o formato XML, que é uma recomendação da W3C para representação e transferência de dados.

2. DESENVOLVIMENTO

O *web service* do Sistema Integrador funcionará como uma ponte entre o atual sistema de controle e gerenciamento do Ciclo do Sangue e os sistemas dos demais hospitais, ele será o responsável por prover as informações referentes aos hemocomponentes e por receber as informações referente aos procedimentos realizados com estes hemocomponentes e de seus respectivos pacientes.

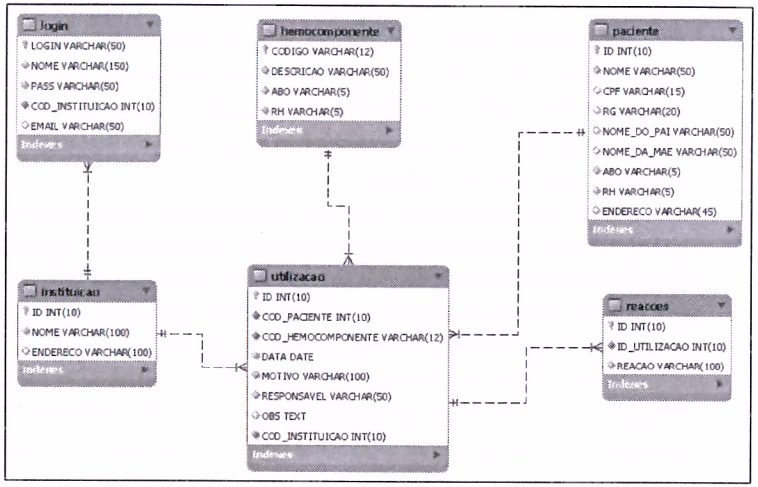
2.1. Banco de dados

Existem diversos tipos de SGBD disponíveis no mercado ~~e, durante o curso tivemos contato com alguns deles~~, mas a escolha do SGBD para este trabalho foi o MySQL, por ser um SGBD gratuito, devido já ser utilizado na empresa, pela baixa complexidade para o desenvolvimento e facilidade de manutenção e gerenciamento do mesmo.

Foi utilizada a ferramenta MySQL *Workbench* para realizar a modelagem e inserção dos dados pré-definidos. O MySQL *Workbench* é uma ferramenta de gerenciamento de SGBD que possui diversas funcionalidades, como *backup* e *restore* de banco de dados, execução de *scripts* e geração de modelos.

O banco de dados do sistema Integrador foi modelado para simular a integração com o sistema de gerenciamento do Ciclo do Sangue, e é constituído das seguintes tabelas: Login, Instituicao, Hemocomponente, Paciente, Utilizacao e Reacoes. A Figura 01 mostra o diagrama de entidade-relacionamento do banco de dados do sistema Integrador, gerado com o MySQL *Workbench*.

Figura 1: Modelo do banco de dados



Login: Armazena as informações referentes ao usuário que estará utilizando o sistema no lado servidor.

Instituicao: Armazena os dados referentes à instituição (hospital), que recebeu o hemocomponente para a realização do procedimento.

Hemocomponente: Armazena os dados dos hemocomponentes que foram distribuídos para os hospitais.

Pacientes: Armazena os dados do paciente que recebeu a transfusão ou tratamento. Estes dados serão recebidos diretamente dos hospitais via arquivos XML.

Utilizacao: Armazena os dados referentes ao procedimento realizado no paciente e suas devidas particularidades. Estes dados serão recebidos diretamente dos hospitais via XML.

Reacoes: Armazena os dados referentes a possíveis reações transfusionais que possam ter ocorrido em algum dos procedimentos. Estes dados serão recebidos diretamente dos hospitais via XML.

2.2. Web Service

Web Service consiste de uma camada de software baseada em uma arquitetura orientada a serviços em que as interfaces são baseadas em protocolos da Internet (HTTP, SMTP, FTP). Além disso, as mensagens enviadas podem ser em XML ou JSON. Essa última apresenta crescente popularidade, embora a primeira ainda seja a mais utilizada e difundida.

O XML, Extensible Markup Language, de acordo com o W3C, é um formato de texto simples, muito flexível derivado do SGML [3] (ISO 8879). Originalmente concebido para enfrentar os desafios da publicação eletrônica em larga escala, o XML também está desempenhando um papel cada vez mais importante na troca de uma ampla variedade de dados na Web e em outros lugares.

Devido a essas características, o XML tornou-se amplamente utilizado na integração de sistemas, já que um documento XML é escalável e facilmente lido por aplicações sendo executadas em plataformas diferentes. Ele também se tornou uma ferramenta essencial em sistemas que utilizam Arquitetura Orientada a Serviços (SOA), sobretudo na troca de mensagens entre Web Services e seus clientes.

O *web service* desenvolvido para este trabalho, denominado IntegraWS, possui as funções básicas e necessárias para a perfeita integração com os sistemas dos hospitais, ele possui seis métodos disponíveis para utilização: getData, getPaciente, insertPaciente, insertUtilizacao, listarReacoes e insertReacoes. Estes métodos foram expostos no *web service* utilizando a anotação *@WebMethod*, respectivamente com os seguintes nomes: *RecebeDadosHemocomponente*, *RecebeDadosPaciente*, *GravaPaciente*, *GravaUtilizacao*, ListaReacoes e GravaReacoes, conforme demonstrado na Figura 02.

Figura 2: *JavaDoc* da Classe IntegraWS

Method Summary

Methods

Modifier and Type	Method and Description
Hemocomponente	getData(java.lang.String numeroBolsa) Método que envia os dados do hemocomponente para o hospital client.
Paciente	getPaciente(int id) Método que envia os dados do paciente para o hospital client.
boolean	insertPaciente(java.lang.String nome, java.lang.String cpf, java.lang.String rg, java.lang.String pai, java.lang.String mae, java.lang.String aboPac, java.lang.String rhPac, java.lang.String endereco) Método que recebe os dados dos pacientes para gravar
boolean	insertReacoes(Reacoes reacao) Método responsável pela gravação das reações adversas nas transfusões
boolean	insertUtilizacao(int codPaciente, java.lang.String codHemocomponente, java.lang.String data, java.lang.String motivo, java.lang.String responsavel, java.lang.String obs, int codInstituicao) Método que grava os dados referente a utilização do hemocomponente
java.util.List<Reacoes>	listReacoes(java.lang.String nomePaciente) Método que envia os dados de reações transfusionais para os hospitais client.

RecebeDadosHemocomponente: recebe o código da bolsa do hemocomponente, busca as informações na base de dados e retorna um objeto do tipo Hemocomponente, que contém as seguintes informações: Código, Descrição, ABO e RH. Estas informações serão recebidas pelos sistemas dos hospitais.

Figura 3: Método RecebeDadosHemocomponente

```
@WebMethod(operationName = "RecebeDadosHemocomponente")
public Hemocomponente getData(@WebParam(name = "numeroBolsa") String numeroBolsa) {
    HemocomponenteDAO hemoDao = new HemocomponenteDAO();
    Hemocomponente bolsa = new Hemocomponente();

    bolsa = hemoDao.getHemocomponente(numeroBolsa);

    return bolsa;
}
```

RecebeDadosPaciente: recebe o código do paciente que irá receber o hemocomponente, busca as informações na base de dados e retorna um objeto do tipo Paciente, que contém as seguintes informações: Código, Nome, CPF, RG, Nome do Pai, Nome da Mãe, ABO, RH e Endereço do paciente.

Figura 4: Método RecebeDadosPaciente

```
@WebMethod(operationName = "RecebeDadosPaciente")
public Paciente getPaciente(@WebParam(name = "id") int id) {
    PacienteDAO pacDao = new PacienteDAO();
    Paciente paciente = new Paciente();

    paciente = pacDao.getPaciente(id);

    return paciente;
}
```

GravaPaciente: método responsável por efetuar o cadastro dos pacientes no banco de dados. Recebe os dados: Nome, CPF, RG, Nome do Pai, Nome da Mãe, ABO, RH e Endereço do paciente e efetua o cadastro no banco de dados. Se o cadastro for efetuado corretamente, o método retorna um valor *booleano* verdadeiro, no caso de acontecer algum erro, retorna falso. (Ver Fig 5)

Figura 5: Método GravaPaciente

```
@WebMethod(operationName = "GravaPaciente")
public boolean insertPaciente(
        @WebParam(name = "nome") String nome,
        @WebParam(name = "cpf") String cpf,
        @WebParam(name = "rg") String rg,
        @WebParam(name = "pai") String pai,
        @WebParam(name = "mae") String mae,
        @WebParam(name = "aboPac") String aboPac,
        @WebParam(name = "rhPac") String rhPac,
        @WebParam(name = "endereco") String endereco) {

    PacienteDAO pacienteDao = new PacienteDAO();
    Paciente paciente = new Paciente();

    paciente.setNome(nome);
    paciente.setCpf(cpf);
    paciente.setRg(rg);
    paciente.setNomeDoPai(pai);
    paciente.setNomeDaMae(mae);
    paciente.setAbo(aboPac);
    paciente.setRh(rhPac);
    paciente.setEndereco(endereco);

    boolean retorno = pacienteDao.insert(paciente);

    return retorno;
}
```

GravaUtilizacao: principal método do *web service* IntegraWS, pois após realizar o cadastro do paciente ele vincula o hemocomponente utilizado com paciente que o recebe. Inclui também todos os dados referente à utilização do hemocomponente. O método GravaUtilização recebe os seguintes parâmetros: Código do Paciente, Código do Hemocomponente, Data do procedimento, Motivo do procedimento, Responsável pelo procedimento, Código da instituição e um campo para Observações. Se o cadastro for efetuado corretamente, o método retorna um valor *booleano* verdadeiro, no caso de acontecer algum erro, retorna falso. (Ver Fig 6)