SPOTCLAIMER

Aplicação móvel de procura de locais para a prática de desportos de ação

---

DISSERTAÇÃO DE MESTRADO EM DESIGN E MULTIMÉDIA
FACULDADE DE CIÊNCIAS E TECNOLOGIAS

DANIEL RIBEIRO CARDOSO

JULHO DE 2014

UNIVERSIDADE DE COIMBRA

ORIENTADORES:
Licínio Gomes Roque
João Miguel Andrade Proença da Cunha

JÚRI:
António José Olaio Correia de Carvalho
Eduardo Miguel Morgado Nunes

O presente documento foi escrito ao abrigo do
Acordo Ortográfico de 1990.

Aos meus pais e à minha irmã.
Aos meus amigos.

Aos meus pais, um obrigado pelo apoio que sempre demonstraram, não só durante este projeto mas também em todas as etapas anteriores da minha vida.

À minha irmã, um obrigado pelas boleias, jantares e conversas.

Um obrigado aos meus orientadores pela disponibilidade, paciência e pela experiência que partilharam ao decorrer deste projeto.

Um obrigado aos amigos do curso e aos amigos de sempre e um pedido de desculpas pelas ausências, prometo que serão compensadas.

Obrigado aos colegas de Aveiro, pela amizade e pela experiência profissional que me proporcionaram ao longo do último ano, sem vocês esta dissertação não seria possível.

Obrigado à Tânia por todo o apoio incondicional e por acreditar sempre em mim.

Um sincero obrigado,
Daniel Cardoso

Coimbra, 2 de Julho de 2014

RESUMO

O número de atletas praticantes de desportos de ação tem aumentado significativamente nos últimos anos, no entanto, os espaços dedicados à prática destes desportos, quer sejam de origem natural ou humana, são relativamente pouco usados e não existe dinamização a nível local para promover a prática destes desportos.

Esta dissertação é uma reflexão sobre o processo de desenvolvimento de uma aplicação que visa aumentar e incentivar a competitividade nos desportos de ação e dinamizar os locais propícios à prática dos mesmos, promovendo também as relações entre atletas. A aplicação possibilitará a criação de informação relacionada com a distribuição geográfica destes locais bem como as suas caraterísticas e permitirá aos atletas criar e partilhar conteúdo multimédia associado a cada local. A aplicação será desenvolvida para dispositivos móveis, o que permitirá o uso de funcionalidades de georreferenciação, e terá na sua base a interação social entre os utilizadores e a partilha de conteúdo multimédia.

Numa primeira fase será feito um levantamento e análise do estado da arte, nomeadamente de aplicações já existentes que tenham propósitos similares. Deste modo, será possível adquirir um melhor conhecimento de todo o trabalho já feito na área.

Após um estudo do trabalho já existente na área serão criados *storyboards* e *mockups* que permitirão, posteriormente, o desenho de uma interface de alta fidelidade e a definição de um modelo de interação com estrutura, navegação e declaração dos objetos de informação, da sua apresentação e das ações possíveis em cada local.

Numa terceira fase será implementado um protótipo funcional da aplicação que sirva como prova de conceito e de intermediário para uma aplicação final.

PALAVRAS CHAVE

Aplicação móvel, Atleta, Desporto de ação, Geolocalização, Local, Social.

ÍNDICE

1. INTRODUÇÃO

Os desportos de ação e as tecnologias móveis são duas áreas distintas em grande crescimento. Os dispositivos móveis acompanham-nos para qualquer lado e começam a ser uma extensão do corpo humano, tal como as pranchas de *surf*, *skate* e *snowboard* são uma extensão do atleta. A criação de eventos como os *X Games* trouxe os desportos de ação para o conhecimento público e tornou-os em atividades que geram bastante investimento e receita. Sendo que este tipo de desportos exigem locais específicos e por vezes difíceis de encontrar, os dispositivos móveis, com as suas funcionalidades de geolocalização, são a ferramenta ideal para encontrar os locais com as melhores condições de acordo com as preferências dos atletas. Esta necessidade de encontrar os locais perfeitos e de perceber como os dinamizar através de uma aplicação móvel é uma das intenções do projeto a desenvolver durante esta dissertação.

Embora a aplicação a desenvolver possa abranger bastantes desportos de ação no futuro, para esta dissertação decidiu-se apenas incluir o *surf*, o *skate* e o *snowboard* de modo a poder otimizar o tempo de desenvolvimento e permitir uma implementação mais cuidada de cada um dos desportos.

A tipologia e as condições do local onde estes desportos são praticados pode fazer a diferença entre uma boa e uma má experiência, sendo os atletas e os iniciantes confrontados diariamente com a escolha de um novo local que lhes permita obter a melhor experiência possível. A aplicação a desenvolver visa tornar esta ação de escolher um novo local o mais simples possível para os praticantes, fornecendo-lhes informação útil e reunida num único sítio, disponível em qualquer lugar.

Embora sejam desportos individuais, é nas interações com outros praticantes que mais se aprende, sendo, nestes desportos, importante potenciar essas relações. Além de fornecer locais, a aplicação tenta também estabelecer estas relações ao criar funcionalidades de competitividade associada a cada um dos locais, levando os atletas a explorarem os mesmos locais e a desafiarem-se mutuamente de modo a ganharem notoriedade no mundo do desporto de ação.

Apesar de ser direcionada para atletas e praticantes, a aplicação a desenvolver também terá interesse para o resto da comunidade que envolve o desporto de ação. Fãs, patrocinadores e marcas poderão usar a aplicação para interagir com os atletas e descobrir novos talentos.

De seguida será abordado o que motivou a escolha deste tema e será analisado o contexto em que este se insere.

MOTIVAÇÃO

O tema desta dissertação surgiu durante trabalho extracurricular que o autor tem vindo a desenvolver direcionado para o desporto de ação. Embora não seja praticante atualmente, a paixão pelos desportos de ação e por tecnologias móveis despertaram o interesse para este projeto.

A ideia de desenvolver uma aplicação móvel surgiu numa das muitas conversas relacionadas com desportos de ação no seio do grupo de trabalho do mestrando mas, apenas mais tarde e após alguma pesquisa e exploração do mercado é que se verificou que realmente existia uma necessidade de desenvolver tal aplicação.

Atualmente, tanto quanto se sabe, não existe nenhuma aplicação que permita juntar atletas de vários desportos numa só plataforma que forneça conteúdo de valor para a prática dos mesmos, permita a partilha de conteúdo multimédia, crie competitividade entre atletas a nível local e que esteja disponível em qualquer lugar. Esta possibilidade de tornar a aplicação num produto disponível no mercado de aplicações foi também um dos principais fatores que motivou a decisão de desenvolver este projeto e, consequentemente, escolher este tema para a dissertação de Mestrado.

Enquanto designer e tendo em conta os interesses do mestrando, achou-se essencial abordar todas as fases de desenvolvimento associadas ao desenvolvimento de uma aplicação para um dispositivo móvel, como o design de interface, a organização de conteúdos e a programação.

ENQUADRAMENTO

Esta dissertação é o produto final de uma Licenciatura e de um Mestrado em Design e Multimédia, durante os quais foram adquiridas competências nas diversas áreas abrangidas pelo plano curricular. Com esta dissertação pretende-se então não só pôr em prática essas competências – mais concretamente as relacionadas com design de interface e com a programação — mas também adquirir novos conhecimentos em novas áreas, neste caso o desenho e desenvolvimento de aplicações móveis.

A falta de uma aplicação móvel que vá de encontro ao que os atletas de desporto de ação necessitam para divulgar os seus feitos e se auto promoverem, que permita a dinamização de um local para a prática, crie competitividade e que seja exclusivamente dedicada a desporto de ação permite efetuar um trabalho que abrange um largo espectro de competências e de fases até atingir um produto final.

ÂMBITO

Esta proposta pretende produzir um conceito validado através de uma prototipagem e sua avaliação de usabilidade que culmine numa aplicação móvel dedicada a atletas de desportos de ação, passível de ser instalada por qualquer pessoa num dos dispositivos móveis atuais, nomeadamente os *smartphones*. Embora o público alvo desta dissertação seja académico, o público alvo da aplicação são os atletas dos desportos de ação, sendo que a aplicação irá abranger uma maior comunidade que inclui as pessoas relacionadas com o meio do desporto de ação como os praticantes, os fãs e as marcas patrocinadoras dos atletas.

Até chegar a uma aplicação final serão realizados estudos que pretendem identificar as relações existentes entre os diferentes utilizadores, de maneira a perceber como estes interagem e quais as suas funções principais no desporto de ação. Serão também estudadas as aplicações móveis já existentes no mercado que estão de certa forma relacionadas com a aplicação a desenvolver. Este estudo permitirá compreender de que forma se irá diferenciar a aplicação a desenvolver das já existentes. Após uma análise do trabalho já existente na área será efetuada uma investigação sobre o impacto das decisões de design e a sua influência no uso das aplicações bem como uma consulta das diretrizes definidas pelos próprios sistemas operativos dos dispositivos móveis. Pretende-se também adquirir um conhecimento sobre o desenvolvimento programático de uma aplicação móvel e as etapas necessárias para a sua criação, desde a prototipagem, passando pelo desenho até à implementação.

A presente dissertação é desenvolvida no âmbito da disciplina Estágio/ Dissertação, a qual está inserida no plano curricular do Mestrado em Design e Multimédia da Faculdade de Ciências e Tecnologias da Universidade de Coimbra.

DECLARAÇÃO DE INVESTIGAÇÃO

Esta dissertação foca-se em analisar e compreender as necessidades que existem por parte dos atletas de desportos de ação, sendo neste caso a investigação feita sobre apenas três desporto: *surf*, *skate* e *snowboard*.

Serão objetos de estudo as culturas, costumes e estilos de vida dos praticantes destes desportos e espera-se, mais tarde, traduzir essas culturas, através da linguagem textual ou gráfica, para uma aplicação móvel que possa criar uma sinergia entre os praticantes desses desportos e ajudar ao crescimento dos mesmos.

Ao nível da aplicação a desenvolver, será realizado um estudo de design e usabilidade das interfaces de aplicações móveis para tentar identificar modelos de interação que satisfaçam os utilizadores. Serão também estudados os conceitos e técnicas de programação orientada para dispositivos móveis que permitirão implementar a aplicação.

CONTRIBUTOS ESPERADOS

Com esta dissertação pretende-se desenvolver uma aplicação que seja essencial para os praticantes dos três desportos abordados: o *surf*, o *skate* e o *snowboard*. Após a investigação feita, não foi encontrada uma plataforma que satisfaça estes três desportos numa só aplicação, no entanto, estes desportos estão bastante relacionados e muitas pessoas praticam mais do que um. Espera-se assim criar uma aplicação que vá de encontro a este nicho de mercado e que sirva também os atletas que se dedicam apenas a um desporto, fornecendo-lhes informação detalhada dos locais onde praticam ou que lhes dê a conhecer novos locais. A dinamização dos locais físicos é algo que a aplicação pretende promover. Sendo uma ferramenta que permite juntar várias pessoas no mesmo local é expetável a sua importância no turismo local. Essa dinamização será possível através da competitividade que a aplicação pretende gerar, onde os utilizadores poderão partilhar o local atual onde se encontram e assim incentivar os restantes utilizadores a dirigirem-se ao local.

ESTRUTURA DE CAPÍTULOS

Neste subcapítulo será referida a estrutura do documento, mencionando o que aborda cada um dos capítulos.
No primeiro capítulo é introduzido o tema da dissertação, explicando como surgiu a motivação para o abordar e como é que este se enquadra dentro das áreas do design e multimédia e explora as competências do mestrando. São introduzidos os contributos, explicando o que se pretende atingir com o trabalho a desenvolver e é feita uma introdução ao que será estudado e analisado durante o processo de desenvolvimento.

O segundo capítulo aborda o contexto em que o projeto está inserido. O leitor é introduzido aos desportos de ação em geral e posteriormente a cada um dos desportos abordados, sendo dedicado um subcapítulo a cada um deles. De seguida é dada atenção às semelhanças e diferenças entre cada um dos três desportos e são referidas algumas das culturas que lhes estão associadas. O último subcapítulo explica o que os atletas procuram nos locais onde praticam e quais as condições que influenciam a escolha dos locais.

O terceiro capítulo é dedicado ao Estado da Arte e está divido em duas secções. Na primeira é feita uma introdução e é dado algum contexto sobre os dispositivos móveis e quais as tecnologias existentes no mercado móvel. Na segunda secção são apresentados os trabalhos que se relacionam com o projeto que se pretende desenvolver e que trouxeram inovações na área. Esta secção está dividida em três subcapítulos: aplicações de georreferenciação, plataformas online relacionadas com desporto de ação e geolocalização e aplicações móveis relacionadas.

No quarto capítulo são definidos os objetivos a atingir e quais as metodologias usadas durante o desenvolvimento desta dissertação, como se pretendem atingir e qual o resultado final de cada etapa.

O quinto capítulo apresenta o plano de trabalho seguido, enunciando todas as tarefas do mesmo e explicando as implicações e interdependências de cada fase de desenvolvimento.

No sexto capítulo é dado início à explicação do processo que levou ao desenvolvimento da aplicação, sendo neste capítulo abordada a prototipagem da mesma.

O sétimo capítulo aborda as decisões de design tomadas. Neste capítulo é apresentada a identidade visual da aplicação e todos os ecrãs desenhados.

No oitavo capítulo é explicado todo o processo de desenvolvimento programático da aplicação, incluindo uma seleção de tecnologias usadas e as linguagens usadas durante o processo.

O nono capítulo aborda a avaliação de usabilidade efetuada à aplicação junto a utilizadores e apresenta os resultados obtidos.

O décimo capítulo contém a conclusão desta dissertação, abordando o trabalho futuro a desenvolver.

Por fim, o décimo primeiro capítulo contém a bibliografia usada durante o desenvolvimento da dissertação, incluindo todas as referências que foram usadas ao longo do documento e referências de outras fontes que foram úteis para o trabalho desenvolvido.

2. CONTEXTO

DESPORTO DE AÇÃO

Os desportos de ação, também chamados de desportos radicais, de aventura ou extremos, são as atividades que estão normalmente associadas a um perigo iminente, ao ar livre e ao lazer. Este tipo de atividades requer bastante treino e o uso de equipamento adequado para evitar acidentes ou consequências de acidentes.

É difícil datar exatamente quando surgiu o termo "desportos de ação" para se referir a certos desportos modernos, no entanto, estima-se que foi no início da década de 1970 quando a escalada e as maratonas começaram a ganhar popularidade, sendo estes dois desportos considerados extremos devido à sua dificuldade e exigência para com os atletas (Matt Williamson, s. d.).

Apesar do termo ter surgido e ter ganho popularidade nas últimas décadas, especialmente na década de 1950, o desporto de ação surgiu bastante antes disso mas não tinha esse nome associado. O ser humano sempre tentou puxar os seus limites físicos ao máximo, sendo assim difícil encontrar uma data em que se possa dizer que marcou o início dos desportos de ação.

Todos os desportos de ação partilham do fator de imprevisibilidade. Sendo desportos de ar livre, muitos deles praticados em locais naturais, podem ser afetados pelas condições meteorológicas, o ambiente e o terreno. São estas variáveis e o risco a que os praticantes se submetem que distinguem estes desportos extremos dos desportos ditos normais (LIV, 2011).

Os desportos de ação podem ser classificados em três grandes áreas que os distinguem pelo ambiente onde são praticados: ar, água e terra. Exemplos de desportos de ação são: o *base jumping*, paraquedismo, *BMX*, escalada, *skate*, *snowboard* e *surf*.

Nas últimas duas décadas o desporto de ação ganhou grande relevância devido ao surgimento dos campeonatos *X Games* em 1998, principalmente pelo grande número de patrocinadores e de atletas profissionais que comparecem no evento.

DESPORTOS A ABORDAR

Durante o desenvolvimento desta dissertação serão abordados três desportos que foram escolhidos devido ao período de tempo para o desenvolvimento da dissertação e de modo a obter melhores resultados nas fases de design e implementação do protótipo.

Os três desportos a abordar são o surf, o skate e o snowboard. A escolha destes três desportos deve-se às relações que existem entre eles e por se realizarem em locais diferentes, permitindo assim pensar como se poderá adaptar a aplicação a estes desportos de modo a manter uma coerência gráfica e possibilitar as mesmas interações em cada um dos desportos explorados. Isto será a base para que no futuro a aplicação se possa expandir e possa oferecer maior número de desportos de ação.

Nos próximos capítulos serão explicados cada um dos desportos escolhidos e as relações existentes entre eles.

SURF

O *surf* é um dos desportos que se insere nos desportos de ação, nomeadamente nos de água. Trata-se de um desporto realizado à superfície da água, usando uma prancha de *surf*, denominada *surfboard*, que permite ao atleta deslizar em pé sobre a água. O atleta efetua movimentos com diferentes graus de dificuldade ao navegar na frente da onda que o transporta para a costa. Um praticante de *surf* é chamado surfista. O desporto é praticado essencialmente no oceano devido à existência constante de ondas a rebentar junto à costa mas também pode ser praticado em lagos, rios e ondas artificiais produzidas em piscinas de ondas. O *surf* moderno como o conhecemos é descrito como um surfista de pé em cima de uma prancha de *surf*, também chamado de *stand-up surfing*, no entanto, não foi nesta forma que o *surf* surgiu e não é a única forma de o praticar, sendo que o termo *surf* se pode referir apenas ao ato de movimento sobre uma onda com ou sem prancha.

Figura 1. Atleta de *surf* numa competição em Mavericks, California. Foto de Shalom Jacobovitz. [1]

Uma das variantes mais praticadas é o *paddle boarding* que consiste numa prancha em que o atleta se coloca de joelhos e dá propulsão com os braços. Nesta variante o objetivo não é apanhar ondas mas sim fazer corridas em oceano aberto, que podem ser feitas entre ilhas ou viagens entre áreas costeiras. Outra variante bastante praticada que consiste em também apanhar ondas tal como o *stand-up surf* é o *bodyboard*. No *bodyboard* o surfista surfa a onda de barriga para baixo e numa prancha diferente da prancha de *surf*, a prancha de *bodyboard*. O *surf* também pode ser efetuado sem o uso de uma prancha, neste caso é o próprio corpo do surfista que desliza sobre a onda até à costa.

Não existe um registo do início da prática do *surf* pelo ser humano. Usar as ondas como meio de diversão e lazer existe provavelmente desde que os primeiros seres humanos começaram a nadar nos oceanos. Apesar de não

[1] Imagem disponível no *link:* http://en.wikipedia.org/wiki/Mavericks,_California#mediaviewer/File:2010_mavericks_competition.jpg

haver registos de onde foram apanhadas as primeiras ondas, a origem do *surf* é atribuída às ilhas da Polinésia, situadas no Pacífico, onde o *surf* era uma forma de demonstrar qualidades e importância na sociedade, sendo normalmente praticado por líderes políticos (Miss Celania, 2012).

Uma vez que para a prática da maioria das variantes de *surf* e do *surf* moderno são necessárias ondas, parte da vida de um surfista é passada fora de água à procura das ondas ideais para surfar. Hoje em dia existem modelos de dados que permitem prever as condições de um determinado local junto à costa e determinar se será um bom dia para a prática de *surf*. Existem vários *websites* que fazem previsão de condições de *surf* e permitem aos atletas identificar quais os dias ideais para saírem de casa com as pranchas. No entanto, nem todos os dias existem condições para a prática de *surf*, levando assim os surfistas a procurar outras formas de surfar. Esta perseverança dos surfistas levou à criação de outros dois desportos muito comuns nos dias de hoje, também eles considerados desportos de ação, o *skate* e o *snowboard*.

SKATE

O *skate* é um dos vários desportos que nasceram com a influência do *surf*. O desporto consiste em mover-se em cima de uma prancha de *skate*, chamada de *skateboard*, no solo, ultrapassando ou deslizando sobre obstáculos. O atleta tem de manter o equilíbrio e, embora possa apenas andar em cima da prancha pode também efetuar manobras com vários graus de dificuldade. Um praticante de *skate* é chamado de *skater*. Apesar do *skate* ter as suas origens provenientes do *surf* hoje em dia a maioria dos praticantes dedica-se apenas a uma das modalidades.

Os primeiros tipos de *skateboards* assemelhavam-se mais a *scooters* e surgiram no início de 1900 (Brooke, 1999).O *skate* surgiu na Califórnia entre as décadas de 1940 e 1950, numa tentativa dos surfistas da zona poderem usufruir também da sensação do *surf* no solo. Para isso, criaram caixas de madeira com rodas de patins, inicialmente muito simples, que permitiam divertirem-se também na época de marés baixas e sem ondas, tentado reproduzir as manobras que efetuavam no mar.

Figura 2. *Skater* num local urbano. Foto de Vjeran Pavic. [1]

O *skate* surgiu com o nome de *sidewalk surf* e rapidamente as caixas de madeira assumiram a forma de pranchas, surgindo assim o *skate* como o conhecemos hoje em dia. Esta prática ganhou força na década de 60 com várias marcas
a começarem a produzir os seus *skateboards* e a promoverem os seus produtos e também com o aparecimento de campeonatos dedicados ao *skate*.
No entanto, na segunda metade da década de 60, o *skate* começou a perder força devido ao perigo associado ao desporto, sendo que as lojas começaram a reduzir as vendas face a este perigo (Owen, 2013). Esta quebra nas vendas manteve-se até ao início da década de 70, afetando a popularidade do desporto. No início dos anos 70, Frank Nasworthy revolucionou o desporto

[1] Imagem disponível no *link:* https://www.flickr.com/photos/vjeran_pavic/8992749657/

ao introduzir rodas para os *skateboards* feitas de poliuretano. Estas rodas permitiam melhores condições, maior aderência ao solo, mais conforto e mais controlo sobre o *skateboard*, facilitando assim a adaptação (Brooke, 1999). No entanto, os *skaters* com alguma experiência continuavam a preferir as rodas feitas de aço, uma vez que lhes permitiam atingir maiores velocidades. Ao ver a rápida adoção das rodas que vendia, Frank decidiu criar a empresa *Cadillac Wheels* e moveu-se para a Califórnia em 1972, ajudando assim ao renascimento do *skate* (Owen, 2013). As vendas inicialmente eram feitas em lojas de *surf* tendo em conta que o *skate* ainda não era o desporto com uma dimensão muito elevada mas era um desporto que despertava o interesse dos surfistas. Com o nascimento de uma revista dedicada somente ao *skate* e a popularização das rodas dentro da comunidade, as vendas das rodas da empresa de Frank aumentaram exponencialmente, sendo que em 1975
os números apontavam para 300 mil conjuntos de rodas vendidas por ano (Gonzalez, 2011). Embora esta introdução de rodas de poliuretano tenha revolucionado o *skate* e aumentado a sua popularidade, o aparecimento de rolamentos de precisão nos *skateboards* permitiu aos atletas efetuar novas manobras e a ter mais controlo sobre o *skateboard* e rapidamente começaram a fazer uso de piscinas sem água para criar manobras mais arriscadas. A isto chamou-se o movimento "vert" do *skate*. Este movimento veio provocar um decréscimo no uso dos parques de *skate*, pois os *skaters* que aderiam ao movimento "vert" construíam as suas próprias rampas e os restantes praticavam estilo livre nas ruas. A introdução deste movimento veio mais uma vez provocar um decréscimo na popularidade do *skate* no início dos anos 80 (Brooke, 1999).

Nas últimas décadas o *skate* teve altos e baixos na sua popularidade, devido principalmente a baixa qualidade dos produtos e problemas de segurança. No entanto, o *skate* continua a ser um dos desportos de ação mais praticados e com menos percentagem de atletas com lesões em comparação a outros desportos (Brooke, 1999).

SNOWBOARD

O *snowboard* é um desporto de neve, praticado nas encostas de montanhas, onde o atleta tem que se equilibrar numa prancha e fazer uma descida com a ajuda da força gravítica. O *snowboard* é também considerado um desporto de ação devido aos perigos associados com as montanhas e as velocidades atingidas pelos praticantes.

Figura 3. *Snowboarder* em montanha. Foto de Zach Dischner. [1]

Embora as raízes do *snowboard* venham definitivamente do *surf*, o *snowboard* é um desporto bastante recente, ainda mais recente que o *skate*. Enquanto que a invenção do *surf* e do *skate* não pode ser atribuída a uma única pessoa, sendo até difícil datar o momento exato em que surgiram, no *snowboard* podemos atribuir a invenção a Sherman Poppen em 1965. Na manhã de Natal de 1965, Sherman Poppen foi à sua garagem, juntou dois skis num só, subiu a colina no seu quintal e começou a surfar na neve (MacArthur, 2010). Rapidamente esta invenção de Poppen se tornou um sucesso e assim nasceu o Snurfer — nome dado à prancha que resulta da junção das palavras snow e surfer. Algumas semanas após Poppen adicionou uma corda à parte da frente da prancha que permitia virar mais facilmente e prevenir a prancha de se afastar em caso de queda e patenteou-a, vendendo cerca de um milhão de unidades nos 15 anos seguintes (MacArthur, 2010).

Em 1976 surgiu a primeira empresa de *snowboard*, fundada por Dimitrije Milovich. Dimitrije, um surfista, é conhecido por ter criado a primeira prancha de *snowboard* moderna, apelidada de "Swallowtail", em 1972. Esta prancha pretendia transmitir a sensação de uma prancha de *surf* mas assemelhar-se a um esqui. A sua empresa "Winterstick", fundada em 1976, começou a vender dois tipos de pranchas de *snowboard*, a "Swallowtail" e a "Round Tail" em 11 países. No entanto, promover a venda das pranchas revelou-se uma tarefa difícil, uma vez que o desporto ainda não tinha surgido com força suficiente para que os revendedores mostrassem interesse nas suas

[1] Imagem disponível no *link:* https://www.flickr.com/photos/35557234@N07/4196764742/

invenções, levando Milovich a fechar a sua empresa (Blacksberg, 2009).

Jake Burton e Tom Sims são os principais responsáveis por tornar o *snowboard* um desporto com bastante popularidade. Rivais nas vendas, Burtons e Sims venderam as suas primeiras pranchas nas épocas de 1978 e 1979, ambos sentiram dificuldades nas vendas mas ambos insistiram e durante uma década tornaram-se as maiores forças no que diz respeito a marcas de pranchas de *snowboard*. Tom Sims, que tal como Milovich era um surfista, preferia praticar *surf* em vez de gerir a sua empresa. Por outro lado, Burton era um homem de negócios e o negócio do *snowboard* era o seu maior interesse, o que lhe permitiu, alguns anos depois, ser o número um nas vendas de material de *snowboard* (MacArthur, 2010).

O ano de 1982 marca o primeiro campeonato de *snowboard* realizado em Vermont e em 1985 é lançada a revista internacional de *snowboard* "Absolutely Radical" por Tom Hsieh, renomeada no ano seguinte para "International Snowboard Magazine". A última edição da revista acabou por ser em 1991 devido às dificuldades em competir com revistas melhores e melhor distribuídas como a "Transworld Snowboarding".

Embora na década de 80 o *snowboard* já tivesse bastante relevância e praticantes, muitas estâncias de esqui não permitiam a entrada a *snowboarders* principalmente devido a não pretenderem que estes afetassem os seus clientes de esqui. Foi realizada uma campanha diplomática cujo sucesso levou a um aumento das estâncias que permitiam a prática de *snowboard* para 476. Também na década de 80 começaram a surgir os primeiros halfpipes nas estâncias de esqui, que eram, inicialmente, bastante pequenos e exigiam bastante mão de obra na sua manutenção. Foi em 1990 que o agricultor Doug Waugh recebeu autorização para construir uma máquina que permitia fazer halfpipes de forma fácil e assim, em 1992, o "Pipe Dragon" foi concluído e acabou por se tornar uma necessidade nas estâncias de esqui (MacArthur, 2010).

Um dos momentos mais importantes na história do *snowboard* foi a sua estreia nos Jogos Olímpicos de Inverno de 1998 no Japão.

AS SEMELHANÇAS/DIFERENÇAS ENTRE O *SURF*, O *SKATE* E O *SNOWBOARD* E AS CULTURAS ASSOCIADAS

Como explicado nos pontos anteriores, o *skate* e o *snowboard* tiveram como sua origem e inspiração o *surf*. Ambos surgiram para preencher a necessidade dos surfistas quando não havia ondas, sendo que o *snowboard* pode-se considerar como o desporto que evolui a partir do *skate*, uma vez que surgiu depois (Guardino, 2010). As semelhanças entre os três desportos são bastante evidentes, todos se realizam em cima de pranchas onde o equilíbrio é uma habilidade essencial que o atleta tem de possuir. No *surf* e no *skate* a prancha não está presa aos pés, numa eventual queda a prancha é livre de ir até onde o pavimento e a água o permitirem (Taylor, 2010), já no *snowboard*, a prancha é presa aos pés para permitir mudanças de direção mais facilmente.

Apesar da semelhança entre os desportos, as suas diferenças são o que os tornam desportos diferentes e que requerem aprendizagem mesmo para quem já pratica um dos três, o que significa que alguém que seja muito bom praticante de um desporto não será necessariamente bom num dos outros (Taylor, 2010). As principais diferenças são claramente os meios onde são praticados e a forma como se ganha movimento. O *surf* é praticado na água e o movimento da prancha é gerado pelas correntes de água e pelas ondas. No *snowboard*, praticado na neve e em encostas de montanhas, o movimento do atleta deve-se à aceleração criada pela força gravítica numa descida duma encosta. Já o *skate* é praticado em superfícies duras e geralmente criadas pelo Homem, desde o asfalto à madeira. O movimento do atleta é criado pelo próprio com um dos seus pés a exercer força no pavimento e a impulsionar a prancha que se move por meio das rodas na sua base. Uma das grandes diferenças entre estes três desportos é o custo inicial para começar a praticar cada um deles, que vai desde o custo da prancha até ao custo do material associado.

O *skate* é sem dúvida o desporto mais barato em termos de custos iniciais e em longo termo, pois a única ferramenta necessária para a sua prática é um *skateboard* e os locais onde se pode praticar são praticamente quaisquer locais com pavimento normalmente sem nenhum custo associado (Guardino, 2010). Por opção, o praticante pode também adquirir material de proteção. No caso do *surf*, o custo maior para um iniciante é a prancha, podendo também adquirir um fato térmico para águas mais frias, sendo que os locais em que pode ser praticado não são tão abundantes como os locais onde se pode praticar *skate*. Já o *snowboard* é o desporto entre estes três que mais caro fica, mesmo para quem já é praticante, pois além do custo da prancha e de todo o equipamento necessário para manter o atleta quente em condições de temperaturas bastantes baixas (Guardino, 2010), há também o custo da deslocação para os locais onde se pode praticar, que são bem mais raros

comparados com os locais propícios para o *surf* e *skate*, e o custo que estes locais impõem aos praticantes para que possam ser usados.

Cada um destes desportos tem uma cultura que se reflete nos praticantes. Ambos os desportos influenciaram a moda, a música e até a maneira como falam, onde o vocabulário é normalmente criado para definir nomes de manobras ou novos estilos dentro de um destes desportos. Os *skaters* são livres de praticar *skate* em praticamente qualquer sítio que pretendam. O *skate* é assim um desporto independente, sem regras e que atrai pessoas com mentalidade de liberdade e com expressão criativa. É um desporto com uma notável energia, associado bastante ao público mais jovem. O *skate* nasceu com base no *surf* mas nos últimos anos o *skate* tem também contribuído bastante para o *surf*. Os surfistas têm realizado algumas das manobras que se achavam impossíveis de fazer em cima duma prancha de *surf* e a inspiração dessas manobras surgem devido à influência do *skate*.

A cultura do *surf* está muito ligada ao oceano. A procura pelas melhores ondas e pelas condições meteorológicas ideais faz parte da cultura do *surf*. Também o fato de se praticar principalmente em zonas costeiras levou a cultura de praia a influenciar os surfistas e vice-versa. O localismo é um dos aspetos da cultura do *surf* que mais se manifesta. Tendo em conta que nem todos os locais são bons para a prática do *surf*, muitos locais têm um grupo de surfistas que tentam impor uma hierarquia que decide a escolha das ondas e quem pode ou não praticar no local. Estes grupos normalmente são constituídos por pessoas que vivem perto do local ou que praticam lá frequentemente. O localismo pode-se manifestar de forma saudável quando os surfistas mais experientes coordenam os iniciantes para evitar acidentes mas também se pode manifestar de forma errada quando existem desentendimentos por um praticante não respeitar essa hierarquia. Com o crescimento do *surf* nos últimos anos a manifestação do localismo tem vindo aos poucos a desaparecer. O vocabulário no *surf* é também influenciado pela sua cultura, sendo vulgar o uso de palavras ou expressões como "flat", "swell", "tow in", entre muitas outras ("Surf Terms", 2011).

A cultura do *snowboard* nasceu do impacto entre os praticantes de esqui e os *snowboarders*. A cultura dos *snowboard*ers rapidamente se tornou uma junção de estilos urbanos e suburbanos, assemelhando-se assim bastante às culturas do *surf* e principalmente do *skate*. Ultimamente a cultura do *snowboard* tem tido cada vez um impacto menor nos seus praticantes e isto deve-se à diversidade internacional de praticantes e fãs e à redução de estereótipos associados ao desporto.

A PROCURA DE LOCAIS PARA A PRÁTICA DO *SURF*, *SKATE* E *SNOWBOARD*

Cada desporto necessita de um local específico para ser praticado e os desportos de ação não são exceção. Para um iniciante, além de ter que comprar o material necessário, procurar um local onde possam praticar é das primeiras dificuldades com que se deparam, ainda mais difícil se torna no caso de se tratar de um desporto de ação, uma vez que os locais normalmente não estão referenciados nas maiores plataformas de geolocalização que existem hoje em dia. Mesmo para os atletas com bastante experiência procurar um novo local para praticarem à procura de melhores condições não é tarefa fácil. Surfistas, *skaters* e *snowboarders* sentem esta dificuldade, uns mais que os outros. De seguida serão expostas algumas das condições que os praticantes de cada um destes três desportos procuram num local.

No *surf*, sendo um desporto que resulta da interação do homem com a natureza, são necessárias as condições naturais ideais para que se possa praticar e as condições dependem naturalmente do local. Vários são os fatores que influenciam o estado do mar e que influenciam a escolha do local, começando pela experiência do praticante. Um iniciante deve procurar locais onde existam nadadores salvadores no caso de algum problema surgir e devem procurar locais conhecidos pela rebentação das ondas ser fraca e onde existam poucos ou nenhuns surfistas também no mar. Os fatores que influenciam a escolha de um local por atletas com alguma experiência são diferentes. Atletas com mais experiência procuram por ondas maiores e de maior duração e estas são resultado de várias condições, sendo as principais: o fundo do mar, o vento e as marés. A velocidade do vento está diretamente ligada à altura da ondulação e à duração das vagas, ou seja, mais vento proporciona maiores ondas e maiores períodos de duração de cada onda.

Embora existam bastantes locais onde se pode praticar *surf*, o número de locais para praticar *skate* é ainda maior e praticamente infinito, uma vez que os *skaters* podem criar e alterar os seus próprios locais. Existem locais dedicados para os praticantes de *skate*, os *skateparks*, os quais se podem encontrar hoje em dia na maioria das cidades, no entanto, estes *skateparks* têm o problema de poderem ficar superlotados devido à sua procura e normalmente têm bastantes restrições. Os *skaters* procuram liberdade nos locais onde praticam e por isso procuram lugares dentro das cidades que lhes possam dar essa liberdade e que tenham vários obstáculos para que possam efetuar as suas manobras. Desde parques de estacionamento, escolas, a redondezas de edifícios, qualquer local dentro de uma cidade pode ser um ótimo local para praticar *skate*. Para os praticantes que vivem fora da cidade também é possível praticarem, qualquer local com um bom pavimento, desde que seja permitido, tem condições para um *skater*.

Em contraste com os *skate*, os locais para praticar *snowboard* são bastante mais limitados e a maioria tem várias regras que têm de ser cumpridas pelos

snowboarders. Os iniciantes no *snowboard* procuram normalmente as estâncias de esqui e *snowboard* para aprender pois estas estão melhor sinalizadas, os percursos são separados por nível de dificuldade e existem instrutores que auxiliam na aprendizagem. Existem centenas de estâncias de esqui no mundo que permitem praticantes de *snowboard*, no entanto, existem muitos mais locais onde é possível praticar e são estes locais que normalmente são procurados pelos *snowboarders* com bastante experiência. Sendo um desporto de ação, os atletas procuram o máximo de adrenalina, e é nas encostas de montanhas mais íngremes e consequentemente com maior grau de dificuldade que os atletas de *snowboard* mais experientes procuram a melhor descida.

3. ESTADO DA ARTE

DISPOSITIVOS MÓVEIS E TECNOLOGIAS ATUAIS

Os dispositivos móveis estão em todo o lado, acompanham-nos para onde vamos e fazem parte do nosso dia a dia. Distantes estão os anos em que os telemóveis serviam apenas para a sua função primária: a comunicação entre indivíduos. Cada vez mais são incluídas novas funcionalidades num dispositivo que cabe no bolso e os telemóveis atuais são algo mais do que um simples dispositivo de comunicação. Estes telemóveis com várias funcionalidades e com a possibilidade de executarem várias aplicações são apelidados de *smartphones*.

Um *smartphone* é um telemóvel apesar de ser muitas vezes considerado um computador de bolso por possuir inúmeras caraterísticas semelhantes às de um computador, como a possibilidade de aceder à *Web* e ao *email*. Os *smartphones* são constituídos por um ecrã de pequenas dimensões que pode ter a função de *input* através de tecnologia de ecrãs sensíveis ao toque. Caso não seja o ecrã com a função de *input* como observamos na maioria dos *smartphones* atuais, o *input* é feito através de um teclado físico. O que leva um telemóvel a ser considerado um *smartphone* é o fato de estes possuírem um sistema operativo que controla todas as operações de *input* e *output* e permite a instalação de outro software, nomeadamente aplicações que foram popularizadas com o nome de *mobile apps* através da abreviação da palavra "applications". Estas *apps* deram nova vida aos dispositivos móveis atuais e continuam a definir os limites do que é possível fazer com um *smartphone*. Além da possibilidade de executar inúmeras aplicações, os *smartphones* possuem vários sensores e hardware que permitem aumentar a sua versatilidade. Além do hardware que permite conetividade às redes móveis para efetuar comunicações e à Internet, temos também o *Wi-Fi*, *GPS*, sensores de movimento, rotação e aceleração, câmeras fotográficas, colunas e mais recentemente sensores de leitura biométrica, tudo isto em dispositivos alimentados por uma bateria incluída. Esta quantidade de hardware com a possibilidade de ser usado por qualquer aplicação permite a criação de inúmeras aplicações que fazem uso de um ou mais componentes para realizar a sua função.

Após o surgimento dos *smartphones* e do aumento de popularidade devido à sua aceitação pelo público, o mercado dos dispositivos móveis evoluiu e com essa evolução apareceram os *tablets*. Os *tablets* são dispositivos com funcionalidades semelhantes às dos *smartphones* mas com dimensões superiores - o que lhes confere uma menor portabilidade. Os primeiros *tablets* utilizavam os mesmos Sistemas Operativos encontrados nos computadores pessoais da altura e pouco acrescentaram à indústria de dispositivos móveis, pois assemelhavam-se demasiado a computadores portáteis.

Os *tablets* e os *smartphones* como os conhecemos hoje em dia devem-se especialmente à *Apple*, empresa norte-americana sediada na Califórnia. Embora os conceitos já existissem previamente e já tivessem sido explorados por várias empresas do meio tecnológico, foi a *Apple* que trouxe os produtos

que revolucionaram o mercado dos *smartphones* e *tablets* e que foram adotados massivamente pelos consumidores, esses produtos foram o iPhone e o iPad, respetivamente.

Steve Jobs, 2007

"Every once in a while a revolutionary product comes along that changes everything."

A apresentação do primeiro iPhone ao mundo foi feita em 2007 e revolucionou o mercado dos dispositivos móveis, colocando todas as empresas que produziam telemóveis em clara desvantagem em relação à *Apple*. O primeiro iPhone consistia na junção de várias tecnologias de ponta das quais se podem destacar o seu ecrã capacitivo que funcionava como *input* por toque e a tecnologia *Wi-Fi*. Uma das decisões de design que diferenciou este *smartphone* de outros já existentes foi o seu ecrã de grandes dimensões que ocupava uma grande parte da dimensão total do dispositivo, principalmente devido à decisão de não incluir um teclado físico e outros botões. O maior fator de diferenciação foi, no entanto, o software que acompanhava o dispositivo. O iPhone foi lançado juntamente com o seu sistema operativo também desenvolvido pela *Apple*, o iOS, e um conjunto de aplicações com uma interface bastante apelativa e funcional, as quais permitiam uma ótima experiência para o utilizador. Exemplos de aplicações existentes eram o *email*, o Youtube e o iPod. No entanto, era no *browser*, o Safari, que o iPhone se distinguia da maioria dos dispositivos móveis existentes na altura. O Safari, na primeira versão do iOS, era um ótimo *browser*, bastante fluido e responsivo, notável pela velocidade no carregamento de páginas *Web* e pela qualidade de visualização de páginas que não tivessem sido desenhadas para se adaptarem a dispositivos móveis. O iPhone e o iOS foram atualizados ao longo dos anos, sendo no momento da escrita desta dissertação o iPhone 5C e o iPhone 5S os modelos atualmente em comercialização e o iOS 7 a versão mais atual do sistema operativo para dispositivos móveis da *Apple*.

Após a revolução no mercado dos *smartphones*, em 2010 a Apple lançou o iPad, o *tablet* que se destacou e popularizou o conceito de *tablet* no mercado. Antes do iPad, várias tentativas de produtos semelhantes tinham sido realizadas, até por parte da *Apple*, no entanto, a tecnologia por toque e toda a qualidade do software que foi herdada do iPhone fez com que o iPad fosse um sucesso na data do seu lançamento. Os *tablets* têm a sua grande vantagem nas dimensões do ecrã e são usados maioritariamente para ver conteúdos publicados, como vídeo, imagens e notícias. Assim como a maioria dos *smartphones*, os *tablets* possuem uma vasta quantidade de hardware que vai desde comunicação por *Wi-Fi* até à localização por satélite através de GPS.

Com o lançamento do iPhone e do iPad, a *Apple* colocou os seus concorrentes mais próximos em desvantagem e conquistou grande parte do mercado de dispositivos móveis, sendo que os seus competidores tiveram, rapidamente, que arranjar soluções para competir no mercado, competição essa que continua a permitir o lançamento de produtos cada vez mais sofisticados e que vão de encontro às necessidades do consumidor. Imediatamente após o lançamento do primeiro iPhone, a *Google* respondeu ainda em 2007 com o seu sistema operativo Android que tinha estado em

desenvolvimento desde 2003. O Android é um sistema operativo, financiado e mais tarde comprado pela *Google*, que assenta na filosofia de código aberto. Esta filosofia permite que possa ser modificado e distribuído por qualquer individual ou empresas, permitindo assim as empresas produtoras de *smartphones* adaptarem o código aos produtos em desenvolvimento. Além de poder ser incluído nos *smartphones*, o Android pode ser instalado em qualquer dispositivo, uma vez que assenta no *kernel* Linux, o que lhe confere bastante portabilidade. Em 2008 apareceram os primeiros dispositivos com Android como sistema operativo, sendo o primeiro lançado pela *HTC*, o HTC Dream. Atualmente o Android é a plataforma para dispositivos móveis com maior fatia de mercado, ultrapassando os 50% globalmente. Além do iOS e do Android existem outros competidores no mercado com significativamente menos preponderância que estes dois. Exemplos de plataformas com relevância são o BlackBerry 10 desenvolvido pela *BlackBerry* e o Windows Phone da *Microsoft*. Existem ainda outras plataformas como o Firefox OS da *Mozilla Foundation* e o Symbian da *Nokia*, mas estes são bastante menos utilizados em comparação com as plataformas apresentadas anteriormente.

O que levou então a este crescimento do mercado de *smartphones* e *tablets* face aos telemóveis comuns? As aplicações. O software fornecido com os sistemas operativos e aquele que se pode adquirir posteriormente são o principal fator de adesão aos *smartphones* e *tablets*. Analisando comparativamente os *tablets/smartphones* com os computadores pessoais podemos identificar tanto vantagens como desvantagens. As principais desvantagens são claramente o menor poder de processamento e a pequena dimensão dos seus ecrãs. Estas caraterísticas fazem com que, embora possam ser ótimas para organizar e consultar conteúdos em qualquer local, não sejam plataformas preferencias para trabalho. São também ótimas plataformas de informação e lazer, existindo imensas aplicações para ler notícias, ouvir música, ver vídeos e até jogos para todos os gostos. Os *smartphones* revolucionaram também a maneira como as aplicações eram até então distribuídas. Mais uma vez, a *Apple* mudou o conceito de distribuição de aplicações quando introduziu a App Store no lançamento do iOS 2 em 2008. Esta faz a distribuição das aplicações digitalmente nas plataformas iOS, tanto para iPhone, iPad e iPod Touch. No Android temos a Play Store e no sistema operativo da *Microsoft* para dispositivos móveis temos a Windows Phone Store. Além de permitirem o descarregamento e a instalação de aplicações, permitem também a classificação das mesmas e descarregar outro tipo de conteúdos, como música e livros. Os conteúdos disponíveis nestes lojas de aplicações podem ser gratuitos ou pagos.

TRABALHOS RELACIONADOS

Nesta secção irão ser abordados trabalhos relacionados com o tema da dissertação e ferramentas que permitam atualmente colocar em prática o desenvolvimento de uma aplicação para dispositivos móveis. Estes trabalhos serão importantes para analisar o que já foi feito dentro do tema desta dissertação e identificar como é que este projeto poderá ser inovador.

A aplicação a desenvolver fará uso de geolocalização para representar os locais num mapa e a informação a eles associada será adicionada pelos utilizadores ou através do uso de outra plataforma. Posto isto, é necessário analisar plataformas que permitam implementar ferramentas de geolocalização numa aplicação móvel e que forneçam informação de valor que possa ser incluída. Serão assim, de seguida, referenciadas as ferramentas de mapas que podem ser incluídas numa aplicação móvel, após isso serão abordadas plataformas online que forneçam informação de localização e condições sobre os locais onde se pratica os desportos abordados anteriormente e por fim serão abordadas implementações já existentes semelhantes à aplicação que se pretende desenvolver.

APLICAÇÕES DE MAPAS

Google Maps, 2005

O Google Maps é o serviço de mapas mais popular para *smartphones*, sendo a plataforma de mapas por defeito do sistema operativo Android. Os mapas da *Google* foram lançados em 2005 e são no momento os mapas com mais detalhe e mais ambiciosos que existem e que podem ser usados em qualquer dispositivo móvel com acesso à Internet. O Google Maps tem várias funcionalidades das quais se destacam a vista de satélite, que permite ver imagens de satélite de todo o planeta com um grande nível de detalhe, e a vista de rua (*Street View*) que já está presente em mais de 50 países e que permite navegar nas ruas com imagens reais recolhidas pelos veículos da *Google*. Além das funcionalidades que esperamos encontrar num mapa, o Google Maps também fornece direções para viagens, onde o utilizador indica o local de partida, o destino e o meio de transporte e os resultados são devolvidos na forma de um trajeto que indica a distância a percorrer e o tempo estimado de viagem para o meio de transporte escolhido.

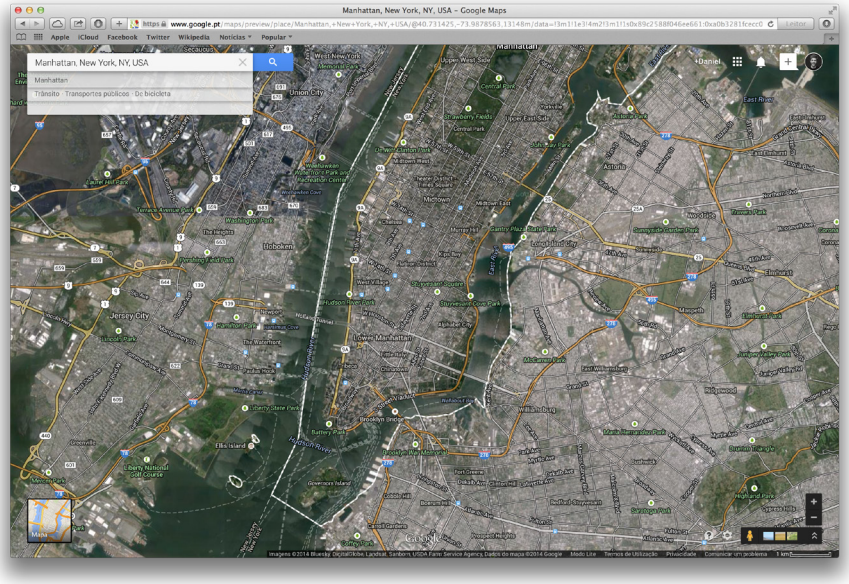


Figura 4. Vista de satélite do Google Maps da cidade Manhattan, Nova Iorque.

O Google Maps pode não só ser acedido nas plataformas móveis através das aplicações nativas desenvolvidas pela *Google*, disponíveis para iOS e Android, como também através da aplicação *Web* para o *browser* de qualquer dispositivo móvel. Os serviços do Google Maps podem ser integrados em aplicações através do uso da API *(Application Programming Interface* - Interface de Programação de Aplicações*)* fornecida e assim oferecer uma experiência de mapas integrada com as suas funcionalidades.
O Google Maps para dispositivos móveis oferece ainda outra funcionalidade que faz uso do hardware de GPS presente na maioria dos *smartphones* para indicar as direções a tomar em tempo real durante um trajeto pré-definido, auxiliando assim os peões ou condutores a chegar ao destino pretendido.

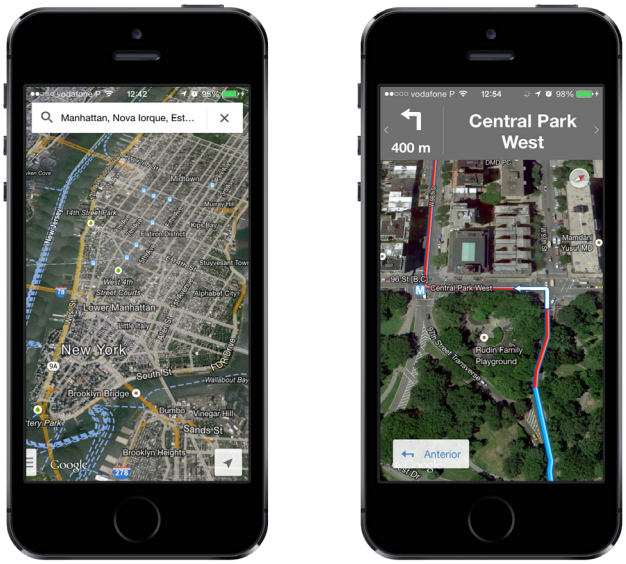


Figura 5. Vista de satélite da aplicação móvel Google Maps para iOS (esquerda) e direções em tempo real (direita).

Os anos de desenvolvimento e as funcionalidades fornecidas pelo Google Maps fazem dele um dos principais serviços de mapas para incluir numa aplicação móvel, tendo em conta também o suporte e comunidade em torno do serviço e as várias implementações já existentes.

Apple Maps, 2013

Até 2013, os mapas que integravam o sistema iOS da *Apple* era fornecidos através do serviço Google Maps. Em outubro de 2013 a *Apple* lançou juntamente com a versão 6 do iOS a sua própria versão dos mapas de maneira a poder competir com as soluções fornecidas pela *Google* no sistema Android. O seu lançamento ficou marcado por bastante criticismo e várias falhas reportadas pelos utilizadores que têm vindo a ser corrigidas. Uma das funcionalidades que se destaca em relação aos mapas da *Google* é o *Flyover* que mostra os edifícios em 3D de algumas das cidades mais populosas do planeta, no entanto, a oferta é bastante menor em comparação à funcionalidade semelhante existente nos Google Maps.

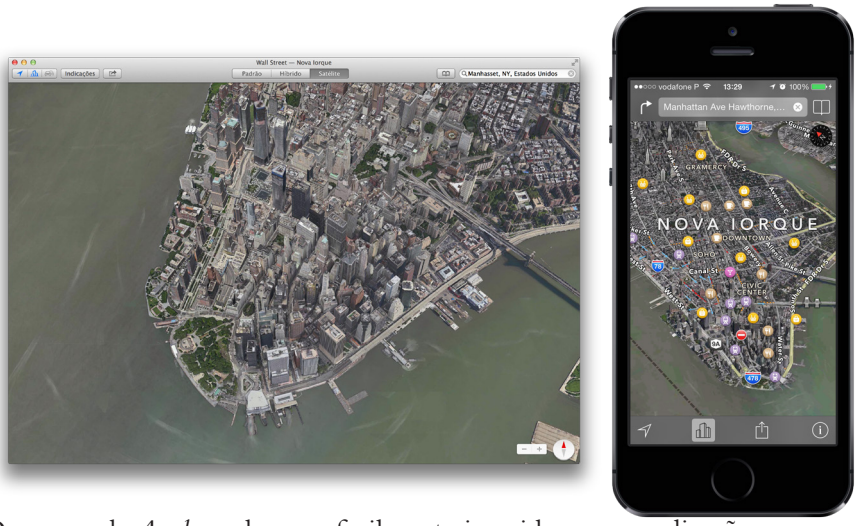


Figura 6. Vista *Flyover* em OS X(esquerda) e em iOS(direita).

Os mapas da *Apple* podem ser facilmente inseridos numa aplicação em desenvolvimento através da *framework MapKit*. A maior parte das funcionalidades existentes na aplicação Maps inserida no iOS estão também presentes em qualquer implementação mais básica desta *framework*.

Os Apple Maps apenas estão disponíveis para as plataformas iOS e OS X e apenas podem ser integrados em aplicações desenvolvidas especificamente para estas plataformas.

PLATAFORMAS ONLINE COM LOCAIS E SUAS INFORMAÇÕES

Extreme Sports Map, 2011

Esta plataforma contém uma fonte de informação online para entusiastas de desporto de ação, com bastantes locais mapeados e com uma componente social e de informação dentro de cada local. A pesquisa pode ser realizada por continente, pelos últimos locais adicionados na plataforma ou pelo tipo de desporto que se pretende praticar. Existem vários tipos de locais pelos quais se pode pesquisar ou que se podem adicionar na plataforma como os parques de *skate*, pistas de *BMX*, locais para a prática de *windsurf* ou *snowboard*.

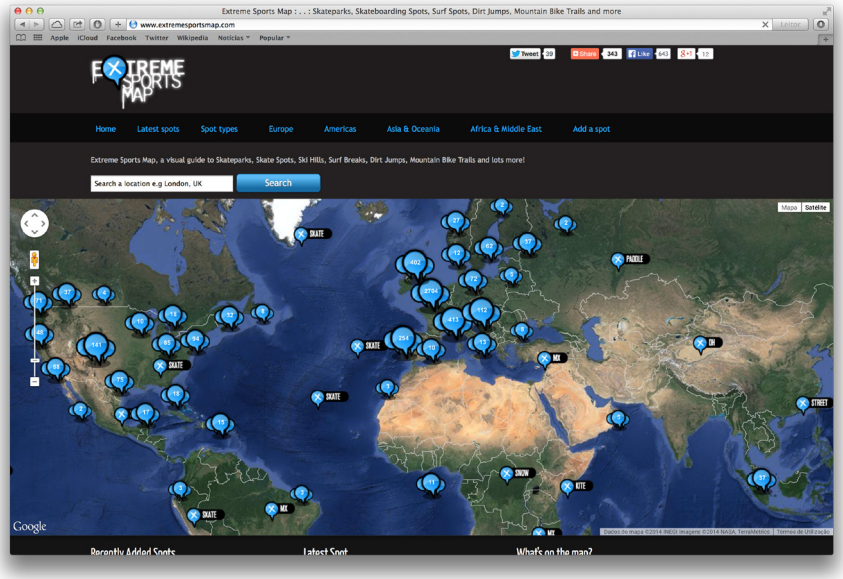


Figura 7. Locais disponíveis em Extreme Sports Map.

Os locais podem ser adicionados por qualquer utilizador da plataforma. Quando é selecionado um local é devolvida uma página com a visualização do mesmo num mapa pormenorizada e uma lista dos locais mais próximos para a prática do mesmo desporto. É possível também ver vídeos associados que foram inseridos pelos utilizadores.

H2S (*How to skate*), 2003

Este site foi lançado em Outubro de 2003 e desde então a sua comunidade cresceu diariamente, o que permitiu criar uma base de dados sobre locais para praticar *skate* por todo o mundo. Além de se dedicar a mostrar os locais para a prática de *skate*, o site também tem outras páginas com informação para ajudar os iniciantes na sua aprendizagem e dicas sobre vários assuntos relacionados com o *skate*, desde técnicas a adquirir, passando por algumas manobras mais difíceis até a instruções de como construir um parque de *skate*. O site contém também uma área para membros onde quem se regista pode amealhar créditos dentro da plataforma ao criar tópicos de discussão, ganhar concursos ou convidar amigos. Esses pontos podem depois ser trocados por material de *skate* como rodas ou *longboards*.

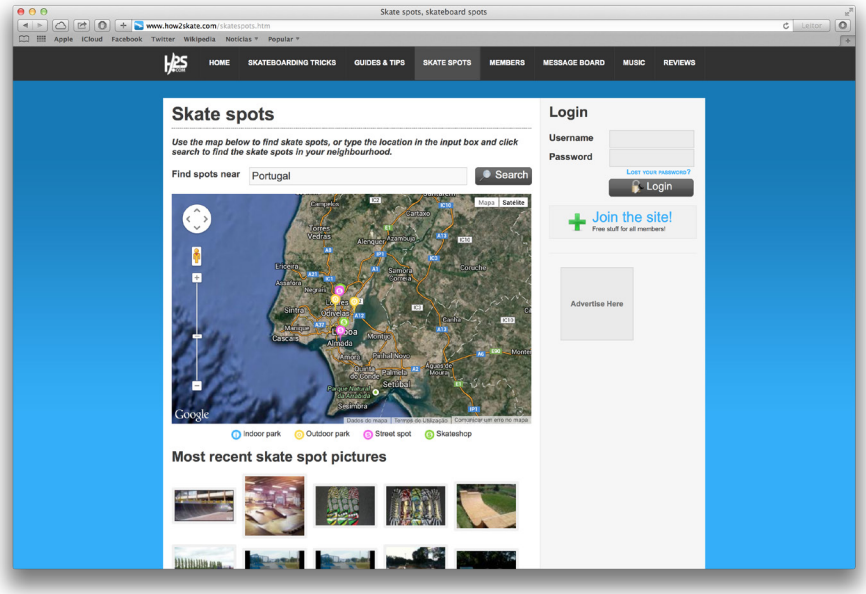


Figura 8. Pesquisa de locais na plataforma H2S.

A pesquisa de locais para praticar *skate* dentro da plataforma pode ser feita através de uma pesquisa pelo nome da cidade ou país onde se pretendem encontrar ou ao navegar simplesmente no mapa. Os locais apresentados no mapa são divididos por quatro tipos de local que são: parques interiores, parques ao ar livre, locais de rua e lojas de *skate*.
Ao selecionar um local é apresentado ao utilizador uma descrição, um endereço e mais detalhes sobre o local, os quais indicam obstáculos existentes, regras de segurança a cumprir e se são ou não de livre entrada.

Windfinder, 1999

O Windfinder é uma plataforma online que não está associada a nenhum desporto de ação mas que fornece dados climatéricos importantes para os praticantes de desportos de água, nomeadamente o *surf*.

O site agrega uma grande quantidade de dados provenientes de mais de 13 mil estações meteorológicas. Esses dados podem ser comprados em forma bruta ou gráfica e o estado atual pode ser visualizado na plataforma online. Os dados atuais podem também ser acedidos através das aplicações móveis disponíveis para os sistemas operativos Android, iOS e Windows Phone

Existem dois tipos de mapas na plataforma: um que permite procurar por locais e ver as condições meteorológicas para o local selecionado e outro tipo que permite observar a direção e intensidade dos ventos, ondulação e outras condições meteorológicas no mundo com bastante detalhe (o utilizador pode escolher a hora pretendida da previsão meteorológica e observar as mudanças durante um certo intervalo que vai até sete dias).

As previsões para os locais existentes são apresentadas sobre a forma de tabela onde são incluídas várias informações como a temperatura do ar, direção e altura das ondas e a precipitação. Existe também uma secção com estatísticas do vento nos meses do ano e outra com as marés.

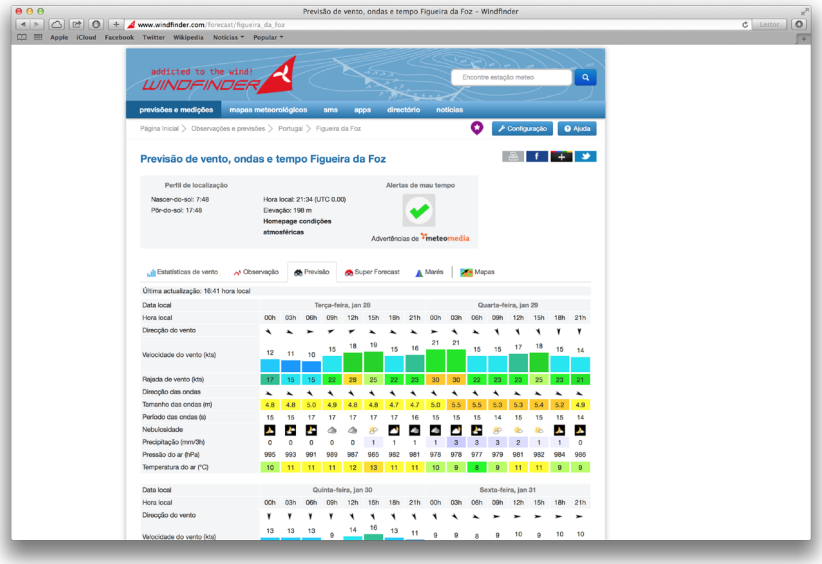


Figura 9. Informação de condições meteorológicas na plataforma Windifnder.

Windguru, 2000

A plataforma Windguru é semelhante ao Windfinder, especializa-se também em previsão meteorológica e é uma das plataformas mais utilizadas por surfistas e atletas de outros desportos de água para saberem as condições meteorológicas de um determinado local. O Windguru oferece previsões para qualquer lugar no mundo.

A grande vantagem do Windguru em relação ao Windfinder é a vasta quantidade de gráficos disponíveis para visualização, sendo que alguns deles apenas estão disponíveis para utilizadores com uma conta Windguru PRO que pode ser adquirida através de um pagamento mensal, anual ou bianual.

Tal como o Windfinder, também são disponibilizadas aplicações para dispositivos móveis para iOS, Android, e Windows Phone.
Além das previsões meteorológicas também estão disponíveis arquivos de previsões meteorológicas anteriores para poderem ser consultados, previsão de marés por local e existe também um fórum de discussão para os utilizadores registados.

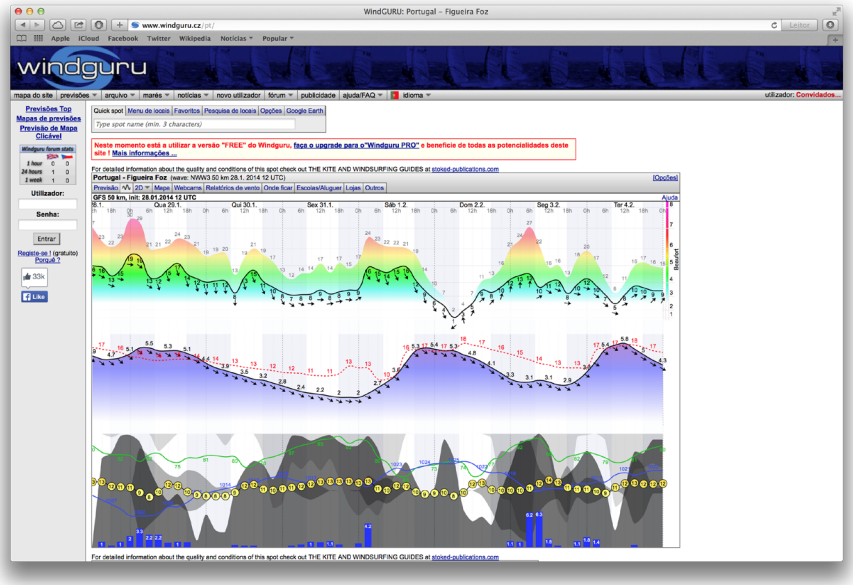


Figura 10. Informação gráfica das condições meteorológicas no site Windguru.

World Snowboard Guide, 1997

O World Snowboard Guide é um site dedicado ao *snowboard* com uma boa organização das estâncias existentes onde se pode praticar e que também fornece aos utilizadores outros serviços como loja, notícias e eventos.

A área de pesquisa de estâncias está muito bem organizada permitindo efetuar uma pesquisa por país, aeroporto mais próximo ou, utilizando filtros avançados, é também possível filtrar os resultados pelo(s) estilo(s) de *snowboard* que é possível praticar, pelas condições e caraterísticas da pista e pela sua classificação ou preço de entrada. Os resultados devolvidos podem ser apresentados numa lista ou num mapa. Na vista de lista é possível verificar a classificação da estância numa escala de 1 a 10 e qual o país onde está localizada. Na vista de mapa é possível observar todas as estâncias presentes na plataforma.

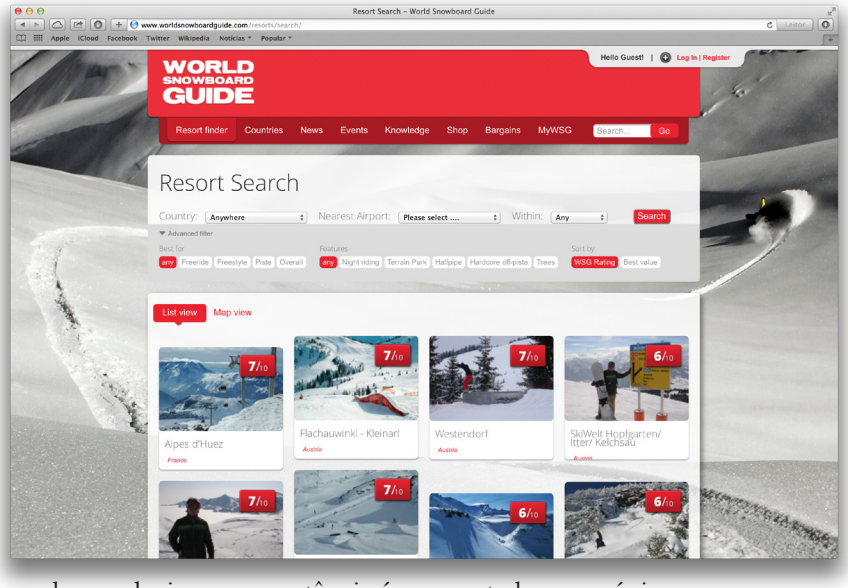


Figura 11. Área de pesquisa de estâncias em World Snowboard Guide.

Quando se seleciona uma estância é apresentada uma página com uma descrição, fotos e vários detalhes. Existe uma área que identifica em percentagem a qualidade da estância para praticar diferentes estilos de *snowboard*, como o *freeride* ou o *freestyle*. Nos detalhes são apresentados dados sobre a montanha como a sua altura e o número de elevadores existentes. É apresentado também o número de pistas existentes e uma percentagem que as divide pelo seu nível de dificuldade. O preço também é apresentado para um dia ou para uma semana e se se trata de época alta ou baixa.

APLICAÇÕES MÓVEIS RELACIONADAS

Foursquare de *Foursquare Labs, Inc.*, 2009

O Foursquare é um dos serviços mais populares de georreferenciação existente para plataformas móveis. No entanto, de todas as plataformas aqui apresentadas é a única que não é destinada ao desporto de ação. Consiste numa aplicação que permite partilhar e guardar os locais que são visitados pelo utilizador e oferece ao utilizador recomendações de locais adequados e que os seus amigos ou outras pessoas tenham visitado. O Foursquare tem uma forte componente social onde cada utilizador pode partilhar imagens de locais que visitou, classificá-los e escrever críticas para que os próximos visitantes possam ter mais informação sobre determinado local. Foi fundado por Dennis Crowley e Naveen Selvadurai que se conheceram em 2007 e lançado em março de 2009 ("About Foursquare", 2014).

A aplicação assenta no conceito de *check-in* que consiste no primeiro passo que as pessoas tomam quando chegam a um aeroporto ou unidade hoteleira e torna este conceito na ação padrão quando o utilizador visita qualquer local. Quando um utilizador faz *check-in* pode adicionar uma imagem ou crítica ao local onde se encontra e os seus amigos no Foursquare podem ver o último local onde esteve e a sua atividade recente na aplicação. Existe também um mapa que permite ver a localização de todos os amigos no mundo.

A plataforma permite descobrir novos locais próximos do utilizador, usando a sua localização atual, e este pode guardar numa lista os locais que pretende visitar no futuro.

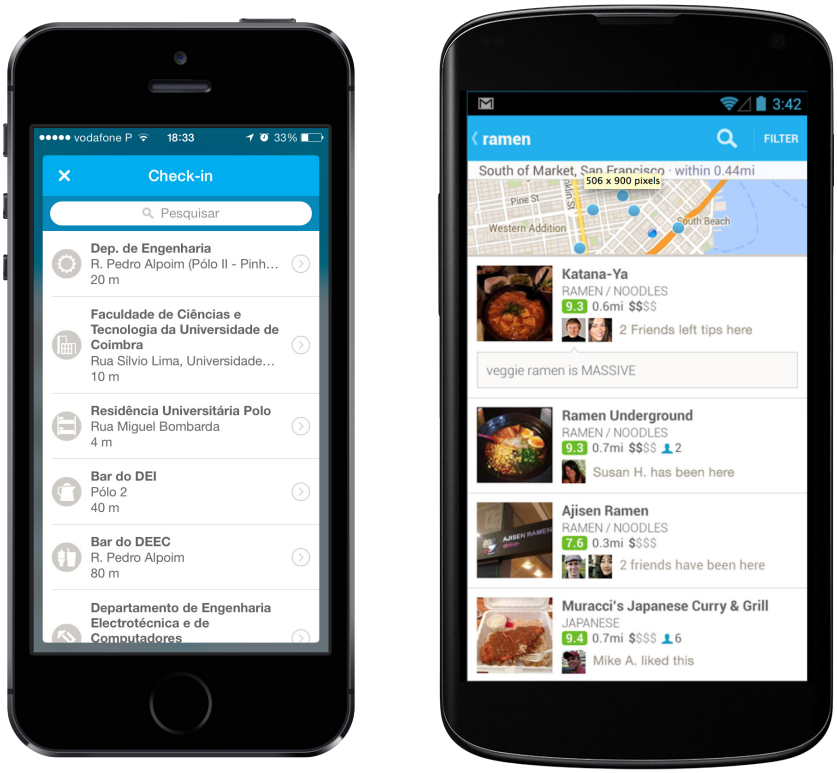


Figura 12. Foursquare em iOS (esquerda) e Android (direita).

Glassy Pro de *Gradient Technologies*, 2012

A aplicação móvel Glassy Pro, disponível para as plataformas iOS e Android, consiste na extensão de uma rede social para amantes do *surf*. Permite aos iniciantes entrarem na comunidade do desporto e conhecerem mais sobre ele. Além disso, possibilita uma interação entre os profissionais, com o objetivo de se ajudarem mutuamente e terem mais diversão (SiliconKarne, 2013).

Glassy Pro assenta na possibilidade de encontrar os melhores locais para praticar *surf* numa determinada altura e partilhar esses dados com os outros utilizadores da plataforma. A aplicação faz uso de geolocalização e permite aos utilizadores adicionar fotos dum determinado local, previsões meteorológicas, melhores alturas para praticar, condições do mar, entre outros. Uma das suas funções principais permite a gravação de uma sessão de *surf* com detalhes como o tempo da sessão total, a altura da maior onda e a direção do vento.

Conta com mais de 4.000 locais em todo o mundo e cada local tem análises feitas por parte dos utilizadores, dando oportunidade a cada utilizador de seguir locais favoritos na plataforma e receber notificações quando as condições esperadas para determinado local são encontradas, condições essas definidas pelo utilizador para esse local. O conteúdo é gerado maioritariamente pelos utilizadores e a sua comunidade cresce diariamente, sendo a sua interface e facilidade de interação dois dos seus pontos fortes.

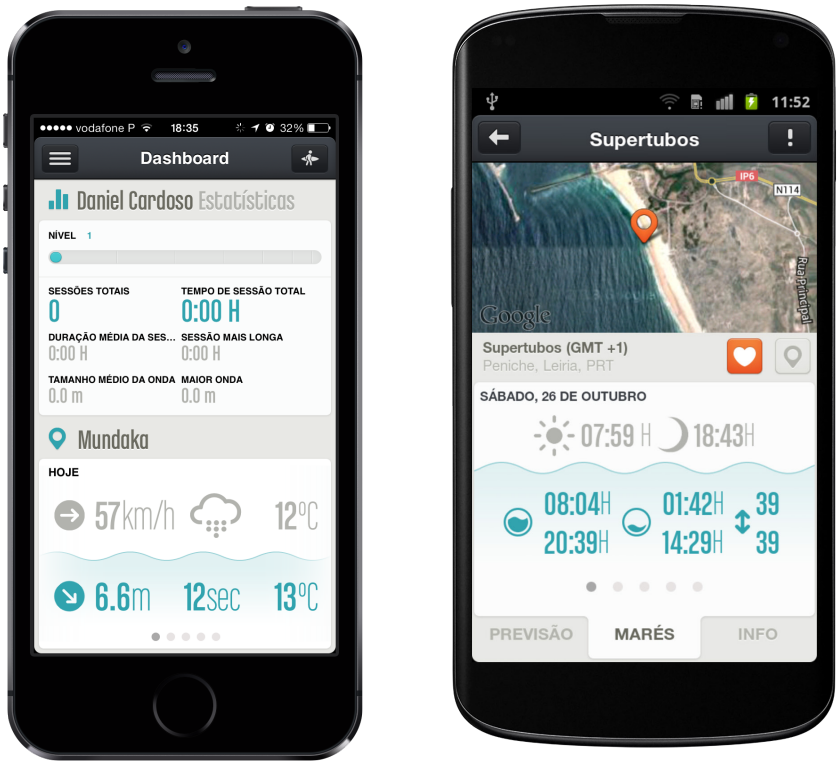


Figura 13. Estatísticas de utilizador em iOS (esquerda) e informação de local em Android (direita) na aplicação Glassy Pro.

goFlow de *GoFlow*, 2013

Bastante semelhante à aplicação Glassy Pro, a aplicação móvel goFlow está apenas disponível para iOS.

A aplicação goFlow existe apenas para dispositivos móveis e permite relacionar os atletas e praticantes dentro da comunidade global do *surf*. Permite encontrar locais próximos para praticar *surf* e partilhar informações sobre um determinado local usando três parâmetros: a altura das ondas, a velocidade do vento e a quantidade de surfistas no local.

Existe um *feed* de notícias que permite ver fotos partilhadas por outras pessoas que podem ser filtradas pela lista de amigos ou seguidores do utilizador na plataforma. Na aplicação, além de se poderem encontrar os locais para praticar *surf* mais próximos do utilizador, também é possível seguir um determinado local, adicionando-o à lista de favoritos. Qualquer utilizador registado na aplicação pode partilhar e combinar sessões com os seus amigos. Tal como na aplicação Glassy Pro, pode-se também destacar a sua interface e organização como pontos.

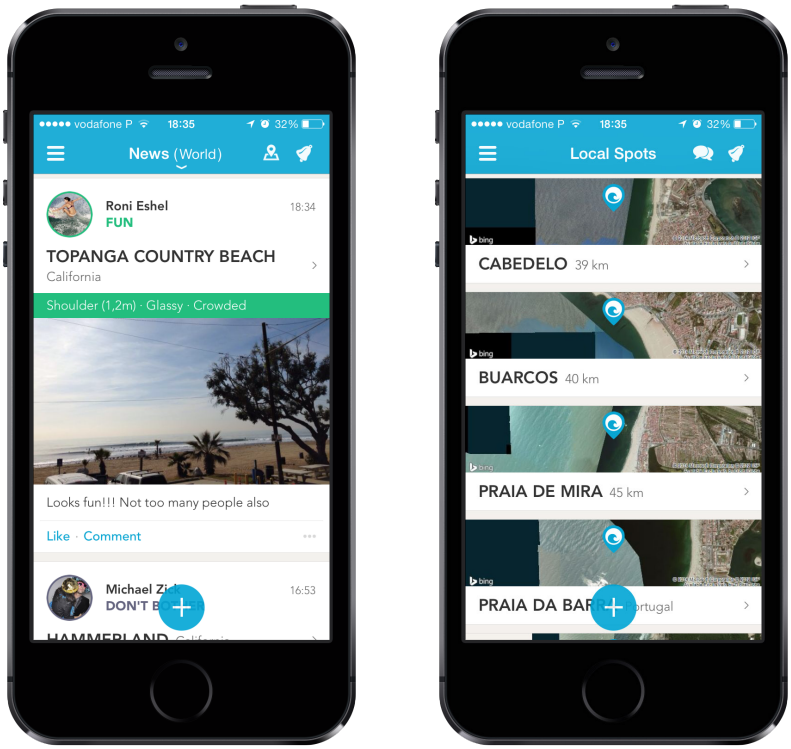


Figura 14. *Feed* de notícias (esquerda) e locais próximos (direita) em goFlow.

MugoSurf de *Mugo d.o.o.*, 2013

Em todo semelhante às aplicações dedicadas ao surf apresentadas anteriormente, a aplicação MugoSurf disponível para iOS é uma comunidade baseada na *web* também com o objetivo de juntar os amantes do *surf* numa só plataforma.

A aplicação foi desenhada para permitir aos surfistas em todo o mundo a partilha de fotos e a consulta de novas atualizações e condições meteorológicas em locais para o *surf*. Uma das funcionalidades que diferencia esta plataforma das anteriores é a existência de uma votação que permite moderar e melhorar os resultados que são apresentados aos utilizadores quando estes procuram pelas condições que pretendem nos locais existentes.

Existem na plataforma mais de 2.000 locais para a prática do *surf* cada um com os seus detalhes visíveis e as previsões meteorológicas dos próximos dias disponíveis. O utilizador pode observar as previsões por horas ao deslocar o dedo sobre o gráfico da previsão. Além de se poder partilhar fotos do local, também é possível partilhar locais interessantes existentes por perto ou outros serviços, como restaurantes ou alojamento.

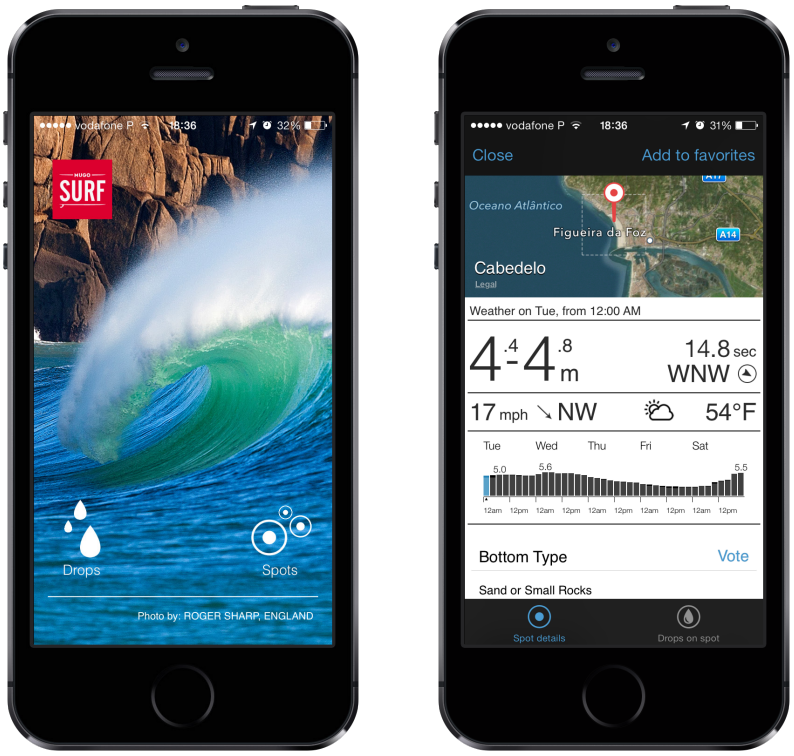


Figura 15. Ecrã inicial (esquerda) e vista de local (direita) na aplicação MugoSurf.

Skate Spots: Skateboarding Spot & Trick Tracker de Sean Herman, 2014

A aplicação Skate Spots está disponível apenas para iOS e como o próprio nome indica, é dedicada apenas para o *skate*.

Ao contrários das aplicações anteriormente introduzidas, esta aplicação não tem uma componente social e consiste apenas numa plataforma móvel que permite armazenar os locais favoritos do utilizador para praticar *skate* e consultá-los numa lista ou num mapa. Ao adicionar os locais o utilizador pode especificar as caraterísticas do local e quais as manobras que costuma fazer. Ao voltar a visitar um local, o utilizador poderá observar quais as manobras que já lá fez e adicionar mais.

Os pontos fortes desta aplicação são sem dúvida a sua interface e a facilidade de uso.

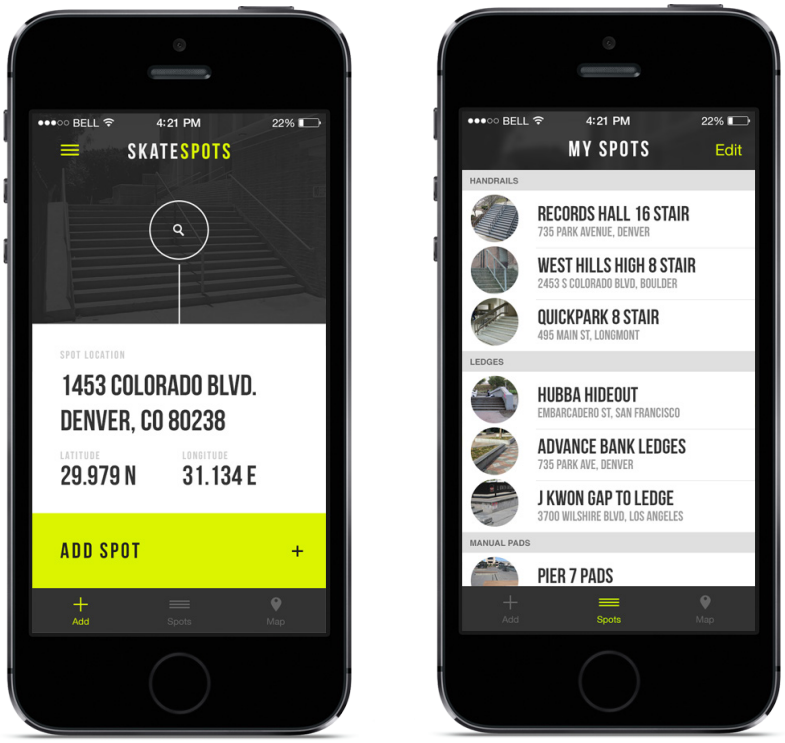


Figura 16. Adicionar local (esquerda) e ver lista de favoritos (direita) na aplicação Skate Spots.

Spot Digger de *Spot Digger Ltd*, 2013

A aplicação Spot Digger é uma aplicação móvel dedicada a dois desportos de ação: o *skate* e o *snowboard*. Está disponível nas plataformas móveis Android e iOS.

Nesta aplicação temos novamente o conceito de *check-in* que permite saber em tempo real onde se encontram os utilizadores que seguimos na plataforma e isto motiva praticantes de cada um dos desportos suportados a encontrarem-se nos mesmos locais.

Semelhante às aplicações anteriormente expostas, nesta podemos também procurar os locais mais próximos em função da localização atual e podemos adicionar locais a uma lista de favoritos. É possível também criar novos locais na plataforma que ainda não existam e assim contribuir para o crescimento da sua base de dados. As funcionalidades da aplicação são as mesmas para os dois desportos suportados.

A aplicação é de fácil uso mas a sua interface pode ser bastante melhorada, não sendo clara a informação presente nalgumas das secções existentes, como por exemplo o *feed* de notícias. O *esqueumorfismo* também cria uma barreira visível entre o interface da aplicação e o interface dos sistemas operativos subjacentes.

Figura 17. Vista de local (esquerda) em iOS e ecrã de escolha de desporto(direita) em Android na aplicação Spot Digger.

Spots de Nick Sorge, 2013

Esta aplicação de Nick Sorge é dedicada ao *skate* e está presente apenas na plataforma iOS. Destaca-se das anteriores pelo seu menu inicial, uma vez que permite fazer uma pesquisa pelo tipo de local onde o utilizador pretende praticar *skate*. Tem também uma secção dedicada às lojas existentes. O utilizador registado na plataforma pode inserir novos locais ou editar os já existentes. Uma funcionalidade bastante interessante é a pesquisa de locais pelo tipo de obstáculos que se pretende encontrar. A aplicação está bem desenhada e é bastante intuitiva, existindo apenas alguns aspetos negativos na apresentação da informação dos locais.

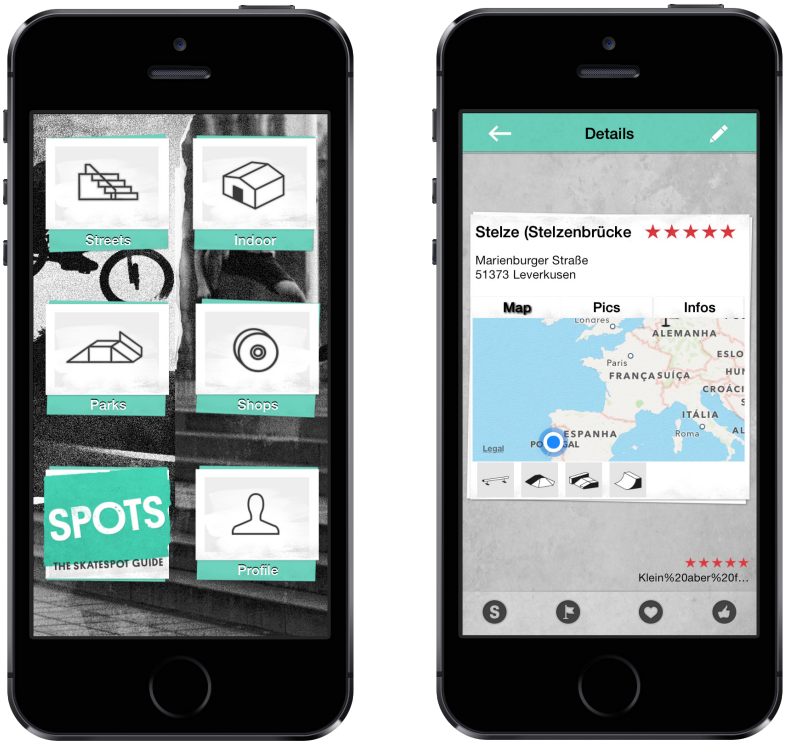


Figura 18. Ecrã inicial de seleção e ecrã de local em Spots.

WeRide de *WeRide Skate LLC*, 2013

De entre as aplicações aqui apresentadas, esta é a melhor entre as que se dedicam ao *skate* e está apenas disponível para iOS. As suas funcionalidades são interessantes e bem desenvolvidas e está bastante bem desenhada. Tal como as restantes permite pesquisar por locais e adicionar novos locais e os resultados são devolvidos em duas vistas, mapas ou lista, entre as quais se pode alternar facilmente. Quando se seleciona um local é possível ver fotos desse local juntamente com a sua classificação, podendo ainda o utilizador adicionar o local aos favoritos, inserir uma foto ou comentário, ou ainda partilhar esse local nas redes sociais. Adicionar o local é um processo bastante simples nesta aplicação, sendo apenas necessário arrastar o mapa para a localização pretendida e selecionar detalhes que identifiquem esse local de uma lista pré-definida.

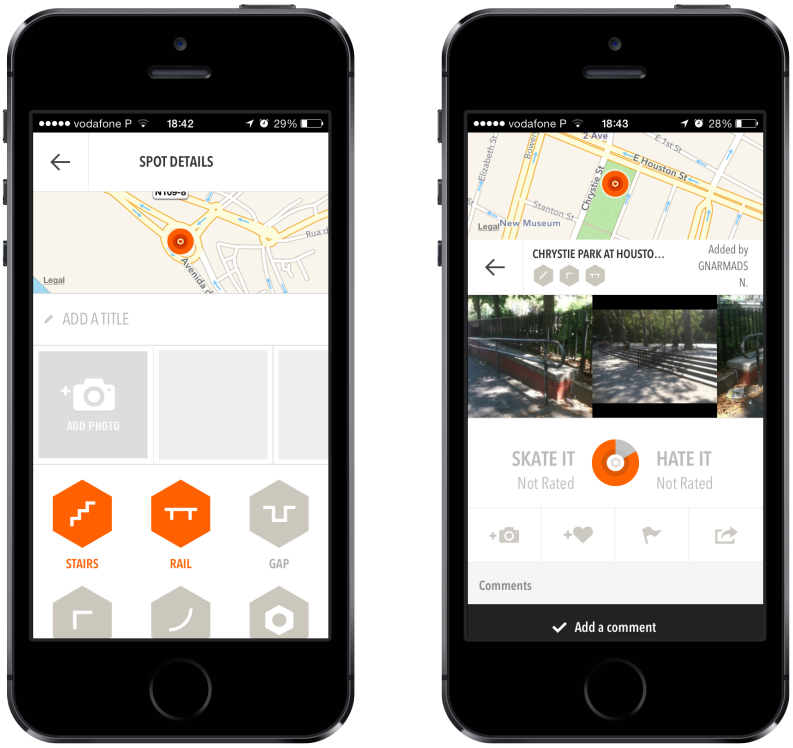


Figura 19. Ecrã de adicionar novo local (esquerda) e vista de local (direita) na aplicação WeRide.

TABELA COMPARATIVA DAS PLATAFORMAS MÓVEIS RELACIONADAS

Na tabela seguinte são comparadas as aplicações móveis previamente analisadas segundo alguns dos critérios que serão a base da aplicação que se pretende criar durante esta dissertação.

Pretende-se assim criar uma plataforma dedicada a desportos de ação que suporte três desportos, nomeadamente o *surf*, o skate, e o *snowboard*. A aplicação terá uma componente social e o conteúdo da mesma será gerado pelos utilizadores. Uma das principais funcionalidades que existirá na aplicação é fazer *check-in* de modo a permitir que os utilizadores saibam da presença de outros utilizadores num determinado local em tempo real.

Como é possível observar na tabela seguinte, a aplicação Spot Digger é a mais semelhante com a plataforma que se pretende desenvolver, cumprindo todos os requisitos. No entanto, o design da aplicação fica um pouco aquém das restantes plataformas analisadas, usando ainda elementos da interface com *esqueumorfismo*, não estando de acordo com os padrões de design atuais. Nesta aplicação apenas estão disponíveis dois desportos, *surf* e *snowboard*, sendo que a diferença visual entre ambos é bastante reduzida. Nesta aplicação também não é possível partilhar conteúdo multimédia quando é efetuado um *check-in*. A funcionalidade de *login* não está funcional, impedindo assim uma avaliação da maioria das funcionalidades da aplicação.

	Desporto de ação	**Múltiplos Desportos**	**Múltiplas Plataformas**	**Social**	**Conteúdo gerado pelo utilizador**	***Check-in***
Foursquare	–	–	✓	✓	✓	✓
Glassy Pro	✓	–	✓	✓	✓	–
goFlow	✓	–	–	✓	✓	–
MugoSurf	✓	–	–	✓	✓	–
Skate Spots	✓	–	–	–	✓	–
Spot Digger	✓	✓	✓	✓	✓	✓
Spots	✓	–	–	–	✓	–
WeRide	✓	–	–	✓	✓	–

4. OBJETIVOS E METODOLOGIAS

OBJETIVOS

– Idealizar e desenvolver conceito para aplicação móvel;

– Estudar o contexto em que o tema da dissertação se insere;

– Efetuar recolha e análise de plataformas relacionadas, ferramentas de interesse para o desenvolvimento da aplicação e explorar funcionalidades que diferenciem a aplicação das que já existem;

– Escolher uma plataforma de desenvolvimento de aplicações multiplataforma segundo requisitos previamente definidos;

– Realizar um estudo sobre usabilidade;

– Desenvolver modelos de interação com a aplicação;

– Criar uma identidade gráfica para a aplicação a desenvolver;

– Implementar um protótipo funcional;

– Avaliar usabilidade de funcionalidades em versões mais estáveis do protótipo;

– Corrigir eventuais erros de interface e de usabilidade.

METODOLOGIAS

Durante o desenvolvimento deste projeto a componente de investigação será a base da componente prática, facilitando e justificando a tomada de decisões na criação da aplicação móvel.

Inicialmente será feito um levantamento das necessidades do público alvo da aplicação e serão estudadas soluções para serem incluídas na prototipagem da aplicação. Após a conclusão do estado da arte será necessário definir como a aplicação a desenvolver se irá destacar das já existentes e o que irá introduzir de inovador. Isto será feito através de uma análise de funcionalidades em falta nas aplicações existentes e de como se poderá melhorar as que já existem. O desenho das novas funcionalidades dependerá de um estudo efetuado sobre usabilidade e sobre interações possíveis nas plataformas móveis.

Numa segunda fase, a qual irá culminar nas interfaces finais da aplicação, serão desenhados os vários locais de interação da plataforma seguindo uma estrutura de navegação previamente definida. A estrutura de navegação consistirá na visualização do resultado de diferentes ações feitas pelo utilizador na aplicação. O desenho da aplicação terá como base as guias de desenvolvimento de aplicações móveis específicas de cada sistema operativo.

Após o desenho da aplicação serão analisadas as plataformas que permitirão criar uma aplicação móvel multiplataforma e, após a escolha da plataforma, será dado início à implementação do protótipo programaticamente.
A aplicação será testada durante o seu desenvolvimento e, numa fase já mais avançada do protótipo, serão feitos testes de usabilidade por um conjunto de utilizadores escolhidos de modo a se detetar falhas de usabilidade e erros ao nível da implementação. Os eventuais erros e falhas de usabilidade encontrados durante os testes de utilizador serão corrigidos no protótipo.

5. PLANO DE TRABALHO E IMPLICAÇÕES

O plano de trabalho identifica as principais tarefas que foram necessárias para a conclusão do projeto e apresenta os períodos de tempo atribuídos a cada uma, durante os quais estas foram concluídas. De seguida serão enunciadas as tarefas e será descrito brevemente o trabalho desenvolvido em cada uma.

1. Contexto da dissertação
2. Estado da Arte
3. Proposta de design
4. Design detalhado
5. Implementação do protótipo
6. Avaliação de usabilidade
7. Escrita da dissertação

1. Contexto da dissertação
Esta tarefa consistiu na escrita de um texto que introduzisse o tema da dissertação ao leitor, explicando como surgiram os três desportos abordados e como estes se relacionam entre si. No contexto introduziu-se o desporto de ação em geral e, de seguida, foram abordados cada um dos três desportos individualmente.

2. Estado da Arte
Nesta tarefa foram recolhidas as informações e projetos relevantes que de certa forma se relacionam com o tema desta dissertação. Foi feita uma introdução aos dispositivos móveis e às tecnologias existentes nos mesmos e foi abordada a evolução das interfaces destes. A segunda parte consistiu na análise dos projetos relacionados e das ferramentas existentes para a produção de aplicações móveis que façam uso de georreferenciação.

3. Proposta de design
Após a conclusão do Estado da Arte foi dado início à proposta inicial de design, na qual foi produzido o modelo de interação da aplicação. A partir deste modelo foram desenhados os primeiros esboços do protótipo.

4. Design detalhado
Nesta tarefa foi definida a identidade gráfica da aplicação e foram desenhados todos os ecrãs de alta fidelidade de acordo com essa identidade. Nesta fase foi também definida a arquitetura de informação da aplicação e criado um modelo da base de dados, a qual permitiu armazenar e gerir a informação necessária através de uma API posteriormente desenvolvida.

5. Implementação do protótipo
Durante o desenho da aplicação final foi iniciada a fase de implementação. Esta fase teve duas etapas: a aprendizagem das ferramentas a usar e da linguagem de programação e a implementação do protótipo, o que incluiu também a implementação da plataforma *web* que deu suporte ao protótipo.

6. Avaliação de usabilidade
Numa fase mais avançada do protótipo foi realizada uma avaliação da usabilidade das funcionalidades já implementadas diante de utilizadores que

representam o público alvo. Esta tarefa decorreu após a implementação do protótipo. Após a análise dos resultados foram efetuadas as devidas melhorias no protótipo.

7. Escrita da dissertação
Esta tarefa decorreu ao longo de todo o projeto e em paralelo com as outras tarefas. Consistiu em desenvolver este documento escrito. Esta tarefa dependeu assim das restantes, uma vez que o documento inclui a descrição de todas as tarefas executadas durante o projeto e todo o processo de desenvolvimento.

Nos gráficos seguintes será apresentado o planeamento previsto inicialmente, o qual foi delineado ate à data da entrega do relatório intermédio, e o plano de trabalhos que foi realizado.

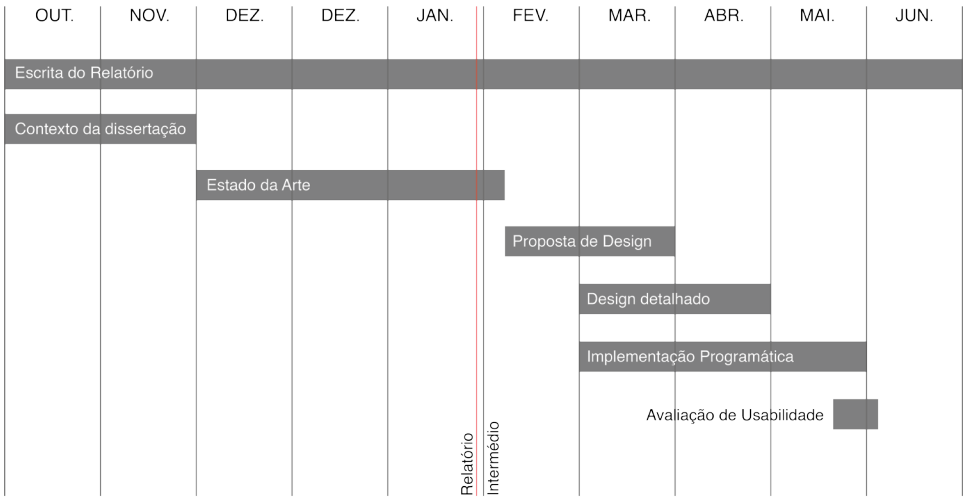


Figura 20. Plano de trabalhos previsto.

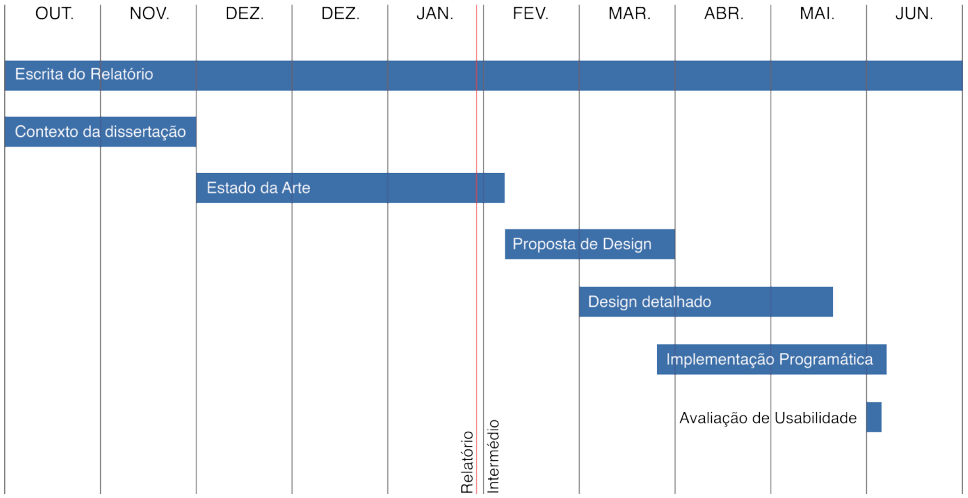


Figura 21. Plano de trabalhos realizado.

Após a entrega do relatório intermédio e posterior defesa intermédia, começou-se a criação da proposta de design, a qual correu como planeado. A implementação começou mais tarde devido a não se ter chegado a um design detalhado satisfatório no início deste processo. Este atraso levou a que a implementação ocupasse ainda algum tempo do mês de junho e que o tempo dedicado para a tarefa mesmo assim fosse menor que o inicialmente previsto, no entanto, todos as funcionalidades a que foram dada prioridade após a definição da proposta de design foram concluídas. Os testes de usabilidade decorreram mais rapidamente do que o inicialmente planeado, tendo ainda existido tempo para efetuar as alterações necessárias no protótipo após a conclusão dos mesmos.

6. PROTOTIPAGEM DA APLICAÇÃO

NOME *SPOTCLAIMER*

O nome atribuído à aplicação surge através da justaposição das palavras "spot" e "claimer". A palavra inglesa *spot* é traduzida para local e a palavra *claimer* refere-se a alguém que fez *claim*, sendo a tradução de *claim* o verbo reivindicar. Assim, a junção das duas palavras refere alguém que reivindica um local, ou seja, que reclama a sua posse.

Desde cedo se pretendeu usar o verbo *claim* no nome da aplicação, sendo que outras opções para o nome da aplicação passaram por "SpotClaim" e "Claim It". A escolha deste verbo deveu-se a este estar relacionado com o desporto de ação, nomeadamente com o *surf.* No *surf* um *claim* representa um gesto realizado pelos atletas, normalmente no fim de surfarem uma onda de elevado grau de dificuldade com sucesso ou de conseguirem pontuar numa fase importante de um campeonato. Existem várias formas de gestualizar o *claim,* sendo que a maioria consiste num gesto de festejo.

Figura 22. *Claim* de Andy Irons em Teahupo'o, Tahiti. [1]

A escolha deste nome influenciou o conceito da aplicação de duas formas: o ato de fazer *check-in* num local passou a ser denominado de *claim;* o atleta/praticante com mais *claims* num local passou a ser o "Spot Claimer" desse local.

[1] Imagem disponível no *link:* http://surfbang.com/videos/2012/08/teahupoo-claims.html

PERSONAS & CENÁRIOS

A criação e posterior análise de cenários de uso da aplicação ajudam na definição de um modelo de interação com a aplicação. As personas são modelos de potenciais utilizadores da aplicação e permitem representar uma pequena amostra do público alvo. Vários fatores como a experiência tecnológica e o conhecimento de aplicações do domínio irão influenciar o uso da aplicação. Os cenários de uso representam possíveis utilizações da aplicação idealizada no mundo real e ajudam a determinar as necessidades que a aplicação deve satisfazer.

Nome: Kelly
Idade: 41 anos
Tipo de utilizador: Surfista profissional
Conhecimento do domínio (1-5): 5 – tem bastante conhecimento do meio do desporto de ação, sendo profissional há 25 anos.
Experiência Tecnológica (1-5): 3 – tem algum à vontade com novas tecnologias.
Experiência Aplicações do Domínio (1-5): 1 – usa apenas redes sociais para partilhar a sua atividade.

Cenário:
O Kelly como surfista profissional viaja muito e produz bastante material multimédia que pretende partilhar com a comunidade de fãs. Durante uma pausa no World Championship Tour decide viajar para o Hawai para passar algum tempo com a família e descobrir novos locais para praticar surf. Possui um smartphone com acesso à Internet mas a sua aplicação de mapas não lhe apresenta os locais mais próximos em que se pode surfar. Após descobrir uma praia com ondas que rebentam na praia, Kelly decide pegar na prancha e ir surfar, pedindo à sua mulher para filmar algumas das suas manobras. No fim de surfar, o Kelly pretende partilhar alguns dos vídeos que a sua mulher filmou nas redes sociais que usa, no entanto as redes sociais não lhe dão a possibilidade de associar os vídeos ao local onde se encontra e não lhe permitem adicionar o local.

Funcionalidades necessárias:
– apresentar locais mais próximos para a prática do desporto;
– partilha de conteúdo multimédia associado a um local;
– possibilidade de adicionar local.

Nome: Ryan
Idade: 23 anos
Tipo de utilizador: Skater profissional
Conhecimento do domínio (1-5): 4 – conhece bastante bem o mundo do skate, é profissional há cerca de 7 anos.
Experiência Tecnológica (1-5): 4 – está à vontade com o uso de *smartphones*.
Experiência Aplicações do Domínio (1-5): 4 – usa várias aplicações para partilhar a sua localização e os seus feitos.

Cenário:
O Ryan encontra-se num evento do campeonato X Games onde estão presentes atletas de várias modalidades, incluindo o skate. Sendo um utilizador das novas tecnologias, o Ryan tem no seu smartphone a maioria das aplicações relacionadas com a prática. No campeonato onde se encontra estão presentes os melhores skaters do mundo e o Ryan quer poder encontrar-se com os que estão presentes no evento. Para isso o Ryan tem que procurar nas redes sociais e nas aplicações que usa para tentar saber se algum dos atletas que conhece está no evento. Embora já seja profissional, o Ryan gosta bastante de mostrar as suas novas manobras aos seus fãs e gosta que estes o sigam ao vivo. De modo a contar com a presença dos seus fãs no evento o Ryan partilha a sua localização nas redes sociais com uma foto de uma manobra sua efetuada no momento.

Funcionalidades necessárias:
– pesquisa de utilizadores registados na aplicação;
– partilha da localização com conteúdo associado.

Nome: Manuel
Idade: 26 anos
Tipo de utilizador: Praticante de surf
Conhecimento do domínio (1-5): 3 – conhece bastante pessoas que praticam surf mas não tem contato com os profissionais.
Experiência Tecnológica (1-5): 4 – está à vontade com as novas tecnologias.
Experiência Aplicações do Domínio (1-5): 2 – usa uma aplicação que contém os locais para praticar surf em Portugal.

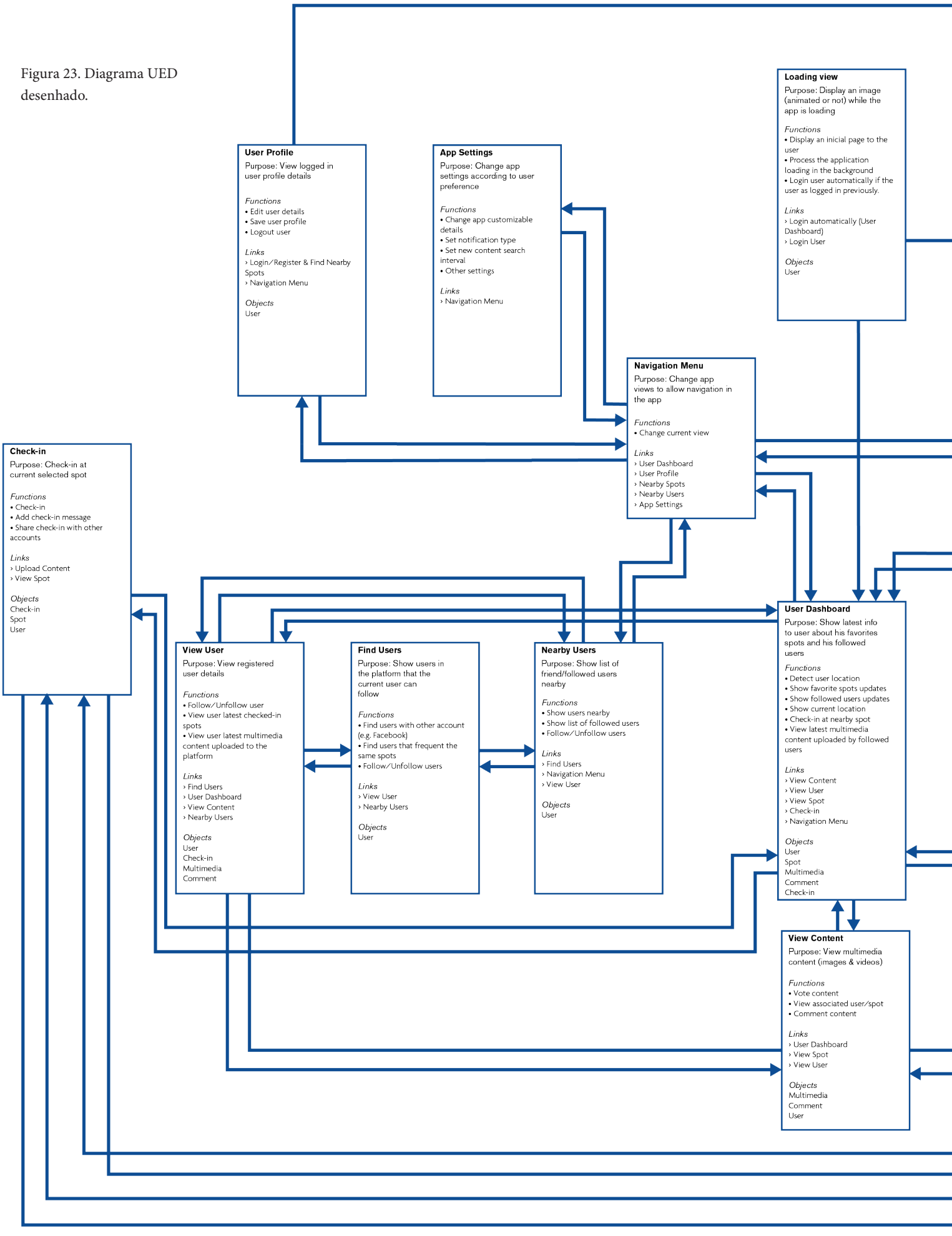
Cenário:
O Manuel, praticante regular de surf, vai em viagem ao longo da costa portuguesa e faz algumas paragens durante a viagem para conhecer novas ondas. Ao abrir a aplicação e ao querer partilhar com os seus amigos vídeos de algumas ondas que apanhou numa praia que visitou, o Manuel observa que essa praia ainda não está disponível na aplicação que usa. Uma vez que a aplicação não dá a possibilidade ao Manuel de adicionar um novo local, o Manuel tem que partilhar os seus vídeos nas redes sociais.

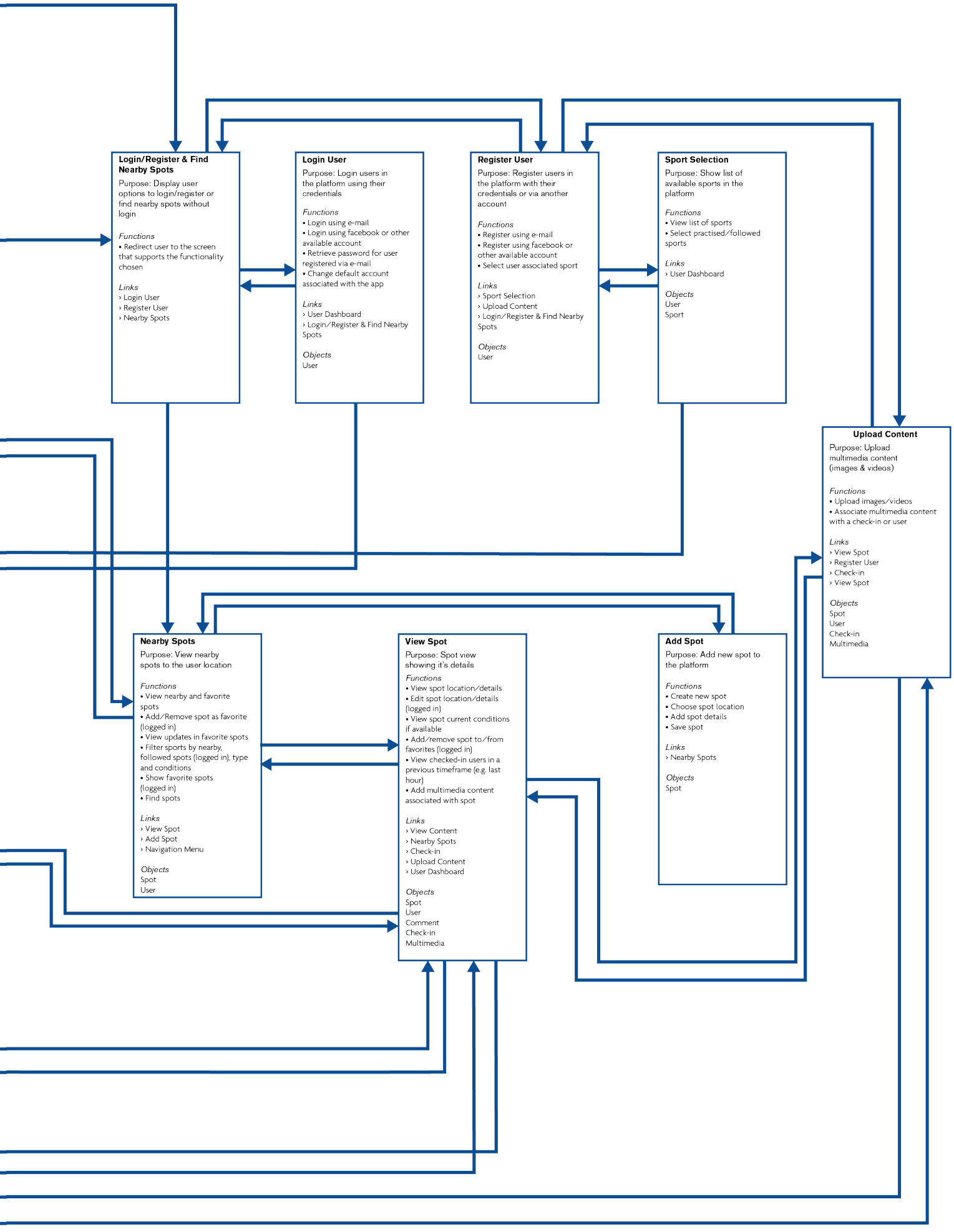
Funcionalidades necessárias:
– possibilidade de adicionar novos locais;
– relacionamentos entre utilizadores na aplicação.

DIAGRAMA UED

Após a realização de um conjunto de cenários e personas recorreu-se a um diagrama UED (*User Environment Design*) para representar as áreas de interação que devem existir na aplicação. Este diagrama permitiu fazer uma divisão da aplicação por áreas de foco, onde cada uma representa uma funcionalidade chave da aplicação. Cada área contém as funções que permitem desempenhar as atividades da aplicação nesse ecrã, bem como os objetos sobre os quais essas funções são realizadas. As ligações entre as áreas representam a navegação que pode ser feita dentro da aplicação. Num diagrama UED os detalhes da interface de utilizador e detalhes de implementação são ignorados. No entanto, o diagrama permite facilmente partir para o desenho dos ecrãs da aplicação, uma vez que ao desenhar a aplicação é possível saber quais as funções que devem estar disponíveis em cada ecrã.
O diagrama UED apresentado nas duas páginas seguintes representa as áreas de interação e as ligações que foram pensadas para a aplicação. No diagrama seguinte, cada área tem quatro parâmetros, os quais têm as seguintes funções:
– *purpose* - descrever sumariamente o que é efetuado nesta área;
– *functions* - funções disponíveis para realizar a tarefa associada a esta área;
– *links* - ligações a outras áreas;
– *objects* - objetos com os quais é realizada a tarefa.

Figura 23. Diagrama UED desenhado.




Login/Register & Find Nearby Spots
Purpose: Display user options to login/register or find nearby spots without login
Functions
• Redirect user to the screen that supports the functionality chosen
Links
› Login User
› Register User
› Nearby Spots
Login User
Purpose: Login users in the platform using their credentials
Functions
• Login using e-mail
• Login using facebook or other available account
• Retrieve password for user registered via e-mail
• Change default account associated with the app
Links
› User Dashboard
› Login/Register & Find Nearby Spots
Objects
User
Register User
Purpose: Register users in the platform with their credentials or via another account
Functions
• Register using e-mail
• Register using facebook or other available account
• Select user associated sport
Links
› Sport Selection
› Upload Content
› Login/Register & Find Nearby Spots
Objects
User
Sport Selection
Purpose: Show list of available sports in the platform
Functions
• View list of sports
• Select practised/followed sports
Links
› User Dashboard
Objects
User
Sport
Upload Content
Purpose: Upload multimedia content (images & videos)
Functions
• Upload images/videos
• Associate multimedia content with a check-in or user
Links
› View Spot
› Register User
› Check-in
› View Spot
Objects
Spot
User
Check-in
Multimedia
Nearby Spots
Purpose: View nearby spots to the user location
Functions
• View nearby and favorite spots
• Add/Remove spot as favorite (logged in)
• View updates in favorite spots
• Filter sports by nearby, followed spots (logged in), type and conditions
• Show favorite spots (logged in)
• Find spots
Links
› View Spot
› Add Spot
› Navigation Menu
Objects
Spot
User
View Spot
Purpose: Spot view showing it's details
Functions
• View spot location/details
• Edit spot location/details (logged in)
• View spot current conditions if available
• Add/remove spot to/from favorites (logged in)
• View checked-in users in a previous timeframe (e.g. last hour)
• Add multimedia content associated with spot
Links
› View Content
› Nearby Spots
› Check-in
› Upload Content
› User Dashboard
Objects
Spot
User
Comment
Check-in
Multimedia
Add Spot
Purpose: Add new spot to the platform
Functions
• Create new spot
• Choose spot location
• Add spot details
• Save spot
Links
› Nearby Spots
Objects
Spot

De seguida serão explicadas as considerações tidas na criação do diagrama UED. Será abordada cada área de interação e serão explicadas as funcionalidades que foram implementadas e as alterações que foram efetuadas ao longo do processo de design e desenvolvimento. Cada área de interação representa um ou vários ecrãs na aplicação, os quais permitem realizar o propósito dessa área, portanto, as áreas de interação serão abordadas como ecrãs nos próximos parágrafos. Nem todas as funcionalidades descritas nas áreas de interação foram implementadas ou desenhadas devido ao tempo limitado para o desenvolvimento da aplicação e porque a aplicação desenvolvida se trata de um protótipo que visa demonstrar um conceito. O conjunto de funcionalidades descritas, no caso de serem implementadas no futuro, permitiria o lançamento da aplicação numa loja de aplicações.

Loading View

A *loading view* trata-se do ecrã apresentado após o carregamento da aplicação. Neste ecrã implementou-se uma animação do logo que é executada paralelamente ao *login* do utilizador na aplicação. O *login* é processado automaticamente caso o utilizador já tenha usado a aplicação com *login* ou *signup*, sendo que neste caso as suas credenciais são guardadas seguramente no *Keychain*. Se o *login* for efetuado com sucesso a aplicação muda para o ecrã de atividade (*User Dashboard*). Caso exista um erro a processar o *login* ou as credenciais não tenham sido guardadas no *Keychain* é apresentado o ecrã que permite ao utilizador escolher entre efetuar *login*, *signup* ou procurar os locais mais próximos da sua localização (caso o utilizador não possua uma conta na aplicação).

Login/Register & Find Nearby Spots

Este é o primeiro ecrã apresentado que necessita da ação do utilizador. Neste ecrã são dadas três opções ao utilizador: efetuar *login*, *signup* ou ir para o ecrã com os *spots* mais próximos. Cada uma destas opções leva o utilizador para um novo ecrã onde as pode depois realizar. A possibilidade de ver os *spots* mais próximos existe para permitir a novos utilizadores da aplicação usufruírem de alguma funcionalidade sem terem que passar pelo processo de registo. Na implementação esta situação não foi considerada, sendo apenas possível visualizar os *spots* mais próximos caso o utilizador tenha efetuado *login* ou *signup* na aplicação.

A partir deste ecrã é possível navegar para os ecrãs *Login User*, *Register User* e *Nearby Spots*.

Login User

Neste ecrã é dada a possibilidade de o utilizador efetuar o *login* na aplicação. Este pode ser feito através do Facebook, caso o utilizador possua conta nesta rede social, ou através de *email*. O *login* por Facebook não foi implementado pois o *login* por *email* exige um maior número de passos e deve ser testado previamente por utilizadores para validar a sua usabilidade. Ambas as opções necessitam que o utilizador tenha efetuado *signup* previamente na aplicação.

Ao efetuar *login* as credenciais do utilizador são guardadas no *Keychain*, caso já existam credenciais guardadas serão substituídas, permitindo assim alterar ou adicionar uma conta de utilizador à aplicação, evitando a repetição dos passos de *login* quando a aplicação é executada novamente. Caso o utilizador se tenha esquecido das suas credenciais de acesso, também é possível recuperá-las através de uma opção disponível neste ecrã, no entanto, esta funcionalidade não foi implementada uma vez que não era crucial para o protótipo.

Ao efetuar *login* com sucesso a aplicação mostra ao utilizador o ecrã *User Dashboard*.

Register User

Semelhante ao ecrã de *login*, neste é possível um utilizador registar-se na aplicação, tendo assim acesso a todas as funcionalidades disponíveis. O registo também pode ser efetuado usando a conta de Facebook do utilizador ou por *email*. Ao efetuar o registo, o utilizador pode selecionar o desporto que pratica profissionalmente, no entanto, optou-se por deixar de fora esta opção na implementação, uma vez que não era importante fazer a distinção entre atletas de diferentes desportos na plataforma e porque a aplicação é também dedicada a praticantes não profissionais e estes podem praticar vários desportos de ação como lazer. Ao ser implementada a opção de escolher o desporto praticado a diferença entre praticantes de diferentes desportos iria ser somente visual e a nível de funcionalidade poderiam existir filtros para pesquisa de utilizadores ou *spots* com o mesmo desporto associado. Ao registar-se é possível também adicionar uma imagem do utilizador através do ecrã *Upload Content*.

Quando o utilizador efetua o registo com sucesso, tal como no *login*, a aplicação apresenta o ecrã *User Dashboard* uma vez que o ecrã *Sport Selection* não foi implementado, senão seria este o ecrã a apresentar de seguida.

Sport Selection

Este ecrã consiste numa lista dos desportos disponíveis na aplicação que podem ser selecionados pelo utilizador de modo a serem seguidos, tendo o utilizador a possibilidade de seguir os desportos que pratica e/ou gosta. Esta lista seria apresentada após o utilizador efetuar registo no ecrã *Register User*. Este ecrã não foi implementado uma vez que a aplicação, na fase de protótipo, só suporta três desportos e cada utilizador segue todos os desportos disponíveis.

User Dashboard

Este é o primeiro ecrã apresentado ao utilizador após este efetuar *login* ou *signup* na aplicação. Neste ecrã é apresentada toda a atividade do utilizador, dos utilizadores que segue e dos *spots* que adicionou aos seus favoritos. A atividade é disposta por ordem cronológica, desde a mais recente até à mais antiga. O conteúdo multimédia associado a cada atividade também é visível neste ecrã. Em cada publicação do registo de atividade o utilizador

pode selecionar o utilizador ou *spot* associado a essa publicação e a aplicação irá mostrar os ecrãs de *View User* e *View Spot*, respetivamente. Neste ecrã também é possível fazer *check-in* num local próximo, sendo necessário verificar a localização do utilizador, no entanto, esta funcionalidade não foi implementada, sendo apenas permitido fazer *check-in* no ecrã *View Spot*. Para aceder a outra área da aplicação o utilizador terá de utilizar o ecrã *Navigation Menu*.

Navigation Menu

O ecrã *Navigation Menu* tem como função permitir a navegação entre diferentes áreas da aplicação. Neste ecrã é apresentada uma lista das diferentes áreas para as quais se pode navegar. Ao selecionar um elemento da lista o ecrã principal é atualizado para apresentar o ecrã selecionado. O *Navigation Menu* aparece em paralelo com ecrã principal, dando assim ao utilizador um contexto do estado atual da aplicação e deixando a possibilidade de voltar ao ecrã atual.

User Profile

Este ecrã pode ser acedido através do *Navigation Menu*. Neste ecrã o utilizador pode ver os seus dados e editá-los, tendo também disponível um registo da sua atividade mais recente. A opção de editar não foi implementada pois não era uma funcionalidade prioritária para o protótipo. A opção de efetuar logout na aplicação também é possível através deste ecrã. Optou-se por se manter a opção de logout aqui e não no ecrã *Navigation Menu* por se tratar de uma opção que será poucas vezes utilizada. Ao efetuar logout as credenciais do utilizador guardadas no *Keychain* serão removidas.

App Settings

Neste ecrã é dada a possibilidade de o utilizador alterar as preferências e definições da aplicação como o tipo de notificações que o utilizador pretende receber, o intervalo de tempo para procurar por novas atualizações, entre outras. Este ecrã não foi implementado pois não era essencial para demonstrar o conceito da aplicação. O utilizador pode aceder a este ecrã através do ecrã *Navigation Menu*.

Nearby Spots

No ecrã *Nearby Spots*, acessível através do *Navigation Menu*, é apresentada ao utilizador uma lista dos *spots* existentes por ordem de proximidade, desde o mais próximo até ao mais distante. É possível adicionar qualquer um dos *spots* da lista aos favoritos do utilizador, mostrar apenas *spots* que sejam favoritos dos utilizadores seguidos, por desporto associado e por favoritos do utilizador. É possível também procurar *spots* por nome. No protótipo desenvolvido apenas são apresentados os *spots* por ordem de proximidade e ao selecionar um deles a aplicação mostra ao utilizador o ecrã *View Spot* do *spot* selecionado. A partir deste ecrã o utilizador também pode ir para o ecrã *Add Spot*.

View Spot

O ecrã *View Spot* é apresentado quando o utilizador seleciona um *spot* na lista de *Nearby Spots* ou numa publicação de atividade. Neste ecrã são apresentados os detalhes do *spot*, incluindo a sua localização e as condições atuais. Pretende-se que estes detalhes possam ser editados mas de momento tanto esta possibilidade de edição como a visualização das condições atuais do *spot* não se encontram implementadas. Neste ecrã é também possível adicionar ou remover o *spot* da lista dos favoritos do utilizador e ver um registo dos últimos utilizadores a fazerem *check-in* no *spot* selecionado. Tal como no ecrã *User Dashboard*, o utilizador pode fazer *check-in* num *spot*, neste caso no *spot* apresentado no ecrã. Além do conteúdo multimédia associado a um *check-in* um utilizador também pode carregar conteúdo multimédia que será apenas associado ao *spot*. Esta última funcionalidade não foi implementada.

Add Spot

O ecrã *Add Spot* tem como principal função adicionar um novo *spot* à aplicação. Neste ecrã é definida a localização do *spot* e os seus detalhes. Após o preenchimento dos detalhes do novo *spot*, este pode ser guardado para ser posteriormente acedido na aplicação. Qualquer utilizador registado pode adicionar novos *spots* à aplicação.

Nearby Users

Este ecrã é semelhante ao *Nearby Spots*, mas neste caso é apresentada uma lista de utilizadores por ordem de proximidade. A localização dos utilizadores é determinada pela localização do último *spot* onde efetuaram *check-in*. Neste ecrã também é possível seguir utilizadores e escolher a opção de apenas mostrar os utilizadores seguidos. Ao selecionar um utilizador da lista, a aplicação apresenta o ecrã *View User*. A ordenação por ordem de proximidade e a filtragem por utilizadores seguidos não foram implementadas. Este ecrã pode ser acedido através do *Navigation Menu*.

Find Users

No ecrã *Find Users* é possível encontrar utilizadores que já se encontrem registados na aplicação. É possível procurar utilizadores que façam parte dos amigos da conta de Facebook do utilizador, caso o utilizador se tenha registado na aplicação através de Facebook ou então dê permissão para que a aplicação aceda aos amigos da rede social. Também são apresentados como sugestões os utilizadores que frequentem os mesmos *spots* que o utilizador ativo. Tal como no ecrã *Nearby Users* é também possível seguir os utilizadores apresentados. Este ecrã não foi implementado porque a funcionalidade de pesquisa de utilizadores não era importante na fase de protótipo, uma vez que no ecrã *Nearby Users* já são listados todos os utilizadores registados.

View User

O ecrã *View User* é bastante semelhante ao ecrã *View Spot*, sendo neste apresentada toda a informação de um utilizador. Neste ecrã também é possível seguir o utilizador e ver o seu registo de atividade, no qual são listados os últimos locais onde se fez *check-in* e o conteúdo multimédia associado.

Check-in

O ecrã *Check-in* tem como principal funcionalidade efetuar o *check-in* num determinado *spot* por parte do utilizador ativo na aplicação. Ao fazer *check-in* pode ser associada uma mensagem e conteúdo multimédia, o qual pode ser inserido através do ecrã *Upload Content*. O *check-in* pode ser partilhado noutras contas, nomeadamente o Facebook, no entanto, esta funcionalidade não foi implementada no protótipo.

Upload Content

Este ecrã tem como principal funcionalidade o carregamento de conteúdo multimédia (imagens e vídeos) na aplicação. Este conteúdo pode ser associado a um utilizador, *check-in*, ou adicionado a um *spot*. O conteúdo é associado a um utilizador quando o é efetuado o *upload* de uma imagem de perfil; no caso de um *check-in* o conteúdo associado pode ser uma imagem ou um vídeo. O *upload* de conteúdo multimédia num *spot* não foi implementado.

View Content

O ecrã *View Content* permite visualizar o conteúdo multimédia num ecrã sem distrações, onde é possível fazer zoom às imagens e reproduzir os vídeos. Neste ecrã é também possível comentar a publicação a que o conteúdo multimédia está associado. Este ecrã não foi implementado uma vez que o conteúdo multimédia suportado no protótipo são apenas imagens e estas podem ser visualizadas no registo de atividade dos utilizadores, *spots* ou do utilizador registado. A funcionalidade de comentários também não foi desenvolvida por não ser crucial na fase de protótipo.

PRIORIDADE DE IMPLEMENTAÇÃO

Após a criação do modelo UED era importante priorizar o desenho e implementação das funcionalidades-chave da aplicação, dando menos importância às funcionalidades que atribuem menos valor à aplicação.
A definição de prioridades permitia que fossem implementadas as funcionalidades chave da aplicação, caso o tempo se tornasse curto para o desenvolvimento da aplicação, uma vez que para a efetuar a implementação também era necessário passar primeiro por um processo de aprendizagem.

As funcionalidades a serem desenvolvidas por ordem de prioridade, da mais prioritária para a menos prioritária, são as seguintes:

1. Login e *signup* de utilizadores.
2. Adicionar novos *spots* à aplicação.
3. Fazer *check-in* num *spot* com conteúdo multimédia associado.
4. Adicionar *spot* aos favoritos.
5. Seguir utilizadores.
6. Ver *spot* e seu registo de atividade.
7. Ver perfil de utilizador e seu registo de atividade.
8. Ver registo da atividade dos *spots* favoritos e utilizadores seguidos na aplicação.

Para implementar cada uma das funcionalidades acima descritas era necessário desenhar e implementar um ou mais ecrãs.

DIAGRAMA ER & MODELO FÍSICO

Para dar suporte à aplicação com uma base de dados e uma API era necessário definir um modelo de dados que incluísse os objetos referidos no diagrama UED. Optou-se por desenhar um diagrama ER (*Entity Relationship Diagram* - Diagrama Entidade Relacionamento), pois permitia definir esses objetos como entidades e especificar as relações entre eles. A partir do diagrama ER foi também possível definir um modelo físico que mais tarde ajudou na criação de uma base de dados. O ER desenvolvido, apresentado na figura seguinte, teve como base a notação *Crow's Foot*.

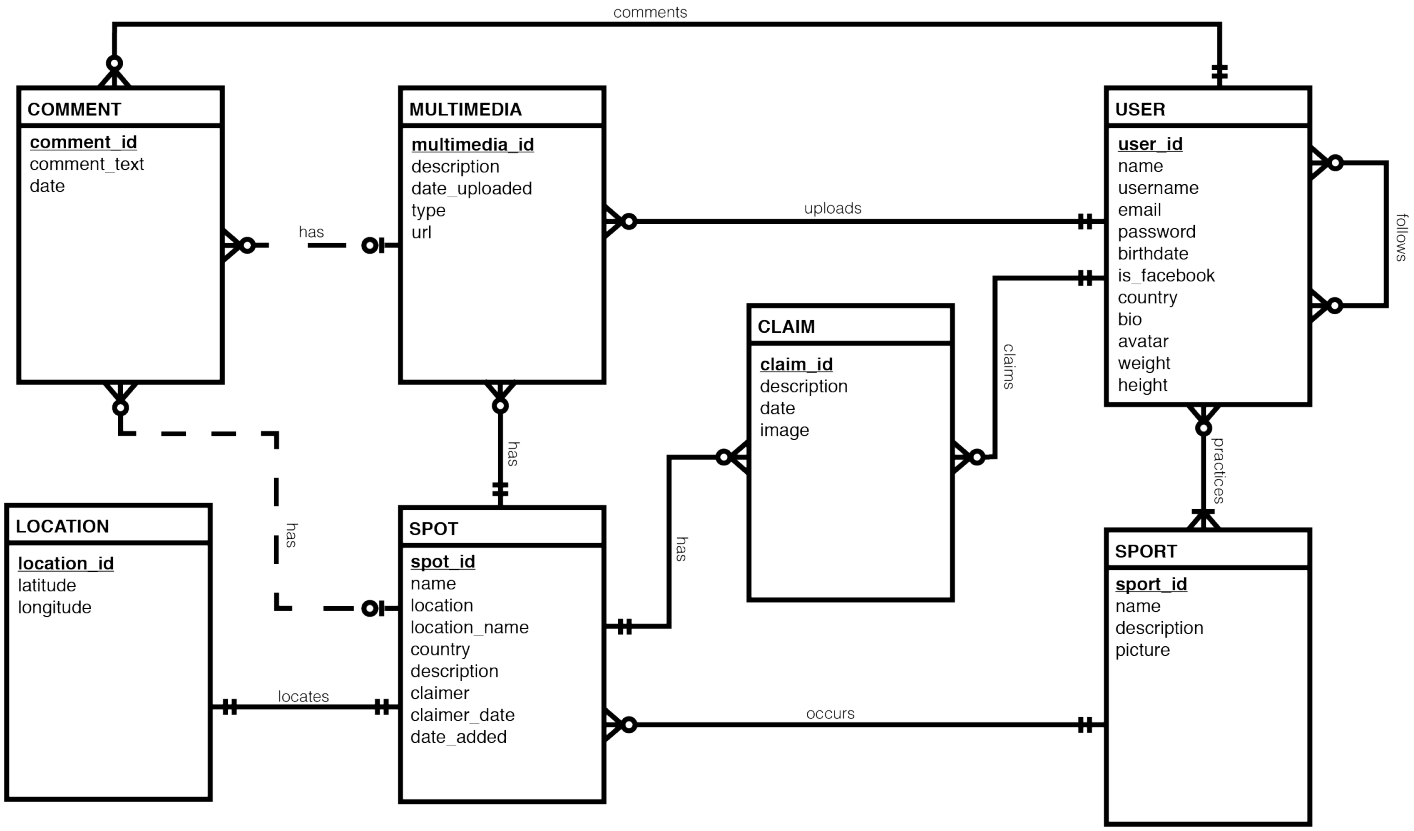


Figura 24. Diagrama ER.

As ligações entre entidades podem ser traduzidas da seguinte forma:

- zero ou um;
- um e um só;
- zero, um ou muitos;
- um ou muitos;

No diagrama cada caixa representa uma entidade. O nome da entidade está visível no topo da caixa e os restantes campos são os atributos da mesma. O primeiro atributo é a chave primária da entidade, sendo assim a chave que identifica cada uma das entidades e dos registos posteriormente criados. Neste diagrama foram deixados de fora o tipo de variáveis de cada atributo, tendo sido posteriormente incluídas no modelo físico.

As relações entre entidades no diagrama são as seguintes, da direita para a esquerda:
– cada utilizador pode seguir zero ou mais utilizadores, sendo que cada utilizador pode ser seguido por zero ou mais utilizadores;
– um utilizador deve seguir um ou mais desportos, cada desporto pode ser seguido por zero ou mais utilizadores;
– um utilizador pode efetuar zero ou mais comentários, cada comentário está associado a um e apenas um utilizador;
– um utilizador pode carregar zero ou mais conteúdos multimédia, cada conteúdo multimédia está associado a um e um só utilizador;
– um utilizador pode efetuar zero ou mais *claims*, sendo que cada *claim* está associado a um e um só utilizador;
– um desporto pode ter vários *spots* associados, cada *spot* está associado a um e um só desporto;
– um *claim* está associado a um e um só *spot*, cada *spot* pode ter zero ou mais *claims* associados;
– um *spot* pode conter zero ou mais conteúdos multimédia, um conteúdo multimédia está associado a um e um só *spot*;
– um *spot* contém uma e uma só localização, uma localização pertence a um e só um *spot*;
– um *spot* pode ter zero ou mais comentários associados, sendo que um comentário pode estar associado a nenhum ou apenas um *spot*;
– um conteúdo multimédia pode ter zero ou mais comentários associados, um comentário pode estar associado a nenhum ou apenas um conteúdo multimédia.

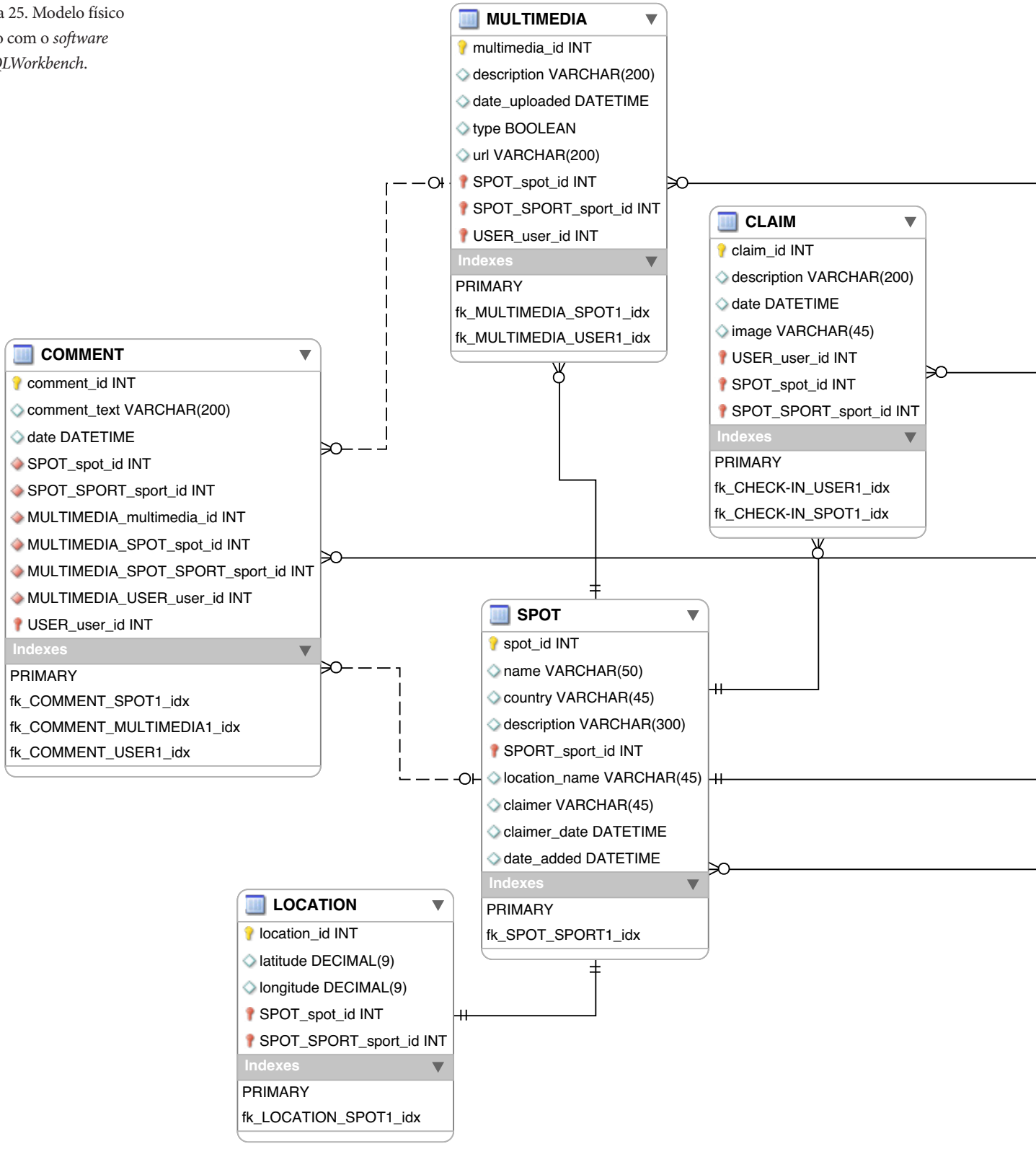
As relações representadas por uma linha a tracejado representam ligações fracas, ou seja, uma das entidades relacionadas é independente da existência das outras. Neste caso um comentário pode estar associado a um conteúdo multimédia ou *spot*, no entanto, não depende de ambos para existir, sendo a sua relação com o utilizador a única da qual depende.

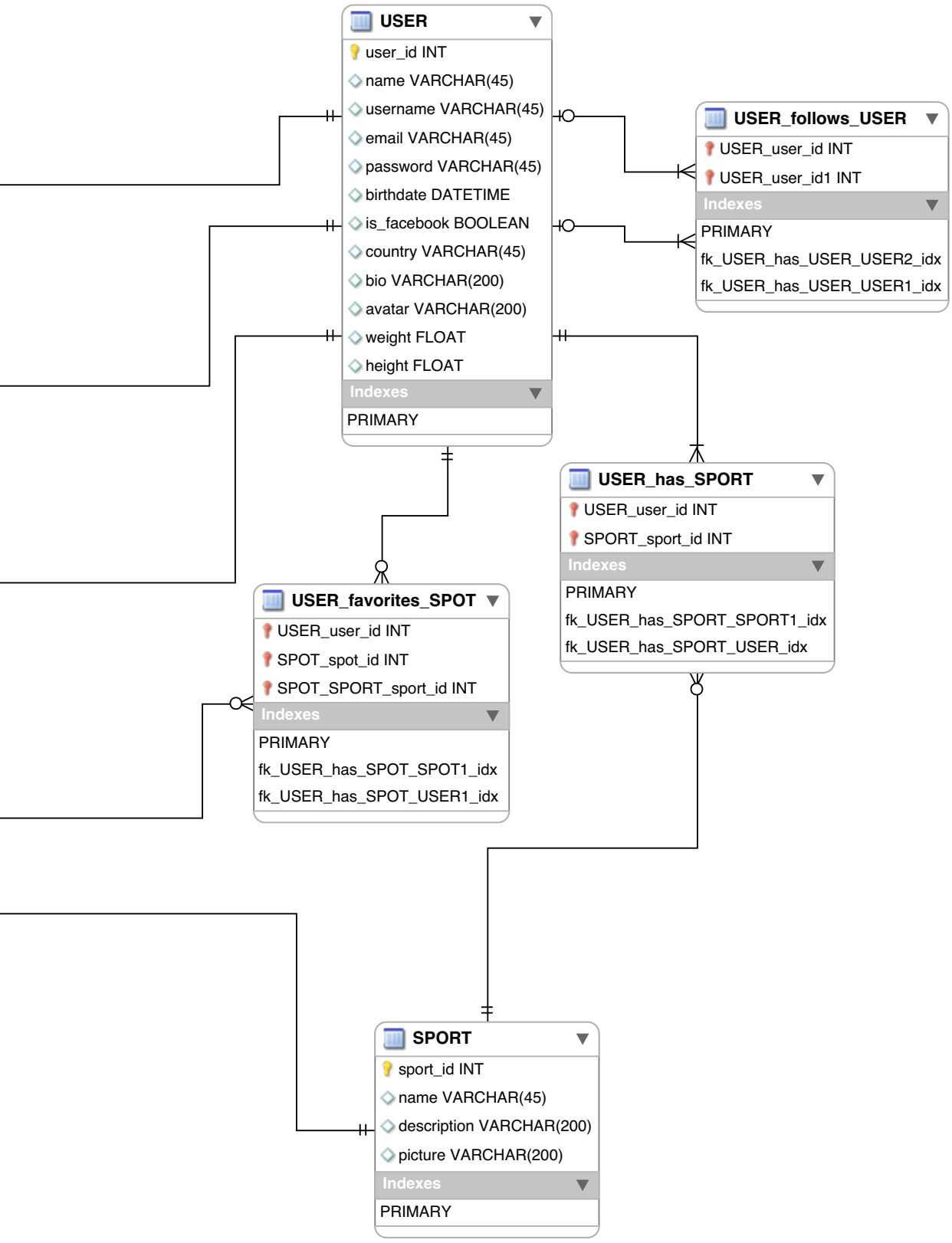
Assim que se conseguiu representar um modelo de dados que pudesse suportar a aplicação, deu-se início à produção do modelo físico. Para tal foi utilizado o software *MySQLWorkbench*, o qual permite desenhar, criar e administrar bases de dados.

Ao criar o modelo físico são geradas novas tabelas com relações de muitos para muitos e são criadas chaves estrangeiras que permitem relacionar as tabelas. No modelo físico foram incluídos todos os atributos necessários para cada entidade.

O modelo criado pode ser visto nas páginas seguintes.

Figura 25. Modelo físico criado com o *software MySQLWorkbench*.




USER
user_id INT
name VARCHAR(45)
username VARCHAR(45)
email VARCHAR(45)
password VARCHAR(45)
birthdate DATETIME
is_facebook BOOLEAN
country VARCHAR(45)
bio VARCHAR(200)
avatar VARCHAR(200)
weight FLOAT
height FLOAT
Indexes
PRIMARY
USER_follows_USER
USER_user_id INT
USER_user_id1 INT
Indexes
PRIMARY
fk_USER_has_USER_USER2_idx
fk_USER_has_USER_USER1_idx
USER_has_SPORT
USER_user_id INT
SPORT_sport_id INT
Indexes
PRIMARY
fk_USER_has_SPORT_SPORT1_idx
fk_USER_has_SPORT_USER_idx
USER_favorites_SPOT
USER_user_id INT
SPOT_spot_id INT
SPOT_SPORT_sport_id INT
Indexes
PRIMARY
fk_USER_has_SPOT_SPOT1_idx
fk_USER_has_SPOT_USER1_idx
SPORT
sport_id INT
name VARCHAR(45)
description VARCHAR(200)
picture VARCHAR(200)
Indexes
PRIMARY

Após a criação do diagrama físico existem quatro diferentes tipos de atributos:
– chaves a amarelo são as chaves primárias das tabelas;
– chaves a vermelho identificam as chaves estrangeiras necessárias para o relacionamento entre tabelas;
– marcas a vermelho identificam as chaves estrangeiras não obrigatórias para o relacionamento entre tabelas;
– marcas a azul claro representam os atributos da tabela.

Da criação do diagrama físico surgem três novas tabelas, resultantes das relações de muitos para muitos entre diferentes utilizadores, entre utilizadores e desportos e entre utilizadores e *spots*. A tabela que resulta da relação entre utilizadores irá armazenar os *ids* dos utilizadores que seguem e são seguidos, ou seja, quando um utilizador segue outro é armazenado no mesmo registo o *id* do utilizador que segue juntamente com o *id* do utilizador seguido.
A tabela resultante da relação entre utilizadores e desportos contém a informação dos desportos que cada utilizador segue na aplicação, sendo registado o *id* do utilizador e o *id* do desporto seguido em cada registo. Na tabela que surge da relação de muitos para muitos entre utilizadores e *spots* são armazenados os *spots* adicionados aos favoritos por cada utilizador, tendo cada registo o *id* do utilizador que adicionou o *spot* aos favoritos e o *id* desse *spot*.

7. DESIGN DA APLICAÇÃO

O design da aplicação foi dividido em três fases: na primeira fase foram desenhadas interfaces de baixa fidelidade, na segunda foi começada a criação de uma identidade visual da aplicação e na terceira fase foram desenhados os ecrãs de alta fidelidade já com a identidade aplicada, os quais foram mais tarde implementados. A segunda e terceira fase decorreram de certa forma em paralelo, uma vez que só se sentiu necessidade de alguns elementos da interface numa fase mais avançada do desenho dos ecrãs de alta fidelidade.

O desenho de baixa fidelidade permitiu uma rápida prototipagem dos ecrãs da aplicação e fez com que se tivesse uma liberdade maior, coisa que não acontece ao desenhar diretamente em digital. Nesta fase, foram desenhados todos os ecrãs que resultaram da análise ao modelo UED. Estes primeiros desenhos permitiram estruturar a aplicação, não se dando, nesta fase, tanta importância à componente gráfica.

Na segunda fase, com o uso dos softwares digitais como o Adobe Photoshop e o Adobe Illustrator, foram desenhados os ecrãs já com atenção ao detalhe visual. A elaboração destes protótipos de alto nível permitiu que a fase de implementação da aplicação decorresse mais rapidamente, uma vez que todos os testes visuais já tinham sido realizados no software de edição.

O desenho dos ecrãs de baixa e alta fidelidade foi realizado em paralelo com a leitura da página *web* "iOS Human Interface Guidelines" disponibilizado pela *Apple*. Este documento introduz o leitor ao design de aplicações para iOS e inclui alguns conceitos básicos relativos à implementação. Aborda temas quer vão desde a identidade e *branding* da aplicação até à navegação
e elementos da interface.

IDENTIDADE VISUAL

Seguiu-se uma tendência minimalista e simplista no desenho da aplicação, pretendendo com isso produzir e reproduzir de forma clara e direta todas as funcionalidades e conteúdos disponíveis na aplicação.
Antes e durante o desenho dos ecrãs de alta fidelidade era crucial incutir uma linguagem visual na aplicação que permitisse manter uma imagem coerente ao longo de todos os ecrãs. Para criar essa linguagem visual foi necessário definir uma paleta de cores, escolher a tipografia a usar e desenhar os ícones necessários. Era também importante desenhar a logomarca da aplicação, no entanto, esta não era absolutamente prioritária para o desenho dos ecrãs de alta fidelidade, uma vez que o seu uso principal seria o ícone da aplicação na grelha de aplicações do sistema iOS.

PALETA DE CORES

Primeiro foi definido um conjunto de cores neutras que funcionassem bem entre si. Estas cores neutras foram importantes no desenho dos primeiros ecrãs, os ecrãs de *login* e *signup*, pois foram as cores escolhidas para os botões. Para representar ações ou validações foram definidas outras duas cores, um tom de verde para ações positivas ou validações bem sucedidas e um tom de vermelho para ações negativas ou validações de erro. Para o texto foi também definida um cinzento escuro, criando assim menos contraste com o branco dos ecrãs e um aspeto mais suave, melhorando a legibilidade.

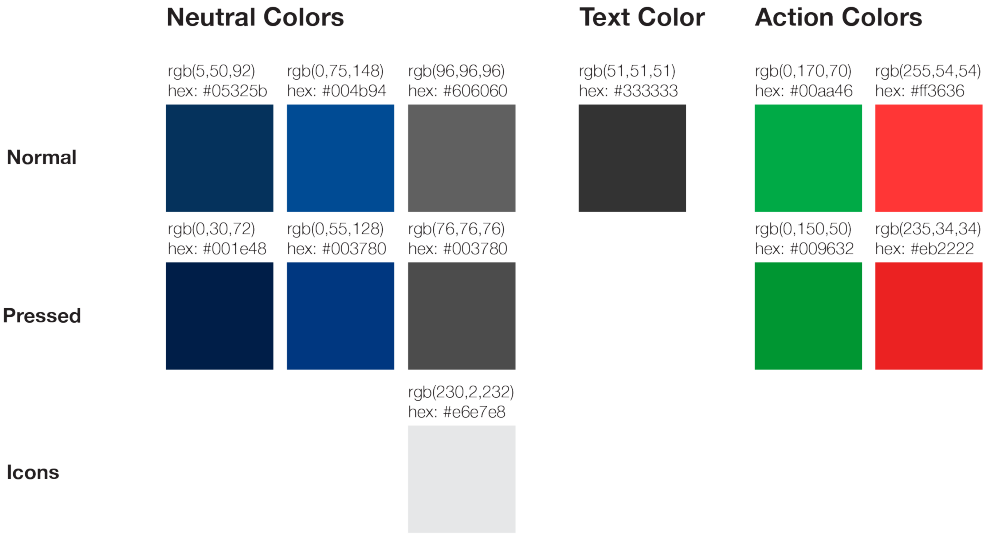


Figura 26. Paleta de cores, com cor usada para texto e validações.

Após serem escolhidas as cores que podiam ser usadas em toda a aplicação em vários elementos da interface foram escolhidas as cores para os desportos que a aplicação iria suportar.

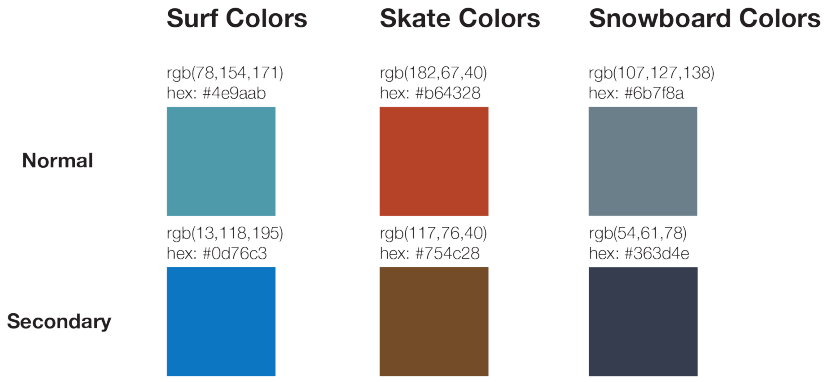


Figura 27. Paleta de cores dos desportos.

As cores escolhidas para os desportos basearam-se nas linguagens do material gráfico associado aos desportos de ação e à linguagem gráfica e filosofia de cada um dos desportos. A cor azul é imediatamente associada ao surf devido à cor do mar. Para o skate optou-se por uma cor mais quente e forte, que representasse de certa forma o dinamismo e irreverência associada ao desporto. A cor escolhida para o snowboard pretende sobretudo criar contraste com a cor branca, a cor da neve, e diferenciar-se das demais.

TIPOGRAFIA

A tipografia escolhida foi a família *Helvetica Neue*, tendo sido usados os pesos *Thin*, *Light* e *Bold*. Esta escolha deveu-se a ser uma tipografia moderna, sem serifa, facilmente legível em dispositivos digitais e sobretudo por ser a tipografia usada por defeito no sistema iOS. Isto cria assim uma consistência entre a aplicação desenvolvida e as restantes aplicações do sistema.

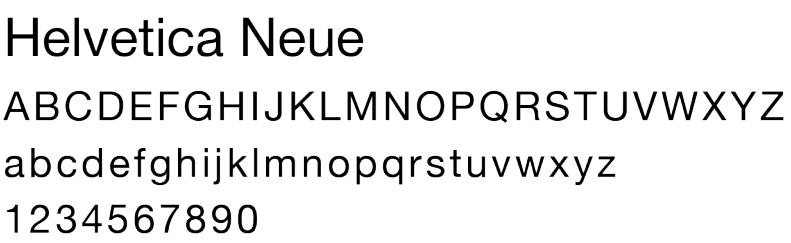


Figura 28. Fonte *Helvetica Neue.*

Figura 29. Pesos usados da família *Helvetica Neue.*

ÍCONES

Durante o desenho dos ecrãs de alta fidelidade foram desenhados os ícones necessários. Estes foram usados para representar um desporto, um utilizador e botões na aplicação.

Os ícones dos desportos tiveram como base as pranchas usadas em cada um dos desportos e as suas características individuais. Foram desenhadas vetorialmente cada uma das pranchas e depois foi desenhado um ícone circular com um bocado da prancha visível.

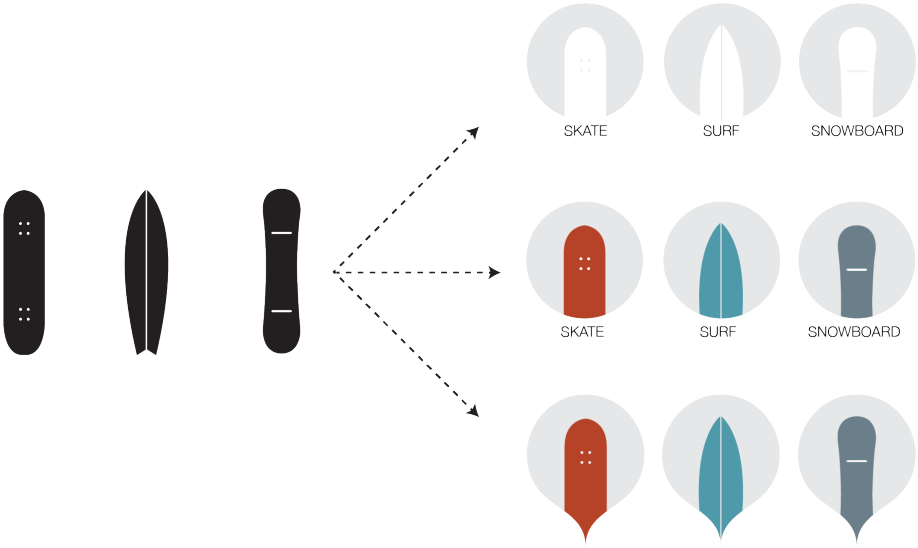


Figura 30. Ícones dos desportos.

Outros ícones foram desenvolvidos para acrescentar botões à aplicação, tendo apenas sido usados os dois primeiros ícones da figura 31, a contar da esquerda para a direita, de cima para baixo. O primeiro ícone é usado para abrir o menu de navegação e o segundo é usado para adicionar fotos na janela de fazer *claim*. Outros ícones foram desenhados com a intenção de serem colocados no menu de navegação, no entanto optou-se por manter o menu o mais simples possível. Nos utilizadores que se registam na aplicação sem foto é apresentada um ícone no lugar da foto como o da imagem à direita.

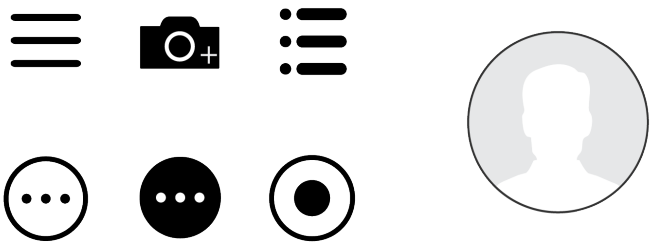


Figura 31. Restantes ícones desenhados.

A aplicação no futuro poderá suportar mais desportos. Tendo em conta essa possibilidade foram desenhados alguns ícones para desportos não suportados atualmente, demonstrando assim também a coerência visual dos ícones desenvolvidos.

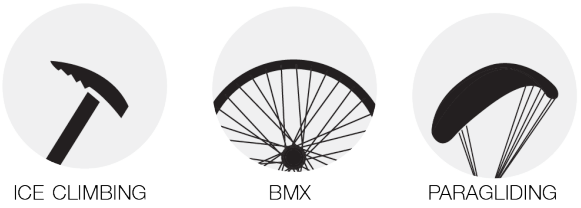


Figura 32. Ícones para possíveis desportos na aplicação.

Mantendo o aspeto circular dos ícones já apresentados, os botões utilizados em vários ecrãs da aplicação assumem este formato circular com um texto a representar uma ação.

Figura 33. Botões de ação.

LOGOMARCA

Para criar uma logomarca começou-se por fazer alguns esboços, tentando transmitir a direção e filosofia da aplicação. Uma vez que era difícil traduzir vários desportos num só gráfico, o caminho tomado foi representar um dos elementos principais da aplicação, os *spots*, e as iniciais do nome *SpotClaimer*, as letras "s" e "c".

Figura 34. Primeiros esboços da logomarca.

Após o desenho dos primeiros esboços procedeu-se ao desenho digital da ideia mais explorada.

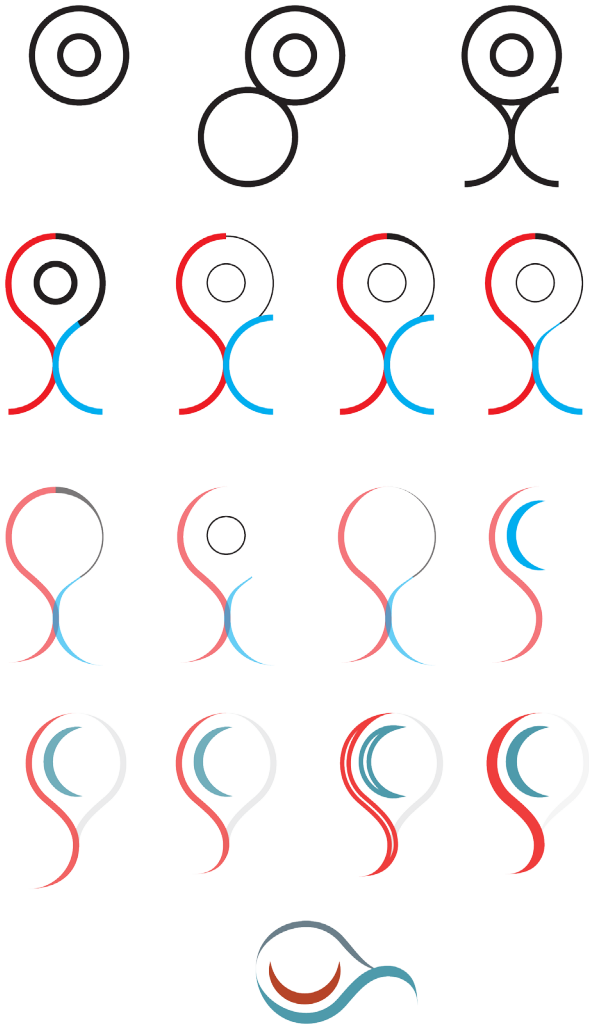


Figura 35. Esboços digitais da logomarca.

A versão mais em baixo, com o ícone na horizontal e as cores dos desportos, permitia traduzir um conceito que representa cada um dos desportos. No caso do surf, a forma da letra "s" representa uma onda. No skate, a forma da letra "c" representa um *half-pipe* e a forma que completa o ícone de *spot* representa uma montanha, associada ao snowboard.

Nos desenhos anteriores, após vários testes a usar a logomarca no ícone da aplicação no dispositivo, não se chegou a um resultado que agradasse. O resultado não satisfatório deveu-se sobretudo à falta de tempo para chegar a uma melhor solução. Optou-se assim por uma abordagem mais direta ao ícone de *spot*, conferindo-lhe um aspeto moderno.

Figura 36. Nova abordagem da logomarca.

As soluções finais para o ícone da aplicação (à esquerda) e para *branding* (à direita) são as seguintes:

Figura 37. Soluções finais para a logomarca.

A fonte usada para a versão de *branding* é a *Titillium Web*, uma fonte gratuita e disponível para download no site Google Fonts.

Embora esta tenha sido adotada para a aplicação, pretende-se no futuro revisitar o processo anterior para tentar chegar a uma logomarca com um conceito semelhante ao descrito e uma melhor solução no desenho da mesma.

ECRÃS DE BAIXA E ALTA FIDELIDADE

Nas páginas seguintes serão apresentados os ecrãs de baixa e alta fidelidade lado a lado e serão descritas as considerações tidas no desenho de cada um. Foram apenas desenhados a alto nível os ecrãs que iriam ser implementados. Ao se dar mais importância ao design destes conseguiu-se abordar a implementação com mais cuidado. Embora os ecrãs de baixa fidelidade dos ecrãs não implementados tenham sido desenhados serão apenas analisados aqueles que passaram pela segunda fase de design. Os ecrãs que apenas foram desenhados em baixa fidelidade não referidos nas páginas seguintes serão colocados no apêndice deste documento. Os nomes para identificar cada ecrã correspondem aos nomes atribuídos no diagrama UED (figura 23) a cada uma das áreas de interação.

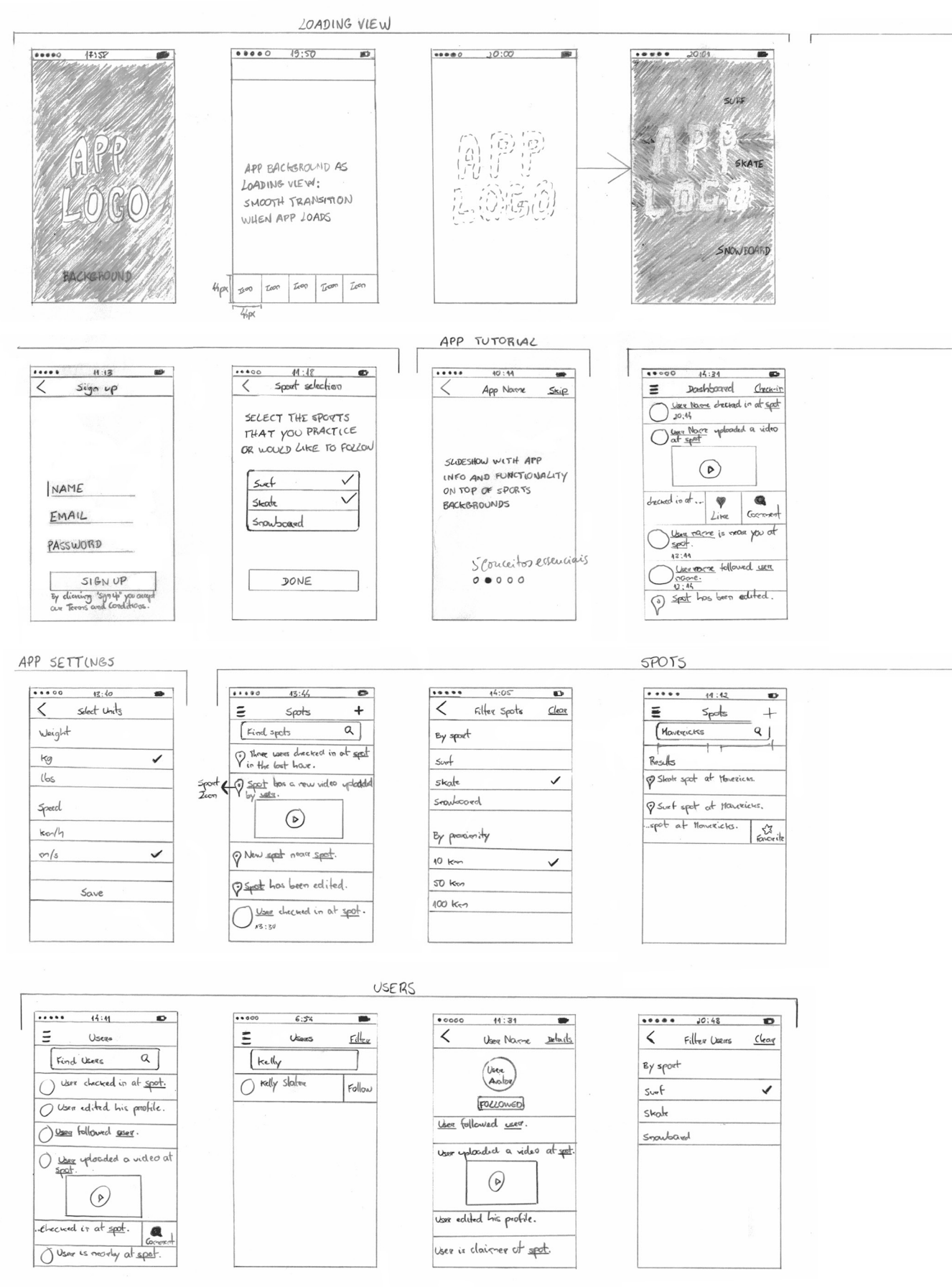
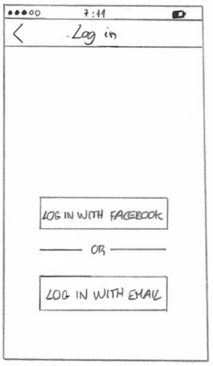
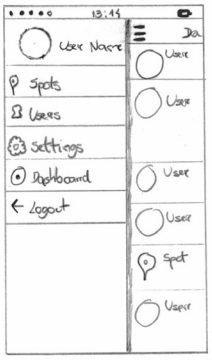


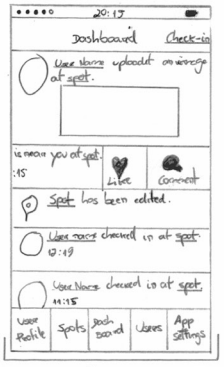
Figura 38. Desenho dos ecrãs de baixa fidelidade.

LOG IN / REGISTER



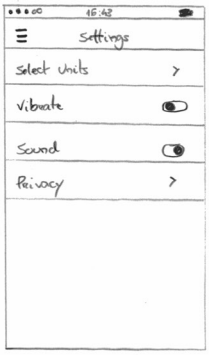
DASHBOARD



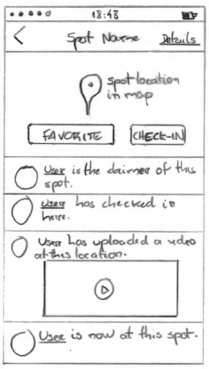


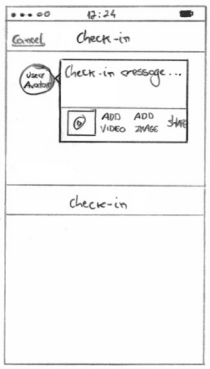
Icons

USER PROFILE

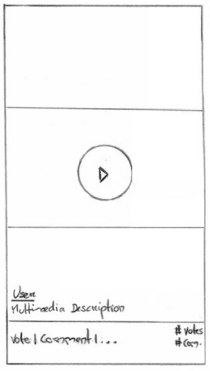


CHECK-IN





VIEW MULTIMEDIA



Loading View

No desenho dos ecrãs de baixa fidelidade foram consideradas várias opções para o carregamento da aplicação, algumas das quais incluíam uma animação e outras apenas imagens estáticas que faziam com que a transição entre o carregamento da aplicação e o ecrã a apresentar fosse o mais suave possível, sendo a solução de usar imagens estáticas recomendada na secção "Launch Images" na página "iOS Human Interface Guidelines". A solução utilizada nos ecrãs de alta fidelidade passa pelo uso de ambas as opções. A status bar desenhada nos ecrãs de baixa fidelidade não é apresentada por opção durante o carregamento da aplicação.

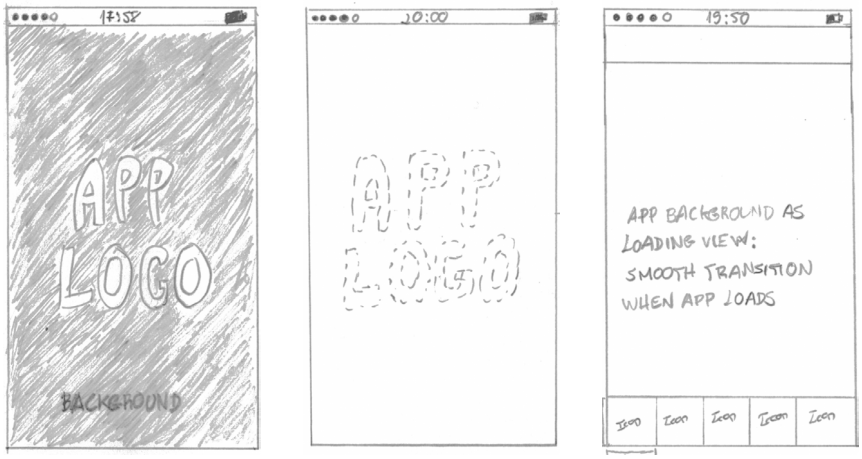


Figura 39. Ecrãs de baixa fidelidade de *Loading View*.

Inicialmente é apresentada uma imagem estática visível na imagem mais à esquerda da figura 40, a qual é incluída no ficheiro *Info.plist* e na pasta *Resources* (descritos no capítulo 8). Assim que é efetuado o carregamento da aplicação em memória é iniciada a animação e em paralelo é efetuado o *login* caso as credenciais do utilizador existam no *Keychain*. A imagem do centro e da direita correspondem a *frames* da animação.

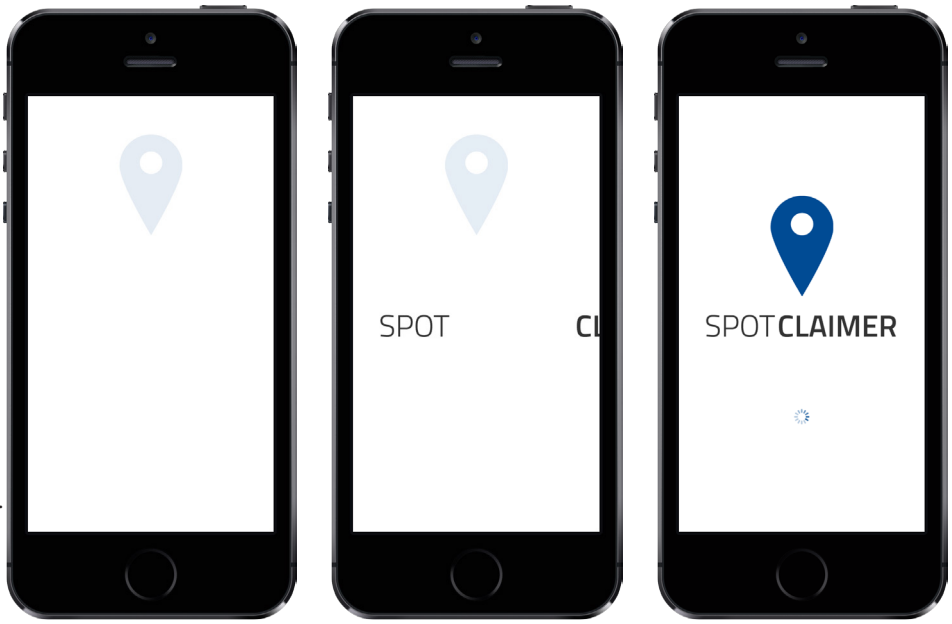


Figura 40. Ecrãs de alta fidelidade de *Loading View*.

Login/Register & Find Nearby Spots

Este é o primeiro ecrã apresentado após a animação de carregamento da aplicação no caso de o utilizador nunca ter utilizado a aplicação ou ter realizado *logout* em algum momento (ao fazer *logout* as credenciais são apagadas do *Keychain*). No ecrã de alta fidelidade optou-se por não incluir o logo como *branding* da aplicação.

iOS Human Interface Guidelines

"For the best user experience, you want to quietly remind users of the brand identity through your choice of font, color, and imagery."

A imagem utilizada como fundo pretende criar uma ligação com o desporto de ação e com os desportos disponíveis. Foi produzida através do uso de três imagens diferentes adquiridas através de pesquisas por imagens sem direitos de autor ou com licença *Creative Commons*. As imagens, de cima para baixo, pertencem aos seguintes autores:
– Arthur Mouratidis[1];
– Éole Wind[2];
– Sob o nome de utilizador *tunaboat* no Flicker[3].

Os botões usados têm alguma transparência de modo a não ocultarem totalmente os atletas visíveis em cada uma das imagens.

Neste ecrã o utilizador pode efetuar *login* ou *signup* ou pode também optar por ver os locais mais próximos. Esta última funcionalidade não foi implementada no protótipo.

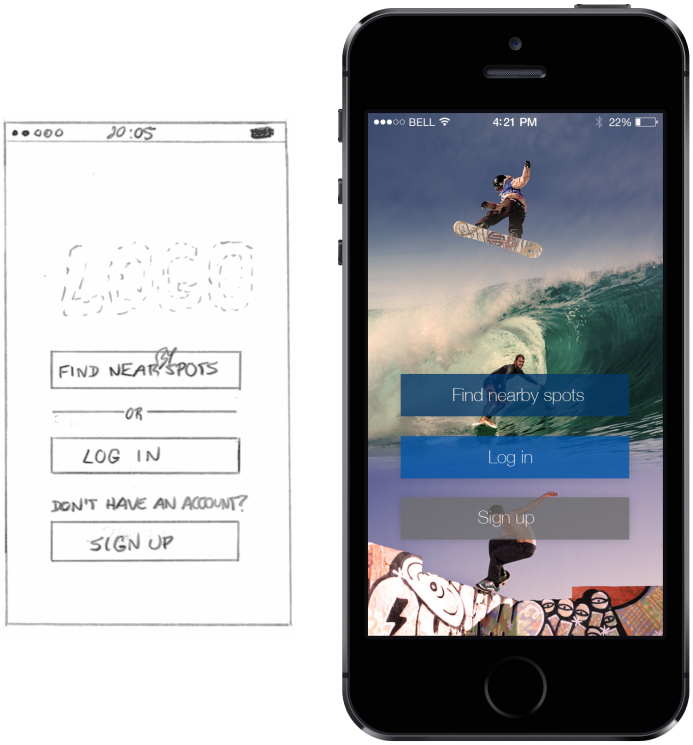


Figura 41. Ecrãs de baixa e alta fidelidade do ecrã *Login/Register & Find Nearby Spots.*

[1] Imagem disponível no *link:* http://commons.wikimedia.org/wiki/File:Snowboard,_Silvia_Mittermüller.jpg
[2] Imagem disponível no *link:* https://www.flickr.com/photos/eole/8138645952/in/photostream/
[3] Imagem disponível no *link:* https://www.flickr.com/photos/tunaboat/3552829441/

Login User - Options

Este é o ecrã apresentado caso o utilizador pressione o botão "Log in" presente no ecrã apresentado anteriormente. É efetuada uma transição dos botões entre o ecrã anterior e este, de modo a manter o mesmo fundo sem que este se mova. Neste ecrã são dadas duas opções ao utilizador para efetuar *login* na aplicação: utilizador pode efetuar *login* com a sua conta Facebook ou com a sua conta de *email*. O *login* por Facebook, quando implementado, irá diminuir o tempo necessário para efetuar *login* na aplicação, reduzindo também o número de passos uma vez que o *login* ficará à distância de um toque no botão "Log in with Facebook". Quando o utilizador pressiona o botão "Log in with e-mail" é apresentado o ecrã que será descrito na página seguinte.

A diferença entre o ecrã de baixa fidelidade e o de alta é ao nível da navegação, uma vez que a transição efetuada para este ecrã foi criada programaticamente sem o uso das transições por defeito do iOS, foi incluído um botão no fundo das opções que permite voltar ao ecrã anterior, em vez da barra de navegação criada por defeito quando se usam as transições entre vistas no iOS.

Figura 42. Ecrãs de baixa e alta fidelidade de *Login User - Options*.

Login User - Email

Caso o utilizador pressione o botão "Log in with e-mail", presente no ecrã anterior, é efetuada uma transição de opacidade entre os ecrãs, sendo apresentada como fundo a mesma imagem dos ecrãs anteriores mas com um efeito de desfoque, simulando assim um efeito de desfoque em tempo real. O efeito de desfoque pretende remover detalhe à foto e assim dar destaque ao conteúdo que tem de ser preenchido pelo utilizador. Ao pressionar um dos campos de introdução de texto é esses elementos e o botão de *login* são movidos para cima e o teclado aparece, mantendo assim o utilizador focado na sua ação atual. Caso o utilizador deseje voltar atrás pode pressionar a cruz presente no canto superior esquerdo do ecrã, o que o leva de volta para o ecrã anterior. O botão de *login* tem dois estados: ativo e inativo, demonstrados pela sua transparência. Assim que os dois campos obrigatórios sejam preenchidos o botão passa ao estado ativo. Ao pressionar o botão é realizado o processamento do *login* do lado do servidor e, caso o *login* não seja efetuado com sucesso, é apresentada uma mensagem de erro conforme é visível na imagem mais à direita. Durante o processamento do *login* o botão é desativado e é apresentada uma animação de carregamento sobre o botão de modo a informar o utilizador que algo está a ser processado pela aplicação.

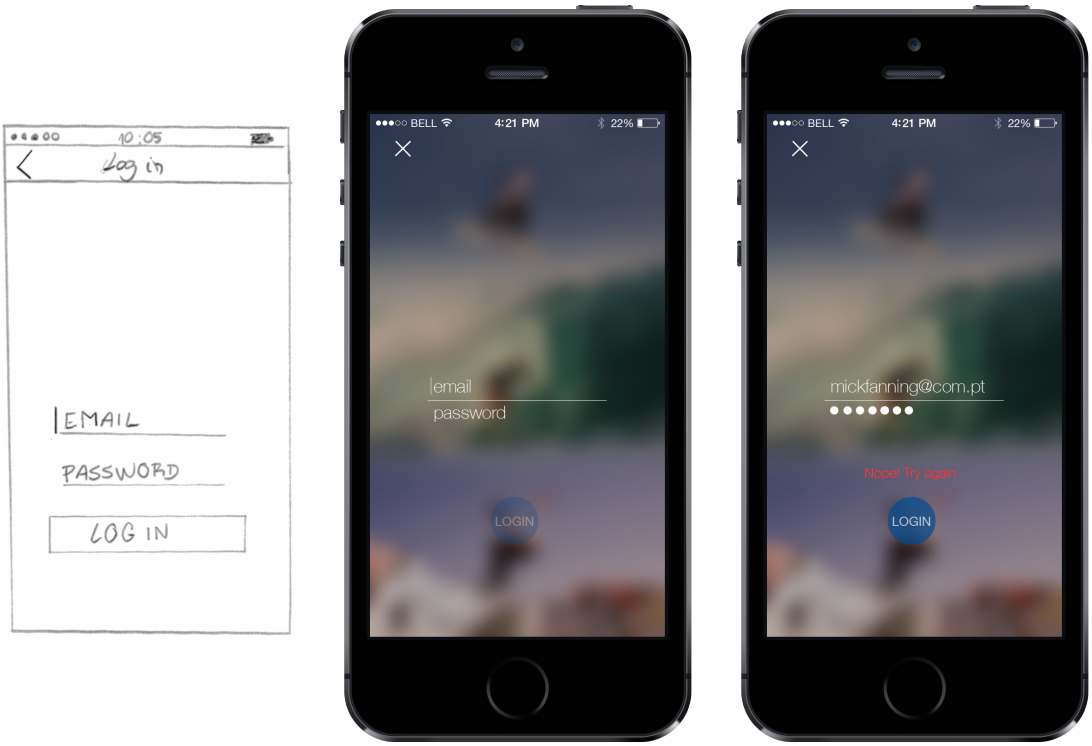


Figura 43. *Login User - Email* em baixa de alta fidelidade.

Register User - Options

Este ecrã de *signup* é acedido através da última opção presente no ecrã referido anteriormente que permite escolher entre *login* e *signup*. Tal como na escolha do *login*, é efetuada uma transição entre os botões dos dois ecrãs, sendo que o botão em baixo com a seta para a esquerda permite retroceder para o ecrã onde foi efetuada a escolha. Entre os ecrãs de baixa e alta fidelidade também é visível esta diferença.

A opção "Sign up with Facebook" pretende dar ao utilizador uma opção de registo bastante rápida, na qual os dados usados para o registo são os dados da sua conta Facebook. Esta opção não está disponível no protótipo.

Ao escolher a opção "Sign up with e-mail" é apresentado ao utilizador um ecrã semelhante ao ecrã de *login* por *email*, sendo, no entanto, necessário o preenchimento de um maior número de campos.

Figura 44. Ecrã *Register User - Options* em baixa e alta fidelidade.

Register User - Email

Ao entrar neste ecrã ocorre uma transição gradual de opacidade, tal como acontece no ecrã de *login* por *email* . Neste ecrã o utilizador tem de preencher os seus dados para se poder registar na aplicação. Pode adicionar, opcionalmente, uma fotografia ao seu perfil ao selecionar o botão de adicionar foto. No fim de a adicionar o utilizador tem a opção de a poder substituir por outra. Após preencher todos os campos e pressionar o botão "Signup" é efetuada uma validação do lado do servidor para verificar se o *email* e o nome de utilizador usados não existem na plataforma *web*. Caso existam o utilizador será avisado com uma mensagem como é visível no ecrã à mais direita. Durante a validação o botão de *signup* é desativado e é apresentada uma animação que informa o utilizador de que uma operação está a ser processada, tal como no ecrã de login. Em caso do registo ser efetuado com sucesso, o utilizador é automaticamente logado na plataforma.

Entre os ecrãs de baixa e alta fidelidade é visível a diferença no número de campos existentes e ao nível da navegação. Em comparação com o ecrã de baixa fidelidade foram adicionados três novos campos: o campo para adicionar uma foto, o *username* e o campo de confirmar *password* uma vez que é comum um utilizador de dispositivos móveis se enganar a preencher os campos. Caso os campos de *password* e de confirmação de *password* não correspondam o utilizador será avisado. A navegação é igual à do ecrã de *login* por *email*: para voltar ao ecrã anterior é necessário pressionar a cruz no canto superior esquerdo. Caso o utilizador pressione essa cruz, é efetuada uma limpeza de todos os campos até então preenchidos.

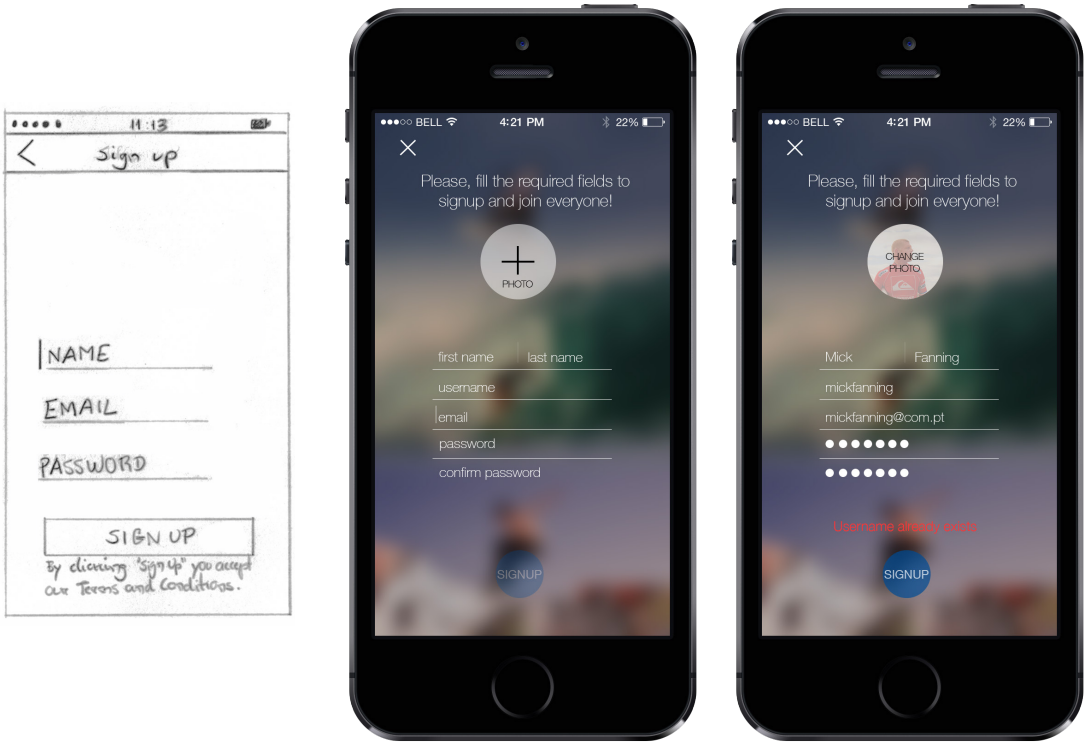


Figura 45. *Register User - Email* em baixa e alta fidelidade.

Welcome Screen

O *Welcome Screen* é o ecrã apresentado ao utilizador sempre que faz *login* ou *signup* com sucesso. Este ecrã tem apenas a função de dar as boas vindas ao utilizador, sendo, no futuro, o ecrã onde deverá ser incluído o tutorial da aplicação pensado no desenho dos ecrãs de baixa fidelidade. Ao pressionar o botão "Go" a aplicação mostra ao utilizador o seu *feed* de atividade no ecrã *User Dashboard*.

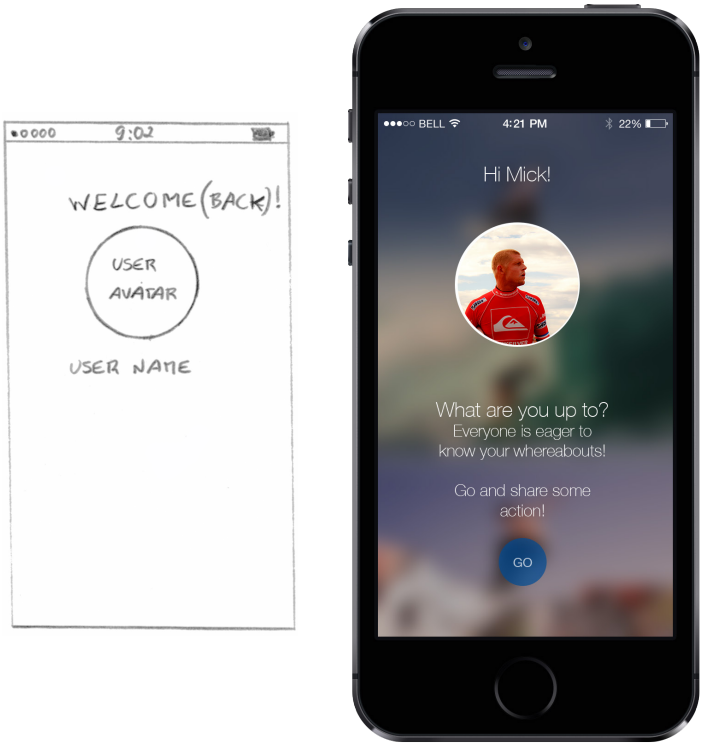


Figura 46. Ecrã *Welcome Screen* em baixa e alta fidelidade.

User Dashboard (Activity)

A primeira diferença visível entre o ecrã de baixa e alta fidelidade é o nome utilizado para o ecrã. Inicialmente foi dado o nome de *Dashboard*, no entanto este foi alterado para *Activity*, o qual se refere a qualquer tipo de atividade realizada na aplicação. Neste ecrã são apresentadas, por ordem cronológica, as atividades efetuadas pelos utilizadores seguidos, as atividades realizadas nos *spots* favoritos e as atividades do utilizador logado. Em ambos os ecrãs é visível a estrutura em tabela, tendo sido mantidas algumas decisões de design como a localização da foto de utilizador e a distinção visual entre o nome dos utilizadores e dos *spots* em relação ao restante texto, uma vez que estes permitem interação com o utilizador. A hora de cada publicação no ecrã de baixa fidelidade foi alterada para uma representação mais *user-friendly*, apresentando não a hora e data da atividade mas sim há quanto tempo atrás foi criada. As fotos também passaram a ocupar quase toda a largura do ecrã e podem ser acompanhadas de uma descrição. Tal como foi explicado anteriormente, a implementação da funcionalidade de *check-in* neste ecrã não foi realizada, tendo sido implementado no ecrã *View Spot*. As ações de gostar ou comentar uma atividade também não foram implementadas no protótipo.

Figura 47. Ecrãs de baixa e alta fidelidade de *User Dashboard*.

Quando o utilizador se regista na aplicação e vai para o ecrã de atividade este encontra-se vazio poiso utilizador ainda não segue nenhum outro utilizador, nem tem nenhum *spot* favorito e ainda não realizou nenhuma atividade. De modo a não se encontrar completamente vazio o ecrã é dada a opção de o utilizador seguir alguém ou de fazer *claim (check-in)* a um *spot* (figura 48, ecrã à esquerda).

No ecrã à direita é visível uma atividade realizada num *spot* favorito por um utilizador que não é seguido pelo utilizador atualmente logado. Neste caso é visível, em vez da imagem do utilizador, o ícone do desporto associado ao *spot*. Caso o utilizador seja seguido, a publicação será apresentada com a sua foto de perfil, não havendo duplicados, ou seja, não são apresentadas ambas as publicações do *spot* e do utilizador referentes à mesma atividade.

Ao efetuar *scroll* é visível o efeito de transparência e desfoque sobre o conteúdo aplicado na barra de navegação da aplicação.

iOS Human Interface Guidelines

"Let translucent UI elements hint at the content behind them. Translucent elements—such as Control Center—provide context, help users see that more content is available, and can signal transience."

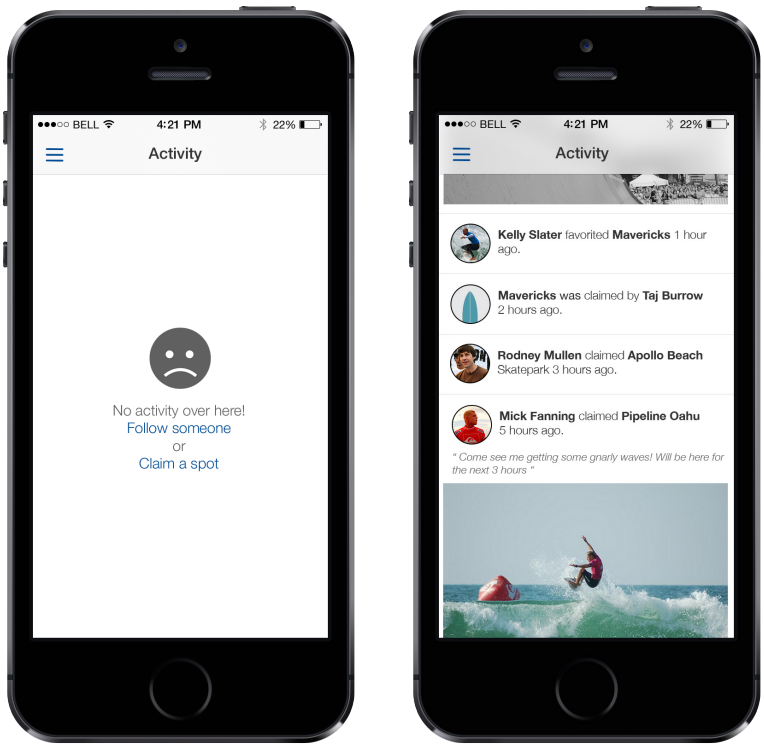


Figura 48. *Activity* sem atividade (esquerda) e translucência na barra de navegação (direita).

Navigation Menu

O menu de navegação permite navegar entre os diferentes ecrãs da aplicação. Nos ecrãs de baixa fidelidade foram abordadas duas alternativas. A do ecrã à esquerda consistia numa navegação através de uma lista de ícones colocada no fundo do ecrã da aplicação. Tinha como vantagem a rapidez com que era possível trocar entre ecrãs mas a desvantagem de ocupar sempre alguma área do ecrã. A segunda alternativa, na imagem ao centro, permitia realizar a navegação através de um menu que aparecia do lado esquerdo da aplicação ao pressionar um botão no ecrã atualmente visível. Ao pressionar um item do menu a aplicação apresentava o ecrã selecionado, sendo que neste caso o menu só era apresentado quando o utilizador pretendia, deixando assim mais espaço para o ecrã atual apresentar mais conteúdo. Optou-se por esta alternativa como é visível no ecrã à direita, no entanto, o ecrã atual é redimensionado de modo a que a barra de estado não fique dividida entre dois ecrãs com fundos diferentes, perdendo assim contraste num deles. Tentou-se manter este menu o mais simples possível, dando no entanto o contexto do ecrã atual através de uma versão à escala do mesmo.
Ao escolher um item do menu ou ao pressionar na área do ecrã atual a aplicação fecha o menu de navegação.

Figura 49. *Navigation Menu* em baixa e alta fidelidade.

Nearby Users

Nos desenhos de baixa fidelidade do ecrã *Nearby Users* começou-se por desenhar uma versão bastante semelhante à do ecrã *User Dashboard* mas apresentando apenas a atividade dos utilizadores seguidos, como é possível ver no ecrã à esquerda. Por baixo da barra de navegação deste ecrã existia uma caixa de pesquisa que permitia encontrar utilizadores e segui-los (isto é visível no ecrã ao centro).

No desenho de alta fidelidade este ecrã passou a apresentar uma lista dos utilizadores registados, ordenados por ordem alfabética. Neste ecrã é possível seguir ou deixar de seguir os utilizadores sem ser necessário visualizar os seus perfis, sendo que ao pressionar outra zona da linha da tabela além do botão "Follow" a aplicação mostra o perfil do utilizador selecionado (ecrã *View User*).

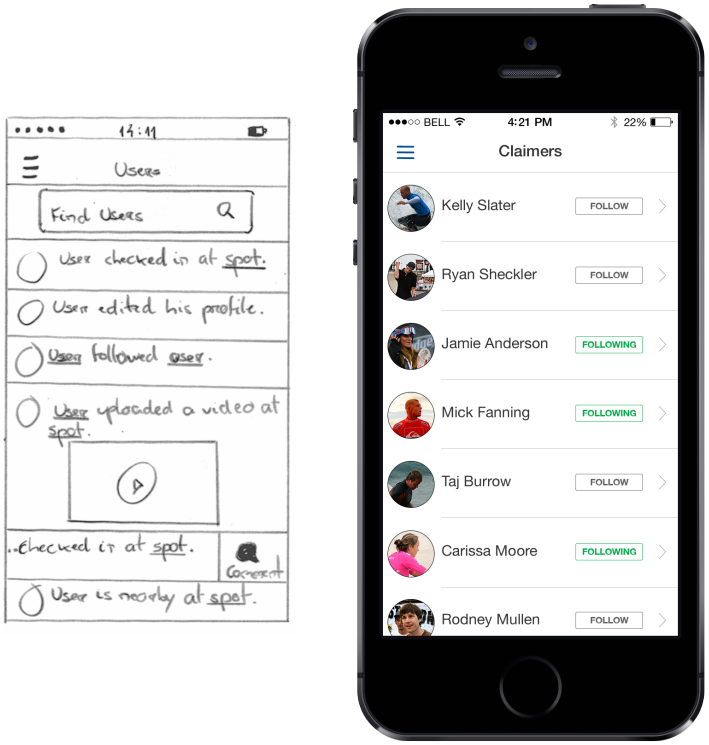


Figura 50. Ecrã *Nearby Users* em baixa e alta fidelidade.

View User

O ecrã *View User* apresenta o perfil do utilizador selecionado, no qual pode ser visto o seu registo de atividade. No desenho de alta fidelidade, além da inclusão do botão para seguir o utilizador foram também incluídas duas outras informações: o número de utilizadores que seguem o utilizador e o número de pessoas que o utilizador segue na aplicação. Ao fazer *scroll* no registo de atividade do utilizador a lista começa a ocupar a totalidade do ecrã, escondendo a foto e informação básica do utilizador e permitindo assim usar uma maior área de ecrã para mostrar o conteúdo.

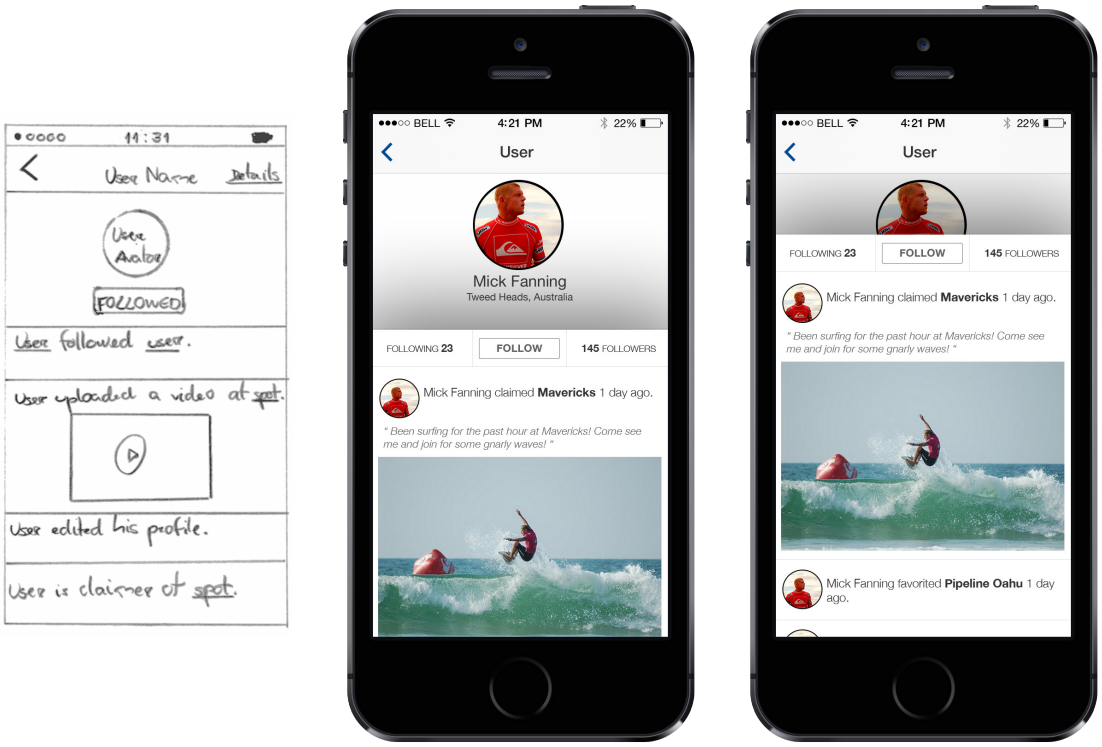


Figura 51. *View User* em baixa e alta fidelidade com *scroll* (direita).

Tal como no ecrã de atividade, caso o utilizador a ser visto ainda não tenha efetuado nenhuma atividade na aplicação será apresentada uma mensagem com essa indicação.

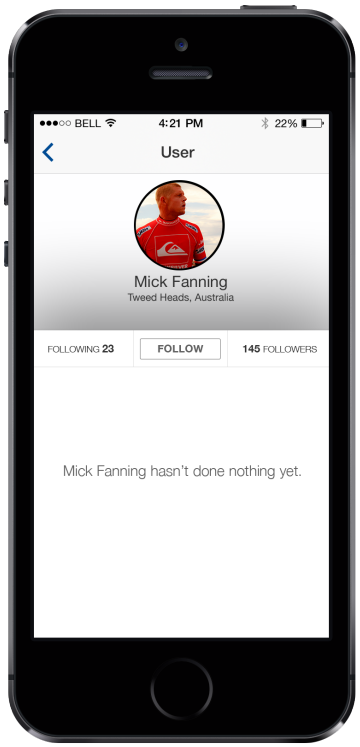


Figura 52. *View User* sem atividade.

Nearby Spots

O processo de desenvolvimento do ecrã *Nearby Spots* é semelhante ao do ecrã *Nearby Users*. Inicialmente, nos ecrãs de baixa fidelidade também se tinha a intenção de apresentar neste ecrã um registo de atividade dos *spots* seguidos pelo utilizador, tendo também uma caixa de pesquisa que permitia pesquisar *spots* na aplicação e adicioná-los aos favoritos. No desenho de alta fidelidade optou-se por uma lista de todos *spots* ordenados por ordem de proximidade em relação à localização do utilizador, do mais próximo até ao mais distante, uma vez que interessa ao utilizador saber que *spots* tem perto de si para praticar desporto. Ao selecionar um dos *spots* da lista é apresentado o ecrã *View Spot*.

Figura 53. Ecrãs *de* baixa e alta fidelidade de *Nearby Spots*.

View Spot

Neste ecrã é possível ver os detalhes de um *spot* e o registo das atividades lá realizadas. Junto ao nome do *spot* foi adicionada uma estrela que permite adicionar o *spot* aos favoritos. O design do ecrã é bastante similar ao do ecrã *View User*, apresentando a informação do *spot*, o número de *claims* que foram feitos neste *spot* e o número de utilizadores que adicionaram este *spot* aos seus favoritos. No centro da linha é indicado o "Spot Claimer", isto é, o utilizador que efetuou mais *claims* neste *spot*.

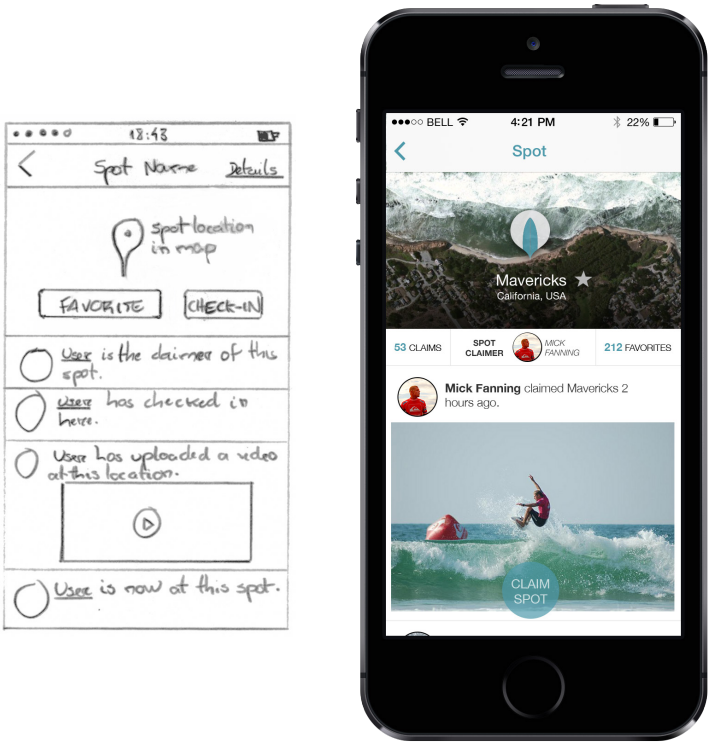


Figura 54. Ecrãs *de* baixa e alta fidelidade de *View Spot*. *Spot* de *surf* (direita).

Nos três ecrãs de alta fidelidade é visível a diferença entre os *spots* de diferentes desportos. Essa diferença é marcada com o uso de cores diferentes na barra de navegação, na primeira linha da tabela e no botão "Claim Spot". É também usado um ícone diferente para marcar a localização do spot no mapa. Tal como no ecrã *View User*, ao deslizar a tabela com o registo de atividade para cima, a área que contém o mapa e a informação básica do *spot* é escondida como é visível no ecrã mais à direita. Caso essa tabela seja deslizada para baixo é realizado um efeito de *parallax* que mostra a descrição do *spot* e faz zoom ao mapa no local do *spot*. Ao pressionar o botão "Claim Spot" é apresentada a janela *Check-in*.

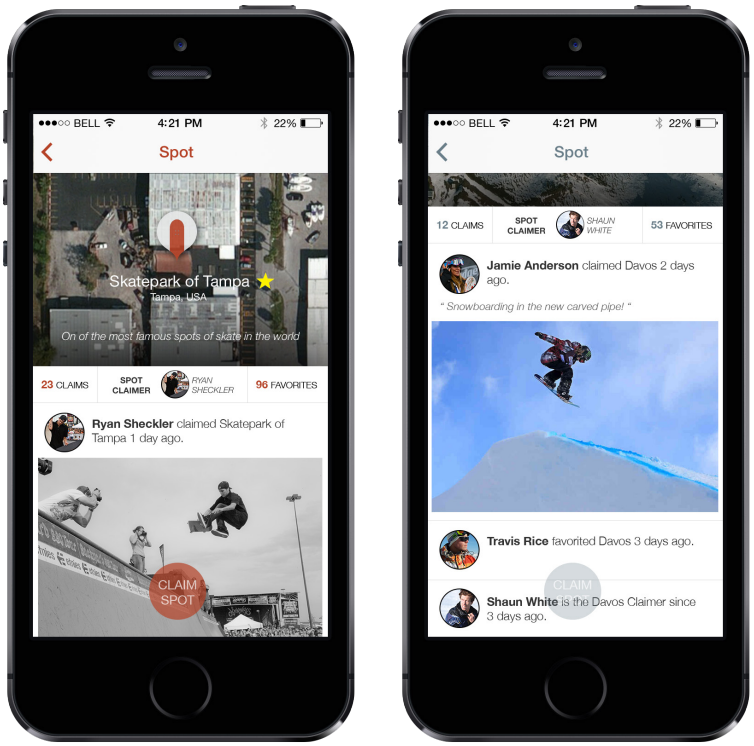


Figura 55. *Spot* de *skate* com *scroll* para baixo (esquerda). *Spot* de *snowboard* com *scroll* para cima (direita).

Check-in (Claim)

O ecrã *Check-in* trata-se de uma janela que é apresentada sobre o ecrã do *spot* onde se pressionou o botão "Claim Spot". Esta janela só será apresenta caso o utilizador não tenha efetuado um *claim* há menos de três horas no *spot*. Neste ecrã o utilizador pode adicionar uma descrição e uma foto ao seu *claim*. Ao pressionar o botão "Claim" é realizado o *upload* da imagem, caso tenha sido adicionada, e da informação para o servidor e é processado o *claim*. Caso seja adicionado com sucesso a janela é fechada e o utilizador irá ver a sua publicação no registo de atividade do *spot* onde abriu a janela.

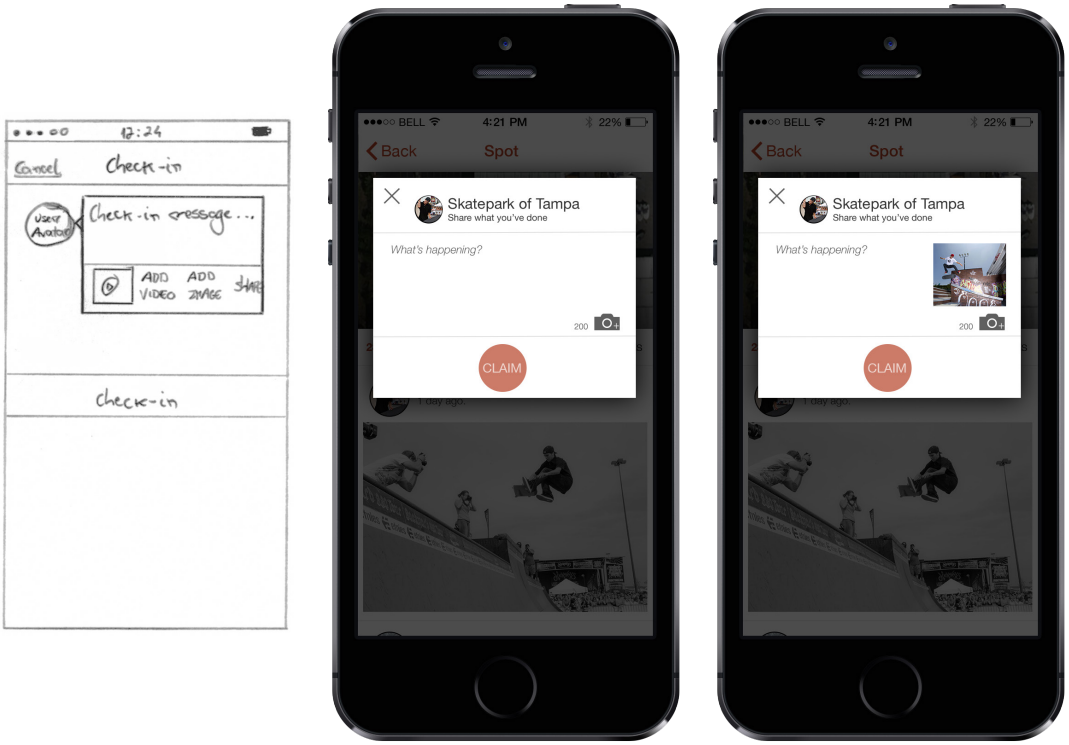


Figura 56. Ecrãs *Check-in* em baixa e alta fidelidade. Ecrã com foto adicionada (direita).

Add Spot

Este é o ecrã que permite aos utilizadores adicionar novos *spots* à aplicação, sendo uma funcionalidade imprescindível uma vez que a aplicação é baseada em conteúdo gerado pelos utilizadores. O ecrã pode ser acedido ao pressionar o botão "+" no ecrã *Nearby Spots*. A maior diferença entre os ecrãs de baixa e alta fidelidade é o processo de adição de um *spot* ter um diferente número de passos. Enquanto que no desenho de baixa fidelidade o mapa seria arrastado na secção logo abaixo da barra de navegação e os detalhes do *spot* seriam editados mais em baixo, no desenho de alta fidelidade dividiu-se esta funcionalidade em dois passos: no primeiro o utilizador arrasta e faz zoom ao mapa de modo a conseguir localizar a posição exata do *spot*; no segundo passo, após pressionar o botão "Done", é apresentada uma outra área que irá permitir adicionar os detalhes do *spot*, mantendo a localização definida no primeiro passo. No fim de preencher todos os campos obrigatórios e de selecionar o desporto praticado no *spot* a adicionar, pode-se pressionar o botão "Save", o qual estará inativo até todos os detalhes serem preenchidos. Ao ser pressionado, este fica inativo até o *spot* ser adicionado na base de dados do servidor e é apresentada uma animação que indica ao utilizador o processamento da informação.

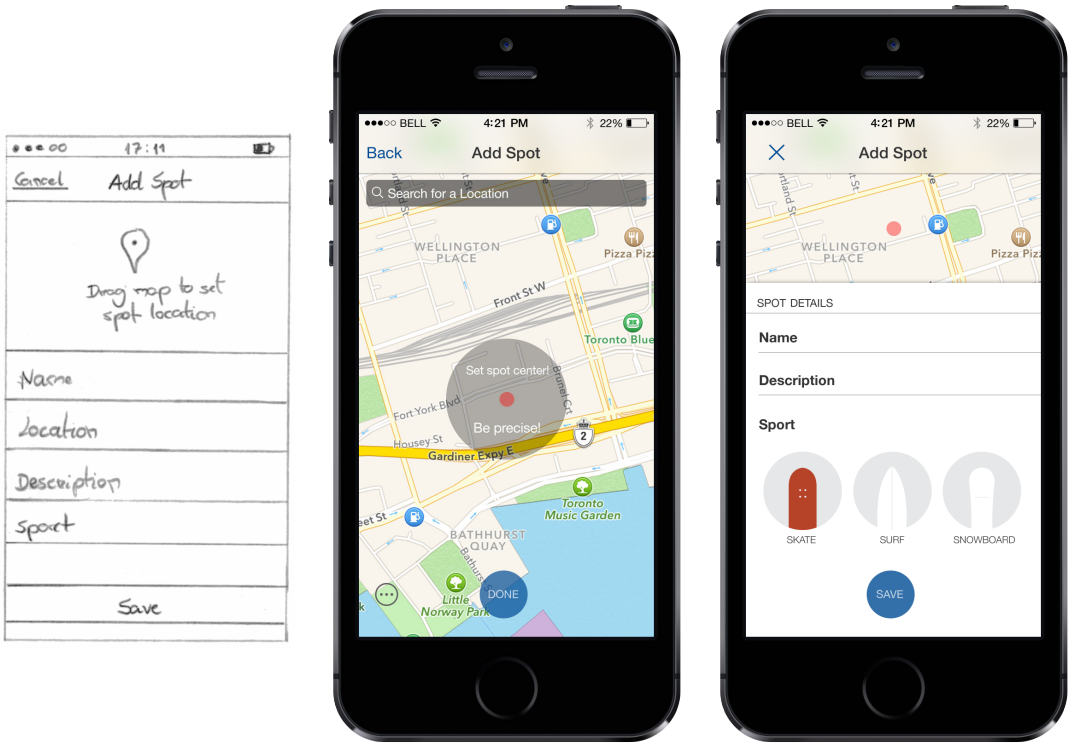


Figura 57. Ecrãs de baixa e alta fidelidade de *Add Spot*. Ecrã para adicionar detalhes (direita).

User Profile

O ecrã *User Profile* é bastante semelhante ao ecrã *View User* e pode ser acedido através do menu de navegação. Neste ecrã é apresentado o perfil do utilizador que efetuou *login* na plataforma. Aqui também se encontra o botão "Logout", presente na barra de navegação, que ao ser premido desconecta o utilizador da aplicação e apresenta o ecrã *Login/Register & Find Nearby Spots*. Ainda no ecrã *User Profile*, é também apresentado o número de utilizadores que seguem o utilizador e o número de utilizadores que este segue. O botão de "Follow" é ocultado neste ecrã uma vez que o utilizador não se pode seguir a si próprio.

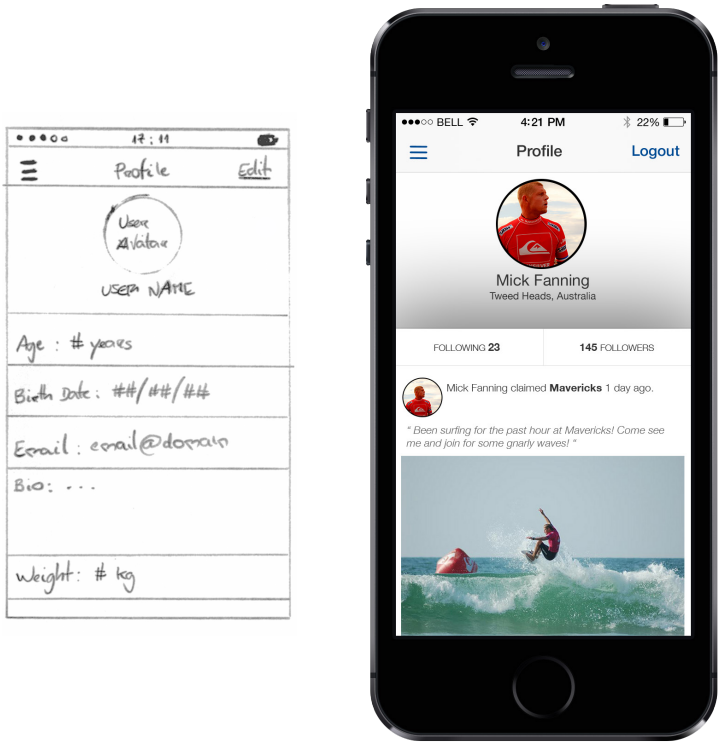


Figura 58. Ecrãs *de* baixa e alta fidelidade de *User Profile.*

8. IMPLEMENTAÇÃO

TECNOLOGIAS USADAS

Ao longo da definição dos cenários e de personas e criação do UED (figura 23) foram investigadas tecnologias atuais que permitissem um rápido desenvolvimento da plataforma *Web* para dar suporte à aplicação a desenvolver. A *framework Django* foi a ferramenta escolhida uma vez que o mestrando já tinha algum conhecimento e experiência com a tecnologia, conhecimento esse adquirido em trabalho extracurricular. Anteriormente à escolha da *framework* para o desenvolvimento da plataforma *Web* foi escolhida a plataforma *Xamarin* para implementar a aplicação. A plataforma *Xamarin* assenta na linguagem C# e permite o desenvolvimento de aplicações móveis e *desktop*, sendo que, no caso dos dispositivos móveis, permite uma grande portabilidade de código entre os sistemas operativos existentes. Para gerir todo o código existente usou-se um sistema de controlo de versões, o software *Git*.

FRAMEWORK DJANGO

A *framework Django* é uma tecnologia que permite o rápido desenvolvimento de plataformas *web* com grande escalabilidade e uma grande camada de abstração em comparação com o uso de linguagens como PHP e MySQL. A *framework Django* adere ao princípio DRY *(Don't repeat yourself)*, princípio esse que tem como objetivo a redução da repetição de código e documentação no caso de plataformas de desenvolvimento, mantendo o código separado em blocos e menos dependentes entre si, permitindo assim alterações locais que não afetem outras partes do sistema.

Desenvolvida em Python, *Django* permite criar uma base de dados em poucas linhas de código e disponibiliza uma API para aceder, guardar e modificar o conteúdo na base de dados, garantindo assim uma rápida prototipagem de uma plataforma *Web*. O código desenvolvido é dividido maioritariamente em três tipos de ficheiros: os modelos, as vistas e os *templates* (*models*, *views* e *templates*) e é organizado e separado por *apps* caso o programador assim o entenda. Cada *app* contém pelo menos um ficheiro para os modelos e outro para as vistas, sendo que os *templates* são normalmente colocados numa pasta separada pois não fazem parte da estrutura de uma *app*.

Os modelos são os ficheiros que têm o código que define a estrutura da base de dados: as tabelas, as relações entre si e os atributos de cada entidade. As vistas incluem as funções que permitem aceder, inserir e modificar o conteúdo na base de dados. Numa vista é possível criar funções que devolvem por exemplo uma lista de desportos presentes na tabela da base de dados que inclui todos os desportos. Essa consulta é efetuada acedendo aos objetos da classe a que os desportos pertencem, classe essa que foi previamente definida no ficheiro de modelos dessa *app*.

As vistas podem incluir algoritmos de consulta bastante simples até algoritmos mais complexos, no entanto, a camada de abstração do *Django* permite que esses algoritmos sejam muito mais rapidamente implementados em comparação com outras *frameworks*.

Os *templates* são os ficheiros que permitem renderizar uma página para mostrar ao utilizador num *browser web*. São ficheiros *.html* em que se pode incluir lógica de servidor através do sistema de *templating* presente no *Django*. As vistas processam todo o conteúdo que é depois usado pelos *templates* para renderizar a página *.html* do lado do servidor e enviar o conteúdo já preparado para o cliente.

O exemplo seguinte mostra como pode ser a estrutura de um projeto de *Django,* onde (+) corresponde a uma pasta e (-) a um ficheiro:

```
+ django project
  + project name
    - templates
      - index.html
    - settings.py
    - urls.py
  + app
    - models.py
    - views.py
  + app2
    - models.py
    - views.py
    - manage.py
```

Um projeto de *Django* pode conter uma ou mais *apps*. As *apps* podem ser criadas pelo programador ou podem ser incluídas como módulos para incluir funcionalidades já desenvolvidas e disponibilizadas por outros programadores. Quando se cria uma *app* esta deve ser incluída no ficheiro *settings.py*, na variável que define a lista de *apps* instaladas.

O ficheiro *settings.py* define várias propriedades do projeto, tais como as definições da base de dados, chaves secretas, opções de *debug*, entre outras.

Outro ficheiro bastante importante num projeto de *Django* é o *urls.py*. Neste ficheiro são definidos todos os URLs (*Uniform Resource Locator* - Localizador Padrão de Recursos) que podem ser acedidos através de um *browser* ou de qualquer software que inclua funcionalidades para executar pedidos *web*. Os URLs podem ser formados recorrendo ao uso de expressões regulares ou apenas palavras como por exemplo "http://www.spotclaimer.com/sports/", sendo que ao aceder a este URL se pode obter uma página com os desportos existentes na plataforma. No caso de não usar um esquema de URLs como o existente no *Django*, o URL anteriormente referido teria de ser representado, por exemplo, pelo URL "http://www.spotclaimer.com/sports.php?sports=all". Isto permite a utilização de URL mais limpos, mais legíveis e que podem facilmente ser referidos a outra pessoa.

O ficheiro *manage.py* permite aceder a várias funções, de entre as quais se destacam as seguintes:

– *python manage.py startapp app_name* — permite criar uma nova *app* dentro do projeto.

– *python manage.py syncdb* — executa um comando que cria as tabelas da base de dados definidas nos modelos e insere dados nelas através de um ou vários ficheiros previamente criados em formato JSON ou XML.

– *python manage.py runserver* — inicia o processo do servidor que permite aceder aos URLs através de um *browser*.

Quando é efetuado um pedido acedendo a um URL num *browser*, o processo até ao utilizador visualizar a página pedida é o seguinte:
– é verificado se o URL corresponde a algum dos URLs presentes na lista de URLs existente no ficheiro *urls.py*. Esta verificação é efetuada pela ordem da lista e caso o URL não exista é devolvida uma página de erro.
– caso o URL exista, o *Django* processa a função da *view* definida no URL e devolve o conteúdo processado para um *template .html* ou num formato de dados como o JSON ou XML.
– se a *view* devolve a resposta num ficheiro de *template .html*, este é processado posteriormente com a lógica de *templating* que este incluir, sendo enviado para o cliente e mostrada ao utilizador a página *.html* totalmente renderizada.

Assim, a *framework Django* é a ideal para criar uma plataforma *web* que sirva de suporte a uma aplicação móvel. É usada por várias plataformas com bastante relevância no mercado, como é o caso do Instagram, Pinterest e Rdio.

XAMARIN – C#

Como já foi referido anteriormente, a plataforma escolhida para o desenvolvimento da aplicação móvel foi a plataforma Xamarin. Tem como base a linguagem C#, uma linguagem moderna com funcionalidades como a inferência de variáveis e a programação assíncrona, as quais permitem um rápido desenvolvimento de aplicações que posteriormente são compiladas para código nativo a fim de serem executadas em múltiplos dispositivos. A partilha de código permite evitar uma aprendizagem específica para plataforma, ajudando o programador a implementar funcionalidades semelhantes em múltiplas plataformas recorrendo ao mesmo código. Mesmo focando o desenvolvimento apenas numa plataforma, como foi o caso desta dissertação, o Xamarin permite uma compatibilidade a 100% entre as API existentes nas linguagens nativas, ou seja, qualquer pedaço de código implementado na linguagem nativa da plataforma pode ser implementado em C# e, na maioria das vezes, recorrendo a menos código. Embora exista portabilidade de código, as interfaces podem ser específicas por plataforma, dando assim liberdade aos designers e programadores de adaptarem as interfaces aos sistemas operativos para os quais a aplicação é destinada.

O Xamarin disponibiliza um IDE (*Integrated Development Environment* - Ambiente Integrado de Desenvolvimento), o *Xamarin Studio*, com todas as funcionalidades necessárias para o desenvolvimento de uma aplicação para OS X e Windows, sendo o único IDE para desenvolvimento existente em OS X. Os utilizadores de Windows podem optar entre o Xamarin Studio e o *Visual Studio* da *Microsoft*. Para o desenvolvimento da aplicação durante esta dissertação foi utilizado o *Xamarin Studio* em ambiente OS X, o que permitiu a integração com o software de desenvolvimento nativo para aplicações iOS, o *Xcode*. A integração do *Xamarin Studio* com o *Xcode* permite criar as interfaces visualmente no *Xcode* para cada ecrã ou através de um *storyboard* que, além de se poder desenhar todos os ecrãs, é também possível definir as ligações e transições entre os mesmos. As alterações efetuadas no *Xcode* são depois sincronizadas no *Xamarin Studio* e podem ser vistas após a compilação e execução da aplicação.

Cada projeto de desenvolvimento de uma aplicação consiste numa solução, a qual contém todos os ficheiros e definições necessárias para executar a aplicação. Quando se pretende testar a aplicação esta deve ser compilada, sendo executada posteriormente. A aplicação pode ser executada em vários simuladores que oferecem uma emulação de alguns dos dispositivos existentes e aos quais a aplicação se destina. Além dos simuladores é também possível executar a aplicação num dispositivo real, no caso do iOS é necessário que o programador tenha uma conta no site Apple Developer (https://developer.apple.com) e que tenha aderido ao *iOS Developer Program*, o qual permite instalar aplicações num dispositivo e publicá-las na loja de aplicações, a App Store, mediante o pagamento de uma quantia anual. Aquando da execução de uma aplicação em modo *debug* ou *release* é possível detectar possíveis erros existentes no código e, quase sempre, identificar a causa destes, dando assim a possibilidade ao programador de os corrigir rapidamente. Deve-se

sempre optar por testar o código em dispositivos reais, uma vez que é nestes que a aplicação irá correr; ao testar em simuladores nem sempre se tem atenção a alguns detalhes, como por exemplo se a área em que um botão pode ser tocado é pequena ou não. Outros fatores como as funcionalidades de localização e de conectividade através de redes móveis ou *wi-fi* devem ser testados em dispositivos físicos. Os simuladores são ideais para o desenvolvimento de funcionalidades e desenho da interface da aplicação uma vez que a execução do simulador é mais rápida do que fazer *upload* da aplicação num dispositivo físico. Os testes em dispositivos físicos são também ideais para permitir efetuar testes de usabilidade com potenciais utilizadores da aplicação.

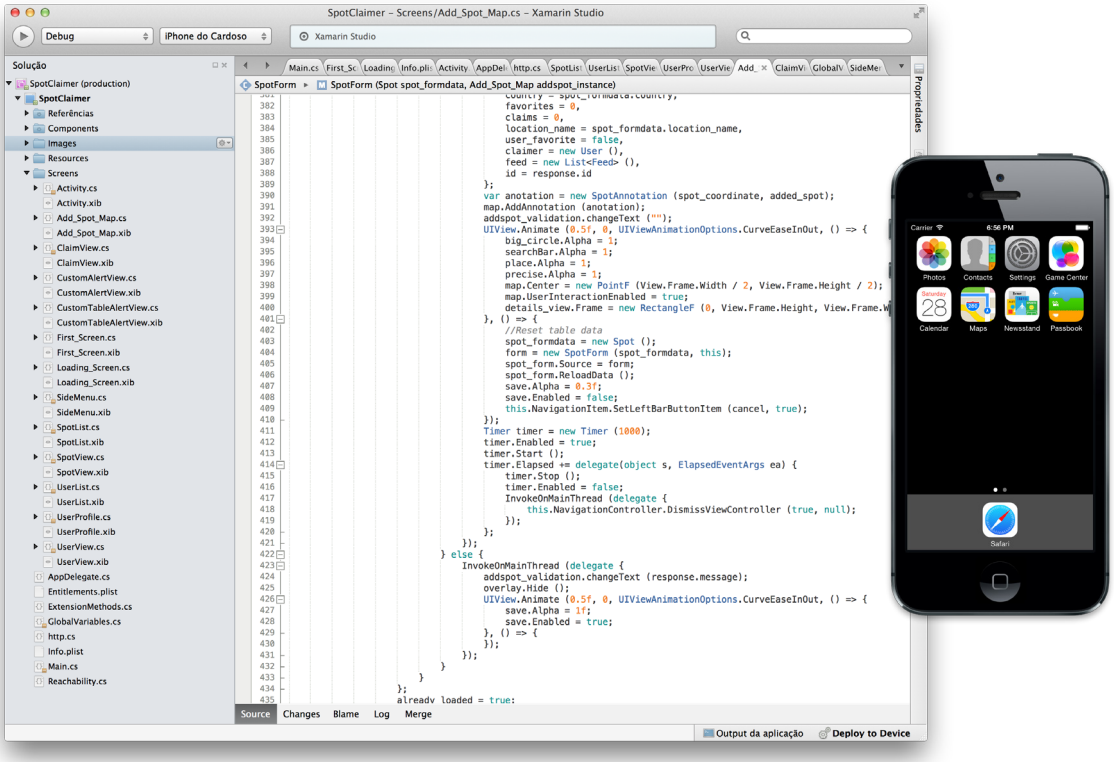


Figura 59. *Xamarin Studio* à esquerda e *iPhone Simulator* à direita.

Na figura 59 é possível ver o IDE *Xamarin Studio* do lado esquerdo e à direita o *iPhone Simulator*. Este último permite executar a aplicação compilada ou outras aplicações do iOS como se se tratasse de um dispositivo real. Estas duas aplicações, juntamente com o *Xcode*, foram o ambiente de desenvolvimento da aplicação desta dissertação.
No *Xamarin Studio*, no bloco à esquerda da aplicação encontram-se os ficheiros da solução. Os ficheiros podem ser organizados dentro de pastas numa solução, permitindo uma maior organização e separação de diferentes tipos de ficheiros.

Como é possível ver, a solução da aplicação foi dividida da seguinte forma:

- SpotClaimer
  + References
  + Components
  + Images
  + Resources
  + Screens
  Info.plist

Na pasta *References* são incluídas todas as bibliotecas que fornecem classes que podem ser usadas para desempenhar diversas funcionalidades na aplicação. Inúmeras bibliotecas podem ser adicionadas para implementar novas funcionalidades na aplicação e ao longo do desenvolvimento da aplicação duas bibliotecas foram adicionadas, *FXBlurView* e *RestSharp*, sendo que apenas a biblioteca *RestSharp* foi usada na versão atual do código.
A biblioteca *FXBlurView* foi adicionada no início do desenvolvimento para criar elementos da interface que fizessem uso de *blur* (desfoque), simulando o efeito introduzido no iOS 7. Esta biblioteca foi desenvolvida para uso em projetos desenvolvidos em Objective-C, ou seja, não era possível ser importada para um projeto desenvolvido em C#. Para tal, foi necessário adquirir conhecimentos sobre *binding* de bibliotecas de Objective-C para C# de modo a que fosse possível utilizar as classes fornecidas por essas bibliotecas ao desenvolver em C#. O vídeo intitulado "Binding Third-Party Objective-C Libraries" de Pierce Boggan foi uma grande ajuda para esta aprendizagem, auxiliando na leitura da documentação fornecida pelos criadores do Xamarin sobre *binding* de bibliotecas de Objective-C. Após sucesso ao criar elementos que fizessem uso do efeito de desfoque verificou-se que ao mover esses elementos sobre um fundo fixo o efeito não era aplicado em tempo real, criando alguns artefatos visuais. Decidiu-se assim abandonar o uso da biblioteca. Não foi possível identificar se o fato de o efeito não ser aplicado em tempo real se deveu ao *binding* efetuado, no entanto, tecnicamente o *binding* não deve ter efeito no desempenho da biblioteca.

A biblioteca *RestSharp* foi adicionada após ser necessário começar a efetuar pedidos Web na aplicação e de desserializar o conteúdo devolvido pelas respostas em JSON para instâncias de classes em C#, ou seja, tornar o conteúdo devolvido pelos pedidos em objetos que pudessem depois ser utilizados para representar esse conteúdo visualmente na aplicação e modificá-lo quando necessário. A biblioteca *RestSharp* permite implementar funcionalidades numa aplicação para que esta se comporte como um serviço REST (*Representational State Transfer* - Transferência de Estado Representativo), sendo que a *framework Django* também dá completo suporte a REST. Aplicações baseadas em REST, normalmente denominadas por RESTful, são aplicações que usam pedidos através do protocolo HTTP (*Hypertext Transfer Protocol* - Protocolo de Transferência de Hipertexto) para fazer quatro tipos de operações: ler, criar, atualizar e eliminar dados. Estas operações são feitas sobre as base de dados que contêm a informação dessas aplicações. Os pedidos REST podem ser efetuados através de URLs que podem conter algum conteúdo enviado, como por exemplo no caso de um

pedido para registar um utilizador com o método POST onde são enviados os dados do utilizador a registar.

A pasta *Components* inclui todos os componentes instalados através da funcionalidade *Xamarin Components* disponível no *Xamarin Studio* através do menu "Projecto -> Get More Components…".

Na pasta *Images* estão presentes todas as imagens utilizadas pela aplicação, quer sejam imagens de fundo, ícones ou imagens para botões. Apenas as imagens incluídas nesta pasta podem ser utilizadas no código e aquando da compilação da aplicação são colocadas na raiz do projeto. Esta pasta permite assim a organização das imagens separadas dos restantes conteúdos do projeto e separadas, caso o programador opte, por subpastas.

Em *Resources* são colocadas as imagens utilizadas para o ícone da aplicação em várias zonas do iOS, sendo que a mais importante é o ícone da aplicação no ecrã inicial. Outros ícones relevantes são o ícone da aplicação na aplicação *Settings* e na App Store. Devem ser utilizados vários ícones para diferentes dispositivos e resoluções de ecrãs, neste caso, ícones para iPad, iPhone e iPod Touch e para as diferenças entre os ecrãs com maior densidade de pixels por polegada, como por exemplo os ecrãs Retina. Além dos ícones da aplicação devem ser colocados nesta pasta as imagens que são utilizadas quando a aplicação é executada e não está em memória. Estas imagens são normalmente utilizadas para dar uma sensação de rapidez ao carregar a aplicação, mostrando algum grafismo da aplicação, ou então para dar início a uma animação enquanto a aplicação é carregada. Assim como os ícones, também devem existir várias imagens de lançamento da aplicação para dispositivos diferentes. Ao utilizar o *Xamarin Studio* estas imagens são automaticamente adicionadas à pasta *Resources*, com o nome utilizado pelo iOS para fazer a detecção das imagens para os diferentes dispositivos. A adição destas imagens é efetuada ao preencher alguma das áreas presentes no ficheiro Info.plist, o qual será descrito mais à frente.

A pasta *Screens* foi criada por necessidade de organização de conteúdos dentro da solução. Nesta pasta são colocados todos os ecrãs ou vistas criados que são necessários para a visualização de conteúdos e navegação entre os mesmos. Os ficheiros com a extensão ".cs" são os ficheiros em que é colocado todo o código do controlador da vista, sendo que também pode ser colocado o código que cria todos os elementos visuais inerentes a uma vista. Caso se opte por utilizador o *Xcode* para implementar a interface visual da aplicação serão utilizados os ficheiros com a extensão ".xib" que sincronizam as alterações realizadas no *Xcode* com o *Xamarin Studio*. No caso da aplicação desenvolvida optou-se por criar os elementos da interface totalmente através de código uma vez que permitia maior liberdade e reduzia a possibilidade de ocorrerem erros ao ligar os elementos visuais às suas funcionalidades.

Ao abrir o ficheiro *Info.plist* no *Xamarin Studio* temos acesso a uma interface que permite personalizar vários aspetos da aplicação entre os quais se destacam:
– nome e versão da aplicação;
– dispositivos alvo (iPhone/iPod, iPad ou Universal);
– versão do iOS a que se destina;
– orientações do dispositivo suportadas;
– o estilo da *status bar* e se esta deve estar escondida aquando do carregamento da aplicação;
– ícones para iPhone, iPad, *Spotlight* e *Settings*;
– imagens utilizadas aquando do carregamento da aplicação;
– integração com os *Maps* do iOS;
– funcionalidades de *background* (código executado quando a aplicação não é a aplicação a ser usada em primeiro lugar).

Em vez de utilizar a interface disponível, o programador pode optar por editar o ficheiro diretamente. Este consiste num dicionário com uma lista de propriedades e o seu valor. Ao ficheiro podem ser adicionadas e removidas propriedades e pode ser também alterado o seu valor.

O ficheiro *Reachability.cs* contém funções para detectar a conectividade de uma aplicação à Internet. Este ficheiro é de grande utilidade pois permite ao programador avisar o utilizador quando a aplicação não possui conectividade à Internet. Este ficheiro foi obtido através do repositório *monotouch-samples* disponível em "https://github.com/xamarin".

SISTEMA DE CONTROLO DE VERSÕES *GIT & BITBUCKET*

Uma vez que esta dissertação consiste no desenvolvimento de uma aplicação móvel seria crucial desenvolver o código para a sua implementação. Sentiu-se a necessidade de fugir dos métodos convencionais de desenvolver todo o código localmente onde não há controlo sobre as versões ao longo do tempo e onde não há um registo do que foi alterado em cada versão, o que impossibilita por exemplo corrigir um problema que foi introduzido há algum tempo atrás. Para solucionar este problema recorreu-se ao *software open source* de controlo e revisão de versões *Git* e à plataforma online *Bitbucket*.
Com o *Git* é possível criar repositórios que armazenam e gerem todo o código de um *software*. É a ferramenta ideal para desenvolvimento de código entre equipas multidisciplinares onde várias pessoas contribuem para o mesmo código, sendo que várias pessoas podem editar os mesmos ficheiros e juntar as alterações posteriormente.
No *Git* cada projeto é armazenado num repositório que contém o registo de todas as alterações efetuadas. Estas alterações são guardadas por ordem cronológica e o seu conjunto representa um ramo (*branch*), sendo que cada repositório pode conter vários ramos com diferentes objetivos. É bastante comum existir um *branch* para o desenvolvimento de cada funcionalidade, outro *branch* para o código mais atual mas não estável (*branch master*) e outro *branch* com o código estável que será lançado para produção.
O nome dos *branches* é definido quando são criados e normalmente descreve sumariamente do que se trata. Inicialmente apenas existe o *branch master* e qualquer *branch* é criado sempre a partir do *branch* em que se está atualmente, ou seja, um *branch* criado quando se está a trabalhar no *branch master* irá conter o registo de todas as alterações do *master* até ao momento da sua criação.
Cada alteração no código, como por exemplo a *correcção* de uma funcionalidade ou a implementação de uma nova, deve ser registada pelo programador através de um comando *git*: o comando *commit*. Um *commit* contém uma mensagem que descreve o que foi realizado e regista as alterações que foram feitas nos ficheiros modificados desde o último *commit*. Estas alterações consistem na adição, remoção ou alteração de linhas de código ou ficheiros.
Cada *branch* num repositório pode conter inúmeros *commits*, sendo comum criar um novo *branch* por tarefa a ser desenvolvida para não afetar os restantes e para haver separação entre código estável e código a ser implementado. Várias pessoas podem trabalhar no mesmo *branch*, sendo que cada um dos seus *commits* é sempre comparado com os anteriores. Se existirem alterações nos mesmos ficheiros em *commits* de pessoas diferentes pode existir um conflito, no entanto esse conflito pode ser facilmente resolvido através do histórico das alterações e do que deve ser mantido de cada um dos *commits* de cada uma das pessoas. Quando uma funcionalidade ou correção se encontra estável pode ser adicionada ao *branch* principal ou a qualquer outro *branch*, através do

comando *merge*. Ao ser realizado um *merge* podem também ocorrer conflitos caso os mesmos ficheiros tenham sido alterados nos dois *branches* a juntar. Isto acontece porque embora um *commit* possa ser mais recente que o outro há a necessidade de manter o código alterado em ambos os *commits*, uma vez que as alterações podem ter propósitos diferentes. Para resolver os conflitos durante um *merge* devem ser revistos todos os ficheiros onde estes aconteceram. Em cada um dos ficheiros são assinalados com *tags* os blocos de código que causaram o conflito, sendo que neste caso se deve optar pelo código correto, eliminando o bloco incorreto e as *tags* que identificam o conflito.

Inicialmente, foi usada no projeto a metodologia de separar o código desenvolvido em vários *branches*. Foram criados três *branches*: um com o nome *ios_app* que incluía todo o código da aplicação móvel; o *branch master* que continha o código do projeto *Django*; e o *branch production*, criado para fazer as alterações necessárias ao projeto *Django* de modo lançar esse código no servidor remoto. Embora esta metodologia tivesse o seu potencial por separar áreas de desenvolvimento diferentes, não se revelou eficaz pois o projeto foi apenas desenvolvido pelo mestrando e foi necessário várias vezes desenvolver código em paralelo e lançar esse código para o servidor. Foi também bastante comum o desenvolvimento de uma funcionalidade na aplicação móvel necessitar de alterações no projeto *Django*. Devido a este desenvolvimento em paralelo nestas diferentes áreas optou-se por desenvolver o código num só *branch*, o *branch production*.

Ao usar o *Git* instalado na máquina de desenvolvimento pode manter-se tudo localmente. No entanto, é comum colocar o repositório numa plataforma online que tenha como intuito armazenar projetos que usem controle de versões. Existem várias plataformas que fornecem este tipo de serviço de armazenamento, entre as quais se destacam o *GitHub* (https://github.com), que aceita projetos que usem *Git*, e o *Bitbucket* (https://bitbucket.org) que suporta *Mercurial* e, desde 2011, *Git*. A escolha pelo *Bitbucket* deveu-se a este permitir o armazenamento de projetos privados sem um custo associado.

Ao usar a plataforma online *Bitbucket* é possível colocar online o código e todos os *commits* feitos localmente através do comando *git push*. No *Bitbucket* existem várias funcionalidades como a pesquisa do código mais atual por *branch*, a consulta da lista de todos os *commits*, na qual é possível ver cada um individualmente, juntamente com as alterações efetuadas no mesmo, a lista dos *branches* existentes e um *issue tracker* que permite adicionar problemas ou funcionalidades a desenvolver como tarefas.

Para evitar a linha de comandos e facilitar o processo de efetuar commits e de colocar o código no servidor foi utilizada a aplicação *GitHub* disponível em https://mac.github.com. Esta aplicação permite também visualizar os *branches* e o histórico dos *commits* por *branch* e, além disso, gerir vários repositórios (figura 60).

O uso de *Git* e a colocação do código online no *Bitbucket* é crucial para que o código possa ser descarregado em várias máquinas e num servidor quando é feito o lançamento (*deploy*) do serviço *web* que suporta a aplicação.

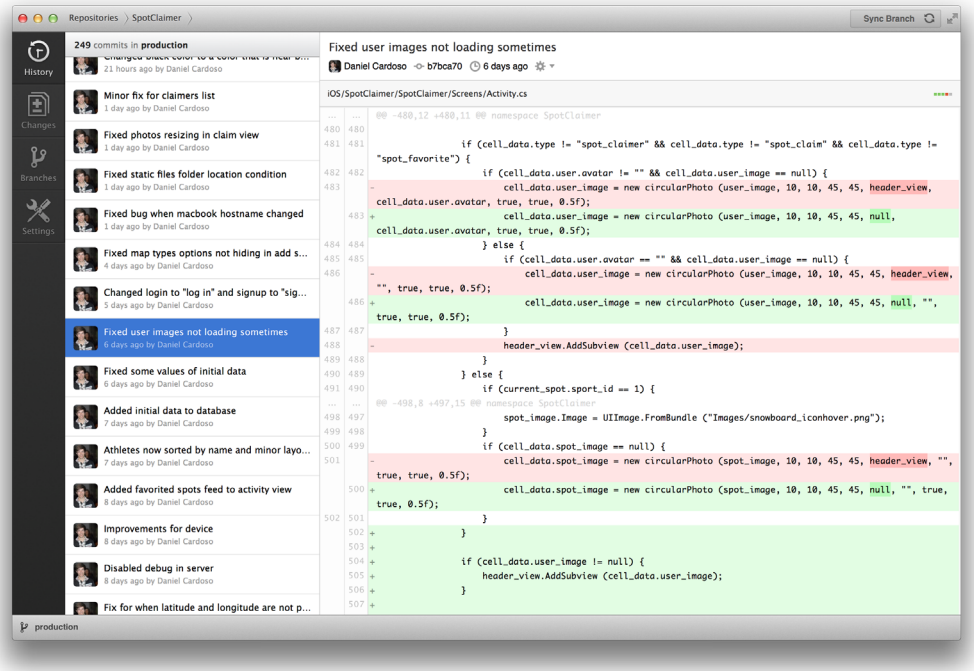


Figura 60. Aplicação *Github* para gestão de repositórios *Git*.

VIRTUAL ENVIRONMENTS PARA PROJETOS *DJANGO*

Um projeto *Django* depende essencialmente de uma instância de Python instalada no sistema e de módulos nativos ou de terceiros. Sendo que num ambiente de desenvolvimento podem existir vários projetos de *Django* que façam uso da mesma instância de Python mas que tenham dependências diferentes é necessário criar um ambiente dedicado a cada um dos projetos, de modo a que uma alteração nas configurações de um destes não afete a integridade e funcionalidade dos restantes. Projetos diferentes também podem necessitar de versões de módulos diferentes ou de uma versão do Python diferente. Um *virtual environment* (ambiente virtual) permite criar essa separação entre projetos. Para criar um *virtual environment* em OS X foram efetuados os seguintes passos:

1. instalar o *Pip*, uma ferramenta que permite instalar módulos de Python, através do comando "sudo easy_install pip". Isto instala o *Pip* em todo o sistema, ficando assim acessível em qualquer local;
2. instalar o *virtual environment* com o comando "sudo pip install virtualenv";
3. criar um diretório onde se irá colocar o código do projeto *Django* desenvolvido, navegando posteriormente para esse diretório na linha de comandos;
4. criar um *virtual environment* para o projeto atual com o comando "virtualenv nome_da_pasta/".

Após efetuar os passos anteriores é criado um *virtual environment* na pasta do projeto e pode-se verificar que esta pasta contém uma instância do Python e de todos os seus módulos, sendo esta a instância usada daí em diante para o projeto *Django*. Sempre que se pretende executar qualquer funcionalidade do projeto ou instalar novas dependências o *virtual environment* deve ser ativado, senão os novos módulos serão instalados em todo o sistema e não apenas no *virtual environment*. Para ativar o *virtual environment* usa-se o comando "source pasta_virtualenv/bin/activate". Para visualizar uma lista dos módulos instalados no v*irtual environment* pode-se usar o comando "pip freeze" e para instalar novos módulos é usado o comando "pip install nome_modulo".

AQUISIÇÃO E CONFIGURAÇÃO DE UM SERVIDOR VPS *(VIRTUAL PRIVATE SERVER)*

De modo a poder testar a aplicação num dispositivo físico era necessária a comunicação com um servidor remoto endereçado num IP (*Internet Protocol* - Protocolo de Internet) fixo, uma vez que ao usar um servidor local seria necessário configurar a aplicação para utilizar o endereço IP do computador que executava o servidor local e o dispositivo móvel teria que estar ligado à mesma rede que o computador. A aquisição de um servidor permitia também ao dispositivo comunicar com o servidor mesmo usando dados móveis e tendo a liberdade de poder utilizar a aplicação em qualquer lugar onde existisse cobertura da rede móvel. Decidiu-se assim pela aquisição de um VPS, um servidor virtual privado.

Um VPS é um servidor que possui máquinas virtuais independentes, cada uma com processos independentes das restantes e acesso de administrador. Neste caso, o que é alugado não é o servidor em si mas sim uma das máquinas virtuais existentes. Cada máquina virtual tem memória e recursos do processador alocados para cada um dos utilizadores, não afetando assim o uso destes recursos as operações das restantes máquinas virtuais existentes. Existem vários sites online que alugam VPS tendo a escolha caído pelos VPS fornecidos pela empresa *Iniz* no site http://iniz.com.

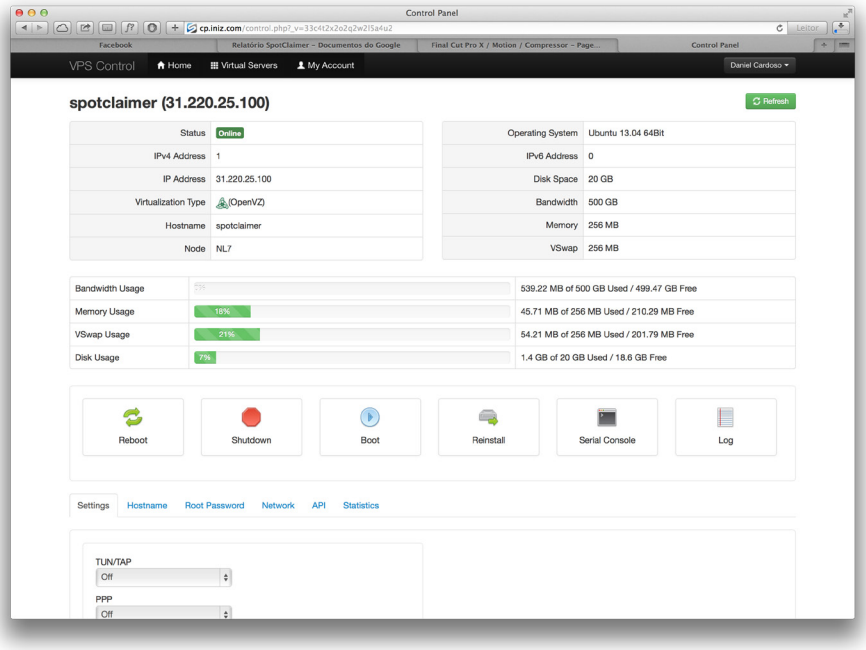


Figura 61. Painel de controlo do VPS.

A máquina virtual adquirida dispõe de 2 cores virtuais, 20 GB de disco rígido, 256 MB de memória RAM e é permitido um tráfego de 500 GB por mês. Este é o plano de custo mais baixo fornecido pela *Iniz* e tem o custo de 3 libras esterlinas por mês. O servidor que contém a máquina virtual adquirida está situado na cidade de Amesterdão, Países Baixos.

Após a aquisição de uma máquina virtual é-nos dada a opção de escolher qual o sistema operativo que pretendemos que seja instalado. Todos os sistemas disponíveis têm como base Linux e optou-se pela distribuição Ubuntu. Assim que o processo de configuração automático é completado somos notificados por *email* com os dados de *login* para um painel de controlo onde podemos gerir as máquinas virtuais alugadas.

Assim que entramos no painel de controlo é-nos apresentada uma lista dos servidores virtuais que temos associados à nossa conta. Neste caso prático apenas foi adquirido um servidor, aparecendo apenas esse na lista. Ao selecionar o servidor virtual existente somos direcionados para uma página que se trata de um painel de controlo com toda a informação do servidor virtual e as opções de configuração disponíveis.

O painel de controlo, demonstrado na imagem acima, indica no título, entre parênteses, o IP ao qual se deve aceder para comunicar com o servidor. É este o IP que será usado pela aplicação para efetuar todos os pedidos quando executada num dispositivo físico. Na primeira tabela são indicadas algumas propriedades do servidor como o seu estado atual, tipo de virtualização, sistema operativo, entre outras. Na segunda tabela é-nos dada informação atual do estado do servidor. Essa informação inclui o tráfego já utilizada, a memória RAM e memória *vSwap* usada e o espaço ocupado do disco. Estes valores são dados em percentagens e em valores reais. Além disso, nesta página também é possível efetuar as seguintes operações no servidor: reiniciar, desligar ou iniciar a máquina virtual, reinstalar o sistema operativo da máquina virtual a partir de uma lista de sistemas que é apresentada após selecionar esta opção, iniciar uma sessão *ssh* através de uma linha de comandos e abrir o registo de alterações efetuadas nas configurações do servidor. Na área ao fundo da página é possível alterar algumas configurações como o *hostname*, a *password* de administrador e ver estatísticas de um determinado período de tempo que pode ir desde uma hora a um ano.

Para configurar o servidor foi necessário recorrer a uma ligação por *ssh* que permite comunicar de forma segura com o servidor através de linha de comandos com um cliente *ssh*. Ao usar uma ligação *ssh* toda a comunicação existente entre o cliente e o servidor é encriptada. De modo a poder efetuar a ligação por *ssh* é primeiro necessário criar uma sessão de *ssh* no servidor e isso é realizado através da opção Serial Console disponível no painel de controlo. O tempo durante o qual a sessão criada se encontra aberta pode ir desde 1 a 8 horas, sendo essa duração escolhida aquando da criação da mesma. Após a criação da ligação é carregada uma página com os dados que nos permitem ligar ao servidor através de um cliente de *ssh*. No caso do sistema operativo OS X é possível efetuar ligações por *ssh* através do terminal, uma vez que este já contem um cliente de *ssh* integrado.

Ao efetuar a ligação com o cliente *ssh* através do terminal temos acesso ao servidor como utilizador *root* (administrador) e os comandos realizados na linha de comandos do computador passam a ser executados no servidor, ou seja, o terminal funciona como se se tratasse de uma linha de comandos executada no servidor, aceitando neste caso comandos de Linux.

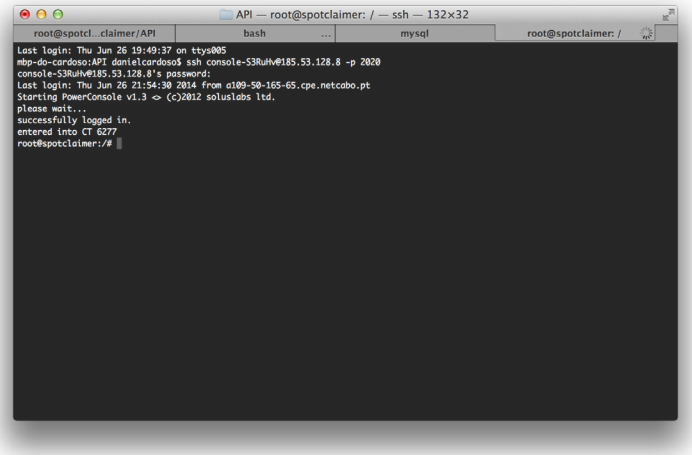


Figura 62. Ligação com sucesso ao VPS por *ssh*.

Assim que se efetuou a ligação com sucesso ao servidor pela primeira vez deu-se então início à configuração do servidor para permitir que as páginas fossem servidas através do *Django*. Começou-se por criar o diretório "home/django-projects/spotclaimer/" que servia para armazenar todo o código desenvolvido em *Django* presente na plataforma *Bitbucket*. Para ir buscar o código ao *Bitbucket* foi instalado o *Git* no servidor e fez-se clone do repositório *SpotClaimer*, ou seja, criou-se um repositório local com origem no repositório online. Após ter o código disponível localmente no servidor criou-se um *virtual environment* e instalaram-se as dependências necessárias, tal como foi realizado do lado do computador usado para desenvolvimento local. De seguida criou-se a base de dados através da ligação ao processo de *mysql* existente no servidor e depois foram criadas as tabelas através do comando "python manage.py syncdb".

Assim que foram concluídas as tarefas anteriores o servidor tinha a mesma configuração que o ambiente para desenvolvimento local. No entanto, o servidor de *Django* não é um servidor indicado para produção, existindo melhores alternativas, quer a nível de performance como a nível de personalização e robustez. No caso da máquina virtual usada, o servidor disponível era Apache. Era então necessário que o servidor de Apache comunicasse com a aplicação de *Django*. Para isso recorreu-se ao módulo *mod_wsgi* que permite servir aplicações *web* de *Python* através de um servidor *Apache*. Para ajudar nos passos necessários recorreu-se ao tutorial escrito por Ayman Farhat no seu blog "The Code Ship", disponível em "http://thecodeship.com". Após algumas dificuldades encontradas no processo de configuração dos ficheiros necessários e algumas alterações necessárias ao ficheiro *settings.py,* o qual contém as configurações da aplicação de Django, conseguiu-se aceder ao primeiro URL através de um *browser*. Assim, a API desenvolvida passou a estar disponível para ser acedida através de qualquer dispositivo, nomeadamente do dispositivo físico utilizado para testar a aplicação.

A configuração aqui descrita foi realizada numa fase intermédia do desenvolvimento da aplicação em que era necessário começar a testar a estabilidade e funcionalidade da aplicação num dispositivo real. Nesta fase algumas comunicações com o servidor já estavam implementadas na aplicação e no código da aplicação *Django*.

IMPLEMENTAÇÃO EM *DJANGO*

Neste subcapítulo será dado mais detalhe à implementação do projeto em *Django*, abordando alguns algoritmos importantes para algumas das funcionalidades implementadas. Uma vez que uma grande parte do projeto desenvolvido nesta dissertação consistiu em implementação é necessário descrever parte desse processo, utilizando assim uma linguagem mais técnica. Começará por ser descrito o projeto *Django* que deu origem à API que dá suporte à camada de comunicação da aplicação desenvolvida em C#.
No subcapítulo seguinte será descrita a implementação da aplicação móvel em C#.

Após a criação do modelo físico apresentado no capítulo 6, foi gerado código SQL (*Structured Query Language* - Linguagem de Consulta Estruturada) para criar a base de dados correspondente ao modelo. Este código é gerado pelo *software* usado para criar o modelo físico, o *MySQLWorkbench*, através da opção "Forward Engineer" no menu "Database". Embora o código SQL gerado permita criar uma base de dados, o objetivo aqui foi ter uma referência para a criação dos modelos de *Django*, pois são estes que vão conter o código necessário para criar a base de dados de modo a esta ser usada pela API do *Django*.

Após a configuração e inicialização do *virtual environment* e consequente instalação dos módulos necessários através do comando "pip install -r requirements.txt", foi criado um projeto de *Django* com o nome "spotclaimer" através do comando "django-admin.py startproject spotclaimer". O ficheiro "requirements.txt" inclui linhas com as referências dos módulos e das versões de cada um, os quais podem ser instalados de uma só vez. As linhas seguintes foram incluídas no ficheiro:

```
- Django==1.6.2
- django-extensions==1.3.3
- MySQL-python==1.2.5
- Pillow==2.4.0
```

Como foi referido na introudção à *framework Django*, cada modelo de *Django* faz parte de uma *App* de *Django*. Para implementar o código necessário para gerar a base de dados a partir dos modelos foram então criadas várias aplicações, sendo que cada modelo de cada aplicação iria consistir em uma ou duas tabelas da base de dados. As aplicações criadas com o comando "python manage.py startapp nome_app" para dar suporte aos modelos foram as seguintes:

– sports_app;
– spots_app;
– users_app;
– claim_app;
– multimedia_app;
– feed_app.

Além destas aplicações foi também criada a aplicação "common" que foi utilizada para implementar código usado em várias partes do projeto.

Antes de proceder à criação das tabelas é necessário criar a base de dados. De maneira a dar suporte ao uso de *emoticons*, a base de dados criada usa o *character set* "UTF8mb4" e a *collation* "utf8mb4_bin". A base de dados é criada através do seguinte comando numa consola de *mysql*: "create database scdb character set UTF8mb4 collate utf8mb4_bin;". A palavra "scdb" consiste no nome atribuído à base de dados e é uma abreviatura de "SpotClaimer Database".

Assim que era criada cada *App*, era implementado o código nos modelos para gerar a base de dados, o processo era o seguinte:

– analisar o bloco de código SQL da criação de uma tabela, por exemplo:

```
“
 CREATE TABLE IF NOT EXISTS ‘mydb’.’SPORT’ (
   ‘sport_id’ INT NOT NULL,
   ‘name’ VARCHAR(45) NULL,
   ‘description’ VARCHAR(200) NULL,
   ‘picture’ VARCHAR(200) NULL,
   PRIMARY KEY (`sport_id`))
   ENGINE = InnoDB;
 ”;
```

– implementar o código python no ficheiro “models.py” da App tendo como referência a documentação dos modelos (Models - Django documentation, 2014) e a documentação dos tipos de campos nos modelos (Model field reference - Django documentation, 2014):

```
“
  from django.db import models
  class Sport(models.Model):
    name = models.CharField(max_length=45, unique=True)
    description = models.CharField(max_length=200, blank=True)
    picture = models.ImageField(upload_to=”site_media/
sports”,blank=True, null=True)
”;
```

– executar o comando “python manage.py syncdb” para criar as tabelas da base de dados com base no código dos modelos.

Após a criação de todas as tabelas existentes no modelo físico deu-se início à implementação das vistas, as quais contêm toda a lógica necessária para devolver, inserir, eliminar e atualizar dados na base de dados. Tal como o código dos modelos, o código das vistas é separado por *Apps* de *Django*. Por exemplo a função que devolve uma lista com os desportos existentes na base de dados está implementada no ficheiro “views.py” da App “sports_app”.

As vistas são executadas pelo *Django* ao processar um pedido a um url e devem sempre devolver algum objeto ou variável. Cada url definido no ficheiro “urls.py” do projeto tem de incluir qual a função das vistas que deve ser executada, assim, ao aceder a um URL num *browser* ou através de uma aplicação essa função será executada e será apresentado no *browser* ou na aplicação o conteúdo devolvido pela vista. Como se pretendia implementar uma plataforma *web* baseada em REST, o conteúdo enviado para o servidor e as respostas devolvidas pelas vistas estão sempre no formato JSON.
O formato JSON é bastante leve, simples de ler e mais fácil de analisar por um *software* analisador em comparação com XML, por exemplo.

Os URLs e funções associadas no ficheiro "urls.py" são os seguintes:

```
(POST) "api/login/" – "email_login_view";
(POST) "api/logout" – "logout_view";
(POST) "api/createuser/" – > "create_user";
(GET) "api/user/" – "view_user";
(GET) "api/user/feed/" – "user_feed";
(GET) "api/users/" – "view_users";
(POST) "api/user/follow/" – "start_following";
(POST) "api/user/unfollow/" – "stop_following";
(GET) "api/feed/" – "get_feed";
(GET) "api/sports/" – "view_sports";
(POST) "api/addsport/" – "add_sport";
(POST) "api/spots/addspot/" – "add_spot";
(GET) "api/spots/" – "view_spot";
(GET) "api/spot/claims/" – "view_spot_claims";
(POST) "api/spot/claim/" – "create_claim";
(POST) "api/spot/favorite/" – "make_favorite";
(GET) "media/" – "django.views.static.serve".
```

Na lista acima é possível ver, entre parênteses, o método que é necessário usar para que a vista seja processada. O método GET é usado quando se quer aceder a dados sem efetuar nenhuma alteração. Para alterar, adicionar e remover dados é usado o método POST. Por defeito, ao aceder a um URL o método usado é o método GET. Para usar o método POST o pedido deve ser configurado para tal e a função que processa o pedido também.

Embora quase todos os dados disponíveis na base de dados sejam públicos, a API desenvolvida só os disponibiliza a utilizadores logados. Para verificar se o utilizador está logado e definir se a função usa o método POST ou não foi implementado um decorador de *Django* que permite executar funcionalidades antes da função que irá processar o URL.

O decorador de *Django* usado aceita dois parâmetros: "login_needed" e "post_required". Por defeito o parâmetro "login_needed" está definido como *true* e o parâmetro "post_required" como *false*. Os seguintes exemplos mostram o uso de decoradores de *Django* em funções implementadas:

```
"
@login_post(login_needed=True, post_required=False)
def view_sports(request, sport_id=None):
";
"
@login_post(login_needed = False, post_required= True)
def create_user(request):
";
```

O decorador é definido pelo símbolo "@" e chama a função "login_post" antes da função que é processada ao aceder a um URL. No primeiro exemplo, para obter uma lista de desportos através da função "view_sports" é necessário que o utilizador esteja logado, no entanto, o pedido efetuado deve utilizar o método GET. No segundo exemplo, quando se pretende criar

um utilizador, o utilizador não pode estar logado uma vez que ainda não se registou, sendo por isso definido o parâmetro "login_needed" como *false*. No entanto, como se pretende inserir dados na base de dados ao criar um novo utilizador o método do pedido deve ser POST, daí a variável "post_required" estar a *true*.

Além do decorador "login_post" foi usado um outro decorador fornecido pelo *Django*, o decorador "@ensure_csrf_cookie". Este decorador força que todas as funções devolvam o cookie "csrftoken". O *cookie* "csrftoken" é usado por questões de segurança, sendo que deve ser sempre enviado para o servidor nos pedidos futuros o *cookie* que foi enviado pelo servidor e guardado num dos pedidos anteriores. Este *cookie* é específico para cada utilizador e é gerado um novo sempre que o utilizador efetua *login* na plataforma.

"api/login/"

Ao aceder a este URL é processado o *login* de um utilizador. Para se processar o *login* é necessário que sejam enviados dois parâmetros para o servidor através do método POST. Os parâmetros necessários são o *email* e a *password* associados à conta com que o utilizador se registou na aplicação.

Para autenticar o utilizador é usado o método "authenticate" do *Django*, pertencente ao módulo "django.contrib.auth". Este método recebe dois parâmetros, *username* e *password*. O *username* é obtido através do registo na base de dados do utilizador com o parâmetro *email* enviado no pedido.

Caso o utilizador exista e o *email* e *password* enviados nos parâmetros correspondam é efetuado o *login* com sucesso e é enviada uma resposta de sucesso, caso contrário é enviada uma resposta de erro avisando o utilizador de que a combinação está incorreta. O *login* consiste numa sessão de utilizador mantida através de um *cookie* com o nome "sessionid". Este *cookie* deve ser guardado na aplicação e deve ser enviado ao servidor sempre que é efetuado um pedido. O servidor ao receber um pedido compara o *cookie* com um registo guardado na base de dados aquando do *login* com sucesso para verificar se o *login* ainda se mantém ativo.

"api/logout/"

A função "logout_view" associada ao URL processa o *logout* do utilizador atualmente logado. O método do pedido efetuado deve ser POST e a função apenas executa o método *logout* que aceita como parâmetro o pedido efetuado, sendo que este método também pertence ao módulo "django.contrib.auth". O método *logout* destrói a sessão atual do utilizador e após ser executado o método é devolvida uma resposta de sucesso.

"api/createuser/"

Este URL deve ser acedido quando se pretende registar um novo utilizador. Os parâmetros que devem ser enviados obrigatoriamente são: o *username*, *email*, *first_name*, *last_name* e *password*.
A função "create_user" executa o seguinte algoritmo:

```
função "create_user":
  - tenta:
    - valida o email
  - excepção:
    - devolve resposta de erro, email inválido
  - se número de utilizadores com o email enviado > 0:
    - devolve resposta de erro, email já existe
  - se número de utilizadores com o username enviado > 0:
    - devolve resposta de erro, username já existe
  - senão:
    - cria novo objeto os dados do utilizador
    - autentica o utilizador
    - devolve resposta de sucesso
```

"api/user/"

Este URL devolve os detalhes do utilizador atual ou de qualquer utilizador registado na plataforma. O método usado para aceder a este URL deve ser o método GET. A função "view_user" é executada caso receba o parâmetro opcional "user_id" com o *id* de um utilizador e devolve os dados desse utilizador, caso esse parâmetro não seja enviado são devolvidos os dados do utilizador atualmente logado.

"api/user/feed/"

Ao aceder a este URL é apresentado o registo de atividade de um utilizador. Tal como na função "view_user", a função "user_feed" aceita o parâmetro opcional "user_id" e processa o registo de atividade do utilizador logado caso o parâmetro não seja enviado ou de outro utilizador caso o *id* seja enviado num pedido GET.

O registo de atividade de um utilizador é gerado através do seguinte pseudocódigo:

```
função "user_feed":
  - cria a lista "feed_list"
  - cria a variável "claims" com os claims do utilizador
  - cria a variável "favorites" com os favoritos do utilizador
  - para cada claim em "claims":
    - adiciona o claim à lista "feed_list"
  - para cada favorito em "favorites":
    - adiciona o favorito à lista "feed_list"
  - ordena a lista "feed_list" pelas datas dos seus elementos
  - humaniza as datas dos elementos da "feed_list"
  - devolve a "feed_list"
```

"api/users/"

A função "view_users" é executada quando este URL é acedido e devolve uma lista dos utilizadores registados na plataforma, excepto o utilizador atualmente logado.

A função executa o seguinte algoritmo:

```
função "view_users":
  - cria a variável "users" com todos os utilizadores excepto o
  utilizador logado
  - cria a lista "context"
  - para cada utilizador em "users":
    - guarda na variável "user_profile" o perfil do utilizador
    - adiciona o perfil do utilizador à lista "context"
  - ordena a lista "context" pelo nome dos utilizadores
  - devolve a lista "context"
```

"api/users/follow/"

Este URL permite ao utilizador logado seguir um utilizador na plataforma. Deve ser enviado o *id* do utilizador que se pretende seguir no parâmetro "followed_id" do método POST. A função "start_following" é chamada ao aceder a este URL e executa o algoritmo descrito de seguida:

```
função "start_following":
  - cria a variável "followed_id" com o id do utilizador a seguir
  - se "followed_id" igual ao id do utilizador logado:
    - devolve resposta de erro, utilizador não se pode seguir a si
    próprio
  - se existe utilizador com o "followed_id":
    - se o utilizador já está a seguir:
      - devolve resposta de erro
    - senão:
      - segue o utilizador
      - devolve resposta de sucesso
  - senão:
    - devolve resposta de erro, utilizador a seguir não existe
```

"api/users/unfollow/"

O algoritmo da função "stop_following", chamada por este url, é bastante semelhante ao algoritmo previamente descrito da função "start_following". Neste caso o utilizador deixa de seguir o utilizador com o *id* enviado no parâmetro "followed_id", sendo assim eliminado o registo da base de dados criado pela função "start_following".

"api/feed/"

Na função "get_feed" associada a este url foi implementado um algoritmo que devolve o registo de atividade do utilizador, dos spots que adicionou aos favoritos e dos utilizadores que segue. Uma versão bastante simplificada do algoritmo é a seguinte:

```
função "get_feed":
  - cria a lista "feed_list"
  - cria a lista "user_claims" com os claims do utilizador logado
  - cria a lista "user_favorites" com os spots favoritos do
  utilizador logado
  - cria a lista "user_claimers" com os spots onde o utilizador
  logado é spot claimer
  - cria a lista "followed_users" com os utilizadores seguidos pelo
  utilizador logado
  - para cada claim em "user_claims":
    - adiciona o claim à lista "feed_list"
  - para cada favorito em "user_favorites:
    - adiciona o favorito à lista "feed_list"
  - para cada claimer em "user_claimers":
    - adiciona o claimer à lista "feed_list":
  - para cada utilizador em "followed_users":
    - cria a lista "user_claims" com os claims do utilizador
    - cria a lista "user_favorites" com os spots favoritos do
    utilizado
    - cria a lista "user_claimers" com os spots onde o utilizador é
    spot claimer
    - para cada claim em "user_claims":
      - adiciona o claim à lista "feed_list"
    - para cada favorito em "user_favorites:
      - adiciona o favorito à lista "feed_list"
    - para cada claimer em "user_claimers":
      - adiciona o claimer à lista "feed_list":
    - para cada spot em "user_favorites":
      - cria a variável "spot_feed" com o registo de atividade do
      spot
    - para cada registo em "spot_feed":
      - adiciona o registo à lista "feed_list"
  - ordena a lista pela data mais recente
  - humaniza as datas dos elementos da "feed_list"
  - devolve a lista "feed_list"
```

"api/sports/"

Ao aceder a este url é devolvida uma lista de todos os desportos inseridos na base de dados. Caso no pedido GET seja enviado um parâmetro com o *id* de um desporto serão devolvidos os detalhes desse desporto.

"api/addsport/"

A função "add_sport" permite adicionar um novo desporto à base de dados, sendo assim um ponto de partida para no futuro adicionar novos desportos à aplicação. A função deve receber dois parâmetros obrigatoriamente, de modo a inserir um novo desporto: o nome e a descrição. Caso já exista um desporto com o mesmo nome, a função devolve uma resposta de erro.

"api/spots/addspot/"

Para adicionar um novo *spot* à base de dados deve ser utilizado este URL. O método do pedido deve ser POST e devem ser enviados no mínimo 7 parâmetros: o *id* do desporto associado, nome do *spot*, país, nome do local (cidade), descrição, latitude e longitude. Ao adicionar um *spot* é verificado se não existe nenhum *spot* associado ao mesmo desporto num raio de 500 metros. O algoritmo para a inserção de um novo *spot* na base de dados é o seguinte:

```
função "add_spot":
  - cria a variável "sport_id" com o id do desporto enviado
  - se número de desportos com o id "sport_id" > 0:
    - cria as variáveis com os valores recebidos nos parâmetros
    - para cada localização na tabela Location:
      - cria booleano "same_sport" a true se o spot associado à
      localização tiver o mesmo sport_id que a variável "sport_id"
      - calcula a distância dessa localização aos valores de latitude
      e longitude enviados
      - se distância < 0.5 km e booleano "same_sport" a true:
        - devolve resposta de erro, existe um spot com o mesmo
        desporto perto
    - se na tabela Location não houver registo com a mesma latitude e
    longitude:
      - adiciona localização à base de dados
      - adiciona spot à base de dados
      - devolve resposta de sucesso
    - senão:
      - devolve resposta de erro, spot já existe
  - senão:
    - devolve resposta de erro, não existem desportos com o id
    submetido
```

"api/spots/"

Ao aceder a este URL é devolvida uma lista com todos os *spots* existentes na base de dados por ordem de proximidade em relação ao utilizador. Caso seja enviado o parâmetro "spot_id" no pedido GET é devolvido o *spot* com o *id* associado e o seu registo de atividade. O algoritmo usado nesta função é o seguinte:

```
função "view_spot":
 - cria a variável "spot_id" caso este tenha sido enviado
 - cria a variável "latitude" caso a localização tenha sido enviada
 - cria a variável "longitude" caso a localização tenha sido
 enviada
 - se "spot_id" existe:
   - se existe spot com o "spot_id" enviado:
     - obtém os dados do spot
     - obtém o registo de atividade do spot
     - devolve os dados do spot e o seu registo de atividade
   - senão:
     - devolve resposta de erro, não existe spot com o id enviado
 - senão:
   - cria lista "spots" com  todos os spots
   - cria a lista "context"
   - para cada spot em "spots":
     - adiciona dados do spot à lista "context"
   - se "latitude" existe:
     - ordena a lista "context" pela distância à latitude e
     longitude
     enviadas
   - senão:
     - ordena a lista "context" pelo spot adicionado mais
     recentemente
   - devolve a lista "context"
```

"api/spot/claims/"

A função "view_spot_claims" associada a este URL seleciona todos os *claims* da tabela *Claim* associados ao *id* de um *spot* enviado no parâmetro "spot_id" e devolve-os numa lista.

"api/spot/claim/"

Quando se quer efetuar um *claim* num *spot* é a este uURLrl que se acede com o método POST. A função "create_claim" insere um novo registo *claim* na base de dados caso o utilizador logado não tenha feito um *claim* no mesmo *spot* nas últimas três horas. Optou-se por não proteger, para efeitos de demonstração, os *claims* nos *spots* no caso do utilizador não se encontrar no *spot*. O algoritmo implementado para criar um novo *claim* é o seguinte:

```
função "create_claim":
  - cria a variável "description" com a descrição enviada
  - cria a variável "date" com a data e tempo atual
  - cria a variável "image" caso tenha sido enviada uma imagem
  - cria a lista "claims" com os claims efetuados pelo utilizador
  logado no spot selecionado:
  - para cada claim em "claims":
    - se a data de claim < "date" - 3 horas:
      - devolve resposta de erro, foi efetuado claim neste spot pelo
      utilizador à menos de 3 horas
  - cria lista "claims_by_user" com os claims por utilizador
  - se "claims_by_user" > 0:
    - para cada claim em "claims_by_user":
      - calcula número de claims por utilizador
      - se os claims do utilizador logado + 1 > utilizador com mais
      claims:
        - utilizador logado é o novo spot claimer
  - senão:
    - utilizador logado é o novo spot claimer
  - cria o claim
  - devolve resposta de sucesso
```

O *spot claimer* é definido pelo utilizador que tiver maior número de *claims* num determinado *spot*. Ao efetuar um novo *claim* é calculado novamente se se deve alterar o *spot claimer* de um *spot*, como demonstrado no algoritmo acima.

"api/spot/favorite/"

Para se marcar um *spot* como favorito ou remover esse *spot* dos favoritos deve-se aceder a este URL com um pedido POST. O algoritmo implementado é bastante simples. É verificado apenas se o *spot* já foi adicionado aos favoritos, sendo removido dos favoritos caso tenha sido adicionado previamente. Caso não esteja nos favoritos do utilizador é então adicionado aos favoritos.

IMPLEMENTAÇÃO EM C#

Neste subcapítulo será descrito o processo de desenvolvimento da aplicação *SpotClaimer* em C#. Serão abordados os ficheiros mais importantes, os quais contêm o código desenvolvido.

Neste capítulo não serão abordados algoritmos uma vez que o código desenvolvido abrange também o código para a interface de utilizador e o protótipo desenvolvido conta com mais de 7000 linhas de código implementado. É, no entanto, importante abordar algumas considerações que se tiveram aquando do desenvolvimento da aplicação e que não foram pensadas durante a fase de desenho. Será também descrita a implementação de algumas funcionalidades.

O desenvolvimento da aplicação móvel com código nativo foi um desafio diferente de todos aqueles com que o mestrando já tinha lidado.

O desenvolvimento em código nativo implica considerações que não são tidas quando se desenvolve uma página *web*, por exemplo. Essas considerações incluem principalmente o cuidado que se deve ter na implementação da comunicação com um servidor. Enquanto que numa página *web* os pedidos são processados através de um recarregamento da página ou da submissão de um formulário, numa aplicação móvel estes pedidos têm que ser programados desde o momento em que é pressionado um botão e os dados são preparados para enviar, até ao momento em que é recebida a resposta e se tem de alterar o estado da aplicação. Isto implica um tempo de desenvolvimento maior em comparação com uma página *web* ou com uma aplicação que assente em tecnologias web. No entanto, tem a vantagem da performance ser maior e de se poder usar todas as ferramentas e funcionalidades disponibilizadas pelas API dos sistemas operativos móveis.

Para dar início ao desenvolvimento começou-se por instalar o IDE Xamarin Studio. Após a instalação do IDE foram testados alguns exemplos disponibilizados na página *web* do Xamarin (Samples - Xamarin, 2014). Estes exemplos, juntamente com o estudo da anatomia das aplicações de iOS (iOS Human Interface Guidelines: iOS App Anatomy, 2014), permitiram uma familiarização com o código e um maior conhecimento sobre os elementos que definem a interface de utilizador.

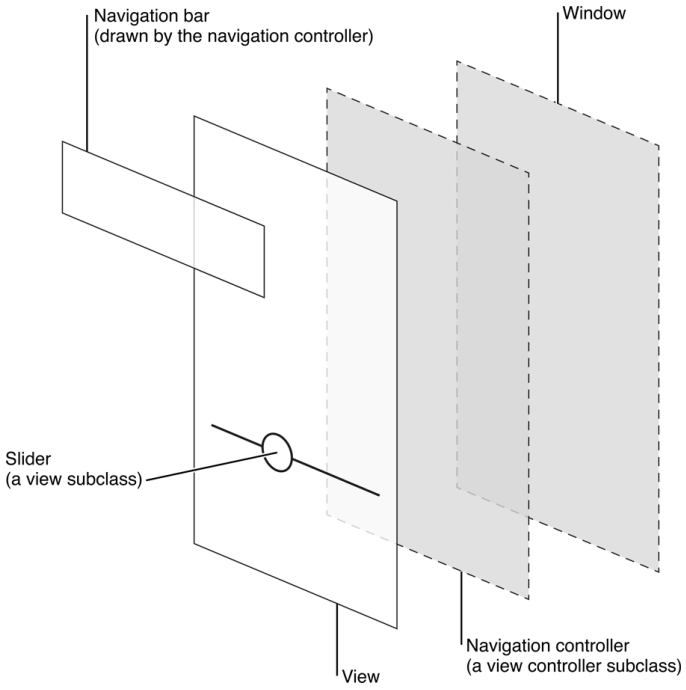


Figura 63. Anatomia de uma aplicação iOS. Retirado da página web "iOS Human Interface Guidelines: iOS App Anatomy".

Numa aplicação iOS, cada elemento da interface de utilizador é considerado uma *view*, uma vez que todos herdam uma *UIView*, que por sua vez faz parte da *framework UIKit*. Uma *view* pode ser desenhada no ecrã e pode-se interagir com a mesma quando se toca no ecrã dentro dos seus limites.

Para conter uma hierarquia de *views*, como se se tratassem de camadas, é necessário usar um *view controller.* Um *view controller* gere como as *views* são apresentadas, implementa a funcionalidade que permite as interações com o utilizador
e gere as transições entre diferentes ecrãs (iOS Human Interface Guidelines: iOS App Anatomy, 2014).

As aplicações iOS assentam no modelo de arquitetura de *software* MVC (*Model-view-controller*) onde o *model* corresponde aos dados a serem mostrados, a *view* é a interface que mostra os dados e o *controller* define como os dados dos *models* são apresentados nas *views*. Isto permite assim separação de conceitos, mantendo o código modularizado e assim mais reutilizável.

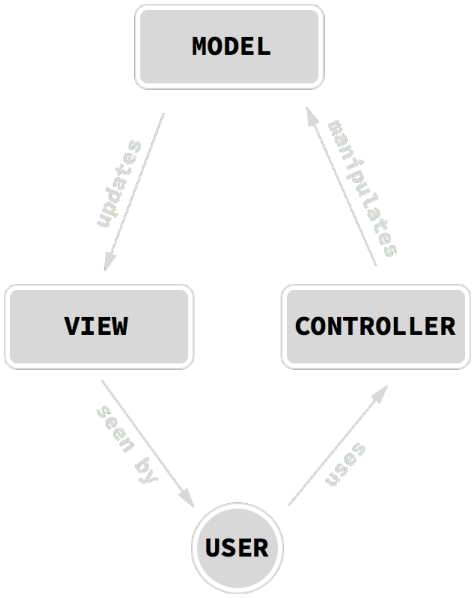


Figura 64. Exemplo do MVC numa aplicação iOS. Retirado da página web "Xamarin: Hello iOS MultiScreen".

O código implementado foi incluído nos ficheiros da lista seguinte. À frente do nome de cada um dos ficheiros está o ecrã ou os ecrãs associados, caso esse ficheiro corresponda à implementação de um ecrã ou mais ecrãs.

– Activity.cs - *User Dashboard*
– Add_Spot_Map.cs - *Add Spot*
– ClaimView.cs - *Check-in (Claim)*
– CustomAlertView.cs
– CustomTableAlertView.cs
– FirstScreen.cs - *Login/Register & Find Nearby Spots, Login User - Options & Email, Register User - Options & Email , Welcome Screen*
– Loading_Screen.cs - *Loading View*
– SideMenu.cs - *Navigation Menu*
– SpotList.cs - *Nearby Spots*
– SpotView.cs - *View Spot*
– UserList.cs - *Nearby Users*
– UserProfile.cs - *User Profile*
– UserView.cs - *View User*
– AppDelegate.cs
– ExtensionMethods.cs
– GlobalVariables.cs
– http.cs
– Main.cs

De seguida serão descritas algumas considerações tidas na implementação do código de cada um dos ficheiros, abordando algoritmos quando necessário. Serão descritos em primeiro lugar os ficheiros que dão suporte à maioria das funcionalidades da aplicação, ou seja, aqueles que não correspondem à implementação de um ecrã. De seguida serão descritos os ficheiros que implementam uma interface de utilizador que pode ser incluída em vários ecrãs e por fim serão descritos os ficheiros que implementam os ecrãs, associando as figuras dos ecrãs aos respetivos ficheiros da implementação.

AppDelegate.cs

Neste ficheiro é inicializada a janela onde a aplicação será executada e é definido o endereço com o qual a aplicação irá comunicar. O endereço é escolhido após verificar onde a aplicação está a ser executada. Caso esteja a ser executada no dispositivo físico o endereço escolhido será o do servidor remoto, caso esteja a ser executada no simulador o endereço será o do servidor local do computador, neste caso o *localhost*. Após a escolha do endereço e a inicialização da janela são criados os contentores que irão conter os ecrãs da aplicação. A necessidade do uso de um contentor que gere vários *view controllers* deveu-se à implementação do menu de navegação, uma vez que seria necessário apresentar dois *view controllers* simultaneamente: o *view controller* que apresenta o menu e o *view controller* que apresenta o ecrã atual redimensionado.

ExtensionMethods.cs

Em *ExtensionMethods.cs* foi colocada uma função que permitisse estender a funcionalidade da biblioteca *RestSharp* para fazer uso das operações da classe *Task* presente na *framework .NET*. Esta classe permite executar código assíncrono, ou seja, código que é executado noutras *threads* em paralelo à que executa o código responsável por gerir a interface visual, evitando assim bloqueios da interface em operações que levam tempo indeterminado, como é o caso dos pedidos Web. O uso de *threading* permite oferecer ao utilizador uma experiência de utilização da aplicação sem bloqueios.

GlobalVariables.cs

No ficheiro *GlobalVariables.cs*, como o próprio nome indica, foram incluídas todas as variáveis globais que são usadas por vários ficheiros dentro da aplicação. Essas variáveis incluem as cores da aplicação, definidas na paleta de cores que são usadas em múltiplos elementos da interface, e as variáveis que contêm o domínio do servidor com que a aplicação deve comunicar, sendo esse domínio escolhido e associado a uma outra variável global aquando da inicialização da aplicação no ficheiro *AppDelegate.cs*, como foi explicado anteriormente.

http.cs

O ficheiro *http.cs* contém quase todas as funções utilizadas para efetuar pedidos ao servidor, ficando apenas de fora as funções que descarregam imagens do servidor através de um URL. No início do ficheiro são declarados todos os objetos da classe *RestRequest*, pertencente à biblioteca *RestSharp*. Estes objetos permitem efetuar todos os pedidos Web, declarando o tipo de método, o URL, os parâmetros e *headers* que devem ser enviados para o servidor. Neste ficheiro também é declarada uma variável com os dados do utilizador que está logado na aplicação, permitindo assim aceder-lhes em qualquer local da aplicação.

As funções incluídas neste ficheiro implementam a execução dos pedidos ao servidor e processam o resultado desse pedido, devolvendo-o posteriormente. Isto permite executar estas funções em qualquer local da aplicação e obter os dados na variável que está associada à chamada destas funções.

Main.cs

O ficheiro *Main.cs* inclui a maioria das classes que implementam elementos da interface desenvolvidos especificamente para a aplicação. Algumas destas classes permitem instanciar elementos como as fotos com forma circular, os botões de *follow* e os botões retangulares dos primeiros ecrãs, isto facilita a criação dos elementos e torna o código mais modular. São também implementadas as classes que permitem instanciar objetos com os dados obtidos do servidor, a classe *Spot*, por exemplo, contém os atributos

que definem um *spot*, sendo que cada um desses atributos tem o nome do atributo recebido do servidor em formato JSON. Estas classes foram criadas com o auxílio da página *web json2csharp* que cria o código C# de uma classe consoante o código JSON introduzido ou um URL que devolva código em formato JSON (json2csharp - generate c# classes from json, 2014).

CustomAlertView.cs

Este ficheiro surgiu após ser detectado que era crucial informar o utilizador quando a aplicação não tinha ligação à Internet. Uma vez que a funcionalidade que detecta a presença de Internet pode ser utilizada em qualquer local da aplicação através do uso da classe *Reachability* presente no ficheiro *Reachability.cs*, o qual foi descrito anteriormente, era necessário que a interface presente neste ficheiro pudesse também ser apresentada em qualquer local da aplicação. Procedeu-se então ao desenho e implementação de um alerta que informasse o utilizador da ausência de ligação à Internet, uma vez que não se pretendia usar o alerta que é apresentado por defeito numa aplicação de iOS pois este necessita da interação do utilizador para desaparecer. Este alerta é apresentado sempre que o utilizador efetua alguma ação que necessita de ligação à Internet e esta não está disponível. O *view controller* onde este alerta é implementado deve ser do tipo *UIViewController*. Além da funcionalidade principal de avisar o utilizador da falta de conectividade à Internet, este alerta permite também avisar o utilizador que alguma funcionalidade não está disponível no protótipo. A figura 65 exemplifica o uso deste alerta quando o utilizador pretende efetuar *login* e não existe conectividade (à esquerda) e quando o utilizador pretende efetuar *login* com a sua conta de Facebook (à direita).

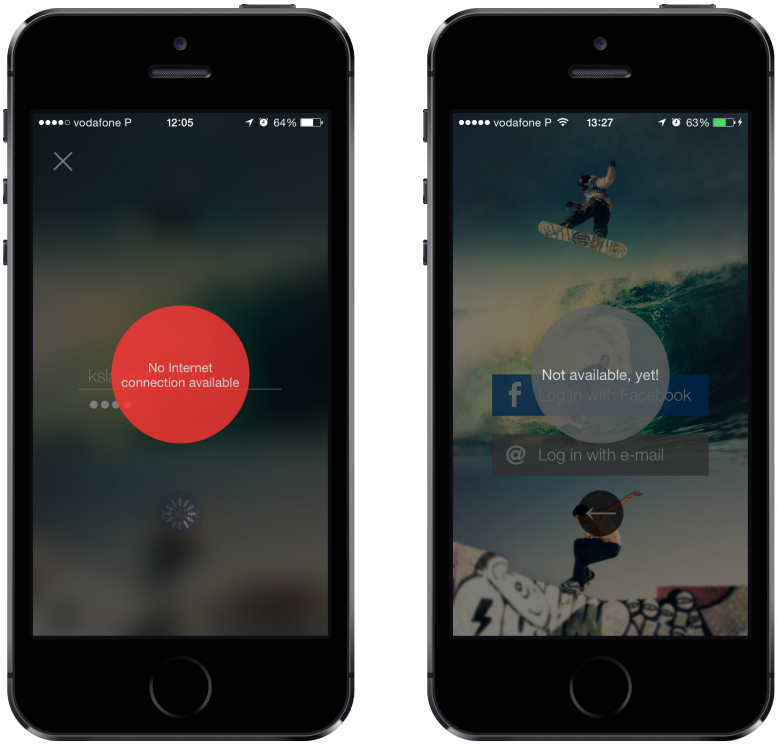


Figura 65.
Exemplo do alerta implementado em *CustomAlertView.cs*.

CustomTableAlertView.cs

Tal como no ficheiro anterior, neste ficheiro é implementado um alerta que avisa o utilizador quando não existe conectividade à Internet, no entanto, este é apresentado quando o *view controller* atual é uma instância da classe *UITableViewController* ou a *view* principal é uma instância de *UITableView*, ou seja, é uma vista que apresenta o conteúdo numa tabela. Além de avisar sobre a falta de conectividade, este alerta também pode avisar o utilizador de alguma ação não permitida. A figura 66 exemplifica o uso deste alerta no ecrã de atividade quando não existe conectividade (à esquerda) e quando um utilizador pretende fazer *claim* de um *spot* que ao qual fez *claim* há menos de três horas (à direita).

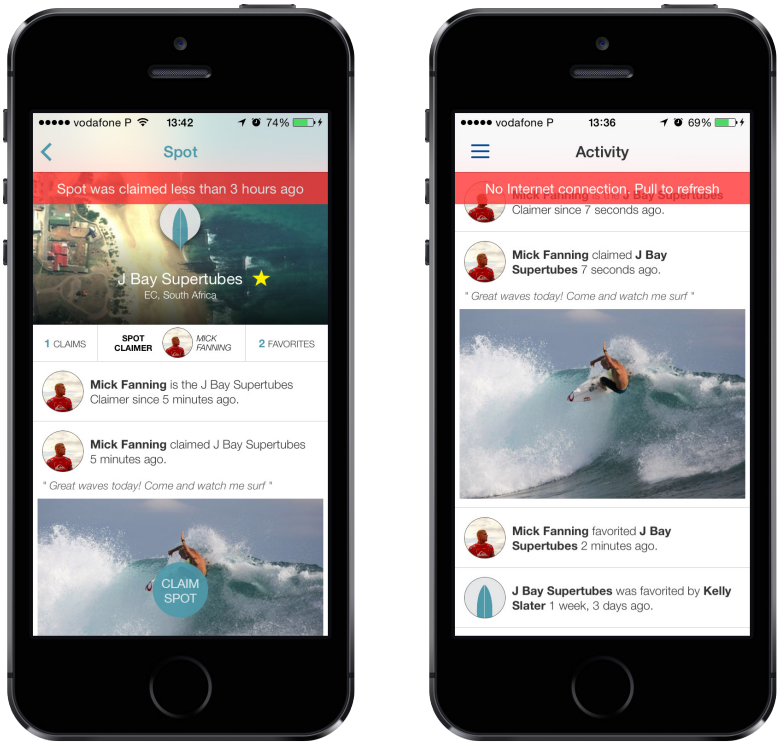


Figura 66. Exemplo do alerta implementado em *CustomTableAlertView.cs* numa *UITableView*.

Activity.cs (figuras 47 e 48)

Neste ficheiro é implementado o ecrã *User Dashboard* com a atividade do utilizador, dos utilizadores seguidos e dos *spots* favoritos, para tal é usada uma *UITableView* para apresentar os registos numa tabela. Sendo que este é o segundo ecrã a ser apresentado ao utilizador após este efetuar *login* ou *signup* é importante que, mesmo não existindo nenhuma atividade, o utilizador seja informado sobre isso e que tenha a possibilidade de ir para um ecrã onde possa seguir os utilizadores já registados ou efetuar *claim* num *spot*. Foi assim implementada uma verificação para que caso não haja atividade sejam dadas essas opções ao utilizador, tal como foi apresentado nos ecrãs de alta fidelidade.

Sempre que o utilizador acede a este ecrã é verificado se houve alguma nova atividade e, caso isto se verifique, o ecrã será atualizado. No entanto,

é também possível forçar a atualização do registo de atividade ao fazer o gesto de deslizar para baixo (*pull to refresh*) quando o início da vista de tabela já se encontra no topo do ecrã.

De modo a visualizar os perfis dos utilizadores e os *spots* que aparecem no registo de atividade foi implementada uma funcionalidade que permite que ao se pressionar o nome de um utilizador ou de um *spot* seja apresentado ao utilizador o ecrã correspondente ao utilizador ou ao *spot*, respetivamente. Esta funcionalidade está também disponível ao pressionar nas fotos de utilizadores ou nos ícones dos *spots*.

AddSpotMap.cs (figura 57)

A implementação do ecrã *Add Spot* resulta em dois ecrãs com a funcionalidade de adicionar um novo *spot* à plataforma; ambos contêm a localização do *spot* a adicionar, sendo esta definida no primeiro ecrã. Embora sejam dois ecrãs, a implementação de ambos foi realizada no mesmo ficheiro de modo a permitir uma transição, a qual mantém o contexto entre os ecrãs. No ecrã inicial é apresentado um mapa, implementado através da *framework MapKit*, centrado na localização do utilizador, no caso de este ter aceitado as permissões pedidas pela aplicação. O mapa contém todos os *spots* adicionados até ao momento na aplicação, sendo possível ir para o ecrã de um *spot* ao selecionar um dos ícones referentes a um desses *spots*. Para facilitar o posicionamento do círculo que marca a localização do *spot* a adicionar foi implementado um menu que permite alterar entre os três tipos de visualização do mapa: estradas, satélite e híbrido. Caso o utilizador pretenda adicionar um *spot* distante da localização onde se encontra, poderá procurar na caixa de pesquisa existente no topo do ecrã o local que pretende. Ao selecionar um dos locais da lista de sugestões da pesquisa, o mapa é centrado na localização selecionada.

Quando o utilizador acaba de situar o local a adicionar deve pressionar o botão "Next". Este botão faz com que ocorra a transição para o segundo ecrã, movendo-se o mapa para uma área mais pequena no topo do ecrã. Na animação são escondidos os seguintes elementos: o menu de seleção do tipo de mapa, o círculo com a informação sobre o posicionamento e a caixa de pesquisa. Durante a animação é apresentada uma vista de tabela que contém os detalhes do *spot* que são de preenchimento obrigatório. Quando todos os campos são preenchidos é ativado o botão "Save". Ao pressionar este botão são enviados os detalhes do *spot* a adicionar para o servidor. Caso a resposta do servidor seja de sucesso é animada a adição do *spot* com uma nova anotação a cair sobre o mapa e, posteriormente, a aplicação volta ao ecrã *Nearby Spots*.

ClaimView.cs (figura 56)

Este ficheiro implementa a janela que apresenta o ecrã para fazer *claim* a um *spot*. Esta janela é apresentada quando o utilizador pressiona o botão "Claim Spot" presente no ecrã "View Spot". A janela não será apresentada caso o utilizador tenha efetuado um *claim* há menos de três horas no *spot* selecionado. Nesta janela o utilizador pode inserir a descrição do *claim* e uma

foto associada. A foto pode ser adicionada ao pressionar o ícone de máquina fotográfica, sendo que o utilizador pode escolher entre uma foto capturada no momento ou uma foto da sua biblioteca de imagens. Ao pressionar o botão "Claim" é efetuado *claim* do *spot*, sendo enviados os dados da descrição e da foto para o servidor caso estes existam. Em caso de *claim* com sucesso a tabela que apresenta a atividade no ecrã *View Spot* é atualizada, apresentando o *claim* efetuado pelo utilizador.

FirstScreen.cs (figuras 41 a 46)

Este ficheiro contém a implementação de todos os ecrãs relativos ao *login* e *signup* de utilizadores na aplicação, assim como do ecrã de boas vindas. A implementação destes ecrãs no mesmo ficheiro deveu-se à animação realizada na transição entre estes. Uma vez que se pretendia manter o fundo estático entre os diferentes ecrãs a implementação não podia ser realizada através da transição de *view controllers*. As transições foram efetuadas através do agrupamento dos botões e das *views* associadas a cada um dos ecrãs, movendo apenas esses grupos e mantendo o fundo na mesma posição. No caso dos ecrãs de *login* e *signup* por *email* foi efetuada uma transição para um fundo com um efeito de desfoque aplicado previamente num *software* de edição de imagem. Tanto este fundo como os restantes elementos da interface destes ecrãs eram sobrepostos ao ecrã anterior, animando a sua opacidade.

No preenchimento dos campos necessários para efetuar *login* ou *signup* é alterada a posição no ecrã destes campos de modo a que todos os campos sejam visíveis com o teclado sobreposto e para que o utilizador não perca o contexto do que está a preencher, visível na figura 67.

Figura 67. Exemplo dos ecrãs de *login* e *signup* durante o preenchimento dos campos com o teclado visível.

Quando o botão "X" no canto superior esquerdo é pressionado, a aplicação volta a apresentar o ecrã anterior, limpando todos os campos do ecrã atual que tinham sido preenchidos, incluindo o campo de foto no ecrã de *signup* por *email*.

Ao pressionar o botão "Login" ou "Signup" os dados preenchidos são enviados para o servidor e, caso a resposta do servidor seja de sucesso, será apresentado ao utilizador o ecrã *Welcome Screen*. No ecrã *Welcome Screen*, ao pressionar o botão "Go" a aplicação apresenta o ecrã *User Dashboard* ao utilizador.

LoadingScreen.cs (figura 40)

Neste ficheiro foi implementado o primeiro ecrã apresentado ao utilizador aquando da execução da aplicação. É com este ficheiro que é efetuada a animação do logo, sendo esta iniciada a partir da imagem que é usada durante o carregamento em memória da aplicação. Durante esta animação é também verificada se existem as credenciais do utilizador no *Keychain* da aplicação, usando a *framework Monotouch.Security*. Caso existam credenciais, é efetuado o *login* automaticamente na aplicação, comunicando com o servidor e, em caso de sucesso, é apresentado o ecrã *User Dashboard*. Neste ecrã é também verificado pela primeira vez se a aplicação tem conectividade à Internet, sendo o utilizador avisado caso esta conexão não exista.

SideMenu.cs (figura 49)

Este ficheiro contém o código implementado relativo ao funcionamento do menu lateral de navegação. Embora já existam implementações semelhantes, decidiu-se implementar este menu de raiz de modo a ter mais controlo sobre a navegação da aplicação e a animação realizada. Este menu consiste num *view controller* que é apresentado em simultâneo com o *view controller* que gere todos os ecrãs da aplicação. Isto permite detectar o *input* do utilizador em ambos os ecrãs apresentados, dando a possibilidade ao utilizador de selecionar um item do menu ou selecionar o ecrã onde estava anteriormente.

SpotList.cs (figura 53)

Neste ficheiro está implementado o ecrã *Nearby Spots*, no qual é apresentada a lista dos *spots* existentes na aplicação por ordem de proximidade, caso o utilizador tenha dado permissões à aplicação para usar a sua localização. Esta lista é obtida através de um pedido ao servidor e é apresentada numa *UITableView*, sendo cada *spot* colocado numa linha da tabela. Para cada uma das linhas é apresentado o nome, descrição, localização do *spot* e a distância até ao seu local. O ícone apresentado junto ao nome do *spot* é diferente consoante o desporto associado ao *spot*. Ao pressionar uma das linhas da tabela é enviada a informação do *spot* para o ecrã *View Spot*, de modo a neste se apresentar o registo de atividade do *spot* selecionado.

SpotView.cs (figuras 54 e 55)

O ficheiro "SpotView.cs" contém a implementação do ecrã *View Spot*. Neste ecrã é apresentado o local do *spot* num mapa juntamente com os seus detalhes. Sobre o mapa foi implementada uma *UITableView* que contém mais informação sobre o *spot* e o registo de toda a atividade realizada no local. Os dados apresentados nesta tabela são obtidos no servidor aquando da apresentação do ecrã. Caso o servidor não devolva dados é apresentada uma mensagem na tabela a informar que não existe atividade no *spot*. Ao detectar o evento de *scroll* nesta tabela é efetuada uma animação no mapa que mostra a localização do *spot*. Sempre que o movimento do *scroll* é efetuado para baixo, é realizada uma animação de zoom no local do *spot* no mapa, consoante o número de píxeis que a tabela se deslocou para baixo. Ao deslizar para baixo é apresentada a descrição do *spot*, a qual se encontra por trás da tabela. A funcionalidade de adicionar o *spot* aos favoritos é também apresentada neste ecrã, sendo que o botão que realiza esta tarefa muda de estado consoante o *spot* está nos favoritos ou não. Ao adicionar ou remover um *spot* dos favoritos e ao fazer *claim* ao *spot* é efetuado o download do registo de atividade do *spot* e é preenchida novamente a tabela com os novos dados.

UserList.cs (figura 50)

A implementação deste ficheiro é bastante semelhante à do ficheiro "SpotList.cs", sendo neste caso apresentada uma lista dos utilizadores registados na plataforma com as suas imagens de perfil associadas. Esta lista é também obtida a partir do servidor aquando do carregamento do ecrã, sendo apresentada também numa *UITableView*. Neste ficheiro foi também implementado um botão de "Follow" associado a cada um dos utilizadores, permitindo assim adicionar um utilizador à lista de utilizadores seguidos. Este botão, tal como o botão de adicionar um *spot* aos favoritos, apresenta um estado diferente caso o utilizador seja seguido ou não pelo utilizador logado na aplicação.

UserProfile.cs & UserView.cs (figuras 58, 51 e 52)

Nestes dois ficheiros estão implementadas as visualizações de perfis dos utilizadores e do seu registo de atividade. A diferença entre a implementação dos dois reside na ausência do botão "Follow" no caso do ficheiro *UserProfile*, sendo que neste ficheiro é adicionada a funcionalidade que implementa o *logout* do utilizador na aplicação. Ao pressionar o botão de logout a aplicação mostra ao utilizador o ecrã *Login/Register & Find Nearby Spots* após processar o *logout* no servidor. Ao efetuar *logout*, as credenciais do utilizador guardadas no *Keychain* são removidas.

9. AVALIAÇÃO DE USABILIDADE

TESTES DE USABILIDADE

Para avaliar as funcionalidades e detectar lacunas no protótipo implementado ao nível da usabilidade foram realizados testes de usabilidade com um grupo de utilizadores. Os utilizadores que realizaram o teste fazem parte do público alvo da aplicação, pois praticam ou já praticaram algum dos desportos de ação disponíveis na aplicação. O desporto praticado pelos utilizadores que participaram não era importante, uma vez que as funcionalidades disponíveis na aplicação são iguais para qualquer desporto.

Antes de proceder à realização dos testes foi necessário desenvolver um guião que incluísse os passos a seguir pelo utilizador na aplicação. Cada um destes passos permitia avaliar uma das funcionalidades da plataforma e detectar se o utilizador tinha dificuldades a executá-lo, sendo isso um indicador de que a funcionalidade não tinha sido bem desenhada, implementada ou não estava clara o suficiente. Além do guião, foi criado um formulário com alguns campos para o utilizador preencher. O formulário incluía vários campos que permitiam aferir, por exemplo, quais os sistemas operativos dos smartphones utilizados pelos utilizadores e quais os desportos de ação que praticam no presente ou que já praticaram. Tanto o guião como o formulário estão disponíveis no apêndice deste documento.

Os testes foram realizados na Praia do Cabedelo, uma das praias da Figueira da Foz com maior afluência de surfistas e onde está localizada uma escola de surf. No dia em que os testes foram realizados o céu estava limpo e o vento estava moderadamente forte. As condições de visualização de um ecrã móvel num dia de sol são reduzidas e o vento não ajudou na comunicação com os utilizadores, tendo sido assim um desafio a realização dos testes mas também um desafio para a aplicação, uma vez que esta seria testada em condições de uso reais.

Nos testes contou-se com a participação de 5 pessoas, uma vez que para obter os melhores resultados não são necessários mais do que 5 utilizadores (Jakob Nielsen, 2000). Embora não fosse uma amostra suficientemente grande para apresentar dados estatísticos, revelou-se suficiente para tirar conclusões relativamente à usabilidade da aplicação.

De seguida serão apresentados os resultados obtidos, bem como as conclusões retiradas e as melhorias efetuadas após as conclusões a que se chegou.

Os gráficos seguintes foram obtidos através dos dados retirados dos formulários fornecidos aos utilizadores.

Figura 68. Dados resultantes do questionário fornecido durante os testes de usabilidade.

Nos dispositivos móveis utilizados, 3 dos 5 utilizadores utilizam Android, isto pode causar alguma estranheza com o sistema operativo iOS onde a aplicação foi testada, no entanto, isto não se verificou durante a realização dos testes. É também possível ver que dos 5 utilizadores, alguns praticam mais do que um desporto, neste caso surf e skate. As idades dos utilizadores estão entre os 16 e os 26 anos.

Na lista seguinte estão os passos que foram pedidos aos utilizadores para estes executarem na aplicação. Durante a execução dos passos foi contabilizado o tempo que cada um levou e foram anotadas as dificuldades tidas pelos utilizadores e o que os levou a demorar mais tempo do que o necessário para concluir a tarefa. Após a conclusão de todos os passos, foi pedido aos utilizadores que avaliassem de 0 a 10 a dificuldade que tiveram a executar cada uma das tarefas, 0 corresponde a nenhuma dificuldade e 10 a muita dificuldade.

– Efetua *signup* com *email*
– Abre o menu lateral e vai para a lista de *claimers*
– Na lista de *claimers* seleciona o *claimer* com o nome Kelly Slater
– Faz *follow* ao Kelly Slater
– Seleciona o último *spot* onde o Kelly Slater fez *claim*
– Adiciona o *spot* selecionado aos teus favoritos
– Vai para a lista de *spots*
– Seleciona um *spot* que se encontre a menos de 30 Km
– Adiciona o *spot* selecionado aos favoritos
– Volta para o ecrã de atividade
– Seleciona o *spot* da primeira publicação que aparece no ecrã de atividade
– Faz *claim* deste *spot* com uma foto tirada no momento
– Adiciona um *spot* de skate no local onde te encontras
– Vai ao teu perfil ver o teu registo de atividade
– Faz *logout* da aplicação

O gráfico seguinte apresenta a média de tempo demorado pelos utilizadores em cada uma das tarefas:

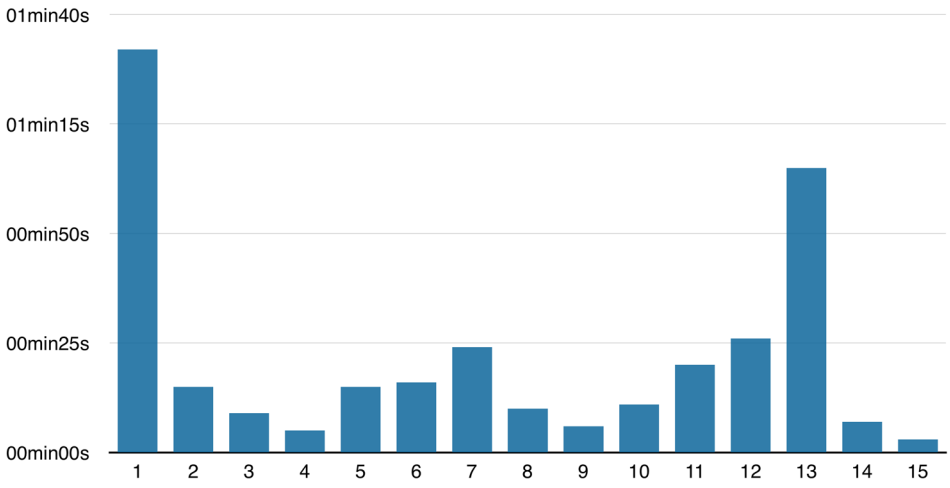


Figura 69. Média de tempo usado por tarefa.

De seguida está apresentada a dificuldade média sentida pelos utilizadores em cada tarefa, numa escala de 0 a 10:

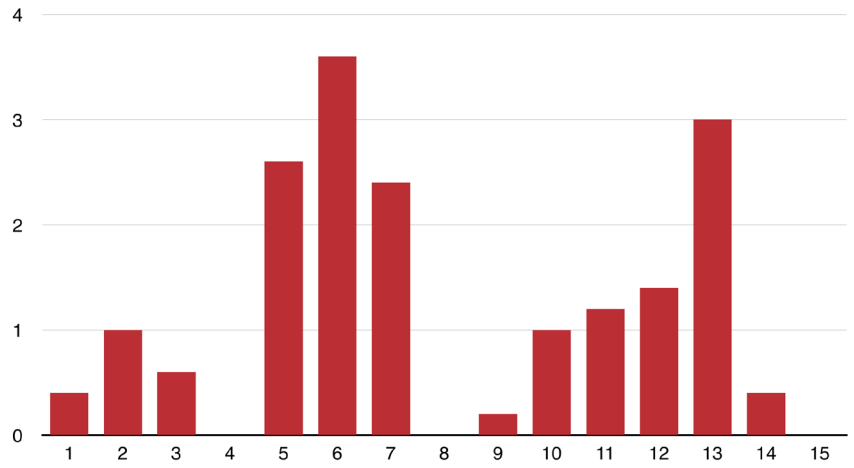


Figura 70. Média de dificuldade por tarefa.

Embora a dificuldade de uma tarefa afete o tempo que cada tarefa leva a executar, o tempo que os utilizadores levaram a executar as tarefas depende essencialmente da natureza da tarefa. Há tarefas em que é necessária a introdução de dados (p. ex. tarefa número 1) levando assim mais tempo a concluir, onde noutras é apenas necessário pressionar um botão (p. ex. tarefa número 4). No entanto, as dificuldades tidas permitem identificar se os utilizadores não perceberam como se executava determinada tarefa.

Na lista seguinte serão apresentadas as conclusões tiradas relativamente a cada passo que os utilizadores executaram:

– Efetua *signup* com *email* — O passo de efetuar *signup* com *email* foi aquele que levou em média mais tempo para ser completado,embora tenha sido um dos que representou menor dificuldade na sua conclusão. Isto deveu-se ao tempo que os utilizadores demoravam a preencher os dados necessários, sendo que por vezes se enganavam na introdução de algum campo. Esta tarefa também serviu para validar as setas de navegação presentes no primeiro ecrã, uma vez que por vezes os utilizadores pressionavam o botão de *login* em vez de *signup* mas imediatamente se apercebiam e voltavam atrás.

– Abre o menu lateral e vai para a lista de *claimers* — Nesta tarefa também foi contabilizado o tempo que os utilizadores levam a ver o ecrã de boas vindas, o *Welcome Screen*. A dificuldade aqui foi baixa uma vez que o botão usado para abrir o menu lateral é bastante semelhante ao usado por inúmeras aplicações, sendo que ao chegarem ao ecrã de atividade os utilizadores rapidamente pressionavam o botão para abrir o menu. A seleção da opção "Claimers" no menu lateral, de modo a ser apresentada a lista de *claimers*, foi também quase imediata.

– Na lista de *claimers* seleciona o *claimer* com o nome "Kelly Slater" — A seleção do *claimer* com o nome "Kelly Slater" levou apenas alguns segundos, sendo que os utilizadores era obrigados a deslizar a lista para baixo de modo a aparecer o utilizador com o nome pretendido. A dificuldade neste passo também foi bastante reduzida.

– Faz *follow* ao Kelly Slater — A ação de efetuar *follow* ao utilizador selecionado foi praticamente imediata, daí o nível de dificuldade reportado pelos utilizadores ter sido nulo.

– Seleciona o último *spot* onde o Kelly Slater fez *claim* — Esta tarefa levou algum tempo a mais que o necessário uma vez que os utilizadores tinham de ver o registo de atividade que o utilizador "Kelly Slater" tinha no seu perfil. Assim que se apercebiam da palavra "claimed" a seleção do *spot* associado à atividade era imediata. A dificuldade é um pouco elevada devido aos utilizadores terem que perceber o que é apresentado em cada publicação do registo de atividade.

– Adiciona o *spot* selecionado aos teus favoritos — O passo de adicionar o *spot* selecionado anteriormente aos favoritos foi a tarefa que demonstrou apresentar maior dificuldade aos utilizadores. Isto deveu-se não à ação em si, uma vez que só consiste em pressionar um botão, mas sim à dificuldade em encontrar o botão que lhes permite adicionar esse spot aos favoritos. Alguns dos utilizadores acabaram por pressionar a caixa que apresenta a informação de quantos utilizadores adicionaram o spot a ser visualizado aos favoritos. Esta dificuldade em encontrar o botão verifica-se no tempo demorado em média pelos utilizadores a realizar uma tarefa bastante simples.

– Vai para a lista de *spots* — Este passo apresentou alguma dificuldade aos utilizadores devido à navegação que já tinham efetuado na aplicação até chegarem a esta tarefa. Da lista de *claimers* para o perfil do utilizador "Kelly Slater" e depois para o *spot* do último *claim* desse utilizador. Para se dirigirem à lista de *spots* os utilizadores teriam que retroceder os dois últimos ecrãs, pressionar o botão para abrir o menu lateral e depois selecionar a opção "Spots". Alguns utilizadores imediatamente retrocederam até ao menu lateral e selecionaram a opção pretendida uma vez que tinham memorizado as opções quando abriram o menu pela primeira vez. Outros acabaram por demorar mais algum tempo até voltarem ao menu lateral, sendo que a seleção da opção "Spots" era imediata. Embora tenha criado alguma dificuldade aos utilizadores, a navegação da aplicação é semelhante a muitas existentes, sendo que requer alguma habituação.

– Seleciona um *spot* que se encontre a menos de 30 Km — Ao selecionarem a lista de "Spots" no passo anterior, os utilizadores rapidamente se apercebiam dos *spots* a menos de 30 Km, uma vez que estes estavam ordenados por distância. A seleção foi praticamente imediata, tendo a dificuldade associada a este passo sido nula.

– Adiciona o *spot* selecionado aos favoritos — A dificuldade deste passo em relação ao passo número 6, no qual também se devia adicionar o *spot* aos selecionados, foi muito menor uma vez que os utilizadores já tinham memorizado como se executava a tarefa.

– Volta para o ecrã de atividade — A dificuldade para voltar ao ecrã de atividade é semelhante à descrita no passo número 7, devendo-se à navegação previamente efetuada na aplicação. No entanto, os passos necessários para completar esta tarefa foram concluídos em relativamente pouco tempo.

– Seleciona o *spot* da primeira publicação que aparece no ecrã de atividade — Tal como no passo número 5, aqui os utilizadores tinham de ver as publicações existentes no registo de atividade e selecionar o *spot* associado à publicação mais recente. Isto foi completado em pouco tempo, tendo a dificuldade sido baixa.

– Faz *claim* deste *spot* com uma foto tirada no momento — A tarefa de efetuar *claim* do *spot* selecionado anteriormente com uma foto tirada no momento foi realizada facilmente pelos utilizadores, sendo que a dificuldade encontrada se deveu um pouco ao posicionamento do botão "Claim Spot". No entanto, os utilizadores rapidamente se aperceberam da sua localização e os passos seguintes foram efetuados rapidamente.

– Adiciona um *spot* de *skate* no local onde te encontras — Adicionar um *spot* no local onde os utilizadores se encontravam foi a segunda tarefa que apresentou maior dificuldade aos utilizadores. Inicialmente, os utilizadores não perceberam que a tarefa a executar se tratava de adicionar um novo *spot* à aplicação, isto deveu-se sobretudo à forma como o enunciado da tarefa foi escrito. Os utilizadores acabaram por se dirigir ao ecrã com a lista de *spots* e rapidamente se aperceberam do botão "+" para poderem adicionar algo, uma vez que este uso é comum em várias aplicações. Após este passo, verificou-se também que a linguagem usada no primeiro ecrã para situar a localização do *spot* não foi a ideal, pois os utilizadores não percebiam que era necessário situar o círculo vermelho sobre o *spot* e de seguida pressionar o botão "Done". Assim que se apercebiam que era necessário pressionar o botão a tarefa era completada em pouco tempo. O tempo associado a esta tarefa deve-se não só à dificuldade mas também ao fato de os utilizadores terem que preencher alguns campos.

– Vai ao teu perfil ver o teu registo de atividade — Esta tarefa foi executada rapidamente por todos os utilizadores, sendo bastante baixo o grau de dificuldade registado.

– Faz *logout* da aplicação — Após estarem nos seus perfis os utilizadores rapidamente concluíram esta tarefa ao pressionar o botão "Logout", tendo sido a dificuldade nula.

Após as conclusões retiradas em cada um dos passos, decidiu-se proceder a uma melhoria nos dois passos onde se sentiram mais dificuldades. Pretendia-se assim melhorar a visibilidade do botão que permite adicionar um *spot* aos favoritos do utilizador e alterar a linguagem no primeiro passo da tarefa de adicionar um novo *spot* à aplicação. As soluções foram as seguintes:

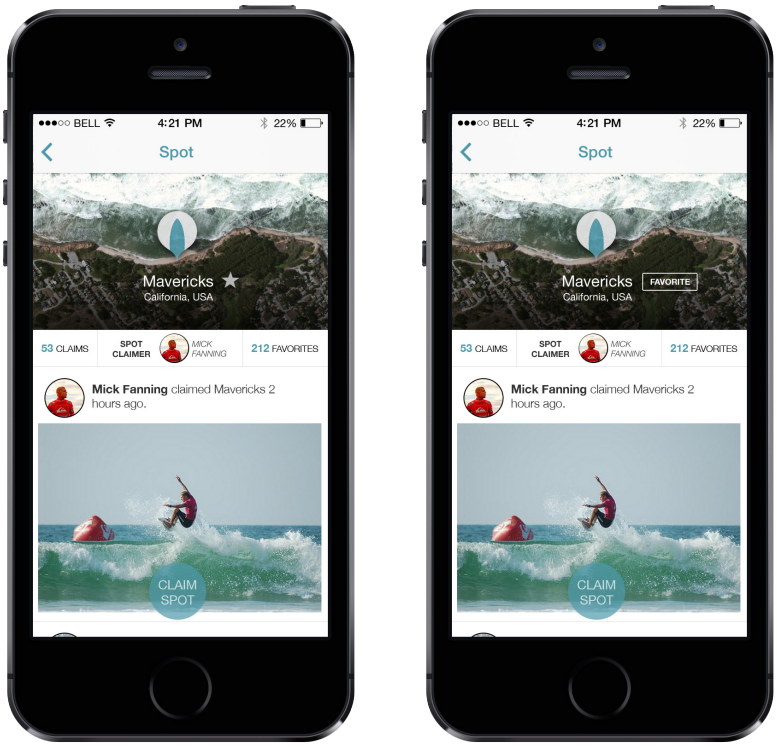


Figura 71. Alteração do botão de favorito. De uma estrela (esquerda) para botão com texto (direita).

No ecrã de um *spot* alterou-se a estrela por um botão semelhante ao botão "follow" presente no ecrã do perfil de um utilizador. Esta alteração visa tornar o botão de adicionar um *spot* aos favoritos mais visível e direto.

No ecrã de adicionar um novo *spot* foi alterada a linguagem, tornando assim mais clara a intenção do ecrã e quais tarefas devem ser executadas pelo utilizador. A alteração do nome do botão "Done" para "Next" permite ao utilizador saber que ainda existe um passo seguinte para completar a tarefa.

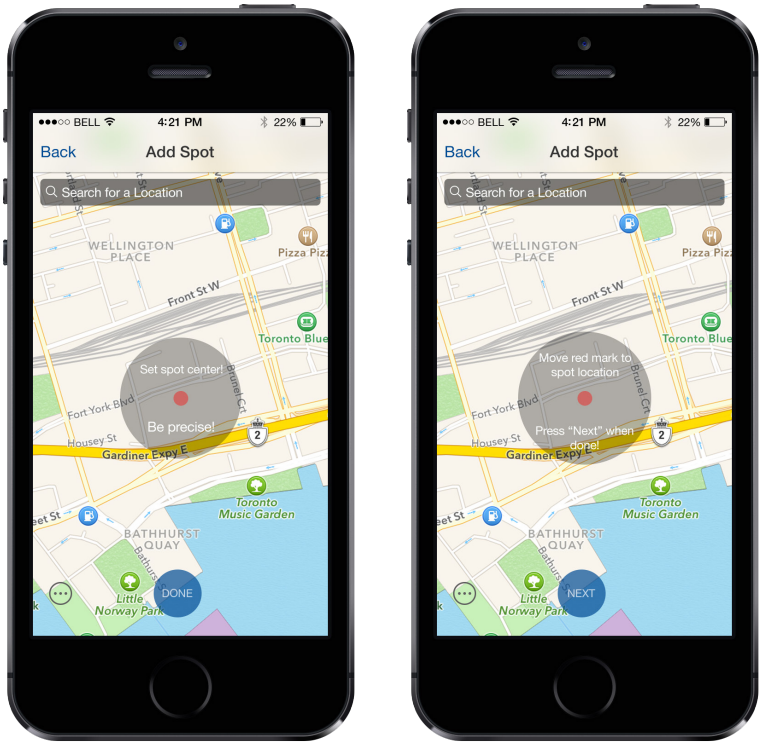


Figura 72. Alteração do texto de informação e do botão "Done" (esquerda) para "Next" (direita).

10. CONCLUSÃO

O desporto de ação tem tido um grande crescimento nos últimos anos. Enquanto alguns desses desportos ainda estão associados a um grande risco, noutros o risco é cada vez menor, contribuindo assim para uma aceitação e adoção por parte da população em geral. Em paralelo com o crescimento dos desportos de ação temos a evolução dos dispositivos móveis. Estes fornecem aos utilizadores novas possibilidades, estando disponíveis em qualquer lugar e sendo cada vez maior a quantidade de aplicações que nos levam a partilhar conteúdo e a comunicar de novas formas.

A aplicação *SpotClaimer* pretende criar uma nova forma de partilha dos conteúdos criados pelos atletas dos desportos de ação, dando-lhes a possibilidade de partilhar as suas localizações com os seus seguidores e, assim, promover relações entre atletas durante a prática do desporto. Ao partilharem as suas localizações e ao atraírem mais atletas para o mesmo local, os utilizadores da aplicação estão indiretamente a contribuir para a dinamização dos locais onde praticam o desporto.

O desenvolvimento deste projeto resultou, sobretudo, em aprendizagem. O uso de novas tecnologias e o desenho e implementação da aplicação móvel revelaram-se um desafio devido às áreas de conhecimento abrangidas. Após o planeamento efetuado das tarefas que seriam necessárias para a conclusão deste projeto, deu-se início à realização do mesmo. Começou-se por uma pesquisa bibliográfica, de modo a conhecer melhor o desporto de ação e os meios tecnológicos para os quais a aplicação era direcionada. Depois desta pesquisa, procedeu-se ao desenvolvimento da aplicação propriamente dita, passando por diversas fases até chegar a um protótipo funcional. Começou-se pela definição das funcionalidades a implementar, sendo que muitas resultaram do processo de prototipagem, no qual se definiram modelos de interação e de dados para dar suporte à aplicação. O modelo de interação foi a base para os esboços e desenho final da interface da aplicação, sendo o modelo de dados a base para a plataforma web. Esta teria de dar suporte às funcionalidades da aplicação. Após as fases de prototipagem e design, começou-se a implementação, sendo constantemente revisitado o trabalho realizado nas fases anteriores, resultando assim num processo de desenvolvimento quase cíclico. A implementação visava a criação dos ecrãs principais da aplicação de modo a demonstrar as suas funcionalidades fulcrais.

Os testes de usabilidade realizados após a conclusão do protótipo funcional ajudaram a identificar e corrigir algumas lacunas existentes ao nível da usabilidade da aplicação.

TRABALHO FUTURO

Tendo em conta que a aplicação desenvolvida se trata apenas de um protótipo funcional, será necessário algum trabalho no futuro de modo a colocar a aplicação numa loja de aplicações. Para tal, será importante adicionar mais funcionalidades ao protótipo e realizar novamente testes com utilizadores, com a intenção de validar essas novas funcionalidades. Além de novas funcionalidades, será também importante desenvolver a aplicação para outro sistema operativo além de iOS, principalmente o sistema Android devido à fatia de mercado que hoje ocupa no segmento dos dispositivos móveis.

Quanto às funcionalidades a desenvolver, será importante implementar o *login* e *signup* por Facebook, de modo a facilitar estes processos na aplicação e reduzir o número de passos necessários. Será também importante a implementação de filtros de pesquisa nos ecrãs que apresentam a lista *spots* e de utilizadores. Para melhorar o conteúdo inserido na aplicação, a possibilidade de editar e eliminar *spots* e a de editar o perfil de utilizador são de relevância acentuada. No ecrã de um *spot* poderão ser desenvolvidas funcionalidades para permitir classificar e criticar o *spot*, obter informação sobre o *spot* em tempo real e aumentar o número de caraterísticas do mesmo. A possibilidade de partilha de conteúdo multimédia sem estar associado a um *claim* também será determinante para o crescimento da plataforma, dando assim a possibilidade aos fãs de inserirem conteúdo relacionado com os atletas e com os *spots*.

Tal como foi mencionado anteriormente, será importante expandir a aplicação a mais desportos, aumentando assim o público alvo.

Assim que a aplicação for lançada nas lojas de aplicações será importante divulgar a mesma. A divulgação poderá ser realizada através de parcerias com eventos, criação de páginas nas redes sociais e publicação de uma página web.

10. BIBLIOGRAFIA

git - About. 2014, de http://git-scm.com/about

json2csharp - generate c# classes from json. 2014, de http://json2csharp.com

Surf Terms. (2011). Acedido em janeiro, 2014, de http://www.surfing-waves.com/surf_talk.htm

Reachability Sample. (2011). 2014, de https://github.com/xamarin/monotouch-samples/tree/master/ReachabilitySample

"MugoSurf". (2013). Acedido em janeiro, 2014, de http://www.dropingood.com/

"Spot Digger". (2013). Acedido em janeiro, 2014, de http://www.spotdigger.com/

"WeRide". (2013). Acedido em janeiro, 2014, de http://werideskate.com/

"goFlow". (2013). Acedido em janeiro, 2014, de http://goflow.me/

"About Foursquare". (2014). Acedido em janeiro, 2014, de https://pt.foursquare.com/about

Making queries - Django Documentation. (2014). 2014, de https://docs.djangoproject.com/en/dev/topics/db/queries/

URL Dispatcher - Django Documentation. (2014). 2014, de https://docs.djangoproject.com/en/1.6/topics/http/urls/

Xamarin. (2014). 2014, de http://xamarin.com

Components - Xamarin. (2014). 2014, de http://components.xamarin.com

Models - Django Documentation. (2014). 2014, de https://docs.djangoproject.com/en/dev/topics/db/models/

Model field reference - Django Documentation. (2014). 2014, de https://docs.djangoproject.com/en/dev/topics/db/models/

Samples - Xamarin. (2014). 2014, de http://developer.xamarin.com/samples-all/

Apple. (2014). Apple Developer. 2014, de https://developer.apple.com

Apple. (2014). iOS Human Interface Guidelines. 2014, de https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/index.html

Beyer, H., & Holtzblatt, K. (1998). Contextual design : defining customer-centered systems. San Francisco, Calif.: Morgan Kaufmann.

Blacksberg, M. (2009). Snowboarding History. Acedido em janeiro, 2014, de http://www.sportinglife360.com/index.php/snowboarding-history-46892/

Boggan, P. (Producer). (2013). Binding Third-Party Objective-C Libraries. Retrieved de http://vimeo.com/70841878

Brooke, M. (1999). *The Concrete Wave: The History of Skateboarding: Warwick Publishing.*

Celania, M. (2012). The History of Surfing. Acedido em janeiro, 2014, de http://www.neatorama.com/2012/08/13/The-History-of-Surfing/#!8kQDW

Chivers, T. (2013). The story of Google Maps. Acedido em janeiro, 2014, de http://www.telegraph.co.uk/technology/google/10090014/The-story-of-Google-Maps.html

Dilger, D. E. (2012). Five years of iPhone. Acedido em janeiro, 2014, de http://appleinsider.com/articles/12/01/09/five_years_of_iphone

Elkstein, M. (2008). Learn REST: A Tutorial. 2014, de http://rest.elkstein.org/2008/02/what-is-rest.html

Enis, M. (2013). Mobile Evolution: How Apps Are Adapting to a New Device Ecosystem. Acedido em janeiro, 2014, de http://www.thedigitalshift.com/2013/02/mobile/mobile-evolution/

Farhat, A. (2013). Deploy Django on Apache with Virtualenv and mod_wsgi. 2014, de http://thecodeship.com/deployment/deploy-django-apache-virtualenv-and-mod_wsgi/

Gonzalez, K. (2011). History of skateboard wheels 1920s to 1970s: A brief overview. de http://sports.yahoo.com/top/news?slug=ycn-8588827

Guardino, P. (2010). Skateboarding v Snowboarding. Acedido em janeiro, 2014, de http://www.sportinglife360.com/index.php/skateboarding-v-snowboarding-11060/

Herman, S. (2014). "Skate Spots". Acedido em janeiro, 2014, de http://seanhermandesign.com/skatespots/#

Lockwood, N. (2014). FXBlurView. 2014, de https://github.com/nicklockwood/FXBlurView

MacArthur, P. J. (2010). The Top Ten Important Moments in Snowboarding History. Acedido em janeiro, 2014, de http://www.smithsonianmag.com/history/the-top-ten-important-moments-in-snowboarding-history-6851590/

Marshall, G. (2013). Apple Maps: one year on. Acedido em janeiro, 2014, de http://www.techradar.com/news/software/applications/apple-maps-one-year-on-1182395

Nielsen, J. 10 Usability Heuristics for User Interface Design. Acedido em junho, 2014, de http://designprinciplesftw.com/collections/10-usability-heuristics-for-user-interface-design

Nielsen, J. (2010). Why You Only Need to Test with 5 Users. Acedido em junho, 2014, de http://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/

Owen, T. (2013). The Evolution Of Skateboarding – A History From Sidewalk Surfing To Superstardom. Acedido em janeiro, 2014, de http://www.skateboardingmagazine.com/the-evolution-of-skateboarding-a-history-from-sidewalk-surfing-to-superstardom/

SiliconKarne. (2013). Surfs up with Glassy.pro app! Acedido em janeiro, 2014, de http://siliconkarne.com/startups/surfs-up-with-glassy-pro/

Taylor, S. (2010). How Is Surfing And Skateboarding The Same? Acedido em janeiro, 2014, de http://www.mademan.com/mm/how-surfing-and-skateboarding-same.html

Williams, L. (2011). The History of Extreme Sports…. so far. Acedido em janeiro, 2014, de http://oldilivextremesite.ilivextreme.com/the-history-of-extreme-sports-so-far.html/

Williamson, M. The history of extreme sports. Acedido em janeiro, 2014, de http://www.catalogs.com/info/sports/history-of-extreme-sports.html

APÊNDICE

SPOTCLAIMER

Testes de usabilidade

Nome: _____

Idade: _____

Ocupação: _____

Experiência com *smartphones*:

iOS(iPhone) ☐ Android ☐ Outro ☐ Qual? _____

Praticas algum desporto de ação?

Sim ☐ Não ☐

Assinala quais dos desportos seguintes já praticaste ou praticas.

Surf ☐ Skate ☐ Snowboard ☐

Na aplicação que vais testar de seguida efetua, por ordem, os passos que estão descritos na página seguinte e após completares todos os passos avalia de 0 a 10 conforme a dificuldade que tiveste a executar cada um dos passos, onde 0 corresponde a nenhuma dificuldade e 10 a muita dificuldade.

Em cada tarefa irá ser medido o tempo que demoras a executá-la.

Executa os passos seguintes na aplicação e no final de completares todos os passos avalia de 0 a 10 o nível de dificuldade sentida para executar com sucesso cada um.

Tarefa	Dificuldade
1. Efetua signup com email	
2. Abre o menu lateral e vai para a lista de *claimers*	
3. Na lista de *claimers* seleciona o *claimer* com o nome Kelly Slater	
4. Faz *follow* ao Kelly Slater	
5. Seleciona o último *spot* onde o Kelly Slater fez *claim*	
6. Adiciona o *spot* selecionado aos teus favoritos	
7. Vai para a lista de *spots*	
8. Seleciona um *spot* que se encontre a menos de 30 Km	
9. Adiciona o *spot* selecionado aos favoritos	
10. Volta para o ecrã de atividade	
11. Seleciona o *spot* da primeira publicação que aparece no ecrã de atividade	
12. Faz claim deste *spot* com uma foto tirada no momento	
13. Adiciona um *spot* de *skate* no local onde te encontras	
14. Vai ao teu perfil ver o teu registo de atividade	
15. Faz logout da aplicação	