UNIVERSIDADE da MADEIRA

Faculdade Ciências Exatas e da Engenharia
Licenciatura em Engenharia Informática
Engenharia de Software

Docentes:
Leonel Nóbrega
Carlos Dória
Nuno Santos

Discentes:
Marcos Fernandes 2041518
Rodrigo Pestana 2079821
Diogo Martins 2082921
Bernardo Coelho 2079221

Índice

1. Introdução

No âmbito da unidade curricular de Engenharia de Software, foi-nos proposto a implementação de um sistema que permita criar uma rede social em que o tema central é o cinema. Nesta rede social, os amantes do cinema poderão partilhar as suas preferências, os seus conhecimentos e as suas opiniões sobre os filmes. O nosso sistema chama-se “Skai Movies”.

Neste relatório será abordado o desenvolvimento e padrões utilizados durante o projeto.

Figura 1 - Logo “Skai Movies”

1.1. Descrição do Projeto

A visão do projeto consiste na criação de uma rede social de tema cinematográfico, onde os utilizadores terão a liberdade de avaliar e/ou comentar filmes, deixando a sua opinião sobre cada um e até podendo participar de um fórum para iniciar várias discussões sobre os mesmos.

Os filmes e as suas informações serão disponibilizados por uma API, e os utilizadores poderão adicioná-los a coleções de filmes. Os utilizadores podem avaliar os filmes com um sistema de “gostos” e “não gostos”, podendo também expressar a sua opinião sobre o filme nos comentários.

As coleções juntam vários filmes ao gosto do utilizador. Estas coleções poderão ser privadas ou públicas, o que dita se estarão visíveis no perfil do utilizador. Existem algumas coleções pré definidas, podendo o utilizador adicionar e editar outras à sua escolha. O perfil também contém os posts feitos no fórum pelo utilizador.

O fórum irá conter vários posts feitos por outros utilizadores, que podem ser comentados e avaliados com um “gosto” ou um “não gosto”. É possível criar um post escolhendo um título e o conteúdo do post em si.

O utilizador terá a liberdade de editar o seu nome, foto de perfil, email e password.

2. Análise da Proposta

2.1. Requisitos

Na etapa anterior, foram solicitados alguns requisitos para o desenvolvimento da aplicação, analisaremos então os requisitos e entenderemos o que foi possível implementar.

2.1.1. Requisitos funcionais

RF01. - O sistema deverá autenticar os utilizadores. ✔️

RF02. - O sistema deverá mostrar ao utilizador o catálogo de filmes disponíveis. ✔️

RF03. - O sistema deverá permitir aos utilizadores avaliar os filmes. ❌

RF04. - O sistema deverá permitir aos utilizadores comentar os filmes. ❌

RF05. - O sistema deverá mostrar as informações sobre os filmes (titulo, realizador, cartaz, duração, gênero, elenco, resumo). ✔️

RF06. - O sistema deverá mostrar os trailers dos filmes. ❌

RF07. - O sistema deverá permitir criar e apagar coleções de filmes. 🟡

RF08. - O sistema deverá permitir adicionar e remover filmes das coleções.🟡

RF09. - O sistema deverá ser capaz de pesquisar filmes. ✔️

RF10. - O sistema deverá ser capaz de filtrar pesquisas. ✔️

RF11. - O sistema deverá permitir aos utilizadores realizar posts. 🟡

RF12. - O sistema deverá permitir aos utilizadores responder a posts. 🟡

RF13. - O sistema deverá ter um fórum com todos os posts e as suas respostas. 🟡

RF14. - O sistema deverá ter um botão para ligar e desligar o dark mode. ❌

2.1.2. Requisitos Não Funcionais

RNF01. - Desempenho: o sistema tem que estar disponível a qualquer altura e ter tempos de resposta reduzidos. ✔️

RNF02. - Segurança: O sistema tem que garantir o tratamento seguro das informações pessoais dos utilizadores. ✔️

RNF03. - Usabilidade: É preciso garantir uma boa experiência ao utilizador, e que seja simples de usar o sistema ✔️

RNF04. - Escalabilidade: O sistema deve ser capaz de lidar com um aumento na carga de trabalho, seja através do aumento do número de utilizadores ou do volume de dados. 🟡

2.1.3. Requisitos Tecnológicos

RT01. - O sistema deverá ser compatível com todos os sistemas operativos.❌

RT02. - O sistema deverá ter acesso a Internet. ✔️

RT03. - O sistema deverá usar uma API para obter os dados sobre os filmes.✔️

RT04. - O sistema deverá ter acesso a uma base de dados. ✔️

3. Desenvolvimento

3.1. Módulos

O sistema é constituído por várias classes, no entanto os alunos acharam melhor dividir o sistema em 4 módulos principais, sendo estes:

- User - tratamento de registo e interação do utilizador com a BD.
- Movies - apresentação das instâncias movie e uma das classes mais importantes para o sistema.
- Collections - conjunto de movies com várias funcionalidades específicas.
- Fórum - onde todos os posts e comments estarão.

3.2. Padrões de desenho

Os padrões de desenho são microarquiteturas que permitem resolver um problema recorrente num dado contexto de programação. Um dos principais objetivos de Engenharia de Software é a implementação de padrões de desenho adequados para cada contexto dos problemas apresentados.

3.2.1. Ideias de implementações

Começámos por ter ideias de implementar builders para criar posts, movies e comentários, no entanto, notámos que este não seria uma boa solução para este problema, dado que só necessitaríamos de um construtor para cada uma destas classes ao longo do projeto inteiro.

Considerámos também o uso de iterators para percorrer várias listas e árvores ao longo do programa, mas este demonstrou-se confuso e pouco eficiente, sendo que utilizámos o comando “foreach” para a maioria dos percursos de listas.

Os alunos pensaram em utilizar observers para utilização com vários botões e com a barra de pesquisa, mas não se demonstrou necessário.

Foi também pensado a utilização do padrão de desenho State para executar o programa de formas diferentes, de modo a distinguirmos entre o user em autenticado e não autenticado, no entanto com a nossa implementação de autenticação, isto não foi necessário. Dos padrões descartados seria este o mais realístico a ser implementado.

Durante a criação de um post, os alunos tentaram implementar o padrão de Prototype, para podermos criar um post protótipo e depois de concluído, fazermos um clone dele próprio, no entanto isto tornou-se muito confuso e os alunos desistiram da ideia.

3.2.2. Composite

Os alunos analisaram o projeto e notaram que o padrão de desenho Composite poderia ser utilizado para criar uma árvore de comentários em cada post, dado que cada post pode ter vários comentários e cada comentário pode ter respostas. No entanto, os alunos modificaram um pouco o padrão, de modo a que cada objeto da árvore seja um Composite, dado que todos os comentários podem ter mais comentários.

A função que cada elemento da árvore executa é um render de uma componente comentário, para que se possa apresentar o comentário no razzor.

```
1   using Microsoft.AspNetCore.Components;
2
3   namespace ProjetoES.Padroes
4   {
        14 references
5       public interface ICommentItem
6       {
            3 references
7           public string User { get; }
            3 references
8           public string Texto { get; }
            2 references
9           public int ID { get; }
10
            3 references
11          public RenderFragment RenderComponent();
12      }
13  }
```

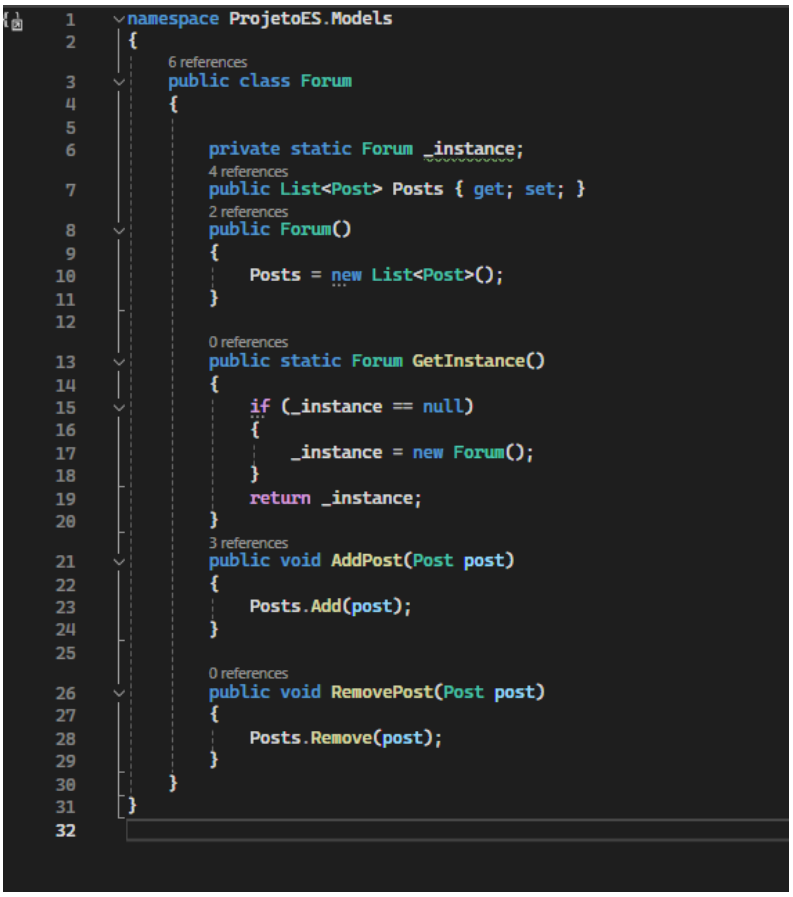
Figura 2 - Implementação da interface ICommentItem

```
3   namespace ProjetoES.Padroes
4   {
        11 references
5       class CompositeComment : ICommentItem
6       {
7           private readonly List<ICommentItem> commentItems = new List<ICommentItem>();
8
            3 references
9           public String User { get; }
            3 references
10          public String Texto { get; }
            2 references
11          public int ID { get; }
12
            4 references
13          public CompositeComment(String user, String texto, int id ):base() {
14              User = user;
15              Texto = texto;
16              ID = id;
17          }
18
19
            3 references
20          public void AddItem(ICommentItem item) => commentItems.Add(item);
            0 references
21          public void RemoveItem(ICommentItem item) => commentItems.Remove(item);
22
            0 references
23          public List<ICommentItem> GetCommentItems()
24          {
25              return commentItems;
26          }
27
            3 references
28          public RenderFragment RenderComponent() => builder =>
29          {
30              // Renderização do CompositeComponent
31              builder.OpenElement(0, "div");
32              builder.AddContent(0, $"User: {User}");
33              builder.AddContent(0, $"Texto: {Texto}");
34              foreach (var child in commentItems)
35              {
36                  builder.AddContent(1, child.RenderComponent());
37              }
38              builder.CloseElement();
39          };
40      }
41  }
```

Figura 3 - Implementação do elemento CompositeComment da árvore

3.2.3. Singleton

Dado que só teremos 1 fórum, e não necessitamos de estar sempre a criar instâncias da classe fórum sempre que aderimos à página fórum, decidimos implementar o padrão de desenho Singleton, de modo a que uma só instância fórum seja utilizada durante a execução do programa.



```
namespace ProjetoES.Models
{
    6 references
    public class Forum
    {

        private static Forum _instance;
        4 references
        public List<Post> Posts { get; set; }
        2 references
        public Forum()
        {
            Posts = new List<Post>();
        }

        0 references
        public static Forum GetInstance()
        {
            if (_instance == null)
            {
                _instance = new Forum();
            }
            return _instance;
        }
        3 references
        public void AddPost(Post post)
        {
            Posts.Add(post);
        }

        0 references
        public void RemovePost(Post post)
        {
            Posts.Remove(post);
        }
    }
}
```

Figura 4 - Implementação do singleton na classe Forum

3.3. API dos filmes

A api utilizada foi a “themoviedb”, e para implementá-la os alunos necessitam de usar métodos assíncronos e implementar uma interface de acesso à api. Existem métodos que retornam uma lista de filmes populares no momento e também métodos que retornam um objeto Movie dependendo do id fornecido. A implementação da API foi relativamente complicada, mas foi feita com sucesso e funciona sem erros.

3.4. Base de Dados e Autenticação

Após assistirmos a várias documentações, tutoriais e consulta de outros projetos, construímos a autenticação utilizando um template, com auxílio de tutoriais, após tentarmos várias vezes fazer a autenticação sozinhos.

A base de dados incorporou a Entity Framework, simplificando assim o acesso e manipulação de dados ao mapear automaticamente entidades do modelo de domínio para tabelas no banco de dados, sendo o único problema a não implementação da visualização dos dados. Os alunos sentiram que este módulo do projeto é dos mais dependentes dos outros, e foi o módulo em que se gastou mais tempo e recursos.

4. Guia de instalação/Execução

Utilizámos o git para interagir entre a equipa durante o desenvolvimento do programa. Abaixo está disponibilizado o link do repositório do git:

https://github.com/dbmartins11/ProjetoES

A execução do projeto é relativamente simples: É necessário se registar para aceder a algumas funcionalidades, mas de resto o programa é bastante intuitivo.

5. Conclusões

Com a realização deste projeto, os alunos conseguiram analisar a ideia inicial do trabalho e tentar aplicar vários padrões de desenho para desenvolvermos a rede social. Aprendemos onde e quando implementar padrões de desenho, descartando várias ideias ao longo do desenvolvimento por não se adequarem ao contexto de programação dado. Poderíamos ter implementado uma melhor interface e melhorado o css de modo, mas o resultado final é satisfatório.

No entanto, os alunos encontraram vários erros e problemas ao longo do processo de autenticação, pois foi necessário implementar várias interfaces obtidas da internet, criar novos projetos e assistir a múltiplos tutoriais na internet para conseguirmos eventualmente implementar uma autenticação de utilizador. Isto demonstrou-se um grande obstáculo, pois as features que consideramos mais importantes estão bastante dependentes da autenticação, da existência de uma classe “user”, e também da ligação à base de dados para guardar e apresentar dados na maioria das páginas.

Esta própria ligação à base de dados demonstrou-se complicada para os estudantes e sentimos que grande parte do tempo de desenvolvimento foi gasto/perdido neste módulo: a criá-lo, no tratamento de erros na base de dados e em migrations. Sentimos que não nos focámos tanto na matéria lecionada durante as aulas teóricas devido a isto e não há dúvidas que os alunos teriam implementado mais padrões como iterators para percorrer listas e a própria árvore de comentários ou observers para alguns botões e até mesmo, por exemplo, a barra de pesquisa, se as ligações às bases de dados não estivessem constantemente a dar erros, por vezes incompreensíveis para os próprios alunos.

Por suma, sabemos que a interação com autenticação e base de dados é uma parte importante na aprendizagem e na implementação do projeto, e até é importante para a vida profissional dos alunos, mas gastou-nos demasiado tempo e sentimos que ficámos a “dever” padrões ao projeto.

6. Referências

Alguns tutoriais utilizados:

https://www.youtube.com/watch?v=CRR6HPUPgsY

https://youtu.be/b8fFRX0T38M?si=37ecX8F9OVP6w4AQ

https://www.youtube.com/watch?v=tNzSuwV62Lw

https://www.youtube.com/watch?v=8ZTpCTJ8FM0

Sites utilizados para padrões:

https://refactoring.guru/design-patterns/composite

https://refactoring.guru/design-patterns/singleton