

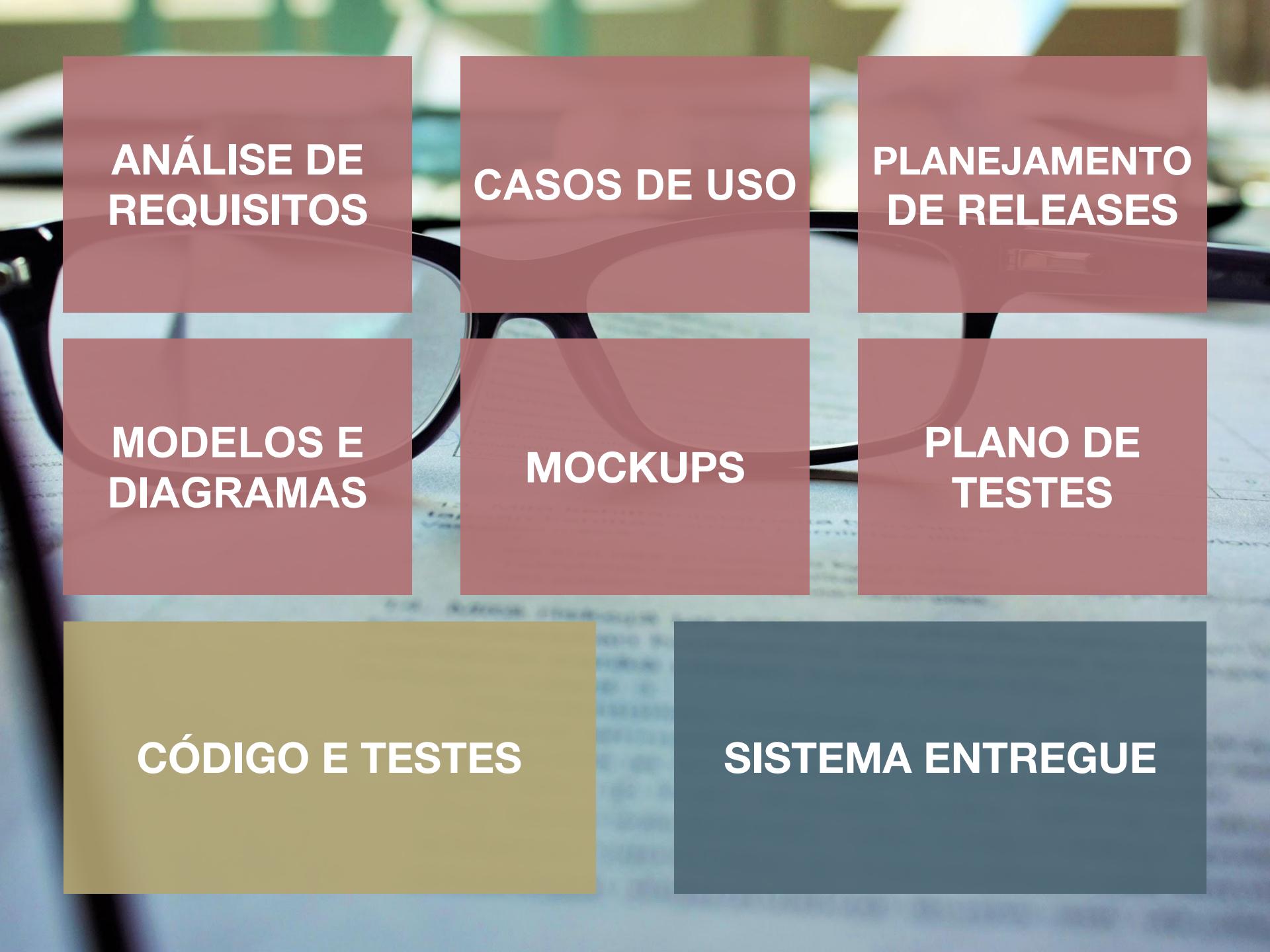
# AGENDA E

## Sistema Integrado de Agendamento de Recursos

---

**Equipe: Brasil**

Caio Cesar      Lucas Lopes  
Pedro Boueke      Vinicius Alves



ANÁLISE DE  
REQUISITOS

CASOS DE USO

PLANEJAMENTO  
DE RELEASES

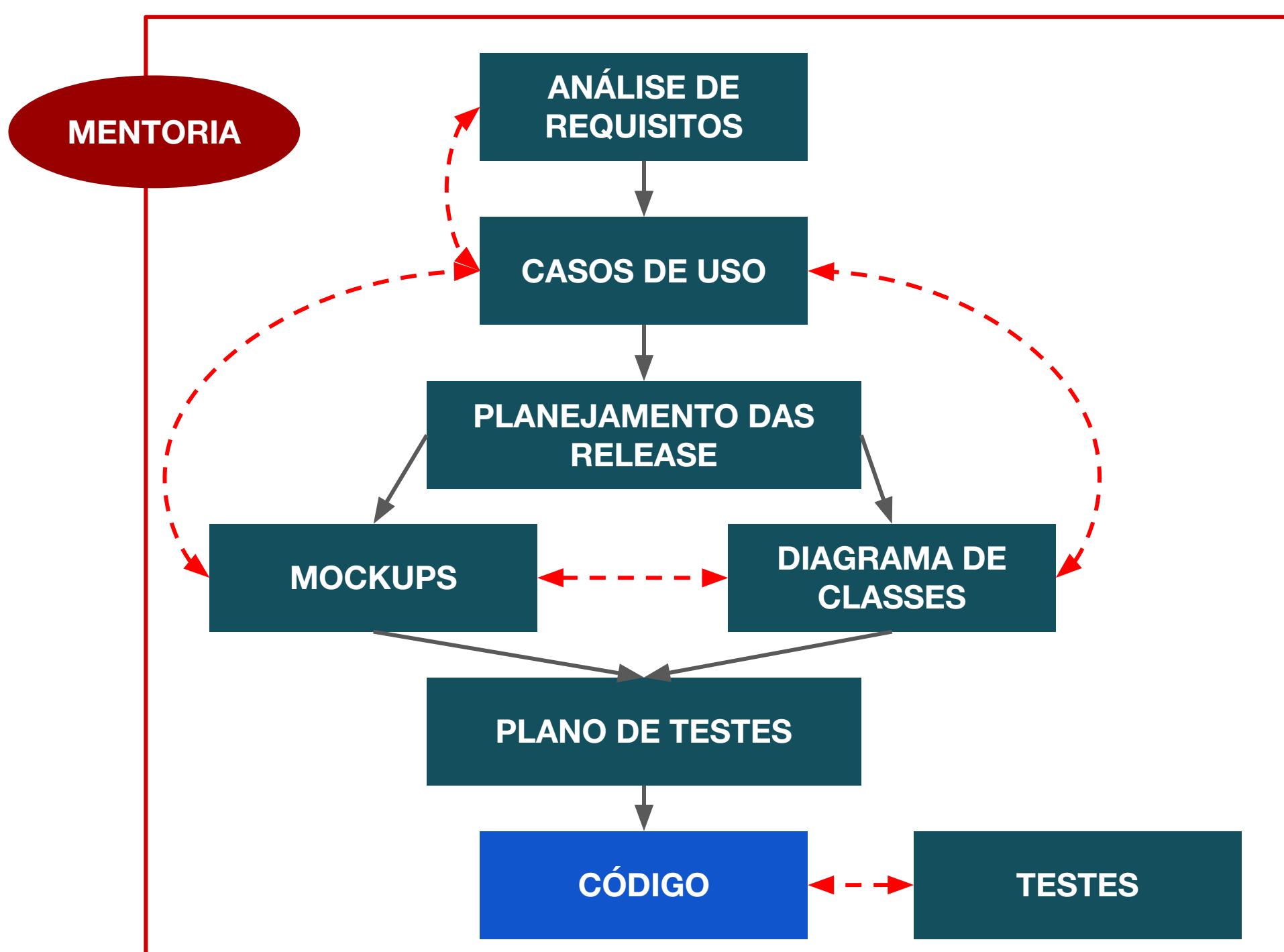
MODELOS E  
DIAGRAMAS

MOCKUPS

PLANO DE  
TESTES

CÓDIGO E TESTES

SISTEMA ENTREGUE



# ANÁLISE DE REQUISITOS



**16** requisitos funcionais

**10** requisitos não  
funcionais

**7** definições

administrador de departamento,  
administrador geral, funcionário,  
recurso departamento, UFRJ

# CASOS DE USO

**20**

Casos de uso

**17**

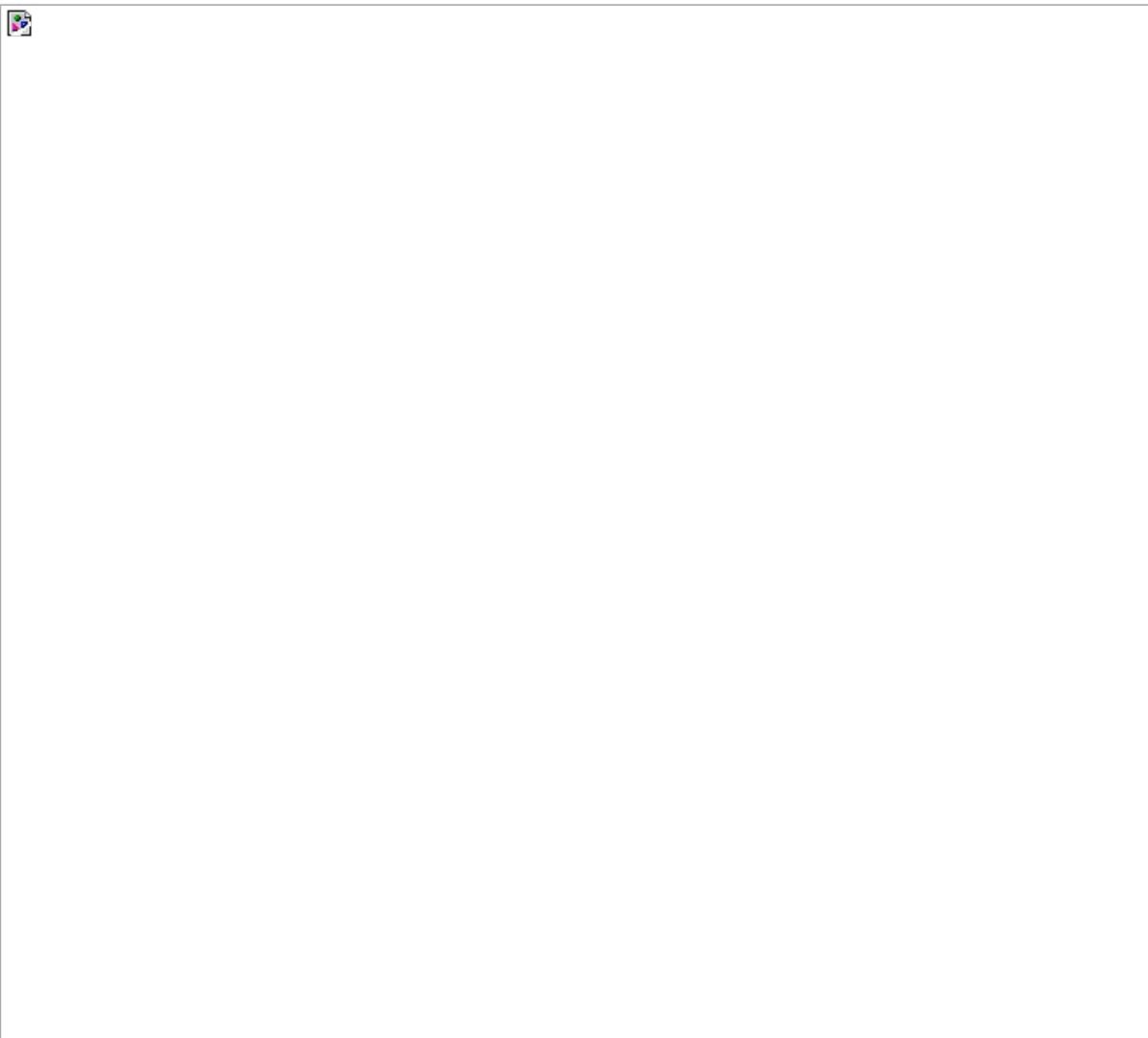
fluxos de  
exceção

**13**

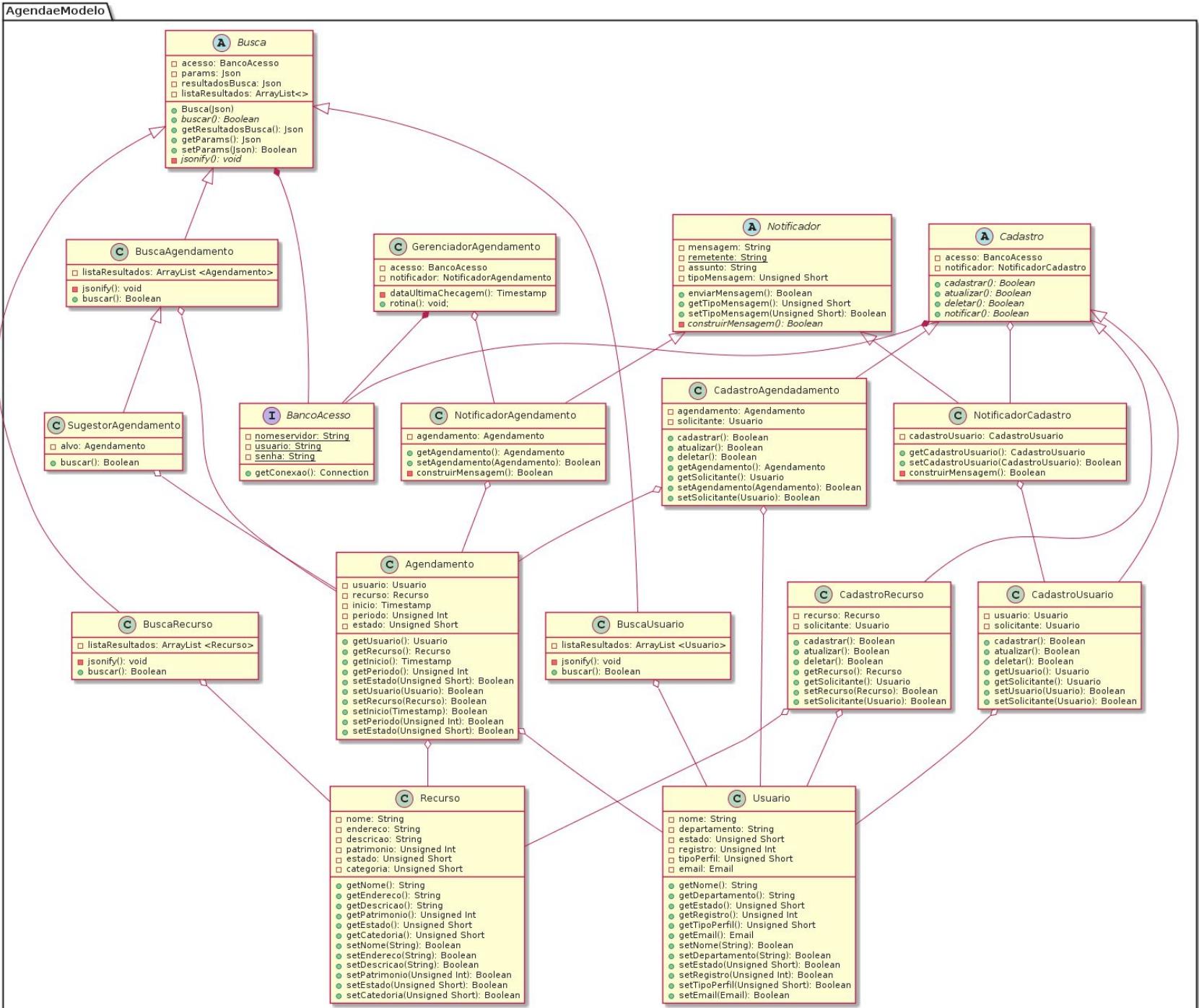
fluxos  
alternativos

Danville  
NEXT 3 EXITS

# DIAGRAMA DE CONTEXTO

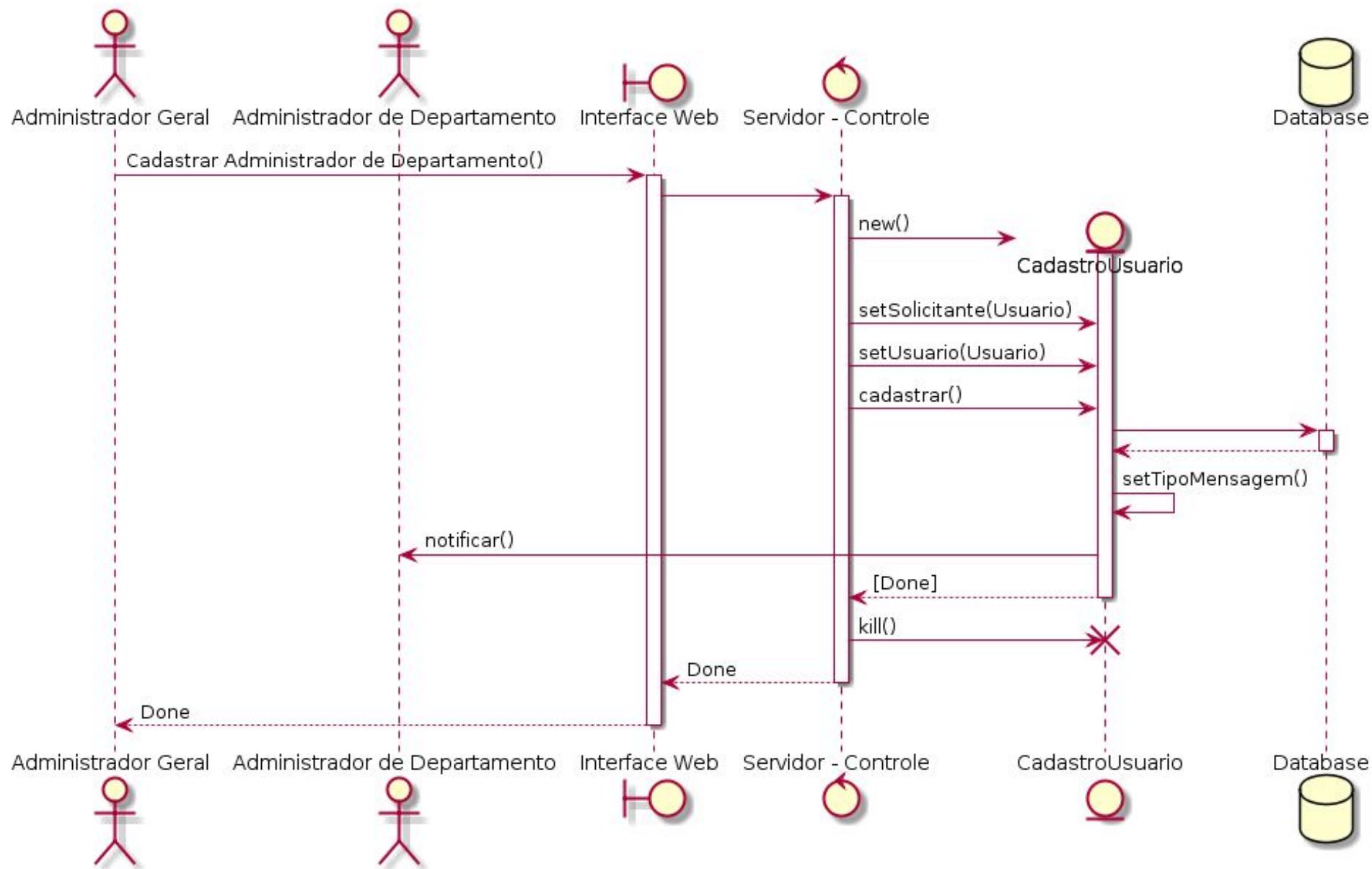


# DIAGRAMA DE CLASSES UML



# DIAGRAMA DE SEQUÊNCIA

## EXEMPLO



# MOCKUPS

**Agenda UFRJ**

<http://www.agendoe.ufrj.br>

**Agendaê**

Início Meu Perfil

Buscar recurso

Recursos > Localização Categoría Novo Recurso

Nome	Localização	Código de patrimônio	Categoria	Estado
Mouse	H218 - Centro de Tecnologia	IDX28134	Periférico	Disponível
Projetor	H218 - Centro de Tecnologia	IDX28192	Ferramenta	Disponível
Ploto	D238 - Centro de Tecnologia	FGX282	material de aula	Indisponível
Microfone	D238 - Centro de Tecnologia	FTVX282	Ferramenta	Indisponível
Osciloscópio	A324 - Instituto de Física	IFX281	Ferramenta	Mantenção

anterior 1 2 3 4 - próximo

**Item desabilitado para funcionários comuns:**

**Ao clicar em um recurso o usuário é redirecionado para a página específica desse recurso com mais detalhes:**

**« FEB 2006 »**

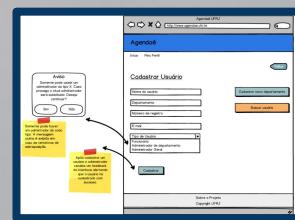
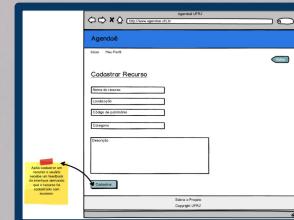
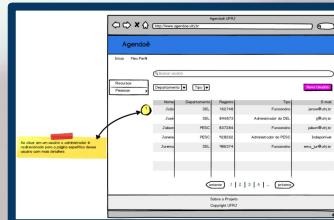
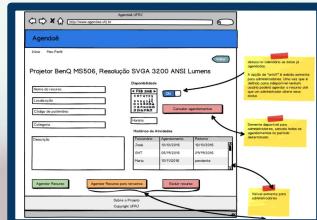
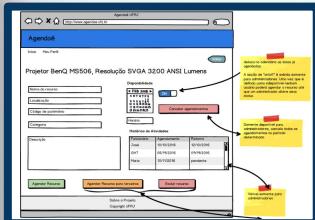
**Ao selecionar uma data, somente recursos que estão disponíveis naquela data poderão ser exibidos.**

**O usuário poderá também escolher um intervalo de tempo como data.**

**O estado do produto deve ser diferenciado por cor, de modo a tornar a identificação desse atributo por parte do usuário imediata.**

- Disponível: verde
- Indisponível: vermelho
- Manutenção: amarelo

**Somente visível para administradores do sistema.**



# AMBIENTE DE DESENVOLVIMENTO



Aplicações:  
**app/ e admin/**

Model:  
**models/\***

View:  
**templates/app/\***

Controller:  
**views.py**

Conteúdo estático:  
**static/\***

# TESTES

## TESTES UNITÁRIOS

Verificam valores de entrada e saída



Classes de teste

Chamadas por

Classes do modelo

## TESTES FUNCIONAIS

Verificam o comportamento do sistema



Classes de teste

implementadas ou replicadas

Funções do sistema

## RESULTADOS

python3 manage.py  
runserver

↓  
Execução de todos os testes

Logs de teste

↓  
Avaliados pelo programador

Resultados da rodada de testes

# EXEMPLO CÓDIGO BUSCA DE RECURSO

DERIVADA



BASE



```
class Busca (models.Model):
    params = "{}"
    def buscar():
        return False
    class Meta:
        abstract = True
        managed = False
        app_label = 'app'
```

```
class BuscaRecurso (Busca):
    def buscar (self):
        if self.params == '{}':
            return None
        query = lambda:None
        query.__dict__ = json.loads(self.params)
        if query.type == 'match':
            try:
                return Recurso.objects.get(patrimonio = query.id)
            except ObjectDoesNotExist:
                return "DoesNotExist ERROR"
        elif query.type == 'complex':
            res = Recurso.objects.exclude(nome="")
            if query.texto.strip():
                res = res.filter(nome_icontains=query.texto)
                res = res | res.filter(descricao_icontains=query.texto)
            if len(query.categorias) > 0:
                res = res.filter(categoria_in=query.categorias)
            if len(query.enderecos) > 0:
                res = res.filter(endereco_in=query.enderecos)
            if len(query.disponibilidades) > 0:
                res = res.filter(estado_in=query.disponibilidades)
            try:
                return res
            except DoesNotExist:
                return "DoesNotExist ERROR"
        return None
    class Meta:
        managed = False
        app_label = 'app'
```

# EXEMPLO TESTE CADASTRO DE RECURSO

```
class RecursoTests( TestCase ):    ← CLASSE DE TESTE
    test_registers_number = 0
    nome = "nome"
    patrimonio_inicial = 1
    endereco = "endereco"
    categoria = "000"
    descricao = "test resource"

    def setUp (self):
        self.test_registers_number = randint(10, 100)
        print ("Testing for " + str(self.test_registers_number) + " resources")
        patrimonio = self.patrimonio_inicial
        cr = CadastroRecurso()
        for i in range(self.test_registers_number):
            cr.cadastrar (self.nome, patrimonio, self.endereco, self.categoria, self.descricao)
            patrimonio += 1

    def test_cadastro(self):
        patrimonio = self.patrimonio_inicial + self.test_registers_number + 1
        cr = CadastroRecurso()    ← MÉTODO TESTADO
        # simple (pass)
        self.assertEqual(cr.cadastrar (self.nome, patrimonio, self.endereco, self.categoria, self.descricao), True)
        # same id (fail)
        self.assertEqual(cr.cadastrar (self.nome, patrimonio, self.endereco, self.categoria, self.descricao), False)
        # delete last register|
        self.assertEqual(cr.deletar (patrimonio), True)
        self.assertEqual(cr.deletar (patrimonio), False)
        # 200 character name (pass)
        patrimonio += 1
        nome = "nxе40kpXTvCPa0T88aJSXemKYWZDXv06ssZfE4gW0xsJgsHKLRWIgamYlYceoZ5hcHGVDAeLZQNJm4tEJxcVypHhV0liPtI9" +
        "mInlcм0MQemP1qS9qPf1I8bVgniH3Y20FXF5t0PmX4NTz2q73YfL660sMYtz7JVQQZfBR8jchSUEo2PRr0BFHuxj52rNMy2ToJ49BvMP"
        self.assertEqual(cr.cadastrar (self.nome, patrimonio, self.endereco, self.categoria, self.descricao), True)
        self.assertEqual(cr.deletar (patrimonio), True)
        # 201 character name (pass)
        patrimonio += 1
        nome += "A2323"
        self.assertEqual(cr.cadastrar (self.nome, patrimonio, self.endereco, self.categoria, self.descricao), True)
        self.assertEqual(cr.deletar (patrimonio), True)
```

# EXEMPLO RESULTADO DE TESTE

```
Destroying test database for alias 'default'...
jawa@Tatooine:~/Documents/repositories/ESEGROUP/Final/Brasil$ 
c^Cjawa@Tatooine:~/Documents/repositories/ESEGROUP/Final/Brasil$ python3 manage.py test app/tests/
Creating test database for alias 'default'...
Testing for 20 resources
.Testing for 11 resources
.Testing for 57 resources
.
Ran 3 tests in 0.272s
OK
Destroying test database for alias 'default'...
jawa@Tatooine:~/Documents/repositories/ESEGROUP/Final/Brasil$ 
jawa@Tatooine:~/Documents/repositories/ESEGROUP/Final/Brasil$ python3 manage.py test app/tests/
Creating test database for alias 'default'...
Testing for 13 resources
.Testing for 84 resources
FTesting for 61 resources
.
=====
FAIL: test_cadastro (teste_recurso.RecursoTests)

Traceback (most recent call last):
  File "/home/jawa/Documents/repositories/ESEGROUP/Final/Brasil/app/tests/teste_recurso.py", line 32
  , in test_cadastro
    self.assertEqual(cr.cadastrar (self.nome, patrimonio, self.endereco, self.categoria, self.descri
cao), False)
AssertionError: True != False
Ran 3 tests in 0.308s
FAILED (failures=1)
Destroying test database for alias 'default'...
jawa@Tatooine:~/Documents/repositories/ESEGROUP/Final/Brasil$ 
```

SUCESSO

FALHA

# AGENDAE

CADASTRO

SOBRE

UFRJ

Versão 0.0.1

Agendae © 2016

# AGENDAE

Sistema integrado de agendamento de recursos da UFRJ

admin

.....

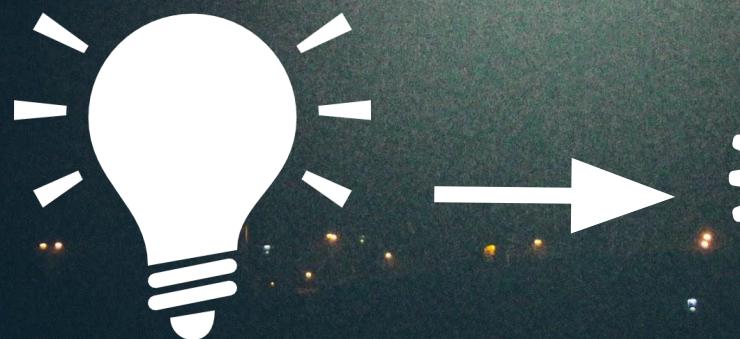
ENTRAR



## DEMONSTRAÇÃO

# ESFORÇOS E RESULTADOS

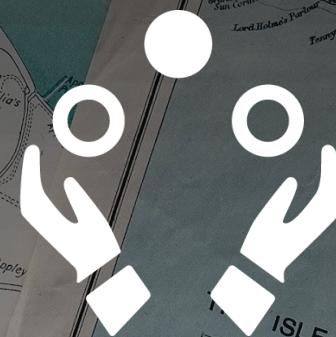
# AGENDA E



# DISTRIBUIÇÃO DE TAREFAS



TRABALHO TOTALMENTE REMOTO E  
ASSÍNCRONO.



DISTRIBUIÇÃO DE TAREFAS BASEADA EM  
EXPERIÊNCIA E DISPONIBILIDADE.



DISTRIBUIÇÃO NÃO CONTABILIZADA, BASEADA  
EM CONFIANÇA E DEPENDÊNCIA MÚTUA.

# CONSTRUÇÃO DE MODELOS

DIAGRAMA DE  
CLASSES

DIAGRAMA DE  
CASOS DE USO



# APERFEIÇOAMENTO



MELHORAR A COMUNICAÇÃO DENTRO DA EQUIPE DE DESENVOLVIMENTO.

MELHOR FORMALIZAR AS TAREFAS A SEREM CONCLUÍDAS DE FORMA QUE TODOS POSSAM PARTICIPAR SEMPRE QUE DISPONÍVEIS SEGUINDO A FORMALIZAÇÃO DA TAREFA.



BUSCAR E ESTUDAR METODOLOGIAS EXISTENTES PARA DISTRIBUIÇÃO DE TAREFAS, DE FORMA A NOS ADAPTARMOS A UM MODELO PRODUTIVO MAIS EFICIENTE

# DIFICULDADES DE GERENCIAMENTO



**POUCO CONTATO SÍNCRONO REDUZ A CAPACIDADE DE TOMADA DE DECISÕES EM CONJUNTO.**

**INVIABILIDADE DE ADOTAR OUTRA ESTRATÉGIA ORGANIZACIONAL DEVIDO À INDISPONIBILIDADE DE HORÁRIOS DOS INTEGRANTES.**



# DIFICULDADES GERAIS

AGENDA ACADÊMICA APERTADA E PRAZOS  
NÃO NEGOCIADOS



MOTIVAÇÃO



LINGUAGEM DE PROGRAMAÇÃO IMPOSTA.



# CONSIDERAÇÕES FINAIS

