WENDER BERALDO LEMES PEREIRA

DESENVOLVIMENTO DE UM PROXY DE PAGAMENTOS PARA UMA REDE DE LABORATÓRIOS

LAVRAS – MG

2022

WENDER BERALDO LEMES PEREIRA

DESENVOLVIMENTO DE UM PROXY DE PAGAMENTOS PARA UMA REDE DE LABORATÓRIOS

Relatório de estágio supervisionado apresentado à Universidade Federal de Lavras, como parte das exigências do Curso de Sistemas de Informação, para obtenção do título de Bacharel.

Prof. DSc. André Pimenta Freire
Orientador

LAVRAS – MG

2022

WENDER BERALDO LEMES PEREIRA

DESENVOLVIMENTO DE UM PROXY DE PAGAMENTOS PARA UMA REDE DE LABORATÓRIOS
DEVELOPMENT OF A PAYMENTS PROXY FOR A LABORATORIES NETWORK

Relatório de estágio supervisionado apresentado à Universidade Federal de Lavras, como parte das exigências do Curso de Sistemas de Informação, para obtenção do título de Bacharel.

APROVADO em 9 de Setembro de 2022.

Prof. DSc. Neumar Costa Malheiros	UFLA
BSc. Vitor Campos e Silva	DTI Digital

Prof. André Pimenta Freire DCC/UFLA

Prof. DSc. André Pimenta Freire
Orientador

LAVRAS – MG
2022

Dedico este trabalho de forma igual aos que persistiram e aos que cederam durante o nosso trajeto. Cada um sabe a extensão dos seus sacrifícios.

AGRADECIMENTOS

Aos meus pais, pelo apoio incondicional e pela paciência que tiveram por minha ausência nesses anos. Aos amigos que fiz durante o curso, com os quais dividi os fardos e as felicidades da jornada. Aos amigos que precedem esse período - em especial ao meu melhor amigo, Gustavo Lopes, que sempre acreditou mais em mim do que eu mesmo. Aos professores que me inspiraram e me tornaram uma pessoa melhor. À minha madrinha, Meire, que plantou essa semente e, onde estiver, tenho certeza, está tão feliz quanto eu pelos seus frutos. Por fim, agradeço a Deus por reforçar minha resiliência a cada dia.

RESUMO

O presente relatório descreve as experiências durante o processo de desenvolvimento de um Proxy de Pagamentos para uma grande rede brasileira de laboratórios. O contato com esse sistema durante o estágio contemplou todas as principais fases do ciclo de vida do software, desde a ideação, passando pela implantação em ambiente produtivo, até a sua entrada no período de manutenção. O relatório apresenta as tecnologias e arquiteturas utilizadas, motivações das escolhas, metodologias de desenvolvimento e, principalmente, como o acompanhamento desse projeto impactou a experiência e a visão do estagiário em relação ao desenvolvimento de software.

Palavras-chave: Proxy de Pagamentos. Desenvolvimento de Software. Ciclo de Vida do Software.

ABSTRACT

The present report describes the experiences during the developing process of a Payments Proxy for a big Brazilian laboratories network. Working on this system over the internship contemplated all the main phases of a software's life cycle, from the idealization, through the use in productive environment, until its maintenance phase. This report will discuss used technologies and architectures, choices, motivations, development methodologies and, above all, how following this project affected the intern's experience and his perspective about software's development.

Keywords: Payments Proxy. Software Development. Software Life Cycle.

SUMÁRIO

1 INTRODUÇÃO

O estágio apresentado neste relatório foi realizado na empresa DTI Digital, tendo duração de doze meses, transcorridos entre agosto de 2020 e agosto de 2021. O foco das atividades foi o desenvolvimento de software através da abordagem ágil, o que implica majoritariamente na contribuição com tarefas de desenvolvimento para evolução e manutenção de código, mas também inclui a recorrente participação nos ritos de desenho da solução. Durante o estágio, o aluno pôde acompanhar todas as etapas do processo de desenvolvimento de um Proxy de Pagamentos para uma rede brasileira de laboratórios.

Um Proxy de Pagamentos, de forma sucinta, é um sistema intermediário que viabiliza ou facilita a comunicação entre determinados atores, visando à realização de transações financeiras – conforme é abordado com mais detalhes nas seções seguintes. Nesse sentido, o estágio em questão relaciona-se ao curso de Sistemas de Informação por proporcionar o contato com as diversas etapas do processo de desenvolvimento de um sistema de informação propriamente dito.

É esperado de um estagiário desenvolvedor que ele possa contribuir com as tarefas de desenvolvimento e que use essa experiência para aprimoramento do seu desenvolvimento profissional. Em outras palavras, o papel do estagiário não é ser um desenvolvedor importante para o projeto desde o princípio, mas tornar-se um.

Este relatório tem como objetivo principal explorar os impactos e as contribuições do estágio para a formação acadêmica do discente. Ele possui a seguinte organização: o Capítulo 2 apresenta os detalhes da organização onde o estágio foi realizado; o Capítulo 3 é focado nas atividades realizadas durante o período de estágio, bem como na forma como elas se relacionam com o curso de Sistemas de Informação; o Capítulo 4 aborda os aspectos técnicos relacionados às atividades realizadas, como tecnologias e ferramentas com as quais o estagiário teve contato; por fim, o Capítulo 5 compreende uma reflexão sobre as experiências adquiridas durante o estágio.

1.1 Objetivos do Estágio

Entre os principais objetivos do estágio estiveram: possibilitar a aplicação das técnicas e dos conhecimentos adquiridos sobre desenvolvimento e qualidade de software, bem como proporcionar o contato com os desafios práticos que, muitas vezes, a noção estritamente teórica é incapaz de evidenciar. Também é valido dizer que o estágio tende a inspirar uma noção

de responsabilidade única do mercado, visto que o estagiário está lidando com soluções que demandam investimento de tempo e dinheiro dos clientes. Além disso, o trabalho em equipe com outros profissionais da área de Tecnologia da Informação permite a troca de conhecimentos que estão, muitas vezes, além do escopo da graduação, o que é um complemento muito bem-vindo.

2 DESCRIÇÃO DO LOCAL DE ESTÁGIO

A DTI Digital (DTI, 2022b) é uma empresa de consultoria de TI sediada em Belo Horizonte, com filial em Lavras e clientes espalhados pelo restante do Brasil. Está no mercado desde 2009 e tem como foco a transformação digital de seus clientes e parceiros. Nesse sentido, oferece aos seus clientes não apenas soluções ágeis, mas o intercâmbio transparente de conhecimentos. Atualmente, conta com mais de mil colaboradores e, desde o início de 2021, integra a multinacional britânica WPP Group.

O portfólio da DTI (DTI, 2022a) é bastante vasto, tanto em termos de quantidade de clientes, quanto em relação às suas respectivas áreas, tendo desenvolvido soluções para organizações nos setores de transporte e logística, farmacêutico, de mineração, telecomunicações, entre outros. Apesar dessa diversidade, o modo de atuação da DTI possui um fator quase "universal", que seria compartilhar os seus processos com os clientes, de maneira que eles possam identificar e absorver o que acharem vantajoso, o que vai além do desenvolvimento de software e embasa a própria designação como empresa de consultoria e transformação digital.

Com o intuito de proporcionar maior dinamicidade e independência às equipes, a DTI trabalha com uma estrutura mista de setores centralizados e equipes distribuídas. A distribuição em questão é feita entre alianças, tribos e squads, cada um mais específico e focado que o anterior. Para melhor entendimento, serão citados abaixo os papéis mais comuns desempenhados dentro dessa estrutura, mas por enquanto vamos dizer que o squad é a unidade de trabalho mais específica, contando com uma equipe multidisciplinar que, de fato, desenvolve e lida diretamente com os projetos e clientes. Os squads são alocados entre tribos, que funcionam como suborganizações, com independência para gerir as equipes - inclusive em relação a novas contratações, orçamentos e projetos que contemplam. Nesse sentido, as formas de organização mais abrangentes seriam as alianças, que gerenciam várias tribos por uma perspectiva ampla.

Como dito acima, também existem setores centralizados na DTI, que têm como propósito servir de referência para as equipes. Tais setores não interagem diretamente com as equipes, mas apoiam as operações que o fazem. Por exemplo, o setor centralizado de Recursos Humanos não cuida das contratações de uma tribo, mas apoia os responsáveis distribuídos entre elas quando encaram situações com as quais não conseguem lidar por conta própria, ou quando precisam de quaisquer outras orientações.

2.1 Papéis Comuns na Organização

2.1.1 Desenvolvedor

Assim como em outras empresas, o desenvolvedor é aquele que trabalha diretamente com a parte técnica do software, usando as linguagens de programação e demais tecnologias disponíveis para concretizar os projetos idealizados. Embora o reconhecimento pelas capacidades individuais e a chamada "alavancagem" ocorram, não são utilizadas como referência na DTI divisões mais tradicionais de categorias entre os desenvolvedores - como júnior, pleno e sênior.

2.1.1.1 Desenvolvedor Líder

Entende-se que o desenvolvedor nesse papel tenha menos atribuições práticas de desenvolvimento, de modo que possa aplicar seu tempo e seus esforços apoiando os demais integrantes da equipe de desenvolvimento. Sendo assim, existem vários caminhos que um desenvolvedor líder pode seguir. Por exemplo, existem os que possuem um arsenal técnico maior e conseguem auxiliar em questões complexas de desenvolvimento, mas nada impede também que o foco desse profissional seja mais voltado para a parte comunicativa e ele ajude removendo empecilhos dessa ordem com o cliente.

2.1.1.2 Arquiteto de Soluções

Ainda entre os papéis mais focados nos aspectos técnicos do software está o arquiteto. O profissional com essa função elabora soluções em um nível mais alto de abstração, pautando-se em padrões de projeto e boas práticas, sempre analisando também o contexto e as necessidades de cada aplicação.

2.1.2 Product Owner

Embora a preocupação com o produto deva ser comum a toda a equipe, o Product Owner é o papel que leva isso como atribuição principal. Nesse sentido, é responsável, entre outras coisas, por entender bem as necessidades do cliente e as regras de negócio da aplicação, sendo capaz de compartilhar esse conhecimento com o time de forma clara.

2.1.3 Designer

Ao lado do Product Owner, o Designer é outro integrante fundamental em relação à qualidade do produto. Mais do que projetar interfaces e soluções, o Designer na DTI auxilia no processo de descoberta das necessidades do cliente, propondo continuamente experiências que ajudam a resignificar o uso e levá-lo além do esperado em termos de valor e qualidade.

2.1.4 Tech Manager

As atribuições de gestão dos squads ficam a cargo do Tech Manager. Esse profissional trata da organização dos times, buscando reunir equipes autossuficientes, ao mesmo tempo considerando as capacidades, forças e fraquezas de cada integrante, bem como se eles se sentem satisfeitos e desafiados com suas atuações. Em um ambiente dinâmico como o que a DTI proporciona, é igualmente difícil e crucial aliar as diversas oportunidades aos interesses de cada profissional.

3 ATIVIDADES DESENVOLVIDAS

O estágio descrito neste relatório foi focado, majoritariamente, no desenvolvimento de software. Sendo assim, o estagiário assumiu atribuições análogas às do papel descrito no capítulo anterior como “Desenvolvedor”, na estrutura de cargos descrita anteriormente. Naturalmente, antes de desempenhar as responsabilidades correspondentes à função, houve um processo de familiarização com os ritos e processos da empresa, conforme abordado na Seção 3.1.

Como o acompanhamento perpassou todo o ciclo de desenvolvimento do software, as atividades contemplam também desde o entendimento inicial e a proposta de um produto mínimo viável (MVP), até o momento em que a inclusão de novas funcionalidades deixou de ser prioritária ou vantajosa para o cliente e o software entrou em estágio puramente de manutenção. Entretanto, a parcela mais substancial do tempo do estagiário realmente foi utilizada durante a participação nas atividades de desenvolvimento.

3.1 Familiarização com os Ritos e Processos da Empresa

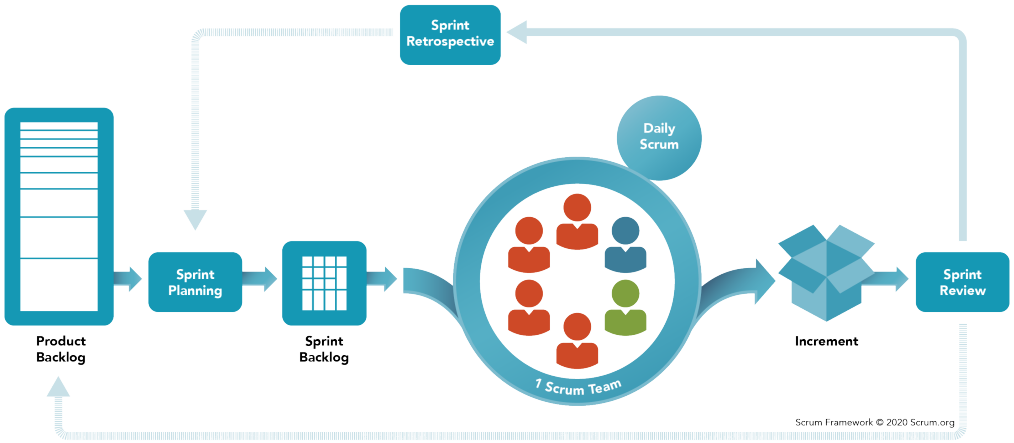
Aqui são descritos os principais processos da DTI, dos quais todo estagiário toma ciência em suas primeiras semanas - e continua executando durante toda a sua estadia. É muito importante que o estagiário esteja satisfatoriamente familiarizado com tais conceitos antes de começar a executar as atribuições do cargo, pois eles orientam como suas atividades devem ser desempenhadas.

3.1.1 SCRUM e o DTI Flow

Este relatório não discute as minúcias conceituais do SCRUM, mas sim como ele é aplicado pela DTI e sua relação com o fluxo de trabalho das equipes - que recebe a alcunha de “DTI Flow”. Tendo isso em mente, é válido definir o SCRUM como um framework que visa padronizar o desenvolvimento de software em ciclos regulares com etapas de entendimento, especificação, implementação e avaliação de funcionalidades do sistema (SCHWABER; SUTHERLAND, 2020). Da mesma forma, cabe dizer que o DTI Flow abraça e expande o SCRUM, incluindo as especificidades que considera interessantes nas lacunas que o SCRUM propositalmente deixa.

A Figura 3.1, retirada da página oficial da organização que mantém o SCRUM, ilustra o seu funcionamento básico, que ocorre em nível de sprints (períodos que vão, habitualmente, de duas a quatro semanas). O fluxo padrão de cada história de usuário da sprint, porém, fica a

Figura 3.1 – Fluxo do SCRUM



Fonte: https://www.scrum.org/resources/what-is-scrum

critério das equipes. É nesse nível que entra o DTI Flow: cada atividade desenvolvida dentro da sprint segue um fluxo que se inicia no entendimento da história, passa pelo planejamento e validação de um roteiro de testes, entra na fase de desenvolvimento e implementação de testes automatizados, chegando, por fim, à etapa de validação do que foi feito. Após passar por todos os estágios mencionados, caso não haja problemas, uma história é dada como pronta.

Por outro lado, vale ressaltar que o DTI Flow, assim como o SCRUM, são geralmente utilizados pelas equipes como forma de orientar os processos cotidianos, mas não são obrigatórios ou imutáveis. Pelo contrário, considerando que os diversos cenários de trabalho são extremamente dinâmicos, a DTI estimula que cada time adapte os processos da forma que melhor lhe convir. Sendo assim, o fluxo supracitado descreve a forma de trabalho da equipe no projeto abordado por este relatório, mas não é necessariamente o modo de operação de todos os demais squads.

3.1.2 Ignitions

A prática de Ignitions faz parte do processo da DTI para familiarização e fixação dos conceitos e valores que a organização considera primordiais. Sendo assim, as Ignitions consistem de apresentações de frequência regular (geralmente mensal) sobre temas como “engenharia”, “produto”, “design”, “testes automatizados”, entre outros. Tais apresentações são de grande importância para os novos colaboradores, visto que não se restringem aos conceitos em si, mas em como eles se aplicam no cotidiano da DTI e porque são considerados essenciais para

o andamento dos processos. Nesse sentido, as Ignitions contam também constantemente com a presença de colaboradores mais experientes – naturalmente, não com o mesmo objetivo de obter o primeiro contato com os valores e processos da empresa, mas de retomá-los e consolidá-los.

3.1.3 Checks Regulares

Um dos pilares do Manifesto Ágil (AGILE, 2001) é o de "Responder às mudanças mais do que seguir um plano". Na prática, isso quer dizer estar sempre atento aos processos e em prontidão para adaptá-los, ou corrigi-los quando eles começam a se afastar do objetivo principal – que é a geração de valor para o cliente. Trazendo isso para o contexto de trabalho, são realizadas regularmente reuniões para a análise de como esses processos estão sendo realizados pela equipe. Durante essas reuniões, a equipe responde questionários com tópicos norteadores para fomentar a discussão sobre aspectos importantes. Via de regra, os questionários são preenchidos individualmente antes da reunião e as respostas são comparadas durante ela, de modo a evitar qualquer viés ou tendência de grupo.

Ao fim dos checks, são gerados planos de ação para melhoria dos pontos falhos mais graves encontrados, para garantir que algo realmente seja feito a respeito e eles sejam, à medida do possível, corrigidos.

3.1.3.1 Check de Engenharia

O Check de Engenharia é mais voltado à equipe técnica (desenvolvedores e arquitetos). Como se pode supor, visa discutir aspectos mais estruturais do software e características que interessam à equipe técnica, como manutenibilidade, modularidade, testes automatizados, logs, ambientes, publicações, versionamento, documentação etc.

3.1.3.2 Check de Produto e Design

Unindo dois pilares bastante sinergéticos, o Check de Produto e Design fomenta discussões a respeito do valor que está sendo entregue ao cliente, como esse valor pode ser mensurado, como ele se faz visível e como a equipe entende o que está produzindo. Além disso, esse momento é importante para avaliar como as atuações dos principais responsáveis pelas duas áreas (Project Owner e Designer) são vistas e assimiladas pelo restante da equipe.

3.1.3.3 Check de Operações

Por sua vez, o Check de Operações trata dos processos em si, permeando todos os demais pilares. Como discutido anteriormente, a DTI possui um fluxo recomendável, que visa nortear todo o processo de trabalho, assimilando a metodologia ágil. O Check de Operações questiona fatores importantes sobre os ritos de modo geral e, acima de tudo, proporciona o alinhamento da equipe sobre a sua situação e sobre os entendimentos e divergências de perspectivas entre todos os membros.

3.1.4 Operation Canvas

O Operation Canvas é um documento que funciona como um contrato informal da equipe a respeito dos seus processos. Nele são descritas as especificidades do modo de operação da equipe, como os ritos que ela realiza, os status de histórias, os horários dos ritos, as formas de comunicação utilizadas etc. Regularmente, este documento é atualizado na presença de todos os integrantes. Além de garantir a concordância da equipe sobre como todos se propõem a trabalhar, este documento transforma-se em um artefato muito útil, em especial, para os novos integrantes.

3.2 Acompanhamento do Processo de Design

Como a DTI trabalha com metodologias ágeis, o processo de design e, mais especificamente, o levantamento de requisitos e a proposta de soluções não ocorrem apenas uma vez, são etapas que se repetem em ciclos contínuos. Todavia, é possível definir no início do projeto uma ideia geral de produto mínimo viável (MVP). No projeto em questão, a rede de laboratórios já tinha em mente várias necessidades bem definidas, distribuídas entre soluções que pretendia substituir e ideias para melhorias e novos fluxos de trabalho. A existência de soluções prévias a serem substituídas forneceu inspirações mais sólidas para o MVP, mas, ao mesmo tempo, criou o risco da replicação de vícios indesejáveis de processos, ou da resistência do cliente em relação à mudança. Sendo assim, o papel do designer teve o desafio de se inspirar parcialmente nas soluções pré-existentes sem se deixar induzir por elas e, simultaneamente, tentando preservar certa familiaridade com os fluxos anteriores.

Um momento destacável do processo de design foi a ideação. Este rito contou com a presença de diversos interessados na solução a ser desenvolvida. A proposta durante as reu-

niões era ter múltiplas perspectivas acerca do problema a ser resolvido, gerando insumos para a definição das principais dores do cliente, ou seja, das suas necessidades mais urgentes e importantes. Vários depoimentos foram coletados e inúmeras experiências relatadas e discutidas. A partir daí, a equipe pôde filtrar aquilo que considerou indispensável para uma proposta inicial de solução - que foi muito bem aceita.

3.3 Desenvolvimento Back-end de Rotas para API

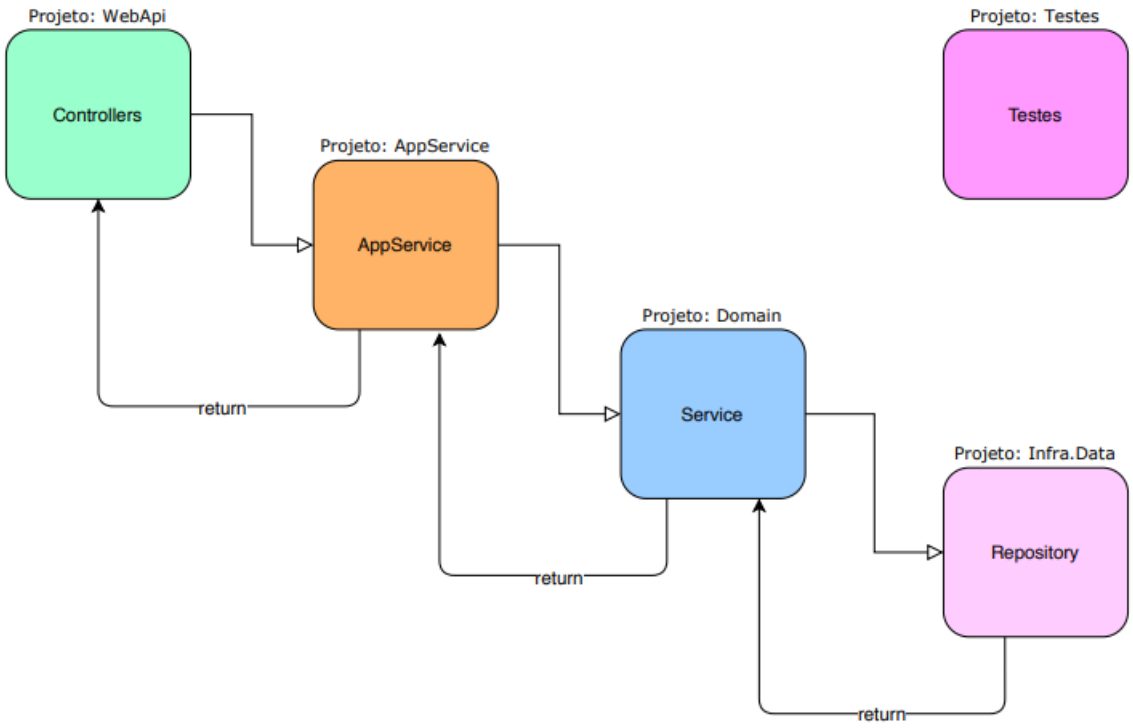
Cabe dizer que a solução desenvolvida tinha a maior parte das suas funcionalidades concentrada no back-end. Como o estágio foi focado nas atribuições e atividades de um desenvolvedor, parte significativa do tempo do estagiário foi também aplicada neste contexto. Também foi pelo back-end que o processo de desenvolvimento do sistema se iniciou, o que fez dele não só o contato majoritário, mas também o inicial em termos de desenvolvimento ágil (MARTIN, 2020).

Um conceito complexo e importante que pôde ser consolidado pelo estagiário nessa etapa foi o de API (REDHAT, 2017). A sigla é traduzida para algo como “Interface de Programação de Aplicação”, e isso por si só não faz muito sentido. Em termos de alto nível, porém, APIs são conjuntos de definições e protocolos que especificam a comunicação entre aplicações com base em contratos bem definidos. Estes contratos definem as entradas esperadas, as possíveis respostas devolvidas e os métodos e protocolos a serem utilizados para que a comunicação aconteça. O desenvolvimento nesses moldes permitiu que muitos conceitos vistos em aula pudessem ser aplicados. Conhecimentos das disciplinas de Redes de Computadores e Sistemas Distribuídos estiveram diretamente relacionados à estrutura de comunicação supracitada. Já no sentido da programação em si, foi possível ver na prática a importância de questões como encapsulamento, isolamento de responsabilidades (MARTIN, 2009), herança, padrões de projeto (GAMMA et al., 2000), o paradigma orientado a objetos, estruturas de dados (CORMEN et al., 2012) etc.

A estrutura básica de divisão do back-end no projeto contava com cinco camadas principais, conforme ilustrado pela Figura 3.2. Vale frisar que essa divisão se deu em um nível de encapsulamento das classes, ou seja, as camadas ainda eram parte de um sistema monolítico. Neste fluxo, a camada de Controllers era responsável por externalizar as rotas da aplicação, também definindo os tipos de entrada e saída dessas rotas; a camada AppService tinha por principal função realizar validações de dados; a camada de Serviços englobava a maior parte das regras

de negócio e manipulações de dados; a camada de Repositórios, por sua vez, definia e realizava acesso a banco de dados e APIs; por fim, a camada de Testes estava apartada das demais porque, na verdade, aplicava-se a todo o fluxo. O estagiário chegou a desenvolver rotas completas, nesse sentido, implementando todas as camadas mencionadas, além dos testes associados a cada uma delas.

Figura 3.2 – Camadas do back-end da aplicação



Fonte: documentação do projeto

O estagiário possuía conhecimentos de experiências anteriores acerca de bancos de dados, tanto pelas disciplinas com esse foco, quanto por já estar inserido no mercado de TI em funções que lidavam com isso. Tais conhecimentos foram colocados à prova e reforçados na implementação de classes de repositório. Nos casos em que os dados eram obtidos e persistidos em um banco de dados próprio da aplicação, as consultas e sentenças ficavam na própria classe de repositório.

Ainda no escopo dos repositórios, alguns deles, como dito, acessavam um banco de dados da própria aplicação, mas não todos. Em determinados cenários, era necessário interagir com APIs de terceiros, como sistemas de adquirentes ou bancos. Nesse sentido, se o estagiário já tinha familiaridade com o desenvolvimento de uma API, aqui teve a chance de entender a outra face da comunicação: o consumo. Foi preciso estudar documentações dos softwares

de terceiros para entender os contratos e configurar as requisições corretamente - inclusive seguindo diferentes protocolos de comunicação e estilos arquiteturais, como SOAP e REST. Esse viés foi importante, pois depender da qualidade da documentação de outros ajudou a entender o quão crucial é também documentar apropriadamente a aplicação que se está desenvolvendo.

3.4 Desenvolvimento Back-end de Rotinas

Ainda no âmbito do back-end, o sistema possuía serviços que deveriam ser executados a intervalos específicos de tempo. Esse cenário é comumente chamado de "rotina". Para implementar esse tipo de código, o estagiário teve que aplicar conceitos de programação assíncrona, bem como entender o funcionamento de bibliotecas que trabalham com tempo e intervalos, como a classe Timer, Microsoft (2022b). De modo geeral, tais bibliotecas usam valores inteiros abstraídos em quantidades de segundos ou frações de segundos a partir de um momento referencial.

Durante esse contato, também ficou evidente para o estagiário a utilidade de seguir padrões de projetos e isolar responsabilidades em camadas. Os serviços necessários eram desenvolvidos independentemente de quando ou por quem deveriam ser utilizados. Assim, o mesmo serviço podia ser acessado periodicamente pela chamada de uma rotina, ou eventualmente por uma requisição externa. A fonte da execução não afetava, nem era afetada, pelo serviço em si. Assim, também a manutenção e os testes dessas camadas se tornavam muito mais simples e específicos.

3.5 Foco em Testes Automatizados e Cobertura do Código

Embora a implementação de testes automatizados fizesse parte do fluxo de desenvolvimento, foi possível perceber após alguns meses de projeto que determinados setores do código estavam ficando debilitados em termos de cobertura e variedade de testes. Tal conclusão foi tirada com o auxílio de um software chamado SonarQube, cujo funcionamento será abordado em mais detalhes no próximo capítulo. O que vale apontar aqui é que essa noção gerou um débito técnico pelo qual o estagiário ficou responsável durante uma das sprints.

Durante o período de duas semanas, o estagiário realizou o levantamento dos arquivos mais deficitários de testes e atuou na implementação de testes para melhorar suas respectivas coberturas. Este contato intensivo favoreceu um entendimento muito mais profundo dos testes e

de seus propósitos. É muito comum que um iniciante se aborreça quando realiza uma alteração simples no código e algum teste quebra - sendo necessário adaptá-lo também. A experiência permitiu perceber que isso é intencional e que ter de adaptar os testes relacionados ao código alterado é simplesmente o preço de saber de imediato quando uma alteração equivocada impacta testes de outros trechos de código como efeito colateral. Na disciplina de Engenharia de Software, estudamos a importância da segurança que os testes proporcionam durante o desenvolvimento e como isso afeta positivamente a qualidade do código, mas ver isso em prática traz um olhar totalmente diferente sobre o assunto.

3.6 Desenvolvimento Front-end de Plataforma de Acompanhamento

Após cerca de quatro meses de desenvolvimento voltado para o back-end, iniciou-se um trabalho para estender o proxy de pagamentos com funcionalidades para que equipes do setor financeiro pudessem utilizá-lo como plataforma de acompanhamento. Como a finalidade do software era intermediar as transações financeiras dos laboratórios, esse viés de acompanhamento tornou-se possível através de uma aplicação front-end que tinha acesso ao back-end já desenvolvido e trabalhava com filtros e relatórios das transações.

O estagiário nessa fase experimentou tecnologias do front-end, como o framework React (REACT, 2022) e a biblioteca Material UI, atualmente conhecida apenas por MUI (MUI, 2022). O desenvolvimento na área de front-end também possibilitou aplicar as boas práticas de programação previamente mencionadas na seção de back-end. Entretanto, houve aqui um desafio adicional: o desenvolvimento de algo com o qual pessoas teriam contato direto. Isso envolveu aplicar também as noções adquiridas na disciplina de Interação Humano-Computador, como usabilidade e acessibilidade. Em outras palavras, houve a oportunidade de pensar nas interações além das telas, ou seja, não só nas necessidades que as interfaces satisfaziam, mas em como elas afetavam o uso pelas diversas pessoas.

Infelizmente, a aplicação do front-end não contou com testes automatizados. Embora muito prejudicial para futuras refatorações (FOWLER, 2020), essa fragilidade existiu por alguns motivos, entre os quais podemos destacar a falta de referências internas com conhecimento adequado de testes para front-end, bem como a simplicidade intrínseca da plataforma de acompanhamento - que tinha poucos componentes e, de modo geral, era focada em operações de consulta. Por outro lado, até mesmo essa experiência foi benéfica para o estagiário, pois instigou reflexões sobre a carência de especialistas com o foco descrito e despertou a sua cu-

riosidade por uma área menos explorada, o que veio a render frutos posteriores ao período de estágio.

3.7 Azure DevOps (Controle de Versionamento e Gestão de Repositório)

Embora o Azure DevOps seja uma ferramenta (cabendo, portanto, na seção 4.4), seria quase incoerente encerrar um capítulo de atividades desenvolvidas sem mencioná-lo. Os softwares de controle de versionamento e gestão de repositório são ferramentas indispensáveis para o trabalho de desenvolvimento em equipes. Foi um passo fundamental para o estagiário entender os princípios de uso e as operações básicas de uma ferramenta desse gênero. Conceitos como repositórios, branches, pull requests, commits, entre outros, são de entendimento relativamente fácil, tornando-se naturais para profissionais mais experientes. No entanto, podem parecer confusos ou intimidadores para alguém que está começando a caminhar na área de desenvolvimento.

Entrando no aspecto do fluxo de trabalho, a equipe utilizava, por padrão, duas *branches* de referência (Develop e Main). Ao realizar uma tarefa, um desenvolvedor criava uma nova *branch* local a partir da Develop e, posteriormente, realizava um *Pull Request* com as alterações de sua *branch* local de volta para a própria Develop. Ao final do período de desenvolvimento, as modificações (devidamente validadas) presentes na Develop eram transferidas para a Main, que espelhava o ambiente produtivo.

Ainda que as IDEs disponibilizem funcionalidades para tornar mais "amigável"a gestão de repositório, o estagiário optou por memorizar as operações básicas por linha de comando, pois assim seria menos impactado ao ter que trocar de IDE.

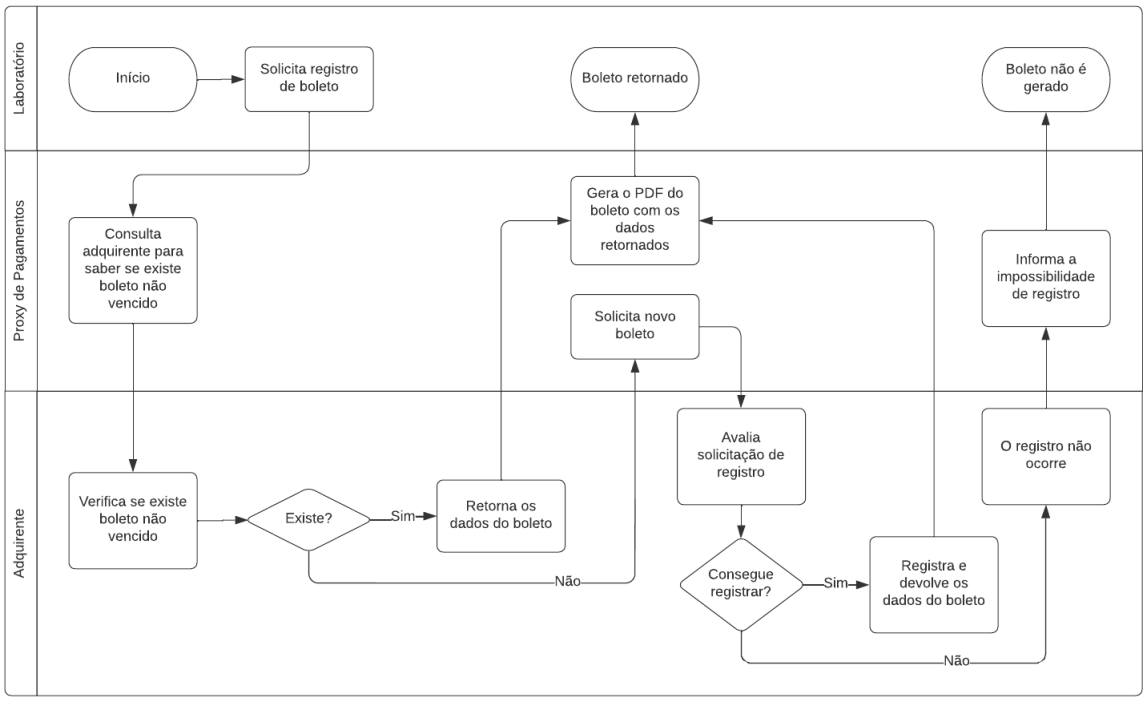
4 ASPECTOS TÉCNICOS ENVOLVIDOS

4.1 Estrutura de Comunicação com Outras Aplicações

A natureza do Proxy de Pagamentos, como sistema intermediário, demandava que ele se comunicasse com diversas outras aplicações. Na Figura 4.1 é possível visualizar um exemplo de fluxo de comunicação entre três "atores", envolvendo o próprio Proxy - trata-se do fluxo de registro de boletos.

O fluxo em questão demonstra uma pequena otimização do processo que gerou benefícios ao cliente: antes de solicitar um novo registro de boleto (que teria custos), o Proxy consultava o sistema do adquirente para verificar a existência do mesmo boleto com data de vencimento em aberto. Quando esse boleto existia, ele era retornado, evitando um novo registro desnecessário.

Figura 4.1 – Geração de boletos - exemplo de fluxo de comunicação



Fonte: autor do relatório

Os boletos não eram o único tipo de pagamento contemplado pelo Proxy, mas sim o mais complexo. Demais tipos de pagamento, como crédito, débito e pix apresentavam uma estrutura de comunicação similar ou mais simples, portanto não se viu grandes ganhos em apresentá-los aqui.

4.2 Tecnologias Utilizadas

O Proxy de Pagamentos não foi o primeiro projeto desenvolvido pela DTI para o cliente em questão. Sendo assim, com uma base de projetos já consolidada, não se fez necessário discutir muito sobre linguagens, frameworks, plataformas e, de modo geral, tecnologias a utilizar. A opção mais natural foi manter as escolhas padronizadas em relação aos demais projetos, pois eles também serviriam de referência para implementações futuras. Além disso, essa padronização diminuiria a curva de aprendizado em caso de movimentações dos desenvolvedores entre outros projetos do mesmo cliente.

4.2.1 Plataforma .Net (Back-end)

O .NET Framework é uma plataforma de desenvolvimento da Microsoft que agrupa diversas linguagens, bibliotecas e ferramentas (como C#, VB.NET, F# etc.) (MICROSOFT, 2022a). Até então, o estagiário havia experimentado no curso de Sistemas de Informação linguagens como Java, Python e C++, mas o presente projeto foi seu primeiro contato com C#. Apesar de algumas diferenças naturais, é válido dizer que os paradigmas e a lógica de programação se sobressaem às linguagens em termos de desafios de aprendizado. Portanto, o contato inicial com uma linguagem se torna muito menos assustador e problemático do que seria sem os fundamentos trabalhados pelo curso.

4.2.2 React (Front-end)

O React (REACT, 2022), por sua vez, é uma biblioteca de código aberto baseada na linguagem JavaScript, mantida por empresas como Facebook e Instagram, além de uma sólida comunidade de desenvolvedores independentes. Sua finalidade principal é voltada para o desenvolvimento de páginas Web. Trata-se de mais uma tecnologia bastante utilizada atualmente, com a qual o aluno possivelmente não teria contato antes de ser formar, não fosse pela oportunidade de estágio.

4.3 Escolhas Arquiteturais

Esta seção discute algumas das escolhas arquiteturais do projeto. Pretende-se expor os dilemas e as dificuldades que requisitaram tais escolhas. No mesmo sentido, são abordadas vantagens e desvantagens de tais escolhas e os pontos que foram cruciais para as decisões.

4.3.1 Modelo do Back-end em Camadas

Conforme explicado na Seção 3.3, o back-end do Proxy de Pagamentos foi desenvolvido e estruturado com uma abstração de cinco camadas. Sem dúvidas, existem inúmeros padrões arquiteturais (MARTIN, 2019) conhecidos, todos eles com benefícios e desvantagens a considerar. Os aspectos que mais pesaram para a escolha em questão foram a simplicidade da abstração, o custo-benefício em termos de ausência de acoplamento e ganhos de coesão e as possibilidades de reaproveitamento de código. Comparada a outros padrões, a arquitetura implementada é bem próxima de um MVC (Modelo, Visão e Controle), com o isolamento adicional da camada para validação de dados.

4.3.2 Padrão de Testes AAA

O uso deste padrão não foi um combinado formal entre os desenvolvedores, mas foi organicamente aplicado em boa parte dos testes unitários presentes. Os três A's fazem referência aos termos do inglês "*Arrange, Act, Assert*", ou em tradução livre "Arranjar, Agir e Aferir". Trata-se de um padrão bem simples e conhecido para testes unitários, que visa dividir a implementação dos testes em três etapas: a preparação de retornos e mocks, como são chamados dados falsos que mimetizam parâmetros em situações reais (arranjar), a chamada de métodos a serem testados (agir) e a averiguação em relação aos resultados alcançados por esses métodos (aferir). Na ausência de um consenso a respeito de padrões de teste, o padrão AAA mostrou-se bem efetivo em termos de organização do código e legibilidade.

4.3.3 Projeto Back-end for Front-end

Conforme discutido nas seções 3.6 e 4.1, o Proxy de Pagamentos não apenas era responsável por intermediar transações financeiras entre sistemas distintos, como por disponibilizar uma plataforma para acompanhamento dessas transações. Com o surgimento da necessidade dessa plataforma front-end, surgiu o questionamento se seria uma boa opção a plataforma consumir os mesmos contratos de API já implementados e disponibilizados para terceiros. Visando a diminuição de acoplamento, a conclusão foi de que as necessidades do front-end seriam bem mais simples e específicas do que as dos laboratórios que consumiam a API do back-end. Sendo assim, optou-se por criar um projeto BFF (Back-end For Front-end) - cuja única responsabilidade é alimentar um front-end específico.

Entre os benefícios dessa estratégia, podemos destacar como o mais relevante a independência das interfaces de comunicação dos projetos. As necessidades de alteração da Web-API original não afetavam o BFF e vice-versa.

4.4 Ferramentas e Aplicações Auxiliares

A presente seção discorre sobre algumas ferramentas e aplicações auxiliares que estiveram presentes nas atividades diárias de desenvolvimento do estagiário.

4.4.1 SonarQube

SonarQube (SONARQUBE, 2022) é uma plataforma que realiza análise estática de código, evidenciando problemas como bugs e code smells, além de fornecer estatísticas a respeito da cobertura de código dos testes automatizados. É possível incluir a sua execução nos fluxos de entrega e integração contínuas, de modo que ele quebre a execução caso o novo código apresente problemas, ou caso a cobertura de testes do código após o merge não atenda aos patamares definidos para o projeto. Nesse sentido, mostra-se como um auxílio muito bem-vindo na busca pela qualidade de código.

4.4.2 Insomnia

Quando trabalhamos com aplicações que se comunicam, é uma boa prática testar as possíveis requisições que um sistema deve aceitar antes de iniciar a implementação das rotas. Embora as APIs geralmente possuam documentações bem completas, não é raro que alguma informação se perca durante manutenções e deixe de ser documentada. Seria, no mínimo, frustrante ter que implementar uma rota em sua completude antes de descobrir que ela não funcionará, pois a ausência de algum campo desconhecido torna a sua requisição inválida.

Nesse sentido, existem ferramentas bem conhecidas como Postman (POSTMAN, 2022) e Insomnia (INSOMNIA, 2022), que possibilitam configurar e simular requisições a APIs sem depender de qualquer implementação. A equipe deste projeto padronizou o uso do Insomnia e isso possibilitou a exportação de pacotes de configurações. Nesse sentido, com a chegada de algum novo integrante, bastava importar em sua ferramenta o compilado de rotas que era frequentemente atualizado pela equipe.

5 CONCLUSÃO

Com o constante lançamento de novas tecnologias e a pluralidade de áreas de atuação que a Tecnologia da Informação oferece aos seus profissionais, é quase impossível para um curso de graduação acompanhar as tendências e aprofundar-se nas minúcias de tantos conteúdos. Nesse sentido, o que está ao alcance da graduação é fornecer aos alunos alicerces consistentes para que, por iniciativa própria, sejam capazes de aperfeiçoar-se nas áreas que desejarem. Não obstante, é comum que essa situação gere insegurança a muitos alunos que desejam entrar no mercado de trabalho. Assim, o estágio apresenta-se como uma ótima maneira de complementar a formação e prover a confiança e a bagagem necessárias para o prosseguimento na área.

A oportunidade de realizar um estágio juntamente com o curso de Sistemas de Informação foi extremamente relevante para a formação do estagiário. Foi possível colocar em prática aprendizados adquiridos durante toda a graduação, em especial a respeito da qualidade e do desenvolvimento de software. De forma similar, também tornou-se viável entender muitos dos processos utilizados instintivamente, graças a conhecimentos adquiridos nas disciplinas ofertadas. Ademais, embora o estagiário já se encontrasse inserido profissionalmente no mercado da Tecnologia da Informação, o estágio permitiu a mudança de foco da sua carreira para a área que almejava: o desenvolvimento de software.

Analisando mais especificamente as disciplinas oferecidas pelo curso, o estágio permitiu a aplicação de diversos conceitos da área de programação, desde os fundamentos de Introdução aos Algoritmos e de Estruturas de Dados, até tópicos mais avançados, como os padrões de projeto abordados em Práticas de Programação Orientada a Objetos. Além disso, o estágio indubitavelmente foi amparado pelos conhecimentos de Engenharia de Software, Qualidade de Software, Interação Humano-Computador e Gerência de Projetos de Software, no sentido da qualidade e usabilidade das entregas.

Em suma, colaborar com o desenvolvimento de software, na companhia de profissionais excelentes em um contexto real do mercado de trabalho é algo que mudou drastica e permanentemente a perspectiva do estagiário, reforçou suas aspirações sobre continuar na área e proporcionou sentido a todo o conhecimento obtido com o curso de Sistemas de Informação.

REFERÊNCIAS

AGILE. **Agile**. 2001. Disponível em: <http://agilemanifesto.org/iso/ptbr/manifesto.html>. Accesso em: 2022-08-31.

CORMEN, T. H. et al. **Algoritmos**: Teoria e prática. 3. ed. São Paulo: Grupo GEN, 2012.

DTI. **O que fazemos**. 2022. Disponível em: <https://www.dtidigital.com.br/o-que-fazemos/>. Accesso em: 2022-08-31.

DTI. **Quem somos**. 2022. Disponível em: <https://www.dtidigital.com.br/venha-ser-dti/>. Accesso em: 2022-08-31.

FOWLER, M. **Refatoração**: Aperfeiçoando o design de códigos existentes. 2. ed. São Paulo: Novatec, 2020.

GAMMA, E. et al. **Padrões de Projeto**: Soluções reutilizáveis de software orientados a objetos. 1. ed. Porto Alegre: Bookman, 2000.

INSOMNIA. **Insomnia**. 2022. Disponível em: <https://insomnia.rest>. Accesso em: 2022-08-31.

MARTIN, R. C. **Código Limpo**: Habilidades práticas do agile software. 1. ed. Rio de Janeiro: Alta Books, 2009.

MARTIN, R. C. **Arquitetura limpa**: O guia do artesão para estrutura e design de software. 1. ed. Rio de Janeiro: Alta Books, 2019.

MARTIN, R. C. **Desenvolvimento ágil limpo**: De volta às origens. 1. ed. Rio de Janeiro: Alta Books, 2020.

MICROSOFT. **Microsoft .Net**. 2022. Disponível em: <https://dotnet.microsoft.com/en-us/>. Accesso em: 2022-07-31.

MICROSOFT. **Timer Class**. 2022. Disponível em: <https://docs.microsoft.com/en-us/dotnet/api/system.timers.timer?view=net-6.0>. Accesso em: 2022-07-30.

MUI. **MUI**. 2022. Disponível em: <https://mui.com/pt/>. Accesso em: 2022-08-31.

POSTMAN. **Postman**. 2022. Disponível em: <https://www.postman.com>. Accesso em: 2022-08-31.

REACT. **React**. 2022. Disponível em: <https://pt-br.reactjs.org>. Accesso em: 2022-08-06.

REDHAT. **O que é uma API**. 2017. Disponível em: <https://www.redhat.com/pt-br/topics/api/what-are-application-programming-interfaces>. Accesso em: 2022-07-23.

SCHWABER, K.; SUTHERLAND, J. **Scrum Guide**. [S.l.], 2020. Disponível em: <https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-PortugueseBR-3.0.pdf>.

SONARQUBE. **SonarQube**. 2022. Disponível em: <https://www.sonarqube.org>. Accesso em: 2022-08-31.