IPG Politécnico da Guarda
Escola Superior de Tecnologia e Gestão

RELATÓRIO DE ESTÁGIO

Curso Técnico Superior Profissional
em Desenvolvimento de Aplicações Informáticas

Miguel Afonso Pereira

agosto | 2021

Relatório de Estágio Curricular

Altran/Capgemini Engineering

TDAI – 2º Ano – 2º Semestre – 2021

Miguel Afonso Pereira
Nº 1704271

03/08/2021

Ficha de Identificação

Estudante:

Nome: Miguel Afonso Pereira

Número: 1704271

Email: miguelap97@gmail.com

Curso: TesP Desenvolvimento de Aplicações informáticas

Instituição: Instituto Politécnico da Guarda - Escola Superior de Tecnologia e Gestão

Local de Estágio:

Empresa: Altran/Capgemini Engineering

Localização: Centro de Negócios e Serviços, Praça Amália Rodrigues, 6230-350, Fundão

Website: capgemini-engineering.com

Informação de estágio adicional:

Supervisor na empresa: José Gouveia (Team Leader)

Professor orientador: Luís Figueiredo

Duração: 750 (setecentas e cinquenta) horas

Data de Inicio: 19/03/2021

Data de conclusão: 03/08/2021

Agradecimentos

Agradeço aos meus pais por todo o seu apoio.

Agradeço ao colega de turma e estágio, João Ferreira. Fazer o estágio com alguém conhecido facilitou todo o processo.

À turma de DAI, obrigado por tudo. Somos poucos, mas bons!

Agradeço ao pessoal na equipa do projeto A2C2F na Capgemini Engineering, todos contribuíram para um ambiente de trabalho fenomenal. A ajuda na minha integração e onboarding neste projeto foram em grande parte facilitados pelo apoio destes colegas:

- Carla, Ana, Vasco, Luís, Salvado e João.

Ao coordenador por parte da empresa, José Gouveia, obrigado por facilitar todo o processo de estágio.

Agradeço, também, ao professor coordenador Luís Figueiredo por responder a todas as minhas questões sobre o processo de estágio.

Obrigado ao pessoal do GESP, nomeadamente, José Martins, pela forma eficiente com que tratou de todas as matérias relacionadas com este estágio.

Finalmente, gostaria de agradecer à Capgemini Engineering e ao IPG pela oportunidade de estagiar nesta prestigiosa empresa de perfil internacional.

Resumo

Este relatório é uma compilação de todas as tarefas, responsabilidades e objetivos de um estágio profissional, para finalização do Curso Técnico Superior de Desenvolvimento de Aplicações informáticas.

Devido à atual situação pandémica, e à política de segurança da Capgemini Engineering, todo o estágio foi feito online, através de um computador portátil fornecido pela empresa.

O estágio recorreu desde 19/03/2021 até 3/08/2021, na empresa Altran, atualmente chamada de Capgemini Engineering.

O foco do estágio foi um projeto de I&D, com o objetivo de criar uma aplicação em Android Automotive, na linguagem de Java e posteriormente Kotlin. Utilizámos o Android Studio para o desenvolvimento da app. Colocar a correr esta aplicação no sistema Polestar 2 da Volvo era também um dos objetivos.

Fui integrado numa equipa que chegou aos 7 (sete) membros, e dividida em duas partes: equipa front-end (onde fui integrado) e equipa back-end.

A função principal desta aplicação é o controlo e segurança de quem tem acesso ao veículo, através de reconhecimento facial. Um utilizador pode registar uma conta de utilizador e adicionar um registo facial para autenticação. Após este registo, cada vez que entrar no veículo, é pedida a autenticação através de reconhecimento facial.

O objetivo secundário do estágio foi a minha integração numa equipa de desenvolvimento, de forma a desenvolver e melhorar atributos profissionais, como o trabalho em equipa.

Todas as ferramentas e nomenclatura interna estão em língua inglesa, tal como os termos utilizados no dia a dia. Desta forma, irei utilizar também esses mesmos termos neste relatório, de forma a proteger a autenticidade do mesmo. Todos estes termos estão traduzidos na lista de abreviaturas/acrónimos/termos (página 6), de forma a facilitar a leitura.

Índice

Lista de abreviaturas/acrónimos/termos

IPG – Instituto Politécnico da Guarda

TeSP – Curso Técnico Superior Profissional

TDAI – TeSP DAI

DAI – Desenvolvimento de Aplicações Informáticas

I&D – investigação e desenvolvimento

App – application (software) - aplicação

Front-end – parte dianteira – desenvolvimento da parte gráfica e lógica da mesma, a parte da aplicação com que o utilizador interage

Back-end – parte traseira – desenvolvimento da parte da aplicação que processa informação e comunica com um servidor, a parte da aplicação com que o utilizador não interage diretamente

BD - Base de Dados

IDE - Integrated Development Environment – ambiente de desenvolvimento integrado

API - Application Programming Interface – Interface de Programação de aplicações

Teams – Microsoft Teams (software) – programa para comunicação digital/organização de equipas

Infotainment – portmanteau de “information” e “entertainment” – Computador de bordo de um veículo, onde o utilizador pode aceder a informações do veículo, ligação internet, GPS, etc.

CE – Capgemini Engineering

String – sequência de caracteres

Hash – Hashing executa uma transformação unilateral de uma senha, transformando-a numa String. Como isto é um processo unilateral, não é possível reverter esta String para a senha original [4]

SotA – State-of-the-Art – Estado da arte – estado atual da tecnologia do tópico determinado

Machine Learning – aprendizagem automática – subclasse de engenharia e ciência computacional. Evoluiu do estudo de reconhecimento de padrões

Deep Learning – aprendizagem profunda - ramo de aprendizagem automática, baseado num conjunto de algoritmos que tentam modelar abstrações de alto nível de dados usando um grafo profundo com várias camadas de processamento

Neural Network – rede neuronal artificial – modelo computacional inspirado pelo sistema nervoso central de animais, "neurónios" interligados

Estrutura do Relatório

Este relatório está dividido em cinco componentes principais:

- Capítulo 1 - Introdução, onde é feita uma breve introdução ao projeto e empresa;

- Capítulo 2 – Métodos e ferramentas de trabalho. É aqui feita uma descrição geral das ferramentas principais usadas durante o projeto, bem como os métodos de trabalho internos na Capgemini Engineering;

- Capítulo 3 – Investigação. Investigação feita durante o começo do projeto;

- Capítulo 4 – Trabalho desenvolvido, onde é descrito o trabalho feito durante o estágio. É importante notar que não são as *user stories*, pois muitas delas são componentes de objetivos descritos no capítulo 4;

- Capítulo 5 - Conclusão. Conclusão final do projeto, com algumas observações pessoais.

Capítulo 1 – Introdução

Introdução

De forma a completar o TeSP de Desenvolvimento de aplicações informáticas, é necessário fazer um estágio curricular numa das empresas acolhedoras. Entre todas as empresas apresentadas, decidi escolher a Altran (nome alterado a meio do estágio para "Capgemini Engineering") por múltiplas razões:

Em primeiro lugar, uma empresa de perfil internacional como a Altran, muito bem estabelecida e criada há décadas, tem um método de trabalho interno refinado, bem testado e eficiente. Este método de trabalho era algo que me interessava aprender, não apenas para a integração na empresa, mas também porque achei ser uma ferramenta bastante valiosa no meu futuro profissional.

Em segundo lugar, a Altran é uma empresa bem conhecida por ofertas de emprego após estágios curriculares. Ter uma oportunidade de trabalho após o estágio era algo bastante atrativo.

Após uma entrevista (Microsoft Teams), foi-me indicado que, devido à situação pandémica, todo o estágio iria ser feito online, através de um computador portátil fornecido pela empresa. Fui também informado que iria ser integrado num projeto de I&D, com o nome interno "A2C2F", especificamente trabalhando com Android Studio para criação de uma aplicação de reconhecimento facial. Esta iria correr em android Automotive, diretamente no sistema de infotainment do veículo.

A programação em si seria na linguagem Java, posteriormente Kotlin.

Também tive uma sessão onde me foi indicado o funcionamento interno da empresa, caso tenha necessidade de abrir "tickets" para instalação de software, bem como as ferramentas usadas, como por exemplo o GitLab.

O método de trabalho praticado seria o SCRUM, um método derivado da metodologia AGILE.

Sobre a empresa

No fim de 2020, a empresa Capgemini adquiriu o grupo Altran. A meio do estágio curricular, a Capgemini reformulou o departamento de engenharia e I&D da Altran, criando a Capgemini Engineering.

A Capgemini Engineering (CE) é parte integrante do Grupo Capgemini. O grupo Capgemini é líder global em consultoria, transformação digital, tecnologia e serviços de engenharia. O Grupo está na vanguarda da inovação para abordar toda a gama de oportunidades dos clientes no mundo em evolução da nuvem, digital e plataformas. Conta com mais de 52.000 engenheiros e cientistas, em mais de 30 países.[1]

Os setores-chave da CE são: aeronáutica, automotiva, ferrovias, comunicações, energia, ciências da vida, semicondutores, software e internet, espaço sideral e defesa e produtos de consumo. [2]

Em Portugal, a Capgemini Engineering tem gabinetes em Lisboa, no Porto e no Fundão.

O grupo gerou receitas de € 15.848 milhões, + 12,2% com a aquisição da Altran [3].

Figura 1 - Logotipo Grupo Capgemini

Organização interna

Internamente, a equipa está dividida em duas partes, equipa front-end e equipa back-end. Existem depois as posições na hierarquia organizacional.

Comecemos pelo chamado Team Leader. É normalmente o individuo com mais experiência no projeto. Organiza a equipa internamente, seja a divisão de tarefas, resolução de problemas ou orçamento. As demonstrações, no nosso caso, foram feitas para apresentar ao Team Leader.

O Scrum master é quem ajuda a organizar a equipa de forma a cumprir com o método Scrum. Lidera reuniões e pontos a seres discutidos, e ajuda a equipa no que for necessário.

Finalmente, estão os desenvolvedores. Estes são conhecidos pelos títulos de engenheiros (caso tenham uma licenciatura) ou técnicos. São os indivíduos nas linhas da frente, programam as *user stories* definidas, fazem testes unitários e resolvem problemas de código. A minha posição na empresa durante este estágio era designada de técnico.

Objetivos do projeto

O objetivo deste projeto, com o nome interno A2C2F, é o desenvolvimento de uma aplicação de segurança para veículos. Consiste em treinar um algoritmo de inteligência artificial para controlo de utilizadores, programar uma interface de utilizador que permita o registo de novas contas de utilizadores e o registo de caras para cada uma dessas contas. Apagar reconhecimentos faciais e contas de utilizador também é uma função necessária para a aplicação.

De forma a proteger os dados dos utilizadores, as contas têm que ser guardadas num servidor externo. Neste servidor serão guardados os dados de acesso dos utilizadores, palavras-chave transformadas com funções *hash*, de forma a não guardar as palavras-chave em si. Também serão guardados dados para a identificação do veículo.

Esta aplicação precisa de correr ou no sistema de infotainment do veículo em si, ou caso isso não seja possível com a tecnologia atual, num smartphone externo, com uma câmara.

Figura 2 - Aplicação a correr em sala virtual

Plano de Estágio/trabalho

O plano de estágio, definido pelo orientador da parte da empresa, tem três objetivos principais:

1. Investigação do sistema operativo Android;
    - Funcionamento interno e diferenças entre Android Auto e Android Automotive;
2. Programação/aprendizagem nas linguagens de programação Java/Kotlin;
    - Desenvolvimento em Android Studio;
3. Desenvolvimento de algoritmos e modelos para reconhecimento facial;
    - Modelos baseados em modelos pré-treinador TensorFlow;

Capítulo 2 – Métodos e Ferramentas de Trabalho

SCRUM

O método utilizado na Capgemini Engineering é intitulado de SCRUM, um método derivado da metodologia AGILE. É utilizado o website/ferramenta Jira para controlo de *User Stories*.

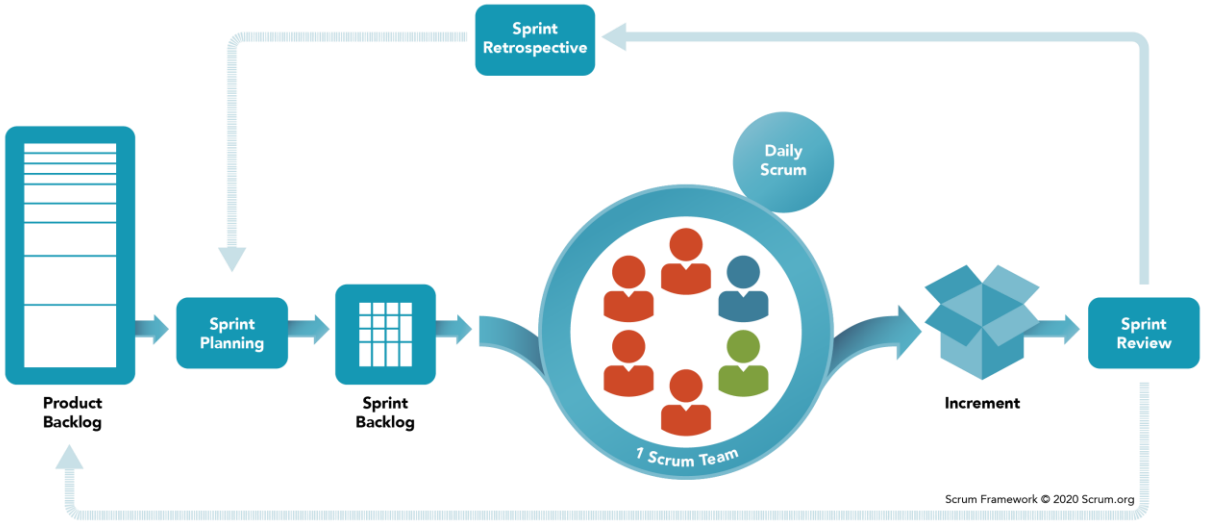

Figura 3 - Workflow método Scrum

O método Scrum substitui uma abordagem algorítmica programada por uma heurística, com respeito pelas pessoas e auto-organização para lidar com a imprevisibilidade e resolver problemas complexos. Indivíduos, equipas e organizações a gerar valor por meio de soluções adaptativas para problemas complexos [5].

Para chegar a este objetivo, o Scrum divide o trabalho em ciclos, chamados sprints. Um sprint tem duração de duas semanas, dez dias uteis.

Ao fazer esta divisão, a equipa adapta um método de trabalho muito mais flexível do que outros métodos ou metodologias de desenvolvimento.

De acordo com este método, todos os dias às 10 (dez) da manhã a equipa faz uma reunião para discutir o trabalho feito no dia anterior, o trabalho planeado para o dia atual e os possíveis bloqueios encontrados até ao momento, bem como maneiras de os resolver. Esta reunião é liderada por um Scrum Master, que tira notas e organiza as dailies.

Ao fim de cada ciclo (*sprint*) é feita uma demonstração do trabalho feito nesse período, bem como uma reunião para planear os passos seguintes.

O projeto fica dividido em *Epics* (características principais da aplicação) e estes subdivididos em *User Stories* (características dos *Epics*). Isto é feito de forma a tornar o desenvolvimento mais modular, onde múltiplos programadores podem trabalhar em características diferentes simultaneamente. Um *Epic* pode ser trabalhado ao longo de múltiplos *Sprints*, mas uma *Story* é normalmente trabalhada apenas num único Sprint, caso não haja bloqueios para a sua conclusão.

Depois de uma *User Story* ser definida, a equipa prevê e vota o quão difícil essa *Story* será para completar. Esta votação serve para atribuir os *story points* a cada *user story*. Este valor diverge entre 1, 2, 3, 5 ou 8. Caso a equipa vote num valor acima de 8 pontos, por exemplo 13, essa *user story* é repartida e reformulada em múltiplas *stories*, de forma a facilitar o desenvolvimento.

Uma *story*, depois de criada e com atribuição de pontos, vai para o chamado backlog de trabalho, onde se encontram todas as *user stories* ainda não completas.

Durante a fase intitulada de sprint planning, são decididas quais as *stories* que a equipa irá trabalhar durante esse sprint.

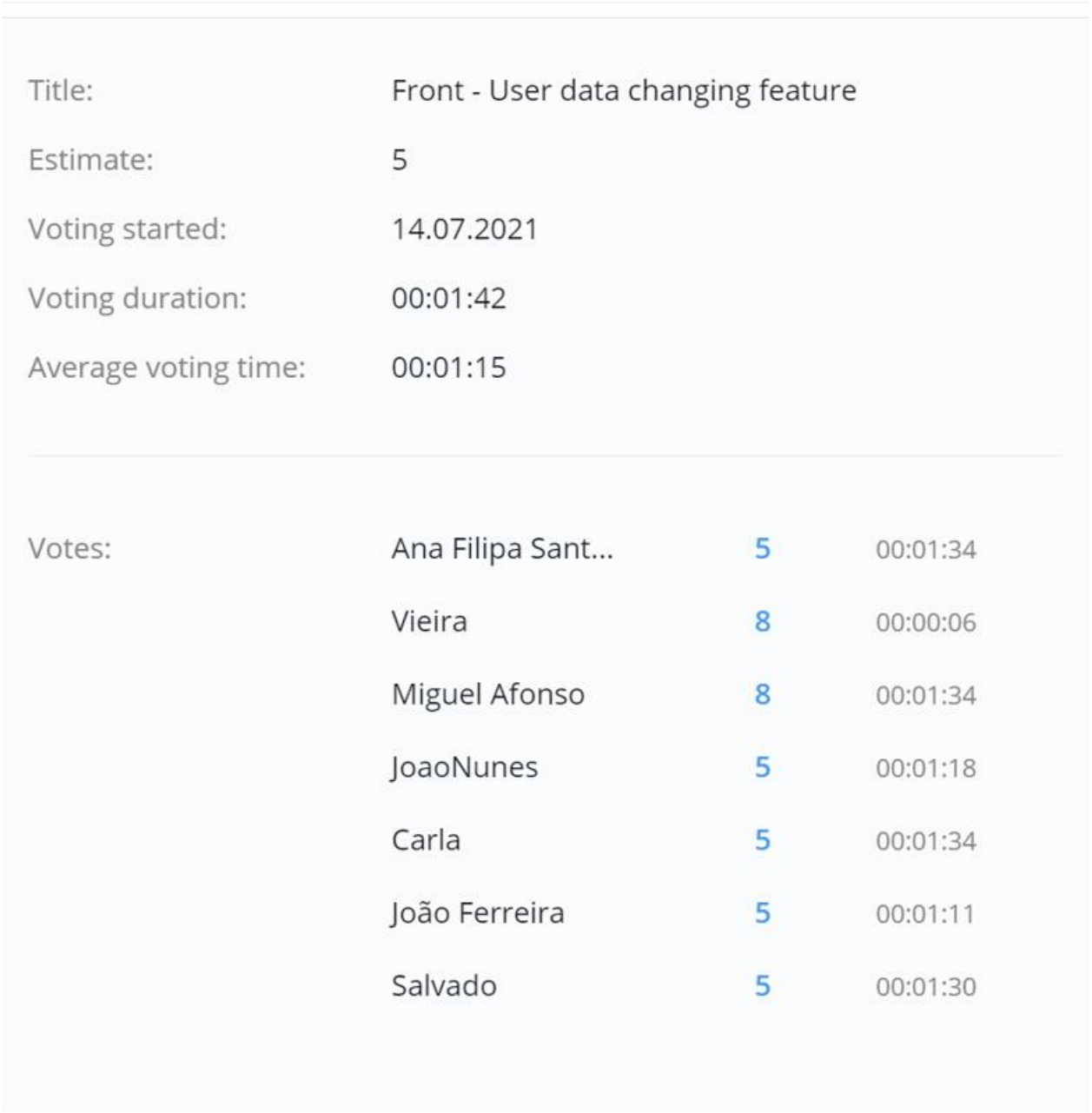

Figura 4 – Exemplo de votação - website PlanitPoker

É importante notar que, normalmente, o cliente comunica as características que quer implementadas na aplicação ao gestor do projeto, que depois as traduz para as *User Stories.* No entanto, no nosso caso em particular foi a equipa em si que as produziu e introduziu no Jira, pois era um projeto de I&D.

O Jira faz parte de uma família de produtos projetados para ajudar equipas de todos os tipos a gerir o trabalho. Originalmente, foi projetado como uma ferramenta para encontrar problemas no software (bugs). Hoje em dia, o Jira evoluiu para uma ferramenta poderosa de gestão de trabalho para todos os tipos de casos de uso, desde gestão de requisitos e casos de teste até o desenvolvimento Agile de software. [6]

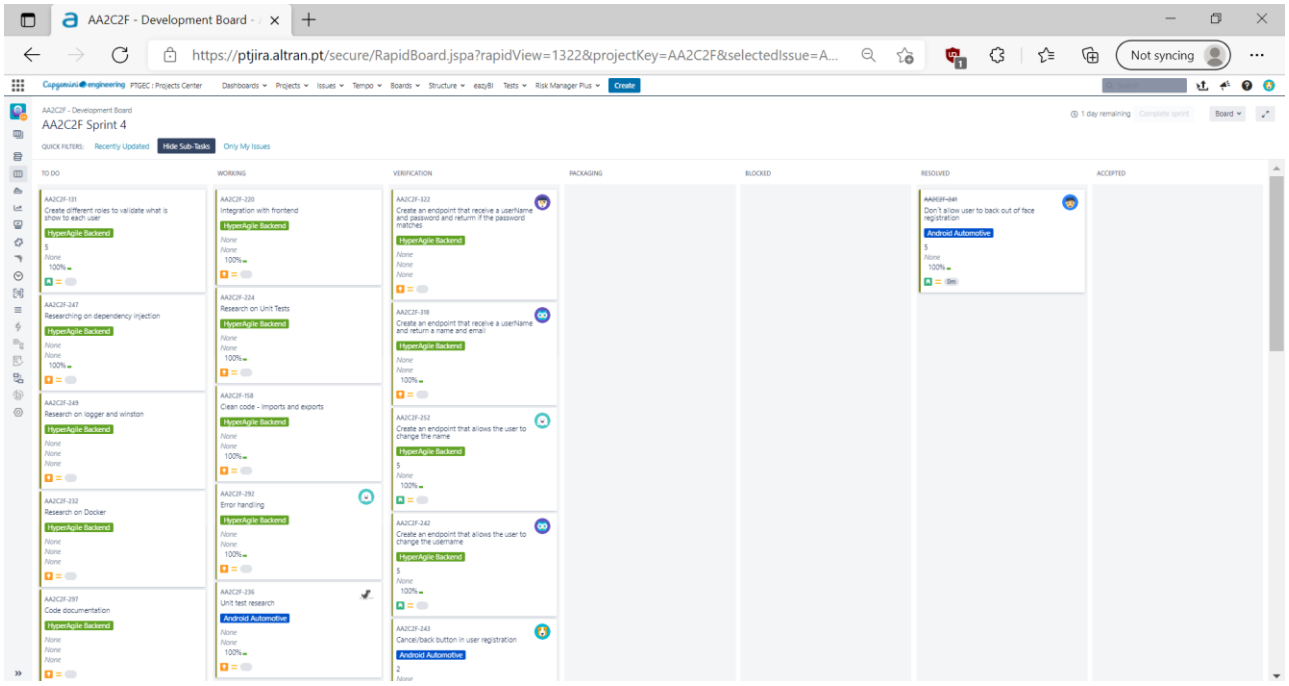

Figura 4 – Página inicial - Jira

Android Studio

O Android Studio é o Ambiente de Desenvolvimento Integrado (IDE) oficial para o desenvolvimento de aplicações Android, baseado no IntelliJ IDEA. Android Studio oferece ainda mais recursos que aumentam a produtividade, como:

- Um sistema de compilação baseado em Gradle flexível;
- Um emulador rápido e rico em recursos;
- Um ambiente unificado onde é possível desenvolver aplicações para todos os dispositivos Android;
- Aplicar alterações para enviar alterações de código e recursos para o aplicativo em execução sem reiniciá-lo;
- Modelos de código e integração com o GitHub para ajudá-lo a criar recursos de aplicativo comuns e importar códigos de amostra;
- Extensas ferramentas e estruturas de teste;
- Ferramentas Lint para detetar desempenho, usabilidade, compatibilidade de versão e outros problemas;
- Suporte a C++ e NDK;
- Suporte integrado para Google Cloud Platform, facilitando a integração do Google Cloud Messaging e do App Engine. [7]

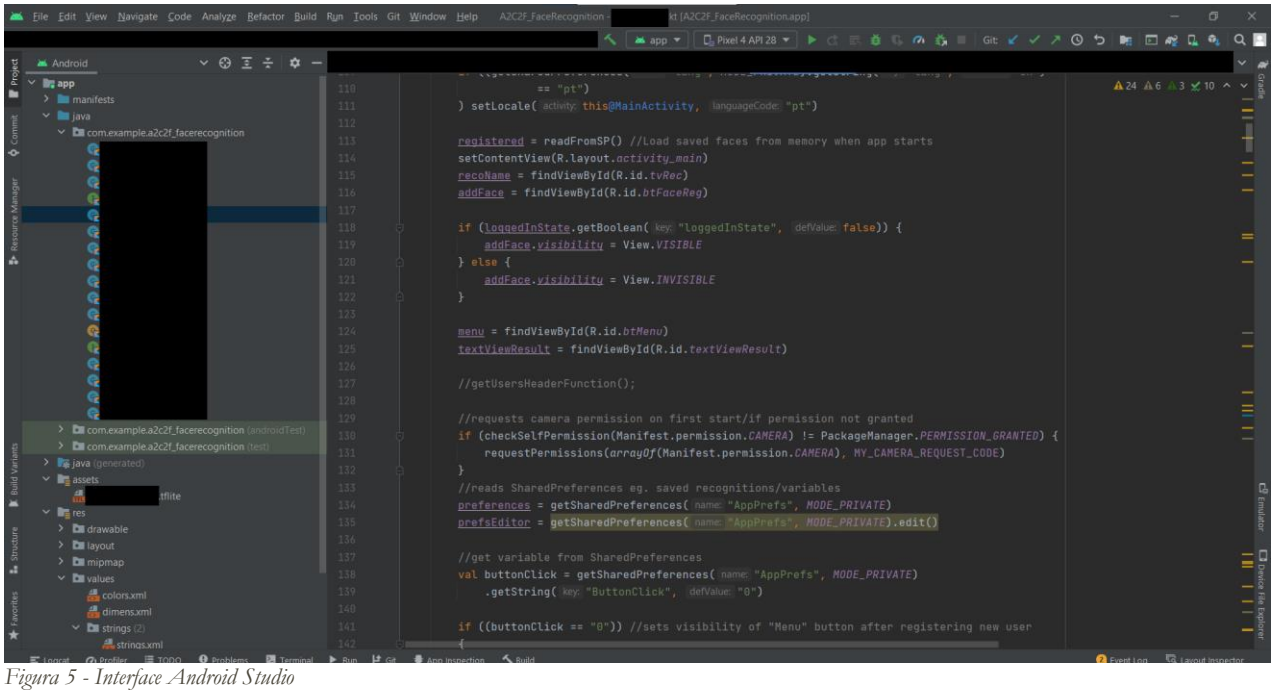

Figura 5 - Interface Android Studio

GitLab – Controlo de versão

A plataforma designada para o controlo de versões é o GitLab.

GitLab é a plataforma DevOps aberta, entregue como uma única aplicação. Isto torna o GitLab único e cria um fluxo de trabalho de software mais simplificado e flexível. [8]

O projeto A2C2F tem múltiplos ramos. O ramo “master” é o ramo para entrega final. Isto é, apenas a versão do software final é enviada para este ramo. Existem depois dois ramos de desenvolvimento distintos, o ramo para o desenvolvimento front-end e o ramo para back-end.

O fluxo de trabalho corre da seguinte maneira: um desenvolvedor escolhe uma das *user stories* que lhe formam atribuídas neste sprint e retira a sua designação do Jira, por exemplo “AA2C2F-222”. Depois é criado um ramo para esta *story* com esta mesma designação. Isto pode ser feito diretamente no Android Studio, assumindo que está interligado ao GitLab. Segue-se, depois, o desenvolvimento do software.

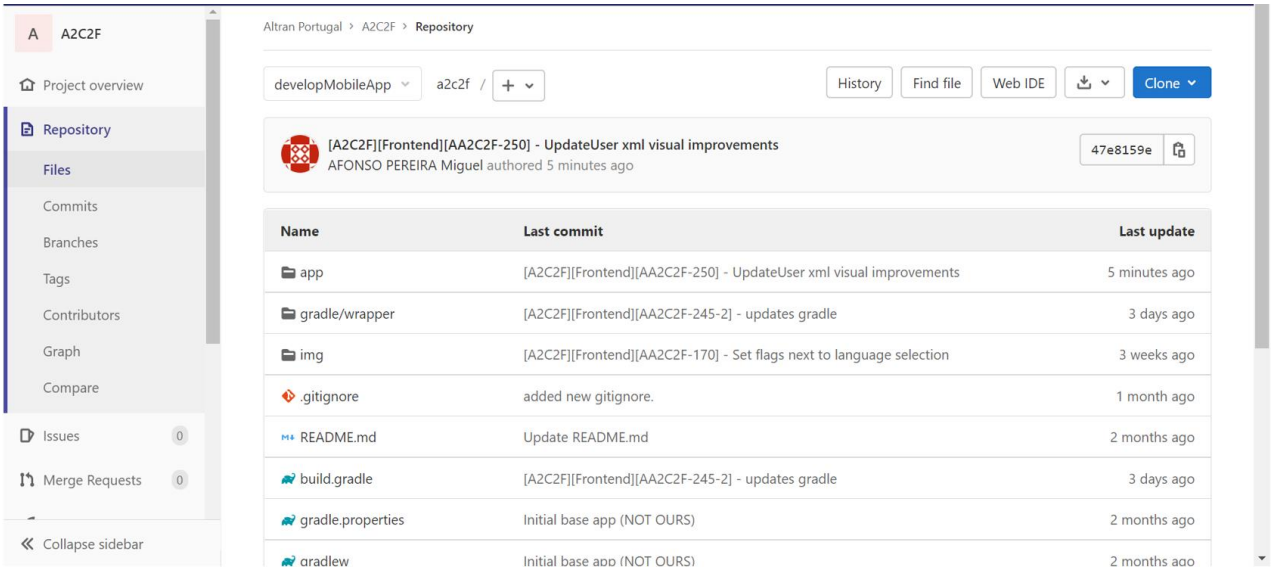

Figura 6 - Ramo ou "branch" no GitLab

Depois do trabalho concluído é feito um “commit” interno. Isto guarda as alterações feitas no ramo interno do computador. Faz-se depois um “push” para o GitLab. Isto é um upload das alterações feitas ao ramo de desenvolvimento guardadas anteriormente no computador local. Estes push têm uma nomenclatura especifica para organização - [nome do projeto][designação da *user story*][equipa desenvolvedora].

Neste nosso caso seria [A2C2F][AA2C2F-222][FRONTEND].

É depois feito um merge request, onde as alterações são depois validadas ou rejeitadas por um membro da equipa que não trabalhou nesta *story* em específico. Caso as mudanças sejam rejeitadas, os problemas encontrados são depois resolvidos, e este mesmo ciclo de validação ou rejeição começa de novo.

Finalmente, caso não sejam encontrados problemas com o código, é feito um merge deste ramo para o ramo de desenvolvimento principal. Isto é, as alterações feitas ao ramo para adicionar o código da *user story* é integrado no projeto.

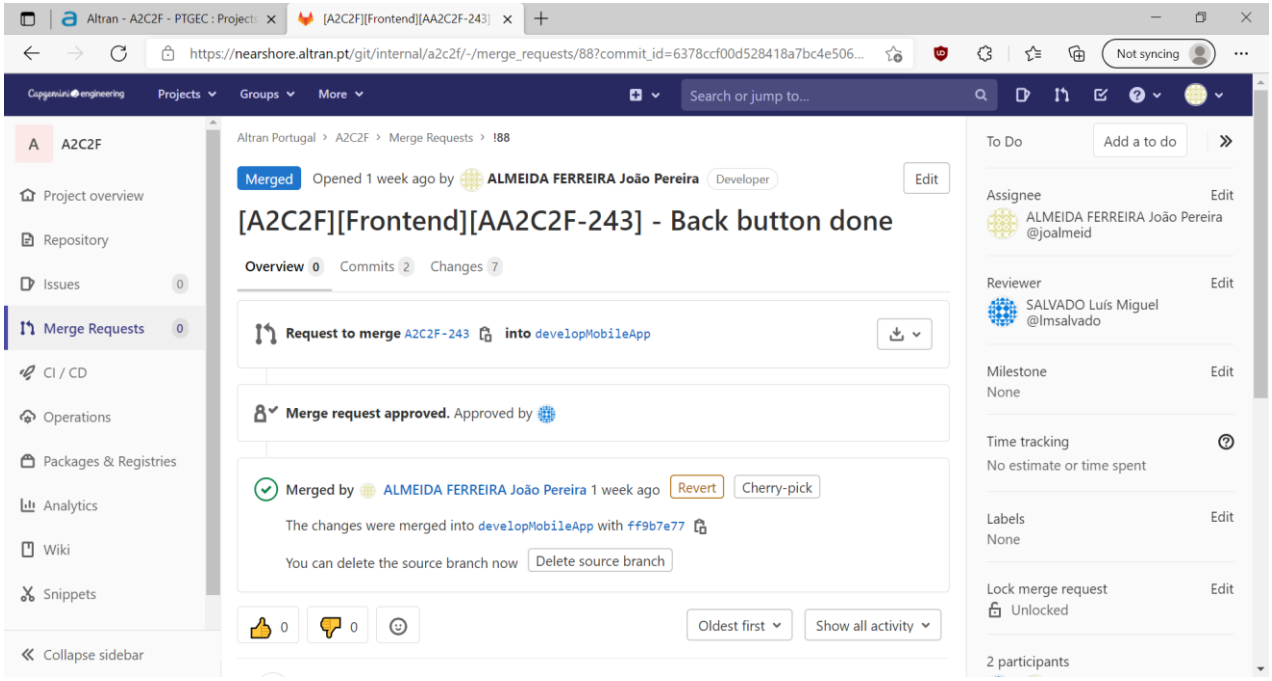

Figura 7 - Exemplo de um merge request

TensorFlow 2.0

TensorFlow 2.0 é uma biblioteca que fornece um ecossistema abrangente de ferramentas para desenvolvedores, investigadores e organizações que desejam criar aplicações escaláveis de machine learning (aprendizagem automática) e deep learning (aprendizagem profunda). [9]

TensorFlow 2.0 é uma biblioteca utilizada na linguagem de programação Python.

De todas as estruturas de Deep Learning disponíveis, o TensorFlow é uma das bibliotecas de Deep Learning mais populares e amplamente usadas em empresas no dia de hoje. Torna mais fácil trabalhar com dados complexos e construir modelos de redes neuronais articifiais para resolver problemas de negócios.

Esta biblioteca não pode correr em dispositivos móveis. Desta forma, é preciso converter o modelo TensorFlow 2.0 em TensorFlow lite, um modelo mais leve e menos complexo, que pode correr em dispositivos moveis.

Retrofit 2

Retrofit é um cliente REST para Java e Android. Facilita a recuperação e envio de JSON (ou outros dados estruturados) por meio de um serviço de web baseado em REST. No Retrofit, é possível configurar qual o conversor que será utilizado para a serialização de dados.

Normalmente, para JSON, é utilizado o GSon (conversor utilizado neste projeto), mas é possível pode adicionar conversores personalizados para processar XML ou outros protocolos. O Retrofit usa uma biblioteca OkHttp para pedidos HTTP. [11]

```
var retrofit = Retrofit.Builder() //retrofit start
    .baseUrl(baseUrl)
    .addConverterFactory(ScalarsConverterFactory.create())
    .addConverterFactory(GsonConverterFactory.create())
    .build()
var jsonPlaceHolderApi = retrofit.create(
    JsonPlaceHolderApi::class.java
)
```

Figura 8- código de utilização de Retrofit

REST

REST significa **Re**presentational **S**tate **T**ransfer.

É um estilo de arquitetura para projetar aplicações vagamente baseadas em HTTP, que é frequentemente usado no desenvolvimento de serviços web. REST não impõe nenhuma regra sobre como deve ser implementado a um nível inferior, apenas coloca diretrizes de design de alto nível e deixa o programador pensar na implementação. [12]

Este estilo arquitetural define seis (6) restrições de um serviço web:

1. Interface uniforme – deve ser decidida a interface de APIs para recursos dentro do sistema. Esta interface deve ser seguida religiosamente;

2. Cliente-servidor - significa que a aplicação cliente e a aplicação servidor devem ser capazes de evoluir separadamente, sem qualquer dependência entre os mesmos;

3. Sem estado – o servidor não armazenará nada sobre o último pedido HTTP feito pelo cliente;

4. “Cacheable” – devem ser guardados em cache todos os dados possíveis, de forma a melhorar a performance;

5. Sistema em camadas – se possível, cada servidor deve ter a sua própria responsabilidade (servidor 1 guarda dados, servidor 2 valida pedidos, etc.);

6. Código disponível – se necessário para o funcionamento da aplicação, enviar código executável diretamente em vez de objetos.

json Server

Json server é uma ferramenta de desenvolvimento. Antes do servidor estar a postos e a funcionar, é possível utilizar esta ferramenta para simular um back-end (REST API) de forma extremamente fácil. [13]

Os dados são guardados num ficheiro json localmente. O json server é corrido numa linha de comandos, usando NodeJS.

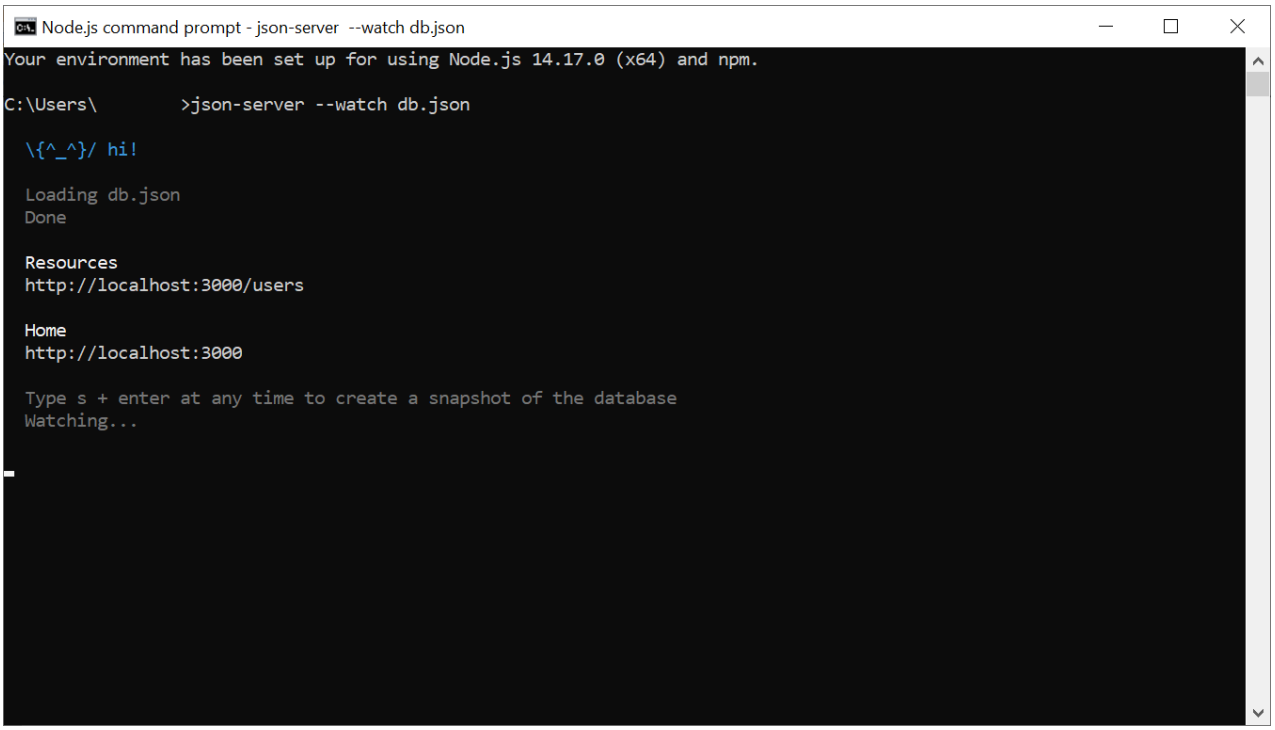

Figura 9 - Linha de comandos NodeJS

Capítulo 3 – Investigação

Inteligência artificial

Este é um projeto de investigação e desenvolvimento. Isto requer métodos de trabalho diferentes de outros projetos, nomeadamente na parte de investigação. Após a divisão de tarefas, foi pedido a cada membro da equipa produzir um documento State-of-the-art (SotA), no meu caso de reconhecimento facial. Este documento dita a situação e tecnologia atual de inteligência artificial, em particular, sobre reconhecimento facial.

De forma abreviada, atualmente a tecnologia de reconhecimento facial está bastante desenvolvida.

Algoritmos de aprendizagem automática constroem um modelo baseado em dados de amostra, conhecidos como dados de treino. O objetivo é treinar uma máquina para fazer previsões ou decisões sem ser explicitamente programada para isso. No nosso caso, o objetivo era utilizar uma câmara para reconhecer e registar, em tempo real, uma face.

Existem múltiplas bases de dados com grandes quantidades de imagens de faces, das mais diversas origens étnicas. Isto permite a um individuo com uma máquina suficientemente potente treinar um modelo de reconhecimento facial que reconhece caras, mesmo com alterações ligeiras, por exemplo óculos ou barba, utilizando a biblioteca TensorFlow 2.0.

Como os computadores portáteis pessoais e o Google Colab (serviço do Google que permite correr código Python diretamente no browser) são lentos demais para o treino de um modelo de inteligência artificial, a equipa chegou à decisão de utilizar um modelo pré-treinado de forma a expedir o desenvolvimento do projeto.

Algoritmos fundamentais de Machine Learning

Existem quatro algoritmos fundamentais de Machine learning:

1. Regressão linear;
2. Classificação;
3. Agrupar (Clustering);
4. Modelos ocultos de Markov.

A regressão linear é uma das mais básicas formas de aprendizagem automática, normalmente utilizada para previsão de valores numéricos. É uma forma de modelar uma relação entre uma resposta escalar (elemento de um campo usado para definir espaço vetorial) e uma ou mais variáveis. Os parâmetros desconhecidos (valores numéricos) são depois estimados dos dados fornecidos.

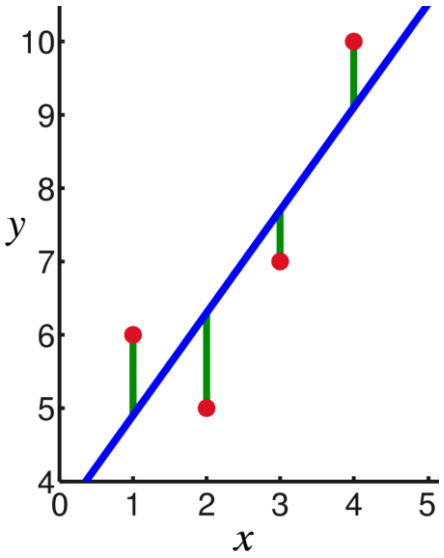

Figura 10 - Exemplo visual regressão linear

A classificação difere da regressão linear, pois não é utilizada para prever ou estimar valores numéricos, mas sim para separar os vários dados em classes com designações diferentes.

Agrupação é uma técnica de aprendizagem automática que envolve o agrupamento de pontos de dados. Em teoria, os pontos de dados que estão no mesmo grupo devem ter propriedades semelhantes, enquanto que os pontos de dados em grupos diferentes devem ter propriedades bastante diferentes.

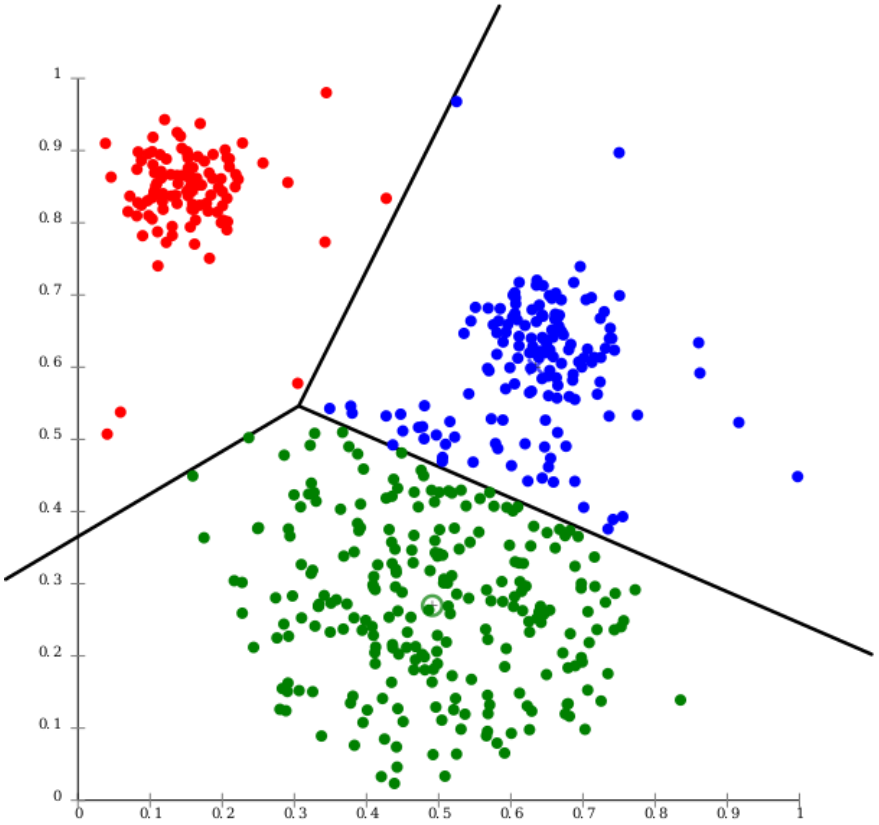

Figura 11 - Exemplo visual - Clustering

Finalmente, o modelo oculto de Markov é um conjunto finito de estados, cada um dos quais está associado a uma distribuição de probabilidade (geralmente multidimensional). As transições entre os estados são governadas por um conjunto de probabilidades chamadas probabilidades de transição. Um modelo oculto de Markov trabalha com probabilidades para prever eventos ou estados futuros.

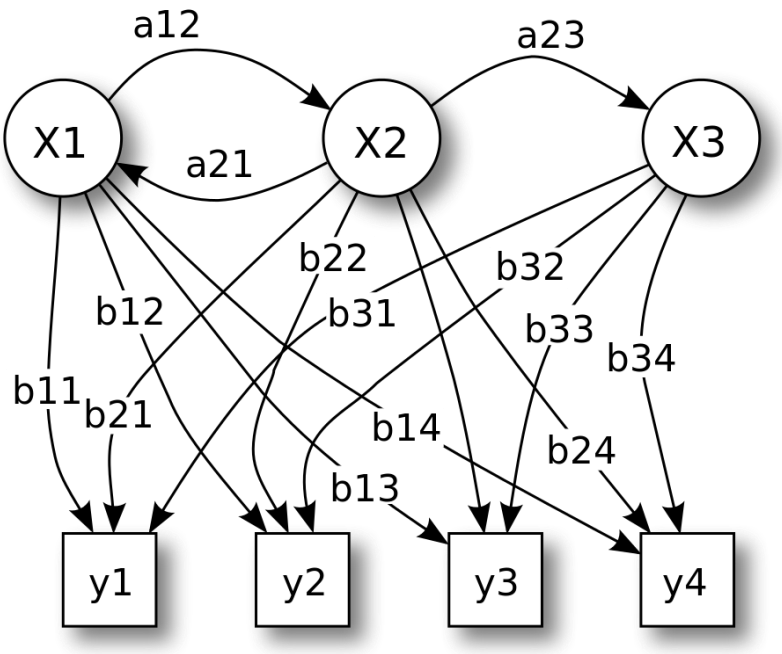

Figura 12 - Parâmetros de probabilidade de um modelo oculto de Markov [10]

X – estados

y - observações possíveis

a - probabilidades de transição de estado

b - probabilidades de saída

Android Automotive

O sistema operativo Android tem três principais versões diferentes, dependendo da plataforma onde corre. Num dispositivo móvel corre o Android comum, a versão mais popular. Mas para utilizar este sistema operativo no infotainment do veículo, existem duas versões distintas.

O Android Auto precisa de um dispositivo de Android externo. É possível depois espelhar o ecrã do smartphone para o sistema de infotainment.

Por outro lado, o Android Automotive permite ao sistema do carro correr este sistema operativo internamente, sem necessidade de um aparelho externo. Isto torna toda a experiência do utilizador mais atrativa.

Inicialmente, o projeto tinha como objetivo correr diretamente no sistema de infotainment do veículo, nomeadamente o sistema Polestar 2 da Volvo. No entanto isto não é possível atualmente, pois o Android Automotive apenas corre aplicações de multimédia, e não é possível aceder a uma câmara externa para o registo facial. Desta forma, foi decidido que a aplicação seria instalada separadamente pelo utilizador, através de uma loja de aplicações.

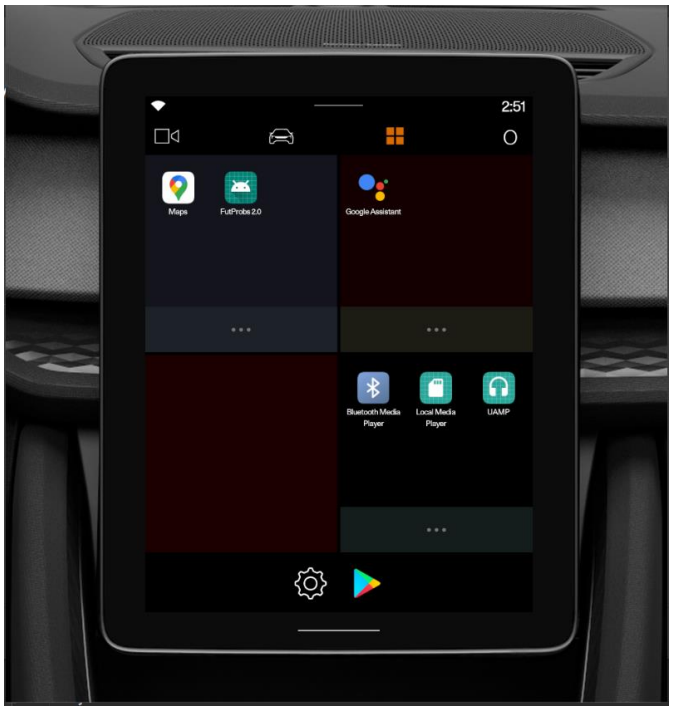

Figura 13 - Sistema infotainment Polestar 2

Capítulo 4 – Trabalho Desenvolvido

Ligação ao Back-end

Como apontei anteriormente, a equipa decidiu utilizar um modelo de reconhecimento facial publicamente disponível, para expedir o processo de desenvolvimento. Desta forma, a prioridade número um do projeto tornou-se a adição de funcionalidades à aplicação de forma a cumprir os objetivos finais do projeto.

Um desses objetivos é a funcionalidade de registar contas dos utilizadores. Era, então, necessário a criação de um back-end para tratar dos dados enviados de forma a serem guardados num servidor externo da empresa.

Esta parte do projeto foi trabalhada por outra parte da equipa. No entanto, a ligação dos dados introduzidos pelo utilizador e o back-end era responsabilidade da equipa de front-end.

A solução encontrada para este problema é o cliente Retrofit 2 e uma ferramenta chamada de json-server.

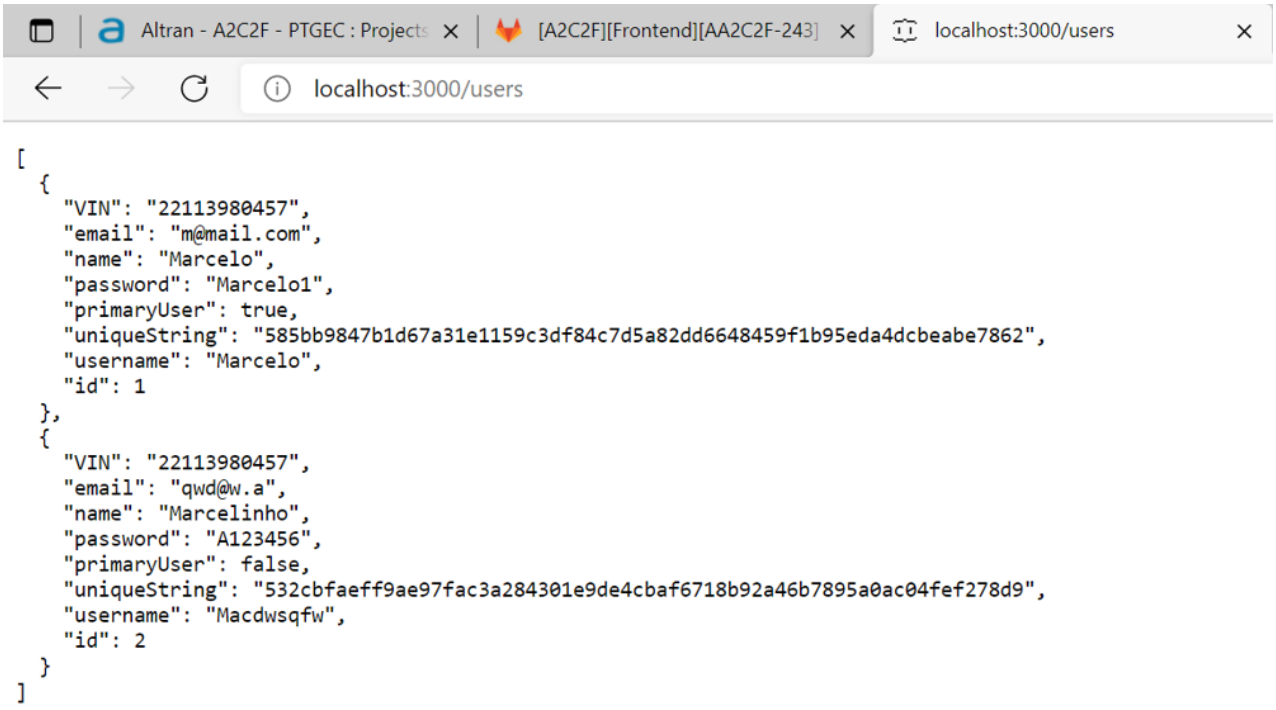

Figura 14 - base de dados local em formato json – dados para demonstração

Tratamento de Dados

Ficou decidido guardar os seguintes dados dos utilizadores no servidor:

- VIN – Vehicle Identification Number – número exclusive de cada veículo;
- email – email do utilizador;
- name – nome associado ao reconhecimento facial guardado, não único;
- password – palavra-chave definida pelo utilizador. Esta não é guardada diretamente, é passada por um algoritmo de Hashing de forma a garantir a segurança dos dados;
- primaryUser – define se o utilizador é primário ou não. O utilizador primário é uma espécie de administrador, tem acesso a certas funcionalidades que os outros não;
- uniqueString – uma String de carateres que identifica cada utilizador unicamente. É associada também ao reconhecimento facial guardado;
- username – nome de utilizador único registado no servidor – distinto do campo "name";
- id – número único associado a cada conta;
- token – estes dados não são enviados, mas gerados por parte do back-end. Permitem a um utilizador ficar com o estado de login.

Estes dados são enviados através de POST, ou no header ou no body do pedido. Caso fossem pedidos dados ao servidor, eram utilizadas funções GET.

```
interface JsonPlaceHolderApi {
    //login system with username and password
    @POST( value: "account/login")
    fun loginUserAlt(
        @HeaderMap headers: Map<String, String>
    ): Call<String>

    //login system with uniqueString
    @POST( value: "account/loginByUniqString")
    fun loginUserUnique(@Header( value: "uniqString") uniqString: String): Call<String>

    //logout system with token
    @POST( value: "account/logout")
    fun logoutUser(@Header( value: "x-access-token") token: String): Call<String>
```

Figura 15 - exemplo de endpoints utilizados

Menu de opções

O menu de opções foi programado com novas funcionalidades em mente. A solução que encontrei foi criar quatro listas: duas com strings dos nomes a apresentar e outras duas a apontar para as funções.

Uma aparece quando o utilizador está com o login feito, outra quando o oposto. Isto permite esconder certas funcionalidades dependendo do estado de login do utilizador. As duas restantes servem para direcionar para a função escolhida de acordo com a posição das opções na lista.

Os itens na lista de strings e funções têm que estar na mesma posição em arrays diferentes, por exemplo, a função “login()” tem a posição 0 no array 1 e a string “login” tem a posição 0 no array 2.

```
menuItems = arrayOf(
    "View Recognition List",
    "Delete your saved recognitions",
    "Import Photo",
    "Register User",
    "Change Language",
    "Change password",
    "Update user",
    "Logout"
)
```

Figura 16 - lista de strings 1

```
when (which) {
    0 -> displayNameListView()
    1 -> deleteConfirmPopUp()
    2 -> loadphoto()
    3 -> registerUser()
    4 -> changeLanguage()
    5 -> passwordChange()
    6 -> updateUser()
    7 -> logout(loggedInState.g
}
```

Figura 17 - Lista direciona para funções

Registo de utilizadores

Para começar o registo é pedido ao utilizador o VIN do veículo para ver se existe na base de dados. O registo de novos utilizadores consiste em receber os dados introduzidos pelo utilizador, validar estas informações, verificar se alguns dos dados únicos existem no servidor (email, username, VIN) e finalmente enviar esses dados através de um POST do Retrofit.

As validações incluem número mínimo de carateres, email com carateres específicos e palavras-chave com mínimos de letras, símbolos e números. Alguns dados não essenciais são também guardados localmente, para depois ser feito um login automático após esse primeiro registo.

Múltiplas formas de Login

Por definição, o projeto precisa que exista um login feito automaticamente ao reconhecer um rosto registado. Para este efeito foi necessário a criação de uma string única para cada utilizador. De acordo com certas características ou dados introduzidos é gerada uma string única, associada ao rosto e guardada no servidor. Por questões de segurança, não posso escrever quais os dados e os métodos utilizados para gerar esta string.

A forma de login secundário foi programada de acordo com o sistema clássico de nome de utilizador ou email e uma palavra-chave. Esses dados são depois enviados para o servidor para validação (palavra-chave com hash) e, se validados, o utilizador tem acesso à sua conta.

```
val postResponse = response.body()
var content = ""
content += "token: $postResponse\n"
Log.d( tag: "loginUserAlt_Success",  msg: "Content: $content")
val editor = userData.edit()
editor.putString("name", uName)
try {
    editor.putString("uString",
        SecureHashHelper.toHexString(SecureHashHelper.getSHA(uName))
    )
    editor.commit()
} catch (e: NoSuchAlgorithmException) {
    e.printStackTrace()
}
val stateEditor = loggedInState.edit()
stateEditor.putBoolean("loggedInState", true)
```

Figura 18 - Excerto caso login seja bem-sucedido

Comentar Código

De acordo com as boas práticas de desenvolvimento de software, comentámos todo o código do projeto até á data.

A notação de comentários em Java/Kotlin é ”//comentário”.

Manter estado de Login

Manter o utilizador com o login feito é um processo complexo. Após discussão interna, decidimos tirar proveito de uma funcionalidade interna do Android chamada SharedPreferences, ou SP. SP permite guardar dados de preferências do utilizador. Isto é, é possível guardar algo como a uniqueString de um certo utilizador e permitir que este tenha acesso a todas as funcionalidades até fazer um logout, manual ou automático, quando o SP é limpo.

Chamamos ao nosso SP particular ao login “LoggedInState”.

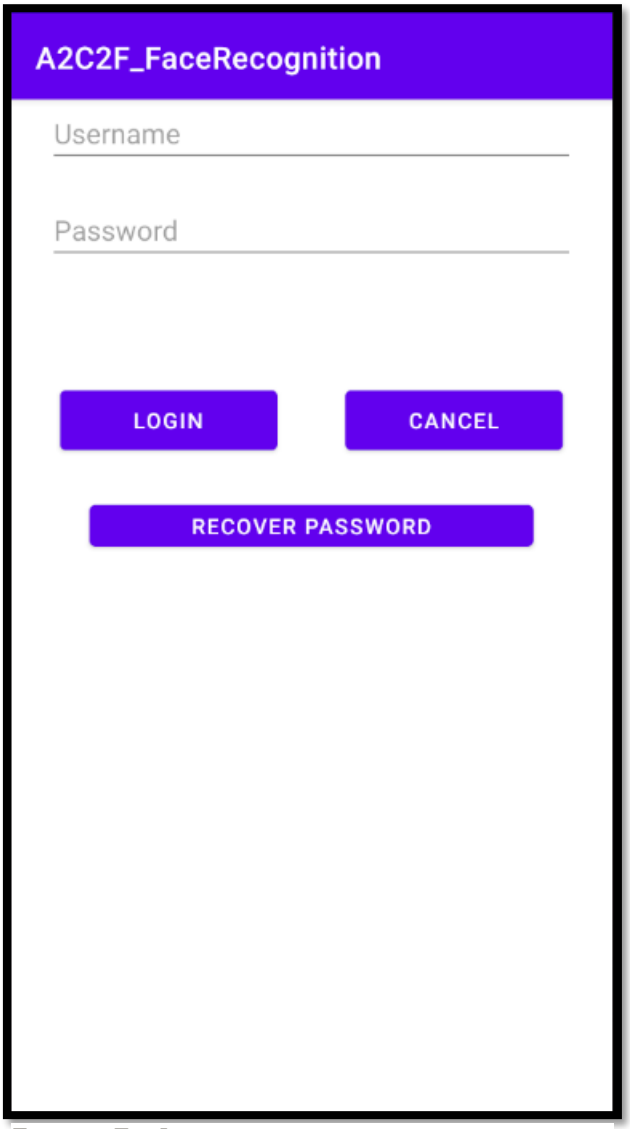

Figura 19 - Ecrã Login

Internacionalização – Suporte para múltiplas linguagens

Para tornar a aplicação compatível com outras linguagens é necessário transformar todos os strings programados diretamente em strings dinâmicos. Desta forma é possível criar múltiplas listas com todos os strings utilizados no projeto e mudar dinamicamente na aplicação a linguagem. Decidimos implementar uma forma de internacionalização de fácil expansão futura, programando duas linguagens atualmente: português e inglês. Em vez de ser programado diretamente onde necessário no código, o texto da aplicação é escrito de forma dinâmica no momento da abertura da aplicação. As traduções de cada string estão guardadas em ficheiros internos, separados por linguagem. Uma variável escolhida pelo utilizador dita qual dos ficheiros de strings será utilizado pela aplicação (inglês como predefinição).

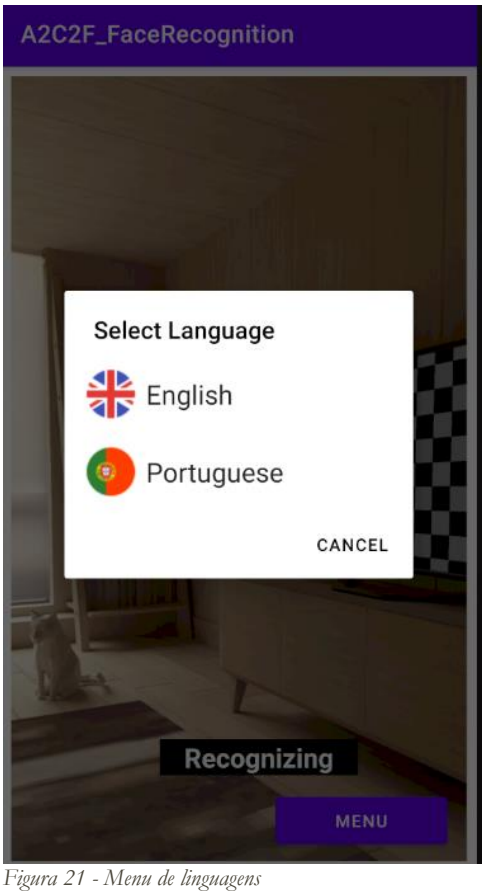

Figura 21 - Menu de linguagens

```
<string name="app_name">A2C2F_FaceRecognition<
<string name="title_activity_login">LoginActiv
<string name="prompt_email">Email</string>
<string name="prompt_password">Password</strin
<string name="action_sign_in">Sign in or regis
<string name="action_sign_in_short">Sign in</s
<string name="welcome">"Welcome!"</string>
<string name="invalid_username">Not a valid us
<string name="invalid_password">Password must
<string name="incorrect_password">Incorrect pa
<string name="login_failed">"Login failed"</st
<string name="prompt_username">Username</strin
<string name="prompt_name">Name</string>
<string name="confirm_password">Confirm your p
<string name="vin">VIN</string>
<string name="login">Login</string>
<string name="logout">Logout</string>
<string name="recover">Recover Password</strin
<string name="logout_success">User was logged
```

Figura 20 - Lista parcial de strings dinâmicas

Logout

Espelhando o sistema de login, também o sistema de logout tem duas componentes: um logout manual e outro automático.

O automático ocorre cada vez que o utilizador feche a aplicação sem fazer logout manual. O token gerado pelo backend é guardado localmente, e ao iniciar a aplicação esse token é comparado ao que está no servidor. Caso a parte do servidor não autentique o token como “logged in”, programamos a aplicação para fazer um logout automático. Isto é, limpar o LoggedInState.

Todos os tokens estão num temporizador que dita o número de horas que um login pode estar ativo.

O logout manual envia um pedido para o backend que limpa o token da lista de tokens autorizados.

Alteração de dados

Programei uma forma de o utilizador poder alterar certos dados registados na base de dados do servidor de forma semelhante ao registo. A diferença é que utilizei um PUT em vez de POST. Um PUT permite a alteração dos dados de uma conta, assumindo que o utilizador está com login feito e que os dados não entrem em conflito com outros dados únicos do servidor.

Testes unitários

Um teste unitário é o teste de uma única função.

O Android Studio possui uma funcionalidade que permite testar qualquer função que retorne algo. Tomando partido disto, é apenas necessário selecionar quais as funções a testar e a IDE trata do resto.

Correr a palavra-chave por um algoritmo hash

Para não comprometer a segurança dos dados dos utilizadores, todas as vias de comunicação da password são feitas após um hashing da mesma. No registo de uma nova conta, login, alteração de palavra-chave, etc. nunca é enviada ou recebida a chave antes de hashing. Cada vez que o utilizador introduz a sua palavra-chave é corrido o algoritmo com SALT (string adicionada à palavra-chave forma de tornar hashing mais seguro). Em caso de login, comparamos as strings geradas depois de hashing, uma gerada no momento do login e a outra guardada no servidor. Como duas strings diferentes têm sempre o mesmo resultado com hash, apenas comparamos esse resultado.

Este processo é diferente de cifrar, pois uma cifra pode ser revertida, enquanto que uma string após hashing não.

Conversão Kotlin

Mais para o final do projeto foi-nos pedido que convertêssemos todo o código para a linguagem Kotlin.

A IDE tem uma funcionalidade que permite converter o código todo, mas não é perfeito. No fim da conversão existiam centenas de erros e defeitos no código, algo que tive que arranjar um a um, para garantir que não me falhasse nenhum.

O processo de conversão de Java para Kotlin especificamente é bastante fácil porque Kotlin é efetivamente Java com nomenclatura e funções simplificadas. A razão pelo qual nos foi pedido isto é que a Google tem feito pressão e adotado oficialmente a linguagem Kotlin como a linguagem de programação principal de Android. Após esta adoção há anos atrás, Kotlin ganhou muita popularidade, e é agora a linguagem mais usada para programação Android.

```
public static void main(String[] args) {
    system.out.println("Hello, world!!!");
}

fun main() {
    println("Hello, world!!!")
}
```

Figura 22 - Exemplo da diferença de complexidade entre Java (cima) e Kotlin (Baixo)

Demonstração final de Sprint

No final de cada sprint a equipa prepara uma demonstração do trabalho feito por todos os membros durante as duas semanas de trabalho.

Essa preparação requer algum trabalho, pois as demonstrações destinam-se a ser apresentadas ao cliente. Noto, novamente, que este projeto em particular é incomum e as demos são preparadas para o líder da equipa.

A preparação começa por rever todas as *stories* feitas com o objetivo de encontrar e corrigir o máximo número de defeitos possível. Depois são organizadas todas as *user stories* na ordem que irão ser apresentadas. Finalmente, é feita uma demonstração teste entre os membros da equipa para receber críticas e melhorar a apresentação.

Durante a apresentação, todas as *stories* trabalhadas são lidas e descritas, fazendo uma partilha de ecrã para mostrar os frutos do trabalho feito, diretamente na aplicação. Cada *story*, depois de apresentada, pede aceitação por parte do cliente (neste caso, líder de equipa). No final da apresentação dá-se o sprint como concluído.

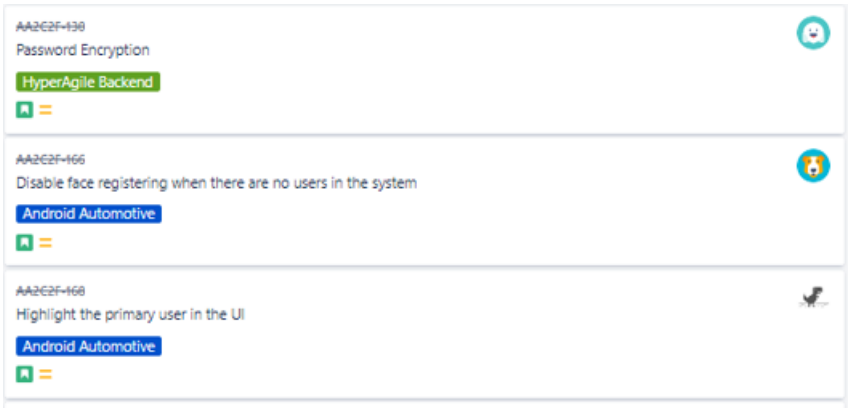

Figura 23 - Exemplo de User Stories completas e aceites

Capítulo 5 – Conclusão

Conclusão

O projeto ficou muito perto se ser concluído, com uma data de conclusão em Setembro. O sistema de reconhecimento facial, tanto para registo como para autenticação, funciona perfeitamente. Em relação ao sistema de envio de dados para o servidor, como registo de contas e login, também não só funcionam sem qualquer erro, mas permitem a adição de novos endpoints de forma fácil. Este é um tema comum por todo o projeto, tornar a estrutura do projeto em algo dinâmico, facilmente escalável.

De acordo com o líder da equipa, tanto o meu trabalho como o trabalho feito pelo resto da equipa atingiram os objetivos, um resultado bastante satisfatório.

De forma geral, penso que evolui bastante nestes últimos seis meses de estágio. Entre os novos conceitos, métodos de trabalho, tópicos extremamente interessantes e excelentes colegas, acho que foi uma experiência muito positiva tanto para a minha experiência profissional como crescimento pessoal.

Nunca na minha vida tinha programado com tanta intensidade. Dos tópicos abordados, inteligência artificial e reconhecimento facial foram sem dúvida a melhor parte de todo o estágio. Investigação sobre os mais variados algoritmos e técnicas é algo que me agradou bastante. Queria ter explorado mais o tema, em específico, treinado um modelo. No entanto, as circunstâncias e recursos não o permitiram.

De um lado profissional, penso que internamente, a Capgemini Engineering precisa de tornar mais eficiente o processo de divulgação de informação. Foi-me comunicado no dia final do estágio que me iria ser oferecido um contrato de trabalho. Infelizmente, semanas depois, fui informado que isto já não iria acontecer. No dia seguinte, após uma troca de emails, informaram-me que estava novamente a ser considerado para essa posição. Após isso, sem notícias. Pessoalmente, gostaria de ter ficado.

Resumindo, gostei bastante da experiência e dos tópicos abrangidos eo resultado foi bastante positivo. Apenas gostaria que a minha estadia fosse prolongada indefinidamente.

Bibliografia

[1] – Capgemini Group – “Our Company”

https://www.capgemini.com/pt-en/our-company/

[2] – Capgemini Engineering – “Quem somos”

https://capgemini-engineering.com/pt/pt-pt/quem-somos/capgemini-engineering-partnerships/

[3] – Fiscal Year 2020 results

https://investors.capgemini.com/en/publication/fy-2020-results/

[4] – Oracle – Password hashing

https://docs.oracle.com/cd/E26180_01/Platform.94/ATGPersProgGuide/html/s0506passwordhashing01.html#:~:text=Hashing%20performs%20a%20one%2Dway,back%20into%20the%20original%20password.

[5] – What is Scrum

https://www.scrum.org/resources/what-is-scrum

[6] – What is Jira used for?

https://www.atlassian.com/software/jira/guides/use-cases/what-is-jira-used-for

[7] – Android Studio

https://developer.android.com/studio/intro

[8] – What is GitLab

https://about.gitlab.com/what-is-gitlab/

[9] – TensorFlow 2.0 tutorials

https://www.simplilearn.com/tutorials/deep-learning-tutorial/tensorflow-2

[10] – Hidden Markov Model – Wikipedia

https://en.wikipedia.org/wiki/Hidden_Markov_model

[11] – What is Retrofit – Vogella

https://www.vogella.com/tutorials/Retrofit/article.html

[12] – REST

https://restfulapi.net/rest-architectural-constraints/

[13] - json-server

https://www.npmjs.com/package/json-server