UNIVERSIDADE PAULISTA - UNIP

SISTEMA DE INFORMAÇÃO PARA GERIR UM PORTFÓLIO DE IMÓVEIS DE PEQUENO PORTE

SÃO PAULO

2019

FICHA CATALOGRÁFICA

Carvalho, Lucas; De Jesus, Luiza; Damasceno, Eric; Silva, Daniel;
SISTEMA DE INFORMAÇÃO PARA GERIR UM PORTFÓLIO SIMPLES DE PEQUENO PORTE/ L. Carvalho, Jundiaí – 2019.

Projeto Integrado Multidisciplinar II – Universidade Paulista UNIP

RESUMO

Este trabalho tem o objetivo de desenvolver e implementar um sistema de apoio à decisão que auxilie o proprietário de um pequeno portfólio de imóveis a otimizar a gestão do seu patrimônio. Tendo isto em vista, os principais problemas que o sistema visa solucionar são os da falta de organização no registro de eventos relacionados aos imóveis, de métodos para controle da rentabilidade por imóvel e de ferramentas para o auxílio do planejamento anual.

Mediante a utilização de método adequado, dados foram levantados para diagnosticar e detalhar tais problemas, requisitos funcionais e não funcionais foram especificados e métricas financeiras foram estabelecidas com o intuito de viabilizar a implementação do sistema em plataforma C++.

O sistema resultante permite realizar o controle do portfólio de imóveis, além da geração automatizada de um de um banco de dados para controle do gestor.

Em relação a atividade de controle, a gestora pode lançar eventos relacionados ao fluxo de aluguel dos imóveis, os quais são automaticamente consolidados e corrigidos por imóvel.

Já em relação ao relatório de banco, este se divide em duas seções: Controle do ano que passou e planejamento do seguinte. Para ambas, os cenários de venda versus manutenção do imóvel são comparados com o intuito de se apurar o resultado geral do ano na seção de controle e de se calcular um aluguel a ser cobrado para atingir *o break-even (*ponto de equilíbrio) no ano seguinte levando em conta as projeções da gestora de valorização patrimonial por imóvel na seção de planejamento.

Palavras-chave: Sistema de apoio à decisão, portfólio de imóveis, gestão de imóveis.

LISTA DE FIGURAS

SUMÁRIO

1 - INTRODUÇÃO

O trabalho a seguir tem, como objetivo, além de melhor absorver o conteúdo das disciplinas o de desenvolver e implementar um sistema de gestão de portfólio, que auxilie o proprietário de uma imobiliária a otimizar a gestão de seu patrimônio. A partir desse ponto, os problemas que o sistema pretende solucionar são os da falta de informação de banco de dados de clientes, organização nos registros e cadastros dos imóveis, implementar filtros de localização, número de quartos, modelo de habitação e comodidades.

O sistema permitirá criar um banco de dados de clientes já consolidados em contrato, como também de clientes em potencial a partir da busca realizada através do website, objetivando a captação através do departamento comercial da imobiliária.

Em relação as atividades de controle, a gestora poderá controlar o fluxo dos processos na ficha cadastral desses eventuais inquilinos, permitindo assim inserir dados dos fiadores, informações pessoais como RG, CPF, declarante ou não de imposto de renda; estado civil e etc.

A impressão desses dados será de grande importância para que os gestores, e o departamento jurídico consiga aprovar os inquilinos no serviço de proteção ao crédito, e confecção do contrato de locação do imóvel para as devidas partes

1.2 - MERCADO IMOBILIÁRIO ATUAL NO BRASIL

É de amplo conhecimento que o Brasil atravessou recentemente um período de recessão econômica. Neste sentido, segundo dados do IBGE (Instituto brasileiro de geografia e estatística), o PIB (Produto interno bruto) do país caiu 3,5% em 2015 e 3,5% em 2016, o que originou uma situação em que muitas pessoas se viram obrigadas a se desfazer de ativos imobilizados para sair de situações financeiramente incômodas.

No entanto, ao mesmo tempo que a oferta de imóveis no mercado aumentou, a demanda pelos mesmos diminuiu devido à instabilidade econômica e a falta de oferta de crédito de financiamento no mercado - como a figura 1 expõe - típicos de períodos de recessão. Tal situação ocasionou um cenário de queda de preços real dos imóveis e esfriamento no mercado imobiliário.

Figura 1 - Variação nos financiamentos de imóveis no Brasil Fonte: Elaborado pelo autor

Seguindo, ainda conforme dados do IBGE, o PIB brasileiro voltou a crescer cerca de 1% em 2017. No entanto, mesmo após este leve sinal de melhoria, não se pode esperar que o mercado imobiliário volte a se aquecer rapidamente uma vez que como a compra de imóveis

requer investimentos relativamente altos por parte do comprador, seu nível de confiança na economia e de estabilidade financeira têm que estar altos por um período considerável.

Considerando esta inércia, é relevante notar que em períodos como o atual, o valor de mercado dos imóveis oscila significantemente e um acompanhamento destes torna-se imprescindível para que um investidor tenha seu capital otimizado até para aproveitar a imensa gama de oportunidades típicas de recessões.

Além disto, torna-se ainda mais importante que a rentabilidade do dinheiro investido em imóveis seja comparada com a fornecida por outros instrumentos simples como títulos do tesouro ou aplicações em fundos de bancos privados com o intuito de se otimizar o retorno sobre o capital investido levando em conta o sempre relevante custo de oportunidade.

1.3 DEFINIÇÃO DO OBJETIVO DESTE TRABALHO

Tendo em vista as necessidades e desejos atuais do gestor do portfólio de imóveis, o objetivo deste trabalho é o desenvolvimento e implementação de um sistema de apoio à decisão que forneça suporte e apoio para a gestor do pequeno portfólio de imóveis gerir de forma otimizada o patrimônio. Tal sistema deve fornecer uma forma consolidada e organizada de controlar as entradas e saídas de cada imóvel do portfólio e ferramentas de controle de resultados e planejamento com o intuito de potencializar a já bem-sucedida gestão do portfólio.

Para determinar este fim, utilizaremos como base o portfólio de imóveis de cadastro e os respectivos problemas e oportunidades de melhoria presentes na forma pela qual o gestor controla entradas e saídas e toma decisões no tocante a planejamento atualmente.

Aqui, vale frisar que julgamos em conjunto que a elaboração de um sistema complexo que aborde oscilações de valor de negociação dos seus e outros imóveis e mapeie oportunidades de realização de transações não apresentaria custo-benefício interessante considerando que a gestor já possui conhecimento profundo deste mercado e contatos com diversos profissionais e imobiliárias que fornecem toda ajuda que ela precisa com esses tópicos.

Tendo o escopo do objetivo deste trabalho sido delineado, é necessário começar a pensar em concretamente viabilizá-lo. Neste ponto, surgem opções como terceirizar esta gestão para instituições financeiras especializadas no assunto. Contudo, as altas taxas cobradas e a vontade de gerir o patrimônio com autonomia e maior fluidez fazem com que alguns proprietários relutem em aceitar tal encaminhamento, como no caso do contexto desse trabalho.

1.3 ESTRUTURA DESTE TRABALHO

O roteiro de elaboração deste trabalho consiste primeiramente na pesquisa e revisão bibliográfica de materiais, acervos, leis, publicações e artigos coerentes com o planejamento, implementação e conteúdo do sistema a ser desenvolvido e implementado. Segue-se a essa análise, a aplicação dos conhecimentos revisados para a estruturação do problema, elaboração do projeto do sistema, implementação do sistema e discussão de seus resultados até o momento de elaboração deste trabalho.

2 REVISÃO DE LITERATURA

2.1 SISTEMAS DE INFORMAÇÃO

2.1.1 DEFINIÇÃO DE SISTEMA

Antes de se definir o que é um sistema de informação, deve-se primeiramente definir o que é um sistema. O termo “sistema” de acordo com Ferreira (2010), define-se como um conjunto de princípios verdadeiros ou falsos reunidos de modo que formem um corpo de doutrina. Neste tom, de acordo com Churchman (2015), apesar das diversas definições de sistema, é consensual que um sistema pode ser compreendido como um conjunto de partes interligadas com alguma finalidade.

2.1.2 DEFINIÇÃO DE SISTEMAS DE INFORMAÇÃO

Segundo Reynolds e Stair (2010), um sistema de informação pode ser definido como um conjunto de componentes inter-relacionados que fornecem um mecanismo de resposta por meio da coleta, manipulação, armazenagem e disseminação de dados e informações. Neste ponto, torna-se relevante salientar que segundo o mesmo autor, dados podem ser compreendidos como fatos inicialmente irrelevantes que quando ordenados de maneira coerente compõem informações, as quais, por sua vez, são capazes de prover valor maior do que os mesmos isoladamente.

Já segundo Laudon, K.J. (2004), um sistema de informação consiste em elementos inter-relacionados que realizam atividades de coleta, processamento, armazenagem e distribuição de informações com a finalidade oferecer base para o processo de tomada de decisões e todas as etapas que o sucede.

2.1.3. PRINCIPAIS ATIVIDADES DE UM SISTEMA DE INFORMAÇÃO

Para Laudon, K.J. (2004), as principais atividades de um sistema de informação entrada, processamento e saída sendo que elas são capazes de gerar conclusões requisitadas pelas organizações para os processos de tomada de decisão.
Ainda conforme o autor, enquanto a entrada realiza a coleta de dados internos ou externos à organização, o processamento converte estes dados em informações e finalmente, a saída aloca tais informações para as pessoas ou atividades intermediárias ou finais. Vale frisar também a participação dos *feedbacks*, os quais auxiliam os membros da organização a avaliar e controlar o processo do sistema dentro de um contexto mais geral

2.2 PROJETO DO SISTEMA

2.2.1 ESTRUTURAÇÃO DO PROBLEMA

Segundo notas de aula da disciplina 288S – Linguagem e Técnicas de Programação ministrado pelo Professor César Tegani Tofanini (2019), administrar pode ser definido como interpretar e obter os objetivos propostos para um sistema por meio de um processo administrativo adequado, destacando que este processo administrativo consiste em PIDC (Planejar, implementar, dirigir e controlar) os recursos do sistema. Neste ponto, ainda segundo o mesmo autor torna-se necessário detalhar o que cada uma destas etapas significa:

Planejar: Decidir antecipadamente o que fazer, quanto fazer, quando fazer, como fazer, com que fazer, que fazer, onde fazer. E, tudo isso, após considerar o para que (porquê) fazer;
Implementar: Fornecer recursos de forma concreta, constituindo o duplo organismo material e social da empresa;
3. Dirigir: Designar pessoas, coordenar esforços, comunicar, motivar, liderar, orientar;

4. Controlar: Comparar os resultados obtidos aos planejados e atuar corretivamente no sistema.

2.2.2 ANÁLISE DE NEGÓCIO

Segundo Paula Filho (2003), a fase de análise de negócio consiste na fase inicial no desenvolvimento de um sistema de informação e inclui três itens importantes: Descrição geral da organização, descrição de objetivos, metas e fatores críticos de sucesso e modelagem do processo de negócio envolvido no escopo de trabalho.

Ainda conforme o autor, existe a necessidade de se explicitar os resultados qualitativos esperados, quantifica-los e descrever os fatores críticos necessários para o sucesso na obtenção destes resultados para que o desenvolvimento do sistema esteja em linha com as necessidades da organização.

2.2.3 REQUISITOS

Segundo Paula filho (2003), para a etapa de planejamento dos requisitos do sistema, pode-se seguir o método para a especificação dos mesmos: Definição de escopo, requisitos funcionais e requisitos não-funcionais.

Neste sentido, o escopo abrange os problemas que o sistema almeja solucionar. Além disto, ele deve definir a missão, ou valor agregado pelo sistema à organização, os limites, ou funcionalidades não apresentadas pelo sistema e, por fim, os benefícios trazidos por ele. (Paula Filho, 2003)

Para tal fim, vale ressaltar que o sistema desenvolvido precisa atender a necessidades do seu usuário sempre visando solucionar seus problemas e dificuldades – tais necessidades são denominadas de requisitos. (Paula Filho, 2003)

Aqui, é relevante ressaltar que os requisitos podem ser divididos entre funcionais e não-funcionais.

Os requisitos funcionais descrevem casos de uso, que por sua vez, caracterizam-se como funções do sistema que beneficiam diretamente o usuário. Cada requisito deve ser consolidado em um diagrama de caso de uso, o qual além de detalha-lo, evidencia como os seus atores se relacionam com o mesmo. (Paula Filho, 2003)

Neste sentido, vale frisar que os casos de uso modelam funções completas do sistema e obrigatoriamente devem prover um benefício ao usuário. Desta forma, quando agrupados,

todos os casos de uso representam o sistema por completo, sem qualquer lacuna ou superposição. Além disto, no contexto da elaboração dos casos de uso, os atores devem ser definidos. Eles servem para modelar atividades a serem exercidas pelo usuário, podendo ser seres humanos ou não, e também para detalhar os requisitos determinados. (Paula Filho, 2003)

Após tal definição, finalmente é hora de elaborar diagramas de casos de uso, nos quais as relações entre os atores definidos e os casos de usos especificados devem ser modeladas. Tais diagramas detalham que atores estão envolvidos em cada caso de uso, e quem começa a interação em cada caso por meio de setas. (Paula Filho, 2003)

Segue-se a elaboração de todos estes casos de uso, a hora de elaborar o diagrama de casos de uso mais essencial de todos na visão de Paula Filho (2003): O diagrama de contexto. Segundo o autor, tal diagrama basicamente exibe todas as interações entre atores e casos de uso. Além disto, ele mostra interações do sistema com sistemas externos.

Já em relação aos requisitos não-funcionais, vale frisar que eles devem ser definidos de forma quantitativa ao incluir requisitos de desempenho, atributos de qualidade do produto, requisitos lógicos e de restrições ao desenho do projeto (Paula Filho, 2003).

2.2.4 PLANEJAMENTO DO PROJETO

O planejamento do projeto incorpora o plano de desenvolvimento do sistema até o seu fim através de um documento gerencial que define custos, prazos e recursos do sistema. (Paula Filho, 2003)

Tais dimensões dependem uma da outra, sendo que a mudança ou variação de uma afeta as outras duas. Entrando de forma mais detalhada em cada uma delas, pode-se salientar que os custos estão relacionados com o tamanho e complexidade do projeto, os prazos estão relacionados a um equilíbrio entre quão rapidamente os recursos podem ser utilizados para compor o sistema final e a necessidade temporal do cliente e finalmente os recursos consistem nos ativos disponíveis para a realização do projeto. (Paula Filho, 2003)

2.2.5 ANÁLISE E DESIGN

Segundo Paula Filho (2003), a fase de análise e design consiste na ponte entre o planejamento do sistema e sua implementação. Enquanto a análise consiste na

caracterização do que o problema abrange, a parte de design aborda a definição de uma estrutura implementável baseada nos requisitos já bem especificados.

No tocante à análise, a primeira atividade a se fazer é identificar as classes do sistema, que são objetos para os quais o sistema deve realizar a armazenagem e operação. Os relacionamentos de tais classes podem ser de quatro tipos: Associação simples, agregação, composição e generalização. (Paula Filho, 2003)

De acordo com Paula filho (2003), enquanto na associação simples as classes relacionam-se de forma direta, na agregação elas compõem uma relação todo-parte em que o todo pode existir sem a parte, na composição elas compõem uma relação todo-parte que, no entanto, o todo não pode existir sem as partes e finalmente na generalização elas fazem parte de uma relação na qual uma classe mais abrangente generaliza outras mais específicas.

Finalmente, ainda segundo o mesmo autor, para as classes identificadas podem ser elaborados diagramas estáticos de relacionamento e atribuídos propriedades desta e ações que a mesma realiza.

Já em relação ao design, Paula Filho (2003) aponta que nesta etapa, ao contrário da análise, deve-se detalhar aspectos funcionais do sistema com o intuito de aproximar o projeto do sistema da sua etapa de implementação.

Tendo isto em vista, conforme o mesmo autor, primeiramente os atributos precisam ser detalhados. Tal detalhamento inclui encontrar um nome autoexplicativo para cada atributo e na especificação do tipo e formato do mesmo.

Em um segundo momento, também pode ser elaborado o diagrama de sequência, o qual cumpre o papel de representar o comportamento dinâmico do sistema e é formado a partir dos casos de uso e do diagrama de classes. De forma resumida, eles descrevem quem faz o que, quando e com cada coisa. Considerando sua elaboração, deve-se partir de um caso de uso com o intuito de se identificar quais operações são realizadas com cada classe e sua ordem de execução. Vale frisar que no diagrama final o tempo é representado na dimensão vertical.

3 PLANEJAMENTO DO SISTEMA

Para iniciarmos o processo de desenvolvimento de um software é necessários planejamento e com algumas atividades básicas já é possível criar um projeto para tal.

O processo de planejamento, pode se dizer, é um conjunto de atividades organizadas em que é preciso definir, desenvolver, testar e manter um software.

Para a construção do sistema, iremos utilizar a linguagem que vem sendo ministrada no tecnólogo: Análise e desenvolvimento de sistemas pelo professor César Tofanini.

3.1 UTILIZAÇÃO DO MÉTODO SCRUM

Engenharia de Software é a área da computação voltada para a parte teórica do desenvolvimento de software, incluindo, também a manutenção de um software já desenvolvido. Visando a produtividade e qualidade do produto.

O método mais utilizado para gerenciamento de projetos, ou seja, a melhor forma de organizar e executar é o Scrum, visando melhor organização no desenvolvimento de um software é o mais utilizado para projetos ágeis no mundo. Para nosso projeto na criação de um software para gerir um portfólio de imóveis de pequeno porte, utilizaremos algumas técnicas Scrum.

Logo abaixo, a figura 2 é a representação das interações entre as atividades no processo do projeto.

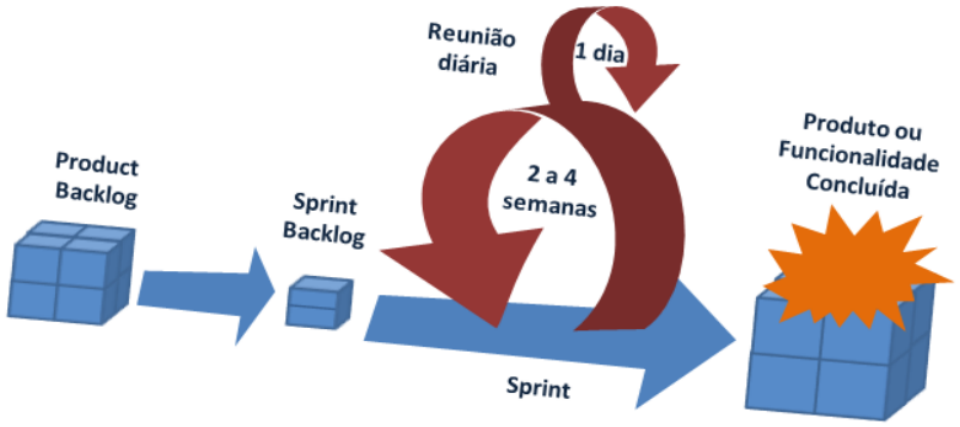

Figura 2 - A metodologia ágil Scrum.
Fonte: *MindMaster* Educação profissional.

3.2 LEVANTAMENTO DE REQUISITOS FUNCIONAIS

Sommerville (2003) define os requisitos de um sistema como o conjunto de suas funções e restrições sobre sua operação e implementação. Define ainda a engenharia de requisitos como o processo de descobrir, analisar, documentar e verificar estas funções e restrições.

Conclui-se que os requisitos representam as funcionalidades e restrições do sistema a ser desenvolvido, em outras palavras, "aquilo que o cliente necessita ou do ponto de vista de um desenvolvedor "o que necessita ser projetado".

Já o levantamento de requisitos, descrito por Bezerra (2007) como um estudo exploratório das necessidades dos usuários e da situação do sistema atual, foi realizado através de observação do ambiente do usuário, realização de entrevistas com o usuário, entrevistas com especialistas do domínio, além de comparação com sistema preexistente do mesmo domínio do negócio.

Os principais requisitos de um sistema, segundo Sommerville (2003), estão descritos a seguir:

- Requisitos de usuário. São os requisitos das pessoas que irão adquirir e utilizar o sistema. Eles devem ser escritos utilizando-se linguagem natural, tabelas e diagramas, de modo que sejam compreensíveis.

- Requisitos de sistema. São descrições mais detalhadas dos requisitos de usuários, especificando de forma completa e consistente todo o sistema. Devem comunicar, de modo preciso, as funções que o sistema tem de fornecer. Podem ser escritos em uma linguagem estruturada, uma linguagem com base em uma linguagem de programação de alto nível ou uma linguagem especial para a especificação de requisitos.

- Requisitos funcionais. São as funções ou os serviços que se espera que o sistema forneça, indicando como o sistema deve reagir a entradas específicas e como se deve comportar em determinadas situações.

Exemplo: "o software deve emitir relatórios de compras a cada quinze dias".

- Requisitos não-funcionais. Não dizem respeito diretamente às funções específicas fornecidas pelo sistema e restringem o sistema a ser desenvolvido. São divididos em requisitos do produto (ex.: eficiência, confiabilidade, manutenibilidade, portabilidade, usabilidade, velocidade, desempenho, robustez), requisitos organizacionais (ex.; requisitos de desenvolvimento e documentação, custos) e requisitos externos (ex.: requisitos legais, éticos, de segurança, de interoperabilidade e de privacidade). Exemplos: "a base de dados deve ser protegida para acesso apenas de usuários autorizados", "o tempo de resposta do sistema não deve ultrapassar 30 segundos", "o software deve ser operacionalizado no sistema Linux" e "o tempo de desenvolvimento não deve ultrapassar seis meses".

Segundo Queiroz (2006), dentre as partes interessadas nos requisitos, pode-se citar: analistas de sistemas, arquitetos de sistema, engenheiros de software, gerentes e

contratantes da organização, usuários finais. Já os problemas mais comuns da atividade de levantamento de requisitos são:

- Os envolvidos, ou parte interessada, não sabem o que eles realmente querem e expressam-se com um vocabulário diferente do dos desenvolvedores.
- Os envolvidos podem ter requisitos conflitantes.
- Fatores organizacionais e políticos podem influenciar os requisitos.
- Novos requisitos podem surgir durante o processo de levantamento, análise e/ou especificação.

3.3 REQUISITOS FUNCIONAIS

RF1. O sistema deve registrar cadastros de imóveis para venda. As informações necessárias para cadastro são: ID Imóvel; tipo de imóvel para venda: apartamento, casa, terreno ou chácara; área total do terreno (ATT), área construída total (ACT); bem como a ficha técnica, detalhando a quantidade de: dormitório(s), suítes(s), sala(s), sala(s) de estar, cozinha(s), banheiro(s); a infraestrutura, se possui área da lazer; valor do imóvel.

RF2. O sistema deve registrar cadastros de imóveis para locação. As informações necessárias para cadastro são: ID Imóvel; tipo de imóvel para locação: apartamento, casa, terreno, chácara; área total do terreno (ATT), área construída total (ACT); bem como a ficha técnica, detalhando a quantidade de: dormitório(s), suítes(s), sala(s), sala(s) de estar, cozinha(s), banheiro(s); a infraestrutura, se possui área da lazer; valor do imóvel.

RF3. O sistema deve registrar cadastros de funcionários corretores: registrar número do crachá; nome; data de contratação; data de nascimento; salário bruto; comissão por vendas de imóveis; salário líquido.

RF4. O sistema deve registrar clientes interessados em adquirir um imóvel. As informações necessárias para cadastro são: número do crachá do corretor que está negociando; ID Imóvel do interesse; nome; CPF; data de nascimento; número do telefone celular; renda mensal; valor a ser pago como entrada.

RF5. O sistema deve registrar toda venda de um imóvel ou a locação de um imóvel; deve registar qual corretor foi o responsável pela venda ou a locação de um imóvel.

RF6. O Sistema deve informar opções de financiamento para Price e SAC.

RF7. O sistema deve ter uma tela de exibição de imóveis disponíveis para venda; nessa tela, além de exibir os imóveis disponíveis para venda deverá ser possível fazer baixas de

proprietários que desistiu de vender seu imóvel, e de fazer baixas de imóveis que tiveram a venda concluída

RF8. O sistema deve ter uma tela de exibição de imóveis disponíveis para locação; nessa tela, além de exibir os imóveis disponíveis para locação deverá ser possível fazer baixas de proprietários que desistiu de locar seu imóvel, e de fazer baixas de imóveis que tiveram locação concluída.

RF9. O sistema deve exibir os corretores contratados da imobiliária; deve ser possível visualizar todas as vendas e/ou locações realizadas por cada corretor; deve exibir o total de comissão recebida, o salário bruto e o salário líquido do mês vigente.

RF10 O sistema deve exibir clientes que tiveram em contato com algum corretor da imobiliária e se interessaram por algum imóvel disponível.

RF11. O sistema deve exibir toda venda e/ou locação que houve na imobiliária; deverá conter todas a informações do imóvel, do cliente e do corretor que concluiu o negócio.

RF12. O sistema deve exibir as opções de financiamento: Sistema de amortização constante (SAC) e *Price*.

3.4 LEVANTAMENTO DE REQUISITOS NÃO FUNCIONAIS

Requisitos não funcionais são aqueles que não necessariamente estão relacionados, de fato, as funcionalidades do sistema. Para levantamento dos requisitos não funcionais é necessário entendermos a arquitetura do sistema que pretendemos desenvolver, trata-se da parte arquitetural e da qualidade do sistema, tendo como exemplos o desempenho, portabilidade, manutenção, confiabilidade e segurando do software.

Na figura 3, de acordo com artigo do site devmedia.com, mostra um subconjunto de requisitos não funcionais, denominados de requisitos de produtos os quais estão associados à arquitetura de um sistema de software. A figura exibe um conjunto de 7 requisitos não funcionais, sendo alguns destes ainda decompostos.

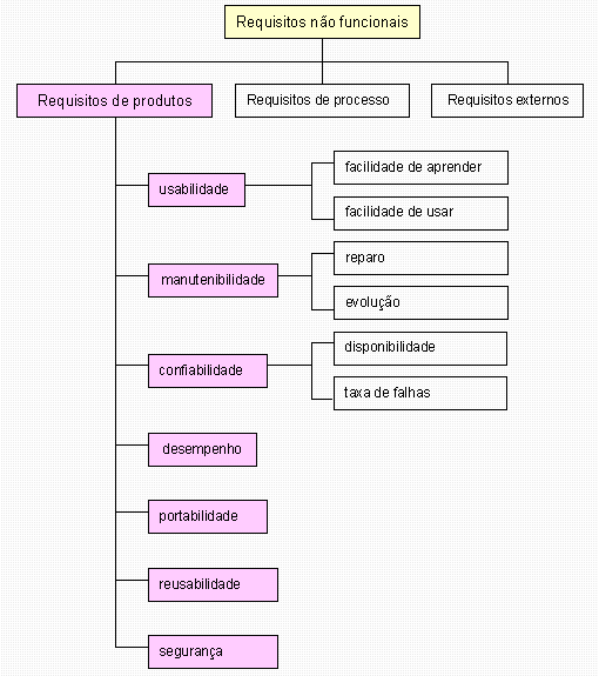

Figura **3** - Subconjunto requisitos não funcionais

Fonte: devmedia.com.br

3.5 REQUISITOS NÃO FUNCIONAIS

RNF1. O sistema deve ser intuitivo, com poucas, mas necessárias telas.

RNF2. O sistema deve ter um tempo de resposta ágil, com interações com o usuário.

RNF3. O sistema deve ter usabilidade alta, em que todos itens disponíveis no menu tem a obrigação de serem utilizáveis. Reduzindo, também, o tamanho do software desenvolvido.

3.6 ENDEREÇO DE IP

Segundo notas de aula da disciplina 209S – Fundamentos de Redes Dados e Comunicação ministrado pela Professora Renata Caceres (2019), para cada dispositivo como: computador, smartfone e até mesmo uma impressora que estejam conectados a uma rede de computadores utilizam um endereçamento IP (*Internet Protocol*) para comunicação. Esse endereço permite identificar o dispositivo e a rede na qual ele pertence.

Existem duas versões de Endereço IP: Ipv4 que é definida por um número de 32bits podendo alcançar até 4.294.967.296 endereços, mas que, devido ao crescimento da internet e restando poucos endereços IPv4, surge outra versão, a IPv6 que é definida por um número de 128bits podendo alcançar: 340.282.366.920.938.463.463.374.607.431.770.000.000

A rede TCP/IP tem um ponto de saída que é para onde vão todos os pacotes recebidos que não são para aquela rede: gateway.

3.7 CLASSES DE ENDEREÇO IP

Para a distribuição de endereços IP foram criadas 5 classes, conforme quadro abaixo:

Classe	Endereço mais baixo	Endereço mais alto
A	1.0.0.0	126.0.0.0
B	128.1.0.0	191.255.0.0
C	192.0.1.0	223.255.255.0
D	224.0.0.0	239.255.255.255
E	240.0.0.0	255.255.255.254

Figura 4 - Apresentação Dados e Comunicação
Fonte: Renata Cáceres

Em redes é utilizado apenas as classes A, B e C para endereço IP.
A classe A consegue endereçar até 16.77.216 máquinas.
A classe B consegue endereçar até 65.536 máquinas.
A classe C consegue endereçar até 256 máquinas.

3.8 MÁSCARA SUB-REDE

Usamos a sub-rede como uma divisão de uma rede de computadores. Criada para reduzir o tráfego de rede e melhorar a performance de rede.
Conforme planejamento do projeto em desenvolver um sistema de informação para gerir um portfólio de imóveis de pequeno porte, utilizaremos a classe C e distribuiremos em 64 sub-redes para distribuição de endereçamento IP de redes para utilização da corporação em geral (financeira, recursos humanos, corretores e etc).
Abaixo a figura ** mostra o cálculo com a máscara de sub-rede 255.255.255.0 da Classe C, IP de rede 192.131.163.0 e a utilização de 64 sub-redes.

Endereços de Sub-Redes	Endereços possiveis de Hosts em cada Sub-Rede	Broadcast
192.131.163.0	192.131.163.1 à	192.131.163.63
225.255.255.192	192.131.163.62	
192.131.163.64	192.131.163.65 à	192.131.163.127
225.255.255.192	192.131.163.126	
192.131.163.128	192.131.163.129 à	192.131.163.191
225.255.255.192	192.131.163.190	
192.131.163.192	192.131.163.193 à	192.31.163.255
225.255.255.192	192.131.163.254	

Figura 5 – Tabela endereçamento de sub-redes Fonte: Elaborado pelo autor

4 SISTEMA IMPLEMENTADO RESULTANTE

4.1 PLATAFORMA UTILIZADA

O sistema de apoio à decisão planejado foi desenvolvido com a utilização do C++ e suas ferramentas de programação. Inicialmente, esta é uma plataforma mais utilizada pelos iniciantes na área de programação. Além disto, levando em conta as necessidades de vínculos relativamente complexos entre os dados, foi necessário um estudo aprofundado do software, para atender os requisitos básicos solicitados pelo escopo do projeto.

4.2 JUSTIFICATIVA

A Plataforma tem como objetivo principal facilitar a negociação e a manipulação dos imóveis, será possível administrar, planejar, visualizar os imóveis disponíveis, a região do imóvel e apresentar especificações dos imóveis aos clientes, usando as ferramentas tecnológicas para o desenvolvimento de um sistema. Tendo como objetivo um bom relacionamento entre as duas partes, sendo possível simular financiamento, sem necessariamente comprovações de renda e/ou consultas de inadimplências.

O sistema imobiliário tem a finalidade de ajudar pessoas com a dificuldade de encontrar imóveis, o sistema torna o trabalho rápido e útil com muitas informações para facilitar a vida do cliente.

A plataforma visa possibilitar algumas informações de compra e locação de imóveis disponíveis no sistema de acordo com as necessidades solicitadas pelo cliente.

4.2.1 HISTÓRIA DA LINGUAGEM C

Segundo o artigo “História do C / C++” (Alisson Santos, 2012). A Linguagem C foi inventada e foi implementada no início dos anos 70 por Dennis Ritchie em um DEC PDP-11, usando o Sistema Operacional UNIX.

A linguagem C é o resultado do processo de desenvolvimento iniciado com outra linguagem, chamada BCPL, desenvolvida por *Martin Richards*. Esta linguagem influenciou a linguagem inventada por Ken Thompson, chamada linguagem B.

Sendo assim a linguagem C é a evolução da linguagem B.

A linguagem C se tornou uma das linguagens de programação mais utilizada, por ser flexível e ainda poderosa, sendo que ela é a responsável pela criação de alguns softwares famosos e a base de outros como jogos.

A linguagem C encontra seus limites quando o tamanho de um projeto ultrapassa certo ponto de 25.000 a 100.00 linhas de código. Para a solução desse problema em 1980 um estudioso chamado *Bjarne Stroustrup* acrescentando várias intenções na linguagem C deu origem à nova linguagem que se chamava inicialmente “C com classes”, e por volta dos anos 1983 o nome foi mudado para linguagem C++. Mais a evolução do C++ não parou com o Bjarne Stroustrup, muitas foram as implementações, até tornarem a linguagem C++ uma linguagem que suporta Programação Orientada a Objetos. As inspirações relacionadas acima devem através de outra linguagem de programação chamada Simula67.

Atualmente a linguagem C / C++ é utilizada nas faculdades de Sistema de Informação e Ciência da Programação em aulas de Lógica de Programação e Estrutura de dados.

5 APRESENTAÇÃO E LÓGICA DO SISTEMA

Entendemos como apresentação e lógica de programação técnicas de desenvolver algoritmos para atingir determinados objetivos dentro de regras de lógica matemática e teorias da Ciência da Computação, aglomerado de informações necessárias para entendimento de um sistema, explicações detalhadas sobre comandos e funcionalidades disponíveis ao usuário.

5.1 MENU INICIAL

Conforme figura 6 - Programação referente a língua utilizada no sistema, utilizamos o comando *setlocale* em conjunto ao cabeçalho *<locale.h>,* para conseguirmos adicionar normas da língua portuguesa se faz necessário essa inclusão (utilizamos para todas as opções de menu existente no sistema).

```
int main()
{
    setlocale(LC_ALL, "Portuguese");
```

Figura 6 - Programação referente a língua utilizada no sistema
Fonte: Elaborado pelo autor

Conforme figura 7 - Programação referente a lógica de inicialização do sistema, utilizamos uma variável com o valor negativo para ser possível a saída do programa com um *looping*. Utilizamos o comando *return 0,* para ser possível sair do sistema não como um erro, mas sim, como uma opção.

```
int operacao = -1; // cria uma variável do tipo inteiro com valor -1. o valor servirá para dar entrada no loop

int i = login(i); // cria uma variável do tipo inteiro de nome i que será igual ao resultado das operações
system("cls");   // descritas na função login(i);

if (i == 0) // se (i for igual a 0)
{
    return 0; //significa que o usuário desistiu de fazer login, e pediu para fechar o programa, sendo assim, o programa fecha.
}
```

Figura 7 - Programação referente a lógica de inicialização do sistema Fonte: Elaborado pelo autor

Conforme figura 8 – Programação referente a exibição de tela de cadastro de corretores, utilizamos o comando *puts* para inserir uma mensagem descrita em tela e pula para a linha de baixo. O comando *printf* utilizamos apenas para organização da mensagem em tela. As variáveis *int* (tipo inteira) se faz necessária para condições de *looping*.

```c
int login(int x)
{
    Corretor Info;
    char senha[12];
    int cracha, op = -1, op2 = 0, vf;
    FILE* Ponteiro;

    inicio:
    system("cls");
    puts("<1> ::: Login");
    puts("<2> ::: Cadastrar");
    printf("Operação: "); scanf("%i", &op); puts("");
    system("cls");
    switch(op)
    {
        case 1:
            puts("\n::: Login :::\n\n");

            printf("Crachá: "); scanf("%i", &cracha); puts("");

            Ponteiro = fopen("Corretores.txt", "r");
```

Figura 8 - Programação referente a exibição de tela de cadastro de corretores
Fonte: Elaborado pelo autor

Conforme figura 9 – Programação referente ao preenchimento de informações para cadastro de corretores, utilizamos o comando *printf o programa* exibirá a mensagem em tela, o usuário precisará seguir com o preenchimento seguindo as informações pedintes em tela.

```c
void listarcorretores ()
{
    int op;
    Corretor Info;
    Ponteiro = fopen("Corretores.txt", "r");

        while(!feof(Ponteiro))
        {
            if(Info.numero_do_cracha == NULL)
            {
                printf("Não existem registros!\n\n");
                return 0;
            }
            fscanf(Ponteiro, "%i %s %s %s %i %f %f\n", &Info.numero_do_cracha, Info.nome, Info.data_contratacao, Info.data_nascimento, &Info.idade, &Info.salario_b, &Info.comissao_pvenda);
            printf("Crachá: %i\n", Info.numero_do_cracha);
            ColocarEspaco(Info.nome);
            printf("Nome: %s\n", Info.nome);
            printf("Data de nascimento: %s\n", Info.data_nascimento);
            printf("Data de Contratação: %s\n", Info.data_contratacao);
            printf("Idade: %i\n", Info.idade);
            printf("Salário Bruto: %0.2f\n", Info.salario_b);
            printf("Comissão por venda: %0.2f\n\n", Info.comissao_pvenda);
        }
        fclose(Ponteiro);
        puts("Fim da listagem!\n\n");
        return 0;
```

Figura 9 - Programação referente ao preenchimento de informações para cadastro de corretores
Fonte: Elaborado pelo autor

Conforme figura 10 – Programa referente ao menu inicial, utilizamos o comando *while* para criação de um *looping* que só irá parar quando uma variável, previamente declarada, nos dê a condição necessária para tal.

Utilizamos o comando *scanf* para ler um valor e armazenar em uma variável declarada.

Utilizamos o comando *system("cls")* para limpar a tela, e abrir espaço para uma próxima operação.

```c
while (operacao < 8) // cria um loop. enquanto(operacao for < 8) ele executa o corpo principal
{
    puts("::: Menu Principal :::\n\n");
    puts("<1> ::: Cadastrar Imóvel");
    puts("<2> ::: Cadastrar Corretor");
    puts("<3> ::: Simulação Financiamento");
    puts("<4> ::: Listar registrados");
    puts("<5> ::: Vender Imovel");
    puts("<6> ::: Alugar Imóvel");
    puts("<7> ::: Apagar Registros");
    puts("<8+> ::: Sair");
    //Lista um menu e pede para o usuário digitar uma das opções
    printf("Operação: ");
    scanf("%i", &operacao); puts("");    //leia ("valor inteiro", e coloque na variável operacao), que serve
    system("cls");                       //justamente para a comparação do while.
```

Figura 10 - Programação referente ao menu inicial Fonte: Elaborado pelo autor

Conforme figura 11 – Programação referente as seleções de menu, utilizamos o comando *switch case*, onde contendo uma variável é comparado com um valor constante, e caso a comparação seja verdadeira, um determinado comando é executado. Para formação de menu é o comando mais utilizado na linguagem C. O *break* sempre é utilizado para finalizar o *case*, sem ele o programa não funciona da forma desejada.

```c
switch (operacao) //Compare (operacao)
{
    case 1: //caso operacao seja igual a 1:
        printf ("::: Cadastrar Imóvel :::\n\n"); //coloque na tela(":::: Cadastrar Imóvel ::: \quebra de linha\quebra de linha");
        Imovel();  //chama a função Imovel();
        break;  //após feita a operação, o programa sai do switch com o comando break;  O mesmo processo se repete nas linhas abaixo.
    case 2:
        printf("::: Cadastrar Corretor :::\n\n");
        corretor();
        break;
    case 3:
        printf("::: Simulação Financiamento :::\n\n");
        SimulacaoFinanciamento();
        break;
    case 4:
        printf("<1> ::: Imóveis p. Venda\n<2> ::: Imoveis p. Aluguel\n<3> ::: Corretores\n<4+> ::: Sair\nOperação: ");
        // pede para o usuário digitar qual tipo imóvel deseja listar.
        scanf("%i", &operacao); //lê e armazena na variável operacao
        system("cls"); //sistema ("limpa a tela");
        switch(operacao) //compare(operacao)
        {
            case 1: //caso operacao seja igual a 1. OBS: mesma lógica aplicada nas linhas abaixo.
                ListagemVendas(); // chama a função listagemVendas();
                break;
            case 2:
                listaraluguel();
                break;
            case 3:
                listarcorretores();
                break;
            default: //se o switch comparou opercao com todos os case e nenhum foi compatível, ele vem até o comando default
                break;  //e sai da operação
        }
        break;

    case 5:
        printf("::: Venda de Imóveis :::\n\n");
        venda_imovel();
        break;
    case 6:
        printf("::: Aluguel de Imóveis :::\n\n");
        alugar_imovel();
        break;
    case 7:
        printf("::: Exclusão de registros :::\n\n");
        excluir_registros();
        break;
    default:
        break;
```

Figura 11 – Programação referente as seleções de menu
Fonte: Elaborado pelo autor

5.2 ESTRUTURAS DO PROGRAMA

Conforme a figura 12 – Programação referente a estrutura do cadastro de imóveis e a figura 13 – Programação referente a estrutura do cadastro de corretores, utilizamos o comando *typedef struct* para definir os tipos de dados que agrupam variáveis sob um mesmo tipo de dado. A ideia de usar uma *struct* é permitir que, ao armazenar os dados de uma mesma entidade, isto possa ser feito com uma única variável.

```
typedef struct {
    int Qtd_Dormitorio, Qtd_Suite, Qtd_Sala, Qtd_Sala_de_Estar, ID, Qtd_Cozinha, Qtd_Banheiro, andar;
    char infraestrutura[1000], Area_de_Lazer[4], endereco[200], n_telefone[15];
    float valor_do_imovel, Area_do_terreno, Area_construida_total;
}Ficha_Tecnica_Imovel;
```

Figura 12 - Programação referente a estrutura do cadastro de imóveis
Fonte: Elaborado pelo autor

```
typedef struct {
    char nome[200], data_contratacao[14], data_nascimento[14], senha[12];
    int idade, imoveis_vendidos, numero_do_cracha;
    float salario_b, comissao_pvenda, salario_l;
}Corretor;
```

Figura 13 - Programação referente a estrutura do cadastro do corretor
Fonte: Elaborado pelo autor

5.3 FUNÇÕES E PONTEIROS

Conforme figura 14 – Programação referente ao menu de opções de locação de imóveis, utilizamos opções de escolhas do tipo de imóvel desejado (apartamento, casas etc.) com comando de estrutura de *ponteiros* e *funções*, que é algo padronizado da linguagem C, seguimos a mesma lógica para todas as telas de menu do programa.

```
void alugar_imovel()
{
        int op, cracha, id = 0, vf = 0; //cria variáveis do tipo inteiro que servirão de parâmetros de comparação futuramente
        Ficha_Tecnica_Imovel Info; //cria uma variável do tipo Ficha_Tecnica_Imovel
        Corretor Info2, Info3;
        FILE* Ponteiro;
        FILE* PonteiroNovo;
        FILE* PonteiroNovo2;


        puts("<1> :::  Apartamentos");
        puts("<2> :::  Casas");
        puts("<3> :::  Chácaras");
        puts("<4> :::  Terrenos");
        puts("<5+> ::: Sair"); printf("Operação: "); scanf("%i", &op); puts("");
        system("cls");

        switch (op)
        {
            case 1:
                printf("::: Alugar Apartamentos :::\n\n");
                printf("<1> ::: Listar Apartamento disponíveis para Aluguel:\n<2> ::: Continuar sem listar\n "); printf("Operação: "); scanf("%i", &op); puts("\n");
                if(op == 1)
                {
                    system("cls");

                    Ponteiro = fopen("AlugarApartamentos.txt", "r");
                    while (!feof(Ponteiro))
                    {
```

Figura 14 - Programação referente ao menu de opções de locação de imóveis
Fonte: Elaborado pelo autor

Conforme figura 15 – Programação referente ao preenchimento de informações para cadastro de imóveis, utilizamos o comando *printf* para exibição em tela, seguimos a mesma lógica para todas as telas de menu do programa.

```
        printf("ID: %i \n", Info.ID);
        printf("Andar: %i\n", Info.andar);
        printf("Quantidade de Dormitório(s): %i \n", Info.Qtd_Dormitorio);
        printf("Quantidade de Suíte(s): %i \n", Info.Qtd_Suite);
        printf("Quantidade de Sala(s): %i \n", Info.Qtd_Sala);
        printf("Quantidade de Sala(s) de Estar: %i \n", Info.Qtd_Sala_de_Estar);
        printf("Quantidade de Cozinha(s): %i \n", Info.Qtd_Cozinha);
        printf("Quantidade de Banheiro(s): %i \n", Info.Qtd_Banheiro);
        ColocarEspaco(Info.infraestrutura);
        printf("Informações da Infraestrutura: %s \n", Info.infraestrutura);
        ColocarEspaco(Info.endereco);
        printf("Endereço: %s \n", Info.endereco);
        printf("Possui área de lazer?: %s\n", Info.Area_de_Lazer);
        printf("Área total: %.2f\n", Info.Area_do_terreno);
        printf("Terreno total construído: %.2f\n", Info.Area_construida_total);
        printf("Valor do Imóvel: %.2f\n", Info.valor_do_imovel);
        printf("Nº de telefone: %s\n\n", Info.n_telefone);
}
fclose(Ponteiro);
printf("Listagem concluída!\n\n");
```

Figura 15 - Programação referente ao preenchimento de informações para cadastro de imóveis

Fonte: Elaborado pelo autor

Conforme figura 16 – Programação referente ao ponteiro para cadastro de corretores, utilizamos o comando *if* como estrutura de decisão e o comando *goto* como estrutura de controle para salto de instruções.

```
cracha:
PonteiroNovo = fopen("Corretores.txt", "r");
Info3.numero_do_cracha = 0;
printf("Por favor informe o crachá: "); scanf("%i", &cracha); puts("");
while(!feof(PonteiroNovo))
{

    fscanf(PonteiroNovo, "%i %s %s %s %i %f %f\n", &Info2.numero_do_cracha, Info2.nome, Info2.data_contratacao, Info2.data_nascimento, &Info2.idade, &Info2.salario_b, &Info2.comissao_pvenda);

    if(Info2.numero_do_cracha == cracha)
    {

        Info3.numero_do_cracha = cracha;
    }
}
fclose(PonteiroNovo);
if (Info3.numero_do_cracha == 0)
{
    printf("Crachá não existe/não registrado!\n\n<1> ::: Tentar Novamente?\n<2+> ::: Sair "); scanf ("%i", &op); system("cls");
    if (op == 1)
    {
        goto cracha;
    }
    else
    {
        goto fim;
    }
}
```

Figura 16 - Programação referente ao ponteiro para cadastro de corretores

Fonte: Elaborado pelo autor

5.4 SIMULAÇÃO DE FINANCIAMENTO

Conforme figura 17 – Programação referente ao menu de opções de financiamento, o usuário terá um pequeno menu disponível para simulações das tabelas PRICE e SAC para escolher a melhor forma de financiamento. Após escolher a tabela, o usuário irá informar o valor, juros e quantidade de meses. Finalizando a simulação de financiamento o software irá mostrar os resultados seguindo a tabela escolhida, conforme figura 18 - Resultados do financiamento, concluindo o financiamento você poderá fazer outro financiamento com a tabela preferida, há

a possibilidade de sair das opções de simulações de financiamento e voltar para o menu principal.

```
void SimulacaoFinanciamento ()
{
       int op;
       float juros=0, juros_c=0, saldo_devedor=0, amortizacao = 0, valor_parcela=0, valor_parcela_aux, tempo=0, parcela = 0;
       Ficha_Tecnica_Imovel Info;


       puts("<1> ::: SAC");
       puts("<2> ::: PRICE");
       puts("<3+> ::: Sair");
       printf("Operação: "); scanf("%i", &op); puts("");

       switch (op)
       {
           case 1:
               printf("Digite o valor: ");
               scanf("%f", &Info.valor_do_imovel);
               printf("Digite o juros: ");
               scanf("%f", &juros);
               printf("Tempo em meses: ");
               scanf("%f", &tempo); puts("/n/n");

               juros = juros / 100;
               amortizacao = Info.valor_do_imovel/tempo;
               saldo_devedor = Info.valor_do_imovel;

               printf("Juros: %f\n", juros);
               printf("Amortizacao: %f\n", amortizacao);
               printf("Saldo devedor: %f\n\n", saldo_devedor);

               printf(":Mês: \t:Amortizacao: \t:Juros: \t:Parcela: \t:Saldo:\n", parcela, amortizacao, juros_c, valor_parcela, saldo_devedor);
               printf("%0.2f  \t %0.2f  \t %0.2f  \t\t %0.2f  \t\t %0.2f\n", parcela, amortizacao, juros_c, valor_parcela, saldo_devedor);
```

Figura 17 - Programação referente ao menu SAC de financiamento
Fonte: Elaborado pelo autor

```
<1> ::: SAC
<2> ::: PRICE
<3+> ::: Sair
Operação: 2

Digite o valor: 5000
Digite o juros: 4,58
Tempo em meses: 10
/n/n
:Mês:    :Amortizacao:    :Juros: :Parcela:        :Saldo:
 0,00     0,00            0,00    634,38           5000,00
 1,00     405,38          229,00  634,38           4594,62
 2,00     423,95          210,43  634,38           4170,67
 3,00     443,37          191,02  634,38           3727,30
 4,00     463,67          170,71  634,38           3263,63
 5,00     484,91          149,47  634,38           2778,72
 6,00     507,12          127,27  634,38           2271,61
 7,00     530,34          104,04  634,38           1741,27
 8,00     554,63          79,75   634,38           1186,63
 9,00     580,03          54,35   634,38           606,60
 10,00    606,60          27,78   634,38           -0,00
Pressione uma tecla para continuar:
```

Figura 18 – Resultados do financiamento
Fonte: Elaborado pelo autor

```
        case 2:
            printf("Digite o valor: ");
            scanf("%f", &Info.valor_do_imovel);
            printf("Digite o juros: ");
            scanf("%f", &juros);
            printf("Tempo em meses: ");
            scanf("%f", &tempo); puts("/n/n");

            juros = (juros / 100)+1;


            saldo_devedor = Info.valor_do_imovel;
            valor_parcela = (pow (juros, tempo)-1);
            valor_parcela_aux = pow (juros, tempo) * (juros - 1);
            valor_parcela = (valor_parcela_aux / valor_parcela) * Info.valor_do_imovel;
            juros = juros -1;
            saldo_devedor = Info.valor_do_imovel;

            printf(":Mês:\t:Amortizacao:\t:Juros:\t:Parcela:\t:Saldo:\n", parcela, amortizacao, juros_c, valor_parcela, saldo_devedor);
            printf(" %0.2f \t %0.2f \t\t%0.2f \t%0.2f  \t%0.2f\n", parcela, amortizacao, juros_c, valor_parcela, saldo_devedor);

                while (parcela < tempo)
                {
                    parcela = parcela + 1;
                    juros_c = juros * saldo_devedor ;
                    amortizacao = valor_parcela - juros_c;
                    saldo_devedor = saldo_devedor - amortizacao;
                    printf(" %0.2f\t %0.2f \t%0.2f \t%0.2f \t\t%0.2f\n", parcela, amortizacao, juros_c, valor_parcela, saldo_devedor);
                }

}
return 0;
```

Figura 19 - Programação referente ao menu PRICE de financiamento
Fonte: Elaborado pelo autor

6 CONCLUSÃO

Este trabalho consistiu no desenvolvimento de um portfólio de imóveis de pequeno porte. No tocante à etapa de planejamento conforme a proposta do Projeto Integrado Multidisciplinar II, nos foi presenteado uma proposta de solução com objetivo de auxiliar usuários que buscam imóveis para comprar e locar afim de exibir detalhadamente toda informação do imóvel desejado para melhor interesse ao consumidor. Com isso, esse sistema visa promover toda a flexibilidade e agilidade possível para atender aos requisitos do cliente, podendo sanar suas dúvidas em relação aos imóveis com a maior rapidez.

Utilizando métodos das disciplinas 219S – Engenharia de Software I; 288S – Linguagem e Técnicas de Programação; 209S – Fundamentos de Redes Dados Comunicação; J584 – Matemática para Computação, sendo: Método Ágil; Linguagem C; Mascara Sub-rede; Calculo tabela PRICE e SAC, respectivamente, solucionamos todos os problemas propostos no planejamento, criação do programa e elaboração do projeto. Desta forma aprendermos, em nosso primeiro contato com uma linguagem de programação a gerenciar, elaborar, planejar e desenvolver um software com a utilização da linguagem C.

7 REFERÊNCIAS

PORTAL G1**. O auge e a queda do mercado imobiliário em uma década**. 2016. Disponível em: < http://g1.globo.com/especial-publicitario/zap/imoveis/noticia/2016/04/o-auge-e-queda-do-mercado-imobiliario-em-uma-decada.html >. Acesso em 5 de out. de 2019.

STAIR, R.M.; REYNOLDS, G.W. ***Princípios de sistemas de informação****. Tradução de Harue Avritscher. 9. ed. São Paulo: Cengage Learning, 2010. 590 p.*

BEZERRA, Eduardo. ***Princípios de Análise e Projetos de Sistemas com UML****. Rio de Janeiro: Elsevier, 2007.*

LAUDON, K.; LAUDON, J. ***Sistemas de informações gerenciais***. Tradução de Luciana do Amaral Teixeira 9. ed. São Paulo: Pearson, 2004. 428 p.

PAULA FILHO, W.P. **Engenharia de software**: Fundamentos, métodos e padrões. 2. ed. Rio de Janeiro: LTC, 2003. 602 p.

SOMMERVILLE, IAN. **Engenharia de Software**. Addison Wesley, 2003.

SANTOS, ALISSON (2012) ***História do C / C++.*** *Disponível em: <* https://www.devmedia.com.br/historia-do-c-c/24029*> Acesso em 24 de nov de 2019.*

.