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

Este documento trata o processo de análise, especificação e implementação da solução Install&Go. Esta vem resolver o problema de comunicação entre a empresa Motorline e os seus técnicos, uma vez que no acontecimento de um problema estes deverão telefonar para a empresa, o que gera sobrecarga.

A resolução deste problema vem com o desenvolvimento de uma plataforma de fórum onde as empresas podem registar os seus técnicos, sendo que estes conseguem então expor as suas questões no fórum onde outros técnicos de outras empresas e/ou Motorline poderão auxiliar. Esta permite também que no caso de um técnico possuir um problema já resolvido, este poderá pesquisar pela solução dada ao mesmo na plataforma.

O desenvolvimento desta solução proveu a aquisição de novas capacidades técnológicas como o desenvolvimento cross-platform e as suas frameworks, sendo que destas foi explorado flutter. Este permitiu também a assimilar capacidades como análise, especificação de projetos e comunicação com clientes. Por fim foi possível desenvolver completamente a solução de acordo com as necessidades e expectativas do cliente.

Abstract

This document relates the analysis, specification and development of the Install&Go software. This software solves the comunication problem between Motorline and their professionals, since that in the event of a problem they must call the company, which generates an overload.

The solution to this problem comes with the development of a forum where companies can register their professionals, and these cand then expose their questions so that other professionals from other companies or Motorline it self can help. This also allow that if a professional has a problem that was already solved, he can search for the solution on the platform.

The development of this solution provided the acquisition of new technical skills such as cross-plataform development and its frameworks, of which flutter was explored. It also allowed the assimilation of skill such as project analysis, specification and comunication with clients. At the end, it was possible to fully develop the solution according to the client's needs and expectations.

Agradecimentos

Em primeiro lugar, gostaria de agradecer à minha família, com destaque à minha mãe, ao meu padrinho, aos meus avós e à minha namorada, que com todo o carinho orientaram-me nesta caminhada que foi a licenciatura.

Também, gostaria de salientar um caloroso agradecimento à empresa Motorline, por me terem sempre feito sentir um membro da equipa, com destaque ao supervisor Helder Remelhe que sempre se apresentou disponível para eventuais dúvidas que surgissem no desenvolvimento do projeto.

Por fim, mas não menos importante, gostaria de enfatizar toda a orientação, disponibilidade, conversa e ensinamentos proporcionados pelo doutor professor Luís Ferreira no de e também aos meus colegas de curso Henrique Cartucho e João Castro por todo o apoio e todos os momentos vividos ao longo da licenciatura.

Conteúdo

0.1 Arquitetura de sistema

A figura 1 expõe a arquitetura do sistema que indica as principais componentes. Entre estas, é possível visualizar a aplicação *frontend*, que realiza pedidos a uma aplicação *backend* e espera respostas. O *backend* é composto por uma *api rest* que recebe e responde aos pedidos e por uma base de dados a qual recebe *queries* e devolve dados para a *api rest*.

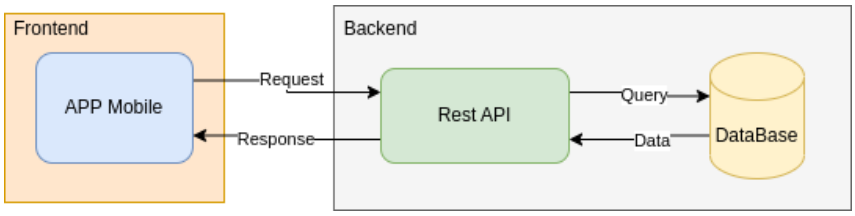

Figura 1: Arquitetura do sistema

0.1.1 Arquitetura de funcional

A especificação da implementação da *api rest* foi realizada através de uma arquitetura de *backend*(Figura 2). Aqui, é possível visualizar que sempre que a *api* recebe um pedido, este é redirecionado primeiramente para o *router*. O *router* tem como função identificar a rota referente ao pedido e deslocar para os respetivos *middlewares*.

Os *middlewares* têm como objetivo realizar todas as ações necessárias antes de proceder à execução do código de rota. Os *middlewares* existentes são o *SessionTokenValidator*, valida a sessão do utilizador a realizar o pedido, de forma similar, o *middleware RefreshTokenValidator*, valida a sessão principal do utilizador e por fim, o *middleware RoleValidator*, valida se o utilizador que efetua o pedido tem cargos suficientes para tal. Na eventualidade de não existir nenhum impedimento, o pedido é direcionado para o *controller*.

O *controller* extrai os dados do pedido, verifica se os dados obrigatórios existem e cumprem as regras de negócio e encaminha o pedido para o serviço. O serviço, em caso de necessidade, irá proceder à interação com a base de dados, esta realiza diversas ações como, obter, atualizar, apagar e inserir dados. Por fim, após todo o código de serviço ser executado, a resposta é formulada e devolvida para o utilizador.

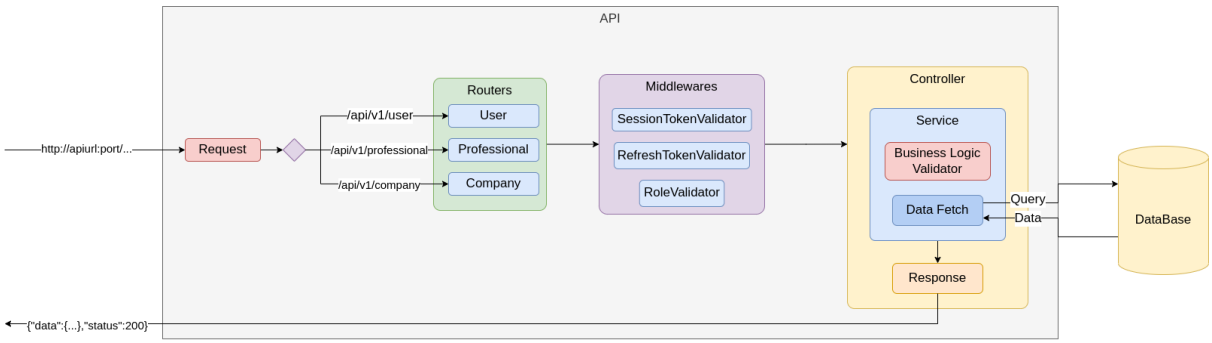

Figura 2: Arquitetura do funcional

0.1.2 Arquitetura de componentes

A arquitetura de componentes contém todos os serviços que deverão ser implementados na *api frontend*, com a identificação dos atores que poderão realizar estes pedidos. Esta foi desenvolvida após uma análise de todos os dados necessários para suporte do *frontend*.

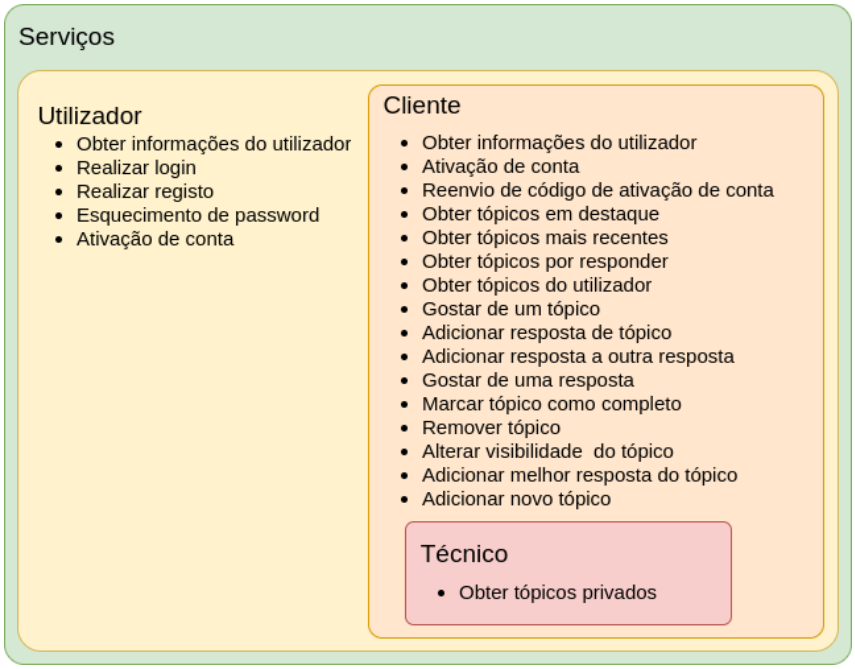

Figura 3: Arquitetura de componentes

0.1.3 Tabela de endpoints

Com o propósito de evitar colisões de *endpoints* durante a implementação, foi desenvolvida uma tabela de *endpoints*. Esta possui dados semelhantes à arquitetura de componentes (Figura 3), mas para cada serviço é indicada a rota e o método a utilizar.

Tabela 1: Tabela de endpoints

Serviço	Ator	Rota	Método
Obter informações do utilizador	Cliente	baseurl/client/:uid	GET
Realizar login	Utilizador	baseurl/login	POST
Realizar registo	Utilizador	baseurl/register	POST
Esquecimento de *password*	Utilizador	baseurl/forgot-*password*	GET
Ativação de conta	Cliente	baseurl/client/:uid/activate	POST
Reenvio de código de ativação de conta	Cliente	baseurl/client/:uid/new-code	GET
Obter tópicos em destaque	Cliente	baseurl/client/topics/featured	GET
Obter tópicos mais recentes	Cliente	baseurl/client/topics/latest	GET
Obter tópicos por responder	Cliente	baseurl/client/topics/to-answer	GET
Obter tópicos do utilizador	Cliente	baseurl/client/topics	GET
Obter tópicos privados	Técnico	baseurl/professional/topics/private	GET
Gostar de um tópico	Cliente	baseurl/client/topics/:topicId/like	PUT
Adicionar resposta a tópico	Cliente	baseurl/client/topics/:topicId/answer	POST
Adicionar resposta a outra resposta	Cliente	baseurl/client/answers/:answerId/	POST
Gostar de uma resposta	Cliente	baseurl/client/answers/:answerId/like	PUT
Marcar tópico como completo	Cliente	baseurl/client/topics/:topicId/completed	PUT
Remover Tópico	Cliente	baseurl/client/topics/:topicId/	DELETE
Alterar visibilidade do tópico	Cliente	baseurl/client/topics/:topicId/visibility	PUT

Continued on next page

Tabela 1: Tabela de endpoints (Continued)

Adicionar melhor resposta do tópico	Cliente	baseurl/client/topics/:topicId/answers/ :answerId/best-answer	PUT
Adicionar novo tópico	Cliente	baseurl/client/topics/	POST

Bibliografia

Bracha, Gilad. 2015. The dart programming language.

vanden Broucke, Seppe & Bart Baesens. 2018. *Practical web scraping for data science*. Apress. doi:10.1007/978-1-4842-3582-9.

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

Developers, Android. 2023. Handling android app links. `https://developer.android.com/training/app-links#android-app-links`. [Abril-2023].

Firebase. 2023. Firebase dynamic links | firebase documentation. `https://firebase.google.com/docs/dynamic-links`. [Abril-2023].

Gamma, Erich, Richard Helm, Ralph Johnson & John Vlissides. 2009. *Design patterns: Elements of reusable object-oriented software*. Addison-Wesley.

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.

Mainkar, Prajyot & Salvatore Giordano. 2019. *Google flutter mobile development quick start guide : Get up and running with ios and android mobile app development*. Packt Publishing Ltd.

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

Mead, Andrew. 2018. *Learning node.js development : learn the fundamentals of node.js, and deploy and test node.js applications on the web*. Packt Publishing Ltd.

Moroney, Laurence. 2017. *The definitive guide to firebase*. Apress. doi:10.1007/978-1-4842-2943-9.

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.