

Universidade Paulista – UNIP EAD

Projeto Integrado Multidisciplinar

Wenderson Francisco Gonçalves Dos Anjos - RA 2322858

VENCERSEMPRE -

Software de Empréstimos de Equipamentos

Trindade-GO

2024

Wenderson Francisco Gonçalves Dos Anjos - RA 2322858

VENCERSEMPRE -

Software de Empréstimos de Equipamentos

Projeto Integrado Multidisciplinar para
obtenção do título de Tecnólogo em Análise
& Desenvolvimento de Sistemas,
apresentado à Universidade Paulista – UNIP
EAD.

Orientador: Giovani Pereira dos Santos

Trindade-GO

2024

RESUMO

Este trabalho visa apresentar um sistema de empréstimo de equipamentos audiovisuais, incluindo suas funcionalidades em camadas: Dados, Apresentação e Negócios. Este projeto segue o padrão MVC, do início ao fim, elaborando requisitos funcionais e não-funcionais após todo o entendimento do software obtido pelas reuniões informais com o cliente. O software de gerenciamento de empréstimos, foi programado em C# utilizando conceitos de POO – Herança, polimorfismo e instanciação, Além de todas as prototipações de interface e casos de testes.

Palavras-chave: empréstimo; equipamento audiovisual; programação; engenharia;

ABSTRACT

This work aims to present an audiovisual equipment loan system, including its layered functionalities: Data, Presentation and Business. This project follows the MVC pattern, from start to finish, developing functional and non-functional requirements after fully understanding the software obtained through informal meetings with the client. The loan management software was programmed in C# using OOP concepts – Inheritance, polymorphism and instantiation, in addition to all interface prototyping and test cases.

Keywords: loan; audiovisual equipment; programming; engineering;

SUMÁRIO

INTRODUÇÃO	6
1. VERIFICAÇÃO ECONÔMICA DO PROJETO	7
1.1. Questionário de pesquisa do cliente	7
1.2. Entendimento do Software	9
1.3. Viabilidade econômica do projeto	11
1.3.1. Estimativas de prazos e investimentos	13
2. ANÁLISE DE REQUISITOS DE SOFTWARE	15
2.1. Requisitos funcionais do software	15
2.2. Requisitos não-funcionais do software	18
2.3. Especificação das regras de negócio	19
2.4. Diagramação da camada de dados	20
2.5. Prototipação da camada de apresentação	21
2.6. Casos e Roteiros de Testes	21
2.7. Metodologia das normas de qualidade	22
CONCLUSÃO	23
REFERÊNCIAS	24

INTRODUÇÃO

Diversos estabelecimentos que necessitam do campo da tecnologia, promove o contrato de softwares especializados em uma questão da área. Cada área tem suas próprias exigências, no entanto, uma boa parte delas pode precisar gerenciar recursos físicos quando estes se relacionam com os recursos humanos.

Pessoas frequentemente utilizam equipamentos para entretenimento ou outras funções. No caso deste software, uma escola de nível fundamental e médio precisou solicitar um sistema que gerencia empréstimos de equipamentos audiovisuais como televisores, DVDs, Microfones, Caixas de som, etc. No entanto, é difícil dizer do quão complicado pode ser este gerenciamento quando não há a tecnologia pra auxiliar.

Foi pensando neste contexto, que este trabalho nasceu para resolver conflitos em empréstimos de datas específicas. A missão deste trabalho é fornecer cadastros e agendamentos de empréstimos, assim como seus cancelamentos, gerenciando datas livres e listando para o monitoramento do usuário.

Primeiramente será elaborado um formulário de avaliação do que o software iria precisar, junto ao cliente numa reunião informal. Após isto, o entendimento do software será otimizado, possibilitando a criação de Mockups e Protótipos iniciais. Com a personalização dos protótipos para uma maneira mais realista, funcionalidade foram documentadas, assim como contextos de acessibilidade, manutenibilidade, dentre outros.

Prosseguindo por esse caminho, enquanto o desenvolvimento estava sendo aplicado, novos testes foram sendo escritos, após o planejamento de testes, constituindo um teste e desenvolvimento ágil, não só dos modelos de dados mas das interfaces e da integração.

Neste trabalho, diversas fases do desenvolvimento serão demonstradas, desde a verificação de prazos, investimentos e custos econômicos, até a análise de requisitos e desenvolvimento do software.

Boa leitura!

1. VERIFICAÇÃO ECONÔMICA DO PROJETO

Neste capítulo será abordado os passos iniciais para o projeto, que introduziremos a uma reunião walkthrough de maneira informal para coletar os requisitos e após isto, compreender o software em sua essência, para assim, analisarmos o real sentido da viabilidade econômica.

1.1. Questionário de pesquisa do cliente

Durante uma reunião hipotética com os nossos stakeholders, as seguintes perguntas foram efetuadas aos clientes, seguido de suas respostas anotadas pelos escrivães, para a confecção dos requisitos funcionais e verificação de viabilidade:

1. O que você deseja que seu software faça?

R = Eu gostaria de um software onde eu poderia inserir um novo empréstimo de equipamentos para professores e equipamentos já cadastrados no sistema, pois novos equipamentos podem entrar em nossa escola, assim como novos professores e isto facilitaria nos campos de busca. Preciso que meu software liste estes empréstimos e verifique datas disponíveis para não dar conflitos. Quero poder cancelar ou dar baixa em um empréstimo, assim como saber quais empréstimos foram baixados.

2. Você gostaria que todas estas funções tivessem em uma janela única? Ou em janelas separadas?

R = Eu gostaria que cada função tivesse uma janela separada, até mesmo para não gerar poluição visual e confundir nossos funcionários. Dividir as informações é a melhor maneira.

3. Cite pra gente cada função que você deseja – Se quiser detalhar um pouco mais, sinta-se à vontade:

R = Okay! Para a nossa escola, é necessário que o software tenha: O sistema de **agenda de empréstimos** onde ele insere o nome do professor, do equipamento, a data inicial e final de cada novo empréstimo, e antes de salvar, ele precisa saber se tem outra pessoa que agendou o empréstimo nessa data, retornando alguma mensagem de erro e forçando com que escolhêssemos outra data. Pra facilitar a busca de datas disponíveis, também preciso de um sistema que **liste as datas disponíveis** para empréstimo, onde eu poderei filtrar por uma data específica ou hora específica. Seguindo esse raciocínio de filtragem, gostaria que esse sistema de filtragem também tivesse em um sistema de **cancelamento de empréstimos** e em outro sistema de

fechamento de empréstimos também. Para cada um destes sistemas, eu gostaria de ver uma lista tabelada com os empréstimos atuais, onde eu possa selecionar uma linha e cancelar/fechar o empréstimo da linha selecionada, porém se for muitos empréstimos difíceis de encontrar, acredito que é melhor só aparecer a lista quando escolhêssemos filtrar pelo nome ou pelo item. Claro que antes de tudo isso, preciso ter a opção de salvar um equipamento novo, inserindo nome, quantidade e o tipo de equipamento. Também quero poder salvar um professor caso novos professores entrem na escola, sendo seu nome, sua disciplina e turmas que ele administra, pois quando eu for filtrar estes dados, eles já vão estar salvos e isso evita de ocorrer conflitos de nomes digitados errados e não encontrar estes dados. Também quero saber quantos equipamentos foram entregues e uma lista de empréstimos independente de filtragem.

- 4. E estas informações que você citou, como deseja acessá-las? Te darei duas opções: Prefere um menu no topo? Ou prefere que sejam botões em uma janela principal?**

R = Então, acredito que botões, de preferência bem grandes, sejam de grande valia para nossos funcionários, pois é bem mais fácil localizar as funções e não ter que ficar “procurando” cada função do software. A 2ª opção é a mais eficaz pra mim.

- 5. Agora introduziremos para as cores: Qual cor mais te agrada? Gostaria de uma interface mais colorida? Ou uma cor padrão com diversos tons semelhantes?**

R = A 2ª opção também é a mais atraente. Eu gosto muito do azul e seria bem legal se tivesse várias paletas de tons do azul, de preferência uma interface mais sutil, sem muitas letras, ou muitas cores separadas e diferentes, ou seja, gostaria de uma interface mais clara e limpa usando a cor azul.

- 6. Sobre o formato de armazenamento de dados: Nós temos duas opções inicialmente – O SQL que é um banco de dados, muito útil para armazenar uma grande quantidade de dados, bem eficiente nesses casos; E o formato de Arquivo de XML, que também facilita nas buscas e organização dos dados, permitindo também uma maior flexibilidade de alteração, porém é mais eficiente para pequenas quantidades de dados. Vai trabalhar com grandes ou pequenas quantidades?**

R = Olha, se você chama os dados de “Empréstimos” ou “Professores”, na escola não terá tantos dados assim, até porque a quantidade de professores não passa de 10 em muitas escolas, já em nosso caso as vezes temos menos e as vezes um pouco mais. Equipamentos também não temos muitos, e dá pra armazenar um só equipamento especificando sua quantidade. Já no caso de

empréstimos, é mais em relação a eventos, palestras, ou interação de professor com alunos, entretenimento, etc. São casos que não ocorrem todos os dias, talvez umas 3 vezes por semana, menos ou mais, no entanto, algumas semanas não há nenhum empréstimo, logo eu acredito que o XML será melhor em nosso caso, por se tratar de menores quantidades de dados.

7. Por último, prefere que suas janelas consomem toda a tela da área de trabalho? Ou janelas menores?

R = Como será um software de contexto mais simples e com poucas funções, até prefiro sim janelas menores, sendo a janela principal um pouco maior. Isso vai nos ajudar a movê-la para o lado e desenvolver outras atividades simultaneamente equivalentes a escola.

Finalizamos aqui a sessão de perguntas e registros de respostas. Vamos elaborar um protótipo inicial aqui no quadro-branco de acordo com todas as respostas para entendermos melhor o software. E assim, veremos se as especificações estão de acordo com o que você almeja. Logo em seguida, após o entendimento do software e a revisão das respostas, vamos conversar com os nossos desenvolvedores e compreender em quanto tempo cada um consegue desenvolver as interfaces e as funcionalidades previstas. Desta forma, lhe daremos uma estimativa de prazos e os custos.

1.2. Entendimento do Software

De acordo com o contexto e o cliente após a sessão de perguntas, o software conterá 8 funcionalidades, cada uma acessada por um botão específico em uma janela principal. Cada botão irá abrir uma nova janela, de tamanho pequeno, onde todas compartilham de uma semelhança na funcionalidade.

Na janela de agendamento de empréstimos, os campos **Nome, Equipamento, Data Inicial, Hora Inicial, Data Final e Hora Final e Data Final** serão as entradas de texto e após o clique do botão **Agendar**, as devidas validações de campos vazios serão feitas, assim como validações de nomes e datas via “Expressão Regular” para verificar se o formato dos dados está correto. Estes seis dados serão salvos em um arquivo XML chamado “Emprestimo.xml” que poderá ser recuperado por funções posteriores.

Na janela de cancelamento de empréstimos, apenas os campos Nome e Item serão adicionados para filtragem de dados. Os dados inicialmente não serão listados, até que o usuário decida “selecionar” o nome do professor em um campo ou o nome do equipamento em outro campo. A listagem ocorrerá logo abaixo, por meio de tabelas filtrando apenas os nomes/itens pesquisados. Cada linha da tabela poderá ser selecionada, e após o clique em **Cancelar**, os dados de empréstimo selecionado serão encontrados em “Emprestimo.xml” e apagados de lá.

Na janela de fechamento de empréstimos, os mesmos campos irão aparecer, como Nome e Item, podendo filtrar da mesma maneira como no caso do anterior, apenas aparecendo dados filtrados (Aqui está a semelhança de funcionalidades). Clicando no botão **Fechar** com a linha listada pré-selecionada, o dado irá ser apagado do arquivo “Empréstimo.xml”, porém, com uma exceção: O dado não pode ser apagado definitivamente, pois diferentemente do cancelamento, o fechamento é a baixa de empréstimos e o cliente necessita que equipamentos entregues também sejam listados. Desta forma, antes da exclusão, os dados serão “Movidos” para outro arquivo chamado “Entregues.xml”.

Na janela de equipamentos entregues, todos os dados de equipamentos entregues serão lidos do arquivo “Entregues.xml”, dados estes movidos pela função descrita anteriormente sobre o fechamento. No entanto, enquanto que as duas janelas anteriores só apresentam os dados na tabela se o usuário filtrar pelo nome e item, no caso de entregues, todos os dados já serão listados imediatamente ao abrir a janela, ainda permitindo que o usuário possa restringir as apresentações por meio dos campos de filtragem.

Na janela de listagem de empréstimos, a mesma situação ocorre como o caso anterior da janela de “Entregues”, no entanto, não lendo mais do arquivo entregues.xml, mas sim, do arquivo empréstimo.xml, onde tal arquivo conterá todos os dados salvos pela janela de agendamento (primeira janela descrita). Todos os dados serão apresentados imediatamente, restringindo também pela filtragem. Há uma diferença entre janelas de entregues e listagem, com as janelas de cancelamento e fechamento e esta diferença está na possibilidade de controlar estes dados. Nos dois primeiros casos, a função da janela é só apresentar e filtrar os dados, enquanto que nos dois últimos casos o usuário poderá “apagar” os dados por meio de um botão.

Na janela de dias disponíveis, ou de “Datas disponíveis”, serão listados imediatamente ao abrir a janela os dados de datas disponíveis, que serão controlados pelo arquivo “Livres.xml”. Este arquivo será todo lido pela janela (Chamado por uma das classes do modelo descritas mais tarde), no entanto, o gerenciamento de datas livres será feito pelo controlador, que é uma classe estática para manipulação destas datas. Visualizando as datas livres, o usuário terá a facilidade de incluir datas disponíveis no formulário de agendamento de empréstimos.

Por fim, teríamos mais duas janelas: **Cadastrar Professor** e **Cadastrar Equipamentos**. Em cadastro de professor, teremos os campos: **Professor**, **Disciplina** e **Turmas**. Cada um destes campos será verificado se está vazio, verificado a formatação de nome e depois salvos no arquivo “Professor.xml” após o clique do botão **Cadastrar**. Já no caso de cadastro de equipamento, os campos: **Equipamento**, **Tipo** e **Quantidade** serão utilizados para salvar no arquivo “Equipamento.xml” após o clique em **Cadastrar**. O campo de texto Quantidade é um tipo que só aceita números e contém uma seta para aumentar/diminuir números. O padrão mínimo é 1 e o máximo é 999.

Os dados de Professor.xml e Equipamento.xml serão utilizados em outros campos de texto, como nos campos de filtragem das janelas: cancelamento, fechamento, listagem, dias disponíveis e entregues. Esta utilização se dá por uma

apresentação em menu, onde o usuário apenas pressiona o ENTER e o campo de texto apresenta um menu logo abaixo com todos os professores ou equipamentos cadastrados, dispensando a necessidade de digitação. Estes campos de textos terão sua utilidade apenas para apresentação dos dados escolhidos.

Sobre a organização de elementos, tipos e padronização de projeto, as seguintes características serão consideradas:

- Todas as entradas de dados serão do tipo **TextBox** com tamanho pequeno ou médio e altura padrão.
- Em todas as janelas, botões e campos serão agrupados usando um **GroupBox**, assim como a janela principal que agrupa os botões em Agendamento, Disponibilidade e Cadastro, as outras janelas agrupa a parte de filtragem (com dois campos) e a parte de listagem.
- Todos os botões terão o mesmo layout, utilizando uma imagem de fundo e formato arredondado, assim como o fundo de janelas, com uma mesma imagem ou semelhante.
- A sessão de listagem dos dados será utilizada um **DataGridView** que permite visualizar os dados em tabela.
- O projeto seguirá o padrão MVC, sendo Model para camada de dados, View para apresentação e Controller para camada de negócios.

1.3. Viabilidade econômica do projeto

A viabilidade econômica do projeto é uma parte fundamental da construção de diversos projetos e deve ser gerenciado até mesmo antes do planejamento do projeto. Para compreendermos os prazos, investimentos, custos e os índices envolvidos, precisamos primeiro identificar os agentes econômicos que vão de certa forma se conectar a este projeto e a empresa de software.

Os agentes econômicos são todos aqueles que se interage de alguma forma com o desenvolvimento, gestão, investimento e obtenção do projeto, e são eles:

- **Gestor de TI:** O gestor é aquele que gerencia cada parte do projeto, estimando prazos e custos relativos ao software que está se desenvolvendo. Ele é o intermédio entre o cliente e o desenvolvedor, essa intermediação pode ser desde ao feedback e comunicação, até verificação da garantia se todos os prazos estão em dia. Seu aspecto econômico está além do que a parte de custos e investimentos financeiros, mas também da gerência de tempo e formas econômicas de se desenvolver um projeto.
- **Desenvolvedores:** Os desenvolvedores é um dos fatores de produção de uma empresa, o que pode ser relacionado a empresa de software hipotética desse presente trabalho, que visa entregar um software funcional para um cliente. Para o software existir, é preciso de desenvolvedores e papéis definidos, um exemplo: É que no contexto deste projeto, é necessário do Front-End para fazer

o lado “Visual” da aplicação, que são as interfaces e do Back-End que fará o lado dos “Fundos” do sistema, tudo que funciona internamente. Os desenvolvedores como partes integrantes da empresa, participam de forma ativa no processo de investimento, uma vez que seus salários se integram no investimento do projeto.

- **Fornecedores:** O fornecimento de ferramentas é essencial para a produção do software. Na maioria dos casos, as empresas utilizam ferramentas terceirizadas para agilizar no desenvolvimento e possibilitar a produtividade de demais projetos. Desta forma, os fornecedores é uma peça no âmbito de economia, todavia, o retorno de investimento pode voltar diretamente para eles – os fornecedores.
- **Investidores:** A área de investimentos é um campo crucial para qualquer tipo de produção, portanto, quem recebe também irá produzir, isto é, quem paga por um software funcional está investindo, que por sua vez pode utilizá-lo para produzir e quem fornece o software – no caso da empresa – também estará depositando seus investimentos financeiros na produção, para assim obter o retorno e reinvestir em projetos terceirizados de fornecedores. Logo, os investidores podem ser: O governo, uma outra empresa, o diretor da escola e o próprio empresário que está fornecendo este trabalho.
- **Usuários:** Mesmo que um determinado tipo de usuário final não interage “diretamente” com os papéis econômicos de uma empresa, é possível que o usuário seja o mesmo que o investidor, “aquele que usa pode ser o que paga”, no entanto, o seu papel econômico neste projeto e demais outros, equivale a feedbacks e identificação de melhorias, que por sua vez abrem margens para manutenções de qualquer tipo (Adaptativas, evolutivas, etc.), nos quais estas manutenções serão um dos fatores para os próximos lucros e capitais da empresa, ou seja, um processo econômico.

Compreendendo o papel fundamental de cada contribuidor na economia da empresa, é possível mensurar os indicadores, observar prazos e custos de acordo com o investimento, e assim, chegar a uma conclusão de lucro e rentabilidade. Portanto, veremos as estimativas de prazos já calculadas pelos dois desenvolvedores hipotéticos (Front-End e Back-End), após a primeira reunião com o cliente, e a identificação dos custos que cada um pode gerar, subtraído do retorno total.

1.3.1. Estimativas de prazos e investimentos

Iniciando pelo prazo de construção, de acordo com o nível de experiência de cada desenvolvedor, considerando que são Juniors, cada janela levaria um dia para ser finalizada, incluindo a construção dos campos de textos, botões, a própria janela e suas validações. Da mesma forma, em paralelo, o desenvolvedor Back-End faria as funções no background para efetuar ações de acordo com o clique, controlando dados em XML. Portanto, considerando que temos 9 janelas ao todo - Janela principal, agendamento, cancelamento, fechamento, listagem, datas disponíveis, entregues, cadastro do professor e cadastro do equipamento – o prazo estimado inicial seria de 9 dias.

Com a exigência do gestor de TI para otimizar o tempo e reduzir os prazos, para assim reservar o tempo para integração das partes do software, testes e implantação, o desenvolvimento pode ser reduzido a 1 semana (7 dias) aproximadamente, quando há um gerenciamento e planejamento melhor deste tipo de software. Logo, podemos considerar que enquanto o desenvolvedor Front-End desenvolve os protótipos de interface, o desenvolvedor Back-End focaria nos requisitos funcionais e classes equivalentes, já aplicando os primeiros modelos prototípicos iniciais no sistema e elaborando os casos de testes, ou seja, a documentação pode ter uma simultaneidade com o desenvolvimento, incluindo a presença periódica do cliente para avaliar o que está sendo desenvolvido, evitando possíveis retrabalhos.

Já em relação aos custos, precisamos entender que de acordo com o prazo, uma rentabilidade é satisfeita ou não, o que significa que, cada intervalo de desenvolvimento terá seu valor monetário em custos e em retornos. Primeiramente, antes de identificar os custos, é necessário saber “quanto” vale cada função e cada elemento. Logo abaixo serão listados os requisitos do negócio, isto é, o preço de cada elemento variando em função da complexidade:

- Entradas de usuário como campos de textos, botões e outros, na empresa teria seu valor mínimo igual a R\$400,00, e a depender de customizações, poderia variar até R\$600,00.
- Elementos estáticos que geralmente não necessitam de interação do usuário, terá seu valor reduzido, como textos estáticos, caixas de grupo, etc. Sendo igual ou aproximadamente a R\$250,00. Este valor pode variar sutilmente a cada conjunto predefinido de caracteres e/ou tamanhos.
- Menus e elementos adicionais em outros elementos podem acarretar em uma alternância de valores. Menus mais básicos podem somar +R\$100,00 ao valor original.
- Cada janela também poderá ter um valor, sendo o mínimo R\$1000,00 para uma janela pura e de tamanho reduzido. De acordo que o tamanho aumenta, este valor pode sofrer alterações de +R\$100,00 a +R\$200,00. Imagens em janelas

somam um valor de +R\$500,00, onde este valor pode ser dividido quando a janela é menor.

- Não apenas interfaces gráficas, mas funções em geral cada uma pode ter um valor específico, que se torna mais variável do que as janelas. Pois cada função contém uma quantidade de linhas e cálculos matemáticos, como também chamadas de outras funções e manipulações de arquivos. Cada funcionalidade pode ser mensurada por um valor. Como este valor é extremamente dinâmico, será dado um padrão de valor a R\$250,00 por cada algoritmo desenvolvido, isto é, por cada função, desde que seja maior que 2 linhas e que tenha cálculos matemáticos ou chamadas de funções externas.

Considerando estes fatores monetários, a janela principal pelo seu tamanho médio com imagem de fundo custará R\$1700,00 reais + 8 janelas pequenas também com imagens custando cada uma R\$1450,00. O valor total se dá pelo cálculo: $1.700 + 1.450 * 8 = 13.300$. Temos no total 8 botões parcialmente customizados da janela principal ($500 * 8 = 4.000$) + 3 botões extras correspondentes a janela de agendamento, cancelamento e fechamento ($500 * 3 = 1.500$) + 16 campos de textos não-customizados de todas as janelas ($400 * 22 = 8.800$), pois o menu é parte dinâmica do algoritmo e pode ser adicionado separadamente, a soma total do valor inicial com todos estes valores daria o resultado de $13.300 + 4.000 + 1.500 + 8.800 = \text{R\$}27.600$ reais.

De acordo com a camada de dados, que controla os arquivos XML, temos classes que herdam de uma mesma classe, contendo praticamente os mesmos algoritmos com algumas variâncias, então os algoritmos principais + os algoritmos diferenciados serão incluídos no orçamento sem contar de modo geral. Sendo eles 11 métodos diferenciados/principais do Model, 3 métodos do Controller e 16 métodos do View, totalizando $(11 + 3 + 16) * 250 = 7.500$. Portanto, o valor final do orçamento seria $27.600 + 7.500 = \text{R\$}35.100$ reais.

Talvez esta não seja uma realidade exata para empresas de pequeno porte que estão iniciando, ainda mais para um software mais simples em vista de muitos outros, mas para empresas de porte médio, por natureza, é comum ver valores chegarem a estas faixas. Logo, se considerarmos dividir todos os preços de elementos pela metade, desconsiderando quaisquer outros requisitos que já foram adicionados, o valor total do software ficaria por R\$17.550. A questão é: Este valor dividido é satisfatório para uma empresa de dois desenvolvedores? Ou melhor, de acordo com os custos, é viável economicamente obtendo um valor abaixo disso no período de 7 dias? Para responder estas perguntas, devemos compreender os “Custos” que foram gerados ao produzir este software, que também se relaciona aos investimentos. Daí em diante, teremos o valor líquido.

Um computador ligado 8 hrs por dia consome em média 40,0kwh/mês de energia. Logo, por ser dois desenvolvedores em trabalho, teríamos 80,0kwh/mês. Se a tarifa de energia for por 9,80 a cada 100 quilowatts-hora, teríamos aproximadamente R\$ 8 reais de custo em 7 dias. Seus salários por ser Junior, numa empresa porte pequeno, seria de R\$2.500,00 + benefícios que somando poderia chegar a R\$

3.000,00 em média. Sem contar os investimentos em softwares e licenças, água consumida no trabalho, e demais outros investimentos, o lucro líquido da empresa seria de $17.550 - ((3.000 * 2) + 9) = 11.541$, o que faz os custos equivaler a 34% do retorno bruto. Com outros custos/investimentos, facilmente o valor subiria pra 50% e o orçamento do software não traria viabilidade econômica. Porém, com o valor original R\$35.100 reais, teríamos custos mínimos de 17%, o que seria 83% de lucratividade líquida da empresa na confecção deste software.

Outro fator relevante é o retorno de investimento a longo prazo, é possível que o software continue gerando receita pela sua existência? A resposta é sim. Se o estado atual permitir que o sistema abra margens para manutenções evolutivas e o diretor da escola/investidor estiver disposto a pagar mensalmente por manutenções e crescimento do software, o investidor poderá parcelar todo o valor aplicando um valor a mais nas parcelas relativo a manutenção mensal. O Software Vencer Sempre contém possibilidades de crescimento, como: Emissão de relatórios semanais, exclusão de professores e equipamentos, gerenciamento de datas atrasadas, entre outros. Com o estado do código atual, facilmente os desenvolvedores poderão implementar sem tanto consumo de tempo, o que levaria a uma maior viabilidade econômica.

2. ANÁLISE DE REQUISITOS DE SOFTWARE

A partir daqui uma série métodos para requisitos serão trabalhados no desenvolvimento do software. Estes seguem um cronograma preparado que vai desde a especificação das funcionalidades, até os casos de testes e prototipação.

2.1. Requisitos funcionais do software

Os requisitos funcionais é uma maneira de avaliar “O que” o sistema deve fazer, isto é, quais são suas características de acordo com entradas, processamentos e saídas esperadas em diferentes condições. Abaixo serão listados todos os requisitos funcionais do sistema:

- **Cadastrar professores e equipamentos:** O sistema deve permitir que dados essenciais de professores, como: Nome, Disciplina e Turmas; sejam inseridas em campos de texto do tipo **TextBox**. Assim poderão ser salvos ao clicar em um botão do tipo **Button** em um arquivo XML Professor.xml. Da mesma forma os equipamentos, que requer os dados Equipamento, Tipo e Quantidade, dos tipos **TextBox**, sendo este último como formato numérico (Tipo **NumericUpDown**), salvando os dados pela classe XDocument em um arquivo Equipamento.xml. Antes do registro, no caso do Professor, o campo Nome deve ser verificado via Regex, que verifica se o campo contém nome e

sobrenome separados por espaços, se não, ele exibe um erro na tela. Da mesma forma se algum dos campos estiverem vazios, em ambos os casos (Professor e Equipamento), um erro deve ser apresentado em MessageBox (Consultar Apêndice A). Os métodos envolvidos para esse fim são: Professor.Registrar() e Equipamento.Registrar(), após as instâncias destas classes.

- **Agendar empréstimos em datas livres e poder visualizá-los:** O sistema inicialmente deve recolher os dados Professor, Equipamento, Data Inicial, Hora Inicial, Data Final e Hora Final, todos do tipo **TextBox** e verifica se estão vazios, se estão, apresentar um erro em MessageBox (Consultar Apêndice B), se não estão, prosseguir verificando o formato de datas e horas utilizando Regex. Tanto a data e a hora, como o Nome, passam por expressões regulares diferentes e se estes testes não passarem, também apresentar um erro, senão, verificar a disponibilidade de datas pelo Controller Agendamento.VerificarDisponibilidade() que é uma classe estática que vai retornar verdadeiro se a data inicial e final passadas como parâmetro estiverem livres para agenda. Se retornar falso, uma mensagem é apresentada dizendo que as datas escolhidas não são possíveis, se não, a instância de empréstimo é criada usando os parâmetros fornecidos pelo usuário e após isto, executado o método Empréstimo.Registrar(). Os dados são salvos no arquivo Empréstimo.xml utilizando a classe XDocument. Em outras janelas, o usuário deve ser capaz de filtrar e listar estes dados, lidos do mesmo arquivo, usando a classe DataGridView e o método Coletar();
- **Método estático para gerenciamento de datas livres:** Agendamento é uma classe estática do controlador que comporta métodos estáticos como: VerificarDisponibilidade() e AtribuirDisponibilidade(), onde este último recebe a sobrecarga (overloading), um incluindo parâmetro de Data inicial e final e o outro sem parâmetros. O 1ª método de verificação, executa Empréstimo.ColetarLivres() para capturar todos os dados XML do arquivo Livres.xml e em seguida, realiza um loop filtrando cada Item desse “Array” de elementos XML. Para cada item, é verificado se a data final contém o termo “Data Indefinida”, se sim, ele identifica se as datas passadas estão “A partir” do intervalo de data inicial livre, caso esteja, ele apenas verifica se data inicial é diferente da data inicial livre, ou é igual, no 1ª caso um novo elemento XML é adicionado com a data indefinida, alterando a data indefinida anterior para uma data inicial, no 2ª caso a data inicial livre é modificada com a data final do usuário. Porém, se a data passada não está a partir do intervalo inicial, ele será menor, então ele continua o loop a procura de novas datas, definindo horarioLivre para “False” temporariamente (esse valor pode ser retornado no final). No entanto, se a data final não contém o termo “Data Indefinida”, esta será uma faixa de valores a ser verificados, tudo o que o código faz é identificar se as datas passadas estão “Entre” esta faixa, podendo utilizar quase a mesma lógica de modificação ou criação de elemento de XML a depender da diferença ou igualdade das datas, da mesma forma, atribuir modificações em ordens

inversas. Em um dos casos, a data livre atual pode ser removida, se caso não houver mais uma faixa livre entre elas. Se em todas verificações do loop não houver uma faixa disponível, a variável `horarioLivre=false` é retornada no fim. `AtribuirDisponibilidade()` pode registrar uma data livre padrão do dia de hoje ou atribuir uma nova data livre imediatamente.

- **Empréstimos devem ser cancelados e fechados, os fechados podem ser monitorados:** Assim como os outros elementos, o cancelamento e fechamento dependem de operações da interface. O cancelamento por exemplo, assim como o fechamento, atribui um evento de tecla para um campo de texto (Nome ou Item) em janelas de listagem, caso for ENTER, é inicializado um menu do tipo **ContextMenuStrip** que é aberto e listado dados vindo da coleta de Professor ou Equipamento, isto é, `Professor.Coletar()` ou `Equipamento.Coletar()`. O conjunto retornado em uma das classes, será listada no menu e atribuída evento de clique (sendo do mouse ou teclado), e este evento vai utilizar a classe **DataGridView** para listar em formato de tabela dados específicos pelo método `Emprestimo.Verificar()`, onde a instância deve conter como argumento o “texto do item”, que pode ser o nome do equipamento clicado ou o nome do professor. Em cada linha da tabela filtrada, será atribuída o ID de cada elemento XML vindo de `Emprestimo.xml`. Este ID será utilizado pelo evento de clique do botão Cancelar ou Fechar, que instanciará novamente a classe `Emprestimo` com o ID como parâmetro, chamando o método `Cancelar()` ou `Fechar()`. No método `Cancelar()`, o `Emprestimo.xml` é carregado e o dado de registro cujo ID for o especificado será retornado, logo em seguida, os atributos Data Inicial e Data Final da classe `Emprestimo` pegará os valores destas mesmas tags neste registro retornado e `AtribuirDisponibilidade()` do Controller é chamado pra armazenar imediatamente a nova data livre (a data cancelada). Após isto, o método `Excluir()` é chamado. Já no caso do método `Fechar()`, o mesmo procedimento de `Cancelar()` é feito, com a exceção de não excluir o dado imediatamente após a atribuição de disponibilidade de data livre e passar não só as datas, mas também o nome e o equipamento, pois estes dados serão salvos para armazenar em `Entregues.xml`. Após armazenar neste arquivo usando `Registrar()`, o dado finalmente é excluído usando `Excluir()`.

A partir destes requisitos, conseguimos compreender a essência do software, avaliando formas alternativas de se recorrer a um mesmo dado, usando polimorfismo e herança. As classes da camada de dados, relativos ao Model, como: `Emprestimo`, `Professor` e `Equipamento`; Herdam características da classe `Cadastro` por ter as mesmas funcionalidades principais: `Registrar()`, `Excluir()`, `Verificar()`, desta forma, sobrescrevendo estes métodos contendo suas próprias lógicas, que podem ser similares entre elas ou não.

Da mesma forma, o método `Verificar()` é sobrecarregado, pode ter definições com parâmetros ou se parâmetros e similar a isto, os construtores de instanciação são sobrecarregados com maiores tipos de alternativas. É neste contexto que encontramos o conceito de polimorfismo para cadastro de dados. Não só os métodos

herdados ou da camada de dados, mas Coletar() é um exemplo que não herda da classe Cadastro() sendo uma característica específica das classes, mas que é sobrecarregado com parâmetros, e seguindo esta lógica, a camada de negócios utiliza sobrecarga no método AtribuirDisponibilidade, podendo ter até duas formas de realizar esta atribuição. Em seguida, veremos sobre os requisitos não-funcionais do software.

2.2. Requisitos não-funcionais do software

Enquanto que os requisitos funcionais se preocupam em “O que” o sistema deve conter para funcionar, que são suas funcionalidades, os requisitos não-funcionais especificam “Como” o sistema deve se comportar, que são seus comportamentos. São características mais difíceis de ser medidas do que os requisitos funcionais, no entanto, elas ainda podem ser verificadas. A seguir serão listados os requisitos não-funcionais do software:

- **Manutenibilidade:** Este software foi escrito pensando na facilidade em dar manutenção em tempos futuros. O software “Vencer Sempre” prioriza a extensibilidade, códigos corretivos e métodos evolutivos. Cada método ou classe contém um nome específico condizente com a realidade daquele método/classe, dividindo as informações e funcionalidades em hierarquias. Um exemplo é classe Emprestimo que pode realizar empréstimos, e seu método mais evidente para isto é o Agendar() que utiliza do método Registrar(), da mesma forma, para cancelar um empréstimo, é utilizado o método Cancelar() que utiliza o método Excluir(), isto é, apenas lendo o nome do método é possível saber o que ele faz. Além de toda herança, cada início de método foi comentado com “Sumários” que podem ser documentados, descrevendo o que a função faz e quais são seus parâmetros. O código está implementado de uma forma que novas extensões podem ser criadas facilmente, assim como adaptações/melhorias, incluindo correções. Um exemplo que é passivo de correção, são classes distintas das interfaces gráficas que executam quase a mesma lógica em um mesmo método, contendo diferenças bem sutis, isto significa que há uma possibilidade de ser dividido em uma classe nova que trata destas questões similares. Outro exemplo são a variedade de métodos sobrecarregados, que seguindo a mesma lógica, novas funcionalidades podem ser implementadas usando uma combinação destes métodos.
- **Usabilidade:** A usabilidade é um conceito amplo e muito complexo de medir ou verificar, muitas vezes dependendo do usuário para avaliar como o software “Reage” ou “Ensina” ele a utilizar. Alguns destes conceitos foram usados na engenharia desse software, que é a questão de ter elementos posicionados e agrupados de uma maneira que o usuário rapidamente pode ver e identificar, assim como saber manipular estes elementos. Além de todas as imagens chamativas porém sutis e leves que combinam com as nuances de cores em azuis da interface, o software pode agir como “professor” do usuário, mostrando

pra ele quais são as funcionalidades do sistema, suas semelhanças, diferenças e utilidades que elas apresentam durante o ciclo de atividade. Cada botão tem seu espaço, seu nome e sua janela. Cada janela compartilha do mesmo pensamento, mas agindo em prol de objetivos diferentes. Será que o modelo mental apresentado aqui é o mesmo modelo transmitido ao usuário neste momento? É um caso a se pensar. Já na questão de menus em campos de texto, isso evita que o usuário possa errar ou esquecer dados relativo aos nomes dos professores ou nomes exatos dos equipamentos. Com a atribuição de menus, essa possibilidade é eliminada, devendo o usuário simplesmente selecionar o dado e visualizar os filtros na tabela correspondentes ao mesmo dado. Os formatos de fontes de texto de toda a interface também priorizaram a leitura, usando Arial tamanho 12.

- **Acessibilidade:** Por último, mas não menos importante, em cada elemento construído da interface, existem as Strings de acessibilidade, que são atribuídas juntamente com seu nome de acessibilidade e tipo de elemento. Estes dados são úteis para softwares que auxiliam deficientes visuais, no qual é necessário “ouvir” narrações destes softwares de acordo com as Strings especificadas. A String de descrição contém uma maneira rica, porém objetiva de descrever o componente, enquanto que a String de nome da acessibilidade é uma maneira direta de identificar o componente, usando um linguajar similar ao contexto de surdos-mudos ao se comunicar em libras. Exemplo: “JanelaListaEntregues”, “BotãoCadastroProfessor”, etc.

Cada um destes requisitos não-funcionais pode ser verificado e inspecionado no código-fonte do software, disponível em uma das pastas deste trabalho. O código pode ser manutenível visando uma maior rentabilidade econômica na criação de novas funcionalidades de forma rápida. O diretor ou secretário da escola facilmente poderá aprender a utilizar a interface para agendar um empréstimo no primeiro dia, além de ter respostas rápidas na utilização. E se caso um destes funcionários tiver um tipo de deficiência auditiva ou visual, o software Vencer Sempre não vai restringi-lo, possibilitando usar novos softwares para auxiliá-lo a identificar os elementos da interface e utilizar normalmente sem dificuldades.

2.3. Especificação das regras de negócio

As regras de negócio é uma parte fundamental do software. Elas se relacionam com as restrições e condicionais que as partes principais do software fornecem ao usuário. Estas condições são tratadas em toda a parte do software, porém, levando em consideração um contexto em comum. A lógica por trás do sistema, que realiza seus diversos cálculos para processamento de entradas e saídas resultantes adequadas, é o resultado e o objetivo das regras de negócio em um software.

De acordo com estas premissas, o software contém 3 regras principais que subsidiam todo o funcionamento interno do software, desde a agendamentos,

gerenciamentos até as listagens. Estas 3 regras permitem avaliar condições de processamento e apresentar mensagens ao usuário.

A regra RN01 (Apêndice A), especifica a base do software, que é o registro dos primeiros dados, sendo eles professores e equipamentos. Tais dados registrados vão facilitar no uso do software pelo usuário, restringindo possíveis erros de esquecimento de nomes ou evitando conflitos entre nomes com caracteres diferentes.

A regra RN02 (Apêndice B), determina a capacidade do software pensar inteligentemente na hora de gravar novos empréstimos. Ele verifica se uma data está disponível, se estiver, ele registra informações de empréstimos, porém, em caso contrário, uma nova data deve ser procurada pelo usuário. Esta é uma outra forma de evitar conflitos entre agendamentos de uma mesma data.

A regra RN03 (Apêndice C), possibilita o usuário a cancelar um empréstimo que foi agendado ou fechá-lo em caso de entregas antes ou no dia final, liberando assim os recursos de agendamento para os próximos professores. O sistema também deve ser capaz de listar os dados filtrados para verificação pelo usuário.

Observando estas 3 regras, compreendemos que o intuito principal do gerenciamento deste software é evitar problemas e conflitos dentro do sistema ou entre professores. O sistema deve saber em qual “Lugar” cada peça deve ser encaixada.

2.4. Diagramação da camada de dados

O diagrama de classes da camada de dados (Model) compreende um modelo arquitetural de um sistema de tratamento de dados, no qual é utilizado conexões entre classes e que por sua vez, estas classes irão se transformar em objetos instanciados pela camada de apresentação.

As 3 classes apresentadas (No Apêndice D), demonstra os atributos e métodos das classes: Empréstimo, Professor e Equipamento. Cada uma destas classes tem uma relação de dependência da Interface IXMLControl que fornece a especificação de dois métodos: XmlCreate() e XmlAppend(); Ambos os métodos manipulam os arquivos XML, no entanto, o primeiro cria o arquivo inicialmente, enquanto que o segundo acrescenta dados em um arquivo existente.

Todas as classes utilizam estes métodos declarados, sendo a base para qualquer operação posterior. A classe Professor tem uma semelhança exata com a classe Equipamento relacionado aos seus métodos, com a diferença dos construtores e alguns atributos específicos equivalentes aos campos de textos de cada um. Esta semelhança compartilha uma característica padrão de herança com a Classe base – Cadastro.

Não só os métodos, mas os atributos também são sobrescritos entre estas classes, enquanto que a classe Empréstimo compartilha dos mesmos recursos de herança e polimorfismo, no entanto, contendo mais métodos específicos.

2.5. Prototipação da camada de apresentação

A interface antes de ser realmente projetada, ela precisa ser idealizada, desta forma, a interface contém um processo gradual de construção, iniciando por um modelo mais básico inicial, até um modelo mais realista e por fim a implementação e otimizações.

Este modelo mais básico (Que consta no Apêndice F) se trata da prototipação de interface de baixa-fidelidade, também conhecido como “Mockups”. Estes modelos são desenhos iniciais das janelas, especificando posições de elementos apenas para se ter uma ideia base. Considere como um “Esqueleto” do projeto de interfaces, podendo ser desenhado em papel ou softwares de desenho.

Já entrando no nível de desenvolvimento, é possível utilizar softwares específicos para elaborar interfaces mais próximas do requisitado, adicionando cores, novos posicionamentos e até efeito de interação e estes são os protótipos de alta-fidelidade (Visto no Apêndice G). Qualquer software pode ser utilizado para confecção destes modelos mais elaborados.

Os protótipos por sua vez podem ser utilizados para otimização ou descartados completamente, servindo apenas como base. Neste projeto, algumas características foram parcialmente descartadas e outras adquiridas para a melhoria da interface e estabelecimento de funcionalidades seguindo as regras de negócio. Através destas regras, com implementação completa da camada de dados e de apresentação, foi possível elaborar uma integração.

2.6. Casos e Roteiros de Testes

Os casos de testes foi uma maneira de verificar/validar as implementações e integrações das interfaces e modelos, no entanto, a partir da diagramação de classes de dados, as primeiras instâncias já foram testadas, para assim ser possível prosseguir. Este se trata do Caso 1 dividido em 5 roteiros (Visto no Apêndice E).

Estes testes iniciais do caso 1 descreve como instanciar cada classe da arquitetura do modelo de dados, antes mesmo de implementar a leitura ou escrita de arquivos. Através destas instanciações, os métodos de Registro, Exclusão e Verificação foram acionados para verificar se a herança estava adequadamente sendo aplicada.

Já no caso do Caso 2 (Apêndice H), os testes já se iniciam após a prototipação e validações de campos. Este caso contém 6 roteiros relacionados a verificação de cada janela, incluindo testes de entradas direcionadas a erros ou sucesso. Normalmente, eventos de botões são acionados, assim como menus do tipo ContextMenuStrip, também realizando formatações em expressões regulares.

Por último, o caso 3 (Apêndice I) são os testes relacionados a integração dos dados XML da camada de dados pela arquitetura Model, sendo instanciados pela camada de apresentação na arquitetura View. Além de testes de validação de campos

vazios, formatações, e armazenamento de arquivos XML, o Controller também entra em jogo, verificando as datas livres antes de aceitar de fato os empréstimos.

OBS.: Os ScreenShots se encontram na pasta “Casos de Testes”.

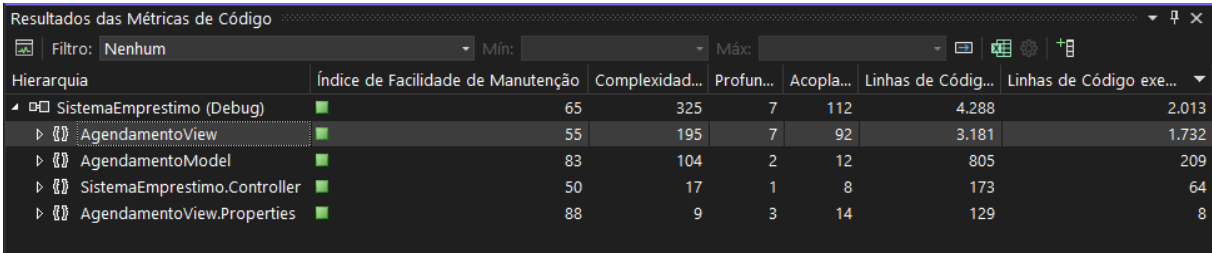
2.7. Metodologia das normas de qualidade

De acordo com as metodologias utilizadas neste software, assim como a classificação do tipo de software, o estado atual em que a empresa se encontra visualizando pela norma MPS.BR, é o “Parcialmente Gerenciado”. A empresa por ser pequena e ser brasileira, fornecendo software para uma escola brasileira, é essencial a adoção do MPS.BR, ao invés do CMMI.

Isto significa que como uma empresa que está em fase de crescimento, não é esperado que tenha um nível de qualidade tão alto ou um nível de maturidade tão elevado em pouco tempo. Portanto, o parcialmente gerenciado neste contexto é mais adequado, já que há uma mínima gerência de requisitos e de software.

Já em relação as metodologias, as melhores fortemente aplicadas para garantia da qualidade foram os roteiros de testes e testes de caixa-preta. Uma maneira mais rápida de testar, é durante o gerenciamento de requisitos, que ao mesmo tempo que um conjunto de requisitos e planejamentos são elaborados, o desenvolvimento já pode ser evidenciado, testado e comprovado. Esta foi uma metodologia de testes ágeis utilizados pela empresa.

Mesmo que há uma relação maior com o parcialmente gerenciado, o projeto pode caminhar para o nível de gerenciado definindo métricas e gerência de configuração. Segundo as métricas abaixo 54% das linhas escritas são códigos executáveis, enquanto que o nível de facilidade de manutenção está entre 62% a 65%:



Hierarquia	Índice de Facilidade de Manutenção	Complexidad...	Profun...	Acopla...	Linhas de Códig...	Linhas de Código exe...
SystemEmprestimo (Debug)	65	325	7	112	4.288	2.013
AgendamentoView	55	195	7	92	3.181	1.732
AgendamentoModel	83	104	2	12	805	209
SistemaEmprestimo.Controller	50	17	1	8	173	64
AgendamentoView.Properties	88	9	3	14	129	8

Figura 1 - Análise de código do Software

Já sobre a gerência de configurações, a maneira mais aproximada é o controle de versões do software utilizadas por um repositório privado no GitHub, que foi controlado desde o início do desenvolvimento. Portanto, uma garantia de qualidade ao menos mínimo pode ser evidenciada.

CONCLUSÃO

Este foi um trabalho realizado com muita dedicação e que provou ser uma forma gerenciada de desenvolvimento de software, aplicando conceitos da POO. Foi aprendido diversos tipos de métodos e classes para atribuir uma funcionalidade específica do software.

Foi possível observar que gerenciando melhor os prazos, investimentos e custos equivalentes, o andamento do software ocorre de maneira mais limpa e mais eficiente, pois cada tempo é verificado de sua melhor maneira.

O processo de desenvolvimento tiveram obstáculos, alguns que levaram mais do que o planejado e outros dentro do prazo estipulado, no entanto, a aplicação de planejamento de requisitos se demonstrou extremamente útil para entender exatamente o que o software deveria fazer, sem necessitar ficar diversas horas apenas pensando no que fazer.

O que foi elaborado neste trabalho é a verificação de viabilidade econômica deste projeto, desde os prazos até os investimentos. Após isto, uma realização de formulário com o cliente para entender o seu problema e a partir daí, especificar em texto diversas funções necessárias.

Não só os requisitos funcionais foram elaborados, mas também os não-funcionais, que principalmente foi identificado após todo o desenvolvimento, que na verdade foi uma comprovação de um dos requisitos. Enquanto que o requisito de acessibilidade, foi criado durante o desenvolvimento.

Todo o software funcionou da maneira que foi planejada e melhor, dentro do tempo estipulado. Foi possível aprender muito com este desenvolvimento, um contexto geral de engenharia de software.

REFERÊNCIAS

REDAÇÃO CRONAPP. **Como fazer a análise de viabilidade de projetos de aplicativos?**. Cronapp, 2019. Disponível em: <<https://blog.cronapp.io/analise-de-viabilidade-de-projetos/>> Acesso em: 15 abr. 2024.

MICROSOFT. Data Grid View Classe. Microsoft, 2024. Disponível em: <<https://learn.microsoft.com/pt-br/dotnet/api/system.windows.forms.datagridview?view=windowsdesktop-8.0/>>. Acesso em: 10 abr. 2024.

Apêndice A – Regra RN01: Professores e Equipamentos devem ser registrados

Identificador:	RN01
Nome:	Professores e equipamentos devem ser registrados
Descrição:	Cada professor entrante na escola ou cada equipamento obtido deve ser registrado para facilitar o manuseio do software, mesmo que ainda há uma possibilidade de agendar empréstimos sem estes cadastros, no entanto, não será possível filtrar empréstimos para listar sem estes cadastros.
Evento/gatilho:	Quando o usuário precisar filtrar um campo na janela de listagem ou cancelamento, um menu vai aparecer mostrando os dados registrados.
Exemplo:	O usuário salva um empréstimo com o nome “José barreiro da silva”, porém não cadastra antes este nome. O empréstimo vai acontecer, porém ele quer verificar quais foram os últimos empréstimos e decide da ENTER no campo de filtragem, O José não vai aparecer na lista. Se não houvesse esse menu, Ele poderia escrever Joao barreiro por esquecer o nome, e assim nunca encontrar a lista até que o José revele o seu nome esquecido.
Pseudocódigo:	SE todos os campos foram preenchidos ENTÃO Professor/Equipamento cadastrado com sucesso SENÃO Preencha o campo X
Documentação:	nenhuma
Regras relacionadas:	nenhuma
Responsável:	Secretário ou outro utilizador do software

Apêndice B – Regra RN02: Agendar empréstimos em datas livres

Identificador:	RN02
Nome:	Agendar empréstimos em datas livres
Descrição:	Um sistema de agendamento de empréstimos que verifica uma faixa de datas livres e cria um novo empréstimo quando as datas fornecidas estiverem entre as faixas de datas livres.
Evento/gatilho:	Quando um novo professor precisa pedir um equipamento audiovisual emprestado e ele precisa agendar em uma data, no entanto, outro professor já agendou para esta data.
Exemplo:	O professor João decide agendar um empréstimo de uma caixa de som para tocar músicas em sua sala de aula na segunda-feira às 7 hrs até as 9. A professora maria chega logo depois e decide agendar um empréstimo de 8 até as 10 hrs do mesmo dia do equipamento.
Pseudocódigo:	SE todos os campos foram preenchidos ENTÃO SE os formatos de datas estão corretos ENTÃO SE a data agendada está livre ENTÃO Agendamento de empréstimo registrado com sucesso! SENÃO A data escolhida não está disponível! SENÃO formato de data inválido SENÃO Preencha o campo X FIMSE
Documentação:	nenhuma
Regras relacionadas:	nenhuma
Responsável:	Secretário ou outro utilizador do software

Apêndice C – Regra RN03: Fechamento e Cancelamento de Empréstimos

Identificador:	RN03
Nome:	Fechamento e Cancelamento de empréstimos
Descrição:	Este sistema permite filtrar dados de professores e equipamentos cadastrados nas quais estes podem ter sido agendados seus empréstimos, então o sistema irá listar os dados de acordo com a pesquisa.
Evento/gatilho:	Quando o professor solicitar o cancelamento de um empréstimo que ele não vai necessitar mais, ou quando ele decide entregar antes da data ou até mesmo na data final agendada, o empréstimo precisa ser dado baixa no sistema. A baixa ocorre excluindo o empréstimo de um ponto e movendo para outro ponto, na lista de empréstimos baixados.
Exemplo:	No dia de segunda, O professor Matheus agenda um empréstimo de um microfone para terça-feira, porém no final da tarde ele percebe que não vai poder realizar mais o tão evento esperado, apenas na próxima semana, então ele precisa cancelar o empréstimo. Porém, a maria decide realizar o evento neste dia, logo, após cancelar o de Matheus, a maria agenda para este dia e após o evento, ela entrega o microfone, e assim o sistema realiza a baixa ou o fechamento.
Pseudocódigo:	SE não há dados listados na lista E botão pressionado ENTÃO não existe dados exibidos! SENÃO SE botão pressionado E linha não selecionada ENTÃO Nenhuma linha selecionada SENÃO O agendamento ID=X foi cancelado/fechado com sucesso! FIMSE
Documentação:	nenhuma
Regras relacionadas:	nenhuma
Responsável:	Secretários, Professores e outros funcionários.

Apêndice D – Diagrama de Classes da Arquitetura Model

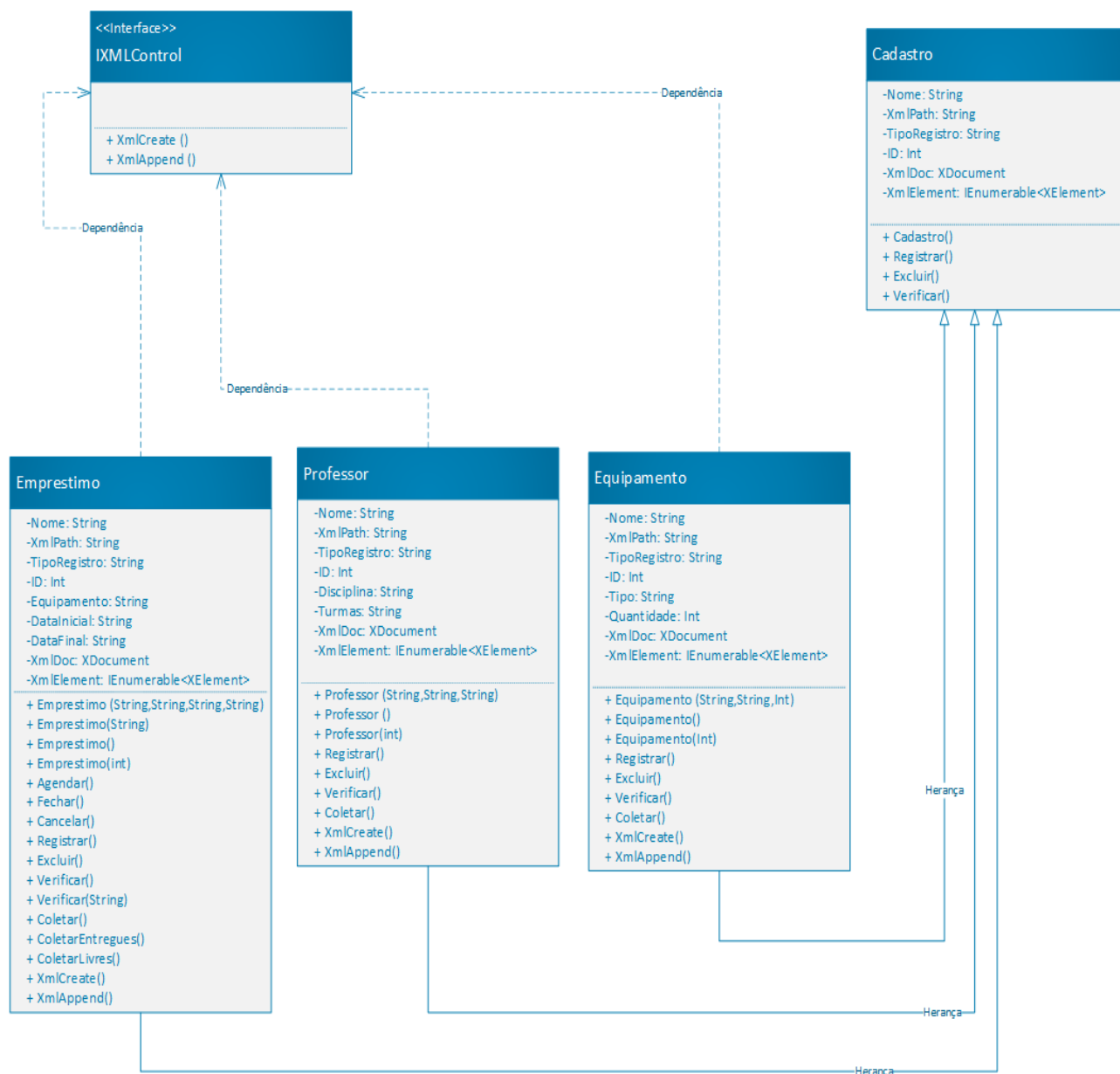


Figura 2 - Diagrama de classes da camada de dados

Apêndice E – Casos de Testes do Model

1ª Caso de Teste: Instâncias e Métodos (Model)

Roteiros

1. Instanciação da Classe "Equipamento":

1.1. Chamar método Registrar()

1.1.1. Saída esperada: Mensagem "Cadastro.Registrar()" + "Registrando Equipamento"

1.2. Chamar método Excluir()

1.2.1. Saída esperada: Mensagem "Cadastro.Excluir()" + "Excluindo Equipamento"

1.3. Chamar método Verificar()

1.3.1. Saída esperada: Mensagem "Cadastro.Verificar()" + "Verificando Equipamento"

status: OK!

2. Instanciação da Classe "Professor":

1.1. Chamar método Registrar()

1.1.1. Saída esperada: Mensagem "Cadastro.Registrar()" + "Registrando Professor"

1.2. Chamar método Excluir()

1.2.1. Saída esperada: Mensagem "Cadastro.Excluir()" + "Excluindo Professor"

1.3. Chamar método Verificar()

1.3.1. Saída esperada: Mensagem "Cadastro.Verificar()" + "Verificando Professor"

status: OK!

3. Instanciação com Passagem de Parâmetros das 2 Classes Construtoras -

Inserindo dados "DataShow" e "Wenderson", respectivamente:

3.1. Chamar método Equipamento.Registrar()

3.1.1. Saída esperada: Mensagem "Registrando DataShow no Banco de Dados"
+ "Equipamento Registrado!"

3.2. Chamar método Equipamento.Verificar()

3.2.1. Saída esperada: Mensagem "Verificando DataShow do Banco de Dados"

+ "Equipamento Verificado!"

3.3. Chamar método Equipamento.Excluir()

3.3.1. Saída esperada: Mensagem "Excluindo DataShow do Banco de Dados"

+ "Equipamento Excluído!"

3.4. Chamar método Professor.Registrar()

3.4.1. Saída esperada: Mensagem "Registrando DataShow no Banco de Dados"

+ "Equipamento Registrado!"

3.5. Chamar método Professor.Verificar()

3.5.1. Saída esperada: Mensagem "Verificando DataShow do Banco de Dados"

+ "Equipamento Verificado!"

3.6. Chamar método Professor.Excluir()

3.6.1. Saída esperada: Mensagem "Excluindo DataShow do Banco de Dados"

+ "Equipamento Excluído!"

status: OK!

4. Instanciação da Classe "Empréstimo" com Passagem de Parâmetros -

Parâmetros de Entrada = Televisor, Wenderson, 09/04/2024 18:00, 09/04/2024 21:00

= DVD, Gabriel, ..., ...

4.1. Chamar método Empréstimo.Agendar()

4.1.1. Saída esperada: Mensagem "Criando agendamento de equipamento televisor em nome de"

+ "Wenderson de 09/04/2024 18:00 até 09/04/2024 21:00"

4.2. Chamar método Empréstimo.Fechar()

4.2.1. Saída esperada: Mensagem "Dando baixa em agendamento de equipamento televisor em nome"

+ "de Wenderson de 09/04/2024 18:00 até 09/04/2024 21:00"

4.3. Chamar método Empréstimo.Cancelar()

4.3.1. Saída esperada: Mensagem "Cancelando agendamento de equipamento televisor em nome"

+ "de Wenderson de 09/04/2024 18:00 até 09/04/2024 21:00"

4.4. Chamar método Empréstimo.Agendar()

4.4.1. Saída esperada: Mensagem "Criando agendamento de equipamento DVD em nome de"

+ "Gabriel de 09/04/2024 18:00 até 09/04/2024 21:00"

4.5. Chamar método Professor.Fechar()

4.5.1. Saída esperada: Mensagem "Dando baixa em agendamento de equipamento DVD em nome"

+ "de Gabriel de 09/04/2024 18:00 até 09/04/2024 21:00"

4.6. Chamar método Professor.Cancelar()

4.6.1. Saída esperada: Mensagem "Cancelando agendamento de equipamento DVD em nome"

+ "de Gabriel de 09/04/2024 18:00 até 09/04/2024 21:00"

status: OK!

5. Instanciação da Classe "DadosDisponíveis" e Chamada de método Emprestimo.Verificar():

5.1. Chama método Emprestimo.Verificar()

5.1.1. Saída esperada: "Verificando se agendamento existe..."

+ "O agendamento mencionado existe!"

5.2. Chamar método DadosDisponiveis.ListarDiasLivres()

5.2.1. Saída Esperada: "Listando dias disponíveis"

5.3. Chamar método DadosDisponiveis.ListarEntregues()

5.3.1. Saída Esperando: "Listando equipamentos entregues"

status: OK!

Apêndice F – Protótipos de Interface de Baixa-fidelidade

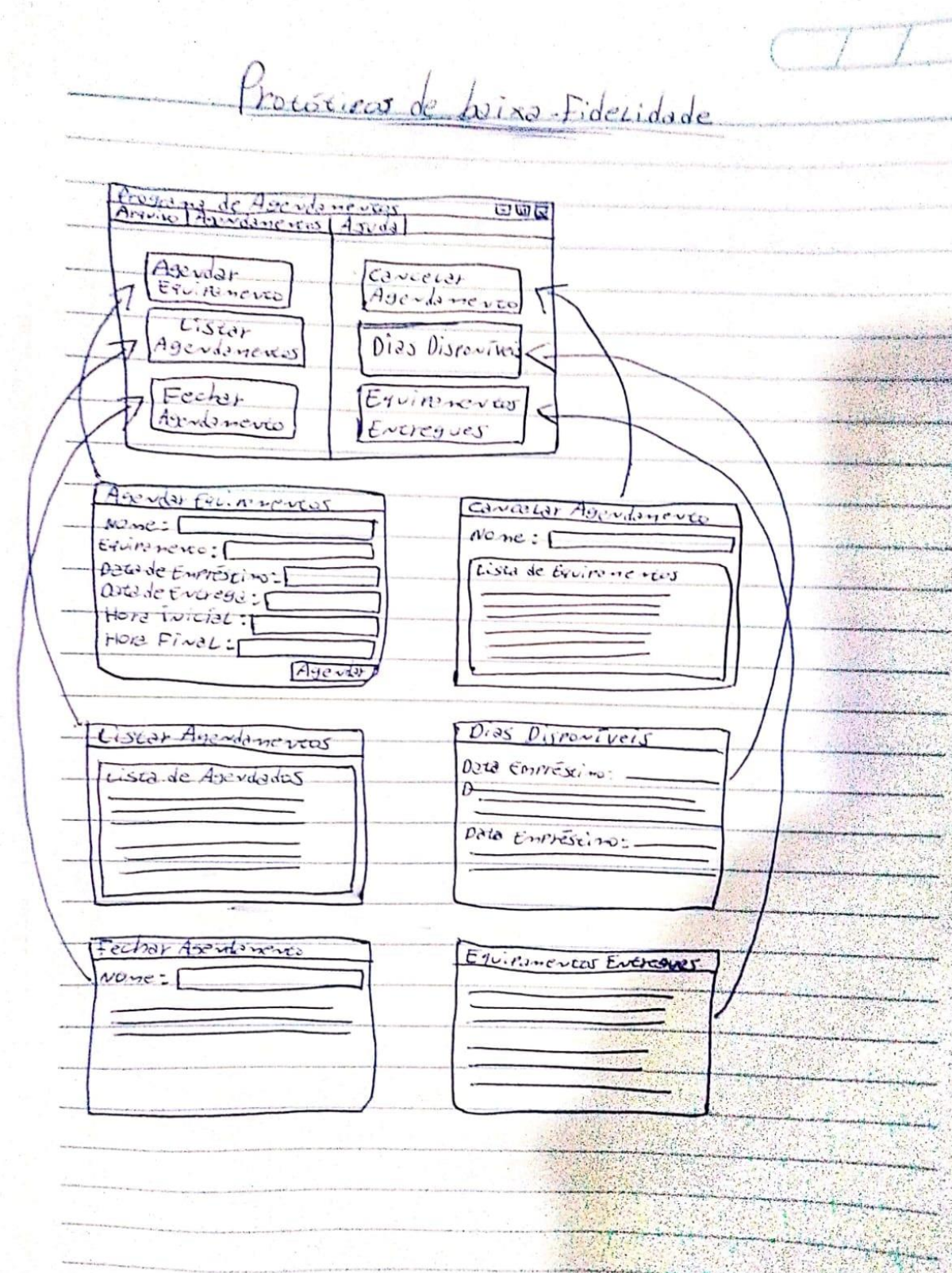


Figura 3 - Protótipos de baixa-fidelidade em papel

Apêndice G – Protótipos de Interface de Alta-fidelidade

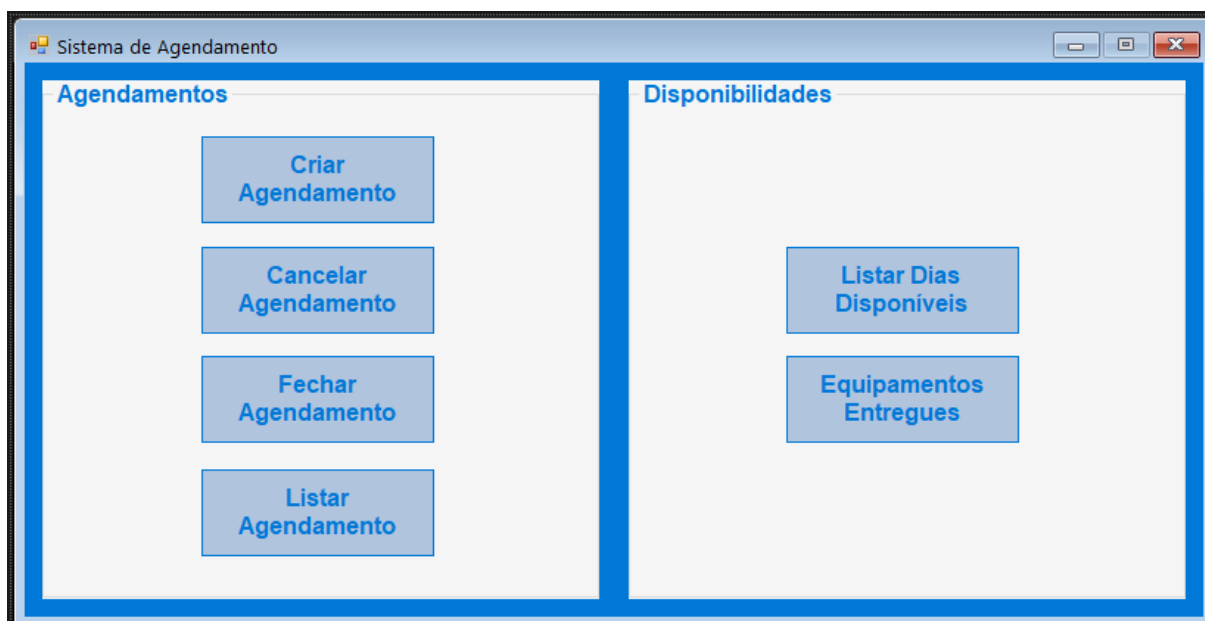


Figura 4 - Protótipo da janela principal

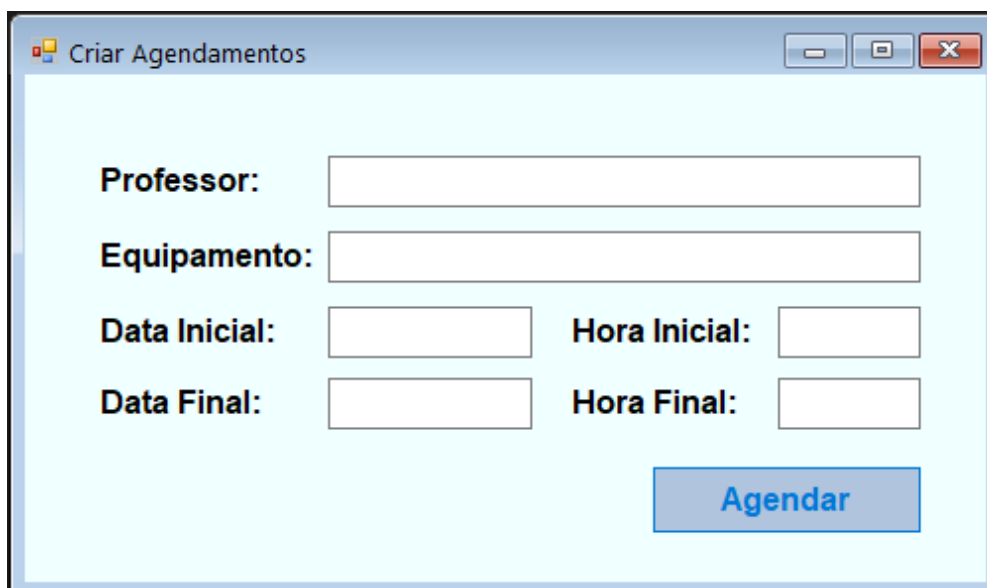
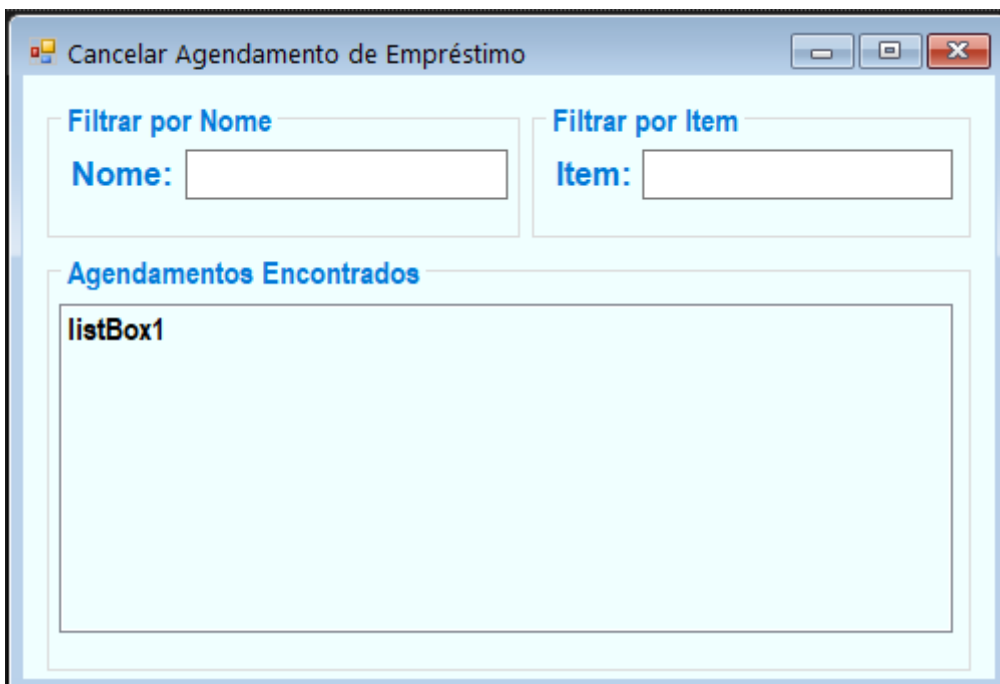


Figura 5 - Protótipo da janela de agendamentos



Cancelamento de Empréstimo

Filtrar por Nome

Nome:

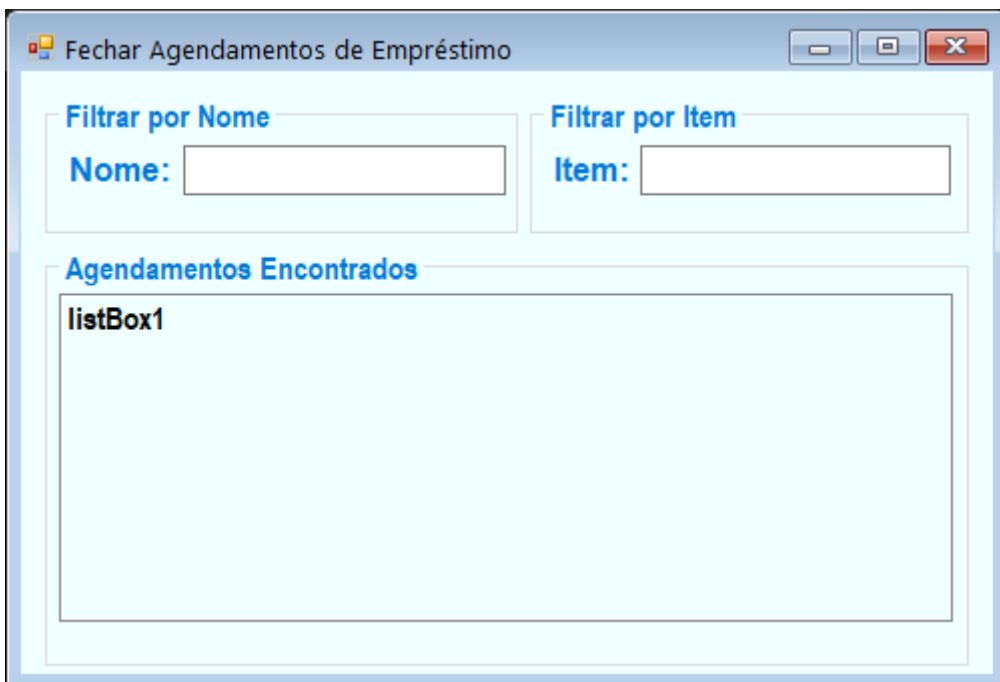
Filtrar por Item

Item:

Agendamentos Encontrados

listBox1

Figura 6 - Protótipo da Janela de cancelamento



Fechamento de Empréstimo

Filtrar por Nome

Nome:

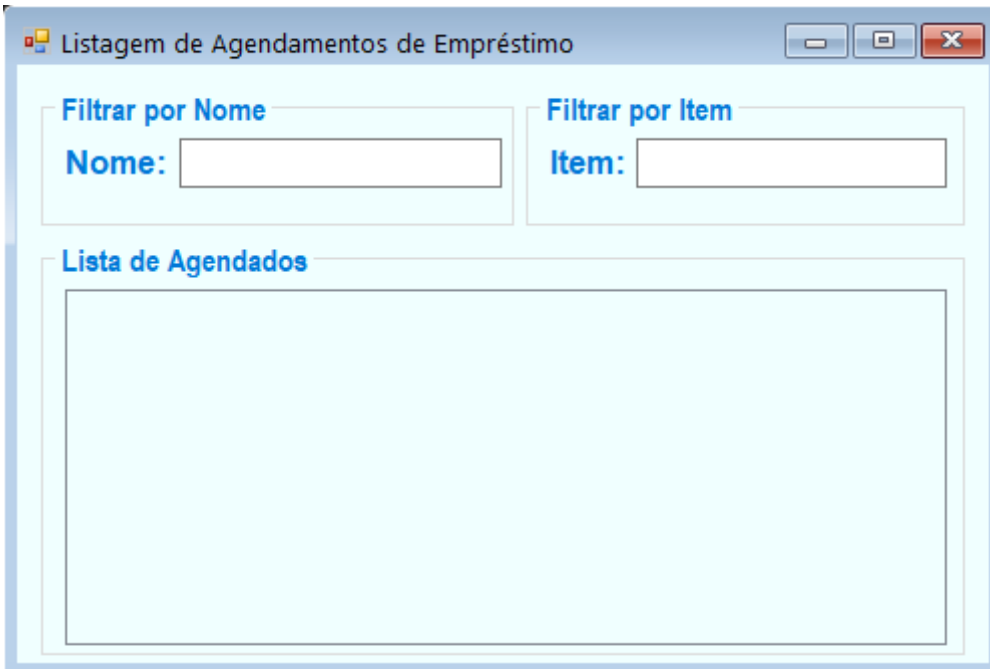
Filtrar por Item

Item:

Agendamentos Encontrados

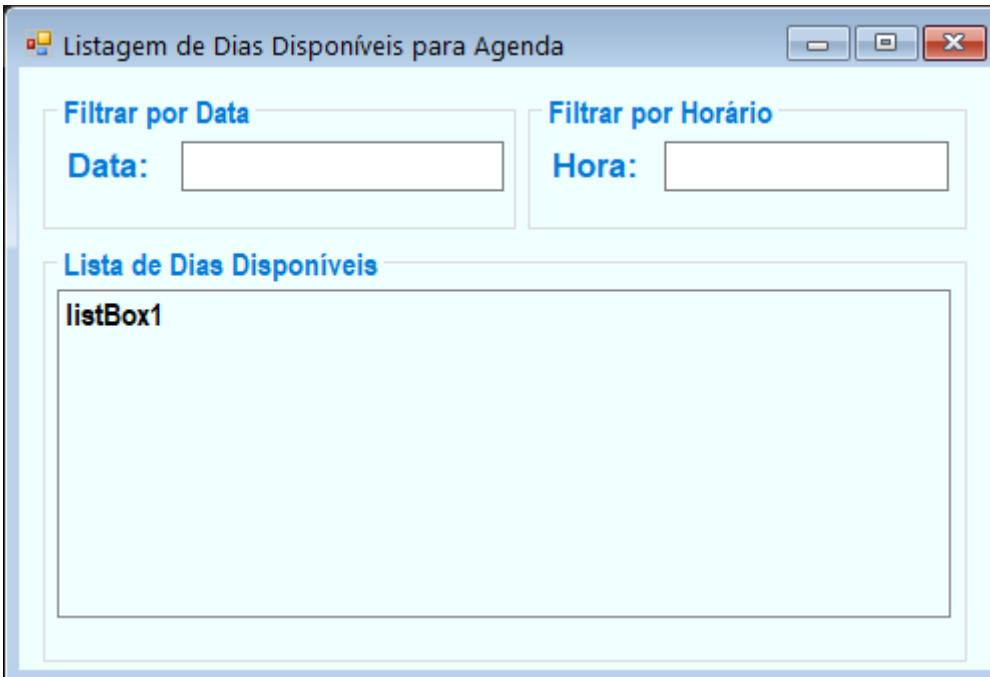
listBox1

Figura 7 - Protótipo da janela de fechamento



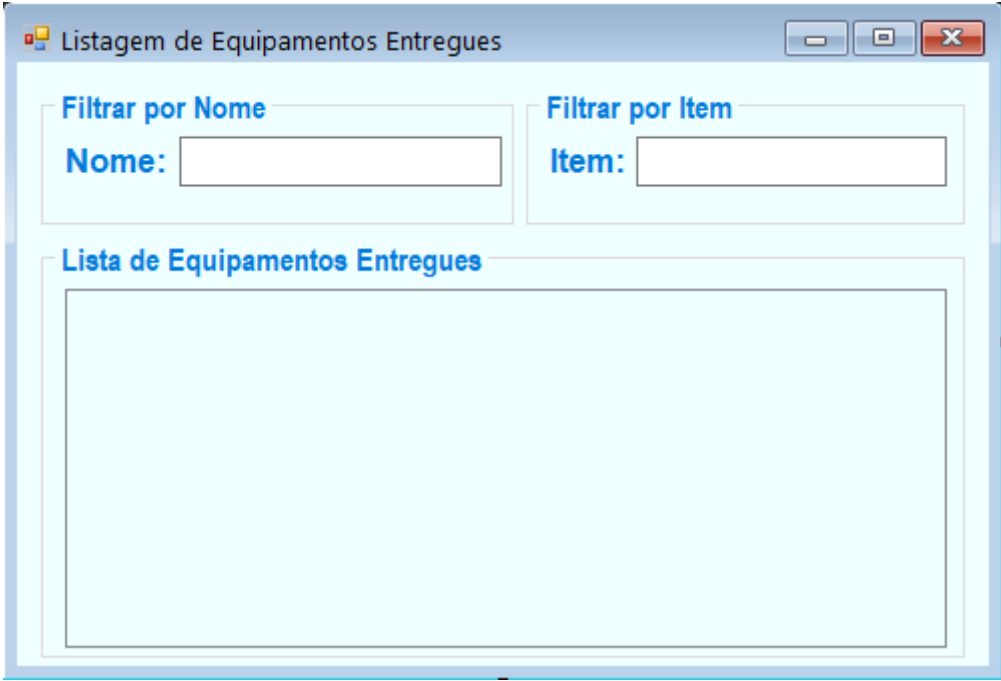
Protótipo de uma janela de software intitulada "Listagem de Agendamentos de Empréstimo". A janela possui uma barra de título com o nome e botões de minimizar, maximizar e fechar. O conteúdo é dividido em seções: "Filtrar por Nome" com um campo de texto rotulado "Nome:", "Filtrar por Item" com um campo de texto rotulado "Item:", e uma seção principal rotulada "Lista de Agendados" que contém uma grande área vazia para a exibição da lista.

Figura 8 - Protótipo da janela de listagem



Protótipo de uma janela de software intitulada "Listagem de Dias Disponíveis para Agenda". A janela possui uma barra de título com o nome e botões de minimizar, maximizar e fechar. O conteúdo é dividido em seções: "Filtrar por Data" com um campo de texto rotulado "Data:", "Filtrar por Horário" com um campo de texto rotulado "Hora:", e uma seção principal rotulada "Lista de Dias Disponíveis" que contém uma grande área vazia para a exibição da lista, com o rótulo "listBox1" no canto superior esquerdo.

Figura 9 - Protótipo da janela de datas disponíveis



Protótipo de uma janela de software intitulada "Listagem de Equipamentos Entregues". A janela possui uma barra de título com o ícone de uma pasta e o texto "Listagem de Equipamentos Entregues", além de botões de minimizar, maximizar e fechar. O conteúdo da janela é dividido em três seções principais:

- Filtrar por Nome:** Uma seção com o rótulo "Filtrar por Nome" e um campo de texto rotulado "Nome:".
- Filtrar por Item:** Uma seção com o rótulo "Filtrar por Item" e um campo de texto rotulado "Item:".
- Lista de Equipamentos Entregues:** Uma seção com o rótulo "Lista de Equipamentos Entregues" e uma grande área vazia, provavelmente destinada a exibir uma lista de itens.

Figura 10 - Protótipo da janela de equipamentos entregues

Apêndice H – Casos de Testes do View

2ª Caso de Teste: Validações de Entrada da Interface (View)

Roteiros

1. Validação de campos vazios em JanelaAgendamento:

1.1. Validar campo textProfessor:

Entrada: Nenhuma

Saída: MessageBox = "Selecione o campo professor!"

1.2. Validar campo textEquipamento:

Entrada: Apenas "Wenderson Francisco" no campo anterior

Saída: MessageBox = "Selecione o campo equipamento!"

1.3. Validar campo textDataInicial:

Entrada: Campo1 + "Projeto de dados" nos campos anteriores

Saída: MessageBox = "Preencha o campo data inicial!"

1.4. Validar campo textDataFinal:

Entrada: Campo1 + Campo2 + "10/04/2024" nos campos anteriores

Saída: MessageBox = "Preencha o campo data final!"

1.5. Validar campo textHoraInicial:

Entrada: Campo1 + Campo2 + Campo3 + "11/04/2024" nos campos anteriores

Saída: MessageBox = "Preencha o campo data inicial!"

1.6. Validar campo textHoraFinal:

Entrada: Campo1 + Campo2 + Campo3 + campo4 + "09:00:00" nos campos anteriores

Saída: MessageBox = "Preencha o campo hora final!"

1.7. Validar todos os campos (Campo1..6 - "10:00:00"):

Entrada: Todos Preenchidos

Saída: MessageBox = "Agendamento de Empréstimo registrado com sucesso!"

status: OK!

2. Validação de campos inválidos de Nome, Data e Hora em JanelaAgendamento:

2.1. Verificar tamanho de Data (Inicial/Final):

Entrada: 10/04/20

Saída: "Tamanho de data inválido!"

2.2. Verificar formato de Data com letras (Inicial/Final):

Entrada: 1o/o4/202A

Saída: "Formato de data inválido!"

2.3. Verificar tamanho de Hora (Inicial/Final):

Entrada: 12:5

Saída: "Tamanho de hora inválido!"

2.4. Verificar formato de Hora com letras (Inicial/Final):

Entrada: 12:5l:OO

Saída: "Formato de hora inválido!"

2.5. Verificar via regex o formato do nome - teste 1:

Entrada: 1 palavra

Saída: "Formato de nome inválido!"

2.6. Verificar via regex o formato do nome - teste 2:

Entrada: 1 palavra + espaço + numero

Saída: "Formato de nome inválido!"

2.7. Verificar via regex o formato do nome - teste 3:

Entrada: 2 palavras + espaço + numero

Saída: "Formato de nome inválido!"

2.8. Verificar via regex o formato do nome - teste 4:

Entrada: 2 palavras + espaço

Saída: "Agendamento de Empréstimo registrado com sucesso!"

2.9. Verificar via refex o formato das datas para dia (Inicial/Final):

Entrada: 00/04/2024

Saída: "Formato de data inválido!"

2.10. Verificar via refex o formato das datas para mês (Inicial/Final):

Entrada: 01/13/2024

Saída: "Formato de data inválido!"

2.11. Verificar via refex o formato das datas para ano (Inicial/Final):

Entrada: 01/04/3024

Saída: "Formato de data inválido!"

2.12. Verificar via refex o formato das datas substituindo barra por . (Inicial/Final):

Entrada: 01.04/2024

Saída: "Formato de data inválido!"

2.13. Verificar via refex o formato das datas (Inicial/Final):

Entrada: 01/04/2024

Saída: "Agendamento de Empréstimo registrado com sucesso!"

2.14. Verificar via refex o formato das horas para hora (Inicial/Final):

Entrada: 24:01:01

Saída: "Formato de hora inválido!"

2.15. Verificar via refex o formato das horas para minuto (Inicial/Final):

Entrada: 18:60:01

Saída: "Formato de hora inválido!"

2.16. Verificar via refex o formato das horas para segundo (Inicial/Final):

Entrada: 18:39:99

Saída: "Formato de hora inválido!"

2.17. Verificar via refex o formato das horas para segundo (Inicial/Final):

Entrada: 18:05:00

Saída: "Agendamento de Empréstimo registrado com sucesso!"

status: OK!

3. Validação de campos vazios em JanelaProfessor:

3.1. Validar campo textProfessor:

Entrada: Nenhuma

Saída: MessageBox = "Preencha o campo professor!"

3.2. Validar campo textDisciplina:

Entrada: Apenas "Wenderson Francisco" no campo anterior

Saída: MessageBox = "Preencha o campo disciplina!"

3.3. Validar campo textTurmas:

Entrada: Campo1 + "Tecnologia" nos campos anteriores

Saída: MessageBox = "Preencha o campo turmas!"

3.4. Validar todos os campos (Campo1..3 - "6ªA e 7ªB"):

Entrada: Todos Preenchidos

Saída: MessageBox = "Professor cadastrado com sucesso!"

status: OK!

4. Validação de campos vazios em JanelaEquipamento:

4.1. Validar campo textEquipamento:

Entrada: Nenhuma

Saída: MessageBox = "Preencha o campo equipamento!"

4.2. Validar campo textTipo:

Entrada: Apenas "Televisor" no campo anterior

Saída: MessageBox = "Preencha o campo tipo!"

4.3. Validar campo textQuantidade:

Entrada: Campo1 + "Visual" nos campos anteriores

Saída: MessageBox = "Preencha o campo quantidade!"

4.4. Validar todos os campos (Campo1..3 - "3"):

Entrada: Todos Preenchidos

Saída: MessageBox = "Equipamento cadastrado com sucesso!"

status: OK!

5. Validação de formato do campo Professor em JanelaProfessor:

5.1. Verificar campo com 1 palavra:

Entrada: "Wenderson"

Saída: MessageBox = "Formato de nome inválido! Não é permitido números ou nomes incompletos"

5.2. Verificar campo com 1 palavra + numero:

Entrada: "Wenderson27"

Saída: MessageBox = "Formato de nome inválido! Não é permitido números ou nomes incompletos"

5.3. Verificar campo com 2 palavras + espaço + numero:

Entrada: "Wenderson Francisco27"

Saída: MessageBox = "Formato de nome inválido! Não é permitido números ou nomes incompletos"

5.4. Verificar campo com 2 palavras + espaço:

Entrada: "Wenderson Francisco"

Saída: MessageBox = "Professor cadastrado com sucesso!"

status: OK!

6. Seleção de Itens em JanelaCancelamento, Fechamento, Listagem, Entregues e Disponibilidade:

6.1. Selecionar Item 1 em textProfessor de JanelaCancelamento:

Entrada: "Wenderson"

Saída: ListBox = "Wenderson : 0 ... Wenderson : 9"

6.2. Selecionar Item 1 em textItem de JanelaCancelamento:

Entrada: "Projeto de Dados"

Saída: ListBox = "Projeto de Dados : 0 ... Projeto de Dados : 9"

6.3. Selecionar Item 2 em textProfessor de JanelaFechamento:

Entrada: "Gabriel"

Saída: ListBox = "Gabriel : 0 ... Gabriel : 9"

6.4. Selecionar Item 2 em textItem de JanelaFechamento:

Entrada: "Televisor 4K"

Saída: ListBox = "Televisor 4K : 0 ... Televisor 4K : 9"

6.5. Selecionar Item 3 em textProfessor de JanelaListagem:

Entrada: "Joana"

Saída: ListBox = "Joana : 0 ... Joana : 9"

6.6. Selecionar Item 3 em textItem de JanelaListagem:

Entrada: "Caixa de Som"

Saída: ListBox = "Caixa de Som : 0 ... Caixa de Som : 9"

6.7. Selecionar Item 1 em textProfessor de JanelaEntregues:

Entrada: "Gabriel"

Saída: ListBox = "Gabriel : 0 ... Gabriel : 9"

6.8. Selecionar Item 1 em textItem de JanelaEntregues:

Entrada: "Televisor 4K"

Saída: ListBox = "Televisor 4K : 0 ... Televisor 4K : 9"

6.9. Selecionar Item 2 em textProfessor de JanelaDisponibilidade:

Entrada: "Dia 3"

Saída: ListBox = "Dia 3 : 0 ... Dia 3 : 9"

6.10. Selecionar Item 2 em textItem de JanelaDisponibilidade:

Entrada: "Hora 2"

Saída: ListBox = "Hora 2 : 0 ... Hora 2 : 9"

status: OK!

Apêndice I – Casos de Testes do View-Model (Integração) + Negócios

3ª Caso de Teste: Armazenamento e Controle de Dados XML (MVC)

Roteiros

1. Cadastrar novo professor

1.1. Inserir 1º professor nos campos - Professor, Disciplina e Turmas:

Entrada: Wenderson Francisco, Tecnologia, 6ª e 7ª

Saída: XML = Dados relativos ao cadastrado em Professor.xml com ID=1

1.2. Inserir 2º professor nos campos - Professor, Disciplina e Turmas:

Entrada: Matheus Henrique, Portugues, 5ª e 8ª

Saída: XML = Dados relativos ao cadastrado em Professor.xml com ID=2

status: OK!

Roteiros

2. Cadastrar novo equipamento

2.1. Inserir 1º equipamento nos campos - Equipamento, Tipo e Quantidade:

Entrada: Caixa de Som, Auditivo, 5

Saída: XML = Dados relativos ao cadastrado em Equipamento.xml com ID=1

2.2. Inserir 2º equipamento nos campos - Equipamento, Tipo e Quantidade:

Entrada: Projetor de Dados, Visual, 3

Saída: XML = Dados relativos ao cadastrado em Equipamento.xml com ID=2

status: OK!

Roteiros

3. Agendar novo empréstimo

3.1. Criar 1ª agenda nos campos - Professor, Equipamento, DataInicial, Hora Inicial, Data Final e Hora Final:

Entrada: Wenderson Francisco, Projetor de Dados, 14/04/2024, 22:00:00, 14/04/2024, 23:00:00

Saída: XML = Dados relativos ao cadastrado em Emprestimo.xml com ID=1

3.2. Criar 2ª agenda nos campos - Professor, Equipamento, DataInicial, Hora Inicial, Data Final e Hora Final:

Entrada: Matheus Henrique, Caixa de Som, 15/04/2024, 07:00:00, 15/04/2024, 09:00:00

Saída: XML = Dados relativos ao cadastrado em Emprestimo.xml com ID=2

3.3. Criar 3ª agenda nos campos - Professor, Equipamento, DataInicial, Hora Inicial, Data Final e Hora Final de Matheus com datas dentro da faixa que a 1ª agenda (Wenderson):

Entrada: Matheus Henrique, Caixa de Som, 14/04/2024, 22:30:00, 14/04/2024, 22:50:00

Saída: Tela de Erro = A data escolhida não está disponível! Por favor, verificar na lista de dias disponíveis!

3.4. Saídas de dias livres diferentes dos agendamentos:

Saída: XML = Intervalos de datas/horários livres

status: OK!

Roteiros

4. Listar dias disponíveis para empréstimo

4.1. Abrir janela de listagem de dias disponíveis:

Saída: Datas diferentes entre as faixas daquelas que foram agendadas

4.2. Entrar com um número do dia no campo Data para Filtragem:

Entrada: 14

Saída: 14/04/2024 21:56:12, 14/04/2024 22:00:00

14/04/2024 23:00:00, 15/04/2024 07:00:00

Entrada: 13

Saída: Nenhuma

Entrada: Data

Saída: 15/04/2024 09:00:00, Data Indefinida

4.3. Entrar com um número de hora no campo Data para Filtragem:

Entrada: 22

Saída: 14/04/2024 21:56:12, 14/04/2024 22:00:00

Entrada: 7

Saída: 14/04/2024 23:00:00, 15/04/2024 07:00:00

Entrada: 9

Saída: 15/04/2024 09:00:00, Data Indefinida

status: OK!

Roteiros

5. Listar empréstimos atuais

5.1. Abrir janelas de empréstimos:

Saída: Datas iguais daquelas que foram agendadas

5.2. Filtrar por nome os dados agendados:

Entrada: Wenderson Francisco

Saída: Wenderson Francisco, Projetor de Dados, ...

5.3. Filtrar por item os dados agendados:

Entrada: Caixa de Som

Saída: Matheus Henrique, Caixa de Som, ...

status: OK!

Roteiros

6. Cancelar empréstimos

6.1. Inserir entrada no campo Nome para a filtragem do emprestimo:

Entrada: Wenderson Francisco

Saída: Wenderson Francisco, Projetor de Dados

6.2. Clicar no botão para cancelar o emprestimo com ID=1 + seleção de linha:

Entrada: Click em "Cancelar"

Saída: XML = Wenderson Francisco Excluído de Emprestito.xml

6.3. Abrir janela pra listar agendamentos:

Saída: Matheus Henrique, Caixa de Som

status: OK!

Roteiros

7. Fechar empréstimos

7.1. Inserir entrada no campo Item para a filtragem do emprestimo:

Entrada: Caixa de Som

Saída: Matheus Henrique, Caixa de Som

7.2. Clicar no botão para fechar o empréstimo com ID=2 + seleção de linha:

Entrada: Click em "Fechar"

Saída: XML = Matheus Henrique Excluído de Empréstimo.xml

7.3. Abrir janela pra listar agendamentos:

Saída: Nenhuma

7.4. Abrir janela para listar entregues:

Saída: Matheus Henrique, Caixa de Som, 15/04/2024 07:00:00, 15/04/2024 09:00:00

7.5. Abrir janela para listar dias disponíveis:

Saída: Novo dia disponível = 15/04/2024 07:00:00, 15/04/2024 09:00:00

status: OK!

Roteiros

8. Listar equipamentos entregues

8.1. Criar +2 agendamentos e fechá-los imediatamente em JanelaAgendamento:

Entradas: Wenderson Francisco, Caixa de Som

Matheus Henrique, Projetor de Dados

Saída: Em JanelaEntregues = Os dois dados fechados + 1 dado anterior (Matheus Henrique)

Em Entregues.xml = Os 3 dados cujos IDs são: 1, 2 e 3

Em Dias Disponíveis = As novas 2 datas

status: OK!
