CENTRO UNIVERSITÁRIO CIDADE VERDE

DEPARTAMENTO DE TECNOLOGIA

ENGENHARIA DE SOFTWARE

DÉBORA ROSADA DE OLIVEIRA

FERNANDA MARTINS VIOTTO DA SILVA

LARRISA AMÉLIA DE ALMEIDA

SpaceNow

MARINGÁ

2025

DÉBORA ROSADA DE OLIVEIRA

FERNANDA MARTINS VIOTTO DA SILVA

LARRISA AMÉLIA DE ALMEIDA

SpaceNow

Trabalho apresentado como requisito parcial de avaliação da disciplina Laboratório de desenvolvimento de Software, ministrada pelo Prof. Gustavo Meneghetti.

MARINGÁ

2025

Sumário

1. Visão geral

1.1 Objetivo do Software

O SpaceNow é uma plataforma web responsiva (desktop e mobile) para cadastro e reserva de salas. O sistema permite que usuários possam se autenticar, visualizar salas disponíveis, cadastrar novas salas e efetuar reservas de forma simples e intuitiva.

1.2 Público-alvo

- Universidades, empresas e coworkings que precisam gerenciar reservas de salas de reunião, estudo ou eventos.
- Usuários finais: estudantes, professores, funcionários e colaboradores.

1.3 Principais características

- Cadastro de salas (com nome, capacidade, descrição, equipamentos etc.).
- Reserva de salas (com data, horário e responsável).
- Listagem e visualização das salas e reservas.
- Cancelamento/exclusão de salas ou reservas.
- Interface web responsiva para desktop e mobile.

2. Problema

Muitas instituições e empresas enfrentam dificuldades em organizar a utilização de salas. Sem um sistema centralizado, há problemas como:

- Choque de reservas.
- Falta de visibilidade sobre disponibilidade.
- Burocracia para cadastros e aprovações.

3. Requisitos

3.1 Funcionais

RF01 - O sistema deve permitir autenticação de usuários via Supabase Auth.

Permite que usuários façam login e logout no sistema com segurança, garantindo que apenas pessoas cadastradas acessem as funcionalidades da plataforma.

RF02 - O sistema deve permitir cadastrar salas com nome, capacidade e recursos.

Possibilita que administradores registrem novas salas, incluindo informações como nome da sala, quantidade de pessoas suportadas e recursos (ex.: projetor, ar-condicionado, TV).

RF03 - O sistema deve listar todas as salas disponíveis.

Exibe para os usuários uma listagem completa das salas cadastradas, incluindo informações relevantes e disponibilidade para reserva.

RF04 - O sistema deve permitir excluir uma sala.

Permite que administradores removam do sistema salas que não estão mais disponíveis ou que foram desativadas.

RF05 - O sistema deve permitir cadastrar reservas associadas a uma sala e a um usuário.

Habilita os usuários autenticados a reservar uma sala específica em determinada data e horário, vinculando a reserva ao seu perfil.

RF06 - O sistema deve listar reservas existentes.

Mostra as reservas já cadastradas, permitindo que usuários e administradores visualizem detalhes como sala, horário e responsável pela reserva.

RF07 - O sistema deve permitir excluir/cancelar uma reserva.

Dá ao usuário ou administrador a possibilidade de cancelar uma reserva, liberando a sala para outros interessados.

RF08 - O sistema deve ser responsivo e acessível em desktop e dispositivos móveis.

Garante que a aplicação funcione corretamente em diferentes tamanhos de tela (computadores, tablets e celulares), mantendo uma boa usabilidade.

3.2 Não-Funcionais

RNF01 - O sistema deve utilizar angular no frontend.

Define a tecnologia principal para a interface do usuário, garantindo modularidade, performance e escalabilidade no frontend.

RNF02 - O backend deve ser implementado em C# (.NET 7).

Especifica a linguagem e framework para o desenvolvimento da lógica de negócio e comunicação com o banco de dados.

RNF03 - O banco de dados deve ser gerenciado via Supabase.

Determina que os dados (salas, reservas, usuários) serão armazenados e consultados utilizando Supabase, que oferece PostgreSQL como serviço.

RNF04 - O sistema deve utilizar autenticação segura com JWT (via Supabase).

As sessões dos usuários serão autenticadas e validadas usando tokens JWT, fornecidos pelo Supabase, para garantir segurança e controle de acesso.

RNF05 - O sistema deve seguir arquitetura Clean Architecture.

O backend será estruturado de forma desacoplada (separando domínio, aplicação, infraestrutura e apresentação), permitindo maior manutenção e escalabilidade.

RNF06 - O sistema deve possuir testes automatizados no backend.

Serão implementados testes unitários e de integração para validar regras de negócio e evitar regressões durante o desenvolvimento.

RNF07 - O sistema deve suportar pelo menos 500 usuários simultâneos.

O sistema deve ser escalável e performático para atender múltiplos acessos concorrentes sem perda de desempenho.

RNF08 - O sistema deve seguir padrões de acessibilidade web (WCAG 2.1).

A interface deve atender critérios de acessibilidade, permitindo que pessoas com deficiências visuais, motoras ou cognitivas utilizem a aplicação.

4. Casos de Uso

UC01 – Autenticação de Usuário

- Ator principal: Usuário
- Descrição: O usuário acessa a tela de login e realiza autenticação via Supabase Auth.
- Pré-condição: O usuário deve estar cadastrado.
- Fluxo principal:
  - Usuário acessa a tela de login.
  - Informa e-mail e senha.
  - O sistema valida as credenciais no Supabase.
  - O sistema gera token JWT e concede acesso.
- Fluxo alternativo: Caso as credenciais estejam incorretas, o sistema informa erro.
- Requisitos relacionados: RF01, RNF04

UC02 – Cadastrar Sala

- Ator principal: Administrador
- Descrição: O administrador cadastra uma nova sala informando nome, capacidade e recursos.
- Pré-condição: Usuário deve estar autenticado como administrador.
- Fluxo principal:
  - Administrador acessa a tela de cadastro de salas.
  - Preenche nome, capacidade e recursos.
  - Confirma o cadastro.
  - O sistema salva os dados no banco.
- Requisitos relacionados: RF02, RNF03

UC03 – Listar Salas

- Ator principal: Usuário
- Descrição: O usuário visualiza todas as salas cadastradas no sistema.
- Pré-condição: Usuário deve estar autenticado.
- Fluxo principal:

  - Usuário acessa a tela de listagem de salas.
  - O sistema exibe as salas com suas informações.
- Requisitos relacionados: RF03

UC04 – Excluir Sala

- Ator principal: Administrador
- Descrição: O administrador remove uma sala do sistema.
- Pré-condição: A sala deve existir.
- Fluxo principal:
  - Administrador seleciona uma sala.
  - Solicita exclusão.
  - O sistema confirma a exclusão.
- Requisitos relacionados: RF04

UC05 – Cadastrar Reserva

- Ator principal: Usuário autenticado
- Descrição: O usuário reserva uma sala informando data e horário.
- Pré-condição: A sala deve estar disponível no horário selecionado.
- Fluxo principal:
  - Usuário acessa a tela de reservas.
  - Seleciona uma sala.
  - Escolhe data e horário.
  - Confirma a reserva.
  - O sistema registra a reserva vinculada ao usuário.
- Requisitos relacionados: RF05

UC06 – Listar Reservas

- Ator principal: Usuário autenticado
- Descrição: O usuário visualiza todas as reservas feitas no sistema.
- Pré-condição: Deve haver reservas cadastradas.
- Fluxo principal:
  - Usuário acessa a tela de reservas.

  - O sistema exibe as reservas (usuário comum vê as próprias; administrador vê todas).
- Requisitos relacionados: RF06

UC07 – Cancelar Reserva

- Ator principal: Usuário autenticado / Administrador
- Descrição: O usuário cancela uma reserva existente.
- Pré-condição: A reserva deve estar cadastrada.
- Fluxo principal:
  - Usuário seleciona reserva.
  - Solicita cancelamento.
  - O sistema exclui a reserva e libera a sala.
- Requisitos relacionados: RF07

UC08 – Usabilidade Responsiva

- Ator principal: Usuário
- Descrição: O sistema deve estar disponível em qualquer dispositivo (desktop/mobile).
- Pré-condição: O usuário acessa via navegador ou dispositivo móvel.
- Fluxo principal:
  - Usuário abre a aplicação.
  - O sistema adapta a interface conforme o dispositivo.
- Requisitos relacionados: RF08, RNF08

5. Planejamento Inicial

4.1 Roadmap

Versão	Funcionalidades
0.1	Setup do projeto, autenticação com Supabase Auth, criação inicial de banco (usuários, salas, reservas).
0.2	Cadastro, listagem e exclusão de salas.
0.3	Cadastro, listagem e cancelamento de reservas.
0.4	Painel administrativo (gestão avançada de salas e reservas).
0.5	Melhorias de UX, acessibilidade, relatórios e testes automatizados.

4.2 Backlog Inicial

- Como usuário, quero me autenticar com login e senha, para acessar a plataforma.
- Como administrador, quero cadastrar salas para disponibilizar no sistema.
- Como usuário, quero visualizar salas disponíveis para reservas.
- Como usuário, quero reservar uma sala para determinado horário.
- Como administrador, quero excluir salas e reservas obsoletas.

6. Critérios de Aceite

- Usuário só pode reservar uma sala se estiver autenticado.
- Não deve ser possível reservar uma sala já ocupada no mesmo horário.
- O sistema deve impedir o cadastro de salas com nomes duplicados.
- O sistema deve permitir cancelamento de reservas apenas pelo usuário responsável ou administrador.

7. Rastreabilidade

Com a rastreabilidade temos um mapa rastreável e consistente, onde conseguimos ver:

- O requisito que originou,
- O caso de uso que o representa,
- A entidade do banco que o implementa.

ID	REQUISITO	CASO DE USO	ENTIDADE
RF01	Autenticação de usuários via Supabase Auth	UC01 – Autenticação de Usuário	**Usuário**
RF02	Cadastrar salas (nome, capacidade, recursos)	UC02 – Cadastrar Sala	**Sala**
RF03	Listar todas as salas	UC03 – Listar Salas	**Sala**
RF04	Excluir sala	UC04 – Excluir Sala	**Sala**
RF05	Cadastrar reservas (sala + usuário)	UC05 – Cadastrar Reserva	**Reserva**
RF06	Listar reservas existentes	UC06 – Listar Reservas	**Reserva**, **Sala**, **Usuário**
RF07	Cancelar reserva	UC07 – Cancelar Reserva	**Reserva**
RF08	Responsividade (desktop e mobile)	UC08 – Usabilidade Responsiva	-
RNF01	Uso de Angular no frontend	-	-
RNF02	Backend em C# (.NET 7)	-	-
RNF03	Banco via Supabase	-	**Usuário**, **Sala**, **Reserva**

RNF04	Autenticação segura com JWT (Supabase)	UC01	**Usuário**
RNF05	Arquitetura Clean Architecture	-	-
RNF06	Testes automatizados backend	-	-
RNF07	Suportar 500 usuários simultâneos	-	-
RNF08	Acessibilidade (WCAG 2.1)	UC08	-

8. Nomeação de Branchs e Commits - Padronização do processo de implantação

8.1 Fluxo de branches

Inicialmente temos 2 branchs fixas que serão utilizadas para centralização de cada feature e para que todo ciclo de desenvolvimento possa ser validado com segurança:

Branchs fixas

Development: Conterá todo o código que será publicado em ambiente de desenvolvimento para testes de novas funcionalidades ou correções de bugs.

Main: Nessa branch terá a última versão válida do código da aplicação, que é a mesma versão que esta publicada em ambiente de produção.

Branchs temporárias

A partir dessas três branches, temos as branches temporárias criadas conforme demanda e necessidade, entre elas temos:

Feature: Esta branch tem o intuito de ser utilizada somente para desenvolvimento da feature, criada a partir de desenvolvimento. Assim que o desenvolvido é concluído deve ser realizado o merge com desenvolvimento novamente.

Bugfix: Criada corrigir bugs e falhas ocorridas em uma determinada funcionalidade em ambiente de desenvolvimento, criada a partir da development.

Exemplos de branches:

bugfix/ erro-listagem-salas

feature/criação-sala

É aconselhado que essas branches temporárias sejam excluídas assim que todo o fluxo de deploy para produção e seus respectivos merges sejam concluídos com êxito, dessa forma será mantido um fluxo limpo e organizado para os próximos desenvolvimentos.

8.2 Políticas de Branches

Para garantirmos a integridade da branch de development e a main, podemos adicionar políticas de branch para garantir que nenhum desenvolvedor edite o código diretamente nessas duas branches.

Dessa forma, é necessário que uma branch a partir da development gere um pull request para que qualquer alteração seja realizada. Dessa mesma forma também é necessário que o Pull Request seja validado por menos um outro desenvolvedor para que assim que aprovado as alterações possam ser mergeadas com a branch de destino.

Uma observação importante é que para merge na branch master é necessário dois reviewer e para a branch development apenas um, reforçando assim a prática de code review.

Segue o fluxo das políticas de branches:

Branch Padrão:

· main como branch principal.

Proteção da Branch Principal:

· Restrições para alterações diretas; uso de pull requests para mesclar alterações.

Revisão de Código:

· Necessidade de revisão de código para todas as alterações propostas.

Padrões de Nomenclatura:

· Utilização de padrões para nomenclatura de branches, como feat/ e hotfix/.

Remoção de Branches Antigas:

· Exclusão de branches temporárias após conclusão do ciclo de deploy.

8.3 Padrões de Commit

A adoção de padrões de commit é fundamental para manter a clareza e consistência no histórico de alterações do repositório. O padrão de commit semântico proposto visa fornecer informações valiosas sobre a intenção de cada alteração realizada no código.

feat - Inclusão de um novo recurso. Este tipo de commit está associado ao conceito de MINOR no versionamento semântico.

fix - Alteração resolve um problema específico (bug fix). Esse tipo de commit está associado ao PATCH no versionamento semântico.

test - Alterações nos testes, incluindo criação, modificação ou exclusão de testes unitários.

docs - Alterações na documentação, como atualizações no README. Este tipo de commit não inclui modificações no código-fonte.

Exemplos:

feat: Adicionar funcionalidade de pesquisa por sala

fix: Corrigir erro de criação de reserva

test: Adicionar teste unitário para validar a autenticação de usuário

docs: Atualização da documentação do projeto

8.4 Gitflow

No projeto existem 2 ramificações fundamentais:

Development - Contém a versão mais recente da aplicação. Os desenvolvedores integram suas conclusões de trabalho primeiramente nessa branch para que possam ser realizados os testes.

main - Nessa branch temos a versão que está liberada para utilização dos usuários. A integração nela vem após validação completa do que foi preparado na development.

9. Banco de dados

O modelo de dados proposto contempla três entidades principais: Usuário, Sala e Reserva, interligadas de acordo com as regras de negócio do sistema de gerenciamento de reservas.

Usuario

Representa os usuários autenticados no sistema. Cada usuário possui atributos como ID_usuario, nome, email e data_cadastro.

Sala

Representa as salas disponíveis para reserva. Inclui atributos como ID_sala, nome_sala, descricao, status_sala, data_criacao e capacidade.

Reserva

Central do sistema, armazena os agendamentos realizados pelos usuários. Contém atributos como ID_reserva, data_inicio, data_fim, status_reserva, data_criacao, motivo_reserva, horario_inicio, horario_fim, publico_externo, quantidade_pessoas e campos de relacionamento com Usuário e Sala.

9.1 Cardinalidades

Usuário (1,1) → (0,n) Reserva

Cada usuário pode realizar várias reservas, mas cada reserva pertence a um único usuário.

Sala (1,1) → (0,n) Reserva

Uma sala pode estar associada a diversas reservas ao longo do tempo, mas cada reserva ocorre em apenas uma sala.

9.2 Aderência aos Requisitos

O modelo de dados atende plenamente aos requisitos definidos para o sistema:

1. Autenticação e Controle de Usuários
   - A entidade usuario está preparada para ser integrada com o sistema de autenticação do Supabase (via JWT).
   - Cada usuário possui identificação única e dados de cadastro.
2. Gestão de Salas
   - A entidade sala armazena informações completas das salas, incluindo nome, descrição, status e capacidade, permitindo o controle da disponibilidade.
3. Gestão de Reservas
   - A entidade reserva vincula usuários às salas, armazenando data, horários, motivo da reserva, público externo e quantidade de pessoas.
   - O campo status_reserva garante o acompanhamento do ciclo de vida da reserva (pendente, confirmada, cancelada).
4. Integridade dos Relacionamentos
   - O uso de chaves estrangeiras (ID_usuario e ID_sala) assegura que toda reserva esteja sempre associada a um usuário válido e a uma sala existente.

10. Lean Inception

O modelo foi desenvolvido e está disponível no link: https://miro.com/welcomeonboard/UlpKSlM4amEyOWZqSDRNVEFXTHYrNVA5QnUrdzRXc0NIR2RIN0pOM2syODdTeXZwa2Z0VXJIZ2t6U2treE5idGpuTGkydUFqZ2

VoR0duL3dBbkN1SUxCcjZKZzNkSG1kVFFYSnhwVmVOTHVrcHo3ZjBTYjhtUTB
UVHdGS2p6ZDJ0R2lncW1vRmFBVnlLcVJzTmdFdlNRPT0hdjE=?share_link_id=48
813599472

11. Prototipação

A prototipação do está disponível no link: https://www.figma.com/design/iLDpgKgKcTz6mfm6TBL5y0/Space-Now?node-id=0-1&m=dev