

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS – 1º SEMESTRE
MATUTINO – 2017

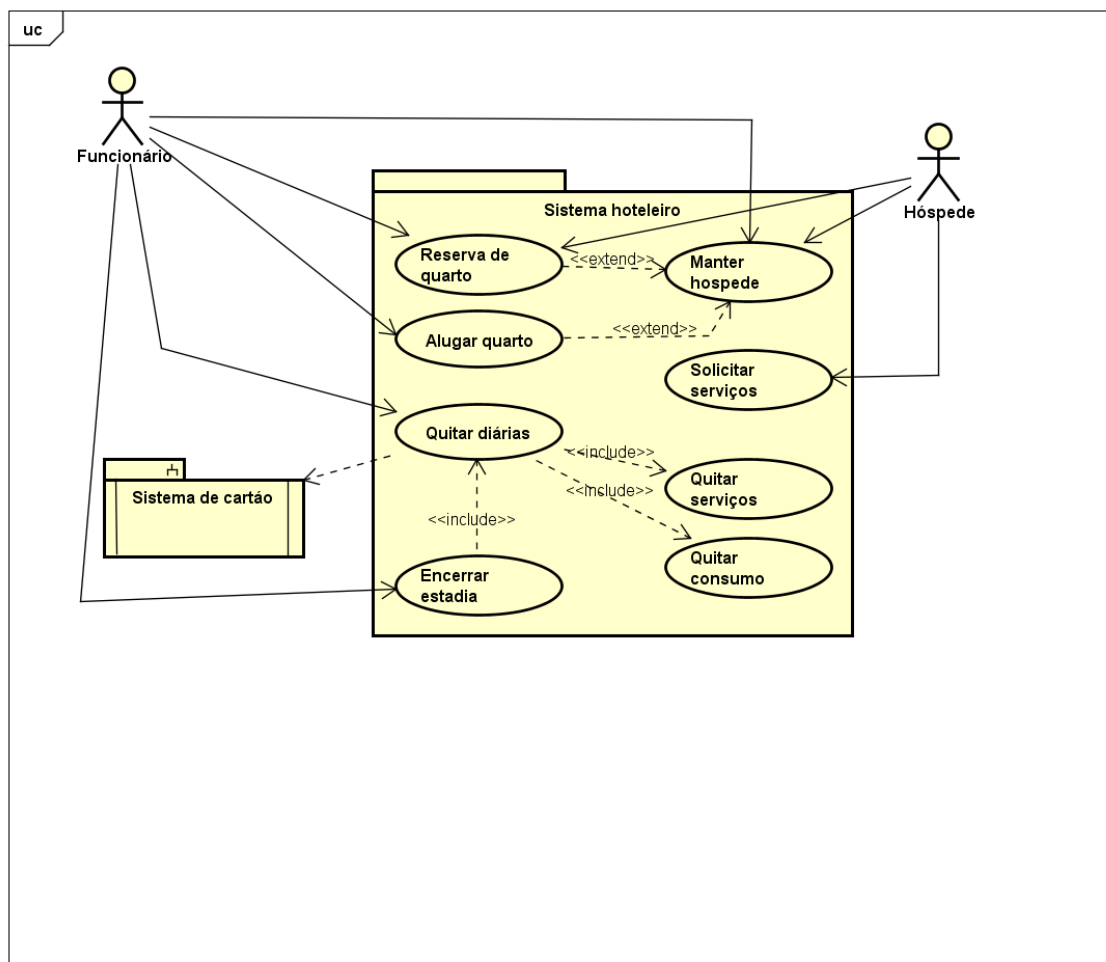
LISTA 1

PROFESSOR: WILSON VENDRAMEL

DISCIPLINA: ENGENHARIA DE SOFTWARE III

ANDREI LUCAS GONÇALVES	RA: 1680481521025
DANIEL SCORDAMAGLIO	RA: 1680481521030
DIEGO DE MELO GONZAGA	RA: 1680481521036
ELIAS KYOHARU SANAI	RA: 1680481521016
KEVENY MARTINS	RA: 1680481521026
RICARDO SILAS MONTEMURRO	RA: 1680481521037

1. Construa um diagrama de casos de uso.



2. Especifique o CSU1 apresentando os fluxos (cenários) principal, alternativo e de exceção. Utilizar um template que seja inteligível. Os casos de uso <<extend>> e <<include>> (se houver) podem ser especificados junto com o caso de uso principal.

CSU01. Reservar Quarto

Pré-condição:

Possuir cadastro

Atores Envolvidos:

Hóspede

Descrição:

O hóspede faz a reserva de um quarto com permanência no hotel

Cenários:

Principal:

1. O hóspede realiza login
2. O hóspede define a data de permanência no hotel
3. O hóspede seleciona qual quarto dos disponíveis deseja
4. O hóspede confirma os dados.

Alternativo:

A1:

1. O hóspede não se lembra do login/senha
2. O hóspede entra em contato com o funcionário para que o mesmo faça o registro do hóspede.
3. Iniciar Manter Hóspede

Exceção:

E1:

O hóspede tenta diversas vezes login/senha incorretamente bloqueando o acesso do mesmo.

Pós-condição:

É registrado a reserva do quarto desejado pelo hóspede

3. Especifique o CSU2 apresentando os fluxos (cenários) principal, alternativo e de exceção. Utilizar um template que seja inteligível. Os casos de uso <<extend>> e <<include>> (se houver) podem ser especificados junto com o caso de uso principal.

CSU02. Alugar Quarto

Pré-condição:

Hóspede estar cadastrado no sistema

Atores Envolvidos:

Hóspede

Funcionário

Descrição:

Faz com que o hóspede registre uma estadia e tenha acesso a um quarto por um período de tempo

Cenários:

Principal:

1. O hóspede solicita um quarto ao funcionário
2. O funcionário pergunta se há reserva
3. O hóspede informa os dados da reserva
4. O funcionário faz a confirmação da reserva
5. O funcionário registra a alocação do quarto.

Alternativo:

A3:

1. O hóspede informa que não há reserva
2. O funcionário informa os quartos disponíveis
3. Retornar ao passo 3 do principal

Exceção:

E A2:

1. O funcionário informa que não há quartos disponíveis para alocação

Pós-condição:

É registrado a alocação do quarto em questão

4. Especifique o CSU3 apresentando os fluxos (cenários) principal, alternativo e de exceção. Utilizar um template que seja inteligível. Os casos de uso <<extend>> e <<include>> (se houver) podem ser especificados junto com o caso de uso principal.

CSU03. Manter hóspede

Pré-condição:

O hóspede precisa apresentar documento de identidade

Atores Envolvidos:

Hóspede

Funcionário

Descrição:

O funcionário faz a manutenção dos dados do hóspede

Cenários:

Principal:

1. O funcionário verifica se o hóspede possui cadastro
2. O hóspede apresenta o documento de identidade ao funcionário
3. O funcionário questiona sobre alterações cadastrais
4. O hóspede informa quais dados foram alterados
5. O funcionário altera os dados informados no sistema

Alternativo:

A2:

1. O hóspede informa que não possui cadastro
2. O funcionário pede o documento de identidade
3. O funcionário realiza o primeiro cadastro
4. O funcionário finaliza o cadastro e informa os dados necessário do cadastro de cliente

Exceção:

Pós-condição:

São alterados/criados os dados do hóspede

5. Especifique o CSU4 apresentando os fluxos (cenários) principal, alternativo e de exceção. Utilizar um template que seja inteligível. Os casos de uso <<extend>> e <<include>> (se houver) podem ser especificados junto com o caso de uso principal.

CSU04. Solicitar Serviço

Pré-condição:

Hóspede possuir um alocação ainda válida

Atores Envolvidos:

Hóspede

Descrição:

O hóspede registra um serviço que deseja que seja realizado

Cenários:

Principal:

1. O hóspede entra no sistema
2. O hóspede faz autenticação
3. O hóspede seleciona na lista o que deseja
4. O hóspede informa detalhes dependendo do serviço
5. O hóspede confirma o serviço

Alternativo:

A5.

1. O hóspede cancela o pedido

Exceção:

Pós-condição:

É registrado para pagamentos o serviço pedido.

6. Especifique o CSU5 apresentando os fluxos (cenários) principal, alternativo e de exceção. Utilizar um template que seja inteligível. Os casos de uso <<extend>> e <<include>> (se houver) podem ser especificados junto com o caso de uso principal.

CSU05. Quitar diárias

Descrição:

Os valores referentes aos gastos da estadia são pagos pelo hóspede .

Principal:

1. O hóspede pede quitação da diária ao funcionário.
2. O funcionário verifica as pendências do hóspede.
3. O funcionário informa os valores que o hóspede deve pagar.
4. O hóspede paga pelas diárias.

Alternativo:

A2:

1. O funcionário verifica os serviços utilizados pelo hóspede.
2. O funcionário verifica os produtos consumidos pelo hóspede.
3. Valores são calculados pelo funcionário.
4. Retorna ao Fluxo Principal 2.

Exceção:

E4:

O hóspede não consegue quitar as diárias por não aprovação do cartão.

O hóspede não consegue quitar as diárias por não possuir dinheiro suficiente.

7. Especifique o CSU6 apresentando os fluxos (cenários) principal, alternativo e de exceção. Utilizar um template que seja inteligível. Os casos de uso <<extend>> e <<include>> (se houver) podem ser especificados junto com o caso de uso principal.

CSU06. Encerrar Estadia

Descrição:

A estadia do hóspede é finalizada e assim vai embora.

Principal:

1. O hóspede pede encerramento da estadia
2. Funcionário avisa dos débitos referentes aos serviços e produtos consumidos
3. Hóspede paga pelos gastos
4. O funcionário atualiza a alocação do quarto
5. Estadia é finalizada

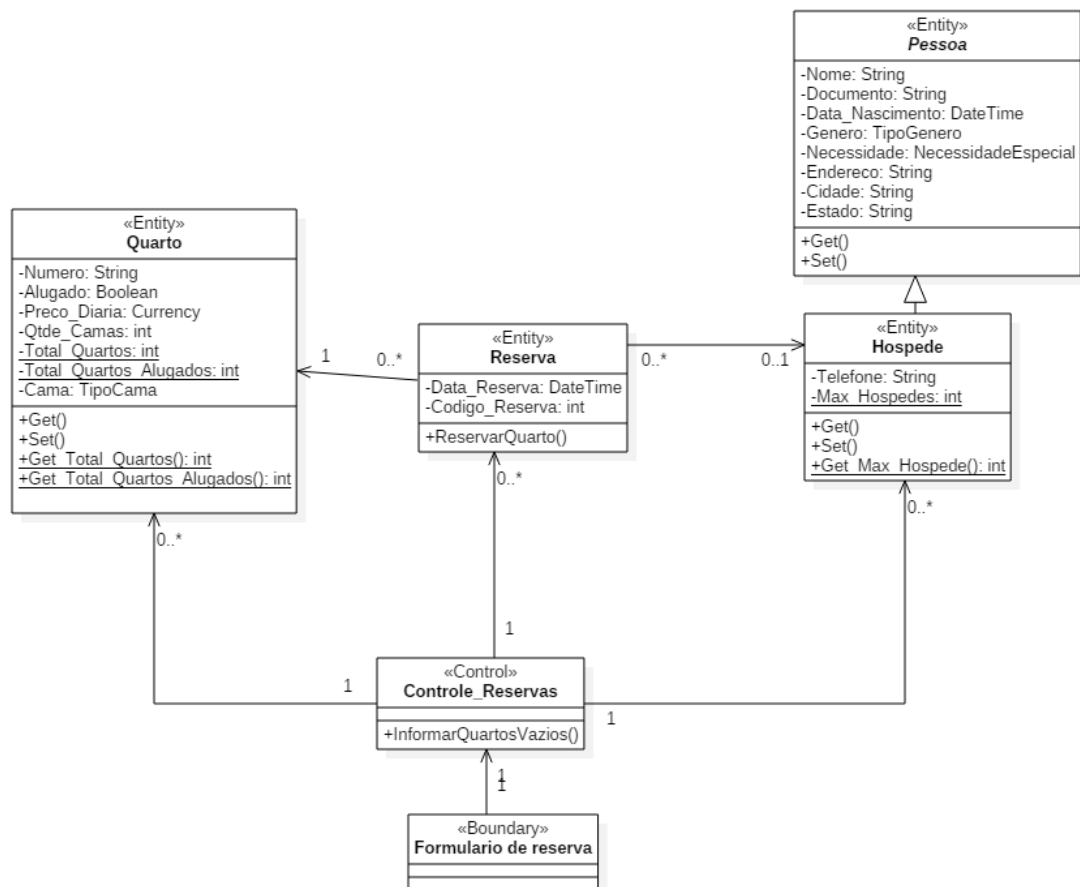
Alternativo:

Exceção:

E3: Transação do cartão não aprovada

E3: Hóspede não possui dinheiro para pagar estadia

8. Modele uma VCP para o caso de uso utilizando a categorização BCE para o CSU01. A classe de controle deve apresentar um método no mínimo e as classes de entidade devem apresentar seus devidos tributos e um método no mínimo.



9. Faça o protótipo de interface de usuário para a classe <<boundary>> do CSU01.



Reserva de quarto

Data de entrada:

Prevista de saída:

Quartos a reservar:

Quarto 01:

Luxo:   

Valor total de diária: **R\$ 0,00**

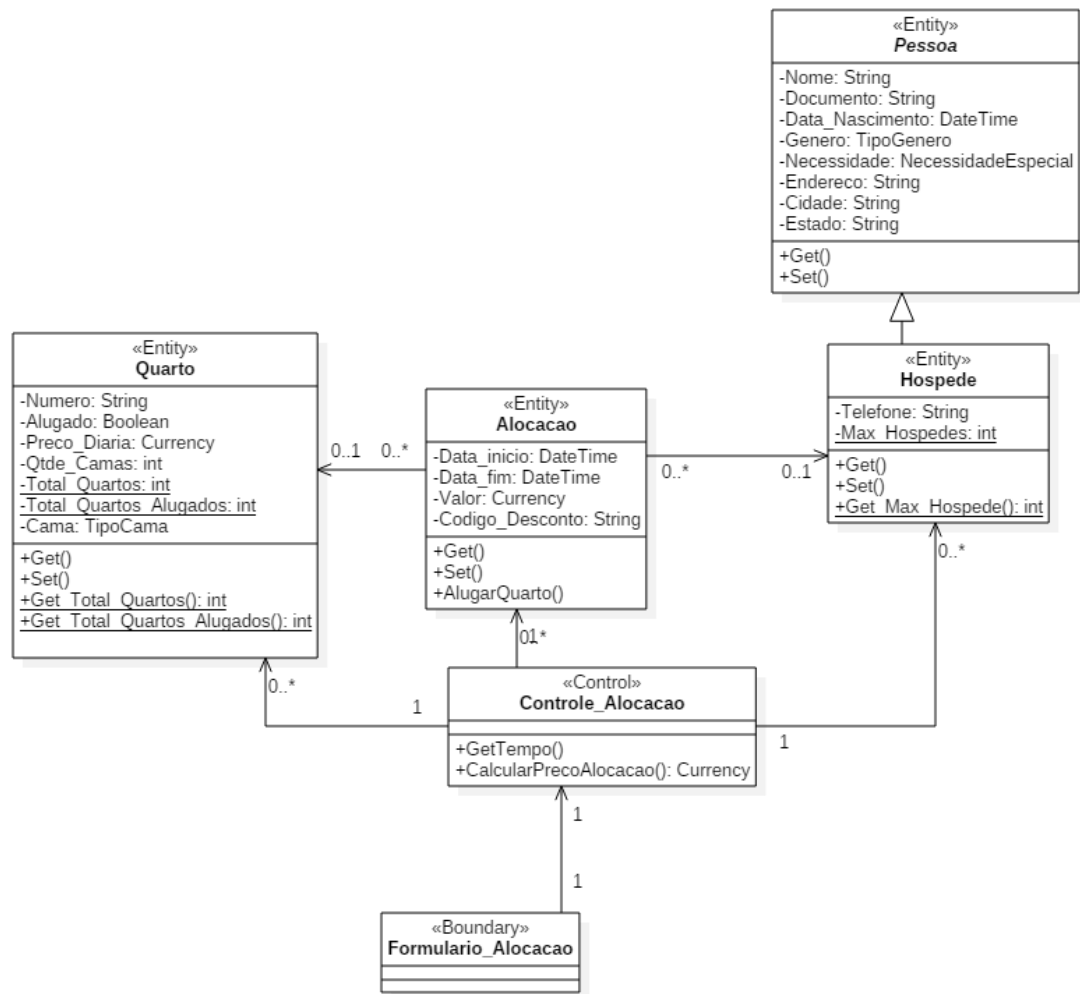
Cupom de desconto:

Total de desconto: **R\$ 0,00**

Descrição do apartamento:

Imagens:

10. Modele uma VCP para o caso de uso utilizando a categorização BCE para o CSU02. A classe de controle deve apresentar um método no mínimo e as classes de entidade devem apresentar seus devidos atributos e um método no mínimo.



11. Faça o protótipo de interface de usuário para a classe <<boundary>> do CSU02.

Alugar quarto

Quartos

Reservado: ☒ Não ☐ Sim

Código de reserva:

Quarto:

Luxo: ☒ ☐ ☐

Descrição:

Hospede

Nome:

Documento:

Informações:

Nome:

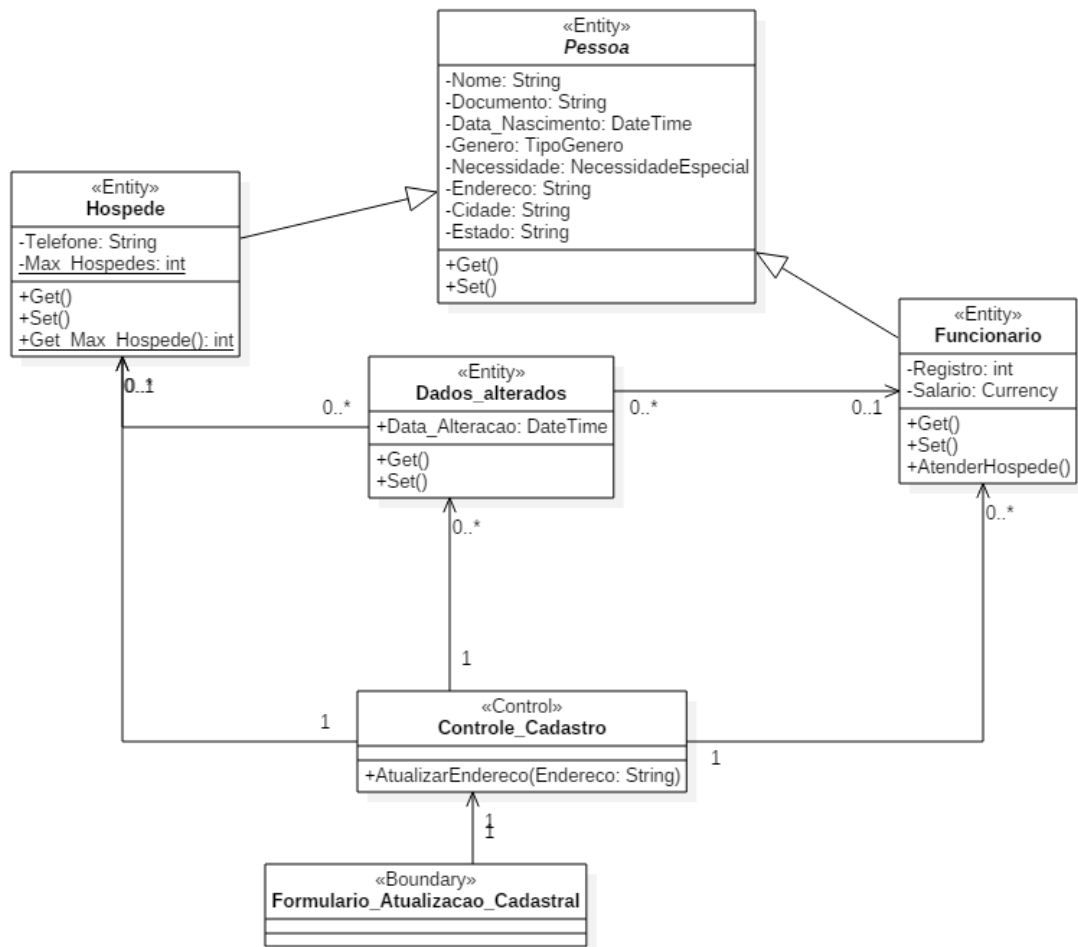
Endereço:

Telefone:

Cidade:

Estado:

12. Modele uma VCP para o caso de uso utilizando a categorização BCE para o CSU03. A classe de controle deve apresentar um método no mínimo e as classes de entidade devem apresentar seus devidos atributos e um método no mínimo.



13. Faça o protótipo de interface de usuário para a classe <<boundary>> do CSU03.

Manter hóspede

Nome:

Nascimento:

Sexo:

Necessidades especiais:

Quais:

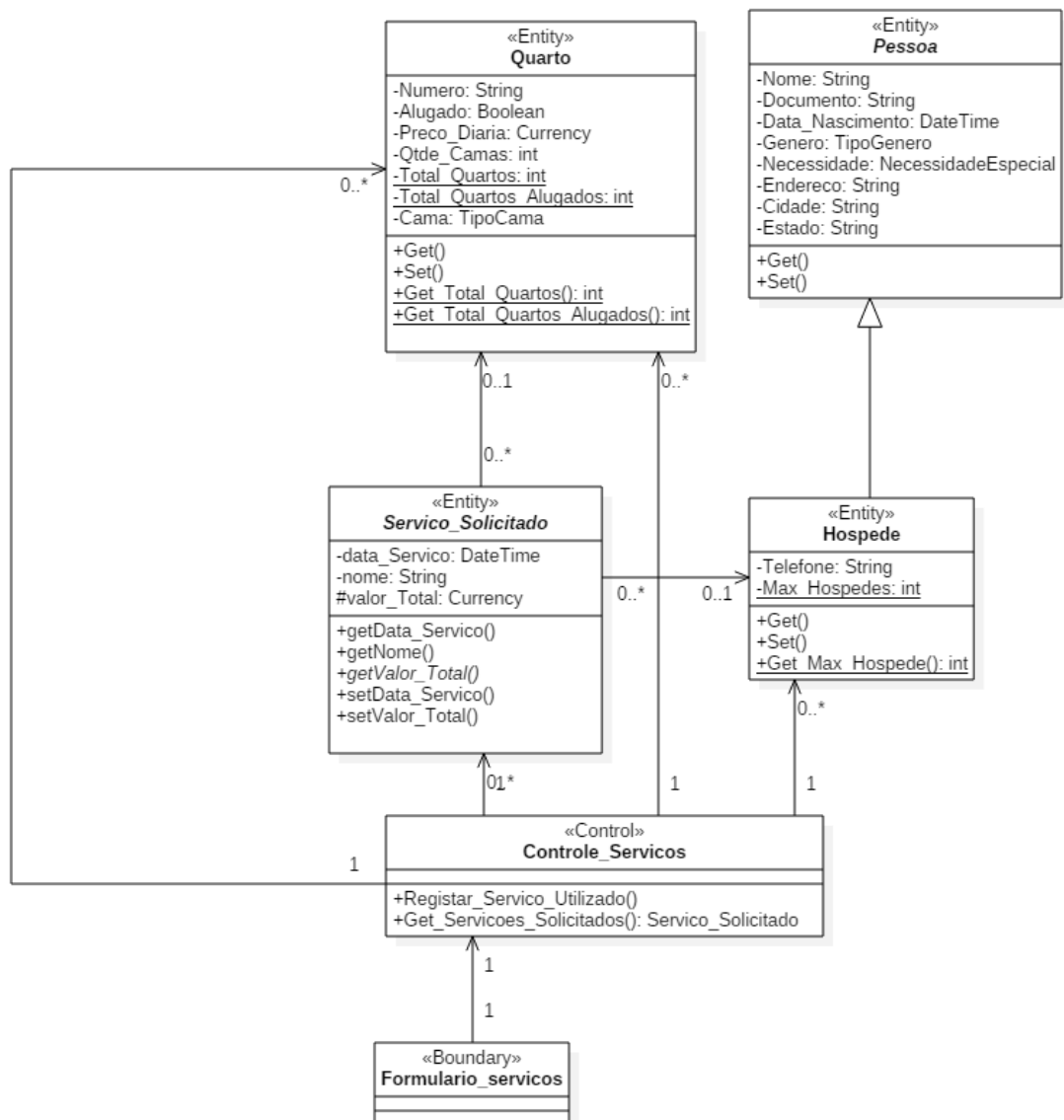
Endereço:

Cidade:

Estado:

Telefone:

14. Modele uma VCP para o caso de uso utilizando a categorização BCE para o CSU04. A classe de controle deve apresentar um método no mínimo e as classes de entidade devem apresentar seus devidos atributos e um método no mínimo.



15. Faça o protótipo de interface de usuário para a classe <<boundary>> do CSU04.

Solicitar Serviços

Quarto: (selecione) ▼

Serviços

Tipo: (selecione) ▼

Descrição:

Inserir no pedido

Cozinha

Bebidas: (selecione) ▼

Pratos: (selecione) ▼

Petiscos: (selecione) ▼

Descrição dos pratos:

Inserir no pedido

Serviços solicitados

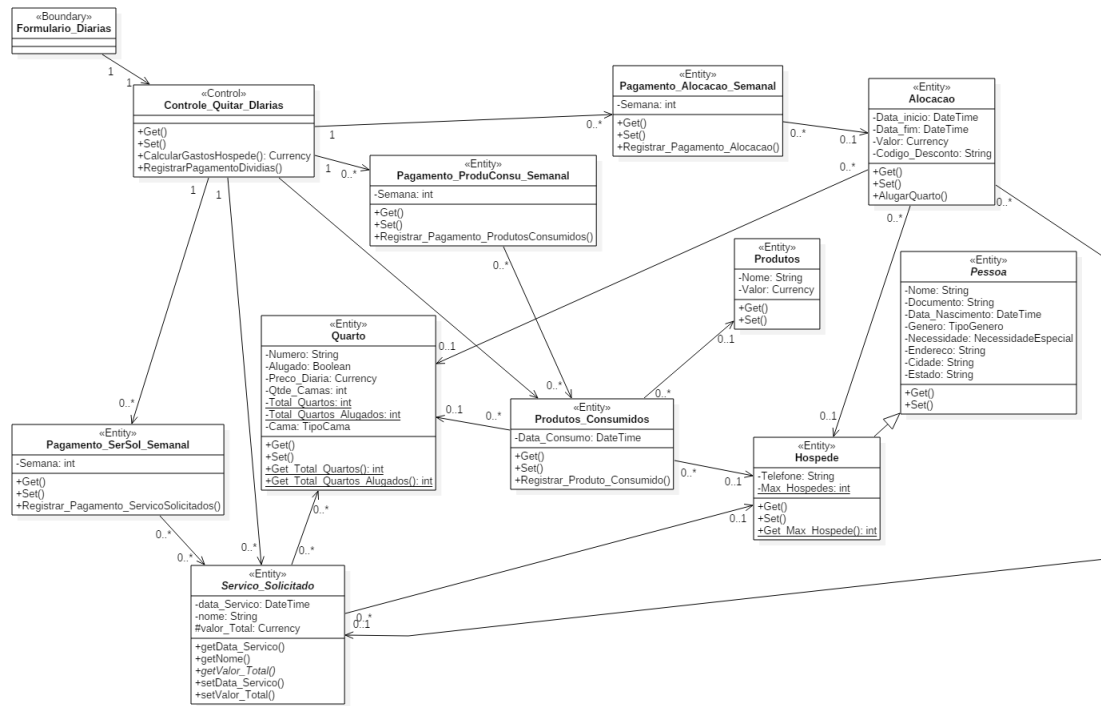
Valor total do pedido: R\$ 0,00

Deletar item do pedido

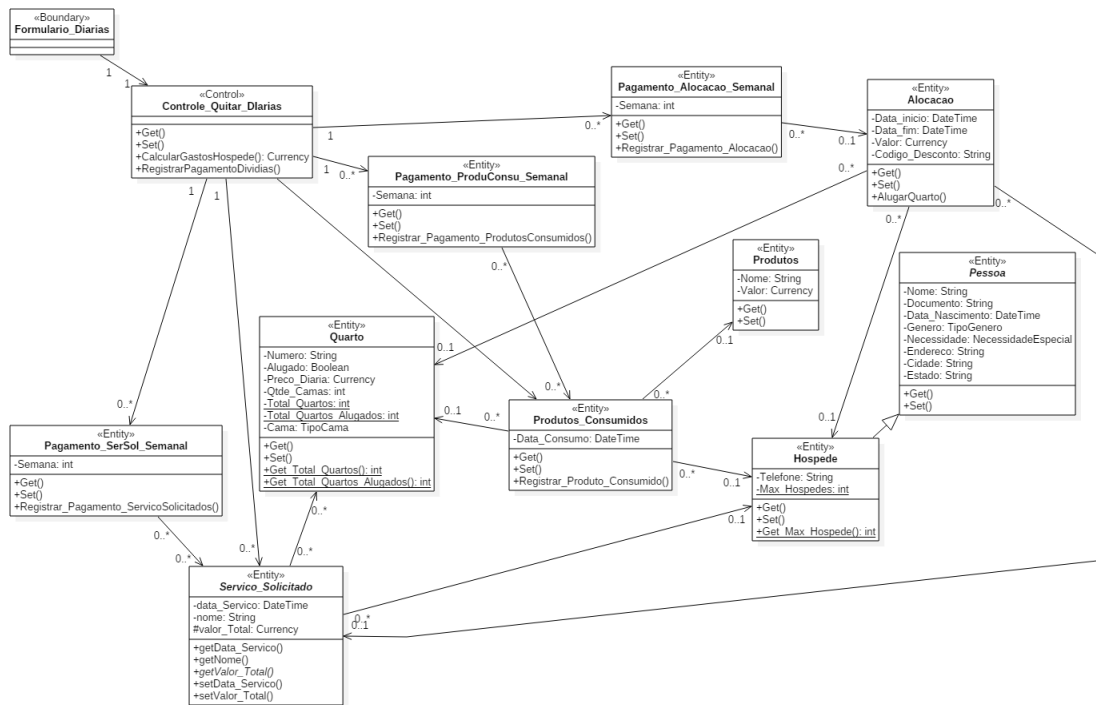
Fechar pedido

Cancelar pedido

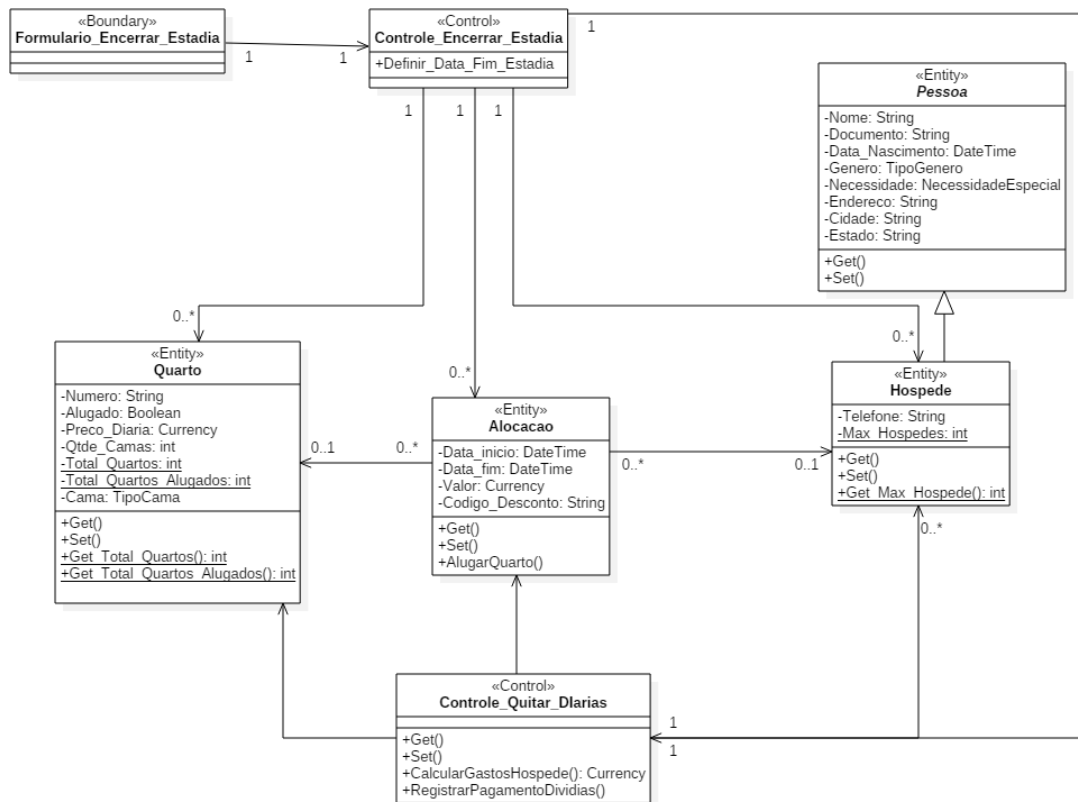
16. Modele uma VCP para o caso de uso utilizando a categorização BCE para o CSU05. A classe de controle deve apresentar um método no mínimo e as classes de entidade devem apresentar seus devidos atributos e um método no mínimo.



17. Faça o protótipo de interface de usuário para a classe <<boundary>> do CSU05.



18. Modele uma VCP para o caso de uso utilizando a categorização BCE para o CSU06. A classe de controle deve apresentar um método no mínimo e as classes de entidade devem apresentar seus devidos atributos e um método no mínimo.



19. Faça o protótipo de interface de usuário para a classe <<boundary>> do CSU06.

Encerrar hospedagem

Quarto: (selecione) ▼

Hospede

Informações:

Nome:

Endereço:

Telefone:

Cidade:

Estado:

Consumo em aberto

Descrição:

Valor de consumo: R\$ 0,00
Valor de diária: R\$ 0,00

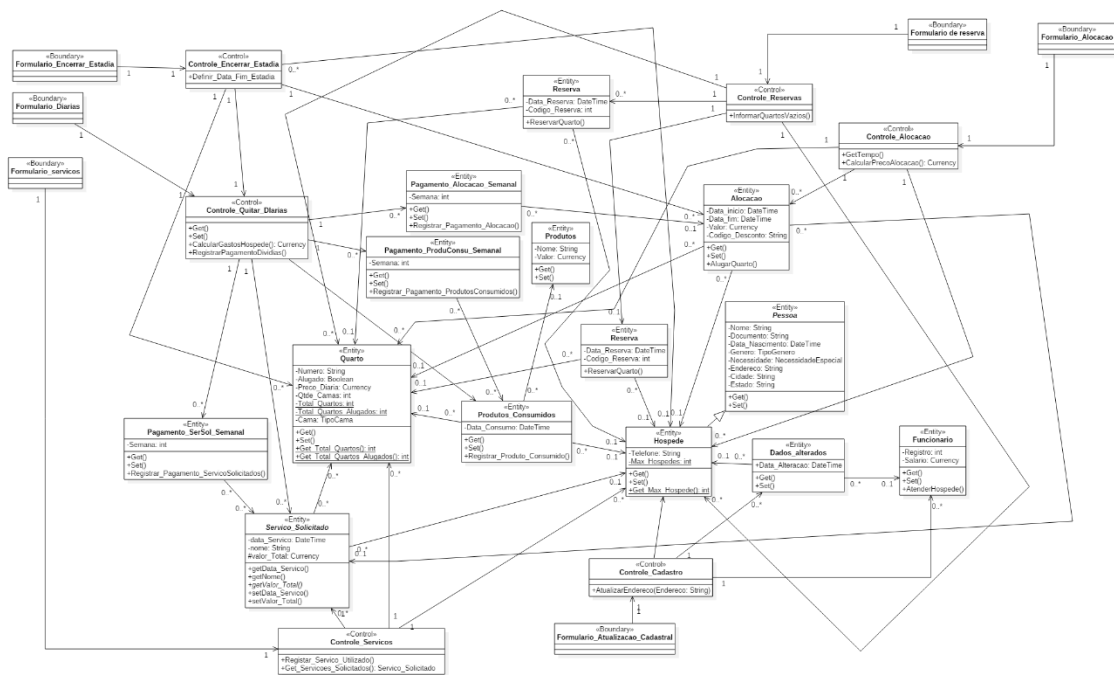
Total a ser pago: R\$ 0,00

Forma de pagamento: ☒ Dinheiro ☐ Cartão de débito ☐ Cartão de crédito

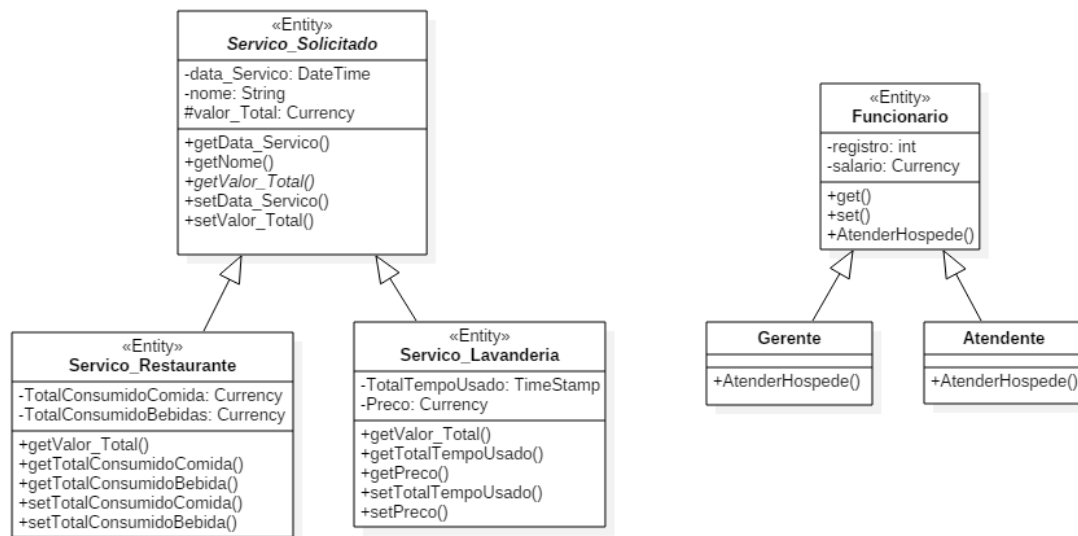
Realizar check-out

Cancelar

20. Modele um diagrama de classes de projeto a partir das VCPs modeladas e mantenha a utilização da categorização BCE. Os devidos atributos e métodos devem continuar sendo exibidos.



21. Modele duas relações de gen/espec e ative o princípio de polimorfismo universal de inclusão em cada uma delas. Justifique a razão de existência de cada gen/espec e das operações polimórficas.



A solicita servico, é uma classe que pode representar muitos tipos de serviços, levando em consideração a expansão do sistema.

O cálculo do valor total de um serviço pode dependendo do caso, pode ter muitas dependencias de outras variáveis.

Na classe Servico_Restaurante se considerou que um usuário faz o cálculo total de comida e bebida para chegar ao total do serviço, mas poderiam existir mais variáveis que não foram inseridas no sistema.

Justificativa da Funcionario:

Tendo um caso onde o Funcionario poderá agir de forma diferente atendendo seus clientes, por exemplo, o gerente tendo mais permissões no sistema.

É possível criar uma herança com o polimorfismo de inclusão universal usando funcionario, gerente e atendente, apesar de não ser o mais usual

22. As relações de gen/espec violam o Princípio de Liskov? Justifique a tua resposta.

Não, porque os serviços foram subdivididos. Ou seja, mesmo os serviços de Lavadeira e Restaurante, continuam sendo serviços.

E também Gerente e Atendente também são Funcionários.

23. Apresente a estrutura básica de código em JAVA, C# ou C++ para implementar as relações de gen/espec e as operações polimórficas.

```
abstract class Solicita_Servico
{
    private DateTime data_Servico;
    private string nome;
    protected decimal valor_Total;

    public DateTime getData_Servico()
    {
        return data_Servico;
    }
    public string getNome()
    {
        return nome;
    }
    public abstract decimal getValor_Total();
    public void setData_Servico(DateTime value)
    {
        data_Servico = value;
    }
    public void setNome(string value)
    {
        nome = value;
    }
}
```

```
class Servico_Restaurante : Solicita_Servico
{

```



```

    public decimal totalConsumidoComida;
    public decimal totalConsumidoBebida;

    public override decimal getValor_Total()
    {
        return totalConsumidoComida + totalConsumidoBebida;
    }
    public decimal getTotalConsumidoComida()
    {
        return totalConsumidoComida;
    }
    public decimal getTotalConsumidoBebida()
    {
        return totalConsumidoBebida;
    }
    public void setTotalConsumidoComida(decimal value)
    {
        totalConsumidoComida = value;
    }
    public void setTotalConsumidoBebida(decimal value)
    {
        totalConsumidoBebida = value;
    }
}

```

```

class Servico_Lavanderia : Solicita_Servico
{
    public TimeSpan totalTempoUsado;
    public decimal preco;
}

```

```

public override decimal getValor_Total()
{
    return (decimal)totalTempoUsado.TotalMinutes * preco;
}

public double getTotalTempoUsado()
{
    return totalTempoUsado.TotalMinutes;
}

public decimal getPreco()
{
    return preco;
}

public void setTotalConsumidoComida(TimeSpan value)
{
    totalTempoUsado = value;
}

public void setTotalConsumidoBebida(decimal value)
{
    preco = value;
}
}

```

```

public abstract class Funcionario
{
    public int registro;
    public decimal salario;

    public abstract void AtenderHospede();
}

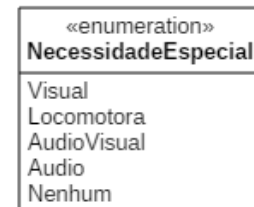
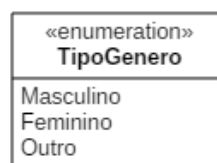
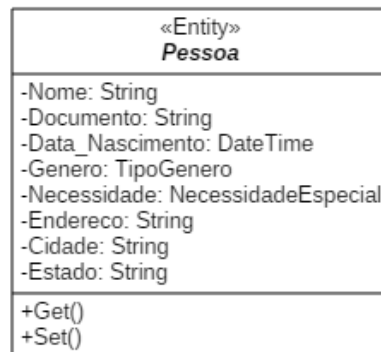
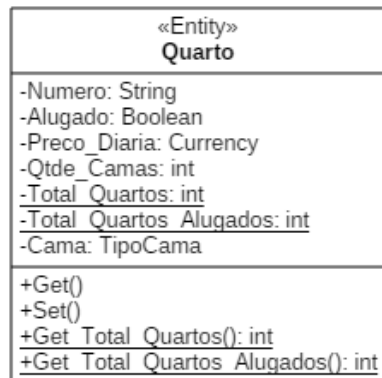
public class Atendente : Funcionario

```

```
{
    public override void AtenderHospede()
    {
        //Atender Hospede como Atendente
    }
}

public class Gerente : Funcionario
{
    public override void AtenderHospede()
    {
        //Atender Hospede como Gerente
    }
}
```

24. Modele três classes enumeradas e utilize as mesmas como tipos de atributos. Justifique a existência de cada uma das classes enumeradas modeladas.



- TipoCama para saber o tipo de acomodação
- TipoGenero para conhecimento do mesmo da Pessoa visto que são valores fechados
- NecessidadeEspecial para ter como conhecimento prévio se a Pessoa precisa de algo de acordo com seu estado

25. Modele seis membros estáticos, sendo três atributos e três métodos. Justifique a criação de existência de cada um dos membros estáticos modelados.

«Entity» Quarto
-Numero: String -Alugado: Boolean -Preco_Diaria: Currency -Qtde_Camas: int -Total Quartos: int -Total Quartos Alugados: int -Cama: TipoCama
+Get() +Set() +Get Total Quartos(): int +Get Total Quartos Alugados(): int

«Entity» Hospede
-Telefone: String -Max Hospedes: int
+Get() +Set() +Get Max Hospede(): int

- Total Quartos para saber qual a quantidade física de quartos
- Total Quartos Alugados para saber quantos dos quartos estão alugados

OBS: A união de ambos acima permite calcular a porcentagem de uso dos quartos

- Max Hospede é quantidade máxima que é comportada por motivos de segurança que o hotel permite

Parte B: Atividade de Abstração

