AMEM: SISTEMA DE GERENCIAMENTO DE EVENTOS DESENVOLVIDO PARA A PASTORAL UNIVERSITÁRIA DA UNIVERSIDADE LA SALLE

Pedro Henrique Ferreira da Silveira[1]

RESUMO

AMEM é a abreviatura de *Academic Ministry Event Manager*, nome escolhido para o software de gerenciamento de eventos desenvolvido especialmente para a pastoral universitária da Universidade La Salle. O software multiplataforma surgiu da necessidade de otimizar o registro e o controle de eventos realizados pela pastoral, incluindo a contabilização de doações recebidas e efetuadas durante os eventos, bem como o registro de horas complementares que serão atribuídas aos voluntários participantes. O sistema foi desenvolvido utilizando o framework React Native, o que garante seu funcionamento em três plataformas distintas: web, Android e iOS.

Palavras-chave: Software Multiplataforma; React Native; Gerenciamento de Eventos; Pastoral Universitária.

ABSTRACT

AMEM is the abbreviation for Academic Ministry Event Manager, the name chosen for the event management software developed especially for the academic ministry at La Salle University. The cross-platform software arose from the need to optimize the registration and control of events carried out by the pastoral, including accounting for donations received and made during the events, as well as recording additional hours that will be allocated to participating volunteers. The system was developed using the React Native framework, which guarantees its operation on three different platforms: web, Android and iOS.

Keywords: Cross-platform Software; React Native; Event Management; Academic Ministry.

1 INTRODUÇÃO

Para aqueles que gostam de aproveitar o final de semana com os amigos em uma boa lanchonete, não é algo tão incomum encontrar estabelecimentos que ainda utilizam o método mais clássico de controle de pedidos: o papel e a caneta. Na maioria das vezes, esse tipo de controle acaba não sendo muito eficiente, o que resulta em diversos problemas como atraso nos pedidos, confecção de pedidos incorretos, entre outros. Essa desorganização, por sua vez, pode causar prejuízos para o estabelecimento, desperdiçar recursos, além de decepcionar os clientes, fazendo com que percam o interesse de retornar ao local.

A engenharia de software surge como uma boa solução para esses problemas, afinal o ato de gerenciar se torna mais prático e eficiente quando o operador tem à sua disposição um sistema totalmente digital. Com um software de

[1] Discente do Curso Ciência da Computação da Universidade La Salle - Unilasalle, matriculado na disciplina Trabalho de Conclusão de Curso II. E-mail: pedro.202010311@unilasalle.edu.br, sob a orientação do Prof. Mozart Lemos de Siqueira. E-mail: mozart.siqueira@unilasalle.edu.br. Data de entrega: 05 de julho. 2024.

qualidade, o registro de informações apresenta dados previamente preenchidos, cálculos automaticamente efetuados e com mínima ou nenhuma margem para erros humanos.

Vale ressaltar que apesar de as lanchonetes terem sido mencionadas, a falta de um sistema digital pode afetar qualquer tipo de empresa, seja ela focada em administração, saúde ou lazer. Um bom exemplo disso é a Pastoral Universitária da Universidade La Salle, responsável por promover eventos acadêmicos com os mais variados objetivos, tais como a arrecadação de alimentos, roupas e outros utensílios para doação. Além disso, os eventos também proporcionam aos alunos a oportunidade de adquirir as horas complementares necessárias para sua formação, através das ações voluntárias.

Embora a Pastoral não realize o controle de seus eventos por meio de papel e caneta, mas sim através de planilhas, ela também enfrenta dificuldades no gerenciamento de seus eventos. Isso ocorre porque apesar de uma planilha ser um método digital, um software oferece diversas funcionalidades específicas que uma planilha não consegue reproduzir, sendo mais indicado para otimizar a gestão e aumentar a produtividade de um processo.

A ausência de um sistema centralizado para efetuar o controle de eventos, o registro de doações e o gerenciamento de voluntários acarreta em adversidades consideráveis. Entre os problemas enfrentados estão a perda e a duplicidade de informações, a dificuldade na contabilização e na filtragem dos dados, a falta de agilidade no processo de organização, entre outras questões que comprometem a capacidade de promover eventos de maneira eficiente e prática.

Com o intuito de solucionar este problema, será desenvolvido um software de gerenciamento de eventos que centralizará todas as informações relevantes e facilitará o registro e o controle de eventos, permitindo uma administração mais eficiente e prática de todas as doações e voluntários. A criação deste software terá um impacto significativo na comunidade acadêmica da Universidade La Salle, pois irá melhorar a organização e a eficiência da Pastoral Universitária. Essa melhoria, por sua vez, aumentará a capacidade da Pastoral de realizar eventos que fortalecem ações sociais e comunitárias.

Ao longo deste trabalho, o leitor encontrará uma análise detalhada de cada componente e tecnologia utilizada no desenvolvimento do AMEM. Inicialmente, a seção 2, Metodologia, apresenta o conceito de software cross-platform, explicando as razões para a escolha do React Native, NativeBase e Firebase, além de discutir suas principais características e funcionalidades. A seção 3, Solução, detalha os requisitos funcionais e não funcionais elicitados, bem como uma lista de interfaces do sistema e as funcionalidades de UX Design implementadas. Por fim, a seção 4, Considerações Finais, discute as limitações encontradas durante o desenvolvimento e detalha como foi o processo de validação do sistema.

2 METODOLOGIA

Inicialmente, é importante destacar que o software foi concebido com a ideia de ser *cross-platform*, um termo utilizado para categorizar sistemas que adaptam automaticamente suas funcionalidades e interfaces para funcionar de forma igualitária em diferentes plataformas, podendo ser web ou mobile:

> Há uma infinidade de alternativas ao desenvolvimento de software móvel nativo. No entanto, o desenvolvimento multiplataforma tornou-se uma escolha popular para empresas que procuram soluções econômicas para o

> desenvolvimento de software. O desenvolvimento multiplataforma permite que os desenvolvedores de software desenvolvam aplicativos para Android e iOS com um único código. (MALIK, 2021, p. 2).

Segundo Malik (2021), os benefícios deste modelo de software são vários: agilizar o processo de desenvolvimento, facilitar manutenções necessárias posteriormente, garantir maior abrangência de plataformas, etc. Com este conceito bem definido, em primeira instância, o foco será explicar como funcionam os softwares *cross-platform*, quais são as linguagens e tecnologias utilizadas para alcançar um resultado eficaz e totalmente funcional para o projeto, bem como quais as vantagens e desvantagens da sua utilização. Em seguida, todos os detalhes técnicos do AMEM serão detalhados, incluindo uma explicação concisa de suas funcionalidades.

Embora a ideia de existir um software capaz de se adaptar automaticamente para diferentes plataformas seja algo relativamente novo, atualmente já existem diversas bibliotecas capazes de executar esta funcionalidade com maestria. Para a composição do back-end do AMEM (estrutura contendo as funcionalidades internas do software), o framework escolhido foi o React Native, e há motivos suficientes para essa escolha.

2.1 React Native

O React Native é um framework desenvolvido pela Meta, empresa dona do Facebook, estando diretamente voltada para o desenvolvimento de aplicativos móveis. Baseado no React (uma biblioteca Javascript), ele aproveita várias de suas vantagens, assim como a divisão do código em componentes e o uso de JSX (integração do Javascript com elementos nativos). Por ser uma ferramenta de desenvolvimento *cross-platform*, o React Native permite que o mesmo código-fonte seja utilizado para distribuir a aplicação tanto em iOS quanto em Android, além de se adaptar perfeitamente para web (necessitando apenas instalar o pacote react-native-web).

A principal característica que diferencia o React Native de outras ferramentas *cross-platform* é sua capacidade de gerar os aplicativos finais com componentes nativos das plataformas, e isso só é possível porque:

> Ao invés de executar o React no navegador e renderizar divs e textos [...], o React Native é executado em uma instância incorporada de JavaScriptCore (iOS) ou V8 (Android) dentro dos aplicativos e renderiza para componentes específicos da plataforma de nível superior. Os componentes JavaScript são declarados usando um conjunto de primitivas integradas suportadas por componentes iOS ou Android. (DANIELSSON, 2016, p. 46).

Conforme dito por Danielsson (2016), o React Native executa o código em uma instância embutida do JavaScriptCore (para iOS) ou V8 (para Android) dentro das aplicações. Sendo assim, o código é escrito com a utilização de componentes do React Native, e quando a aplicação final é gerada, o próprio framework se encarrega de converter esses componentes para elementos nativos da plataforma de destino. Essa abordagem, embora apresente uma renderização mais lenta em comparação com a escrita do código utilizando elementos propriamente nativos, resulta em uma melhor experiência do usuário e um desempenho superior em comparação a outras soluções.

2.2 NativeBase

Para a composição do front-end do AMEM (estrutura de elementos visuais do software), foi utilizado o pacote NativeBase. Este pacote oferece uma solução simples e prática, aproveitando componentes padrão do React Native para disponibilizar uma ampla variedade de elementos prontos, o que facilita significativamente o processo de criação de layouts. O NativeBase inclui diversos componentes como caixas de seleção, botões rádio, mensagens toast, auxiliares tooltip, além de uma vasta gama de outros elementos que, caso fossem criados diretamente no React Native, demandariam muito mais tempo e esforço.

A principal vantagem de utilizar NativeBase é a aceleração do processo de desenvolvimento. Com uma biblioteca rica em componentes predefinidos e estilizados, é possível focar mais na lógica e na funcionalidade do software, sem se preocupar tanto com a criação e estilização minuciosa de cada elemento da interface. Além disso, o NativeBase oferece consistência visual e responsividade, mantendo o aspecto *cross-platform* do React Native.

2.3 Firebase

Para o banco de dados da aplicação, a tecnologia escolhida foi o Firebase. Uma das principais vantagens do Firebase é o fato de ser um banco de dados *cloud*, o que possibilita a fácil integração com as plataformas web e mobile. A configuração necessária para essa integração é simples e pode ser realizada diretamente no código-fonte. Outra característica interessante do Firebase é o fato de ser um banco de dados *Not Only SQL* (NoSQL), o que é recomendado para bancos armazenados em nuvem:

> Os bancos de dados relacionais não são adequados para ambientes em nuvem porque eles não suportam pesquisas Full Content Data e é difícil escaloná-los além de um limite. No entanto, os bancos de dados NoSQL são a melhor solução para bancos de dados em nuvem porque todas as características que definem os bancos de dados NoSQL são muito desejáveis para bancos de dados em nuvem. (MOHAMED et al., 2014, p. 600).

Segundo Mohamed et al. (2014), os bancos de dados cloud devem proporcionar disponibilidade, escalabilidade, desempenho e flexibilidade, além de serem capazes de lidar com dados não estruturados, semiestruturados ou estruturados. Sendo assim, os bancos de dados do tipo NoSQL cumprem todos os requisitos.

O Firebase é segmentado entre vários módulos, os quais oferecem diferentes tipos de operações com dados. O módulo Authentication do Firebase proporciona uma forma segura e eficiente para gerenciar a autenticação e a criação de usuários, suportando diversos métodos de login, assim como o método de login via email e senha, utilizado pelo AMEM. Além disso, o próprio módulo oferece opções de envio de e-mail para recuperação de senha. Todas essas funcionalidades pré-programadas facilitam muito a criação do software.

Outro módulo super importante é o Realtime Database, que por sua vez oferece uma solução robusta para o armazenamento de dados em tempo real. Ele permite que os dados sejam sincronizados instantaneamente entre todos os clientes conectados, proporcionando uma experiência de usuário fluida e responsiva. Este

recurso é particularmente útil para aplicações que exigem atualizações constantes e imediatas, assim como o gerenciamento de eventos e doações que ocorrem no AMEM.

3 SOLUÇÃO

Nesta seção serão apresentados os requisitos funcionais e não funcionais considerados para o desenvolvimento do AMEM, além de uma exibição completa contendo todos os detalhes técnicos, incluindo os benefícios e limitações que são atrelados ao desenvolvimento de software *cross-platform*.

3.1 Requisitos

Os requisitos foram definidos após um processo de elicitação de requisitos realizado com o pastor Felipe Almeida, responsável pela administração da Pastoral Universitária, a fim de garantir que o sistema atendesse completamente às necessidades identificadas. Sendo assim, conforme o que foi apontado durante a reunião, foram considerados os seguintes requisitos:

Quadro 1 - Requisitos funcionais

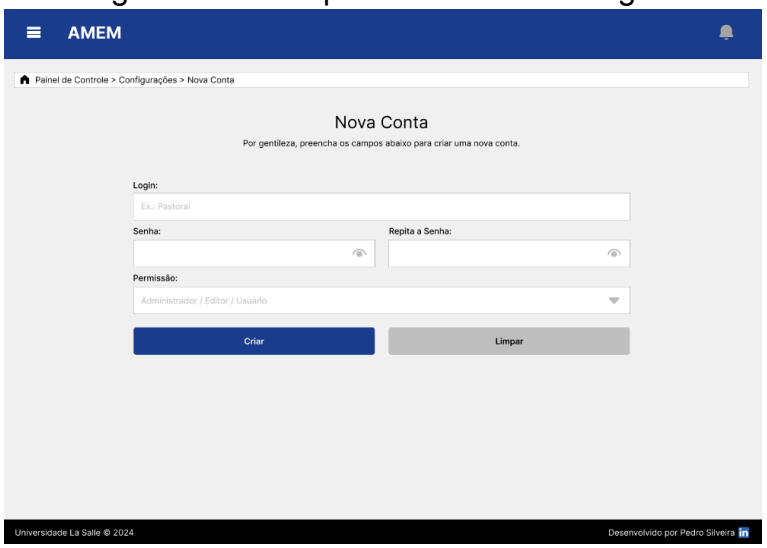
Identificação	Descrição
RF001	Exibir informações estatísticas sobre os eventos, incluindo a contabilização de doações recebidas e de voluntários envolvidos.
RF002	Listar os eventos com filtros por nome, data e status.
RF003	Alertar para eventos próximos.
RF004	Cadastrar novos eventos (disparando um e-mail com os detalhes do evento para os usuários envolvidos).
RF005	Visualizar e editar as informações de um evento específico.
RF006	Cadastrar doações recebidas e doações efetuadas para cada evento.
RF007	Registrar voluntários que atuaram em cada evento.

Fonte: elaborado pelo autor.

Além dos requisitos listados, foi criado um mockup contendo o protótipo inicial do layout das telas através do Figma, uma ferramenta amplamente utilizada por desenvolvedores de software. Vale destacar que o Figma é uma excelente solução para quem deseja desenvolver software utilizando métodos ágeis, pois um protótipo visual das telas proporciona uma melhor concepção do que deve ser desenvolvido.

Dessa forma, é possível evitar dois grandes problemas durante o desenvolvimento: atrasos na entrega do projeto devido ao tempo gasto na elaboração de extensas documentações e devido ao tempo necessário para compor as interfaces simultaneamente à escrita do código.

Figura 1 - Protótipo de tela feito via Figma



Fonte: elaborada pelo autor.

3.2 Interfaces

O software conta com 13 interfaces para a execução de todas as funcionalidades, sendo cada interface responsável por uma funcionalidade específica. A seguir, são descritas as telas e suas respectivas funcionalidades em detalhes:

Quadro 2 - Interfaces do software

Interface	Funções
Autenticação	Permite que os usuários façam login no sistema utilizando e-mail e senha; e Inclui uma opção para envio de e-mail de recuperação de senha, caso o usuário esqueça suas credenciais.
Controle de Eventos	Exibe uma lista de todos os eventos com opções de filtro por nome do evento, intervalo de datas e tipo ("encerrados" ou "planejados"); e Inclui um dashboard que contabiliza o total de eventos registrados, separados por "encerrados" e "planejados", além de mostrar a quantidade de doações e voluntários registrados.
Todas as Doações	Exibe uma lista contendo todas as

	doações de todos os eventos; e Inclui um campo para filtrar pelo nome do material doado ou pelo nome da organização, bem como um seletor para filtrar entre doações recebidas, efetuadas ou todas.
Todos os Voluntários	Exibe uma lista de todos os voluntários registrados em todos os eventos; e Inclui um filtro de busca pelo nome do voluntário para facilitar a localização.
Cadastrar Evento	Permite o cadastro de um novo evento, incluindo informações como nome do evento, data prevista, local, investimento inicial e observações; e Permite selecionar usuários cadastrados no sistema para enviar um e-mail com as informações do evento ao clicar em "cadastrar".
Detalhes do Evento	Exibe os detalhes de um evento específico ao clicar no item do evento na lista; Permite a edição das informações caso o usuário tenha permissão (administrador ou editor); Inclui opções para excluir o evento ou marcá-lo como "encerrado"; e Exibe listas de doações e voluntários registrados durante o evento, com botões para registrar novas doações e novos voluntários.
Detalhes da Doação	Exibe informações detalhadas de uma doação ao clicar no item da doação na lista; Permite a edição das informações caso o usuário tenha permissão (editor ou administrador); e Inclui um botão para excluir a doação.
Detalhes do Voluntário	Exibe informações detalhadas de um voluntário ao clicar no item do voluntário na lista; Permite a edição das informações caso o usuário tenha permissão (editor ou administrador); e Inclui um botão para excluir o voluntário.

Registrar Doação	Permite o registro de uma nova doação para o evento selecionado, incluindo detalhes como material da doação, quantidade, unidade de medida, tipo (efetuada ou recebida) e a organização responsável pela doação.
Registrar Voluntário	Permite o registro de um novo voluntário para o evento selecionado, incluindo informações como nome do voluntário, curso matriculado, e-mail, telefone e horas complementares que serão atribuídas.
Controle de Usuário	Acessível apenas por administradores, exibe uma lista de todos os usuários cadastrados no sistema; e Inclui filtros pelo nome do usuário e dispõe de um botão para cadastrar novos usuários.
Detalhes do Usuário	Permite visualizar as informações do usuário e editar o nível de permissão; e Inclui um botão para excluir o usuário.
Novo Usuário	Permite a criação de um novo usuário no sistema, incluindo informações como nome, e-mail, senha, confirmação da senha e nível de permissão; e Níveis de permissão disponíveis: "usuário" (apenas visualiza informações), "editor" (visualiza e edita informações de eventos), e "administrador" (tem acesso total para visualizar, editar e excluir informações, além de cadastrar novos usuários).

Fonte: elaborado pelo autor.

3.3 UX Design

Existem funcionalidades que, embora não sejam essenciais, agregam valor a um software de forma significativa. Muitas vezes, essas funcionalidades passam despercebidas pelos usuários, mas sua ausência certamente seria notada. Esse conceito está diretamente relacionado à experiência do usuário, também conhecida como UX Design. Sendo assim, em se tratando do AMEM, o sistema foca em proporcionar uma excelente experiência ao usuário, incorporando funcionalidades adicionais que ressaltam ainda mais o compromisso do sistema com a qualidade.

Quadro 3 - Funcionalidades do software

Funcionalidade	Descrição
Foco Automático em Formulários de Cadastro	Facilita a inserção de dados pelo usuário.
Gatilho de Foco pela Tecla Enter	Melhora a navegação e eficiência na entrada de dados.
Indicação Clara da Localização do Usuário	Facilita a navegação pelo sistema.
Exibição de Mensagens de Erro ou Confirmação	Aumenta a transparência e a usabilidade.
Filtros para Organização	Permite a visualização de dados de maneira organizada.
Três Níveis de Permissão	Garantem segurança e versatilidade na utilização do sistema.
Modais para Confirmar Ações Sensíveis	Previne ações não intencionais.
Interface Moderna e Responsiva	Proporciona uma navegação intuitiva e eficiente, com informações exibidas conforme a necessidade do usuário.

Fonte: elaborado pelo autor.

3.4 Limitações

Apesar das vantagens, a utilização de uma abordagem cross-platform como o React Native apresenta algumas limitações. Funcionalidades como o elemento hover, por exemplo, não são suportadas de forma consistente em todas as plataformas. Além do elemento hover, há diversas outras funcionalidades exclusivas de uma plataforma que não podem ser implementadas em outras.

Outro ponto importante é que a interface precisa ser projetada para funcionar tanto em ambientes web quanto mobile. Isso pode resultar em componentes que se adaptam bem a uma plataforma, mas não tão bem a outra. Por exemplo, a funcionalidade de retornar à página anterior teria sua melhor composição como um botão escrito “voltar” no final de uma página web, enquanto uma seta para a esquerda no menu de navegação seria mais adequada para uma visualização mobile. Essa característica acaba obrigando o desenvolvedor a optar por um layout que seja um meio-termo entre as plataformas.

Em relação ao uso do Firebase, existe uma limitação no número de registros que o banco de dados armazena, o que pode restringir o crescimento do sistema a longo prazo. Além disso, o Firebase Authentication trata as ações como se fossem executadas pelo próprio usuário, o que dificulta a implementação de um perfil administrador. Por exemplo, ao criar um usuário, o novo usuário é automaticamente autenticado, como se ele próprio tivesse criado a conta. Também só é possível editar ou excluir informações de uma conta se essa conta estiver autenticada no momento da ação.

Essa limitação existe por questões de segurança, afinal operações administrativas estariam sendo executadas diretamente no lado do cliente. Segundo o próprio Firebase (2024), existe um conjunto de bibliotecas que resolvem esse problema: o Firebase Admin SDK. Diferentemente do uso comum do Firebase, o Admin SDK funciona em um servidor, permitindo a execução de ações críticas de administração em um ambiente privilegiado. No entanto, a utilização do Admin SDK requer a existência de um aplicativo de servidor, o que seria inviável com o tempo disponível para o desenvolvimento do AMEM.

Por conseguinte, ao considerar o desenvolvimento de um software *cross-platform* com React Native e Firebase, é crucial avaliar essas limitações e planejar alternativas que possam mitigar seus impactos no desenvolvimento e operação do sistema.

4 CONSIDERAÇÕES FINAIS

A escolha do React Native foi motivada por sua facilidade de escrita e manutenção em um código-fonte único, mas também pela necessidade de criar uma aplicação *cross-platform* que proporcionasse a flexibilidade de utilizar o sistema no conforto do escritório, através do computador, ou até mesmo diretamente no recebimento de doações, através do celular. No entanto, encontramos algumas limitações inerentes a essa abordagem, como a necessidade de adaptar componentes para se comportarem de maneira adequada em ambas as plataformas.

Os testes e a validação do sistema foram conduzidos pelo pastor Felipe Almeida, responsável pela Pastoral Universitária. O pastor realizou testes manuais para avaliar o sistema, identificando possíveis melhorias e validando as funcionalidades implementadas. A avaliação considerou a eficiência, usabilidade e segurança do sistema, resultando em um feedback positivo.

Em suma, a experiência de desenvolvimento com React Native e Firebase foi enriquecedora e desafiadora, proporcionando um aprendizado significativo e um sistema funcional que atende às necessidades identificadas. Dado que o sistema resolve o problema proposto por este artigo e cumpre os requisitos especificados, o trabalho é considerado um sucesso.

Por fim, como projeções futuras, pretende-se buscar soluções para contornar as limitações encontradas durante o desenvolvimento do software, minimizando seus impactos e melhorando a funcionalidade do sistema. Ainda, caso seja de interesse da Universidade La Salle, existe o desejo de efetivar a implementação do sistema na Pastoral Universitária.

REFERÊNCIAS

DANIELSSON, William. React Native application development: a comparison between native Android and React Native. **Linköpings universitet**, Suécia, v. 10, n. 04, 2016.

MALIK, Kashif. Appsheet vs React Native: Evaluation of performance and development of Android Apps. **Metropolia University of Applied Sciences**, Finlândia, 2021.

MOHAMED, M; ALTRAFI, G; ISMAIL, O. Relational vs. NoSQL Databases: A Survey. **International Journal of Computer and Information Technology**, Sudão, v. 03, n. 03, 2014.

FIREBASE. **Adicionar o SDK Admin do Firebase ao servidor**. 2024. Disponível em: <https://firebase.google.com/docs/admin/setup>. Acesso em: 6 jun. 2024.