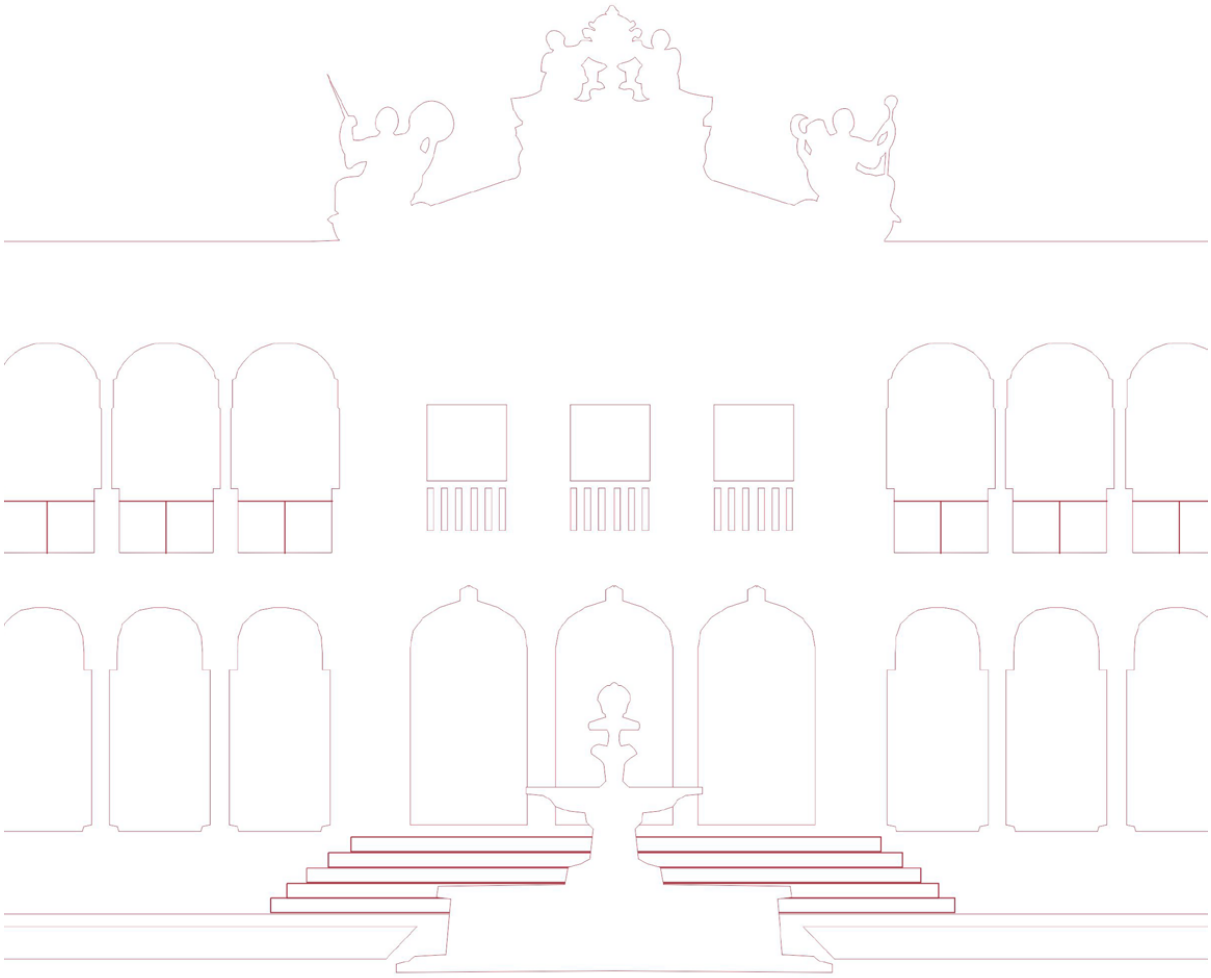
Estágio de Programador de Software - Location Intelligence

UNIVERSIDADE DE ÉVORA
ESCOLA DE CIÊNCIAS E TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA

Licenciatura em Eng. Informática
Estágio-Projecto 2020-2021



Leonardo Catarro

Orientador na empresa: Ricardo Francisco
Orientador no departamento: Professor José Saias

Trabalho desenvolvido na empresa FocusBC no âmbito da disciplina de Estágio-Projeto da Licenciatura de Engenharia Informática.

Évora, 25 de junho de 2021

Conteúdo

1 Introdução

Este relatório foi concebido para a unidade curricular de estágio-projeto da licenciatura de engenharia informática. A realização deste trabalho foi efectuado para a empresa FocusBC.

O desenvolvimento de software é caracterizado por diversos processos, a maioria deles obedece a algumas atividades básicas, tais como o levantamento de requisitos, análise de requisitos, projeto, implementação, testes e implantação. Durante o estágio na FocusBC, o sistema desenvolvido foi implementado como uma webapp-api, durante este procedimento foram utilizadas várias ferramentas e linguagens para a desenvolver.

As ferramentas mais utilizadas neste projeto foram a *DotNet[1, 2, 3], Google Data Studio[4, 5] e PostgreSQL[6, 7].* A primeira é desenvolvida pela Microsoft e caracterizada pelos seus recursos interessantes na conexão a bases de dados, comunicações de redes e criptografia, para além disto apresenta uma pacote repleto de soluções codificadas para problemas mais frequentes dentro da programação, uma vez que o framework suporta diversas linguagens de programação , o segundo é utilizado para a visualização de dados nos *dashboards* e por fim a *PostgreSQL* é um poderoso sistema de base de dados de código aberto de objectos-relacionais.

A linguagem utilizada foi o *CSharp*, esta é orientada a objetos, que foi desenvolvida pela Microsoft e faz parte da plataforma .NET. Embora a linguagem *CSharp* tenha sido criada do zero, foi baseada na linguagem C++ e tem muitos elementos da linguagem *Pascal e Java.*

Os empregados da Focus escolheram vários alunos da licenciatura para estagiar na empresa. Para o desenvolvimento deste projeto fui escolhido com mais um dos meus colegas João Cavaco, tendo sido um trabalho de grupo onde foram estabelecidos horários que permitissem a presença de ambos no desenvolvimento do projeto. A implementação destas técnicas permitiu desenvolver alguns dos meus objetivos pessoais como também perceber como funciona o mercado de trabalho.

Este relatório está dividido em quatro partes, estas descrevem as atividades desenvolvidas ao longo do estágio, como também, enquadra o desenvolvimento efetuado durante o trabalho.

1.1 Enquadramento

O estágio foi realizado remotamente, devido ao COVID-19, para a empresa FocusBC, sedeada em Lisboa. no âmbito da unidade curricular Estágio-Projeto do 3ºAno de licenciatura em engenharia informática da Universidade de Évora. Neste projeto estivemos integrados numa das equipas de *Delivery* da empresa, constituída por 4 profissionais.

Os horários deste estágio foram bastante flexíveis. Como trabalhei em *Pair Programming[8]* todo o estágio, nem sempre os horários foram compátiveis. Para resolver esta situação, no inicio do estágio estabelecemos um horário comum aos dois, e sempre que um não pudesse, o outro faria o horário normal e o que faltou compensaria noutra data que lhe fosse conveniente.

Como mencionei acima, foi tudo feito remotamente e visto que eu e o meu colega estudamos na mesma instituição(Universidade de Évora), foi possível trabalhar lado-a-lado na Univerisidade, tornando-se assim mais dinâmico o desenvolvimento do projeto.

1.2 Objetivos

O objetivo do trabalho de estágio desenvolvido foi garantir a existência de uma plataforma(*WebApp)*, regularmente atualizada, onde fosse possível ter acesso às taxas de utilização, novos utilizadores criados e entre outras métricas, de um dos seus produtos, o *LIS ou CaaP(City As A Platform).*

Foram também objetivos deste estágio, o desafio de desenvolver soluções simplificadas baseadas em *Google Cloud[9]*, em ambiente web como recurso às mais recentes tecnologias, a aplicação de conhecimentos

e técnicas de programação para a criação de soluções funcionais de forma a mostrar o conhecimento adquirido no desenvolvimento de software. Por último, e não menos importante, conhecer como é trabalhar em um ambiente profissional real.

1.3 Contribuições

Com o desenvolvimento desta *webapp-api[10]*, a empresa consegue o total controlo na expansão do seu produtos por todos os seus clientes.

Foram concluídas todas as metas propostas no âmbito deste projeto, finalizando assim com o projeto de estágio completamente terminado e funcional.

1.4 Estrutura do documento

Seguidamente, temos o Estado da Arte, onde será mencionado o levantamento das várias soluções propostas, ferramentas ou metodologias e os fatores que levaram à escolha de cada uma delas.

Posteriormente ao Estado da Arte, seguir-se-á, no ponto 4, o Ambiente de Desenvolvimento, com os sub-tópicos Ambiente técnico(Hardware e Software usados no desenvolvimento), Ambiente aplicacional e Metodologia de trabalho.

Para o ponto 5 será relatado o Trabalho desenvolvido, começado por um breve resumo e aprofundado no ponto 5.1 - Descrição detalhada.

Finalmente, no ponto 6, uma Avaliação crítica relativamente aos conhecimentos adquiridos na experiência, de um ponto vista técnico e não-técnico, como por exemplo nas áreas de trabalho de equipa e comunicação.

2 Ambiente empresarial

A FocusBC é uma empresa parceira da Google e que opera em diversos pontos do globo. É um empresa privada, constituída por cerca de 24 profissionais e as suas operações variam desde a entrega de localização inteligente e localização inteligente em tempo-real à criação de plataformas escaláveis para empresas.

É uma equipa jovem que faz a diferença no panorama da inteligência de localização. A empresa está a crescer e têm uma equipa bastante diversificada: desenvolvedores júniores e seniores especializados em vários tipos de linguagens de programação. Têm também especialistas em *SIG(Sistema de Informação Geográfica)*, gestores de projetos, consultores empresariais, líderes técnicos e proprietários de produtos.

A empresa está localizada na ensolarada Lisboa. Com um edifício muito recente, tem um grande espaço, sala de reuniões, espaço de trabalho e sala comum onde são feitas sessões técnica de partilha.

É um grupo, com um espírito muito forte de cooperação, trabalho de equipa e um excelente ambiente de trabalho, garantindo que há sempre equilíbrio entre a vida profissional e social.

Na empresa existem várias equipas separadas. Durante o estágio, derivado à situação de pandemia atualmente vivida, trabalhei remotamente na equipa de *Delivery* da empresa(constituída por 4 profissionais), tomando a posição de *BackEnd Developer*, sempre recebendo instruções do orientador na empresa, também membro desta equipa.

3 Estado da arte

No âmbito do trabalho de estágio, fiz o levantamento de várias soluções possíveis para o desenvolvimento do projeto. Inicialmente, o levantamento de possíveis modelos de dados a usar no desenvolvimento do projeto, apresentando, após conclusão de cada solução, a estrutura das tabelas *Postgres[6, 7]* para o armazenamento da informação necessária. Na análise das diferentes soluções cheguei á conclusão que estávamos a complicar um problema simples, tendo sido uma das primeiras soluções apresentadas com algumas correções.

Como foi mencionado, a cada modelo de dados criado ou alterado era feita uma apresentação ao project manager e à equipa. Ao nível de ferramentas usadas, utilizei um software criado pela empresa, para a geração inicial dos ficheiros do projeto e para tornar o desenvolvimento do *BackEnd* independente do *FrontEnd*.

Ao nível da linguagem de programação utilizada foi dada opção de escolha. Pudemos optar pelo software *NodeJS[11]*, que recorre à linguagem *JavaScript* para desenvolver código ou à *framework DotNet[2]*, que usa a linguagem de programação *CSharp[12, 13]*. Derivado á semelhança na sintaxe do *CSharp* com o *Java*, a equipa optou pela escolha do *DotNet*, tornando assim o desenvolvimento da minha parte mais eficiente e rápido.

4 Ambiente de desenvolvimento

Ao nível de hardware, utilizei as minhas próprias máquinas para o desenvolvimento do projeto.

Ao nível de software, usamos o *FBC-Cli, o Visual Studio Code(VSCode)[14], DotNet[1, 2, 3], CSharp[12, 13], o Slack[15], Google Meets[16], Google Data Studio[4, 5]*, e, finalmente, o *Bitbucket[17]* com integração do *JIRA Software[18].*

A minha experiência com estes software foi bastante positiva, embora já tivesse trabalho com alguns dos softwares descritos, não só em projetos da universidade, como também em projetos pessoais. Foi na utilização (*DotNet, CSharp, Slack, JIRA, FBC-Cli, Terraform e Typescript*) que consegui ter uma maior noção da vantagem que estes oferecem ao desenvolvimento individualmente e em equipa.

4.1 Ambiente técnico

4.1.1 Hardware

A minha máquina é um portátil Asus, com um processador Intel Core i7 7700HQ, 16 GB de memória, uma placa gráfica Nvidia GeForce GTX 1050 de 4GB e um disco SSD Nvme de 1 TB, juntamente com um disco HDD de 1TB.

Por vezes também utilizava o meu Desktop, com um processador Intel Core i7 9700K, 16 GB de memória, uma placa gráfica Nvidia GeForce RTX 2060 de 6GB e um disco SSD Nvme de 1 TB, juntamente com um disco HDD de 1TB, também.

Ambas as máquinas possuem *dual-boot* de Windows 10 com o Linux, distribuição Ubuntu versão 20.04. Tendo sido usado o Ubuntu como Sistema Operativo para o desenvolvimento do projeto.

Para aceder às bases de dados da empresa ou qualquer software por eles usado, foi nos garantido acesso através do IP da máquina e apenas era possível a autenticação através do meu email da empresa, inicialmente fornecido.

A empresa opera num conjunto de servidores provisionados pela *GCP(Google Cloud Platform)[9]* e completamente manuseáveis/configuráveis através da interface da plataforma da *Google Cloud*. Estes serviços em *Cloud*, são comummente usados, e cada vez vais vistos pelas empresas como uma vantagem, visto que tem um custo reduzido e eliminam o custo inicial de hardware físico. Importante mencionar que os servidores usados pela empresa suportam *BitBucket*.

4.1.2 Software

No meu trabalho de estágio usei o *Slack e Google Meets*, softwares usados para comunicação entre membros internos e externos á equipa onde estava inserido.

Para o desenvolvimento foi usado a *framework DotNet[2, 1, 3]*, desenvolvida pela microsoft, utilizando a linguagem *CSharp[12, 13]*, como principal linguagem de programação.

Usamos o *FBC-Cli*, ferramenta desenvolvida pela empresa internamente, que garante a independência e acelera o processo de desenvolvimento entre o *BackEnd e o FrontEnd*, produzindo todos os ficheiros iniciais de uma *WebApp* de exemplo, criando todas as *Entities, DTO's, Services, Repositories e Controllers* necessários ao desenvolvimento(posteriormente alteráveis através de um ficheiro *JSON[19]*, que contenta a estrutura das novas entidades a serem usadas).

Para a persistência dos dados filtrados, foi usado uma Base de Dados *Posgresql[6, 7]*, cujo *schema* tinha a estrutura do modelos de dados inicialmente criado.

Para a construção do *dashboard* com a informação filtrada e armazenada recorremos ao *Google Data Studio [4, 5]*, um software desenvolvido pela google adaptado para a criação de *dashboards*. Posteriormente este *dashboard* foi embebido no *FrontEnd* produzido.

Foi usado ainda o *Bitbucket[17]* como ferramenta de controlo de versões para o projeto e a integração do *JIRA Software[18]* para garantir monitoramento de tarefas e acompanhamento do projeto, garantindo o a fácil gestão de todas as suas atividades num único lugar.

Para manter todo o desenvolvimento em *cloud*, usamos o *Google Cloud Platform[9]* e algumas das ferramentas por ele oferecidas.

Finalmente, usamos também o *Visual Studio Code[14]*, um editor de código-fonte desenvolvido pela Microsoft para Windows, Linux e macOS. Este inclui suporte para depuração, controlo de versionamento *Git* incorporado, realce de sintaxe autocomplete de código.

4.1.3 Ambiente aplicacional

O projeto enquadra-se na monitorização das instâncias do LIS(produto da empresa). Projeto este que garante o acesso às taxas de utilização das instâncias do produto em geral ou cada uma em particular, sempre sendo visualizável as mesmas métricas(requisitos). Contém diversas métricas apresentadas desde o número de novos utilizadores até á taxa de utilização das aplicações das instâncias.

Com este projeto, será possível produzir dados estatísticos relativos ao consumo do produto pelos clientes e verificar quais as instâncias que são mais ou menos utilizadas num intervalo de tempo escolhido pelo utilizador do dashboard.

O projeto faz parte de uma aplicação maior o *LIS ou CaaP* e o seu papel é sincronizar dados de todas as suas instâncias de forma á empresa conseguir visualizar a sua evolução, global ou individual(instância a instância)

4.2 Metodologia de trabalho

No desenvolvimento do projeto foi adoptado o Desenvolvimento Ágil de Software[20], este baseia-se no desenvolvimento progressivo, em curtos períodos(no âmbito do estágio, 15 dias), de forma a minimizar o risco e enfatizando a comunicação em tempo real. Em cada uma destas iterações eram fornecidos tickets para o desenvolvimento do projeto e o nosso trabalho era implementar as funcionalidades no ticket descritas para futuramente apresentar.

Para cada ticket era criado um *branch no BitBucket[17]* de forma a separar o código principal(código *deployed*) do código implementado para a resolução do ticket escolhido. Após resolução de um ticket era submetido um *PR(Pull Request)* de forma ao *project manager* conseguir perceber o que foi feito e dar algum *feedback*.

No fim dos 15 dias, procedia-se a uma reunião, chamada *Bi-Weekly*, onde era apresentado o estado atual de todos os projetos da empresa, tabelas e gráficos relativos á conclusão do que tinha sido previsto para os 15 dias passados. Após isso eram mostradas as novas metas a serem alcançadas até à próxima *Bi-Weekly*.

A interação com a restante equipa de trabalho era diária, sempre que era necessário ajuda bastava agendar uma chamada via *Slack ou Google Meets[15, 16]*, expondo o problema, e em *Pair programming[8]*(dois programadores trabalham juntos numa estação de trabalho) este era resolvido, implementando o código necessário.

5 Trabalho desenvolvido

Quando iniciei o estágio na empresa, fui enviado para um curso de formação de 1 semana, fornecido pela Google, para aprender superficialmente o *GCP(Google Cloud Platform)[9] e as API's do Google Maps* existentes, api's estas que estão inseridas dentro da *GMP(Google Maps Platform)[21].* Este curso é um procedimento normal de qualquer novo profissional que comece na empresa.

Após a conclusão do curso e uma breve apresentação dos diferentes softwares usados na empresa que por sua vez iriam ser usados durante o projeto. Juntei-me á equipa de *Delivery* da empresa, constituída por 4 pessoas, responsáveis por todo o *brainstorming* para a entrega de um produto ao público. Na altura os membros da equipa estavam responsáveis por diferentes projetos.

Fui desafiado, em conjunto com o meu colega de estágio João Cavaco, aluno da licenciatura em Engenharia Informática da Universidade de Évora, a produzir um *dashboard* onde fosse possível verificar a evolução de um produto da empresa.

Esta tarefa, dividida em várias sub-tarefas, ocupou toda a duração do estágio, com apresentações recorrentes do estado atual do produto, excluindo as 2 apresentações feitas a toda a empresa(a meio e no final do estágio). Neste projeto iniciamos pelo desenvolvimento do *backend* e terminamos no desenvolvimento do *frontend(dashboard).*

5.1 Descrição detalhada

5.1.1 Credenciais Introdutórias

Começei por fazer uma formação em *Google Cloud Platform e Google Maps API's [9, 21].* Neste curso foram feitas 4 credenciais, com exames finais, de forma a testar o meu conhecimento adquirido ao longo dos capítulos do curso.

A primeira credencial de *GCP[9]* era relativamente à venda do produto, em nada interessante para o nosso projeto, porém necessária para conseguir atingir as próximas credenciais. Já na segunda de *GCP[9]*, passei por toda a sua interface e funcionalidades para perceber como funciona e potencialidade deste produto no desenvolvimento de software.

Na parte de *Google Maps API's[21]*, e numa primeira credencial, também passei pela venda do produto, novamente nada interessante, porém necessária pelo mesmo motivo acima referido. A segunda abrangeu todo um conhecimento superficial das API's existentes no *Google Maps* suas utilizações e casos práticos de uso das mesmas. Estas API's estão divididas em três grupos maiores: *Maps, Routes e Places.* Estas API's são onze ao todo.

Embora interessante a potencialidade de algumas destas api's, esta parte no âmbito do nosso projeto de estágio acabou por não ser usada por mim, mas sim por outros colegas que também realizaram o seu estágio na mesma empresa.

5.1.2 Breve introdução e aprendizagem

Após a primeira semana de estágio, onde foram finalizadas formações introdutórias, seguiu-se, juntamente com o nosso orientador de estágio, uma breve introdução aos softwares que iriam a ser usados no desenvolvimento.

No resto da semana procedi a uma aprendizagem autónoma, lendo alguma documentação e visualizando alguns tutoriais fornecidos pelo orientador e mesmo outros escolhidos por mim, de forma a aprofundar o meu conhecimento em algumas das ferramentas que já conhecia e tinha usado e para ganhar um conhecimento superficial relativamente às ferramentas que ia usar pela primeira vez, nomeadamente a *framework DotNet[1, 2, 3], a linguagem CSharp[12, 13] e o ORM usado: Dapper[22].*

5.1.3 Desenvolvimento do projeto

Como já foi mencionado o nosso projeto e as suas funcionalidades foram dividias por *tickets* no *JIRA[18]*, de forma a ter uma organização para o desenvolvimento do produto. A nomenclatura dos *tickets* seguia uma estrutura especifica, iniciada por LISDAS(acrónimo de *LIS Dashboard*), seguido no número do ticket e terminando com o nome do ticket(funcionalidade/aspetos a desenvolver). De seguida, seguem os tickets e a exposição do que foi tratado/implementado em cada um ao longo do desenvolvimento do projeto:

LISDAS-1-Analyse the client requirements and work on a database model proposal: o cliente forneceu-nos os requisitos do projeto numa apresentação pouco formal, de seguida fomos, eu e o meu colega, desenvolver um modelo de dados de forma a conseguir armazenar os dados pedidos nos requisitos numa Base de Dados *Posgres[6, 7]* central para de seguida ser mostrada a informação no *dashboard*. Neste ticket ao longo do desenvolvimento do modelo, fomos apresentando à equipa e ao orientador de estágio para garantir que tudo estava a prosseguir como esperado e procedendo às alterações aconselhadas em cada uma dessas reuniões.

LISDAS-2-Generate scafold frontend and backend: neste ticket usamos o *software FBC-Cli*, mencionado no ponto 4.1.2, que nos produziu os ficheiros iniciais, com um *fake backend em C Sharp[12, 13]* e um *frontend em HTML, CSS[23, 24]* e *TypeScript[25, 26]* que posteriormente foi alterado.

Para gerar o *frontend* era aberto um terminal da diretória do projeto, e executados os comandos *fbc new* e escolhendo a opção *angular-spa* opção que gera uma *Single Page Application[27]* escrita em *Angular[28]*, plataforma e *framework* para construção da interface de aplicações usando *HTML, CSS[23, 24]* e, principalmente, *TypeScript[25, 26]*, criada pelos desenvolvedores da Google e de seguida apenas era necessário aceitar os padrões de configuração. O processo para gerar o *backend* é em tudo idêntico, apenas seria necessário em vez de *angular-spa* escolher *webapp-api [10]*.

Para correr o projeto apenas era executado o comando fbc run em dois terminais distintos. Um na pasta *frontend* e outro na pasta *backend*, geradas no *fbc new* executado previamente.

LISDAS-3-Insert Data base model: neste ticket e usando um ficheiro JSON[19] onde inserimos toda a estrutura das tabelas desenvolvidas no modelo de dados(colunas e tipos de dados das colunas), criamos com *FBC-Cli* os ficheiros de código necessários ao nosso modelo de dados, ou seja, o ficheiro JSON[19] foi responsável por criar no projeto as *Entities, DTO's e ListDTO's* inseridos no JSON[19].

LISDAS-4-LIS instance synchronisation service: aqui criamos o *service LisSynchronizationService*, que basicamente gere todo o processo de sincronização, onde inicialmente e apartir do repositório de Instâncias do LIS, instância a instância, conectava-se à sua base de dados, puxava todas as métricas definidas nos requisitos, através de *queries SQL* e atualizava a nossa base de dados central com a nova informação. Criamos a função *Synchronize(função responsável por executar a sincronização de dados para o dashboard)*, e faz a sincronização da instância passada como argumento.

LISDAS-5-Fixing bugs in Synchronize function: Bugs surgiram ao testar o nosso serviço de sincronização, então abriu-se este ticket dedicado ao processo de correção desses mesmos bugs. Os principais bugs detetados foram ao níveis das *queries*, visto que foram desenvolvidas *queries* de dimensão nunca antes feita em projetos nossos. Daí termos perdido algum tempo a tentar perceber onde se encontravam alguns destes erros. Ainda neste ticket, após a resolução dos bugs mencionados, procedemos também a alguma otimização do processo, pois apenas com 2 instâncias a performance deixava muito a desejar.

LISDAS-6-LIS synchronisation endpoint: este ticket baseou-se na criação de um *endpoint* no *swagger[29]* dentro da api de forma a conseguirmos chamar manualmente o processo de sincronização. Ao ser feito o *request* neste *endpoint* presente no *swagger[29]*, o mecanismo de sincronização é ativado, iterando por todas as instâncias do LIS, executando as *queries* para, posteriormente, agregar os dados na base de dados central e, finalmente, atualizar os dados no *dashboard*.

LISDAS-7-Dashboard development: para este ticket fizemos uma breve introdução ao *Data Studio[4]* e nas semanas seguintes construímos o *dashboard* com base nos dados armazenados na base de dados. Ao longo do desenvolvimento surgiram enumeras dúvidas e bugs que nos fizeram perder algum tempo a tentar corrigi-los, ainda assim nada que tenha feito atrasar as *timelines* previamente marcadas. Após terminado o desenvolvimento, este *dashboard* foi embebido no *frontend da webapp*.

LISDAS-8-Setup infrastructure: provisionamento da infraestrutura com o *Terraform[30]* e a ajuda da plataforma Google, o GCP[9], e algumas das suas ferramentas.

LISDAS-9-Get additional instance data from instance CMLisboa: após testado o desenvolvimento na base de dados de desenvolvimento da empresa(base de dados de testes), partimos para um cliente real do LIS e adicionámos a instância referente á Câmara Municipal de Lisboa e testámos o nosso processo de sincronização de dados. Aqui verificaram-se alguns bugs mínimos nas *queries*, que facilmente foram resolvidos.

LISDAS-10-setup cloud scheduler: em fase final do desenvolvimento, configurou-se o *Cloud scheduler[31]*, ferramenta do *GCP[21]*, simulando, como o nome indica, um relógico, permitindo agendar a execução periódica da sincronização, executando o *endpoint* criado no LISDAS-6 todos os dias e atualizar os dados das instâncias

LISDAS-11-Get all LIS instances: o ultimo ticket do nosso projeto, adicionámos todas as instâncias do produto, manualmente no *frontend* da nossa *webapp[10]*, cerca de 20, e testámos o nosso processo de sincronização para posteriormente prepararmos a apresentação do projeto á empresa. Neste ticket verificámos o sucesso do desenvolvimento deste projeto, visto que ao testarmos todo o processo de sincronização para estas 20 instâncias, verificamos a finalização com sucesso do mesmo.

5.1.4 Apresentação ao cliente

Finalmente, e após toda a resolução destes tickets, o projeto ficou finalizado e corretamente desenvolvimento. Pronto a ser usado pela empresa.

De seguida, preparou-se uma apresentação do produto ao cliente futuro(empresa), onde começamos por explicar o que desenvolvemos e como desenvolvemos o projeto ao longo do estágio, mostrámos também que foram concluídas com sucesso todas as *timelines* inicialmente definidas e onde mostrados o produto final, com todas as funcionalidades do mesmo a serem executadas sem qualquer problema.

Esta reunião ocorreu no *GoogleMeet[16]* , no dia 11 de Junho de 2021, Terça-Feira entre as 11:30h e as 12:30. Esta plataforma é bastante utilizada pela empresa, como forma de comunicação entre todos os membros na mesma e onde ocorrem *Tech Sharing* frequentes. Estas *Tech Sharing* podem ser feitas por qualquer membro da empresa e o seu objetivo é apresentar a todos um novo software que entregue vantagens ao desenvolvimento e passível de ser usado no em alguns projetos na empresa.

Com esta apresentação termina assim o nosso estágio e o acompanhamento da empresa.

6 Avaliação crítica

Fiquei bastante surpreendido, pela positiva, com a complexidade de produção de um produto a nível empresarial. Eu tinha a noção que seria algo um tanto complexo e progressivo, mas não da maneira que realmente é feito! Aprendi que pequenos passos e bem feitos, levam a grandes produtos com futuro!

A empresa recebeu-me como recebe qualquer um dos novos profissionais que iniciem lá a exercer as suas funções. Houve sempre entre-ajuda. Além de ser apenas um estágio, a equipa e empresa em geral sempre ajudou, perdendo o tempo que fosse necessário e sempre preocupados com o meu bem-estar e progresso.

Conheci e aprendi diversas ferramentas novas e o quão úteis podem ser, ou não, nos diferentes tipos de produtos desenvolvidos.

Houve sempre uma revisão do código produzido e recorrentes apresentações de forma a saber se tudo estava a ser desenvolvido como proposto nos requisitos. O que demonstra o interesse da empresa na garantia que tudo corre como planeado e que tudo está a ser bem feito.

A exigência imposta pela empresa foi bastante bem recebida, visto que nos levou a fazer as coisas bem feitas e otimizadas. Com isto, compreendi que só escrever o código e ele estar funcional não chega! É necessário que o código seja universal, facilmente lido por qualquer pessoa que necessite de agarrar no projeto no futuro e garantir que a execução do projeto e das suas funcionalidades está o mais otimizado possível. Este aspeto garante uma maior eficiência e no futuro um cliente feliz com o produto final desenvolvido!

Tecnicamente, aprendi como programar utilizando a *framework DotNet[1, 2, 3]* e a linguagem de programação *CSharp[12, 13]*, a utilizar corretamente, *ORM's(Object-Relation Mapping)[32]*, neste caso o *Dapper[22]* - experiência que espero um dia voltar a repetir - e como usar algumas das funcionalidades da *Google Cloud Platform[9]*.

Na parte menos técnica, com este estágio desenvolvi também bastante a minha capacidade de trabalho em equipa, a minha comunicação/apresentação a grandes grupos de pessoas e a apresentar produtos a futuros clientes.

Caso tivesse a oportunidade de repetir esta experiência faria-o nesta empresa sem qualquer problema, visto que a experiência foi positiva em todos os pontos.

Referências

[1] dotNet. .net core 101 — Youtube. https://www.youtube.com/watch?v=eIHKZfgddLM&list=PLdo4fOcmZOoWoazjhXQzBKMrFuArxpW80.

[2] Microsoft. Dotnet framework. https://dotnet.microsoft.com.

[3] Microsoft. Dotnet documentation. https://docs.microsoft.com/en-us/dotnet.

[4] Google. Overview to data studio. https://datastudio.google.com/u/0/.

[5] Wikipedia contributors. Google data studio — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Google_Data_Studio, 2021.

[6] PostgreSQL Global Development Group. Postgresql: The world's most advanced open source relational database. https://www.postgresql.org/, 1996.

[7] PostgreSQL Global Development Group. Postgresql: The world's most advanced open source relational database - documentation. https://www.postgresql.org/docs/, 1996.

[8] Wikipedia contributors. Pair programming — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Pair_programming.

[9] Google. Cloud computing services - google cloud. https://cloud.google.com/, 2008.

[10] Wikipedia contributors. Web api — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Web_API, 2021.

[11] OpenJS Fundation. Node.js. https://nodejs.org/en/.

[12] Wikipedia contributors. C sharp(programming language) — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/C_Sharp_(programming_language), 2021.

[13] Microsoft. Csharp documentation. https://docs.microsoft.com/en-us/dotnet/csharp.

[14] Microsoft. Visual studio code - code editing. https://code.visualstudio.com/, 2015.

[15] Slack Technologies. Slack. https://slack.com/intl/en-pt/, 2014.

[16] Google. Google meet. https://meet.google.com/.

[17] Atlasian. Bitbucket - the git solution for professional teams. https://bitbucket.org/product/, 2012.

[18] Atlassian. Jira - issues and project tracking software. https://www.atlassian.com/software/jira#, 2002.

[19] Mozilla. Json - javascript mdn. https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/JSON.

[20] Wikipedia contributors. Agile software development — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Agile_software_development, 2021.

[21] Google. Google maps platform - google developers. https://developers.google.com/maps/documentation, 2005.

[22] Dapper. Dapper tutorial. https://dapper-tutorial.net/dapper.

[23] Mozilla. Html: Hypertext markup language. https://developer.mozilla.org/pt-BR/docs/Web/HTML.

[24] W3Schools. Css tutorial. https://www.w3schools.com/css/.

[25] Microsoft. Typescript language. https://www.typescriptlang.org.

[26] Microsoft. Typescript documentation. https://www.typescriptlang.org/docs.

[27] Wikipedia contributors. Single page application — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Single-page_application.

[28] Google. Angular. https://angular.io.

[29] Tony Tam. Api documentation and design tools for teams. https://swagger.io/.

[30] HashiCorp. Terraform by hashicorp. https://www.terraform.io/.

[31] Google. Cloud scheduler - google cloud. https://cloud.google.com/scheduler.

[32] Wikipedia contributors. Object–relational mapping — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Object%E2%80%93relational_mapping.