Professor Guilherme Ditzel Patriota

DICAS PARA CRIAÇÃO DE UM DIAGRAMA DE CLASSES

PRÉ-REQUISITOS

- Possuir levantamento de requisitos do sistema
- Possuir lista de requisitos funcionais do sistema
- Possuir lista de casos de uso e/ou diagrama de casos de uso

CRIAÇÃO DE LISTA DE CLASSES

Diagrama de classes não aparecem os atores

Tem foco no programador que precisa entender o que programar

Uma Classe é um molde para criação de um objeto

Um objeto tem capacidade de realizar tarefas no sistema

Criamos cada classe como um pequeno conjunto de dados (atributos) e funções (métodos)

Atributos são coisas, adjetivos e dados que um objeto possui

Métodos são ações, tarefas, funções que um objeto pode executar

Para criar uma classe, unimos todos os métodos e atributos que sejam parte de um grupo de funções em comum

EXEMPLOS DE REQUISITOS FUNCIONAL

RF001:

O aplicativo deve **permitir acesso** por **identificação facial** ao **aplicativo do banco**.

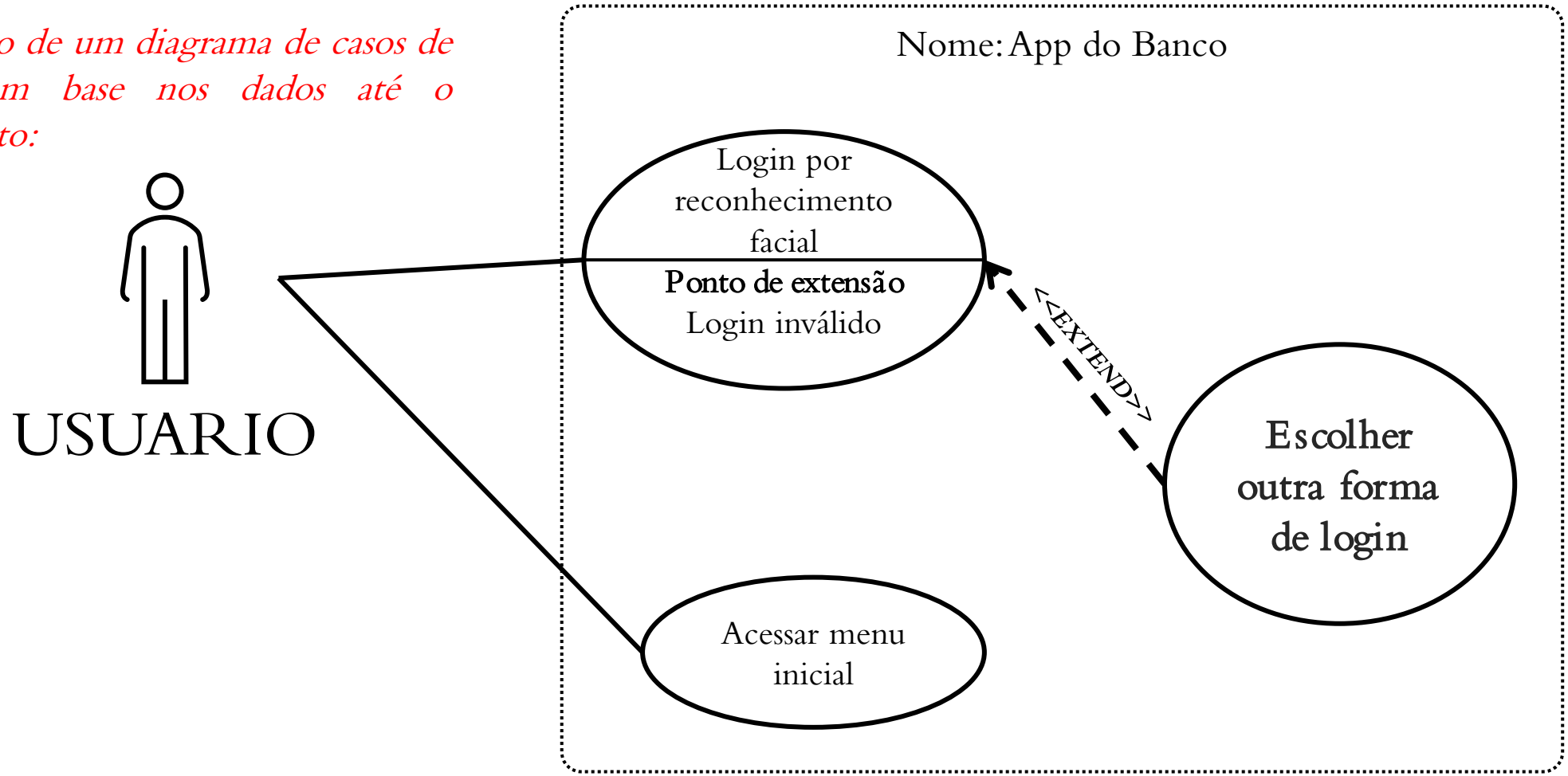
ANÁLISE DE REQUISITOS FUNCIONAL

DEFINIÇÃO DO CASO DE USO

- Do nosso requisito funcional de exemplo, podemos encontrar o seguinte caso de uso:
  1. UC001 – Um usuário, ao clicar no ícone do aplicativo do nosso banco, verá uma tela inicial que automaticamente tentará realizar a leitura facial para desbloqueio do aplicativo. Caso a leitura facial seja identificada como uma identificação válida, o aplicativo fará o login e mostrará a tela principal da aplicação.

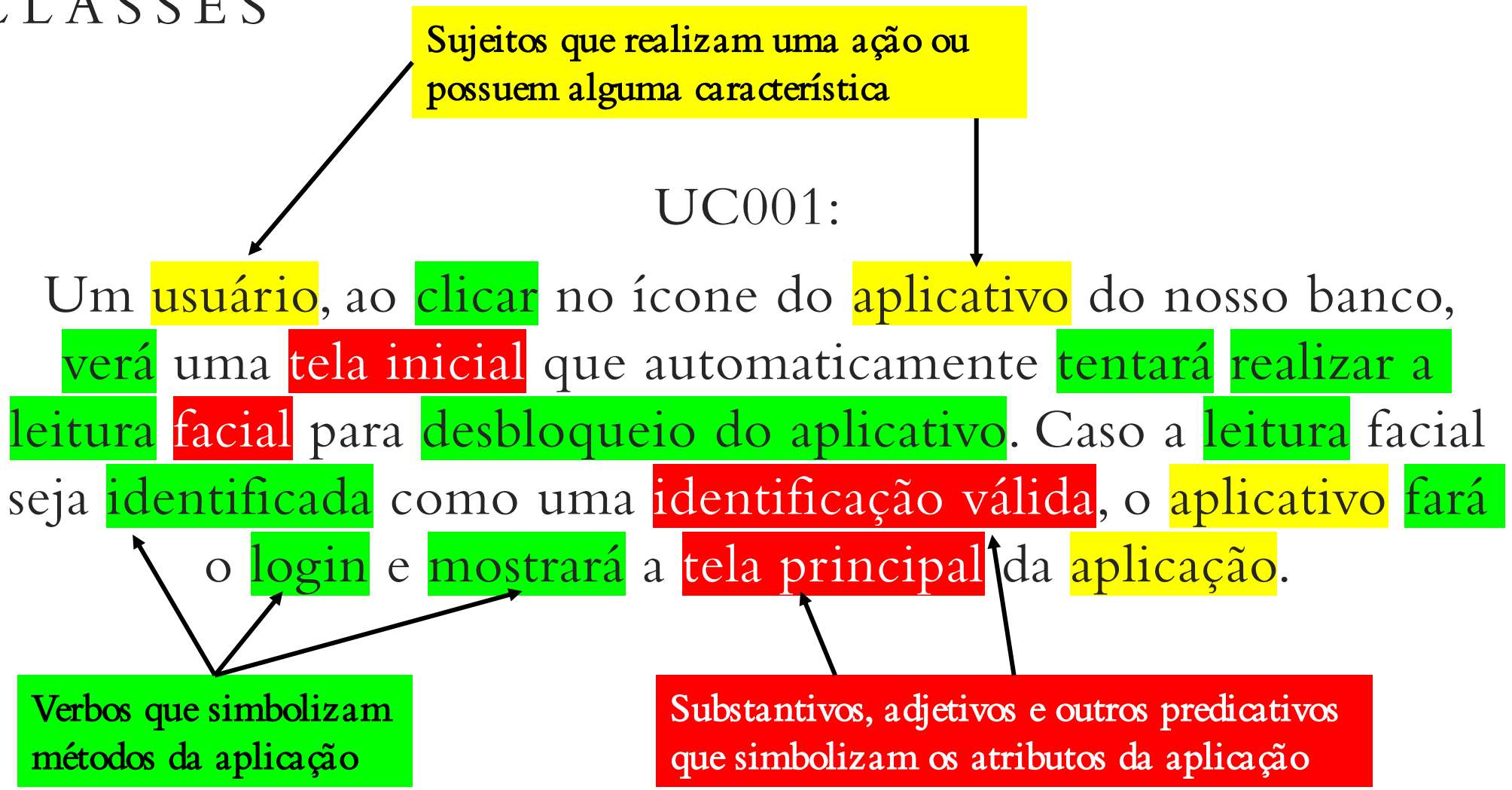
Este caso de uso pode gerar mais de um diagrama de casos de uso e o requisito também dá margem para geração de outros casos de uso.

Exemplo de um diagrama de casos de uso com base nos dados até o momento:



Observe que este diagrama de casos de uso é bem simples e só mostra os acessos que o usuário possui, independente do que vem antes ou depois. Ele não é um diagrama sequencial, de modo geral.
Este diagrama não nos permite mostrar grandes planejamentos para a estrutura do software.

ANÁLISE DO CASO DE USO PARA CLASSES



DEFINIÇÃO DE MÉTODOS

Passando para o diagrama de classes, temos que criar os conjuntos de métodos e atributos necessários para que o usuário consiga fazer o que dissemos que ele conseguirá fazer pelo diagrama de casos de uso.

Para isso, temos que identificar quais métodos, atributos e objetos são necessários para realizar as tarefas.

Tarefas que precisamos ter (cada uma é uma função/método diferente):

- acessarLogin()
- analisarFace()
- validarAcesso()
- mostrarTelaDeLogin()
- mostrarMenuPrincipal()
- alterarFormaDeLogin()

DEFINIÇÃO DE ATRIBUTOS

Passando para o diagrama de classes, temos que criar os conjuntos de métodos e atributos necessários para que o usuário consiga fazer o que dissemos que ele conseguirá fazer pelo diagrama de casos de uso.

Para isso, temos que identificar quais métodos, atributos e objetos são necessários para realizar as tarefas.

Dados que precisamos ter (cada um é um atributo diferente):

- nomeDeUsuario
- faceUsuario
- facesAutorizadas
- itensMenu
- tiposDeLogin
- loginEscolhido

CRIAÇÃO DE CONJUNTOS DE ATRIBUTOS E MÉTODOS

- Podemos pensar nestes objetos como pedaços de software e agrupar cada método e atributo em uma determinada Classe:

  Pedaço 1:

  - nomeDeUsuario: String
  - tiposDeLogin: Array[String]
  - loginEscolhido: String
  - acessarLogin()
  - alterarFormaDeLogin(tiposDeLogin):loginEscolhido

  SUGESTÃO DE NOME PARA ESTA CLASSE: **Usuario**

Perceba a padronização no formato dos nomes de atributos, métodos e classes. Sem acentuações e nem espaços. Com uso de mixedCase para atributos e métodos e CamelCase para as classes.

Suguestão de leitura (Python): https://peps.python.org/pep-0008/#naming-conventions

CRIAÇÃO DE CONJUNTOS DE ATRIBUTOS E MÉTODOS

- Podemos pensar nestes objetos como pedaços de software e agrupar cada método e atributo em uma determinada Classe:

  Pedaço 2:

  - faceUsuario: Binário
  - analisarFace():faceUsuario

  SUGESTÃO DE NOME PARA ESTA CLASSE: **LeitorDeFaces**

CRIAÇÃO DE CONJUNTOS DE ATRIBUTOS E MÉTODOS

- Podemos pensar nestes objetos como pedaços de software e agrupar cada método e atributo em uma determinada Classe:

  Pedaço 3:

  - faceUsuario: Binário
  - facesAutorizadas: Array[Binário]
  - validarAcesso(faceUsuario, facesAutorizadas): Booleano

  SUGESTÃO DE NOME PARA ESTA CLASSE: **ValidadorDeAcesso**

CRIAÇÃO DE CONJUNTOS DE ATRIBUTOS E MÉTODOS

- Podemos pensar nestes objetos como pedaços de software e agrupar cada método e atributo em uma determinada Classe:

  Pedaço 4:

  - tiposDeLogin: Array[String]
  - loginEscolhido: String
  - itensMenu: Array[String]
  - mostrarMenuPrincipal(itensMenu)
  - mostrarTelaDeLogin(loginEscolhido)

  SUGESTÃO DE NOME PARA ESTA CLASSE: **ControleDoApp**

CRIAÇÃO DE CONJUNTOS DE ATRIBUTOS E MÉTODOS

Observação:

Saiba que todos estes métodos e atributos não foram criados de uma vez só.

Iniciamos com os mais óbvios, como "fazerLogin" ou "nomeDeUsuario" e ao criarmos cada Classe, perceberemos o que falta para fazermos ajustes.

Somente no final que incluímos os formatos de dados dos atributos e de entradas e saídas dos métodos.

POR FIM, NOSSAS CLASSES FICARAM ASSIM:

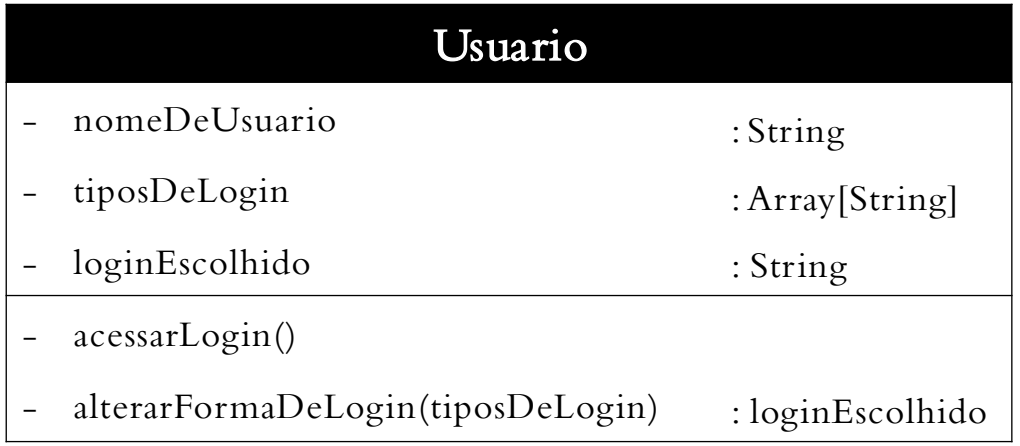
Usuario:

Atributos

- nomeDeUsuario: String
- tiposDeLogin: Array[String]
- loginEscolhido: String

Métodos

- acessarLogin()
- alterarFormaDeLogin(tiposDeLogin):loginEscolhido



POR FIM, NOSSAS CLASSES FICARAM ASSIM:

LeitorDeFaces:

Atributos

- faceUsuario: Binário

Métodos

- analisarFace():faceUsuario

LeitorDeFaces	
- faceUsuario	: Binário
- analisarFace()	: faceUsuario

POR FIM, NOSSAS CLASSES FICARAM ASSIM:

CalidadorDeAcesso:

Atributos

- faceUsuario: Binário
- facesAutorizadas: Array[Binário]

Métodos

- validarAcesso(faceUsuario, facesAutorizadas): Booleano

CalidadorDeAcesso	
- faceUsuario	: Binário
- facesAutorizadas	: Array[Binário]
- validarAcesso(faceUsuario, facesAutorizadas)	: Booleano

POR FIM, NOSSAS CLASSES FICARAM ASSIM:

ControleDoApp:

Atributos

- tiposDeLogin: Array[String]
- loginEscolhido: String
- itensMenu: Array[String]

Métodos

- mostrarMenuPrincipal(itensMenu)
- mostrarTelaDeLogin(loginEscolhido)

ControleDoApp	
- tiposDeLogin	: Array[String]
- loginEscolhido	: String
- itensMenu	: Array[String]
- mostrarMenuPrincipal(itensMenu)	:
- mostrarTelaDeLogin(loginEscolhido)	:

OBSERVAÇÕES FINAIS

- Perceba que não foram alteradas as visibilidades

  Todos os métodos e atributos estão com sinal de menos (-) na frente = privado

- É necessário alterar isto para que o sistema fique coerente.

  Colocar o sinal de + na frente dos métodos e atributos que serão utilizados entre classes = público.

- Também não foram colocados os relacionamentos nem multiplicidades destes relacionamentos, o que é muito importante quando você for terminar seu diagrama.