APLICATIVO DE AVISOS E GESTÃO DE ROL DE MEMBROS DE IGREJAS

Dylan Oliveira, Bruno de Sousa Monteiro

Resumo: Esse trabalho tem como objetivo modernizar o controle de rol de membros de igrejas, para que seja possível tomar mais decisões baseadas em dados, trazendo também a função de mostrar os eventos aos membros comuns e visitantes em uma única plataforma. Esta função, além de ser mais uma opção de canal de comunicação para manter os usuários engajados nas programações da igreja, torna-se também um motivador para que os integrantes acessem a aplicação e façam o auto cadastro, facilitando o trabalho do gerente de rol de membros, evitando que ele precise manualmente cadastrar e manter os dados dos membros atualizados, permitindo uma gestão mais eficiente da gerência da igreja. A plataforma foi desenvolvida a partir de estudo de caso da Igreja Presbiteriana Central de Mossoró, juntamente com pesquisa documental, de projetos concorrentes, que culminou na concepção do modelo conceitual e de dados, arquitetura e definição de tecnologias. Por fim, as aplicações web e mobile foram validadas com usuários, cujos relatos apontam para objetivos alcançados, e também para os próximos passos para que o sistema possa entrar em produção.

Palavras-chave: sistema web; igreja; avisos.

1. INTRODUÇÃO

No Brasil, segundo a Secretaria Executiva de Estatísticas da Igreja Presbiteriana do Brasil, com dados de 2021, mais de 700 mil membros compõem sua sociedade, possuindo um crescimento estimado de 2% ao ano [1]. Para cada novo membro, faz-se necessário a criação de seu registro para acompanhar o crescimento da instituição, com o propósito de conhecer aqueles que passam a ter vínculo com a igreja, a fim de manter o seu bom funcionamento e a comunhão daqueles que a frequentam.

O trabalho a seguir foi desenvolvido a partir das demandas da Igreja Presbiteriana Central de Mossoró[1] (IPCM), que tem um processo pouco informatizado para sua gestão de membros e avisos. O software utilizado atualmente é o *iCalvinus*[2], que não recebe atualizações nem melhorias de acordo com o que foi reportado pelos seus usuários, fazendo assim necessário recorrer a planilhas e documentos paralelos para suprir as faltas do sistema. Desse modo, foi levantada a problemática sobre o quanto esse problema é comum entre as igrejas do Brasil, e como o modelo tradicional pode ser otimizado.

O sistema oficial da Igreja Presbiteriana do Brasil[3] (IPB), *iCalvinus*, foi promovido por sua secretaria executiva em 2010, com o propósito de informatizar os processos e facilitar a captação de registros para as comissões executivas. Esse sistema contempla, hoje, como indicado em sua plataforma [2], 60% da base total de igrejas da organização no Brasil, o que nos mostra que as melhorias são necessárias em um cenário muito além da realidade mossoroense.

É notório que o *iCalvinus* tornou-se insuficiente para suprir as demandas de uma igreja local que possui a preocupação em armazenar informações detalhadas sobre os seus membros e mantê-los atualizados das atividades e eventos que acontecem na instituição. Além disso, foi relatado por membros da equipe responsável pelo rol do sistema da IPCM, que, devido a falta de atualizações do sistema, ele é pouco intuitivo aos usuários, fazendo com que a integração de novos administradores na plataforma seja um processo lento e difícil.

[1] Igreja Presbiteriana Central de Mossoró: http://ipcentralmossoro.com
[2] *iCalvinus:* https://igreja.icalvinus.net
[3] Igreja Presbiteriana do Brasil: https://www.ipb.org.br

1.1 Objetivo

Com base nos relatos apresentados, este trabalho tem como objetivo conceber, desenvolver e validar um novo sistema para a IPCM, com o intuito de fornecer um melhor gerenciamento de dados dos membros; comunicar aos membros da igreja das programações de forma mais ativa; e desonerar o gerente de rol de membros possibilitando o auto cadastro dos usuários.

1.2 Atividades de Pesquisa e Desenvolvimento

Para alcançar o objetivo estabelecido, foram executadas as seguintes atividades:

1. Entrevistar *stakeholders* para entender as necessidades da igreja e as dificuldades enfrentadas atualmente para atingir seus objetivos.
2. Realizar pesquisa de trabalhos correlatos, para entender quais softwares hoje em dia cumprem funções semelhantes, levando em consideração: funcionalidades, custo de aquisição, interface intuitiva e disponibilidade de multi-plataforma.
3. Elencar requisitos do sistema ideal, com base na entrevista.
4. Esboçar interface de usuário com base nos requisitos.
5. Definir arquitetura base, modelo de dados e tecnologias utilizadas para o desenvolvimento do software.
6. Desenvolver a infraestrutura de cliente/servidor, interfaces de usuário, serviços e banco de dados.
7. Validar o sistema implementado com os *stakeholders*.
8. Corrigir os erros identificados.

Com tais atividades executadas, espera-se ter uma plataforma pronta para expandir seu número de usuários e repetir os processos de: coletar *feedbacks,* desenvolver ou corrigir funcionalidades e por último, testar.

2. DESENVOLVIMENTO

Nesta seção, observa-se o projeto desde a sua concepção, partindo de onde os requisitos foram levantados, passando pelo planejamento da infraestrutura criada, modelo de dados, arquitetura e tecnologias, chegando na publicação da plataforma completa e os testes que foram realizados com os usuários, até fazer a análise dos resultados atingidos.

2.1. Levantamento de Requisitos

A primeira etapa do projeto foi definir os objetivos da aplicação, bem como as tarefas para alcançá-los. Para tal, seguem-se duas frentes neste projeto: um estudo de caso e uma pesquisa documental.

O estudo de caso ocorreu com a Igreja Presbiteriana Central de Mossoró, especificamente através de reuniões com a equipe responsável pelo rol de membros. O objetivo é entender como a equipe trabalha diariamente, quais ferramentas são utilizadas e quais as maiores dificuldades enfrentadas. Esse processo levou ao entendimento que os requisitos do sistema, que são:

- Ser capaz de registrar membros, e também gerenciar seus dados.
- Possibilitar que o membro mantenha seus dados atualizados na plataforma (somente como solicitação, pois a alteração definitiva deve ser feita somente por pastores e gerentes de rol de membros).
- Obter Sub-listas da lista de membros com base em uma ou mais características em comum entre os membros.
- Ter um espaço acessível para que todos possam visualizar as programações da igreja (classificando por grupos, como geral, sociedades internas, grupo de louvor etc.)

Ao final da implementação o sistema deve ser validado segundo os requisitos citados acima, atendendo as necessidades das partes interessadas na plataforma.

Quanto à pesquisa documental, mais especificamente a análise de concorrentes, foram comparados os critérios nos sistemas analisados: valores da mensalidade cobrada, plataformas disponíveis, funcionalidade de gerenciamento de membros, controle financeiro, controle de grupos, controle de agendas e notificações de avisos. Sendo assim, chegamos nos resultados que mostram a Tabela 1 a seguir:

Tabela 1. Comparação de funcionalidades entre alguns sistemas disponíveis.

Software	Plat.	Preço/mês	Membros	Financeiro	Grupos	Calendário	Notificação
ICalvinus	Web	Sistema exclusivo da IPB	Sim	Não informa	Sim	Não informa	Não
Cathedral	Web, mobile	R$43,90	Sim	Sim	Sim	Sim	Não
Eklesia	Mobile	R$45,00	Sim	Sim	Sim	Sim	Sim
Igreja Digital	Web, mobile	R$39,90	Sim	Sim	Sim	Não	Não
Church	Web, mobile	Requer orçamento	Sim	Sim	Não	Sim	Sim
Sigi	Web	R$47,00	Sim	Sim	Não	Não	Não

Fonte: autoria própria

Com a análise de concorrentes foi possível obter uma média de preços e uma perspectiva de funcionalidades futuras, como a criação de um módulo de gerenciamento de finanças. Contudo, nota-se que todos os produtos são pensados para serem genéricos independente da denominação, reforçando a ideia de que o sistema deve ser projetado para ser facilmente adaptado para futuramente se adequar a outras igrejas.

2.2. Modelo Conceitual

Com os requisitos bem estabelecidos, o próximo passo foi definir quais tipos de pessoas irão utilizar o sistema. Foram estabelecidos empiricamente cinco perfis baseado nos casos de uso do sistema, sendo o primeiro deles, o pastor, que é um homem de 30 a 60 anos, responsável pela gestão da igreja e que deve delegar tarefas, criar grupos e avisos. Já o segundo perfil é do gerente de rol de membros, seu objetivo é zelar e atualizar os dados dos membros e visitantes. O terceiro perfil corresponde ao administrador de grupo, que pode gerenciar membros, e avisos do seu grupo. Já o quarto perfil é o usuário comum, esse é o membro da igreja que pode solicitar acesso aos grupos, atualizar seus dados pessoais e visualizar membros. O último perfil é o de visitante, capaz somente de ver os avisos da igreja.

Com os perfis definidos, a próxima etapa foi criar a versão macro ou esboço de baixa fidelidade do sistema web/mobile. Essa versão tem como objetivo definir o modelo conceitual, mapeando o objetivo dos usuários e comos eles serão concluídos dentro da interface base do sistema, pensando na localização dos botões e na navegação do usuário pelo site.

A interface desse sistema é pensada para facilitar a utilização da aplicação para os mais diversos públicos, focando o usuário de mais idade. Sendo assim, é importante deixar as opções de uso do sistema bem visíveis evitando a convencional barra de menus e submenus. Assim os objetos importantes estão centralizados na tela principal, e os botões auxiliares ficam localizados nas bordas da janela. Outro ponto importante é a utilização de fontes grandes e legíveis, sem abreviações nas frases e evitar botões sem nenhum texto descritivo (botões apenas com o ícone).

O sistema está disponível para acesso web e mobile, sendo recomendado que o acesso administrativo seja feito pelo desktop, onde existe maior disponibilidade de recursos e visibilidade das funcionalidades aplicadas. A Figura 1 a seguir, mostra parte do esboço desenvolvido.

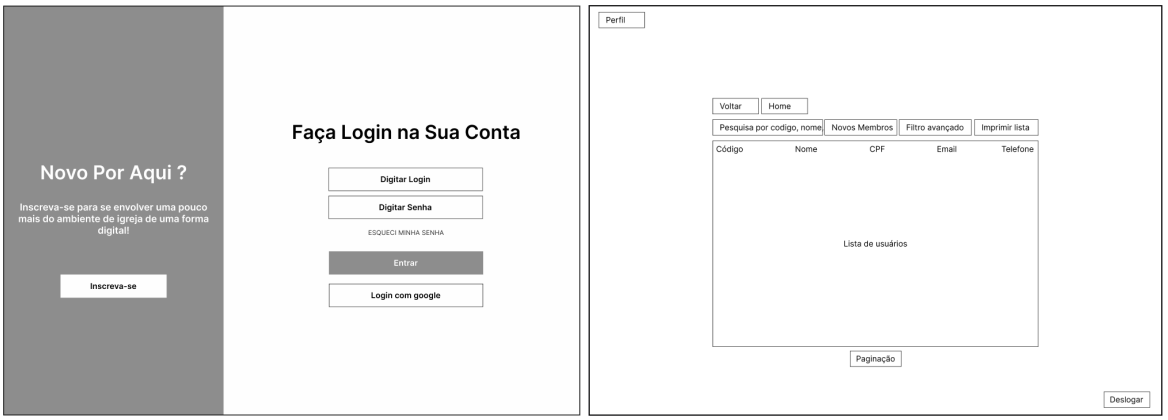

Figura 1. Exemplos de esboço de tela de baixa fidelidade. (Autoria própria)

Após o desenvolvimento da versão macro, é feito um desenho mais detalhado, trazendo a preocupação das fraseologias, imagens, design dos botões, ícones, tipografia, cores e estilização da página.

2.3. Modelo de Dados e Funcionalidades

Com os esboços desenvolvidos, chegamos ao início do processo de planejamento do modelo de dados, que se trata da estruturação e formalização das entidades ou classes e seus relacionamentos na plataforma desenvolvida. Durante esse processo, baseado nas discussões previamente apresentadas, chegamos à conclusão de que o sistema possui três entidades principais: Pessoas, Grupos e Eventos. Nesta seção será detalhada cada uma das três entidades com informações adicionais sobre as suas respectivas funções dentro do sistema.

2.3.1. Pessoas

A entidade "Pessoa" diz respeito a um ser humano comum, e contém as informações cadastrais pessoais como nome completo, CPF, endereço, família, profissão e também dados congregacionais, como data de batismo, ofícios dentro da igreja, entre outros. A diferença entre um pastor, um gerente, um membro e um visitante está na entidade “Auth”, que contém dados de acesso como email, senha, nível de acesso e a pessoa que a contém. Segue na Figura 2 um modelo *Unified Modeling Language* (UML) das entidades citadas.

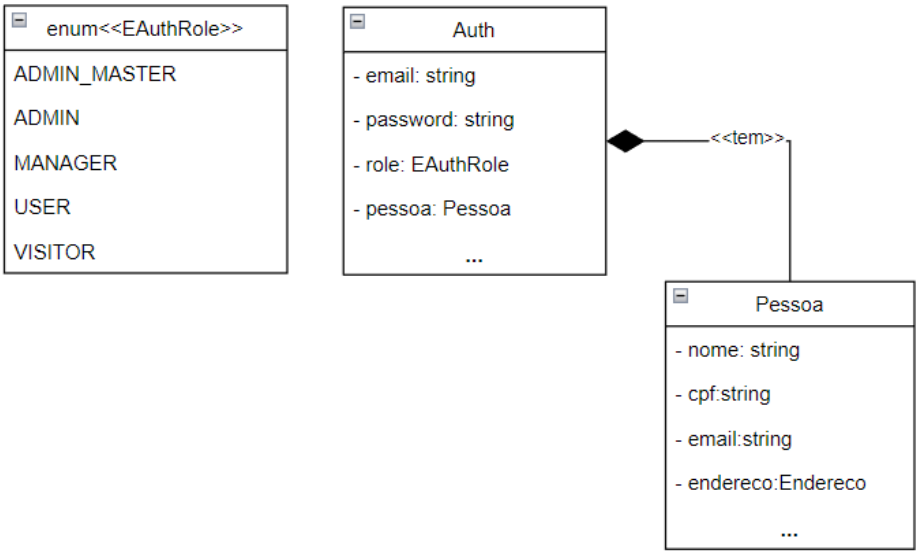

Figura 2. Recorte do modelo UML de Auth e Pessoa. (Autoria própria)

A pessoa registrada no sistema não depende de uma entidade “Auth”, pois ela pode ter sido cadastrada pelo gerente da igreja apenas para fins estatísticos, e, nesse caso, é uma pessoa sem conta na plataforma. Contudo, para os usuários que desejam utilizar a aplicação a fim de acessar grupos e ver os eventos, precisam criar uma conta. Para isso, existem duas maneiras, podendo ser feito o cadastro manual diretamente pelo sistema, ou realizando uma autenticação com a conta da *Google*. Essa autenticação permite que com apenas dois cliques, seja possível estar cadastrado no sistema. Isso é importante para que os membros e visitantes tenham o mínimo de atrito possível ao acessar a plataforma. Assim, uma vez que o membro se cadastrou, automaticamente ele tem uma entidade “Auth” e uma entidade “Pessoa”.

Sobre a autenticação do sistema, quando um usuário criar uma conta manualmente, é necessário criar uma senha através de um link específico que é enviado por email. A senha escolhida é encriptada com *SHA-256* e armazenada no banco de dados. O usuário é criado com o nível de acesso de visitante, podendo solicitar ser um membro da igreja. Ao efetuar o login, a senha enviada para conferência é encriptada e então, o sistema verifica se o texto enviado e a senha armazenada no banco são iguais. Em caso de sucesso, é gerado um token do tipo *JSON Web Token (JWT)*[4] para a aplicação do usuário, a partir daí o usuário é autenticado. No caso de login via *Google*, o processo é semelhante, todavia, ao invés de comparar as senhas encriptadas, o sistema verifica se existe uma conta com o email fornecido pelo token gerado pelo *oauth*, só então é gerado o token JWT para o usuário ser autenticado.

[4] JWT: https://jwt.io/

Sobre o disparo de notificações ativas, acontece toda vez que uma entidade do tipo “Evento” é criada dentro de um “Grupo” que o usuário pertence (às entidades “Evento” e “Grupo” serão detalhados posteriormente). Para que esse processo seja possível, o usuário deve aceitar o recebimento de notificações que é solicitado pelo navegador, uma vez aceito, sempre que o usuário entra na aplicação, é gerado um *Firebase Cloud Functions + Cloud Messaging* (FCM) *token*[5], que também é salvo no objeto “Auth” do usuário, assim antes de disparar é consultado uma lista de todos os FCM *token* dos usuários de um grupo, e então as notificações são enviadas com base nessa lista.

Referente a atualização dos dados pessoais, o sistema não permite que o usuário altere diretamente seus dados. Partindo do princípio que a missão principal do sistema é manter os dados reais atualizados na plataforma, a fim de evitar que algum usuário modifique informações erroneamente em sua ficha, o sistema entende a atualização cadastral como uma solicitação de alteração de dados. Essa solicitação é exposta ao gerente da igreja que irá aceitar ou rejeitar a alteração realizada, após a conferência da veracidade dessa modificação. A Figura 3 mostra a tela de edição de endereço, onde o usuário solicitou alterar o campo “Complemento”, para o gerente do sistema o campo aparece de três formas, o valor original, o valor que o usuário solicitou e o valor final que o gerente escolhe. Já a Figura 4 mostra a tela de gestão de usuários acessível somente para administradores e gerentes da plataforma.

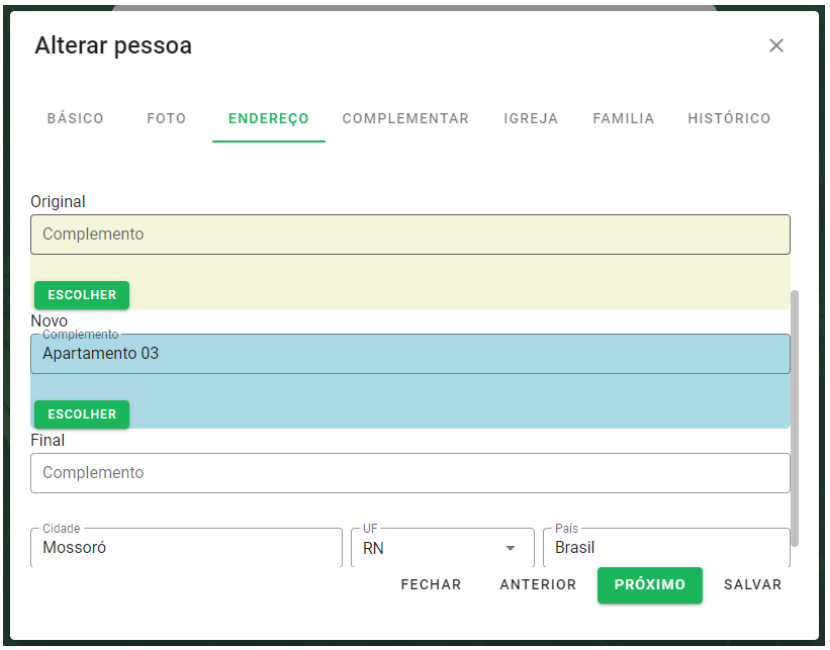

Figura 3. Tela de atualização de usuário. (Autoria própria)

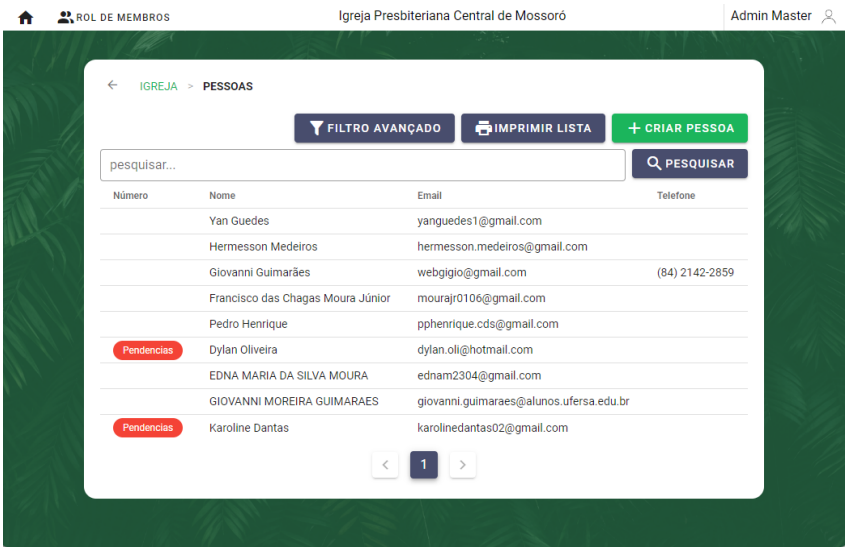

Figura 4. Tela de gestão de usuários. (Autoria própria)

[5] FCM: https://firebase.google.com/docs/cloud-messaging?hl=pt-br

2.3.2. Grupos

A entidade “Grupo” é pensada para ser generalista entre os diversos grupos criados. O objetivo é criar uma estrutura do tipo árvore, no qual um grupo principal possui diversos subgrupos, sendo a raiz da árvore a denominação IPB que opera a nível nacional. Seus nós são as igrejas que, por sua vez, se subdividem em folhas, sendo estas as congregações, departamentos e sociedades internas como demonstra a Figura 5. Essa estrutura foi pensada para realizar a escalabilidade do sistema sem grandes empecilhos, caso ele venha a oferecer suporte a IPB a nível regional ou mesmo a outras denominações com uma estrutura diferente.

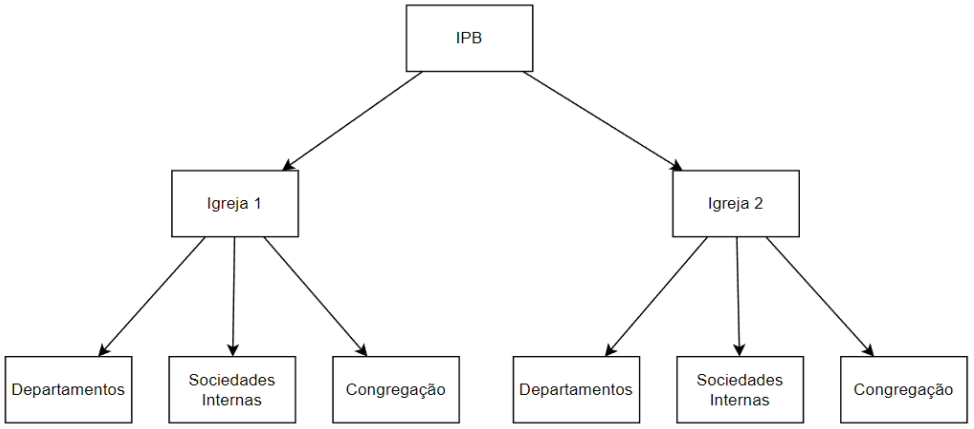

Figura 5. Estrutura de árvore ilustrativa da IPB segundo o sistema. (Autoria própria)

Todos os grupos possuem atributos genéricos, como nome, descrição, eventos e uma lista de uma relação. Essa lista está denominada “PessoaGrupo”, que contém a pessoa vinculada e seus atributos referentes àquela relação, como a data em que o usuário aderiu ao grupo e se ele é administrador. O que diferencia um grupo de outro é um enumerador, que tem atualmente os seguintes itens: denominação, igreja, sociedade interna, departamento e congregação. O modelo UML dessa entidade está representada na Figura 6 abaixo:

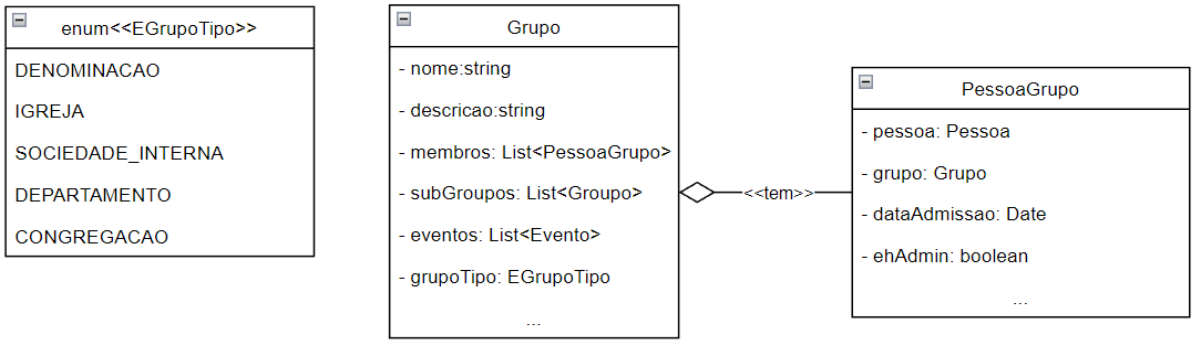

Figura 6. Recorte do modelo UML de Grupo e PessoaGrupo. (Autoria própria)

Estar em um grupo significa que o usuário poderá ver seus eventos e membros, além de ser notificado quando um novo evento é criado dentro desse grupo. Para que um usuário entre em um grupo, ele pode ser convidado por um administrador ou pode solicitar a entrada, nesse caso o administrador também é responsável por aceitar ou rejeitar a solicitação. Já os administradores, além de adicionar novos membros, também podem adicionar novos administradores e criar eventos. O Pastor e os gerentes da igreja, são automaticamente administradores e não precisam solicitar a entrada em outros grupos. A Figura 7 a seguir é a visão da igreja local, que como citado anteriormente, também é um grupo.

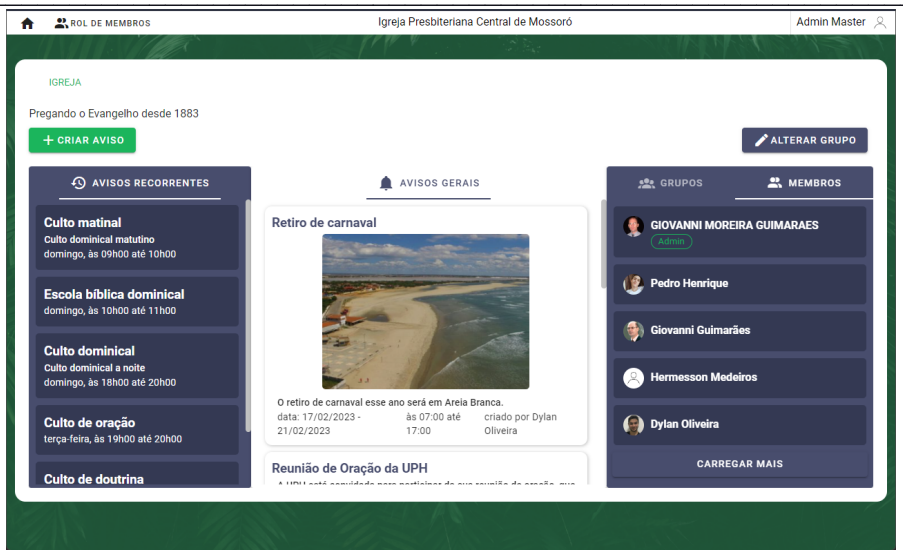

Figura 7. Tela de um grupo. (Autoria própria)

2.3.3. Eventos

Em “Eventos” ou como é apresentado no *frontend* “Avisos”, temos a funcionalidade de informar aos usuários sobre as programações de um grupo. A entidade se divide em dois grupos, sendo um deles o de eventos recorrentes, (programações que ocorrem semanalmente, como o culto dominical ou culto de oração), e os não recorrentes (acontecem sem a necessidade de se repetir, como um culto de ação de graças, retiro ou ação social). Os atributos de um evento são: data de início e fim, data de publicação, quem o criou, o grupo a qual pertence, um título e uma descrição. Assim, quando um evento é criado, os participantes são notificados. Segue na Figura 8 o modelo UML do evento e na Figura 9 a tela de cadastro de eventos.

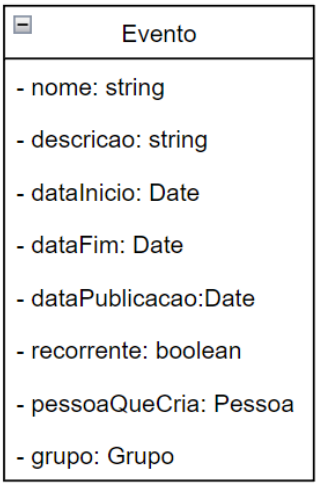

Figura 8. Recorte do modelo UML de Evento. (Autoria própria)

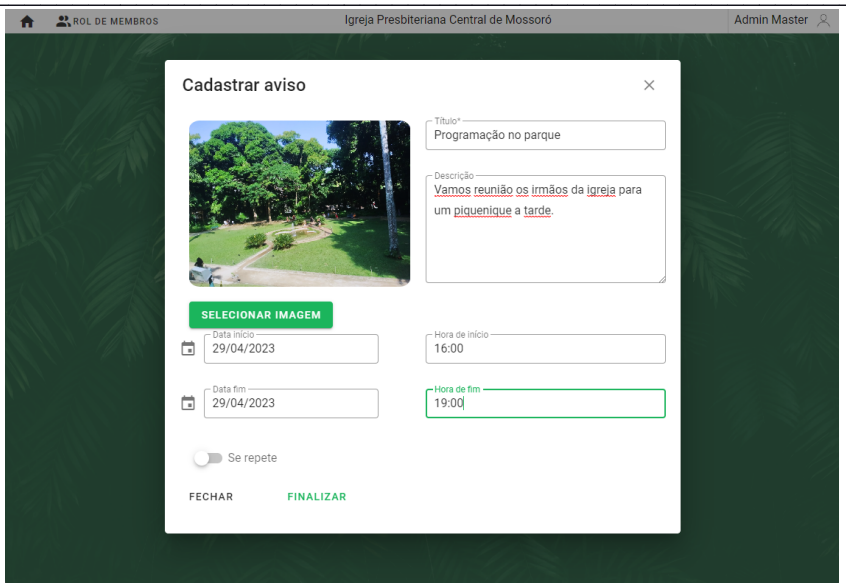

Figura 9. Tela de criação de eventos. (Autoria própria)

2.4. Arquitetura do Sistema

O sistema é todo pensado para ser escalável baseando-se nos conceitos de *Clean Code* [3] e *SOLID* [4], e, com isso em mente, o projeto é dividido em dois serviços: o *frontend* para o cliente e *backend* para o servidor. Dessa forma, é possível adaptar o projeto a outras denominações religiosas apenas com a criação de uma nova variação do *frontend*, por exemplo.

O *backend* é uma *Application Programming Interface* (API) que fornece todos os serviços da plataforma. Ele segue um modelo em camadas sendo a primeira a *controllers*, camada esta responsável por expor os serviços da API para outras aplicações, além de fazer o tratamento inicial dos dados como os parâmetros enviados pelo cliente ao fazer a requisição e se ele tem o *token* de acesso. Em seguida o *controller* chama o *service* que compõem a segunda camada. Ela é responsável por definir todas as regras de negócios (incluindo permissões de acesso ao serviço em específico), fazendo chamadas a serviços de terceiros ou consultando a próxima camada definida como *repositories* que tem comunicação direta com o banco de dados, logo, tem a função de fornecer e alterar dados persistentemente. Existe uma última camada de *models,* responsável por definir as entidades que trafegam entre as demais camadas do *backend*.

Já o *frontend* utiliza os conceitos de componentização, no qual todas as partes de uma página web, das maiores às menores, são componentes. Tendo isso em mente, este serviço se divide nos seguintes pacotes: *assets*, que armazena conteúdo de mídia estático, como, ícones e logos; *components,* que responsável por agrupar o componentes reutilizáveis da aplicação; *models*, que semelhante ao *backend*, traz a definição das entidades; o pacote *routers* gerencia as rotas da aplicação, onde cada rota faz chamada a uma página, localizada no pacote de *views*; O próximo pacote é o *services*, que faz as chamas a serviços externos, como a próxima API da aplicação; o último se chama *storage*, que cria e controla os estados da aplicação. Segue na Figura 10 um modelo ilustrativo da arquitetura do sistema.

2.5. Tecnologias Adotadas

Em relação às tecnologias, o *backend* é desenvolvida em ASP.NET[6] na sua versão 7 com a linguagem C#. Na camada de serviço, foi utilizado na autenticação o Google oAuth[7] e para o envio de notificação ativa o Google Firebase[8], para a manipulação de *JavaScript Object Notation* (JSON) foi utilizado a biblioteca *Newtonsoft.Json*[9], para a criação de listas de membros em formato de *Portable Document Format* (PDF), o sistema usa as ferramentas do *iTextSharp*[10], para chamadas do tipo *Hypertext Transfer Protocol* (HTTP) é usado a biblioteca *RestSharp* [11]. Olhando para a camada de *Repository*, pode-se destacar o banco de dados utilizado,

[6] ASP.NET*:* https://dotnet.microsoft.com/en-us/apps/aspnet
[7] Google oAuth*:* https://developers.google.com/identity/protocols/oauth2
[8] Google Firebase: https://firebase.google.com/?hl=pt
[9] Newtonsoft.Json:https://www.newtonsoft.com/json
[10] iTextSharp: https://itextpdf.com/products/itextsharp
[11] RestSharp: https://restsharp.dev/

sendo este o PostgreSQL[12], para a comunicação entre a aplicação e o banco de dados é utilizado a biblioteca *Npgsql*[13]. Outras tecnologias que não foram utilizadas em uma camada específica, mas na aplicação como um todo foram: o *Quartz*[14] para o agendamento de tarefas, como remoção de mídia temporária e o *NUnit*[15] para a realização de testes unitários.

No serviço *frontend,* o esboço foi feito inicialmente com o Figma. Quanto ao desenvolvimento foi utilizado a *framework* Vue JS[16] na sua segunda versão na linguagem *Typescript*, com estilização baseada no *Material Design*[17] através da biblioteca Vuetify[18]. Pensando na compatibilidade com mobile o *frontend,* utiliza-se a padronização de *Progressive web app* (PWA). Na camada de *components* foram utilizadas as seguintes bibliotecas: *vue-property-decorator*[19] para integrar melhor o *typescript* no vueJS, *momentJs*[20] para manipulação do tempo, *jwt_decode*[21] para a leitura de JWT, *vue-toast-notification*[22]. Para criar componentes de notificação, *vue-the-mask*[23] para máscaras no texto, vue-google-oauth2[24] para autenticação com o google, *vue-cropperjs*[25] para criação dos componentes de corte de imagem. Para a camada de *storage* é utilizado o *vuex*[26] para o gerenciamento de estados. Na camada de Router é usado o *vou-router*[27]. Para o controle de rotas, por último na camada Service é utilizado o *axios* [28]para fazer requisições HTTP.

2.6. Validação com Usuários

Uma vez que o sistema foi desenvolvido, é necessário testar com os usuários com a finalidade de saber se a interface está realmente intuitiva e se os objetivos de usuário foram contemplados. Para os testes em questão, foi utilizado o protocolo verbal *think aloud* [5, 6], no qual a pessoa que testa recebe a orientação com missões a cumprir, e, até atingir o objetivo, ela tem de falar em voz alta seus pensamentos a respeito do que está fazendo e de suas impressões. A ideia aqui é obter o máximo de *feedback* possível, com a finalidade de saber se o usuário possui alguma dificuldade para utilizar o sistema.

Os testes foram realizados com cinco participantes, que foram escolhidos com base em cada perfil de usuário do sistema. Essa amostragem de cinco participantes, conforme Nielsen (1994)[7], é capaz de identificar aproximadamente 70% dos problemas de interface mais críticos. Assim, foram selecionados, dois pastores, um presbítero responsável por manter a atualização das informações dos membros, um membro sem atribuições e um visitante.

Os pastores tinham as seguintes missões: fazer login no sistema, modificar a descrição da igreja, consultar um usuário no rol de membros, tornar um usuário como gerente do sistema, criar um evento recorrente na igreja, criar criar um evento não recorrente na igreja, adicionar um membro em um grupo e por último tornar o membro adicionado como administrador do grupo.

O presbítero responsável pelo rol de membros tinha as seguintes missões: fazer login no sistema, consultar membros no rol de membros, cadastrar um usuário, emitir uma lista com todos os membros da igreja, emitir uma lista com todos os membros da igreja que tenham menos de 30 anos de idade, criar o grupo de música da igreja, adicionar dois membros no grupo criado e promover um dos membros com administrador do grupo.

Com relação aos últimos papéis, sendo estes, os membros sem nenhuma atribuição tem os seguintes objetivos: fazer login no sistema, consultar seus dados pessoais, atualizar seu endereço, solicitar entrada em um grupo qualquer e acessar o grupo, e o visitante que tem apenas duas missões: fazer login no sistema e atualizar seu endereço na plataforma.

2.6. Resultados e Discussões

Nesta seção, serão apresentados os resultados dos testes com os cinco usuários previamente escolhidos.

[12] PostgreSQL: https://www.postgresql.org/
[13] Npgsql: https://www.npgsql.org/
[14] Quartz: https://www.quartz-scheduler.net/
[15] NUnit: https://learn.microsoft.com/pt-br/dotnet/core/testing/unit-testing-with-nunit
[16] Vue Js: https://vuejs.org/
[17] Material Design: https://m2.material.io/
[18] Vuetify: https://vuetifyjs.com/en/
[19] vue-property-decorator: https://www.npmjs.com/package/vue-property-decorator
[20] momentJs: https://momentjs.com/
[21] jwt_decode: https://www.npmjs.com/package/jwt-decode
[22] vue-toast-notification: https://www.npmjs.com/package/vue-toast-notification
[23] vue-the-mask: https://www.npmjs.com/package/vue-the-mask
[24] vue-google-oauth2: https://www.npmjs.com/package/vue-google-oauth2
[25] vue-cropperjs: https://www.npmjs.com/package/vue-cropperjs
[26] vuex: https://vuex.vuejs.org/
[27] vou-router: https://router.vuejs.org/
[28] axios: https://axios-http.com/ptbr/docs/intro

Baseado em seus relatos foi possível entender quais as melhores formas de melhorar a experiência de uso do sistema e também estudar os *feedbacks* com propostas de desenvolver novas funcionalidades.

Os testes foram iniciados com os dois pastores convidados. A primeira missão, fazer login no sistema, foi realizada sem dificuldades por ambos. Pode-se destacar, por exemplo, o seguinte relato: “Essa ideia de usar a conta google é muito útil, pois a maioria dos meus acessos e programas eu faço pelo google, o que me ajuda a não precisar lembrar da senha; isso é muito mais fácil”.

Na segunda missão de modificar a descrição da igreja, os usuários encontraram dificuldades em realizá-la, devido a falta de clareza na descrição do botão que estava como “Alterar Grupo”, então a tendência era ignorar esse botão e clicar no nome “Igreja”, localizado na tela principal, ou no nome “Igreja Presbiteriana Central de Mossoró”, no menu de topo.

Quanto à missão “consultar o usuário chamado ‘Dylan Oliveira’ no rol de membros”, foi executada sem problemas pelos pastores.

A quarta missão, de tornar o usuário com o nome “Dylan Oliveira” como um gerente do sistema, houve uma dificuldade inicial por parte de um usuário que não entendeu que deveria clicar em “Alterar Dados” antes de realmente atualizar o cadastro, e, após a orientação, o processo ocorreu normalmente.

A quinta missão, de criar um evento não recorrente na igreja, foi cumprida sem muitas dificuldades. O objetivo de criar um evento recorrente na igreja, por sua vez, houve uma dificuldade de associar que o mesmo botão “Criar Aviso” serve para avisos recorrentes e não recorrentes, além disso, ambos os pastores tiveram problemas para entender como ocorria a frequência do evento e sugeriram algo semelhante ao google calendar, que traz opções como “toda quarta”.

As últimas missões, de adicionar um membro no grupo da União Presbiteriana de Homens (UPH) e torná-lo administrador do grupo, foram concluídas por ambos os usuários sem dificuldades.

O próximo teste a ser realizado foi com o presbítero responsável pelo rol de membros, cuja primeira etapa é a de logar no sistema, que foi completada com sucesso.

Na segunda missão, a de consultar integrantes no rol de membros, o usuário passou algum tempo navegando na tela, mas localiza a função corretamente.

Na missão de cadastrar um usuário, o presbítero usou primeiramente a função de pesquisar pessoas, mas logo se corrigiu e também cumpriu o objetivo.

As missões de “emissão de relatórios de todos os membros” e “apenas membros com menos de 30 anos”, foram executadas corretamente, apesar de uma breve confusão no começo para identificar o botão “Imprimir Lista”, que fica na tela de rol de membros.

Na missão de criar o grupo de música, o presbítero tentou inicialmente o botão de “Alterar Grupo”, mas, falhando, foi dada a orientação de acessar a aba de grupos, então ele conseguiu criar o grupo. Antes de finalizar a criação do grupo, o participante tentou adicionar os membros na própria tela de criação, e, não havendo essa opção, ele finalizou e adicionou posteriormente.

Sua última missão de promover um dos membros como administrador do grupo só foi possível realizar após orientação.

Tanto o membro sem atribuições como o visitante realizaram seus objetivos sem nenhuma dificuldade. Segue o pensamento em voz alta do visitante durante a missão de atualizar endereço: “eu vou no meu perfil, por que eu acho que é lá. Tem aqui no canto, minha foto, aqui ‘ver perfil’. Alterar meu endereço, né?! Pronto, já tá aqui ‘endereço’. Vou pesquisar aqui o meu CEP, já tá aqui nas sugestões. Agora vou botar o número que não completou. Complemento? não tem. Vou botar em ‘salvar’.”.

Após a realização dos testes com os participantes, foram levantados alguns pontos de melhorias para a evolução do software, tais como:

- O botão “Alterar Grupo” deve ter seu texto alterado para “Alterar” mais o nome do grupo, para facilitar o entendimento do usuário.
- Alguns usuários tentaram modificar o grupo clicando no seu título. Então vale considerar o desenvolvimento desta funcionalidade.
- Recorrentemente os usuários tentaram modificar os dados dos membros clicando em seu nome na aba de “Membros” do grupo ao invés de ir até “Rol de Membros”. Vale considerar o desenvolvimento dessa funcionalidade, ou deixar menos enfático a opção de "membros" do grupo.
- Não ficou muito claro para os usuários a ideia de que os dois tipos de avisos são criados no mesmo botão. Uma possibilidade é separar os botões, ou então evidenciar mais na tela a possibilidade de criar os dois tipos de avisos.
- Deixar mais claro a recorrência de um evento, para tal, pegar referências do google calendar.
- Verificar a viabilidade de colocar a adição de membros na tela da criação de um grupo.

Esses pontos devem ser avaliados para que sejam tratados e desenvolvidos da melhor maneira no sistema.

3. CONCLUSÕES

Após a análise dos dados e desenvolvimento do presente trabalho, foi possível identificar demandas da igreja local que não conseguem ser contempladas pelas ferramentas atualmente disponibilizadas pela IPB A partir dos testes realizados, foi verificado que o presente projeto tem potencial para ser implantado na IPCM e ainda cobrir demandas de outras congregações que apresentem necessidades similares.

Desse modo, com base no estudo apresentado, foi possível notar que o software tem capacidade para evoluir cada vez mais à medida que seu uso for colocado em prática pela comunidade, tendo em vista o seu potencial de escalabilidade. Todavia, em sua primeira versão, o sistema, como foi relatado nos testes, já consegue cumprir seu objetivo inicial de gerenciar os membros da igreja e comunicar de forma mais ativa sobre as programações agendadas. A Tabela 2 a seguir mostra os requisitos definidos no ponto 2.1 deste artigo e como eles foram validados durante a execução da metodologia de testes na seção 2.6.

Tabela 2. Relação dos requisitos e validações.

Requisito	Validação
Ser capaz de registrar membros, e também gerenciar seus dados.	Validado pelo presbítero durante a missão de cadastrar usuário
Possibilitar que o membro mantenha seus dados atualizados na plataforma	Validado pelo membro sem atribuição na missão de atualizar seu próprio endereço.
Obter Sub-listas da lista de membros	Validado pelo presbítero durante a missão de emitir uma lista com todos os membros da igreja que tenham menos de 30 anos de idade
Ter um espaço acessível para que todos possam visualizar as programações da igreja	Validado pelo presbítero durante as missões de criar grupos, adicionar membros no grupo e criar eventos

Fonte: autoria própria

Isso possibilita aumentar o nível de informatização como um todo, a começar pela IPCM. O software desenvolvido tem o potencial de facilitar a coleta de dados da instituição, já que o sistema busca atacar os pontos de dores que a equipe de rol de membros relatou. Com base nisso, espera-se que as pessoas que operam esse trabalho terão a facilidade e agilidade de utilizar um software atualizado e com funções específicas para suas demandas.

Além disso, a aplicação aqui desenvolvida promove o aumento do engajamento dos membros das igrejas em suas atividades como um todo, através das funcionalidades de notificações, quadro de avisos e tornando o acesso fácil para todas as idades. Ainda se tratando da escalabilidade, deve-se considerar que o sistema tem possibilidade de evoluir no quesito de funcionalidades como: bíblia virtual; hinário; controle de finanças; entre outras opções como: auto cadastro via Quick Response Code (QR Code); votar em enquetes; restrições para entrar em grupos e nível de acesso em grupos.

Do ponto de vista da Ciência da Computação, esse artigo busca trazer uma contribuição nas discussões sobre desenvolvimento de sistemas web, estruturas de dados, uma vez que, nesta aplicação em específico, o modelo de dados da entidade “Grupo” funciona baseado em “árvore” para que a aplicação seja escalável com sua ramificação. Além disso, ainda é abordado o tema da experiência do usuário, em que a aplicação em questão traz conceitos de interface acessível a pessoas da terceira idade, e, a partir dos testes, obtemos insumos sobre quais pontos funcionam e quais necessitam evoluir, como a posição de botões e menus. Se tratando da arquitetura, é reforçada a importância da organização em camadas, de modo a separar as responsabilidades de cada serviço, como é feio em aplicações modernas. Adiante, foi colocado em prova a contribuição do método *think aloud* na Computação, para entender como os usuários chegam a determinadas conclusões ao utilizar um sistema, sendo possível mapear seus pensamentos durante a experiência de uso.

Por fim, existe a possibilidade de expandir o conteúdo deste artigo com novos estudos sobre como melhorar o engajamento dos usuários na plataforma; como incentivar a atualização de dados pessoais, por exemplo, por meio de mecânicas de gamificação; e investigar em quais outras áreas da igreja o sistema poderia ser útil, por exemplo, em eleições internas.

4. REFERÊNCIAS BIBLIOGRÁFICAS

[1] Secretaria executiva do supremo concílio. Disponível em <https://www.executivaipb.com.br/estatisticas/> . Acesso em: 06 de março de 2023.

[2] iCalvinus - SC/IPB. Disponível em <https://sc.icalvinus.app/#/>. Acesso em: 06 de março de 2023.

[3] MARTIN, Robert C. Clean Code-Refactoring, Patterns, Testen und Techniken für sauberen Code: Deutsche Ausgabe. MITP-Verlags GmbH & Co. KG, 2013.

[4] The Principles of OOD. Disponível em <http://butunclebob.com/ArticleS.UncleBob.PrinciplesOfOod>. Acesso em: 11 de maio de 2023.

[5] LEHNHART, ELIETE DOS REIS; LÖBLER, Mauri Leodir; TAGLIAPIETRA, Rafaela Dutra. DISCUSSÃO E APLICAÇÃO DO PROTOCOLO THINK ALOUD EM PESQUISAS SOBRE PROCESSO DECISÓRIO. Revista Alcance, v. 26, n. 1, p. 13-29, 2019.

[6] VAN SOMEREN, Maarten; BARNARD, Yvonne F.; SANDBERG, J. The think aloud method: a practical approach to modelling cognitive. London: AcademicPress, v. 11, p. 29-41, 1994.

[7] NIELSEN, Jakob. Usability Engineering. Elsevier, 1994.