Lição 3

Engenharia de Requisitos – Parte 2

Especificação de Requisitos

- Serve como base para o projeto e a construção do software

- Documentos Necessários:
  - Modelo de Requisitos
  - Documentos de Arquitetura de Software

Modelo de Análise

- Técnica onde o modelo de análise é derivado do modelo de requisitos
- Define as classes iniciais, chamadas de classes de análise do sistema
- Identifica as responsabilidades das classes de análise através da colaboração entre as classes

Produto Resultante

- Classes de Análise
  - Criado usando o Diagrama de Classe
  - É a entrada principal da atividade de engenharia de projeto
  - Representa o aspecto estático do sistema através das classes de análise

- Modelo Comportamental
  - Criado usando os Diagramas de Seqüência e Colaboração
  - Representa o aspecto dinâmico do sistema
  - Mostra interações e colaborações

Propósito da Análise de Caso de Uso

- Identificar as classes de análise que irão formar o fluxo de eventos do caso de uso
- Definir o comportamento das classes de análise
- Identificar responsabilidades, atributos e associações das classes de análise

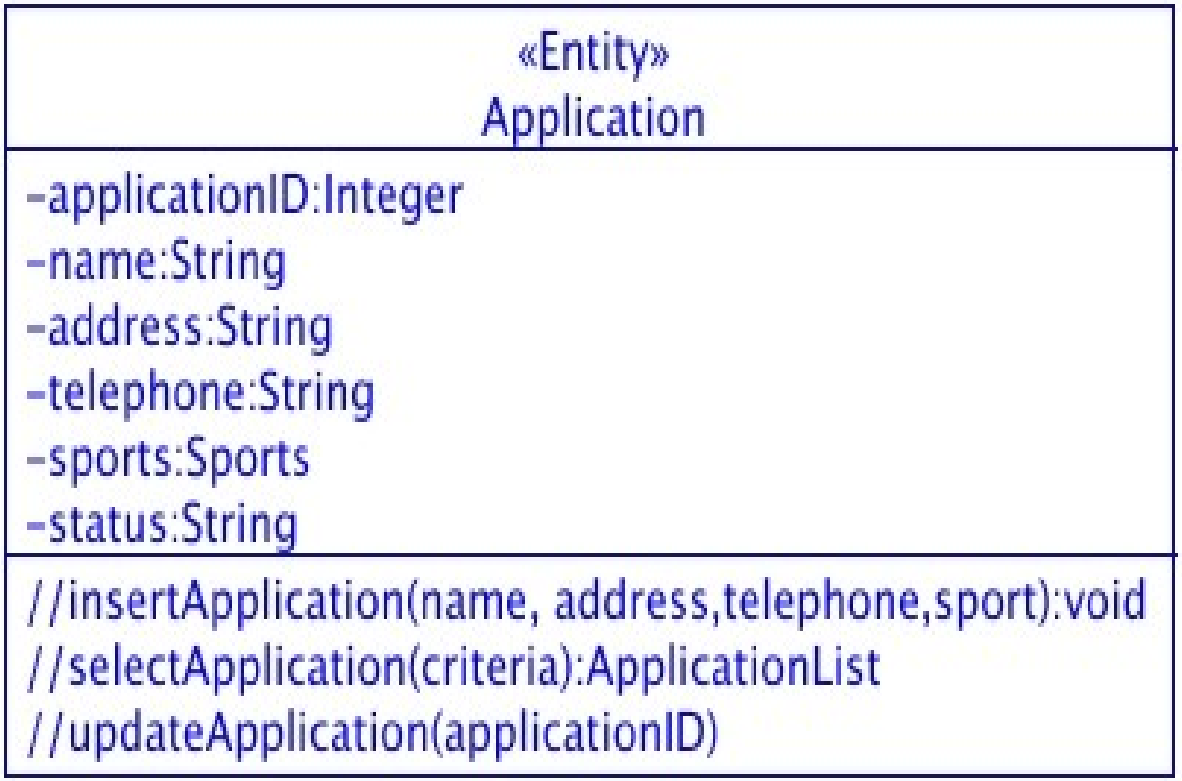
Passo 1: Valide o Modelo de Caso de Uso

- Se o Modelo de Caso de Uso não foi validado ainda, cheque o modelo para garantir conformidade com os requisitos
- Alguém pode achar que alguns requisitos estão incorretos ou não tão bem entendidos. Nesse caso, o fluxo de eventos original deve ser atualizado, isto é, é necessário voltar e fazer a análise de requisitos
- Como exemplo, qual deles está claro?
  - A seleção comitê busca formulários de aplicação
  - A seleção comitê retorna os formulários de aplicação da aplicação que está agendada para o simulado a partir do arquivo com o título “Aplicação do simulado”

Passo 2: Para cada caso de uso localizar classes de análise

- Um conjunto de classes de análise candidatas (elementos modelo) é identificado
- Classes de análise são capazes de executar o comportamento descrito pelos casos de uso
- Classes de análise devem ser nomeadas e descritas brevemente em poucas sentenças

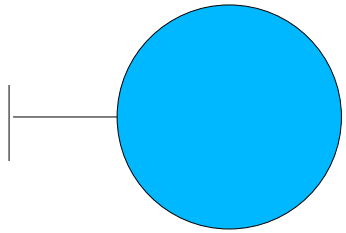
Classe

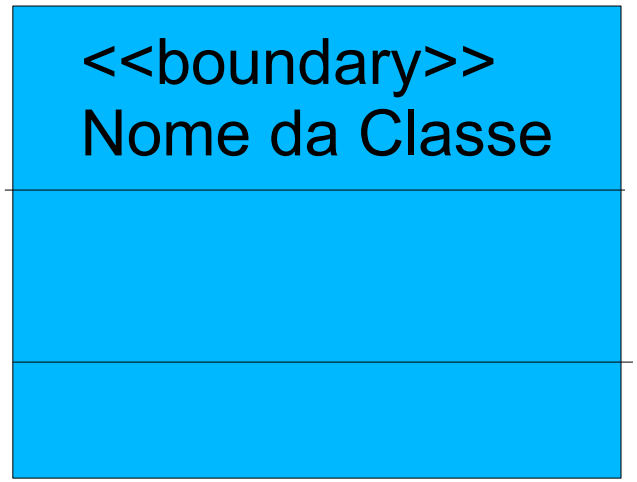

«Entity»
Application
-applicationID:Integer
-name:String
-address:String
-telephone:String
-sports:Sports
-status:String
//insertApplication(name, address,telephone,sport):void
//selectApplication(criteria):ApplicationList
//updateApplication(applicationID)

Três Perspectivas

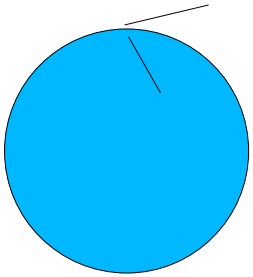
- Fronteira entre o sistema e seus atores
- Informação que o sistema usa
- Controle lógico do sistema

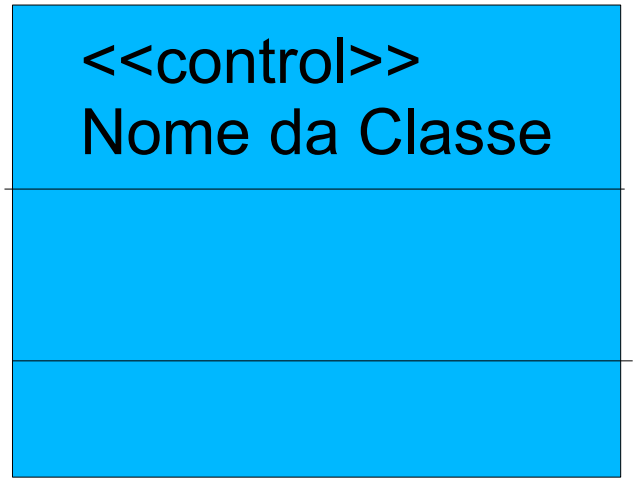
Classes de Fronteira



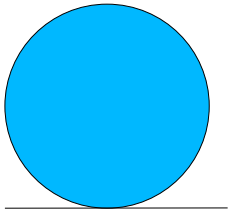


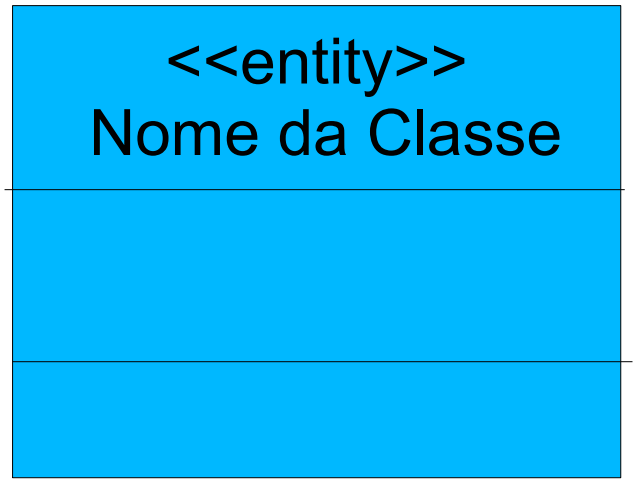
Classes de Controle





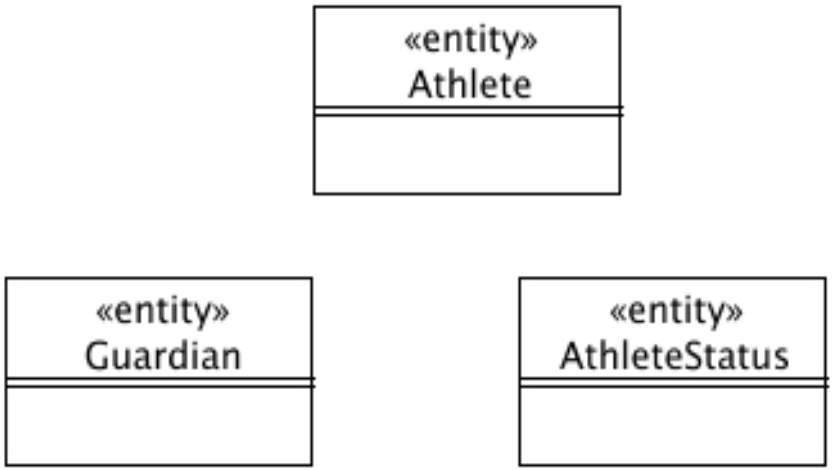
Classes de Entidade





Classes de Entidade

- Encontradas por:
  - Usando fluxo de eventos de um caso de uso como entrada
  - Obtendo abstrações chave de casos de uso
  - Filtrando substantivos



Passo 3: Modelar o comportamento das classes de análise

- Comportamento é observado através da colaboração das classes
- Usa os Diagramas de Interação ou Evento:
  - Diagramas de Seqüência
  - Diagramas de Colaboração
- Ajuda a definir as responsabilidades das classes de análise
- Nível de objeto é usado para permitir que cenários e eventos utilizem mais de uma instância de uma classe

Objetos

- São instâncias de uma classe
- Podem ser objetos nomeados ou objetos anônimos

Objetos Nomeados

Objetos Anônimos

Mensagens

- Forma de comunicação entre objetos
- Têm o seguinte formato:

*** [condição] : ação(lista de parâmetros) : TipoDeRetorno**

- Exemplos:
  - enviarAplicacao()
  - [atletaEstaAqui = sim] jogaBasquetebol(atleta):void
  - * [para cada atleta] atletaEstaAqui(atleta):Boolean

Diagramas de Evento/Interação

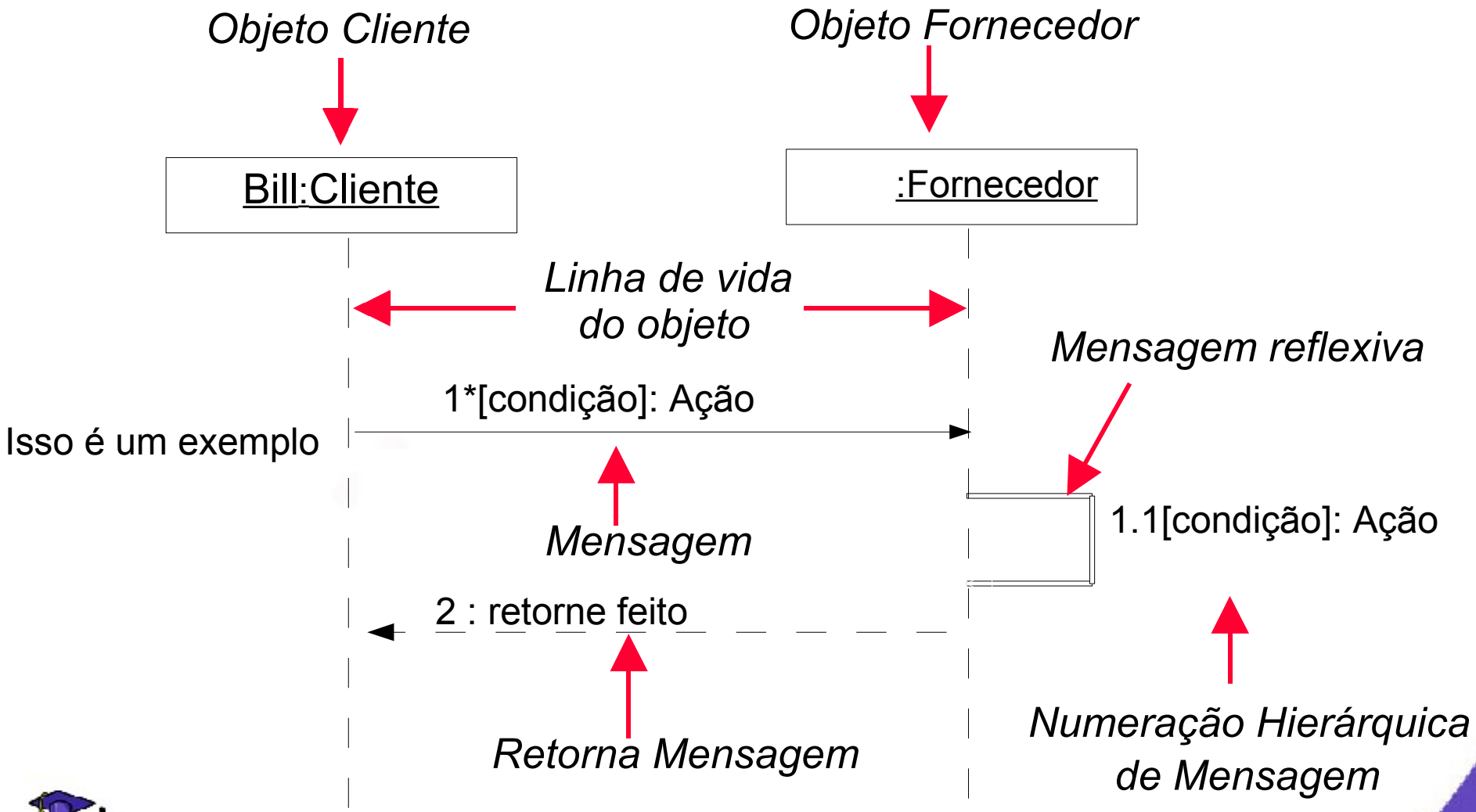
- Eles modelam a natureza dinâmica dos objetos dentro de um conjunto de classes

- Eles modelam o comportamento do sistema.
  - Como o sistema responderá a certas ações do usuário?
  - Como os objetos são criados? Modificados?
  - Como os dados são modificados?

Diagrama de Seqüência

- O propósito é modelar interações entre objetos, os quais mapeiam interações seqüenciais para interações de objeto
- Mostra a seqüência explícita de mensagens sendo passadas de um objeto para outro
- É melhor usado para especificações de tempo real e para cenários complexos
- Mostra as interações seqüenciais entre objetos

Notação Básica de um Diagrama de Seqüência



Notação Aprimorada do Diagrama de Seqüência

- A adição explícita de *return* não é seguida por todos os projetistas
- Criação e término de objeto
- Mensagens Customizadas

Criação de Objeto

- Posicionar o objeto no topo do diagrama implica que o objeto existe antes do cenário iniciar

- Se o objeto é criado durante a execução do cenário, posicione-o no ponto onde foi criado

Ativação e Término do Objeto

- Para mostrar que um objeto está ativo: incremente a linha de vida do objeto para um retângulo
- O topo do retângulo indica quando um objeto se torna ativo; o parte de baixo do objeto indica quando um objeto se tornou inativo
- Para mostrar que um objeto terminou, posicione um X no ponto onde a terminação ocorreu

Mensagens Customizadas

- Mensagens customizadas (para eventos comuns de programação)

- Mensagens Síncronas
- Mensagens Assíncronas

- Eventos de *Timeout*

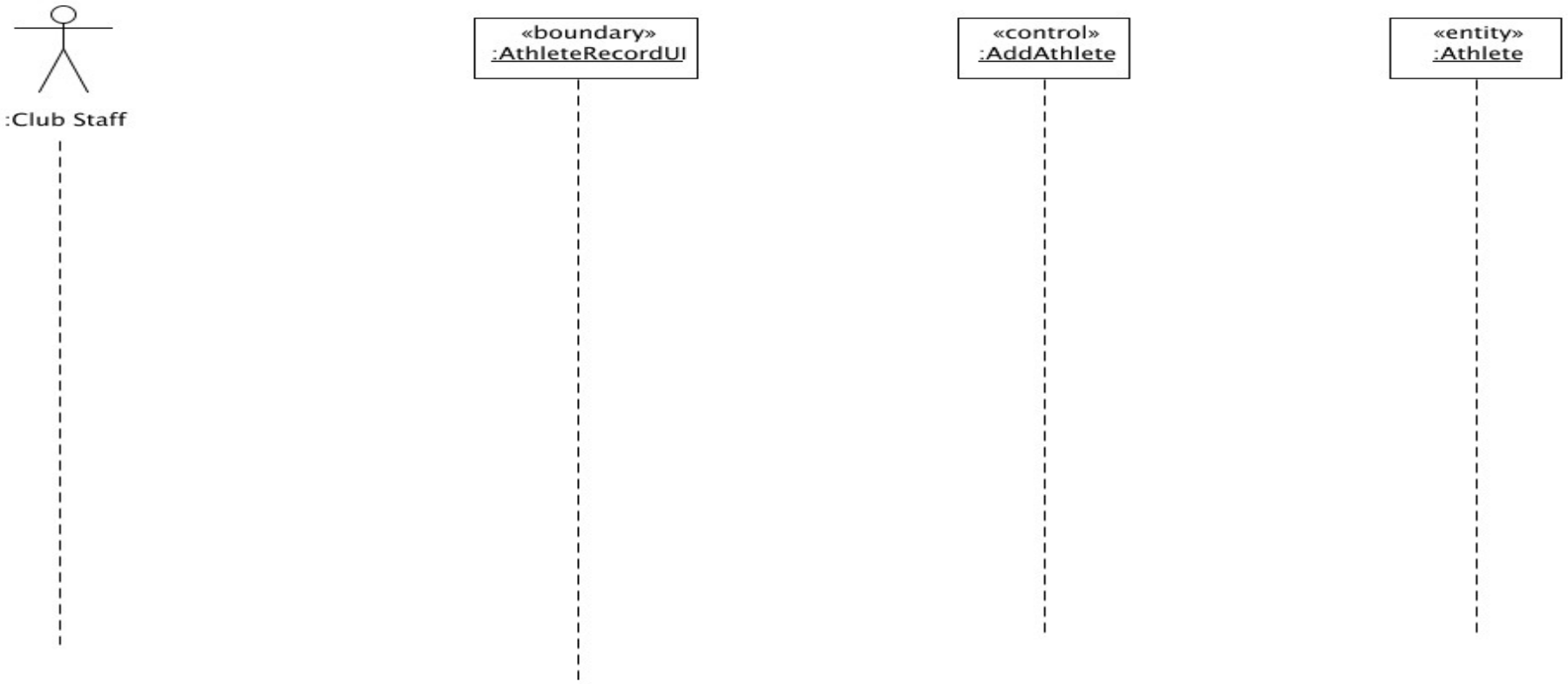
Construindo um Diagrama de Seqüência

- **Passo 1**: Alinhar os atores e objetos participantes
- **Passo 2**: Criar as linhas de vida dos atores e objetos
- **Passo 3**: Criar as mensagens de um objeto para outro
- **Passo 4**: Criar as mensagens de retorno
- **Passo 5**: Enumerar as mensagens por ordem cronológica
- **Passo 6**: Elaborar as mensagens
- **Passo 7**: Aumentar opcionalmente os Diagramas de Seqüência

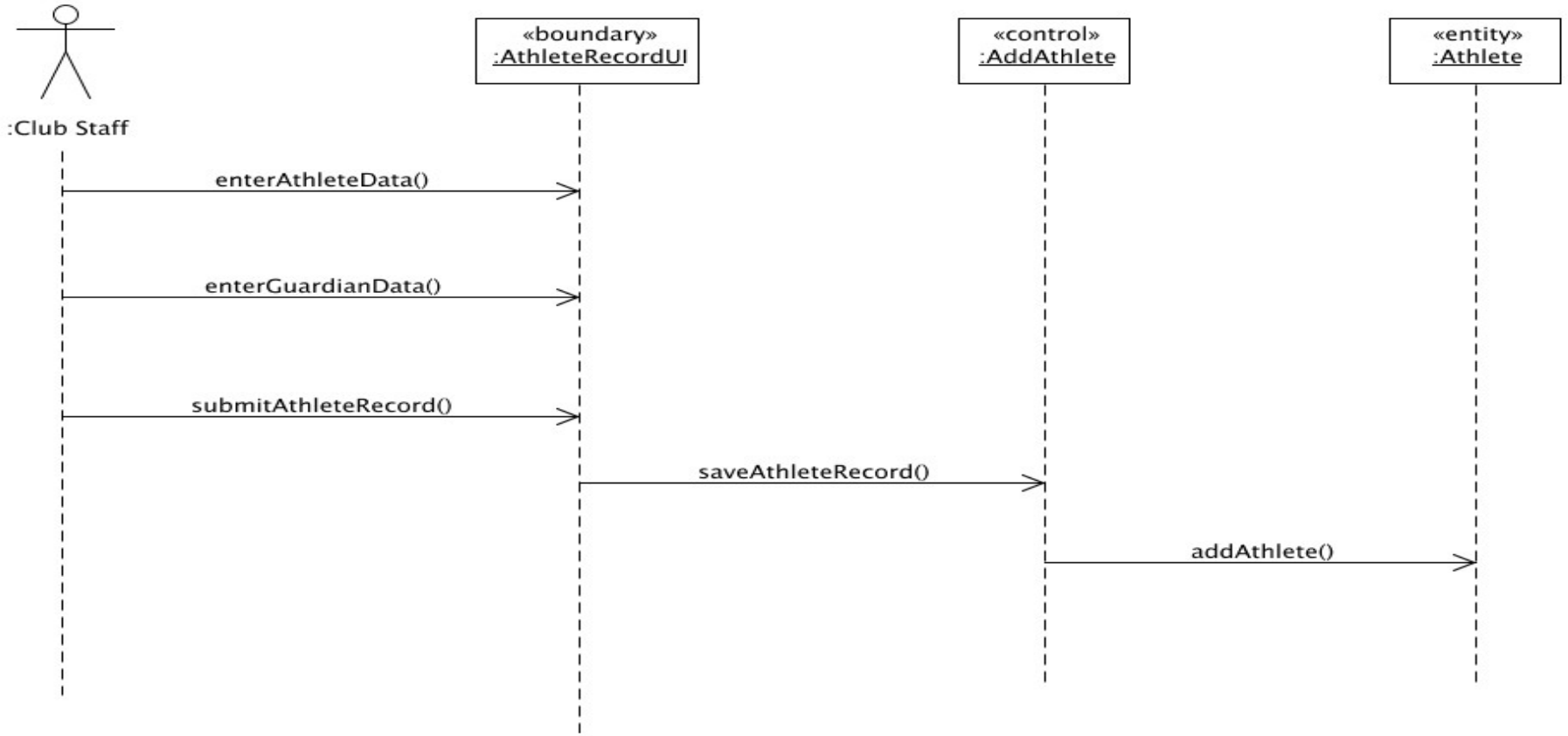
Exemplo de Cenário:
Funcionário do Clube adiciona um registro do atleta

Primeiro passo

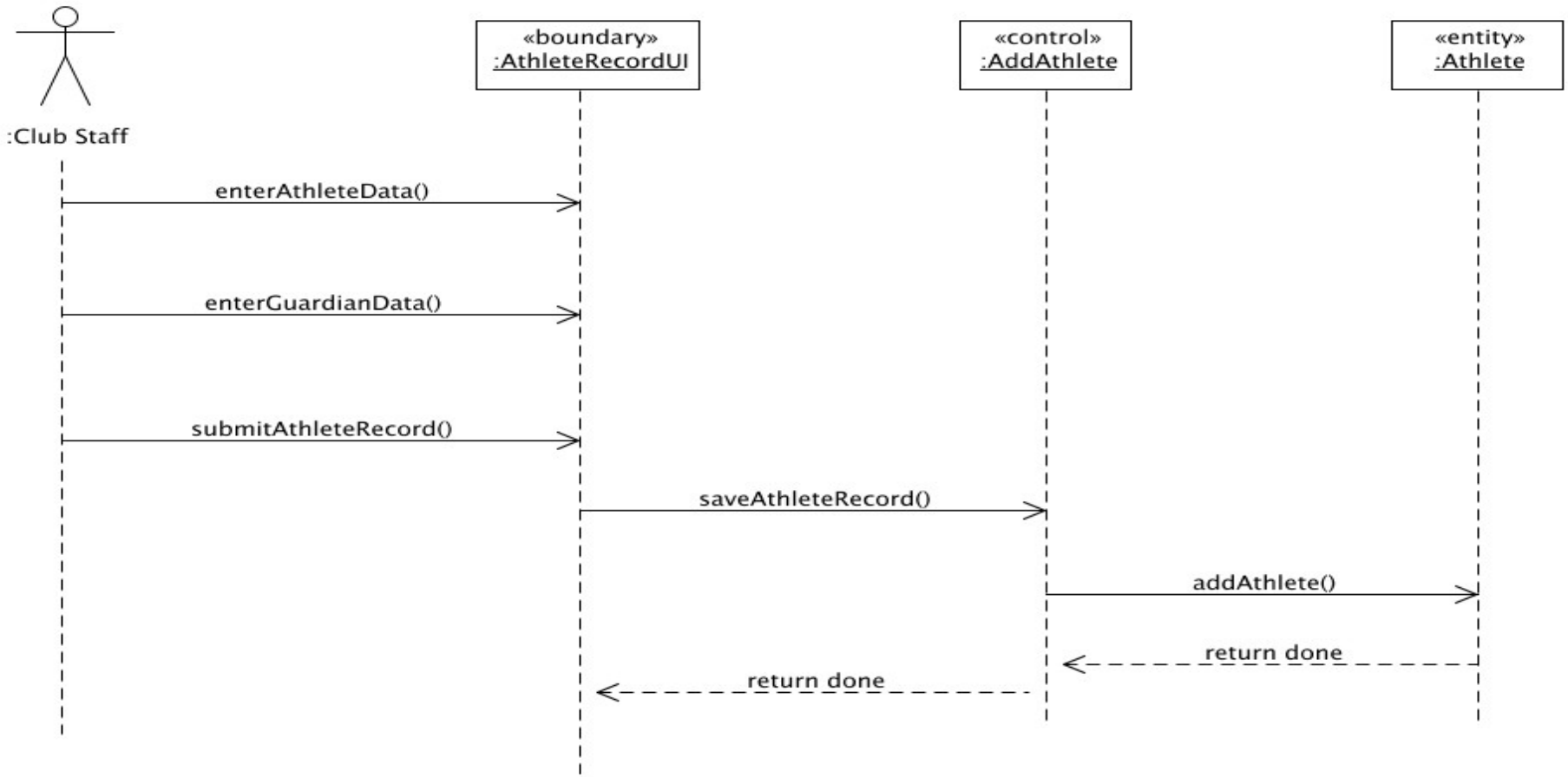
Segundo Passo



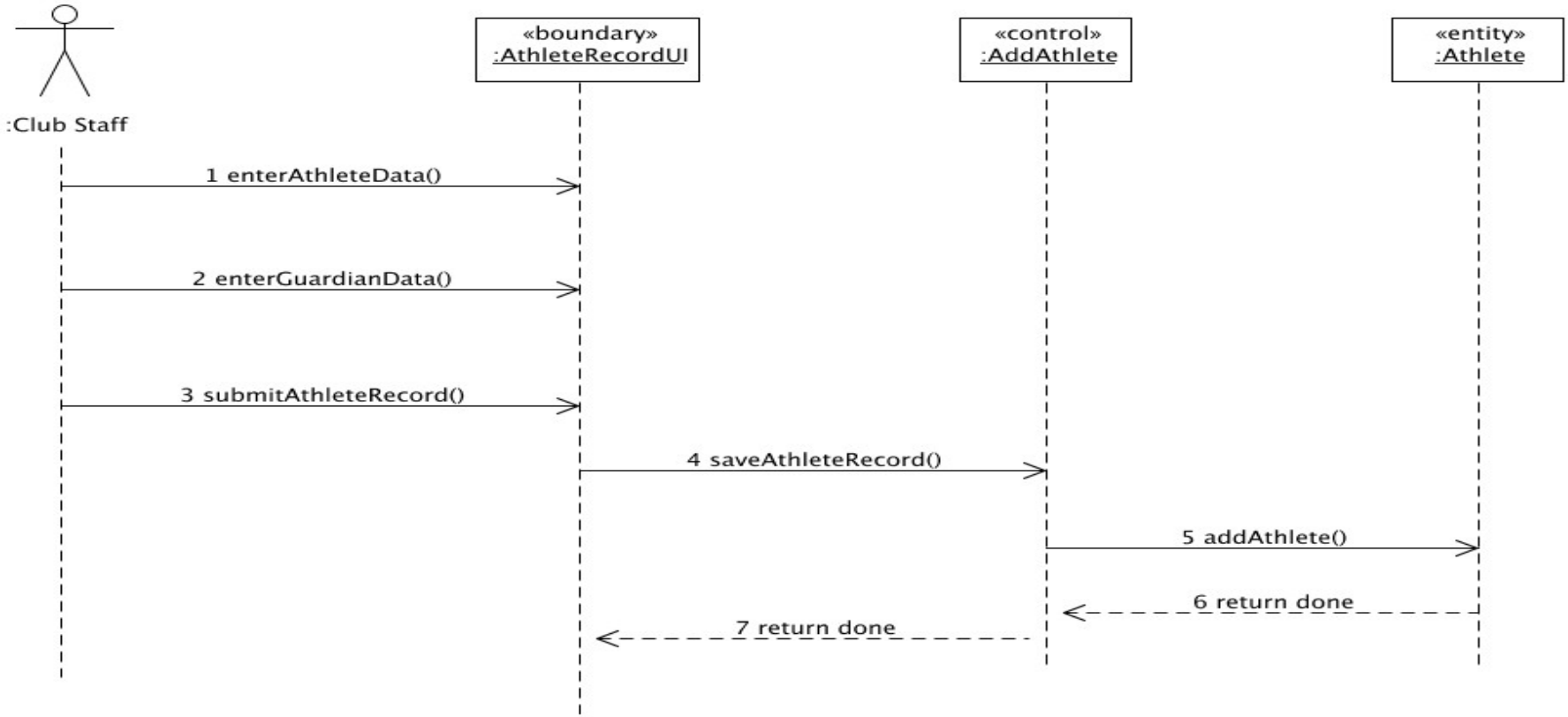
Terceiro Passo



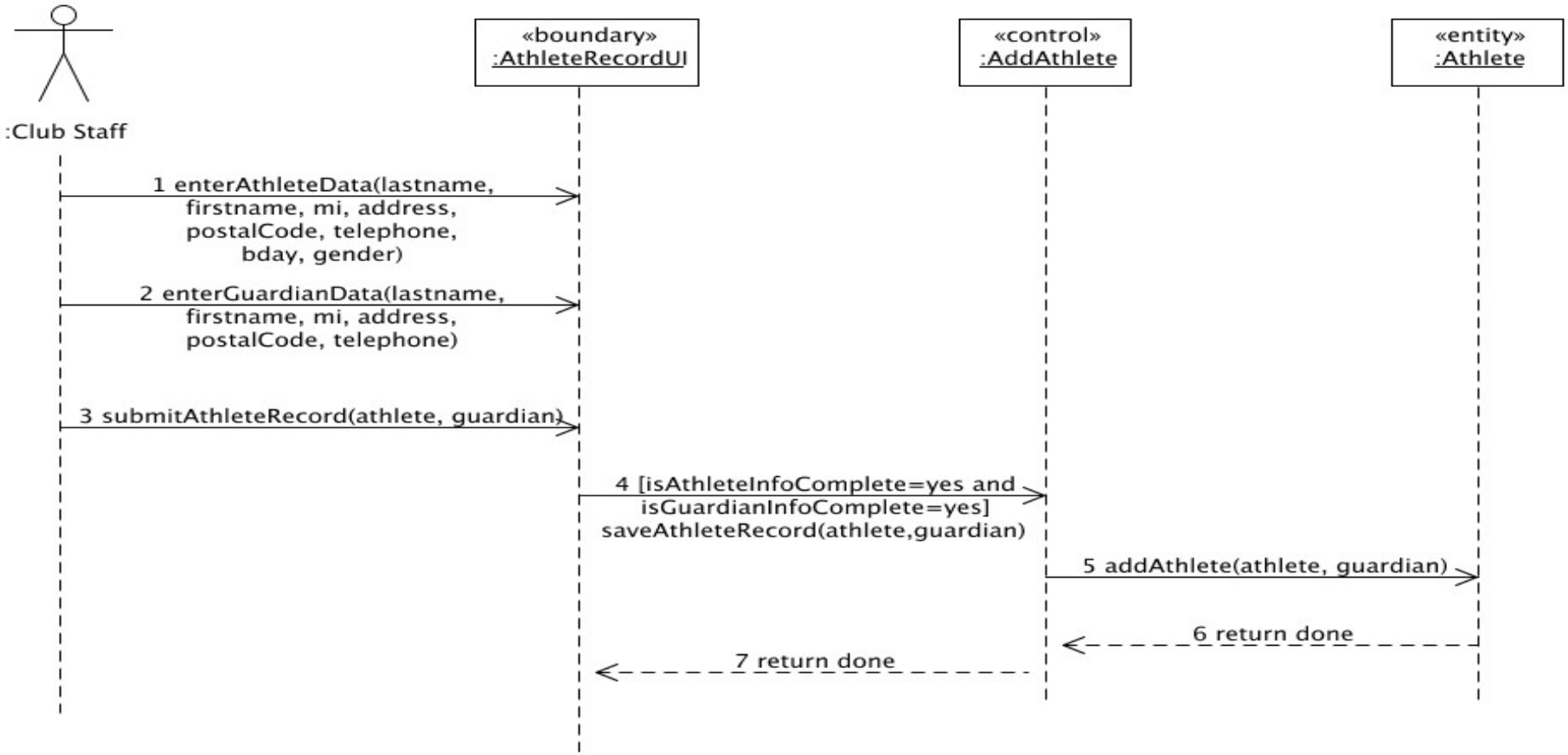
Quarto Passo



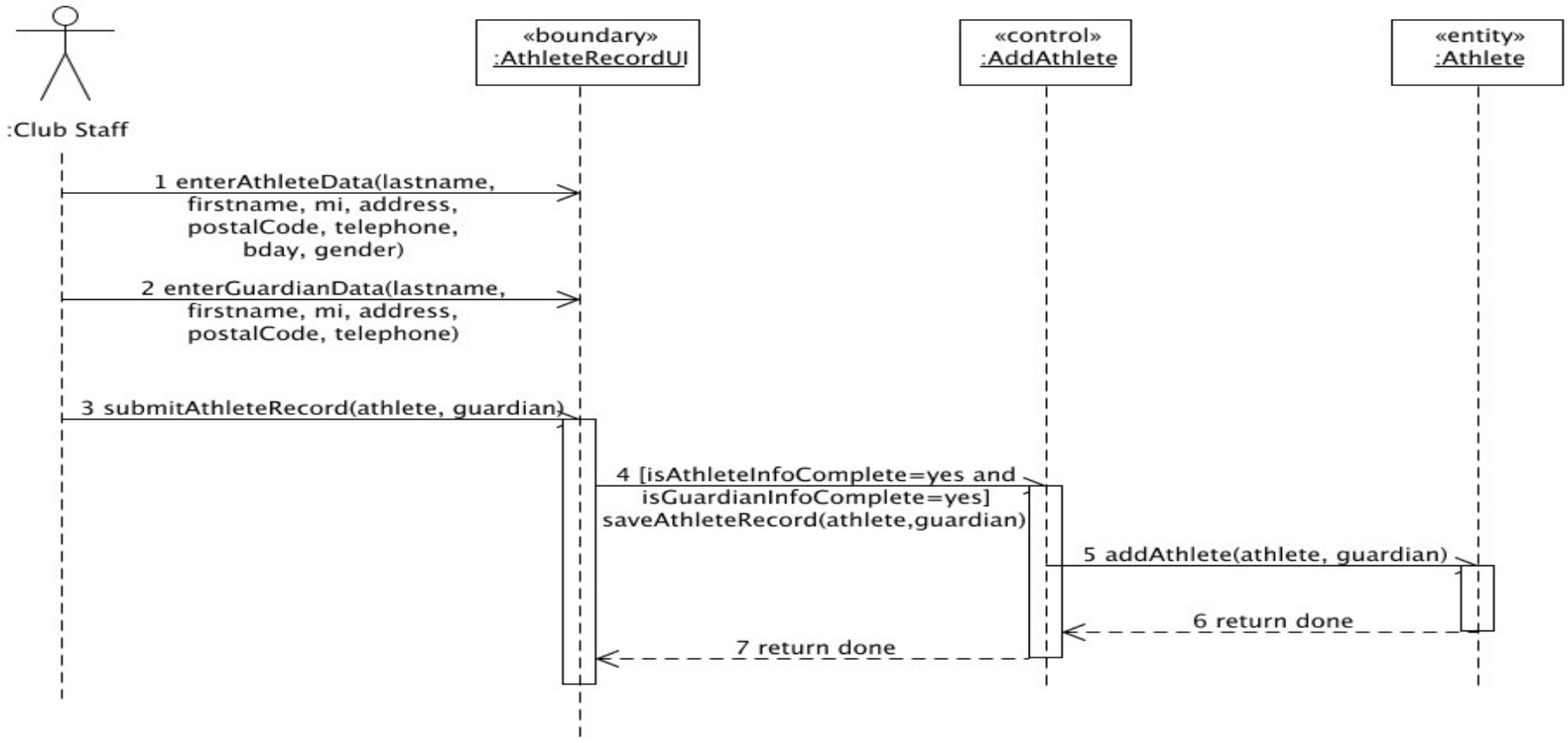
Quinto Passo



Sexto Passo


:Club Staff
«boundary»
:AthleteRecordUI
«control»
:AddAthlete
«entity»
:Athlete
1 enterAthleteData(lastname, firstname, mi, address, postalCode, telephone, bday, gender)
2 enterGuardianData(lastname, firstname, mi, address, postalCode, telephone)
3 submitAthleteRecord(athlete, guardian)
4 [isAthleteInfoComplete=yes and isGuardianInfoComplete=yes] saveAthleteRecord(athlete,guardian)
5 addAthlete(athlete, guardian)
6 return done
7 return done

Sétimo Passo



Final da Parte 2

- Continua...

Parceiros

- **Os seguintes parceiros tornaram JEDI™ possível em Língua Portuguesa:**