UNIVERSIDADE FEDERAL DO CEARÁ

INSTITUTO UNIVERSIDADE VIRTUAL

CURSO DE GRADUAÇÃO EM SISTEMAS E MÍDIAS DIGITAIS

DÉBORA MOURA MARINHO DE OLIVEIRA

DESENVOLVIMENTO DE UM APLICATIVO MÓVEL PARA AUXILIAR O USO DE BICICLETAS COMPARTILHADAS COMO FORMA DE MOBILIDADE URBANA SUSTENTÁVEL

FORTALEZA

2019

DÉBORA MOURA MARINHO DE OLIVEIRA

DESENVOLVIMENTO DE UM APLICATIVO MÓVEL PARA AUXILIAR O USO DE BICICLETAS COMPARTILHADAS COMO FORMA DE MOBILIDADE URBANA SUSTENTÁVEL

Relatório Técnico apresentado ao Curso de Graduação em Sistemas e Mídias Digitais do Instituto Universidade Virtual da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Sistemas e Mídias Digitais.

Orientadora: Profa. Dra. Maria de Fátima Costa de Souza

FORTALEZA

2019

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

O46d Oliveira, Débora Moura Marinho de.
Desenvolvimento de um aplicativo móvel para auxiliar o uso de bicicletas compartilhadas como forma de mobilidade urbana sustentável / Débora Moura Marinho de Oliveira. – 2019.
61 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, 3, Fortaleza, 2019.
Orientação: Profa. Dra. Maria de Fátima Costa de Souza.

1. Mobilidade Urbana Sustentável. 2. Desenvolvimento Mobile iOS. 3. Sistemas Compartilhados de Bicicleta. I. Título.

CDD

DÉBORA MOURA MARINHO DE OLIVEIRA

DESENVOLVIMENTO DE UM APLICATIVO MÓVEL PARA AUXILIAR O USO DE BICICLETAS COMPARTILHADAS COMO FORMA DE MOBILIDADE URBANA SUSTENTÁVEL

Relatório Técnico apresentado ao Curso de Graduação em Sistemas e Mídias Digitais do Instituto Universidade Virtual da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Sistemas e Mídias Digitais.

Aprovada em:

BANCA EXAMINADORA

Profa. Dra. Maria de Fátima Costa de Souza (Orientadora)
Universidade Federal do Ceará (UFC)

Prof. Dr. Leonardo Oliveira Moreira
Universidade Federal do Ceará (UFC)

Profa. Dra. Ticiana Linhares Coelho da Silva
Universidade Federal do Ceará (UFC)

AGRADECIMENTOS

À minha orientadora, Profa. Fátima Souza, por todo o auxílio na construção deste trabalho e por sempre acreditar que daria tudo certo.

Aos meus pais, por todos os sacrifícios que foram necessários para que eu tivesse uma boa educação, além dos recursos básicos, me incentivando à dar o meu melhor nos meus estudos. Por todos os nãos que me deram para eu me tornar quem sou hoje. À minha mãe, que é minha referência de mulher forte e amorosa, e motivação para eu ser sempre uma pessoa melhor.

À minha irmã Priscila, que me ajudou a aliviar a mente em tantos momentos de pressão, fosse assistindo séries ou cantando músicas da Disney.

Ao meu companheiro e melhor amigo Jonas, por ter topado entrar nessa jornada comigo. Por me levar para sair, tocar violão para mim, inventar mil coisas para me fazer sentir melhor nos momentos difíceis. Por me lembrar das minhas qualidades quando eu as esquecia.

Ao meu cachorinho Loki, que me trouxe tanta alegria nesses últimos anos. Por estar comigo durante as noites em claro trabalhando, por me fazer rir quando eu estava exausta, por deixar que eu o abraçasse quando isso era tudo que eu precisava.

Aos amigos que conquistei dentro do PET Computação UFC. Por terem vivido comigo tantos momentos bons e me ensinado tanta coisa. Ao PET, que foi uma grande escola para mim e pontapé inicial para tudo que estaria por vir depois.

Aos meus amigos e mentores do Apple Developer Academy IFCE, por momentos incríveis de conversas e muita troca de conhecimento. Ao programa, por ter me dado todo o suporte, material e incentivos necessários para desenvolver este trabalho e me preparar profissionalmente.

A mim, por não desistir. Por me dedicar à tudo que eu me propus e sempre tentar evoluir para uma versão melhor de mim.

A Deus, que sei que esteve comigo por toda essa jornada.

A todos que direta ou indiretamente me ajudaram nesse caminho durante os quatro anos de graduação.

"Imagination is the Discovering Faculty, pre-eminently. It is that which penetrates into the unseen worlds around us, the worlds of Science."

(Ada Lovelace)

RESUMO

Com a revolução industrial ocorreu o deslocamento de milhares de pessoas do campo para a cidade, ocasionando o rápido crescimento de centros urbanos e com eles um grande problema de mobilidade urbana. Esse problema inclui o aumento de poluição atmosférica e sonora, além de aumentar o tempo de deslocamento por conta do trânsito. A partir desse contexto, iniciou-se o debate sobre os fatores que caracterizariam um modelo de mobilidade urbana sustentável. A fim de incentivar a mobilidade urbana sustentável, no início do século XXI surgiu a iniciativa de disponibilizar bicicletas para uso compartilhado como meio de transporte sustentável nas cidades, conhecido como Sistemas de Compartilhamento de Bicicletas (SCB). Bicicletas são uma ótima opção de transporte para viagens curtas, pois possuem várias vantagens para saúde, para o meio ambiente e para a redução do trânsito nas vias. Com a evolução dos SCB a sua última geração conta com a Tecnologia da Informação (TI) para disponibilizar as bicicletas, normalmente fazendo uso de um aplicativo móvel. Com isso, surgiram dados abertos acerca dos SCB que podem ser consultados, estimulando a criação de iniciativas como o *CityBikes*, comunidade que divulga esses dados. Tendo em vista o acesso aos dados abertos dos SCB, foi pensado em criar um aplicativo que pudesse incentivar o uso da bicicleta como meio sustentável de transporte, o *Bikezera*. A proposta do aplicativo é apresentar de forma mais amigável para o usuário as informações acerca dos SCB e suas estações. Neste presente relatório é apresentado o contexto para desenvolvimento móvel *iOS*, informações fornecidas pela API *CityBikes*, aspectos da ideação do aplicativo, informações sobre seu público-alvo, *design* da interface, detalhes da codificação do aplicativo e seu processo de publicação na *App Store*. Por fim é apresentado o aplicativo finalizado e seu *link* para *download* na loja.

Palavras-chave: Mobilidade Urbana Sustentável. Desenvolvimento *Mobile iOS*. Sistemas Compartilhados de Bicicleta.

ABSTRACT

After the industrial revolution, thousands of people moved from the countryside to the city, causing a fast increase in urban centers and with them a major problem of urban mobility. This problem includes increase in air and noise pollution, and in travel time. From this context, a debate began on the factors that would characterize a model of sustainable urban mobility. In order to encourage sustainable urban mobility, at the beginning of the 21st century, an initiative emerged to make bicycles available for shared use as a sustainable means of transport, known as *Shared Bike-Systems* (SBS). Bicycles are a great transport option for short trips, as they have several advantages for health, the environment and the reduction of road traffic. With the evolution of SBS its latest generation relies on Information Technology to make bicycles available, usually using a mobile application. As a result, open data about the SBS that can be consulted emerged, encouraging the creation of initiatives such as *CityBikes*, a community that disseminate this data. In view of open access to the SBS data, it was thought of creating an application that could encourage the use of bicycles as a sustainable means of transport, the *Bikezera* app. The purpose of the application is to present user-friendly information about the SBS and their stations. This report presents the context for *iOS* mobile development, information provided by *CityBikes* API, aspects of app ideation, information about target audience, interface design, app coding details and the process to publish it on the *App Store*. Finally, the finished application and its *link* for download in the store are presented.

Keywords: Sustainable Urban Mobility. *iOS* Mobile Development. Shared bike-systems.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE CÓDIGOS-FONTE

LISTA DE ABREVIATURAS E SIGLAS

PCs	*Personal Computers*
API	*Application Programming Interfaces*
HIG	*Human Interface Guidelines*
IBGE	Instituto Brasileiro de Geografia e Estatistica
IDE	*Integrated Development Environment*
JSON	*JavaScript Object Notation*
MVC	*Model View Controller*
MVVM	*Model View View-Model*
NIB	*NeXT Interface Builder*
SBS	*Shared Bike-Systems*
SCB	Sistemas de Compartilhamento de Bicicletas
SDK	*Software Development Kit*
TI	Tecnologia da Informação
V.I.P.E.R.	*View Interactor Presenter Entity and Routing*
WWDC	*Apple Worldwide Developers Conference*
XIB	*XML Interface Builder*
XML	*Extensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

A necessidade de locomover-se acompanha o ser humano desde os primórdios. Com o desenvolvimento de diversos meios de transporte a humanidade conseguiu espalhar-se por todo o mundo, construindo grandes centros urbanos, o que resultou na criação de complexas redes de transportes de pessoas e cargas. Essa complexa rede abriu espaço para a discussão de uma nova temática: a mobilidade urbana.

Diferentemente do que se possa imaginar a mobilidade urbana é muito mais do que apenas veículos e trânsito, ela se refere ao resultado da interação entre os deslocamentos de pessoas e bens com a cidade (CARTILHA, 2005). O incentivo à obras de infraestrutura adequados para os deslocamentos de pessoas e bens em uma área da cidade pode ajudar a desenvolvê-la.

As condições de mobilidade urbana vêm se degradando nos últimos anos. Estudos realizados no Brasil apontam que o aumento do uso de veículos motorizados privados ocasionou um crescimento dos números de acidentes de trânsito com vítimas, congestionamentos urbanos e também dos poluentes veiculares (CARVALHO, 2016).

A partir desse contexto, iniciou-se o debate sobre os fatores que caracterizariam um modelo de mobilidade urbana sustentável para as grandes cidades e quais caminhos devem ser seguidos para alcançá-la (CARVALHO, 2016). Dessa forma, espera-se contribuir para uma melhor qualidade de vida de seus moradores, mais respeito ao meio ambiente e aumento do índice de desenvolvimento humano dessas cidades.

Para promover a mobilidade urbana sustentável é imprescindível a redução do uso de veículos motorizados privados (SILVEIRA, 2010), além de estimular a adoção de práticas sustentáveis no dia a dia, como as caminhadas, o uso do transporte coletivo e das bicicletas. Cidades com maior percentual de viagens com os transportes citados apresentam menores níveis de poluição atmosférica e sonora, além de serem mais funcionais do que as cidades com alto grau de uso de veículos motorizados privados (CARVALHO, 2016). Em viagens de curta distância, a bicicleta se apresenta como a melhor opção de deslocamento, trazendo vantagens tanto para o ciclista quanto para o meio ambiente e a comunidade em geral (SILVEIRA, 2010).

Pesquisas apresentam que o ciclismo é uma forma de atividade física que melhora a aptidão cardiorrespiratória e metabólicas do corpo, proporcionando muitos benefícios à saúde. Para adultos e idosos existem várias vantagens no ciclismo, como: menor risco de morte prematura, desenvolvimento de doenças cardíacas, acidente vascular cerebral, diabetes tipo 2,

hipertensão arterial e depressão; proporciona uma melhor função cognitiva; previne quedas; entre outras diversas vantagens para a saúde (OJA *et al.*, 2011). Portanto, não é surpresa que "o ciclismo tenha sido reconhecido como um importante meio potencial para promover a saúde pública." (BAUMAN; RISSEL, 2009).

No intuito de viabilizar o uso de bicicletas como meio de transporte, incentivando a mobilidade urbana sustentável, surgiram os Sistemas de Compartilhamento de Bicicletas (SCB) em grandes cidades do mundo. Desde o início do século XXI, os SCB se desenvolveram e se difundiram, criando uma nova forma de mobilidade em cidades de todo o mundo. Cada vez mais as prefeituras das cidades estão adotando esse tipo de sistema como estratégia para multiplicar as opções de viagem, promover o uso de modos ativos de transporte, diminuir a dependência do veículos motorizados privados e especialmente reduzir as emissões de gases poluentes (CONTARDO *et al.*, 2012).

Os SCB estão se espalhando pelo mundo: em junho de 2014, estima-se que os SCB estavam disponíveis em 50 países nos cinco continentes, incluindo 712 cidades, com aproximadamente 806.200 bicicletas em operação em 37.500 estações (SHAHEEN *et al.*, 2014).

Uma grande parte dos SCB existentes hoje utilizam da TI para a sua operação, onde o usuário faz um cadastro prévio em um sistema virtual e aluga as bicicletas por meio de aplicativos ou cartões. A utilização da TI para o gerenciamento e aluguel de bicicletas desses sistemas abre uma gama de possibilidades para a criação de ferramentas digitais auxiliares para o aluguel, avaliação, busca e escolha das bicicletas. Além disso, as informações referentes às bicicletas, seu compartilhamento e às estações são disponibilizadas por *Application Programming Interfaces* (API) ou serviços digitais.

1.1 Objetivos

Tendo como pressuposto o acesso em tempo real à informações de tais SCB, é levantada a questão: como um aplicativo móvel pode auxiliar no uso de bicicletas compartilhadas como meio sustentável de transporte em grandes cidades? Para responder tal questionamento o presente trabalho tem como objetivo geral desenvolver um aplicativo móvel para a identificação de estações de bicicletas compartilhadas em grandes cidades do mundo.

Como apresentado, o incentivo à uma mobilidade urbana sustentável é uma preocupação dos atuais governos de cidades, a fim de promover uma melhor qualidade de vida para a população. Portanto, o aplicativo desenvolvido neste trabalho poderá auxiliar no uso

de bicicletas compartilhadas para viagens de média e curta distância, assim, incentivando a mobilidade urbana sustentável, desafogando o trânsito, diminuindo a emissão de poluentes no ar e trazendo benefícios para a saúde dos ciclistas.

Como objetivos específicos, este trabalho visa: (1) definir um conjunto de funcionalidades que sejam relevantes para o usuário final; (2) especificar o uso das técnicas e padrões utilizados no desenvolvimento do aplicativo; (3) desenvolver um aplicativo iOS que permita ao usuário final identificar estações de bicicletas presentes em algumas das cidades dos cinco continentes; (4) e, por fim, disponibilizar na *App Store* (loja de aplicativos da empresa *Apple Inc.*) o aplicativo final desenvolvido para que qualquer usuário de dispositivos *iOS* possa acessá-lo gratuitamente.

2 REFERENCIAL TEÓRICO

Nesta seção, reunem-se todos os conceitos necessários para a compreensão deste trabalho, além do contexto histórico da temática. São explicados conceitos de mobilidade urbana e mobilidade urbana sustentável, o como funcionam os SCB, seu histórico e os principais elementos que os compõem.

2.1 Mobilidade urbana

Com a revolução industrial iniciada no século XX ocorreu o deslocamento de milhares de pessoas do campo para a cidade, ocasionando o rápido crescimento de centros urbanos. A rotina dos antigos camponeses se transformou completamente, uma nova relação com o trabalho foi estabelecida e surgiu a necessidade de se deslocar para as fábricas todos os dias para trabalhar. É nessa época também que se iniciaram as emissões de CO_2 na atmosfera, período onde as cidades eram conhecidas por serem sempre cinzas por conta da fumaça emitida pelas fábricas (HALL, 1998). O crescente desenvolvimento de centros urbanos iniciou uma preocupação com a mobilidade urbana e o desenvolvimento de políticas públicas que garantam ao cidadão o seu direito de ir e vir.

O conceito de mobilidade urbana é bem amplo. A cartilha do ministério das cidades do Brasil de 2005, traz a seguinte afirmação sobre a mobilidade urbana:

> Pensar a mobilidade urbana é, portanto, pensar sobre como se organizam os usos e a ocupação da cidade e a melhor forma de garantir o acesso das pessoas e bens ao que a cidade oferece (locais de emprego, escolas, hospitais, praças e áreas de lazer) não apenas pensar os meios de transporte e o trânsito (CARTILHA, 2005).

Algumas definições tratam a mobilidade urbana como um mecanismo de equilíbrio entre os recursos da cidade, proporcionando uma melhor distribuição de oportunidades para todos. Para possibilitar que o cidadão tenha uma escolha quanto ao modo que deseja utilizar para se locomover é preciso que lhe proporcionem uma infraestrutura planejada de acordo com suas necessidades, já que mobilidade é:

> um aspecto essencial à qualidade de vida da cidade, primeiro, por ser um fator essencial para todas as atividades humanas; segundo, por ser um elemento determinante para o desenvolvimento econômico e para a qualidade de vida; e, terceiro, pelo seu papel decisivo na inclusão social e na equidade da apropriação da cidade e de todos os serviços urbanos (SEMOB, 2004).

Está intrínseco no conceito de mobilidade urbana que ela deve ser projetada para alcançar todos os cidadãos da cidade, de forma justa. Porém, a mobilidade urbana se tornou um grande problema nas cidades brasileiras e em outras partes do mundo. Uma forma de contornar esse problema é colocar o pedestre em primeiro lugar, como se propõe a mobilidade urbana sustentável, favorecendo os modos de transporte coletivo motorizado e o individual não-motorizado.

2.1.1 A mobilidade urbana sustentável

A mobilidade urbana sustentável pode ser definida como um conjunto de políticas públicas de circulação e transporte que visa proporcionar o acesso amplo e democrático aos espaços urbanos, priorizando os meios de transportes coletivos motorizados e individuais não-motorizados. Sugere de forma concreta que não existam segregações espaciais, mas sim a mobilidade urbana seja socialmente inclusiva e ecologicamente sustentável (SEMOB, 2004).

Não refere-se apenas ao ato de se locomover, a mobilidade urbana sustentável é imprescindível para promover ao cidadão uma facilidade de acesso à oportunidades e funções econômicas e sociais da cidade. Priorizando o transporte público coletivo viabiliza-se a inclusão social, redução de congestionamentos, acidentes e poluição atmosférica.

> As cidades que implementam políticas sustentáveis de mobilidade oferecem um maior dinamismo das funções urbanas, numa maior e melhor circulação de pessoas, bens e mercadorias, que se traduzem na valorização do espaço público, na sustentabilidade e no desenvolvimento econômico e social (BERGMAN; RABI, 2005).

2.2 Sistemas de compartilhamento de bicicletas

Visando dar a oportunidade para os cidadãos de se locomoverem de forma sustentável, surgiram os SCB. Desde o início do século XX a bicicleta já era considerada como um meio de transporte ideal para se percorrer pequenas e médias distâncias. Nos dias atuais ela tem se mostrado como uma solução para a crise de mobilidade dos centros urbanos, tanto para realização viagens de pequenas e médias distâncias quanto como um meio de transporte intermediário entre viagens. O ciclismo é um modo de transporte não poluente, preserva os espaços públicos, não exige a reserva de grandes áreas para estacionamento, não utiliza combustível fóssil, é barato e não provoca congestionamentos e emissão de gases poluentes como acontece com os veículos motorizados privados (PEZZUTO; SANCHES, 2004).

Pesquisas apontam que os SCB estão cada vez mais aumentando seu número de cidades e estações: em 2009, estima-se que cerca de 100 SCB foram implementados em 125 cidades do mundo, totalizando mais de 140.000 bicicletas (SHAHEEN *et al.*, 2010). Em junho de 2014, os SCB estavam disponíveis em 50 países nos cinco continentes, incluindo 712 cidades, com aproximadamente 806.200 bicicletas em operação em 37.500 estações (SHAHEEN *et al.*, 2014).

2.2.1 Evolução dos SCB

A evolução dos SCB é categorizada em três gerações (SHAHEEN *et al.*, 2010): (1) *white bikes*, bicicletas brancas em português, esses sistemas eram caracterizados por um conjunto de bicicletas que eram disponíveis sem custos para o usuário. Normalmente, as estações eram localizadas perto de estações de metrô ou postos públicos que tinham sua própria equipe e eram responsáveis pela identificação dos usuários. O primeiro sistema de compartilhamento de bicicletas surgiu em Amsterdã, Holanda, em 1965. O sistema falhou após o seu lançamento devido a danos de bicicleta e roubos; (2) sistemas de depósito de moedas, onde as bicicletas não estavam disponíveis gratuitamente, uma vez que os usuários tinham que usar uma moeda para destravar a bicicleta das estações. Nessa geração iniciou-se a preocupação sobre a localização das estações de bicicletas para garantir a eficiência do sistema. O sistema de depósito de moedas não resolveu o problema dos roubos; e (3) sistemas baseados em TI (telefones celulares, cartões magnéticos, *smartcards* ou códigos) para desbloquear as bicicletas das estações, permitindo a identificação automática dos usuários. Os usuários casuais pagam um depósito de segurança para garantir o retorno da bicicleta e o uso das bicicletas é pago dependendo do tempo de uso. Normalmente, o serviço é gratuito no primeiro intervalo de tempo especificado e o preço aumenta gradualmente após o esgotamento do tempo. Esse sistema é mais simples de gerenciar em termos de recursos humanos, porém requer um investimento maior em tecnologia. Algumas das grandes vantagens da introdução da tecnologia são a possibilidade de atendimento 24 horas, a facilidade de localização das estações na cidade e a coleta de dados sobre o uso do serviço.

Shaheen, Guzman e Zhang (2010) identificaram também a quarta geração de SCB: os SCB multimodais. Sua principal preocupação é a melhoria do atendimento às necessidades do usuário, ou seja, funciona à demanda. Inclui uma melhoria nos mecanismos tecnológicos nas estações e bicicletas que facilitam a sua utilização e compartilhamento, bicicletas elétricas, localizações de bicicletas e a integração dos vários serviços de transporte no mesmo cartão de

acesso (transporte público ou compartilhamento de carros).

2.2.2 Componentes essenciais de um SCB

Segundo Peter Midgley (MIDGLEY, 2011) os SCB são tipicamente compostos pelos seguintes componentes: (1) as bicicletas. As bicicletas compartilhadas precisam ser fáceis de usar, adaptáveis a usuários de diferentes tamanhos, mecanicamente confiáveis, resistentes a vandalismo ou roubo e com aparência diferenciada (fácil de distinguir); (2) as estações. Podendo ser fixa-permanente, fixa-portátil ou flexível. As estações são os locais onde as bicicletas são trancadas em *racks* próprios quando não estão em serviço; (3) um sistema digital e registro do usuário. Para acessar as bicicletas nas estações, os usuários precisam desbloqueá-la. O usuário pode usar de um cartão para liberar a bicicleta ou então se conectar via celular e desbloquear a bicicleta por meio de um aplicativo ou *site*; (4) sistemas de informação com *status*. São sistemas digitais que fornecem informações em tempo real sobre a disponibilidade de bicicletas para cada estação. Alguns incluem mapas com ciclovias marcadas, além de fornecer atualizações meteorológicas; (5) sistema de manutenção. Administra a manutenção e logística das estações, recolhendo bicicletas defeituosas e corrigindo problemas de TI; (6) os mecanismos de redistribuição de bicicletas. Para garantir que uma SCB funcione otimizadamente é preciso antecipar as demandas de viagens, tendo em vista que essa demanda normalmente é assimétrica. É utilizado um veículo que transporta as bicicletas de estações lotadas para estações vagas. Esse mecanismo é usado para reequilibrar as demandas do sistema.

O componente 4 dos SCB traz a possibilidade de empresas e desenvolvedores independentes construírem ferramentas digitais para o auxílio da utilização das bicicletas e estações do sistema.

3 CONTEXTO DO DESENVOLVIMENTO

O presente trabalho está divido em três grandes etapas que são: (1) apresentação do contexto do desenvolvimento do aplicativo; (2) modelagens e protótipos; e (3) a codificação do aplicativo.

Nesta seção é descrito o contexto do desenvolvimento do aplicativo móvel *iOS* para o auxílio do uso de bicicletas compartilhadas como um meio de mobilidade sustentável. Traz um resumo sobre a evolução dos *smartphones* e o uso dos aplicativos móveis; um breve histórico sobre o *iPhone* e seu sistema operacional *iOS*, e a sua atual participação no mercado mundial; apresenta os dispositivos, programas e plataformas utilizados para o desenvolvimento do aplicativo, suas respectivas versões e configurações; por fim, apresenta a *API CityBikes* que é a fornecedora dos dados em tempo real necessários para a construção do aplicativo.

3.1 Aplicativos móveis

No início da *internet* no Brasil, na década de 1990, o seu acesso era realizado majoritariamente pelo uso de computadores e, em escala muito menor, dos telefones celulares. Porém, é notório que a posse e o uso de dispositivos móveis vem aumentando a cada ano no Brasil. Um pesquisa do Instituto Brasileiro de Geografia e Estatistica (IBGE) divulgada em fevereiro de 2018 identificou que o celular estava presente em 92,6% dos 69,3 milhões de domicílios pesquisados. Entre os usuários de *internet* com 10 anos ou mais de idade, 94,6% se conectaram via celular. Da população de 10 anos ou mais, 77,1% tinham celular para uso pessoal (IBGE, 2016).

Por meio de pesquisas como essa é possível perceber que o uso da *internet* está migrando para os dispositivos móveis. Isso só foi possível com os avanços tecnológicos que resultaram na transformação dos telefones celulares típicos dos anos 90 em modernos *smartphones* com larga capacidade de processamento.

Atualmente, no Brasil, já existem mais *smartphones* ativos do que habitantes. Um pesquisa realizada pela Fundação Getúlio Vargas revelou que são 220 milhões de celulares em funcionamento no país, contra 207,6 milhões de habitantes - de acordo com os dados mais recentes do IBGE (EAESP, 2018). Entretanto, não é a quantidade de dispositivos que é o mais impressionante, mas sim o avanço tecnológico que estes passaram. Hoje, os *smartphones* funcionam como computadores de bolso com capacidade de processamento semelhante ou

superior aos computadores do início da década de 2010.

A empresa *Apple Inc*. afirmou em um de seus eventos de lançamento em outubro de 2018 que seus novos *iPads Pros* são mais rápidos do que 92% de todos os *laptops*, *tablets* e *Personal Computers* (*PCs*) comercializados em 2017, isso inclui *PCs* com processadores *Intel Core i7*, um dos mais potentes do mercado.

Essa alta capacidade de processamento aumentou a possibilidades de aplicativos que podem rodar nesses aparelhos. Hoje é bastante comum encontrar aplicativos com realidade aumentada, jogos com magníficos gráficos 3D ou aplicativos com diversas requisições *web* em tempo real. Uma recente pesquisa revelou que, juntas, as lojas de aplicativos *App Store* e *GooglePlay* possuem mais de 5,7 milhões de produtos disponíveis (APPFIGURES, 2018).

3.2 Desenvolvimento móvel para *iOS*

O ano de 2007 foi marcado pelo lançamento do primeiro dispositivo móvel da *Apple Inc.*, o *iPhone*. O dispositivo possuía capacidade de realizar ligações, acessar a *internet*, ouvir música, dentre diversas outras funcionalidades. A empresa revolucionou o mercado de telecomunicações mundial ao reinventar e redefinir o conceito de *smartphone*. Desde que Steve Jobs anunciou o *iPhone* em 2007, a *Apple* vendeu cerca de 1,5 bilhão deles (STATISTA, 2018), criando oportunidades de negócios gigantes para desenvolvedores de aplicativos e fabricantes de acessórios, tornando-o um dos *smartphones* mais usados no mundo.

Seis meses após o seu lançamento em 2007, o *iPhone* começou a ser comercializado. As pessoas fizeram fila para comprá-lo e a *Apple* vendeu 270 mil *iPhones* somente no primeiro fim de semana. Um ano depois foi lançado o *iPhone* 3G que possuía suporte para redes de *internet* 3G, oferecendo acesso muito mais rápido a *e-mails* e páginas *web*. Porém, o mais relevante lançamento do ano de 2008 foi a *App Store*, a loja de aplicativos para *iPhone*. A loja permitiu que desenvolvedores de todo o mundo criassem e vendessem *software* para milhões de proprietários de *smartphones*. Essa contribuição foi uma das mais importantes para a indústria de tecnologia e a sociedade em geral, pois permitiu que um novo ramo de mercado fosse criado: o desenvolvimento de aplicativos móveis.

3.2.1 Configurações do ambiente de desenvolvimento

Esta seção informa a descrição dos programas e plataformas *online* usados para o desenvolvimento, nas tabelas 2 e 3 respectivamente. Informa também os atributos de *hardware* do equipamento utilizado, na tabela 1.

Tabela 1 – *Hardware* utilizado.

Item	Valor
Nome do Modelo	MacBook Pro
Identificador do Modelo	MacBookPro 14,1
Nome do Processador	Dual-Core Intel Core i5
Velocidade do Processador	2,3 GHz
Número de Processadores	1
Número Total de Núcleos	2
Cache L2 (por Núcleo)	256 KB
Cache de L3	4 MB
Tecnologia Hyper-Threading	Ativado
Memória	8 GB
Versão da ROM de Inicialização	202.0.0.0.0
Versão do SMC (sistema)	2.43f7
Gráficos	Intel Iris Plus Graphics 640 1536 MB
Armazenamento	251 GB Flash
Monitor	13,3 polegadas (2560 x 1600)

Fonte: Elaborado pela autora (2019).

Tabela 2 – *Softwares* utilizados.

Software	Descrição
Sistema Operacional	MacOS Mojave 10.14 (18A391)
XCode	Versão 11.0 (11A420a)
Sketch	Versão 56.3 (2019)

Fonte: Elaborado pela autora (2019).

Tabela 3 – Ferramentas *online*.

Site	Endereço
GitHub	github.com
Overleaf	overleaf.com
Google Drive	drive.google.com
Lucidchart	lucidchart.com

Fonte: Elaborado pela autora (2019).

3.3 API *CityBikes*

Para acessar informações em tempo real sobre os SCB, foi utilizada a *Application Programming Interface* (API) *CityBikes*. A API fornece informações sobre as cidades que possuem SCB, como nome, país, latitude e longitude; informações sobre as estações espalhadas pelas cidades, como nome, endereço, latitude e longitude; *status* dessas estações; quantidades de bicicletas disponíveis e quantidade de vagas em cada estação; nome e *site* da empresa responsável; dentre vários outros dados opcionais que variam entre cada SCB.

3.3.1 História do CityBikes

Há alguns anos um grupo de desenvolvedores de Barcelona, Espanha, decidiu criar a *CityBikes* após a tentativa frustrada de produzir um aplicativo para o seu SCB local, pois se depararam com o problema de não terem acesso centralizado aos dados abertos. Descobriram que outros sistemas ao redor do mundo tinham exatamente o mesmo problema. O *CityBikes* foi criado a fim de ser uma API que fornece dados abertos de SCB para aplicativos, pesquisas e projetos que desejem fazer seu uso pra análise, estatísticas ou aplicativos interativos.

Os criadores da API acreditam que qualquer projeto resultante de uma parceria público-privada deve disponibilizar os dados exatamente onde eles pertencem: ao público. O principal objetivo foi liberar os dados para que fossem construídos vários aplicativos para diversas plataformas, não apenas o aplicativo inicial que os revelou o problema da inacessibilidade desses dados.

Até julho de 2018, o *CityBikes* suportava mais de 400 cidades e era conjunto de dados mais amplamente usado para a construção de projetos de mobilidade sobre compartilhamento de bicicletas. O *CityBikes* faz uso da biblioteca *PyBikes* para acessar as informações dos SCB.

3.3.2 PyBikes

A *PyBikes* é um biblioteca escrita em Python que fornece um conjunto de ferramentas para coletar dados de SCB de diferentes sites e APIs, fornecendo, assim, um conjunto coerente de classes e métodos para acessar as informações sobre as estações.

A biblioteca foi criada e distribuída com o intuito de fornecer dados abertos e gerar estatísticas para serem usados em sistemas que desejem compartilhar esses dados. Porém, o papel mais importante da biblioteca é alimentar o projeto *CityBikes*.

O projeto é *open-source* e qualquer desenvolvedor da comunidade pode contribuir para ele. Pode ser acessado por meio do endereço: <https://github.com/eskerda/pybikes>.

3.3.3 Informações e documentação da API

A API *CityBikes* possui dois *endpoints*. O primeiro é o que fornece informações sobre todas as cidades suportadas. Endereço do *endpoint* é <http://api.citybik.es/v2/networks> e ele fornece um objeto *JavaScript Object Notation* (JSON) com uma lista contendo mais de 400 cidades e as suas seguintes informações:

`name`: Nome do SCB

`company`: Nome da empresa responsável

`href`: Endereço completo do *network endpoint*

`id`: ID único para a cidade

`city`: Nome da cidade

`country`: Abreviação do país

`latitude`: Coordenada de latitude da cidade

`longitude`: Coordenada de longitude da cidade

Código-fonte 1 – JSON de retorno no *endpoint* de cidades.

```
{
  "networks": [
    {
        "name": "Bicicletar"
        "company": "Mobilicidade Tecnologia LTD",
        "href": "/v2/networks/bicicletar",
        "id": "bicicletar",
        "location": {
            "city": "Fortaleza",
            "country": "BR",
            "latitude": -3.7321944,
            "longitude": -38.510347
        },
    },
    ...
  ]
}
```

O segundo *endpoint* é acessado pelo endereço <http://api.citybik.es/v2/networks/bicicletar>, onde a última parte do caminho se refere ao ID da cidade específica pesquisada. Ele traz informações sobre a cidade, além de uma lista com todas as suas estações e outras seguintes informações:

name: Nome da estação

timestamp: A data em que a estação foi atualizada

latitude: Coordenada de latitude da estação

longitude: Coordenada de longitude da estação

free_bikes: Número de bicicletas disponíveis

empty_slots: Número de vagas disponíveis

id: ID único para a estação

extra: Informações adicionais

Código-fonte 2 – JSON de retorno no *endpoint* de única cidade.

```
{
  "network": {
    "name": "Bicicletar",
    "stations": [
    {
        "name": "Reitoria UFC",
        "timestamp": "2019-10-07T23:31:13.588000Z"
        "latitude": -3.7410964,
        "longitude": -38.539249,
        "free_bikes": 4,
        "empty_slots": 8,
        "id": "8e889612b3859ec14833a9fbbead3eb5",
        "extra": {
            "address": "UFC",
            "description": "Avenida da Universidade, 2854 / Esquina Av. 13 de Maio",
            "status": "open"
        }
      },
      ...
    ]
  }
}
```

4 MODELAGEM E PROTÓTIPOS

O aplicativo proposto por este trabalho inicialmente possuiu o nome *BikeStyle*. Foi desenvolvido pela própria autora em novembro de 2018 e postado na loja de aplicativos da *Apple* em março de 2019. A versão 1.0 do aplicativo possuía um escopo reduzido e alguns *bugs* a serem corrigidos. Para este trabalho foi desenvolvida uma versão 1.1 do mesmo aplicativo, em que ele foi renomeado para *Bikezera*. Na versão 1.1 do aplicativo foram adicionadas diversas novas funcionalidades, além de corrigidos os problemas identificados na versão 1.0.

Nesta seção são descritos itens relativos à ambas versões (1.0 e 1.1), como o modelo conceitual e análise do potencial usuário. Descreve-se também suas funcionalidades e protótipos de média fidelidade, onde nessas partes serão destacadas quais são referentes à versão 1.0 e quais são referentes à versão 1.1.

4.1 Modelo conceitual do sistema

Modelo conceitual é uma representação ou interpretação de ideias, pensamentos ou definições sobre algo. Ao criar um modelo conceitual não são usados termos técnicos ou especificações de interface. Ele representa o sistema proposto, de forma conceitual a respeito do que ele deve fazer, de como deve se comportar e com o que deve se parecer, de forma que seja compreendido por um usuário ou pessoa que ainda não tenha profundo conhecimento sobre o produto.

De forma mais concreta, o modelo conceitual é um diagrama que apresenta as relações entre diferentes conceitos abstratos - normalmente representados por substantivos. Esses conceitos abstratos são ligados uns aos outros por meio de relações - normalmente descritas por verbos - que vão indicar como eles interagem entre si.

Na Figura 1 está a proposta do modelo conceitual do aplicativo *Bikezera* no momento da sua idealização.

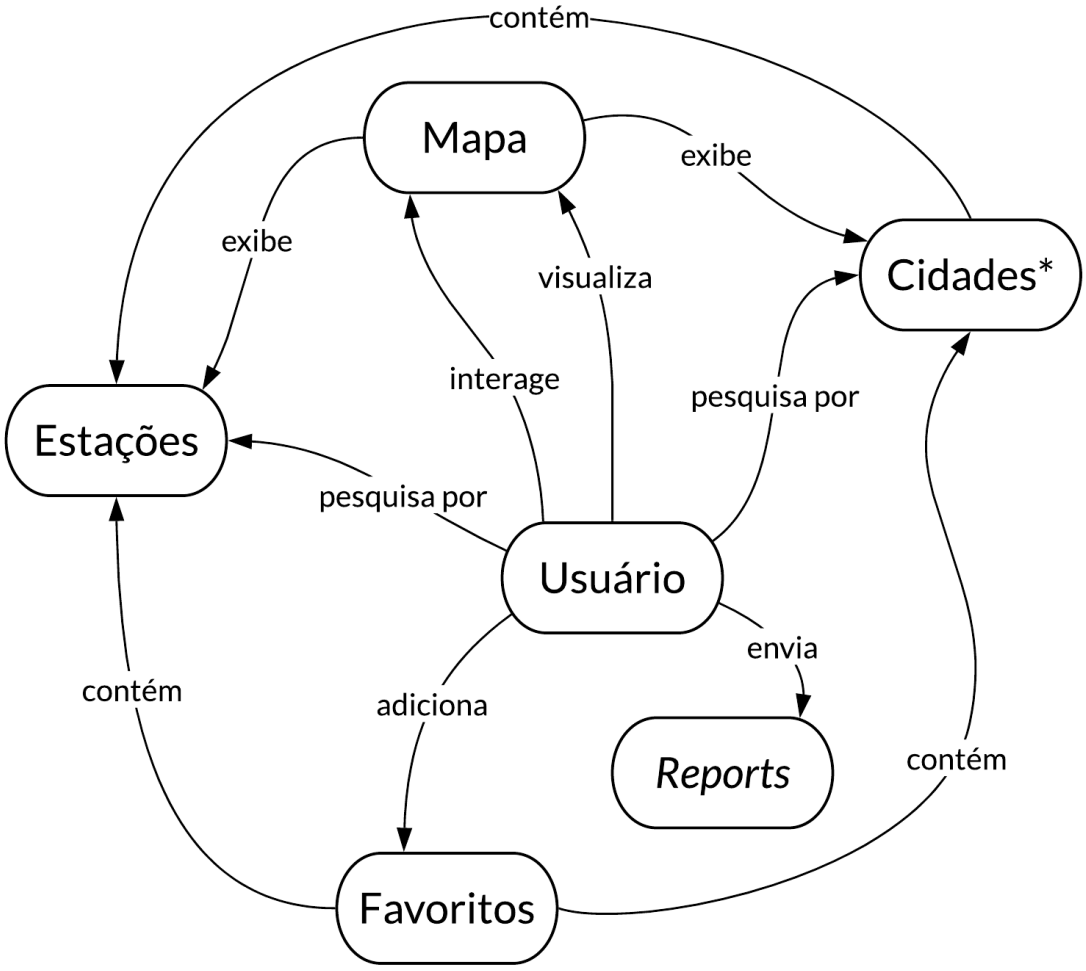
4.2 Análise do usuário

A fim de tornar o produto relevante para o usuário final, sentiu-se a necessidade de entender melhor sobre as suas dores; quais os seus objetivos ao usar um serviço de bicicletas compartilhadas e qual seu contexto de uso.

Por isso foi elaborado uma persona e um contexto de uso, para que a interface e as

funcionalidades pudessem ser construídas de forma a suprir as necessidades do usuário.

Figura 1 – Diagrama do modelo conceitual do aplicativo.



* Também podem ser referenciadas como *networks*

Fonte: Elaborado pela autora (2019).

4.2.1 Persona

Foto Disponível na Figura 2

Nome Gabriela Machado

Idade 22 anos

Ocupação Estagiária de arquitetura

Estado civil Solteira

Mora Com uma prima

Ganha R$ 1.500/mês

Figura 2 – Foto da persona.

Fonte: Unsplash (2019)

Bio Gabriela mora em Recife e está no último ano do curso de arquitetura e ama andar pela cidade para visitar prédios antigos, edifícios modernos e igrejas a fim de entender melhor sobre a sua arquitetura. Ela mora perto do escritório onde trabalha, então costuma ir a pé. Gosta bastante de viajar pelo Brasil e as vezes para fora do país, nessas horas ela gosta de fazer seus passeios de bicicleta, pois se sente mais conectada com a cidade. Nessas viagens ela costuma alugar bicicletas em empresas locais. Em seus momentos livres gosta de sair com os amigos para a praia ou bares.

Frustrações A insegurança na cidade. Não possuir uma bicicleta. O estresse do estágio.

Objetivos Ter mais tempo para viajar. Formar-se na faculdade. Conseguir estabilidade financeira.

4.2.2 Contexto de uso

Gabriela foi viajar para Salvador para um congresso de arquitetura. Aproveitando uma tarde livre, ela decidiu alugar uma bicicleta para poder visitar o centro histórico da cidade. Como era uma cidade nova, ela não tinha certeza do quão seguro seria. Junto de seu amigo Lucas, Gabriela entrou em um pequeno restaurante (Odoyá) no pelourinho para usar seu celular se sentindo mais segura. Ela abriu o aplicativo *Bikezera* e selecionou a cidade de salvador no mapa para ver suas estações. Identificou que a estação mais próxima ficava a 300 metros de onde ela estava, na praça da Sé, e que tinham 5 bicicletas disponíveis. Então ela selecionou a opção de visualizar a rota a pé da sua localização atual até a estação da praça da Sé, decorou a rota, guardou o celular e caminhou até a estação com seu amigo.

4.3 Descrição das funcionalidades

As funcionalidades do aplicativo foram baseadas principalmente nas informações fornecidas pela API *CityBikes*. Além disso, também foi considerado o tempo disponível para o desenvolvimento e os recursos disponibilizados pelo `MapKit`, que será explanado na seção 5.3. O conjunto de funcionalidades selecionadas buscam entregar ao usuário funções com utilidade e valor.

Na versão 1.0 do aplicativo estas eram as funcionalidades que ele possuía:

[F-01] Visualizar cidades

No mapa principal do aplicativo o usuário visualizará marcadores nas localizações das cidades que possuem SCB registrados na API *CityBikes*. Nos marcadores estão informações do nome da cidade em destaque e o nome da empresa responsável pelas bicicletas naquela cidade.

[F-02] Pesquisar por cidades

O usuário é capaz de pesquisar por cidades digitando seu nome ou o nome da empresa responsável pelo SCB local.

[F-03] Visualizar estações

Após selecionar uma cidade o usuário visualiza marcadores nas localizações de cada estação de bicicleta disponível para aquela cidade. Nos marcadores estão informações do nome da estação em destaque e a quantidade de bicicletas e vagas disponíveis nela.

[F-04] Pesquisar por estações

O usuário pode pesquisar por cidades digitando seu nome.

[F-05] Visualizar detalhes das estações

O usuário pode visualizar o nome da estação; nome da empresa responsável pelo SCB local, número de bicicletas disponíveis, número de vagas disponíveis e endereço detalhado.

[F-06] Traçar rota do local atual para a estação

É possível utilizar o *Maps*, aplicativo nativo para mapas do *iOS*, para traçar uma rota da localização atual do usuário para a estação desejada.

[F-11] Recarregar conteúdo

O usuário pode atualizar o conteúdo do mapa quando desejar, buscando informações mais recentes da API *CityBikes* acerca das cidades e estações.

Para a versão 1.1 do aplicativo *Bikezera* foram adicionadas funcionalidades que identificou-se ser de grande relevância para uma melhor experiência do usuário. Essas funci-

onalidades foram implementadas durante o período de construção deste presente trabalho. As funcionalidades adicionadas foram:

[F-07] Função bússola

O usuário poderá ativar e desativar uma função bússola durante o uso o aplicativo. A função se utiliza do sensor magnetômetro do *iPhone* e indica para qual ponto cardeal o usuário está apontando o aparelho.

[F-08] Visualizar localização atual

Caso o usuário permita que o aplicativo acesse sua localização atual, será possível visualizar sua localização no mapa em tempo real.

[F-09] Adicionar conteúdo aos favoritos

O usuário poderá adicionar cidades e estações à sua lista de favoritos. A lista ficará salva na memória local do celular e será perdida caso o aplicativo seja desinstalado.

[F-10] Visualizar favoritos

Após adicionar conteúdo aos favoritos, o usuário poderá visualizar os elementos salvos em uma lista.

[F-12] Enviar comentários para a equipe de desenvolvimento

O usuário poderá enviar comentários com sugestões, críticas, elogios ou informando *bugs* durante o uso do aplicativo.

Além do acréscimo dessas novas funcionalidades, na versão 1.1 do aplicativo foi acrescentada a tradução para o Português. O processo de internacionalização do aplicativo será melhor descrito na seção 5.5. As correções feitas a partir da versão 1.0 do produto são listadas no seção 7.

4.4 Prototipagem da interface

A interface do aplicativo passou por uma fase de prototipação, onde foram elaborados protótipos de média fidelidade junto ao diagrama de fluxo de telas. Foram usadas as diretrizes do *Human Interface Guidelines* (HIG) para guiar essa etapa (APPLE, 2019c). Nesta seção é apresentado o HIG e os resultados de prototipação.

4.4.1 Componentes do HIG

O HIG é o manual de diretrizes de *design* que todos os aplicativos postados na *App Store* devem seguir. O objetivo é prover aos desenvolvedores um conjunto básico de boas práticas de *design* e, assim, melhorar a experiência para os usuários, tornando as interfaces dos aplicativos mais intuitivas e consistentes.O HIG permite que crie-se uma experiência consistente em todo o ecossistema de *software* da *Apple*, fazendo com que o usuário sinta-se mais acostumado com os padrões da interface.

Os aplicativos que não seguirem essas recomendações estão sujeitos à serem rejeitados durante o processo de revisão para serem postados na *App Store*. A *Apple* é conhecida por ser bastante rigorosa com os aplicativos que são publicados em sua loja, principalmente em questão de privacidade e segurança de dados. O manual pode ser acessado pelo endereço <https://developer.apple.com/design/human-interface-guidelines/>.

A fim de garantir a consistência na plataforma *iOS* e maximizar o impacto e o alcance, o HIG apresenta seis conceitos que os desenvolvedores e *designers* devem manter em mente quando estão construindo novos aplicativos:

Integridade estética Representa a relação entre a aparência do aplicativo com a sua função. Projetar de modo que a estética do aplicativo seja condizente com a tarefa que ele se propõe a cumprir.

Consistência Envolve implementar padrões e paradigmas familiares ao usuário, usando componentes padrões do sistema; ícones conhecidos; estilos de texto padrão, além de garantir que o aplicativo se comporta da maneira que o usuário espera.

Manipulação direta Com a manipulação direta, os usuários podem ver resultados imediatos de suas ações, como por exemplo ao rotacionar o dispositivo ou fazer gestos o conteúdo na tela é afetado.

Feedbacks É sempre muito importante manter o usuário atualizado sobre o estado atual do aplicativo, seja um erro ou ação completada com sucesso.

Metáforas O usuário aprende mais rápido quando os objetos e ações do aplicativo são metáforas de experiências familiares, sejam elas enraizadas no mundo real ou digital.

Controle de Usuário No *iOS* as pessoas estão no controle, não o aplicativo. A tomada de decisão deve ser assumida pelo usuário. O aplicativo pode alertar sobre ações destrutivas, mas não deve proibir o usuário de fazê-las (APPLE, 2019c).

Além de orientações sobre *design*, o HIG explica como usar os componentes disponi-

bilizados pelo `UIKit`, *framework* de componentes visuais da *Apple*. O `UIKit` pode ser separado em três categorias: *Bars*; *Views* e *Controls*. O uso de correto de cada um desses componentes podem sem encontrados no HIG. Na Figura 3 estão três exemplos de componentes nativos do *iOS*, uma *View*, uma *Bar* e um *Control*, respectivamente.

Figura 3 – Componentes padrões do *iOS*. `UITableView`, `UITabBar` e `UISwitch`.

Fonte: Human Interface Guidelines, Apple (2019).

O *Bikezera* se utiliza de componentes nativos do `UIKit` para que o usuário de *iPhone* não tenha dificuldade ao manusear o aplicativo. Porém, foi adicionado mais personalidade aos componentes, a fim de que a interface pudesse se destacar positivamente com relação à outros aplicativo similares.

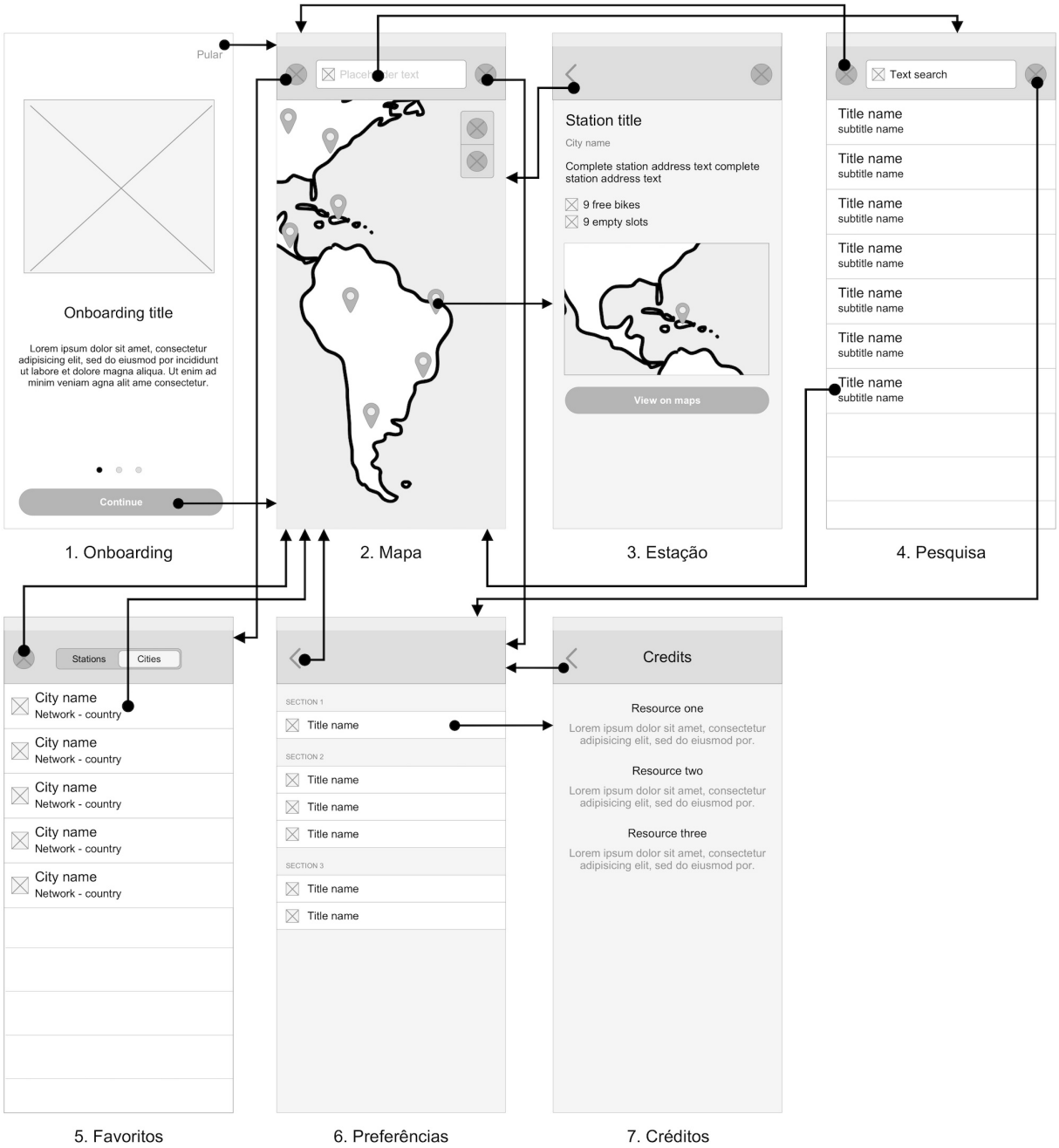
As boas práticas de *design* indicadas pelo HIG foram observadas e seguidas durante todo o planejamento da interação com a interface do aplicativo. Um exemplo de boa prática aplicada ao *Bikezera* é o fato de a solicitação da permissão à localização atual do usuário ser feita apenas depois do tutorial inicial, explicando de forma clara quais as vantagens que o usuário teria ao fornecer esse dado para o aplicativo.

4.4.2 Fluxo e protótipo de telas

Na Figura 4 está o protótipo de média fidelidade das telas que compõem o aplicativo, além do seu fluxo. A extremidade das setas que possuem um círculo indica a área de interação do usuário e a extremidade com um triângulo aponta para a tela que é exibida a partir da interação. Foi usado o programa *Sketch* para o desenvolvimento dos protótipos.

O protótipo apresentado na Figura 4 está atualizado de acordo com as mudanças feitas para a versão 1.1 do produto. Em comparativo com a versão 1.0, foram modificadas as telas 1, 2 e 6; além de adicionada a tela 5.

Figura 4 – Diagrama de fluxo de telas e protótipo de média fidelidade.



Fonte: Elaborado pela autora (2019)

5 IMPLEMENTAÇÃO

Nesta seção é apresentada uma descrição técnica de aspectos abordados na implementação do aplicativo *Bikezera*. Os tópicos abordados são: a arquitetura de *software* escolhida para o desenvolvimento e a justificativa para seu uso; o *framework* `MapKit` e seus componentes ultilizados; a construção de *views* para aplicativos *iOS* usando XIB e *storyboard*; o uso da classe `URLSession` para as requisições HTTP; o processo de internacionalização do aplicativo.

5.1 Arquitetura do *software*

A arquitetura de *sofware* escolhida para o projeto foi a *Model View Controller* (MVC). MVC é uma arquitetura bastante difundida ao redor do mundo desde a sua proposição no final da década de 1970. Criada por Trygve Reenskaug, a arquitetura foi pensada inicialmente para sistemas *web*, porém logo foi incorporada em outros tipos de desenvolvimento de *software*.

A *Apple* indica a MVC como arquitetura principal para o desenvolvimento de aplicativos *iOS*. Em sua documentação ela disponibiliza informações relevantes sobre as responsabilidades de cada camada, além de apresentar padrões de projetos que podem ser agregados à arquitetura a fim de torná-la mais robusta.

Na MVC formulado por Reenskaug existem três camadas que possuem responsabilidades específicas, são elas: *Model*; *View* e *Controller*. As camadas relacionam-se entre si de forma bem determinada. As relações entre as camadas da MVC foram adaptadas pela *Apple* para o desenvolvimento de aplicativos *iOS*. Cada uma das camadas e suas funções, segundo a abordagem da *Apple*, estão listadas abaixo:

Model Contém as estruturas e classes que compõem o modelo da aplicação. Gerencia itens como persistência dos dados, camada de rede, *data sources* e *delegates*, constantes, *helpers* e camadas de abstração.

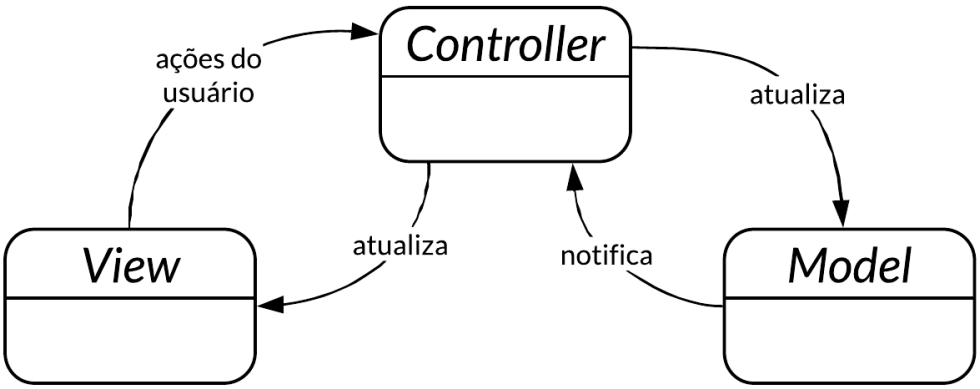
View Contém o código da interface renderizada para o usuário, ou seja `UIKit`, `Core Animation`, `Core Graphics`. Também lida com *inputs* do usuário.

Controller Intermedia a relação entre a *Model* e a *View*. Ela quem decide o que fazer quando o usuário interage com a *View*. Também é ela que atualiza as informações da *View* a partir do que está na *Model*.

O diagrama na Figura 5 representa as relações entre as camadas *Model*, *View* e *Controller* na abordagem da *Apple*. Note que a camada *model* jamais interage com a camada

view.

Figura 5 – Diagrama da arquitetura MVC na abordagem da *Apple*.



Fonte: Elaborado pela autora (2019).

A arquitetura MVC se revelou uma boa opção para o projeto *Bikezera* por alguns motivos: (1) é um aplicativo simples; (2) possui poucas telas; (3) a equipe de desenvolvimento possui apenas um membro; (4) não têm-se a pretensão de se expandir o projeto de forma gigantesca.

A MVC é bastante simples e possui uma enorme quantidade de literatura disponível para consulta, além de ser a arquitetura de *software* indicada pela própria *Apple*. Porém, em casos de aplicativos complexos, com muitas telas ou com um grande time de desenvolvimento, ela provavelmente não seria indicada. Isso ocorre pois ela possui poucas camadas de abstração - apenas três, portanto se torna difícil a separação do código em partes menores. Por isso que, geralmente, para grandes empresas com times enormes existem outras arquiteturas de software mais indicadas como o *Model View View-Model* (MVVM), *Clean Arquitecture* ou *View Interactor Presenter Entity and Routing* (V.I.P.E.R.). Essas possuem várias camadas de abstração, o que possibilita uma melhor aplicação do Princípio da Responsabilidade Unica - um dos cinco princípios da programação orientada à objetos criados por Robert C. Martin.

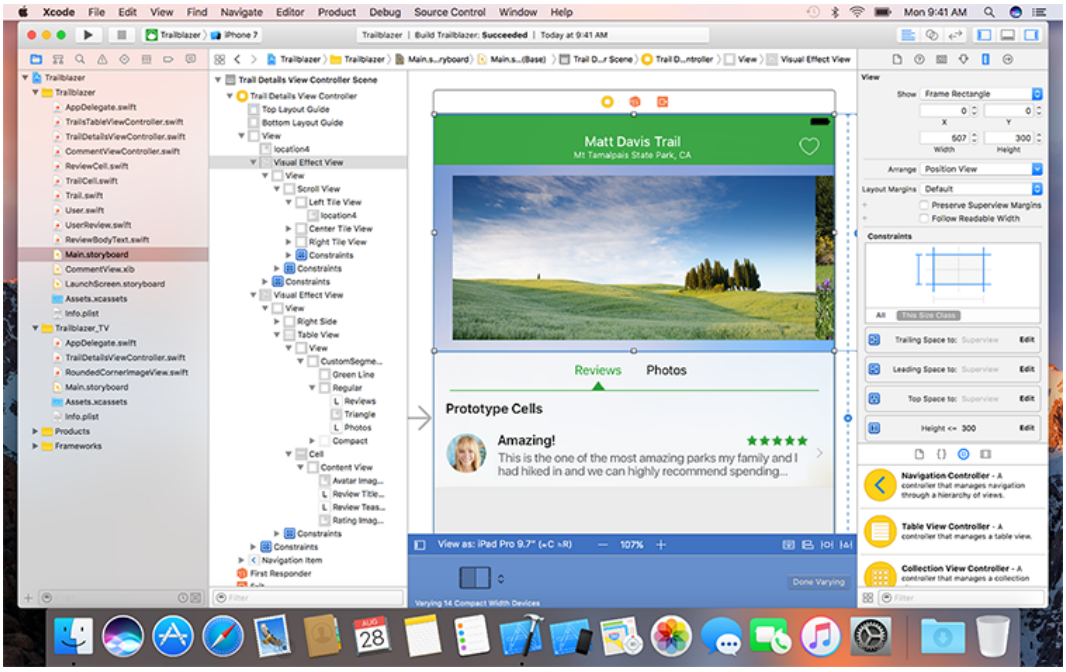
5.2 *XIB*s e *Storyboards*

Como mencionado na seção 3.2, o lançamento da *App Store* no ano de 2008 permitiu que desenvolvedores de todo o mundo pudessem criar seus primeiros aplicativos *iOS* nativos. Atualmente, para o desenvolvimento *iOS* nativo é usado o *XCode* como *Integrated Development Environment (IDE)*. O *XCode* contém um *Interface Builder* integrado que permite aos desenvol-

vedores criar as telas - *Views* - dos aplicativos usando uma interface gráfica. Uma tela construída no *Interface Builder* é armazenada como um arquivo no formato *NeXT Interface Builder* (NIB), XIB ou, mais recentemente, como um arquivo no formato *Storyboard*.

O editor do *Interface Builder* no *XCode* simplifica a construção de telas completas sem ser necessário escrever sequer uma linha de código. O seu uso se dá apenas clicando e posicionando os elementos disponíveis, pode-se visualizar um *screenshot* do *Interface Builder* no *XCode* na Figura 6. Esses elementos são *Views*, *Controls* e *Bars*, descritos na seção 4.4.1.

Figura 6 – *Interface Builder* no *XCode*.



Fonte: Apple Developer, Apple 2019

No início do desenvolvimento *iOS* cada classe que herdasse de *UIViewController* possuía um arquivo XIB associado à ela. Um arquivo XIB é um documento que possui texto no formato *Extensible Markup Language* (XML) que, por sua vez, contém as regras de construção da interface. Porém, a pessoa desenvolvedora não precisa necessariamente entender a sintaxe XML, pois tudo fica abstraído no *Interface Builder*.

Vale ressaltar duas ações que o formato de arquivo XIB não fornece imediatamente: edição manual e fácil *merging*. Um arquivo XIB é uma representação XML de uma tela complexa, logo, é considerado perigoso editá-lo manualmente. Fazer um *merge* entre duas versões do mesmo arquivo XIB, usando ferramentas de versionamento de código como *Git* por exemplo, provavelmente será problemático.

Com o *Software Development Kit* (SDK) do *iOS* 5 a *Apple* introduziu o conceito de

storyboard para simplificar e gerenciar melhor a criação das telas de um aplicativo. Um arquivo no formato *storyboard* é um único arquivo para a construção de todas as telas do aplicativo em desenvolvimento, além de apresentar todos os seus fluxos. É possível adicionar transições entre telas, indicando o modo que serão exibidas. O uso de *storyboard* em um projeto *iOS* minimiza o seu número total de arquivos, além de diminuir a quantidade de código escrito pelos desenvolvedores.

Outra forma para construir *Views* no desenvolvimento para *iOS* é construindo-a programaticamente em *Swift*. Para isso é necessário criar classes que são subclasses de `UIView`. Esse tipo de abordagem é conhecida como *View Code*. Ela é bastante indicada para projetos extensos ou que possuem uma grande equipe trabalhando na mesma base de código. Como cada aspecto das *Views* são programados diretamente pela pessoa desenvolvedora, os *merges* de versões diferentes do código se tornam muito mais fáceis e a reutilização desses componentes também.

O *Bikezera* foi criado utilizando *Storyboards* e XIB. Os motivos para a escolha dessas abordagens foram: (1) é um aplicativo simples; (2) possui poucas telas; (3) a equipe de desenvolvimento possui apenas um membro; (4) as modificações na estrutura e fluxos das telas são visualmente representadas; (5) desenvolvimento mais rápido.

5.3 MapKit

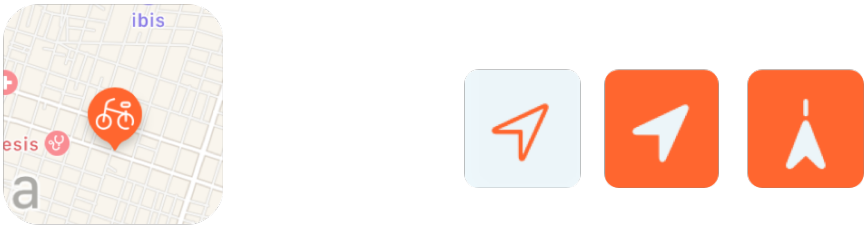
A principal tela do aplicativo *Bikezera* é formada por uma mapa onde são apresentados marcadores com as cidades e estações disponíveis. Para embutir um componente de mapa no aplicativo foi utilizado um *framework* nativo para *iOS*, o `MapKit`. O `MapKit` possui a classe `MKMapView` que incorpora uma visualização de mapa à tela. O *framework* provê diversos métodos e componentes para customizar uma `MKMapView`, alterando sua aparência ou adicionando itens para serem exibidos nela, como compasso e escala.

No aplicativo foram usados os seguintes componentes:

`MKAnnotationView` Um marcador fixado em uma latitude e longitude específicas no mapa. (Figura 7)

`MKUserTrackingButton` Um botão especializado que permite ao usuário alternar entre os modos de rastreamento disponíveis (Figura 7).

Figura 7 – Componentes usados na `MKMapView`.



`MKAnnotationView` `MKUserTrackingButton`

Fonte: Elaborado pela autora (2019).

5.3.1 Localização atual do usuário

O `MapKit` também disponibiliza informações relevantes de rastreamento, como a localização atual do usuário. Para acessar a localização atual do usuário durante o uso do aplicativo é preciso adicionar um novo item no arquivo *Info.plist*. O arquivo *Info.plist* é do tipo *property list* e possui metadados específicos para o aplicativo em desenvolvimento. Esses metadados são usados em diversos momentos, como exibir mensagens em caixas de diálogo solicitando permissões ao usuário. De forma mais concreta, o *Info.plist* é construído a partir de chaves indicando o metadado e valores que descrevem os comportamentos e opções de configuração que se deseja adicionar ao aplicativo. Ao criar um projeto no *XCode* um arquivo *Info.plist* é criado com um conjunto inicial de chaves e valores padrão apropriados. A Figura 8 apresenta um arquivo *Info.plist* padrão gerado pelo *XCode*. Esse é o formato de visualização de um arquivo do tipo *property list*.

Figura 8 – Arquivo Info.plist padrão gerado pro um projeto iOS no *XCode*.

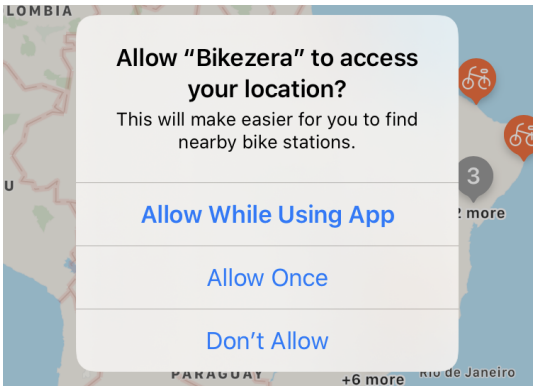
Key	Type	Value
▼ Information Property List	Dictionary	(14 items)
Localization native development region	String	en
Executable file	String	$(EXECUTABLE_NAME)
Bundle identifier	String	$(PRODUCT_BUNDLE_IDENTIFIER)
InfoDictionary version	String	6.0
Bundle name	String	$(PRODUCT_NAME)
Bundle OS Type code	String	APPL
Bundle versions string, short	String	1.0
Bundle version	String	1
Application requires iPhone environment	Boolean	YES
Launch screen interface file base name	String	LaunchScreen
Main storyboard file base name	String	Main
▶ Required device capabilities	Array	(1 item)
▶ Supported interface orientations	Array	(3 items)
▶ Supported interface orientations (iPad)	Array	(4 items)

Fonte: Medium (2019).

A chave `NSLocationWhenInUseUsageDescription` foi acrescentada ao arquivo

Info.plist do projeto *Bikezera* para indicar que o aplicativo deseja acessar informações de localização quando estiver sendo executado em primeiro plano. Junto à chave foi acrescentada uma mensagem explicando o motivo pelo qual o aplicativo está solicitando o acesso à localização do usuário. Essa mensagem será apresentada ao usuário em forma de caixa de diálogo no momento da solicitação da autorização, como é possível ver na Figura 9. A *Apple* exige que a mensagem fornecida seja suficientemente clara para o usuário. É importante atentar-se à isso ou o aplicativo pode ser rejeitado no momento da publicação na *App Store*. Na versão 1.0, quando o aplicativo ainda se chamava *BikeStyle*, ocorreram alguns problemas com relação à isso que serão melhor explicados no seção 6 deste trabalho.

Figura 9 – Caixa de diálogo de acesso à localização.



Fonte: Elaborado pela autora (2019).

Caso o usuário não permita o acesso à sua localização, sempre que o aplicativo for aberto novamente a caixa de diálogo será apresentada, possibilitando que o usuário permita o acesso à sua localização, caso queira. Também é possível permitir o acesso à localização a partir da tela de configurações do aplicativo.

Para ter acesso de fato à localização do usuário, é preciso instanciar um objeto da classe `CLLocationManager` e da classe `CLLocation`. A `CLLocationManager` fornece serviços que determinam a localização geográfica, altitude e orientação de um dispositivo ou sua posição em relação a um dispositivo *iBeacon* nas proximidades. A classe reúne dados usando todos os componentes disponíveis no dispositivo, incluindo *Wi-Fi*, *GPS*, *Bluetooth*, magnetômetro, barômetro e capacidade de processamento (APPLE, 2019a).

A função `getUserLocation`, no Código-fonte 3, é chamada sempre que o aplicativo é aberto. Ela verifica o status atual da autorização do usuário com relação à sua localização e define se será necessário exibir a caixa de diálogo com a solicitação de permissão ou não.

Na linha 13 do Código-fonte 3 o *delegate* do objeto `locationManager`, instância de `CLLocationManager`, é atribuído à instância atual da classe `MKManager` (`self`). A partir disso é possível ter acesso aos métodos que o *delegate* irá invocar durante a execução do aplicativo. Um dos métodos mais relevantes é o `locationManager(_ manager:CLLocationManager, didUpdateLocations locations:[CLLocation])`, que disponibiliza um vetor de objetos `CLLocation` que contém os dados de localização. Esse vetor sempre contém pelo menos um objeto que representa a localização atual do usuário. Os objetos no vetor são organizados na ordem em que ocorreram, portanto, a atualização mais recente da localização é o último elemento do vetor.

Sempre que a localização do usuário for atualizada pelo `locationManager` o método descrito no Código-fonte 4 será executado e atualizará a variável `userLastLocation`.

Código-fonte 3 – Função `getUserLocation`.

```
1  func getUserLocation(_ map: MKMapView) {
2      let status = CLLocationManager.authorizationStatus()
3      switch status {
4      case .notDetermined:
5          locationManager.requestWhenInUseAuthorization()
6          break
7      case .denied, .restricted:
8          delegate?.locationPermissionDenied()
9          return
10     case .authorizedAlways, .authorizedWhenInUse:
11         break
12     }
13     locationManager.delegate = self
14 }
```

Código-fonte 4 – Implementação de `CLLocationManagerDelegate`.

```
1  extension MKManager : CLLocationManagerDelegate {
2      func locationManager(_ manager: CLLocationManager, didUpdateLocations locations: [
       CLLocation]) {
3          userLastLocation = locations.last
4      }
5  }
```

5.3.2 Renderização do mapa

Para embutir um mapa no aplicativo é necessário adicionar uma *view* do tipo `MKMapView` na cena do *storyboard*. Na `UIViewController` responsável pela cena é necessário

conter `import MapKit` no inicio do documento. Na configuração inicial do mapa é indicado seu centro, com as coordenadas de latitude e longitude, e um *span* que define o nível de *zoom* atual na visualização do mapa.

Para encapsular o gerenciamento da localização do usuário, inserção e remoção de `MKAnnotationViews` e área visível do mapa foi criada uma classe `MKManager`. A classe `MKManager` possui todos os métodos necessários para o gerenciamento do mapa.

O *Bikezera* possui dois contextos de mapa: o primeiro é o contexto das cidades e o segundo é o contexto das estações. A `MKMapView` que renderiza essas informações é a mesma, o que mudam são as `MKAnnotationViews` - exibida na Figura 7 - que são apresentadas em cada contexto.

Ao abrir o aplicativo uma primeira requisição é feita à API *CityBikes* e é retornado todas as cidades que possuem SCB. Esse vetor contendo informações das cidades é usado para criar as `MKAnnotationViews` que estarão no mapa.

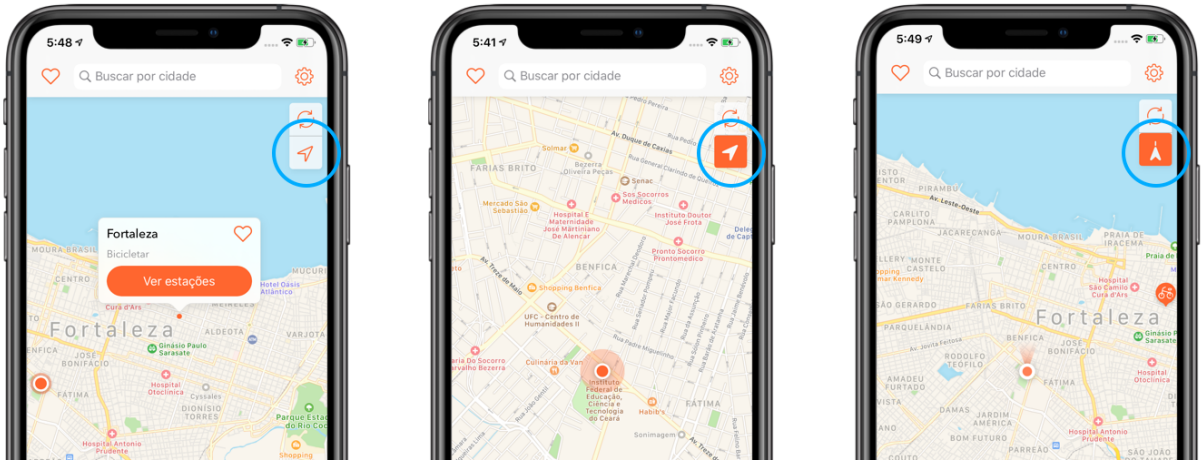
Para posicionar cada `MKAnnotationView` em seu lugar e contendo as informações sobre a cidade é necessário implementar um método provido pelo *delegate* da `MKMapView`. Esse método é o `mapView(_ mapView:MKMapView, viewFor annotation:MKAnnotation) -> MKAnnotationView?`. O método é chamado cada vez que uma `MKAnnotation` é adicionada ao mapa, ou seja, uma `MKAnnotation` para cada elemento no vetor de cidades trazido pela API. Quando o contexto é mudado para as estações esse método é chamado inúmeras vezes novamente. Porém, desta vez iterando o vetor com as informações das estações.

Como mencionado anteriormente, é feita uma requisição à API *CityBikes* para obter as informações de todas as cidades disponíveis, porém, a API ainda não retorna as informações de suas estações - como é possível perceber no Código-fonte 1. O usuário precisa escolher uma cidade específica para ver suas estações. A forma de escolher uma cidade é clicando numa das `MKAnnotationViews` do a mapa, esta então exibe um componente chamado *callout view* que apresenta uma XIB customizada. Toda `MKAnnotationView` possui uma *callout view* associada. No primeiro screenshot da Figura 10 é possível visualizar a *callout view* da cidade de Fortaleza ativa.

Outro elemento renderizado no mapa é um botão especializado que permite ao usuário alternar entre os modos de rastreamento disponíveis. Ao clicar no botão o mapa é recentralizado para exibir a localização atual do usuário, caso o clique novamente é ativado o modo bússola. Esse componente pode ser utilizado a partir da classe `MKUSerlocation`. É uma

forma de manter consistência nas experiências que envolvem o uso de mapas no *iOS*. Na Figura 10 os diferentes modo de rastreamento estão circulados na cor azul.

Figura 10 – `MKUSerlocation` e *Callout View*.



Fonte: Elaborado pela autora (2019).

5.3.3 Agrupamento de marcadores

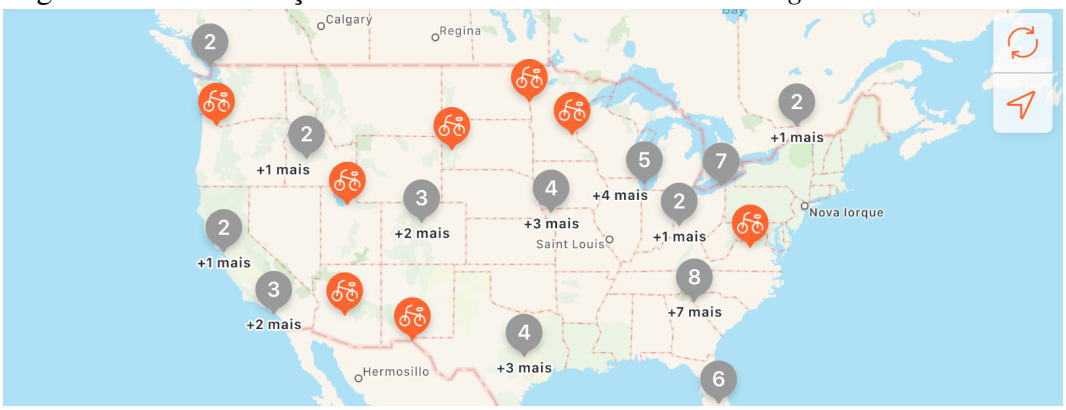
Na *Apple Worldwide Developers Conference* (WWDC) de 2017 a *Apple* lançou novas ferramentas para o `MapKit` que funcionam em dispositivos com *iOS* 11 ou superiores. O *Bikezera* se utilizou de duas dessas novidades: `MKMarkerAnnotationView` e *Clustering*. A `MKMarkerAnnotationView` é um novo tipo customizado de `MKAnnotationView`, com estética mais elegante e animações mais fluidas. A funcionalidade de *Clustering* possibilita que as `MKMarkerAnnotationViews` sejam agrupadas em *clusters* quando estiverem em uma visualização muito distante. Na Figura 11 é possível observar as `MKMarkerAnnotationViews` que representam uma cidade em cor laranja no mapa, enquanto as cidades muito próximas são representados por uma `MKMarkerAnnotationView` em cinza que contém o número de cidades agrupadas.

Para utilizar o sistema de *clustering* do `MapKit` é necessário primeiro configurar a prioridade de renderização da `MKAnnotationView`. Depois é preciso registrar na `MKMapView` uma classe customizada para o *cluster* - essa classe deve herdar de `MKAnnotationView`, associando-a ao identificador padrão *MKMapViewDefaultClusterAnnotationViewReuseIdentifier* - exemplificado no Código-fonte 5.

Além disso, deve-se criar uma classe que herde de `MKMarkerAnnotationView` para o marcador que deseja ser agrupado, no caso atual refere-se ao marcador laranja. É preciso

definir sua cor, seu ícone e informar qual será o `clusteringIdentifier` daquele grupo de marcadores customizados. Isso pode ser observado no Código-fonte 6, nas linhas 5,6 e 7 respectivamente.

Figura 11 – Visualização de *Marker Annotations* e *Clustering*.



Fonte: Elaborado pela autora (2019).

Código-fonte 5 – Configurando um mapa com *Clustering*.

```
1 annotationView.displayPriority = .required
2 mapView.register(CustomMarkerClusterView.self, forAnnotationViewWithReuseIdentifier:
      MKMapViewDefaultClusterAnnotationViewReuseIdentifier)
```

Código-fonte 6 – Criando um `MKMarkerAnnotationView` customizado.

```
1     internal final class CustomMarkerView: MKMarkerAnnotationView {
2     ...
3     func configure(with annotation: MKAnnotation) {
4         if annotation is CityAnnotation {
5             markerTintColor = UIColor.tintColor
6             glyphImage = #imageLiteral(resourceName: "bike")
7             clusteringIdentifier = String(describing: CustomMarkerView.self)
8         }
9         ...
10    }
```

5.4 Requisições *Web*

Para consumir as informações da API *CityBikes* é necessário realizar requisições *web* HTTP a partir do aplicativo. Existem várias opções para realizar esta tarefa, uma delas é utilizar bibliotecas externas de terceiros. Porém, no *Bikezera* foi escolhida um forma nativa de performar as requisições, utilizando a classe `URLSession`.

A classe `URLSession` e as classes relacionadas disponibilizam métodos e objetos que podem ser usados para baixar e fazer *upload* de dados em *endpoints* na *web*, indicados por URLs - inclusive enquanto o aplicativo estiver sendo executado em segundo plano.

Esta seção explica sobre a implementação da camada de requisições *web* no aplicativo *Bikezera* utilizando `URLSession`. Explica também como foram extraídas as informações vindas da API *CityBikes* (exemplificada no Código-fonte 2), além de descrever como foi gerenciado o retorno de chamadas assíncronas.

5.4.1 URLSession

A classe `URLSession` é capaz de criar uma ou mais sessões, onde cada uma delas coordena um grupo de *tasks* relacionadas à transferência dos dados.

As *tasks* de uma sessão compartilham um mesmo objeto de configuração. Ele define parâmetros da conexão, como o número máximo de conexões simultâneas feitas a um *host*. Existem três tipos de configurações possíveis para uma `URLSession`: sessão padrão; sessão efêmera e sessão em segundo plano. Porém, também é possível usar um objeto *singleton* chamado `shared` para requisições básicas. O objeto `shared` não é tão personalizável quanto as sessões customizadas, mas funciona para projetos que tenham requisitos simples. No aplicativo *Bikezera* esse objeto foi utilizado, pois as requisições feitas não precisaram de nenhuma configuração mais avançada.

Depois de instanciar um objeto do tipo `URLSession` com a configuração desejada ou usar o objeto `shared`, é preciso criar uma *task*. Existem três diferentes tipos de *tasks*: *data tasks*; *upload tasks* e *download tasks*.

As *tasks* de *upload* e *download* permitem o gerencimento de arquivos mais robustos navegando entre o aplicativo e a *web*, contudo esses tipos de *tasks* não foram usadas neste projeto. Foi utilizada somente a *data task*. Esta envia e recebe dados usando objetos do tipo `NSData`. A *data task* destina-se à requisições curtas, geralmente repetitivas, para um servidor. Este tipo de *task* é ideal para obter um objeto JSON, por exemplo.

No aplicativo *Bikezera* foi criada uma classe especializada para lidar com as requisições *web*, chamada `APIManager`. A classe possui métodos que gerenciam as requisições *web*, encapsulando essa responsabilidade. As `UIViewControllers` interessadas em obter as informações vindas da API possuem uma instância de `APIManager` para invocar seus métodos.

O Código-fonte 7 exemplifica um dos métodos que compõem a classe `APIManager`.

O método `getAllNetworks` primeiro cria um objeto URL com o endereço completo do *end-point* que será requisitado, depois se utiliza da classe `URLSession` e do método `dataTask` para executar a requisição. O método `dataTask` provê como retorno o JSON indicado no Código-fonte 1, em formato `NSData`. Esse objeto é convertido em uma *struct* personalizada, depois retornado em uma *completion* de forma assíncrona.

Código-fonte 7 – Função para requisição de todas as cidades.

```
1  func getAllNetworks(completion: @escaping (Error?, Networks?) -> Void) {
2      guard let cityBikesUrl = URL(string: "https://api.citybik.es/v2/networks") else {
       return }
3      URLSession.shared.dataTask(with: cityBikesUrl) { (data, response
4          , error) in
5          guard let data = data else { return }
6          do {
7              let decoder = \gls{JSON}Decoder()
8              let networks = try decoder.decode(Networks.self, from: data)
9              completion(nil, networks)
10         } catch let err {
11             print(err)
12             completion(err,nil) }
13         }.resume()
14 }
```

5.4.2 Decodificação de estruturas JSON

No desenvolvimento de aplicativos móveis é muito comum enviar dados por uma conexão de rede, salvar dados localmente, ou enviar dados para APIs e serviços. Para que seja possível realizar essas tarefas geralmente é necessário que os dados sejam codificados e decodificados para um formato intermediário enquanto estão sendo transferidos.

A partir do *Swift* 4 é possível utilizar uma abordagem padronizada para codificação e decodificação de dados. Para adotar essa abordagem é necessário implementar os protocolos `Encodable` e `Decodable` nas classes e *structs* personalizadas. Ao fazer isso, as implementações padrões dos protocolos poderão codificar ou decodificar os dados de formatos como JSON ou *property list*, por exemplo. A forma de uma classe ou *struct* ter suporte tanto ao `Encodable` como `Decodable` é declarar conformidade ao `Codable`.

Apenas indicando a conformidade ao `Codable` a *struct* já está apta a ser codificada e decodificada de forma padrão, sem necessidade de implementação de nenhum método. Porém, isso ocorre somente se seus atributos forem de tipos que originalmente já são `Codable`, como: `String`, `Int`, `Double`, `Date`, `Data`, `URL`, `Array`, `Dictionary`, `Optional`, dentre outros.

Para o *Bikezera* foram criadas diversas *structs* que representam os objetos recuperados da API *CityBikes* em formato JSON. No Código-fonte 8 está a *struct* `BikeNetwork` que é usada para armazenar as informações de cada SCB.

Foi necessário sobrescrever a implementação do método `init(from decoder: Decoder)` pois a API *CityBikes* retorna os dados da cidade encapsulados dentro de uma chave `network`. Portanto, para obter as informações desejadas era necessário acessar o `nestedContainer` do *container* fornecido pelo `Decoder` - isso pode ser observado nas linhas 15 e 16 do Código-fonte 8.

Código-fonte 8 – *Struct* para armazenar dados de uma cidade.

```
1  struct BikeNetwork : Decodable {
2      var id: String?
3      var name: String?
4      var company: Company?
5      var location: BikeNetworkLocation?
6      var stations: [BikeStation]?
7
8      enum CodingKeys: String, CodingKey {
9          case network
10     }
11     enum BikeNetworkDataKeys: String, CodingKey {
12         case id, name, company, location, stations
13     }
14     init(from decoder: Decoder) throws {
15         let container = try decoder.container(keyedBy: CodingKeys.self)
16         let nestedContainer = try container.nestedContainer(keyedBy: BikeNetworkDataKeys.
       self, forKey: .network)
17         self.id = try nestedContainer.decode(String.self, forKey: .id)
18         ...
19     }
20 }
```

5.4.3 Chamadas assíncronas

Como a maioria dos *frameworks* para requisições *web*, a `URLSession` é altamente assíncrona. Ela retorna dados chamando uma *completion* - em outras linguagens de programação é conhecida como função *callback* - do tipo *closure* quando uma transferência é concluída com êxito ou com um erro.

Closure é uma função anônima que pode capturar valores do contexto ao seu redor. Ela pode ser salva em variáveis e até mesmo passada como um parâmetro para uma outra função. Em outras linguagens de programação as *closures* são conhecidas como funções *lambdas*.

No Código-fonte 9 as linhas 4 a 12 somente são executadas quando o resultado da chamada assíncrona é finalizado.

Código-fonte 9 – Requisição de todas as estações.

```
1  func showAllStations(from networkName: String) {
2      ...
3      APIManager.shared.getBikeNetwork(of: networkName, completion: { [weak self] (error,
       bikeNetwork) in
4          if let network = bikeNetwork, stations = network.stations {
5              self?.actualNetwork = network
6              DispatchQueue.main.async {
7                  ...
8                  self?.createStationPoints(stations, in: self?.worldMap)
9                  ...
10             }
11             ...
12         }
13     })
14 }
```

Sempre que deseja-se acessar um atributo de classe dentro de uma *closure* é obrigatório o uso do `self`. Porém, *closures* podem ocasionar *memory leaks* do tipo referência cíclica, já que por padrão elas têm uma referência `strong` das classes nas quais são chamadas. Nesse caso, o conteúdo permanece na memória mesmo ao deixar de ser utilizado pelo aplicativo.

No Código-fonte 9 é usada a lista de captura `[weak self]` na *closure* enviada como parâmetro da função `getBikeNetwork`. Isso foi feito para evitar que ocorra um *retain cycle* na aplicação. Usando o `[weak self]` a *closure* deixa de possuir uma referência `strong` para o objeto `self` e este é tratado como um tipo opcional - por isso se faz necessário usar o `?` para acessar os atributos e métodos da classe dentro da *closure*.

Um outro detalhe importante no Código-fonte 9 está na linha 6, com o uso do `DispatchQueue.main`. O *DispatchQueue* é utilizado quando deseja-se gerenciar em qual *thread* o trecho de código deve ser executado.

Um aplicativo *iOS* é executado em várias *threads*, onde a *Main Thread* é o ponto inicial inicial de execução. Ela executa um *loop* a cada frame que redesenha a tela atual, caso seja necessário; lida com as interações na interface; e executa o conteúdo do `DispatchQueue.main`. Repetindo esse processo até que o aplicativo seja encerrado.

A *Main Thread* possui uma prioridade extremamente alta, onde tudo nela é executado quase que imediatamente. Portanto, é necessário encaminhar códigos que lidam com atualizações de interface para a *Main Thread*, pois caso sejam executados em uma *Background Thread* as

atualizações da interface sofrerão *delays*, ocasionando uma péssima experiência para os usuários.

Requisições *web* normalmente são *tasks* demoradas, por isso são executadas em *Background Thread*. Por esse motivo foi necessário utilizar o `DispatchQueue.main` para executar o método `createStationPoints` e a interface se atualizar rapidamente.

5.5 Internacionalização

A *Apple* é uma empresa presente mundialmente, logo seus produtos também possuem um caráter global. Luca Maestri, chefe financeiro da *Apple*, declarou em 2019 que 900 milhões de *iPhones* estavam em uso em todo o mundo. A *App Store* disponibiliza aplicativos para 155 países ou regiões, em mais de 40 idiomas. Portanto, quando cria-se um aplicativo para a plataforma *iOS* é preciso ter em mente que é um mercado mundial.

A empresa incentiva que os desenvolvedores "Criem aplicativos para o mundo". Para isso oferece não apenas ferramentas de internacionalização e tradução dos aplicativos em si, mas também possui diversas palestras sobre o assunto na sua conferência anual para desenvolvedores - a WWDC. Uma lista extensa de vídeos sobre o assunto pode ser encontrada no endereço: <https://developer.apple.com/videos/frameworks/internationalization-and-localization>.

Mais do que somente traduzir o aplicativo, internacionalizá-lo significa adaptar seu conteúdo para diferentes culturas. Isso pode envolver diferentes orientações da interface, o linguajar dos textos no aplicativo ou o significado das cores utilizadas. Estas são algumas das recomendações dadas pela *Apple* para preparar um aplicativo a fim de atingir um público global:

Estruturar o aplicativo corretamente Deve-se oferecer suporte a conteúdo localizado, criando textos e imagens de forma isolada ao código em execução. Utilizar os frameworks nativos para formatar valores automaticamente, como: datas, pesos, preços e símbolos monetários.

Traduzir o conteúdo É possível criar arquivos contendo o conteúdo traduzido para diversos idiomas diferentes. O *XCode* possui suporte para a tradução de cada arquivo XIB ou *storyboard*.

Pensar na culturalização Uma boa internacionalização reflete uma compreensão dos costumes e tendências locais. É indicado incorporar conteúdo específico da cultura local no aplicativo.

Localização na *App Store* Na *App Store* é possível alterar o nome do aplicativo e outros metadados para cada região (APPLE, 2019b).

Afim de seguir as orientações da *Apple* e aumentar a visibilidade, o aplicativo

Bikezera foi traduzido. Também foi levado em consideração que o aplicativo pode ser usado mundialmente, já que contém informações de SCB de mais de 400 cidades. Logo, é importante que seja traduzido em diversos idiomas.

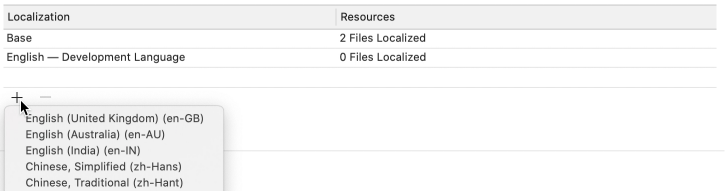
5.5.1 Traduzindo o aplicativo

A versão 1.0 do aplicativo *Bikezera* possuía uma versão apenas em Inglês. Como uma das modificações para este trabalho, na versão 1.1, ele foi trazido também para Português. Na *App Store* também foram adicionadas descrições e palavras chave para os dois idiomas.

Nesta seção é descrito o processo para a tradução de um aplicativo *iOS* em mais de uma língua. É importante ressaltar que os passos seguidos podem variar dependendo da versão do *XCode* instalado na máquina. Para este projeto foi utilizado o *XCode* 11.0.

Para ativar a localização em um aplicativo usando *XCode* 11.0 é necessário primeiro clicar no nome do projeto que fica no topo na área de navegação do lado esquerdo da interface. Na área de edição, deve-se clicar na opção *Info* e adicionar um novo idioma - como demonstrado na Figura 12.

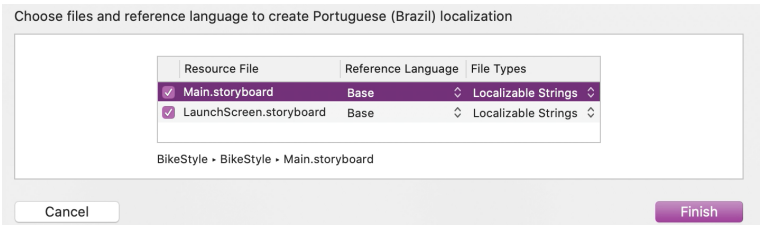
Figura 12 – Adicionando um novo idioma ao projeto.



Fonte: Elaborado pela autora (2019)

Ao selecionar um idioma, o *XCode* apresenta a caixa de diálogo da Figura 13, onde deve-se indicar quais os arquivos de *storyboards* desejam ser traduzidos. Ao fazer isso o aplicativo está apto a ser traduzido para cada um dos idiomas adicionados.

Figura 13 – Caixa de diálogo para escolha de storyboards traduzidas.



Fonte: Elaborado pela autora (2019)

O *XCode* automaticamente cria um arquivo no formato *strings* contendo todos os

textos atualmente escritos nas *storyboards*, com chaves únicas. Os textos que devem ser alterados para novo idioma são os valores do lado direito. No código-fonte 10 é possível observar o arquivo `Main.strings` localizado em Português. Nesse momento as *storyboards* do projeto já estão localizadas, porém para localizar os arquivos XIB é necessário seguir alguns passos a mais.

Código-fonte 10 – Arquivo `Main.strings` localizado em Português.

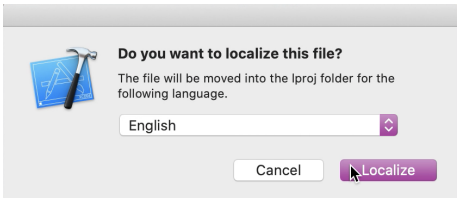
```
1 ...
2 "1d4-0q-I8g.text" = "bicicletas livres";
3 "Cwf-IS-sPr.text" = "Este aplicativo usa CityBikes API. Clique para saber mais sobre esse
      projeto.";
4 "OL3-Dy-gBk.headerTitle" = "Mais";
5 "sRd-gJ-vr6.normalTitle" = "Voltar para cidades";
6 ...
```

Primeiro seleciona-se o arquivo XIB na área de navegação. Na área de utilidades aparece a seção *Localization*, como na Figura 14. Ao clicar no botão "*Localize...*"a caixa de diálogo representada na Figura 15 é exibida, onde deve-se selecionar o idioma no qual deseja traduzir o arquivo XIB. Com isso o *XCode* criará arquivos do tipo *string* para cada um dos idiomas. Ao final, a estrutura do projeto é semelhante à da Figura 16. Esse processo foi repetido para cada uma das XIBs do projeto.

Figura 14 – Habilitar localização no arquivo.

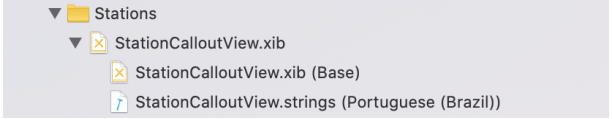
Fonte: Elaborado pela autora (2019)

Figura 15 – Opções de localização no arquivo.



Fonte: Elaborado pela autora (2019)

Figura 16 – Estrutura final de um XIB localizado.



Fonte: Elaborado pela autora (2019)

Incluir os arquivos do tipo *string* no projeto com as devidas traduções, permite que o usuário final visualize o aplicativo com o idioma padrão selecionado no seu sistema operacional.

Na versão 1.1 o *Bikezera* foi traduzido apenas o Português, por questões de escopo. Porém, para adicionar mais idiomas ao aplicativo basta seguir os passos apresentados nessa seção novamente para cada idioma.

6 PROCESSO DE PUBLICAÇÃO NA *APP STORE*

O processo de publicação de um aplicativo *iOS* na *App Store* possui muitos pequenos detalhes, não é a toa que a maioria do desenvolvedores não gosta desse momento. Existem ferramentas que facilitam esse processo, como o *Fastlane*, porém para o presente trabalho o processo foi feito de forma manual. Os passos necessários para a publicação são descritos a seguir. Vale ressaltar que esses passos podem variar dependendo da versão do *XCode* ou de mudanças no portal *Apple Developer* e *App Store Connect*.

Primeiramente, é preciso possuir uma conta de desenvolvedor da *Apple*. Caso não possua, deve-se criar uma nova conta através do endereço <http://developer.apple.com/account/>. Para ter o direito publicar aplicativos na *App Store* é necessário pagar uma taxa anual no valor de 100 dólares. Depois do pagamento é possível publicar aplicativos com a conta de desenvolvedor.

Deve-se entrar no portal do desenvolvedor e adicionar um novo *App Id* vinculado à conta. Esse *App Id* possui o mesmo *bundle identifier* que o projeto do *XCode* que se deseja publicar. Caso a conta ainda não possua um certificado de distribuição válido ou não possua *provisioning profiles*, estes devem ser criados no mesmo portal.

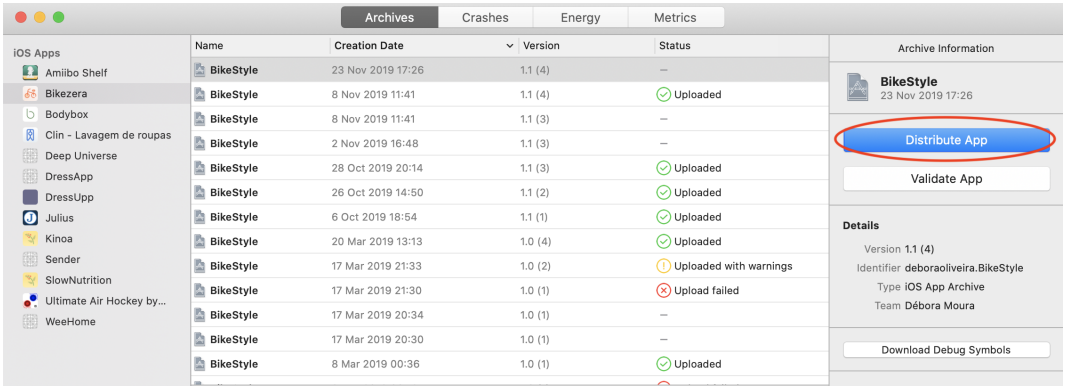
Também é necessário criar um novo registro de aplicativo no portal *App Store Connect* - pode ser acessado pelo endereço <https://appstoreconnect.apple.com>. No novo registro de aplicativo serão acrescentadas informações como nome, idiomas disponíveis e o identificador do pacote - que é o identificador do *App Id* criado no passo anterior. Também é obrigatório preencher metadados como nome, descrição, *screenshots* do aplicativo, palavras chave, endereço para acesso à política de privacidade, entre outros.

Após as configurações no portal *Apple Developer* e *App Store Connect* é preciso voltar para o projeto no *XCode*. Na aba "*Signing & Capabilities*"deve-se conferir se o time de desenvolvimento está selecionado corretamente, dessa forma os certificados de distribuição e *provisioning profiles* automaticamente serão preenchidos.

Após isso, ainda no *XCode*, deve-se acessar *Product* > *Archive*, isso iniciará o processo de *build* do aplicativo. Ao finalizar o *build*, deve-se clicar em "*Distribute App*", um botão do lado direito da janela - representada na Figura 17. Caso não seja necessário alguma configuração avançada pode-se clicar em continuar até o fim do processo - assim foi feito no *Bikezera*.

Quando o *upload* terminar, a compilação ficará disponível no portal *App Store Connect* para iniciar testes internos ou publicar na *App Store*.

Figura 17 – Caixa de diálogo com opções de exportação.



Fonte: Elaborado pela autora (2019)

Ao escolher publicar na *App Store*, o aplicativo entra em um processo de revisão pela equipe da *Apple*. Existe uma extensa lista de orientações para publicação de aplicativos na *App Store*, chamada *App Store Review Guidelines*. Caso o aplicativo fira alguma das *guidelines* ele será rejeitado e a *Apple* solicitará modificações. Depois de alterar o que for necessário, deve-se repetir todos os passos novamente. As orientações são constantemente atualizadas e podem ser acessadas pelo endereço <https://developer.apple.com/app-store/review/guidelines/>.

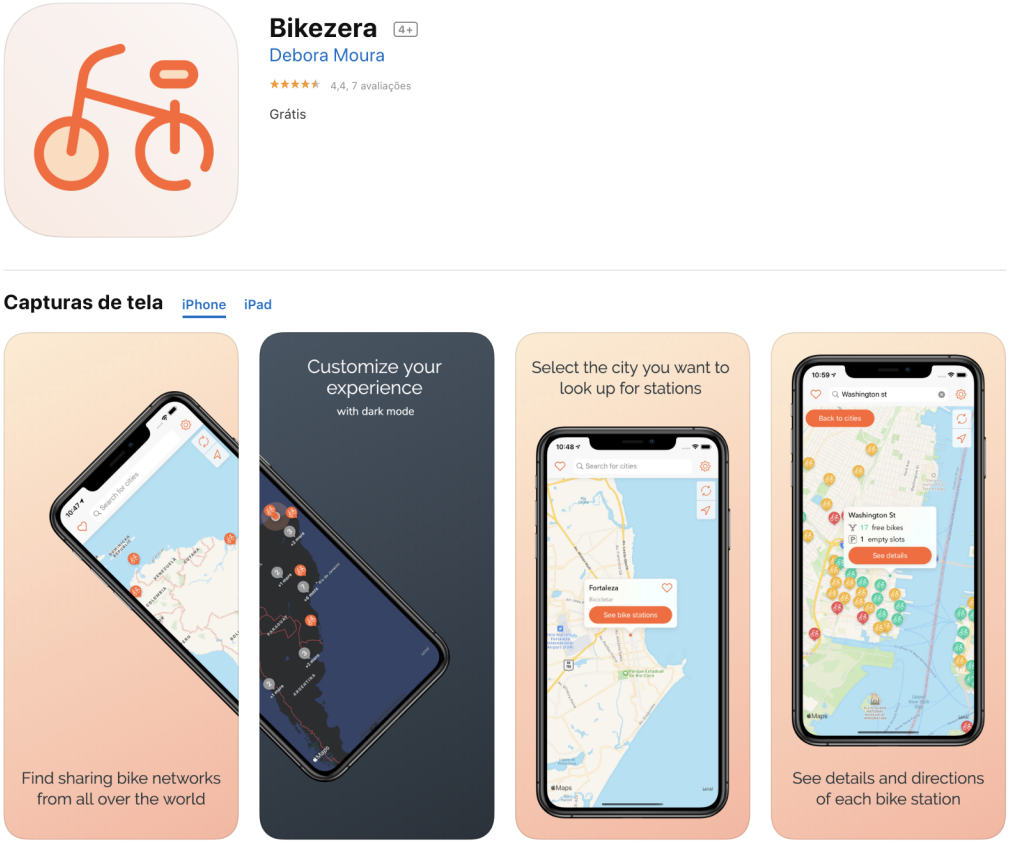
A versão 1.0 do *Bikezera*, ainda com nome *BikeStyle*, foi rejeitada. O motivo foi que o texto de solicitação para acesso à localização do usuário não estava claro o suficiente, então foi preciso reescrever esse texto de forma que informasse ao usuário o motivo pelo qual era relevante que o *Bikezera* tivesse acesso à sua localização atual. O problema foi corrigido e alguns dias depois a primeira versão do aplicativo foi aceita e publicada na loja.

A versão 1.1 do *Bikezera*, com as alterações propostas neste trabalho, não sofreu rejeições e foi aceita para a *App Store* na primeira tentativa. O processo para a revisão e aprovação do aplicativo durou cerca de 3 dias.

O aplicativo *Bikezera* pode ser encontrado na *App Store* simplesmente buscando por "*Bikezera*"na barra de pesquisa ou pelo endereço <https://apps.apple.com/br/app/bikezera/id1455587148>. As palavras chave e a descrição do aplicativo estão localizadas em Português e Inglês.

A Figura 18 apresenta um *screenshot* com o aplicativo na *App Store*.

Figura 18 – *Print* do produto final publicado na *AppStore*.



Fonte: Elaborado pela autora (2019)

7 *BIKEZERA*

Nesta seção é apresentado o resultado final do produto desenvolvido neste trabalho, com *screenshots* das telas e descrição de como performar suas funcionalidades. O produto pode ser obtido de forma gratuita na *App Store*.

O aplicativo final possui todas as telas prototipadas na seção 4.4. Além delas, foi acrescentada uma tela com uma mensagem indicando quando a internet estiver indisponível. Os *screenshots* das principais telas do *Bikezera* podem ser visualizados na figura 19.

Na versão 1.1 do aplicativo *Bikezera*, relatada neste presente trabalho, foram consertados problemas, modificados aspectos e acrescentadas funcionalidades. Abaixo está a lista com todas as modificações:

- A caixa de diálogo solicitando a permissão de localização do usuário é apresentada depois do tutorial inicial;
- Caso o usuário não permita o acesso à sua localização inicialmente, o aplicativo periodicamente sugere que ele o faça;
- O *autolayout* das *views* foram ajustadas para funcionar adequadamente em *iPhone* X ou posteriores;
- O tutorial inicial é chamado como modal na tela de preferências. (tela 1 na figura 19);
- O botão de preferências foi movido para o lado direito da barra de navegação. (tela 2 na figura 19);
- Inserção do botão que alterna entre os modos de rastreamento disponíveis: bússola e centralização da localização;
- Separação de dois pontos com as mesmas coordenadas no mapa. (É possível observar essa mudança na cidade de São Paulo/Brasil);
- Adicionada uma área para envio de comentário dos usuários para a equipe de desenvolvimento;
- Suporte para *Dark Mode* no *iOS* 13 (tela 12 na figura 19);
- Ao clicar em um marcador do tipo *Cluster*, o mapa é ampliado para exibir as cidades agrupadas;
- Pesquisa realizada ignorando os acentos e caracteres especiais;
- Inclusão da funcionalidade de adicionar cidades e estações aos favoritos;
- Inclusão da funcionalidade de visualizar cidades e estações salvas nos favoritos (tela 5 na figura 19);

- Tradução do aplicativo pra Português e Inglês.

Abaixo estão listadas as orientações para executar as funcionalidades especificadas na seção 4.3, no produto final. Todas as telas indicadas estão disponíveis na figura 19.

[F-01] Visualizar cidades

Iniciar o aplicativo e estar com o dispositivo conectado à *internet*. As cidades poderão ser visualizadas na tela principal, apresentada na **Tela 2**.

[F-02] Pesquisar por cidades

Clicar na barra de pesquisa no topo da **Tela 2**. Após isso, digitar o texto a ser pesquisado. Ao digitar o texto, os resultados da busca serão listados como na **Tela 4**.

[F-03] Visualizar estações

Selecionar uma cidade, clicando no seu respectivo marcador no mapa da **Tela 2**. Clicar no botão "Ver estações"na cidade selecionada; representado na **Tela 9**. As estações serão listadas como na **Tela 10**.

[F-04] Pesquisar por estações

Seguir as mesmas ações da **[F-02]**. A diferença é que o mapa deve estar no contexto de estações.

[F-05] Visualizar detalhes das estações

Selecionar uma estação, clicando no seu respectivo marcador no mapa da **Tela 10**. Clicar no botão "Ver detalhes"na estação selecionada; representado na **Tela 11**. Os detalhes da estação aparecerão na **Tela 3**.

[F-06] Traçar rota do local atual para a estação

Visualizar os detalhes de uma estação seguindo os passos de **[F-05]**. Clicar no botão "Ver rotas no Maps". O aplicativo *Maps* será inciado com a rota até a estação.

[F-07] Função bússola

Clicar duas vezes no segundo botão da lista localizada no canto superior direito da tela principal. Pode ser visualizado na **Tela 2**.

[F-08] Visualizar localização atual

Clicar apenas uma vez no botão referenciado na **[F-07]**.

[F-09] Adicionar conteúdo aos favoritos

Para adicionar uma cidade aos favoritos, deve-se selecionar a cidade no mapa e clicar no botão com símbolo de coração no canto superior direito da *callout view* da cidade selecionada. O botão pode ser encontrado na **Tela 9**.

Para adicionar uma estação aos favoritos, deve-se seguir as mesmas ações de **[F-05]**. Clicar no botão com símbolo de coração no canto superior direito da **Tela 3**.

[F-10] Visualizar favoritos

Clicar no botão localizado ao lado esquerdo da barra de navegação na tela principal. Será exibida uma lista com as cidades e estações salvas, a lista pode ser visualizada na **Tela 5**.

[F-11] Recarregar conteúdo

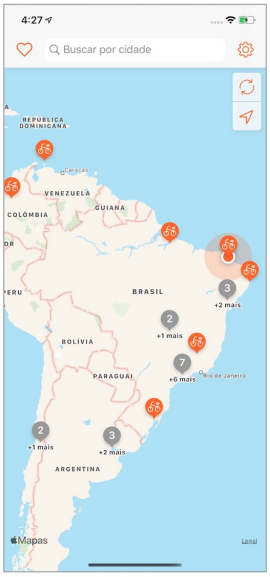
Clicar apenas uma vez no primeiro botão da lista localizada no canto superior direito da tela principal. Pode ser visualizado na **Tela 2**.

[F-12] Enviar comentários para a equipe de desenvolvimento

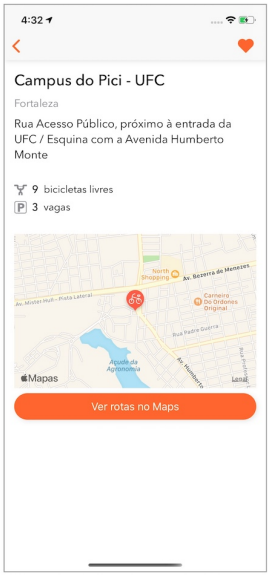
Entrar na tela de configurações, clicando no botão localizado ao lado direito da barra de navegação na tela principal. Na seção "MAIS", clicar em "Enviar comentários".

Figura 19 – Telas do produto finalizado.

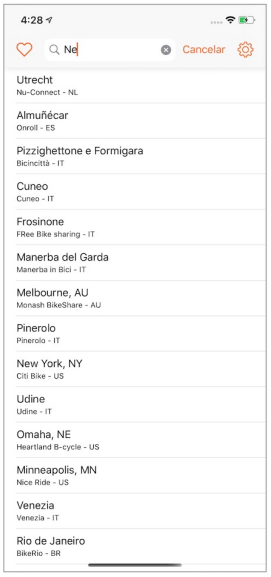
1. Onboarding



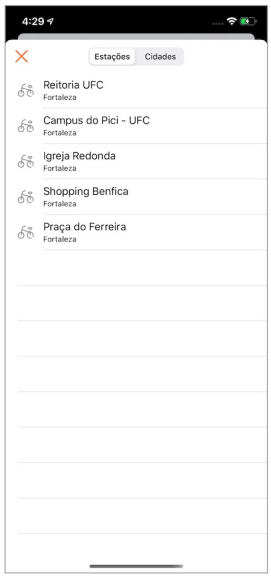
2. Mapa com cidades



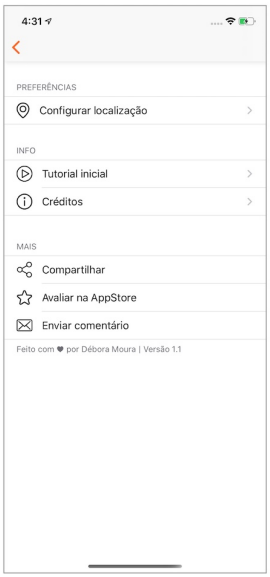
3. Estação



4. Pesquisa



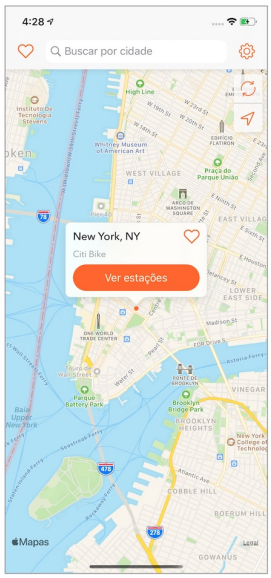
5. Favoritos



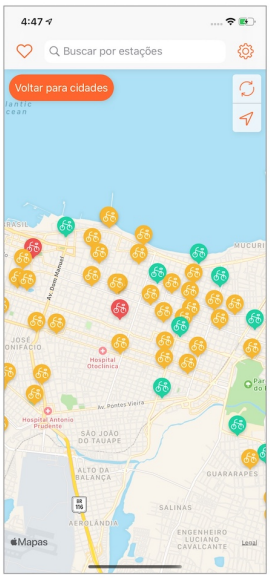
6. Preferências

7. Créditos

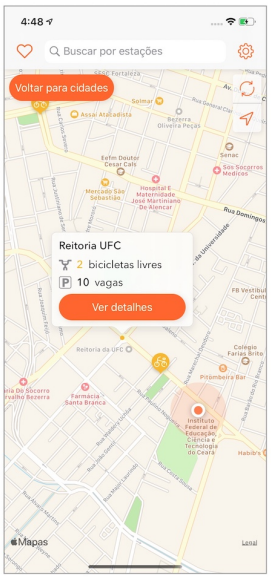
8. Não conectado



9. Cidade selecionada



10. Mapa com estações



11. Estação selecionada

12. Mapa em modo noturno

Fonte: Elaborado pela autora (2019)

8 CONCLUSÕES E TRABALHOS FUTUROS

Com o aumento do fluxo de pessoas nos grandes centros urbanos, iniciou-se uma discussão mais profunda sobre o que caracterizaria um modelo de mobilidade urbana sustentável. Esse modelo incentiva o uso de transportes públicos coletivos para viagens longas, caminhadas para viagens curtas e uso de bicicletas para viagens curtas e medianas.

Com a evolução dos SCB foi possível ter acesso à dados abertos acerca de suas estações tempo real. Foi pensado, então, em criar um aplicativo que se utilizasse desses dados, criando uma interface amigável para o usuário, afim de incentivar o uso de tais bicicletas compartilhadas como meio de transporte sustentável nas cidades.

Este trabalho relatou detalhes da implementação do aplicativo *Bikezera* em suas camadas mais importantes, como a renderização de um mapa *user-friendly* e requisições *web* para obter os dados em tempo real. Relatou-se também a contextualização sobre desenvolvimento *iOS*, protótipos do produto, análise do usuário e a arquitetura de software utilizada. Foram apresentadas as funcionalidades que compuseram a versão 1.0 do aplicativo, postada no início de 2019 na *App Store* e desenvolvida pela própria autora, e as melhorias e funcionalidades acrescentadas para a versão 1.1, postada em outubro de 2019 na *App Store*.

As funcionalidades levantadas para a versão 1.1 foram todas finalizadas, porém durante a escrita deste trabalho foram descobertas novas correções que seriam relevantes para o produto:

- Mudar o contexto do mapa (cidade/estação) a partir do zoom do usuário;
- Clicar no mapa da tela de estação única e expandir sua visualização para a tela toda;
- Indicar para o usuário a cidade mais próxima, perguntando se ele deseja ver as bicicletas dessa cidade.

Além das correções citadas foram levantadas possíveis novas funcionalidades para o produto. Essas funcionalidades demandariam um tempo maior de desenvolvimento e dependeriam de outros serviços. As funcionalidades são:

- Redirecionamento para o respectivo aplicativo do SCB local;
- Integrar com informações de outros meios de transporte, como estações de metrô ou ônibus, próximos da estação de bicicleta selecionada;
- Fornecer o aluguel das bicicletas diretamente pelo *Bikezera*;

O estado atual do aplicativo *Bikezera* encontra-se com as funcionalidades descritas neste trabalho no capítulo 7. Sua aparência mantém-se idêntica aos *screenshots* apresentados na

figura 19.

Como trabalhos futuros pretende-se adicionar as seguintes funcionalidades ao aplicativo (1) adicionar mais idiomas, (2) permitir o compartilhamento de uma estação ou cidade, (3) adicionar um cronômetro para marcar o tempo corrido do aluguel de uma bicicleta, (4) permitir que os usuários realizem avaliações de cidades e estações, como parâmetro de medição de qualidade, (5) realizar as correções citadas nesta atual seção.

REFERÊNCIAS

APPFIGURES. Survey from app store analytics firm appfigures. 2018. Disponível em: <https://blog.appfigures.com/ios-developers-ship-less-apps-for-first-time/>. Acesso em: 27 ago. 2019.

APPLE. Apple developer documentation. 2019. Disponível em: <https://developer.apple.com/documentation/>. Acesso em: 02 nov. 2019.

APPLE. Apple developer internationalization documentation. 2019. Disponível em: <https://developer.apple.com/internationalization/>. Acesso em: 13 nov. 2019.

APPLE. Human interface guidelines. 2019. Disponível em: <https://developer.apple.com/design/human-interface-guidelines/>. Acesso em: 10 out. 2019.

BAUMAN, A. E.; RISSEL, C. Cycling and health: an opportunity for positive change? **Medical journal of Australia**, v. 190, n. 7, p. 347, 2009.

BERGMAN, L.; RABI, N. I. Mobilidade e política urbana: subsídios para uma gestão integrada. **Rio de Janeiro: IBAM**, 2005.

CARTILHA, D. M. U. Ministério das cidades e instituto pólis. brasília, 2005. **Arquivo Digital. Disponível em www. cidades. gov. br. Acessado em abril de 2019**, 2005.

CARVALHO, C. H. R. d. Mobilidade urbana sustentável: conceitos, tendências e reflexões. 2016.

CONTARDO, C.; MORENCY, C.; ROUSSEAU, L.-M. **Balancing a dynamic public bike-sharing system**. [S.l.]: Cirrelt Montreal, Canada, 2012. v. 4.

EAESP, F. Pesquisa anual do uso de ti. 2018. Disponível em: <https://eaesp.fgv.br/sites/eaesp.fgv.br/files/pesti2018gvciappt.pdf>. Acesso em: 27 ago. 2019.

HALL, P. G. Cities in civilization. Pantheon Books New York, 1998.

IBGE. 94, 2% das pessoas que utilizaram a internet o fizeram para trocar mensagens. **PNAD Contínua TIC 2016**, v. 11, 2016. Disponível em: <https://agenciadenoticias.ibge.gov.br/agencia-sala-de-imprensa/2013-agencia-denoticias/releases/20073-pnad-continua-tic-2016-94-2-das-pessoas-que-utilizaram-ainternet-o-fizeram-para-trocar-mensagens Acesso em: 18 ago. 2019.

OJA, P.; TITZE, S.; BAUMAN, A.; GEUS, B. D.; KRENN, P.; REGER-NASH, B.; KOHLBERGER, T. Health benefits of cycling: a systematic review. **Scandinavian journal of medicine & science in sports**, Wiley Online Library, v. 21, n. 4, p. 496–509, 2011.

PEZZUTO, C. C.; SANCHES, S. d. P. Identificação dos fatores que influenciam no uso da bicicleta. In: **XVIII Congresso de Pesquisa e Ensino em Transportes, ANPET. Florianópolis, SC**. [S.l.: s.n.], 2004.

SEMOB, B. Política nacional de mobilidade urbana sustentável. **Princípios e diretrizes aprovadas**, 2004.

SHAHEEN, S. A.; GUZMAN, S.; ZHANG, H. Bikesharing in europe, the americas, and asia: past, present, and future. **Transportation Research Record**, SAGE Publications Sage CA: Los Angeles, CA, v. 2143, n. 1, p. 159–167, 2010.

SHAHEEN, S. A.; MARTIN, E. W.; COHEN, A. P.; CHAN, N. D.; POGODZINSKI, M. Public bikesharing in north america during a period of rapid expansion: Understanding business models, industry trends & user impacts, mti report 12-29. Mineta Transportation Institute, 2014.

SILVEIRA, M. O. da. **Mobilidade Sustentável: A bicicleta como um meio de transporte integrado**. Tese (Doutorado) — Universidade Federal do Rio de Janeiro, 2010.

STATISTA. Unit sales of the apple iphone worldwide from 2007 to 2018 (in millions). 2018. Disponível em: <https://www.statista.com/statistics/276306/global-apple-iphone-sales-since-fiscal-year-2007/>. Acesso em: 03 set. 2019.