Relatório de Estágio

Mestrado em Engenharia Informática – Computação Móvel

GreenDriving

Luís Henrique Santos Torres

Leiria, 28 de setembro de 2018

Relatório de Estágio

Mestrado em Engenharia Informática – Computação Móvel

GreenDriving

Luís Henrique Santos Torres

Leiria, 28 de setembro de 2018

Dedicatória

Dedico este projeto à minha família e a todos os meus amigos.

Agradecimentos

Em primeiro lugar quero agradecer à minha família que tanto me apoiou durante todo o meu percurso académico.

Queria deixar um especial agradecimento ao João Domingos, Pedro Marques e Joel Fernandes, da Tecmic S.A. por todo o apoio, formação e ensinamentos transmitidos durante todo o período de estágio.

Quero também agradecer ao Professor Ricardo Martinho, pela orientação dada durante o desenvolvimento e estruturação do presente relatório.

Resumo

O presente documento descreve o período de estágio curricular no âmbito do ano final de Mestrado em Engenharia Informática – Computação Móvel, durante o qual foi desenvolvida uma aplicação de apoio à eficiência energética de frotas de veículos. Foram desenvolvidas duas arquiteturas para realizar este projeto. Uma arquitetura recorre a um microcontrolador Bluetooth ELM327 que traduz a interface OBD (*on-board diagnostics*) presente na maioria dos veículos atuais. A outra recorre a uma caixa proprietária da entidade de acolhimento denominada MK3 que se conecta a um tablet Android através de uma porta serial. A informação recolhida é então tratada, transformada e apresentada em tempo real num dispositivo móvel Android. Essa informação é subdividida em 3 principais âmbitos, informação em tempo real, estatísticas e códigos de diagnostico do veículo.

Existe atualmente um protótipo funcional que será colocado nos dispositivos alvo, encontrando-se numa fase pré testes de aceitação por parte dos motoristas.

Com esta plataforma, a Tecmic espera reforçar a sua posição no mercado relativo a software de eficiência energética, em que os seus clientes poderão usufruir de uma poupança de combustível até 20%.

Abstract

This document describes the curricular internship period which took event in the final year of the Master's degree in Computer Science - Mobile Computing, during which an application to enhance the energy efficiency of vehicle fleets was developed. Two architectures were developed to carry out this project. One architecture uses an ELM327 Bluetooth microcontroller that translates the raw that of the OBD (on-board diagnostics) interface present in most current vehicles. The other uses a proprietary box controller called MK3 (provided by the internship host entity) that connects to an Android tablet via a serial port. The collected information is then treated, transformed and presented in real time in an Android mobile device. This information is subdivided into 3 main scopes, real-time information, statistics and vehicle diagnostic trouble codes.

There is currently a functional prototype that will be placed on the target devices, which is currently in a pre-acceptance test fase that will be accomplished by the fleet drivers. With this platform, Tecmic hopes to strengthen its position in the market for energy efficiency software, where its customers can save up to 20% on fuel consumption.

Índice

Índice de Figuras

Índice de Tabelas

Lista de Siglas

SIGLA	DESCRIÇÃO
ECU	Engine Control Unit
OBD	On Board Diagnostics
PID	Parameter Identifier
ISO	International Organization for Standardization
IDE	Integrated Development Environment
MVVM	Model-View-ViewModel
GPS	Global Positioning System
FMS	Fleet Management System
API	Application Programming Interface
PWA	Project Web Application
DTC	Diagnostic Trouble Codes
MAF	Mass Air Flow

1. Introdução

O presente relatório foi elaborado no âmbito do Estágio Curricular decorrido no 2º ano do Mestrado em Engenharia Informática – Computação móvel, lecionado na Escola Superior de Tecnologia e Gestão de Leiria e supervisionado pelo Professor Ricardo Martinho.

Este estágio decorreu de 28 de agosto de 2017 a 18 de junho de 2018, nas instalações da unidade de Leiria da empresa Tecmic S.A, situada na IDDNET (Incubadora D. Dinis), sob orientação de João Domingos e Pedro Marques.

Durante o decorrer do estágio pretendeu-se criar uma aplicação móvel (GreenDriving) que comunique com a unidade de controlo do motor do veículo, recolhendo, em tempo real, informação relativa à condução. De seguida, pretende-se transformar e tratar esses dados, contextualizando-os em padrões de eficiência energética, apoio à condução e prevenção na degradação dos veículos.

Deste modo, este relatório pretende documentar todo o trabalho realizado, pesquisa e aprendizagem relativos ao período de estágio e ao desenvolvimento da aplicação GreenDriving, contextualizando-a no seu mercado aplicacional, definindo os seus requisitos funcionais, o processo de planeamento, as dificuldades encontradas, as opções tomadas, as metodologias utilizadas, a descrição e esquemática do projeto, bem como as funcionalidades que se pretendam realizar em trabalho futuro.

1.1. Entidade de Acolhimento

A entidade de acolhimento deste estágio denomina-se Tecmic S.A. O nome Tecmic deriva de "Tecnologias de Microeletrónica". A empresa foi fundada em 1988 e pertence atualmente ao grupo Trailtec.

A principal vertente de desenvolvimento da Tecmic passa pela gestão de frotas de veículos, oferecendo um leque de soluções desde componentes de *hardware* a aplicações de *software* que os complementam.

A Tecmic desenvolve soluções de alta qualidade para empresas de renome como a Luís Simões e a Carris, entre muitas outras.

1.2. Enquadramento

Neste documento pretende-se descrever o trabalho realizado durante o período de estágio no âmbito do segundo ano de Mestrado em Computação Móvel. Este trabalho consiste no desenvolvimento de uma plataforma digital que inclui uma aplicação móvel para apoio a uma condução mais eficiente e económica de frotas de veículos. Mais especificamente, pretende-se criar um produto que ajude os condutores a analisar a sua condução em tempo real bem como a longo prazo, recorrendo a métricas bem definidas relativas à eficiência da sua condução.

O desenvolvimento deste projeto segue no contexto de um módulo já existente na gama de soluções oferecidas pela Tecmic S.A. denominado "Atividade Energética". Este módulo é responsável pela análise dos dados de frotas, permitindo reduções de custos de mais de 20%, otimizando o consumo de combustível e diminuindo os custos de manutenção.

Atualmente o módulo de Atividade Energética da Tecmic S.A existe num formato de aplicação *web* em tempo não-real, sendo que a informação é recolhida, processada e posteriormente apresentada em forma de relatórios de eficiência energética de condução. Desde modo surge a necessidade de criar um módulo para dispositivos móveis (a aplicação GreenDriving), para que haja a possibilidade de alertar em tempo real o condutor do seu modo de condução/estatísticas de consumo energético, diminuindo o período entre a recolha de dados, o seu processamento e finalmente a sua apresentação.

Na implementação atual (aplicação *web*), apenas os operadores de frota têm acesso a informação de eficiência energética, o que se significa que os condutores não têm apoio em tempo real para melhorarem o seu modo de condução. A aplicação móvel GreenDriving serve então o propósito de colmatar a falta de informação em tempo real disponibilizada aos condutores.

1.3. Motivação e Objetivos

Uma das principais motivações no desenvolvimento da aplicação passa pela empresa Tecmic disponibilizar aos seus clientes uma solução de apoio a uma condução eficiente em tempo real, diminuindo os custos de manutenção dos veículos dos clientes e aumentando a sua autonomia.

A necessidade de desenvolvimento da aplicação passa também por um dos problemas que atualmente afetam a humanidade, o consumo elevado de combustíveis fósseis que contribuem para a poluição do planeta. Com o objetivo de minimizar este problema as empresas de transportes têm apostado cada vez mais em técnicas e soluções de redução de consumo de combustível das suas frotas.

Existem atualmente alguns estudos realizados com o propósito de averiguar se a utilização de aplicações móveis no apoio à condução resultam numa redução no consumo de combustível por parte dos condutores. Um estudo realizado por Johannes Tulusan, Thorsten Staake e Elgar Fleisch, denominado "*Providing eco-driving feedback to corporate car drivers: what impact does a smartphone application have on their fuel efficiency?*"[1] recorreu a uma amostra de condutores para avaliar se o apoio de uma aplicação móvel levaria a uma redução no consumo de combustível. Este estudo teve resultados positivos na redução de consumo de combustível. Neste estudo são também referenciados outros estudos com resultados positivos em que houve um incentivo monetário adicional aliado ao apoio de uma aplicação móvel.

Como foi referido na secção anterior a Tecmic já disponibilizou a alguns dos seus clientes uma solução de atividade energética (aplicação *web* não apropriada para dispositivos móveis), sendo que o objetivo deste trabalho passou por desenvolver uma solução direcionada para dispositivos móveis e com informação disponível em tempo real, para "guiar" o condutor numa direção de melhoria na eficiência energética enquanto conduz.

Foi também referido anteriormente que o módulo de eficiência energética atual é apenas apresentado em formato web, não sendo em tempo real e apenas disponível para

operadores de frota, deste modo, os condutores não percebem exatamente como e quando corrigir comportamentos que contrariem uma condução económica e energeticamente eficiente, surgindo então a necessidade de criação de uma aplicação móvel e em tempo real de apoio ao condutor.

Portanto, o principal objetivo deste trabalho foi a criação da aplicação móvel GreenDriving e respetiva integração com os módulos Bluetooth e caixa proprietária MK3 da Tecmic, bem como o desenvolvimento de competências e métodos de trabalho num âmbito empresarial. Para tal houve um período de adaptação para conhecer a empresa, as suas metodologias de trabalho bem como as pessoas que a constituem.

1.4. Estrutura do Documento

Neste primeiro capítulo é introduzido o tema deste projeto de estágio, onde é também referenciada a entidade de acolhimento e os principais objetivos deste estágio.

No segundo capítulo é apresentado o estado da arte, listando algumas das aplicações já existentes no mercado, indicando os seus pontos fortes e eventuais desvantagens do ponto de vista do objetivo da aplicação GreenDriving. É também descrita uma solução já existente no leque de aplicações da Tecmic e serviu de base para o desenvolvimento da aplicação GreenDriving.

No terceiro capítulo é descrito o planeamento seguido durante o período de estágio, bem como a metodologia de desenvolvimento aplicada na realização do projeto.

No capítulo quatro é descrito o projeto desenvolvido, mais especificamente as principais características que constituem a aplicação. São descritas as tecnologias utilizadas na realização deste projeto, as arquiteturas desenvolvidas e os vários passos tomados durante todo o período de implementação.

No capítulo cinco são apresentados os modelos do projeto, algoritmos implementados, protótipos e *layouts*.

Finalmente, o capítulo seis conclui o relatório, resumindo todo o trabalho realizado ao longo deste estágio e referindo o possível trabalho futuro.

2. Estado da arte

Neste capítulo é contextualizada a aplicação móvel GreenDriving no seu respetivo mercado aplicacional, apresentando também exemplos de aplicações já existentes, as suas características e objetivos específicos, bem como os critérios de pesquisa dessas aplicações e o motivo destas serem apresentadas neste documento. São identificados também os prós e contras de cada uma dessas aplicações e o seu contraste com o objetivo principal da aplicação GreenDriving.

Finalmente é apresentada uma tabela comparativa, evidenciando as principais funcionalidades disponibilizadas pela aplicação GreenDriving em relação às aplicações mencionadas.

2.1. Critérios de Pesquisa

Neste subcapítulo são referidos os critérios de pesquisa e o motivo pelo qual as aplicações descritas se encontram no estado da arte.

Relativamente aos repositórios aplicacionais foram escolhidas aplicações que se encontram disponíveis para download na *Google PlayStore* ou na *AppStore* da *Apple*. Esta opção aliou-se ao facto de serem aplicações para dispositivos móveis que o utilizador pode facilmente descarregar e utilizar.

Todas as aplicações referidas recorrem a uma arquitetura de comunicação semelhante, recorrendo a uma comunicação *Bluetooth* entre o dispositivo móvel e um microcontrolador ELM327 para aceder à informação do veículo.

Foram excluídas todas as aplicações *Desktop* que recorressem a um computador para aceder à informação dos veículos, ou aplicações que apenas apresentem informação relativa a alarmes ou erros encontrados no veículo. Foram também excluídas aplicações cujo objetivo principal passa por modificar controlos e parâmetros do veículo.

Deste modo as aplicações descritas de seguida foram escolhidas porque, além de cumprirem os critérios acima referidos, apresentam funcionalidades que foram consideradas interessantes de apresentar e discutir no âmbito deste projeto.

As chaves de pesquisa utilizadas nos repositórios referidos foram: OBD (*On Board Diagnostics*); OBD2; ELM327; DTC's (*Diagnostic Trouble Codes*); *Eco Driving*; *Scan* OBD *Tools*. Do resultado destas pesquisas foram encontradas várias aplicações que, depois de aplicados os critérios acima mencionados resultaram na escolha das 4 aplicações referidas de seguida.

2.2. Torque Pro

A aplicação Torque Pro [2] está disponível para download tanto para iOS como para Android e representa uma das opções mais escolhidas pelos utilizadores. Apresenta um ecrã com dados em tempo real customizável, uma vez que o utilizador pode escolher os parâmetros que pretende destacar no ecrã.

Figura 1 – Ecrãs de Estatísticas da aplicação Torque Pro

A possibilidade de se poder customizar um ecrã pode ser aliciante e ter as suas vantagens partindo do pressuposto de que o utilizador está dentro das tecnologias. No entanto, muitas vezes isso pode não acontecer tendo como consequência deixar o utilizador baralhado.

É possível aceder à informação do veículo sobre as suas emissões e códigos de erro. Possui também um ecrã de navegação *Global Positioning System* (GPS) que permite ao utilizador associar um percurso a uma viagem e verificar os consumos de combustível, período da viagem, entre outros aspetos.

Um dos principais aspetos presente na aplicação GreenDriving e que não existe na aplicação Torque Pro é a capacidade de analisar a eficiência da condução.

2.3. BlueDriver OBD2 Scan Tool

A aplicação BlueDriver OBD2 *Scan Tool* [3] encontra-se disponível para download tanto na *Android Play Store* como na *Apple Store*.

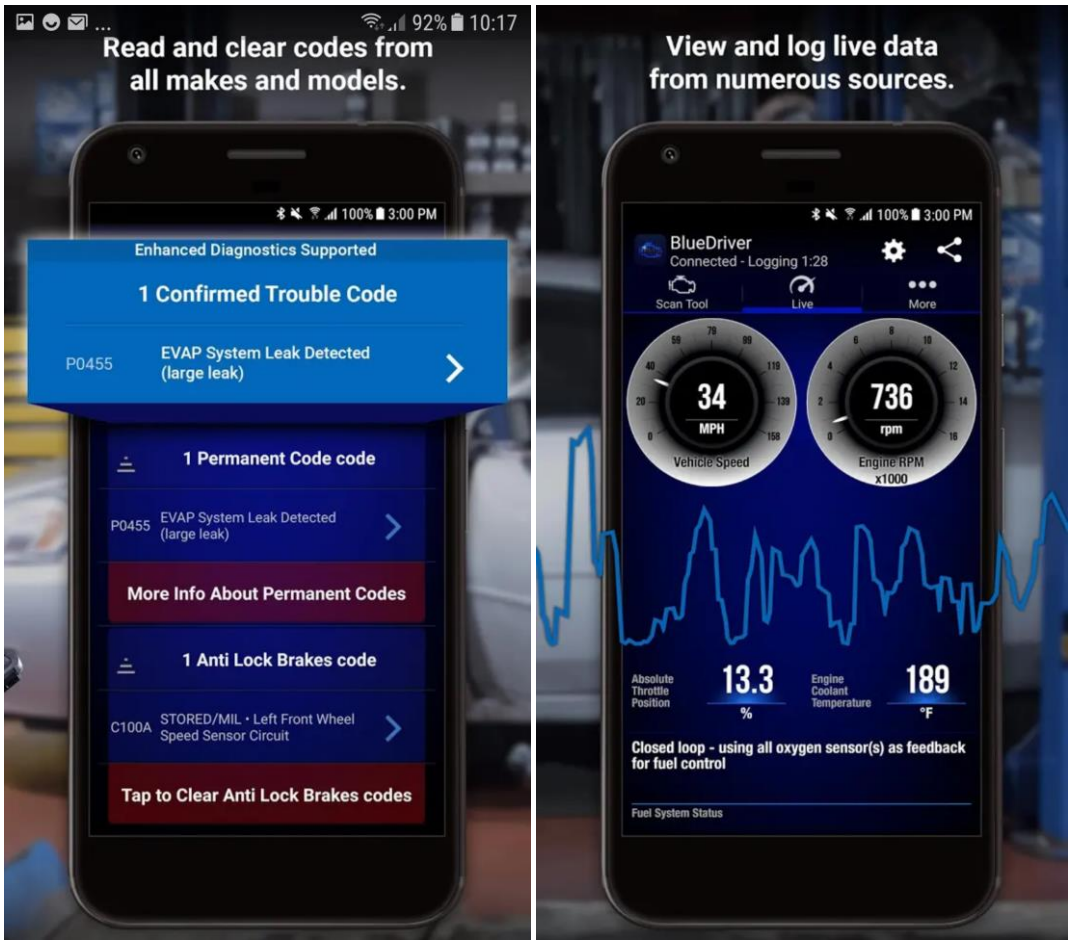


Figura 2 – Ecrãs de Alarmísticas e de Tempo Real da aplicação BlueDriver

A aplicação apresenta um ecrã com dados do veículo em tempo real como a posição do acelerador, rotações, estado do combustível entre outros. É possível analisar códigos de erro que sejam detetados no veículo bem como a possibilidade de os apagar (remover a sua anotação do *dashboard* do veículo). Uma das funcionalidades de destaque desta aplicação é a maneira como contextualiza um determinado erro detetado apresentando uma solução baseada em mais de 6.6 milhões de soluções reportadas para determinados códigos de erro, dividindo essas soluções em 3 principais camadas divididas em *ranking*. Deste modo o utilizador terá ao seu dispor, além da descrição do código de

erro, uma possível solução ou lista de soluções, apresentando também uma previsão de como o veículo poderá ser reparado.

Além dos aspetos acima referidos, a aplicação BlueDriver apresenta também informação sobre as emissões de gases poluentes do veículo.

Apesar de apresentar dados em tempo real a aplicação não os contextualiza num determinado tipo de condução.

Outra desvantagem da aplicação é que não funciona com dispositivos ELM *third party*. Isto significa que o utilizador terá de comprar um sensor exclusivo para ligar ao veículo denominado BlueDriver OBD2 Sensor.

2.4. DashCommand (OBD ELM App)

A aplicação DashCommand [4] encontra-se disponível para download na *Android Play Store*. Apresenta ecrãs com dados em tempo real como aceleração, cavalagem e tração instantânea do veículo, bem como o consumo de combustível atual e em média relativa a até 3 viagens.

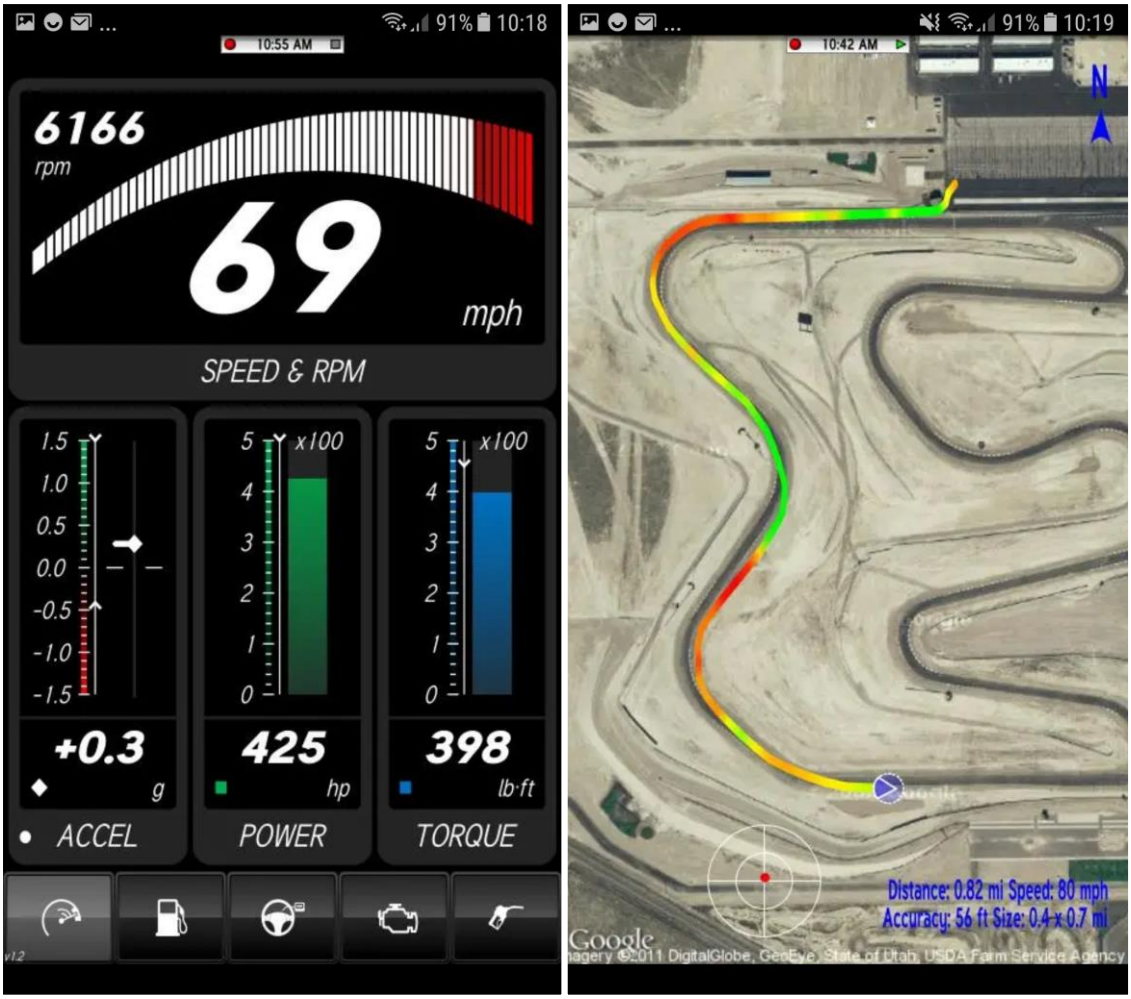


Figura 3 – Ecrãs de Tempo Real e Análise de Percurso da aplicação DashCommand

A aplicação apresenta também uma estimativa da distância média que o veículo poderá viajar até ficar sem combustível.

Umas das principais funcionalidades da aplicação é a capacidade de guardar estatísticas até 5 viagens, indicando num ecrã de mapas uma determinada viagem e quais as zonas do percurso onde o condutor apresentou uma condução económica (indicando uma linha a verde por cima da zona do percurso) ou as zonas onde o utilizador tem uma condução mais "agressiva" (indicando a zona do percurso a vermelho). Deste modo, os utilizadores poderão, *a posteriori*, analisar a sua condução e possivelmente detetar certos hábitos de condução que poderão aumentar o consumo de combustível bem como as emissões de gases provenientes da sua combustão.

As vistas de tempo real da aplicação são customizáveis dando ao utilizador algum poder de decisão.

2.5. Smart Control Pro (OBD & Car)

A aplicação Smart Control Pro (OBD & Car) [5] está disponível para download na Android Play Store. Apresenta ecrãs com dados em tempo real como velocidade, cavalagem, tração instantânea do veículo, carga no motor, consumo de combustível, entre outros.

Figura 4 – Aplicação Smart Control Pro (OBD & Car) apresentando estatísticas de condução e diagnósticos de viagens

Uma das funcionalidades principais desta aplicação passa por apresentar estatísticas dos dados adquiridos do veículo, bem como a possibilidade de iniciar e finalizar uma viagem, associando as estatísticas a um determinado percurso, guardando uma lista de vários percursos.

Esta aplicação apresenta em cada um dos seus ecrãs um considerável volume de informação (ver Figura 4).

2.6. Análise Comparativa entre Aplicações

Neste subcapítulo é apresentada uma análise comparativa entre as aplicações acima descritas e a aplicação GreenDriving, de acordo com as funcionalidades consideradas mais relevantes no âmbito deste projeto.

Tabela 1 – Análise Aplicacional Comparativa

Funcionalidades	Torque Pro	Dash Command	Blue Driver	Smart Control Pro	GreenDriving
Dados em Tempo Real	X	X	X	X	X
Estatísticas	X	X	X	X	X
Alarmisticas	X	X	X	X	X
Cálculos de Condução Económica		X		X	X
Análise de Custo de Viagens	X	X		X	
Contadores de Eventos de Condução não Económica					X
Apoio a Condução Económica em Tempo Real		X		X	X

Como podemos observar na Tabela 1, todas as aplicações possuem funcionalidades comuns, das quais, dados em tempo real, estatísticas e alarmísticas (DTC's - *Diagnostic Trouble Codes*).

É possível observar que nem todas as aplicações apresentam informação de apoio a uma condução económica. Uma das principais razões poderá dever-se ao facto do seu motivo inicial não ser necessariamente esse. Muitos utilizadores recorrem a este tipo de aplicações com o objetivo de obter informações sobre componentes do veículo que normalmente não se encontram disponíveis no *dashboard* do veículo.

No entanto algumas das aplicações apresentadas disponibilizam também funcionalidades interessantes para o apoio a uma condução económica, como é o caso da aplicação Torque Pro, DashCommand e Smart Control Pro que apresenta custos de viagem para um determinado percurso. As aplicações DashCommand e Smart Control Pro apresentam também funcionalidades de analise de condução, atribuindo uma determinada cotação às viagens efetuadas, indicando valores como o valor médio de consumo de combustível associado.

No caso da aplicação Dash Command, é também indicado em que zonas do percurso se registou um consumo maior ou menor de combustível, para uma análise mais aprofundada dos trechos dos percursos utilizados.

Podemos observar que, apesar da quantidade de informação disponibilizada por algumas das aplicações indicadas, o modo como esta se encontra distribuída pelas aplicações poderá levar a eventuais distrações do utilizador. No âmbito do apoio a uma condução económica torna-se importante apresentar a informação ao condutor de uma forma menos distraidora, evitando perdas de concentração na condução. Deste modo pretende-se que a informação em cada ecrã seja clara e concisa. Se o condutor sentir a necessidade de olhar atentamente para o ecrã para extrair informação isso poderá comprometer uma condução segura.

O desenvolvimento da aplicação GreenDriving vem então tentar colmatar alguns objetivos não implementados na totalidade pelas aplicações descritas. Um dos principais objetivos passará por apresentar a informação necessária e relevante num só ecrã, de maneira a evitar, tanto quanto possível, a distração por parte dos utilizadores. É pretendido que a aplicação categorize ainda o modo de condução atual e o consumo de

combustível instantâneo. Pretende-se também que haja medidores de tempo real que apresentem informação adicional acerca de modos de condução diretamente associados ao consumo de combustível.

É pretendido ainda apresentar contadores de ações associadas a um gasto de combustível adicional, indicando o número de vezes em que se incorre numa determinada ação, ou o tempo que se despende numa determinada ação. Adicionalmente é desejado apresentar cartões informativos com um fundo de cor representativa da classificação dada na ação em questão (Ex: Número de travagens bruscas pode estar a verde, amarelo ou vermelho).

Tendo isto em conta, e recorrendo à visão periférica, o utilizador poderá obter uma informação geral da sua condução atual sem que, para isso, tenha de olhar atentamente para o ecrã do dispositivo ou executar gestos, evitando assim perdas de concentração na condução desnecessárias.

3. Metodologia e Planeamento

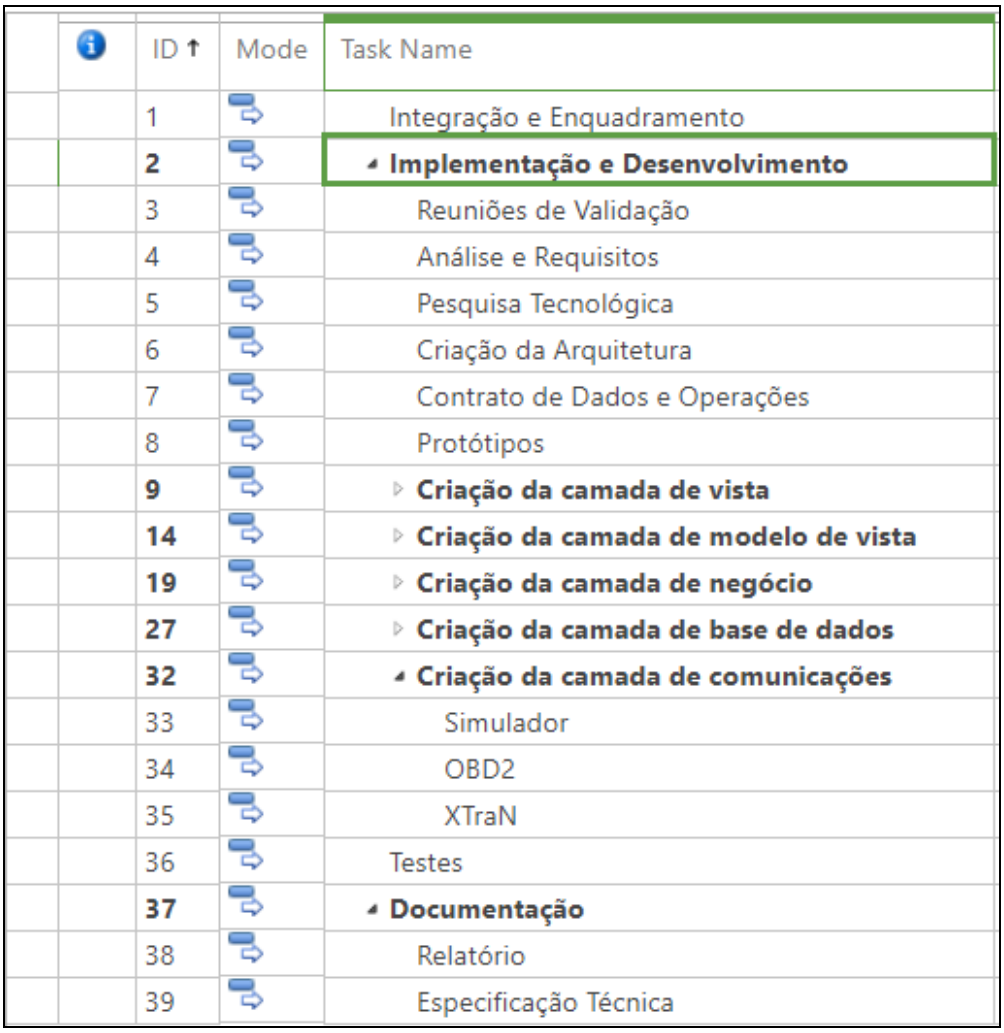
Neste capítulo é indicado o planeamento desde a fase inicial do projeto até à sua fase final, indicando as decisões tomadas por parte da empresa e qual o motivo direcionado no sentido do sucesso do produto desenvolvido.

De seguida é descrita a metodologia de desenvolvimento utilizada no âmbito da aplicação móvel GreenDriving, refletindo como o desenvolvimento aplicacional e as relações entre os elementos da empresa funcionam.

3.1. Planeamento

Para o planeamento deste projeto recorreu-se a uma ferramenta utilizada na empresa denominada de PWA (*Project Web Application*), ferramenta esta que recorre ao programa Microsoft Project numa versão Web.

É no PWA que são registadas as tarefas a realizar, a data de início, e o tempo despendido em cada tarefa. Depois de efetuada uma determinada tarefa, é então submetida para análise realizada pelos superiores hierárquicos.



ID	Mode	Task Name
1		Integração e Enquadramento
2		**Implementação e Desenvolvimento**
3		Reuniões de Validação
4		Análise e Requisitos
5		Pesquisa Tecnológica
6		Criação da Arquitetura
7		Contrato de Dados e Operações
8		Protótipos
9		**Criação da camada de vista**
14		**Criação da camada de modelo de vista**
19		**Criação da camada de negócio**
27		**Criação da camada de base de dados**
32		**Criação da camada de comunicações**
33		Simulador
34		OBD2
35		XTraN
36		Testes
37		**Documentação**
38		Relatório
39		Especificação Técnica

Figura 5 – ScreenShot retirado da aplicação Microsoft Project

A Figura 5 representa o planeamento criado numa fase inicial do período de estágio, apresentando as principais funcionalidades a realizar. As funcionalidades estão subdivididas em duas principais áreas: 1) Implementação e Desenvolvimento; e 2) Documentação.

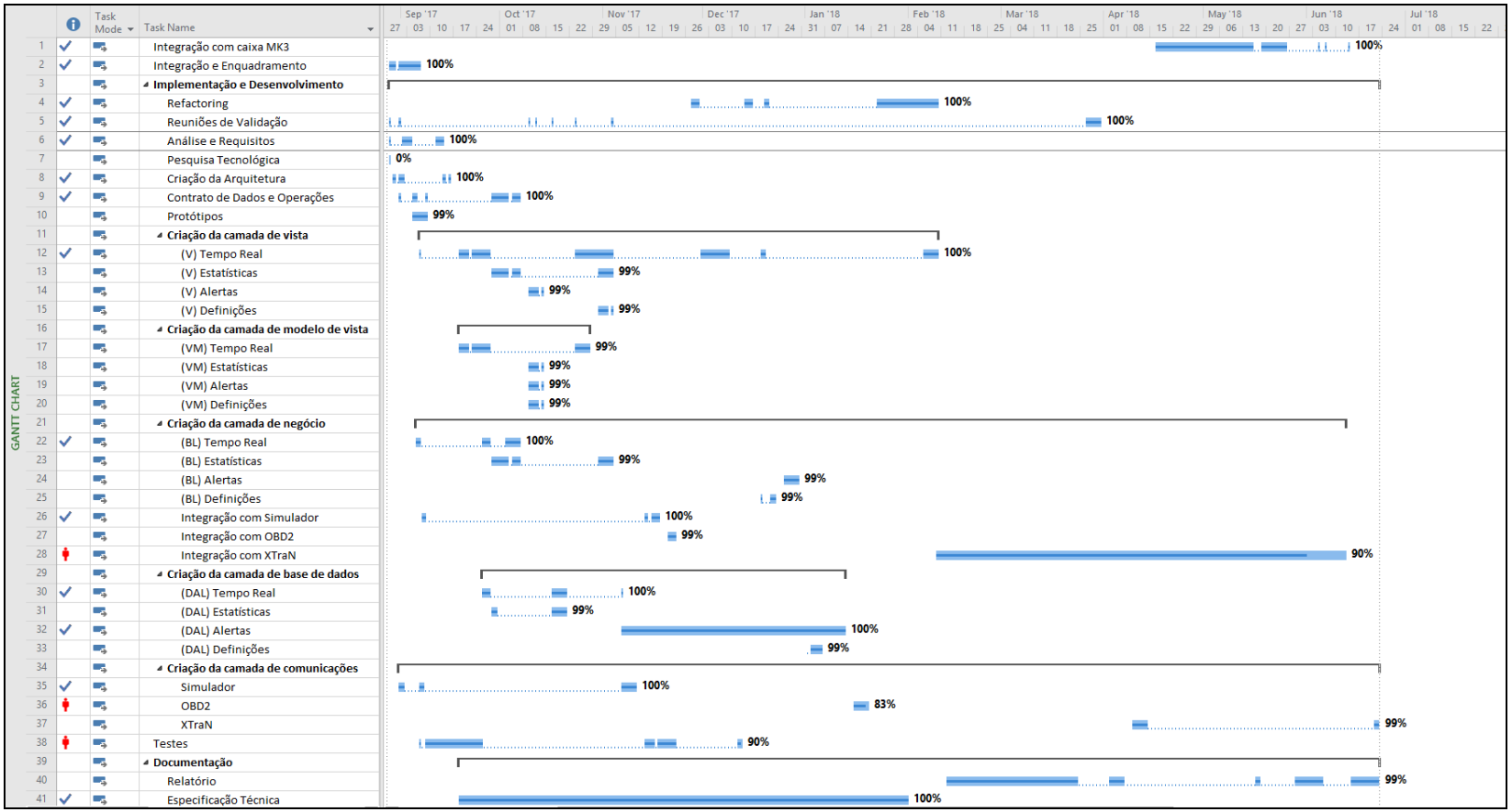


Figura 6 - Esquema de gestão de tempo

A Figura 6 representa esquematicamente as funcionalidades efetuadas e que foram inseridas no programa PWA ao longo do tempo.

Como podemos visualizar, a maior parte das funcionalidades foram desenvolvidas iterativamente ao longo do tempo, devido a constantes reformulações, novas ideias e *refactoring*.

3.2. Metodologia

Este projeto surgiu inicialmente de uma ideia base para uma aplicação de apoio a uma condução sustentável. Deste modo, ao longo deste estágio houve períodos de pesquisa e recolha de informação, criação de protótipos, desenho de interface, implementação e testes. No entanto, estes processos nem sempre seguiram uma ordem concreta, existindo vários períodos de reestruturação. Estas modificações derivaram de decisões arquiteturais que inicialmente não estavam completamente definidas, uma vez que se tratava de um teste de conceito. Estas decisões incluíam aspetos como o modo de comunicação arquitetural ou o dispositivo móvel onde a aplicação iria ser utilizada, que foram sofrendo alterações ao longo do estágio.

Tipicamente eram efetuadas pelo menos uma a duas reuniões semanais via *Skype*. Essas reuniões consistiam na revisão do trabalho efetuado até à data, os principais avanços e problemas encontrados, que poderiam, ou não, levar a uma reestruturação do projeto.

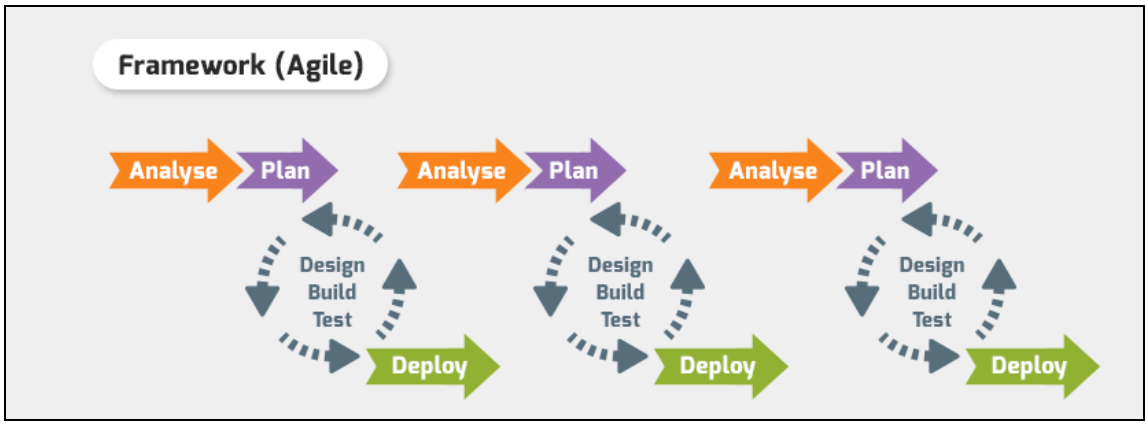


Figura 7 – Esquema de desenvolvimento recorrendo a metodologias ágeis [6]

Como podemos ver na Figura 7, os períodos de desenvolvimento foram tipicamente definidos por um processo de planeamento e análise de requisitos, onde se definiram as principais funcionalidades desejadas na iteração em questão. De seguida, houve períodos de desenvolvimento de uma a duas semanas, dependo da complexidade ou dificuldades encontradas. No final de cada iteração eram analisadas as implementações efetuadas, bem como a viabilidade a curto, médio e longo prazo das mesmas.

Caso se aprovasse a viabilidade das implementações efetuadas era feita uma nova análise de funcionalidades a incluir para a próxima iteração.

Caso se verificasse que o planeamento efetuado iria causar um aumento excessivo de complexidade, ou se se chegasse à conclusão que a opção tomada se tornou inviável eram analisadas as outras opções existentes. Caso necessário, poderiam mudar-se os aspetos não viáveis e dava-se seguimento ao desenvolvimento. Num caso mais extremo, na pior das hipóteses o procedimento mais comum seria fazer um *rollback* no desenvolvimento até ao ponto de desenvolvimento considerado necessário.

4. Requisitos, arquitetura e tecnologias da aplicação GreenDriving

No âmbito do projeto de estágio GreenDriving pretende-se avaliar o modo de condução de veículos ligeiros e pesados por parte dos respetivos condutores. Para tal é necessário recolher dados provenientes do veículo, processá-los e contextualizá-los. Alguns destes dados são normalmente apresentados no painel de indicadores do veículo, como é o caso da velocidade atual, rotações, quantidade de combustível no depósito entre outros, variando dependendo da marca e modelo em questão. No entanto as ECU's (*Engine Control Unit*) dos veículos têm disponíveis muitos outros dados que se consideram importantes e que normalmente não são apresentados aos condutores, como é o caso da percentagem de carga no motor do veículo, a posição do pé no acelerador ou travão, consumo de combustível instantâneo e médio, entre outros.

Recorrendo aos dados acima descritos é também possível aferir o número de travagens bruscas, excessos de velocidade, acelerações bruscas, acelerações em curva, bem como tempo de marcha de inércia.

É também possível identificar algum problema que esteja a ocorrer no veículo evitando assim que um condutor circule com uma viatura danificada sem ter a noção disso.

A aplicação GreenDriving pretende ir ao encontro dos casos acima descritos tornando-se numa mais valia para uma condução mais eficiente, reduzindo os consumos de combustível e se possível reduzir e/ou retardar os eventuais desgastes que ocorrem naturalmente nos veículos durante o seu ciclo de vida.

4.1. Principais requisitos

A informação que é apresentada na aplicação é obtida a partir do veículo por duas alternativas.

Na primeira alternativa recorre-se a um microcontrolador ELM327 que traduz os OBD (*On Board Diagnostics*) da ECU (*Engine Controller Unit*) do veículo que por sua vez comunica a informação à aplicação Green Drivingatravés de uma comunicação *Bluetooth* entre o microcontrolador e o dispositivo móvel.

Na segunda opção recorre-se a uma caixa da Tecmic (MK3) que executa um processo idêntico ao do microcontrolador ELM327 acima referido, sendo que depois a própria caixa trata e processa os dados adquiridos da ECU e os comunica à aplicação GreenDriving através de uma porta série (o dispositivo que recebe a informação da caixa trata-se de um tablet com o sistema operativo Android com versão mínima 4.4 Kit-Kat).

Para o desenvolvimento da aplicação móvel GreenDriving existem alguns aspetos necessários que devem ser abordados e mostrados na aplicação de modo a apoiar o condutor a melhorar o seu tipo de condução, tornando-a mais eficiente e se possível com gastos de combustível reduzidos, dos quais se podem resumir os seguintes requisitos funcionais e não-funcionais principais:

- **Informação em tempo real –** apresentação da informação necessária em tempo real na aplicação para apoiarem uma condução sustentável, indicando ao utilizador se a sua condução é económica ou não;

- **Dados acumulados (Contadores) –** cálculo e apresentação de contadores através da acumulação de dados em tempo real indicando as situações em que ocorreram conduções com consumos acima do normal ou pouco eficientes;

- **Alertas –** alertas ao condutor no de alguma anomalia no veículo;

Nos subcapítulos seguintes são descritos os 3 requisitos acima referidos, desdobrados em *user stories* (uma forma de requisitos típica das metodologias ágeis de desenvolvimento de *software*).

4.1.1. User Stories

Nesta secção apresentam-se as principais *user stories* derivadas numa fase inicial de desenvolvimento e conceptualização da aplicação, representando os principais objetivos que serviram de base para a criação da aplicação GreenDriving.

Tabela 2 - Descrição de User Stories

Nº	Descrição
US1	Como utilizador quero receber informação atual relativa ao veículo, em tempo real.
US2	Como utilizador quero poder escolher o dispositivo de conexão ao veículo.
US3	Como utilizador quero poder aceder a estatísticas de condução em tempo real.
US4	Como utilizador quero poder visualizar e interagir com gráficos informativos.
US5	Como utilizador quero ser notificado de alarmes ou alertas encontrados pelo veículo.
US6	Como utilizador quero persistir os dados relativos ao veículo e visualizá-los posteriormente.
US7	Como utilizador quero visualizar dados e quantificadores de condução económica em tempo real.

4.1.2. Tempo Real

No âmbito do projeto GreenDriving pretende-se avaliar o modo de condução de veículos ligeiros e pesados. Para tal, é necessário recolher dados em tempo real provenientes dos veículos que reflitam sobre o modo de condução dos condutores. Da variedade de dados disponíveis de adquirir do veículo são selecionados determinados atributos que melhor ajudam a compreender diferentes tipos de condução, dos quais:

- **Consumo Instantâneo** – Representa os litros consumidos ao longo de 100 quilómetros de acordo com o modo de condução atual.
- **Autonomia de combustível** – Representa uma previsão da distância de autonomia que pode ser percorrida com base na quantidade de combustível e de acordo com o modo de condução atual.
- **Banda Ecológica –** Representa visualmente a classificação da condução recorrendo a mudança de cor nas barras representativas no ecrã de tempo real.
- **Carga no Motor** – Este dado indica a percentagem de carga exercida (esforço) no motor em tempo real.
- **Posição de acelerador** – Representa a percentagem de acelerador exercida no momento actual.
- **Velocidade atual –** Representa a velocidade atual do veículo
- **Rotações do motor –** Representa as rotações atuais do motor do veículo

4.1.3. Contadores

De modo a contextualizar (espaço/tempo) os valores de tempo real obtidos da ECU do veículo, pretende-se detetar e contabilizar dados que verifiquem uma condução fora dos valores pretendidos. Para tal é necessário recorrer a cálculos que são efetuados na caixa

MK3, que já devolve um conjunto de campos calculados ao utilizador de modo a que o mesmo se aperceba da eficiência da sua condução, a saber:

- **Número Travagens Bruscas:** travagens bruscas desgastam os travões/pneus entre outros componentes do carro desnecessariamente. Sendo que, depois de uma travagem brusca é necessário recorrer ao processo de aceleração novamente (aumentando assim os consumos de combustível). Para obter esta informação recorremos a valores subitamente reduzidos de velocidade/rotações do motor.

- **Número Excessos de Velocidade:** nesta fase apenas serão considerados excessos de velocidade se o veículo circular a uma velocidade superior a 120 km/h.

- **Número Acelerações Bruscas:** pretende-se avaliar se a aceleração efetuada foi desnecessariamente acentuada. Para tal podemos também recorrer aos dados de rotação do motor, velocidade, carga no motor e posição do acelerador.

- **Número Acelerações em Curva:** de acordo com informação obtida conclui-se que, ao acelerar-se numa curva há desperdício de combustível. Deste modo pretende-se que os condutores evitem este tipo de condução. Para tal é necessário recorrer a informação sobre a posição do acelerador e o ângulo das rodas/ volante.

- **Tempo de Marcha de Inércia:** uma das melhores maneiras de poupar combustível (sempre que possível) passa por recorrer à marcha de inércia. Neste tipo de condução pretende-se que o condutor mantenha uma determinada mudança "engatada" sem acelerar (Ex: numa descida deixa a 5ª mudança engatada sem acelerar). Deste modo a condução, para além de ser mais segura, reduz também o consumo de combustível, em comparação à opção de meter o carro em ponto morto (torna o carro mais instável e consome mais combustível). Para obter informação sobre a marcha de inercia recorre-se às rotações do motor e posição do acelerador.

Ao guardarmos estes valores conseguimos ter uma noção do modo de condução de condutor para condutor, podemos ver onde uns conseguem poupar combustível e onde outros o gastam excessiva e desnecessariamente ao longo de um determinado espaço de tempo. Sendo esta a principal razão para disponibilizarmos tantos dados de tempo real e

dados acumulativos ao longe do tempo. Uma vez que, se o condutor tentar não obter avisos de condução “agressiva” em tempo real irá melhorar o seu método de condução.

4.1.4. Alarmísticas

De modo a apoiar o condutor na perceção do seu modo de condução torna-se essencial que a aplicação móvel o transpareça facilmente, por exemplo através do um código de cores. Para tal a aplicação GreenDriving poderá apresentar, por exemplo, barras indicativas visualmente parecidas com passadeiras que mudam de cor baseado no tipo de condução em que está a inferir. Deste modo o utilizador poderá recorrer à sua visão periférica para analisar em tempo real qual é a eficiência da sua condução sem necessariamente perder o foco na condução.

Pretende-se também que a aplicação informe o condutor caso algum problema seja detetado no veículo e que apresente informação específica do problema em si. O objetivo passa também por antecipar possíveis casos críticos que obriguem a uma possível paragem de uma determinada viagem o que poderia atrasar planos previamente determinados.

Os alarmes a indicar serão detetados pela caixa MK3, que depois informa a aplicação que por sua vez alerta o utilizador se algo se encontra fora do normal. A informação relativa a alarmes vem codificada de acordo com o *standard Fleet Managment System (FMS)* [7] criado por várias empresas fabricantes de veículos pesados, trata-se de uma interface comum que permite uma conexão de dados segura para os veículos das marcas incluídas no standard.

4.2. Arquitetura geral da plataforma

Nesta secção é descrita a arquitetura adotada para a realização deste projeto. À partida, existem duas maneiras da aplicação móvel GreenDriving se conectar ao veículo.

Uma recorre a um dispositivo ELM que se conecta ao veículo através da sua entrada OBD (*On Board Diagnostics*). Esse dispositivo ELM possui um microcontrolador que comunica com a ECU do carro que, de seguida, comunica esses dados ao dispositivo móvel através de uma conexão *Bluetooth*.
A outra maneira da aplicação móvel se conectar ao veículo recorre a uma caixa MK3 da Tecmic, que trata da ligação à ECU, sendo responsável pela recolha da informação e dados em tempo real do veículo, comunicando por sua vez esses dados ao dispositivo móvel através de uma porta série.
Nas secções seguintes estão descritas estas duas formas de ligação mais pormenorizadamente, evidenciando as vantagens e desvantagens de cada uma.

4.2.1. Caixa MK3 / Tablet

Esta forma de ligação constitui a principal motivação no desenvolvimento deste projeto, uma vez que engloba uma das soluções já existentes no leque de ofertas da Tecmic. A caixa MK3 trata-se de uma consola que se conecta à ECU do veículo, recolhendo a informação disponível relevante para analise de consumos e eficiência energética. Estes dados estão originalmente codificados em vetores (*arrays*) de bytes no formato ASCII.
A maneira como a comunicação entre o veículo e a caixa MK3 é efetuada é a base de fazer um pedido específico com um identificador da questão seguido de uma resposta por parte da ECU.
A caixa MK3 tem incorporada lógica responsável pela descodificação dos dados. Esta consola é também responsável pelos cálculos de eficiência energética. Esses dados, depois de tratados, são enviados para o dispositivo móvel, neste caso um *tablet* com um sistema operativo *Android*.
Existem disponíveis a partir da caixa MK3 um conjunto de *tags*. Cada *tag* representa uma determinada estrutura de dados, previamente agrupada de acordo com a lógica pretendida.
No âmbito deste projeto recorre-se à informação disponibilizada por duas *tags* principais:

- **CBus** – Esta *tag* incorpora dados relativos a indicadores de tempo real, como é o caso dos valores da carga do motor, posição do acelerador, economia de combustível, consumo atual, velocidade, rotações, entre outros.

- **Ecsm** – Esta *tag* devolve os dados dos contadores calculados relativos a aspetos de condução económica. São contabilizados valores relativos ao número de acelerações bruscas, travagens bruscas, tempo de condução em excesso de velocidade, tempo de viagem em marcha de inércia, entre outros. Esta *tag* apresenta também informação alarmística relativa ao veículo.

O objetivo final passa por informar (em tempo real) o utilizador do seu modo de condução, baseado na lógica de eficiência calculada pela caixa MK3.
A ligação entre o dispositivo tablet e a caixa faz-se através de portas série.
Sempre que a caixa tiver novos dados relacionados com as *tags* subscritas, envia-os para o *tablet* para serem posteriormente apresentadas ao utilizador.
Na Figura 8 apresenta-se esquematicamente a arquitetura descrita.

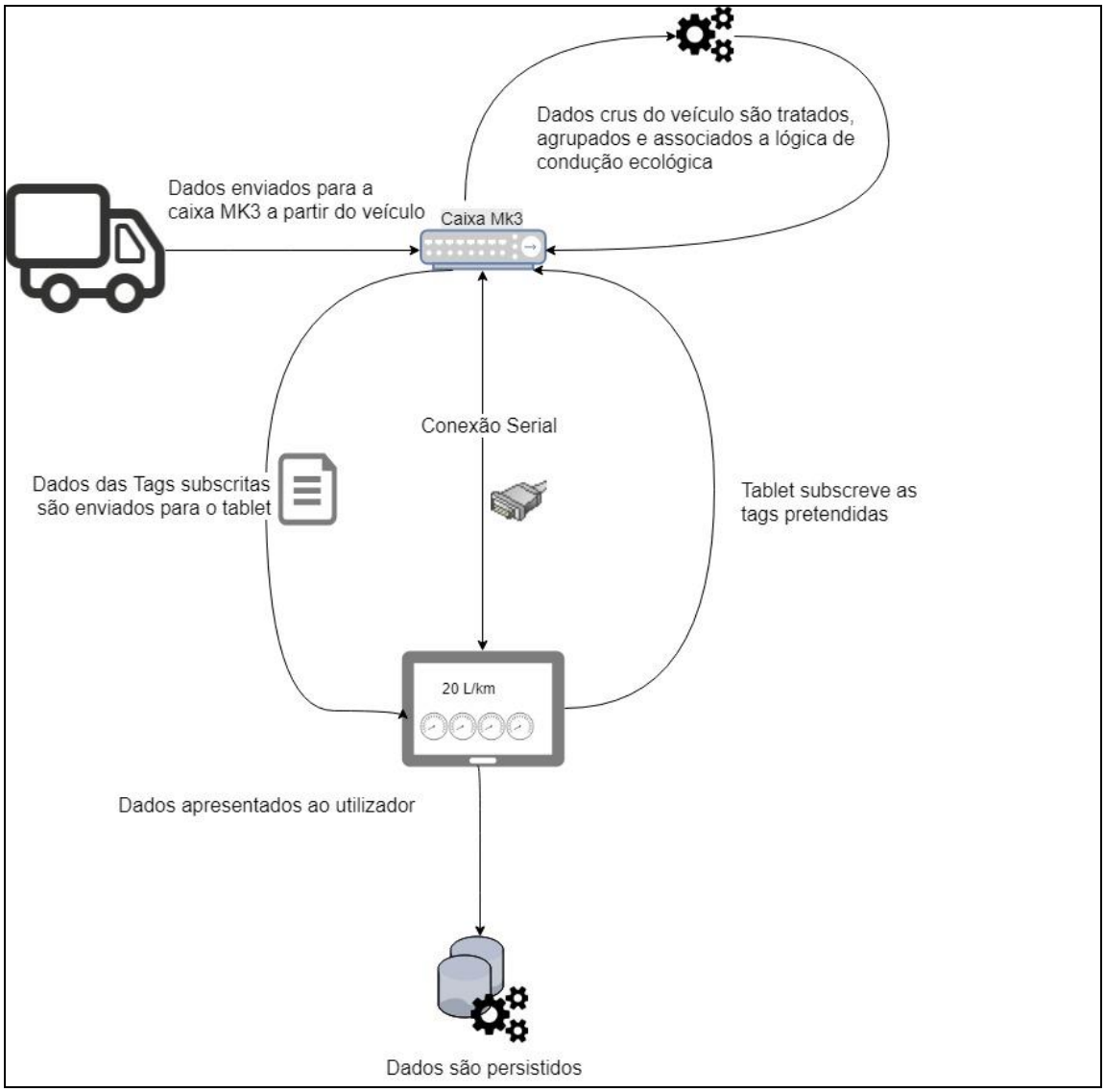


Figura 8 – Mapa arquitetural do projeto GreenDriving, com forma de ligação via caixa MK3 e porta série.

4.2.1.1. Arquitetura MVVM

Na figura seguinte apresenta-se a lógica e fluxo de dados na aplicação móvel de acordo com o padrão de desenvolvimento MVVM.

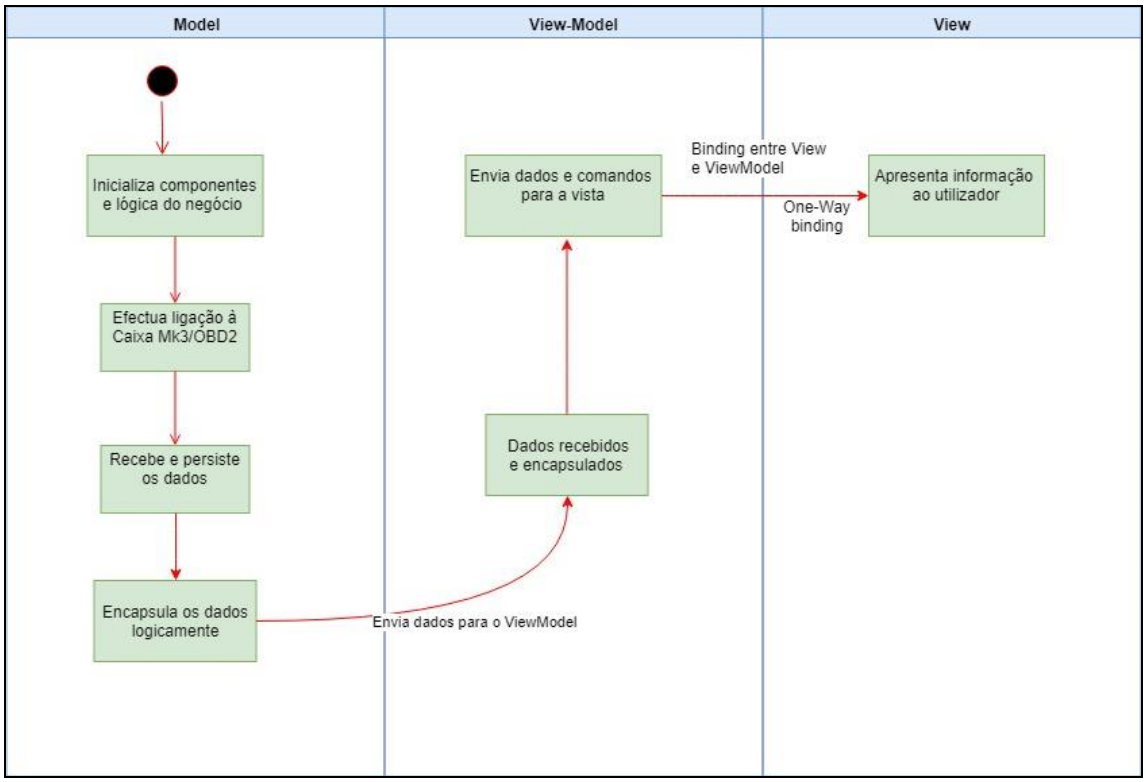


Figura 9 – Arquitetura de acordo com o padrão MVVM

Podemos observar na Figura 9 a estrutura da aplicação móvel GreenDriving de acordo com o padrão de desenvolvimento MVVM. O modelo é responsável pelos processos de lógica aplicacional, inicialização de componentes, ligação e comunicação entre a caixa MK3/OBD2 e a aplicação móvel e o tratamento dos dados enviados e recebidos.

Quando os dados estão preparados o modelo envia-os para os view-models, que são responsáveis por controlar e enviar os dados para as vistas correspondentes, para que o condutor os possa visualizar. Na secção 4.4) é apresentada informação mais detalhada relativa ao padrão de desenvolvimento MVVM.

4.2.1.2. Protocol Buffers

Os *protocol buffers* tratam-se de uma linguagem neutra na Google, sendo independentes de uma determinada plataforma e que servem principalmente como um mecanismo para serializar estruturas de dados[8].

Na documentação oficial[8], os *protocol buffers* são comparados a XML (Extensible Markup Language) sendo, no entanto, mais simples, rápidos e estruturalmente mais pequenos. Até à data os protocol buffers suportam código gerado nas seguintes linguagens de programação:

- Java
- Python
- Objective-C
- C++
- Go
- Ruby
- C#

Como podemos ver, existe uma grande quantidade de linguagens de programação que são suportadas, dando ao desenvolvedor uma maior flexibilidade na sua decisão.

No âmbito deste projeto os *protocol buffers* foram utilizados para serializar e deserializar informação entre a caixa MK3 e o dispositivo móvel *Android*. Cada vez que uma mensagem é enviada, seja a partir da caixa, seja do dispositivo móvel, essa informação é serializada e enviada, para depois ser desserializada e lida pelo dispositivo em questão.

Para começar a utilizar *protocol buffers* é necessário criar um ficheiro no formato *.proto* onde definimos a estrutura das mensagens de dados, como podemos ver na Figura 10.

```
// ------------------------------------------------------------------------
// --- CBUS Tag Messages ------------------------------------------------
message TermCbusCcvs1Msg {
        bool                    error_flag                      = 1 ;
          uint32                  wheel_speed                     = 2 ;
        bool                    parking_break_state             = 3 ;
          bool                    clutch_state                    = 4 ;
          bool                    brake_state                     = 5 ;
          bool                    cruise_control_state            = 6 ;
          bool                    power_take_off_state            = 7 ;
        uint32                  parking_break_count             = 8 ;
        uint32                  clutch_count                    = 9 ;
        uint32                  brake_count                     = 10 ;
        uint32                  cruise_control_count            = 11 ;
        uint32                  power_take_off_count            = 12 ;
}
//
message TermCbusEec2Msg {
        bool                    error_flag                      = 1 ;
          uint32                  accelerator_pedal_position      = 2 ;
        uint32                  engine_percent_load             = 3 ;
        uint32                  accelerator_count               = 4 ;
}
//
message TermCbusLfcMsg {
        bool                    error_flag                      = 1 ;
          uint32                  total_fuel_used                 = 2 ;
}
//
message TermCbusDdMsg {
        bool                    error_flag                      = 1 ;
          uint32                  fuel_level_1                    = 2 ;
          uint32                  fuel_level_2                    = 3 ;
}
//
message TermCbusEec1Msg {
        bool                    error_flag                      = 1 ;
          uint32                  actual_eng_percent_torque       = 2 ;
        uint32                  engine_speed                    = 3 ;
}
//
message TermCbusVwMsg {
        bool                    error_flag                      = 1 ;
          uint32                  axle_and_tire_location          = 2 ;
        uint32                  axle_weight                     = 3 ;
}
//
message TermCbusEhrMsg {
        bool                    error_flag                      = 1 ;
          uint32                  engine_hours                    = 2 ;
          uint32                  engine_kilo_rot                 = 3 ;
}
//
message TermCbusViMsg {
        bool                    error_flag                      = 1 ;
          string                  vehicle_identification          = 2 ;
}
```

Figura 10 – Trecho retirado do ficheiro .proto

A Figura 10 representa uma parte do ficheiro .proto relativo às mensagens de tempo real com dados do veículo. Como podemos verificar, estão apresentadas 8 mensagens relativas à *tag* CBUS. Cada uma das mensagens apresenta informação como a velocidade atual, percentagem de acelerador, quantidade de combustível existente e utilizado, entre outras.

Uma vez que o nosso ficheiro .proto está criado é necessário compilá-lo para a linguagem de programação pretendida. Para tal recorremos ao compilador disponibilizado pela *Google*. Para compilar os ficheiros é necessário correr o comando seguinte.

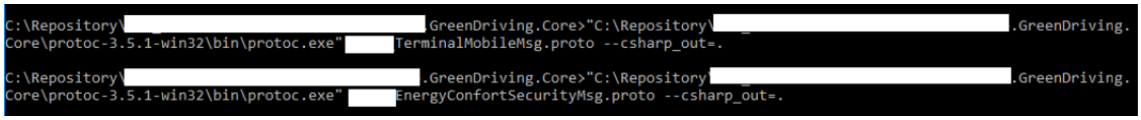


Figura 11 – Comandos para gerar as classes derivadas dos ficheiros .proto

É necessário definir as diretorias do ficheiro final, ficheiro .proto e a especificação da linguagem na qual se pretendem gerar os ficheiros de classes.

A partir do momento em que tanto a caixa MK3 como o dispositivo móvel têm as suas classes de mensagens criadas a partir do(s) mesmo(s) ficheiro(s) .proto (é possível criar e incluir referências para vários ficheiros .proto) a comunicação e transferência de informação torna-se notoriamente mais intuitiva, uma vez que basta enviar uma mensagem baseada nas classes geradas pelo ficheiro .proto para ambas as partes compreenderem a informação que recebem e integrá-las em cada lógica de modelo. Para tal é necessário associar a mensagem recebida à sua estrutura correspondente. Sempre que é recebida uma mensagem, basta identificar o seu tipo de classe de acordo com o ficheiro .proto e o objeto é preenchido automaticamente, sendo possível de seguida realizar operações sobre o mesmo.

4.2.2. OBD2 / Dispositivo móvel

Nesta outra forma de ligação recorre-se a um dispositivo ELM327 que se conecta à entrada OBD (*On Board Diagnostics*) do veículo. A partir de 2004 tornou-se obrigatório para os fabricantes de veículos incluir uma entrada OBD em todos os veículos. Isso tornou o diagnóstico de erros/alertas mais acessível, tanto para mecânicos como para utilizadores que estejam interessados em adquirir um dispositivo ELM327 ou um cabo que ligue a entrada OBD a um computador. É possível aceder à informação em tempo real, como rotações do motor, carga do motor, percentagem do acelerador pressionado, consumo de combustível instantâneo, entre muitos outros valores. É também possível aceder a alertas caso se encontre algo de errado no veículo. Todos estes dados são importantes na análise efetuada de eficiência energética.

Cada tipo de dados tem associado um determinado identificador. Esse identificador é utilizado para aceder ao valor instantâneo correspondente. Assim sendo, sempre que se deseja aceder a um valor envia-se o seu identificador à ECU do veículo que por sua vez devolve a informação associada.

O dispositivo móvel conecta-se ao dispositivo ELM327 através de uma ligação Bluetooth.

4.2.2.1. Conexão Bluetooth

Como foi dito na secção 4.2.2 a comunicação entre o módulo OBD2 e o dispositivo móvel é efetuada recorrendo a *sockets Bluetooth*. Para tal pode recorrer-se à opção de emparelhamento disponibilizada pelo sistema operativo em questão (*Android* ou *iOS*).

A Figura 12 mostra uma possível sequência de instruções de *Java* do SDK do *Android* para se efetuar esta ligação através de uma aplicação móvel.

```
BluetoothAdapter btAdapter = BluetoothAdapter.getDefaultAdapter();

BluetoothDevice device = btAdapter.getRemoteDevice(deviceAddress);

UUID uuid = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");

BluetoothSocket socket =
device.createInsecureRfcommSocketToServiceRecord(uuid);

socket.connect();
```

Figura 12 – Exemplo de código de conexão entre o dispositivo móvel e o ELM

Depois de se criar a coneção do *socket Bluetooth* podemos começar a comunicar com a ECU do veículo.

Para receber informação do veículo é necessário requisitá-la ao mesmo tempo. Para tal, cada tipo de dados têm um determinado código associado. Ao enviar um determinado código à ECU do veículo esta responde com a informação associada, neste caso o dispositivo ELM serve de *gateway* entre a ECU do veículo e a aplicação móvel/dispositivo móvel.

O primeiro código que se deve enviar representa qual o protocolo de comunicação usado pela ECU do veículo. Dependendo da marca do veículo em questão, os protocolos de comunicação podem variar, para tal recorreu-se à documentação do site "OUTILS OBD FACILE" [9] que lista grande parte dos veículos existentes, a sua marca, modelo e ano, (entre outros aspetos) e o protocolo correspondente usado pela ECU do veículo em questão. De seguida é necessário saber qual o código que corresponde a cada protocolo existente, para descobrir qual o código a usar. Na Tabela 3 são apresentados todos os protocolos existentes e o respetivo código, de acordo com a informação disponibiliza pelo website "obdtester" [10].

Tabela 3 – Correspondência entre o código e o protocolo da ECU

Código	Protocolo
ATSP0	Automatic protocol detection
ATSP1	SAE J1850 PWM (41.6 kbaud)
ATSP2	SAE J1850 VPW (10.4 kbaud)
ATSP3	ISO 9141-2 (5 baud init, 10.4 kbaud)
ATSP4	ISO 14230-4 KWP (5 baud init, 10.4 kbaud)
ATSP5	ISO 14230-4 KWP (fast init, 10.4 kbaud)
ATSP6	ISO 15765-4 CAN (11 bit ID, 500 kbaud)
ATSP7	ISO 15765-4 CAN (29 bit ID, 500 kbaud)
ATSP8	ISO 15765-4 CAN (11 bit ID, 250 kbaud) - used mainly on utility vehicles and Volvo
ATSP9	ISO 15765-4 CAN (29 bit ID, 250 kbaud) - used mainly on utility vehicles and Volvo

Depois de se enviar o código corresponde ao protocolo da ECU, e partindo do suposto que se enviou o código correto e se o dispositivo ELM suporta o protocolo em questão, dá-se por finalizada a fase de inicialização da comunicação e pode-se começar a enviar códigos correspondentes aos valores desejados. É necessário transformar as *strings* que representam os códigos em *arrays* de *bytes*, uma vez que é neste formato que as ECU's comunicam, tanto no processo de envio, como no processo de recolha da informação.

Cada código tem associado uma fórmula para tradução da informação para dados legíveis para o utilizador. Ou seja, mesmo depois de se receber a informação, é necessário tratá-la para esta ser percetível.

É também possível aferir se existem mensagens de erro detetadas no veículo através do comando “07”. A informação proveniente deste código apresenta-se também codificada. Foi criado um dicionário de dados com todos os códigos apresentados no website “totalcardiagnostics” [11]. Depois de receber a informação referente às mensagens de erro o dicionário é percorrido e é preenchida uma lista com todos os erros que foram associados à descrição de acordo com a informação do dicionário.

4.3. Xamarin

Xamarin trata-se de um plataforma híbrida de desenvolvimento de software e também de uma empresa de software fundada em São Francisco, California, em Maio de 2011 [12]. Mais tarde, em 2016, a empresa foi comprada pela Microsoft [13].

Na realização deste projeto recorreu-se À plataforma Xamarin (integrada com Visual Studio). Esta plataforma híbrida permite, tendo por base uma única base de código fonte, compilar aplicações móveis para *Android*, iOS e *Windows*.

O principal motivo para a utilização deste ambiente de desenvolvimento trata-se da possibilidade de criação de aplicações *cross platform*. Isto significa que é possível desenvolver a aplicação para dispositivos *Android* e iOS sem que seja necessário desenvolver as aplicações em separado. Neste ambiente *cross platform* as aplicações partilham o código base (core) da aplicação escrito na linguagem de programação C# (C Sharp). Isto significa que o modelo/camada lógica da aplicação é independente da plataforma móvel em questão.

Isto permite reduzir os custos de desenvolver a mesma aplicação para várias plataformas, uma vez que reduz drasticamente a repetição de código. Além disto, “obriga” a uma divisão bem pensada das camadas aplicacionais, uma vez que, de modo a otimizar ao máximo a aplicação, pretende-se abstrair ao máximo todo o código que possa ser partilhado, o que normalmente engloba todo o modelo de negócio e lógica da aplicação. Idealmente a única parte da aplicação que não seria partilhada seriam os componentes das vistas e componentes independentes e próprias de cada plataforma.

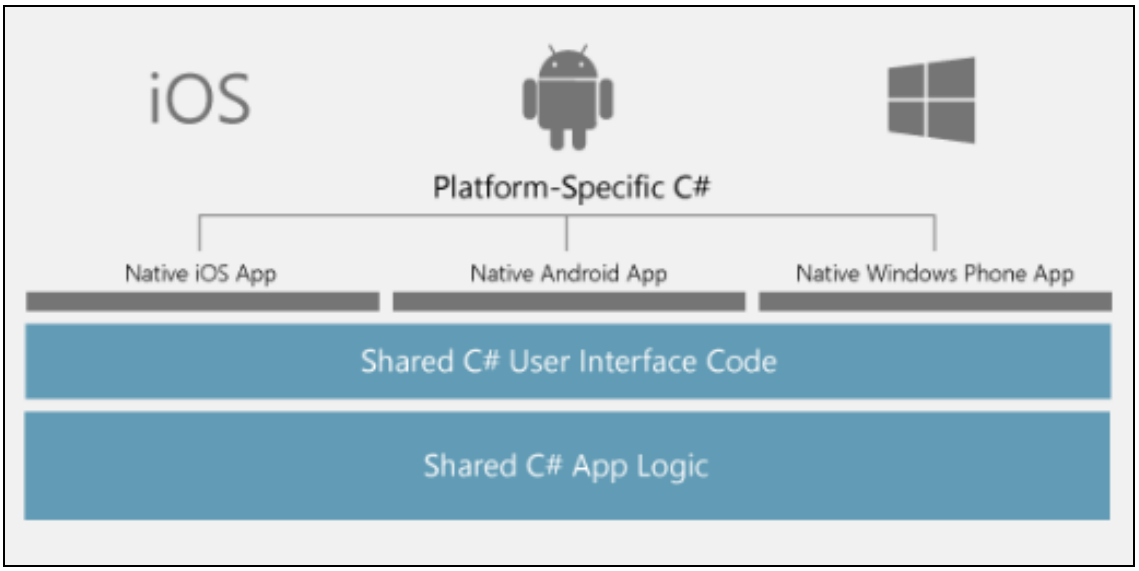


Figura 13 – Esquema representativo de desenvolvimento em Xamarin [14]

4.4. MVVM (Model-View-ViewModel)

Os padrões de desenvolvimento definem-se como uma solução geral reutilizável que serve de base para o desenvolvimento aplicacional. São definidos padrões de desenho e boas práticas que ajudam na resolução de problemas comuns no desenho de sistemas aplicacionais.

Existem vários padrões de desenvolvimento, cada um com os seus benefícios e desvantagens, uma vez que não existe uma solução perfeita para todos os problemas.

Um dos aspetos importantes que devem ser considerados na leitura desde documento é que o padrão utilizado na realização deste projeto indica propósitos específicos optados pela entidade de acolhimento, não se tratando de um padrão superior ou inferior a outros existentes, mas que simplesmente se considerou a melhor opção neste caso, ou de acordo com as políticas e interesses de desenvolvimento da empresa.

O padrão de desenvolvimento utilizado na realização deste projeto denomina-se MVVM (Model-View-ViewModel)[15].

Podemos observar na secção 4.2.1.1) a arquitetura da aplicação de acordo com este padrão de desenvolvimento, indicando como a aplicação é estruturada. De seguida é apresentada a estrutura do padrão MVVM mais detalhadamente.

Como o próprio nome indica, este padrão divide a aplicação em 3 principais categorias:

- **Modelo** – Representa o modelo de domínio da aplicação, lógica de negócio e validação. Alguns exemplos incluem objetos, DTO's (Data Transfer Object), entidades entre outros. Por outras palavras o modelo representa o núcleo (core) aplicacional. Isto significa que, independentemente da plataforma móvel para a qual se estiver a desenvolver no momento, caso se desenvolva a mesma aplicação para outra plataforma, o modelo deverá ser, sempre que possível, partilhado;

- **Vista** – Representa a camada de interação entre o utilizador e a aplicação. Por outras palavras, representa a interface onde é apresentada a informação da aplicação ao utilizador. É também a interface onde o utilizador interage e acede, modifica, adiciona ou remove informação. Esta camada é independente e particular de cada plataforma móvel em questão, ou seja, existem diferenças entre vistas implementadas para Android ou para iOS por exemplo, o que significa que as camadas, num contexto de *cross platform,* requerem implementação específica para cada plataforma. Idealmente a vista não inclui lógica aplicacional;

- **View Model** – Representa a cama intermediaria entre a vista e o modelo. Por outras palavras o View Model cria um *binding* (ligação/conexão) entre as outras duas camadas (vista e modelo). Uma das principais funções do view model passa por interagir com o modelo, invocando métodos do modelo. De seguida o modelo providencia os dados do modelo de maneira a que a vista os possa usar facilmente. Se necessário o View Model pode alterar o formato dos dados de modo a que a vista os possa mais facilmente "digerir". Do mesmo modo, qualquer interação na vista por parte do utilizador (e.g. clicar num botão da vista) pode despoletar uma determinada ação no view model. O view model pode também ser responsável na definição de mudanças de estados lógicos no aspeto da vista [15];

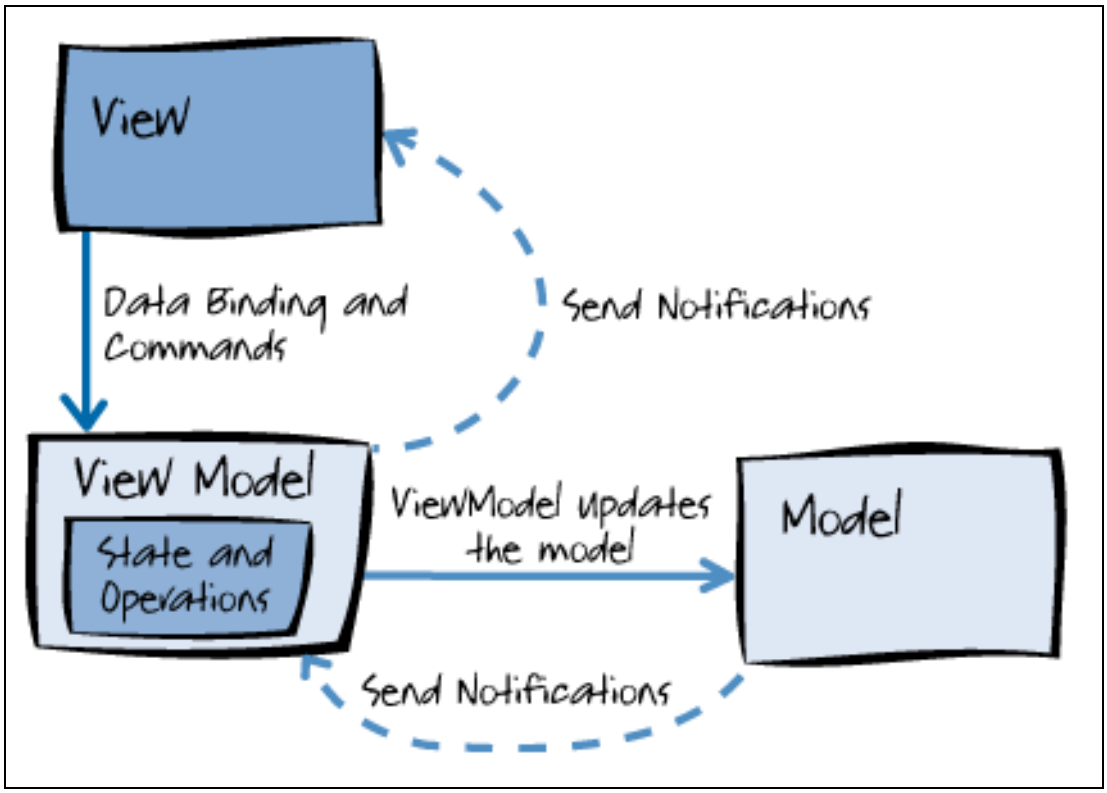


Figura 14 – Esquema arquitetural do padrão MVVM [15]

A Figura 14 representa visualmente o esquema arquitetural base de acordo com o padrão de desenvolvimento MVVM, esquema este que foi utilizado no desenvolvimento da aplicação GreenDriving como podemos visualizar na secção 4.2.1.1), Figura 9.

Este padrão de desenvolvimento permite que os componentes sejam independentes uns dos outros, permitindo[15]:

- Fácil troca de componentes.
- Implementações internas podem ser alteradas sem afetar outros componentes.
- Possibilidade de trabalhar nos diferentes componentes independentemente.
- Testes unitários isolados.

5. Implementação

Neste capítulo são especificados os aspetos relativos à implementação do projeto. São descritos os modelos de domínio de dados e lógica do projeto, bem como o modelo de arquitetura aplicacional.

São também especificados os principais algoritmos implementados para extração de códigos de erro do veículo, contextualizando-os na lógica aplicacional.

De seguida são apresentadas as evoluções de protótipos criados ao longo do tempo, explicando as modificações realizadas. Tendo em conta os protótipos criados são apresentados os *layouts* implementados desde uma fase intermédia até à fase final.

Finalmente são especificados os testes realizados, os métodos utilizados, o âmbito em que se inserem, indicando também quais as arquiteturas testadas, quais não foram e indicando os respetivos motivos.

5.1. Modelo de dados do Domínio

O modelo de domínio deste projeto é baseado em parte num módulo aplicacional de eficiência energética já existente na empresa. Esse modelo é subdivido em duas vertentes, uma delas é responsável pelos campos calculados no âmbito da lógica de negócio da empresa (1-10). A outra é responsável por reunir e encapsular a informação de tempo real relativa a sensores e indicadores de condução (11-16).

Tabela 4 – Modelo de dados do domínio

Descrição	Unidade dos Dados
1 - Tempo decorrido com dados válidos	Segundos
2 - Tempo decorrido	Segundos
3 - Distância total percorrida	Quilómetros
4 - Número de eventos de aceleração excessiva	Inteiro
5 - Número de eventos de desaceleração excessiva	Inteiro
6 - Distância total em *cruise control*	Quilómetros
7 - Tempo decorrido em *cruise control*	Segundos
8 - Tempo decorrido em aceleração	Segundos
9 - Tempo decorrido em desaceleração	Segundos
10 - Tempo em velocidade constante	Segundos
11 - Consumo de combustível instantâneo	Litros/100Km
12 - Carga do motor	%
13 - Posição do acelerador	%
14 - Posição do travão	%
15 - Rotações do motor	Inteiro
16 - Velocidade	Inteiro

5.2. Modelo de lógica de negócio

Nesta secção é apresentado o modelo de negócio aplicacional. Este modelo é responsável pela gestão das principais tarefas a executar durante todo o período de execução, dos quais:

- **Ligação à base de dados** – No arranque da aplicação a classe **BusinessManager** é responsável por inicializar e gerir a ligação à base de dados da aplicação;

- **Carregar ficheiros de configuração** – A classe **BusinessManager** é também responsável pelo carregamento dos ficheiros de configuração. Estes ficheiros são utilizados para definir que módulos da aplicação são carregados ou quais as configurações a utilizar em certas zonas da aplicação. Um destes exemplos aplica-se na opção de carregar o módulo de conexão *Bluetooth*, recorrendo ao um dispositivo ELM, ou na conexão à caixa MK3 por porta série;

- **Ligação à caixa MK3** – Relativamente à conexão à caixa MK3 existem 2 classes principais, a classe **ConsoleBusinessManager** e a classe **ConsoleCommunicationManager**. Ambas servem propósitos distintos. A primeira é instanciada pela classe **BusinessManager** sendo responsável por preparar a conexão à caixa MK3 e respetivos registos das *tags* pretendidas. A segunda (**ConsoleCommunicationManager**), é instanciada pela primeira classe e é responsável tanto pelo processo de comunicação direto à caixa MK3 (instanciando a classe gerada de *protocol buffers* referida na secção 4.2.1.2), uma vez que essa informação precisa de ser transformada de modo a que a caixa consiga interpretar os pedidos que lhe são dirigidos. É igualmente responsável por receber a informação da caixa, transformando-a em objetos e estruturas de fácil compreensão para o programador;

- **Descodificação, transformação e modulação de dados –** A classe gerada através do ficheiro .proto (referenciado na secção 4.2.1.2) é responsável pela transformação da informação entre a caixa MK3 e a aplicação GreenDriving e

vice-versa. Essa informação é então recebida na classe **ConsoleCommunicationManager** que popula os objetos necessários, enviando-os para a classe **ConsoleBusinessManager**, que por sua vez os distribui pelas zonas aplicacionais pretendidas;

- **Lançamento de eventos para os ViewModels da aplicação –** Quando a informação chega à classe **BusinessManager** através da classe **ConsoleBusinessManager** é então enviada para os ViewModels que se registaram nos eventos pretendidos, Esses ViewModels são ligados às respectivas vistas (*binding*) que, de seguida, apresentam a informação ao utilizador da aplicação;

De seguida, na Figura 15, é apresentado o diagrama de classes que é responsável pelos pontos acima referidos e a estrutura de ligação entre elas.

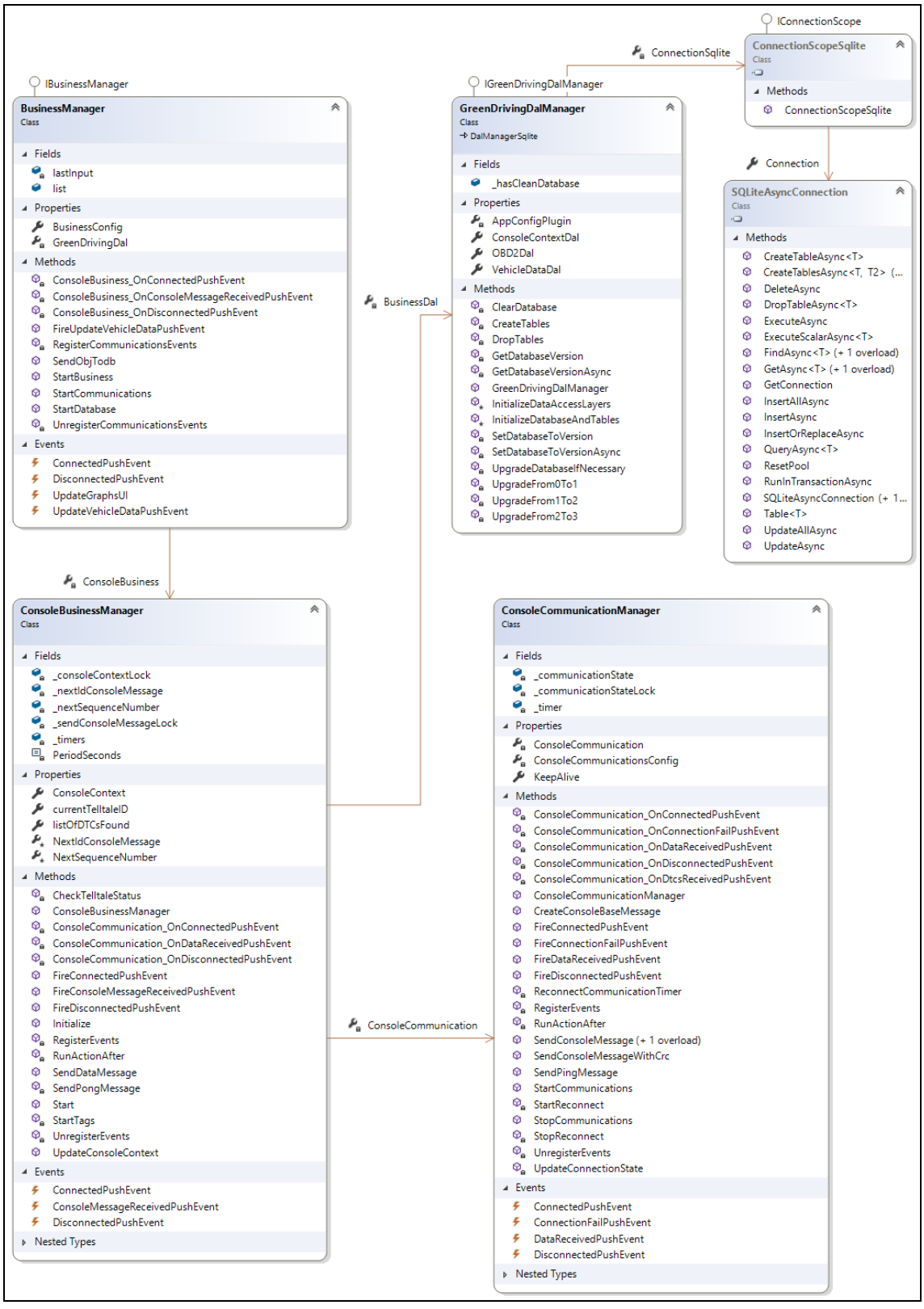


Figura 15 – Modelo de lógica de negócio

Como podemos observar na Figura 15 a classe **BusinessManager** é a classe "mãe" responsável pelo início de todo o processo de negócio aplicacional. Esta classe instancia de seguida a classe **ConsoleBusinessManager** que, como foi dito anteriormente, é responsável pela preparação do modelo de dados relativo à comunicação com a caixa MK3, instanciando a classe **ConsoleCommunicationManager** que trata diretamente dessa comunicação.

A classe **ConsoleBusinessManager** é também responsável por popular a tabela de base de dados com os dados provenientes da caixa MK3.

5.3. Modelo de ViewModels (Padrão MVVM)

Nesta secção apresenta-se o modelo de View-Models responsável por gerir os componentes do tipo *Activity* do *Android*.

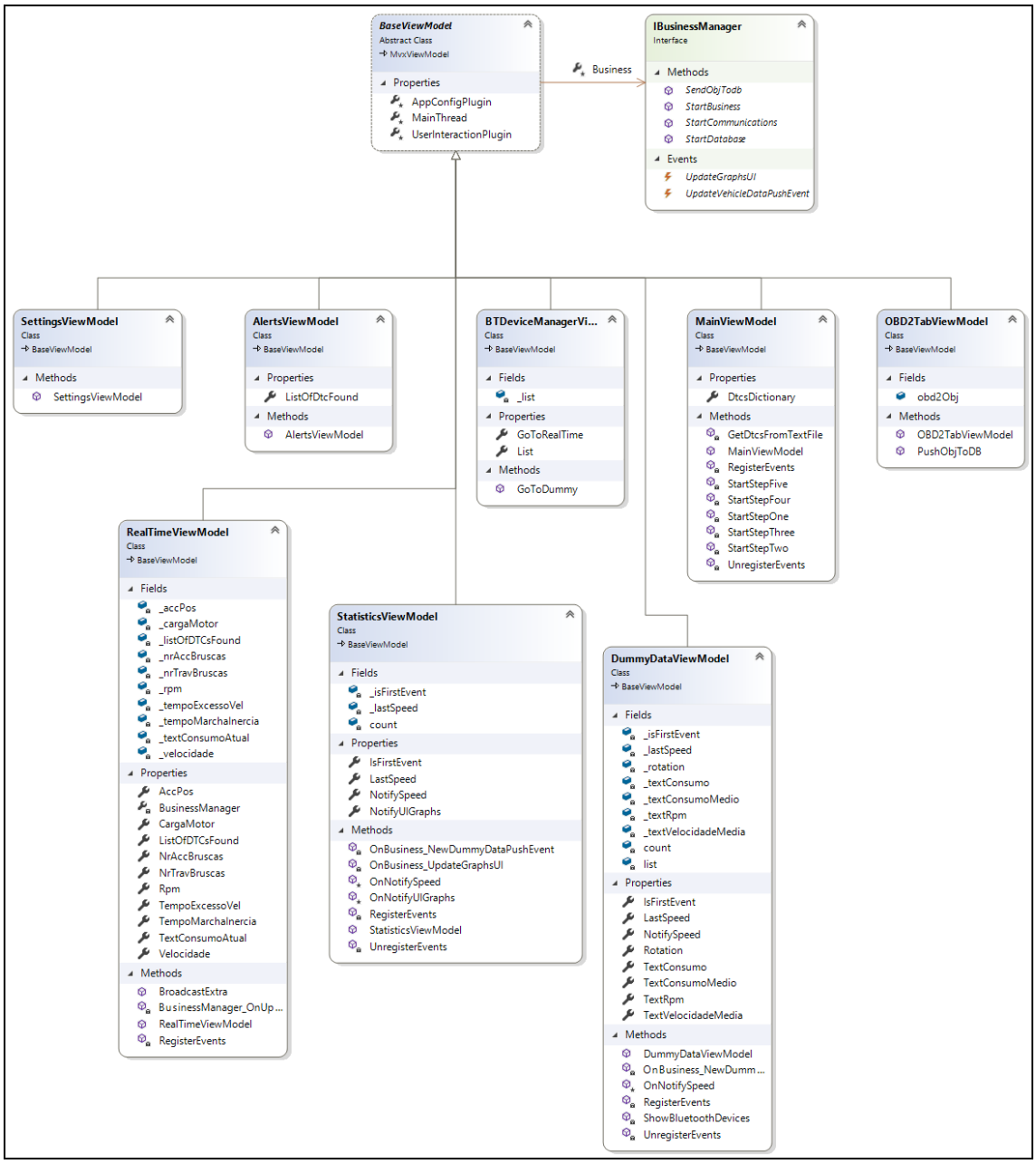


Figura 16 – Modelo de View-Models (Padrão MVVM)

A Figura 16 representa o modelo de view-models da aplicação GreenDriving. Como podemos verificar, todos os view-models estendem de um view-model base. Cada view-model controla todas as propriedades que são apresentadas nas suas respetivas vistas da aplicação GreenDriving. O view-model **RealTimeViewModel** é responsável por controlar a vista apresentada na secção 5.5.5), tratando-se da principal vista da aplicação GreenDriving. Como podemos observar, este view-model tem acesso aos eventos de dados da classe **BusinessManager** referenciada na secção 5.2). Os dados recebidos são

associados à secção de propriedades da classe **RealTimeViewModel** que, de seguida, notifica a vista com a nova informação atualizada em tempo real.

5.4. Algoritmo de extração de Telltales

Nesta secção é apresentada a lógica e implementação de um dos principais algoritmos da aplicação no que toca à transformação de dados "crus" (*raw data*) relativos aos códigos de erro, denominados de "*Telltales*" de acordo com o protocolo FMS-Standard (*Fleet Management System*)[7]. Como foi referido na secção 4.1.4, o standard FMS trata-se de um standard criado através de uma parceria entre marcas de veículos pesados. Este standard encontra-se disponível para consulta e download no respetivo site [7] em formato .pdf. No documento encontram-se descritos todos os tipos de dados relativos a informação do veículo que são disponibilizados e a forma como eles podem ser transformados para um tipo de dados legível.
A camada física rege-se de acordo com o ISO 11898 (250kb/s). A camada aplicacional encontra-se de acordo com o SAE J1939/71 e a camada de ligação de dados encontra-se de acordo com o SAE J1939/21.

1.1.16 FMS Tell Tale Status: FMS1

0x00FD7D								PGN Hex
64,893								PGN
1000 ms								Rep. Rate
Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7	Data Byte 8	Byte No
8 7 6 5 4 3 2 1	8 7 6 5 4 3 2 1	8 7 6 5 4 3 2 1	8 7 6 5 4 3 2 1	8 7 6 5 4 3 2 1	8 7 6 5 4 3 2 1	8 7 6 5 4 3 2 1	8 7 6 5 4 3 2 1	Bit No.
Telltale Block ID see table for Block ID and Telltale ID 1111 = don't care	Telltale Status 2 000 = off 001 = Cond. Red 010 = Cond. Yellow 011 = Cond. Info 100–110 = Reserved 111 = not available	Telltale Status 4 000 = off 001 = Cond. Red 010 = Cond. Yellow 011 = Cond. Info 100–110 = Reserved 111 = not available	Telltale Status 6 000 = off 001 = Cond. Red 010 = Cond. Yellow 011 = Cond. Info 100–110 = Reserved 111 = not available	Telltale Status 8 000 = off 001 = Cond. Red 010 = Cond. Yellow 011 = Cond. Info 100–110 = Reserved 111 = not available	Telltale Status 10 000 = off 001 = Cond. Red 010 = Cond. Yellow 011 = Cond. Info 100–110 = Reserved 111 = not available	Telltale Status 12 000 = off 001 = Cond. Red 010 = Cond. Yellow 011 = Cond. Info 100–110 = Reserved 111 = not available	Telltale Status 14 000 = off 001 = Cond. Red 010 = Cond. Yellow 011 = Cond. Info 100–110 = Reserved 111 = not available	Name values values values values values values
Telltale Status 1 000 = off 001 = Cond. Red 010 = Cond. Yellow 011 = Cond. Info 100–110 = Reserved 111 = not available	Telltale Status 3 000 = off 001 = Cond. Red 010 = Cond. Yellow 011 = Cond. Info 100–110 = Reserved 111 = not available	Telltale Status 5 000 = off 001 = Cond. Red 010 = Cond. Yellow 011 = Cond. Info 100–110 = Reserved 111 = not available	Telltale Status 7 000 = off 001 = Cond. Red 010 = Cond. Yellow 011 = Cond. Info 100–110 = Reserved 111 = not available	Telltale Status 9 000 = off 001 = Cond. Red 010 = Cond. Yellow 011 = Cond. Info 100–110 = Reserved 111 = not available	Telltale Status 11 000 = off 001 = Cond. Red 010 = Cond. Yellow 011 = Cond. Info 100–110 = Reserved 111 = not available	Telltale Status 13 000 = off 001 = Cond. Red 010 = Cond. Yellow 011 = Cond. Info 100–110 = Reserved 111 = not available	Telltale Status 15 000 = off 001 = Cond. Red 010 = Cond. Yellow 011 = Cond. Info 100–110 = Reserved 111 = not available	Name values values values values values values
Not defined (set to "1")	Not defined (set to "1")	Not defined (set to "1")	Not defined (set to "1")	Not defined (set to "1")	Not defined (set to "1")	Not defined (set to "1")	Not defined (set to "1")	Name values values values values values

Figura 17 - FMS Telltale Status [7]

A Figura 17 representa 1 de 5 blocos de 8 *bytes* de informação que compõem a mensagem relativa aos *telltales* recebida. A descrição detalhada de todos os *telltales* encontra-se disponível em anexo na Figura 25 e Figura 26.

Cada *telltale* encontra-se numa ordem fixa e pré-determinada de modo a facilitar a associação da informação recebida ao *telltale* correspondente.

Como podemos ver na figura, todos os blocos disponibilizam o seu identificador, seguido de 15 *telltales*, existindo um total de 40 *telltales* diferentes. O primeiro byte de cada bloco é tratado de maneira diferente, uma vez que apresenta informação acerca do identificador e o primeiro *telltale*. Os bytes restantes de cada bloco são tratados de igual modo apresentado cada um informação relativa a 2 *telltales*.

Cada *telltale* poderá ter apenas um dos seguintes estados:

- **Off** – Representado por "000": significa que não existem alarmes para o telltale corresponde.
- **Condition Red** – Representado por "001": significa que se detetou um alarme de nível vermelho.
- **Condition Yellow** – Representado por "010": significa que se detetou um alarme de nível amarelo.
- **Condition Info** – Representado por "011": significa que se detetou um alarme de nível informativo.
- **Reserved** – Representado por "100-110": significa que este telltale é reservado, logo não existe informação disponível ao público relativo à sua definição.
- **Not available** – Representado por "111": significa que não a informação relativa a este telltale não se encontra disponível.

Depois de percorrida a mensagem são registados todos os *telltales* que tenham registado um valor diferente de "*Off*", "*Reserved*" ou "*Not available*".

De seguida é apresentada a logica algorítmica criada para a conversão dos dados recebidos:

Lógica algorítmica:

- 1-Verificar se a mensagem (*string* de *bits*) tem um tamanho igual a 40 *bytes* (320 *bits*): (8 (*bytes*)* 5 (*block ID's*)) (Existem 5 *Block ID's*).

- 2- Dividir 40 por 5 = 8 (dividir a *string* por cada *block ID* disponível)

- 3- Para cada byte de cada conjunto de 8, o primeiro *bit* é sempre = 1 (ignora-se o primeiro bit):

- 3.1- O primeiro *byte* é tratado de maneira diferente
- 3.1.1- Os últimos 4 *bits* representam o *block ID* correspondente
- 3.1.2- Os *bits* 2,3,4 representam o primeiro *telltale*

- 3.2- Os restantes 7 *bytes* do *block ID* são tratados todos iguais:
- 3.2.1- 2,3,4 *bits*: status do *telltale* N
- 3.2.1- 6,7,8 *bits*: status do *telltale* N+1

A Figura 18 representa o pedaço de código criado para a leitura de *telltales* descrita anteriormente.

```
try
{
    int currentTelltaleID;
    if (a.Length == 320)
    {
        List<string> list = new List<string>();
        List<Tuple<string, string>> listOfDtcFound = new List<Tuple<string, string>>();
        int offSet = 0;
        currentTelltaleID = 1;

        for (int i = 0; i < a.Count() / 64; i++)
        {
            list.Add(a.Substring(offSet, 64));

            var id = list[i].Substring(3, 4);
            var telltale1 = list[i].Substring(1, 3);

            if (!telltale1.Equals("000") && !telltale1.Equals("111"))
            {
                var status = GetTelltaleStatusInfoDictionary(telltale1);
                var info = GetTelltateInfoDictionary(currentTelltaleID);

                if (status != null && info.Description != null)
                { /*add à lista de DTC's */
                    listOfDtcFound.Add(new Tuple<string, string>(status, info.Description));
                }
            }
            //check value
            currentTelltaleID++;
            int index = 8;

            for (int j = 0; j < 8 - 1; j++)
            {
                var tellinfo = list[i].Substring(index, 8);
                var first = tellinfo.Substring(1, 3);

                if (!first.Equals("000") && !first.Equals("111"))
                {
                    var status = GetTelltaleStatusInfoDictionary(first);
                    var info = GetTelltateInfoDictionary(currentTelltaleID);

                    if (status != null && info.Description != null)
                    { /*add à lista de DTC's */
                        listOfDtcFound.Add(new Tuple<string, string>(status, info.Description));
                    }
                }
                //check value
                currentTelltaleID++;
                var second = tellinfo.Substring(5, 3);

                if (!second.Equals("000") && !second.Equals("111"))
                {
                    var status = GetTelltaleStatusInfoDictionary(second);
                    var info = GetTelltateInfoDictionary(currentTelltaleID);

                    if (status != null && info.Description != null)
                    { /*add à lista de DTC's */
                        listOfDtcFound.Add(new Tuple<string, string>(status, info.Description));
                    }
                }
                //check value
                currentTelltaleID++;
                index += 8;
            }
            offSet += 64;
        }
        return listOfDtcFound;
    }
    else
    {
        return null;
    }
}
catch (Exception) { return null; }
```

Figura 18 - Código de leitura da mensagem de telltales

5.5. Protótipos

Durante a realização deste projeto foram desenvolvidos vários protótipos em várias fases do desenvolvimento da aplicação, isto deveu-se a várias mudanças, quer no objetivo pretendido, quer devido a questões de otimizações de interface.

Nas secções seguintes são apresentadas as evoluções da interface da aplicação, indicando o motivo das mudanças ao longo do tempo.

5.5.1. Protótipos Iniciais

Inicialmente foram efetuados protótipos da aplicação com base num ecrã de um *smartphone* na vertical, uma vez que nessa altura o projeto se tratava de um *proof of concept* e iria sofrer alterações visuais no futuro.

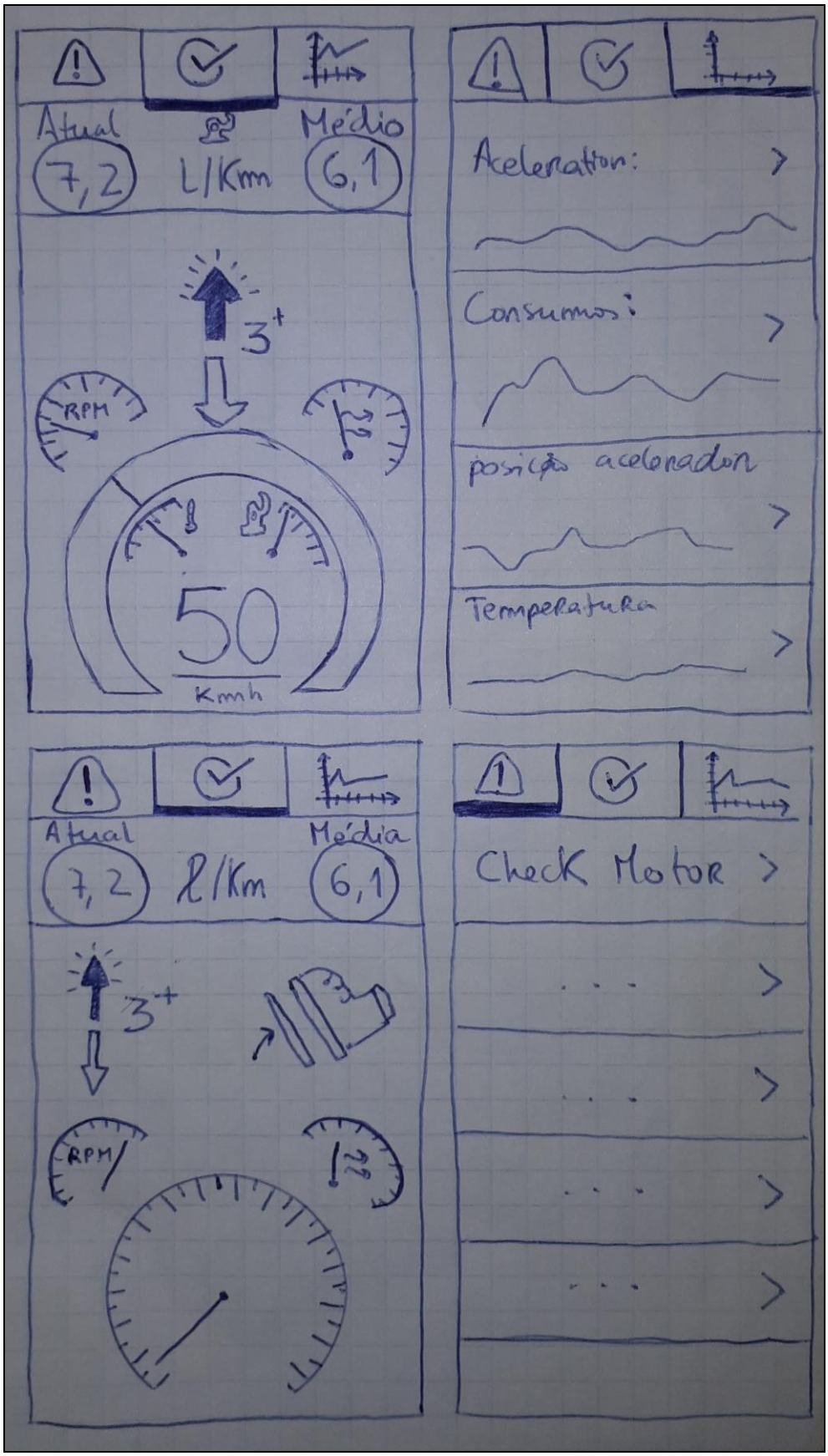


Figura 19 - Protótipos iniciais

A Figura 19 representa os primeiros protótipos criados nos primeiros dias de desenho de esboços da aplicação. Elementos destes protótipos foram utilizados para criar os protótipos funcionais dos ecrãs de alertas e estatísticas presentes na secção 5.5.2).

5.5.2. Protótipos Intermédios

Nesta secção é apresentado um protótipo funcional criado num momento inicial do desenvolvimento.

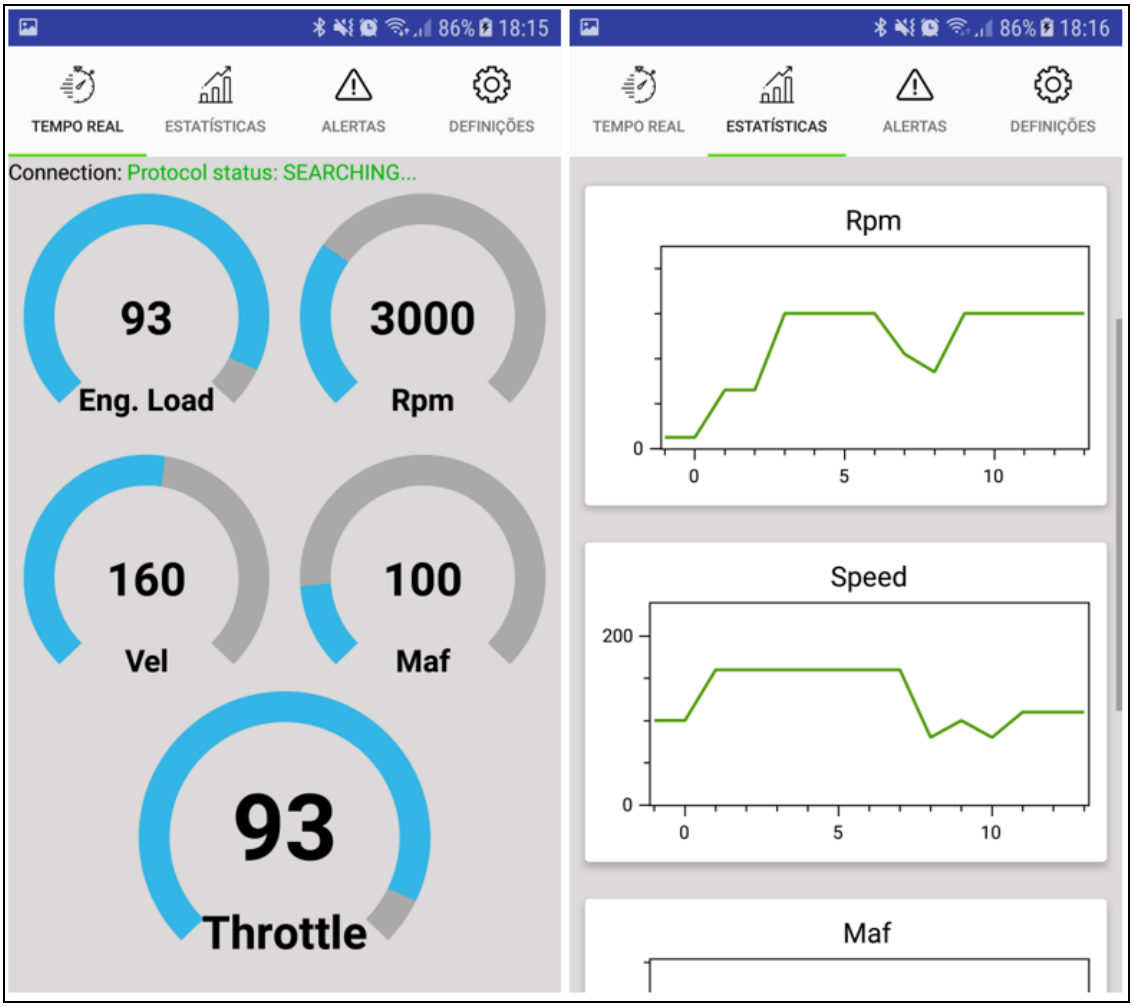


Figura 20 - Protótipos intermédios funcionais dos ecrãs de tempo real e estatísticas

Na Figura 20 podemos observar os protótipos funcionais dos ecrãs de tempo real e estatísticas. O ecrã da esquerda apresenta dados relativos à carga do motor, rotações, velocidade, MAF (*Mass Air Flow*) e posição do acelerador. O ecrã da direita apresenta a persistência dos dados recebidos em tempo real disponibilizados em forma de gráfico ao longo do tempo, sendo que os gráficos são atualizados em tempo real.

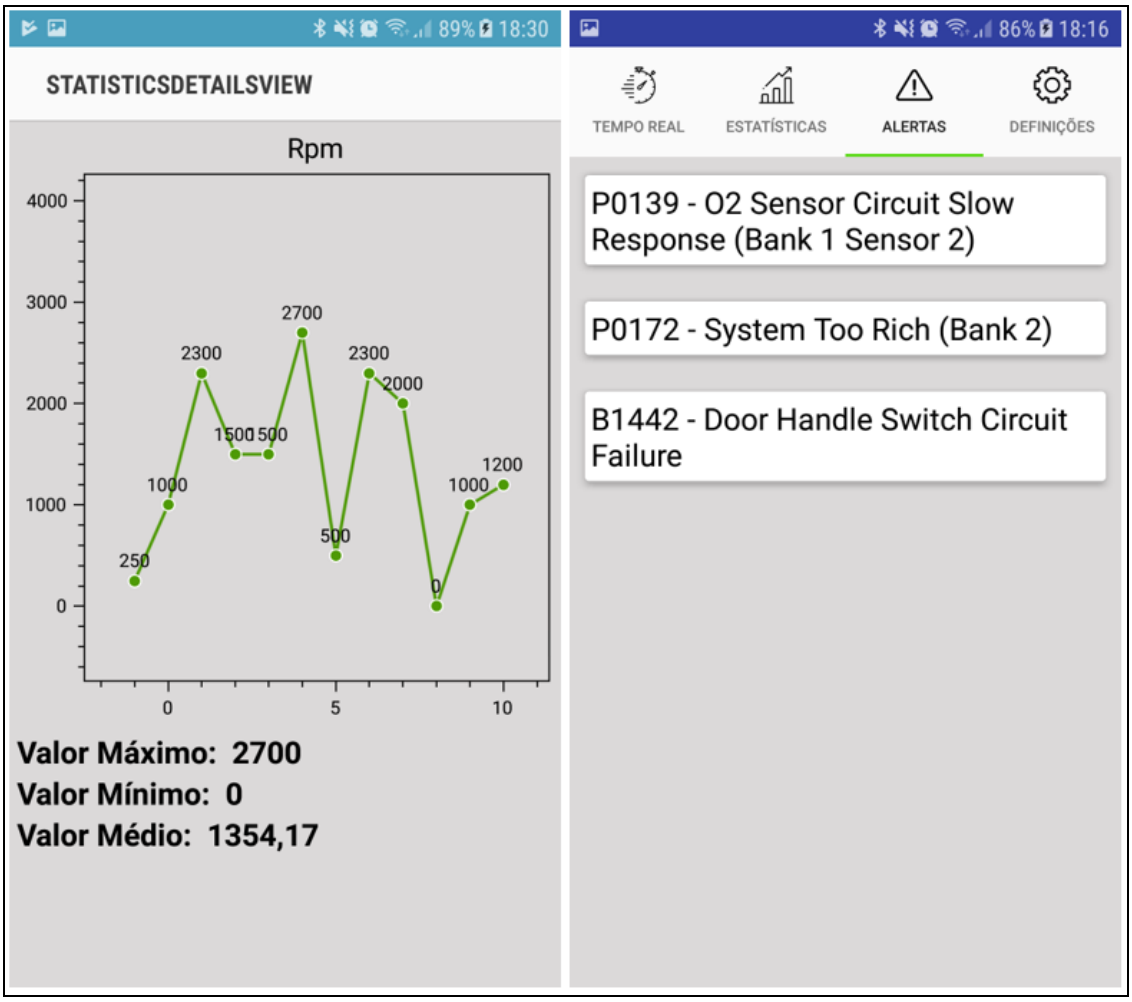


Figura 21 - Protótipos intermédios funcionais dos ecrãs de detalhes de estatísticas e ecrã de alertas

A Figura 21 representa a continuação dos protótipos intermédios funcionais. À esquerda podemos observar uma vista de detalhes de uma determinada estatística selecionada pelo utilizador. Esta vista inclui um gráfico interativo para melhor compreensão dos valores obtidos.

À direita da figura podemos observar o ecrã de alertas. Este ecrã apresenta novos alertas caso estes sejam detetados durante a condução, notificando assim o condutor já exista algum problema detetado no veículo.

5.5.3. Protótipos Finais

Num momento posterior aos primeiros protótipos efetuados foi decidido que a aplicação teria como principal implementação um ecrã de *tablet* na horizontal. Deste modo foi necessário repensar toda a interface da aplicação, aproveitar o espaço adicional e redefinir quais os elementos informativos que mereciam mais ou menos destaque. Foi efetuada uma reunião para definir estes aspetos, o resultado pode ser visualizado na Figura 22.

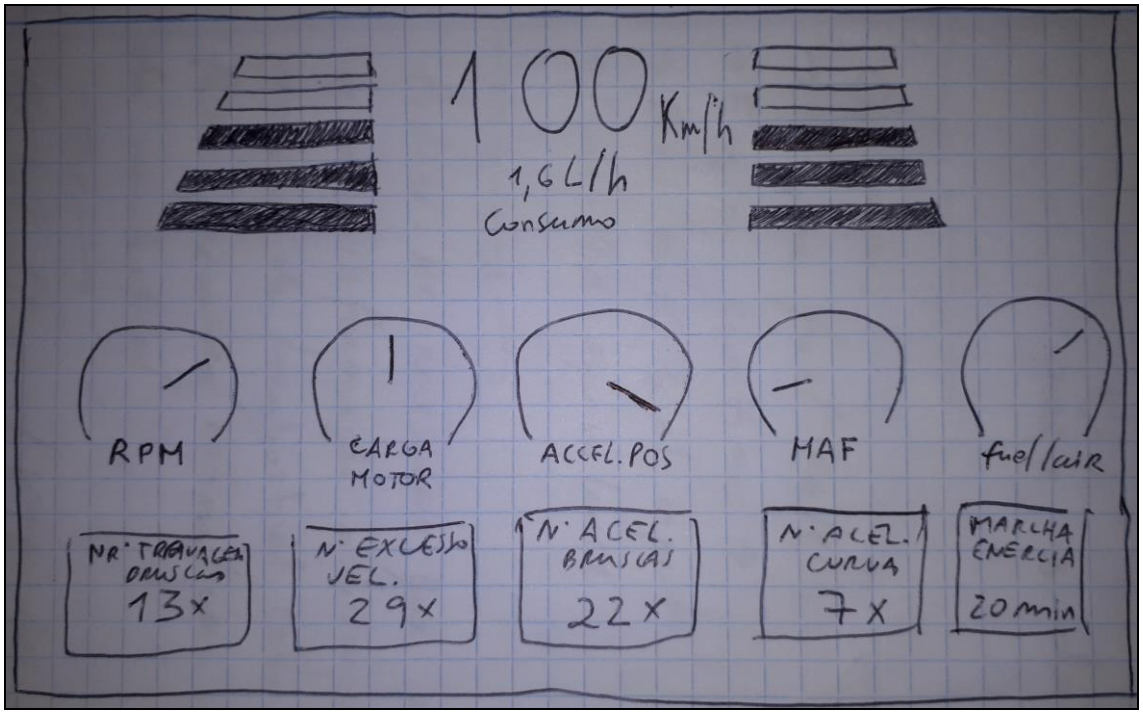


Figura 22 – Protótipo do ecrã para tablet

5.5.4. Layout Intermédio

Inicialmente, como se pode visualizar nas secções de protótipos iniciais e intermédios, foi definido que a aplicação iria ter vários ecrãs, separando informação de tempo real, estatísticas e alertas.

No entanto, depois de algumas reuniões e discussão, foi definido que se iria juntar toda a informação num único ecrã. Uma das principais razões deve-se ao facto de se entender que era importante não distrair o condutor com gestos no ecrã para aceder a determinada informação.

Outra das razões deve-se ao facto de um dos objetivos passar por integrar a aplicação noutra já existente dentro do leque de aplicações da empresa. Deste modo, para simplificar todo o ecossistema da aplicação "Mãe". Nesta altura foi feito outro *mockup*, que se pode visualizar na Figura 23.

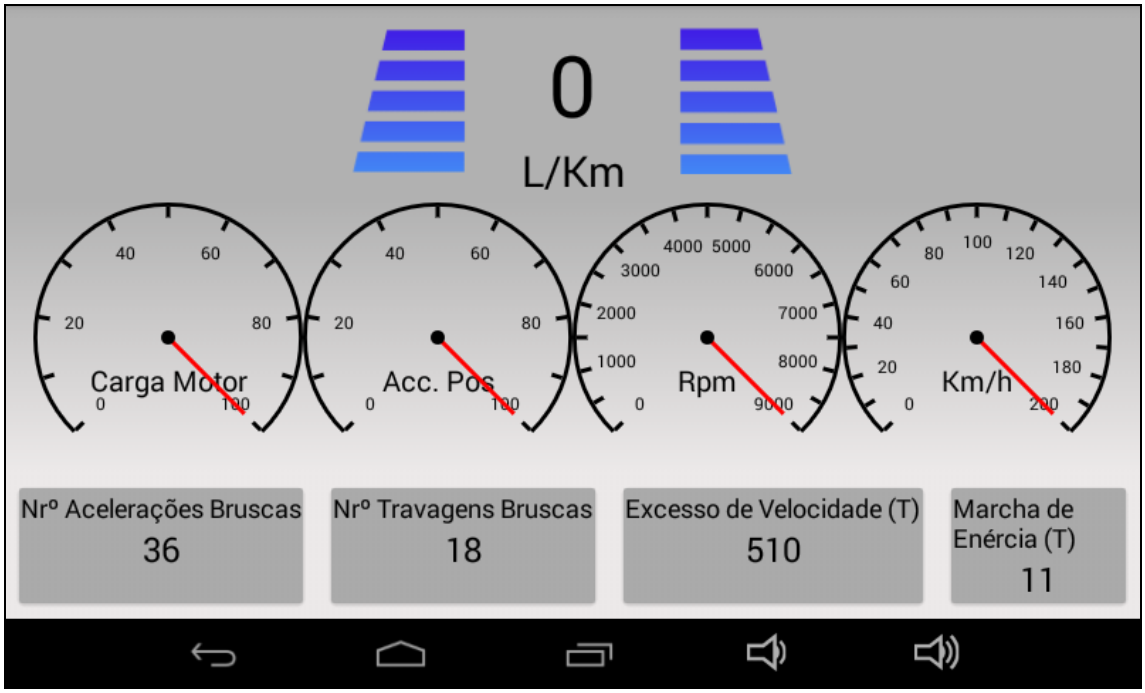


Figura 23 - Layout Intermédio

5.5.5.Layout Final

Nesta secção é apresentado o *design* do ecrã final da aplicação GreenDriving. De seguida é apresentada uma descrição do ecrã, os elementos que se inserem no mesmo, bem como uma descrição dos mesmos.

Figura 24 - Layout Final

Como podemos observar na Figura 24, o ecrã final é muito semelhante ao *layout* intermédio representado na Figura 23, com a adição dos eventuais alertas a serem apresentados na parte superior da aplicação. Ao carregar nestes alertas podemos visualizar um *popup* com uma descrição mais detalhada.

Por baixo da secção dos alertas, é indicado o consumo de combustível instantâneo atual, aliado a uma barra indicativa desse mesmo consumo, com o objetivo de alertar

visualmente o condutor sem que este tenha de olhar fixamente para o ecrã e deste modo evitando perdas de atenção desnecessárias.

Na parte intermédia do ecrã são apresentados os medidores de tempo real, estes medidores podem apresentar valores relativos a vários tipos de dados, de acordo com o que o condutor desejar visualizar.

Na parte inferior da aplicação temos os contadores relativos a ações em que o condutor incorre num tipo de condução pouco eficiente, esses contadores são depois calculados gerando uma determinada pontuação relativa à condução.

5.5.6. Testes Realizados

Nesta secção são apresentados os testes de usabilidade efetuados durante o desenvolvimento da aplicação GreenDriving.

Os testes de seguida enumerados foram efetuados na versão da aplicação com a arquitetura **Caixa MK3-Tablet**, uma vez que foi a principal arquitetura desenvolvida no decorrer do estágio. O principal motivo para esta escolha deveu-se ao facto de se inserir de melhor no contexto empresarial em questão. Deste modo, trata-se da arquitetura mais extensamente testada.

Para simular os dados provenientes do veículo foi utilizado um controlador criado na empresa para o efeito, que simula a geração e transmissão destes dados. Deste modo foi possível testar que o código recetor e de descodificação dos dados funciona como esperado.

Durante o período de desenvolvimento da aplicação foram feitos vários testes práticos, dos quais se destacam:

- **Desconexão e reconexão dos vários cabos do esquema arquitetural** – Um dos principais testes a implementar durante a implementação da aplicação passou por analisar o comportamento da aplicação se o utilizador desligasse os cabos necessário para todo o sistema funcionar corretamente. Para tal houve uma preparação prévia do código para conseguir suportar tal comportamento. Foi

também testado o caso da aplicação se inicializar sem qualquer tipo de conexão fazendo-a passado algum tempo e verificando se o fluxo aplicacional decorre normalmente. Depois de se repetir este processo inúmeras vezes e no processo contínuo de desenvolvimento da aplicação inferiu-se que a aplicação estava preparada para esses casos;

- **Dados inválidos provenientes do veículo** – Existe sempre a possibilidade de existir alguma avaria, quer no veículo, nos cabos de ligação, caixa MK3 entre outros. Uma vez que a camada de comunicação entre a caixa MK3 e a aplicação se rege de acordo com o mesmo ficheiro protobuf, referenciado em capítulos anteriores, qualquer mensagem que não se encontra de acordo com os parâmetros definidos será automaticamente descartada. Deste modo se a informação vier corrompida a aplicação não irá abaixo inesperadamente.

- **Comportamento da aplicação em *background*** – O modelo da aplicação corre em *threads* independentes da vista, deste modo a aplicação corre normalmente em *background*, continuando a processar os dados recebidos mesmo que o utilizador a mantenha em segundo plano, enquanto a gestão de memória do dispositivo móvel o permitir.

Todos os testes acima descritos foram realizados em todas as fases de desenvolvimento aplicacional, garantindo assim a consistência da aplicação.

Sempre que se desenvolveu uma nova funcionalidade a mesma foi testada por vários programadores da empresa, de modo a tentar ampliar o volume de testes realizados.

No decorrer do estágio não foram efetuados testes de aceitação uma vez que a aplicação ainda se encontra num estado pré-testes reais. Isto deveu-se a vários fatores, dois quais alguns atrasados na chegada de recursos necessários ao desenvolvimento da aplicação, bem como a necessidade de desenvolver outros projetos da empresa.

No entanto, entende-se que o módulo arquitetural entre caixa simulador-MK3-aplicação esteja pronto para começar a realizar os primeiros testes “reais” em veículos. Em relação à arquitetura entre o microcontrolador ELM327 e a aplicação, foram realizados testes práticos em 2 veículos particulares, das marcas SEAT e VW, com resultados positivos. Os dados provenientes dos veículos foram corretamente recebidos, transformados e apresentados na aplicação.

6. Conclusões

A realização deste projeto permitiu desenvolver competências em várias áreas de desenvolvimento de *software*, quer na integração de sistemas já existentes e que foram ampliados, integrando novas funcionalidades, quer na exploração de comunicação de dados entre veículos e aplicações móveis.

Apesar do desenvolvimento deste projeto ter como base uma metodologia de desenvolvimento ágil, definir bem o ambiente de desenvolvimento e o IDE no qual o código iria ser desenvolvido numa fase inicial tornou-se importante para evitar eventuais mudanças drásticas. Estes aspetos foram também de encontro com a metodologia de desenvolvimento utilizado na empresa, sendo que desta maneira a aplicação foi facilmente inserida no leque aplicacional já existente. Outro aspeto positivo proveniente deste método foi a fácil utilização de ferramentas e código partilhado e comum que foi assim incluído neste projeto.

O desenvolvimento da aplicação GreenDriving permitiu adquirir conhecimentos de comunicação e integração de sistemas entre veículos e aplicações móveis. A aplicação GreenDriving é capaz de comunicar com veículos através de diferentes protocolos e arquiteturas de comunicação, apresentando informação em tempo real relativo ao modo de condução do condutor. A informação apresentada pela aplicação é subdividida em alertas, indicadores de tempo real e contadores calculados.

Os alertas apresentados na aplicação estão de acordo com a informação detalhada no protocolo FMS criado por uma associação de várias marcas que manufaturam veículos de transporte pesados. Os indicadores de tempo real apresentam informação de sensores existentes no veículo que estão diretamente relacionados com o consumo de combustível. Adicionalmente são apresentados aos condutores campos calculados que registam ações em que se incorre num tipo de condução pouco eficiente, nomeadamente, acelerações e travagens bruscas, acelerações em curva e marcha de inercia.

Optou-se pela criação de um design simples que apresentasse ao condutor todos os parâmetros considerados de maior importância, sem que houvesse necessidade de interações físicas entre o condutor e a aplicação GreenDriving durante o período de condução.

Durante o desenvolvimento da aplicação foram adquiridos e desenvolvidos conhecimentos na área da eficiência energética, gestão de frotas de veículos, lógicas arquiteturais e regras de negócio que permitiram criar uma aplicação robusta e capaz de se adaptar a eventuais modificações. Isto deveu-se a vários aspetos, desde o padrão de desenvolvimento utilizado desde o momento inicial, uma estrutura base inicial preparada para ser flexível, bem como diversos conhecimentos ensinados durante todo o período de estágio curricular.

Foram desenvolvidas aptidões interpessoais num contexto empresarial quer no processo de adaptação, quer na estruturação e discussão de objetivos. Foram melhoradas as aptidões de gestão de tempo e metas, bem como a sua importância para o bom funcionamento empresarial.

Existem várias melhorias e implementações a realizar na aplicação em trabalho futuro. Inicialmente é pretendido inserir este módulo aplicacional numa aplicação mais abrangente já existente e em desenvolvimento na empresa. Essa aplicação insere-se no âmbito de gestão de frotas e de apoio aos motoristas de determinadas empresas para as quais a Tecmic desenvolve software, sendo pretendido inserir este módulo (aplicação GreenDriving) de eficiência energética.

É também pretendido analisar os dados resultantes das viagens de frotas de veículos ao longo do tempo com o objetivo de descobrir eventuais fatores que determinem comportamentos de condução. Eventualmente pretende-se criar regras/modelos de condução que indiquem o que se deve ou não fazer quando se pretende otimizar a eficiência energética na condução, reduzindo os consumos de combustível. Para tal é necessário recolher uma quantidade significativa de dados, para garantir alguma consistência nas regras e análise proveniente desses dados.

Apesar de ter sido feito um protótipo de estatísticas com os dados de condução em tempo real, essa funcionalidade foi reservada para um período futuro. No entanto pretende-se também criar um ecrã para visualização fora de viagem em que o utilizador consegue aceder e analisar as estatísticas da sua condução, com o objetivo de ter uma

noção mais abrangente da sua avaliação, e principais aspetos a apontar, sejam eles positivos ou negativos. Eventualmente será também possível comparar modos de condução, quer entre diferentes períodos ou viagens do mesmo condutor, quer entre condutores, tudo isto aliado a uma quantidade considerável de dados obtidos.

Uma vez que na Tecmic existem módulos de gestão de viagens e itinerários, uma funcionalidade que merece ser estudada seria a diferença de eficiência energética relativa a diferentes percursos e como o relevo, trânsito, horário, entre outros aspetos poderão afetar a eficiência energética.

Finalmente pretende-se também, num momento futuro, acabar e consolidar o módulo de integração com um microcontrolador ELM327 uma vez que se deu prioridade à arquitetura com a caixa MK3, uma vez que fazia mais sentido no contexto empresarial desenvolver para a arquitetura em questão.

Bibliografia

[1] J. Tulusan, T. Staake, and E. Fleisch, “Providing eco-driving feedback to corporate car drivers,” in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing - UbiComp ’12*, 2012, p. 212.

[2] “Torque Pro (OBD2 / Carro) – Aplicações no Google Play.” [Online]. Available: https://play.google.com/store/apps/details?id=org.prowl.torque&hl=pt_PT. [Accessed: 18-Jun-2018].

[3] “BlueDriver OBD2 Scan Tool – Aplicações no Google Play.” [Online]. Available: https://play.google.com/store/apps/details?id=com.lemurmonitors.bluedriver. [Accessed: 18-Jun-2018].

[4] “DashCommand (OBD ELM App) – Aplicações no Google Play.” [Online]. Available: https://play.google.com/store/apps/details?id=com.palmerperformance.DashCommand. [Accessed: 18-Jun-2018].

[5] “Smart Control Pro (OBD & Car) – Aplicações no Google Play.” [Online]. Available: https://play.google.com/store/apps/details?id=it.smartapps4me.smartcontrol.pro. [Accessed: 18-Jun-2018].

[6] “Is Scrum a methodology? | QuickScrum.” [Online]. Available: https://www.quickscrum.com/Article/ArticleDetails/2026/1/Is-Scrum-a-methodology. [Accessed: 18-Jun-2018].

[7] “FMS-Standard.” [Online]. Available: http://www.fms-standard.com/Truck/index.htm. [Accessed: 03-Sep-2018].

[8] “Protocol Buffers | Google Developers.” [Online]. Available: https://developers.google.com/protocol-buffers/. [Accessed: 05-Jun-2018].

[9] “List of OBD2 and ELM327 compatible vehicles - Outils OBD Facile.” [Online]. Available: https://www.outilsobdfacile.com/vehicle-list-compatible-obd2/. [Accessed: 19-Jun-2018].

[10] “OBDTester.com: ELM-USB commands.”

[11] “Complete List of OBD Codes: Generic OBD2 (OBDII) & Manufacturer | Car OBD Diagnostics, ECU Chip Tuning & Auto Repair Support.” [Online]. Available: http://www.totalcardiagnostics.com/support/Knowledgebase/Article/View/21/0/genericmanufacturer-obd2-codes-and-their-meanings. [Accessed: 19-Jun-2018].

[12] *Announcing Xamarin*. Miguel de Icaza.

[13] “Microsoft to acquire Xamarin and empower more developers to build apps on any device,” *Off. Microsoft Blog*.

[14] “Desenvolvimento de Aplicativos Xamarin com Visual Studio | Visual Studio.” [Online]. Available: https://www.visualstudio.com/pt-br/xamarin/. [Accessed: 18-Jun-2018].

[15] “The MVVM Pattern.” [Online]. Available: https://msdn.microsoft.com/en-us/library/hh848246.aspx. [Accessed: 12-Apr-2018].

Anexos

Table for Telltale status:

Block ID	Telltale Status	Telltale ID	ISO No.	Name	Mandatory Truck only		Block ID	Telltale Status	Telltale ID	ISO No.	Name	Mandatory Truck only
0	1	1	27	Cooling air conditioning			2	1	31	2441	Steering failure	
0	2	2	82	High beam, main beam			2	2	32	2461	Height Control (Levelling)	
0	3	3	83	Low beam, dipped beam			2	3	33	2574	Retarder	
0	4	4	84	Turn signals			2	4	34	2596	Engine Emission system failure (Mil indicator)	
0	5	5	85	Hazard warning			2	5	35	2630	ESC indication	
0	6	6	100	Provision for the disabled or handicapped persons			2	6	36	no	Brake lights	
0	7	7	238	Parking Brake	X		2	7	37	no	Articulation	
0	8	8	239	Brake failure/brake system malfunction			2	8	38	no	Stop Request	
0	9	9	242	Hatch open			2	9	39	no	Pram request	
0	10	10	245	Fuel level	X		2	10	40	no	Bus stop brake	
0	11	11	246	Engine coolant temperature	X		2	11	41	2946	AdBlue level	
0	12	12	247	Battery charging condition			2	12	42	no	Raising	
0	13	13	248	Engine oil	X		2	13	43	no	Lowering	
0	14	14	456	Position lights,side lights			2	14	44	no	Kneeling	
0	15	15	633	Front fog light			2	15	45	no	Engine compartment temperature	
1	1	16	634	Rear fog light			3	1	46	no	Auxillary air pressure	
1	2	17	637	Park Heating			3	2	47	2432	Air filter clogged	
1	3	18	640	Engine / Mil indicator	X		3	3	48	2452	Fuel filter differential pressure	
1	4	19	717	Service, call for maintenance			3	4	49	249	Seat belt	
1	5	20	1168	Transmission fluid temperature			3	5	50	no	EBS	
1	6	21	1396	Transmission failure/malfunction			3	6	51	2682	Lane departure indication	
1	7	22	1407	Anti-lock brake system failure			3	7	52	no	Advanced emergency braking system	
1	8	23	1408	Worn brake linings			3	8	53	2581	ACC	
1	9	24	1422	Windscreen washer fluid/windshield washer fluid			3	9	54	no	Trailer connected	
1	10	25	1434	Tire failure/malfunction			3	10	55	2444/2445	ABS Trailer 1,2	
1	11	26	1603	Malfunction/general failure			3	11	56	2108	Airbag	
1	12	27	2426	Engine oil temperature			3	12	57	no	EBS Trailer 1,2	
1	13	28	2427	Engine oil level			3	13	58	no	Tachograph indication	
1	14	29	2429	Engine coolant level			3	14	59	2649	ESC switched off	
1	15	30	2440	Steering fluid level			3	15	60	no	Lane departure warning switched off	

Figura 25 – Anexo A: Descrição de telltales parte 1.

Block ID	Telltale Status	Telltale ID	ISO No.	Name	Mandatory Truck only		Block ID	Telltale Status	Telltale ID	ISO No.	Name	Mandatory Truck only
4	1	61	2433	Engine emission filter (Soot Filter)								
4	2	62	2633	Electric motor failures								
4	3	63	no	AdBlue tampering								
4	4	64	no	Multiplex System								
4	5	65		Reserved for FMS-Standard								
4	6	66		Reserved for FMS-Standard								
4	7	67		Reserved for FMS-Standard								
4	8	68		Reserved for FMS-Standard								
4	9	69		Reserved for FMS-Standard								
4	10	70		Reserved for FMS-Standard								
4	11	71		Reserved for FMS-Standard								
4	12	72		Reserved for FMS-Standard								
4	13	73		Reserved for FMS-Standard								
4	14	74		Reserved for FMS-Standard								
4	15	75		Reserved for FMS-Standard								

Figura 26 – Anexo B: Descrição de telltales parte 2.