

ESPECIFICAÇÃO FORMAL

INTRODUÇÃO

Este documento apresenta a especificação formal dos requisitos do jogo Eco Runner.

PLANEJAMENTO

A especificação formal dos requisitos será realizada utilizando notação Z e redes de petri. Para cada requisito funcional e não funcional, haverá uma especificação.

REQUISITOS FUNCIONAIS

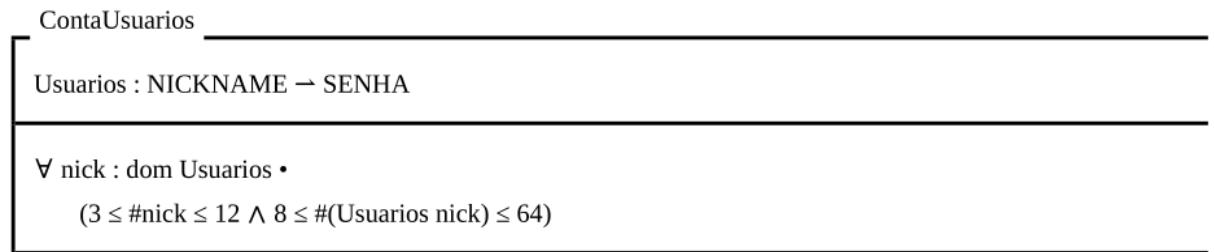
RFO1

- Notação Z
- Tipos Básicos:

[NICKNAME, SENHA]

RespostaLogin ::= valido | erro_nick_invalido | erro_senha_invalida

- Esquema de estado:



- Esquemas de operação:

LoginSucesso

ΞContaUsuarios

nick? : NICKNAME

senha? : SENHA

resp! : RespostaLogin

nick? ∈ dom Usuarios

Usuarios(nick?) = senha?

resp! = valido

NickInvalido

ΞContaUsuarios

nick? : NICKNAME

senha? : SENHA

resp! : RespostaLogin

nick? ∉ dom Usuarios

resp! = erro_nick_invalido

SenhaInvalida

ΞContaUsuarios

nick? : NICKNAME

senha? : SENHA

resp! : RespostaCadastro

nick? ∉ dom Usuarios

Usuarios nick ≠ senha?

resp!= erro_senha_invalida

- Operação completa:

$\text{Login} \triangleq \text{LoginSucesso} \vee \text{NickInvalido} \vee \text{SenhaInvalida}$

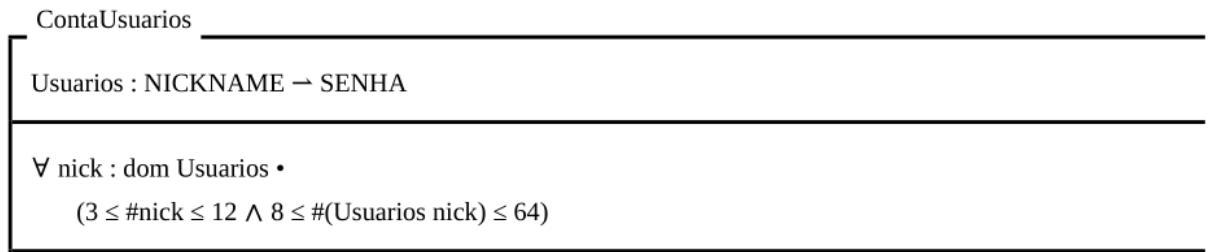
RFO2

- Notação Z
- Tipos Básicos:

[NICKNAME, SENHA]

RespostaCadastro ::= valido | erro_nick_existe | erro_formato

- Esquema de estado:



- Esquemas de operação:

CadastrarSucesso

ΔContaUsuarios

nick? : NICKNAME

senha? : SENHA

resp! : RespostaCadastro

$3 \leq \#nick? \leq 12$

$8 \leq \#senha? \leq 64$

nick? \notin dom Usuarios

Usuarios' = Usuarios \cup {nick? \mapsto senha?}

resp! = valido

FormatoInvalido

ΞContaUsuarios

nick? : NICKNAME

senha? : SENHA

resp! : RespostaCadastro

$(3 > \#nick? \vee \#nick? > 12) \vee$

$(8 > \#senha? \vee \#senha? > 64)$

resp! = erro_formato

NickJaExiste

ΞContaUsuarios

nick? : NICKNAME

senha? : SENHA

resp! : RespostaCadastro

$3 \leq \#nick? \leq 12$

$8 \leq \#senha? \leq 64$

nick? \in dom Usuarios |

resp! = erro_nick_existe

- Operação completa:

Cadastrar \triangleq CadastrarSucesso \vee FormatoInvalido \vee NickJaExiste

RF04

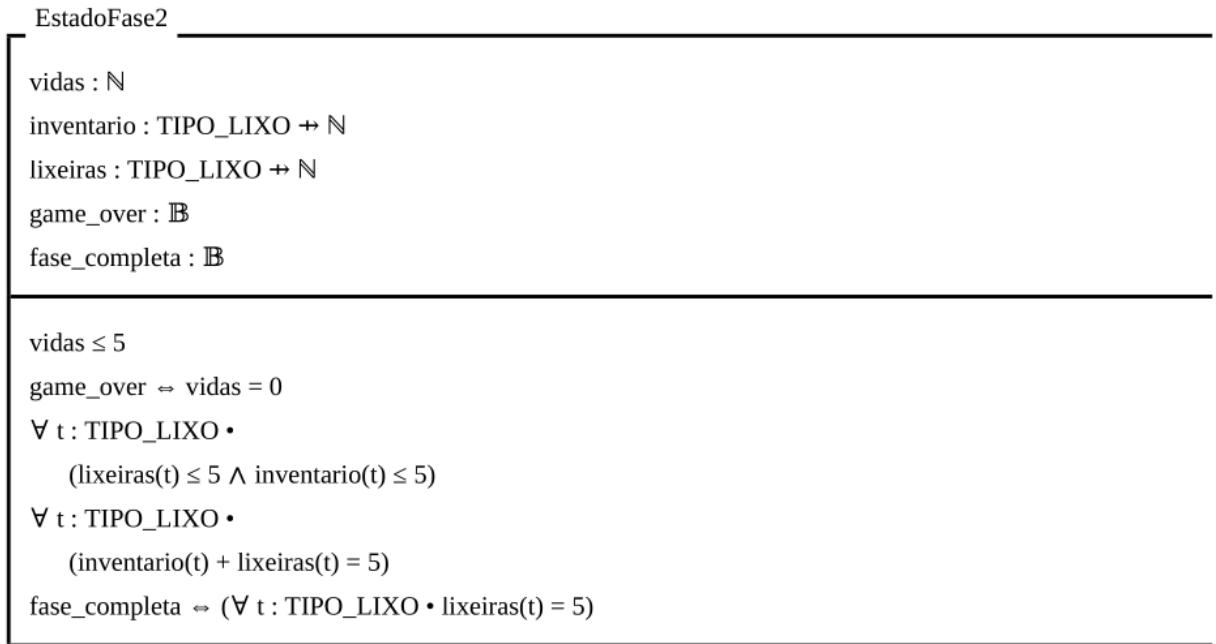
- Notação Z

- Tipos básicos

$TIPO_LIXO ::= papel | plastico | vidro | metal$

$RespostaSort ::= valido | erro_vidas_zero | erro_inventario_vazio$

- Esquema de estado:



- Esquemas de operação:

SortItemCorreto

Δ EstadoFase2

item? : TIPO_LIXO

lixeira? : TIPO_LIXO

resp! : RespostaSort

\neg game_over \wedge \neg fase_completa

item? = lixeira?

inventario(item?) > 0

lixeiras(lixeira?) < 5

inventario' = inventario \oplus {item? \mapsto inventario(item?) - 1}

lixeiras' = lixeiras \oplus {lixeira? \mapsto lixeiras(lixeira?) + 1}

vidas' = vidas

Resp! = valido

SortItemErrado

Δ EstadoFase2

item? : TIPO_LIXO

lixeira? : TIPO_LIXO

resp! : RespostaSort

\neg game_over \wedge \neg fase_completa

item? \neq lixeira?

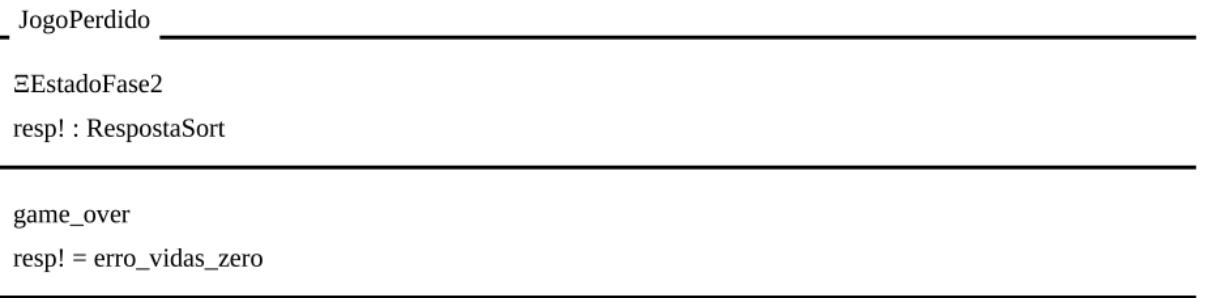
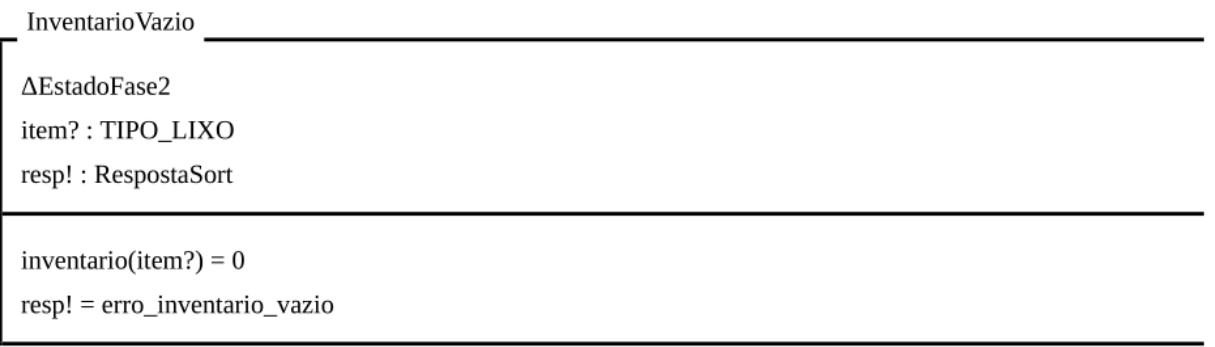
inventario(item?) > 0

vidas' = vidas - 1

inventario' = inventario

lixeiras' = lixeiras

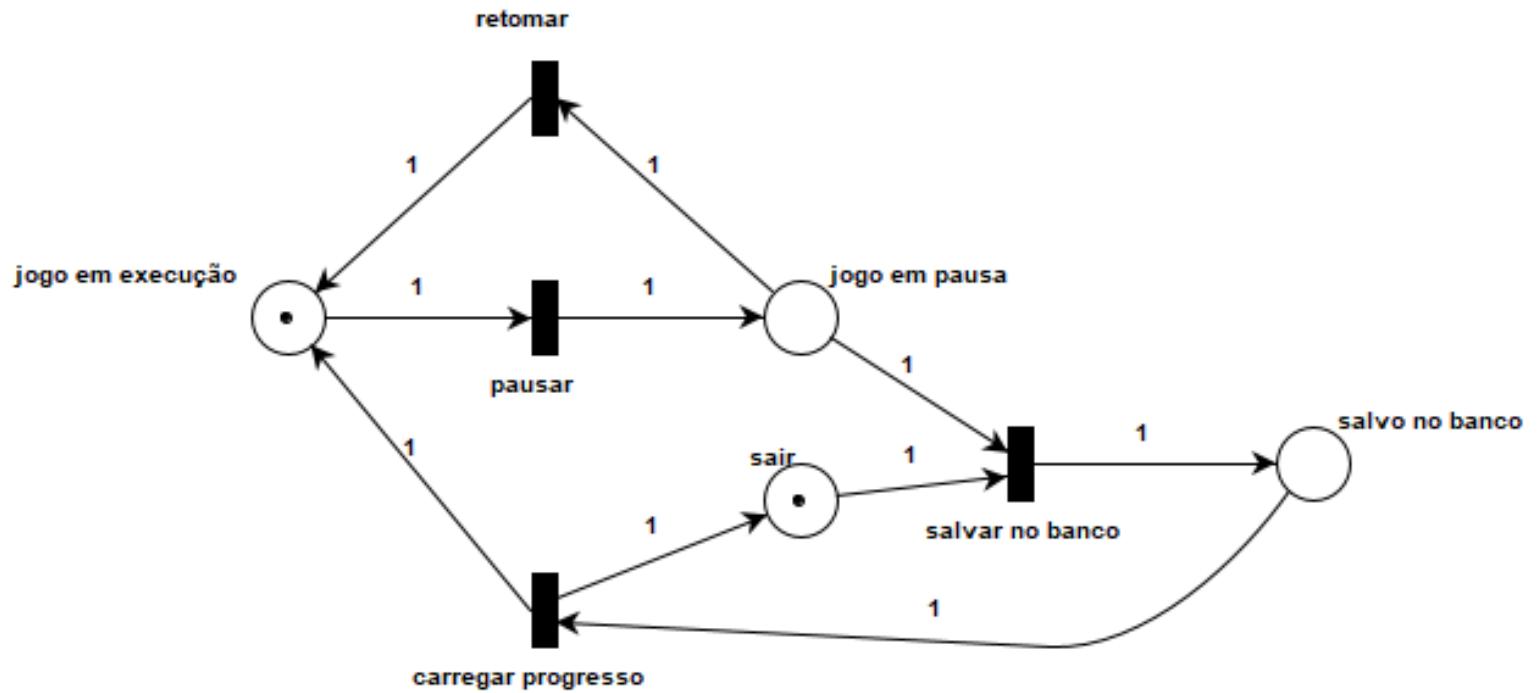
resp! = valido



- Operação completa:

$\text{SortItem} \triangleq \text{SortItemCorreto} \vee \text{SortItemErrado} \vee \text{InventarioVazio} \vee \text{JogoPerdido}$

RF06



REQUISITOS NÃO FUNCIONAIS

RNF02

- Notação Z
- Tipos básicos
[TELA, COR, POSICAO, BOTAO]
- Esquema de estado:

Interface

telas : \mathbb{P} TELA

botoes: \mathbb{P} BOTAO

cor_botao_em_tela : (TEL A \times BOTAO) \rightarrow COR

pos_botao_em_tela : (TEL A \times BOTAO) \rightarrow POSICAO

$\forall t_1, t_2 : \text{telas} ; b : \text{botoes} \bullet$

$((t_1, b) \in \text{dom cor_botao_em_tela} \wedge (t_2, b) \in \text{dom}$

$\text{cor_botao_em_tela}) \Rightarrow \text{cor_botao_em_tela}(t_1, b) = \text{cor_botao_em_tela}(t_2, b)$

$\forall t_1, t_2 : \text{telas} ; b : \text{botoes} \bullet$

$((t_1, b) \in \text{dom pos_botao_em_tela} \wedge (t_2, b) \in \text{dom}$

$\text{pos_botao_em_tela}) \Rightarrow \text{pos_botao_em_tela}(t_1, b) = \text{pos_botao_em_tela}(t_2, b)$

RNF03

- Notação Z
- Tipos básicos:
 - SO_VALIDO ::= windows10 | windows11 | ubuntu22_04
- Esquema de estado

AmbienteDeExecucao

so_atual: SO_VALIDO

RNF04

- Notação Z
 - Tipos básicos
 - [PROGRESSO]
- RespostaSave ::= sucesso | falha

- Esquema de estado

SistemaDePersistencia _____

progresso_atual: PROGRESSO
progresso_salvo: PROGRESSO

- Esquemas de operação

SalvarSucesso _____

Δ SistemaDePersistencia
resp! : RespostaSave

progresso_salvo' = progresso_atual
progresso_atual' = progresso_atual
resp! = ok

SalvarFalha _____

Δ SistemaDePersistencia
resp! : RespostaSave

progresso_salvo' = progresso_salvo
progresso_atual' = progresso_atual
resp! = falha

CarregarJogo _____

Δ SistemaDePersistencia

progresso_atual' = progresso_salvo
progresso_salvo' = progresso_salvo

- Operação completa

$$\text{SalvarJogo} \triangleq \text{SalvarSucesso} \vee \text{SalvarFalha}$$

