UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CURSO DE TECNOLOGIA EM SISTEMAS PARA INTERNET
CÂMPUS GUARAPUAVA

VINICIUS NADIN

NEXME! APLICATIVO PARA PROMOÇÃO DE EVENTOS E INTERAÇÃO SOCIAL

TRABALHO DE CONCLUSÃO DE CURSO

GUARAPUAVA
2017

VINICIUS NADIN

NEXME! APLICATIVO PARA PROMOÇÃO DE EVENTOS E INTERAÇÃO SOCIAL

Trabalho de Conclusão de Curso apresentado ao Câmpus Guarapuava da Universidade Tecnológica Federal do Paraná, como requisito parcial para a obtenção do título de Tecnólogo em Sistemas para Internet.

Orientador: Prof. Ms. Emerson André Fedechen

GUARAPUAVA
2017

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Guarapuava
Curso Superior de Tecnologia em Sistemas Para a Internet

ATA DE DEFESA DE MONOGRAFIA DE TRABALHO DE CONCLUSÃO DE CURSO DO CURSO DE TSI

No dia 27 de novembro de 2017, às 14:30 horas, nas dependências da Universidade Tecnológica Federal do Paraná Câmpus Guarapuava, ocorreu a banca de **defesa da monografia** de Trabalho de Conclusão de Curso intitulada: "**NEXME! Aplicativo para Promoção de Eventos e Interação Social**" do acadêmico **Vinicius Nadin** sob orientação do professor **Prof. Me. Emerson André Fedechen** do Curso de Tecnologia em Sistemas para Internet.

Banca Avaliadora	
Membro	**Nome**
Orientador	Prof. Me. Emerson André Fedechen
Coorientador	
Avaliador 1	Prof. Me. Guilherme Costa da Silva
Avaliador 2	Prof. Dr. Roni Fabio Banaszewski

Situação do Trabalho	
Situação	( ) Aprovado ( ) Aprovado com ressalvas ( ) Reprovado ( ) Não Compareceu
Encaminhamento do trabalho para biblioteca	( ) Pode ser encaminhado para biblioteca. ( ) Manter sigilo para publicação ou geração de patente.

Prof. Me. Emerson Andre Fedechen

Prof. Me. Guilherme Costa da Silva

Prof. Dr. Roni Fabio Banaszewski

Acadêmico: Vinicius Nadin

Guilherme da C. Silva

Roni Fabio Banaszewski

Vinicius Nadin

Guarapuava, 27 de novembro de 2017.

RESUMO

NADIN, Vinicius. NexMe! Aplicativo para Promoção de Eventos e Interação Social. 2017. 36 f. Trabalho de Conclusão de Curso – Câmpus Guarapuava, Universidade Tecnológica Federal do Paraná. Guarapuava, 2017.

Com o crescimento do mercado de aplicativos móveis no decorrer do tempo cresce também o mercado de aplicações móveis. Muitas aplicações passaram a disponibilizar a versão mobile garantindo maior praticidade aos seus usuários. Além disso, muitas aplicações passaram a disponibilizar funcionalidades exclusivas para a versão mobile, utilizando os recursos disponíveis em smartphones. Diante deste cenário e da observação do comportamento de usuários em redes sociais surgiu a ideia do aplicativo NexMe! para promoção de eventos e interação social. As principais funcionalidades do aplicativo são os perfis de usuários, cadastro e publicação de eventos e chat público e privado para proporcionar interação entre os usuários participantes de um evento. Identificou-se a necessidade de desenvolver o aplicativo para as plataformas Android, iOS e também a versão web, acessível por um navegador, levando em consideração as peculiaridades e importância de cada base de usuários. Sendo assim o objetivo do presente trabalho foi desenvolver a versão para a plataforma iOS do aplicativo NexMe!, para isso foi feita a análise das tecnologias e ferramentas de desenvolvimento disponíveis para essa plataforma e posteriormente a modelagem e implementação da aplicação.

Palavras-chave: Aplicativos Móveis. Promoção de Eventos. Interação Social. iOS.

ABSTRACT

NADIN, Vinicius. NexMe! Event Promotion and Social Interaction App. 2017. 36 f. Trabalho de Conclusão de Curso – Câmpus Guarapuava, Universidade Tecnológica Federal do Paraná. Guarapuava, 2017.

With the growth of the mobile applications market over time also grows the market for mobile applications. Many applications now available to mobile version ensuring greater convenience to its users. In addition, many applications now provide unique features to the mobile version using the features available on smartphones. Given this scenario and the observation of user behavior on social networks came the NexMe app! idea, to event promotion and social interaction. The application features are user profiles, registration and publication of events and public and private chat to provide interaction between users participating in an event. It identified the need to develop the application for the Android platform, iOS and also the web version, accessible by a browser, taking into account the peculiarities and importance of each user base. Thus the objective of this study was to develop a version of NexMe! application for the iOS platform , for this analysis technologies and development tools available for that platform and then the modeling and application implementation will be done.

Keywords: Mobile Apps. Event Promotion. Social Interaction. iOS.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

IDE	Integrated Development Environment
JSON	JavaScript Object Notation
MVC	Model-View-Controller
MVP	Model-View-Presenter
MVVM	Model-View-ViewModel
RAD	Rapid Application Development
RX	ReactiveX Framework
SGBD	Sistema de Gerenciamento de Banco de Dados
UX	User Experience

SUMÁRIO

1 INTRODUÇÃO

Pode-se considerar que a utilização de smartphones vem crescendo nos últimos anos. Muitas atividades que antes eram feitas exclusivamente pelo computador atualmente podem ser feitas pelo smartphone. O mercado de aplicativos também está em constante expansão. De acordo com pesquisa elaborada pela Flurry, ferramenta de mobile analytics da Yahoo, o consumo de aplicativos aumentou em 58% em 2015 (KHALAF, 2016).

Desta forma, acabou-se criando muitas oportunidades de negócios antes inexistentes, que no cenário atual são viáveis, ou seja, aplicações pensadas para serem utilizadas especificamente em um smartphone, devido aos recursos que este tipo de dispositivo oferece, como por exemplo, mobilidade, rapidez de acesso, geolocalização, conectividade constante, entre outros recursos.

Um exemplo desse novo cenário é a popularização de aplicativos de redes sociais, como Instagram e Snapchat, que utilizam recursos do smartphone (gps, câmera) e possuem funcionalidades que fazem mais sentido se utilizadas diretamente do próprio smartphone, como por exemplo, postagem de fotos e vídeos em tempo real, que nestas redes sociais são chamados de stories e snaps, rescpectivamente, focando na praticidade e agilidade, visto que seria de certa forma inviável utilizar tais funcionalidades em outros dispositivos como laptops, dada a faciliidade e alta disponibilidade de um smartphone no cotidiano das pessoas. Além disso, outras aplicações estão apostando nessa tendência de tornar a utilização mais rápida e instantânea, um exemplo disso é a funcionalidade do Facebook de transmissão ao vivo, que permite ao usuário fazer uma transmissão ao vivo para a sua rede de amigos. Essa nova tendência pode ser associada ao caráter imediatista que a sociedade vem adquirindo com o decorrer do tempo.

Diante deste cenário, embasado em observações sobre comportamento dos usuários e formas de utilização das redes sociais já consolidadas, surgiu a ideia de desenvolver um aplicativo para promoção de eventos e interação social, que centralize algumas funções já disponíveis em outras aplicações, em um ambiente focado para rápido acesso e praticidade ao usuário. Desta forma, surgiu o aplicativo NexMe!, um projeto de estudantes do curso de Sistemas para Internet, que foi hospedado no Hotel Tecnológico da UTFPR Câmpus Guarapuava.

Sendo assim o objetivo do presente trabalho foi desenvolver a versão para a plataforma iOS do aplicativo NexMe!, fazendo um estudo das tecnologias e ferramentas disponíveis para esta plataforma, e posteriormente a modelagem e implementação do aplicativo.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

O objetivo do trabalho presente é desenvolver a versão para plataforma iOS do aplicativo de promoção de eventos e interação social NexMe!

1.1.2 Objetivos Específicos

- Desenvolver o módulo de promoção de eventos do aplicativo NexMe! para plataforma iOS;
- Desenvolver o módulo de chat público nos eventos do aplicativo NexMe! para plataforma iOS.
- Desenvolver o módulo de chat privado entre usuários do aplicativo NexMe! para plataforma iOS.
- Desenvolver o módulo de seguir perfis de usuários do aplicativo NexMe! para plataforma iOS.

1.2 JUSTIFICATIVA

A utilização de smartphones para realização de diversas tarefas inseridas no cotidiano tem aumentado. Muita facilidade e praticidade pode ser extraída ao se utilizar os recursos e tecnologias disponíveis nos aparelhos para prover algum tipo de serviço ao usuário. Neste sentido, facilitar uma tarefa cotidiana ou recorrente do usuário, como por exemplo, descobrir quais eventos estão acontecendo em sua região em determinada data, pode se tornar um fator bastante relevante.

Segundo estatísticas de uso de smartphones no Brasil, aproximadamente 73% dos brasileiros que possuem um smartphone, não saem de casa sem ele e para os jovens é considerado o item mais importante a ser levado à um evento, à frente inclusive de documentos e dinheiro (OPUS, 2017).

Sendo assim, justifica-se o desenvolvimento do aplicativo proposto, devido ao crescimento constante da base de usuários de smartphones, bem como, devido à sua proposta principal, de centralizar a função de busca e divulgação de eventos em um aplicativo específico para este fim, além da possibilidade de interação entre os usuários participantes do evento.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão apresentados conceitos das tecnologias utilizadas para realização do projeto, com intuito de adquirir maior conhecimento teórico sobre estas tecnologias e também proporcionar uma visão mais abrangente sobre como o projeto foi executado.

2.1 APLICATIVOS MÓVEIS

Um aplicativo móvel ou aplicação móvel é um sistema desenvolvido para ser instalado em um dispositivo eletrônico móvel, como tablets e smartphones. Os aplicativos são normalmente conhecidos como "apps" ou "app mobile". A sigla "app" é uma abreviatura de "aplicação de software". Em 2010, o termo se tornou tão popular que foi assinalado como "palavra do ano" pela Sociedade Americana de Dialeto (FERNANDES, 2017).

Existem diversos tipos de aplicativos, que podem ser classificados por exemplo, como aplicativos de serviços, que fornecem informações e conteúdo de modo simplificado e ágil, aplicativos de informações, que dão acesso à conteúdos atualizados em tempo real ou que têm utilidade permanente, como guias de compras/lojas, telefones úteis, promoções, entre outros. Podem ser também aplicativos de comunicação, que permitem a conexão entre pessoas, como o Skype, ou aplicativos de integração como as redes sociais. E também aplicativos de entretenimento, destinados para diversão. A indústria de jogos é a que tem maior faturamento entre todos os segmentos do entretenimento (PORTO, 2017).

Existe diferença entre um aplicativo móvel e um site responsivo. Um aplicativo móvel exige uma instalação, um site responsivo é um site que possui um layout que se adapta ao tamanho de tela do dispositivo, podendo alterar posição de elementos entre outras características, não exigindo qualquer tipo de instalação. Porém, um site responsívo, por exemplo, depende de conectividade constante para seu devido funcionamento, fator que pode ser contornado ao se utilizar um aplicativo, que pode fornecer alternativas para momentos em que o usuário estiver sem conexão com a internet.

Apesar dos sites responsivos serem adaptáveis, a experiência dos usuários não costuma ser eficiente, fazendo com que ele acabe desistindo de navegar pelo site em seu dispositivo e, assim, a conversão acaba não sendo atingida conforme o esperado(FERNANDES, 2017).

Com o aplicativo móvel a experiência do usuário é muito mais positiva, pois o aplicativo pode ser moldado de várias formas para melhorar esta experiência. Por essa razão, muitas empresas têm aderido ao uso de aplicativos próprios para celulares, pois dessa forma o cliente fica mais próximo da sua marca (FERNANDES, 2017).

2.2 PLATAFORMAS MÓVEIS

Existem diversas plataformas móveis e estas possuem seus próprios sistemas operacionais. Estes sistemas operacionais podem rodar em dispositivos próprios, como é o caso do sistema operacional iOS da Apple, que roda apenas em hardware da própria Apple, ou podem rodar em dispositivos diversos, como é o caso do sistema operacional Android da Google, que roda no hardware da Google mas também em dispositivos de terceiros.

Os principais sistemas operacionais existentes e que possuem o maior market share são o iOS da Apple, e o Android da Google. Porém, existem outros sistemas operacionais menos utilizados, como por exemplo o Windows Phone da Microsoft, e o BlackBerry OS da BlackBerry (DUNN, 2017).

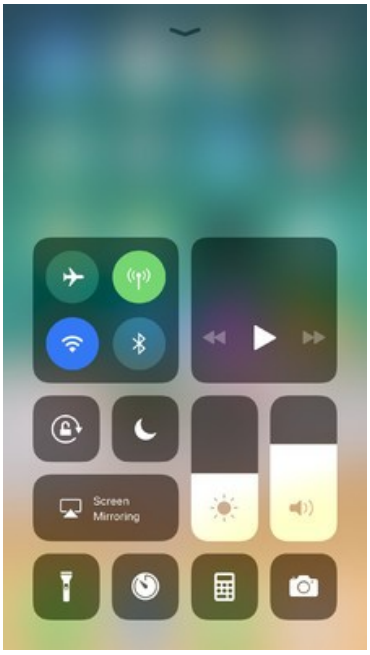
No presente trabalho será abordado mais detalhadamente o sistema operacional iOS, em virtude de ser a plataforma escolhida para o desenvolvimento do projeto.

2.2.1 Sistema Operacional iOS

iOS é o sistema operacional móvel da Apple, desenvolvido inicialmente para rodar no iPhone e posteriormente ganhando versões para outros dispositivos móveis da empresa. A empresa Apple só permite que este sistema seja instalado em hardware próprio, conforme lista de dispositivos compatíveis com o sistema, contemplando somente dispositivos iPhone, iPad e iPod, todos da própria Apple (APPLE, 2017b).

Anualmente são lançadas as versões principais do sistema, que se encontra atualmente na versão 11. O sistema vem com alguns aplicativos nativos e também possui uma loja de aplicativos, que por muitos desenvolvedores é considerada como tendo políticas mais rígidas para publicação de aplicativos, devido a um controle de qualidade mais rígido aplicado pela empresa, contemplando diversas etapas para publicação de um aplicativo. Também existe grande preocupação com proteção da propriedade intelectual dos aplicativos publicados em sua loja (CABRAL, 2011).

Figura 1 – Central de Controle iOS 11



Fonte: Apple (2017a)

2.3 LINGUAGENS DE PROGRAMAÇÃO

Diversas são as linguagens de programação utilizadas pelas plataformas móveis. Algumas plataformas utilizam linguagens mais generalistas, como por exemplo a plataforma Android que utilizada linguagem Java, outras plataformas utilizam linguagens específicas e proprietárias, como por exemplo a plataforma iOS, que utiliza as linguagens Objective C e Swift, linguagens estas que serão abordadas mais detalhadamente no presente trabalho, por fazerem parte da plataforma escolhida e terem sido utilizadas para desenvolvimento do aplicativo proposto.

2.3.1 Linguagens de Programação Plataforma iOS

As linguagens de programação utilizadas pela plataforma iOS são Objective C e Swift, sendo a primeira a mais antiga e ainda a mais utilizada, e a segunda, mais recente, considerada por muitos desenvolvedores mais moderna e flexível, devido algumas de suas características como ser *type safe*, usar conceito de *optionals* para tratar valores nulos, possuir melhor gerenciamento de memória, entre outras características (CISSOTO, 2015).

2.3.1.1 Linguagem Objective C

A principal linguagem de programação utilizada para desenvolvimento de aplicativos para a plataforma iOS é Objective C, uma linguagem que foi criada baseada na sintaxe da linguagem C ANSI e nos principais conceitos de Smalltalk, uma das primeiras linguagens

orientadas a objeto, desta forma possuir todos os recursos necessários para dar suporte ao desenvolvimento orientado a objetos (LECHETA, 2016).

Esta linguagem foi criada na década de 1980 por Brad Cox e Tom Love, sendo uma camada muito fina construída sobre a linguagem C. Atualmente é popularmente chamada de ObjC ou Obj-C e traz alguns recursos voltados para o paradigma funcional, como por exemplo, passar funções como parâmetro para outras funções, entre outros recursos.

2.3.1.2 Linguagem Swift

Outra linguagem utilizada no desenvolvimento para a plataforma iOS é a linguagem Swift, uma linguagem relativamente recente, e por ser nova, ainda possui espaço para pequenas mudanças que podem alterar a forma de desenvolver uma aplicação, no entanto já é considerada madura o suficiente para que novos aplicativos sejam criados com ela (JARDIM; SILVEIRA, 2016).

Por ser uma linguagem nova, foram implementados conceitos amplamente utilizados na comunidade de desenvolvedores. Esta linguagem apresenta uma sintaxe simples e moderna, seu objetivo é deixar o desenvolvimento de aplicativos mais produtivo (LECHETA, 2016).

Swift foi desenvolvida pela própria Apple e está ganhando cada vez mais espaço e sendo cada vez mais utilizada, inclusive por grandes aplicações, como por exemplo LinkedIn, Airbnb, Lyft, entre outros, que já possuem código Swift em suas aplicações.

De acordo com a própria Apple, aplicativos criados em Swift são mais rápidos e dinâmicos. Um algoritmo de busca, por exemplo, obtém resultados muito mais rápidos, chegando a ser até 2,6 vezes mais rápido que Objective C e 8,4 vezes mais rápido que Python 2.7 (APPLE, 2017c).

2.4 AMBIENTES DE DESENVOLVIMENTO

Um ambiente de desenvolvimento integrado, ou IDE (Integrated Development Environment), é um conjunto de ferramentas que trabalham de forma integrada para auxiliar no desenvolvimento, com intuito de agilizar tal processo.

Durante o desenvolvimento de uma aplicação mobile é fundamental otimizar todo tipo de processo, desde a escrita do código até sua execução. Quando o processo de desenvolvimento é otimizado, o desenvolvedor atinge maior produtividade e despende energia com o que é realmente necessário: a lógica da aplicação que está sendo criada, bem como, sua aplicabilidade com os problemas e necessidades da vida real (SCUDERO, 2017).

Geralmente os IDEs possibilitam técnicas de RAD (Rapid Application Development) que visam maior produtividade dos desenvolvedores (COLLETA, 2017).

Os IDEs contemplam escrita, teste, compilação e execução de código de maneira rápida e correta. Estes ambientes possuem vários recursos integrados, sendo os mais comuns, um editor de texto para escrita do código, um compilador, um depurador, além de mecanismos

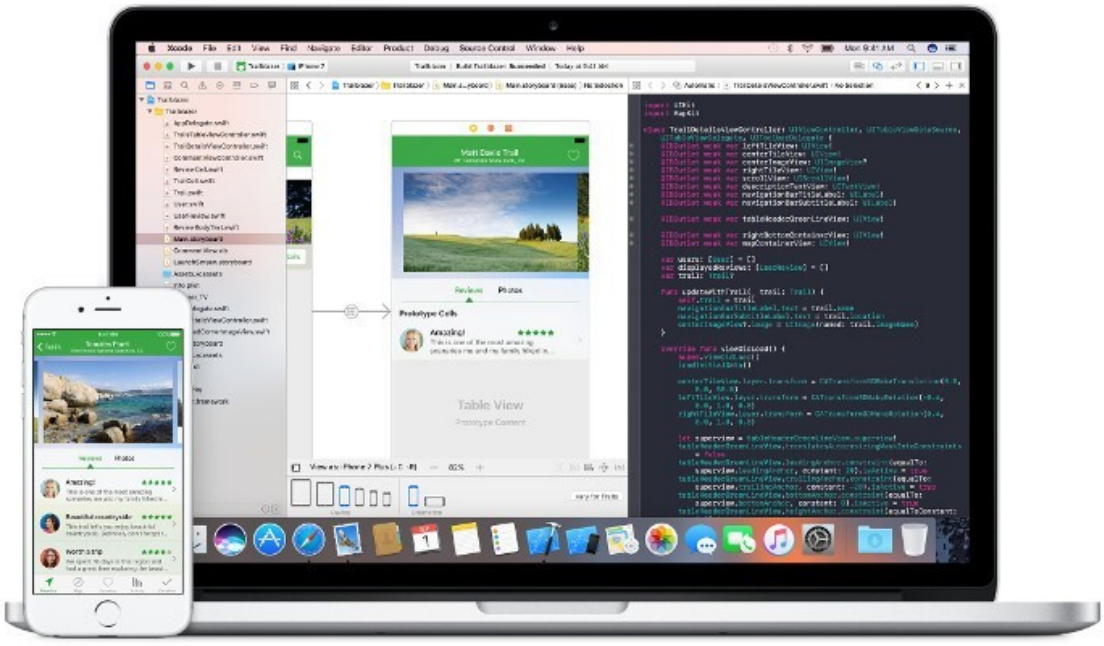
de distribuição que auxiliam no processo de criação de instalador do aplicativo. Desta forma, é necessário que o desenvolvedor tenha domínio sobre a IDE utilizada, o que impactará na agilidade do desenvolvimento, bem como, na qualidade final da aplicação.

Entre os IDEs mais conhecidas para desenvolvimento de aplicativos móveis, estão o Android Studio, para desenvolvimento de aplicativos para a plataforma Android, e o Xcode, para desenvolvimento de aplicativos para plataforma iOS, que será abordado em maiores detalhes no presente trabalho devido ter sido utilizado para desenvolvimento da aplicação.

2.4.1 IDE Xcode

O Xcode é o IDE (Ambiente de Desenvolvimento Integrado) oficial da Apple, somente disponível para rodar no sistema operacional Mac Os. No Xcode é possível optar por utilizar tanto a linguagem Objective C quanto a linguagem Swift, este IDE também possui um simulador integrado para dispositivos iPhone e iPad com intuito de facilitar e agilizar o desenvolvimento (LECHETA, 2016).

Figura 2 – IDE Xcode



Fonte: Apple (2016b)

2.5 BANCO DE DADOS

Aplicativos móveis podem usar diversos tipos de SGBD (Sistema de Gerenciamento de Banco de Dados), desde os tradicionais, como por exemplo o MySQL, PostgreSQL, SQL Server, ou até mesmo o SQLite, uma biblioteca que implementa um banco de dados SQL

embutido na aplicação, e que é amplamente utilizado em aplicações pequenas. A utilização do SGBD apropriado estará dependente das necessidades específicas de cada aplicação.

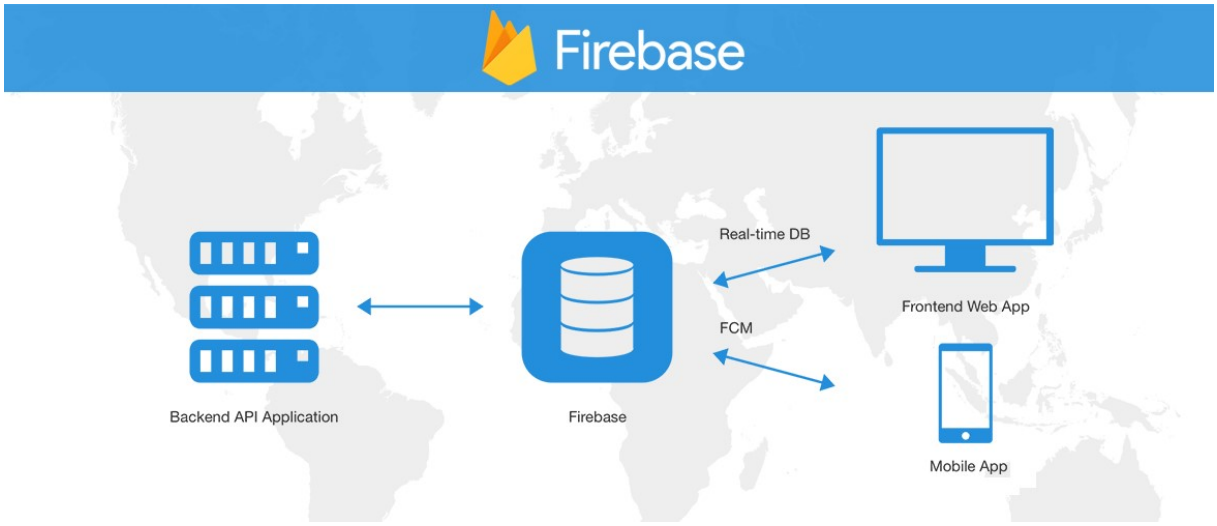
Uma outra alternativa à arquitetura tradicional é utilizar um Realtime Database, como por exemplo o serviço oferecido pela plataforma Firebase da Google, que será abordada em maiores detalhes adiante.

2.5.1 Firebase Realtime Database

A plataforma Firebase vai além de um simples banco de dados, esta plataforma fornece diversos outros serviços úteis, como, por exemplo, autenticação, mensagens, armazenamento, configurações remotas, testes, reporte de erros, entre outros, porém uma das suas funcionalidades mais interessantes se trata do Realtime Database.

Realtime Database é uma base de dados na cloud em que os dados são guardados utilizando JSON (Javascript Object Notation) e a medida em que os dados são modificados todas as aplicações clientes, sejam Web, Android ou iOS, são atualizadas instantaneamente, sendo necessário apenas a estruturação de como se quer guardar os dados (MUNGOI, 2016).

Figura 3 – Firebase Realtime Database



Fonte: Ficode (2017)

2.6 PADRÕES DE ARQUITETURA

A arquitetura a ser utilizada em uma aplicação é de suma importância e influencia diretamente no resultado final. Durante a fase de projeto é necessário tal preocupação com a arquitetura a ser utilizada, pois nesta fase, o foco está nas tecnologias utilizadas e elementos estruturais do projeto, diferente da fase de análise onde o foco é esboçar o problema a ser resolvido.

A arquitetura de software de um sistema computacional pode ser definida como as suas estruturas, que são compostas de elementos de software, de propriedades externamente visíveis desses componentes, e do relacionamento entre eles. Ou seja, a arquitetura define os elementos de software e como eles interagem entre si (MEDEIROS, 2017).

Diversos são os padrões arquiteturais existentes e por muitas vezes é necessário combinar elementos de mais de um padrão para atender às necessidades de um projeto. Desta forma, é importante conhecer as características básicas de cada padrão para selecionar o mais adequado ou combinar as características de mais de um padrão. Entre os principais padrões de arquitetura estão P2P (Peer to Peer), Arquitetura de Camadas, Arquitetura de Dados Compartilhados, MVC, MVP e MVVM, sendo este último padrão o utilizado no presente projeto e que será abordado em maiores detalhes.

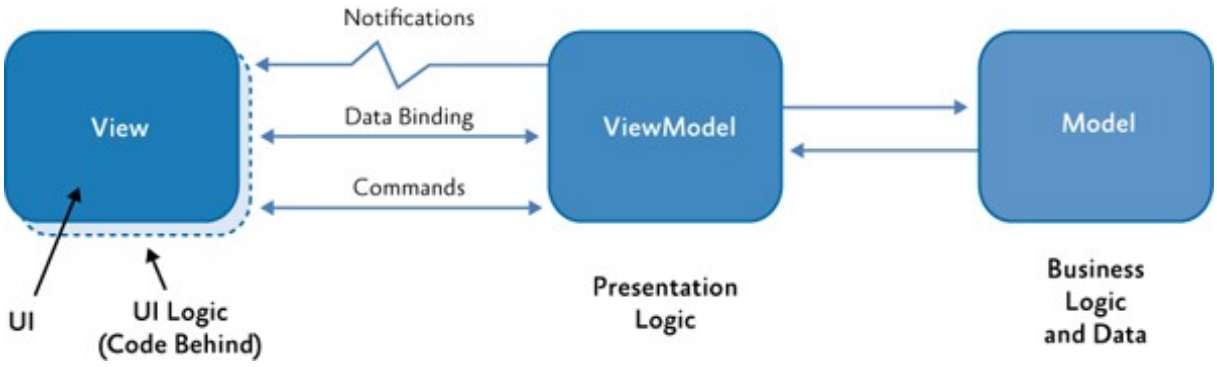
2.6.1 Padrão Arquitetural MVVM

O MVVM (Model-View-ViewModel) é um padrão que foi criado em 2005 por John Gossman, um dos arquitetos do WPF e Silverlight na Microsoft. O MVVM assemelha-se em alguns aspectos ao MVC (Model View Controller) e ao MVP (Model View Presenter), pode-se até dizer que ele é uma especialização do MVP (SOUZA, 2010).

O MVVM visa estabelecer definição e separação clara de responsabilidades em uma aplicação, mantendo uma espécie de façade entre o Modelo de Objetos, ou seja, as classes de negócio, serviços externos e até mesmo acesso a banco de dados, e a View, que é a interface, com a qual o usuário interage.

Desta forma há uma clara separação das camadas. A camada Model não conhece a View e vice-versa, pois a View conhece a ViewModel e se comunica com ela através do mecanismo de binding. São os mecanismos de binding, eventos roteados e comandos roteados, que fazem do MVVM um padrão poderoso. A comunicação ocorre conforme figura a seguir:

Figura 4 – Padrão Arquitetural MVVM



Fonte: Microsoft (2017)

A View através do Data Binding interage com a ViewModel notificando sobre os eventos ocorridos, bem como, disparo de comandos. Então a ViewModel responde tais notificações por

meio de alguma ação no Model, que pode ser obter algum dado ou até mesmo inserindo e atualizando informações.

2.7 FRAMEWORKS DE DESENVOLVIMENTO

Frameworks para desenvolvimento de aplicações são abstrações que utilizam as características comuns de aplicativos feitos com essas ferramentas para facilitar e acelerar o processo de desenvolvimento. Diversos são os Frameworks que podem ser utilizados no desenvolvimento de aplicativos móveis, e a escolha de quais Frameworks utilizar varia de acordo com as necessidades e características de cada projeto. No presente trabalho serão abordados com mais detalhes os principais Frameworks utilizados.

2.7.1 Framework Foundation

O Foundation é um framework que foi criado para facilitar o desenvolvimento utilizando a linguagem Objective C, visando tornar o desenvolvimento mais produtivo por meio das facilidades oferecidas.

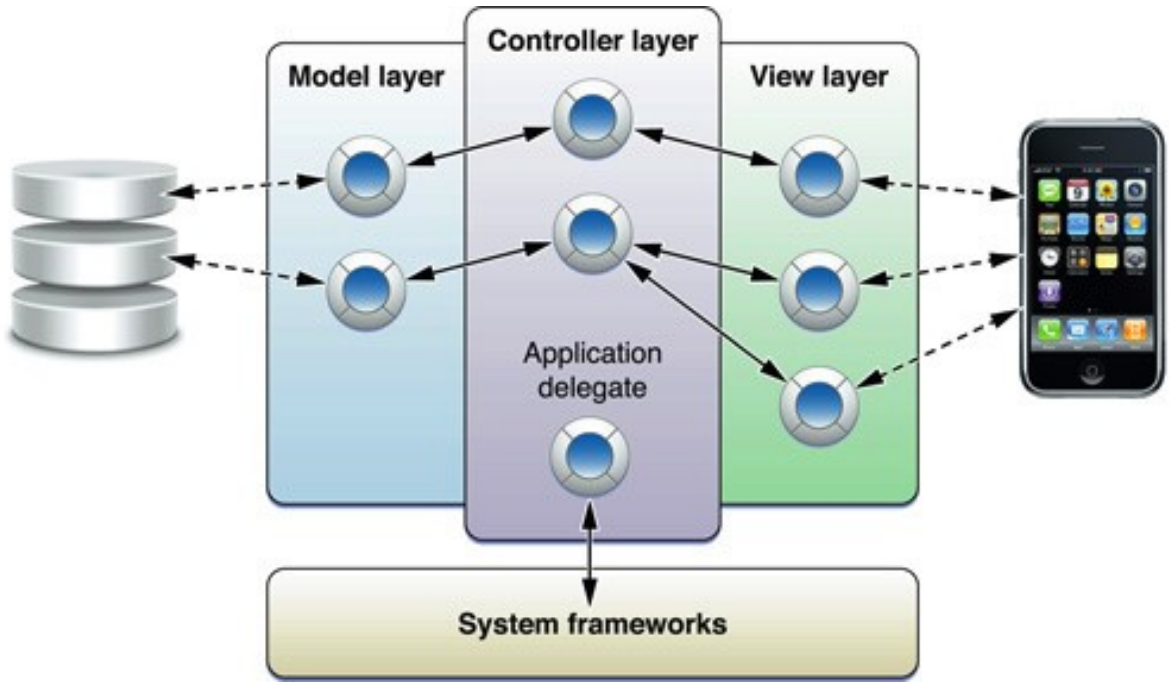
Este framework contém um conjunto de classes escritas em Objective C para tornar o desenvolvimento mais rápido, como por exemplo as classes NSObject, NSString, NSDate, NSNumber, NSArray, NSDictionary, NSURL, entre outras (LECHETA, 2016).

2.7.1.1 Framework Cocoa Touch

O framework Cocoa Touch foi baseado no framework Foundation, é bem integrado ao ambiente de desenvolvimento Xcode, utilizado na plataforma iOS, facilitando a utilização de vários tipos de recursos de multimídia, networking, animações, entre outros, procurando esconder a complexidade desses recursos.

O Cocoa Touch é baseado no padrão MVC (Model View Controller). A camada de View é construída utilizando-se o Interface Builder, que oferece recursos de design facilitadores. A camada Controller é composta pelas classes filhas de UIViewController, e basicamente são elas que definem o ciclo de vida das telas, tratam eventos do usuário, controlam a navegação e interagem com a camada Model, que contém as classes responsáveis pela lógica do negócio (LECHETA, 2016).

Figura 5 – Framework Cocoa Touch



Fonte: Apple (2016a)

2.7.1.2 Framework ReactiveX (Rx)

ReactiveX (Rx) é uma abstração genérica de computação expressa por meio de elementos Observable de interface. Programação reativa não é um assunto novo, já é bastante difundido e utilizado em diversas linguagens.

Atualmente o Framework ReactiveX contempla diversas linguagens e plataformas, como por exemplo Java, JavaScript, Scala, Lua, Ruby, Python, Go, Kotlin, entre outras, incluindo Framework para desenvolvimento Android (RxAndroid) e para plataforma iOS (RxSwift e RxCocoa) que serão abordados em maiores detalhes no presente trabalho.

2.7.1.2.1 Frameworks RxSwift e RxCocoa

RxSwift e RxCocoa fazem parte do conjunto de ferramentas de ReactiveX (geralmente abreviado como "Rx") que abrangem múltiplas linguagens de programação e plataformas (SHAPIRO, 2017).

RxSwift é um Framework para interagir com a linguagem de programação Swift, enquanto RxCocoa é um Framework que ajuda a tornar as APIs Cocoa usadas em iOS mais fáceis de usar com técnicas reativas. Assim como no ReactiveX, a intenção é facilitar a composição de operações assíncronas e fluxos de dados.

De maneira resumida, a utilização de ReactiveX, por meio de RxSwitf e RxCocoa, tornará o código mais reutilizável, declarativo, compreensivo e conciso, aumentando o nível de abstração.

2.8 TECNOLOGIAS AUXILIARES

Existem diversas tecnologias auxiliares que podem facilitar e agilizar o desenvolvimento de maneira geral, seja por meio da utilização de Frameworks, bibliotecas, entre outras tecnologias.

Se tratando da plataforma iOS, é possível agilizar e facilitar o desenvolvimento, bem como, incluir novos recursos em um projeto, utilizando bibliotecas externas, chamadas de Pods. Para isto é necessário a utilização de um gerenciador de dependências, chamado CocoaPods, que é fortemente inspirado na combinação de projetos Ruby com RubyGems e Bundler. No presente trabalho serão abordados alguns dos principais Pods utilizados no projeto.

2.8.1 SwiftyJSON

Swift é bastante rigoroso em relação à tipos. E embora a digitação explícita seja boa para salvar o desenvolvedor de erros, torna-se complicado ao lidar com JSON e outras áreas que são, por natureza, implícitas sobre os tipos. SwiftyJSON torna fácil lidar com dados JSON em Swift (SWIFTYJSON, 2017).

Este Pod facilita a conversão de JSON para objeto, simplificando a maneira como os campos de JSON são capturados.

2.8.2 Kingfisher

Kingfisher é uma biblioteca leve e puramente em Swift, para download e cache de imagens da web. O projeto é fortemente inspirado pelo popular SDWebImage e oferece a possibilidade de usar uma alternativa de Swift puro (KINGFISHER, 2017).

Algumas de suas características são download assíncrono de imagens e armazenamento em cache, cache de múltiplas camadas, downloads e tarefas de processamento canceláveis para melhoria de desempenho, componentes independentes, ou seja, possibilidade de usar o downloader e sistema de cache separadamente, placeholder customizável durante carregamento de imagens, entre outras características.

2.8.3 Popup Dialog

Popup Dialog é um popup de diálogo simples e personalizável escrito em Swift. PopupDialog é uma subclasse de UIViewController e, como tal, pode ser adicionado ao controller da view de forma modular (POPUPDIALOG, 2017).

De maneira geral, no desenvolvimento iOS existe o componente UIAlertDialog, usado como padrão, porém com o referido Pod é possível customizá-lo de maneira mais fácil e simples.

2.8.4 Pastel

Pastel é um Pod para animações com efeito gradiente, possibilitando que views assumam, ao invés de cores sólidas, gradientes com transições e animações (PASTEL, 2017).

Com este Pod é possível aplicar de forma fácil e rápida estilos de gradiente em views, colaborando para melhoria na experiência do usuário.

2.8.5 Hero

Hero é uma biblioteca para criação de transições, possibilitando a customização de transições de maneira fácil e rápida, por meio das propriedades heroModifiers. Todas as animações suportam interação por meio gestos dos usuários (HERO, 2017).

Com este Pod é possível aplicar transições personalizadas enriquecendo experiência do usuário, contribuindo para melhorias de design da aplicação.

2.8.6 SlideMenuControllerSwift

SlideMenuControllerSwift é um Slide Menu para iOS baseado em aplicativos Google, escrito totalmente em Swift (SLIDEMENUCONTROLLERSWIFT, 2017).

Este Pod possibilita a implementação de um menu lateral, menu este bastante comum em aplicativos Android, visto que é um padrão adotado pela Google e também uma característica bastante utilizada para user experience (UX) em aplicativos Android. Diferentemente da plataforma Android, onde é possível obter este recurso praticamente implementado, na plataforma iOS não existe um padrão já implementado. Desta forma, com este Pod a implementação deste recurso é facilitada.

2.8.7 Unbox

Unbox é um decodificador de Swift JSON fácil de usar. É leve e não exige que se use uma subclasse. É possível de ser utilizado em qualquer model com facilidade (UNBOX, 2017).

Basicamente, se faz a herança de unboxable e se implementa desserialização do JSON, automatizando o processo, visto que a classe transformará as informações em um objeto de seu tipo.

2.9 FERRAMENTAS AUXILIARES

Existem diversas ferramentas que podem auxiliar o desenvolvedor em um projeto, independente da fase em que se tal projeto, estas ferramentas, que vão além de um IDE, podem facilitar e agilizar muitos processos, como, por exemplo, ferramentas para modelagem de diagramas, ferramentas para prototipagem de interfaces gráficas e até mesmo ferramentas para gerenciamento de versionamento de código, ferramentas estas utilizadas em diferentes

fases do projeto. Neste capítulo serão abordadas sobre algumas ferramentas auxiliares utilizadas no projeto.

2.9.1 Astah – Modelagem de Diagramas

Como parte do desenvolvimento de um sistema é bastante importante a etapa de modelagem, esta etapa é feita geralmente utilizando-se Linguagem de Modelagem Unificada, conhecida como UML (Unified Modeling Language). Nesta etapa são construídos os principais diagramas que representam o sistema, como por exemplo, diagrama de casos de uso e diagrama de classes.

A modelagem define os seus sistemas de forma mais fácil de entender, mais simples de se comunicar e mais em contato com as pessoas que as utilizam (ASTAH, 2017).

Para modelagem e criação destes diagramas existem diversas ferramentas disponíveis, sendo uma das mais utilizadas a ferramenta Astah, que possui versão paga e gratuita, onde é possível a criação de uma grande variedade de fluxogramas e diagramas. Tal ferramenta foi utilizada para modelagem dos diagramas do presente projeto.

2.9.2 Sketch – Modelagem de Interfaces

Fazer a prototipagem da interface gráfica do sistema é um passo importante e que facilita bastante o entendimento do projeto, pois neste passo é possível estipular quais os elementos essenciais em cada tela ou página do sistema.

Existem diversas ferramentas para criação de protótipos de telas, com diferentes recursos. Dentre estas ferramentas se destaca a ferramenta Sketch, que possui diversos recursos para elaboração de interfaces gráficas, de maneira simples e ágil.

O Sketch é um conjunto de ferramentas de design criado para auxiliar na criação desde as primeiras ideias até a arte final (SKETCH, 2017).

A referida ferramenta foi utilizada no presente projeto para elaboração dos protótipos das principais telas do aplicativo.

2.9.3 GitKraken – Interface Gráfica para Git

Fazer versionamento de código é um dos elementos de maior relevância durante o desenvolvimento de uma aplicação, principalmente quando se está trabalhando em equipe, pois é possível manter o histórico de todas as alterações feitas.

Existem diversas formas de fazer versionamento de código, sendo que a utilizada no presente projeto foi versionamento com Git. Tal versionamento foi feito localmente e também em repositório remoto para garantir redundância e maior segurança.

Para agilizar o processo de versionamento, visto que Git utiliza linhas de comando, utilizou-se uma interface gráfica, chamada GitKraken, onde tais ações podem ser executadas de maneira mais intuitiva e muita mais rápida se comparado à utilização de linhas de comando.

3 ESTADO DA ARTE

3.1 PROJETO NEXME!

O aplicativo NexMe! cujo nome vem das palavras "Next to Me" que em português significa "Perto de mim" é um aplicativo para promoção de eventos e interação entre usuários, com natureza de rede social e objetivo de facilitar a forma como os usuários encontram novos lugares e também a forma como interagem e conhecem pessoas novas.

As principais características do aplicativo são os perfis de usuários, promoção de eventos públicos e privados, que aparecerão no feed dos usuários de acordo com os filtros desejados e chat público para proporcionar interação entre os usuários participantes de um evento.

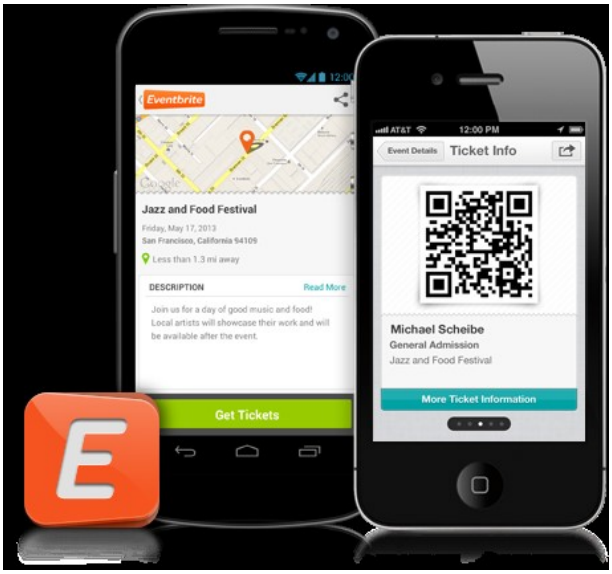
O aplicativo foi desenvolvido para a plataforma iOS, plataforma Android e versão Web, acessível diretamente pelo navegador. Neste trabalho foi desenvolvida a versão para a plataforma iOS. O desenvolvimento para esta plataforma justificou-se, pois, apesar de possuir uma base de usuários menor, os usuários desta plataforma, de maneira geral, apresentam maior interesse na compra de aplicativos, o que pode impactar no modelo de negócios do aplicativo, que futuramente será vender publicidade, bem como, poderá influenciar em módulos a serem implementados futuramente, diretamente ligados à venda de ingressos ou produtos. De acordo com pesquisa feita pela Panorama Mobile Time, cerca de 46% dos usuários de iOS já efetuaram compra de aplicativos, já na plataforma Android, cerca de 18% dos usuários (PAIVA, 2016).

3.2 APLICATIVOS SIMILARES

3.2.1 Eventbrite

O Eventbrite é uma aplicação para smartphone que propõe a divulgação de eventos e venda de ingressos. Uma de suas principais funcionalidades é de promover eventos próximos ao usuário (também sendo possível visualizar eventos de outras cidades) e mostrar informações sobre o evento como: localização do evento, quem é o promotor do evento, bem como a comercialização dos ingressos. De acordo com o site da aplicação Eventbrite, há uma grande quantidade de inscrições nos eventos promovidos através da plataforma, contabilizando no total 2 milhões de inscrições por ano (EVENTBRITE, 2016b).

Figura 6 – Aplicativo Eventbrite



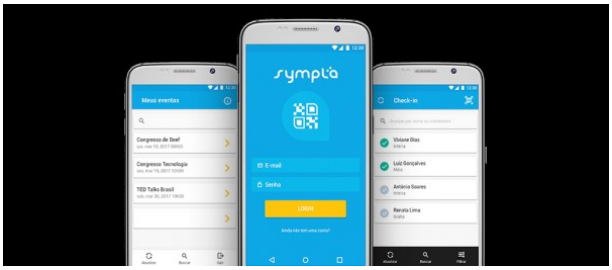
Fonte: Eventbrite (2016a)

3.2.2 Sympla

Segundo o site da própria aplicação Sympla, a aplicação é uma plataforma inteligente de gestão de eventos e venda de ingressos voltada para produtores de eventos de pequeno e médio porte, possibilitando ao produtor criar uma página personalizada, atrair público via redes sociais, ter controle de bilheteria e vender ingressos físicos e online.

O aplicativo permite aos usuários visualizar uma ampla quantidade de eventos com diversas informações, como: local do evento, horário, informações sobre o organizador, e comercialização do ingresso.

Figura 7 – Aplicativo Sympla



Fonte: Sympla (2016)

3.2.3 Considerações e Comparações

Existem outras aplicações que contemplam a mesma categoria dos aplicativos citados, porém dentre os sistemas estudados, optou-se pela apresentação destes por possuírem números consideráveis sobre a utilização da aplicação.

As aplicações citadas já estão no mercado por um período de tempo considerável, Eventbrite desde 2006 e Sympla desde 2012, evidenciando que existe demanda por serviços nessa área (promoção de eventos), visto que tais empresas ainda estão ativas no mercado e atuando neste segmento. Muito semelhantes entre si, estes aplicativos também demostram semelhança com a aplicação proposta, visto que um dos objetivos principais é a promoção de eventos, objetivo este consolidado por ambos aplicativos, Eventbrite e Sympla.

Cabe destacar que estas aplicações citadas possuem uma certa burocracia quando da utilização para cadastrar e publicar um evento, contemplando etapas de aprovação que podem tornar o processo demorado.

No quadro 1 é possível visualizar um comparativo entre os aplicativos citados e o aplicativo NexMe!, evidenciando as principais funcionalidades presentes em cada uma das aplicações. Desta forma, é possível notar que o aplicativo NexMe! contempla várias funcionalidades que não estão presentes nos demais aplicativos.

Quadro 1 – Comparativo entre Aplicativos

Aplicativo	**Venda de Ingressos**	**Publicação de Eventos por Empresas**	**Publicação de Eventos por Usuários**	**Chat nos Eventos**	**Chat entre Usuários**	**Filtro de Eventos por Localização**
Eventbrite	X	X				
Sympla	X	X				X
NexMe!		X	X	X	X	X

Fonte: Elaborado pelo autor

4 PROCEDIMENTOS METODOLÓGICOS

Inicialmente foi feito levantamento de requisitos para se entender quais as características e dimensão do projeto. Então este foi dividido em módulos, com objetivo de se identificar quais as principais funções que precisavam ser desenvolvidas prioritariamente, visto que a dimensão do projeto pode ser grande não havendo tempo hábil para desenvolvimento do projeto completo.

Após a definição dos módulos a serem desenvolvidos (cadastro de usuários, cadastro de eventos, chat público e chat entre usuários), foi feita a modelagem do aplicativo, contendo diagrama dos principais casos de uso (com detalhamento) e diagrama de classes.

Então foi feita uma análise das tecnologias e ferramentas de desenvolvimento para a plataforma iOS. Após seleção das ferramentas adequadas, foi elaborada a prototipagem das telas a serem implementadas, com intuito de facilitar a visualização do projeto como um todo, e também a estruturação do banco de dados em Realtime Database.

Por fim, foi feita a implementação do aplicativo para a plataforma iOS utilizando prioritariamente a linguagem Swift.

5 DESENVOLVIMENTO

Na aplicação desenvolvida é possível a publicação e visualização de eventos no feed principal, bem como possibilidade de seguir perfis, é possível também interagir com os demais usuários por meio de chat público visível a todos os participantes do evento, bem como chat privado, entre 2 usuários.

5.1 LEVANTAMENTO DE REQUISITOS

5.1.1 Requisitos Funcionais

A primeira etapa de desenvolvimento contemplou o levantamento dos principais requisitos funcionais que o aplicativo deveria possuir, bem como a verificação da consistência desses requisitos. Os principais requisitos funcionais (RF) são:

- **RF01 - Cadastrar nova conta de usuário:** O sistema deve permitir a criação de nova conta de usuário;
- **RF02 - Autenticar usuário:** O sistema deve permitir a autenticação da conta de usuário existente;
- **RF03 - Publicar evento:** O sistema deve permitir a publicação de um evento pelo usuário autenticado;
- **RF04 - Participar de evento:** O sistema deve permitir a um usuário participar de um evento do feed;
- **RF05 - Visualizar eventos:** O sistema deve permitir visualização dos eventos no feed principal baseado na localização do usuário;
- **RF06 - Buscar evento e usuário:** O sistema deve permitir a busca de eventos e usuário por nome;
- **RF07 - Chat público:** O sistema deve permitir que usuário ingresse em chat público contendo todos os participantes do evento;
- **RF08 - Chat privado:** O sistema deve permitir que um usuário ingresse em chat privado com outro usuário;
- **RF09 - Seguir usuários:** O sistema deve permitir que um usuário siga outro perfil de usuário;

5.1.2 Requisitos Não Funcionais

Os requisitos não funcionais do aplicativo estão relacionados aos requisitos externos, de produto e de processo, conforme descritos a seguir:

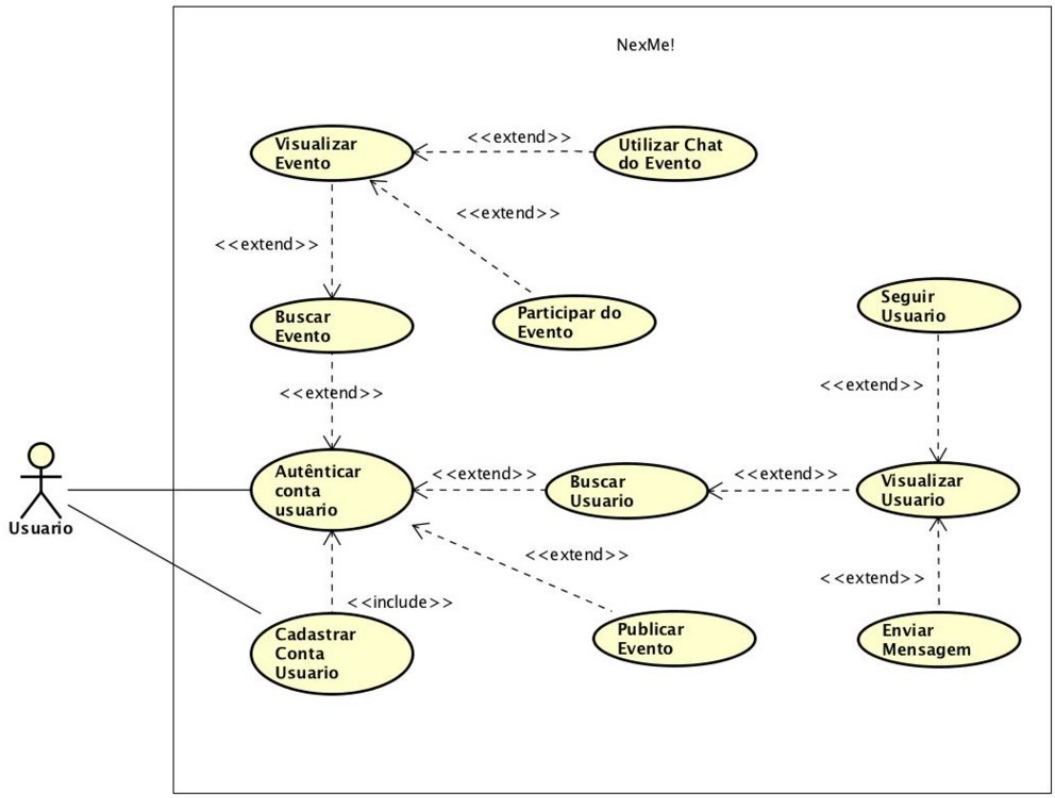
- **RNF01 - Usabilidade:** O sistema deve ter interface agradável e intuitiva;

- **RNF02 - Desempenho:** O sistema precisa apresentar bom desempenho na troca de telas para diferentes funcionalidades bem como rapidez na busca de informações;
- **RNF03 - Segurança:** O sistema deve possuir meios de proteção de dados dos usuários;
- **RNF04 - Manutenção:** O sistema deverá ser implementado utilizando-se padronização de código para facilitar manutenção.

5.2 DIAGRAMA DE CASOS DE USO

No diagrama de casos de uso é possível verificar as principais interações entre os usuários e o sistema, por meio das funcionalidades disponíveis.

Figura 8 – Diagrama de Casos de Uso



Fonte: Elaborado pelo Autor

5.3 ESTRUTURA DO BANCO DE DADOS

A Aplicação utiliza o Firebase Realtime Database, onde os dados são armazenados como objetos JSON, o banco de dados funciona como uma árvore JSON hospedada na nuvem. Ao contrário de um banco de dados SQL, não há tabelas nem registros. Quando um dado é adicionado à árvore JSON, ele se torna um node na estrutura JSON com uma chave associada. Os modelos de dados, ou seja, as classes do banco de dados da aplicação estão apresentados na Figura 9.

Figura 9 – Estrutura do Banco de Dados

Fonte: Elaborado pelo Autor

As principais classes do projeto são: a classe User, que representa os usuários do sistema, a classe Event, que representa os eventos, a classe Categories, representando as categorias de um evento, bem como, a classe Messages que representa as mensagens, podendo estar atreladas ao evento ou somente entre 2 usuários.

5.4 ARQUITETURA DA APLICAÇÃO

A arquitetura utilizada no projeto é o padrão MVVM (Model-View-ViewModel), padrão este que se assemelha tanto ao padrão MVC (Model-View-Controller) quanto ao padrão MVP (Model View Presenter). Este padrão visa estabelecer uma separação muito clara de responsabilidades, ou seja, existe uma clara separação das camadas. A camada Model não conhece a View e vice-versa, pois a View conhece a ViewModel e se comunica com ela através do mecanismo de binding. São os mecanismos de binding, eventos roteados e comandos roteados, que fazem do MVVM um padrão poderoso.

5.5 PROTOTIPAGEM DE INTERFACE GRÁFICA

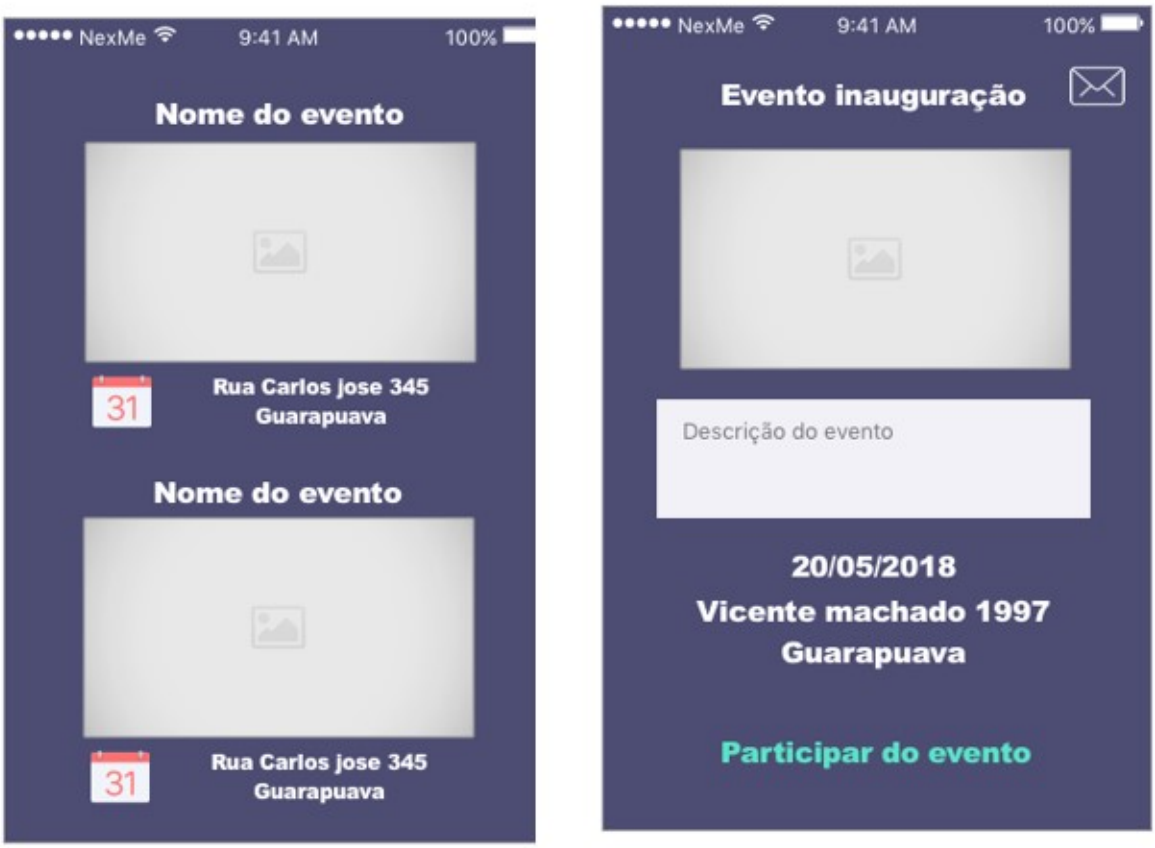
Nesta seção serão apresentados os protótipos das interfaces gráficas da aplicação. Estes protótipos foram modelos preliminares das telas do aplicativo utilizados com intuito de apresentar o conceito de design que se pretende aplicar.

Figura 10 – Protótipo Tela de Login

Fonte: Elaborado pelo Autor

A Figura 10 ilustra a tela de login, que será apresentada ao usuário quando for utilizar o aplicativo pela primeira vez ou quando efetuar logout de sua conta. Nesta tela é possível efetuar login ou acessar a tela para criação de conta onde é possível fazer o cadastro de nova conta de usuário.

Figura 11 – Protótipo Tela de Listagem de Eventos (esquerda) e Tela de Detalhes de Evento (direita)



Fonte: Elaborado pelo Autor

A Figura 11 ilustra a tela com a listagem de eventos, tela esta que mostrará os eventos disponíveis na região selecionada pelo usuário ou de acordo com o filtro utilizado, nesta tela é possível ver as informações principais do evento e também acessar a tela com detalhamento do evento (direita) com maiores detalhes do evento e possibilidade de participar do evento.

Figura 12 – Protótipo da Tela Perfil de Usuário

Fonte: Elaborado pelo Autor

A Figura 12 ilustra a tela com informações do perfil de usuário, nesta tela é possível visualizar informações como nome do usuário, bem como quantidade de seguidores e de perfis que está seguindo. Também é possível visualizar quantos eventos esse usuário já publicou, bem como quantos eventos está participando.

Figura 13 – Protótipo da Tela de Cadastro de Eventos

Fonte: Elaborado pelo Autor

A Figura 13 ilustra a tela para cadastro de um evento, nesta tela é possível inserir informações como, por exemplo, imagem para o evento, descrição, data, localização, entre outras informações.

6 O APLICATIVO NEXME!

NexMe! é um aplicativo para promoção de eventos e interação entre usuários, com natureza de rede social e objetivo de facilitar a forma como os usuários encontram novos lugares e também a forma como interagem e conhecem pessoas novas.

As principais características do aplicativo são os perfis de usuários, promoção de eventos, que aparecerão no feed dos usuários de acordo com os filtros desejados e chat público para proporcionar interação entre os usuários participantes de um evento, bem como, possibilidade de chat privado entre 2 usuários.

6.1 IDENTIDADE VISUAL

A identidade visual do aplicativo foi construída levando-se em conta o nome da aplicação (NexMe!) que remete ao conceito de "Next to Me" e também levando em conta o conceito de localização tanto do usuário quanto dos locais de eventos.

As cores utilizadas foram tons de azul, verde e branco. Tais cores foram utilizadas como padrão para a grande maioria das telas do aplicativo.

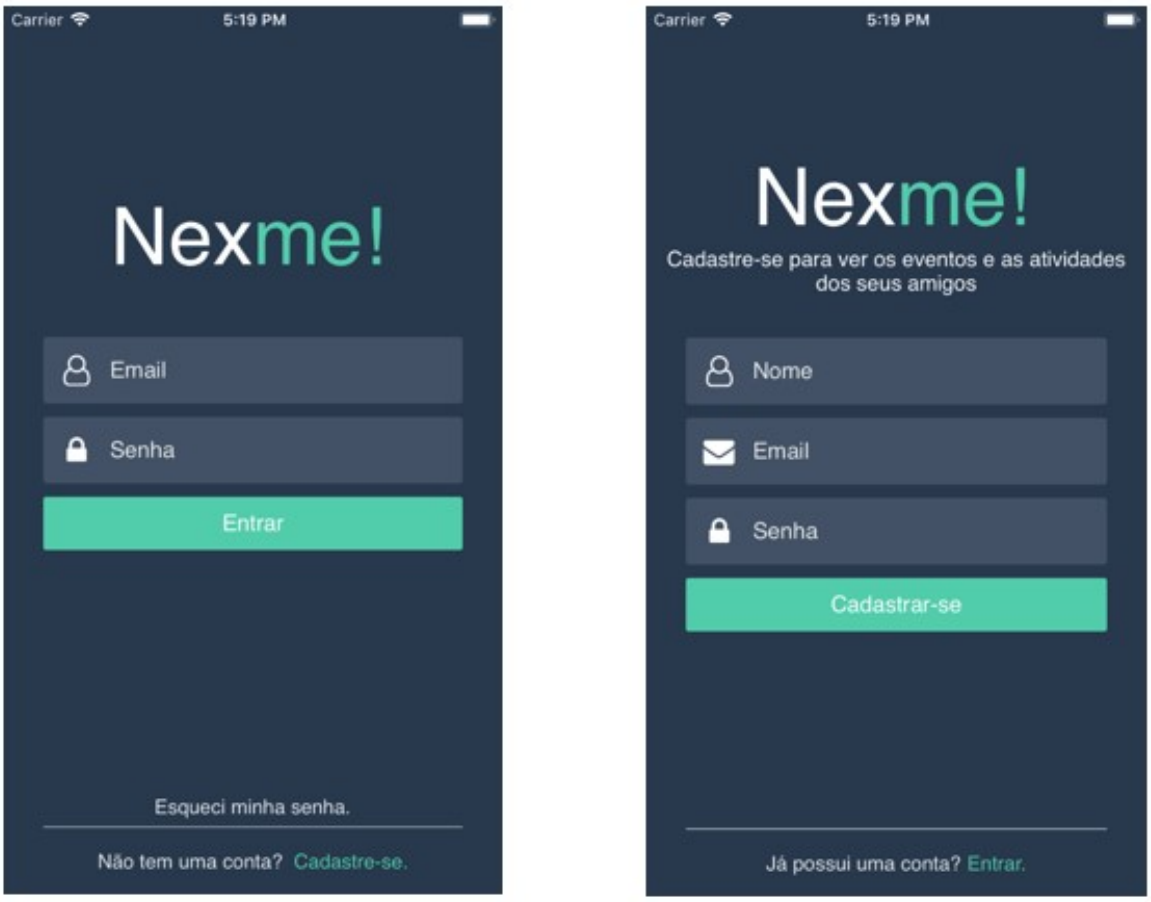
Figura 14 – Identidade Visual da Aplicação NexMe!

Fonte: Elaborado pelo autor

6.2 CADASTRO E AUTENTICAÇÃO DE USUÁRIOS

O aplicativo NexMe! inicia com a tela de autenticação de usuário, conforme Figura 15, nesta tela é possível inserir e-mail e senha para efetuar login na aplicação ou solicitar recuperação de senha.

Figura 15 – Tela de Login (esquerda) e Tela de Cadastro (direita)



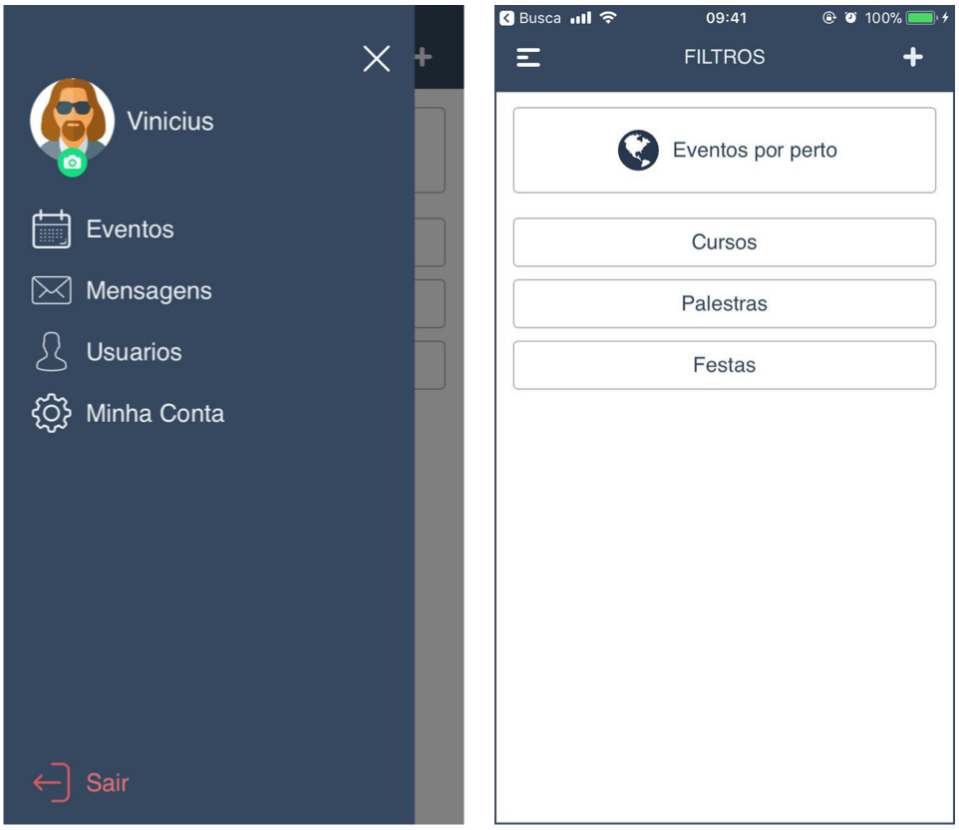
Fonte: Elaborado pelo autor

Caso o usuário ainda não possua conta no aplicativo é necessário que faça o cadastro, preenchendo os campos de nome, e-mail e senha, conforme mostra a Figura 15.

6.3 TELA PRINCIPAL E FILTROS DE EVENTOS

Após autenticação do usuário no aplicativo, este será redirecionado para a tela principal, que contém um menu lateral com algumas opções, bem como os filtros disponíveis para busca de eventos, conforme Figura 16. Na tela principal também é possível iniciar a criação de um evento por meio do botão na barra superior da tela.

Figura 16 – Menu Lateral (esquerda) e Tela Principal (direita)



Fonte: Elaborado pelo autor

6.4 CRIAÇÃO E PUBLICAÇÃO DE EVENTOS

Para criação de um evento, o usuário deve acessar a página de criação de eventos, por meio do botão na barra superior da tela principal do aplicativo. Então o aplicativo retornará a tela de criação de eventos, conforme Figura 17, onde é possível inserir todas as informações e salvar o evento. Após salvar o evento, este será publicado automaticamente no feed e também ficará atrelado ao perfil do usuário que o criou.

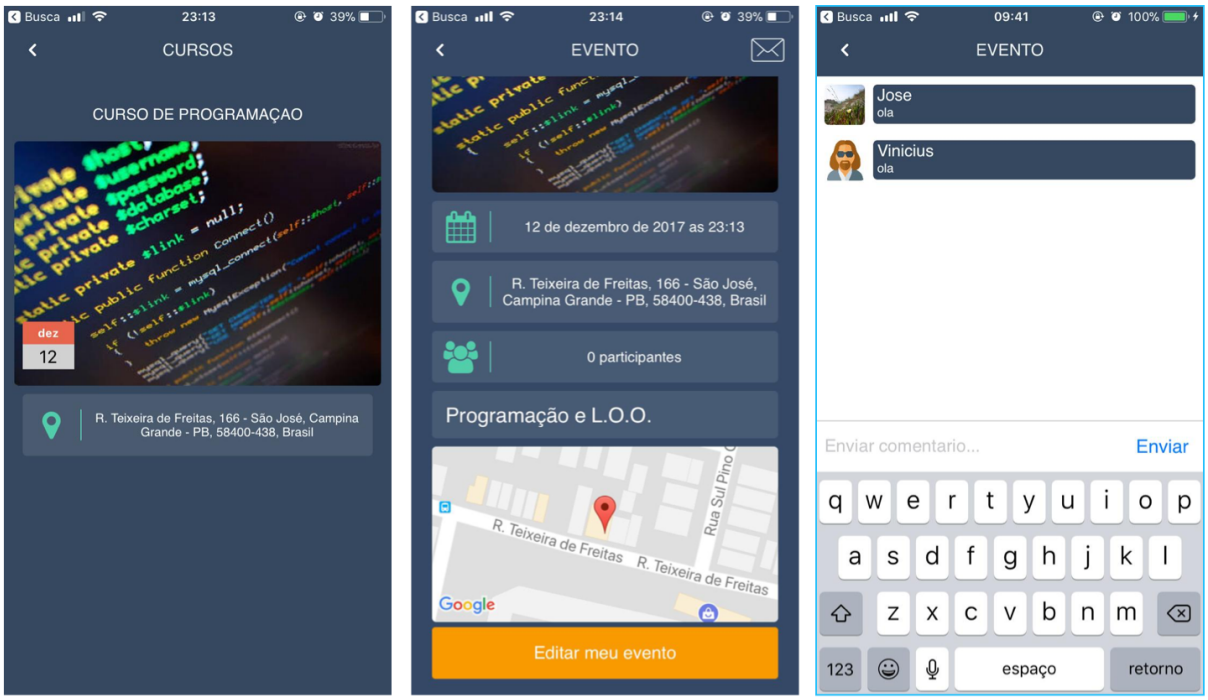
Figura 17 – Tela para Criação e Publicação de um Evento

Fonte: Elaborado pelo autor

6.5 LISTAGEM E DETALHAMENTO DE EVENTOS

A listagem de eventos ocorre de acordo com o filtro selecionado. Sempre que um filtro é selecionado uma listagem com os eventos correspondentes aparece, conforme Figura 18. Caso o usuário deseje obter mais informações, basta clicar sobre o evento e a tela de detalhamento do evento será mostrada. Na tela de detalhamento é possível visualizar as informações do evento, confirmar presença no evento e acessar o chat público do evento, onde ficam armazenadas as mensagens enviadas pelos participantes do evento, conforme ilustra Figura 18.

Figura 18 – Tela de Listagem de Eventos (esquerda) e Tela de Detalhes do Evento (centro) e Tela Chat Público do Evento (direita)

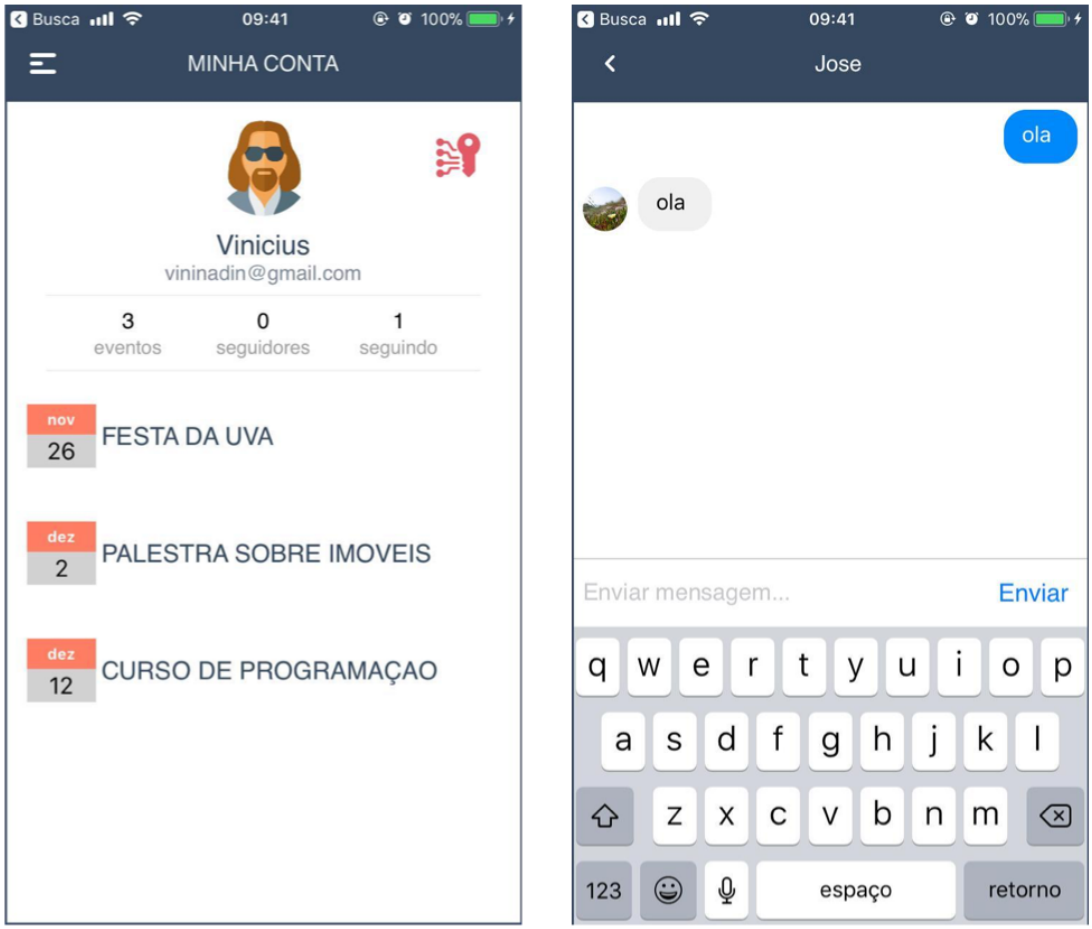


Fonte: Elaborado pelo autor

6.6 PERFIS DE USUÁRIOS E FUNCIONALIDADE DE SEGUIR

No aplicativo também é possível visualizar as informações do perfil de um usuário, como quantidade de seguidores e seguindo, bem como eventos publicados pelo usuário, conforme ilustra Figura 19. Também é possível seguir um perfil de usuário e mandar mensagem direta ao usuário.

Figura 19 – Tela de Perfil do Usuário (direta) e chat privado (esquerda)



Fonte: Elaborado pelo autor

6.7 CONSIDERAÇÕES

Conforme exposto nas seções anteriores, o aplicativo NexMe! contempla as funções de cadastro e autenticação de usuários, criação e publicação de eventos, listagem e detalhamento de eventos, onde é possível filtrar a listagem dos eventos por categorias e localização geográfica. Além disso, os usuários tem a possibilidade de seguir outros usuários, enviar mensagens privadas à outros usuários, bem como interagir com demais usuários no chat público dentro dos eventos em que se está participando, objetivando a interação social, um dos principais intuitos da aplicação.

Além das funcionalidades desenvolvidas no presente trabalho, existem outras possíveis funcionalidades que se enquadrariam na proposta do aplicativo, e que poderão ser desenvolvidas como trabalhos futuros, funcionalidades referentes por exemplo ao módulo de pagamento de ingressos, entre outras.

7 CONSIDERAÇÕES FINAIS

O mercado de aplicativos móveis vem crescendo consideravelmente e aliado a isso surgem novas oportunidades de negócios. Entendendo-se que a sociedade vem sofrendo mudanças comportamentais, exigindo praticidade e agilidade para tarefas cotidianas, algumas empresas estão se adaptando a essa nova realidade, adaptando seus serviços para torná-los mais práticos e ágeis. Aplicações com funcionalidades exclusivas para a versão mobile estão se tornando mais comuns. Diante deste cenário e da observação do comportamento dos usuários em redes sociais, chegou-se na ideia de desenvolvimento do aplicativo NexMe! para promoção de eventos e interação social.

O diferencial do aplicativo é centralizar algumas funções já existentes em aplicações diversas facilitando a forma como o usuário encontra novos lugares, fica por dentro dos eventos que estão acontecendo em sua região e também pela maneira como interage com as pessoas que participam de um evento.

Identificou-se a necessidade de desenvolver uma versão para a plataforma iOS, pois apesar de possuir uma base de usuários menor, os usuários desta plataforma, de maneira geral, apresentam maior interesse em efetuar compras dentro de um aplicativo, conforme exposto na presente pesquisa. Este fator pode impactar em um possível modelo de negócios a ser usado para o aplicativo, que poderá ser a venda de publicidade por exemplo, bem como, poderá influenciar em módulos a serem implementados futuramente, diretamente ligados à venda de ingressos ou produtos.

Desta forma, no presente trabalho foram abordados os aspectos relacionados ao desenvolvimento da versão iOS do aplicativo NexMe! para promoção de eventos e interação social, que está disponível para download e instalação na Apple Store, loja de aplicativos da Apple.

Como trabalhos futuros, pode-se considerar o desenvolvimento de outros módulos do aplicativo, como por exemplo, módulo para compra e pagamento de ingressos dos eventos, além disto, outras opções de chat, como por exemplo chat privado de grupos, entre outros módulos que possam agregar valor ao aplicativo e melhorar a experiência do usuário.

REFERÊNCIAS

APPLE. **About the Basic Programming Concepts for Cocoa and Cocoa Touch**. 2016. Disponível em: <https://developer.apple.com/library/content/documentation/General/Conceptual/CocoaEncyclopedia/Introduction/Introduction.html>. Acesso em: 31 de outubro de 2016. Citado na página 11.

APPLE. **Xcode IDE**. 2016. Disponível em: <https://developer.apple.com/xcode/ide/>. Acesso em: 1 de novembro de 2016. Citado na página 7.

APPLE. **Inside iOS 11: Apple's Control Center grows modular, gets customizable**. 2017. Disponível em: <http://appleinsider.com/articles/17/06/07/inside-ios-11-apples-control-center-grows-modular-gets-customizable>. Acesso em: 31 de outubro de 2017. Citado na página 5.

APPLE. **iOS 11**. 2017. Disponível em: <https://www.apple.com/br/ios/ios-11/>. Acesso em: 31 de outubro de 2017. Citado na página 4.

APPLE. **Swift: uma linguagem aberta e poderosa para todo mundo criar apps incríveis**. 2017. Disponível em: <https://www.apple.com/br/swift/>. Acesso em: 31 de outubro de 2017. Citado na página 6.

ASTAH. **Astah is modeling**. 2017. Disponível em: <http://astah.net/>. Acesso em: 1 de novembro de 2017. Citado na página 14.

CABRAL, S. S. **Política de aprovação da App Store considerará infrações de direitos autorais com maior rigidez**. 2011. Disponível em: <https://macmagazine.com.br/2011/02/18/politica-de-aprovacao-da-app-store-considerara-infracoes-de-direitos-autorais-com-maior-rigidez/>. Acesso em: 05 de novembro de 2017. Citado na página 4.

CISSOTO, L. **7 motivos para trocar Objective-C por Swift**. 2015. Disponível em: <https://macmagazine.com.br/2015/05/22/7-motivos-para-trocar-objective-c-por-swift/>. Acesso em: 05 de novembro de 2017. Citado na página 5.

COLLETA, L. F. **Ambiente de desenvolvimento integrado (IDE)**. 2017. Disponível em: <http://luizcoletta.com/wp-content/uploads/sites/9/2016/07/EBS211-Aula2.1.pdf>. Acesso em: 05 de agosto de 2017. Citado na página 6.

DUNN, J. **There's no hope of anyone catching up to Android and iOS**. 2017. Disponível em: <http://www.businessinsider.com/smartphone-market-share-android-ios-windows-blackberry-2016-8>. Acesso em: 12 de julho de 2017. Citado na página 4.

EVENTBRITE. **Organizar Eventos**. 2016. Disponível em: <https://eventbrite.com.br/l/organizar-eventos/>. Acesso em: 18 de setembro de 2016. Citado na página 16.

EVENTBRITE. **Quem somos**. 2016. Disponível em: <https://www.eventbrite.com.br/about/>. Acesso em: 18 de setembro de 2016. Citado na página 15.

FERNANDES, M. C. **O que é um aplicativo móvel**. 2017. Disponível em: <http://blog.stone.com.br/aplicativo-movel/>. Acesso em: 20 de junho de 2017. Citado na página 3.

FICODE. **Firebase Realtime Database: Installation and Setup**. 2017. Disponível em: <http://www.ficode.co.uk/blog/firebase-realtime-database-installation-setup/>. Acesso em: 1 de novembro de 2017. Citado na página 8.

HERO. **Elegant transition library for iOS and tvOS**. 2017. Disponível em: <https://github.com/lkzhao/Hero>. Acesso em: 05 de setembro de 2017. Citado na página 13.

JARDIM, J.; SILVEIRA, G. **Swift Programação para iPhone e iPad**. 1. ed. São Paulo: Casa do Código, 2016. Citado na página 6.

KHALAF, S. **Crescimento para o segmento móvel**. 2016. Disponível em: <https://yahoobr.tumblr.com/post/137102240333/aplicativos-de-notícias-produtividade-e>. Acesso em: 05 de setembro de 2016. Citado na página 1.

KINGFISHER. **A lightweight, pure-Swift library for downloading and caching images from the web**. 2017. Disponível em: <https://github.com/onevcat/Kingfisher>. Acesso em: 05 de setembro de 2017. Citado na página 12.

LECHETA, R. **Desenvolvendo para iPhone e iPad**. 4. ed. São Paulo: Novatec, 2016. Citado 3 vezes nas páginas 6, 7 e 10.

MEDEIROS, H. **Introdução ao Padrão MVC**. 2017. Disponível em: <https://www.devmedia.com.br/introducao-ao-padrao-mvc/29308>. Acesso em: 05 de agosto de 2017. Citado na página 9.

MICROSOFT. **Implementing the MVVM Pattern**. 2017. Disponível em: <https://msdn.microsoft.com/en-us/library/gg405484(v=pandp.40).aspx>. Acesso em: 05 de setembro de 2017. Citado na página 9.

MUNGOI, D. **Firebase**. 2016. Disponível em: <https://medium.com/google-developer-experts/primeira-impress˜ao-do-novo-firebase-afaeaeaa1395>. Acesso em: 22 de maio de 2017. Citado na página 8.

OPUS. **Estatísticas de uso de celular no Brasil**. 2017. Disponível em: <https://www.opus-software.com.br/estatisticas-uso-celular-brasil/>. Acesso em: 15 de junho de 2017. Citado na página 2.

PAIVA, F. **Uso de Apps no Brasil**. 2016. Disponível em: <http://www.cienciianasnuvens.com.br/site/wp-content/uploads/2015/03/2016.02.18_PANORAMA-DE-USO-DE-APPS-NO-BRASIL.pdf>. Acesso em: 05 de setembro de 2016. Citado na página 15.

PASTEL. **Gradient animation effect iOS**. 2017. Disponível em: <https://github.com/cruisediary/Pastel>. Acesso em: 05 de setembro de 2017. Citado na página 13.

POPUPDIALOG. **A simple, customizable popup dialog for iOS written in Swift**. 2017. Disponível em: <https://github.com/Orderella/PopupDialog>. Acesso em: 05 de setembro de 2017. Citado na página 12.

PORTO, F. **Aplicativos Mobile: Definições, História e Previsões**. 2017. Disponível em: <http://tectriadebrasil.com.br/blog/mercado-de-midias-sociais-blog/aplicativos-mobile-definicoes-historia-e-previsoes/>. Acesso em: 20 de junho de 2017. Citado na página 3.

SCUDERO, E. **A trajetória de um desenvolvedor mobile**. 2017. Disponível em: <https://becode.com.br/trajetoria-de-um-desenvolvedor-mobile/>. Acesso em: 05 de agosto de 2017. Citado na página 6.

SHAPIRO, E. **Getting Started With RxSwift and RxCocoa**. 2017. Disponível em: <https://www.raywenderlich.com/138547/getting-started-with-rxswift-and-rxcocoa>. Acesso em: 05 de setembro de 2017. Citado na página 11.

SKETCH. **Sketch the digital design toolkit**. 2017. Disponível em: <https://www.sketchapp.com/>. Acesso em: 1 de novembro de 2017. Citado na página 14.

SLIDEMENUCONTROLLERSWIFT. **iOS Slide Menu View based on Google+**. 2017. Disponível em: <https://github.com/dekatotoro/SlideMenuControllerSwift>. Acesso em: 05 de setembro de 2017. Citado na página 13.

SOUZA, C. E. F. de. **Entendendo o Pattern Model-View-ViewModel (MVVM)**. 2010. Disponível em: <https://imasters.com.br/artigo/18900/desenvolvimento/entendendo-o-pattern-model-view-viewmodel-mvvm?trace=1519021197&source=single>. Acesso em: 20 de julho de 2017. Citado na página 9.

SWIFTYJSON. **The better way to deal with JSON data in Swift**. 2017. Disponível em: <https://github.com/SwiftyJSON/SwiftyJSON>. Acesso em: 05 de setembro de 2017. Citado na página 12.

SYMPLA. **Sobre o Sympla**. 2016. Disponível em: <ttp://sympla.com.br/sobre-sympla>. Acesso em: 18 de setembro de 2016. Citado na página 16.

UNBOX. **The easy to use Swift JSON decoder**. 2017. Disponível em: <https://github.com/JohnSundell/Unbox>. Acesso em: 05 de setembro de 2017. Citado na página 13.