UNIVERSIDADE DO ESTADO DE SANTA CATARINA - UDESC
CENTRO DE CIÊNCIAS TECNOLÓGICAS

LAVÍNIA RAFAELA DE MARCO
MATHEUS AMORIM DA CRUZ

PROJETO FINAL DE ENGENHARIA DE SOFTWARE: SISTEMA DE ADMINISTRAÇÃO DE CONDOMÍNIO

JOINVILLE
2024

SUMÁRIO

1. Descrição do Problema

A administração de condomínios possui vários problemas operacionais que podem comprometer a eficiência e a satisfação dos moradores. A gestão manual de informações sobre moradores, unidades de apartamentos, cobranças financeiras, reservas de áreas comuns e agendamento de reuniões pode demandar mais tempo, se tornar trabalhosa e sujeita a erros. Além disso, a comunicação eficiente entre a administração do condomínio e os moradores é importante para manter um ambiente harmonioso e bem organizado. Assim, o sistema de administração de condomínios proposto, visa realizar a gestão financeira e de moradores, além de controlar as reservas de áreas comuns aos condôminos, dentro do espaço de convivência.

1.1 Escopo e Funcionalidades

O sistema de administração de condomínios fornecerá uma solução que atenda as atividades envolvidas na gestão de administração de um condomínio. Para isso, o software abrangerá as principais funcionalidades a seguir:

a) Gestão de Moradores:
   Cadastro e remoção de informações dos moradores;
   Cadastro e controle de unidades de apartamento;
   Comunicação direta ao morador por mensagens ou avisos.

b) Reserva de Áreas Comuns:
   Agendamento de espaços comuns (como salão de festas, quadras e churrasqueiras);
   Controle de disponibilidade dos espaços.

c) Reuniões:
   Agendamento e controle de assembleias e registro de pautas.

   Para a melhor visualização, foi elaborado o escopo do projeto presente na Figura 1.

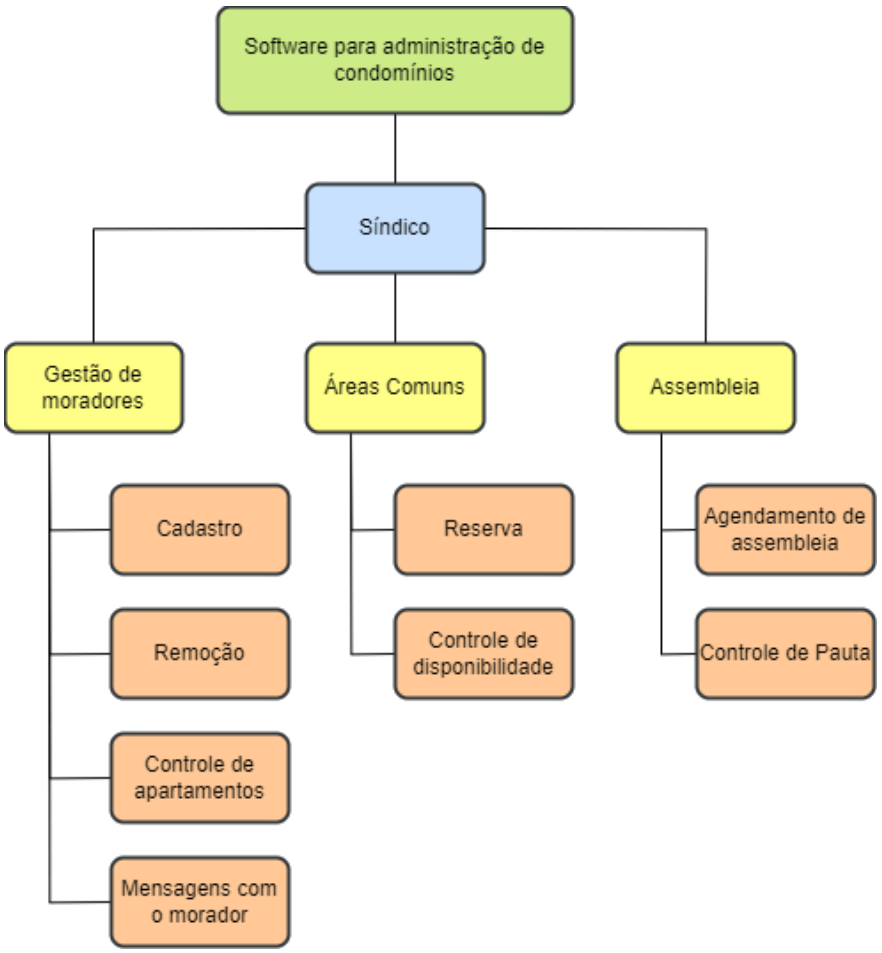

Figura 1: diagrama contendo o escopo do sistema de administração de condomínio.

1.2 *Stackeholders*

As pessoas que fazem parte deste projeto, são as seguintes:

1. Síndico: Responsável pela administração do condomínio e usuário chave do sistema.
2. Moradores: Utilizam sistema para comunicação e reserva de áreas comuns e pagamento de taxas.
3. Funcionários do condomínio. Precisam de acesso para registro de ocorrências e acompanhamento de pedidos de manutenção.
4. Desenvolvedores de Software: Equipe responsável pelo desenvolvimento, manutenção e atualização do sistema.

2. Requisitos

Requisitos são as especificações das necessidades e expectativas dos stakeholders que o sistema deve atender. Eles são fundamentais para a

construção de um sistema que atenda às necessidades dos usuários e das partes interessadas (VALENTE, 2019). Desse modo, nas sessões 2.1 e 2.2 são apresentados os requisitos funcionais e não funcionais, respectivamente.

2.1 Requisitos Funcionais

Os requisitos funcionais especificam o comportamento e as funções que o sistema deve realizar, bem como as interações entre os usuários e o sistema. Segundo Valente (2019), requisitos funcionais são essencialmente declarações das funcionalidades ou serviços que o sistema deve oferecer. Para tanto, os requisitos funcionais determinados, foram:

RF1: Sistema deve permitir o cadastro e exclusão de moradores;

RF2: Sistema deve permitir vinculação de moradores aos apartamentos;

RF3: Sistema deve permitir envio de mensagens e avisos ao email dos moradores;

RF6: Sistema deve permitir que moradores visualizem a disponibilidade das áreas comuns e permitir possíveis reservas;

RF8: Sistema deve agendar assembleias;

RF9: Sistema deve registrar pautas e atas da assembleia;

2.2 Requisitos não Funcionais

Os requisitos não funcionais especificam os critérios de qualidade que o sistema deve atender. De acordo com Valente (2019), os requisitos não funcionais são restrições sobre os serviços ou funções oferecidos pelo sistema. Nesse contexto, com base nos requisitos funcionais, foram elaborados os seguintes requisitos não funcionais:

Para RF1:

RNF1: Sistema deve solicitar credenciais de acesso ao sistema;

RFN2: Sistema deve controlar permissões com base nos perfis de determinados usuários.

Para RF2:

RNF3: Apenas usuários autorizados podem vincular um morador a um apartamento;

Para RF8 e RF9:

RNF4: Sistema deve notificar todos participantes corretamente sobre o agendamento;

RNF5: Sistema deve garantir que apenas usuários autorizados possam registram pautas e atas.

RNF6: Sistema deve permitir adição de novas funcionalidades sem grande modificações na arquitetura existente (escalabilidade);

3. Estimativa de Duração do Projeto

Para este trabalho, será utilizado o método de COCOMO para estimar o tempo necessário para realizar todas as etapas do projeto. Para isso, será classificado os elementos do software, determinado o seu nível de complexidade, definir o peso para cada elemento conforme sua complexidade, encontrar os Pontos de Função Não Ajustados (PFNA) e convertê-los em linhas de código (LOC), e por fim, calcular o tempo de desenvolvimento do projeto como um todo conforme o COCOMO.

3.1 Elementos do Software

Os elementos do software podem ser classificados como: Entradas Externas (EE), Saídas Externas (SE), Consultas Externas (CE), Arquivos Lógicos Internos (ALI) e Arquivos Lógicos Externos (ALE). Desse modo, as funcionalidades foram classificadas da seguinte forma:

EE:

1. Cadastro de Moradores
2. Cadastro de Unidades de Apartamento
3. Envio de Mensagens/Avisos
4. Agendamento de Áreas Comuns
5. Agendamento de Assembleias.

SE:

1. Relatórios de Uso das Áreas Comuns;
2. Relatório de Moradores;
3. Relatório de Funcionários.

CE:

1. Consulta de Moradores;
2. Consulta de Unidades;
3. Consulta de Disponibilidade das Áreas Comuns;
4. Consulta de Assembleias.

ALI:

1. Dados de Moradores;
2. Dados de Reservas;
3. Dados de Assembleias.

ALE:

1. Integração com Serviços de Email.

3.2 Nível de Complexidade e Definição de Pesos

Os pesos são atribuídos conforme o nível de complexidade de cada elemento. Em geral, os pesos são classificados em complexidade baixa, média e alta. Cada categoria possui um peso específico, conforme a Tabela 1.

Tipo do Elemento	**Complexidade**		
	Baixa	**Média**	**Alta**
EE	3	4	6
SE	4	5	7
CE	3	4	6
ALI	7	10	15
ALE	5	7	10

Tabela 1: pesos para cada tipo de elementos e suas respectivas complexidades.

Considerando que todos os parâmetros de medição possuem complexidade média, calcula-se (Tabela 2):

Tipo do Elemento	**Quantidade encontrada**	**Fator de complexidade média**			**Total**
EE	5	x	4	=	20
SE	3	x	5	=	15
CE	4	x	4	=	16
ALI	3	x	10	=	30
ALE	1	x	7	=	7

Tabela 2: cálculo do elemento por seu fator de complexidade média.

3.3 Pontos de Função Não Ajustados (PFNA)

A partir da definição de pesos e o cálculo para cada elemento, pode-se calcular os PFNAs somando o total de cada função vezes o seu peso, ou

seja, PFNA = Σ(elemento x peso). Portanto, para PFNA = 20 + 15 + 16 + 30 + 7, então PFNA é igual a 88.

3.4 Conversão de PFNA em LOC

A conversão de PFNA para Linhas de Código (LOC) pode variar dependendo da linguagem de programação utilizada. No caso do desenvolvimento deste projeto, foi Java. Para essa linguagem, um valor comum utilizado é que 1 Ponto de Função (PF) equivale aproximadamente a 53 Linhas de Código (LOC), de acordo com dados do International Function Point Users Group (IFPUG). Assim, pode-se calcular: LOC = PFNA x 53, logo, o número de LOC é igual a 4.664, ou 4,664 KLOC.

3.5 COCOMO

Considerando que este projeto seja simples, é possível utilizar a fórmula básica do COCOMO para calcular o esforço dado em pessoa-mês: Esforço = 2.4 x (KLOC) ^ 1.05. Dessa forma, substituindo o valor de KLOC encontrado anteriormente, de forma arredondada, se obtêm que o **esforço é igual a 12 pessoas-mês**.

Já para o cálculo da duração do projeto utilizando COCOMO, pode-se obter através da fórmula: Duração = 2.5 x (Esforço) ^ 0,38, dado em meses. Assim, substituindo o valor do esforço, se chega ao resultado aproximado de que a **duração é igual a 6,45 meses** (6 meses e 13 dias e meio).

4. Diagrama de Classes do Projeto UML

Para melhor visualização do sistema de administração de condomínios, foi criado um diagrama de classes UML (Figura 2), contendo todas as classes, seus atributos e métodos. Posteriormente, foi implementado um programa com base nessa imagem, que possibilitou a realização dos testes unitários (JUnit). Além disso, por se tratar de uma imagem com muitos elementos, o arquivo desta foi anexado à pasta do projeto, para que se possa aproximar e analisá-la adequadamente.

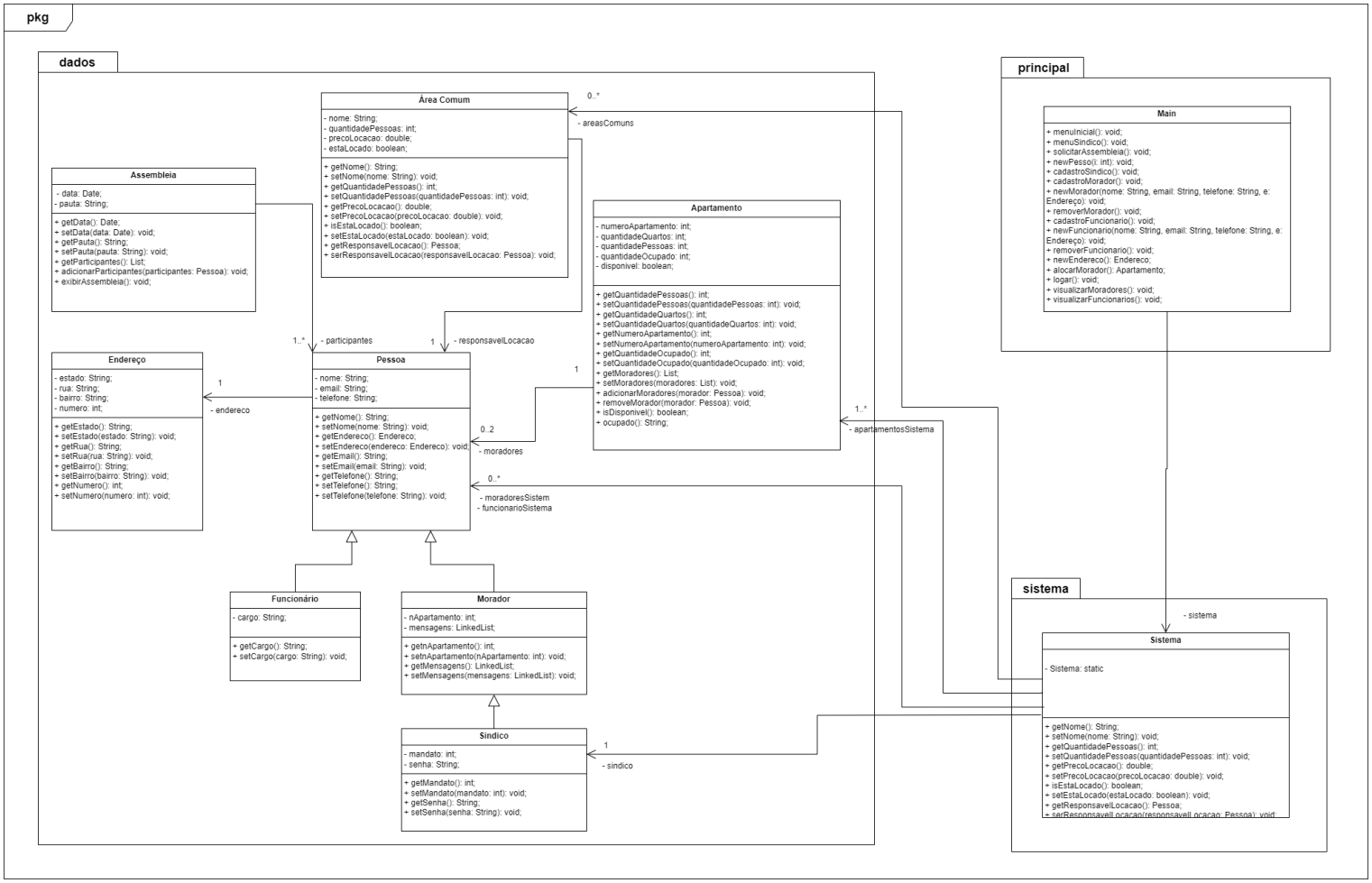

Figura 2: diagrama de classes UML do sistema de administração de condomínios.

5. Testes Unitários

Tanto o código do sistema de administração de condomínios quanto os seus testes unitários estão na pasta “ProjetoFinalSoft” e no link do github:

6. Referências

International Function Point Users Group (IFPUG). Disponível em: https://www.ifpug.org/. Acesso em: 01 jul. 2024.

VALENTE, Marco Tulio. **Engenharia de Software Moderna: Princípios, Conceitos e Métodos**. 2019. Disponível em: https://engsoftmoderna.info/. Acesso em: 01 jul. 2024.