UNIVERSIDADE PAULISTA - UNIP EAD

Projeto Integrado Multidisciplinar (PIM) V

Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

Davi Cabral - 2199029

Matheus Soares Lima - 0598353

Tiago S F Neves - 0443889

Marcelo Aparhyan de Jesus - 0439966

SOFTWARE EM DESENVOLVIMENTO: AS RESPONSABILIDADES, DESAFIOS E METODOLOGIAS

São Paulo, 2022

Davi Cabral - 2199029

Matheus Soares Lima - 0598353

Tiago S F Neves - 0443889

Marcelo Aparhyan de Jesus - 0439966

SOFTWARE EM DESENVOLVIMENTO: AS RESPONSABILIDADES, DESAFIOS E METODOLOGIAS

Projeto integrado multidisciplinar para a obtenção do título de graduação em Tec em Análise e Desenvolvimento de Sistemas à universidade paulista –UNIP EaD.

Orientador: Prof. Angel Antonio Gonzalez Martinez

São Paulo

2022

RESUMO

Este trabalho tem o objetivo de demonstrar os tópicos e assuntos abordados nas matérias presentes no currículo deste segundo semestre da graduação do curso de Análise e Desenvolvimento de sistemas. O tema proposto diz respeito à elaboração de um projeto de desenvolvimento de software para a reserva de equipamentos audiovisuais. Ao longo do texto, itens como desenvolvimento e qualidade de software, assuntos relacionados à economia e mercado, e também ao projeto de interface de software serão abordados.

Palavras-chave: Desenvolvimento de Software, Qualidade de Software, Interface, C/C++ (linguagem de programação).

ABSTRACT

This work has the objective to demonstrate the topics and subjects treaties in this second semester of graduation in the system analysis and development course. The proposed theme concerns about the elaboration of a project to the development of a software for booking audiovisual equipment. Along the text, items like development and quality of software, topics related to economy and market, and also about the interface's project of software will be approached.

Keywords: Software Development, Software Quality, Interface, C/C++ (programming language).

SUMÁRIO

1. INTRODUÇÃO

O Colégio Vencer Sempre disponibiliza equipamentos de informática e vídeo como ferramentas de apoio aos professores e coordenadores da instituição, disponibilizando-os em salas de aula e auditórios, mediante o pedido dos colaboradores.

Entretanto, dada a demanda altamente volátil que existe para o serviço, e a dificuldade de organizar as reservas manualmente – maneira com que são feitas hoje – muitos professores não conseguem utilizar os recursos que precisam, resultado do agendamento ineficiente que é praticado.

A solução, portanto, é automatizar as reservas de maneira digital. O desafio neste trabalho é pensar o planejamento do processo de desenvolvimento de software, afim de obter uma entrega satisfatória, de acordo com o que foi estabelecido durante a contratação do serviço por parte da instituição.

Para atingir este objetivo, é necessário abordar os temas acerca da viabilidade econômica da execução do projeto, do processo de engenharia de software propriamente dito, dos assuntos relacionados ao código-fonte e aos conceitos de programação orientada a objetos, e também sobre o projeto de interface da aplicação.

2. A VIABILIDADE DE UM PROJETO

Como uma empresa de desenvolvimento de *software* pode decidir sobre a viabilidade que há por trás da execução de um projeto?

No mercado de TI, empresas que atuam na frente de desenvolvimento de aplicações para empresas que demandam sistemas próprios para as suas rotinas de trabalho, exercem sua função de maneira muito semelhante às empreiteiras. Cada contrato fechado entre duas empresas parceiras é fruto de uma extensa conversa envolvendo todas as partes interessadas, até que a viabilidade do projeto seja verificada por cada uma delas.

Para a empresa responsável pela contratação do serviço, é importante verificar o valor que há na compra dele, além de garantir que todas as suas demandas sejam atendidas pelo contrato final da prestação do serviço. Muitas vezes, o contratante deseja, além da aplicação plenamente funcional, um serviço de suporte: uma garantia contra problemas e *bugs* indesejáveis que o sistema pode apresentar. Além disso, existem requisitos estabelecidos pela equipe que será responsável por fazer o uso do *software* em produção.

Por isso, é importante para a empresa responsável pelo trabalho de desenvolvimento de software a análise de todos os custos envolvidos para a obtenção do produto final, a fim de avaliar a viabilidade econômica do projeto.

A estimativa de custos em um projeto deste tipo é essencial para a tomada de decisão dos *stakeholders*. É importante que estas decisões sejam embasadas em números que correspondam à realidade em que a equipe está inserida.

Para o projeto, é indispensável levar em consideração o planejamento do tempo a decorrer até a entrega do produto, a remuneração dos profissionais envolvidos, a infraestrutura necessária para o time de desenvolvimento e os custos adicionais esperados, de acordo com o pensamento marginal que deve estar atrelado às estimativas.

Estas projeções de custos não são obrigações exclusivas das grandes empresas, mas também devem ser realizadas pelas pequenas equipes de profissionais, afim de que a saúde financeira de toda a empresa seja garantida.

2.1. A QUEM INTERESSA AS ANÁLISES DE CUSTOS

É de interesse comum de todos os envolvidos com o projeto a análise de custos envolvida nele. São eles os *stakeholders*.

Para além dos profissionais envolvidos, são também chamados de *stakeholders* os acionistas, gestores, proprietários e quaisquer que façam parte do time. Para além destes, também são interessados os clientes e empresas terceiras que fornecem infraestrutura à equipe.

2.2. A ENTREGA DO PRODUTO

Para a obtenção das estimativas de custos do projeto, é de suma importância entender quanto tempo deve levar até a sua entrega.

Note: essa questão é também contratual, e é estabelecida entre ambas as partes, antes de que qualquer acordo de contratação de serviço de desenvolvimento de software seja fechado.

Não é uma missão fácil estimar o tempo de desenvolvimento que um software vai demandar. São muitas as questões, e todas elas pedem uma margem que deve estar presente no planejamento da equipe, com o objetivo de que todos os seus prazos sejam respeitados.

Para esta estimativa, devem ser considerados: fase de diagnóstico – onde o problema é identificado –, a concepção, o levantamento e a análise de requisitos, a fase de desenvolvimento, a manutenção e o *debug*, até a entrega.

Outra missão é entender o software que está sendo desenvolvido. Ora, uma aplicação de gerenciamento de equipamento de som, com *features* que atendam às demandas dos usuários, que tenha seu desempenho aprovado junto aos ambientes de produção propriamente dito, com um bom trabalho de interface, não é uma tarefa que poderá ser feita em pouco tempo.

Por isso, pensando neste projeto, visando a entrega de um produto funcional, seguro, e totalmente validado pelo crivo do time, foi estabelecida uma agenda de 120 dias de trabalho de desenvolvimento.

É estimado que 80% deste tempo seja gasto com as duas últimas etapas do projeto, que de fato demandam o trabalho mais extenso.

2.3. GASTOS COM COLABORADORES

Para o desenvolvimento do código-fonte do projeto, é necessário um time de desenvolvedores aptos a trabalhar lado a lado, de acordo com a metodologia de desenvolvimento e do cronograma necessário para que as datas e metas sejam atingidas.

Considerando um time com cinco desenvolvedores e a média salarial de um programador da linguagem C/C++ no Brasil na cidade de São Paulo no ano de 2021 – que é de R$5.300, de acordo com uma pesquisa feita pelo *blog Geek Hunter*, em um levantamento com as principais plataformas de oferta e procura de vagas de emprego no país –, o custo projetado para a remuneração desta mão de obra durante o período do projeto é de **R$106.000**.

Além disso, existe um importante integrante para a equipe. É necessário o trabalho do *scrum master*.

Segundo um levantamento do site Análise de requisitos, a média salarial deste profissional responsável por orquestrar o time é de R$8.038.

Em vista disso, o valor final estimado para a remuneração deste profissional é de **R$32.152**. Esta é uma média que pode apresentar variações de acordo com as horas trabalhadas por estes profissionais, além das condições contratuais, tarifárias e burocráticas, acerca da contratação dos processos de contratação.

2.4. EQUIPAMENTOS DE TRABALHO NECESSÁRIOS

Por último – mas não menos importante – é de suma importância que a infraestrutura necessária seja provisionada aos membros do time de desenvolvimento.

A principal demanda quanto à infraestrutura para este projeto, portanto, é a de **seis *notebooks***, que atendam ao *setup* de: um processador i5, da 8ª geração, ou Ryzen 5; 8GB de RAM; Memórias SSD nVME.

Adicionalmente, equipamentos como *mouses* e *headsets* também são de desejo comum.

Para a estimativas de custo destes equipamentos, podemos contar com a ajuda desta estimativa feita pelo site Buscapé. O esperado é que a aquisição deles não saia por menos de R$3.500,00 a unidade.

Por isso, estabelecendo uma margem, o valor estimado de reserva para esta estrutura é de **R$21.000,00**.

3. MODELO DE PROCESSO DE SOFTWARE

Os modelos de desenvolvimento de *software* são modelos que monitoram a criação, desenvolvimento e finalização do software, envolvendo até mesmo seus menores detalhes.

Ao longo dos anos, diversas empresas e profissionais desenvolveram diferentes modelos para a tarefa de desenvolvimento de software. Historicamente, alguns autores hoje apontam que o primeiro modelo utilizado foi aquele em que na verdade não havia um modelo definido; isto é, o cliente ia até o desenvolvedor, apresentava o seu problema, e imediatamente o desenvolvedor começava a codificar. Quando obtinha o primeiro esboço do sistema, ia até o cliente e apresentava o resultado. O cliente então fazia seus apontamentos, e o desenvolvedor voltava para o código-fonte para trabalhar em cima do *feedback* que havia recebido.

Não demorou muito para que o mercado de desenvolvimento de *software* percebesse que trabalhar dessa maneira acarretava em inúmeros problemas.

Os modelos de desenvolvimento de *software*, então, surgiram para ajudar os desenvolvedores e empresas a se organizar nesta tarefa, com o objetivo de que os

prazos sejam respeitados, os recursos otimizados, e as perdas, gastos e despesas minimizados.

3.1. O MPS.BR

O MPS-BR ou Melhoria de Processos do Software Brasileiro, é um modelo de qualidade de processo criado em 2003 pela *Softex* (Associação para Promoção da Excelência do Software Brasileiro), para melhorar a capacidade de desenvolvimento de software nas empresas brasileiras.

De maneira geral, ele sumariza boas práticas de outros modelos internacionalmente reconhecidos, como o CMMI, as normas ISO/IEC 12207 e ISO/IEC 15504, aplicando-as à realidade do mercado brasileiro de software.

3.1.2. A PROPOSTA DO MPS.BR

Começando pelo que esse modelo é formado: são 4 componentes, 7 níveis de maturidade e 19 processos.

Modelo de referência ao software: São as referências que visam um exemplo de "rosto", com desempenho e melhoria do software.

Modelo de referência de serviços: Refere-se à utilidade do software para o cliente, oferecendo-lhe um serviço de qualidade, com desempenho e eficiência.

Modelo de avaliação: É uma forma de avaliação, visando uma crítica construtiva sobre o desenvolvimento do software, com seus pontos positivos e negativos.

Modelo de negócios: Refere-se ao público alvo do software, visando quase sempre sua utilidade, além dos lucros e perdas, incluindo a popularidade que possui nas empresas de pequeno até o alto porte financeiro.

Seus níveis de maturidade são:

• Nível A: Otimizado.

• Nível B: Gerenciado quantitativamente.

• Nível C: Definido.

• Nível D: Largamente definido.

• Nível E: Parcialmente definido.

• Nível F: Gerenciado.

• Nível G: Parcialmente gerenciado.

Figura 1 – Os níveis de maturidade do MPS.BR.

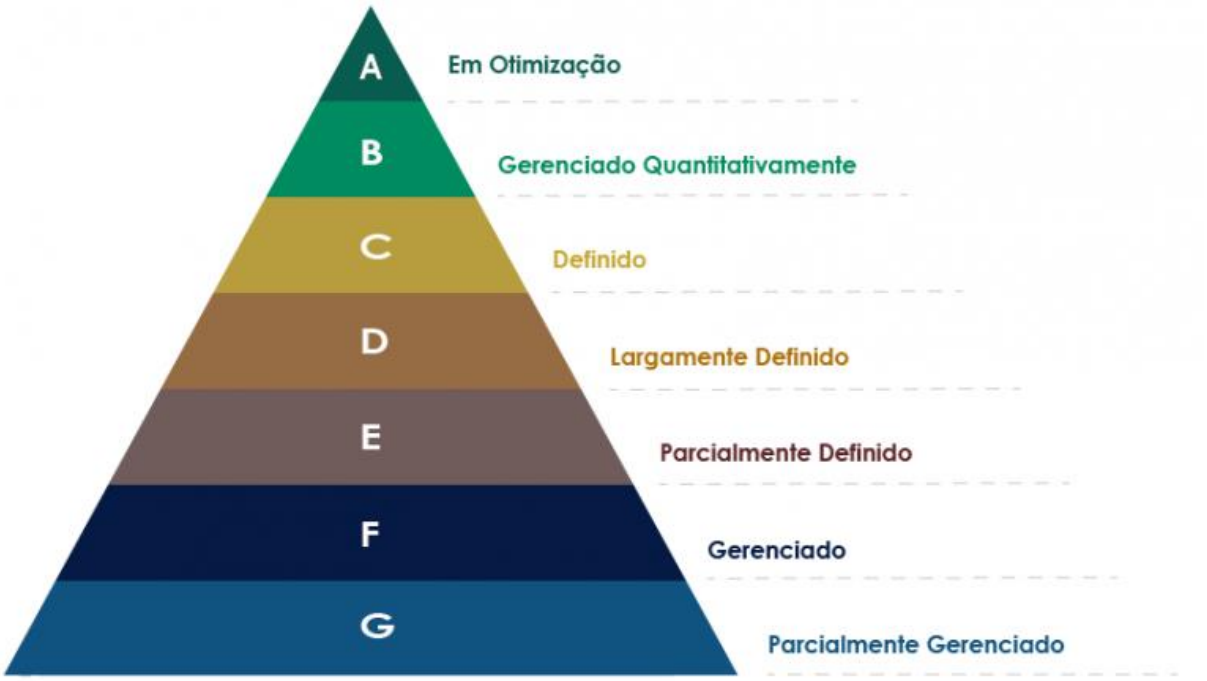


Imagem fonte: https://asrconsultoria.com.br/index.php/mps-br/

3.2. POR QUE A ESCOLHA?

O MPS.BR é um modelo que examina cada detalhe do processo de desenvolvimento, espelhando as normas de qualidade que são aplicadas no mundo todo, com o foco no mercado nacional de software.

É importante frisar que a proposta do MPS.BR é bastante extensa, minuciosa e rígida, conferindo grande qualidade às empresas capazes de aplica-la.

Além disso, outras normas de qualidade de software demandam um custo de aplicação nas empresas que desejam obtê-las. O MPS.BR ganha neste comparativo, já que foi desenvolvido a fim de obter uma maior adesão no mercado brasileiro de software, ajudando empresas de pequeno e médio porte a competirem com empresas maiores no mercado nacional.

4. ENGENHARIA DE REQUISITOS

A engenharia de requisitos trata de um conjunto de tarefas a serem executadas pelo sistema, gerando um produto final que é a documentação destes requisitos. Isto é, tudo o que estiver presente nestes documentos deve estar na versão final do sistema.

Muitos problemas que surgem ao longo do processo de desenvolvimento de software podem ser evitados com a engenharia de requisitos.

4.1. REQUISITOS FUNCIONAIS E NÃO FUNCIONAIS

Os requisitos funcionais são todos os problemas e necessidades que devem ser atendidos no software por meio de funções e serviços.

Os requisitos não funcionais, por outro lado, são todos aqueles relacionados à forma como o software tornará realidade os que está sendo planejado. Ou seja, enquanto os requisitos funcionais estão focados no que será feito, os não funcionais descrevem como serão feitos.

Figura 2 – Listagem dos requisitos funcionais do sistema.

- O sistema deverá permitir ao usuário efetuar o login no sistema
- O sistema deverá permitir ao usuário consultar reservas disponíveis
- O sistema deverá permitir ao usuário confirmar a reserva
- O sistema deverá permitir ao usuário consultar as reservas efetuadas
- O sistema deverá permitir ao usuário um relatório das reservas efetuadas
- O sistema deverá ter um controle de entrada e saída dos equipamentos
- O sistema deverá emitir um protocolo de devolução do equipamento
- O sistema deverá disparar um e-mail ao usuário caso ocorra uma alteração em uma reserva
- O sistema deverá disparar um e-mail ao usuário caso tenha alteração em algum equipamento
- O sistema deverá permitir a alteração da situação dos equipamentos para "ativo", "em manutenção" e "excluído".
- O sistema deverá emitir um relatório dos equipamentos

Figura 3 – Listagem dos requisitos não funcionais do sistema.

- O sistema deverá ter uma interface gráfica com ícones representativos
- O sistema deverá utilizar banco de dados MySQL
- O sistema deverá ser desenvolvido na linguagem C/C++

4.1.2. ROTEIRO DE TESTES PARA OS REQUISITOS MENCIONADOS

As tabelas abaixo mostram o roteiro de testes elaborado para os requisitos funcionais propostos, em uma descrição detalhada do passo a passo para a execução do sistema, a fim de verificar cada caso de teste.

Caso de Teste: Efetuar Login
Procedimento inicial: abrir o software na área de trabalho

ID	Passo para execução	Dado de entrada	Resultado esperado
01	Sistema exibe tela para realizar login	-	Dados exibidos: Campos, *e-mail* e senha
02	Usuário informa *e-mail* e senha e clica em entrar	lucia@colegio.com.br	Login realizado com sucesso

Caso de teste: Solicitar Reserva de Equipamento
Procedimento inicial: Realizar Login no sistema, acessar o menu “Reserva” e clicar em “Solicitar Reserva”

ID	Passo para execução	Dados de entrada	Resultado esperado
01	Sistema exibe tela para solicitar os equipamentos	-	Dados exibidos: campos equipamento, professor, Turma horário, data, data da reserva e observação
02	Usuário informa o equipamento desejado	Retroprojetor Epson Original 12973	Sistema selecionou o equipamento com sucesso
03	Usuário informa o nome do solicitante	Marcos da silva	Sistema selecionou o sistema com sucesso

04	Usuário informa a turma desejada	1° Ano 1A	Sistema selecionou a turma com sucesso
05	Usuário informa horário desejado	18:00h	Sistema selecionou o horário com sucesso
06	Usuário informa a data da reserva	25/01/2022	Sistema selecionou a data com sucesso
07	Usuário clica em cadastrar	-	Sistema apresenta tela com a reserva concluída e o protocolo da reserva

5. CÓDIGO-FONTE

Para a codificação do sistema, devem ser avaliadas as questões à cerca das linguagens de programação necessárias, da arquitetura do sistema e de como serão as entradas e saídas de dados – assunto este que também será abordado nos capítulos adiante.

Acerca dos assuntos relacionados à programação orientada a objetos na linguagem de desenvolvimento *C++*, existem alguns assuntos que precisam ser destrinchados para a estruturação correta do código-fonte.

A orientação a objetos é um paradigma de análise, de projeto e programação de sistemas de informação baseado na composição e interação entre diversas unidades de software chamadas de objetos. O termo “abstração” é comumente utilizado para designar a técnica de análise de um contexto do mundo real para criar representações no desenvolvimento de soluções informatizadas.

Para entender este paradigma, é imprescindível entender o conceito de objetos e classes.

5.1. OBJETOS E CLASSES

A **POO** (programação orientada a objetos) propõe uma análise do desenvolvimento de aplicações a partir do mundo real. A classe, neste caso, é uma abstração das entidades existentes no mundo real.

Conforme foi estabelecido nos requisitos funcionais do sistema, a aplicação precisa ser capaz de fazer o registro dos profissionais que desejam obter os equipamentos audiovisuais. Neste caso, podemos dizer que estes profissionais serão identificados dentro do código-fonte como sendo uma classe, que pode ser chamada, por exemplo, de cliente.

Esta classe, por sua vez, terá atributos pertencentes a ela, como por exemplo: nome, telefone, endereço, área de atuação, data de nascimento, e-mail corporativo e/ou pessoal.

O objeto, então, seria a instância derivada da classe. Ainda com o exemplo do cadastro dos profissionais que desejam obter os equipamentos audiovisuais, seriam exemplos de classes: Prof João, Profª Natália, Diretor Pedro, etc.

Também podem ser classes os tipos de equipamentos audiovisuais que estão no banco de dados do sistema (como projetores, notebooks, entre outros).

5.2. HERANÇA E POLIMORFISMO

A herança proporciona o reuso de software: novas classes são criadas a partir de outras já existentes, absorvendo atributos e comportamentos, e adicionando os seus próprios.

Dessa forma, classes criadas a partir de outras classes herdam os seus atributos, além de apresentar os atributos de sua própria classe.

Figura 4 – Exemplo de herança

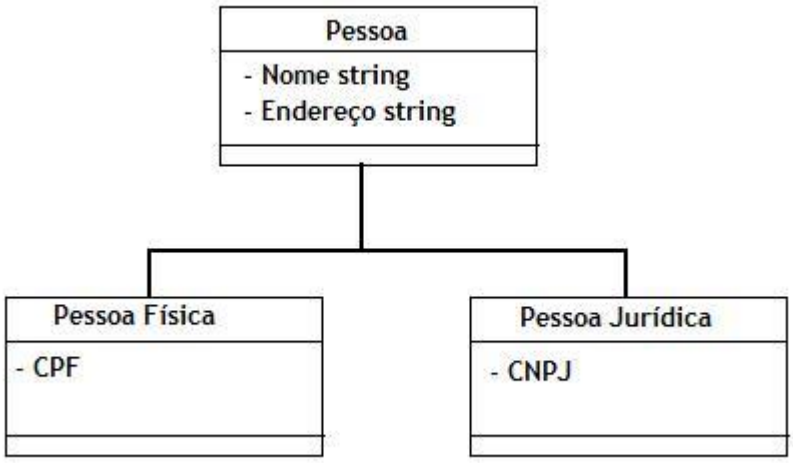


Imagem fonte: https://www.macoratti.net/11/05/oop_cph1.htm

É exatamente desta maneira que o conceito de herança funciona em um código orientado a objetos: a classe mãe – neste caso, pessoa – possui seus atributos (nome e endereço, ambas variáveis do tipo *string*). As classes criadas sob a classe mãe herdam estes atributos, além de adicionar os seus próprios atributos.

Por sua vez, o polimorfismo é um conceito inter-relacionado com o conceito de classes. De maneira semelhante ao de heranças, em que classes podem herdar os atributos de classes-mãe, o polimorfismo permite que objetos possam herdar métodos de outras classes. Acompanhe o exemplo na figura.

Figura 5 – Exemplo do funcionamento do polimorfismo

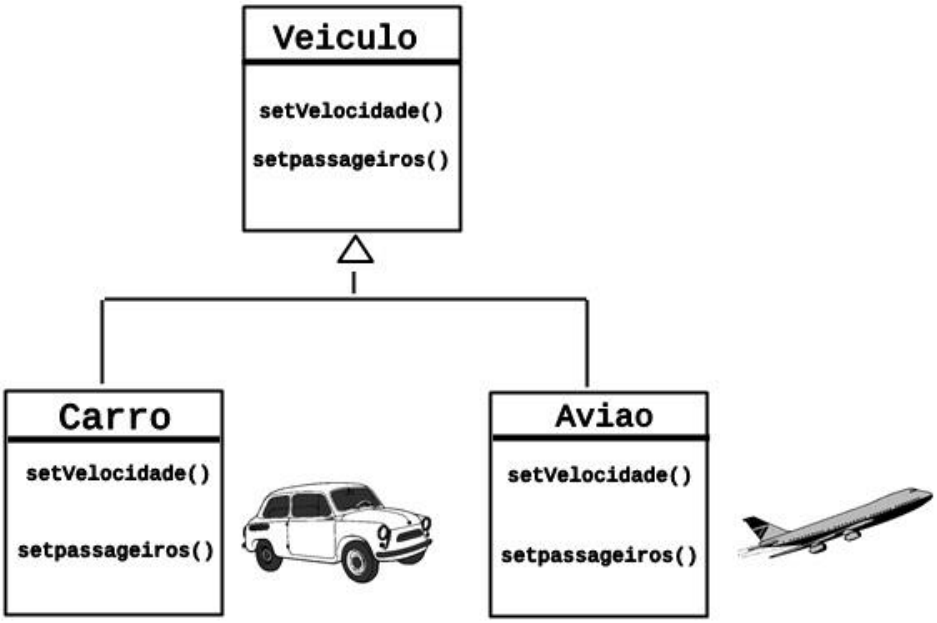


Imagem fonte: https://www.devmedia.com.br/sobrecarga-e-sobreposicao-de-metodos-em-orientacao-a-objetos/33066

Observe o exemplo: a classe veículo possui dois métodos, que são herdados pelos objetos criados sob ela. Dessa forma, esses métodos não precisam ser replicados consecutivas vezes, de maneira que o código-fonte é beneficiado. Podemos considerar que ambos os conceitos podem ser aplicados no nosso sistema, com os objetos de controle de entrada e saída dos equipamentos, ou de reservas.

6. O SISTEMA E A INTERFACE

Um apontamento necessário com relação à aplicação é sobre a importância da *interface* do sistema. Durante muito tempo, a *interface* teve sua importância minimizada no processo de desenvolvimento de software. Hoje, no entanto, é bem diferente.

Conforme sabemos, para o usuário, a *interface* é, de fato, o sistema. Uma vez que a *interface* é a única maneira que ele possui de se relacionar com o sistema, para ele não faz diferença o que há por trás dela.

Se uma *interface* de usuário for ruim, ele pode deixar de utilizar a aplicação. Por isso a importância de desenvolver uma boa interface.

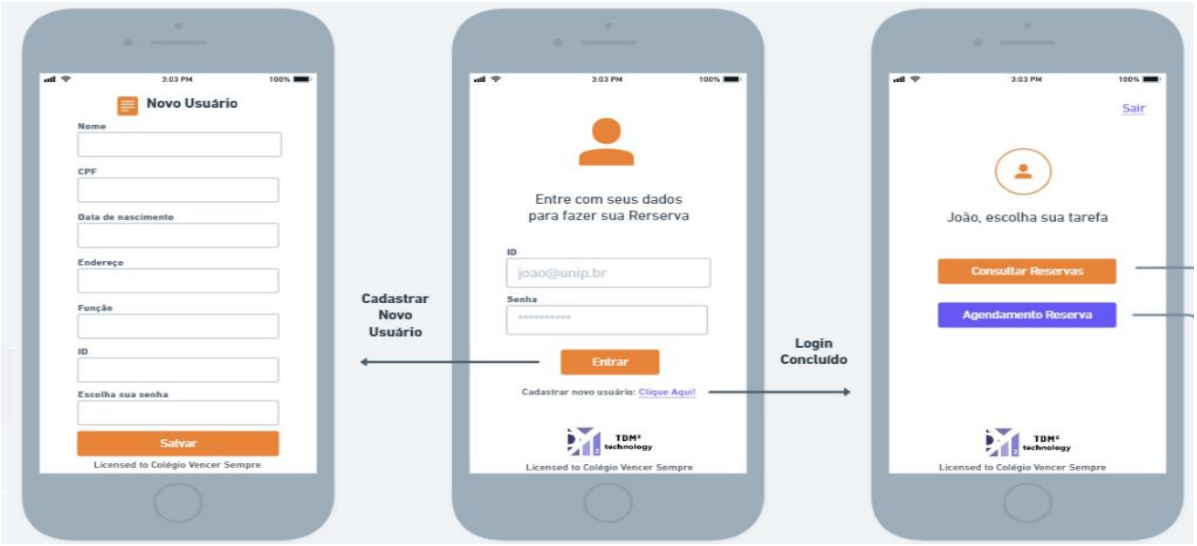
6.1. PROTOTIPAGEM E PROTOTIPAGEM EVOLUTIVA

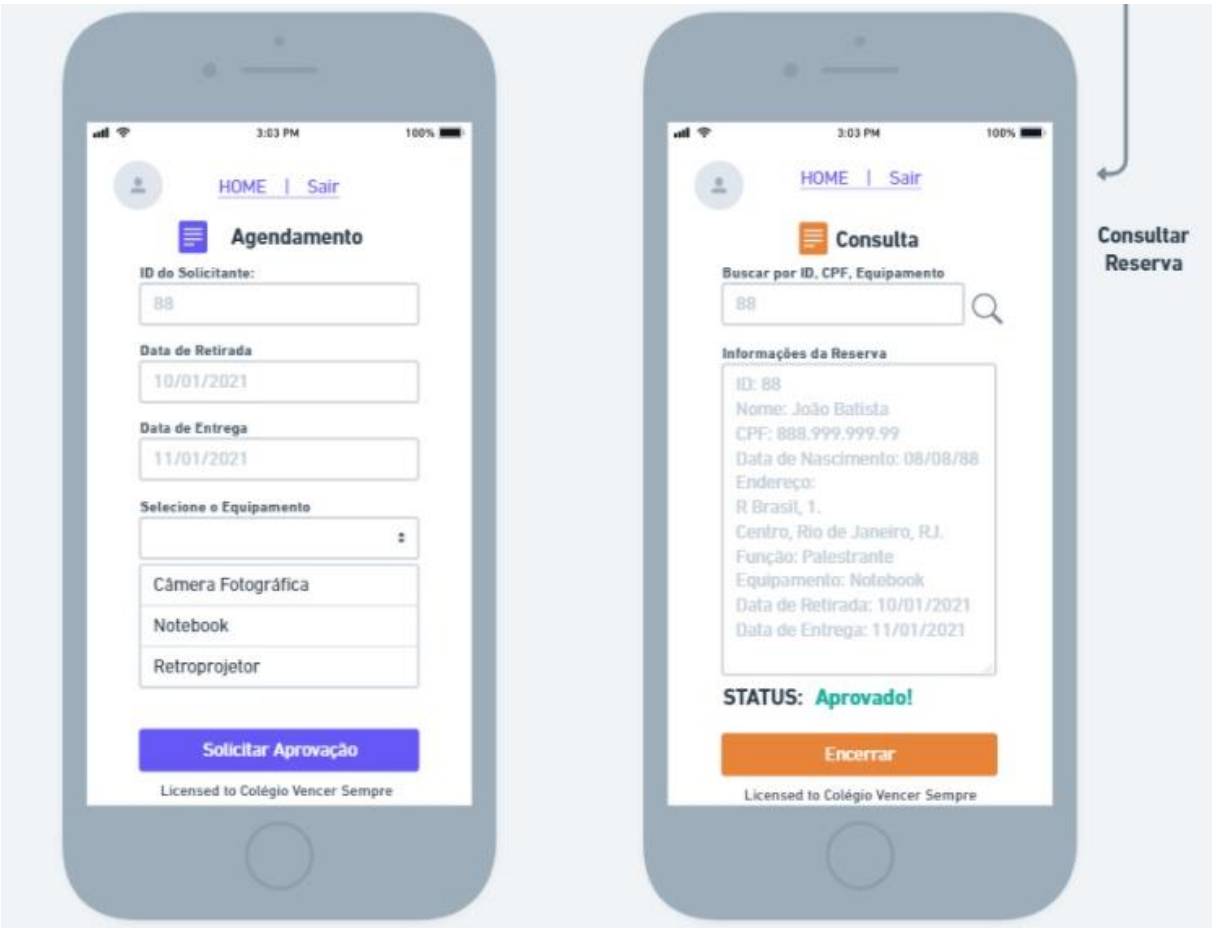
Para a tarefa de desenvolver uma boa interface, existem também alguns ciclos de desenvolvimento. Talvez o mais famoso deles seja a prototipagem evolutiva.

Neste ciclo de vida, algumas *interfaces* são desenvolvidas, afim de que todas sejam avaliadas e testadas, com o objetivo de obter erros e acertos dela, até a sua fase final.

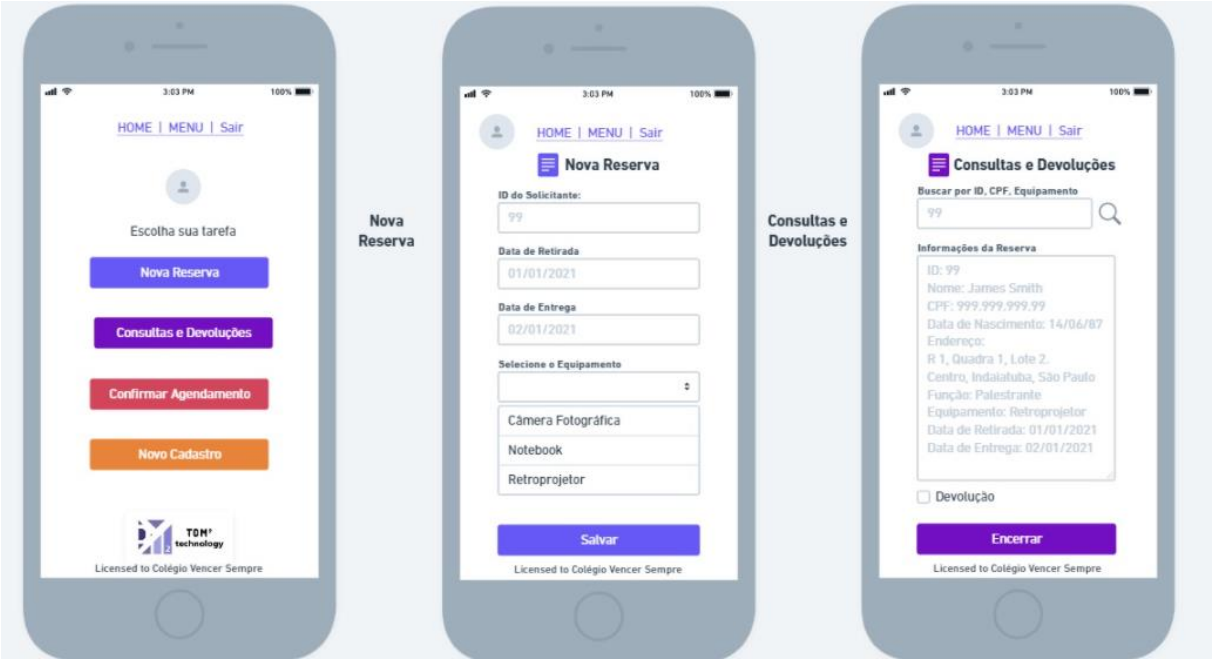
Nas figuras seguintes estão *interfaces* desenvolvidas pensando no projeto.

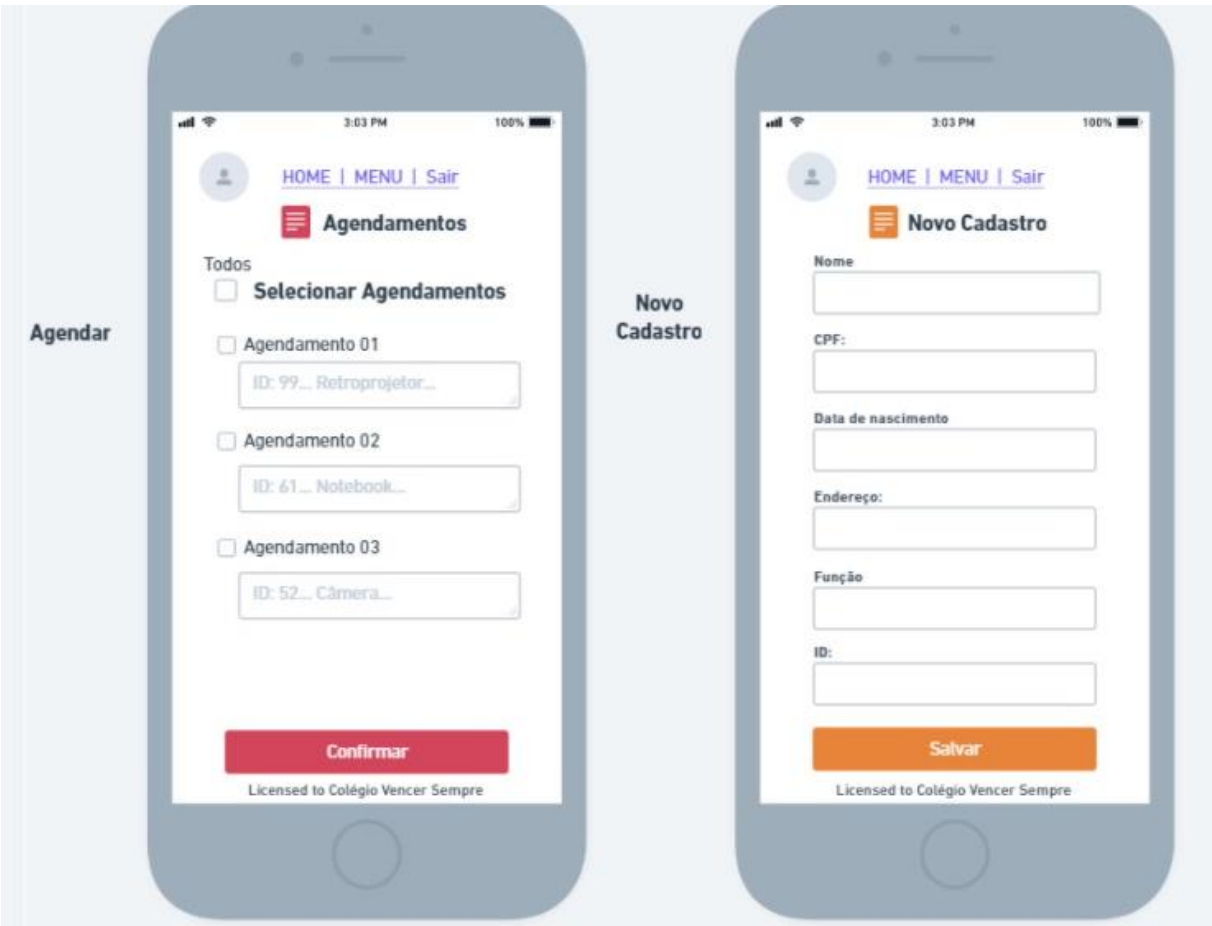
Figuras 7 e 8 – *Layout mobile* para acesso do usuário





Figuras 9 e 10 – *Layout mobile* do administrador





Figuras 11 e 12 – *Layout mobile* de acessos do administrador

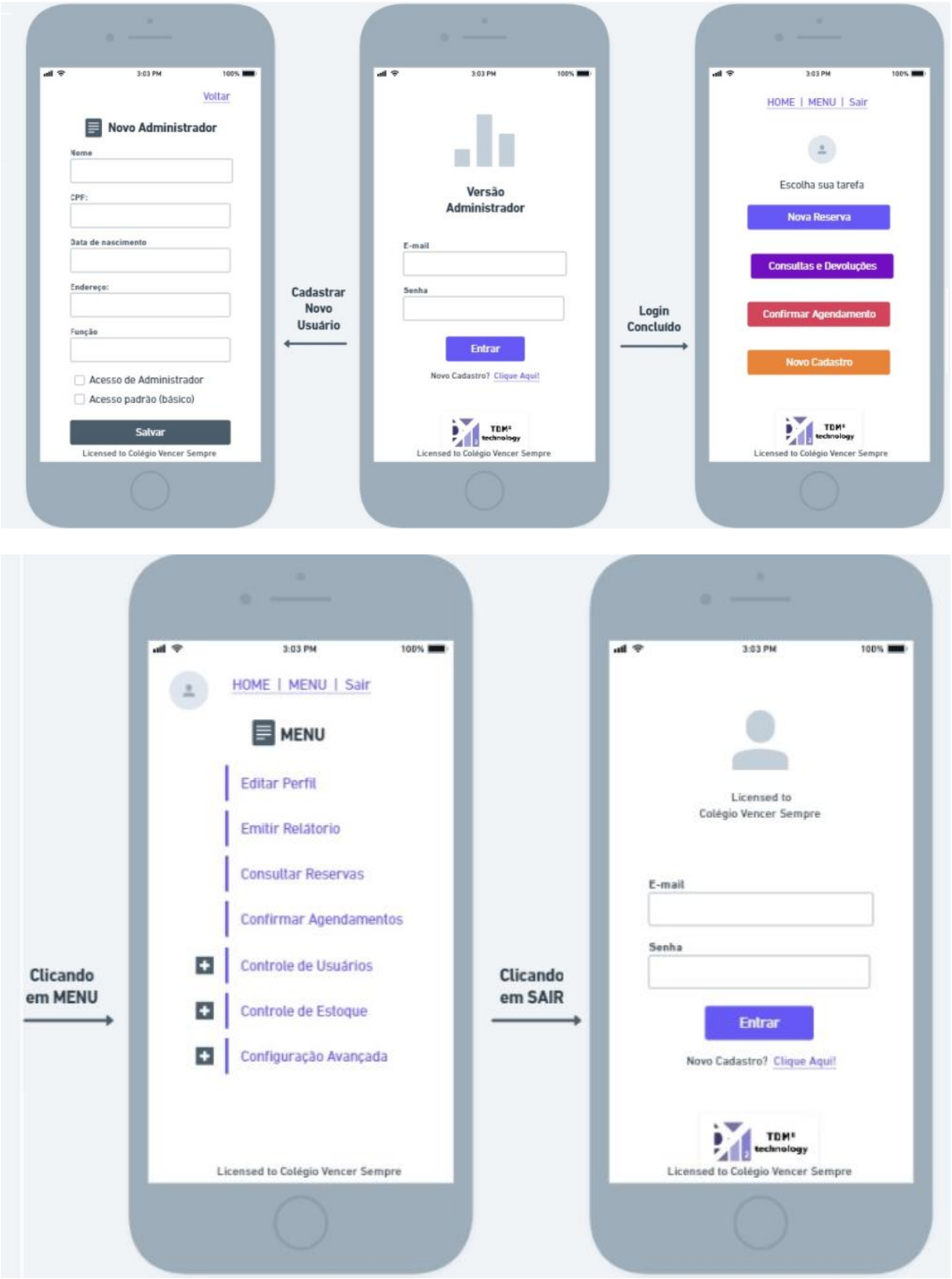
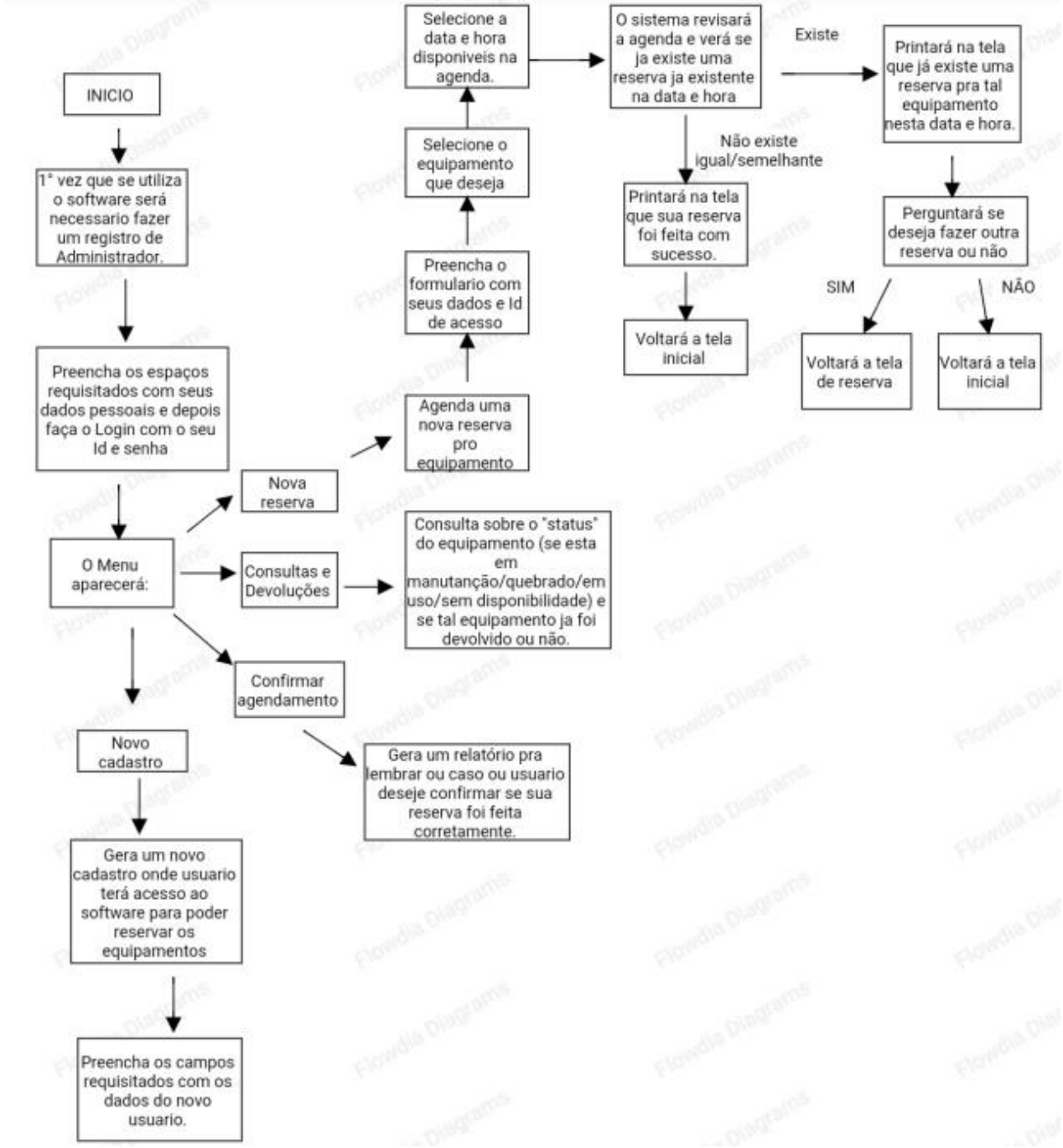


Figura 13 – Fluxograma exemplificando o esquema de entrada e saída de dados nas interfaces



Note que o roteiro de testes mencionado no **capítulo 4.1.2.** está aplicado às *interfaces*.

As propostas de *interface* mostram os diferentes acessos e *layouts* que um usuário possui no fluxo de dados do sistema, que é representado, logo em seguida, na **Figura 13**.

7. CONCLUSÃO

O desenvolvimento das *interfaces*, apresentado no último capítulo, marca o fim do nosso exercício neste trabalho. É importante dizer que a fase de desenvolvimento de *interface* em uma aplicação geralmente ocorre em paralelo ao desenvolvimento do sistema. Isto significa que, em muitas das vezes, o *back-end* ainda não estará **entregue**.

Os demais tópicos abordam outros assuntos que dizem respeito à rotina de uma equipe de desenvolvimento de *software*, além de questões correlatas com o mercado nacional de **TI** (tecnologia da informação), das questões acadêmicas acerca da **POO** (programação orientada a objetos), refletindo o conteúdo curricular deste segundo semestre de graduação do curso de análise e desenvolvimento de sistemas.

A importância destes exercícios, tão necessários aos desenvolvedores de *software*, conforme dito no **terceiro capítulo**, surge daquilo que muitos autores apontam como sendo o primeiro ciclo de vida de desenvolvimento adotado pelas empresas e profissionais de TI (tecnologia da informação), que funcionando de forma totalmente anárquica – fruto da situação que a área de prestações de serviço de TI (tecnologia da informação), que era àquela época prematura – acarretava em custos e atrasos para todos os *stakeholders*.

É por isso que, além de refletir o conteúdo curricular acadêmico deste segundo semestre de graduação, o texto apresentado também reflete os principais campos de estudo e de interesse da área da tecnologia da informação. E, aplicados estes conhecimentos ao exercício proposto pelo projeto, obtivemos este trabalho entregue.

REFERÊNCIAS

JUN, Guilherme. **Etapas de desenvolvimento de *software*.** ICMC Junior, 2021. Disponível em: https://icmcjunior.com.br/desenvolvimento-de-software/. Acesso em: 28/03/2022.

VICTÓRIA, Penélope. **Salário de programador: quais cargos mais bem pagos de 2021**. *Geek Hunter*, 2021. Disponível em: https://blog.geekhunter.com.br/salario-de-programador-cargos-em-alta-2021/. Acesso em: 01/04/2022.

Alff, Chico. ***Scrum Master* Salário em 2022: O que faz e quanto ganha um *Scrum Master*.** Análise de requisitos, 2022. Disponível em: https://analisederequisitos.com.br/scrum-master-salario/. Acesso em: 08/04/2022.

Salles, Filipe. **Quanto custa um *notebook*? Entenda os preços e encontre a melhor opção: Aprenda quanto custa um *notebook* e o que torna alguns modelos mais caros que outros.** Buscapé, 2021. Disponível em: https://www.buscape.com.br/notebook/conteudo/quanto-custa-um-notebook. Acesso em: 08/04/2022.

Silva, Daiany. **O que é o MPS-BR?** Blog da qualidade, 2013. Disponível em: https://blogdaqualidade.com.br/o-que-e-o-mps-br/. Acesso em: 10/04/2022.

Cunha, Fernando. **Requisitos funcionais e não funcionais: o que são?** Mestre da Web, 2022. Disponível em: https://mestresdaweb.com.br/tecnologias/requisitos-funcionais-e-nao-funcionais-o-que-sao/. Acesso em: 10/04/2022.

Vanessa de Almeida, Isabela. **Herança x Composição**. Blog da UFCG. Disponível em: http://www.dsc.ufcg.edu.br/~pet/jornal/junho2011/materias/recapitulando.html. Acesso em: 15/04/2022.