Relatório de Estágio

Mestrado em Engenharia Informática – Computação Móvel

Desenvolvimento da aplicação Carris para o transporte público de Lisboa

João Francisco Pereira Fernandes

Leiria, setembro de 2018

Relatório de Estágio

Mestrado em Engenharia Informática – Computação Móvel

Desenvolvimento da aplicação Carris para o transporte público de Lisboa

João Francisco Pereira Fernandes

Estágio de Mestrado realizado na Tecmic sob a orientação da Doutora Maria Micaela Gonçalves Pinto Dinis Esteves, Professora da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria e coorientação da Doutora Sónia Maria Almeida da Luz, Professora da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria.

Leiria, setembro de 2018

Esta página foi intencionalmente deixada em branco

Dedicatória

À Minha Família

Esta página foi intencionalmente deixada em branco

Agradecimentos

Em primeiro lugar quero agradecer às professoras Sónia Luz e Micaela Esteves pelo apoio prestado durante este ano.

Da Tecmic, agradeço ao António Marcelo pela oportunidade que me foi dada e também aos meus colegas, Joel Fernandes, André Leal, André Sousa, Joni Batista, José Pedro, Luís Torres e à "Caixinha" pela motivação, preocupação e também pelo apoio no trabalho.

O meu muito obrigado aos meus pais que sempre estiveram disponíveis para me ajudar naquilo que precisasse. Obrigado pela paciência, pelo apoio naqueles dias menos bons, pela educação que me deram e por serem os melhores pais do mundo. Obrigado, Francisco e Paula Fernandes.

Um agradecimento e um grande abraço aos meus companheiros, Diogo Martins, Renato Santos, José Guerra, André Assunção e Guilherme Mendes, Pedro Jorge e Válter Bento.

Por último, mas não menos importante, um grande obrigado á minha namorada, Patrícia Silva, pelas conversas, sugestões e pela preocupação constante sobre o trabalho.

Esta página foi intencionalmente deixada em branco

Resumo

O presente estágio curricular foi realizado no âmbito da Unidade Curricular de Estágio pertencente ao mestrado de Engenharia Informática – Computação Móvel, lecionado na Escola Superior de Tecnologia e Gestão (ESTG), pertencente ao Instituto Politécnico de Leiria (IPLeiria). Este estágio teve como objetivo aplicar os conhecimentos obtidos durante o percurso académico. Foi realizado numa empresa de soluções informáticas, a Tecnologias de Microelectónica (Tecmic). Durante este estágio, foi possível conhecer mais detalhadamente o mundo do mercado de trabalho e foram adquiridas novas competências para a vida profissional. O trabalho realizado foi o resultado de uma necessidade da Tecmic e de um cliente de transportes públicos, a Carris. O trabalho teve como objetivo o desenvolvimento de uma aplicação para dispositivos móveis (Android e iOS) ajustada às necessidades do cliente que fosse o mais intuitiva possível, respeitando a diversificação das idades dos utentes da Carris. Esta aplicação, para além de promover a imagem da Carris, incentiva o uso dos transportes públicos na região de Lisboa. A aplicação permite obter informação detalhada sobre as paragens e as linhas pertencentes à Carris tal como as estimativas de tempo de espera correspondentes às paragens. A aplicação responde à contextualização geográfica do utilizador apresentando as paragens mais próximas de si ou através do planeamento de uma viagem. O trabalho desenvolveu-se segundo as boas práticas de desenvolvimento de software, seguindo as *guidelines* da Apple e da Google, tendo sido colocado em produção com base nos requisitos, nos testes realizados e na aprovação do cliente. As aplicações Android e iOS ficaram alojadas nas respetivas *stores* contando até ao momento com um grande número de utilizações e instalações de ambas as plataformas.

Palavras-chave*: Aplicações móveis, transporte público, tempo real, desenvolvimento móvel*

Esta página foi intencionalmente deixada em branco

Abstract

The present curricular internship was carried out within the scope of the Internship Curricular Unit belonging to the Master's in Computer Engineering - Mobile Computing, taught at the Higher School of Technology and Management (ESTG), belonging to the Polytechnic Institute of Leiria (IPLeiria). This internship had as the objective to apply the knowledge obtained during the academic course. It was carried out in a computer solutions company, Tecnologias de Microeletónica (Tecmic). During this stage, it was possible to know more about the world of the labor market and acquired new skills for working life. The work was the result of a need for Tecmic and a public transport customer, Carris. The objective of the work was to develop an application for mobile devices (Android and iOS) adjusted to the needs of the client that was as intuitive as possible, respecting the diversification of the ages of Carris users. This application, in addition to promoting the image of Carris, encourages the use of public transport in the Lisbon region. The application allows you to obtain detailed information about the stops and the routes belonging to Carris as well as the waiting time estimations of the stops. The application responds to the user's geographic context by presenting the stops closest to him or by planning a trip. The work was developed according to good software development practices, following the guidelines of Apple and Google, and was put into production based on requirements, tests performed and customer approval. Android and iOS apps have been housed in their stores with many uses and installations of both platforms.

Keywords: *Mobile applications, public transportation, real time, mobile development*

Esta página foi intencionalmente deixada em branco

Lista de figuras

Esta página foi intencionalmente deixada em branco

Lista de tabelas

Esta página foi intencionalmente deixada em branco

Lista de siglas

SIGLA	DESCRIÇÃO
AMP	Grande Área Metropolitana do Porto
APIs	Interface de programação de aplicações
APK	Android Package
ART	Android Runtime
DLL	Dynamic link library
DTO	Data Transfer Object
ETA	Estimated Time of Arrival
IDE	Integrated Development Environment
IPA	iOS App Store Package
JSON	JavaScript Object Notation
OPT	Optimização e Planeamento de Transportes S.A
POO	Programação Orientada a Objetos
SDK	Kit de desenvolvimento de software
SVN	Apache Subversion
TFS	Team Foundation Server
UI	User Interface
UX	User Experience
XML	Extensible Markup Language

Esta página foi intencionalmente deixada em branco

Índice

1. Introdução

Atualmente, com a presença de tecnologia sofisticada e a diversidade de dispositivos móveis, existe a possibilidade de simplificar as tarefas que envolvem o dia-a-dia do utilizador. Para isso são criadas aplicações móveis que, de certa forma, ajudam a orientar e informar o utilizador sobre o ambiente em que se encontra. O uso desta tecnologia tornou-se cada vez mais comum e permite que o tempo despendido, na espera do autocarro, seja usado de forma mais produtiva.

A aplicação Carris surge com o objetivo de mostrar a informação em tempo real, melhorando a interação do utilizador sobre os serviços Carris. Para além de incentivar o uso do transporte público, esta aplicação permite promover a imagem da própria Carris e eliminar as barreiras à utilização dos seus serviços.

1.1. Motivação e enquadramento

O trabalho apresentado neste documento foi elaborado no âmbito da unidade curricular de Estágio do Mestrado em Engenharia Informática – Computação Móvel, na Escola Superior de Tecnologia e Gestão (ESTG) pertencente ao Instituto Politécnico de Leiria [1].

A Companhia de Carris de Ferro de Lisboa (Carris) é uma empresa de transporte público de passageiros da cidade de Lisboa, tendo sido fundada em 1872. A Carris trabalha desde 2013 com um Sistema de Ajuda à Exploração (SAEIP), desenvolvido pela entidade de acolhimento do estágio, a Tecmic, que permite realizar operações em tempo real. Este sistema é baseado no conhecimento preciso da posição de cada veículo (através do sistema GPS) que é transmitido para a Central de Comando de Tráfego. Este processo permite transmitir informação fiável e em tempo real aos clientes, promove a segurança dos passageiros, tripulantes e material circulante, realiza uma melhor gestão das carreiras em exploração e garante eficácia no planeamento operacional [2]. A Carris possui diversos mecanismos de informação disponibilizando-os aos seus clientes, que permitem ir ao encontro das suas necessidades de obtenção de informação em tempo real.

Estes mecanismos são [3]:

- Os painéis eletrónicos - a Carris iniciou, em abril de 2001, o processo de implementação de painéis eletrónicos nas paragens, com informação sobre o tempo previsto de chegada (em minutos) dos veículos, permitindo assim ao cliente uma gestão do seu próprio tempo de espera nas paragens. Este é um processo contínuo da empresa que atualmente conta com 350 painéis instalados;
- O serviço de mensagens - o sistema de mensagens SMS ao minuto tem como principal objetivo proporcionar informação, via telemóvel, sobre os horários reais de passagem dos veículos nas paragens;
- O serviço de email - o serviço e-mail ao minuto tem como principal objetivo proporcionar informação, via internet, sobre os horários reais de passagem dos veículos nas paragens.

Ao longo dos últimos anos, a Carris tem procurado aperfeiçoar o seu funcionamento face às necessidades do mercado, aumentando assim a sua eficácia e promovendo a melhoria de qualidade dos seus serviços. Assume-se como uma empresa moderna com uma cultura organizacional baseada na produtividade, orientada para a qualidade do serviço ao cliente, otimizando a utilização de recursos e mantendo a sua responsabilidade de serviço social [4] .

A manutenção dos painéis eletrónicos e do serviço de mensagens é bastante dispendiosa para a Carris e daí surgir a necessidade de uma aplicação móvel que conseguisse fornecer a informação aos utilizadores de uma forma rápida permitindo-lhes um melhor aproveitamento do seu tempo.

1.2. Objetivos

Pretende-se implementar uma aplicação móvel com as seguintes características:

- Apresentação das paragens da rede Carris;
- Apresentação das linhas da rede Carris;
- Apresentação das estimativas de tempo previsto de chegada dos veículos;
- Apresentação de notificações;
- Planeamento de viagens entre diversos pontos;

- Contextualização geográfica mostrando as paragens mais próximas do utilizador;
- Guardar localizações de casa e trabalho assim como linhas e paragens favoritas.

1.3. Entidade de acolhimento

A entidade de acolhimento do estágio foi a Tecmic S.A. - Tecnologias de Microeletrónica, S.A. que foi fundada em 1988. Esta empresa está sediada em Leiria, contendo outros escritórios em Lisboa e no Brasil. Cresceu como membro de um dos maiores grupos em tecnologias de informação, a AITEC [5]. Foi originada num instituto de investigação e desenvolvimento de renome nacional e internacional, o INESC [6]. A origem da Tecmic surgiu num processo de transferência de tecnologia Universidade – Indústria e a partir daí, a Tecmic manteve parcerias e instalou capacidade própria de inovação e desenvolvimento continuo.

A Tecmic é uma empresa com soluções integradas de engenharia e sistemas de informação que otimizam a gestão dos recursos das organizações, tais como sistemas de gestão de transportes públicos, sistemas de gestão da logística e transporte de mercadorias sistemas de comando e controle de Forças de Segurança e Emergência, entre outros. A Tecmic comemorou em 2018 o seu trigésimo aniversário e é na atualidade uma referência no mercado de sistemas de gestão de frotas, meios e ocorrências. Tem como clientes, a Carris, o Grupo Luís Simões, os Horários do Funchal, os CTT, a Motorola, a Esegur, o ACP, etc. - com presença em países como, Espanha, Reino Unido, Alemanha, Áustria, Suíça, Brasil, entre outros.

A Tecmic conta na atualidade com plataformas nas áreas de Gestão de Transporte, nomeadamente no transporte público de passageiros (XTraN Passenger), na Gestão de Emergências (XTraN 4Forces) e na Gestão Integrada da Recolha de Resíduos (Ecogest)

1.4. Estrutura do Documento

Este documento está organizado em 7 capítulos. O primeiro capítulo descreve o tema deste estágio, o enquadramento e os objetivos que se pretendem alcançar. No capítulo 2 o estado da arte é apresentado, abordando as várias aplicações móveis orientadas aos transportes públicos, nomeadamente o Citymapper, o Lisboa Move-Me, o Google Maps, o Moovit, o Waze e por fim a XtraN Passenger app. No capítulo 3 é descrito a metodologia utilizada para o desenvolvimento do projeto, assim como o seu planeamento. No capítulo 4 é apresentado a arquitetura geral do sistema. No capítulo 5 são descritas as tecnologias utilizadas no desenvolvimento, incluindo as ferramentas utilizadas no desenvolvimento e as ferramentas utilizadas para o design. No capítulo 6 é apresentado o trabalho realizado durante o estágio, incluindo a XTranPassenger API e a aplicação Carris. No capítulo 7 são apresentadas as conclusões, onde são discutidos os resultados obtidos e o trabalho futuro.

2. Estado da arte

Neste capítulo descreve-se o estado da arte referente às várias tecnologias envolvidas na realização deste projeto. Inicialmente é feita uma comparação entre as duas plataformas móveis utilizadas na realização do projeto, Android e iOS. Para além disto são apresentadas as principais aplicações móveis, relevantes para o desenvolvimento deste projeto, destinadas ao transporte público. Após isso, efetua-se uma pequena análise sobre o desenvolvimento de aplicações nativas e híbridas incluindo as suas principais vantagens e desvantagens.

2.1. Plataformas móveis

Com a evolução da tecnologia eletrónica existe um aumento exponencial de características e funcionalidades que um dispositivo móvel tem vindo a ganhar. Esta tecnologia não parou de evoluir propagando-se de forma a facilitar as atividades diárias do utilizador. Com isso, possibilitou existir alguma variedade de modelos e plataformas, sendo as mais conhecidas e usadas:

- Android
- iOS

2.1.1. Android

O Android é um sistema operativo baseado no *software* Linux [7] e é desenvolvido pela Google. Tem como principal característica ser *open source* de modo que este sistema seja facilmente customizável. Este sistema é projetado para dispositivos móveis (*smartphones* e *tablets*), para TV (usando o Android TV), para o Carro (usando o Android Auto) ou até para o *smartwatch* (usando o Android Wear). As aplicações móveis são escritas utilizando o *Software Development Kit* (SDK) do Android e é recomendado a utilização do *Integrated Development Environment* (IDE) Android Studio [8], uma vez que este é o IDE oficial do Android. Também é possível utilizar outros IDE's que suportem a linguagem Java ou Kotlin, tais como o Eclipse [9] ou o InteliJ [10]. Com o Android Studio é oferecido ao programador uma interface gráfica para a realização de interfaces, uma ferramenta de *debug* e um emulador Android para o programador testar as suas aplicações sem necessitar de um dispositivo físico [11].

A versão de Android mais recente é o 8.1 (Oreo) tendo sido lançado em dezembro de 2017, no entanto só alguns dispositivos têm acesso a esta versão. Neste momento a versão mais utilizada pelo mercado é a versão 6.0 (Marshmallow) como poderá ser visualizado na Figura 1.

.

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.3%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.4%
4.1.x	Jelly Bean	16	1.7%
4.2.x		17	2.6%
4.3		18	0.7%
4.4	KitKat	19	12.0%
5.0	Lollipop	21	5.4%
5.1		22	19.2%
6.0	Marshmallow	23	28.1%
7.0	Nougat	24	22.3%
7.1		25	6.2%
8.0	Oreo	26	0.8%
8.1		27	0.3%

Data collected during a 7-day period ending on February 5, 2018.
Any versions with less than 0.1% distribution are not shown.

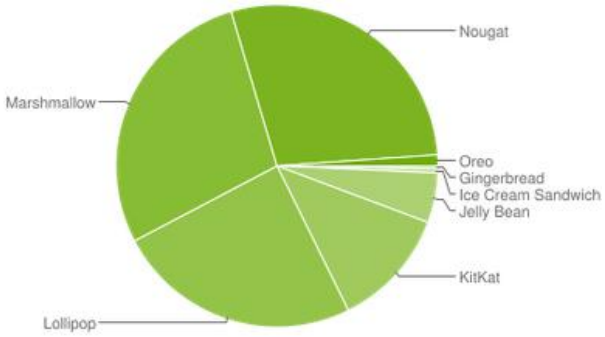


Figura 1 - Distribuição das versões de dispositivos Android[1]

2.1.2. iOS

O iOS é uma plataforma móvel proprietária da Apple que é baseada no Sistema Operativo Darwin [12], destinada aos dispositivos da marca Apple, nomeadamente o iPhone, o iPad, e o iPod Touch. Esta plataforma contém algumas aplicações padrão tais como, o *browser* safari, o *iCloud*, o *Finder*, entre outros. Os programadores que pretendam criar novas aplicações deverão usar o SDK destinado ao iOS, que inclui o IDE Xcode [13]. Uma das principais características do Xcode é a *Interface Builder* (ecrã principal do design) [14]. Este assistente mostra o que se

[1] **https://developer.android.com/about/dashboards/index.html [Acedido a 21-Mar-2018]**

está a ser editado, como o cabeçalho, a superclasse ou o controlador. Além desta característica, o Xcode é uma ferramenta de *debug* e um simulador de ambientes iOS. Este SDK apenas permite ser instalado em ambientes Apple com o sistema operativo macOS, tornando a sua programação limitada a vários programadores.

A versão de iOS mais recente é o iOS 11, tendo sido lançado em setembro de 2017. É utilizado por cerca de 79% dos dispositivos Apple como se pode verificar na Figura 2.

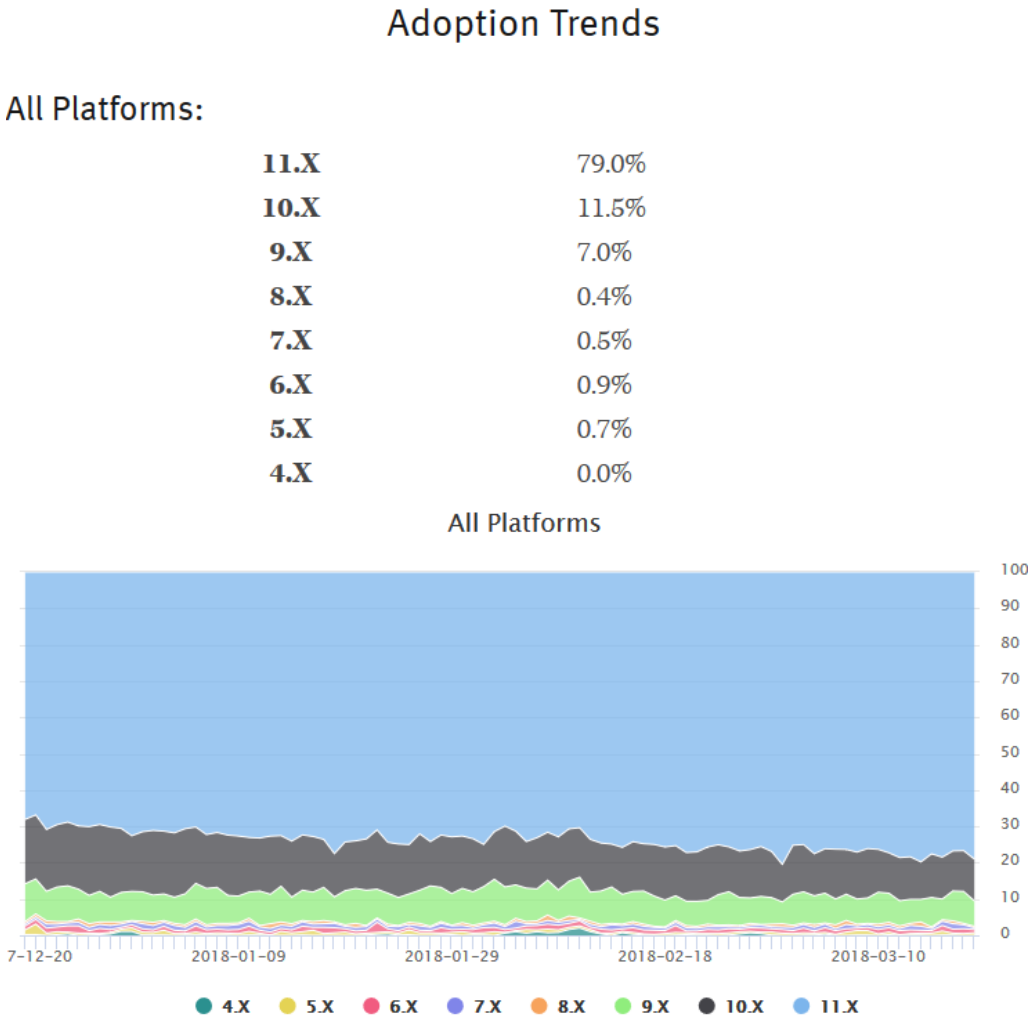


Figura 2 - Distribuição das versões de dispositivos iOS[2]

[2] **https://david-smith.org/iosversionstats/ [Acedido a 21-Mar-2018]**

2.2. Comparação entre plataformas móveis

Na Tabela 1 é feita uma comparação entre as duas principais plataformas móveis, Android e iOS.

Tabela 1 - Comparação entre Plataformas móveis

Plataformas Móveis	Android	iOS
IDE	Android Studio (recomendado), Eclipse, InteliJ	Xcode
Sistema Operativo suportado	Linux, Windows e MacOS	MacOS
Linguagens de Programação	Java (parte do código pode ser C/C++), Kotlin	Objective C, Swift
Aplicação	iOS App Store Package (IPA)	Android Package (APK)

Pela análise da Tabela 1, pode-se concluir que existe uma maior versatilidade no desenvolvimento de aplicações móveis Android, por ser a única linguagem que pode ser programada usando qualquer Sistema Operativo (Linux, Windows e MacOS). Na linguagem iOS existe a desvantagem de só ser possível ser programada no Sistema Operativo proprietário da Apple. Em relação às linguagens de programação utilizadas no desenvolvimento, as duas plataformas utilizam linguagens diferentes, no entanto, ambas são baseadas na programação orientada a objetos (POO). O Android é desenvolvido utilizando a linguagem Java ou Kotlin, sendo que o Kotlin tem uma interoperabilidade total com o Java [15]. Em contrapartida, o iOS poderá ser desenvolvido utilizando o Objective C ou o Swift.

As principais plataformas móveis possuem características semelhantes, como por exemplo o suporte do *multi-touch* e do *multitasking*. No entanto, elas também contêm algumas características próprias, como por exemplo, no iOS é possível transferir informação através do *clipboard*, entre dispositivos da própria marca [16]. No Android é mais fácil customizar o dispositivo, com temas, *widgets*, entre outros.

Na tabela seguinte estão representadas as principais características/diferenças entre estas duas plataformas [17] [18].

Tabela 2 - Principais características do Android e iOS

Sistemas Operativos Móveis	Android	iOS
Suporte ecrã tátil	Sim	Sim
Clipboard entre vários dispositivos	Não	Sim
Customização	Muito customizável	Limitado
Multi-Touch	Sim	Sim
Comandos de voz	Google Now, Google Assistant	Siri
Aplicação de mapas	Google Maps	Apple Maps (Google Maps disponível via download)
Multitasking	Sim	Sim (a partir do iOS 7)
Store	Play Store	App Store
Updates de firmware	Sim	Sim
Segurança	Sujeito a malware	Sim
Fragmentação de versões	Bastante evidente	Pouco evidente
Suporta Tablets	Sim	Sim
Suporte de hardware	Grande variedade de hardware suportado	iPhone, iPod touch e iPad

Como se pode verificar pela Tabela 2, existe uma grande evidência de fragmentação das versões no sistema operativo Android, devido à enorme variedade de modelos disponíveis. Isso torna-se um problema quando muitos dos utilizadores Android utilizam os dispositivos com versões mais antigas do Sistema Operativo. Desta forma, o dispositivo fica mais vulnerável e sujeito a possíveis ataques. No caso da Apple este problema já não é tão evidente dado que existem menos modelos de dispositivos devido à pouca fragmentação das versões do iOS [19].

Este problema encontra-se refletido na Figura 3 e Figura 4, onde é apresentado a distribuição das versões do sistema operativo Android e iOS.

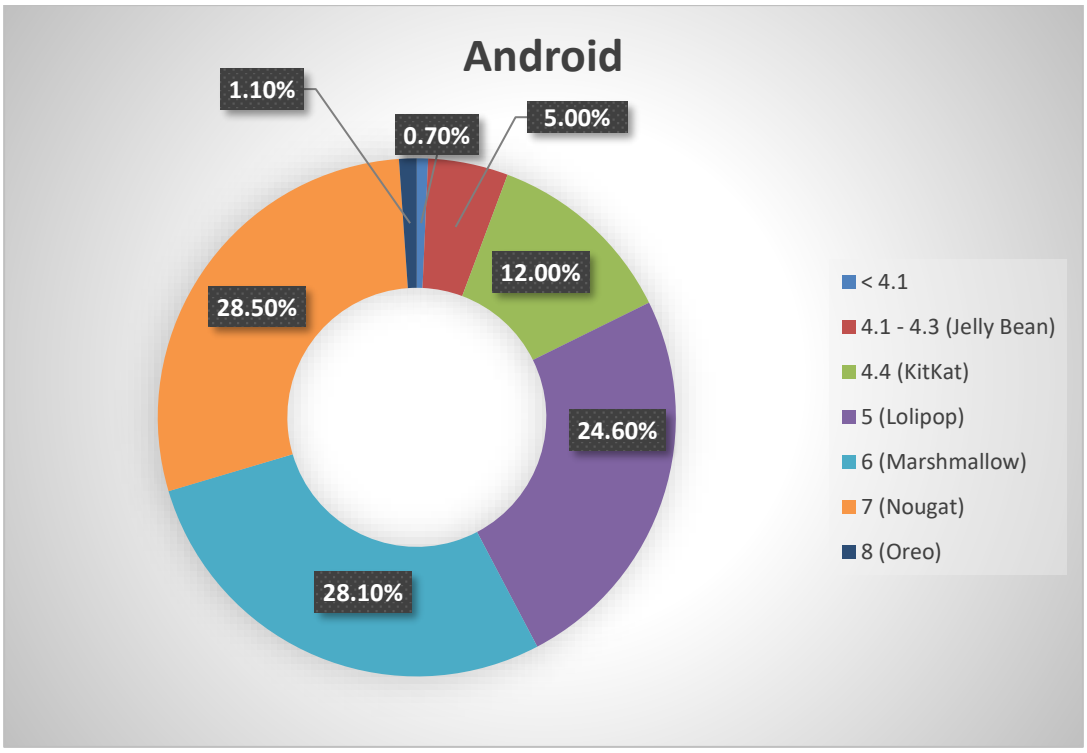


Figura 3 - Distribuição das versões no sistema operativo Android

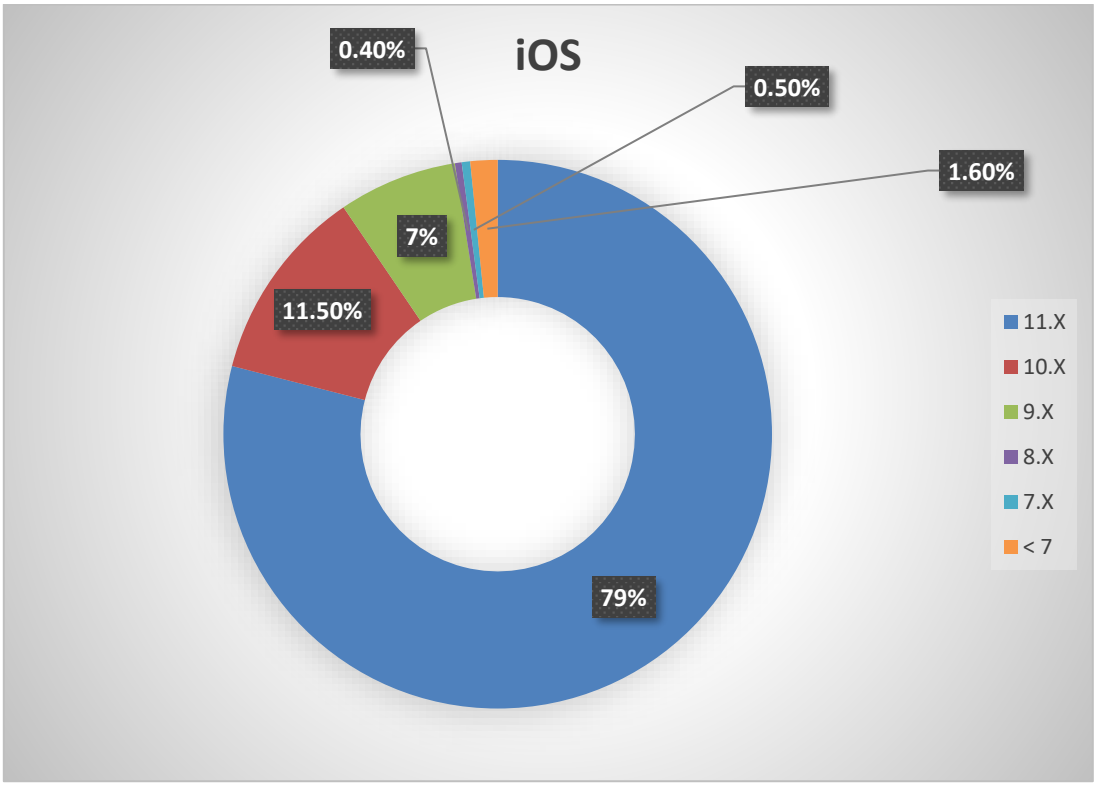


Figura 4 - Distribuição das versões no sistema operativo iOS

Pela análise dos gráficos anteriores pode-se analisar que as versões que estão a ser mais utilizadas pelos utilizadores iOS são:

- iOS 9 com 7% dos utilizadores;
- iOS 10 com 11.5% dos utilizadores;
- iOS 11 com 79% dos utilizadores;

Pelos utilizadores Android são:

- Versão 5 (Lolipop) com 24.6% dos utilizadores;
- Versão 6 (Marshmallow) com 28.1% dos utilizadores;
- Versão 7 (Nougat) com 28.5% dos utilizadores.

A fragmentação é uma fraqueza do ecossistema Android, um problema para os programadores que contribuem para o alcance global do sistema operativo. Os dispositivos Android existem em grande número (Figura 5), com diferentes formas e tamanhos, com níveis de desempenho e tamanhos de ecrã muito diferentes. Além disso, existem muitas versões diferentes do Android que estão simultaneamente ativas ao mesmo tempo. Assim o desenvolvimento de aplicações que funcionam em toda a gama de dispositivos Android é desafiador e demorado [20].

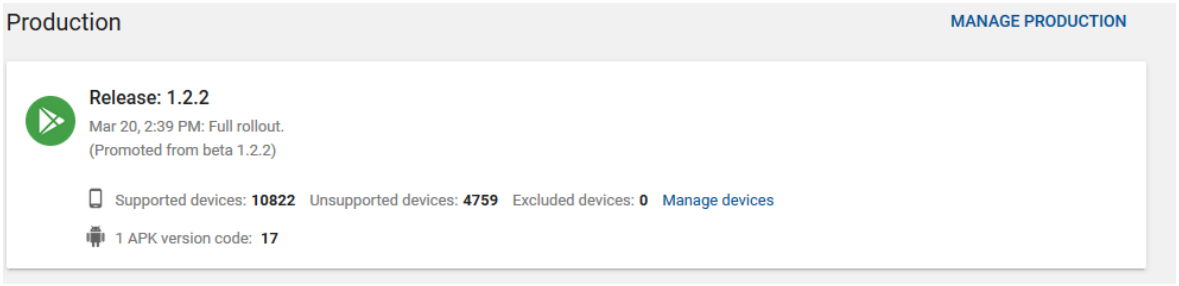


Figura 5 - Número de dispositivos Android suportados a partir da versão 4.2

2.3. Quota de mercado

As vendas globais de *smartphones* para utilizadores finais totalizaram cerca de 380 milhões de unidades no primeiro trimestre de 2017, um aumento de 9,1% em relação ao primeiro trimestre de 2016 [21].

No mercado dos sistemas operativos, a batalha está claramente entre Android e iOS. Com os outros sistemas a lutar por impulso (por exemplo, o Windows Phone), o Android aumentou o seu mercado em 2% em relação ao ano anterior (Tabela 3). A crescente aceitação das marcas

chinesas nos mercados globais está a impulsionar o domínio do sistema operativo Android. Além disso, com o anúncio da sua nova versão (Android Oreo) o sistema operativo está a caminhar para um crescimento contínuo.

Tabela 3 - Quota de mercado das plataformas móveis de 2016 e 2017

Plataforma	Mercado no ano de 2016 (%)	Mercado no ano de 2017 (%)
Android	84.1	86.1
iOS	14.8	13.7
Outros	1.1	0.2

2.4. Aplicações móveis destinadas ao transporte público

Hoje em dia, com a presença de tecnologia sofisticada e a diversidade de dispositivos móveis, existe a possibilidade de tornar simples as tarefas que rodeiam o dia--a-dia do utilizador. Para isso são criadas aplicações móveis que, de certa forma, ajudam a orientar e informar o utilizador sobre o ambiente onde se encontra. O uso desta tecnologia tem-se tornado cada vez mais comum e permite que o tempo gasto à espera do autocarro seja usado de forma mais produtiva [22].

Esta seção serve para apresentar as principais aplicações destinadas aos passageiros de transportes públicos, nomeadamente o Citymapper, o Lisboa Move-Me, o Google Maps, o Moovit e por fim a XtraN Passenger app.

2.4.1. Citymapper

O Citymapper é uma aplicação desenvolvida pela Citymapper Limited em 2012 e é bastante utilizada na área dos transportes públicos urbanos, nomeadamente no uso do autocarro, do elétrico, do metro e também do comboio. Ela inclui uma ampla gama de funcionalidades, nomeadamente o planeador de viagem que utiliza informações em tempo real e permite selecionar os locais de partida e destino. Para além destas funcionalidades, apresenta variações

de rotas ao utilizador que envolvam percursos de autocarros, metro, percursos sem chuva, entre outros, incluindo informações sobre o contexto atual, tais como o tempo no local de destino. A aplicação indica questões cruciais para o utilizador frequente, tais como o encerramento de uma linha ou a congestão da via [23] e é útil por esse mesmo motivo [24]. Esta aplicação é adaptada para diversas cidades do mundo tais como, Lisboa, Londres, Paris, Bruxelas, Madrid, entre outras.

A aplicação Citymapper tem como principais funcionalidades [25]:

- Informações detalhadas, de todo o percurso, no cálculo de uma rota para viagens de autocarros da Carris, para o metro de Lisboa, Comboios, Elétricos e Táxi;
- Informação sobre as paragens dos autocarros da Carris com respetivos horários de partida;
- Mapa do metro e dos comboios urbanos de Lisboa disponíveis em modo *offline*;
- Possibilidade de guardar percursos de transportes em modo *offline*;
- Possibilidade de guardar informação sobre as paragens da Carris, estações do metro de Lisboa ou estações de comboio da CP. Sendo também possível guardar lugares favoritos;
- Notificações sobre o percurso casa - trabalho, mesmo quando o ecrã se encontra bloqueado;
- Alerta para sair quando o autocarro estiver a chegar à paragem destinada.
- O utilizador pode enviar a sua localização atual no percurso que se encontra a realizar, de forma a que os amigos/familiares possam acompanhar em tempo real. Deste modo permite ter uma estimativa de chegada em tempo real incluindo possíveis imprevistos na estrada.

A Figura 6 mostra uma visão geral da aplicação **Citymapper** para iOS.

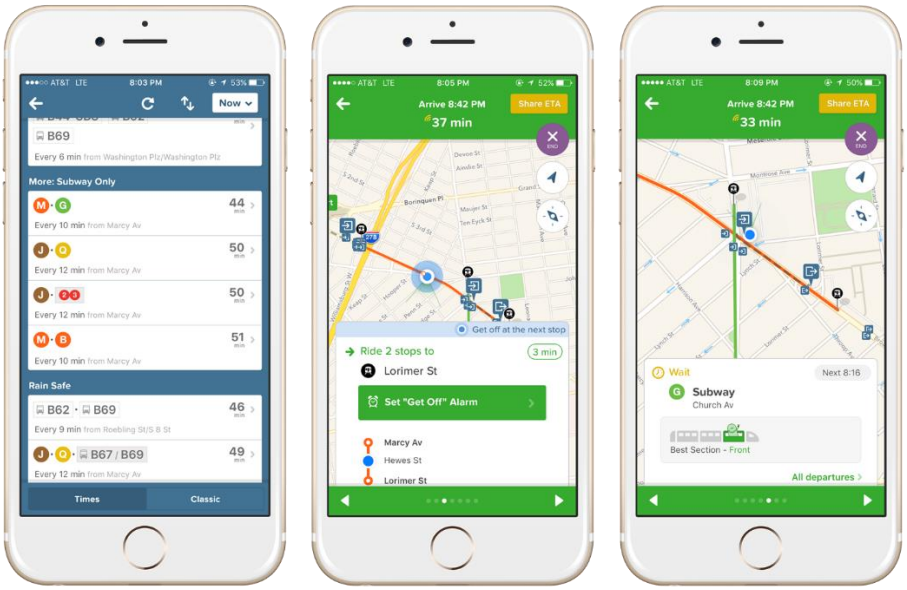


Figura 6 - Visão geral do Citymapper[3]

2.4.2. Google Maps

O Google Maps é uma aplicação para *smartphones* desenvolvida pela Google para o sistema operativo Android, em 2008 [26] e para iOS em 2012 [27]. Esta aplicação disponibiliza indicação de direção, estado do trânsito em tempo real, vista de satélite, *street view*, mapas *offline* e informações locais para que os utilizadores das aplicações de transportes públicos se possam orientar. A aplicação é suportada para os sistemas operativos iOS, Android, Windows Mobile, BlackBerry OS, entre outros [28] [29].

A aplicação Google Maps apresenta como principais funcionalidades [30]:

- Mapas *offline* para pesquisar e navegar sem ligação à Internet;
- *Street view* e imagens de interiores de restaurantes, lojas, museus entre outros;
- Mapas interiores de edifícios para encontrar rapidamente o que o utilizador procure em locais de grande dimensão, como aeroportos, centros comerciais e estádios;

[3] http://travelamusement.com/wp-content/uploads/2016/09/mobile-app-3.png

- Mapas precisos e abrangentes referentes a duzentos e vinte países, horários de transportes públicos, incluindo mapas para mais de quinze mil cidades e informações detalhadas de empresas em mais de 100 milhões de locais;
- Restaurantes com classificações realizadas por outros utilizadores e lojas locais;
- Possibilidade de guardar locais onde o utilizador pretenda ir ou que visita com maior regularidade: deste modo permite encontrar estes locais rapidamente em qualquer computador ou dispositivo;
- Trânsito com *Estimated Time of Arrival* (ETAs) e condições de trânsito em tempo real;
- Informações de trânsito em tempo real para autocarros e comboios;
- Redirecionamento automático baseado em trânsito, acidentes e fechos de estradas, em tempo real;
- Orientação de faixa;
- Indicação de paragens ao longo do trajeto planeado, como estações de serviço e cafés.

A Figura 7 mostra uma visão geral da aplicação **Google Maps** para iOS.

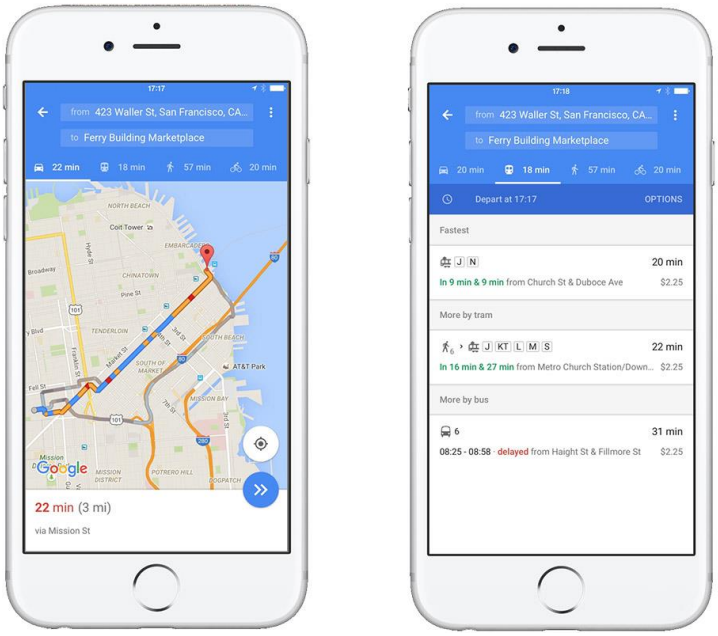


Figura 7 - Visão geral do Google Maps[4]

[4] https://s.aolcdn.com/hss/storage/midas/29e63b9b07c13f8b6215b83a194e9fd5/202731027/google-maps-travel-time-ios-lede.jpg

2.4.3. Move-Me

O Move-Me é um serviço baseado em localização, distribuído nos mercados móveis (Android e iOS) em Lisboa (Aplicação Lisboa MOVE-ME) [31], Porto (aplicação MOVE-ME.AMP) [32] e Coimbra (Aplicação Coimbra MOVE-ME) [33], desenvolvido em 2012 [34] pela empresa de software OPT [35]. Estas aplicações acrescentam valor à rede de transporte local, permitindo aos utilizadores a pesquisa por autocarros que passem numa determinada paragem, a possibilidade de definir uma rota pela rede e a obtenção de paragens perto do utilizador ou qualquer posição geográfica. O sistema, no entanto, depende do envolvimento dos operadores de redes de transporte e está a guardar dados que, se analisados, podem resultar em métricas importantes para aumentar a eficácia da rede de transportes e a satisfação do utilizador [36].

A aplicação Move-Me apresenta as seguintes funcionalidades principais [32]:

- Planeamento de percursos intermodais (combinação de dois ou mais tipos de transporte) em tempo real e para diferentes meios de transporte na cidade;
- O utilizador pode pesquisar por próximos autocarros;
- O utilizador pode consultar os tempos de espera e os destinos associados aos próximos autocarros a passar num determinado local, indicado pelo utilizador;
- É permitido também a identificação das paragens que se encontram mais próximas da posição do utilizador, os próximos autocarros e os respetivos destinos;
- Através dos pontos de interesse mais próximos, é possível selecionar um raio de pesquisa e assim obter um percurso até ao ponto de interesse escolhido;
- É possível ainda fazer o cálculo de rota, onde o utilizador pode calcular uma rota que interligue vários pontos por si definidos.

A Figura 8 mostra uma visão geral da aplicação **MOVE-ME.AMP** para Android.

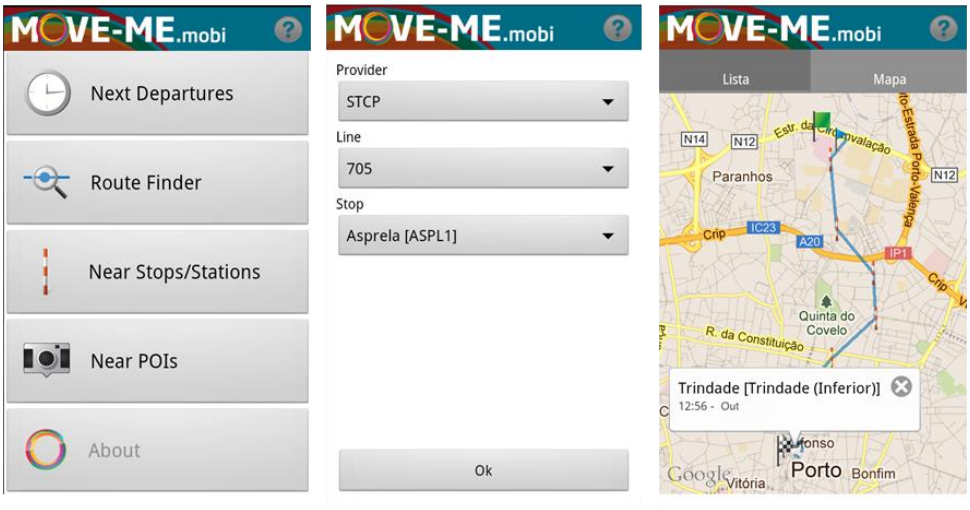


Figura 8 - Visão geral do MOVE-ME.AMP[5]

2.4.4. Moovit: Bus Train Live Info

O Moovit é uma aplicação desenvolvida pela start-up israelense Tranzmate para Android e iOS em 2012, que oferece informações de transporte público em tempo real e navegação GPS nas principais cidades da América do Norte, América do Sul, Europa e Ásia. É uma ferramenta que obtém informações que permitem planear uma viagem utilizando os transportes públicos, nomeadamente, metro, autocarros, comboios e estações de bicicletas [28][37].

A aplicação **Moovit** é apresentada com as seguintes funcionalidades principais [37]:

- Itinerários detalhados – Instruções passo-a-passo de um ponto A ao ponto B;
- Notificações para sair na próxima estação;
- Locais favoritos – Guardar locais favoritos, como a localização de Casa ou do Trabalho;
- Estimativas de tempo de uma paragem;
- Estações à volta da posição atual do utilizador incluindo as suas linhas ativas;
- Estações de comboio ou de metro – Informações importantes tais como horários ou a última viagem da noite;

[5] http://www.opt.pt/imagens/image/move-me_imagens_EN.png

- Horários das Linhas e Mapas – Visualização de horários, estações, rotas e mapas para uso *offline*;
- Alertas de Serviço – Notificações caso exista algum "imprevisto" na rota;
- Possibilidade ao utilizador de guardar as suas linhas favoritas.

A Figura 9 mostra uma visão geral da aplicação **Moovit** para Android.

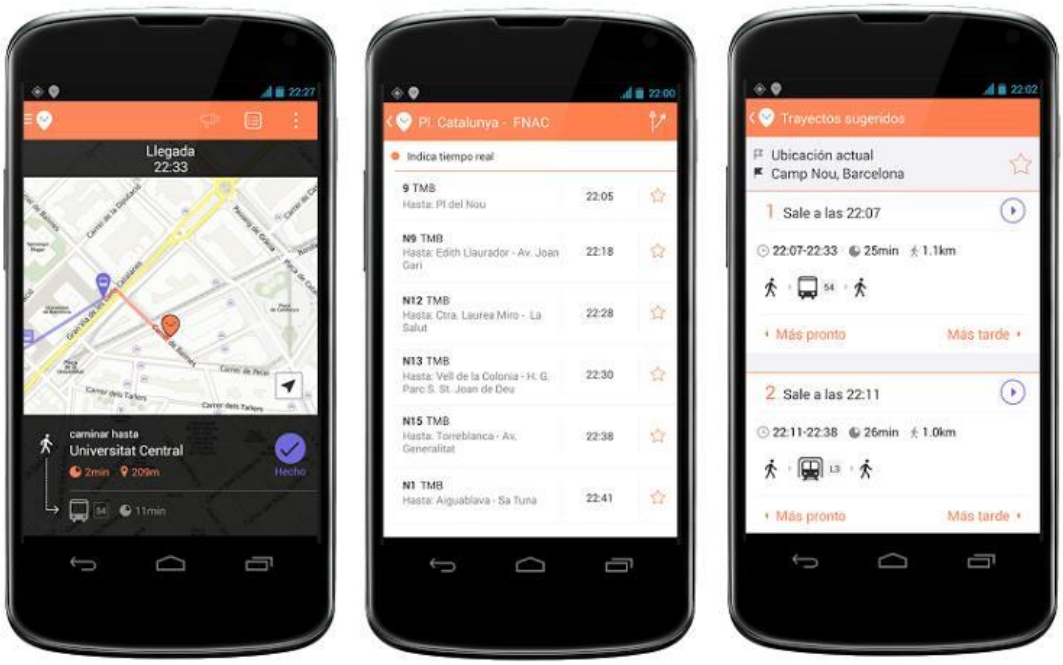


Figura 9 - Visão geral do Moovit[6]

2.4.5. XTraN Passenger app

O XtraN Passenger é um sistema de informação de transporte público desenvolvido pela Tecmic e é utilizado por diversos operadores de transporte público em Portugal e no Brasil. Este sistema é personalizado para cada cliente e está projetado para produzir informações detalhadas e estatísticas em tempo real sobre a atividade das frotas de transporte, dando capacidade aos operadores de melhorar os seus serviços [38].

[6] http://geoawesomeness.com/wp-content/uploads/2015/01/Moovit.jpg

Na Figura 10 está representada a arquitetura deste sistema e a ligação entre os vários módulos do mesmo.

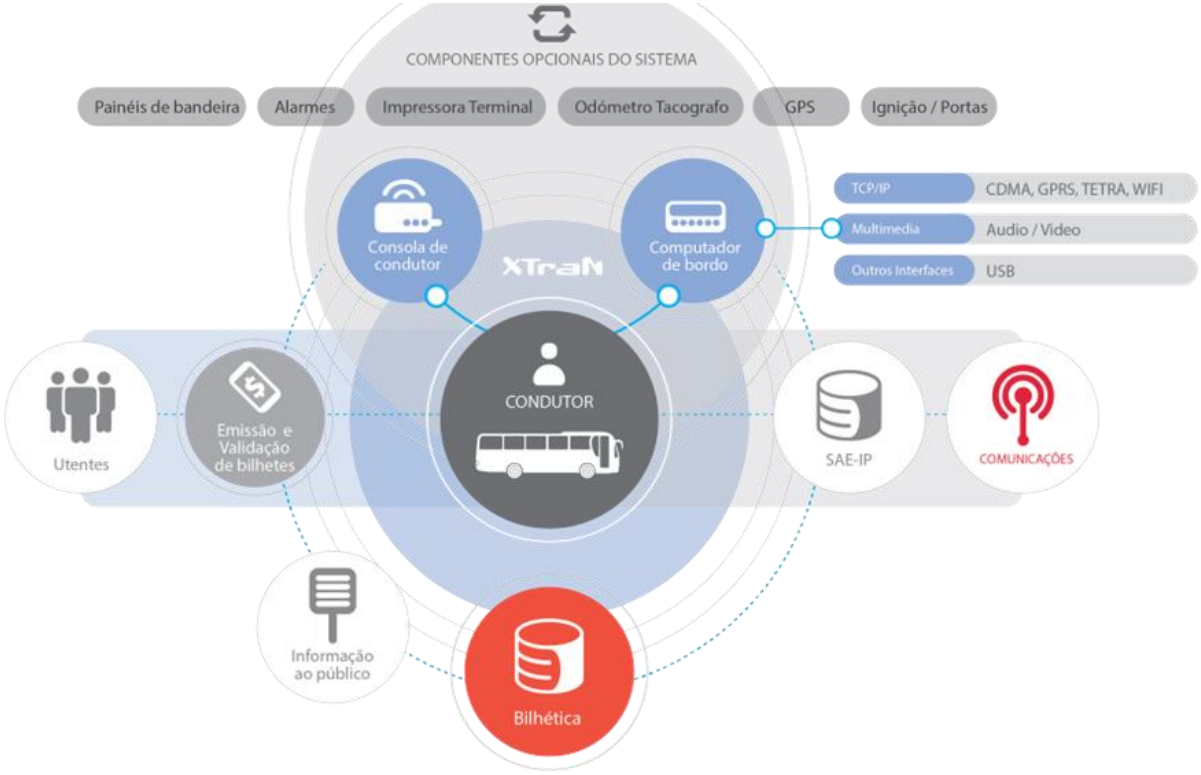


Figura 10 - Arquitetura da solução XTranPassenger[7]

O XtraN Passenger app é um protótipo funcional concebido para dispositivos Android, desenvolvido com a ferramenta Xamarin. Este protótipo foi inicialmente desenvolvido pela Tecmic, para um cliente no Brasil, mas não teve sucesso na sua apresentação principalmente devido ao design pouco intuitivo, sendo necessário a criação de ecrãs de ajuda de modo a facilitar a utilização da aplicação. Foi assim preparado como resposta às necessidades de apresentar a informação de forma mais evidente e móvel para o utilizador. É assim destinada e orientada às necessidades do passageiro e desta forma tem como principal objetivo fornecer dados em tempo real sobre autocarros, paragens e linhas [39].

Apesar do seu insucesso este protótipo serviu como ponte entre a Tecmic e a Carris na medida em que, com uma reestruturação de funcionalidades, foi apresentada uma versão melhorada deste mesmo protótipo ao cliente.

[7] http://www.tecmic.com/wp-content/uploads/2014/06/arquitetura-passenger_wide.png

Com a aceitação do cliente foi assim criado um caderno de encargos com novas funcionalidades as quais foram extraídas para serem integradas neste relatório (Anexo A - Requisitos Funcionais da Aplicação Carris).

Assim este protótipo insere-se numa categoria especial porque, para além de ter sido o ponto de partida, foi a partir deste molde em que o projeto App Carris se realizou.

As principais funcionalidades do protótipo desenvolvido são:

- Contextualização geográfica do utilizador;
- Consulta das linhas disponibilizadas pelo operador;
- Consulta da informação/localização de todas as linhas;
- Consulta de informação por paragem;
- Consulta da localização geográfica dos próximos autocarros;
- Consulta de informação sobre as paragens mais próximas.

Na Figura 11 é apresentada a visão geral do protótipo apresentado à Carris.

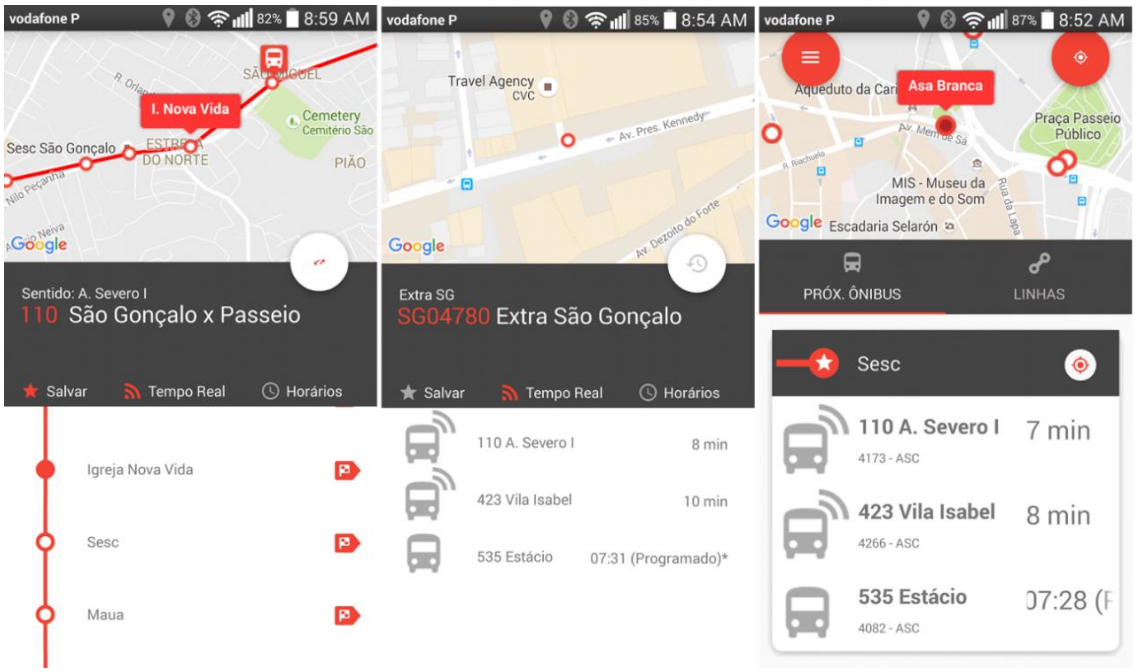


Figura 11 - Visão geral da XtraN Passenger app

2.5. Resumo da comparação entre as aplicações móveis

Tabela 4 - Comparação entre as aplicações móveis

Aplicações móveis	Citymapper	Google Maps	Move-me	Moovit	Xtran Passsenger app
Uso de Gps	Sim	Sim	Sim	Sim	Sim
Planeamento de um percurso	Sim	Sim	Sim	Sim	Não
Informação de tempos de espera numa paragem	Sim	Sim	Sim	Sim	Sim
Informação de uma linha	Sim	Sim	Não	Sim	Sim
Guardar paragens como favoritas	Sim	Sim	Sim	Sim	Sim
Guardar linhas como favoritas	Sim	Não	Não	Sim	Sim
Guardar percursos em modo *offline*	Sim	Sim	Não	Sim	Não
Classificações e modo social	Sim	Sim	Não	Sim	Não
Uso da aplicação em diferentes cidades	Sim	Sim	Não	Sim	Não
Paragens próximas em relação à localização do utilizador	Sim	Sim	Sim	Sim	Sim

Pela análise da Tabela 4 pode-se concluir que todas as aplicações apresentadas fazem uso do GPS, para informar ao utilizador de quais são suas paragens mais próximas. É de notar que

todas permitem guardar paragens bem como a apresentação dos seus tempos de espera até um autocarro chegar.

Em relação ao planeamento de um percurso todas as aplicações com a ressalva da **Xtran Passenger app** têm essa utilidade.

Com a exceção da aplicação **Move-me,** é possível com que cada uma das aplicações apresente a informação de uma linha, incluindo as paragens que a constituem. No que se refere a guardar linhas como favoritas só o **Citymapper**, o **Moovit** e a **Xtran Passenger app** é que possuem essa funcionalidade.

2.6. Desenvolvimento de aplicações móveis nativas vs. híbridas

As aplicações móveis híbridas combinam os recursos das aplicações Web e das aplicações móveis "nativas". Como são aplicações Web, elas são implementadas com linguagens para desenvolvimento web, independentes da plataforma, tais como HTML e JavaScript. Como aplicações nativas, estas aplicações têm acesso direto aos recursos locais do dispositivo - sistema de arquivos, localização, câmara, contactos, etc. [40].

As aplicações nativas são normalmente desenvolvidas utilizando linguagens de programação, como o Java/Kotlin para o Android, Objective-C/Swift para iOS, ou C# para Windows Phone. As *Application Programming Interfaces* (APIs) nativas são fornecidas para o programador como parte do SDK da plataforma. As APIs da plataforma são normalmente concebidas para fornecer aplicações nativas com acesso fácil às capacidades de hardware, como por exemplo, a câmara do dispositivo ou o emparelhamento de Bluetooth. Além disso, os utilizadores podem ser capazes de usar essas aplicações sem uma conexão com a Internet [41].

As aplicações híbridas estão inseridas numa categoria específica de aplicações Web que estendem o ambiente de aplicação baseado na Web através do uso de APIs de plataformas nativas disponíveis num determinado dispositivo. O padrão de desenho de uma aplicação híbrida é aplicável aos ambientes móveis e de desktop [42].

As principais diferenças entre desenvolver aplicações nativas e híbridas estão representadas na Tabela 5.

Tabela 5 - Diferenças entre aplicações nativas e híbridas[8]

Plataformas Móveis	Nativo	Hibrido
Linguagem de desenvolvimento	Só nativo	Nativo e web/ só web
Acesso ao dispositivo	Completo	Completo
Características específicas do dispositivo	Java (parte do código pode ser C/C++), Kotlin	Objective C, Swift
Velocidade	iOS App Store Package (IPA)	Android Package (APK)
Loja de aplicações	Disponível	Disponível
Processo de aprovação	Mandatório	Baixa sobrecarga
Portabilidade de código	Nenhuma	Alta
Gráficos avançados	Altos	Moderados
User interface (UI) / User experience (UX)	Alto	Moderado
Acesso a API's Nativas	Alto	Moderado
Custo de desenvolvimento	Custo elevado	Custo razóavel

As aplicações híbridas adicionam uma camada extra entre o código fonte e a plataforma móvel alvo: uma *framework* híbrida particular, como Ionic [43], Cordova [44], entre outros. O resultado da utilização destas ferramentas é uma possível perda de desempenho, mas este fator pode variar de aplicação para aplicação. Estas aplicações são adequadas a situações cuja principal prioridade seja o desenvolvimento rápido ou quando existe um custo elevado para o desenvolvimento baseado em plataformas com uma aplicação nativa individual [45].

[8] https://julyrapid.com/wp-content/uploads/2016/10/hybrid-vs-native-apps-comparison-table.png

2.7. Síntese do capítulo

Neste capítulo, fez-se uma resenha das principais aplicações móveis, existentes no mercado, relacionadas com informações sobre diversos tipos de transporte, mas incidindo maioritariamente sobre os transportes públicos. Verificou-se que existe sempre informação sobre as paragens, que o utilizador pode consultar e, uma vez que as aplicações tiram partido do GPS, possibilita o cálculo das suas paragens mais próximas.

Por fim, foi feita uma comparação entre desenvolver aplicações móveis nativas e aplicações híbridas.

No capítulo seguinte irá explicar-se e aprofundar-se a metodologia e o planeamento planejados, no que diz respeito ao trabalho realizado neste estágio. Também irá ser feita uma comparação entre as plataformas móveis, Android e iOS, no que diz respeito ao estudo do mercado.

3. Planeamento e Metodologia

Este capítulo descreve a metodologia de desenvolvimento escolhida para utilizar no desenvolvimento do projeto "App Carris", bem como o respetivo planeamento.

3.1. Metodologia

As metodologias de desenvolvimento de software têm como objetivo estruturar, planear e controlar o processo de desenvolvimento e o ciclo de vida de um software. Existem dois tipos principais de metodologias: as tradicionais e as ágeis [46].

As metodologias tradicionais caracterizam-se por possuírem fases bem definidas no ciclo de vida do software, mais documentação sobre as fases de desenvolvimento do produto e uma maior formalidade na comunicação, entre membros da equipa. Exemplo de algumas metodologias: o modelo em cascata (*waterfall*), a prototipagem (*prototyping*) e o modelo em espiral (*Spiral)* [47].

As metodologias ágeis apresentam vários ciclos incrementais na realização do software. Ao contrário das metodologias tradicionais referidas anteriormente, estas possuem uma menor documentação e um maior foco no desenvolvimento do produto. Para além disso, existe uma comunicação mais informal entre os membros, havendo assim uma maior facilidade na estruturação da equipa. A título de exemplo são apresentadas algumas metodologias como, o *Agile,* a programação extrema (*XP*), o *Scrum,* o *Kanban,* entre outros [48]**.**

Para auxiliar o desenvolvimento do projeto, foi necessária a definição de uma metodologia de suporte ao desenvolvimento. A metodologia utilizada pela empresa é baseada no *waterfall,* mas com adaptações que permitam a navegação bidirecional entre as várias etapas do desenvolvimento [49] [50].

No entanto, considerou-se que a metodologia mais adequada para a realização do projeto seria uma metodologia baseada numa combinação no *Scrum* e no *waterfall*. Esta opção foi fundamentada devido, não só ao envolvimento e à participação constante do cliente durante o desenvolvimento do projeto, mas também ao estado de transição de processos em que a empresa se encontrou. Esta metodologia foi sempre adaptada com as necessidades do projeto, tendo sido apresentadas várias iterações de software funcional através de diversas reuniões com o cliente:

em cada reunião eram discutidos os ajustes a realizar ao projeto pois estes ajustes nunca poderiam afetar os requisitos principais documentados inicialmente.

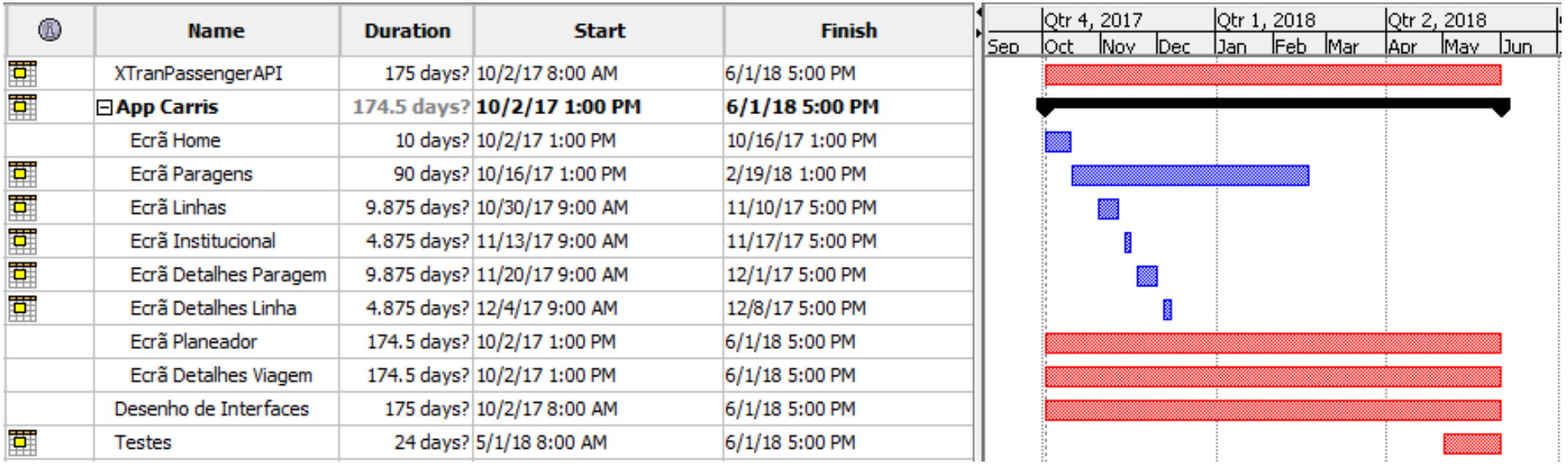
Para a implementação desta metodologia foram seguidos os seguintes princípios da metodologia [51]:

- **Indivíduos e Interações** – Foco nos indivíduos e a interação entre os mesmos, contrariamente à utilização de processos e ferramentas mais complexas;
- **Software exequível** – Prioridade na criação de versões simples e funcionais em vez de se proceder à criação de documentação excessiva;
- **Colaboração do cliente** – Apesar de ter havido uma planificação inicial, durante todo o desenvolvimento do projeto, o cliente foi colaborando com a equipa de desenvolvimento através de reuniões e da comunicação via email ou até mesmo por Skype [52];
- **Respostas rápidas a possíveis mudanças** – Foi necessário dar resposta de forma rápida a alterações ou a novos requisitos que pudessem aparecem durante o desenvolvimento;

Tal como referido anteriormente, o cliente esteve sempre presente durante todo o desenvolvimento deste projeto. Foram realizadas algumas reuniões presenciais, mas devido à distância física entre a equipa de desenvolvimento e o cliente, foram utilizadas diversas formas alternativas de comunicação à distância como o email, as chamadas telefónicas e o Skype.

3.2. Planeamento

O estágio teve início a 2 de outubro de 2017 e terminou a 1 de julho de 2018. O estágio decorreu em contexto empresarial e não tendo sido necessário um período de adaptação porque já estava inserido na equipa de desenvolvimento, nas instalações da empresa, em Leiria. Na Figura 12 está representado um cronograma com as principais tarefas realizadas ao longo do estágio.



Name	Duration	Start	Finish
XTranPassengerAPI	175 days?	10/2/17 8:00 AM	6/1/18 5:00 PM
App Carris	174.5 days?	**10/2/17 1:00 PM**	**6/1/18 5:00 PM**
Ecrã Home	10 days?	10/2/17 1:00 PM	10/16/17 1:00 PM
Ecrã Paragens	90 days?	10/16/17 1:00 PM	2/19/18 1:00 PM
Ecrã Linhas	9.875 days?	10/30/17 9:00 AM	11/10/17 5:00 PM
Ecrã Institucional	4.875 days?	11/13/17 9:00 AM	11/17/17 5:00 PM
Ecrã Detalhes Paragem	9.875 days?	11/20/17 9:00 AM	12/1/17 5:00 PM
Ecrã Detalhes Linha	4.875 days?	12/4/17 9:00 AM	12/8/17 5:00 PM
Ecrã Planeador	174.5 days?	10/2/17 1:00 PM	6/1/18 5:00 PM
Ecrã Detalhes Viagem	174.5 days?	10/2/17 1:00 PM	6/1/18 5:00 PM
Desenho de Interfaces	175 days?	10/2/17 8:00 AM	6/1/18 5:00 PM
Testes	24 days?	5/1/18 8:00 AM	6/1/18 5:00 PM

Figura 12 - Diagrama de Gantt com as atividades desenvolvidas

Inicialmente, para o projeto da Carris, foi necessário realizar uma análise de requisitos e tal como foi referido no subcapítulo 2.4.5, elaborou-se um caderno de encargos que determinou as funcionalidades que iriam ser implementadas na aplicação. Este processo não está representado no gráfico de *Gantt*, Figura 12, uma vez que tinha sido realizado anteriormente ao estágio. Contudo, dado que o estagiário já estava inserido na empresa e acompanhou o processo de definição dos requisitos, considerou-se importante para a contextualização deste projeto, descrevê-lo neste documento.

No decorrer do estágio o planeamento foi realizado com 5 tarefas (XTranPassenger API, App Carris, Desenho de Interfaces e Testes). Para a tarefa da App Carris foi feita uma divisão em subtarefas por ecrãs (Ecrã Home, Ecrã Paragens, Ecrã Linhas, Ecrã Institucional, Ecrã Detalhes Paragem, Ecrã Detalhes Linha, Ecrã Planeador e Ecrã Detalhes Viagem).

Algumas tarefas sofreram atrasos, incluindo o ecrã do planeador e o ecrã dos detalhes de uma viagem, mas foi possível recuperar o tempo na realização de outras tarefas

A aplicação Carris foi realizada desde outubro de 2017 até junho de 2018, ou seja, durante o decorrer do estágio e paralelamente existiu a necessidade de desenvolver a XTranPassenger API que a sustenta. Durante esse tempo também foi feito o desenho das interfaces das aplicações uma vez que era um trabalho continuo e propenso a melhorias, a pedido do cliente.

Na parte final do estágio efetuaram-se testes de integração, de funcionalidade e de usabilidade que permitiram mitigar possíveis erros durante o uso da aplicação.

Inicialmente, com as diversas tarefas que foram proporcionadas durante o desenrolar do estágio, utilizou-se o Trello [53] como ferramenta para organizar as tarefas. Foi utilizada esta ferramenta uma vez que foi bastante utilizada pelos restantes colaboradores da Tecmic, facilitando a comunicação entre os mesmos.

O Trello, representado na Figura 13, é uma ferramenta que permite organizar as tarefas através de cartões, onde é possível a inserção de imagens, comentários, descrições e até atribuir elementos da equipa às tarefas.

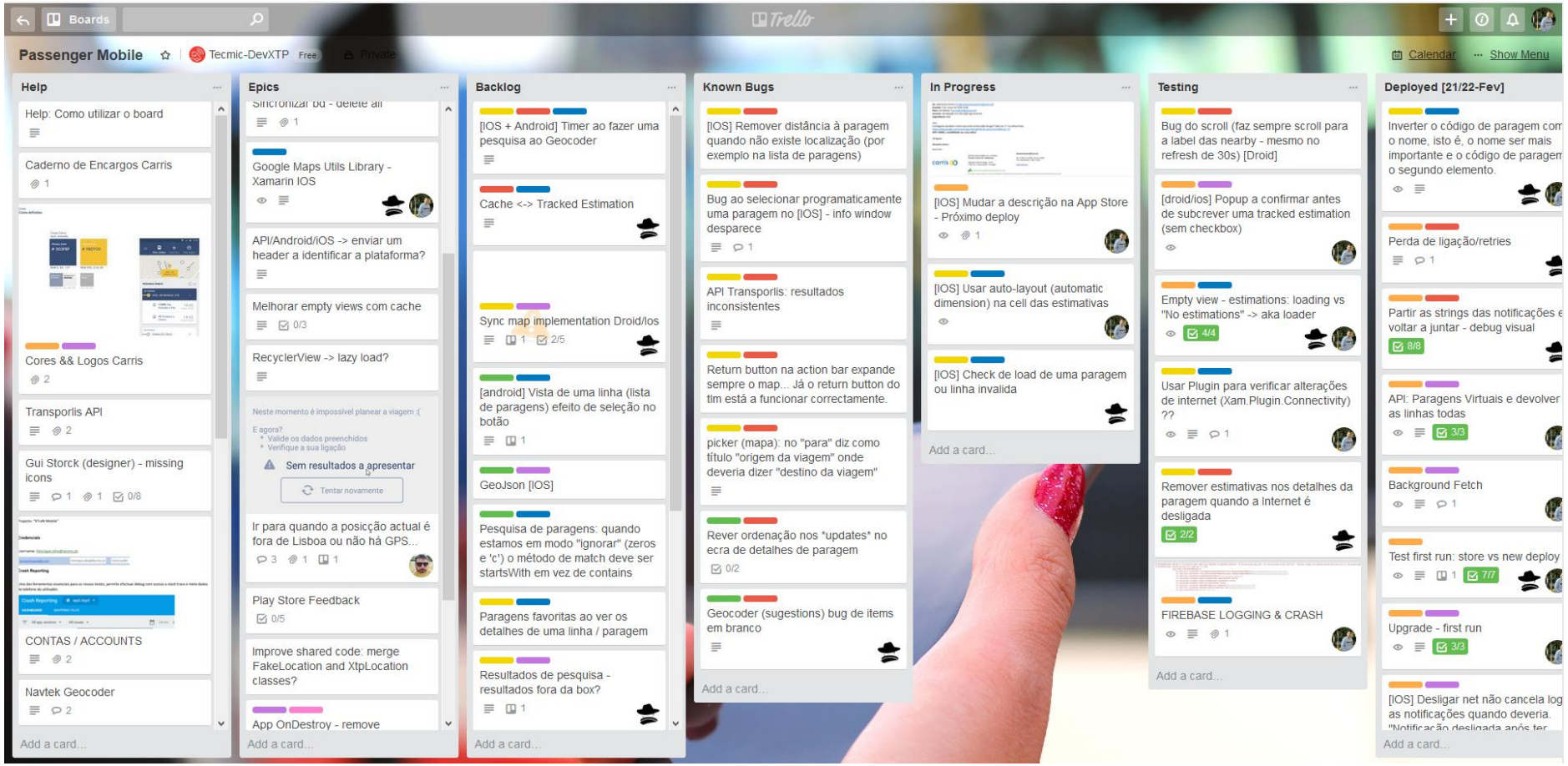


Figura 13 - Visão geral do Trello para a App Carris

No decorrer do estágio a empresa passou do sistema de versões Apache Subversion (SVN) [54] para o Git [55] utilizado o Team Foundation Server (TFS) [56] da Microsoft. O projeto da aplicação Carris foi um dos primeiros projetos a adotar o TFS na Tecmic, tendo sido realizado um processo de migração do projeto (SVN) para um novo repositório Git.

O TFS é uma ferramenta que permite fazer a gestão de projetos e de requisitos para equipas de desenvolvimento ágil. Esta ferramenta proporciona um sistema de controlo de versões, utilizando o Git ou o Team Foundation version control (TFVC). O TFS tem a vantagem de ser compatível com a ferramenta utilizada no desenvolvimento (Visual Studio). Para além da gestão do código, o TFS possui uma ferramenta que permite fazer a gestão das tarefas, o Agile Tools [57]. A partir da transição do projeto Carris, foram inseridas as tarefas através do *board* específico desta ferramenta, (Figura 14), substituindo assim o Trello.

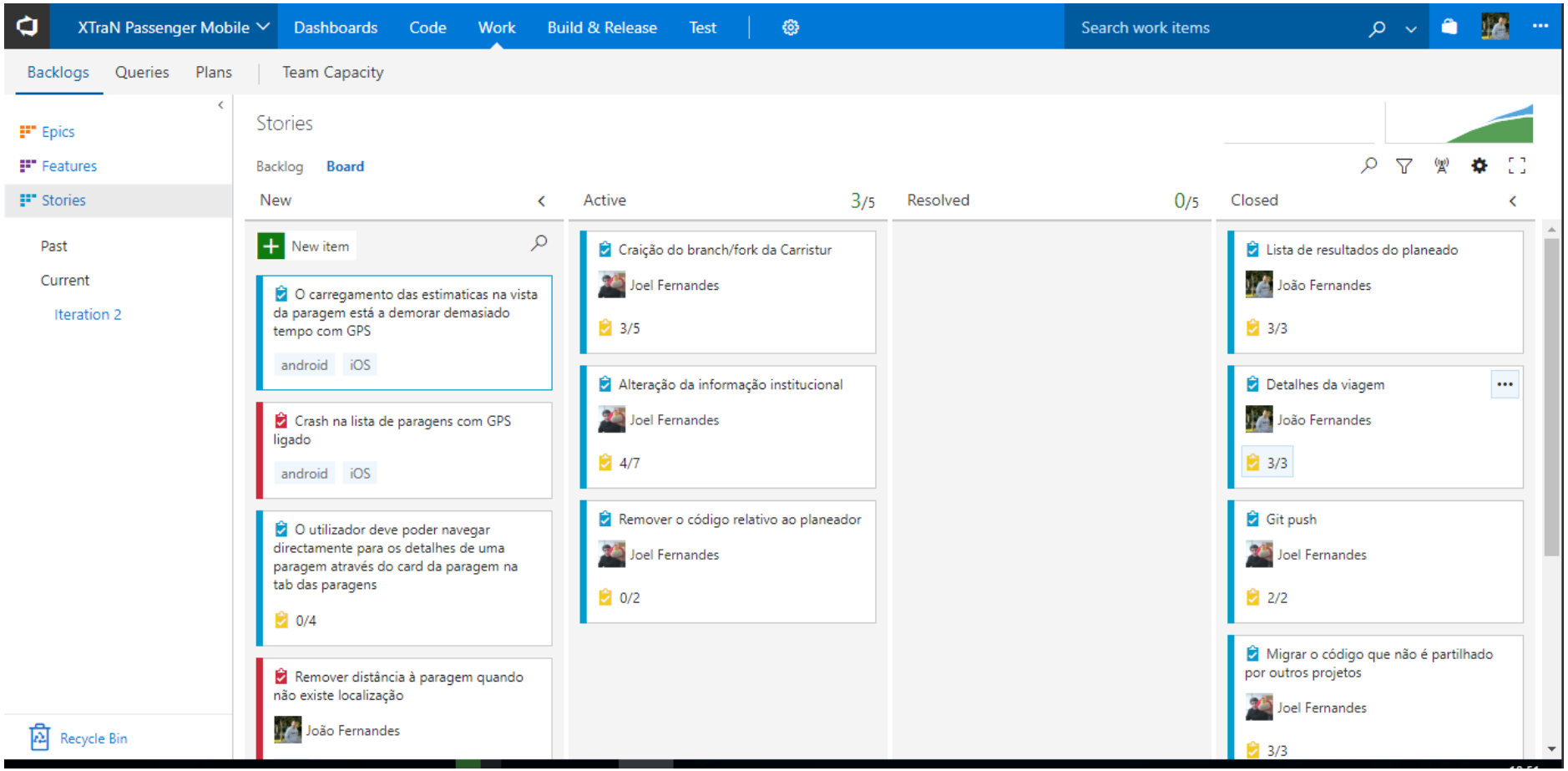


Figura 14 - Visão geral do board do TFS para a App Carris

3.3. Síntese do capítulo

Neste capítulo apresentou-se a metodologia utilizada na realização do projeto incluindo o processo de transição da empresa. Abordou-se ainda o planeamento utilizado durante a realização do estágio, incluindo a explicação de algumas ferramentas utilizadas e apresentou-se o diagrama de *Gantt* onde é descrito o tempo despendido na realização das tarefas propostas.

No capítulo seguinte irá ser abordada a arquitetura geral das aplicações realizadas.

4. Arquitetura geral

Este capítulo tem o objetivo de descrever a arquitetura geral de alto nível das aplicações realizadas. Esta arquitetura consiste na definição dos diversos componentes, nas suas propriedades e de que forma estes se relacionam. Na Figura 15 está representado a arquitetura global das aplicações móveis realizadas.

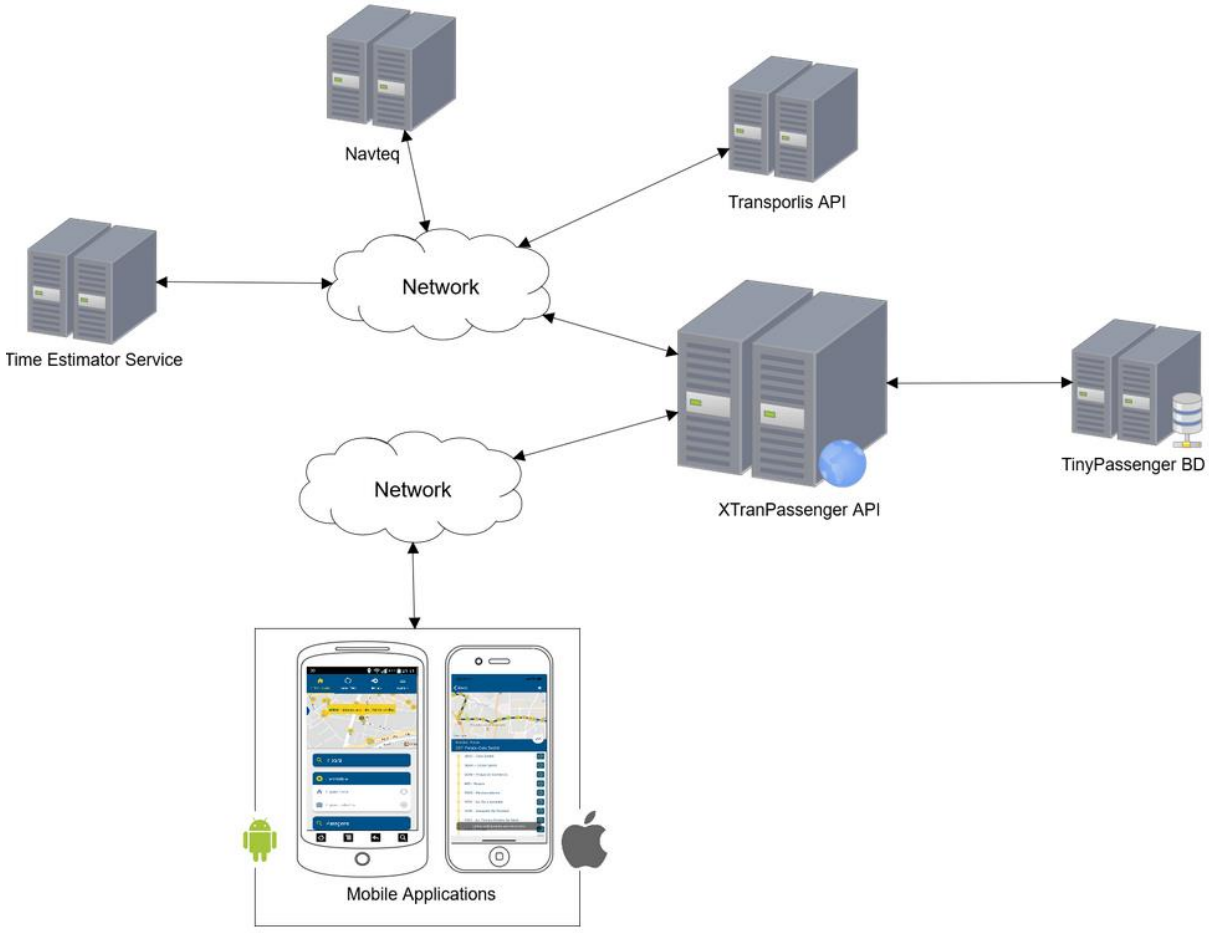


Figura 15 - Arquitetura global das aplicações móveis

Pela arquitetura geral pode-se concluir que as aplicações móveis estão dependentes da XTranPassenger API. A partir desta têm acesso à informação localizada na TinyPassenger BD incluindo também os serviços disponíveis pelos outros módulos como o Time Estimation Service, a Transporlis e o Navteq. Sendo a API a ponte de ligação entre eles, as aplicações não têm acesso direto a estes serviços.

A TinyPassenger BD é a base de dados do sistema onde se encontram os principais objetos utilizados, tais como as paragens e as linhas.

O Time Estimation Service é um serviço utilizado para a recolha de estimativas de tempo de uma ou mais paragens. As estimativas de tempo são apresentadas durante o uso das aplicações em diversas ocasiões em alguns ecrãs e também no cálculo da apresentação das notificações ao utilizador.

O Navteq (HERE) é um serviço utilizado para a pesquisa e tradução de coordenadas para uma morada específica. Este serviço é utilizado pela aplicação nos ecrãs em que o utilizador realiza uma pesquisa de um local, nomeadamente durante o Fluxo do "Ir Para" (Figura 46) ou através da definição dos favoritos (Figura 47). Estes serão explicados mais detalhadamente no Capítulo 6.

Por último, o serviço Transporlis permite fazer o cálculo de trajetos e fornece uma informação sobre o tempo de uma viagem e o seu custo, através de um ponto de origem e de um ponto de destino. Nas aplicações, este serviço é utilizado no planeador de viagem quando o utilizador já tiver definido os pontos previamente identificados.

A informação dos serviços descritos é agregada de modo a que a XTranPaasenger API possa responder às necessidades das aplicações móveis.

4.1. Síntese do capítulo

Neste capítulo foi apresentada a estrutura e a forma de como o sistema está organizado. As aplicações móveis estão inteiramente dependentes da XTranPassenger API e é através desta que acedem aos diversos serviços, nomeadamente o Time Estimator, a Transporlis e o Navteq.

No capítulo seguinte irão ser abordados as tecnologias e as ferramentas utilizadas no desenvolvimento.

5. Tecnologias utilizadas no desenvolvimento

No âmbito do projeto Carris, este capítulo descreve as tecnologias e as ferramentas utilizadas no seu desenvolvimento.

5.1. Ferramentas utilizadas no desenvolvimento

A aplicação móvel Carris foi desenvolvida utilizando a ferramenta Xamarin, com suporte ao Visual Studio. Estas ferramentas irão ser descritas detalhadamente nos subcapítulos seguintes.

5.1.1. Xamarin

O Xamarin é uma ferramenta utilizada no desenvolvimento de aplicações móveis com recurso à linguagem C#. O Xamarin converte o código e compila aplicações nativas, ao contrário de outras ferramentas de desenvolvimento *cross-platform* que permitem criar aplicações híbridas. Esta é a principal vantagem do Xamarin. Isto também significa que a performance é semelhante a aplicações nativas (que em Android são geralmente desenvolvidas em Java ou Kotlin; e em iOS em Objective C ou Swift), mas que utiliza uma única linguagem para ambas as plataformas. Esta abordagem traz vantagens ao facilitar a escrita de módulos de software partilháveis, não só entre Android e iOS, mas também com módulos integrados com o *backend*. Nas figuras seguintes estão representadas as arquiteturas Xamarin.Android (Figura 16 [58]) e do Xamarin.iOS (Figura 17 [59]).

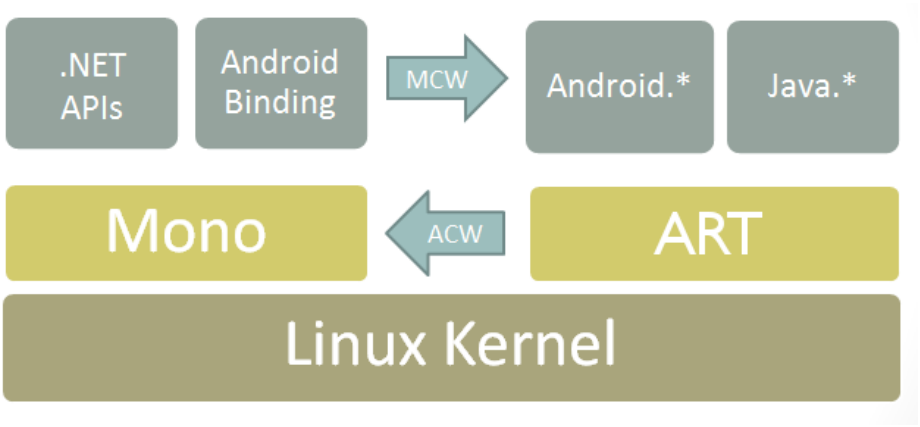


Figura 16 - Arquitetura do Xamarin.Android

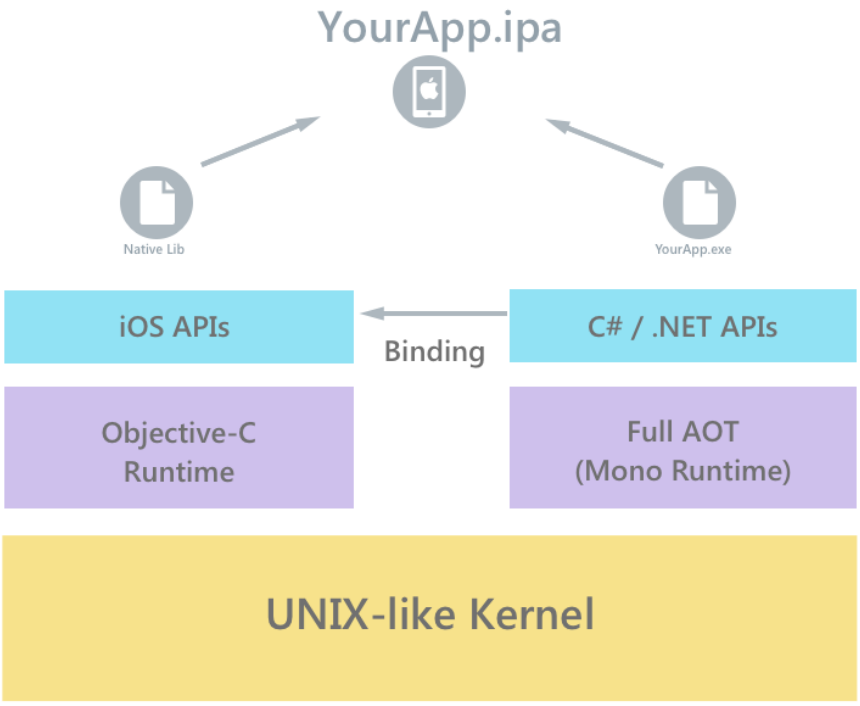


Figura 17 - Arquitetura do Xamarin.iOS

As aplicações Xamarin.Android são executadas no ambiente Mono[9] [60] e utilizam o tipo de compilação *Just in Time* (JIT). Este tipo de compilação é executado lado a lado com a máquina virtual Android Runtime (ART) e ambos são executados no *kernel* (núcleo) do Linux e têm a particularidade de disponibilizar várias API's, permitindo que os programadores possam aceder ao sistema subjacente.

[9] Mono – ambiente de execução que permite converter código da linguagem C# para linguagem nativa.

No Android a maioria da estrutura do sistema, como o Áudio, os Gráficos, o Open GL não estão diretamente disponíveis nas aplicações nativas. Estes são disponibilizados, somente através das Java API's no *runtime* do Java (nos *namespaces* Java* ou nos *namespaces* Android*) [58].

No Xamarin.Android (Figura 16) é possível aceder aos vários recursos do sistema através da utilização das .NET API's ou utilizando as classes disponíveis nos *namespaces* Android através de uma *bridge* para as Java API's [58]. Este tipo de aplicações contêm os *Android Callable Wrappers* (ACW) e os *Managed Callable Wrappers* (MCW). Os ACW possibilitam ao Android fazer a chamada ao código da aplicação (*managed code* [61]) enquanto que os MCW são utilizados sempre que o código da aplicação utilize código Android.

As aplicações Xamarin.iOS (Figura 17), tal como as aplicações referidas anteriormente, são executadas no ambiente Mono. Utilizam o tipo de compilação *Ahead of Time* (AOT) para compilar código C# para a linguagem ARM assembly e são executadas com o *runtime* do Objective-C. Estes ambientes são executados num *kernel* do tipo UNIX e como tal, são disponibilizadas várias API's aos programadores, permitindo assim, aceder ao sistema nativo da plataforma [59].

Compilação JIT versus Compilação AOT

Quando uma aplicação da plataforma Xamarin é compilada (Figura 18) o Mono será executado e converte o código C# numa *Intermediate Language* (IL). Se a aplicação que estiver a ser executada for do tipo Xamarin.Android ou do tipo Xamarin.iOS (neste caso, apenas através do simulador), é feita uma compilação da IL utilizando o compilador JIT.

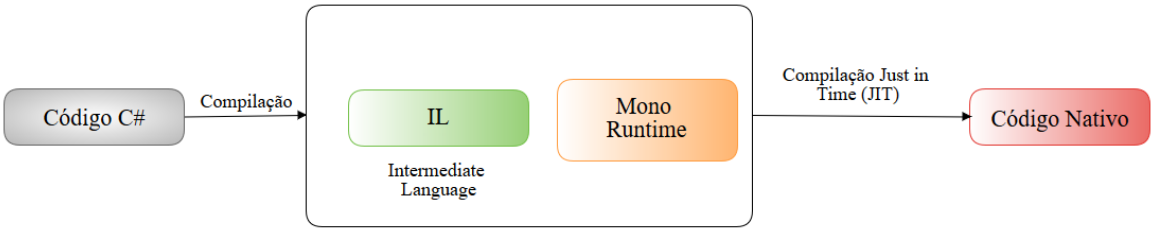


Figura 18 - Compilação Just In Time

O processo de compilação AOT está representado na Figura 19. No entanto, há uma restrição de segurança no iOS, definida pela Apple, que não permite a execução de código

gerado dinamicamente num dispositivo. Para garantir a conformidade com esses protocolos de segurança, o Xamarin.iOS utiliza um compilador AOT para compilar o código gerado. Isso produz um binário nativo do iOS, que, opcionalmente poderá ser otimizado com a *Low Level Virtual Machine* (LLVM)[10] para dispositivos móveis, podendo assim ser inserido no processador ARM da Apple [59].

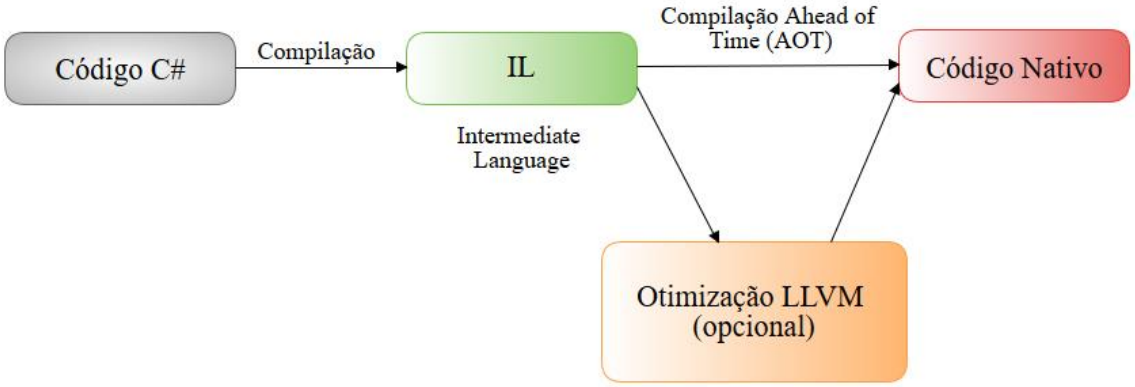


Figura 19 - Compilação Ahead of Time

5.1.2. Visual Studio

O desenvolvimento dos projetos é realizado com recurso ao IDE Visual Studio 2017 que é um sistema proprietário da Microsoft. Existem versões disponíveis para os sistemas operativos macOS e Windows mas, no nosso caso foi utilizada a versão Windows, uma vez que é o sistema operativo utilizado nos computadores da empresa. Com esta ferramenta existe a possibilidade de [62]:

- Criar aplicações .NET Core que são executados no Windows, macOS e Linux.
- Criar aplicações móveis para iOS, Android e Windows em C# e F# utilizando o Xamarin.
- Utilizar tecnologias da Web padrão - HTML, CSS e JavaScript - para criar aplicações móveis para iOS, Android e Windows usando o Apache Cordova.
- Criar jogos 2D e 3D em C # utilizando o Visual Studio Tools for Unity.

[10] Biblioteca que é utilizada para construir, otimizar e produzir código intermediário e/ou binário

- Criar aplicações C++ nativas para dispositivos iOS, Android e Windows compartilhando código comum através de bibliotecas criadas para iOS, Android e Windows.
- Implementar, testar e fazer *debug* em aplicações com o emulador do Android.

5.2. Ferramentas utilizadas no design

Durante o desenvolvimento das aplicações móveis houve bastantes dificuldades em construir o layout através do Visual Studio, uma vez que o suporte ao design desta ferramenta encontrava-se num estado bastante limitado de utilização. Assim, por exemplo, durante o desenvolvimento das aplicações Android surgiram casos em que o editor do XML falhava a abertura do layout preview[11]. Para mitigar estes problemas foram utilizadas as duas ferramentas oficiais de desenvolvimento de aplicações móveis nativas Android e iOS – o Android Studio e o XCode, respetivamente. Estas tiveram como principal função a criação dos layouts e, uma vez concluídos, eram transferidos para os projetos Xamarin.Android e Xamarin.iOS.

5.2.1. Android Studio

O Android Studio é conhecido como o IDE oficial de desenvolvimento nativo de aplicações Android, substituindo o Eclipse. Surgiu em 2013 e está disponível, gratuitamente, para os sistemas operativos Windows, macOS, Linux. Possui alguns recursos relevantes, tais como [63]:

- Um sistema de compilação baseado no *Gradle*;
- Um emulador com inúmeros recursos;
- Um ambiente de desenvolvimento para todos os dispositivos Android;
- *Instant Run* para realizar alterações em modo de execução sem haver necessidade de uma nova compilação.
- Integração com o GitHub
- Ferramentas e estruturas de testes
- Compatibilidade com C++ e NDK

[11] Ferramenta que permite saber como é que o layout será renderizado no dispositivo do utilizador

- Entre outros...

5.2.2. XCode

O XCode é um IDE gratuito proprietário da Apple para o macOS utilizado para o desenvolvimento de software para o próprio sistema operativo, para o iOS, o watchOS, e o tvOS. Foi lançado em 2003, e a sua instalação é possível ser feita, através da loja oficial da Apple (Mac App Store). Algumas das suas principais características são [13]:

- *Debugger* gráfico
- Framework XCTest para realizar testes unitários
- Compatibilidade com a Integração contínua
- Capturas OpenGL
- Editor de Versões – suporte com o Git
- Interface Builder
- Entre outros...

5.3. Síntese do capítulo

Neste capítulo foram apresentadas as tecnologias e as ferramentas utilizadas no desenvolvimento - Xamarin e Visual Studio - e também as ferramentas utilizadas no design – Android Studio e XCode. Foi explicado, com mais detalhe, o funcionamento do Xamarin e seus modos de compilação e foram expostas as principais características e funcionalidades das ferramentas de design e do Visual Studio.

No capítulo seguinte irá ser abordado o trabalho realizado durante o estágio, nomeadamente a aplicação Carris e XTranPassenger API.

6. Trabalho realizado

Este capítulo descreve o desenvolvimento da aplicação móvel Carris. Para além disso irá ser explicado como se procedeu à implementação da XTranPassenger API e de que modo esta está relacionada com as aplicações móveis. Serão também apresentadas as decisões tomadas no desenvolvimento destes projetos.

6.1. XTranPassenger API

A XTranPassenger API é um *webservice RESTful*, que tem como principal objetivo servir de suporte às aplicações móveis Carris, tendo capacidade para suportar outras aplicações cliente, como sites ou outras aplicações móveis. A API integra e gere vários serviços e fontes de dados (Transpolis [64], Navteq [65], outros produtos da Tecmic), agrupando a informação de modo a que esta possa responder às necessidades das aplicações cliente.

A API é um serviço ASP.NET Core desenvolvido em C#, alojada em máquinas com o sistema operativo Windows Server. A principal estrutura de suporte da API é uma base de dados implementada em SQL Server. Esta base de dados (TinyPassenger BD) é utilizada como repositório de dados em conjunto com o sistema Carris, que suporta o sistema geral. Todos os métodos relativos aos pedidos da API são do tipo GET e devolvem dados com o formato JavaScript Object Notation (JSON) [66]. Os pedidos permitem aceder por Paragens, Linhas, Planeador de viagem, Estimativas e *Geocoding*. A documentação da API está disponível no Anexo B - Documentação da XTranPassenger API.

6.1.1. Modelo de domínio

Tal como dito anteriormente no Capítulo 4, a API utiliza vários serviços independentes entre si – o Time Estimator Service, o Navteq, a Transporlis e a TinyPassenger BD. Apesar da API ter o seu próprio modelo de domínio (correspondendo ao seu conjunto de serviços) no âmbito deste relatório, por questões de simplicidade e de compreensão do sistema em questão, optou-se por apresentá-lo pelos serviços acima referidos.

6.1.1.1. Serviço TinyPassenger BD

Seguidamente serão apresentados os objetos que compõem o serviço TinyPassenger BD. Na Figura 20 estão representados os principais objetos deste serviço.

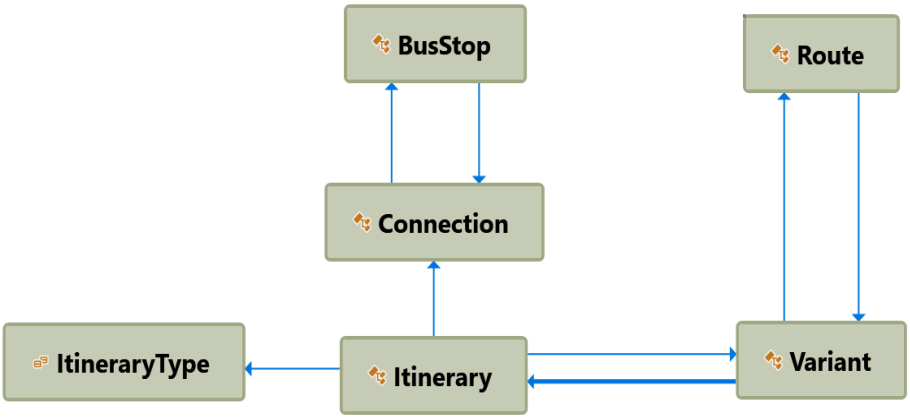


Figura 20 - Modelo de domínio TinyPassenger BD

Route

Uma *route* é uma entidade que representa uma linha. Cada *route* caracteriza-se pelo número de linha (identificador público), pelo seu nome, se é uma linha circular ou não, se é visível ao público e pelo seu *timestamp*. Para além destes atributos está ainda associada a uma lista de variantes **(*Variant*)**.

Variant

Variant é uma entidade que representa uma variante. Uma variante significa que a linha poderá sofrer várias alterações e assim mudar a sua forma, consoante a altura do ano. Cada variante é caracterizada por uma linha, pelo número da variante, pela data da criação e de expiração e também pelo seu itinerário (***Itinerary***).

Itinerary e ItineraryType

Itinerary é uma entidade que representa um itinerário. Este possui um objeto do tipo **ItineraryType**, uma variante e também uma *shape*. Nas aplicações móveis, uma *shape* é utlizada para definir a forma da linha. Para além destes atributos, possui um *timestamp* e uma lista de troços (***Connections)***.

ItineraryType é uma enumeração que define o tipo do itinerário através dos atributos: ascendente, descendente ou circular.

Connection

Connection é uma entidade que representa um troço de um itinerário, caracterizando-se pela distância e pela paragem **(BusStop).**

BusStop

Uma *BusStop* é uma entidade que representa uma paragem. Uma paragem possui um número público, um nome, um nome curto, uma latitude, uma longitude, um *timestamp*, se é visível ao público, uma data de eliminação e uma lista de troços.

6.1.1.2. Serviço Time Estimator Service

Seguidamente será apresentado o objeto que compõe o serviço Time Estimator Service. A Figura 21 representa o principal objeto deste serviço.

Figura 21 - Entidade que representa o Time Estimator Service

Time Estimation

Time Estimation é uma entidade que representa uma estimativa de tempo que o autocarro demora a chegar a uma paragem. Uma estimativa de tempo possui os seguintes atributos: data estimada de chegada, o número da viagem e o número da paragem. Possui também alguma informação sobre a linha, tais como o número, o nome e o destino da linha. Esta informação, em agregado com alguns dos atributos anteriores, é utilizada como fator de comparação entre estimativas.

6.1.1.3. Serviço Navteq

Seguidamente serão apresentados os objetos que compõem o serviço Navteq. Na Figura 22 estão representados os dois objetos deste serviço.

Figura 22 - Entidades que representam o serviço Navteq

Suggestion

A entidade *Suggestion* representa uma sugestão de um local. Uma *Suggestion* é representada por uma cidade e por um nome de rua/local (*header*). Para além disso, possui um identificador de localização que poderá ser utilizado pela entidade ***Result.*** Na aplicação Carris esta entidade fornece os dados que irão ser utilizados pela entidade mapeada (***ResultDto***). Permite apresentar várias sugestões de pesquisa ao utilizador (Figura 32) nas Principais funcionalidades da aplicação Carris.

Result

Result é uma entidade que representa um local, resultante duma pesquisa. Um *result* é representado por uma latitude e uma longitude, uma morada e um *header*. Na aplicação Carris esta entidade é utilizada para apresentar um resultado de pesquisa ao utilizador, através do

identificador de localização fornecido na entidade *Suggestion* ou através da escolha de um ponto no mapa.

6.1.1.4. Serviço Transporlis

Seguidamente serão apresentados os objetos que compõem o serviço Transporlis. Na Figura 23 estão representados os objetos deste serviço.

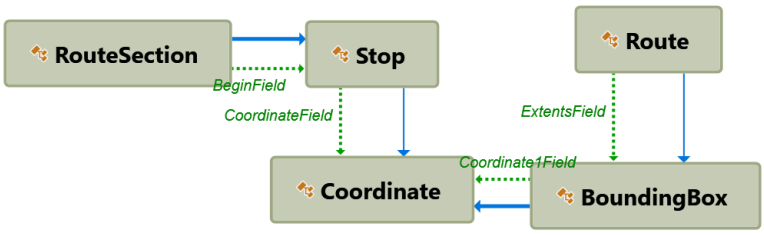


Figura 23 - Modelo de domínio Transporlis

Route

No serviço Transporlis, *Route* é uma entidade que representa uma solução do cálculo de um percurso. Uma *route* é caracterizada por uma data de início e de fim, uma lista de passos, **RouteSection**, e um objeto do tipo **BoundingBox**.

RouteSection

RouteSection é uma entidade que representa um passo (*step*) do percurso. Uma *RouteSection* é identificada por duas entidades **Stop**, que indicam os pontos de início e do fim do percurso. Para além disso, possui o tipo de transporte, a distância, o preço, o tempo de espera, o tempo de viagem, uma descrição e uma lista de pontos geográficos (**Coordinate)**.

Stop

A entidade *Stop* representa o ponto inicial ou final de um percurso e representa-se por um nome, um ponto geográfico e por uma data.

BoundingBox

A entidade *BoundingBox* representa os limites do percurso calculado. Os únicos atributos desta entidade são dois pontos geográficos, o inicial e o final.

Coordinate

Coordinate é uma entidade que representa um ponto geográfico e é representada através de uma latitude e uma longitude.

6.2. Aplicação Carris

A aplicação Carris surge com o principal objetivo colocar em serviço e manutenção, uma solução digital de integração da informação sobre os serviços da Carris, com camadas de apresentação para diferentes ambientes. O objetivo da Carris foi desenvolver novos canais de informação e comunicação com os seus utilizadores e assim a aplicação Carris aparece como um meio de informar os utilizadores do "antes e durante" de cada viagem.

A aplicação foi desenvolvida com recurso ao Xamarin, para Android e iOS, através do Visual Studio e com a mesma linguagem que os restantes produtos da Tecmic (C#), o que facilita a integração e o reaproveitamento de código de bibliotecas partilhadas (*dynamic link libraries* - DLLs). As aplicações possuem a sua própria BD local, de modo a fazer *cache* dos recursos (Favoritos, estimativa de tempo a seguir, etc.). Esta BD é implementada com recurso a SQLite [67], sendo uma das ferramentas de bases de dados mais utilizadas em desenvolvimento móvel [68]. Esporadicamente, os editores oficiais (Android Studio e XCode) são utilizados como ferramentas de suporte. A título de exemplo, o Android Studio é por vezes utilizado para visualização de *logs* (para efeitos comparativos entre aplicações) e para a criação dos *layouts*. O XCode é maioritariamente utilizado para gerir os certificados e também para a criação de *layouts* da aplicação.

6.2.1. Modelo de domínio

O modelo de domínio da aplicação Carris é apresentado como um conjunto de objetos *Data Transfer Objects* (DTOs). Estes objetos são utilizados para movimentar os dados entre camadas diferentes: denominam-se, assim objetos de transferência e são utlizados para transmitir dados, não contendo nenhuma lógica de negócio.

Estes objetos foram criados a partir da XTranPassenger API, onde foram realizados os mapeamentos dos seus objetos de domínio. Para realizar este processo foi utilizada a biblioteca AutoMapper [69] através do NuGet[12] [70]. Este tipo de mapeamento (objeto para objeto) funciona através da transformação de um objeto de entrada num objeto de saída de um tipo diferente. Através da Figura 24 e Figura 25, respetivamente, é possível ver um exemplo deste método e a classe mapeada.

```
public TinyPassengerMapperProfile()
{
  CreateMap<Route, RouteDto>()
        .ForMember(dto => dto.ID, opt => opt.MapFrom(obj => obj.ID))
        .ForMember(dto => dto.Timestamp, opt => opt.MapFrom(obj => obj.Timestamp))
        .ForMember(dto => dto.RouteNumber, opt => opt.MapFrom(obj => obj.RouteNumber))
        .ForMember(dto => dto.Name, opt => opt.MapFrom(obj => obj.Name.ToTitleCase()))
        .ForMember(dto => dto.IsPublicVisible, opt => opt.MapFrom(routeDto =>
        routeDto.IsPublicVisible));
}
```

Figura 24 - Mapeamento de Route para RouteDto

```
public class RouteDto
{
        public int ID { get; set; }

        public string RouteNumber { get; set; }

        public string Name { get; set; }

        public bool IsPublicVisible { get; set; }

        public DateTime Timestamp { get; set; }

}
```

Figura 25 - Classe mapeada RouteDto

[12] Ferramenta utilizada no .NET/.NET Core como gestor de dependências e bibliotecas

Foi aplicado este processo para as restantes entidades identificadas no modelo do domínio da XTranPassenger API e assim, foram criados um conjunto de DTOs utilizados pela aplicação Carris. Estas entidades mapeadas estão representadas nas tabelas seguintes incluindo as entidades precedentes.

Tabela 6 - Mapeamento TinyPassenger

Entidades a mapear	Entidades mapeadas
BusStop	**BusStopDto**
BusStop	**NearbyBusStopDto**
BusStop	**EstimationBusStopDto**
Route	**RouteDto**
Route	**FullRouteDto**
Variant	**VariantDto**
Itinerary	**ItineraryDto**
Connection	**ConnectionDto**

Tabela 7 - Mapeamento Time Estimator

Entidade a mapear	Entidade mapeada
TimeEstimator	**EstimationDto**

Tabela 8 - Mapeamento Navteq

Entidades a mapear	Entidades mapeadas
Result	**GeoResultDto**
Suggestion	**GeoSuggestionDto**

Tabela 9 - Mapeamento Transporlis

Entidades a mapear	Entidades mapeadas
Coordinate	**CoordinateDto**
BoundingBox	**BoundingBoxDto**
Route	**SolutionDto**
RouteSection	**SolutionStepDto**
Stop	**StepPoinDto**

Pela análise das tabelas anteriores pode-se concluir que foram realizados vários mapeamentos da mesma entidade, mas de forma a dar origem a entidades diferentes. Um exemplo deste caso é a entidade ***BusStop,*** que por sua vez deu origem às entidades ***BusStopDto, NearbyBusStopDto*** **e** ***EstimationBusStopDto.*** Apesar de derivarem da mesma entidade, estas têm funções diferentes. No Anexo C - Classes do domínio com os seus atributos estão representadas estas entidades com os seus respetivos atributos.

BusStopDto

A entidade *BusStopDto* representa uma paragem do sistema da Carris.

NearbyBusStopDto

A entidade *NearbyBusStopDto* representa o mesmo que a entidade *BusStopDto* com a particularidade de retornar uma distância. Representa assim, uma paragem que esteja próxima de uma localização em particular.

EstimationBusStopDto

A entidade ***EstimationBusStopDto*** representa uma paragem com uma lista de estimativas.

Os restantes DTOs expostos nas tabelas anteriores apresentam as mesmas funções que as entidades representadas no capítulo 6.1.1, relativo ao modelo de domínio da XTranPassenger API.

Na Figura 26 está identificado o modelo de domínio completo, com os diversos DTOs e as suas ligações.

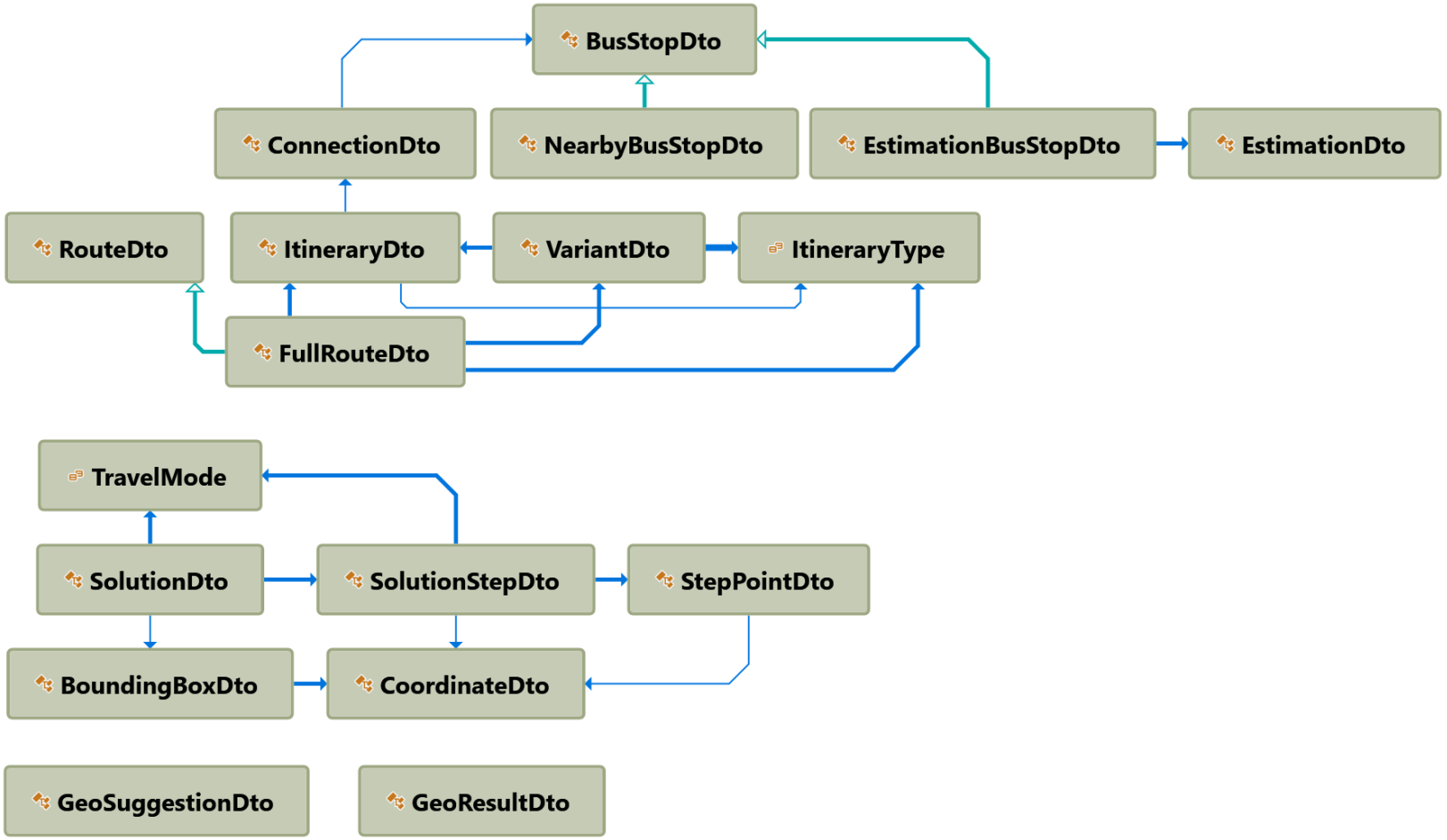


Figura 26 - Modelo de domínio (DTOs) da aplicação Carris

6.2.2. Arquitetura da aplicação

Este subcapítulo tem como objetivo identificar e explicar os vários componentes e módulos em que a aplicação Carris se insere. Tanto a aplicação Android como a de iOS possuem um módulo partilhado (*Shared*) – *Shared Android* e *Shared iOS*, respetivamente. Além do mais, as aplicações utilizam outros módulos partilhados incluindo, o *Shared Utils*, o *Business Logic*, os API *Objects*, a *Local DB* e a *Cache*. Estes componentes estão representados na Figura 27.

No seguimento da Figura 27 irá ser explicado cada um destes módulos mais detalhadamente e a forma como eles se relacionam.

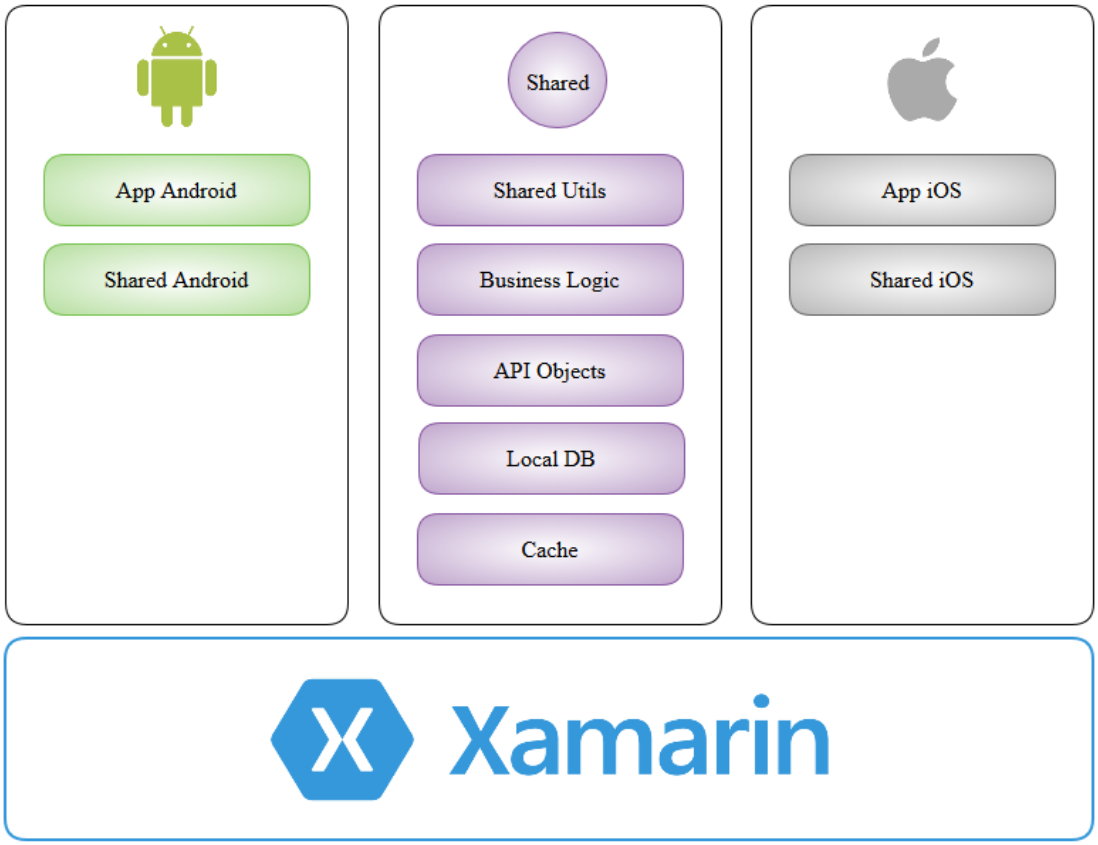


Figura 27 - Arquitetura da aplicação Carris

App Android e iOS

Para cada plataforma existe a sua própria aplicação, cada uma com as suas interfaces e componentes específicos. Apesar da lógica ser partilhada, por exemplo: o que acontece quando se clica num botão, são necessários mecanismos específicos para desenhar esse botão nas diferentes plataformas.

De igual modo, estes projetos possuem chaves próprias e configurações específicas para as respetivas *stores* (Google Play para a aplicação Android e a Apple Store para a aplicação iOS).

Shared Android e iOS

Estes projetos são repositórios de utilitários e ferramentas específicas (de Android ou iOS), mas que podem ser reaproveitados por outras aplicações Xamarin, por exemplo: utilitários para lidar com *Activities* (janelas do Android) e utilitários para lidar com *UiViewControllers* (janelas do iOS) fazem parte destes projetos.

Shared Utils

Este módulo é o modo mais genérico de utilitários. É partilhado ao longo de todos os módulos da aplicação móvel, mas não só, também é incluído no código da XTranPassenger API e no código de outros projetos da Tecmic.

Este módulo incorpora *extensions* em objetos de sistema e lida com tarefas independentes de plataforma ou contexto, como ler dados JSON (que, por exemplo, a XTranPassenger API envia e os clientes recebem).

Business Logic

Este módulo é composto pela lógica das aplicações móveis que é independente de plataformas, por exemplo: a lógica relacionada com integração da Transporlis e do Geocoding não depende das plataformas móveis Android ou iOS.

Este módulo também permite unificar diferenças entre as plataformas, ao trabalhar comportamentos na forma de interfaces. Isto é uma forma do padrão de desenho **Bridge**

Pattern em que se trabalha apenas com interfaces e as implementações específicas são injetadas dinamicamente.

API Objects

Neste módulo estão incluídos os objetos que a XTranPassenger API conhece e devolve, na forma de JSON, mas que são utilizáveis nas aplicações cliente. Deste modo, caso um determinado método da XTranPassenger API devolva novos campos, as aplicações móveis ficam a conhecer estas alterações.

Local DB

Este módulo é responsável por criar e gerir bases de dados SQLite. Este modo é partilhado entre ambas as aplicações, uma vez que ambos usam SQLite como suporte primário de armazenamento local.

Cache

Este módulo é responsável por gerir, em conjunto com o módulo da Local DB, os principais objetos do sistema: **paragens, linhas, viagens gravadas, locais favoritos e estimativas**. Estão disponíveis diferentes operações sobre a *cache*, dependendo do tipo de objeto em questão:

- **Linhas e paragens**: se existirem linhas/paragens novas ou modificadas no servidor, desde a última vez que a aplicação foi executada, é necessário atualizar;
- **Locais favoritos e viagens offline**: gestão feita localmente no dispositivo utilizando *Shared Preferences* na aplicação Android e *User Defaults* na aplicação iOS;
- **Estimativas**: mantêm os dados das estimativas para as paragens necessárias durante um período curto de tempo, descartando-as passado esse período. Este comportamento impede *refresh´s* com períodos demasiado curtos, mas não interfere nos tempos de atualização. Ainda neste tópico, é a cache que gere as **notificações** para as subscrições do utilizador.

6.2.3. Principais funcionalidades

Este capítulo descreve as principais funcionalidades da aplicação Carris. A aplicação é destinada aos utentes dos transportes da rede Carris. Com esta visão em mente, o objetivo foi tornar a interface da aplicação o mais simples possível, facilitando assim a utilização por pessoas de diversos contextos socioeconómicos e idades. Dada a diversidade de dispositivos dentro dos sistemas Android e iOS decidiu-se, com base no estudo de distribuição de versões Android e iOS no mercado (subcapítulo 2.1), em suportar dispositivos Android com versão igual ou superior à API 17 (Android Jelly Bean) e iOS com versão igual ou superior a 8.0. A aplicação dá suporte a duas linguagens, Português e Inglês, adaptando-se consoante a linguagem definida no dispositivo.

Ecrã inicial

Este ecrã é o ponto de entrada na aplicação, sendo possível realizar a maioria das opções a partir deste ecrã. De modo a facilitar a interpretação da informação, por parte do utilizador, agrupou-se a informação em quatro *tabs* –Home, Paragens, Linhas e Informação institucional, de acordo com a sua finalidade. Para fazer a organização das *tabs* na aplicação Android, foi utilizado o componente *TabLayout* e a disposição das *tabs* foram no topo do ecrã. Na aplicação iOS aplicou-se o componente *TabBarController*. e, em contraste, com a aplicação Android, as *tabs* foram colocadas no fundo do ecrã, com o objetivo de respeitar os padrões e as *guidelines* [71] da Apple.

Ainda neste ecrã apresenta-se um mapa global, através da implementação do Google Maps, onde estão indicadas as paragens da Carris.

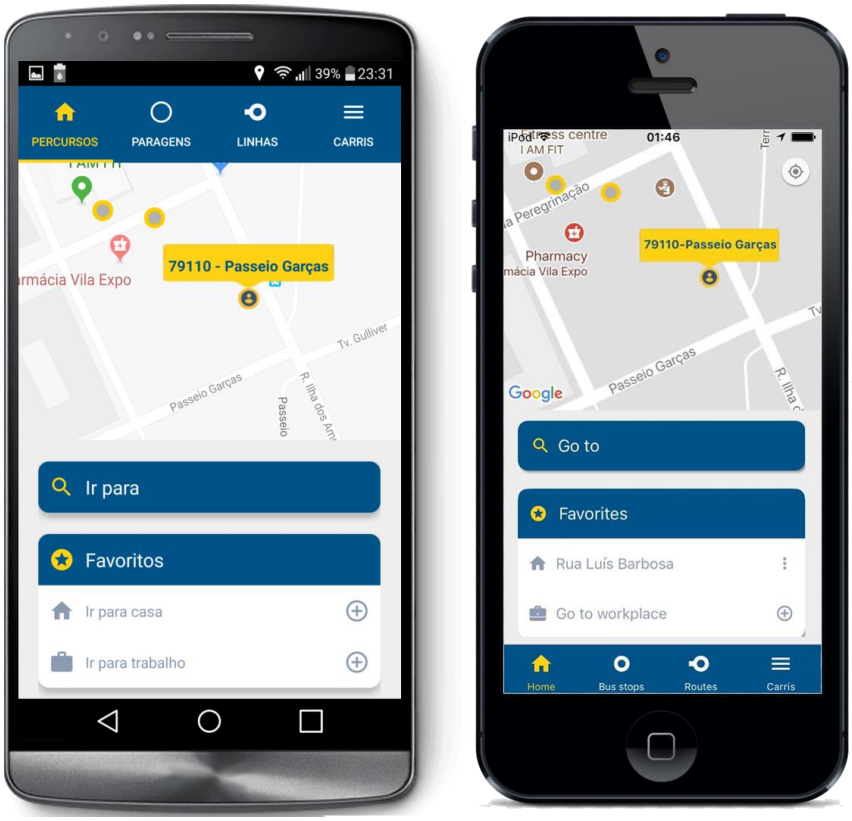


Figura 28 - Ecrãs Iniciais

Tab Home

Este ecrã (Figura 28) permite aceder ao planeador de viagens, com atalhos para planear uma viagem para os locais favoritos – Casa e Trabalho. Estes favoritos são também geridos através desses atalhos. O utilizador poderá ainda planear uma viagem definindo a localização de origem e o destino à sua escolha.

As viagens gravadas no planeador estão também disponíveis neste ecrã, para consulta. Em situações que o utilizador subscreve notificações para uma viagem, estas também aparecem listadas nesta *tab*.

Tab Paragens

Este ecrã (Figura 29) serve para mostrar as paragens mais relevantes para o utilizador, de acordo com o contexto, assim como as estimativas de tempo. As paragens são apresentadas de acordo com o seguinte critério:

1. Caso a localização do utilizador esteja ativa, as paragens mais próximas serão apresentadas em primeiro lugar;

2. Caso o utilizador tenha uma paragem selecionada no mapa, as paragens próximas da paragem selecionada aparecem em vez das paragens mais próximas do utilizador;

3. As paragens que o utilizador selecionou como favoritas aparecem a seguir às paragens próximas.

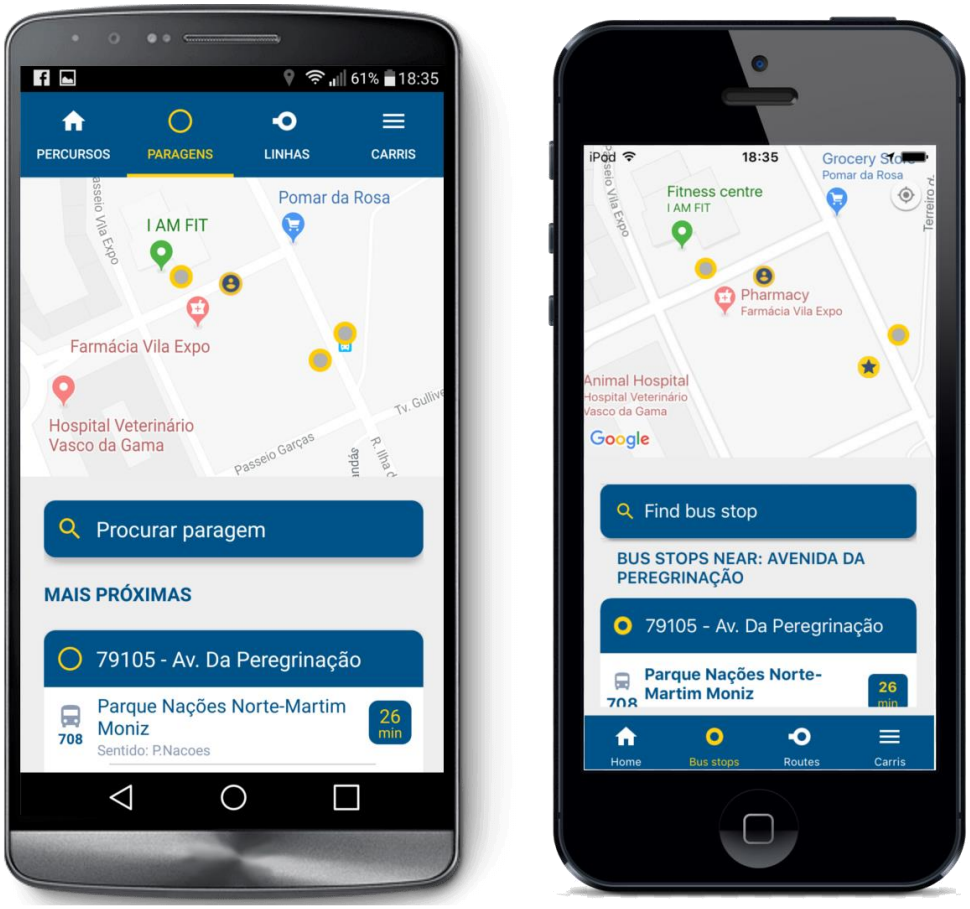


Figura 29 - Ecrãs da *Tab* Paragens

A partir das estimativas apresentadas é possível subscrever notificações que indicam quanto tempo falta para o autocarro passar na paragem. Estes alertas são apresentados quando faltarem 10, 5, 2 e 1 minutos até o autocarro chegar à paragem. Caso a paragem que o utilizador deseja não esteja a aparecer, este pode procurar no conjunto total de paragens da rede Carris, através do botão de pesquisa.

Tab Linhas

Esta *tab* (Figura 30) lista as linhas que o utilizador declarou como favoritas, de modo a rapidamente poder consultar o seu percurso e a sua lista de paragens. É também possível procurar uma linha de entre as várias linhas da rede Carris.

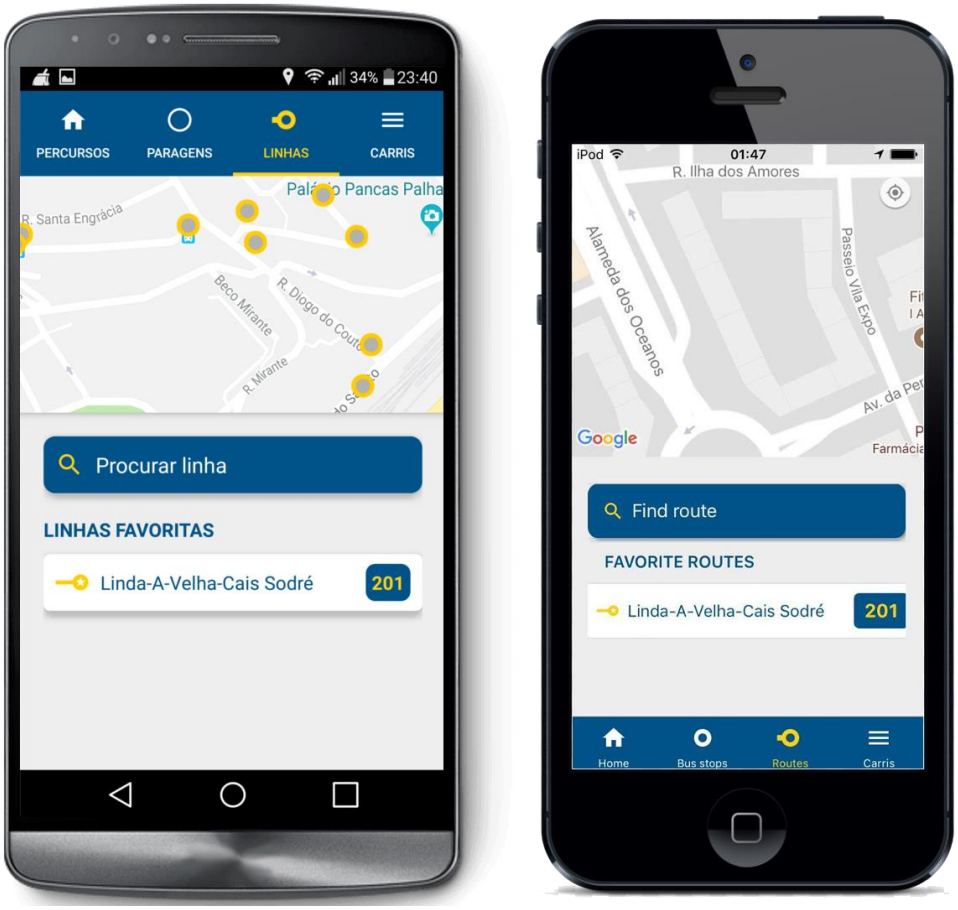


Figura 30 - Ecrãs da *Tab* Linhas

Tab Informação institucional

Esta *tab* (Figura 31) permite ver informação institucional da Carris. Nesta área o utilizador pode ainda dar feedback acerca da aplicação, via email. Este feedback é importante para permitir melhorar a aplicação com base em quem a usa diariamente.

Esta área pode evoluir para acompanhar necessidades da Carris, de modo a listar avisos e outras informações de carácter útil para o utilizador.

Figura 31 - Ecrãs da *Tab* de Informação Institucional

Planeador de viagem e pesquisa de moradas

Pelos requisitos da própria Carris, o planeador de viagem resulta da integração do sistema da Transporlis, que fornece o motor do planeador de viagem através de um *webservice*. No contexto da utilização da aplicação é realizado um cruzamento entre as linhas das redes Carris e Metro.

O utilizador pode navegar diretamente para o planeador através dos favoritos na Tab Home (origem - localização do utilizador, destino - favorito) ou pode escolher o ponto de origem e destino manualmente. Para este efeito o utilizador pode escolher no mapa ou também pesquisar por um local. Para esta pesquisa esta a ser utilizado o sistema da Navteq - Here Geocoder API. Os ecrãs da pesquisa de locais estão representados na Figura 32.

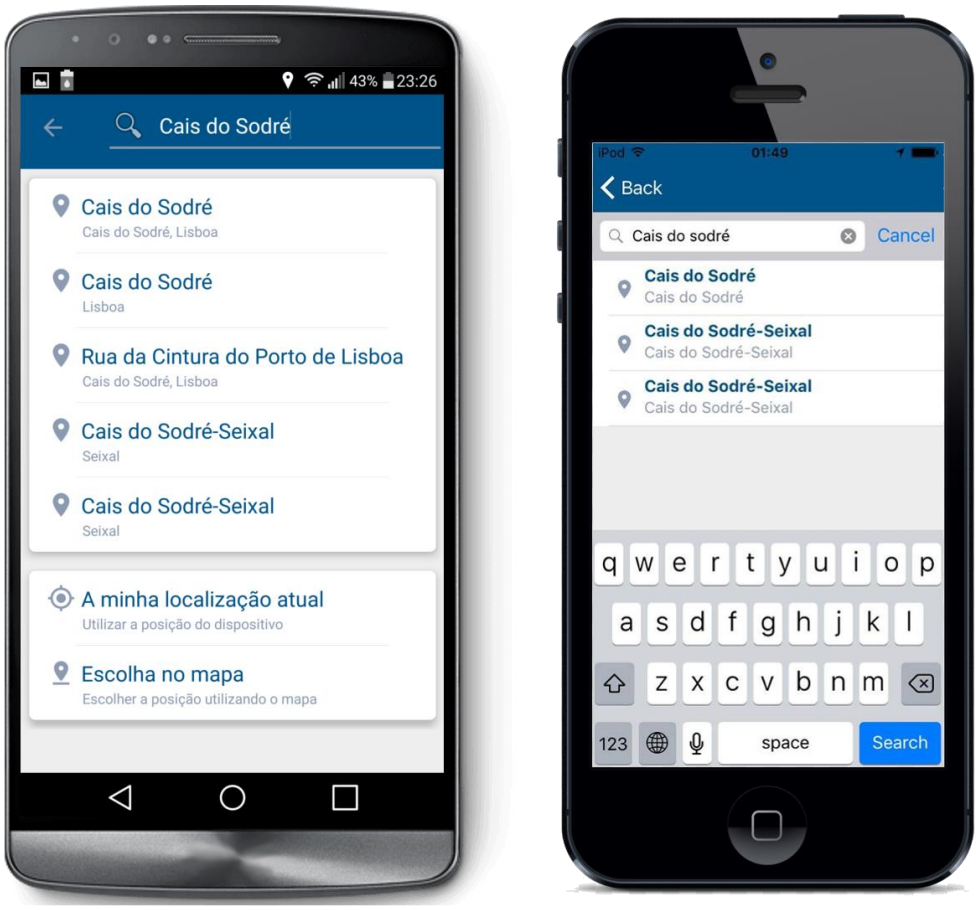


Figura 32 - Ecrãs da pesquisa de locais

Ao ser feito o planeamento de uma viagem (Figura 33) a aplicação assume a data atual como hora desejada de partida. O utilizador tem a opção de escolher uma outra data para o planeamento, por exemplo programar uma viagem no fim do dia de trabalho.

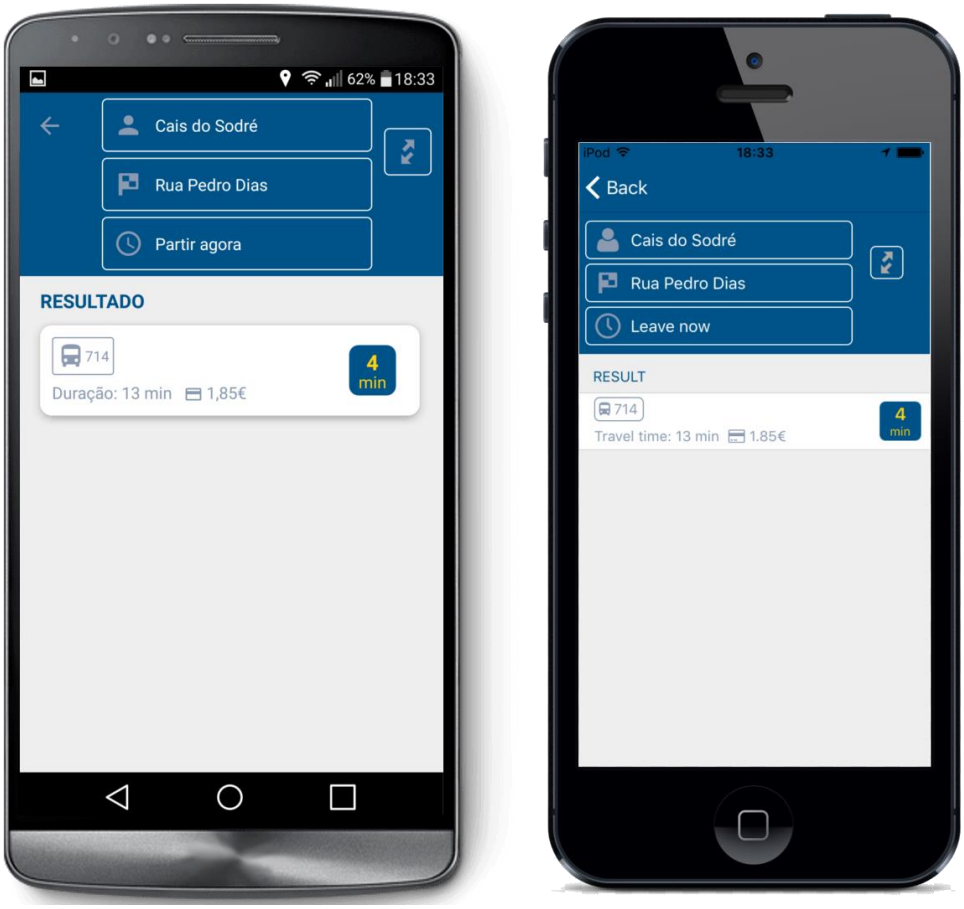


Figura 33 - Ecrãs do Planeador de viagem

Após a obtenção do plano de viagem, a aplicação lista os vários passos que o utilizador deverá seguir para essa viagem (Figura 34), incluindo informação do tipo de percurso (a pé, de autocarro ou de metro), preço, tempo de viagem, etc. O percurso é também desenhado no mapa para que o utilizador o possa acompanhar. Finalmente os percursos podem ser guardados para consulta posterior, mesmo quando a aplicação está em modo *offline*.

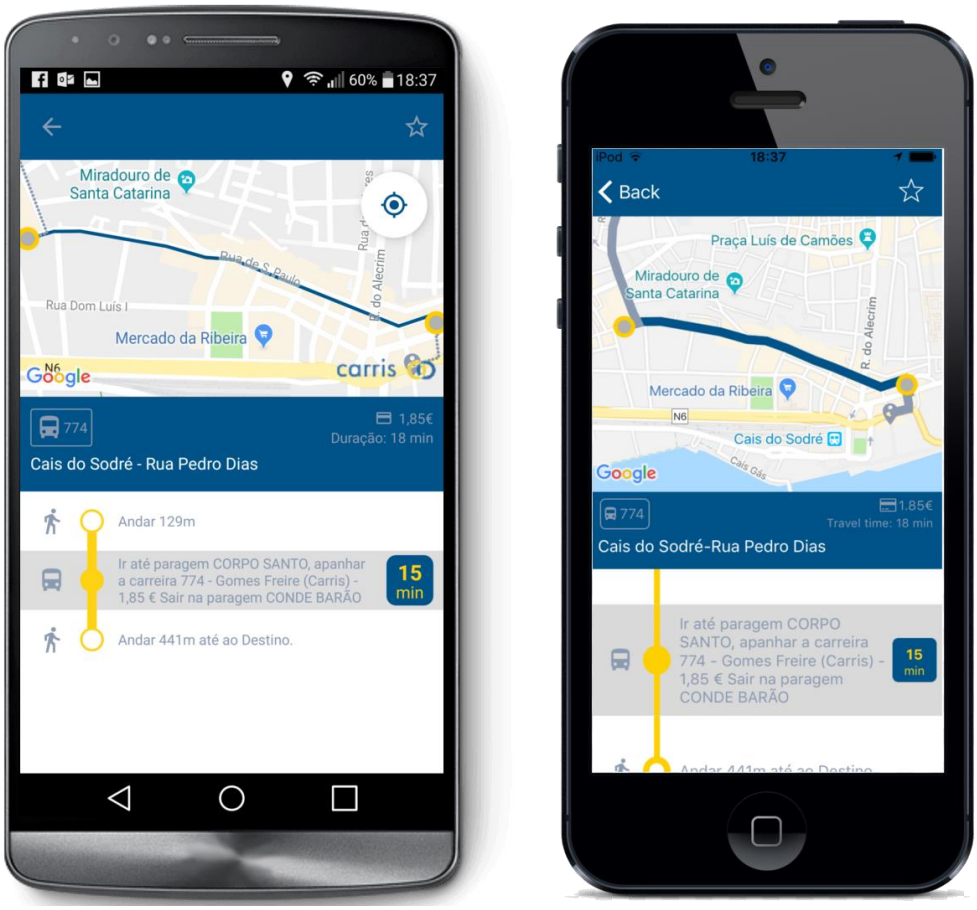


Figura 34 - Ecrãs dos detalhes de uma viagem

Estes 2 serviços - planeamento de viagem (Transporlis) e *Geocoder* - são os únicos componentes utilizados que não foram desenvolvidos pela Tecmic.

Paragens

A aplicação tenta, sempre que possível, apresentar as paragens mais relevantes para o utilizador. No entanto, caso o mesmo pretenda saber informação de outra paragem, existe a opção de pesquisar todas as paragens da rede Carris. O ecrã de pesquisa de paragens (Figura 35) lista em primeiro lugar as paragens mais próximas do utilizador, seguido das paragens favoritas e finalizando com as restantes.

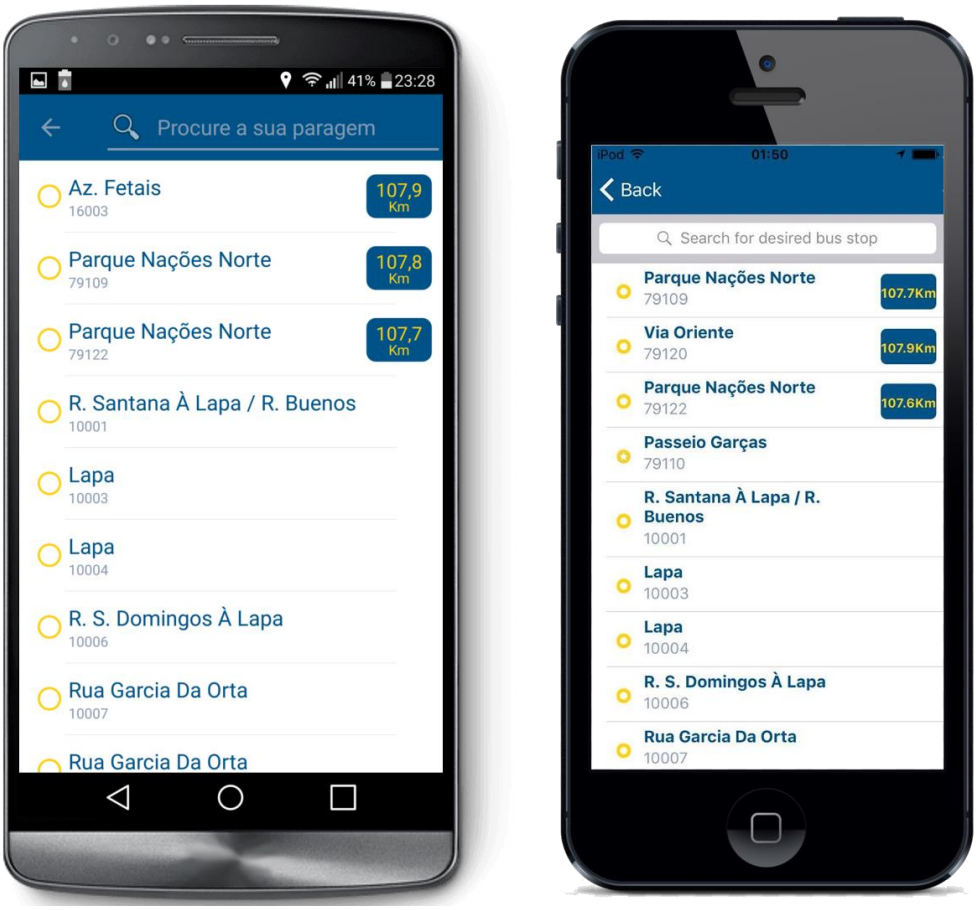


Figura 35 - Ecrãs relativos à lista de paragens

Selecionando uma paragem desta listagem, o utilizador irá para o ecrã dos detalhes da paragem. Aqui poderá ver a representação da paragem selecionada no mapa, assim como um número mais alargado de estimativas de tempo dos autocarros que estão para chegar, no máximo de 10 autocarros. Os ecrãs dos detalhes de uma paragem estão representados na Figura 36.

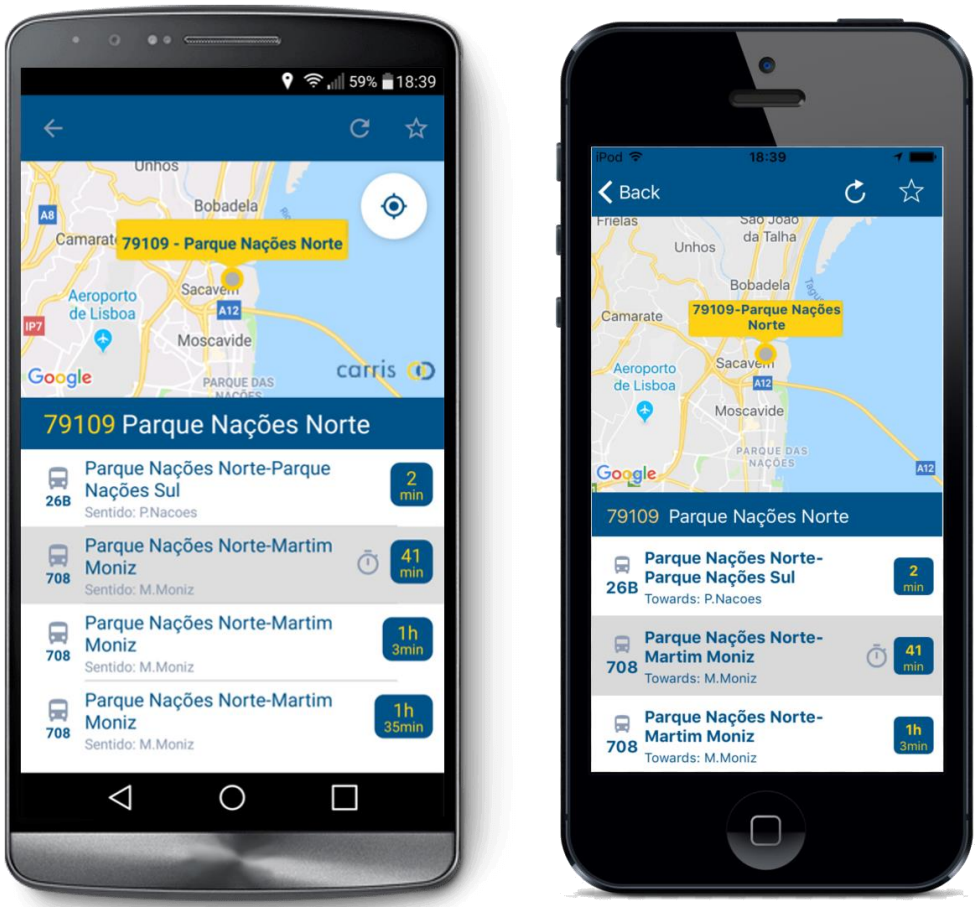


Figura 36 - Ecrãs dos detalhes de uma paragem

Neste ecrã é permitido ao utilizador subscrever estimativas (assim como na Tab Paragens), de modo a receber notificações sobre a proximidade do autocarro à paragem. Ainda neste ecrã, o utilizador poderá fazer *refresh* às estimativas de tempo ou definir a paragem como favorita.

Linhas

Estes ecrãs (Figura 37) permitem a pesquisa de linhas da rede Carris. Após selecionar uma linha é apresentada a mesma num formato "espinha", incluindo a lista de paragens e a sua representação (*shape*) no mapa. O utilizador tem ainda a opção de poder visualizar o percurso no sentido ascendente (ida) ou descendente (volta). No ecrã dos detalhes de uma linha é possível ver diretamente os detalhes de uma paragem, através da seleção do botão com a forma de um relógio, que redireciona para a lista de estimativas de tempo.

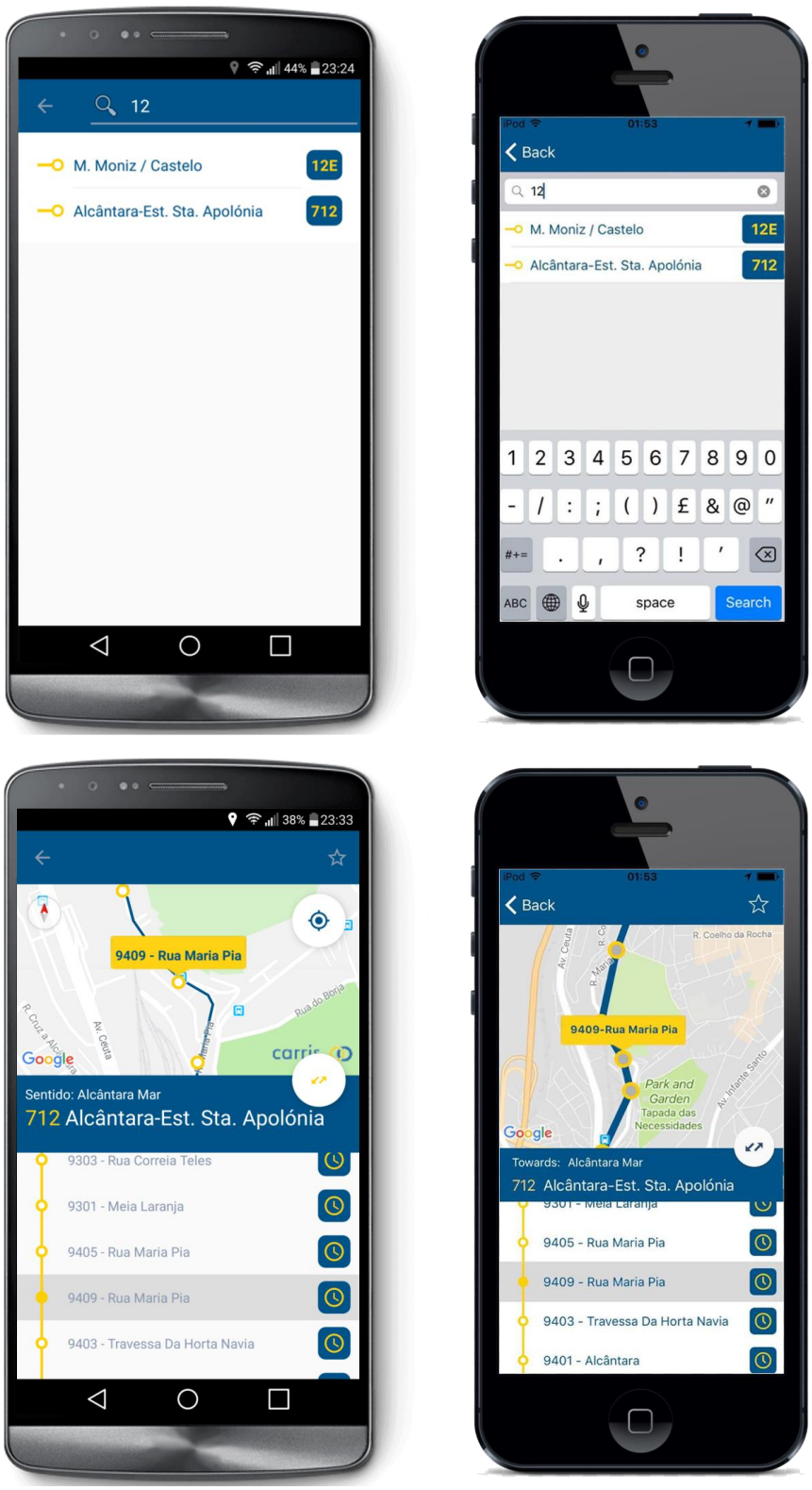


Figura 37 - Ecrãs relativos à listagem e aos detalhes de linhas

6.2.4. Implementação do projeto

Este subcapítulo tem como principal objetivo apresentar a implementação de algumas funcionalidades da Aplicação Carris, incluindo uma explicação mais detalhada sobre as mesmas e as dificuldades encontradas durante o processo.

Classes base: XtpActivity e XtpViewController

Para a aplicação Carris foram implementadas duas classes base**:** XtpActivity para a aplicação Android e XtpViewController para a aplicação iOS. Estas permitem uma simplificação substancial do código, uma vez que existiam ações e código repetido nas diversas vistas da aplicação. Estas classes implementam alguns dos métodos *callback* referentes aos ciclos de vida das *activities* [72] – OnCreate, OnResume e OnPause – e das *view controllers* [73] – ViewDidLoad, ViewWillAppear e ViewWillDisappear.

Foram criados dois métodos *virtual*[13], OnReceive (aplicação Android), ViewDidReceive (aplicação iOS) e uma propriedade *virtual*, SubscribingActions, que têm como principal objetivo fazer a gestão dos *broadcasts* que irá ser explicado mais à frente neste relatório.

As classes base permitem ainda configurar (mostrar ou esconder) os *banners*, representados nas Figura 38 e Figura 39, que indicam quando é que a aplicação está a ser executada sem ligação à Internet ou quando o telefone está sem o GPS ligado. É registado, através dos mecanismos dos *broadcasts*, quando existem mudanças de ligação destes contextos.

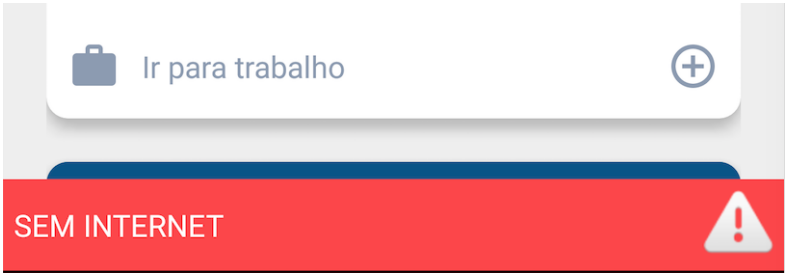


Figura 38 - Banner da app Android quando não há ligação à Internet

[13] O termo virtual é utilizado para modificar um método ou uma propriedade, e permite que sejam substituídos numa *sub-classe*

Figura 39 - Banner da app iOS quando não há ligação à Internet

Cache de pedidos

Por motivos de poupança de recursos (ligações) foi implementado um sistema de cache dupla entre clientes (aplicações móveis) e servidor (XTranPassenger API). O esquema apresentado na Figura 40 representa este processo.

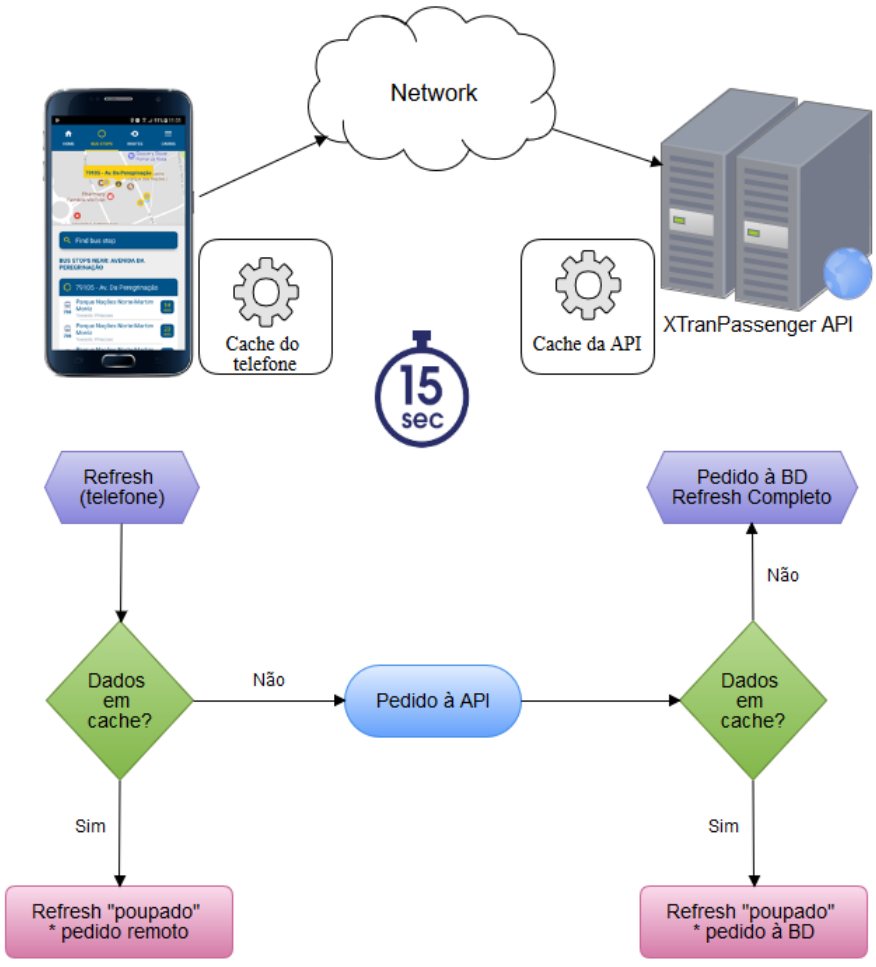


Figura 40 - Processo de Cache API/Telefone

Por exemplo, quando uma aplicação móvel necessita de fazer *refresh* às estimativas de tempo, primeiro tenta verificar se não pediu já esses dados, utilizando a *cache* do telefone. Caso possua dados com menos de 15 segundos estes são reutilizados; caso contrário irá ser feito um pedido ao servidor. Tudo isto permite poupar no número de ligações de tempo real e a bateria dos *smartphones*. Apesar de, individualmente esta diferença poder ser desconsiderada, pensando no número total de utilizadores, que o sistema poderá vir a ter, esta questão ganha uma relevância substancial.

O servidor implementa um modo de *cache* de estimativas semelhante. Isto significa que, quando o servidor recebe um pedido de estimativas, tenta reutilizar os dados em vez de estar sempre a aceder à base de dados. Esta cache na API permite reutilizar pedidos recentes de outros utilizadores para ir alimentando essa cache, uma vez que em 15 segundos podem chegar ao servidor um número considerável de pedidos de estimativas.

No limite, as aplicações podem esperar até 30 segundos para obter dados mais recentes.

Pedidos iniciais e uso de *timestamps*

Quando a aplicação é iniciada uma das suas principais tarefas é obter os dados das paragens e das linhas à XtranPassenger API, através das chamadas **/api/busstops** e **/api/routes**, respetivamente. Caso a aplicação esteja a ser executada pela primeira vez, os pedidos anteriores serão exercidos, guardando assim as paragens e as linhas, na base de dados do telefone. Durante esse processo é guardado o *timestamp,* com a data em que foram realizados os pedidos, nas preferências do utilizador[14].

Posteriormente, caso o utilizador volte a utilizar a aplicação, irá ser verificado se existe alguma alteração nas paragens/linhas guardadas na base de dados. Esta verificação é realizada através das chamadas **/api/busstops/{timestamp}** e **/api/routes/{timestamp}** utilizando o *timestamp* guardado anteriormente. No caso de existir alguma remoção ou alteração nos dados é feito um *delete* ou *update* à base de dados do telefone mantendo assim, as paragens e as linhas atualizadas. Quando o utilizador iniciar a aplicação, caso exista uma nova versão (*update*) da aplicação, o *timestamp* será removido, e serão obtidas todas as paragens e linhas novamente. Na Figura 41 estão representados os dois métodos que verificam, se é a primeira vez que a aplicação está a ser executada.

[14] *Shared preferences* no Android e *UserDefaults* no iOS

```
        private static void CheckFirstRun()
        {
            const string keyCode = "version_name";

            var savedVersioncode = UserPrefs.GetString(keyCode);
            var currentVersionCode = Platform.GetAppVersionName();

            var hasVersionCode = !savedVersioncode.IsBlank();
            if (hasVersionCode == false || savedVersioncode != currentVersionCode)
            {
                EraseOldPrefs();
                UserPrefs.SetString(keyCode, currentVersionCode);
            }

            var message = hasVersionCode ? "Found version: " + savedVersioncode :
"New install version: " + currentVersionCode;
            Logger.Debug(typeof(AppCache), message);
        }

        private static void EraseOldPrefs()
        {
            SaveDate<DbStop>(DateTime.MinValue);
            SaveDate<DbRoute>(DateTime.MinValue);
            Estimations.ClearTracked();
        }
```

Figura 41 - Verificação da primeira execução da aplicação e remoção dos timestamps

Uso de broadcasts/notifications

No final de cada ação ou de um pedido durante o uso da aplicação é comum o uso de *broadcasts* na aplicação Android e *notifications* na aplicação iOS. A implementação destes mecanismos tem como principal função realizar operações para atualizar a UI ou para a inserção de dados em listas, que possam estar a ser utilizadas por vistas. Estas vistas obrigatoriamente necessitam de estar a subscrever ("observar") para receber os *broadcasts/notifications*. Na Figura 42 e Figura 43 estão representadas as implementações, Android Broadcaster e o iOS Broadcaster, respetivamente.

No âmbito deste relatório, vai ser utilizado o termo *broadcasts* para referir de igual forma os dois mecanismos (*broadcasts* e *notifications*), uma vez que os dois têm o mesmo objetivo, apesar de implementações diferentes.

```
public class AndroidBroadcaster : IBroadcaster
{
        public const string ExtraKey = "value";

        public void Broadcast(string message)
        {
            Application.Context.SendBroadcast(new Intent(message));
        }

        public void Broadcast<TValue>(string message, TValue tValue)
        {
            if (tValue is IParcelable) { throw new Exception("Not yet implemented");}

            Application.Context.SendBroadcast(new Intent(message).PutExtra(ExtraKey,
tValue));
        }
}
```

Figura 42 - Android Broadcaster

```
public class IosBroadcaster : IBroadcaster
{
        public void Broadcast(string message)
        {
            this.Debug("Broadcasting " + message);
            DispatchQueue.MainQueue.DispatchAsync(() =>
NSNotificationCenter.DefaultCenter.PostNotificationName(message, null, null));
        }

        public void Broadcast<TValue>(string message, TValue tValue)
        {
            this.Debug("Broadcasting " + message);
            var anObject = new Extra<TValue>(tValue);
            DispatchQueue.MainQueue.DispatchAsync(() =>
NSNotificationCenter.DefaultCenter.PostNotificationName(message, anObject, null));
        }

}
```

Figura 43 - iOS Broadcaster

Tal como dito anteriormente, para gerir estes *broadcasts*, foram implementados dois novos métodos: OnReceive, ViewDidReceive e uma propriedade, SubscribingActions nas classes base. Nas sub-classes é através de um *override* da propriedade, que é possível estabelecer quais as ações disponíveis que irão ser subscritas. É através do *override* dos novos métodos, que ficam à escuta dos *broadcasts*, que irão ser executadas as ações respetivas a cada *broadcast*. As implementações estão representadas na Figura 44 e Figura 45.

```
protected virtual string[] SubscribingActions => new string[] { };

protected virtual void OnReceive(Context context, Intent intent) { }
```

Figura 44 - Método OnReceive e propriedade SubscribingActions na XtpActivity (Android)

```
protected override string[] SubscribingActions => new[]
        {
            AppCache.BusStopsAvailable,                // This is needed to start timer

            XtpLocation.LocationChangedAction,         // When user location changes
significantly, refresh nearby stops

};


protected override void OnReceive(Context context, Intent intent)
        {
            switch (intent.Action)
            {
                case AppCache.BusStopsAvailable:
                    OnBusStopsAvailable();
                    break;

                case XtpLocation.LocationChangedAction:
                    OnLocationChanged();
                    break;

              }
        }
```

Figura 45 - Implementação da propriedade SubscribingActions e do método OnReceive numa sub-classe Android

Fluxo da aplicação do “Ir Para” - planeador de viagem

Esta funcionalidade consiste em obter um percurso, dadas duas localizações, uma de origem (“De”) e outra de destino (“Para”). Para a origem, o utilizador pode selecionar a sua localização atual (caso tenha a localização ativa) ou escolher uma posição através do mapa. Além destes é possível escolher um favorito, caso os tenha definido, ou através da pesquisa de uma morada. No caso do “Para” as opções são as mesmas, excluindo a opção da localização atual. Este fluxo está a ser apresentado através da Figura 46.

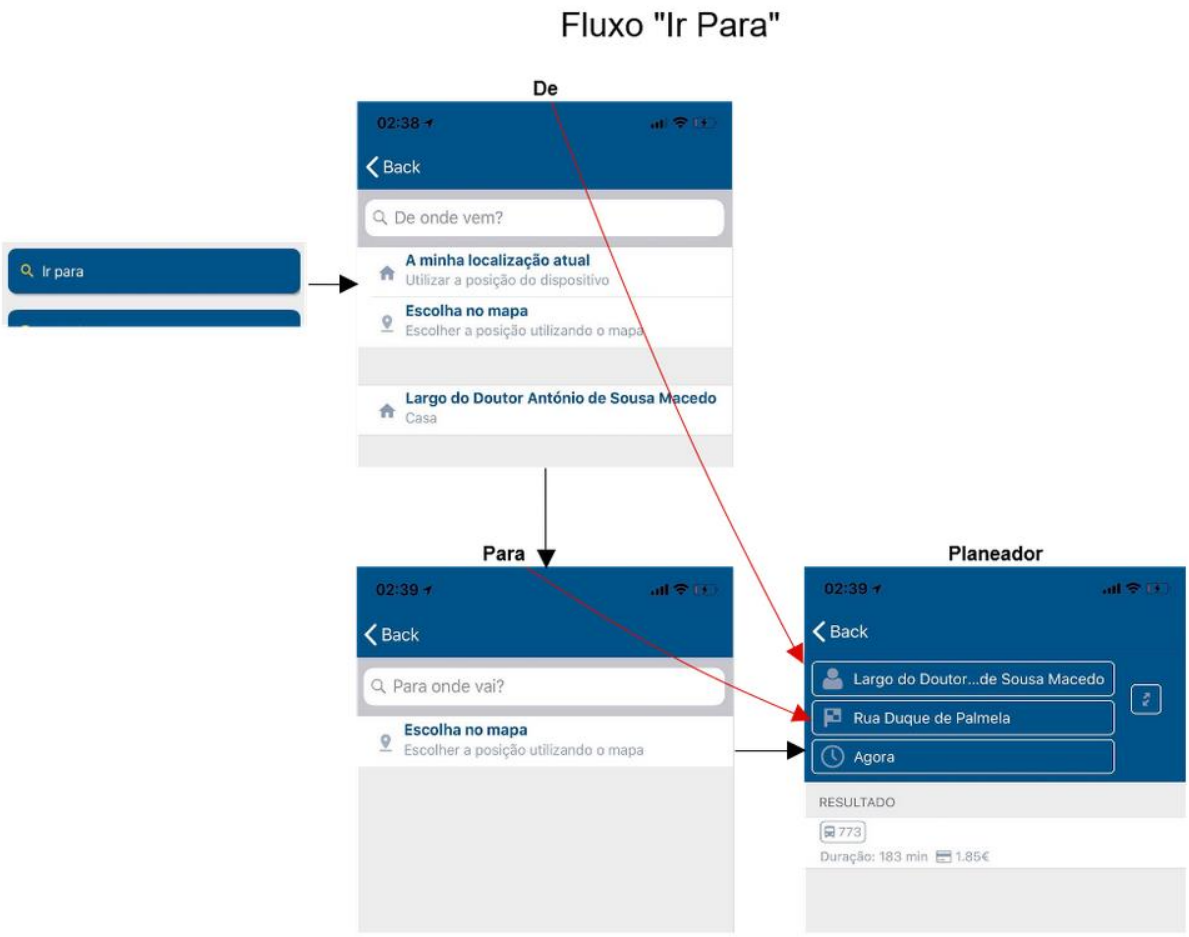


Figura 46 - Diagrama do Fluxo "Ir Para"

O utilizador é encaminhado para o planeador de viagem e é calculado um percurso, com base na localização de origem e destino definidos anteriormente. Nesta vista, o utilizador pode trocar a ordem das localizações, através do botão respetivo e ainda mudar a localização de origem ou de destino através da seleção das mesmas.

Fluxo da aplicação na definição dos favoritos – planeador de viagem

Esta funcionalidade tem como objetivo definir um local favorito – Casa ou Trabalho – através dos botões de atalho da *Tab Home*. Quando a localização do utilizador estiver ativa o "De" é automaticamente preenchido com a mesma; doutra forma o "De" terá que ser definido pelo próprio utilizador através da escolha de um local no mapa, de um outro favorito ou através da pesquisa. Com este passo realizado é necessário definir um "Para", que será o local favorito.

Com o “De” e o “Para” definidos é apresentado ao utilizador o ecrã do planeador de viagem com o melhor percurso calculado. O fluxo desta funcionalidade está representado na Figura 47.

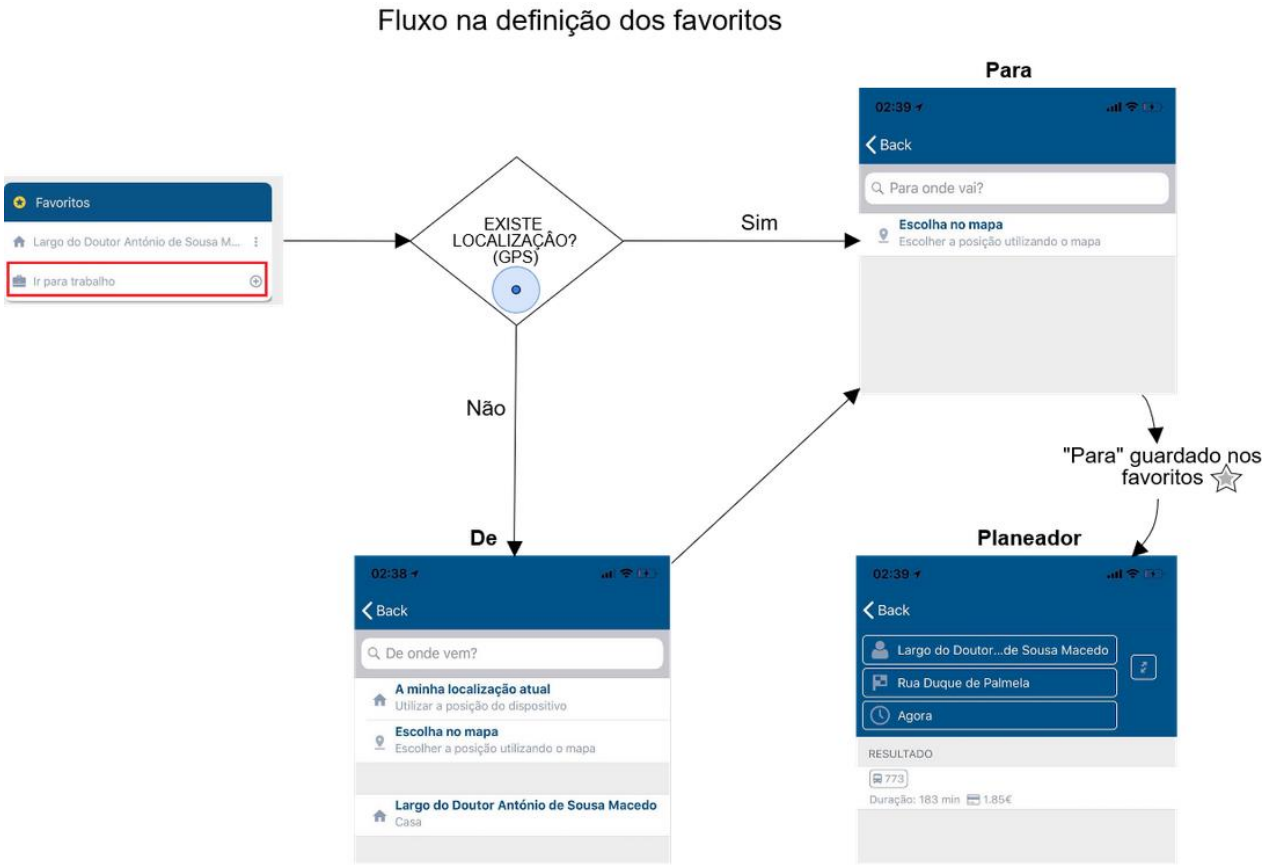


Figura 47 - Diagrama do Fluxo na definição de um favorito

Depois deste processo, quando o utilizador regressa à *Tab Home* é apresentada a localização favorita definida anteriormente, na seção dos favoritos.

Caso o utilizador selecione este favorito, no caso de ter a localização ativa será redirecionado para o planeador de viagens, com o “De” e o “Para” associados à sua localização atual e à localização favorita, respetivamente. Se o utilizador não possuir localização ativa é apresentada a vista para definir o “De”.

Para a definição dos fluxos foram necessários muitos processos de revisão de modo a tornar estas funcionalidades o mais intuitivas possível para o utilizador. Durante a implementação surgiram múltiplos problemas no controlo dos diversos estados (GPS, variáveis “De e Para”, etc.).

Paragens próximas

Esta funcionalidade tem como objetivo apresentar ao utilizador as paragens mais próximas, dependendo do como utiliza a aplicação. No caso do utilizador ter a sua localização ativa é apresentado, na *Tab Paragens*, as 3 paragens mais próximas de si, cada uma com as suas respetivas estimativas (Figura 48).

Figura 48 - Ecrã com paragens mais próximas do utilizador

Na hipótese do utilizador selecionar uma paragem no mapa ser-lhe-ão apresentadas as 3 paragens mais próximas do local, onde essa paragem se encontra. Ao nível da implementação, esta é considerada uma *"Fake Location"*, fazendo com que os pedidos às estimativas de tempo sejam sobre as paragens próximas do ponto escolhido no mapa e não sobre as paragens próximas do utilizador. Na Figura 49 é apresentado o ecrã da *Tab Paragens* com uma paragem selecionada.

Figura 49 - Ecrã com paragens próximas de um ponto do mapa

Para realizar esta implementação é verificado se existe uma localização atual e uma paragem selecionada no mapa (*Fake Location*). Perante esse resultado é feito um pedido das paragens mais próximas. No caso de estarmos perante uma *Fake Location*, é realizado um pedido extra que devolve o nome da rua onde se encontra essa paragem. Como se pode analisar pela Figura 49, este resultado é mostrado ao utilizador antes da apresentação das paragens na lista. Na Figura 50 apresenta-se um extrato de código que elucida esta funcionalidade.

```
    private void OnLocationChanged()
{
     var isFake = FakeLocation.IsFake;
     var currentLocation = isFake ? FakeLocation.Point
XtpLocation.Value?.Point;

            if (currentLocation == null)
            {
               AppCache.Estimations.ClearNearby();
               _nearbyList.Clear();
               TableView.RefreshTable();

               return;
            }

            if (isFake)
        {
           Geocoder.Reverse(FakeLocation.Point.Lat, FakeLocation.Point.Lng,
           geoResult =>
                  {
                      if (geoResult == null) { return; }
                      _geoResult = geoResult;
                  });
            }
            else
            {
                  _geoResult = null;
            }

         var url = string.Format(CultureInfo.InvariantCulture,
     Str.Url.RequestNearbyStops, currentLocation.Lat,
     currentLocation.Lng);

           new WsRequest<List<NearbyBusStopDto>>(url)
                    .OnComplete(nearby =>
                    {
                        AppCache.Estimations.SetNearby(nearby);
                        AppCache.Estimations.RestartUiTimerIfRunning();
                        TableView.RefreshTable();
                    })
                    .Get();
    }
```

Figura 50 - Método que permite a apresentação das paragens próximas

Notificações e apresentação de estimativas

Na *Tab* Paragens e nos detalhes de uma paragem são apresentadas ao utilizador estimativas de tempo dos autocarros. Na primeira vista, por omissão, irão ser apresentadas no máximo 3 estimativas por cada paragem representada: já nos detalhes de uma paragem são representadas até 10 estimativas. Estes números surgiram devido a um requisito do próprio cliente. Um exemplo desta representação está identificado na Figura 51.

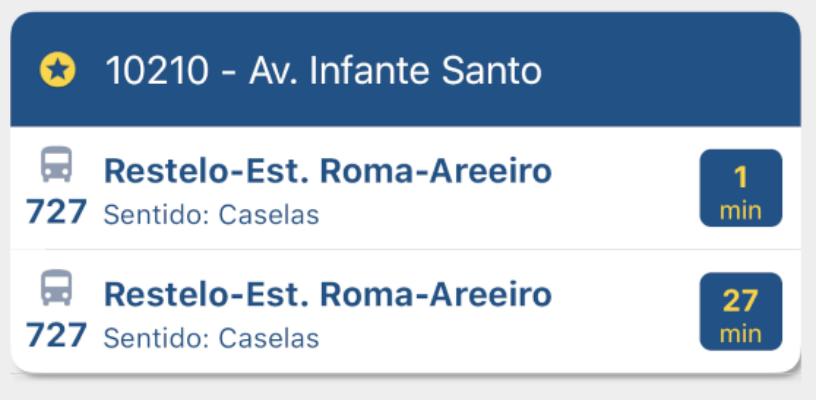


Figura 51 - Estimativas de tempo de uma paragem na Tab Paragens

A informação, representada na Figura 51, está organizada da seguinte forma:

- "10210 – Av. Infante Santo" – Número público e o nome da paragem;
- "727" – Número da linha;
- "Restelo-Est.Roma-Areeiro" – Nome da linha;
- "Caselas" – Sentido da linha;
- "1 min" – Tempo estimado de chegada do autocarro à paragem.

Através da seleção de uma das estimativas (Figura 52), o utilizador subscreve a mesma, podendo assim receber notificações sobre o estado do autocarro na chegada à paragem. Por exemplo, se subscrever uma estimativa que indique 11 minutos, o utilizador receberá no mínimo 4 notificações, caso não faça o cancelamento da subscrição. Estas notificações indicam quando é que faltam 10, 5, 2 e 1 minutos para o autocarro chegar à paragem (Figura 53). O utilizador poderá ainda receber notificações caso o autocarro se atrase. Quando a estimativa de tempo chegar ao fim é diretamente apontada para a próxima estimativa, considerando que o número, o nome e o sentido da linha sejam iguais.

A subscrição é cancelada através destas três situações:

- O utilizador cancelou através da *Tab Home*, representado na Figura 54, ou se selecionar a estimativa subscrita;
- Caso não exista uma próxima estimativa que contenha o mesmo número, o nome e o sentido da linha;
- Caso perfaça 2 horas desde o momento em que o utilizador subscreveu a estimativa.

Figura 52 - Seleção de uma estimativa de tempo

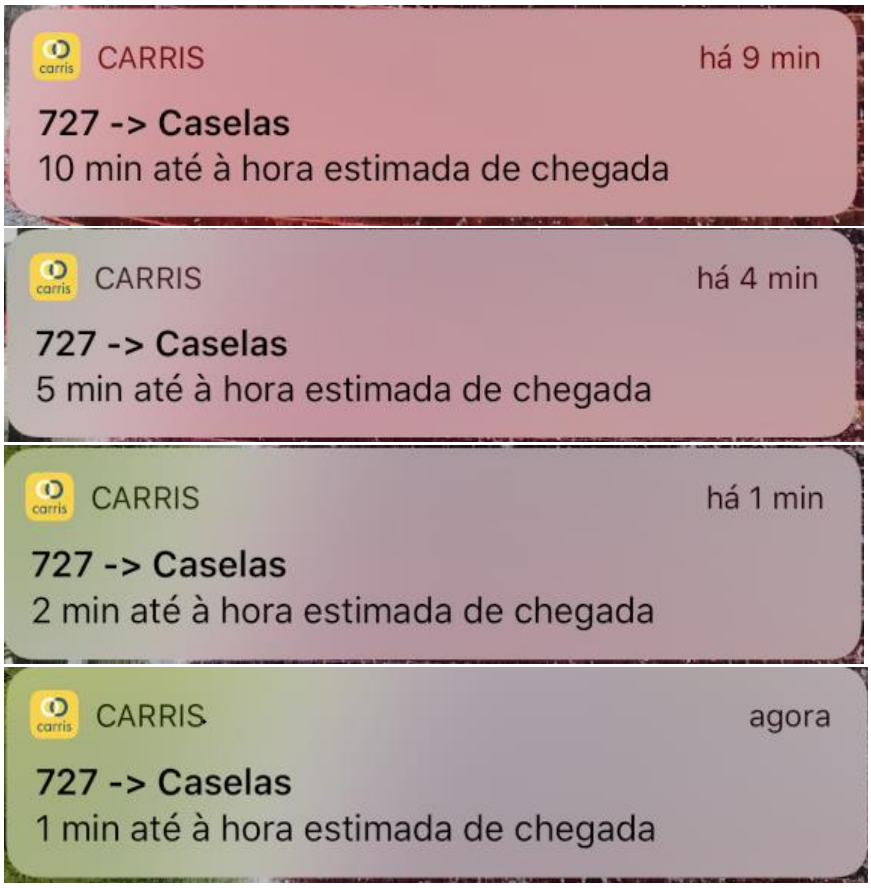


Figura 53 - Notificações de 10, 5, 2 e 1 minutos

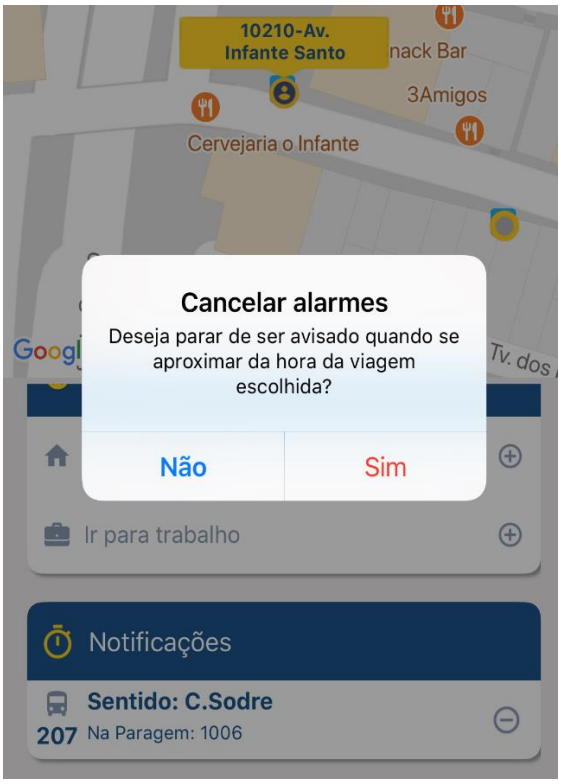


Figura 54 - Cancelamento de uma estimativa de tempo

Quando a estimativa é subscrita, as aplicações Android e iOS criam um *timer* interno ao nível da *Application*. Este *timer* permite fazer o pedido das estimativas de tempo à API, sendo realizado de 30 em 30 segundos, permitindo assim que a estimativa subscrita se mantenha atualizada. Antes de cada pedido é guardado o tempo da estimativa como fator de comparação da "nova", fazendo o cálculo para agendar as notificações e/ou lançar a notificação em como o autocarro se atrasou.

Durante esta implementação surgiram diversos problemas na construção de um algoritmo que permitisse fazer este cálculo sem causar problemas de performance na aplicação. Como só estão a ser utilizadas notificações locais, inicialmente, foi bastante difícil gerir os cálculos e o agendamento das notificações, sendo necessário utilizar uma cache que permitisse fazer a gestão destas. A solução ideal seria utilizar Notificações *Push* mas devido à limitação dos recursos não foi possível optar por esse tipo de abordagem.

Layout do Ecrã Inicial

Na aplicação Android, os dados desta vista estão organizados através de uma *RecyclerView* na sua raiz e cada objeto contido nela é do tipo *CardView*¸ compondo assim um conjunto de *cards* para os diferentes objetos. Já na aplicação iOS utilizou-se uma UITableView e cada *cell* da tabela é manipulada diretamente, sempre com o objetivo de tornar o design das duas aplicações o mais semelhante possível.

Nas duas aplicações, adotou-se um sistema de *Clustering*, como se pode verificar na Figura 55, uma vez que existe um número alargado de paragens pertencentes ao sistema da Carris. Assim, permitiu agrupar as diversas paragens com o objetivo de melhorar o desempenho da aplicação.

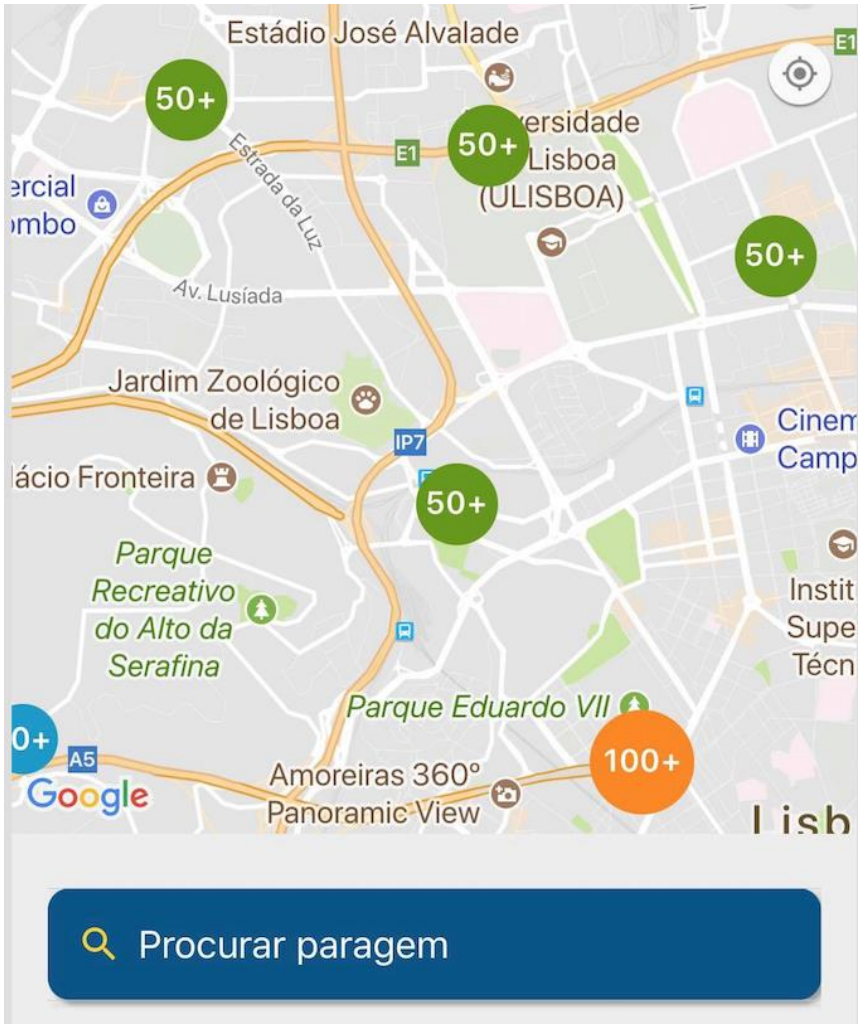


Figura 55 - Clustering de paragens na aplicação iOS

Durante a realização deste ecrã surgiram diversas dificuldades na construção do conjunto dos objetos. Na aplicação Android, foi necessário construir uma lista com itens, onde cada item tinha uma lista de sub-itens. Já no iOS, foi realizada uma manipulação ao *header* da tabela e às suas respetivas *sections.*

6.2.5. Testes realizados

No final do desenvolvimento de cada fase da aplicação Carris, foi necessário realizar testes para verificar se a aplicação estava de acordo com o que se pretendia e se as funcionalidades implementadas estavam a funcionar de forma correta. Os testes realizados foram:

- Testes funcionais;
- Testes de integração;
- Testes de usabilidade.

Os testes funcionais, também conhecidos por testes de funcionalidade, são utilizados no desenvolvimento do software, no qual este é testado para garantir que está em conformidade com os requisitos definidos [74]. O software é testado através do fornecimento de alguns *inputs* relacionados de modo a que o *output* possa ser avaliado com base na conformidade dos requisitos definidos. Um exemplo de um teste funcional da aplicação Carris está representado na Tabela 10. Os restantes testes realizados encontram-se no Anexo D - Testes funcionais da Aplicação Carris.

Tabela 10 - Teste funcional da aplicação Carris

Descrição	Pré-condições	Saídas Esperadas
Planear uma viagem	Ter um ponto de origem e destino	É apresentado uma viagem

Os testes de integração consistem em testar a combinação entre os vários módulos do sistema. Estes testes são alimentados pelos módulos agrupando-os assim em componentes, como estipulado no plano de teste, que resulta num sistema integrado. O objetivo dos testes de integração consiste em verificar os requisitos funcionais, de desempenho e de confiabilidade na

modelagem do sistema [75]. Com estes testes é possível descobrir erros de interfaces, violações de integridade de ficheiros e estruturas de dados globais, problemas de configurações, tratamento de erros, entre outros. Um exemplo de um teste de integração da aplicação Carris está representado na Tabela 11. Os restantes testes realizados encontram-se no Anexo E - Testes de Integração da Aplicação Carris.

Tabela 11 - Teste de integração da aplicação Carris

Descrição	Pré-condição	Saídas Esperadas
Fazer um pedido das linhas à API	A aplicação Carris instalada com ligação à internet	Guarda as linhas da base de dados local do smartphone

Os testes de usabilidade pretendem avaliar a eficácia, a eficiência e a satisfação do cliente. Este é avaliado posteriormente quanto à sua eficiência bem como à sua facilidade de interação.

Durante o desenvolvimento da aplicação Carris foram realizadas várias alterações de modo a corrigir problemas de usabilidade que iam sendo identificados. Na altura da entrega do documento foi realizado um teste de usabilidade mais detalhado, o guia encontra-se disponível no Anexo F - Guia de utilização. Este teste foi feito a 15 pessoas entre os 21 e 65 anos de idade. Dessas 15, 3 acompanharam o desenvolvimento da aplicação Carris desde o início do desenvolvimento. Os tempos de execução das tarefas ficaram em média dentro do tempo esperado, tendo sido mais rapidamente executadas pelas 3 pessoas familiarizadas com o sistema e mais lentamente por aqueles que menos usam o telemóvel no seu dia a dia.

Após a realização destes testes, foram pedidas várias sugestões sobre o que poderiam melhorar na aplicação. De seguida, serão apresentadas algumas das sugestões para cada ecrã da aplicação.

No ecrã inicial:

- Reduzir o tamanho do mapa
- Reduzir o tamanho dos ícones do mapa
- Aumentar o tamanho das letras dos botões
- Diminuir o espaço entre os botões e o texto
- O utilizador poder subscrever mais do que uma estimativa de tempo
- Adicionar uma *view* de carregamento para o refrescamento das estimativas
- Adicionar uma *badge* azul para se destacar os tempos das estimativas
- Adicionar uma *badge* azul para se destacar o número de uma linha favorita

No ecrã de pesquisa de paragens:

- Adicionar uma *badge* azul para se destacar a distância da paragem
- Aumentar o tamanho dos ícones
- Aumentar o tamanho das células

No ecrã de pesquisa de linhas:

- Adicionar uma *badge* azul para se destacar o número da linha
- Aumentar o tamanho dos ícones
- Aumentar o tamanho das células

No ecrã de detalhes de uma linha:

- Selecionar no *header* para mostrar a forma original da linha
- Ajustar o espaço entre o número e o nome da linha

No ecrã de detalhes de uma paragem:

- Adicionar uma *view* de carregamento para o refrescamento das estimativas

No ecrã do planeador de viagem:

- Adicionar uma *badge* azul para se destacar o tempo da estimativa
- Ajustar os espaços entre as *badges* do resultado do planeador

6.3. Síntese do capítulo

Neste capítulo foi apresentado o trabalho realizado, nomeadamente a XTranPassenger API e a aplicação Carris. Foi apresentado o modelo de domínio das plataformas, incluindo a explicação do mapeamento realizado na API que deu origem aos DTO's utilizados na aplicação Carris. Ainda foi apresentada a arquitetura, as principais funcionalidades e a implementação da aplicação Carris. Por último foram apresentados os testes realizados, incluindo os testes funcionais, os testes de integração e os testes de usabilidade.

7. Conclusões e trabalho futuro

Com o aumento dos dispositivos móveis e da sua diversidade, o número de aplicações móveis tem vindo a aumentar nos últimos anos. A dependência atual do utilizador com os *smartphones* e o uso frequente das aplicações móveis no dia a dia faz com que a criação de uma aplicação móvel seja uma boa solução.

O estágio consistiu no desenvolvimento da aplicação Carris, de raiz. Foi implementada, com o principal objetivo de colocar em serviço e manutenção, uma solução digital de integração da informação sobre os serviços da Carris. Para além disto foi uma forma da Carris promover a sua imagem e um modo de promover a utilização dos transportes públicos na área Metropolitana de Lisboa. Neste relatório foram abordadas diversas características sobre a forma de desenvolvimento de aplicações móveis com o intuito de mostrar as dificuldades que possam aparecer durante o desenvolvimento, nomeadamente à limitação de certas ferramentas utilizadas.

Os objetivos inicialmente definidos foram cumpridos. Foi criada uma aplicação funcional à medida do utilizador que permitisse informá-lo sobre os diversos serviços da Carris. Durante a implementação foram realizados testes funcionais, de integração e de usabilidade. Nestes últimos testes, apesar da amostra de utilizadores ter sido relativamente pequena complementou-se com os comentários dos utilizadores nas *stores*.

O desenvolvimento da aplicação móvel Carris permitiu fundamentar novos conhecimentos em desenvolvimento Android e iOS. Com o uso da ferramenta *Cross-Platform* Xamarin surgiu a vantagem de haver um reaproveitamento do código, existindo bibliotecas partilhadas entre as plataformas. Desta forma estas bibliotecas poderão ser utilizadas em futuras aplicações móveis ou em projetos já existentes.

As aplicações iOS e Android foram oficialmente lançadas no dia 20 de março de 2018 [76] e colocadas nas respetivas *stores* (Figura 56), Google Play e Apple Store [77] [78]. Até à data, a aplicação Android conta com 51000 instalações e a aplicação iOS conta 29000 instalações[15].

A primeira etapa do projeto Carris ficou concluída e aprovada pelo cliente, faltando alguns requisitos a serem implementados, nomeadamente o início e acompanhamento de uma

[15] Dados retirados das plataformas App Store Connect e Google Play Console no dia 07/09/2018

viagem planeada. Estes requisitos serão implementados numa segunda etapa do projeto, com outras possíveis funcionalidades propostas pelo cliente. Devido às aplicações estarem colocadas nas *stores* é necessário fazer a análise de possíveis erros ou *crashes*, bem como, observar os comentários feitos pelos utilizadores para possíveis melhorias nas funcionalidades existentes. Para além disso é necessário manter as aplicações atualizadas de modo a serem compatíveis com as versões mais recentes do sistema operativo.

Com o impacto da aplicação Carris, surgiram outros clientes, nomeadamente clientes do Brasil e da Carristur, interessados em obter o mesmo tipo de serviço.

Do trabalho realizado neste relatório resultou o artigo com o título "*Public transportation using Carris app*", que foi aceite para apresentação na *International Conference on Interactive Mobile Communication, Technologies and Learning* (IMCL) [79] de 2018.

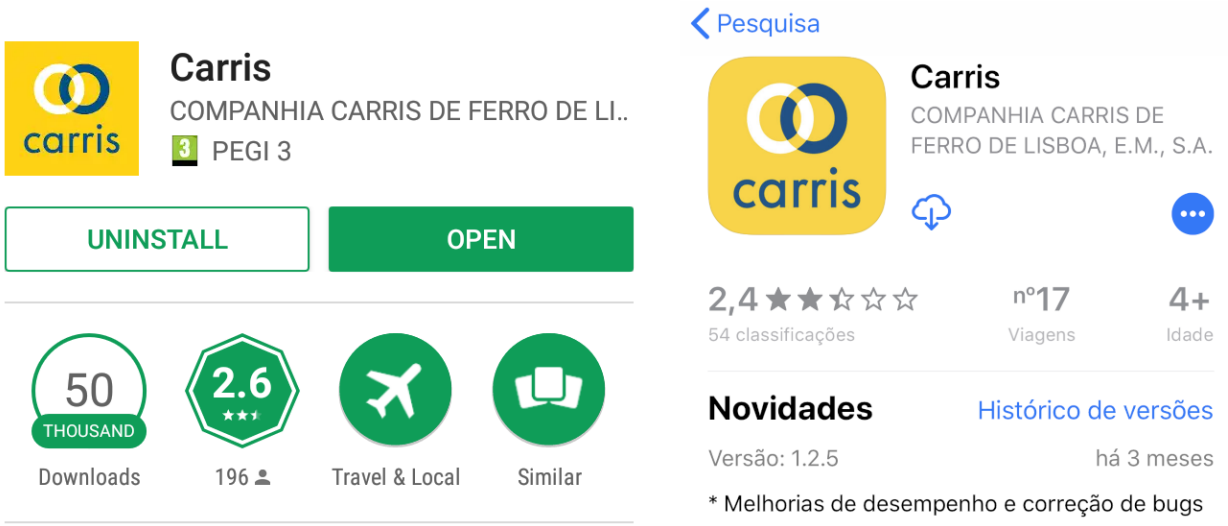


Figura 56 - Aplicação Carris nas stores

7.1. Trabalho futuro

Apesar da primeira etapa do projeto estar concluída, existem sempre várias funcionalidades que poderiam ter sido melhoradas. É importante e necessário existir uma evolução e otimização dos sistemas. Na aplicação Carris é apresentado como trabalho futuro:

- Implementação do início e acompanhamento de uma viagem planeada;

- Implementação de um sistema de bilhética que permita adquirir passes ou bilhetes da Carris;
- Utilização dos serviços da Google nos resultados do planeador e de pesquisa de locais;
- Utilização de *push notifications* nas estimativas de tempo.

Esta página foi intencionalmente deixada em branco

Bibliografia

[1] "Escola Superior de Tecnologia e Gestão." [Online]. Available: https://www.ipleiria.pt/estg/. [Accessed: 26-Jul-2018].

[2] "Missão, Visão e Valores." [Online]. Available: http://www.carris.pt/pt/missao-visao-e-valores/. [Accessed: 13-Sep-2018].

[3] "Tempo de Espera." [Online]. Available: http://www.carris.pt/pt/informacao-ao-passageiro/. [Accessed: 06-Sep-2018].

[4] "Carris utiliza o XTraN Passenger - Sistema de Ajuda à Exploração e Informação aos Passageiros - TECMIC." [Online]. Available: http://www.tecmic.com/carris-utiliza-o-xtran-passenger-sistema-de-ajuda-a-exploracao-e-informacao-aos-passageiros/. [Accessed: 12-Sep-2018].

[5] "Grupo Aitec." [Online]. Available: http://www.aitec.pt/. [Accessed: 26-Jul-2018].

[6] "INESC - HOME." [Online]. Available: http://www.inesc.pt/en/. [Accessed: 26-Jul-2018].

[7] "The Linux Kernel Archives." [Online]. Available: https://www.kernel.org/. [Accessed: 12-Mar-2018].

[8] "Download Android Studio and SDK Tools | Android Studio." [Online]. Available: https://developer.android.com/studio/index.html. [Accessed: 12-Mar-2018].

[9] "Getting Started with Eclipse." [Online]. Available: http://www.eclipse.org/getting_started/. [Accessed: 12-Mar-2018].

[10] "IntelliJ IDEA: The Java IDE for Professional Developers by JetBrains." [Online]. Available: https://www.jetbrains.com/idea/. [Accessed: 12-Mar-2018].

[11] "Update the IDE and SDK Tools | Android Studio." [Online]. Available: https://developer.android.com/studio/intro/update.html. [Accessed: 12-Mar-2018].

[12] "PureDarwin | Moving the Darwin Community in the Right Direction!" [Online]. Available: http://www.puredarwin.org/. [Accessed: 11-Mar-2018].

[13] "Xcode - Apple Developer." [Online]. Available: https://developer.apple.com/xcode/. [Accessed: 11-Mar-2018].

[14] "Xcode - Interface Builder - Apple Developer." [Online]. Available: https://developer.apple.com/xcode/interface-builder/. [Accessed: 11-Mar-2018].

[15] “Calling Java from Kotlin - Kotlin Programming Language.” [Online]. Available: https://kotlinlang.org/docs/reference/java-interop.html. [Accessed: 20-Mar-2018].

[16] “Apple Beats Microsoft at Releasing a Universal Clipboard Feature.” [Online]. Available: http://news.softpedia.com/news/apple-beats-microsoft-to-releasing-a-universal-clipboard-feature-505219.shtml. [Accessed: 21-Mar-2018].

[17] “Android vs iOS - Difference and Comparison | Diffen.” [Online]. Available: https://www.diffen.com/difference/Android_vs_iOS. [Accessed: 21-Mar-2018].

[18] “How to multitask more efficiently with 3D Touch on iPhone 6s.” [Online]. Available: http://appleinsider.com/articles/15/10/02/how-to-multitask-more-efficiently-with-3d-touch-on-iphone-6s. [Accessed: 21-Mar-2018].

[19] “Apple Vs Android—A comparative study 2017 – AndroidPub.” [Online]. Available: https://android.jlelse.eu/apple-vs-android-a-comparative-study-2017-c5799a0a1683. [Accessed: 21-Mar-2018].

[20] “Android Fragmentation Report August 2015 - OpenSignal.” [Online]. Available: https://opensignal.com/reports/2015/08/android-fragmentation/. [Accessed: 22-Mar-2018].

[21] “Gartner Says Worldwide Sales of Smartphones Grew 9 Percent in First Quarter of 2017.” [Online]. Available: https://www.gartner.com/newsroom/id/3725117. [Accessed: 25-Mar-2018].

[22] M. C. Ferreira, T. Fontesz, V. Costa, T. G. Dias, J. L. Borges, and J. F. E Cunha, “Evaluation of an integrated mobile payment, route planner and social network solution for public transport,” *Transp. Res. Procedia*, vol. 24, pp. 189–196, 2017.

[23] K. Zavitsas, I. Kaparias, and M. G. H. Bell, “Transport problems in cities.”

[24] C. Samsel, S. Beul-Leusmann, M. Wiederhold, K.-H. Krempels, M. Ziefle, and E.-M. Jakobs, “Cascading Information for Public Transport Assistance.”

[25] “Citymapper - Metro & Comboios – Aplicações Android no Google Play.” [Online]. Available: https://play.google.com/store/apps/details?id=com.citymapper.app.release&hl=pt_PT. [Accessed: 30-Jan-2018].

[26] “Google on Android - Official Google Mobile Blog.” [Online]. Available: http://googlemobile.blogspot.pt/2008/09/google-on-android.html. [Accessed: 05-Apr-2018].

[27] “Apple Maps App Officially Debuts, Google Maps Dropped (PHOTOS) | HuffPost.” [Online]. Available: https://www.huffingtonpost.com/2012/06/11/apple-debuts-maps-app_n_1587726.html. [Accessed: 05-Apr-2018].

[28] M. T. Françoso and N. Custódio De Mello, “INFLUÊNCIA DOS APLICATIVOS DE SMARTPHONES PARA TRANSPORTE URBANO NO TRANSITO.”

[29] “Google Maps.” [Online]. Available: https://www.google.com/maps/@39.7424367,-8.8307734,15z. [Accessed: 05-Apr-2018].

[30] “Google Maps - App Store.” [Online]. Available: https://itunes.apple.com/pt/app/google-maps-gps-navigation/id585027354?mt=8. [Accessed: 14-Jan-2018].

[31] “Lisboa MOVE-ME – Aplicações Android no Google Play.” [Online]. Available: https://play.google.com/store/apps/details?id=com.opt.lisboamoveme&hl=pt_PT. [Accessed: 28-Feb-2018].

[32] “MOVE-ME.AMP – Aplicações Android no Google Play.” [Online]. Available: https://play.google.com/store/apps/details?id=com.moveme&hl=pt_PT. [Accessed: 30-Jan-2018].

[33] “Coimbra.MOVE-ME – Aplicações Android no Google Play.” [Online]. Available: https://play.google.com/store/apps/details?id=com.moveme.coimbra. [Accessed: 28-Feb-2018].

[34] “OPT | Produtos | MOVE-ME - Ligações Intermodais em Tempo Real.” [Online]. Available: http://www.opt.pt/produto.asp?codProduto=14. [Accessed: 05-Apr-2018].

[35] “OPT.” [Online]. Available: http://www.opt.pt/. [Accessed: 01-Mar-2018].

[36] J. D. A. Anes, “Evolução Move-Me,” Aug. 2014.

[37] “Moovit - Transporte Público.” [Online]. Available: https://www.company.moovitapp.com/pt-recursos?utm_medium=Organic&utm_source=company_website. [Accessed: 26-Jan-2018].

[38] “Gestão de Transportes de Passageiros - XTraN Passenger - TECMIC.” [Online]. Available: http://www.tecmic.com/portfolio/xtran-passenger/. [Accessed: 08-Mar-2018].

[39] “An Android-Based Personalized Public Transportation Advisor.”

[40] M. Georgiev, S. Jana, and V. Shmatikov, “Breaking and Fixing Origin-Based Access Control in Hybrid Web/Mobile Application Frameworks.,” *NDSS Symp.*, vol. 2014, pp. 1–15, Feb. 2014.

[41] L. Luquetti, B. Da Silva, D. F. Pires, and S. C. Neto, “Desenvolvimento de Aplicações para Dispositivos Móveis: Tipos e Exemplo de Aplicação na plataforma iOS Alternative Title: Application Development for Mobile Devices: Types and Application Example on the iOS platform.”

[42] N. K. Nizamettin Gok, *Building Hybrid Android Apps with Java and JavaScript*. O’Reilly Media, 2013.

[43] “Welcome to Ionic - Ionic Framework.” [Online]. Available: https://ionicframework.com/docs/v1/guide/preface.html. [Accessed: 01-Mar-2018].

[44] “Architectural overview of Cordova platform - Apache Cordova.” [Online]. Available:

https://cordova.apache.org/docs/en/latest/guide/overview/. [Accessed: 01-Mar-2018].

[45] B. Canavesi, "BENEFITS AND DISADVANTAGES OF HYBRID MOBILE APPLICATIONS," 2016. [Online]. Available: https://www.linkedin.com/pulse/benefits-disadvantages-hybrid-mobile-applications-brooks-canavesi/. [Accessed: 04-Feb-2018].

[46] S. Eder, E. C. Conforto, D. C. Amaral, S. Luis, and D. Silva, "Diferenciando as abordagens tradicional e ágil de gerenciamento de projetos Palavras-chave," *Production*, 2012.

[47] "SELECTING A DEVELOPMENT APPROACH," 2005.

[48] M. D. S. Soares and M. dos S. Soares, "Metodologias Ágeis Extreme Programming e Scrum para o Desenvolvimento de Software," *Rev. Eletrônica Sist. Informação*, vol. 3, no. 1, Jun. 2004.

[49] A. Filipe and S. Leal, "Forces Smart Teams – Inteligência nas Forças de Segurança."

[50] J. R. Fernandes, "SAFER Response," 2013.

[51] "Manifesto para o Desenvolvimento Ágil de Software." [Online]. Available: http://agilemanifesto.org/iso/ptpt/manifesto.html. [Accessed: 19-Apr-2018].

[52] "Skype | Ferramenta de comunicação para chamadas e conversas gratuitas." [Online]. Available: https://www.skype.com/pt/. [Accessed: 19-Apr-2018].

[53] "Boards | Trello." [Online]. Available: https://trello.com/. [Accessed: 03-Apr-2018].

[54] "Apache Subversion." [Online]. Available: https://subversion.apache.org/. [Accessed: 03-Apr-2018].

[55] "Git - Documentation." [Online]. Available: https://git-scm.com/doc. [Accessed: 03-Apr-2018].

[56] "Share Code. Track Work. Ship Software. | Team Foundation Server." [Online]. Available: https://www.visualstudio.com/tfs/. [Accessed: 03-Apr-2018].

[57] "Agile Tools | Visual Studio Team Services." [Online]. Available: https://visualstudio.microsoft.com/team-services/agile-tools/. [Accessed: 10-Jul-2018].

[58] "Architecture - Xamarin.Android | Microsoft Docs." [Online]. Available: https://docs.microsoft.com/en-us/xamarin/android/internals/architecture. [Accessed: 09-May-2018].

[59] "iOS Architecture - Xamarin | Microsoft Docs." [Online]. Available: https://docs.microsoft.com/en-us/xamarin/ios/internals/architecture. [Accessed: 09-May-2018].

[60] “Home | Mono.” [Online]. Available: https://www.mono-project.com/. [Accessed: 10-May-2018].

[61] “What is managed code? | Microsoft Docs.” [Online]. Available: https://docs.microsoft.com/en-us/dotnet/standard/managed-code. [Accessed: 10-May-2018].

[62] “Overview of Visual Studio 2017 - Visual Studio | Microsoft Docs.” [Online]. Available: https://docs.microsoft.com/en-us/visualstudio/ide/visual-studio-ide. [Accessed: 29-May-2018].

[63] “Conheça o Android Studio | Android Developers.” [Online]. Available: https://developer.android.com/studio/intro/. [Accessed: 30-May-2018].

[64] “Transporlis > Home.” [Online]. Available: http://www.transporlis.pt/Default.aspx?tabid=36. [Accessed: 29-Apr-2018].

[65] “Update your navigation system map - HERE Maps (NAVTEQ).” [Online]. Available: https://mapupdate.navigation.com/landing/en-GB/index.html. [Accessed: 29-Apr-2018].

[66] “JSON.” [Online]. Available: https://www.json.org/. [Accessed: 08-Jul-2018].

[67] “SQLite Home Page.” [Online]. Available: https://www.sqlite.org/index.html. [Accessed: 16-May-2018].

[68] “Top 5 Most Popular Database for Mobile Apps | Trigent.” [Online]. Available: https://blog.trigent.com/five-of-the-most-popular-databases-for-mobile-apps/. [Accessed: 16-May-2018].

[69] “Getting Started Guide — AutoMapper documentation.” [Online]. Available: http://docs.automapper.org/en/stable/Getting-started.html. [Accessed: 13-Jun-2018].

[70] “What is NuGet and what does it do? | Microsoft Docs.” [Online]. Available: https://docs.microsoft.com/en-us/nuget/what-is-nuget. [Accessed: 13-Jun-2018].

[71] “Guidelines and Resources - App Store - Apple Developer.” [Online]. Available: https://developer.apple.com/app-store/guidelines/. [Accessed: 19-May-2018].

[72] “Activity | Android Developers.” [Online]. Available: https://developer.android.com/reference/android/app/Activity#ActivityLifecycle. [Accessed: 24-May-2018].

[73] “UIViewController - UIKit | Apple Developer Documentation.” [Online]. Available: https://developer.apple.com/documentation/uikit/uiviewcontroller. [Accessed: 24-May-2018].

[74] “What is Functional Testing? - Definition from Techopedia.” [Online]. Available: https://www.techopedia.com/definition/19509/functional-testing. [Accessed: 13-Jul-2018].

[75] “Testing in Software Development - Martyn A. Ould, British Computer Society. Working Group on Testing - Google Books.” [Online]. Available: https://books.google.pt/books?id=utFCImZOTEIC&pg=PA73&dq=integration+test&hl=en&sa=X&ei=4EpTVOvJMayu7Aak5YCIDA&redir_esc=y#v=onepage&q=integration test&f=false. [Accessed: 13-Jul-2018].

[76] “Com a nova app da Carris pode saber quanto tempo falta para o autocarro.” [Online]. Available: https://nit.pt/out-of-town/back-in-town/a-nova-app-da-carris-diz-lhe-quanto-tempo-falta-para-chegar-o-autocarro. [Accessed: 30-Jul-2018].

[77] “Carris - Apps on Google Play.” [Online]. Available: https://play.google.com/store/apps/details?id=pt.carris.tecmic. [Accessed: 30-Jul-2018].

[78] “Carris na App Store.” [Online]. Available: https://itunes.apple.com/pt/app/carris/id1321889486?mt=8. [Accessed: 30-Jul-2018].

[79] “IMCL2018 – International Conference on Interactive Mobile Communication, Technologies and Learning.” [Online]. Available: http://www.imcl-conference.org/current/. [Accessed: 25-Sep-2018].

Anexo A - Requisitos Funcionais da Aplicação Carris

Planeamento de uma viagem
Definir um ponto de partida • **Localização atual** • **Escolha de um local no mapa** • **Inserção de uma morada**
Definir um ponto de chegada • **Escolha de um local no mapa** • **Inserção de uma morada**
Alterar a hora da viagem
Guardar uma viagem planeada

Informação do utilizador
Guardar paragens
Guardar linhas

Informação na espera
Apresentação das três paragens mais próximas do utilizador
Caso o utilizador selecione uma paragem no mapa é apresentado as três paragens mais próximas do ponto selecionado
Apresentação de estimativas de tempo nas paragens, com o número e sentido da linha
Subscrição de uma estimativa: • **Apresentação de notificações quando faltar 10, 5, 2 e 1 minutos até o autocarro chegar à paragem** • **Atrasos desde a última notificação**
Tempos de espera refrescados de 30 em 30 segundos

Informação sobre as próximas paragens
Enquanto o utilizador estiver numa viagem ativa, baseado na sua localização, deve poder visualizar as próximas paragens
Se houver mais segmentos numa viagem planeada, a próxima paragem de destino deve atualizar de forma automática
Deve ser possível ao utilizador indicar que iniciou a sua viagem planeada
Deve ser possível indicar que terminou a sua viagem ou a aplicação perceber isso automaticamente pela posição do utilizador

Anexo B - Documentação da XTranPassenger API

Paragens

Nome do pedido	Descrição do pedido
/api/BusStops	Retorna uma lista de paragens
/api/BusStops/timestamp/{timestamp}	Retorna uma lista de paragens com mudanças através do *timestamp* enviado
/api/BusStops/near/lat/{lat}/lon/{lng}	Retorna uma lista de 3 paragens mais próximas do ponto enviado por parâmetro.

Linhas

Nome do pedido	Descrição do pedido
/api/Routes	Retorna uma lista de linhas
/api/Routes/timestamp/{timestamp}	Retorna uma lista de linhas com mudanças através do *timestamp* enviado
/api/Routes/{routeNumber}	Retorna uma linha pelo seu código público (intitulado routeNumber)

Planeador de viagem

Nome do pedido	Descrição do pedido
/api/Planner/startlat/{startLat}/startlon/{startLon} /endLat/{endLat}/endLon/{endLon}/start/{startDate}	Planeamento de uma viagem de um ponto para outro

Estimativas

Nome do pedido	Descrição do pedido
/api/Estimations	Verifica se existe uma conexão válida ao estimador de tempos.
/api/Estimations/busStop /{busStop}/top/{results}	Retorna uma lista de estimativas com um máximo de resultados (results) para uma determinada paragem passada por parâmetro.
/api/Estimations/estimationbusstop /{busStops}/top/{results}	Retorna uma lista de estimativas para múltiplas paragens. O resultado retorna dados das paragens, cada uma contendo a sua lista de estimativas.

Geocoding

Nome do pedido	Descrição do pedido
/api/Geocoding/{search}	Retorna uma lista de possíveis pontos correspondentes ao endereço fornecido (search)
/api/Geocoding/reverse/lat/{lat}/lng/{lng}	Retorna um endereço para o ponto geográfico especificado

/api/Geocoding/suggest/{query}	Preenchimento automático para entradas do utilizador, para que ele possa procurar um endereço
/api/Geocoding/fromId/{locationId}	Retorna os dados de um endereço através de uma pesquisa do utilizador

Esta página foi intencionalmente deixada em branco

Anexo C - Classes do domínio com os seus atributos

BusStopDto {

id	integer($int32)
name	string
publicId	string
lat	number($double)
lng	number($double)
isPublicVisible	boolean
timestamp	string($date-time)

}

EstimationDto {

routeNumber	string
routeName	string
destination	string
time	string($date-time)
busNumber	string
plateNumber	string
voyageNumber	integer($int32)

}

NearbyBusStopDto {

distance	number($double)
id	integer($int32)
name	string
publicId	string
lat	number($double)
lng	number($double)
isPublicVisible	boolean
timestamp	string($date-time)

}

EstimationBusStopDto {

estimationList	[**EstimationDto** {...}]
id	integer($int32)
name	string
publicId	string
lat	number($double)
lng	number($double)
isPublicVisible	boolean
timestamp	string($date-time)

}

GeoResultDto {

lat	number($double)
lng	number($double)
address	string
header	string
isLocalMatch	boolean

}

GeoSuggestionDto {

locationId	string
address	string
header	string

}

SolutionDto {

beginDateTime	string($date-time)
endDateTime	string($date-time)
steps	[SolutionStepDto{...}]
extents	BoundingBoxDto{...}

}

SolutionStepDto {

begin	StepPointDto{...}
end	StepPointDto{...}
beginDateTime	string($date-time)
endDateTime	string($date-time)
travelMode	integer($int32)Enum: Array [8]
distance	number($float)
price	number($float)
waitingTime	string
travelTime	string
description	string
coordinates	[CoordinateDto{...}]
alert	string

}

BoundingBoxDto {

coordinate1	CoordinateDto{...}
coordinate2	CoordinateDto{...}

}

StepPointDto {

id	string
name	string
coordinate	CoordinateDto{...}
dateTime	string($date-time)
locality	string

}

CoordinateDto {

lat	number($double)
lng	number($double)

}

RouteDto {

id	integer($int32)
routeNumber	string
name	string
isPublicVisible	boolean
timestamp	string($date-time)

}

FullRouteDto {

isCirc	boolean
variants	[VariantDto{...}]
id	integer($int32)
routeNumber	string
name	string
isPublicVisible	boolean
timestamp	string($date-time)

}

VariantDto {

id	integer($int32)
variantNumber	integer($int32)
isActive	boolean
upItinerary	ItineraryDto{...}
downItinerary	ItineraryDto{...}
circItinerary	ItineraryDto{...}

}

ItineraryDto {

id	integer($int32)
type	integer($int32)Enum: [0, 1, 2]
connections	[ConnectionDto{...}]
shape	string

}

ConnectionDto {

id	integer($int32)
distance	number($float)
orderNum	integer($int32)
busStop	BusStopDto{...}

}

Esta página foi intencionalmente deixada em branco

Anexo D - Testes funcionais da Aplicação Carris

Pressupostos: A aplicação Carris está instalada

Descrição	Pré-condições	Saídas Esperadas
Planear uma viagem	Ter um ponto de origem e destino	É apresentado uma viagem
Definir um favorito (Casa ou Trabalho)	N/A	O favorito é guardado na base de dados local do sistema
Apresentação das paragens mais próximas do utilizador	GPS ativo e existe ligação à Internet	São apresentadas as 3 paragens mais próximas do utilizador
Procurar uma paragem	Lista de paragens guardadas na base de dados do sistema	É apresentado a paragem procurada
Apresentação das estimativas de tempo de uma paragem	Existe ligação à Internet	São apresentadas as estimativas de tempo de uma paragem
Procurar uma linha	Lista de linhas guardadas na base de dados do sistema	É apresentado a linha procurada
Apresentação dos detalhes de uma linha	A linha é valida (com variantes ativas) Existe ligação à Internet	É apresentado os detalhes de uma linha
Guardar uma paragem nos favoritos	A paragem é valida	A paragem é guardada na base de dados local do sistema

Guardar uma linha nos favoritos	A linha é valida	A linha é guardada na base de dados local do sistema

Anexo E - Testes de Integração da Aplicação Carris

Descrição	Pré-condição	Saídas Esperadas
Fazer um pedido das linhas à API	A aplicação Carris instalada com ligação à internet	Guarda as linhas da base de dados local do smartphone
Fazer um pedido das paragens à API	A aplicação Carris instalada com ligação à internet	Guarda as paragens da base de dados local do smartphone
Iniciar o serviço Google Maps	A aplicação Carris instalada com ligação à internet e uma chave da Google correta	Apresenta o mapa do serviço Google Maps no ecrã principal
Fazer um pedido das estimativas de tempo de uma paragem à API através do serviço Time Estimator	A aplicação Carris instalada com ligação à internet	Apresenta as estimativas relativamente à paragem consultada
Fazer um pedido à API de um local através do serviço Geocoder (Navteq) utilizando a barra de pesquisa da aplicação	A aplicação Carris instalada com ligação à internet	Apresenta várias sugestões de locais

Fazer um pedido à API de um local através do serviço Geocoder (Navteq) através da escolha de uma sugestão apresentada	A aplicação Carris instalada com ligação à internet	É guardado o local, temporariamente, para ser utilizado no planeador de viagens
Fazer um pedido à API de um planeamento de uma viagem	A aplicação Carris instalada com ligação à internet e com localizações de origem e destino definidas	Apresenta a viagem ao utilizador no ecrã do planeamento de uma viagem

Anexo F - Guia de utilização

De forma a testar a usabilidade do projeto foi dado o seguinte conjunto de tarefas para o utilizador realizar:

1. Entrar na aplicação.
2. Planear uma viagem .
3. Definir a localização de origem como “Sete Rios”.
4. Escolher o primeiro resultado apresentado na tabela dos resultados.
5. Definir a localização de destino como “Rossio”.
6. Escolher o primeiro resultado apresentado na tabela dos resultados.
7. Voltar até ao ecrã inicial.
8. Selecionar a Tab Paragens.
9. Procurar a paragem 79120 - Via Oriente.
10. Guardar a paragem como favorita.
11. Voltar até ao ecrã inicial.
12. Selecionar a Tab Linhas.
13. Procurar a linha 12E – M.Moniz /Castelo.
14. Guardar a linha como favorita.
15. Fechar a aplicação.

Esta página foi intencionalmente deixada em branco