**Relatório de Projeto**

**de**

**Laboratório de Projeto em Engenharia Informática**

**da**

**Licenciatura em Engenharia Informática**

| Nome do aluno | Número do aluno |
| --- | --- |
| **Pedro Afonso Pires** | **77181** |

| Título do trabalho |
| --- |
| Sistema de Gestão de Projetos - Backend |

| Nome do orientador |
| --- |
| **Luís Filipe Leite Barbosa** |

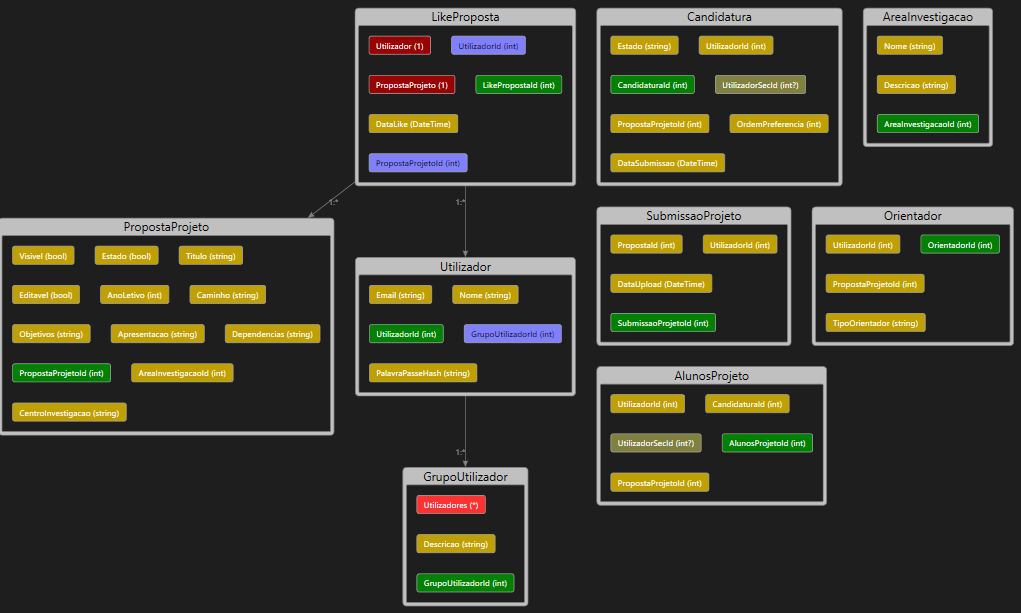
| Nome do(s) coorientador(es) |
| --- |
| **António Manuel Ribeiro de Sousa** |

**1 – Tema do projeto**

*A gestão dos Projetos de Final de Cursono contexto da unidade curricular de Laboratório de Projeto em Engenharia Informática tem sido, tradicionalmente, um processo manual e descentralizado. Atualmente, candidaturas são feitas através do preenchimento de folhas de Excel, a comunicação entre alunos e docentes ocorre por e-mail, e o registo de propostas, orientadores e grupos é realizado de forma isolada. Esta abordagem está sujeita a erros, atrasos e retrabalho, dificultando o acompanhamento e controlo de todo o processo.*

*Face a este cenário, foi proposto o desenvolvimento de uma* ***aplicação web de gestão de projetos****, com o objetivo de digitalizar e automatizar estas tarefas. O sistema é composto por um* ***frontend (interface web)*** *e um* ***backend (API RESTful)*** *que comunica com a base de dados e fornece os recursos necessários às várias operações dos utilizadores.  
Este relatório foca-se exclusivamente no* ***desenvolvimento da API backend****, a qual assegura a gestão centralizada da informação e coordenação entre os vários intervenientes (alunos, docentes, orientadores).*

**2 – Diagrama de Base da Dados**

****

Abaixo, uma breve descrição das principais tabelas e respetivo propósito no contexto da aplicação:

* Utilizador: Representa os utilizadores da plataforma (alunos, docentes, coordenadores).
* GrupoUtilizador: Define os perfis dos utilizadores e respetivos privilégios.
* PropostaProjeto: Armazena as propostas de projetos submetidas por docentes.
* Candidatura: Regista as candidaturas dos alunos às propostas existentes.
* AlunosProjeto: Faz a associação entre alunos e projetos atribuídos.
* Orientador: Armazena a informação de orientadores atribuídos a propostas, podendo incluir internos e externos.
* AreaInvestigacao: Categoriza as propostas com base em áreas científicas.
* SubmissaoProjeto: Guarda os dados das entregas finais realizadas pelos alunos.
* LikeProposta: Permite aos utilizadores marcar propostas como favoritas.

**3 – Principais funcionalidades**

A aplicação foi desenhada para cobrir os principais fluxos envolvidos na gestão de projetos. As funcionalidades implementadas na API incluem:

1. **Gestão de Utilizadores**: Suporte a perfis distintos (aluno, docente, coordenador), com controlo de acesso através de tokens JWT.
2. **Gestão de Propostas**: Os docentes podem criar, atualizar e eliminar propostas. As propostas são visíveis aos alunos.
3. **Candidaturas**: Alunos podem candidatar-se a propostas abertas durante o período definido.
4. **Associação de Alunos a Projetos**: Permite formar grupos de trabalho. O número de alunos por grupo é definido pela edição da UC.
5. **Gestão de Orientadores**: Atribuição de um ou mais orientadores (internos ou externos) às propostas. Sem limite fixo de coorientadores.
6. **Áreas de Investigação**: As propostas podem ser associadas a uma ou mais áreas temáticas.
7. **Submissão Final de Projetos**: Funcionalidade para alunos enviarem os ficheiros finais.
8. **Sistema de Favoritos (Likes)**: Os utilizadores podem marcar propostas como favoritas, ajudando a destacar as mais atrativas.

**4 - Tecnologias Utilizadas**

A escolha das tecnologias teve em conta critérios como robustez, integração com outras ferramentas, e familiaridade com o ecossistema .NET. A seguir descreve-se o propósito de cada tecnologia:

* **ASP.NET Core 8**: Framework principal utilizada para desenvolver a API REST, pela sua performance, segurança e suporte a aplicações escaláveis.
* **Entity Framework Core**: ORM (Object-Relational Mapping) que permite interagir com a base de dados de forma eficiente, reduzindo a necessidade de SQL explícito.
* **SQL Server**: Sistema de base de dados relacional escolhido pelo seu bom desempenho com aplicações .NET.
* **Swagger**: Utilizado para documentação e teste da API, permitindo que endpoints possam ser visualizados e testados de forma interativa.
* **JWT (JSON Web Token)**: Sistema de autenticação leve e seguro, utilizado para validar a identidade de utilizadores em cada pedido HTTP.

**Alternativas possíveis**:

* Para ORM: Dapper ou NHibernate.
* Para base de dados: PostgreSQL ou MySQL.
* Para autenticação: OAuth2 ou IdentityServer4.

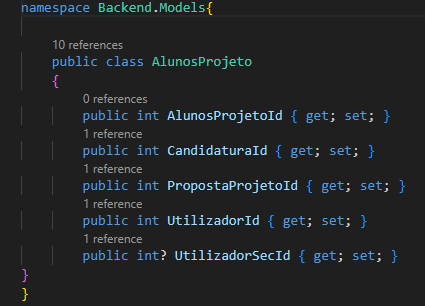
**5 - Arquitetura**

A arquitetura da aplicação segue o padrão **MVC (Model-View-Controller)**, com uma clara separação entre responsabilidades. A estrutura da API backend inclui:

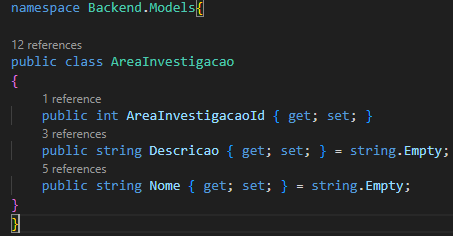
* **Modelos**: Representam as entidades do domínio.
* **DTOs (Data Transfer Objects)**: Utilizados para transferir dados entre camadas, protegendo o modelo de domínio de exposições externas.
* **Controladores**: Contêm a lógica de encaminhamento dos pedidos HTTP e chamadas ao serviço correspondente.
* **Serviços** (quando aplicável): Camada de lógica de negócio.
* **DbContext**: Responsável pela ligação entre a aplicação e a base de dados via Entity Framework.

A autenticação JWT permite verificar a identidade do utilizador em cada requisição, garantindo segurança e personalização de funcionalidades.

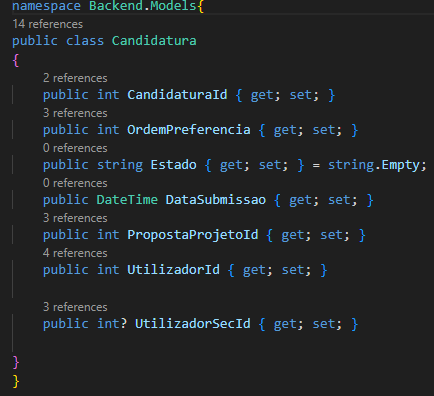
**6 – Modelo de Dados**

AlunosProjeto.cs  
****

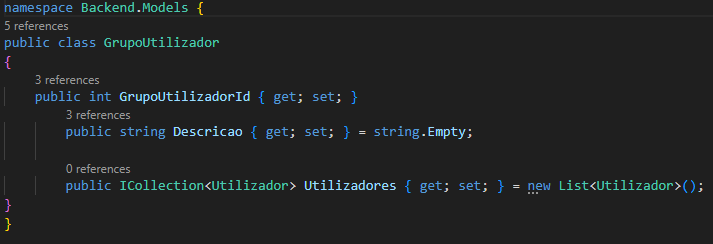
AreaInvestigacao.cs

******

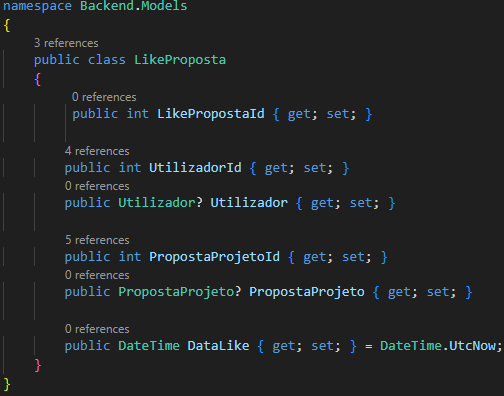
Candidatura.cs

******

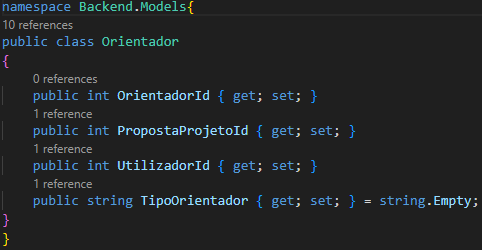
GrupoUtilizador.cs

******

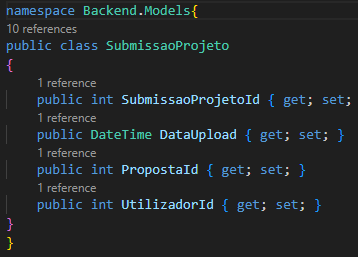
LikeProposta.cs

******

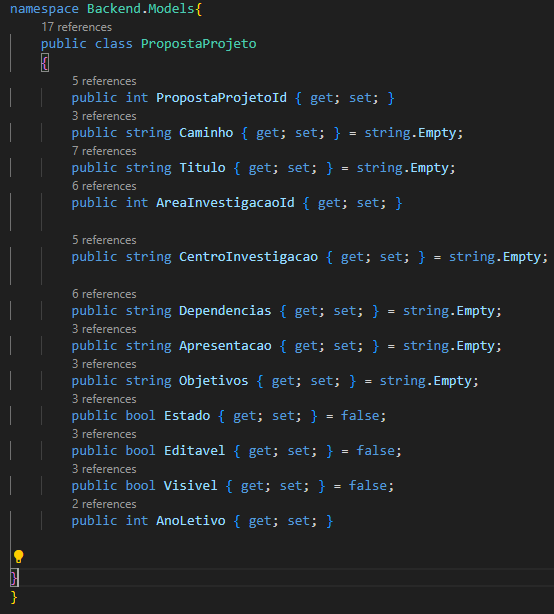
Orientador.cs



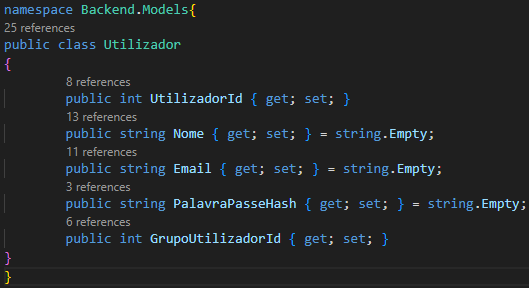
SubmissaoProjeto.cs

******

PropostaProjeto.cs

******

Utilizador.cs

****

***7 - Endpoints e o seu significado***

A API foi estruturada em torno de diferentes controladores, cada um responsável por um subconjunto funcional do sistema. Abaixo, apresenta-se uma descrição dos controladores (Controllers) e os principais endpoints que os compõem:

#### AccountController

**Função:** Responsável pela autenticação simulada dos utilizadores.  
 **Endpoints principais:**

* POST /api/Account/LoginSimulado:   
  Simula o login de um utilizador com base no email, devolvendo um token JWT.  
   Não existe validação na base de dados; este método serve apenas para testes.

#### AlunosProjetoController

**Função:** Permite a associação de alunos a projetos.  
 **Endpoints principais:**

* POST /api/AlunosProjeto: Associa um aluno a uma proposta de projeto.
* GET /api/AlunosProjeto: Lista todas as associações criadas.
* DELETE /api/AlunosProjeto/{id}: Remove uma associação específica.

#### AreaInvestigacaoController

**Função:** Gestão das áreas de investigação associadas às propostas de projeto.  
 **Endpoints principais:**

* GET /api/AreaInvestigacao: Lista todas as áreas.
* GET /api/AreaInvestigacao/{id}: Obtém os detalhes de uma área.
* POST /api/AreaInvestigacao: Cria uma nova área de investigação.
* PUT /api/AreaInvestigacao/{id}: Atualiza uma área existente.
* DELETE /api/AreaInvestigacao/{id}: Elimina uma área.

#### CandidaturaController

**Função:** Gerir candidaturas dos alunos às propostas.  
 **Endpoints principais:**

* GET /api/Candidatura: Lista todas as candidaturas.
* GET /api/Candidatura/{id}: Consulta uma candidatura específica.
* POST /api/Candidatura: Submete uma nova candidatura.
* PUT /api/Candidatura/{id}: Atualiza a candidatura.
* DELETE /api/Candidatura/{id}: Remove uma candidatura.

#### OrientadorController

**Função:** Atribuição de orientadores a propostas de projeto.  
 **Endpoints principais:**

* POST /api/Orientador: Atribui um orientador a uma proposta.
* GET /api/Orientador: Lista todos os orientadores atribuídos.
* DELETE /api/Orientador/{id}: Remove uma associação de orientador.

#### PropostasController

**Função:** Gestão completa das propostas de projeto.  
 **Endpoints principais:**

* GET /api/Propostas: Lista todas as propostas.
* GET /api/Propostas/{id}: Detalhes de uma proposta.
* POST /api/Propostas: Cria uma nova proposta.
* PUT /api/Propostas/{id}: Atualiza uma proposta existente.
* DELETE /api/Propostas/{id}: Elimina uma proposta.
* POST /api/Propostas/{id}/like: Adiciona um “like” a uma proposta.
* GET /api/Propostas/{id}/likes: Consulta o número de “likes”.
* DELETE /api/Propostas/{id}/remover-like: Remove o “like” de um utilizador.

#### SubmissaoProjetoController

**Função:** Permite a submissão final dos projetos pelos alunos.  
 **Endpoints principais:**

* GET /api/SubmissaoProjeto: Lista todas as submissões.
* GET /api/SubmissaoProjeto/{id}: Consulta uma submissão específica.
* POST /api/SubmissaoProjeto: Submete o projeto final.
* DELETE /api/SubmissaoProjeto/{id}: Remove uma submissão.

#### UtilizadorController

**Função:** Gestão de utilizadores da plataforma.  
 **Endpoints principais:**

* GET /api/Utilizador: Lista todos os utilizadores.
* GET /api/Utilizador/{id}: Consulta os detalhes de um utilizador.
* POST /api/Utilizador: Cria um novo utilizador.
* PUT /api/Utilizador/{id}: Atualiza os dados de um utilizador.
* DELETE /api/Utilizador/{id}: Elimina um utilizador.

**8 – Conclusão**

O desenvolvimento deste Sistema de Gestão de Projetos permitiu criar uma plataforma robusta e escalável, centrada na automação de tarefas essenciais associadas à unidade curricular de Laboratório de Projeto em Engenharia Informática. Através da criação de uma API REST estruturada, foi possível integrar funcionalidades críticas como a autenticação de utilizadores, a gestão de propostas, candidaturas, e submissões finais, assegurando uma experiência coerente e eficiente para todos os intervenientes – alunos, docentes e orientadores.

A adoção do padrão arquitetural MVC, aliada ao uso de tecnologias modernas como ASP.NET Core 8, Entity Framework Core, JWT e Swagger, garantiu uma implementação modular, segura e facilmente testável. A estrutura em camadas, com separação entre modelos, DTOs e controladores, reforça a manutenibilidade e extensibilidade do sistema.

Com este projeto, não só se respondeu a necessidades reais de digitalização e centralização dos processos académicos, como também se aplicaram de forma prática conhecimentos adquiridos ao longo do curso, contribuindo para o crescimento técnico e profissional dos intervenientes.

Vila Real, 27 de Fevereiro de 2025

*Pedro Afonso Pires*

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**