INSTITUTO POLITÉCNICO
DO CÁVADO E DO AVE
ESCOLA SUPERIOR DE TECNOLOGIA

RELATÓRIO DE TRABALHO PRÁTICO

Sistema que permita gerir uma urgência de um hospital - Fase Final

RICARDO SAMPAIO E CLÁUDIO SILVA

ALUNOS Nº18827 E Nº18843

REPOSITÓRIO GIT:
HTTPS://GITHUB.COM/RICARDOSAMPAIO12121/18827_18843_LP2

Trabalho realizado sob a orientação de:

Luís Ferreira

1 Introdução

Neste capítulo falamos sobre a contextualização do projeto, a motivação e os objetivos e, para terminar, descrevemos a estrutura do documento.

1.1 Contextualização

Cada vez mais os hospitais aderem às tecnologias mais recentes de maneira a melhorar o atendimento e o tempo de espera das pessoas.

Com este projeto pretendemos criar uma aplicação em C# que permita gerir o serviço de urgências de um hospital pelos enfermeiros e médicos.

C# é uma linguagem de programação, desenvolvida pela *Microsoft,* orientada a objetos.

A programação orientada a objetos (POO) é um paradigma de programação baseado no conceito de “objetos” que podem conter atributos e métodos.

1.2 Motivação e objetivos

O objetivo deste projeto foi criar uma aplicação que fosse capaz de gerir o serviço de urgências de um hospital.

As funções da aplicação são:

→Fazer a ficha dos pacientes e editar o nome e a data se necessário;

→Adicionar pacientes à fila de espera para a triagem;

→"Criar" médicos, dando-lhes um código único;

→Editar nome de médicos e eliminá-los se necessário;

→ Chamar pacientes para a triagem, sendo-lhes dado um código de 1(pouco grave) a 5(muito grave) de grau de prioridade dado pelo "médico" e adicioná-los à fila de espera para a consulta;

→ Chamar paciente para a consulta de acordo com a prioridade;

→É também possível listar todas as filas de espera se necessário.

1.3 Estrutura do documento

Este documento está estruturado em três partes.

Na primeira parte falamos sobre a contextualização do problema, a motivação e objetivos e a estrutura do documento.

Na segunda parte falamos acerca das estruturas de dados utilizadas e apresentamos um diagrama de classes.

2 Implementação

2.1 Estrutura de dados

Todos os dados dos pacientes e dos médicos são guardados num ficheiro binário individual e, o log de cada paciente, é guardado num ficheiro de texto, escolhemos fazer este num ficheiro de texto apenas para experimentar os dois tipos de escrita.

Para guardar os pacientes que estão à espera para entrar na triagem, utilizamos uma *Queue genérica* visto ser ideal para este caso, que é inserir cada paciente no fim da *Queue* e retirar no início da *Queue*.

Os médicos em trabalho são armazenados numa *List* já que a remoção de cada um da lista não pode ser ordenada como na *Queue.*

Os pacientes, depois de passaram na triagem, são inseridos numa *Sorted List genérica,* escolhemos esta estrutura de dados porque, neste caso, precisamos de utilizar prioridades. Cada paciente, ao sair da triagem, é lhe atribuído uma prioridade dada pelo utilizador e, de seguida, são inseridos na *SortedList p*ela ordem de prioridade (de 5 mínimo até 1 máximo). Para isto utilizamos o nosso próprio *Icomparer* dado que o comparador predefinido do *SortedList* não permite a inserção de prioridades repetidas.

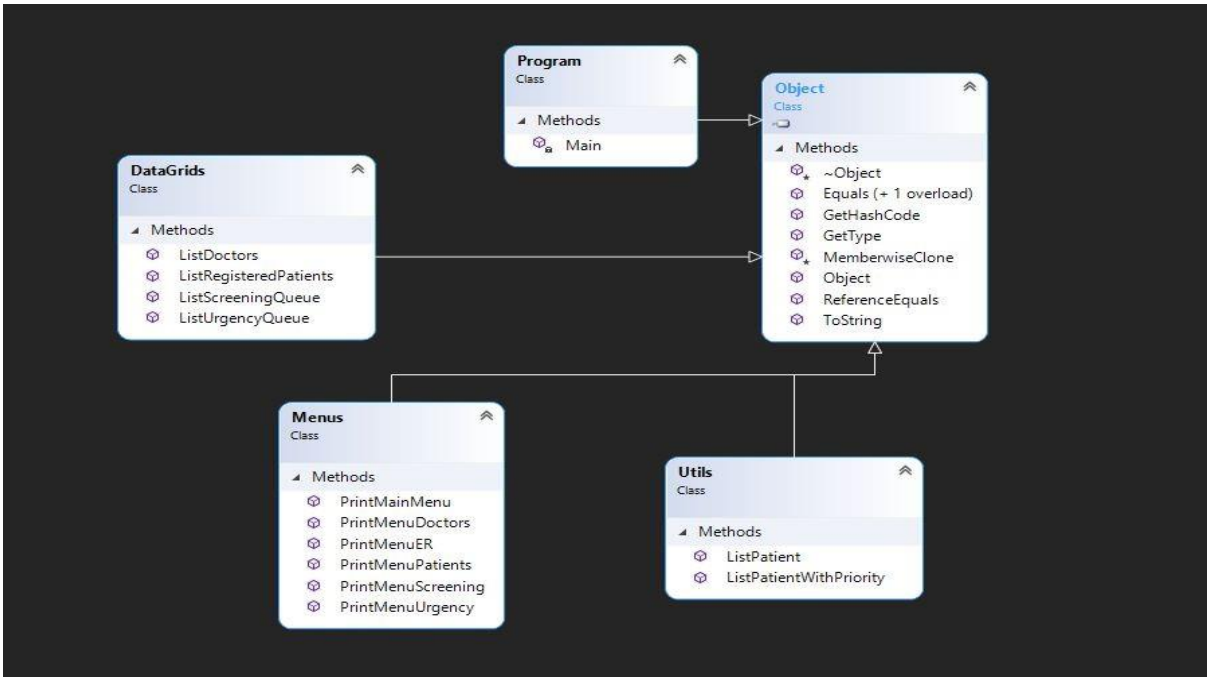
2.2 Design pattern

Como design pattern utilizamos o N-Tier por ser o que mais nos agradou.
O programa está dividido em 6 projetos(.dll) que são os seguintes:

1) HospitalER: Trata da interação com os utilizadores;
2) HospitalBO: Trata dos objetos;
3) DataLayer: Trata dos dados;
4) HospitalRN: Trata de ser o intermediário entre o HospitalER e o DataLayer;
5) Exceptions: Trata das exceções;
6) Valitations: Trata de validar os dados introduzidos pelo utilizador.

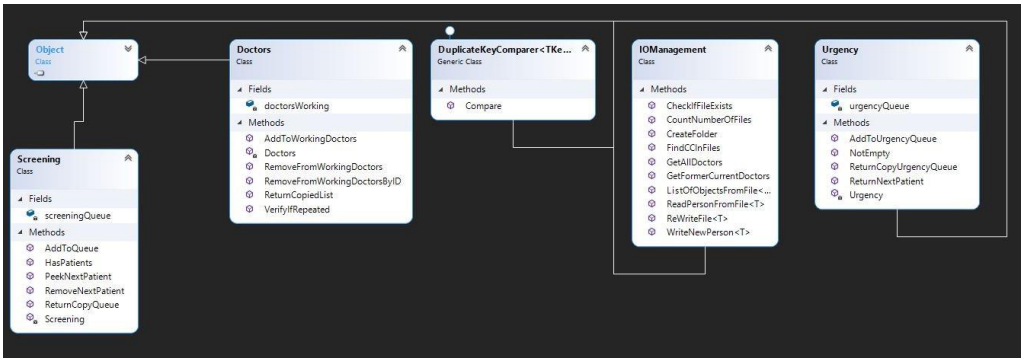
2.3 Diagrama de classes

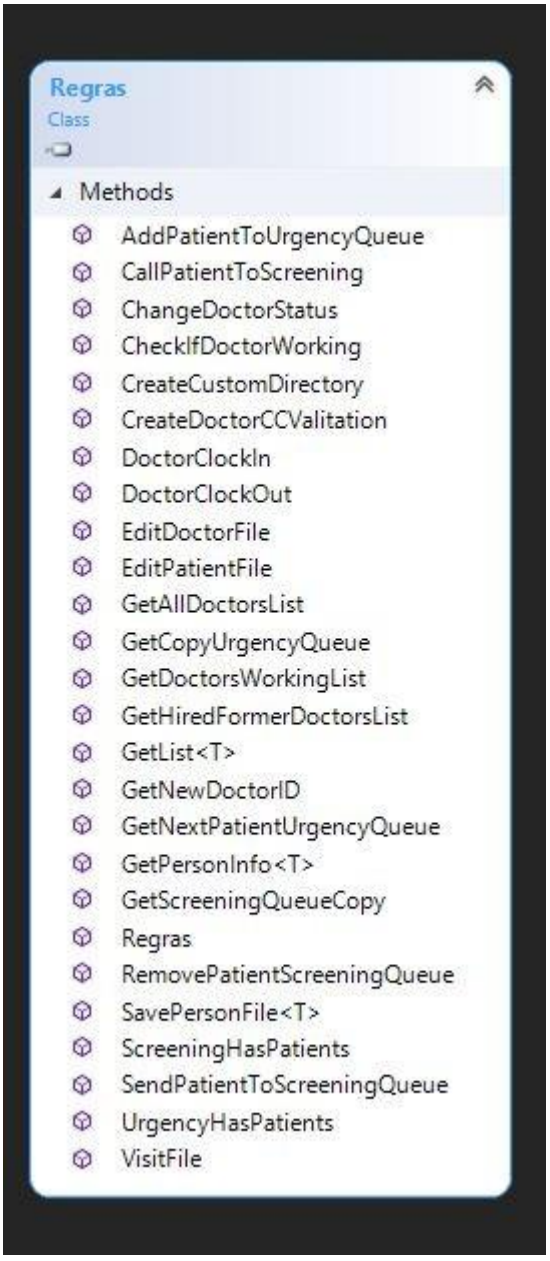
HospitalER



HospitalBO

Datalayer




Regras
Class
Methods
AddPatientToUrgencyQueue
CallPatientToScreening
ChangeDoctorStatus
ChecklfDoctorWorking
CreateCustomDirectory
CreateDoctorCCValitation
DoctorClockIn
DoctorClockOut
EditDoctorFile
EditPatientFile
GetAllDoctorsList
GetCopyUrgencyQueue
GetDoctorsWorkingList
GetHiredFormerDoctorsList
GetList<T>
GetNewDoctorID
GetNextPatientUrgencyQueue
GetPersonInfo<T>
GetScreeningQueueCopy
Regras
RemovePatientScreeningQueue
SavePersonFile<T>
ScreeningHasPatients
SendPatientToScreeningQueue
UrgencyHasPatients
VisitFile

3 Conclusão

Com este projeto conseguimos implementar e aprimorar o que aprendemos na cadeira de LP2 até à data.

Gostaríamos de ter implementado mais funcionalidades tal como guardar o log dos médicos, mas infelizmente a falta de tempo não nos permitiu faze-lo.