### 

### **Manual Técnico de Implementação**



**Grupo:** 7

**Autores:** Maria Spínola & Ricardo Reis

**Curso:** Programação em Java

[1. Introdução 3](#_Toc29501)

[Ambito deste Manual Técnico CedricArt: 3](#_Toc28039)

[Arquitetura do Sistema: 3](#_Toc22232)

[Diagrama de Classes e de Pacotes: 3](#_Toc23229)

[Bibliotecas Utilizadas: 4](#_Toc21655)

[Documentação de Classes e Métodos: 4](#_Toc25799)

[Procedimento para Geração do Javadoc: 4](#_Toc2284)

[Considerações Técnicas: 5](#_Toc30431)

[Resultados Alcançados e Requisitos não Implementados: 5](#_Toc22683)

[2. Configuração do Ambiente de Desenvolvimento 6](#_Toc6851)

[Configuração do Projeto Java: 6](#_Toc14464)

[Base de Dados: 7](#_Toc16089)

[Configurar REST API Artsy: 8](#_Toc10382)

[3. Estrutura Geral do Projeto 8](#_Toc25313)

[A arquitetura do Sistema está dividida em 4 módulos: 8](#_Toc17542)

[1. Camada de Apresentação (Front-End) 9](#_Toc18671)

[2. Camada de Negócios (Back-End) 9](#_Toc28186)

[Modelo de Domínio: 9](#_Toc7191)

[3. Camada de Persistência 10](#_Toc23441)

[4. Integração e Middleware 10](#_Toc6914)

[5. Testes e Manutenção 10](#_Toc5252)

[6. Infraestrutura e Deployment 10](#_Toc11305)

[4. Implementação do Backend (módulos appimport e appserver) 11](#_Toc5546)

[Implementação da REST API Aplicacional (módulo appserver): 11](#_Toc20743)

[Requisições à REST API Artsy e à REST API Aplicacional (módulo appimport): 11](#_Toc23492)

[Camada de Dados: 14](#_Toc22463)

[5. Integração com a API Artsy 14](#_Toc22313)

[Consumo da API: 14](#_Toc26376)

[Mapeamento dos Dados: 14](#_Toc5256)

[Importação de Dados 15](#_Toc7772)

[6. Implementação do Frontend (módulo appclient) 15](#_Toc3285)

[Conexão com a API REST: 15](#_Toc5308)

[Interface do Utilizador: 15](#_Toc23932)

[Integração com Recursos Visuais: 16](#_Toc4751)

[7. Testes 16](#_Toc29518)

[Testes Unitários (falta completar): 16](#_Toc1883)

[Testes de Integração (falta completar): 16](#_Toc17865)

[Testes de Sistema (falta implementar): 17](#_Toc621)

[Testes de Desempenho (falta implementar): 17](#_Toc28072)

[Testes de Segurança (falta implementar): 17](#_Toc28080)

[Testes de Usabilidade (falta implementar): 17](#_Toc30939)

[Automação de Testes (por implementar): 18](#_Toc1086)

[8. Implementação 18](#_Toc12121)

[Configuração do Servidor: 18](#_Toc2672)

[Processo de Deploy: 19](#_Toc22376)

[Monitoramento e Manutenção: 19](#_Toc15223)

[9. Manutenção e Escalabilidade 20](#_Toc15701)

[Monitoramento do Sistema: 20](#_Toc16586)

[Atualizações e Escalabilidade: 20](#_Toc8622)

[Procedimentos de Manutenção: 21](#_Toc27113)

[10. Conclusão 21](#_Toc2679)

[Sumário do Projeto: 22](#_Toc20287)

[Próximos Passos: 22](#_Toc5884)

[Apêndices 22](#_Toc8503)

[Referências: 23](#_Toc11444)

[Glossário: 23](#_Toc11891)

# 1. Introdução

O CedricArt é um sistema dedicado à gestão de galerias de arte, exposições e parcerias no contexto do mundo artístico. Seu principal objetivo é oferecer uma plataforma que permita aos utilizadores (visitantes, gestores e administradores de galerias e outros parceiros), uma gestão eficientemente as informações relacionadas a galerias, artistas, obras de arte e exposições.

## Ambito deste Manual Técnico CedricArt:

Este manual técnico destina-se a programadores e analistas envolvidos no projeto CedricArt. O documento aborda detalhes específicos relacionados à implementação do sistema, análise de código, e informações relevantes para aqueles que precisam entender, analisar, estender ou corrigir o programa. Os principais tópicos abordados incluem:

### Arquitetura do Sistema:

* + Descrição da arquitetura geral do CedricArt
  + Detalhes sobre os componentes principais e sua interação

### Diagrama de Classes e de Pacotes:

* + Apresentação do diagrama de classes, destacando as principais classes do sistema e suas relações
  + Diagrama de pacotes para fornecer uma visão de alto nível da estrutura do código

## Bibliotecas Utilizadas:

* + Lista das bibliotecas e frameworks utilizados no desenvolvimento do CedricArt.
  + Explicação sobre como essas bibliotecas contribuem para a funcionalidade geral do sistema.

## Documentação de Classes e Métodos:

* + Informações detalhadas sobre as classes principais, métodos e suas funcionalidades.
  + Exemplos de uso e considerações especiais.

## Procedimento para Geração do Javadoc:

* + Instruções passo a passo sobre como gerar a documentação do código-fonte usando o Javadoc no IntelliJ IDEA.

## 

## Considerações Técnicas:

* + Detalhes sobre escolhas técnicas específicas, padrões de codificação e boas práticas seguidas no desenvolvimento.

## Resultados Alcançados e Requisitos não Implementados:

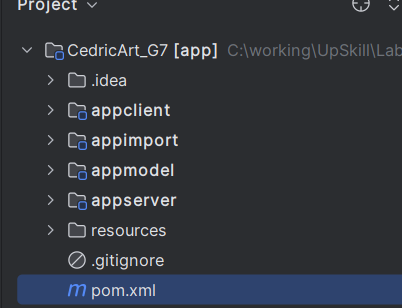
* + Avaliação dos resultados alcançados em relação aos objetivos do projeto.
  + Identificação de requisitos que não foram completamente implementados.

O objetivo é fornecer uma visão abrangente e técnica do sistema CedricArt, permitindo que os leitores compreendam a estrutura, funcionamento interno e possíveis áreas de melhoria. Este manual visa facilitar a manutenção, extensão e compreensão profunda do código-fonte para aqueles envolvidos no desenvolvimento e análise do sistema.

# 2. Configuração do Ambiente de Desenvolvimento

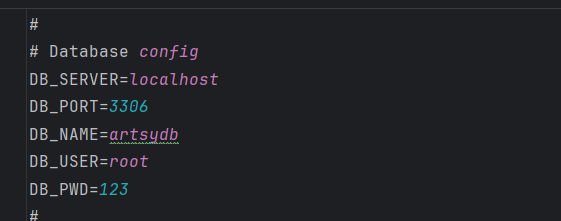
* Ferramentas Necessárias: IntelliJ, Java JDK, MySQL Workbench, Git para controle de versões
* Configuração Inicial: Passo a passo para configurar o ambiente de desenvolvimento:

## Configuração do Projeto Java:

* Utilizar Maven para gerir dependências, compilar e construir
* Estrutura de diretórios bem definida para organização do código-fonte.
* Criar um projecto IntelliJ com os seguintes módulos:
  + 

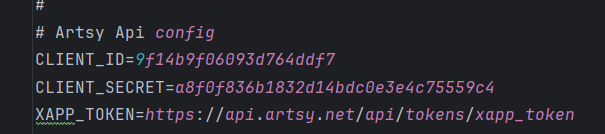
## Base de Dados:

* Instalar um programa gestor de Base de Dados MySql: por exemplo Workbench
* Configurar a conexão à base de dados (tanto no programa gestor de Base de Dados, mencionado no ponto anterior), assim como no ficheiro “CedricArt.config” (que encontra na directoria “resources\config”)



## Configurar REST API Artsy:

* Criar uma aplicaçao em <https://developers.artsy.net/client_applications>, e alterar para os dados “corretos”, no ficheiro “CedricArt.config” (que encontra na directoria “resources\config”)



# 3. Estrutura Geral do Projeto

## A arquitetura do Sistema está dividida em 4 módulos:

* “**appclient**”: Camada de Apresentação
* “**appmodel**”:Camada de modelo (classes das “entidades” partilhadas entre todos os módulos
* “**appimport**”: Camada de Negócio com a responsabilidade de importação de dados da REST Api Artsy para a Base de Dados Local, através da REST Api aplicacional (que se encontra no módulo “appserver”.
* “**Appserver**”: Camada de Negócio com a responsabilidade de ofererer um serviço REST API, usada pelas camadas “appclient” e “appimport” para a gestão e persistência dos dados na base de dados

## 1. Camada de Apresentação (Front-End)

**Interface de Utilizador(UI):** Desenvolvida em JavaFX para aplicação desktop (“appcliente”)

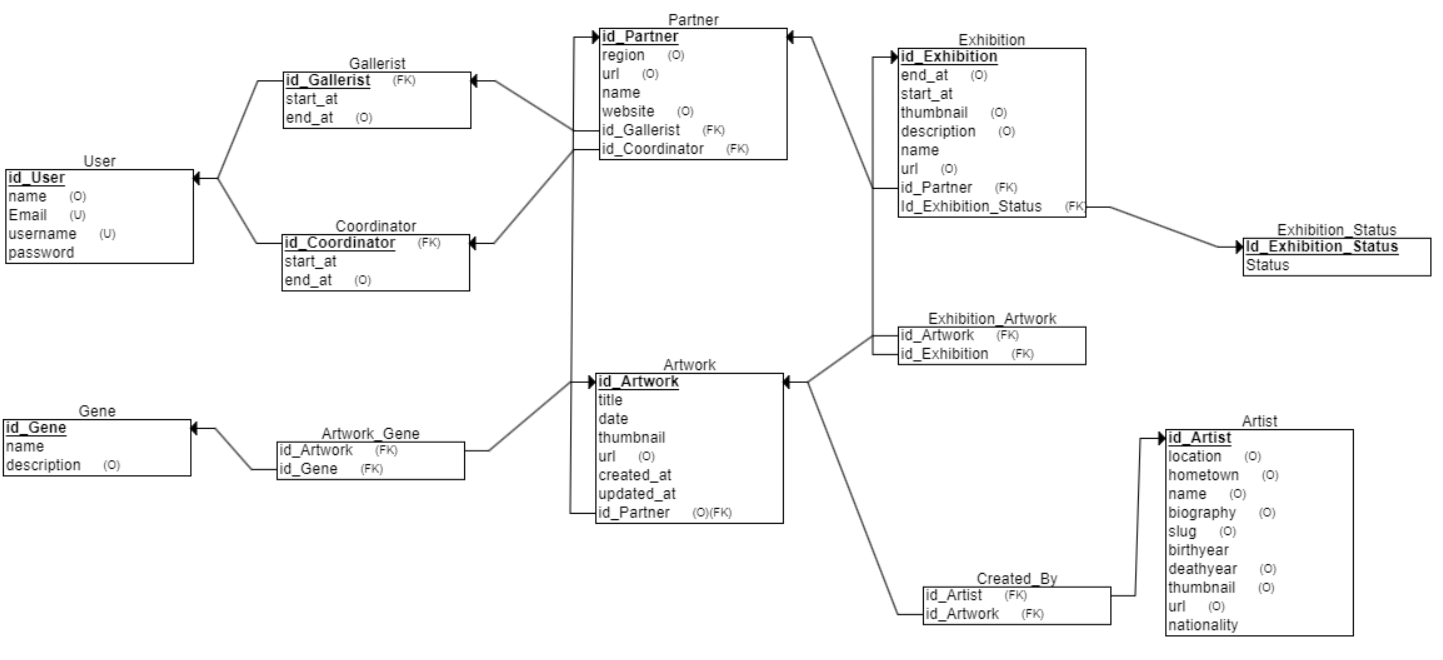
**Navegação e Fluxo de Trabalho:** O módulo “appclient” controla como os utilizadores interagem com a aplicação, navegando entre diferentes funções como visualização de galerias, obras de arte, artistas, etc.

Esse módulo está implementado segundo uma arquitetura MVP (as classes do “Modelo” estão no módulo “appmodel”

## 2. Camada de Negócios (Back-End)

**Serviços de Negócio:** Incluem lógica de negócios específica para a gestão de galerias, artistas, obras de arte, géneros artísticos e exposições. (módulos “appimport” e “appserver”)

## Modelo de Domínio:



**Serviços de Apoio:** Serviços auxiliares como autenticação, autorização, logging, etc. não foram implementados, nesta versão.

## 3. Camada de Persistência

**Base de Dados:** Um sistema de gestão de base de dados MySQL, onde os dados da aplicação são armazenados.

### 

## 4. Integração e Middleware

**API Server:**  Utilização de uma REST API aplicacional para a comunicação entre os dados importados da REST API Artsy “appimport”, a camada de “appmodelo” e a Base de Dados “appserver”. Assim como a comunicação entre os dados da camada “appclient” e a Base de Dados “appserver”)

## 5. Testes e Manutenção

**Testes** de Integração: Uso de frameworks como JUnit para garantir a qualidade e o funcionamento correto da aplicação.

**Manutenção e Atualização:** Processos para atualização contínua da aplicação e correção de bugs, através do Git

## 

## 6. Infraestrutura e Deployment

Servidores: Dado que é uma aplicação de desktop, mas que necessita de acesso à Internet, é necessário configurar servidores web, aplicacionais(REST API aplicacional) e Base de Dados

# 4. Implementação do Backend (módulos appimport e appserver)

O módulo appimport (tal como mencionado anteriormente) tem como principal responsabilidade a importação dos dados da REST API Artsy para a Base de Dados das galerias CedricArt.

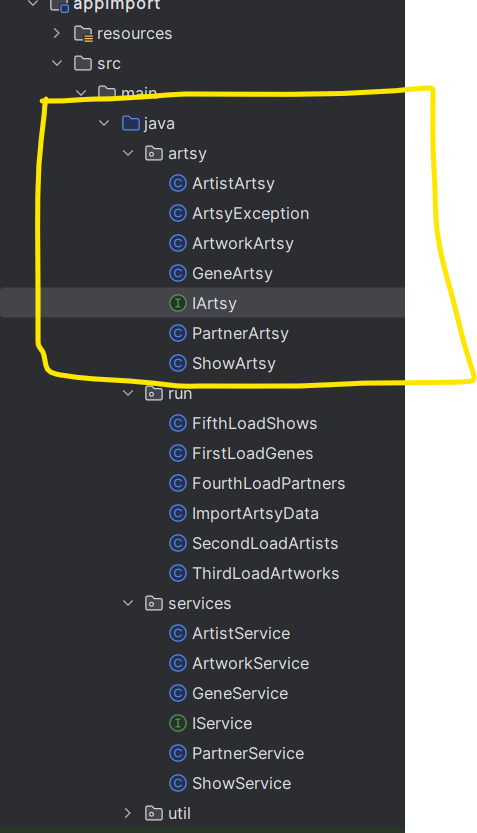
O módulo appserver (e também como mencionado anteriormente) tem como principal responsabilidade oferecer endpoints para as operações CRUD (Create, Read, Update e Delete) sobre as entidades do modelo do domínio.

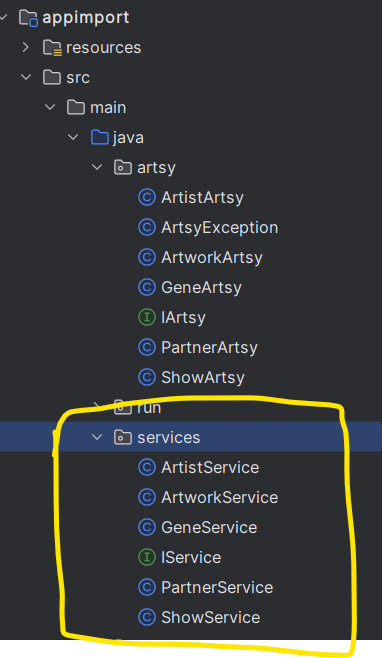
## Implementação da REST API Aplicacional (módulo appserver):

* Implementação baseada em Java Spark (ver class server.RunAPIServer.java)

## Requisições à REST API Artsy e à REST API Aplicacional (módulo appimport):

* Consumo dos endpoints oferecidos pela REST API Artsy com recurso à biblioteca GSON (ver package artsy)



* Consumo dos endpoints oferecidos pela REST API Aplicacional com recurso à biblioteca GSON (ver package services)
* 

NOTA: Neste módulo (appimport) optamos por não usar o conceito de Generics de Java, ao contário do que foi implementado no módulo appclient (ver class presenter.GenericPresenter<T> no módulo appclient).

A grande desvantagem de não termos usados Generics (no módulo appimport) foi a duplicação de código e lógica nas várias classes (GeneArtsy, ArtistArtsy, ArworkArsty, PartnerArsty e ShowArtsy, assim como às classes respondentes: ArtistService,etc.)

Logo, aconselhamos que seja aplicado o conceito de Generics de Java neste módulo, assim como foi aplicado no módulo appclient.

## Camada de Dados:

* Persistência de dados através de JDBC (ver class persistence.DBStorage.java)
* Utilização classes DAO (módulo “appmodel”) para interação com a REST API Artsy, a REST API aplicacional e a base de dados

# 5. Integração com a API Artsy

Abaixo estão os principais aspectos relacionados à integração com a API Artsy:

## Consumo da API:

* Utilização de bibliotecas em Java para realizar requisições HTTP à API Artsy.
* Implementação de métodos ou serviços dedicados para consumir endpoints específicos da API.

## Mapeamento dos Dados:

* Transformação dos dados recebidos da API Artsy para o modelo de dados CedricArt. (ver módulo appmodel)
* Ajuste de formatos e estruturas para garantir consistência e compatibilidade com o sistema.

## Importação de Dados

Essa integração permite que o CedricArt mantenha seu banco de dados atualizado com informações relevantes da API Artsy, oferecendo aos usuários uma experiência enriquecedora e informações precisas sobre o mundo da arte. O agendamento regular garante que os dados estejam sempre atualizados e reflete as mudanças dinâmicas na plataforma Artsy.

# 6. Implementação do Frontend (módulo appclient)

A implementação do frontend no CedricArt visa oferecer uma experiência intuitiva e amigável para os usuários, tanto para os gestores de galerias quanto para os visitantes. Abaixo estão os principais aspectos relacionados à implementação do frontend:

## Conexão com a API REST:

* Utilização de métodos HTTP (GET, POST, PUT, DELETE) para interagir com os serviços disponibilizados pelo backend.
* Manipulação dos dados recebidos da API para apresentação na interface do utiilizador.

## Interface do Utilizador:

* Desenvolvimento de interfaces intuitivas para os visitantes das galerias permitindo a consulta de galerias, exposições, obras de arte e artistasxposições, obras de arte e informações relacionadas.

## Integração com Recursos Visuais:

* Incorporação de recursos visuais, como imagens de obras de arte e logotipos de galerias, para tornar a interface mais atraente e informativa.

# 7. Testes

A realização de testes é crucial para garantir a qualidade, confiabilidade e desempenho do sistema CedricArt. Abaixo estão os principais tipos de testes (a maioria ainda não estão implementados nesta versão):

## Testes Unitários (falta completar):

* Utilização de frameworks como JUnit para testar unidades individuais de código, como métodos e classes.
* Verificação de casos esperados e inesperados para garantir que cada parte do código funcione conforme o esperado.

## Testes de Integração (falta completar):

* Foco na verificação da interação harmoniosa entre diferentes partes do sistema.
* Testes para garantir que os componentes individuais se integrem corretamente e que a comunicação entre eles ocorra sem problemas.

## Testes de Sistema (falta implementar):

* Avaliação do sistema como um todo para verificar se atende aos requisitos especificados.
* Verificação de cenários de uso típicos e excepcionais para garantir que o sistema seja robusto e confiável.

## Testes de Desempenho (falta implementar):

* Avaliação do desempenho do sistema sob carga para garantir que ele possa lidar com um número significativo de transações (especialmente no módulo appimport, dado o volume de dados a importar a API Artsy)
* Identificação e otimização de possíveis gargalos de desempenho.

## Testes de Segurança (falta implementar):

* Avaliação da segurança do sistema para identificar e corrigir possíveis vulnerabilidades.
* Testes de penetração para simular ataques e garantir a resistência do sistema a ameaças.

## Testes de Usabilidade (falta implementar):

* Realização de testes com utilizadores reais para avaliar a usabilidade da interface do utilizador.
* Identificação de áreas de melhoria na experiência do utilizador.

## Automação de Testes (por implementar):

* Automação de testes sempre que possível para facilitar a execução contínua e garantir uma rápida detecção de problemas.
* Uso de ferramentas adequadas para automação de testes unitários e de integração.

A implementação abrangente de testes é essencial para assegurar a estabilidade e a qualidade do CedricArt, proporcionando uma experiência positiva aos utilizadoresfinais e facilitando a manutenção contínua do sistema.

NOTA: A

# 8. Implementação

A implementação do CedricArt envolve alguns passos cruciais, incluindo a configuração do servidor e o processo de deploy para garantir que o sistema esteja pronto para ser utilizado em um ambiente de produção. Abaixo estão os principais pontos relacionados a essa etapa:

## Configuração do Servidor:

* Escolha do servidor: O CedricArt pode ser implantado em servidores. A escolha do servidor dependerá dos requisitos específicos do projeto e da infraestrutura disponível.

## Processo de Deploy:

* Empacotamento do Aplicativo: Utilização de ferramentas como Maven ou Gradle para empacotar o CedricArt em um formato adequado para deploy.
* Configuração do Ambiente: Preparação do ambiente de produção, incluindo a instalação do servidor escolhido e configurações de base de dados, assim como a configuração das propriedades no ficheiro
* Base de Dados: Configuração da conexão com a base de dados, garantindo que o sistema tenha acesso aos dados necessários.
* Arquivos Estáticos: Gerenciamento de arquivos estáticos, como imagens, para garantir que sejam acessíveis e carregados corretamente.
* Segurança: Implementação de medidas de segurança adequadas, como configuração de firewalls e certificados SSL, para proteger o sistema contra ameaças.

## Monitoramento e Manutenção:

* Implementação de ferramentas de monitoramento para rastrear o desempenho do sistema em tempo real.
* Estabelecimento de procedimentos de manutenção regular para garantir a integridade e a disponibilidade contínua do CedricArt.

A correta execução desses passos garantirá uma implementação suave e eficiente do CedricArt em um ambiente de produção, proporcionando aos usuários uma experiência confiável e segura.

# 9. Manutenção e Escalabilidade

A manutenção e escalabilidade são aspectos cruciais para garantir que o CedricArt continue funcionando de maneira eficiente e atenda às demandas crescentes. Abaixo estão os pontos relevantes para essas áreas:

## Monitoramento do Sistema:

* Ferramentas de Monitoramento: Implementação de ferramentas de monitoramento para acompanhar o desempenho do CedricArt em tempo real. Ferramentas como Prometheus, Grafana ou soluções específicas da plataforma podem ser úteis.
* Logs do Sistema: Estabelecimento de registros detalhados para rastrear eventos, identificar possíveis problemas e facilitar a resolução de questões de desempenho.

## Atualizações e Escalabilidade:

* Atualizações de Software: Implementação de um processo eficiente de atualização de software para garantir que o CedricArt esteja sempre utilizando as versões mais recentes, corrigindo possíveis vulnerabilidades e bugs.
* Escalabilidade Horizontal: Projeto do CedricArt de forma a possibilitar a escalabilidade horizontal, permitindo adicionar mais instâncias do aplicativo conforme a procura aumenta.
* Bancos de Dados: Utilização de estratégias de escalabilidade para o banco de dados, como particionamento de tabelas, replicação e balanceamento de carga, para garantir o desempenho contínuo.

## Procedimentos de Manutenção:

* Backup Regular: Implementação de rotinas de backup regulares para proteger os dados contra perdas acidentais.
* Testes de Recuperação: Realização de testes regulares de recuperação para garantir a eficácia dos procedimentos de backup.
* Ajustes de Desempenho: Monitoramento contínuo do desempenho do sistema e ajustes conforme necessário para otimizar a eficiência.

Ao abordar esses pontos, o CedricArt estará preparado para enfrentar os desafios de manutenção e escalabilidade, garantindo uma operação suave e confiável ao longo do tempo.

# 

# 10. Conclusão

A conclusão do projeto CedricArt marca o encerramento da fase inicial de desenvolvimento, implementação e documentação. Aqui estão os elementos importantes a serem considerados:

## Sumário do Projeto:

* Recapitulação dos Objetivos: Revisão dos objetivos iniciais do CedricArt, incluindo a criação de uma plataforma para gerenciamento de galerias de arte, exposições, artistas e obras de arte.
* Principais Conquistas: Destaque das principais conquistas alcançadas durante o desenvolvimento, como a integração com a API Artsy, a implementação do frontend, a segurança do sistema e outros marcos significativos.

## Próximos Passos:

* Direções Futuras: Identificação de áreas para melhorias futuras e expansões. Isso pode incluir a adição de novos recursos, aprimoramentos na interface do utilizador, otimizações de desempenho e integração com novas tecnologias.
* Feedback do Utilizador: Consideração do feedback do utilizador para orientar melhorias contínuas. A interação com utilizadoresreais pode revelar oportunidades de aprimoramento não previstas durante o desenvolvimento inicial.

# Apêndices

* Manual do Utilizador CedricArt\_G7.docx
* Ficheiro compactado com o código fonte e JavaDoc, assim como ficheiro de configurações das propriedades (“CedricArt.config) e os scripts da base de dados que estão na directoria “Resources”.
* Nota: Todos estes recursos também estão disponíveis em https://github.com/formandoUpskill/CedricArt\_G7

## Referências:

* REST API Artsy - Documentação
* JUnit 5 - User Guide

## Glossário:

* API: Interface de Programação de Aplicações. Conjunto de definições e protocolos que permite a comunicação entre sistemas.
* Frontend: Parte de um sistema que interage diretamente com o utilizador.
* Backend: Parte de um sistema que processa dados, realiza operações e interage com o base de dados, sem interface direta com o usuário.
* REST API: Arquitetura de transferência de estado representacional, usada para comunicação entre sistemas via HTTP.
* Maven: Sistemas de automação de compilação e gerenciamento de dependências em projetos Java.
* JUnit: Framework de testes unitários para a linguagem de programação Java.

Observação: Certifique-se de consultar a documentação oficial para obter informações mais detalhadas sobre cada tecnologia e conceito mencionado neste manual técnico do CedricArt.