FATESG Senai
Análise e Desenvolvimento de Sistemas
Grupo com Integrantes Projeto Integrador

AGENDA SWING: EVOLUÇÃO ARQUITETURAL DIDÁTICA

Documento de Arquitetura de Software

[ADS3] – [Agenda Swing: Evolução Arquitetural Didática]
Integrantes
Alexander Nogueira
Caio Nunes de Abreu
Cassiano Nunes de Abreu
Gabriel Naoki Uto Turigoe
Wyllian Mariano Nogueira

Objetivo deste Documento

Este documento tem como objetivo descrever as principais decisões de projeto tomadas durante o desenvolvimento e os critérios considerados para a transição arquitetural. Suas informações detalham os requisitos funcionais, a organização das camadas de software (UI, Business e Data) e as especificações de hardware necessárias para a execução do sistema.

Histórico de Revisão

Data	Demanda	Autor	Descrição	Versão
[11/02/2026]	Projeto Acadêmico	Todos Integrantes	Criação inicial do documento, definição de escopo e objetivos do sistema.	1.0

Sumário

1. INTRODUÇÃO

1.1 Finalidade

Este documento tem como finalidade definir os requisitos e as especificações técnicas para o desenvolvimento de um sistema de **Agenda Pessoal**. O foco primordial deste artefato não é apenas a descrição das funcionalidades do software, mas a documentação da transição de um modelo de código funcional simples para uma **Arquitetura em Camadas (Layered Architecture)** de nível profissional.

O objetivo é garantir que o sistema seja construído com uma separação clara entre a interface do usuário (desenvolvida em **Java Swing**), a lógica de negócios e a camada de persistência de dados. Esta abordagem visa assegurar a escalabilidade, a facilidade de manutenção e a demonstração de competências avançadas em **Engenharia de Software** aplicadas ao ecossistema Java.

O documento adotará uma estrutura baseada na visão “4+1” de modelo de arquitetura [KRU41].

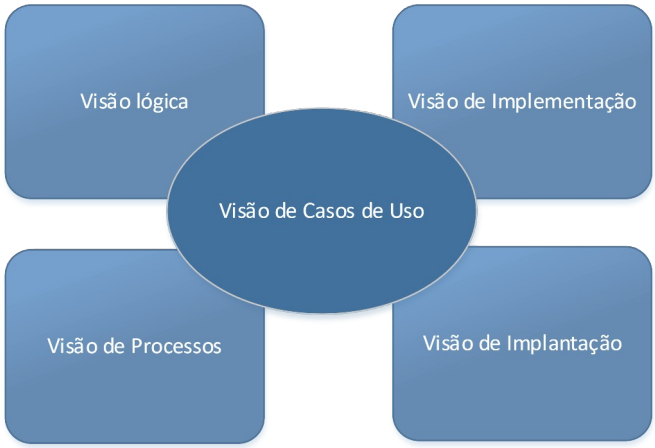

Figura 1 – Arquitetura 4+1

1.2 Escopo

O escopo deste projeto abrange o desenvolvimento de uma aplicação desktop para gerenciamento de contatos pessoais, estruturada para demonstrar a aplicação prática de padrões de projeto e separação de responsabilidades.

Abrangência (O que será desenvolvido):

- **Interface Gráfica (GUI):** Telas intuitivas desenvolvidas em Java Swing para cadastro, consulta, edição e exclusão de contatos.
- **Camada de Negócio (Business Layer):** Implementação de regras de validação e processamento de dados independentes da interface.
- **Camada de Persistência (Data Access Object – DAO):** Estrutura para comunicação com banco de dados, permitindo que as informações sejam salvas e recuperadas permanentemente.
- **Refatoração Arquitetural:** O projeto prevê a transição documentada de um código monolítico para a Arquitetura em Camadas.

1.3 Definições, Acrônimos e Abreviações

Esta subseção fornece as definições de todos os termos, acrônimos e abreviações utilizados neste documento, facilitando a interpretação técnica do projeto.

- **ARS:** Artefato de Requisitos de Software. Documento que descreve as funcionalidades e restrições de um sistema.
- **Java Swing:** API (Application Programming Interface) do Java utilizada para a criação de interfaces gráficas (GUI) em aplicações desktop.
- **Arquitetura em Camadas (Layered Architecture):** Padrão de arquitetura de software que organiza o sistema em camadas horizontais, onde cada camada possui uma responsabilidade específica (ex: Interface, Negócio e Dados).
- **DAO (Data Access Object):** Padrão de projeto utilizado para abstrair e isolar o acesso à base de dados, permitindo que a lógica de negócio não dependa diretamente de tecnologias de armazenamento.
- **GUI (Graphic User Interface):** Interface Gráfica do Utilizador, que permite a interação com o sistema através de elementos visuais como botões, menus e janelas.
- **JDK (Java Development Kit):** Conjunto de ferramentas necessárias para o desenvolvimento de aplicações Java.
- **CRUD:** Acrónimo para *Create, Read, Update and Delete*. Representa as quatro operações básicas de manipulação de dados em um sistema.

1.4 Referências

[KRU41]: The “4+1” view model of software architecture, Philippe Kruchten, November 1995, http://www3.software.ibm.com/ibmdl/pub/software/rational/web/whitepapers/2003/Pbk4p1.pdf

2. REPRESENTAÇÃO ARQUITETURAL

Este documento detalhará as visões baseado no modelo “4+1” [KRU41], utilizando como referência os modelos definidos na MDS. As visões utilizadas no documento serão:

Lógica	Analistas	Realização dos Casos de Uso	Detalhamento da separação das classes em camadas (**UI, Service/Business e DAO**).
Processo	Integradores	Performance, Escalabilidade, Concorrência	Fluxo de processamento de um contato, desde a entrada na interface até a persistência.
Implementação	Programadores	Componentes de Software	Organização dos pacotes Java e gerenciamento de dependências via **Maven**.
Implantação	Gerência de Configuração	Nodos físicos	Especificação do ambiente de execução (**JRE**) e o banco de dados local (**SQLite/MySQL**).
Caso de Uso	Todos	Requisitos funcionais	Representação das interações do usuário, como Cadastrar, Editar e Excluir contatos.
Dados	Especialistas em dados Administradores de dados	Persistência de dados	Modelo de entidade e relacionamento para o armazenamento da agenda.

Tabela 1 – Visões, Público, Área e Artefatos da MDS

3. REQUISITOS E RESTRIÇÕES ARQUITETURAIS

Esta seção descrever os requisitos de software e restrições que tem um impacto significante na arquitetura.

Requisito	Solução
Linguagem	**Java (JDK 17 ou superior).** A escolha do JDK 17 garante estabilidade e suporte a recursos modernos de POO (Programação Orientada a Objetos).
Plataforma	**Java Virtual Machine (JVM).** Como aplicação desktop, o software será executado em qualquer sistema operacional que possua o JRE instalado (Windows, Linux ou macOS).
Segurança	**Validação em Camada de Negócio.** O sistema implementará sanitização de entradas para evitar caracteres inválidos e lógica de persistência protegida contra falhas de integridade de dados.
Persistência	**Padrão DAO com SQLite ou MySQL.** A persistência será isolada em uma camada específica, permitindo que o mecanismo de armazenamento seja trocado sem afetar a interface ou a lógica.
Internacionalização (i18n)	**Resource Bundles.** A aplicação será estruturada para suportar múltiplos idiomas através de arquivos de propriedades (.properties), facilitando a expansão futura.

Tabela 2 – Exemplo de requisitos e restrições

4. VISÃO DE CASOS DE USO

Esta seção lista as especificações centrais e significantes para a arquitetura do sistema, detalhando as funcionalidades que o usuário final poderá executar na Agenda Pessoal.

Caso de Uso [01] – **Cadastrar Contatos**

- **Descrição:** Permite ao usuário incluir um novo contato na base de dados da agenda.
- **Ator Principal:** Usuário.
- **Fluxo Principal:**
  1. O usuário acessa o formulário de cadastro na interface Java Swing.
  2. O usuário preenche os campos Nome, Telefone e E-mail e clica no botão “Salvar”.
  3. A Camada de Negócio (Service) recebe os dados e valida se os campos obrigatórios foram preenchidos corretamente.
  4. A Camada de Persistência (DAO) recebe o objeto e executa o comando SQL para gravar o contato no banco de dados.
  5. O sistema exibe uma mensagem de confirmação de sucesso para o usuário.

- **Fluxo de Exceção:** Caso a validação na Camada de Negócio falhe (ex: nome vazio), o sistema interrompe o processo e exibe um alerta de erro, sem acionar o banco de dados.

Caso de Uso [02] – **Acessar lista de contatos**

- **Descrição:** Permite ao usuário visualizar todos os contatos atualmente armazenados na agenda.
- **Ator Principal:** Usuário.
- **Fluxo Principal:**
  1. O usuário abre a aplicação ou aciona o comando de atualização da lista.
  2. A interface solicita os dados à Camada de Negócio.
  3. A Camada DAO executa uma consulta (SELECT) no banco de dados e retorna uma lista de objetos do tipo Contato.
  4. A interface Swing renderiza esses dados em um componente de tabela ou lista para o usuário.

Caso de Uso [03] – **Inativar contato cadastrado**

- **Descrição:** Permite ao usuário remover um contato selecionado da visualização ativa da agenda.
- **Ator Principal:** Usuário.
- **Fluxo Principal:**
  1. O usuário seleciona um contato na lista exibida na interface.
  2. O usuário aciona a opção “Excluir” ou “Inativar”.
  3. O sistema solicita confirmação da exclusão.
  4. A Camada de Negócio envia o ID do contato para a Camada DAO.
  5. O DAO executa a remoção (DELETE) ou atualização de status no banco de dados.
  6. A lista é atualizada na interface para refletir a alteração.

4.1 Casos de Uso significantes para a arquitetura

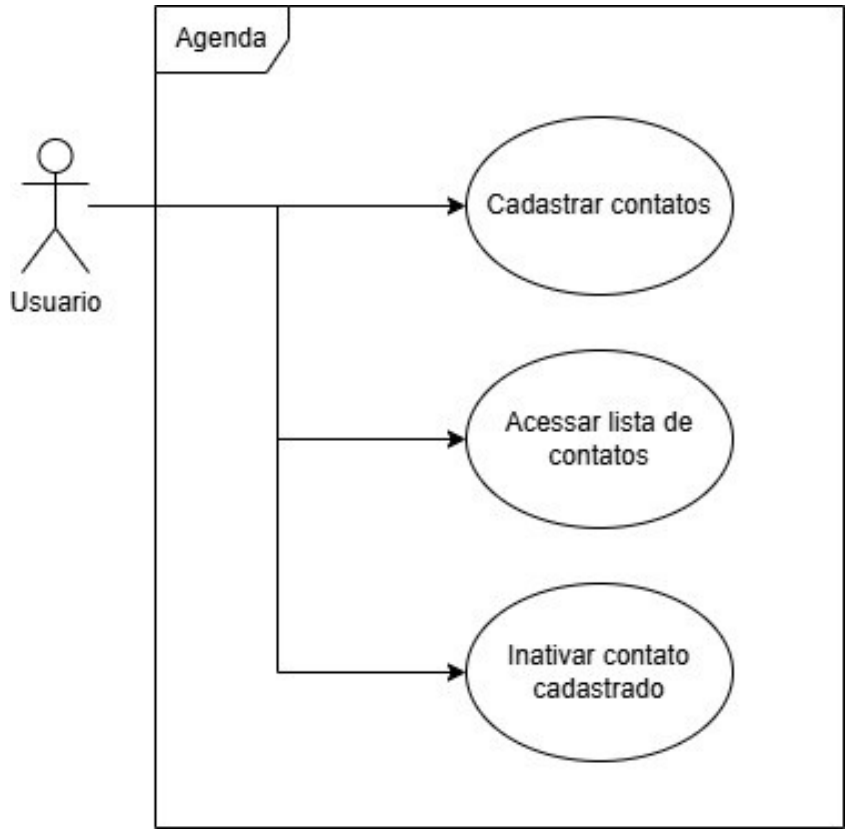

Figura 1 – Diagrama com os casos de uso significativos e atores

5. VISÃO LÓGICA

Esta seção descreve a organização das classes e subsistemas do software, evidenciando a separação de responsabilidades através da Arquitetura em Camadas. Abaixo, detalha-se a estrutura de pacotes utilizada para garantir a manutenibilidade e escalabilidade do sistema.

5.1 Visão Geral – pacotes e camadas

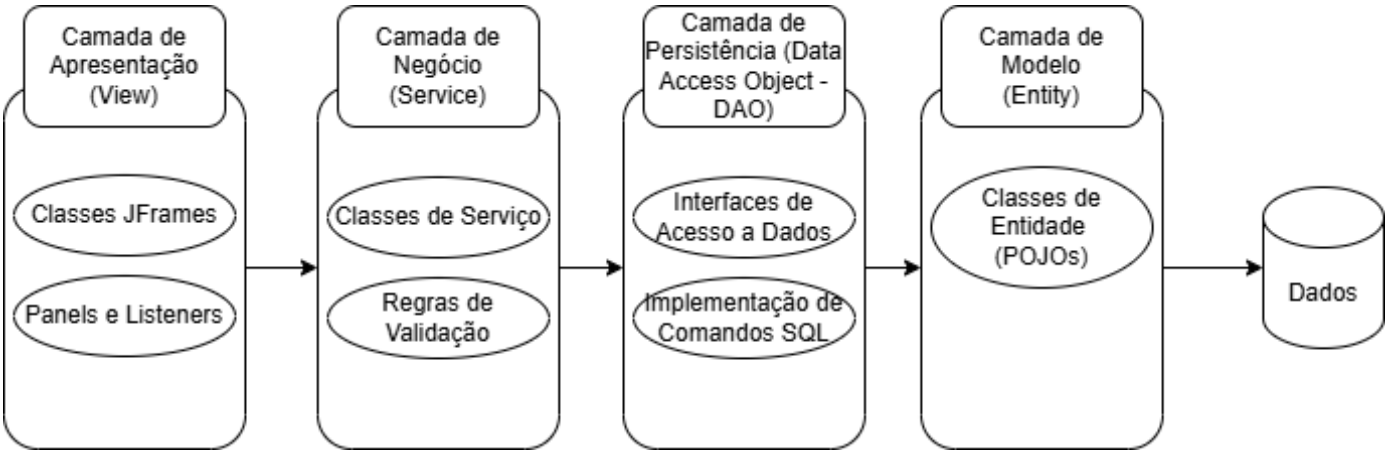

Figura 2 – Diagrama de Camadas da Aplicação

6. VISÃO DE IMPLEMENTAÇÃO

Esta visão foca na organização dos componentes físicos do software (código-fonte) e na sua realização através de classes Java.

6.1 Diagrama de Classes

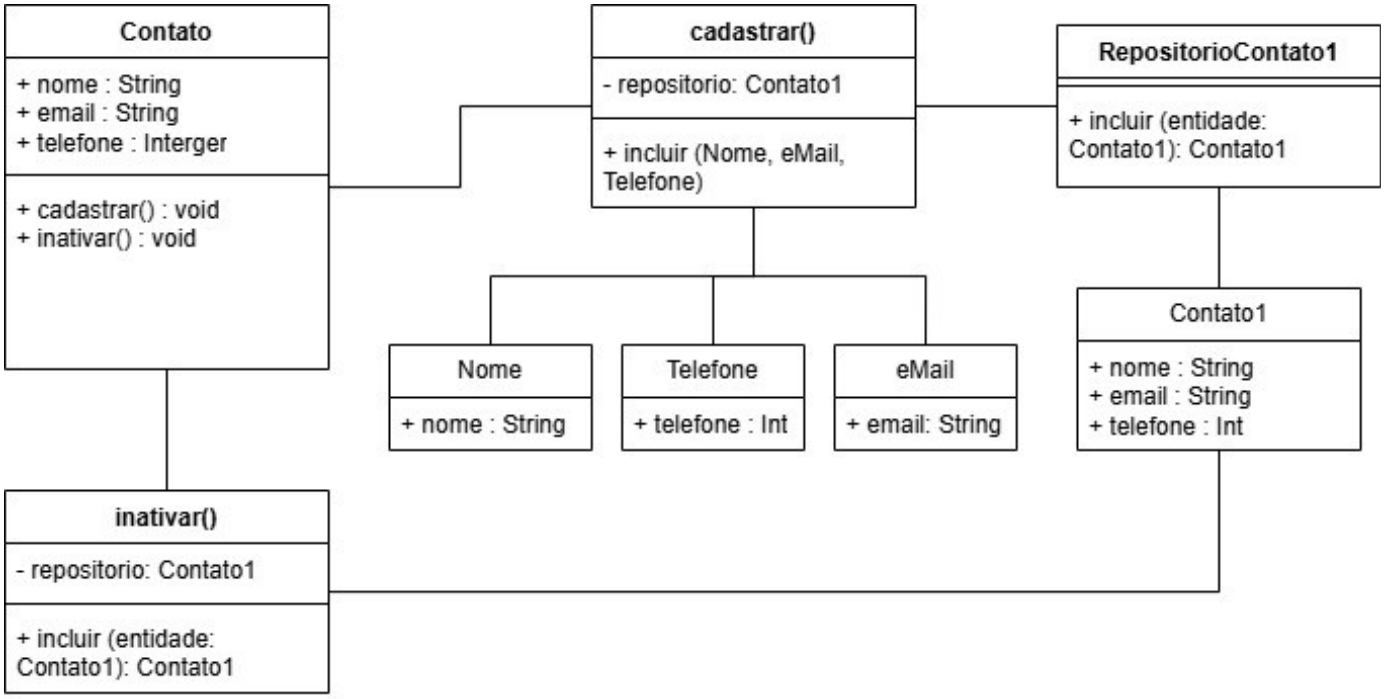

Figura 1 – Diagrama de Classes

6.1.1 Diagrama de Sequência

O Diagrama de Sequência (Figura 1) detalha o comportamento dinâmico do sistema durante o cadastro de um novo contato. O fluxo evidencia que a persistência no banco de dados só ocorre após a validação bem-sucedida pelo Controlador, garantindo que apenas dados íntegros sejam registrados no Repositório.

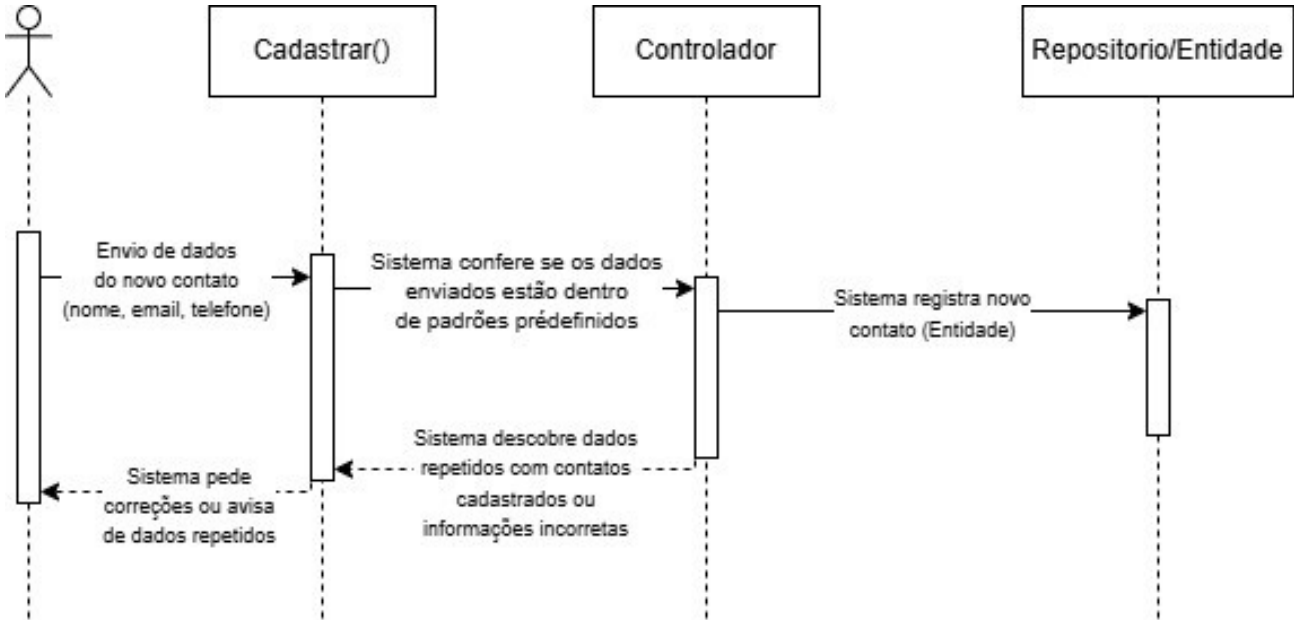

Figura 1 – Diagrama de Sequência

7. VISÃO DE IMPLANTAÇÃO

Descrever os nodos físicos, as configurações e os artefatos que serão implantados.

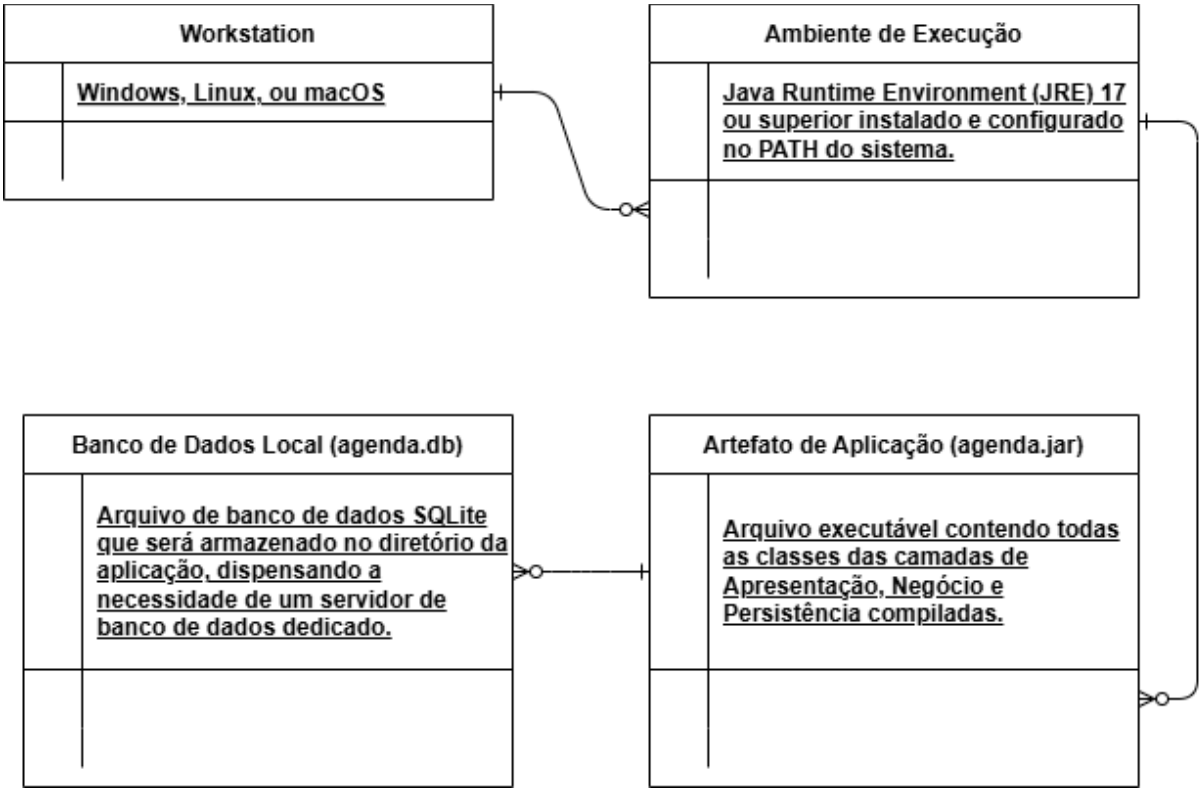

Figura 1 – Diagrama de Implantação Java

8. DIMENSIONAMENTO E PERFORMANCE

8.1 Volume

Enumerar os itens relativos ao volume de acesso aos recursos da aplicação:

- **Número de estimado usuários:** 01 (Aplicação *standalone* de uso pessoal).
- **Número estimado de acessos diários:** Frequentemente variado (estimado entre 1 a 10 acessos para consultas e edições).
- **Número estimado de acessos por período:** Suporte para até 10.000 contatos sem perda de performance perceptível (limitação técnica baseada no armazenamento do **SQLite**).
- **Tempo de sessão de um usuário:** Indeterminado (permanece ativo enquanto a janela da aplicação estiver aberta).

8.2 Performance

Enumerar os itens referentes à resposta esperada do sistema:

- **Tempo máximo para a execução de determinada transação:** Menos de 2 segundos para operações de CRUD (Cadastrar, Consultar, Atualizar e Deletar).

9. QUALIDADE

Esta seção enumera os atributos de qualidade considerados significativos para a arquitetura da Agenda Pessoal, garantindo que o sistema seja robusto e confiável.

Item	Descrição	Solução
Escalabilidade	Capacidade de suportar o aumento do volume de dados sem perda de performance.	O uso do banco de dados **SQLite** permite que a agenda suporte milhares de registros mantendo a velocidade de acesso local.
Confiabilidade, Disponibilidade	Garantia de que os dados não serão perdidos e o sistema estará pronto para uso.	Por ser uma aplicação **standalone** (local), o sistema não depende de conexão com a internet, garantindo 100% de disponibilidade offline.
Portabilidade	Facilidade de executar o sistema em diferentes ambientes.	A aplicação é empacotada em um arquivo **JAR**, permitindo execução em Windows, Linux ou macOS através da **JVM**.
Segurança	Proteção da integridade dos dados armazenados.	Implementação de validações rigorosas na **Camada de Negócio (Service)** para evitar a persistência de dados inconsistentes ou corrompidos.