2021/2 Orientação a Objetos - TP2

Faculdade UnB Gama
Profa. Fabiana Freitas Mendes

Aluguel de Bicicletas compartilhadas

Grupo: T3.2

1. LISTA DE OBJETOS, ATRIBUTOS E MÉTODOS

❖ Usuário

- ***Atributos:***email,senha.nome, identidade ,cpf, data de nascimento, telefone, idade
- ***Métodos:*** cadastrar, editar e deletar, listar, buscar,atualizar.

❖ Local

- ***Atributos:*** lugar de check in(local de saída), lugar de check out (local de chegada),Bicicleta(para que possa fazer check-in no local preciso ter uma bicicleta nesse local)
- ***Métodos:*** cadastrar, editar, deletar,atualizar,ver,listar.

❖ Tempo

- ***Atributos:*** hora de check-in, hora de check-out, tempo total gasto
- ***Métodos:*** cadastrar, editar, deletar,ver,atualizar.

❖ Bicicleta

- ***Atributos:***, tipo de bicicleta(normal ou elétrica) , disponibilidade (se há a bicicleta disponível para se fazer o check-in em uma determinada estação)
- ***Métodos:***Cadastrar, editar e deletar,atualizar, ver, buscar

❖ Pedido

- ***Atributos:*** preço por hora, preço final, usuário,bicicleta,tempo
- ***Métodos:*** ver,atualizar,listar.

❖ Forma de pagamento

- ***Atributos:*** usuário, pedido,bandeira(qual banco),número do cartão, código de segurança, tipo de pagamento (crédito ou débito)
- ***Métodos:*** Cadastrar, editar, deletar,atualizar.

- **Pessoa (Classe abstrata)**
- **Atributos:**nome, identidade ,cpf, data de nascimento, telefone, idade
- **Métodos:** cadastrar, editar e deletar

2. LISTA DE FUNCIONALIDADES DO SOFTWARE

2.1 Requisitos Funcionais

- **RF1:** Deve ser possível realizar CRUD de Usuário
- **RF2:** Deve ser possível realizar CRUD de Local
- **RF3:** Deve ser possível realizar CRUD de tempo
- **RF4:**Deve ser possível ver os tipos de bicicletas disponíveis para um determinado local e fazer o CRUD de escolha deste “produto”
- **RF5:** Deve ser capaz de listar todos os pedidos cadastrados na lista de pedidos no software
- **RF6: É** possível fazer um CRUD de forma de pagamento.
- **RF7:** Deve ser capaz de listar o nome de todos os usuários cadastrados na lista de usuários no software
- **RF8:** Deve ser capaz de listar todos os locais cadastrados na lista de locais cadastrados no software.
- **RF9:** Deve ser possível buscar por clientes utilizando seu nome ou CPF.
- **RF10:** Deve ser possível buscar quais as bicicletas disponíveis para cada local.
- **RF11** O software deve possuir um conjunto de dados pré-cadastrados.
- **RF12**: Deve ser capaz cadastrar, editar e deletar “pessoa” e com elas se fazer o cadastro como usuario.

2.2 Requisitos Não Funcionais

- **RNF1:** O software deve ser desenvolvido em Java
- **RNF2:** O software deve ser desenvolvido utilizando o paradigma orientado a objetos
- **RNF3:** A interação com o usuário deverá ser feita por meio de interface gráfica

- ❖ **RNF4:** O software desenvolvido será para ambiente desktop

2.3 Prioridade dos Requisitos

Prioridade	Requisito(s)
1	RF1,RF2,RF3,RF4,RF6,RF9,RF11,RF12, RNF1, RNF2, RNF3, RNF4
2	RF5,RF8,RF10
3	RF7

TP3: Link do Diagrama e meu raciocínio para os relacionamentos.

- https://lucid.app/lucidchart/22b42d11-f7bf-427d-a832-9ec523f836ce/edit?invitationId=inv_82a5d26d-dcc9-40fd-8779-86a12e14ff76 (Link do novo Diagrama➜Mais atualizado).
- Bicicleta para local é uma **agregação,** pois posso ter uma estação sem bicicletas(não vai ser um local onde possa fazer check-in vista que não há bicicletas) assim como posso ter bicicletas paradas em uma estação ou em trânsito (a bicicleta continua existindo mesmo fora da estação). A multiplicidade é a ideia que em um mesmo lugar posso ter várias bicicletas, mas só posso ter uma bicicleta por lugar.
- Bicicleta para pedido é uma **Composição**, visto que para que eu tenha um pedido eu necessariamente preciso alugar uma bicicleta e sem bicicleta não há pedido. A multiplicidade é a ideia que cada usuário possa alugar apenas uma única bicicleta (para melhor ter controle das bicicletas vista que facilita encontrar o responsável por ela)
- Tempo para pedido é uma **Herança**, visto que meu preço é calculado por hora... então obrigatoriamente preciso do tempo total para poder calcular o valor total (mesma lógica de estacionamento de shopping) e consequentemente se eu herdar esses dados de tempo para pedido facilito a conta. A multiplicidade é de 1 tempo total para um pedido (terei o valor final do aluguel com esse tempo total).

- De pedido para tempo é uma **Associação**, vista que tempo é atributo para o pedido(pois precisamos do tempo total para que possa ser feito o pedido)
- De pedido para usuário é uma **Associação**, visto que usuário é atributo para pedido (eu preciso de um usuário para que possa alugar a bicicleta). A multiplicidade é que eu vou ter um único usuário para cada pedido, mas cada usuário pode fazer infinitos pedidos.
- De pedido para forma de Pagamento é uma **Composição**, visto que não existe forma de pagamento sem que exista um pedido (pois não teria o que pagar e nem porque eu cadastrar um cartão já que não há compras). A multiplicidade é de um pagamento para uma forma de pagamento escolhida para esse pedido (não preciso pagar sempre com o mesmo cartão e nem com o mesmo tipo de pagamento todas as vezes).
- De forma de pagamento para pedido é uma **Associação**, visto que o pedido é um atributo para a forma de pagamento (eu preciso de um pedido para que eu possa pagar).
- De forma de pagamento para usuário é uma **Agregação,** visto que eu herdo os outros dados do usuário para poder cadastrar seu cartão (naturalmente quando se cadastra cartão em qualquer site você precisa dos dados pessoais do usuário) com isso iremos automatizar o processo. A multiplicidade é de que cada pagamento é de um único usuário mas, cada usuário pode fazer infinitos pagamentos (vista que pode fazer infinitos pedidos).
  ➜Maybe eu preciso de uma classe como “Pessoa/entidade” para se ter uma classe abstrata para ser de herança de usuário. E essa classe receberia como atributos algumas características das pessoas por trás dos usuários como: Nome, Data de Nascimento, idade,Telefone e identidade...e usuário herdaria isso tudo e teria o email e senha como atributos extras.