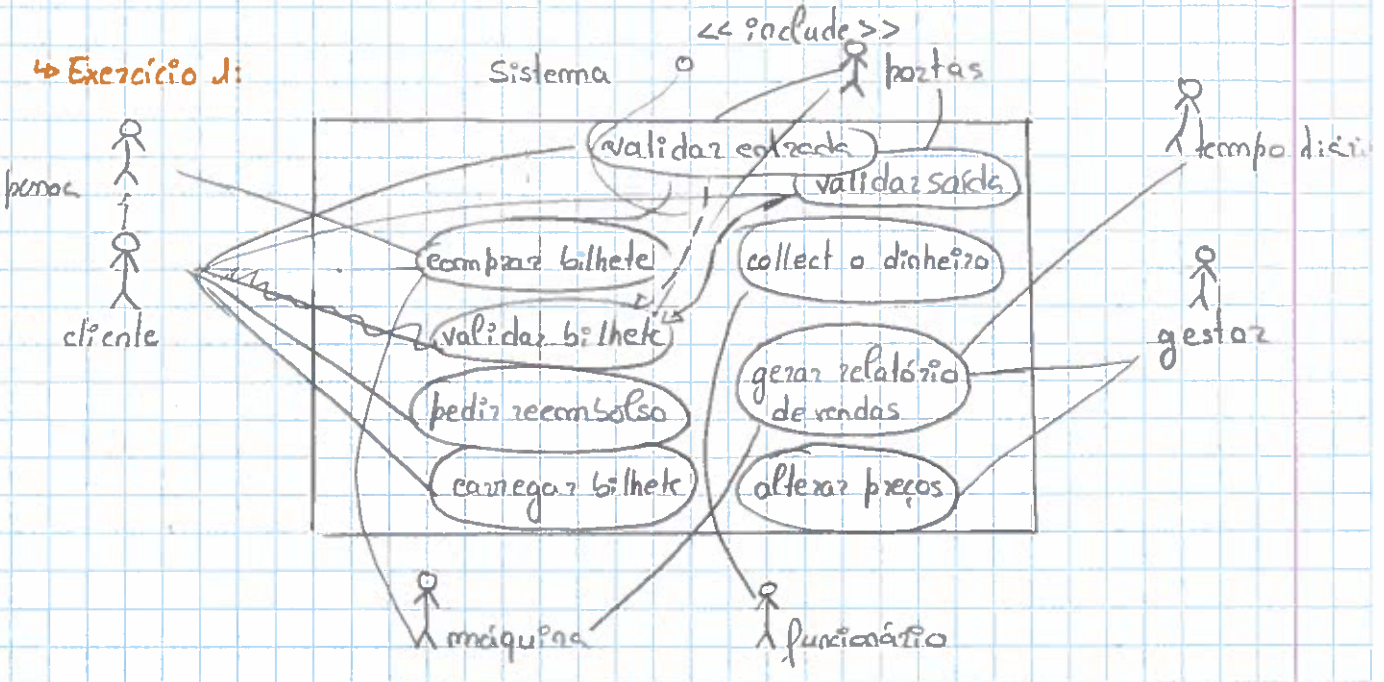
Lab.3: UML Use Cases

Exercício 1:



Nome : Comprar bilhete

Descrição: uma pessoa realiza a compra de um bilhete

Atores: • Principais: Pessoa • Secundários: Máquina

Pré-condições: a pessoa não tem bilhete

Fluxo Principal:

1. a pessoa dirige-se à máquina
2. o sistema disponibiliza as opções possíveis
3. a pessoa escolhe a opção de comprar bilhete
4. o sistema dispõe o valor a pagar
5. a pessoa insere o dinheiro na máquina
6. o sistema associa o valor inserido a um bilhete
7. a máquina devolve esse bilhete e o troco correspondente ao dinheiro inserido e o valor do bilhete
8. a pessoa recolhe o bilhete e o troco (se houver)
9. o use case termina

Fluxos alternativos: dinheiro insuficiente

Pós - condição : nenhuma

Nome: Validar Bilhete

Desc: O cliente valida o seu bilhete nas portas automáticas

Atores: • Principal: cliente • Secundário: portas

Pré-condição: —

Main flow:

1. o cliente passa o bilhete
2. o sistema liga a luz verde
3. o sistema ordena a abertura das portas ao actor portas
4. o cliente passa pelas portas
5. o sistema desliga a luz verde
6. o sistema ordena o fecho das portas ao actor portas

Fluxos Alt: o bilhete é inválido, bloqueio das portas

Pós condição: nenhuma

Nome: Validar Entrada

Descrição: o cliente inicia a viagem, logo tem de validar o bilhete

Pré-condição: nenhuma

MF: 1. Include (validar bilhete)
2. o sistema decrementa o saldo do bilhete
3. o sistema devolve o bilhete
4. o use case termina

Fluxo Alt/excetivos: saldo insuficiente

Pós: nenhuma

Nome: Validar Saída

Desc: o cliente termina a viagem

Pré-condição: necessário validar primeiro a entrada

MF: 1. Include (Validar Bilhete)
2. o sistema mostra uma mensagem a dizer quanto saldo o bilhete tem
3. o use case termina

Nome: Pedir Reembolso

Desc: o cliente pretende ter o dinheiro que depositou no bilhete

Atores: • P: o cliente • S: a máq

Pré-condição: o cliente tem bilhete

FEP: 1. o cliente dirige-se à máquina
2. o sistema disponibiliza as opções
3. o cliente escolhe a opção de pedir um reembolso
4. o sistema pede ao utilizador para inserir o cartão
5. o cliente insere o bilhete
6. o sistema devolve ao cliente o dinheiro correspondente ao saldo do bilhete
7. o use case termina

Pós: nenhuma

Fluxos Alt: bilhete inválido, sem saldo no bilhete, bilhete expirou

—"—

Nome: Carregar Bilhete

Atores: P = cliente • S: máquina

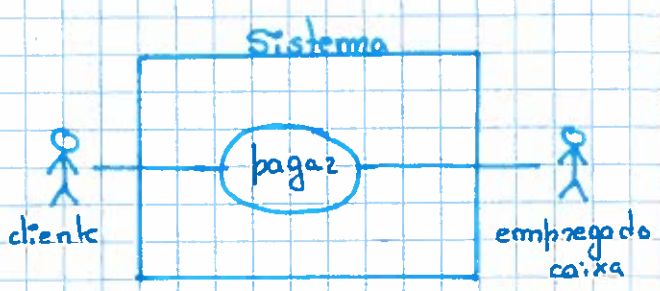
Pré-condição: o cliente tem bilhete

FEP: 1. o cliente chega à máq.
2. o " "
3. o cliente escolhe a opção de carregar bilhete
4. o cliente insere o bilhete na máquina
5. o sistema dispõe uma mensagem a perguntar o valor de carregamento
6. o cliente insere o dinheiro
7. o sistema incrementa o saldo do bilhete
8. a máquina devolve o bilhete e o recibo
9. o use case termina

Name: Gerar relatório de rondas

Desc: o sistema

→ Lab. 4 - UML - Use Cases



- Nome: pagar
- descrição:
- ator principal: cliente
- atores secundários: empregado
- pré-condição: o cliente fez cartão de pontos

1. o cliente chega à caixa
2. o sistema diz olá
3. o cliente indica o método de pagamento
   :
10. o use case acaba

} cenário com happy ending (tudo corre bem)

- pós-condição: o cartão de pontos está atualizado
- cenários secundários: empregado ausente, ..., ...

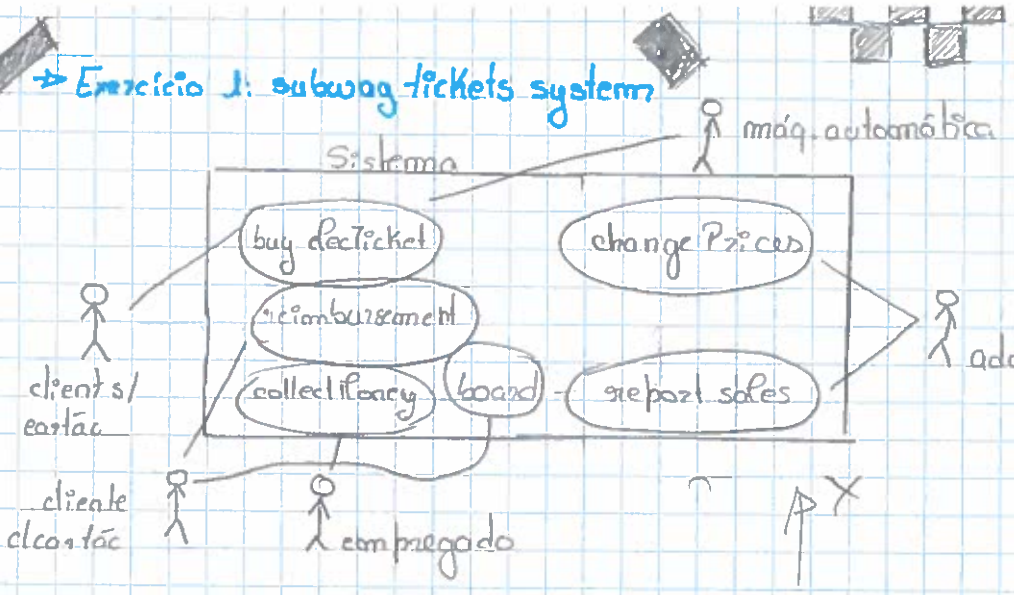
↑ um cenário secundário ——// ↑ um cenário principal

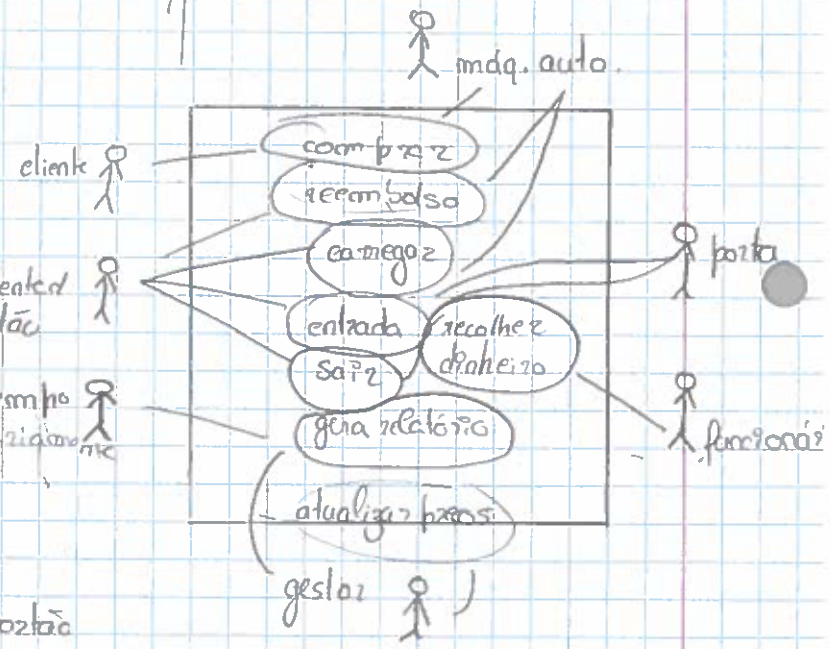
- Nome: Empregado ausente
  pagar: empregado ausente
- use case inicial: pagar
- ator principal: cliente
- atores secundários: empregado
- pré-condição: —

1. no ponto 1 do use case principal o empregado está ausente
2. o sistema liga alarme do empregado
3. o empregado volta à caixa e mostra cartão
4. o use case continua no passo 2. do cenário principal

- pós-condição: —

→ Exercício 1: subway tickets system





↳ Use cases:

Nome: comprar cartão

- Desc: realiza a compra do cartão
- Atores:
  - → principal: cliente
  - → secundários: máq. auto
- Pré-condição: o cliente não tem cartão
- Main flow:
  1. o cliente à máq. auto
  2. o sistema disponibiliza as opções do cliente
  3. o cliente seleciona a opção de comprar bilhete
  4. a máq. auto dispõe a valor a pagar
  5. o cliente insere o dinheiro na máq. auto
  6.

nota: tudo o que provoca alteração no sistema é um ator

o ecdc ⬭ é um use case

Nome: Entrada

Desc: o cliente inicia a viagem

Ator:

→ principal: cliente e/ cartão

→ secundário: porta

Pré-condição: —

Main flow:

1. o cliente e/ cartão passa o cartão
2. o sistema liga luz verde
3. o sistema ordena abertura de porta ao actor porta
4. o cliente passa
5. o sistema desliga luz verde
6. o sistema ordena o fecho da porta
7. o sist. descontacta valor do bilhete no cartão
8. o uc acaba

Pós-condição: —

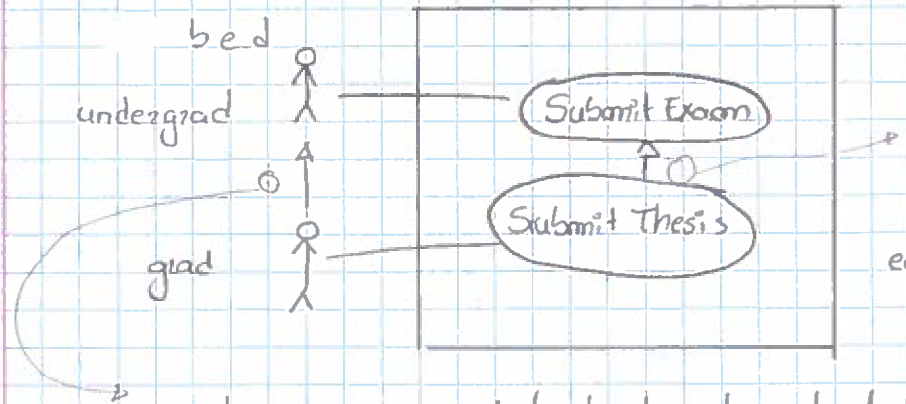
Cenários: cartão inválido, cartão expirado, saldo insuficiente, cliente não passou, bloqueio de porta

---//---

Alternative flow: Ticket is expired

Fluxo alternativo:

→ Exercício (3):



→ o submit thesis herda então as caracteristicas do submit exam, como também os seus atores

→ "aqui dizemos que grad herda de undergrad e pode submit exams " & envolver de alguma maneira o undergrad"

→ Exercício (4):

(a) V (b) V (c) F (d) F (e) F (f) ✓

(c) → use cases são UMA ferramenta, ≠ A ferramenta

(e) → o flow está nos cenários não no diagrama

use case → o que o sistema faz pelo ator

(d) → com os diagramas não sabemos o "como"

→ Exercício (5): (a)

→ Exercício (6):

1. existe uma "ilha" ; um use case sozinho

2. não há sequências nos use cases

3. ator sist. é estúpido, o sistema é a caixa

4. ator universidade que não faz nada

5. relação entre uni e aluno is fucking stupid bc it doesn't exist

6. ator DI não faz um caralho

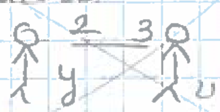
7. em diagramas use case não declaramos mensagens de erro, visto que isto que queremos é o ponto de vista de negócio

8. "Espera" ≠ reflete nada ao ponto de negócio

9. coisas como "espera", "valida" não representa nenhuma ação ~~entre~~ atividade de algo que faço shit c/o sistema

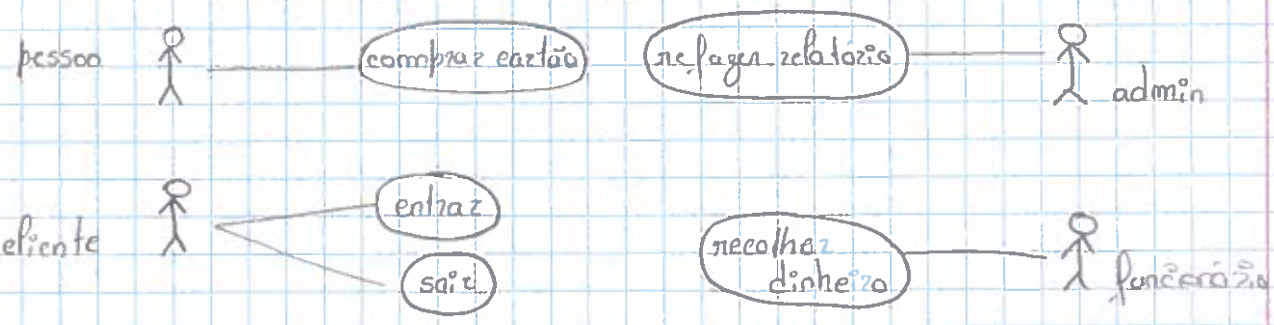
nota: comprar cartão envolve 2 pessoas → pode-se por a cardinalidade da relação ator — use case

↳ tal não existe entre atores



-> Exercício (7): herança entre Actor A e B

→ subway-ticket: agora com abstrações



↳ imaginando que agora queríamos que em cada 1000 bilhetes alguém recebia um bilhete 2 extends