Texto

Descrição gerada automaticamente

21-12-2024

Rodrigo Moura

Instituto Politécnico do Cávado e do Ave

Space System Report

Subject: Object-Oriented Programming

Teacher: Ernesto Casanova

[1. *Project Context and Objective* 2](#_Toc185539743)

[2. *General System* 3](#_Toc185539744)

[The system was designed with a modular architecture that promotes a clear division of responsibilities among components, ensuring scalability and ease of maintenance. This approach allows each module to function independently while integrating seamlessly into the larger system. The main modules include: 3](#_Toc185539745)

[This modular architecture also enables future integration with external systems, such as real-time space monitoring APIs, trajectory simulators, or visual dashboards for data analysis. 3](#_Toc185539746)

[3. *System Features* 4](#_Toc185539747)

[4. *Testes e Validação* 5](#_Toc185539748)

[5. *Tecnologias e medologias utilizadas* 6](#_Toc185539749)

[*6. Flow and Class Diagram* 7](#_Toc185539750)

[1. *Objetivo da 2ª Fase* 10](#_Toc185539751)

[2. *Arquitetura do Sistema* 11](#_Toc185539752)

[3. *Tecnologias e Ferramentas utilizadas* 12](#_Toc185539753)

[4. *Desenvolvimento do Projeto* 13](#_Toc185539754)

[5. *Vantagens do Sistema expandido* 14](#_Toc185539755)

[6. *Medologias de Implementação* 15](#_Toc185539756)

[7. *Modelagem de Dados* 16](#_Toc185539757)

[8. *Considerações Finais* 17](#_Toc185539758)

# **1.** *Project Context and Objective*

Space exploration has become one of the most dynamic and innovative sectors of modern technology, contributing significantly to technological advancements, international collaborations, and new economic opportunities. Major organizations, such as government space agencies and private companies, have invested in solutions to expand the boundaries of human knowledge, promote space tourism, and develop sustainable technologies for humanity's future in the cosmos.

This project was developed with the objective of simulating and managing space operations in an integrated manner, covering:

1. **Interplanetary Missions:** Detailed simulations for missions to other planets, such as Mars, focusing on exploration and colonization.
2. **Space Station Operation**: Management of orbital stations and facilities dedicated to scientific research and tourism.
3. **Space Tourism**: Support for simulations of tourist experiences, including capacity management, safety, and maintenance.
4. **Launch Coordination**: Planning and execution of launches based on realistic parameters

The solution was designed to meet the needs of organizations aiming to conduct realistic simulations for training, research, or as a foundation for developing applications in the aerospace sector. Its modular architecture allows the system to be tailored to the specific demands of different segments of the space industry, ensuring scalability, reliability, and integration with emerging technologies.

# **2.** *General System*

* **Astronaut Management:** Responsible for registering, preparing, and monitoring astronauts involved in missions.
* **Launch Coordination**: Handles the planning and execution of launches, as well as calculating detailed schedules.
* **Mission Administration**: Focused on managing the objectives and specific characteristics of each mission.
* **Space Station Operations**: Ensures the operation and maintenance of orbital and tourism stations.

Each module was developed extensively using the C# programming language, adhering to object-oriented programming (OOP) principles. The project incorporates fundamental concepts such as:

* **Inherance**: For code reuse and extensibility among related classes. For example, classes like MarsMission and OrbitalMissioninherit functionality from the base class CMission.
* **Interfaces**: To standardize expected behaviors across different types of objects. Interfaces such as ILaunchable and ILaunchableMission ensure consistency in the implementation of critical functionalities.
* **Polymorphism**: To provide flexibility in the behavior of components in different contexts. For instance, the method PrepareForLaunch()is implemented specifically by classes like Astronaut and Rocket.

# This modular architecture also enables future integration with external systems, such as real-time space monitoring APIs, trajectory simulators, or visual dashboards for data analysis.

# **3.** *System Features*

##### **3.1 Astronaut Management**

* Registration of astronauts with detailed information, such as name, rank, and mission role.
* Evaluation of astronauts' physical and psychological readiness to ensure safety and mission effectiveness.
* Record of each astronaut s mission history

##### **3.2** Launch Coordination

* Planning of launches based on weather conditions and resource availability.
* Countdown simulation and preparation of the team and rocket.
* Relatórios de prontidão que verificam a adequação de todos os componentes para o lançamento.

##### **3.3 Administração de Missões**

* Criação e gerenciamento de missões com objetivos específicos, como exploração científica ou turismo espacial.
* Registro de missões passadas, incluindo detalhes como desempenho, desafios e resultados.
* Análise de dados históricos para otimização de futuras missões.

##### **3.4 Operação de Estações Espaciais**

* Monitoramento em tempo real de recursos da estação, como energia, oxigênio e suprimentos alimentares.
* Planejamento de atividades científicas e rotinas de manutenção.
* Coordenação de atividades de turismo espacial, garantindo segurança e conforto para os visitantes.

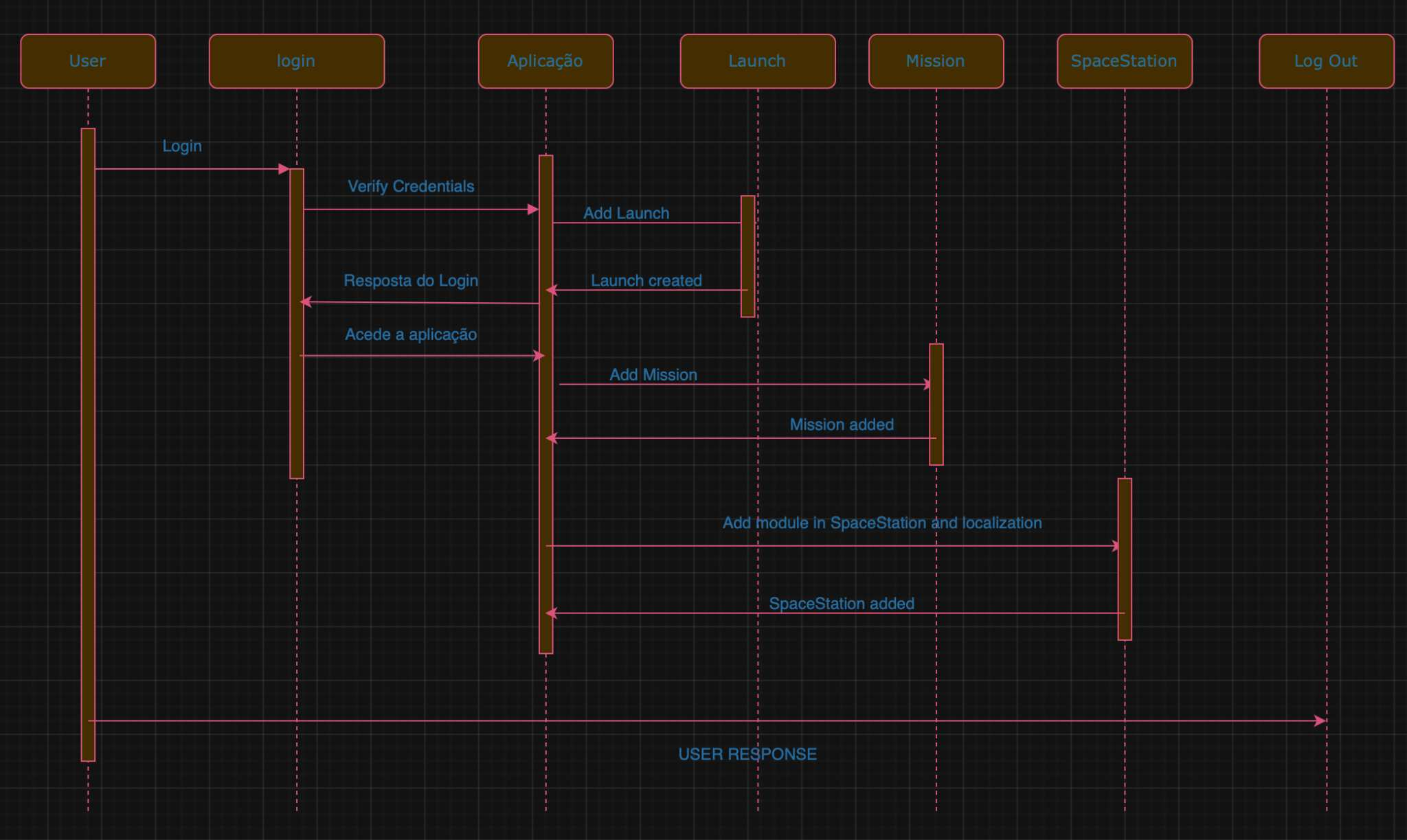
# **4.** *Testes e Validação*

* **Testes Unitários e de Integração**: Garantia de que os módulos do sistema funcionem corretamente de forma isolada e integrada.
* **Testes de Performance**: Verificação da escalabilidade e resposta do sistema sob cargas elevadas.
* **Testes de Usabilidade**: Avaliação da interface web para assegurar uma experiência intuitiva para o usuário.

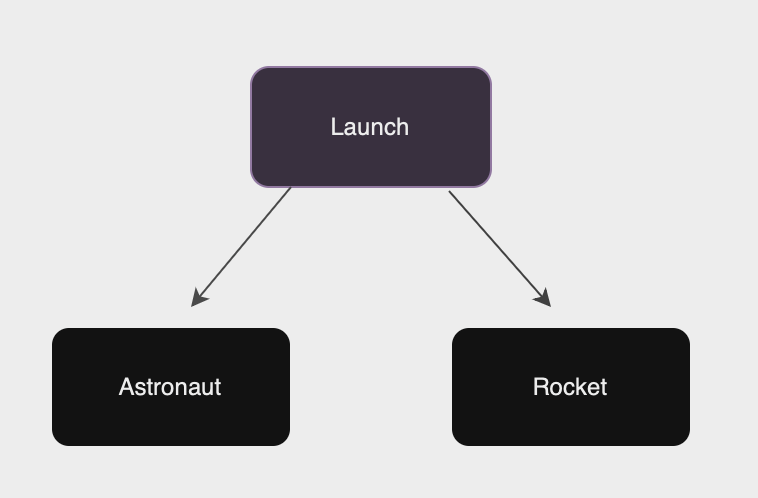
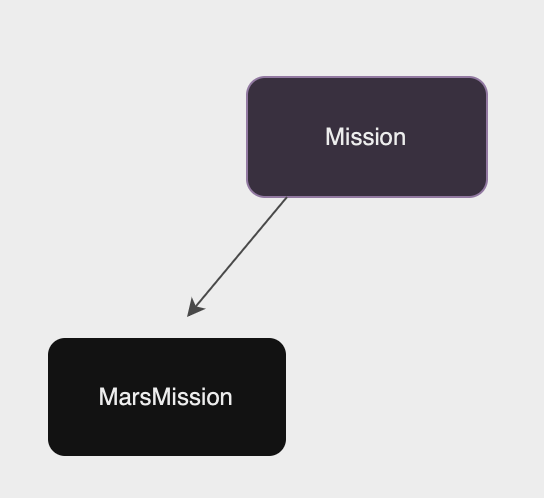
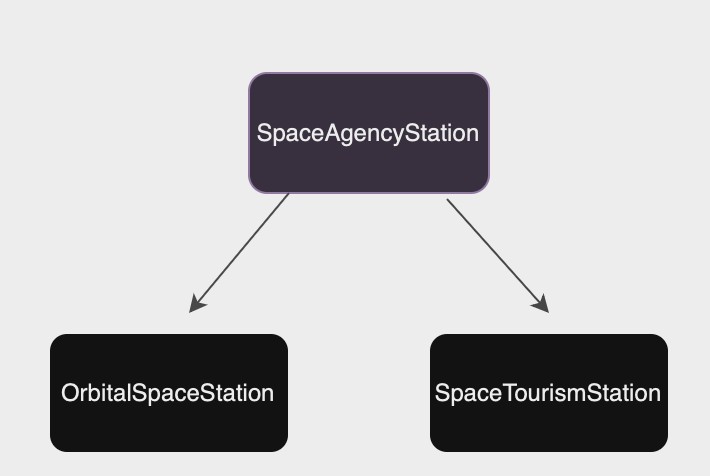
# **5.** *Tecnologias e medologias utilizadas*

* **Linguagem de Programação**: C#.
* **Paradigma**: Programação orientada a objetos (POO).
* **Práticas Adotadas**:
  + Uso extensivo de herança para maximizar a reutilização de código.
  + Implementação de interfaces para garantir consistência e padronização.
  + Polimorfismo para flexibilidade na execução de comportamentos específicos.

# *6. Flow and Class Diagram*



1.Flow Diagram



#### **7. Conclusão e Recomendações**

Este Sistema de Gestão Espacial oferece uma solução robusta e versátil para atender às demandas do setor aeroespacial, com aplicações que vão desde a exploração científica até o turismo espacial. A implementação de funcionalidades avançadas, aliada à arquitetura flexível, posiciona este sistema como uma ferramenta indispensável para organizações e instituições interessadas no futuro da exploração espacial.

Recomendações para evolução futura:

1. Desenvolvimento de uma interface gráfica para facilitar a interação com o sistema.
2. Integração com dados de satélites em tempo real para aprimorar a precisão das simulações.
3. Implementação de funcionalidades de aprendizado de máquina para prever e mitigar riscos em missões futuras.

2ª FASE DO PROJETO

(projeto sofreu uma alteração-Windows-Forms foi alterado para a realização de uma página web e foi decidido usar uma base de dados)

# **1.** *Objetivo da 2ª Fase*

A segunda fase do Sistema de Integração Espacial tem como objetivo central a expansão das capacidades da solução original, posicionando-a como uma plataforma tecnológica completa, segura e de alta performance para o gerenciamento e a simulação de operações espaciais. Essa etapa visa integrar um banco de dados relacional robusto, desenvolver uma API RESTful escalável e criar uma interface web responsiva e intuitiva. A implementação dessas melhorias tem como finalidade assegurar a persistência confiável dos dados, ampliar a acessibilidade ao sistema e proporcionar uma experiência de usuário aprimorada, consolidando a solução como uma referência no setor aeroespacial.

Nesta fase, será enfatizada a adoção de padrões arquiteturais modernos e a utilização de tecnologias de ponta, como ASP.NET Core e React.js, alinhadas às necessidades do mercado e às melhores práticas do setor de software. O resultado esperado é a entrega de um sistema altamente eficiente e flexível, capaz de atender demandas operacionais atuais e futuras, enquanto promove inovação tecnológica e colaboração interorganizacional.

# **2.** *Arquitetura do Sistema*

O sistema utiliza uma arquitetura baseada em camadas, separando responsabilidades em diferentes módulos:

* **SpaceSystem.API**: Contém os controladores e configurações da API.
* **SpaceSystem.Data**: Inclui os modelos de dados e o contexto do banco de dados.
* **SpaceSystem.Infrastructure**: Gerência os repositórios e interações com o banco de dados.

**Estrutura de Pastas:**

* Controllers/: Controladores da API.
* Data/: Contexto do banco de dados e migrações.
* Models/: Classes que representam as entidades do sistema.
* Repositories/: Interfaces e implementações de repositórios.

# **3.** *Tecnologias e Ferramentas utilizadas*

* **Linguagem**: C# (ASP.NET Core). (A mais utilizada ao longo do projeto)
* **Banco de Dados**: PostgreSQL.
* **ORM**: Entity Framework Core.
* **Ferramentas de Build e Deploy**: Visual Studio, Docker.
* **Documentação da API**: Swagger/OpenAPI.

# **4.** *Desenvolvimento do Projeto*

A segunda fase será composta por três principais vertentes:

1. **Persistência de Dados**:
   * Implementação de uma base de dados relacional para armazenar as informações do sistema, como astronautas, missões, estações espaciais e lançamentos.
   * Integração do banco de dados com a arquitetura existente utilizando Entity Framework Core.
2. **Desenvolvimento de API RESTful**:
   * Criação de endpoints para operações CRUD (Create, Read, Update, Delete) nos principais módulos do sistema.
   * Implementação de autenticação e autorização via JSON Web Tokens (JWT).

# **5.** *Vantagens do Sistema expandido*

A evolução proposta nesta fase trará os seguintes benefícios:

* **Persistência e Confiabilidade**:
  + Os dados serão armazenados de forma segura em um banco de dados relacional, garantindo que informações não sejam perdidas.
* **Acessibilidade Universal**:
  + A interface web e a API RESTful permitirão acesso ao sistema de qualquer dispositivo conectado à internet.
* **Escalabilidade**:
  + A arquitetura modular permite expansões futuras, como novos módulos ou integrações com outros sistemas.
* **Eficiência Operacional**:
  + Relatórios dinâmicos e dashboards intuitivos agilizarão a tomada de decisão e a gestão de operações espaciais.

# **6.** *Medologias de Implementação*

1. **Persistência de Dados**:
   * Banco de Dados: SQL Server ou PostgreSQL será utilizado para o armazenamento de dados.
   * Modelagem Relacional:
     + **Astronautas**: Id, Nome, Patente, Função.
     + **Missões**: Id, Nome da Missão, Objetivo, Tipo de Missão.
     + **Lançamentos**: Id, Missão Relacionada, Data, Nome do Foguete.
     + **Estações Espaciais**: Id, Nome, Tipo, Capacidade de Tripulação, Capacidade Turística.
   * Integração via Entity Framework Core para mapeamento dos modelos existentes para tabelas no banco de dados.
2. **API RESTful**:
   * Desenvolvimento de endpoints usando ASP.NET Core:
     + GET /api/astronauts: Retorna a lista de astronautas.
     + POST /api/missions: Cria uma nova missão.
     + PUT /api/stations/{id}: Atualiza os dados de uma estação.
     + DELETE /api/launches/{id}: Remove um lançamento.
   * Segurança:
     + Implementação de autenticação com JWT para proteger operações sensíveis.
3. **Interface Web**:
   * Desenvolvimento de frontend utilizando React.js, Angular ou Blazor.
   * Funcionalidades previstas:
     + **Dashboard**: Exibe resumo de dados, como próximos lançamentos e status das estações.
     + **Formulários Interativos**: Para gerenciamento de dados de astronautas, missões e lançamentos.
     + **Relatórios Gráficos**: Visualização de dados em gráficos dinâmicos para facilitar análises.

# **7.** *Modelagem de Dados*

**Entidades Principais:**

* **Astronaut**: Representa os astronautas.
  + Atributos: Id, Name, Role, MissionId.
* **Mission**: Gerencia missões espaciais.
  + Atributos: Id, Name, Type, StartDate, EndDate.
* **SpaceStation**: Controla estações espaciais.
  + Atributos: Id, Name, Location.
* **Rocket**: Representa espaçonaves.
  + Atributos: Id, Name, Capacity.

**Diagrama UML:** (Diagrama incluído separadamente.)

# **8.** *Considerações Finais*

A segunda fase do Sistema de Integração Espacial representa um passo significativo na evolução e ampliação das capacidades do sistema. Essa etapa não apenas consolida a solução como uma ferramenta robusta para o gerenciamento de operações espaciais, mas também a posiciona como uma referência para soluções tecnológicas no setor aeroespacial.

A incorporação de tecnologias modernas, como ASP.NET Core, React.js e base de dados relacionais, oferece alta performance e escalabilidade, essenciais para atender às demandas de usuários e organizações em cenários dinâmicos e complexos.