,

Universidade Federal de Uberlândia
Faculdade de Computação
Bacharelado em Sistemas de Informação

Steffan Martins Alves

Desenvolvimento de um Sistema de Gestão para Condomínios

Uberlândia
2018

Steffan Martins Alves

Desenvolvimento de um Sistema de Gestão para Condomínios

Trabalho de Conclusão de Curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia como requisito exigido parcial à obtenção do grau de Bacharel em Sistemas de Informação.

Área de concentração: Sistemas de Informação

Orientador: Daniel Antônio Furtado

Uberlândia
2018

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

Os abaixo assinados, por meio deste, certificam que leram e recomendam para a Faculdade de Computação da Universidade Federal de Uberlândia a aceitação do Trabalho de Conclusão de Curso intitulado "**Desenvolvimento de um Sistema de Gestão para Condomínios**" por **Steffan Martins Alves** como requisito exigido parcial à obtenção do grau de **Bacharel em Sistemas de Informação**.

Uberlândia, 05 de Dezembro de 2018

Orientador: _____
Prof. Dr. Daniel Antônio Furtado
Universidade Federal de Uberlândia

Banca Examinadora:

_____
Prof. Dr. André Ricardo Backes
Universidade Federal de Uberlândia

_____
Profª. Drª. Maria Adriana Vidigal de Lima
Universidade Federal de Uberlândia

Este trabalho é dedicado aos síndicos, contadores e administradores que um dia aceitaram a tarefa de gerir um condomínio, mas mal sabiam o que os esperava.

Agradecimentos

Agradeço aos moradores do Condomínio do Edifício Residencial Palmeiras II que me deram a oportunidade de ser seu síndico, o que trouxe a inspiração para desenvolver esta aplicação. Aos meus amigos da faculdade, que além de me motivar também apresentaram algumas das tecnologias que tornaram este projeto possível e nunca deixaram de me ajudar quando os procurei. Aos meus pais, familiares e amigos pela compreensão nas ausências em finais de semana e feriados e por todo apoio que me deram. Ao meu orientador, pelas dicas e sugestões ao longo do trabalho. E a todos os professores que tive a oportunidade de conhecer durante o curso, as suas lições jamais serão esquecidas. Muito obrigado!

"No meio da dificuldade encontra-se a oportunidade."
Albert Einstein

Resumo

Este documento descreve o processo de desenvolvimento de um Sistema de Gestão para Condomínios que tem por objetivo auxiliar os síndicos nos seus deveres cotidianos, agregando segurança e qualidade nas informações através de uma ferramenta moderna e pensada especificamente para o setor condominial que visa a simplificação de rotinas e obrigações. Este sistema foi desenvolvido em Java utilizando o grande potencial do ecossistema Spring e uma arquitetura totalmente na nuvem. Ainda há espaço para mais recursos, mas o sistema já pode contribuir com a ampliação da capacidade administrativa do setor e reduzir seus gastos.

Palavras-chave: Gestão de Condomínios, Síndico, Software para Condomínios, Java, Spring.

Lista de ilustrações

Figura 1 – Tela da aplicação SIN. 18
Figura 2 – Tela da aplicação Immobile Condomínio 19
Figura 3 – Diagrama do Caso de Uso Geral do Sistema 26
Figura 4 – Modelo Entidade-Relacionamento do Banco de Dados 27
Figura 5 – Tela do acesso ao sistema 34
Figura 6 – Tela do painel de gestão do condomínio 35
Figura 7 – Tela de cadastro de moradia 35
Figura 8 – Tela de visualização de moradia 36
Figura 9 – Tela de listagem de movimentos 37
Figura 10 – Modal explicativo de classes de categoria 38
Figura 11 – Tela de cadastro de condômino com mensagens de validação 38
Figura 12 – Modal de confirmação para exclusão de um bloco 39
Figura 13 – Tela de cadastro de cobrança 40
Figura 14 – Telas do sistema na exibição de um *smartphone* 40
Figura 15 – Relatório Balancete visualizado no navegador 41

Lista de siglas

API Application Programming Interface

CNAB Centro Nacional de Informação Bancária

CRUD Create-Read-Update-Delete

CSRF Cross-Site Request Forgery

CSS Cascading Style Sheets

DBaaS Database as a Service

DBMS Database Management System

EJB Enterprise JavaBeans

eSocial Sistema de Escrituração Fiscal Digital das Obrigações Fiscais Previdenciárias e Trabalhistas

FN Forma Normal

GUI Graphical User Interface

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

IaaS Infrastructure as a Service

JPA Java Persistence API

JS JavaScript

JSP Java Server Page

JVM Java Virtual Machine

MVC Model-View-Controller

NoSQL Not Only SQL

ORM Object Relational Mapper

PaaS Plataform as a Service

PF Pessoa Física

PJ Pessoa Jurídica

RF Requisito Funcional

RNF Requisito Não Funcional

SGBD Sistema Gerenciador de Banco de Dados

SQL Structured Query Language

WWW World Wide Web

XMLNS Extensible Markup Language Namespace

Sumário

1 INTRODUÇÃO 12
1.1 Motivação 12
1.2 Objetivo 13
1.3 Contribuições 13
1.4 Organização do Trabalho 14

2 FUNDAMENTAÇÃO TEÓRICA 15
2.1 Conceitos Adotados 15
2.2 Trabalhos Correlatos 17

3 DESENVOLVIMENTO 20
3.1 Modelagem do Sistema 20
3.1.1 Atores 20
3.1.2 Requisitos 21
3.1.3 Caso de Uso 25
3.1.4 Modelo de Dados 26
3.2 Tecnologias Utilizadas 28
3.2.1 Spring Framework 28
3.2.2 Interface Gráfica do Usuário 31
3.2.3 Infraestrutura 32

4 RESULTADOS 34
4.1 Apresentação 34
4.2 Validação 42

5 CONCLUSÃO 44
5.1 Desafios Encontrados 44
5.2 Trabalhos Futuros 45

REFERÊNCIAS **48**

APÊNDICES **51**

APÊNDICE A – EXEMPLOS DE RELATÓRIOS **52**
A.1 Livro Caixa **53**
A.2 Balancete **54**
A.3 Orçamento **55**
A.4 Inadimplência **56**

Capítulo 1

Introdução

Controlar finanças é algo comumente praticado por pessoas e empresas hoje em dia. Pode ser algo manual e simples ou eletrônico e bem elaborado, mas cada entidade monitora, de alguma forma, diversos aspectos de suas atividades econômicas.

No âmbito empresarial a prática é obrigatória, não somente devido à exigência governamental, mas principalmente pela necessidade administrativa das organizações que visam sucesso do próprio negócio, que buscam gerenciar suas atividades, analisá-las e criar estratégias para a sua sustentabilidade e crescimento.

Dentro deste conjunto de entidades há os condomínios, edificações de propriedade coletiva e que, como existe uma pluralidade de sujeitos envolvidos, elegem ao menos um representante, o síndico. Este será o responsável pela administração da sociedade e, desta forma, pela sua saúde financeira.

No Brasil recai sobre o síndico a obrigatoriedade de apresentar aos condôminos (proprietários da edificação) diversos relatórios financeiros e contábeis. O síndico tem a obrigação legal de prestar contas à assembleia dos condôminos (REPÚBLICA, 1964) e elaborar o orçamento da receita e da despesa relativa a cada ano (REPÚBLICA, 2002), além de cumprir e fazer cumprir a convenção, o regimento interno e as determinações da assembleia, textos que são escritos por cada condomínio e que podem estipular outras obrigações, bem como relatórios ou controles financeiros específicos de interesse dos condôminos.

A utilização de uma ferramenta de administração eficaz é essencial para que o síndico consiga cumprir o seu dever.

1.1 Motivação

Diante desta gama de responsabilidades o síndico tem a seu dispor algumas opções: controlar sozinho todos os dados do condomínio e usar cadernos, planilhas eletrônicas ou então adquirir um sistema computadorizado específico para ajudar, ou contratar um especialista ou empresa administradora de condomínios para tal atribuição.

A terceirização pode ser, a princípio, uma boa solução. Entretanto, em um condomínio modesto onde o síndico já precisará conviver com a inadimplência e com gastos imprevistos, ter uma conta extra com administração pode ser inviável financeiramente.

O uso de cadernos é claramente a pior maneira de se resolver o problema. Controles totalmente manuais são ineficazes para grandes volumes de operações e também estão muito sujeitos a erros, falta de informações e dificuldade para localizar e analisar dados. Até mesmo a deterioração pode ser um revés, já que é preciso armazenar todos os registros por pelo menos cinco anos (REPÚBLICA, 1977).

Utilizar documentos ou planilhas eletrônicas é uma alternativa de baixo custo. Como se trata de um sistema extremamente genérico, caberá ao síndico elaborar todos os modelos de relatório, cada fórmula de cálculo, e ainda ter cautela com todos os dados e referências. Portanto o síndico precisa estar familiarizado com pacotes *office* para isto não resultar em desorganização.

Finalmente, a aquisição de um sistema computacional de fim específico também não se mostra, ainda, muito vantajosa. Apesar de trazer toda a organização necessária e dispensar o síndico de certos conhecimentos, as soluções oferecidas hoje possuem alguns gargalos: alto custo para os condomínios, falta de funcionalidades essenciais e dependência de computadores locais ou interfaces gráficas que não proporcionam uma boa experiência ao usuário.

Em um cenário onde a intenção é registrar dados de forma eficiente e simples para atender às obrigações legais, sem agregar custos elevados ao condomínio, as alternativas atuais não são suficientes para oferecer ao setor uma solução satisfatória.

1.2 Objetivo

Este projeto consiste na implementação de um sistema que permita ao síndico gerir seu condomínio a fim de atender a legislação e as demandas dos condôminos, sem absorver altos custos. Para uma melhor experiência o sistema deve ser acessível em diversos dispositivos, contar com armazenamento na nuvem e uma interface amigável e intuitiva que dispense conhecimentos específicos. Desta forma pretende-se que, com poucos comandos, o sistema possa gerar relatórios, controles, posições financeiras e gráficos, além de gerenciar outros aspectos relacionados ao negócio.

1.3 Contribuições

O sistema desenvolvido neste trabalho poderá ser usado por condomínios dos mais variados portes, sendo mais adequado aos residenciais, com baixa arrecadação mensal e que possuem poucos ou nenhum funcionário. Com esta nova ferramenta o setor tem à

disposição uma alternativa para minimizar seus gastos e, ao mesmo tempo, ampliar sua capacidade administrativa.

1.4 Organização do Trabalho

O restante deste documento está organizado como segue:

- **Capítulo 2 — Fundamentação Teórica:** apresenta a revisão bibliográfica com os conceitos fundamentais para o entendimento e desenvolvimento do trabalho e cita alguns sistemas correlatos encontrados no mercado.
- **Capítulo 3 — Desenvolvimento:** Expõe a metodologia aplicada no decorrer do desenvolvimento do trabalho e introduz as ferramentas e *frameworks* utilizados para o desenvolvimento do sistema, além de descrever os serviços selecionados para fornecer a infraestrutura necessária à aplicação.
- **Capítulo 4 — Resultados:** apresenta a aplicação desenvolvida neste projeto, exibindo suas principais características e validando sua adequação à solução do problema proposto através da aplicação de um teste de aceitação.
- **Capítulo 5 — Conclusão:** as considerações finais e contribuições para a realização deste trabalho são destacadas, concluindo o estudo realizado com um destaque dos desafios enfrentados e levantando as intenções de trabalhos futuros para melhoria e ampliação do sistema desenvolvido.

Capítulo 2

Fundamentação Teórica

Este capítulo apresenta a fundamentação teórica e os conceitos necessários para o desenvolvimento deste trabalho, bem como apresenta alguns sistemas já existentes relacionados ao problema.

A seção 2.1 apresenta as tendências tecnológicas, as linguagens de programação e infraestrutura escolhidas para a solução do problema de gestão de condomínios, e a seção 2.2 traz alguns sistemas já conhecidos no mercado que buscam solucionar um pouco deste desafio, a fim de analisá-los e montar uma visão global do cenário atual de gestão de condomínios.

2.1 Conceitos Adotados

Após a popularização da *Internet* e a propagação das redes móveis, como a *3G* e a *4G*, o armazenamento na *nuvem* se tornou uma realidade para muitos. O termo nuvem foi adotado popularmente devido ao modelo deste ambiente, pois o acesso aos programas, serviços e arquivos é remoto, através da *Internet*. A praticidade e segurança oferecidas por esta forma de armazenamento tem atraído diversos usuários, fazendo várias pessoas abandonarem o hábito de armazenar informações em dispositivos locais, como discos rígidos e *pendrives*, para usar um dos inúmeros serviços de armazenamento *online* disponíveis.

Este é um tema que está em alta desde o início da década e que engloba diversas áreas da computação. Graças ao conceito de computação em nuvem, passou-se também o tempo em que era obrigatório instalar determinada aplicação no computador para usufruir de seus benefícios, pois o desenvolvedor pode oferecer este sistema diretamente na *Web*. Os usuários, agora, podem iniciar seu trabalho no escritório e terminá-lo em outro local, usando até mesmo um celular ou um *tablet*.

Para o desenvolvimento de aplicações para a *Internet* a linguagem adotada por padrão para confecção das páginas *Web* é a Hypertext Markup Language (HTML), uma

linguagem de marcação de texto que é interpretada e processada pelos navegadores, tendo assim suas informações renderizadas na tela do dispositivo do usuário de forma gráfica.

Originalmente, a HTML foi projetada principalmente como uma linguagem para descrever semanticamente documentos científicos. Seu design geral, no entanto, permitiu que ela fosse adaptada, nos anos subsequentes, para descrever vários outros tipos de documentos e até mesmo aplicativos (SMITH, 2013). Várias versões da HTML já foram publicadas, sendo a HTML 5 a versão mais recente. Uma das maiores vantagens de desenvolver aplicações baseadas em HTML é sua característica multiplataforma, isto é, tem compatibilidade com dispositivos de diversos sistemas operacionais e diferentes marcas, o que possibilita uma única frente de desenvolvimento e, ao mesmo tempo, garante o alcance a uma parcela consideravelmente maior de usuários.

Entretanto, por si só, esta linguagem não oferece um visual gráfico rico e intuitivo. Por isto o desenvolvimento de uma aplicação para esta linguagem faz uso do Cascading Style Sheets (CSS) e do JavaScript (JS). Em conjunto, estas três linguagens são os principais pilares de desenvolvimento para a World Wide Web (WWW).

A CSS é uma linguagem para descrever a renderização de documentos estruturados (como aqueles criados com a HTML) na tela (ETEMAD; JR.; RIVOAL, 2017) . Trata-se de um meio para adicionar cores, sombras, bordas e algumas animações, dentre outros elementos visuais. Com a CSS também é possível garantir responsividade ao documento, fazendo com que seu conteúdo se adapte ao dispositivo de saída que o está exibindo, garantindo legibilidade e acessibilidade à aplicação independentemente do tamanho ou resolução do aparelho do usuário.

Usando o poder de personalização da CSS 3, a versão mais recente da linguagem, é possível seguir uma tendência atual do mercado, que é a criação de interfaces de usuário seguindo princípios do *Flat Design*, conceito voltado ao estilo minimalista. Este conceito foi amplamente adotado a partir de 2013, quando a *Microsoft*, *Apple* e *Google* começaram a aplicar tais estilos em seus sistemas operacionais e aplicativos, permitindo que os *designs* de interface fossem mais racionais e simples, além de oferecer um visual mais objetivo (WIKIPéDIA, 2018).

A JS é uma linguagem de programação interpretada que pode funcionar diretamente no dispositivo do usuário, executando funções programadas em tempo real, deixando o documento mais dinâmico e interativo. Com a JS podemos, por exemplo, validar formulários, alterar atributos de elementos e capturar eventos.

Apesar da força destas três linguagens, para construir uma aplicação funcional as páginas precisam ser mais dinâmicas, apresentar informações que mudam de acordo com o contexto da aplicação e persistir os dados digitados para futuras consultas e processamentos. As páginas HTML exibem apenas conteúdo estático ao usuário, ou seja, independentemente do momento que a página é acessada ou do usuário que está ativo, a informação exibida é sempre a mesma. Para levar conteúdo dinâmico à HTML é necessário usar uma

linguagem de servidor que faça um pré-processamento e possa retornar, a cada requisição distinta, um conteúdo HTML diferente, quando necessário.

Para o desenvolvimento de aplicações nestes moldes há uma variedade de linguagens, e uma das mais conhecidas e difundidas é a *Java*, uma linguagem de programação orientada a objetos que apresenta alto desempenho e escalabilidade para aplicações na *Web* (BARISH, 2002). Diferente da maioria das linguagens de programação, que são compiladas para código nativo de máquina, a linguagem Java é compilada para um código que só é interpretado por uma máquina virtual específica, a Java Virtual Machine (JVM). Quando se trata de um sistema na nuvem, esta interpretação só ocorre no servidor e o usuário não percebe o processo, pois ele recebe apenas o resultado em HTML desta execução.

E, finalmente, para completar a aplicação é preciso uma estrutura para persistência de dados. Para tanto, pode ser usado um Database Management System (DBMS) que adote o modelo Relacional, uma estrutura de armazenamento que grava os dados de forma que eles sejam percebidos pelos usuários como tabelas (ROUSSOPOULOS; DELIS, 1991). Neste modelo, a linguagem Structured Query Language (SQL) é a mais adotada para efetuar consultas e gravar dados.

Assim, para a construção de uma aplicação completa, que agrade ao usuário em termos de compatibilidade, usabilidade, interatividade, relevância e consistência, o sistema será hospedado em um servidor *online* e construído sob a HTML, melhorado com CSS e JS, montado dinamicamente com Java e persistido em um banco de dados relacional. Desta forma ele ficará acessível ininterruptamente e em qualquer dispositivo do usuário que possa se conectar à rede mundial de computadores e ofereça um navegador *Web*.

2.2 Trabalhos Correlatos

No mercado podemos encontrar alguns sistemas que possuem a mesma proposta deste projeto, como o *SIN* (ICONDEV, 2017), um *software* de gestão de condomínios desenvolvido pela *Icondev*, e o *Immobile Condomínio* (ALTERDATA, 1989), a solução da *Alterdata*.

O SIN oferece aos síndicos ou administradoras de condomínio a tecnologia necessária para gerenciar condomínios com cadastro de usuários, blocos, unidades habitacionais, condôminos, fornecedores, plano de contas, registros de consumo, contas a pagar e a receber, caixa e bancos, boletos ou carnês, renegociações, comunicados internos e acessos na portaria. Além disso ele também oferece relatórios e um portal para o condômino.

Dentre os relatórios disponíveis, destacam-se os relatórios de cadastros, livro diário de lançamentos, demonstrativo de resultado e diversos relatórios financeiros. É possível optar por salvar tais relatórios ou enviá-los por e-mail, de acordo com a necessidade do usuário.

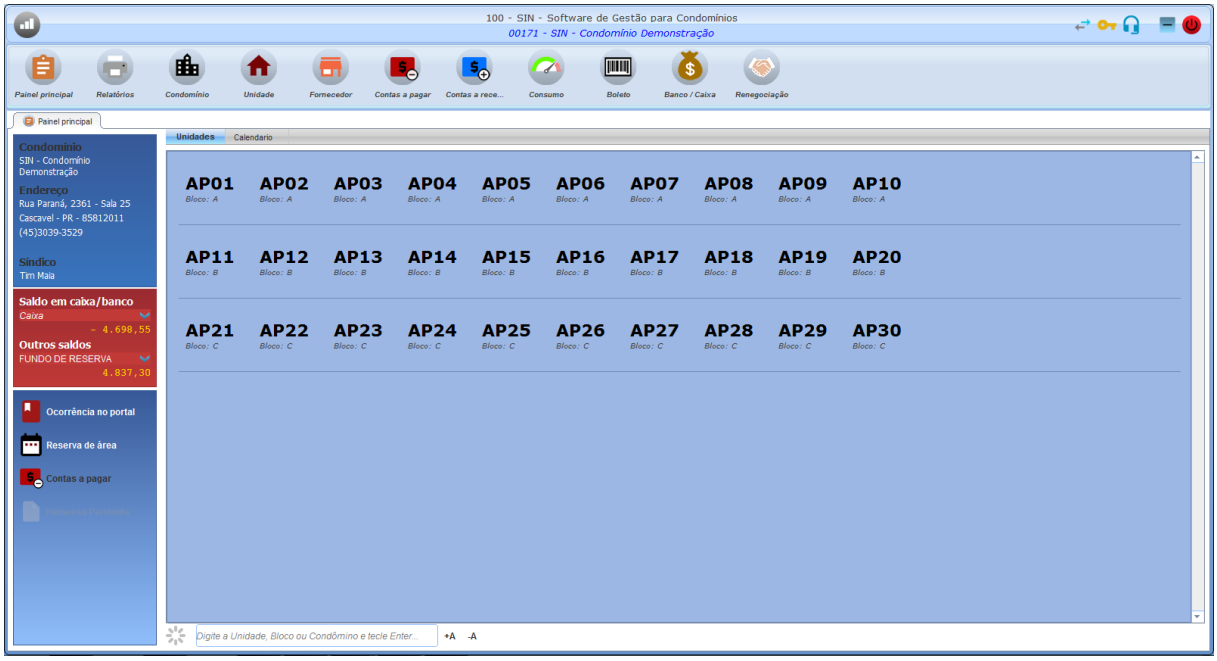


Figura 1 – Tela da aplicação SIN.

Este sistema possui uma tela principal (Figura 1) com informações centralizadas do condomínio, trazendo acesso rápido aos saldos de caixa, contas bancárias e contas contábeis, avisos de contas a pagar, de solicitações de locações de áreas comuns, dentre outros dados de interesse do síndico. As outras telas são acessíveis através de um menu, e todas as operações podem ser realizadas através da interface gráfica da aplicação.

A Icondev também tirou proveito de recursos da computação na nuvem para armazenamento dos dados dos clientes. Assim o sistema pode ser utilizado em qualquer computador conectado à *Internet* e o síndico não precisa se preocupar com questões como vírus ou *backup*, conforme cita a desenvolvedora em sua página (ICONDEV, 2017).

Apesar de bastante completa, esta aplicação possui alguns pontos negativos. Um dos principais equívocos da desenvolvedora foi não fornecer uma funcionalidade para montagem da previsão orçamentária, uma obrigação legal e com penalidades para o condomínio (REPÚBLICA, 2002). O segundo ponto a ser observado é que, apesar de ter um funcionamento totalmente *online*, a aplicação exige a instalação de um componente no computador do usuário e, além disso, como este é um software escrito utilizando a linguagem de programação Java, é necessário que o computador tenha a JVM previamente instalada para que o sistema funcione. Esta questão, além de criar uma barreira de dificuldade para os clientes, os impedem de acessar o sistema pela maioria dos *smartphones* e *tablets* comercializados no mercado, pois estes requisitos limitam o uso do sistema a computadores. Por fim, apesar de simples e intuitiva, a interface gráfica do sistema não é responsiva, ficando ilegível em dispositivos com visores menores.

O Immobile Condomínio é um *software offline* desenvolvido para a gestão de rotinas

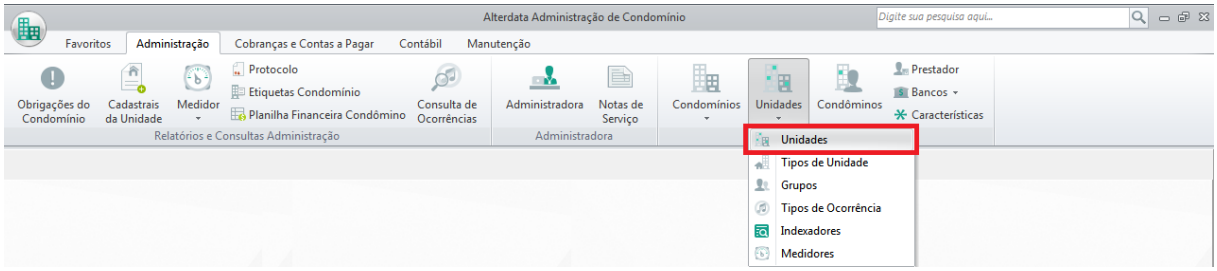


Figura 2 – Tela da aplicação Immobile Condomínio

das administradoras de condomínios residenciais e comerciais, a fim de facilitar o trabalho realizado e a comunicação entre administradora, condôminos e síndicos. O sistema possui diversas ferramentas, muitas opções de relatórios, e também conta com uma interface intuitiva e amigável (Figura 2).

Por uma assinatura à parte, o sistema da Alterdata também pode ser acessado por uma versão online, utilizando-se de virtualização de máquina para oferecer o serviço na nuvem. Este é um serviço adicional onde o usuário vai precisar apenas de um computador para se conectar à *Internet* para abrir o Alterdata Cloud (ALTERDATA, 2018), que está disponível nos servidores da desenvolvedora e permite ao cliente operar o sistema sem precisar se preocupar com a instalação, *backups*, custos com manutenção e infraestrutura ou com ataques ao banco de dados, pois a assinatura inclui estas garantias.

A Alterdata também criou um aplicativo para Android e iOS com alguns recursos básicos para os condôminos, como visualização de boletos, avisos e balancetes, além da possibilidade de reservar áreas de lazer do condomínio.

A Alterdata não deixou de fora a funcionalidade de previsão orçamentária, conforme exigido pelo Código Civil, mas esbarrou nos outros dois problemas da mesma forma que a Icondev. O sistema completo da empresa não tem uma interface responsiva, tornando-o incompatível ou, no mínimo, difícil de usar em *smartphones* e *tablets* devido ao tamanho do visor. Mesmo que a desenvolvedora tenha criado uma versão *mobile*, esta possui apenas recursos de visualização, sendo obrigatório o uso da versão completa no computador. E a versão disponibilizada na nuvem requer uma assinatura adicional, elevando os gastos do condomínio com a ferramenta.

Capítulo 3

Desenvolvimento

Este capítulo apresenta em detalhes a metodologia e as tecnologias utilizadas para o desenvolvimento deste trabalho.

A seção 3.1 descreve passo-a-passo a modelagem da aplicação, desde o levantamento dos atores e requisitos à modelagem do banco de dados, e a seção 3.2 apresenta os *frameworks*, ferramentas e serviços escolhidos para a concepção do projeto.

3.1 Modelagem do Sistema

A modelagem é uma das principais atividades que levam à criação de um bom *software*. A **modelagem de *software*** utiliza vários modelos que "ajudam a visualizar o sistema como ele é ou como desejamos que ele seja; permitem especificar a estrutura ou o comportamento de um sistema; proporcionam um guia para a construção do sistema; e documentam as decisões tomadas no projeto" (RUMBAUGH; JACOBSON; BOOCH, 2005).

Dentre todos os modelos existentes, os atores, os requisitos do sistema, um caso de uso e um modelo de dados foram escolhidos para ajudar a compreender melhor o sistema elaborado.

3.1.1 Atores

Os seguintes atores terão contato com o sistema:

- **Visitante:** é o usuário que chega pela primeira vez ao sistema, não possui um cadastro e nem credenciais para autenticação. Ele não poderá utilizar as funcionalidades de negócio do sistema, apenas poderá criar o seu cadastro e ler as informações públicas disponibilizadas pelo desenvolvedor. Após criar um cadastro e definir credenciais de acesso, este ator deixa de ser um Visitante e se torna um Síndico.

- **Síndico:** é o usuário principal do sistema. Ele acessa o painel do condomínio e todas as funcionalidades disponíveis para gerenciar um único condomínio. Ele não terá acesso nem conhecimento sobre os condomínios gerenciados por outros síndicos.

3.1.2 Requisitos

Os requisitos são as funcionalidades e limitações que estabelecem a forma de funcionamento de um *software*. O levantamento destes requisitos é uma parte fundamental para o desenvolvimento e, quanto mais completo ele for, menos esforço será exigido do desenvolvedor nas fases finais do projeto, onde novos requisitos podem ser identificados. Os requisitos podem ser funcionais ou não, e normalmente todo software possui requisitos de ambos os tipos.

3.1.2.1 Requisitos Funcionais

Um **Requisito Funcional (RF)** diz o que o sistema deve fazer, trata da regra de negócio em si. Baseado no Código Civil e nas Leis brasileiras citadas no Capítulo 1, bem como em casos de uso reais junto a condomínios, os seguintes RFs foram levantados:

RF1 **Cadastro de Usuário:** o sistema deve disponibilizar um formulário para registro do síndico de forma independente de outro usuário ou administrador.

RF2 **Redefinição de Senha:** o sistema deve enviar um *link* por *e-mail* para redefinição de senha, caso um usuário esqueça a mesma.

RF3 **Autenticação:** o sistema deve ter uma tela de entrada (*log-in*) para que o síndico possa informar suas credenciais e acessar a área restrita da aplicação. Nesta área, além das funcionalidades esperadas da aplicação, também deve existir uma opção de sair do sistema e finalizar a sessão (*log-out*).

RF4 **Alteração de Dados:** o sistema deve fornecer ao usuário a possibilidade de alteração de seus dados pessoais e credenciais de acesso.

RF5 **Cadastro do Condomínio:** o sistema deve ter um formulário para que o síndico possa cadastrar os dados de seu condomínio, bem como alterá-los a qualquer tempo.

RF6 **Cadastro de Blocos:** o sistema deve ter um Create-Read-Update-Delete (CRUD) para blocos do condomínio. Um condomínio deve suportar um ou mais blocos.

RF7 **Cadastro de Moradias:** o sistema deve ter um CRUD de moradias, que são as unidades habitacionais dentro de um bloco. Cada bloco deve comportar inúmeras moradias.

RF8 **Cadastro de Condôminos:** o sistema deve ter um CRUD de condôminos. Os condôminos podem ser proprietários ou inquilinos das moradias, e podem ser uma Pessoa Física (PF) ou Pessoa Jurídica (PJ).

RF9 **Cadastros Cruzados:** ao cadastrar um condômino, o síndico deve ser capaz de relacioná-lo a uma moradia previamente cadastrada, e vice-versa. Uma moradia pode pertencer ou estar locada para mais de um condômino, bem como um condômino pode possuir ou locar mais do que uma moradia.

RF10 **Cadastro de Contas:** o sistema deve ter um CRUD de contas do condomínio. Esta contas podem ser bancárias ou não, e deve haver a possibilidade de informar um saldo inicial. A qualquer momento o síndico deve ser capaz de saber o saldo atual das contas.

RF11 **Cadastro de Cobranças:** o sistema deve ter um CRUD para emissão de cobranças. Uma cobrança deve sempre ser relacionada a uma moradia. Após o recebimento, a cobrança deve mudar de estado.

RF12 **Atualização de Cobranças:** as cobranças vencidas devem ter seus valores atualizados da seguinte maneira:

Multa: é o valor original da cobrança multiplicado pelo percentual de multa.

Juros: é o valor original da cobrança multiplicado pelo percentual mensal de juros previamente dividido por trinta, resultado que é ainda multiplicado pela quantidade de dias corridos entre a data de vencimento e a data atual.

Total: é o valor original da cobrança, acrescido da multa e dos juros calculados, bem como dos outros acréscimos ou decréscimos já registrados na cobrança.

RF13 **Relatório de Inadimplência:** o sistema deve fornecer ao síndico um relatório de inadimplência contendo os valores totais de todas as cobranças vencidas já devidamente atualizadas. O relatório deve ser agrupado por moradia e ordenado por data de vencimento.

RF14 **Cadastro de Categorias:** o sistema deve oferecer um CRUD de categorias para organizar os tipos de receitas e despesas do condomínio. As categorias devem suportar escalonamento, partindo de grupos gerais para mais específicos, em até 4 níveis.

RF15 **Cadastro de Períodos:** o sistema deve ter um CRUD de períodos de gestão. Períodos iniciam e terminam em datas determinadas pelo síndico e não podem ser sobrepostos. Uma vez encerrado, orçamentos e movimentos dentro deste período não devem ser modificados. Períodos, entretanto, podem ser reabertos.

RF16 **Cadastro de Orçamentos:** o sistema deve ter um CRUD para o registro da Previsão Orçamentária de determinado período de gestão, onde cada entrada deve especificar uma categoria e um valor pretendido para aquele período.

RF17 **Cadastro de Movimentos:** o sistema deve ter um CRUD para registrar movimentações dentro de um período de gestão. Os movimentos podem ser transferências ou lançamentos. As transferências são movimentações entre contas, portanto possuem duas contas envolvidas. Os lançamentos são registros de receitas ou despesas e precisam ser relacionados a uma conta e uma categoria.

RF18 **Relatório de Orçamento:** o sistema deve apresentar ao síndico um relatório de orçamento mostrando, para determinado período de gestão, os valores e percentuais orçados e realizados em cada categoria. O relatório deve mostrar todos os níveis de categorias, bem como seus subtotais. Categorias sem orçamento e sem movimentos não devem ser apresentadas.

RF19 **Balancete:** o sistema deve apresentar ao síndico um relatório de prestação de contas no modelo de balancete. O relatório deve trazer analiticamente as receitas e despesas realizadas em um intervalo de datas determinado, bem como o resultado positivo ou negativo destes lançamentos.

RF20 **Livro Caixa:** o sistema deve fornecer ao síndico a possibilidade de gerar um relatório listando os lançamentos em um determinado intervalo no modelo de livro caixa. Este relatório deve exibir o saldo inicial e final do período, considerando todas as contas, e ser ordenado por data.

RF21 **Painel de Gestão:** o sistema deve ter um painel principal centralizando informações relevantes ao síndico, como saldo em conta, total da inadimplência, resumo da receita e da despesa do mês bem como dos valores orçados e os já realizados no período de gestão atual.

3.1.2.2 Requisitos Não Funcionais

Um **Requisito Não Funcional (RNF)** indica como o sistema deve realizar suas tarefas. Eles tratam de aspectos como a usabilidade, desempenho e segurança do sistema, dentre outros. Para este projeto os seguintes RNFs foram considerados relevantes:

RNF1 **Cifragem de Senha:** as senhas escolhidas pelos usuários devem ser processadas pelo algoritmo **Bcrypt** (PROVOS; MAZIERES, 1999).

RNF2 **Manter Conectado:** o sistema deve fornecer ao usuário a opção de mantê-lo conectado, sem que sejam exigidas credenciais de entrada ao usar o mesmo dispositivo.

RNF3 **Expiração de Token:** o *link* para redefinição de senha não deve funcionar no dia seguinte e nem ser utilizado para gerar mais de uma nova senha.

RNF4 **Envio de *e-mails*:** sempre que o sistema enviar *e-mails* deverá fazê-lo de forma assíncrona para não prender o usuário na tela durante sua execução.

RNF5 **Dependências dos Cadastros:** o sistema não pode permitir que o síndico cadastre blocos, contas, condôminos, categorias ou períodos sem ter finalizado o cadastro principal do condomínio. Da mesma forma, moradias não podem ser criadas sem ao menos um bloco registrado, cobranças precisam previamente de cadastros de condôminos e moradias, orçamentos exigem a existência de períodos e categorias, e movimentos requerem previamente contas e períodos, além de categorias caso estes não sejam transferências.

RNF6 **Registros Únicos:** Nomes de usuário não podem se repetir. Além disso, um mesmo condomínio não pode ser registrado por mais de um síndico. De forma semelhante, blocos, moradias, condôminos, contas, cobranças, categorias, períodos, orçamentos e movimentos não podem se repetir no condomínio.

RNF7 **Validação de Entradas:** o sistema deve verificar entradas informadas pelo usuário, como endereços de *e-mail*, CNPJs, CPFs, datas e valores monetários (inclusive cálculos), dentre outras que sejam passíveis de validação, impedindo o registro de valores incorretos e fornecendo *feedback* ao usuário.

RNF8 **Valores Padrão:** todas as variáveis de entrada essenciais para o funcionamento do sistema precisam ter valores padrão e tais valores devem ser usados sempre que tais dados de entrada forem omitidos.

RNF9 **Paginação de Listas:** todos as entidades devem ter uma página de listagem que exibe o que já foi cadastrado e fornece opções de visualização, edição e exclusão. Esta lista deve ser paginada, e somente os resultados a serem exibidos na página atual devem ser recuperados do DBMS. Cada página poderá exibir, por padrão, até 20 registros.

RNF10 **Exclusão em Cascata:** o sistema não deve deixar restos de informação não recuperáveis ou sem referência no banco de dados. Se um registro for excluído e possuir registros filhos, estes também devem ser apagados. Neste caso, o usuário deve ser informado e precisará confirmar a operação.

RNF11 **Transações com o DBMS:** em caso de qualquer tipo de falha, o sistema não pode comprometer a integridade dos dados. Desta forma, todas as operações de leitura ou gravação no banco de dados devem ser feitas dentro de transações.

RNF12 **Rotina de Atualização de Cobranças:** as cobranças vencidas devem ser atualizadas diariamente às 00:01h através de um evento automático e recorrente.

RNF13 **Rotina de Atualização de Saldo:** imediatamente após a inclusão, alteração ou exclusão de um movimento, o saldo atual da conta relacionada deve ser atualizado.

RNF14 **Cálculo do Total da Cobrança:** o sistema não deve solicitar que o usuário informe o total da cobrança; este deve ser calculado a partir dos demais campos monetários da cobrança.

RNF15 **Painel de Gestão com Gráficos:** os resumos da receita e da despesa do mês, dos valores orçados e dos já realizados no período devem ser exibidos de forma gráfica no painel de gestão do condomínio.

RNF16 **Relatórios para Impressão:** os relatórios gerados pelo sistema devem ser exibidos já no formato apropriado para impressão em papel A4.

RNF17 **Acessos simultâneos:** o sistema deve permitir que vários usuários o utilizem paralelamente.

RNF18 ***Pool* de Conexões com o DBMS:** o sistema deve manter um número configurável de conexões abertas simultaneamente com o banco de dados, utilizando um *pool* de conexões para atender requisições simultâneas dos usuários.

RNF19 **Disponibilidade:** o sistema deve estar disponível 99% do tempo.

RNF20 **Interface:** o sistema deverá ser acessado completamente via navegadores *Web* exigindo do usuário apenas suporte a HTML 5, CSS 3 e JS.

RNF21 **Compatibilidade com Navegadores:** o sistema deve funcionar nas versões mais recentes dos navegadores Chrome, Internet Explorer, Edge, Firefox, Safari e Opera.

RNF22 **Compatibilidade com Dispositivos:** o sistema deve ser legível em qualquer dispositivo que possua um navegador *Web* compatível e visor com pelo menos 3,5 polegadas, inclusive quando possuírem tecnologia *touchscreen*.

RNF23 **Versionamento:** uma ferramenta **Git** deve ser utilizada durante todas as fases do desenvolvimento para registrar a evolução do projeto.

3.1.3 Caso de Uso

Dentre outras características, os **casos de uso** fazem a modelagem da interação entre atores e o sistema, enfatizando os objetivos e perspectivas do usuário. Com base nos atores e requisitos especificados até aqui é possível mostrar toda a funcionalidade do sistema através de um diagrama genérico e de alto nível de casos de uso (Figura 3).

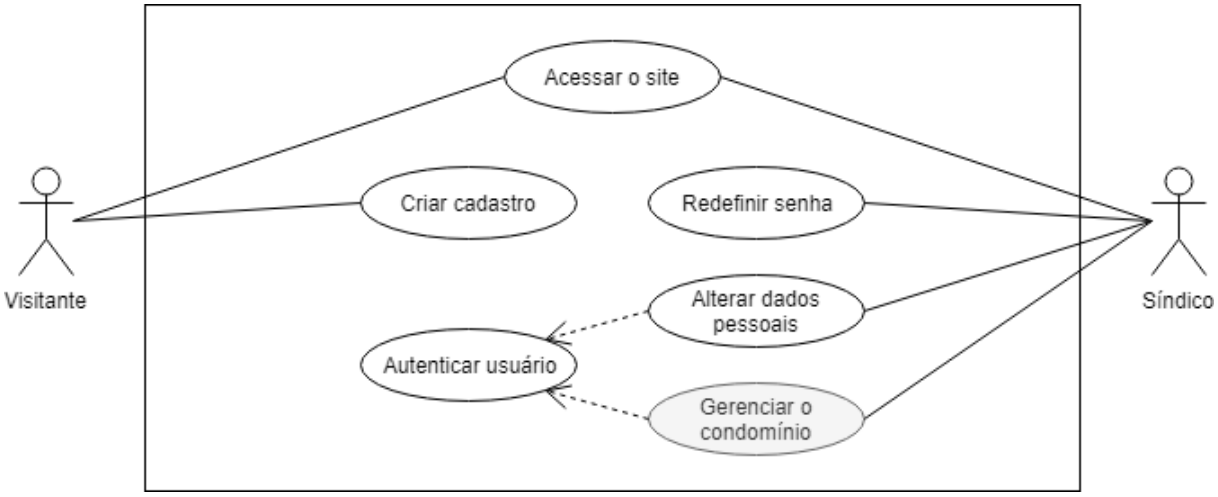


Figura 3 – Diagrama do Caso de Uso Geral do Sistema

Neste diagrama, o caso de uso 'Gerenciar o condomínio' engloba as demais atividades presentes no sistema e disponibilizadas para o Síndico, representando as funcionalidades do RF5 ao RF21, bem como os RNF relacionais a eles.

3.1.4 Modelo de Dados

A modelagem do banco de dados é um passo crucial para o desenvolvimento. Definir as entidades e seus atributos, bem como as as relações ou dependências entre elas é a base para a criação do banco de dados e de toda lógica computacional envolvida na persistência e recuperação das informações. Um banco de dados mal estruturado pode dificultar a gravação e a recuperação de informações, gerar consultas repetitivas e demoradas, duplicar informações e até mesmo comprometer a sua integridade.

Após montado, um esquema relacional precisa passar por um processo de normalização para identificar erros. Estas regras visam reduzir a redundância de dados e aumentar a sua integridade e desempenho (ELMASRI; NAVATHE, 2011). Para ser considerado adequado, o esquema relacional deve ser analisado e adaptado a cada Forma Normal (FN) a seguir:

- **1ª FN:** todos os atributos de uma entidade precisam ser atômicos e monovalorados, o que significa que uma tabela não pode ter grupos de repetição.
- **2ª FN:** os atributos não chave de uma entidade devem depender unicamente de sua chave primária.
- **3ª FN:** os atributos não chave de uma entidade devem ser funcionalmente independentes uns dos outros.

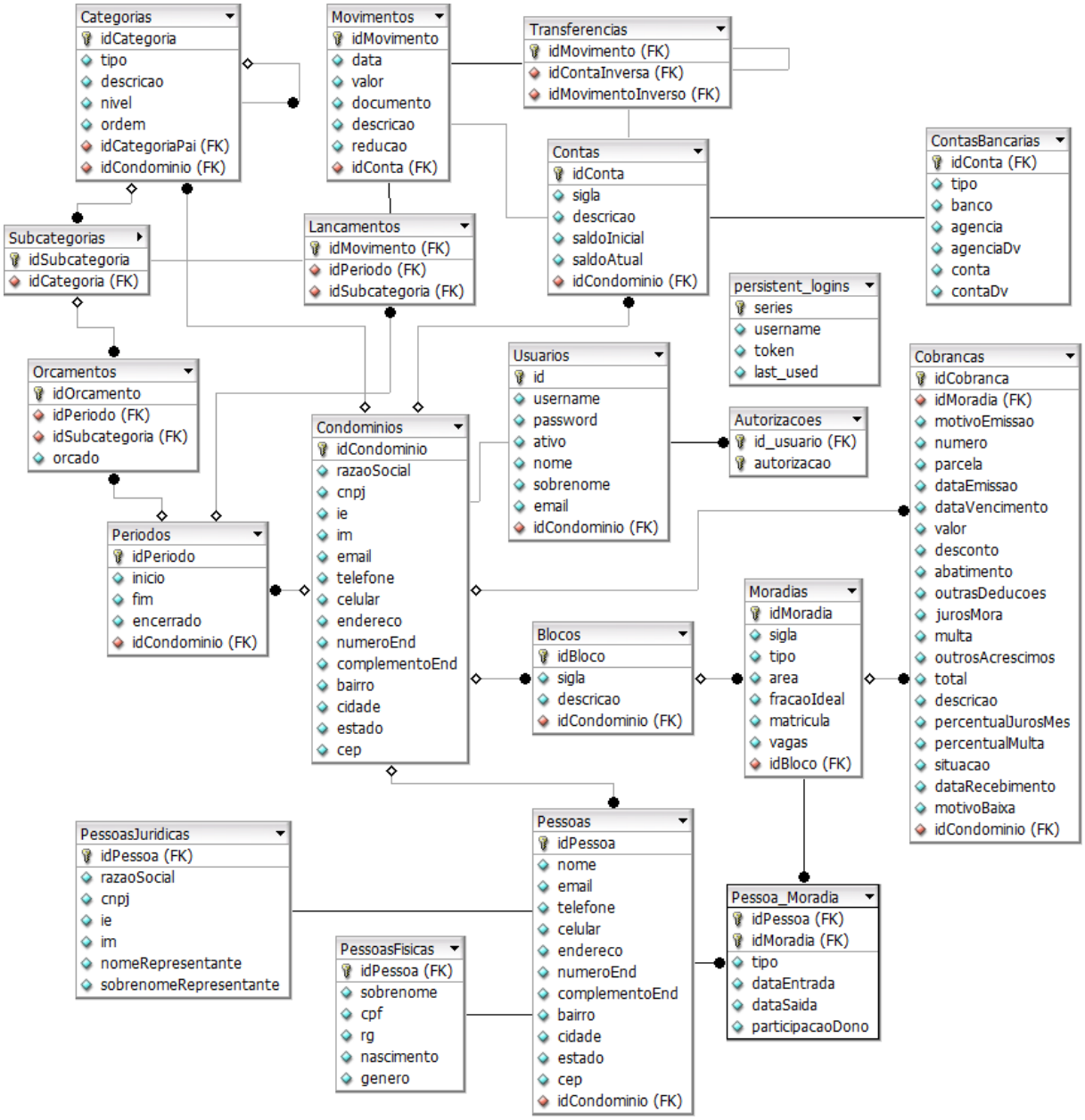


Figura 4 – Modelo Entidade-Relacionamento do Banco de Dados

Alguns autores modernos preveem outras FNs, entretanto este trabalho se limitou a estas três regras, as quais já foram amplamente adotadas e difundidas no meio acadêmico.

A relação do condômino com a moradia é construída respeitando estas regras, uma vez que este poderá possuir ou locar mais de uma moradia. E como o condômino pode ser tanto um proprietário quanto um locatário, esta relação carrega um atributo para especificar o tipo do condômino, o que não é restrito apenas a estas duas opções.

O escalonamento de categorias previsto no RF14 foi resolvido com uma chave estrangeira que referencia a própria entidade. Também está presente no modelo de dados o conceito de herança, evitando repetição de informações comuns entre as pessoas físicas

e jurídicas e entre as contas, que são normais ou bancárias.

Todo Movimento é um Lançamento ou uma Transferência, e esta especificação foi modelada com a utilização de herança. Os atributos comuns entre Lançamentos e Transferências são armazenados apenas na tabela Movimento, e somente as informações específicas de cada tipo de movimento são alocadas nas tabelas de Lançamentos ou Transferências.

O Esquema Relacional deste projeto (Figura 4) buscou observar estas regras em cada uma das entidades e relações modeladas, com uma única exceção: de acordo com as especificações do RF12 e do RNF14, a entidade 'cobrança' precisa manter um valor 'total', que deriva dos demais valores monetários da entidade. Neste caso, entretanto, um *trigger* programado no Sistema Gerenciador de Banco de Dados (SGBD) irá garantir que o 'total' seja corretamente preenchido sempre que um registro for criado ou modificado.

3.2 Tecnologias Utilizadas

Para o desenvolvimento de uma aplicação de sucesso é preciso estar atento aos passos tecnológicos tomados pelo mercado. Um sistema, para ser bem colocado, consiste não somente em um *software* funcional e que atenda aos requisitos, mas também que possua um código fonte estruturado, de fácil compreensão e manutenção, além de fácil escalabilidade, maior segurança e total integridade da informação. No sentido de facilitar estas características, a comunidade de desenvolvedores e empresas de tecnologia tem criado e mantido inúmeras ferramentas para apoiar o desenvolvimento de aplicações. Neste trabalho procurou aproveitar-se destas tecnologias para trazer ao projeto uma qualidade mais próxima ao que é criado hoje pelas empresas de desenvolvimento, tendo uma estrutura bem conhecida pelos profissionais da área, além de segurança e integridade garantidas pelas ferramentas adotadas.

3.2.1 Spring Framework

O **Spring** é um *framework* de código aberto para a plataforma Java criado por Rod Johnson (JOHNSON, 2002) e baseia-se nos padrões de projeto 'inversão de controle' e 'injeção de dependência'. Desta forma o programador não se preocupa com a sequência das chamadas dos métodos nem com as dependências entre os módulos e pode focar seus esforços nas regras do negócio.

O Spring é gratuito (PIVOTAL, 2018d), e uma das características observadas na escrita do código-fonte quando se usa este *framework* é a presença de anotações e arquivos de configuração, que são processados pelo Spring em tempo de execução, facilitando também o desenvolvimento em diversos ambientes diferentes com variáveis personalizadas.

Este *framework* é modularizado e pode ser utilizado de acordo com as necessidades do projeto. Cada módulo traz bibliotecas com finalidades específicas, mas totalmente

integradas entre si. Neste projeto, como há necessidade de uma interface *Web* com autenticação de usuários e persistência de dados, os módulos **Spring Web MVC**, **Spring Security** e **Spring Data** foram acoplados. Além destes, para facilitar a configuração do *framework* e a distribuição do *software* final, o módulo **Spring Boot** também foi adicionado. Cada um destes componentes é explicado a seguir.

3.2.1.1 Spring Web MVC

O módulo Spring Web MVC (PIVOTAL, 2018f), como o próprio nome já declara, traz consigo a estrutura do padrão de arquitetura Model-View-Controller (MVC), já dispensando o programador de mais esta tarefa no desenvolvimento.

Outra vantagem deste módulo é a conversão de dados enviados através de formulários *Web*. Em um cenário padrão, quando o desenvolvedor recebe dados de um formulário, tudo é interpretado como texto, inclusive datas e valores numéricos — já que este é o comportamento do Hypertext Transfer Protocol (HTTP) — e o programador precisa realizar a conversão de cada uma das entradas manualmente. Com esta ferramenta os dados já são convertidos de acordo com o tipo das variáveis que os recebem, sem necessidade de código adicional.

Indo ainda mais longe, o módulo é capaz de instanciar objetos inteiros a partir da submissão de um formulário. Por exemplo: suponha a classe Pessoa com os atributos nome, idade e sexo, e também imagine um formulário de cadastro com os mesmos campos, ao receber os dados deste formulário é possível optar por receber uma instância da classe Pessoa, já com todos os atributos preenchidos, ao invés de ler cada variável separadamente para construir um objeto deste tipo.

3.2.1.2 Spring Security

O Spring Security (PIVOTAL, 2018e) é o bloco responsável pelo gerenciamento de usuários, incluindo a cifragem de senhas, autenticação, sessões e autorização através de papéis, além de fornecer proteção contra diversos ataques à aplicação.

Todos os aspectos deste módulo são configuráveis, como o tempo da sessão, o algoritmo de cifragem das senhas e também as chamadas que podem ser realizadas por determinado papel de usuário a métodos, objetos e, principalmente, requisições *Web*, pois graças à sua integração com o Spring Web MVC visto na seção 3.2.1.1, antes de atender a uma requisição *Web* o controlador irá verificar junto a este módulo se o usuário da sessão atual possui tal permissão.

Com este componente a aplicação também fica protegida contra ataques como Cross-Site Request Forgery (CSRF), *Session Fixation*, *Clickjacking*, dentre outros, pois o sistema irá analisar cada requisição e barrar grande parte do conteúdo malicioso.

3.2.1.3 Spring Data

A persistência de dados pode ser um processo trabalhoso e repetitivo para a comunidade, uma vez que a programação é orientada a objetos e o banco de dados é relacional e formado por tabelas. Uma série de códigos em SQL precisa ser misturada à logica da aplicação, e para executá-los é necessário gerenciar uma conexão com o banco de dados, manipular objetos de consulta e os seus resultados.

Para minimizar este esforço a comunidade desenvolveu a tecnologia Object Relational Mappers (ORMs), *frameworks* capazes de mapear um objeto a uma tabela, bem como as relações entre eles, dispensando a utilização de códigos SQL nativos e manipulação direta de consultas e resultados, além de controlar conexões e transações. Esta iniciativa, entretanto, veio despadronizada e dificultava a migração de um ORM para outro, uma vez que os modelos eram incompatíveis. Com isto surgiu o Java Persistence API (JPA), uma Interface de Programação de Aplicativos, do inglês Application Programming Interface (API), com o intuito de padronizar tais *frameworks* de persistência de dados.

Em um projeto Spring, o Spring Data (PIVOTAL, 2018b) tem por objetivo facilitar o trabalho com a persistência de dados de uma forma geral, em qualquer tipo de base de dados, desde os tradicionais modelos relacionais às base de dados Not Only SQL (NoSQL). Dentro deste módulo há um sub-componente chamado Spring Data JPA (PIVOTAL, 2018c), que é uma abstração da API JPA e visa facilitar ainda mais o seu uso pelo desenvolvedor em um projeto Spring, trazendo suporte ao JPA para todo o ambiente de maneira integrada.

Estas adições trazem ao projeto vantagens como interfaces prontas para CRUD, criação automática de consultas com base no nome do método, possibilidade de paginação dos resultados, conversão de tipos entre a aplicação e o banco de dados, validação de dados, prevenção contra ataques como o *SQL Injection*, dentre outros recursos.

Entretanto, como o JPA é apenas uma especificação, e o Spring Data JPA é apenas um pacote para integrar, dar suporte e facilitar o uso do JPA no ambiente Spring, uma implementação da API propriamente dita ainda se faz necessária. Desta forma, o ORM **Hibernate** (REDHAT, 2018) é adotado para implementar todos estes recursos. Este *framework* foi um dos criados pela comunidade e que depois padronizou-se à JPA, sendo hoje um dos mais adotados no mundo.

3.2.1.4 Spring Boot

Com todos estes módulos e possibilidades de personalização oferecidos pelo Spring, uma central de configuração do projeto se torna indispensável. O intuito do Spring Boot (PIVOTAL, 2018a) é facilitar a criação de projetos, sua configuração e, ainda, sua execução.

Com este módulo, várias configurações que precisariam ser especificadas pelo de-

senvolvedor em cada um dos outros módulos são centralizadas em um único local ou até mesmo automatizadas, trazendo clareza e facilitando a criação do projeto. Mas a mais importante funcionalidade do Spring Boot é vista na fase de implantação do sistema, onde há possibilidade de criar aplicações autossuficientes (*stand-alone*) com poucas instruções.

No modelo tradicional, para executar uma aplicação Java é necessário que a máquina possua uma JVM. Uma aplicação Java para *Web* precisa de um servidor provendo Web Containers e Enterprise JavaBeans (EJB) Containers, como o **Apache** (APACHE, 2018). Com o Spring Boot estes requisitos são dispensados, pois ele é capaz de encapsular dentro do próprio pacote o interpretador necessário, tornando o software autossuficiente e facilitando, também, sua distribuição.

3.2.2 Interface Gráfica do Usuário

Se de um lado existem *frameworks* para facilitar o desenvolvimento de aplicações minimizando escrita de código no *back-end*, de outro lado também surgiram ferramentas que visam melhorar a criação de Graphical User Interfaces (GUIs), a camada visível do sistema, ou *front-end*. Para uma aplicação *Web*, as tecnologias primordiais são o HTML, JS e CSS, já discutidas no Capítulo 2.1, mas sua aplicação é potencializada, facilitada e refinada com os recursos a seguir apresentados.

3.2.2.1 Thymeleaf

Para trazer conteúdo dinâmico a páginas *Web* de uma aplicação Java, o HTML é embutido em Java Server Pages (JSPs), onde códigos em Java são intercalados com conteúdo escrito em HTML. Estas páginas são sempre interpretadas pelo servidor, onde todo código Java é executado e substituído pelo seu resultado, para então ser enviado ao cliente de forma natural. Sem a interferência do servidor elas não seriam visualizadas corretamente, pois o código escrito em Java é ilegível para o navegador.

Da mesma forma que a mistura de SQL ao Java causa incômodos, a mistura de Java ao HTML também não agrada, pois dificulta a colaboração entre equipes de desenvolvimento, deixa a página desorganizada e com muita lógica de programação mesclada às marcações da interface. Para resolver este conflito vieram as *Template Engines*, ferramentas que possibilitam a escrita de código com lógica computacional usando somente marcações válidas de HTML, CSS e JS. Isto não significa que os navegadores serão capazes de interpretar e executar a lógica ali contida, mas estes poderão ignorá-las e apresentar a página *Web* sem quebras nem códigos ilegíveis quando um servidor não está disponível.

O **Thymeleaf** (THYMELEAF, 2018) é um *Template Engine* moderno e dedicado ao Java e possui módulos adicionais para integração ao Spring Framework e ao Spring Security, o que o torna ainda mais poderoso neste projeto. Com o Thymeleaf não é mais necessária a utilização de JSPs e parcelas de código em Java, e sim somente conteúdo

em HTML. Toda parte lógica é escrita nativamente definindo um Extensible Markup Language Namespace (XMLNS) no início do documento, tornando aquela marcação suportada pelos navegadores.

Não obstante, o uso do Thymeleaf também facilita a criação de iterações, laços, condições e estruturas de dados necessárias para atingir o dinamismo desejado em cada página *Web*, pois abstrai os conceitos do JSP e fornece um dialeto próprio mais simples e com inúmeras funcionalidades. Sua integração com o Spring melhora a conversão de tipos entre a aplicação e a camada *Web*, além de simplificar o uso de autorizações para exibir ou não certos elementos na página de acordo com o papel do usuário da sessão.

Por fim, dentre outros recursos, cabe destacar que o Thymeleaf também proporciona um alto nível de reaproveitamento de código através da possibilidade de criação de moldes para páginas inteiras ou blocos de elementos menores, onde trechos que se repetem em diversas páginas ou lugares (como um menu de navegação ou um rodapé, por exemplo) podem ser definidos uma única vez e referenciados depois em todos os outros locais, o que ainda facilita futura modificações, poupando o desenvolvedor do trabalho de replicar alterações em diversos lugares.

3.2.2.2 Outras Ferramentas

A variedade de dispositivos com conexão à *Internet* hoje torna essencial criar sites voltados ao mercado *mobile*, e o **Bootstrap** (OTTO, 2018) é um dos *frameworks front-end* que ajuda os programadores a fazer isto acontecer sem precisar digitar uma linha de CSS. Criar uma página usando os componentes e elementos da biblioteca Bootstrap, além de já torná-lo responsivo, geralmente deixa a interface mais bonita e eficiente, uma vez que há facilidade para implementar barras de navegação, modal, *slideshows*, dentre outros sem a menor dificuldade.

A combinação do **Font Awesome** (FONTICONS, 2018) ao Bootstrap melhora ainda mais a experiência do usuário, pois traz ícones à interface e torna possível a aplicação de um conceito de Interação Humano-Computador, a metaforização de ações do sistema a partir de elementos do mundo real, permitindo ao usuário a criação de modelos mentais eficazes, melhorando sua produtividade ao utilizar o sistema.

Há ainda uma ferramenta para criação de gráficos com pouquíssimo esforço do programador, o **Chart.js** (TIMBERG, 2018). Esta ferramenta cria automaticamente gráficos em HTML 5 a partir de simples comandos em JS. Traz vários modelos de gráficos e uma vasta gama de opções para personalização.

3.2.3 Infraestrutura

Todo sistema precisa de uma infraestrutura física para ser executado. O intuito é oferecer esta aplicação na *Internet*, então é essencial contar com um servidor sempre

online que seja capaz de responder a requisições simultâneas e que possa suportar o crescimento do *software* ao longo do tempo, tanto em termos de aumento de usuários quanto de utilização de memória e armazenamento. Além disso é preciso fazer *backups* regularmente para prevenir a perda dos dados dos usuários.

O investimento em *hardware* e sua manutenção adequada tem um custo elevado, além de exigir profissionais capacitados. Uma alternativa recente a este investimento é a contratação de serviços na nuvem para execução e oferta do sistema, podendo ter um custo menor e tirando do desenvolvedor a responsabilidade de manutenção da infraestrutura do seu *software*, preocupando-se somente com o código em si.

3.2.3.1 Heroku

O **Heroku** (SALESFORCE, 2018) é um fornecedor de Plataform as a Service (PaaS), um tipo de solução que vem de encontro com a necessidade de abstrair o desenvolvedor dos detalhes de infraestrutura, disponibilizando espaços para instalação das aplicações, o que facilita a manutenção, extensão e escalabilidade, dando maior agilidade para disponibilizar uma aplicação na *Web* com um custo inicial menor.

Diferentemente da Infrastructure as a Service (IaaS), a qual o cliente contrata máquinas e é responsável pela instalação de bibliotecas e montagem de todas as estruturas do sistema, dentre outros recursos, o PaaS é ainda mais específico e de alto nível, pois o fornecedor já entrega um ambiente pronto para executar determinado tipo de aplicação, como, por exemplo, um com capacidade de executar um sistema baseado em Java.

No mercado há uma infinidade de fornecedores de PaaS, como o AWS (Amazon) e o Azure (Microsoft), dentre outros. Basicamente o que difere cada um deles são as linguagens que eles permitem utilizar, a quantidade de serviços adicionais, o preço e também facilidade de criação e manutenção de uma aplicação. Neste último quesito o Heroku se destaca, pois criar, manter e escalar uma aplicação é bem mais simples que nos demais.

3.2.3.2 JawsDB

O serviço do Heroku visto na seção anterior não traz consigo um banco de dados, pois ele é focado na execução da aplicação. Assim é necessário buscar uma solução para a persistência das informações. O **JawsDB** (JAWSDB, 2018) é um fornecedor, dentre muitos outros, de Database as a Service (DBaaS), uma abordagem para armazenamento e gerenciamento de dados estruturados na nuvem que também suporta escalabilidade, *backups* e replicação de dados em diversos servidores para maior disponibilidade.

Este projeto adota o MySQL, entretanto a migração para outro tipo de banco de dados não é um gargalo, visto que a uniformidade do SQL e a arquitetura deste projeto possibilitam a alteração do DBMS sem necessidade de revisão do código, limitando-se apenas à replicação dos dados já existentes em um banco de dados para o outro.

Capítulo 4

Resultados

Este capítulo apresenta as partes mais relevantes do sistema desenvolvido. Algumas das funcionalidades implementadas são descritas e as imagens das respectivas telas são apresentadas na seção 4.1. Na seção 4.2 é apresentado um caso de uso prático do sistema, definido como forma de validação do mesmo.

4.1 Apresentação

Este sistema é uma aplicação *Web* composta por duas áreas: uma de acesso público, que não necessita de autenticação; e outra de acesso restrito, que necessita de um nome de usuário e senha. Após se autenticar (Figura 5), o síndico é redirecionado para o painel de gestão (Figura 6), uma central de informações com cartões e gráficos.

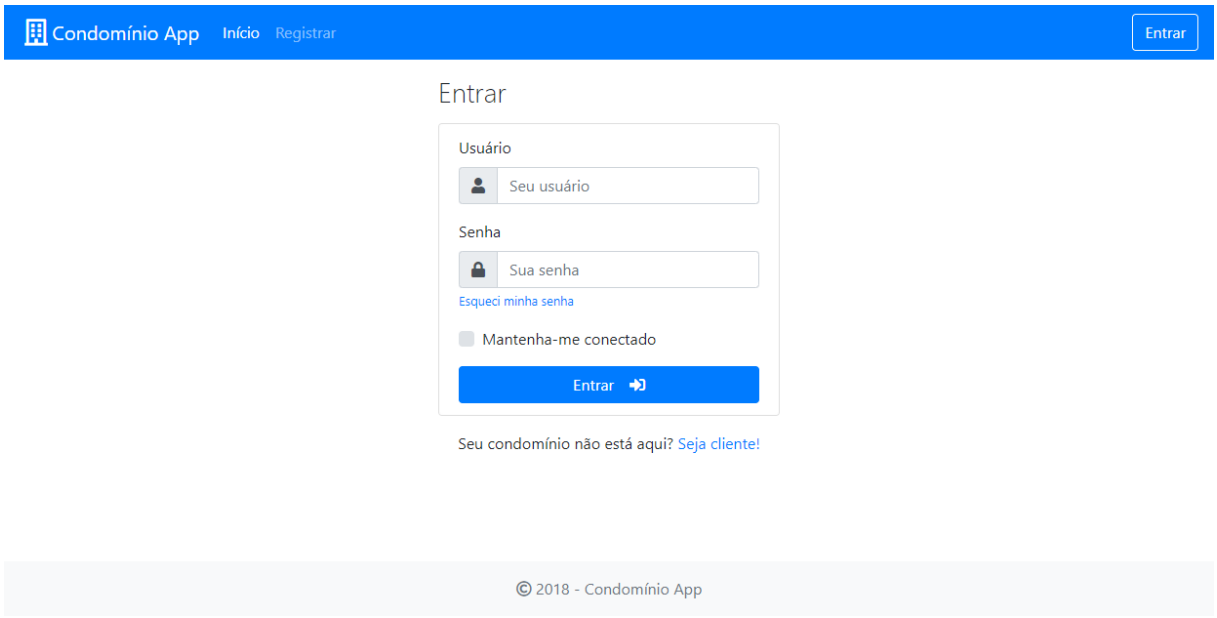


Figura 5 – Tela do acesso ao sistema

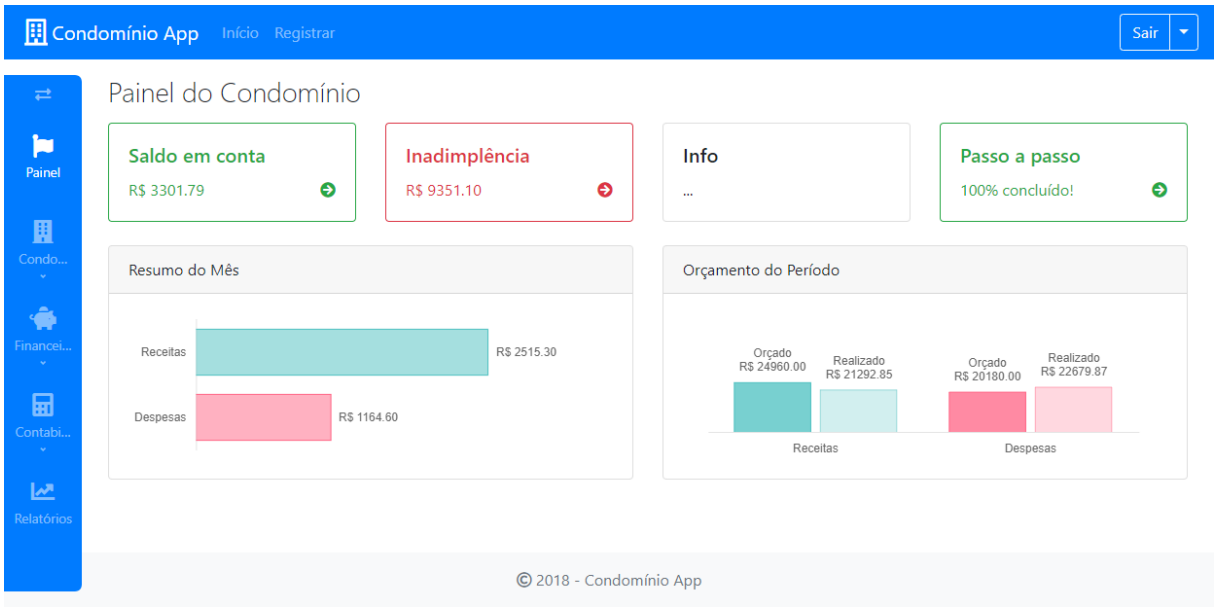


Figura 6 – Tela do painel de gestão do condomínio

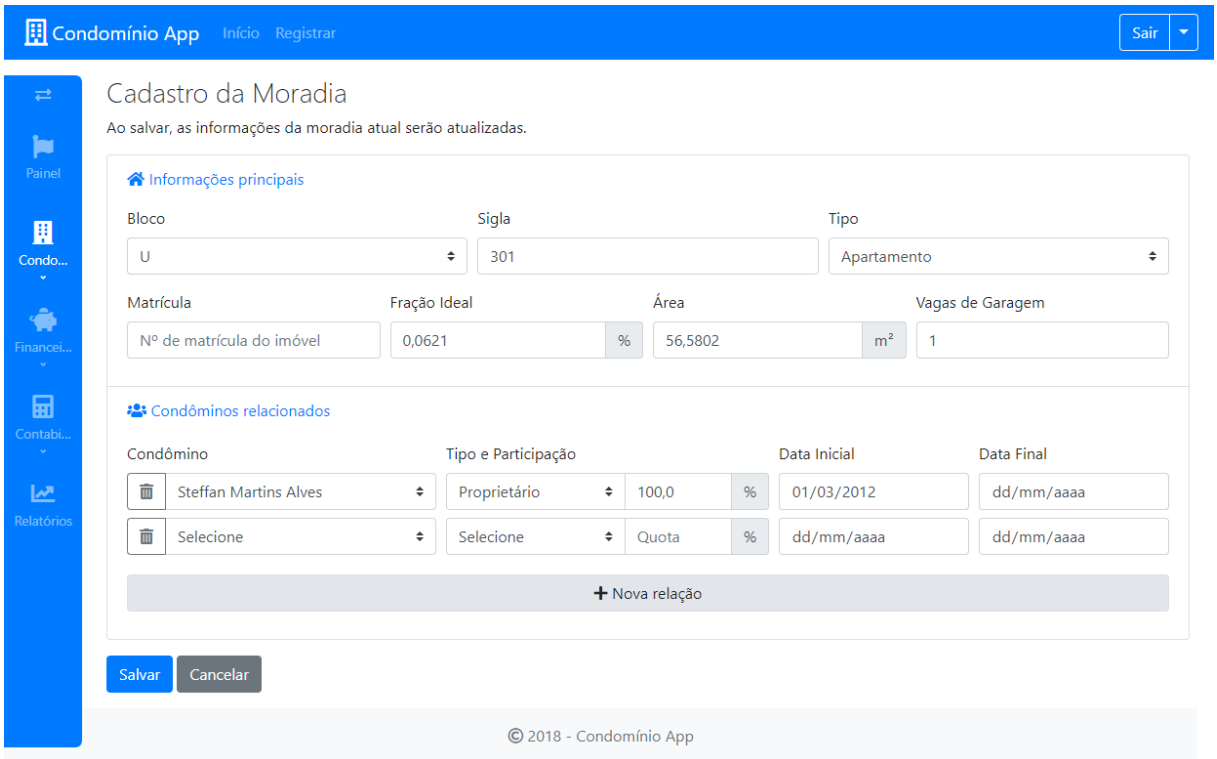


Figura 7 – Tela de cadastro de moradia

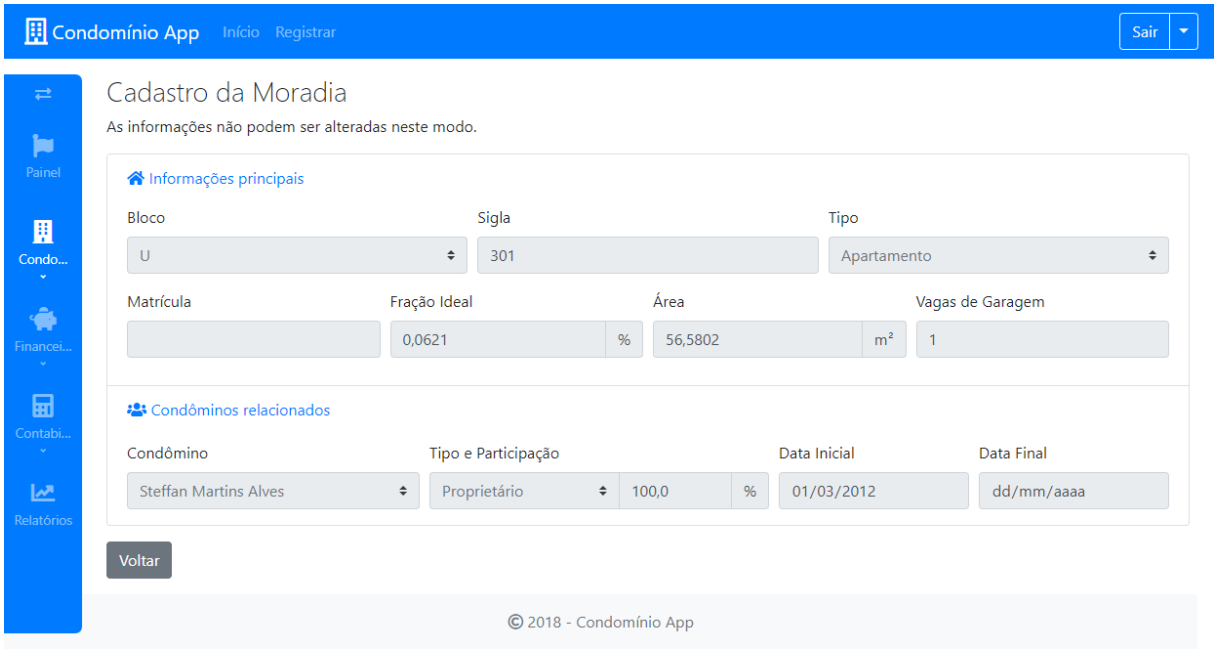


Figura 8 – Tela de visualização de moradia

Todas as funcionalidades do sistema podem ser acessadas através do menu de navegação lateral e estão agrupadas em cinco categorias que se expandem para revelar mais opções: Painel, Condomínio, Financeiro, Contabilidade e Relatórios.

Em seu primeiro acesso o síndico precisa cadastrar as informações do condomínio que irá gerir, tal como nome, CNPJ, endereço, dentre outros, para somente depois começar a registrar os blocos e suas moradias, bem como condôminos e demais entidades, sendo alertado caso tente avançar esta etapa.

Realizado este primeiro cadastro, o síndico pode seguir uma ordem mais livre. É permitido, por exemplo, cadastrar os blocos e as moradias, e então depois cadastrar os condôminos, ou até mesmo cadastrar primeiramente estes, deixando aqueles para um segundo momento. Além disso, como moradias e condôminos se relacionam, durante o cadastro de qualquer um dos dois o síndico já pode especificar isto com mais detalhes.

A Figura 7 mostra a etapa de inclusão de uma moradia, onde esta função é utilizada para já relacioná-la a um condômino previamente cadastrado. Não há um limite de relações, e toda relação criada em uma moradia também será exibida na tela dos condôminos em questão, e vice-versa.

Cabe destacar que apenas uma página foi criada para cada entidade, e esta única página se adapta de acordo com a ação requisitada. A página de visualização de uma entidade (Figura 8), por exemplo, é levemente diferente da respectiva página de inclusão/edição, pois alguns elementos são escondidos (como dicas e botões de ação) e todos os campos tem sua edição desabilitada, garantido que alterações não sejam feitas na entidade.

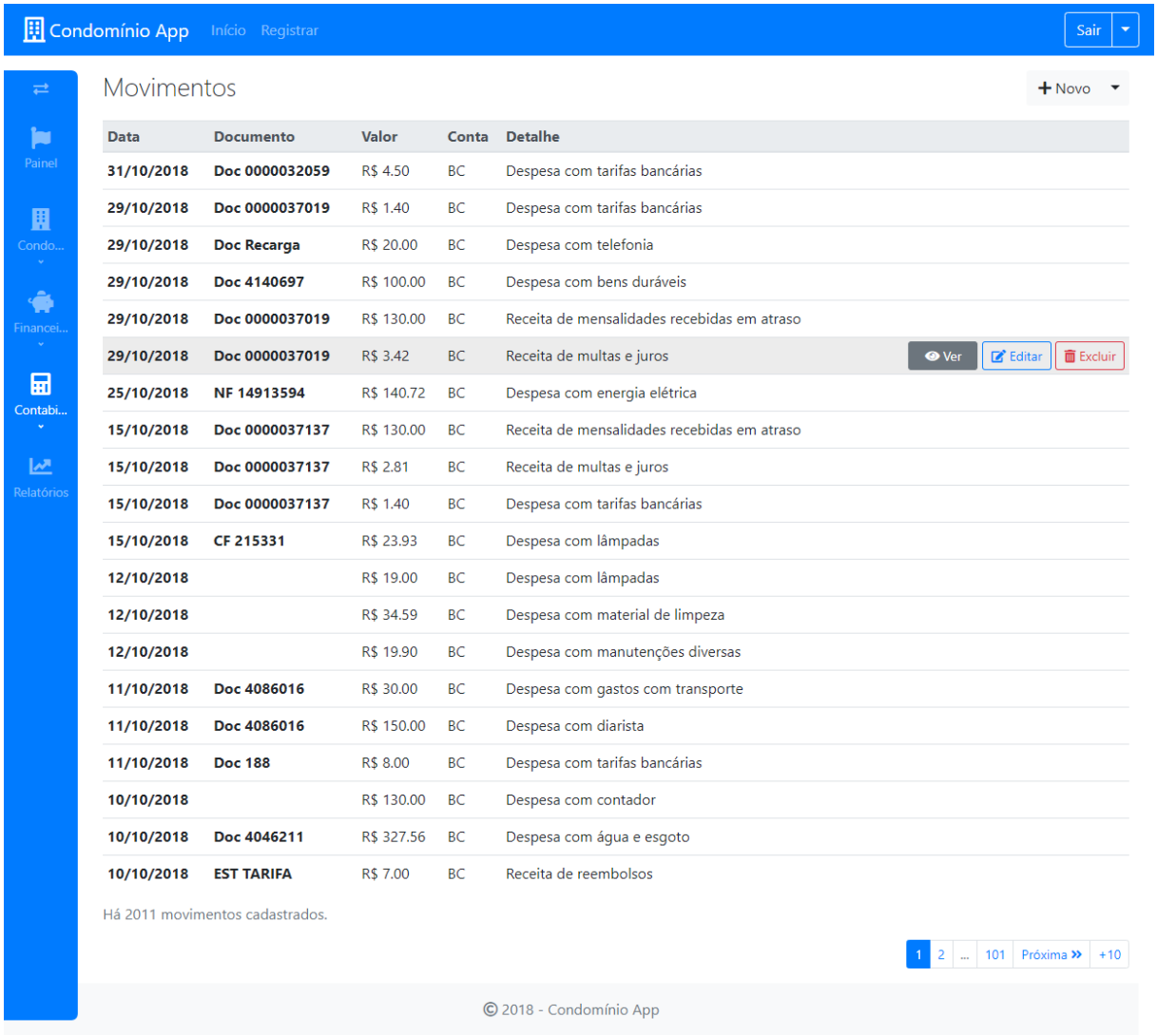


Figura 9 – Tela de listagem de movimentos

Todos os CRUDs do sistema tem exatamente o mesmo funcionamento: ao acessar a funcionalidade, abre-se uma tela de listagem. Esta tela terá os registros já criados e um botão para inclusão de um novo. Além disso, ao destacar um registro, o sistema exibe as opções de ver, editar ou excluir tal linha, como é mostrado na Figura 9, que traz a lista de movimentos como exemplo. Estas listas possuem um sistema de paginação e somente os registros da página atual são recuperados do DBMS, assim o aplicativo responde ao usuário com maior rapidez e evita leituras desnecessárias na base de dados.

Em alguns campos ou entidades o usuário encontrará um ícone de ajuda, onde é possível ler uma explicação detalhada da funcionalidade em uso para auxiliá-lo em dúvidas, como no caso de registros de Categorias (Figura 10), onde é explicado ao síndico como funciona a estrutura que o sistema usa de uma forma simples e exemplificada. Há também ajuda nos campos de CEP e de código de instituições bancárias, direcionando o usuário a páginas de pesquisas das agências que os mantém.

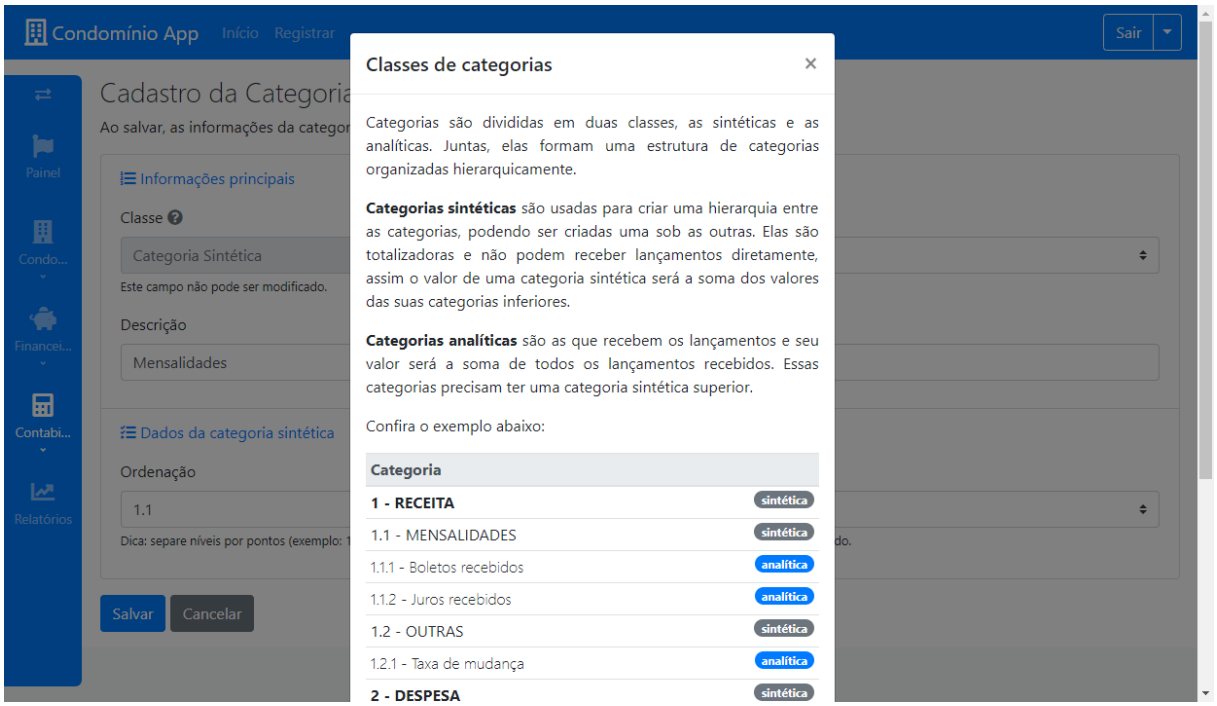


Figura 10 – Modal explicativo de classes de categoria

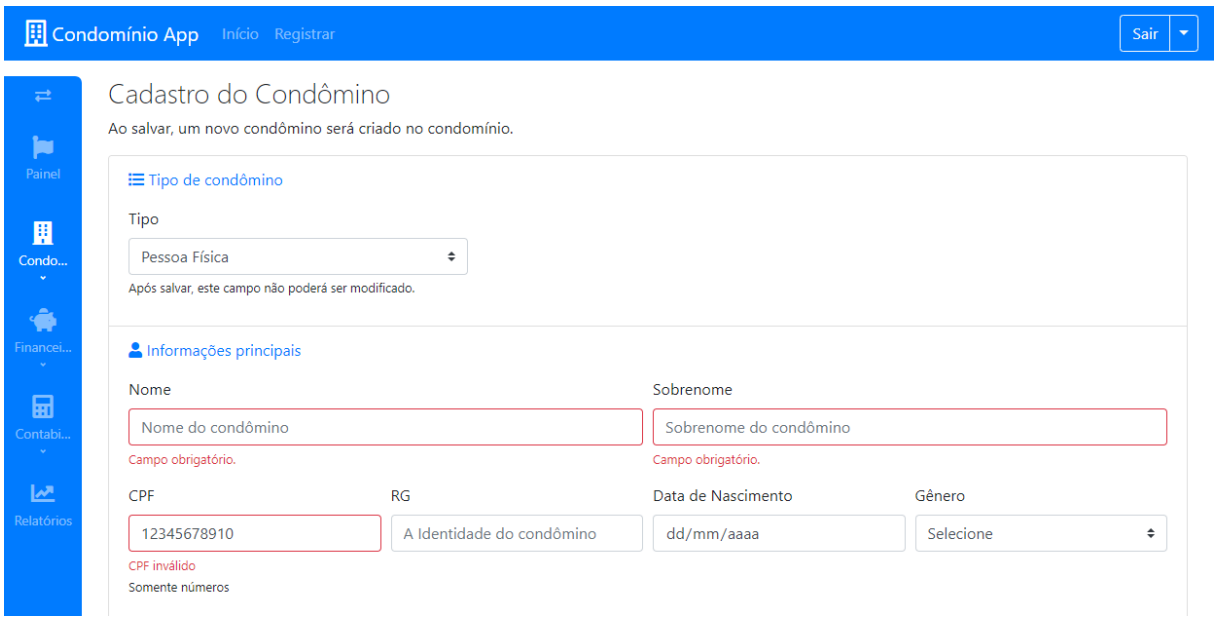


Figura 11 – Tela de cadastro de condômino com mensagens de validação

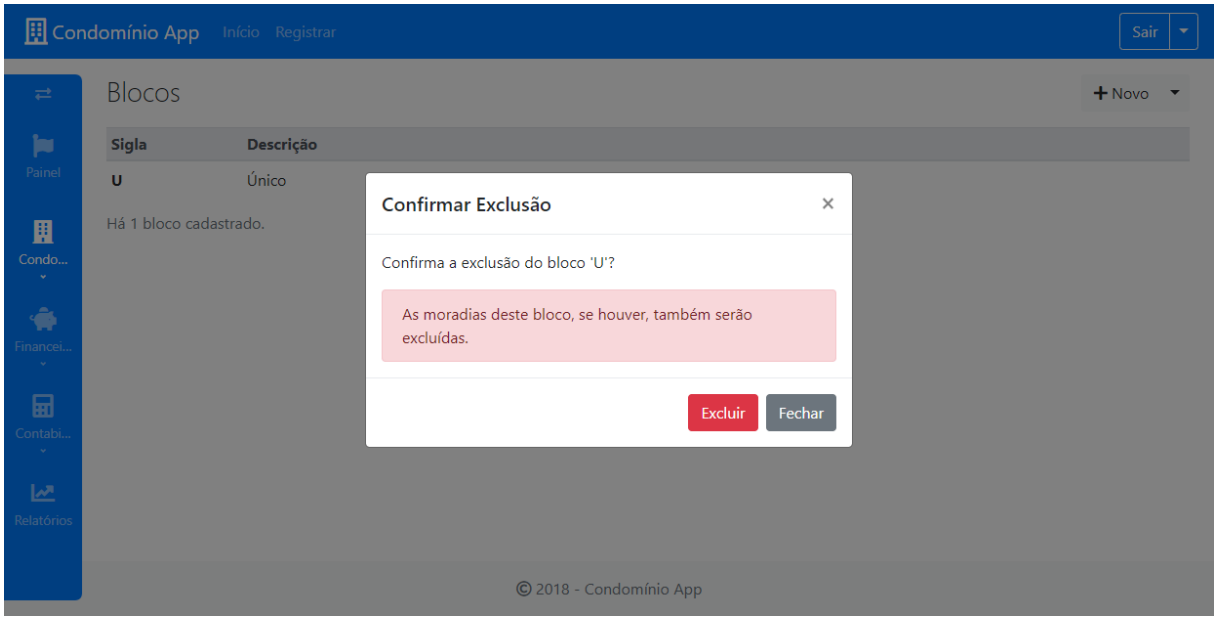


Figura 12 – Modal de confirmação para exclusão de um bloco

Toda entrada do usuário no sistema é feita através de submissão de formulários, portanto, vulnerável a erros de digitação e omissões. Desta forma, requisições recebidas com campos obrigatórios em branco ou com campos inválidos não são processadas, e um *feedback* é passado ao usuário com orientação sobre o equívoco (Figura 11). Esta validação é feita sempre pelo servidor para evitar que uma falha no lado do cliente afete a integridade do aplicativo, e também para facilitar validações mais profundas, como as que consultam se um registro já existe no banco de dados. Dentre as validações realizadas nos formulários se destacam: validação de CPF, CNPJ, *e-mail*, unicidade de registros, valores monetários, porcentagens, campos obrigatórios ou derivados.

Conforme o RNF10, para excluir um registro qualquer sempre há necessidade de confirmação do usuário, oportunidade em que informações adicionais sobre a entidade são reveladas, como a possibilidade de perda de registros filhos (Figura 12). A exclusão dos registros relacionados é garantida em dois níveis: no DBMS todas as chaves estrangeiras foram criadas com a regra para exclusão em cascata; e no *back-end* da aplicação todas as relações entre as entidades mantidas pelo Hibernate foram registradas com a mesma regra. A precedência é pela exclusão através da aplicação, porém o DBMS garante que o registro sem referência seja apagado se esta falhar.

A usabilidade da aplicação também recebe bastante atenção. Todas as páginas foram cuidadosamente desenhadas para facilitar a compreensão do sistema pelo seu utilizador e proporcionar uma experiência mais agradável. Com este objetivo, campos relacionados foram sempre agrupados em seções, e cada um destes campos possui dicas de preenchimento ou formatação, bem como ícones, prefixos ou sufixos que indicam unidades de medida ou características particulares.

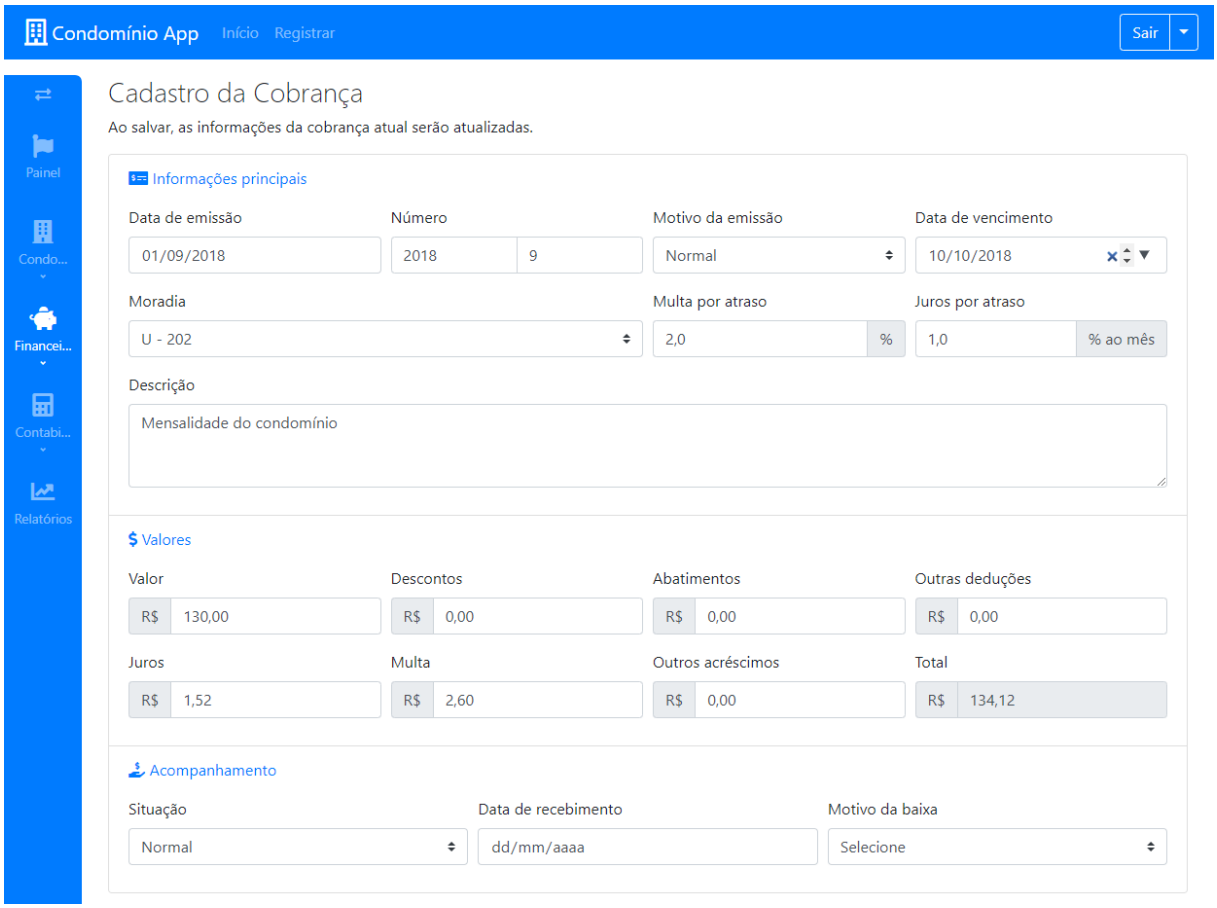


Figura 13 – Tela de cadastro de cobrança

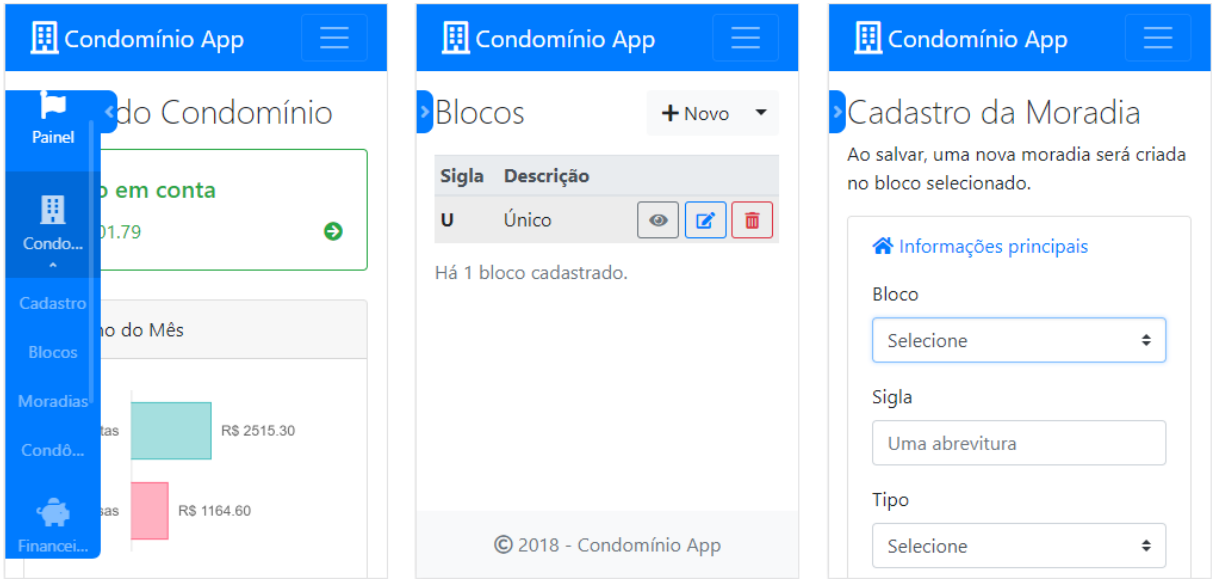
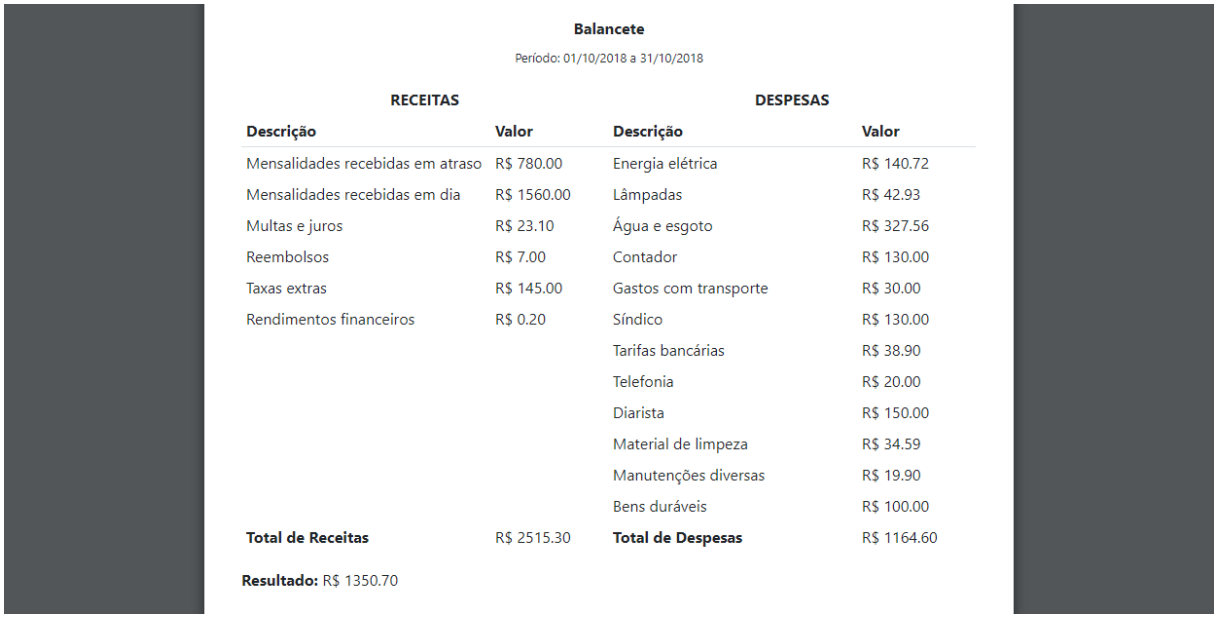


Figura 14 – Telas do sistema na exibição de um *smartphone*



Balancete

Período: 01/10/2018 a 31/10/2018

RECEITAS		DESPESAS	
Descrição	**Valor**	**Descrição**	**Valor**
Mensalidades recebidas em atraso	R$ 780.00	Energia elétrica	R$ 140.72
Mensalidades recebidas em dia	R$ 1560.00	Lâmpadas	R$ 42.93
Multas e juros	R$ 23.10	Água e esgoto	R$ 327.56
Reembolsos	R$ 7.00	Contador	R$ 130.00
Taxas extras	R$ 145.00	Gastos com transporte	R$ 30.00
Rendimentos financeiros	R$ 0.20	Síndico	R$ 130.00
		Tarifas bancárias	R$ 38.90
		Telefonia	R$ 20.00
		Diarista	R$ 150.00
		Material de limpeza	R$ 34.59
		Manutenções diversas	R$ 19.90
		Bens duráveis	R$ 100.00
Total de Receitas	R$ 2515.30	**Total de Despesas**	R$ 1164.60

Resultado: R$ 1350.70

Figura 15 – Relatório Balancete visualizado no navegador

A Figura 13 mostra como este conceito é bem aplicado no cadastro de uma cobrança, onde há, também, automação do valor total e preenchimento padrão de alguns campos, outra facilidade para o usuário. Parece simples, mas se o usuário precisasse se preocupar em computar o total da cobrança ou com os campos zerados ele perderia tempo fazendo cálculos e poderia, ainda, errar a informação. Desta forma, à medida que ele insere alguns valores já pode acompanhar o total, que é preenchido em tempo real, e no servidor os campos vazios recebem seus valores padrões.

O *site* também é responsivo. Com um *smartphone* de apenas 3,5¨ de visor, por exemplo, o usuário pode acessá-lo e executar qualquer uma das operações que faz no computador, pois nenhuma funcionalidade deixou de ser adaptada à tela menor (Figura 14). Em dispositivos assim, o menu é totalmente recolhido para dar espaço ao conteúdo. Além disso, alguns botões são redesenhados e os campos são reorganizados para melhor apresentar as informações ou facilitar a interação com a tela sensível ao toque.

Por fim, estão presentes os requisitos de recuperação de senha com envio de e-mail assíncrono contendo um *token* único e temporário, senhas estas que são gravadas no DBMS somente após cifradas por um algoritmo de segurança, há possibilidade de se manter conectado e também de alteração dos dados pessoais do síndico, atualização diária de cobranças vencidas através de um evento recorrente programado no DBMS MySQL, capacidade de acessos simultâneos em qualquer horário e menos espera por respostas graças a um *pool* de conexões com o banco de dados controlado pelo **HikariCP** (WOOLDRIDGE, 2012), além de integridade garantida por transações.

O sindico irá utilizar rotineiramente o sistema para inserir os movimentos ocorridos no condomínio, tais como pagamentos de contas e recebimentos de mensalidades,

relacionando-os a contas e categorias específicas, transferências entre contas, emissão e baixas de cobranças, etc. A qualquer momento, havendo informação inserida no sistema, o síndico pode visualizar, salvar ou imprimir diversos relatórios, sendo estes o resultado final do trabalho e de todo o esforço envolvido neste projeto.

O síndico informará períodos ou intervalos de data para abrir os relatórios 'Livro Caixa', 'Balancete' e 'Orçamento'. Os relatórios podem abranger quaisquer datas, mas o comum é usar períodos anuais para orçamentos e mensais para os demais. Além destes, também é possível gerar uma relação atualizada de inadimplentes. O Apêndice A traz um exemplo de cada um destes relatórios, e a Figura 15 mostra como ele é exibido no navegador do usuário, já adequado à impressão e corretamente formatado.

4.2 Validação

O sistema foi submetido a inúmeros testes durante e após o desenvolvimento. Testes funcionais, de performance e de volume, por exemplo, foram feitos para identificar e corrigir falhas no desenvolvimento. Após considerada finalizada, a aplicação passou por um teste de aceitação por usuários, da seguinte maneira:

- **Escolha de voluntários:** nesta fase foram convidados o síndico e um condômino do Condomínio do Edifício Residencial Palmeiras II, no bairro Jardim Holanda, em Uberlândia (MG), que se dispuseram a fazer parte do processo.
- **Teste funcional:** todas as funcionalidades do sistema foram testadas pelo síndico, desde os cadastros iniciais aos registros dos movimentos e impressão de relatórios. Os relatórios foram comparados pelo síndico e pelo condômino com os que o condomínio utiliza atualmente.
- ***Feedback*:** os voluntários foram ouvidos e expuseram suas impressões sobre a aplicação.

O síndico aprovou o funcionamento de todo o sistema, alegando não ter encontrado dificuldade ou falhas durante a utilização. Apenas ressaltou que o sistema poderia permitir incluir vários orçamentos de uma só vez, para agilizar um pouco mais esta tarefa, e importar planilhas para registrar as informações dos períodos anteriores que já estão documentadas, sem ter que digitá-las outra vez para ter um histórico completo e tirar proveito total dos relatórios do sistema. Para fins de testes, foi realizada a importação das planilhas cedidas através de comandos SQL, e com todos os registros o síndico percebeu que também seria importante haver uma forma de filtrar as listas para facilitar a localização de determinados registros.

Os relatórios, que também foram analisados pelo condômino, se mostraram fiéis àqueles entregues pelo contador do condomínio. Os valores estavam idênticos, validando

assim os cálculos e conceitos adotados, e a disposição das informações também se mostrou familiar. Destacou-se o relatório de inadimplência, que traz todos os detalhes das cobranças já com valores atualizados, diferente do utilizado pelo condomínio, que não se preocupa com cálculos de juros e multas.

No geral, os voluntários que participaram do teste se mostraram extremamente satisfeitos com o sistema, e as melhorias sugeridas pelo síndico são colocadas como trabalhos futuros.

Capítulo 5

Conclusão

Para este trabalho foi idealizado e implementado um Sistema de Gestão para Condomínios com o objetivo de facilitar o trabalho do síndico como um gestor, livrando-o de tarefas manuais oferecendo suporte aos processos de forma simples, com qualidade e eficiência.

Este objetivo só foi atingido graças aos estudos realizados durante o trabalho, como a realização de cursos sobre os *frameworks* utilizados, unidos aos conhecimentos prévios absorvidos no decorrer do curso de graduação em Sistemas de Informação, incluindo aqueles relacionados às disciplinas de Programação Orientada a Objetos, Banco de Dados, Estrutura de Dados, Redes de Computadores, Modelagem de Software e Programação para Internet. Também teve igual contribuição o conhecimento prévio em Contabilidade, graças a uma graduação anterior e o exercício da profissão, além da experiência pessoal como síndico de um condomínio há alguns anos.

Com o objetivo de validar a aplicação, um condomínio real foi convidado para utilizar o sistema. Nesta atividade os resultados foram satisfatórios, além de contribuir para a identificação de algumas novas funcionalidades, objeto de trabalhos futuros.

Pretende-se continuar o desenvolvimento da aplicação, adicionando novas funcionalidades, melhorando outras e posteriormente planejar sua comercialização com o intuito de buscar um valor justo para o desenvolvedor e, ao mesmo tempo, mínimo para o condomínio, contribuindo para o desenvolvimento saudável do setor e a qualidade de vida das pessoas afetadas direta ou indiretamente pela gestão do síndico.

5.1 Desafios Encontrados

Durante o desenvolvimento do projeto algumas dificuldades foram enfrentadas. Modelar o banco de dados até que ficasse consistente e adequado levou certo tempo. Depois, devido à falta de experiência, programar todos os relacionamentos entre as entidades no Hibernate foi complexo e exigiu muito esforço, em especial na relação do tipo “muitos para muitos” e com atributos que há entre moradias e condôminos. Este caso não é di-

retamente suportado pelo *framework* e precisou de implementação. Este mesmo tipo de relação causou dificuldades no desenvolvimento *front-end*. Por falta de suporte do Thymeleaf as telas de cadastro de moradias e condôminos precisaram de especial atenção, sendo necessário a criação manual de um complexo algoritmo em JS para manipulação dos elementos.

A montagem dos relatórios também exigiu esforço extra. Houve necessidade de elaboração de consultas mais complexas em SQL e uso de diversas estruturas de dados para apresentar as informações da forma desejada. Além disso, o Thymeleaf não dá suporte a cálculos com variáveis para, por exemplo, computar a soma de uma lista de lançamentos. Na maioria dos casos este cálculo precisou ser feito no *back-end* e disponibilizado junto à lista de resultados para poder ser exibido ao usuário pelo Template Engine.

Por fim, a implantação da aplicação em ambiente de produção exigiu a observância de algumas especificidades do fornecedor de PaaS, sendo necessário tornar o projeto compatível com diversas variáveis de ambiente.

Todos estes obstáculos foram oportunidades para estudo e incremento no aprendizado. Cada superação contribuiu não só para o desenvolvimento de uma aplicação robusta como também para meu crescimento profissional, pois a aquisição de tais experiências ajudarão a enfrentar novos desafios.

5.2 Trabalhos Futuros

O projeto, neste momento, atende às obrigações legais de forma a amparar o síndico e melhorar sua capacidade de gestão, mas melhorias em algumas funcionalidades e adição de novas ferramentas são possíveis, pois a área é muito abrangente e há uma gama de oportunidades a explorar. Os itens a seguir são uma pequena lista de requisitos que podem ser implementados ou melhorados.

- **Mais relatórios:** os relatórios atuais visam atender ao mínimo exigido pela legislação, mas o sistema já guarda e pode vir a reter informações que permitem a criação de inúmeros relatórios de gestão úteis ao síndico, como extrato de conta, razão de categoria, ficha cadastral de condômino, relatório de adimplência, etc. É ainda possível enriquecer os relatórios com gráficos e algumas opções de personalização.

- **Filtros:** dar ao usuário a possibilidade de filtrar as entidades é uma forma de aumentar a produtividade e ao mesmo tempo melhorar a satisfação com o sistema. Assim, um filtro capaz de pesquisar por qualquer atributo do objeto é um bom adicional ao sistema.

- **Importação:** alguns condomínios já possuem planilhas de controle, e seria uma grande vantagem oferecer aos síndicos uma forma de importá-las para que todo

o histórico de movimentos do condomínio não fique em um ambiente separado, o que além de enriquecer os relatórios é um fator motivador para a adesão ao sistema. Como não há uma padronização seria necessário trabalhar com modelos bem definidos, e caberia ao síndico o ajuste de suas planilhas a tais modelos antes da importação.

- **Cobranças em lote:** em certo ponto do mês o síndico cria cobranças para cada uma das moradias, e estas geralmente possuem o mesmo valor e vencimento. Para evitar a repetição da tarefa o sistema poderia permitir a criação destas cobranças para todas as moradias de uma só vez.

- **Integração entre módulos:** a exemplo da integração entre moradia e condômino, onde alteração em um registro impacta no outro, alguns outros módulos também podem ser integrados. Um caso é o módulo de cobrança e o de movimentos, onde um lançamento poderia ser automaticamente criado quando uma cobrança for marcada como recebida, mantendo-se uma relação entre eles para o caso de estornos ou alterações.

- **Emissão de boletos:** uma funcionalidade muito almejada pelos síndicos é a emissão de boletos. Da forma atual, as cobranças são um registro gerencial da dívida, e os condomínios que utilizam boletos bancários precisam emiti-los através do *software* do próprio banco. Trazer uma integração da funcionalidade de cobranças com a emissão de boletos bancários no padrão definido pelo Centro Nacional de Informação Bancária (CNAB) seria um diferencial.

- **Novos atores:** o sistema pode ser expandido para suportar, ao menos, dois novos atores: o condômino e a administradora de condomínios. O condômino teria interesse em acessar a prestação de contas do condomínio a qualquer momento e ver a situação das cobranças emitidas contra sua moradia. A administradora de condomínios é uma prestadora de serviços que realiza o papel de síndico para os inúmeros condomínios que a contratam e estaria interessada em utilizar o sistema de forma plural, podendo gerenciar mais de um condomínio.

- **Assembleias:** as atas das assembleias do condomínio poderiam ser armazenadas no sistema, incluindo também o controle de presença da reunião.

- **Áreas de lazer:** a maioria dos condomínios possui uma ou mais áreas de lazer. Como são de uso comum, há necessidade de gerenciar reservas e até mesmo cobrar taxas de uso. Uma funcionalidade que traga uma agenda para cada área de lazer e opções de reserva e cobrança, apesar de não crítica, é muito interessante.

- **Portaria:** condomínios de médio a grande porte costumam ter porteiros e fazem registro de visitantes, veículos e encomendas. Este controle pode ser implementado no

sistema, deixando toda informação do condomínio em um único local para facilitar a gestão do síndico.

- **Fundos de reserva:** condomínios precisam manter fundos de reserva para os momentos de crise ou para obras e melhorias. Idealmente estes fundos de reserva têm uma conta exclusiva, mas nem sempre é assim que acontece na prática. Alguns síndicos deixam todo o valor da reserva misturado ao valor corrente, o que dificulta o seu controle. Este é um problema a ser entendido e solucionado na intenção de facilitar o controle do fundo de reserva sem obrigar a adequação do gestor ao método ideal.

- **Comunicados:** o sistema também pode ser uma plataforma de comunicação entre o síndico e os condôminos, permitindo envio de mensagens de ambos os lados. O síndico poderia enviar a prestação de contas, as cobranças e também avisos gerais aos condôminos, e estes poderiam registrar queixas e demandas ao síndico. Este canal de comunicação pode vir a ser uma área no *site*, *e-mail* ou até mesmo mensagens de celular.

- **Funcionários:** alguns condomínios tem funcionários e precisam calcular, dentre outros, a folha de pagamento, férias, décimo terceiro, horas extras e rescisões de contrato, bem como entregar uma série de declarações ao governo, como o envio obrigatório de informações ao Sistema de Escrituração Fiscal Digital das Obrigações Fiscais Previdenciárias e Trabalhistas (eSocial) a partir de 2019 (ESOCIAL, 2018). Este seria o maior desafio de desenvolvimento neste cenário e sua disponibilização influenciaria o preço final do produto devido à alta complexidade dos registros trabalhistas, necessidade de observação de leiautes específicos e integração com certificados digitais, necessitando ser oferecido separadamente aos clientes que se interessem para não onerar condomínios menores.

- **Portal de notícias:** a área pública do sistema desenvolvido possibilita apenas que o síndico faça seu cadastro e efetue *log-in*. Utilizar a área pública para trazer notícias relacionadas ao setor de condomínios e promover a aplicação é uma ideia a ser analisada. Esta área seria gerenciada pela equipe de vendas e consultoria do sistema, e não pelos síndicos.

Referências

ALTERDATA. **Software para Gestão de Condomínio - Immobile**. 1989. <https://www.alterdata.com.br/immobile/condominio>. (Accessado em 20/06/2018).

_____. **Cloud**. 2018. <https://www.alterdata.com.br/cloud>. (Accessado em 21/06/2018).

APACHE. **Welcome to The Apache Software Foundation!** 2018. <https://www.apache.org/>. (Accessado em 15/11/2018).

BARISH, G. **Building scalable and high-performance Java Web applications using J2EE technology**. pub-AW:adr: Addison-Wesley, 2002. xviii + 392 p. ISBN 0-201-72956-3.

ELMASRI, R.; NAVATHE, S. **Sistemas de banco de dados**. [S.l.]: PEARSON BRASIL, 2011. ISBN 9788579360855.

ESOCIAL, C. D. D. **RESOLUÇÃO DO COMITÊ DIRETIVO DO ESOCIAL Nº 5, DE 2 DE OUTUBRO DE 2018 - eSocial**. 2018. <http://portal.esocial.gov.br/institucional/legislacao/resolucao-do-comite-diretivo-do-esocial-no-5-de-2-de-outubro-de-2018>. (Acessado em 18/11/2018).

ETEMAD, E.; JR., T. A.; RIVOAL, F. **CSS Snapshot 2017**. [S.l.], 2017. Https://www.w3.org/TR/2017/NOTE-css-2017-20170131/. (Accessado em 21/06/2018).

FONTICONS. **Font Awesome**. 2018. <https://fontawesome.com/>. (Accessado em 15/11/2018).

ICONDEV. **SIN - Sistema de Gestão de Condomínios**. 2017. <http://www.sistemacondominioonline.com.br/index.html>. (Accessado em 20/06/2018).

JAWSDB. **JawsDB Database-as-a-Service**. 2018. <https://www.jawsdb.com/>. (Accessado em 15/11/2018).

JOHNSON, R. **Expert One-on-One J2EE Design and Development**. Birmingham, UK, UK: Wrox Press Ltd., 2002. ISBN 0764543857, 9780764543852.

OTTO, M. **Bootstrap, the most popular HTML, CSS, and JS library in the world.** 2018. <https://getbootstrap.com/>. (Accessado em 15/11/2018).

PIVOTAL. **Spring Boot**. 2018. <https://spring.io/projects/spring-boot>. (Accessado em 15/11/2018).

_____. **Spring Data**. 2018. <https://spring.io/projects/spring-data>. (Accessado em 15/11/2018).

_____. **Spring Data JPA**. 2018. <https://spring.io/projects/spring-data-jpa>. (Accessado em 15/11/2018).

_____. **Spring Framework**. 2018. <https://spring.io/projects/spring-framework>. (Accessado em 15/11/2018).

_____. **Spring Security**. 2018. <https://spring.io/projects/spring-security>. (Accessado em 15/11/2018).

_____. **Spring Web MVC**. 2018. <https://docs.spring.io/spring/docs/current/spring-framework-reference/web.html>. (Accessado em 15/11/2018).

PROVOS, N.; MAZIERES, D. **USENIX Annual Technical Conference**. 1999. <https://www.usenix.org/legacy/event/usenix99/provos/provos_html/index.html>. (Accessado em 17/11/2018).

REDHAT. **Hibernate ORM, your relational data. Objectively.** 2018. <http://hibernate.org/orm/>. (Accessado em 15/11/2018).

REPÚBLICA, P. D. **Lei Nº 4.591, de 16 de Dezembro de 1964**. 1964. <http://www.planalto.gov.br/ccivil_03/leis/l4591.htm>. (Accessado em 06/05/2018).

_____. **Lei Nº 6.434, de 15 de Julho de 1977**. 1977. <http://www.planalto.gov.br/ccivil_03/leis/L6434.htm>. (Accessado em 06/05/2018).

_____. **Lei Nº 10.406, de 10 de Janeiro de 2002**. 2002. <http://www.planalto.gov.br/ccivil_03/leis/2002/l10406.htm>. (Accessado em 06/05/2018).

ROUSSOPOULOS, N.; DELIS, A. Modern client–server dbms architectures. **Sigmod**, ACM, v. 20, n. 3, p. 52–61, set. 1991.

RUMBAUGH, J.; JACOBSON, I.; BOOCH, G. **The Unified Modeling Language Reference Manual**. [S.l.]: Addison-Wesley, 2005. (The Addison-Wesley object technology series, v. 1). ISBN 9780321245625.

SALESFORCE. **Cloud Application Platform Heroku**. 2018. <https://www.heroku.com/>. (Accessado em 15/11/2018).

SMITH, M. **HTML: The Markup Language (an HTML language reference)**. [S.l.], 2013. Http://www.w3.org/TR/2013/NOTE-html-markup-20130528/. (Accessado em 21/06/2018).

THYMELEAF. **Thymeleaf**. 2018. <https://www.thymeleaf.org/>. (Accessado em 15/11/2018).

TIMBERG, E. **Chart.js, open source HTML5 Charts for your website**. 2018. <https://www.chartjs.org/>. (Accessado em 15/11/2018).

WIKIPéDIA. **Flat design**. 2018. <https://pt.wikipedia.org/wiki/Flat_design>. (Accessado em 20/06/2018).

WOOLDRIDGE, B. **HikariCP, a solid, high-performance, JDBC connection pool at last.** 2012. <https://github.com/brettwooldridge/HikariCP>. (Accessado em 17/11/2018).

Apêndices

APÊNDICE **A**

Exemplos de Relatórios

Este apêndice contém exemplos dos relatórios gerados no Sistema de Gestão de Condomínios desenvolvido neste trabalho.

A.1 Livro Caixa

Condomínio X

Rua Z, 999

Bairro ABC, Uberlândia - Minas Gerais

Síndico: Steffan Martins Alves

Livro Caixa

Período: 01/10/2018 a 31/10/2018

Saldo inicial: R$ 1951.09

Data	Descrição	Documento	Entrada	Saída	Saldo
01/10/2018	Tarifa bancária - boletos	Doc 0000035015		R$ 4.20	R$ 1946.89
01/10/2018	Rec. Atrasados - Aptos 104 e 304	Doc 0000035015	R$ 390.00		R$ 2336.89
01/10/2018	Rec. Juros - Aptos 104 e 304	Doc 0000035015	R$ 11.85		R$ 2348.74
02/10/2018	Rec. condomínio ref. Setembro/18 - Apto 302	Doc 0000037144	R$ 130.00		R$ 2478.74
02/10/2018	Tarifa bancária - boletos	Doc 0000037144		R$ 1.40	R$ 2477.34
04/10/2018	Rec. condomínio ref. Setembro/18 - Apto 204	Doc 0000037080	R$ 130.00		R$ 2607.34
04/10/2018	Tarifa bancária - boletos	Doc 0000037080		R$ 1.40	R$ 2605.94
05/10/2018	Rec. condomínio ref. Setembro/18 - Aptos 103 e 201	Doc 0000035119	R$ 260.00		R$ 2865.94
05/10/2018	Tarifa bancária - boletos	Doc 0000035119		R$ 4.20	R$ 2861.74
05/10/2018	Outros créditos	COBR 0,20	R$ 0.20		R$ 2861.94
05/10/2018	Rec. Atrasados - Apto 404	Doc 0000035119	R$ 130.00		R$ 2991.94
05/10/2018	Rec. Juros - Apto 404	Doc 0000035119	R$ 5.02		R$ 2996.96
08/10/2018	Rec. condomínio ref. Setembro/18 - Aptos 102, 203 402	Doc 0000037033	R$ 390.00		R$ 3386.96
08/10/2018	Rec. Atrasados - Apto 301	Doc 0000037033	R$ 145.00		R$ 3531.96

A.2 Balancete

Condomínio X

Rua Z, 999

Bairro ABC, Uberlândia - Minas Gerais

Síndico: Steffan Martins Alves

Balancete

Período: 01/10/2018 a 31/10/2018

RECEITAS		DESPESAS	
Descrição	**Valor**	**Descrição**	**Valor**
Mensalidades recebidas em atraso	R$ 780.00	Energia elétrica	R$ 140.72
Mensalidades recebidas em dia	R$ 1560.00	Lâmpadas	R$ 42.93
Multas e juros	R$ 23.10	Água e esgoto	R$ 327.56
Reembolsos	R$ 7.00	Contador	R$ 130.00
Taxas extras	R$ 145.00	Gastos com transporte	R$ 30.00
Rendimentos financeiros	R$ 0.20	Síndico	R$ 130.00
		Tarifas bancárias	R$ 38.90
		Telefonia	R$ 20.00
		Diarista	R$ 150.00
		Material de limpeza	R$ 34.59
		Manutenções diversas	R$ 19.90
		Bens duráveis	R$ 100.00
Total de Receitas	R$ 2515.30	**Total de Despesas**	R$ 1164.60

Resultado: R$ 1350.70

A.3 Orçamento

Condomínio X

Rua Z, 999

Bairro ABC, Uberlândia - Minas Gerais

Síndico: Steffan Martins Alves

Orçamento

Período: 01/01/2018 a 31/12/2018

Categoria	Orçado	Realizado	Atingido
1 - RECEITAS	**R$ 24960.00**	**R$ 21292.85**	**85.31 %**
1.1 - MENSALIDADES	R$ 24960.00	R$ 18956.37	75.95 %
1.1.1 - Mensalidades recebidas em atraso	R$ 0.00	R$ 3790.00	Não orçado
1.1.2 - Mensalidades recebidas em dia	R$ 24960.00	R$ 15080.00	60.42 %
1.1.3 - Multas e juros	R$ 0.00	R$ 86.37	Não orçado
1.2 - PERIÓDICAS	R$ 0.00	R$ 1472.00	NÃO ORÇADO
1.2.3 - Reembolsos	R$ 0.00	R$ 7.00	Não orçado
1.2.4 - Taxas extras	R$ 0.00	R$ 1395.00	Não orçado
1.2.5 - Vendas de usados	R$ 0.00	R$ 70.00	Não orçado
1.3 - FINANCEIRAS	R$ 0.00	R$ 864.48	NÃO ORÇADO
1.3.2 - Empréstimos	R$ 0.00	R$ 833.39	Não orçado
1.3.3 - Rendimentos financeiros	R$ 0.00	R$ 31.09	Não orçado
2 - DESPESAS	**R$ 20180.00**	**R$ 22679.87**	**112.39 %**
2.1 - CONSUMO	R$ 5460.00	R$ 5201.99	95.27 %
2.1.1 - Água e esgoto	R$ 3840.00	R$ 3269.49	85.14 %
2.1.2 - Coleta de lixo	R$ 0.00	R$ 464.20	Não orçado
2.1.3 - Energia elétrica	R$ 1320.00	R$ 1324.98	100.38 %
2.1.4 - Lâmpadas	R$ 300.00	R$ 143.32	47.77 %
2.2 - ADMINISTRATIVAS	R$ 5400.00	R$ 4501.30	83.36 %
2.2.2 - Contador	R$ 1560.00	R$ 1300.00	83.33 %
2.2.3 - Gastos com transporte	R$ 150.00	R$ 387.00	258.00 %
2.2.4 - Material de escritório	R$ 240.00	R$ 20.40	8.50 %
2.2.5 - Síndico	R$ 2760.00	R$ 2300.00	83.33 %
2.2.6 - Tarifas bancárias	R$ 540.00	R$ 372.90	69.06 %
2.2.7 - Telefonia	R$ 150.00	R$ 121.00	80.67 %
2.3 - LIMPEZA E CONSERVAÇÃO	R$ 7620.00	R$ 5941.25	77.97 %
2.3.1 - Controle de pragas	R$ 0.00	R$ 115.90	Não orçado
2.3.2 - Diarista	R$ 3320.00	R$ 2960.00	89.16 %
2.3.3 - Jardinagem	R$ 1200.00	R$ 300.00	25.00 %
2.3.4 - Limpeza de caixas d'água	R$ 400.00	R$ 0.00	0.00 %
2.3.5 - Limpeza de caixas de esgoto	R$ 1500.00	R$ 1070.00	71.33 %
2.3.6 - Limpeza de telhados e calhas	R$ 600.00	R$ 1150.00	191.67 %
2.3.7 - Material de limpeza	R$ 600.00	R$ 345.35	57.56 %
2.4 - MANUTENÇÕES	R$ 640.00	R$ 895.60	139.94 %
2.4.2 - Manutenção de cercas elétricas	R$ 0.00	R$ 150.00	Não orçado
2.4.3 - Manutenção de extintores	R$ 340.00	R$ 326.70	96.09 %
2.4.4 - Manutenção de portões	R$ 300.00	R$ 300.00	100.00 %
2.4.5 - Manutenções diversas	R$ 0.00	R$ 118.90	Não orçado

A.4 Inadimplência

Condomínio X

Rua Z, 999

Bairro ABC, Uberlândia - Minas Gerais

Síndico: Steffan Martins Alves

Inadimplência

Cobranças vencidas até: 17/11/2018

Moradia: U - 202

Cobrança	Vencimento	Valor	Multa	Juros	Outros	Total atual
2018 - 9	10/10/2018	R$ 130.00	R$ 2.60	R$ 1.60	R$ 0.00	R$ 134.20
					Subtotal	R$ 134.20

Moradia: U - 302

Cobrança	Vencimento	Valor	Multa	Juros	Outros	Total atual
2013 - 2	10/03/2013	R$ 50.00	R$ 1.00	R$ 34.62	R$ -30.00	R$ 55.62
CERCA	10/03/2013	R$ 50.00	R$ 0.00	R$ 0.00	R$ 0.00	R$ 50.00
2015 - 2	10/03/2015	R$ 100.00	R$ 2.00	R$ 44.90	R$ 0.00	R$ 146.90
2015 - 10	10/11/2015	R$ 100.00	R$ 2.00	R$ 36.73	R$ 0.00	R$ 138.73
2017 - 5	10/06/2017	R$ 100.00	R$ 2.00	R$ 17.47	R$ 0.00	R$ 119.47
OBRA - 1	10/07/2017	R$ 145.00	R$ 0.00	R$ 0.00	R$ 0.00	R$ 145.00
2017 - 6	10/07/2017	R$ 100.00	R$ 2.00	R$ 16.47	R$ 0.00	R$ 118.47
OBRA - 2	10/08/2017	R$ 145.00	R$ 0.00	R$ 0.00	R$ 0.00	R$ 145.00
OBRA - 3	10/09/2017	R$ 145.00	R$ 0.00	R$ 0.00	R$ 0.00	R$ 145.00
2017 - 8	10/09/2017	R$ 100.00	R$ 2.00	R$ 14.40	R$ 0.00	R$ 116.40
2017 - 9	10/10/2017	R$ 100.00	R$ 2.00	R$ 13.40	R$ 0.00	R$ 115.40
2017 - 10	10/11/2017	R$ 100.00	R$ 2.00	R$ 12.37	R$ 0.00	R$ 114.37
2017 - 11	10/12/2017	R$ 100.00	R$ 2.00	R$ 11.37	R$ 0.00	R$ 113.37
2017 - 12	10/01/2018	R$ 130.00	R$ 2.60	R$ 13.43	R$ 0.00	R$ 146.03
2018 - 1	10/02/2018	R$ 130.00	R$ 2.60	R$ 12.09	R$ 0.00	R$ 144.69
2018 - 2	10/03/2018	R$ 130.00	R$ 2.60	R$ 10.88	R$ 0.00	R$ 143.48
2018 - 3	10/04/2018	R$ 130.00	R$ 2.60	R$ 9.53	R$ 0.00	R$ 142.13
2018 - 4	10/05/2018	R$ 130.00	R$ 2.60	R$ 8.23	R$ 0.00	R$ 140.83
2018 - 5	10/06/2018	R$ 130.00	R$ 2.60	R$ 6.89	R$ 0.00	R$ 139.49
2018 - 6	10/07/2018	R$ 130.00	R$ 2.60	R$ 5.59	R$ 0.00	R$ 138.19
2018 - 8	10/09/2018	R$ 130.00	R$ 2.60	R$ 2.90	R$ 0.00	R$ 135.50
					Subtotal	R$ 2654.07

Moradia: U - 402

Cobrança	Vencimento	Valor	Multa	Juros	Outros	Total atual
2012 - 6	10/07/2012	R$ 40.00	R$ 0.00	R$ 0.00	R$ 0.00	R$ 40.00
2012 - 7	10/08/2012	R$ 40.00	R$ 0.00	R$ 0.00	R$ 0.00	R$ 40.00
2012 - 8	10/09/2012	R$ 40.00	R$ 0.00	R$ 0.00	R$ 0.00	R$ 40.00
2012 - 9	10/10/2012	R$ 40.00	R$ 0.00	R$ 0.00	R$ 0.00	R$ 40.00
2012 - 10	10/11/2012	R$ 70.00	R$ 1.40	R$ 51.26	R$ 0.00	R$ 122.66
2014 - 10	10/11/2014	R$ 100.00	R$ 2.00	R$ 48.90	R$ 0.00	R$ 150.90
2014 - 11	10/12/2014	R$ 100.00	R$ 2.00	R$ 47.90	R$ 0.00	R$ 149.90
2015 - 3	10/04/2015	R$ 100.00	R$ 2.00	R$ 43.87	R$ 0.00	R$ 145.87