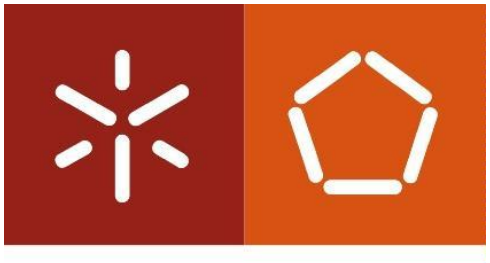


Interface Pessoa-Máquina

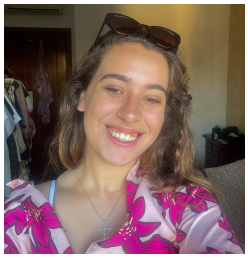
Relatório Fase Final
Plataforma de Turnos
Grupo 19

LEI - 3º Ano - 2º Semestre
Ano Letivo 2024/2025

Tiago Guedes
A97369

Diogo Goncalves
A101919

Diogo Pinto
A100551



Mariana Pinto
A100756

Braga,
4 de maio de 2025

Link do Protótipo: `https://www.figma.com/design/HtgBE8xL32uNmZ8CriRIBT/IPM-Mockup?node-id=203-78&t=cbhQmrOVBYeqO6c1-1`

Conteúdo

Lista de Figuras

1 Objetivos

O objetivo principal deste trabalho foi desenvolver uma aplicação web para a gestão de alunos e os seus turnos num âmbito de um curso. A aplicação foi projetada para atender às necessidades de dois tipos de utilizadores: alunos e diretor do curso respetivo.

Entre as funcionalidades implementadas estão:

- Visualização e gestão de turnos e horários.
- Submissão e gestão de pedidos de alteração de turno/sala.
- Alocação de alunos a turnos.
- Alocação de turnos a salas.
- Notificações automáticas para alterações de horários.

2 Alterações

Durante a implementação, algumas alterações foram realizadas no protótipo inicial devido a questões de design e implementação:

- **Alteração 1:** Perfil sofreu alterações na forma como mostramos a informação, tanto do aluno como do diretor. Sendo a informação a mesma, apenas sofreu alterações de layout. *Justificação:* Consideramos alterar pois fica mais simples, minimalista e responsivo da forma que se encontra na aplicação agora, sendo o perfil uma página simples, achamos fazer todo o sentido a mudança.

- **Alteração 2:** Alteração da barra lateral principal também em ambas as visões apenas no botão de logout e uma identificação de como que entidade está o utilizador autenticado. *Justificação:* Apenas por considerar mais apresentável e de acordo com o resto da aplicação.

- **Alteração 3:** Na página de detalhe dos alunos na visão de diretor de curso foi introduzido um efeito caso o aluno tenha algum conflito ou pedido pendente.
  Justificação: Garantir que é mais visual para o diretor de curso de forma a que seja mais fácil a identificação do conflito/pedido.

- **Alteração 4:** Foi introduzida uma nova página na visão de aluno para que possa acompanhar os pedidos que foram submetidos ao diretor de curso.
  Justificação: Foi nos recomendado no prototipo e achamos fazer todo o sentido devido ao facto de o aluno poder saber em que estado se encontra cada pedido que efetuou individualmente.

- **Alteração 5:** Foi alterada a página de UC's na visão de aluno, e também o popup que é ativado nessa mesma página quando se pressiona o botão da UC específica. *Justificação:* Achamos pertinente mudar visto que da forma que se encontra de momento é muito mais percetível, com mais informação acerca do turno e mais responsivo do que estava anteriormente no prototipo.

3 Implementação

O sistema «gestãoalunos» foi desenvolvido em **Vue.js** recorrendo à *Composition API*. A aplicação disponibiliza duas perspetivas principais:

- **Diretor**, com interfaces para gestão de alunos e turnos;
- **Aluno**, com visualização de horário e gestão de pedidos.

A estrutura de pastas está organizada de forma a garantir clareza e facilidade de manutenção:

```
src/
|- components/       % Componentes genéricos e reutilizáveis
|- views/            % Páginas específicas por perfil
|  |- director/      % Views do diretor
|  \- student/       % Views do aluno
|- router/           % Definição de rotas e guardas de acesso
|- services/         % Wrappers para chamadas HTTP (Axios)
\- main.js           % Inicialização da aplicação e plugins
```

3.1 Decomposição em Componentes

A arquitetura foi concebida para maximizar a reutilização de código e manter as *views* focadas na lógica de negócio:

3.1.1 Layout

- `AppHeader.vue`, `AppSidebar.vue`: responsáveis pela navegação e identidade visual.

3.1.2 Controlo de Formulários

- `Checkbox.vue`, `ToggleSwitch.vue`: encapsulam o estilo e o comportamento de *inputs* personalizados.

3.1.3 Modals

- `ConfirmationModal.vue`, `SuccessModal.vue`, `ErrorModal.vue`, `LoadingState.vue`: utilizados transversalmente para confirmações, notificações de sucesso, mensagens de erro e loadings de paginas.

3.1.4 Botões de Ação

- `NotificationButton.vue`, `LogoutButton.vue`, `NewRequestModal.vue`, `ProfileInfo.vue`: encapsulam interações comuns e apresentação de informações do utilizador.

3.1.5 Views por Perfil

- `director/StudentsView.vue` e `director/StudentDetailsView.vue`: listagem e detalhe de alunos, com filtros e paginação.
- `student/ScheduleView.vue`, `student/UCsView.vue`: visualização do horário, unidades curriculares e formulários de pedido de troca de turno.

3.2 Serviços e Integração HTTP

As interações com a API utilizam **Axios**, configurado em `services/apiServices.js`:

- **URL base** centralizada;
- **Interceptadores**: injeção do token JWT e tratamento global de erros;
- Funções dedicadas por recurso (p.ex., `getStudents()`, `updateShift()`).

3.3 Mapeamento e Proteção de Rotas

Utiliza-se **vue-router** com:

- Definição de rotas em `router/index.js`, incluindo metadados (`requiresAuth`, `role`);
- Guardas globais (`router.beforeEach`) para validação do token e perfil do respetivo utilizador.

3.4 Estado Reativo e Ciclo de Vida

Recorre-se à *Composition API* para gerir estado e ciclos de vida:

- `ref`, `computed`: variáveis reativas e valores derivados;
- `onMounted`, `watch`: carregamento inicial de dados e reatividade a alterações de parâmetros;
- `defineProps`, `defineEmits`: comunicação entre componentes pai e filho.

Esta abordagem modular proporciona legibilidade, manutenção facilitada, separação clara de responsabilidades e uma base sólida para futuras extensões.

4 Reflexão sobre a Aplicação

A aplicação desenvolvida para a gestão de alunos procura facilitar a relação entre estudantes e diretores de curso, oferecendo um conjunto de funcionalidades pensadas para responder às necessidades de cada perfil. Nesta análise, destacamos os principais pontos fortes e fracos da aplicação do ponto de vista do utilizador.

4.1 Pontos Fortes

4.1.1 Arquitetura Modular e Reutilizável

A aplicação foi construída com uma arquitetura modular, recorrendo a componentes reutilizáveis. Isto traz várias vantagens:

- **Consistência na Interface**: O uso de componentes como modals de erro, sucesso, confirmação e loading de paginas garante que a experiência é uniforme em toda a aplicação. O utilizador reconhece facilmente os padrões de interação, independentemente da tarefa.
- **Desempenho Otimizado**: Graças a um loading dinâmico de componentes, apenas o necessário é carregado em cada momento, o que resulta em tempos de espera mais reduzidos e numa navegação mais fluida.
- **Transições Suaves**: As mudanças entre diferentes estados (loading, sucesso, erro) são tratadas de forma coerente, tornando a experiência mais previsível e agradável.

4.1.2 Interface Adaptada a Cada Perfil

A aplicação distingue claramente entre alunos e diretores de curso, apresentando interfaces ajustadas a cada um:

- **Perfil de Aluno**: Focado nas tarefas mais comuns, como consultar o horário, gerir pedidos de troca de turno e aceder às informações das disciplinas. A navegação é simples e direta.
- **Perfil de Diretor**: Com acesso a funcionalidades mais avançadas, como a gestão de alunos, turnos, salas e aprovação de pedidos. A interface é mais rica e orientada para tarefas administrativas.
- **Rotas Separadas**: O sistema de navegação garante que cada utilizador só vê as funcionalidades relevantes ao seu papel, evitando confusões e acessos indevidos.

4.1.3 Gestão Eficiente de Processos Complexos

A aplicação lida de forma eficaz com processos que, por natureza, podem ser complexos:

- **Gestão de Conflitos de Horário**: Os conflitos são detetados automaticamente e apresentados de forma visual, ajudando o utilizador a tomar decisões informadas.
- **Pedidos de Alteração**: Todo o ciclo de vida dos pedidos (criação, consulta, aprovação ou rejeição) é claro, com feedback visual em cada etapa.
- **Proteção de Acesso**: O sistema assegura que cada utilizador só acede ao que lhe é permitido, prevenindo erros e acessos não autorizados.

4.1.4 Design Responsivo e Experiência de Utilizador

A aplicação foi desenhada com atenção à usabilidade:

- **Carregamento Condicional**: Os componentes só são carregados quando necessários, o que melhora o desempenho.
- **Feedback Visual**: O uso de modals e notificações torna o sistema mais transparente, informando o utilizador sobre o estado das operações.
- **Gestão de Estados de Carregamento**: O utilizador é sempre informado quando a aplicação está a processar dados, evitando incertezas.

4.2 Pontos Fracos

4.2.1 Complexidade em Alguns Fluxos

Apesar da modularidade, há situações onde a experiência pode ser menos intuitiva:

- **Navegação Profunda**: Algumas rotas, como detalhes de turnos e salas, podem obrigar o utilizador a navegar por vários níveis, tornando certas tarefas mais demoradas.
- **Várias Confirmações**: Em determinadas operações, são apresentadas várias caixas de diálogo, o que pode tornar o processo repetitivo e cansativo.
- **Excesso de Informação**: Algumas páginas apresentam demasiada informação, o que pode ser confuso, sobretudo para utilizadores ainda não habituados 'a aplicação.

4.2.2 Dependência do Armazenamento Local para Autenticação

A autenticação baseia-se no `localStorage`, o que traz algumas limitações:

- **Segurança Reduzida**: Guardar dados de autenticação no `localStorage` não é a opção mais segura, estando vulnerável a ataques.
- **Ausência de Tokens**: Não é utilizado nenhum sistema de tokens (como JWT), limitando a segurança e a gestão de sessões.
- **Sessão Sem Expiração**: Não existe um mecanismo claro de expiração automática da sessão, o que pode ser problemático se o utilizador se esquecer de terminar sessão.

4.2.3 Gestão de Erros

A aplicação ainda tem margem para melhorias nestes aspetos:

- **Gestão de Erros Limitada**: Apesar de existir um modal de erro, não são tratados erros de rede, por exemplo.
- **Falta de Retentativas**: Em caso de erro, o utilizador, em alguns casos, tem de repetir todo o processo, em vez de apenas tentar novamente a operação que falhou.

4.2.4 Acessibilidade

Existem algumas limitações ao nível da acessibilidade:

- **Sem Atalhos de Teclado**: Não há atalhos para facilitar a navegação, o que pode dificultar a vida a utilizadores com necessidades especiais.
- **Dependência Visual**: A interface depende fortemente de elementos visuais, o que dificulta a utilização por pessoas com deficiência visual.
- **Pouca Personalização**: Não há opções evidentes para personalizar o aspeto da interface, como mudar para modo escuro ou aumentar o tamanho do texto.

A aplicação de Gestão de Alunos apresenta uma estrutura sólida e uma interface bem pensada, adequada tanto para alunos como para diretores de curso. A modularidade, a separação clara de funcionalidades e o feedback visual contribuem para uma boa experiência de utilização.
Ainda assim, há espaço para melhorias, nomeadamente na simplificação de alguns processos, no reforço da segurança, na gestão de erros e na acessibilidade. No geral, trata-se de uma aplicação equilibrada, que consegue conjugar funcionalidade e facilidade de uso, respondendo bem às necessidades dos seus utilizadores.

5 Manual de Utilização

A aplicação desenvolvida destina-se à gestão de turnos e horários de um curso, respondendo às necessidades de dois perfis distintos de utilizador: Diretor de Curso e Aluno. Este manual descreve os principais fluxos de utilização da aplicação para cada perfil, destacando as funcionalidades disponíveis e o modo de navegação, especificando como os mesmos satisfazem os cenários descritos.

5.1 Utilização pelo Diretor de Curso

5.1.1 Acesso

O diretor de curso deve autenticar-se na aplicação com as suas credenciais institucionais. Após o login, tem acesso a funcionalidades administrativas.

5.1.2 Perfil

Nesta página estão presentes os dados do diretor de curso, inclusive a sua informação pessoal e os seus objetivos e desafios.

5.1.3 Pedidos

Nesta página o diretor de curso pode visualizar o histórico de todos os pedidos efetuados por alunos ou professores. Cada pedido contém detalhes acerca do que se trata e permite ao diretor de curso aprovar ou rejeitar o mesmo. Além disso, permite ao utilizador aplicar filtros de pesquisa relativos a status (pendentes, aprovados ou rejeitados) e a tipo (troca de turno ou mudança de sala).

5.1.4 Alunos

Nesta página o diretor de curso consegue visualizar todos os estudantes inscritos no curso, bem como detalhes relativos ao seu estatuto, UC's inscritas e turnos. Quando determinado aluno efetua algum pedido de mudança de turno ou tem algum conflito de horários, esta página também fornece essa informação. Por último, esta página permite a publicação de horários, notificando depois todos os alunos do sucedido. Esta funcionalidade responde diretamente ao **cenário 1** do enunciado, permitindo ao diretor de curso resolver conflitos e reatribuir alunos.

5.1.5 Turnos

Esta página dá, ao diretor de curso, acesso a todos os turnos existentes para cada unidade curricular, bem como a informações sobre a ocupação de cada um. Fornece ainda a possibilidade de remover ou alocar alunos a um determinado turno, bem como a mudança da sua sala, mostrando que salas estão disponíveis para troca. No caso de ter sido efetuado um pedido de mudança de sala para um determinado turno, essa informação também é disponibilizada ao diretor de curso nesta secção. Esta página, tal como a referida anteriormente, permite também a publicação de horários. Esta funcionalidade responde diretamente aos **cenários 1 e 2** do enunciado, permitindo ao diretor de curso consultar os horários de cada UC, realocar alunos e efetuar mudanças de sala conforme solicitado por docentes.

5.2 Utilização pelo Aluno

5.2.1 Acesso

O aluno autentica-se com as suas credenciais. Após o login, tem acesso restrito às funcionalidades relevantes ao seu perfil.

5.2.2 Perfil

Nesta página estão presentes os dados do aluno, inclusive a sua informação pessoal e as disciplinas inscritas.

5.2.3 Horário

Nesta página o aluno pode consultar o seu horário atribuído, organizado por dias da semana e horas. Esta funcionalidade responde diretamente ao **cenário 4** do enunciado.

5.2.4 Meus Pedidos

Nesta página o aluno pode consultar o histórico de pedidos submetidos, verificando o estado atual de cada pedido (pendente, aceite ou rejeitado) e a data do mesmo. É fornecida ainda ao aluno a opção de cancelar um pedido que esteja ainda pendente.

5.2.5 Unidades Curriculares

Nesta página o aluno pode consultar as unidades curriculares em que está inscrito, com detalhe sobre os turnos existentes e o respetivo horário no qual eles decorrem. Esta página fornece a opção de pesquisa de disciplinas e filtros relativos ao semestre e ao ano.

5.2.6 Solicitação de Mudança de Turno

Esta funcionalidade está presente nas 3 últimas páginas referidas anteriormente. Em cada uma delas é possível ao aluno solicitar um novo pedido de mudança de turno, indicando o turno atual, o turno desejado e o motivo da mudança. Esta funcionalidade responde diretamente ao **cenário 4** do enunciado.

5.3 Funcionalidades comuns

- **Sistema de Alertas**: Modals e alertas informam o utilizador sobre o sucesso ou falha das ações (ex: submissão de pedidos, alterações guardadas, etc.).
- **Logout**: A qualquer momento, o utilizador pode terminar a sessão de forma segura através do botão visível na barra lateral.
- **Notificações**: O utilizador possui uma aba de notificações que fornece informações constantes acerca de ações que são efetuados no sistema. Por exemplo, um diretor de curso recebe notificações sempre que existe um pedido de mudança de turno novo e um aluno recebe notificações quando o diretor de curso aceita ou rejeita o seu pedido ou quando existe uma atualização no seu horário.

5.4 Mapa de Navegação

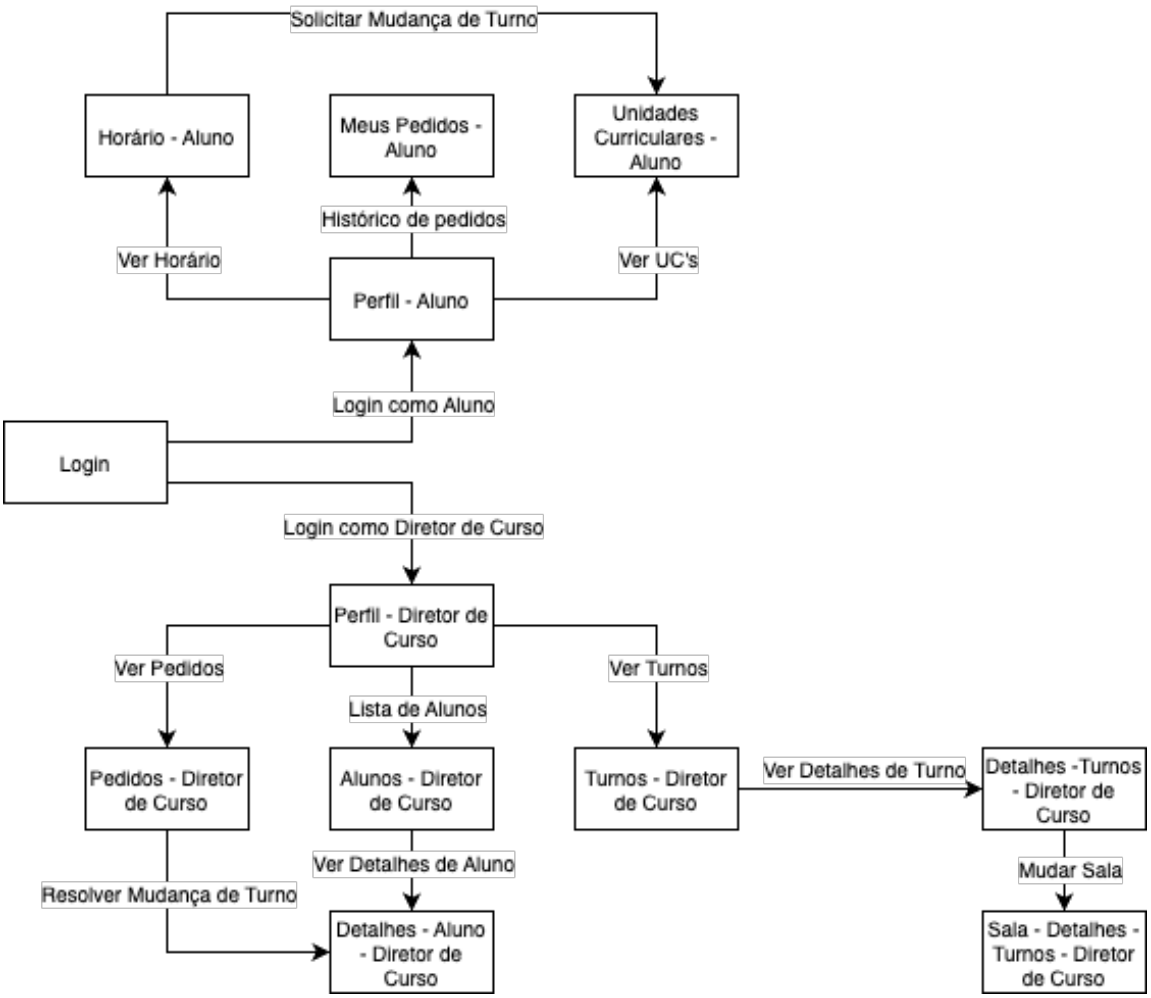


Figura 1: Mapa de Navegação – Fluxo de Utilização por Perfil

6 Conclusão

Ao longo do desenvolvimento deste projeto, enfrentámos vários desafios, mas também tivemos oportunidade de explorar e apreciar as potencialidades do **Vue.js**.

Principais Dificuldades Sentidas

Uma das maiores dificuldades prendeu-se com a **gestão do estado da aplicação**, especialmente quando diferentes componentes precisavam de aceder ou atualizar dados partilhados. A escolha entre manter o estado localmente ou recorrer a soluções globais (como contextos ou stores) obrigou-nos a repensar a arquitetura em alguns momentos.

Outro desafio significativo foi a **proteção e gestão de rotas** para diferentes perfis de utilizador. Garantir que cada utilizador só acedia às funcionalidades adequadas ao seu papel exigiu uma configuração cuidada do `vue-router` e a implementação de guardas de navegação robustos.

A **gestão de erros** também se revelou mais complexa do que inicialmente previsto. Foi necessário garantir que todos os fluxos de interação apresentavam feedback adequado ao utilizador, mesmo em situações de falha, o que implicou a criação de componentes de erro reutilizáveis e a integração de notificações em vários pontos da aplicação.

Por fim, a adaptação da interface a diferentes dispositivos e a preocupação com a **usabilidade** e **acessibilidade** obrigaram a múltiplos testes e ajustes.

O Que Mais Gostámos em Vue.js

Apesar dos desafios, o **Vue.js** revelou-se uma ferramenta extremamente versátil e agradável de utilizar. Destacamos especialmente:

- **Composition API**: A abordagem baseada em funções reativas (`ref`, `computed`, `watch`) permitiu-nos organizar o código de forma mais modular e reutilizável, facilitando a manutenção e a leitura.
- **Componentização**: A facilidade com que se criam e reutilizam componentes em Vue.js tornou o desenvolvimento mais ágil e permitiu-nos manter a interface consistente e escalável.
- **Documentação e Comunidade**: A documentação oficial é clara e completa, e a comunidade ativa facilitou a resolução de dúvidas e a procura de boas práticas.
- **Integração com Ferramentas Modernas**: A integração com bibliotecas como o `Axios` para chamadas HTTP e o `vue-router` para navegação foi simples e intuitiva.

Considerações Finais

Em resumo, este projeto permitiu-nos consolidar conhecimentos sobre desenvolvimento de aplicações web modernas e compreender melhor os desafios reais associados à construção de sistemas para diferentes perfis de utilizador. Apesar das dificuldades, a experiência foi bastante enriquecedora e reforçou a nossa confiança no ecossistema Vue.js para futuros projetos.