UNIVERSIDADE
BEIRA INTERIOR

Desenvolvimento Web Full-Stack na Empresa
Tetrapi SA

Dário Vítor Freitas Santos

Relatório de Estágio para obtenção do Grau de Mestre em
Engenharia Informática
(2º ciclo de estudos)

Orientador: Prof. Doutora Maria Paula Prata de Sousa
Supervisor da Empresa: Doutor João António Câmara Correia

outubro de 2025

Relatório de Estágio para obtenção do Grau de Mestre em Engenharia Informática elaborado por Dário Vítor Freitas Santos, Licenciado em Engenharia Informática pela Universidade da Beira Interior, sob orientação da Doutora Maria Paula Prata de Sousa, Professora Auxiliar do Departamento de Informática da Universidade da Beira Interior, e Investigadora do Instituto de Telecomunicações no âmbito de trabalho financiado pela FCT/MECI através de fundos nacionais e quando aplicável cofinanciado por fundos comunitários no âmbito do UID/50008: Instituto de Telecomunicações.

Declaração de Integridade

Eu, Dário Vítor Freitas Santos, que abaixo assino, estudante com o número de inscrição M10929 de Engenharia Informática da Faculdade de Engenharia, declaro ter desenvolvido o presente trabalho e elaborado o presente texto em total consonância com o Código de Integridades da Universidade da Beira Interior.

Mais concretamente, afirmo não ter incorrido em qualquer das variedades de Fraude Académica, e que aqui declaro conhecer, que em particular atendi à exigida referenciação de frases, extratos, imagens e outras formas de trabalho intelectual, e assumindo assim na íntegra as responsabilidades da autoria.

Universidade da Beira Interior, Covilhã 04/10/2025

Assinado por: **Dário Vítor Freitas Santos**
Num. de Identificação: 14816220
Data: 2025.10.04 17:33:07+00'00'

Agradecimentos

Ao longo do meu percurso académico fui apoiado por amigos, família, colegas e professores. Queria primeiramente agradecer à minha família, a qual sempre me apoio e permitiu a minha evolução pessoal e profissional, em especial à minha mãe, pai, irmã e avó.
Num segundo ponto, queria também agradecer à minha orientadora de estágio, a professora Doutora Maria Paula Prata de Sousa, que me acompanhou e orientou ao longo de todo o estágio. A Universidade da Beira Interior e os seus colaboradores que me moldaram, e em especial ao meu antigo professor Doutor Simão Melo de Sousa.
Queria também agradecer à empresa Tetrapi, que sem a mesma não seria possível a realização do estágio, ao meu supervisor Doutor João Correia, e a todos os meus colegas de trabalho com os quais enfrentamos novos desafios todos os dias.
Por fim, queria agradecer à minha namorada que me atura todos os dias e me ajuda a enfrentar um dia de cada vez.

Resumo

Este documento descreve o estágio realizado para a unidade curricular Dissertação ou Estágio no segundo ciclo de Engenharia Informática da Universidade da Beira Interior. Este estágio focou-se no desenvolvimento de uma plataforma *web* na empresa Tetrapi SA com recurso à *framework* `Symfony`.
O estágio teve como objetivo principal o desenho e desenvolvimento de uma plataforma *web* para a Subdireção Regional dos Transportes Terrestres, e teve como objetivos adicionais a orientação de outras equipas de desenvolvimento, a orientação de um estágio curricular e a manutenção de uma plataforma *web* em fase de validação pelo cliente.
Com o intuito de apresentar o máximo de informação possível, o presente relatório está dividido em múltiplos capítulos que abordam a empresa e o plano de estágio; o estado da arte no que consta ao desenvolvimento de plataformas *web* na *framework* `Symfony`; e o desenvolvimento propriamente dito da plataforma Sistema de Gestão de Contra Ordenações (SGCO).

Palavras-chave

`Symfony`, `Tailwind`, `Backend`, `Frontend`, `Symfony UX`, `Doctrine`, Arquitetura `MVSC`, Contra-ordenações

Abstract

This document describes the internship carried out for the Dissertação ou Estágio subject, in the context of the Computer Science master's degree, at the University of Beira Interior. The internship aimed for a developer team's incorporation at Tetrapi, in the scope of a web platform's development, using the Symfony framework.
The internship had has its main objective the planning and develpoment of a web platform for the Subdireção Regional dos Transportes Terrestres, and had has its secondary objectives the mentoring of other development teams, the orientation of a curricular internship and the maintenance of a web platform in the customer validation phase.
Aiming for a meticulous description, this paper has been partitioned into multiple chapters that address the company and internship plan; the state of the art in the development of web platforms in the `Symfony` framework; and the development of the SGCO platform itself.

Keywords

`Symfony`, `Tailwind`, `Backend`, `Frontend`, `Symfony UX`, `Doctrine`, `MVSC` Architecture, Administrative offenses

Índice

Lista de Figuras

Lista de Excertos de Código

Acrónimos

PM Polícia Municipal

API *Application Programming Interface*

CSS *Cascading Style Sheets*

DTO *Data Transfer Object*

GNR Guarda Nacional Republicana

IDE *Integrated Development Environment*

MVC *Model View Controller*

ORM *Object-Relational Mapper*

PDF *Portable Document Format*

PME Pequena ou Média Empresa

PSP Polícia de Segurança Pública

SQL *Structured Query Language*

HTML *HyperText Markup Language*

AJAX *Asynchronous JavaScript and XML*

ANSR Autoridade Nacional de Segurança Rodoviária

MVCS *Model View Controller Service*

SGCO Sistema de Gestão de Contra Ordenações

SRTT Subdireção Regional dos Transportes Terrestres

CTeSP Cursos Técnicos Superiores Profissionais

Capítulo 1

Introdução

1.1 Enquadramento

Com a crescente digitalização dos serviços públicos e privados, e um aliciante mercado de trabalho no que toca à área de Engenharia Informática, um estágio acaba por se revelar como um caminho atraente para quem deseja desenvolver as suas competências num ambiente corporativo, sejam estas técnicas, de gestão ou de colaboração.
Neste sentido, surgiu a oportunidade de realizar um estágio curricular na empresa Tetrapi SA, facilitando a integração no tecido empresarial e a transferência de conhecimentos apenas adquiríveis por experiência profissional. A Tetrapi SA é uma empresa de renome regional que apresenta serviços educativos e de desenvolvimento de *software* com sede na Região Autónoma dos Açores.

1.2 Motivação

Com o avançar da tecnologia surge a tentativa da sua adoção nos diferentes meios, para usufruir da produtividade adicional que novas ferramentas apresentam. Em meios mais clássicos, em que reina o uso do papel e processos manuais, a introdução de ferramentas tecnológicas pode revolucionar o método de trabalho e agilizar as tarefas realizadas.
Atualmente, o Governo Regional dos Açores, mais concretamente, a Subdireção Regional dos Transportes Terrestres (SRTT), opera com processos convencionais, que já não acompanham a crescente modernização e os sistemas atuais. Estes, portanto, acabam por aumentar o tempo de espera, diminuindo a eficiência de toda a cadeia de trabalho, acrescido aos custos económicos adicionais, num mundo em que cada vez mais se espera agilidade e flexibilidade.
É através desta necessidade de modernização que surge a participação da Tetrapi SA em conjunto com o Governo Regional dos Açores para o desenvolvimento de uma plataforma de gestão de contraordenações, que visa a agilização das tarefas desempenhadas pela SRTT.
A digitalização do procedimento de contraordenação reforça, não só, o apostar do Governo Regional dos Açores na sua modernização, mas também na melhoria significativa da vida dos cidadãos da região, promovendo um método de trabalho mais ágil e transparente entre as diferentes entidades rodoviárias.
Este projeto envolve o desenvolvimento de uma plataforma *Web* que permitirá à subdireção a digitalização do fluxo das contraordenações leves, graves e muito graves, que atualmente são efetuadas em papel.
O facto de todo o fluxo ser tratado fisicamente implica que atualmente, uma contraordenação cometida na Ilha do Faial necessite de ser transportada para a ilha de São Miguel consumindo dias preciosos no fluxo de uma contraordenação, que com a introdução de uma solu-

ção tecnológica iria permitir o acesso imediato de uma contraordenação após a sua inserção na plataforma.

1.3 Objetivos

Este estágio teve como objetivo principal a minha participação na equipa de desenvolvimento da plataforma Sistema de Gestão de Contraordenações, a qual irá de forma decisiva, impulsionar a modernização da Subdireção, promovendo uma maior eficácia na gestão das regras do código de estrada, e permitindo uma redução dos custos administrativos e maior agilidade do fluxo dos processos de contraordenação.
Mais concretamente, participei em todo o processo de desenvolvimento da plataforma, desde o desenho das suas funcionalidades à implementação das mesmas. Destacando-se a definição e implementação de fluxos de contraordenações, e o desenvolvimento de testes unitários.
No decorrer deste projeto, devido à eficiência e experiência demonstrada no desempenhar das tarefas que compunham o plano de estágio, foram-me atribuídas novas tarefas.
Foram-me atribuídas as funções de *lead programmer* [2], da equipa de desenvolvimento da plataforma Sistema de Gestão de Contraordenações, onde fiquei encarregue de acompanhar, e aconselhar, a restante equipa na execução das suas tarefas.
Estas novas responsabilidades também incluíram a manutenção e evolução de uma segunda plataforma *web*, esta para a Direção Regional das Comunicações e da Transição Digital do Governo Regional dos Açores. Adicionalmente, desempenhei funções de orientação em duas equipas de desenvolvimento, as quais estão a participar na implementação de dois projetos internos. Nestas, acompanhei o desenho e implementação das funcionalidades das plataformas, aconselhando a direção a ser tomada e proporcionando um ambiente melhor para que as equipas pudessem desempenhar as suas funções com a melhor eficiência possível. Por fim, fui ainda responsável por supervisionar um estágio final curricular de Cursos Técnicos Superiores Profissionais (CTeSP) da Universidade dos Açores do curso Desenvolvimento de Aplicações Web. Onde acompanhei todo o percurso do estagiário na empresa, desde a sua integração e formação, até o cumprimento das tarefas que compunham o seu plano de estágio.

1.4 Organização do Documento

De modo a refletir o trabalho realizado, este documento encontra-se estruturado da seguinte forma:

1. O primeiro capítulo – **Introdução** – apresenta o estágio e a sua motivação, o enquadramento para o mesmo, os seus objetivos principais e secundários, e a respetiva organização do documento;

2. O segundo capítulo – **Empresa e Planeamento** – descreve a empresa que acolheu o estágio, e as tarefas que o compõem;

3. O terceiro capítulo – **Estado da Arte** – debate o estado da arte atual no que conta à elaboração de plataformas *Web* na *framework* `Symfony`;
4. O quarto capítulo – **Tecnologias e Ferramentas** – debate as tecnologias e as ferramentas utilizadas no decorrer do estágio;
5. O quinto capítulo – **Desenvolvimento da Plataforma SGCO** – aborda o desenvolvimento concreto das tarefas propostas no plano de estágio;
6. O sexto capítulo – **Conclusão** – concluí o documento, relatando as notas finais e os principais problemas enfrentados.

Capítulo 2

Empresa e Planeamento

2.1 Introdução

A Tetrapi SA (https://tetrapi.pt) é uma empresa de renome açoriana, fundada em janeiro de 2003 na ilha de São Miguel. Esta conta com cerca de 200 colaboradores, e atua em diversas áreas de mercado e é uma referência no setor da educação, ciência e tecnologia.

2.2 História

Inicialmente sob a forma de centro de explicações, a janeiro de 2003, surge no berço do oceano atlântico a empresa açoriana Tetrapi SA, sob o lema ”Melhor Educação, Melhor Cidadania”. Crescendo a passos largos, e mudando constantemente de espaço, a Tetrapi SA adquire um conjunto de terrenos e inicia a construção do Colégio Castanheiro (figura 2.1), uma escola privada de renome.

Figura 2.1: Edifício do Colégio Castanheiro.

Não satisfeita, a Tetrapi SA coloca os seus olhos no mercado tecnológico investindo na abertura da unidade Creative Solutions, que atualmente possui os seus escritórios na sua sede (figura 2.2) no coração de Ponta Delgada.

Figura 2.2: Escritórios na sede da Tetrapi SA.

Atualmente a Tetrapi SA, possui vinte e dois anos de história, e o seu compromisso com a transformação digital ao serviço da cidadania, da eficiência e da integridade pública, são reconhecidas nacionalmente pelas distinções nacionais e setoriais, das quais se destacam, **10 Melhores Pequena ou Média Empresa (PME) do setor Educacional** e ***Top* 5 das Melhores PME nacionais**.
A Tetrapi SA, apresenta hoje um portefólio constituído por:

1. Colégio Castanheiro;
2. Entidade Formadora;
3. Centro de Explicações;
4. Creative Solutions;
5. Ciência Divertida;
6. Tetra Fun Party;
7. Castanheiro Fitness Center;
8. Conceitos + Ideias.

2.3 Planeamento

Após reuniões com a empresa e com o cliente para levantamento de requisitos, foi elaborado um plano de estágio que fosse conciso nas suas tarefas e no seu domínio, assim, as tarefas que compõem o plano de estágio são maioritariamente relacionadas à gestão das contraordenação grave/muito grave. A figura 2.3 ilustra o diagrama de Gantt e a planificação do estágio.

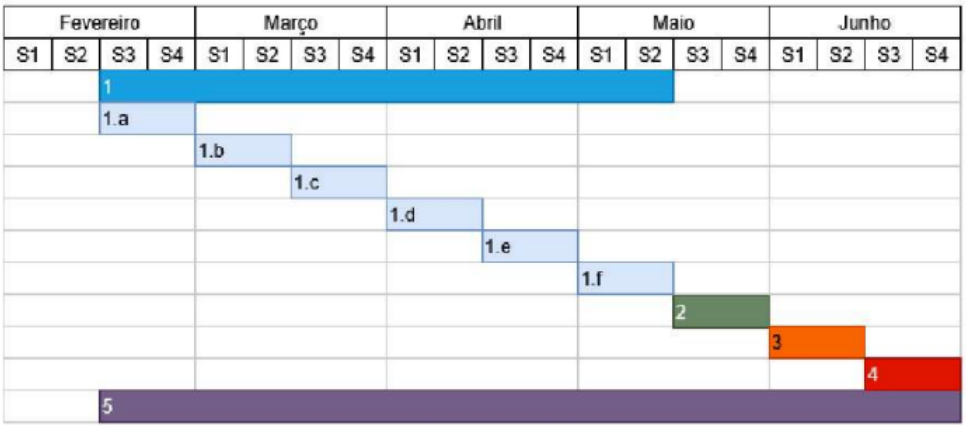

Figura 2.3: Diagrama de Gantt das tarefas do estágio.

O plano do estágio inicial é constituído pelas seguintes tarefas:

1. Desenho e implementação do fluxo do processo grave/muito grave;
   (a) Implementação da etapa Decisão;
   (b) Implementação dos Pagamentos;
   (c) Implementação da etapa Cumprimentos;
   (d) Implementação da etapa Recurso;
   (e) Implementação da etapa Execução;
   (f) Implementação dos Pedidos.
2. Implementação da funcionalidade de agrupamento de processos;
3. Unificação do fluxo dos processos grave/muito grave com o fluxo dos processos leves;
4. Definição e implementação do fluxo do processo leve agravado;
5. Implementação de testes unitários.

Para além das tarefas que compõem o plano de estágio, foram também desempenhadas outras tarefas, nomeadamente a orientação da equipa de desenvolvimento da plataforma SGCO, a orientação de duas equipas de desenvolvimento de outros projetos, a orientação de um estágio curricular, e a manutenção de uma outra plataforma *web*.
Para além das tarefas principais anteriormente listadas foram também desempenhadas as seguintes tarefas:

1. Desenho das funcionalidades da plataforma SGCO;
2. Orientação da equipa de desenvolvimento da plataforma SGCO;
3. Orientação de duas equipas de desenvolvimento de outros projetos;
4. Orientação de um estágio curricular da Universidade dos Açores;
5. Manutenção e evolução de uma plataforma *web*.

Capítulo 3

Estado da Arte

3.1 Introdução

A gestão de contraordenações é uma tarefa comum e desempenhada em todos os países que possuem um código de estrada. Em Portugal continental, esta tarefa é desempenhada pela Autoridade Nacional de Segurança Rodoviária (ANSR)(https://www.ansr.pt) que possui uma plataforma interna para a gestão de contraordenações. Por questões de segurança, a ANSR optou por não partilhar informações sobre o funcionamento da sua plataforma à empresa para auxiliar o desenvolvimento da plataforma *web* SGCO.

Este capítulo aborda a metodologia ágil, arquiteturas de plataformas *web*, e a implementação de funcionalidades comuns em plataformas *web* na *framework* `Symfony`.

3.2 Metodologia

A metodologia utilizada pela empresa é o *scrum* [3], um modelo ágil de gestão de projetos. A metodologia *scrum* foca-se na velocidade de resposta ao cliente, através de um acompanhamento contínuo que visa a validação de módulos para uma evolução incremental do produto.

Na metodologia *scrum*, o produto é dividido num conjunto de tarefas agrupadas em módulos. Estes módulos são desenvolvidos e aprovados gradualmente em reuniões, *sprints*, em intervalos de uma a quatro semanas. A equipa semanalmente reúne-se para debater o estado atual do produto e a sua evolução, não impedindo *check-ups* diários. A figura 3.1 ilustra esta metodologia.

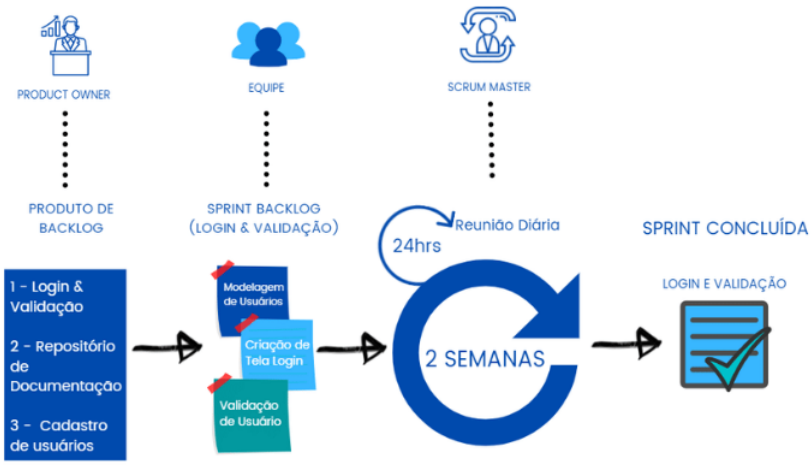

Figura 3.1: Exemplificação da metodologia *scrum* [1].

3.2.1 Pilares do *Scrum*

Esta metodologia assenta em três pontos principais para o seu bom funcionamento [4]:

- Transparência;
- Inspeção;
- Adaptabilidade.

3.2.1.1 Transparência

A transparência é o coração da metodologia *scrum*, que se manifesta numa boa comunicação entre todos os intervenientes do projeto, quer entre a equipa de desenvolvimento quer entre equipas. Um ambiente que eleve a colaboração entre os seus membros em vez da implementação de burocracias, remove passos adicionais desnecessários, agilizando assim o desenvolvimento do projeto.

3.2.1.2 Inspeção

O segundo pilar da metodologia *scrum* é a inspeção, que consiste na constante revisão e avaliação do estado to projeto. Membros e equipas são capazes de identificar irregularidades entre os requisitos pedidos e as suas implementações, podendo assim ser efetuada um correção da trajetória do projeto o mais cedo possível. Estas introspecções são normalmente realizadas durante as reuniões diárias/semanais internas pelas diferentes equipas.

3.2.1.3 Adaptabilidade

A adaptabilidade é o último principio desta metodologia, e é inseparável da inspeção 3.2.1.2. Pois, após uma inspeção da evolução e estado do projeto, pode ser necessário a adaptação do mesma face a estas alterações.
A resposta pode ir desde a alteração das tarefas planeadas, alteração da planificação dos *sprints*, ou a alteração de funcionalidades implementadas face ao levantamento de novas informações por parte do cliente. Mais do que a inspeção e transparência, a adaptabilidade é a chave central para o sucesso da metodologia *scrum*.

3.2.2 Pontos Fracos da Metodologia *Scrum*

Como é inerente de qualquer metodologia de desenvolvimento, a metodologia *scrum* possui riscos. Nesta metodologia os riscos podem surgir em diversas alturas do desenvolvimento.
O principal risco desta metodologia surge quando o cliente não possui uma visão clara das suas necessidades. Dado o cliente ser uma figura mais presente na fase de desenvolvimento, um cliente que não assente bem as suas necessidades, alterando-as constantemente, obriga a constante adaptabilidade e retrocesso no desenvolvimento da plataforma. Quer através de novos requisitos, pela alteração completa de funcionalidades ou até requisitos que contrariem outros já implementados.

Outro risco abraça-se a um dos seus pontos fortes, a autonomia da equipa. Por vezes a metodologia *scrum* é utilizada para mitigar problemas de gestão da sua equipa, tentando para isso substituir este papel fundamental pelos seus elementos. Porém a inexistência de um bom gestor inibe a boa planificação das tarefas do projeto, e consequentemente a sua evolução.

3.2.3 Pontos Fortes da Metodologia *Scrum*

Um ponto forte desta metodologia assenta na sua capacidade de adaptabilidade. Por melhor que corra a fase de levantamento de requisitos e a sua planificação, ao longo da fase de desenvolvimento o cliente irá apresentar novas informações que podem se refletir em alterações de lógica e fluxo. Através do acompanhamento constante da evolução do projeto por parte do cliente, estas alterações são mitigadas o mais cedo possível, tendo um impacto negativo menor na fase de desenvolvimento. Outro ponto forte, é a colaboração e a transparência entre as diferentes equipas, o que oferece um ambiente mais agradável e inclusivo de trabalho.

3.2.4 Abordagem Utilizada

No projeto SGCO, foi utilizada a metodologia *scrum*, onde possuíamos reuniões semanais internas com toda a equipa de desenvolvimento, realizadas na plataforma de comunicação *MIM*, e reuniões de acompanhamento com o cliente a cada duas semanas para validação do trabalho desenvolvido, na plataforma de comunicação *Microsoft Teams*. Para além destas reuniões, os membros da equipa de desenvolvimento comunicavam entre si diariamente. A figura 3.2 ilustra a calendarização destas reuniões.
Foi utilizada a plataforma de gestão de tarefas *Trello* para realizar a distribuição das tarefas entre os membros da equipa de desenvolvimento, e o *Figma* para consultar os *mockups* realizados pela equipa de *design*.

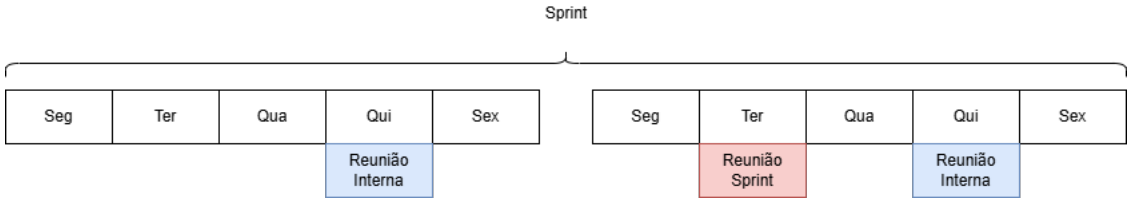

Figura 3.2: Exemplo da planificação de um *sprint* no projeto SGCO.

3.3 Arquitetura

Os requisitos funcionais e não funcionais de uma plataforma *web* podem variar imenso entre projetos, sendo que não existe uma arquitetura correta, mas sim um conjunto de arquiteturas aconselhadas consoante o tamanho e as dependências da mesma.
As arquiteturas de código mais populares são a monolítica e a de micro-serviços. Enquanto que a arquitetura monolítica consiste na elaboração de um único bloco de código com grandes dependências entre as suas componentes, a de micro-serviços remove estas dependências ao

dividir as diferentes componentes em *serviços* que não comunicam diretamente entre si, mas sim por protocolos.

3.3.1 Arquitetura Monolítica

A arquitetura monolítica [5] é a mais tradicional, e consiste no desenvolvimento de um único bloco de código. Nesta arquitetura (figura 3.3) as diferentes funcionalidades que constituem a aplicação são interdependentes devido ao facto de não existirem divisões claras entre o início e o fim de cada componente.

Esta arquitetura é ideal para projetos de dimensões pequeno-médias e permite um desenvolvimento inicial mais rápido em troca de um custo de manutenção mais elevado.

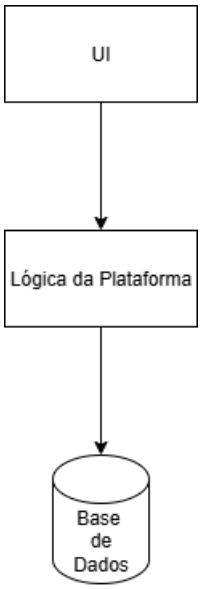

Figura 3.3: Estrutura da arquitetura monolítica.

3.3.2 Arquitetura Micro-serviços

A arquitetura de micro-serviços [6] surge como uma tentativa de resolver alguns dos problemas existentes com a arquitetura monolítica. Mais concretamente, a existência de grandes dependências entre as diferentes funcionalidades que surge com a arquitetura monolítica. Na arquitetura de micro-serviços (figura 3.4) uma plataforma é dividida em vários serviços cada um com uma responsabilidade única e que comunicam entre si via *Application Programming Interface* (API).

A divisão de responsabilidades oferecida por esta arquitetura permite a atribuição de várias equipas de desenvolvimento a um mesmo projeto, visto que estando cada uma a trabalhar no seu serviço o código desenvolvido não deverá interferir com os serviços das outras equipas.

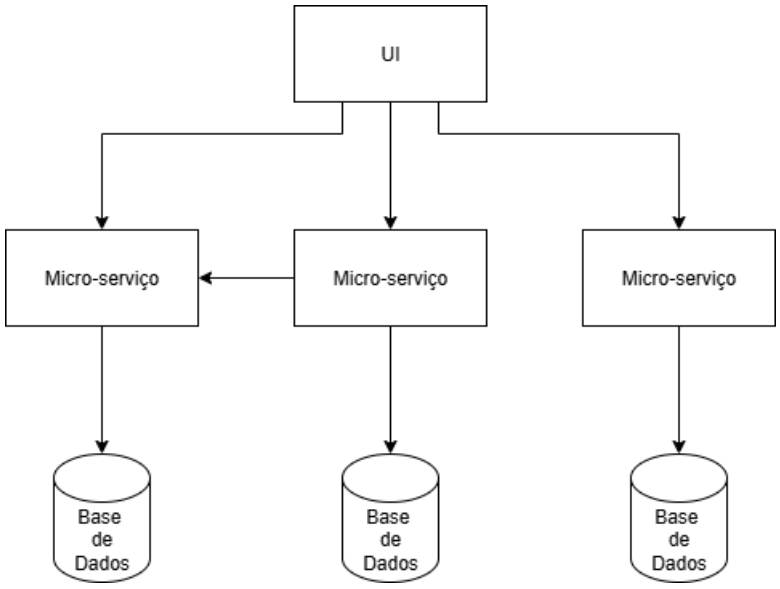

Figura 3.4: Exemplo de uma estrutura da arquitetura de micro-serviços.

Esta arquitetura é ideal para projetos de média-grande dimensão pois como referido, permite o desenvolvimento contínuo de múltiplas equipas reduzindo consideravelmente os seus ciclos de desenvolvimento. Também fornece normalmente uma manutenção mais ágil pois com a divisão de responsabilidades a alteração de um serviço não impacta o funcionamento da restante plataforma.

3.3.3 Monolítica ou Micro-serviços

Ambas as arquiteturas possuem as suas vantagens e desvantagens sendo ambas boas escolhas desde que adequadas à dimensão do projeto e da equipa de desenvolvimento [7].

A arquitetura monolítica é a mais comum, e nesta a aplicação é desenvolvida na forma de um único bloco de código. Esta unificação permite à equipa um ambiente de desenvolvimento mais simplificado e oferece uma melhor performance. Porém, com a evolução da aplicação e o seu eventual crescimento, o custo de desenvolvimento de uma aplicação monolítica é exponencialmente mais elevado devido às múltiplas interligações entre as diferentes componentes que a compõem.

A arquitetura de micro-serviços exige um custo maior na fase de desenvolvimento inicial, mas este sacrifício resulta numa fase de manutenção mais ágil e simplificada. Esta também proporciona às suas equipas de desenvolvimento um ambiente mais ágil removendo as dependências entre diferentes funcionalidades da aplicação, porém resulta numa aplicação com uma performance pior devido à adicional camada de comunicação dos seus serviços.

Assim, a arquitetura monolítica é recomendada para projetos de pequena-média dimensão, desenvolvidos por pequenas equipas de desenvolvimento. Enquanto que a arquitetura de micro-serviços é aconselhada para projetos de média-grande dimensão, desenvolvidos por múltiplas equipas.

3.3.4 Arquitetura Utilizada

A plataforma SGCO utilizou a arquitetura exigida pelo cliente, a monolítica. Esta abordagem permite inicialmente uma implementação mais rápida, sacrificando a sua futura manutenção.
A plataforma foi desenvolvida seguindo uma variação dos princípios *Model View Controller* (MVC), a *Model View Controller Service* (MVCS) [8]. Nestes princípios, existe uma divisão de responsabilidades entre a receção de um pedido e a ação que este irá despoletar.

3.4 Componentes de Uma Plataforma *Web*

Uma plataforma *web* é composta por múltiplas componentes e áreas, que vão desde os seus meios de comunicação com um sistema de bases de dados, à renderização do *HyperText Markup Language* (HTML) que irá compor a resposta ao cliente pelo servidor. Sendo a *framework* `Symfony`, uma plataforma de código aberto que segue uma filosofia de colaboração aberta à comunidade para desenvolver novos módulos, acabam por surgir inúmeras soluções para um mesmo problema.
Nesta secção será abordada a implementação das componentes mais comuns de uma plataforma *web* de gestão utilizando a *framework* `Symfony`, com foco principal nas componentes que compõem as tarefas do plano de estágio.
Uma plataforma *web* pode ser dividida na sua visão mais macro em duas áreas, o *backend* e o *frontend*.

1. *Backend* - Responsável pela receção dos pedidos ao servidor e as suas respostas, validação de dados, comunicação com a base de dados, entre outros;

2. *Frontend* - Responsável pela elaboração das páginas HTML, a sua estilização e responsividade, que munida pela informação proveniente do *backend* tenta elaborar a página mais intuitiva e responsiva para uma boa usabilidade por parte do utilizador.

3.4.1 *Backend*

O *backend* é o cérebro da plataforma *web*, sendo contido neste toda a lógica de negócio. O *backend* consiste nas ligações à base de dados, na criação e tratamento de formulários, na definição dos fluxos dos seus processos, e em todas as funcionalidades lógicas que compõem a plataforma.

3.4.1.1 Formulários

Um dos tipos mais comum de páginas que compõem plataformas de gestão, são formulários. Estes formulários são compostos por inúmeros campos e cuja informação necessita de ser validada, e migrada entre o cliente e o servidor. Para responder a esta necessidade, é utilizado o conceito de DTO [9], uma classe constituída pelos campos que compõem o formulário e as suas respetivas validações.

A implementação de formulários na *framework* `Symfony` usufrui de uma das suas componentes nativas. O excerto de código 3.1 exemplifica um DTO constituído por dois campos de texto, nome e email, acompanhados pelas suas validações. Neste exemplo, estamos a aplicar as validações de email, para garantir que o email introduzido segue um formato válido, e que o campo nome está devidamente preenchido. As validações em `Symfony` são especificadas pelo atributo `[Assert]` [10].

Excerto de Código 3.1: Exemplificação de um DTO.

```
class UserDTO
{
    #[Assert/NotBlank]
    public ?string $name = null;

    #[Assert/NotBlank]
    #[Assert/Email]
    public ?string $email = null;
}
```

Este DTO, é então utilizado por uma classe formulário (excerto de código 3.2) de modo a que o `Symfony` trate de forma automática da sua submissão e validação. De modo a alcançar um código mais estruturado é atribuído o mesmo nome ao DTO e ao formulário, de modo a facilitar a pesquisa entre as dezenas de formulários e DTO que irão constituir o projeto.

Excerto de Código 3.2: Exemplificação de um formulário em *Symfony*.

```
class UserType extends AbstractType
{
    public function buildForm(
        FormBuilderInterface $builder,
        array $options
    ): void {
        $builder
            ->add('nome', TextType::class)
            ->add('nome', EmailType::class);
    }

    public function configureOptions(OptionsResolver $resolver): void
    {
        $resolver->setDefaults([
            'data_class' => UserDTO::class
        ]);
    }
}
```

3.4.1.2 Conexão à Base de Dados

A conexão à base de dados, e as suas consequentes comunicações são realizadas via *Doctrine* [11]. O *Doctrine* é responsável por toda a comunicação com o sistema de base de dados incluindo:

1. Criação e edição das tabelas da base de dados;
2. Realização de *queries* à base de dados.

A criação e edição das tabelas que compõem a base de dados é realiza via dois comandos executados no terminal do servidor. Um responsável pela criação das instruções *Structured Query Language* (SQL), que contém as alterações a serem aplicadas na base de dados, e um segundo responsável pela execução destas instruções.
O excerto de código 3.3 ilustra uma função responsável pela *query* à tabela `User`, retornando todos os utilizadores que possuírem um nome igual ao passado por argumento.

Excerto de Código 3.3: Exemplo de uma *query* via `Doctrine`.

```
public function findUsersByName(string $name): array
{
    $query = $this->createQueryBuilder('u')
        ->where('u.name LIKE :name')
        ->setParameter('name', $name);

    return $query->getQuery()->execute();
}
```

3.4.1.3 `Symfony Scheduler`

O `Symfony Scheduler` [12] é uma biblioteca `Symfony` que permite implementar `crons`, tarefas que se repetem seguindo um determinado período. Por exemplo, utilizando esta componente é possível a criação de ações diárias de manutenção, ou verificação dos conteúdos da base de dados.
O excerto de código 3.4, ilustra um exemplo de uma `cron`, que se irá repetir diariamente à meia noite e tem como função o envio dum email a cada utilizador ativo. Neste excerto é também possível verificar a utilização de expressões `cron` para a definição do período de repetição.

Excerto de Código 3.4: Exemplo de um `Cron Job` em `Symfony`.

```
#[AsCronTask('0 0 * * *')]
class SendUserEmailCommand extends Command
{
    public function __construct(
        private UserRepository $userRepository,
        private MailerInterface $mailer,
```

```
    ) {
        parent::__construct();
    }

    protected function execute(): int
    {
        $users = $this->userRepository->findAll();

        foreach ($users as $user)
        {
            if ($user->getStatus() === StatusEnum::active)
            {
                $this->mailer->send(
                    $this->emailFactory->createNotificationEmail($user)
                );
            }
        }

        return Command::SUCCESS;
    }
}
```

3.4.1.4 Definição de Fluxos

Uma plataforma por vezes necessita de realizar tarefas com um fluxo definido, constituído por diversos estados e transições. Mais do que isto, é por vezes necessário aplicar um conjunto de regras às diversas transições de um fluxo.
A *framework* `Symfony` possui uma biblioteca que visa simplificar a criação e utilização de fluxos. A biblioteca `Symfony Workflow` [13] utiliza autómatos [14] para representar os estados e transições que constituem um fluxo.
A definição de um fluxo nesta biblioteca é realizado no ficheiro de configuração `workflow.yaml`, neste é-lhe atribuído um nome e são identificados os estados e as transições que constituem o fluxo. O excerto de código 3.5 ilustra o exemplo de um fluxo constituído pelos estados *registered*, *reviewed*, *created* e *canceled* e as transições *review*, *create* e *cancel*.

Excerto de Código 3.5: Exemplo da definição de um fluxo em `Symfony Workflow`.

```
user_create:
    type: 'state_machine'
    audit_trail:
        enabled: true
    marking_store:
        type: 'method'
        property: 'place'
```

```
    supports:
        - App\Entity\UserCreate
    initial_marking: registered
    places:
        - registered
        - reviewed
        - created
        - canceled
    transitions:
        review:
            from: registered
            to:   reviewed
        create:
            from: reviewed
            to:   created
        cancel:
            from: reviewed
            to:   canceled
```

Ao fluxo pode ser associado um guarda para realizar verificações específicas por transição, por exemplo, bloquear uma transição caso o utilizador não possua a permissão adequada. O excerto de código 3.6 ilustra um exemplo de um guarda em que é realizada uma verificação na transição `review`, esta verifica se o utilizador que está a tentar realizar esta transição possui o cargo de administrador.

Excerto de Código 3.6: Exemplo da definição de um guarda de um fluxo.

```
readonly class UserCreateSubscriber implements EventSubscriberInterface
{
    public function __construct(
        private AuthorizationCheckerInterface $checker
    ) {
    }

    public function guardReview(GuardEvent $event): void
    {
        if ($this->checker->isGranted("ADMIN"))
        {
            $event->setBlocked(true);
        }
    }

    #[Override]
    public static function getSubscribedEvents(): array
    {
```

```
        return ['workflow.user_create.review' => 'guardReview'];
    }
}
```

Um *workflow* pode ser utilizado no código através da utilização da classe `WorkflowInterface`, esta permite a aplicação de transições a um fluxo. O excerto de código 3.7 ilustra um exemplo de utilização desta classe para aplicar a transição `create` do fluxo.

Excerto de Código 3.7: Exemplo da utilização de um fluxo.

```
public function create(
    WorkflowInterface $userCreateStateMachine,
    UserCreate $userCreate
): void {
    if ($userCreateStateMachine->can('create', $userCreate))
    {
        $userCreateStateMachine->apply('create', $userCreate);
    }

    ...
}
```

3.4.1.5 Geração de Documentos

Existem múltiplos pacotes de geração de documentos para a *framework* `Symfony`, sendo `KnpSnappyBundle` [15] o principal eleito devido à sua licença de *copyright*. Este pacote usufrui de *templates* `Twig`, para a geração de documentos no formato *Portable Document Format* (PDF).
A geração de documentos neste pacote é alcançada pela utilização do método `getOutputFromHtml`. O excerto de código 3.8, ilustra a utilização deste pacote para a geração de um documento PDF. Neste é utilizado o sistema de *templates* do `Twig` para se gerar um código HTML para ser convertido para formato PDF. A função `file_put_contents` [16] é a instrução final do processo de geração, e permite escrever o conteúdo PDF gerado num documento.

Excerto de Código 3.8: Exemplo de geração de documentos na *framework* `Symfony`.

```
$html = $this->twig->render("template.html.twig");

$pdf = $this->knpSnappyPdf->getOutputFromHtml(html);

file_put_contents("documento_gerado.pdf", $pdf);
```

3.4.2 *Frontend*

O *frontend* é responsável por todas as componentes visuais da plataforma.

3.4.2.1 Twig

Uma componente fundamental no *frontend* de aplicações *web* é a escolha de um motor de *templates*. Na *framework* Symfony, é recomendado a utilização do motor Twig [17], desenvolvida pela equipa do Symfony, que permite ao programador a criação de componentes e a utilização de filtros e funções que auxiliam na manutenção e desenvolvimento do HTML que irá constituir as páginas da plataforma.
O excerto de código 3.9, exemplifica a utilização do Twig. Neste é realizada uma verificação de autorização.

Excerto de Código 3.9: Excerto de código Twig.

```
{% if is_granted("ADMIN") %}
    <p> Bem vindo Administrador, {{ user.name }}!</p>
{% else %}
    <p> Bem vindo Utilizador, {{ user.name }}!</p>
{% endif %}
```

3.4.2.2 Symfony UX

O Symfony UX é um esforço conjunto da comunidade Symfony, que visa servir de ponte entre o ecossistema Javascript e a *framework* Symfony. Este é um conjunto de pacotes, que possui, por exemplo, integrações com frameworks *Javascript* tais como, React e Vue [18].
Foram utilizados dois pacotes em particular:

1. *Live Components*;
2. *Dynamic Forms*.

Live Components
Live Components [19], são componentes *Twig* com a capacidade de realizar pedidos *Asynchronous JavaScript and XML* (AJAX) [20]. Através deste pacote, é possível a criação de componentes dinâmicas que reagem a interações do utilizador sem a necessidade de recarregar a página.
Uma *Live Component* é composta por duas partes, a lógica da componente e o seu *template*.
Os excertos de código 3.10 e 3.11, ilustram respetivamente a componente lógica e o *template* de uma componente. Esta possui uma ação, que ao se clicar no botão "Gerar Número" o número ilustrado é alterado aleatoriamente no intervalo $[0, 1000]$.
Na componente lógica, é possível observar-se a utilização dos atributos LiveProp e LiveAction, o primeiro indica que o valor de uma variável é mutável, o segundo atributo é utilizado na função generate e permite indicar que uma função pode responder a pedidos AJAX.

Excerto de Código 3.10: Componente lógica de uma *Live Component*.

```
#[AsLiveComponent]
class RandomNumber
{
    use DefaultActionTrait;
```

```
    #[LiveProp]
    public int $number = 0;

    #[LiveAction]
    public function generate()
    {
        $this->number = random(0, 1000);
    }
}
```

O *template* da componente possui um `div` obrigatório que é utilizado pelo `Symfony` para se inicializar os atributos da componente, e prepará-la para a realização de pedidos AJAX.

Excerto de Código 3.11: *Template* de uma *Live Component*.

```
<div {{ attributes }}>
    <p>Número: <p>
    <p>{{ this.number }}</p>

    <button
        data-action="live#action"
        data-live-action-param="generate">
        Gerar Número
    </button>
</div>
```

Dynamic Forms

O pacote *Dynamic Forms* [21] permite ao programador, a criação de formulários dinâmicos, ou seja, formulários cuja composição dos seus campos possa ser alterada durante o seu preenchimento pelo utilizador. Este pacote, funciona em conjunto com as *Live Componentes* e aproveita-se de chamadas AJAX para alcançar estas funcionalidades.

O excerto de código 3.12 ilustra um formulário com uma dependência direta entre dois dos seus campos, mais concretamente a dependência entre o campo `email` e o `hasEmail`. Este formulário apenas possui o campo `email` quando o campo `hasEmail` estiver marcado.

Excerto de Código 3.12: Exemplo de um formulário dinâmico.

```
class AddUser extends AbstractType
{
    public function buildForm(
        FormBuilderInterface $builder,
        array $options
    ): void {
        $builder =  new DynamicFormBuilder($builder);

        $builder->add('hasEmail', CheckboxType::class, [
```

```
            'label' => 'Utilizador tem Email?'
        ]);

        $builder->addDependent(
            'email',
            'hasEmail',
            function(DependentField $field, bool $hasEmail) {
                if (!hasEmail) {
                    return;
                }

                $field->add(EmailType::class, [
                    'label' => 'Email',
                ]);
            }
        );
    }
}
```

3.5 Conclusões

Este capítulo abordou o estado da arte na implementação das componentes mais comuns de plataformas de gestão *web*, com recurso à *framework* `Symfony`, também abordou a metodologia utilizada pela empresa e algumas arquiteturas de código. No próximo capítulo serão abordadas as tecnologias e ferramentas utilizadas no decorrer do estágio.

Capítulo 4

Tecnologias e Ferramentas Utilizadas

4.1 Introdução

Este capítulo irá descrever as diversas tecnologias e ferramentas utilizadas no decorrer do estágio. Desde as tecnologias utilizadas no desenvolvimento da plataforma *web*, às ferramentas utilizadas para gestão da equipa e colaboração.

4.2 Tecnologias *Backend*

No desenvolvimento do estágio foram utilizadas tecnologias para realizar tarefas de *backend* que constituem o plano de estágio.

4.2.1 Linguagem `PHP`

Inicialmente lançada em 1995, `PHP` [22] é uma língua de programação muito popular na área de desenvolvimento *web* [23], possuindo duas das *frameworks* mais populares na área, `Laravel` e `Symfony`. Atualmente na versão 8.4, a equipa responsável pela sua manutenção e evolução têm apostado na sua modernização via implementação de funcionalidades há muito desejadas, e já presentes noutras linguagens de programação, tais como `enums`, atributos e propriedades.

4.2.2 *Framework* `Symfony`

Atualmente o estado da arte em *framerworks* de desenvolvimento *web* em `PHP`, o `Symfony` permite a criação de plataformas *web* mantendo o código simples e conciso. O facto desta *framework* ser construída em módulos, permite ao programador apenas a utilização das componentes que realmente necessita. Segue também uma filosofia de dar à comunidade, sendo a equipa responsável pelo `Symfony` uma das que mais contribui com projetos *open-source* no ecossistema do desenvolvimento *web* `PHP`.
O `Symfony` possui módulos de desenvolvimento *web* que vão desde a autenticação, autorização, construção de formulários dinâmicos, ao desenvolvimento de API.

4.2.3 MySQL

MySQL é uma sistema de gestão de base de dados relacional *open-source*, lançado em 1995 e desenvolvido pela Oracle. MySQL permite o desenvolvimento de bases de dados relacionais de forma concisa e é suportada nativamente pelo `Symfony`.

4.2.4 Doctrine

O Doctrine é a *Object-Relational Mapper* (ORM) utilizada pela *framework* `Symfony`. Este permite a elaboração de bases de dados com uma abordagem de código primeiro, e é capaz de tratar automaticamente das conexões à base de dados. O seu propósito é o de realizar a ligação entre a base de dados e o código do projeto, e possui ferramentas que permitem a escrita de *queries* à base de dados de forma mais natural e com proteção à injeção de SQL [24].

4.2.5 PHPUnit

O PHPUnit é a biblioteca de testes principal no ecossistema `PHP`, e suporta desde testes unitários a testes de integração. O PHPUnit é suportado nativamente pelo `Symfony`, e apresenta uma forma simples e eficaz de desenvolver baterias de testes para garantir o bom funcionamento do código do projeto.

4.3 Tecnologias *Frontend*

Foi utilizado um conjunto de tecnologias *frontend*, na implementação das diferentes tarefas que compõem o plano de estágio.

4.3.1 Typescript

Desenvolvido e mantido pela Microsoft, o Typescript é uma extensão do Javascript. Utilizado em desenvolvimento *web*, o Typescript tenta resolver algumas das inconsistências, e problemas, que surgem com a utilização do Javascript. Através da implementação de um sistema estático de tipos, e indicação de tipos, o Typescript almeja reduzir erros com origem da utilização de valores de tipos de naturezas diferentes, por exemplo, a comparação de números e `strings`.

4.3.2 Tailwind

O Tailwind é um conjunto de classes *Cascading Style Sheets* (CSS) que tenta agilizar o processo de estilização de páginas, mantendo o poder de costumização no programador. Atualmente na versão 4, o Tailwind facilita desde as tarefas mais simples como a estilização de parágrafos, às mais complexas como a elaboração de animações.
Ao contrário de outras *frameworks* de CSS como o Bootstrap, o Tailwind não oferece um conjunto predefinido de componentes a utilizar, mas oferece ao programador a capacidade de criar as suas componentes.

4.4 Ferramentas de Gestão

Na Tetrapi SA, são utilizadas algumas ferramentas para a realização do acompanhamento e gestão dos seus colaboradores. Nomeadamente a plataforma interna de comunicações, e a

plataforma utilizada para o acompanhamento do estado das tarefas que compõe o desenho e desenvolvimento dos seus projetos.

4.4.1 Trello

Trello é uma plataforma digital que dispõe de quadros seguindo o modelo *Kanban* [25] que permitem emitir/distribuir tarefas e problemas para com a equipa. É utilizada de modo a agilizar a distribuição de tarefas, e para consulta do progresso de desenvolvimento do projeto.

4.4.2 MIM

MIM é uma plataforma digital de comunicação, desenvolvida utilizando um protocolo aberto descentralizado. A plataforma MIM é uma plataforma desenvolvida pela Tetrapi SA, e é utilizada para todas as comunicações e videoconferências internas da empresa.

4.5 Ferramentas de Colaboração

Para garantir uma colaboração mais ágil nos diversos projetos, são utilizadas algumas ferramentas que visam a facilitação na partilha de código e informações entre os colaboradores da empresa.

4.5.1 GitLab

GitLab é um serviço que permite gerir repositórios e versões de controlo do *Git*, na empresa é utilizado um servidor *self-hosted* de GitLab. Neste os projetos são organizados por domínio e a cada qual é aplicado uma *pipeline* [26] que confere e garante que o código desenvolvido cumpre com um conjunto de normas de qualidade de código, e o seu sucesso nos testes unitários.

4.5.2 GitKraken

GitKraken é uma aplicação que permite abstrair o utilizador dos comandos e ferramentas *Git*, desenvolvido pela *Axosoft e LLC*. O *GitKraken* dispõe de duas ferramentas: *Git Client* e *Glo Boards*. Os programadores conseguem facilmente seguir alterações e atualizações dos seus projetos, bem como trocar entre ramos de trabalho. A interface gráfica intuitiva do programa permite visualizar facilmente o *workflow*, removendo quaisquer dúvidas sobre o estado do projeto bem como aumentando a produtividade dos membros. Tem integração com vários sistemas de versão de controlo, tais como *GitHub*, *GitLab* e *BitBucket*.

4.5.3 Figma

O Figma é uma ferramenta *web* para o desenvolvimento de designs, que possui funcionalidades como a colaboração em tempo real e o desenvolvimento de protótipos. É utilizada para

o desenvolvimento de *mockups* de baixa e alta fidelidade. Ao ser consultado pelo programador no momento da implementação do design, oferece informações não só de espaçamentos, mas códigos de cor, fonte e outros.

4.6 Ferramentas de Desenvolvimento

No decorrer do estágio foram utilizadas algumas ferramentas de desenvolvimento de *software*. Nomeadamente uma *Integrated Development Environment* (IDE) e um sistema de contentorização. A empresa não força aos seus colaboradores a utilização de uma IDE específica ficando esta ao critério de cada trabalhador, porém, possui um conjunto de IDE aconselhadas.

4.6.1 Docker

O Docker é uma ferramenta de *containters* com origem no sistema operativo Linux [27]. Este permite a construção de *containers* e a sua distribuição. É uma ferramenta crucial em desenvolvimento de *software* visto que permite a isolação das dependências de um projeto e agiliza o processo de colaboração.

4.6.2 PHPStorm

PHPStorm é uma IDE desenvolvida pela empresa checa JetBrains. É o estado da arte em IDE na linguagem PHP e possui integração nativa com a ferramenta Docker e a *framework* `Symfony`. Para além de dispor de funcionalidades como a colaboração em tempo real e a visualização de bases de dados, dispõe também de uma panóplia de *plugins* que enriquecem as suas capacidades, e permitem a sua costumização.

4.7 Conclusões

Este capítulo descreveu de forma detalhada as tecnologias e ferramentas utilizadas durante o decorrer do estágio, para a realização de todas as suas tarefas principais e secundárias. No próximo capítulo será abordado a implementação de cada uma das tarefas principais que compõem o plano de estágio.

Capítulo 5

Desenvolvimento da Plataforma SGCO

Este capítulo irá abordar a estrutura da plataforma SGCO e os seus tipos de utilizador, tal como o desenvolvimento das tarefas que constituem o plano inicial de estágio.
Neste capítulo são utilizados termos técnicos relacionados diretamente com as funções desempenhadas pela plataforma desenvolvida, estes são:

- Processo – O processo de contraordenação;
- Auto – Auto de contraordenação, um documento oficial levantado, ou mandado levantar, por um agente da autoridade;
- Entidades – As entidades rodoviárias, mais concretamente a Polícia de Segurança Pública (PSP), Guarda Nacional Republicana (GNR) e a Polícia Municipal (PM).

5.1 Introdução

A plataforma SGCO tem como objetivo principal a gestão de processos de contraordenação, desde a sua introdução na plataforma até a sua conclusão. Uma contraordenação pode possuir um de vários fluxos, sendo o fluxo atribuído a uma contraordenação diretamente dependente do nível da infração cometida.
O nível de infração de uma contraordenação pode ser um de três: leve; grave e muito grave.
O fluxo de um processo pode ser dividido em duas partes principais, de acordo com a responsabilidade da sua gestão, a primeira gerida pelas entidades rodoviárias, e a segunda gerida pela SRTT. Todos os processos possuem a parte inicial, da responsabilidade das entidades, sendo a segunda parte, a da SRTT, apenas presente nos processos graves e muito graves. Devido ao facto dos processos graves e muito graves possuírem um fluxo idêntico estes são tratados em conjunto.
Como indicado, o fluxo destes processos é composto por múltiplas etapas, as quais são desempenhadas por diversos tipos de utilizador. Os tipos de utilizador que compõem a plataforma são os seguintes:

1. Gestor;
2. Jurista;
3. Subdiretor;
4. Secretária;
5. Administrador;

6. Entidade rodoviária;

7. Agente.

Estes cargos, são utilizados na plataforma para gerir a visibilidade do conteúdo e as ações que podem ser realizadas pelos utilizadores.

A figura 5.1 ilustra a página inicial da plataforma, que é apresenta ao utilizador após iniciar sessão na mesma. Esta página apresenta um conjunto de estatísticas, nos diversos cartões que constituem a página, e botões de rápido acesso aos seus alertas e notificações (canto superior direito da figura).

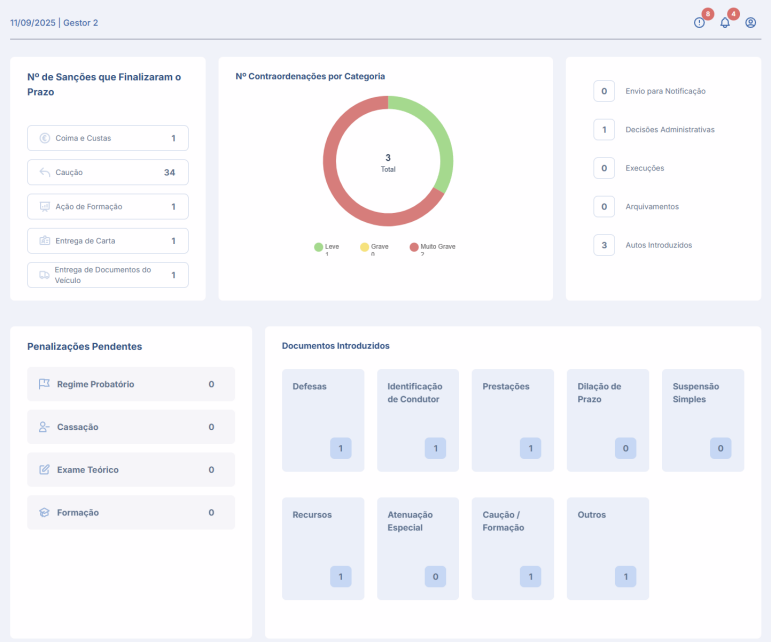

Figura 5.1: Página inicial da plataforma.

5.2 Tarefa 1 - Desenho e Implementação do Fluxo do Processo Grave/Muito Grave

Utilizando os requisitos levantados pela equipa com o cliente, foi desenhado e implementado o fluxo de uma contraordenação grave/muito grave. Porém, dado este fluxo ser extenso com múltiplas variações, o mesmo foi dividido em etapas implementadas uma por vez para validação gradual com o cliente.

O diagrama na figura 5.2 ilustra o fluxo simplificado de uma contraordenação, onde cada um dos cartões indica uma etapa do fluxo de uma contraordenação grave/muito grave. O ciclo de vida da contraordenação pode ser encerrado em dois momentos distintos, na etapa *Decisão* (caso marcado para arquivamento), ou após os estados ”Cumprimentos” e ”Pagamentos” estarem ambos concluídos.

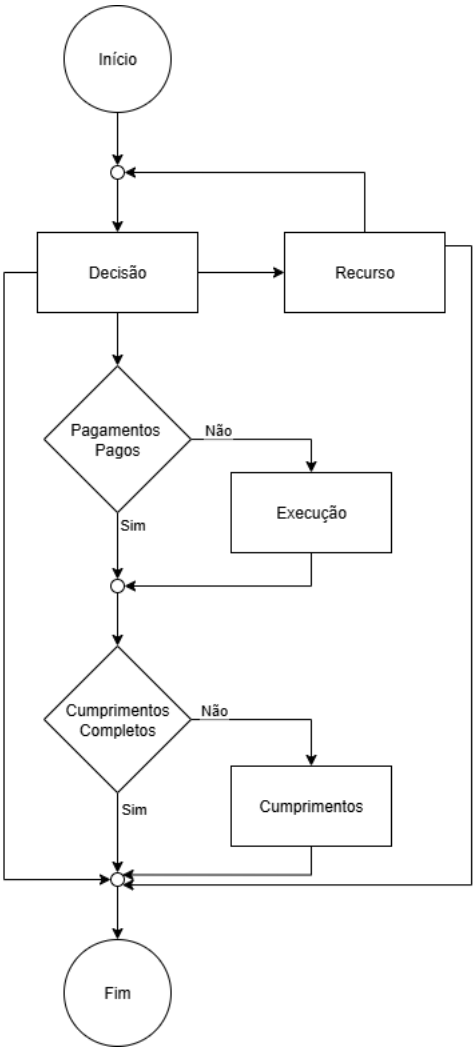

Figura 5.2: Fluxo simplificado de uma contraordenação grave/muito grave.

O diagrama da figura 5.2 é constituído pelas diversas etapas que constituem um processo de contraordenação, e os dois blocos condicionais ”Pagamentos Pagos” e ”Cumprimentos Completos”, que representam as condições ”Pagamentos” e ”Cumprimentos” respetivamente.

Numa primeira fase, foi implementado o percurso direto do fluxo ilustrado na figura anterior, onde os seus pagamentos pendentes são pagos e os cumprimentos são completos. Após validação da fase inicial, as funcionalidades ”Execução”, ”Recurso”, ”Pagamentos” e ”Cumprimentos” foram implementadas gradualmente, dando lugar ao fluxo completo de uma contraordenação grave/muito grave.

A página da contraordenação na plataforma (figura 5.3) está dividida em três secções, um cabeçalho de informações rápidas (cartão superior), um cartão de informações gerais (cartão intermédio) e as *tabs* que representam cada uma das funcionalidades da contraordenação (cartão inferior).

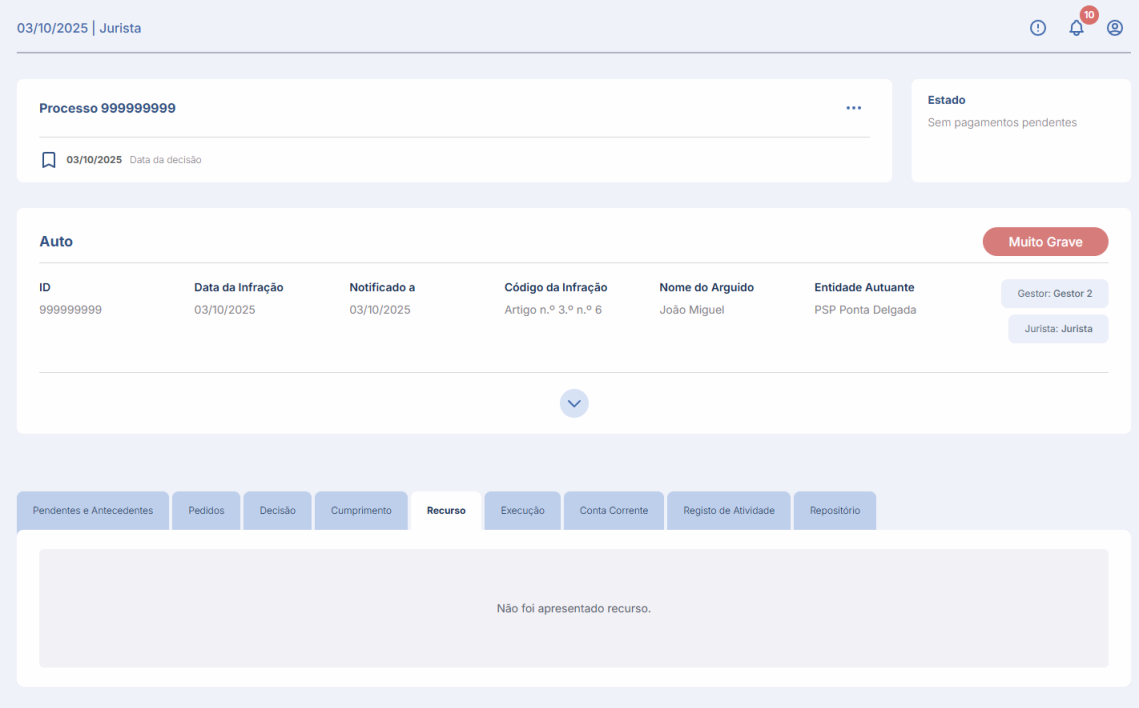

Figura 5.3: Página do perfil da contraordenação grave/muito grave.

5.2.1 Implementação da Etapa Decisão

A etapa *Decisão* é a primeira etapa do fluxo na SRTT e contempla os passos defesa; análise da defesa; decisão; sanção; assinatura da decisão administrativa e a notificação da decisão administrativa. Esta etapa é de caráter obrigatório e é responsável por munir o processo com a informação necessária para gerar os pagamentos e os cumprimentos das etapas futuras.

A enumeração seguinte indica, por ordem, o utilizador responsável por cada um dos passos que compõem esta etapa. A figura 5.4 ilustra o diagrama do fluxo desta etapa.

1. Defesa – Gestor;
2. Análise da Defesa – Jurista;
3. Decisão – Jurista;
4. Sanção – Jurista;
5. Assinatura da Decisão Administrativa – Subdiretor;
6. Notificação da Decisão Administrativa – Gestor.

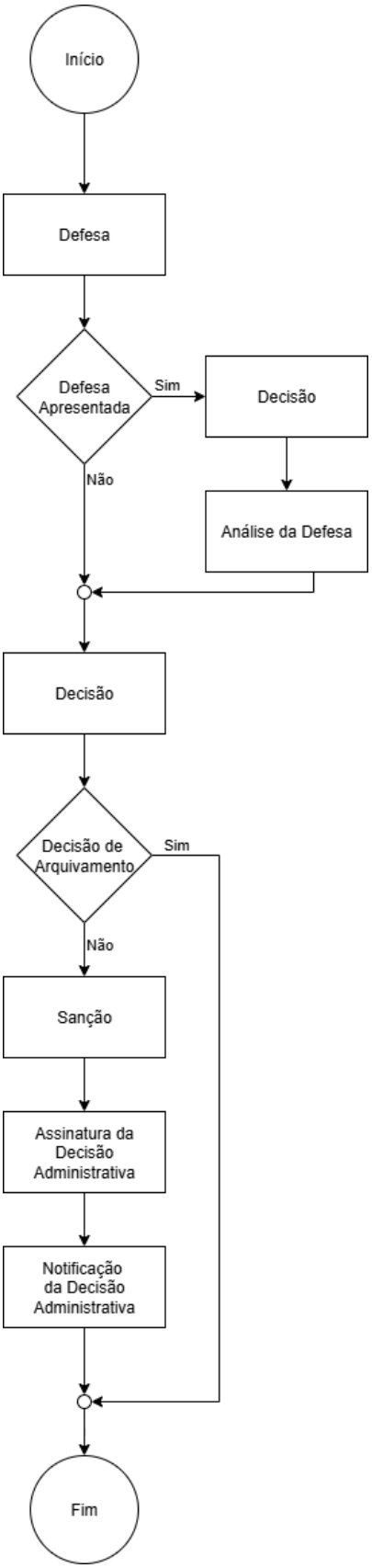

Figura 5.4: Fluxo da etapa *Decisão* de um processo grave/muito grave.

De modo a respeitar a arquitetura utilizada no projeto (secção 3.3), foi desenvolvido um con-

trolador, e o correspondente serviço, de modo a organizar o código desta etapa. Também foi criado um conjunto de tabelas responsáveis pelo armazenamento da informação dos passos decisão, sanção e análise de defesa, o diagrama da figura 5.5 ilustra as suas relações.

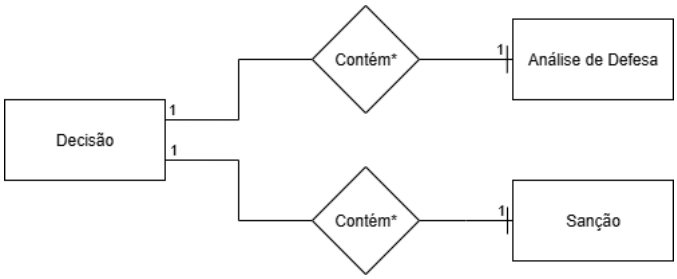

Figura 5.5: Diagrama Entidade/Associação da decisão.

A figura 5.6 ilustra a página que contém os formulários da etapa *Decisão*.

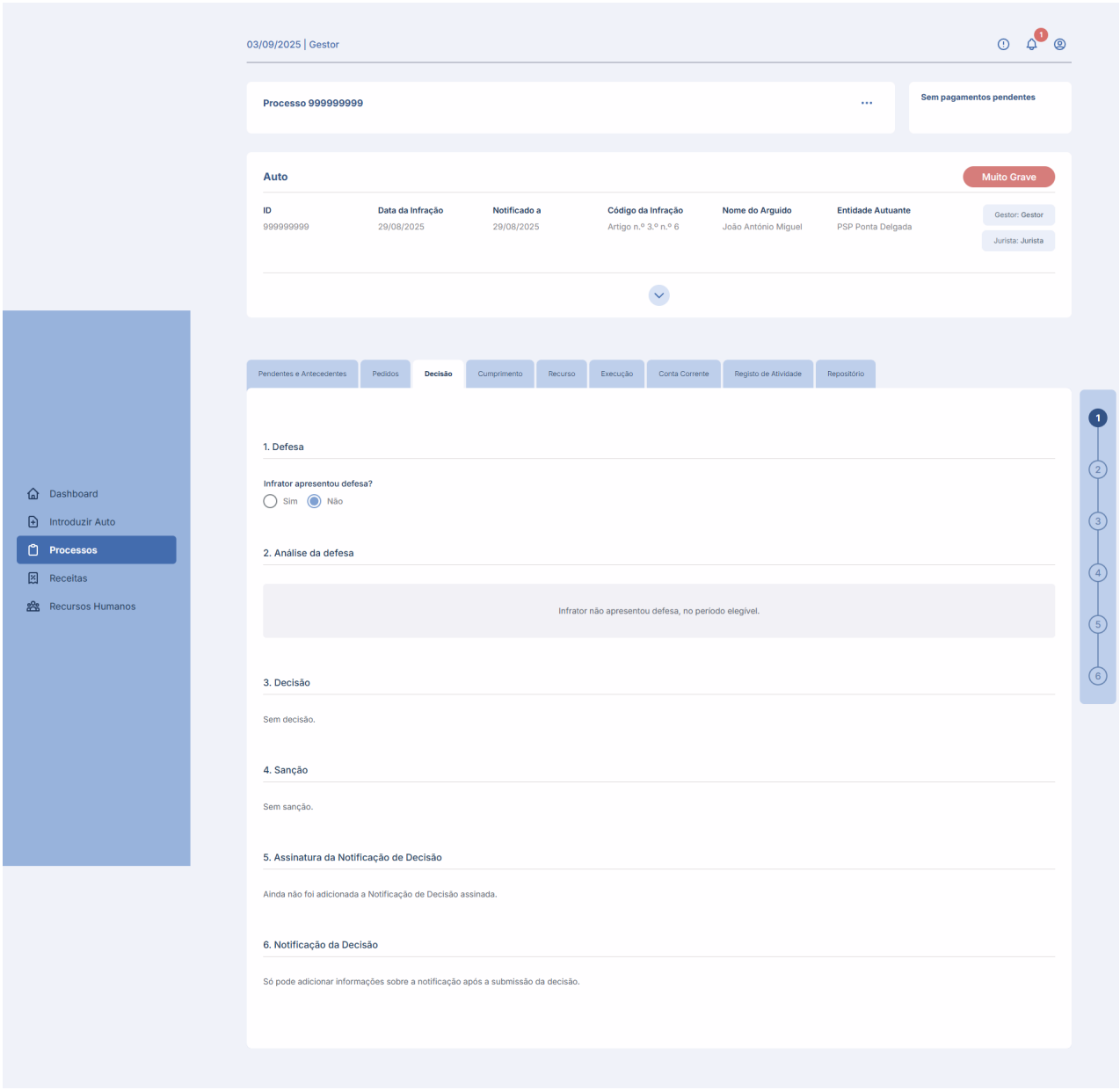

Figura 5.6: Página vazia da etapa *Decisão* de um processo grave/muito grave.

5.2.1.1 Defesa

A defesa é o primeiro passo da etapa *Decisão* e é desempenhada pelo utilizador *Gestor*. Este passo é opcional, e o formulário correspondente deve ser preenchido nos primeiros quinze dias após a receção do auto pela SRTT. A figura 5.7 ilustra o formulário que compõe este passo.

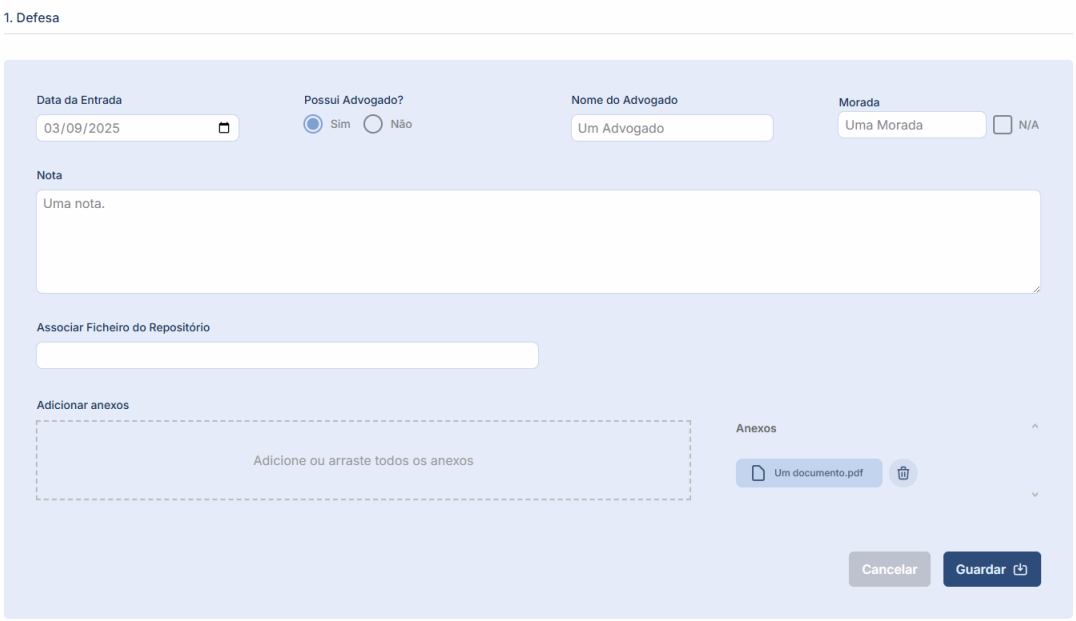

Figura 5.7: Formulário do passo defesa da etapa *Decisão* de um processo grave/muito grave.

5.2.1.2 Análise da Defesa

A análise da defesa é o segundo passo da etapa *Decisão*, e é da responsabilidade do utilizador *Jurista*. Este passo apenas ocorre em processos em que tenha sido apresentada uma defesa no período estipulado. A figura 5.8 ilustra a página responsável por este passo.

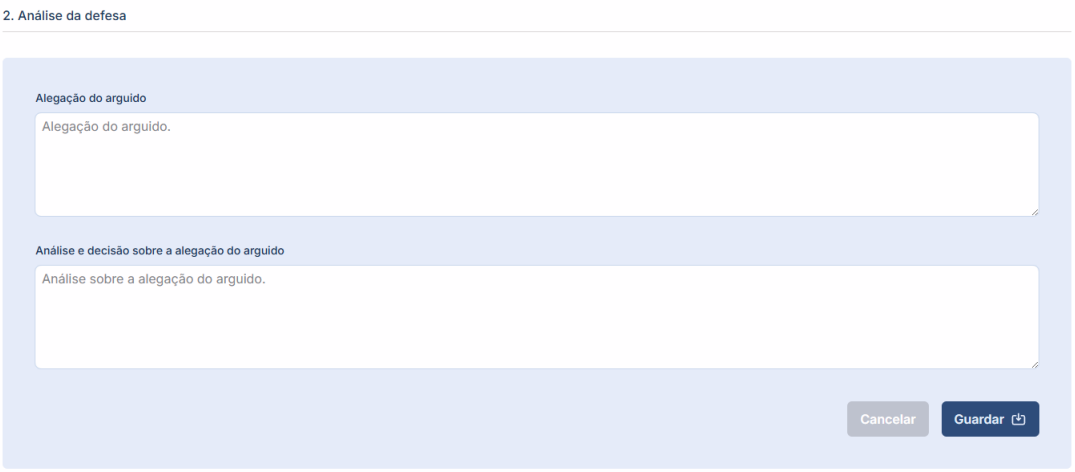

Figura 5.8: Formulário do passo análise de defesa da etapa *Decisão* de um processo grave/muito grave.

5.2.1.3 Decisão

O passo decisão, também da responsabilidade do utilizador *Jurista*, é o primeiro passo obrigatório da etapa *Decisão* e permite uma bifurcação lógica do processo. Esta ramificação é controlada pelo campo ”Natureza da Decisão” do formulário e aceita dois valores:

1. Condenatória – O arguido é condenado, e o processo segue o seu fluxo normal;

2. Não condenatória – A contraordenação é indicada como não condenatória o que implica o encerramento prematuro do processo através do seu arquivamento.

A figura 5.9 ilustra o formulário deste passo.

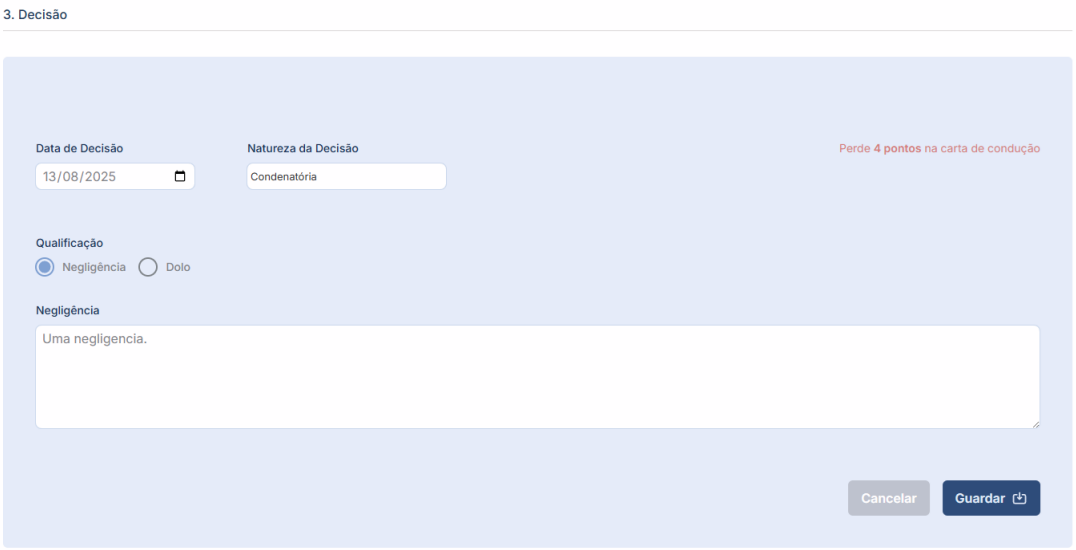

Figura 5.9: Formulário do passo decisão da etapa *Decisão* de um processo grave/muito grave.

Este passo pode ser editado pelo *Jurista* enquanto o processo não evoluir para o passo assinatura da decisão administrativa (secção 5.2.1.5).

5.2.1.4 Sanção

Da responsabilidade do utilizador *Jurista*, a sanção é o segundo passo obrigatório da etapa *Decisão*. Este passo é responsável por munir o processo das informações relativas às condições de encerramento, os ”Cumprimentos” e os ”Pagamentos”. É neste que são indicados quais os cumprimentos que irão constituir o processo, e o montante a ser pago pelo arguido.

Na submissão deste formulário (figura 5.10) a plataforma gera automaticamente o documento `Decisão Administrativa`, que deverá ser assinado pelo *Subdiretor* no passo seguinte.

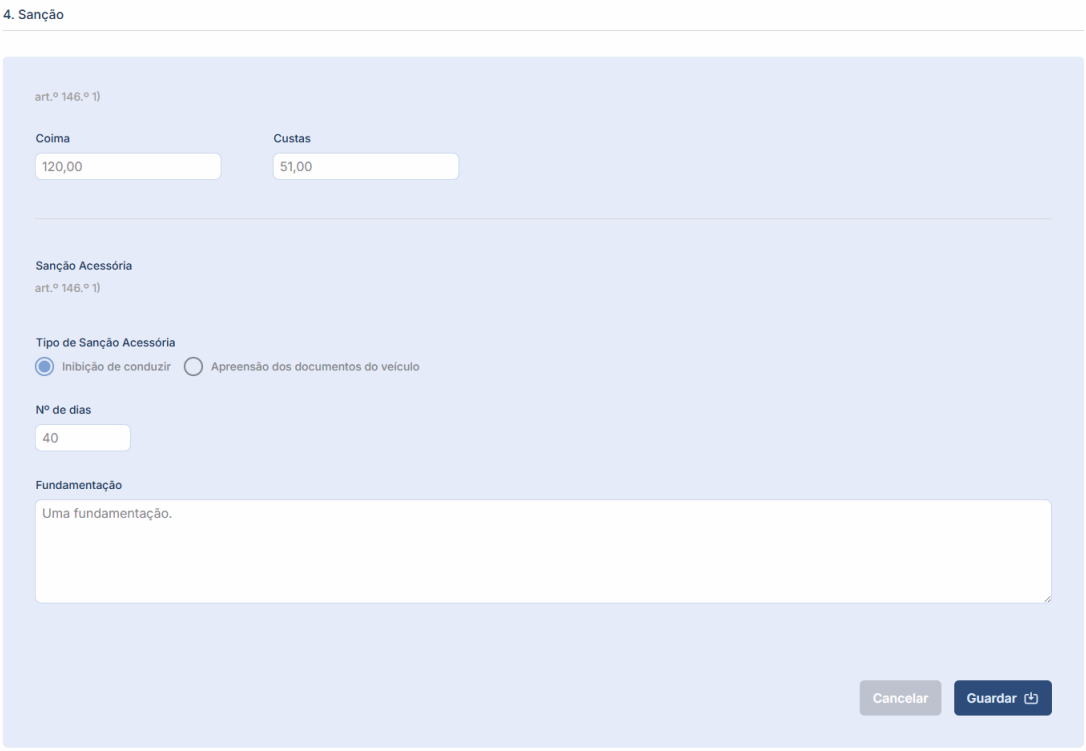

Figura 5.10: Formulário do passo sanção da etapa *Decisão* de um processo grave/muito grave.

5.2.1.5 Assinatura da Decisão Administrativa

A assinatura da decisão administrativa é o terceiro passo obrigatório da etapa *Decisão*, da responsabilidade do utilizador *Subdiretor*, e têm como função a geração do documento *Decisão Administrativa* com a assinatura do *Subdiretor*.

A geração de documentos na plataforma está unificada, estando a sua lógica presente num serviço próprio (`FileGenerateService`). A lógica do método de geração permite a criação de documentos no formato PDF seguindo um *template* `twig` (secção 3.4.2.1) como base.

A figura 5.11 ilustra a página da assinatura da decisão administrativa após a submissão do seu formulário.

Figura 5.11: Página do passo assinatura da decisão administrativa da etapa *Decisão* de um processo grave/muito grave.

5.2.1.6 Notificação da Decisão Administrativa

O passo notificação da decisão administrativa, da responsabilidade do utilizador *Gestor*, é o último passo da etapa *Decisão*. A figura 5.12 ilustra o formulário responsável por este passo.

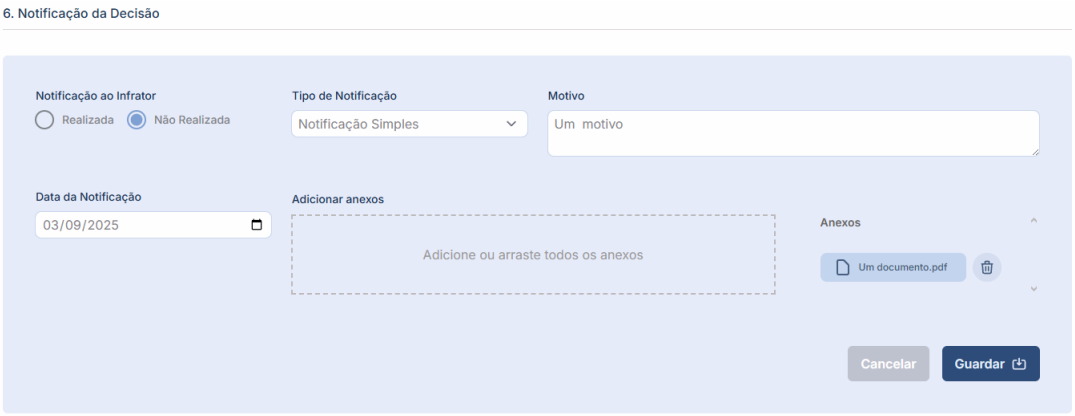

Figura 5.12: Formulário do passo notificação da decisão administrativa da etapa *Decisão* de um processo grave/muito grave.

No preenchimento do formulário deste passo o utilizador possui a escolha de marcar o campo ”Notificação ao Infrator” como ”Realizada” ou ”Não Realizada”, enquanto este formulário não for preenchido com o valor ”Realizada” o passo não é marcado como concluído. A figura 5.13 demonstra um exemplo em que este formulário foi preenchido múltiplas vezes como ”Não Realizada” antes de ser preenchida uma última vez como ”Realizada”.

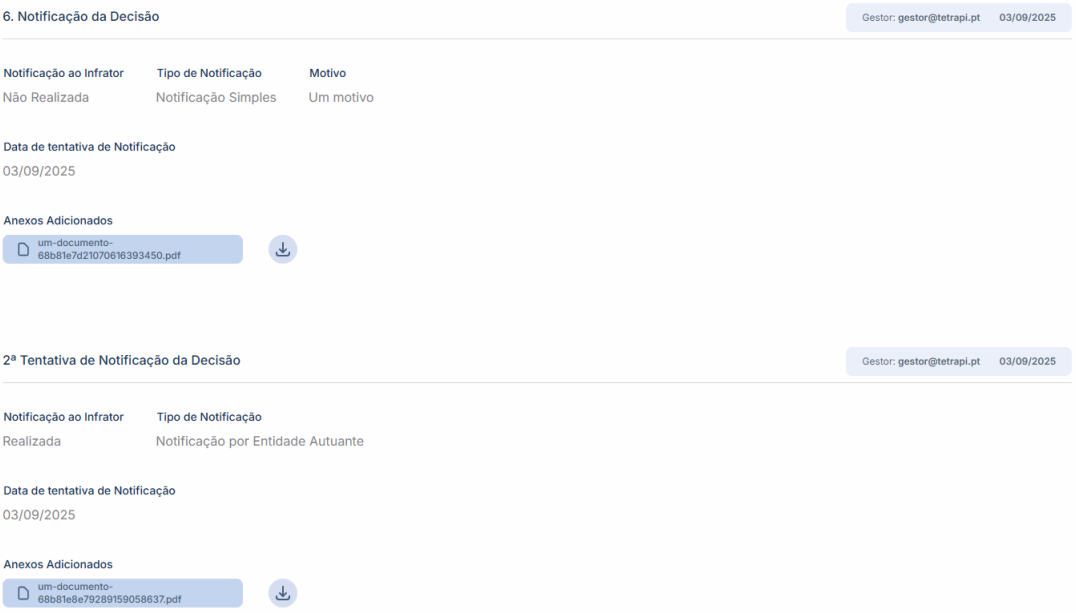

Figura 5.13: Página do passo notificação da decisão administrativa da etapa *Decisão* de um processo grave/muito grave, após a submissão de múltiplas notificações.

Após a realização da notificação da decisão administrativa, a plataforma, munida da informação do passo sanção, irá gerar os pagamentos correspondentes da coima e custas a serem pagas, e os cumprimentos a serem cumpridos. Estas duas condições são necessárias para o encerramento do processo de contraordenação.

5.2.2 Implementação dos Pagamentos

A funcionalidade *Pagamentos* é uma das funcionalidades do processo, e ocorre em simultâneo com a etapa *Cumprimentos* (secção 5.2.3), e que determina o estado dos pagamentos, uma das duas condições de encerramento do processo. A gestão dos pagamentos é realizada na plataforma pela *tab* "Conta Corrente" na página do processo de contraordenação.
A *tab* "Conta Corrente" permite ao utilizador a gestão de créditos e débitos, tal como a verificação do estado do pagamento da coima, custas e prestações. Para a realização desta funcionalidade foi implementado o controlador `CurrentAccountController` e o seu respetivo serviço `CurrentAccountService` de modo a organizar as responsabilidades lógicas desta etapa seguindo a arquitetura MVCS (secção 3.3).
Um processo pode ser pago em dois momentos distintos do fluxo do processo, antes da realização do passo defesa da etapa *Decisão*, ou após a notificação da decisão administrativa. No primeiro momento está disponível para o arguido a realização do pagamento da coima mínima atribuída à infração (segundo o código de estrada). No segundo momento, caso o arguido não tenha realizado o pagamento da coima mínima, terá que proceder à realização do pagamento de uma coima agravada e dos custos processuais.
Como o arguido pode possuir múltiplos pagamentos pendentes por processo, em que cada pagamento pendente pode ser pago através de múltiplas transferências monetárias, e as quais podem ser superiores à sua dívida, foi implementado um sistema seguindo quatro entidades distintas para contemplar todas as situações resultantes. Estas entidades são:

1. `PaymentSchedule` – Possui a informação dos pagamentos pendentes do processo;
2. `Payment` – Guarda a informação dos pagamentos realizados pelo arguido num processo;
3. `PaymentCredit` – Relaciona os pagamentos realizados aos pagamentos pendentes dum processo;
4. `PaymentDebit` – Armazena a informação do montante que deverá ser devolvido ao arguido.

Como os pagamentos pendentes possuem prazos legais, foi implementado um `cron job` (secção 3.4.1.3) que verifica diariamente os seus prazos. Quando o prazo de um pagamento expira, o estado deste pagamento evolui para `Expired`, e o estado da condição "Pagamentos" do processo passa para o valor `Canceled`. Quando a condição "Pagamentos" apresenta o valor `Canceled`, o processo evoluí para a etapa *Execução* de preenchimento obrigatório para o encerramento do processo.
A figura 5.14 ilustra o design da página conta corrente, onde é possível identificar quatro cartões informativos:

1. Fase Instrução – Ilustra ao utilizador o estado do pagamento da coima mínima do processo;
2. Fase de Pós Decisão Condenatória – Ilustra ao utilizador o estado do pagamento da coima/custas do processo;

3. Créditos – Contém a informação dos pagamentos realizados do processo, e a sua respetiva atribuição a cada um dos pagamentos pendentes;

4. Débitos – Contém a informação do débito do processo, e permite ao utilizador a revisão do débito para a sua devolução.

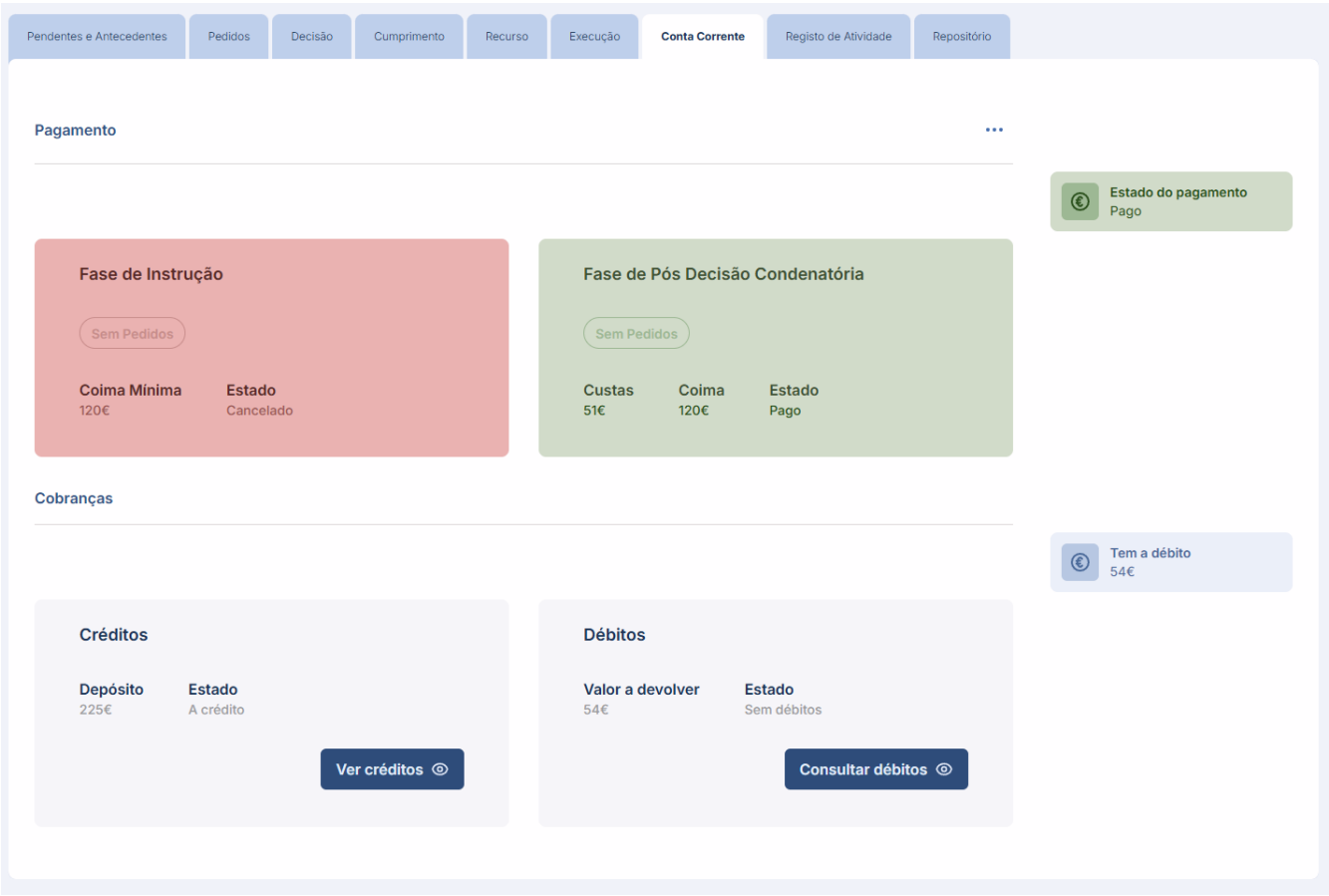

Figura 5.14: Página da conta corrente do processo grave/muito grave.

A *tab* ”Conta Corrente” possui duas variações, caso seja realizado um pedido de ”prestações” (secção 5.2.6.2) ou um pedido de ”dilação de prazo” (secção 5.2.6.3). As figuras 5.15 e 5.16 ilustram respetivamente o *design* da plataforma após a realização autorizada destes pedidos, nestas figuras é possível observar-se que o cartão ”Fase Pós Decisão Condenatória” apresenta uma cor amarela e novas informações relativas ao pedido realizado.

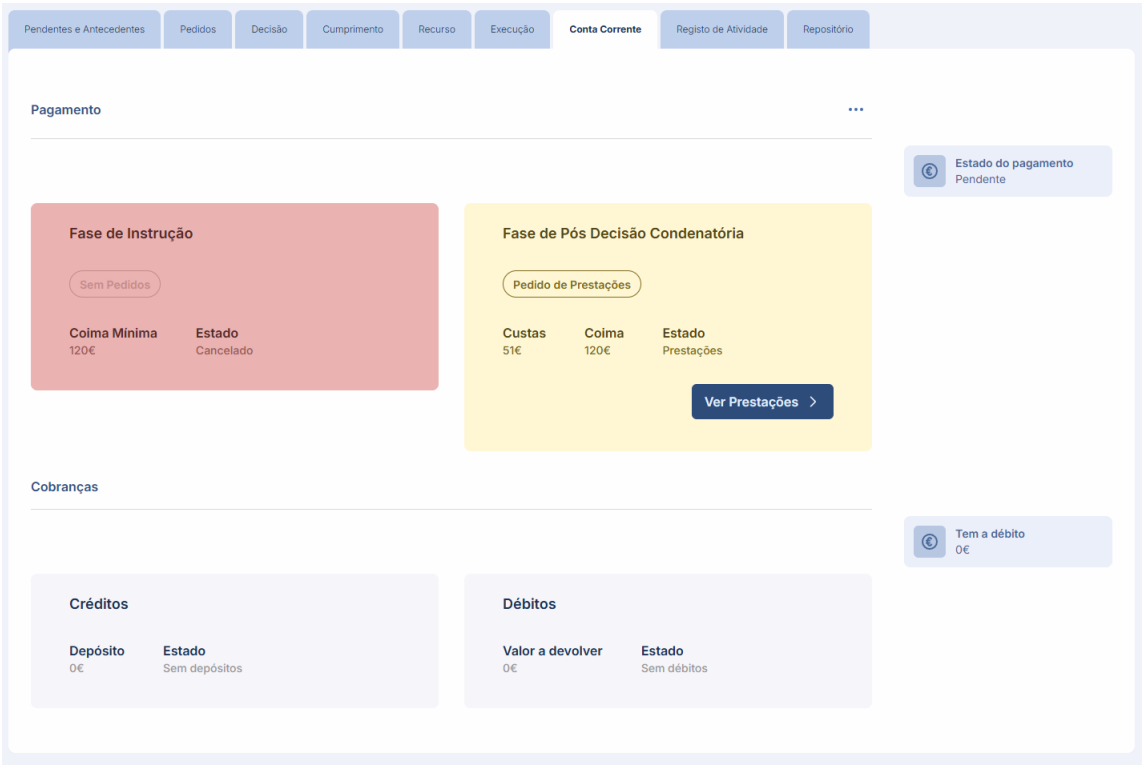

Figura 5.15: Página da conta corrente do processo grave/muito grave, quando existe um pedido de ”prestações”.

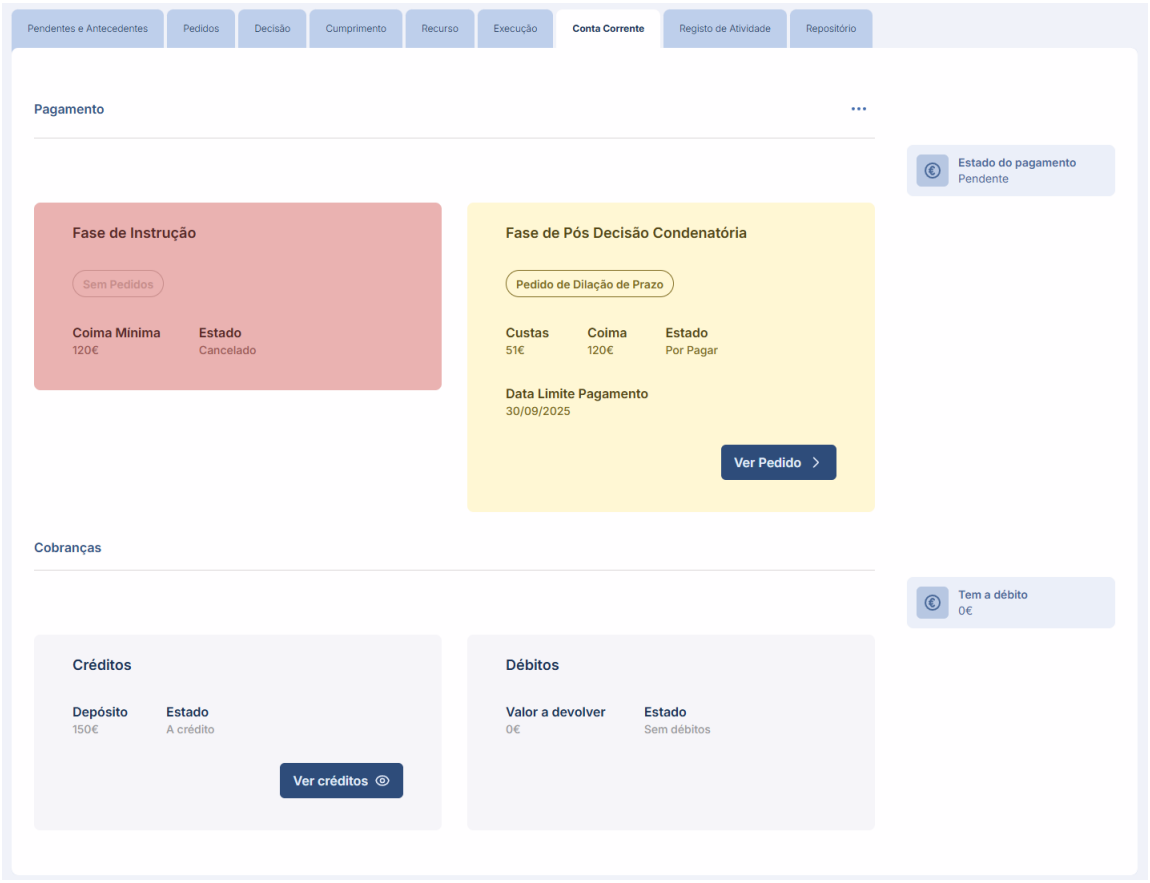

Figura 5.16: Página da conta corrente do processo grave/muito grave, quando existe um pedido de ”dilação de prazo”.

5.2.3 Implementação da Etapa Cumprimentos

A etapa *Cumprimentos* é uma etapa lógica que ocorre em simultâneo com a funcionalidade *Pagamentos* (secção 5.2.2), após a realização da etapa *Decisão*, e que determina uma das duas condições de encerramento do processo, o estado dos cumprimentos. Os cumprimentos podem ser consultados e geridos na plataforma pela *tab* ”Cumprimentos” na página do processo (figura 5.17) .

Figura 5.17: Página dos cumprimentos do processo grave/muito grave.

Os cumprimentos que compõem um processo dependem dos pedidos realizados na funcionalidade *Pedidos*, e do passo sanção da etapa *Decisão*. Porém, na situação em que o fluxo do processo segue sem a realização de qualquer pedido, são gerados dois cumprimentos pré-definidos: a ”entrega de documentos” e a ”inibição de conduzir”.
Um processo pode possuir os seguintes cumprimentos:

1. Entrega de documentos;
2. Inibição de conduzir;
3. Ação de formação;
4. Caução de boa conduta;
5. Suspensão simples.

Cada um dos diferentes cumprimentos possui um fluxo próprio para ser determinada a sua conclusão. Após a conclusão de todos os cumprimentos que compõem um processo, a condição ”Cumprimentos” evoluí para concluído, ficando o processo apenas a aguardar a conclusão da condição ”Pagamentos” (secção 5.2.2) para o seu encerramento.

5.2.3.1 Entrega de Documentos

A ”entrega dos documentos” é um dos cumprimentos pré-definidos de um processo e é responsável pela lógica que acompanha a entrega dos documentos. Estes podem ser a **carta de condução** ou os **documentos do veículo**. A figura 5.18 ilustra o formulário do cumprimento na sua fase inicial enquanto este se encontra a aguardar pela entrega dos documentos no prazo estabelecido.

Figura 5.18: Passo inicial do cumprimento ”entrega de documentos” de um processo grave/muito grave.

Este cumprimento segue o fluxo especificado no diagrama da figura 5.19. Sendo que caso não sejam entregues os documentos no prazo estabelecido, este cumprimento poderá ser enviado ao tribunal para uma execução judicial. A figura 5.20 ilustra este cumprimento concluído pelo envio para o tribunal.

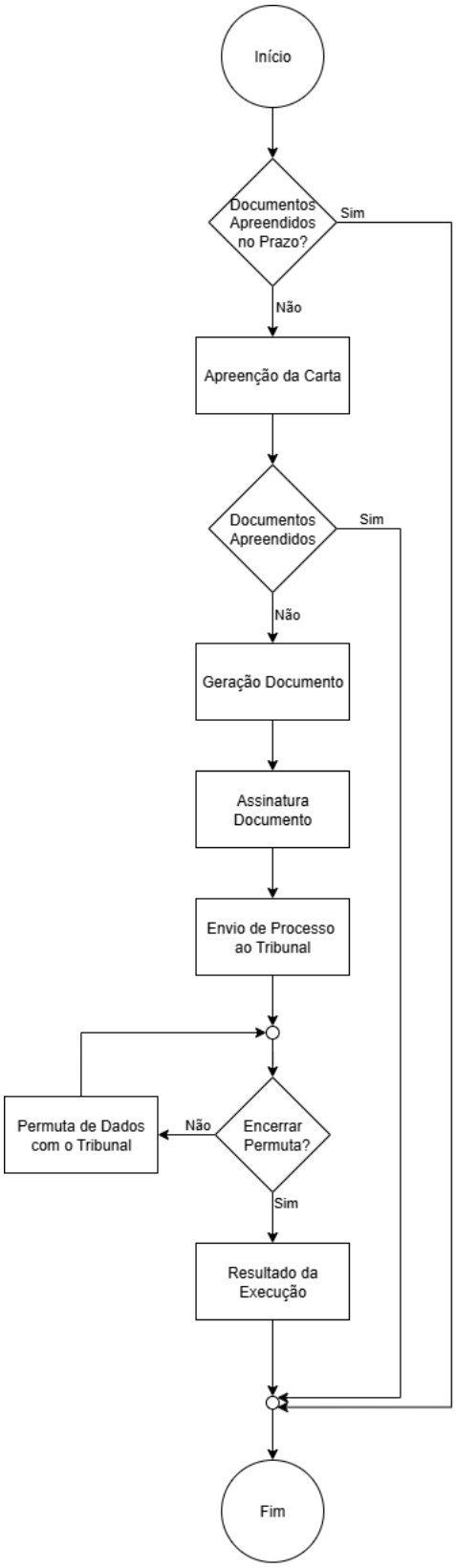

Figura 5.19: Fluxo do cumprimento ”entrega de documentos” de um processo grave/muito grave.

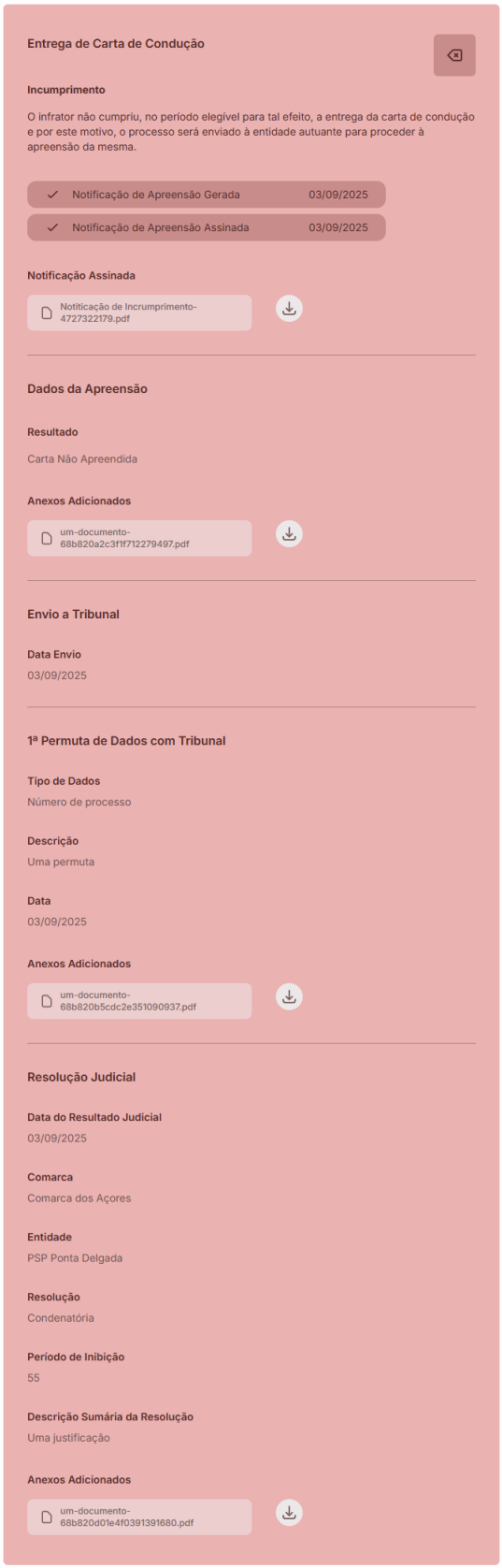

Figura 5.20: Cumprimento ”entrega de documentos” de um processo grave/muito grave, concluído via tribunal.

5.2.3.2 Inibição de Conduzir

A ”inibição de conduzir” é um cumprimento que não possui ações, sendo apenas um período estipulado no passo sanção da etapa *Decisão*.

Este cumprimento não decorre em simultâneo com os restantes, sendo que o período de inibição de condução, apenas começa a contar após a conclusão dos restantes cumprimentos do processo. Por exemplo, caso um processo possua os cumprimentos de ”entrega de documentos” e ”inibição de conduzir”, o segundo irá apenas iniciar a contagem do período de inibição após a conclusão do cumprimento de ”entrega de documentos”.

Este cumprimento é finalizado em duas situações distintas:

1. Após a finalização do período de inibição estipulado, onde o cumprimento é marcado como concluído;

2. Caso o arguido cometa uma infração durante o período de inibição de conduzir, o que iria finalizar a inibição de conduzir por cancelamento.

5.2.3.3 Ação de Formação

O cumprimento de ”ação de formação” implica a realização de pedidos (secção 5.2.6) próprios e substitui o cumprimento pré-definido da ”entrega de documentos”.

Este cumprimento possui uma fase inicial de 180 dias, em que o arguido deverá inscrever-se numa ação de formação, numa escola de condução. No caso de incumprimento deste período estipulado o processo irá gerar automaticamente um cumprimento de ”entrega de documentos”.

O cumprimento de ”ação de formação” segue o fluxo da figura 5.21, e após a sua conclusão bem sucedida o período de inibição de conduzir é desbloqueado e é iniciada a sua contagem de tempo. A figura 5.22 ilustra a página dos cumprimentos após a conclusão bem sucedida deste cumprimento.

Figura 5.21: Fluxo do cumprimento ”ação de formação” de um processo grave/muito grave.

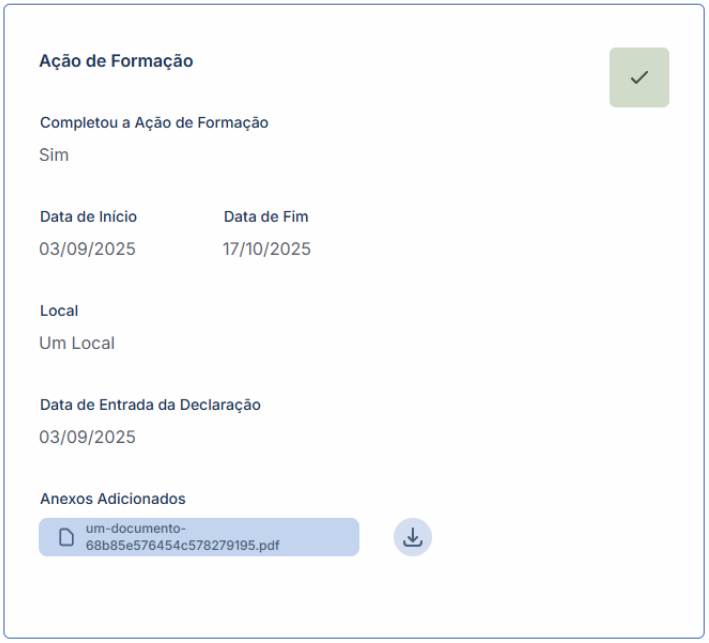

Figura 5.22: Cumprimento ”ação de formação” de um processo grave/muito grave concluído.

5.2.3.4 Caução de Boa Conduta

À semelhança do cumprimento de ”ação de formação”, o cumprimento de ”caução de boa conduta” implica a realização de pedidos (secção 5.2.6) próprios e substitui o cumprimento pré-definido da ”entrega de documentos”.
Este cumprimento possui uma fase inicial de trinta dias nos quais deve ser realizado o pagamento da caução (pelo formulário da figura 5.23), seguido de um período de retenção de 365 dias. Após a conclusão do período de retenção, a plataforma gera automaticamente um

pedido de revisão de débito correspondente ao montante da caução, que deverá então ser analisado pelo utilizador na página da conta corrente (secção 5.2.2) para devolução ao arguido.

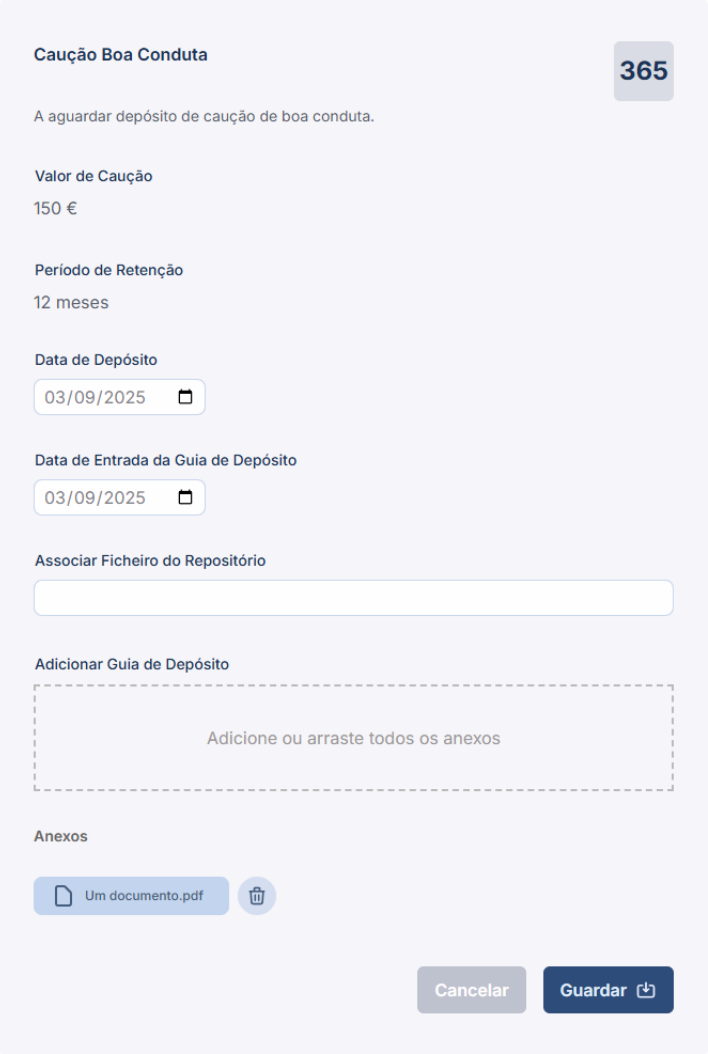

Figura 5.23: Formulário de pagamento do cumprimento ”caução de boa conduta” de um processo grave/muito grave.

O incumprimento do pagamento da caução, ou a realização de infrações rodoviárias durante o período de retenção da mesma, implica a geração do cumprimento de ”entrega de documentos”.

O cumprimento ”caução de boa conduta” segue o fluxo da figura 5.24, e após a sua conclusão bem sucedida o período de inibição de conduzir é desbloqueado e é iniciada a sua contagem. A figura 5.25 ilustra a página dos cumprimentos após a conclusão bem sucedida deste cumprimento.

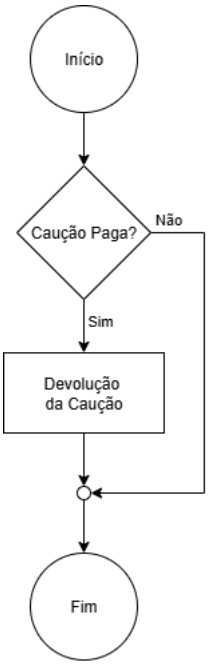

Figura 5.24: Fluxo do cumprimento ”caução de boa conduta” de um processo grave/muito grave.

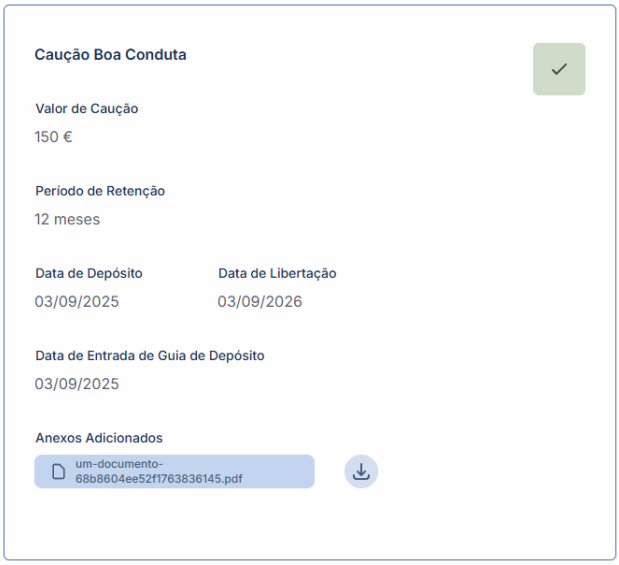

Figura 5.25: Cumprimento ”caução de boa conduta” de um processo grave/muito grave concluído.

5.2.3.5 Suspensão Simples

O cumprimento de ”suspensão simples”, à semelhança da ”caução de boa conduta”, requer a realização de um pedido próprio. Este cumprimento é o único que substitui os dois cumprimentos pré-definidos.
O cumprimento de ”suspensão simples” é finalizado em duas situações distintas:

1. Após a finalização do período de suspensão, em que o cumprimento é marcado como concluído e finalizado automaticamente;
2. Caso o arguido cometa infrações rodoviárias durante o período de suspensão;

5.2.4 Implementação da Etapa Recurso

Após a implementação, e validação, das funcionalidades *Decisão*, *Cumprimentos* e *Pagamentos* (secção 5.2.2), foi realizada a implementação da etapa *Recurso*. A realização desta etapa é opcional, não sendo necessária para o encerramento do processo de contraordenação.

Um processo de contraordenação apenas possui um recurso caso este tenha sido apresentado à SRTT nos primeiros quinze dias, após a notificação da decisão administrativa. A figura 5.26 ilustra a *tab* "Recurso" antes desta etapa ser iniciada.

Figura 5.26: Página da etapa *Recurso* do processo grave/muito grave.

O diagrama da figura 5.27 demonstra o fluxo da etapa *Recurso*, no qual é possível observar as variações que o constituem. Nomeadamente, caso o recurso seja marcado para ser enviado ao tribunal são acrescentados os passos: envio de processo ao tribunal; permuta de dados com o tribunal; e decisão judicial.

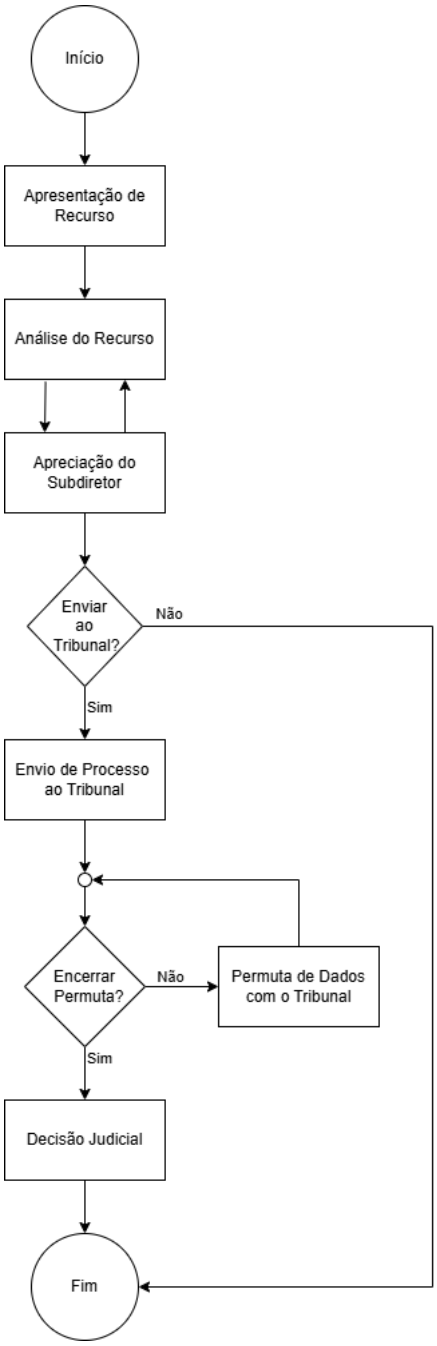

Figura 5.27: Fluxo da etapa *Recurso* do processo.

Independentemente do fluxo, que a etapa *Recurso* tome, este pode possuir um de três finais distintos. Cada um destes finais pode modificar o processo de uma forma distinta:

1. Revogação Administrativa – Nesta situação a decisão administrativa é revogada, o que implica o retrocesso do processo ao início da etapa *Decisão*, a qual terá que ser preenchida novamente para ser tomada uma nova decisão administrativa;

2. Arquivamento – Nesta situação a decisão administrativa é revogada, o que implica o retrocesso do processo ao início da etapa *Decisão*, a qual terá que ser preenchida para ser tomada uma decisão administrativa da natureza ”Arquivamento/Não condenatório”;

3. Manter Decisão – Nesta situação a decisão administrativa é mantida, o que significa que o processo não sofre qualquer tipo de alteração.

À semelhança das etapas anteriores, foi desenvolvido o controlador `AppealController` e o respetivo serviço `AppealService`, de modo a agrupar os diferentes métodos que constituem cada um dos passos da etapa *Recurso*.

5.2.4.1 Apresentação de Recurso

O passo apresentação de recurso, da responsabilidade do utilizador *Gestor*, inicia o fluxo da etapa *Recurso* e deve ser preenchida nos primeiros quinze dias após a notificação da decisão administrativa. A figura 5.28 ilustra o formulário que constitui este passo.

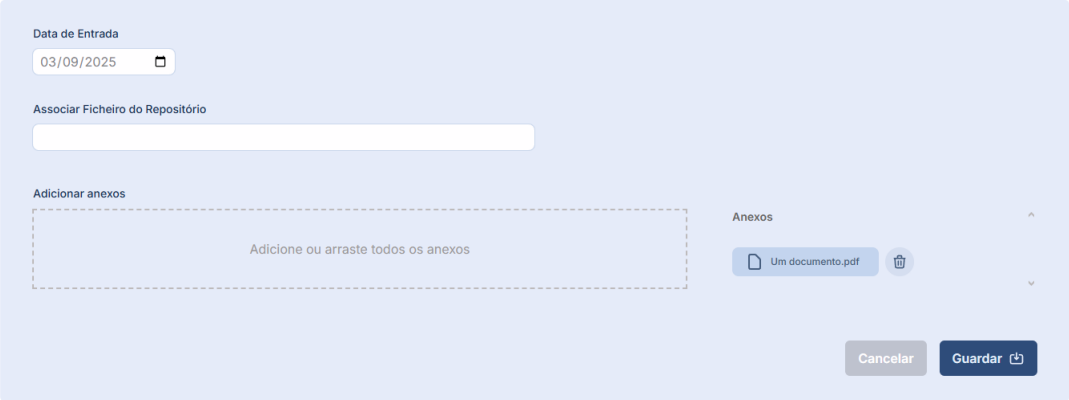

Figura 5.28: Formulário do passo apresentação de recurso da etapa *Recurso* de um processo grave/muito grave.

5.2.4.2 Análise do Recurso

A análise do recurso, da responsabilidade do utilizador *Jurista*, é o segundo passo da etapa *Recurso*, e apresenta ao utilizador a escolha do fluxo que esta etapa irá tomar. Neste passo é possível optar por revogar a decisão administrativa ou evoluir o processo para análise judicial. Independentemente da opção tomada, esta necessita de ser apreciada e autorizada pelo *Subdiretor* na etapa seguinte. A figura 5.29 demonstra o formulário responsável por este passo.

Figura 5.29: Formulário do passo análise do recurso da etapa *Recurso* de um processo grave/muito grave.

5.2.4.3 Apreciação do Subdiretor

Da responsabilidade do utilizador *Subdiretor*, o passo apreciação do subdiretor é o terceiro passo da etapa *Recurso*. Neste passo o utilizador pode optar por autorizar, ou não, a análise do recurso. Caso o utilizador opte por não autorizar, o processo irá retornar ao passo análise do recurso para que o *Jurista* realize uma nova análise, a figura 5.30 ilustra o formulário nestas condições.

Figura 5.30: Formulário do passo apreciação do subdiretor da etapa *Recurso* de um processo grave/muito grave.

Na situação em que o recurso não tenha sido designado para ser enviado ao tribunal, este passo apresenta a finalização da etapa *Recurso* sendo os efeitos da opção tomada refletidos no processo.

5.2.4.4 Envio de Processo ao Tribunal

O passo envio de processo ao tribunal, da responsabilidade do utilizador *Gestor*, está apenas presente no fluxo da etapa *Recurso* caso este tenha sido marcado para envio ao tribunal, e apresenta um formulário muito simples com o único intuito do registo da data de envio.

5.2.4.5 Permuta de Dados com o Tribunal

O passo permuta de dados com o tribunal, da responsabilidade do utilizador *Gestor*, apresenta um passo opcional da etapa *Recurso* com o intuito de registar as comunicações realizadas entre a SRTT e o tribunal. A figura 5.31 demonstra o formulário que constitui este passo.

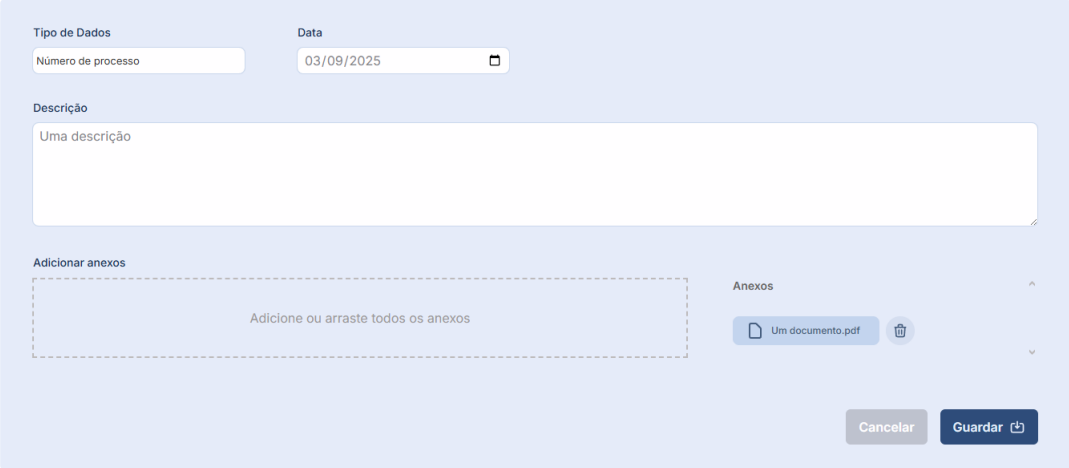

Figura 5.31: Formulário do passo permuta de dados com o tribunal da etapa *Recurso* de um processo grave/muito grave.

O processo apenas evolui para o passo seguinte quando o utilizador optar por encerrar a permuta de dados com o tribunal, se o utilizador não indicar o mesmo o processo permite a adição de qualquer quantidade de permutas de dados. A figura 5.32 ilustra uma situação em que foram realizadas múltiplas permutas de dados.

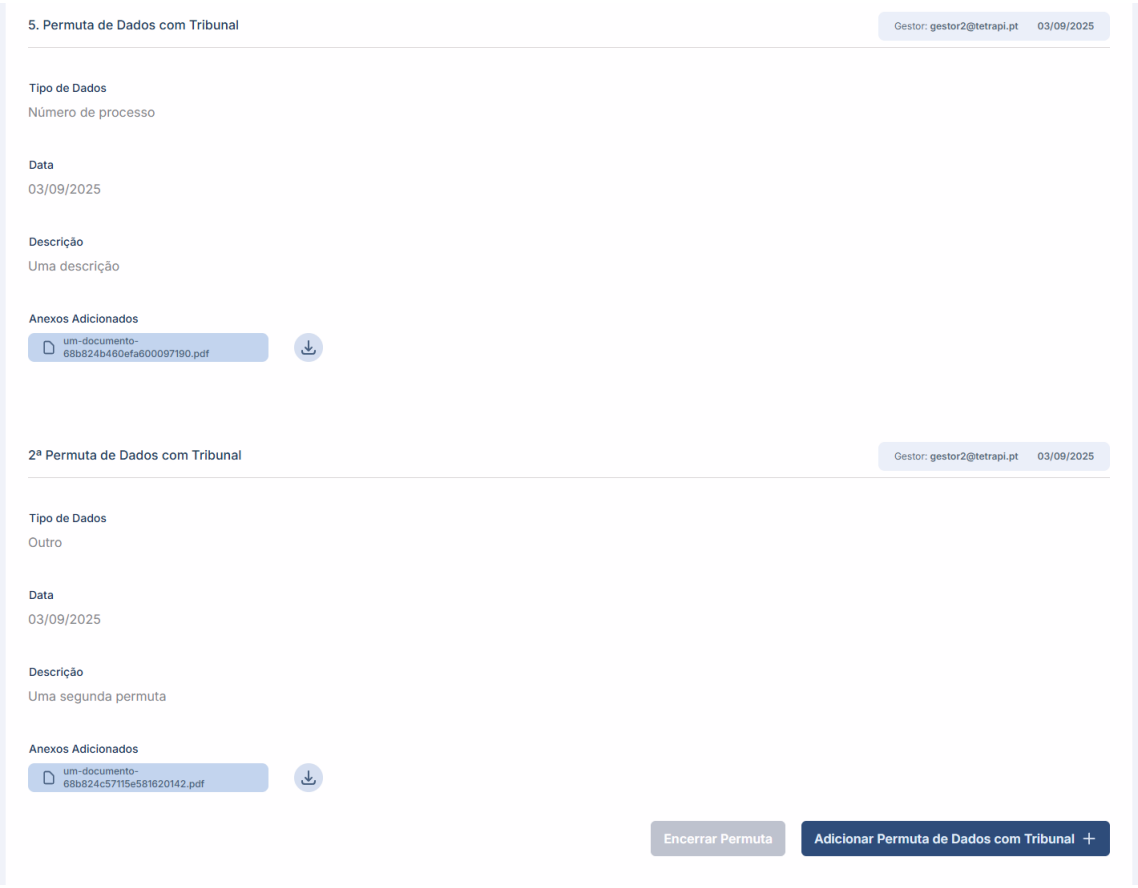

Figura 5.32: Página da etapa *Recurso* de um processo grave/muito grave, após a submissão de múltiplas permutas de dados.

5.2.4.6 Decisão Judicial

O passo decisão judicial, da responsabilidade do utilizador *Gestor*, apresenta o passo final da etapa *Recurso*, e tal como indicado na secção 5.2.4, irá aplicar um conjunto de modificações ao fluxo do processo de contraordenação, dependendo da opção escolhida neste formulário (figura 5.33).

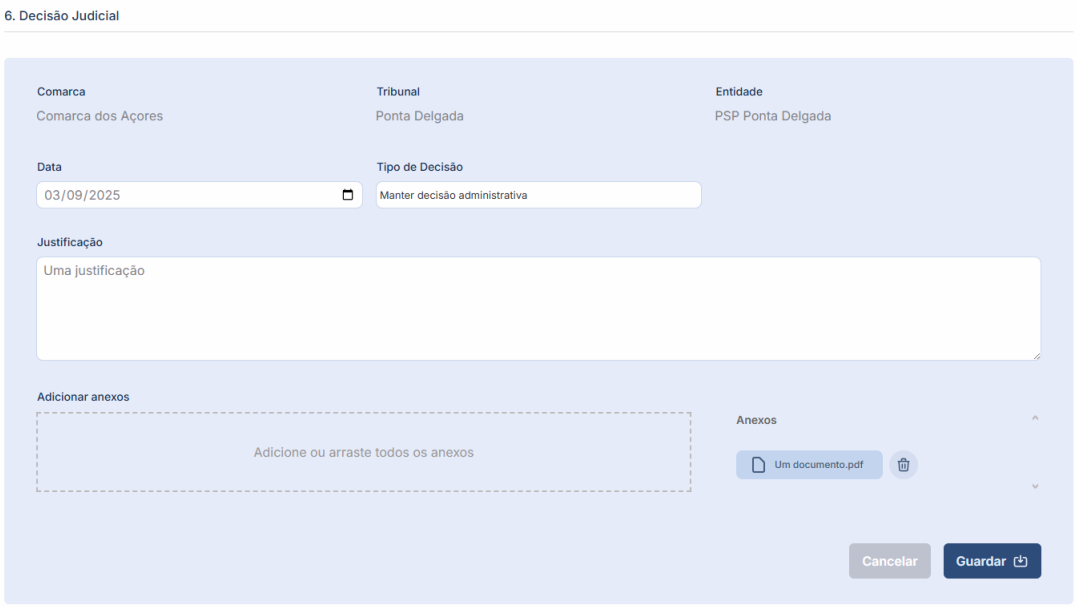

Figura 5.33: Formulário do passo decisão judicial da etapa *Recurso* de um processo grave/muito grave.

5.2.5 Implementação da Etapa Execução

Após a implementação da etapa *Recurso*, foi implementada a etapa *Execução*. A realização desta etapa é apenas obrigatória em determinadas condições. Estas são:

1. Não pagamento total/parcial da coima e/ou custas;

2. Não pagamento total/parcial das prestações.

Caso o processo se encontre numa destas situações, a etapa *Execução* torna-se obrigatória para o encerramento do processo.

Esta etapa, é composta por seis passos, os quais são acessíveis na plataforma pela *tab* "Execução" na página do processo, e seguem o fluxo da figura 5.34. O fluxo desta etapa não possui grandes ramificações, possuindo apenas um passo opcional.

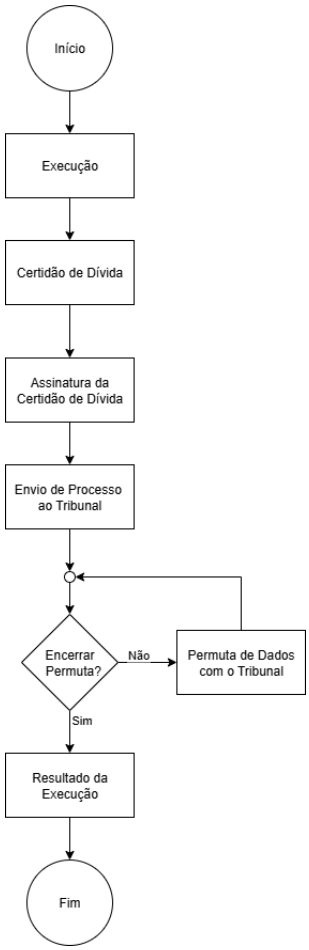

Figura 5.34: Fluxo da etapa *Execução* do processo.

Com a conclusão da etapa *Execução*, o estado da condição "Pagamentos" do processo evolui para o valor concluído, finalizando uma das duas condições necessárias para o encerramento do processo.

5.2.5.1 Execução

O passo execução, da responsabilidade do utilizador *Gestor*, inicia o fluxo da etapa *Execução* e pode apenas ser preenchido caso se reúnam as condições necessárias, nomeadamente se um dos pagamentos pendentes do processo expirar, ou seja, caso os pagamentos pendentes do processo não sejam pagos no prazo estipulado.

A figura 5.35 ilustra o formulário deste passo.

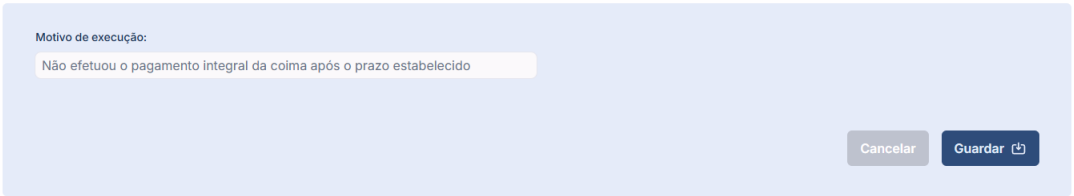

Figura 5.35: Formulário do passo execução da etapa *Execução* de um processo grave/muito grave.

5.2.5.2 Certidão de Dívida

A certidão de dívida é o segundo passo da etapa *Execução*, e é da responsabilidade do utilizador *Jurista*. Este passo permite ao utilizador a análise da dívida e consequente geração do documento `Certidão de Dívida`.
A figura 5.36 ilustra o formulário responsável por este passo.

Figura 5.36: Formulário do passo certidão de dívida da etapa *Execução* de um processo grave/muito grave.

5.2.5.3 Assinatura da Certidão de Dívida

O passo assinatura da certidão de dívida, da responsabilidade do utilizador *Subdiretor*, é o terceiro passo da etapa *Execução*. Este passo é responsável pela assinatura do documento gerado no passo anterior (certidão de dívida).
A figura 5.37 ilustra a página do formulário responsável por este passo.

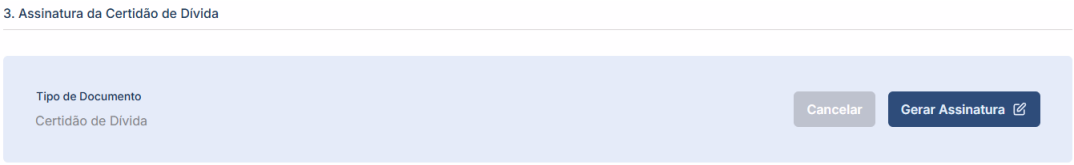

Figura 5.37: Formulário do passo assinatura da certidão de dívida da etapa *Execução* de um processo grave/muito grave.

5.2.5.4 Envio de Processo ao Tribunal

O passo envio de processo ao tribunal é o quarto passo da etapa *Execução*, e é da responsabilidade do utilizador *Gestor*. Este passo possui apenas um formulário muito simples cuja única responsabilidade é a do registo da data da evolução da execução para tribunal.

5.2.5.5 Permuta de Dados com o Tribunal

O passo permuta de dados com o tribunal, da responsabilidade do utilizador *Gestor*, apresenta um passo opcional, do fluxo da etapa *Execução*, com o intuito de registar as comunicações realizadas entre a SRTT e o tribunal. O processo de contraordenação apenas evolui para o passo seguinte, quando o utilizador optar por encerrar a permuta de dados com o tribunal, se o utilizador não indicar o mesmo o processo permite a adição de qualquer quantidade de permutas de dados.

A figura 5.38 ilustra parte da página que contém a informação das permutas de dados da etapa *Execução*, quando este passo é encerrado sem a realização de uma permuta de dados.

Figura 5.38: Secção da página da etapa *Execução* do passo permuta de dados com o tribunal, quando não são apresentadas permutas de dados.

5.2.5.6 Resultado da Execução

O resultado da execução é o passo final da etapa *Execução*, e é da responsabilidade do utilizador *Gestor*.
A submissão do formulário que constitui este passo (figura 5.39), finaliza a etapa *Execução*, o que se reflete com a evolução do estado da condição "Pagamentos" do processo para concluído, uma das condições necessárias para o encerramento do processo.

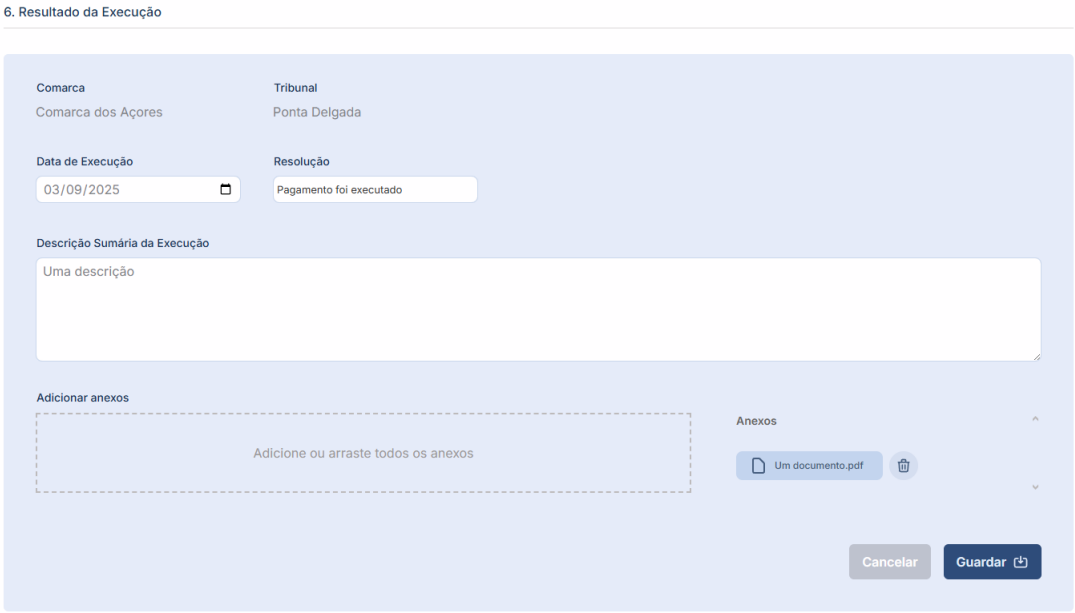

Figura 5.39: Formulário do passo resultado da execução da etapa *Execução* de um processo grave/muito grave.

5.2.6 Implementação dos Pedidos

Os pedidos são uma funcionalidade que pode ser requisitada, em momentos específicos do fluxo do processo de contraordenação. Quando é iniciado um pedido, o processo de contraordenação fica suspenso até à finalização do pedido, ou seja, os prazos estipulados do processo de contraordenação são congelados até à finalização do pedido. Dependendo do pedido realizado, este pode modificar o processo de contraordenação de formas distintas, podendo modificar o fluxo do processo de contraordenação ou apenas os prazos de pagamento.
Esta funcionalidade está disponível na plataforma pela *tab* "Pedidos" na página do processo de contraordenação (figura 5.40), nesta página são apresentados ao utilizador sete cartões,

cada um destes cartões representa um pedido distinto. Estes pedidos são:

1. Identificação de condutor;
2. Prestações;
3. Dilação de prazo;
4. Suspensão simples;
5. Atenuação especial;
6. Caução de boa conduta/formação;
7. Outros.

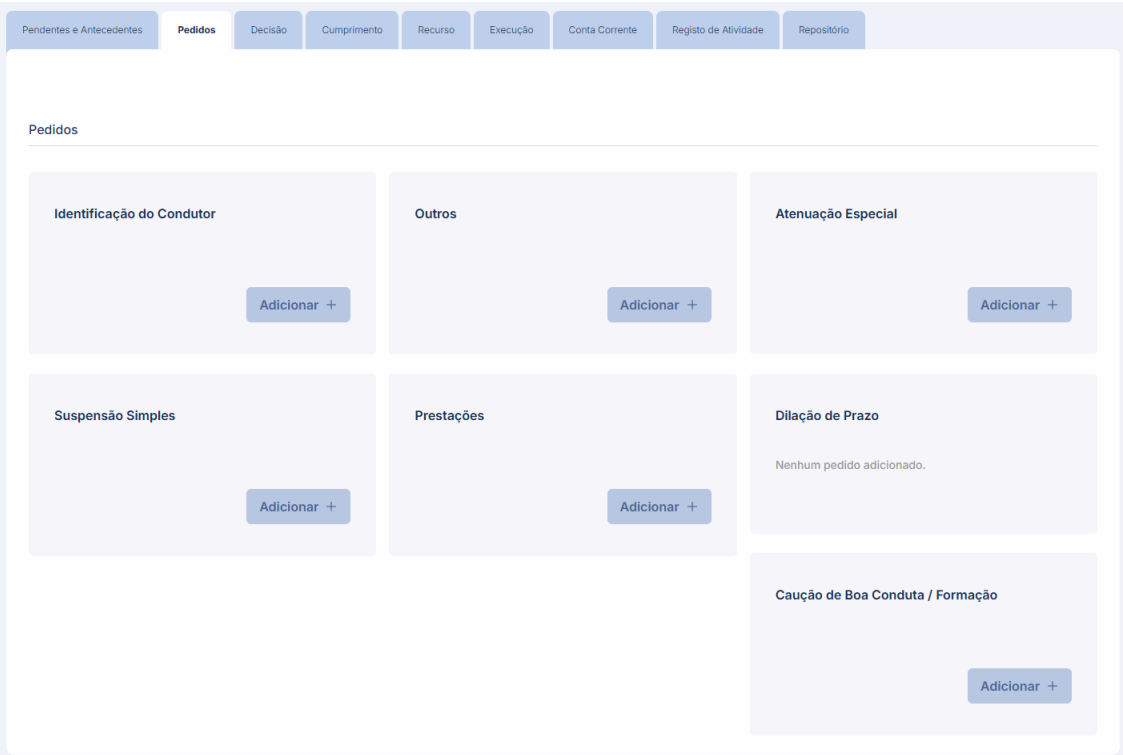

Figura 5.40: Página da funcionalidade pedidos de um processo grave/muito grave.

Cada pedido necessita que o processo se encontre em condições específicas para que possa ser iniciado, e modifica o fluxo do processo de contraordenação de forma distinta. O fluxo dos pedidos segue o diagrama da figura 5.41

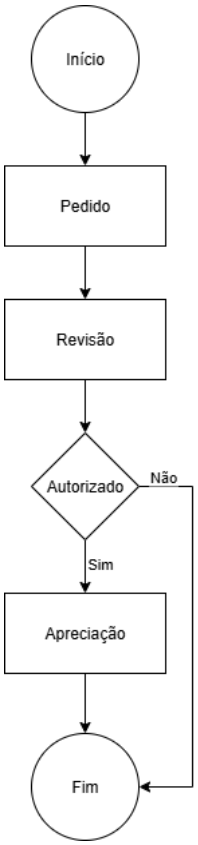

Figura 5.41: Fluxo comum dos pedidos de um processo de contraordenação.

Como pode ser realizada qualquer combinação de pedidos, a informação correspondente a cada um é armazenada numa entidade própria. Da mesma forma, cada pedido possui um serviço próprio, responsável pela sua lógica. Esta divisão de responsabilidades permite alcançar um código mais organizado e de fácil manutenção.

5.2.6.1 Identificação de Condutor

A ”identificação de condutor”, é um dos pedidos do processo e é utilizado quando é identificado o arguido incorreto. Este pedido pode ser realizado durante o período de apresentação de defesa (secção 5.2.1.1). A finalização, e autorização, deste pedido implica que o processo de contraordenação seja arquivado por motivo de ”auto conexo”.

A figura 5.42 ilustra a página dos pedidos após a finalização de um pedido de ”identificação de condutor”.

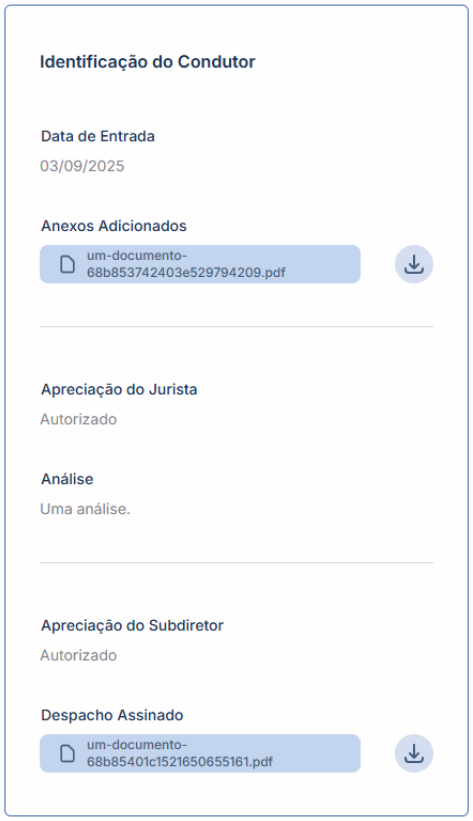

Figura 5.42: Parte da página dos pedidos de uma contraordenação, após conclusão do pedido de ”identificação de condutor”.

5.2.6.2 Prestações

O pedido de ”prestações” é utilizado pelo arguido para a realização da divisão dos pagamentos pendentes em prestações. Este pode ser realizado durante o período de apresentação de defesa, ou durante o período de pagamento, da coima e custas, após a notificação da decisão administrativa.

A finalização do pedido de ”prestações” aplica um conjunto de alterações aos pagamentos pendentes do processo, nomeadamente o seu cancelamento e a geração de novos pagamentos pendentes correspondentes a cada uma das prestações.

A figura 5.43 ilustra a página dos pedidos após a finalização de um pedido de ”prestações”.

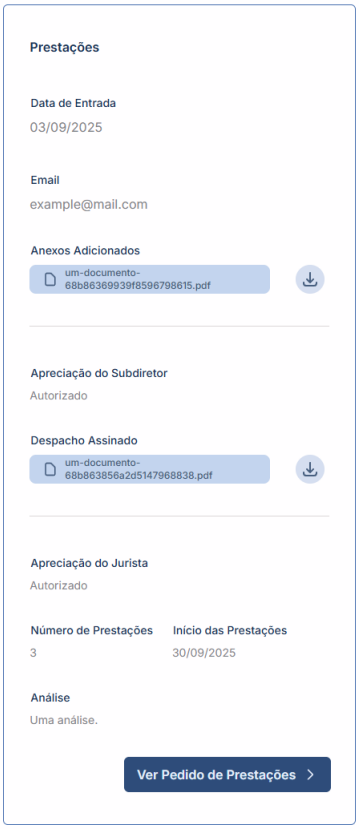

Figura 5.43: Parte da página dos pedidos de uma contraordenação, após conclusão do pedido de ”prestações”.

No entanto, caso este pedido seja realizado durante o período de apresentação de defesa, a última etapa é realizada em simultâneo ao passo da sanção na etapa *Decisão*. Esta modificação é apresentada na figura 5.44 e permite ao utilizador o preenchimento da informação das prestações no formulário da sanção.

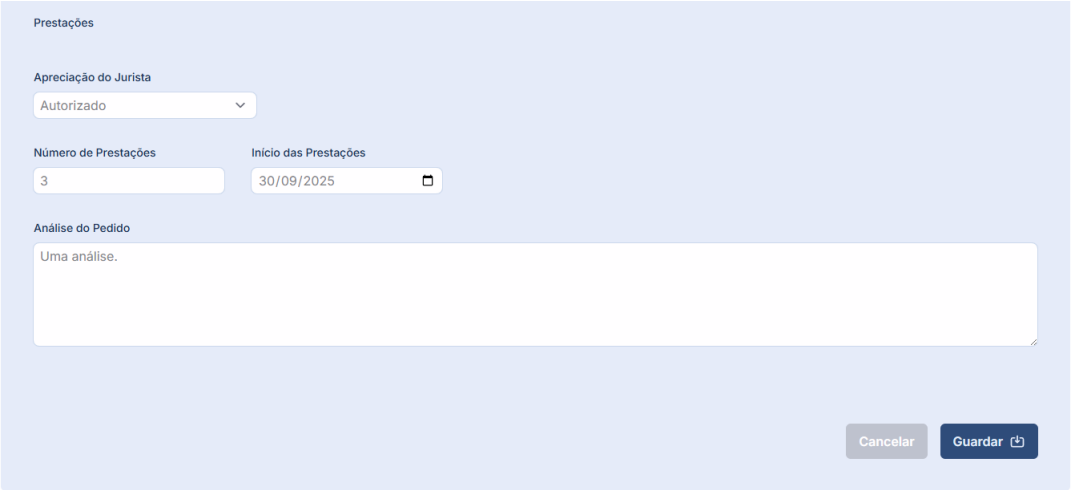

Figura 5.44: Formulário das prestações do passo sanção da etapa *Decisão* do processo grave/muito grave.

5.2.6.3 Dilação de Prazo

O pedido ”dilação de prazo” é utilizado para alterar o prazo dos pagamentos pendentes, e pode ser realizado durante o período de apresentação de defesa, ou durante o período de pagamento após a notificação da decisão administrativa.

Caso este pedido seja realizado durante o período de apresentação de defesa, a dilação de prazo será aplicada ao pagamento pendente de coima mínima. No entanto, caso seja realizado após a notificação da decisão administrativa, a dilação será aplicada aos pagamentos pendentes da coima e custas.

A figura 5.45 ilustra a página pedidos após a conclusão deste pedido.

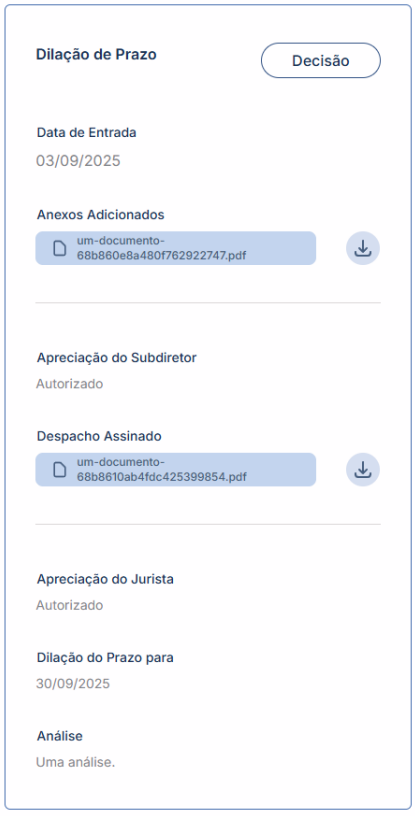

Figura 5.45: Parte da página dos pedidos de uma contraordenação, após conclusão do pedido de ”dilação de prazo”.

5.2.6.4 Suspensão Simples

O pedido ”suspensão simples” é utilizado para modificar os cumprimentos (secção 5.2.3) que serão aplicados ao processo. Este pedido pode ser realizado durante o passo de apresentação de defesa da etapa *Decisão*.

A figura 5.46 ilustra a página pedidos após o encerramento deste pedido.

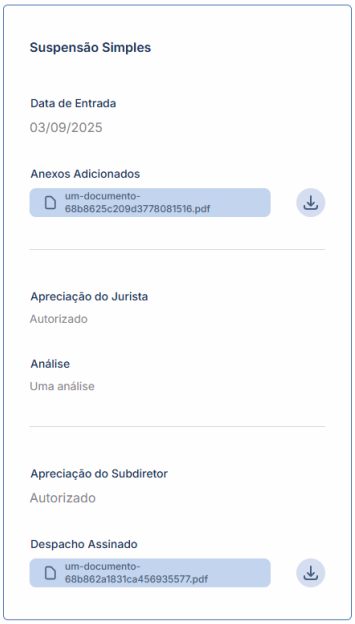

Figura 5.46: Parte da página dos pedidos de uma contraordenação, após conclusão do pedido de ”suspensão simples”.

A conclusão do pedido aplica uma alteração no formulário do passo sanção da etapa *Decisão*. Esta alteração reflete-se na adição de um novo formulário (figura 5.47) que permite a escolha do cumprimento que irá compor o processo. Este formulário permite a escolha de um dos seguintes cumprimentos:

1. Suspensão simples;
2. Ação de formação;
3. Caução de boa conduta;
4. Amnistia.

Figura 5.47: Formulário da suspensão simples do passo sanção da etapa *Decisão* do processo grave/muito grave.

Deste modo, após a notificação da decisão administrativa da etapa *Decisão*, a plataforma irá adicionar ao processo o cumprimento selecionado.

5.2.6.5 Atenuação Especial

O pedido de ”atenuação especial” é utilizado para alertar o *Jurista* que irá realizar o passo sanção da etapa *Decisão* do processo de contraordenação. Este pedido pode ser realizado durante o passo de apresentação de defesa da etapa *Decisão*.

A figura 5.48 ilustra a página pedidos após a conclusão deste pedido.

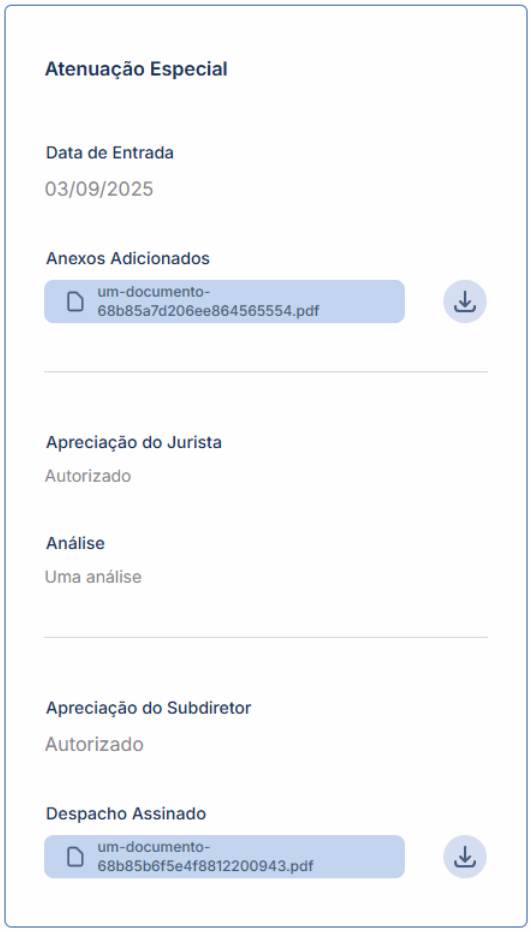

Figura 5.48: Parte da página dos pedidos de uma contraordenação, após conclusão do pedido de”atenuação especial”.

A conclusão do pedido aplica uma alteração no formulário do passo sanção da etapa *Decisão*. Esta alteração reflete-se na adição de um novo formulário (figura 5.49) que permite o pre-enchimento do período de inibição de condução e a especificação dos documentos a serem entregues.

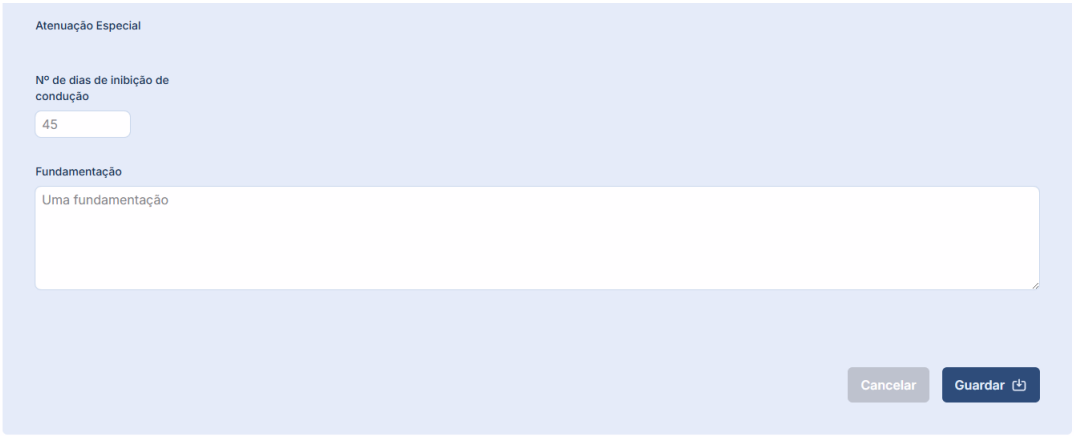

Figura 5.49: Formulário da atenuação especial do passo sanção da etapa *Decisão* do processo grave/muito grave.

5.2.6.6 Caução de Boa Conduta/Formação

O pedido de ”caução de boa conduta/formação” é utilizado para substituir os cumprimentos que serão aplicados ao processo, pelo cumprimento ”caução de boa conduta” ou pela ”ação de formação”. Este pedido apenas pode ser iniciado durante o período de apresentação de defesa da etapa *Decisão*, e não pode ser iniciado caso já exista um pedido de ”suspensão simples” associado ao processo de contraordenação.
A figura 5.50 ilustra a página pedidos após a conclusão deste pedido.

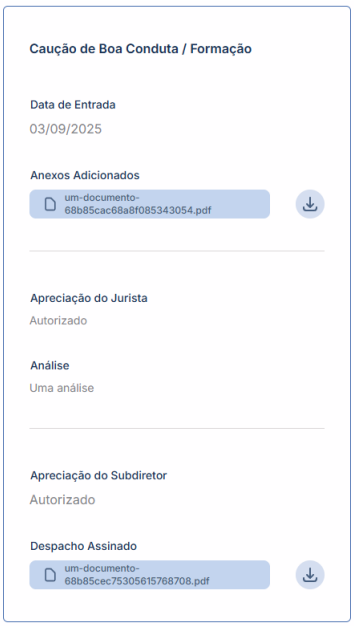

Figura 5.50: Parte da página dos pedidos de uma contraordenação, após conclusão do pedido de ”caução de boa conduta/formação”.

A conclusão do pedido aplica uma alteração no formulário do passo sanção da etapa *Decisão*. Esta alteração reflete-se na adição de um novo formulário (figura 5.51) que permite a seleção do cumprimento que irá ser adicionado ao processo de contraordenação.

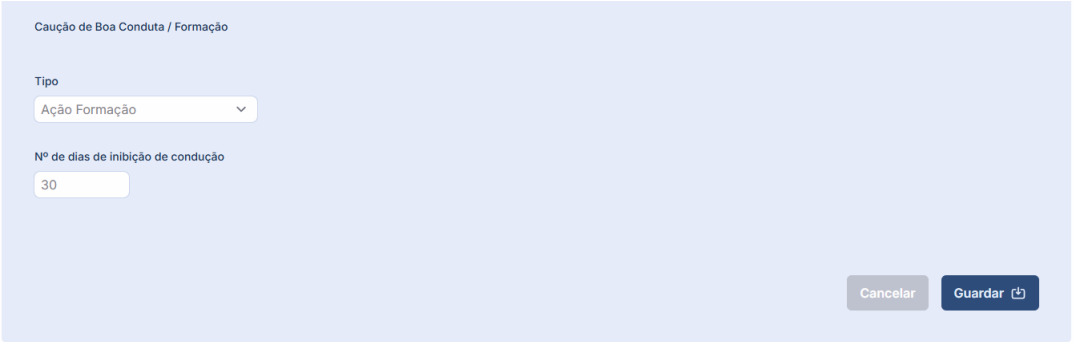

Figura 5.51: Formulário da caução de boa conduta/formação do passo sanção da etapa *Decisão* do processo grave/muito grave.

Deste modo, após o passo notificação da decisão administrativa da etapa *Decisão*, a plataforma irá adicionar ao processo o cumprimento selecionado.

5.2.6.7 Outros

O pedido ”outros”, permite ao utilizador a adição de um pedido que não seja contemplado por nenhum dos anteriormente referidos. Este pedido apenas pode ser realizado durante o período de apresentação de defesa, e é o único que não modifica o processo de contraordenação, sendo utilizado para fins de registo.

A figura 5.52 ilustra a página pedidos após a conclusão deste pedido.

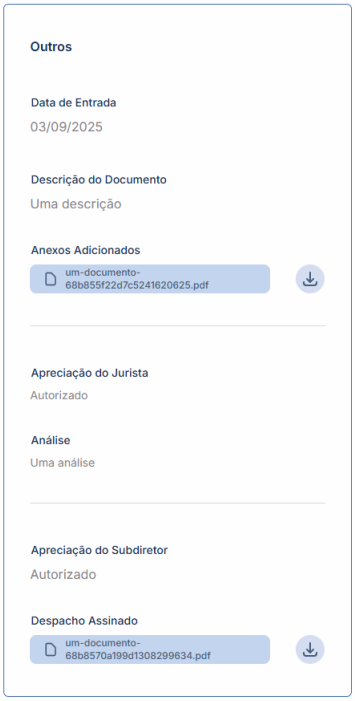

Figura 5.52: Parte da página dos pedidos de uma contraordenação, após conclusão do pedido de ”oturos”.

5.3 Tarefa 2 - Definição e Implementação da Funcionalidade de Agrupamento de Processos

Após a implementação do fluxo de uma contraordenação grave/muito grave, foi implementada a funcionalidade de agrupamento de processos. Esta funcionalidade permite que múltiplas contraordenações sejam agrupadas num único processo de contraordenação.
Nem todos os processos de contraordenação podem ser agrupados, sendo que um conjunto de processos de contraordenação necessita de cumprir um conjunto de condições específicas para ser agrupado num único processo. Estas condições são:

1. Estado do processo – Os processos de contraordenação necessitam de estar pendentes, no passo defesa da etapa *Decisão*;

2. Arguido em comum – Os processos de contraordenação a serem agrupados, necessitam de partilhar o mesmo arguido;

3. Nível de infração – Apenas podem ser agrupados processos graves e muito graves.

Qualquer processo que não cumpra todas estas condições, não poderá ser agrupado.
Na plataforma, o agrupamento de processos pode ser realizado pelo utilizador *Jurista* na *tab* "Pendentes e Antecedentes" (figura 5.53). É também possível desagrupar processos enquanto que o passo decisão administrativa da etapa *Decisão* não for submetido para assinatura.

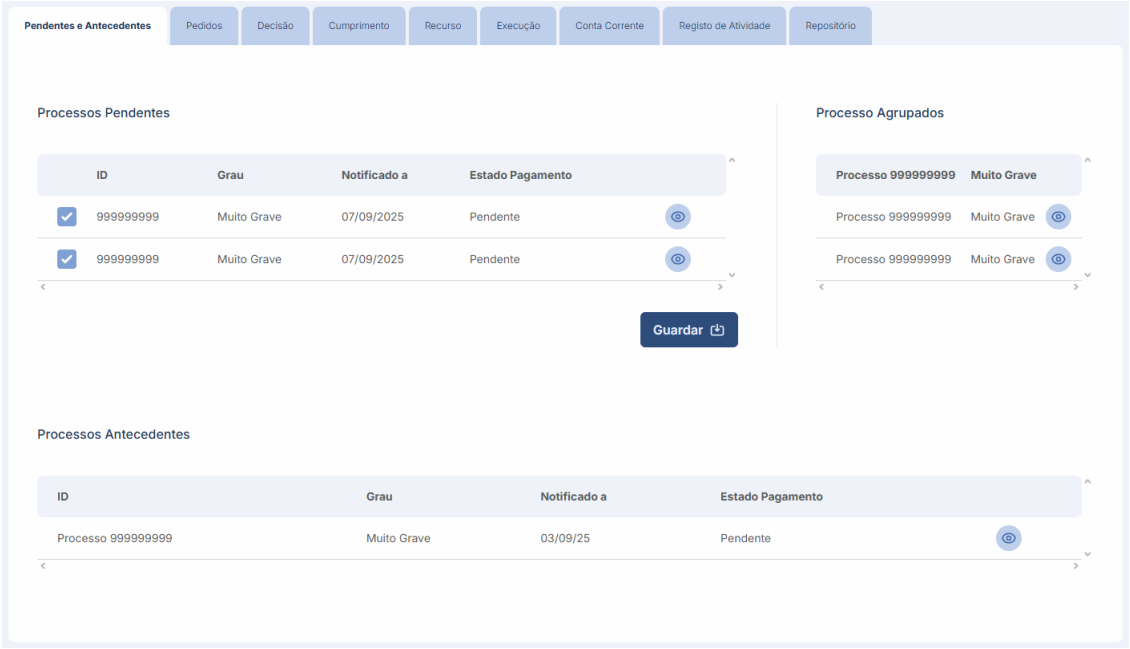

Figura 5.53: Página dos pendentes e antecedentes do processo grave/muito grave.

O fluxo de um processo de contraordenação agrupado segue um fluxo idêntico ao de um processo não agrupado, porém, o formulário do passo decisão da etapa *Decisão* possui alguns

campos adicionais. Mais concretamente, o formulário modificado (figura 5.54) do passo decisão (secção 5.2.1.3) apresenta ao utilizador um modal por processo agrupado, de preenchimento obrigatório. Este modal permite ao utilizador o preenchimento da decisão parcial de cada um dos processos agrupados.

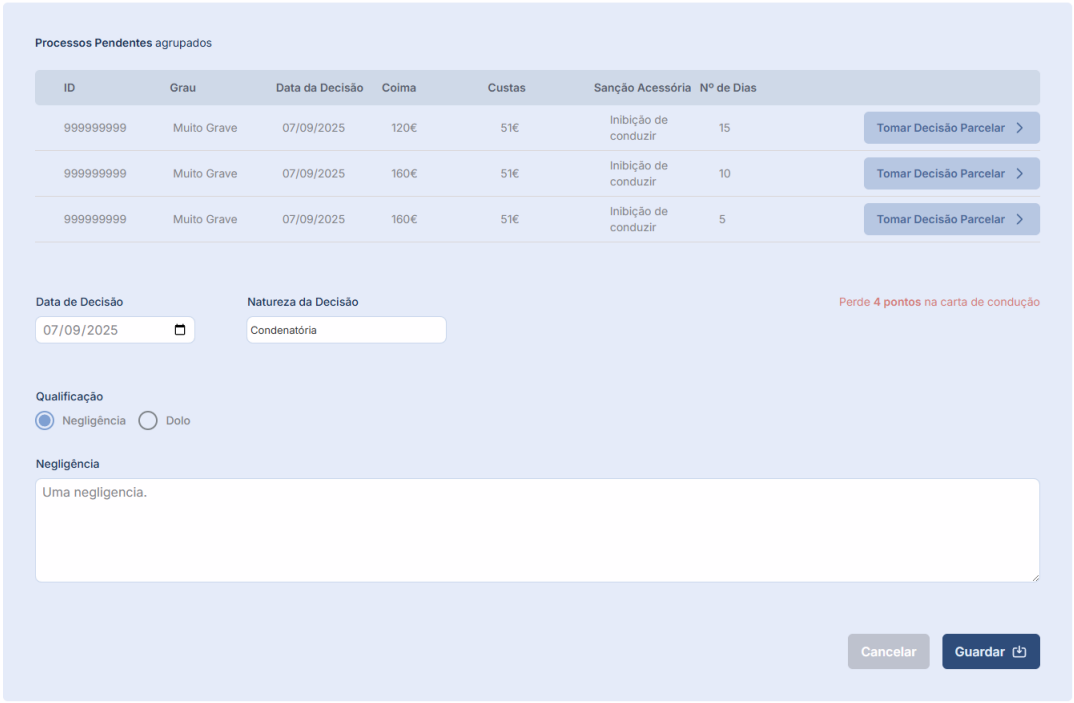

Figura 5.54: Formulário do passo decisão da etapa *Decisão* dum processo grave/muito grave agrupado.

Para acomodar as alterações do formulário do passo *decisão*, foi adicionada uma relação à entidade `Decisão` que representa a ligação da decisão principal a cada uma das decisões parcelares que a compõem. Assim, uma decisão possui um conjunto de decisões dependentes com as suas respetivas sanções. A figura 5.55 ilustra estas relações.

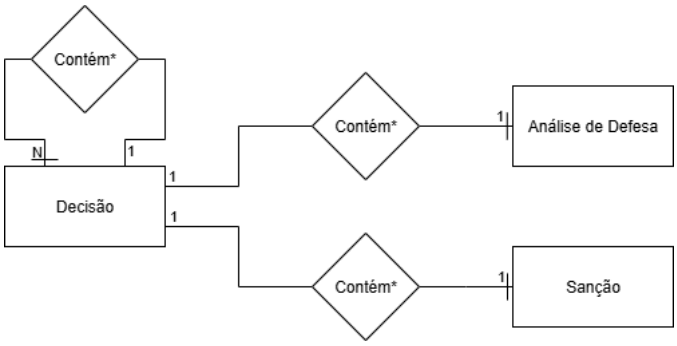

Figura 5.55: Diagrama Entidade/Associação da decisão após implementação da funcionalidade de processos agrupados.

5.4 Tarefa 3 - Unificação do Fluxo dos Processos Grave/Muito Grave e os Processos Leve

Esta tarefa teve como objetivo a unificação do fluxo dos processos de contraordenação grave/muito grave e leve. Numa fase inicial, estes dois tipos de processo de contraordenação foram implementados de forma independente com os seus respetivos controladores, serviços e entidades. A partir da constante validação e levantamento de requisitos com o cliente, apercebeu-se que a abordagem inicial era incompleta e não representava corretamente as necessidades do cliente. Mais concretamente, o fluxo dos processos grave/muito grave encontrava-se incompleto, faltando-lhes o fluxo dos processos leve como etapa inicial.

Como estes dois fluxos estavam separados, e armazenados em entidades distintas, foi realizado um esforço em conjunto com os colegas da equipa de desenvolvimento, responsáveis pela implementação do fluxo leve, para unificar estes dois fluxos.

Nomeadamente, foram realizadas alterações ao nível da base de dados, e do fluxo do processo grave/muito grave. Os dois fluxos distintos foram unificados para dar lugar a um único fluxo que contempla todo o ciclo de vida de um processo de contraordenação, independente do seu nível de infração. Assim, um processo grave/muito grave, em vez de iniciar na etapa de *Decisão*, inicia agora na etapa *Entidade*, que posteriormente irá evoluir para a etapa *Decisão* percorrendo o fluxo abordado na primeira tarefa (secção 5.2).

O diagrama da figura 5.56 ilustra de forma simplificada, o fluxo dos processos de contraordenação grave/muito grave e leve após a unificação dos seus fluxos. Neste é possível observar a ramificação após a conclusão da etapa *Entidade*, que poderá evoluir para um encerramento imediato caso o processo seja da tipologia leve. No diagrama, a etapa *Secretaria* representa o fluxo da figura 5.2, desempenhado pela SRTT.

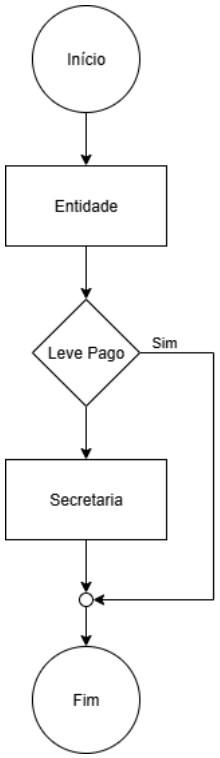

Figura 5.56: Fluxo unificado simplificado de um processo de contraordenação.

5.5 Tarefa 4 - Definição e Implementação do Fluxo do Processo Leve Agravado

A quarta tarefa, que compõe o plano inicial de estágio, é uma continuação imediata da terceira tarefa. Como anteriormente discutido, um processo leve é normalmente encerrado após a realização da etapa *Entidade*, não passando por todas as etapas do fluxo de um processo de contraordenação. Porém, caso o processo de contraordenação leve esteja em condições específicas, este em vez de ser encerrado, evolui para um estado agravado, que implica a realização quase total do fluxo de um processo de contraordenação grave/muito grave.

Um processo leve é agravado caso a sua coima mínima não seja paga no prazo estabelecido, este prazo é normalmente de quinze dias após a notificação ao arguido da contraordenação. Nesta situação, o processo leve evolui para uma condição de ”Agravado”, e em vez de ser encerrado avança para a etapa *Decisão*.

Ao contrário de um processo grave/muito grave um processo leve não possui cumprimentos, ou seja, um processo leve possui apenas uma condição de finalização, a condição ”Pagamentos”. Esta alteração significa que o processo leve agravado pode ser encerrado pelo utilizador *Gestor* quando os seus pagamentos pendentes forem marcados como pagos. Para além da inexistência de cumprimentos, um processo leve agravado não possui o passo decisão da etapa *Decisão*. A figura 5.57 ilustra a página da etapa *Decisão* de um processo leve agravado.

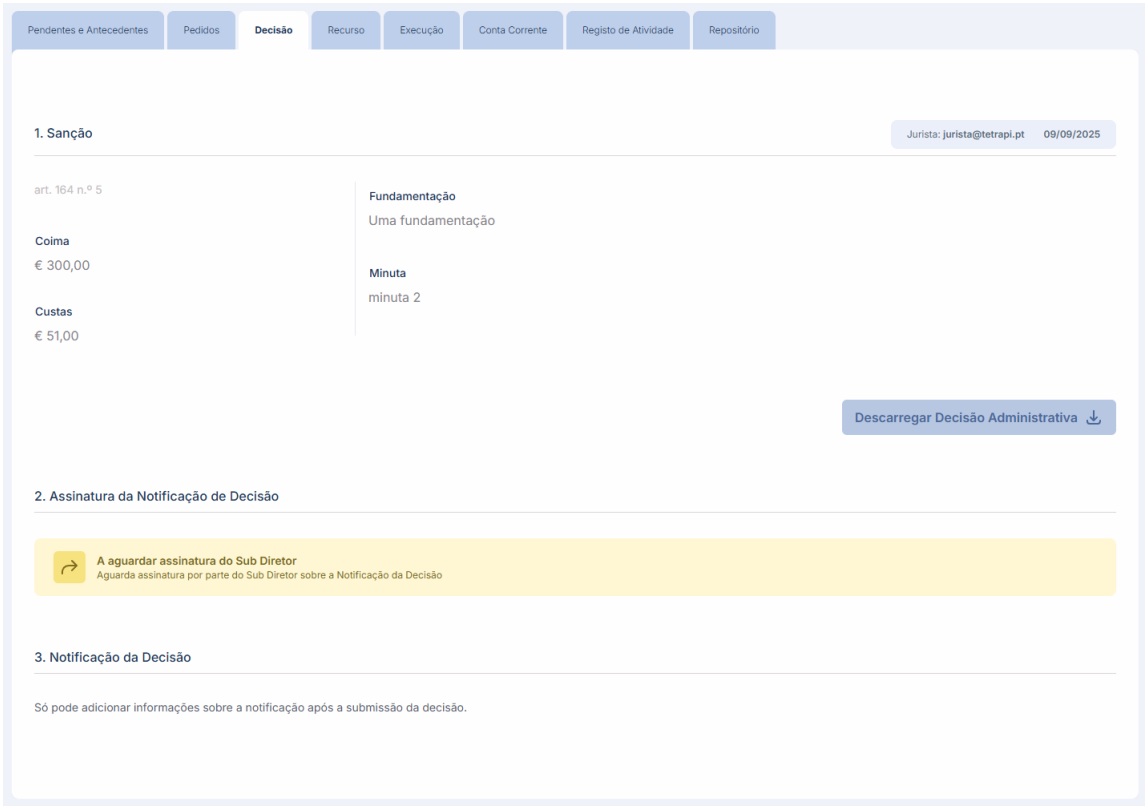

Figura 5.57: Página da etapa *Decisão* de um processo leve agravado.

O diagrama da figura 5.58 ilustra o fluxo simplificado de um processo leve agravado.

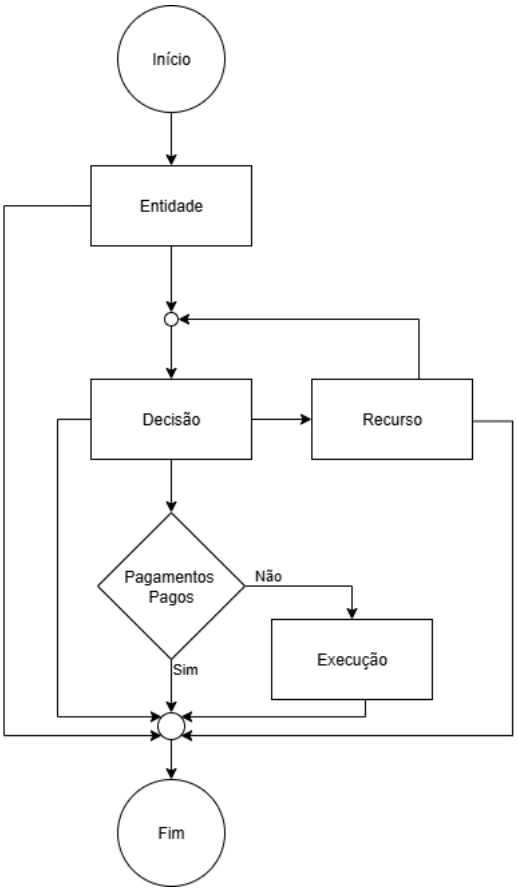

Figura 5.58: Fluxo simplificado de um processo leve agravado.

5.6 Tarefa 5 - Implementação de Testes Unitários

A tarefa cinco é a última tarefa que compõe o plano de estágio, e foi desempenhada no decorrer das tarefas anteriores. Esta teve como objetivo o desenvolvimento e manutenção de testes unitários que visavam validar os diferentes campos que compunham os formulários. Para cada um dos formulários desenvolvidos, foi implementado um ficheiro com os testes unitários respetivos ao mesmo. Neste, cada uma das situações foi dividida no seu método próprio.
Os testes unitários desenvolvidos, faziam parte das *pipelines* (secção 4.5.1) do projeto sendo que estes necessitavam de concluir em sucesso para que as alterações fossem aplicadas no servidor.
Os testes unitários foram desenvolvidos utilizando as seguintes técnicas de testes *black-box* [28]:

1. Partição de entrada – Para agrupar os diferentes valores que os campos do formulário podem possuir;
2. Fronteira – Para realizar testes com valores que ultrapassem o limite dos valores válidos esperados;
3. Tabelas de decisão – Para testar diferentes combinações de valores.

De forma a cobrir o maior número de casos e as ramificações das instruções de fluxo e ciclos, foram desenvolvidos testes almejando uma métrica de *code coverage* [28] total.

5.7 Conclusão

Este capítulo abordou a plataforma SGCO, o seu objetivo, e cada uma das tarefas principais constituintes do plano de estágio. O próximo capítulo irá concluir o presente relatório.

Capítulo 6

Conclusão e Trabalho Futuro

6.1 Conclusão

O presente relatório, abordou com o maior detalhe as tarefas principais que compunham o plano de estágio e as suas respetivas implementações. Todas as tarefas propostas foram concluídas no prazo pretendido seguindo todas as regras estabelecidas pela empresa.
A participação no desenvolvimento deste projeto, permitiu enriquecer o meu conhecimento na área de desenvolvimento *web*, expondo-me a novos desafios e boas práticas. Pude aprender sobre ferramentas e técnicas com as quais nunca havia trabalhado, tais como a geração de documentos e a utilização dos pacotes `Symfony UX` de modo a reduzir o código `JavaScript` que compõe uma plataforma *web*.
Este estágio, não se limitou apenas à minha participação no projeto SGCO, estando em constante contacto com todos os colaboradores da Tetrapi SA, o que permitiu a minha evolução pessoal estando exposto a diversas situações profissionais.
Para além das tarefas principais que compunham o estágio, foi-me confiado também um conjunto de responsabilidades pela empresa, reconhecendo o bom trabalho desenvolvido.
Realizei funções de manutenção e evolução de uma aplicação *web*, também esta para o Governo Regional dos Açores, que visa a atualização do procedimento da gestão dos seus recursos móveis. Tive também o gosto de orientar duas equipas de desenvolvimento, acompanhando-as e guiando-as no desenho e implementação das diversas funcionalidades dos seus projetos. E supervisionei e acompanhei por completo um estágio curricular da Universidade dos Açores, desde a sua integração na empresa, à supervisão do desenho e implementação das tarefas que compunham o estágio curricular.

6.2 Trabalho Futuro

O estágio curricular focou-se na minha participação no desenvolvimento da plataforma SGCO, a qual se encontra na fase final de desenvolvimento. Assim, irei continuar a desempenhar as minhas funções na equipa de desenvolvimento até a conclusão da plataforma.
Algumas das funcionalidades que poderão ainda ser implementadas na plataforma, são as integrações desta com outras plataformas. Nomeadamente, a integração com o sistema de gestão de pontos na carta da ANSR, a integração com um sistema de pagamentos, e uma integração com um sistema de envio automatizado de correio.
Para além disto, irei continuar a desempenhar as minhas funções de acompanhamento de outros projetos atualmente em desenvolvimento e a integração de novos estagiários na empresa.

Bibliografia

[1] E. C. C. Leão, M. R. Martins, and J. S. Toquica, "Desenvolvimento de um sistema web para gerenciamento de projetos ti baseado no pmbok," 2021. [Online]. Available: https://www.researchgate.net/publication/354922163_Desenvolvimento_de_um_Sistema_Web_para_Gerenciamento_de_Projetos_TI_Baseado_no_PMBOK xiii, 9

[2] I. Editorial Team, 2025. [Online]. Available: https://www.indeed.com/career-advice/finding-a-job/lead-programmer 2

[3] K. Schwaber and J. Sutherland. (2020) The scrum guide. [Online]. Available: https://scrumguides.org/scrum-guide.html 9

[4] Atlassian. Three pillars of scrum: Understanding scrum's core principles. [Online]. Available: https://www.atlassian.com/agile/project-management/3-pillars-scrum 10

[5] C. Harris. [Online]. Available: www.atlassian.com/en/microservices/microservices-architecture/microservices-vs-monolith 12

[6] Redhat. (2023). [Online]. Available: https://www.redhat.com/pt-br/topics/microservices/what-are-microservices 12

[7] Amazon, "Qual é a diferença entre arquitetura monolítica e de microsserviços?" [Online]. Available: https://aws.amazon.com/compare/the-difference-between-monolithic-and-microservices-architecture/ 13

[8] Quantiphi. (2021, Mar). [Online]. Available: https://quantiphi.com/blog/an-introduction-to-mvcs-architecture 14

[9] M. Fowler, *Patterns of Enterprise Application Architecture*, 1st ed. Addison-Wesley Professional, 2002, p. 388–413. 14

[10] Symfony. (2025) Validation. [Online]. Available: https://symfony.com/doc/current/validation.html 15

[11] Doctrine. (2025) Welcome to doctrine orm's documentation! [Online]. Available: https://www.doctrine-project.org/projects/doctrine-orm/en/3.5/index.html 16

[12] Symfony. (2025) Scheduler. [Online]. Available: https://symfony.com/doc/current/scheduler.html 16

[13] ——. (2025) Workflow. [Online]. Available: https://symfony.com/doc/current/workflow.html 17

[14] P. Linz, *An Introduction to Formal Languages and Automata*, 6th ed. Jones & Bartlett Learning, 2016, p. 35–69. 17

[15] KnpLabs. (2025, Jan) Knpsnappybundle. [Online]. Available: https://github.com/KnpLabs/KnpSnappyBundle 19

[16] C. PHP. file_put_contents. [Online]. Available: https://www.php.net/manual/en/function.file-put-contents.php 19

[17] Symfony. (2019) Twig documentation. [Online]. Available: https://twig.symfony.com/doc/3.x 20

[18] ——. (2025, Aug) Ux packages. [Online]. Available: https://ux.symfony.com/packages 20

[19] ——. (2025) Symfony ux live components documentation. [Online]. Available: https://symfony.com/bundles/ux-live-component/current/index.html 20

[20] Amazon. [Online]. Available: https://aws.amazon.com/what-is/ajax/ 20

[21] SymfonyCasts. (2024, Nov) Dynamic / dependent symfony form fields. [Online]. Available: https://github.com/SymfonyCasts/dynamic-forms 21

[22] E. Mann, *PHP Cookbook*, 1st ed. O'Reilly Media, 2023. 23

[23] G. Staff. (2024, Nov) Octoverse: Ai leads python to top language as the number of global developers surges. [Online]. Available: https://github.blog/news-insights/octoverse/octoverse-2024 23

[24] M. Nystrom, *SQL injection defenses*. O'Reilly, 2007. 24

[25] C. Staff. (2024, Sep) What is a kanban board? [Online]. Available: https://www.coursera.org/articles/kanban-board 25

[26] D. Farley, *Continuous Delivery Pipelines: How to Build Better Software Faster*. Dave Farley, 2021, p. 11–19. 25

[27] R. Osnat. (2020, Oct) A brief history of containers: From the 1970s till now. [Online]. Available: https://www.aquasec.com/blog/a-brief-history-of-containers-from-1970s-chroot-to-docker-2016 26

[28] Y. Singh, *Software testing*. Cambridge University Press, 2013. 72, 73

Glossário

Auto	Documento legal que regista uma infração.
Arguido	Sujeito processual a quem corre processo penal.
Coima	Sanção principal de um procedimento contraordenacional, a montante da coima é definida no código de estrada num intervalo.
Custas	São as despesas concretas geradas no decorrer do processo de contraordenação.
LaTeX	Conjunto de macros para o processador de textos TeX, utilizado amplamente para a produção de textos matemáticos e científicos devido à sua alta qualidade tipográfica.