Mapeamento IDL

- Mapeamento:
  - Define como as estruturas declaradas em IDL são vistas na linguagem alvo.
  - Mapeamentos são padrões estabelecidos pela OMG.
- Porque o mapeamento é importante:
  - Construir parâmetros para a invocação de métodos
  - Interpretar valores de retorno e exceções
- Mapeamentos atuais:
  - C++, C, Java, Smalltalk, Ada, Cobol, Ruby, Python...

Implementação do Servidor

Objetos Distribuídos - CORBA

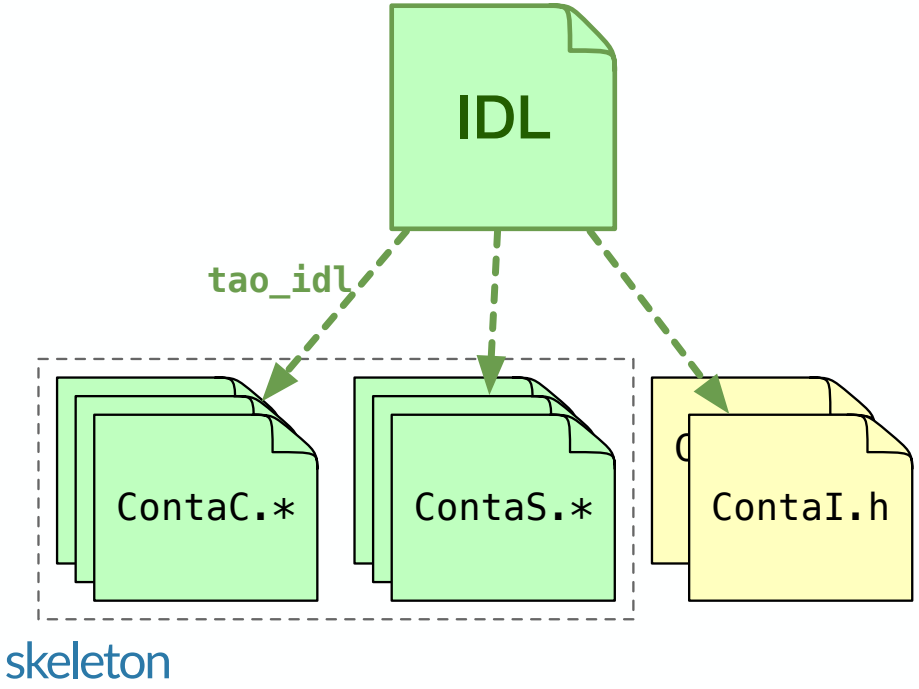
Visão Geral

- Servidor
  - Definir a **interface do objeto servidor** em IDL.
  - **Compilar** o arquivo **IDL** para gerar o código do *skeleton* na linguagem-alvo (Java, C++, C, …).
  - Escrever o **código dos métodos** na linguagem-alvo.
  - Criar código de **inicialização e instanciação** de servants.

Implementação
(Conta C++ ACE/TAO)

- Criar arquivo IDL
- Compilar arquivo IDL para gerar:
  - Stub (`ContaC.*`)
  - Skeleton (`ContaS.*` + `ContaC.*`)
  - Classes de implementação (`ContaI.*`)
- TAO:
  - `tao_idl -Gstl -GI Conta.idl`

Implementação
(Conta C++ ACE/TAO)



Implementação
(Conta C++ ACE/TAO)

- Customizar arquivos de implementação:
  - `ContaI.h`
  - `ContaI.cpp`

Implementação: Contal.h
(Conta C++ ACE/TAO)

```
class  Conta_i  : public virtual POA_Conta {
public:
    Conta_i ();
    ~Conta_i () override = default;

    ::CORBA::Float saldo () override;
    void deposito (::CORBA::Float valor) override;
    void saque (::CORBA::Float valor) override;
    void shutdown (const std::string senha) override;
private:
    std::string     id_;
    ::CORBA::Float  saldo_;
};
```

Implementação: Contal.cpp
(Conta C++ ACE/TAO)

```
Conta_i::Conta_i()
{
    saldo_ = 0.0;
}
CORBA::Float Conta_i::saldo()
{
    return saldo_;
}

void Conta_i::deposito(
            CORBA::Float valor)
{
    saldo_ += valor;
}
```

```
void Conta_i::saque(
            CORBA::Float valor)
{
    if (saldo_ >= valor)
        saldo_ -= valor;
    else
        throw SaldoInsuficiente();
}
```

Implementação
(Conta C++ ACE/TAO)

- Criar arquivo do servidor (`servidor.cpp`):
- Tipicamente:
  1. Inicializar **ORB**
  2. Ativar **gerente do POA** (`root POA`) = ativar POA
  3. Instanciar ***servants***
  4. **Registrar *servants*** no POA (tornam-se ”objetos CORBA”)
  5. **Publicar** referências dos objetos CORBA (arquivo, NS, etc.)
  6. **Aguardar** requisições
  7. **Finalizar** (poa + orb)

Servidor: resumo da implementação

- Estrutura de **diretórios**
- Geração do *stub* + *skeleton*:
  - `tao_idl -Gstl –GI Conta.idl`
- Copiar `ContaI.*` para `servidor/`
- Implementar **métodos remotos**
- Implementar **servidor**