**Programação de Computadores III**

**Armando de Freitas**

**Documentação – Clínica Hospitalar**

**Dependências utilizadas:**

**Excel**: Apache POI

**JSON**: ORG.JSON

**XML**: JAXB

**Video:** <https://www.youtube.com/watch?v=o5O6tBDUaZI>

**Repositório:** <https://github.com/armandomtf/ClinicaHospitalar>FX

**Introdução**

Na minha opinião, o desenvolvimento desta parte do projeto, referente a N2, foi mais tranquilo que o da N1, afinal muito código e muita lógica foram reaproveitados. Logo, se deu de forma mais fluída o processo de desenvolvimento das funcionalidades.

A interface do SceneBuilder para manipulação do fxml facilita bastante o processo de desenvolvimento de telas, dessa forma achei mais fácil lidar do que o Swing. Os controles melhores, mais bonitos e mais tranquilos de lidar.

Os maiores desafios foram na implementação das novas funcionalidades requeridas: persistência em JSON e XML. Para o primeiro, após as pesquisas me pareceu consenso que a org.json era a melhor para esta extensão. A lógica foi parecida com a da Apache POI, o que facilitou bastante. Já para o XML, percebi que havia algo mais nativo da linguagem para persistência neste tipo de arquivo, logo encontrei a JAXB que era a api mais usada nesse sentido e percebi que teria que fazer um mapeamento das classes para a conversão em XML.

Este mapeamento facilitou bastante na parte de importação, pois usando o unmarshaller foi bem tranquilo de instanciar os objetos vindos do arquivo XML dado pelo usuário.

**Organização do projeto**

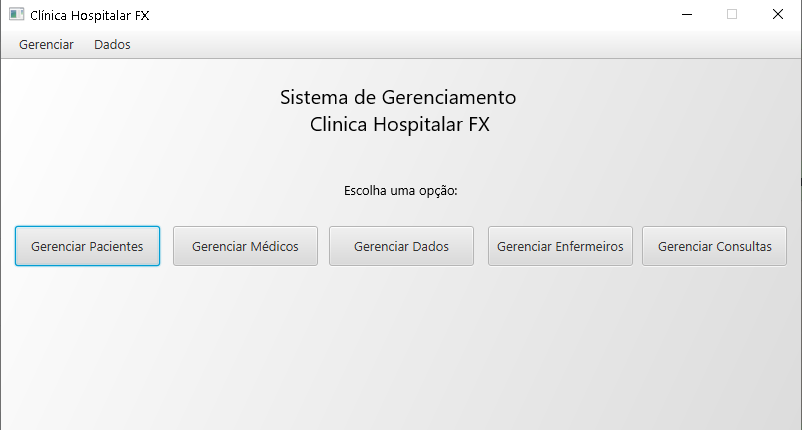
Para organizar os arquivos dentro do projeto, tentei utilizar algo parecido com um MVC, mas travei em algumas limitações do framework. Dessa forma, organizei desta maneira:

1. **java.com.armando.prj\_clinicahospitalarfx** – Reúne as classes “controllers” respectivas aos fxml’s. Também armazena o arquivo inicializador App.java
2. **java.com.armando.prj\_clinicahospitalarfx.data** – Reúne as classes de persistência Excel, JSON e XML
3. **java.com.armando.prj\_clinicahospitalarfx.model** – Reúne as entidades modelos do diagrama dado. Também possui as classes de parse XML necessárias para o funcionamento da api de XML.
4. **resources.com.armando.prj\_clinicahospitalarfx** – Reúne a parte visual do projeto, ou seja, os arquivos .fxml.
5. **<default package>** - Contém o arquivo de configuração module-info.java

**Telas**

A aplicação visual foi dividida da seguinte forma:

1. **HomePage** – Página inicial para redirecionamento de telas;



1. **Dados** – Reúne as funcionalidades Importar/Exportar na extensão desejada;

A screenshot of a computer

Description automatically generated

1. **Gerenciar Consultas** – Cadastrar/Consultar/Alterar/Remover Consultas;

A screenshot of a computer

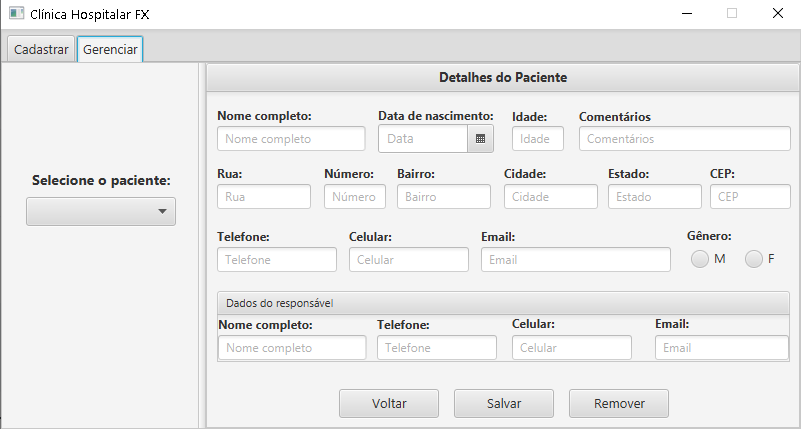
Description automatically generated**A screenshot of a computer

Description automatically generated**

1. **Gerenciar Pacientes** ­- Cadastrar/Consultar/Alterar/Remover Pacientes;

**A screenshot of a computer

Description automatically generated**

****

1. **Gerenciar Médicos** ­- Cadastrar/Consultar/Alterar/Remover Médicos;

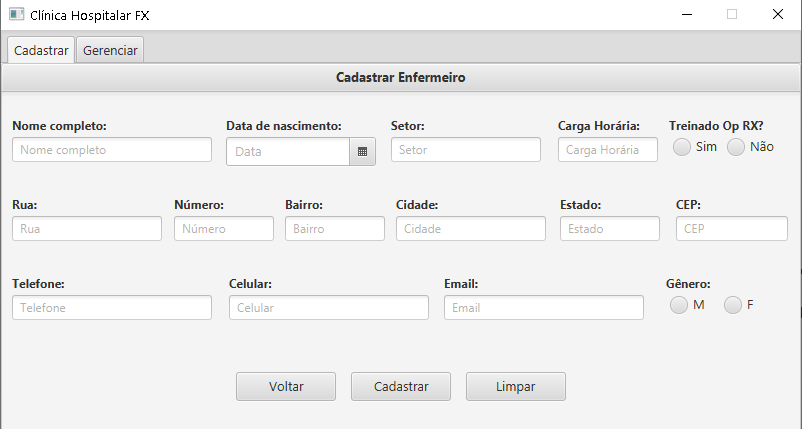
**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

1. **Gerenciar Enfermeiros** ­- Cadastrar/Consultar/Alterar/Remover Enfermeiros;



A screenshot of a computer

Description automatically generated

**Classes**

As classes foram criadas de acordo com o diagrama dado no exercício:

A diagram of a computer

Description automatically generated

Todos os construtores foram criados de acordo com as necessidades de instanciação dos objetos pertinentes.

Na HomePage foram criadas as arraylists estáticas de controle que foram usadas para termos estes objetos armazenados de alguma forma durante a execução da aplicação.

Além dessas, foram criadas as classes “ExportarExcel”, “ImportarExcel”, “ExportarJSON”, “ImportarJSON”, “ExportarXML” e “ImportarXML” que as api’s pertinentes para persistência em cada extensão requisitada.

Foram também, criadas classes auxiliares “XMLPacientes”, “XMLMedicos”, “XMLEnfermeiros”, “XMLConsultas e “XMLCombine”. Classes necessárias para o correto funcionamento do JAXB, sendo a última uma classe de “concatenação” das outras 4 para gerar um output único.

Por último, as classes “GerenciarConsultasController”, “GerenciarEnfermeirosController”, “GerenciarMedicosController”, “GerenciarPacientesController”, “HomePageController”, “ImportarExportarController”, que são as classes de controle dos fxml’s correspondentes.

**Conclusão**

Foi um projeto bem desafiador, mas foi de bastante aprendizado durante o desenvolvimento. Dediquei muitas horas para a construção e execução dele. Além deste README, também comentei todas as funções auxiliares e métodos presentes no código para que fique de fácil entendimento. Me disponho para esclarecer quaisquer dúvidas pertinentes.