
UNIVERSIDADE D
COIMBRA

Carolina Afonso Leite Montezuma de Carvalho

# PULSAR

## Acessibilidade

**Dissertação no âmbito do Mestrado em Engenharia Informática, especialização em Engenharia de Software, orientada pelo Professor Doutor Raul Barbosa e Engenheiro Simão Castro, apresentada à Faculdade de Ciências e Tecnologia / Departamento de Engenharia Informática**

setembro de 2020

This page is intentionally left blank.

## Abstract

Nowadays, the World Wide Web is a fundamental part of most citizen's daily lives, being an asset most take for granted. However, although there is a notion that Web access is somewhat widespread, bad conducts regarding accessibility impose barriers to users who are visually impaired. This prevents these users from experiencing the web with the same degree of easiness. Common activities such as online commerce, leisure, communication or even education end up being complex, sometimes even impractical. Moreover, these constraints may compromise productivity on a professional level if the necessary web accessibility levels are not met. Considering the scope of the present scholar internship, this document gathers a study on Web accessibility practices for the blind, also exploring some significant obstacles establishing a valuable groundwork to reach its main goal: improving the accessibility in two modules of Critical Software's internal application PULSAR: Holidays/Absence Request and Effort Report. This application is used daily by the company's coworkers to perform various tasks such as requesting holidays, reporting expenses or reporting daily effort. Considering that Critical Software has visually impaired coworkers, it is our goal to upgrade their interactive experience with the specified PULSAR modules. In addition, we present all the practical process conducted, including the requirements definition, implementation and user testing. The gathered results suggest a better experience after addressing the reported accessibility issues.

## Keywords

Accessibility, Web, Assistive Technologies, Blind, Guidelines.

This page is intentionally left blank.

## Resumo

A *World Wide Web* é, nos dias de hoje, uma parte fundamental do quotidiano do cidadão comum, sendo praticamente um dado adquirido. No entanto, embora exista a perceção de que o acesso à *Web* se encontra banalizado, más condutas relativamente à acessibilidade impõem barreiras a utilizadores que padeçam de alguma deficiência visual. Isto impede que estes utilizadores usufruam da *web* com o mesmo grau de facilidade. Atividades comuns como comércio *on-line*, lazer, comunicação ou até mesmo educação acabam por ser complexas, muitas vezes impraticáveis. Para além disto, estes constrangimentos podem comprometer a produtividade a nível profissional se os níveis necessários de acessibilidade *web* não se verificarem. No âmbito do presente estágio curricular, este documento reúne um estudo sobre as práticas da acessibilidade *web* para invisuais, explorando ainda algumas barreiras significativas, estabelecendo bases para atender os objetivos do mesmo: melhorar a acessibilidade de dois módulos da aplicação *web* PULSAR da Critical Software: Pedido de Férias/Ausências e Reportar Esforço. Esta aplicação é usada diariamente pelos colaboradores da empresa e com diversos propósitos, como marcação de férias, registo de gastos ou registo de esforço diário. Considerando que a empresa em questão conta com colaboradores invisuais, é nosso objetivo melhorar a sua experiência interativa com a aplicação PULSAR nos módulos propostos. Adicionalmente, apresentamos todo o processo prático realizado, nomeadamente o levantamento de requisitos, implementação e testes de utilizador. Os resultados reunidos apontam para uma melhor experiência, tendo sido possível eliminar os problemas levantados relativamente às falhas de acessibilidade.



## Palavras-Chave

Acessibilidade, Web, Tecnologias de Assistência, Invisuais, Diretrizes.

This page is intentionally left blank.

# Agradecimentos

Ao longo deste ano, foram várias as pessoas que contribuiram não apenas para a realização do presente projeto como também para o meu enriquecimento a nível pessoal e profissional. Assim, agradeço a toda a equipa PULSAR por ter tornado a minha integração na equipa um processo fácil. Ainda, agradeço a disponibilidade dos colaboradores invisuais da empresa, cuja disponibilidade facilitou as implementações de acessibilidade.

Por fim, agradeço ao meu orientador da empresa, Engenheiro Simão Castro, todo o apoio e disponibilidade, e ainda ao meu orientador do Departamento de Engenharia Informática, Prof. Doutor Raul Barbosa.
Agradeço ainda à minha família e amigos, sempre presentes ao longo do meu percurso académico.

This page is intentionally left blank.

# Conteúdo

This page is intentionally left blank.

This page is intentionally left blank.

# Acronyms

**CSS** Cascading Style Sheets. 24, 26, 39, 44, 45, 49, 50, 66, 67

**HTML** HyperText Markup Language. xv, 12, 13, 17, 24, 26–28, 40, 42, 44, 45, 61, 89

**NVDA** Non Visual Desktop Access. 19

**W3C** World Wide Web Consortium. 17, 18, 22, 24

**WAI** Web Accessibility Initiative. 21–23

**WAI-ARIA** Accessible Rich Internet Applications. 18, 24–27, 31, 39, 42, 56, 89

**WCAG** Web Content Accessibility Guidelines. 3, 4, 17, 18, 20, 22, 23, 28, 31, 32, 76, 89

**web** World Wide Web. xv, 2–5, 8, 9, 16–28, 37, 44, 51, 76, 96, 97

This page is intentionally left blank.

# Lista de Figuras

This page is intentionally left blank.

# Lista de Tabelas

This page is intentionally left blank.

This page is intentionally left blank.

# Capítulo 1

# Introdução

Considerando que vivemos numa era digital, na qual simples tarefas quotidianas, como consultar a meteorologia, acompanhar as notícias ou fazer compras, podem ser realizadas *on-line* de forma rápida e eficaz, torna-se imperativo garantir a disponibilidade destes recursos para todos os utilizadores de igual forma.

Idealmente, a web seria um sistema de informação universalmente acessível a todos os utilizadores, incluindo aqueles que padecem de alguma deficiência, quer seja esta visual, auditiva, física, neurológica, cognitiva ou na fala [46]. No entanto, isto nem sempre é verdade. Os utilizadores que tirariam maior partido da interação com a web - dadas as dificuldades inerentes às suas deficiências e que, muitas vezes, os tornam pessoas pouco autónomas no dia-a-dia - são também as pessoas que se deparam com o seu acesso condicionado a certos conteúdos digitais [74]. O conceito de acessibilidade na web diz respeito a conteúdo e funcionalidades que podem ser acedidos por todas as pessoas de igual forma, independentemente do *software* e *hardware*, idioma, localização ou aptidão do utilizador em questão [69]. Se o conteúdo presente na web fosse acessível, iria permitir que, por exemplo, uma pessoa com mobilidade reduzida, possivelmente dependente de terceiros, realizasse compras sem necessitar de sair de casa. Na verdade, a falta de acessibilidade faz com que estes utilizadores se encontrem em desvantagem em comparação com os utilizadores comuns, sendo que esta ausência de cuidado está associada à falta de consideração das necessidades destes utilizadores, por parte dos programadores, durante o desenvolvimento do conteúdo web. Contudo, a acessibilidade não é apenas vantajosa para utilizadores com algum tipo de incapacidade permanente, mas também para pessoas que utilizem, por exemplo, dispositivos com ecrã pequeno (e.g, telemóveis, *smart watches*), idosos que perderam algumas capacidades devido à idade, incapacidades temporárias (e.g., um braço partido), limitações inerentes a uma determinada restrição (e.g., um ambiente no qual não pode ouvir áudio), fraca ligação à Internet ou largura de banda cara e limitada [46].

Hoje em dia, existe uma forte tendência para criar *websites* baseados em componentes visuais [69] - permitindo tornar a interação com os mesmos mais simples e apelativa. No entanto, isto torna-se um fator de exclusão para pessoas com deficiências visuais, caso o *website* não se encontre acessível a este tipo de utilizadores [69] .

Segundo a Organização Mundial de Saúde, existem cerca de 2.2 mil milhões de pessoas com uma incapacidade visual ou cegueira [53]. Uma percentagem deste valor necessita do auxílio de tecnologias de assistência como leitores de ecrã, *browsers* de voz ou linhas de *braille* para conseguirem aceder ao conteúdo digital. Porém, para que estas tecnologias sejam eficazes, e consigam transmitir de forma efetiva a informação, as páginas têm que ser corretamente programadas seguindo o documento composto por diretrizes de acessibilidade

Web Content Accessibility Guidelines (WCAG) - apresentado na secção 3.4 -, tendo ainda em consideração as dificuldades destes utilizadores de forma a atender às necessidades dos mesmos durante o desenvolvimento do conteúdo.

A forte presença do mundo digital no dia-a-dia do cidadão comum promove a noção de uma web para todos. Porém, tal não se verifica, dado que um elevado número de utilizadores é afetado pela ausência ou baixos níveis de acessibilidade. No âmbito do estágio curricular, pretende-se o melhoramento de módulos referentes à aplicação web PULSAR da Critical Software, elevando os níveis de acessibilidade dos mesmos por forma a melhorar a experiência dos profissionais invisuais da empresa.

O presente documento descreve o trabalho realizado durante este primeiro semestre, assim como o plano de trabalho futuro e definição de objetivos. Adicionalmente, este documento reúne a pesquisa elaborada a partir de trabalhos relacionados, pretendendo assim agregar informação sobre a importância da acessibilidade no melhoramento de *websites* como ponto de partida para o trabalho a desenvolver.

## 1.1 Contextualização

O presente documento surge no âmbito da realização do estágio curricular na empresa Critical Software. O trabalho desenvolvido ao longo deste ano letivo teve como objetivo global melhorar a acessibilidade na aplicação Web da empresa: PULSAR - apresentada mais à frente neste documento (secção 2.1).

Como referido anteriormente, a acessibilidade cobre diversos tipos de deficiências. O trabalho desenvolvido ao longo do estágio focou-se na área de acessibilidade para pessoas invisuais, tendo por princípio melhorar a interação dos colaboradores da empresa que padecem desta deficiência com a aplicação. Desta forma, toda a investigação e conjunto de práticas realizados ao longo deste ano letivo contemplou apenas pessoas totalmente invisuais, excluindo indivíduos parcialmente invisuais ou com qualquer outro tipo de deficiência visual (e.g., miopia, daltonismo).

No contexto desta tese, os *stakeholders* são a própria empresa (Critical Software), a equipa de desenvolvimento do PULSAR, os colaboradores invisuais da Critical Software e o responsável pela orientação do estágio. Enquanto parte interessada, a Critical Software pretendia, com o presente projeto, obter uma plataforma mais acessível, nomeadamente perante os seus colaboradores invisuais. Relativamente às suas responsabilidades e papéis neste projeto, a equipa de desenvolvimento irá contribuir para a validação da acessibilidade de cada módulo. Os colaboradores invisuais da Critical Software são, por sua vez, os utilizadores finais, sendo que o presente projeto de estágio foi desenvolvido com o intuito de melhorar a acessibilidade da aplicação PULSAR tendo em vista as suas limitações. Estes contribuíram para o desenvolvimento do mesmo através da partilha das suas dificuldades e necessidades, tornando claros os pontos a abordar e a melhorar. Desta forma, estes colaboradores foram um ponto-chave para a avaliação dos módulos especificados. Por fim, o responsável pela orientação do estágio - destacado pela empresa - foi uma entidade fulcral no auxílio do desenvolvimento e supervisionamento do trabalho desenvolvido.

O presente projeto, sendo um estágio curricular, previu dezasseis horas de trabalho semanal durante o primeiro semestre, sendo que no segundo semestre foram estipuladas quarenta horas semanais.

## 1.2 Objetivos

O presente estágio curricular teve como objetivo melhorar a acessibilidade da aplicação web da Critical Software - PULSAR. Uma vez que não seria possível trabalhar a acessibilidade em toda aplicação durante o período disponível, foram especificados e priorizados módulos tendo em conta as necessidades dos utilizadores invisuais da empresa e a relevância desses mesmos módulos no contexto da aplicação.

Para tal, um colaborador invisual da empresa identificou quatro módulos do PULSAR nos quais sentia maiores dificuldades, tendo especificado, também, os problemas que enfrentava durante a interação com os mesmos.

Os módulos foram priorizados da seguinte forma:

- Objetivos Principais:
    1. Pedido de Férias/Ausências
    2. Reportar Esforço
- Objetivos Secundários:
    1. Mapa de Despesas
    2. Pedido de Viagem

Numa fase inicial, tivemos em consideração o desenvolvimento de acessibilidade para os módulos que considerámos serem os objetivos principais: Pedido de Férias/Ausências e Reportar Esforço. Estes dois módulos foram destacados como principais atendendo ao facto de serem módulos consultados com muita frequência por todos os colaboradores da empresa.

A implementação de acessibilidade nos seguintes módulos - Mapa de Despesas e Pedido de Viagem - encontrava-se dependente do sucesso da implementação de acessibilidade nos outros dois módulos. Isto é, apenas abordaríamos os objetivos secundários a partir do momento em que os dois primeiros módulos fossem implementados e devidamente testados.

A implementação de acessibilidade teve em conta, principalmente, as dificuldades reportadas pelo colaborador invisual e, ainda, as diretrizes do documento WCAG 2.1 - apresentado na secção 3.4 - consideradas adequadas ao contexto dos módulos e ao âmbito do projeto.

Durante a fase de implementação, os colaboradores invisuais da Critical Software estiveram sempre envolvidos no processo, permitindo testar atempadamente e receber *feedback* dos mesmos sobre as alterações realizadas aos módulos.

Para além da validação destes objetivos estarem dependentes da análise dos utilizadores finais, será tido como fator de sucesso a implementação dos casos de uso destacados com prioridade Alta. Os casos de uso definidos para o presente projeto são apresentados na secção 5.1.

As funcionalidades inerentes a estes quatro módulos são detalhadas na secção 2.3 deste documento.

## 1.3 Estrutura do Documento

O presente documento encontra-se divido da seguinte forma:

- **Enquadramento**, onde apresentamos uma breve descrição do sistema PULSAR, entrando em mais detalhe nos módulos especificados para implementação de acessibilidade. As tecnologias utilizadas no âmbito do presente projeto de estágio são também apresentadas neste capítulo, assim como a metodologia de desenvolvimento usada pela equipa.
- **Trabalho Relacionado**, onde são apresentados estudos relacionados com a acessibilidade na web, incluindo as tecnologias de assistência utilizadas por pessoas invisuais, uma compilação de barreiras de acessibildade associadas a pessoas invisuais e métodos de avaliação de acessibilidade na web. Em adição, apresentamos um conjunto de diretrizes e especificações técnicas que visam combater com estas barreiras.
- **Trabalho Preliminar**, onde apresentamos o trabalho realizado ao longo do primeiro semestre.
- **Especificação de Requisitos**, onde apresentamos os requisitos (funcionais e não-funcionais) definidos para o presente projeto de estágio.
- **Implementação**, onde descrevemos todo o processo de implementação de acessibilidade nos módulos propostos como objetivos principais, acompanhado pelas dificuldades e *feedback* fornecidos pelos colaboradores invisuais. Complementarmente, toda a descrição é acompanahda - e suportada - pelas noções de boas práticas da acessibilidade estudadas no primeiro semestre.
- **Teste de Software**, onde apresentamos os diferentes estágios de testes conduzidos ao longo do processo de implementação, apresentando por fim os Testes de Usabilidade, que nos permitiram avaliar o impacto das melhorias de acessibilidade conduzidas.
- **Discussão**, onde analisamos e discutimos de forma detalhada os resultados obtidos, tal como o seu impacto no contexto do presente projeto e expetativas futuras. Adicionalmente, fazemos ainda um balanço do trabalho conduzido.
- **Plano de Trabalho**, onde apresentamos as tarefas conduzidas no primeiro semestre, assim como o plano de trabalho elaborado para o segundo semestre e respetivo desempenho.
- **Conclusão**, onde refletimos através de uma visão geral sobre todo trabalho realizado neste estágio, concluindo com algumas notas finais.

This page is intentionally left blank.

This page is intentionally left blank.

# Capítulo 2

# Enquadramento

O presente capítulo descreve a aplicação web PULSAR da Critical Software e os módulos especificados para implementação de acessibilidade. Considerámos pertinente a inclusão de uma secção que descrevesse de forma breve esta aplicação, com o intuito de enquadrar o leitor relativamente à função da mesma. Complementarmente, este capítulo apresenta ainda uma descrição de cada módulo sobre os quais a acessibilidade foi implementada, através de uma visão geral das suas funcionalidades. Por fim, são apresentadas as tecnologias que serão usadas para o desenvolvimento do presente projeto.

## 2.1 PULSAR

O PULSAR é uma aplicação World Wide Web (web) da Critical Software utilizada por todos os seus colaboradores e que visa à gestão de operações da empresa. Operações como gestão da informação dos colaboradores (e.g., contactos, férias, informação contratual) ou gestão de projetos a decorrer na empresa (e.g., custos associados, recursos) podem ser realizadas através do uso deste sistema interno da empresa. Cada conjunto de operações está dividida em módulos, sendo que a aplicação é composta por 20 módulos. Toda a informação presente na aplicação está disponível em inglês ou português.

Tendo em consideração as suas funcionalidades, o PULSAR é utilizado diariamente pelos colaboradores da Critical Software. Assim, sendo uma aplicação de consulta frequente, uma boa acessibilidade da mesma assume-se como fundamental para a interação dos colaboradores invisuais. Como apontado no capítulo anterior (secção 1.2), para o presente projeto foram destacados quatro módulos nos quais se verificam mais problemas. Na secção seguinte iremos apresentar em mais detalhe os módulos especificados para a implementação de acessibilidade.

A Figura 2.1 ilustra a arquitetura da aplicação PULSAR, do ponto de vista da web. A aplicação, desenvolvida em Angular, interage com a camada de serviços através de uma API REST, fazendo uso dos protocolos HTTPS e WSS (WebSocket Secure), sendo que a arquitetura REST permite a transferência de informação no formato JSON. Adicionalmente, a aplicação tira partido dos serviços fornecidos pelo Keycloak, como fim de autenticação. Embora o projeto aqui apresentado tenha sido desenvolvido no *front-end* da aplicação (i.e., unicamente no módulo web apresentado), optámos por incluir uma ilustração da arquitetura como descrição complementar.

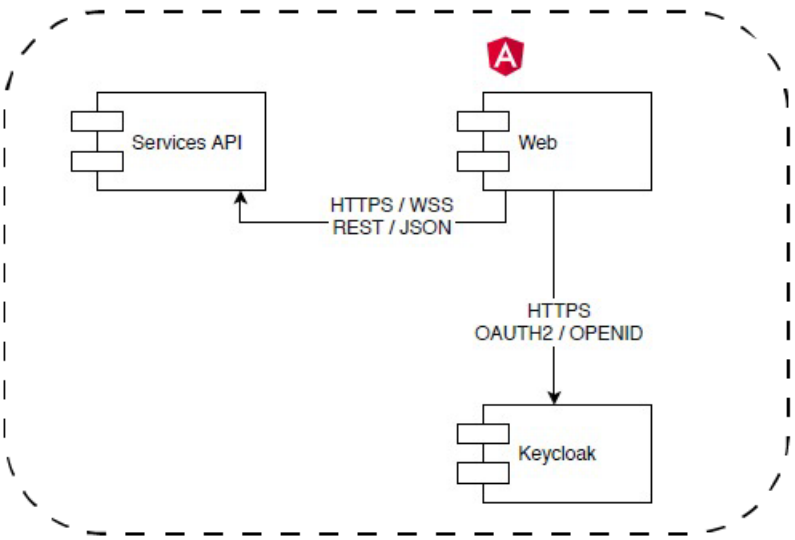


Figura 2.1: Arquitetura PULSAR (vista web)

## 2.2 Metodologia do Projeto

O projeto PULSAR segue uma metodologia de desenvolvimento *agile* - *Scrum*. A metodologia (ou *framework*) Scrum permite a utilização de vários processos e técnicas durante o desenvovimento de um produto, invés de fazer uso de um único processo (ou técnica) [70].

No presente estágio, o processo de desenvolvimento era composto por *sprints* com a duração de duas semanas. No último dia de cada *sprint* tinha lugar uma reunião (*Sprint Planning*), na qual eram definidas as tarefas pertencentes ao *Product Backlog* (lista de funcionalidades a serem implementadas no projeto), que deveriam ser adicionadas ao *sprint* seguinte (adicionando-as ao *Sprint Backlog* - tarefas definidas para o *sprint*). Antes de cada *sprint* terminar, era realizado um *Refinement*, que consistia numa reunião entre a equipa de desenvolvimento e o *Product Owner*. Aqui, eram apresentadas tarefas do *Product Backlog* que necessitavam de ser ajustadas, clarificadas ou estimadas relativamente ao esforço associado às mesmas.

Durante o decorrer do *sprint*, eram realizadas reuniões diárias com o objetivo de ir acompanhando o processo das tarefas realizadas na equipa. Os membros da equipa partilhavam o estado das tarefas que estavam a realizar no momento, assim como eventuais problemas que estivessem a impedir a conclusão das mesmas.

Por fim, no último dia de cada *sprint*, era realizada uma demonstração na qual cada membro da equipa apresentava as tarefas realizadas durante a *sprint* que já se encontravam em produção.
Adicionalmente, no fim de cada demonstração, tinha ainda lugar uma retrospetiva da *sprint* passada com todos os membros da equipa, na qual cada membro refletia sobre aspetos positivos e negativos relativos à *sprint* passada. Isto permitia uma discussão sobre o que deveria ser suprimido ou continuado a seguir, assim como um debate sobre melhorias futuras sobre o desempenho da equipa.

## 2.3 Módulos Especificados

### 2.3.1 Pedido de Férias/Ausências

A Figura 2.2 apresenta o módulo responsável pela marcação/gestão de férias e ausências dos colaboradores da empresa, estando disponível para os respetivos gestores de equipa diretamente no sistema. O módulo é composto por um calendário com os meses de janeiro do ano atual a março do ano seguinte. A informação relativa as férias e ausências como dias marcados, novos dias pedidos, dias marcados com pedido de cancelamento é apresentada no calendário com recurso a codificação de cores.

O significado de cada uma das cores utilizada para a criação deste mapeamento é apresentado numa legenda juntamente com a totalidade de dias para cada uma das situações. Adicionalmente, o módulo apresenta ainda uma secção onde podem ser anexados ficheiros, como atestados médicos ou outro tipo de justificação para a justificação de ausência.

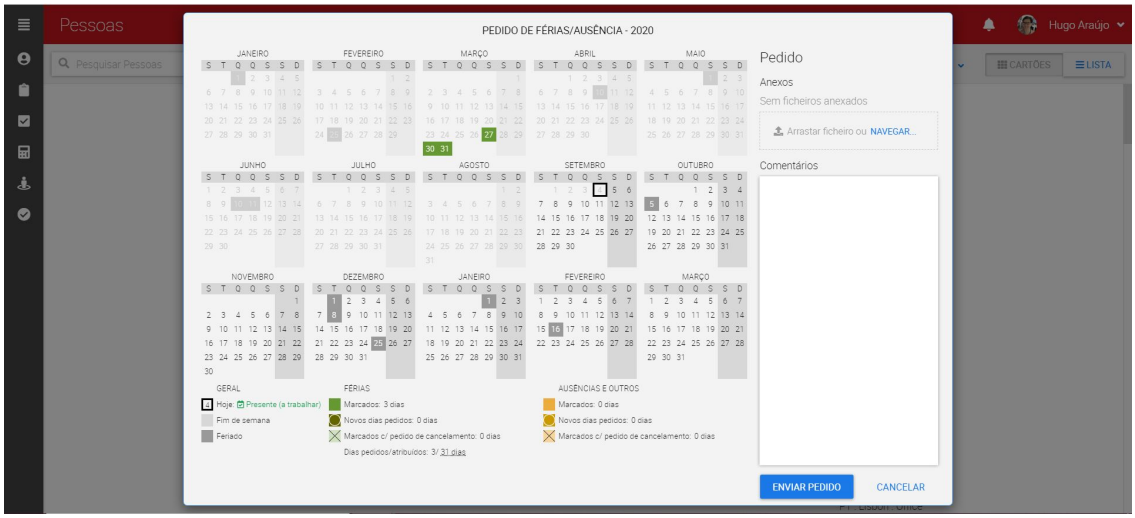


Figura 2.2: Módulo Pedido de Férias/Ausências

### 2.3.2 Reportar Esforço

A Figura 2.3 apresenta o módulo que permite aos utilizadores reportar o seu esforço diário em projetos e atividades, permitindo que os gestores de equipa validem o esforço das mesmas. Para tal, o utilizador deve reportar em cada dia o tempo (em horas e minutos) despendido nos mesmos.

De forma semelhante ao módulo Pedido de Férias/Ausências apresentado anteriormente, existe uma codificação de informação com cores (e.g., férias, ausências e esforço). Todos os dias com esforço reportado serão apresentados no calendário utilizando a cor roxa e, caso tenham sido reportadas mais ou menos horas do que aquelas que eram expectáveis, o dia passa a ser representado com uma variação de tons de roxo (mais escuro e mais claro, respetivamente). Desta forma, os utilizadores conseguem facilmente perceber quais os dias que podem ter sido mal reportados. No entanto, como acontece no módulo anterior, as pessoas invisuais não conseguem ter esta percepção à partida quando abrem o calendário, ao contrário dos restantes utilizadores visuais. Para conseguirem saber quais são os dias com esforço reportado, ou ainda, por exemplo, aqueles que indicam esforço excedido, os utilizadores invisuais terão que percorrer todos os dias do mês utilizando o teclado para

verificar quais os dias que já contêm a informação. O mesmo se aplica à informação relativa às férias ou ausências marcadas.

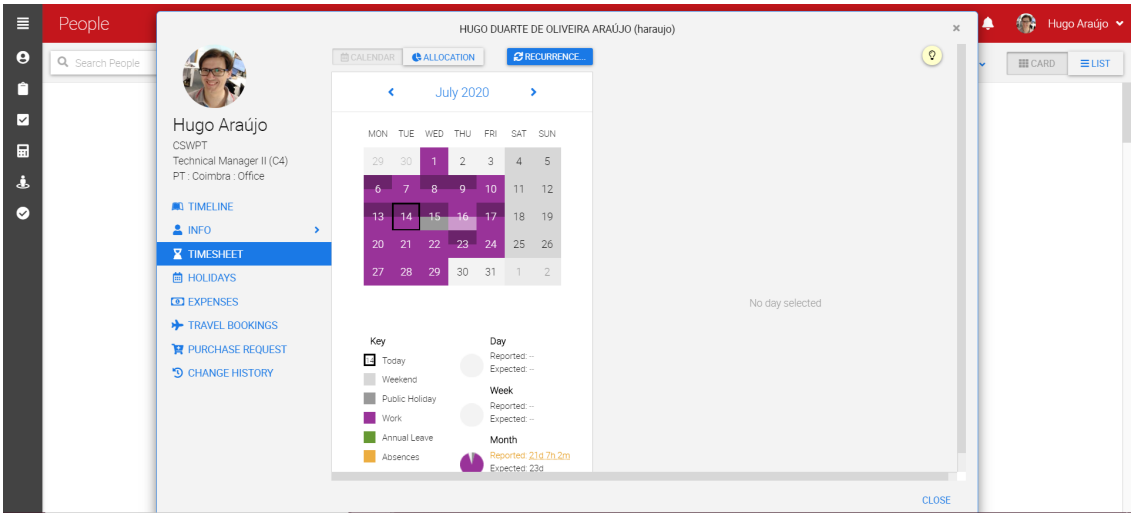


Figura 2.3: Módulo Reportar Esforço

### 2.3.3 Mapa de Despesas

A Figura 2.4 apresenta o módulo responsável pelos registos de despesas realizados no âmbito de uma atividade empresarial, permitindo posterior reembolso ao colaborador. Para tal, os colaboradores têm de preencher os campos necessários para o registo de despesa. São estes: nome do mapa de despesas, tarefa associada à despesa e tipo de despesa - preenchendo os campos correspondentes a cada tipo e anexando o comprovativo de despesa caso seja aplicável.

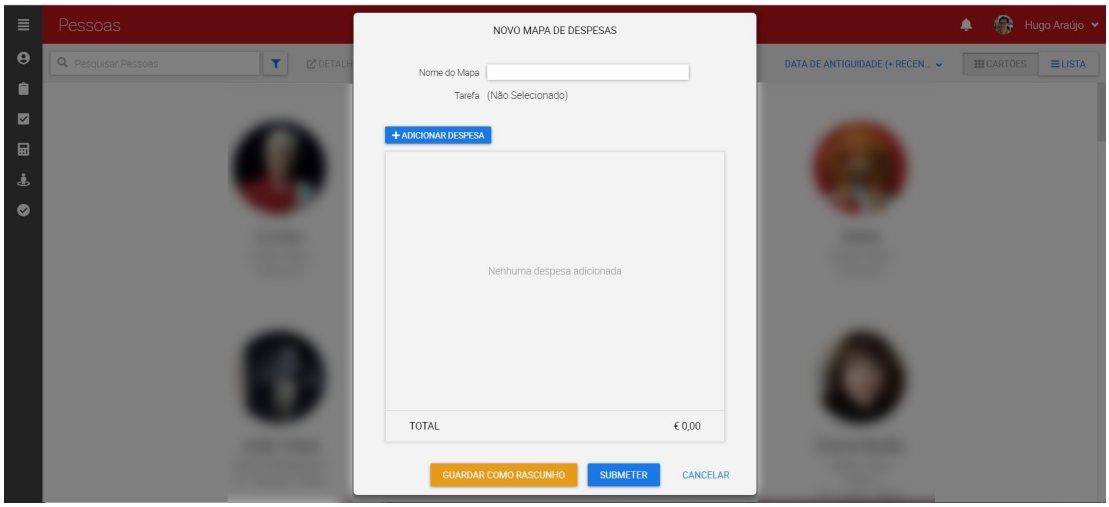


Figura 2.4: Módulo Mapa de Despeses

### 2.3.4 Pedido de Viagem

A Figura 2.5 apresenta o módulo responsável pelo pedido de uma viagem realizada no âmbito de projetos ou atividades da empresa. Para a realização do pedido, os utilizadores deverão preencher campos referentes ao nome da viagem, a tarefa associada à viagem,

nomes dos participantes (podendo ainda mencionar a existência de participantes externos), o país de origem bem como a cidade, o país de destino, o tipo de viagem (ida e volta ou só ida), a data de partida e a data de regresso - se for o caso, existe a possibilidade de informar que existe flexibilidade relativamente às horas. Como informação adicional, o utilizador pode ainda selecionar alguns requisitos (Alojamento, Aluguer de viatura, Declaração de seguro de viagem, Estacionamento no aeroporto, passagem aérea, reserva de lugar, transfer e visto) ou deixar uma nota escrita.

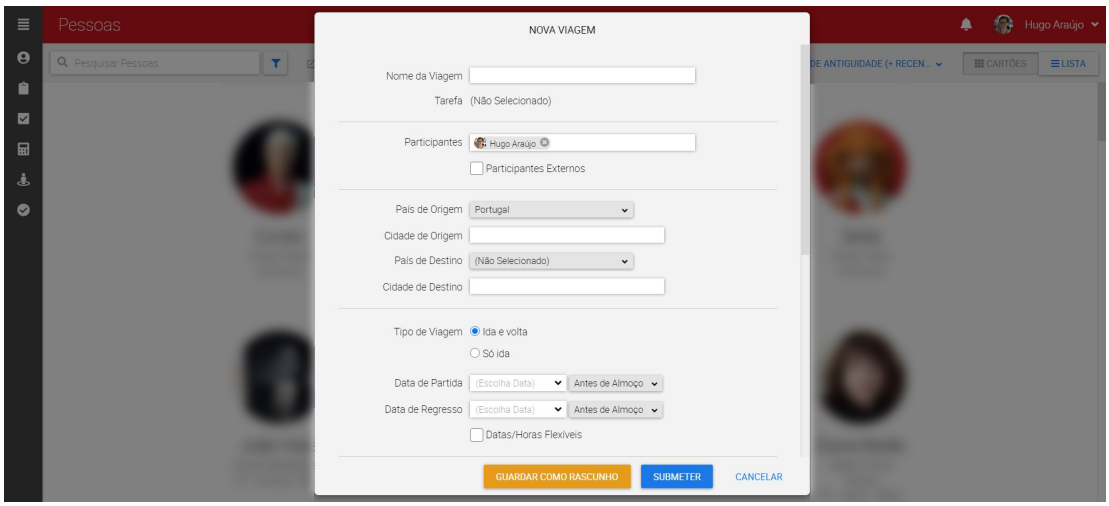


Figura 2.5: Módulo Pedido de Viagem

## 2.4 Tecnologias Utilizadas

Nesta secção listamos as tecnologias que foram utilizadas ao longo do próximo semestre. Uma vez que estas são as tecnologias usadas pela equipa do PULSAR, e considerando ainda que a implementação da acessibilidade passa pela análise e reformulação de código existente, foi necessário que tais se mantivessem. São estas:

- **HTML:** Linguagem *markup* para desenvolvimento de páginas web [17].
- **Typescript:** Linguagem de programação *open-source* criada a partir de Javascript. A linguagem Typescript estende a linguagem Javascript através da adição de tipos Typescript [28].
- **Sass:** Extensão da linguagem CSS que permite a utilização de funcionalidades que não existem no css como, por exemplo a definição de variáveis ou a definição de *mixins* (grupos de declarações css que se pretendem usar mais do que uma vez) [11].
- **Angular**: *Framework* que permite desenvolver aplicações de página única (*single-page applications*), usando HyperText Markup Language (HTML) e Typescript. A arquitetura de uma aplicação desenvolvida a partir de Angular tem por base determinados conceitos fundamentais. No contexto da presente tese, será relevante descrever sucintamente Componentes e Serviços Angular. Cada componente define uma classe que contém informação sobre a aplicação, sendo associado a um template HTML, que tratará de exibir essa mesma informação. Complementarmente, cada componente usa serviços Angular. Os serviços fornecidos por esta *framework* podem ser partilhados por diferentes componentes, sendo injetados nestes últimos como dependências. Fundamentalmente, um único serviço é uma categoria vasta,

podendo abranger qualquer valor, função, ou característica que a aplicação necessite. Esta segmentação de funcionalidades - nomeadamente a separação entre o template HTML ( *views*, como nomeado na documentação) e módulos (que contém componentes) - destingue esta *framework*, sendo que estas relações pretendem tornar o código modular, reutilizável e eficiente [2].

- **Angular CLI**: Interface da linha de comandos que permite inicializar, desenvolver e realizar a manutenção de aplicações em Angular [8].

- **Yarn**: *Package Manager* que armazena em *cache* todas as *packages* para que não seja necessário fazer o *download* novamente [12].

- **Bootstrap**: *Toolkit* de desenvolvimento de aplicações web utilizada juntamente com a linguagem SASS [4].

- **Tslint**: Ferramenta que permite fazer a verificação de código Typescript, mais concretamente erros de funcionalidade, manutenção e de legibilidade [27].

- **SonarLint:** Extensão do IDE que permite a deteção de erros durante a escrita do código (antes ser efetuado *commit*) como, por exemplo, *bugs* e vulnerabilidades a nível de segurança. Complementarmente, o SonarLint fornece ainda sugestões para resolver os problemas detetados [24].

- **Jira**: *Software* utilizado de forma a auxiliar equipas no planeamento de tarefas (i.e., *issues*) a realizar ao longo de uma *sprint*, por forma a acompanharem o desenvolvimento das mesmas. O termo *issue* utilizado no contexto do Jira diz respeito a blocos de trabalho que têm de ser implementados (e.g., *user stories* ou resolução de *bugs*). Complementarmente, as tarefas podem ainda ser priorizadas tendo em consideração a sua relevância. O Jira pode apresentar diferentes *workflows* que consistem numa lista de passos (representados em diferentes colunas no Jira) pelo qual os *issues* têm que passar desde a fase inicial até a sua conclusão. No PULSAR, o *workflow* adotada seguia os seguindes passos: TO DO - IN PROGRESS - INTEGRATION - DONE. Em primeira instância, uma tarefa entrava na coluna TO DO após ser atribuída a um membro da equipa. Posteriormente, a tarefa avançava para a coluna IN PROGRESS assim que o processo de implementação da tarefa fosse iniciado, permanecendo neste passo até à conclusão da mesma. Quando a implementação da tarefa era dada como concluída, a mesma tinha que ser revista por outros membros da equipa (criando um Pull Request, detalhado na subsecção 7.1 do Capítulo Testes de Software). Por fim, após a tarefa ser revista e aprovada, é realizado o *merge* e passa para a coluna DONE [20].

- **Bitbucket:** Serviço de *hosting* de projetos e ferramenta de colaboração. Para além de permitir controlo de versões, promove a eficiência através de ligação de *branches*, *commits* e *pull requests* de software Jira [6].

- **Microsoft Teams:** Aplicação utilizada para comunicação entre equipa.

- **Jenkins:** Servidor de automação *open-source* utilizado para automatizar variadas tarefas relacionadas com testes, *building*, *delivering* ou *deploying* de software [19].

- **Confluence:** Ferramenta de colaboração que visa auxiliar a colaboração de equipas, assim como a partilha eficiente de conhecimentos. Sendo um editor colaborativo, oferece diversas funcionalidades como projetar planos, requisitos de produtos e notas de reuniões [10].

Este conjunto de tecnologias permitiu a implementação de acessibilidade nos referidos módulos da aplicação web PULSAR.

This page is intentionally left blank.

# Capítulo 3

# Trabalho Relacionado

O presente capítulo resulta da análise de diversos estudos, artigos e documentos relativos à acessibilidade na web e às principais barreiras encontradas por utilizadores invisuais. Esta pesquisa permitiu reunir informação que fornece uma visão geral sobre a acessibilidade na web, observando que é um fator ainda bastante descurado aquando da implementação de um *website*. Adicionalmente, são exploradas as principais tecnologias de assistência, assim como diretrizes existentes que visam promover uma boa acessibilidade e, ainda, alguns métodos utilizados para a avaliação de acessibilidade nas páginas web.

Considerando o âmbito do presente projeto, todo este estudo é maioritariamente centrado a utilizadores invisuais.

## 3.1 Acessibilidade na Web

A web é considerada um dos principais meios de comunicação remota [55]. Para além disso, é ainda uma valiosa fonte de informação e de serviços para os seus utilizadores [55] [69], permitindo que estes possam realizar diversas ações diariamente como consulta de notícias, plataformas de comércio ou, ainda, ou atividades sociais [74]. Citando Tim Berners-Lee, *"O poder da web reside na sua universalidade. O acesso a todas as pessoas, independentemente da sua incapacidade, é um aspeto essencial."* [46]. No entanto, embora a web tenha sido concebida com o objetivo de ser universalmente acessível [72] e de a acessibilidade ser um fator legalmente exigido em muitos países [72] [73] [59], esta não se encontra acessível a todas as pessoas de igual forma [74].

Idealmente, a web permitiria quebrar barreiras de interação e comunicação que os utilizadores com deficiências (e.g., invisuais, atendendo ao âmbito do presente estágio) enfrentam no mundo físico [46], permitindo assim que estes utilizadores realizassem atividades que até então não conseguiam realizar autonomamente [66]. No entanto, são também os utilizadores invisuais que, geralmente, encontram mais barreiras durante a navegação, acabando por ser excluídos desta fonte de informação [65]. Um estudo recente realizado pela *Nucleus Research* [77] focado em pessoas invisuais ou com baixa visão - maioritariamente residentes na América do Norte - revelou que mais de 70% dos *websites* analisados são inacessíveis para este tipo de utilizadores. Como reportado em [56], as dificuldades de interação encontradas pelos utilizadores podem estar associadas à falta de acessibilidade e usabilidade nas tecnologias web. Durante o desenvolvimento de um *website*, a acessibilidade e usabilidade devem ser consideradas em conjunto uma vez que a acessibilidade por si só não é suficiente, sendo importante garantir que os utilizadores invisuais conseguem realizar as suas tarefas

de forma eficaz e satisfatória [66].

Em [59] os autores definem acessibilidade na web como a capacidade de compreender, navegar e interagir com a informação apresentada aos utilizadores, independentemente de estes serem portadores de uma deficiência ou não. Já a usabilidade, por sua vez, é definida através do grau de conformidade entre a interface e o conhecimento dos utilizadores para realizar uma determinada tarefa num *website* [59]. Como referido em [66], para melhorar a usabilidade na web é necessário ter em conta as necessidades específicas dos utilizadores, uma vez que a conformidade com as normas de acessibilidade por si não garante que os utilizadores invisuais consigam alcançar os seus objetivos através de uma quantidade de esforço e tempo razoáveis [66].

Além de a falta de acessibilidade e usabilidade dos *websites* ser indesejada para todos os utilizadores, no caso dos utilizadores invisuais surgem problemas adicionais quando estes tentam realizar tarefas *on-line* [56] auxiliados por tecnologias de assistência, que consistem em *software* e *hardware* que permitem facilitar a interação destas pessoas com a web [50].

A acessibilidade está dependente do funcionamento conjunto de vários componentes de desenvolvimento web e interação. Estes componentes incluem: "*user agents*" (e.g., *web browsers*), tecnologias de assistência (e.g., leitores de ecrã), conhecimento e experiência dos utilizadores, programadores, *authoring tools* e ferramentas de avaliação [51]. Adicionalmente, o conteúdo de uma página web também é uma parte importante no bom funcionamento da acessibilidade, sendo que requerem iguais cuidados de implementação, como veremos mais à frente (3.6). Nesse sentido, é também relevante perceber o impacto do comportamento do foco na web, particularmente no campo que estudamos.
Quando falamos de foco, referimo-nos ao elemento na página (um campo de *input*, *checkbox*, botão ou *link*, por exemplo) que no momento recebe *input* do teclado [62]. Para utilizadores que usem predominantemente o teclado como meio de navegação pela página, o foco assume um papel importante, já que é através deste que percecionam o alcance dos elementos no ecrã [62].
É importante notar que nem todos os elementos recebem foco, como parágrafos ou *divs*, já que geralmente é desnecessário focar elementos com os quais o utilizador não consegue interagir [62]. Porém, um elemento que - através do teclado - não recebe foco por definição, pode ser programado para receber foco [63]. Pelo contrário, um elemento que recebe pelo teclado (tecla Tab) pode ser programado para não receber foco [63]. Tal é possível através do atributo HTML `tabindex`. Adicionalmente, através deste atributo, podemos modificar a ordem pela qual os elementos recebem foco aquando da navegação com a tecla Tab [63]. Este atributo é definido seguido de um número que define o seu comportamento. Isto é , o valor -1 define que o elemento não recebe foco através do teclado (podendo receber foco de forma programática), sendo que qualquer valor inteiro maior que 1 define a sua prioridade de forma ascendente. Por sua vez, quando o atributo `tabindex` é igual a 0, o elemento poderá receber foco de forma programatica ou, ainda, através do uso do teclado (pela ordem que é apresentado no DOM)[63]. Ainda, sempre que se pretende controlar ou alterar a ordem pela qual os elementos são focados, utilizar os valores inteiros positivos no atributo `tabindex` não é uma boa prática, sendo sempre preferível alterar a ordem dos elementos diretamente no ficheiro HTML [26].
Considerando que utilizadores com determinado tipo de deficiências - nomeadamente utilizadores invisuais - têm na navegação por teclado um apoio importante à sua experiência, o foco é um factor crítico [62]

O World Wide Web Consortium (W3C) desenvolveu padrões de acessibilidade na web para os diferentes componentes. São estes: *Authoring Tool Accessibility Guidelines* (ATAG) - documento que lida com as "*authoring tools*"; *Web Content Accessibility Guidelines* Web

Content Accessibility Guidelines (WCAG) - lida com o conteúdo da web e é usado não só pelos programadores como pelas "*authoring tools*"e de avaliação; e, por fim, *User Agent Accessibility Guidelines* (UAAG) - lida com *web browsers* e alguns aspectos das tecnologias de assistência [51]. Na secção 3.4 apresentamos em mais detalhe o documento (WCAG), uma vez que servirá como um dos alicerces para o presente projeto de estágio.

O W3C definiu ainda especificações técnicas que lidam diretamente com acessibilidade a Accessible Rich Internet Applications (WAI-ARIA) - apresentada na secção 3.5. Estas especificações visam suprimir as diversas barreiras encontradas por utilizadores invisuais [51].

Em [73], os autores apresentam a definição de barreira - no que a acessibilidade diz respeito - como sendo uma condição que dificulta a realização de tarefas num *website* por parte dos utilizadores de tecnologias de assistência. Consequentemente, isto pode resultar na necessidade de ajuda de terceiros [69]. No entanto, a falta de cuidado ao seguir as diretrizes de acessibilidade durante o desenvolvimento de um página web pode dificultar este processo [55]. Todas estas questões relacionadas com problemas comuns com os quais os utilizadores invisuais se deparam são apontadas com mais detalhe na secção 3.3. Os invisuais são um grupo significativo de utilizadores que interagem com a web de forma muita distinta comparativamente aos utilizadores visuais [56]. Assim, boas práticas de acessibilidade servem de base a *software*, ferramentas e dispositivos que visam melhorar esta experiência.

Em [64], os autores apresentam uma extensão para *web browsers* que permite melhorar a acessibilidade das páginas. Esta extensão permite enriquecer *links*, imagens e a navegação. Muitas vezes, os *links* encontram-se invisíveis para os leitores de ecrã uma vez que não têm um texto associado que seja acessível a estas ferramentas. Desta forma, a extensão em causa adiciona automaticamente o atributo `aria-labelledby` (secção 3.5.1) às *tags* `<a>` encontradas na página, permitindo assim que os utilizadores invisuais tenham mais informação sobre as mesmas. Para além disso, esta extensão percorre todas as imagens encontradas numa dada página que não tenham um texto descritivo, associando texto automaticamente. Esta extensão adiciona ainda um *link* apenas visível para os leitores de ecrã que permite ao utilizador saltar para o conteúdo principal sem ter que ouvir novamente o menu de navegação. Com estas técnicas implementadas, os autores apresentam resultados revelando que estas permitiram eliminar 50% dos problemas de acessibilidade apontados pelos utilizadores nos *websites* que foram avaliados.

Estes resultados sugerem que a melhoria da acessibilidade geral de uma página web enriquece a experiência interativa com a mesma. Sabendo que as tecnologias de assistência são um meio extremamente relevante para esta interação - considerando utilizadores invisuais -, será pertinente estudar de forma mais aprofundada as suas características. A seguinte secção trata disso mesmo.

## 3.2 Tecnologias de Assistência

As tecnologias de assistência utilizadas por pessoas invisuais incluem leitores de ecrã, linhas de *braille* e *web browsers* de voz [55][69]. Todas estas transformam o conteúdo web tornando-o percetível para o utilizador [55]. Porém, para que isto se verifique, as páginas têm de ser desenvolvidas e implementadas devidamente permitindo compatibilidade com estas tecnologias, caso contrário a acessibilidade permanecerá um problema [69].

Os leitores de ecrã são *software* responsável pelo processamento conteúdo de *desktop* e dos *browsers*, convertendo-o em *text-to-speech* ou *braille* [50], identificando e interpretando

o conteúdo textual apresentado, permitindo depois que essa informação seja apresentada ao utilizador através de uma voz sintética (no caso de *text-to-speech*) [56]. A leitura é realizada de forma sequencial - feita de cima para baixo e da esquerda para a direita - uma linha de cada vez. Em alternativa, o utilizador pode navegar entre *headings* ou, ainda fazer uso da tecla Tab para saltar entre *links* [47] [56]. Como tal, os autores em [56] caracterizam a interação entre um utilizador invisual e uma página web como sendo uma atividade auditiva e com acesso sequencial, contrariamente à interação direta realizada pelos utilizadores comuns. Para além disso, a interação é também realizada através do teclado guiada através de *feedback* dado pelo leitor de ecrã [56]. Esta interação é limitada na medida em que os leitores de ecrã não conseguem descrever imagens a não ser que estas tenham um texto alternativo associado. Para além disso, estas tecnologias não permitem descrever a página na sua totalidade da mesma forma que é apresentada ao utilizador comum [47].

As linhas de *braille*, por sua vez, são dispositivos mecânicos que permitem apresentar o conteúdo de uma página web através de caracteres *braille* [65]. Estes são compostos por pontos atualizáveis que se elevam e recolhem, permitindo formar diferentes caracteres *braille* - entre 40 a 80 caracteres [50] - representando o conteúdo de uma página web [65]. No entanto, esta opção, para além de dispendiosa - com o preço a variar entre 900$ a 3000$ (aproximadamente entre 810€ e 2071€) -, requer ainda que os utilizadores saibam linguagem *braille*, o que nem sempre se verifica uma vez que esta é uma linguagem difícil de aprender para quem não é invisual desde nascença [65]. Comparando as duas tecnologias apresentadas - leitores de ecrã e linhas de *braille* - os primeiros são, geralmente, mais utilizados pelas pessoas invisuais durante a interação com conteúdo web [56].

Os *browsers* de voz são tecnologias semelhantes aos leitores de ecrã, porém, estes geralmente apenas processam conteúdo web, não sendo desenvolvidos com o fim de serem uma tecnologia de assistência mas sim uma alternativa (para dispositivos móveis ou semelhante, por exemplo) [50]. Um exemplo de um *browser* de voz, sendo também um dos mais antigos, é o *IBM's Home Page Reader* [58].

Os autores em [65] comparam os diferentes sistemas operativos Windows 10 e MacOS no que à acessibilidade diz respeito. O sistema operativo MacOS disponibiliza gratuitamente um *software* para leitura de ecrã - VoiceOver. No entanto, este leitor não é compatível com muitas aplicações, o que força a que os utilizadores a instalar outro leitor de ecrã. O mesmo acontece para o Windows 10, que também disponibiliza um leitor de ecrã integrado - Narrator. No entanto, este apresenta o mesmo problema que o VoiceOver na medida em que a compatibilidade com aplicações é limitada [65]. Por essa razão, os utilizadores vêem-se obrigados a instalar outro leitor de ecrã. Dentro dos leitores mais conhecidos estão o Jaws, criado por Freedom Scientific, Dolphin por Dolphin Computer Access Ltd. Technology House, System Access por Serotek e o Non Visual Desktop Access (NVDA). Todos estes *softwares* apresentados, à exceção do NVDA, requerem a compra de uma licença. O preço de aquisição pode variar entre 400$ a 1300$ (aproximadamente entre 360€ e 1171€). Por contraste, o NVDA não tem qualquer custo associado, no entanto, tratando-se de um *software* desenvolvido por uma comunidade (i.e., *open source*) a sua manutenção está dependente da referida comunidade de programadores, assim como de apoio financeiro externo [65].

## 3.3 Barreiras

Ao longo deste documento, foram apontadas algumas limitações e constrangimentos a que os utilizadores invisuais estão sujeitos - como, a título de exemplo, acontece quando olhamos para os custos envolvidos com determinadas tecnologias de assistência. Em [65], os autores referem o preço destas tecnologias como sendo uma barreira que os utilizadores invisuais enfrentam, sendo assim um obstáculo à sua interação com a web. Para além deste exemplo, existem variados problemas que consideramos pertinentes. Assim, nesta secção apresentamos com maior detalhe algumas barreiras recorrentes.

Como referido anteriormente, a web pode ser um meio de exclusão para as pessoas invisuais [66]. Embora seja claro que esta exclusão exista em larga escala, será conveniente perceber de forma mais detalhada a origem deste fenómeno, explorando barreiras geralmente encontradas por invisuais ao interagir com conteúdo web. A persistência dos problemas de acessibilidade e usabilidade podem tornar a navegação através de um leitor de ecrã numa experiência difícil e frustrante [58]. Mesmo quando as páginas web estão de acordo com as diretrizes do WCAG, as pessoas invisuais continuam a encontrar problemas de navegação na web [75]. Um estudo realizado em [74] contendo problemas encontrados por 32 utilizadores invisuais em 16 *websites* revelou que apenas 50.4% dos problemas encontrados são abrangidos nos critérios de sucesso do WCAG 2.0, sendo que 16.7% dos *websites* implementaram as técnicas recomendadas pelo mesmo, no entanto, estas não resolveram os problemas.

Como apontado, os autores em [66] destacam a importância do conhecimento sobre as necessidades destes utilizadores, o que não deverá ser descurado para evitar barreiras durante a interação. Em adição, os autores apresentam ainda sete razões (ou problemas) possíveis para a origem dessas barreiras que dificultam ou tornam impossível o acesso dos utilizadores invisuais à web. Primeiramente, existe informação que se torna difícil de representar por formas não-visuais. O segundo problema apresentado diz respeito à disposição dos elementos gráficos nas interfaces - a localização dos elementos na interface facilita a memorização por parte dos utilizadores invisuais. Porém, atualmente, os ambientes *multitask* promovem um posicionamento dos elementos menos estáticos. Os dois problemas seguintes, prosseguindo a análise dos autores, centram-se nos programadores, já que estes utilizam ferramentas que tornam o conteúdo parcialmente - ou totalmente - inacessível e, em adição, a falta de conhecimento e de treino sobre boas práticas com o fim de respeitar os princípios da acessibilidade. Um outro problema, ou barreira, encontra a sua causa na falta de recursos temporais e financeiros. Já o sexto problema apresentado tem por base ideias erradas que se mantém acerca da acessibilidade. Entre estes, encontram-se preconceitos sobre a estética em volta de *websites* acessíveis (i.e., a ideia de que um *website* acessível é, *a priori*, pouco apelativo e pouco original), preconceitos sobre questões técnicas (i.e., um site acessível é difícil de desenhar e de conceber) ou ainda aspetos económicos (i.e., um *website* acessível envolve custos elevados no seu processo de desenvolvimento e será pouco rentável ou lucrativo). Por fim, o sétimo e último problema descrito pelos autores diz-nos que a conformidade com os padrões da acessibilidade não garante, por si só, bons níveis de usabilidade da interface (como já discutido anteriormente neste documento) [66].

Em [59], os autores referem que os problemas enfrentados pelos utilizadores invisuais estão relacionados com informação e componentes da interface do utilizador que não são apresentados de forma a que o utilizador invisual consiga perceber, componentes da interface do utilizador e navegação que não é operável, informações e ações da interface do utilizador que não são compreensíveis e conteúdo que não é suficientemente robusto para ser interpretado pelas tecnologias de assistência. Assim, estes pontos reúnem o que poderá

ser visto como uma visão geral da origem destas barreiras, limitando o acesso comum a conteúdo web. Estas causas estão relacionadas com os quatro princípios de acessibilidade - *Perceivable*, *Operable*, *Understandable*, *Robust* - apresentados na próxima secção.

As pessoas invisuais, de uma forma geral, fazem uso de tecnologias de assistência e ferramentas quando acedem à web que permitem apresentar o conteúdo das páginas, de forma a que a informação das mesmas seja percetível para os utilizadores [55]. Para que tal seja possível, os programadores dos *websites* têm que assegurar que a apresentação do conteúdo web é independente da estrutura subjacente e que esta estrutura está corretamente programada. Isto porque, caso as páginas web não tenham sido desenhadas corretamente ou não tenham sido tidas em conta as diretrizes de acessibilidade, estas tecnologias e ferramentas podem enfrentar barreiras quando tentam alcançar o seu objetivo - devolver a informação aos utilizadores [55].

Em [66] os autores apresentam problemas relacionados com a utilização de leitores de ecrã resultantes da transposição de informação visual para informação auditiva. Como descrito no artigo, existem diferenças entre as modalidades auditivas e visuais, nomeadamente na informação visual que é oralmente descrita, havendo a possibilidade de empobrecimento de informação. Problemas como a imposição de uma ordem e a distância temporal entre elementos de informação torna difícil a combinação entre dois ou mais itens (e.g., cor, tamanho). Para além disso, existe ainda a questão temporal: cada vez que um utilizador seleciona um item, o leitor de ecrã inicia a leitura do conteúdo da página do ínicio, sendo que este percorre todo o conteúdo até alcançar o objeto desejado. Para além destes problemas indicados, em [56] os autores dão ainda alguns exemplos de problemas associados à interação com a web sendo esta uma atividade auditiva. São estes:

- Interação sequencial realizada na web - significando que o utilizador pode perder toda a informação contextual e perceber apenas uma porção do conteúdo;
- Dificuldade em filtrar conteúdo relevante numa página;
- Quando as páginas web apresentam layouts complexos, o *feedback* por parte do leitor de ecrã pode tornar-se ambíguo. Para além disso, existe o risco de os leitores de ecrã pronunciarem mal algumas palavras;
- A variedade de funcionalidades oferecidas pelos leitores de ecrã pode dificultar a memorização por parte do utilizador, assim como a utilização apropriada dos comandos e funções durante a interação com a web.
- Os recursos cognitivos são divididos em 3 partes: o utilizador tenta perceber o *web browser*, o *website* e o leitor de ecrã em simultâneo. Isto contribui para uma sobrecarga cognitiva durante a interação.

O Web Accessibility Initiative (WAI) [43] apresenta ainda algumas barreiras que poderão ser vistas como complemento a todos os problemas apresentados até agora:

- Elementos que não apresentam texto alternativo equivalente (e.g., imagens);
- Ausência de pistas de orientação não-visuais;
- Conteúdo de vídeo que não tem áudio alternativo;
- Mecanismos de navegação da página ou funções que são inconsistentes, imprevisíveis ou excessivamente complicados;

- *Websites* e *browsers* que não oferecem suporte total de teclado.

Como podemos constatar, as barreiras impostas aos utilizadores invisuais apresentam-se em quantidade significante e, embora documentadas e de características diversas, o maior defeito reside na sua origem (i.e., desenho e desenvolvimento dos *websites*). Esta falta de cuidado contribui não apenas para a discriminação - no sentido em que os utilizadores invisuais não têm o acesso facilitado a conteúdo web - mas também para o empobrecimento da experiência como um todo, independentemente dos utilizadores.

Um estudo realizado em [73] com o objetivo de comparar o humor de utilizadores invisuais e utilizadores sem qualquer deficiência - bem como a eficiência e a satisfação ao interagirem com um *website* acessível e um não-acessível - permitiu concluir que os dois grupos de utilizadores apontaram resultados mais positivos ao interagir com um *website* acessível do que com um *website* não-acessível. Em suma, verificamos que os utilizadores invisuais encaram regularmente com frustração as interações com a web e que, embora alguns problemas sejam alheios ao desenvolvimento dos *websites* (como mau funcionamento por parte dos leitores de ecrã) [67], boas práticas de acessibilidade podem prevenir grande parte destes problemas.

## 3.4 Web Content Accessibility Guidelines (WCAG)

A primeira versão do documento *Web Content Accessibility Guidelines* (WCAG 1.0) foi lançada em 1999 pela WAI, pertencente ao W3C [76][74] - comunidade internacional responsável pelo desenvolvimento de *standards* para conteúdo web [33]. O documento é composto por um conjunto de 14 diretrizes que visam auxiliar os programadores a tornar os *websites* mais acessíveis para pessoas com deficiências [74][55][76][44] e mais usável pelas pessoas no geral [55].

Para além destas diretrizes, este apresenta 65 *checkpoints* repartidos por cada uma dessas diretrizes [74][76][44]. A cada um está associado um nível de prioridade – entre 1 a 3 - baseado no impacto que o mesmo terá na acessibilidade [74][44]. O primeiro nível de prioridade diz respeito a *checkpoints* que têm de ser satisfeitos (*must*) para que um dado grupo de utilizadores consiga interagir com os documentos web. Já o segundo nível de prioridade diz respeito a *checkpoints* que devem ser satisfeitos (*should*), permitindo remover barreiras significativas face à acessibilidade. Por fim, o nível 3 engloba *checkpoints* que podem ser satisfeitos (*may*) - permitindo melhorar o acesso a documentos web. O nível de conformidade de um *website* pode ser representado por A, AA e AAA sendo que, se uma página web respeita todos os *checkpoints* de Prioridade 1, diz-se que o *website* está em conformidade com o nível A. Já para existir conformidade com o nível AA, é necessário que os *checkpoints* de prioridade 1 e 2 sejam respeitados. Por fim, se forem respeitados todos os *checkpoints* (prioridade 1, 2 e 3) diz-se que o *website* está em conformidade com o nível AAA [74].

Durante aproximadamente uma década, este documento serviu como *standard* para acessibilidade na web, permitindo consciencializar as pessoas para a criação de conteúdo mais acessível para pessoas com deficiências [74]. No entanto, alguns problemas relativos a esta versão levaram o W3C a lançar a segunda versão do mesmo [76]. Estes problemas relacionavam-se com algumas dificuldades sentidas por parte dos programadores durante a avaliação de acessibilidade de *websites* utilizando o documento. Entre eles, estavam a navegação ao longo deste, a linguagem utilizada no documento e o nível de conhecimento técnico exigido de forma a ser possível interpretar as diretrizes [74] e a necessidade de

atualização do documento [76].

O documento WCAG 2.0 foi lançado em 2008 pela WAI [74] tendo sido aprovado como ISO standard: ISO/IEC 405000:2012 em 2012 [55]. Em comparação com a versão anterior, o WCAG 2.0 não apresenta diretrizes escritas para tecnologias específicas [76] [14], sendo estas informações técnicas fornecidas em documentos separados [74]. Para além disso, o documento passou a estar organizado em 4 princípios, 12 diretrizes e 61 critérios de sucesso [76].

Os quatro princípios subjacentes a este documento são aqueles que são considerados como fundamentais para criar conteúdo acessível. São estes: *Perceivable*, *Operable*, *Understandable* e *Robust*. Todos estes princípios têm de ser verdadeiros de forma a que os utilizadores com deficiências possam utilizar a web [18].

O WCAG 2.0 [45] define os princípios apresentados da seguinte forma:

- *Perceivable*: Toda a informação e componentes da interface do utilizador devem ser apresentados aos utilizadores de forma a que seja percetível aos mesmos. Isto é, os utilizadores devem conseguir compreender a informação apresentada (não pode ser invisível a todos os seus sentidos).

- *Operable*: As componentes da interface do utilizador e a navegação devem ser operáveis, no sentido em que a interface não deve exigir qualquer tipo de interação que o utilizador não consegue realizar.

- *Understandable*: A informação e operações da interface do utilizador devem ser inteligíveis – o utilizador deve perceber a informação e operações da interface.

- *Robust*: O conteúdo deve ser suficientemente robusto, permitindo que este seja interpretado de forma confiável por tecnologias de assistência.

Como referido anteriormente, cada um destes princípios é composto por diretrizes e critérios de sucesso que permitem ir ao encontro desses princípios [18]. Essas diretrizes visam assegurar que o conteúdo é acessível para o máximo de utilizadores possível e capaz de ser apresentado de diferentes formas, permitindo responder às necessidades de cada utilizador [18]. Cada uma das diretrizes apresentadas no documento contém um conjunto de critérios de sucesso escritos na forma de declarações que podem ser facilmente testadas através de avaliação automática ou humana [14][18]. Cada um destes critérios é representado pelos já mencionados níveis de conformidade A (nível mais baixo), AA e AAA (nível mais alto) [14][45]. Em adição aos princípios, diretrizes e critérios de sucesso, o documento contém ainda técnicas suficientes e consultivas para cada diretriz e critério de sucesso [29]. Importa notar que estas técnicas são meramente informativas, ou seja, não são necessárias para que exista conformidade com o WCAG 2.0. Para tal, apenas é necessário cumprir os critérios de sucesso [29].

A versão atual do documento – WCAG 2.1 – foi lançada em 2018 e é uma extensão do documento anterior - todos os critérios de sucesso presentes no WCAG 2.0 estão também presentes na versão 2.1 [29]. Esta nova versão do documento surgiu com o objetivo de responder a necessidades de utilizadores com deficiências cognitivas ou de aprendizagem, assim como utilizadores com baixa visão, e ainda considerar acessibilidade *mobile* [30]. Conformidade com a versão atual - WCAG 2.1 - implica conformidade com WCAG 2.0 [52]. Considerando a extensa lista de *guidelines* oferecidas pelo WCAG, listamos abaixo aquelas que são mais relevantes no contexto do presente projeto:

- **Guideline 1.1 – Text Alternatives**: Fornecer alternativas textuais para qualquer conteúdo não-textual para que possa ser convertido em diferentes formas necessárias, como impressões, braile, discurso, símbolos ou linguagem mais simples.
- **Guideline 1.3 – Adaptable**: Criar conteúdo que possa ser apresentado de diferentes formas (por exemplo, *layouts* mais simples) sem perder informação ou descurar a estrutura.
- **Guideline 1.4 – Distinguishable**: Tornar mais fácil para os utilizadores ouvir o conteúdo.
- **Guideline 2.1 – Keyboard Accessible**: Tornar todas as funcionalidades disponíveis através do teclado.
- **Guideline 3.3 – Input Assistance**: Auxiliar os utilizadores a evitar e corrigir erros.
- **Guideline 4.1 – Compatible**: Maximizar a compatibilidade com *user agents* (e.g., *browsers*) atuais e futuros, incluindo tecnologias de assistência.

O conjunto de *guidelines* podem ser consultadas na sua totalidade no documento *online* (`https://www.w3.org/TR/WCAG21/`).

## 3.5 Accessible Rich Internet Applications (WAI-ARIA)

A especificação técnica WAI-ARIA (*Accessible Rich Internet Applications*) - publicada como recomendação do W3C em 2014 - fornece uma *framework* que permite melhorar a acessibilidade e a interoperabilidade de conteúdo e aplicações web [48][49], sendo por isso considerada um passo importante na criação de conteúdo e aplicações web mais acessíveis para pessoas com deficiências [34].

Esta especificação pretende lidar com problemas de acessibilidade encontrados em funcionalidades dos *websites* por pessoas com deficiências - nomeadamente no que diz respeito a problemas de pessoas que não conseguem utilizar rato e pessoas que façam uso de leitores de ecrã [48]. Algumas funcionalidades necessárias para tornar os *websites* usáveis por utilizadores de tecnologias de assistência, ou que dependam do teclado para navegar, não são nativamente incluídas nas linguagens utilizadas para a criação de conteúdo dinâmico (e.g., HTML, JavaScript, Cascading Style Sheets (CSS)) [35]. Como referido na secção 3.2, as tecnologias de assistência - como é o caso dos leitores de ecrã - permitem transformar a apresentação do conteúdo, fazendo com que o utilizador interaja com o mesmo da forma mais apropriada às suas necessidades (e.g., utilização do teclado para interagir com os elementos da interface invés do rato) . No entanto, para que tal seja possível, estas tecnologias têm de conhecer a semântica do conteúdo [49]. A semântica, no caso da acessibilidade, está relacionada com informação sobre o significado e o objetivo de um elemento da interface do utilizador [35]. Em WAI-ARIA 1.1 [49], a semântica divide-se em *roles*, *states* e *properties*. Um *role* define um elemento da interface ou a sua função, enquanto que *states* e *properties* consideram-se dois termos que se referem a características semelhantes - ambos permitem adicionar semântica ou significado adicional aos elementos [42] [49]. No entanto, a principal diferença entre ambos relaciona-se com a alteração dos valores das prioridades de cada um ao longo do ciclo de vida do programa. Ao contrário dos *states*, o valor das *properties* não tende a mudar ao longo da execução do programa [42] [49].

A WAI-ARIA é uma boa forma de cobrir áreas de uma aplicação web com problemas de acessibilidade que não conseguem ser tratados com HTML nativo. Através dos atributos ARIA pode ser adicionada informação em falta num determinado elemento, permitindo assim que o leitor de ecrã o interprete corretamente [61]. É importante reter a noção de que a ARIA não modifica o comportamento intrínseco dos elementos. Como exemplo, não permite que um elemento receba foco automaticamente, assim como não incorpora eventos de teclado de forma automática [61]. Outro ponto que importa realçar, passa pelo facto de não ser necessário redefinir semânticas originais ou padrão (por exemplo, um elemento `<input type="checkbox»` não necessita de um atributo `role="checkbox"` ).

O WAI-ARIA fornece aos programadores [48]:

- *Roles* para descreverem o tipo de *widget* apresentado (e.g., "*menu*", "*button*");
- *Roles* para descreverem a estrutura da página web(e.g., "*headings*");
- *Properties* que descrevem o estado do *widget* (e.g., "*checked*"no caso de uma checkbox);
- *Properties* para definir regiões que podem sofrer alterações dentro da página web.

Por fim, dentro dos diversos atributos disponibilizados pela WAI-ARIA, destacamos e descrevemos de seguida aqueles que tiveram impacto no trabalho desenvolvido no segundo semestre - `aria-label`, `aria-required`, `aria-describedby`, `aria-hidden` e `aria-live`. Uma listagem completa de todos os atributos fornecidos pode ser consultada em [49].

O atributo `aria-label` permite associar uma *string* a um elemento para que a mesma seja interpretada como nome do elemento [49]. Este atributo pode ser usado quando temos algum tipo de indicação visual sobre o propósito de um determinado elemento, como um botão que utilize apenas uma imagem para indicar o seu fim. Nestes casos, é necessário clarificar o propósito deste botão para todos os utilizadores que não tenham acesso a essa indicação visual.
Já o atributo `aria-required`, permite informar as tecnologias de assistência relativamente à obrigatoriedade do elemento que contém o atributo [49]. Por sua vez, o atributo `aria-describedby` permite identificar (através de referência por *id*) um ou mais elementos como descrição do elemento que contém o atributo [49].
Adicionalmente, a WAI-ARIA fornece um mecanismo para esconder conteúdo das tecnologias de assistência - atributo `aria-hidden` [49]. Porém, este atributo não previne que o elemento receba foco, sendo necessário para tal adicionar ainda o atributo `tabindex` com valor -1 [13].

Por último, apresentamos o atributo `aria-live`. Essencialmente, este permite identificar, aos leitores de ecrã, quais as partes da página web que irão sofrer alterações [49]. Assim,qualquer atualização deverá ser transmitida aos utilizadores, independentemente de onde estes se encontrem na página [60]. Os valores que podem ser associados a este atributo dizem respeito ao grau de importância [49]. São estes:

- `aria-live="polite"`, que permite que as tecnologias de assistência alertem o utilizador da atualização; porém, regra geral, as atualizações não interrompem a tarefa corrente (prioridade baixa).
- `aria-live="assertive"`, que permite que as tecnologias de assistência interrompem a tarefa corrente para anunciar as atualizações (prioridade alta).

- `aria-live="off"` , que permite suspender interrupções promovidas pelo atributo aria-live.

Em adição, é-nos disponibilizado o documento WAI-ARIA Authoring Practices 1.1, que fornece orientações sobre como usar a WAI-ARIA para criar aplicações web acessíveis. Essencialmente, é um documento complementar que recomenda abordagens para tirar partido das especificações apresentadas, descrevendo considerações que possam não ser tão evidentes na WAI-ARIA. O documento pode ser consultado em [35]
Por forma a compreender melhor a importância de boas práticas semânticas, a seguinte subsecção descreve alguns exemplos que ilustram não apenas a pertinência destas especificações técnicas como ainda uma visão geral sobre a sua implementação.

### 3.5.1 Exemplos de Uso

As tecnologias de assistência identificam os elementos interpretando os seus valores semânticos. Tal acontece por definição com elementos HTML, como o caso de botões, por exemplo. A Figura 3.1 abaixo apresenta uma linha de código HTML que representa um botão com o texto *"Click Me"*. Desta forma, a *tag* `<button>` é automaticamente interpretada pelas tecnologias de assistência como um botão, passando assim esta informação ao utilizador.

```
<button>Click Me</button>
```

Figura 3.1: Exemplo de linha de código HTML (botão).

Considerando os leitores de ecrã, as semânticas definidas pelo WAI-ARIA controlam a interpretação (ou renderização) das experiências não-visuais. Assim, o mau uso deste conjunto de propriedades deturpa as experiências visuais - fazendo com que sejam potencialmente interpretadas de forma errónea [35]. No guia de utilização do WAI-ARIA [35], são apresentados dois princípios de uso destas especificações. O primeiro princípio diz respeito ao compromisso do programador com as suas declarações. Observando o exemplo de código na Figura 3.2, verificamos que o programador utilizou uma `<div>` para criar um botão (o que poderá ser feito em JavaScript e CSS, em alternativa dos botões definidos em HTML puro). Assim, a única informação existente para os leitores de ecrã é o atributo *'role'*, que nos diz que o conteúdo daquele excerto é um botão. Como é compreensível, esta atribuição fica do lado do programador. Desta forma, o primeiro princípio do WAI-ARIA defende que esta declaração significa um compromisso por parte do programador, certificando-se que aquela `<div>` incorpora, efetivamente, o comportamento esperado de um botão.

```
<div role="button">Place Order</div>
```

Figura 3.2: Exemplo de linha de código HTML (`<div>`) com *role* botão.

O segundo princípio apresentado alerta para o facto de esta especificação técnica ser simultaneamente poderosa e perigosa. Tal acontece porque, como apontado, o programador passa a ter em mãos a capacidade de definir todos os elementos por forma a serem interpretados pelas tecnologias de assistência. Como defendido pelos autores da especificação [35], este é o poder do WAI-ARIA: poder reescrever e introduzir novos significados nos componentes que constituem uma página web. Porém, os autores defendem que esta liberdade de poder é também, por extensão, um dos perigos do WAI-ARIA, existindo a possibilidade de um programador reescrever erradamente atributos semânticos, ou simplesmente defini-los de forma errada. A imagem abaixo (Figura 3.3) representa um excerto de código HTML que ilustra isto mesmo: uma tabela à qual foi atribuída o *role 'list'*. Isto significa que um utilizador invisual, por exemplo, iria ter a percepção que estaria perante uma lista e não uma tabela - fruto da interpretação da tecnologia de assistência.

```
<table role="list">

</table>
```

Figura 3.3: Exemplo de linha de código HTML (tabela) com *role* lista.

Em suma, o WAI-ARIA oferece diversos conjuntos semânticos que promovem a acessibilidade geral de uma página. No entanto, para tal se verificar, é importante ter em conta que o código base merece uma análise cuidada para que a qualidade semântica se verifique e, consequentemente, permita às tecnologias de assistência interpretar e descrever de forma fidedigna o contéudo presente na página web. Todo o contéudo referente ao atributos *role* - assim como os *states* e *properties* - podem ser consultados na sua totalidade no documento *on-line* (`https://www.w3.org/TR/wai-aria-1.1/`), onde se encontram listados exemplos relativos à sua implementação.

## 3.6 Métodos de avaliação

Durante o processo desenvolvimento ou redesenho de um *website*, é necessário avaliar atempadamente a acessibilidade de forma a ser possível detetar eventuais erros, permitindo facilitar todo o processo inerente à sua resolução [46]. No entanto, como apontado anteriormente, uma boa implementação da acessibilidade nem sempre se verifica (secção 3.3). Em [71], os autores atribuem a falta de acessibilidade na grande maioria dos *websites* a dois fatores: falta de experiência na área de acessibilidade por parte dos programadores e escassez de informação no que diz respeito a formas eficientes para avaliar problemas de acessibilidade em páginas web. Consequentemente, sabendo que as boas práticas relativamente à acessibilidade tendem a ser descuradas, torna-se necessária a existência de métodos para avaliar isso mesmo.

A avaliação da acessibilidade referente a conteúdo web é o processo que permite avaliar a facilidade com que pessoas com algum tipo de deficiência - neste caso, utilizadores invisuais - conseguem interagir com este mesmo conteúdo [68]. Esta avaliação resulta da combinação de aspetos técnicos - como avaliação da conformidade com os *standards* e diretrizes - e aspetos não-técnicos através da avaliação realizada por utilizadores finais [68].

Existem três técnicas diferentes que permitem realizar a avaliação de acessibilidade - uma

automática e duas manuais [69]. A primeira consiste na utilização de ferramentas de avaliação - *software* ou serviços *on-line* - que permitem ajudar a determinar se o conteúdo vai ao encontro das diretrizes de acessibilidade [69]. Esta técnica não necessita de intervenção humana e apresenta benefícios a nível de custos [68]. As ferramentas automáticas foram criadas com o objetivo de auxiliar os programadores a verificar a acessibilidade dos *websites*, uma vez que perceber as diretrizes de acessibilidade e verificar a conformidade dos *websites* com as mesmas pode ser difícil [58]. Em [55] são apresentados alguns exemplos de ferramentas para avaliação de acessibilidade, entre eles AChecker, The Called Markup Validator e aXe. O Called Markup Validator é um serviço que permite validar os documentos web e suporta a maioria das linguagens *markup* (e.g., HTML). A ferramenta ACheker produz um relatório com todos os problemas relativos à conformidade com vários padrões de acessibilidade, incluindo o WCAG. A ferramenta aXe está disponível como extensão do Mozilla Firefox e do Google Chrome que analisa o conteúdo de uma página web e apresenta uma lista de erros encontrados, juntamente com um *link* com informação relativa à diretriz de acessibilidade em questão [55]. Para além disso, apresenta ainda diretrizes para lidar com o problema e informação acerca da gravidade do mesmo. No entanto, embora eficientes a detetar alguns problemas de acessibilidade [67], estas ferramentas por si só não são suficientes para determinar se o *website* está de acordo com as diretrizes, necessitando, para tal, de uma avaliação realizada por um entidade experiente [46]: questões relacionadas com a significância de um texto alternativo de uma imagem não podem ser resolvidas de forma automática [58] .

As técnicas manuais são compostas por duas avaliações distintas: avaliação dos *websites* realizada por indivíduos experientes - tendo em conta as diretrizes e os princípios de acessibilidade -, e uma avaliação realizada por utilizadores finais enquanto interagem com os *websites* [69]. Em [58] os autores relatam um estudo no qual se observou que a utilização de um leitor de ecrã durante a avaliação por parte dos programadores contribui para uma validação mais correta.

Relativamente à primeira técnica de avaliação manual, o autor em [1] aponta uma avaliação heurística. Com base na definição apresentada, onde nos é descrito que numa avaliação heurística *"especialistas avaliam se cada elemento está em conformidade com os principios de usabilidade estabelecidos"*, o mesmo é transposto para a acessibilidade. Isto é, no cenário de uma avaliação heurística aplicada à acessibilidade, especialistas da mesma avaliam a se os elementos se apresentam conforme os princípios de acessibilidade [1].
Ainda sobre a avaliação de acessibilidade, o autor apresenta *Design Walkthroughs* aplicados à acessibilidade. Geralmente, este tipo de abordagem visa encontrar potenciais problemas de usabilidade, guiando um utilizador representativo através de tarefas a partir de *mockups*. Estes são representações primárias de elementos da interface (como janelas ou menus *dropdown*) que são alterados por um membro da equipa ao longo da interação [1]. Para transpor esta metodologia com a finalidade de avaliar a acessibilidade, é proposto o envolvimento de *personas* que representem utilizadores com deficiências, incluindo cenários que contenham estratégias adaptativas para completar tarefas. Utilizando a ilustração exata do autor, um exemplo neste cenário seria um utilizador que desempenhasse uma pessoa invisual, e um membro da equipa desempenharia o papel de um leitor de ecrã [1].
Por fim, podemos ainda aplicar testes de usabilidade como forma e reunir dados qualitativos e quantitativos sobre um determinado produto, podendo efetuar alterações de modo a incluir participantes com deficiências. No entanto, sabendo que os testes de usabilidade nos podem fornecer informação importante sobre o produto desenvolvido, não nos permite averiguar a sua conformidade com os padrões de acessibilidade [1].

Contudo, embora estas três técnicas (manuais e automática) sejam semelhantes, cada uma delas é mais adequada para lidar com um tipo específico de problema [68]. Será ainda

relevante sublinhar que as ferramentas de avaliação automática podem ser benéficas a nível de eficiência - poupando tempo e esforço. No entanto, deverão ser vistas como um complemente à avaliação humana e não uma alternativa [1].

# Capítulo 4

# Trabalho Preliminar

No presente capítulo descrevemos sucintamente as atividades realizadas no contexto do estágio curricular ao longo do primeiro semestre.

Numa fase inicial do estágio, estabeleceu-se um período de familiarização com as tecnologias utilizadas no projeto PULSAR. Posteriormente, foi proposta a integração nos *sprints* da equipa - tendo sido atribuída uma tarefa - por forma a adquirir experiência não apenas com as tecnologias mas principalmente com a plataforma PULSAR, bem como os métodos de trabalho da equipa. Sucintamente, a primeira tarefa realizada no PULSAR tinha como objetivo adicionar uma nova funcionalidade no módulo referente aos projetos a decorrer na empresa. Cada projeto tinha *workpackages* associadas, sendo que o objetivo era adicionar - na secção de recursos de uma *workpage* - um *toggle button* que permitisse variar entre os recursos atuais e os planeados. No caso dos recursos planeados, era necessário serem apresentadas três colunas referentes às unidades de cada recurso, custo esperado e esforço esperado. Estes eram três campos que seriam editáveis, sendo que qualquer alteração num deles deveria recalcular os outros, automaticamente. No caso dos recursos atuais, os campos apresentados seriam o esforço atual e o custo atual, sendo que, neste caso não seriam editáveis.

Durante esse período teve ainda lugar uma reunião com um colaborador invisual da empresa, que nos reportou os módulos do PULSAR nos quais sentia maiores dificuldades - Pedido de Férias/Ausências, Reportar Esforço, Mapa de Despesas e Pedido de Viagem.

Tendo em conta a relevância dos módulos, foi decidido junto do orientador e gestor de equipa que a Pedido de Férias/Ausências e o Reportar Esforço seriam prioritários a tratar no segundo semestre. A maior dificuldade reportada pelo utilizador relativamente a estes dois módulos estava relacionada com a codificação de cores apresentada anteriormente na secção referente aos dois módulos (secções 2.3.1 e 2.3.2). Este *feedback* de cores apresentado, tanto no calendário do módulo de Reportar Esforço como no de Pedido de Férias/Ausências, não é adequado para um utilizador invisual, tornando a tarefa de verificação mais exaustiva, já que têm de percorrer o calendário em busca de informação reportada. Foram ainda identificados problemas como falta de *headings* - que auxiliam na navegação nestes módulos -, botões que não são reconhecidos pelo leitor de ecrã, problemas associados à interação com *dropdowns* e comportamentos irregulares do foco. Toda esta informação foi considerada útil não apenas para uma primeira identificação de problemas mas também para a priorização de tarefas. Assim, os módulos destacados como objetivos secundários (Mapa de Despesas e Pedido de Viagem) não foram analisados de forma tão detalhada como os restantes, dando prioridade à definição dos requisitos e funcionalidades dos módulos principais. Uma vez que a implementação da acessibilidade nestes módulos se

encontrava dependente do tempo consumido pelos primeiros, os casos de uso dos módulos secundários seriam apenas definidos consoante o tempo disponível para a implementação dos mesmos.

Concluída a tarefa, realizou-se um estudo focado nas diretrizes do documento WCAG 2.1 e nas especificações técnicas WAI-ARIA tendo em vista uma apresentação para toda a equipa, permitindo assim dar a conhecer estes dois documentos - e respectivos conceitos - aos restantes membros, sendo que toda esta pesquisa contribuiu também como fonte bibliográfica para o Trabalho Relacionado (Capítulo 3). Após a pesquisa e estudo de documentação considerada relevante para ao âmbito do presente estágio ter sido iniciada, o restante tempo foi destacado para a realização do presente documento, assim como a definição dos primeiros requisitos a implementar no segundo semestre - tendo em conta os casos de uso retirados da reunião com o colaborador invisual. Por fim, destacámos ainda algum tempo para um primeiro estudo dos leitores de ecrã, através de experimentação local.

Todas as tarefas realizadas - assim como o tempo consumido pelas mesmas - pode ser consultado na Figura 9.1 do Capítulo 9, assim como o plano previsto para o segundo semestre (Figura 9.2).

# Capítulo 5

# Especificação de Requisitos

No presente capítulo são descritos os requisitos funcionais assim como o único requisito não-funcional (i.e., atributo de qualidade) definidos no âmbito do presente estágio curricular.

Os requisitos definidos abaixo englobam apenas os módulos especificados como objetivos Principais - Pedido de Férias/Ausências e Reportar Esforço - trabalhados ao longo do segundo semestre. Assim, não serão detalhados requisitos para os módulos definidos como objetivos secundários - Mapa de Despesas e Pedido de Viagem -, uma vez que estes não chegaram a ser implementados.

Os requisitos funcionais, escritos na forma de caso de uso, permitem definir as funcionalidades que deverão ser implementadas de forma a tornar os dois módulos referidos acessíveis a utilizadores invisuais. Por sua vez, o atributo de qualidade especificado pretende qualificar os requisitos funcionais apresentados [57].

## 5.1 Casos de Uso

Os casos de uso descrevem as interações entre o ator (i.e., utilizador invisual) e o sistema (neste caso, os dois módulos definidos como objetivos principais). Desta forma, são descritas as funcionalidades expectáveis, de forma a tornar os módulos destacados acessíveis a pessoas invisuais.

Os presentes casos de uso foram definidos com base em quatro fatores: as necessidades e problemas identificados a partir dos testes realizados com colaboradores invisuais da empresa, considerações apresentadas no documento WAI Authoring Practices [35] tidas como pertinentes, bem como as diretrizes descritas no documento WCAG 2.1 [52] que são aplicáveis ao presente contexto.

Os primeiros casos de uso foram definidos ainda no primeiro semestre, após uma avaliação preliminar da acessibilidade com um dos colaboradores invisuais - antes de terem sido implementadas alterações -, na qual foram reportadas as principais dificuldades sentidas durante a interação com os mesmos (descrito mais à frente na secção 7.2.1). Assim, serviu como base para a definição dos primeiros casos de uso, sendo que estes foram sendo atualizados consoante o *feedback* recebido ao longo dos testes com os utilizadores e, complementarmente, informação recolhida sobre acessibilidade.

Cada caso de uso será definido com recurso às seguintes propriedades [9][32]:

- **Ator**: Quem ou o que interage com o sistema.

- **Nome**: Identificar e descrever brevemente o caso de uso.
- **Descrição:** Descreve o propósito do caso de uso.
- **Pré-condições:** Condições que se devem verificar antes de o caso de uso iniciar.
- **Cenário Principal:** Sequência de passos que descreve o caso de uso caso nada corra mal.
- **Cenários Alternativos:** Variações do cenário principal. (Esta propriedade apenas será especificada caso se verifique no caso de uso em questão).

Cada caso de uso apresentado tem uma prioridade associada, podendo esta ser Alta, Média ou Baixa. Porém, uma vez que os casos de uso apresentados são necessários para garantir a acessibilidade dos módulos principais, todos foram definidos com prioridade Alta. Como referido na secção 1.2, no âmbito do presente estágio tem-se como fator de sucesso a implementação dos casos de uso de prioridade Alta.

Abaixo apresentamos os casos de uso definidos para o módulo Pedido de Férias/Ausências (Tabela 5.1) e para o módulo Reportar Esforço (Tabela 5.2). Por questões de legibilidade e facilidade de consulta, os casos de uso listados nestas tabelas são versões simplificadas dos mesmos. As tabelas de descrição completa de cada caso de uso podem ser consultada no Apêndice A deste documento.

| **ID** | **Nome** |
|---|---|
| CS-1 | Navegar por *headings* no módulo Pedido de Férias/Ausências |
| CS-2 | Identificar elementos interativos |
| CS-3 | Definir o foco inicial no módulo Pedido de Férias/Ausências |
| CS-4 | Navegar entre elementos do módulo Pedido de Férias/Ausências |
| CS-5 | Definir foco após interação com um elemento do módulo Pedido de Férias/Ausências |
| CS-6 | Definir foco inicial na janela *popup* |
| CS-7 | Navegar entre elementos da janela *popup* |
| CS-8 | Definir foco ao fechar a janela *popup* |
| CS-9 | Aceder à informação de férias/ausências |
| CS-10 | Receber *feedback* das opções do *dropdown* |
| CS-11 | Confirmar ação realizada |
| CS-12 | Identificar campos obrigatórios |
| CS-13 | Disponibilizar *feedback* durante a realização de uma ação no módulo Pedido de Férias/Ausências |
| CS-14 | Aceder à informação sobre os atalhos de teclado |
| CS-15 | Aceder à informação do dia |
| CS-16 | Navegar para o dia atual no calendário no módulo Pedido de Férias/Ausências |
| CS-17 | Navegar entre dias do calendário com eventos |

Tabela 5.1: Casos de Uso (Pedido de Férias/Ausências)

| ID | Nome |
|---|---|
| CS-18 | Navegar por *headings* no módulo Reportar Esforço |
| CS-19 | Definir foco inicial no módulo Reportar Esforço |
| CS-20 | Navegar entre elementos interativos do módulo Reportar Esforço |
| CS-21 | Definir foco após interação com um elemento do módulo Reportar Esforço |
| CS-22 | Definir foco no painel lateral |
| CS-23 | Definir foco inicial na janela de recorrência |
| CS-24 | Navegar entre elementos da janela de recorrência |
| CS-25 | Definir foco ao fechar a janela de recorrência |
| CS-26 | Definir foco após interação com um elemento da janela de recorrência |
| CS-27 | Aceder à informação apresentada no calendário |
| CS-28 | Aceder à informação do esforço através das legendas |
| CS-29 | Identificar campos obrigatórios |
| CS-30 | Disponibilizar *feedback* durante a realização de uma ação no módulo Pedido de Reportar Esforço |
| CS-31 | Aceder à informação sobre os atalhos de teclado |
| CS-32 | Disponibilizar atualizações da legenda do dia |
| CS-33 | Aceder à informação do dia |
| CS-34 | Navegar para o dia atual no calendário |
| CS-35 | Navegar para o dia selecionado no calendário |

Tabela 5.2: Casos de Uso (Reportar Esforço)

## 5.2 Atributos de Qualidade

Seguindo o livro Software Architecture in Practice (Len Bass et al., 2015), os atributos de qualidade (ou requisitos não-funcionais) são uma propriedade mensurável ou testável de um sistema e indicam o quão bem o sistema satisfaz as necessidades dos *stakeholders* [57].

Nesta secção é apresentado o atributo de qualidade definido para os módulos especificados.

### 5.2.1 Usabilidade

O atributo de qualidade usabilidade lida com a facilidade com que um utilizador consegue protagonizar uma determinada tarefa e o tipo de suporte fornecido ao utilizador pelo sistema [57]. No presente contexto, pertende-se garantir que as alterações realizadas no âmbito da acessibilidade são usáveis para os utilizadores invisuais.

A tabela abaixo apresenta um cenário que pretende descrever o atributo de qualidade especificado. O cenário é composto por 6 características, são estas:

- **Estímulo:** Evento que chega ao sistema (e.g., uma operação do utilizador);
- **Fonte de Estímulo:** Origem do estímulo;
- **Resposta:** Atividade resultante da chegada do estímulo;

- **Medida de Resposta:** Permite determinar se a resposta do sistema é satisfatória;
- **Ambiente:** Conjunto de condições no qual o cenário acontece.
- **Artefacto:** Porção do sistema ao qual o requisito se aplica.

| **ID** | AQ-1 |
|---|---|
| **Estímulo** | O utilizador invisual tenta realizar tarefas nos módulos de forma eficiente. |
| **Fonte de Estímulo** | Utilizador invisual |
| **Resposta** | Os módulos disponibilizam as funcionalidades necessárias para que o utilizador invisual consiga realizar as tarefas de forma eficiente. |
| **Medida de Resposta** | • Rácio de tarefas concluídas com sucesso relativamente ao total de tarefas;<br>• Problemas identificados durante a execução das tarefas;<br>• Dificuldade associada à realização das tarefas. |
| **Ambiente** | Módulos em condições normais de funcionamento |
| **Artefacto** | Módulos Pedido de Férias/Ausências e Reportar Esforço |

Tabela 5.3: Atributo de Qualidade (Usabilidade)

# Capítulo 6

# Implementação

No presente capítulo, apresentamos as soluções implementadas ao longo do estágio, mais concretamente no segundo semestre, que pretendem ir ao encontro das necessidades dos colaboradores invisuais da empresa. Todas as implementações descritas neste capítulo têm por base os requisitos funcionais (casos de uso) especificados no Capítulo 5 (secção 5.1).

Embora alguns processos de implementação - e mesmo algumas soluções encontradas - tenham sido aplicadas em ambos os módulos de forma semelhante, considerámos que se tornaria mais claro retratar as abordagens a cada módulo separadamente. Desta forma, o capítulo apresentado está dividido em duas secções: implementações no módulo Pedido de Férias/Ausências e Reportar Esforço.

O processo de implementação visava responder aos objetivos traçados, isto é, tornar os dois módulos especificados acessíveis a colaboradores invisuais da empresa - que, até ao momento, são três. Este processo englobou, também, a implementação dos comportamentos especificados no documento WAI Authoring Practices [35], nomeadamente em componentes que, até ao momento, não se encontravam acessíveis. Para além disso, os dados recolhidos inicialmente, assim como o contacto permanente com os colaboradores invisuais, foram fontes cruciais para o melhoramento dos módulos a nível de acessibilidade, tendo tido impacto na soluções implementadas.
Em cada subsecção - referente a uma implementação de acessibilidade específica de cada módulo -, expomos os problemas encontrados ou reportados, pretendendo com isto ilustrar as dificuldades com as quais os utilizadores invisuais se deparavam. Após a exposição destas dificuldades, apresentamos as soluções seguidas para suprimir as mesmas.
Por questões de clareza, considerando que as implementações de acessibilidade relativas ao comportamento do foco foram as mais exaustivas, as subsecções referentes ao mesmo encontram-se divididas, como veremos mais à frente.

## 6.1 Módulo Pedido de Férias/Ausências

As diferentes implementações de acessibilidade no módulo Pedido de Férias/Ausências, podem ser agrupadas em seis categorias: *Headings*, Nomes Acessíveis, Comportamento do Foco, Transposição de Cores, Atalhos de Teclado e Notificações do Utilizador. Desta forma, estas correspondem às subsecções que apresentamos de seguida.

### 6.1.1 *Headings*

A presente subsecção pretende descrever o processo de implementação do caso de uso CS-1 (Tabela 5.1), referente à utilização de *headings* para navegação no presente módulo. A falta de *headings* foi um problema identificado pelo colaborador invisual com o qual tivemos o primeiro contacto, tendo este também mencionado que a existência de *headings* no módulo serviria para facilitar - signitivamente - o processo de navegação no mesmo, sendo portanto uma lacuna que necessitava ser suprimida.

Numa página web, a definição de *headings* é realizada através da atribuição de *tags* `<h>` a elementos, sendo que estes podem variar entre diferentes níveis (i.e., `<h1>` a `<h6>`), criando uma relação hierárquica entre os diferentes conteúdos apresentados na página.
Um *heading* de nível 1 - definido pela *tag* `<h1>` - é utilizado para definir o título de uma página web [31]. Por essa razão, é aconselhada a utilização de apenas um *heading* de nível 1 (<h1>) por página [54], sendo que deverá sempre existir pelo menos um [31]. A utilização de *headings* de nível dois (*tag* `<h2>`) permite representar subsecções numa página e, consequentemente, os *headings* de nível três permitem dividir as subsecções, assim sucessivamente, até ao *heading* de nível 6 [31].

Em contextos visuais, os *headings* são geralmente associados a texto com fonte maior e mais carregada, facilitando a percepção dos utilizadores sobre a organização e estrutura da página [31].
Embora os utilizadores invisuais não tirem partido do aspeto visual destes elementos, o facto de estarem definidos numa página permite que estes consigam não só adquirir uma melhor perceção da estrutura da página, bem como do conteúdo que poderão encontrar na mesma. Tal é alcançado tirando partido da utilização do leitor de ecrã e das funcionalidades existentes no mesmo (e.g., listar os *headings* presentes numa página, navegar para o *heading* seguinte). Caso contrário, se não forem definidos este tipo de elementos, os utilizadores invisuais terão de depender do leitor de ecrã para que este leia toda a página de forma a terem conhecimento do conteúdo presente na mesma. Porém, como também foi verificado através da análise conjunta com os colaboradores invisuais, regra geral, estes não esperam que os leitores de ecrã leiam toda a página, uma vez que tal se torna num processo moroso. Assim, os *headings* permitem percepcionar a estrutura e conteúdo da página de uma forma mais prática e rápida. A título de exemplo, a Figura 6.1 apresenta o resultado de uma pesquisa pelos *headings* disponíveis na página, através do leitor de ecrã NVDA.

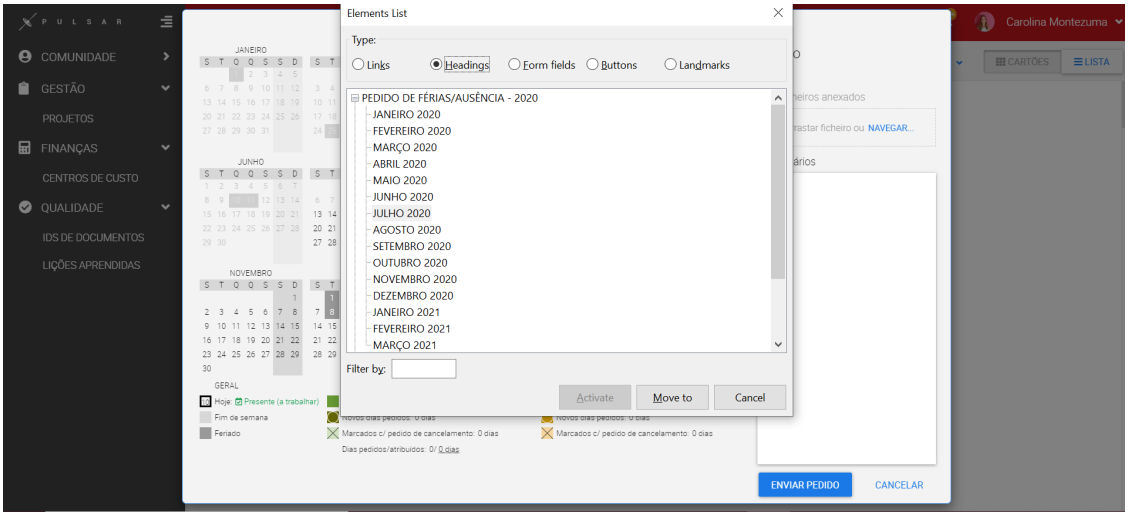


Figura 6.1: Lista de *headings* (NVDA)

Adicionalmente, os *headings* permitem agilizar o processo de navegação dos utilizadores invisuais pela página através das funcionalidades do leitor de ecrã apresentadas acima. Desta forma, o leitor de ecrã irá realizar a leitura da página apenas a partir desse ponto.

No âmbito do presente requisito, foram definidos no total 21 *headings*: nível 2 (5), nível 3 (15) e nível 4 (1). No PULSAR, apenas existia um *heading* que tinha sido definido anteriormente, sendo este o título da página (`Pessoas`), definido com *heading* de nível 1 (`<h1>`) (Figura 2.2). Ao observar a figura, notamos que o módulo Pedido de Férias/Ausências consiste numa janela secundária sobreposta à página principal (i.e., *dialog*, especificado mais à frente na subsecção 6.1.3), cujo título tinha já sido definido com a *tag* `<h1>`. Como tal, não foi adicionado mais nenhum *heading* de nível 1.

A solução para a definição de *headings* no módulo teve em vista a divisão do mesmo nas seguintes partes: calendário, anexo de ficheiros e legendas referentes à informação apresentada no calendário (Geral, Férias e Ausências e Outros). Para tal, os elementos `PEDIDO DE FÉRIAS/AUSÊNCIAS - 2020, Pedido, Geral, Férias e Ausências e Outros` (Figura 2.2), foram definidos utilizando a *tag* `<h2>`.
Todos os títulos referentes aos meses apresentados no calendário (nome dos meses) foram definidos como *headings* de nível 3 (`<h3>`). A implementação de *headings* em cada um dos títulos dos meses visava agilizar o processo de navegação aos utilizadores invisuais pelos diferentes meses do calendário. Como referido, os leitores de ecrã disponibilizam funcionalidades específicas que permitem, por exemplo, saltar para o *heading* seguinte (ou anterior) que, no contexto do calendário, permitiria navegar para o mês seguinte ou anterior do calendário. Durante testes realizados com os utilizadores no presente módulo, foi notório que a utilização de *headings* simplificou este processo, assim como facilitou a implementação de atalhos de teclado no presente módulo, discutida mais à frente subsecção 6.1.5.
Por último, a Figura 6.2 ilustra a janela *popup* que é apresentada de cada vez que é selecionado um dia no calendário. Na presente janela *popup*, foi definido como *heading* de nível 4 (`<h4>`) o título da mesma (i.e., data referente ao dia selecionado).

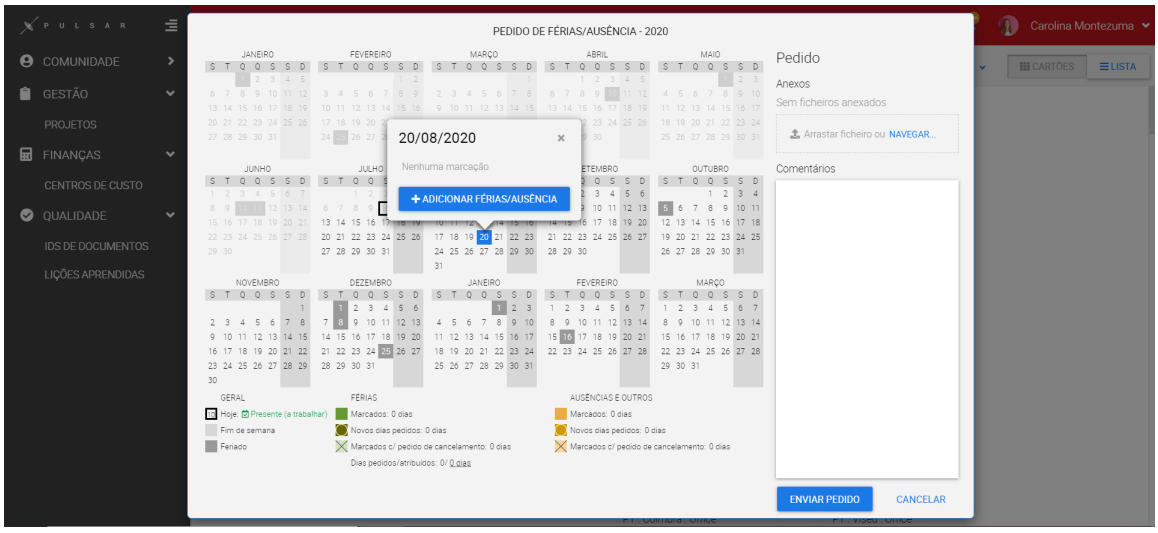


Figura 6.2: Janela *popup* do dia selecionado

Um outro problema identificado pelo colaborador invisual na fase inicial (Capítulo Testes de *Software*, secção 7.2.1), dizia respeito ao aparecimento desta janela *popup*, mais concretamente, ao facto de o foco não se mover automaticamente para um elemento pertencente à mesma (detalhado na subsecção 6.1.3). Por essa razão, também já tinha sido reforçado pelo utilizador invisual a importância da definição de pelo menos um *heading* na janela

*popup*, de forma a contornar este problema referente ao foco, simplificando o processo de mover o foco para a janela. Isto é, se fosse definido um *heading* na janela *popup*, o colaborador invisual poderia tirar partido das funcionalidades dos leitores de ecrã para navegar para o *heading* definido e, dessa forma, mover o foco para a janela *popup*.
A atribuição das *tags* `<h2>`, `<h3>` e `<h4>` aos elementos referidos provocou, também, uma mudança - não desejada - no aspeto visual dos mesmos (i.e., alteração do tamanho da fonte e espessura da mesma), como se pode observar na Figura 6.3.

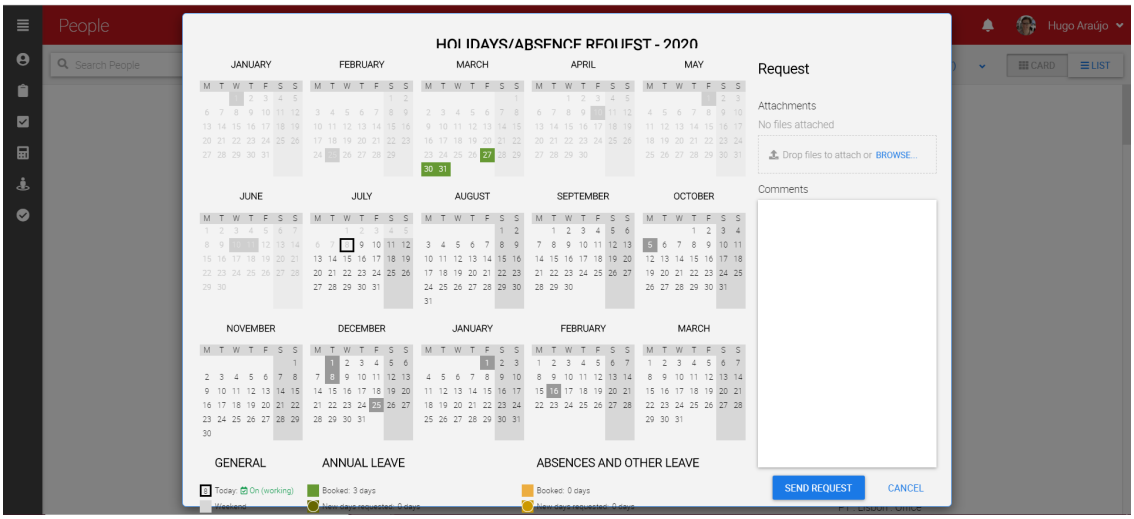


Figura 6.3: *Headings (após implementação das tags)*

Por conseguinte, foi necessário ainda reverter as alterações visuais resultantes da atribuição das *tags* através da utilização de propriedades CSS. Em suma, foram ajustados os valores das propriedades `margin`, `font-weight` e `font-size`, garantindo assim que o aspeto visual do módulo permanecesse igual.

### 6.1.2 Nomes Acessíveis

A presente subsecção diz respeito a um problema que, tal como o a falta de *headings*, também este é comumente reportado por utilizadores invisuais: a ausência de nomes acessíveis associados a elementos interativos da interface (e.g., botões).
O nome acessível de um elemento permite identificar a finalidade do mesmo e distinguilo - perante os leitores de ecrã - dos outros elementos presentes numa página. Todos os elementos presentes numa página que recebem foco e com os quais o utilizador consegue interagir deverão conter um nome acessível [39].
Quando os elementos contêm um texto visível associado, os leitores de ecrã interpretam-no como sendo o nome acessível do mesmo. Todavia, quando não existe um texto associado ao elemento (e.g., botão criado a partir de uma imagem), deverá ser adicionado um nome ao elemento através da utilização dos atributos `aria-label` ou `aria-labelledby`, apresentados na especificação WAI-ARIA [49].

No presente módulo, os elementos identificados que não tinham nome acessível associado diziam respeito, maioritariamente, a botões criados com recurso a elementos gráficos,não tendo texto associado. Nestes casos, o leitor de ecrã apenas irá anunciar o tipo do elemento (e.g., *"button"*). Porém, sem um nome, o utilizador invisual não saberá o propósito do elemento (e.g., "Submeter Pedido").
O problema pode ser facilmente identificado através da utilização de ferramentas automá-

ticas de avaliação de acessibilidade (como a extensão aXe [3]) e que, através da análise do HTML, verificam se os elementos têm um nome associado. No caso de elementos criados a partir da *tag* nativa `<button>` ou com recurso à utilização do atributo `role` (`role="button"`), a ferramenta verifica se existe texto associado ao botão ou se o elemento contém algum atributo `aria-label` ou `aria-labelledby`, também com texto associado [7]. Adicionalmente, o leitor de ecrã pode sempre ser usado como ferramenta de teste por parte dos programadores.

Por via de testes locais, foram identificados nove elementos (botões) que não apresentavam nenhum nome acessível, sendo estes: o botão para eliminar um ficheiro na secção Pedido; os botões para saltar para o próximo dia ou para o dia anterior referentes à data de início (Começa em) e à data de fim (Termina em) ilustrados na Figura 6.4; o botão para expandir/colapsar (destacado a azul na Figura 6.4; os botões para navegar para o mês anterior e mês seguinte presentes no calendário *popup* ilustrado na Figura 6.5; e, por fim, o botão para fechar a janela *popup*.

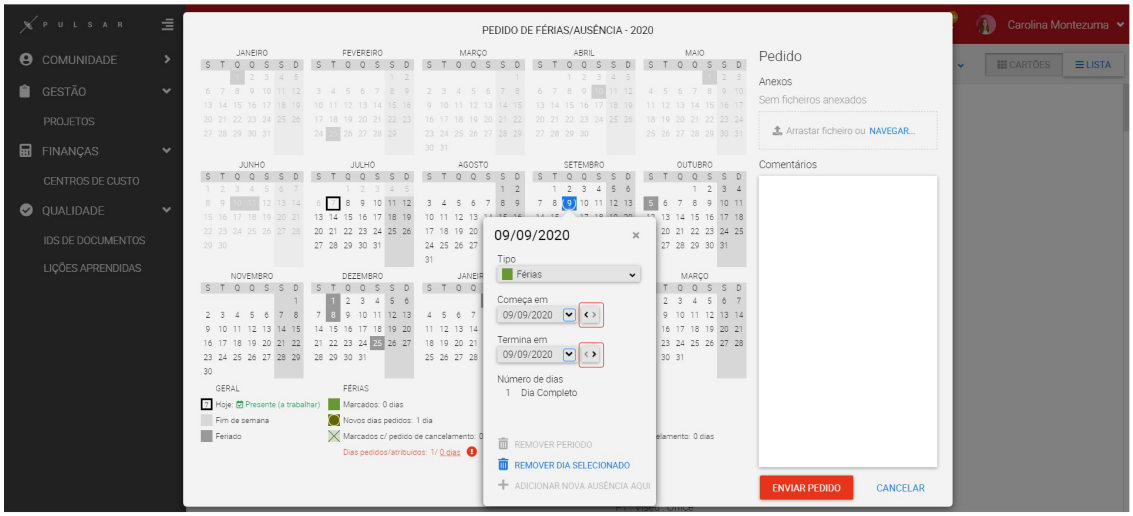


Figura 6.4: Botões janela *popup*

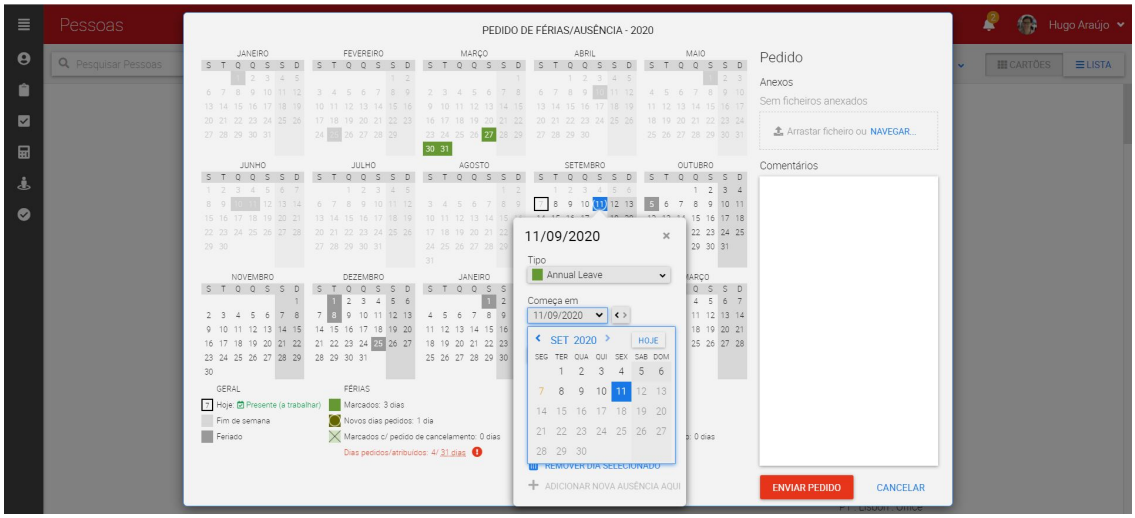


Figura 6.5: Calendário *popup*

Ao atribuir um nome a cada um destes elementos, pretendíamos que estes fossem apenas

disponibilizados aos utilizadores de leitores de ecrã, não sendo visíveis no módulo. A solução passou por adicionar o atributo `aria-label` a cada um dos botões referidos acima, cujo valor associado dizia respeito ao nome acessível que se pretendia adicionar a cada um dos elementos (e.g., *"Expandir calendário"* para o botão cuja funcionalidade expandia o calendário). Todavia, em [39] os autores referem que é preferível a utilização de texto visível para nome de um elemento uma, vez que torna a manutenção mais fácil e previne a existência de *bugs*. Desta forma, não sendo o nome apresentado visualmente, é mais fácil passar despercebido aquando realizada uma atualização quando necessário.

Adicionalmente, houve mais uma situação no contexto do presente módulo na qual se justificou a utilização do atributo `aria-label` - neste caso, relativo aos *headings* definidos dos meses no calendário. Durante um teste realizado com um dos utilizadores finais, foi possível identificar um constrangimento que dizia respeito a três *headings* duplicados, referentes aos meses de janeiro, fevereiro e março do ano 2020 e 2021. Embora os meses não sejam - visualmente - diferenciados (Figura 2.2), um utilizador visual consegue facilmente ter a percepção que se tratam de meses de anos diferentes. Porém, para um utilizador invisual, esta distinção torna-se mais difícil, entre os meses apresentados em duplicado.
A utilização do atributo `aria-label` em elementos que já contêm um texto visível associado, em termos visuais, não têm qualquer tipo de impacto uma vez que o texto (ou nome) do elemento permanece igual. No entanto, para os leitores de ecrã, o valor associado ao atributo `aria-label` - regra geral - tem precedência sobre o conteúdo do elemento, não sendo anunciado o valor vísivel associado ao elemento. Assim, uma vez que não se desejava alterar o valor visualmente, foi associado a cada elemento referente ao título do mês um atributo `aria-label`, cujo valor dizia respeito ao mês e ano em questão. Desta forma, quando os leitores de ecrã anunciam os elementos referentes aos títulos dos meses do calendário, é anunciado não só o mês mas, também, o ano em questão.

### 6.1.3 Comportamento do Foco

A presente subsecção engloba as soluções implementadas para os casos de uso CS-3, CS-4 e CS-5, que descrevem os comportamentos expectáveis do foco no módulo Pedido de Férias/Ausências. Adicionalmente, é apresentada ainda a implementação dos casos de uso CS-6, CS-7 e CS-8, que dizem respeito à janela *popup* apresentada quando um dia do calendário é selecionado.
As implementações de acessibilidade apresentadas nesta subsecção foram, em certa medida, mais complexas e trabalhosas que as restantes. Consequentemente, a descrição do processo torna-se mais extensa. Assim, optámos por dividir esta subsecção em duas partes: implementações na Janela do Módulo e implementações na Janela *Popup*.

#### 6.1.3.1 Janela do Módulo

O módulo apresentado consiste numa janela secundária, sobreposta à página principal, designada por *dialog*. As *dialogs* podem ser *modal* ou *non-modal*. A distinção entre uma *dialog modal* e *non-modal* baseia-se na interação com o conteúdo da página principal: se o conteúdo fica inerte (i.e., o utilizador não pode interagir com o mesmo) trata-se de uma *dialog modal* - como é o caso do presente módulo. Caso contrário, poderá ser disponibilizada uma forma para que o utilizador consiga interagir com o conteúdo da página principal mesmo quando a *dialog* está aberta (*non-modal*) [38].

Como se pode observar na Figura 2.2, o conteúdo apresentado na página principal torna-se inativo quando o módulo é aberto, permanecendo inerte até que o mesmo seja fechado.

Desta forma, o módulo Pedido de Férias/Ausências trata-se de uma *dialog modal*.
O documento WAI Authoring Practices [35] apresenta as considerações que devem ser tidas em conta na criação de um elemento *dialog*, a partir da utilização de elementos não-semânticos do HTML (e.g., `<div>`, `<span>`).

Neste documento, para além da interação do teclado e comportamento expectável do foco neste tipo de elementos, são ainda abordados os atributos *role*, *states* e *properties* da WAI-ARIA, necessários para a criação de elementos *dialog* (`role=''dialog''`, `aria-modal=''true''`, `aria-label` ou `aria-labelledby` com o nome associado à *dialog*). Porém, estes atributos já se encontravam implementados e, por essa razão, foi apenas necessário ter em conta o comportamento expectável do foco considerando a interação do teclado especificada.

O caso de uso CS-3 diz respeito ao foco inicial no módulo Pedido de Férias/Ausências. Isto é, quando o módulo abre, o foco deverá mover-se automaticamente para um dos elementos presentes no mesmo - regra geral, para o primeiro elemento interativo [38].
Até ao momento, uma vez que foco não se movia automaticamente para a *dialog* quando esta abria, os utilizadores invisuais teriam que recorrer a métodos alternativos para realizar o processo manualmente. Foram ainda anotados dois cenários possíveis, resultantes deste comportamento irregular, que poderiam comprometer a boa experiência dos utilizadores invisuais com a aplicação. O primeiro cenário diz respeito ao facto de o aparecimento da *dialog* (i.e., janela do módulo) passar despercebido aos utilizadores invisuais. Tal acontecia porque, tendo em consideração que o foco não se movia automaticamente para a *dialog*, o leitor de ecrã não teria percepção da sua existência e, como tal, o seu aparecimento no ecrã não seria anunciado. Consequentemente, isto poderia levar o utilizador invisual a crer que estaria a pressionar o botão para abrir o módulo sem despoletar nenhuma ação.

O segundo cenário possível nasce, essencialmente, da familiaridade dos utilizadores invisuais com a ausência de foco, partindo do princípio que o módulo foi aberto mesmo não tendo qualquer indicação de que tal tenha acontecido. Como tal, ao pressionar o botão que abre o módulo, pressupõe que - embora não anunciado pelo leitor de ecrã - a *dialog* tenha aparecido no ecrã. Posteriormente, de forma a mover o foco para a *dialog*, foram conhecidas as técnicas adotadas pelos dois colaboradores invisuais de forma a contornar este problema. Um dos colaboradores, até ao momento, pesquisava por referências existentes no módulo, fazendo com que o foco se movesse para esse elemento através deste método alternativo. A Figura 6.6 apresenta um exemplo da pesquisa realizada no módulo pelo utilizador.

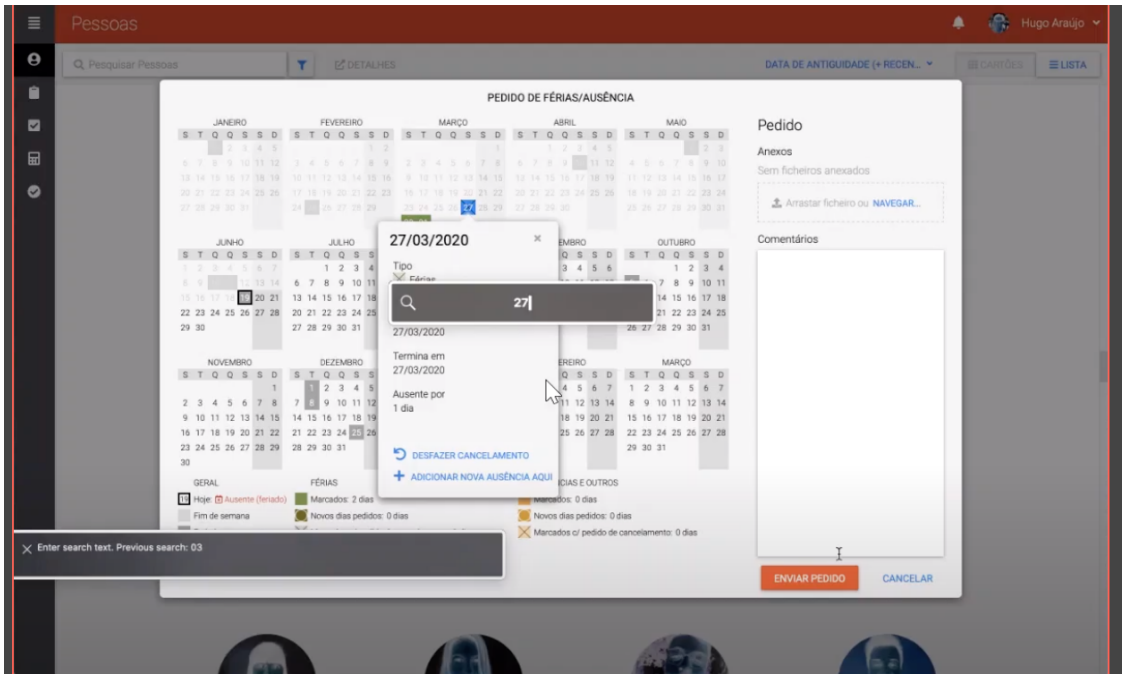


Figura 6.6: Método de Pesquisa

Embora este método alternativo seja utilizado de forma recorrente, não é um método infalível e está longe de ser o ideal. Isto porque, em primeiro lugar, o termo pesquisado pode não corresponder a nenhum dos elementos existentes na *dialog* e, consequentemente, não será possível mover o foco utilizando o método referido. Em segundo lugar, o termo pesquisado pode existir em várias secções da página, levando, possivelmente, os utilizadores a passar primeiro por elementos fora da mesma. Para além disto, os utilizadores podem não ter a percepção de que os elementos encontrados estão fora da *dialog*. Estes fatores contribuem, frequentemente, para uma navegação e experiência de certo modo incertas e irregulares.

Já outro colaborador tinha conhecimento de que, habitualmente, este tipo de janelas costumavam ser apresentadas no topo ou no fim da página e, como tal, navegava para estas secções. Caso a *dialog* estivesse aberta numa destas secções, o leitor de ecrã (neste caso, o NVDA) anunciava ao utilizador que existia uma *dialog* naquela secção. Porém, mesmo nesta situação, o foco não se movia automaticamente e o utilizador tinha que pressionar a tecla Enter para que o foco passasse para a *dialog*. Ainda assim, este processo é mais eficiente apenas por o utilizador estar já habituado a este tipo de comportamento.
Nitidamente, este processo torna-se moroso em comparação com os utilizadores visuais, aos quais basta clicar no botão para abrir o módulo podendo, de imediato, interagir com o mesmo. Por essa razão, é crucial implementar as considerações especificadas em [38]. Ainda, sendo a *dialog* um elemento utilizado em vários módulos no PULSAR (e.g. módulo Reportar Esforço, Mapa de Despesas, Pedido de Viagem), a falta de acessibilidade neste tipo de elementos inviabiliza a experiência de utilizadores invisuais não apenas no módulo Pedido de Férias/Ausências, mas na aplicação em geral.
No presente módulo, o foco inicial - quando a *dialog* abre - foi adicionado ao dia atual no calendário. Considerando que o elemento que iria receber foco se tratava de um elemento `<div>` - que por norma não recebe foco -, foi necessário adicionar o atributo `tabindex` ao mesmo. Recordando, este permite adicionar foco a elementos que, à partida, não o recebem. No caso do elemento do dia atual, apenas seria desejável que este recebesse foco quando era chamado o método `focus()`, mas não através do teclado. Como tal, foi

declarado o valor -1 ao atributo `tabindex` associado ao dia atual.

O segundo caso de uso referido (CS-4), pretendia que fosse implementado o comportamento expectável do foco durante a navegação com o teclado nas *dialogs*. Por navegação do teclado, no presente contexto, entende-se a utilização das teclas Tab e combinação `Shift+Tab`. Como referido em [38], a utilização da tecla `Tab` deverá mover o foco para o elemento interativo seguinte, enquanto que a combinação de teclas `Shift+Tab` deverá mover o foco para o elemento interativo anterior. Embora este comportamento seja expectável em toda a página web, no caso específico das *dialogs* é importante ter em consideração um aspeto adicional: durante a navegação com o teclado, o foco não deve sair das mesmas até que estas sejam fechadas.

Após a realização de testes locais (Capítulo Testes de *Software*, subsecção 7.2.2), foram detetadas duas situações nas quais não se verificavam os comportamentos esperados do foco durante a navegação com o teclado: dois elementos interativos nunca recebiam foco através do teclado, e o foco ficava "preso" num dos elementos. Isto é, o utilizador, ao navegar através da tecla `Tab`, conseguia mover o foco para o elemento, no entanto, caso pressionasse a tecla `Tab` novamente, o foco já não se movia para fora do elemento.

Como já mencionado, foram encontrados dois elementos que nunca recebiam foco, sendo estes o *input* de ficheiros da secção de anexos ('Navegar...', Figura 2.2) e o botão *fechar* presente na janela *popup* (6.2). Porém, os dois problemas identificados eram consequências de duas situações distintas.
No caso do *input* de ficheiros pensou-se, inicialmente, que a solução passaria apenas por adicionar o atributo `tabindex` ao elemento. Neste caso, como era desejável que o elemento recebesse foco através do teclado, deveria ser adicionado o atributo `tabindex` com valor 0 (tabindex = 0). Porém, o mesmo elemento continuou a ser inalcançável (i.e., não recebia foco) através da tecla Tab. Assim, após pesquisa e análise do código base, tornou-se claro que o problema estaria relacionado com a classe associada ao elemento HTML (`<input type=''file''>`), que visava esconder o elemento, fazendo também com que o mesmo não fosse alcançável pelo teclado. Tal acontecia dado que a classe CSS associada ao elemento continha uma propriedade `visibility: hidden` que, para além de esconder o elemento, eliminava a possibilidade do mesmo receber foco.
A solução passou então por alterar a classe associada ao elemento para que este continuasse não-visível mas que pudesse ser alcançável através do teclado, focando o mesmo. No que diz respeito ao botão para fechar a janela *popup* (Figura 6.2), embora para utilizadores visuais (utilizando o rato) funcionasse como seria esperado para um botão, o mesmo não acontecia para os utilizadores invisuais uma vez que não teriam a percepção que o elemento existia na mesma. No presente elemento, foram detetados dois problemas que o tornavam inacessível a utilizadores invisuais: em primeiro lugar, o elemento referente ao botão tinha a si associado o atributo `aria-hidden` com valor a *true* (`aria-hidden = true`) e, em segundo lugar, o (aparente) botão não tinha sido criado utilizando a tag nativa do HTML designada para botões - `<button>` - mas sim através da utilização de elementos não-semânticos do HTML (i.e., elementos que, por definição, não incorporam a interação de teclado esperada, não recebem foco, nem especificam a sua função).
De forma a evitar a implementação manual dos comportamentos expectáveis num botão descritos no documento WAI Authoring Practices [37], foi apenas substituído o elemento `<div>` utilizado para a criação do botão por um elemento criado a partir da tag nativa HTML `<button>`. Assim, permitimos que todos os pontos essenciais mencionados fossem garantidos de forma automática.
Adicionalmente, foi necessário remover o atributo `aria-hidden`. Embora o elemento recebesse foco através do teclado, este atributo fazia com que, o mesmo não fosse identificado

pelos leitores de ecrã. Isto é, caso o botão recebesse foco, o leitor de ecrã não iria anunciar nenhuma informação, não sendo ainda possível interagir com o elemento fazendo uso do teclado.

O segundo problema identificado dizia respeito ao foco ficar "preso" num elemento, sendo este o *dropdown* (`Tipo`) presente na janela *popup* (ilustrado na Figura 6.7).

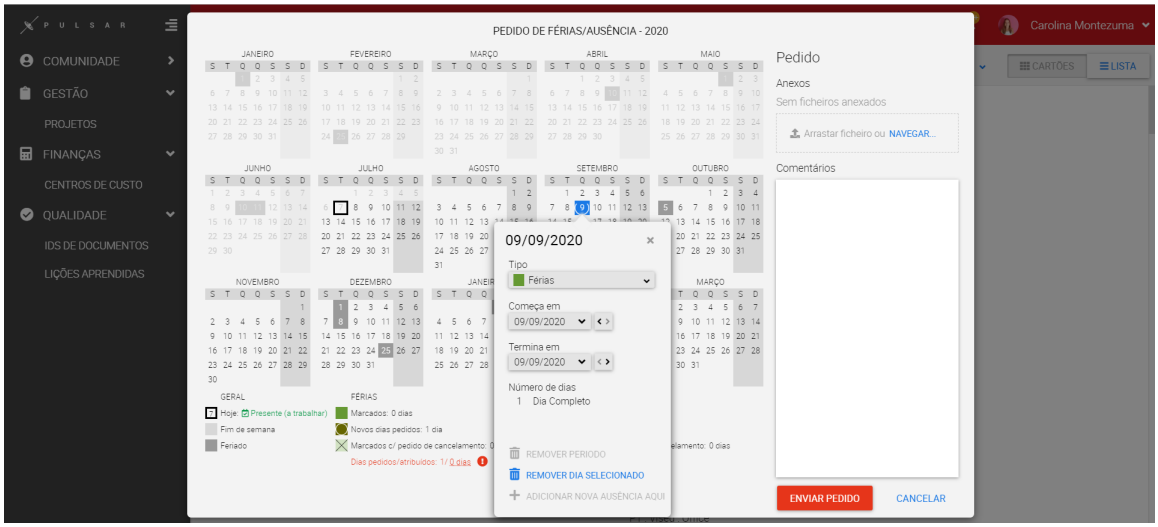


Figura 6.7: Janela *popup* (Pedido de Férias)

O problema estava relacionado com o facto de, no componente *dropdown*, cada vez que era detetado um evento associado à tecla `Tab` ser pressionada, o método `preventDefault()` era chamado - que cancela o evento [23]. Por essa razão, o foco não se movimentava para o elemento interativo seguinte. O problema ficou resolvido com a remoção deste método que, para além de ser a fonte do problema em questão, a sua exclusão não alterou o bom funcionamento da plataforma. Ainda, o problema apresentado não se verificava apenas no presente módulo, mas também no *dropdown* apresentado no módulo Reportar Esforço. Isto porque o *dropdown* é um componente Angular, implementado em várias partes da aplicação. Por essa razão, ao resolver o problema no componente, todos os elementos *dropdown* existentes na aplicação (inclusive no módulo Reportar Esforço) foram automaticamente resolvidos.

Por último, relembramos a noção de que o foco não deverá sair da *dialog* até que esta seja fechada. Caso o foco se encontre no último elemento interativo, ao pressionar a tecla `Tab`, o foco deve mover-se para o primeiro elemento interativo da *dialog*. Ainda, caso o foco se encontre no último elemento interativo, este deverá voltar para o primeiro elemento interativo quando pressionadas as teclas `Shift+Tab`. De forma a implementar estes comportamentos, foram criadas duas funções. A primeira tem como objetivo recolher todos os elementos interativos presentes na *dialog*, sendo que a segunda assegura a lógica da navegação, garantindo que o foco não se move para fora da *dialog*. Na primeira função, foi utilizado o método `querySelectorAll()` que permite retornar todos os elementos presentes no ficheiro HTML que coincidam com os selectors CSS passados como parâmetro [16]. A título de exemplo, caso fosse passado como parâmetro o valor '*button*', a função iria retornar uma lista com todos os botões (criados com a *tag* `<button>`) presentes no ficheiro. A lista de elementos devolvida encontra-se ordenada pela ordem em que os elementos aparecem no ficheiro. Por essa razão, poderíamos assumir que o primeiro elemento presente na lista corresponderia ao primeiro elemento interativo módulo.

Relativamente à segunda função, esta faz uso da lista que é devolvida na função anterior. Caso o utilizador pressione a tecla `Tab`, e caso o elemento ativo (com foco) seja o último elemento da lista (i.e., o último elemento interativo da *dialog*), é adicionado o foco - através do método `focus()` - ao primeiro elemento interativo. De forma idêntica, se forem utilizadas as teclas `Shift+Tab`, e caso o elemento ativo for o primeiro elemento da lista (i.e., primeiro elemento interativo), o foco é adicionado ao último elemento interativo. Esta implementação permite, assim, que a navegação dentro das *dialogs* através de teclado seja cíclica.
Como referido anteriormente, o módulo Pedido de Férias/Ausências não é o único módulo no PULSAR criado a partir de uma *dialog*. Como tal, de forma a facilitar a implementação do comportamento apresentado acima nos outros módulos, criámos um serviço Angular no qual foram adicionadas as duas funções descritas acima. Desta forma, o processo fica facilitado para futuras implementações deste comportamento noutras *dialogs*, uma vez que as funções não terão de ser novamente criadas nos componentes, bastando apenas injetar o serviço nos respetivos componentes e chamar as funções presentes no serviço.

Por fim, abordamos implementação do caso de uso CS-5. Em [35], os autores apontam as considerações a ter durante a implementação de uma *dialog* (*modal*). Porém, não é mencionado explicitamente que, aquando da implementação de uma *dialog*, também deverá ser tido em conta o comportamento do foco quando o utilizador interage com um dos elementos presentes na mesma. Assim, é igualmente importante garantir que, nesta situação, o foco não se move para fora da *dialog*.
Durante os testes locais, foi anotada uma situação irregular: ao interagir com um dos elementos presentes na *dialog* - botão Submeter Pedido -, se ocorresse um erro ao submeter o pedido, o foco movia-se para outro elemento (fora da mesma). A solução para manter o foco na *dialog* teve em consideração o comportamento do foco que é documentado em [37] quando um botão é ativado. Entre as considerações apresentadas no que diz respeito ao foco, os autores referem que quando um botão é ativado e não existe uma alteração no contexto da página, o foco deverá permanecer no botão que foi ativado. Assim, o foco foi adicionado novamente ao botão (utilizando o método `focus()`).

#### 6.1.3.2 Janela *Popup*

Na Figura 6.2, podemos observar a janela *popup* que aparece cada vez que um dia no calendário é selecionado. Seria esperado que, neste tipo de elementos, também fossem implementados os comportamentos descritos acima (i.e., foco inicial, navegação no *popup* e, por último, foco quando a *dialog* fecha) [41].

Primeiramente, quando um dia no calendário é selecionado e a janela *popup* abre, o foco foi adicionado ao primeiro elemento interativo presente na mesma (botão fechar).

De seguida, tirámos partido do serviço Angular criado anteriormente, implementando o mesmo no componente da janela *popup*. Relembrando, este serviço contém funções que permitem guardar todos os elementos interativos presentes no módulo e, a partir daí, implementar a lógica de navegação esperada (utilizando as teclas Tab e Shift+Tab). Assim, garantimos foco automático num elemento pertencente à janela *popup* quando esta abre, garantindo ainda a lógica de navegação esperada num elemento *dialog*. Desta forma, o foco não se move para fora da *dialog* até que esta seja fechada.

Por último, quando a janela *popup* é fechada, o foco deverá regressar ao elemento responsável pela sua abertura - neste caso, ao dia no calendário anteriormente selecionado.

No entanto, para que tal fosse possível, foi necessário corrigir um comportamento: a tecla Escape não fechava apenas a janela *popup*, mas também a janela do próprio módulo. Para impedir que tal acontecesse, garantimos que, quando um evento relativo à tecla Escape é detetado, o mesmo não se propague para a janela do módulo. Para isso, é necessário chamar dois métodos: `preventDefault()`, e ainda o método `stopPropagation()`, uma vez que não era desejado que o evento se propagasse para o elemento pai [25].

### 6.1.4 Transposição de Cores

Observando a Figura 2.2, na qual é apresentado o módulo Pedido de Férias/Ausências com 3 dias de férias marcados, um utilizador visual consegue indicar rapidamente que que se tratam dos dias de 27 a 31 de março de 2020. Não será erróneo afirmar que, para um utilizador visual, esta consulta levaria apenas breves segundos. Em contraste, a consulta de dias com férias ou ausências para utilizadores invisuais é um processo tão moroso que se torna praticamente inexequível. Como será apresentado no capítulo referente aos Testes de *Software* (subsecção 7.2.4), quando foi pedido a dois colaboradores invisuais para consultarem os dias de férias marcados no calendário, ambos desistiram prontamente da realização da tarefa precisamente por esta se encontrar apenas codificada por cores. Isto porque, de forma a consultar essa informação, teriam que percorrer todos os dias, selecionar cada um dos dias e verificar a informação que é apresentada na janela *popup*, apresentada a título de exemplo na Figura 6.8.

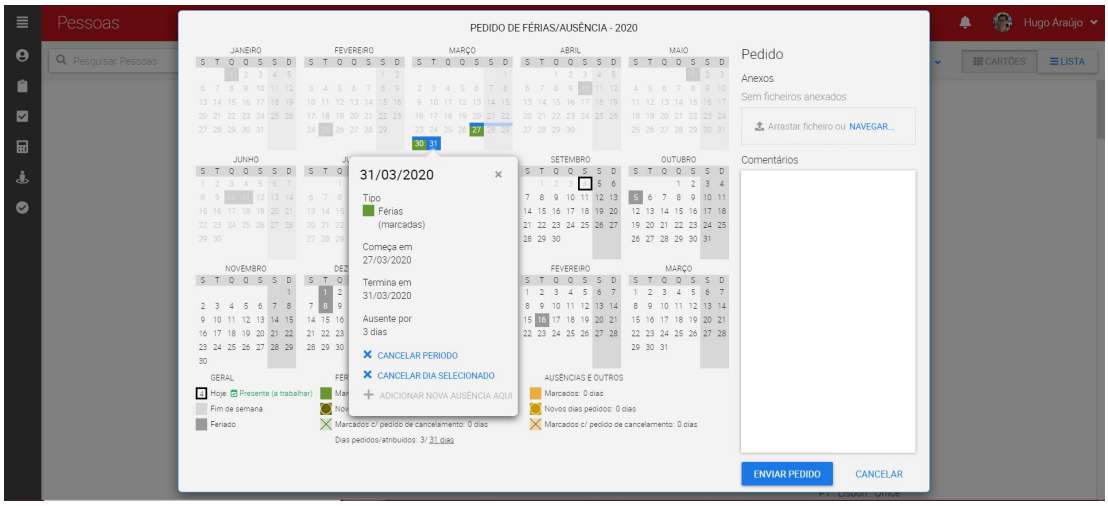


Figura 6.8: Janela *popup* com dia de férias marcado

Não obstante, tendo em consideração que ainda não tinham sido implementadas as alterações relativas ao comportamento do foco, o mesmo não era movido automaticamente para a janela *popup* quando esta abria. Por essa razão, era necessário recorrer ao processo de mover (manualmente) o foco para o *popup* (descrito na subsecção anterior). Este processo iria repetir-se até que os utilizadores invisuais terminassem a consulta de toda a informação que necessitavam. Adicionalmente, os utilizadores teriam sempre que memorizar o último dia que tinham consultado, caso contrário, poderiam estar a repetir a consulta de dias.

Consequentemente, uma vez que se tornava um processo muito repetitivo e exaustivo, os colaboradores invisuais da empresa recorriam a métodos alternativos para realizar a consulta de informação módulo. Era frequente pedirem ajuda a colegas, consultarem os *e-mails* de confirmação que recebiam de cada vez que realizavam um pedido ou, ainda, anotarem a

informação numa agenda ou num ficheiro de texto. Por estas razões, era fulcral encontrar uma solução que lhes permitisse percecionar, de um modo geral, a informação relativa a férias e ausências apresentada no calendário, equivalente à experiência dos utilizadores visuais.

A solução para este problema não foi tão linear comparativamente às soluções implementadas para os outros problemas já referidos, como falta de *headings* em páginas ou elementos sem nomes associados (subsecção 6.1.1 e 6.1.2). Por serem soluções mais diretas, são também soluções facilmente testáveis pelos programadores, não sendo necessária a intervenção recorrente de utilizadores invisuais.

A primeira solução idealizada consistia em adicionar a informação - de férias ou ausências - apresentada por cores no calendário à legenda que é apresentada abaixo do mesmo (Figura 2.2). Para tal, a cada um dos elementos presentes nas legendas `Férias` e `Ausências e Outros` (Número de dias marcados, Número de dias pedido, Número de dias marcados com pedido de cancelamento), seria adicionada mais informação - não visível - em cada um dos elementos. Assim, um utilizador invisual ao navegar pela legenda iria ter imediatamente acesso a informação apresentada por cores no calendário. Todavia, esta solução não foi implementada, uma vez que se considerou ser mais intuitivo ter esta informação organizada por meses em vez de dividir a informação pelas diferentes secções da legenda.

Considerando que os títulos dos meses foram definidos como *headings* (subsecção 6.1.1), a solução passou por permitir que os *headings* referentes a cada mês tivessem acesso a mais informação. Para tal, são criados - dinamicamente - elementos não visíveis na página (`<div>`) para cada um dos meses do calendário, cada um com um *id* associado. Estes elementos contêm, assim, a informação pretendida. Para que os *headings* referidos consigam aceder a esta informação, foi adicionado o atributo `aria-describedby`, já que este atributo permite aceder à mesma recorrendo ao *id* de cada elemento.

A utilização do atributo `aria-describedby` para adicionar informação aos *headings* permite que, ao navegar para um dos *headings* referentes aos meses do calendário, o leitor de ecrã anuncie aos utilizadores invisuais toda a informação que é apresentada por cores no mês em questão. Assim, a título de exemplo, ao navegar para o *heading* relativo ao mês de março de 2020 o leitor de ecrã irá anunciar a seguinte informação: "Férias Dias Marcados: 27, 30, 31". Adicionalmente, no caso de não existir nenhuma informação no mês, o leitor de ecrã informa o utilizador que não existe marcação no mesmo.

Complementarmente, a informação relativa a férias ou ausências apresentada em cada um dos dias - com recurso à utilização de cores - foi transposta para texto (apenas disponibilizado para os leitores de ecrã). Desta forma, ao navegar pelos dias utilizando as setas do teclado, utilizador invisual receberá *feedback* imediato da informação de cada um dos dias. Uma vez que a informação do dia do mês (visível) não era suficiente para identificar o dia aos utilizadores invisuais, a cada um dos elementos tinha sido adicionado - antes do presente estágio - o atributo `aria-label` cujo valor dizia respeito à data completa (e.g., Domingo, 12 de Julho 2020). Como já tivemos oportunidade de discutir, para os leitores de ecrã, o valor associado a este atributo tem precedência sobre o texto associado aos elementos, pelo que será unicamente anunciada a data completa. Todavia, tal não se verificou utilizando o leitor de ecrã NVDA uma vez que, no presente contexto, o *feedback* dado dizia respeito ao texto visível associado ao elemento (i.e., o dia apenas). Como tal, antes de transpor para texto a informação codificada por cores (garantido que os leitores de ecrã pudessem aceder à mesma), foi necessário corrigir este comportamento irregular identificado com o leitor de ecrã NVDA.

O problema passava pelo facto de cada dia do calendário ter sido criado com recurso a

um elemento `<div>`, sendo que se verificou que, no caso destes elementos, o leitor de ecrã NVDA ignora o atributo `aria-label` anunciando apenas o texto visível. De forma a resolver esta questão, foi adicionado o atributo `role=''button''` aos dias do calendário, uma vez que cada um dos dias tinha o comportamento de um botão (as teclas `Space` e `Enter` ativavam o dia do calendário e despoletavam o aparecimento do *popup*). Desta forma, o leitor de ecrã NVDA passa a dar precedência ao valor associado ao atributo `aria-label`. Posteriormente, adicionámos ainda ao valor associado ao atributo `aria-label` a informação representada por cores (caso o dia tivesse algum evento). Com isto, a título de exemplo, ao navegar pelo calendário, o utilizador, caso se encontre num dia em que tem férias marcadas, irá receber *feedback* do leitor de ecrã: *"Férias Dia Marcado Domingo, 12 de Julho 2020"*.
Tendo em consideração que os calendários apresentados nos módulos Pedido de Férias/Ausências e Reportar Esforço correspondiam ao mesmo componente Angular, quando foi realizada a alteração referida anteriormente (i.e., adicionar o atributo `role="button"`), o problema ficou também resolvido no módulo Reportar Esforço.

As soluções apresentadas nesta subsecção permitiriam tornar o processo de consulta de informação no calendário exequível para utilizadores invisuais, ação que até ao momento não era realizada pelos mesmos. Não obstante, foi também notório que os utilizadores conseguiam realizar a tarefa com agilidade e, por essa razão, consideramos que foram duas soluções pertinentes para resolver o problema. Como tal, como iremos referir mais à frente na subsecção referente às cores no módulo Reportar Esforço (subsecção 6.2.3), as soluções adoptadas foram muito semelhantes. Tendo ainda em vista agilizar o processo de consulta de informação no calendário, foi implementado um atalho de teclado que será apresentado na subsecção seguinte.

### 6.1.5 Atalhos de Teclado

Apresentamos agora as implementações relativas aos atalhos de teclado, no contexto do módulo Pedido de Férias/Ausências.
Os casos de uso CS-16 e CS-17 descrevem funcionalidades que se pretendiam implementar a partir da utilização de atalhos de teclado, sendo estas: navegar apenas entre dias com eventos (i.e., férias ou ausências) e navegar para o dia atual no calendário.

O primeiro atalho é um complemento às alterações apresentadas na subsecção anterior, como forma de agilizar o processo de consulta de eventos no calendário para os utilizadores invisuais.
A funcionalidade do atalho baseia-se em mover o foco para o próximo dia do calendário que tenha a si associado algum tipo de evento, referente a férias ou ausências. Assim, a utilização do mesmo permite agilizar a navegação entre os dias com eventos e, consequentemente, facilitar a percepção aos utilizadores invisuais de quais os dias no calendário que têm eventos associados. Tendo em conta que os dias do calendário já tinham associado a informação apresentada por cores ao valor do atributo `aria-label`, os utilizadores invisuais conseguiam não só ter a percepção de quais os dias com eventos, como também quais os eventos associados a cada um dos dias.
Tendo em vista a implementação do presente atalho, foi necessário recolher todos os dias do calendário com eventos. Para tal, foi utilizado novamente o método `querySelectorAll()` (descrito na subsecção 6.1.3). Os valores passados como parâmetro neste método diziam respeito aos CSS *selectors* referentes às classes CSS associadas aos dias com eventos. Assim, sabendo que o método retornava uma lista ordenada dos elementos, é verificado qual o próximo elemento a partir daquele que está ativo no momento. Posteriormente, é chamado o método `focus()` para adicionar o foco no respetivo elemento.

Relativamente ao segundo atalho referido (navegar para o dia atual no calendário), a implementação do mesmo visava não só disponibilizar uma forma rápida de mover o foco para o calendário, como também regressar para um dia que poderia servir como referência. De forma a identificar o elemento correspondente ao dia atual no calendário, é chamado o método `querySelector()` que retorna o primeiro elemento no documento que corresponde ao CSS *selector* passado como parâmetro (*'.today'*) [15].

No que diz respeito à implementação de atalhos de teclado, em [40] são apresentadas algumas considerações que devem ser tidas em conta durante o processo.
Um dado importante passa pela noção de que os atalhos de teclado definidos devem servir apenas como complemento. Tal significa que qualquer funcionalidade que responda a um certo atalho deverá, de igual modo, ser exequível através do acesso convencional por teclado. Para além disso, os autores salientam que uma dada funcionalidade do atalho deve estar relacionada com navegação (mover o foco para um elemento), ativação (ativar um elemento que não está a receber foco e que pode não estar visível na página), ou a junção dos dois. Outra consideração aponta para o compromisso entre uma boa experiência e uma carga cognitiva moderada: a ausência (ou falta) de atalhos de teclado pode diminuir a eficiência da experiência, porém, a existência de atalhos em excesso poderá aumentar a carga cognitiva exigida aos utilizadores.
Durante o processo de implementação dos atalhos de teclado, foram encontradas duas dificuldades associadas à definição dos mesmos. Primeiramente, foram encontrados alguns conflitos na implementação dos atalhos com atalhos já definidos (a nível de sistema operativo e *browser*). Uma outra dificuldade encontrada, embora não estivesse relacionada com a implementação dos atalhos propriamente dita, passava por estabelecer uma forma intuitiva de dar a conhecer, apenas aos utilizadores invisuais, os atalhos disponíveis no módulo. Isto porque os atalhos por nós implementados não são vantajosos para utilizadores visuais - uma vez que o foco não se encontra visível -, e regra geral estes utilizadores não fazem uso de leitores de ecrã. Assim, a solução para a apresentação dos atalhos disponíveis no módulo é detalhada mais à frente na subsecção 6.1.6.

Para as duas funcionalidades especificadas acima, numa fase inicial, foram implementados dois atalhos, sendo estes: `Alt+N` para saltar entre dias com eventos e `Alt+T` para saltar para o dia atual no calendário. As letras associadas pretendiam seguir a lógica da funcionalidade de cada um deles de forma a serem facilmente memorizados: N de *next*, uma vez que o atalho permitiria navegar para o seguinte dia com evento no calendário; e a tecla T, utilizada para fazer referência à palavra *today*, já que se tratava de um atalho para navegar para o dia atual.
Os atalhos, quando implementados, apenas foram testados localmente (sistema operativo Windows no *browser* Google Chrome, fazendo uso do leitor de ecrã NVDA), não tendo sido detectadas falhas. Todavia, nos testes realizados com os utilizadores finais, foram detetados conflitos em duas situações distintas. Na primeira, na qual também era usado o sistema operativo Windows - com o leitor de ecrã NVDA, com o *browser* Mozilla Firefox -, o atalho definido para navegar para o dia atual entrava em conflito com um atalho já existente do próprio *browser*. Na segunda situação, ambos os atalhos de teclado definidos revelaram conflitos, não funcionando em MacOS, com o leitor com o leitor de ecrã VoiceOver e *browser* Google Chrome.

Por estas razões, uma vez que seria difícil encontrar um único atalho transversal aos diferentes sistemas operativos, leitores de ecrã e *browsers*, foram implementados dois atalhos para cada uma das funcionalidades apresentadas. Assim, para navegar para o dia atual do calendário, definiram-se dois atalhos diferentes com a mesma funcionalidade - `Ctrl+Shift+t ou Alt+t`. De igual modo, foram definidos dois atalhos de teclado para a funcionalidade de navegar entre dias com eventos, sendo estes: `Ctrl+Shift+n ou Alt+n`.

A atribuição de dois atalhos de teclado para a mesma funcionalidade, embora possa evitar conflitos resultantes de atalhos de teclado já definidos a nível de sistema operativo, *browser* ou leitor de ecrã, acarreta desvantagens como o aumento da carga cognitiva. Tal acontece porque serão apresentados o dobro dos atalhos de teclado ao utilizador. Não obstante, numa primeira experiência, o utilizador terá sempre que testar qual dos atalhos funciona no seu caso, dependendo das tecnologias usadas. No entanto, considerando os problemas encontrados a nível de conflitos entre tecnologias, a garantia de um bom funcionamento dos atalhos de teclado - independente das tecnologias usadas - foi tida como um bom compromisso.

Os atalhos de teclado podem ser vantajosos, no sentido em que pretendem facilitar determinadas ações dos utilizadores numa página, como é o caso da consulta de dias com eventos. Neste exemplo, através do atalho definido, o utilizador facilmente terá acesso à informação. No entanto, importa notar novamente que os mesmos só deverão ser implementados como complementos. Na subsecção seguinte, será detalhada a solução adoptada para dar a conhecer aos utilizadores invisuais os atalhos de teclado implementados no presente módulo, entre outras notificações.

### 6.1.6 Notificações do Utilizador

Na presente subsecção, apresentamos as situações nas quais se considerou vantajoso a utilização de notificações para os utilizadores invisuais, bem como a descrição do processo de implementação das mesmas.
A implementação de notificações permite que os utilizadores invisuais tenham perceção da conclusão de tarefas, erros que possam ocorrer durante a realização de uma tarefa ou eventuais alterações dinâmicas nas páginas.
Em [22], são apresentadas algumas das vantagens do uso de notificações. Em primeiro lugar, o conteúdo presente na página torna-se mais usável para pessoas que não estão familiarizadas com a aplicação; em segundo, o conteúdo torna-se mais usável para pessoas que não se sentem seguras ao utilizar computadores ou a fazerem uso da web; e por último, o conteúdo torna-se menos confuso e não suscita dúvidas aos utilizadores.

No âmbito do módulo Pedido de Férias/Ausências, foram anotadas diferentes situações nas quais seria necessário implementar notificações por forma a garantir que os utilizadores invisuais tivessem uma experiência similar à dos utilizadores visuais. Estas implementações partiram da premissa de que os utilizadores invisuais deveriam também ter acesso às mensagens que são apresentadas no ecrã. Adicionalmente, era ainda importante garantir que os mesmos tinham acesso ao *feedback* apresentado por cores, conferindo-lhes mais segurança relativamente às ações realizadas no módulo.
De seguida, serão apresentadas as situações nas quais considerámos necessária a implementação de notificações, apresentando por fim a solução - única - adoptada para dar resposta às diferentes situações.

Começando pela situação mais direta, seria expectável que as mensagens que são apresentadas nos leitores de ecrã, e que visam transmitir aos utilizadores *feedback* referente a uma ação, fossem também disponibilizadas aos utilizadores invisuais. A título de exemplo, as Figuras 6.9 e 6.10 apresentam as mensagens que são disponibilizadas no ecrã quando um pedido é submetido com sucesso ou quando ocorre um erro a submeter o pedido, respetivamente. As mensagens são apresentadas num componente *toast* (topo da janela), utilizado para disponibilizar alertas no ecrã apenas por alguns segundos [5].

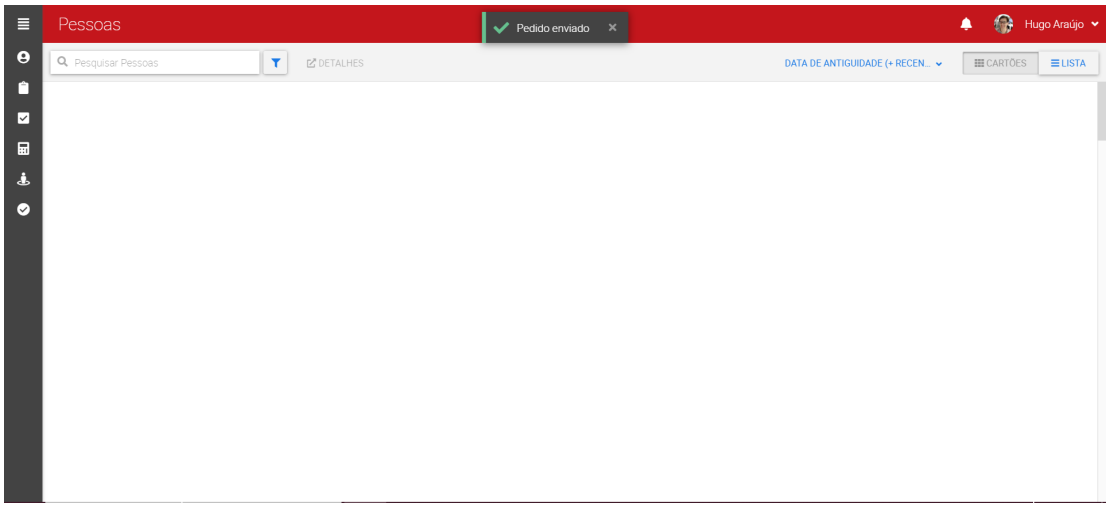


Figura 6.9: Pedido enviado

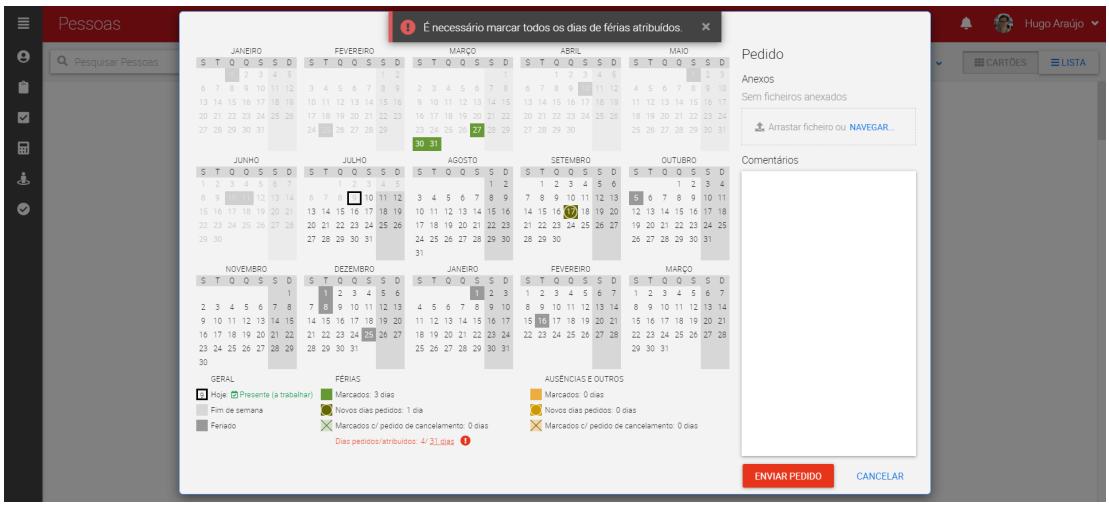


Figura 6.10: Erro ao submeter pedido

Embora as mensagens estejam disponíveis e sejam apresentadas no ecrã, as mesmas não são anunciadas pelos leitores de ecrã. Isto acontece dado que, quando existe uma alteração num elemento da página, regra geral, o foco não se irá mover para o elemento que sofreu uma atualização - exceto se este for programado previamente para que tal aconteça. Neste caso, o foco não se move para o *toast* que contém as mensagens apresentadas. Como tal, o leitor de ecrã não irá ter perceção de que um novo elemento foi disponibilizado na página e, por consequência, as notificações existentes no PULSAR não eram acessíveis a utilizadores de leitores de ecrã.

Num outro cenário, quando o utilizador preenche um novo dia parcial de ausências, é disponibilizada uma caixa de texto para o utilizador preencher as horas referentes à ausência (`'horas/dia'`, apresentada na Figura 6.11).

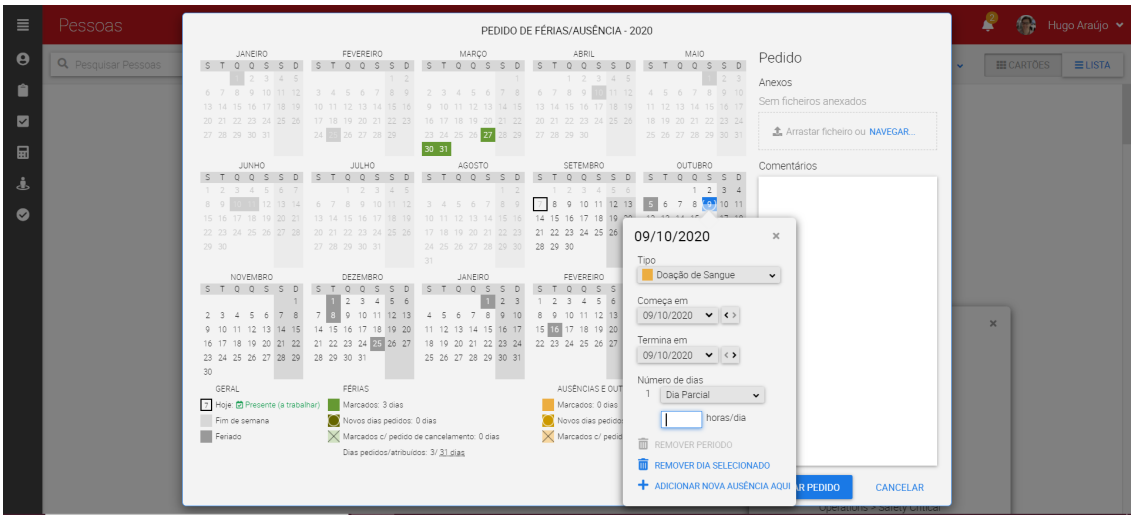


Figura 6.11: Campo dia parcial (horas/dia)

Embora o campo não seja apresentado visualmente como sendo obrigatório (e.g., utilizando um caracter * junto à caixa), o utilizador terá necessariamente de preencher o mesmo, sendo que o valor definido também não deverá ultrapassar as horas laborais de um dia. Quando uma destas duas situações se verifica (campo por preencher ou horas inválidas), um utilizador visual consegue facilmente perceber que existe um erro no campo, uma vez que este é apresentado a vermelho, como podemos observar na Figura 6.12. Complementarmente, é ainda apresentada uma *tooltip* que informa o utilizador relativamente ao problema em questão caso este passe com o rato sobre a mesma (*mouseOver*). Também esta situação passava despercebida aos utilizadores invisuais, que não iriam ter acesso à informação apresentada na *tooltip*.

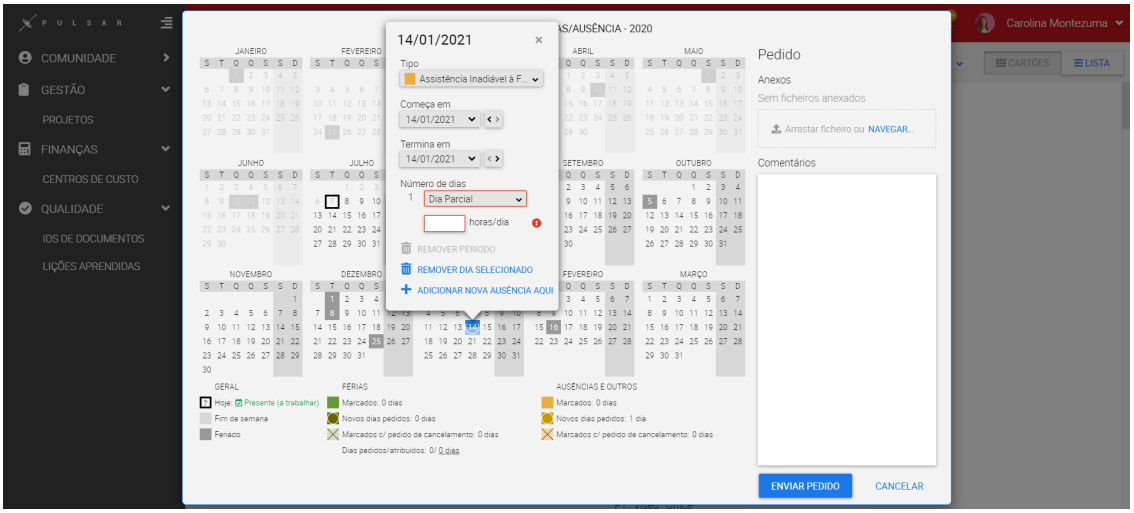


Figura 6.12: Campos por preencher

Outra situação identificada dizia respeito ao elemento *dropdown* apresentado na janela *popup* (Figura 6.7). Quando o utilizador faz uso do teclado - setas para cima e para baixo - para navegar entre as diferentes opções apresentadas nos elementos do *dropdown*, a opção selecionada é alterada, porém, tal não é anunciado pelos leitores de ecrã. Assim, um dos colaboradores mencionou que teria de selecionar uma das opções desprovido de certeza, mover o foco para fora do elemento *dropdown* e movê-lo novamente para a área do mesmo.

Só desta forma conseguiria saber qual a opção em questão.

Durante os testes paralelos à fase de implementação, foi identificado outro problema, neste caso relativo à ação de pedir de um novo dia de férias (ou ausências), ou ao cancelar um dia marcado de férias ou ausências. Embora este tenha sido um problema reportado por todos os colaboradores invisuais, apresentamos de seguida um exemplo concreto de um dos utilizadores.
Este colaborador pretendia cancelar um dia de férias marcado. Para tal, ao selecionar o dia em questão, pressionou o botão '`Cancelar Dia Selecionado`' ilustrado na Figura 6.8, que levou a uma atualização da janela *popup*.

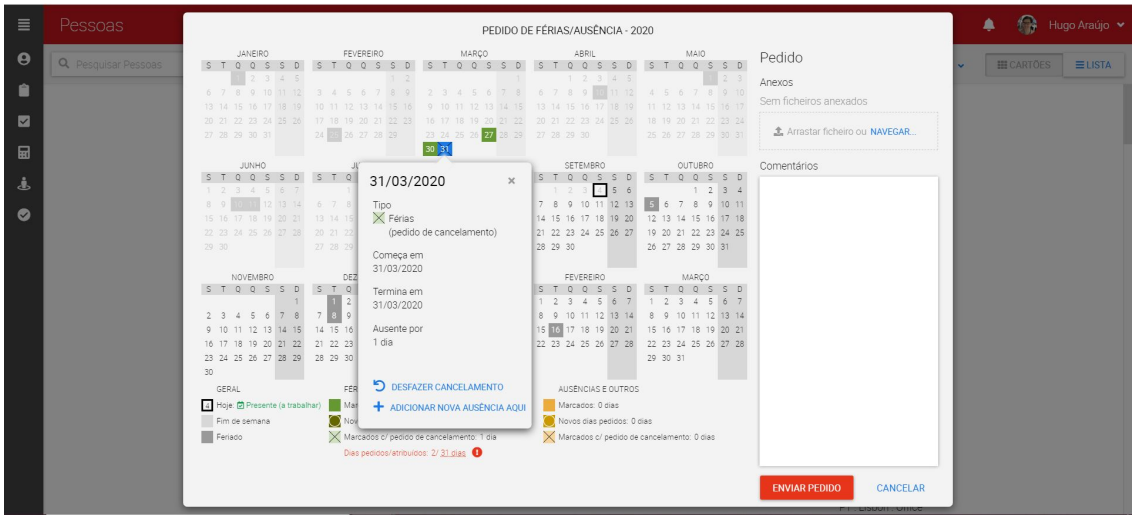


Figura 6.13: Janela *popup* após selecionar 'Cancelar Dia Selecionado'

Assim, como se pode observar na Figura 6.13, a janela *popup* apresenta agora a informação relativa à ação realizada, assim como as opções '`Desfazer Cancelamento`', '`Adicionar Nova Ausência Aqui`' e o botão para fechar o *popup*. Neste ponto, o colaborador reportou duas dificuldades. Em primeiro lugar, não apresentava certezas sobre qual seria o próximo passo a tomar, uma vez que não encontrava nenhuma opção que lhe garantisse que as alterações realizadas iriam ser guardadas. Por exclusão de hipóteses, acabou por fechar a janela *popup*, mas referiu que não sentia segurança, uma vez que não tinha certezas de que estivesse a guardar as alterações realizadas, não sendo uma ação intuitiva. Isto conduz-nos à segunda dificuldade apresentada, que diz respeito ao facto de os utilizadores invisuais não terem uma confirmação que a ação foi efetivamente realizada ao fechar o *popup*.
Contrariamente, os utilizadores visuais não só tinham *feedback* de cores (quando a janela *popup* fechava, o dia era apresentado com a respetiva cor, como apresentado na Figura 6.14), como também a respetiva legenda com o número total de dias era alterada.

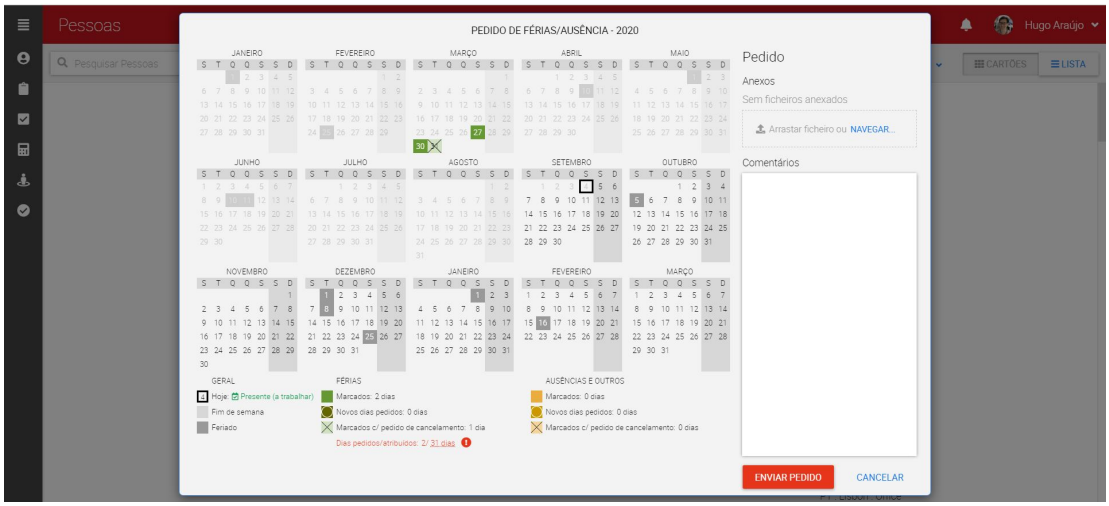


Figura 6.14: Dia com pedido de cancelamento

Porém, mesmo tendo acesso à legenda, os utilizadores invisuais não iriam ter percepção que a mesma tinha sido alterada, a não ser que navegassem atè à respetiva legenda e consultassem a informação. Adicionalmente, precisavam também de saber de antemão os valores antigos da legenda, para garantirem que estes tinham sido atualizados.
Ainda no exemplo prático do utilizador invisual, este apresentou dúvidas após fechar a janlea *popup*, não tendo a certeza de que esta seria a ação final ou se teria que efetuar mais algum passo cancelar um dia de férias marcado. Para os utilizadores visuais, é fácil ter a perceção que a ação não é final uma vez que reconhecem o botão '`Submeter Pedido`' (Figura 2.2). Porém, para os utilizadores invisuais que não têm esta percepção geral da página, podendo não saber da existência do botão, pode ser susceptível a este tipo de dúvidas.
As mesmas dificuldades verificavam-se quando o utilizador pedia um novo dia de férias (ou ausências). Porém, importa notar que as dificuldades apresentadas surgem maioritariamente de um problema de usabilidade da aplicação e não de acessibilidade. Não obstante, e independentemente deste facto, os utilizadores invisuais deveriam ter *feedback* quando fecham a janela *popup* da mesma forma que acontece para os utilizadores visuais.

Por último, como mencionado na secção anterior, foi necessário adoptar um método para apresentar os atalhos de teclado disponíveis apenas aos utilizadores invisuais. Isto porque, os atalhos de teclado definidos não foram considerados úteis para utilizadores visuais - fazendo uso comum da aplicação -, considerando que o foco não é visível no calendário.
Desta forma, a solução adoptada foi a definição de uma notificação para ser anunciada pelos leitores de ecrã. Embora esta tenha sido a solução seguida para dar resposta à apresentação dos atalhos disponíveis, a mesma não é infalível, uma vez que os utilizadores invisuais - habitualmente - têm a velocidade de leitura dos leitores de ecrã programada como alta, o que significa que muitas vezes estas mensagens podem passar despercebidas. Um dos colaboradores afirmou ainda que a desvantagem desta solução dizia respeito à mensagem só ser lida uma vez e, como tal, não percebendo o seu conteúdo desde logo, não tinha forma de a repetir. Por fim, percebemos também que os colaboradores invisuais interrompem a leitura do leitor de ecrã recorrentemente por preverem o que será anunciado. Considerando que isto podia acontecer de imediato ao abrir o módulo, tal levaria a que a notificação nunca fosse anunciada.

Como primeira abordagem para colmatar essa dificuldade, foi adicionado mais um atalho de teclado que, quando utilizado, fazia com que o leitor de ecrã repetisse novamente os

atalhos de teclado disponíveis no módulo. Assim, ao abrir o módulo, eram anunciados os atalhos de teclado disponíveis já referidos e, ainda, o novo atalho de teclado definido que permitia ouvir os atalhos novamente. Porém, esta solução também não se verificou eficaz, não só por aumentar a carga cognitiva, como a mensagem poderia continuar a passar despercebida - como descrevemos no parágrafo anterior. Por último, foi ainda pensado adicionar um botão não-visível à página que, ao ser pressionado, anunciaria os atalhos de teclado. Porém, o *feedback* dos utilizadores invisuais refletia que essa solução não era intuitiva, até porque poderiam não saber da existência do botão.
Por fim, acabámos por definir apenas a notificação inicial relativa aos dois atalhos de teclado definidos no módulo, sendo que o elemento que continha a informação - tal como os restantes - se encontrava no topo do módulo onde, regra geral, são definidas as mensagens de acessibilidade. Adicionalmente, os utilizadores invisuais mostraram-se acostumados a navegar até ao topo das páginas para consultar mensagens de acessibilidade, quando têm percepção da sua existência.

A solução para todas as situações expostas teve como base a utilização do atributo `aria-live`, já especificado no documento (Capítulo 3, Secção 3.5). Relembrando, o atributo `aria-live` é utilizado para alertar os leitores de ecrã relativamente à atualização de um elemento da página. Para que tal aconteça, o elemento que poderá sofrer atualizações deverá ter a si associado o atributo `aria-live`. Desta forma, o leitor de ecrã irá estar alerta e, caso o elemento sofra uma atualização, o leitor de ecrã anuncia a mesma (i.e., o novo conteúdo do elemento que sofreu a alteração).
Para isso, para cada uma das situações apresentadas acima, foi criado um elemento `<div>` (não-visível, e inicialmente vazio) que tinha como único objetivo incorporar as mensagens que seriam eventualmente anunciadas pelos leitores de ecrã. Estes elementos foram posicionados no topo do módulo, considerando que os colaboradores invisuais se mostraram habituados a esta prática (caso pretendessem consultar as notificações novamente).
Os elementos `<div>` têm a si associado o atributo `aria-live` com valor *polite*, ou seja, as notificações terão prioridade baixa relativamente à tarefa corrente do leitor de ecrã. Isto significa que a alteração apenas será anunciada quando o leitor de ecrã terminar a tarefa que decorre no momento. Os elementos que incorporam as mensagens serão atualizados de cada vez que uma das situações apresentadas acima se verifique (e.g., o pedido foi submetido com sucesso). Para que o leitor de ecrã anuncie as atualizações, o elemento tem que ser atualizado sempre que se pretender disparar a notificação, mesmo que seja atualizado exatamente com o mesmo conteúdo.
Relativamente às situações em que se pretendia implementar as mensagens apresentadas no ecrã (Figuras 6.9 e 6.10), o elemento definido para incorporar estas mensagens era atualizado quando se verificava uma das situações apresentadas nestas figuras (i.e., o pedido era enviado ou, occoria um erro a submeter o pedido), sendo que o mesmo iria conter a mensagem igual à que era apresentada no ecrã.

Para a situação referente ao preenchimento do campo 'horas/dia' (Figura 6.11), caso o utilizador não tenha preenchido o campo, será adicionada uma mensagem ao respetivo elemento que permite notificar o utilizador de que um campo obrigatório não foi preenchido, especificando ainda o campo em falta: 'Informação necessária: horas/dia'. Ainda nesta situação, e considerando que o campo não é visualmente identificado como sendo obrigatório, por forma a evitar erros dos utilizadores invisuais foi adicionado o atributo `aria-required`, disponível na especificação técnica WAI-ARIA, tendo o seu valor igual a *true*. Assim, o elemento passa a ser identificado pelos leitores de ecrã como sendo obrigatório.

No que diz respeito à notificação apresentada quando o utilizador fecha a janela *popup*, após cancelar ou pedir um novo dia (de férias ou ausências), esta poderia ter passado apenas por adicionar o atributo `aria-live` aos elementos que constituem a legenda. Desta

forma, quando a legenda era atualizada, o leitor de ecrã iria anunciar a nova informação da legenda. Porém, uma vez que os utilizadores não sentiam segurança a realizar a ação, preferiram ter uma notificação que lhes pudesse dar toda a informação referente à ação realizada e, inclusive, os informasse que para finalizarem o pedido (ou o cancelamento) teriam que pressionar o botão '`Submeter Pedido`'.
No que diz respeito à situação do elemento *dropdown*, o atributo `aria-live` foi também a solução para o problema identificado: ao navegar com as setas entre as opções apresentadas no mesmo, estas não eram anunciadas pelo leitor de ecrã. Assim, com o atributo `aria-live` associado ao elemento `<button>` presente no componente do *dropdown*, de cada vez que o valor do elemento é alterado (através da navegação com as setas para cima ou para baixo), o mesmo é anunciado pelos leitores de ecrã.
Por fim, como já referido, foi adicionada uma notificação que é anunciada pelo leitor de ecrã de cada vez que o módulo é aberto, que contém a informação relativa aos dois atalhos de teclado definidos no âmbito do presente módulo.

## 6.2 Módulo Reportar Esforço

As implementações apresentadas na última secção contribuíram, em certa medida, para o trabalho desenvolvido neste segundo módulo.
Os problemas apontados no módulo Reportar Esforço assemelham-se aos discutidos, pelo que as soluções adotadas foram muito idênticas. Para além disso, partimos para as implementações de acessibilidade neste módulo com um conhecimento mais sólido, nomeadamente no que às tecnologias auxiliares diz respeito. Isto contribuiu para que os testes efetuados com o leitor de ecrã levassem menos tempo, e a crescente familiarização com o ambiente e boas práticas a seguir permitiram uma implementação mais rápida e pragmática.

### 6.2.1 *Headings*

A primeira abordagem prática relativamente ao melhoramento da acessibilidade no módulo Reportar Esforço, passou pela definição de *headings* no mesmo CS-9. Tal como no módulo Pedido de Férias/Ausências, a falta de *headings* era um problema transversal à aplicação PULSAR, na medida em que apenas existia um *heading* definido anteriormente e que dizia respeito ao título da página principal - Pessoas. Como tal, também no módulo Reportar Esforço, não tinham sido definidos *headings* até ao momento.

A existência de *headings* permite não só obter uma noção geral da estrutura e conteúdo da página, como, mais importante ainda, simplificar a navegação dos utilizadores invisuais pela mesma. Recordando, existem funcionalidades incorporadas nos leitores de ecrã que permitem listar todos os *headings* presentes numa página ou saltar para o *heading* seguinte (ou anterior). Tal como no módulo anterior, a ausência de *headings* no módulo Reportar Esforço foi apontada como um problema por um colaborador invisual na fase inicial de levantamento dos primeiros requisitos para o presente módulo.
No total, foram definidos 11 *headings* - quatro de nível 2 (`<h2>`)> e 7 de nível 3 (`<h3>`). Como sucedido, e discutido, no módulo Pedido de Férias/Ausências, não foram definidos *headings* de nível 1 uma vez que, por norma, este é apenas utilizado no título da página.

Assim, o módulo foi dividido em duas partes: secção do calendário e secção do painel lateral direito, que é aberto de cada vez que um dia no calendário é selecionado (ilustradas na Figura 6.15). Para tal, o título do calendário (representado pela informação do mês e

ano em questão) foi definido como *heading* de nível 2 (`<h2>`), juntamente com o título do painel lateral direito (i.e., data referente ao dia atual), apresentado sempre que um dia no calendário é selecionado.

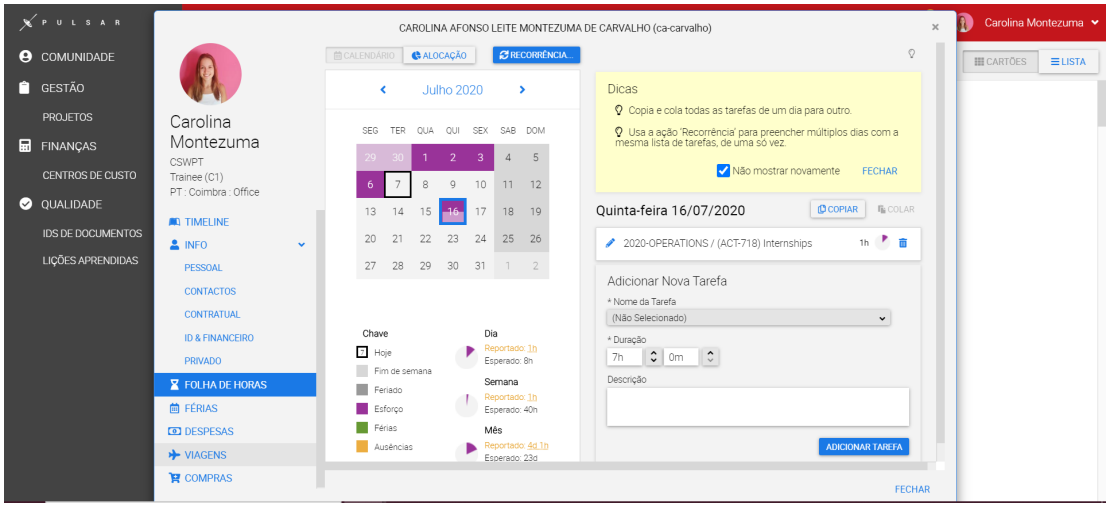


Figura 6.15: Módulo Reportar Esforço (com painel lateral direito)

A definição do título do calendário como *heading* surgiu não só como forma de facilitar a navegação para o calendário, como também para possibilitar a implementação de uma abordagem semelhante à que foi apresentada na subsecção relativa à transposição de cores ( 6.1.4) do módulo Pedido de Férias/Ausências, facilitando também a navegação para o calendário e a consulta de informação no mesmo. A implementação relativa à tranposição de cores do presente módulo será detalhada mais à frente no documento (subsecção 6.2.3). No painel lateral, foi definido o título do mesmo como *heading* de nível 2 (`<h2>`), tendo em vista facilitar o processo da navegação dos utilizadores para o mesmo. Isto porque, uma vez que - até ao momento - o foco não passava automaticamente para o painel lateral quando um dia no calendário era selecionado, a existência de *headings* definidos no painel lateral permitia auxiliar a navegação dos utilizadores invisuais no mesmo.

Dentro das duas subsecções principais definidas, foram estabelecidos como *headings* de nível 3 (`<h3>`) os títulos apresentados na legenda do calendário - `Dia`, `Semana` e `Mês`. Foram ainda definidos, como *heading* de nível 3, o título da secção de adicionar tarefas - `Adicionar Nova Tarefa` - ou, ainda, caso o utilizador selecione uma tarefa para edição, será apresentada uma secção referente à edição de tarefa, como se pode observar na Figura 6.16. Também esse título (`EDIÇÃO Tarefa`) foi definido como *heading* de nível 3 (`<h3>`).

Na Figura 6.17 é apresentada *dialog* de recorrência - apresentada após o botão Recorrência ser pressionado -, na qual também foram definidos *headings*. O título da *dialog* (Recorrência) foi definido como *heading* de nível 2. Por último, os títulos das secções para adicionar uma nova tarefa (`Adicionar Nova Tarefa`) e da secção referente à edição de tarefa apresentado na Figura 6.16, também foram definidos como *headings* nível 3 (`<h3>`).

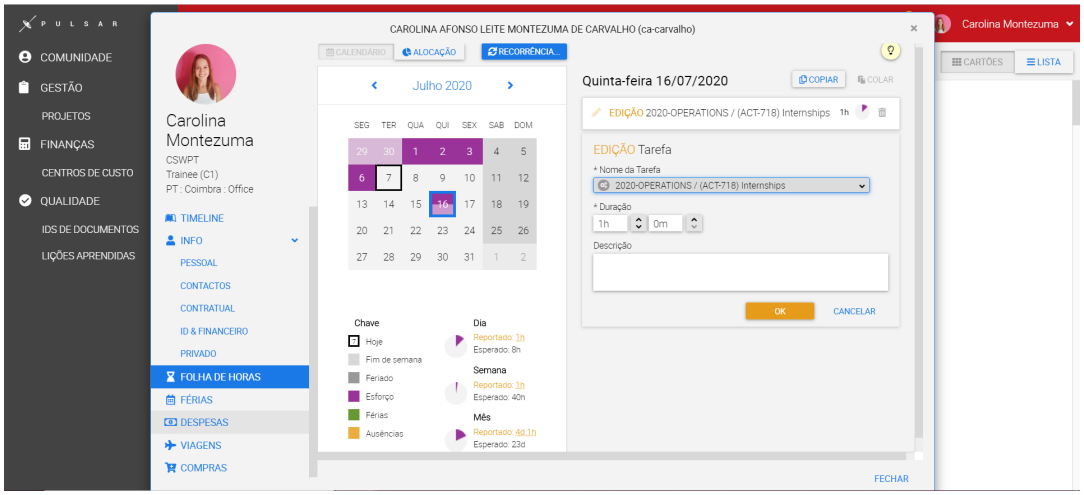


Figura 6.16: Edição de Tarefa

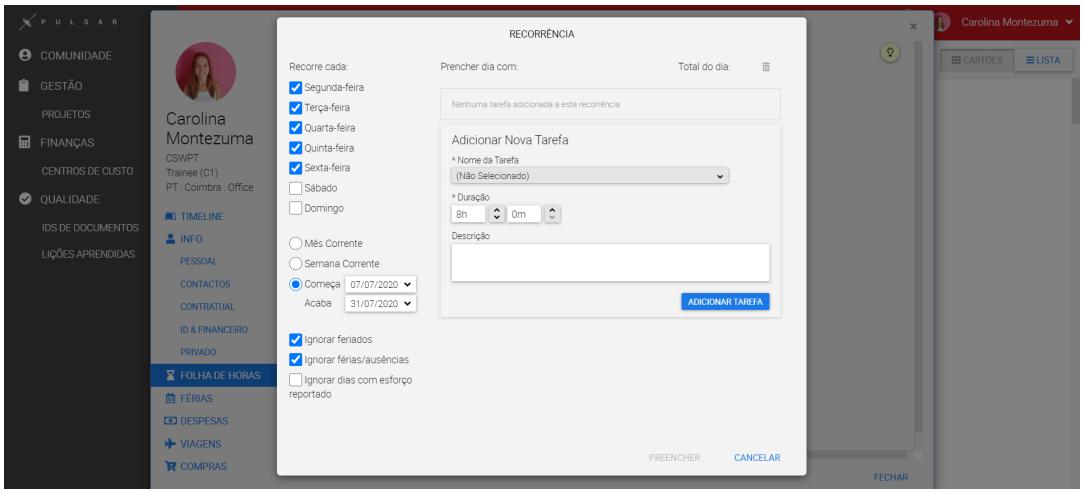


Figura 6.17: Janela Recorrência

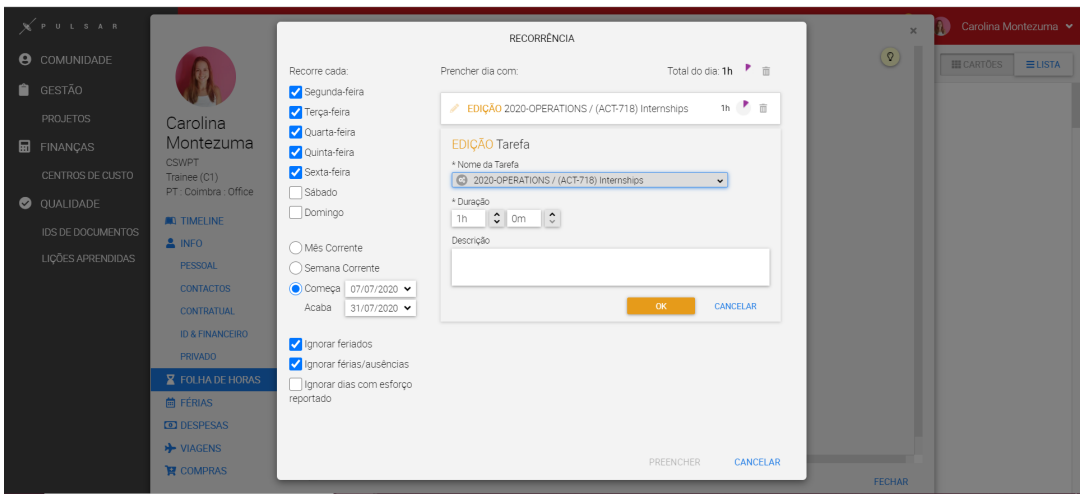


Figura 6.18: EDIÇÃO Tarefa (Janela Recorrência)

Por último, na secção referente à alocação, apresentada na Figura 6.19, foi ainda definido um *heading* nível 2 no elemento que dizia respeito ao título do projeto (e.g., 2020-OPERATIONS).

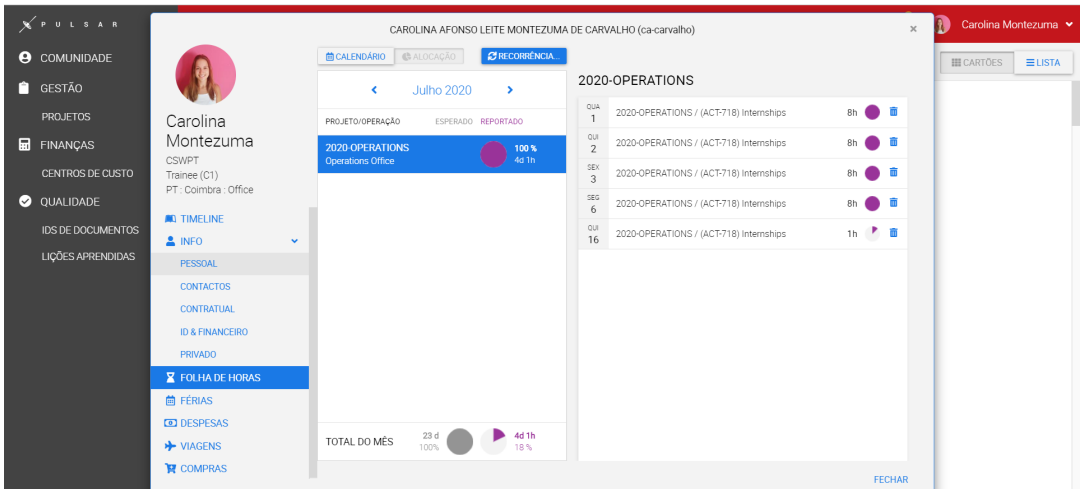


Figura 6.19: Alocação

### 6.2.2 Comportamento do Foco

Na Figura 2.3 podemos observar que, similarmente ao módulo anterior, também o módulo Reportar Esforço se trata de uma *dialog* (i.e., uma janela secundária que se sobrepõem à página principal). Porém, no presente módulo o conteúdo apresentado na página principal não fica inerte e, como tal, é designada por *dialog non-modal*.
Embora o documento [35] não estabeleca uma referência detalhada sobre *dialogs non-modals*, os autores mencionam uma diferença no que toca à implementação de uma *dialog modal*: ao contrário destas últimas, uma *dialog non-modal* apresenta, muitas vezes, uma forma de mover o foco entre a *dialog* e a página principal. À excepção da funcionalidade apresentada, a implementação de uma *dialog non-modal* passa exatamente pelo mesmo processo referido anteriormente (subsecção 6.2.2.2). Assim, deverá ser igualmente implementado foco inicial na *dialog*, assegurar a correta navegação com o teclado (Tab e Shift+Tab) e, ainda, garantir que ao navegar com o teclado o foco não sai da *dialog*.
No entanto, neste caso específico, não foi considerado vantajoso incorporar a funcionalidade adicional referente às *dialogs non-modals*, uma vez que tal implicaria a implementação de mais atalhos de teclado, aumentando assim a carga cognitiva exigida ao utilizador invisual. Para além disso, embora em [38] se evidencie que poderão haver diferenças entre *dialogs modal* e *non-modal*, este último elemento não é aprofundado. Ainda, esta decisão não afetaria minimamente a experiência do utilizador.

Os casos de uso CS-19, CS-20.CS-21 e CS-22 definidos no Capítulo 5, dizem respeito aos comportamentos do foco - análogos aos que foram descritos para o módulo Pedido de Férias/Ausências -, que deverão ser tidos em consideração para facilitar a interação de utilizadores invisuais com o presente módulo. Por essa razão, as soluções serão semelhantes às soluções descritas acima (subsecção 6.1.3).
Adicionalmente, foram ainda definidos os casos de uso CS-23, CS-24, CS-25 e CS-26 referentes à janela de recorrência.

#### 6.2.2.1 Janela do Módulo

Primeiramente, foi definido foco, quando o módulo é aberto, no primeiro elemento interativo presente no mesmo (i.e., botão de fechar presente no topo da *dialog* (Figura 6.15). No entanto, uma vez que a maioria das funcionalidades do módulo parte da utilização do calendário, também poderia ter sido apropriado definir o foco inicial no dia atual do calendário - como aconteceu no módulo Pedido de Férias/Ausências. Porém, neste caso específico, trata-se de uma *dialog* com diversas *tabs* laterais, que não engloba apenas o módulo Reportar Esforço. Para que o foco inicial ficasse uniforme - independentemente da *tab* em questão -, este foi então definido no botão fechar que, para além de comum em qualquer uma das *tabs*, é também o primeiro elemento interativo presente da *dialog*.

Em segundo lugar, foi necessário garantir a correta navegação com as teclas `Tab` e `Shift+Tab`. Isto é, ao pressionar a tecla Tab, o foco move-se para o elemento interativo seguinte e, ao pressionar as teclas `Shift+Tab`, o foco move-se para o elemento interativo anterior.
Existia, no entanto, uma situação irregular na navegação pelo módulo que dizia respeito ao elemento *segmented control* (i.e., um conjunto linear com duas opções, sendo apenas possível selecionar uma) (`Calendário/Alocação`) [Figura 6.15]. Esta irregularidade passava pelo facto de a opção disponível não ser a única a receber foco do teclado, sendo que a opção indisponível (com a qual o utilizador não podia interagir) também recebia foco. Isto acontecia porque o componente *segmented control* era criado através de dois botões, sendo que um deles visava apenas criar o efeito visual (*disabled*). Como apontado, os botões, quando são criados com a *tag* nativa do HTML - `<button>` - incorporam automaticamente o foco pelo teclado. Como tal, os dois botões eram focados, embora apenas fosse desejado focar o botão ativo.
Por forma a resolver esta lacuna, foi utilizado o atributo `tabindex`. O presente atributo é geralmente utilizado para adicionar foco a elementos que, por norma, não recebem. Porém, pode também ser utilizado para situações nas quais queremos que um elemento que - regra geral - recebe foco (como o botão) deixe de receber foco pelo teclado. Para tal, o atributo `tabindex` terá que ser adicionado ao elemento com o valor -1. Este valor permite que o elemento receba foco, mas apenas de forma programática (através do método `focus()`) e não através do do teclado. Assim, foi adicionado ao botão 'inativo' o atributo `tabindex=-1`, para que apenas o botão ativo recebesse foco pelo teclado (utilizando a tecla Tab).

Por último, ainda no contexto da navegação com o teclado, era necessário implementar o comportamento esperado nas *dialogs*, de modo a impedir que, ao navegar com as teclas Tab ou Shift+Tab, o foco se movesse para fora das mesma até que esta fosse fechada. Uma vez que no âmbito do módulo Pedido de Férias/Ausências já tinha sido criado um serviço Angular que permitia implementar esse comportamento, bastou injetar o presente serviço no componente referente à *dialog* do módulo Reportar Esforço. Assim, apenas foi necessário chamar as duas funções pertencentes ao mesmo: primeiramente, chamar a função que iria recolher todos os elementos interativos presentes na *dialog* e, de seguida, chamar a função que continha a lógica da navegação.

Como já apontámos no presente documento, durante a criação de *dialogs* é importante ter em atenção o comportamento do foco quando o utilizador interage com os elementos presentes na mesma. No presente módulo, foram anotados vários elementos em que, ao interagir com os mesmos, o foco saía da *dialog*. De seguida, descrevemos os diferentes elementos que apresentaram irregularidades, bem como a solução adoptada para cada uma delas.

No painel lateral (Figura 6.15), todos os botões apresentavam um comportamento erróneo. Isto é, caso o utilizador pressionasse os botões que permitiam editar ou eliminar uma tarefa,

o botão `Adicionar Tarefa` ou os botões para incrementar e decrementar a duração de uma tarefa ou, ainda, os botões `Ok` e `Cancelar` (Figura 6.16), o foco iria sempre mover-se para fora da *dialog*.
Assim, de forma a implementar um correto comportamento do foco após a interação com os elementos mencionados, a solução baseou-se no que os autores apontam no documento WAI Authoring Practices [37] referente ao comportamento do foco quando o botão é ativado. Reforçando o que foi anteriormente mencionado, o comportamento esperado pelo foco no contexto das *dialogs*, no caso de o botão abrir ou fechar uma *dialog*, o foco deverá mover-se para um elemento dentro da mesma e regressar para o elemento que a abriu (se possível). Relativamente ao contexto da página, caso este não seja alterado, o foco poderá manter-se no próprio botão (se este permanecer disponível). No entanto, caso exista, visto que ocorre uma alteração no contexto da página e que o botão pode deixar de estar disponível, os autores referem que o foco poderá ser adicionado a um elemento inicial permitindo, desta forma, contextualizar os utilizadores invisuais.
Começando pelo botão `Adicionar Tarefa`, uma vez que ao ativar o botão não existe mudança de contexto (i.e., o botão continua disponível), o foco foi programado para se manter no botão. Para tal, quando este é pressionado, o foco é adicionado ao botão (utilizando o método `focus()`).
Contrariamente, os botões apagar e editar associados a uma tarefa (Figura 6.15), `Ok` e `Cancelar` (relativos à edição de tarefa, Figura 6.16), deixam de estar disponíveis quando são ativados. Assim, para estes botões, o foco foi adicionado a outro elemento presente da *dialog*. No caso de o botão editar associado a cada tarefa ser ativado, foi adicionado foco no primeiro campo editável, tendo em consideração que, pelo processo lógico, seria o campo para o qual os utilizadores iriam navegar após selecionar a edição de tarefa. Quando os botões `Ok` e `Cancelar` apresentados na secção de edição de uma tarefa (Figura 6.16) são pressionados, existe uma mudança no painel: a secção deixa de estar disponível, voltando a aparecer a secção para adicionar uma tarefa (Figura 6.15). Por essa razão, o foco foi adicionado ao *heading* inicial do painel, referente à data do dia selecionado.

No caso do botão para apagar tarefa, a interação com o mesmo despoleta o aparecimento de uma janela *popup*. Consequentemente, o foco deverá mover-se para a mesma - neste caso, para o botão `Apagar`. Ao fechar o *popup*, pressionando o referido botão, o foco deveria retornar ao botão apagar que abriu a *dialog*. Porém, a ordem de renderização dos elementos fazia com que o botão fosse carregado depois do processo do foco, o que levava a que o último não fosse atribuído ao botão, como desejado. De forma a evitar a utilização de *timeouts* para esperar que o botão estivesse disponível e conseguisse, assim, receber foco, foi adicionado foco novamente no *heading* inicial.
Ainda no contexto de apagar uma tarefa, no caso de o utilizador ter reportado esforço através de recorrência, ao tentar apagar a mesma, é apresentada uma *dialog modal* com quatro botões presentes na mesma - apresentada na Figura 6.20.

Esta contém diferentes opções para apagar recorrências (`Apagar Esta Tarefa Apenas`, `Apagar Recorrência A partir De Hoje`, `Apagar Recorrência Meses abertos`) e um botão para fechar a *dialog* (Cancelar). Como tal, de forma a definir o foco num elemento da *dialog* quando esta abre, o foco inicial foi adicionado ao primeiro botão apresentado (`Apagar Esta Tarefa Apenas`). Ainda, quando a *dialog* fecha, o foco é adicionado ao *heading* inicial (à semelhança do que foi descrito acima)

Na *dialog*, houve ainda mais três situações que inspiraram cuidados relativamente ao comportamento do foco. No caso do elemento *segmented control* (`Alocação/Calendário`), ao selecionar a opção disponível, o foco movia-se para fora da *dialog*. Para corrigir este comportamento, estabeleceu-se que ao pressionar uma das opções do *segmented control*, o foco passaria a mover-se para a outra opção disponível.

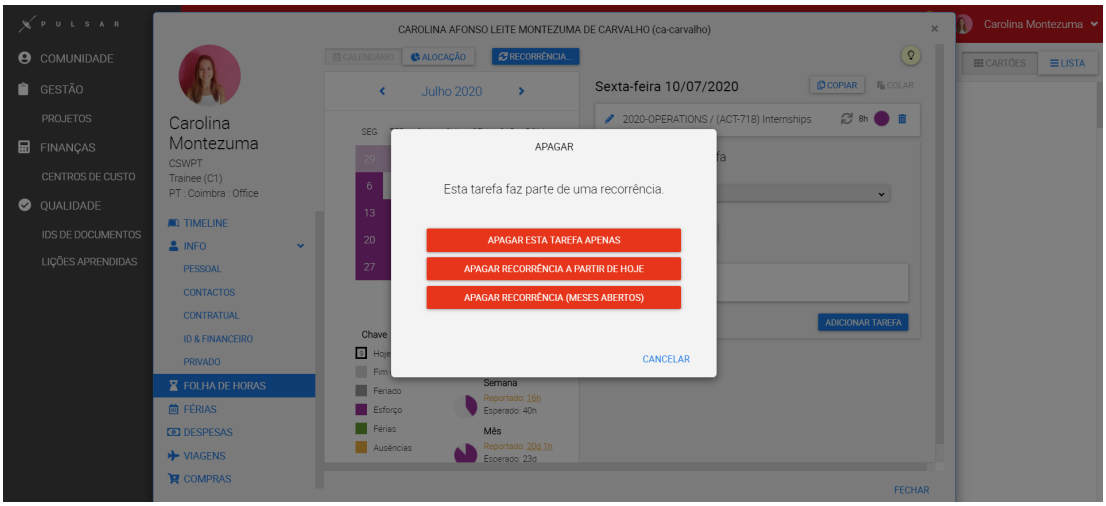


Figura 6.20: Apagar Recorrência

A *dialog* apresenta também um botão para dicas - que abre um *popup* com a informação de ajuda - cujo comportamento do foco também era erróneo. Quando o botão era ativado, o foco deslocava-se para outro elemento fora da *dialog*. Assim, a resolução deste problema passou por programar o foco de forma a que este se deslocasse para o botão `FECHAR`, retornando ao botão inicial quando o *popup* era fechado (Figura 6.15).

No caso dos botões incrementar e decrementar, apresentados também na Figura anterior, quando atingem o valor máximo ou o valor mínimo, respetivamente, ficam indisponíveis e, como tal, o foco move-se para o outro botão disponível (incrementar ou decrementar).

Por último, o caso de uso CS-22 diz respeito ao comportamento do foco quando um dia no calendário é selecionado. Este representa a necessidade - apresentada por um dos colaboradores - de mover o foco automaticamente para o painel lateral quando este abre. Desta forma, já não teria que passar pelo processo de navegação para o painel. Para tal, quando um dia era selecionado no calendário, o foco foi adicionado ao *heading* inicial do painel lateral.

#### 6.2.2.2 Janela de Recorrência

O foco inicial - ao abrir a *dialog* de recorrência (Figura 6.17) - foi adicionado à primeira *checkbox* apresentada, que é também o primeiro elemento interativo presente na *dialog*, e faz parte de uma lista de *checkboxes* apresentadas que dizem respeito aos dias da semana que irão fazer parte da recorrência.

De forma equivalente ao que foi implementado na janela principal no módulo Reportar Esforço, foram implementadas as duas funções especificadas anteriormente (pertencentes ao serviço implementado em 6.1.3), que permitiu garantir que o foco se mantinha dentro da *dialog*. Neste caso, a esta também se trata de uma *dialog modal* e, como tal, o foco nunca deverá sair desta até que seja fechada (i.e., utilizando a tecla Escape ou o botão `Cancelar`).
Para garantir que o foco não se movia para fora da janela de recorrência, foi necessário programar o foco (utilizando o método `focus()`) aquando da interação com certos elementos da mesma. Foram estes: Adicionar Tarefa; editar ou apagar (associado a uma tarefa); apagar (relativo a todas as tarefas); `Ok`, e `Cancelar` (secção de edição de tarefa);

No primeiro caso (botão `Adicionar Tarefa`, Figura 6.17), uma vez que o botão se mantém disponível após a interação do utilizador, o foco foi adicionado ao próprio elemento. Quando o utilizador pressiona o botão editar (associado a uma tarefa), é apresentada a secção relativa à edição de tarefa (como observamos na Figura 6.21). O foco foi programado para o primeiro campo interativo da secção de edição de tarefa (i.e., *dropdown* com os nomes das tarefas), equivalente ao que foi realizado na *dialog* principal do módulo Reportar Esforço.

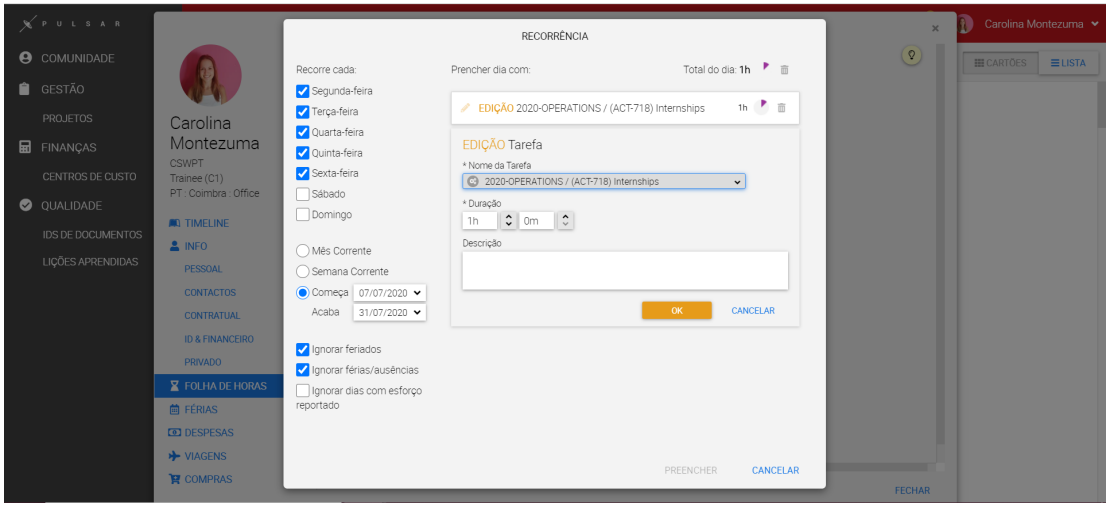


Figura 6.21: EDIÇÃO Tarefa (Janela de Recorrência)

Relativamente aos botões apagar associados a cada tarefa e o botão que permite apagar todas as tarefas adicionadas ao dia, o foco é adicionado ao *heading* `Adicionar Nova Tarefa`, uma vez que ficam indisponíveis após o utilizador interagir com os mesmos.

Após a interação por parte do utilizador com os botões `Ok` e `Cancelar` - relativos à edição de uma tarefa -, foi adicionado foco no botão `Preencher` já que, em ambos os casos, os botões deixam de estar disponíveis na interface. Para além disso, assumindo que após a edição o utilizador quererá terminar o preenchimento, foi tomada a decisão de adicionar o foco no botão `Preencher` 6.17.

Ainda, a partir de testes realizados utilizando o teclado, detetámos que o comportamento do *radio group* (i.e., um conjunto de *radio buttons*) apresentado não era o correto. Isto porque, ao pressionar a tecla `Tab`, o foco não se movia para a opção selecionada - o que seria expectável -, e movia-se para o primeiro *radio button*, fosse este o selecionado ou não. Em [36], os autores mencionam que, ao utilizar as teclas `Tab` ou `Shift+Tab`, caso não esteja nenhum *radio button* selecionado, o foco move-se para o primeiro *radio button* e, caso exista um selecionado, o foco deverá mover-se para esse. Para além disso, a navegação com as teclas cima e esquerda, baixo e direita, deverá mover o foco para o próximo *radio button* pertencente ao radio group, ou para o *radio button* anterior, respetivamente ( sendo que a opção selecionada será sempre a que tem o foco). De forma a incorporar os comportamentos descritos acima, foi necessário atribuir a cada um dos *radio button* o atributo `name` sendo que o valor associado ao mesmo tinha que coincidir para os três *radio button* apresentados para que estes funcionassem como um *radio group*.

Por último, foi necessário definir o foco quando a janela de recorrência é fechada, tendo o mesmo sido definido no botão Recorrência (Figura 6.15), considerando que foi o botão que a abriu em primeiro lugar.

### 6.2.3 Transposição de Cores

No módulo Reportar Esforço, a utilização de cores com o fim de fornecer informação no calendário foi, também, o problema mais enfatizado pelos utilizadores invisuais, tal como aconteceu no módulo Pedido de Férias/Ausências.

Na Figura 2.3 podemos observar os dias com esforço reportado no calendário. Os dias que cujo esforço reportado é equivalente ao expectável são apresentados a roxo. Por sua vez, os dias que têm esforço parcial (i.e., foram reportadas menos horas do que as expectáveis) são representados com duas tonalidades de roxo (uma mais clara). Por fim, os dias com esforço em excesso apresentam também duas tonalidades de roxo (uma mais escura). Embora não apresentada na figura, existe ainda informação relativa a férias (verde), ausências (amarelo) e feriados (cinzento escuro).

O processo de consulta da informação mencionada - dias sem esforço, dias com esforço parcial ou dias com esforço excedente -, torna-se exaustivo para os utilizadores invisuais, uma vez que a informação representada por cores no calendário não se encontra acessível a este tipo de utilizadores. Como tal, para consultar a informação das horas diárias, os utilizadores invisuais até ao momento teriam que percorrer todos os dias do mês e, para cada dia, selecionar o mesmo de forma a que a legenda (`Dia`) fosse atualizada 2.3. Após selecionarem o dia, teriam então que navegar - utilizando o teclado - para a legenda e, de seguida, regressar novamente ao calendário até verificarem todos os dias do mês. Para além disso, à medida que a navegação era feita pelo calendário, não existia informação disponível que indicasse se o dia em questão era um feriado ou continha férias ou ausências marcadas.

As soluções apresentadas na presente subsecção serão fortemente baseadas nas que foram adoptados no módulo anterior. Isto porque, para além de termos recebido *feedback* positivo sobre as soluções anteriormente implementadas, o facto de adoptar uma abordagem semelhante nos dois módulos permitia facilitar o processo de aprendizagem dos utilizadores invisuais.
Primeiramente, foi adicionada informação sobre o mês no *heading* referente ao título do calendário. Assim, navegando para o mesmo, o utilizador terá acesso imediato a um conjunto de informação referente ao mês. A informação indica os dias que ainda não têm esforço reportado, os dias com esforço parcial ou esforço excedente. No entanto, esta informação não é detalhada no caso de ainda não ter sido reportado esforço em nenhum dia do mês ou já ter sido reportado todo o esforço expectável no mesmo. Neste cenário, será indicada a informação de que tal está em falta ou que o esforço mensal foi atingido, respetivamente.
Esta informação foi adicionada ao *heading* através do atributo `aria-describedby`, tal como aconteceu no módulo anterior. Relembrando a funcionalidade do atributo, este permite adicionar a informação presente num elemento (identificado por um *id*) à descrição de outro elemento. Isto é, foi criado um elemento `<div>` com a informação relativa ao mês. Este elemento tinha a si associado um *id* que era usado como referência no atributo `aria-describedby` associado ao elemento *heading* do mês.

Ao atributo `aria-label` associado a cada um dos dias do calendário, foi adicionada informação equivalente à que é representada pelas cores no calendário, em adição à informação já presente e que dizia respeito à data do dia em questão. Desta forma, o utilizador, ao navegar pelo calendário com o teclado (utilizando as setas), terá *feedback* do leitor de ecrã equivalente ao que é dado através das cores. Isto é, caso o dia já tenha horas reportadas, a *label* do dia irá conter informação relativa ao esforço do dia (i.e., esforço expectável, excedido ou parcial). Adicionalmente, irá informar quais as tarefas associadas (Esforço, Ausências, Feriado ou Férias), tal como aparece na legenda. Caso o dia não te-

nha horas preenchidas, não será adicionada informação complementar ao valor do atributo `aria-label`, ficando apenas com a informação que já existia anteriormente.

Por último, verificámos que os colaboradores invisuais recorrem habitualmente à legenda - Dia, Semana e Mês (Figura 2.3) -, para consultar informação. Assim, a informação sobre os dias sem esforço, com esforço parcial ou esforço excedente, foi também adicionada à legenda (acessível apenas a leitores de ecrã), organizada por semana e por mês.
Para adicionar esta informação relativa à semana e ao mês, tendo em conta que a mesma não deveria ser apresentada na página, foram adicionados dois novos elementos (não-visíveis) (`<div>`), posicionados abaixo da informação relativa às horas esperadas da semana (legenda `Semana`) e às horas esperadas do mês (legenda `Mês`).

Desta forma, os utilizadores invisuais, ao navegarem na legenda como recurso às setas para cima e para baixo do teclado (como fazem habitualmente), conseguem aceder também a essa informação. No entanto, a informação referida será ser disponibilizada na legenda caso o mês não esteja já devidamente preenchido (i.e., de acordo como o expectável).

### 6.2.4 Atalhos de Teclado

As funcionalidades implementadas através de atalhos de teclado no presente módulo contêm semelhanças às implementações verificadas no primeiro módulo. Paralelamente, ambas visavam facilitar o processo de navegação dos utilizadores no calendário, neste contexto suportados pelos casos de uso CS-34 e CS-35. Estes descrevem a navegação para o dia atual e para o dia selecionado no calendário, respetivamente.

O atalho de teclado para navegar para o dia selecionado surgiu por sugestão de um dos utilizadores invisuais. Isto porque, durante um teste realizado com o utilizador, após ter sido adicionada uma nova tarefa (o foco encontrava-se no painel lateral), o utilizador queria regressar para o calendário para continuar a reportar esforço nos dias seguintes. Para isso, teria que navegar com o teclado até chegar novamente ao dia selecionado. Por essa razão, o utilizador reportou que seria útil existir um atalho que permitisse regressar imediatamente ao dia selecionado no calendário.
De forma a implementar o atalho mencionado, foi utilizada a função `querySelector()`, a qual recebia como parâmetro o valor `'.selected'` - *selector* referente à classe CSS presente no dia selecionado no calendário. Assim, a função retorna o elemento referente ao dia selecionado no calendário. Desta forma, quando o atalho de teclado é utilizado, será adicionado foco (utilizando o método `focus()`) ao elemento devolvido pela função (i.e., dia selecionado no calendário).
Foram definidos dois atalhos de teclado para esta funcionalidade, baseados nos que já tinham sido implementados anteriormente. O utilizador poderá assim navegar para o dia selecionado utilizando um dos seguintes atalhos: `Alt+s` ou `Ctrl+Shift+s` (*s* referente a *selected*). Desta forma, pretendeu-se, em primeiro lugar, seguir o mesmo princípio de atalhos definidos no módulo anterior (subsecção 6.2.4) utilizando iniciais e, em segundo lugar, evitar eventuais conflitos com atalhos de teclado já existentes (quer a nível de sistema operativo, *browser* ou leitores de ecrã). Caso o atalho de teclado seja utilizado mas não exista nenhum dia selecionado, o foco irá manter-se na mesma posição.

Por último, de forma a o utilizador conseguir navegar para o dia atual no calendário, foram adicionados dois atalhos, análogos aos que tinham sido adicionados no módulo Pedido de Férias/Ausências: `Alt+t` e `Ctrl+Shift+t` (*t* referente a *today*). O presente atalho foi definido para permitir ao utilizador a navegação para o calendário - caso não exista dia selecionado -, sendo que o dia atual serve assim como dia de referência.

A implementação do presente atalho foi semelhante à implementação do atalho anterior, também com recurso à função `querySelector()`, divergindo apenas no parâmetro passado na função. Neste caso, o parâmetro é *'.today'*, que equivale à classe CSS adicionada ao dia atual no calendário. Assim, a função retorna o elemento referente ao dia atual no calendário, sendo que o mesmo irá receber foco através do método `focus()`.

Os atalhos de teclado disponibilizados no módulo terão que ser apresentados (i.e., anunciados) aos utilizadores invisuais de forma a que estes sejam conhecidos pelos mesmo. Esse processo será apresentado na secção seguinte.

### 6.2.5 Notificações do Utilizador

As notificações para utilizadores invisuais foram implementadas para dar resposta a diferentes situações. Numa primeira instância, foi importante disponibilizar as mensagens que eram apresentadas no ecrã em *toasts* (referido na secção 6.1.6 do módulo anterior). Estas não se encontravam acessíveis aos utilizadores invisuais, uma vez que os leitores de ecrã não identificavam as mensagens quando apareciam na página. Complementarmente, era necessário garantir que os utilizadores invisuais seria alertados quando existisse informação por preencher, dado que até ao momento, como é ilustrado nas Figuras 6.22, 6.23 e 6.24, tal informação era apenas transmitida por cores.
Ainda, a legenda referente às horas diárias (`Dia`) é atualizada em diversas situações, sendo estas: é selecionado um dia no calendário; uma tarefa é adicionada; uma tarefa é editada; uma tarefa é apagada. Para todas as situações foi igualmente importante garantir que os utilizadores invisuais tinham *feedback* imediado destas alterações.

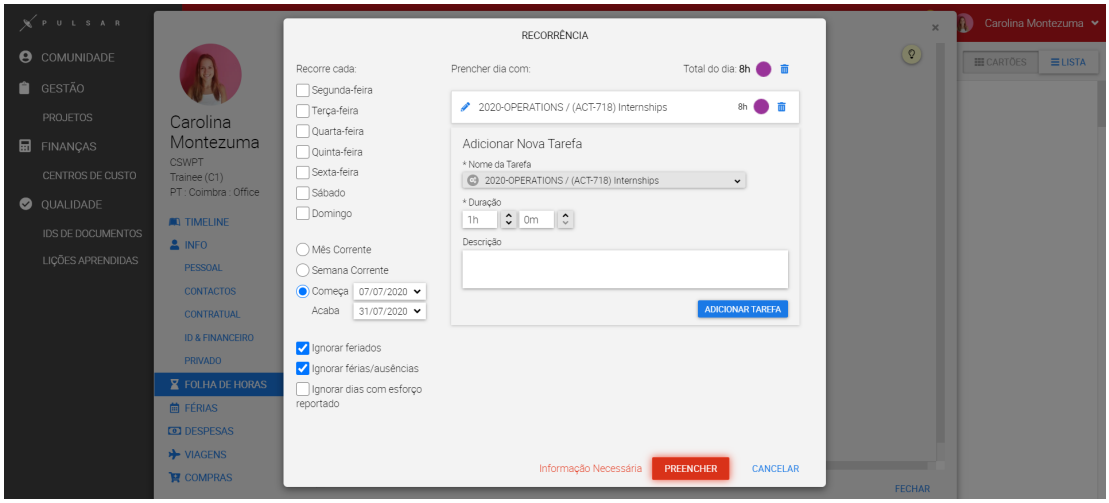


Figura 6.22: Informação Necessária (Preencher Recorrência)

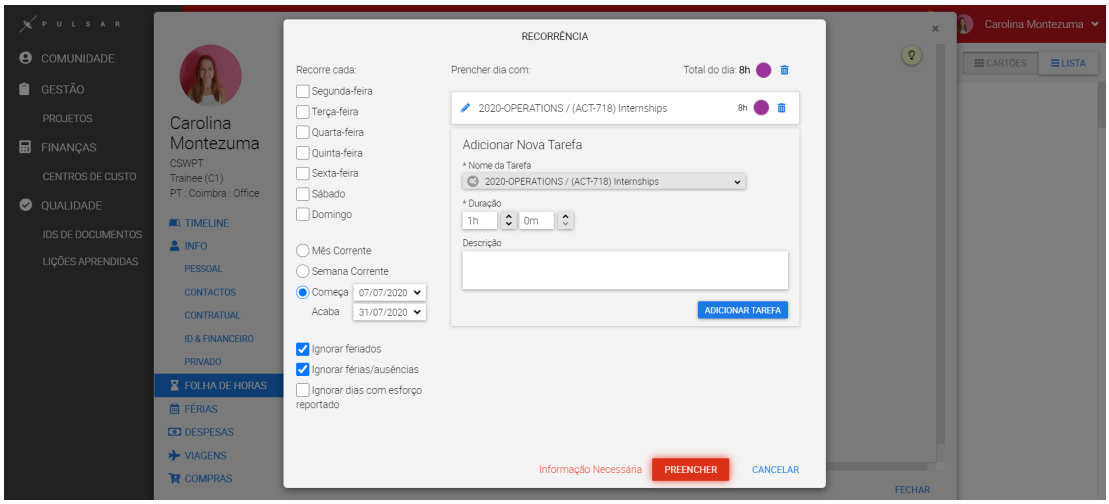


Figura 6.23: Informação Necessária (Preencher Recorrência)

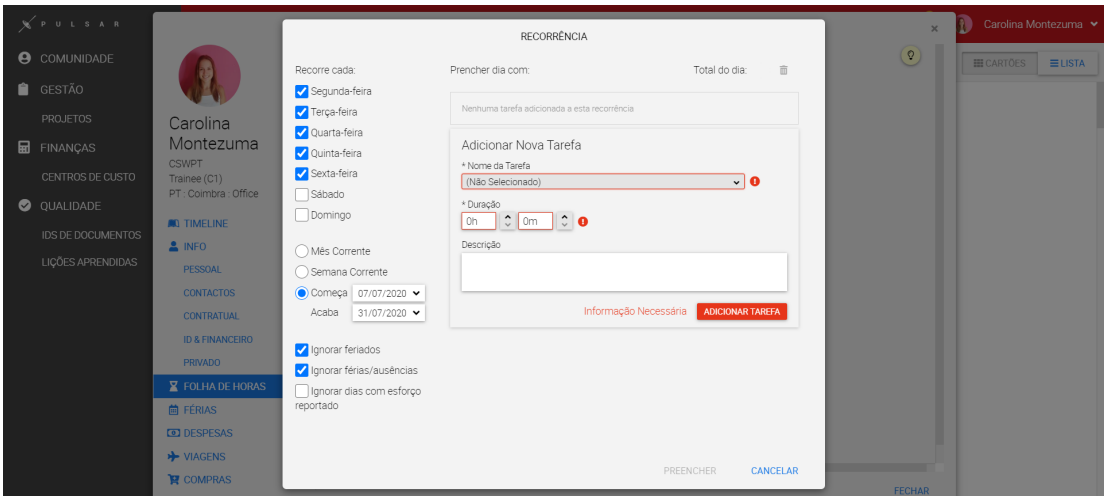


Figura 6.24: Informação Necessária (Preencher Recorrência)

Finalmente, tal como aconteceu no módulo Pedido de Férias/Ausências, foi adicionada uma notificação para dar a conhecer os atalhos de teclado definidos no presente módulo (subsecção 6.2.4).

De forma semelhante, criámos elementos não-visíveis (`<div>`) que irão conter as mensagens informativas. Estes elementos foram posicionados no início do módulo (local onde habitualmente são posicionadas as mensagens referentes à acessibilidade; isto é, no topo). A implementação das notificações anunciadas pelos leitores de ecrã teve por base a utilização do atributo `aria-live`. Como vimos (subsecção 6.1.6), o atributo `aria-live` permite informar os leitores de ecrã que o elemento ao qual o atributo estará associado poderá sofrer atualizações. Assim, quando o mesmo é atualizado, o leitor de ecrã irá anunciar essas alterações.
Quando um utilizador finaliza uma ação referente a adicionar uma tarefa ou eliminar uma tarefa, uma mensagem é temporariamente apresentada no ecrã, como se pode observar nas Figuras 6.25 e 6.26.

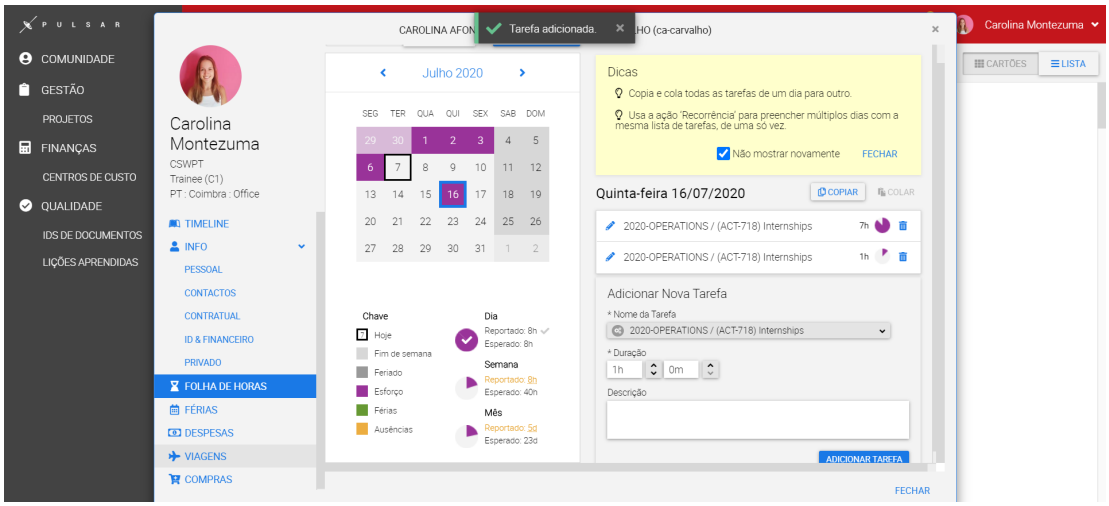


Figura 6.25: Tarefa Adicionada

e

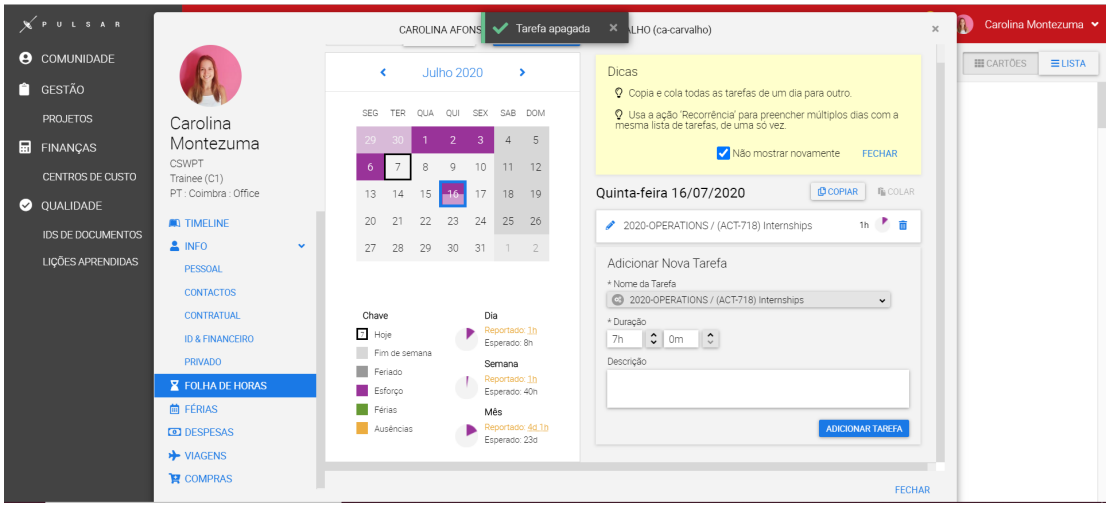


Figura 6.26: Tarefa apagada

As notificações adicionadas para os utilizadores invisuais nestas situações correspondem às mensagens apresentadas no *toast*. Assim, caso o utilizador elimine uma tarefa ou adicione uma nova, o elemento com o atributo `aria-live` associado será atualizado com a mensagem equivalente à que é apresentada no ecrã.

No que diz respeito à atualização da legenda `Dia`, um dos colaboradores invisuais referiu que, quando adicionava uma nova tarefa a um dia do calendário, uma vez que não existia nenhum *feedback* relativo aos novos dados da legenda (nem ao sucesso da ação), teria sempre que navegar até à legenda por forma a consultar a informação.
Por esta razão, foi adicionada uma notificação sempre que a legenda é atualizada (e.g., "`Horas reportadas: 6 Horas esperadas: 8`"). Esta informação é anunciada pelos leitores de ecrã caso se verifique uma das situações já mencionadas anteriormente (i.e.,

selecionar um dia no calendário; adicionar tarefa; editar tarefa; apagar tarefa).
Desta forma, o utilizador não terá que navegar para a legenda do dia para confirmar as horas no fim de cada uma destas ações.

No que diz respeito às situações em que falta preencher informação necessária - quer na *dialog* do módulo ou na *dialog* de recorrência -, caso o utilizador esteja a adicionar uma nova tarefa ou a editar tarefa, é apresentada uma mensagem junto dos respetivos campos que informa o utilizador que falta informação ("Informação Necessária") [(Figuras 6.22 e 6.23]. Complementarmente, os campos de preenchimento obrigatório (i.e., Nome da Tarefa e Duração) que se encontrem por preencher serão apresentados com a cor vermelha. Porém, como esta informação gráfica não é acessível aos utilizadores invisuais, a notificação adicionada engloba também os campos por preencher. Em adição, para os utilizadores invisuais terem conhecimento dos campos de preenchimento obrigatórios assim que entram no módulo, foi adicionado o atributo `aria-required` (com valor *true*) aos mesmos.

Ainda no que diz respeito aos erros por falta de informação necessária, na *dialog* de recorrência, quando o utilizador clica no botão que finaliza o preenchimento - `Preencher` - e não preencheu nenhuma das *checkboxes* relativas aos dias de recorrência, é apresentada uma mensagem na *dialog* - `‘‘Informação Necessária’’`. No entanto, ao contrário das situações apresentadas acima, como podemos observar na Figura 6.24 os campos de preenchimento obrigatório não são identificados à partida, como também não são sinalizados os campos que foram identificados como por preencher (em caso de erro). Assim, para que esta informação fosse disponibilizada igualmente aos utilizadores invisuais, foi adicionada uma notificação que os informa disso mesmo (i.e., `‘‘Informação Necessária’’`).

Na *dialog* de recorrência, existe também uma legenda referente ao total do dia (em horas). Quando os utilizadores adicionam, editam ou eliminam uma tarefa, essa legenda é atualizada. Para que esta informação fosse anunciada aos utilizadores invisuais, neste caso, foi adicionado o atributo aria-live diretamente ao elemento referente à legenda, com o valor *polite*.

Como referido na subsecção 6.2.4, foram definidos dois atalhos de teclado no presente módulo - navegar para o dia atual e navegar para o dia selecionado. Como tal, seria necessário dar a conhecer os atalhos de teclado disponíveis aos utilizadores invisuais quando estes entram no módulo. Para isso, foi adicionada uma notificação quando o utilizador abre o módulo.

This page is intentionally left blank.

# Capítulo 7

# Testes de *Software*

Ao longo deste documento, após destacarmos o trabalho preliminar realizado, apresentámos as várias soluções implementadas atendendo aos requisitos definidos, mantendo sempre em vista os dados recolhidos e apresentados no Capítulo 3 (Trabalho Relacionado). Tendo terminado a fase de implementação, e com base no que apresentámos no último capítulo, podemos verificar que os objetivos principais aos quais nos propusemos foram alcançados. Porém, sabendo que todo este projeto tem a si associado uma forte componente de interação utilizador/aplicação, seria imprudente concluir que os objetivos foram atingidos sem realizar testes que não só reforçassem esta ideia, mas que permitiriam também medir, em determinada escala, o impacto prático trazido pelo melhoramento da acessibilidade dos módulos (Pedido de Férias/Ausências e Reportar Esforço). Para tal, realizámos testes de usabilidade, que serão descritos mais à frente neste capítulo.

Como já apontado, todo o trabalho conduzido durante este segundo semestre teve, paralelamente, testes auxiliares: inicialmente como base para o levantamento dos requisitos, e durante todo o processo de implementação, no qual o *feedback* recorrente dos colaboradores invisuais foi fulcral para o desenvolvimento do mesmo, assim como testes manuais e automáticos.

Os testes de *software* dividiram-se em dois passos: verificação e validação. A verificação de *software* consistiu na análise estática do código, enquanto que a validação teve como objetivo avaliar a acessibilidade dos módulos dois módulos trabalhados.
Numa fase inicial, pretendia-se investigar os problemas de acessibilidade existentes em cada um dos módulos, enquanto que na fase final da implementação procurou-se validar as alterações realizadas no âmbito das melhorias de acessibilidade nos módulos.

Assim, no presente capítulo descrevemos de forma mais clara e detalhada todo o processo de avaliação estabelecido ao longo do semestre, apresentando também os testes finais realizados com os utilizadores, que nos permite fazer um balanço mais preciso de todo o trabalho elaborado.

## 7.1 Análise do Código

A análise estática do código verificava-se após a conclusão do processo de implementação de uma tarefa. Com efeito, a tarefa era dada como terminada aquando da análise do código, assegurando assim que, posteriormente, as novas alterações realizadas pudessem

ser enviadas para produção. Esta análise pressupunha uma componente manual, isto é, uma revisão por membros da equipa de *front-end*, e outra automática, através da extensão do IDE (SonarLint, secção 3.2). Desta forma, mantínhamos uma revisão automática que decorria simultâneamente à implementação do código, permitindo desta forma identificar problemas atempadamente.

Para cada uma das tarefas implementadas no âmbito do presente estágio era sempre criado um *branch*, para que as alterações realizadas no âmbito da tarefa não afetassem o código principal. Quando a implementação de uma tarefa era dada como terminada, após terem sido realizados *commits* com todo o código e o mesmo ter sido enviado para o Bitbucket (ao fazer *push* do código), era criado um *Pull Request*.

Um *Pull Request* é uma funcionalidade (associada ao Bitbucket) que permite agilizar o processo de revisão de novo código por parte de terceiros (i.e., diferentes membros da equipa) antes do mesmo ser integrado no código principal [21]. O *Pull Request* permite também que os membros da equipa, destacados para a revisão do código, sejam notificados. Isto assegura que os membros responsáveis tenham conhecimento das respectivas implementações, podendo então dar seguimento à revisão da tarefa.
Através da análise ao *Pull Request*, os outros programadores iriam ter acesso a todas as alterações realizadas ao código no respetivo *branch*, podendo fazer comentários com sugestões para melhorar a solução implementada, melhorar a qualidade do código realizado e detetar eventuais *bugs*. É, também, uma forma de dar a conhecer o código realizado no âmbito de uma tarefa a outros membros da equipa. Tudo isto visa promover o destacamento de eventuais erros, ou até mesmo sugestões de otimização do código.

Quando um *Pull Request* era criado, eram adicionados - regra geral - entre três a quatro membros da equipa à revisão do código. Quando um membro termina a revisão do código, existem duas opções no Bitbucket: *Needs Work* ou *Approved*. A primeira é selecionada caso o revisor considere que o código poderá ter que sofrer alterações. Quando o revisor não considera que seja necessário realizar alterações, o *Pull Request* é marcado com a segunda opção (*Approved*). Quando os revisores sugeriam alterações ao código, após as novas alterações serem implementadas, estas necessitavam de ser revistas novamente. Para que uma tarefa fosse concluída, era necessário que dois revisores marcassem o *Pull Request* como *Approved*, e só depois poderia ser realizado um *merge* das alterações (i.e., juntar o código realizado ao código principal).

A análise de código é um processo relevante na medida em que não só pode prevenir que eventuais *bugs* entrem em produção, como também permite otimizar a qualidade do código realizado, através da troca de opiniões e conhecimento por parte dos programadores.
No âmbito do presente estágio, este processo tornou-se muito importante dada a falta de experiência nas tecnologias utilizadas e, como tal, o processo de revisão e discussão com outros membros permitia não só conhecer mais sobre as tecnologias como adquirir boas práticas. Apenas após este conjunto de testes se verificar (i.e., após o código relativo à implementação da tarefa ser revisto), passaríamos aos testes de acessibilidade, descritos mais à frente neste capítulo (subsecção 7.2.2).

## 7.2 Avaliação de Acessibilidade

De seguida iremos descrever o processo de validação realizado ao longo de todo o desenvolvimento, e que diz respeito à avaliação de acessibilidade nos módulos Pedido de Férias/Ausências e Reportar Esforço.
A presente secção encontra-se dividida em quatro subsecções. São estas: testes prelimina-

res, que dizem respeitos aos testes realizados com intenção de definir os primeiros requisitos; testes de desenvolvimento, que dizem respeito aos testes realizados durante o processo de implementação; conformidade com as diretrizes; e, por fim, os testes de usabilidade realizados após a fase de implementação dos dois módulos.

### 7.2.1 Testes preliminares

A primeira fase de avaliação de acessibilidade passou por registar os módulos nos quais os utilizadores invisuais sentiam mais dificuldades. Assim, com o auxílio de um colaborador invisual da empresa, apontámos aqueles que seriam os módulos mais críticos em termos de acessibilidade e de relevância. Como sabemos, foram destacados dois módulos como objetivos principais - Pedido de Férias/Ausências e Reportar Esforço.

Com o auxílio de um dos colaboradores invisuais da empresa, foram definidos os primeiros requisitos (funcionais) para os dois módulos. Para isso, o colaborador interagiu com ambos os módulos e reportou quais as dificuldades sentidas durante a interação com o módulo. Embora já discutidas neste documento, apresentamos abaixo, de forma sucinta, a lista com os problemas e sugestões apontados:

- Falta de *headings* nos dois módulos;
- Não existia uma forma eficaz para consultarem a informação presente nos calendários presentes nos dois módulos, uma vez que a informação existia apenas em cores;
- O módulo Pedido de Férias/Ausências apresentava elementos sem nomes acessíveis;
- No módulo Pedido de Férias/Ausências, quando o utilizador selecionava um dia no calendário e abria uma janela *popup*, o foco não passava automaticamente para esta.
- No módulo Reportar Esforço, ao selecionar um dia no calendário, iria ajudar se o foco passasse automaticamente para o painel lateral que é apresentado;
- Os elementos *dropdown* apresentados nos dois módulos não funcionam como esperado uma vez que ao navegar entre as opções, estas não eram anunciadas pelo leitor de ecrã.

Esta avaliação foi bastante relevante, uma vez que numa fase inicial do projeto era fundamental compreender quais as dificuldades que os utilizadores invisuais sentem, como é que navegam pelas páginas com o leitor de ecrã e como operam perante alguns dos problemas. Todos estes testes e análises foram importantes para clarificar as necessidades dos utilizadores invisuais e como melhorar os módulos de modo a responder às mesmas.

### 7.2.2 Testes de Desenvolvimento

Durante todo o processo de implementação, foram realizados regularmente testes em ambos os módulos. Estes testes dividiam-se em duas categorias: testes locais e os testes de utilizadores.

Os testes locais ocorriam par a par com o processo de implementação, sendo que, por regra, cada alteração ao código era acompanhada por testes. Tal permitia verificar se as novas alterações não tinham comprometido a funcionalidade e o aspeto visual do que tinha sido implementado até então - o que era de extrema importância assegurar. Adicionalmente,

tirámos partido do leitor de ecrã, que nos permitia testar a acessibilidade no imediato. Desta forma, poderíamos corrigir eventuais erros de acessibilidade antes dos testes com os utilizadores invisuais, prevenindo que tais alongassem o tempo de teste desnecessariamente.
Porém, importa notar que os primeiros testes com o leitor de ecrã revelaram-se morosos numa fase inicial, fruto da inexperiência perante esta nova tecnologia. Sendo uma tecnologia desconhecida até então, algumas funcionalidades não eram diretas, como os diferentes atalhos providenciados pelo leitor de ecrã. Adicionalmente, era por vezes difícil garantir que o *feedback* dado pelo leitor de ecrã era o esperado ou o correto, na medida em que não estávamos familiarizados com o tipo de informação anunciada.

Geralmente, os testes com os utilizadores tinham lugar após a conclusão de uma determinada tarefa, depois de já ter sido concluída a análise do código realizado. Após a análise ao código, os testes realizados com os utilizadores finais (consoante a disponibilidade dos mesmos) serviam como validação dessa tarefa, havendo a possibilidade de serem posteriormente adicionados requisitos (ou reajustados) consoante o *feedback* recebido ao longo dos testes.
As excepções passavam por situações em que surgiam dúvidas durante o processo de implementação, onde se verificava mais vantajoso pedir *feedback* a, pelo menos, um utilizador invisual, antes da tarefa ser dada como concluída (i.e., poder ser enviada para produção). Quando existiam dúvidas relativamente à implementação, era pedido a pelo menos um dos colaboradores invisuais que testasse a funcionalidade em questão. Para isso, o código alterado era disponibilizado num ambiente de teste para que o utilizador conseguisse testar e só depois, após o *feedback* e caso não fosse necessário fazer mais alterações, era dada a tarefa como concluída.

A título de exemplo, na primeira tarefa implementada, que dizia respeito à definição *headings*, quando estes foram implementados e testados localmente, não eram apresentados na lista de *headings* do leitor de ecrã (ou seja, não era detetada a existência de *headings* no módulo). Após vários testes em diferentes *browsers*, e ainda ponderando a possibilidade de o erro partir do código implementado, verificámos que o problema passava pelo leitor de ecrã usado nos testes locais, que apresentava algumas falhas. Este problema foi identificado quando o mesmo cenário foi testado pelos utilizadores invisuais, sendo que nesses casos os leitores de ecrã respondiam como esperado e programado. Dessa forma, ficou claro que o problema passava pelo leitor de ecrã usado na nossa parte (Narrator), sendo que, a partir daí, optámos por usar o NVDA - o mesmo usado pelos colaboradores invisuais (usando Windows). É importante reter que um dos colaboradores que outrora já tinha utilizado o Narrator também apontou algumas limitações, pelo que passou de igual forma a usar o NVDA.

Este impasse deixou-nos alerta para esta outra dimensão problemática, que revela que o problema nem sempre parte da estrutura implementada, podendo residir nas tecnologias de assistência dos utilizadores invisuais. Porém, dadas as dúvidas existentes, foi necessário confirmar com os utilizadores invisuais que conseguiam aceder a todos os *headings* definidos, inclusive o utilizador que fazia uso do MacOS com o leitor de ecrã voiceOver para garantir que não existia esse problema. É importante notar que esta diversidade de tecnologias, isto é, o facto de os utilizadores usarem sistemas operativos, *browsers* e leitores de ecrã diferentes, contribuiu para a uma compreensão mais ampla da acessibilidade. Assim, verificando que existem problemas específicos de cada tecnologia (e.g., problemas verificados num *browser*/leitor de ecrã que não existiam em mais nenhum dos restantes), fica claro que a implementação terá que ser robusta para atender a estas divergências.

Nenhum dos testes foi realizado de forma presencial com os utilizadores. Alguns destes

foram elaborados isoladamente pelos utilizadores sendo que, no final do teste, reportavam os problemas encontrados, bem como sugestões ou eventuais melhorias sentidas nas novas alterações. Porém, a maioria dos testes realizados foram mediados através de chamada de voz e partilha de ecrã. Tal aconteceu, por exemplo, em testes no módulo Pedido de Férias/Ausências, onde dois dos colaboradores invisuais nunca tinham conseguido efetuar determinadas ações sem ajuda de terceiros, pelo que se sentiram mais seguros realizando os testes com supervisão. Os testes, quando supervisionados, têm a vantagem da identificação mais rápida de problemas, assim como a deteção de erros que poderão passar despercebidos aos utilizadores quando os estão a realizar sozinhos.

Por fim, e de forma esporádica, era utilizada a extensão aXe, que permitia analisar a página em questão, apresentando um relatório de erros de acessibilidade. No entanto, esta ferramenta foi tida apenas como auxiliar. Isto porque, embora a extensão detete vários erros de acessibilidade, percebemos que não apresentava os erros considerados críticos - ou de maior importância - apontados pelos colaboradores invisuais. Desta forma, considerámos que não seria um método robusto para verificar os níveis de acessibilidade de forma regular.

### 7.2.3 Conformidade com as Diretrizes

No que diz respeito à conformidade com as diretrizes, para que uma página web seja acessível segundo as diretrizes WCAG 2.1, é necessário ir ao encontro de um dos níveis - A, AA ou AAA. Isto é, por exemplo, para a página ser acessível segundo o nível AA do documento WCAG 2.1, terá que responder a todos os critérios de sucesso A e AA apresentados. No entanto, estes critérios de sucesso não se cingem apenas às necessidades de utilizadores invisuais, mas também às de vários utilizadores com diferentes tipos de deficiências. Desta forma, torna-se difícil destacar os critérios que, por si só, garantam uma boa acessibilidade para utilizadores invisuais.

Consequentemente, para que os módulos aqui discutidos fossem dados como acessíveis segundo as diretrizes do WCAG 2.1., era necessário que estes fossem implementados tendo em vista uma acessibilidade generalizada (i.e., para qualquer utilizador, independentemente da sua deficiência). Para além de tal ser impraticável na janela temporal disponível, deslocava-se do âmbito do presente estágio.
Assim, embora as diretrizes tenham sido importantes - servindo também como auxilio para implementação da acessibilidade nos módulos -, não podemos afirmar que estes se encontram em conformidade total com o apontado pelo documento WCAG 2.1., pelas razões apontadas acima.

### 7.2.4 Testes de Usabilidade

Posteriormente à fase de implementação nos dois módulos - módulo Pedido de Férias/Ausências e Reportar Esforço -, foram realizados testes de usabilidade com os três colaboradores invisuais da empresa.
Não obstante ao facto de ter sempre existido *feedback* recorrente por parte destes colaboradores, os testes de usabilidade realizados após a fase de implementação visavam formalizar o processo de testes com os utilizadores finais e recolher algumas métricas quantitativas e qualitativas.

Durante todo o processo de implementação, os utilizadores finais (i.e., colaboradores invisuais da empresa) tiveram um papel-chave no desenvolvimento de soluções que fossem ao

encontro das suas necessidades. Porém, estes primeiros testes tiveram um carácter mais informal, na medida em que não eram definidas tarefas específicas para serem realizadas pelos colaboradores. Além disso, alguns dos testes não foram mediados e os utilizadores apenas reportavam o seu *feedback* posteriormente, consoante a sua disponibilidade.

Desta forma, os testes de usabilidade serviram de base para análise final de todas as implementações feitas ao longo do semestre. Com isto pretendíamos, em ambiente controlado, reunir dados que nos permitissem avaliar o impacto dos objetivos definidos para o presente projeto de estágio. De seguida, descrevemos a metodologia dos testes de usabilidade, assim como os resultados retirados dos mesmos.

#### 7.2.4.1 Metodologia dos Testes

Os testes de usabilidade tiveram como objetivo comparar a realização de tarefas nos dois módulos referidos, antes e depois de terem sido realizadas as alterações no âmbito da acessibilidade. Para isso, foram disponibilizados dois ambientes de teste aos três colaboradores invisuais, um com o código mais recente, com as alterações finais de acessibilidade, e outro com o código antigo (ainda sem qualquer alteração no âmbito da acessibilidade).

Os testes consistiram na realização de 5 tarefas em cada um dos módulos, realizadas na versão anterior e posterior às alterações de acessibilidade. Estas tarefas foram definidas tendo com base em ações comumente realizadas nos módulos, tentando também englobar algumas ações que se esperavam difíceis de realizar para os utilizadores invisuais. Assim, pretendíamos estudar o impacto das implementações de acessibilidade no desempenho dos colaboradores face a situações apontadas como críticas no início do semestre. Desta forma, foi possível retirar algumas métricas de comparação entre os dois módulos (na versão acessível e não acessível). As tarefas definidas para cada um dos módulos são apresentadas nas tabelas 7.2 e 7.3.

Dada a situação demográfica de cada um dos colaboradores invisuais e a situação pandémica vivida, os testes de usabilidade foram, forçosamente, realizados remotamente. Cada utilizador realizou os testes com as tecnologias que regularmente usam, no seu ambiente e no seu computador. A tabela 7.1 especifica as tecnologias utilizadas por cada um dos colaboradores invisuais durante os testes de usabilidade.
Os testes foram moderados remotamente - o que também já tinha acontecido em testes de desenvolvimento -, porém, os presentes testes foram gravados (som e ecrã), como forma de facilitar o processo de recolha de informação e análise. Para tal, foi utilizada a ferramenta Open Broadcaster Software (OBS) que permitiu gravar os testes de usabilidade.
Para que os testes fossem gravados, antes de dar início aos mesmos, foi questionado a cada um dos colaboradores se concordava que a sessão fosse gravada (áudio e ecrã) sob anonimato e apenas para fins académicos.

Aos utilizadores era dada a opção de, enquanto iam desenvolvendo as tarefas, irem reportando verbalmente aquilo que estavam a fazer, inclusive eventuais dificuldades que encontrassem durante a execução da tarefa. Caso assim preferissem, quando terminassem a tarefa, era-lhes pedido que descrevessem sucintamente o processo. Optámos por estabelecer estas duas opções, já que haveria a possibilidade de o ato de reportarem o processo enquanto a tarefa era realizada desviar a atenção do utilizador das informações fornecidas pelo leitor de ecrã, o que poderia comprometer a experiência do utilizador.

Neste caso específico de testes, foi transmitido aos participantes que, caso se encontrassem num impasse, poderiam pedir auxílio para prosseguirem a realização da tarefa. Esta decisão baseou-se no facto de algumas tarefas, nomeadamente na versão antiga da plataforma,

terem sido apontadas inicialmente como praticamente irrealizáveis. Foi ainda mencionado aos utilizadores que a qualquer momento poderiam desistir da realização da tarefa tendo em consideração que, na versão antiga dos módulos, existiam diversas barreiras de acessibilidade.

| **Utilizador** | **Sistema Operativo** | **Leitor de ecrã** | **Browser** |
|---|---|---|---|
| U1 | MacOS | VoiceOver | Safari |
| U2 | Windows | NVDA | Google Chrome |
| U3 | Windows | NVDA | Mozilla Firefox |

Tabela 7.1: Tecnologias Utilizadas

| # | **Tarefa** |
|---|---|
| 1 | Verificar quais os dias de férias marcados |
| 2 | Verificar quantos dias de férias foram pedidos e quantos foram atribuídos |
| 3 | Cancelar uma dia de ausência marcado no calendário |
| 4 | Pedir um novo dia de ausência (parcial) do tipo 'Doação de Sangue' |
| 5 | Verificar se existem eventos (férias ou ausências) no mês atual |

Tabela 7.2: Tarefas do Módulo Pedido de Férias/Ausências

| # | **Tarefa** |
|---|---|
| 1 | Reportar oito horas no dia atual |
| 2 | Verificar os dias com esforço parcial na semana atual |
| 3 | Verificar os dias do mês atual que não têm esforço reportado |
| 4 | Eliminar a tarefa 'Feriado' associada ao dia atual |
| 5 | Editar a duração de uma tarefa no dia atual |

Tabela 7.3: Tarefas do Módulo Reportar Esforço

#### 7.2.4.2 Resultados

Nesta subsecção apresentamos os dados recolhidos, resultantes das sessões dos testes de usabilidade. Adicionalmente, faremos também uma análise aos mesmos, com o objetivo de retirar o máximo de informação útil. De um modo geral, os resultados sugerem que as implementações de acessibilidade melhoraram a experiência dos utilizadores, facilitando o processo de interação com os módulos. A análise de resultados será dividida pelos dois módulos, verificando dados obtidos em cada um destes primeiramente, seguindo depois

para uma visão geral dos resultados obtidos.

Antes de prosseguirmos, existem alguns fatores que consideramos relevantes expor. Como veremos, em cada módulo, será feita uma análise geral dos dados, no sentido em que estes serão segmentados em diferentes tabelas com fim de clareza (tarefas concluídas, problemas identificados e dificuldade associada a cada tarefa). Após esta análise geral, foram destacados três dados cujo cálculo considerámos importantes, referentes à comparação do ambiente antes e depois das implementações de acessibilidade: Taxa de sucesso, Redução de problemas e Nível de dificuldade. Relembrando que foi dada a possibilidade aos participantes de comentarem as suas questões, dificuldades ou opiniões enquanto realizavam as tarefas, os problemas encontrados que aqui apresentamos foram aqueles que considerámos impactantes. Adicionalmente, alguns problemas foram apontados com base na nossa análise aos testes, mesmo que não tenham sido reportados pelos utilizadores.
Um quarto dado ponderado, que dizia respeito ao tempo de execução de cada tarefa, pretendia avaliar se as implementações de acessibilidade tiveram impacto a este nível. Embora tenhamos considerado este dado como sendo relevante, acabámos por descartar esta análise, uma vez que foram várias as tarefas (no ambiente antigo, sem acessibilidade) que não foram concluídas ou sequer realizadas, dado o nível de dificuldade inerente às mesmas. Adicionalmente, uma vez que um dos utilizadores não realizou os testes na versão antiga (como explicamos adiante), não teríamos forma de comparar o tempo de realização de tarefas desse utilizador. Assim, este quarto dado acabou por ficar de fora desta análise, embora tenha sido registado.

A duração de cada teste foi outro fator discutido, na medida em que, em média, cada teste levou cerca de uma hora e trinta minutos. Dado que tal seria potencialmente exaustivo para os participantes, foi ponderado separar os testes em duas partes distintas (e em dias diferentes): Ambiente antigo e, noutra altura, ambiente novo. Contudo, e tendo os participantes mostrado disponibilidade para realizar ambos os testes de uma vez apenas, foi decidido realizar um único teste.
Por fim, esclarecemos antecipadamente a ausência de registo do teste de um dos três utilizadores, relativo ao antigo ambiente. Os testes que aqui apresentamos, realizados no mês de junho, acompanharam uma situação atípica, forçada pela pandemia atual, que obrigou a realização dos testes a partir do domicílio. Como descrito, o código antigo foi recuperado e alocado em ambiente de teste, acessível apenas pela VPN da empresa, o que apenas era possível através do computador fornecido pela mesma. Acontece que um dos colaboradores invisuais não teve possibilidade de o ter consigo aquando da realização dos testes, pelo que apenas tinha acesso à versão de produção (i.e., a versão final). Por esta razão, apenas dois dos três participantes tiveram acesso ao ambiente antigo, testando o mesmo.

A tabela apresentada acima (7.4), apresenta as tarefas concluídas por cada utilizador no módulo Pedidos de Férias/Ausências, sendo possível comparar a prestação na versão antiga e recente (salvo o utilizador U3, pelas questões expostas anteriormente). Relembramos que as tarefas pedidas, apresentadas na secção anterior, podem ser consultadas na tabela 7.2. O primeiro utilizador concluíu três das cinco tarefas pedidas na antiga versão, conseguindo realizar as mesmas tarefas no ambiente recente. Já o segundo utilizador, apenas concluíu uma tarefa no ambiente com a versão antiga. À semelhança do que se verificou com o primeiro utilizador, tanto o segundo como o terceiro conseguiram realizar com sucesso todas as tarefas no ambiente recente. Olhando para estes dados como um todo, observamos que a taxa de sucesso na versão antiga, considerando as tarefas em questão, é de 40% (4 concluídas em 10 possíveis), sendo que a versão recente do ambiente aponta uma taxa de sucesso de 100% (15 tarefas concluídas em 15 possíveis).

| Utilizador | Versão | Tarefa#1 | Tarefa#2 | Tarefa#3 | Tarefa#4 | Tarefa#5 |
|---|---|---|---|---|---|---|
| U1 | Antiga | Não | Sim | Sim | Sim | Não |
| U1 | Recente | Sim | Sim | Sim | Sim | Sim |
| U2 | Antiga | Não | Sim | Não | Não | Não |
| U2 | Recente | Sim | Sim | Sim | Sim | Sim |
| U3 | Antiga | N/A | N/A | N/A | N/A | N/A |
| U3 | Recente | Sim | Sim | Sim | Sim | Sim |

Tabela 7.4: Tarefas Concluídas (Módulo Pedido de Férias/Ausências)

Por questões de organização, os problemas identificados foram divididos em 3 tabelas (por utilizador). Relembrando, o utilizador 3 (U3) não realizou o teste na versão antiga e, por essa razão, não existem dificuldades reportadas.

Ao observarmos a tabela referente ao utilizador U1 (Figura 7.5), verificamos que as dificuldades apresentadas pelo utilizador na versão antiga deixaram de existir após a implementação das alterações de acessibilidade. Importa notar que embora tenham sido detetadas dificuldades durante a realização de todas as tarefas (na versão antiga), apenas as dificuldades associadas à codificação de informação por cores no calendário (tarefa #1 e #5) impediram a realização das tarefas.

Relativamente aos testes do utilizador U2, podemos verificar que foram identificados 8 problemas no total durante a realização das tarefas na versão antiga. Ainda, observando os dados da versão recente, podemos notar que também na versão recente foram identificados novos problemas. Os problemas apresentados diziam respeito à falta de intuição das opções apresentadas na janela *popup* - após ter pressionado o botão para cancelar o dia marcado de férias ou após preencher os campos necessários para o pedido de um novo dia de ausência, o utilizador não sabia qual seria o próximo passo a tomar. No entanto, estes problemas - embora prejudicassem a experiência do utilizador - foram classificados como problemas de usabilidade já existentes na aplicação.

No que diz respeito aos problemas que impediram a execução das tarefas, nas tarefas #1 e #5 o utilizador não realizou as tarefas, fruto da codificação de cores no calendário. Ainda, no caso das tarefas #3 e #4 (as quais também não foram concluídas na versão antiga), as dificuldades que impediram o utilizador de realizar as mesmas estavam relacionadas com o facto deste não conseguir identificar qual o mês em questão (apenas recebia *feedback* do dia do mês) quando navegava pelos dias do calendário. Para além disso, o utilizador referiu que não conseguia mudar para outro mês. Por estas razões, o utilizador desistiu da execução das tarefas, sendo que o facto de não saber o mês em questão também foi igualmente apontado como uma dificuldade que o impedia de realizar a tarefa #5.

Por fim, ainda dentro dos dificuldades associadas ao módulo Pedido de Férias/Ausências, podemos observar os resultados dos testes de usabilidade do utilizador U3. Uma vez mais, verificamos que na versão recente não foram encontradas dificuldades na realização das tarefas #1, #2 e #5. Porém, nas tarefas #3 e #4, o utilizador apresentou também dificuldades associadas à falta de intuição das opções disponibilizadas na janela *popup* (à semelhança do que foi identificado com o utilizador U2).

Concluíndo a análise das tabelas, embora o tratamento de problemas relacionados com a usabilidade da aplicação (mais concretamente, dos módulos tratados) ultrapasse o âmbito do presente projeto, considerámos importante listar os mesmos por duas razões. Em

| **Tarefa** | **Versão antiga** | **Versão Recente** |
|---|---|---|
| **#1** | • Só conseguia consultar a informação pedida se verificasse dia-a-dia (processo moroso). | Nenhum |
| #2 | • Apenas realizou a tarefa rapidamente porque já estava familiarizado com o módulo, "saltando"mais rápido para a informação pedida utilizando um método de pesquisa. Porém, só o conseguia fazer por já conhecer o módulo. | Nenhum |
| #3 | • O foco não se moveu automaticamente para a janela;<br>• Ao pressionar o botão para cancelar o dia selecionado, o foco saiu da *dialog*;<br>• Após realizar concluir a tarefa, não sabia como confirmar. | Nenhum |
| #4 | • O foco não se moveu automaticamente para a janela;<br>• Ao pressionar o botão para cancelar o dia, o foco saiu da janela *popup*;<br>• Quando navega no calendário não consegue identificar o ano;<br>• Não recebe *feedback* das opções do *dropdown*. | Nenhum |
| #5 | • Só conseguia consultar a informação pedida se verificasse dia-a-dia (processo moroso). | Nenhum |

Tabela 7.5: Problemas encontrados pelo utilizador 1 (Módulo Pedido de Férias/Ausências)

primeiro lugar, tendo sido notas apontadas pelos participantes e, tendo sido notório o impacto dos mesmos na experiência dos utilizadores, optámos por registar as mesmas, independentemente das causas em questão. Em segundo lugar, considerámos interessante reter o facto de que após as dificuldades de acessibilidade serem suprimidas, verificarmos que emergem certos problemas relacionados com a usabilidade do sistema. A nosso ver, tal suporta o facto de que bons níveis de acessibilidade, por si só, não são suficientes para uma experiência exímia - tal como apontámos no capítulo 3 (subsecção 3.1) deste documento.

No decorrer dos testes, sempre que cada tarefa era dada por concluída - com ou sem sucesso -, era pedido a cada participante que registasse a dificuldade dessa mesma tarefa. Para tal, estabelecemos uma escala de *Likert*, sendo que 1 correspondia a *muito difícil*, 2 a *dfícil*, 3 a *nem fácil nem difícil*, 4 a *fácil* e 5 a *muito fácil*. Com isto, pretendíamos ter mais um dado para estudar o impacto das implementações de acessibilidade realizadas neste segundo semestre. A tabela 7.8 apresenta os dados recolhidos nos testes do presente módulo.

Como observamos, e pelas razões já apresentadas, o participante 3 (U3), não realizou testes à versão antiga, pelo que apenas temos dados relativos à versão mais recente do módulo. Embora não seja possível estabelecer uma comparação entre versões, percebemos que as tarefas realizadas após as implementações de acessibilidade foram cotadas entre 4 e 5, ou seja, foram apontadas como fáceis de realizar. Olhando para os valores indicados pelos restantes utilizadores (U1 e U2), verificamos que, de um modo geral, a versão antiga apresentou dificuldades consideráveis, enquanto que as mesmas tarefas realizadas na versão recente foram avaliadas como mais fáceis - algumas de forma significativa. As exceções observam-se nas tarefas 3 e 4 realizadas pelo utilizador 2 (U2). Porém, importa notar que o utilizador associou a dificuldade destas tarefas à falta de intuição das opções apresentadas

| Tarefa | Versão antiga | Versão Recente |
|---|---|---|
| #1 | • Não tem acesso à informação pedida uma vez que a mesma é apresentada através de cores. | Nenhum |
| #2 | Nenhum | Nenhum |
| #3 | • A janela do módulo não recebe foco quando abre;<br>• Não consegue identificar o mês quando está a navegar pelos dias do calendário;<br>• Não consegue navegar entre meses. | • Após selecionar o botão para cancelar o dia de férias, não percebeu qual seria a opção que devia escolher. |
| #4 | • Não consegue identificar o mês quando está a navegar pelos dias do calendário;<br>• Não consegue navegar entre meses. | • Após preencher os campos para adicionar a ausência, apresentou dúvidas a concluir a tarefa. |
| #5 | • Não consegue identificar o mês quando está a navegar pelos dias do calendário;<br>• Não tem acesso à informação por esta ser apresentada por cores. | Nenhum |

Tabela 7.6: Problemas identificados pelo utilizador 2 (Módulo Pedido de Férias/Ausências)

(i.e., problema de usabilidade da aplicação). A tarefa 2 foi considerada muito fácil (5) em ambas as versões. Isto verificou-se dado que o colaborador invisual tirou partido da navegação através das setas, sendo que no caso do utilizador 1 (U1), a navegação era feita através de pesquisa. Uma vez que as implementações levadas a cabo facilitaram a navegação no módulo (através do definição dos *headings*), e que a navegação através das setas não apresentavam problemas, apenas o primeiro utilizador tirou partido dessa melhoria. Já no caso da tarefa 3 e 4, na qual o utilizador 2 (U2) apontou melhorias pouco significativas, tal é justificado pelos problemas emergentes relacionados com a usabilidade, uma vez que as opções disponibilizadas não eram intuitivas, como discutido. Assim, de um modo geral, observamos que as implementações de acessibilidade facilitaram o processo de interação com o módulo em questão.

Olhemos agora para a prestação dos utilizadores no módulo Reportar Esforço, comparando o número de tarefas concluídas. Analisando a tabela 7.9, à semelhança do módulo anterior, o ambiente que apresentava as novas implementações de acessibilidade revelam uma taxa de sucesso de 100% (15 tarefas em 15 possíveis). Comparando com o número de tarefas concluídas no ambiente antigo, observamos melhorias, já que neste foram concluídas sete tarefas em dez possíveis.

Assim como no módulo anterior, apresentamos as tabelas com os registos das dificuldades encontradas durante os testes ao módulo Reportar Esforço, para cada um dos utilizaores, tanto na versão antiga como na recente. Observando a tabela 7.10, referente aos testes do utilizador U1 nas versões antiga e recente, respetivamente, verificamos que a maioria dos problemas de acessibilidade foram resolvidos na última. Porém, houve dois problemas que permaneceram e que dizem respeito à falta de *feedback*, relativo ao sucesso ao concluir a tarefa de eliminar ou editar uma tarefa. Importa referir que estes problemas foram prontamente resolvidos após os testes de usabilidade. Observamos agora a tabela 7.11 que descrevem as dificuldades encontradas pelo utilizador 2 (U2). Novamente, na versão

| **Tarefa** | **Versão antiga** | **Versão Recente** |
|---|---|---|
| #1 | N/A | Nenhum |
| #2 | N/A | Nenhum |
| #3 | N/A | • A ação a tomar após pressionar o botão de cancelar o dia não é intuitiva. |
| #4 | N/A | • A ação a tomar após preencher os campos de ausência não é intuitiva. |
| #5 | N/A | Nenhum |

Tabela 7.7: Problemas identificados pelo utilizador 3 (Módulo Pedido de Férias/Ausências)

| **Utilizador** | **Versão** | **Tarefa#1** | **Tarefa#2** | **Tarefa#3** | **Tarefa#4** | **Tarefa#5** |
|---|---|---|---|---|---|---|
| U1 | Antiga | 1 | 2 | 1 | 1 | 1 |
| U1 | Recente | 4 | 4 | 5 | 5 | 5 |
| U2 | Antiga | 1 | 5 | 1 | 1 | 1 |
| U2 | Recente | 5 | 5 | 2 | 3 | 5 |
| U3 | Antiga | N/A | N/A | N/A | N/A | N/A |
| U3 | Recente | 5 | 4 | 4 | 4 | 4 |

Tabela 7.8: Dificuldade associada a cada tarefa (Módulo Pedido de Férias/Ausências)

recente foram detetadas as mesmas dificuldades que tinham sido descritas acima, referentes à falta de *feedback* após eliminar ou editar uma tarefa. Por último, no caso do utilizador U3 (tabela 7.12) - que apenas realizou testes na versão nova -, os problemas identificados estavam também relacionados com a ausência de *feedback* após a realização de ações.

Importa ainda notar que as dificuldades que levaram os utilizadores U1 e U2 a não concluir algumas das tarefas na versão antiga (Tarefa #3 e Tarefas #2 e #3, respetivamente), estavam associadas ao facto da informação no calendário estar apenas codificada por cores, assim como aconteceu no módulo anterior. Como tal, o processo de consulta por parte dos utilizadores invisuais no calendário tornava-se um processo moroso, que levava os utilizadores a desistir deste tipo de tarefas.

| **Utilizador** | **Versão** | **Tarefa#1** | **Tarefa#2** | **Tarefa#3** | **Tarefa#4** | **Tarefa#5** |
|---|---|---|---|---|---|---|
| U1 | Antiga | 3 | 2 | 1 | 3 | 3 |
| U1 | Recente | 5 | 5 | 5 | 4 | 4 |
| U2 | Antiga | 3 | 1 | 1 | 2 | 2 |
| U2 | Recente | 5 | 5 | 5 | 5 | 5 |
| U3 | Antiga | N/A | N/A | N/A | N/A | N/A |
| U3 | Recente | 5 | 4 | 4 | 5 | 4 |

Tabela 7.13: Dificuldade associada a cada tarefa (Módulo Reportar Esforço)

| Utilizador | Versão | Tarefa#1 | Tarefa#2 | Tarefa#3 | Tarefa#4 | Tarefa#5 |
|---|---|---|---|---|---|---|
| U1 | Antiga | Sim | Sim | Não | Sim | Sim |
| U1 | Recente | Sim | Sim | Sim | Sim | Sim |
| U2 | Antiga | Sim | Não | Não | Sim | Sim |
| U2 | Recente | Sim | Sim | Sim | Sim | Sim |
| U3 | Antiga | N/A | N/A | N/A | N/A | N/A |
| U3 | Recente | Sim | Sim | Sim | Sim | Sim |

Tabela 7.9: Tarefas Concluídas (Módulo Reportar Esforço)

| Tarefa | Versão antiga | Versão Recente |
|---|---|---|
| #1 | • Não se lembrava qual o termo que usava para saltar para o painel lateral. | Nenhum |
| #2 | • Ter que realizar a verificação dia-a-dia. | Nenhum |
| #3 | • Ter que realizar a verificação dia-a-dia. | Nenhum |
| #4 | • Não recebeu *feedback* ao eliminar a tarefa. | • Não recebeu *feedback* ao eliminar a tarefa. |
| #5 | • Não recebeu *feedback* ao editar a tarefa. | • Não recebeu *feedback* ao editar a tarefa. |

Tabela 7.10: Problemas identificados pelo utilizador 1 (Módulo Reportar Esforço)

Por fim, analisamos os valores atribuídos à dificuldade associada às diferentes tarefas em ambas as versões (Tabela 7.13). À semelhança do que foi feito no módulo anterior, os utilizadores utilizaram a mesma escala para avaliar as tarefas antes e depois das implementações de acessibilidade. Tal como se verificou anteriormente, embora o terceiro participante (U3) não tenha tido possibilidade de testar a versão antiga - impedindo assim uma comparação direta -, verificamos que a versão recente não apresentou dificuldades, de um modo geral. O mesmo é observável nos valores atribuídos pelos restantes participantes, havendo sempre melhorias a nível de dificuldade.

### 7.2.5 Análise

Após a realização dos testes, e tendo corrigido os últimos problemas reportados, estamos em posição para tirar algumas conclusões que consideramos importantes de apontar. Para além das questões tratadas neste capítulo, observámos ao longo dos testes realizados com os utilizadores finais (durante o desenvolvimento e os testes de usabilidade), contratempos recorrentes na interação com elementos *dialog*. Estes contratempos eram fruto de um comportamento irregular do foco neste tipo de elementos. No contexto do presente estágio, para os colaboradores invisuais, estes elementos revelaram ser de particular dificuldade interativa, sendo um problema a nível de acessibilidade. Isto porque, mesmo que sejam tidos em conta os comportamentos esperados numa *dialog* descritos no documento WAI Authoring Practices [35], a interação com elementos presentes nas *dialogs* (fazendo uso do teclado) pode levar a que o foco saia das mesmas. Isto, logicamente, é um comportamento irregular, e que leva os utilizadores invisuais a procurar novamente a *dialog*, de forma

| **Tarefa** | **Versão antiga** | **Versão Recente** |
|---|---|---|
| **#1** | • Ter que percorrer todos os dias até ao dia atual. | Nenhum |
| #2 | • Não existe um método rápido para consultar a informação. | Nenhum |
| #3 | • Não consegue ter acesso à informação pedida. | Nenhuma |
| #4 | • Ao selecionar o botão para apagar a tarefa o foco saiu do módulo;<br>• Não recebeu *feedback* ao eliminar a tarefa. | • Não recebeu *feedback* ao eliminar a tarefa. |
| #5 | • O foco saiu do módulo após a interação com elementos;<br>• Não recebeu *feedback* ao editar a tarefa. | • Não recebeu *feedback* ao editar a tarefa; |

Tabela 7.11: Problemas identificados pelo utilizador 2 (Módulo Reportar Esforço)

| **Tarefa** | **Versão antiga** | **Versão Recente** |
|---|---|---|
| #1 | N/A | Nenhum |
| #2 | N/A | Nenhum |
| #3 | N/A | Nenhum |
| #4 | N/A | • Não recebeu *feedback* ao eliminar a tarefa; |
| #5 | N/A | • Não recebeu *feedback* ao editar a tarefa; |

Tabela 7.12: Problemas identificados pelo utilizador 3 (Módulo Reportar Esforço)

a mover o foco para a mesma. Assim, de modo a colmatar este problema, aquando da implementação de *dialogs*, terá sempre de ser verificado o comportamento do foco após a interação com todos os elementos presentes na mesma e, caso seja necessário, programar o foco de forma a que este se mantenha no mesmo ou num outro elemento pertencente à *dialog*.

Embora nos testes de usabilidade com os dois utilizadores (U1 e U2) nas versões antigas tenham apresentado alguns problemas com o foco - tornando a realização da tarefa morosa -, tal não impediu os utilizadores de concluirem a tarefa. No entanto, importa referir que tal se verificou dado que os utilizadores invisuais estavam familiarizados com estes comportamentos erróneos e que, de certa forma, tinham já métodos estabelecidos para contornar o problema. Caso se tratasse de um utilizador invisual com menos experiente (a nível da aplicação ou do próprio leitor de ecrã), os problemas poderiam prejudicar a interação utilizador/aplicação. Ainda, verificaram-se algumas situações nas quais os utilizadores pressionavam uma tecla desprepositadamente, levando a que o foco saisse da *dialog*. Porém, este é um tipo de comportamento que não é praticável prevenir. Como tal, apontamos que, sempre que possível, a utilização de *dialogs* deverá ser evitada, já que parte considerável do tempo de testes foi maioritariamente consumida por problemas relacionados com *dialogs*. Desta forma, consideramos que estas apenas deveriam ser utilizadas para situações mais pontuais, como apresentação de alertas que exigem a atenção imeadiata do utilizador. No entanto, dado que diversos módulos aqui tratados são construídos por *dialogs*, destacamos esta questão como consideração futura, como meio de criar módulos mais acessíveis a utilizadores invisuais. Assim, reiteramos a ideia de que a acessibilidade deverá ser tida em conta aquando do desenho da aplicação. Tratada paralelamente, não só

se atribui o mesmo nível de importância à implementação da acessibilidade, como poderá prevenir, ou facilitar, eventuais alterações futuras.

Outro fator que representou grandes dificuldades e que levou a que várias tarefas não fossem concluídas, dizia respeito à codificação de informação por cores. Embora a utilização de cores para facultar informação aos utilizadores visuais possa ser eficaz, as cores não devem ser a única forma de transmitir informação. Isto porque, como observado ao longo de todo o processo de testes, uma vez que não tinham perceção das cores, as tarefas tornavam-se bastante morosas e, por essa razão, não realizavam essas ações autonomamente no dia-a-dia. Como descrito pelos próprios, para realizarem as tarefas, os colaboradores invisuais recorriam a métodos alternativos como apontar a informação em ficheiros de texto ou pedir ajuda a terceiros. Embora tenhamos, neste projeto, resolvido este problema com sucesso, enfatizamos a noção de que esta questão deverá ser cuidada desde início.

Destacamos ainda o papel facilitador dos *headings* relativamente à navegação. Como vimos, desde cedo pecebemos que os utilizadores invisuais tiram partido dos *headings*, facilitando consideravelmente a sua navegação pelos módulos. Para além disso, permite-lhes adquirir uma noção geral da página e sua estrutura. Em testes iniciais, um dos colaboradores invisuais afirmou que desconhecia que determinada informação existia no módulo, até os *headings* terem sido definidos. Por isto, acreditamos que o cuidado relativamente aos *headings* é uma boa prática, no sentido em que ajudou a promover uma boa experiência aos utilizadores invisuais, nomeadamente na definição de títulos dos meses dos calendários, que permitiu uma navegação mais rápida e eficiente.
Adicionalmente, relembramos a importância do cuidado perante os nomes dos elementos interativos das páginas e, ainda, a relevância das notificações para utilizadores invisuais que, mais do que os utilizadores visuais, dependem das mesmas para se sentirem seguros aquando da interação com as páginas.
Por último, importa enfatizar o facto de que os utilizadores invisuais U2 e U3, que não utilizavam o módulo Pedido de Férias/Ausências sem a ajuda de terceiros, conseguiram realizar todas as tarefas de forma praticamente autónoma - salvo problemas de usabilidade. No fim, após testarem as implementações de acessibilidade no módulo, afirmaram que se sentiam mais seguros ao usá-lo.

This page is intentionally left blank.

# Capítulo 8

# Discussão

Apresentados os resultados dos testes de usabilidade e respetiva análise, considerámos relevante estabelecer um capítulo como base para uma reflexão sobre os mesmos, fazendo ainda um balanço do trabalho realizado ao longo deste ano letivo e considerações sobre trabalho futuro. Os resultados apresentados no capítulo precedente apontam melhorias na experiência interativa dos utilizadores invisuais com os módulos Pedido de Férias/Ausências e Reportar Esforço, sugerindo que as implementações de acessibilidade representaram um impacto positivo relativamente à versão antiga . Para além disso, olhando para os objetivos principais aos quais nos propusemos, podemos afirmar que foram alcançados e todos os casos de uso associados a cada módulo foram implementados. No entanto, é importante salientar algumas questões, clarificando o que poderá ser visto como um funcionamento potencialmente exímio dos módulos em questão (ou qualquer outro módulo de difícil interação por parte de utilizadores invisuais).

Como sabemos, as diretrizes para uma boa acessibilidade das plataformas é um universo vasto, estendendo-se a diversos tipos de deficiência. Estando o nosso objetivo focado especificamente em melhorar a experiência de utilizadores invisuais, não nos é possível afirmar categoricamente que os módulos tratados assumem níveis de excelência no que à acessibilidade diz respeito. O que podemos afirmar, é que os dois módulos tratados neste projeto de tese respondem, atualmente, às necessidades dos colaboradores invisuais da Critical Software. Logicamente, é preciso ter em conta que alterações futuras (como atualizações) consideráveis nas plataformas poderão, eventualmente, perturbar o bom funcionamento da acessibilidade implementada. Com isto, pretendemos reforçar a ideia de que a acessibilidade (e ainda a usabilidade) devem ser processos indispensáveis, paralelos à restante implementação, que não devem ser descurados. Adicionalmente, é preciso ter em conta a singularidade de cada utilizador. Como observámos, nem todos os utilizadores realizam as mesmas tarefas através dos mesmos métodos, pelo que a facilidade que um determinado utilizador tem em realizar uma tarefa não significa necessariamente que o mesmo se verifique nos restantes. Isto verificou-se em algumas tarefas aquando dos testes de usabilidade, onde alguns níveis de dificuldade atribuídos à mesma tarefa variaram consoante o utilizador.

Assim, a acessibilidade deverá ser transversal a todo o funcionamento das aplicações, cobrindo todos os caminhos possíveis de uma determinada ação. Por fim, sublinhamos ainda a relevância da usabilidade na aplicação como um todo. Durante o processo de implementação, garantimos que todas as alterações feitas no âmbito da acessibilidade eram, também, usáveis. No entanto, embora ultrapasse o âmbito do presente projeto, reforçamos que toda a plataforma deverá responder com bons níveis de usabilidade, como complemento neces-

sário à acessibilidade. Estes fatores são, a nosso ver, relevantes e necessários ter em conta em trabalho futuro.

Refletindo sobre todo o percurso realizado ao longo deste projeto, consideramos que todos os passos tomados contribuíram de forma positiva para o resultado final que apresentámos, apesar de contratempos inevitáveis. Tendo em consideração que cada fase de implementação era realizada em código existente, assumimos desde cedo com cuidados redobrados com todas as alterações efetuadas. Desta forma, foi fundamental dispender tempo com o fim de compreender todas as componentes do código que, sendo base de uma aplicação tão diversa como o PULSAR, verificava alguma complexidade a nível de estrutura. Ainda, a necessidade de estudar linguagens com as quais tínhamos pouco contacto prévio, nomeadamente HTML, SASS e Typescript; ou até Angular, tecnologia com a qual nunca tínhamos trabalhado. Naturalmente, tudo isto consumiu tempo, contudo, consideramos que passámos por uma aprendizagem, de certa forma, exponencial, na medida em que cumprimos os objetivos principais no tempo de estágio previsto.

O primeiro semestre foi parte importante de todo o processo, na medida em que se revelou uma boa base para os restantes meses de trabalho. Embora o trabalho prático para o projeto em si tenha sido, em certa medida, escasso, as tarefas realizadas no primeiro semestre contribuíram para a familiarização com a plataforma, métodos de trabalho e contacto com a equipa do PULSAR. A primeira tarefa realizada no PULSAR (descrita no capítulo 4), assumiu um certo nível de dificuldade, no entanto, revelarou-se vantajosa na medida em que permitiram adquirir experiência numa linguagem nunca antes trabalhada. Adicionalmente, é importante salientar que paralelamente foram sendo reunidos, e estudados, diversos materiais referentes à acessibilidade que, sendo também um campo completamente novo e nunca explorado, foram cruciais para uma melhor preparação: as diretrizes (WCAG) e especificações técnicas fornecidas pelo WAI (WAI-ARIA) e o documento WAI Authoring Practices que serve como auxílio , assim como diversos artigos que nos elucidaram sobre a importância da acessibilidade e erros comuns, foram fonte principal de informação.

Tudo isto permitiu-nos desenhar os casos de uso de forma mais clara, ajudando-nos a compreender melhor as dificuldades apresentadas pelos colaboradores invisuais num primeiro levantamento e definir as abordagens aos problemas. A boa definição de requisitos, bem como o seu planeamento, é fundamental para a estruturação da fase de implementação. Tendo este ideal presente desde início, salientamos a influência dos colaboradores invisuais (utilizadores finais) neste processo inicial, cuja disponibilidade foi importante para que tal se verificasse. Adicionalmente, assinalamos também a disponibilidade e presença destes utilizadores ao longo do segundo semestre, permitindo suprimir erros ou inconsistências na implementação no imediato, abrindo espaço para outras melhorias.

Com os objetivos principais alcançados, os testes de usabilidade permitiram-nos aferir o impacto destes mesmos objetivos através da comparação prática entre os dois módulos preexistentes (que eram usados antes das implementações de acessibilidade levadas a cabo pelo presente projeto) e os módulos recentes, após conclusão da fase de implementação. Os resultados dos testes revelaram melhorias a nível de experiência e de satisfação dos colaboradores invisuais, sugerindo que os restantes módulos da aplicação PULSAR que não foram abordados neste projeto, poderiam proporcionar uma melhor experiência aos referidos colaboradores através de melhorias de acessibilidade. Assim, embora boas práticas de acessibilidade, por si só, não sejam suficientes para melhorar totalmente a experiência, acreditamos que para além dos dois módulos tratados neste projeto (os mais usados), um melhoramento da acessibilidade no PULSAR, em todo o seu domínio, seria vantajoso. Para além disso, estes testes finais permitiram colmatar pequenos erros que nos tinham passado despercebidos, reforçando, desta forma, o melhoramento dos módulos. Numa nota sobre os

objetivos secundários definidos, tais não foram implementados por restrições temporais - como sublinhámos inicialmente, estes estariam sempre dependentes da implementação dos objetivos principais. Desta forma, tendo em conta que toda a fase de testes se revelou parte essencial no processo de desenvolvimento, não considerámos viável abordar os objetivos secundários definidos numa janela temporal tão curta.

Concluída a fase de testes, e antes do término do período de estágio, elaborámos ainda um pequeno guia de acessibilidade para a equipa PULSAR, a pedido da mesma. Este contém descrições sobre o processo de implementação de acessibilidade levado a cabo neste projeto, com o objetivo de poder ser consultado mais tarde pela equipa. Assim, e tendo por princípio os conhecimentos que adquirimos neste ano, este guia pretende servir de base para os restantes colaboradores, tido como um bom ponto de partida para a implementação de acessibilidade nos restantes módulos da aplicação PULSAR. Escrito na ferramenta Confluence [10] (Tecnologias Usadas 2.4), podendo assim ser consultado pela equipa, este documento retém ainda aquelas que consideramos serem boas práticas de acessibilidade, relevantes no contexto da empresa. Com isto, pretendemos ainda salvaguardar as implementações realizadas, sendo que eventuais atualizações futuras poderiam comprometer estes avanços, já que grande parte do que foi implementado não é visível nas páginas.

Assim, considerando o trabalho realizado neste projeto, fazemos um balanço positivo do mesmo, acreditando que para além de objetivos alcançados, o trabalho conduzido produziu melhorias nos módulos estudados. De qualquer forma, salientamos o constante apoio técnico providenciado pela equipa PULSAR, assim como a disponibilidade dos colaboradores invisuais da empresa Critical Software. Por fim, retiramos como positiva esta primeira experiência em ambiente empresarial, terminando este projeto com noções muito claras sobre a importância de plataformas acessíveis a qualquer utilizador.

This page is intentionally left blank.

# Capítulo 9

# Plano de Trabalho

## 9.1 Primeiro Semestre

O diagrama de Gantt apresentado na Figura 9.1 descreve o trabalho realizado ao longo do primeiro semestre. Tendo o estágio tido início no dia 23 de setembro (16 horas semanais), as primeiras quatro semanas serviram de contextualização com as tecnologias usadas na empresa, sendo que as quatro semanas seguintes foram destacadas para uma familiarização prática com as tecnologias por via da implementação de uma tarefa. Seguiram-se cinco semanas de pesquisa bibliográfica, coincidindo com a já mencionada apresentação de equipa. Por fim, a elaboração do presente documento levou cinco semanas, sendo que os casos de uso foram definidos dentro deste período.

## 9.2 Segundo Semestre

O diagrama de Gantt apresentado na Figura 9.2 descreve o plano de trabalho definido para o segundo semestre, agora atualizado com o esforço real.

No segundo semestre, na segunda semana de fevereiro, demos início à fase de implementação da acessibilidade no primeiro módulo - Pedido de Férias/Ausências. Como previsto, esta teve a duração de oito semanas, com os testes paralelos a começarem uma semana mais tarde. Uma vez que todos os testes estavam dependentes da disponibilidade dos colaboradores invisuais, iniciámos a implementação do módulo Reportar Esforço (na figura, Módulo 2) antes de terminarmos os testes do primeiro módulo, até reunirmos o *feedback* desejado.
Como previmos no primeiro semestre, as implementações de acessibilidade no segundo módulo demoraram menos tempo que as primeiras implementações. Isto foi fruto da familiarização com as tecnologias, metodologias de equipa e o próprio conhecimento da acessibilidade no seu todo. Ainda, algumas implementações foram semelhantes às realizadas no módulo Pedido de Férias/Ausências, pelo que tal foi concluído em quatro semanas (em seis previstas). Por sua vez, os testes relativos a este módulo levaram cinco semanas, como previsto.
Embora estes testes, paralelos à implementação, tenham verificado também ajustes de acordo com o *feedback* dado, houve ainda seis semanas de Refinamento. Esta fase permitiu realizar algumas alterações aos módulos com base nos últimos testes realizados durante o processo de desenvolvimento (após ter sido dada por concluída a implementação dos mesmos). Complementarmente, também nos Testes de Usabilidade (os quais não tinham sido

ponderados no semestre anterior) foram identificadas algumas lacunas que foram tratadas neste período.

Certificando que as melhorias a nível de acessibilidade estavam de acordo com as necessidades dos colaboradores finais, os últimos dias ainda deram lugar à elaboração do Guia de Acessibilidade pedido pela equipa, não estando destacado no diagrama em questão.

Por fim, a elaboração final do presente documento durou mais duas semanas do que previsto, sendo que a sua realização foi adiada para os últimos meses, dando primasia ao tempo útil de estágio.

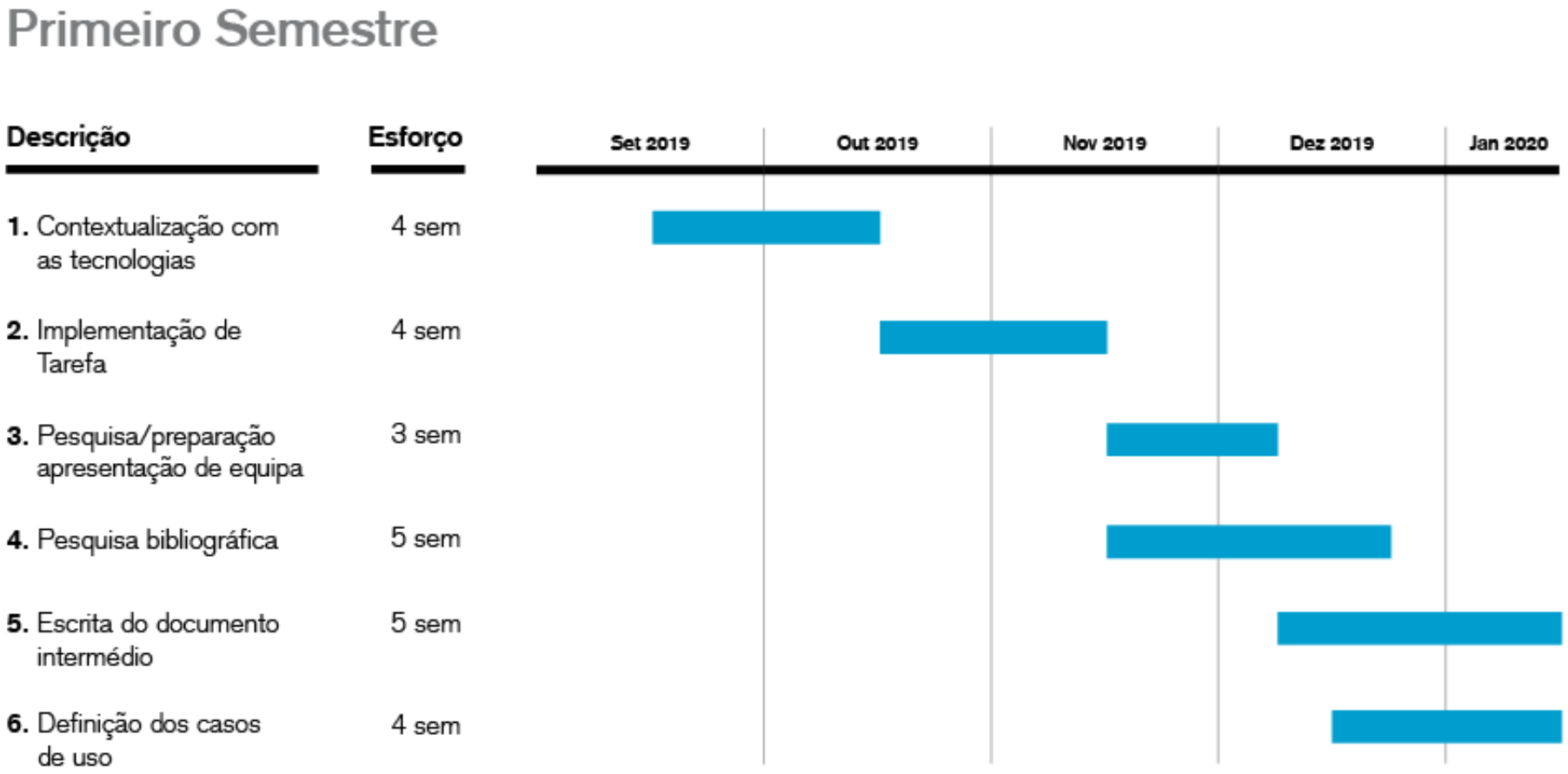


Figura 9.1: Diagrama de Gantt: Primeiro Semestre.

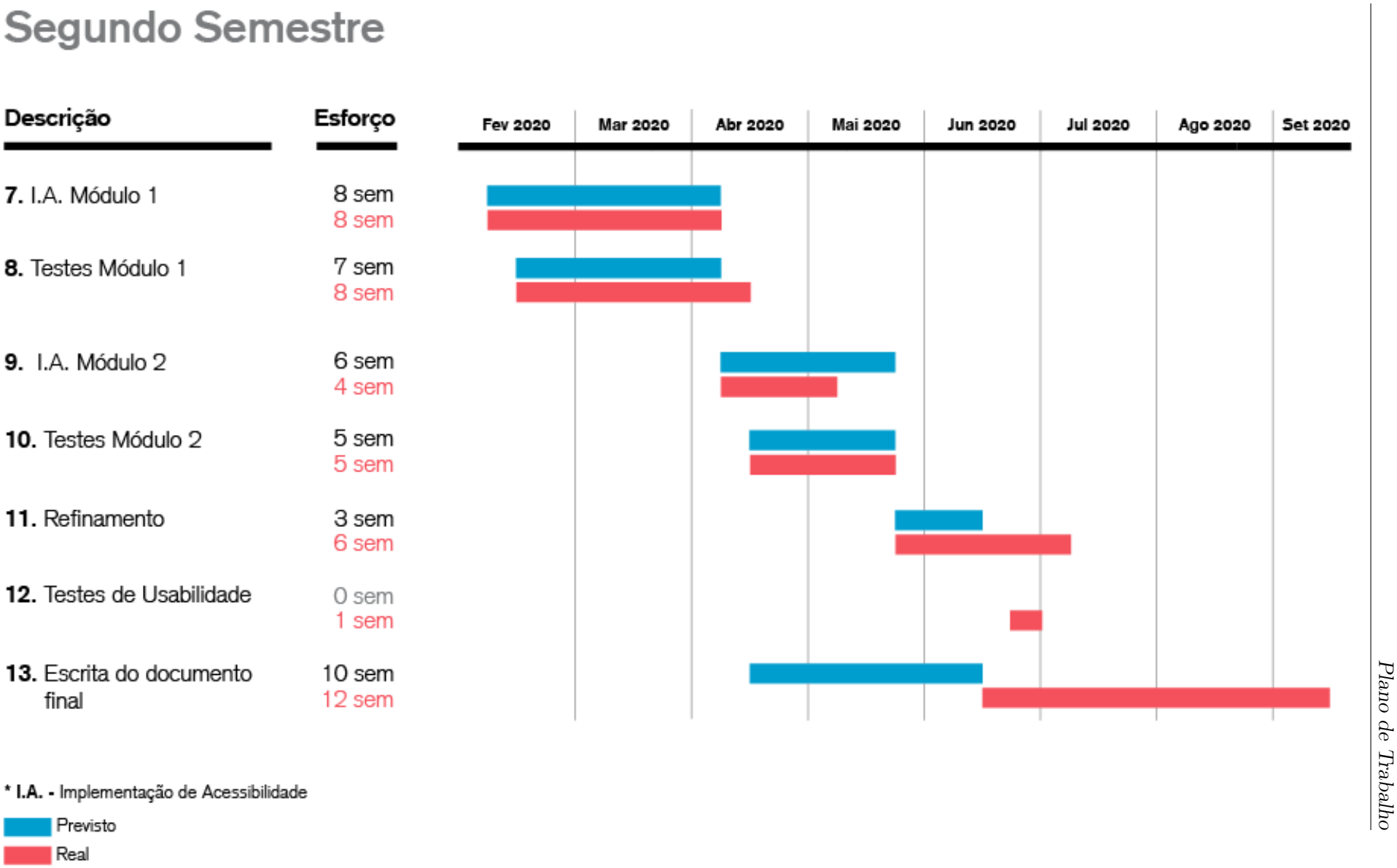


Figura 9.2: Diagrama de Gantt: Segundo Semestre (Esforço Previsto e Real).

# Capítulo 10

# Conclusão

O desenvolvimento exponencial da tecnologia verificado neste novo século, contribuiu para que atualmente dispositivos como o telemóvel ou o computador façam parte do quotidiano de todos. Com isto, a web estendeu-se sobre diversas áreas, sendo agora fonte regular de informação. A agilização dos processos de comércio *online* vincou ainda mais a presença que a web tem no nosso dia-a-dia, sendo já há muito um meio constante de comunicação.

Embora se crie a ideia de que estes avanços tecnológicos chegam a qualquer utilizador, tal não se verifica: utilizadores com algum tipo de deficiência - regra geral - não conseguem aceder ao conteúdo da mesma forma que qualquer outro utilizador. Para isto, a acessibilidade na web é fundamental para assegurar que a experiência seja semelhante para qualquer utilizador, independentemente da sua condição. Contudo, tal não se verifica, com a esmagadora maioria dos *websites* a descurarem a acessibilidade. Para além disso, a falta de acessibilidade pode também ser um constrangimento para profissionais que possuam determinada deficiências, e cujo trabalho dependa de interações com estes dispositivos.

Ao longo deste documento, apresentámos o projeto de tese no âmbito do estágio curricular na empresa Critical Software, onde tivemos oportunidade de estudar estas falhas relativamente à acessibilidade. O projeto lançado previa melhorar a acessibilidade da aplicação PULSAR - aplicação usada diariamente por todos os colaboradores da empresa. Uma vez que o âmbito do estágio foi restringido à acessibilidade para utilizadores invisuais, todo o nosso estudo foi direcionado nesse sentido.

Tendo em conta o tempo disponível, definimos, conjuntamente com um dos colaboradores invisuais da empresa, os dois módulos cuja necessidade a nível de acessibilidade era mais crítica.

O primeiro semestre permitiu-nos estabelecer contacto próximo com as metodologias da empresa e novas tecnologias, enquanto paralelamente procurávamos saber mais sobre as boas práticas da acessibilidade na web. Tudo isto se revelou útil aquando da primeira reunião com o colaborador invisual, tornando a definição de requisitos mais eficaz, na medida em que os conhecimentos reunidos nos permitiram definir abordagens desde cedo.

No segundo semestre, levámos a cabo as implementações dos casos de uso definidos, num processo de contacto regular com os utilizadores finais. Este contacto permanente foi importante na medida em que nos permitiu suprimir problemas no imediato. Considerando ainda que as diretrizes de acessibilidade abrangiam variadas deficiências, deparámo-nos regularmente com problemas específicos dos colaboradores invisuais, pelo que todo o *feedback* dado pelos mesmos viabilizou medidas ajustadas especificamente às suas necessidades. Por fim, os testes finais permitiram-nos verificar que as implementações realizadas melhoraram

a experiência destes utilizadores, tendo sido ainda possível corrigir algumas dificuldades adicionais.

De um modo geral, olhamos para todo o trabalho realizado com resultados positivos. A nível prático, conseguimos melhorar a acessibilidade nos módulos definidos como objetivos principais, evitando potenciais frustrações por parte dos utilizadores finais, melhorando a sua experiência e, eventualmente, promover uma melhor produtividade.

Em suma, consideramos que, acima de tudo, o projeto aqui apresentado foi relevante a diferentes níveis. Com os objetivos principais atingidos, foi ainda possível adquirir experiência em ambiente empresarial, usando tecnologias novas e estudar um tema nunca antes estudado.

Numa nota final, apontamos que os objetivos secundários - tal como toda a aplicação PULSAR - deverá, semelhantemente, verificar melhorias a nível de acessibilidade. Embora o tempo disponível apenas nos tenha permitido assegurar dois dos módulos mais utilizados, seria proveitoso que tal fosse transversal a toda a aplicação, embora reconhecamos a sua complexidade. Assim, esperamos que este documento reforce, também, a importância da acessibilidade em meios digitais, assim como a necessidade de uma web para todos.

# Referências


[1] Accessibility in user-centered design: Evaluating for accessibility. http://www.uiaccess.com/accessucd/evaluate.html. [Online; accessed 2020-09-06].

[2] Angular. https://angular.io/guide/architecture. [Online; accessed 2020-08-30].

[3] axe - the standard in accessibility testing. https://www.deque.com/axe/. [Online; accessed 2020-09-05].

[4] Bootstrap. https://getbootstrap.com/. [Online; accessed 2020-01-12].

[5] Bootstrap 4 toast. https://www.w3schools.com/bootstrap4/bootstrap_toast.asp. [Online; accessed 2020-08-01].

[6] A brief overview of bitbucket. https://bitbucket.org/product/guides/getting-started/overview. [Online; accessed 2020-08-30].

[7] Buttons must have discernible text. https://dequeuniversity.com/rules/axe/3.5/button-name. [Online; accessed 2020-08-02].

[8] Cli overview and command reference. https://angular.io/cli. [Online; accessed 2020-01-12].

[9] Concept: Use case. http://www.utm.mx/~caff/doc/OpenUPWeb/openup/guidances/concepts/use_case_BB199D1B.html. [Online; accessed 2020-08-01].

[10] Confluence: Features functions. https://confluence.atlassian.com/confeval/confluence-evaluator-resources/confluence-features-functions. [Online; accessed 2020-09-06].

[11] Css with superpowers. https://sass-lang.com/. [Online; accessed 2020-08-30].

[12] Fast, reliable, and secure dependency management. https://yarnpkg.com/lang/en/. [Online; accessed 2020-01-12].

[13] How to do an accessibility review. https://developers.google.com/web/fundamentals/accessibility/how-to-review. [Online; accessed 2020-09-06].

[14] How wcag 2.0 differs from wcag 1.0. https://www.w3.org/WAI/WCAG20/from10/diff.php. [Online; accessed 2019-12-29].

[15] Html dom queryselector(). https://www.w3schools.com/jsref/met_document_queryselector.asp. [Online; accessed 2020-08-10].

[16] Html dom queryselectorall() method. https://www.w3schools.com/JSREF/met_document_queryselectorall.asp. [Online; accessed 2020-08-10].

[17] Html tutorial. https://www.w3schools.com/html/. [Online; accessed 2020-08-30].

[18] Introduction to understanding wcag 2.0. https://www.w3.org/TR/UNDERSTANDING-WCAG20/intro.html#introduction-fourprincs-head. [Online; accessed 2019-12-27].

[19] Jenkins user documentation. https://www.jenkins.io/doc/. [Online; accessed 2020-09-01].

[20] Jira software. https://www.atlassian.com/software/jira. [Online; accessed 2020-08-30].

[21] Making a pull request. https://www.atlassian.com/git/tutorials/making-a-pull-request. [Online; accessed 2020-07-30].

[22] Notifications and feedback). https://www.w3.org/WAI/perspective-videos/notifications/. [Online; accessed 2020-09-07].

[23] preventdefault() event method. https://www.w3schools.com/jsref/event_preventdefault.asp. [Online; accessed 2020-08-10].

[24] Sonarlint. https://www.sonarsource.com/products/sonarlint/. [Online; accessed 2020-08-30].

[25] stoppropagation() event method. https://www.w3schools.com/JSREF/event_stoppropagation.asp. [Online; accessed 2020-07-30].

[26] tabindex. https://developer.mozilla.org/pt-BR/docs/Web/HTML/Global_attributes/tabindex. [Online; accessed 2020-09-06].

[27] Tslint. https://palantir.github.io/tslint/. [Online; accessed 2020-01-12].

[28] Typed javascript at any scale. https://www.typescriptlang.org/. [Online; accessed 2020-08-30].

[29] Understanding techniques for wcag success criteria. https://www.w3.org/TR/UNDERSTANDING-WCAG20/understanding-techniques.html. [Online; accessed 2019-12-27].

[30] Understanding techniques for wcag success criteria. https://www.w3.org/WAI/standards-guidelines/wcag/new-in-21/. [Online; accessed 2019-12-27].

[31] Usability web accessibility. https://usability.yale.edu/web-accessibility/articles/headings. [Online; accessed 2020-08-13].

[32] Use cases. https://www.usability.gov/how-to-and-tools/methods/use-cases.html. [Online; accessed 2020-08-01].

[33] W3c. https://www.w3.org/. [Online; accessed 2020-01-13].

[34] W3c's accessible rich internet applications (wai-aria) 1.0 expands accessibility of the open web platform. https://www.w3.org/2014/03/aria.html.en. [Online; accessed 2020-01-10].

[35] Wai-aria authoring practices 1.1. https://www.w3.org/TR/wai-aria-practices-1.1/. [Online; accessed 2020-09-07].

[36] Wai-aria authoring practices 1.1 (3.17 radio button). https://www.w3.org/TR/wai-aria-practices-1.1/. [Online; accessed 2020-09-07].

[37] Wai-aria authoring practices 1.1 (3.5 button). https://www.w3.org/TR/wai-aria-practices-1.1/. [Online; accessed 2020-09-07].

[38] Wai-aria authoring practices 1.1 (3.9 dialog (modal). https://www.w3.org/TR/wai-aria-practices-1.1/. [Online; accessed 2020-09-07].

[39] Wai-aria authoring practices 1.1 (5.1 what are accessible names and descriptions? https://www.w3.org/TR/wai-aria-practices-1.1/. [Online; accessed 2020-09-07].

[40] Wai-aria authoring practices 1.1 (6.9 keyboard shortcuts). https://www.w3.org/TR/wai-aria-practices-1.1/. [Online; accessed 2020-09-07].

[41] Wai-aria authoring practices 1.1 (dialog popup keyboard interaction). https://www.w3.org/TR/wai-aria-practices-1.1/. [Online; accessed 2020-09-07].

[42] Wai-aria basics. https://developer.mozilla.org/en-US/docs/Learn/Accessibility/WAI-ARIA_basics. [Online; accessed 2020-01-12].

[43] Wai-diverse abilities and barriers. https://www.w3.org/WAI/people-use-web/abilities-barriers/. [Online; accessed 2020-01-13].

[44] Web content accessibility guidelines 1.0. https://www.w3.org/TR/WAI-WEBCONTENT/. [Online; accessed 2019-12-22].

[45] Web content accessibility guidelines (wcag) 2.0. http://www.w3.org/TR/WCAG20/. [Online; accessed 2019-12-22].

[46] WAI - introduction to web accessibility. https://www.w3.org/WAI/fundamentals/accessibility-intro/, 2005. [Online; accessed 2020-09-01].

[47] Visual disabilities (blindness). https://webaim.org/articles/visual/blind#howblind, 2013. [Online; accessed 2020-12-10].

[48] Wai-aria overview. https://www.w3.org/WAI/standards-guidelines/aria/, 2016. [Online; accessed 2020-09-06].

[49] Accessible rich internet applications (wai-aria) 1.1. https://www.w3.org/TR/wai-aria/, 2017. [Online; accessed 2020-01-07].

[50] Wai-tools and techniques. https://www.w3.org/WAI/people-use-web/tools-techniques/, 2017. [Online; accessed 2019-12-10].

[51] Wai-essential components of web accessibility. https://www.w3.org/WAI/fundamentals/components/, 2018. [Online; accessed 2020-09-01].

[52] WCAG 2.1. http://www.w3.org/TR/WCAG/, 2018. [Online; accessed 2020-01-13].

[53] Blindness and vision impairment. https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment, 2019. [Online; accessed 2020-01-03].

[54] Semantic structure: Regions, headings, and lists. https://webaim.org/techniques/semanticstructure/, 2020. [Online; accessed 2020-08-13].

[55] Suliman K Almasoud and Hassan I Mathkour. Instant adaptation enrichment technique to improve web accessibility for blind users. In *Proceedings of the 2019 3rd International Conference on Information System and Data Mining*, pages 159–164, 2019.

[56] Rakesh Babu, Rahul Singh, and Jai Ganesh. Understanding blind users' web accessibility and usability problems. *AIS Transactions on Human-Computer Interaction*, 2(3):73–94, 2010.

[57] Len Bass, Paul Clements, and Rick Kazman. *Software Architecture in Practice*, pages 61–79,175–185. Addison-Wesley, 2015.

[58] Yevgen Borodin, Jeffrey P Bigham, Glenn Dausch, and IV Ramakrishnan. More than meets the eye: a survey of screen-reader browsing strategies. In *Proceedings of the 2010 International Cross Disciplinary Conference on Web Accessibility (W4A)*, page 13. ACM, 2010.

[59] Tiago do Carmo Nogueira, Deller James Ferreira, Sérgio Teixeira de Carvalho, Luciana de Oliveira Berretta, and Mycke R Guntijo. Comparing sighted and blind users task performance in responsive and non-responsive web design. *Knowledge and Information Systems*, 58(2):319–339, 2019.

[60] Meggin Kearney & Dave Gash & Rob Dodson. Hiding and updating content. `https://developers.google.com/web/fundamentals/accessibility/semantics-aria/hiding-and-updating-content`. [Online; accessed 2020-09-06].

[61] Meggin Kearney & Dave Gash & Rob Dodson. Introduction to aria. `https://developers.google.com/web/fundamentals/accessibility/semantics-aria`. [Online; accessed 2020-09-06].

[62] Meggin Kearney & Dave Gash & Rob Dodson. Introduction to focus. `https://developers.google.com/web/fundamentals/accessibility/focus`. [Online; accessed 2020-09-06].

[63] Meggin Kearney & Dave Gash & Rob Dodson. Using tabindex. `https://developers.google.com/web/fundamentals/accessibility/focus/using-tabindex`. [Online; accessed 2020-09-06].

[64] Mexhid Ferati and Lirim Sulejmani. Automatic adaptation techniques to increase the web accessibility for blind users. In *International Conference on Human-Computer Interaction*, pages 30–36. Springer, 2016.

[65] Ombretta Gaggi, Giacomo Quadrio, and Armir Bujari. Accessibility for the visually impaired: State of the art and open issues. In *2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–6. IEEE, 2019.

[66] Stéphanie Giraud, Pierre Thérouanne, and Dirk D Steiner. Web accessibility: Filtering redundant and irrelevant information improves website usability for blind users. *International Journal of Human-Computer Studies*, 111:23–35, 2018.

[67] Ramiro Gonçalves, Tânia Rocha, José Martins, Frederico Branco, and Manuel Au-Yong-Oliveira. Evaluation of e-commerce websites accessibility and usability: an e-commerce platform analysis with the inclusion of blind users. *Universal Access in the Information Society*, 17(3):567–583, 2018.

[68] Simon Harper and Yeliz Yesilada. *Web Accessibility: A Foundation for Research*, pages 79–107. Springer-Verlag London Limited, 2008.

[69] Mohammed Saleh Hassouna, Noraidah Sahari, Amirah Ismail, et al. University website accessibility for totally blind users. *Journal of ICT*, 16(1):63–80, 2017.

[70] Evan Leybourn. *Introduction to Scrum - Student Guide*, page 11.

[71] Jennifer Mankoff, Holly Fait, and Tu Tran. Is your web page accessible?: a comparative study of methods for assessing web page accessibility for the blind. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 41–50. ACM, 2005.

[72] Tiago do Carmo Nogueira, Deller James Ferreira, and Matheus Rudolfo Diedrich Ullmann. Impact of accessibility and usability barriers on the emotions of blind users in responsive web design. In *Proceedings of the 18th Brazilian Symposium on Human Factors in Computing Systems*, pages 1–8, 2019.

[73] Afra Pascual, Mireia Ribera, Toni Granollers, and Jordi L Coiduras. Impact of accessibility barriers on the mood of blind, low-vision and sighted users. *Procedia Computer Science*, 27:431–440, 2014.

[74] Christopher Power, André Freire, Helen Petrie, and David Swallow. Guidelines are only half of the story: accessibility problems encountered by blind users on the web. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 433–442. ACM, 2012.

[75] Bujar Raufi, Mexhid Ferati, Xhemal Zenuni, Jaumin Ajdari, and Florije Ismaili. Methods and techniques of adaptive web accessibility for the blind and visually impaired. *Procedia-Social and Behavioral Sciences*, 195:1999–2007, 2015.

[76] Dagfinn Rømen and Dag Svanæs. Validating wcag versions 1.0 and 2.0 through usability testing with disabled users. *Universal Access in the Information Society*, 11(4):375–385, 2012.

[77] Rebecca Wettemann and Trevor White. The internet is unavailable. Research Note T 1 0 3, Nucleus Research, 100 State Street, Boston, MA, 02109, 2019.

# Appendices

This page is intentionally left blank.

## Apêndice A

# Casos de Uso

**Módulo Pedido de Férias/Ausências**

| | |
|---|---|
| **ID** | CS-1 |
| **Nome** | Navegar por *headings* no módulo Pedido de Férias/Ausências |
| **Descrição** | O utilizador invisual, fazendo uso das funcionalidades disponíveis no leitor de ecrã, pode navegar no módulo com recurso aos *headings* definidos. |
| **Ator** | Utilizador invisual |
| **Pré-condições** | • O utilizador invisual está autenticado no PULSAR.<br>• O utilizador invisual está a fazer uso de um leitor de ecrã.<br>• O módulo Pedido de Férias/Ausências está aberto.<br>• O utilizador invisual está a fazer uso do teclado. |
| **Cenário Principal** | 1. O caso de uso começa quando o utilizador invisual faz uso de uma das funcionalidades disponíveis no leitor de ecrã para navegar entre *headings*;<br>2. O sistema disponibiliza *headings* ao leitor de ecrã;<br>3. O leitor de ecrã inicia a sua leitura a partir do *heading* selecionado;<br>4. Os passos 1, 2 e 3 podem repetir-se até o utilizador invisual concluir a tarefa. |

Tabela 1: Navegar por *headings* (Pedido de Férias/Ausências)

| **ID** | CS-2 |
|---|---|
| **Nome** | Identificar elementos interativos |
| **Descrição** | O utilizador invisual, ao navegar para um elemento interativo presente no módulo, recebe *feedback* do leitor de ecrã relativamente ao nome associado ao elemento. |
| **Ator** | Utilizador invisual |
| **Pré-condições** | • O utilizador invisual está autenticado no PULSAR.<br>• O utilizador invisual está a fazer uso de um leitor de ecrã.<br>• O módulo Pedido de Férias/Ausências está aberto.<br>• O utilizador invisual está a fazer uso do teclado. |
| **Cenário Principal** | 1. O caso de uso começa quando o utilizador invisual navega até um elemento interativo presente no módulo.<br>2. O sistema disponibiliza o nome do elemento ao leitor de ecrã.<br>3. O leitor de ecrã anuncia o nome que identifica o respetivo elemento interativo.<br>4. Os passos 1, 2 e 3 podem repetir-se até o utilizador invisual terminar a tarefa. |

Tabela 2: Identificar elementos interativos (Pedido de Férias/Ausêncas)

| **ID** | CS-3 |
|---|---|
| **Nome** | Definir foco inicial no módulo Pedido de Férias/Ausências |
| **Descrição** | Quando o utilizador invisual abre o módulo, o sistema move o foco automaticamente para um dos elementos interativos presentes na janela do módulo. |
| **Ator** | Utilizador invisual |
| **Pré-condições** | • O utilizador invisual está autenticado no PULSAR.<br>• O utilizador invisual está a fazer uso de um leitor de ecrã.<br>• O utilizador invisual está a fazer uso do teclado. |
| **Cenário Principal** | 1. O caso de uso começa quando o utilizador invisual pressiona o botão para abrir o módulo;<br>2. O sistema apresenta a janela do módulo;<br>3. O sistema move o foco para um elemento interativo presente na janela do módulo;<br>4. O caso de uso termina com sucesso. |

Tabela 3: Definir foco inicial (Pedido de Férias/Ausências)

| **ID** | CS-4 |
|---|---|
| **Nome** | Navegar entre elementos do módulo Pedido de Férias/Ausências |
| **Descrição** | O utilizador invisual ao fazer uso do teclado - teclas Tab (avançar) ou Shift+Tab (recuar) - pode navegar (mover o foco) entre elementos interativos pertencentes à janela do módulo. |
| **Ator** | Utilizador invisual |
| **Pré-condições** | • O utilizador invisual está autenticado no PULSAR.<br>• O utilizador invisual está a fazer uso de um leitor de ecrã.<br>• O módulo Pedido de Férias/Ausências está aberto. |
| **Cenário Principal** | 1. O caso de uso começa quando o utilizador invisual pressiona a tecla Tab (ou Shift+Tab) para navegar entre elementos interativos presentes no módulo;<br>2. O sistema move o foco para o elemento interativo seguinte (ou para o elemento interativo anterior) presente no módulo;<br>3. Os passos 1 e 2 podem repetir-se até o utilizador invisual terminar a tarefa. |
| **Cenários Alternativos** | **O foco encontra-se no último elemento interativo**:<br>1. O utilizador invisual pressiona a tecla Tab.<br>2. O sistema move o foco para o primeiro elemento interativo no módulo.<br>**O foco encontra-se no primeiro elemento interativo:**<br>1. O utilizador invisual pressiona as teclas Shift+Tab.<br>2. O sistema move o foco para o último elemento interativo no módulo. |

Tabela 4: Navegar entre elementos interativos (Pedido de Férias/Ausências)

| **ID** | CS-5 |
|---|---|
| **Nome** | Definir foco após interação com um elemento do módulo Pedido de Férias/Ausências |
| **Descrição** | Quando o utilizador invisual interage com um dos elementos presentes no módulo, o foco permanece no elemento. |
| **Ator** | Utilizador invisual |
| **Pré-condições** | • O utilizador invisual está autenticado no PULSAR.<br>• O utilizador invisual está a fazer uso de um leitor de ecrã.<br>• O módulo Pedido de Férias/Ausências está aberto.<br>• O utilizador invisual está a fazer uso do teclado. |
| **Cenário Principal** | 1. O caso de uso começa quando o utilizador invisual navega para um dos elementos interativos presentes no módulo;<br>2. O utilizador invisual interage com o elemento;<br>3. O sistema define o foco no elemento;<br>4. O caso de uso termina com sucesso. |
| **Cenários Alternativos** | **Após a interação, o elemento deixa de estar disponível**:<br>1. O sistema move o foco para outro elemento presente no módulo. |

Tabela 5: Definir foco após interagir com um elemento (Pedido de Férias/Ausências)

| **ID** | CS-6 |
|---|---|
| **Nome** | Definir foco inicial na janela *popup* |
| **Descrição** | Quando o utilizador invisual seleciona um dos dias do calendário, e é apresentada uma janela *popup* referente ao dia, o sistema move o foco automaticamente para um dos elementos interativos presentes na janela *popup*. |
| **Ator** | Utilizador invisual |
| **Pré-condições** | • O utilizador invisual está autenticado no PULSAR.<br>• O utilizador invisual está a fazer uso de um leitor de ecrã.<br>• O módulo Pedido de Férias/Ausências está aberto. |
| **Cenário Principal** | 1. O caso de uso começa quando o utilizador invisual seleciona um dos dias do calendário.<br>2. O sistema apresenta uma janela *popup* referente ao dia selecionado.<br>3. O sistema move o foco para um dos elementos interativos presentes na janela *popup*.<br>4. O caso de uso termina com sucesso. |

Tabela 6: Definir foco inicial na janela *popup* (Pedido de Férias/Ausências)

| **ID** | CS-7 |
|---|---|
| **Nome** | Navegar entre elementos da janela *popup*. |
| **Descrição** | O utilizador invisual ao fazer uso do teclado - teclas Tab (avançar) ou Shift+Tab (recuar) - pode navegar (mover o foco) entre os elementos interativos pertencentes à janela *popup*. |
| **Ator** | Utilizador invisual |
| **Pré-condições** | • O utilizador invisual está autenticado no PULSAR.<br>• O utilizador invisual está a fazer uso de um leitor de ecrã.<br>• O módulo Pedido de Férias/Ausências está aberto.<br>• A janela *popup* está aberta. |
| **Cenário Principal** | 1. O caso de uso começa quando o utilizador invisual pressiona a tecla Tab (ou Shift+Tab) para navegar entre elementos interativos presentes na janela *popup*;<br>2. O sistema move o foco para o elemento interativo seguinte (ou para o elemento interativo anterior) presente na janela *popup*;<br>3. O passo 1 e 2 podem repetir-se até o utilizador terminar a tarefa. |
| **Cenários Alternativos** | **O foco encontra-se no último elemento interativo da janela *popup***:<br>1. O utilizador invisual pressiona a tecla Tab.<br>2. O sistema move o foco para o primeiro elemento interativo na janela;<br>**O foco encontra-se no primeiro elemento interativo da janela *popup*:**<br>1. O utilizador invisual pressiona as teclas Shift+Tab.<br>2. O sistema move o foco para o último elemento interativo na janela; |

Tabela 7: Navegar entre elementos da janela *popup* (Pedido de Férias/Ausências)

| **ID** | CS-8 |
|---|---|
| **Nome** | Definir foco ao fechar a janela *popup* |
| **Descrição** | Quando o utilizador invisual fecha a janela *popup*, o sistema adiciona o foco automaticamente ao dia do calendário que abriu a janela *popup*. |
| **Ator** | Utilizador invisual |
| **Pré-condições** | • O utilizador invisual está autenticado no PULSAR.<br>• O utilizador invisual está a fazer uso de um leitor de ecrã.<br>• O módulo Pedido de Férias/Ausências está aberto.<br>• A janela *popup* está aberta. |
| **Cenário Principal** | 1. O caso de uso começa quando o utilizador invisual fecha a janela *popup*;<br>2. O sistema fecha a janela *popup*;<br>3. O sistema adiciona foco ao dia do calendário que abriu a janela *popup*;<br>4. O caso de uso termina com sucesso. |

Tabela 8: Definir foco ao fechar a janela *popup* (Pedido de Férias/Ausências)

| **ID** | CS-9 |
|---|---|
| **Nome** | Aceder à informação de férias/ausências |
| **Descrição** | O utilizador invisual pode aceder à informação apresentada no calendário relativa às férias e ausências (dias marcados, novos dias pedidos e marcados com pedido de cancelamento) sem, para tal, necessitar de percorrer dia-a-dia. |
| **Ator** | Utilizador invisual |
| **Pré-condições** | • O utilizador invisual está autenticado no PULSAR.<br>• O utilizador invisual está a fazer uso de um leitor de ecrã.<br>• O módulo Pedido de Férias/Ausências está aberto.<br>• O utilizador invisual está a fazer uso do teclado. |
| **Cenário Principal** | 1. O caso de uso começa quando o utilizador invisual navega para uma secção na página com o intuito de verificar os dias com eventos (férias ou ausências);<br>2. O sistema disponibiliza uma mensagem aos leitores de ecrã com a informação relativa aos dias com eventos (férias ou ausências);<br>3. O leitor de ecrã anuncia os dias com eventos (férias ou ausências);<br>4. O caso de uso termina com sucesso. |
| **Cenário Alternativos** | **Não existe informação de férias ou ausências no calendário:**<br>1. O sistema disponibiliza uma mensagem aos leitores de ecrã de que não existe nenhum dia com eventos;<br>2. O leitor de ecrã anuncia a mensagem;<br>3. O caso de uso termina com sucesso. |

Tabela 9: Aceder à informação de férias/ausências (Pedido de Férias/Ausências)

| **ID** | CS-10 |
|---|---|
| **Nome** | Receber *feedback* das opções do *dropdown* |
| **Descrição** | O utilizador invisual, ao navegar entre as diferentes opções do *dropdown*, recebe *feedback* das opções. |
| **Ator** | Utilizador invisual |
| **Pré-condições** | • O utilizador invisual está autenticado no PULSAR.<br>• O utilizador invisual está a fazer uso de um leitor de ecrã.<br>• O módulo Pedido de Férias/Ausências está aberto.<br>• A janela *popup* está aberta.<br>• O utilizador invisual está a fazer uso do teclado. |
| **Cenário Principal** | 1. O caso de uso começa quando o utilizador invisual interage com o elemento *dropdown* de forma a expandi-lo;<br>2. O sistema apresenta as opções do *dropdown*;<br>3. O utilizador invisual navega para uma opção do *dropdown*;<br>4. O leitor de ecrã anuncia a opção;<br>5. Os passos 3 e 4 podem repetir-se até o utilizador invisual encontrar a opção desejada. |

Tabela 10: Receber *feedback* das opções do *dropdown* (Pedido de Férias/Ausências)

| **ID** | CS-11 |
|---|---|
| **Nome** | Confirmar ação realizada |
| **Descrição** | Quando o utilizador invisual fecha a janela *popup* após realizar um novo pedido ou pedido de cancelamento, o sistema disponibiliza uma mensagem aos leitores de ecrã com a informação relativa à ação realizada e, ainda, a informar qual o passo seguinte para concluir a ação. |
| **Ator** | Utilizador invisual |
| **Pré-condições** | • O utilizador invisual está autenticado no PULSAR.<br>• O utilizador invisual está a fazer uso de um leitor de ecrã.<br>• O módulo Pedido de Férias/Ausências está aberto.<br>• A janela *popup* está aberta. |
| **Cenário Principal** | 1. O caso de uso começa quando o utilizador invisual realiza a ação de pedir um novo dia ou cancelar o dia marcado (de férias ou ausências) na janela *popup*;<br>2. O utilizador invisual fecha a janela *popup*;<br>3. O sistema disponibiliza uma mensagem aos leitores de ecrã relativa à ação realizada;<br>4. O leitor de ecrã anuncia a mensagem;<br>5. O caso de uso termina com sucesso. |

Tabela 11: Confirmar ação realizada (Pedido de Férias/Ausências)

| **ID** | CS-12 |
|---|---|
| **Nome** | Identificar campos obrigatórios. |
| **Descrição** | O utilizar invisual recebe *feedback* quando o foco se encontra num elemento de preenchimento obrigatório. |
| **Ator** | Utilizador invisual |
| **Pré-condições** | • O utilizador invisual está autenticado no PULSAR.<br>• O utilizador invisual está a fazer uso de um leitor de ecrã.<br>• O módulo Pedido de Férias/Ausências está aberto.<br>• O utilizador invisual está a fazer uso do teclado. |
| **Cenário Principal** | 1. O caso de uso começa quando o utilizador invisual navega até um campo de preenchimento obrigatório;<br>2. O sistema informa o leitor de ecrã que o campo é obrigatório;<br>3. O leitor de ecrã anuncia que o campo é de preenchimento obrigatório;<br>4. O caso de uso termina com sucesso. |

Tabela 12: Identificar campos obrigatórios (Pedido de Férias/Ausências)

| **ID** | CS-13 |
|---|---|
| **Nome** | Disponibilizar *feedback* durante a realização de uma ação no módulo Pedido de Férias/Ausências |
| **Descrição** | O utilizador invisual recebe *feedback* quando está a realizar uma ação no módulo. |
| **Ator** | Utilizador invisual |
| **Pré-condições** | • O utilizador invisual está autenticado no PULSAR.<br>• O utilizador invisual está a fazer uso de um leitor de ecrã.<br>• O módulo Pedido de Férias/Ausências está aberto. |
| **Cenário Principal** | 1. O caso de uso começa quando o utilizador invisual inicia a realização de uma ação no módulo;<br>2. O utilizador invisual preenche os campos referentes à ação;<br>3. O utilizador invisual pressiona um botão para concluir a ação;<br>4. O sistema disponibiliza uma mensagem de sucesso aos leitores de ecrã;<br>5. O leitor de ecrã anuncia a mensagem;<br>6. O caso de uso termina com sucesso; |
| **Cenários Alternativos** | **O utilizador invisual não preenche todos os campos obrigatórios:**<br>1. O sistema disponibiliza uma mensagem de alerta aos leitores de ecrã;<br>2. O leitor de ecrã anuncia a mensagem;<br>**O utilizador invisual não preencheu corretamente os campos:**<br>1. O sistema disponibiliza uma mensagem de alerta aos leitores de ecrã;<br>2. O leitor de ecrã anuncia a mensagem; |

Tabela 13: Disponibilizar *feedback* durante a realização de uma ação (Pedido de Férias/Ausências)

| **ID** | CS-14 |
|---|---|
| **Nome** | Aceder à informação sobre os atalhos de teclado |
| **Descrição** | O utilizador invisual é informado sobre os atalhos de teclado disponíveis no módulo quando este abre. |
| **Ator** | Utilizador invisual |
| **Pré-condições** | • O utilizador invisual está autenticado no PULSAR.<br>• O utilizador invisual está a fazer uso de um leitor de ecrã. |
| **Cenário Principal** | 1. O caso de uso começa quando o utilizador invisual abre o módulo;<br>2. O sistema disponibiliza informação ao leitor de ecrã relativa aos atalhos de teclado disponíveis no módulo;<br>3. O leitor de ecrã anuncia a informação dos atalhos de teclado;<br>4. O caso de uso termina com sucesso. |

Tabela 14: Aceder à informação sobre os atalhos de teclado (Pedido de Férias/Ausências)

| **ID** | CS-15 |
|---|---|
| **Nome** | Aceder à informação do dia |
| **Descrição** | O utilizador invisual, ao navegar pelos dias do calendário, recebe *feedback* sobre a informação apresentada por cores (relativa a férias ou ausências) em cada um dos dias. |
| **Ator** | Utilizador invisual |
| **Pré-condições** | • O utilizador invisual está autenticado no PULSAR.<br>• O utilizador invisual está a fazer uso de um leitor de ecrã.<br>• O módulo Pedido de Férias/Ausências está aberto.<br>• O utilizador invisual está a fazer uso do teclado. |
| **Cenário Principal** | 1. O caso de uso começa quando o utilizador invisual navega para um dos dias do calendário;<br>2. O sistema disponibiliza a informação relativa à informação apresentada por cores aos leitores de ecrã;<br>3. O leitor de ecrã anuncia a informação do dia (codificada por cores);<br>4. Os passos 1,2 e 3 repetem-se até o utilizador invisual terminar a tarefa; |
| **Cenários Alternativos** | **O dia não contém informação apresentada por cores:**<br>1. O sistema disponibiliza apenas a informação relativa à data aos leitores de ecrã;<br>2. O leitor de ecrã anuncia a data; |

Tabela 15: Aceder à informação do dia (Pedido de Férias/Ausências)

| **ID** | CS-16 |
|---|---|
| **Nome** | Navegar para o dia atual no calendário no módulo Pedido de Férias/Ausências |
| **Descrição** | O utilizador invisual, ao fazer uso de um atalho de teclado, pode navegar (mover o foco) para o dia atual apresentado no calendário. |
| **Ator** | Utilizador invisual |
| **Pré-condições** | • O utilizador invisual está autenticado no PULSAR.<br>• O utilizador invisual está a fazer uso de um leitor de ecrã.<br>• O módulo Pedido de Férias/Ausências está aberto. |
| **Cenário Principal** | 1. O caso de uso começa quando o utilizador invisual faz uso do atalho teclado definido para navegar para o dia atual no calendário;<br>2. O sistema adiciona foco no dia atual do calendário;<br>3. O caso de uso termina com sucesso; |

Tabela 16: Mover o foco para o dia atual (Pedido de Férias/Ausências)

| **ID** | CS-17 |
|---|---|
| **Nome** | Navegar entre dias do calendário com eventos |
| **Descrição** | O utilizador invisual, ao fazer uso de um atalho de teclado, pode navegar (mover o foco) apenas entre dias do calendário que contenham algum evento relativo a férias ou ausências (dias marcados, novos dias pedidos ou marcados com pedido de cancelamento). |
| **Ator** | Utilizador invisual |
| **Pré-condições** | • O utilizador invisual está autenticado no PULSAR.<br>• O utilizador invisual está a fazer uso de um leitor de ecrã.<br>• O módulo Pedido de Férias/Ausências está aberto. |
| **Cenário Principal** | 1. O caso de uso começa quando o utilizador invisual faz uso do atalho teclado definido para navegar para entre dias do calendário com eventos;<br>2. O sistema adiciona foco no próximo dia do calendário com evento;<br>3. Os passos 1 e 2 podem repetir-se até o utilizador terminar a tarefa; |
| **Cenários Alternativos** | **O calendário não contêm nenhum dia com evento:**<br>1. O utilizador invisual faz uso do atalho teclado definido para navegar para entre dias do calendário com eventos;<br>2. O sistema mantem o foco no elemento ativo no momento; |

Tabela 17: Navegar entre dias do calendário com eventos (Pedido de Férias/Ausências)

### Módulo Reportar Esforço

| | |
|---|---|
| **ID** | CS-18 |
| **Nome** | Navegar por *headings* no módulo Reportar Esforço |
| **Descrição** | O utilizador invisual, ao fazer uso das funcionalidades disponíveis no leitor de ecrã, consegue navegar entre *headings* definidos no módulo. |
| **Ator** | Utilizador invisual |
| **Pré-condições** | • O utilizador invisual está autenticado no PULSAR.<br>• O utilizador invisual está a fazer uso de um leitor de ecrã.<br>• O módulo Reportar Esforço está aberto.<br>• O utilizador invisual está a fazer uso do teclado. |
| **Cenário Principal** | 1. O caso de uso começa quando o utilizador invisual faz uso de uma das funcionalidades disponíveis no leitor de ecrã para navegar entre *headings*;<br>2. O sistema disponibiliza *headings* ao leitor de ecrã;<br>3. O leitor de ecrã inicia a sua leitura a partir do *heading* selecionado;<br>4. Os passos 1 e 2 repetem-se até o utilizador invisual concluir a tarefa. |

Tabela 18: Navegar por *headings* (Reportar Esforço)

| **ID** | CS-19 |
|---|---|
| **Nome** | Definir foco inicial no módulo Reportar Esforço |
| **Descrição** | Quando o utilizador invisual abre o módulo, o sistema move o foco automaticamente para um dos elementos interativos presentes na janela do módulo. |
| **Ator** | Utilizador invisual |
| **Pré-condições** | • O utilizador invisual está autenticado no PULSAR.<br>• O utilizador invisual está a fazer uso de um leitor de ecrã. |
| **Cenário Principal** | 1. O caso de uso começa quando o utilizador invisual pressiona o botão para abrir o módulo;<br>2. O sistema apresenta a janela do módulo;<br>3. O sistema move o foco para um dos elementos interativos presentes na janela do módulo;<br>4. O caso de uso termina com sucesso; |

Tabela 19: Definir foco inicial (Reportar Esforço)

| **ID** | CS-20 |
|---|---|
| **Nome** | Navegar entre elementos interativos do módulo Reportar Esforço |
| **Descrição** | O utilizar invisual faz uso do teclado - teclas Tab (avançar) ou Shift+Tab (recuar) - para mover o foco apenas entre elementos interativos pertencentes à janela do módulo. |
| **Ator** | Utilizador invisual |
| **Pré-condições** | • O utilizador invisual está autenticado no PULSAR.<br>• O utilizador invisual está a fazer uso de um leitor de ecrã.<br>• O módulo Reportar Esforço está aberto. |
| **Cenário Principal** | 1. O caso de uso começa quando o utilizador invisual pressiona a tecla Tab (ou Shift+Tab) para navegar entre elementos interativos presentes no módulos;<br>2. O sistema move o foco para o elemento interativo seguinte (ou para o elemento interativo anterior) presente no módulo;<br>3. Os passos 1 e 2 podem repetir-se até o utilizador terminar a tarefa. |
| **Cenários Alternativos** | **O foco encontra-se no último elemento interativo do módulo**:<br>1. O utilizador invisual pressiona a tecla Tab.<br>2. O sistema move o foco para o primeiro elemento interativo no módulo;<br>**O foco encontra-se no primeiro elemento interativo do módulo:**<br>1. O utilizador invisual pressiona as teclas Shift+Tab.<br>2. O sistema move o foco para o último elemento interativo no módulo; |

Tabela 20: Navegar entre elementos (Reportar Esforço)

| **ID** | CS-21 |
|---|---|
| **Nome** | Definir foco após interação com um elemento do módulo Reportar Esforço |
| **Descrição** | Quando o utilizador invisual interage com um dos elementos presentes no módulo Reportar Esforço, fazendo uso do teclado, o foco permanece no elemento. |
| **Ator** | Utilizador invisual |
| **Pré-condições** | • O utilizador invisual está autenticado no PULSAR.<br>• O utilizador invisual está a fazer uso de um leitor de ecrã.<br>• O módulo Reportar Esforço está aberto.<br>• O utilizador invisual está a fazer uso do teclado. |
| **Cenário Principal** | 1. O caso de uso começa quando o utilizador invisual navega para um dos elementos presentes no módulo;<br>2. O utilizador invisual interage com o elemento;<br>3. O sistema mantém foco no elemento;<br>4. O caso de uso termina com sucesso. |
| **Cenários Alternativos** | **Após a interação, o elemento deixa de estar disponível**:<br>1. O sistema move o foco para outro elemento presente na janela do módulo. |

Tabela 21: Definir foco após interação com um elemento (Reportar Esforço)

| **ID** | CS-22 |
|---|---|
| **Nome** | Definir foco no painel lateral |
| **Descrição** | Quando o utilizador invisual seleciona um dos dias do calendário, sendo apresentado um painel lateral com informação referente ao dia, o sistema move o foco automaticamente para um dos elementos presentes no painel lateral. |
| **Ator** | Utilizador invisual |
| **Pré-condições** | • O utilizador invisual está autenticado no PULSAR.<br>• O utilizador invisual está a fazer uso de um leitor de ecrã.<br>• O módulo Reportar Esforço está aberto. |
| **Cenário Principal** | 1. O caso de uso começa quando o utilizador invisual seleciona um dos dias do calendário;<br>2. O sistema apresenta um painel lateral referente ao dia selecionado;<br>3. O sistema adiciona foco a um dos elementos interativos presentes no painel lateral;<br>4. O caso de uso termina com sucesso. |

Tabela 22: Definir foco no painel lateral (Módulo Reportar Esforço)

| **ID** | CS-23 |
|---|---|
| **Nome** | Definir foco inicial na janela de recorrência |
| **Descrição** | O quando o utilizador invisual abre a janela de recorrência, o sistema move o foco automaticamente para um dos elementos interativos presentes na janela |
| **Ator** | Utilizador invisual |
| **Pré-condições** | • O utilizador invisual está autenticado no PULSAR.<br>• O utilizador invisual está a fazer uso de um leitor de ecrã.<br>• O módulo Reportar Esforço está aberto. |
| **Cenário Principal** | 1. O caso de uso começa quando o utilizador invisual pressiona o botão para abrir a janela de recorrência;<br>2. O sistema apresenta a janela de recorrência;<br>3. O sistema adiciona foco a um dos elementos interativos presente na janela de recorrência;<br>4. O caso de uso termina com sucesso; |

Tabela 23: Definir foco inicial na janela de recorrência (Reportar Esforço)

| **ID** | CS-24 |
|---|---|
| **Nome** | Navegar entre elementos da janela de recorrência. |
| **Descrição** | O utilizar invisual ao fazer uso do teclado - teclas Tab (avançar) ou Shift+Tab (recuar) - pode navegar (mover o foco) entre os elementos interativos pertencentes à janela de recorrência. |
| **Ator** | Utilizador invisual |
| **Pré-condições** | • O utilizador invisual está autenticado no PULSAR.<br>• O utilizador invisual está a fazer uso de um leitor de ecrã.<br>• O módulo Reportar Esforço está aberto.<br>• A janela de recorrência está aberta. |
| **Cenário Principal** | 1. O caso de uso começa quando o utilizador invisual pressiona a tecla Tab (ou Shift+Tab) para navegar entre elementos interativos presentes na janela de recorrência;<br>2. O sistema move o foco para o elemento interativo seguinte (ou para o elemento interativo anterior) presente na janela de recorrência;<br>3. Os passos 1 e 2 podem repetir-se até o utilizador terminar a tarefa. |
| **Cenários Alternativos** | **O foco encontra-se no último elemento interativo da janela**:<br>1. O utilizador invisual pressiona a tecla Tab.<br>2. O sistema move o foco para o primeiro elemento interativo na janela;<br>**O foco encontra-se no primeiro elemento interativo da janela:**<br>1. O utilizador invisual pressiona as teclas Shift+Tab.<br>2. O sistema move o foco para o último elemento interativo na janela; |

Tabela 24: Navegar entre elementos da janela de recorrência (Reportar Esforço)

| **ID** | CS-25 |
|---|---|
| **Nome** | Definir foco ao fechar a janela de recorrência |
| **Descrição** | Quando o utilizador invisual fecha a janela de recorrência, o sistema adiciona foco automaticamente ao botão que abriu a janela de recorrência. |
| **Ator** | Utilizador invisual |
| **Pré-condições** | • O utilizador invisual está autenticado no PULSAR.<br>• O utilizador invisual está a fazer uso de um leitor de ecrã.<br>• O módulo Reportar Esforço está aberto.<br>• A janela de recorrência está aberta. |
| **Cenário Principal** | 1. O caso de uso começa quando o utilizador invisual fecha a janela de recorrência;<br>2. O sistema fecha a janela de recorrência;<br>3. O sistema adiciona foco ao botão que abriu a janela de recorrência;<br>4. O caso de uso termina com sucesso. |

Tabela 25: Adicionar foco ao fechar a janela de recorrência (Reportar Esforço)

| **ID** | CS-26 |
|---|---|
| **Nome** | Definir foco após interação com um elemento da janela de recorrência |
| **Descrição** | Quando o utilizador invisual interage com um dos elementos presentes na janela de recorrência, o foco deverá permanecer no elemento. |
| **Ator** | Utilizador invisual |
| **Pré-condições** | • O utilizador invisual está autenticado no PULSAR.<br>• O utilizador invisual está a fazer uso de um leitor de ecrã.<br>• O módulo Reportar Esforço está aberto.<br>• A janela de recorrência está aberta.<br>• O utilizador invisual está a fazer uso do teclado. |
| **Cenário Principal** | 1. O caso de uso começa quando o utilizador invisual navega para um dos elementos presentes na janela de recorrência;<br>2. O utilizador invisual interage com o elemento;<br>3. O sistema mantém o foco no elemento;<br>4. O caso de uso termina com sucesso. |
| **Cenários Alternativos** | **Após a interação, o elemento deixa de estar disponível**:<br>1. O sistema adiciona foco a um dos elementos presentes à janela. |

Tabela 26: Definir foco após interação com um elemento da janela de recorrência (Reportar Esforço)

| **ID** | CS-27 |
|---|---|
| **Nome** | Aceder à informação apresentada no calendário |
| **Descrição** | Quando o utilizador invisual navega para o *heading* referente ao título do calendário, recebe *feedback* dos dias com esforço no mês (equivalente à informação apresentada por cores no calendário). |
| **Ator** | Utilizador invisual |
| **Pré-condições** | • O utilizador invisual está autenticado no PULSAR.<br>• O utilizador invisual está a fazer uso de um leitor de ecrã.<br>• O módulo Reportar Esforço está aberto.<br>• O utilizador invisual está a fazer uso do teclado. |
| **Cenário Principal** | 1. O caso de uso começa quando o utilizador invisual navega, fazendo uso das funcionalidades do leitor de ecrã, para o *heading* referente ao título do calendário;<br>2. O leitor de ecrã anuncia a informação;<br>3. O caso de uso termina com sucesso; |
| **Cenários Alternativos** | **Não existem dias no calendário com horas reportadas**:<br>1. O leitor de ecrã anuncia que não existem dias reportados no mês;<br>**Todos os dias estão reportados**:<br>1. O leitor de ecrã anuncia que todos dias do mês já se encontramm reportados; |

Tabela 27: Aceder à informação do calendário (Reportar Esforço)

| **ID** | CS-28 |
|---|---|
| **Nome** | Aceder à informação do esforço através das legendas. |
| **Descrição** | Quando o utilizador invisual navega nas legendas referentes à semana e ao mês, pode aceder à informação relativa aos dias sem esforço, com esforço parcial ou esforço excedente (por semana ou por mês). |
| **Ator** | Utilizador invisual |
| **Pré-condições** | • O utilizador invisual está autenticado no PULSAR.<br>• O utilizador invisual está a fazer uso de um leitor de ecrã.<br>• O módulo Reportar Esforço está aberto.<br>• O utilizador está a fazer uso do teclado. |
| **Cenário Principal** | 1. O caso de uso começa quando o utilizador invisual seleciona um dia no calendário;<br>2. O utilizador invisual navega na legenda (Semana ou Mês);<br>3. O sistema disponibiliza a informação com os dias sem esforço, com esforço excedente ou parcial (por semana ou mês);<br>4. O leitor de ecrã anuncia a informação;<br>5. O caso de uso termina com sucesso. |

Tabela 28: Aceder à informação do esforço através das legendas (Reportar Esforço)

| **ID** | CS-29 |
|---|---|
| **Nome** | Identificar campos obrigatórios. |
| **Descrição** | O utilizar invisual recebe *feedback* quando o foco se encontra num elemento de preenchimento obrigatório. |
| **Ator** | Utilizador invisual |
| **Pré-condições** | • O utilizador invisual está autenticado no PULSAR.<br>• O utilizador invisual está a fazer uso de um leitor de ecrã.<br>• O módulo Reportar Esforço está aberto.<br>• O utilizador invisual está a fazer uso do teclado. |
| **Cenário Principal** | 1. O caso de uso começa quando o utilizador invisual navega até um campo de preenchimento obrigatório;<br>2. O sistema informa o leitor de ecrã que o campo é obrigatório;<br>3. O leitor de ecrã anuncia que o campo é de preenchimento obrigatório;<br>4. O caso de uso termina com sucesso. |

Tabela 29: Identificar campos obrigatórios (Módulo Reportar Esforço)

| **ID** | CS-30 |
|---|---|
| **Nome** | Disponibilizar *feedback* durante a realização de uma ação no módulo Pedido de Reportar Esforço |
| **Descrição** | O utilizador invisual recebe *feedback* quando está a realizar uma ação no módulo. |
| **Ator** | Utilizador invisual |
| **Pré-condições** | • O utilizador invisual está autenticado no PULSAR.<br>• O utilizador invisual está a fazer uso de um leitor de ecrã.<br>• O módulo Reportar Esforço está aberto. |
| **Cenário Principal** | 1. O caso de uso começa quando o utilizador invisual inicia a realização de uma ação no módulo;<br>2. O utilizador invisual preenche os campos referentes à ação;<br>3. O utilizador invisual pressiona um botão para concluir a ação;<br>4. O sistema disponibiliza uma mensagem de sucesso aos leitores de ecrã;<br>5. O leitor de ecrã anuncia a mensagem;<br>6. O caso de uso termina com sucesso; |
| **Cenários Alternativos** | **O utilizador invisual não preenche todos os campos obrigatórios:**<br>1. O sistema disponibiliza uma mensagem de alerta aos leitores de ecrã;<br>2. O leitor de ecrã anuncia a mensagem;<br>**O utilizador invisual não preencheu corretamente os campos:**<br>1. O sistema disponibiliza uma mensagem de alerta aos leitores de ecrã;<br>2. O leitor de ecrã anuncia a mensagem; |

Tabela 30: Disponibilizar *feedback* durante a realização de uma ação (Reportar Esforço)

| **ID** | CS-31 |
|---|---|
| **Nome** | Aceder à informação sobre os atalhos de teclado |
| **Descrição** | O utilizador invisual é informado sobre os atalhos de teclado disponíveis no módulo quando este abre. |
| **Ator** | Utilizador invisual |
| **Pré-condições** | • O utilizador invisual está autenticado no PULSAR.<br>• O utilizador invisual está a fazer uso de um leitor de ecrã. |
| **Cenário Principal** | 1. O caso de uso começa quando o utilizador invisual abre o módulo;<br>2. O sistema disponibiliza informação relativa aos atalhos de teclado disponíveis no módulo;<br>3. O caso de uso termina com sucesso. |

Tabela 31: Aceder à informação sobre os atalhos (Reportar Esforço)

| **ID** | CS-32 |
|---|---|
| **Nome** | Disponibilizar atualizações da legenda do dia |
| **Descrição** | O utilizador invisual recebe *feedback* referente à legenda do dia (horas esperadas e horas reportadas) quando é realizada uma ação que despoleta uma alteração na legenda (selecionar um dia no calendário, adicionar, eliminar ou editar uma tarefa). |
| **Ator** | Utilizador invisual |
| **Pré-condições** | • O utilizador invisual está autenticado no PULSAR.<br>• O utilizador invisual está a fazer uso de um leitor de ecrã.<br>• O módulo Reportar Esforço está aberto. |
| **Cenário Principal** | 1. O caso de uso começa quando o utilizador invisual realiza uma ação que despoleta uma atualização da legenda Dia;<br>2. O sistema atualiza a legenda Dia;<br>3. O sistema disponibiliza uma mensagem com a informação das horas diárias esperadas e reportadas aos leitores de ecrã;<br>4. O leitor de ecrã anuncia a mensagem;<br>5. O caso de uso termina com sucesso; |

Tabela 32: Disponibilizar atualizações da legenda do dia (Reportar Esforço)

| **ID** | CS-33 |
|---|---|
| **Nome** | Aceder à informação do dia |
| **Descrição** | O utilizador invisual, ao navegar pelos dias do calendário, recebe *feedback* sobre a informação apresentada por cores (esforço parcial, esforço excedente, esforço expectável, feriado, férias ou ausência) em cada um dos dias. |
| **Ator** | Utilizador invisual |
| **Pré-condições** | • O utilizador invisual está autenticado no PULSAR.<br>• O utilizador invisual está a fazer uso de um leitor de ecrã.<br>• O módulo Reportar Esforço está aberto. |
| **Cenário Principal** | 1. O caso de uso começa quando o utilizador invisual navega para um dos dias do calendário;<br>2. O sistema disponibiliza a informação relativa à informação apresentada por cores aos leitores de ecrã;<br>3. O leitor de ecrã anuncia a informação do dia (codificada por cores);<br>4. Os passos 1,2 e 3 podem repetir-se até o utilizador invisual terminar a tarefa; |
| **Cenários Alternativos** | **O dia não contém informação representada por cores:**<br>1. O sistema disponibiliza apenas a informação relativa à data aos leitores de ecrã;<br>2. O leitor de ecrã anuncia a data; |

Tabela 33: Aceder à informação do dia (Reportar Esforço)

| **ID** | CS-34 |
|---|---|
| **Nome** | Navegar para o dia atual no calendário |
| **Descrição** | O utilizador invisual, ao fazer uso de um atalho de teclado, pode navegar (mover o foco) para o dia atual no calendário. |
| **Ator** | Utilizador invisual |
| **Pré-condições** | • O utilizador invisual está autenticado no PULSAR.<br>• O utilizador invisual está a fazer uso de um leitor de ecrã.<br>• O módulo Reportar Esforço está aberto. |
| **Cenário Principal** | 1. O caso de uso começa quando o utilizador invisual faz uso do atalho teclado definido para navegar para o dia atual;<br>2. O sistema adiciona foco no dia atual do calendário;<br>3. O caso de uso termina com sucesso. |
| **Cenários Alternativos** | **O calendário não se encontra no mês atual:**<br>1. O utilizador invisual faz uso do atalho teclado definido para navegar para o dia atual no calendário;<br>2. O sistema mantém o foco no elemento com foco no momento; |

Tabela 34: Navegar para o dia atual (Reportar Esforço)

| **ID** | CS-35 |
|---|---|
| **Nome** | Navegar para o dia selecionado no calendário |
| **Descrição** | O utilizador invisual, ao fazer uso de um atalho de teclado, pode navegar (mover o foco) para o dia selecionado no calendário. |
| **Ator** | Utilizador invisual |
| **Pré-condições** | • O utilizador invisual está autenticado no PULSAR.<br>• O utilizador invisual está a fazer uso de um leitor de ecrã.<br>• O módulo Reportar Esforço está aberto. |
| **Cenário Principal** | 1. O caso de uso começa quando o utilizador invisual faz uso do atalho de teclado definido para navegar para o dia selecionado no calendário;<br>2. O sistema adiciona foco no dia selecionado do calendário;<br>3. O caso de uso termina com sucesso; |
| **Cenários Alternativos** | **Não existe nenhum dia selecionado no calendário:**<br>1. O utilizador invisual faz uso do atalho de teclado definido para navegar para o dia selecionado no calendário;<br>2. O sistema mantém o foco no elemento ativo de momento; |

Tabela 35: Navegar para o dia selecionado (Reportar Esforço)