---

# Documentação de Projeto de Software

## Projeto: Integralizador Curricular

**Integrantes: Jansen, Renan e João**

---

## 1. Visão Geral do Projeto

O projeto "Integralizador Curricular" é uma aplicação de desktop desenvolvida em Java com o objetivo de auxiliar estudantes universitários no gerenciamento de seu progresso acadêmico. A ferramenta permite ao usuário carregar a estrutura curricular de um curso (PPC - Projeto Pedagógico de Curso) a partir de um arquivo de dados, visualizar a grade de disciplinas de forma interativa e acompanhar as matérias já concluídas.

O sistema valida automaticamente os pré-requisitos, oferecendo um feedback visual claro sobre quais disciplinas estão disponíveis para serem cursadas. Adicionalmente, fornece relatórios de progresso, gráficos e a capacidade de salvar e carregar o avanço do aluno, tornando-se uma ferramenta valiosa para o planejamento acadêmico e a tomada de decisões.

## 2. Escopo do Projeto

O escopo do projeto está focado em fornecer uma solução local e de uso individual para o acompanhamento da integralização curricular.

**Dentro do Escopo:**

- Carregamento de um único arquivo de PPC (em formato XML) por vez.
- Acompanhamento do progresso de um único aluno por vez.
- Validação de pré-requisitos com base nos dados do PPC carregado.
- Geração de relatórios e gráficos baseados no estado atual da interface.
- Persistência do progresso do aluno em arquivos de texto locais.

**Fora do Escopo:**

- Integração com sistemas de gestão acadêmica de universidades (APIs).
- Suporte a múltiplos usuários ou perfis de alunos simultaneamente.
- Conexão com bancos de dados para armazenamento de dados.
- Gerenciamento de notas, faltas ou aspectos financeiros.

---

## 3. Requisitos do Sistema

### 3.1. Requisitos Funcionais (RF)

- **RF01: Carregar Estrutura do Curso (PPC)**: O sistema deve ser capaz de carregar e interpretar a estrutura curricular de um curso a partir de um arquivo no formato XML.
- **RF02: Exibir Grade Curricular**: O sistema deve exibir a grade curricular completa em uma tabela, contendo as colunas: Semestre, Ordem, Disciplina, Carga Horária, Créditos e Pré-Requisitos.
- **RF03: Marcar Progresso do Aluno**: O usuário deve poder marcar e desmarcar qualquer disciplina como "Cursada" através de um checkbox na tabela.
- **RF04: Validação de Pré-Requisitos**: O sistema deve validar os pré-requisitos em tempo real. Uma disciplina só pode ser marcada como "Cursada" se todas as suas disciplinas de pré-requisito já estiverem marcadas.
- **RF05: Feedback Visual de Status**: O sistema deve indicar visualmente o estado de cada disciplina na tabela, utilizando cores distintas para: disponíveis, bloqueadas e já cursadas.
- **RF06: Diferenciação de Tipos de Disciplina**: O sistema deve distinguir visualmente as disciplinas obrigatórias das CCCG, exibindo separadores na tabela.
- **RF07: Exibir Relatório de Progresso**: O sistema deve calcular e exibir um resumo do progresso do aluno, incluindo a carga horária total, a carga horária concluída e o percentual de conclusão.
- **RF08: Gerar Gráfico de Progresso**: O sistema deve gerar um gráfico do tipo "pizza" que represente visualmente a proporção de disciplinas cursadas versus não cursadas.
- **RF09: Gerar Relatório de Disciplinas Disponíveis**: O sistema deve permitir ao usuário gerar um relatório em arquivo `.txt` que liste as disciplinas disponíveis para cursar, incluindo o nome do aluno e do curso.
- **RF10: Salvar Progresso Atual**: O usuário deve poder salvar o estado atual de seu progresso (disciplinas cursadas) em um arquivo de texto.
- **RF11: Carregar Progresso Salvo**: O usuário deve poder carregar um arquivo de progresso salvo para restaurar o estado das disciplinas na tabela.
- **RF12: Tratamento de Erros de Arquivo**: Em caso de tentativa de carregamento de um arquivo XML com formato inválido, o sistema deve exibir uma mensagem de erro clara ao invés de travar.

### 3.2. Requisitos Não Funcionais (RNF)

- **RNF01: Interface Gráfica do Usuário (GUI)**: O sistema deve possuir uma interface gráfica de desktop intuitiva e de fácil utilização.
- **RNF02: Portabilidade**: A aplicação deve ser executável em qualquer sistema operacional (Windows, macOS, Linux) que possua uma Java Runtime Environment (JRE) compatível.
- **RNF03: Formato de Dados de Entrada (Restrição)**: O sistema deve ser estritamente compatível com o formato XML pré-definido para o PPC.
- **RNF04: Formato de Dados de Persistência**: Os arquivos de salvamento e relatórios devem ser em formato de texto (`.txt`), para legibilidade e portabilidade.
- **RNF05: Desempenho**: A interface deve responder rapidamente às ações do usuário, com atualização visual instantânea.
- **RNF06: Manutenibilidade**: O código-fonte deve ser organizado em classes com responsabilidades bem definidas para facilitar futuras manutenções.

---

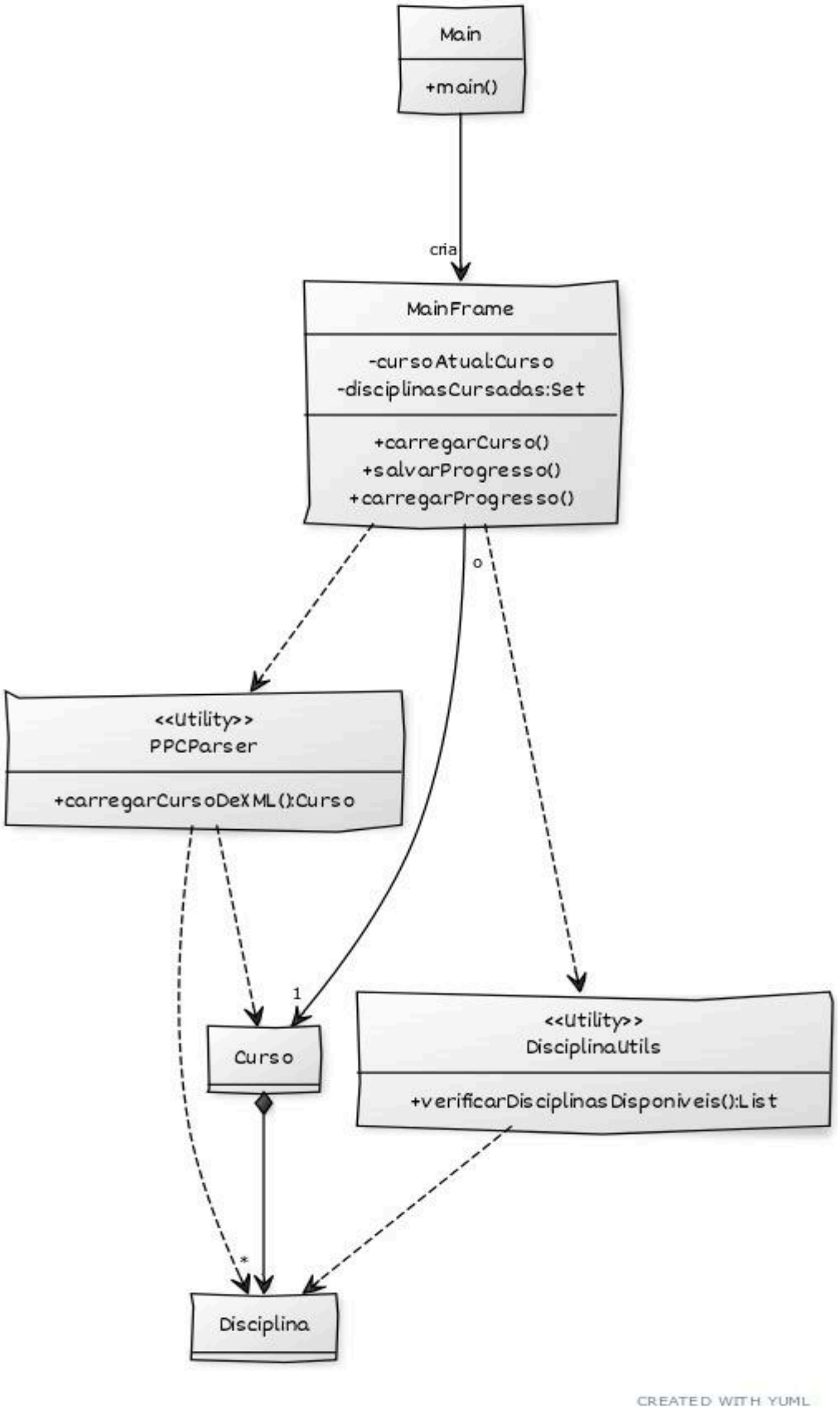
## 4. Arquitetura do Sistema

O projeto segue uma arquitetura com separação de responsabilidades, onde classes de interface, modelos de dados e lógica de negócio são mantidas em componentes distintos.

### 4.1. Descrição das Classes Principais

- **`Main`**: Ponto de entrada da aplicação, responsável por iniciar a interface gráfica na thread correta.
- **`MainFrame`**: Classe principal que gerencia a janela, a tabela, os botões e todos os eventos de interação do usuário. Orquestra a comunicação entre as demais partes do sistema.
- **`Curso`** **e** **`Disciplina`**: Classes de modelo (POJOs) que representam as entidades do domínio. `Disciplina` armazena os dados de uma matéria e `Curso` armazena os dados do curso e uma lista de disciplinas.
- **`PPCParser`**: Classe utilitária responsável por ler e interpretar o arquivo `ppc.xml`, traduzindo seus dados para os objetos Java que a aplicação utiliza.
- **`DisciplinaUtils`**: Classe utilitária que contém a lógica de negócio principal, como o algoritmo para verificar quais disciplinas estão disponíveis com base nos pré-requisitos cumpridos.

### 4.2. Diagrama de Classes (UML)



---

## 5. Guia Rápido de Utilização

1. **Iniciar a Aplicação**: Execute o programa. A tela inicial de boas-vindas será exibida.
2. **Carregar PPC**: Clique no botão "Carregar PPC do Curso" e selecione o arquivo `ppc.xml` correspondente ao seu curso. A tabela de disciplinas será preenchida.
3. **Marcar Progresso**: Marque os checkboxes na coluna "Cursada?" para as disciplinas que você já concluiu. O sistema atualizará as cores da tabela em tempo real.
4. **Analisar Progresso**: Utilize os botões na parte inferior da tela para:
    - **Ver Progresso**: Exibe um resumo da sua carga horária.
    - **Ver Gráfico**: Mostra um gráfico de pizza do seu avanço.
    - **Ver Disciplinas Disponíveis**: Lista as matérias que você pode cursar no próximo semestre.
5. **Salvar/Carregar**:
    - Use **"Salvar Progresso"** para guardar seu estado atual em um arquivo.
    - Use **"Carregar Progresso"** (após carregar um PPC) para restaurar um estado salvo anteriormente.
6. **Exportar Relatório**: Clique em **"Salvar Relatório"** para gerar um arquivo `.txt` com as disciplinas disponíveis, ideal para planejamento de matrícula.

---

## 6. Tecnologias Utilizadas

- **Linguagem de Programação:** Java
- **Framework de Interface Gráfica:** Java Swing
- **Ferramenta de Build:** Apache Maven
- **Bibliotecas Externas:**
    - `JFreeChart`: Para a geração de gráficos.

## 7. Conclusão e Trabalhos Futuros

O projeto atingiu com sucesso seu objetivo de criar uma ferramenta funcional para o gerenciamento da integralização curricular. A aplicação cumpre todos os requisitos propostos, oferecendo uma interface clara e funcionalidades úteis para o planejamento acadêmico.

Como possíveis melhorias e trabalhos futuros, pode-se considerar:

- Integração com um banco de dados local (como SQLite) para gerenciar múltiplos perfis de alunos e cursos.
- Desenvolvimento de uma versão web da aplicação para acesso universal.
- Implementação de um sistema de sugestão de grade horária para o próximo semestre.
- Flexibilização da leitura de dados para aceitar outros formatos além do XML (como JSON ou CSV).

## 8. Referências Bibliográficas


1. **PRESSMAN, Roger S.** *Engenharia de Software: Uma Abordagem Profissional.* 8ª Edição. Porto Alegre: AMGH, 2016.
2. **DEITEL, P. J.; DEITEL, H. M.** *Java: Como Programar.* 10ª Edição. São Paulo: Pearson Prentice Hall, 2016.
3. **MARTIN, Robert C.** *Código Limpo: Habilidades Práticas do Agile Software.* Rio de Janeiro: Alta Books, 2009.
4. **ORACLE CORPORATION.** *The Java™ Tutorials: Creating a GUI With Swing.* Disponível em: https://docs.oracle.com/javase/tutorial/uiswing/. Acesso em: 09 de julho de 2025.
5. **GILBERT, David.** *JFreeChart: a free Java chart library.* JFree.org. Disponível em: https://www.jfree.org/jfreechart/. Acesso em: 09 de julho de 2025.
6. **WORLD WIDE WEB CONSORTIUM (W3C).** *Extensible Markup Language (XML) 1.0 (Fifth Edition).* Disponível em: https://www.w3.org/TR/xml/. Acesso em: 09 de julho de 2025.
7. **APACHE SOFTWARE FOUNDATION.** *Apache Maven Project.* Disponível em: https://maven.apache.org/. Acesso em: 09 de julho de 2025.