RELATÓRIO

Aluno: Jessica de Sousa Costa e Mateus Vitor Silva Andrade

Professor: Igor Muzzeti

Disciplina: Engenharia de Software 2

1. Introdução

Este relatório documenta o processo de desenvolvimento do Sistema de Agendamento de Compromissos, abordando as fases desde a especificação de requisitos até a implementação e testes. O sistema foi desenvolvido seguindo um modelo iterativo, dividido em três sprints, cada um abordando aspectos essenciais do desenvolvimento de software.

2. Modelo de Processo

O modelo de processo adotado foi baseado em uma abordagem incremental. Na primeira sprint, foram especificados os requisitos funcionais e não-funcionais, estabelecendo um planejamento e documentando as necessidades do sistema. Na segunda sprint, ocorreu a elaboração do projeto arquitetural, definindo histórias de usuário, cenários de teste e diagramas UML, incluindo diagramas de classes, componentes e sequência. A terceira sprint envolveu a implementação do sistema em Java, seguindo a arquitetura previamente definida, além da realização de testes automatizados utilizando JUnit para garantir a funcionalidade e a estabilidade da aplicação. Cada sprint teve duração de 2 semanas.

2.1 Sprint 1 - Especificação de Requisitos

- Definição do modelo de processo e planejamento do cronograma
- Identificação dos principais requisitos funcionais e não-funcionais
- Documentação dos requisitos em um formato estruturado

2.2 Sprint 2 - Projeto Arquitetural

- Detalhamento dos requisitos funcionais em histórias de usuário
- Definição de cenários de teste
- Elaboração dos diagramas UML (Classes, Componentes e Sequência)

2.3 Sprint 3 - Implementação e Testes

- Desenvolvimento da aplicação em Java seguindo a arquitetura definida
- Implementação de testes automatizados utilizando JUnit
- Validação da funcionalidade e correção de erros

3. Especificação de Requisitos

A especificação de requisitos do sistema contemplou aspectos funcionais e não-funcionais essenciais para o seu funcionamento.

3.1 Requisitos Funcionais

RF1 - Cadastro de Usuário

- O sistema deve permitir o cadastro de novos usuários.
- Dados obrigatórios: nome, e-mail e senha.
- O e-mail deve ser único no sistema.

RF2 - Autenticação

- O sistema deve permitir o login de usuários cadastrados.
- O login deve ser realizado com e-mail e senha.

RF3 - Criação de Agendamento

- O sistema deve permitir a criação de novos agendamentos.
- Dados obrigatórios: data, hora e descrição.
- A data não pode ser anterior à data atual.

RF4 - Visualização de Agendamentos

- O sistema deve listar todos os agendamentos do usuário logado.

RF5 - Cancelamento de Agendamento

- O sistema deve permitir o cancelamento de agendamentos.
- Apenas o usuário que criou o agendamento pode cancelá-lo.

3.2 Requisitos Não-Funcionais

RNF1 - Interface Gráfica

- A interface deve ser intuitiva.

RNF2 - Performance

- O sistema deve responder em no máximo 2 segundos.

4. Histórias de Usuário

HU1 - Cadastro de Usuário

Como um novo usuário do sistema, **quero** poder criar uma conta, **para** ter acesso ao sistema de agendamentos.

Critérios de aceitação:

- O usuário deve fornecer nome, e-mail e senha.
- O e-mail deve ser único no sistema.

HU2 - Login

Como um usuário cadastrado, **quero** poder fazer login, **para** acessar minhas funcionalidades de agendamento.

HU3 - Criação de Agendamento

Como um usuário logado, **quero** poder criar um novo agendamento, **para** registrar meus compromissos.

Critérios de aceitação:

- O usuário deve fornecer data, hora e descrição.
- A data não pode ser anterior à data atual.

5. Projeto Arquitetural

O sistema foi desenvolvido utilizando a arquitetura MVC (Model-View-Controller), que permite uma separação clara entre as camadas de lógica de negócio, interface e controle.

5.1 Diagramas UML

Diagrama de Classes

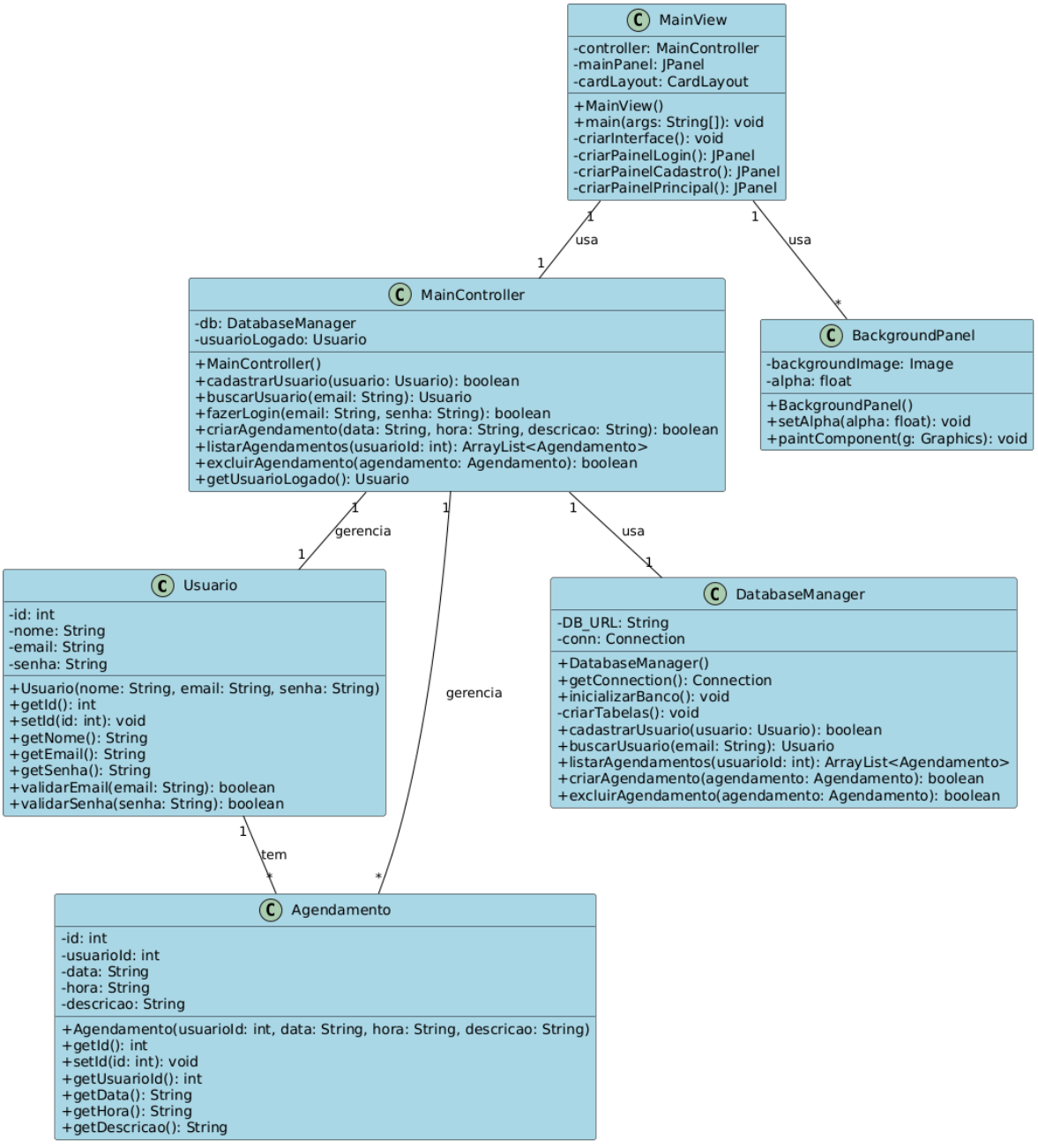


Diagrama de Componentes

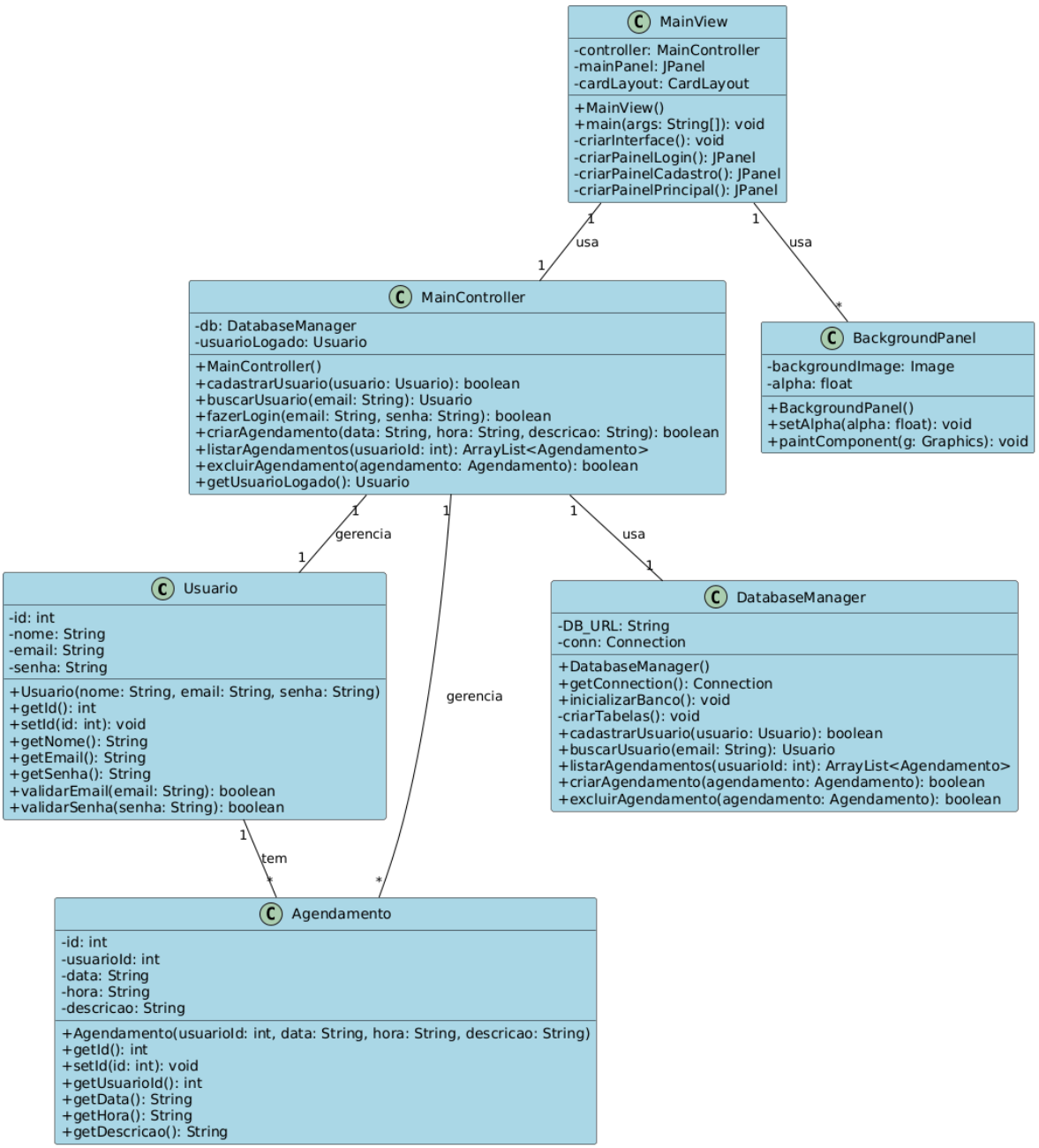
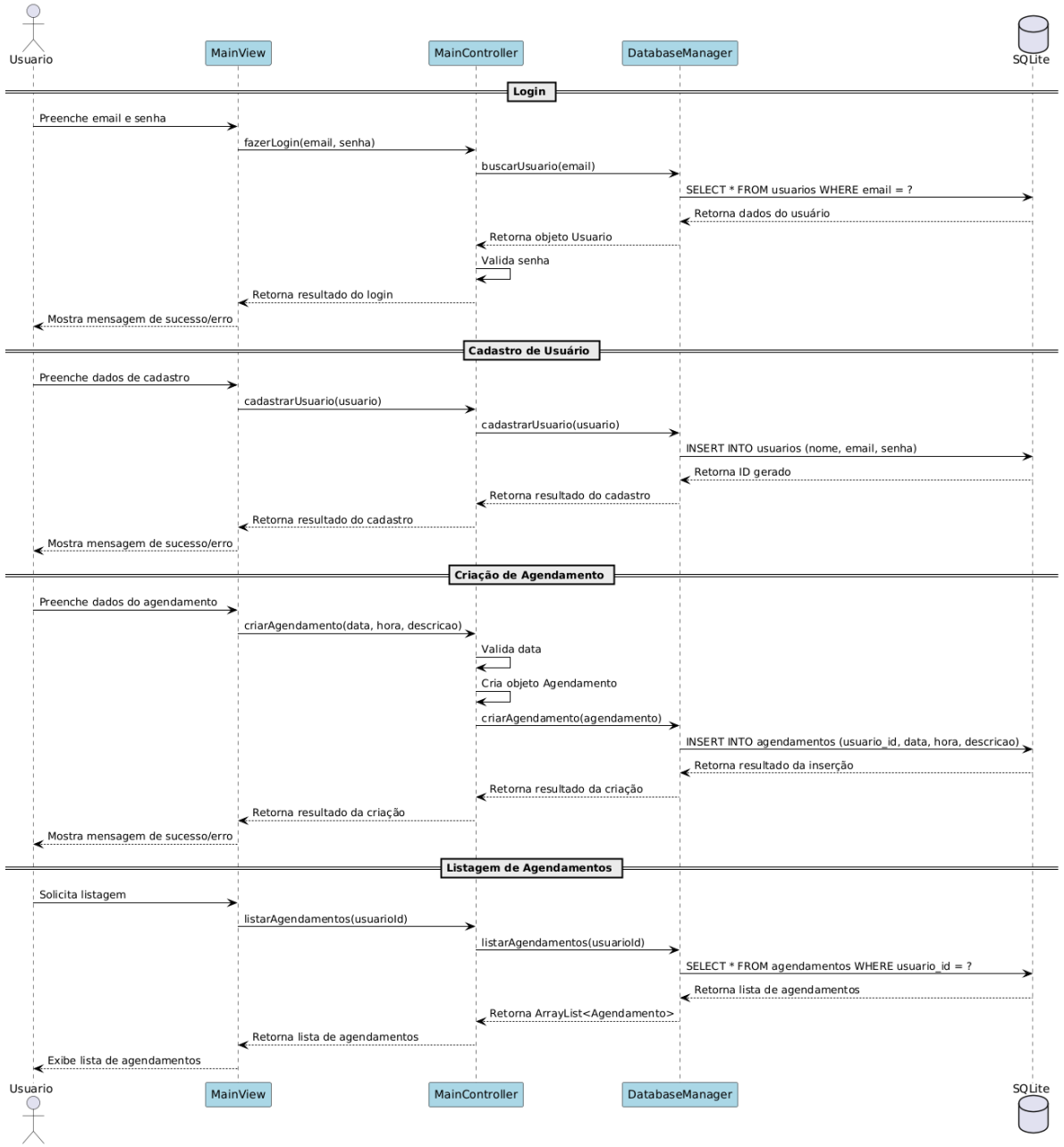


Diagrama de Sequência



6. Implementação

O sistema foi implementado em Java utilizando Swing para a interface gráfica, a escolha da biblioteca Swing se deu por conta, principalmente, da sua integração nativa com Java e por não ser tão complicada de trabalhar. Além disso, sua compatibilidade com a arquitetura MVC facilita a separação entre a lógica de apresentação e a lógica de negócio, garantindo um código mais modular e de fácil manutenção.

Para o armazenamento de dados, optamos pelo banco de dados SQLite devido à sua facilidade de integração com aplicações Java. Como um banco de dados relacional embutido, o SQLite retira a necessidade de um servidor externo, o que o torna uma solução viável para aplicações de pequeno porte.

As principais classes incluem:

- Usuario: Gerencia informações de usuários.
- Agendamento: Representa os compromissos do usuário.
- DatabaseManager: Gerencia todas as operações com o banco de dados SQLite.
- MainController: Gerencia a lógica de negócio coordena as interações entre a View e o Model.
- MainView: Interface principal do sistema, gerencia todas as telas (login, cadastro, principal)
- BackgroundPanel: Painel personalizado para exibir a imagem de fundo nas telas.

7. Testes Automatizados

Os testes automatizados foram conduzidos com JUnit para validar a confiabilidade do sistema. Os testes de unidade verificaram a criação e autenticação de usuários, assegurando que apenas credenciais válidas permitam acesso. Além disso, a funcionalidade de agendamento foi testada para garantir o correto registro e validação das informações inseridas.

7.1 Testes de Unidade

Teste de Cadastro de Usuário

- Criar um usuário válido e verificar se é salvo corretamente.
- Testa criação, validação de email e senha.

Teste de Agendamento

- Criar agendamentos válidos e verificar se estão salvos corretamente.
- Testar agendamentos com data inválida.

Teste de Database

- Testa operações com o banco de dados.

Teste de MainController

- Testa fluxos de negócio.

Teste de MainView

- Testa criação e visibilidade dos componentes.

8. Conclusão

O desenvolvimento do Sistema de Agendamento de Compromissos foi realizado seguindo boas práticas de engenharia de software. Apesar de ter sido um projeto de pequeno porte e básico, conseguimos captar bem a lógica por trás do desenvolvimento de um sistema de forma total, desde o modelo de processo até os testes. A divisão em sprints permitiu um desenvolvimento incremental, de forma que cada funcionalidade fosse bem definida, implementada e testada. O sistema atende aos requisitos definidos e pode ser expandido para futuras melhorias.