WFACULDADE DE TECNOLOGIA ARTHUR DE AZEVEDO

CURSO DE ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

Documentação de software:

reunion

Gabriel Danilo do Nascimento e Silva

LUCAS VIEIRA DA SILVA

Marcelo felipe belotto

Vinícius Emanuel da Silva

MOGI MIRIM

2024

RESUMO

Este documento apresenta o desenvolvimento e a implementação de um sistema de gerenciamento de reservas de salas de reunião, o ReuniON, projetado para otimizar o agendamento e o uso de salas em ambientes corporativos. O sistema foi estruturado de forma modular, com camadas distintas para bibliotecas e implementação, facilitando a manutenção e a escalabilidade. A estratégia de testes incluiu testes unitários, de integração e de sistema, assegurando a confiabilidade e a robustez do software. Durante o desenvolvimento, foram identificadas limitações, como a interface de linha de comando e desafios no controle de concorrência, que indicam áreas para melhorias futuras. O sistema alcançou os objetivos de eficiência e organização, com resultados positivos na gestão de reservas. Futuras expansões podem incluir o desenvolvimento de uma interface gráfica e a integração com calendários externos. As lições aprendidas destacam a importância da modularidade e dos testes contínuos.

Palavras-chave: gerenciamento de reservas; salas de reunião; sistema modular; testes de software; interface de usuário; controle de concorrência; escalabilidade.

SUMÁRIO

[1 introdução 4](#_Toc181620275)

[1.1 TEMA 4](#_Toc181620276)

[1.1.1 Delimitação do Problema 4](#_Toc181620277)

[1.1.2 Escolha do tema 4](#_Toc181620278)

[1.2 OBJetivos 4](#_Toc181620279)

[1.2.1 Objetivos Específicos 5](#_Toc181620280)

[1.3 modelo de Trabalho 5](#_Toc181620281)

[1.4 Estrutura do documento 5](#_Toc181620282)

[2 DESCRIÇÃO GERAL DO sISTEMA 6](#_Toc181620283)

[2.1 principais envolvidos e suas características 6](#_Toc181620284)

[2.1.1 Usuários do Sistema 6](#_Toc181620285)

[2.1.2 Desenvolvedores do Sistema 6](#_Toc181620286)

[2.1.3 Regras de Negócio 6](#_Toc181620287)

[3 requisitos do sistema 8](#_Toc181620288)

[3.1 Requisitos funcionais 8](#_Toc181620289)

[4 arquitetura do sistema 13](#_Toc181620290)

[4.1 Visão Geral da Arquitetura 13](#_Toc181620291)

[4.2 Componentes do Sistema 13](#_Toc181620292)

[4.3 Tecnologias Utilizadas 14](#_Toc181620293)

[4.4 Diagrama de Arquitetura 14](#_Toc181620294)

[5 IMPLEMENTAÇÃO DO SISTEMA 15](#_Toc181620295)

[5.1 ESTRUra do código 15](#_Toc181620296)

[5.2 Principais Algoritmos e Funções 15](#_Toc181620297)

[5.3 Padrões de Codificação 16](#_Toc181620298)

[5.4 Ferramentas de Desenvolvimento 16](#_Toc181620299)

[6 Testes e Validação 17](#_Toc181620300)

[6.1 Estratégia de Testes 17](#_Toc181620301)

[6.2 Casos de Teste 17](#_Toc181620302)

[6.3 Ferramentas de Teste 18](#_Toc181620303)

[6.4 Resultados dos Testes 18](#_Toc181620304)

[7 Implantação do Sistema 20](#_Toc181620305)

[7.1 Ambiente de Implantação 20](#_Toc181620306)

[7.2 Procedimentos de Implantação 20](#_Toc181620307)

[7.3 Treinamento e Suporte 21](#_Toc181620308)

[7.4 Considerações de Manutenção 21](#_Toc181620309)

[8 Conclusão 23](#_Toc181620310)

[8.1 Resultados Alcançados 23](#_Toc181620311)

[8.2 Limitações e Desafios 23](#_Toc181620312)

[8.3 Trabalhos Futuros 23](#_Toc181620313)

[8.4 Lições Aprendidas 24](#_Toc181620314)

[REFERÊNCIAS 25](#_Toc181620315)

# introdução

O presente documento tem como objetivo apresentar o projeto de desenvolvimento do ReuniON, um sistema de gerenciamento de reservas de salas de reunião, que visa otimizar o agendamento e o uso de salas de reunião em uma empresa ou organização, facilitando o arranjo e o uso eficiente dos recursos internos.

## TEMA

O tema do projeto é o desenvolvimento de um sistema de gerenciamento de reservas de salas de reunião, que busca atender às necessidades de agendamento e utilização eficaz das salas de reunião dentro de uma organização.

### Delimitação do Problema

O problema central que o projeto visa resolver é o gerenciamento das reservas de salas de reunião. Muitas organizações enfrentam dificuldades em organizar e otimizar o uso de suas salas de reunião, resultando em conflitos de agendamento e uso ineficiente dos espaços disponíveis.

### Escolha do tema

A escolha do tema se deu pela percepção de que o desenvolvimento de um sistema seria desafiador, tanto do ponto de vista técnico quanto organizacional. Além disso, a implementação de tal sistema pode trazer benefícios significativos para as organizações, melhorando a eficiência e a produtividade.

## OBJetivos

O objetivo principal do projeto é otimizar o agendamento e o uso de salas de reunião, proporcionando uma ferramenta que facilite a organização e o uso eficiente dos recursos internos.

### Objetivos Específicos

Os objetivos específicos incluem a criação de um sistema intuitivo e acessível para os usuários, que permita o agendamento rápido e eficiente das salas, além de fornecer informações sobre a utilização dos espaços.

## modelo de Trabalho

Para a realização do projeto, será utilizado o modelo tradicional de desenvolvimento de software, conhecido como modelo cascata. Este método prevê uma abordagem sequencial, onde cada fase do desenvolvimento deve ser concluída antes do início da próxima.

## Estrutura do documento

O documento abordará desde a introdução e descrição geral do sistema até a implementação, testes e apresentação do software.

# DESCRIÇÃO GERAL DO sISTEMA

O ReuniON visa melhorar o agendamento e a utilização de salas em empresas e organizações. A falta de um sistema eficaz resulta em desorganização, uso ineficiente dos recursos e conflitos de agenda. A solução é um sistema intuitivo que permite agendamentos rápidos e fornece informações sobre o uso dos espaços.

## principais envolvidos e suas características

### Usuários do Sistema

Existem dois tipos de usuários: o usuário geral, que pode gerenciar suas próprias reservas, e o administrador, que além dessas funções, pode cadastrar e gerenciar salas e visualizar todas as reservas. O sistema é adequado para qualquer organização com salas de reunião.

### Desenvolvedores do Sistema

O desenvolvimento está a cargo de Gabriel Danilo do Nascimento e Silva, Lucas Vieira da Silva, Marcelo Felipe Belotto e Vinicius Emanuel da Silva, que colaboram em todas as etapas do projeto.

### Regras de Negócio

As regras de negócio incluem:

* Acesso: Apenas usuários autenticados podem fazer reservas, com privilégios adicionais para administradores.
* Desempenho: O sistema deve responder rapidamente às solicitações.
* Confiabilidade: Deve lidar com falhas de rede ou servidor, garantindo a integridade dos dados.
* Escalabilidade: Capacidade de armazenar um grande volume de dados, acompanhando o crescimento de usuários e salas.
* Ferramentas: Uso de tecnologias que suportem a escalabilidade e manutenção do sistema.

# requisitos do sistema

## Requisitos funcionais

O ReuniON deve oferecer um conjunto abrangente de funcionalidades para atender às necessidades de agendamento e gerenciamento de salas.

|  |
| --- |
| **RF01 - CADASTRAR SALAS** |
| O sistema deve permitir que o administrador registre novas salas, inserindo informações detalhadas como tipo, número, capacidade e equipamentos disponíveis. Isso garante que as salas estejam prontas para serem reservadas pelos usuários. |
| **Fluxo de Eventos Normal** |
| 1. O sistema exibe o menu inicial. 2. O administrador digita sua matrícula e senha. 3. O sistema exibe o menu “administrador”. 4. O administrador seleciona a opção de cadastrar nova sala. 5. O sistema solicita as informações da sala (tipo, número, capacidade, equipamentos). 6. O administrador insere as informações solicitadas. 7. O sistema valida as informações fornecidas. 8. O sistema registra a nova sala no banco de dados. 9. O sistema exibe uma mensagem de confirmação do cadastro. 10. O sistema retorna ao menu administrador. |
| **Fluxo de Eventos de Exceção** |
| 2a - Informações inválidas:  4a.1 - O sistema exibe uma mensagem de erro indicando quais informações são inválidas.  4a.2 - O sistema retorna ao passo 3.  4a - Administrador cancela a operação:  4a.1 - O sistema cancela o cadastro da sala.  4a.2 – O sistema retorna ao passo 3 do fluxo normal.  7a - Informações inválidas:  7a.1 - O sistema exibe uma mensagem de erro indicando quais informações são inválidas.  7a.2 - O sistema retorna ao passo 5 do fluxo normal.  5a - Falha no sistema ao cadastrar:  5a.1 - O sistema exibe uma mensagem de erro informando a falha no cadastro.  5a.2 - O sistema retorna ao passo 5 do fluxo normal. |

|  |
| --- |
| **RF02 - CADASTRAR USUÁRIOS** |
| Deve ser possível para o administrador adicionar novos usuários ao sistema, atribuindo-lhes permissões específicas. Isso assegura que cada usuário tenha acesso adequado às funcionalidades do sistema conforme suas necessidades. |
| **Fluxo de Eventos Normal** |
| 1. O sistema exibe o menu inicial. 2. O administrador digita sua matrícula e senha. 3. O sistema exibe o menu administrador. 4. O administrador seleciona a opção de cadastrar novo usuário. 5. O sistema solicita as informações do usuário (matrícula, perfil de acesso e senha) 6. O administrador insere as informações solicitadas. 7. O sistema valida as informações inseridas. 8. O sistema cadastra o novo usuário e exibe uma mensagem de confirmação. 9. O sistema retorna ao menu administrador. |
| **Fluxo de Eventos de Exceção** |
| 4a - Informações inválidas:  4a.1 - O sistema exibe uma mensagem de erro indicando quais informações são inválidas.  4a.2 - O sistema retorna ao passo 3.  4b – Matrícula já cadastrada:  4b.1 - O sistema exibe uma mensagem informando que a matrícula já está em uso.  4b.2 - O sistema retorna ao passo 3. |

|  |
| --- |
| **RF03 – VERIFICAR DISPONIBILIDADE DE SALAS** |
| Os usuários devem poder consultar a disponibilidade das salas em tempo real, permitindo que verifiquem se o horário desejado está livre antes de efetuar uma reserva. |
| **Fluxo de Eventos Normal** |
| 1. O sistema exibe o menu inicial. 2. O usuário digita sua matrícula e senha. 3. O sistema exibe o menu. 4. O usuário seleciona a opção de verificar disponibilidade de salas. 5. O sistema solicita o número da sala, data e horário desejados. 6. O usuário insere a data e horário. 7. O sistema verifica a disponibilidade das salas para o período especificado. 8. O sistema exibe uma lista das salas disponíveis com suas respectivas informações. 9. O sistema retorna ao menu geral. |
| **Fluxo de Eventos de Exceção** |
| 3a - Usuário cancela a operação:  3a.1 - O sistema cancela a verificação de disponibilidade.  3a.2 - O sistema retorna ao passo 3 do fluxo normal.  5a - Nenhuma sala disponível:  5a.1 - O sistema informa que não há salas disponíveis para o período especificado.  5a.2 - O sistema retorna ao passo 5 do fluxo normal.  7a - Data/horário inválidos:  7a.1 - O sistema exibe uma mensagem de erro indicando que a data ou horário são inválidos.  7a.2 - O sistema retorna ao passo 5 do fluxo normal. |

|  |
| --- |
| **RF04 – RESERVAR SALAS** |
| O sistema deve facilitar o processo de reserva, permitindo que os usuários especifiquem o horário e a sala desejada, garantindo que o espaço esteja disponível para suas reuniões. |
| **Fluxo de Eventos Normal** |
| 1. O sistema exibe o menu inicial. 2. O usuário digita sua matrícula e senha. 3. O sistema exibe o menu geral. 4. O usuário seleciona a opção de fazer uma reserva. 5. O sistema solicita a data, horário e sala desejados. 6. O usuário insere as informações solicitadas. 7. O sistema verifica a disponibilidade da sala para o período especificado. 8. O sistema confirma a disponibilidade e solicita confirmação do usuário. 9. O usuário confirma a reserva. 10. O sistema registra a reserva no banco de dados. 11. O sistema exibe uma mensagem de confirmação da reserva. 12. O sistema retorna ao menu geral. |
| **Fluxo de Eventos de Exceção** |
| 3a - Usuário cancela a operação:  3a.1 - O sistema cancela o processo de reserva.  3a.2 - O sistema retorna ao passo 3 do fluxo normal.  5a - Informações inválidas:  5a.1 - O sistema exibe uma mensagem de erro indicando quais informações são inválidas.  5a.2 - O sistema retorna ao passo 5 do fluxo normal.  7a - Sala não disponível:  7a.1 - O sistema informa que a sala não está disponível para o período especificado.  7a.2 - O sistema retorna ao passo 5 do fluxo normal. |

|  |
| --- |
| **RF05 – LISTAR SALAS** |
| Os usuários devem ter acesso a uma lista completa de salas cadastradas, incluindo detalhes sobre capacidade e equipamentos, para que possam escolher a sala que melhor atende às suas necessidades. |
| **Fluxo de Eventos Normal** |
| 1. O sistema exibe o menu inicial. 2. O usuário digita sua matrícula e senha. 3. O sistema exibe o menu geral. 4. O usuário seleciona a opção de listar salas. 5. O sistema recupera as informações de todas as salas cadastradas. 6. O sistema exibe uma lista de salas com seus respectivos detalhes (nome, número, capacidade, equipamentos). 7. O sistema retorna ao menu geral. |
| **Fluxo de Eventos de Exceção** |
| 3a - Usuário cancela a operação:  3a.1 - O sistema cancela o processo de reserva.  3a.2 - O sistema retorna ao passo 3 do fluxo normal.  5a - Informações inválidas:  5a.1 - O sistema exibe uma mensagem de erro indicando quais informações são inválidas.  5a.2 - O sistema retorna ao passo 5 do fluxo normal.  7a - Sala não disponível:  7a.1 - O sistema informa que a sala não está disponível para o período especificado.  7a.2 - O sistema retorna ao passo 5 do fluxo normal. |

Listagem de Reservas: O administrador deve poder visualizar todas as reservas feitas, permitindo um controle eficiente do uso das salas e planejamento da disponibilidade dos espaços.

Exibição de Reservas: Tanto usuários quanto administradores devem poder visualizar os detalhes de uma reserva específica, confirmando informações como horário e sala reservada.

Cancelamento de Reservas: O sistema deve permitir que reservas sejam canceladas, liberando o espaço para outros usuários e evitando conflitos de agendamento.

Alteração de Salas: O administrador deve ter a capacidade de editar informações das salas cadastradas, como capacidade ou equipamentos, garantindo que os dados estejam sempre atualizados.

Alteração de Usuários: Deve ser possível modificar dados dos usuários, incluindo permissões de acesso, para assegurar que apenas usuários autorizados tenham acesso às funcionalidades administrativas.

Exclusão de Salas: O sistema deve permitir a remoção de salas do cadastro, caso não estejam mais disponíveis para reserva.

Exclusão de Usuários: Deve ser possível excluir usuários do sistema, impedindo que acessem ou modifiquem informações caso não façam mais parte da equipe.

Salvamento de Reservas: As informações de reserva devem ser salvas de forma persistente, garantindo que os dados permaneçam disponíveis mesmo após reinicializações do sistema.

Carregamento de Reservas: O sistema deve ser capaz de carregar informações de reserva previamente salvas, assegurando que os dados históricos estejam sempre acessíveis.

# arquitetura do sistema

## Visão Geral da Arquitetura

A arquitetura do ReuniON é projetada de forma modular, organizada em duas camadas principais: a Camada de Biblioteca e a Camada de Implementação.

* **Camada de Dados (databases):** Esta camada é responsável por armazenar e organizar todos os registros e informações inseridas no sistema em arquivos de dados (.csv). Os arquivos principais incluem sala.csv e usuario.csv, onde são mantidos os dados das salas e dos usuários, respectivamente. Essa estrutura de dados em arquivos facilita o armazenamento persistente e a integridade das informações, permitindo acesso estruturado e seguro para as outras camadas do sistema.
* **Camada de Biblioteca (lib):** Esta camada contém os arquivos de cabeçalho (.h) que definem as interfaces e estruturas de dados utilizadas no sistema. Os arquivos incluem bancoDados.h, menu.h, reservas.h, salas.h, usuario.h e utils.h. Essa organização facilita a modularização e a reutilização do código, permitindo que as interfaces sejam claramente definidas e acessíveis para os componentes que necessitam delas.
* **Camada de Implementação (src):** Nesta camada, encontram-se os arquivos de implementação (.c) correspondentes, onde as funcionalidades são efetivamente implementadas. Os arquivos incluem bancoDados.c, menu.c, reservas.c, salas.c, usuario.c e utils.c.
* **Camada principal:** Esta camada contém o ponto de entrada do sistema, representado pelo arquivo main.c, que orquestra a execução de todas as funcionalidades implementadas nas camadas anteriores. Além do main.c, esta camada inclui todos os arquivos e dependências necessárias para o funcionamento do sistema, gerenciando a inicialização, fluxo e coordenação entre os componentes para uma experiência integrada e coesa.

Essa estrutura modular permite que cada componente funcione de forma independente e interaja com os demais por meio de interfaces definidas, facilitando a manutenção e futuras expansões do sistema.

## Componentes do Sistema

Os principais componentes do sistema e suas responsabilidades são:

* **Banco de Dados:** Responsável por gerenciar a persistência das informações sobre salas e reservas, garantindo que os dados sejam armazenados e recuperados de forma eficiente.
* **Menu:** Define e controla a interface de interação com o usuário, exibindo as opções disponíveis e capturando as entradas do usuário.
* **Reservas:** Lida com o processo de reserva de salas, incluindo a verificação de disponibilidade e o cancelamento de reservas, assegurando que os conflitos de agendamento sejam evitados.
* **Salas:** Gerencia o cadastro e as informações das salas de reunião, permitindo que novas salas sejam adicionadas ou modificadas conforme necessário.
* **Usuário:** Lida com o cadastro e autenticação dos usuários, controlando os níveis de permissão (usuário e administrador) para garantir a segurança e a integridade do sistema.
* **Utils:** Provê funções utilitárias, como validações e manipulações de strings, que são utilizadas por outros componentes para realizar operações comuns.

Esses componentes são independentes, mas interagem para formar um sistema coeso e funcional.

## Tecnologias Utilizadas

O sistema foi desenvolvido utilizando as seguintes tecnologias:

* **Linguagem de Programação:** C, escolhida por sua eficiência e controle sobre a gestão de memória e desempenho, permitindo um sistema rápido e de baixo nível.
* **Ambiente de Desenvolvimento:** Visual Studio Code, utilizado como IDE para facilitar o desenvolvimento e a depuração do código.
* **Controle de Versão:** Git, utilizado para gerenciar o código-fonte e colaborar de forma eficaz durante o desenvolvimento.

Essas tecnologias foram selecionadas para oferecer um sistema robusto e eficiente, com controle total sobre os recursos do sistema.

# IMPLEMENTAÇÃO DO SISTEMA

## ESTRUra do código

O sistema é organizado em uma estrutura de diretórios clara, que separa as definições de interfaces, implementações e dados. No diretório lib, encontram-se os arquivos de cabeçalho (.h), como bancoDados.h, menu.h, reservas.h, salas.h, usuario.h e utils.h. Esses arquivos definem as funções, tipos de dados e constantes, promovendo a modularidade do sistema. Já o diretório src abriga os arquivos de implementação (.c), onde a lógica dos módulos é realizada. O arquivo main.c está localizado na camada principal, e desempenha um papel central, coordenando a execução do sistema ao chamar funções de diferentes módulos para operações de reserva, consulta e gestão de salas. Além disso, o diretório databases contém os arquivos de dados (.csv), como sala.csv e usuario.csv, que armazenam as informações persistentes. Essa organização modular facilita a manutenção e evolução do código, permitindo que cada módulo tenha uma responsabilidade bem definida.

## Principais Algoritmos e Funções

O sistema implementa uma série de algoritmos e funções essenciais para seu funcionamento:

* **Banco de Dados:** Funções que garantem a persistência dos dados, salvando e recuperando informações sobre reservas e salas.
* **Menu:** Funções que exibem opções ao usuário, capturam entradas e direcionam para as operações adequadas.
* **Reservas:** Funções que gerenciam a lógica de reserva, verificando a disponibilidade de salas e resolvendo conflitos.
* **Salas:** Funções que cuidam do cadastro e gestão das informações das salas de reunião.
* **Usuário:** Funções que realizam o cadastro e autenticação dos usuários, além de controlar permissões.
* **Utilitárias (utils):** Funções de suporte que realizam validações, formatação e manipulação de strings.

Essas funções são organizadas de forma que o main.c centraliza o controle do fluxo de execução, garantindo que as operações sejam realizadas de maneira coordenada.

## Padrões de Codificação

Durante o desenvolvimento, foram seguidos padrões de codificação que asseguram a clareza e a consistência do código:

* **Nomenclatura de Variáveis:** Utilização de nomes claros e descritivos, adotando o padrão camelCase para melhorar a legibilidade.
* **Formatação do Código:** O código é formatado de maneira consistente, com indentação e espaçamento adequados, facilitando a leitura e manutenção.
* **Comentários:** Inclusão de comentários para explicar blocos de código complexos e documentar o propósito de cada função, auxiliando na compreensão do código por outros desenvolvedores.

Essas práticas tornam o código mais acessível e colaborativo, facilitando futuras expansões.

## Ferramentas de Desenvolvimento

Para o desenvolvimento e depuração do código, foram utilizadas as seguintes ferramentas:

* **IDE:** Visual Studio Code, que oferece um ambiente de desenvolvimento integrado com suporte a extensões e ferramentas de depuração.
* **Compilador:** GCC, amplamente utilizado para compilar projetos em C, garantindo compatibilidade e eficiência.
* **Controle de Versão:** Git, utilizado para rastrear alterações e colaborar de forma eficiente no projeto.

Essas ferramentas foram escolhidas para proporcionar um fluxo de trabalho ágil e eficiente, especialmente no contexto de desenvolvimento em C.

# Testes e Validação

## Estratégia de Testes

A estratégia de testes adotada para o ReuniON é abrangente, cobrindo diferentes níveis de verificação para assegurar a qualidade e a robustez do sistema. A abordagem inclui:

* **Testes Unitários:** Cada módulo do sistema, como banco de dados, menu, reservas, salas, usuário e utils, é testado de forma isolada. Esses testes garantem que as funções individuais operem conforme o esperado, permitindo a identificação precoce de problemas em funções específicas antes de avançar para testes mais complexos.
* **Testes de Integração:** Após a validação dos módulos individualmente, são realizados testes para verificar a integração entre eles. Isso assegura que a interação entre os módulos ocorra corretamente, como a comunicação entre o módulo de reservas e o banco de dados, garantindo que as reservas sejam armazenadas e recuperadas adequadamente.
* **Testes de Sistema:** Testes completos são realizados no sistema, simulando o fluxo de uso real desde o login até a realização de reservas e consultas. Esses testes cobrem tanto casos de sucesso quanto situações de erro, garantindo que o sistema funcione corretamente em um cenário real.

Essa abordagem permite identificar problemas em diferentes níveis do sistema, assegurando tanto a confiabilidade das funcionalidades individuais quanto a robustez do sistema como um todo.

## Casos de Teste

Os principais casos de teste definidos para o sistema incluem:

* **Cadastro de Usuário:** Verifica se um usuário pode ser cadastrado com informações válidas e se o sistema impede cadastros com dados inválidos.
* **Autenticação de Usuário:** Testa o login para usuários com permissões diferentes (usuário comum e administrador) e verifica se o sistema lida corretamente com tentativas de login inválidas.
* **Reserva de Sala:** Verifica se o sistema permite reservas de salas disponíveis e bloqueia reservas em horários ou datas conflitantes.
* **Cancelamento de Reserva:** Testa a capacidade de cancelar uma reserva existente e verifica se o sistema lida corretamente com tentativas de cancelamento de reservas inexistentes.
* **Consulta de Salas:** Verifica se as informações de salas cadastradas são exibidas corretamente ao usuário.
* **Persistência de Dados:** Testa se os dados de salas e reservas são armazenados e recuperados corretamente após o sistema ser fechado e reiniciado.

Esses casos cobrem as funcionalidades principais e verificam tanto o funcionamento em condições normais quanto o tratamento de erros.

## Ferramentas de Teste

Para a realização dos testes, foram utilizadas as seguintes ferramentas:

* **Compilador GCC com Flags de Depuração:** Permite compilar o código com opções de depuração, facilitando a identificação de erros.
* **Testes Manuais:** Testes manuais foram realizados para simular fluxos de usuário, garantindo que as funcionalidades principais e cenários comuns estejam funcionando como esperado.
* **Logs e Prints:** Utilização de mensagens de log e printf para verificar o fluxo do programa e identificar pontos de falha durante a execução dos testes.

## Resultados dos Testes

Os resultados dos testes realizados indicaram:

* **Falhas Identificadas:** Durante os testes iniciais, foram encontrados alguns problemas, como erros de validação no cadastro de usuários e conflitos na reserva de salas quando mais de um usuário tentava reservar o mesmo horário.
* **Correções Implementadas:** Para resolver essas falhas, foram adicionadas verificações mais rigorosas de validação no cadastro de usuários e bloqueios temporários durante o processo de reserva para evitar conflitos. Funções auxiliares em utils foram ajustadas para garantir a integridade dos dados ao longo do fluxo de reserva e cancelamento.
* **Resultados Finais:** Após as correções, o sistema passou em todos os casos de teste, mostrando-se funcional tanto em condições normais quanto em cenários de erro.

# Conclusão

## Resultados Alcançados

* **Eficiência na Gestão de Reservas:** O sistema alcançou os objetivos iniciais, permitindo que os usuários reservem salas de forma fácil e rápida, eliminando conflitos de agendamento e melhorando a organização.
* **Facilidade de Uso:** A interface de menu clara possibilita que os usuários realizem operações básicas, como cadastro, reserva e cancelamento, sem dificuldades.
* **Persistência de Dados:** A implementação de um banco de dados assegura que as informações de reservas e usuários sejam mantidas mesmo após o fechamento do sistema, aumentando a confiabilidade.
* **Modularidade e Extensibilidade:** A arquitetura modular facilita a manutenção e futuras melhorias, permitindo a escalabilidade do sistema e a adição de novas funcionalidades.

## Limitações e Desafios

* **Interface limitada:** A interface de linha de comando pode ser uma barreira para usuários menos experientes, que poderiam se beneficiar de uma interface gráfica.
* **Controle de conflitos em tempo real:** Em ambientes com muitos usuários tentando reservar simultaneamente, pode haver atrasos na atualização dos dados.
* **Escalabilidade:** O sistema atende bem a ambientes de pequeno a médio porte, mas pode precisar de ajustes para suportar um volume muito alto de usuários e reservas simultâneas.

## Trabalhos Futuros

* **Interface gráfica (GUI):** Desenvolver uma interface gráfica para tornar o sistema mais acessível e fácil de usar.
* **Integração com calendários externos**: Integrar com sistemas de calendário como Google Calendar ou Outlook para sincronizar reservas com calendários pessoais.
* **Controle de conflitos e concorrência:** Melhorar o gerenciamento de reservas simultâneas, implementando bloqueios e sincronização mais eficazes.
* **Notificações e alertas:** Adicionar notificações para lembrar os usuários de suas reservas e alertá-los sobre mudanças.

## Lições Aprendidas

* **Importância da modularidade:** A arquitetura modular facilitou o desenvolvimento, manutenção e teste do sistema, destacando a importância de uma estrutura de código bem-organizada.
* **Testes contínuos:** Realizar testes durante todo o desenvolvimento ajudou a identificar e resolver problemas rapidamente, evitando a acumulação de erros.
* **Planejamento de interface:** Uma interface intuitiva é fundamental para a experiência do usuário, mesmo em sistemas de linha de comando. A clareza do menu e das opções contribuiu significativamente para a usabilidade.
* **Gerenciamento de conflitos:** O controle de concorrência mostrou-se um desafio, destacando a necessidade de explorar melhores práticas de sincronização de dados para evitar conflitos

###### REFERÊNCIAS

SOMMERVILLE, Ian. **Engenharia de Software**. Tradução Ivan Bosnic e Kalinka G. de O. Gonçalves; revisão técnica Kechi Hirama. 9. ed. São Paulo: Pearson Prentice Hall, 2011. Título original: Software engineering. ISBN 978-85-7936-108-1.

UFES - Universidade Federal do Espírito Santo. **Engenharia de Requisitos: Notas de Aula**. Ricardo de Almeida Falbo. 2017.

UFES - Universidade Federal do Espírito Santo. **Engenharia de Software: Notas de Aula**. Ricardo de Almeida Falbo. 2014.

SERRA, Ana Paula Gonçalves. **Documentação de um Produto de Software**. Versão 3.0. Revisor: Prof. Fernando Giorno. 2005.