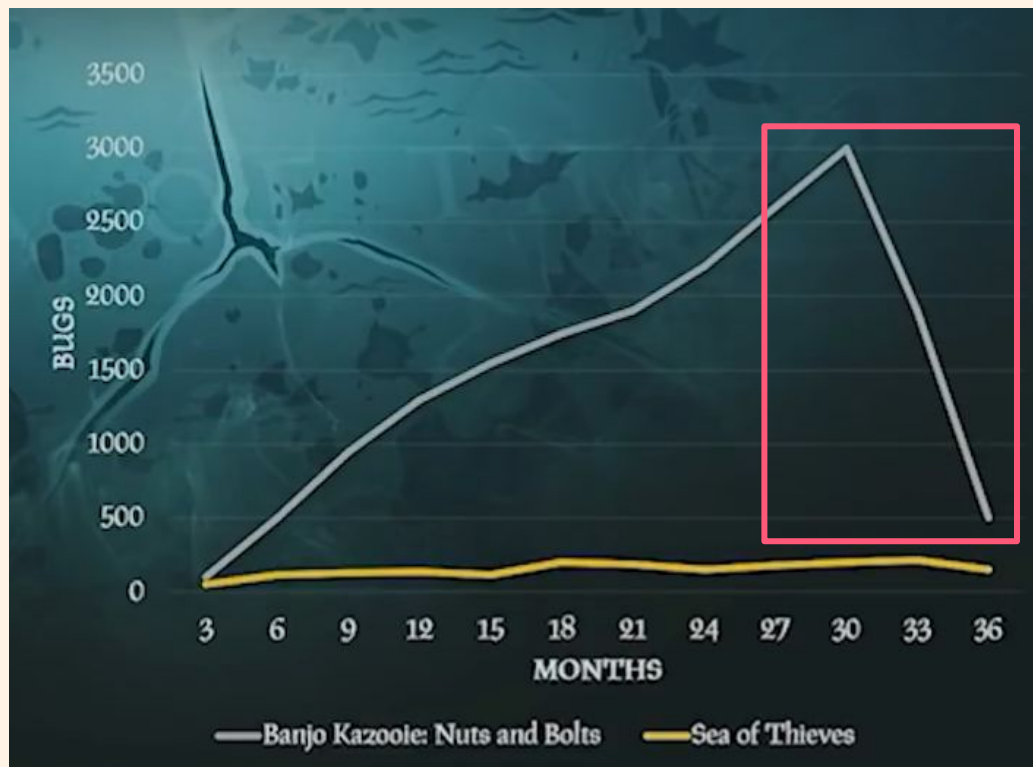
# Shift left testing



"Automated Testing of Gameplay Features in 'Sea of Thieves'", Robert Masella, GDC



# Shift left testing



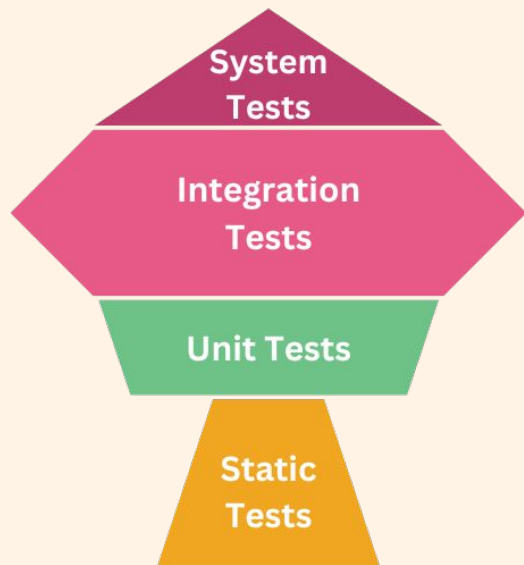
¡Crunch!

"Automated Testing of Gameplay Features in 'Sea of Thieves'", Robert Masella, GDC

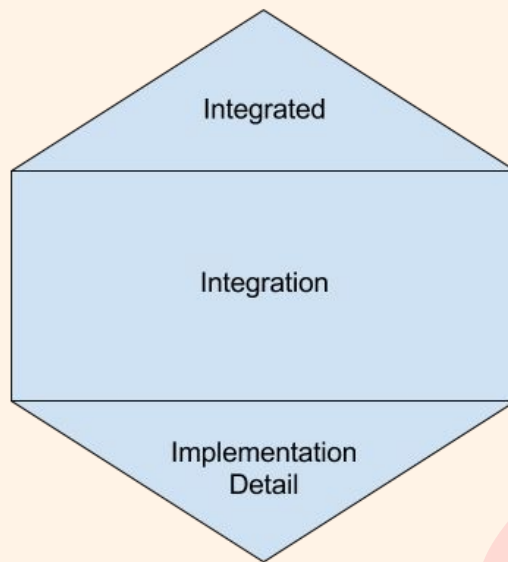


# Implementación de pruebas

“Write tests. Not too many. Mostly integration” — Guillermo Rauch



Trophy testing model, Kent C. Dodds



Microservices honeycomb testing model, Spotify



[empty] X +

File Edit Search Go To Debug

Online Docs Search Help

Filter Scripts

- character\_selection.gd
- main\_menu.gd
- main\_menu\_options.gd
- player\_preview.gd
- scene\_manager.gd
- smoke\_test.gd
- test\_game\_rules.gd

scene\_manager.gd

Filter Methods

- \_ready
- change\_scene
- persist

```
1 extends Node
2 ## Manages scene transitions
3
4 @onready var curtain: Curtain = $UI/Curtain
5
6 func _ready() -> void:
7     EventBus.change_scene.connect(change_scene)
8
9 ## Change scenes with the curtain animation
10 func change_scene(event: EventBus.ChangeSceneEvent) -> void:
11     curtain.play_animation()
12     await curtain.change_scene_now
13
14 for child in $CurrentScene.get_children():
15     child.queue_free()
16     use_instance = event.scene_instance_id()
```

100 % 6 : 16 Tabs

Run All Current: test\_game\_rules.gd test\_scoreboard\_ui\_shown\_on\_player\_death

Passing 19.0 Failing 0.0 Pending 0.0 Orphans 0.0 Errors 0.0 Warnings 0.0

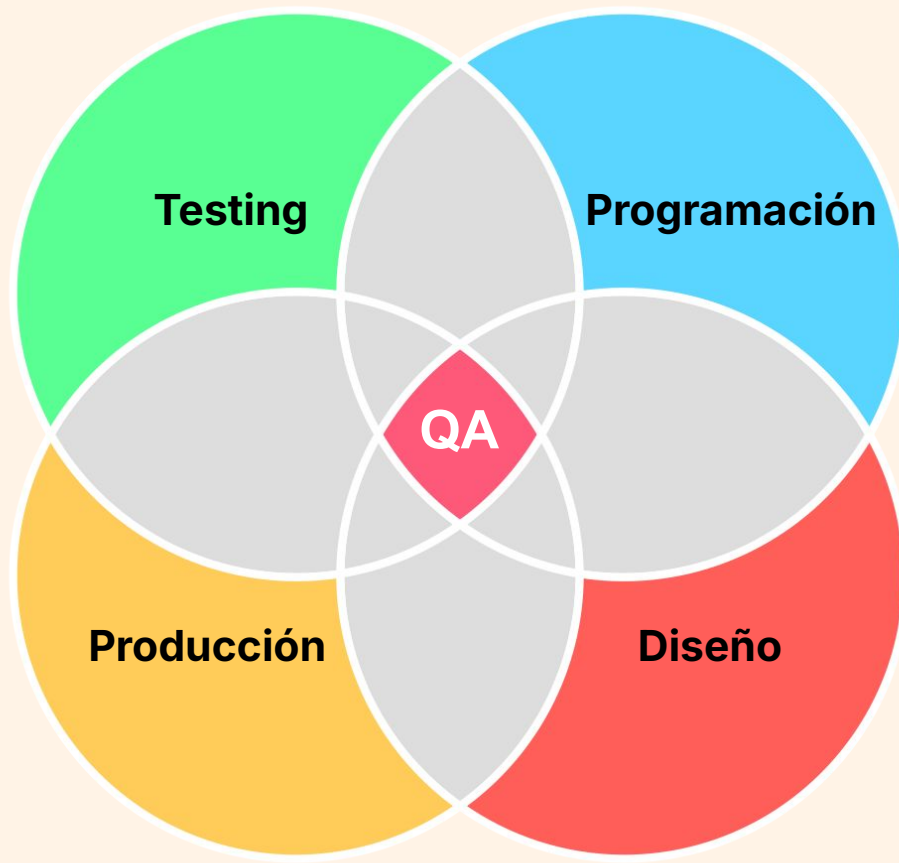
All: Passing Sync:

Q ...

Copy Clear

Everything passed!

**Problema: ¿quién sabe todo esto?**



# Guía alto nivel

0. Interésate por QA, infórmate

1. Lidera con el ejemplo

- Enseñar que otra forma de trabajar es posible

2. Busca apoyo

- Perspectivas del resto del equipo

3. Formalizar y escalar

- Construir carreteras para facilitar esta innovación



# Guía alto nivel

0. Lee artículos/libros, haz pruebas, habla con gente...
1. Piensa en cómo mejorar tu flujo de trabajo
  - Scripts, tests, herramientas...
  - Mide el antes y el después
2. Comparte con tu equipo
  - Propón hacer un experimento
3. Si les convences, hacedlo más seguido



# ¿Experimento?

## DMAIC

VISIT:  **GLSS.APP**

DMAIC is a five-step method for improving existing process problems with unknown causes.



### DEFINE

Clarify  
the problem  
and process



### MEASURE

Quantify the  
problem and  
map the  
process



### ANALYZE

Determine  
the root  
causes



### IMPROVE

Confirm  
the solutions  
work



### CONTROL

Ensure  
the gains are  
sustained



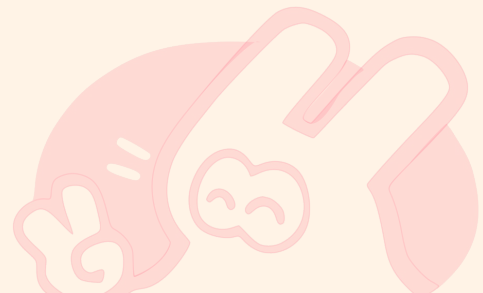
# Mejora continua

¿Cómo se testea un proceso?

- Echando la **vista atrás**: bueno, malo
- Pensando en qué **cambiar** para **mejorarlo**

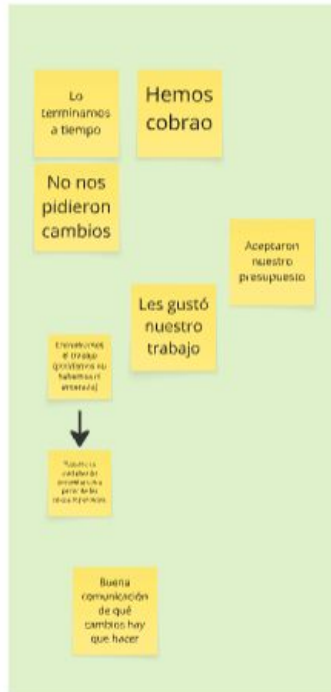
Iterar desbloquea la **mejora continua**

(ej. retrospectiva, postmortem, DMAIC...)



# Retrospectiva

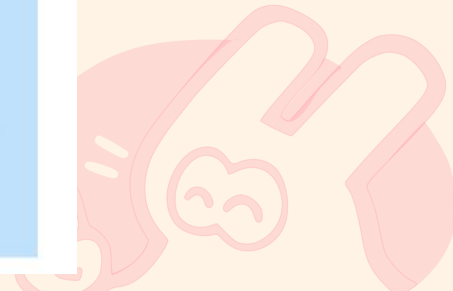
## What went well



## What didn't go well



## How will we adjust in the future





# Hoja de ruta

Para cada rol:



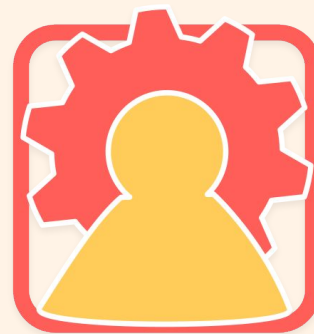
QA



Programmer



Design/  
Artist



Publisher/  
Project Lead



## Hoja de ruta: QA

1. Sirve de ejemplo
2. Guía al resto del equipo
3. Localiza mejoras
4. Vender los éxitos



# Hoja de ruta: QA

## 1. Sirve de ejemplo

- Automatiza un test importante y repetitivo como **proof-of-concept**
- **Integra** este test en las builds automatizadas (si las tenéis)
- **Expande** gradualmente este test suite
  - Céntrate en las interacciones más comunes/vistasas
  - Queremos resultados visibles



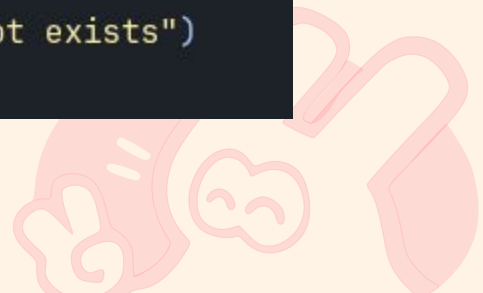
# ¿Cómo hago mi primer tests?

- **GIVEN:**
  - Preparar la **escena**
- **WHEN:**
  - **Actuar** sobre ella, llamar al método
- **THEN:**
  - Hacer comprobaciones, **asserts**



# ¿Cómo hago mi primer tests?

```
▼ >I func test_state_machine_exists():  
  >I >I #GIVEN  
  >I >I var gato: Node = add_child_autofree(sut.instantiate())  
  
  >I >I #WHEN  
  >I >I var state_machine: Node = gato.find_child("StateMachine")  
  
  >I >I #THEN  
  >I >I assert_not_null(state_machine, "Gato StateMachine does not exists")
```



# ¿Cómo hago mi primer tests?

```
» | const ACTION_NAME := "act"
» | const TEMP_FILE := "user://temp.txt"
» | const FILE_CONTENT := '{"control_schemes":[{"input_actions":[],"name":"one","toggle_joystick":fa
▼ | func test_reset_changes() -> void:
» | | # GIVEN
» | | InputMap.add_action(ACTION_NAME)

» | | var file := FileAccess.open(TEMP_FILE, FileAccess.WRITE)
» | | file.store_string(FILE_CONTENT)
» | | file.close()

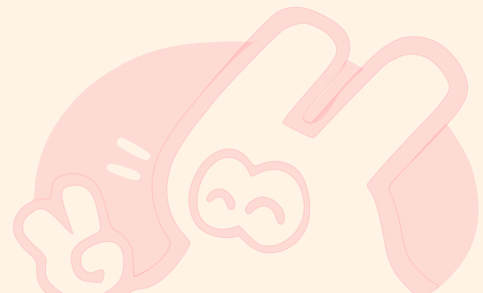
» | | var storage_stub = load("res://addons/input_remapper/service/storage_service.gd").new()
» | | storage_stub.settings_file = TEMP_FILE
» | | var input_remapper = sut.new(storage_stub)

» | | # simulate user changes
▼ | | var input_action = load("res://addons/input_remapper/model/input_action_button.gd")\
» | | | .new(ACTION_NAME, Helper.create_input_event(KEY_0))
» | | | input_remapper.get_current_scheme().input_actions.append(input_action)

» | | # WHEN
» | | input_remapper.reset_changes()

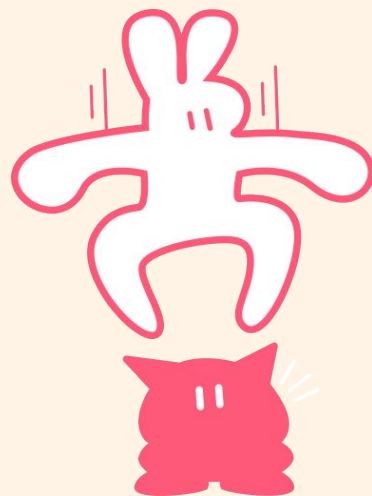
» | | # THEN
» | | assert_eq(input_remapper.get_current_scheme().name, "one")
» | | assert_eq(input_remapper.get_current_scheme().input_actions.size(), 0)
» | | assert_true(InputMap.has_action(ACTION_NAME))
» | | var events := InputMap.action_get_events(ACTION_NAME)
» | | assert_eq(events.size(), 0, "The action has no input events")
» | | input_remapper.free()

» | | InputMap.erase_action(ACTION_NAME)
» | | DirAccess.remove_absolute(TEMP_FILE)
```



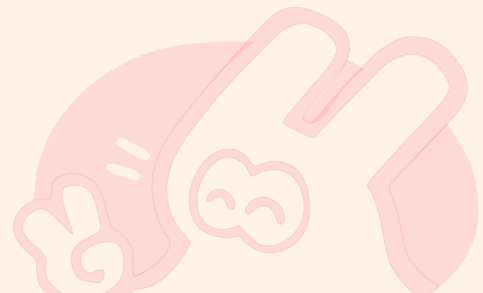
# ¿Por qué sólo 1 acción/tests pequeños?

- **GIVEN:**
  - Personaje y un enemigo
- **WHEN:**
  - Saltas y te pones sobre el enemigo y caes sobre él
- **THEN:**
  - El enemigo muere y el personaje sigue con la misma vida



# ¿Por qué sólo 1 acción/tests pequeños?

- **WHEN:**
  - Saltas
  - Y te pones sobre el enemigo
  - Y caes sobre él





# ¿Por qué sólo 1 acción/tests pequeños?

- **WHEN:**
  - Saltas
    - No input salto
    - No personaje en el nivel
  - Y te pones sobre el enemigo
  - Y caes sobre él



# ¿Por qué sólo 1 acción/tests pequeños?

- **WHEN:**
  - Saltas
  - Y te pones sobre el enemigo
    - No input de movimiento
    - Salta menos de lo esperado
    - Enemigo no existe
  - Y caes sobre él



# ¿Por qué sólo 1 acción/tests pequeños?

- **WHEN:**
  - Saltas
  - Y te pones sobre el enemigo
  - Y caes sobre él
    - Personaje y enemigo en capas de físicas distintas
    - El enemigo detecta la colisión antes y mata al player...



# ¿Qué test nos da más información?

¿Uno que puede **fallar por cualquiera** de estos motivos?

- No personaje en el nivel
- No input salto
- Salta menos de lo esperado
- No input de movimiento
- Enemigo no existe
- Personaje y enemigo en capas de físicas distintas
- El enemigo detecta la colisión antes y mata al player



# ¿Qué test nos da más información?

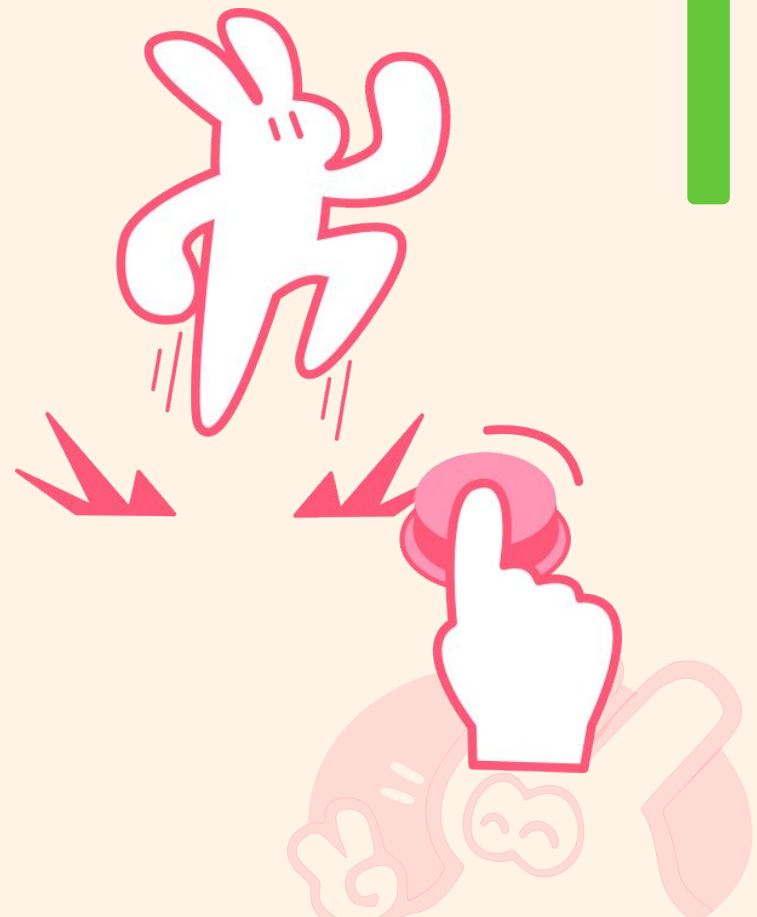
¿O uno que **sólo falla** cuando se rompe esto?

- No personaje en el nivel
- No input salto



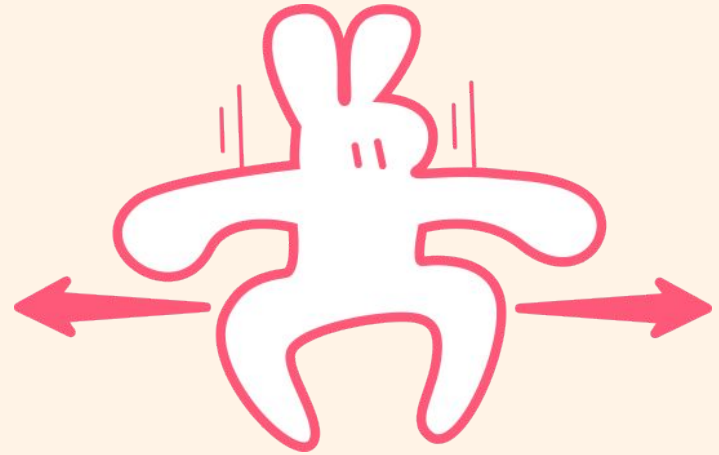
# Divide y vencerás

- **GIVEN:**
  - Personaje y un nivel cargado
- **WHEN:**
  - Saltas
- **THEN:**
  - El personaje salta 3 unidades de alto



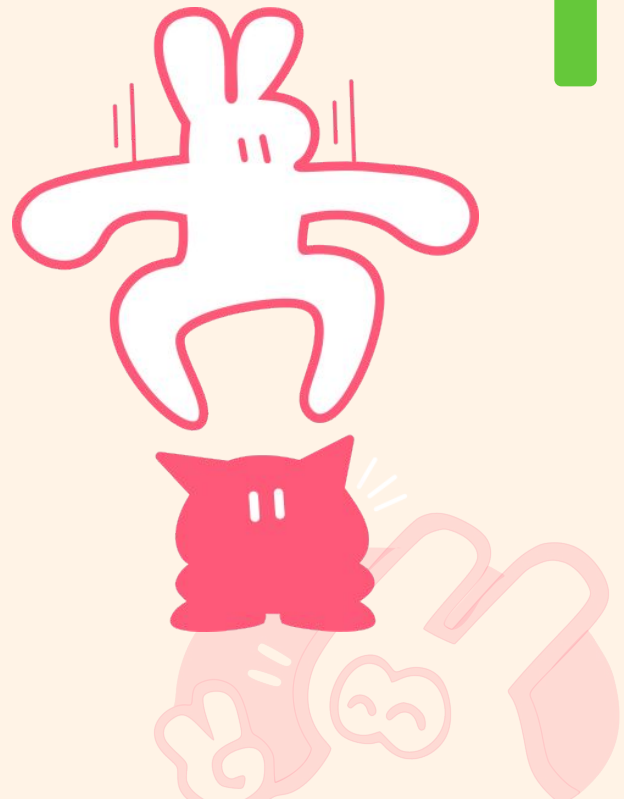
# Divide y vencerás

- **GIVEN:**
  - Personaje en el aire
- **WHEN:**
  - Te mueves a los lados
- **THEN:**
  - El personaje sigue cayendo
  - El personaje se mueve a los lados



# Divide y vencerás

- **GIVEN:**
  - Personaje en el aire sobre un enemigo
- **WHEN:**
  - El personaje toca al enemigo
- **THEN:**
  - El enemigo muere
  - El personaje sigue con la misma vida





# Hoja de ruta: QA

## 2. Guía al resto del equipo

- **Identifica** los test cases manuales más repetitivos
  - Son los primeros en automatizarse
  - Ayuda a que tus compis se animen a hacerlo
- Sesiones de **pair** para enseñar herramientas y técnicas (**feedback!**)
- **Mentoriza** a compis sobre automatización



# Hoja de ruta: QA

## 3. Localiza mejoras

- Pregunta a **otros roles** qué tareas tediosas puedes automatizar
  - De forma **proactiva**
- Ponte la mascarilla primero, luego al resto
  - Genera **confianza** en ti como persona que resuelve problemas








# Hoja de ruta: QA

## 4. Vender los éxitos

- **Comparte los éxitos** con el equipo
  - Convierte errores en tiempo salvado
    - Tiempo desde **detección** del bug a la **entrega** del fix (**MTTR**)
    - Por cada error (\*, no todos!) os habéis **ahorrado** esas horas
- Cuando haya **fallos**, comparte también
  - ¿Qué ha pasado? ¿Cómo se ha solucionado? → Confianza en tools



# Métricas

-  **Passed:** no te dicen nada, ni bueno ni malo
-  **Fallos:** ¡buenísimo! Tradúcelo por **"error prevenido"** ♥
-  **Flakiness...** uff, tu **peor enemigo**. Mina **confianza** 
-  **Tiempo ejecución:** menos es **más tests** ejecutados
- **Coverage:** pasar de 0% a X% está bien, pero llegar al 100% no es útil
  - Podéis usarla como **métrica de mejora** pero no os obsesionéis



# ¿Qué hacer si tenéis flakiness?

1. Establecer un **threshold** (X fallos/día, % de errores)
2. Cuando un test lo cruza, lo **desactivamos** (cuarentena)
  - Si nos falla a nosotros, le falla también al resto del equipo
  - *"Este test falla pero yo no he tocado esa parte"* → mina la **confianza** en los tests
3. Para **no olvidarlo**, creamos un ticket
4. Le asignamos la misma **prioridad** que a un bug
  - ¡Es un bug en el código de test!



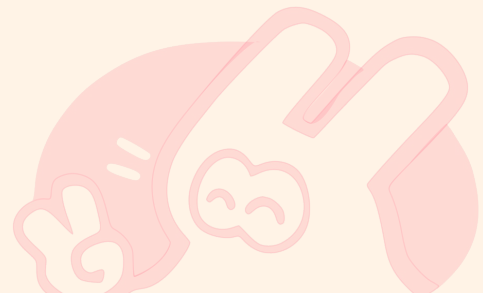
## Hoja de ruta: QA

1. Sirve de ejemplo
2. Guía al resto del equipo
3. Localiza mejoras
4. Vender los éxitos



# Obstáculos: QA

- Exigencias poco **realistas** (todo o nada)
  - Es "mejora **continua**", no "mejora definitiva"
  - Se puede empezar con 1 test e ir añadiendo poco a poco



# Obstáculos: QA

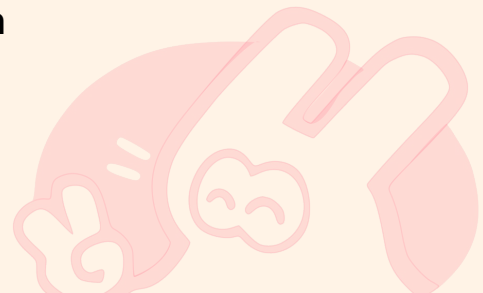
- Exigencias poco **realistas** (todo o nada) → **Mejora continua**
- Falta de **conocimiento**
  - Aprender a programar, falta de perspectiva...
  - ¡**Pregunta** a tus compis! ¡Pide code reviews y sesiones de pair! ¡Pide mentorazgo!





# Obstáculos: QA

- Exigencias poco **realistas** (todo o nada) → **Mejora continua**
- Falta de **conocimiento** → **Pregunta**
- Añadir cosas nuevas es más importante que reforzar las antiguas
  - Redobla el **visibilizar** los **éxitos** de los tests
  - Calcula cuánto tiempo/dinero **ahorran** esos éxitos
  - Calcula cuánto tiempo/dinero **gastáis** por no usar automatización



# Obstáculos: QA



- Exigencias poco **realistas** (todo o nada) → **Mejora continua**
- Falta de **conocimiento** → **Pregunta**
- Añadir cosas nuevas es más importante que reforzar las antiguas  
→ Habla de **tiempo/dinero**



# Hoja de ruta: Programmer

1. Sirve de ejemplo
2. Colaboración cercana con QA
3. Comparte éxitos



# Hoja de ruta: Programmer

## 1. Sirve de ejemplo

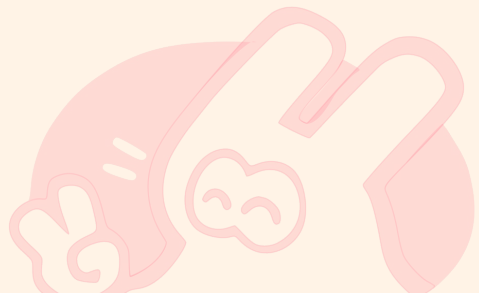
- Automatiza un test importante y repetitivo como **proof-of-concept**
- **Integra** este test en las builds automatizadas si las tenéis
- **Expande** gradualmente esta test suite
- Automatiza la **build**
- Crea **herramientas**



# Hoja de ruta: Programmer

## 2. Colaboración cercana con QA

- Incluye a QA en reuniones de **diseño técnico**
- "¿De qué manera se puede **romper** esto?"
- Pair con QA para **ayudarle** con cosas más técnicas



# Hoja de ruta: Programmer

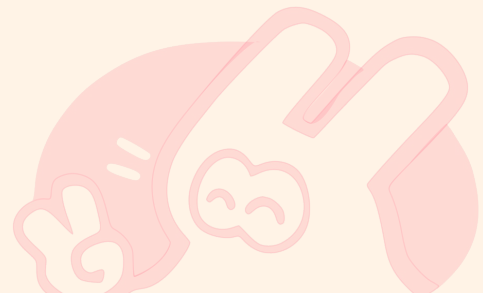
## 3. Comparte éxitos

- Eres vital para recoger **métricas**
- **Comparte** el bug que ha encontrado y cómo te ahorra tiempo
- Si tienes review, incluye **métricas** de testing que has mejorado



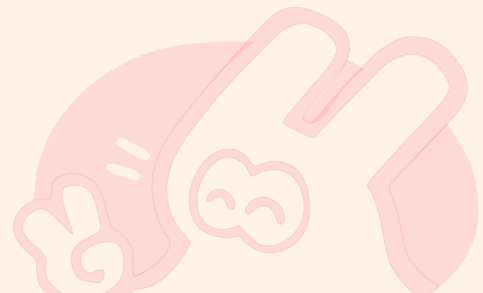
# Obstáculos: Programmer

- No sabes cómo escribir tests
  - GIVEN-WHEN-THEN
  - 1 acción, scope mínimo
  - Fallos = ¡bien! Flaky = terrible



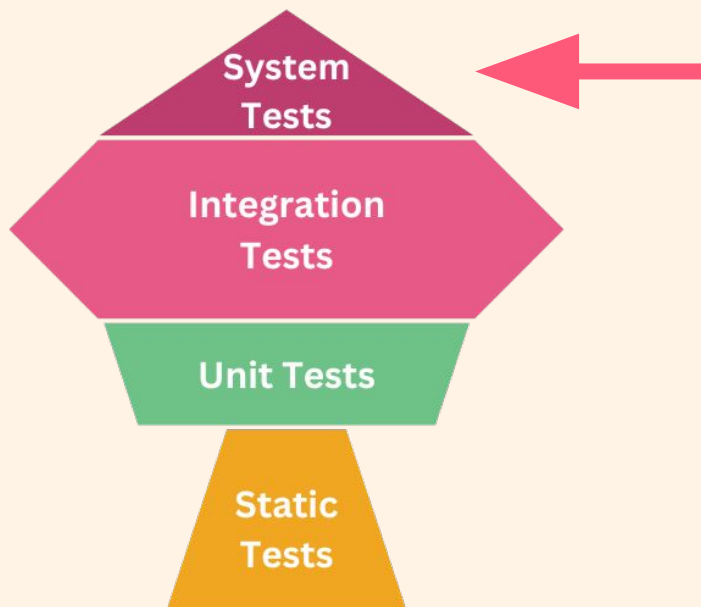
# Obstáculos: Programmer

- No sabes cómo escribir tests
- No sabes qué tests escribir/quieres resultados rápido





# Saltarse las normas cuando tiene sentido



Trophy testing model,  
Kent C. Dodds



# Smoke test




# Smoke test

Abre las escenas del juego para ver si tienen errores

```
74 - Testing res://src/core/game_menu/character_selection/character_selection.tscn...
75 - Testing res://src/core/game_menu/character_selection/player_selection_preview.tscn...
76 - Testing res://src/core/game_menu/character_selection/roster_character.tscn...
77 - Testing res://src/core/game_menu/credits/credits.tscn...
78 - Testing res://src/core/game_menu/customize_game_selection/game_mode_selection.tscn...
79 SCRIPT ERROR: Parse Error: Invalid argument for "new()" function: argument 4 should be "bool" but is "Resource".
80     at: GDScript::reload (res://src/core/game_menu/game_mode_selection/game_mode_selection.gd:22)
81 ERROR: Failed to load script "res://src/core/game_menu/game_mode_selection/game_mode_selection.gd" with error "Parse
error".
82     at: load (modules/gdscript/gdscript.cpp:3022)
83 - Testing res://src/core/game_menu/game_mode_selection/game_mode_selection.tscn...
```





# Smoke test



NUEVO

Tests Michiball **APP** 20:23

Smoke test execution found 34 errors  and 8 warnings .

Check the results in [this url](#).



# Screenshot test

List of items:

- Item 1
- Item 2
- Item 3
- Item 4

CAPTURA ORIGINAL

List of items:

- Item 1
- Item 2
- Item 3
- Item 4
- Item 5

DIFERENCIA

■ No existe en la nueva

■ Existe en la nueva

List of items:

- Item 1
- Item 2
- Item 3
- Item 4
- Item 5

CAPTURA DEL TEST



# Screenshot tests



Pantalla de selección de personaje, Michiball

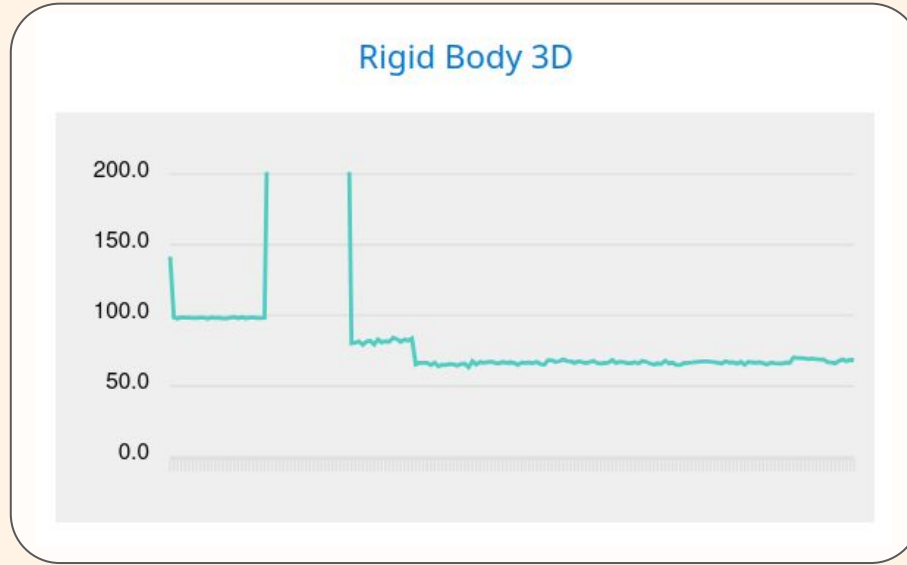


Menú principal, Michiball



# Performance test/benchmark

Ej. <https://benchmarks.godotengine.org/>

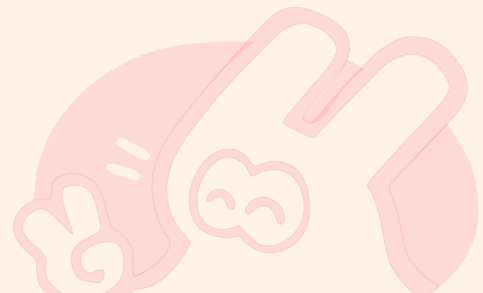


Benchmarks de Godot Engine



# Programmer: Obstáculos

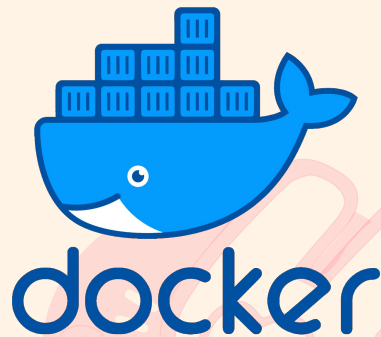
- No sabes cómo escribir tests
- No sabes qué tests escribir/quieres resultados rápido
- ¡Ayuda! ¡Ahora todo el mundo me pide ayuda!





# Integración continua (CI)

- Ejecutar scripts en **respuesta a**:
  - Subir un commit
  - Hacer merge de una pull request
  - Cada X tiempo
  - Pulsar un botón
- Automatización al **alcance** de todo el equipo
- Iterar más **rápido**, de forma más **consistente**



# Entorno de CI: ejemplos de uso

- Generar **builds**
- **Subirlas** a Itch.io, Steam, App Stores...
- Lanzar los **tests** → si fallan, **bloquear** cambios
- **Validar assets**
- Señalar **texto sin localizar** añadido en cada cambio
- Actualizar un excel con **datos de diseño** con cada cambio (ej. economía)
- **Notificar** (ej. con un mensaje en Discord)



# Entorno de CI: configuración

```
1  name: Build + Deploy
2  on: workflow_dispatch
3
4  env:
5    - ITCHIO_USERNAME: nokorpo
6    - ITCHIO_GAME: michiball
7    - BUTLER_API_KEY: ${ secrets.BUTLER_API_KEY }
8    - GODOT_VERSION: 4.4.1
9
10 jobs:
11   - web:
12     - name: Build and deploy to itch.io
13     - runs-on: ubuntu-latest
14     - container:
15       - image: barichello/godot-ci:4.4.1
16     - steps:
17       - name: Checkout
18       - ...
19       - name: Setup
20       - ...
21       - name: Web Build
22       - ...
23       - name: Itch.io Deploy
24       - ...
```

← Cuándo se ejecuta

← Variables:

Datos, nombres, passwords...

← Pasos a seguir

Configuración de build +  
deploy en Github Actions



# Hoja de ruta: Designer/Artist



1. Define (y usa) **convenciones de nombrado** para assets
2. **Reuniones tempranas** con QA+programmers
  - ¿Cómo va a implementarse? ¿De qué forma se puede **romper**?
  - Comparte tu flujo de trabajo: ¿se puede **automatizar** algo?
3. **Comparte** cómo estas herramientas mejoran tu workflow
  - **Celebra** cuando una release va bien gracias a ello



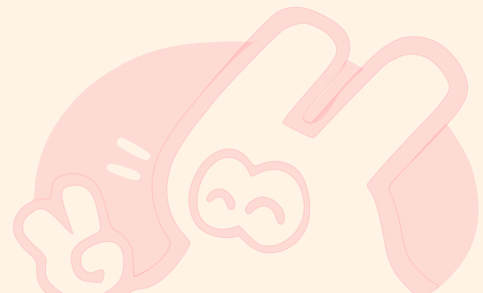
# Oportunidades: Designer/Artist

- Aplicar **configuración** automáticamente
  - Scripts de import/export
  - Custom tooling
- **Validar** assets en muchos lugares
  - Nombrado
  - Configuración correcta
  - Performance
- No tener que **repetir** los mismos 5 clicks 30 veces



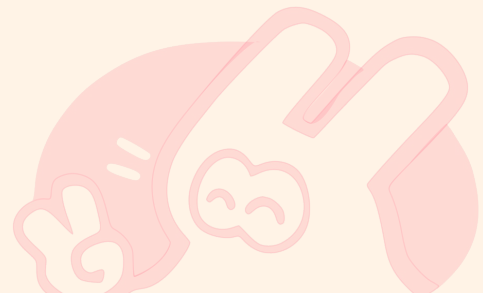
# Obstáculos: Designer/Artist

- Aprender otro programa más
  - Pide pair session (**feedback!**), exige documentación



# Obstáculos: Designer/Artist

- Aprender otro programa más → **pair + documentación**
- Tarea repetitiva: “pues toca hacerlo así”
  - No, pregunta si se puede automatizar
  - Ej. exportar 30 modelos distintos en el mismo fichero de Blender
  - Ej. arrastrar el fichero al proyecto y que se le aplique la configuración



# Obstáculos: Designer/Artist

- Aprender otro programa más → **pair + documentación**
- Tarea repetitiva → **automatizar**
- Sólo programmer/QA pueden usarlo
  - Pide entorno de CI para todo el equipo



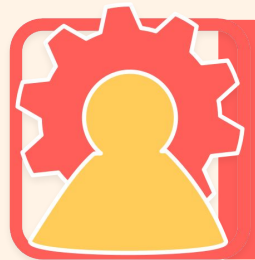


# Obstáculos: Designer/Artist

- Aprender otro programa más → **pair + documentación**
- Tarea repetitiva → **automatizar**
- No tienes acceso → **entorno CI**



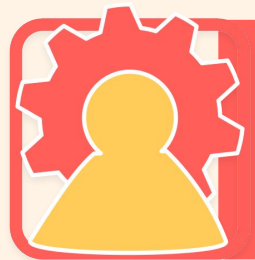
# Hoja de ruta: Publisher/Project Lead



1. Exige unas **quality gates** en el contrato
  - Vincula pagos por milestones a esas métricas de calidad
2. Reconoce y recompensa a los equipos con **alta estabilidad**
  - Comparte métricas y benchmarks de los mejores proyectos
  - **Forma** y trata de **replicar** esa infraestructura en otros proyectos
3. **Financia**/crea tus propios **recursos** centralizados de automatización



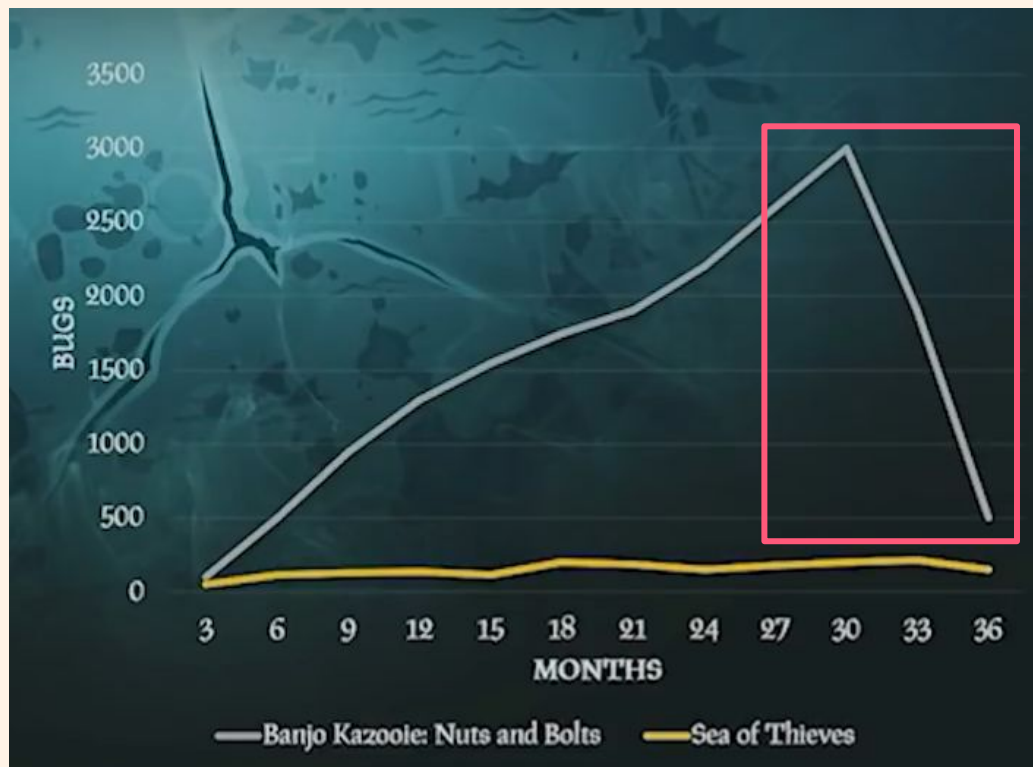
# Oportunidades: Publisher/Project Lead



- Encontrar bugs antes (**ahorro** de tiempo y \$\$\$)
  - No lanzar versiones rotas protege tu **reputación** y la del equipo
  - Menos probabilidad de un mega-bug que te **arruine** la **release**



# Oportunidades: Publisher/Project Lead



¡Crunch!

"Automated Testing of Gameplay Features in 'Sea of Thieves'", Robert Masella, GDC

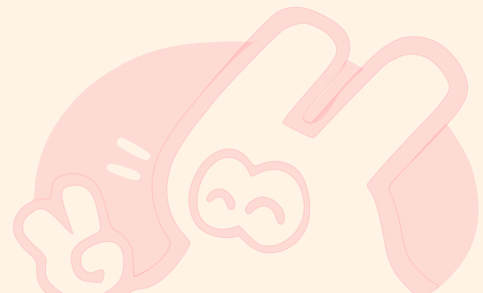
# Oportunidades: Publisher/Project Lead

- Encontrar bugs antes (**ahorro** de tiempo y \$\$\$)
  - No lanzar versiones rotas protege tu **reputación** y la del equipo
  - Menos probabilidad de un mega-bug que te **arruine** la **release**
- Facilita A/B testing para una toma de **decisiones data-driven**
- **Acelera** el delivery y reduce el time-to-market
  - Agilidad, cambiar de rumbo cuando hace falta
  - No tienes que abrir camino, usas la carretera que has construido



# Obstáculos: Publisher/Project Lead

- Nadie en el equipo sabe cómo hacer tests/automatizar



# Obstáculos: Publisher/Project Lead

- Nadie en el equipo sabe cómo hacer tests/automatizar

¡Contáctanos! → [contact@nokorpo.com](mailto:contact@nokorpo.com)

(si eres indie también, te compartimos recursos)



# ¿Preguntas?



<https://nokorpo.com>

- Slides
- Más información
- Enlaces relevantes
- Contacto

