RECA
REVISTA ELETRÔNICA DE COMPUTAÇÃO APLICADA

Departamento de Computação
Uni-FACEF

PRATELEIRA: desenvolvimento de uma solução mobile para controle de medicamentos e rotinas medicinais

Wilson Donisette Guiraldelli Filho
Graduando em Engenharia de Software – Uni-FACEF
will.guiraldelli@outlook.com

José Higor Ferreira de Oliveira
Graduando em Engenharia de Software – Uni-FACEF
jhigorfoliveira@gmail.com

Orientador: Prof. Me. Carlos Alberto Lucas
Mestre em Educação – Uni-FACEF
projetos@profcarloslucas.com.br

Resumo
Dispositivos móveis, como smartphones e smartwatches entraram de vez na vida do ser humano na última década, aparelhos de tamanhos variados que carregam em si informações importantes para o cotidiano de cada indivíduo. Dentre essas informações, queremos destacar a saúde dos usuários e pessoas próximas a eles; em um mundo globalizado como o que vivemos hoje, muitas vezes alguém longe em distância se mantém próximo a partir da tecnologia, e por que não manter a saúde destes entes queridos, na palma da mão? Com este questionamento e a facilidade que os dispositivos nos permitem de carregar informações e serem programados para nos auxiliar com tarefas do cotidiano, buscamos apresentar, neste artigo, o desenvolvimento do Prateleira, um aplicativo para gerenciamento de estoque compartilhado de medicamentos e programação de rotinas de medicação, apresentamos artefatos, documentos, diagramas e resultados produzidos nas fases de Engenharia de Software e Desenvolvimento dos módulos de Autenticação, Prateleira, Categorias e Medicamentos.

Palavras-chave: Desenvolvimento. *Software.* Prateleira. Medicamento. Compartilhamento. Rotina.

Abstract
Mobile devices, like smartphones and smartwatches, became once for all parts of humankind's lives in the last decade. These devices, of different sizes, contain important information about daily life of each individual. Among this information, we would like to emphasize the health of the users and close people; in a globalized world from nowadays, so many times someone far in distance is close because of technology, so, why not bring the health data from those closer, in our hands? Given this questinament and the facility that the mobile devices provide us to be programmed for helping our life, we introduced in this article, the development of Prateleria, a mobile device app for management of shared store of medicines and scheduling routines of medication, was shown artifacts, documents, diagrams and produced results from the Software Engineering and Development phases for the modules Authentication, Prateleira, Categories and Medicines.

Keywords: Development. Software. Prateleira. Medicine. Sharing. Routine

1 Introdução

Com o constante avanço da medicina e da tecnologia, ambas têm promovido melhores condições e qualidade de vida para as pessoas, e, assim, descobertas de tratamentos, identificação de doenças são feitas a cada dia. Geralmente, tratamentos ou administração de medicamentos devem ser acompanhados de maneira correta e exata, para assim, garantir uma boa qualidade e um resultado satisfatório, e o uso incorreto de medicamentos, além de não fornecer efeito ao paciente, pode trazer graves prejuízos à saúde.

É fato que a utilização e uso de *smartphones* na vida das pessoas cresce cada dia mais, e em uma larga escala, nota-se que o uso dos *smartphones* está presente em todas as faixas etárias.

Juntando os recursos acima, a proposta deste trabalho é a criação de um aplicativo *mobile* que sirva de utilidade e ferramenta para pacientes, profissionais e usuários que fazem o uso de medicamentos e tratamentos medicinais.

O aplicativo Prateleira foi pensado e desenvolvido baseado na necessidade de uso de medicamentos, tanto para pessoas que fazem uso de múltiplos medicamentos dependentes do tratamento por uma doença crônica, como diabetes, por exemplo, e que faz parte de sua rotina, tanto para aqueles que fazem a utilização temporária. No fim das contas, não aderir ao tratamento, seja ele contínuo ou temporário, acaba gerando gastos e desperdícios.

De tal modo, o aplicativo tem como objetivo facilitar, organizar e monitorar a rotina medicinal de cada paciente, auxiliando nos horários de administração através de notificações, juntamente com o gerenciamento de estoque domiciliar, que visa controlar a quantidade de medicamentos de acordo com o uso informado e as datas de vencimento dos medicamentos além de possibilitar o acompanhamento por um responsável, através da funcionalidade de compartilhamento de prateleiras entre usuários.

Os procedimentos metodológicos adotados para o desenvolvimento foram: revisão bibliográfica de tópicos referentes ao tema, com estudos realizados em livros e artigos científicos; levantamento dos requisitos e funcionalidades com as partes interessadas e *stakeholders* da área da saúde; gerência através da metodologia *Scrum* para dividir as tarefas e organizar as entregas, construção dos diagramas de BPMN, Caso de Uso, de Classe UML, desenvolvimento da aplicação utilizando a linguagem *React Native*, *Node,js* juntamente com o SGBD (Sistema de Gerenciamento de Banco de Dados) *PostgreSQL.*

O atual projeto é apresentado em sessões, abordando os seguintes tópicos: referencial teórico, contendo a conceituação e definição dos componentes que foram utilizados no desenvolvimento do projeto; uma seção destinada aos fundamentos do curso de Engenharia de *Software* e outra contendo um aspecto do projeto, como a Qualidade de *Software*, as demais contém os itens da solução proposta; a apresentação dos resultados obtidos; e, por fim, as considerações finais sobre a realização do projeto.

A documentação completa de todos artefatos apresentados neste antigo, podem ser encontrado no GitHub.

2 Referencial teórico

Nesta seção do trabalho serão abordados temas e ferramentas de desenvolvimento do projeto, tais como Engenharia de *Software* e suas características, Qualidade de *software* e os seus respectivos aspectos e a importância dentro do contexto global de um *software*, como a tecnologia pode auxiliar na adesão de medicamentos, a importância da organização de medicamentos, e a relação das ferramentas como *React Native*, *NodeJS* e o sistema de gerenciamento de banco de dados *PostgreSQL* com o desenvolvimento.

2.1 Engenharia de *Software* – Fundamentos

Engenharia de *software* é a matéria da engenharia cujo objetivo está em todos os artefatos, aspectos da produção e desenvolvimento de um *software*, desde a parte inicial, da especificação do *software* e dos requisitos, até o momento da usabilidade e manutenção do *software*, momento que o *software* já está em produção com o cliente, como aponta Sommerville (2011). Além disso, esta disciplina tem como foco apoiar o desenvolvimento de *software*, mais do que a programação individual. Ela inclui técnicas que apoiam a especificação do projeto, documentação e evolução de *softwares*.

"A engenharia de *software* pode se caracterizar por um desenvolvimento de *software* prático, ordenado e medido para produzir sistemas satisfatórios aos usuários e que respeitem prazos e orçamentos" (PETERS; PEDRYCZ apud REZENDE, 2001, p.3).

No momento em que falamos sobre engenharia de *software*, não pensamos apenas no programa em si, mas em todos os processos técnicos do desenvolvimento e as ferramentas, métodos e teorias para apoiar o desenvolvimento do projeto. Os principais itens da disciplina são o gerenciamento do projeto e documentação, meios necessários para fazer o programa funcionar corretamente e atender a todos os requisitos esperados do cliente. Para isto, é feita uma abordagem sistêmica para o desenvolvimento de *software*, analisando as opções de custo, prazo e confiança, assim como a necessidade do sistema e a necessidade do cliente, ainda segundo Sommerville (2011).

2.2 Engenharia de *Software* - Qualidade de *Software*

A qualidade de um *software* é um processo que abrange a estruturação, sistematização e a realização de atividades que buscam alcançar o desempenho, qualidade e concordância com cada fase de desenvolvimento, visando atender todos os padrões e requisitos definidos em cada processo. Estão envolvidos os testes de verificação, testes de validação de *softwares* em todas as partes do ciclo de desenvolvimento, segundo Bartié (2002).

Para chegar a uma boa qualidade no desenvolvimento de um *software* e em todos os artefatos a ele relacionados, como documentos, diagrama etc, faz-se necessário não apenas atender aos requisitos funcionais, mas também a outros itens qualificados como requisitos não funcionais, como por exemplo: extensibilidade, capacidade de suporte ao *software* (manutenção), reuso do código, desempenho, escalabilidade, usabilidade e confiabilidade nas informações, elementos apresentados pelo *software,* como aponta Engholm Jr. (2010).

De acordo com um especialista da área, comenta Pressman e Maxim (2016, p. 414), “No sentido mais geral, a qualidade de *software* pode ser definida como: uma gestão de qualidade efetiva aplicada de modo a criar um produto útil que forneça valor mensurável para aqueles que o produzem e para aqueles que o utilizam”.

Um dos itens e artefatos que podem realizar a validação da qualidade de um *software* são os aspectos passados pela ISO.

> Vários significados têm sido dados no sentido de prover um entendimento adequado do termo “qualidade” no contexto da produtividade. Atendimento às expectativas do cliente, conformidade com a especificação, conhecimento do processo para melhorá-lo, efetividade e usabilidade. Esses aspectos poderiam ser consolidados na definição dada pela *International Organization* for *Standardization* (ISO), segundo a qual qualidade é “a totalidade das características de uma entidade que lhe confere a capacidade de satisfazer às necessidades explícitas e implícitas” (NBR ISO 8402) (VASCONCELOS *et.al*, 2006, p. 73).

A Norma NBR ISO/IEC 9126-1:2003 define *qualidade de software* como “a totalidade de características de um produto de *software* que lhe confere a capacidade de satisfazer necessidades explícitas e implícitas”. Um novo modelo de qualidade foi criado em 2011, e que segundo Wazlawick (2013, p. 229) "uma das motivações para a criação de uma nova norma está no fato de que as antigas aplicavam-se apenas ao processo de desenvolvimento e uso do produto de *software*, mas pouco tinham a dizer em relação à definição do produto".

Este modelo, denominado de ISO/IEC 25010, ainda de acordo com Wazlawick (2013), define um conjunto de oito características internas e externas de produto de *software*, subdivididas em subcaracterísticas e mais cinco características de *software* em uso, algumas das quais também são subdivididas em subcaracterísticas. Vide imagem ilustrada abaixo

Figura 1 - Modelo de qualidade da ISO 25010:2011

Tipo	Características	Subcaracterísticas	Termo em inglês
Caracteristicas do produto	Adequação funcional (*functional suitability*)	Completude funcional Corretude funcional (acurácia) Funcionalidade apropriada	*Functional completeness* *Functional correctness (accuracy)* *Functional appropriateness*
	Confiabilidade (*reliability*)	Maturidade Disponibilidade Tolerância a falhas Recuperabilidade	*Maturity* *Availability* *Fault tolerance* *Recoverability*
	Usabilidade (*usability*)	Apropriação reconhecível Inteligibilidade Operabilidade Proteção contra erro de usuário Estética de interface com usuário Acessibilidade	*Appropriateness recognisability* *Learnability* *Operability* *User error protection* *User interface aesthetics* *Accessibility*
	Eficiência de desempenho (*performance efficiency*)	Comportamento em relação ao tempo Utilização de recursos Capacidade	*Time behavior* *Resource utilization* *Capacity*
	Segurança (*security*)	Confidencialidade Integridade Não repúdio Rastreabilidade de uso Autenticidade	*Confidentiality* *Integrity* *Non-repudiation* *Accountability* *Authenticity*
	Compatibilidade (*compatibility*)	Coexistência Interoperabilidade	*Co-existence* *Interoperability*
	Capacidade de manutenção (*maintainability*)	Modularidade Reusabilidade Analisabilidade Modificabilidade Testabilidade	*Modularity* *Reusability* *Analyzability* *Modifiability* *Testability*
	Portabilidade (*portability*)	Adaptabilidade Instalabilidade Substituibilidade	*Adaptability* *Instalability* *Replaceability*
Características do uso	Efetividade (*effectiveness*)	Efetividade	*Effectiveness*
	Eficiência (*efficiency*)	Eficiência	*Efficiency*
	Satisfação (*satisfaction*)	Utilidade Prazer Conforto Confiança	*Usefulness* *Pleasure* *Comfort* *Trust*
	Uso sem riscos (*freedom from risk*)	Mitigação de risco econômico Mitigação de risco a saúde e segurança Mitigação de risco ambiental	*Economic risk mitigation* *Health and safety risk mitigation* *Environmental risk mitigation*
	Cobertura de contexto (*context coverage*)	Completude de contexto Flexibilidade	*Context completeness* *Flexibility*

Fonte: Wazlawick (2013).

2.3 Como a tecnologia pode auxiliar a adesão de medicamentos

A medicina alterou para sempre nossa habilidade de viver com doenças e elevou nossa expectativa de vida. No entanto, algumas vezes, os medicamentos causam grandes danos à saúde se tomados incorretamente, mal monitorados ou resultados de um erro, acidente ou problema de comunicação e, por isso, a tomada correta de medicamento, seja de forma temporária ou permanente, torna-se um compromisso, conforme a citação:

> Uma vez que a doença crônica exige um tratamento permanente, faz-se necessário que o indivíduo cultive hábitos e atitudes que promovam a consciência para o autocuidado. Portanto, aderir ao tratamento é imprescindível para o controle da condição crônica e o sucesso da terapia proposta. (TADDEO, 2012, p. 2926).

Em um estudo realizado com pacientes com câncer de mama, Paladino (2019) afirma que pacientes que não aderem a 100% de sua medicação AET,

ou descontinuam o tratamento, não terão todo o efeito intencionado pela receita, o que aumenta o risco de mortalidade. Esse é um exemplo entre tantos tratamentos que devem ser realizados de forma correta, cujo descumprimento pode causar graves danos.

Com o intuito de prevenir e auxiliar na adesão de medicamento, o terceiro Desafio Global de Segurança do Paciente, com foco na 'Medicação sem danos' (ISMP, 2018), buscará, ao longo de cinco anos, reduzir em 50% os danos graves e evitáveis relacionados a medicamentos.

Erros com medicamentos podem ser grandemente reduzidos ou prevenidos, melhorando os sistemas e práticas de medicação

Com o intuito de implementar soluções e melhorias, os objetivos do Terceiro Desafio Global de segurança do Paciente, incluem:

- "Desenvolver guias, documentos, tecnologias e ferramentas para dar suporte à criação de sistemas de utilização de medicamentos mais seguros, que resultem na diminuição da ocorrência de erros de medicação" (ISMP, 2018, p. 3).
- "Engajar os principais envolvidos, parceiros e indústria para sensibilizá-los quanto aos problemas de segurança no uso de medicamentos, levando-os a atuar ativamente em busca de formas de reduzir os problemas relacionados a medicamentos" (ISMP, 2018, p. 3).

Com estes objetivos e o questionamento de como a tecnologia e os sistemas de saúde podem auxiliar na adesão de medicamentos, além dos estudos de Paladino (2019), que relata os resultados positivos de uma pesquisa quantitativa de pacientes com câncer de mama que receberam notificações sobre o câncer de mama semanalmente, buscaremos implementar uma solução que cumpra com os objetivos de engajar e contribuir para adesão correta de tratamentos medicinais.

2.4 A importância da organização de medicamentos.

Diversas pesquisas quantitativas realizadas em pequenos ou grandes municípios evidenciam o uso da prática do armazenamento, que não é proibida, mas deve ser realizada de forma correta.

> Há várias causas para a sobra de medicamentos, como a dispensação em quantidade além da necessária para o tratamento, amostras-grátis distribuídas pelos laboratórios farmacêuticos como forma de propaganda e o gerenciamento inadequado por parte de farmácias e demais estabelecimentos de saúde (Walter da Silva Jorge João, Vice-Presidente do CFF) (JOÃO, 2011, p. 15).

De acordo com (FERNANDES e PETROVICK apud SCHWINGEL, 2004) a má orientação ao uso do medicamento pode ocasionar no acúmulo, o que está diretamente ligado ao uso incorreto do mesmo. Ainda com (FERNANDES e PETROVICK apud SCHWINGEL, 2004) o armazenamento incorreto e inadequado desses medicamentos pode gerar grande risco a saúde.

2.5 *React Native*

Segundo Danielsson (2016), o propósito do *React Native* é simples, um desenvolvedor não deveria gastar um tempo precioso para criar a mesma aplicação duas vezes para IOS e *Android*, dado que um dos principais aspectos divergentes entre as plataformas é o *layout*. Então, a ideia foi criar uma única linguagem que usa um *layout* e componentes nativos de acordo com a plataforma em questão. Ainda de acordo com Danielsson (2016), o *React Native* é um *framework open source*, baseado no *ReactJS do* Facebook Inc que é um *framework JavaScript*.

O *react native* comunica-se com as APIs nativas através de uma 'ponte' que possui o *JavaScriptCore* 10 como um interpretador. A ponte *JavaScript* conecta o lado *JavaScript* ao lado nativo da aplicação. O lado *JavaScript* roda em uma *thread* assíncrona separada da *thread* principal, o que não interfere na UI nativa. Quando uma chamada é realizada pelo *JavaScript*, uma mensagem é armazenada na fila se ela não puder ser enviada imediatamente. Antes de enviar a informação para o lado nativo, a ponte *JavaScript* irá converter os tipos de dados *JavaScript* para os tipos de dados nativos automaticamente. Um componente nativo roda a 60 *frames* por segundo, o que a ponte *javascript* também faz, o que maximiza a performance do *React Native*. (HANSSON; VIDHALL, 2016)

2.6 *Node.js*

Ryan Dahl introduziu o *Node.js* em 2009, e o interesse na utilização do *JavaScript* no servidor aumentou de maneira irrecuperável, tanto nos negócios quanto na ciência. O *Node.js*, que está sendo executado na implementação *JavaScript* V8 do Google, introduziu o conceito de um modelo de eventos de *I/O(Input/Output)* sem bloqueio, trabalhando em um único *thread* em vez de usar vários *threads*, como é conhecido nos servidores da web tradicionais. Esse *loop* de eventos de *thread* único leva a um consumo aprimorado de recursos de *hardware* e a um número significativamente maior possível de conexões simultâneas de clientes, tornando o *Node.js* uma alternativa considerável para o desenvolvimento de aplicativos Web do lado do servidor (KAIMER; BRUNE, 2018).

Segundo Pereira (2014), o *event-loop* tem o papel de interpretar os eventos no sistema, em outras palavras é um *loop* infinito, que faz a verificação dos eventos na fila e realiza a validação dos que já foram emitidos em cada interação. Quando algum evento está em desenvolvimento, podemos programar qualquer lógica dentro dele, quando é emitido um evento, o *event-loop* executa e transfere para a fila dos executados, tudo isso devido ao mecanismo de função *callback* do *JavaScript.*

2.7 *PostgreSQL*

De acordo com Momjian (2001), os bancos de dados são usados principalmente para armazenamento e recuperação de dados. Pode-se usar um processador de texto ou planilha para armazenar pequenas quantidades de dados. No entanto, para grandes volumes ou aqueles que precisam ser recuperados e atualizados com frequência, os bancos de dados são a melhor opção, permitindo armazenamento ordenado, recuperação rápida e análise complexa.

O *PostgreSQL*, uma ferramenta *Open source*, tem suporte às operações ACID (Atomicidade, Consistência, Isolamento, Durabilidade). Cada uma dessas operações valida a qualidade dos serviços realizado pelo banco de dados. Ainda, o *PostgreSQL* disponibiliza extensão para uso de algoritmos de criptografia,

como: SHA1, MD5, itens que garantem a segurança dos dados e a navegabilidade sigilosa no sistema, como observa Milani (2008).

> [...] O *PostgreSQL* é um Sistema Gerenciador de Base de Dados Relacional (SGBDR) de alta performance, de fácil administração e utilização em projetos. Esse SGBDR permite a utilização da linguagem SQL, *triggers* (disparadores) e tantos outros recursos presentes nos mais famosos sistemas SGBDR do mercado, como *Oracle, InterBase, SQL Server, MySQL*, etc.[...]Todos esses recursos proporcionam ao programador e ao administrador de banco de dados realizar suas tarefas e atender suas perspectivas mais específicas[...] (SOUZA, AMARAL e LIZARDO, 2011, p. 4).

3 Metodologia de desenvolvimento e planejamento

Para metodologia de desenvolvimento, foram definidos o *Scrum* e *Kanban*, sendo o *Scrum* um *framework* para planejamento e execução. Assim, o projeto tem seus entregáveis sendo divididos em ciclos, chamados *sprints*, sendo o tempo deste ciclo definido de acordo com necessidade da equipe.

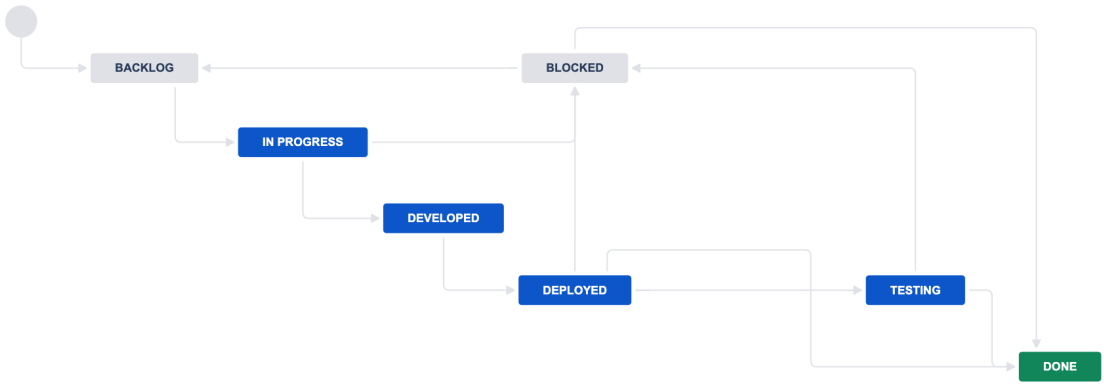
O *Kanban*, metodologia criada pela Toyota na década de 60, visa estruturar de maneira organizada em um *board* as tarefas prioritárias a serem realizadas pela equipe, essas tarefas sendo descritas e adicionas ao *board* e sendo atualizada de acordo com um fluxo de trabalho.

A fim de extrair o melhor de cada, o projeto em questão foi dividido em *Sprints*, sendo seus entregáveis/tarefas sendo controlados via *Kanban*, para auxílio do controle usamos da ferramenta Jira.

3.1 Fluxo de trabalho

O fluxo se inicia no *backlog*, passando para as respectivas fases no diagrama abaixo, o status *Blocked* é utilizado quando uma tarefa possui um problema, sendo ele podendo ser reintroduzido ao desenvolvimento ou concluído quando descartado. Alguns itens não necessitam do estágio de teste, portanto, a possibilidade de serem concluído diretamente.

Figura 2 - Diagrama de fluxo de trabalho



Fonte: Os autores (2020)

3.2 Planejamento em *Sprint*

As *Sprints* foram mapeadas por mês, por incluírem itens tanto de artefatos de engenharia de *software* a serem realizados, quanto a desenvolvimento de código do aplicativo. Conforme exemplo abaixo, do planejamento do mês de março.

Artefatos

- Definição do problema do projeto
- Definição da justificativa e motivação do trabalho
- Definição dos objetivos gerais específicos do projeto.
- Definição dos procedimentos metodológicos do trabalho
- Definição da estrutura do artigo
- Construção da revisão bibliográfica
- Levantamento de requisitos
- Reuniões com as partes interessadas e profissionais da área
- Validação da ideia com os profissionais e partes interessadas

Desenvolvimento

- RF-001 - Cadastrar usuário
- RF-002 - Realizar Autenticação
- RF-003 - Recuperação de senha
- RF-018 - Disponibilizar página de perfil / editar perfil
- RF-019 - Alterar senha
- RF-020 - *Logout*

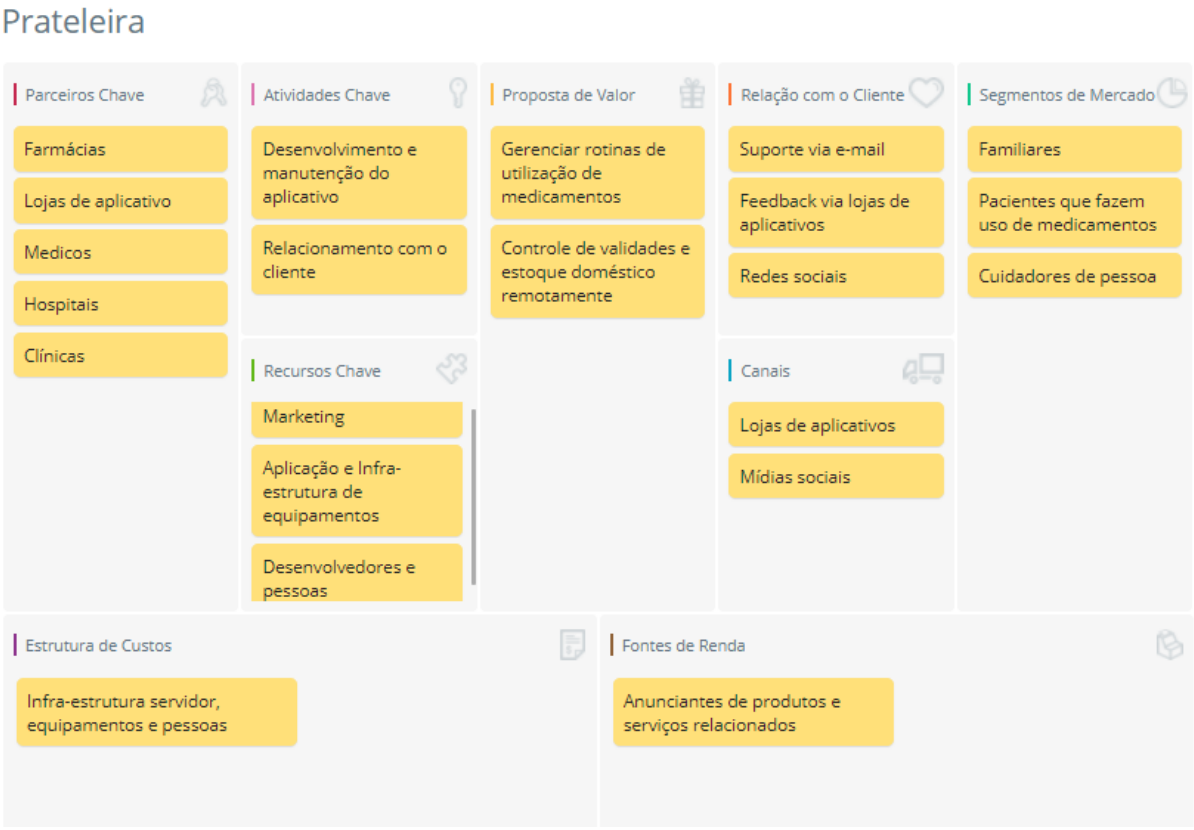
4 Canvas (*Business Model Canvas*)

O *Business Model Canvas* conhecido como Canvas é uma solução empreendedora para planejamento e construção de um modelo estratégico, por este quadro visual é onde nós podemos construir um modelo de negócio de alguma startup.

É possível definir as atividades necessários para produzir valor ao seu público-alvo/clientes, juntamente com a definição dos 9 elementos em blocos do quadro de negócio e o raciocínio da viabilidade desta startup.

Abaixo está disponibilizado o quadro empreendedor do modelo de negócio do Prateleira, com a estratégia inicialmente de adesão de usuários através da proposta de valor de gerenciar rotinas de utilização de medicamentos com o controle de validades e controlar estoque doméstico remotamente para as famílias, pacientes que fazem uso de medicamentos e cuidadores de pessoas, visando facilitar a gestão de medicamentos, realizar a redução de gastos com medicamentos e o aprimoramento do tratamento medicinal. Buscamos fechar parcerias com hospitais, médicos, farmácias, clínicas e lojas de aplicativos.

Figura 3 - Canvas



Fonte: Os autores (2020)

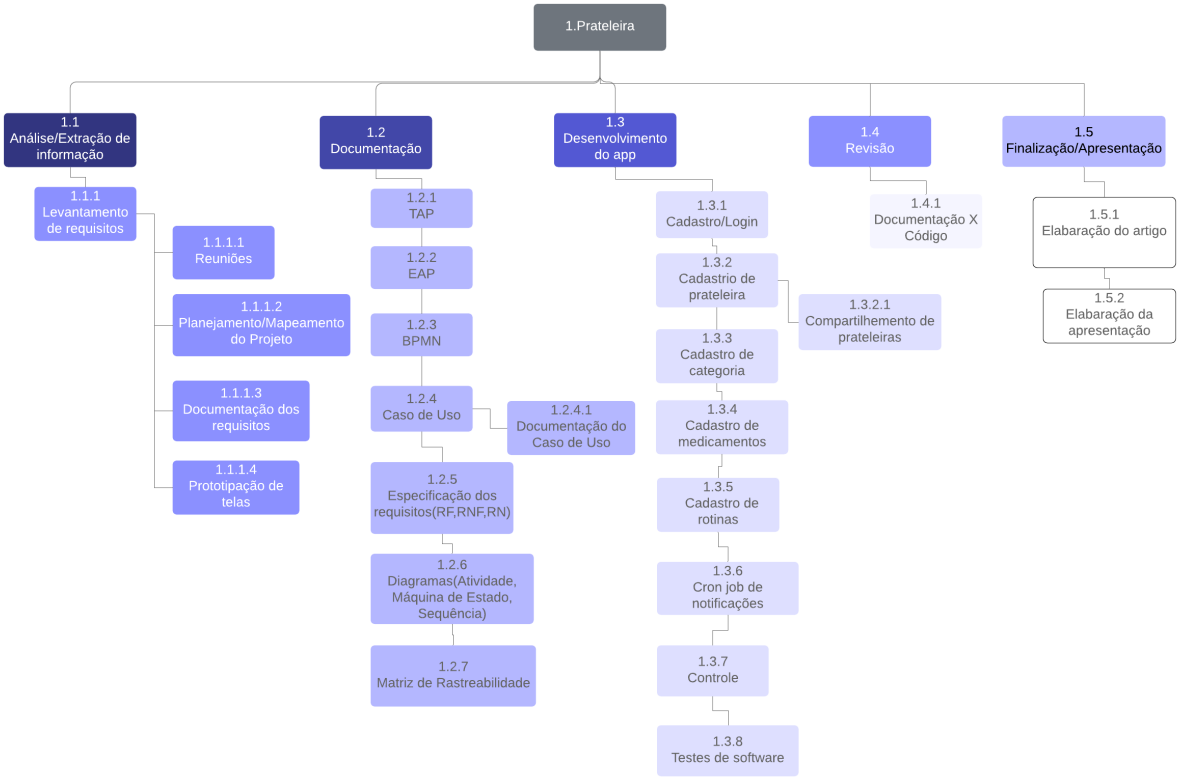
5 Discussão dos resultados

5.1 EAP (Estrutura Analítica do Projeto)

O EAP (Estrutura Analítica do Projeto) visa decompor o trabalho em partes menores e fazer a definição do escopo do projeto, e o objetivo primário é organizar o que deve ser feito para produzir as entregas do projeto.

O escopo foi definido pela equipe durante as reuniões de levantamento de requisitos. Os elementos a serem desenvolvidos nas fases de Análise/Extração de Informação, Documentação, Desenvolvimento da aplicação, Revisão e Finalização/Apresentação foram organizados conforme a imagem da estrutura abaixo:

Figura 4 - EAP



Fonte: Os autores (2020).

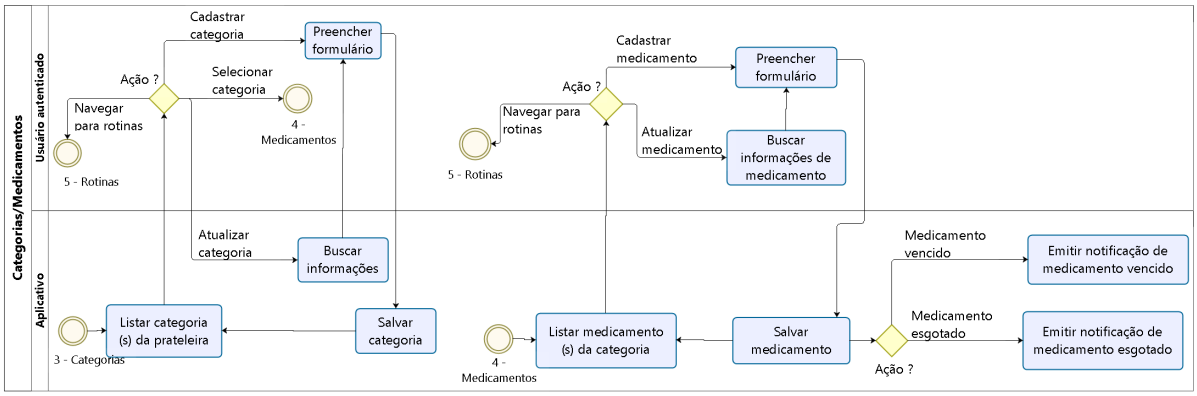
5.2 BPMN (*Business Process Model and Notation*)

O BPMN é um método de um processo de modelagem e notação de negócio, tendo como base o mapeamento visual de todos os processos a um determinado negócio, melhorando a comunicação entre processos e pessoas, a fim de ter um processo bastante eficiente.

A partir do levantamento de requisitos feito com as partes interessadas, utilizando dos artefatos gerados nas reuniões, foi feita uma análise e extração de informações, da qual identificamos os atores, condições, ações e atividades para execução dos processos do projeto.

Devido ao tamanho do BPMN do projeto e para uma melhor organização dos componentes, fizemos a divisão dos processos em módulos, abaixo está o processo do módulo de categorias/medicamentos, os demais módulos estão disponíveis no repositório do *GitHub*.

Figura 5 - BPMN - Processo categoria e medicamento



Fonte: Os autores (2020)

5.3 Documentação dos Requisitos

A documentação dos requisitos detalha todos os itens ou funcionalidades que os desenvolvedores devem implementar no projeto, tanto requisito funcional, não funcional e regras de negócios, neste documento é descrito as respectivas características com suas devidas especificações e restrições. O foco principal deste documento é detalhar as características essenciais das funcionalidades de cada requisito e que seja feito da forma correta.

Os requisitos funcionais são nada mais que funcionalidades do sistema, as funções que o sistema/aplicativo deve possuir para atender o negócio e suas regras.

Perante os requisitos não funcionais, podemos referenciar os mesmos de requisitos de qualidade, podem ser relacionados características de ambiente do sistema, aspectos como: segurança, disponibilidade, interface, usabilidade entre outros.

Sobre regras de negócio, podemos definir como restrições ou ações para alguns determinados processos do projeto, sendo políticas de um negócio, que visa satisfazer o foco do negócio, funcionando de maneira esperada, conforme as regras estabelecidas.

Abaixo, demonstraremos apenas alguns requisitos de todo o documento, contendo requisitos funcionais, requisitos não funcionais e regras de negócio, a documentação dos requisitos completa está disponível no *GitHub*.

Tabela 1- RF 002

RF 002 – *Realizar autenticação*	Categoria: (X)Oculto ( ) Evidente	Prioridade: (X) Altíssima ( ) Alta ( ) Média ( ) Baixa
Descrição: funcionalidade para inserção e validação de credenciais de acesso. Usuário deverá informar: - E-mail, campo para inserção de e-mail do usuário, o formulário deverá validar se o e-mail é válido (de acordo com o padrão xx@xx.xx, sendo x os valores do usuário), máximo de 255 caracteres, informação obrigatória - Senha, campo aberto para inserção de senha. Deve possui entrada segura de dados, ou seja, os valores digitados devem ser trocados por * para impedir visualização do conteúdo por terceiros, informação obrigatória Funcionalidade deve validar se usuário confirmou o e-mail enviado no cadastro Funcionalidade deve validar se usuário não está no processo de troca de senha Funcionalidade deve validar se senha informada está correta		

Fonte: Os autores (2020)

Tabela 2 - RF 011

RF 011 – *CRUD de Categoria*	Categoria: ( )Oculto (X) Evidente	Prioridade: (X) Altíssima ( ) Alta ( ) Média ( ) Baixa

Descrição: **Create**: Estando dentro de uma **Prateleira,** disponibilizar funcionalidade de cadastro de Categoria, informando:

- Nome, campo aberto para inserção de letras, máximo de 255 caracteres, informação obrigatória
- Cor, a mesma sendo gerada randomicamente pelo Aplicativo a princípio, com possibilidade de alteração.

Read: Listagem de todas as categorias da **Prateleira**

Update: Funcionalidade para editar as informações de uma categoria de acordo com as validações do formulário de cadastro.

Delete: Funcionalidade para deletar **Categoria.**

Fonte: Os autores (2020)

Tabela 3 - Requisitos Não Funcionais

Requisitos não funcionais				
Nome	**Restrição**	**Categoria**	**Obrigatoriedade**	**Permanência**
RNF002 - Duplicidade	Não serão aceitos cadastros em duplicidade.	Interface	( ) Desejável (X) Obrigatório	(X) Permanente ( ) Transitório
RNF 006 - Disponibilidade	O sistema deve estar disponível, 24 horas por dia, 7 dias por semana e 365 dias no ano	Confiabilidade	( ) Desejável (X) Obrigatório	(X) Permanente ( ) Transitório

Fonte: Os autores (2020)

Tabela 4 - RN 001

RN 001 – Controle de Estoque
Descrição: O aplicativo deve realizar o controle de medicamentos em estoque, assim que o usuário confirmar à tomada de um medicamento em algum horário de medicação no aplicativo, a quantidade deverá ser subtraída do estoque

Fonte: Os autores (2020)

Tabela 5 - RN 003

RN 003 – Controle de Medicamentos Esgotados
Descrição: O software deve fazer o controle dos medicamentos que estão esgotados, deve deixar com a tag cinza claro, os medicamentos que acabaram de esgotar.

Fonte: Os autores (2020)

Tabela 6 - RN 004

RN 004 – Notificação de rotinas
Descrição: O aplicativo deve notificar o usuário, proprietário da Prateleira de rotinas de participantes sem cadastros ligados a ele.

Fonte: Os autores (2020)

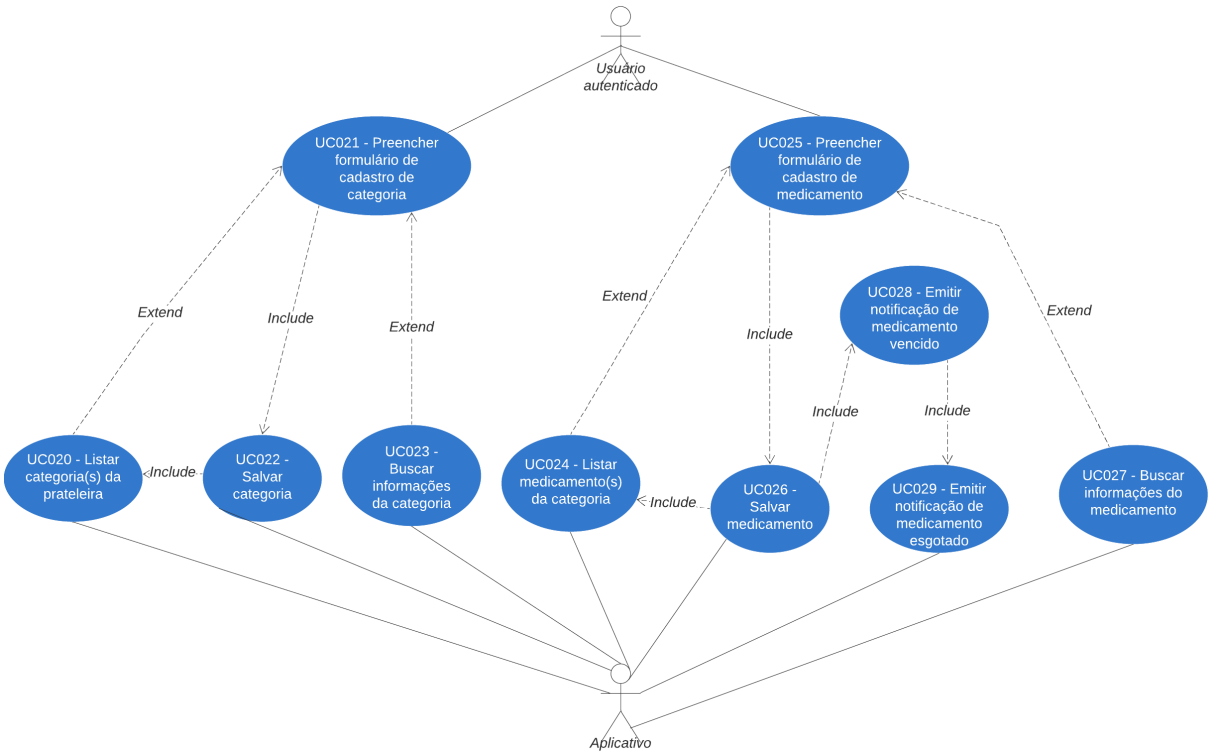
5.4 Caso de Uso

O caso de uso é um diagrama que descreve as principais funcionalidades do sistema e as interações delas com os usuários, descreve as ações dos atores por meio de um fluxograma que foi utilizado para a elaboração do sistema.

Após o levantamento de requisitos, mapeamento do projeto, analisamos o processo de modelagem do BPMN e a partir da derivação das especificações dos requisitos foi feito um levantamento de informações para a criação dos casos de usos, com seus respectivos atores, eventos e relacionamentos/ações (*includes* e *extends*).

Abaixo mostraremos apenas um módulo do caso de uso, devido ao seu tamanho e para uma melhor organização dos componentes, fizemos a divisão dos processos em módulos, abaixo está o processo do módulo de categorias/medicamentos, os demais módulos estão disponíveis no repositório do *GitHub*.

Figura 6 - Caso de Uso - Processo categoria e medicamento



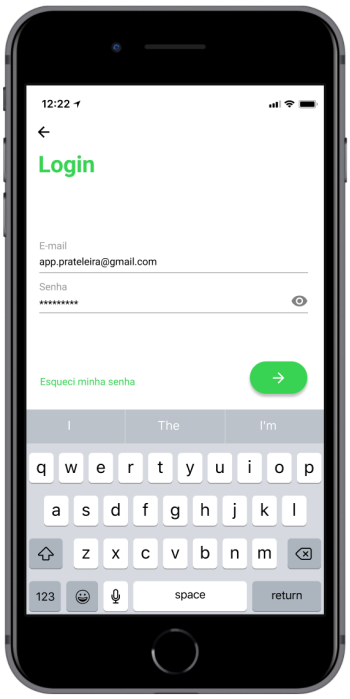
Fonte: Os autores (2020)

5.5 Prototipação de telas

A prototipação das telas do aplicativo é uma importante fase do desenvolvimento, a partir dela temos ideia de como o *app* final ficará, passando essa ideia para as partes interessadas e equipe de desenvolvimento, com o protótipo é possível identificar como elementos de UI (*user interface)* e *UX (user experience)* serão feitos e fazer a validação desses elementos antes de desenvolvê-los, também pode ser usado como documento de validação, como utilizamos a fim de verificar e validar com as partes interessadas as funcionalidades do sistema. Abaixo, algumas telas desenhadas pelo Figma (ferramenta de prototipação), exibidas no dispositivo Iphone 8 (a esquerda) e o *App* finalizado sendo executado por emulador no dispositivo Iphone X (a direita). O protótipo completo e navegável está disponível no Figma.

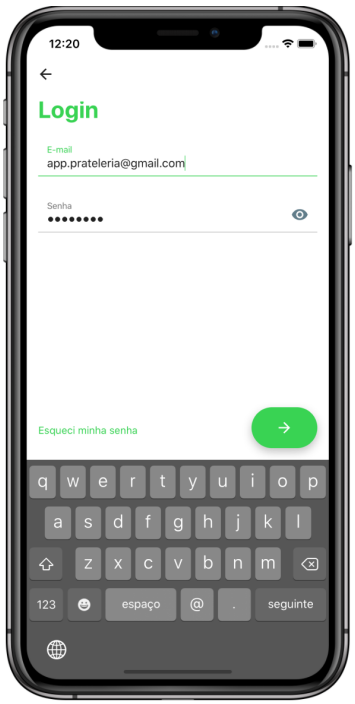
5.5.1 *Login*

Figura 7 - Protótipo login



Fonte: Os autores (2020)

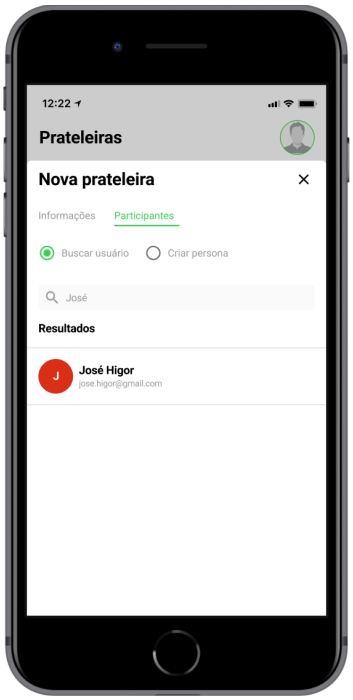
Figura 8 - Tela Login



Fonte: Os autores (2020)

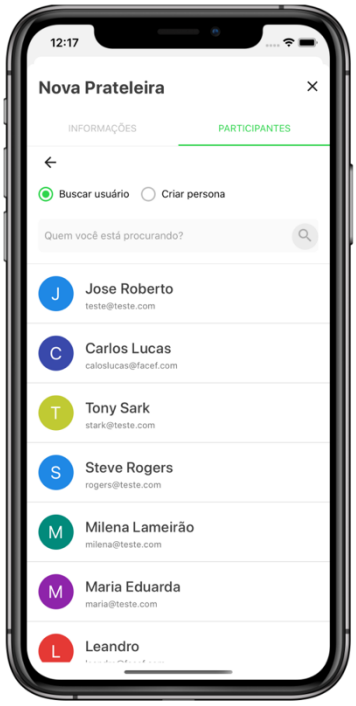
5.5.2 Busca de usuário na criação/edição de Prateleira

Figura 9 - Protótipo cadastro de prateleira



Fonte: Os autores (2020)

Figura 10 - Tela cadastro de prateleira

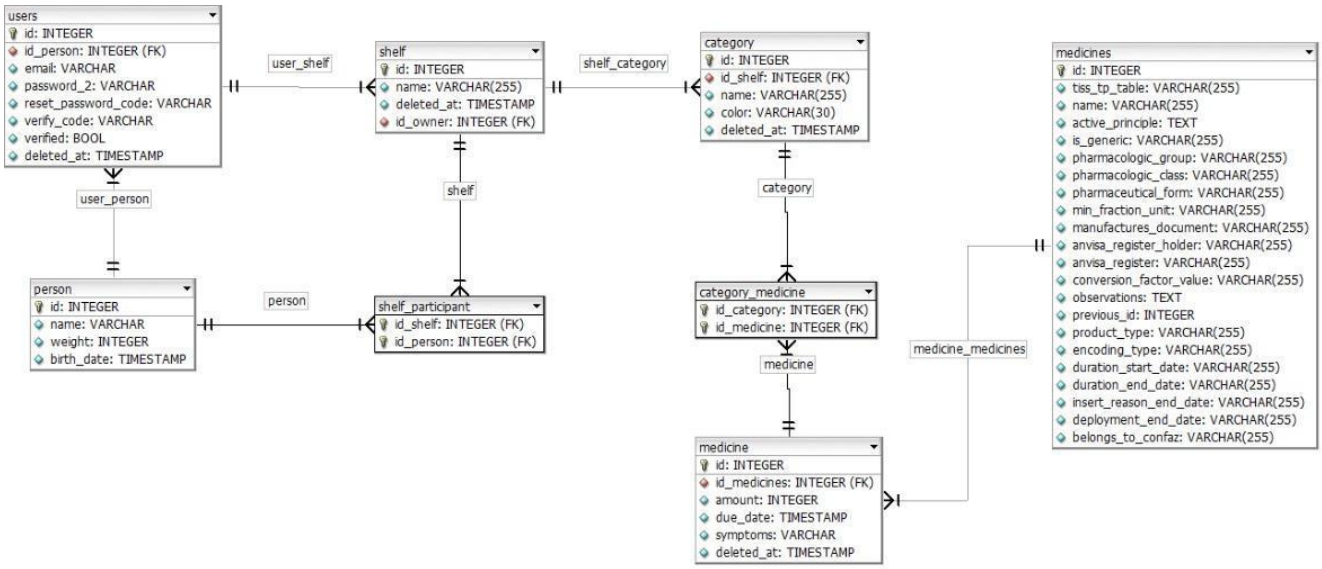


Fonte: Os autores (2020)

5.6 Banco de dados

Utilizando o DER (Diagrama Entidade Relacionamento), buscamos neste tópico ilustrar a estrutura do banco de dados da aplicação, o DER consiste em um fluxograma que utiliza de símbolos (como retângulos e linhas) para expressar o relacionamento entre entidades.

Figura 11 - DER



Fonte: Os autores (2020)

5.7 Qualidade do produto de *Software*

Como método para se atingir alguns critérios de qualidade e eficiência do produto, definidos pela ISO, foram realizados testes de Caixa Preta nas funcionalidades do aplicativo, conforme a imagem abaixo, os testes realizados para a funcionalidade de Cadastro.

Tabela 12 - Modelo de teste de caixa preta

Casos de Teste - RF 001 – Cadastrar usuário

	Entradas						Saídas esperadas
Nome	**E-mail**	**Data de nascimento**	**Peso (kg)**	**Senha**	**Confirmação de senha**	**Usuário já existe ?**	
11!@	jhigorfoliveira@gmail.com	25/05/1999	80	123qwe	123qwe	N	Error | "Não é permitido números, caracteres e símbolos no campo Nome. Só é possível letras na inserção."
vazio/null	jhigorfoliveira@gmail.com	03/05/1999	75	Seiasenha	Seiasenha	N	Error | "O campo nome não pode ser vazio".
Rodolfo	rodolf*()@gmail.com	25/03/1990	100	abcdef	abcdef	N	Error | "Não é permitido caracteres e símbolos no campo E-mail. Só é possível a inserção de e-mail no seguinte modelo: xx@xx.xx
João	vazio/null	1/1/1999	85	Joao1234567	Joao1234567	N	Error | "O campo e-mail não pode ser vazio".
Wilson	will@gmail.com	1/9/2020	82	123454321	123454321	N	Error | "Não é permitido caracteres, símbolos e letras no campo Data de Nascimento. O campo de seleção é no formato data(DD/MM/YYYY) e data de nascimento não poderá ser uma data futura.
José	jhigorfoliveira@gmail.com	vazio/null	110	godisbigger	godisbigger	N	Error | "O campo data nascimento não pode ser vazio".
Henrique	henrique@gmail.com	22/10/1999	Noventa	123454321	123454321	N	Error | "Não é permitido caracteres, símbolos e letras no campo Peso. Só é possível a inserção de peso com números de até 4 caracteres.
Leonardo	leonardo@gmail.com	20/02/1999	vazio/null	Leonardo10	Leonardo10	N	Error | "O campo peso não pode ser vazio".
Gabriel	gabriel@gmail.com	05/05/1980	120	1234567	1234567	N	Error | "O campo senha deve possuir no minímo 8 digítos".
Matheus	matheus@gmail.com	23/08/1998	80	vazio/null	Matheus99	N	Error | "O campo senha não pode ser vazio".
Guilherme	guilherme@gmail.com	15/01/1999	75	guilherme10	guilherme11	N	Error | "O campo confirmação de senha deve possuir no minímo 8 digítos e deve ser a mesma senha passada no campo anterior.".
Rafael	rafael@outlook.com	30/06/2000	60	rafael2000	vazio/null	N	Error | "O campo confirmação de senha não pode ser vazio".
José Roberto	jroberto@gmail.com	20/02/1950	110	ZeRoberto20	ZeRoberto20	S	Error | "Usuário com o e-mail: jroberto@gmail.com já existe criado na base de dados".
vazio/null	vazio/null	vazio/null	vazio/null	vazio/null	vazio/null	N	Error | "Os campos do formulário de cadastro não podem ser vazios."
José Higor	jhigorfoliveira@gmail.com	03/05/1999	110	Zehigor11	Zehigor11	N	Msg | "Usuário cadastrado com sucesso".

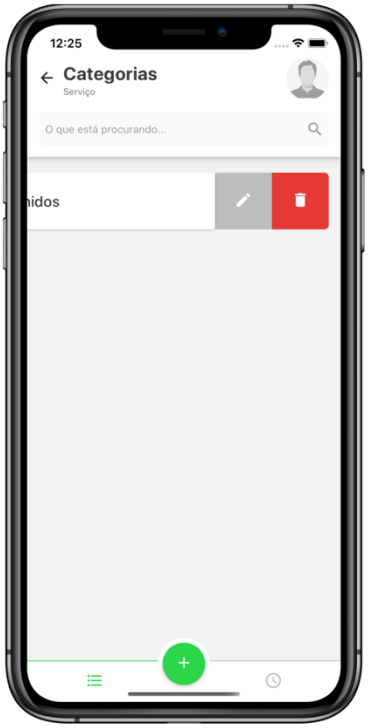
Fonte: Os autores (2020).

As atividades de Testes de Caixa preta têm o intuito de garantir a qualidade do produto de acordo com o documento de requisitos, e garantindo características da ISO de 2011 como: adequação funcional, usabilidade, efetividade e eficiência.

Perante a definição da característica de Usabilidade, afirma Wazlawick (2013, p. 234),” A usabilidade avalia o grau no qual o produto tem atributos que permitem que seja entendido, aprendido, usado e que seja atraente ao usuário, quando usado sob condições especificadas.”. A fim de satisfazer a inteligibilidade, que é o fato do aplicativo ser entendível e intuitivo, utilizamos recursos e itens que facilitam a percepção e o entendimento dos usuários. Através de ícones, símbolos e cores, é possível que os usuários possam saber seu intuito, ação e significado.

Abaixo, na tela de listagem de categoria do aplicativo, utilizamos o símbolo de adição, que indica adição de uma nova categoria, para edição de informações em um determinada categoria, usufruímos do ícone no formato de um lápis, para filtragem utilizamos o ícone de uma lupa, que visa realizar algum filtro/busca em alguma categoria, e, para exclusão, utilizamos o ícone de no formato de uma lixeira, além da coloração avermelhada indicando atenção, para apagar determinada categoria.

Figura 13 - Tela de listagem de categorias

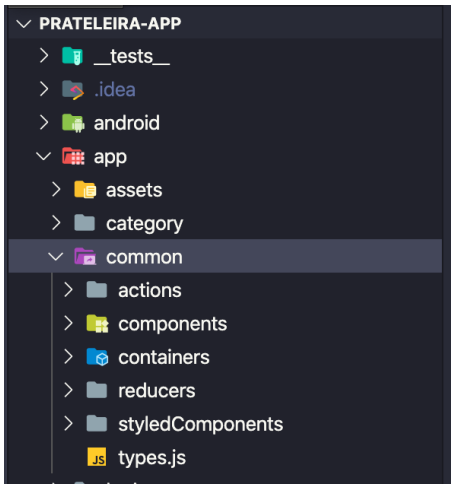


Fonte: Os autores (2020).

Sobre a definição da capacidade de manutenção de *software*, de acordo com Wazlawick (2013) é um aspecto oculto de *software* que é notável diretamente apenas pelos desenvolvedores, dentro de suas subcaracterísticas, a reusabilidade visa avaliar o grau em quais partes do sistema de *software* podem ser reutilizados, a fim de construir novas partes ou novos sistemas de *softwares*.

O *React Native*, linguagem para desenvolvimento do aplicativo, tem como princípio a componentização, desta maneira incentivando a reutilização de componentes em várias funcionalidades. Por exemplo, o *Header* de todas as entidades é o mesmo componente, assim como o FAB (*Floating action Button*), os componentes comuns entre entidades são armazenados na pasta *common* > *Components*.

Figura 14 - Pasta commons



Fonte: Os autores (2020)

No quesito segurança da ISO, também buscamos atender algumas subcaracterísticas, tais como, confidencialidade que segundo Wazlawick (2013, p. 234), é "o grau em que as informações e funções do sistema são acessíveis por quem tem a devida autorização para isso" e Integridade definida por Wazlawick (2013, p. 234), "o grau em que os dados e funções do sistema são protegidos contra acesso por pessoas ou sistemas não autorizados.” realizando a autenticação do usuário através de e-mail e senha, as alterações em entidades como Categorias e Medicamentos podendo ser realizadas apenas por participantes adicionados.

As API's também são autenticadas, ou seja, consumi-las sem um *token* válido de acesso não será possível, *token* que só é conseguido com o *login* citado anteriormente. As credenciais para *login* são feitas no Cadastro, todos os dados de senha são criptografados, não podendo ser decifrados pela equipe de desenvolvimento.

O cadastro atende a norma de Autenticidade da ISO, para Wazlawick (2013, p. 234), "o grau em que a identidade de uma pessoa ou recurso é efetivamente aquela que se diz ser". Ao informar um endereço de e-mail, o usuário deve acessar um *link* enviado para o endereço de e-mail informado, desta maneira não podendo utilizar o e-mail de outra pessoa para acessar o *app*.

A funcionalidade de exclusão de itens atende a subcaracterística de recuperabilidade que segundo Wazlawick (2013, p. 234) "está relacionada à sua capacidade de recuperar dados e colocar-se novamente em operação após uma situação de desastre.", pertencente à confiabilidade do *software*, de maneira que nenhum item é excluído do banco de dados, apenas inativado, sendo exibidos apenas os dados ativos para o usuário mas mantendo os excluídos para uma possível recuperação necessária.

6 Considerações finais

O projeto se deu por início no objetivo de aprimoramento e implementação de uma ideia, com levantamento de requisitos com especialistas da área, a fim de termos um MVP (Mínimo Produto Viável) a ser entregue; prosseguiremos

para o próximo passo de planejamento de execução do projeto, que incluíam definição de escopo, implementação de metodologia de desenvolvimento e artefatos a serem realizados da Engenharia de *Software*.

A partir da definição do escopo, foram mapeados os entregáveis a serem feitos por *Sprint* utilizando a metodologia de desenvolvimento ágil *Scrum*, para o controle de tarefas e prazo foi utilizada a metodologia *Kanban* com auxílio da ferramenta Jira. Os entregáveis, sendo os artefatos de engenharia de *software* e o desenvolvimento do aplicativo, foi iniciado e seguido de acordo com o planejamento mensal.

Dentre os problemas encontrados durante o projeto, podemos destacar a conciliação entre o que foi pensado anteriormente à consulta do profissional da área da saúde com o que deveria de fato ser realizado, ocorrendo, após a consulta, uma drástica mudança de perspectiva em relação à área da saúde e seus usuários e grande mudança de escopo.

Concluindo, as definições de escopo para o MVP, foram atendidas, sendo realizados todos os artefatos de engenharia de software e desenvolvimento do aplicativo para os módulos de Autenticação, Prateleira, Categorias e Medicamento. Para evolução e aprimoramento do aplicativo, as futuras implementações buscarão aplicar as funcionalidades de rotinas e notificações, para atendimento do escopo completo mencionado neste artigo, também uma melhora no banco de dados a fim de atender pontos triviais ao primeiro MVP.

Referências

AMARAL, Hugo Richard; LIZARDO, Luis Eduardo Oliveira; DE SOUZA, Arthur Camara Vieira. PostgreSQL: uma alternativa para sistemas gerenciadores de banco de dados de código aberto. In: Anais do Congresso Nacional Universidade, EAD e Software Livre, 2011.

BARTIÉ, Alexandre. *Garantia da qualidade de software*. Rio de Janeiro: Gulf Professional Publishing, 2002.

DANIELSSON, William. React Native application development. *Linköpings universitet*, Swedia, v. 10, p. 4, 2016.

DE VASCONCELOS, Alexandre Marcos Lins et al. Introdução à Engenharia de Software e à Qualidade de Software. Minas Gerais: Universidade Federal de Lavras, 2006.

ENGHOLM JR, Hélio. *Engenharia de software na prática*. São Paulo: Novatec Editora, 2010.

HANSSON, Nicolas; VIDHALL,Tomas. *Effects on performance and usability for cross-platform application development using React Native*. Institutionen för datavetenskap. Linköpings universitet, Linköpings, Suécia, 2016.

ISMP (Instituto para Práticas Seguras no Uso de Medicamentos): DESAFIO GLOBAL DE SEGURANÇA DO PACIENTE MEDICAÇÃO SEM DANOS. Minas Gerais, 2018.

JOÃO, Walter da Silva Jorge. Descarte de medicamentos. *Revista Pharmacia Brasileira*, v. 82, n. 82, p.14-16, 2011.

KAIMER, Fabian; BRUNE, Philipp. Return of the js: Towards a node. js-based software architecture for combined cms/crm applications. *Procedia Computer Science*, v. 141, p. 454-459, 2018.

MILANI, André. *PostgreSQL*-Guia do Programador. Novatec Editora, 2008.

MOMJIAN, Bruce. *PostgreSQL:* introduction and concepts. New York: Addison-Wesley, 2001.

PALADINO, Andrew J. *et al*. THRIVE study protocol: a randomized controlled trial evaluating a web-based app and tailored messages to improve adherence to adjuvant endocrine therapy among women with breast cancer. *BMC Health Services Research*, v. 19, n. 1, p. 1-12, 2019.

PEREIRA, Caio Ribeiro. *Aplicações web real-time com Node. js.* Editora Casa do Código, 2014.

PRESSMAN, Roger; MAXIM, Bruce. *Engenharia de Software*. 8. ed. Brasil: McGraw Hill Brasil, 2016.

REZENDE, Denis Alcides. Engenharia de software e sistemas de informação. 3. ed. Rio de Janeiro: Brasport, 2006.

SCHWINGEL, Débora et al. Farmácia caseira x uso racional de medicamentos. Revista Caderno Pedagógico, v. 12, n. 3, 2015.

SOMMERVILLE, Ian. *Engenharia de Software*. 9. ed. São Paulo: Pearson, 2011.

TADDEO, Patricia da Silva *et al*. Acesso, prática educativa e empoderamento de pacientes com doenças crônicas. *Ciência & Saúde Coletiva*, v. 17, p. 2923-2930, 2012.

WAZLAWICK, Raul. *Engenharia de Software - Conceitos e Práticas*. 1. ed. Rio de Janeiro: Elsevier, 2013. Disponível em https://integrada.minhabiblioteca.com.br/#/books/9788595156173. Acesso em: 14 jun. 2020