App Install & Go

Roberto Filipe Manso Barreto
(nrº 21123, regime diurno)

Orientação de
Luís Gonzaga Martins Ferreira

Licenciatura em Engenharia em Sistemas Informáticos
Escola Superior de Tecnologia
Instituto Politécnico do Cávado e do Ave

Identificação do aluno

Roberto Filipe Manso Barreto
Aluno número 21123, regime diurno
Licenciatura em Engenharia em Sistemas Informáticos

Orientação

Luís Gonzaga Martins Ferreira

Informação sobre o Estágio

Motorline Eletrocelos S.A
Travessa do Sobreiro, 29 Rio Côvo (Sta. Eugénia) 4755-474 Barcelos
Eng. Helder Remelhe

Resumo

Resumo do trabalho realizado. Deve ser sucinto, e cobrir todo o relatório: uma introdução ao problema que se pretendeu resolver, um pequeno resumo da abordagem realizada, e algumas conclusões do trabalho atingido.

Poderão ser criados vários parágrafos, até para que cada um corresponda às três fases de introdução, desenvolvimento e conclusão.

Não é relevante colocar no resumo o local de estágio ou a referência ao curso. Essa informação já consta da capa.

Abstract

This is the translation of the previous text. It should say the exact same thing. Please do not use directly Google Translator.

Agradecimentos

[A secção de agradecimentos é a parte pessoal do documento, e o único sítio onde o aluno pode escrever de forma menos formal, usando o tipo de linguagem que lhe parecer adequado para as pessoas a quem agradece.]

Conteúdo

1. Estado da arte

De forma a organizar todo o trabalho a desenvolver visto que este é dividido com mais uma colega, foi então utilizada a técnica de desenvolvimento ágil, conseguindo assim organizar todas as tarefas entre os elementos de desenvolvimento do projeto.

1.1 Ferramentas de trabalho utilizadas

A organização de todas as tarefas, foi realizada na ferramenta *Github Projetcs*, esta permite ligar um projeto a um repositório de *Github*, conseguindo também personalizar completamente todo o projeto e parâmetros das tarefas o que permite uma organização minuciosa destas.

O Microsoft Excel foi também utilizado para a engenharia de *software* onde foram descritos os requisitos do projeto, *user stories* e também a especificação de casos de uso. Esta ferramenta foi também utilizada para a organização de reuniões com o cliente e redação de tópicos a abordar e abordados nesta.

Para o desenvolvimento do *design do software* foi utilizado a ferramenta *figma*, que permite o *design* de todas as componentes tendo em conta as reais dimensões de um dispositivo. Esta ferramenta permite também criar uma apresentação interativa que consegue demonstrar o comportamento da aplicação como completamente desenvolvida, dando também suporte à implementação.

O *draw.io* foi também utilizado para o desenho das arquiteturas do projeto tendo se revelado de grande auxílio visto que este permite uma grande liberdade no desenho. Esta ferramenta permite também conexão com *github* conseguindo assim facilmente guardar estes projetos e ter acesso a partir de qualquer dispositivo.

A engenharia de *software* foi realizada através da utilização *Visual Studio paradigm*, esta é uma ferramenta muito completa contendo modelos e regras para a engenharia de *software*. Esta ferramenta tornou-se um grande auxílio no desenvolvimento da base de dados, pois é possível desenhar o modelo e exportar para um ficheiro de criação de base de dados.

1.2 Tecnologias utilizadas

1.2.1 Serviços Backend

De forma a realizar a integração entre a aplicação *frontend* e os dados, foi necessário desenvolver uma API para dar suporte a todos os serviços necessários para a aplicação. API sigla para *Application Programming Interface exposes a set of data and functions to facilitate interactions between computer programs and allow them to exchange information.* (Masse, 2011). Estas ferramentas apesar de serem *designed to work with other programs, they're mostly intended to be understood and used by humans writing those other programs* (Jin et al., 2018).

1.2.1.1 Serviços RestFull e SOAP

Os seviços RestFull *expose data as resources and use standard HTTP methods to represent Create, Read,Update, and Delete (CRUD) transactions against these resources* (Jin et al., 2018), sendo que a sua resposta é realizada em JSON *due to its simplicity and ease of use with JavaScript, JSON has become the standard for modern APIs.*(Jin et al., 2018).

Já SOAP é *nothing more than a simple XML-based envelope for the information being transferred, and a set of rules for translating application and platform-specific data types into XML representations*(Snell et al., 2002) sendo que a sua resposta é realizada em XML o que leva a que *It relies heavily on XML standards like XML Schema and XML Namespaces for its definition and function*(Snell et al., 2002). A utilização de XML permite que *two applications, regardless of operating system, programming language, or any other technical implementation detail, may openly share information using nothing more than a simple message encoded in a way that both applications understand.*(Snell et al., 2002)

Por fim foi decidido utilizar Rest devido a ser *much lighter compared to SOAP. It does not require formats like headers to be included in the message, like it is required in SOAP architecture.* Outra maior valia é que *it parses JSON, a human readable language designed to allow data exchange and making it easier to parse and use by the computer. It is estimated to be at around one hundred times faster than XML.*(Halili & Ramadani, 2018).

1.2.1.2 Typescript

O typescript *is a bit unusual as a language in that it neither runs in an interpreter (asPython and Ruby do) nor compiles down to a lower-level language (as Java and Cdo).*(Vanderkam, 2019) isto porque este *compiles to another high-level language, JavaScript.*(Vanderkam, 2019), isto faz com que o Typescript seja visto como *a superset of JavaScript in a syntactic sense.*

Todos os programas JavaScript *are TypeScript programs, but the converse is not true... This is because TypeScript adds additional syntax for specifying types.*. O sistema de tipagens do TypeScript tem como objetivo *to detect code that will throw anexception at runtime, without having to run your code. ... The type checker cannot always spot code that will throw exceptions, but it will try.*(Vanderkam, 2019).

Esta linguagem de programação foi escolhida para o backend devido à sua capacidade de assegurar as tipagens, o que leva a um maior nível de segurança em termos de lidar com dados recebidos, assim como também a agilização do processo de programação devido à sua capacidade de prever a maioria dos erros de código.

1.2.1.3 Logs e Logging

Logs *are a very useful source of information for computer system resource management (printers, disk systems, battery backup systems, operating systems, etc.), user and application management (login and logout, application access, etc.), and security*(Chuvakin et al., 2012). Um Log é *what a computer system, device, software, etc. generates in response to some sort of stimuli.What exactly the stimuli are greatly depends on the source of the log message*(Chuvakin et al., 2012), ou seja perante um estimulo desejado, um log poderá ser gerado. Os dados dos logs são *the intrinsic meaning that a log message has*(Chuvakin et al., 2012), significando isto que estes contêm apenas dados relevantes ao objetivo do log. Logging é o nome que se dá ao processo de geração de logs.

Neste contexto logging poderá ser utilizado para realizar a monitorização de pedidos e erros. Estas informações poderão até auxiliar na toma de decisões sobre o software e em quais funcionalidades deste software colocar mais atenção.

1.2.1.4 Morgan

Morgan é uma biblioteca que permite extrair dados de um pedido, assim como também a criação de logs, este atua como um middleware do servidor, recebendo qual o tipo de log a ser escrito, sendo estes tipos definidos pela biblioteca. Os principais dados obtidos pela biblioteca são a data e hora do pedido, o tipo de pedido, o serviço pedido, os dados recebidos, a resposta devolvida e também a descrição do sistema utilizado para realizar o pedido. Com estes dados é possível saber que plataforma é mais utilizada no software, quais as horas de maior utilização e quais os serviços mais executados, estes dados permitem direcionar mais recursos para uma indicada plataforma e/ou serviço, assim como também escolher os melhores horários de manutenção dos servidores.

1.2.1.5 Envio de emails

Para o envio de emails para os utilizadores, foi utilizada a biblioteca nodemailer,que permite a utilização de um servidor de SMTP para o envio de emails. Esta ferramenta foi escolhida devido a ser uma das mais utilizadas para esta tipo de necessidade, o que permite que exista mais informação sobre a mesma facilitando a resolução e identificação de erros.

Para desenvolver o conteúdo dos emails foi utilizada a ferramenta Tabular Email, esta ferramenta permite realizar o design do conteúdo de um email, sendo possível de seguida exportar o mesmo para html, a dificuldade desta ferramenta é que não permite a utilização de acentuação e visto que o html é gerado por uma máquina este torna-se complicado de navegar e traduzir.

1.2.1.6 Agendamento de tarefas

Um requisito para este projeto é o envio de emails com relatório diário de notificação todos os dias ao final do dia. Para realizar este primeiramente foi pesquisado que ferramentas existem para realizar este tipo de ações, pelo que foram encontradas o cronetab e o node-cron. A grande diferença este estas duas ferramentas é, o cronetab funciona a nivel de servidor sendo que sempre que se encontra na hora programada, este executa um comando indicado, este comando poderá por exemplo executar um código para enviar emails. Já o node-cron trata-se de uma biblioteca de NodeJs que trabalha com base no crontab, este permite o fácil agendamento de tarefas de forma programática assim como também a indicação do código a ser executado sem necessidade de criar comandos de execução de código.

Visto que a hora de execução do código de envio de emails poderá variar e necessitar de reprogramação foi então optado pela utilização do node-cron devido á sua facilidade de utilização, agilizando assim o processo de reprogramação de horas de envio de relatório.

1.2.1.7 Encriptação de passwords

De forma a garantir a segurança das passwords dos utilizadores é necessário encriptar estas, a encriptação poderá ser feita manualmente ou com o auxílio de ferramentas, a grande diferença é que manualmente poderá não se obter uma cifra tão segura como com o auxílio de uma ferramenta, sendo assim foi decidido utilizar uma ferramenta para encriptar passwords, a ferramenta escolhida foi bcrypt, esta foi escolhida devido a ser vastamente utilizada e até ao momento sem problemas em relação à sua cifra sendo que não existem registos de ataques bem sucedidos a esta ferramenta. Esta ferramenta oferece um conjunto de métodos sendo tendo sido utilizados os métodos de cifra e de comparação. O método de cifra permite através de um valor, chamado salt, indicar a complexidade a aplicar sobre a cifra, sendo de seguida devolvida a password cifrada. O método comparação permite comparar uma password cifrada com uma password sem cifra, devolvendo verdadeiro ou falso conforme as passwords sejam iguais ou não.

1.2.1.8 Cifragem de configurações do servidor

De forma a garantir um nivel de segurança maior foram realizadas pesquisas sobre as principais falhas de segurança no NodeJs, pelo que foi descoberto que as principais formas de ataque a esta ferramenta é o desenvolvimento de bibliotecas de malware e o ataque às bibliotecas com o objetivo de obter dados de acesso a servidores que se encontram nos ficheiros de configuração.

Por norma todas as configurações de servidores são colocadas num ficheiro env, este ficheiro no momento de iniciar o servidor é utilizado para carregar todas as variáveis para o ambiente do mesmo, sendo assim qualquer um com acesso ao ficheiro ou às variáveis de ambiente poderá ver todas as configurações do servidor.

A solução mais indicada para este problema é a cifragem do ficheiro env e das variáveis de ambiente. A biblioteca mais utilizada para este objetivo é a secur-env, esta permite realizar a cifragem de um ficheiro indicando uma password. A password deverá ser indicada no processo de inicialização do servidor de forma a ser possível ao mesmo decifrar o ficheiro, sendo que a gestão das variáveis cifradas passa então a estar encarregue desta biblioteca.

Mesmo com esta solução existem possibilidades de ataque, pois é possível ver o histórico do terminal do servidor, pelo que e possível obter a password escrita no mesmo, para resolver este problema é indicada a biblioteca readline, pois esta possui o modo de password que apaga o histórico do terminal sempre que utilizado, esta contém a vertente assincrona, readline e a vertente síncrona, readline-sync. Para o projeto foi utilizada a versão síncrona da biblioteca visto que o objetivo é o servidor apenas inciar após a indicação da password, sem nenhum serviço a correr em simultâneo.

1.2.1.9 Firebase

1.3 Frontend

Um dos requisitos do projeto é o desenvolvimento do frontend utilizando a ferramenta Flutter, visto que a empresa já utiliza esta ferramenta, sendo assim necessário o aprendizado desta ferramenta e da sua linguagem de programação o dart.

1.3.1 Flutter

EXPLICAR O QUE E FLUTTER

1.3.2 Dart

EXPLICAR O QUE É DART

1.3.3 Processo de aprendizagem

De forma a realizar a aprendizagem da ferramenta foi então procedido para a desenvolvedora da mesma, a google, esta dispõe de uma lista de workshops e projetos a seguir de forma a aprender as bases da ferramenta, sendo a lista a seguinte:

1. MDC-101 no Flutter: noções básicas dos componentes do Material Design | Google Codelabs
2. MDC-102 no Flutter: estrutura e layout do Material Design | Google Codelabs
3. MDC-103 Flutter: temas do Material Design com cores, formas, elevação e tipo | Google Codelabs
4. MDC-104 Flutter: componentes avançados do Material Design | Google Codelabs
5. Apps adaptáveis no Flutter | Google Codelabs

Após a realização destes workshops, foi possível entender como funciona a base desta ferramenta e também encontrar fontes para pesquisa de widgets da comunidade a nível gráfico e funcional, provando estes serem de grande auxílio no desenvolvimento da aplicação frontend.

1.3.4 Qualidade de código

A qualidade do código desenvolvido é de extrema importância de forma a possibilitar e facilitar a manutenção da solução desenvolvida, assim como também a melhoria da segurança da mesma. Esta permite a perceção de objetivo de código assim como também a estruturação do mesmo de acordo com normas estabelecidas perante a comunidade. A qualidade de código tem como objetivo também diminuir a complexidade do mesmo, pois nem sempre documentacao e estruturação é o suficiente para o código ser de qualidade, sendo que este deverá ser simplificado de forma a ser interpretável. Todos estes pontos poderão ser implementados através do uso de design patterns, documentação e testes de código.

1.3.4.1 Design patterns

1.3.4.2 Documentação

De forma a ser possível manter todo o projeto desenvolvido foi criada documentação a diferentes níveis, sendo esta desenvolvida a nivel de serviços explicando o objetivo do serviço e que dados este recebe, como também a nivel de código explicando o código desenvolvido em cada script existente. Para estes níveis de documentação foram utilizadas diferentes ferramentas, para documentação de serviços, foi utilizada a ferramenta swagger e para a documentação foi utilizado typedoc.

1.3.4.3 Typedoc

Typedoc é uma ferramenta que faz utilização de comentários de código para gerar a sua documentação, esta documentação utiliza chaves específicas para detetar as informações de documentação, estas permitem também criar categorias para melhor organizar toda a documentação. Esta documentação permite também a interligação entre si mesma permitindo ao visualizador desta seguir todo o processo. Após a realização de geração de documentação, a ferramenta gera um website onde é possível navegar por toda a documentação gerada.

1.3.4.4 Swagger

Swagger é uma ferramenta que permite gerar documentação a nivel de serviços, esta documentação é acessível a partir do mesmo servidor indicando uma rota para o mesmo, evitando assim outro servidor para hospedar a documentação, a base de toda a documentação encontra-se em um ficheiro no formato json. Esta documentação poderá ser gerada a partir de comentários a nivel de código ou a partir de um ficheiro em formato json como mencionado anteriormente. Este ficheiro em formato json poderá ser mantido manualmente ou automaticamente.

1.3.4.5 Testes de código

Aquando o fim do desenvolvimento de cada serviço é necessário testar este de forma a verificar se a funcionalidade se encontra de acordo com o desejado e/ou se existem erros de código. Para realizar estes testes poderão ser utilizadas ferramentas de auxílio ou então poderão ser realizados manualmente. O grande problema de testes manuais é que são exaustivos devido a serem longos e propensos a erros, pelo que estes testes são realizados em menor escala do projeto. Para os testes deste projeto foram utilizadas ferramentas de auxílio, sendo as ferramentas escolhidas mocha e chai.

A realização de estes de código foi muito importante pois com esta ferramenta foi possível encontrar erros de lógica de negócio bem como também erros de código tanto a nível da formulação de respostas como a nivel de código.

1.4 Planificação do trabalho

De forma a obter uma visão geral do projeto e uma previsão de finalização foi realizada uma planificação expectável de tarefas.

Figura 1.1: Planeamento de sprints

Bibliografia

Chuvakin, Dr. Anton A., Kevin J. Schmidt & Christopher Philips. 2012. Logging and log management.

Halili, Festim & Erenis Ramadani. 2018. Web services: A comparison of soap and rest services. *Modern Applied Science* 12. 175. doi:10.5539/mas.v12n3p175.

Jin, Brenda, Saurabh Sahni & Amir Shevat. 2018. *Designing web apis*. O'Reilly Media, Inc.

Masse, Mark. 2011. *Rest api design rulebook*. O'Reilly Media, Inc.

Snell, James., Doug. Tidwell & Pavel. Kulchenko. 2002. *Programming web services with soap*. O'Reilly & Associates.

Vanderkam, Dan. 2019. Effective typescript 62 specific ways to improve your typescript.