Notas importantes:

- ***Os repositórios devem ter o nome do grupo.***
- ***Deverão ser criadas branchs para cada funcionalidade nova a ser desenvolvida.***
- ***Devem enviar o nome do grupo e elementos dizendo qual o aluno respetivo para o email fsa@isep.ipp.pt até dia 3 de dezembro (atrasos serão penalizados), ex. :***
  - *Nome do grupo:* ***OsMelros***
  - *Aluno 1:* ***John Doe, nº1234567, j.doe@email.pt***
  - *Aluno 2:* ***Jane Doe, nº7654321, jane.doe@email.pt***
  - *Etc…*

Trabalho de Grupo Final

Sistema de Gestão de um Hospital de Urgências

Contexto

Pretende-se o desenvolvimento de uma aplicação em Java que simule a gestão das Urgências de um hospital, tendo em vista a aplicação prática dos conceitos de Programação Orientada a Objetos, herança, polimorfismo, encapsulamento, abstração, arrays/listas e manipulação de ficheiros.

O utilizador desempenha o papel de gestor hospitalar, responsável por coordenar o atendimento dos pacientes que chegam às urgências, atribuir médicos consoante as especialidades, monitorizar o estado dos doentes e garantir que o hospital opera de forma eficiente.

Objetivos

A aplicação deverá permitir:

- Ler ficheiros de entrada contendo informação sobre médicos e pacientes;
- Simular o processo de triagem e atendimento;
- Calcular automaticamente índices de urgência, atribuição de médicos e relatórios de desempenho;
- Interagir com o utilizador através de menus que permitam avançar o dia, alocar pacientes e consultar o estado do hospital;
- Consultar estatísticas;

Guardar, em ficheiro, todo o histórico de entradas e saídas, tanto de médicos, como pacientes.

A aplicação deverá ser desenvolvida em consola e deverá permitir a interação do utilizador com um menu para:

- Gerir médicos, especialidades e sintomas;
- Gerir o funcionamento do hospital;
- Consultar estatísticas e logs;
- Configurações.

Na área das configurações será uma zona onde iremos atualizar dados que não serão dinâmicos à aplicação:

- Caminho onde lê/grava os ficheiros;
- Separador utilizado nos ficheiros;
- Tempo de consulta
  - Baixa: 1 un.
  - Média: 2 un.
  - Urgente: 3 un.
- Tempo de descanso necessário dos médicos após horas trabalhadas: 1un de descanso por cada 5 horas seguidas trabalhadas (por exemplo)
- Elevação de nível de urgência com base em tempo esperado:
  - Baixa para Média: 3 un.
  - Média para Urgente: 3 un.
  - Urgente para saída de urgência: 2 un.
- Password de acesso às configurações;

Gestão de dados

Para o hospital funcionar corretamente, dados serão necessários serem introduzidos, seja por via de ficheiro, ou manualmente pelo utilizador.

Médico

- Nome
- Especialidade
- Hora de entrada
- Hora de saída
- Valor hora (em €)

Especialidade

- Código
- Nome

Sintoma

- Nome
- Nível de urgência :
    - Verde (Baixa)
    - Laranja (Média)
    - Vermelha (Urgente)
- Especialidades (opcional)

Para os **utentes**, os seus dados apenas serão inseridos no momento da admissão, ou seja, na triagem:

- Nome
- Sintomas

Funcionamento Geral do Hospital

Como referido anteriormente, o grande objetivo da aplicação será gerir o funcionamento de uma urgência de um hospital. Para isso, iremos simular a entrada e saída de utentes e atribuí-los a médicos disponíveis.
Ao iniciar a aplicação, a mesma terá de ler o ficheiro de médicos, especialidades e sintomas.

Depois da leitura dos dados existentes, o gestor deve ser capaz de registar os clientes, à medida que vão chegando à urgência e chamá-los para triagem, quando esta estiver livre.

Os utentes, ao realizarem triagem, indicam que sintomas têm, e com base nesses sintomas, o seu nível de urgência deve ser calculado bem como a especialidade, quando for possível. Importante realçar que os sintomas podem ser adicionados no momento da triagem, se for um sintoma novo.

Para ajudar o utilizador, deve ser implementado uma forma inovadora de pesquisa de sintomas que seja intuitiva, rápida e fácil de usar, sem comprometer com a experiência de utilização do software.

Após a triagem ser concluída, o utente pode ser encaminhado automaticamente para um médico disponível conforme a sua especialidade, ou então aguardar na sala de espera até que o gestor encaminhe o mesmo para um médico.

O sistema para automatizar o encaminhamento de clientes, deve calcular automaticamente a especialidade, com base nos sintomas (se possível). A especialidade é calculada com base no número maior de sintomas por nível de urgência. No caso de sintomas urgentes, deve ser, automaticamente, este sintoma que indica qual a especialidade, ignorando os sintomas leves e moderados.

Se não for possível calcular, por exemplo porque todos os sintomas não têm especialidade, o cliente não pode ser encaminhado automaticamente.

O sistema, deve também, mostrar ao utilizador que existem sintomas que poderão ser catalogados numa certa especialidade, visto que, geralmente, 80% ou mais de clientes com estes sintomas, são vistos por um médico de especialidade X.

De forma a avaliar o desempenho do hospital, necessitamos de algumas estatísticas que nos ajudem:

- Média de utentes atendidos por dia
- Tabela de salários recebidos por médico por dia
- Tabela com número de utentes com os sintomas
- Top 3 de especialidades com % de pacientes

Gestão das unidades de tempo

De forma a conseguirmos cumprir com os tempos necessários para apresentação da aplicação, iremos simular as unidades de tempo com base no avanço das mesmas, pelo utilizador.

Cada dia tem 24 unidades de tempo e à medida que avançamos uma unidade de tempo, através de uma opção no menu da aplicação, a hora avança. Isto significa, que informamos a aplicação que estamos a avançar no tempo à medida que vamos escolhendo a opção.

Cada vez que avançamos com o tempo, teremos de ter uma notificação que avisa o utilizador que algo aconteceu (Médico X ficou disponível; Cliente Y, com urgência nível Z avançou para nível seguinte). Estas notificações poderão ser estendidas conforme as necessidades e ficam ao critério de cada grupo de as ter ou não mas são altamente recomendadas.

Quando se passarem 24 unidades de tempo, recomeçamos novamente da unidade de tempo 1 e todos os cálculos serão realizados novamente.

Tal como já explicado, os médicos têm hora de entrada e hora de saída.
Importante realçar que o médico deve sempre ficar disponível na hora que está configurada na sua ficha, mas pode sair depois da hora configurada, se estiver a meio de um serviço.

Desenvolvimento do trabalho

Sugere-se que cada aluno seja responsável por uma área específica da aplicação de forma que não sejam prejudicados na apresentação. Sugestão de divisão para um grupo de 4 alunos:

- Aluno 1: Gestão de CRUDs (Create, Read, Update, Delete) necessários;
- Aluno 2: Gestão de estatísticas e cálculos automáticos;
- Aluno 3: Implementação dia-a-dia;
- Aluno 4: Configurações, desenvolvimento menu da aplicação, notificações, logs.

Esta informação deve ser enviada no momento do envio de e-mail com a informação do grupo!

Cada funcionalidade deverá estar numa branch específica e deve ser juntada depois de testada com a master através de um Pull Request.

Toda a informação da aplicação deve ser persistente, isto é, deve ser gravada em ficheiros e a informação não deverá ser perdida. No entanto, apenas deverá ser gravada quando saímos da aplicação ou numa opção para o mesmo efeito.

No arranque da aplicação, toda a informação da aplicação deverá ser lida através de ficheiros existentes (clientes, mesas, estatísticas, etc..). Mais tarde, será comunicado a estrutura que os ficheiros deverão respeitar.

Todo o trabalho pode ser feito em paralelo com os colegas e não é necessário esperar por qualquer desenvolvimento. Para isso, podem criar métodos que simulem os dados que esperarão quando algum colega terminar alguma parte pela qual é responsável:

```
using System;

public class Program
{
    public static void Main()
    {
        MostrarClientesAEspera();
    }

    public static void MostrarClientesAEspera(){
        // aqui deviamos ir buscar os clientes à espera.
        // esta informação está num ficheiro de texto mas como ainda não tenho esta funcionalidade pronta
        // fazemos assim:

        string[] clientes = ObterNomesClientes();

        Console.WriteLine("Clientes à espera de momento: ");

        for(int i=0;i<clientes.Length;i++){
            Console.WriteLine(clientes[i]);
        }
    }

    public static string[] ObterNomesClientes(){
        string[] nomes = new string[3];

        nomes[0] = "Vitor Cunha";
        nomes[1] = "Miguel Sá";
        nomes[2] = "Luiz Faria";

        return nomes;
    }
}
```

Avaliação

- Funcionalidade – 50%
- Estrutura do código – 30%
- Interatividade – 10%
- GIT – 10%

Mais Valias

Funcionalidades extra não pedidas, mas que melhorem o trabalho, são permitidas só e apenas com todas estes requisitos desenvolvidos. Não serão avaliadas mais-valias se alguma das funcionalidades descritas não existirem.

Notas

- É totalmente proibida a utilização de ArrayLists. Não será avaliado nenhum trabalho com ArrayLists.
- Deverão guardar apenas em ficheiro ao sair da aplicação ou em alguma opção para tal.
- Todos os desenvolvimentos devem seguir boas práticas aprendidas até ao momento. Sempre que possível, utilize reutilização de código.
- Relembramos que todo o trabalho deve ser realizado pelo grupo. Qualquer suspeita de cópia de código seja por onde for, levará a desconsiderar a avaliação e todo o grupo terá 0 como valor final.

[20 valores]