DEPARTAMENTO DE ELETRÓNICA, TELECOMUNICAÇÕES E INFORMÁTICA

SERVICE STATUS UA

ENGENHARIA INFORMÁTICA

Autores:
Eleandro Laureano, 83069
Frederico Avó, 79900
Gonçalo Pereira, 93310
José Mello, 87456

Orientador:
Prof. Cláudio Teixeira

Junho, 2021

Abstract

Dada a quantidade de serviços disponibilizados pela Universidade de Aveiro, pela complexidade na utilização e pela falha na comunicação com os seus utilizadores, ou seja, pela falta de feedback dos mesmos acerca dos seus períodos de indisponibilidade, houve uma necessidade em se resolver este problema de forma a que houvesse uma melhor comunicação com toda a comunidade académica.

Sendo assim, surge o *Service Status UA* que tem o propósito de resolver os problemas anteriormente citados de forma a facilitar a vida dos seus utilizadores oferecendo uma interface *user-friendly*. Então, para isto, este projeto foca-se em providenciar uma lista de todos os serviços e sistemas que fazem parte da Universidade de Aveiro, onde o utilizador pode começar por filtrá-los pela barra de pesquisa, pode ver as próximas manutenções que estão agendadas e por fim, também pode ver os serviços disponíveis e indisponíveis em tempo real e em quanto tempo os mesmos voltarão a estar operacionais.

Keywords

- Status
- Service
- Health
- Server

Agradecimentos

Somos extremamente gratos ao nosso orientador Cláudio Teixeira por orientar-nos da melhor forma possível e também ao seu colega de trabalho Dimitri Silva que nos deu o input necessário e mostrou-se muito útil para ajudar-nos durante o desenvolvimento deste projeto.

Abreviações

- **API** - Application Program Interface
- **SCOM -** System Center Operations
- **SDK -** Software Development Kit
- **DNS -** Domain Name System
- **REST -** Representational State Transfer
- **JS -** Javascript
- **BD -** Base de Dados
- **CSS -** Cascading Style Sheets
- **HTML -** HyperText Markup Language
- **SQL -** Structured Query Language
- **JSON -** JavaScript Object Notation
- **XML -** Extensible Markup Language

1. Introdução

1.1 Contexto

Este projeto foi desenvolvido para a cadeira de Projeto em Informática no curso de Engenharia Informática da Universidade de Aveiro. Além disso, foi orientado pelo Prof. Cláudio Teixeira.

Sendo assim, o objetivo foi de desenvolver uma aplicação web que mostrasse a comunidade sobre os períodos de indisponibilidade dos serviços académicos da Universidade de Aveiro, bem como os períodos de manutenção dos mesmos e foi realizado durante o segundo semestre do ano letivo 2020/2021.

1.2 Motivação

Então, dada a quantidade e a complexidade dos serviços oferecidos à comunidade académica, sentia-se a necessidade de melhorar a comunicação entre eles, relativamente ao estado de operação dos serviços e dos sistemas disponíveis dentro da Universidade. E, para melhorar essa comunicação seria necessário informar aos utilizadores das manutenções agendadas acerca de certos serviços e sistemas que poderão ou não causar a sua indisponibilidade.

De igual modo, a aplicação web também poderá ser utilizada para informar em tempo real de problemas que estejam a ocorrer em determinados serviços que causem a sua indisponibilidade.

1.3 Objetivos

Os objetivos para este projeto são de construir uma aplicação web que consiga fornecer informação relevante acerca dos serviços e sistemas da Universidade de Aveiro, relativamente à sua indisponibilidade, dando visibilidade a toda a gente. Além disso, deve comunicar com o utilizador de forma eficaz e dar a possibilidade de ver a estimativa de tempo para a resolução de um determinado problema e de ver datas de futuras manutenções de um determinado serviço.

Por último, deve possuir um **Admin** que será responsável por fornecer as datas das manutenções agendadas e também as estimativas de tempo para a resolução dos problemas.

1.4 Estrutura

Para além deste primeiro capítulo introdutório, este documento contém mais 5 capítulos. No capítulo 2, abordamos acerca dos requisitos e arquitetura do sistema, incluindo os requisitos funcionais e não funcionais, passando pelos atores e casos de uso, até a descrição da arquitetura do projeto. Em seguida, no capítulo 3, é abordado o estado da arte, ou seja, outros serviços que se assemelham a esta aplicação web e posteriormente fala-se da tecnologia utilizada para a criação e desenvolvimento deste projeto. No capítulo 4, é mostrada como foi feita a implementação desta aplicação web e as *features* que inclui. No capítulo 5, é feito um sumário acerca do projeto, mostra-se os principais resultados e aborda-se sobre o trabalho futuro e possíveis *features* que podem ser incluídas no mesmo. Por fim, no 6º e último capítulo mostra-se as referências que foram utilizadas e consultadas para a realização deste projeto.

2. Requisitos e Arquitetura

2.1 Requisitos do Sistema

Nesta seção, iremos falar dos requisitos do nosso sistema, tanto os funcionais como os não funcionais e explicar detalhadamente a arquitetura desenhada especificamente para o nosso sistema.

2.1.1 Requisitos Funcionais

- Histórico de falhas de cada serviço
- Apresentar uma estimativa do tempo de resolução do problema
- Apresentar as datas das manutenções agendadas
- Autenticação somente a nível do administrador da aplicação web

2.1.2 Requisitos Não Funcionais

- **Performance:** é o quão rápida a aplicação web responde às ações efetuadas pelos utilizadores.
  - **Prioridade**: Média

- **Usabilidade:** é a capacidade do sistema providenciar aos utilizadores condições para serem capazes de realizar as tarefas de forma eficaz e eficiente enquanto desfrutam da sua experiência.
  - **Prioridade**: Média

- **Integridade dos dados:** é a capacidade de garantir a precisão e consistência dos dados apresentados.
  - **Prioridade**: Alta

- **Intuitivo:** a aplicação web deve ser autoexplicativa.

➢ **Prioridade**: Média

❖ **Disponibilidade:** é a capacidade que o sistema tem de estar disponível para os utilizadores sempre que eles desejarem consultar a informação respetiva a um sistema ou serviço.

➢ **Prioridade**: Alta

2.1.3 Atores e Casos de Uso

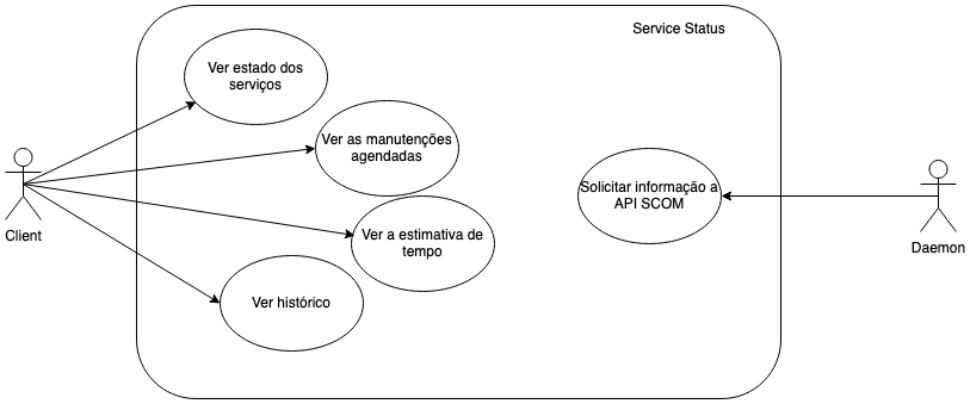
Neste projeto, os utilizadores considerados como público-alvo são qualquer pessoa dentro da comunidade académica da Universidade de Aveiro, tal como, docentes, estudantes, investigadores, entre outros que são os clientes da aplicação web. Além disso, a aplicação web está disponível para utilizadores fora da comunidade académica da Universidade de Aveiro.

Sendo assim, os atores do sistema são:

- **Clientes:** os utilizadores do sistema que fazem parte da comunidade académica.
- **Daemon:** é um dispositivo externo que é responsável por fazer pedidos a API dentro de um intervalo de tempo.

Além disso, os use cases são os seguintes:

- Ver o estado dos serviços.
  - ➢ **Prioridade**: Alta
- Ver histórico.
  - ➢ **Prioridade**: Média
- Solicitar informação à API SCOM.
  - ➢ **Prioridade**: Alta
- Ver a estimativa de tempo.
  - ➢ **Prioridade**: Alta
- Ver as manutenções agendadas.
  - ➢ **Prioridade**: Alta



Fig, 1 - Use Cases

2.2 Arquitetura do Sistema

Nesta seção, iremos abordar o que foi utilizado e implementado no nosso projeto, mostrando diagramas para ajudar a complementar e melhorar a compreensão.

2.2.1 Domain Model

Em seguida, temos o seguinte diagrama para ajudar na compreensão do seguinte processo:

O cliente consulta a informação que está disponível ao interagir com a aplicação web. Esta aplicação web comunica com o *backend* para a obtenção dos seus dados que já estão guardados na BD. Estes dados, que foram guardados pelo *Daemon*, que age como um Agente, sendo um dispositivo externo que corre em *background* e faz pedidos de 2 em 2 minutos a **API SCOM**, de forma a atualizar os dados dos serviços e sistemas e ser possível guardá-los na BD para posteriormente mostrá-los aos utilizadores.

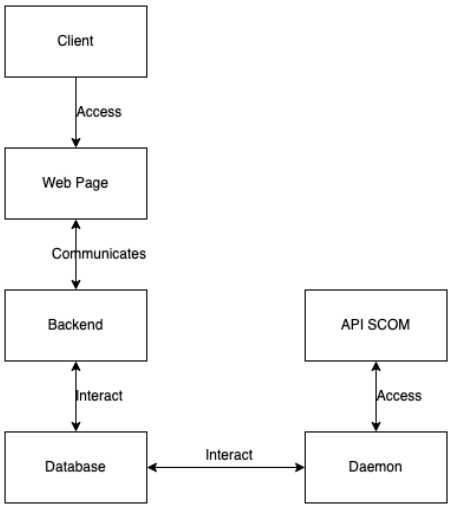


Fig. 2 - Domain Model

2.2.2 Arquitetura

Em seguida, este próximo diagrama relativamente a arquitetura, representa essencialmente como o nosso sistema irá funcionar.

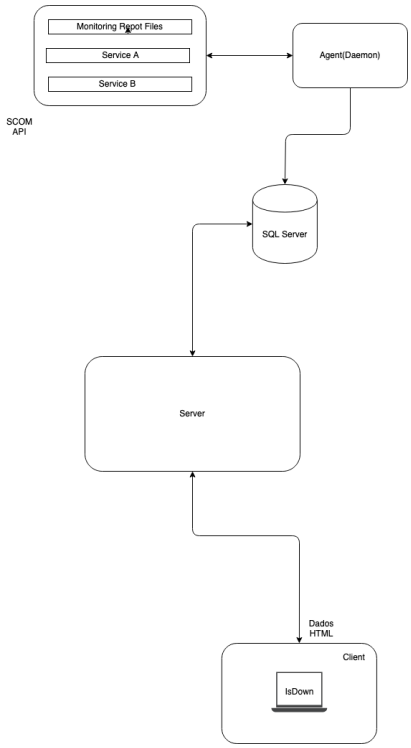


Fig. 3 - Arquitetura

3. Estado da Arte

3.1 Projetos Relacionados

No que diz respeito aos projetos relacionados, irá mostrar-se em seguida exemplos de aplicações web que são idênticas à nossa. As principais diferenças são, que na nossa aplicação web é possível consultar as datas em que há manutenções agendadas e é possível ver a estimativa de tempo para a resolução de um problema dada a indisponibilidade de um serviço.

3.1.1 AWS Service Health Dashboard

É uma plataforma da Amazon, com o propósito de ver o estado dos serviços proporcionados por eles e que estão distribuídos por diferentes localizações, desde a América, tanto do Sul como do Norte, passando pela Europa, África até o Oriente Médio. Então, o problema acaba por ser que não envia notificações de serviços específicos para os seus clientes e o histórico dos serviços encontra-se no fundo da página principal ao invés de ser numa página em separado.

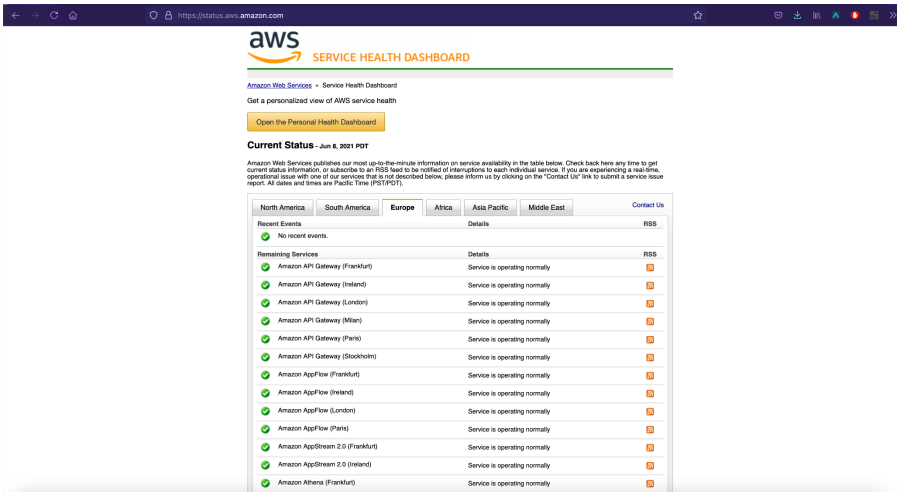


Fig. 4 - AWS

3.1.2 Google Cloud Status Dashboard

É uma plataforma proporcionada pelo Google, que também fornece informação relativamente ao estado dos seus serviços. Além disso, possui um histórico de falhas onde especifica o tempo em que o serviço estava indisponível e o mostra de uma forma muito mais prazerosa ao olho do utilizador dada a sua simplicidade e brevidade, e isto deve-se também ao facto de não possuir uma página inicial muito extensa. Apesar disso, esta plataforma não providencia uma estimativa de tempo para a resolução de um problema e não é possível a subscrição de determinados serviços para a recepção de notificações.

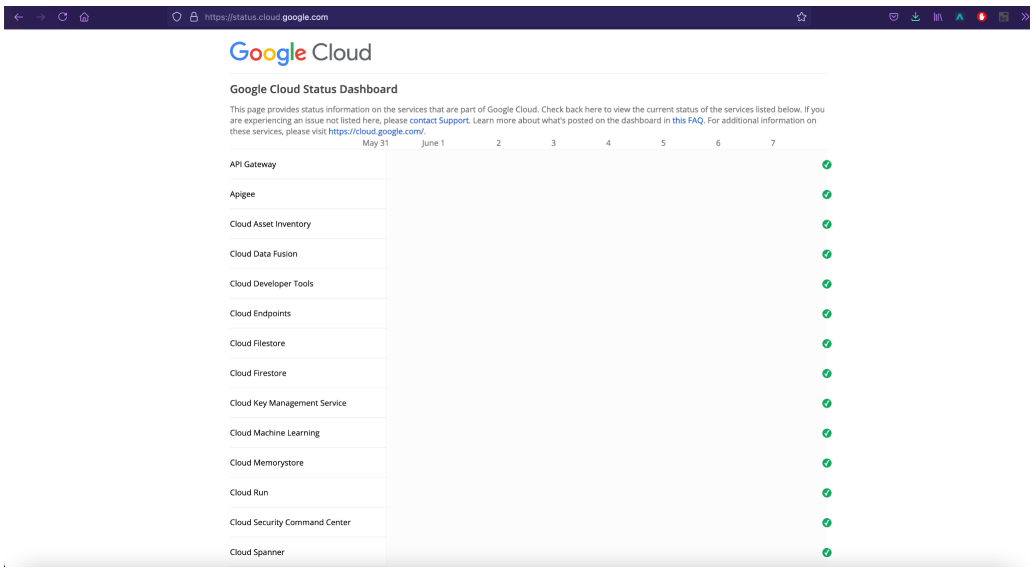


Fig. 5 - Google Cloud

3.2 Tecnologia Utilizada

Nesta seção, irá se proceder a uma explicação da tecnologia utilizada para a criação e desenvolvimento deste projeto.

3.2.1 ReactJS

O **ReactJS** é uma biblioteca ou *framework* do **JS** de código aberto e com o propósito de criar interfaces para os utilizadores em aplicações web. Esta escolha

para o *frontend* do projeto foi feita pelo nosso orientador, apesar de existirem outras opções tais como o **HTML**. Entretanto, nós decidimos continuar com a escolha apesar de ninguém no grupo estar familiarizado. Claramente que foi um desafio, e que estivemos dispostos a aceitar. Além disso, percebemos que o **ReactJS** era uma boa escolha por causa das suas vantagens, tais como, ter uma melhor performance, contém *hot reloading* que atualiza a aplicação web em menos de 1 segundo quando existem mudanças no código mesmo que sejam projetos grandes e complexos. Além disso, é flexível, é fácil de migrar entre versões e é fácil de aprender, desde que o programador tenha um *background* com **HTML** e **CSS**.

3.2.2 .NET

O **.NET**, que antigamente chamava-se **.NET Core**, é um *framework* livre e de código aberto para diversos sistemas operacionais e que foi desenvolvido pela **Microsoft** e suporta o **Visual Basic .NET**, o **F#** e o **C#** como linguagens de programação.

Então, mais uma vez, não havia nenhum tipo de *background* ou de familiaridade com este *framework* ou com alguma destas linguagens de programação por parte dos elementos do grupo, mas pelo facto do **C#** possuir diversas familiaridades com a linguagem de programação **Java,** esta acabou por ser a escolha para o desenvolvimento do projeto, visto que todos os elementos do grupo possuem *background* com a mesma.

3.2.3 Persistence: SQL

Em seguida, a BD em **SQL** foi a escolha para o uso e criação de uma base de dados na plataforma da **Microsoft Azure** foi fácil, devido a grande familiaridade por parte de todos os elementos do grupo e as suas vantagens tais como, alto desempenho na sua memória, processamento rápido e inteligente de consultas avançadas, e pode servir para uma grande variedade de aplicações por permitir processar dados relacionais e não relacionais, tais como **JSON** e **XML**.

3.2.4 SCOM

A API denominada **SCOM** é a interface utilizada para a criação e desenvolvimento deste projeto. Esta API foi criada e desenvolvida pelo orientador deste projeto, o Prof. Cláudio Teixeira, utilizando o **SDK .NET** do **SCOM** da **Microsoft**. Além disso, é importante realçar que a API utilizada por nós e a interface providenciada pela **Microsoft** possuem o mesmo nome.

Sendo assim, a utilização desta **API** é de extrema importância porque é por meio dela que se obtém a informação que diz respeito aos domínios e aos serviços e sistemas que existem na Universidade de Aveiro.

Em seguida, as imagens irão mostrar os *endpoints* utilizados por nós para se conseguir obter os dados citados anteriormente.

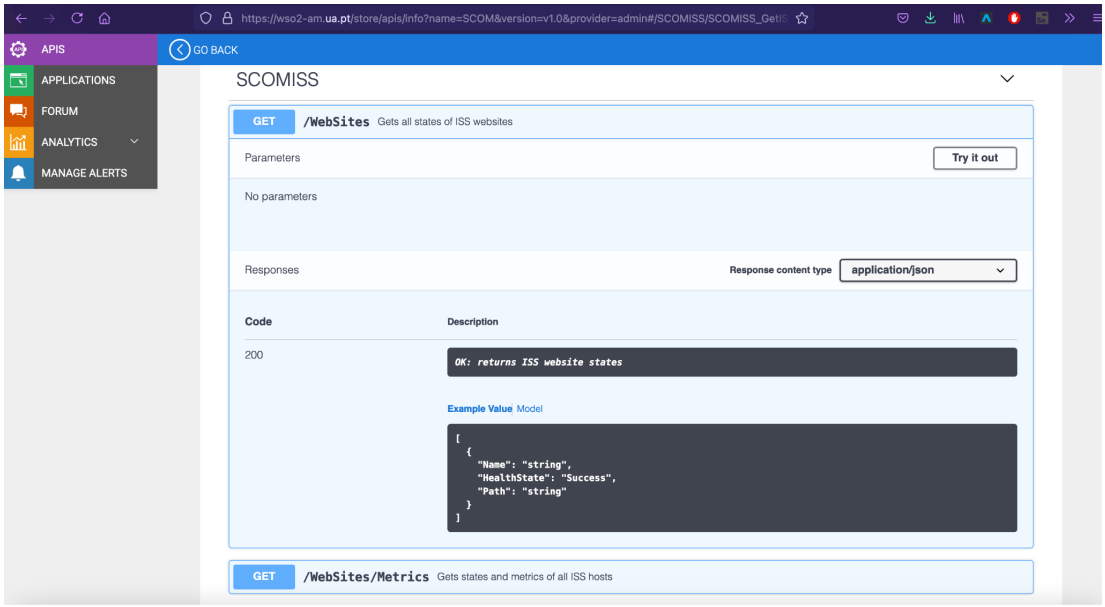


Fig. 6 - WebSites Endpoint

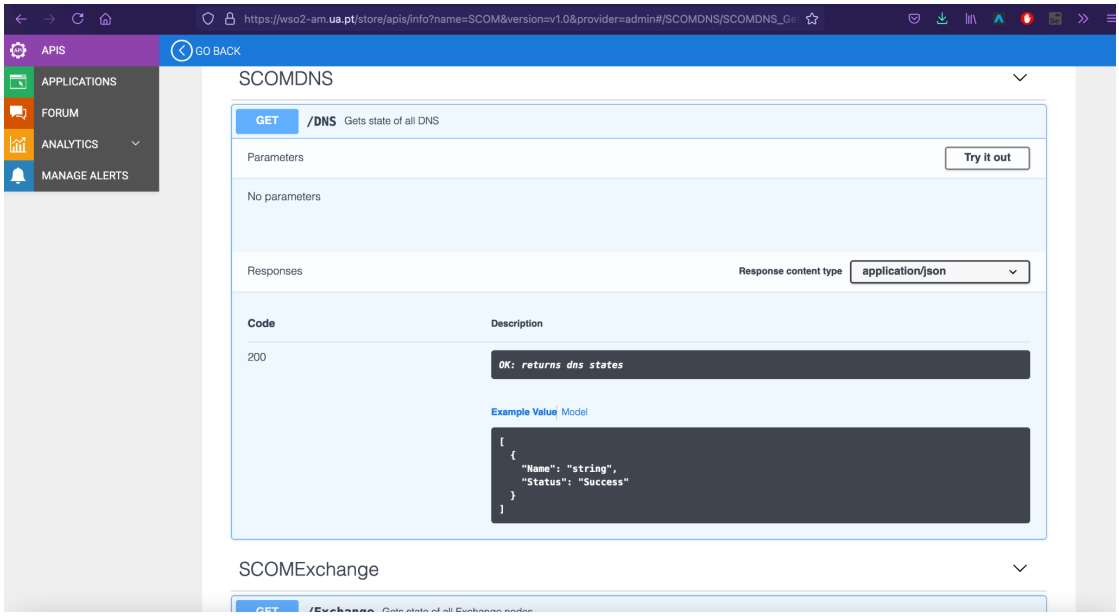


Fig. 7 - DNS Endpoint

3.2.5 Modelo Tecnológico

O diagrama a seguir, representa a arquitetura do nosso sistema, assim como as tecnologias utilizadas em cada camada.

Para a camada do cliente, a aplicação web foi desenvolvida em **ReactJS** que irá comunicar com o *backend*, onde a escolha feita foi o framework **.NET** com o **C#** como linguagem de programação, como dito anteriormente e indo recolher os dados na BD em **SQL** da plataforma **Microsoft Azure** que foram guardados pelo Agente, ou seja, pelo *Daemon***,** com os dados recolhidos sendo providenciados pela API **SCOM**.

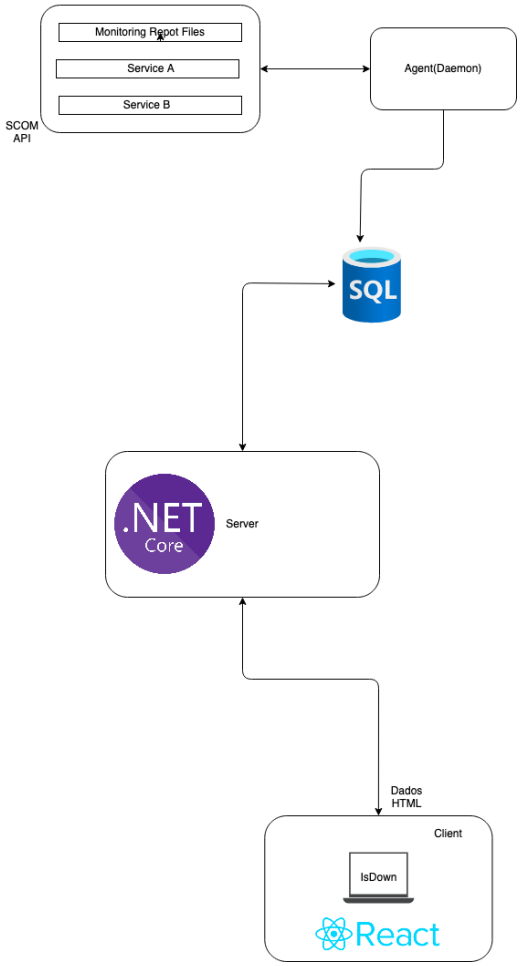


Fig. 8 - Modelo Tecnológico

3.2.6 Ferramentas: Discord, Google Docs and Microsoft Teams

Em relação às ferramentas de colaboração entre a equipa, temos em primeiro, o **Discord** que inicialmente foi uma plataforma criada para comunidades de jogos, hoje em dia, acaba por ser para muito mais além disso e é bastante útil para outras comunidades e para a comunicação contínua entre elementos de um grupo, contendo a partilha de documentos, a criação de canais tanto para texto ou para voz, que acaba por ajudar aos elementos do grupo a se distribuírem se estiverem a trabalhar em partes diferentes do projeto. Além disso, inclui chamadas, videochamadas incluindo partilha de ecrã e por fim a integração de serviços externos como o **Github** ou **Gitlab**. E por isso, foi a nossa escolha como plataforma principal de comunicação.

Em seguida, a plataforma de escolha para a partilha de documentos tais como, as apresentações em Power Point e o relatório do trabalho, utilizou-se o **Google Docs** pela familiaridade que os elementos do grupo têm e por ser uma plataforma que permite a edição dos documentos em tempo real, permitindo no final guardar os documentos em formato Word ou até PDF. Além disso, também é uma plataforma que funciona de forma síncrona e assíncrona.

Por fim, o **Microsoft Teams** foi a plataforma escolhida para as reuniões semanais com o nosso orientador, que suporta a partilha de documentos, a colaboração com um bate-papo, videochamadas e armazenamento de arquivos.

3.2.7 Ferramentas: Microsoft Azure and Github

A seguir, no que diz respeito às ferramentas utilizadas para guardar a parte de construção de código, realizar o *web deploy* do mesmo e a criação de serviços, utilizamos tanto a **Microsoft Azure** como o **Github.**

Em primeiro lugar, a **Microsoft Azure** é uma plataforma destinada à criação de aplicações e serviços que baseiam o seu funcionamento em conceitos de computação em nuvem. Além disso, serviu como plataforma para a publicação do nosso projeto e para a criação da BD em **SQL** e do *Daemon* como dispositivo externo.

Em segundo lugar, temos o **Github** que é uma plataforma que serviu para hospedar o código desenvolvido no projeto e para controle de versões do mesmo.

4. Implementação

Nesta seção, irá se abordar acerca dos recursos implementados e desenvolvidos neste projeto.

4.1 Services

A tabela **Services** é a tabela principal da aplicação web, é o foco principal onde os utilizadores irão obter todas as informações necessárias, desde encontrarem os serviços que pretendem, depois verem a sua disponibilidade no momento para saber se os sistemas estão *up* ou *down* com as indicações *Success* e *Error*, respetivamente. Então, no caso de estarem com a indicação de *Error*, os utilizadores têm a possibilidade de verem a estimativa de tempo na coluna *Time*, para a resolução da falha no serviço. E também, pretendemos até a data de entrega final do projeto, implementar a *feature* que faz com que a estimativa seja atualizada conforme o tempo passa até o mesmo esgotar e o serviço voltar a estar operacional.

Além disso, na coluna *Maintenance* os utilizadores têm a possibilidade de ver a manutenção com a data mais próxima dos serviços, uma *feature* implementada após a apresentação do projeto, e sendo possível agendar várias manutenções para o mesmo serviço, aplicando assim a sugestão do Prof. José Nuno Lau e facilitando o trabalho dos administradores e melhorando a comunicação com os futuros utilizadores do *Service Status UA*.

Além disso, na coluna *Maintenance* os utilizadores têm a possibilidade de ver as datas das manutenções agendadas de determinado serviço ou sistema e do lado direito dessa data, existe o botão *Schedule* que permite um utilizador ver o calendário das próximas manutenções já agendadas. Por último, os utilizadores também têm a possibilidade de ver o histórico no botão *History* e ver as datas específicas e os respetivos horários em que houve falhas em determinados serviços ou sistemas.

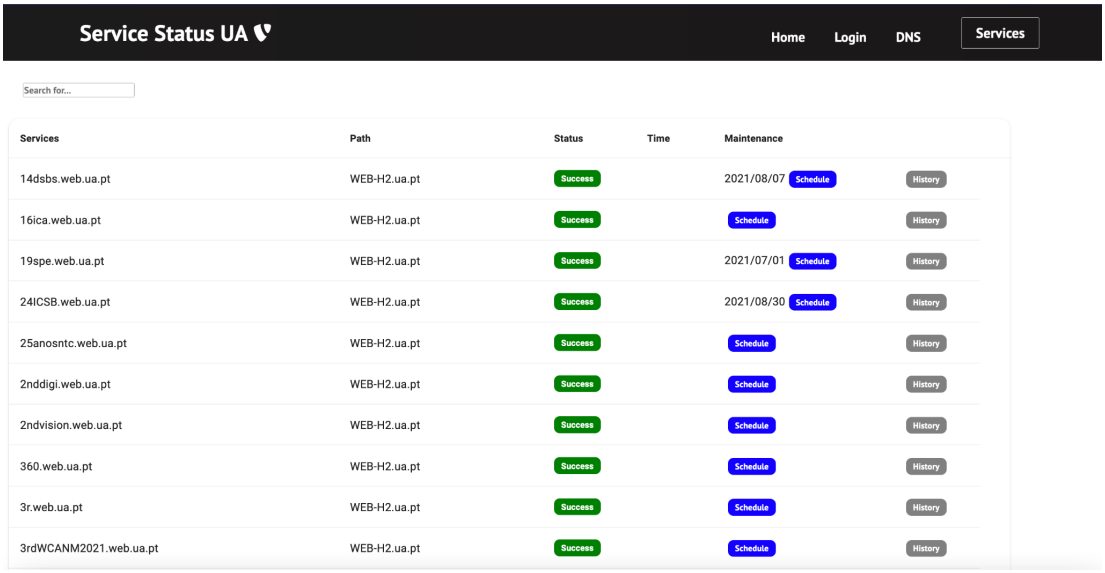


Fig. 9 - Services

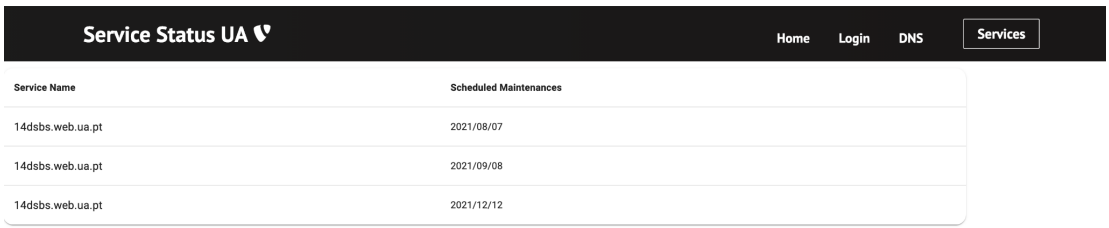


Fig. 10 - Schedule

A página a seguir, mostra o que o utilizador vê ao clicar no botão *History* para conseguir ver o histórico de falhas de um determinado serviço.

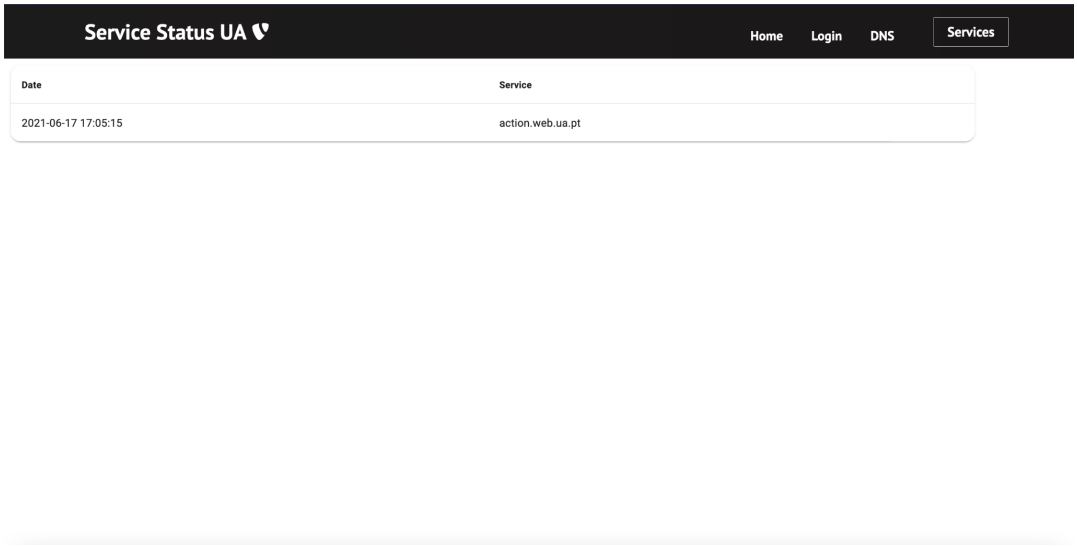


Fig. 11 - History

Em seguida esta mesma página possui no topo, uma barra de pesquisa que tem o propósito de ajudar o utilizador a encontrar o serviço da sua preferência de forma mais eficiente sem ser necessário passar pelos inúmeros serviços e sistemas existentes na aplicação web.

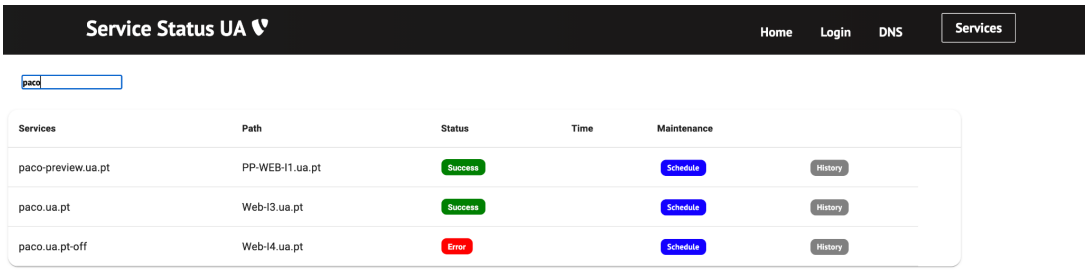


Fig. 12 - Filtro (Services)

4.2 Dns

O **DNS**, é um sistema que tem como objetivo associar nomes de domínios a endereços IP, utilizados para a identificação e localização de serviços e sistemas. Então, como um elemento extra para este projeto e para os futuros utilizadores que desejam ver este tipo de informação, para além da tabela dos serviços e sistemas apresentada anteriormente, adicionou-se uma tabela que corresponde ao **DNS** da Universidade de Aveiro cujos dados são providenciados pela API **SCOM**. Na tabela a seguir, consegue-se verificar quantos domínios existem e qual é o seu estado atual dentro da Universidade.

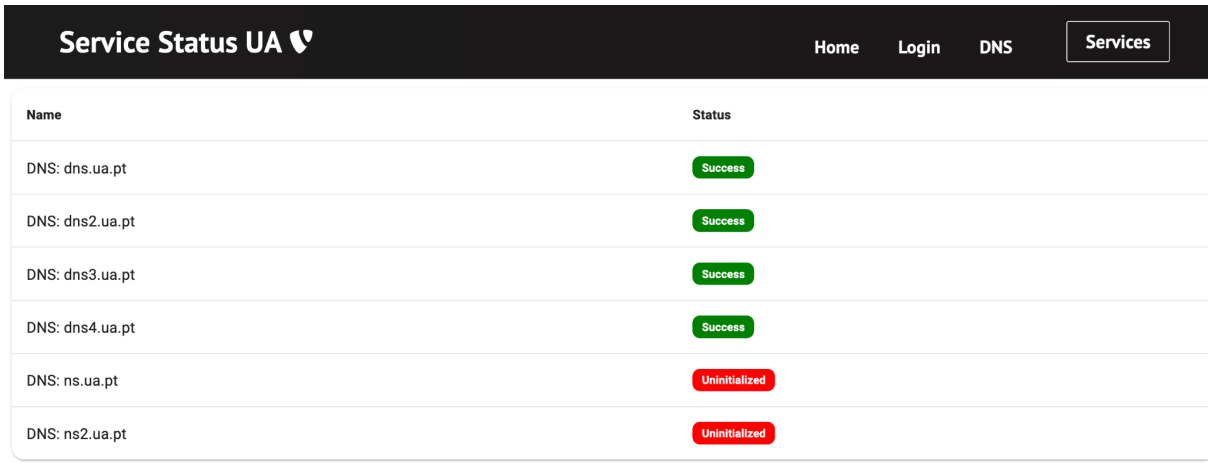


Fig. 13 - DNS

4.3 Admin

O **Administrador**, mais comumente chamado de **Admin** é um dos recursos adicionados a este projeto com o objetivo de indicar em que momento um serviço ou sistema terá uma manutenção agendada em um dia específico.

Além disso, também terá a função de indicar a estimativa de tempo que um problema irá levar para ser resolvido. Então, é importante realçar desde já, que o papel de **Admin** será executado por uma pessoa muito competente e experiente nesta área, de forma a saber que determinados tipos de falha, têm uma tendência a demorar um certo período de tempo para serem solucionados.

Antes de mais, devemos dizer que foi criada uma API com o propósito de guardar a informação escrita pelo **Admin** visto que não é possível executar queries para inserir dados em **SQL** a partir do **ReactJS**. Então, após o **Admin** escrever nas caixas de texto, é gerada essa informação na API e em seguida o *Daemon* vai buscar esta mesma informação através de um *endpoint* que contém o nome do serviço ou sistema, a data da manutenção agendada ou a estimativa de tempo para a resolução do problema e atualiza assim a BD de 1 em 1 minuto.

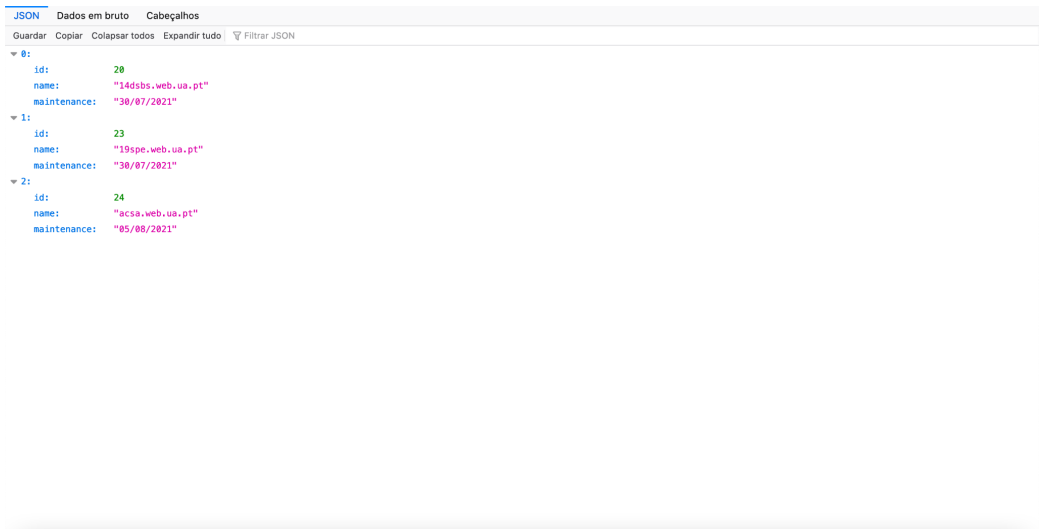


Fig. 14 - Maintenance Endpoint

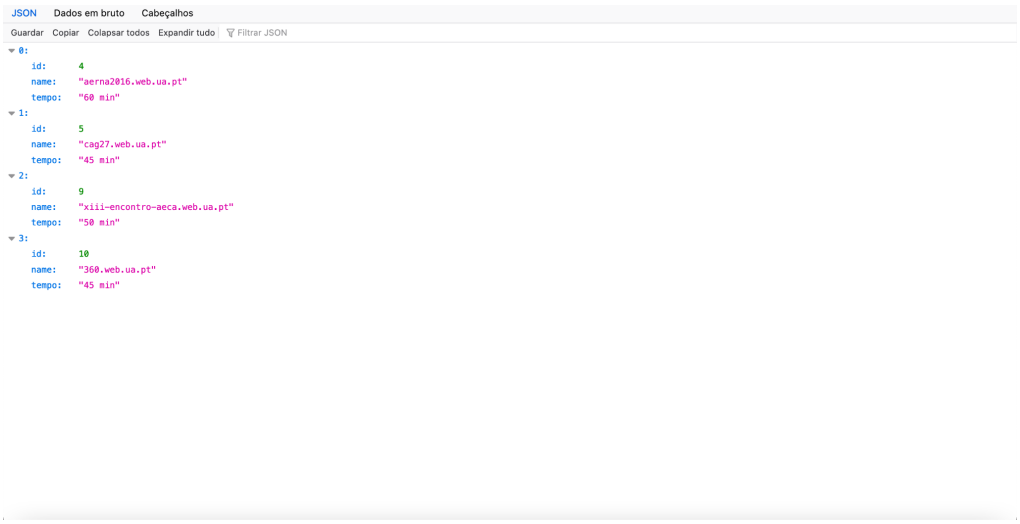


Fig. 15 - Time Endpoint

Fig. 16 - Documentação da API no Swagger

Deste modo, o **Admin** ao escrever o nome do serviço ou sistema e a indicar a sua manutenção agendada ou a estimativa de tempo para a resolução da falha irá informar os utilizadores e guardar os dados na BD criada. Então, o **Admin** deverá fazer o *login* na plataforma para conseguir aceder a uma interface diferente da que é vista pelos utilizadores e assim conseguir realizar as tarefas correspondentes à sua função.

Sendo assim, nas imagens a seguir, iremos demonstrar os passos e a interface vista pelo **Admin**, desde o *login*, passando pela página das manutenções agendadas até a página das estimativas de tempo, onde o **Admin** coloca o nome do serviço ou sistema e a sua data de manutenção ou estimativa de tempo para a resolução da falha.

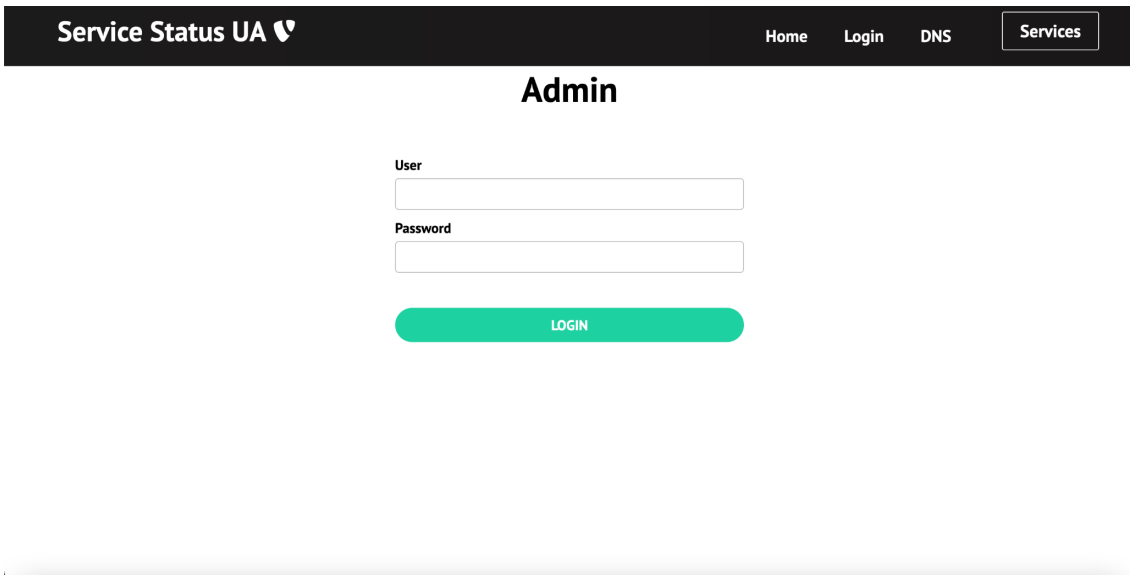


Fig. 17 - Admin Login

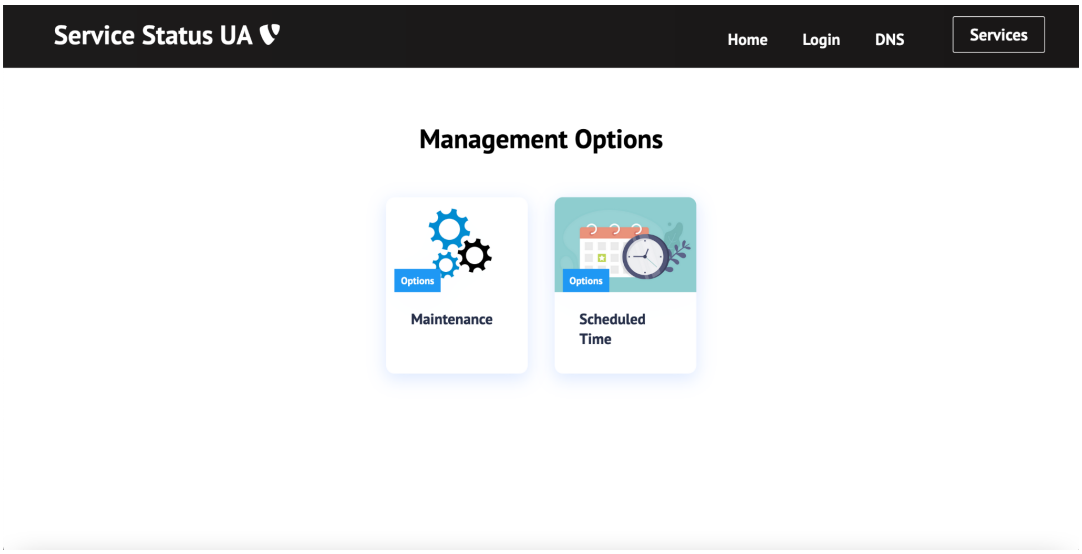


Fig. 18 - Management Options

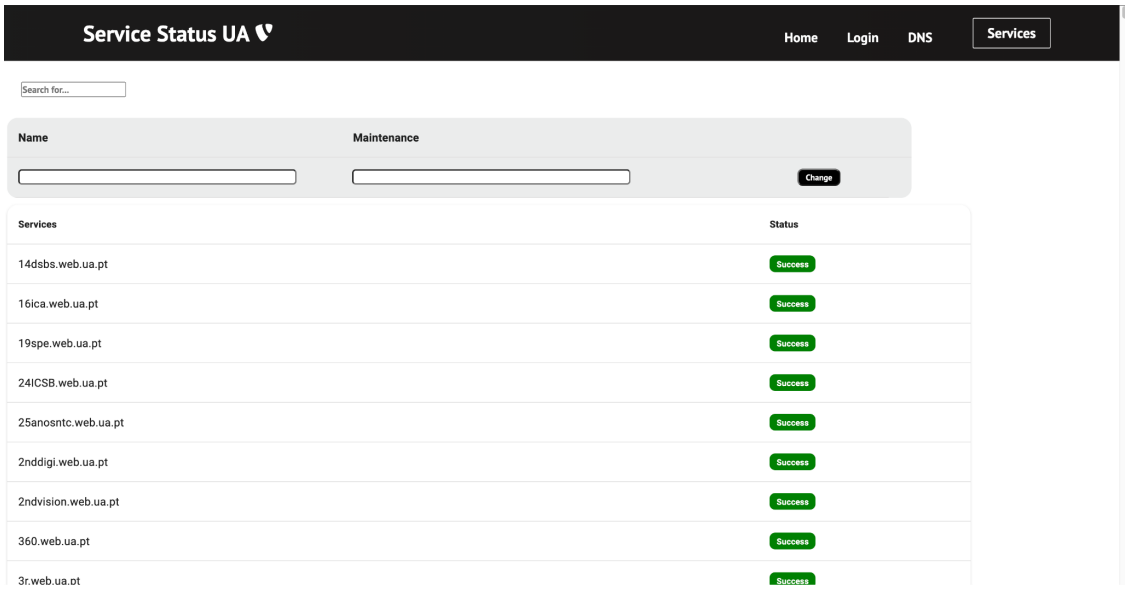


Fig. 19 - Maintenance

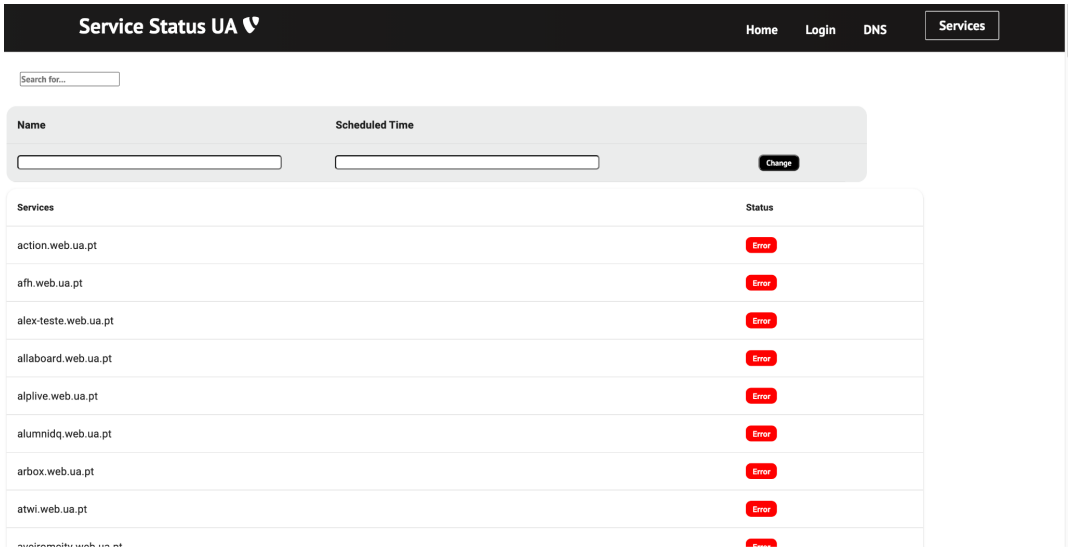


Fig. 20 - Scheduled Time

5. Conclusão

5.1 Sumário

Este relatório começa por explicar o contexto do projeto desenvolvido, mostra o seu propósito, os objetivos a alcançar e o problema que está a tentar resolver.

Em seguida, há uma seção que contém informações acerca da arquitetura do sistema e os seus requisitos. Sendo assim, esta mesma seção mostra como o sistema foi construído e como as diferentes partes do mesmo comunicam, trabalham juntas, oferecendo assim uma visão clara da estrutura do sistema.

No capítulo a seguir, faz-se uma análise de produtos similares, das funcionalidades que providenciam e o motivo pelo qual os mesmos não acabam por resolver o problema dos utilizadores e que o nosso projeto se propôs para fazê-lo.

A seguir, é feita uma abordagem da tecnologia utilizada para o desenvolvimento do projeto e o motivo pelo qual estas mesmas tecnologias foram escolhidas. Como ainda, aborda-se as ferramentas utilizadas pelos membros do grupo para a comunicação, partilha de documentos e a publicação da aplicação web. Além disso, no capítulo a seguir, explica-se de forma detalhada a implementação aplicada ao projeto, as funcionalidades que oferece ao utilizador e como as mesmas funcionam.

Por fim, deu-se a criação da aplicação web denominada *Service Status UA* com o propósito de resolver os problemas existentes na comunidade académica e que foram citados nos capítulos anteriores.

5.2 Resultados

Os objetivos traçados no começo da construção e desenvolvimento deste projeto pareciam muito simples e diretos, apesar da sua implementação não o ser. E, dado esses mesmos objetivos, o projeto fornece informações sobre o que é de facto o mais importante, que é a disponibilidade de cada serviço ou sistema, o seu histórico de falhas, a possibilidade de ter um **Admin** que forneça as estimativas de tempo para resolução de problemas caso haja uma falha recentemente, e além

disso, fornece informação sobre as manutenções agendadas de cada serviço ou sistema.

Por último, como um elemento extra, fornece também o estado dos sistemas de nomes de domínio, ou seja, o **DNS**.

5.3 Trabalho Futuro

Houve *features* que não foi possível serem implementadas e que são importantes para os utilizadores, e para serem implementadas no futuro para ajudar a evolução deste projeto. É importante realçar que, desde o princípio que tínhamos como objetivo alcançar todas as metas traçadas mas infelizmente não foi possível devido aos contratempos que tivemos. Então, essas *features* estão relacionadas de forma intrínseca e são as seguintes: a causa das falhas de cada problema nos serviços ou sistemas e consequentemente a explicação mais detalhada da falha dentro da *History* de cada serviço ou sistema para uma melhor compreensão e comunicação com os dos utilizadores da plataforma.

Além disso, também se pretende implementar notificações para permitir que os utilizadores possam subscrever e receber alertas acerca da disponibilidade dos serviços que fazem parte do seu interesse. E, desta forma continuaremos a melhorar o nosso projeto.

6. Referências

[1] Youtube, https://www.youtube.com/
[2] StackOverflow, https://stackoverflow.com/
[3] Microsoft, https://docs.microsoft.com/pt-pt/
[4] Microsoft Azure, https://docs.microsoft.com/pt-pt/
[5] MDN Web Docs, https://developer.mozilla.org/en-US/
[6] DZone, https://dzone.com/
[7] TutorialsPoint, https://www.tutorialspoint.com/index.htm
[8] StackExchange, https://stackexchange.com/
[9] Morioh, https://morioh.com/explore?next=%2F
[10] CodeSandBox, https://morioh.com/explore?next=%2F
[11] SQLHack, https://www.sqlshack.com/
[12] DigitalOcean, https://www.digitalocean.com/community