📝 Product Backlog - Sistema de Visualização de Horários

**Épico:** Sistema Web para Gerenciamento e Consulta de Horários e Visualização de Ambientes Acadêmicos

**História 1 - Prototipação (Figma)**

Como time de desenvolvimento,

Quero ter uma prototipagem validada no Figma,

Para garantir uma interface intuitiva antes da codificação.

**Tarefas:**

* Criar Wire frames detalhados para as principais páginas do sistema no Figma.
* Definição de Cor primária e secundária do projeto.
* Validar com o cliente/usuário.
* Refinar até aprovação final.

**Prioridade: Alta**

**Critérios de Aceite:**

* Wire frames contemplam todas as funcionalidades solicitadas.
* Todos os fluxos principais do sistema são representados no protótipo.
* Aprovar o protótipo antes do desenvolvimento do front-end.

**História 2 – Desenvolvimento do Front-End- Design e Responsividade**

Como usuário,

Quero que o sistema funcione bem em dispositivos móveis e desktop,

Para ter uma boa experiência em qualquer contexto.

**Tarefas:**

* Criar as telas transformando layout do figma em código.
* Garantir layout adaptável (CSS puro, sem frameworks).
* Ajustar elementos para telas menores.
* Testes em dispositivos diferentes.

**Prioridade: Alta**

**Critérios de Aceite:**

* Sistema é utilizável em smartphones e tablets.
* Layout não quebra em diferentes tamanhos de tela.
* Sistema navega entre as telas corretamente.

**História 3– Desenvolvimento do Front-End - Grade de Horários**

RNF01

Como secretaria acadêmica,

Quero que o sistema possua uma grade de horários no qual exibe

O semestre, curso, período, disciplinas, docentes e ambientesque serão consumidos do banco de dados.

**Tarefas:**

* Garantir layout adaptável (CSS puro, sem frameworks).
* Endpoint de consulta no backend
* Tela de manutenção da grade.
* Criar interface de filtro e busca.
* Tela de somente exibição da grade.
* Exibir resultados em formato de tabela responsiva.
* Ajustar elementos para telas menores.
* Testes em dispositivos diferentes.

**Prioridade: Alta**

**Critérios de Aceite:**

* Filtros por turma, professor, turno e ambiente funcionam corretamente
* Resultados atualizados dinamicamente
* Grade é utilizável em smartphones e tablets.
* Layout não quebra em diferentes tamanhos de tela.
* Sistema navega entre os diferentes mapas corretamente.

**História 4– Desenvolvimento do Front-End - Mapa Interativo**

RNF01

Como secretaria acadêmica,

Quero que o sistema possua um mapa interativo no qual exibe,

Mapa de salas com a opção interativa de clicar e visualizar a situação

Do ambiente (ex. aulas alocadas ao longo dos turnos do dia).

**Tarefas:**

* Garantir layout adaptável (CSS puro, sem frameworks).
* Criar interface de filtro e busca
* Endpoint de consulta no backend
* Tela de manutenção do mapa.
* Exibir resultados em formato de mapa responsivo
* Tela de somente exibição do mapa.
* Ajustar elementos para telas menores.
* Testes em dispositivos diferentes.

**Prioridade: Alta**

**Critérios de Aceite:**

* Filtros por turma, professor, turno e ambiente funcionam corretamente
* Resultados atualizados dinamicamente
* Mapa é utilizável em smartphones e tablets.
* Layout não quebra em diferentes tamanhos de tela.
* Sistema navega entre os diferentes mapas corretamente.

**História 5 – Desenvolvimento do Front-End - Usabilidade**

Como desenvolvedor Front-End,

Quero que o sistema se comunique com o Back-End e esteja bem estruturado,

Para consumir os dados do banco e tratá-los para exibir ao usuário.

**Tarefas:**

* Desenvolver lógicas para consumir os dados coletados nos end-points da API.
* Aplicar métodos de segurança no front-end.

**Prioridade: Alta**

**Critérios de Aceite:**

* Sistema com lógicas para consumo dos end-points funcionando.
* Sistema com validações de segurança mínimas.

**História 6- Interatividade de Ambientes**

RNF01

Como usuário,

Quero ver um mapa das salas e clicar em cada sala para ver as alocações do dia,

Para ter uma visão clara e visual da ocupação da faculdade.

**Tarefas:**

* Implementar interação via click (mostrar horários).
* Exibir status (livre/ocupada) no mapa.
* Configurar o mapa para receber informações do back-end para a interação.

**Prioridade: Alta**

**Critérios de Aceite:**

* Mapa exibe status visual de ocupação.
* Clique em cada ambiente abre painel com horários detalhados.
* Mapa está consumindo as informações do back-end corretamente.

**História 7 - Estrutura de Pastas do Front-End**

Como desenvolvedor,

Quero criar uma estrutura de pastas organizada no projeto do front-end,

Para facilitar o desenvolvimento, manutenção e escalabilidade do código.

Tarefas:

* Definir a estrutura inicial das pastas:
  1. Criar diretórios principais para componentes, páginas, estilos, serviços, imagens e utilitários.
  2. Seguir uma abordagem modular para isolar responsabilidades.
* Configurar o diretório de componentes:
  1. Criar subpastas para cada componente reutilizável, com arquivos para o código (.js ou .tsx) e estilos (.css ou .scss).
* Organizar o diretório de páginas:
  1. Criar subpastas para cada página principal do sistema.
  2. Garantir que cada página tenha seus próprios arquivos específicos (ex.: layout, lógica e testes).
* Criar estrutura para serviços:
  1. Configurar um diretório para serviços de API, com arquivos separados para cada recurso (ex.: authService.js, userService.js).
* Configurar o diretório de estilos globais:
  1. Criar um arquivo de estilos globais (ex.: global.css ou main.scss).
  2. Criar subpastas para variáveis de tema, mixins e reset de estilos, se aplicável.
* Criar um diretório para assets:
  1. Adicionar pastas para imagens, ícones e outros recursos estáticos.
* Configurar estrutura para utilitários:
  1. Criar um diretório para funções ou classes reutilizáveis (ex.: formatação de datas, validação de entradas).

Critérios de Aceite:

1. Estrutura de pastas está organizada, seguindo boas práticas para projetos front-end.
2. Cada funcionalidade principal (componentes, páginas, estilos, serviços, utilitários) possui um local definido na estrutura.
3. Arquivos e diretórios estão nomeados de forma clara e padronizada.
4. O projeto é escalável, permitindo a adição de novos componentes, páginas e recursos sem comprometer a organização.

**História 8- Definição de estrutura BackEnd**

Como time de desenvolvimento,

Quero ter uma estrutura de como vai ser o projeto do servidor,

Para garantir um código organizado e robustopara fácil entendimento e manutenção.

**Tarefas:**

* Definir o design do projeto (Estrutura das Pastas).
* Escolher a Stack do back-end (ex.: Node.js com Express).
* Configurar o ambiente inicial de desenvolvimento.
* Configurar variáveis de ambiente (ex.: dotenv).
* Configurar um servidor básico que responde "OK" na rota raiz.

**Prioridade: Alta**

**Critérios de Aceite:**

* Tecnologias bem definidas.
* Design do projeto organizado e estrutura coesa.
* O projeto está configurado com as ferramentas escolhidas e o ambiente é executável.
* A rota raiz (GET/) responde com status 200 e uma mensagem de confirmação.
* Estrutura de pastas está organizada para suportar escalabilidade.

**História 9- Banco de Dados**

Como desenvolvedor,

Quero modelar e configurar o banco de dados,

Para armazenar os dados necessários ao sistema de forma eficiente e organizada.

**Tarefas:**

* Escolher o banco de dados relacional (PostgreSQL).
* Modelar e criar as tabelas iniciais:
  1. Semestre
  2. Cursos
  3. Período (Horários e dias)
  4. Disciplinas
  5. Docentes
  6. Ambientes
* Adicionar migrações para facilitar mudanças futuras no banco.
* Testar a conexão com o banco e garantir que as tabelas foram criadas corretamente.

**Prioridade: Alta**

**Critérios de Aceite:**

* Banco de dados configurado e acessível pelo back-end.
* Tabelas necessárias criadas conforme especificado na modelagem.
* Conexão testada e funcional.

**História 10- Endpoints CRUD**

RF02

Como desenvolvedor Back-End,

Quero criar endpoints para operações CRUD de semestres, cursos, períodos (horários e dias), matérias, docentes e ambientes,

Para permitir a manipulação desses dados pelo sistema.

**Tarefas:**

* Criar rotas para cada entidade:
  1. GET /entidade (Listar)
  2. POST /entidade (Criar)
  3. PUT /entidade/:id (Atualizar)
  4. DELETE /entidade/:id (Excluir)
* Implementar controladores para cada operação.
* Validar campos obrigatórios antes de salvar ou atualizar registros.
* Retornar mensagens de erro e códigos de status apropriados.

**Prioridade: Alta**

**Critérios de Aceite:**

* Endpoints CRUD estão implementados e integrados ao banco de dados.
* Operações retornam respostas adequadas (ex.: 201 para criação, 400 para erros de validação).
* Validações de dados garantem integridade mínima das informações.

**História 11- Exportação em PDF**

RF04

Como secretaria acadêmica,

Quero gerar um PDF com os horários e alocações,

Para facilitar a impressão ou o envio digital do cronograma.

**Tarefas:**

* Criar endpoint de geração de PDF no backend
* Definir layout do PDF (mapa e tabelas)
* Botão de exportar na interface de consulta

**Prioridade: Média**

**Critérios de Aceite:**

* PDF gerado contém todas as informações necessárias
* Layout legível e pronto para impressão

**História 12- Upload de CSV e Processamento**

RF01

Como secretaria acadêmica,

Quero fazer upload de arquivos CSV com dados do sistema,

Para alimentar rapidamente o banco de dados com informações em massa

**Tarefas:**

* Criar endpoint para upload de arquivos CSV.
* Implementar lógica de leitura e validação de arquivos CSV:
  1. Verificar formato do arquivo.
  2. Validar campos obrigatórios (ex.: semestre, periodo).
* Salvar dados válidos no banco de dados.
* Retornar feedback ao usuário:
  1. Quantidade de registros processados com sucesso.
  2. Lista de erros encontrados.

**Prioridade: Média**

**Critérios de Aceite:**

* O sistema aceita arquivos CSV com estrutura válida.
* Dados válidos são salvos no banco; inválidos são listados como erros.
* Feedback detalhado sobre o processamento é exibido.

**História 13 - Validação de Regras de Negócio**

RF03

Como secretaria acadêmica,

Quero validar as regras de alocação de horários,

Para evitar conflitos como professores em dois lugares ao mesmo tempo.

**Tarefas:**

* Desenvolver lógica de validação para:
  1. Conflito de horários de professores.
  2. Duplicidade de ambiente.
  3. Horários sobrepostos para a mesma turma.
* Implementar sistema para listar pendências identificadas.
* Salvar parcialmente dados válidos e marcar pendências para correção posterior.

**Prioridade: Alta**

**Critérios de Aceite:**

* Sistema bloqueia alocações inválidas e lista erros para revisão.
* Pendências são salvas no sistema para correção futura.
* Dados válidos são armazenados sem interrupções.

**História 14 - Segurança**

Como desenvolvedor,

Quero implementar autenticação e autorização no sistema,

Para proteger endpoints e garantir que apenas usuários autorizados tenham acesso.

**Tarefas:**

* Configurar autenticação utilizando JWT.
* Implementar middleware de autorização para proteger rotas sensíveis.
* Criar sistema de roles (ex.: administrador, usuário padrão) para restringir ações específicas.
* Adicionar controle de sessões para expiração de tokens.

**Prioridade: Alta**

**Critérios de Aceite:**

* Endpoints protegidos por autenticação e autorização.
* Tokens JWT são gerados e validados corretamente.
* Ações são restritas de acordo com o nível de permissão do usuário.

**História 15 – API Restful**

**Como desenvolvedor**, eu preciso implementar uma API RESTful para nossa

plataforma de gerenciamento de usuários, para que os sistemas front-end e

aplicativos móveis possam consumir dados de forma padronizada e segura.

**Tarefas:**

* Definir os endpoints para operações CRUD de usuários
* Implementar o padrão de requisições HTTP (GET, POST, PUT, DELETE)
* Desenvolver a estrutura de resposta em formato JSON
* Configurar autenticação via token JWT
* Implementar tratamento de erros e códigos de status HTTP adequados
* Realizar testes de integração dos endpoints

**Prioridade: Alta**

**Critérios de Aceite:**

* A aplicação deve estar acessível através de uma URL pública fornecida pelo Render
* O processo de build deve ser concluído sem erros
* A aplicação deve carregar corretamente e estar funcional em diferentes navegadores
* O tempo de carregamento inicial não deve exceder 3 segundos
* As variáveis de ambiente devem estar corretamente configuradas no ambiente de produção
* O deploy automático deve funcionar corretamente após cada push na branch principal
* A documentação do processo de deploy deve ser clara e detalhada
* O sistema deve manter-se dentro dos limites gratuitos do plano do Render

**História 16 – Deploy Front-End**

**Como desenvolvedor**, eu preciso realizar o deploy da aplicação front-end em um

ambiente de produção, para disponibilizar o sistema aos usuários finais de forma

segura e eficiente.  
  
**Tarefas:**

* Preparar o projeto para build de produção
* Configurar variáveis de ambiente para o ambiente de produção
* Otimizar assets (imagens, CSS, JavaScript) para melhor performance
* Realizar configuração de HTTPS para comunicação segura
* Implementar monitoramento e alertas de disponibilidade
* Configurar pipeline de CI/CD para automatizar o processo de deploy

**Prioridade: Alta**

**Critérios de Aceite:**

* A aplicação deve estar acessível através de uma URL pública
* O processo de build deve ser concluído sem erros
* A aplicação deve carregar corretamente e estar funcional em diferentes navegadores (Chrome, Firefox, Safari, Edge)
* O tempo de carregamento inicial não deve exceder 3 segundos
* O certificado SSL deve estar válido e implementado corretamente
* A aplicação deve manter pelo menos 99.5% de disponibilidade
* O pipeline de CI/CD deve realizar deploy automaticamente após aprovação

**História 17- Deploy Back-End**

**Como desenvolvedor**, eu preciso realizar o deploy da aplicação back-end em um

ambiente de produção, para disponibilizar a API e serviços necessários para o

funcionamento completo do sistema.

**Tarefas:**

* Configurar ambiente de produção no servidor/cloud
* Preparar o projeto para build de produção
* Configurar variáveis de ambiente para produção
* Implementar conexão com o banco de dados de produção
* Implementar monitoramento de recursos e performance
* Configurar backup automático do banco de dados
* Implementar pipeline de CI/CD para automatizar o deploy
* Configurar HTTPS e certificados SSL

**Prioridade: Alta**

**Critérios de Aceite:**

* O serviço back-end deve estar acessível através de endpoint seguro
* As conexões com o banco de dados devem ser estabelecidas corretamente
* Todas as variáveis de ambiente devem estar configuradas adequadamente
* Os certificados SSL devem estar válidos e implementados corretamente
* O pipeline de CI/CD deve realizar deploy automaticamente após aprovação
* Medidas de segurança contra ataques comuns (DDoS, SQL Injection) devem estar implementadas

**História 18- Deploy Banco de Dados**

Como desenvolvedor, eu preciso realizar o deploy do banco de dados em ambiente

de produção, para garantir a persistência e integridade dos dados da aplicação de

forma segura e eficiente.

**Tarefas:**

* Configurar servidor/instância de banco de dados no ambiente de produção
* Criar scripts de migração para estrutura do banco de dados
* Implementar estratégia de versionamento do esquema do banco
* Configurar usuários e permissões com privilégios mínimos necessários
* Implementar mecanismos de backup e recuperação automatizados
* Configurar replicação para alta disponibilidade
* Otimizar índices e consultas para melhor performance
* Implementar monitoramento de performance e uso de recursos
* Configurar firewall e regras de acesso ao banco de dados
* Documentar estrutura do banco e procedimentos de manutenção
* Criar scripts de seed para dados iniciais necessários
* Implementar estratégia de rotação de logs

**Prioridade: Alta**

**Critérios de Aceite:**

* O banco de dados deve estar acessível apenas pela aplicação e através de canais seguros
* As migrações devem ser executadas corretamente sem perda de dados
* O tempo de resposta para consultas comuns deve ser inferior a 200ms
* O sistema de backup deve realizar cópias completas diariamente e incrementais a cada 6 horas
* O processo de recuperação deve ser testado e documentado
* Devem existir alarmes configurados para uso de CPU, memória e espaço em disco
* A replicação deve estar funcionando com latência máxima de 5 segundos
* Os logs do banco de dados devem ser armazenados por pelo menos 30 dias
* O acesso ao banco deve ser restrito à rede interna ou VPN
* O banco de dados deve suportar no mínimo 100 conexões simultâneas
* A documentação deve incluir diagrama ER atualizado e descrição das tabelas principais
* Todos os dados sensíveis devem estar criptografados em repouso