IPG Politécnico da Guarda
Polytechnic of Guarda

RELATÓRIO DE PROJETO

Licenciatura em Engenharia Informática

Diogo André Baltazar Paredes

outubrol | 2016

Escola Superior de Tecnologia e Gestão

Instituto Politécnico da Guarda

APLICAÇÃO ANDROID

ITCHALLENGE

DIOGO ANDRÉ BALTAZAR PAREDES

RELATÓRIO PARA A OBTENÇÃO DO GRAU DE LICENCIADO

EM ENGENHARIA INFORMÁTICA

07/2016

APLICAÇÃO ANDROID

ITCHALLENGE

Diogo André Baltazar Paredes

Aluno de Engenharia Informática (2º Ciclo) 3º Ano, nº1011072

Datas do Estágio Curricular: - Inicio: 6 de Junho de 2016

- Fim: 27 de Julho de 2016

Estágio realizado no âmbito da disciplina de Projecto de Informática, do 2º semestre, do 3º ano, da Licenciatura em Engenharia Informática do Instituto Politécnico da Guarda / Escola Superior de Tecnologia e Gestão.

Instituto Politécnico da Guarda

Escola Superior de Tecnologia e Gestão

Av. Dr. Francisco Sá Carneiro 50, 6300-559 Guarda

Telefone: 271220120 – Fax: 271220150

Email: estg-geral@ipg.pt

IT Sector

Sistemas de Informação, S.A.

Rua José Falcão, 151, 1º/2º, 4050-317 Porto

Telefone: 222058272 – Fax: 222058274

Email: porto@itsector.pt

Plano de Estágio Curricular

O estágio teve a duração de 280 (duzentas e oitenta) horas, aproximadamente um mês e meio, tendo sido iniciado no dia 6 de Junho de 2016 e finalizado no dia 27 de Julho de 2016. O estágio e realização do projeto em contexto de estágio foi realizado na empresa IT Sector, no Porto.

O orientador no Instituto Politécnico foi o Doutor Noel Lopes e na instituição o Eng. Pedro Bacelar. Na atribuição de requisitos para o projeto e aprovação do mesmo na empresa tiveram também contribuição a Dr. Maria Inês Domingues e a Dr. Gabriela Santos.
Na parte gráfica do projeto teve contribuição a UI Designer Priscila Dias.

O plano de estágio curricular teve as seguintes fases e respetivas durações:

1. Academia Android, para aquisição de conhecimentos sobre a tecnologia a ser usada para a realização da aplicação. **(3 semanas)**
2. Análise de requisitos. **(1 dia)**
3. Desenvolvimento da Aplicação. **(2 semanas)**
4. Testes da Aplicação. **(1 semana)**
5. Elaboração do Relatório Final. **(4 semanas)**

Resumo

Neste projeto foi desenvolvida uma aplicação móvel para dispositivos do sistema operativo Android, que teve como objetivo auxiliar o departamento de recursos humanos na área do recrutamento, colmatando assim também a necessidade de tornar o processo o mais abrangente possível. A aplicação tem o nome ITChallenge.

A aplicação foi desenvolvida com base nos requisitos fornecidos pelo departamento de recursos humanos da empresa.

A aplicação regista os dados dos candidatos e fornece um *quiz*, com um total de 20 perguntas, a partir de um total de 6 categorias das quais o candidato escolhe no mínimo 1 e no máximo as 6. Após as 20 questões serem respondidas a aplicação fornece um resultado (percentagem de respostas certas) e dá a hipótese do mesmo ser submetido para a empresa para aprovação do candidato e possível entrevista de emprego.

Para o departamento a aplicação fornece um certo conjunto de dados sobre o candidato, nomeadamente o nome, o *email*, o número de telemóvel, o curso, o grau académico, a universidade e outras informações que o candidato considere relevante.

Para o candidato a aplicação envia, para o *email* do mesmo o resultado obtido, no caso de este submeter o resultado.

O principal objetivo da aplicação era conseguir recolher dados de candidatos em grande escala, por isso foi desenvolvida de forma a ser de fácil e rápida utilização. Com ecrãs bem desenhados e estruturados de forma a proporcionar aos utilizadores uma experiência agradável.

Palavras-Chave: Aplicação Móvel, Android, Empresa, Recursos Humanos, Recrutamento.

Agradecimentos

Expresso os meus sinceros agradecimentos ao meu orientador Doutor Noel Lopes e ao meu orientador na instituição Eng. Pedro Bacelar, pelo apoio, compreensão, colaboração e ajuda que me concederam na preparação e realização do projeto.

Agradeço à Dr. Maria Inês pela hipótese de estágio na empresa.

Agradeço à minha namorada pelo apoio, paciência e compreensão.

Finalmente agradeço a toda a minha família e amigos sem desprezar nenhum.

A todos os referidos, muito obrigado

Diogo Paredes

Índice Geral

Índice de Figuras

Glossário

- **Outsourcing –** É um processo através do qual uma organização (contratante) contrata outra (subcontratado), na perspetiva de manter com ela um relacionamento mutuamente benéfico, de médio ou longo prazo, com vista ao desempenho de uma ou várias atividades, que a primeira não pode ou não lhe convém desempenhar e que a segunda é tida como especialista. [2]
- **Nearshore** – é uma forma de terceirização integral de serviços e processos, definido e acordado com o cliente, que se refere a serviços externos e fornecidos a partir de uma região adjacente (país) ou nas proximidades do local onde esses serviços são prestados.
- **Interface** – Em Java, é um conjunto de tipos de métodos que as classes podem implementar.
- **IDE** – Integrated Development Environment – Ambiente Integrado de Desenvolvimento – É um programa que reúne características e ferramentas de apoio ao desenvolvimento de software com o objetivo de agilizar este processo.
- **Web Service –** é uma solução utilizada na integração de sistemas e na comunicação entre aplicações diferentes. Permitem às aplicações enviar e receber dados em formato XML, JSON, etc.
- **Git –** É um sistema de controle de versões distribuído e um sistema de gerenciamento de código fonte, com ênfase na velocidade.
- **API** – Application Programming Interface – Interface de Programação de Aplicações – É um conjunto de funções e padrões estabelecidos por um software para a utilização das suas funcionalidades em aplicações.

1. Introdução

1.1. Fundamentação do Projeto

Este projeto vem ajudar o departamento de Recursos Humanos na importante tarefa de recrutamento, numa fase em que a empresa se encontra em grande expansão e a precisar de novos talentos frequentemente.

A empresa está neste momento a crescer e a expandir para novos mercados internacionais. Encontra-se numa fase em que o número de projetos e a projeção para o futuro cria a necessidade de aumentar o número de colaboradores para poder cumprir com as metas.

Para além disso a empresa está perto de inaugurar mais um escritório, no final de setembro em Aveiro e possivelmente outro para o ano (2017) em Faro.

O recrutamento é algo que pode levar o seu tempo e que é difícil de realizar em grande escala daí a necessidade de se automatizar o processo, para assim serem alcançadas grandes massas. A aplicação desenvolvida no projeto visa colmatar este pormenor e proporcionar assim uma espécie de triagem e registo de um grande número de candidatos a possíveis cargos na empresa.

A aplicação irá ser disponibilizada pelo departamento de RH da empresa quando este for a eventos ou instituições de ensino podendo assim os alunos ou portadores de grau académico de licenciado ou superior, efetuarem um simples registo e um teste de aptidões consoantes as categorias que escolherem (provavelmente nas quais se sentirem mais à vontade ou que gostem mais). A empresa depois recebe essa informação filtrada e por categorias respondidas, facilitando imenso a seleção por área de aptidão para entrevistas dos candidatos.

1.2. A Empresa

- A empresa, cujo logotipo está representado na figura 1, onde o estágio foi realizado é a IT Sector. "Fundada em 2005, a IT Sector – Sistemas de Informação, S.A., é uma Empresa Portuguesa de Desenvolvimento de Software, criada com o objetivo de oferecer ao mercado Soluções de Sistemas de Informação de elevado valor acrescentado."(Manual do Colaborador ITsector, 2015).

Figura 1 - Logotipo da empresa (Manual do Colaborador ITsector, 2015)

"A rede empresarial, representada na figura 2, do Grupo IT Sector é constituída por: a ebankIT, uma empresa especializada em produtos multicanal direcionada para o mercado financeiro; a Bitmaker, uma empresa de desenvolvimento de software e de serviços de outsourcing especializada para a entrega de projetos críticos; e com o objetivo de fortalecer as alianças com a África, a IT sector formou parceria estratégica com Angola CPCÁfrica e é a acionista da CPCÁfrica Moçambique."(Manual do Colaborador ITsector, 2015).

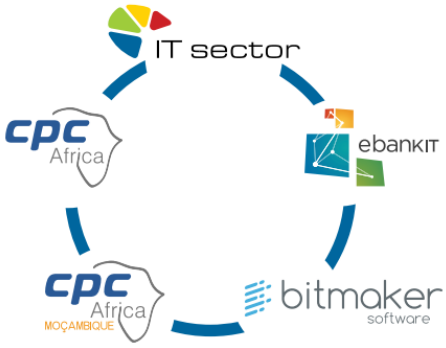


Figura 2 - Rede Empresarial (Manual do Colaborador ITsector, 2015)

"Os serviços de *Outsourcing* e de desenvolvimento em regime *Nearshore*, são a principal valência, sendo que a partir de localizações estratégicas em Portugal, a empresa desenvolve software para mais de 20 países, dos quais alguns são representados na figura 3 como por exemplo: Angola, Moçambique, Reino Unido, Luxemburgo, Polónia, Rússia, França, Islândia, Dinamarca, Macau, Timor, África do Sul, entre outros."(Manual do Colaborador ITsector, 2015).

Figura 3 - Clientes da ITSector (Manual do Colaborador ITsector, 2015)

Soluções em mercados financeiros, telecomunicações e saúde, representados na figura 4.

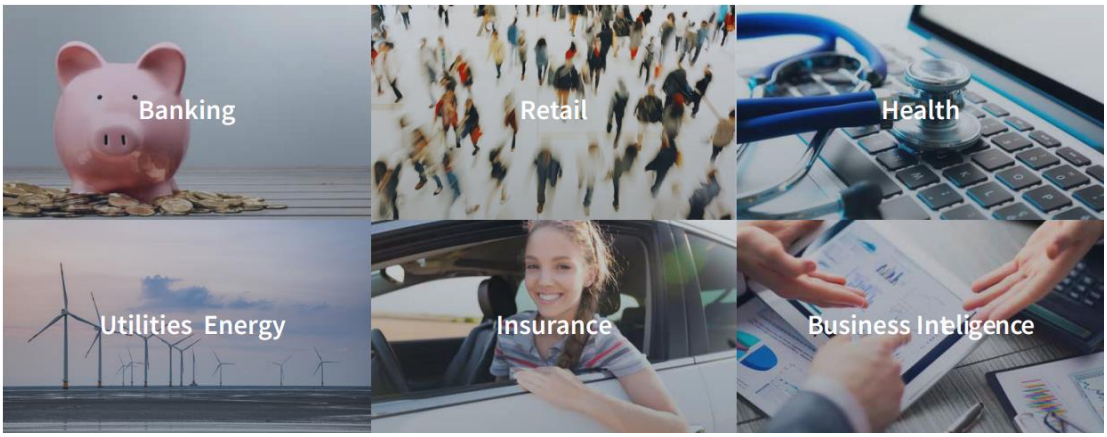


Figura 4 – Soluções (Manual do Colaborador ITsector, 2015)

Os pilares da cultura da IT Setor são:

- Satisfação do Cliente, Promovendo a Confiança e a Credibilidade
- Ética e Responsabilidade Social
- Iniciativa, Dinamismo e Inovação
- Flexibilidade e Autonomia
- Desenvolvimento dos Recursos Humanos

Organigrama da empresa representado na figura 5.

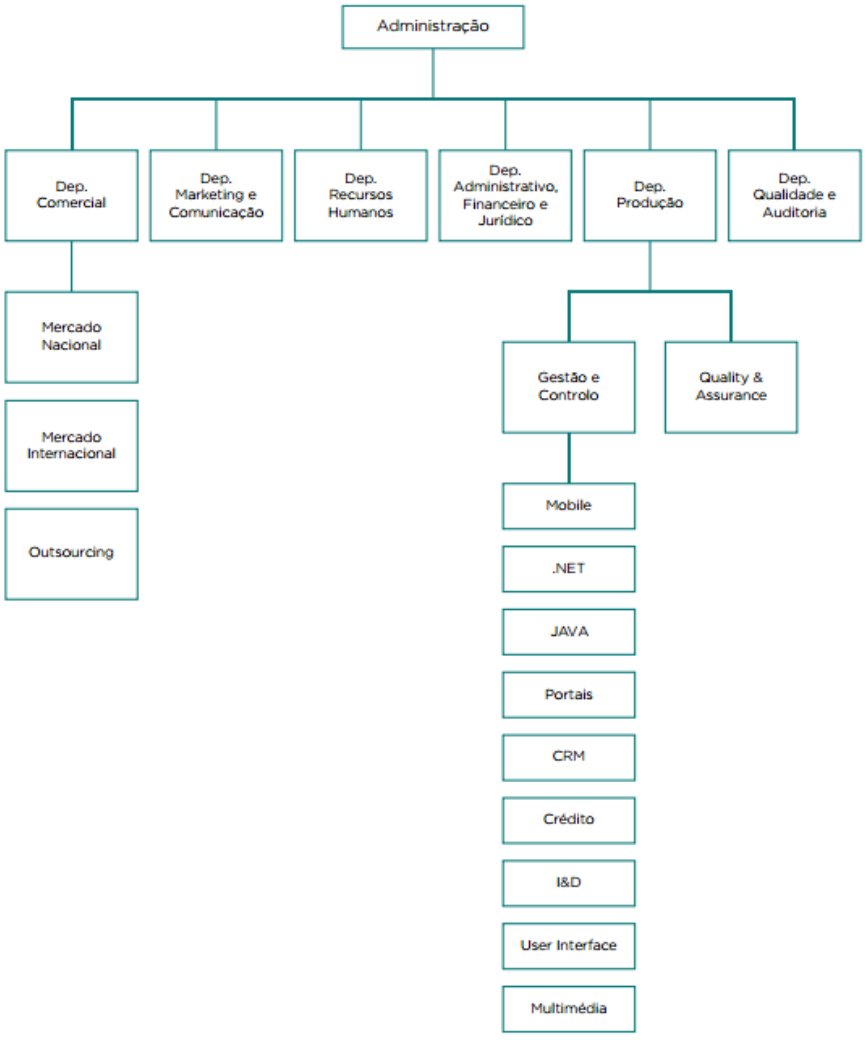


Figura 5 - Organigrama da Empresa (Manual do Colaborador ITsector, 2015)

Na figura 6 estão representados os escritórios da IT Sector.

Figura 6 – Escritórios (Manual do Colaborador ITsector, 2015)

A empresa foi em 2015 PME Líder, contudo este ano 2016 deixou de ser PME. A IT Sector tem neste momento em fase de finalização a abertura de mais um escritório em Portugal, este em Aveiro e para o ano está previsto a abertura de um em Faro.

No escritório onde o estágio foi realizado, departamento do Porto, a IT Sector conta com 134 colaboradores. Sendo a maior parte dos colabores da área de .NET.

1.3. Motivação

O desenvolvimento deste projeto foi motivado pela possibilidade de trabalhar com novas tecnologias, como o desenvolvimento de aplicações móveis para o sistema operativo Android. Algo que também me motivou foi a oportunidade de realizar o projeto num ambiente empresarial, o que me ajudou em termos profissionais a adquirir conhecimentos e experiência.

Por fim algo que também me deu motivação é o facto da aplicação ter uma grande utilidade para a empresa, nomeadamente para o departamento de Recursos Humanos.

1.4. Objetivos do Projeto

Os objetivos da aplicação móvel são os seguintes:

- Facilitar o processo de recrutamento para o departamento de RH;
- Fornecer os dados obtidos ao departamento de RH de uma forma clara, enviando um email estruturado com os dados fornecidos pelo utilizador no formulário e o resultado do Quiz.
- Enquadrar os candidatos de acordo com as suas aptidões tendo em conta as necessidades da empresa;
- Abranger uma larga escala de utilizadores (neste caso estudantes);
- Fornecer uma *interface* de fácil utilização;
- Fornecer uma aplicação leve e ao mesmo tempo rápida, a aplicação deve gastar pouca memória do dispositivo e ao mesmo tempo poucos recursos;
- Fornecer uma forma eficiente de registo dos dados do estudante, dados estes fornecidos pelo mesmo no formulário da aplicação;
- Fornecer perguntas consoantes as categorias escolhidas pelo estudante;
- Fornecer aleatoriamente perguntas e respostas para evitar que o utilizador corra a aplicação várias vezes e tenha várias perguntas repetidas;

1.5. Estrutura do documento

O documento para além deste capítulo introdutório contém mais seis capítulos. Estando assim organizado da seguinte forma:

- No segundo capítulo é descrito o estado da arte com aplicações a servirem de exemplo;
- No terceiro capítulo é descrita a metodologia usada no projeto, as tecnologias e software;
- No quarto capítulo é apresentado a academia, onde foram aprendidos os conhecimentos base para trabalhar com o Android, e algum do trabalhado desenvolvido na mesma;
- No quinto capítulo é apresentada a solução, com alguns pormenores de forma mais detalhada;
- No sexto capítulo são apresentados os resultados após a finalização da aplicação;
- No sétimo e último capítulo são feitas algumas considerações finais sobre o projeto e ideias para implementações futuras.

2. Estado da Arte

2.1. Enquadramento Teórico

As aplicações móveis, conhecidas normalmente pelo seu nome abreviado de *app*, são um software desenvolvido para ser instalado num dispositivo móvel, como por exemplo um *smartphone* ou um *tablet*.

Uma grande parte das aplicações disponíveis são gratuitas, enquanto outras são pagas. Estas aplicações podem ser pré-instaladas no dispositivo diretamente na fábrica, descarregadas pelos utilizadores dos mesmos de várias plataformas de distribuição de software móvel, como por exemplo a Google Play ou Play Store no caso de dispositivos Android ou a AppStore no caso de dispositivos do sistema operativo iOS.

O número de downloads de aplicações móveis está em forte crescimento. Esta tendência está associada com a venda de *smartphones*, que tem tido um grande crescimento ao longo dos anos.

No âmbito da utilização das aplicações móveis para a finalidade de recrutamento não foi encontrado nada igual, contudo há aplicações que ajudam e tornam o processo de recrutamento para as empresas mais simples, embora não do mesmo modo que a aplicação desenvolvida neste projeto.

Através destas aplicações é possível para uma empresa registar propostas de emprego e recolher dados dos candidatos ou simplesmente pesquisar do leque de utilizadores das aplicações por qualidades ou qualificações específicas. Neste projeto serão analisadas as seguintes aplicações: LinkedIn e Jobs by CareerBuilder.

No caso do LinkedIn é uma rede social de negócios fundada em dezembro de 2002 e lançada a 5 de maio de 2003. É comparável a redes sociais de relacionamentos como por exemplo o Facebook, só que é principalmente utilizada por profissionais.

O objetivo do LinkedIn é fornecer uma aplicação web, ou seja, disponível a partir de um browser, e agora também em versão móvel. Na qual qualquer pessoa se pode registar, fornecer as suas aptidões, qualificações e informações que ache relevantes para as

empresas. Os departamentos de Recursos Humanos das empresas podem assim usar a aplicação para procurar por diversos interesses, sejam eles por qualidades ou certificações, como por percurso académico ou experiência profissional.

Percebemos assim que o LinkedIn não vai ao encontro do nosso objetivo para o projeto pois é pretendida uma aplicação que possa ser disponibilizada ao utilizador e neste caso o utilizador terá de ser "encontrado" pela empresa, enquanto que neste projeto a empresa fica logo a conhecer o candidato que apenas tem de ser submetido a uma seleção.

No caso do Jobs by CarrerBuilder, é uma aplicação móvel criada a partir da aplicação web CarrerBuilder, que fornece ao utilizador a possibilidade de pesquisar por oportunidades de empregos. O utilizador pode pesquisar por palavras-chave, localização da oferta, etc. Após encontrar uma oferta que agrade ao utilizador este pode assim submeter a candidatura à oferta, enviando os seus dados, o currículo e uma carta de recomendação que será opcional.

Ao contrário da aplicação analisada em cima (LinkedIn), nesta o utilizador em vez de se registar e esperar por um contacto de uma empresa, tem de procurar entre as ofertas disponíveis publicadas por empresas ou outros utilizadores, uma que seja do seu interesse e candidatar-se à mesma.

De seguida é possível ver a aplicação CarrerBuilder web na figura 7 e a versão móvel na figura 8.

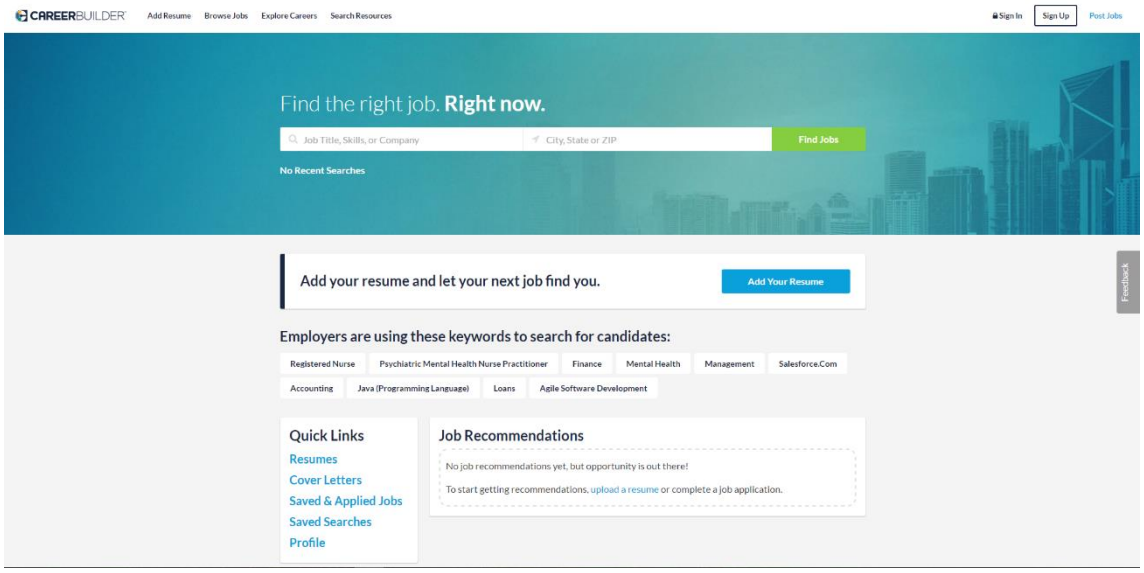


Figura 7 - CarrerBuilder website

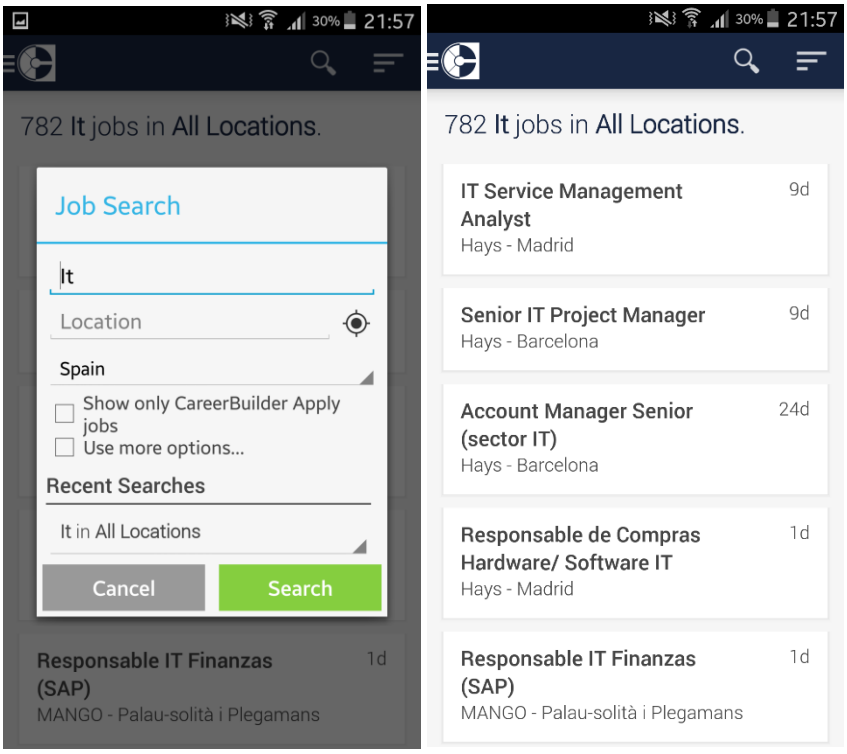


Figura 8 - CarrerBuilder versão móvel

Foi também efetuada uma pesquisa e estudo sobre o recrutamento em geral. O recrutamento divide-se em dois tipos: o interno e o externo. Neste caso o estudo focou-se no recrutamento externo, que é o recrutamento utilizado para atrair candidatos de fora da instituição, porque é o recrutamento que vai de encontro ao objetivo da aplicação desenvolvida. Pois o recrutamento interno é apenas dentro da empresa.

"O recrutamento externo tem as suas vantagens e desvantagens:

Vantagens:

• introduz sangue novo na organização: talentos, habilidades e expectativas.

• enriquece o patrimônio humano, pelo aporte de novos talentos e habilidades.

• aumenta o capital intelectual ao incluir novos conhecimentos e destrezas.

• renova a cultura organizacional e a enriquece com novas aspirações.

• indicado para enriquecer mais intensa e rapidamente o capital intelectual.

Desvantagens:

• afeta negativamente a motivação dos atuais funcionários da organização.

• reduz a fidelidade dos funcionários ao oferecer oportunidade a estranhos.

• requer aplicação de técnicas seletivas para escolha dos candidatos externos.

Isso significa custos operacionais.

• exige esquemas de socialização organizacional para os novos funcionários.

• é mais custoso, oneroso, demorado e inseguro que o recrutamento interno." (Murillo 2007).

Com esta informação podemos entender que para a aplicação ter sucesso terá de ter em conta as desvantagens e tentar de alguma forma supera-las ou minimiza-las.

Foi estudado também como as empresas da área da Informática e das Tecnologias de Informação (TI) fazem o seu recrutamento. É dado o exemplo de 3 empresas, a Portugal Telecom (PT), a NOS e a Apple:

- No caso da Portugal Telecom (PT), o recrutamento é feito a partir do site da empresa, onde o candidato tem que ir à zona das carreiras e assim submeter uma candidatura espontânea, sem garantias que a mesma seja vista;
- No caso da NOS, o recrutamento é feito tal como na Portugal Telecom, ou seja, a partir do site da empresa, contudo com a diferença que para além das candidaturas espontâneas, o candidato pode ainda ver uma lista de vagas disponíveis que a empresa tem e submeter candidatura a essas vagas ou então enviar currículo para um programa de treino. Nesse programa o candidato é treinado para uma função especifica da empresa, este programa denomina-se "NOS Alfa";
- No caso da Apple, a empresa opta por um recrutamento mais restrito em que continuando com o recrutamento a partir do site da empresa, esta só permite aos candidatos submeterem candidaturas para vagas que a mesma

mostre disponíveis no site, não permitindo assim candidaturas espontâneas, nem programas internos de treino.

2.2. Análise Crítica

Embora existam aplicações que disponibilizem formas de recrutamento, e embora todas elas bastante completas, falham no aspeto da portabilidade e da proximidade.

O que esta aplicação fornece é uma possibilidade de a empresa recrutar candidatos na hora, ou seja, se a empresa estiver num evento seja ele onde for, esta tem a possibilidade de fornecer algo que as pessoas podem usar para se candidatarem a um emprego nessa empresa. Contrariando assim o atual método das empresas que reside no facto de encaminhar as pessoas para a sua página e fazendo as mesmas procurar pela zona de recrutamento, carreira ou o nome que a empresa lhe der. Isto faz com que as pessoas tenham de ter acesso a um dispositivo com acesso à internet e a um browser, onde depois terão que aceder ao site dessa mesma empresa, terão de ver as oportunidades existentes e submeter uma candidatura espontânea, sem nenhuma garantia que irão receber resposta ou que a candidatura será vista.

A aplicação desenvolvida neste projeto dá uma hipótese às pessoas interessadas fazerem algo inovador no meu ponto de vista. Um pequeno teste de aptidões, no qual ficam a saber as áreas em que a empresa está a apostar. Para além disso garante às pessoas uma hipótese de decisão entre enviar os dados para a empresa ou cancelar. A aplicação dá também a garantia que se forem submetidos os dados para a empresa esta irá analisar os mesmos.

Por fim, creio que assim a experiência de recrutamento torna-se melhor não só para a empresa, que não perde tanto tempo a registar dados e pode por exemplo apresentar melhor a empresa e os projetos em que trabalham, assim como tecnologias e afins, como para os próprios candidatos que em vez de terem de ir visitar um site à procura de oportunidades de emprego, podem testar os seus conhecimentos e ao mesmo tempo arranjar emprego no futuro.

3. Métodos e Tecnologias

3.1. Metodologia

A metodologia usada para o desenvolvimento do projeto foi uma metodologia ágil, o Scrum.

"A ideia principal do Scrum é que o desenvolvimento de software envolve muitas variáveis técnicas e do ambiente, como requisitos, recursos e tecnologia, que podem mudar durante o processo. Isto torna o processo de desenvolvimento imprevisível e complexo, requerendo flexibilidade para acompanhar as mudanças. O resultado do processo deve ser um software que é realmente útil para o cliente."(Schwaber, 1995).

No Scrum, os projetos são divididos em ciclos (tipicamente mensais) chamados de Sprints. O Sprint representa prazo dentro do qual um conjunto de atividades devem ser executadas. As metodologias ágeis de desenvolvimento de software são iterativas, ou seja, o trabalho é dividido em iterações, que no caso do Scrum são chamadas de *Sprints*. As funcionalidades a serem implementadas num projeto são mantidas numa lista que é conhecida como *Product Backlog*. No início de cada Sprint, faz-se um *Sprint Planning Meeting*, ou seja, uma reunião de planeamento. Na qual o *Product Owner* define prioridades para os itens do *Product Backlog* e a equipa seleciona as atividades que ela será capaz de implementar durante o *Sprint*. As tarefas alocadas num *Sprint* são transferidas do *Product Backlog* para o *Sprint Backlog*.

A cada dia do *Sprint*, a equipa faz uma breve reunião chamada *Daily Scrum*. O objetivo é disseminar conhecimento sobre o que foi feito no dia anterior, identificar impedimentos e priorizar o trabalho do dia seguinte. Ao final de um *Sprint*, a equipa apresenta as funcionalidades implementadas numa *Sprint Review Meeting*. Finalmente, faz-se uma *Sprint Retrospective* e a equipe parte para o planeamento do próximo *Sprint*. Assim reinicia-se o ciclo. Como se pode ver na figura 9.

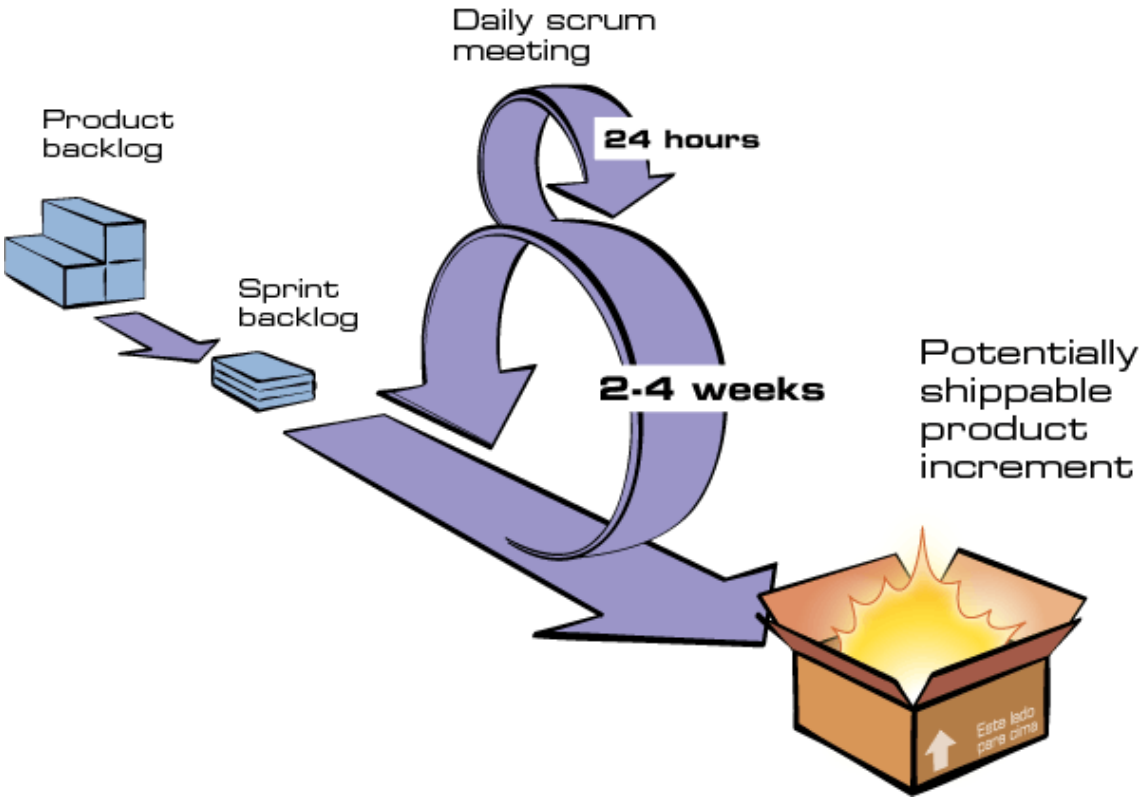


Figura 9 - Scrum Life Cycle

3.2. Tecnologias

Durante o projeto foram utilizadas algumas tecnologias que vão ser abordadas nos subcapítulos a seguir.

Estas tecnologias foram usadas no projeto para diversos casos:

- Android – base da aplicação;
- Java – linguagem de programação usada para programar para o Android;
- XML – design da aplicação e ficheiros com as perguntas e respostas;
- HTML – para estrutura do *email* enviado pela aplicação;
- SMTP – para o envio dos *emails*;
- MIME – para resolver o problema das imagens enviadas por *email*.

3.2.1. Android

O Android é um sistema operativo para dispositivos móveis, baseado em Linux e que é atualmente desenvolvido pela empresa de tecnologia Google. O Android é projetado principalmente para dispositivos móveis com um ecrã sensível ao toque como é o caso dos *smartphones* e *tablets*. Atualmente também se encontram versões disponíveis para televisões inteligentes (Android TV), para o carro (Android Auto) e para relógios (Android Wear).

O Android é o sistema operativo móvel mais utilizado no mundo. Grande parte do seu sucesso vem do facto de a própria Google disponibilizar o código do mesmo sob licença de código aberto. O que faz com que empresas que produzem equipamentos tecnológicos e que precisem de um sistema operativo para o mesmo possam assim usar o Android, pois este é um software de baixo custo, já pronto a ser utilizado e de fácil personalização.

O facto de ter o seu código aberto encorajou ao longo dos anos uma grande comunidade de programadores e entusiastas que adicionam assim recursos ao sistema, e até nalguns casos levam o sistema a dispositivos que inicialmente não eram aptos a correr o mesmo.

3.2.2. Java

Java é uma linguagem de programação orientada a objetos. Diferente das linguagens de programação convencionais, que são compiladas para código nativo, a linguagem Java é compilada para um *bytecode* que é executado por uma máquina virtual. A linguagem de programação Java é a linguagem convencional da Plataforma Java.

3.2.3. XML

XML é uma linguagem de marcação, que serve para criar uma infraestrutura única para diversas linguagens.

No caso deste projeto é usada para criar os ficheiros de onde depois vão ser carregadas as perguntas e respostas, e é usada pelo Android para a parte gráfica da aplicação é para o manifesto da aplicação. O manifesto apresenta informações essenciais sobre a aplicação ao sistema Android, necessárias para o sistema antes que ele possa executar o código da aplicação. Entre outras coisas, o manifesto serve para:

- Dar um nome à aplicação;
- Declarar as permissões;
- Declarar o nível mínimo da API do Android que a palicação requer para puder ser executada num dispositivo;
- Listar as bibliotecas às quais a aplicação vai aceder.

3.2.4. HTML

HTML é uma linguagem de marcação utilizada na construção de páginas *web*. Ela usa um conjunto de *tags* de marcação para descrever as páginas da *web*. Essas *tags* são palavras-chave como <html>, elas são usadas em pares como <b> e </ b>. A primeira *tag* num par é a *tag* de início enquanto a segunda é a do fim. A figura 10 ilustra um pedaço de código HTML.

```
<!DOCTYPE html>
<html>
<head>
<meta>
<title>Exemplo HTML</title>
</head>
<body>
<h1>Olá Mundo!</h1>
<p>Isto é um exemplo</p>
</body>
</html>
```

Figura 10 - HTML Exemplo

3.2.5. SMTP

SMTP ou Protocolo de transferência de correio simples, é o protocolo padrão para envio de e-mails através da Internet. Ele foi usado para a aplicação enviar o *email* com os dados do candidato e resultado para a empresa, e o *email* de "participação" com o resultado para o candidato. No caso deste projeto o servidor SMTP usado foi o da Google. A figura 11 ilustra o processo de um *email* enviado por SMTP.

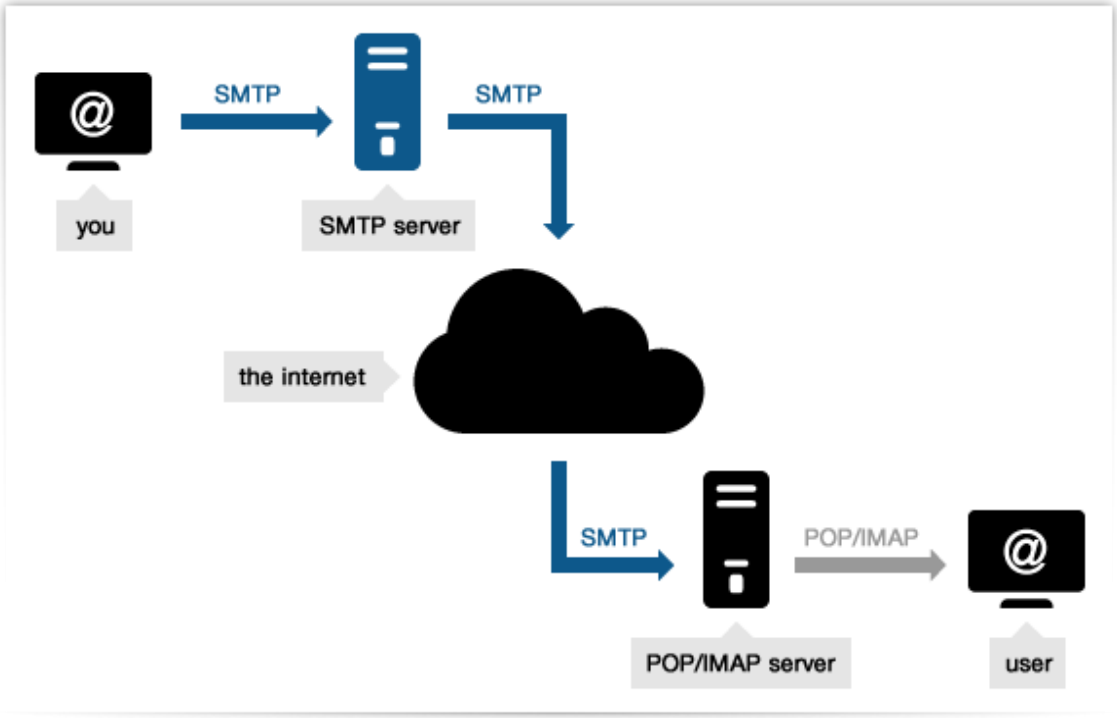


Figura 11 - SMTP

3.2.6. MIME

MIME é uma extensão que permite a troca de diferentes tipos de dados pela internet, como por exemplo: som, vídeo, imagens, aplicações, etc.

Ele foi usado no projeto para conseguir enviar imagens no *email* e evitar assim que elas fossem bloqueadas pelos serviços de *email* como o Gmail da Google e o Outlook da Microsoft.

3.3 Software

O *IDE* utilizado para desenvolver a aplicação Android foi o oficial da Google, o Android Studio (representado na figura 12).

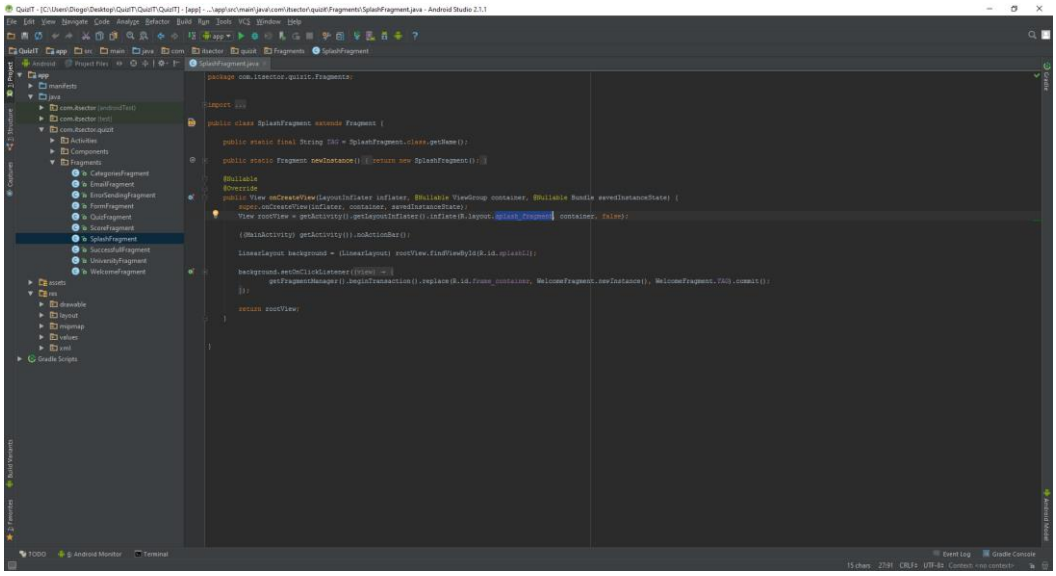


Figura 12 - Android Studio

Para controlo de versões era usado o *Git*, para evitar o uso da linha de comandos foi instalado o TortoiseGit que torna a utilização do *Git* mais fácil pois acrescenta interface gráfico, como se pode ver na figura 13.

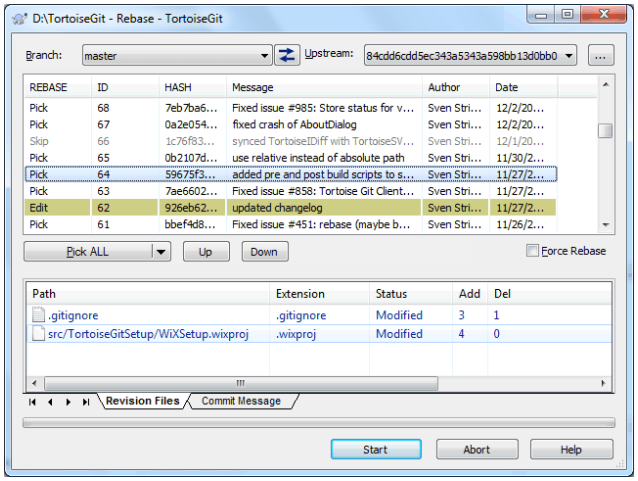


Figura 13 - TortoiseGit

Para comunicação com a equipa de design era usado o Skype.

4. Academia Android

4.1. Introdução

Como na licenciatura a cadeira de Programação para Dispositivos Móveis era opcional e não foi escolhida pelo aluno, a empresa ofereceu a possibilidade de frequentar uma academia interna de 3 semanas na qual foram adquiridos conhecimentos fundamentais para o projeto puder ser realizado.

Durante a academia foram realizados pelo aluno tutoriais dos quais se adquiriram conhecimentos básicos e intermédios sobre o sistema operativo e como programar para ele. Foram desenvolvidas aplicações que serviam assim para consolidar conhecimentos adquiridos e testar os mesmos. A academia seguiu um programa de ensino apoiado pela própria Google, o que a tornava muito bem estruturada.

4.2. Aplicações desenvolvidas

Algumas das aplicações que foram criadas para testar os conhecimentos adquiridos foram:

- Uma aplicação para registar o nome do cliente e mediante o pedido efetuado, criar assim um *email* (na aplicação de email do dispositivo) com os dados da encomenda pronto a enviar, como se pode ver na figura 14.

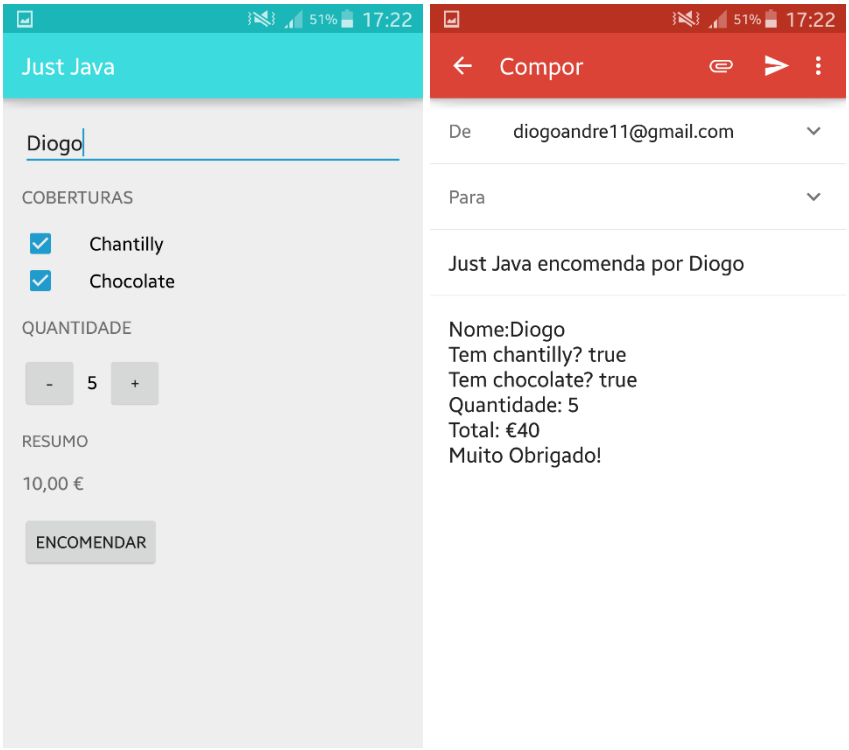


Figura 14 - Just Java App

- Uma aplicação para ir guardando o resultado de um jogo de basketball, como se pode ver na figura 15.

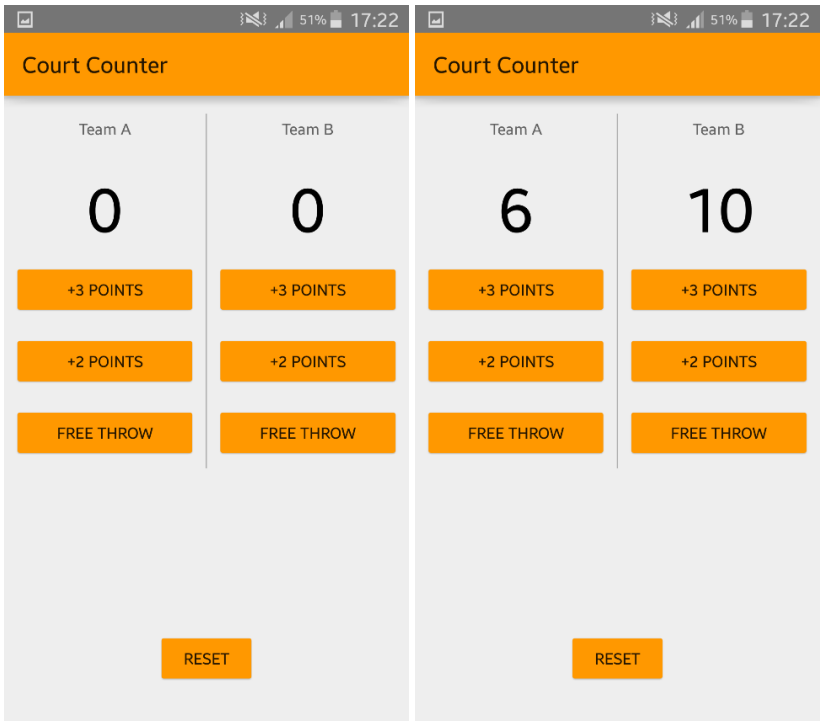


Figura 15 - Court Counter App

4.3. Aplicação final da Academia Android

Para além das aplicações desenvolvidas durante a academia foi também desenvolvida na última semana da mesma, e como projeto final da academia, uma aplicação que englobasse todos os conhecimentos adquiridos.

O objetivo da aplicação final da academia era que a mesma fornecesse os dados meteorológicos de uma região à escolha do utilizador. A aplicação teria de recolher essa informação a uma *API* gratuita (OpenWeather) e traduzir a mesma para uma *interface* de fácil compreensão para o utilizador, representada na figura 16.

Essa informação deveria cobrir um período de 15 dias a partir do dia atual do sistema. Selecionando um dia em específico, como por exemplo o dia 4 de Agosto, a aplicação fornece um ecrã com uma informação mais detalhada como se pode ver na figura 17.

Figura 16 - Sunshine Ecrã Principal

Figura 17 - Sunshine Ecrã Dia

Para além destas funcionalidades a aplicação permite ainda a partilha dos dados através de várias formas como por exemplo mensagem, *email* e outras. A aplicação guarda a informação numa base de dados interna atualizando a mesma automaticamente após um período de 1 semana desde a última atualização, quando uma atualização acontece o utilizador é avisado através de uma notificação que a aplicação cria.

Quando uma atualização dos dados é feita os dados mais antigos são apagados da base de dados interna, evitando assim que o tamanho da aplicação aumente ao longo do tempo, visto ser necessário apenas manter a informação atual e futura.

A aplicação permite ao utilizador para além de escolher a região como já referido, a possibilidade de desativar as notificações e de escolher se quer os dados em unidades métricas (Celcius) ou unidades imperiais (Fahrenheit), permite também ao utilizador consoante a região selecionada ver o mapa da mesma através da aplicação de mapa do dispositivo em que está instalada.

Após a conclusão da academia foi feita uma reunião com os representantes do departamento de RecursosHumanos e iniciado o desenvolvimento da aplicação ITChallenge.

5. Implementação da Solução

5.1. Fluxograma da Aplicação

A aplicação pode ser interpretada a partir de um simples fluxograma como representado na figura 18.

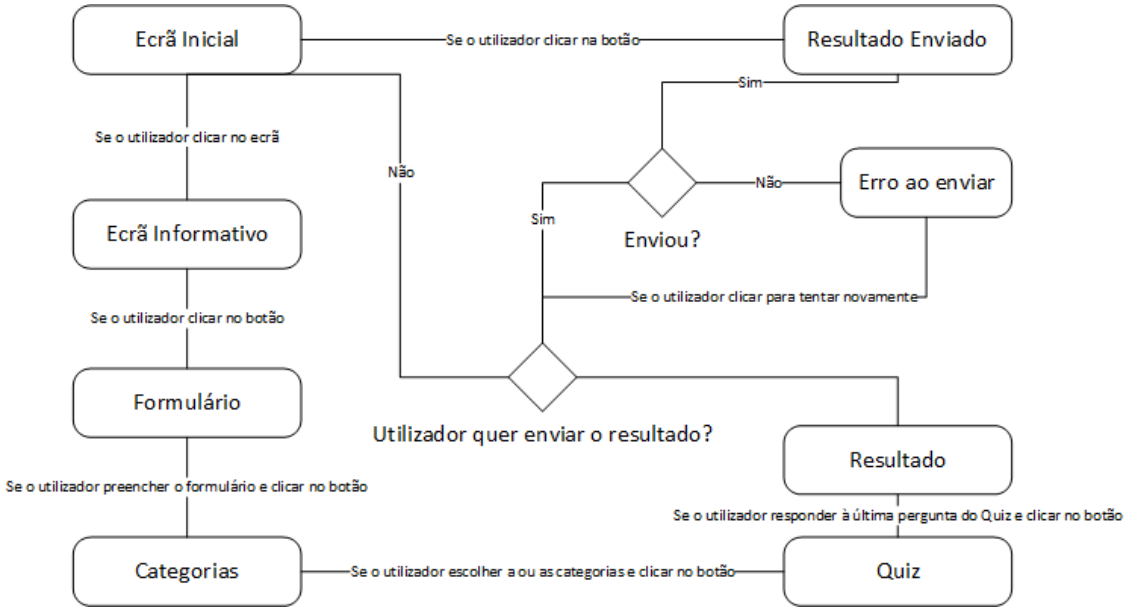


Figura 18 – Fluxograma da Aplicação

5.2. Arquitetura da Aplicação

A arquitetura da aplicação, ilustrada na figura 19, é a que vai de encontro aos objetivos da mesma. Na altura de desenvolvimento a empresa queria os dados diretamente para o *email*, e também não havia hipótese de se criar um *Web Service* e base de dados para a aplicação num dos servidores da empresa.

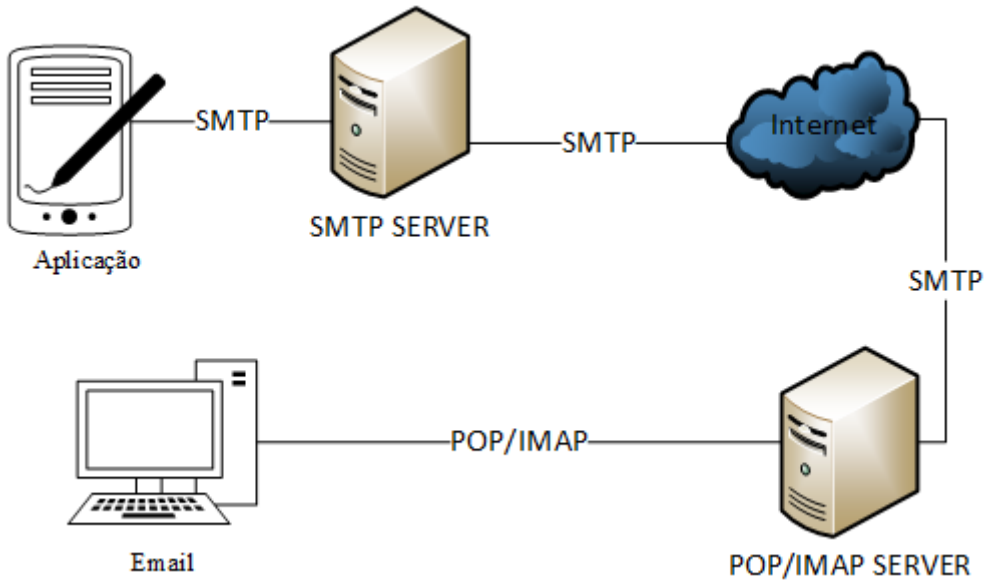


Figura 19 – Arquitetura da Aplicação

A arquitetura ideal era a aplicação comunicar com um *Web Service* e armazenar os dados dos candidatos para estes depois poderem ser analisados ao pormenor e nunca se perder informação. Isto ia também proporcionar uma melhor manutenção da aplicação pois seria possível atualizar perguntas e respostas pois estas seriam obtidas diretamente do servidor ao contrário do modelo atual em que são obtidas por ficheiros internos da aplicação. A figura 20 ilustra a arquitetura ideal.

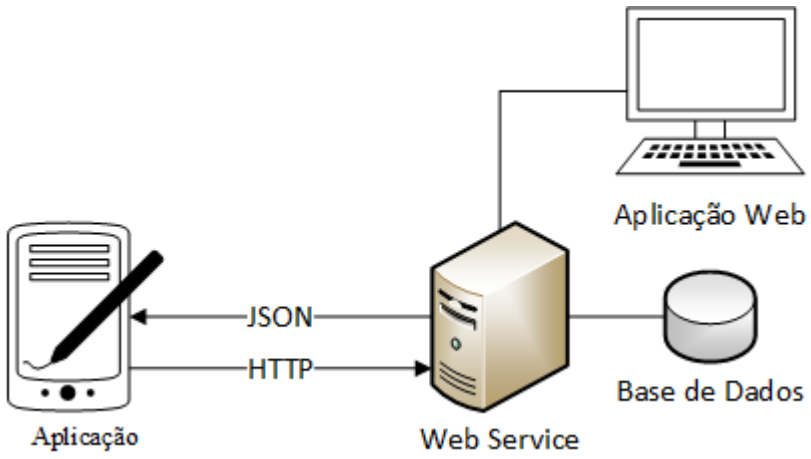


Figura 20 - Arquitetura ideal para a Aplicação

5.3. Design

O Design da aplicação foi definido pela equipa de design que criou vários ecrãs para serem usados como referências e *guidelines* para serem seguidas.

Algumas das especificações do design que estão presentes nas *guidelines* são:

- Tipografia: foram definidos dois tipos de letra para serem usados na aplicação;
- Cores: foram definidas 7 cores e apenas estas podiam fazer parte, sendo que 4 eram variantes do azul (para dar várias tonalidades);
- Grelha: foram definidos limites e margens para os ecrãs da aplicação;
- Barra de notificações e da aplicação: foram definidas as cores a usar em ambas as barras;
- Ecrã do Formulário: foram definidos requisitos objetos deste ecrã, como por exemplo os títulos, os campos, os botões, etc;
- Lista de Universidades: foram definidas margens para a lista das universidades e especificações para o campo de procura;
- *Quiz*: foram definidas as margens entre as respostas, margens do campo das perguntas e especificações para ambas;

As *guidelines* podem ver vistas nos anexos.

A equipa de design criou e forneceu também os ícones para a aplicação. Estes foram todos criados a pensar em várias resoluções de dispositivos como se pode ver no caso do ícone da aplicação na figura 21.

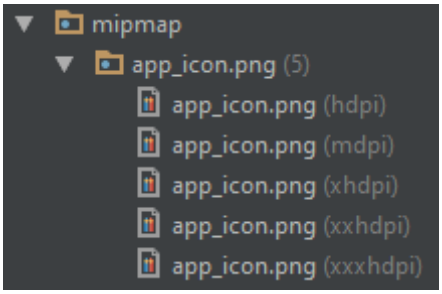


Figura 21 - Várias Resoluções de Ícone

5.4. Estrutura e código

A aplicação foi desenvolvida com a estrutura que se pode ver na figura 22.

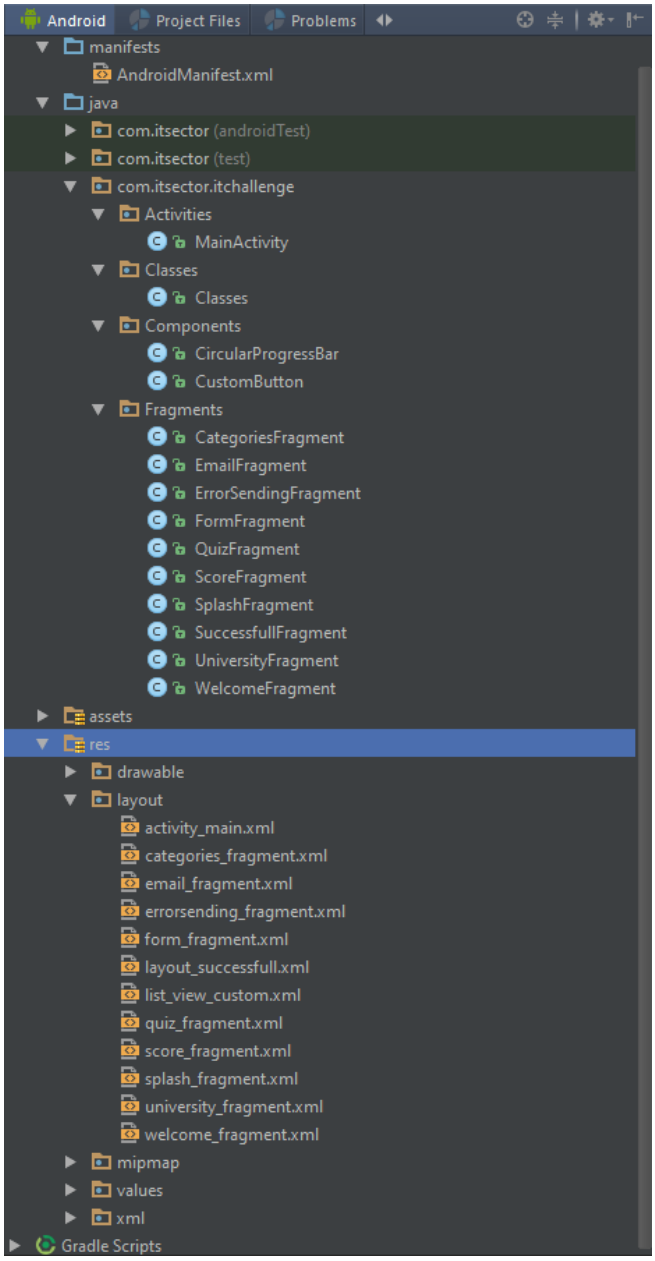


Figura 22 - Estrutura do Projeto

A aplicação é constituída por uma atividade e um conjunto de fragmentos, os fragmentos representam o comportamento ou uma parte da *interface* do utilizador numa atividade.

Assim que é iniciada ela corre a atividade principal (MainActivity), e esta carrega o primeiro fragmento. A aplicação tem 10 fragmento que são:

- SplashFragment;
- WelcomeFragment;
- FormFragment;
- UniversityFragment;
- CategoriesFragment;
- QuizFragment;
- ScoreFragment;
- EmailFragment;
- SuccessfullFragment;
- ErrorSendingFragment;

5.4.1. SplashFragment

O ecrã inicial da aplicação tem apenas um propósito introdutivo à mesma. É apenas uma simples imagem de fundo que avança para o próximo fragmento assim que o utilizador clica em qualquer ponto do ecrã. O código é bastante simples como se pode ver na figura 23, em que se cria um evento que vai aguardar um clique em qualquer parte da imagem de fundo e quando esse clique acontecer faz uma transição para outro fragmento.

```java
public class SplashFragment extends Fragment {

    public static final String TAG = SplashFragment.class.getName();

    public static Fragment newInstance() { return new SplashFragment(); }

    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container,
                             @Nullable Bundle savedInstanceState) {
        super.onCreateView(inflater, container, savedInstanceState);
        View rootView = getActivity().getLayoutInflater().inflate(
                R.layout.splash_fragment, container, false);

        ((MainActivity) getActivity()).noActionBar();

        LinearLayout background = (LinearLayout) rootView.findViewById(R.id.splashLl);

        background.setOnClickListener((view) -> {
                getFragmentManager().beginTransaction().replace(R.id.frame_container,
                        WelcomeFragment.newInstance(), WelcomeFragment.TAG).commit();
        });
        return rootView;
    }
}
```

Figura 23 - Código do Ecrã Inicial

A nível de design o ecrã inicial é apenas uma imagem de fundo, o código para criar o design deste ecrã é ilustrado pela figura 24.

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/splashLl"
    android:background="@drawable/splash">
</LinearLayout>
```

Figura 24 - Código de Design do ecrã inicial

Este ecrã está ilustrado na figura 25.

Figura 25 - Ecrã Inicial

5.4.2. WelcomeFragment

O ecrã de boas vindas da aplicação fornece ao utilizador um pequeno texto e um botão para dar início ao desafio. Este é criado por uma imagem que representa o logotipo da aplicação, um texto e um botão que faz a aplicação avançar para o próximo fragmento. O código deste ecrã é bastante parecido ao do ecrã inicial, com a diferença que em vez de esperar um clique numa imagem de fundo aqui há um evento para quando o botão é clicado, ativando assim a transição para o próximo fragmento.

Em termos de design o código já varia, sendo que a figura 26 representa o código para o design do fragmento do ecrã de boas vindas.

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/info_screen"
    android:background="@drawable/welcomescreen">

    <ImageView
        android:id="@+id/imageView"
        android:contentDescription="Bem Vindo Imagem"
        style="@style/Info_ImageView"/>

    <TextView
        android:id="@+id/textView7"
        style="@style/Info_TextView"/>

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <com.itsector.itchallenge.Components.CustomButton
            android:id="@+id/letsGo"
            style="@style/Info_CustomButton"/>
    </RelativeLayout>

</LinearLayout>
```

Figura 26 - Código de Design do ecrã de boas vindas

O ecrã de boas vindas está representado pela figura 27.

Figura 27 - Ecrã de boas vindas

5.4.3. FormFragment

O ecrã do formulário é um ecrã que pede ao utilizador para preencher alguns dados que são fundamentais para a empresa no processo de seleção. Esses dados são:

- Nome completo;
- *Email*;
- Telemóvel;
- Grau académico;
- Curso;
- Universidade.

Tem também um campo opcional em que o utilizador pode escrever um pouco sobre ele. A figura 28 ilustra o fragmento do formulário da aplicação.

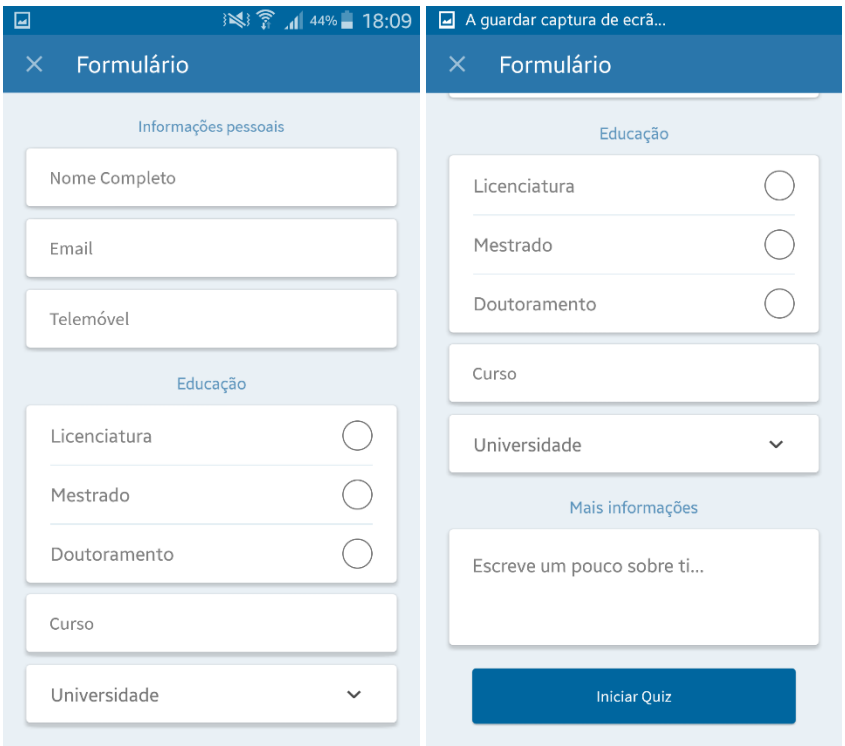


Figura 28 - Ecrã do Formulário

O formulário é em termos de design o mais complexo pois envolve várias caixas de texto, vários títulos, alguns botões e um esquema de deslizar. Este esquema faz com que se não for possível visualizar todo o formulário, o utilizador pode deslizar para visualizar todo o conteúdo. Isto torna o formulário adaptado a vários dispositivos com vários tamanhos de ecrã e várias resoluções. O código da parte de design do formulário é relativamente grande.

É ilustrado pela figura 29 a parte do título das "Informações Pessoais", o campo "Nome Completo" e o facto de se usar uma ScrollView a tal vista em que o utilizador desliza no ecrã para visualizar todo o seu conteúdo. Na figura 32 também se pode ver que se usa uma FrameLayout que parece não ter nenhuma utilidade, com o id "frame_dummy". Contudo ela é usada para permitir registar cliques fora dos campos de interesse, o que vai ajudar a fazer validações instantâneas nos campos de texto do formulário mais à frente.

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/Form_Scroll"
    style="@style/Form_ScrollView">

    <FrameLayout
        android:id="@+id/frame_dummy"
        style="@style/Form_FrameLayout">


        <LinearLayout style="@style/Form_Main_LinearLayout">

            <TextView
                android:id="@+id/form_personal_info_textview"
                style="@style/Form_TextView_Titles"
                android:text="Informações pessoais" />

            <EditText
                android:id="@+id/name"
                style="@style/Form_EditText_Fields"
                android:hint="Nome Completo"
                android:inputType="textPersonName" />

            <View
                android:id="@+id/form_name_validation_bar"
                style="@style/Form_Validation_Bars" />

            <EditText
                android:id="@+id/e_mail"
                style="@style/Form_EditText_Fields"
                android:hint="Email"
                android:inputType="textEmailAddress" />

            <View
                android:id="@+id/form_email_validation_bar"
                style="@style/Form_Validation_Bars" />
```

Figura 29 - Código do Design do formulário

Na figura 30 é possível ver a parte do código responsável por criar o design da área da “Educação”, onde são usados RadioButtons (botões no Android) dentro de um RadioGroup o que permite que o utilizador só possa escolher uma das três opções. Pois assim que escolhe uma essa fica selecionada e se trocar de escolha a anterior e desmarcada e marcada a nova.

```
<RadioGroup
    android:id="@+id/form_radiogroup_degree"
    style="@style/Form_RadioGroup">

    <RadioButton
        android:id="@+id/rb1"
        style="@style/Form_RadioButtons"
        android:text="Licenciatura" />

    <View style="@style/Form_Degree_Divider_View" />

    <RadioButton
        android:id="@+id/rb2"
        style="@style/Form_RadioButtons"
        android:text="Mestrado" />


    <View style="@style/Form_Degree_Divider_View" />

    <RadioButton
        android:id="@+id/rb3"
        style="@style/Form_RadioButtons"
        android:text="Doutoramento" />

</RadioGroup>
```

Figura 30 - Código de Design do formulário 2

Em termos de código há 3 pontos que iremos abordar:

- A validação dos campos;
- A escolha da universidade;
- A verificação final.

O formulário tem validações para os campos e mostra as mesmas ao utilizador através de barras com cores distintas nos campos.

- Campo selecionado - cor azul, figura 31;
- Campo inválido – cor vermelha, figura 32;
- Campo válido – cor verde, figura 33.

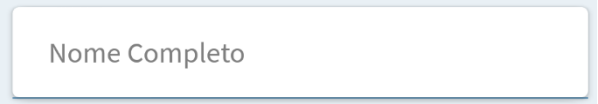


Figura 31 - Campo Selecionado

Figura 32 - Campo Inválido

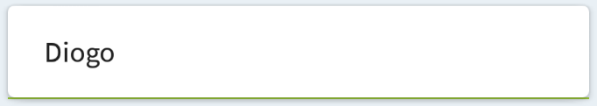


Figura 33 - Campo Válido

Para efetuar estas validações foi adicionado ao campo um método do Android neste caso para verificar quando o campo tem ou perde o foco do utilizador. Com este método é possível saber quando o utilizador está ou não a utilizar um campo. Com esta informação podemos dar ao utilizador o teclado para usar o campo e definir a linha azul por baixo quando este está a usar o campo, como podemos tirar o teclado e validar o campo quando este deixa de o usar. O código da figura 34 representa esta solução.

```
email_ET.setOnFocusChangeListener((v, hasFocus) → {
        if (hasFocus) {
            showKeyboard(v);
            bar_email.setBackgroundColor(ContextCompat.getColor(getContext(),R.color.Blue));
        } else {
            // validate here and show error if invalid and
            // set focus to this element again using requestFocus.
            if (!android.util.Patterns.EMAIL_ADDRESS.matcher(email_ET.getText()).matches()) {
                bar_email.setBackgroundColor(ContextCompat.getColor(getContext(),R.color.Red));
            } else {
                bar_email.setBackgroundColor(ContextCompat.getColor(getContext(),R.color.Green));
            }
            hideKeyboard(v);
        }
});

cel_ET.setOnFocusChangeListener((v, hasFocus) → {
        if (hasFocus) {
            showKeyboard(v);
            bar_cel.setBackgroundColor(ContextCompat.getColor(getContext(),R.color.Blue));
        } else {
            // validate here and show error if invalid and
            // set focus to this element again using requestFocus.
            if (!PHONE_PATTERN.matcher(cel_ET.getText()).matches()) {
                bar_cel.setBackgroundColor(ContextCompat.getColor(getContext(),R.color.Red));
            } else {
                bar_cel.setBackgroundColor(ContextCompat.getColor(getContext(),R.color.Green));
            }
            hideKeyboard(v);
        }
});
```

Figura 34 - Validação dos campos do formulário

A universidade é escolhida através do fragmento da universidade como iremos abordar a seguir. Quando o utilizador clica para selecionar uma universidade o fragmento guarda os dados atuais do utilizador e reencaminha para o fragmento da universidade que após o utilizador escolher uma universidade (como vamos ver mais à frente no fragmento da universidade), retorna ao formulário com a escolha. O código que representa esta ação é ilustrado pela figura 35.

```
drop.setOnClickListener((view) -> {
        //forces the focus so that the textviews can validate
        dummy.requestFocus();
        dummy.requestFocusFromTouch();
        hideKeyboard(view);
        name = name_ET.getText().toString();
        email = email_ET.getText().toString();
        cel = cel_ET.getText().toString();
        moreinfo = moreinfo_ET.getText().toString();
        course = course_ET.getText().toString();
        uni = uniTv.getText().toString();
        int corNome = (((ColorDrawable) bar_name.getBackground()).getColor());
        int corEmail = (((ColorDrawable) bar_email.getBackground()).getColor());
        int corTelefone = (((ColorDrawable) bar_cel.getBackground()).getColor());
        int corCurso = (((ColorDrawable) bar_course.getBackground()).getColor());
        getActivity().getSupportFragmentManager().beginTransaction().add(R.id.frame_container,
                UniversityFragment.newInstance(uni, name, email, cel, level, course, moreinfo,
                corNome, corEmail, corTelefone, corCurso), UniversityFragment.TAG).commit();
});
```

Figura 35 - Código para selecionar universidade no formulário

A verificação final é feita por passos. Quando o utilizador após ter preenchido o formulário clica no botão para dar inicio ao *Quiz*, a aplicação vai efetuar a verificação.

O primeiro passo que a aplicação faz é forçar os campos a perderem o foco, o que faz com que se por algum motivo algum campo não tenha sido ainda validado vai ser. Após esse primeiro passo a aplicação vai verificar se todos os campos estão válidos, neste caso verificando se a cor das validações é a verde (figura 37). Se a informação nos campos for válida a aplicação vai verificar se o utilizador definiu um grau de educação, verificando se este requisito não é nulo, visto que selecionando uma opção carrega essa opção numa variável. Depois é verificado se uma universidade foi escolhida, aqui a verificação é feita verificando se o campo não está como o pré-definido (Universidade).

Se todas estas verificações forem bem-sucedidas o utilizador é encaminhado para o fragmento das categorias.

Se em qualquer passo da verificação a mesma falhar o utilizador é encaminhado para a parte do formulário que contem o erro. Esta verificação final está ilustrada em código na figura 36.

```
button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        //forces the focus so that the textviews can validate
        dummy.requestFocus();
        dummy.requestFocusFromTouch();
        //if all validations bars are green
        if ((((ColorDrawable) bar_name.getBackground()).getColor()) ==
                (ContextCompat.getColor(getContext(),R.color.Green)) &&
                (((ColorDrawable) bar_email.getBackground()).getColor()) ==
                        (ContextCompat.getColor(getContext(),R.color.Green)) &&
                (((ColorDrawable) bar_cel.getBackground()).getColor()) ==
                        (ContextCompat.getColor(getContext(),R.color.Green)) &&
                (((ColorDrawable) bar_course.getBackground()).getColor()) ==
                        (ContextCompat.getColor(getContext(),R.color.Green))) {
            //if level not defined
            if (!level.equals("")) {
                //if university not selected
                if (uni.equals("Universidade")) {
                    f_sc.fullScroll(View.FOCUS_DOWN);
                } else {
                    name = name_ET.getText().toString();
                    email = email_ET.getText().toString();
                    cel = cel_ET.getText().toString();
                    course = course_ET.getText().toString();
                    uni = uniTv.getText().toString();
                    moreinfo = moreinfo_ET.getText().toString();
                    // -- close keyboard
                    hideKeyboard(view);
                    getActivity().getSupportFragmentManager().beginTransaction().add(
                            R.id.frame_container, CategoriesFragment.newInstance(),
                            CategoriesFragment.TAG).commit();
                }
            } else {
                f_sc.fullScroll(View.FOCUS_DOWN);
            }
        } else {
            if ((((ColorDrawable) bar_name.getBackground()).getColor()) ==
                    (ContextCompat.getColor(getContext(),R.color.Red)) ||
                    (((ColorDrawable) bar_name.getBackground()).getColor()) == 0) {
                name_ET.requestFocus();
            } else if ((((ColorDrawable) bar_email.getBackground()).getColor()) ==
                    (ContextCompat.getColor(getContext(),R.color.Red)) ||
                    (((ColorDrawable) bar_email.getBackground()).getColor()) == 0) {
                email_ET.requestFocus();
            } else if ((((ColorDrawable) bar_cel.getBackground()).getColor()) ==
                    (ContextCompat.getColor(getContext(),R.color.Red)) ||
                    (((ColorDrawable) bar_cel.getBackground()).getColor()) == 0) {
                cel_ET.requestFocus();
            } else if ((((ColorDrawable) bar_course.getBackground()).getColor()) ==
                    (ContextCompat.getColor(getContext(),R.color.Red)) ||
                    (((ColorDrawable) bar_course.getBackground()).getColor()) == 0) {
                course_ET.requestFocus();
            }
        }
    }
});
```

Figura 36 - Verificação final do formulário

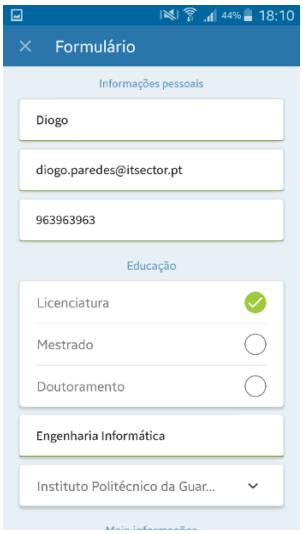


Figura 37 - Campos válidos no formulário

5.4.4. UniversityFragment

A universidade é escolhida através de um fragmento próprio. É escolhida pelo utilizador de uma lista pré-definida pela empresa e presente na aplicação como se pode ver na figura 38. A lista permite a pesquisa pelo nome da instituição como se pode ver também na figura 38.

Figura 38 - Ecrã das Universidades

Em termos de design é constituído por um campo que serve para pesquisar por nome da universidade, uma lista com universidades e um botão (em forma de seta) para voltar ao formulário caso não pretenda escolher uma universidade. O código para o design da lista é ilustrado na figura 39.

```
<ListView
    android:id="@+id/uni_listview"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="20dp"
    android:layout_marginRight="20dp"
    android:layout_marginTop="25dp"
    android:divider="@drawable/list_divider"
    android:dividerHeight="1dp"/>
```

Figura 39 - Código de Design da Lista de Universidades

A nível de código iremos analisar apenas a parte que filtra o que o utilizador escreve no campo de pesquisar e que retorna da lista de universidade o que é parecido ao filtrado. A aplicação vai carregar de um ficheiro XML (parte do ficheiro ilustrado na figura 40) as universidades, o ideal seria ir buscar a um serviço, mas no tempo útil de estágio não foi possível implementar um serviço como já referido.

Esse ficheiro é carregado num *array* de *strings* que depois passa para um *adatper* onde depois vai ser exibido na ListView (listauni). A parte do filtro é aplicada com o evento que vai verificar alterações no texto do campo de pesquisa. Este evento vai comparar a entrada no campo de pesquisa com os valores presentes no *adatper*. A figura 41 ilustra o código desta parte.

```xml
<string-array name="university_list">
    <item>Academia Militar (AM)</item>
    <item>Academia da Força Aérea (AFA)</item>
    <item>Faculdade de Ciências da Universidade do Porto (FCUP)</item>
    <item>Faculdade de Engenharia da Universidade do Porto (FEUP)</item>
    <item>Instituto Politécnico Gaya (ISPGaya)</item>
    <item>Instituto Politécnico da Guarda (IPG)</item>
    <item>Instituto Politécnico de Beja (IPBEJA)</item>
    <item>Instituto Politécnico de Bragança (IPB)</item>
    <item>Instituto Politécnico de Castelo Branco (IPCB)</item>
    <item>Instituto Politécnico de Coimbra (IPC)</item>
    <item>Instituto Politécnico de Gaya (ISPGAYA)</item>
    <item>Instituto Politécnico de Leiria (IPLEIRIA)</item>
    <item>Instituto Politécnico de Lisboa (IPL)</item>
    <item>Instituto Politécnico de Portalegre (IPPORTALEGRE)</item>
    <item>Instituto Politécnico de Santarém (IPSANTAREM)</item>
    <item>Instituto Politécnico de Setúbal (IPS)</item>
    <item>Instituto Politécnico de Tomar (IPT)</item>
    <item>Instituto Politécnico de Viana do Castelo (IPVC)</item>
    <item>Instituto Politécnico de Viseu (IPV)</item>
    <item>Instituto Politécnico do Cavado e do Ave (IPCA)</item>
    <item>Instituto Politécnico do Oeste (ISPO)</item>
    <item>Instituto Politécnico do Porto (IPP)</item>
```

Figura 40 - Parte do ficheiro das Universidades

```java
final String[] universidades = getResources().getStringArray(
        R.array.university_list);

universidadesAdapter = new ArrayAdapter<>(getActivity(),
        R.layout.list_view_custom, universidades);

listauni.setAdapter(universidadesAdapter);

uniet.addTextChangedListener(new TextWatcher() {

    @Override
    public void onTextChanged(CharSequence arg0, int arg1, int arg2, int arg3) {
        UniversityFragment.this.universidadesAdapter.getFilter().filter(arg0);
    }

    @Override
    public void beforeTextChanged(CharSequence arg0, int arg1, int arg2,
                                  int arg3) {
    }

    @Override
    public void afterTextChanged(Editable arg0) {
    }
});
```

Figura 41 - Filtro para as universidades

Após o utilizador selecionar uma universidade ou voltar a trás, clicando na seta no canto esquerdo superior do ecrã a aplicação reencaminha novamente para o formulário.

5.4.5. CategoriesFragment

O ecrã das categorias dá a escolher ao utilizador as categorias (figura 42) que quer que façam parte das perguntas. De um leque de 6 categorias o utilizador tem de escolher no mínimo 1 delas e no máximo pode escolher as 6. Estas categorias estão já definidas na aplicação sendo que se as perguntas fossem obtidas através de um serviço as categorias também o seriam, contudo não é o caso.

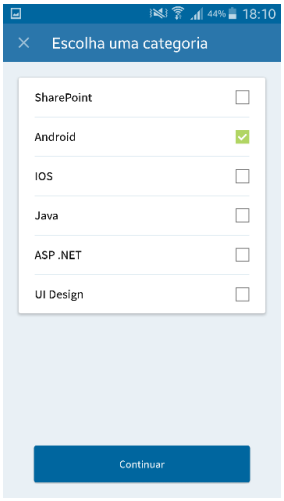


Figura 42 - Ecrã das Categorias

As categorias escolhidas são guardadas num *array* com o valor 1 se for escolhida e o valor 0 se não for escolhida, isto para depois ser verificado na criação do *Quiz* e atribuição do resultado por categorias. É também verificado se foi escolhida no mínimo uma das categorias antes de deixar avançar o utilizador, caso não tenha sido o utilizador é avisado. A figura 43 ilustra o código das últimas duas caixas de seleção e da verificação final.

```
cb5.setOnCheckedChangeListener((buttonView, isChecked) -> {
        if (isChecked) {
            cats[4] = 1;
        } else {
            cats[4] = 0;
        }
});

cb6.setOnCheckedChangeListener((buttonView, isChecked) -> {
        if (isChecked) {
            cats[5] = 1;
        } else {
            cats[5] = 0;
        }
});

final Button next = (Button) rootView.findViewById(R.id.form_next_button);
next.setTypeface(regular);
next.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        //quiz
        if (cb1.isChecked() || cb2.isChecked() || cb3.isChecked()
                || cb4.isChecked() || cb5.isChecked() || cb6.isChecked()) {
                getActivity().getSupportFragmentManager().beginTransaction().add(
                        R.id.frame_container, QuizFragment.newInstance(cats),
                        QuizFragment.TAG).commit();
        } else {
            Toast.makeText(getActivity(), "Escolha uma categoria!",
                    Toast.LENGTH_LONG).show();
        }
    }
});
```

Figura 43 - Código das Categorias

A figura 44 ilustra parte do código do design do ecrã, onde se vê que temos um esquema relativo da janela para puder ajustar os objetos em relação ao ecrã, ou seja nos cantos ou a meio por exemplo. Esse esquema é composto por outro linear que inclui as checkbox das categorias e no fim um botão, sendo que depois de cada checkbox é introduzida uma view apenas para criar a linha de separação entre categorias como se pode ver na figura 42.

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:clickable="true"
    android:background="@color/blue4">

    <LinearLayout
        android:layout_marginTop="20dp"
        style="@style/Categories_LinearLayout">

        <CheckBox
            android:id="@+id/cbShare"
            android:text="SharePoint"
            style="@style/Categories_CheckBox"/>

        <View style="@style/Form_Degree_Divider_View" />

        <CheckBox
            android:id="@+id/cbAndroid"
            android:text="Android"
            style="@style/Categories_CheckBox" />


        <View style="@style/Form_Degree_Divider_View" />

        <CheckBox
            android:id="@+id/cbIOS"
            android:text="IOS"
            style="@style/Categories_CheckBox" />

        <View style="@style/Form_Degree_Divider_View" />

        <CheckBox
            android:id="@+id/cbJava"
            android:text="Java"
            style="@style/Categories_CheckBox" />

        <View style="@style/Form_Degree_Divider_View" />
```

Figura 44 - Código do Design das Categorias

5.4.6. QuizFragment

O ecrã do *Quiz* é onde o utilizador vai responder ao "desafio". Depois de escolher a categoria ou as categorias o utilizador pode continuar e a aplicação vai criar o *Quiz* a partir das categorias escolhidas. São sempre 20 perguntas por isso a divisão da quantidade atribuída a cada categoria depende de quantas são escolhidas. Aqui houve um problema

na fase de desenvolvimento que era quando o utilizador escolhia 3 ou as 6 categorias pois a divisão para 20 não dá número inteiro. A solução foi:

- No caso de o utilizador escolher 3 categorias: nas duas primeiras a partir do leque de perguntas de cada uma serão selecionadas 6 (aleatoriamente), sendo que a ultima categoria escolhida, ou seja a 3ª, irá preencher as restantes perguntas do *Quiz* até perfazer as 20, isto é 8.
- No caso de o utilizador escolher 6 categorias: nas cinco primeiras a partir do leque de perguntas de cada uma serão selecionadas 3 (aleatoriamente), sendo que a ultima categoria neste caso é sempre a UI Design, como tal irá preencher 5 perguntas do *Quiz*, perfazendo assim as 20.

As perguntas e respostas são obtidas a partir de ficheiros que estão criados por categoria, sendo que as perguntas escolhidas para fazer parte do *Quiz* são aleatórias evitando assim a repetição correndo a aplicação várias vezes. A ordem das respostas é também diferente para a mesma pergunta, de forma a evitar qualquer tipo de memorização por parte do utilizador.

Para se entender melhor a parte da criação do *Quiz* que é referida em cima vamos dividir por partes. As perguntas são carregadas num *adatper* de forma aleatória, do número de perguntas existentes para aquela categoria as que vão ser carregadas são escolhidas aleatoriamente. O *adapter* é carregado com perguntas consoantes as categorias escolhidas. Ele é corrido num ciclo (20 vezes) para em cada iteração do mesmo carregar num *array* a pergunta e a resposta certa, e noutro *array* à parte as 4 respostas possíveis (1 certa e 3 erradas). As respostas para uma determinada pergunta são baralhadas nesta fase, em que é criado um *array* novo que vai buscar ao das respostas as 4 para aquela pergunta específica e as vai dispor de forma aleatória. A figura 45 ilustra esta parte excluindo o carregamento do *adapter*.

```
int x = 0;
//for randomize the order of the answers
List<Integer> intList = new ArrayList<>();
for(int aL = 0; aL < 4; aL++) intList.add(aL);
//list that is store answers in the "for"
List<String> resp = new ArrayList<>();
for (int qA = 0; qA < questionsAdapter.getCount(); qA++) {
    //get the question
    String per = questionsAdapter.getItem(qA);
    //for each question there are 4 answers
    for (int aA = x; aA < x + 4; aA++) {
        //add the answers to the resp array
        resp.add(answersAdapter.getItem(aA));
    }
    //add the object Question
    list.add(new Classes.Question(per, resp.get(x), WhatCat.get(qA)));
    //shuffle the intList of 0,1,2,3
    Collections.shuffle(intList);
    //random the answers
    listPerg.add(new Classes.Order(resp.get(x + intList.get(0)),
            resp.get(x + intList.get(1))
            ,resp.get(x + intList.get(2)),
            resp.get(x + intList.get(3))));
    //increment x for next set of answers of the next question
    x = x + 4;
}
```

Figura 45 - Código do Quiz

Para a ordem das perguntas ser diferente cada vez que a aplicação é corrida é criado um *array* com números de 0 a 19 que depois vai ser baralhado. Para cada pergunta é escolhida a corresponde à posição desse *array*, como se pode ver na figura 46.

```
//sets the actual question number and the total to the user
QuestText = ("Pergunta" + " " + i + "/20");
numQuestion.setText(QuestText);
//generates a lsit of integers from 0 to 19 to shuffle
// and define the order of the question loaded previous
for (int k = 0; k <= 19; ++k) alreadyTaken.add(k);
//shuffle the list
Collections.shuffle(alreadyTaken);
//gets the first question using the shuffled list
numberRandom = alreadyTaken.get(i - 1);
questionstv.setTypeface(myTypeface);
//shows the question to the user
questionstv.setText(list.get(numberRandom).getPergunta());
//stores the correct answer to give score to user if we choses the correct
correctAnswer = list.get(numberRandom).getRespostaCerta();
```

Figura 46 - Ordem aleatório do Quiz

A aplicação sabe sempre qual é a resposta certa para a pergunta em questão graças à classe Question, ilustrada na figura 47. Com esta classe a aplicação sabe que para uma determinada pergunta, a resposta é a indicada na classe através do método getRespostaCerta e que a categoria dessa pergunta é obtida pelo método getCat. A classe também é usada na figura 46 quando se vai buscar uma pergunta aleatória, em que corremos o *array* das perguntas e vamos numa posição indicada pelo *array* dos números aleatórios buscar essa pergunta.

```
public static class Question {

    private final String pergunta;
    private final String RespostaCerta;
    private final int cat;

    public Question(String pergunta, String RespostaCerta, int cat) {
        this.pergunta = pergunta;
        this.RespostaCerta = RespostaCerta;
        this.cat=cat;
    }

    public String getPergunta() { return this.pergunta; }

    public String getRespostaCerta() { return this.RespostaCerta; }

    public int getCat() { return this.cat; }

}
```

Figura 47 - Classe da Pergunta

A aplicação só permite ao utilizador avançar após escolher uma das quatro respostas possíveis. No entanto esta permite ainda voltar à pergunta anterior, mas não só à imediatamente anterior. O utilizador pode, portanto, estar na última pergunta e regressar à primeira mantendo a seleção já efetuada em cada uma das perguntas por onde passou. No caso de o utilizador avançar até à última pergunta as escolhas já feitas são mantidas se o mesmo não as alterar. Na figura 48 está representado o ecrã do *Quiz*.

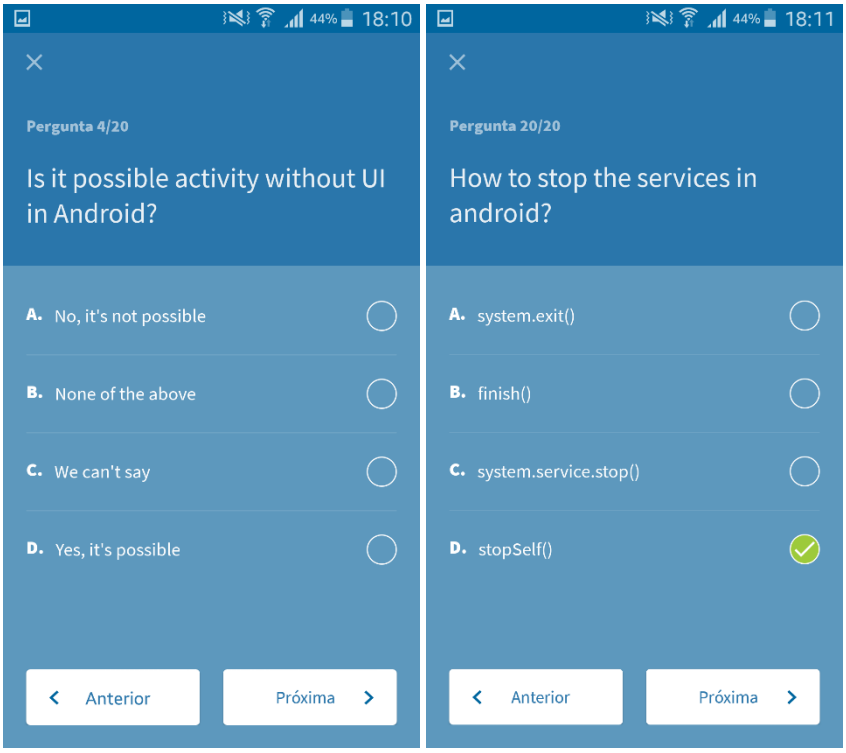


Figura 48 - Ecrã do Quiz

A pontuação é atribuída a nível geral e por categoria como se pode ver na figura 49.

```
if (yourAnswer.equals(correctAnswer)) {
    //user score increment in general and on the specified category
    z = z + 1;
    if(list.get(numberRandom).getCat()==0)score_list.get(0).setScore(score_list.get(0).getScore()+1);
    else if(list.get(numberRandom).getCat()==1)score_list.get(1).setScore(score_list.get(1).getScore()+1);
    else if(list.get(numberRandom).getCat()==2)score_list.get(2).setScore(score_list.get(2).getScore()+1);
    else if(list.get(numberRandom).getCat()==3)score_list.get(3).setScore(score_list.get(3).getScore()+1);
    else if(list.get(numberRandom).getCat()==4)score_list.get(4).setScore(score_list.get(4).getScore()+1);
    else if(list.get(numberRandom).getCat()==5)score_list.get(5).setScore(score_list.get(5).getScore()+1);
    listscore.add("Acertou");
} else {
    listscore.add("Errou");
}
```

Figura 49 - Pontuação no Quiz

Após todas as perguntas terem sido respondidas, o utilizador é encaminhado para o fragmento do resultado onde lhe é apresentado o resultado obtido.

5.4.7. ScoreFragment

O ecrã do resultado é responsável por mostrar ao utilizador depois de este concluir o *Quiz* o seu resultado e dar a hipótese ao utilizador de enviar os dados ou cancelar. O ecrã do resultado está ilustrado na figura 50.

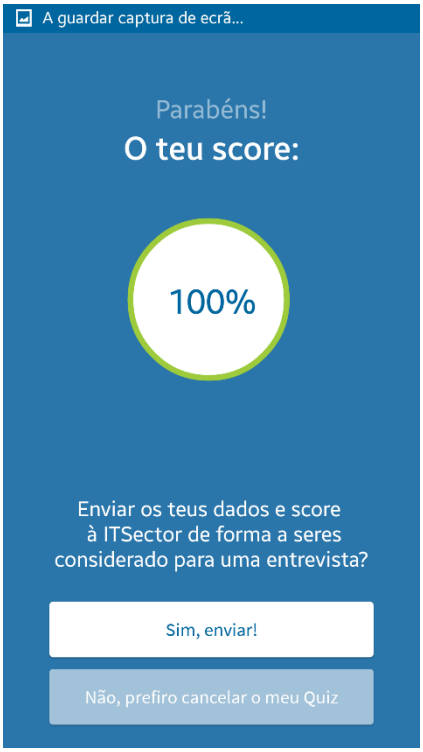


Figura 50 - Ecrã do resultado

Neste caso na figura 50 o utilizador teve o resultado máximo. Este resultado é obtido através de um rácio entre o número de respostas corretas multiplicado por 100 e o número total de perguntas (que é sempre 20). Isto dará uma percentagem que depois é usada pela aplicação para criar uma barra de progresso circular que dá o efeito visto na figura 50. A figura 51 ilustra o código para o ecrã do resultado e a figura 52 parte do código para a componente da barra de progresso circular.

```
CircularProgressBar c1 = (CircularProgressBar) rootView.findViewById(R.id.c1);

((MainActivity) getActivity()).setTitle("");
((MainActivity) getActivity()).noActionBar();

int cper = getArguments().getInt(ZKEY);
cperc = (cper * 100) / 20;

//define the color of the score circle (outside circle)
c1.setProgressColor(ContextCompat.getColor(getContext(),R.color.Green));
//defines the color for the text that represents the percentage of correct answers
c1.setTextColor(ContextCompat.getColor(getContext(),R.color.blue1));
c1.setProgressWidth();//width of the progress bar
//defines the "progress" of the bar from 0 to the score of the user
c1.setProgress((cper * 100) / 20);//the score from 0 to 20 x 100
// dividing for the number of the questions to give us the percentage
```

Figura 51 - Código para criar o resultado

```
private void drawOutlineArc(Canvas canvas) {
    //define the diameter
    final int diameter = Math.min(mViewWidth, mViewHeight) - (mStrokeWidth * 2);
    //defines measures for the outer circle
    //animation for the efect of progress
    final RectF outerOval = new RectF(mStrokeWidth, mStrokeWidth, diameter, diameter);
    //defines measures for the inner circle
    final RectF Oval = new RectF(mStrokeWidth, mStrokeWidth, diameter, diameter);
    //color and attributes to the background
    Paint testPaint = new Paint(Paint.ANTI_ALIAS_FLAG);
    testPaint.setStrokeWidth(50);
    testPaint.setColor(ContextCompat.getColor(getContext(),android.R.color.white));
    //draw the arc
    canvas.drawArc(Oval, -90, 360, false, testPaint);
    //color and attributes from the progress arc (green)
    mPaint.setColor(mProgressColor);
    mPaint.setStrokeWidth(mStrokeWidth);
    mPaint.setAntiAlias(true);
    boolean mRoundedCorners = true;
    mPaint.setStrokeCap(Paint.Cap.ROUND);
    mPaint.setStyle(Paint.Style.STROKE);
    //draw the arc
    canvas.drawArc(outerOval, -90, mSweepAngle, false, mPaint);
}

private void drawText(Canvas canvas) {
    //defines color and attributes for the text
    mPaint.setTextSize(Math.min(mViewWidth, mViewHeight) / 5f);
    mPaint.setTextAlign(Paint.Align.CENTER);
    mPaint.setStrokeWidth(0);
    mPaint.setColor(mTextColor);
    //to center the text
    int xPos = (canvas.getWidth() / 2);
    int yPos = (int) ((canvas.getHeight() / 2)-((mPaint.descent() + mPaint.ascent())/2));
    //draw the text
    canvas.drawText(calcProgressFromSweepAngle(mSweepAngle) + "%", xPos, yPos, mPaint);
}
```

Figura 52 - Barra de Progresso Circular

A aplicação fornece também ao utilizador para além do resultado duas opções:

- Enviar os dados para a empresa;
- Cancelar o "desafio", podendo o mesmo voltar a tentar de novo se assim o entender.

Em ambos os casos os botões têm código para cumprir um requisito da equipa de design que era o da sensação de o botão estar a ser clicado, que neste caso é alcançado carregando a fonte do texto do botão, como se pode ver na figura 53.

```
//for changing the type of text on the button according to the button state
sim_enviar.setOnTouchListener(new View.OnTouchListener() {
    @Override
    public boolean onTouch(View v, MotionEvent event) {

        switch (event.getAction()) {
            //when button pressed
            case MotionEvent.ACTION_DOWN:
                sim_enviar.setTypeface(bold);
                break;
            //when button released
            case MotionEvent.ACTION_UP:
                sim_enviar.setTypeface(regular);
                break;
        }
        return false;
    }
});
```

Figura 53 - Código do Botão

No caso de o utilizador não enviar os dados a aplicação volta ao ecrã inicial (SplashFragment).

No caso de o utilizador enviar os dados, podem acontecer dois cenários:

- A aplicação não consegue enviar os dados e deixa o utilizador voltar a tentar, isto pode acontecer quando a ligação à internet do dispositivo está com problemas (ErrorSendingFragment);
- A aplicação consegue enviar os dados e avisa o utilizador do mesmo (SuccessfullFragment).

5.4.8. EmailFragment

O fragmento do *email* não tem ecrã visível para o utilizador pois ele é usado para enviar o *email* para a empresa e para o candidato, e consoante houver falha ou sucesso no envio reencaminhar para os respetivos fragmentos. Sendo assim ele apenas tem um fundo azul para ajudar na transição. Quando a aplicação envia os dados estes não vão só para a empresa, como já referido em cima é enviado um *email* para a empresa e outro para o candidato.

Para a empresa ele é estruturado com os dados obtidos pela aplicação. A aplicação usa os dados do formulário em conjunto com as categorias escolhidas e as respostas certas para fornecer à empresa uma informação completa do candidato.

Exemplos de *emails* que a empresa pode receber estão ilustrados pelas figuras 54, 55 e 56, sendo que a diferença entre elas é o facto das categorias escolhidas pelo utilizador serem diferentes. Dada essa diferença conclui-se que quanto maior o número de categorias escolhidas pelo utilizador, mais fácil será para a empresa observar as aptidões do candidato. O nome não aparece no texto do *email* pois é incluído no assunto.

Email	Número	Curso	Grau Académico	Universidade	Mais Informações	Pontuação Geral
diogo.paredes@itsector.pt	963963963	Engenharia Informática	Licenciatura	Instituto Politécnico da Guarda (IPG)		100%

Categorias: Android

Categoria	Certas	Totais
Android	20	20

Figura 54 - Email que a empresa recebe

Email	Número	Curso	Grau Académico	Universidade	Mais Informações	Pontuação Geral
diogo.paredes@itsector.pt	963963963	Engenharia Informática	Licenciatura	Instituto Politécnico da Guarda (IPG)		90%

Categorias: Android Java

Categoria	Certas	Totais
Android	10	10
Java	8	10

Figura 55 - Email que a empresa recebe (2)

Email	Número	Curso	Grau Académico	Universidade	Mais Informações	Pontuação Geral
diogo.paredes@itsector.pt	963963963	Engenharia Informática	Licenciatura	Instituto Politécnico da Guarda (IPG)		65%

Categorias: SharePoint Android Java

Categoria	Certas	Totais
SharePoint	3	6
Android	6	6
Java	4	8

Figura 56 - Email que a empresa recebe (3)

Este modelo pode ser posteriormente alterado facilmente pois a estrutura dele é baseada em duas tabelas HTML, este código HTML não será apresentado posteriormente.

Para enviar o *email* é usado SMTP da Google, mais especificamente o SMTP do Gmail, para isto é definida uma sessão e criada uma mensagem que depois é enviada para um destinatário através desse servidor. A Google obriga a ser usada uma sessão válida de uma conta Gmail para se puder usar o serviço deles. Um pedaço do código responsável por criar a sessão, criar a mensagem e enviar está ilustrado na figura 57.

Onde podemos ver que quando criamos a mensagem podemos definir o campo do remetente, neste caso "ITSector-ITChallenge". Foi usado o hífen pois o SMTP não permite espaços a separar palavras. No campo onde temos o texto "EMAIL_TO" é definido o *email* da empresa para onde deve ser enviado o *email* com os dados do *Quiz*. Este *email* não é divulgado por questões de confidencialidade. Para criar a sessão temos vários dados genéricos e outros fornecidos pela Google para se usar o serviço SMTP dela.

Por fim como já referido em cima temos o uso de uma conta válida do serviço de *email* Gmail da Google para puder criar uma sessão no servidor de SMTP da Google. Esta conta é usada nos campos "EMAIL" e "PASS", onde são introduzidos o *email* e respetiva senha da conta. Estes dados são guardados por questões de segurança em variáveis definidas no ficheiro build.gradle evitando assim o acesso às mesmas.

```
private void sendMail(String subject, String body) {
    Session session = createSessionObject();
    try {
        Message message = createMessage(subject, body,
                session);
        new SendMailTask().execute(message);
    } catch (UnsupportedEncodingException | MessagingException e) {
        e.printStackTrace();
    }
}

private Message createMessage(String subject,
                              String body, Session session)
        throws MessagingException,
        UnsupportedEncodingException {
    Message message = new MimeMessage(session);
    message.setFrom(new InternetAddress("ITSector-ITChallenge"));
    message.setRecipients(Message.RecipientType.TO,
            InternetAddress.parse("EMAIL_TO"));
    message.setSubject(subject);
    message.setContent(body, "text/html; charset=utf-8");
    return message;
}

private Session createSessionObject() {
    Properties props = new Properties();
    props.put("mail.smtp.host", "smtp.gmail.com");
    props.put("mail.smtp.socketFactory.port", "465");
    props.put("mail.smtp.socketFactory.class",
            "javax.net.ssl.SSLSocketFactory");
    props.put("mail.smtp.auth", "true");
    props.put("mail.smtp.port", "465");

    Session.getInstance(props);
    return Session.getInstance(props, getPasswordAuthentication() -> {
            return new PasswordAuthentication("EMAIL", "PASS");
    });
}
```

Figura 57 - SMTP email

Para o candidato a aplicação envia um *email* no qual o remetente é a empresa e o assunto é o resultado no desafio, como se pode ver na figura 58.

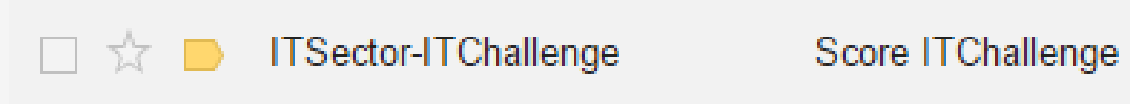


Figura 58 - Email que o utilizador recebe

O corpo do *email* para o candidato é constituído por uma imagem criada pela aplicação em HTML com o logótipo da mesma, partes de texto e uma imagem que irá representar a percentagem obtida no *Quiz*, como está ilustrado na figura 59.

Figura 59 - Texto do email que o utilizador recebe

Esta parte teve alguns problemas a nível de implementação pois alguns serviços de *email* como o Gmail da Google e o Outlook da Microsoft bloqueiam imagens enviadas, tanto como anexo como parte do texto. E isto impedia o envio do logótipo da aplicação, tal como a imagem representativa da percentagem.

A solução foi alcançada após algum tempo em pesquisas e foi criada um código HTML, que o serviço de *email* interpreta como texto, contudo este levaria uma imagem embutida no meio do HTML através de um MIME. Isto é possível pois a aplicação usa um serviço de *email* SMTP.

Na figura 60 podemos ver que o *email* é enviado com um texto em formato HTML, contudo no meio do texto há duas TAGs especiais para indicar images. Estas TAGs são a “” e “”, que fazem com que no

código HTML sejam inseridas as imagens por meio de um MIME. Este MIME carrega as imagens a partir das localizações das mesmas.

```
MimeMessage message = new MimeMessage(session);
message.setSubject("Score ITChallenge");
message.setFrom(new InternetAddress("DoNotReply@itsector.pt",
        "ITSector-ITChallenge"));
message.addRecipient(Message.RecipientType.TO,
        new InternetAddress(FormFragment.email));//FormFragment.email

//
// This HTML mail have to 2 part, the BODY and the embedded image
//
MimeMultipart multipart = new MimeMultipart("related");
// first part  (the html)
BodyPart messageBodyPart = new MimeBodyPart();
String htmlText = "<table style='background-color:#2A76AB' >" +
        "<tr>" +
        "<th>" +
        "<img src='cid:imagelogo' />" +
        "</th>" +
        "</tr>" +
          ...
        "<tr>" +
        "<th>" +
        "<img src='cid:image' />" +
        "</th>" +
        "</tr>" +
        "</table>";

messageBodyPart.setContent(htmlText, "text/html; charset=utf-8");
// add it
multipart.addBodyPart(messageBodyPart);
// second part (the image)
messageBodyPart = new MimeBodyPart();
DataSource fds = new FileDataSource(getURL(score));
messageBodyPart.setDataHandler(new DataHandler(fds));
messageBodyPart.setHeader("Content-ID", "<image>");

// add it
multipart.addBodyPart(messageBodyPart);
//second image
BodyPart messageBodyPart2 = new MimeBodyPart();
DataSource fdo = new FileDataSource
        ("/data/data/com.example.itsector.itchallenge/files/logo_...");
messageBodyPart2.setDataHandler(new DataHandler(fdo));
messageBodyPart2.setHeader("Content-ID", "<imagelogo>");

multipart.addBodyPart(messageBodyPart2);
// put everything together
message.setContent(multipart);

new SendMailTask().execute(message);
```

Figura 60 - MIME

Para complementar a solução e visto que o Android não permite enviar imagens diretamente da aplicação para um serviço de *email*, foi criado um pedaço de código que grava a imagem a enviar na memória interna do dispositivo (figura 61). Podendo esta ser então acedida e usada para o *email*, sendo que após o *email* ser enviado esse mesmo pedaço de código é responsável por apagar essa imagem da memória interna do dispositivo (figura 62).

```
private void saveToInternalStorage(Bitmap bitmapImage, String name) throws IOException {
    try {
        // Use the compress method on the Bitmap object to write image to
        // the OutputStream
        FileOutputStream fos = getContext().openFileOutput(name, Context.MODE_PRIVATE);

        // Writing the bitmap to the output stream
        bitmapImage.compress(Bitmap.CompressFormat.PNG, 100, fos);
        fos.close();
        //path where images are stored
        //Log.e("test",""+getContext().getFilesDir());

    } catch (Exception e) {
        Log.e("saveToInternalStorage()", e.getMessage());
    }
}
```

Figura 61 - Guardar Imagem

```
File f1 = new File("/data/data/com.example.itsector.itchallenge/files/", "score_" + score + ".png");
f1.delete();
File f2 = new File("/data/data/com.example.itsector.itchallenge/files/", "logo_email.png");
f2.delete();
```

Figura 62 - Apagar Imagem

Resumindo, assim que o utilizador escolhe a opção de enviar os dados, o pedaço de código irá escolher dependendo da percentagem obtida a imagem correta de um leque de 21 imagens. Este leque é constituído por todos os resultados possíveis, ou seja, no mínimo o utilizador tem um resultado de 0% e no máximo 100% com intervalos de 5%, uma vez que são sempre 20 perguntas apenas os múltiplos de 5 poderão ser um resultado. Vai guarda-la na memória do dispositivo assim como o logótipo da aplicação. Após estes serem enviados o mesmo pedaço de código irá ao sítio da memória onde os guardou e irá apagá-los. Isto torna-se viável devido ao facto de só existirem 21 possibilidades de resultados.

5.4.9. SuccessfullFragment

O fragmento de sucesso serve apenas para informar o utilizador que o resultado foi submetido com sucesso para a empresa, e se este quiser pode repetir o "desafio". O código é apenas para apresentar uma imagem de sucesso, um texto informativo do mesmo e um botão (para se o utilizador quiser repetir o *Quiz*), a figura 63 ilustra o ecrã do fragmento. Sendo que a mensagem "Email enviado com sucesso" é temporária.

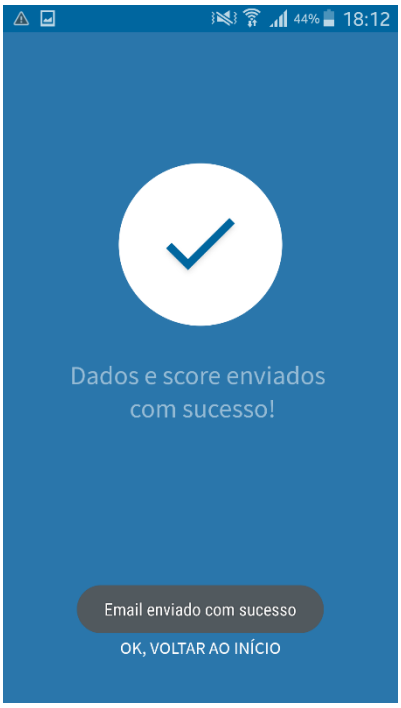


Figura 63 - Ecrã Sucesso

A figura 64 ilustra o código para obter o design. Código que vai de encontro ao referido em cima, ou seja, uma imagem, um texto e um botão que neste caso é criado sendo texto dentro de um esquema em que se pode clicar.

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/blue2"
    android:orientation="vertical">

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="120dp"
        android:contentDescription="Sucesso"
        android:src="@drawable/ic_success" />

    <TextView
        android:id="@+id/textView7"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginTop="30dp"
        android:text="Dados e score enviados  com sucesso!"
        android:textAlignment="center"
        android:textColor="#80FFFFFF"
        android:textSize="24sp" />

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <TextView
            android:id="@+id/success_startagain_textview"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentBottom="true"
            android:layout_centerHorizontal="true"
            android:layout_marginBottom="40dp"
            android:text="OK, voltar ao início"
            android:textAllCaps="true"
            android:textColor="#FFFFFF"
            android:textSize="16sp" />
    </RelativeLayout>
</LinearLayout>
```

Figura 64 - Ecrã Sucesso Código Design

5.4.10. ErrorSendingFragment

O fragmento de erro no envio serve apenas para informar o utilizador que o resultado não foi submetido com sucesso para a empresa, e permite ao utilizador tentar novamente o envio ou então clicando na cruz no canto superior esquerdo cancelar e voltar ao inicio. A figura 65 ilustra o ecrã do fragmento.

Figura 65 - Ecrã de erro no envio

Em termos de código é similar ao fragmento de sucesso. O botão de cancelar tem um evento associado tal como o texto "Voltar a Tentar" que funciona como um botão e ambos encaminham para fragmentos diferentes.

O design é constituído por duas imagens e duas caixas de texto. Sendo as imagens a cruz de cancelar e a imagem de erro, e os textos são o informativo e o "Voltar a Tentar" que como já referido neste caso atua como botão. A figura 66 ilustra o código do design deste ecrã.

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    style="@style/NotSent_Main_LinearLayout">

    <ImageView
        android:id="@+id/quit_image"
        android:contentDescription="Sair"
        style="@style/NotSent_Quit_Button"/>

    <ImageView
        android:id="@+id/imageView"
        android:contentDescription="Erro"
        style="@style/NotSent_Error_Image" />

    <TextView
        android:id="@+id/textView7"
        style="@style/NotSent_TextView" />

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

    <TextView
        android:id="@+id/error_tryagain_textview"
        android:textAllCaps="true"
        style="@style/NotSent_Button_TryAgain"/>

    </RelativeLayout>
</LinearLayout>
```

Figura 66 - Ecrã de Erro no envio Código Design

6. Resultados

Ao longo do desenvolvimento a aplicação sofreu algumas alterações e ajustes, tanto a nível gráfico como a nível de otimizações no código, sendo as otimizações após a finalização da mesma. Uma das grandes mudanças foi a nível estético, pois inicialmente a aplicação tinha o nome de "QuizIT" e perto do fim do desenvolvimento a aplicação mudou de nome para o nome atual e final da mesma "ITChallenge". Isto implicou a criação por parte da equipa de design de novas imagens com o novo nome, após esse trabalho realizado a aplicação foi atualizada e concluída.

Após aprovação por parte da empresa, a aplicação foi finalizada e está neste momento a funcionar.

Cumpre os requisitos estabelecidos pela empresa e alcança os objetivos da mesma, pois incorpora uma *interface* de fácil utilização, é uma aplicação leve e rápida, fornece uma forma eficiente de registar os dados dos candidatos. Cria o *Quiz* consoante as categorias escolhidas pelos candidatos, as perguntas e respostas sofrem uma ordem aleatória de um leque geral da própria categoria o que torna as memorizações difíceis e evita a criação de um *Quiz* repetido. Finalmente trata a informação obtida e envia de forma estruturada e clara para o departamento de recursos humanos da empresa.

A nível de design também ficou de acordo com o esquema que a equipa de design definiu no inicio do desenvolvimento. As orientações foram seguidas a nível estético. O que fez com que o desenho projetado pela equipa de design fosse alcançado pela aplicação, não se notando assim diferença visível.

7. Conclusões

A aplicação ficou a funcionar e a cumprir os requisitos para a qual foi criada, contudo há sempre aspetos que podem ser melhorados e funcionalidades que podem ser adicionadas.

Alguns melhoramentos que poderão ser efetuados são:

- Obter a lista de perguntas e respostas através de uma *API*. (Ficou por implementar pois a empresa não decidiu no tempo do estágio como preferia atualizar as perguntas e respostas, e como estas seriam carregadas para a aplicação);

- Fornecer os dados dos candidatos que submeterem a informação para um base de dados da empresa em vez de enviar um *email*;

- Desenvolver uma versão da mesma aplicação para dispositivos iOS, podendo assim ser disponibilizada para utilizadores que tenham dispositivos da Apple.

- Permitir ao utilizador puder juntar como anexo o seu currículo diretamente na aplicação, evitando assim que caso seja selecionado para uma entrevista a empresa tenha que pedir o currículo.

Estes melhoramentos não afetam o estado atual da aplicação, contudo podem trazer vantagens para a mesma.

Bibliografia

- Manual do Colaborador ITsector, 2015
- Santos, António; *Outsourcing e Flexibilidade*; 1998; Texto Editora - http://www.pmelink.pt/article/pmelink_public/EC/0,1655,1005_5051-3_41097--View_429,00.html
- Schwaber, K; *Scrum Development Process*, OOPSLA'95 Workshop on Business Object Design and Implementation. Springer-Verlag. (1995)
- Schwaber, K. e Beedle, M. *Agile Software Development with SCRUM*, Prentice-Hall, (2002)
- Michel, Murillo. 2007; *Tipos de recrutamento e sua importância para uma gestão adequada de pessoas aplicadas a empresas*. REVISTA CIENTÍFICA ELETÔNICA DE ADMINISTRAÇÃO. 13: 7.

Anexos

Anexo I

Este anexo contém a primeira parte da 1º Guideline da Equipa de Design.

QuizIT
USER INTERFACE GUIDELINES - PART 1

---

1. Typography

Lato Sans Pro Regular — AaBbCc 0123#

Source Sans Pro Black — **AaBbCc 0123#**

---

2. Colors

- Blue 1 (strong) - #00669f
- Blue 2 (not so strong) - #2a76ab
- Blue 4 (form background & active text field line) - #e9f0f5
- Blue 5 (quiz answers background & quiz titles) - #5d95bd
- Dark - #4b4b4b // DO NOT use black #000000
- Red - #ce2129
- Green - #9dcb3c

Anexo II

Este anexo contém a segunda parte da 1º Guideline da Equipa de Design.

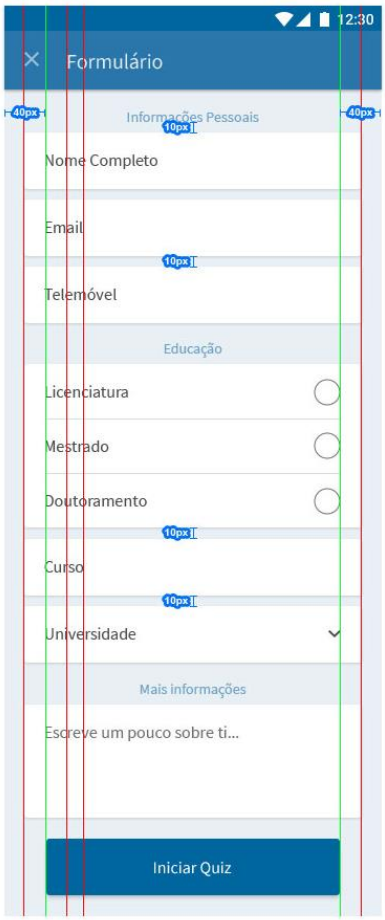
3. Grid

The grid alows the UI elements to be easily aligned and organized within the app screen.

The main grid (red lines) are provided by Google's Material Design Guidelines. Please take a look at:
https://www.google.com/design/spec/layout/metrics-keylines.html#metrics-keylines-keylines-spacing

The app grid (green lines) aims to support the main one by providing extra guidance.

Always compare the live app to the provided design. If an elements alignment does not seem to be correct, please get in touch with the designer.



Anexo III

Este anexo contém a terceira parte da 1º Guideline da Equipa de Design.

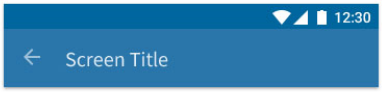
---

4. Status and App Bar

Both elements should respect Googles Material Design Guidelines.
Please take a look at:
https://www.google.com/design/spec/layout/structure.html#structure-toolbars

Nav Bar backgound:
Blue 1 - #00669f

App Bar background color:
Blue 2 - #2a76ab



---

5. Top and Bottom margins

Top:
Between the App Bar and the next UI element of the screen there will (almost) always be a margin of 20px.

Bottom:
Between the bottom of the app screen and and the last UI element there should also be a margin of 20px. If the content is higher than the device screen and scroll is needed, this margin will only be visible after the very last UI element.

Anexo IV

Este anexo contém a quarta parte da 1º Guideline da Equipa de Design.

---

6. Form

Screen background color:
Blue 4 - #e9f0f5

6.1 Groups titles

FONT
Source Sans Pro
14 px
#5d95bd

6.2 Input Field

SHAPE
Corner radius: 3 px
Fill: #FFFFFF / rgb(255,255,255)

DROP SHADOW
#000000 / rgb(0,0,0)
Opacity: 15 % (multiply)
Angle: 90 / Offset: 1 px
Blur Size: 3 px

Hint text
FONT
Source Sans Pro
16 px
#4B4B4B / rgb(75,75,75)
Opacity: 80 %

INDICATOR
#73a4c6

Not ok text
FONT
Source Sans Pro
16 px
#4B4B4B / rgb(75,75,75)

LINE
Fill: #ce2129
1px

Active text field
FONT
Source Sans Pro
16 px
#4B4B4B / rgb(75,75,75)

LINE
Fill: #73A4C6
1px

Disabled field
FONT
Source Sans Pro
16 px
#4B4B4B / rgb(75,75,75)
Opacity: 30 %

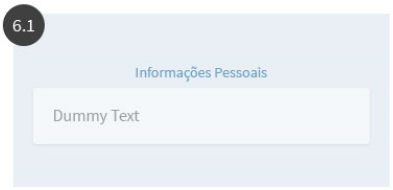
Ok text
FONT
Source Sans Pro
16 px
#4B4B4B / rgb(75,75,75)

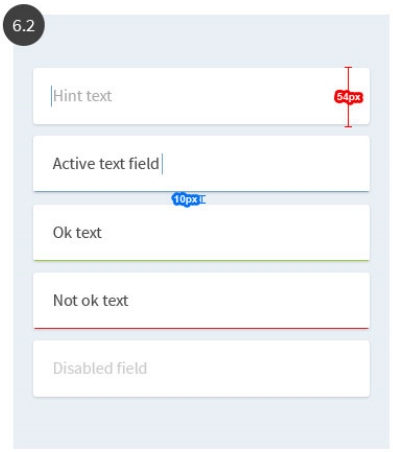
LINE
Fill: #9dcb3c
1px

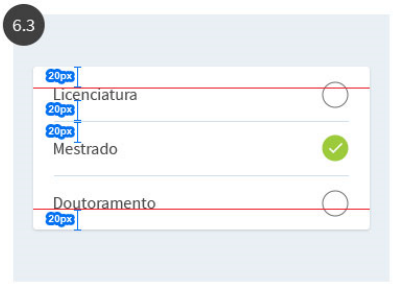
6.3. Radio buttons group

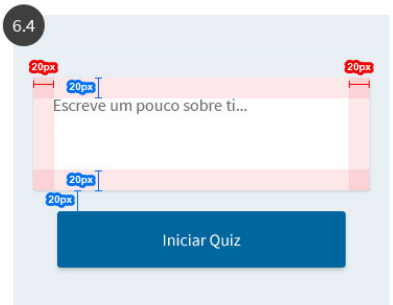
6.4. Open text field

FONT
Source Sans Pro
16 px
#4B4B4B / rgb(75,75,75)
Opacity: 80 %









Anexo V

Este anexo contém a quinta parte da 1º Guideline da Equipa de Design.

---

7. Universities List

Screen background color: Blue 4 - #e9f0f5

LIST ITEMS FONT
Source Sans Pro
16 px
Dark - #4B4B4B / rgb(75,75,75)

How it works:
When the user clicks on the "Universidade" btn (with a dropdown-like arrow) in the form screen, a new screen wil apppear conaining a search field and the universities list.

The list will update as letters are written on the search field. That will make it easier for the user to select an item.

---

8. Quiz

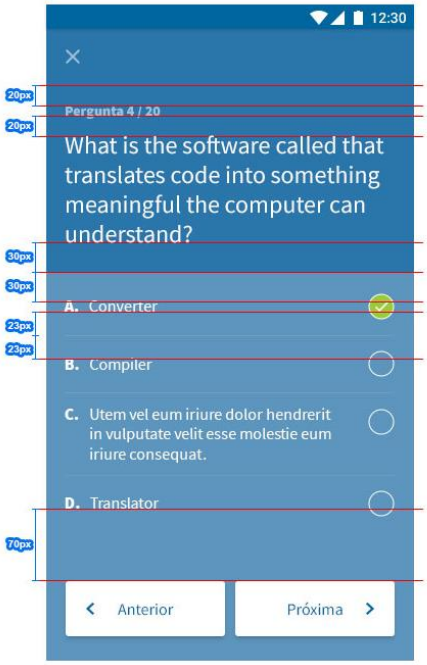
Question background color: Blue 2 - #2a76ab
Answers background color: Blue 5 - #5d95bd

QUESTION COUNTER FONT
Source Sans Pro Black
14 px
#A2C2D9 / rgb(162,194,217)

QUESTION FONT
Source Sans Pro
24 px

LETTER INDICATOR FOR EACH ANSWER FONT
Source Sans Pro Black
16 px

ANSWER FONT
Source Sans Pro Regular
16 px



Anexo VI

Este anexo contém a primeira parte da 2º Guideline da Equipa de Design.

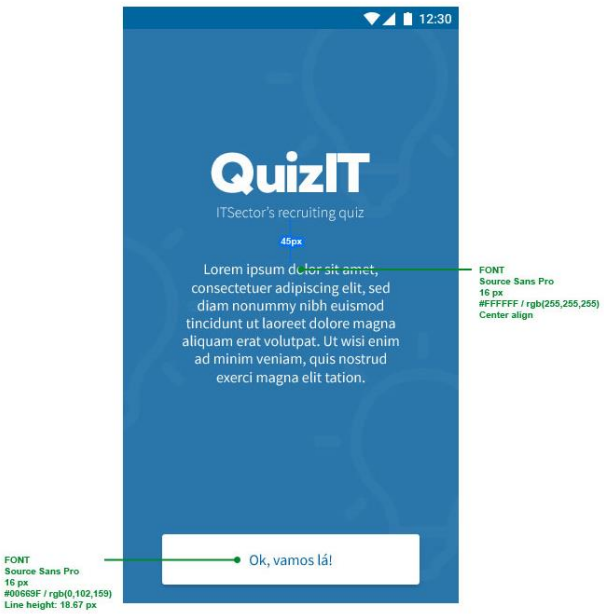
QuizIT
USER INTERFACE GUIDELINES - PART 2

---

9. Splash Screen

---

10. Info Screen

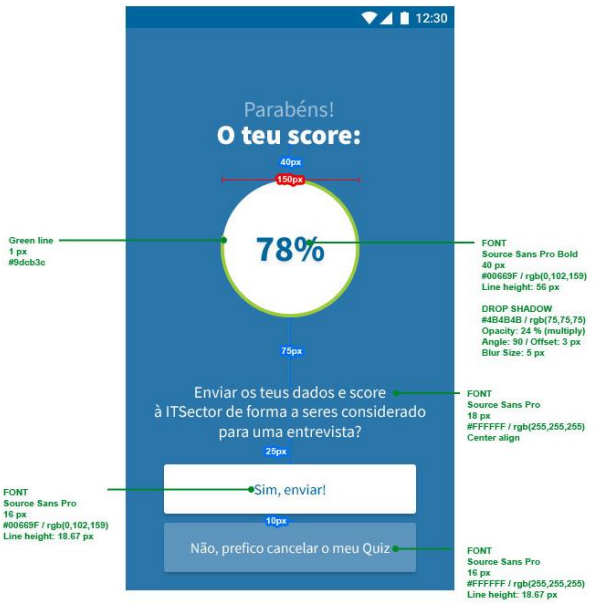


Anexo VII

Este anexo contém a segunda parte da 2º Guideline da Equipa de Design.

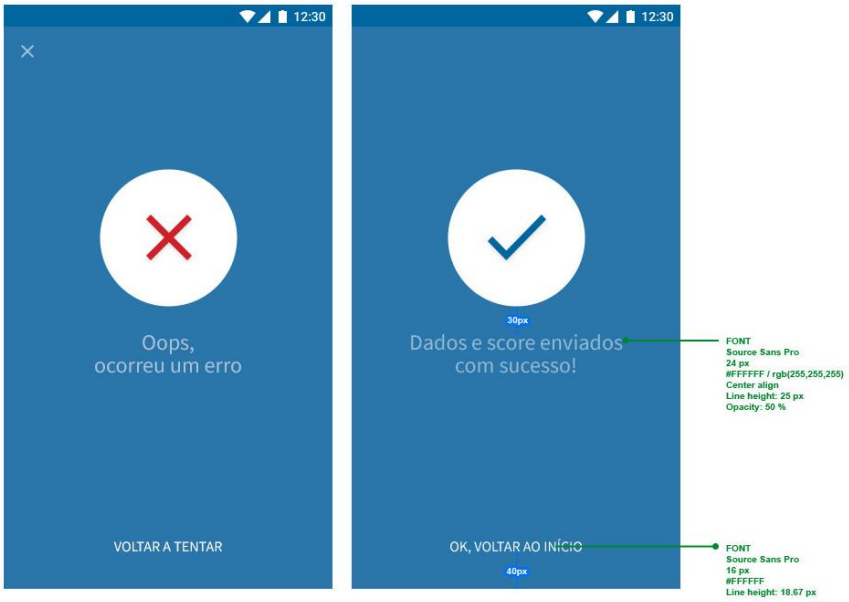
---

11. Score Screen



---

12. Success & error screen



Anexo VIII

Este anexo contém a terceira parte da 2º Guideline da Equipa de Design.

13. Buttons

For all buttons:
SHAPE
Corner radius: 3 px

DROP SHADOW
#000000
Opacity: 24 % (multiply)
Angle: 90 / Offset: 3 px
Blur Size: 5 px (Spread: 6 %)

FONT
Source Sans Pro
16 px
#FFFFFF OR #00669f
Line height: 18.67 px

On Hover, buttons will keep their original appearance except for the font which will go Source Sans Pro Bold

Button 1 bg color:
#00669f

Button 2 & 3 bg color:
#ffffff

Button 4 bg color:
#a2c2d9