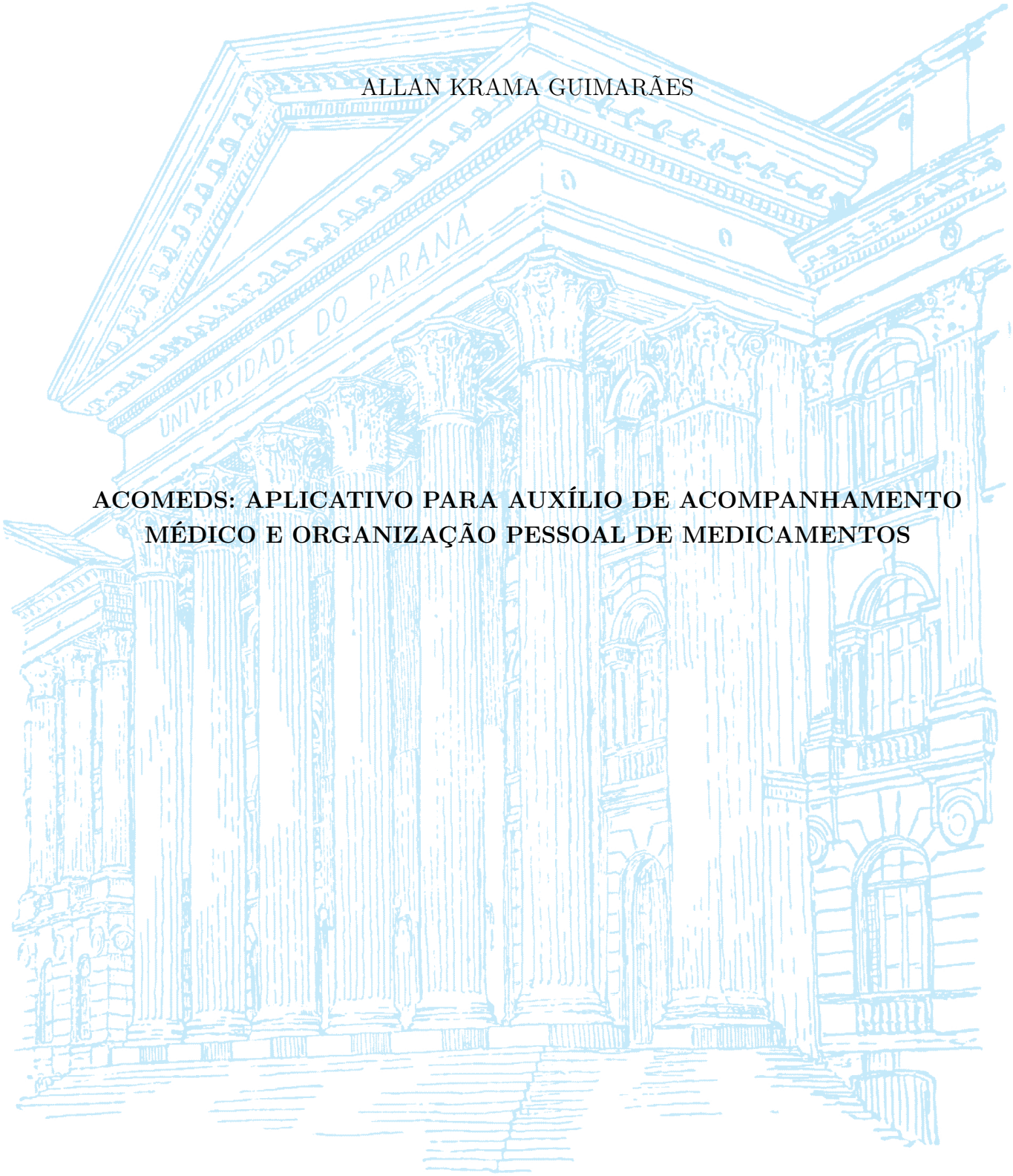
UNIVERSIDADE FEDERAL DO PARANÁ

ALLAN KRAMA GUIMARÃES



ACOMEDS: APLICATIVO PARA AUXÍLIO DE ACOMPANHAMENTO MÉDICO E ORGANIZAÇÃO PESSOAL DE MEDICAMENTOS

CURITIBA

2017

ALLAN KRAMA GUIMARÃES

ACOMEDS: APLICATIVO PARA AUXÍLIO DE ACOMPANHAMENTO MÉDICO E ORGANIZAÇÃO PESSOAL DE MEDICAMENTOS

Trabalho de Conclusão de Curso apresentado ao Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas da Universidade Federal do Paraná como requisito à obtenção do título de grau de Graduado em Análise e Desenvolvimento de Sistemas
Orientador: Prof. Dr. Mario de Paula Soares Filho

CURITIBA

2017

TERMO DE APROVAÇÃO

ALLAN KRAMA GUIMARÃES

ACOMEDS: APLICATIVO PARA AUXÍLIO DE ACOMPANHAMENTO MÉDICO E ORGANIZAÇÃO PESSOAL DE MEDICAMENTOS

Trabalho de Conclusão de Curso apresentado ao Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas da Universidade Federal do Paraná como requisito à obtenção do título de grau de Graduado em Análise e Desenvolvimento de Sistemas, pela seguinte banca examinadora:

Prof. Dr. Mario de Paula Soares Filho
Orientador - Setor de Educação Profissional e Tecnológica da Universidade Federal, UFPR.

Professor Dr. Dieval Guizelini
Setor de Educação Profissional e Tecnológica da Universidade Federal, UFPR.

Professor Dr. Luiz Antônio Pereira Neves
Setor de Educação Profissional e Tecnológica da Universidade Federal, UFPR.

Curitiba, 07 de dezembro de 2017

RESUMO

O presente trabalho refere-se à um aplicativo que tem como proposta ajudar pessoas a controlarem o uso de seus medicamentos e a acompanharem indicadores de saúde. A metodologia de desenvolvimento de software caracteriza-se predominantemente pelos métodos ágil e de programação orientada a objetos. Para que fosse possível abranger o maior número possível de usuários em potencial, o aplicativo foi construído visando a disponibilização multiplataforma com uma interface de usuário amigável e fluída. Para que tal implementação fosse realizada, foi utilizada a ferramenta Qt Quick. O resultado do desenvolvimento gerou um aplicativo subdividido em dois módulos : um de acompanhamento de saúde e outro de medicamentos.

Palavras-chave: Acompanhamento de saúde . Medicação . Remédio . Agendamento . Saúde . Multiplataforma.

ABSTRACT

The present work concerns about an app that aims to help people to manage medications and to follow up healthmarkers. The software development methodology characterizes itself predominantly by the object oriented and agile methods. In order to reach the biggest amount of potential users, the app was built aiming the cross-platform availability with a friendly and smooth user interface. To realize such implementations, the framework Qt Quick was used. The result of the development was consisted in an app subdivided in two modules : one for health follow up and another for medications.

Keywords: Health follow up . Medication . Drug . Scheduling . Health . Crossplatform.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

APK – *Android Package Kit*
CPU – *Central Processing Unit*
CRUD – *Create, Read, Update and Delete*
DER – Diagrama Entidade Relacionamento
DP – *Double Precision*
FLOPS – *Floating-point Operations Per Second*
GCC – *GNU Compiler Collection*
GPL – *GNU General Public License*
GPU – *Graphics Processing Unit*
IDE – *Integrated Development Environment*
IPC – *Instructions Per Cycle*
ISO – *International Organization for Standardization*
JSON – *JavaScript Object Notation*
LGPLv3 – *GNU Lesser General Public License v3.0*
MER – Modelo Entidade Relacionamento
MinGW – *Minimalist GNU for Windows*
MVC – *Model, View and Controller*
POC – *Proof of Concept*
ROP – *Raster Operation Pipeline*
SCM – *Source Code Management*
SI – *Système international (d'unités)*
SMT – *Simultaneous multithreading*
SP – *Single Precision*
STL – *Standard Template Library*
TMU – *Texture Mapping Unit*
UML – *Unified Modeling Language*
XML – *eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

O avanço da medicina e tecnologia têm proporcionado melhores condições de saúde e qualidade de vida às pessoas.

> *Nos últimos 100 anos houve um aumento da expectativa de vida em mais de 30 anos. Rehens (apud GLOBO, 2011, § 2º)*

. Em paralelo, o número de usuários de dispositivos móveis no Brasil aumenta a cada ano (EXAME, 2016). Unindo estes fatos a proposta deste trabalho é a criação de um aplicativo multiplataforma que sirva de ferramenta para pacientes que fazem uso medicamentoso.

O aplicativo Acomeds foi pensado e desenvolvido baseado na necessidade das pessoas que fazem uso medicamentoso, pois há vários casos de pacientes que fazem usos de múltiplos fármacos, cuja dependência faz parte de suas rotinas. O Acomeds tem a proposta de facilitar, organizar e controlar a rotina medicamentosa de cada paciente, possibilitando o acompanhamento pelo médico.

Considerando a crescente e contínua acessibilidade aos recursos tecnológicos, ao aumento de usuários de dispositivos móveis e a característica intuitiva do projeto, consideramos que as novas gerações não teriam dificuldades em fazer uso deste aplicativos.

O aplicativo baseia-se no cadastro de um medicamento, utilizando as variáveis de periodicidade, quantidade, duração, com a possibilidade de alerta e foto da prescrição médica. Através dos dados informados, o aplicativo calcula os horários em que o remédio precisa ser tomado e dispara uma notificação afim de avisar o usuário no dispositivo sendo utilizado. Essa função também pode ser desligada.

Considerando que cada vez mais os dispositivos móveis terão um papel importante para o acompanhamento de saúde, como já tem sido evidenciado (ESTADãO, 2013), foi implementado um módulo de indicadores de saúde. Para que os usuários possam listar e acompanhar dados de sua saúde como peso, colesterol, glicose e dados pertinentes. a partir dos resultados o aplicativo pode gerar um gráfico para facilitar a visibilidade e entendimento ao usuário.

1.1 OBJETIVO GERAL

Desenvolver um aplicativo voltado ao controle de tratamentos e terapias medicamentosas, capaz de armazenar receituários e exames clínicos gerando histórico confiável para pacientes e profissionais da área da saúde.

1.2 OBJETIVOS ESPECÍFICOS

A) Possibilitar o registro de medicamentos, com suas respectivas posologias, e notificar o usuário quando o mesmo optar pela notificação de acordo com a posologia.

B) Possibilitar o acompanhamento de índicadores de saúde, mostrando um gráfico para análise facilitada dos resultados.

C) Disponibilizar o aplicativo para as plataformas *desktop* e *mobile*.

D) Apresentar interface de usuário baseada nos padrões de *design* Material e Universal.

1.3 JUSTIFICATIVA

Cuidar da saúde e ingerir corretamente os medicamentos prescritos são duas necessidades que ocorrerão durante toda a vida de qualquer pessoa. Elaborou-se então o aplicativo Acomeds, visando possibilitar uma pessoa tanto a monitorar seus índices de saúde e acompanhar sua progressão, quanto a monitorar o uso de medicamentos para que sejam ingeridos no tempo certo e da maneira adequada.

O Acomeds tem o intuito de ser um aplicativo capaz de centralizar todas estas informações em um único lugar, permitir o compartilhamento destas somente com quem o usuário desejar, garantindo o total sigilo das informações.

1.3.1 ANÁLISE DO PÚBLICO ALVO

Doenças ocorrem frequentemente no cotidiano das pessoas. Isso faz com que elas procurem médicos para se consultarem procurando ficarem saudáveis novamente. Entretanto, nem todos os casos de adoecimentos permitem que haja melhora imediata e requerem - assim como na maioria dos tratamentos - prescrição de fármacos com posologia determinada, alguns ainda requerem realização de exercícios físicos e monitoria de indicadores de saúde para que o paciente possa auxiliar o médico na análise de resultados.

Esse controle - de posologia de remédios, exercícios físicos e indicadores de saúde - é realizado pelo usuário que muitas vezes pode acabar se enganando ou se esquecendo de alguns detalhes. Há ainda doenças que afetam a memória (como Alzheimer por exemplo) que podem agravar ainda mais a condição do paciente de monitorar seus horários.

> A idade é o principal fator de risco para as doenças predominantes em países desenvolvidos : câncer, doenças cardiovasculares e neurodegeneração. (NICCOLI; PARTRIDGE, 2012, § 1º, tradução nossa).

Envelhecer é um fator de risco para a saúde e com isso a frequência de problemas relacionados a enganos referentes ao uso de fármacos é potencializado uma vez que a dosagem e duração dos mesmos pode variar.

Haja vista que alguns tratamentos requerem medicamentos em que o uso deve ser constante, que algumas doenças que requerem estes tratamentos podem afetar tanto pessoas de idade avançada quanto pessoas jovens, e que até exercícios físicos também carecem de acompanhamento, se torna extremamente útil e eficiente, para um número significativamente grande de pessoas, um aplicativo que auxilia na organização dos medicamentos e na monitoria de saúde.

É interessante também notar que a população - em sua maioria - tem acesso a tecnologias como smartphones, computadores, tablets e notebooks e o alcance deste

aplicativo abrangeria tanto faixas etárias diversas, como estas várias plataformas e seus sistemas operacionais.

Tornar o aplicativo operável sem conexão com a internet também ajudaria a alcançar mais usuários. Porém, ele não precisaria ser inteiramente desconectado.

2 MATERIAIS E MÉTODOS

A proposta deste capítulo é descrever como o aplicativo foi desenvolvido, apresentando : a maneira como o projeto foi gerenciado, os paradigmas empregados no desenvolvimento do código, a maneira como a documentação foi realizada e as ferramentas utilizadas para o desenvolvimento.

2.1 GERENCIAMENTO DO PROJETO

No gerenciamento de projeto de software, é empregado a metodologia ágil, mais especificamente seguindo o processo de controle e gerenciamento Scrum.

2.1.1 MÉTODO ÁGIL

O manifesto ágil possui 12 princípios básicos e resumidamente prioriza as pessoas envolvidas, o relacionamento com o cliente, o desenvolvimento da aplicação e a adaptação às mudanças.

Os doze princípios podem ser resumidos em :

> *Princípios por trás do Manifesto Ágil*
>
> *Nós seguimos estes princípios:*
>
> *Nossa maior prioridade é satisfazer o cliente através da entrega contínua e adiantada de software com valor agregado.*
>
> *Mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento. Processos ágeis tiram vantagem das mudanças visando vantagem competitiva para o cliente.*
>
> *Entregar frequentemente software funcionando, de poucas semanas a poucos meses, com preferência à menor escala de tempo.*
>
> *Pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto.*
>
> *Construa projetos em torno de indivíduos motivados. Dê a eles o ambiente e o suporte necessário e confie neles para fazer o trabalho.*
>
> *O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é através de conversa face a face.*
>
> *Software funcionando é a medida primária de progresso.*
>
> *Os processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente.*
>
> *Contínua atenção à excelência técnica e bom design aumenta a agilidade.*
>
> *Simplicidade–a arte de maximizar a quantidade de trabalho não realizado–é essencial.*

> *As melhores arquiteturas, requisitos e designs emergem de equipes auto-organizáveis.*
>
> *Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento de acordo. (AGILEMANIFESTO, 2001, tradução nossa)*

A metodologia ágil preza mais pela comunicação do cliente e entrega do produto, sendo esse último normalmente subdividido em subprodutos menores e funcionais visando atingir cada objetivo do produto inicial.

Essa forma de divisão do produto visa minimizar o risco atrelado à entrega do produto final de duas maneiras principais :

1. O cliente é capaz de sinalizar alterações no projeto de acordo com as entregas de subprodutos e sua avaliação sobre as mesmas.

2. Caso ocorra algum imprevisto ou mudança durante o percurso de desenvolvimento do projeto, o que já foi entregue já encontra-se funcional para o cliente e as alterações a partir daquele ponto exercem um impacto menor para o cliente comparado a um desenvolvimento tradicional seguindo o modelo cascata.

Após identificadas as funcionalidades de cada subproduto, o desenvolvimento da aplicação passa a ocorrer priorizando a comunicação com o cliente final sobre a especificação documental. Dessa maneira, caso haja algo que não está conforme o cliente gostaria ou que o cliente tenha notado que necessita ser alterado, a mudança ocorre causando impactos muito pequenos no processod e desenvolvimento.

Cada finalização de um subproduto ou funcionalidade que é comunicado e disponibilizado para o cliente caracteriza uma entrega.

O desenvolvimento de cada parte de uma entrega ocorre de forma iterativa, dividindo cada entrega em partes menores como objetivo de desenvolvimento (a divisão do produto em entregas de produtos menores pode ser considerado a primeira iteração). O modo de funcionamento desse processo iterativo irá depender da metodologia sendo utilizada pela equipe no desenvolvimento.

Uma metodologia que segue os princípios do método ágil age como um guia para a equipe de desenvolvimento, indicando como a equipe deve atuar para que o resultado esperado seja atingido, minimizando os riscos inerentes ao desenvolvimento.

2.1.2 METOGOLOGIA SCRUM

O Scrum é uma metodologia ágil que divide o projeto de desenvolvimento de software em *Releases* que são subdivididas em *Sprints*.

Cada *Release* representa uma parte, versão ou funcionalidade entregue ao cliente para utilização e portanto, em funcionamento, e deste modo, a entrega de uma *Release* envolve a apresentação do que foi desenvolvido em forma de uma demonstração para o cliente.

Cada *Sprint* representa um curto espaço de tempo medido em poucas semanas, normalmente de 2 a 3, que visa completar o desenvolvimento de alguma funcionalidade interna necessária para a *Release*, sendo assim, no início de uma *Sprint* os desenvolvedores devem fazer um planejamento preferencialmente comunicando o cliente, e a cada dia deve-se realizar uma reunião rápida entre os desenvolvedores - chamada de Daily *Meeting* - para atualizar o estado de suas tarefas e discutir possíveis soluções em caso de problemas.

A maneira de divisão de tarefas no Scrum é realizada com as informações coletadas como objetivos do sistema, o *Product Backlog*, essas informações no formato de requisitos devem estar listadas com todas as sub-tarefas necessárias para completá-la. A partir dessa lista de objetivos do sistema, é escolhido estrategicamente um objetivo para definir a entrega de uma *Release* e a partir das sub-tarefas são formados os *Sprint-backlogs* que são basicamente a lista de tarefas a serem realizadas dentro de determinada *Sprint*.

2.1.2.1 ADAPTAÇÃO DA METODOLOGIA ÀS NECESSIDADES DA EQUIPE

No desenvolvimento do aplicativo Acomeds, inicialmente foram realizadas *Sprints* semanais porém logo foi percebido que não estava sendo eficiente devido o curto espaço de tempo principalmente para funcionalidades que envolviam uma quantidade maior de horas dedicadas às tarefas, passou-se então a serem realizadas *Sprints* com duração de 2 a 4 semanas, dependendo da complexidade da funcionalidade sendo desenvolvida.

As *Daily Meetings* foram realizadas a cada período de até 10 horas de desenvolvimento, devido ao tempo disponível para desenvolvimento do projeto, e portanto não foram necessariamente realizadas diariamente mas sim a cada 2 ou 3 dias. Dessa maneira tornou-se mais eficaz informar as atualizações e mitigar problemas no desenvolvimento.

2.1.3 TRELLO - FERRAMENTA

A ferramenta online Trello(TRELLO, 2017) é utilizada como forma de registro das tarefas realizadas em cada etapa de desenvolvimento de funcionalidades do sistema. Essa ferramenta permite o gerenciamento de atividades usando *Boards* (quadros) que representam as *Sprints* e listas dentro desses quadros que possibilitam o registro de atividades que seriam desenvolvidas dentro de uma *Sprint*.

Cada tarefa dentro de um quadro no Trello poderia ser atrelada a uma ou mais pessoas, era possível adicionar a descrição da tarefa, adicionar observações e etiquetas, também era possível mover uma tarefa de uma lista para outra, possibilitando melhor organização. Ainda era possível utilizar outras funcionalidades na ferramenta, porém, não foram utilizadas no gerenciamento das funcionalidades pela equipe.

Para o desenvolvimento deste projeto, cada quadro no Trello representa uma *Sprint*. Cada quadro era subdividido em tarefas.

Cada tarefa possuia um nome, uma prioridade e uma etiqueta após ser iniciada. A prioridade de uma tarefa valia-se pela ordem em que aparecia na lista : quanto mais ao topo, maior a prioridade.

Se uma tarefa estiver com uma etiqueta significa que sua implementaçãao havia sido iniciada, as cores das etiquetas representam o estado das tarefas :

1. Etiqueta verde : a tarefa foi iniciada e finalizada dentro do prazo estipulado para a *Sprint*.

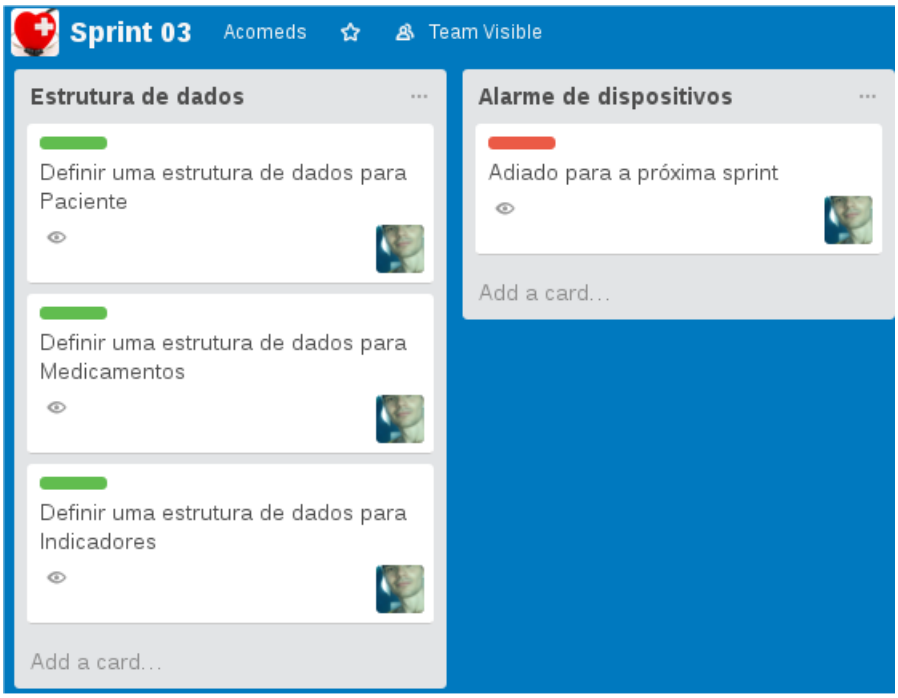
2. Etiqueta amarela : a tarefa foi iniciada mas não houve tempo suficiente para finalizá-la dentro da *Sprint* e então ela foi movida ou adaptada para a próxima *Sprint* de acordo com a demonstração final das funcionalidades, em que era possível identificar problemas a serem solucionados e modificações a serem realizadas.

3. Etiqueta vermelha : a tarefa foi iniciada e posteriormente cancelada devido a algum problema que impossibilitaria sua implementação ou por ter-se observado a não necessidade ou não coerência da realização dessa sub-funcionalidade. Foram adicionados comentários justificando o cancelamento da tarefa.

4. Várias etiquetas : tarefas que não foram bem divididas e deveriam ser melhor divididas para a próxima *Sprint* se necessário continuá-las. Portanto, como exemplo, se uma tarefa tivesse as etiquetas verde e vermelho significaria que parte dela foi concluída e outra parte teria de ser descontinuada por motivos descritos nos comentários.

A figura 1 demonstra um exemplo de como ocorreram as organizações de uma *Sprint* no Trello.

FIGURA 1. EXEMPLO DA FERRAMENTA TRELLO.



Fonte. O AUTOR (2017).

2.1.3.1 DIVISÃO DE TAREFAS

As tarefas foram divididas em 5 partes :

1. C++ para manipulação de objetos e *JavaScript Object Notation* (JSON)

2. QML para construção de telas

3. QML para adequação da prototipação do aplicativo e documentação textual

4. Gerência do projeto
5. Documentação do projeto

O foco inicial foi o aprendizado da linguagem QML e posteriormente as tarefas foram revezadas.

A cada final de *Sprint* discutia-se o andamento e as possíveis melhorias na execução de tarefas juntamente com o tutor da equipe, o acadêmico Mário de Paula Soares Filho, que realizava um *feedback* com as observações e recomendações pertinentes.

2.2 PARADIGMAS DE DESENVOLVIMENTO

É apresentado a maneira que o aplicativo foi desenvolvido, bem como as adaptações realizadas e os métodos adotados.

O escopo das explicações neste capítulo limita-se ao desenvolvimento do código fonte, ou seja, não engloba o projeto na sua totalidade.

A respeito do paradigma usado para efetivamente desenvolver o projeto, foi escolhido como modelo principal o orientado a objetos, mais comumente conhecido por Programação Orientada a Objetos (POO).

Além do POO também são utilizados os paradigmas funcional e estruturado no *backend* do sistema, de acordo com o benefício observado para uso nos códigos de implementação das classes. Implicitamente há a manipulação de processamento vetorial. Para o *frontend* há ainda o modelo de programação orientado a eventos.

A metogologia POO era a mais utilizada para construção de sistemas backend no momento da construção desse trabalho. O desempenho não situava-se entre os melhores porém a inteligibilidade mais eficaz e o potencial de produtividade considerando a maturidade de seus padrões - comparado com os outros modelos - e hardware disponíveis foram os fatores cruciais para sua popularidade.

No sistema Acomeds, a camada de arquitetura da aplicação está elaborada visando a POO enquanto que as outras modalidades estão esporadicamente aplicadas na implementação das especializações dos objetos (quando vantajoso) sem que houvessem quaisquer impactos nas camadas abstratas do sistema.

2.2.1 PROGRAMAÇÃO ORIENTADA A OBJETOS

Essa metodologia visa representar cada elemento em termos de um objeto, ou classe(MACHADO, 2018). O objetivo disso é facilitar o entendimento das informações sentro tratadas através de abstrações do que se percebe no mundo real.

> Um objeto é uma estrutura que possui estados, comportamentos, identidade e possui algum significado para o problema em questão."(BOOCH, 1991 apud NAGAO, 2018)(RUMBAUGH et al., 1991 apud NAGAO, 2018).

Uma classe permite armazenar informações através de atributos, quando são instanciadas. Usualmente, as linguagens de programação (incluindo a C++) definem a classe como um modelo para ser instanciado, e quando isso ocorre há a formação de um objeto. É possível obter conhecimento sobre esses objetos através dos métodos definidos para a classe.

Portanto, essa metodologia permite que os dados que transitam em um programa possam ser transformados em informação e conhecimento através dos objetos que trafegam nele.

2.2.2 PROGRAMAÇÃO ESTRUTURADA

O paradigma estruturado filtra as informações através de estruturas de dados, que tem um papel mais simples que o da classe : cada estrutura de dados armazena somente informações, sendo que a obtenção de conhecimento pode opcionalmente e de maneira não usual ser obtida através de uma informação que aponta para uma função que realiza o trabalho. Em geral a maneira de obter conhecimento dessas informações dá-se pela definição de funções denominadas globais que podem ser acessadas de qualquer lugar do sistema, sendo assim, o controle dessas informações tende a ser mais centralizado que POO.

O modelo estruturado disseminou-se antes do POO principalmente pelo potencial elevado de execução mais veloz de programas para microcomputadores com poder de processamento muito limitados e com foco em desempenho vertical [1], mas tem sido substituído na arquitetura de sistemas pelo mesmo ao longo dos anos. Contudo, sua utilização ainda pode ser bem aproveitada na engenharia do código uma vez que permite melhores otimizações de velocidade e praticidade de código para arquiteturas de máquinas baseadas em x86, principalmente para trechos de código monolíticos.

2.2.3 PARADIGMA FUNCIONAL

O paradigma funcional visa resolver operações computacionais através de funções matemáticas. Esse paradigma busca tratar os dados de maneira a serem imutáveis afim de otimizar as operações *multithread*.

[1] Nota : Até meados do início dos anos 2000 era muito raro encontrar no varejo, devido ao preço, microcomputadores com *Central Processing Unit* (CPU)s multinúcleo e *Graphics Processing Unit* (GPU)s que tivessem capacidade de processamento além da simples composição de pixels em uma tela

No contexto da linguagem C++, por exemplo, é possível executar trechos de códigos que se utilizam desse paradigma através de funções lambda. Dessa maneira é possível usufruir desse paradigma mesmo que se utilizando de outro paradigma como sendo o predominante.

Apesar de estar se disseminando mais nos últimos anos, esse modelo não é novo e sempre foi ofuscado pelo alto custo computacional para sistemas que se beneficiam de maior poder de processamento sequencial. Com o avanço das tecnologias de microcomputadores, que estão cada vez mais aumentando horizontalmente o seu desempenho, [2], esse paradigma tem ficado cada vez mais popular.

2.2.4 PROGRAMAÇÃO VETORIAL

O modelo de processamento vetorial que implicitamente é utilizado pelas GPUs consiste basicamente na divisão de uma tarefa em muitas pequenas tarefas. Assemelhando-se muito com a mistura de um modelo estruturado que visa a subdivisão de tarefas para serem executadas de maneira similar à execução num modelo funcional. Com o decorrer do avanço tecnológico, há indícios de que o próprio chip central de uma GPU irá ser paralelizado assemelhando-se a um modelo funcional que subidivide tarefas baseadas em modelos funcionais.

A GPU é utilizada constantemente nos computadores atuais: podem ser *offboard* , *onboard* e integradas diretamente no chip que contém a CPU. Ela é responsável principalmente por renderizar imagens em um display - normalmente usando API OpenGL, DirectX ou Vulkan, sendo o OpenGL aceito por quase todos os dispositivos disponíveis no mercado - mas também pode ser utilizada para manipular dados arbitrários pelo desenvolvedor. A tecnologia escolhida para desenvolvimento do sistema permite tanto o uso de bibliotecas javascript para essa manipulação (como o Canvas utilizado na construção dos gráficos de acompanhamento) quanto bibliotecas nativas escritas para a linguagem C++, como o OpenCl [3] que é aceito por quase todos os processadores gráficos disponíveis no varejo.

[2] NOTA : Para exemplo, no momento da concepção desse trabalho um consumidor poderia optar por comprar no varejo produtos voltados para as categorias *mainstream* e *high end* :

1. CPU desktop AMD Ryzen 1800X, com 8 núcleos e 16 threads via *Simultaneous multithreading* (SMT), de alto desempenho.

2. Um *smartphone* com uma CPU Qualcomm Snapdragon 805 de 4 núcleos de alto desempenho.

3. Uma placa gráfica "RX 580"de processamento vetorial com 2304 *stream processors* , 32 *Raster Operation Pipeline* (ROP)s e 144 *Texture Mapping Unit* (TMU)s, totalizando em média 6.17 T*Floating-point Operations Per Second* (FLOPS) de precisão simples.

4. Um smartphone com uma GPU Adreno 330 com 32 *pipelines* (havia pouca informação disponível pois eram quase que na totalidade restritas).

[3] http://blog.qt.io/blog/2015/04/20/qt-quick-with-the-power-of-opencl-on-embedded-linux-devices/

2.2.5 PROGRAMAÇÃO ORIENTADA A EVENTOS

O desenvolvimento *frontend* quase sempre utiliza-se da metodologia de programação orientada a eventos, ou uma mistura disso com outro modelo. Esse paradigma consiste em configurar eventos que serão disparados a partir de uma interação externa com o sistema, isto é, os eventos são criados com o propósito de esperar que algo aconteça, sinalizando que devem ser executados.

O disparo um evento poderia ser, por exemplo :

- O anúncio do sistema operacional para o sistema de que o usuário pressionou um botão do mouse ou do teclado.
- A movimentação do mouse na tela.
- Manter o mouse ou o dedo sobre determinado ponto interativo na tela.
- O indicativo de pouca luminosidade captada pelo sensor de luminosidade, indicando que um usuário poderia ter parado de interagir com o dispositivo.

2.2.6 MVC

Model, View and Controller (MVC) é um acrônimo de *Model View Controller*. Esse padrão visa separar as camadas de modelo - que consiste na manipulação de regras do negócio - da camada de visualização - que consiste nas telas do aplicativo - através de uma terceira camada de controle que é responsável pela comunicação entre as camadas de visualização e modelo. Um projeto é considerado bem padronizado em MVC quando a camada de modelo nunca se comunica diretamente com a camada de controle.

O MVC é utilizado em muitos projetos devido à arquitetura que possui, o que possibilita a divisão do projeto em camadas muito bem definidas. **??**

2.2.7 MODELO DE DADOS

Um modelo entidade relacionamento (Modelo Entidade Relacionamento (MER)) é um processo que visa retratar como os dados de um sistema estarão relacionados para armazenamento. Essa retratação pode ser conceitual, lógica ou física sendo que a representação conceitual está mais próxima da realidade do negócio enquanto que a representação física está mais próxima dos requisitos computacionais, e desse modo, a representação lógica representa o meio-termo.

Para efetivamente representar graficamente um modelo baseado no MER é criado um diagrama entidade relacionamento (Diagrama Entidade Relacionamento (DER)).

Para este projeto é empregado a notação de Peter Chen(CHEN, 1979) para construção do DER, que visa representar conceitualmente os dados do sistema.

Os elementos de um DER - que segue a notação original proposta por Peter Chen - são :

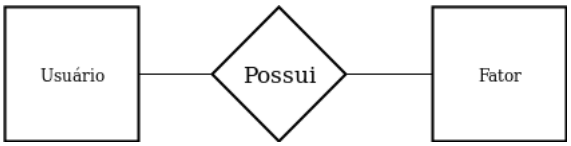
1. Retângulos representam as entidades .
2. Losangos representam os relacionamentos entre as entidades.
3. Elipses representam os atributos das entidades.
4. Linhas unem os elementos do modelo. Podem indicar a cardinalidade do relacionamento.

A cardinalidade pode possuir valores 1 ou N, sendo que N significa muitos.

O diagama presente neste documento não emprega o uso de elipses por se tratar de um modelo que visa organizar os dados de maneira abstrata para auxiliar a construção de um JSON Schema.

Um exemplo de um DER pode ser visualizado na figura 2;

FIGURA 2. EXEMPLO DE UM DER.



Fonte. O AUTOR (2017).

2.3 UML - DOCUMENTAÇÃO

Linguagem de Modelagem Unificada[4](UML, 2017) é um padrão amplamente utilizado para expecificar e documentar sistemas de informação. Esse padrão é aceito e publicado pela *International Organization for Standardization* (ISO).

Embora não force nem dependa de metodologias ou processos, é amplamente utilizada - e na maioria das vezes - para documentação de sistemas que seguem o modelo de orientação a objetos.

A visualização da informação no UML é obtida através da confecção de diagramas que permitem entender facilmente como o sistema será feito (estrutura), como as atividades serão realizadas e qual será o comportamento para cada ato executado. Cada uma das 3 partes está representada neste documento de acordo com a lista :

1. Estrutura através de um diagrama de classes.
2. Comportamento através de diagramas de caso de uso e fluxos de eventos.
3. Interação através de diagramas de sequência.

[4] Do inglês, *Unified Modeling Language* (UML) - *Unified Modeling Language*.

2.3.1 DIAGRAMA DE CLASSES

Um diagrama de classes representa as classes que serão utilizadas para a formação de instâncias (objetos) no sistema. É possível observar sua formação e relação com outras classes.

O diagrama de classes pode ser construído afim de oferecer uma entre as três perspectivas :

1. Conceitual : apresenta somente um conceito da organização entre as classes do sistema.

2. Especificação : apresenta os principais métodos das classes e dá ênfase às principais interfaces do sistema.

3. Implementação : especifica detalhadamente as classes existentes no sistema.

O diagrama utilizado neste trabalho visa oferecer a perspectiva de implementação, nomeando atributos e métodos. Os métodos triviais de acesso às propriedades, também conhecidos como *getters*[5] e *setters*[6], estão omitidos do diagrama.

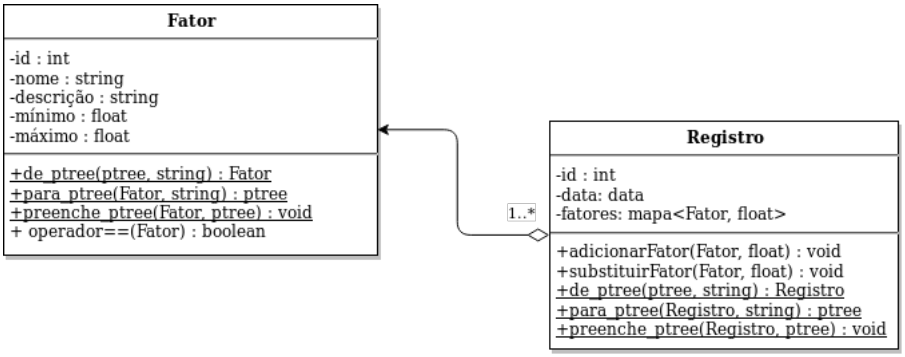
Um diagrama de classes possui basicamente 2 elementos :

4. Entidades : representam as classes contendo no topo o seu nome, no meio os seus atributos e na base os métodos.

5. Relacionamentos : indicam a quantidade de instâncias de determinada entidade que a instância da entidade que aponta a seta pode conter.

A figura 3 exemplifica um diagrama de classes.

FIGURA 3. EXEMPLO DE UM DIAGRAMA DE CLASSES SIMPLIFICADO.



Fonte. O AUTOR (2017).

[5] Método que consiste apenas em retornar o valor de um atributo.
[6] Método que consiste apenas em alterar o valor de um atributo.

2.3.2 DIAGRAMA DE CASO DE USO

Representa as possíveis ações que podem ser tomadas pelos atores e os casos de uso, bem como as relações que podem ocorrer entre elementos do diagrama.

Um ator é representado por um boneco e indica um usuário que interage com o sistema.

Um caso de uso é representado por uma elipse com um nome e define uma função abstrata do sistema.

A relação entre um ator e um caso de uso ocorre através de uma seta aberta de associação, que representa uma funcionalidade do sistema, do ponto de vista do ator.

Atores e casos de uso podem ser relacionados homogeneamente através de uma seta fechada. Indica que um terá as características herdadas do outro além das suas particulares.

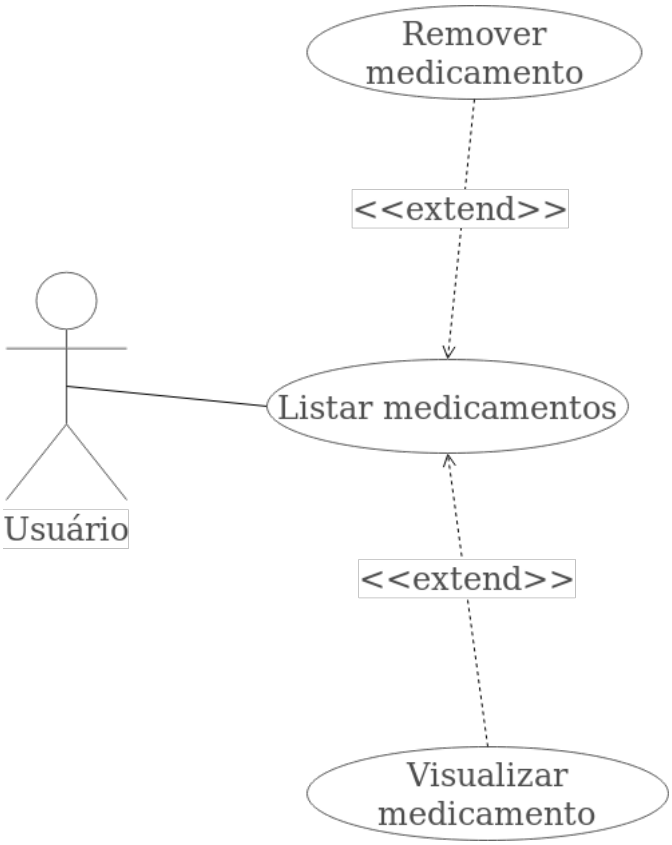
Há ainda dois relacionamentos para casos de usos : include e extend. Ambos são setas nomeadas e tracejadas

O include indica que o caso de uso apontado sempre é executado pelo caso de uso que aponta.

O extend indica que o caso de uso que aponta pode ou não ser executado pelo caso de uso apontado.

A figura 4 exemplifica um caso de uso.

FIGURA 4. EXEMPLO DE UM DIAGRAMA DE CASO DE USO SIMPLIFICADO.



Fonte. O AUTOR (2017).

2.3.2.1 FLUXO DE EVENTOS

Demonstra textualmente por meio de passos o comportamento esperado de acordo com cada ação executada num fluxo denominado básico.

Fluxos secundários podem ser adicionados para eventos que possam ocorrer entre os passos do fluxo básico.

Fluxos alternativos são identificados pelo prefixo **A** com um suffixo numérico, por exemplo : **A1** . Esses fluxos servem para detalhar um fluxo diferente decorrente de alguma condição que possa existir na execução de algum passo.

Subfluxos são identificados pelo prefixo **S** com um suffixo numérico, por exemplo : **S1**. Têm como finalidade decompor fluxos de eventos e executar operações simples, melhorando a legibilidade.

Exemplo de um fluxo básico de eventos :

Caso de uso Cadastrar música

Ator : Usuário

Fluxo básico

1. O usuário procura pela música desejada

2. Usuário clica em adicionar música na lista de músicas

3. A música é adicionada

2.3.3 DIAGRAMA DE SEQUÊNCIA

Representa de maneira sequencial a troca de mensagens entre instâncias de classes, atores e componentes para execução de uma determinada funcionalidade de um sistema.

É possível então identificar a ordem dos acontecimentos para que o objetivo de um caso de uso seja atingido.

Um ator interage com o sistema dando início ao processo de troca de mensagens descrita no diagrama de sequência.

Essa solicitação é entregue à instância de objeto correspondente que então dará continuidade ao processo executando alguma destas duas opções:

1. Enviando outra solicitação para uma outra instância de objeto.
2. Retornando o resultado esperado se aplicável.

Cada instância que participa de um diagrama de sequência possui uma linha de vida que possibilita acrescentar as trocas de mensagens sequencialmente de cima para baixo.

As instâncias utilizadas nos diagramas nesse documento estão divididas em :

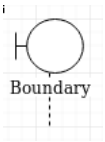
- Ator é o usuário do sistema, comunica-se somente com objetos *Boundary*. Representado na figura 5.

FIGURA 5. REPRESENTAÇÃO DE ATOR NUM DIAGRAMA DE SEQUÊNCIA.

Fonte. O AUTOR (2017).

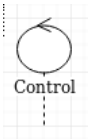
- *Boundary object* representa algo que interage diretamente com o ator e portanto é comumente utilizado para representar interfaces do sistema. Além do ator, esse objeto pode se comunicar com o *Control object* que efetivamente delegará as mensagens para o processamento necessário. Representado na figura 6.
- *Control object* representa um objeto que fará a ponte entre o *Boundary* e os outros objetos do sistema. Representado na figura 7.
- *Entity object* representa entidades e portanto está relacionado à manipulação dos dados do sistema. Representado na figura 8.

FIGURA 6. REPRESENTAÇÃO DE *BOUNDARY OBJECT* NUM DIAGRAMA DE SEQUÊNCIA.



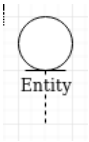
Fonte. O AUTOR (2017).

FIGURA 7. REPRESENTAÇÃO DE *CONTROL OBJECT* NUM DIAGRAMA DE SEQUÊNCIA.



Fonte. O AUTOR (2017).

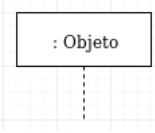
FIGURA 8. REPRESENTAÇÃO DE *ENTITY OBJECT* NUM DIAGRAMA DE SEQUÊNCIA.



Fonte. O AUTOR (2017).

- Objeto representa um objeto no diagrama, comumente utilizado para representar a camada de modelo. Neste documento esse tipo de objeto é usado para representar o modelo. Representado na figura 9.

FIGURA 9. REPRESENTAÇÃO DE OBJETO NUM DIAGRAMA DE SEQUÊNCIA.



Fonte. O AUTOR (2017).

É possível demarcar áreas retangulares onde cada cabeçalho indicará um tipo diferente de funcionamento (ver tabela 1).

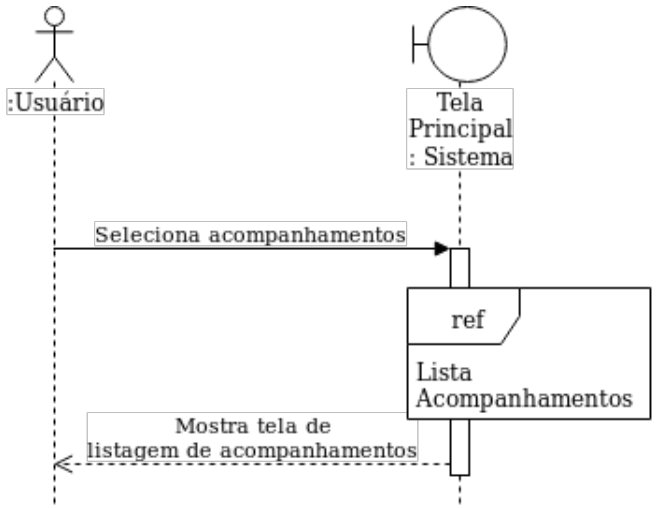
A figura 10 exemplifica um diagrama de sequência.

TABELA 1. TABELA DE CABEÇALHOS PARA CAIXAS ESPECIAIS DE UM DIAGRAMA DE SEQUÊNCIA.

Cabeçalho	**Descrição**
`ref`	Referencia outro diagrama de sequência.
`opt`	Executado somente se uma determinada condição for válida.
`alt`	Pertence a um fluxo alternativo.
`loop`	Executado repetidamente respeitando uma condição.

Fonte. O AUTOR (2017).

FIGURA 10. EXEMPLO DE UM DIAGRAMA DE SEQUÊNCIA SIMPLIFICADO.



Fonte. O AUTOR (2017).

2.4 MATERIAIS/FERRAMENTAS

O critério de seleção das tecnologias envolvidas está fundamentado na sua política de utilização e customização, se haviam versões gratuitas por tempo indeterminado disponíveis e a possibilidade de execução do aplicativo final em mais de uma plataforma. Foi considerado também a preferência e conhecimento da equipe.

O *framework* (conjunto de ferramentas de desenvolvimento) Qt(QT, 2017) foi escolhido pois foca em ter como especialidade a disponibilização de ferramentas multi-plataforma para desenvolvimento de aplicativos. As licenças para uso dos módulos podem ser *GNU Lesser General Public License v3.0* (LGPLv3), *GNU General Public License* (GPL) 2, GPL 3 e comercial.

As plataformas disponíveis para implementação usando Qt eram : Android a partir da API 16, Windows Phone a partir da versão 7, IOS, MacOS, Windows e sistemas baseados em Unix (como o GNU/Linux).

A respeito dos componentes Qt no desenvolvimento da aplicação, aplica-se a licença LGPLv3 (*Integrated Development Environment* (IDE) Qt Creator e Qt Quick). Entretanto,

a licença GPL (2 e 3) se aplicam na utilização do módulo Qt Charts.

2.5 QT QUICK

Para o desenvolvimento do aplicativo é utilizado o módulo Qt Quick 5.9 que é o módulo estável mais atual disponibilizado pela companhia The Qt Company , proprietária do Qt.

Essa ferramenta foi escolhida após um comparativo entre soluções similares que também visavam o desenvolvimento de aplicações multiplataforma sendo o principal motivo de escolha o desempenho nativo oferecido para todas as plataformas quando aplicável e disponível.

Considera-se que uma ferramenta entrega performance nativa quando ela busca utilizar componentes e comportamentos já disponíveis na plataforma em que o software gerado se destina visando a melhoria de velocidade uma vez que se disponíveis para uso pelo programa essas implementações já estariam pré-compiladas e otimizadas para uso no dispositivo.

Como comparativo didático : quando um componente era customizado ou emulado pelo software sendo utilizado, o dispositivo obrigava-se a calcular as formas e características usando as regras estabelecidas pelo software, mas se esse mesmo componente fosse usado nativamente com as implementações já disponíveis no dispositivo, o software apenas especificaria as características desejadas e o dispositivo iria se encarregar e usar pré-compilações já possivelmente existentes para que não houvesse tanto gasto de ciclos de processamento executando as regras do componente customizado ou emulado, ou seja, usaria o código já otimizado para a plataforma.

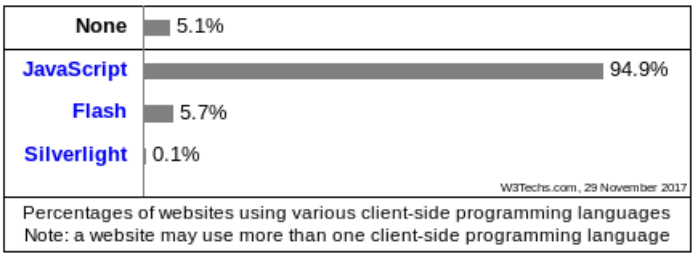
O motivo de se escolher uma ferramenta que oferecesse desempenho e comportamento nativos na execução das tarefas é o fato de normalmente as instituições terem de desenvolver várias versões do mesmo sistema/aplicativo para plataformas diferentes por causa da diferença de velocidade no processamento de instruções computacionais em dispositivos diferentes e o comportamento não adaptado ao dispositivo que podem apresentar quando utilizadas ferramentas que emulam ou customizam manualmente componentes e comportamentos.

O módulo Qt Quick é voltado para aplicações multiplataformas de alto desempenho. As bibliotecas deste módulo são disponibilizadas para uso nas linguagens QML e C++. Quando não disponíveis automaticamente pelo instalador do Qt, o usuário deve configurar manualmente os compiladores e bibliotecas necessárias para geração dos arquivos binários executáveis para a plataforma desejada.

QML é uma linguagem própria do Qt com sintaxe similar à linguagem CSS e com

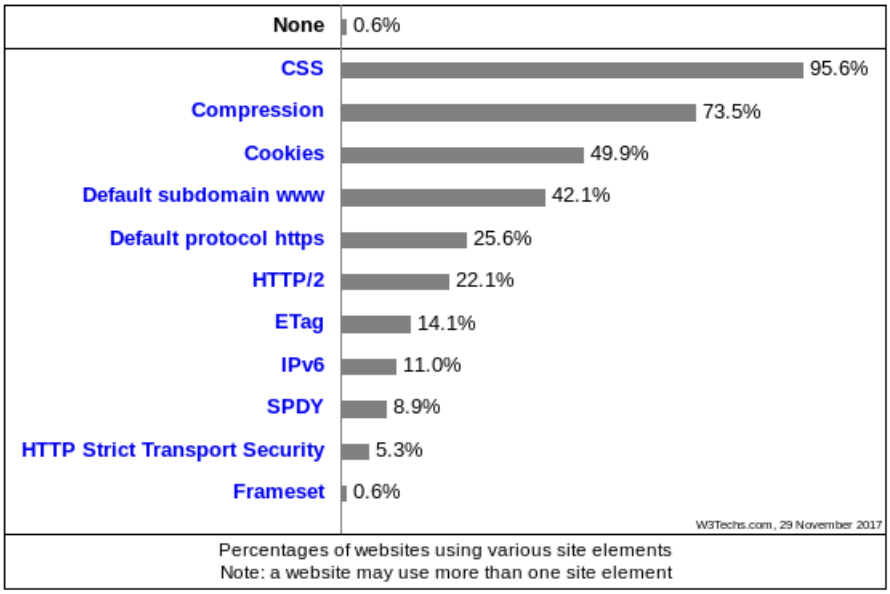
suporte nativo à linguagem Javascript, sendo que as duas últimas linguagens já encontram-se estabelecidas no mercado como linguagens altamente difundidas, como mostra as figuras 11 e 12.

FIGURA 11. "PORCENTAGENS DE WEBSITES USANDO DIVERSAS LINGUAGENS DE PROGRAMAÇÃO PARA *CLIENT-SIDE*", TRADUÇÃO NOSSA.



Fonte. (W3TECHS, 2017a).

FIGURA 12. "PORCENTAGENS DE WEBSITES USANDO DIVERSOS ELEMENTOS DE SITE. NOTA: UM WEBSITE PODE USAR MAIS DE UM ELEMENTO DE SITE.", TRADUÇÃO NOSSA.



Fonte. (W3TECHS, 2017b).

2.5.1 C++

C e C++ são linguagens já estabelecidas no mercado como linguagens amplamente utilizadas no desenvolvimento de sistemas e com frequência escolhidas para sistemas que exigem performance nativa. Essas linguagem são comumente citadas em conjunto pois a linguagem C++ foi criada baseando-se na linguagem C e suporta códigos escritos na linguagem C com uma diferença muito pequena de características, que podem ser resumidas principalmente em : na maneira de manipular conversões de tipo implícitas, a existência do tipo bool e na maneira de declarar structs. Quando diferentes, há sempre uma maneira de adaptar facilmente o código escrito em C para ser compilado em um compilador feito para C++ porém o contrário não é verdadeiro.

Foram utilizados os padrões C++14[7] e C++11[8], a depender da disponibilidade no compilador disponível tendo como prioridade o padrão C++14. O compilador utilizado foi o *GNU Compiler Collection* (GCC), sendo :

1. GCC 4.9 64bits para sistemas baseados em Unix com arquiteturas amd64 e arm8.
2. GCC 4.9 32 bits para sistemas baseados em Unix com arquitetura armv7.
3. *Minimalist GNU for Windows* (MinGW)-GCC 4.8 32 bits para sistemas Windows.

Além das bibliotecas descritas neste documento, também foram utilizadas as bibliotecas padrões das linguagens C e C++, incluindo a *Standard Template Library* (STL) que provê várias funcionalidades e estruturas de dados.

2.5.1.1 C++ E QML

As duas linguagens são capazes de se comunicarem entre si nativamente pelo *framework* uma vez que o QML é construído usando a linguagem C++ primariamente.

2.5.1.2 BOOST

Para conversão dos dados de JSON para C++ e vice-versa, testes unitários com C++ e tipos de variáveis opcionais é utilizado o conjunto de bibliotecas Boost. Essas bibliotecas já são amplamente utilizadas[9] no contexto da linguagem C++ e alguns de seus

[7] Um padrão que aprimora as funcionalidades do C++11, mais informações disponíveis em https://isocpp.org/wiki/faq/cpp14 (acessado em 30 de novembro de 2017).

[8] Um padrão mais recente para a linguagem que visa disponibilizar recursos similares aos das linguagens de mais alto nível além de melhorias e inovações no conjunto de bibliotecas já existentes. Mais informações disponíveis em https://isocpp.org/wiki/faq/cpp11 (acessado em 30 de novembro de 2017).

[9] Demonstrações de uso: http://www.boost.org/users/uses_shrink.html , http://www.boost.org/users/uses_open.html e http://www.boost.org/users/uses_inhouse.html. Endereços acessados em 30 de novembro de 2017.

componentes inclusive já foram incluídos como extensões do padrão C++03, que mais tarde se tornaram parte do padrão C++11 (OPENSTD, 2003).

2.5.1.3 INTERPRETAÇÕES DE FÓRMULAS MATEMÁTICAS

Na realização dos cálculos de expressões matemáticas é utilizada a biblioteca ExprTk e para compilações voltadas ao ambiente Windows é utilizada a biblioteca MuParser devido a limitações do compilador MinGW-GCC 4.8 32 bits que não conseguia indexar todas as seções necessárias no arquivo único *header* da biblioteca ExprTk.

Embora a biblioteca MuParser possa ser utilizada também para ambientes não Windows (Unix nesse caso), não é a preferência quando ambas estão disponíveis para utilização porque :

1. A biblioteca MuParser utiliza padrões revogados da linguagem C++ (em suma o tipo de variável *auto_ptr*).
2. Ao contrário da MuPaser, a biblioteca ExprTk não apresenta *warnings* de compilação, assegurando melhor o seu funcionamento.

2.5.1.4 NOTIFICAÇÕES *DESKTOP*

A biblioteca Snorenotify(KDE, 2017) é utilizada para disparar notificações para ambientes *desktop*. Trata-se de uma biblioteca multiplataforma que possibilita o envio de notificações locais para Windows, Unix e Mac. Também é possível enviar notificações remotas para dispositivos *mobile* Android e IOS.

É possível notificar o usuário *desktop* com a classe QSystemTrayIcon através de um ícone na área de *system tray* da interface gráfica sendo utilizada. Entretanto, essa biblioteca mostra-se mais interessante que a utilização da classe QSystemTrayIcon por se tratar somente de notificações, enviando uma mensagem mais adequada ao contexto para o usuário.

2.5.2 ARMAZENAMENTO DOS DADOS

Para armazenamento dos dados utiliza-se a notação JSON que é em suma o modelo de representação de objetos utilizado na linguagem Javascript. E para formatar esses dados é utilizado o padrão JSON Schema(SCHEMA, 2017).

Para a formação dos documentos em JSON Schema foi idealizado um diagrama básico de entidade e relacionamento.

O documento JSON gerado no sistema é armazenado em um arquivo, e portanto, não há uso de um banco de dados. Isso ocorre porque dispositivos com menor poder de processamento[10] irão se beneficiar da velocidade de carregamento e atualização dos dados, uma vez que não há um volume demasiadamente grande de dados[11] e iniciar um banco de dados, mesmo que carregado na memória principal, poderia causar um impacto grande[12] para o usuário final.

2.5.2.1 JSON

É um modelo que visa representar objetos através do formato `"item": "valor"` sendo que as àspas podem ser omitidas.

É possível utilizar:

- chaves (`{}`) para demarcar valores complexos, podendo ser um objeto.
- colchetes (`[]`) para demarcar uma lista de valores, denotando que aque item representa uma lista.
- vírgulas para separar itens.

O código 1 mostra um exemplo de um documento JSON.

Código 1 – Exemplo de JSON.

```
1    "placa_mae" : {
2        "frequencias_de_ram_sem_oc" : [2133, 2400, 2666] ,
3        "nome" : "Placa X",
4        "tipo_ram" : "DDR4",
```

[10] Em 2017 era possível adquirir *smartphones* e computadores pessoais contendo uma CPU com frequência base menor que 1.5GHz e/ou poucos núcleos (<= 2) e/ou *Instructions Per Cycle*(IPC)(*Single Precision*(SP)) <= 8 e/ou IPC(*Double Precision*(DP)) <= 2. Exemplos de microarquiteturas de CPU que podem estar nessas condições: Bay Trail, Cortex-A, Bobcat, Kryo.

[11] O documento gerado pelo sistema é esperado para ocupar kilobytes de dados em uso comum. O espaço livre de armazenamento em *smartphones* e computadores pessoais novos é medido em gigabytes.

[12] Considerando o volume de dados sendo processado e o custo envolvido para uso de um banco de dados.

```
5           "canais_de_ram" : {
6               "slot_ram_0" : 1,
7               "slot_ram_1" : 2,
8               "slot_ram_2" : 1,
9               "slot_ram_3" : 2
10          },
11          "socket_cpu" : "AM4"
12      }
```

2.5.2.2 JSON SCHEMA

É um padrão de formatação, em forma de vocabulário escrito em JSON, de arquivos JSON que utiliza o formato `"vocábulo": "valor"` para definir como as informações serão estruturadas em um arquivo JSON.

Usando o conceito de orientação a objetos, cada classe seria especificada em um documento JSON Schema e a formação dos objetos seriam baseadas nesse documento para a criação de um novo documento JSON.

Cada item de um documento JSON estará representado entre chaves, dessa maneira todas as informações que estão dentro das chaves correspondem ao item, seguindo estas especificações :

- O campo `"vocábulo"` corresponde ao tipo de dado que será incluído para acrescentar informação no que está contido entre as chaves, por exemplo : o vocábulo "*título*"nomeia o item.
- O campo `"valor"` corresponde ao valor correspondente ao vocábulo em questão, por exemplo : "Pessoa"seria o valor do vocábulo "*título*"para representação de uma classe chamada Pessoa.

Afim de exemplificar o uso dessa notação, é apresentado uma estrutura usando o padrão JSON Schema e um possível documento JSON gerado baseado nessa notação :

- Um exemplo de JSON Schema para uma classe chamada "GPU"pode ser visualizado no código 2.

Código 2 – Exemplo de JSON Schema.

```
1       {
2           "título" : "GPU",
3           "tipo" : "objeto",
4           "propriedades" : {
```

```
5               "quantidade_de_memoria" : {
6                   "tipo" : "int",
7                   "descrição" : "tamanho em MiBs"
8               }
9           },
10          "obrigatório" : ["quantidade_de_memoria"]
11      }
```

- Um exemplo de JSON usando o JSON Schema previamente descrito para o item "GPU X"pode ser visualizado no código 3.

Código 3 – Exemplo de implementação de Json baseado no código 2

```
1       "GPU X" : {
2           "quantidade_de_memoria" : 4096
3       }
```

2.5.3 COMPILAÇÃO E INSTALAÇÃO

Para que o código possa ser executado no dispositivo desejado é necessário utilizar uma ferramenta de compilação e posteriormente instalar apropriadamente o produto compilado no dispositivo.

Neste capítulo é apresentado as ferramentas utilizadas com esse objetivo.

2.5.3.1 QMAKE

Para compilar e executar o aplicativo resultante, a IDE Qt Creator usa uma ferramenta de compilação chamada qmake como padrão.

Não é necessário mexer diretamente com a ferramenta, porém, é necessário configurar apropriadamente o arquivo com extensão `.pro` .

Nesse arquivo foram incluídas informações de acoplamento para bibliotecas externas (Boost, MuParser, exprTk), incluindo informações extras para plataformas que necessitavam de operações específicas (como o Android).

2.5.3.2 CMAKE

CMake(CMAKE, 2017) é uma ferramenta multiplataforma para gerencialmento da compilação de aplicações que possui uma presença considerável no mercado [13].

[13] Lista de projetos conhecidos que usam CMake : https://cmake.org/Wiki/CMake/Projects (acessado em 30 de novembro de 2017).

Essa ferramenta permite um gerenciamento mais prático e organizado da compilação de uma aplicação, uma vez que possibilita buscar e compilar automaticamente uma biblioteca requisitada pelo aplicativo e que forneça parâmetros de configuração para incluir no projeto que usa CMake.

Com a utilização do CMake torna-se mais prático também a utilização da biblioteca SnoreNotify para compilações voltadas à plataformas *desktop*.

Contudo, até o momento da conclusão desse projeto em sua versão inicial, era possível gerar um arquivo *Android Package Kit* (APK) para Android, mas não houve tempo suficiente para configurar sua instalação. Portanto, para Android, é necessário executar o modo de compilação e *deploy* usando o qmake.

2.5.3.3 HUNTER

Foi utilizado também o gerenciador de pacotes Hunter(RUSLO, 2017) para auxiliar e simplificar ainda mais a compilação usando CMake.

Esse gerenciador permite compilar e ligar pacotes antes de executar o mesmo processo para o projeto em questão. Há uma lista de pacotes[14] - cadastrados pelo desenvolvedor - que podem ser usados.

A grande vantagem de usar o Hunter é a diminuição de linhas no arquivo de configuração do CMake, além de melhorar a inteligibilidade dos processos de inclusão de bibliotecas sendo executados.

2.5.4 VERSIONAMENTO DO CÓDIGO

O software Git(GIT, 2017) é usado para gerenciamento de configuração de software do aplicativo , sendo que a hospedagem online utilizada para o Git é o website Gitlab(GITLAB, 2017).

Git é um software para gerenciamento de código fonte (*Source Code Management* (SCM)) bastante difundido e robusto[15] e foi escolhido por oferecer simplicidade, leveza e confiabilidade.

A ferramenta *online* Gitlab é uma plataforma de desenvolvimento de software no qual é possível armazenar projetos que usam o Git. Além de exercer o papel de repositório, também oferece recursos úteis para o projeto como : Gerenciamento de *bugs* (*issues*),

[14] Lista disponível em https://docs.hunter.sh/en/latest/packages.html, acessado em 30 de novembro de 2017.

[15] É possível verificar uma lista contendo as empresas principais na página principal do projeto git : https://git-scm.com/ (acessado em 30 de novembro de 2017).

continuous integration, gerenciamento de membros do projeto, estatísticas do projeto e configuração gratuita de permissões de acesso ao projeto.

3 RESULTADOS E DISCUSSÃO

Nesse capítulo são apresentados o resultado obtido (produto final) como aplicativo, a arquitetura e as especificações do aplicativo Acomeds.

Uma vez que o aplicativo visa - resumidamente - auxiliar no controle de medicamentos bem como no acompanhamento de indicadores de saúde, o nome escolhido para o aplicativo é Acomeds, que é a junção de partes das palavras acompanhamento (aco) e medicamentos (meds).

O aplicativo Acomeds é subdividido em dois módulos : o módulo de medicamentos e o módulo de acompanhamento de indicadores de saúde que será tratado como módulo de acompanhamento. Inicialmente o aplicativo foi idealizado para disponibilizar somente o módulo de medicamentos mas posteriormente, por solicitação do cliente, foi incluído também o módulo de acompanhamento.

Separar o sistema em módulos permite que uma característica não fique amarrada em outra, o que complicaria o uso do aplicativo por usuário que desejam utilizar somente 1 módulo por exemplo.

3.1 O APLICATIVO

O resultado final para o usuário nesse trabalho ocorreram através das interfaces (telas) que são disponibilizadas para utilizar os recursos do sistema.

Para os usuários de dispositivos móveis, as telas possibilitaram uma experiência de uso condizente com a realidade do usuário, uma vez que possuem características visuais e de funcionamento similares aos dispositivos móveis sendo utilizados.

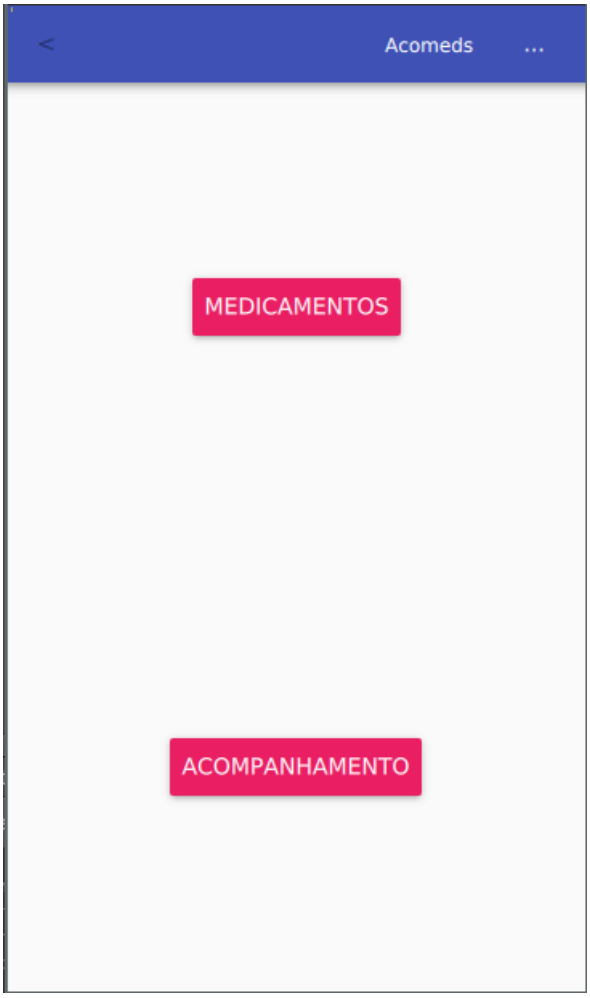
Para os usuários de computadores de mesa, as telas mantiveram-se idênticas. A similaridade com alguma interface gráfica nesse tipo de plataforma é inexata uma vez que podem ser variados e com aspectos diferentes.

3.1.1 TELA PRINCIPAL

A tela principal (figura 13) permite a seleção entre o módulo de medicamentos e o módulo de acompanhamento de indicadores por meio de dois botões principais. Há também um menu que possibilita acessar as configurações do sistema.

É disponibilizado também nessa tela e em todas as demais telas um botão com o símbolo “<” que representa a funcionalidade de “voltar” de uma tela para outra tela que foi aberta anteriormente.

FIGURA 13. TELA PRINCIPAL.



Fonte. O AUTOR (2017).

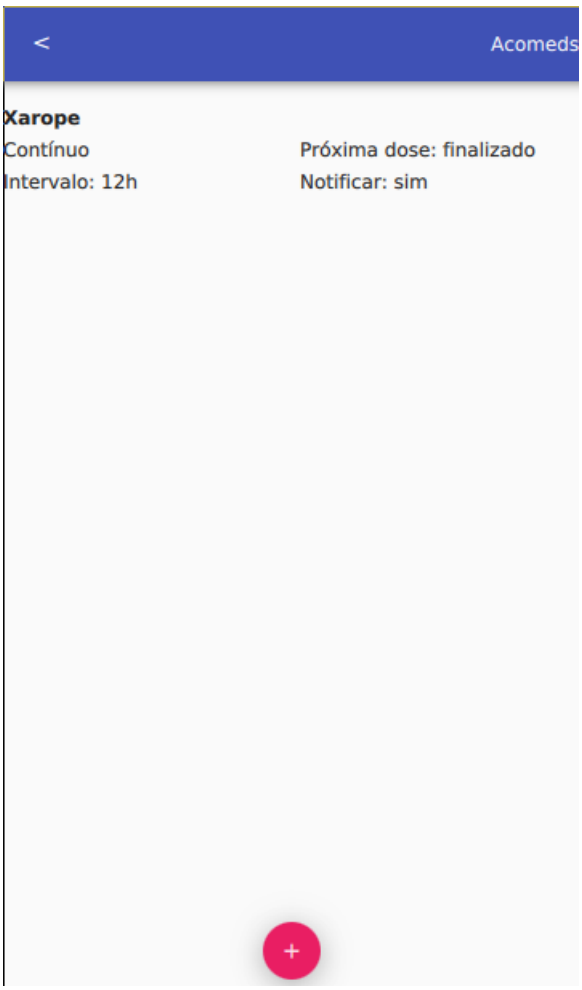
O botão voltar permanece com cor de botão desativado na tela principal - essa cor irá depender do tema escolhido e as cores primárias e secundárias, mas no exemplo é a cor preta - e portanto não é possível acioná-lo, e em todas as telas que podem aparecer em seguida esse botão torna-se acionável usando a cor primária de um botão de acordo com a configuração estabelecida.

O botão que executa a funcionalidade de retornar à uma tela anterior não é citado novamente uma vez que está presente em todas as telas com o mesmo propósito e é definido através de um header.

3.1.2 MEDICAMENTO - LISTAGEM

Após clicar na opção de Medicamentos na tela inicial, o usuário é direcionado para a primeira tela de medicamentos que contém a lista de medicamentos (figura 14).

FIGURA 14. LISTA DE MEDICAMENTOS.



Fonte. O AUTOR (2017).

Os medicamentos são listados por data decrescente de criação, sendo que o seu nome aparece em negrito na primeira linha do item e logo abaixo são especificados quatro informações para leitura facilitada, sendo elas respectivamente :

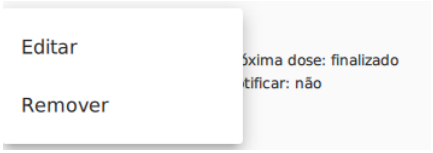
- Campo que indica o tempo restante válido para o medicamento, o valor pode ser “Restam :” seguido de uma quantidade de tempo usando siglas do *Système international (d’unités)* (SI) ou “Contínuo” para caso o medicamento seja configurado como contínuo.
- “Próxima dose“ : campo que indica qual o horário da próxima dose do medicamento ou então o texto “finalizado” caso este medicamento tenha chegado ao fim da sua duração.

- “Intervalo” : campo que indica o intervalo de tempo entre uma dose e outra para ingestão medicamento.
- “Notificar” : indica se o medicamento estava configurado para notificar o usuário por meio de notificação padrão do sistema sendo utilizado.

Além das informações listadas nos medicamentos há ainda outras 3 funcionalidades nesta tela :

1. Botão arredondado “+” : direciona para o formulário de criação de um novo medicamento.
2. Clique curto sobre um medicamento : mostra todas as informações disponíveis do medicamento.
3. Clique longo sobre o medicamento : mostra um submenu que permite editar ou remover o medicamento selecionado (pode ser visualizado na figura 15).

FIGURA 15. MENU DE CONTEXTO DE MEDICAMENTOS.

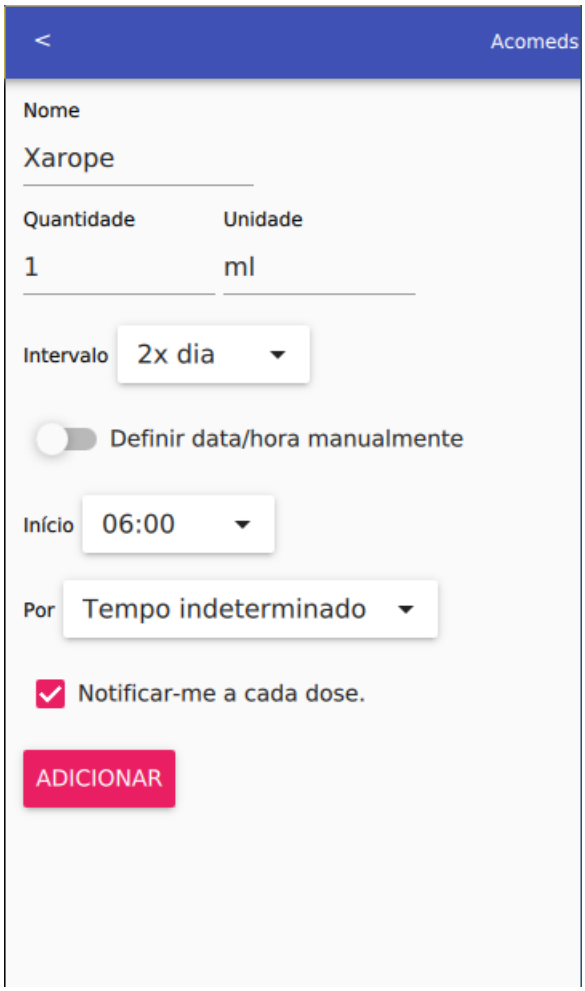


Fonte. O AUTOR (2017).

3.1.3 MEDICAMENTOS - CRIAÇÃO E EDIÇÃO

Ao clicar no botão "+" para criação de um novo medicamento ou ao clicar no botão "Editar" para edição de um medicamento já existente o aplicativo abre a tela de formulário de medicamentos representada pela figura 16.

FIGURA 16. CADASTRO DE MEDICAMENTO.



Fonte. O AUTOR (2017).

A diferença entre a criação e a edição é que quando a ação é de edição de medicamento, o formulário é preenchido automaticamente com as informações disponíveis e a data de criação do medicamento é mantida como a original.

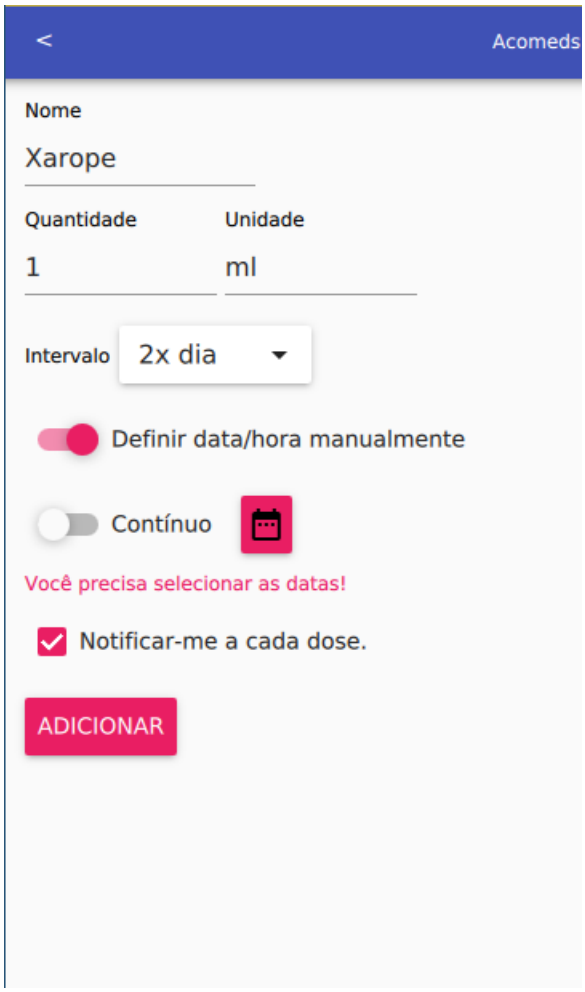
Nesta tela é possível cadastrar as informações do medicamento sendo que para a seleção de data e hora de início e fim do medicamento pode ser cadastrada de duas maneiras :

1. Simplificada : a maneira padrão definida automaticamente a partir de valores pré-estabelecidos quando o *switch* "Definir data/hora manualmente" estiver desativado.

2. Avançada : é possível que o usuário escolha a data e hora de cadastro manualmente, como demonstrado na figura 17.

    - O usuário pode escolher a data e hora de cadastro através de um *date picker*[1], representado na figura 18. O calendário está baseado no calendário padrão do módulo QtQuick.Controls 1.4 e sua aparência não se adapta automaticamente para as cores e formas esperadas para o tema escolhido. Isso ocorre porque o módulo 1.4 da ferramenta não foi projetado com a ideia de *guidelines* de design no momento de sua concepção.
    - As horas são escolhidasa cada vez que uma data é escolhida no calendário, através de *tumblers*, como representado na figura 19.

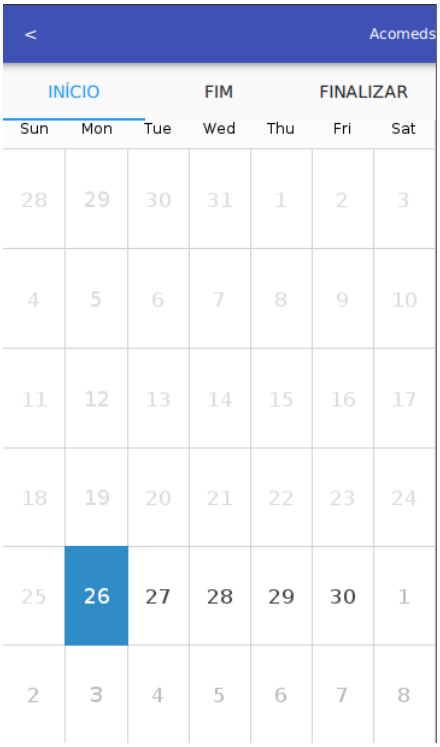
FIGURA 17. FORMULÁRIO DE MEDICAMENTO COM OPÇÕES AVANÇADAS.



Fonte. O AUTOR (2017).

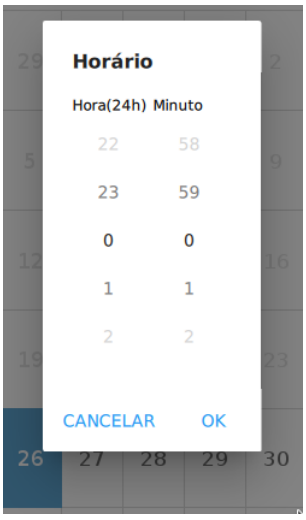
[1] jargão utilizado para funcionalidade que permite a seleção de datas através de um calendário selecionável.

FIGURA 18. CALENDÁRIO PARA SELEÇÃO DE DATAS DE MEDICAMENTO.



Fonte. O AUTOR (2017).

FIGURA 19. TUMBLER PARA SELEÇÃO DE HORÁRIO DE INÍCIO E FIM DE MEDICAMENTO.

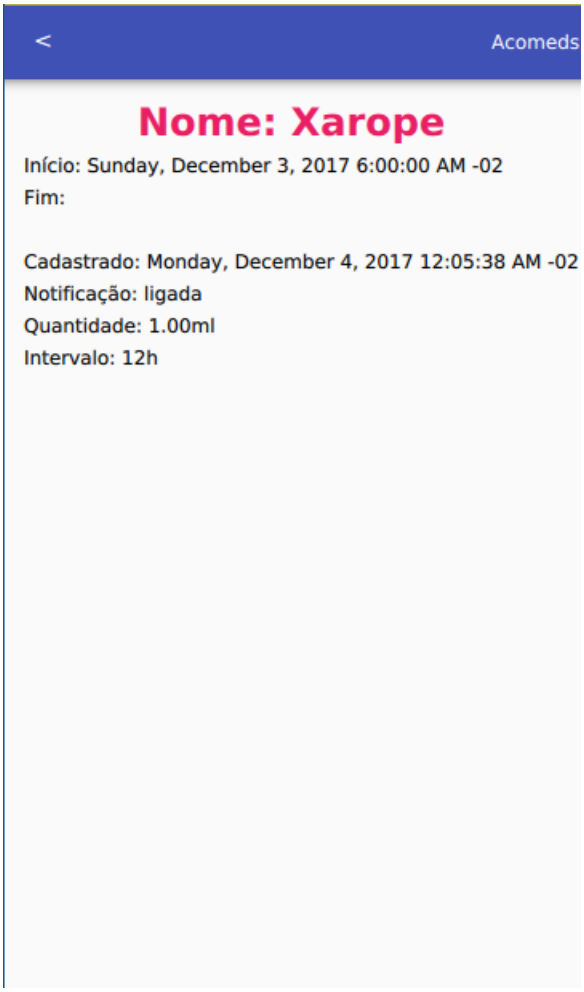


Fonte. O AUTOR (2017).

3.1.4 MEDICAMENTO - VISUALIZAÇÃO

Ao clicar em algum medicamento da lista é possível visualizar seus detalhes, como na figura 20.

FIGURA 20. VISUALIZAÇÃO DE MEDICAMENTO.

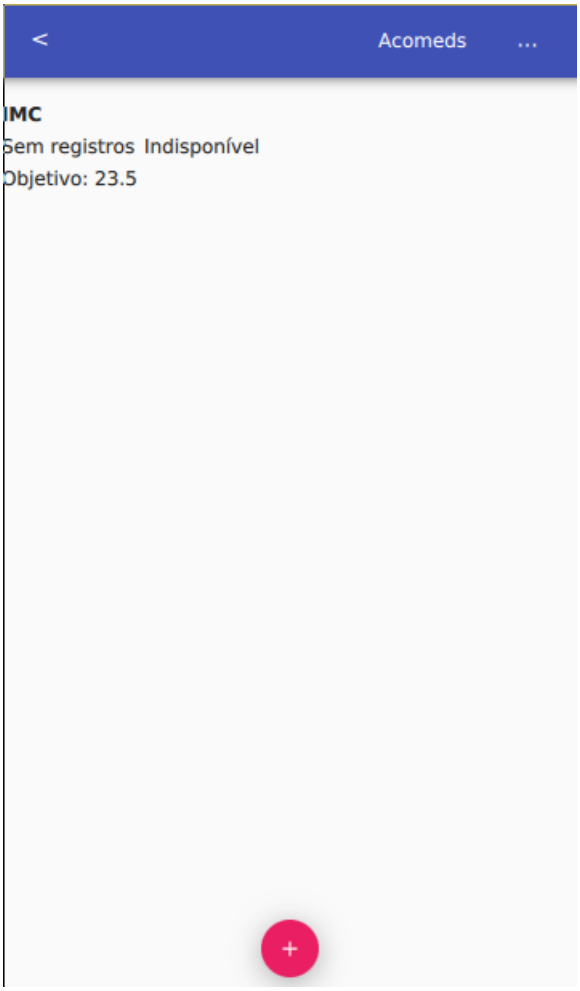


Fonte. O AUTOR (2017).

3.1.5 ACOMPANHAMENTO - LISTAGEM

Após clicar na opção de Acompanhamentos na tela inicial, o usuário é direcionado para a primeira tela de acompanhamentos que contém a lista de acompanhamentos (figura 21).

FIGURA 21. LISTA DE ACOMPANHAMENTOS.



Fonte. O AUTOR (2017).

Os acompanhamentos são listados por data decrescente de criação, sendo que o seu nome aparece em negrito na primeira linha do item e logo abaixo são especificados quatro informações para leitura facilitada, sendo elas respectivamente :

- Campo que indica a última vez que foi feito um registro no acompanhamento.
- Campo que indica o objetivo almejado.

Além das informações listadas nos acompanhamentos havia ainda outras 3 funcionalidades nesta tela :

- Botão arredondado “+” : direciona para o formulário de criação de um novo acompanhamento.
- Clique curto sobre um acompanhamento : mostra todas as informações disponíveis do acompanhamento.
- Clique longo sobre o acompanhamento : mostra um submenu que permite editar ou remover o acompanhamento selecionado (figura 22).

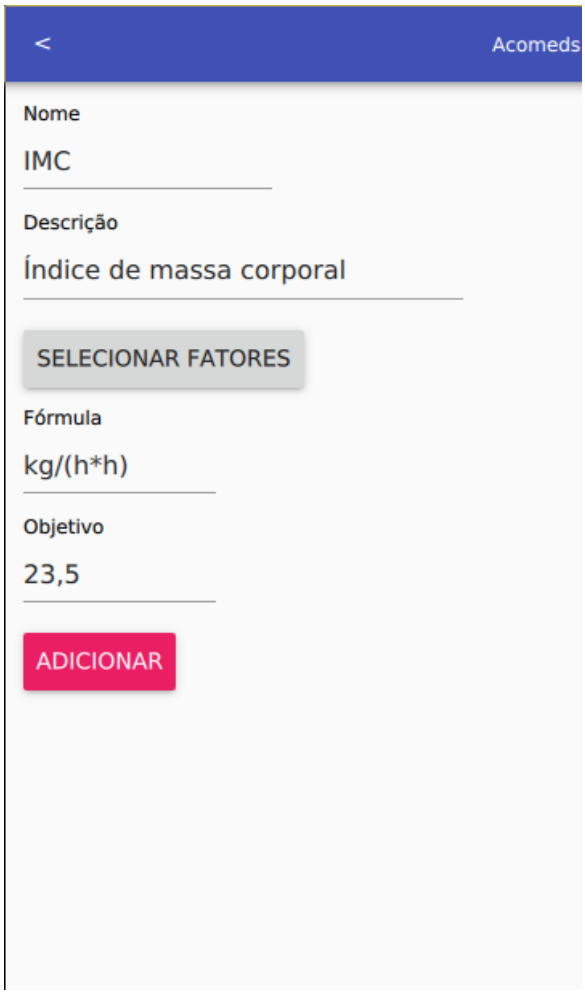
FIGURA 22. MENU DE CONTEXTO DA LISTA DE ACOMPANHAMENTOS.

Fonte. O AUTOR (2017).

3.1.6 ACOMPANHAMENTOS - CRIAÇÃO E EDIÇÃO

Ao clicar no botão “+” para criação de um novo acompanhamento ou ao clicar no botão “Editar” para edição de um acompanhamento já existente o aplicativo abre a tela de formulário de acompanhamentos representada pela figura 23.

FIGURA 23. CADASTRO E ALTERAÇÃO DE ACOMPANHAMENTO.

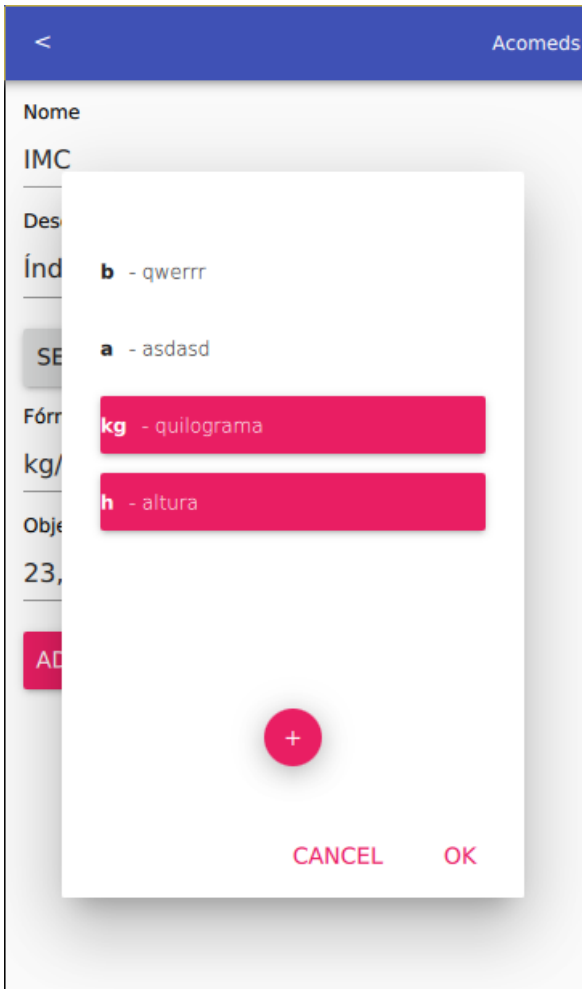


Fonte. O AUTOR (2017).

A diferença entre a criação e a edição é que quando a ação é de edição de acompanhamento, o formulário é preenchido automaticamente com as informações disponíveis e a data de criação do acompanhamento é mantida como a original.

Selecionando o botão "Selecionar fatores"é possível selecionar fatores para o acompanhamento (figura 24). Se o usuário desejar ele pode incluir um novo fator para ser posteriormente selecionado. O botão de novo fator mostraria a tela de cadastro de fatores.

FIGURA 24. SELEÇÃO DE FATORES PARA UM ACOMPANHAMENTO.



Fonte. O AUTOR (2017).

3.1.7 ACOMPANHAMENTO - VISUALIZAÇÃO

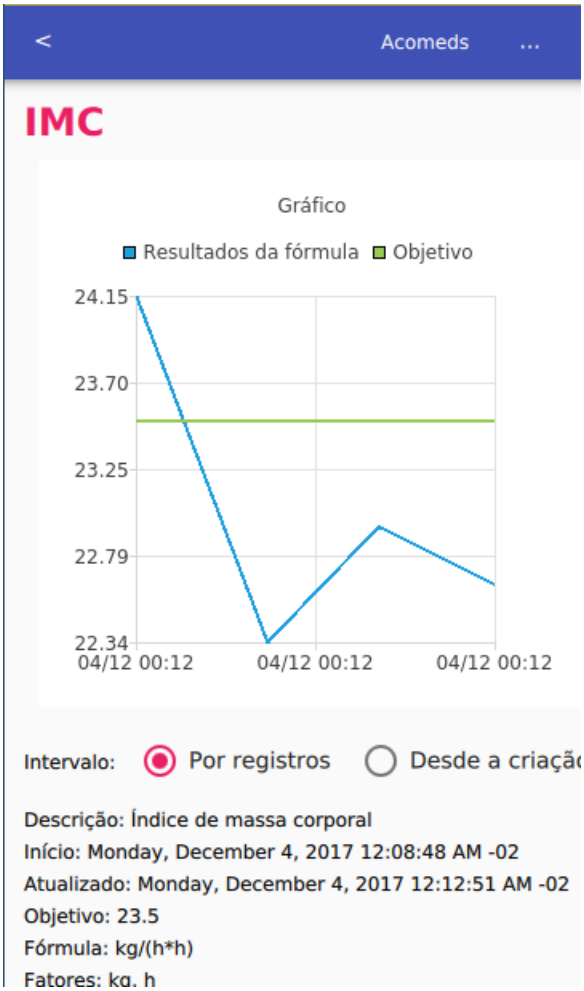
Ao clicar em algum acompanhamento da lista é possível visualizar seus detalhes.

Um gráfico também pode ser apresentado, sendo que é possível observar a progressão de duas maneiras :

1. Pelo período em que há registros (figura 25).
2. Pelo período desde a criação do acompanhamento (figura 26).

As figuras 25 e 26 representam a visualização de um acompanhamento.

FIGURA 25. VISUALIZAÇÃO DE ACOMPANHAMENTO - GRÁFICO MOSTRANDO SOMENTE O INTERVALO EM QUE HÁ REGISTROS.

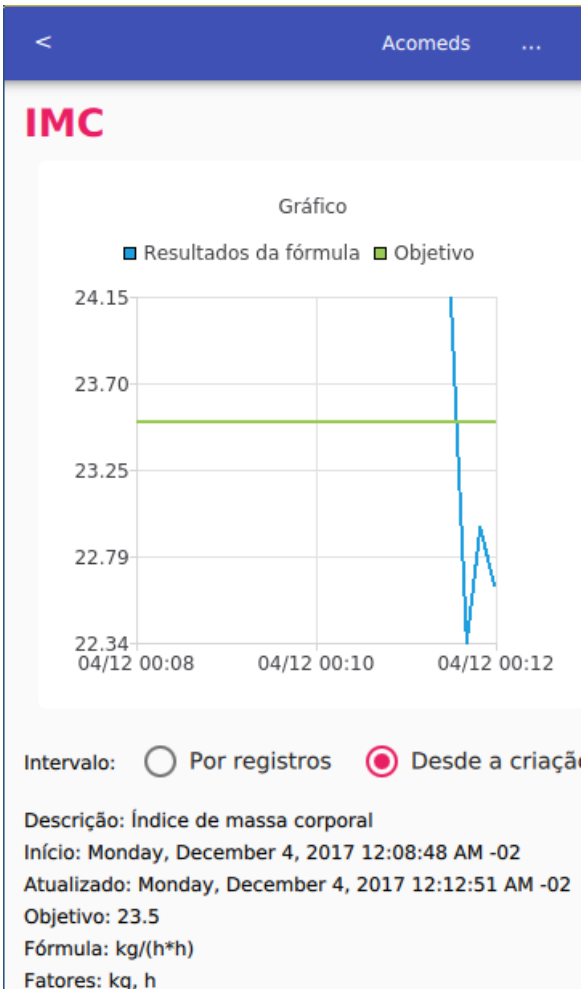


Fonte. O AUTOR (2017).

O gráfico é mostrado somente se as condições listadas forem verdadeiras:

- Mais de 1 registro existir, com os fatores preenchidos corretamente.

FIGURA 26. VISUALIZAÇÃO DE ACOMPANHAMENTO - GRÁFICO MOSTRANDO O INTERVALO DESDE QUE FOI CRIADO.



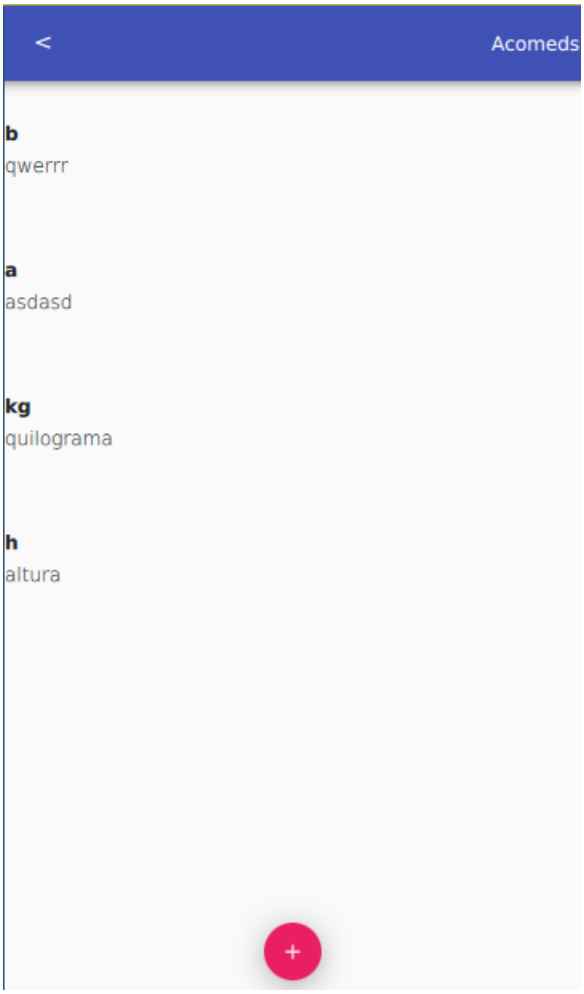
Fonte. O AUTOR (2017).

- Uma fórmula estiver cadastrada.
- O acompanhamento possuir fatores associados a ele.

3.1.8 FATOR - LISTAGEM

Após clicar na opção de "Lista de fatores"na tela de listagem de acompanhamentos, o usuário é direcionado para a primeira tela de fatores que contém a lista de fatores (figura 27). Cada item na lista mostra o nome do fator.

FIGURA 27. LISTA DE FATORES CADASTRADOS.

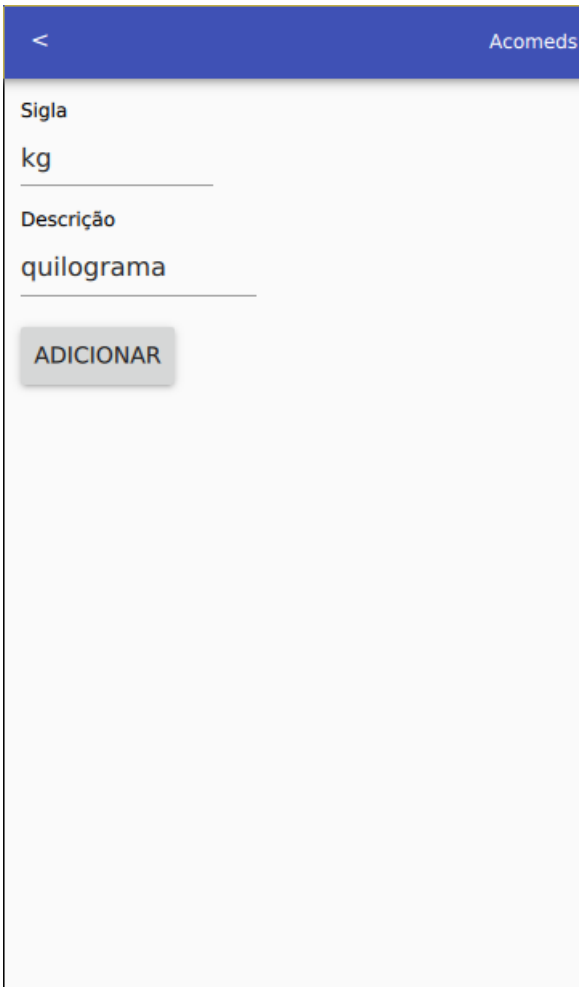


Fonte. O AUTOR (2017).

3.1.9 FATOR - CRIAÇÃO E EDIÇÃO

Ao clicar no botão "+" para criação de um novo fator ou ao clicar no botão "Editar" para edição de um fator já existente o aplicativo abre a tela de formulário de acompanhamentos representada pela figura 58.

FIGURA 28. CADASTRO E ALTERAÇÃO DE FATOR.

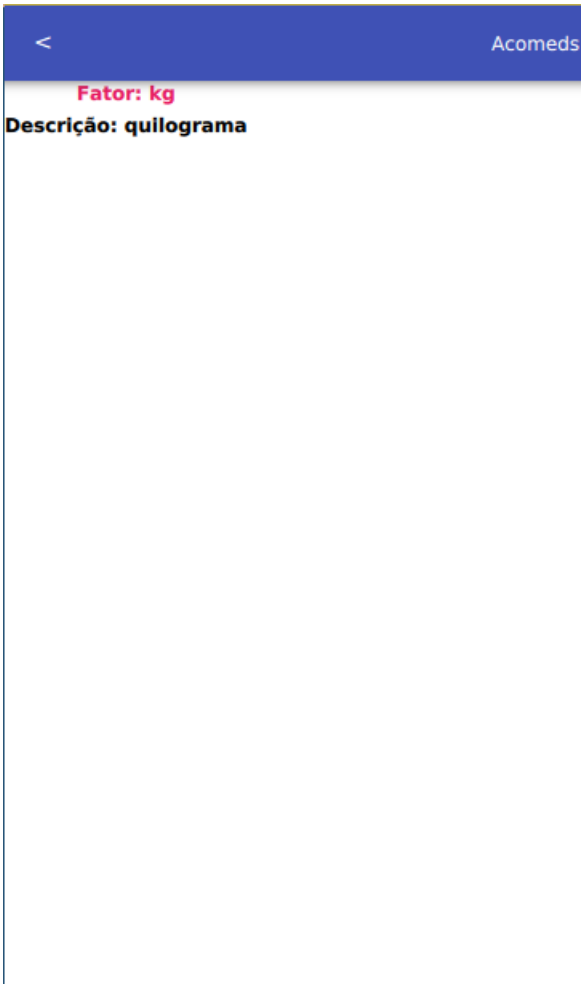


Fonte. O AUTOR (2017).

3.1.10 FATOR - VISUALIZAÇÃO

Ao clicar em algum medicamento da lista é possível visualizar seus detalhes, como na figura 29.

FIGURA 29. VISUALIZAÇÃO DE FATOR.

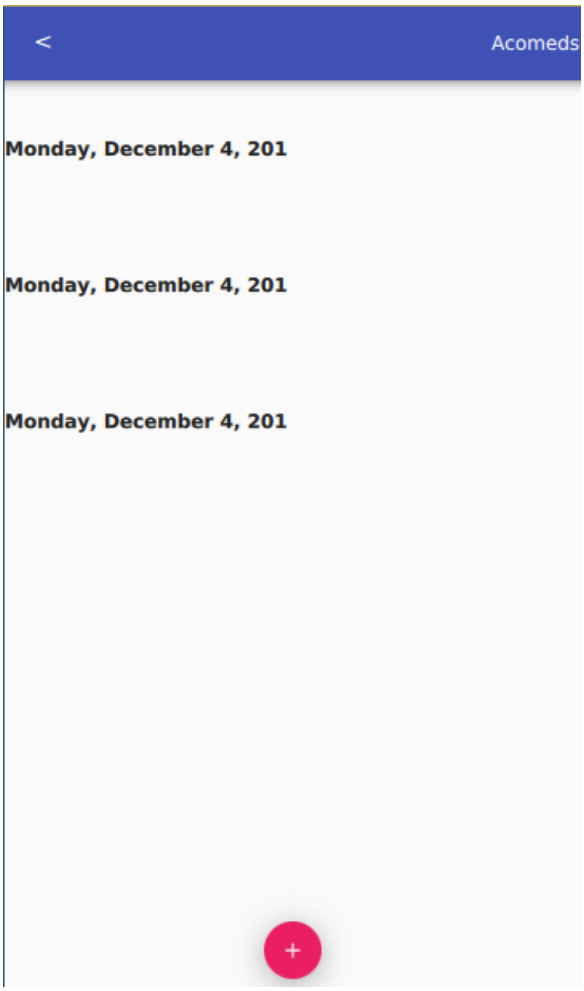


Fonte. O AUTOR (2017).

3.1.11 REGISTRO DE ACOMPANHAMENTO - LISTAGEM

Após clicar na opção de "Lista de registros"na tela de visualização de acompanhamentos, o usuário é direcionado para a primeira tela de registros, que contém a lista de registros (figura 30). Cada item na lista mostra a data do registro (o formato da data irá se adequar à língua do sistema).

FIGURA 30. LISTA DE REGISTROS CADASTRADOS.

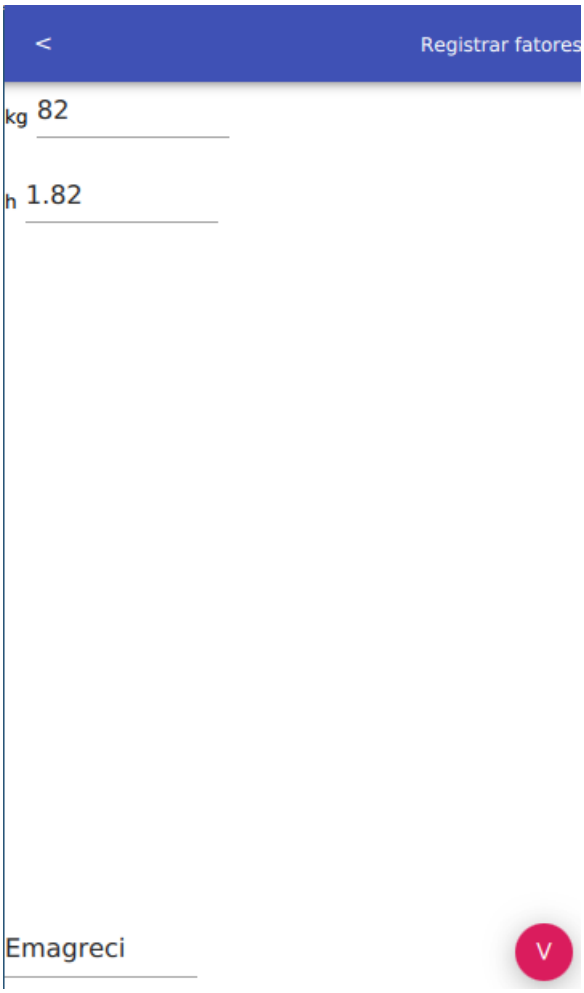


Fonte. O AUTOR (2017).

3.1.12 REGISTRO DE ACOMPANHAMENTO - CRIAÇÃO E EDIÇÃO

Ao clicar no botão “registrar” no menu de contexto de um acompanhamento da lista de acompanhamentos ou ao clicar no botão “Editar” para edição de um registro já existente o aplicativo abre a tela de formulário de acompanhamentos representada pela figura 31.

FIGURA 31. CADASTRO E ALTERAÇÃO DE REGISTRO.

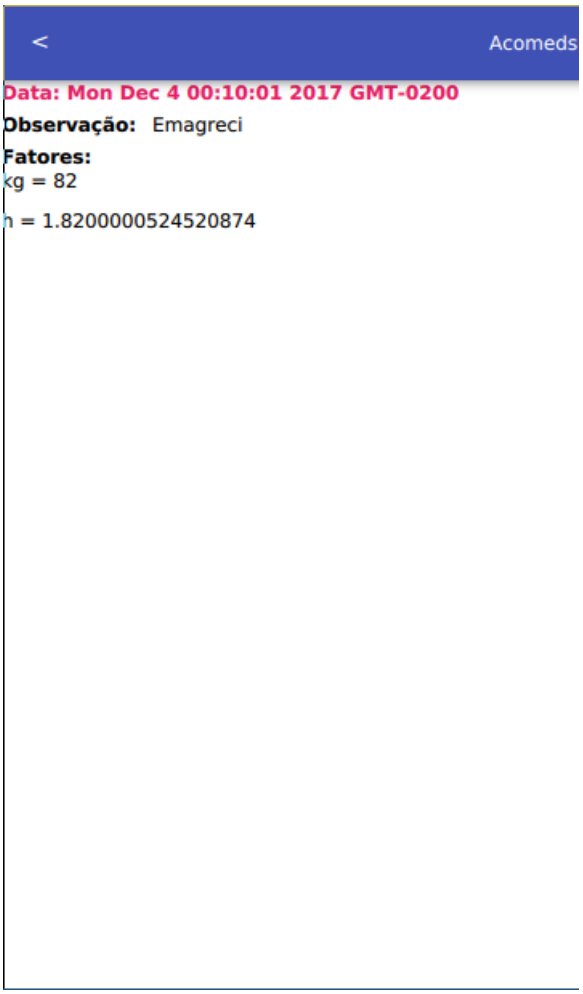


Fonte. O AUTOR (2017).

3.1.13 REGISTRO - VISUALIZAÇÃO

Ao clicar em algum registro da lista é possível visualizar a observação incluída no momento de seu cadastro, como na figura 32.

FIGURA 32. VISUALIZAÇÃO DE REGISTRO.



Fonte. O AUTOR (2017).

3.1.14 FUNCIONALIDADES DO SISTEMA

Neste capítulo serão apresentados fluxos exemplificando o uso prático do aplicativo para cada um dos módulos.

3.1.14.1 NOTIFICAR MEDICAMENTO

Para fazer com que o aplicativo exiba uma notificação de um medicamento, os passos listados podem ser utilizados como exemplo :

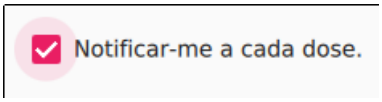
1. Na tela principal (figura 13) selecionar o módulo de medicamentos.

2. Na tela de lista de medicamentos (figura 14) selecionar o botão "+"para adicionar um medicamento.

3. Na tela de cadastro de medicamentos (figura 16), informar os dados desejados e deixar a opção para notificar medicamentos marcada. O botão "Adicionar"deve ser pressionado.

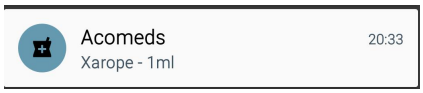
4. A partir desse momento, enquanto o aplicativo estiver em execução, uma notificação será mostrada de acordo com a posologia informada. Exemplos de notificações podem ser visualizados nas figuras 34 e 35;

FIGURA 33. OPÇÃO DE NOTIFICAÇÃO ATIVADA NO CADASTRO DE MEDICAMENTOS.



Fonte. O AUTOR (2017).

FIGURA 34. EXEMPLO DE NOTIFICAÇÃO NO SISTEMA OPERACIONAL ANDROID.



Fonte. O AUTOR (2017).

FIGURA 35. EXEMPLO DE NOTIFICAÇÃO NO SISTEMA OPERACIONAL GNU/LINUX PARA COMPUTADORES DE MESA (INTERFACE GRÁFICA I3WM).

Fonte. O AUTOR (2017).

3.1.14.2 ACOMPANHAR INDICADOR DE SAÚDE

Para acompanhar a progressão de um indicador de saúde visualizando um gráfico, os passos listados podem ser utilizados como exemplo :

1. Na tela principal (figura 13) selecionar o módulo de acompanhamentos.
2. Na tela de lista de acompanhamentos (figura 21) selecionar o botão "+"para adicionar um acompanhamento.
3. Se os fatores desejados não estiverem cadastrados, cadastre um por um apertando o botão "+"disponível na seleção de fatores (figura 24).
    3.1. Para cadastrar o fator, informe os dados na tela que aparecerá (figura 58). A sigla do fator é o valor que pdoerá ser utilizado na fórmula.
4. Selecione os fatores desejados e informe os dados do acompanhamento.
5. No campo fórmula utilize somente operadores matemáticos, números e siglas de fatores cadastrados.
6. Conclua o cadastro do acompanhamento.
7. Após ser direcionado novamente para a lista de acompanhamentos (figura 21), execute um clique longo sobre o acompanhamento que foi criado, até que apareça um menu de contexto (figura 22).
8. Selecione a opção registrar.
9. Informe os valores de cada fator na tela de registro(figura 31), opcionalmente adicione uma observação e então conclua o registro.
10. Execute pelo menos mais um registro para o mesmo acompanhamento.
11. Selecione o acompanhamento com um clique simples, possibilitando visualizar suas informações.
12. Um gráfico será mostrado para o acompanhamento, juntamente com seus dados (figura refapp:acgraph1).

3.2 USO DAS FERRAMENTAS

Esta etapa visa explicar mais aprofundadamente a arquitetura do sistema Acomeds desenvolvido no formato de aplicativo e seu fluxo de funcionamento.

3.2.1 ARQUITETURA DO SISTEMA

O aplicativo está desenvolvido usando o paradigma de orientação a objetos na camada de C++ e o paradigma orientado a eventos na camada de QML. O padrão MVC é utilizado na divisão das camadas do software, sendo que a implementação em C++ é responsável pelas camadas *Model* e *Controller* enquanto que a parte em QML é responsável pela camada *View*.

As três camadas do MVC foram bem definidas, sendo que a camada de modelo e controle estão implementadas usando C++, e a camada de visualização está implementada usando QML. As comunicações de dados e requisições foram feitas através da camada de controle, deste modo, a camada de visualização não se comunica diretamente com a camada de modelo.

O armazenamento de dados no sistema é realizado através da formatação JSON. Para manipulação em C++ do formato JSON utiliza-se o módulo *Property tree* (ptree) do conjunto de bibliotecas Boost para a linguagem. Esse módulo também permite a manipulação de dados usando o formato *eXtensible Markup Language* (XML).

3.2.2 ESPECIFICAÇÕES DE TELA

Como trata-se de um aplicativo multi-plataforma, foram utilizados recursos da linguagem QML que buscavam se adequar ao tamanho da tela ou janela em que o aplicativo estivesse sendo renderizado, porém, o tamanho mínimo recomendado para a utilização é de 360 x 615 pixels disponibilizados para uso exclusivo do aplicativo, sendo respectivamente os pixels correspondentes à largura x altura para um dispositivo em que a razão entre pixels físicos e pixels disponibilizados pelo sistema do dispositivo seja 1.

Essa resolução é utilizada porque representa aproximadamente uma resolução baixa para um dispositivo *mobile* (*smartphone*) e, portanto, como a tendência dos dispositivos é de aumentar cada vez mais a resolução da tela, essa é uma boa resolução para garantir seu funcionamento.

Alterações em proporções e tamanhos da tela ou janela não afetariam seu funcionamento tampouco dificultariam a visualização de seus componentes.

Para dispositivos em que a densidade de pixels fosse maior do que a densidade padrão disponível e que a razão entre os pixels físicos e os pixels disponibilizados fosse maior que 1 o aplicativo também se adaptaria através de funcionalidades nativas da linguagem QML.

3.2.3 ESPECIFICAÇÕES DE DESIGN

O design utilizado no aplicativo é baseado em primeiro lugar no *guideline* disponibilizado pelo Google[2] para sua especificação de design de software chamado Material Design.

Um *guideline* de design para desenvolvimento de aplicativos é um guia desenvolvido por alguém ou alguma instituição que tem como objetivo especificar formas de design, aparência, e comportamentos esperados para interações com o usuário para o aplicativo.

Um guia de design é útil para padronizar ações e componentes em um sistema e dessa maneira acostumar o usuário com os tipos de ações disponíveis para uma funcionalidade a partir das características de seus componentes. Os principais guias disponíveis na época do desenvolvimento do aplicativo eram o Material Design da Google, o Universal da Microsoft e o Human Interface da Apple.

O motivo de ter sido escolhido o Material Design como guia principal é o grande número de usuários do sistema Android quando comparado com as outras plataformas móveis disponíveis[3], sendo assim, a maior parte das pessoas já saberiam utilizar o sistema de forma intuitiva sem que fosse necessário introduções à sua maneira de funcionamento.

Inicialmente cada componente utilizado no aplicativo foi personalizado a mão para que correspondesse ao guia da Google. Porém no decorrer do projeto a empresa responsável pelo *framework* utilizado disponibilizou nativamente opções de aparência do software que usavam os guias Material Design e Universal.

A partir do momento em que as novas opções demonstraram ser estáveis pela comunidade o as telas do projeto Acomeds foram gradualmente migradas dos componentes personalizados para as implementações nativas e o motivo dessa migração é a quantidade de componentes já prontos para serem utilizados e também a disponibilidade de uso de implementações nativas que possivelmente já poderiam existir na plataforma sendo utilizada.

Vale ressaltar que no início do desenvolvimento do projeto não era certo a data de lançamento dessa nova feature na ferramenta que ainda teria de ser disponibilizada em uma versão estável após correções de possíveis problemas encontrados na sua utilização. Então optou-se por garantir que a maior parte do guia estivesse implementado.

[2] Possível encontrar em https://material.io/guidelines/ (acessado em 30 de novembro de 2017)

[3] http://www.idc.com/promo/smartphone-market-share/os informa entre 80% e 90% de *market share* na data de 29 de novembro de 2017

3.2.4 MANIPULAÇÃO DOS DADOS NO APLICATIVO

Ao iniciar a aplicação o JSON é carregado como objetos na memória principal do sistema como forma de otimizar as operações e a cada operação de inserção, edição e remoção o JSON também é atualizado para evitar que o aplicativo seja finalizado pelo usuário sem que as informações estejam salvas.

3.2.5 MÉTODO ÁGIL

O método ágil ajudou muito no desenvolvimento do aplicativo. O aplicativo teve de ser alterado várias vezes durante o projeto devido a imprevistos ocorridos durante o desenvolvimento.

Esse método também se mostra útil para a alteração de escopo do projeto, uma vez que esse foi primeiramente aumentado e posteriormente diminuído.

Com essa versatilidade demonstrada pelo método, é possível afirmar que a implementação sofreu um impacto pequeno dado o número de alterações ocorrido. A iteração que visava entregas de funcionalidades ajudou a manter o projeto com poucas partes inacabadas durante o desenvolvimento.

3.2.6 ARMAZENAMENTO DE DADOS

O modelo utilizado e a forma de armazenamento se mostraram suficientes para o escopo atual do projeto.

A forma de armazenamento (JSON) possibilita também o armazenamento futuro em um servidor de modo a deixá-lo mais simples, uma vez que poderá somente armazenar o arquivo com as informações e associá-lo a algum usuário.

4 FERRAMENTA UTILIZADA

A principal ferramenta utilizada no desenvolvimento do aplicativo Acomeds é o *framework* Qt.

O *framework* Qt se mostra eficaz e prático, porém, com algumas ressalvas :

1. A funcionalidade de notificação para *desktop* está precária e depende de uma biblioteca externa para executar uma notificação sem o uso de um ícone de *system tray* (bandeja de aplicações em segundo plano). Embora seja possível simular uma notificação em ambientes *desktop* através de uma janela, o resultado não seria incorporado ao aspecto do sistema operacional atual do usuário.

2. A funcionalidade de notificação para Android envolve um processo não facilitado e com implementação de um código em linguagem Java, com um respectifo arquivo xml de configuração. Isso se deve principalmente ao fato de que o sistema operacional Android em si só aceita o seu modo específico de exibir uma notificação, impossibilitando o uso de bibliotecas multi-plataformas com esse propósito. Entretanto, uma vez que o processo de implementação de uma notificação para esse sistema é sempre o mesmo, poderia haver alguma funcionalidade que facilitasse o seu uso.

É importante destacar que não houveram impedimentos na integração de uma biblioteca difernete para solucionar algo que falta com esse framework, como no caso de notificações *desktop*.

4.1 COMENTÁRIOS DE USUÁRIOS

Assim que o aplicativo foi mostrado para outras pessoas, foi possível notar que as funcionalidades em conjunto presentes no aplicativo está em falta no mercado. Cinco entre seis pessoas afirmaram não conhecer ou usar nenhum aplicativo semelhante.

Ao decorrer da utilização foi notado a carência de interações *online*, uma vez que quatro entre seis usuários questionaram por interações com redes sociais.

É importante citar que cinco entre seis usuários citaram que frequentemente se esquecem de tomar medicações e que o aplicativo poderia auxiliar.

Mas o resultado mais interessante ao final da pesquisa é notar que todos os usuários alegaram que o aplicativo seria muito bom para familiares idosos, que precisam se medicar muitas vezes durante um curto período de tempo e que além disso, ainda auxiliaria no acompanhamento de índices de saúde como diabetes e doenças crônicas.

É possível afirmar então, a partir da pesquisa, que o aplicativo pode ser uma ótima opção para idosos e uma opção considerável para as demais faixas etárias.

4.2 OPERABILIDADE DO APLICATIVO

Neste capítulo é apresentado

4.2.1 DISPOSITIVOS *DESKTOP*

Para dispositivos *desktop* é imperceptível o tempo de execução entre uma operação e outra. Mesmo quando testado em um sistema relativamente lento - para os padrões da época em que o aplicativo foi testado[1] - e com uma versão de compilação desvantajosa (MinGW 32bits), o aplicativo executa suas funções de maneira satisfatória (rápida) e sem afetar o desempenho do sistema operacional.

O melhor cenário *desktop* para execução do aplicativo ocorre usando o sistema operacional GNU/Linux com arquitetura amd64, onde é possível notar completa fluidez do sistema[2].

4.2.2 DISPOSITIVOS *MOBILE*

Os dispositivos mobile utilizados nos testes[3] executaram de maneira satisfatória as operações do aplicativo.

As operações executaram de acordo com o poder de processamento do dispositivo, sendo que, caso alguma operação estivesse demorando poucos segundos para responder, um indicativo de carregamento era apresentado ao usuário.

Eventuais diminuições na velocidade de execução ocorreram somente para dispositivos considerados lentos para a época [4].

Para dispositivos que não são considerados lentos, o uso do aplicativo funciona de maneira igual ou próxima ao uso no *desktop*. Entretanto, vale ressaltar que: uma vez que dispositivos móveis visam o baixo consumo de energia - desativando núcleos de processamento e diminuindo drásticamente a frequência de operação dos mesmos - o

[1] Atlhlon X2 250 com 4GiB de memória RAM DDR2 667MHz e placa mãe com frequência de barramento abaixo do recomendado (de 1000MHz, o processador se beneficiaria de até 2000MHz) rodando o sistema operacional Windows 10.

[2] É importante observar aqui que o uso de dispositivo de captura de imagem (por exemplo : câmera para registrar a foto de uma receita) dependerá da requisição e ativação do mesmo, podendo levar uma quantidade notável de tempo (poucos segundos)

[3] Exemplo : Samsung Galaxy S5, Samsung Galaxy On7, Positivo S480.

[4] Menos de 4 núcleos e/ou frequência de *clock* de CPU inferior a 1,5GHz e/ou IPC(SP) de CPU inferior a 8.

sistema operacional como um todo estará sujeito a oscilações de desempenho até que os ajustes sejam finalizados. Essas oscilações afetarão todos os aplicativos em execução no sistema.

5 CONCLUSÃO

Quanto aos objetivos específicos :

- No módulo de medicamentos é possível registrar medicamentos com as informações pertinentes -incluindo uma foto de receita ou exame- e opcionalmente ser notificado dos mesmos.
- No módulo de acompanhamentos é possível realizar o registro de indicadores cadastrados para acompanhamento de progressão através de gráficos.
- É possível executar o aplicativo nas diversas plataformas disponibilizadas pelo Qt Quick, sendo que foram realizados testes nos sistemas : Android, GNU/Linux e Windows.
- O aplicativo é apresentado baseando-se nas diretrizes Material e Universal, proporcionando desta maneira uma interface amigável para o usuário.

Após a sumarização dos objetivos específicos e os resultados obtidos é possível afirmar que o objetivo geral foi atingido.

Em geral é possível observar que o framework entrega um aplicativo fluído e eficaz, desde que o código QML não seja sobrecarregado e a camada de código C++ seja utilizada para aproveitar a compilação de código, uma vez que o QML por padrão é interpretado no dispositivo. Até o momento da finalização deste documento, o pré-compilador de javascript e QML para o binário final [1] ainda não estava disponível para uso com a licença gratuita e, portanto, ainda é possível melhorar ainda mais o desempenho.

5.1 SUGESTÕES

Após a visualização do resultado final, nota-se que alguns aspectos do sistema poderiam ser melhorados :

- Organizar melhor os elementos na tela, a partir de estudos mais profundos de UX (Experiência do usuário, do inglês, *User Experience*).
- Executar a compilação e *deploy* para plataforma Android utilizando a ferramenta CMake com o objetivo de facilitar a inclusão das bibliotecas.

[1] Compilador de javascript que promete melhorar significativamente a velocidade final da aplicação. Disponível em http://doc.qt.io/QtQuickCompiler/ , acessado dia 30 de novembro de 2017

6 TRABALHOS FUTUROS

Alguns desejos iniciais não puderam ser disponibilizados na finalização do projeto, principalmente por falta de tempo, que estão listados aqui como sugestões para continuação do projeto :

- Adicionar uma forma de *backup* em nuvem.
- Adicionar um módulo adicional para que um usuário possa observar e interagir com outro usuário.
- Utilizar bancos de dados (relacionais ou não) carregados na memória principal do sistema, sincronizando os dados persistidos. Entretanto, seria necessário fazer uma análise para verificar se o aplicativo não iria se tornar mais pesado[1] para o usuário final.
- Continuar o processo de internacionalziação do aplicativo.

[1] Termo utilizado para denotar um aplicativo que, por demorar ao executar suas rotinas, pode afetar negativamente o desempenho do sistema operacional e/ou de si mesmo.

REFERÊNCIAS

AGILEMANIFESTO. *Principles behind the Agile Manifesto*. 2001. Disponível em: <http://agilemanifesto.org/principles.html>. Citado na página 19.

BOOCH, G. *Object Oriented Design*. 1991. Citado na página 24.

CHEN, P. P. S. *Entity-Relationship Approach to Systems Analysis and Design*: Proceedings of the international conference on entity-relationship approach to systems analysis and design. [S.l.]: North-Holland Publishing Company, 1979. Citado na página 27.

CMAKE. *CMake - Página principal*. 2017. Disponível em: <https://cmake.org/>. Citado na página 42.

ESTADãO. *Celular é ferramenta de saúde e bem-estar*. 2013. Disponível em: <http://link.estadao.com.br/noticias/geral,celular-e-ferramenta-de-saude-e-bem-estar,10000032831>. Citado na página 14.

EXAME. *Estatísticas de uso de celular no Brasil*. 2016. Disponível em: <https://exame.abril.com.br/negocios/dino/estatisticas-de-uso-de-celular-no-brasil-dino89091436131/>. Citado na página 14.

GIT. *Git - Página principal*. 2017. Disponível em: <https://git-scm.com/>. Citado na página 43.

GITLAB. *Gitlab - Página principal*. 2017. Disponível em: <https://gitlab.com>. Citado na página 43.

GLOBO, O. *Avanços da medicina aumentam expectativa de vida para quem está na faixa dos 40 anos*. 2011. Disponível em: <https://oglobo.globo.com/sociedade/ciencia/avancos-da-medicina-aumentam-expectativa-de-vida-para-quem-esta-na-faixa-dos-40-anos-2720967>. Citado na página 14.

KDE. *SnoreNotify - Página principal*. 2017. Disponível em: <https://github.com/KDE/snorenotify>. Citado na página 39.

MACHADO, H. *POO : Os 4 pilares da Programação Orientada a Objetos*. 2018. Disponível em: <https://www.devmedia.com.br/os-4-pilares-da-programacao-orientada-a-objetos/9264>. Citado na página 23.

NAGAO, F. *Linguagens : abaseadas em objetos e orientadas a objetos - iMasters Blog (artigo publicado originalmente no iMasters)*. 2018. Disponível em: <https://www.ibm.com/developerworks/community/blogs/fd26864d-cb41-49cf-b719-d89c6b072893/entry/linguagens_baseada_em_objetos_e_orientada_a_objetos2?lang=en>. Citado na página 24.

NICCOLI, T.; PARTRIDGE, L. *Ageing as a Risk Factor for Disease*. 2012. [Online; publicado em 2012-09-11]. Disponível em: <http://www.cell.com/current-biology/fulltext/S0960-9822(12)00815-9>. Citado na página 16.

OPENSTD. *Library Technical Report*. 2003. Disponível em: <http://www.open-std.org/jtc1/sc22/wg21/docs/library_technical_report.html>. Citado na página 39.

QT. *Qt Cross-platform software development for embedded and desktop*. 2017. Disponível em: <https://www.qt.io/>. Citado na página 35.

REHENS, S. Citado na página 14.

RUMBAUGH, J. et al. *Object-Oriented Modeling and Design*. 1991. Citado na página 24.

RUSLO. *Hunter - Página principal*. 2017. Disponível em: <https://github.com/ruslo/hunter>. Citado na página 43.

SCHEMA, J. *JSON Schema - Página principal*. 2017. Disponível em: <http://json-schema.org/>. Citado na página 40.

TRELLO. *Trello - Página principal*. 2017. Disponível em: <https://trello.com/>. Citado na página 21.

UML. *Welcome To UML Web Site!* 2017. Disponível em: <http://www.uml.org/>. Citado na página 28.

W3TECHS. *Usage of client-side programming languages for websites*. 2017. Disponível em: <https://w3techs.com/technologies/overview/client_side_language/all>. Citado na página 37.

W3TECHS. *Usage of site elements for websites*. 2017. Disponível em: <https://w3techs.com/technologies/overview/site_element/all>. Citado na página 37.

GLOSSÁRIO

ISO abrev. de International Organization for Standardization (Organização Internacional de Normalização)..

alert Palavra em inglês que significa alerta, alarme..

Alzheimer Também conhecida por esclerose e caduquice. Doença ou Mal de Alzheimer é uma enfermidade incurável, de caráter neurodegenerativo; agravando-se ao longo do tempo, contudo há tratamento. A maioria de seus pacientes são pessoas idosas..

amount Palavra em inglês que significa soma, quantia, total. ..

begin Palavra em inglês que significa começar, iniciar, fazer a primeira parte..

continuous Palavra em inglês que significa contínuo, ininterrupto, constante, incessante ..

date Palavra em inglês que significa data, período, prazo, duração ..

description Palavra em inglês que significa enunciado de um problema com todos os detalhes que possibilitem armar sua lógica para programar e executar sua solução..

desktop Palavra em inglês utilizada para denominar computadores pessoais..

end Palavra em inglês que significa fim, acabar, concluir, terminar ..

factor Palavra em inglês que significa fator, elemento, momento, circunstância que concorre para um resultado ..

feedback Palavra em inglês utilizada para denominar o procedimento que consiste no provimento de informação a uma pessoa sobre o desempenho, conduta, ou ação executada por esta, objetivando reorientar ou estimular comportamentos futuros mais adequados..

formula Palavra em inglês que significa preceito, doutrina, norma, regra fixa, receita, receita que indica o quantitativo das substâncias que compõem um remédio ..

framework Palavra em inglês utilizada para designar enquadramento, estrutura. Processo que consiste em selecionar de uma corrente de fluxo contínuo de bits agrupamentos de bits que representam um ou mais caracteres. Conjunto de elementos e suas interligações constituindo a base de um sistema ou projeto. Software..

goal Palavra em inglês que significa meta, baliza, marco..

health Palavra em inglês que significa saúde ..

interval Palavra em inglês que significa intervalo, distancia, pausa ..

length Palavra em inglês que significa comprimento. É um termo que define a dimensão de um objeto..

marker Palavra em inglês que significa marcador, anotador, objeto que serve como marca, sinal..

maximum Palavra em inglês que significa máximo, maior que todos..

medication Palavra em inglês que significa medicação..

minimum Palavra em inglês que significa o menor grau ..

mobile Palavra em inglês utilizada para denominar telefone celular..

name Palavra em inglês que significa nome, título, epíteto..

notebook Palavra em inglês utilizada para denominar computadores pessoais portáteis..

posology Palavra em inglês que significa posologia..

properties Palavra em inglês que significa propriedade..

record Palavra em inglês que significa registro, inscrição, anotação..

release Palavra em inglês utilizada para denominar liberação pública de uma nova versão de um programa ao qual foram adicionadas correções e melhorias..

required Palavra em inglês que significa preciso, necessário, exigido..

smartphone Um smartphone é um telefone celular que combina recursos de computadores pessoais, com funcionalidades avançadas que podem ser estendidas por meio de programas aplicativos executados pelo seu sistema operacional..

tablet Palavra em inglês utilizada para denominar um dispositivo pessoal em formato de prancheta que pode ser usado para acesso à Internet, organização pessoal, visualização de fotos, vídeos, leitura de livros, jornais e revistas e para entretenimento com jogos..

title Palavra em inglês que significa Em comunicação de dados e tratando-se de interconexão de sistemas abertos, o título é um identificador permanente de uma entidade..

unit Palavra em inglês que significa unidade, 1 quantidade, de um todo, 2 pessoa ou coisa isoladamente, 3 grupo, 4 o menor número inteiro..

website Palavra em inglês utilizada para designar um conjunto de páginas web, isto é, de hipertextos acessíveis geralmente pelo protocolo HTTP na internet..

[title=GLOSSÁRIO]

APÊNDICE A – TABELA DE TECNOLOGIAS E SUAS RESPECTIVAS VERSÕES

Esta tabela (2) têm como objetivo disponibilizar as versões utilizadas das tecnologias presentes no aplicativo.

TABELA 2. TABELA DE VERSÕES DAS TECNOLOGIAS.

Tecnologia	**Versão**
GCC	7.2.0
MinGW-GCC	4.8
Boost	1.63
ExprTk	*commit* de 27 de novembro de 2017
MuParser	2
Qt	5.9.3
qmake	3.1
CMake	3.9
SnoreNotify	0.7
QtCreator	4.4.1
HunterGate	v0.19.87

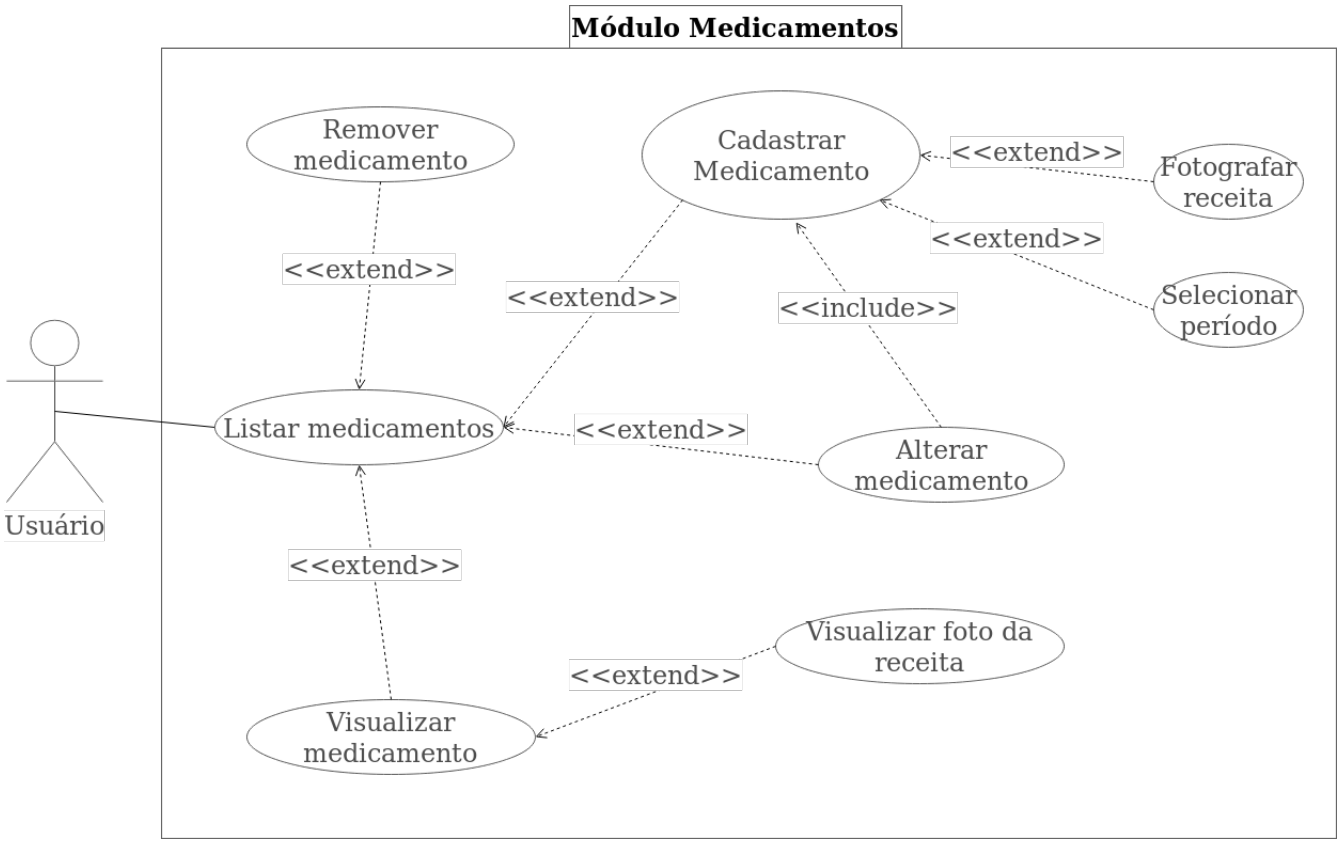
Fonte. O AUTOR (2017).

APÊNDICE B – CASOS DE USO

B.1 MEDICAMENTO

A figura 36 representa o caso de uso do módulo de medicamentos. Identificação

FIGURA 36. CASO DE USO MÓDULO MEDICAMENTO.



Fonte. O AUTOR (2017).

dos atores : Ator 01 - Usuário : A pessoa que está usando o app.

B.1.1 UC-01 LISTAR MEDICAMENTOS

Este caso de uso mostra na tela uma lista interativa informando o usuário sobre os medicamentos anteriormente cadastrados no aplicativo. Permitindo o completo gerenciamento dos medicamentos (*Create, Read, Update and Delete* (CRUD)).

(a) Atores: Usuário

(b) Pré-Condições: Nenhuma

(c) Pós-Condições: Nenhuma

(d) Requisitos Funcionais: A listagem deve onter informações básicas sobre os medicamentos, entre nome e a notificação.

(e) Requisitos não funcionais: Nenhum

Fluxo básico

1. O caso de uso começa quando o usuário seleciona o módulo de medicamentos.
2. O sistema mostra uma tela contendo a lista de medicamentos cadastrados.
3. Para cada medicamento háverá as informações: nome, posologia e se o usuário deve ser notificado.

Fluxo secundário

1. Caso não haja medicamentos cadastrados nenhum medicamento será mostrado na lista e o caso de uso termina.

B.1.2 UC-03 CADASTRAR MEDICAMENTO

Aqui o usuário pode inserir na aplicação novos medicamentos, configurando suas informações e notificações

(a) Atores: Usuário

(b) Pré-Condições: Nenhuma

(c) Pós-Condições: Caso selecionado, o aplicativo deve agendar as notificações

(d) Requisitos Funcionais: Cadastrar um medicamento no sistema

(e) Requisitos não funcionais: Caso selecionado, agendar as notificações para lembrar o usuário sobre os horários de tomar o remédio

Fluxo básico

1. O caso de uso começa quando o usuário seleciona o módulo de medicamentos.
2. {Mostrar lista de medicamentos}
3. Ao pressionar o botão "+"o usuário é levado para a tela de cadastro de novo medicamento.

4. O usuário insere as informações do medicamento.
5. O usuário {seleciona o intervalo do medicamento}
6. O usuário {tira uma foto da receita}
7. O usuário pressiona o botão "Adicionar"
8. Executar subfluxo validar informações de medicamento.
9. O medicamento é adicionado.
10. {Mostrar lista de medicamentos}

Fluxo secundário

1. A qualquer momento após o passo 3 do fluxo básico e antes de pressionar o botão "Adicionar"o usuário pode apertar o botão para voltar para a tela anterior. O medicamento não é cadastrado e o caso de uso termina.
2. Em {tira uma foto da receita} caso o usuário não deseje fotografar uma receita a foto não será cadastrada e o caso de uso continua.

A1 - Mostrar lista de medicamentos

1. Em {Mostrar lista de medicamentos} quando não há medicamento cadastrado.
2. Nenhum medicamento é exibido.
3. Portanto as opções de edição e remoção não podem ser acionadas.

A2 - Eleger intervalo customizado

1. Em {seleciona o intervalo do medicamento} caso o usuário opte por informar manualmente a posologia.
2. È possível selecionar a data e hora através de um calendário selecionável.

A3 - Fotografar receita para consulta futura

1. Em {tira uma foto da receita} caso o usuário pressione o botão para tirar uma foto.
2. O sistema mostra uma tela onde é possível registrar uma fotografia para anexar ao medicamento.

S1 - validar informações de medicamento.

1. Sistema verifica se campo Nome está preenchido.
2. Sistema sinaliza se a informação não foi validada.
3. Sistema verifica se campo Quantidade está preenchido.
4. Sistema sinaliza se a informação não foi validada.
5. Sistema verifica se campo Unidade está preenchido.
6. Sistema sinaliza se a informação não foi validada.

B.1.3 UC-04 ALTERAR MEDICAMENTO

Aqui o usuário pode inserir na aplicação novos medicamentos, configurando suas informações e notificações

(a) Atores: Usuário

(b) Pré-Condições: Algum medicamento precisa estar previamente cadastrado

(c) Pós-Condições: Caso selecionado, o aplicativo deve agendar as notificações

(d) Requisitos Funcionais: Editar um medicamento no sistema

(e) Requisitos não funcionais: Caso selecionado, agendar as notificações para lembrar o usuário sobre os horários de tomar o remédio

1. O caso de uso começa quando o usuário seleciona o módulo de medicamentos.
2. A lista de medicamentos é mostrada.
3. Ao executar um clique longo sobre um medicamento da lista um menu de contexto referente à esse medicamento é aberto.
4. O usuário seleciona a opção "Editar"
5. A tela de cadastro de medicamentos é mostrada com os dados do formulário preenchidos com as informações do medicamento sendo referenciado.
6. O usuário pressiona o botão "Editar"
7. Executar subfluxo validar informações de medicamento
8. O medicamento é atualizado e salvo.
9. Executar o subfluxo mostrar lista de medicamentos.

Fluxo secundário

1. No passo 5 o usuário pode apertar o botão para voltar para a tela anterior. O medicamento não é atualizado e o caso de uso termina.
2. No passo 2 caso não haja nenhum medicamento cadastrado, o sistema não mostra nenhum item na lista e portanto não é possível acessar nenhum menu de contexto relacionado diretamente a um medicamento. O caso de uso termina.

S1 - validar informações de medicamento.

1. Sistema verifica se campo Nome está preenchido.
2. Sistema sinaliza se a informação não foi validada.
3. Sistema verifica se campo Quantidade está preenchido.
4. Sistema sinaliza se a informação não foi validada.
5. Sistema verifica se campo Unidade está preenchido.
6. Sistema sinaliza se a informação não foi validada.

B.1.4 UC-02 REMOVER MEDICAMENTO

Este caso de uso remove um medicamento do aplicativo, apagando seu registro, não notificando mais o ator.

(a) Atores: Usuário

(b) Pré-Condições: Algum medicamento precisa estar cadastrado no app

(c) Pós-Condições: O medicamento precisa ser totalmente apagado do sistema

(d) Requisitos Funcionais: Remover completamente o remédio dos jsons do sistema

(e) Requisitos não funcionais: Nenhum

1. O caso de uso começa quando o usuário seleciona o módulo de medicamentos.
2. A lista de medicamentos é mostrada.
3. Ao executar um clique longo sobre um medicamento da lista um menu de contexto referente à esse medicamento é aberto.
4. O usuário seleciona a opção "Remover".

5. Uma caixa de diálogo e apresentada para que o usuário confirme a remoção.
6. O medicamento é removido.

Fluxo secundário

1. No passo 5 caso o usuário opte por não remover o medicamento, esse não é removido e o caso de uso termina.
2. No passo 5 caso o usuário pressione o botão voltar, o medicamento não é removido e o caso de uso termina.
3. No passo 2 caso não haja nenhum medicamento cadastrado, o sistema não mostra nenhum item na lista e portanto não é possível acessar nenhum menu de contexto relacionado diretamente a um medicamento. O caso de uso termina.

B.1.5 UC-05 VISUALIZAR MEDICAMENTO

Uma simples tela mostrando todas as informações que o medicamento pode ter.

(a) Atores: Usuário

(b) Pré-Condições: Algum medicamento precisa estar previamente cadastrado

(c) Pós-Condições: Nenhuma

(d) Requisitos Funcionais: Mostrar todas as informações de um medicamento

(e) Requisitos não funcionais: Nenhum

1. O caso de uso começa quando o usuário seleciona o módulo de medicamentos.
2. A lista de medicamentos é mostrada.
3. O usuário seleciona um medicamento executando um clique padrão.
4. Uma tela de visualização de medicamento é mostrada contendo as informações do medicamento.
5. Caso haja uma fotografia disponível, ela é disponibilizada para visualização.

Fluxo secundário

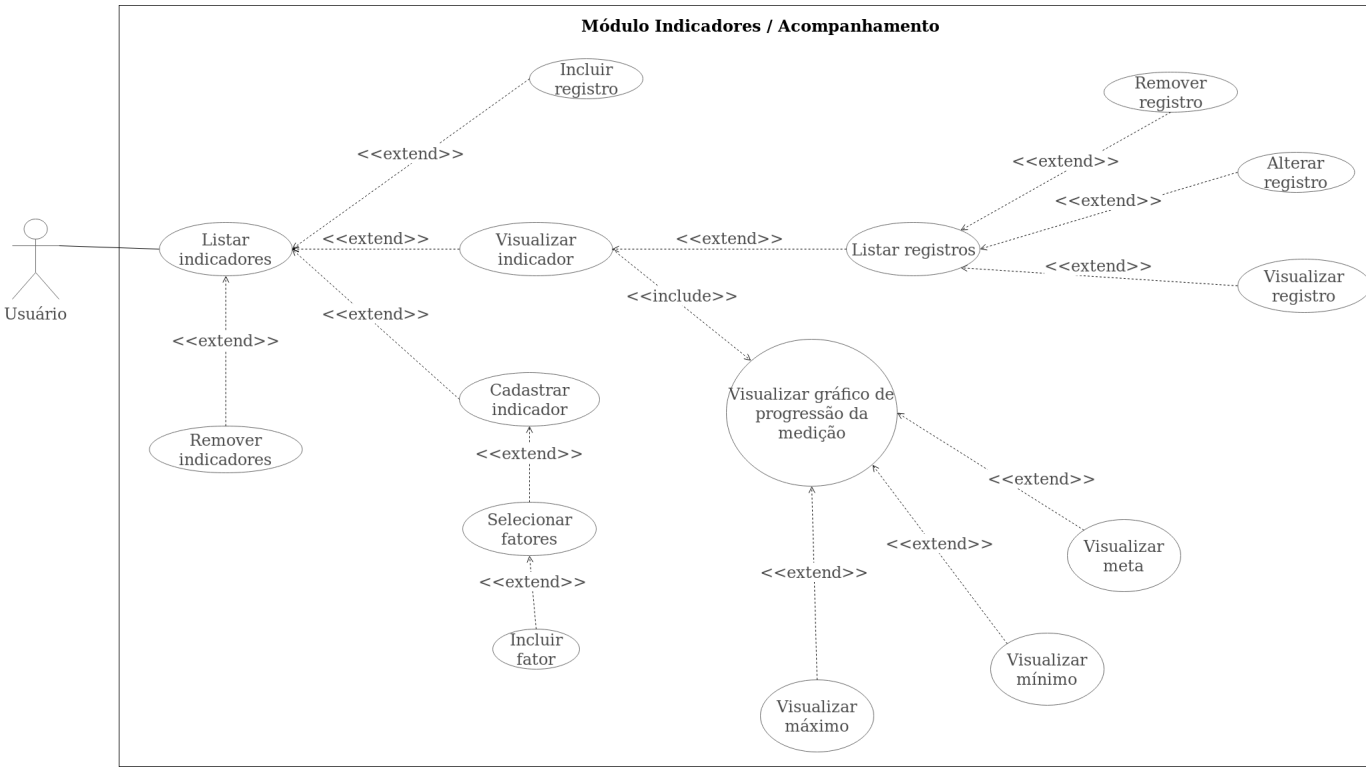
1. No passo 2 caso não haja nenhum medicamento cadastrado, o sistema não mostra nenhum item na lista e portanto não é possível acessar nenhum menu de contexto relacionado diretamente a um medicamento. O caso de uso termina.

2. A partir do passo 4, se o usuário pressionar a tecla voltar, a tela de visualização e fechada e o caso de uso termina.

B.2 ACOMPANHAMENTO

A figura 37 representa o caso de uso do módulo de acompanhamentos. Identificação

FIGURA 37. CASO DE USO MÓDULO ACOMPANHAMENTO.



Fonte. O AUTOR (2017).

dos atores : Ator 01 - Usuário : A pessoa que está usando o app.

B.2.1 UC-06 LISTAR ACOMPANHAMENTOS

Este caso de uso mostra na tela uma lista interativa informando o usuário sobre os acompanhamentos anteriormente cadastrados no aplicativo. Permitindo o completo gerenciamento dos acompanhamentos (CRUD).

(a) Atores: Usuário

(b) Pré-Condições: Nenhuma

(c) Pós-Condições: Nenhuma

(d) Requisitos Funcionais: A listagem deve onter informações básicas sobre os acompanhamentos, nome, quantidade de registros e objetivo.

(e) Requisitos não funcionais: Nenhum

Fluxo Básico:

1. O ator abre a lista
2. Os acompanhamentos são listados

B.2.2 UC-07 REMOVER ACOMPANHAMENTO

Este caso de uso remove um acompanhamento do aplicativo, apagando totalmente seu registro.

(a) Atores: Usuário

(b) Pré-Condições: Algum acompanhamento precisa estar cadastrado no app

(c) Pós-Condições: O acompanhamento precisa ser totalmente apagado do sistema

(d) Requisitos Funcionais: Remover completamente o acompanhamento dos jsons do sistema

(e) Requisitos não funcionais: Nenhum

Fluxo Básico:

1. O ator abre a lista
2. Os acompanhamentos são listados
3. O ator escolhe a ação "Remover"em um dos acompanhamentos
4. O acompanhamento é apagado do sistema

B.2.3 UC-08 CADASTRAR ACOMPANHAMENTO

Aqui o usuário pode inserir na aplicação novos acompanhamentos, configurando suas informações e fatores

(a) Atores: Usuário

(b) Pré-Condições: Nenhuma

(c) Pós-Condições: Nenhum

(d) Requisitos Funcionais: Cadastrar um acompanhamento no sistema

(e) Requisitos não funcionais: Nenhum

Fluxo Básico:

1. O ator abre a lista
2. Os acompanhamentos são listados
3. O ator escolhe a ação "Criar"
4. O formulário é mostrado na tela
5. O ator preenche o formulário
6. 1 Ao clicar em selecionar fatores, abre um tela de listagem de fatores
7. O ator clica em "salvar"e o acompanhamento é registrado
8. Volta para listagem

Fluxo Alternativo 1:

1. No passo 6, ao clicar em "salvar"e algum campo obrigatório não foi preenchido, é mostrada uma mensagem de erro
2. Não volta para tela de listagem, continua no formulário

B.2.4 UC-09 ALTERAR ACOMPANHAMENTO

Aqui o usuário pode inserir na aplicação novos acompanhamentos, configurando suas informações e fatores

(a) Atores: Usuário

(b) Pré-Condições: Algum acompanhamento precisa estar previamente cadastrado

(c) Pós-Condições: O acompanhamento deve ser modificado na base de dados

(d) Requisitos Funcionais: Editar um acompanhamento no sistema

(e) Requisitos não funcionais: Nenhum

Fluxo Básico:

1. O ator abre a lista
2. Os acompanhamentos são listados
3. O ator escolhe a ação "Editar"em algum acompanhamento
4. O formulário é mostrado na tela com os dados preenchidos
5. O ator preenche o formulário
6. 1 Ao clicar em selecionar fatores, abre um tela de listagem de fatores
7. O ator clica em "salvar"e o acompanhamento é registrado
8. Volta para listagem, não atualizada

Fluxo Alternativo 1:

1. No passo 6, ao clicar em "salvar"e algum campo obrigatório não estiver preenchido, é mostrada uma mensagem de erro
2. Não volta para tela de listagem, continua no formulário

B.2.5 UC-10 VISUALIZAR ACOMPANHAMENTO

Uma simples tela mostrando todas as informações que o acompanhamento pode ter.

(a) Atores: Usuário

(b) Pré-Condições: Algum acompanhamento precisa estar previamente cadastrado

(c) Pós-Condições: Nenhuma

(d) Requisitos Funcionais: Mostrar todas as informações de um acompanhamento

(e) Requisitos não funcionais: Nenhum

Fluxo Básico:

1. O ator abre a lista
2. Os acompanhamentos são listados
3. O ator clicla em algum acompanhamento
4. Uma tela com o gráfico dos fatores (segundo a fórmula) pelo tempo é mostrada

B.2.6 UC-11 REGISTRAR ACOMPANHAMENTO

Um formulário para registrar os níveis dos fatores de um acompanhamento naquele momento, alimentando um ponto do gráfico

(a) Atores: Usuário

(b) Pré-Condições: 1 - Algum acompanhamento precisa estar previamente cadastrado 2 - O acompanhamento precisa ter ao menos um fator selecionado

(c) Pós-Condições: O registro precisa ser gravado com a data atual na base de dados, para uso no gráfico da visualização do acompanhamento

(d) Requisitos Funcionais: Montar um formulário dinâmicamente, possuindo um campo numérico para cada fator

(e) Requisitos não funcionais: Nenhum

Fluxo Básico:

1. O ator abre a lista
2. Os acompanhamentos são listados
3. O ator escolhe a ação "Registrar"em algum acompanhamento
4. Um formulário é mostrado, cada fator registrado no acompanhamento terá um campo
5. O ator preenche os campos
6. O ator clica em "salvar"e o registro é gravado
7. Volta para listagem

B.2.7 LISTAGEM DE FATORES DE FATORES

B.2.8 UC-01 LISTAR FATORES

Este caso de uso mostra na tela uma lista interativa informando o usuário sobre os fatores anteriormente cadastrados no aplicativo. Permitindo o completo gerenciamento dos fatores (CRUD).

(a) Atores: Usuário

(b) Pré-Condições: Nenhuma

(c) Pós-Condições: Nenhuma

(d) Requisitos Funcionais: A listagem deve conter informações sobre os fatores, nome e descrição.

(e) Requisitos não funcionais: Nenhum

Fluxo Básico:

1. O ator abre a lista
2. Os fatores são listados

B.2.9 UC-02 REMOVER FATOR

Este caso de uso remove um fator do aplicativo, apagando seu registro.

(a) Atores: Usuário

(b) Pré-Condições: Algum fator precisa estar cadastrado no app

(c) Pós-Condições: O fator precisa ser totalmente apagado do sistema

(d) Requisitos Funcionais: Remover completamente o fatores dos jsons do sistema

(e) Requisitos não funcionais: Nenhum

Fluxo Básico:

1. O ator abre a lista
2. Os fatores são listados
3. O ator escolhe a ação "Remover"em um dos fatores
4. O fator é apagado do sistema

B.2.10 UC-03 CADASTRAR FATOR

Aqui o usuário pode inserir na aplicação novos fatores, configurando suas informações

(a) Atores: Usuário

(b) Pré-Condições: Nenhuma

(c) Pós-Condições: Nenhuma

(d) Requisitos Funcionais: Cadastrar um fator no sistema

(e) Requisitos não funcionais: Nenhum

Fluxo Básico:

1. O ator abre a lista
2. Os fatores são listados
3. O ator escolhe a ação "Criar"
4. O formulário é mostrado na tela
5. O ator preenche o formulário
6. O ator clica em "salvar"e o fator é registrado
7. Volta para listagem

Fluxo Alternativo 1:

1. No passo 6, ao clicar em "salvar"e algum campo obrigatório não foi preenchido, é mostrada uma mensagem de erro
2. Não volta para tela de listagem, continua no formulário

B.2.11 UC-04 ALTERAR FATOR

Aqui o usuário pode editar fatores previamente cadastrados, configurando suas informações

(a) Atores: Usuário

(b) Pré-Condições: Algum fator precisa estar previamente cadastrado

(c) Pós-Condições: Nenhuma

(d) Requisitos Funcionais: Editar um fator no sistema

(e) Requisitos não funcionais: Nenhum

Fluxo Básico:

1. O ator abre a lista
2. Os fatores são listados
3. O ator escolhe a ação "Editar"em algum fator
4. O formulário é mostrado na tela com os dados preenchidos
5. O ator preenche o formulário
6. O ator clica em "salvar"e o fator é registrado
7. Volta para listagem

Fluxo Alternativo 1:

1. No passo 6, ao clicar em "salvar"e algum campo obrigatório não estiver preenchido, é mostrada uma mensagem de erro
2. Não volta para tela de listagem, continua no formulário

1. 3. Caso de Uso: Gerenciamento de Registros

B.2.12 UC-01 LISTAR REGISTROS

Este caso de uso mostra na tela uma lista interativa informando o usuário sobre os registros anteriormente cadastrados no aplicativo. Permitindo o completo gerenciamento dos registros (CRUD).

(a) Atores: Usuário

(b) Pré-Condições: Nenhuma

(c) Pós-Condições: Nenhuma

(d) Requisitos Funcionais: A listagem deve conter informações sobre os registros, mostrando a data do registro.

(e) Requisitos não funcionais: Nenhum

Fluxo Básico:

1. O ator abre a lista
2. Os registros são listados

B.2.13 UC-02 REMOVER REGISTRO

Este caso de uso remove um registro do aplicativo, apagando seu registro.

(a) Atores: Usuário

(b) Pré-Condições: Algum registro precisa estar cadastrado no app

(c) Pós-Condições: O registro precisa ser totalmente apagado do sistema

(d) Requisitos Funcionais: Remover completamente o registros dos jsons do sistema

(e) Requisitos não funcionais: Nenhum

Fluxo Básico:

1. O ator abre a lista
2. Os registros são listados
3. O ator escolhe a ação "Remover"em um dos registros
4. O registro é apagado do sistema

B.2.14 UC-03 CADASTRAR REGISTRO

Aqui o usuário pode inserir na aplicação novos registros, inserindo o valor apropriado para cada fator cadastrado para o acompanhamento

(a) Atores: Usuário

(b) Pré-Condições: Nenhuma

(c) Pós-Condições: Nenhuma

(d) Requisitos Funcionais: Cadastrar um registro no sistema

(e) Requisitos não funcionais: Nenhum

Fluxo Básico:

1. O ator abre a lista
2. Os registros são listados
3. O ator escolhe a ação "Criar"

4. O formulário é mostrado na tela
5. O ator preenche o formulário
6. O ator clica em "salvar"e o registro é registrado
7. Volta para listagem

Fluxo Alternativo 1:

1. No passo 6, ao clicar em "salvar"e algum campo obrigatório não foi preenchido, é mostrada uma mensagem de erro
2. Não volta para tela de listagem, continua no formulário

B.2.15 UC-04 ALTERAR REGISTRO

Aqui o usuário pode editar registros previamente cadastrados, configurando suas informações

(a) Atores: Usuário

(b) Pré-Condições: Algum registro precisa estar previamente cadastrado

(c) Pós-Condições: Nenhuma

(d) Requisitos Funcionais: Editar um registro no sistema

(e) Requisitos não funcionais: Nenhum

Fluxo Básico:

1. O ator abre a lista
2. Os registros são listados
3. O ator escolhe a ação "Editar"em algum registro
4. O formulário é mostrado na tela com os dados preenchidos
5. O ator preenche o formulário
6. O ator clica em "salvar"e o registro é registrado
7. Volta para listagem

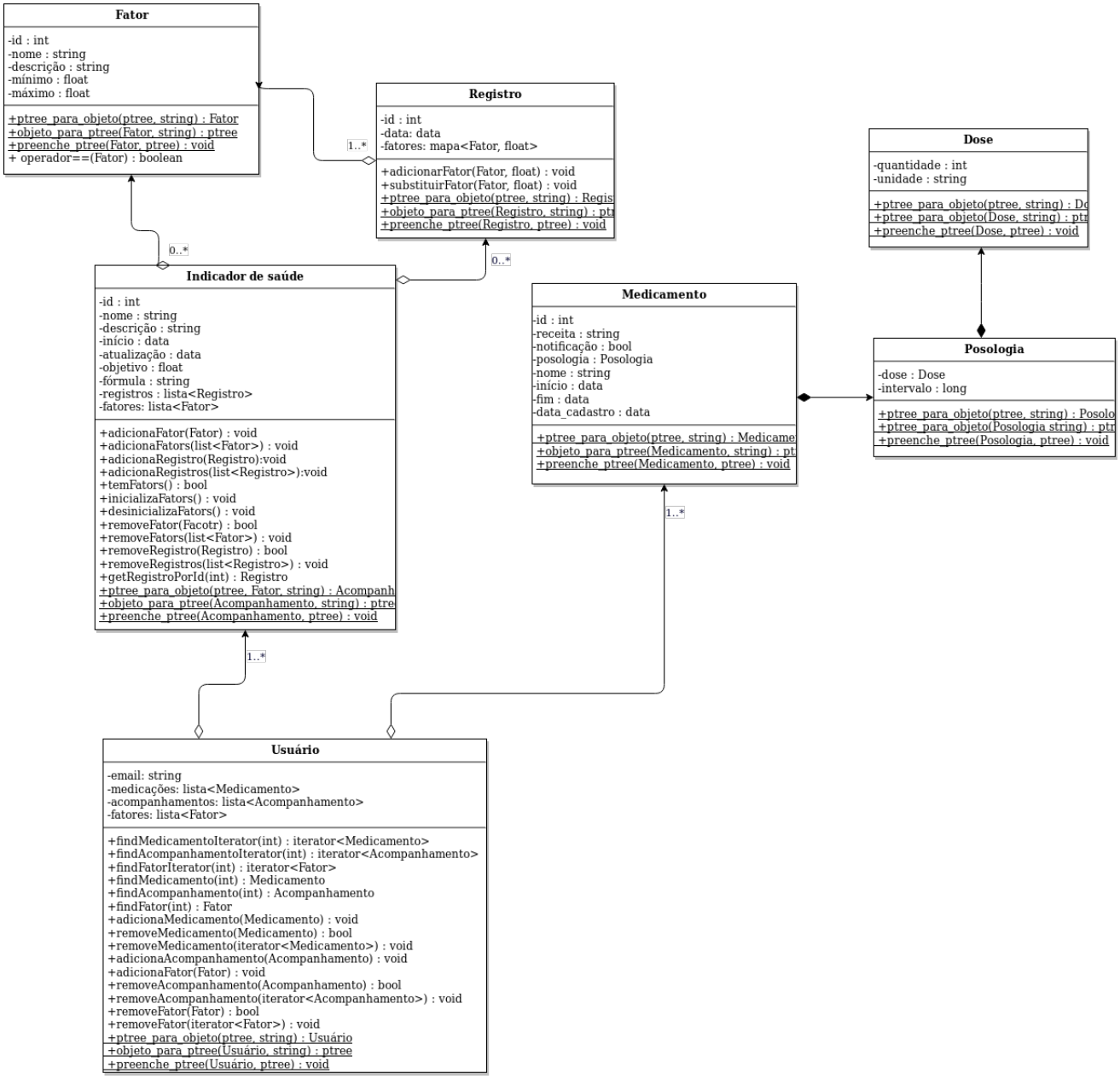
Fluxo Alternativo 1:

1. No passo 6, ao clicar em "salvar"e algum campo obrigatório não estiver preenchido, é mostrada uma mensagem de erro
2. Não volta para tela de listagem, continua no formulário

APÊNDICE C – DIAGRAMA DE CLASSES

A figura 38 representa o diagrama de classes do sistema.

FIGURA 38. DIAGRAMA DE CLASSES DO SISTEMA.



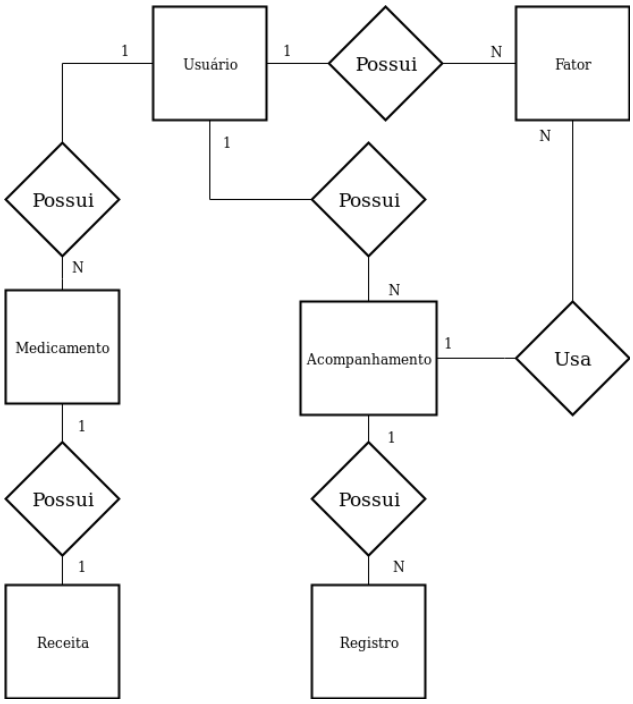
Fonte. O AUTOR (2017).

APÊNDICE D – DOCUMENTAÇÃO JSON

D.1 DER

As figura 39 representa o DER do sistema.

FIGURA 39. DIAGRAMA DE ENTIDADE RELACIONAMENTO CONTEITUAL.



Fonte. O AUTOR (2017).

D.2 JSON SCHEMA

D.2.1 MEDICAMENTO

Código 4 – JSON Schema de medicamento.

```
{
        "título": "Medicamento",
        "tipo": "objeto",
        "propriedades": {
            "id": {
                "tipo": "inteiro",
                "descrição": "Identificador único para dinstinguir o obje
            }
                "nome": {
                        "tipo": "texto",
                        "tamanho": "30"
                },
                "posologia": {
                    "tipo":"tupla",
                        "dose":{
                                "quantidade": {
                                    "tipo":"ponto flutuante",
                                },
                                "unidade": {
                                    "descrição":"A unidade a ser utilizad
 (por exemplo : ml, cp.)",
                                    "tipo":"texto",
                                    "tamanho":"15"
                                }
                        },
                        "intervalo": {
                        "tipo":"tempo"
            }
                },
                "início": {
                    "descrição": "Data e hora do início do medicamento.",
                    "tipo": "data-tempo",
                    "precisão": "minutos"
                },
```

```
35                  "fim": {
36                      "descrição": "Data e hora do fim do medicamento.",
37                      "tipo": "data-tempo",
38                      "precisão": "minutos"
39                  },
40                  "receita": {
41                      "descrição": "caminho para a foto da receita, opciona
42                      "tipo": "texto"
43                  },
44                  "data": {
45                      "tipo": "data-tempo",
46                      "descrição": "Data e hora do cadastro do medicamento.
47                  },
48                  "notificar": {
49                      "tipo": "boolean",
50                      "descrição": "Indicativo para notificar ou não o usuá
51                  },
52                  "continuous":{
53                      "tipo": "boolean",
54                      "descrição": "Indicativo se um medicamento é
    contínuo ou não."
55                  }
56
57          },
58          "obrigatório": ["nome", "posologia", "início", "data"]
59  }
```

D.2.1.1 DESCRIÇÃO TEXTUAL

Medicamento

(a) id

- Validação
  - somente números
  - deve ser um número único para o tipo do objeto.
  - maior ou igual a 0

(b) nome

- Validação

        - somente letras e números
        - comprimento maior que 0 e menor ou igual a 30

(c) posologia

- Validação
    - deve ser do tipo tupla (dosagem e intervalo)
- Elementos
    - Intervalo
        * Validação
            · somente números
            · deve ser maior que 0
            · medido em millisegundos
    - Dosagem
        * Elementos
            · Quantidade
            · somente números (ponto flutuante)
            · deve ser maior que 0
            · Unidade
            · somente números
            · deve ser maior que 0

(d) início

- Validação
    - tipo data
    - anterior ao fim se não for contínuo

(e) fim

- Opcional
- Validação
    - tipo data
    - posterior ao início

(f) receita

- Opcional
- Validação

  - somente caracteres de caminhos de arquivo no sistema operacional

(g) data

- Validação
  - tipo data

(h) notificar

- Validação
  - tipo booleano

(i) contínuo

- Validação
  - tipo booleano

D.2.2 ACOMPANHAMENTO

Código 5 – JSON Schema de acompanhamento.

```
{
        "título": "Acompanhamento",
        "tipo": "objeto",
        "propriedades": {
            "id":{
              "tipo": "inteiro",
              "descrição": "Identificador único de acompanhamento."
            },
                "nome": {
                        "tipo": "texto",
                        "tamanho": "20"
                },
                "descrição": {
                        "tipo": "texto",
                        "tamanho": "200",
                        "descrição":"A descrição do acompanhamento."
                },
                "início": {
                        "tipo": "data"
                },
                "atualização": {
                    "descrição":"A última data de atualização deste fator
                        "tipo": "data-tempo"
                },
                "fatores": {
                    "descrição":"Os fatores permitidos para serem
usados como entrada de um registro.",
                    "tipo":"vetor",
                    "itens":{
                        "tipo":"marker-factor"
                    }
                },
                "registros": {
                    "descrição":"The array of records",
                    "tipo":"vetor",
                    "itens":{
```

```
37                          "tipo":"Registro"
38                      }
39                  },
40                  "objetivo": {
41                      "descrição":"O objetivo do acompanhamento, opcional."
42                      "tipo":"ponto flutuante"
43                  },
44                  "formula":{
45                      "tipo":"texto",
46                      "tamanho":"500",
47                      "descrição":"Uma fórmula que utiliza
48 fatores para gerar um resultado para o registro, opcional."
49                  }
50          },
51          "obrigatório": ["nome", "início"]
52 }
```

D.2.2.1 DESCRIÇÃO TEXTUAL

Acompanhamento

(a) id

- Validação
  - somente números
  - deve ser um número único para o tipo do objeto.
  - maior ou igual a 0

(b) nome

- Validação
  - somente letras e números
  - comprimento maior que 0 e menor ou igual a 30

(c) descrição

- Validação
  - somente letras e números
  - comprimento maior que 0

(d) início

- Validação
  - tipo data

(e) atualização

- Opcional
- Validação
  - tipo data

(f) fatores

- Validação
  - lista de fatores permitidos para o acompanhamento
  - quantidade de elementos maior ou igual a zero
  - elemento do tipo fator

(g) registros

- Validação
  - quantidade de elementos maior ou igual a zero
  - elemento do tipo registro

(h) objetivo

- Opcional
- Validação
  - somente números (ponto flutuante)

(i) formula

- Opcional
- Validação
  - somente caracteres de operadores matemáticos, letras e números
  - palavras iniciadas por uma letra devem ser nomes de fatores válidos

D.2.3 ACOMPANHAMENTO - FATOR

Código 6 – JSON Schema de fator de acompanhamento.

```
1  {
2          "título": "Fator",
3          "descrição":"Unidade de medida para ser utilizada em um acompanha
4          "tipo": "objeto",
5          "propriedades": {
6              "id":{
7                "tipo":"inteiro",
8                "descrição":"Identificação única do fator."
9              },
10                 "nome":{
11                     "tipo":"texto",
12                     "tamanho":"8",
13                     "descrição":"Uma abreviação, iniciais ou
14 nome curto para o fator. Somente caracteres alfanuméricos são
15  permitidos, sendo que deve começar com uma letra para
16  possibilitar inclusão na fórmula."
17                 },
18                 "descrição":{
19                     "tipo":"texto",
20                     "tamanho":"50",
21                     "descrição":"A descrição do fator."
22                 }
23         }
24         "obrigatório": ["nome"]
25 }
```

D.2.3.1 DESCRIÇÃO TEXTUAL

Fator

(a) id

- Validação
  - somente números
  - deve ser um número único para o tipo do objeto.
  - maior ou igual a 0

(b) nome

- Validação
  - somente letras e números
  - comprimento maior que 0 e menor ou igual a 8

(c) descrição

- Validação
  - somente letras e números
  - comprimento maior que 0

D.2.4 ACOMPANHAMENTO - REGISTRO

Código 7 – JSON Schema de registro de acompanhamento.

```
{
        "título": "Registro",
        "descrição":"Registro de acompanhamento.",
        "tipo": "objeto",
        "propriedades": {
            "id":{
              "tipo":"inteiro",
              "descrição":"Identificador único do registro."
            },
            "data":{
                "tipo":"data"
            },

            "fatores":{
                "descrição":"A lista de fatores referentes ao acompanham
                "tipo": "vetor",
                "itens": {
                    "tipo":"tuple",
                    "id_fator":{
                        "descrição":"O identificador único de fator",
                        "tipo": "inteiro"
                    },
                    "value":{
                        "tipo":"ponto flutuante",
                        "descrição":"O valor do fator"
                    }
                }
            },
            "observação":{
                "tipo": "texto",
                "descrição": "The optional note for this record."
            }
        }
        "obrigatório": ["data", "factors"]
}
```

D.2.4.1 DESCRIÇÃO TEXTUAL

Registro

(a) id

- Validação
  - somente números
  - deve ser um número único para o tipo do objeto.
  - maior ou igual a 0

(b) data

- Validação
  - tipo data

(c) fatores

- Validação
  - lista de ids de fator
  - cada elemento é um número maior ou igual a 0
  - cada elemento deve corresponder a um id de fator válido

(d) observação

- Validação
  - somente letras e números
  - comprimento maior que 0

D.2.5 USUÁRIO

Código 8 – JSON Schema de usuário.

```
{
        "título": "User",
        "tipo": "objeto",
        "propriedades": {
                "medicamentos":{
                    "tipo":"lista"
                    "itens":{
                        "tipo":"medication"
                    }
                },
                "acompanhamentos":{
                    "tipo":"lista",
                    "itens":{
                        "tipo":"acompanhamento"
                    }
                }
        },
        "obrigatório": []
}
```

D.2.5.1 DESCRIÇÃO TEXTUAL

Usuário

(a) fatores

- Validação
  - lista de ids de fator
  - cada elemento é um número maior ou igual a 0
  - cada elemento deve corresponder a um id de fator válido

(b) acompanhamentos

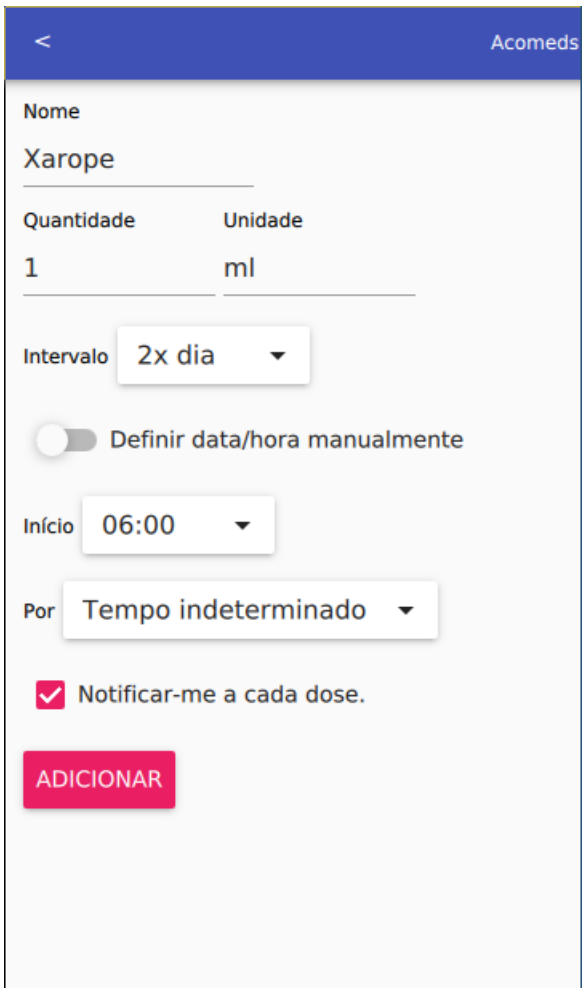
- Validação
  - lista de ids de acompanhamento
  - cada elemento é um número maior ou igual a 0
  - cada elemento deve corresponder a um id de acompanhamento válido

APÊNDICE E – DIAGRAMAS DE SEQUÊNCIA

E.1 MEDICAMENTO

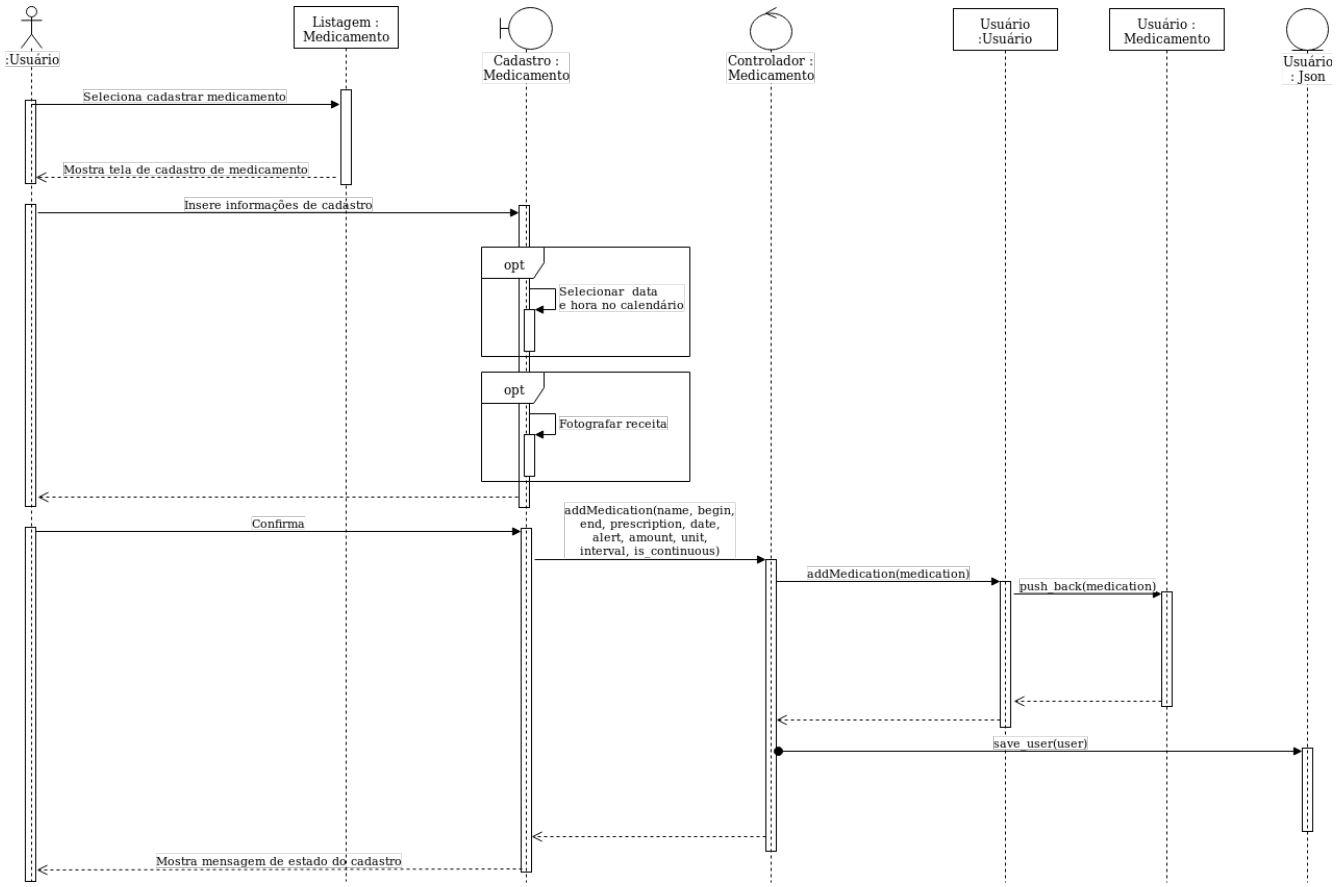
As figuras 41, 43, 45, 46 e 48 representam diagramas de sequência de medicamento. São apresentadas também as telas referentes aos diagramas.

FIGURA 40. CADASTRO DE MEDICAMENTOS. REFERENTE AOS DIAGRAMAS 41 E 43.



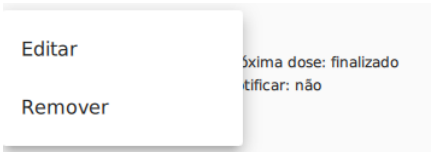
Fonte. O AUTOR (2017).

FIGURA 41. DIAGRAMA DE SEQUÊNCIA DE CADASTRO DE MEDICAMENTO.



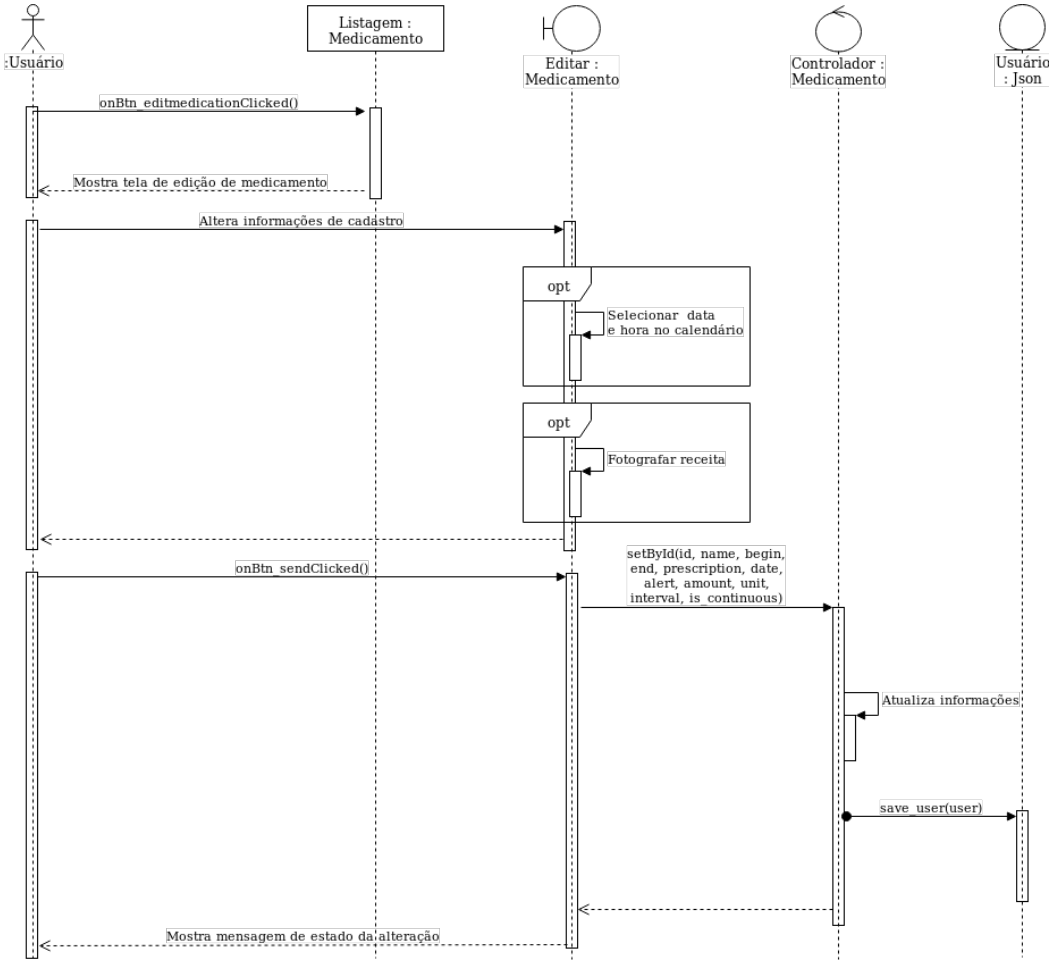
Fonte. O AUTOR (2017).

FIGURA 42. MENU DE CONTEXTO DE MEDICAMENTOS. REFERENTE AOS DIAGRAMAS 43 E 46.



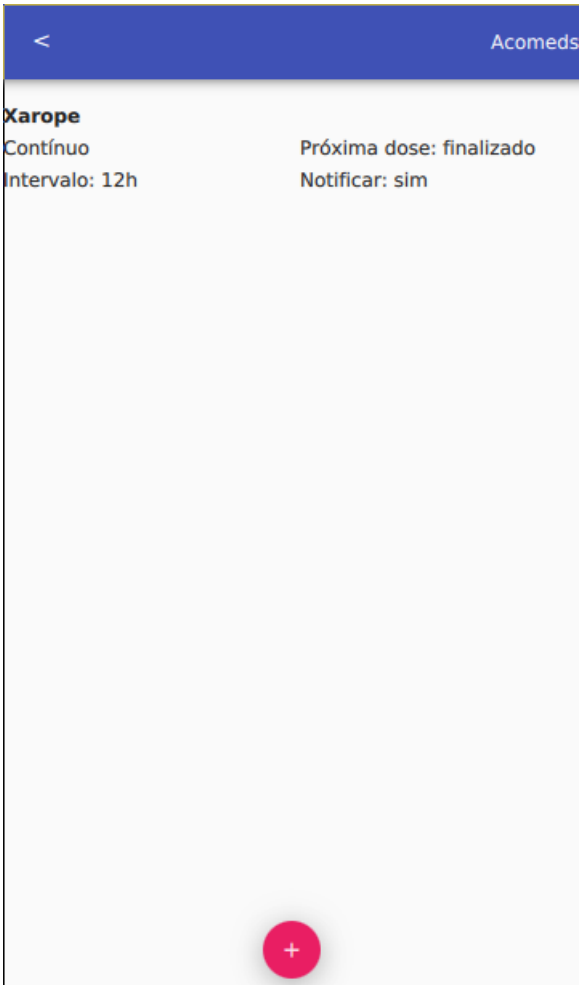
Fonte. O AUTOR (2017).

FIGURA 43. DIAGRAMA DE SEQUÊNCIA DE EDIÇÃO DE MEDICAMENTO.



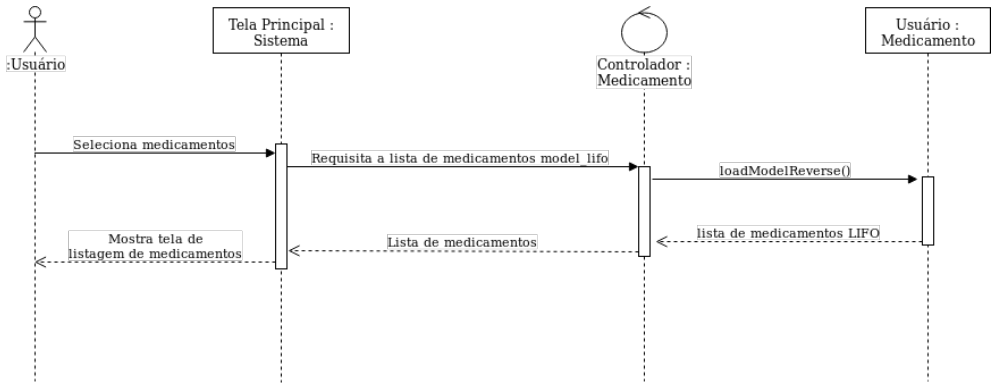
Fonte. O AUTOR (2017).

FIGURA 44. LISTA DE MEDICAMENTOS. REFERENTE AO DIAGRAMA 45.



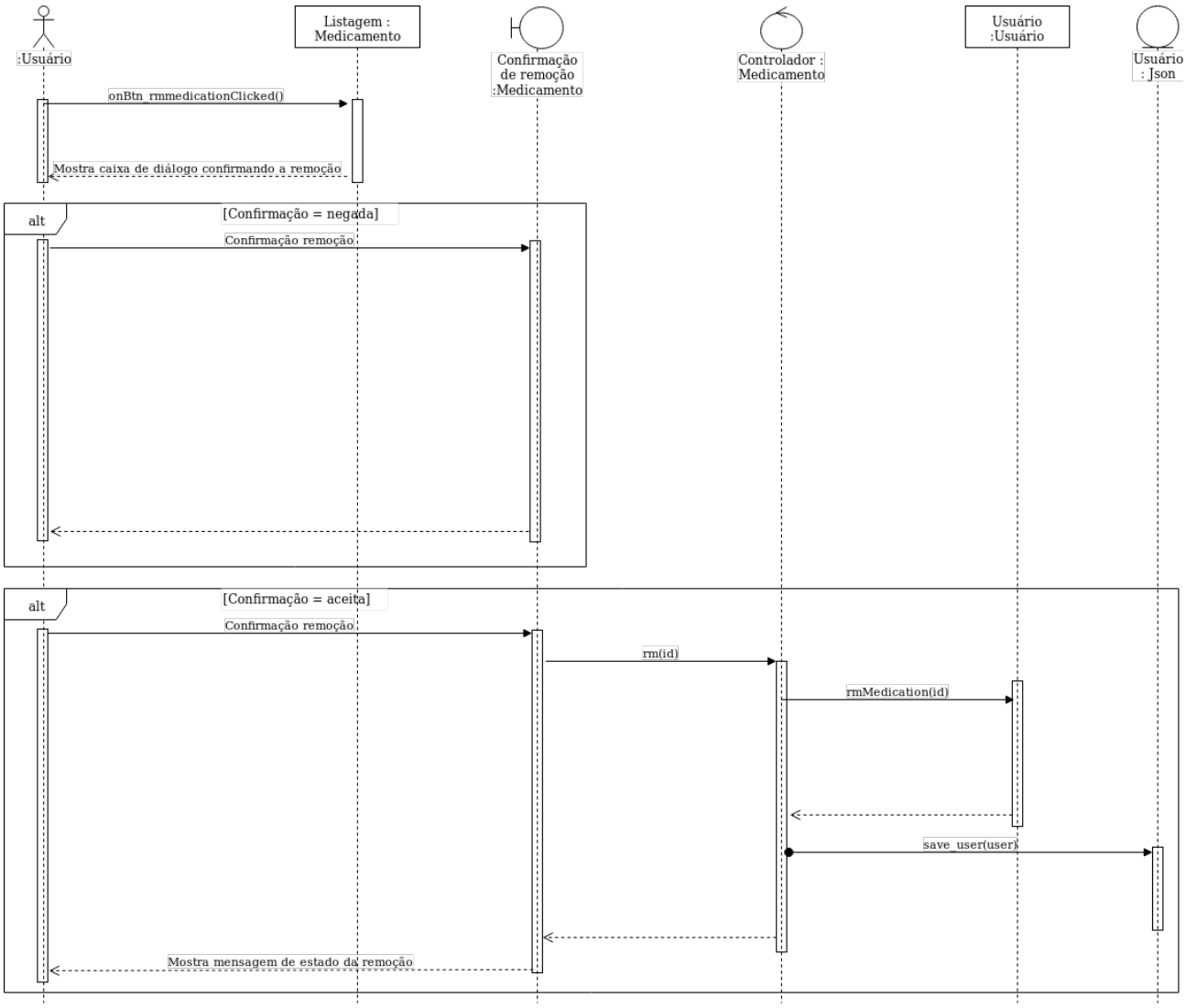
Fonte. O AUTOR (2017).

FIGURA 45. DIAGRAMA DE SEQUÊNCIA DE LISTAGEM DE MEDICAMENTO.



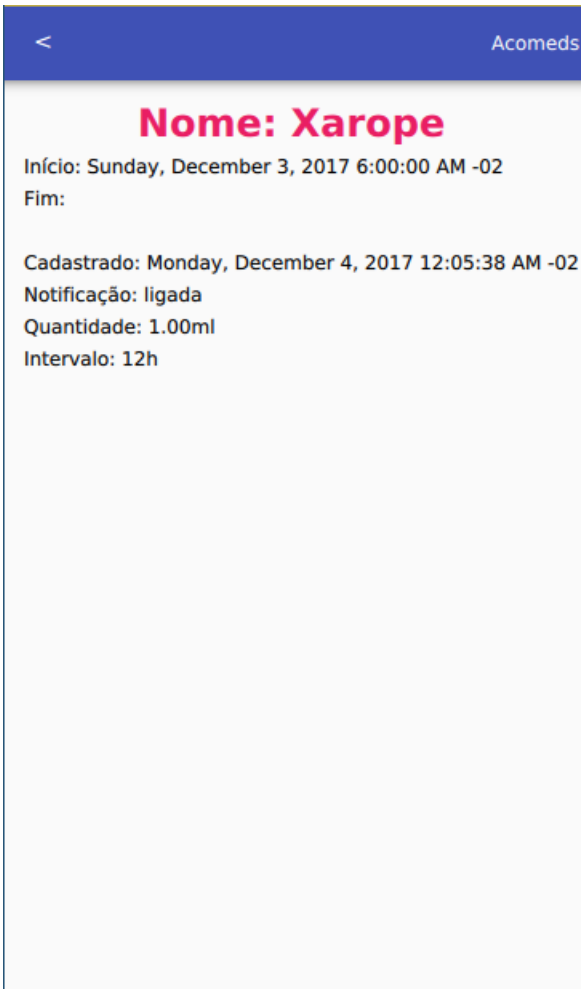
Fonte. O AUTOR (2017).

FIGURA 46. DIAGRAMA DE SEQUÊNCIA DE REMOÇÃO DE MEDICAMENTO.



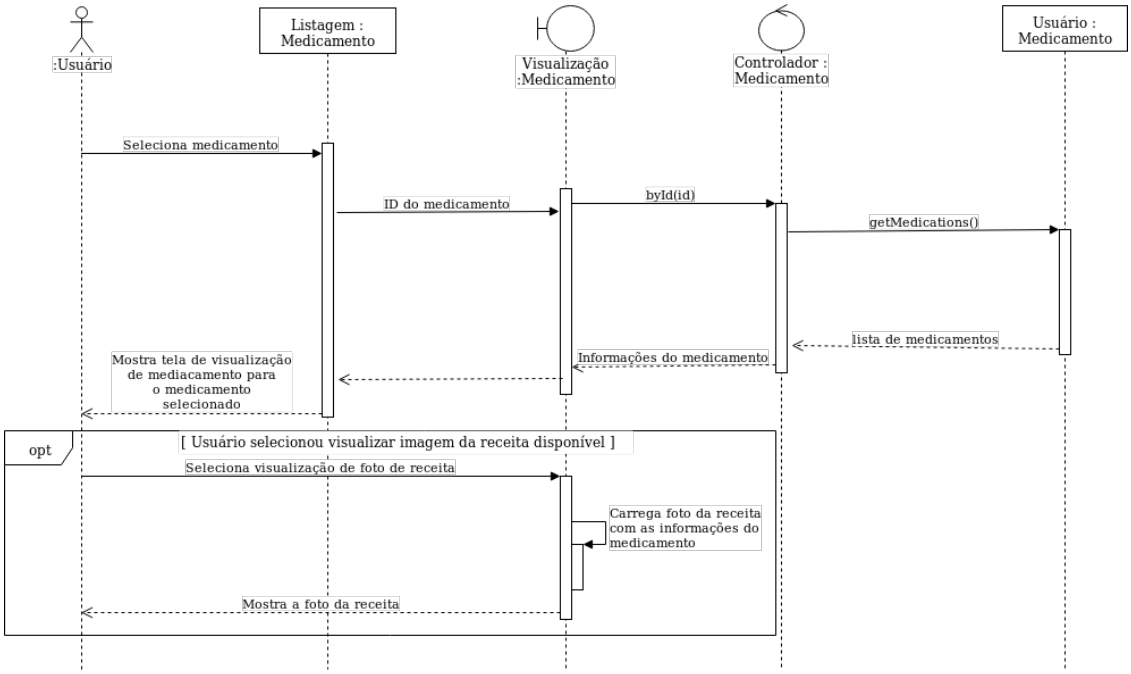
Fonte. O AUTOR (2017).

FIGURA 47. VISUALIZAÇÃO DE MEDICAMENTO. REFERENTE AO DIAGRAMA 48.



Fonte. O AUTOR (2017).

FIGURA 48. DIAGRAMA DE SEQUÊNCIA DE VISUALIZAÇÃO DE MEDICAMENTO.

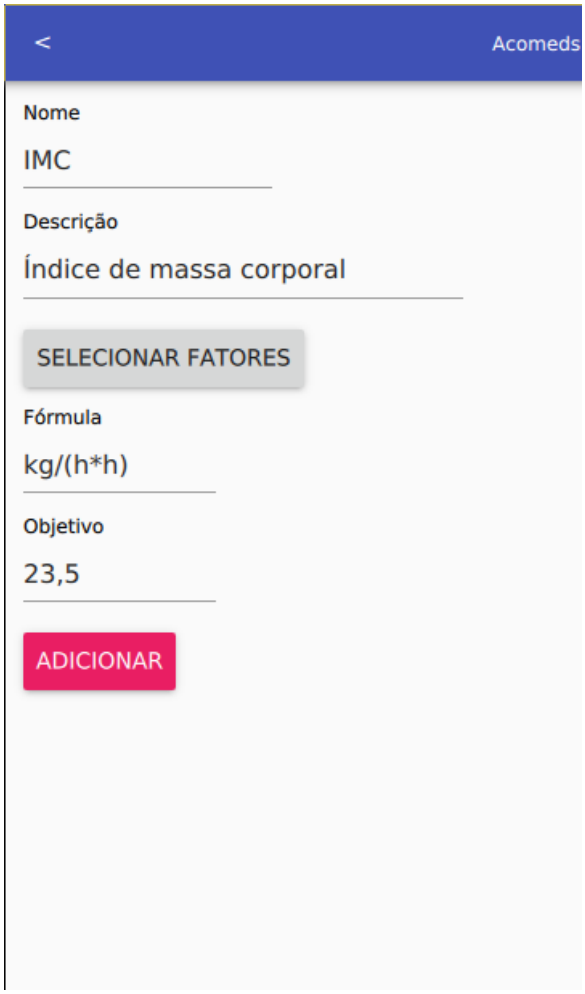


Fonte. O AUTOR (2017).

E.2 ACOMPANHAMENTO

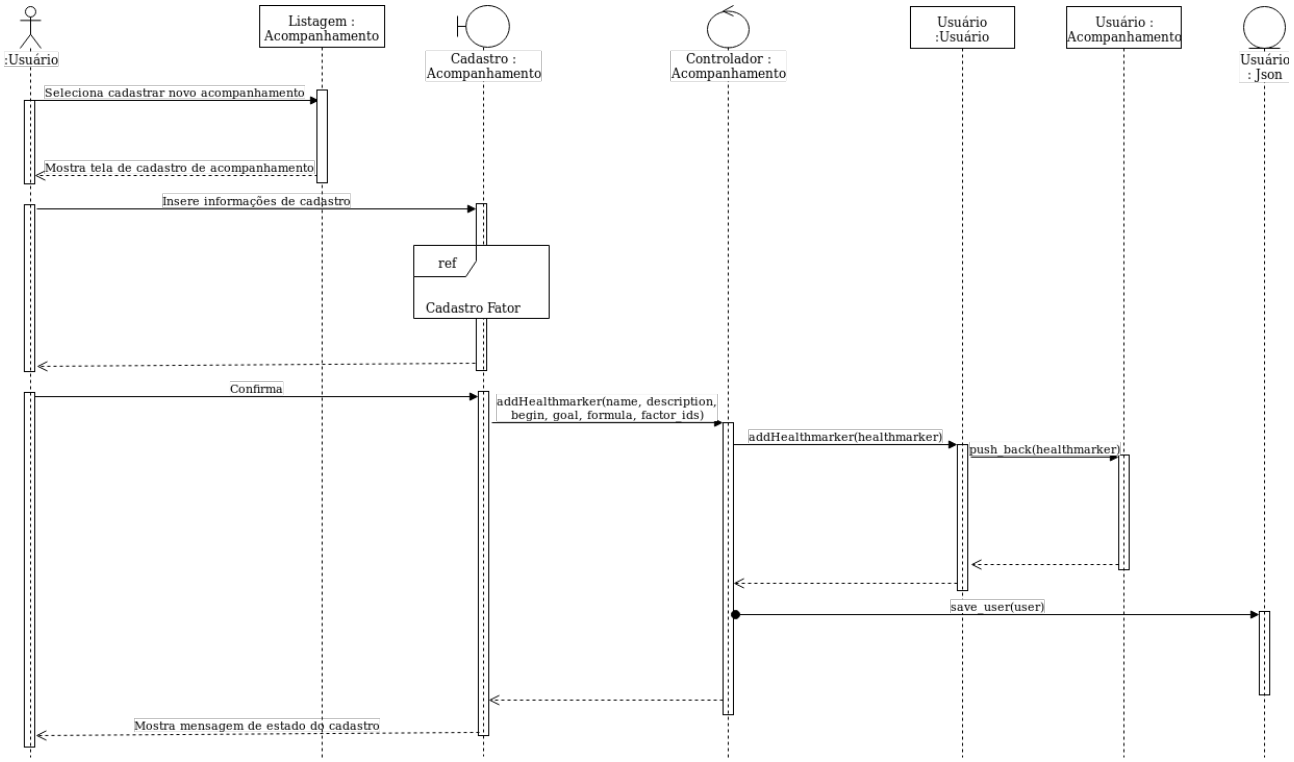
As figuras 50, 52, 54, 55 e 57 representam diagramas de sequência de acompanhamento. São apresentadas também as telas referentes aos diagramas.

FIGURA 49. CADASTRO E ALTERAÇÃO DE ACOMPANHAMENTO. REFERENTE AOS DIAGRAMAS 50 E 52.



Fonte. O AUTOR (2017).

FIGURA 50. DIAGRAMA DE SEQUÊNCIA DE CADASTRO DE ACOMPANHAMENTO.

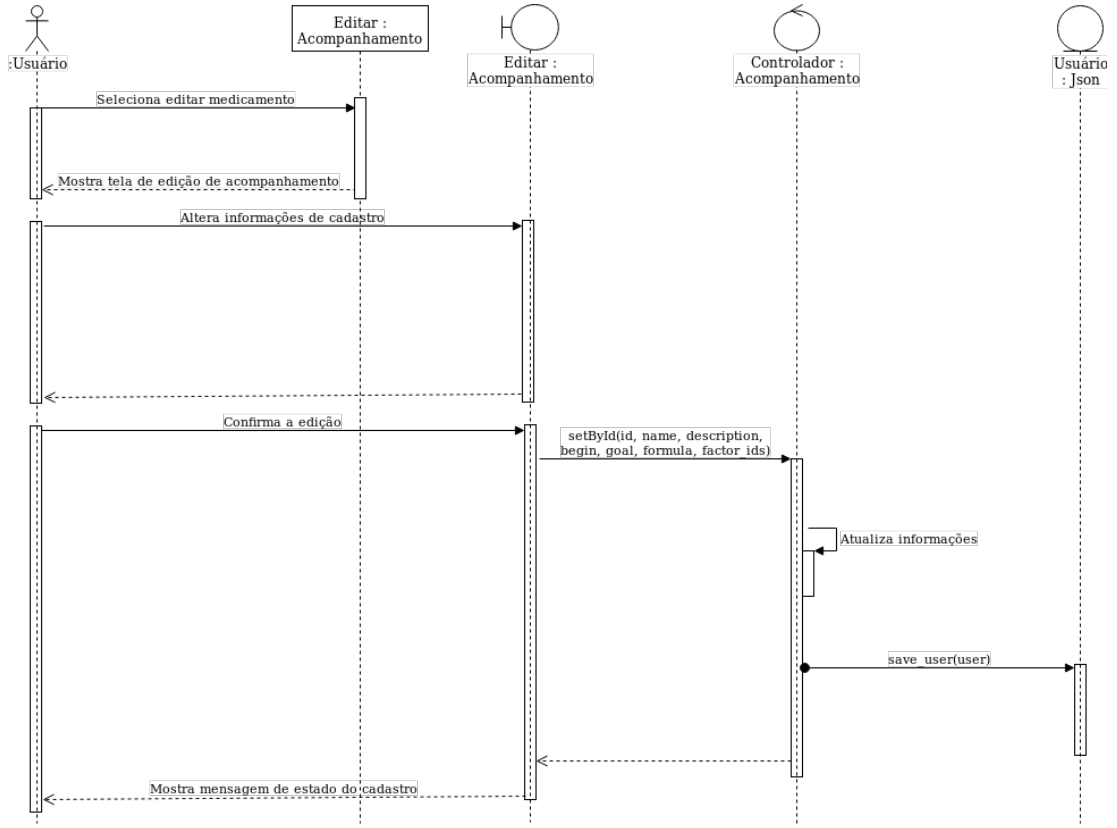


Fonte. O AUTOR (2017).

FIGURA 51. MENU DE CONTEXTO DA LISTA DE ACOMPANHAMENTOS. REFERENTE AOS DIAGRAMAS 52 E 55.

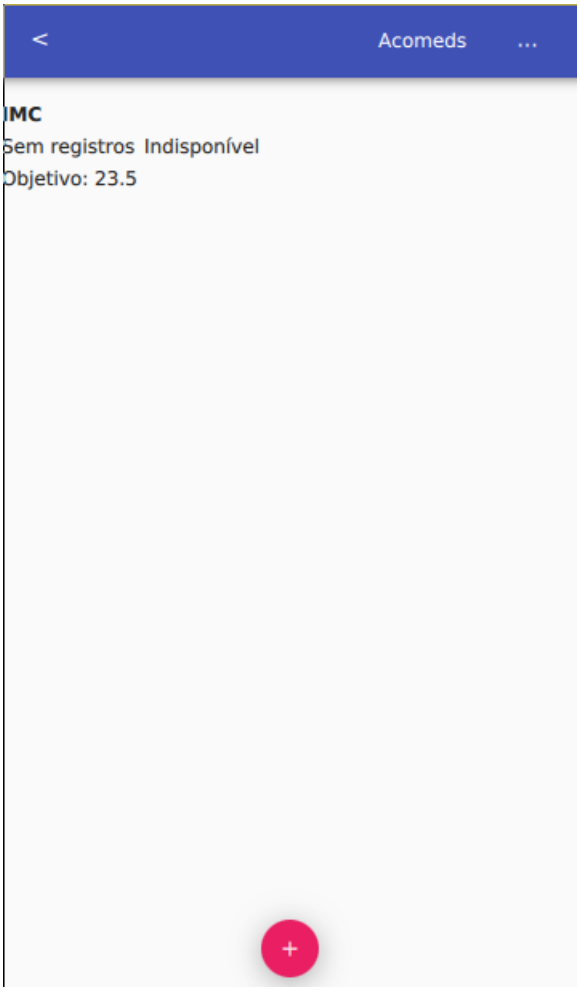
Fonte. O AUTOR (2017).

FIGURA 52. DIAGRAMA DE SEQUÊNCIA DE EDIÇÃO DE ACOMPANHAMENTO.



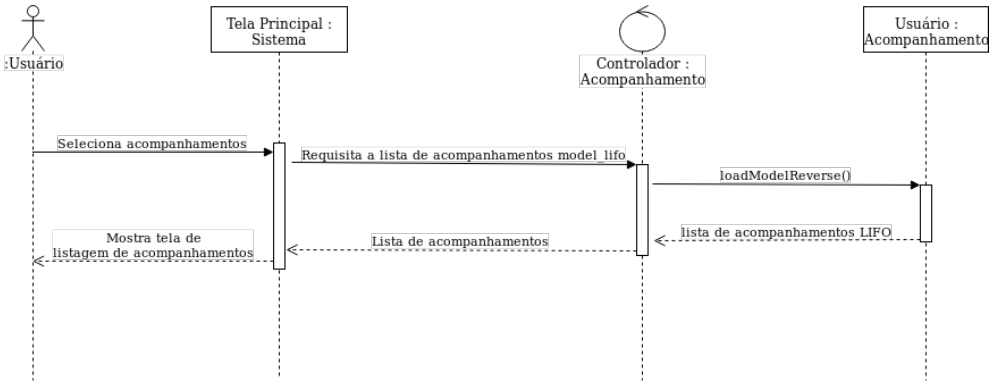
Fonte. O AUTOR (2017).

FIGURA 53. LISTA DE ACOMPANHAMENTOS. REFERENTE AO DIAGRAMA 54.



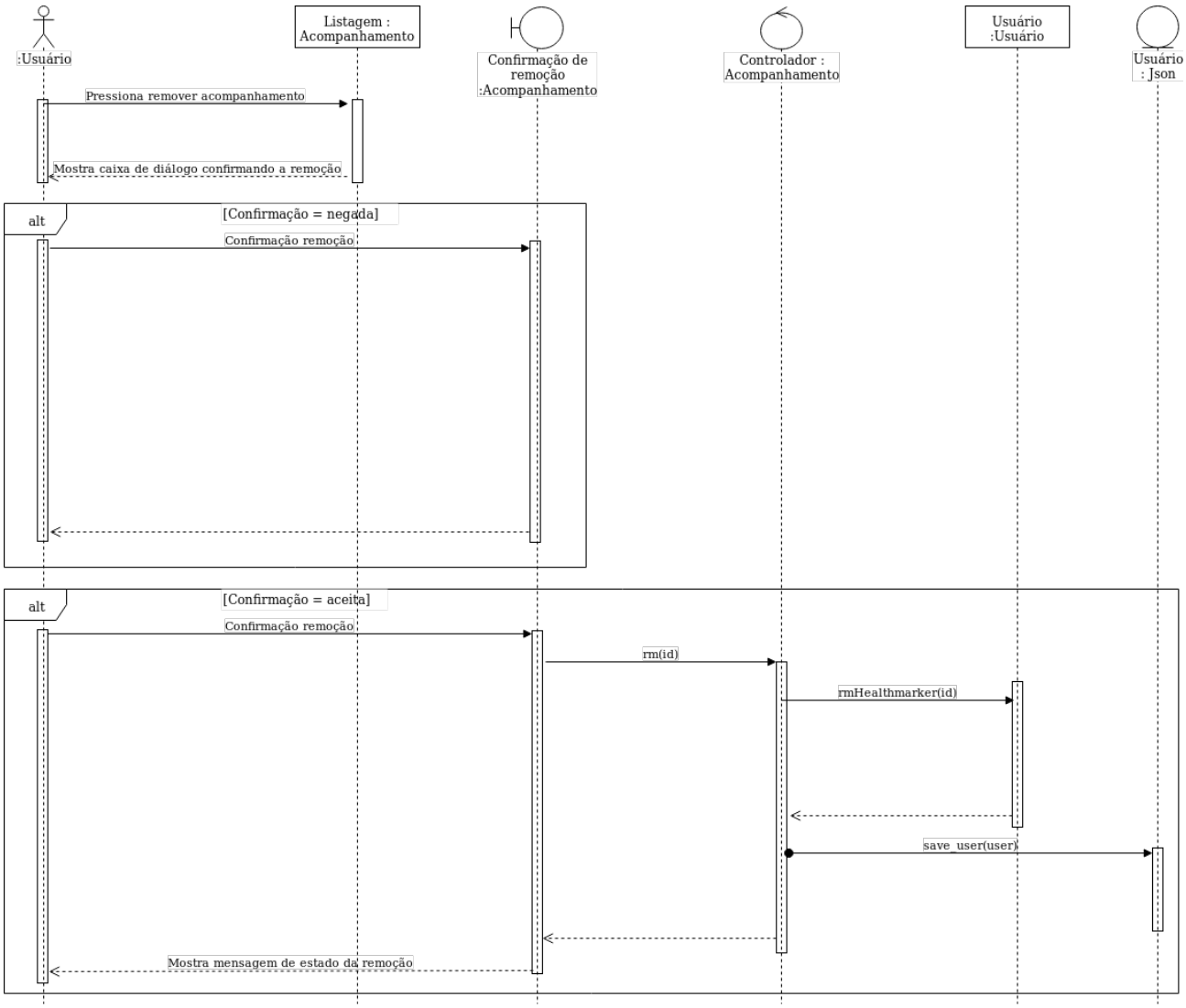
Fonte. O AUTOR (2017).

FIGURA 54. DIAGRAMA DE SEQUÊNCIA DE LISTAGEM DE ACOMPANHAMENTO.



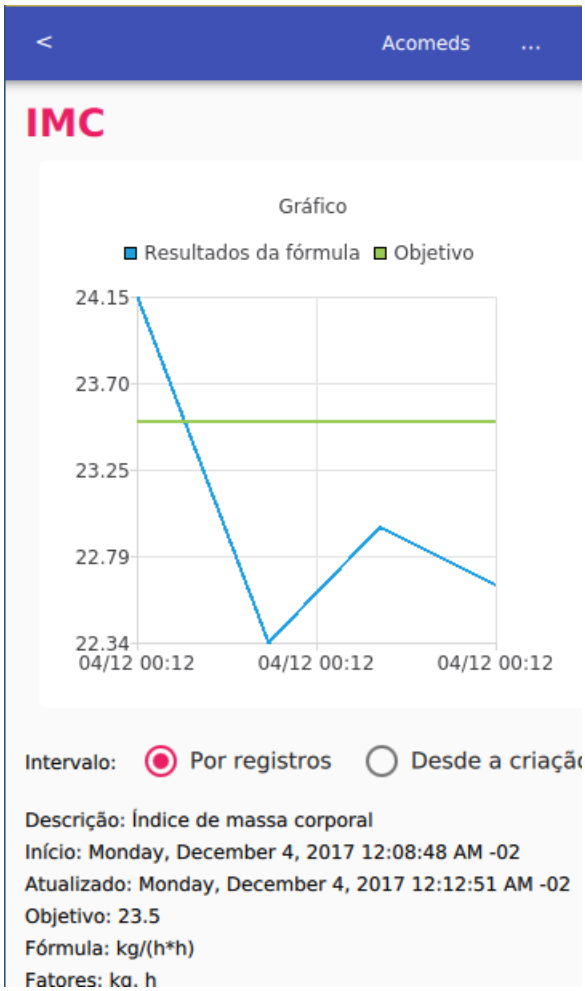
Fonte. O AUTOR (2017).

FIGURA 55. DIAGRAMA DE SEQUÊNCIA DE REMOÇÃO DE ACOMPANHAMENTO.



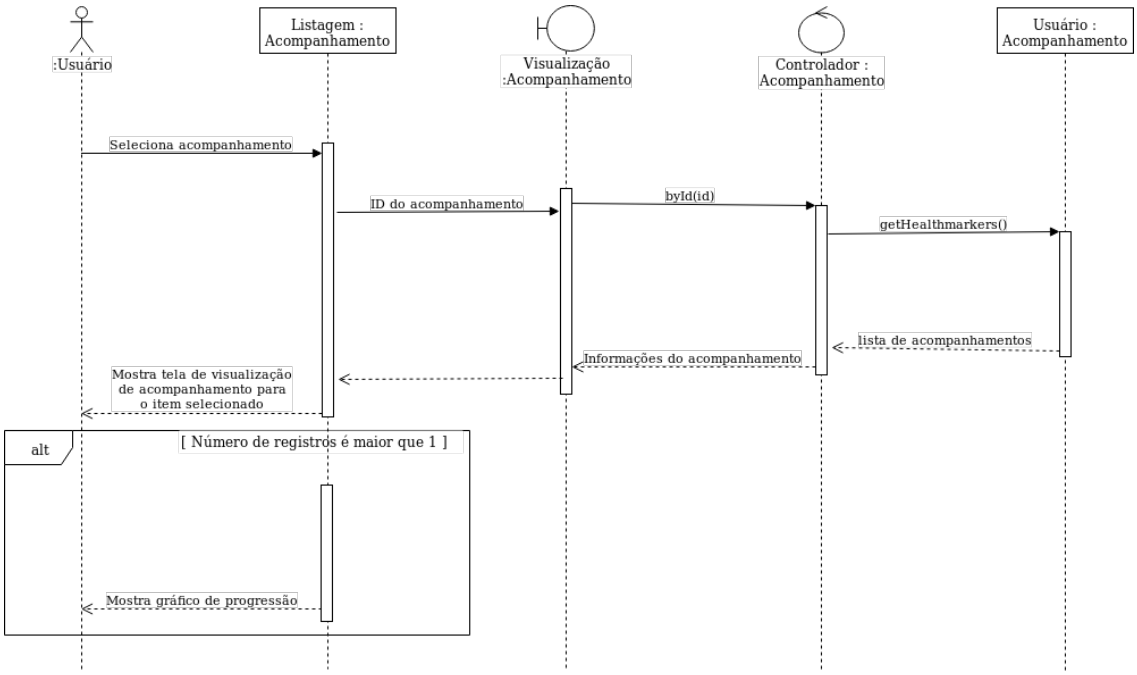
Fonte. O AUTOR (2017).

FIGURA 56. VISUALIZAÇÃO DE ACOMPANHAMENTO. REFERENTE AO DIAGRAMA 57



Fonte. O AUTOR (2017).

FIGURA 57. DIAGRAMA DE SEQUÊNCIA DE VISUALIZAÇÃO DE ACOMPANHAMENTO.

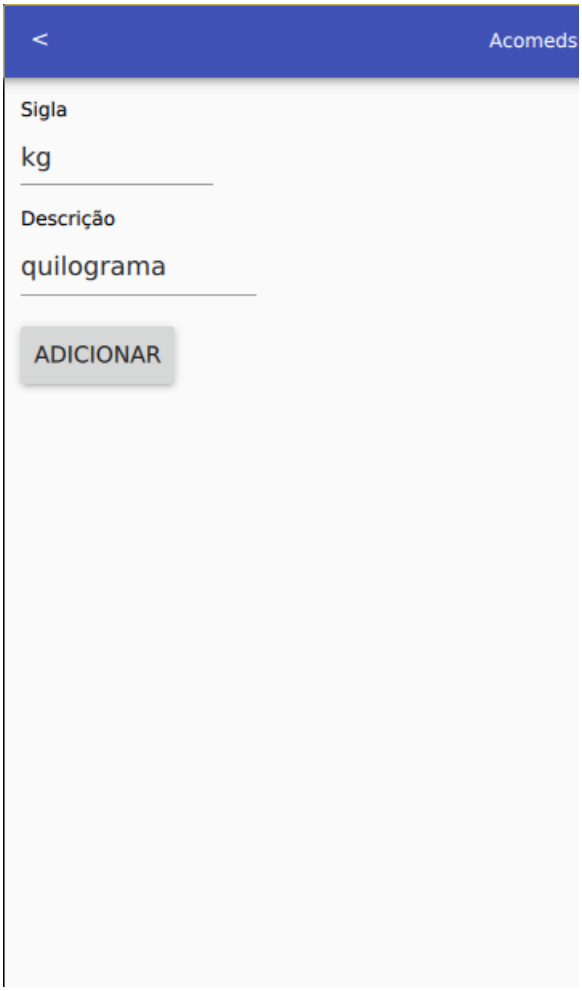


Fonte. O AUTOR (2017).

E.3 FATOR

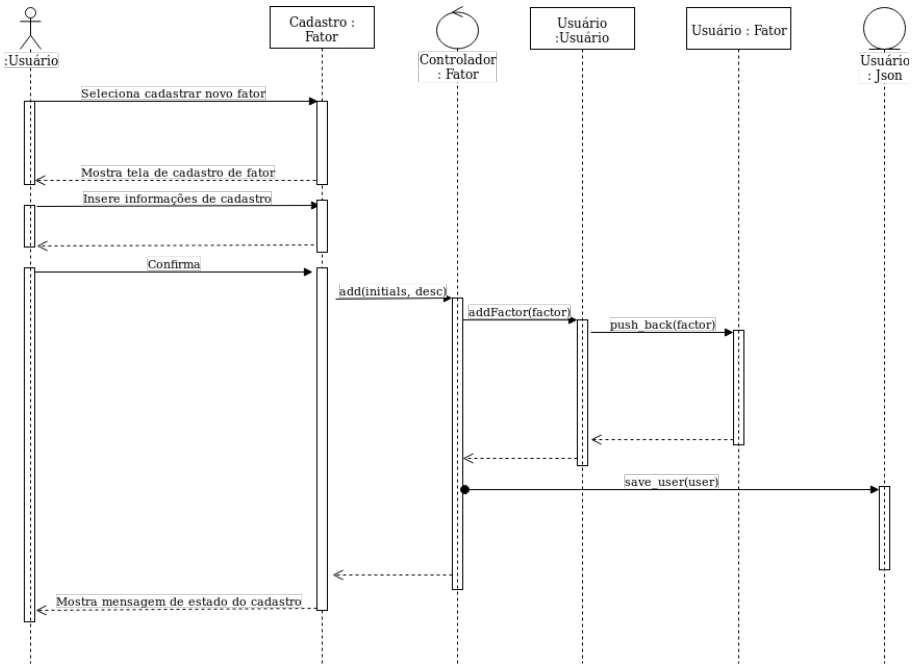
As figuras 59, 60, 62, 63 e 65 representam diagramas de sequência de fator. São apresentadas também as telas referentes aos diagramas.

FIGURA 58. CADASTRO E ALTERAÇÃO DE FATOR. REFERENTE AOS DIAGRAMAS 59 E 60.



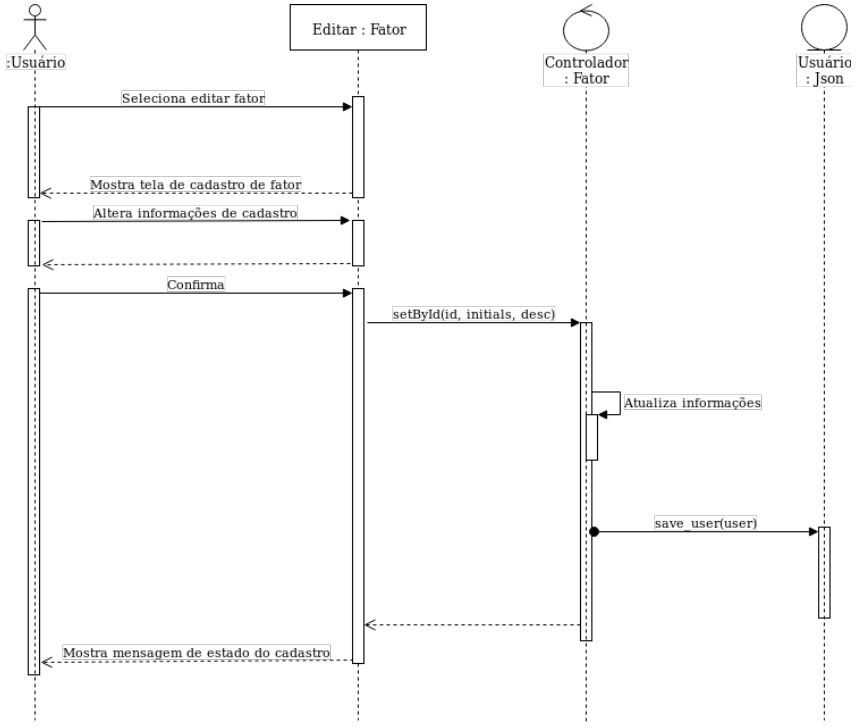
Fonte. O AUTOR (2017).

FIGURA 59. DIAGRAMA DE SEQUÊNCIA DE CADASTRO DE FATOR.



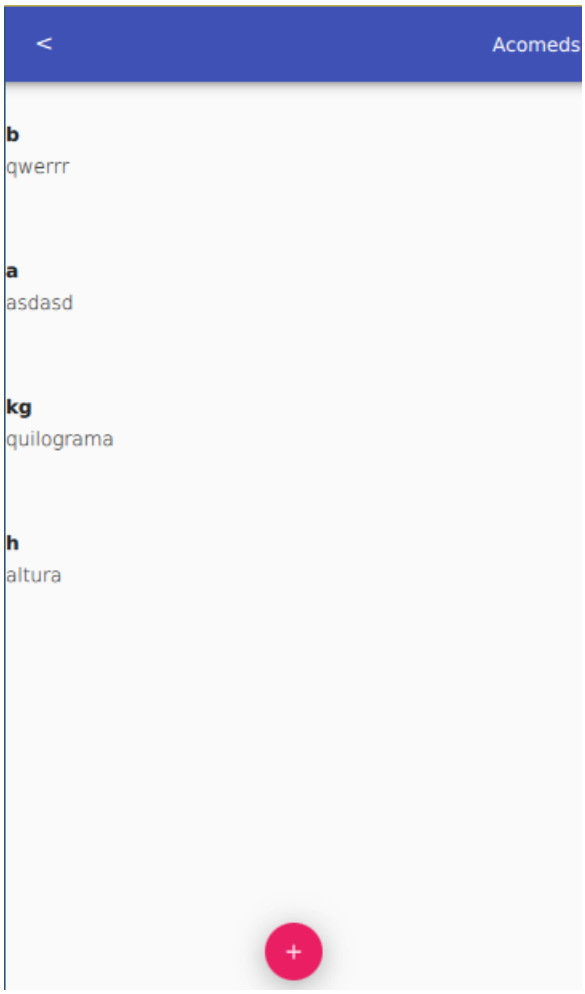
Fonte. O AUTOR (2017).

FIGURA 60. DIAGRAMA DE SEQUÊNCIA DE EDIÇÃO DE FATOR.



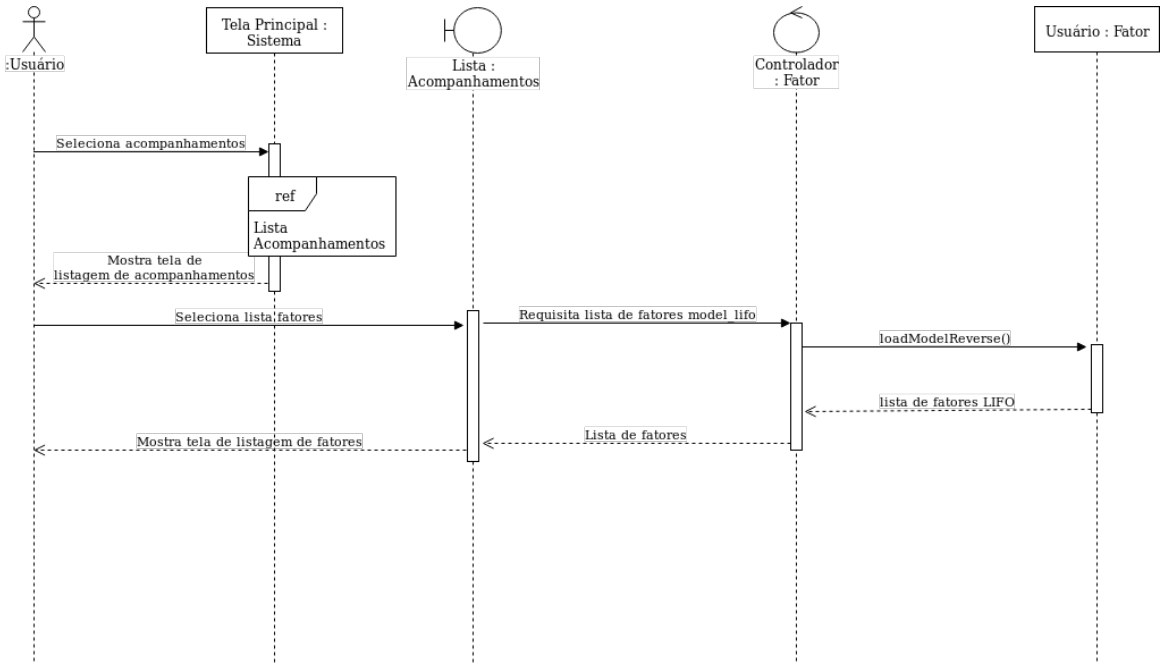
Fonte. O AUTOR (2017).

FIGURA 61. LISTA DE FATORES CADASTRADOS. REFERENTE AO DIAGRAMA 62.



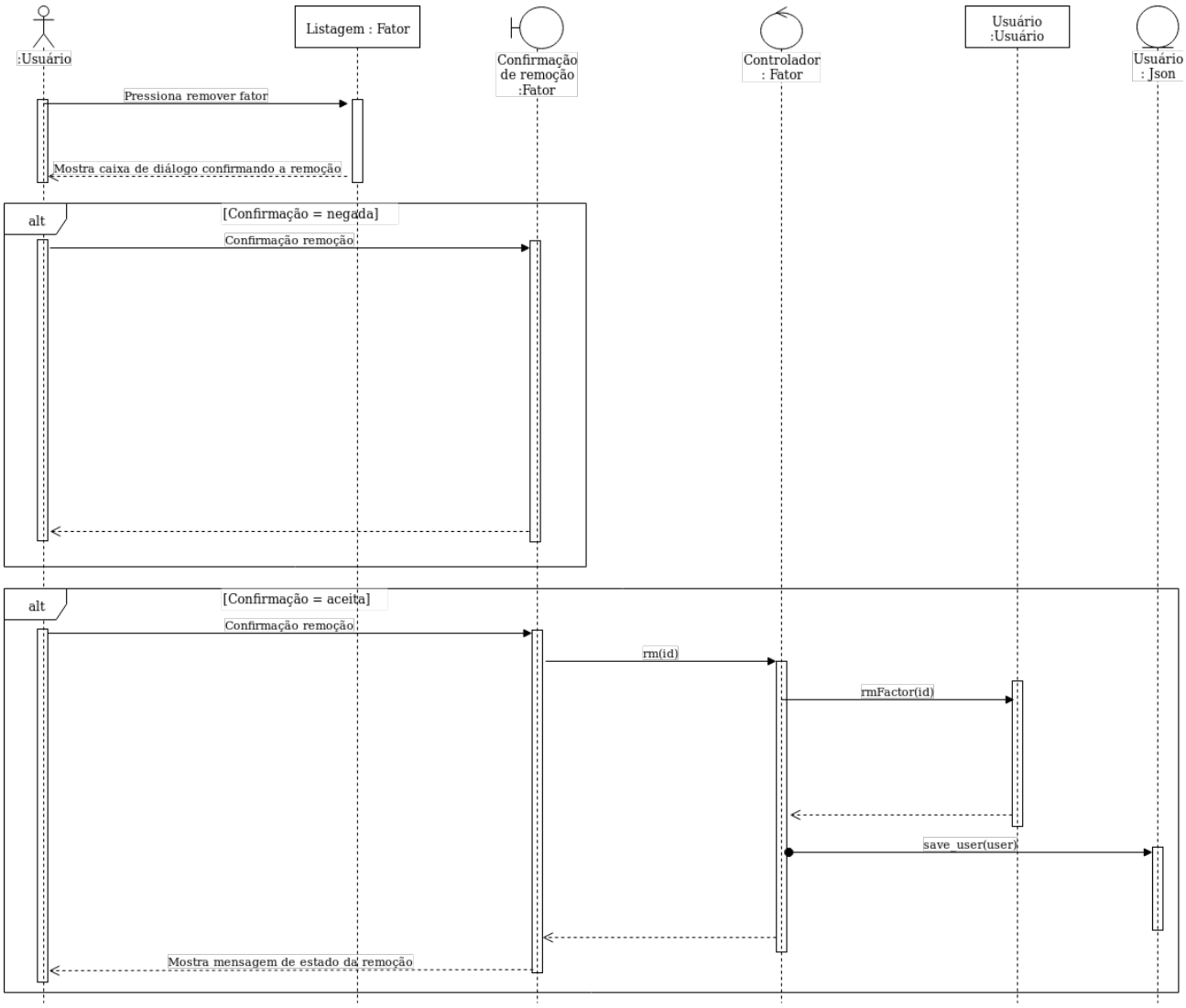
Fonte. O AUTOR (2017).

FIGURA 62. DIAGRAMA DE SEQUÊNCIA DE LISTAGEM DE FATOR.



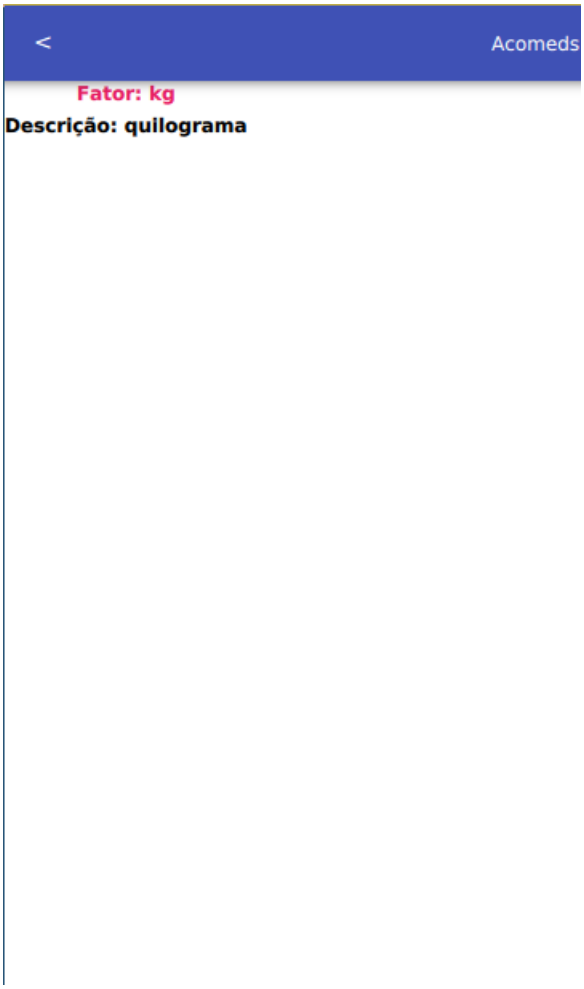
Fonte. O AUTOR (2017).

FIGURA 63. DIAGRAMA DE SEQUÊNCIA DE REMOÇÃO DE FATOR.



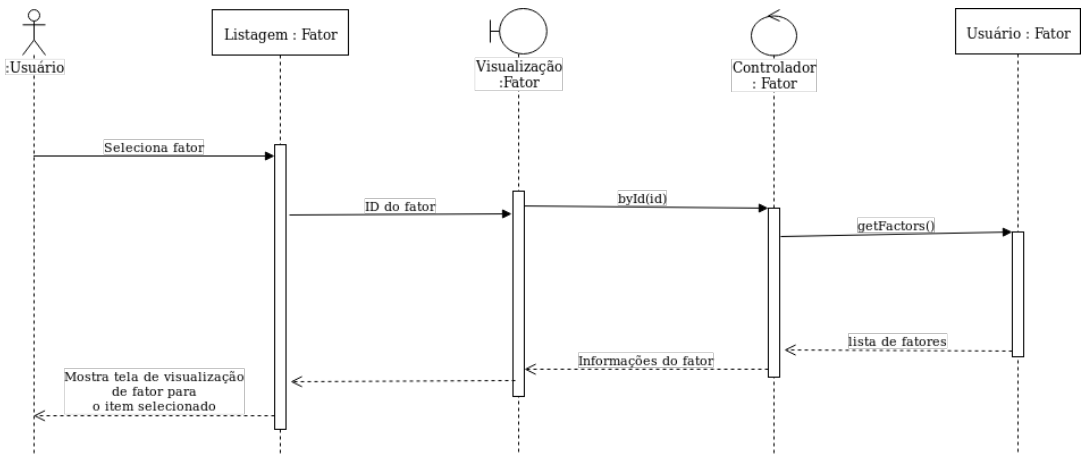
Fonte. O AUTOR (2017).

FIGURA 64. VISUALIZAÇÃO DE FATOR. REFERENTE AO DIAGRAMA 65.



Fonte. O AUTOR (2017).

FIGURA 65. DIAGRAMA DE SEQUÊNCIA DE VISUALIZAÇÃO DE FATOR.

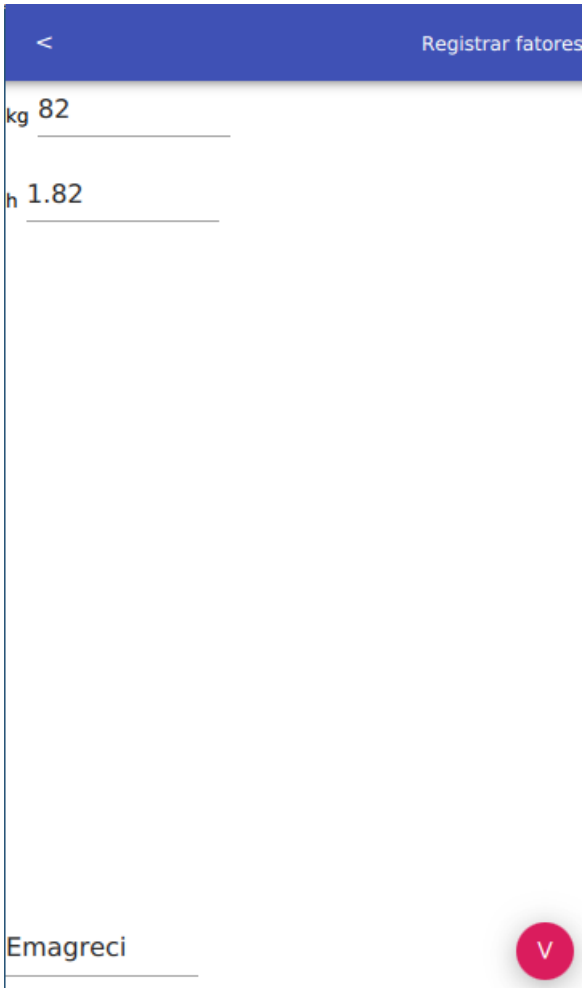


Fonte. O AUTOR (2017).

E.4 REGISTRO

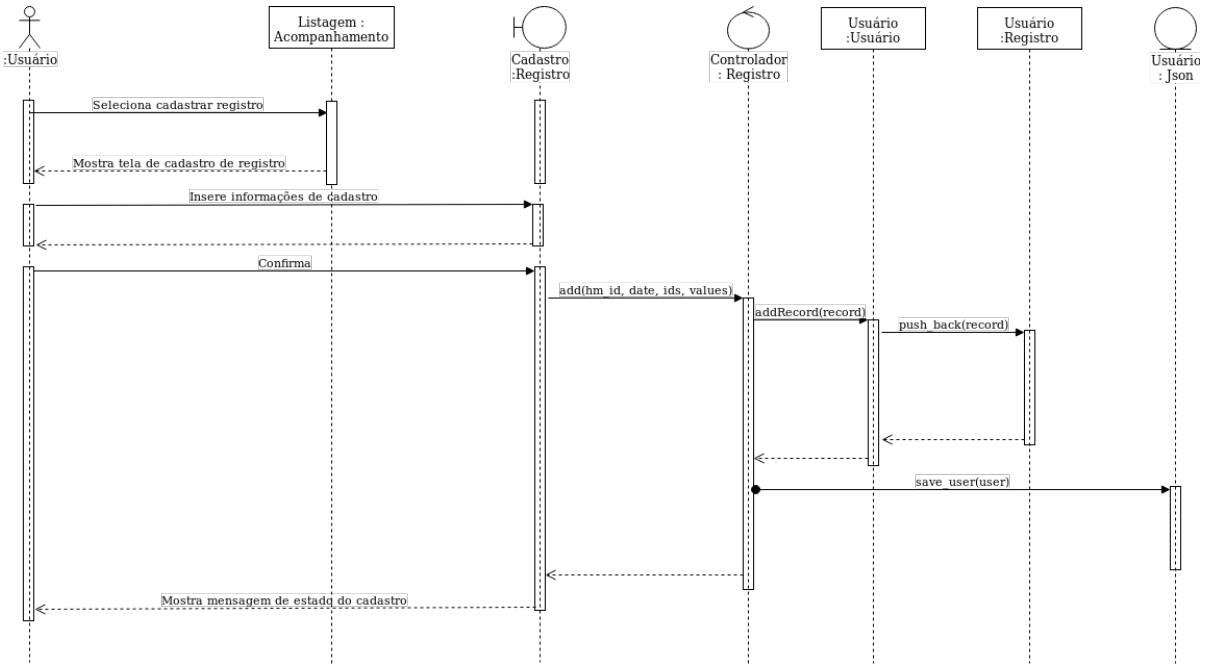
As figuras 67, 68, 70, 71 e 73 representam diagramas de sequência de fator. São apresentadas também as telas referentes aos diagramas.

FIGURA 66. CADASTRO E ALTERAÇÃO DE REGISTRO. REFERENTE AOS DIAGRAMAS 67 E 68.



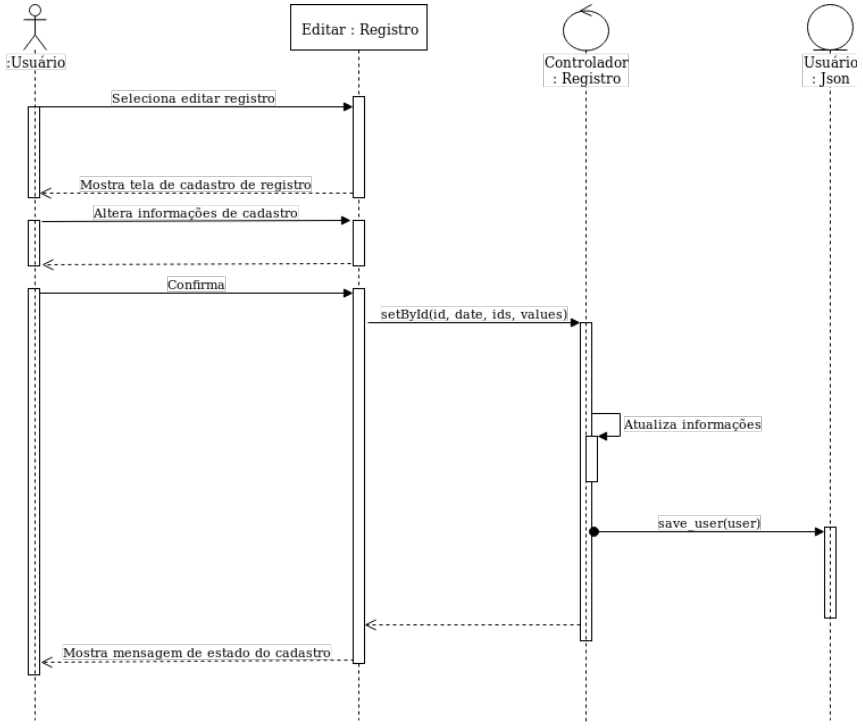
Fonte. O AUTOR (2017).

FIGURA 67. DIAGRAMA DE SEQUÊNCIA DE CADASTRO DE REGISTRO.



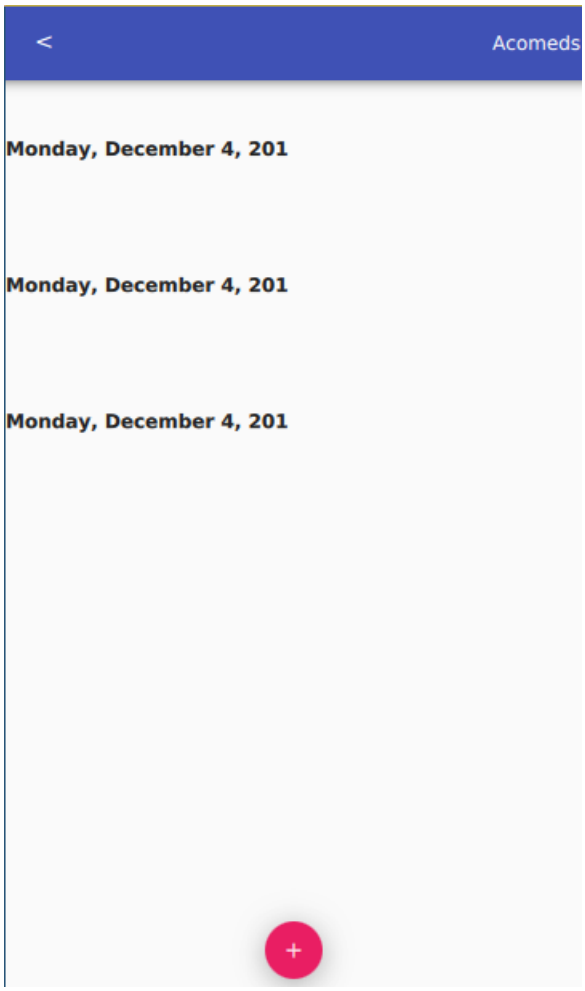
Fonte. O AUTOR (2017).

FIGURA 68. DIAGRAMA DE SEQUÊNCIA DE EDIÇÃO DE REGISTRO.



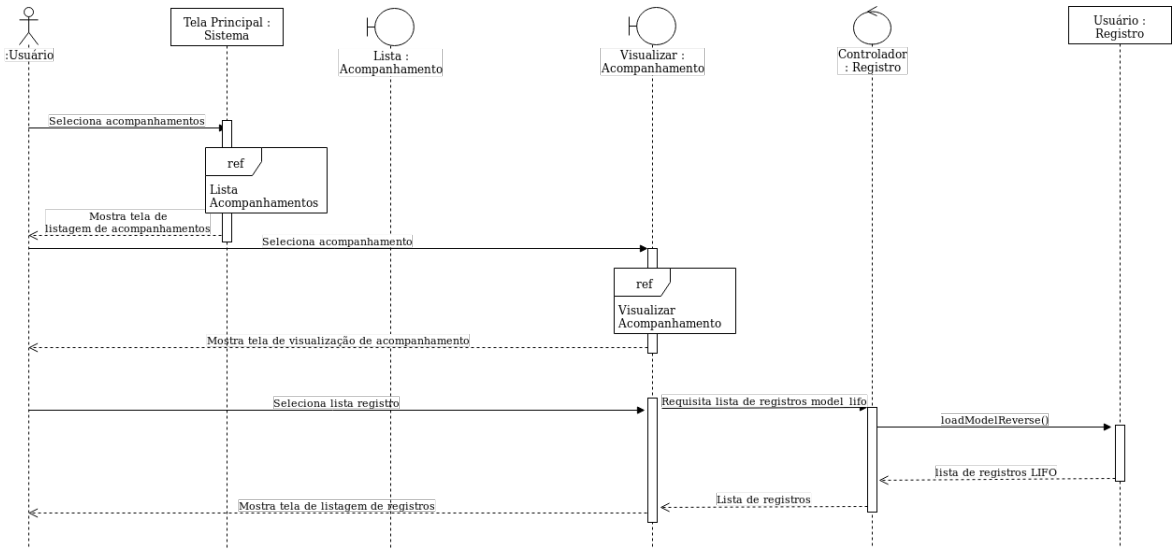
Fonte. O AUTOR (2017).

FIGURA 69. LISTA DE REGISTROS CADASTRADOS. REFERENTE AO DIAGRAMA 70.



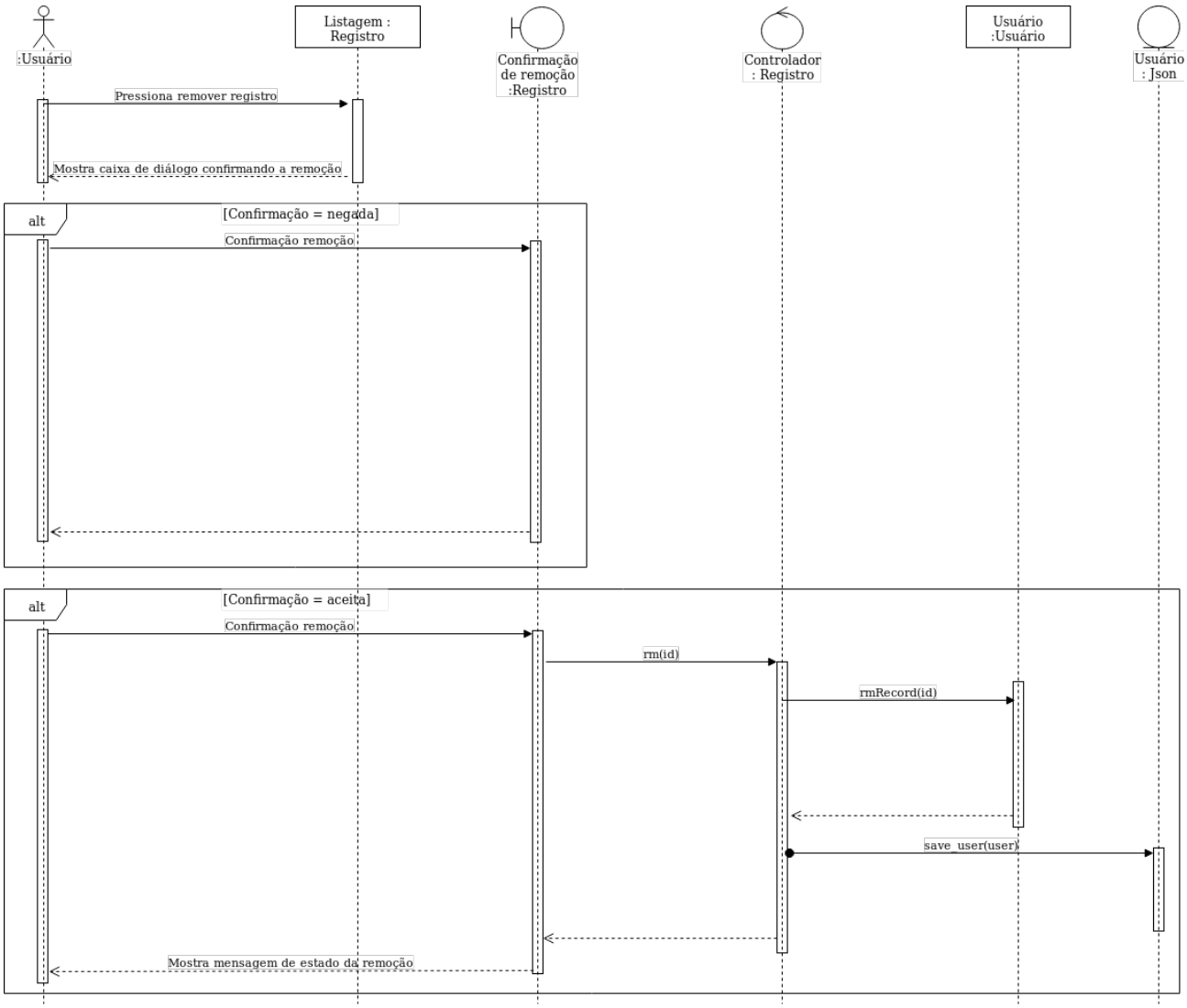
Fonte. O AUTOR (2017).

FIGURA 70. DIAGRAMA DE SEQUÊNCIA DE LISTAGEM DE REGISTRO.



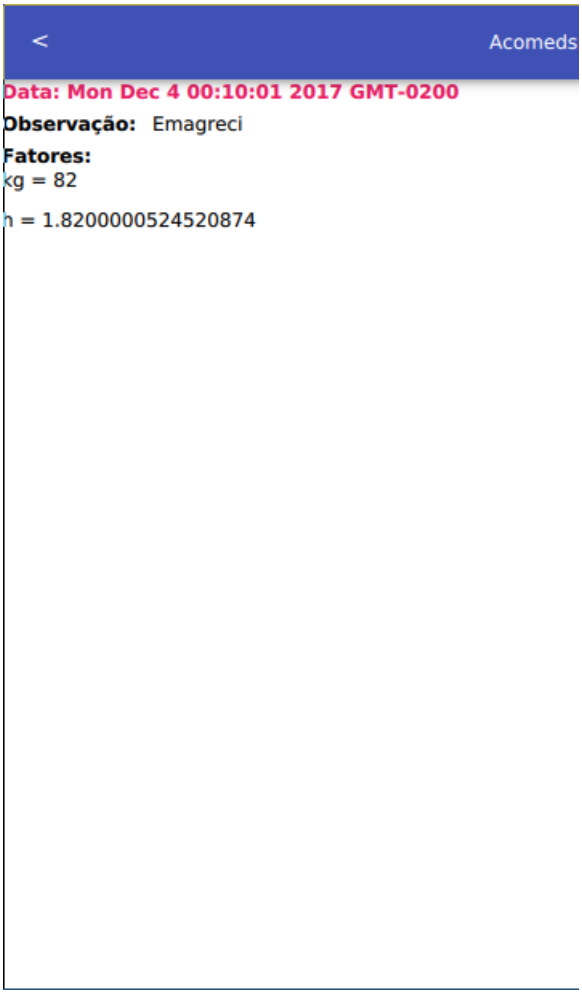
Fonte. O AUTOR (2017).

FIGURA 71. DIAGRAMA DE SEQUÊNCIA DE REMOÇÃO DE REGISTRO.



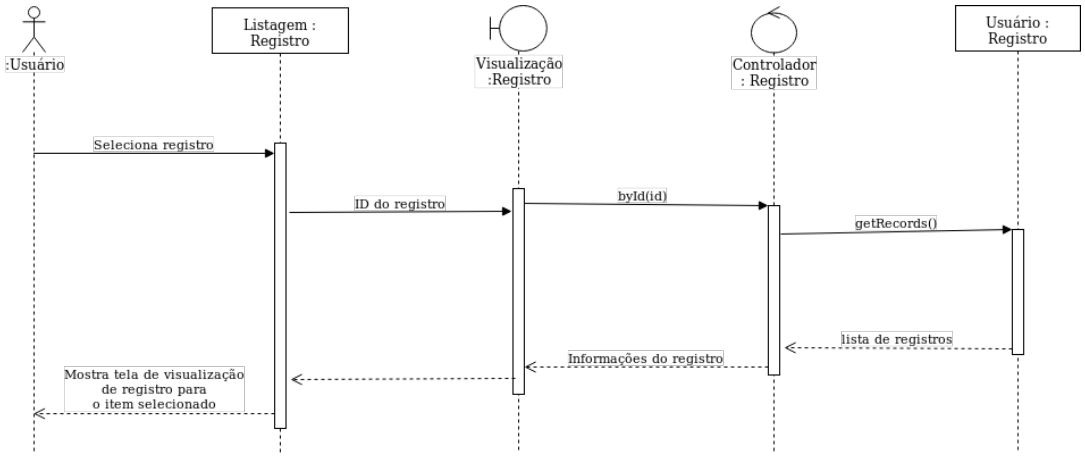
Fonte. O AUTOR (2017).

FIGURA 72. VISUALIZAÇÃO DE REGISTRO. REFERENTE AO DIAGRAMA 73.



Fonte. O AUTOR (2017).

FIGURA 73. DIAGRAMA DE SEQUÊNCIA DE VISUALIZAÇÃO DE REGISTRO.



Fonte. O AUTOR (2017).

APÊNDICE F – DESCRIÇÕES DE *SPRINTS*

F.1 SPRINT 1

Nessa sprint foi decidida a direção inicial referente às funcionalidades do aplicativo e estudo das ferramentas necessárias que seriam utilizadas para a programação do aplicativo. Foi feito o comparativo entre as opções de tecnologias disponíveis e averiguação dos prós e contras das ferramentas analisadas.

Obteve-se a conclusão de que deveriam ser utilizados: Qt Quick e o módulo Qt charts para representação de gráficos lineares para o módulo de acompanhamento de indicadores de saúde. A ferramenta Qt quick se mostrou a mais vantajosa e funcional para o projeto. Também foram sumarizados todos os requisitos com o cliente aonde encontramos o necessário para o funcionamento básico do aplicativo.

Foi realizado um protótipo à mão das telas.

E iniciou-se um *Proof of Concept* (POC) (*Proof of Concept*), um mini projeto feito com o intuito de validar se a tecnologia atende aos requisitos do projeto e se a ferramenta deveria ser substituída por outra.

Dentre os temas de estudo específicos abordados, os que foram mais aprofundados referente ao Qt foram: o funcionamento do QML e seus componentes, bem como o funcionamento padrão de aplicativos para mobile.

F.2 SPRINT 2

Finalizamos o POC usando Qml para desenvolvimento do aplicativo e até então não encontramos problemas. O módulo que disponibiliza o padrão de design Material Design ainda não estava disponível e portanto iniciamos o desenvolvimento de componentes que atendessem os requisitos do guideline da Google para que mesmo se na finalização do projeto não houvesse ainda a disponibilidade do módulo, o aplicativo pudesse ter a aparência de um aplicativo nativo para Android.

Usando o formato de manipulação de dados JsonCpp, configurou-se um exemplo com a finalidade de definir se seria viável ou não a utilização da biblioteca.

Foi definido no trello um backlog, que nos traz uma visão geral dos requisitos do produto final foi criado o backlog contendo as funcionalidades esperadas para o usuário

A POC resumiu-se em uma lista demonstrativa de medicamentos e acompanhamentos que poderiam ser acessadas através de uma tela Home

Nessa sprint iniciamos a produção dos protótipos e das telas protótipo do aplicativo iniciamos com o protótipo em QML que visava exemplificar as funcionalidades do aplicativo. já nas telas protótipo iniciamos a produção das telas de login, home, medicamentos, about e gráfico de indicadores.

F.3 SPRINT 3

Nessa sprint iniciou-se a produção dos protótipos e das telas do aplicativo ao mesmo tempo que iniciamos com o aprendizado em QML, formando assim um protótipo de médio nível que visava exemplificar as funcionalidades do aplicativo.

O protótipo de nível baixo consiste em um tipo de protótipo sem muito detalhamento visando os principais elementos esperados para o produto final. O protótipo de nível médio já começa a incluir alguns elementos a mais comparado ao de baixo nível enquanto que o protótipo de nível alto possui um rico ou total detalhamento das interfaces da aplicação podendo inclusive ser funcional (sem a implementação da regra de negócios em si).

Nas telas de baixo nível foi elaborado o fluxo principal do programa : cadastro, edição, visualização e remoção de medicamentos e acompanhamentos.

Já nas telas de médio nível iniciamos a produção das telas de login, home, visualização de medicamentos, about e gráfico de indicadores

F.4 SPRINT 4

Na quarta Sprint continuou-se o estudo da linguagem QML para aplicação dos conceitos estabelecidos no protótipo de baixo nível. Além disso, estudos e testes foram realizados usando o framework próprio para testes unitários do Qt e também foi testada a ferramenta de testes da biblioteca Boost.

Embora fosse prático aplicar testes usando o Qt, priorizou-se o uso da biblioteca Boost por ser mais difundido, por ter um melhor tratamento de exceções no código e por ter melhores mensagens de erro.

F.5 SPRINT 5

Na quinta Sprint a implementação das classes foi iniciada. Baseando-se na estrutura de dados definida, a classe de medicamentos foi formulada e testes unitários foram implementados para a mesma.

F.6 SPRINT 6

A sexta Sprint foi direcionada para a comunicação entre o código escrito em C++ com o código escrito em QML.

Foi definida uma maneira padrão de comunicação entre as linguagens, que consiste basicamente na formação de uma classe intermediária em C++ que herda da classe QObject do Qt, exercendo assim o papel de Controller de uma aplicação que se baseia no padrão MVC .

Houve a necessidade de migração da biblioteca JsonCpp para a biblioteca Ptree do conjunto de bibliotecas Boost. Os motivos principais para isso foram problemas encontrados na manipulação de tipos numéricos e facilidade de uso com o pré-compilador da IDE QtCreator.

A classe de Medicamentos foi readequada para utilizar a biblioteca Ptree.

Quanto ao QML : iniciou-se a criação da tela de cadastro de medicamentos, o protótipo para anexar uma receita foi reformulado e sombras foram implementadas seguindo os padrões do Material Design.

F.7 SPRINT 7

Na sétima Sprint foi definida a maneira como a sessão seria manipulada pelo sistema. Além disso foi finalizado a implementação do cadastro de medicamentos e mais elementos baseados no Material Design foram implementados.

F.8 SPRINT 8

Na oitava Sprint foram implementados:

- O carregamento de um medicamento do arquivo Json para o programa.
- A visualização básica de medicamentos.
- Elementos que faltavam no formulário de cadastro de medicamento.

Além disso, problemas encontrados foram corrigidos.

F.9 SPRINT 9

Na nona Sprint tornou-se necessário compilar o conjunto de bibliotecas Boost para a arquitetura ARM para que então fosse possível iniciar os testes do programa no sistema

Android.

Além da compilação :

Foi iniciada a implementação da classe de Indicadores.

Possibilitou-se o registro de uma foto da receita sem que o formulário fosse perdido.

Foi incluído uma maneira de selecionar datas.

A tela de medicamentos foi reformulada afim de servir de referência para a tela de indicadores.

F.10 SPRINT 10

Na décima Sprint tornou-se estável uma nova versão do framework Qt (5.8) que incluía elementos baseados no padrão de design Material Design da Google e também o padrão Universal da Microsoft. Com isso, estudos foram realizados quanto a utilização desses elementos e concluiu-se que seria muito mais eficaz usar os elementos disponibilizados pelo framework ao invés de utilizar os elementos criados manualmente.

Os códigos criados manualmente visando empregar o conceito do Material Design foram removidos para dar lugar aos elementos já disponíveis na ferramenta.

O calendário de seleção de datas foi corrigido e finalizado.

F.11 SPRINT 11

Na décima primeira Sprint foram criadas as telas de Indicadores, o desenvolvimento das mesmas foi muito acelerado uma vez que foram inteiramente baseadas nas telas de Medicamentos, provando que a estratégia de criação de criação de elementos robustos e versáteis para as telas de Medicamentos tornou-se útil. As listas de visualização de Medicamentos e Indicadores foram sincronizadas com as ações executadas no aplicativo (CRUD).

F.12 SPRINT 12

Nesta Sprint iniciou-se a revisão de elementos da documentação do sistema e identificou-se que não haveria um meio único e rápido para enviar notificações locais para o sistema operacional com códigos independentes de plataformas.

Testes foram realizados usando classes de notificação para o systray disponíveis no Qt. Foi incluído no sistema, porém, o visual não era satisfatório e as notificações não funcionariam corretamente para Android.

F.13 SPRINT 13

Nessa Sprint a documentação continuou a ser desenvolvida e além disso o projeto foi habilitado para compilação usando a ferramenta CMake com o gerenciador Hunter.

Com a compilação usando CMake tornou-se muito mais prático usar a biblioteca Boost pois é possível somente incluir a biblioteca como dependência para que o gerenciador compilasse as bibliotecas Boost necessárias.

Além disso, foi incluída a biblioteca SnoreNotify para que notificações Desktop nativas pudessem ser visualizadas, tornando a experiência do usuário desktop melhor.

F.14 SPRINT 14

O foco da décima quarta Sprint foi a revisão da documentação textual e pesquisa de meios para notificar dispositivos mobile a partir do aplicativo.

Infelizmente concluiu-se que dispositivos móveis necessitam de implementações específicas pois não utilizam meios comuns para gerenciar suas notificações. E então a implementação para notificação Android foi iniciada.

No momento não é possível testar o sistema em sistemas da Apple (MacOS e IOS) por indisponibilidade de dispositivos e portanto, as notificações do sistema são esperadas para notificar somente o sistema operacional MacOS entre esses dois, uma vez que a biblioteca utilizada não se propõe a notificar nativamente um dispositivo que usa o sistema IOS.

F.15 SPRINT 15

Essa Sprint foi focada totalmente para a finalização da documentação, implementaçãod e notificação para android e entrega do projeto.