Módulo de gestão financeira e otimização de algoritmo de cálculo de contas

Kaio Duarte dos Santos

Relatório Final de Estágio apresentado à Escola Superior de Tecnologia e de Gestão de Bragança para obtenção do Grau de Mestre em Sistemas de Informação no âmbito da dupla diplomação com a Universidade Tecnológica Federal do Paraná.

Trabalho realizado sob a orientação de:

Prof. Doutor Paulo Alexandre Vara Alves

Prof. Doutor José Eduardo Moreira Fernandes

Marcin Włodarczyk

Este relatório não inclui as críticas e sugestões feitas pelo Júri.

Bragança

Fevereiro de 2020

Módulo de gestão financeira e otimização de algoritmo de cálculo de contas

Kaio Duarte dos Santos

Relatório Final de Estágio apresentado à Escola Superior de Tecnologia e de Gestão de Bragança para obtenção do Grau de Mestre em Sistemas de Informação no âmbito da dupla diplomação com a Universidade Tecnológica Federal do Paraná.

Trabalho realizado sob a orientação de:

Prof. Doutor Paulo Alexandre Vara Alves

Prof. Doutor José Eduardo Moreira Fernandes

Marcin Włodarczyk

Este relatório não inclui as críticas e sugestões feitas pelo Júri.

Bragança

Fevereiro de 2020

Resumo

O setor imobiliário português tem apreciado um desenvolvimento significativo ao longo dos últimos anos. Neste contexto, a empresa Riskivector atua desde 2009 no atendimento das demandas de estudantes, sobretudo aqueles advindos de programas de mobilidade estudantil. Atrelado a isso, o atual cenário do mercado empresarial exige respostas rápidas e eficazes das organizações, sendo, portanto, imprescindível a eficiência dos processos internos. Dessa forma, visando sanar a ineficiência detectadas nos processos da companhia em questão, o presente trabalho tem como objetivo apresentar os novos módulos desenvolvidos para gerenciar movimentações financeiras e prover meios de otimizar a precisão do algoritmo de cálculo de contas, bem como relatar a experiência adquirida ao longo desse processo. Para alcançar tais metas, foi feito um processo etnográfico na empresa a fim de coletar requisitos adequados e desenhar casos de uso. Os requisitos avaliados tiveram os resultados esperados, enquanto que para os demais estima-se que trarão impacto positivo no fluxo de trabalho, proporcionando comodidade para os clientes e diminuindo os esforços dos colaboradores da empresa.

Palavras-chave: Aplicação web, gestão financeira, estágio.

Abstract

The Portuguese real estate sector has enjoyed a significant development over the last few years. In this context, Riskivector has been active since 2009 in meeting the demands of students, especially those arising from student mobility programs. Linked to this, the current scenario of the business market demands fast and effective responses from organizations, and therefore the efficiency of internal processes is essential. Thus, in order to remedy the inefficiency detected in the processes of the company in question, this work aims to present the new modules developed to manage financial movements and provide means to optimize the accuracy of the account calculation algorithm, as well as to report the experience acquired throughout this process. To achieve these goals, an ethnographic process has been done in the company in order to collect appropriate requirements and design use cases. The evaluated requirements had the expected results, while for the others it is estimated that they will have a positive impact on the workflow, providing convenience for customers and reducing the efforts of the company's employees.

Keywords: Web application, financial management, internship.

Conteúdo

1 Introdução **1**

1.1 Enquadramento 1

1.2 A Empresa 1

1.3 Objetivos 2

1.4 Estrutura do Documento 2

2 Revisão da Literatura **3**

2.1 Gestão Financeira 3

2.1.1 Fluxo de Caixa 4

2.2 *World Wide Web* 5

2.3 Protocolo HTTP 6

2.3.1 *Uniform Resource Locator* 7

2.3.2 Métodos 8

2.3.3 Códigos de Estado 8

2.3.4 Cabeçalho 9

2.4 *HyperText Markup Language* 10

2.5 *Cascading Style Sheet* 11

2.5.1 Pré-processadores 12

2.6 JavaScript 13

2.6.1 NodeJS 13

2.6.2 TypeScript 14

2.7 Serviços web 15
2.7.1 Simple Object Access Protocol. 15
2.7.2 Representational State Transfer 16
2.8 Single Page Application 18
2.8.1 Virtual DOM 20
2.8.2 Change Detector 20
2.9 Business Process Model and Notation. 22
2.9.1 Elementos BPMN 22
2.9.1.1 Objetos de uxo 23
2.9.1.2 Objetos de conexão 24
2.9.1.3 Dados 24
2.9.1.4 Swimlanes. 25
2.9.1.5 Artefatos 26
2.10 Tecnologias 26
2.10.1 ExpressJS 27
2.10.2 SequelizeJS 27
2.10.3 Angular 28
2.10.4 Git 28
2.10.5 Docker 29

3 Análise de Requisitos e Modelação 31
3.1 Âmbito do projeto 31
3.2 Mapeamento de processos de negócio 32
3.3 Requisitos 38
3.3.1 Requisitos Funcionais 38
3.3.2 Requisitos Não Funcionais 41
3.3.3 Diagrama de casos de uso 42
3.3.4 Diagrama de classes 44
3.4 Considerações nais 47

4 Desenvolvimento e Resultados 49
4.1 Trabalho Colaborativo e Integração Contínua ao Projeto Existente 49
4.1.1 Git 49
4.1.2 GitLab 50
4.1.3 Docker 51
4.2 Restruturação da Arquitetura do Serviço Web 51
4.2.1 Arquitetura de camadas 51
4.2.1.1 Camada de apresentação 52
4.2.1.2 Camada de regras de negócios e persistência 52
4.2.1.3 Camada de banco de dados 52
4.3 Limitações de Bibliotecas de Terceiros 53
4.3.1 SequelizeJS 53
4.4 Módulo de gestão nanceira 53
4.4.1 Interfaces para gerência de movimentações e emissão de contas dos inquilinos 54
4.4.2 Interfaces disponíveis aos funcionários 58
4.4.3 Interfaces do administrador 60
4.4.4 Integração com provedor de pagamento e faturação 68
4.5 Otimização no algoritmo de cálculo de contas 69
4.6 Considerações nais 75

5 Conclusões e trabalhos futuros 77

Lista de Tabelas

2.1 Métodos HTTP. 8
2.2 Códigos e frases de estado de mensagens de resposta do protocolo HTTP. . 9
2.3 Cabeçalhos básicos do protocolo HTTP. 10

Lista de Figuras

2.1 Visão do Fluxo Econômico Simples. 5
2.2 Solicitação de documento através do método GET do protocolo HTTP. Fonte: Forouzan (2010). 7
2.3 Fluxo de aplicações web tradicionais. Fonte: Wasson (2013). 18
2.4 Fluxo de aplicações web com o uso de requisições AJAX. Fonte: Wasson (2013). 19
2.5 Mecanismo de detecção de mudanças utilizado no VDOM. Fonte: Frachet (2018). 20
2.6 Árvore de componentes e seus respectivos CDs. Fonte: Arora (2018). . . . 21
2.7 Árvore de componentes durante a detecção de um evento e execução do CDs. Fonte: Arora (2018). 22
2.8 Exemplos de eventos do BPMN, na sequência por linha: evento de início, intermediário, de nalização, de início a partir de uma mensagem, de início a partir de um temporizador, e de nalização com o envio de uma mensagem. 23
2.9 Exemplos degatewaysdo BPMN, na sequência por linha:gatewayexclusivo, paralelo, inclusivo, complexo, e baseado em eventos. 24
2.10 Objetos de conexão do BPMN, na sequência: uxo de sequência, de mensagens e associações. 24
2.11 Elementos de dados do BPMN, na sequência: objeto de dado, objeto de entrada de dado, objeto de saída de dado e repositório de informações. . . 25
2.12 Pool com duaslanes 25

2.13 Exemplo de agrupamento de elementos BPMN com a nalidade de documentação. 26
2.14 Exemplo de anotações de texto a elementos BPMN. 26

3.1 Processo de negócio de transferência de dinheiro entre funcionários. 34
3.2 Exemplo de registro e célula acumuladora de valores presentes no uxo de caixa da companhia. 36
3.3 Processo de negócio de registro de movimentações feito pelo gerente. 36
3.4 Processo de negócio de emissão de contas. 37
3.5 Processo de negócio de pagamento de contas dos inquilinos. 37
3.6 Processo de negócio de registro e envio de faturas. 38
3.7 Diagrama de casos de uso. 43
3.8 Diagrama de classes dos novos módulos. 45
3.9 Pacote Fluxo de caixa com ênfase na entidade Transaction. 46

4.1 Modelo de trabalho Git Flow. Fonte: Driessen (2010). 50
4.2 Quadros de atividades do GitLab. 51
4.3 Arquitetura de camadas escolhida para o projeto. Adaptado de Richards (2015). 52
4.4 Interface com movimentações (contas e pagamentos) de um inquilino. . . . 55
4.5 Diagrama ER das entidades envolvidas nas movimentações do inquilino. . . 55
4.6 Interface para emissão de contas para um inquilino: Primeiro passo, escolha do tipo de conta a emitir. 56
4.7 Interface para emissão de contas para um inquilino, tipo Tenant Bill. . . 57
4.8 Interface para emissão de contas para um inquilino, tipo Tenant Miscellaneous Bill. 57
4.9 Interface com per l do funcionário. 58
4.10 Interface com as transações do colaborador. 59
4.11 Interface do funcionário para a criação de transações. 60
4.12 Interface do administrador com as transações da companhia. 61

4.13 Interface de visualização das informações detalhadas de uma transação. . . 62
4.14 Interface de atualização de uma movimentação. 63
4.15 Interface do administrador para criação de transações. 64
4.16 Interface para criação de transações, cujo método de pagamento escolhido é Cash e o tipo escolhido é Tenant Payment (1/3). 65
4.17 Interface para criação de transações, cujo método de pagamento escolhido é Cash e o tipo escolhido é Tenant Payment (2/3). 66
4.18 Interface para criação de transações, cujo método de pagamento escolhido é Cash e o tipo escolhido é Tenant Payment (3/3). 67
4.19 Interface para criação de transações, cujo método de pagamento escolhido é Bank Transfer e o tipo escolhido é Other Income. 68
4.20 Folha de cálculo utilizado para o cálculo de contas de um apartamento. . . 71
4.21 Interface com os BCTs. 72
4.22 Interface de criação de BCTs. 73
4.23 Ferramenta de linha de comando criada para auxiliar na progressão do algoritmo de cálculo de contas. 74
4.24 Resultados de execução da ferramenta de linha de comando. 75

Siglas

AJAX Asynchronous JavaScript And XML. xiii, 18, 19

API Application Programming Interface. 13, 16, 32, 53

BCD Bills Calculation Data. 71

BCR Bills Calculation Result. 71

BCT Bills Calculation Test. xv, 41, 71 73

BPMN Business Process Model And Notation. 22, 33, 42

CD Change Detector. x, xiii, 19 22

CDG Caixa Geral de Depósitos. 68

CERN European Center for Nuclear Physics. 5

CSS Cascading Style Sheet. 11, 12

DOM Document Object Model. 10, 19, 20

ECMA European Computer Manufacturer's Association. 13

ER Entidade Relacionamento. xiv, 55

FTP File Transfer Protocol. 7

HTML HyperText Markup Language 6, 10, 11, 18, 19

HTTP HyperText Transfer Protocol. 6, 15, 17, 27

IPB Instituto Politécnico de Bragança. 1, 2

LESS Leaner Style Sheet 12

MPA Multiple Page Application. 18, 19

OMG Object Management Group 22

ORM Object Relational Mapping 27, 53

POS Point of Sale. 35

REST Representational State Transfer. x, 15 17, 68

RFC Request for Comments 6

RPC Remote Procedure Call 15

SASS Syntactically Awesome Style Sheets. 12

SMTP Simple Mail Transfer Protocol. 7

SOAP Simple Object Access Protocol. x, 15, 16, 68

SPA Single Page Application. 19, 28

TCP Transmission Control Protocol. 6

UML Uni ed Modeling Language. 42

URI Uniform Resource Identi er. 16

URL Uniform Resource Locator. ix, 6, 7

VDOM Virtual DOM . x, xiii, 19, 20, 22

W3C World Wide Web Consortium 6, 12

WSDL Web Services De nition Language 15

WWW World Wide Web. 5, 6, 11

XHTML Extensible HyperText Markup Language 11

XML Extensible Markup Language 11, 15

Capítulo 1

Introdução

1.1 Enquadramento

As constantes mudanças nas mais diversas áreas de atuações, incluindo o cenário econômico, impulsionam as organizações a buscar qualidade nas informações gerenciais, se transformando em um fator determinante para a sua sobrevivência e continuidade no mercado. Portanto, a melhoria dos processos internos é de extrema importância para a estratégia das empresas.

Sendo assim, o escopo deste trabalho é de nido pelo projeto, desenvolvimento e testes de novos módulos que, aplicando o conceito de gestão nanceira e engenharia de software, viabilizem o planejamento e o controle dos recursos nanceiros, proporcionando uma visão clara da administração do capital da empresa.

1.2 A Empresa

A Riskivector, criada em 2009 e incubada no Instituto Politécnico de Bragança (IPB) desde 2010, atua no ramo imobiliário com foco em atender as demandas de estudantes e/ou turistas através de atividades de exploração e intermediação de alojamentos locais, além de desenvolver ainda atividades de manutenções gerais e serviços de gestão e apoio em reservas de alojamento, viagens e eventos.

O crescimento no número de alunos do IPB advindos de programas de mobilidade estudantil ao longo dos anos impulsiona o mercado local e, consequentemente, as atividades da empresa que continua em expansão e conta com escritórios em Mirandela e Vila Real.

1.3 Objetivos

Este relatório tem como objetivo apresentar os novos módulos desenvolvidos para o sistema web vigente na empresa durante a realização do estágio. Para a conclusão deste objetivo, os seguintes objetivos especí cos foram propostos:

- Desenvolvimento de um módulo de gestão nanceira;
- Desenvolvimento de mecanismos para otimizar a precisão do algoritmo de cálculo de contas dos clientes;
- Realizar um processo etnográ co na empresa a m de coletar requisitos adequados;
- Realizar avaliações com funcionários a m de obter informações sobre a usabilidade do sistema desenvolvido.

1.4 Estrutura do Documento

Este trabalho está organizado da seguinte forma: O Capítulo 2 apresenta uma fundamentação teórica sobre o desenvolvimento de aplicações web, gestão nanceira e modelação de soluções através dos processos de negócio. O Capítulo 3 apresenta uma análise do processo de negócio, o qual permitiu a de nição dos requisitos funcionais e não funcionais e modelação de casos de uso.

Após a fase de modelação da aplicação, no Capítulo 4 é descrito o processo de desenvolvimento e resultados do sistema deste trabalho. Por m, o Capítulo 5 apresenta as considerações nais deste projeto e sugestões de trabalhos futuros a serem realizados.

Capítulo 2

Revisão da Literatura

O entendimento adequado deste documento exige, inicialmente, que sejam apresentados os conceitos e as ferramentas utilizadas, para que sirvam de base para o corpo do trabalho. Portanto, neste capítulo são descritos alguns dos principais conceitos pertinentes à gestão nanceira, construção de aplicações web e análise e modelagem de processos de negócios.

2.1 Gestão Financeira

Para alcançar o objetivo de uma empresa, isto é, criação de valor para seus acionistas, é preciso planejar, executar e controlar recursos. Na concepção de Arantes (1998), tais ações associadas a instrumentos de gestão constituem um ferramental que contribui para a e cácia e e ciência da administração empresarial.

Entre os instrumentos de gestão os quais permitem acompanhar as entradas e as saídas de recursos nanceiros da empresa e o saldo de caixa estão o uxo caixa, controle diário de caixa, controle bancário, controle de estoque, etc. Em virtude da exibilidade de adoção em empresas de diferentes portes e necessidades, o uxo de caixa será explorado com mais detalhes neste trabalho.

2.1.1 Fluxo de Caixa

Kuhn (2012) de ne uxo de caixa como o registro ordenado no tempo, do total das entradas e saídas de caixa de uma empresa, representando um instrumento auxiliar para a correta gestão dos recursos nanceiros. O objetivo básico desse recurso é controlar a capacidade de pagamento da empresa, isto é, sua liquidez.

Adjacente a isso, no entendimento de Gitman (2006), os uxos de caixa da empresa podem ser divididos três em categorias, sendo elas:

- Fluxos operacionais :
  Entradas e saídas diretamente associadas à venda e à produção de bens e serviços pela empresa, sendo, portanto, o mais importante.

- Fluxos de investimentos :
  Fluxos associados à compra e à venda de ativos imobilizados e participações em outras empresas.

- Fluxos de nanciamento :
  Fluxos resultantes de operações de captação de recursos de terceiros e capital próprio.

A Figura 2.1 ilustra os uxos de caixa básicos de uma empresa.

Figura 2.1: Visão do Fluxo Econômico Simples.

Nessa linha de raciocínio, atualmente exige-se que os patrimônios sejam mais bem geridos devido aos avanços dos sistemas de informação e a alta competitividade do mercado, promovendo assim informações mais con áveis, as quais diminuem o risco nos empreendimentos.

2.2 World Wide Web

De acordo com Tanenbaum (2003), a World Wide Web (WWW) é uma estrutura arquitetônica que permite o acesso a documentos vinculados espalhados por milhares de máquinas na Internet. Essa estrutura foi criada pela European Center for Nuclear Physics (CERN) e surgiu em 1989, a partir da necessidade de colaboração nas pesquisas entre os cientistas de diversos países através da troca de relatórios, plantas, fotos e outros documentos.

Na década de 1990, quando a Internet deixou de ser utilizada apenas nas universidades e foi levada até aos computadores dos lares e empresas, surgiram as primeiras

implementações dos componentes base da Web sendo eles: uma linguagem de marcação, a HyperText Markup Language (HTML); um protocolo para comunicação entre computadores, o HyperText Transfer Protocol (HTTP); e um programa utilizado para visualizar os documentos, o navegador (Kurose, 2013).

Devido à popularização da Web, iniciou-se uma era de desenvolvimento de software para a rede de computadores, alavancando empresas como Netscape Communications Corp. e Microsoft. A m de padronizar os protocolos e diretrizes que garantam o crescimento Web (Consortium, 2017), foi criado, em 1994, o World Wide Web Consortium (W3C).

2.3 Protocolo HTTP

O HTTP é um protocolo da camada de aplicação para sistemas de informações hipermídia distribuídos e colaborativos (Force, 1999), utilizado na WWW e é de nido nas Request for Comments s (RFCs) 1945 e 2616. Ademais, o HTTP é um protocolo sem estado (stateless) e utiliza os serviços do Transmission Control Protocol (TCP), como o estabelecimento de conexões entre cliente e servidor, para a transmissão de dados.

Nesse sentido, a comunicação se dá a partir da troca de mensagens, as quais possuem formatos bem de nidos. Sendo elas uma mensagem de solicitação (ou requisição) por parte do cliente, onde o servidor responde enviando uma mensagem de resposta.

A Figura 2.2 ilustra como se dão as comunicações no protocolo HTTP, mostrando os principais elementos das mensagens: métodos, cabeçalhos, Uniform Resource Locator (URL) e código de estado. Estes elementos serão apresentados nas subseções seguintes.

Figura 2.2: Solicitação de documento através do método GET do protocolo HTTP. Fonte: Forouzan (2010).

2.3.1 Uniform Resource Locator

Para identi car uma página web é preciso informar o seu nome, onde ela está localizada e como acessá-la, para isso foram criados os URLs que funcionam como um nome universal da página (Tanenbaum, 2003). Os URLs têm a seguinte estrutura:

Protocolo :// Host :Porta / Caminho

- Protocolo : é o programa usado para acessar os documentos, entre estes protocolos estão o File Transfer Protocol (FTP), Simple Mail Transfer Protocol (SMTP) e HTTP.
- Host : é o computador que hospeda as informações e esse geralmente é representado por um nome alternativo, também conhecido como domínio.
- Porta : um URL pode conter o número da porta do servidor.

- Caminho : informa a localização do arquivo no qual as informações estão armazenadas.

2.3.2 Métodos

Embora o HTTP tenha sido projetado para utilização na Web, ele foi criado de modo mais abrangente, visando as futuras aplicações orientadas a objetos (Tanenbaum, 2003). Por essa razão, são aceites operações chamadas Métodos , classi cados na Tabela 2.1, diferentes da simples solicitação de uma página da web.

Nome	Descrição
GET	Solicita um documento ao servidor
HEAD	Solicita informações sobre um documento
POST	Envia informações para o servidor
PUT	Envia informações para o servidor que podem alterar algum documento
TRACE	Ecoa uma solicitação que chega
CONNECT	Método reservado para usos futuros com proxy
DELETE	Solicita a remoção de um documento no servidor
OPTIONS	Solicita detalhamento sobre opções disponíveis

Tabela 2.1: Métodos HTTP.

2.3.3 Códigos de Estado

O código de estado é um elemento presente na mensagem de resposta e é composto por três dígitos e uma frase de estado, os quais descrevem o estado da requisição recebida. Estão apresentados na Tabela 2.2 os códigos e frases mais comuns.

Código	Frase	Descrição
200	OK	Requisição bem-sucedida
301	Moved Permanently	Objeto requisitado foi removido em de nitivo e o novo URL é especi cado na resposta
400	Bad Request	Indica que a requisição não pôde ser entendida pelo servidor
404	Not Found	O documento requisitado não existe no servidor
500	Internal Server Error	Há um erro, como um crash, por exemplo, no servidor

Tabela 2.2: Códigos e frases de estado de mensagens de resposta do protocolo HTTP.

2.3.4 Cabeçalho

O cabeçalho permite a troca de informações adicionais entre um cliente e um servidor (Kurose, 2013) e podem ser comparadas aos parâmetros de uma chamada de procedimento (Tanenbaum, 2003). Sendo assim, essas informações estão presentes tanto na mensagem de requisição, quanto na resposta e possuem o seguinte formato:

Nome do cabeçalho: Valor do cabeçalho

Nesse sentido, essas informações podem ser utilizadas, por exemplo, para solicitar um documento em um formato e idioma especí co, bem como enviar informações de autenticação. E, os cabeçalhos são categorizados como cabeçalho geral, cabeçalho de solicitação, cabeçalho de resposta e cabeçalho de entidade, conforme é demonstrado na Tabela 2.3.

Cabeçalho	Descrição
Accept	Especi ca os tipos de mídia aceitos na resposta
Authorization	Informa ao servidor as credenciais de acesso ao recurso solicitado
Content-Length	Informa ao cliente o tamanho total, em bytes, do recurso solicitado
Content-Type	Informa ao cliente o tipo de mídia do recurso solicitado
Content-Encoding	Indica que o conteúdo foi codi cado e que mecanismo deve ser aplicado para obter o tipo de mídia indicado no cabeçalho Content-Type
User-Agent	Contém informações sobre o cliente que fez a requisição, podendo limitar a resposta do conteúdo baseado nas limitações do cliente

Tabela 2.3: Cabeçalhos básicos do protocolo HTTP.

2.4 HyperText Markup Language

A HTML é uma linguagem de marcação que descreve através de comandos de formatação também conhecidos como *tags* como os documentos devem ser formatados (Tanenbaum, 2003). Além disso, a HTML é a linguagem de marcação padrão dos navegadores.

Os documentos são estruturados na forma de árvore, onde os elementos HTML são os nós, e essa estrutura é chamada de *Document Object Model* (DOM). O Código 2.1 ilustra o DOM de um documento.

```
1 <html >
2   <head>
3     <title >Documento</title >
4   </ head>
```

```
5    <body></body>
6 </html>
```

Código 2.1: Comandos (tags) básicos para criar um documento HTML.

Ademais, na WWW existem outras linguagens de marcação populares como Extensible Markup Language (XML) e Extensible HyperText Markup Language (XHTML).

A XML é amplamente utilizada para marcar qualquer tipo de dado por possuir um mecanismo de validação de conteúdo, impedindo que tags estejam fora de ordem ou que não tenham sido fechadas adequadamente.

A XHTML, por sua vez, é uma extensão da HTML com as validações do documento provenientes da XML.

2.5 Cascading Style Sheet

Cascading Style Sheet (CSS) é uma linguagem de estilo utilizada para descrever a apresentação de um documento escrito em HTML, XML e XHTML (Network, 2019a). Com CSS é possível alterar algumas propriedades visuais como fontes, cores, imagens de fundo e posicionamento dos elementos na página, isolando assim, a estilização da parte estrutural do documento.

O Código 2.2 representa uma forma de estilizar os elementos HTML descendentes, modi cando o espaçamento e a disposição dos mesmos.

```
1 nav ul {
2     margin: 0;
3     padding: 0;
4     list-style: none;
5 }
6
7 nav ul > li { display: inline-block; }
```

Código 2.2: Estilização de elementos descendentes.

Devido às limitações da linguagem que tornam o processo de desenvolvimento difícil de manter e por vezes repetitivo, surgiram os pré-processadores de CSS para contorná-las.

2.5.1 Pré-processadores

Os pré-processadores CSS são ferramentas que adicionam funcionalidades, as quais por vezes são complexas de implementar na linguagem padrão, com uma estrutura mais legível e simples de manter como mixins, seletores aninhados, condicionais, etc (Network, 2019b).

Entre os pré-processadores, alguns se destacam devido à sua popularidade, sendo eles: Syntactically Awesome Style Sheets (SASS), Leaner Style Sheet (LESS), Stylus e PostCSS.

O Código 2.3 é equivalente ao Código 2.2 e ilustra a funcionalidade de seletores aninhados implementado em boa parte dos pré-processadores CSS.

```
1  nav {
2      ul {
3          margin: 0;
4          padding: 0;
5          list-style: none;
6
7          & > li {
8              display: inline-block;
9          }
10     }
11 }
```

Código 2.3: Seletores aninhados do pré-processador SASS.

É válido ressaltar que tais ferramentas possuem um compilador que produz código CSS, pois atualmente os navegadores somente conseguem interpretá-lo como linguagem de estilização. Atrelado a isso, o desenvolvimento do CSS é mantido de forma constante pelo W3C e, portanto, algumas funcionalidades presentes apenas com a utilização de pré-processadores podem ser incorporadas à linguagem.

2.6 JavaScript

A linguagem de programação foi criada em 1995 na Netscape, na fase inicial da Web, e teve o nome JavaScript licenciado e registrado como marca comercial da Oracle (Flanagan, 2013). Atualmente seu nome o cial é ECMAScript, por conta da associação responsável pelas padronizações e atualizações da linguagem, European Computer Manufacturer's Association (ECMA).

JavaScript foi criado como uma forma de adicionar programas às páginas web, que até então eram estáticas, no navegador Netscape Navigator (Haverbeke, 2018). Desde então a linguagem é amplamente utilizada nos navegadores e se tornou a linguagem de programação da Web.

2.6.1 NodeJS

Como resultado da popularização do ECMAScript, foi criado em 2009 um interpretador que poderia ser utilizado fora dos navegadores, o NodeJS. Tal interpretador é orientado a eventos assíncronos, contribuindo assim para o desenvolvimento de aplicações web escaláveis (Foundation, 2019).

As diferenças entre o NodeJS e o JavaScript que é executado no navegador estão no acesso às Application Programming Interfaces (APIs) do sistema operacional, presentes apenas no NodeJS. Da mesma forma que só é possível acessar informações sobre a janela do navegador através do JavaScript.

O Código 2.4 apresenta as semelhanças e diferenças entre a versão da linguagem que é executada nos navegadores e no servidor, como variáveis e métodos globais particulares ao ambiente de execução e a declaração de funções e variáveis.

```
1 // Apenas no NodeJS
2 process.env // Variaveis de ambiente do servidor
3 fs.readFile(filename, callback) // Leitura de arquivos do
    servidor
```

```
4
5 // Apenas JavaScript (navegador)
6 window.location // Atual URL
7 document.querySelector( '.card ') // Seleciona elemento do DOM
8
9 // Comum
10 console.log(1 + 1)
11 function sum(a, b) {
12   return a + b
13 }
```

Código 2.4: Semelhanças e diferenças entre NodeJS e JavaScript.

2.6.2 TypeScript

Dado que JavaScript é uma linguagem interpretada e de tipagem fraca, é difícil garantir que os parâmetros e retornos de funções possuem o tipo esperado especialmente quando se utiliza bibliotecas de terceiros , di cultando o processo de depuração durante o desenvolvimento. Em virtude disso, criou-se várias convenções de anotações de tipo para JavaScript (Haverbeke, 2018), entre elas se destaca o TypeScript.

TypeScript acrescenta notações de tipos ao JavaScript sem causar grandes mudanças na sintaxe da linguagem, agregando valor através dos benefícios dos tipos. O Código 2.5 exempli ca essa notação de tipos.

```
1 function sum(a: number, b: number): number {
2   return a + b
3 }
```

Código 2.5: Exemplo de notação de tipos com TypeScript.

É importante destacar que essas ferramentas de anotações de tipos funcionam de forma análoga aos pré-processadores CSS, isto é, é gerado código JavaScript após a etapa de

compilação.

2.7 Serviços web

Com os avanços da Web, os servidores passaram a fazer mais do que apenas buscas por documentos e introduziu-se o processamento de dados nos documentos antes de serem retornados para os clientes (Tanenbaum, 2007). Nessa perspectiva, a expansão desse modelo para aplicações remotas sem interações imediatas de usuários nais leva ao conceito de serviços web (Alonso, 2004).

De acordo com W3C (2004) um serviço web é um sistema projetado para suportar interações interoperáveis entre máquinas através da Internet. Tal interoperabilidade permite que esses serviços sejam escritos em diferentes linguagens de programação. Desse modo é possível que haja a colaboração com serviços legados, especialmente em instituições com recursos humanos ou nanceiros limitados, onde a reescrita de tais serviços é inviável (Kalin, 2013).

Atrelado a isso, os serviços web podem ser divididos em aproximadamente dois grupos: os que são baseados no Simple Object Access Protocol (SOAP) e os que seguem os padrões Representational State Transfer (REST).

2.7.1 Simple Object Access Protocol

Segundo Tanenbaum (2003) SOAP é um modo de executar Remote Procedure Calls (RPCs) entre aplicações de forma independente da linguagem de programação e do sistema. A comunicação é feita através da troca de documentos XML com o protocolo HTTP, sendo que o formato desses documentos é de nido em uma Web Services De nition Language (WSDL), a qual descreve os procedimentos, os tipos de dados e assim por diante.

Os documentos XML transitados nas mensagens SOAP possuem um elemento Envelope que identi ca o documento como uma mensagem SOAP e um elemento Body que

contém as informações para a chamada e a resposta do procedimento, conforme é apresentado no Código 2.6.

```
1  <?xml version ="1.0"?>
2  <soap:Envelope
3      xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
4      soap:encodingStyle="http://www.w3.org/2003/05/soap-
         encoding">
5      <soap:Body xmlns:m="http://www.example.org/stock">
6        <m:GetStockPrice>
7          <m:StockName>IBM</m:StockName>
8        </m:GetStockPrice>
9      </soap:Body>
10 </soap:Envelope>
```

Código 2.6: Exemplo de requisição do protocolo SOAP, invocando o procedimento GetStockPrice passando o parâmetro StockName

Em contrapartida, os mesmos mecanismos que garantem a integridade da comunicação nesse protocolo também o tornam complexo e, como efeito, essa tecnologia atualmente é pouco utilizada. Em 2017, a plataforma ProgrammableWeb[1], que possui uma vasta lista de APIs públicas, publicou uma análise de seus dados, onde indica que de 18279 APIs apenas 9.36% utilizam SOAP (Santos, 2017).

2.7.2 Representational State Transfer

Estilo arquitetural para aplicações de hipermídia distribuídas proposto por Roy Thomas Fielding em sua dissertação de doutorado no ano de 2000. A abstração chave da proposta é o recurso, que pode ser qualquer elemento no sistema que possa ser nomeado identi cado por um Uniform Resource Identi er (URI) (Fielding, 2000).

Sendo assim, o estilo em questão possui um conjunto de restrições sendo elas:

[1] https://www.programmableweb.com

- Interface Uniforme: Principal restrição que distingue essa arquitetura das demais. Nela é sugerido que deve existir uma forma uniforme de interagir com o servidor independente do dispositivo e do tipo da aplicação, geralmente utilizando o protocolo HTTP.

- Stateless : A comunicação deve ser stateless, isto é, uma vez que as requisições entre cliente e servidor devem conter todas as informações necessárias para que sejam entendidas, não deve-se armazenar contextos no servidor. Essa restrição otimiza propriedades como visibilidade, confiabilidade e escalabilidade.

- Cache: Melhora a eficiência da rede ao armazenar os dados da resposta, permitindo o reuso da mesma, caso a requisição tenha sido marcada como cacheável, bem como por quanto tempo essa informação é válida.

- Cliente-Servidor: A separação de responsabilidades entre a interface do usuário e o armazenamento de dados melhora a portabilidade da interface do usuário entre múltiplas plataformas e melhora também a escalabilidade ao simplificar os componentes do servidor.

- Sistemas de Camadas: Uma aplicação pode ser composta de múltiplas camadas, onde cada camada não sabe da existência das demais. Também melhoram a disponibilidade do sistema com mecanismos de balanceamento de carga, cache e compartilhado.

- Código sob Demanda: Restrição opcional que permite a extensão por meio da obtenção e execução de códigos na forma de applets ou scripts.

Atrelado a isso, serviços web que seguem todas estas restrições são chamados de RESTful.

O uso do HTTP como protocolo de comunicação fornece interfaces bem definidas e bem conhecidas que, por sua vez, trouxeram simplicidade para a arquitetura. Devido à tal simplicidade, no mesmo estudo mencionado na Subseção 2.7.1, REST aparece como o modelo mais utilizado para se construir serviços web.

2.8 Single Page Application

Nas aplicações web tradicionais, também conhecidas como Multiple Page Applications (MPAs), sempre que uma página é solicitada, é feita uma requisição para o servidor solicitando a criação e/ou retorno do arquivo HTML em questão. Ao receber a resposta, o navegador recarrega a página, conforme ilustrado na Figura 2.3.

Figura 2.3: Fluxo de aplicações web tradicionais. Fonte: Wasson (2013).

Em contrapartida, com essa abordagem as aplicações perdem a fluidez da navegação e, consequentemente, comprometem a experiência do usuário devido aos sucessivos recarregamentos de páginas entre as requisições. Tendo consciência dessas implicações, introduziu-se o conceito de requisições Asynchronous JavaScript And XML(AJAX), permitindo que as páginas pudessem solicitar ou enviar dados sem que houvesse o recarregamento das mesmas, havendo apenas nas transições de páginas.

A Figura 2.4 descreve o fluxo de aplicações web que utilizam requisições AJAX para alterar seu conteúdo.

Figura 2.4: Fluxo de aplicações web com o uso de requisições AJAX. Fonte: Wasson (2013).

Atrelado a isso, visando preencher as de ciências de aplicações MPAs que utilizam AJAX, emergiu-se o conceito de *Single Page Application* (SPA). Como o nome sugere, as aplicações desse modelo possuem apenas uma página HTML. As demais páginas são exibidas através de manipulação do DOM e a comunicação com o servidor para a transferência de dados - exclui-se os arquivos de marcação - se dá através de requisições AJAX, tornando o servidor uma camada de serviço.

Apesar das SPAs proporcionarem uma melhor experiência do usuário, seu uso é relativo ao objetivo a ser alcançado através da aplicação. Por exemplo, por dependerem da execução de *scripts*, geralmente através de JavaScript, os motores de busca não conseguem indexar as páginas da aplicação, di cultando a identi cação das mesmas, essa é uma grande desvantagem para aplicações como *e-commerces*, *blogs* e *websites* de notícias.

No ecossistema de *frameworks* e bibliotecas que auxiliam no desenvolvimento de aplicações SPAs se destacam: Angular, React e Vue. As discrepâncias entre essas ferramentas se dão no *design* da solução e no algoritmo responsável pela detecção de mudanças no DOM. Dentre tais algoritmos estão o *Virtual DOM* (VDOM) - utilizado no React e Vue - e o *Change Detector* (CD) - utilizado no Angular.

Segundo Simon (2019) a manipulação nos elementos do DOM é custosa e, portanto,

se faz necessária a criação de mecanismos para diminuí-la.

2.8.1 Virtual DOM

O VDOM é uma representação em memória do real DOM. Minimiza a manipulação dos elementos do DOM através de comparações entre as árvores virtual e real, alterando no DOM apenas os nós que sofreram mudanças, conforme ilustrado na Figura 2.5.

Figura 2.5: Mecanismo de detecção de mudanças utilizado no VDOM. Fonte: Frachet (2018).

Vale ressaltar que apesar de suavizar a manipulação na árvore de elementos, alterações mal planejadas podem ocasionar na reconstrução de toda a sub-árvore de elementos. Portanto, é fundamental que as mudanças sejam feitas cuidadosamente por parte do utilizador da tecnologia, para que não deteriore a performance da aplicação.

2.8.2 Change Detector

O mecanismo utilizado no framework Angular funciona através da detecção de mudanças entre as variáveis definidas no componente e a ligação das mesmas no template. Essa

ligação entre componente e template é demonstrada no Código 2.7.

```
1 // exemplo.component.ts
2 class ExemploComponent {
3   nome = 'John Doe'
4 }
5
6 // exemplo.component.html (template)
7 <p>{{nome}}</p>
```

Código 2.7: Declaração de variáveis no componente e da utilização das mesmas no template.

De acordo com Arora (2018), cada componente possui o seu próprio CD (ver Figura 2.6), cuja responsabilidade é monitorar propriedades do componente utilizadas no template.

Figura 2.6: Árvore de componentes e seus respectivos CDs. Fonte: Arora (2018).

Nesse sentido, sempre que houverem eventos que alterem as variáveis dos componentes como eventos do navegador, interação do usuário, requisições para aplicações externas ou timers (setInterval e setTimeout), haverá uma verificação de todos os CDs da sub-árvore desse componente, conforme ilustrado na Figura 2.7. É importante destacar que assim como

no Virtual DOM , o mecanismo em questão está sujeito à deteriorações de performance se mal utilizado.

Figura 2.7: Árvore de componentes durante a detecção de um evento e execução do CDs. Fonte: Arora (2018).

2.9 Business Process Model and Notation

Segundo Pavani (2011), a modelagem dos processos é a ferramenta básica e primordial para a execução de gestão por processos. Adjacente a isso, o objetivo da modelagem é representar gra camente um processo de forma clara e objetiva às partes interessadas. Para isso existem os modelos de notação, dentre eles o Business Process Model And Notation (BPMN).

O BPMN é uma linguagem de diagramação e um padrão para uxo de processos de negócio criado e mantido pela Object Management Group (OMG). A criação desse padrão é crucial para a compreensão dos diagramas por todas as pessoas envolvidas no processo, uma vez que há uma integração entre os mundos de negócios e tecnologia da informação (Silver, 2011).

2.9.1 Elementos BPMN

Para que a representação dos processos seja feita de forma clara se faz necessária a utilização de elementos grá cos semânticos, que podem ser categorizados como objetos de uxo, dados, objetos de conexão, swimlanes e artefatos.

A seguir serão brevemente descritos os elementos de acordo com a especi cação do BPMN 2.0 (Group, 2011).

2.9.1.1 Objetos de uxo

São os principais elementos que de nem o comportamento de um processo de negócio e são compostos por eventos, atividades e gateways.

Eventos: Um evento é algo que acontece durante o curso de um processo, podendo indicar o início ou o m de um uxo. Os eventos são representados por círculos com o centro vazio, permitindo a adição de elementos que diferenciem os eventos. Há três tipos de eventos: de início, intermediário e de nalização. A Figura 2.8 exempli ca alguns eventos do BPMN.

Figura 2.8: Exemplos de eventos do BPMN, na sequência por linha: evento de início, intermediário, de nalização, de início a partir de uma mensagem, de início a partir de um temporizador, e de nalização com o envio de uma mensagem.

Atividades: Uma atividade é um termo genérico para um trabalho realizado em um processo. As atividades são divididas em subprocesso e tarefa e são representados por retângulos arredondados.

Gateways: Um gateway é usado para controlar o uxo de sequência de um processo, podendo divergir ou convergir os caminhos. Os gateways são representados com losango e, assim como os eventos, possui o centro dedicado para indicar os tipos de comportamentos, conforme ilustrado na Figura 2.9

Figura 2.9: Exemplos de gateways do BPMN, na sequência por linha: gateway exclusivo, paralelo, inclusivo, complexo, e baseado em eventos.

2.9.1.2 Objetos de conexão

Há algumas formas de conectar objetos de uxo entre si ou outras informações, e essas estão apresentadas a seguir e representadas na Figura 2.10.

Fluxos de sequência: Um uxo de sequência é usado para mostrar a ordem em que as atividades serão executadas no processo.

Fluxos de mensagens: Um uxo de mensagem é usado para mostrar o uxo de mensagens entre dois participantes, indicando a colaboração entre eles.

Associações: Uma associação é usada para ligar informações e artefatos a elementos BPMN, enriquecendo assim a representação dos diagramas.

Figura 2.10: Objetos de conexão do BPMN, na sequência: uxo de sequência, de mensagens e associações.

2.9.1.3 Dados

Os elementos associados a dados complementam o entendimento do uxo de informações de um processo. Entre eles estão os elementos de entrada/saída, bem como repositórios

para outras informações, conforme expressado na Figura 2.11.

Figura 2.11: Elementos de dados do BPMN, na sequência: objeto de dado, objeto de entrada de dado, objeto de saída de dado e repositório de informações.

2.9.1.4 Swimlanes

É possível fazer o agrupamento dos elementos primários da modelagem através das *swimlanes*

Pool: Uma *pool* é uma representação de um participante na colaboração do processo, pode conter mais informações sobre a participação ou também pode ser utilizado como uma caixa preta apenas para indicar o envolvimento.

Lane: Uma *lane* é uma subpartição dentro de um processo, às vezes dentro de uma *pool*, e são utilizadas para organizar e categorizar atividades.

A Figura 2.12 demonstra o uso dos elementos *pool* e *lane*.

Figura 2.12: *Pool* com duas *lanes*

2.9.1.5 Artefatos

Os artefatos são usados para fornecer informações adicionais sobre o processo e há apenas dois artefatos padronizados pelo BPMN, apresentados a seguir.

Grupo: Um grupo é um agrupamento de elementos grá cos que estão dentro da mesma categoria para ns de documentação ou análise. A Figura 2.13 ilustra esse artefato.

Figura 2.13: Exemplo de agrupamento de elementos BPMN com a nalidade de documentação.

Anotações de texto: As anotações são um mecanismo para o modelador facilitar a percepção de informações para o leitor do diagrama. A Figura 2.14 ilustra esse artefato.

Figura 2.14: Exemplo de anotações de texto a elementos BPMN.

2.10 Tecnologias

O desenvolvimento de sistemas web pode requerer trabalho repetitivo ou de complexidade considerável. Para limitar o escopo do projeto, foram utilizadas diferentes bibliotecas e

ferramentas, que são descritas nessa seção. Quanto à linguagem de programação, essencialmente utilizou-se JavaScript nas aplicações do cliente e servidor (NodeJS), devido à sua simplicidade e performance.

2.10.1 ExpressJS

Na visão de Brown (2014), ExpressJS é um *framework* web inspirado no outro *framework* da linguagem Ruby, o Sinatra. Ambos têm como foco tornar o desenvolvimento de aplicações web mais rápido, mais e ciente e mais manutenível.

Com o uso dessa ferramenta é possível criar e con gurar um servidor HTTP com o menor número de funcionalidades necessárias para o funcionamento de um servidor de tal natureza e, se necessário, é possível estendê-las através de *plugins*.

O Código 2.8 ilustra a simplicidade de se trabalhar com a ferramenta.

```
1 const app = require( 'express')()
2 app.get( '/', (req, res) => res.send(  'Hello World! '))
3 app.listen(8080)
```

Código 2.8: Exemplo mínimo de um servidor feito com ExpressJS.

2.10.2 SequelizeJS

A utilização de ferramentas de armazenamento, como banco de dados, é de suma importância para o desenvolvimento de aplicações web atualmente. Para que seja feita a comunicação entre as aplicações e tais ferramentas, existem diversas bibliotecas para os diversos bancos de dados existentes. No entanto, se faz necessária a criação de outra camada de abstração que reduza as particularidades de cada plataforma.

O SequelizeJS é uma biblioteca que provê essa e outras abstrações através da técnica de mapeamento de objetos relacionais (do inglês, *Object Relational Mapping* (ORM)). Tal técnica permite que pensemos na estrutura do banco de dados como objetos ao invés da

abstração de linhas em uma tabela, promovendo uma melhor experiência na manipulação do banco de dados em aplicações com grande número de tabelas e relacionamentos.

2.10.3 Angular

Angular é um framework para a criação de SPAs criado pelo Google em 2009. Através de padrões de desenvolvimento como injeção de dependência, modularização de componentes visuais, lazy loading de módulos e componentes, o mecanismo de detecção de mudanças apresentado na Seção 2.8.2, checagem de tipos através da utilização da linguagem TypeScript, entre outros, permite a manutenibilidade em projetos de qualquer dimensão.

Enquanto que funcionalidades com data binding e interface de comandos (CLI) para geração de componentes Angular (componentes, serviços e módulos), permitem a otimização no processo de desenvolvimento.

2.10.4 Git

Segundo Loeliger (2012), Git é um sistema de versionamento poderoso, exível e leve, que torna mais efetivo o desenvolvimento colaborativo.

De acordo com Chacon (2014) a ferramenta foi criada em 2005 por Linus Torvald após um con ito com a empresa BitKeeper, a qual era proprietária do sistema de controle de versão utilizado no projeto Linux. O novo sistema tinha como objetivo ser rápido, simples, seguro, totalmente distribuído e capaz de lidar com projetos grandes como o do kernel Linux.

A ferramenta é utilizada atualmente, segundo o site o cial do projeto[2], pelos seguintes projetos: Google, Facebook, Microsoft, Twitter, LinkedIn, Net ix, Android, Linux Kernel, entre vários outros. Também foi utilizada na implementação prática deste trabalho.

Ademais, é possível hospedar repositórios Git em plataformas como GitHub, GitLab, BitBucket, etc, que facilitam a interação e colaboração em projetos através dos quadros de atividades é possível de nir as tarefas, bem como suas prioridades, e assim monitorar o

[2] https://git-scm.com/

progresso das mesmas através de grá cos burndown. Vale frizar que a plataforma utilizada nesse trabalho foi o GitLab.

2.10.5 Docker

No entendimento de Nickolo (2019), Docker é uma ferramenta que resolve problemas comuns como instalar, remover, atualizar, distribuir, con ar e gerenciar software. O mecanismo consegue sanar tais problemas devido a utilização do recurso de contentores presente na maioria dos sistemas operacionais.

Contêiner é um mecanismo que permite o encapsulamento de aplicações com suas dependências, tornando-as portáveis e auto-contidas (Mouat, 2015). Apesar da similaridade com máquinas virtuais, os contentores se distinguem por não utilizarem virtualização de hardware, mas se comunicam diretamente com o kernel do hospedeiro. Dessa forma são consumidos menos recursos, permitindo a utilização de máquinas menos potentes e inicialização rápida das aplicações.

Face ao exposto, a ferramenta permite a criação de ambientes colaborativos que simulem o ambiente onde a aplicação será executada, evitando problemas de compatibilidade entre as diversas con gurações de máquinas utilizadas nos times de desenvolvimento.

Capítulo 3

Análise de Requisitos e Modelação

Este capítulo é focado em descrever os requisitos que foram identi cados e analisados para a concepção dos novos módulos da aplicação. Estes requisitos foram de nidos com base na aplicação de técnicas de etnogra a, visando obter uma descrição qualitativa do comportamento e necessidade dos utilizadores do sistema no seu ambiente natural.

Em linhas gerais, as etapas seguidas estão descritas no Diagrama 3.1.

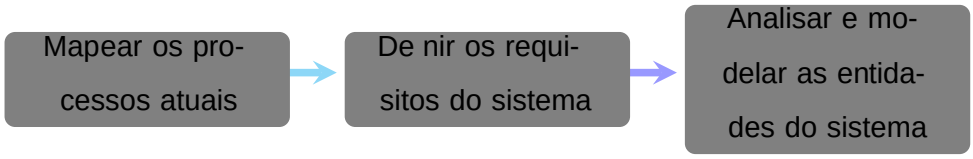

Diagrama 3.1: Etapas para levantamento de requisitos.

3.1 Âmbito do projeto

A empresa Riskivector possui diversas de suas atividades informatizadas em uma aplicação web, a qual permite a agilidade nos processos e gerência sobre recursos como:

ˆ Apartamentos

Informações sobre contas;

Informações sobre registros de água, gás, eletricidade e Internet;

Informações sobre leituras mensais;

Informações sobre quarto(s).

- Quartos;
- Inquilinos

Informações pessoais;

Informações sobre reserva(s);

Informações sobre pedido(s) de mudança;

Informações sobre quarto(s).

Nesse contexto, o desenvolvimento de novas funcionalidades que permitam a redução de tempo e tarefas repetitivas é de interesse para a expansão do negócio. Sendo assim, o projeto tem como foco:

- Melhorar a precisão do algoritmo de cálculo de contas;
  A melhoria está relacionada a uma solução para facilitar os testes e comprovar a e ciência dos valores computados.
- Registro de movimentações no caixa da empresa;
- Emissão de contas para clientes;
- Pagamento de contas de clientes através de APIs de bancos parceiros;
- Emissão de faturas de pagamentos de clientes através de sistema de faturação parceiro.

3.2 Mapeamento de processos de negócio

Nesta etapa, realizou-se o levantamento de informações relacionadas aos processos existentes na empresa para compreender a situação atual ou, como tratada por alguns autores,

a situação as is. Sendo assim, criou-se os diagramas BPMN utilizando a ferramenta online BPMN Viewer and Editor[1], apresentados a seguir, que ilustram os processos de negócio referentes ao escopo do projeto.

A Figura 3.1 representa o processo de negócio de transferência de dinheiro entre funcionários que pode ter diversas nalidades, entre elas estão: abastecimentos e reparos de automóveis; compra de materiais em geral e antecipação de salários dos funcionários.

É importante ressaltar que o funcionário responsável pelo levantamento do dinheiro também é responsável por trazer o recibo que comprove a necessidade de tal levantamento, caso contrário desconta-se o valor do salário do mesmo.

[1] https://bpmn.io

Figura 3.1: Processo de negócio de transferência de dinheiro entre funcionários.

A Figura 3.3 representa o processo de negócio de registro de movimentações no uxo de caixa da empresa feito pelo gerente, sendo que o uxo de caixa principal é registrado numa folha de cálculo no Google Sheet. Além dos tipos de despesas mencionados para

transferência de dinheiro, há também o registros dos seguintes tipos:

- Pagamentos via Point of Sale (POS)
- Pagamentos Online
- Transferências bancárias
- Pagamentos de faturas dos apartamentos
- Pagamentos do aluguer aos senhorios
- Recibos trazidos pelos funcionários

Para que haja uma organização nos registros, há padrões de descrições para cada tipo de movimentação, como:

- Recebimento do <nome do inquilino> referente a tabela de contas de Maio (debito de 1268.65€) (<alcunha do apartamento>)
- Recebimento ONLINE (158.30€, on 06-MAY by <nome da pessoa que enviou o pagamento>) da <nome do inquilino> referente a tabela de contas de Maio (<alcunha do apartamento>)
- Depositos no Riskivector Montepio (<nome do funcionário>) (18-DEC)
- P.A.CEPSA BRAGANÇA, Gasoleo (<informações do veículo>, Readings: <referência do recibo>) (<nome do funcionário>) (23-JUL).

Entretanto, apesar da utilização de planilhas para a gestão das movimentações, há uma falta de controle dos registros provenientes da organização dos mesmos. Por exemplo, para registros de movimentação interna de dinheiro, cujo real propósito da quantia levantada só é conhecido no momento que o funcionário entregar o recibo, alguns valores são acumulados em uma única célula (vide Figura 3.2) até o momento da con rmação da despesa, di cultando o processo organizacional.

Figura 3.2: Exemplo de registro e célula acumuladora de valores presentes no uxo de caixa da companhia.

Figura 3.3: Processo de negócio de registro de movimentações feito pelo gerente.

A Figura 3.4 exibe o processo de negócio de emissão de contas dos inquilinos, o qual consome tempo devido às exceções encontradas tais como a troca de apartamento no meio do mês, presença de convidados, saída no meio do mês e até mesmo a combinação dos casos mencionados.

Figura 3.4: Processo de negócio de emissão de contas.

Na Figura 3.5 é demonstrado o processo de negócio de pagamento de contas dos inquilinos, processo este responsável pela renda da companhia.

Figura 3.5: Processo de negócio de pagamento de contas dos inquilinos.

Na Figura 3.6 é representado o processo de negócio de criação e envio de faturas para os inquilinos. As faturas são geridas através do KeyInvoice[2], um software de gestão e faturação online.

Figura 3.6: Processo de negócio de registro e envio de faturas.

3.3 Requisitos

Os Requisitos Funcionais (RF) e Não Funcionais (RNF) do sistema são apresentados nesta seção, descrevendo os serviços e funcionalidades que o software deverá fornecer (ações), além das características que o mesmo deverá apresentar (restrições).

3.3.1 Requisitos Funcionais

RF-1. Gerenciar movimentações :

Para que seja possível rastrear as movimentações de dinheiro feitas dentro da empresa é necessário que os atores envolvidos façam parte da manutenção destas operações.

[2] https://www.keyinvoice.com

RF-1.1. Exibir movimentações :

O software deverá exibir as movimentações condicionadas à autorização do usuário que as solicita.

RF-1.2. Detalhes de movimentação :

O software deverá exibir os dados de uma movimentação selecionada.

RF-1.3. Criação de movimentações :

O software deverá permitir criar novas movimentações.

RF-1.4. Atualizar movimentações :

O software deverá permitir atualizar uma movimentação selecionada.

RF-2. Gerenciar contratos :

Para que se tenha controle dos acordos estabelecidos com os funcionário, é imprescindível que o sistema permita a gerência de contratos dos funcionários.

RF-2.1. Exibir contratos :

O software deverá exibir os contratos de um funcionário.

RF-2.2. Detalhes do contrato :

O software deverá exibir os dados de um contrato selecionado.

RF-2.3. Criação de contratos :

O software deverá permitir criar novos contratos.

RF-2.4. Atualizar contratos :

O software deverá permitir atualizar um contrato selecionado.

RF-2.5. Remover contratos :

O software deverá permitir a remoção de um contrato selecionado.

RF-3. Gerenciar contas :

Para que seja possível rastrear a quantia de dinheiro da empresa ou quantia em débito que cada funcionário possui, é fundamental que o software permita a gerência de contas dos funcionário.

RF-3.1. Exibir contas :

O software deverá exibir as contas de um funcionário.

RF-3.2. Detalhes de conta :

O software deverá exibir os dados de uma conta selecionada.

RF-3.3. Criação de contas :

O software deverá permitir criar novas contas.

RF-3.4. Atualizar contas :

O software deverá permitir atualizar uma conta selecionada.

RF-3.5. Remover contas :

O software deverá permitir a remoção de uma conta selecionada.

RF-4. Emitir despesas para inquilino :

O sistema deve permitir que os funcionários emitam despesas para o(s) inquilino(s), sendo esse o mecanismo que gera parte da receita da companhia.

RF-5. Gerenciar movimentações de conta dos inquilinos :

O software deve fornecer aos usuários meios de gerir as movimentações de conta dos inquilinos.

RF-5.1. Exibir movimentações :

O software deverá exibir as movimentações de um inquilino.

RF-5.2. Detalhes de movimentação :

O software deverá exibir os dados de uma movimentação selecionada.

RF-5.3. Criação de movimentações :

O software deverá permitir criar novas movimentações.

RF-5.4. Atualizar movimentações :

O software deverá permitir atualizar uma movimentação selecionada.

RF-6. Consultar saldo de inquilinos :

O software deve ser capaz de exibir o saldo de um inquilino selecionado.

RF-7. Gerenciar dados (Bills Calculation Tests (BCTs)) para otimizar algoritmo de emissão automática de contas:

O software deve fornecer aos usuários meios de gerir dados que permitirão a otimização do algoritmo de emissão de contas.

RF-7.1. Exibir BCTs:

O software deverá exibir os BCTs de um apartamento.

RF-7.2. Detalhes de BCTs:

O software deverá exibir os dados de um BCT selecionado.

RF-7.3. Criação de BCTs:

O software deverá permitir criar novos BCTs.

RF-7.4. Remover BCTs:

O software deverá permitir a remoção de um BCT selecionado.

RF-8. Emitir fatura-recibo:

O software deve ser capaz de emitir fatura-recibo através da plataforma de faturação KeyInvoice.

RF-9. Prover interface de pagamento:

O software deve ser capaz de fornecer um endpoint para o pagamento através de serviços de bancos parceiros.

3.3.2 Requisitos Não Funcionais

Os Requisitos Não Funcionais do sistema estão relacionados a seguir:

RNF-1. Linguagem de Programação

O software deverá ser escrito em NodeJS.

RNF-2. Banco de Dados

O software deverá utilizar banco de dados relacional.

RNF-3. Modelagem

O software deverá ser projetado utilizando diagramas Unified Modeling Language (UML) e BPMN.

RNF-4. Plataforma

O software deverá disponibilizar todos os serviços previstos através da plataforma web.

RNF-5. Interface Gráfica

O software deverá apresentar uma interface gráfica semelhante às páginas presentes no projeto.

RNF-6. Idioma da Interface Gráfica

O software deverá apresentar uma interface gráfica no idioma inglês.

3.3.3 Diagrama de casos de uso

Esta seção apresenta os usos e aplicações do sistema por meio de diagramas de casos de uso. A construção destes diagramas corresponde a uma das fases iniciais de um projeto de software, atuando como instrumento e ciente para determinação e documentação dos serviços a serem desempenhados.

Um diagrama de casos de uso utiliza como primitivas: atores, casos de uso e associações. Atores, de acordo com Pereira (2011), são alguém ou alguma coisa que interagem com o sistema sendo modelado. Basicamente, a interação dos atores com o sistema ocorre por meio de troca de mensagens. Os atores do sistema são:

- Funcionário

  Representa um funcionário da empresa autorizado a utilizar o software por meio de autenticação, para gerenciar suas movimentações e dos inquilinos, bem como emitir contas.

- Gerente

  Representa um funcionário da empresa autorizado a utilizar o software por meio

de autenticação, para realização de tarefas administrativas como gerenciar todas as movimentações da companhia, gerenciar contratos e contas.

- Aplicação de clientes

  Representa uma outra aplicação da companhia responsável por prover interação com o inquilino.

Figura 3.7: Diagrama de casos de uso.

3.3.4 Diagrama de classes

As classes são elementos fundamentais na composição de softwares orientados a objetos, pois elas descrevem a estrutura do sistema a ser projetado. Dessa forma, após a etapa de levantamento de requisitos e modelação dos casos de uso, compôs-se o diagrama de classes para os novos módulos da aplicação.

O diagrama de classes ilustrado na Figura 3.8 apresenta as associações entre as classes modelo.

Figura 3.8: Diagrama de classes dos novos módulos.

Para o pacote de funcionalidades denominado Fluxo de caixa , modelou-se algumas

entidades e, em especial, a entidade central nomeada Transaction. Esse domínio armazena informações cruciais para uma movimentação nanceira, conforme ilustrado na Figura 3.9, bem como metainformações como tags, recibos e denominações em entidades relacionais.

Figura 3.9: Pacote Fluxo de caixa com ênfase na entidade Transaction.

É importante salientar a semântica dos atributos sender e bene ciary, os quais fazem referência a contas de funcionários. Sendo assim, quando o atributo sender tiver valor nulo signi ca que o dinheiro recebido é de uma fonte externa, por exemplo pagamentos de inquilinos. Enquanto que para a propriedade bene ciary isso representa uma despesa da companhia.

Com o intuito de tornar as movimentações mais semânticas e complementar a relação com outras entidades, torna-se necessária a criação de entidades de extensões, pertencentes ao pacote Pontos de extensão apresentado na Figura 3.8. A modelagem desempenhada nesse trabalho apresenta duas entidades de extensão, EmployeeRemuneration e TenantPayment, referentes, respectivamente, aos pagamentos dos funcionários e de inquilinos. Assim sendo, essa abordagem permite que novas entidades sejam facilmente agregadas.

3.4 Considerações nais

Neste capítulo foram apresentados alguns elementos essenciais para as fases iniciais de um projeto de software, como análise de requisitos, diagramas de casos de uso e de classes. Enfatizou-se a apresentação dos processos de negócio as is , os quais tiveram papel fundamental na modelação de funcionalidades que se adaptam às necessidades dos usuários envolvidos. Também foram apresentados os componentes não triviais da modelagem do diagrama de classes.

Nos capítulos seguintes será apresentada a metodologia empregada para realizar o desenvolvimento, bem como os resultados que foram obtidos.

Capítulo 4

Desenvolvimento e Resultados

Neste capítulo é descrito o trabalho de implementação e os resultados obtidos no desenvolvimento dos novos módulos do sistema da Riskivector, salientando os pontos mais relevantes dessa etapa, assim como di culdades e soluções encontradas.

4.1 Trabalho Colaborativo e Integração Contínua ao Projeto Existente

Uma vez que o presente trabalho faz parte de um projeto existente na companhia, fez-se necessário o trabalho em colaboração com os demais desenvolvedores envolvidos no mesmo projeto ou em projetos que dependam de funcionalidades daquele em andamento. Dessa maneira, foram utilizadas ferramentas que dessem suporte a essa necessidade como Git, GitLab e Docker.

4.1.1 Git

A ferramenta de controle de versão teve papel fundamental no desenvolvimento deste trabalho, pois permitiu a constância e organização do uxo de trabalho. Tais objetivos foram alcançados devido à funcionalidade de rami cação (branch) e uso do modelo de trabalho Git Flow criado por Driessen (2010).

O modelo seguido se baseia na utilização de branches que separem o código que está sendo utilizado em produção geralmente utilizado branch master das funcionalidades em desenvolvimento, conforme ilustrado na Figura 4.1.

Figura 4.1: Modelo de trabalho Git Flow. Fonte: Driessen (2010).

4.1.2 GitLab

Para a gestão de atividades e integração contínua (CI) utilizou-se a ferramenta GitLab. Através dos quadros de atividades (ver Figura 4.2) foi possível definir as tarefas, bem como suas prioridades, e assim monitorar o progresso das mesmas através de gráficos burndown

Figura 4.2: Quadros de atividades do GitLab.

Atrelado a isso, também foram utilizadas as *pipelines* da plataforma, que permitiu a automação das atividades de implantação das novas versões do sistema.

4.1.3 Docker

Visando replicar o ambiente de produção durante o processo de desenvolvimento, foi utilizada a ferramenta de contentorização Docker. Para isso criou-se imagens com o ambiente configurado utilizando as mesmas versões dos pacotes utilizados no serviço de hospedagem da aplicação.

4.2 Restruturação da Arquitetura do Serviço Web

A aplicação existente não seguia nenhuma separação de responsabilidade em muitas de suas funções, gerando, por consequência, um alto acoplamento para com as bibliotecas de terceiros. Sendo assim, se fez necessária a restruturação da arquitetura da aplicação para a criação dos novos módulos e para isso foi escolhida a arquitetura de camadas.

4.2.1 Arquitetura de camadas

Na visão de Richards (2015) a arquitetura de camadas é a mais difundida entre arquitetos, designers e desenvolvedores por ter sido utilizada em muitas aplicações Java. Nessa

arquitetura cada camada desempenha um papel especí co dentro da aplicação, além disso não há uma padronização quanto à quantidade de camadas.

Nesse trabalho foram utilizadas três camadas, sendo elas: apresentação, regras de negócio + persistência e banco de dados (ver Figura 4.3). A segunda camada é uma combinação de duas camadas devido ao acoplamento da biblioteca de persistência de dados.

Figura 4.3: Arquitetura de camadas escolhida para o projeto. Adaptado de Richards (2015).

4.2.1.1 Camada de apresentação

Camada responsável por lidar com toda a interface de usuário e lógica de comunicação do navegador, seja servindo arquivos de marcação, seja fornecendo apenas informações.

4.2.1.2 Camada de regras de negócios e persistência

Camada responsável pela execução de regras de negócio especí cas e associadas com a requisição, e por manipular o dado obtido pelos componentes de persistência.

4.2.1.3 Camada de banco de dados

Camada responsável por efetuar operações diretamente no banco de dados, geralmente essas ações são delegadas por componentes de persistência.

4.3 Limitações de Bibliotecas de Terceiros

4.3.1 SequelizeJS

Apesar de ser o ORM mais popular do ecossistema NodeJS, a biblioteca apresenta alguns problemas de consistência de sua API e apresentação da documentação, bem como algumas limitações no que diz respeito à comunicação com bancos de dados.

Tais problemas comprometeram parte do desenvolvimento por requerer muito tempo na investigação dos mesmos. Uma das limitações diz respeito ao uso de triggers, o qual não possui suporte e foi necessário recorrer a scripts na linguagem Bash. Em contrapartida a solução não é a ideal, por criar dependência de recursos fora do controle da principal linguagem de programação do sistema.

4.4 Módulo de gestão financeira

Baseado nos requisitos coletados apresentados no Capítulo 3 produziu-se um módulo que, permite a informatização e melhor integração dos dados de elementos de gestão financeira como o fluxo de caixa.

Atendendo aos requisitos, o sistema é capaz de gerenciar as movimentações financeiras (tanto da empresa quanto dos inquilinos) e contratos dos funcionários. As funcionalidades são restritas por controle de acesso e/ou funcionam de forma diferente dependendo do perfil autenticado.

As problemáticas acerca desse módulo estão atreladas à limitação de conteúdo baseado no tipo de permissão que o usuário possui, o número de tabelas envolvidas em uma movimentação ou transação, apresentadas no 3.3.4 e decisões sobre otimização de consultas.

As interfaces da aplicação foram desenvolvidas utilizando o design system Material criado pelo Google, visando manter a consistência com os outros módulos da aplicação.

4.4.1 Interfaces para gerência de movimentações e emissão de contas dos inquilinos

Essa seção apresenta as interfaces disponíveis para a gerência de movimentações dos inquilinos na plataforma web da companhia, que abrangem a visualização das movimentações, a emissão de contas e o pagamento das mesmas.

A Figura 4.4 exibe a interface principal do sistema, na qual os funcionários podem visualizar as movimentações de um inquilino selecionado e o saldo do mesmo. As movimentações vigentes são exibidas numa tabela, onde as colunas são respectivamente:

1. Date : A data em que a movimentação (contas e pagamentos) foi registrada.
2. Amount : Montante da movimentação.
3. Running Balance : Informação adicional que representa o saldo disponível após a atual movimentação.
4. Type : Tipo de movimentação pode ser de pelo menos três tipos: Tenant Payment, Tenant Bill e Tenant Miscellaneous Bill, sendo que o terceiro pode possuir subtipos.
5. Operator : Funcionário responsável pelo registro da movimentação.
6. Description : Descrição da movimentação, vale salientar que essa informação somente é disponibilizada para os funcionários. Nessa coluna são colocados os valores padronizados apresentados na Seção 3.2.

Figura 4.4: Interface com movimentações (contas e pagamentos) de um inquilino.

Para construir essa visualização é necessário unir dados heterogêneos de três entidades, sendo elas:tenant_bill, tenant_miscellaneous_bill e tenant_payment. A relação entre essas entidades é expressa na Figura 4.5.

Figura 4.5: Diagrama Entidade Relacionamento (ER) das entidades envolvidas nas movimentações do inquilino.

Ademais, o saldo do inquilino é um elemento computado e armazenado no banco de

dados, tornando a seleção dos elementos mais leve. Conforme discutido, a necessidade de unir dados de algumas entidades faz com que sejam utilizados mecanismos como triggers, os quais permitem atualizar o atributo quando qualquer uma dessas entidades for alterada.

A Figura 4.6 exibe a interface para emissão de contas para um inquilino. Essa tela apresenta duas opções de contas a serem emitidas: Tenant Bill e Tenant Miscellaneous Bill. A primeira opção é pertinente a casos excepcionais onde é preciso cobrar os custos de estadia fora do fluxo regular de cobrança, pois em casos regulares as contas são emitidas por outro módulo. Enquanto que a segunda opção diz respeito a gastos de tipos variados, entre os tipos conhecidos estão: pagamento diversos (caução, impressão, etc), coimas, etc.

Figura 4.6: Interface para emissão de contas para um inquilino: Primeiro passo, escolha do tipo de conta a emitir.

A Figura 4.7 demonstra o formulário para emissão de uma conta do tipo Tenant Bill.

Figura 4.7: Interface para emissão de contas para um inquilino, tipo Tenant Bill.

A Figura 4.8, por sua vez, demonstra o formulário para emissão de uma conta do tipo Tenant Miscellaneous Bill.

Figura 4.8: Interface para emissão de contas para um inquilino, tipo Tenant Miscellaneous Bill.