isec
Engenharia

MESTRADO EM INFORMÁTICA E SISTEMAS

PROVISÓRIO

C3 Cloud Control

Autor

Nuno Filipe Faneca Bento

Orientador

José Marinho

INSTITUTO POLITÉCNICO
DE COIMBRA

INSTITUTO SUPERIOR
DE ENGENHARIA
DE COIMBRA

Coimbra, julho de 2020

isec
Engenharia

DEPARTAMENTO DE INFORMÁTICA E SISTEMAS

C3 Cloud Control

Relatório de Estágio de Natureza Profissional para a obtenção do grau de Mestre em Informática e Sistemas

Especialização em Desenvolvimento de Software

Autor

Nuno Filipe Faneca Bento

Orientador

José Marinho

Supervisor na empresa Critical Links

Hélder Pereira

Coimbra, julho de 2020

INSTITUTO POLITÉCNICO
DE COIMBRA

INSTITUTO SUPERIOR
DE ENGENHARIA
DE COIMBRA

"We are what we repeatedly do.

Excellence, then, is not an act, but a habit" – Will Durant

AGRADECIMENTOS

Agradeço ao Professor José Marinho por todo o apoio e dedicação que me deu enquanto meu orientador, apesar de eu não ter cooperado afincadamente. Agradeço ainda a todos os docentes do ISEC que me ajudaram na minha formação enquanto aluno de Licenciatura em Engenharia Informática e do Mestrado Informática e Sistemas.

À minha equipa de trabalho na Critical Links e colegas da Critical Software pelo fantástico acolhimento, por toda a ajuda e conhecimentos que me deram. Ao Hélder Pereira e ao Mário Monteiro pelo conhecimento transmitido.

À minha namorada Ana Ferreira, aos meus pais e sogros, por me apoiarem na decisão de continuar a estudar e pelos valores que me transmitem. Estou muito grato obrigado pelo apoio incansável, paciência e compreensão em todos os momentos, sem vocês não teria conseguido.

RESUMO

O estágio descrito neste documento foi realizado na empresa Critical Links, no âmbito da unidade curricular de Projeto/Estágio do Mestrado em Informática e Sistemas - Ramo de Desenvolvimento de Aplicações, do Instituto Superior de Engenharia de Coimbra.

No âmbito de melhorar parte dos serviços de software da Critical Links, o principal foco deste projeto foi o serviço C3 Cloud, que permite sincronizar conteúdos educativos para servidores localizados em escolas ou outros locais de ensino. O C3 Cloud está disponível desde 2012 e sincroniza conteúdos a mais de 4000 servidores em vários países. No entanto, este serviço não foi planeado para suportar um número tão elevado de servidores, apresentando sérias limitações ao nível da escalabilidade e disponibilidade. Neste sentido, um dos objetivos principais deste estágio consistiu em substituir o componente C3 Cloud de uma forma faseada e suave, de forma a minimizar o impacto da sua completa substituição. Outro objetivo importante foi a alteração da interface web, que ambiciona transmitir uma boa experiência de utilização, resolvendo alguns dos problemas existentes do serviço do C3 Cloud na sua 1ª versão (C3 Cloud v1).

Foi realizado um estudo sobre topologias de rede e ferramentas de sincronização, com o objetivo de otimizar a sincronização de conteúdos pelos vários servidores. Foi escolhida uma topologia mista entre centralizada e distribuída, de forma a obter um melhor controlo da rede e a vantagem decorrente da sincronização de conteúdos entre vários servidores em forma ponto-a-ponto. Foi escolhida a ferramenta de sincronização Syncthing e composta uma análise detalhada sobre a mesma, de forma a evidenciar a sua viabilidade para a solução. Este estudo foi efetuado com o objetivo de apoiar as decisões do desenvolvimento do C3 Cloud v2.0.

Os objetivos propostos foram seguidos até ao fim com sucesso, nomeadamente uma nova versão do C3 Cloud e a substituição gradual e faseada da anterior. O serviço apresenta uma interface web moderna e de fácil utilização. O C3 Cloud v2 encontra-se em utilização com dispositivos C3 localizados em várias escolas pelo mundo.

Palavras-chave: C3 Cloud, Educação, Ponto-a-Ponto, Sincronização, Rede Centralizada, Rede distribuída, Syncthing.

ABSTRACT

The internship described in this document was carried out at the company Critical Links, within the scope of the curricular unit of Project / Internship of the Master in Informatics and Systems - Application Development Branch, of the Instituto Superior de Engenharia de Coimbra.

To improve part of Critical Links' software services, the main focus of this project was the C3 Cloud service, which allows to synchronize the educational content of servers located in schools or other teaching locations. The C3 Cloud has been available since 2012 and synchronizes the content of more than 4000 servers in several countries. However, this service was not planned to support such a large number of servers, with serious limitations in terms of scalability and availability. In this sense, one of the main objectives of this internship was to replace the C3 Cloud component in a phased and smooth manner, in order to minimize the impact of its complete replacement. Another important objective was to replace the web interface for providing a better user experience, solving some of the problems of the C3 Cloud service in its first version (C3 Cloud v1).

A study was carried out on network topologies and synchronization tools to optimize the synchronization of educational content over the various servers. A mixed topology was chosen (i.e., centralized and distributed) to better control the network and increase performance through a point-to-point synchronization approach. The Syncthing synchronization tool was chosen and analyzed in detail to demonstrate its viability for the solution. This study was carried out with the objective of supporting the C3 Cloud v2.0 development decisions.

The proposed objectives were successfully achieved, namely a new version of the C3 Cloud and the gradual and phased replacement of the previous one. This service features a modern and user-friendly web interface. Currently, C3 Cloud v2 is in use with C3 devices located in schools around the world.

Keywords: C3 Cloud, Centralized Network, Distributed Network, Education, Peer-to-Peer, Synchronization, Syncthing.

ÍNDICE

Agradecimentos i

Resumo iii

Abstract v

Índice vii

Índice de figuras xi

Índice de tabelas xiii

Acrónimos xv

1 Introdução 1

1.1 Critical Links 1

1.2 Instituto Superior de Engenharia de Coimbra 2

1.3 Enquadramento 3

1.4 Objetivos e plano de trabalhos 4

1.5 Estrutura do relatório 5

2 O Projeto - C3 Cloud Control 7

2.1 C3 Cloud Control v1.0 7

2.1.1 Visão geral 7

2.1.2 Funcionamento 8

2.1.3 Limitações e problemas 11

2.2 C3 Cloud Control v1.5 12

2.2.1 Arquitetura 12

2.2.2 Aspetos relevantes da implementação 13

2.2.3 Problemas do CCC v1.0 corrigidos 14

2.3 C3 Cloud Control 2.0 17

2.4 Análise Comparativa do C3 Cloud Control 18

3 Otimização da Sincronização de Conteúdos 21
3.1 Topologias de Redes 21
3.2 Ferramentas de Sincronização 22
3.3 Análise da ferramenta Syncthing 24
3.4 Centralizar o Syncthing 26
3.5 Integração com o C3 Cloud 27
3.6 Funcionamento 28
4 Desenvolvimento do C3 Cloud 2.0 29
4.1 Planeamento 29
4.2 Requisitos 29
4.2.1 Requisitos funcionais 29
4.2.2 Requisitos não funcionais 31
4.3 Arquitetura do Sistema 32
4.3.1 Desenho e Especificação da Arquitetura Geral 32
4.3.2 Desenho e Especificação da Arquitetura Interna 33
4.4 Ferramentas e Tecnologias Utilizadas 34
4.4.1 Frontend 34
4.4.2 Backend 36
4.4.3 Controlo de Versões 37
4.5 Integração do C3 Cloud com o C3 38
4.6 Problemas encontrados no C3 Cloud v2.0 38
5 Estudo Comparativo da Sincronização entre as Versões 1.0 e 2.0 do C3 Cloud Control 41
5.1 Cenários de Sincronização 41
5.2 Resultados 42
5.3 Discussão dos resultados 47
6 Resultados da Implementação 51

6.1 Autenticação 51
6.2 Dashboard 51
6.3 Conteúdos 52
6.4 Dispositivos 57
6.5 Grupos de dispositivos 63
6.6 Relatórios 66
6.7 Definições 67
7 Conclusões e trabalho futuro 69
7.1 Conclusões 69
7.2 Trabalho Futuro 70
7.3 Considerações finais 70
Referências 73
Anexo A – Proposta de Estágio A-1
Anexo B – Análise do C3 Cloud v1.0 ..... B-1
Anexo C - Documentação do C3 Cloud v2.0 ..... C-1

ÍNDICE DE FIGURAS

Figura 1-1 – Dispositivos C3 para instalação em escolas 1
Figura 1-2 - Visão geral das soluções da Critical Links, os dispositivos C3 e o C3 Cloud 2
Figura 1-3 – Plano faseado da evolução do serviço C3 Cloud Control 4
Figura 2-1 - Exemplo da página de conteúdos do C3 Cloud v1 9
Figura 2-2 - Exemplo da página dos dispositivos do C3 Cloud v1 10
Figura 2-3- Exemplo do ecrã de utilizadores da interface web do C3 Cloud v1 11
Figura 2-4- Arquitetura do C3 Cloud na versão 1.5 13
Figura 2-5 - Limitação de edição de múltiplos dispositivos na versão 1.0 do C3 Cloud 14
Figura 2-6 - Solução ao problema de edição de vários dispositivos na versão 1.5 do C3 Cloud 15
Figura 2-7 - Limitação de upload de conteúdos para a cloud na versão 1.0 do C3 Cloud 16
Figura 2-8 - Solução para melhor usabilidade no upload de conteúdos para a cloud na versão 1.5 do C3 Cloud 16
Figura 2-9 - Solução para procurar facilmente dispositivos na versão 1.5 do C3 Cloud .. 17
Figura 2-10- Exemplo da topologia de rede do C3 Cloud v2 com os C3 18
Figura 3-1 - Principais topologias de rede [8] 21
Figura 3-2 - Exemplo de topologia de rede centralizada e distribuída [8] 22
Figura 3-3 - Número de dispositivos com o syncthing instalado, ao longo do tempo e por cada versão, a nível mundial [18] 26
Figura 3-4 - Exemplo de funcionamento da característica Introdutora 27
Figura 3-5 - Screenshot de exemplo da interface gráfica da ferramenta Syncthing 28
Figura 4-1 – Topologia de rede com respetivas conexões dos dispositivos C3 e C3 Cloud 32
Figura 4-2 - Estrutura interna do C3 Cloud Control (servidor central) e de um dispositivo C3 33
Figura 4-3 Diagrama de fluxo do arranque inicial de um C3 38
Figura 4-4 - Arquitetura do C3 Cloud versão 3.0 (simplificada) 40
Figura 5-1 - Servidores de teste para ambos os cenários 42
Figura 5-2 - Interface gráfica da ferramenta ntopng, com o tráfego enviado/recebido por protocolo e aplicação 43

Figura 5-3 – Comparação da sincronização do conteúdo 1 por ordem sequencial nos cenários 1 e 2 46
Figura 5-4 - Comparação da sincronização do conteúdo 2 por ordem sequencial nos cenários 1 e 2 47
Figura 5-5 - Screenshot da interface do Syncthing, no Servidor 4 48
Figura 5-6 - Rede local de servidores C3 a sincronizar um conteúdo 49
Figura 5-7 - Exemplo de duas redes locais com servidores C3 online e offline a sincronizar entre si 49
Figura 6-1 – Página de login do C3 Cloud 51
Figura 6-2 - Página de entrada do C3 Cloud 52
Figura 6-3 - Criação de um novo conteúdo 53
Figura 6-4 – Listagem de conteúdos 53
Figura 6-5 - Filtros de conteúdos 54
Figura 6-6 - Detalhes de um conteúdo 54
Figura 6-7 - Gestor de ficheiros 55
Figura 6-8 – Formas de fazer uploads para um conteúdo 55
Figura 6-9 - Dispositivos subscritos a um conteúdo 56
Figura 6-10 - Subscrever múltiplos dispositivos a um conteúdo 56
Figura 6-11 - Registar um dispositivo 57
Figura 6-12 – Registo de vários dispositivos com um ficheiro CSV 58
Figura 6-13 - Lista de todos os dispositivos 59
Figura 6-14 - Filtragem de dispositivos por versão 59
Figura 6-15 - Detalhes de um dispositivo 60
Figura 6-16 - Conteúdos e estado de sincronização de um dispositivo 61
Figura 6-17 - Análise de dados sobre o uso de conteúdos 62
Figura 6-18 – Abertura de um túnel SSH para controlo remoto de um dispositivo 62
Figura 6-19 - Controlo remoto avançado de um dispositivo 63
Figura 6-20 – Organização de dispositivos por grupos 63
Figura 6-21 - Criação de um novo grupo 64
Figura 6-22 - Grupo de dispositivos 65
Figura 6-23 - Conteúdos de um grupo de dispositivos 65
Figura 6-24 - Cancelar subscrição de conteúdos num grupo de dispositivos 66
Figura 6-25 - Subscrever conteúdos para os dispositivos de um grupo 66
Figura 6-26 - Sincronização dos dispositivos 67
Figura 6-27 - Definições e visualização das instâncias Syncthing 68
Figura 6-28 - Configurações do Syncthing para todos os dispositivos conectados ao C3 Cloud 68

ÍNDICE DE TABELAS

Tabela 2-1 - Resumo das principais limitações da solução, respetivas consequencias e ação tomada 11
Tabela 2-2 - Resumo das principais limitações da solução, respetivas consequencias e ação tomada (continuação) 12
Tabela 2-3 - Comparação de características chave das três versões do C3 Cloud 19
Tabela 3-1 - Comparação de ferramentas de sincronização 24
Tabela 3-2 - Principais características da ferramenta Syncthing 25
Tabela 4-1 - Permissões por perfil de utilizador 31
Tabela 4-2 - Arquitetura interna de software no C3 Cloud Control 34
Tabela 4-3 - Módulos mais relevantes no desenvolvimento do frontend 35
Tabela 4-4 - Módulos mais relevantes no desenvolvimento do backend 36
Tabela 4-5 – Serviços e ferramentas utilizadas para controlo de versões de código 37
Tabela 5-1 - Resultados da sincronização de "tudo ao mesmo tempo" 43
Tabela 5-2 - Resultados da sincronização com ordem sequencial no cenário 1 (C3 Cloud v1.0) 44
Tabela 5-3 - Resultados da sincronização com ordem sequencial no cenário 2 45

ACRÓNIMOS

API – Application Programming Interface

CCC – C3 Cloud Control

IPC – Instituto Politécnico de Coimbra

ISEC – Instituto Superior de Engenharia de Coimbra

HTML – HyperText Markup Language.

HTTP – Hyper Text Transfer Protocol

HTTPS – Hyper Text Transfer Protocol Secure

JSON – JavaScript Object Notation

ISO – ficheiro com extenção .iso

LAN - Local Area Network

MVC – Model View Controller

P2P – Peer-to-Peer

PDF – Portable Document Format

TAR – ficheiro com extenção .tar

TLS – Transport Layer Security

UI – User Interface

URL – Uniform Resource Locator

UX – User Experience

XML - Extensible Markup Language

ZIM – ficheiro com extenção .zim

ZIP – ficheiro com extenção .zip

1 INTRODUÇÃO

Este documento descreve o estágio desenvolvido pelo aluno Nuno Filipe Faneca Bento na entidade de acolhimento Critical Links, uma empresa sediada em Coimbra, no âmbito da unidade curricular de Projeto/Estágio do Mestrado de Informática e Sistemas (MIS) do Instituto Superior de Engenharia de Coimbra (ISEC).

1.1 Critical Links

A empresa de acolhimento do estagiário foi a Critical Links [1], que começou em 2006 como uma divisão do Grupo Critical Software [2]. Esta entrega soluções tecnológicas na área da educação a nível mundial, nomeadamente em países da América Central, América do Sul, Europa, Ásia e África, sendo o foco principal a educação em países em desenvolvimento. A empresa tem como visão aproximar os estudantes ao conhecimento, especialmente em condições desafiantes por limitações de infraestruturas e de conectividade à internet, reduzindo deste modo a divisão digital [3] na educação.

Uma das soluções da empresa é o C3, um tipo de dispositivo representado na Figura 1-1, que permite o acesso a conteúdos educativos em qualquer escola, independentemente da área geográfica onde se insere, mesmo com conectividade de Internet limitada. Cada C3 consiste num conjunto de *hardware* e *software* com funcionalidades apropriadas para escolas. Este serve de ponto de acesso por Wi-Fi para que os alunos ou professores se possam ligar para aceder às suas funcionalidades, nomeadamente a gestão e acesso dos alunos à internet por Wi-Fi, acesso aos conteúdos (*offline*) armazenados em disco. É possível administrar o acesso à Internet por papel de utilizador (alunos, professores ou outros), atribuindo permissões desejadas de acesso. Tem funcionalidade *plug-and-play* para facilitar a instalação, mesmo por clientes com pouco conhecimento técnico.

Figura 1-1 – Dispositivos C3 para instalação em escolas

Para que seja disponibilizado um vasto repositório de conteúdos educativos, existe um serviço complementar responsável pela criação e sincronização de conteúdos pelos diversos C3. Este serviço é o C3 Cloud Control (CCC), ou simplesmente C3 Cloud (CC). A sua arquitetura é centralizada e apresenta uma visão global dos C3 conectados presentes nas escolas, possibilitando que seja feito um controlo mais apurado dos conteúdos que são colocados nesses dispositivos. A Figura 1-2 mostra a visão geral da solução da empresa com os dois serviços principais: os C3 e o C3 Cloud.

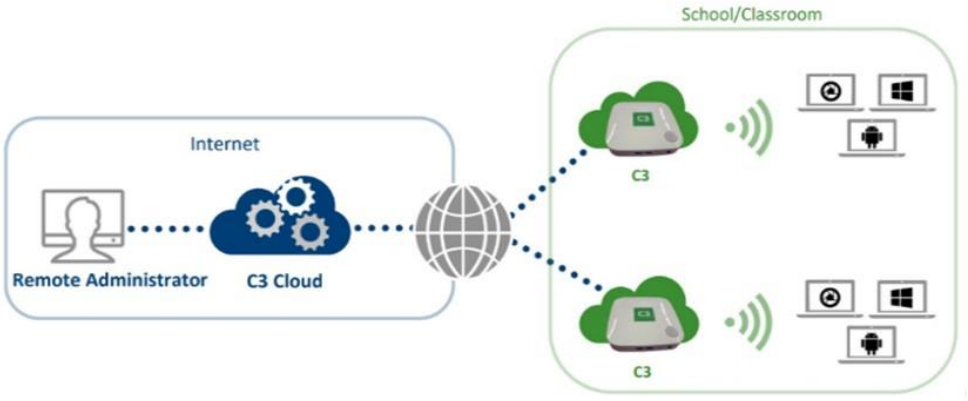


Figura 1-2 - Visão geral das soluções da Critical Links, os dispositivos C3 e o C3 Cloud

1.2 Instituto Superior de Engenharia de Coimbra

O ISEC (Instituto Superior de Engenharia de Coimbra) [4] é uma instituição fundada em 1974 e que se integrou no Instituto Politécnico de Coimbra (IPC) em 1988. São lecionados cursos de especialização tecnológica, licenciaturas, mestrados e pós-graduações em diferentes áreas de engenharia. A sua missão consiste na criação, transmissão e difusão de cultura, ciência e tecnologia, cabendo-lhe ministrar uma formação de nível superior para o exercício de atividades profissionais no domínio da engenharia e promover o desenvolvimento da região em que se insere.

Na área de informática e sistemas, o ISEC apresenta um leque formativo para graduar estudantes com licenciaturas, mestrados e CTeSP. O Mestrado em Informática e Sistemas (MIS) tem por objetivo formar mestres em Informática e em Sistemas capazes de exercerem a sua atividade profissional com um elevado nível de competência técnica, científica e profissional em cada uma das áreas de especialização propostas.

Este organiza-se em duas especializações: “Desenvolvimento de Software” e “Tecnologias da Informação e do Conhecimento”. A formação dada na especialização em Desenvolvimento de Software visa a preparação de profissionais competentes no exercício profissional no domínio

de desenvolvimento de software em todos as suas etapas, incluindo os aspetos importantes de gestão de projetos, equipas e garantia de qualidade. Os graduados do mestrado nesta especialização serão técnicos com pleno domínio científico dos temas relacionados com engenharia de software, possuindo capacidade de resposta a exigências constantes de inovação e modernização.

1.3 Enquadramento

O projeto descrito neste documento inseriu-se num contexto real de trabalho e, no final, resultou numa solução que está colocada num ambiente de produção. Atualmente, o serviço C3 Cloud Control serve mais de 4000 clientes C3 em vários países. No entanto, na sua génese, este serviço não foi planeado para suportar este número elevado de clientes, apresentando sérias limitações ao nível da escalabilidade e disponibilidade. Além disso, não expõe qualquer tipo de API (*Application Programming Interface*) standard e bem documentada que permita a interação com sistemas de terceiros.

Neste sentido, este estágio teve como objetivo substituir o componente C3 Cloud Control que já se encontrava em produção à data de arranque do estágio, de uma forma faseada e suave e, deste modo, minimizar o impacto da sua completa substituição nos clientes atuais. Este inseriu-se num processo que a Critical Links tem vindo a levar a cabo para modernizar e estabilizar o seu portfólio de produtos e serviços e permitiu ao estagiário um contacto direto com os clientes finais.

O código do projeto foi maioritariamente implementado pelo estagiário. No entanto a equipa de desenvolvimento da Critical Links também participou em algumas ocasiões, tendo sido composta por:

- André Marques – programador de *frontend*;
- Mário Monteiro – programador de *frontend*, *backend* e *DevOps*[1]*;*
- Helder Pereira – gestor do projeto;
- Nuno Bento – programador de *frontend, backend* e *DevOps.*

Foi utilizado uma metodologia ágil, o SCRUM, uma vez que era uma equipa pequena e que tinha contacto direto com os clientes. Esta reunia-se diariamente para resolver os obstáculos que eram encontrados, esclarecer o que tinha sido feito no dia anterior e o que era para fazer no próprio dia. O tempo era dividido por *sprints* de duas semanas, servindo estas para planear o que seria para fazer nas duas semanas seguintes. Era ainda feita uma retrospetiva do que tinha acontecido, incluindo o que tinha corrido bem ou mal.

[1] *DevOps* – Diversas ações no sistema operativo para manipular o *software* desenvolvido

1.4 Objetivos e plano de trabalhos

O principal objetivo a alcançar com o projeto proposto pela Critical Links consistiu na modernização do C3 Cloud, tornando-o escalável para poder suportar até 10 000 clientes. Este também teve que se tornar adaptável para possibilitar a integração de novos componentes e/ou tecnologias e, o mais importante, reduzir os requisitos de largura de banda em relação à solução atual. Pretendeu-se, igualmente, com a nova versão o desenvolvimento de uma interface web que transmitisse uma boa experiência de utilização, resolvendo os problemas existentes no serviço. Conforme já foi referido, o foco principal neste relatório foi o componente C3 Cloud e as suas interações com os dispositivos C3.

Na proposta de estágio (Anexo A), estavam definidas 3 fases distintas que pretendiam atingir os seguintes objetivos genéricos, o que foi cumprido com sucesso:

- **Fase 1**: desenvolvimento de uma API REST standard para comunicar com o C3 Cloud Control existente;
- **Fase 2**: desenvolvimento de uma interface web que recorresse à API REST criada na fase anterior;
- **Fase 3**: desenvolvimento e substituição do *backend* do C3 Cloud Control.

A Figura 1-3 representa visualmente a evolução do C3 Cloud Control (versões 1.0, 1.5 e 2.0) ao longo do estágio, em que à esquerda está a versão inicial do serviço (simplificada), com um *backend* e uma UI (*User Interface*). As fases 1 e 2 dos objetivos do estágio encontram-se no centro da figura, que apresenta uma nova interface gráfica utilizando um *adapter/broker*. Para finalizar o projeto, foi construído um novo *backend* escalável e adaptável, como previsto na fase 3.

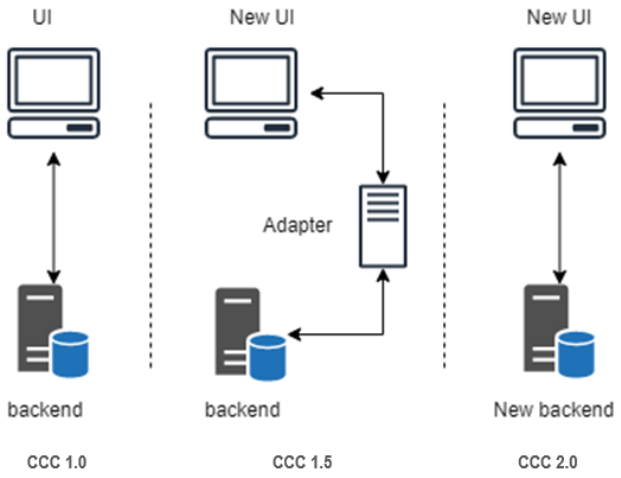


Figura 1-3 – Plano faseado da evolução do serviço C3 Cloud Control

O plano inicial foi seguido até ao fim com sucesso e o resultado do estágio (CCC 2.0) encontra-se em utilização com dispositivos C3 localizados em várias escolas pelo mundo. Quando o plano inicial terminou, foi necessário aplicar correções e otimizações à solução final, o que resultou na criação do C3 Cloud 3.0 (referido na seção 4.6).

1.5 Estrutura do relatório

Além deste primeiro capítulo de introdução e contextualização, o resto do relatório está estruturado da seguinte forma:

- **Capítulo 2: O Projeto C3 Cloud Control** – expõe os problemas da primeira versão da solução do C3 Cloud e tem como objetivo evidenciar a necessidade de evolução. Este capítulo descreve ainda as novas versões do C3 Cloud Control desenvolvidas pelo estagiário, explicando para cada uma como foram solucionados os problemas e limitações encontrados na primeira versão.
- **Capítulo 3: Otimização da Sincronização de Conteúdos** – apresenta o resultado de um estudo sobre ferramentas de sincronização e as topologias de rede mais conhecidas. Este foi efetuado com o objetivo de apoiar as decisões do desenvolvimento do *backend* do C3 Cloud v2.0. Neste capítulo, também é apresentada qual foi a ferramenta de sincronização escolhida e ainda uma análise detalhada sobre a mesma, de forma a evidenciar a sua viabilidade para a solução.
- **Capítulo 4: Desenvolvimento do C3 Cloud Control 2.0** – inclui os principais aspetos de desenvolvimento da versão mais recente do C3 Cloud Control, com descrição das principais escolhas, arquitetura e ferramentas utilizadas.
- **Capítulo 5: Testes Comparativos da Sincronização** – descreve a realização de cenários para testar a nova rede de sincronização, utilizando as versões 1.0 e 2.0 do C3 Cloud Control e, deste modo, tornar possível uma comparação destas ao nível da sincronização de conteúdos. Também são apresentados e analisados os resultados obtidos.
- **Capítulo 6: Conclusões e Trabalho Futuro –** conclui o presente documento com uma apreciação geral do trabalho realizado e, ainda, uma discussão sobre trabalhos futuros.

2 O PROJETO - C3 CLOUD CONTROL

Este capítulo encontra-se organizado de acordo com as três versões do C3 Cloud Control desenvolvidas até ao momento. A primeira seção refere-se a primeira versão, com uma contextualização de como funciona, para que serve, a sua interação com os servidores C3, bem como os seus problemas e limitações. Nas duas secções seguintes, estão descritas as novas versões do C3 Cloud que foram estudadas e desenvolvidas maioritariamente pelo estagiário, com auxílio da equipa da Critical Links, com a finalidade de criar um sistema moderno, capaz de escalar com o aumento do número de dispositivos C3.

2.1 C3 Cloud Control v1.0

Esta seção engloba uma visão geral do Serviço C3 Cloud Control na versão 1.0, explicando como funciona e de que forma interage com os dispositivos C3. São apresentadas as principais páginas do serviço com imagens da interface web. Por fim, são apresentadas as maiores limitações desta versão.

No início do estágio, foi realizado um estudo da versão 1.0 do C3 Cloud Control em que o principal objetivo consistiu em recolher a informação necessária para desenvolver o mesmo serviço a partir de um novo projeto de software, com uma arquitetura escalável com o aumento de clientes. Este estudo incidiu sobre os *endpoints* da API REST utilizada e sobre os perfis de utilizador suportados para efeitos de acessos e permissões.

2.1.1 Visão geral

A primeira versão do C3 Cloud foi lançada em janeiro de 2012 [5] e a sua função principal consiste em sincronizar conteúdos para os C3. As principais funcionalidades de um dispositivo C3 são apresentadas de seguida de forma concisa:

- **Wi-Fi** – tem um ponto de acesso que permite conexões por professores e alunos para conceder acesso a determinadas funcionalidades como, por exemplo, acesso à Internet e conteúdos;
- **Repositório** – funciona como um repositório de conteúdos educativos;
- **Cache** – é possível guardar páginas HTTP e HTTPS em cache, aumentando a velocidade e diminuindo o consumo de alguma largura de banda;
- **Acesso à Internet** – é possível gerir o acesso à Internet, seja por utilizadores ou por papel de utilizador (professor e aluno);
- **LMS e Moodle** – o C3 tem um uma aplicação web embutida de gestão de aulas;
- **Compatibilidade** – é compatível com *desktops*, portáteis, *tablets* e *smartphones* dos alunos ou dos equipamentos da escola.

Quando um dispositivo C3 é enviado para uma escola, este tem apenas alguns conteúdos que vêm por omissão. No entanto, o aspeto chave da solução é conter vários conteúdos educativos para a escola, tanto para alunos como professores. É aqui que entra o C3 Cloud Control, que tem como função principal gerir os conteúdos dos vários C3.

O C3 Cloud Control funciona como o servidor central da topologia de toda a rede C3, com uma interface gráfica que permite gerir os dispositivos C3, conteúdos e utilizadores. Este permite também abranger uma visão global de todo o ecossistema C3, potenciando um controlo mais apurado do conteúdo que é colocado nos dispositivos que estão nas escolas.

2.1.2 Funcionamento

A lógica de utilização do C3 Cloud divide-se em três categorias principais que são os utilizadores, os dispositivos C3 e os conteúdos.

Em relação aos conteúdos, todos têm um nome, descrição e palavras-chave para facilitar a pesquisa. Cada um também tem um tipo que permite que os C3 os possam interpretar e instalar de forma correta. Por exemplo, certos conteúdos têm de ser instalados em diferentes diretorias no disco do C3. Os vários tipos de conteúdo são:

- **C3** – pode incluir qualquer tipo de ficheiro, tais como PDFs, imagens ou vídeos;
- **Moodle** – semelhante ao conteúdo C3, mas aparecem num sítio diferente, no moodle integrado;
- **Kiwix** – conteúdos do tipo Wikimedia em formato ZIM (Wikimedia engloba as Wikipédias, Wikilivros e Wikcionário, entre outros);
- **Hidden** – são conteúdos do tipo C3, mas num estado escondido da interface gráfica do C3.

A Figura 2-1 mostra o ecrã de conteúdos, com uma lista do lado esquerdo, detalhes do conteúdo selecionado com ficheiros e pastas no centro e, por fim, várias informações individuais relativas ao ficheiro selecionado do lado direito.

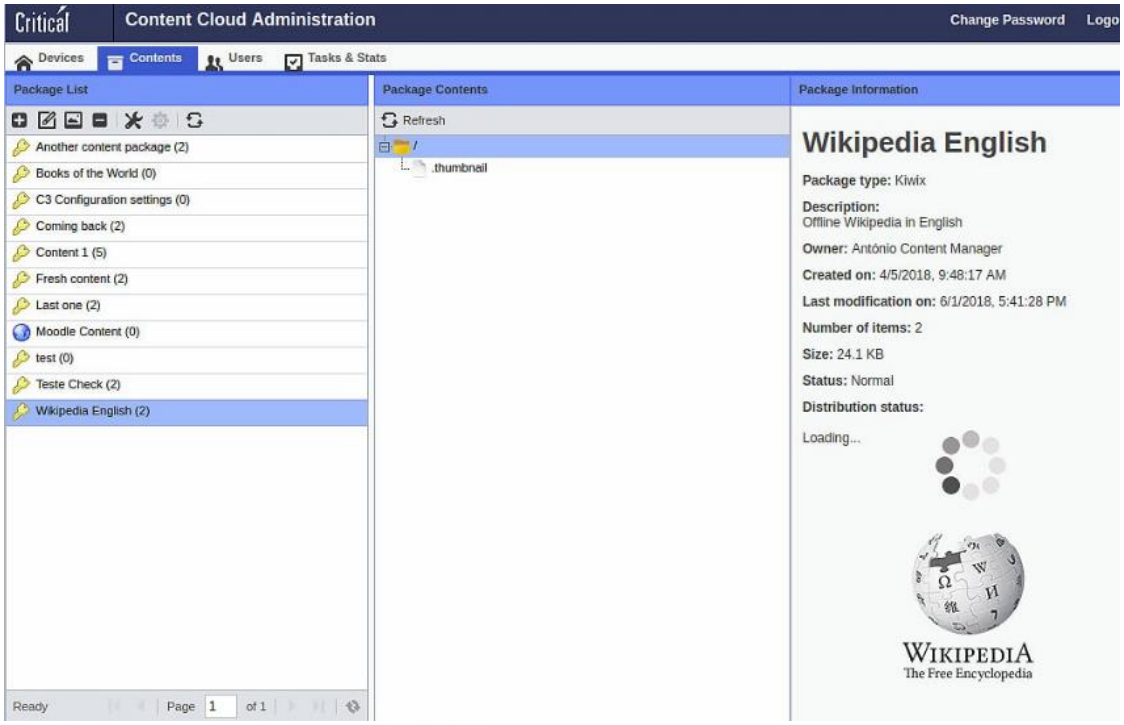


Figura 2-1 - Exemplo da página de conteúdos do C3 Cloud v1

Acerca dos dispositivos, o C3 Cloud permite gerir vários dos seus aspetos, tais como definições de conexão, quais os utilizadores que têm acesso ao dispositivo e quais os conteúdos que o servidor C3 deve ter. Os dispositivos conectados ao C3 Cloud aparecem como online e podem ser sincronizados com o conteúdo selecionado na interface.

A Figura 2-2 é um exemplo do ecrã de dispositivos, onde é possível ver os grupos de dispositivos do lado esquerdo, estando, por omissão, selecionado o grupo *All*. No centro estão todos os dispositivos que o grupo selecionado contém e o lado direito mostra informações apenas sobre o dispositivo selecionado.

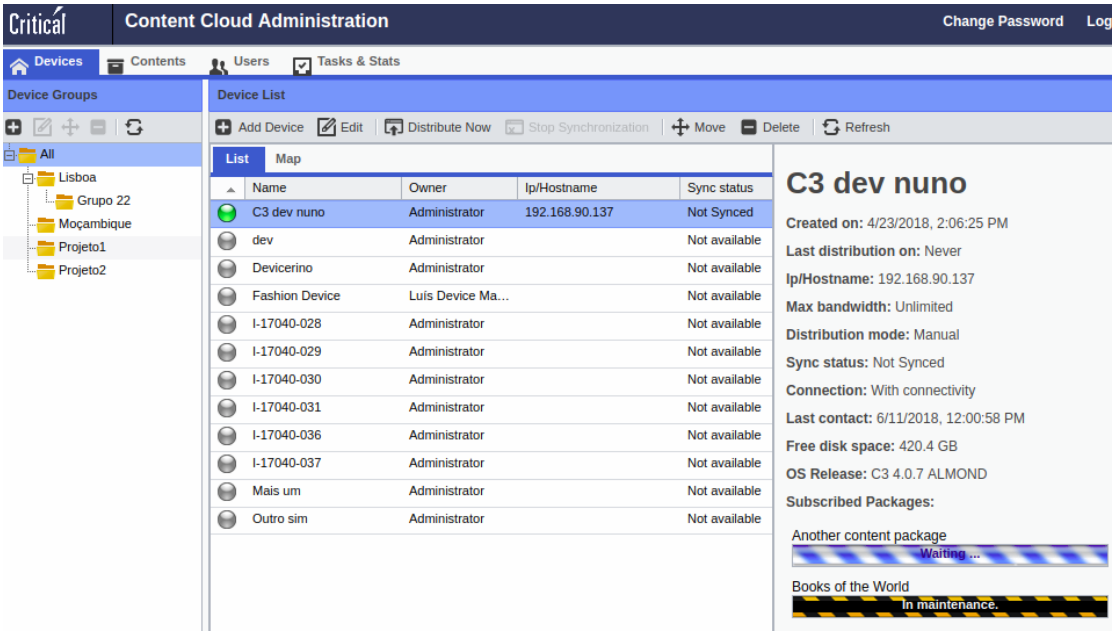


Figura 2-2 - Exemplo da página dos dispositivos do C3 Cloud v1

Por último, os utilizadores têm três tipos de perfis:

- **Content manager** – permissões para criar e gerir os seus conteúdos;
- **Device manager** – permissões para criar e gerir os seus conteúdos, e gerir os C3 a que tem acesso; permissão para registar e controlar os seus dispositivos;
- **Administrador** – acesso a todos os conteúdos e dispositivos existentes. Este perfil é o único que permite criar e modificar utilizadores. sendo assim, para aceder ao C3 Cloud é preciso que o administrador crie credenciais de acesso. Apenas existe um administrador e este tem também as mesmas funcionalidades dos utilizadores *device manager* e *content manager*.

A Figura 2-3 apresenta a página de utilizadores a que apenas o administrador tem acesso.

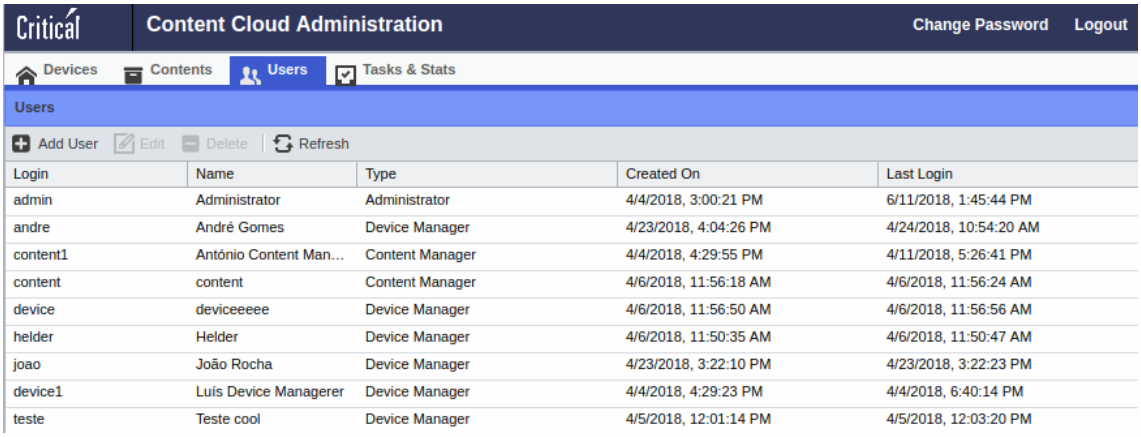


Figura 2-3- Exemplo do ecrã de utilizadores da interface web do C3 Cloud v1

2.1.3 Limitações e problemas

O C3 Cloud Control foi desenvolvido para manipular os C3 remotamente e enviar-lhes conteúdos educativos. No entanto, com o aumento do número de C3, o desempenho e estabilidade desta versão do CC diminuiu, bem como a velocidade nas transferências. A Tabela 2-1 e Tabela 2-2 contêm as principais limitações da versão 1.0 do C3 Cloud.

Tabela 2-1 - Resumo das principais limitações da solução, respetivas consequencias e ação tomada

Limitação / problema	Consequência	Ação tomada
Arquitetura centralizada: A arquitetura deste sistema de partilha de conteúdos é feita através de um servidor central para todos os C3.	Uso excessivo de largura de banda.	Alterar a arquitetura de sincronização de centralizada para descentralizada/distribuída. Ação tomada no C3 Cloud v2
Desenvolvimento parado há muito tempo: A equipa responsável pelo desenvolvimento, entretanto, saiu da empresa e não deixou documentação de desenvolvimento.	Maior curva de aprendizagem	Refazer o *backend* Ação tomada no C3 Cloud v2

Tabela 2-2 - Resumo das principais limitações da solução, respetivas consequencias e ação tomada (continuação)

Limitação / problema	Consequência	Ação tomada
Alteração de requisitos: Os requisitos mudam com o passar do tempo e alterá-los ou adicionar novos a um projeto parado é dispendioso.	Flexibilidade inexistente. A flexibilidade é um requisito não-funcional que deve estar presente para ser possível alterar requisitos e/ou adicionar software mais facilmente.	Refazer o *backend*. Ação tomada no C3 Cloud v2
Baixo suporte para manipular vários C3: A arquitetura do sistema não foi desenvolvida para permitir manipulação de vários dispositivos e, como tal, não existem certas operações, tais como adicionar um conteúdo a um grupo de dispositivos.	No caso de haver um número elevado de dispositivos, existe má experiência de utilizador com a UI.	Implementar uma camada intermédia para adaptar a novos requisitos, tais como subscrever um conteúdo a múltiplos dispositivos com apenas um pedido HTTP. Ação tomada no C3 Cloud v1.5
Interface do utilizador não apelativa: A interface não é responsiva e está antiquada com componentes não intuitivos e difíceis de compreender.	Má experiência de utilizador e de design.	Aplicar novas técnicas e componentes mais intuitivos e de fácil acesso. Ação tomada no C3 Cloud v1.5

2.2 C3 Cloud Control v1.5

Com o aumento do número de dispositivos C3, conteúdos e utilizadores, houve uma necessidade de mudança e de encontrar uma solução que permitisse manipular todos estes dados de forma facilitada, com mais funcionalidades em relação à versão anterior.

Esta seção exibe a versão 1.5 do serviço, descrevendo a arquitetura dos componentes, os requisitos desenvolvidos e, por fim, os principais objetivos alcançados.

2.2.1 Arquitetura

Nesta versão intermédia do C3 Cloud, foram criadas duas camadas na arquitetura: uma de interface gráfica, também chamada de *frontend*, e outra, denominada *backend*, que inclui uma camada intermédia (um *broker* que expõe uma API REST) e o *backend* existente do C3 Cloud v1 (Figura 2-4).

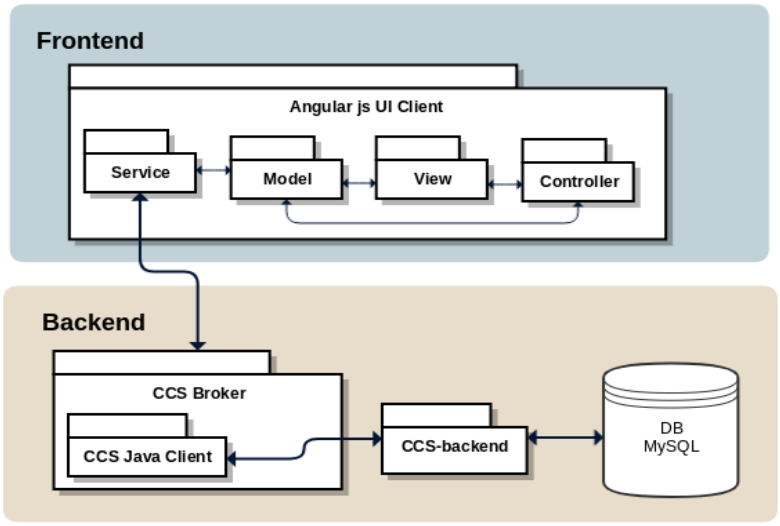


Figura 2-4- Arquitetura do C3 Cloud na versão 1.5

A camada intermédia, designada *adapter* ou *broker*, possui como função aumentar a flexibilidade ao orquestrar os pedidos HTTP enviados pelo *frontend* através da API REST, permitindo uma maior adaptabilidade para fazer alterações, o que não seria possível sem a sua existência (ou seja, se o acesso fosse diretamente feito pelo *frontend* ao CCS *backend*).

2.2.2 Aspetos relevantes da implementação

A equipa que desenvolveu o projeto C3 Cloud v1.0 já não se encontra disponível e o código já não tem suporte. No entanto, essa equipa deixou uma biblioteca em Java (*CCS Java Client*) que implementa os métodos existentes no *backend* v1.0, sendo possível comunicar e controlar o C3 Cloud v1.0 com esta biblioteca.

Os aspetos mais relevantes envolvidos no desenvolvimento desta versão 1.5 foram:

- A camada da UI (*User Interface*) foi completamente remodelada, fornecendo ao utilizador uma nova experiência de utilização;
- Foi desenvolvida uma camada adicional, o *broker*, promovendo maior flexibilidade.

O *broker* expõe uma API REST que permite receber os pedidos vindos do *frontend*, feitos pelo utilizador, e utiliza a biblioteca *CCS Java Client* para comunicar com o *backend* para devolver a resposta desejada. O maior foco de desenvolvimento nesta versão consistiu em criar uma API REST standard e bem documentada. Para isso, foi escolhido e documentado um conjunto de regras e melhores práticas para uma melhor escrita de código e desenvolvimento, tais como:

- Validações de inputs;
- Versionamento da API – para o caso de alterações significativas;
- Organização e estruturação de código;
- Pedidos e respostas consistentes em JSON.

2.2.3 Problemas do CCC v1.0 corrigidos

Esta fase do estágio possibilitou resolver alguns problemas indicados na Tabela 2-1 e Tabela 2-2, relativos ao design do website e, por consequente, à melhoria da experiência do utilizador. Esta camada intermédia, o *broker*, possibilitou a extensão de pedidos novos que não foram pensados na 1ª versão, sendo estes relativos à gestão de um aglomerado de dispositivos C3, coordenando utilizadores, conteúdos e definições dos servidores C3.

Uma das limitações da interface na versão 1.0 consiste na edição de vários dispositivos ao mesmo tempo, quer seja editar definições de sincronização e conexão ou subscrever/remover conteúdos e/ou utilizadores. Existe o conceito de grupo, mas não é possível editar as propriedades de todos os dispositivos dentro deste, sendo preciso editar individualmente os dispositivos pretendidos.

A Figura 2-5 evidencia o problema de gerir conteúdos de cinco C3 presentes no grupo "Lisboa" (neste exemplo, a funcionalidade de adicionar dois conteúdos aos cinco C3 do grupo "Lisboa"). Após selecionar o grupo, é necessário editar manualmente os dispositivos, adicionando os conteúdos pretendidos, um dispositivo de cada vez. Neste exemplo são apenas 5 dispositivos, mas pode haver grupos de cem dispositivos, o que torna difícil a manutenção destes. Esta limitação não existe apenas para adicionar/remover conteúdos, mas também para gerir utilizadores e definições da conexão.

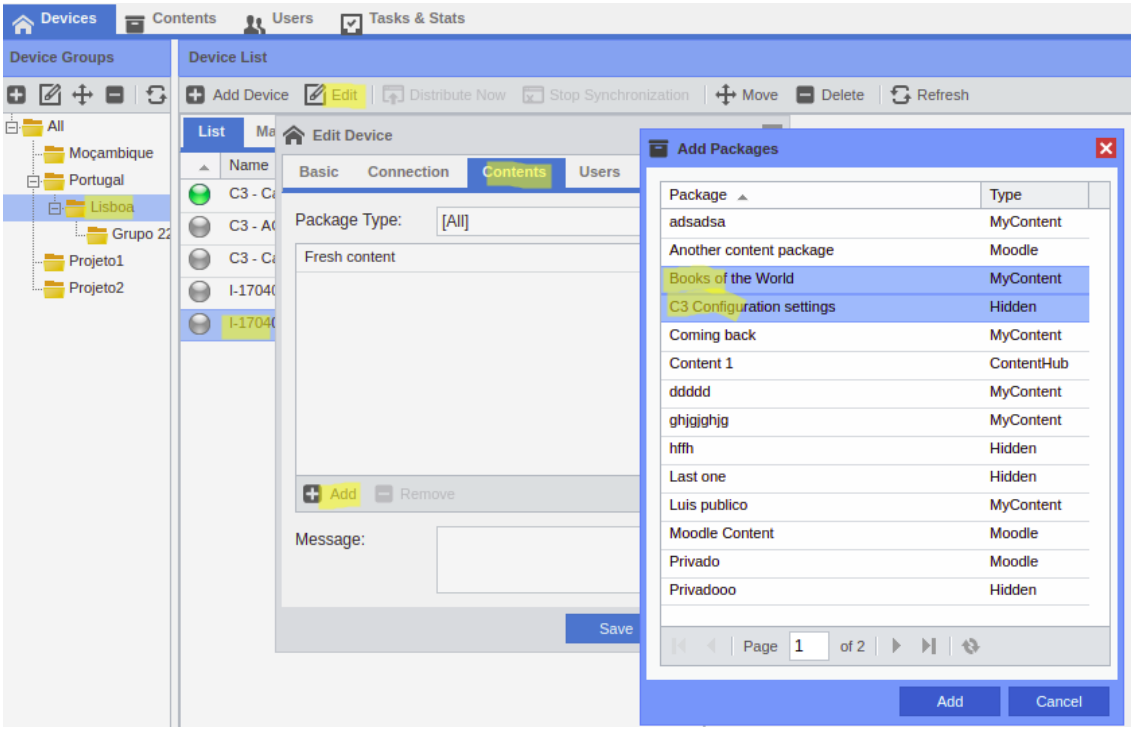


Figura 2-5 - Limitação de edição de múltiplos dispositivos na versão 1.0 do C3 Cloud

A versão 1.5 da solução foi implementada com o objetivo de enfrentar limitações de escalabilidade como a que a foi apresentada na Figura 2-5, pelo que a camada intermédia desenvolvida nesta fase, o *adapter*, permitiu adicionar novos pedidos HTTP e, desta forma, gerir vários dispositivos ao mesmo tempo, conforme ilustrado na Figura 2-6. Do lado esquerdo dessa figura, estão os grupos disponíveis e uma listagem dos respetivos dispositivos. Do lado direito está uma caixa de diálogo com duas colunas de conteúdos que são: todos os conteúdos e conteúdos no grupo. Para adicionar basta carregar no conteúdo pretendido na coluna com todos os conteúdos (tem um ícone verde de adicionar).

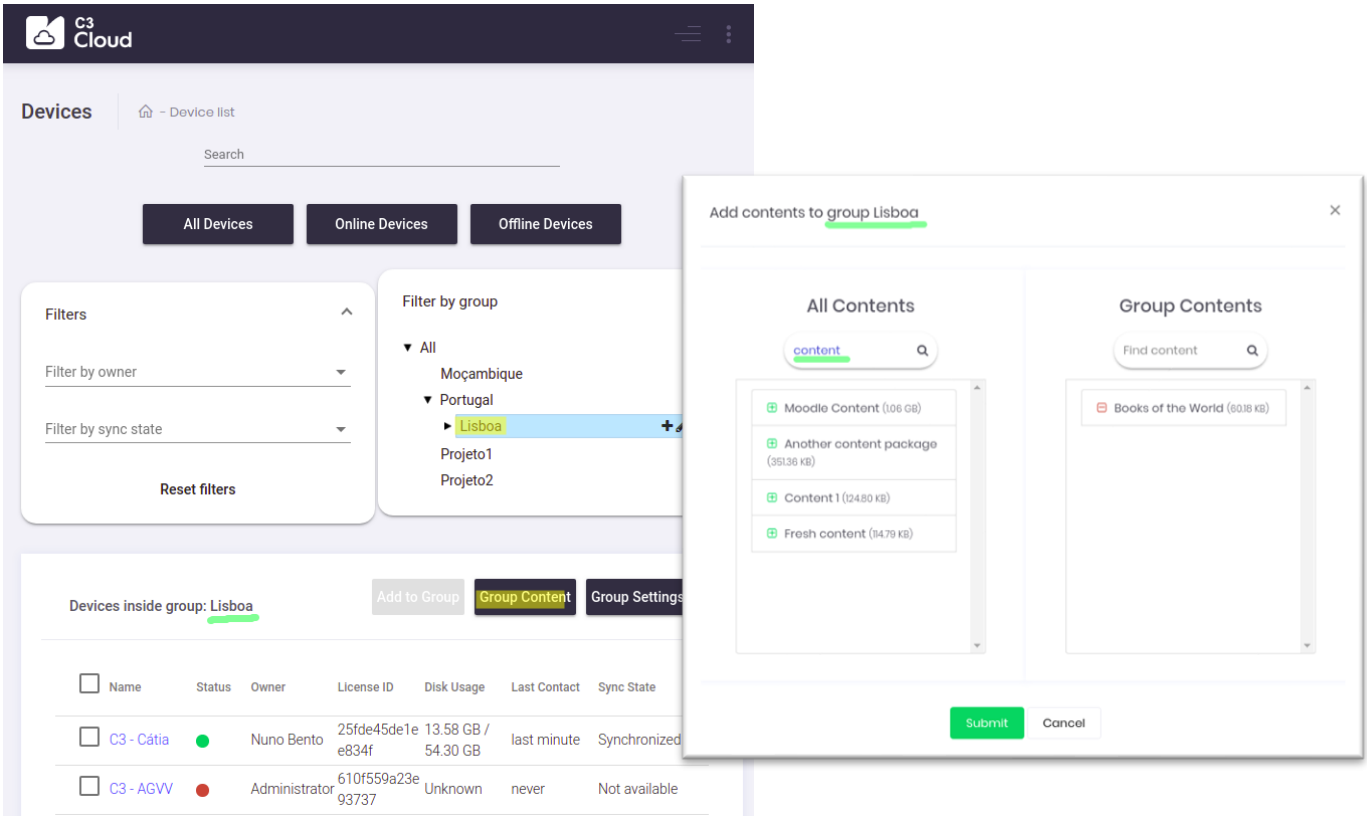


Figura 2-6 - Solução ao problema de edição de vários dispositivos na versão 1.5 do C3 Cloud

Na versão 1.0 do C3 Cloud, a interface apresenta mais limitações ao nível de usabilidade. Por exemplo, não é possível procurar dispositivos ou conteúdos específicos e a criação de conteúdos não é trivial. A página de dispositivos e conteúdos é extensa, o que torna a navegação complicada, não sendo possível procurar pelo nome. Após criar um conteúdo na interface gráfica, apenas é possível editar detalhes e a foto de *preview* de conteúdo, chamada *thumbnail*.

A Figura 2-7 mostra um conteúdo criado na interface que fica em modo de manutenção, que significa que está disponível para ser modificado através de um cliente FTP.

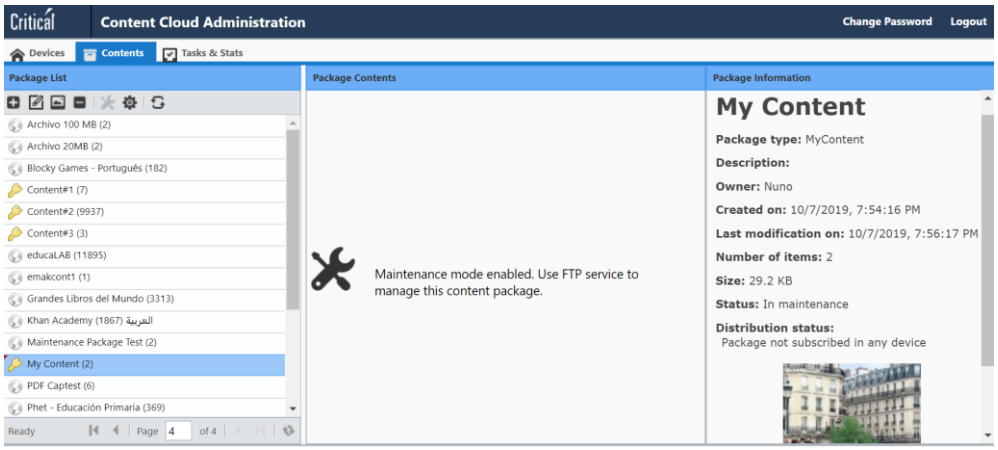


Figura 2-7 - Limitação de upload de conteúdos para a cloud na versão 1.0 do C3 Cloud

A Figura 2-8 mostra o sistema de *uploads* da versão 1.5 com informação relativa aos ficheiros e que veio solucionar o problema de *upload* de conteúdos da versão 1.0. Esta mostra também uma árvore de ficheiros deste conteúdo que permite organizar os ficheiros e pastas arrastando-as com o rato para a pasta pretendida. É possível criar pastas para melhor organização do conteúdo.

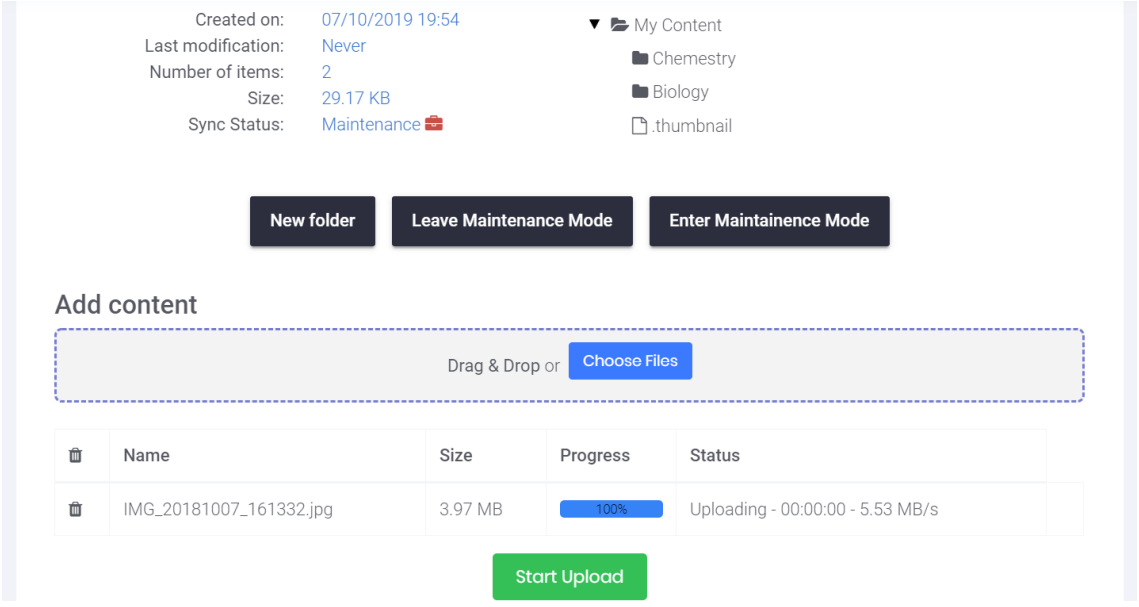


Figura 2-8 - Solução para melhor usabilidade no upload de conteúdos para a cloud na versão 1.5 do C3 Cloud

Foi solucionado o problema da procura facilitada de dispositivos e de conteúdos utilizando uma barra de pesquisa e aplicando filtros. A Figura 2-9 mostra a página de dispositivos na versão 1.5, em que aparece uma barra de pesquisa e filtros genéricos sobre os dispositivos, tais como *online* e *offline*, ou o estado atual de sincronização.

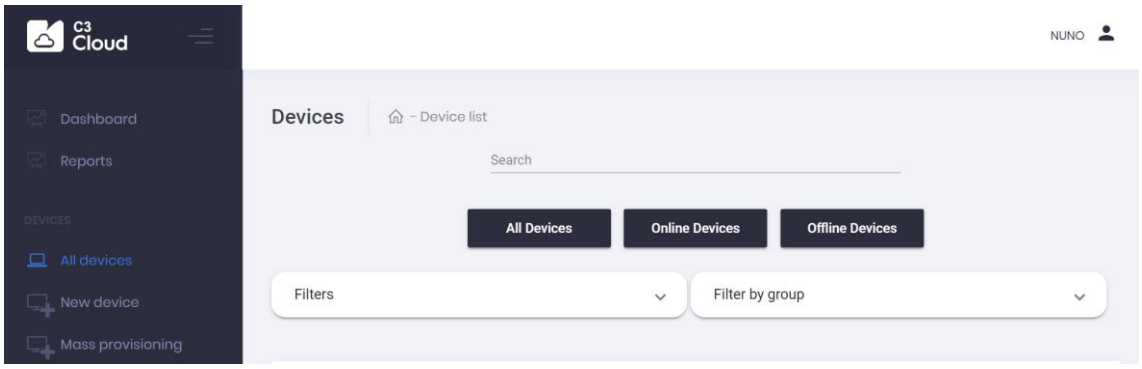


Figura 2-9 - Solução para procurar facilmente dispositivos na versão 1.5 do C3 Cloud

2.3 C3 Cloud Control 2.0

Após uma grande remodelação na interface gráfica, chegou a parte da remodelação da estrutura da parte funcional do C3 Cloud, que foi lançada em janeiro de 2019 [6]. Esta fase consistiu em replicar toda a lógica do negócio existente e, por fim, aplicar melhoramentos. Também foi reformulada a arquitetura de sincronização entre o servidor C3 Cloud e os clientes C3.

A Figura 2-10 ilustra a topologia de rede do tipo *peer-to-peer* aplicada na versão 2.0. Neste exemplo, existem duas redes locais que representam escolas, em que apenas alguns dispositivos C3 têm conectividade à internet (os C3 verdes). Estes conseguem comunicar com o C3 Cloud e com outros C3 que estejam *online*, independentemente da escola onde se encontram. Os restantes C3, sem conectividade, podem obter conteúdos comunicando com os C3 que estejam na mesma rede.

Com esta versão, foram resolvidos os problemas identificados na tabela 1, em específico os relacionados com o *backend*:

- **Adaptabilidade e Flexibilidade –** O *backend* foi modernizado e estruturado para ser adaptável a possíveis mudanças de requisitos, mantendo as funcionalidades existentes na primeira versão do C3 Cloud;
- **Escalabilidade** – Foi alterada a topologia de rede, deixando esta de estar baseada num único servidor central. Os C3 atuam como servidores e clientes, sendo a carga distribuída por estes em vez de centralizada apenas no C3 Cloud Control;

- **Eficiência** – A rede obtida do tipo *peer-to-peer* permite mais conexões e, assim, distribuir a carga entre vários servidores de forma mais eficaz e obter melhores velocidades de distribuição (ver resultados do estudo descrito na secção 5.2).

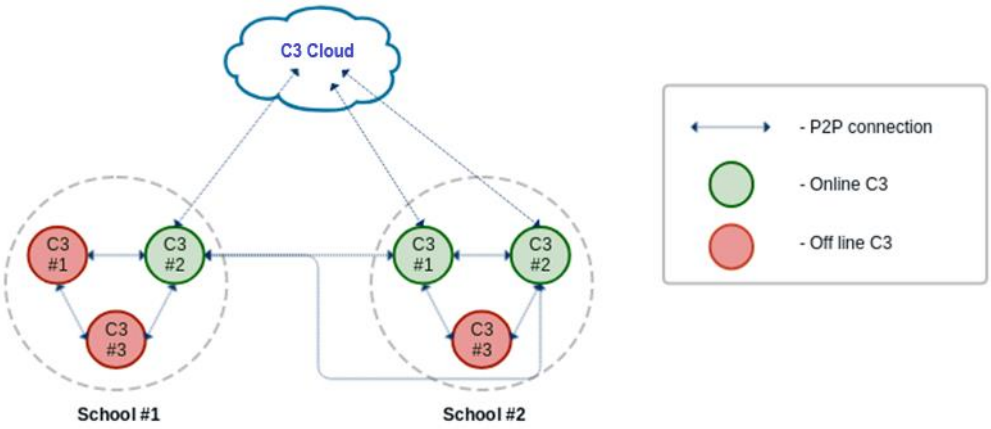


Figura 2-10- Exemplo da topologia de rede do C3 Cloud v2 com os C3

Em relação à segurança do sistema, este está protegido de atacantes cibernéticos através de encriptação TLS v1.2 [7], o que impossibilita a descodificação do tráfego intercetado. Outra medida de segurança consiste em não ser possível entrar na rede privada do C3 sem "convite", significando que apenas são admitidos dispositivos conhecidos e válidos. Mais detalhes sobre segurança encontram-se na secção 3.3.

2.4 Análise Comparativa do C3 Cloud Control

A Tabela 2-3 mostra a evolução do serviço C3 Cloud comparando as características chave entre as três versões deste.

De seguida, apresentam-se as definições das características mais técnicas (menos intuitivas) presentes na Tabela 2-3, de forma a esclarecer o significado de cada uma:

- **Agendamento** – Agendar as sincronizações para determinadas alturas do dia e/ou para determinados dias da semana;
- **Deteta conflitos** – Mostra se existem servidores C3 ou conteúdos com problemas a sincronizar;
- **Versionamento** – Guarda versões de conteúdos anteriores e permite repor uma dessas versões, caso seja necessário;

- **Compressão**– Permite comprimir meta dados, que são dados relativos ao *software* de sincronização, e também pacotes de dados relativos aos ficheiros que estão a sincronizar;
- **Suporta mudar o nome / mover de pasta** – Permite alterar o nome ou a localização dos ficheiros no servidor. O servidor remoto também altera o nome/caminho do ficheiro. Soluções que não tenham esta característica apagam o conteúdo e voltam a sincronizar;
- **Discovery Server** – Permite encontrar novos servidores C3 na rede local ou através da internet.

Tabela 2-3 - Comparação de características chave das três versões do C3 Cloud

Característica	*CC v1.0*	*CC v1.5*	*CC v2.0*
Arquitetura Centralizada	✔	✔	
Agendamento	✔	✔	✔
Dados encriptados	✔	✔	✔
Compressão	✔	✔	✔
Suporte na UI para vários C3		✔	✔
UI/UX moderno		✔	✔
Arquitetura P2P			✔
Escalável			✔
Upload de ficheiros pela interface web			✔
Deteta conflitos			✔
Suporta mudar o nome / mover de pasta			✔
Discovery Server			✔

3 OTIMIZAÇÃO DA SINCRONIZAÇÃO DE CONTEÚDOS

Este capítulo apresenta o resultado de um estudo sobre ferramentas de sincronização e, também, sobre as topologias de rede mais conhecidas. Estes estudos foram efetuados com o objetivo de apoiar nas decisões do desenvolvimento do *backend* do C3 Cloud v2.0. É apresentada a ferramenta de sincronização escolhida e, ainda, a análise detalhada sobre a mesma que conduziu à conclusão da sua viabilidade.

3.1 Topologias de Redes

Há muitos aspetos a ter em conta ao criar uma rede de conteúdos educativos destinados a instalações educacionais, como escolas ou universidades, nomeadamente a escolha de uma topologia adequada para possibilitar a sincronização de vários servidores, de forma segura e privada. No caso da solução C3, deve ser um serviço escalável para poder receber e sincronizar mais de 10 000 servidores. Podem ser identificadas três topologias de rede principais, como mostra a Figura 3-1. Estas têm os conceitos subjacentes de nó (servidor e/ou cliente) e nó central (servidor central), ou seja, os C3 e o C3 Cloud, respetivamente, no caso do projeto descrito neste documento.

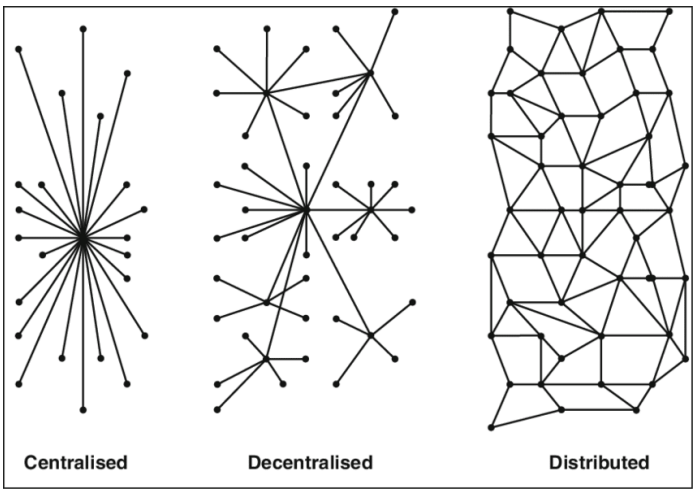


Figura 3-1 - Principais topologias de rede [8]

Uma topologia centralizada [8] tem o benefício de permitir o controlo e supervisionamento de todos os nós, acedendo a apenas a um nó central (servidor central) com informações e possíveis configurações dos restantes (clientes). Por outro lado, se este deixar de estar operacional, pode ficar caótico para a rede, sendo esta uma situação de ponto único de falha.

As topologias de rede descentralizada e distribuída, também designada *peer-to-peer*, permitem aumentar a escalabilidade ao nível do número de nós na rede. Uma rede descentralizada permite ter mais nós centrais, evitando a existência de um único ponto de falha e permitindo a distribuição de carga. Uma rede distribuída permite que os nós consigam distribuir os conteúdos entre si. No entanto, não existindo um ponto central para geri-los, torna-se complexa de manter e controlar.

Para solucionar o problema de distribuição de conteúdos e escalabilidade de servidores na solução C3, a estratégia a adotar consistem em combinar as vantagens das três topologias referidas. Desta forma, é possível ter o benefício da gestão centralizada de servidores e conteúdos e, ainda, as vantagens da abordagem descentralizada e distribuída, sendo o objetivo a distribuição de conteúdos por todos os nós. A Figura 3-2 mostra um exemplo da rede distribuída e centralizada. Esta tem três servidores centrais e vários servidores conectados uns com os outros (abordagem *peer-to-peer*, conforme mencionado na secção 2.3).

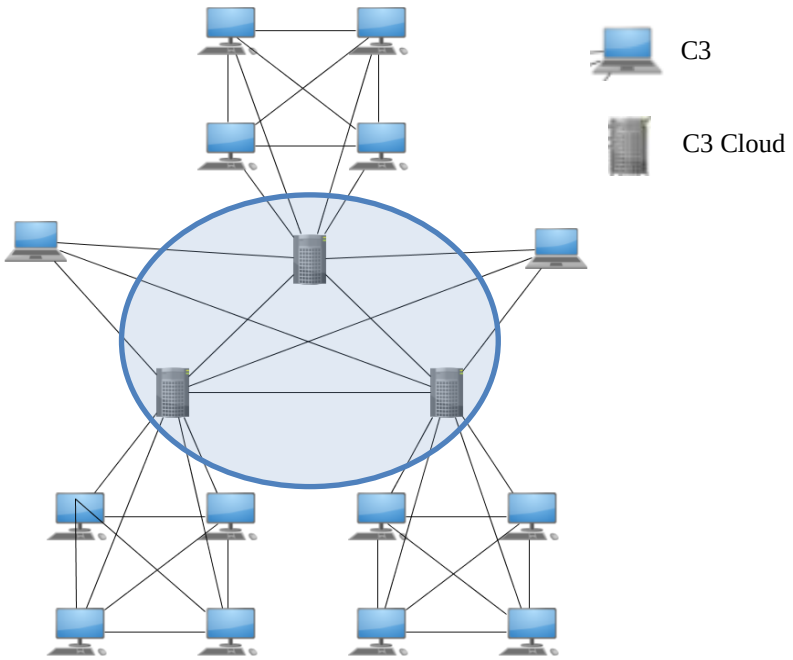


Figura 3-2 - Exemplo de topologia de rede centralizada e distribuída [8]

3.2 Ferramentas de Sincronização

Existem inúmeras ferramentas para sincronizar os conteúdos entre servidores. No entanto, algumas são mais indicadas para casos em que é preciso ter uma rede central e distribuída ao mesmo tempo. Embora existam muitos serviços conhecidos de sincronização e partilha de

conteúdos, como Dropbox [9], OneDrive [10] ou Google Drive [11], a maioria deles apresenta as seguintes características:

- Abordagem cliente-servidor, em que os dados residem em grandes *data centers* externos;
- *Software* proprietário (*closed-source*);
- O utilizador deve concordar com os respetivos preços e termos de serviço;
- O utilizador não tem acesso ao código fonte do *software*, sendo retirado o conhecimento técnico do programa;
- Baixas possibilidades de inovação no código-fonte.

Encontrar a ferramenta certa é certamente uma ajuda na criação da topologia de rede descrita na seção 3.1. A Tabela 3-1 apresenta várias ferramentas de sincronização de ficheiros com as características técnicas pretendidas. As seguintes definições servem para clarificar o significado de algumas das suas colunas:

- **Agendamento** – permite agendar as sincronizações para determinadas alturas do dia e/ou para determinados dias da semana;
- **Mover ou renomear** – quando um ficheiro/pasta é movido/renomeado, os servidores cliente vão detetar que o ficheiro/pasta foi excluído e a sincronização é feita novamente. Os servidores cliente com esta característica vão repetir a ação efetuada no central, economizando largura de banda;
- **Portátil** – a ferramenta foi implementada para ser executada sem necessidade de modificar a configuração do computador onde é instalada. As aplicações podem ser executadas em qualquer computador para o qual foram projetados, mesmo que o utilizador não tenha privilégios administrativos no servidor.
- **Sincronização** – indica o tipo de arquitetura de rede.

Conforme já mencionado, o tipo de sincronização pretendido para a solução C3 deve ser distribuído e *peer-to-peer*, como mostra a Figura 3-2. Numa rede *peer-to-peer*, cada um dos servidores funciona tanto como cliente como servidor, permitindo a distribuição de conteúdos mesmo sem a presença de um servidor central. Das ferramentas de sincronização analisadas, apenas o Resilio e o Syncthing conseguem criar a rede distribuída. No entanto, o Syncthing possui todas as características pretendidas, além de ser *open-source* e ter atingido um ponto de maturidade elevado. Foi, por estas razões, a solução de sincronização adotada.

Tabela 3-1 - Comparação de ferramentas de sincronização

	Agendamento	Open Source	Mover/ Renomear	Portátil	Software Livre	Sincronização
Rsync [12]	Parcial (ferramenta externa)	✔	x	✔	✔	Cliente e Servidor
SeaFile [13] (Community Edition)	✔	✔	✔	x	✔	Cliente e Servidor
OwnCloud [14]	✔	✔	✔	✔	✔	Cliente e Servidor
Resilio [15]	Premium	✔	✔	✔	Parcial	Distribuída *peer-to-peer*
Syncthing [16]	✔	✔	✔	✔	✔	Distribuída *peer-to-peer*

3.3 Análise da ferramenta Syncthing

O Syncthing é uma ferramenta de sincronização de ficheiros que cria uma rede distribuída, do tipo *peer-to-peer*, como pretendido. É simples e fácil de configurar. Possui, uma interface gráfica disponível no navegador web que mostra o que está a acontecer de maneira intuitiva.

"*Your data is your data alone and you deserve to choose where it is stored, whether it is shared with some third party, and how it's transmitted over the internet*", é a primeira frase a aparecer no site oficial [17] e acaba por ser o conceito principal da ferramenta. A privacidade e segurança dos ficheiros são alguns dos conceitos chave do Syncthing. A Tabela 3-2 apresenta as principais características desta ferramenta de sincronização.

Tabela 3-2 - Principais características da ferramenta Syncthing

Característica	Descrição
Estável contra perda de dados	O principal objetivo do Syncthing é o transporte confiável de informações. Não há recursos adicionais, como calendários e edição de documentos, pois o interesse principal é sincronizar ficheiros.
Seguro	Protegido com autenticação mútua com protocolo TLS (*Transport Layer Security*) [7], isto é, dois dispositivos apenas se conectam se ambos concordarem manualmente. A característica "introdutora" do Syncthing, fornece uma maneira de aceitar automaticamente estas aprovações explícitas para ingressar na rede. Todos os dados são criptografados usando TLS.
Compressão	A sincronização tem uma opção para compactar os pacotes enviados para dispositivos, o que é desejável para a solução C3. Três opções podem ser definidas: • Nenhum – para desativar a compressão; • Meta-dados – para comprimir pacotes de meta-dados; • Sempre – para comprimir todos os pacotes, incluindo dados de ficheiros.
Popular	Tem mais de 45 milhões de nós [18] e 28 200 estrelas no GitHub (em outubro de 2019) [19].
Nós mortos permitidos	Todas as instalações possuem os mesmos recursos e, se um nó sair da rede, a sincronização não será interrompida se houver outros nós com os mesmos conteúdos sincronizados. Caso não haja nós a serem conectados ou todas as conexões tenham sido perdidas, quando as conexões voltarem, a sincronização continuará onde foi interrompida pela última vez.
Plataforma cruzada	Este software tem compatibilidade em vários sistemas operativos, até mesmo em Android.

A Figura 3-3 mostra o crescimento da utilização do Syncthing ao longo do tempo. O eixo Y representa o número de dispositivos com o syncthing instalado, com cores por cada versão, a nível mundial. No entanto, no gráfico só aparecem as instalações em que os utilizadores aceitam o envio de relatórios anónimos de utilização da ferramenta e muitos podem optar por não o fazer, não entrando na contabilização do gráfico.

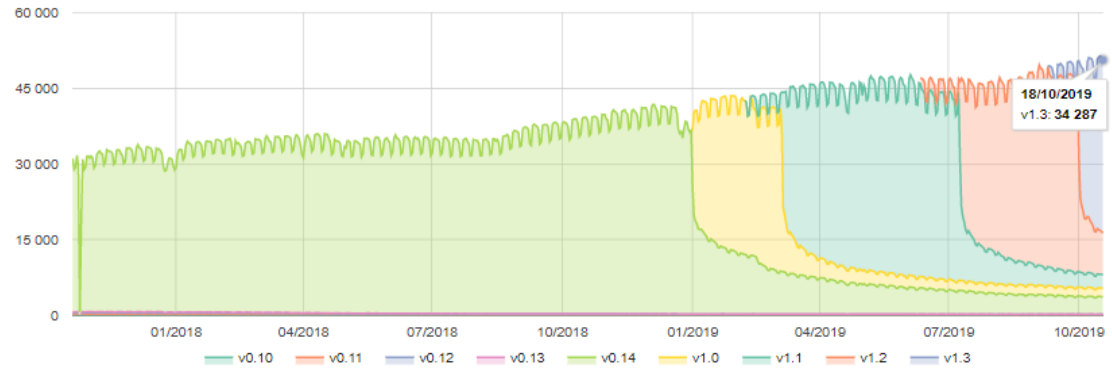


Figura 3-3 - Número de dispositivos com o syncthing instalado, ao longo do tempo e por cada versão, a nível mundial [18]

3.4 Centralizar o Syncthing

Apesar de todas as características distribuídas e descentralizadas inerentes ao Syncthing, falta a gestão centralizada de todos os nós e conteúdos existentes na rede. O Syncthing foi desenvolvido para ser descentralizado. No entanto, pode ser configurado para ter a gestão central, conforme procurávamos para a solução C3 Cloud. Para centralizar o Syncthing, é necessário ter:

- O nó central com conteúdo em *send only* - a intenção deste modo de conteúdo é de armazenar a "cópia principal". Não é esperado que esses ficheiros sejam alterados por outros dispositivos, ou seja, qualquer alteração é indesejável.
- Os restantes nós com conteúdo em *receive only* - este modo é o oposto lógico de *send only*. Esse modo é útil para destinos de replicação, destinos de *backup* ou outros casos em que nenhuma modificação local é permitida.
- O nó central em modo introdutor - permite que um dispositivo adicione automaticamente novos dispositivos. Quando dois dispositivos se conectam, estes trocam uma lista de conteúdos partilhados, e, automaticamente, o nó central vai apresentar novos dispositivos para ajudar a sincronizar.

A Figura 3-4 exemplifica como as conexões podem ser estabelecidas usando um nó central. A parte de cima mostra a configuração inicial com 3 nós, onde o nó central está conectado a outros nós e possui 2 conteúdos. Os dois servidores estão configurados para confiar no servidor central como introdutor, o que significa que quando o nó central partilha conteúdos com ambos os servidores (na figura o conteúdo partilhado é álgebra), este vai apresentar os servidores que querem o mesmo conteúdo. Desta forma, o nó 1 adiciona automaticamente o nó 2 uma vez que tem o mesmo conteúdo para sincronizar (álgebra). A configuração de confiar no nó central sendo o introdutor é muito importante para conectar os C3 automaticamente uns aos outros, sem precisar de aprovar a conexão manualmente.

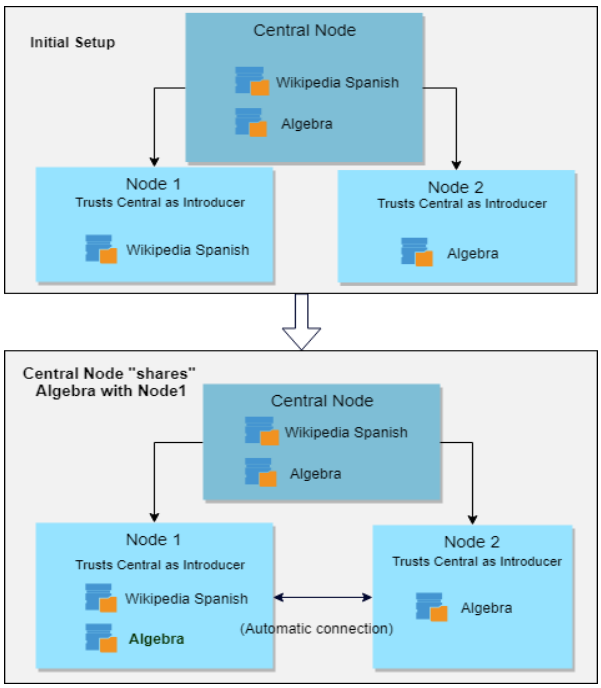


Figura 3-4 - Exemplo de funcionamento da característica Introdutora

3.5 Integração com o C3 Cloud

Para começar, é necessário ter a ferramenta Syncthing instalada no servidor central e também nos dispositivos C3. As instalações nos C3 devem ter as pastas configuradas como *receive only* enquanto que no servidor central devem ser *send only*. Além disto, os servidores C3 devem confiar no servidor central como sendo introdutor. Desta forma, todos os nós clientes adicionarão o servidor central automaticamente, partilhando uma lista de dispositivos e conteúdos em comum para os C3 se conhecerem automaticamente.

O Syncthing tem uma interface gráfica web, ilustrada na Figura 3-5, para gerir os dispositivos e conteúdos conhecidos, entre outras definições. No entanto, não é obrigatório utilizar esta interface para controlar o Syncthing. A ferramenta expõe uma API REST que permite controlar o serviço. O C3 Cloud v2.0 faz pedidos ao Syncthing através desta API REST, sendo assim possível criar e gerir os nós e conteúdos da rede, entre vários aspetos dos conteúdos e nós que existem.

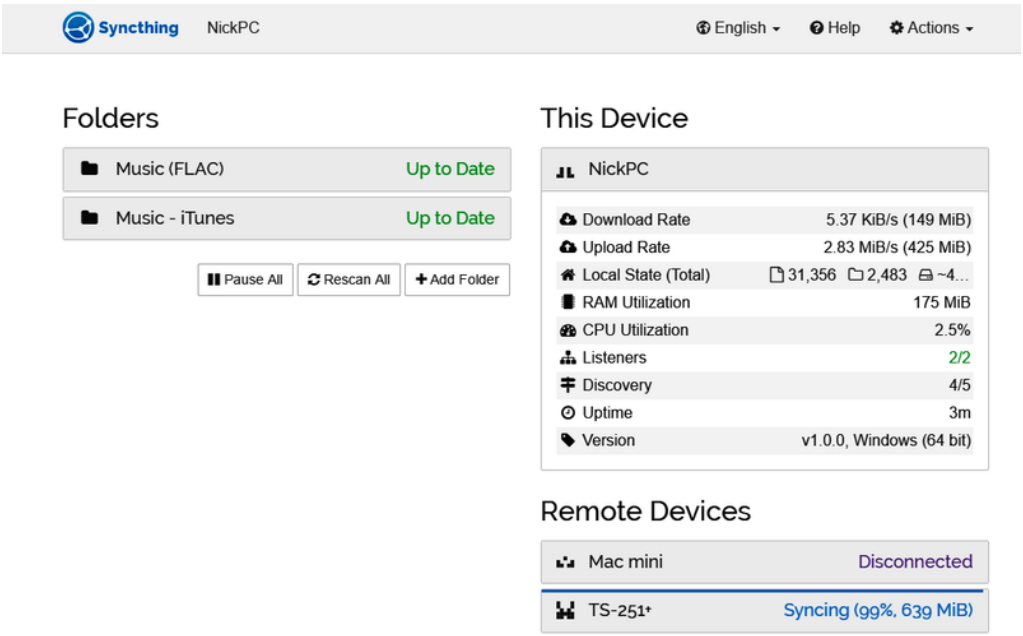


Figura 3-5 - Screenshot de exemplo da interface gráfica da ferramenta Syncthing

3.6 Funcionamento

O Syncthing permite a partilha de pastas numa topologia distribuída do tipo *peer-to-peer* (P2P). Não há servidor central ou sistemas de terceiros para guardar ou gerir o conteúdo, o que significa que os conteúdos residem nos servidores da rede. Todos os servidores são validados pelo Syncthing para que se conectem e comecem a partilhar conteúdos diretamente entre si, ou seja, sem o recurso a servidores intermediários.

Em alguns casos, a conexão direta entre nós não é possível por algum motivo como, por exemplo, não ser permitido abrir o porto necessário da *firewall* de um determinado dispositivo. Nesses casos, o Syncthing ainda pode funcionar, mas com uma estratégia de conexão diferente que passa pela utilização de um servidor de retransmissão que serve de intermediário da conexão entre os dois nós. Estes servidores não conseguem ver os dados que estão a ser retransmitidos uma vez que estão encriptados com TLS. Os servidores de retransmissão são públicos e gratuitos para utilizadores da ferramenta. No entanto, é possível instalar o servidor de retransmissão num servidor privado, visto que o *software* e toda a documentação é disponibilizado pelos criadores do Synthing [20]. Desta forma pode ter-se mais privacidade nos dados transmitidos, e, ainda, obter melhores desempenhos da funcionalidade de retransmissão, uma vez que retransmite apenas dados dos C3 que aí estão conectados.

4 DESENVOLVIMENTO DO C3 CLOUD 2.0

Este capítulo explica os principais aspetos de desenvolvimento do C3 Cloud Control 2.0, incluindo a arquitetura do sistema e as ferramentas utilizadas.

4.1 Planeamento

Um dos objetivos deste projeto consistiu em substituir o serviço C3 Cloud por outra versão com componentes mais atuais e que conseguisse escalar com o aumento de clientes. Para este fim, foi planeado substituir os componentes do serviço de forma faseada e suave, o que resultou nas seguintes três fases, que também já foram mencionadas na seção 1.4:

- **Fase 1**: desenvolvimento de uma API REST standard para comunicar com o C3 Cloud existente;
- **Fase 2**: desenvolvimento de uma interface web que recorresse à API REST criada na fase anterior;
- **Fase 3**: desenvolvimento da nova versão do *backend* do C3 Cloud e substituição da anterior (v1).

As fases 1 e 2 estão associadas à versão intermédia do C3 Cloud (v1.5), com dois novos componentes: uma interface gráfica do utilizador e uma API REST. Nestas duas fases, o objetivo consistiu em construir uma nova interface gráfica do utilizador sem ter de refazer toda a lógica (*w23q*).

De forma a perceber-se a complexidade e a lógica da versão inicial do C3 Cloud, foi realizado um estudo do serviço que está descrito na seção 2.1. Com este estudo, foi possível entender o funcionamento e, assim, manter as funcionalidades existentes. Também foi documentada a API REST da versão 1.0 que serviu de modelo de base para criar a API REST da versão 1.5 e, posteriormente, da 2.0.

4.2 Requisitos

Sendo que foram utilizadas metodologias ágeis no desenvolvimento do projeto, apenas foi registada a documentação necessária. De facto, existe uma tendência em documentar menos quando se usa um método ágil, já que esta abordagem valoriza a comunicação direta entre os programadores e também com o cliente.

4.2.1 Requisitos funcionais

Uma vez que se pretendeu manter as funcionalidades da primeira versão do C3 Cloud Control, a maioria dos requisitos funcionais foram definidos a partir do que já existia. A definição dos

requisitos permitiu a criação da API REST da versão 2.0, que se encontra especificada no anexo C (Documentação C3 Cloud 2.0). De seguida, estão enumerados os requisitos funcionais definidos para desenvolvimento, estando estes organizados por categorias:

- Autenticação:
  - Log in – entrar na aplicação com credenciais de acesso;
  - Log out – sair da aplicação;
  - Obter informação sobre o utilizador atualmente *logado*;
- Dispositivos:
  - Pesquisar todos – apenas os que tem acesso;
  - Pesquisar dispositivo por identificador;
  - Registar dispositivo;
  - Editar;
  - Remover;
  - Dar ou remover acesso – é necessário acesso ao dispositivo para dar/remover acesso a outros utilizadores;
- Conteúdos:
  - Pesquisar todos – conteúdos de acesso publico e os que o utilizador logado criou;
  - Pesquisar conteúdo por identificador;
  - Criar conteúdo – atribuir um nome, descrição, língua, *thumbnail* (imagem) e tipo de conteúdo;
  - *Upload* de ficheiros – deve ser possível fazer *upload* de ficheiros e pastas para um conteúdo;
  - *Upload* de ficheiros – deve ser possível fazer *upload* de ficheiros e pastas para um conteúdo;
  - Editar – apenas o criador;
  - Apagar – apenas o criador;
  - Subscrever conteúdo(s) a dispositivo(s);
  - Remover conteúdo(s) de dispositivo(s);
- Utilizadores:
  - Pesquisar todos;
  - Pesquisar utilizador por identificador;
  - Criar utilizador;
  - Editar;
  - Remover.

A Tabela 4-1 contém um resumo do anexo B (Análise do C3 Cloud Control v1.0), com informação acerca dos perfis de utilizador e o respetivo acesso às funcionalidades do C3 Cloud.

Tabela 4-1 - Permissões por perfil de utilizador

Perfil	Acesso e Permissões	Característica
Administrador	• Acesso a qualquer C3 ou conteúdo; • Único perfil que permite criar utilizadores;	Só existe um que gere todo o C3 Cloud
Gestor de C3	• Pode registar dispositivos; • Acesso aos C3 e conteúdos a que tem acesso;	
Gestor de conteúdo	• Acesso aos conteúdos que tem acesso;	

4.2.2 Requisitos não funcionais

Existiram vários requisitos não-funcionais (atributos de qualidade e metas a atingir) a ter em consideração para o desenvolvimento da versão 2.0:

- **Usabilidade** – Pretende-se que a interface consiga transformar operações complexas com uma interface de fácil interação. Para isto foram utilizados componentes (botões, listas, tabelas, entre outros) com um design mais moderno e consistente por toda a aplicação web e, assim, melhorar a experiência do utilizador;
- **Disponibilidade** – Pretende-se que este serviço esteja online e disponível para todos os C3 que estão conectados e utilizadores que utilizem a interface do serviço;
- **Escalabilidade** – O C3 Cloud nesta versão deve estar preparado para escalar com o aumento do número de clientes e conexões, uma vez que a topologia é distribuída;
- **Segurança** – Acessos não autorizados ao sistema não são permitidos, devendo este estar protegido contra ataques cibernéticos para que não seja possível extrair informação a partir do tráfego intercetado;
- **Adaptabilidade e Flexibilidade** – No caso de serem pedidos novos requisitos, a arquitetura do sistema deve estar organizada e preparada de forma a suportar alterações por parte da equipa de desenvolvimento.

4.3 Arquitetura do Sistema

O C3 Cloud inclui um serviço que permite gerir todos os C3 na rede e, na versão 2.0, foi reconstruída toda a camada do *backend*, desde o funcionamento do sistema até à comunicação com os servidores C3.

4.3.1 Desenho e Especificação da Arquitetura Geral

Como pretendido, a topologia de rede criada consegue ser centralizada e distribuída ao mesmo tempo. A Figura 4-1 mostra a topologia desenvolvida nesta nova versão, com uma rede de dispositivos C3 do tipo *peer-to-peer full mesh* (ou seja, com comunicação direta entre quaisquer dois dispositivos) geridos de forma centralizada pelo C3 Cloud Control.

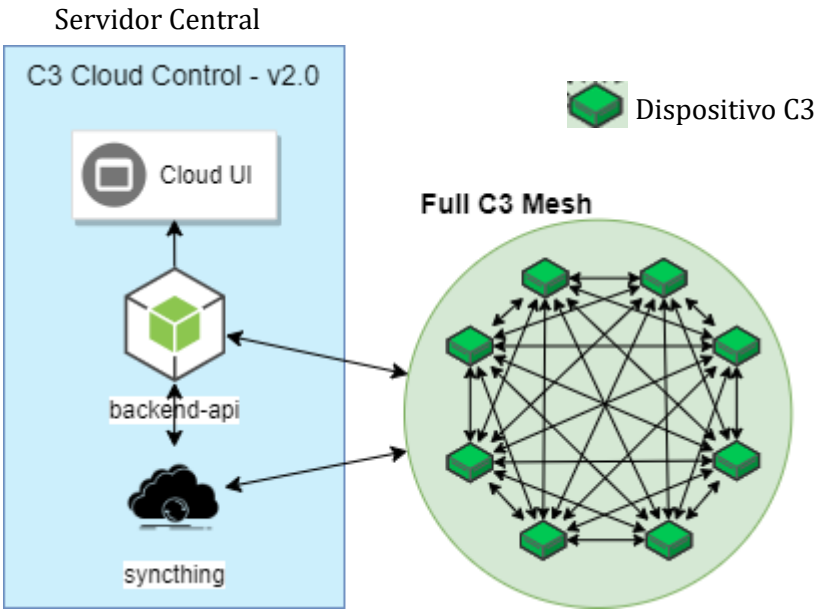


Figura 4-1 – Topologia de rede com respetivas conexões dos dispositivos C3 e C3 Cloud

Os C3 comunicam com o C3 Cloud de duas formas: através da ferramenta Syncthing e diretamente pelo *backend*. O Syncthing permite enviar os conteúdos para os C3 criando a rede *mesh* pretendida para partilha de conteúdos. Estes comunicam com o *backend* para indicar o seu estado de conexão (*online/offline*) e informações de sistema, tais como métricas relativas ao disco, CPU e RAM. Enviam ainda o estado de sincronização dos conteúdos e dados relativos a outras funcionalidades.

Dentro da rede *full-mesh*, os C3 comunicam entre si para sincronizar os conteúdos pretendidos, utilizando o Syncthing. No entanto, as conexões são estabelecidas apenas se os servidores tiverem conteúdos em comum para sincronizar, o que corresponde à característica Introdutora

do Syncthing (referida na seção 3.4). Com isto, a *full-mesh* é gerida pelo Syncthing Introdutor que se encontra no servidor central.

4.3.2 Desenho e Especificação da Arquitetura Interna

O C3 Cloud Control divide-se em quatro camadas principais (Figura 4-2):

- **Interface gráfica (Cloud UI)** – permite a interação dos utilizadores com a lógica do serviço;
- **REST API (node-api)** – contém as funcionalidades e lógica do C3 Cloud;
- **Camada de dados (cloudDB)** – armazena dados que os outros componentes necessitam. Estes dados estão numa base de dados em MongoDB [21]e, também, no sistema de ficheiros do servidor;
- **Syncthing** – ferramenta que sincroniza os conteúdos entre os C3 através de uma abordagem P2P.

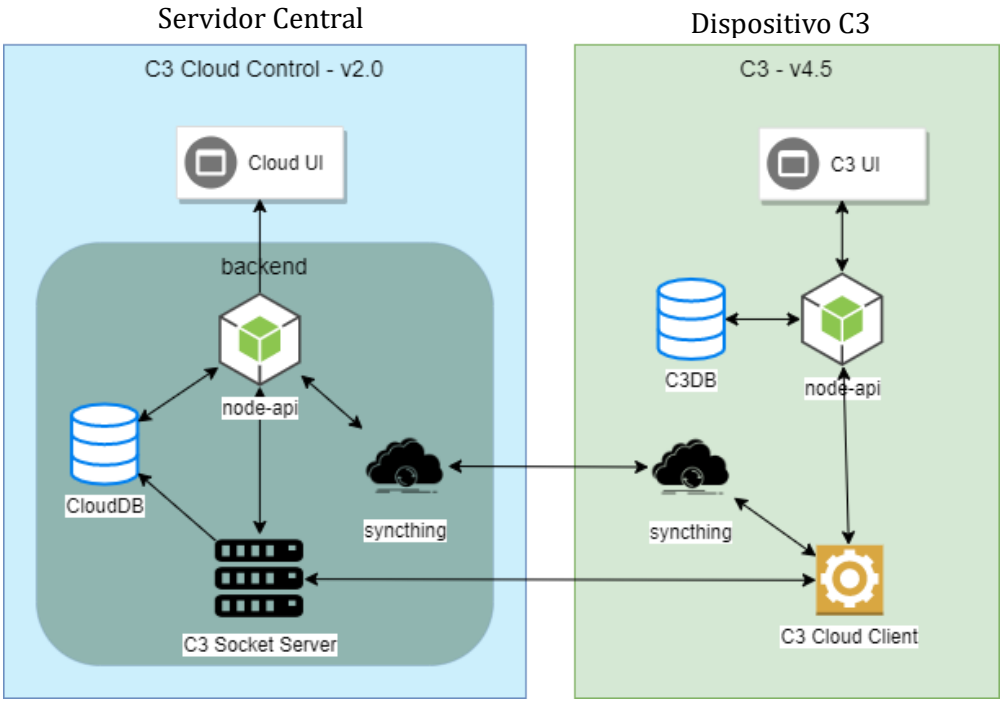


Figura 4-2 - Estrutura interna do C3 Cloud Control (servidor central) e de um dispositivo C3

Os C3 têm uma camada de software, chamada *C3 Cloud Client*, que se conecta ao *C3 Socket Server* no C3 Cloud Control, permitindo a comunicação bidirecional de dados entre os C3 e o C3 Cloud.

Na Tabela 4-2, encontra-se uma explicação das camadas de software do C3 Cloud Control.

Tabela 4-2 - Arquitetura interna de software no C3 Cloud Control

Componente	Descrição
Cloud UI	Permite a interação com o C3 Cloud Control. Consiste numa aplicação web que o utilizador pode aceder em qualquer browser.
Node API	Camada de software central: comunica com a interface de utilizador, o Syncthing, a base de dados e o servidor de *sockets* (*C3 Socket Server*). Expõe uma REST API que é implementado pela interface web, desencadeando operações complexas relativas a conteúdos e/ou dispositivos; interage com o sistema de ficheiros do C3 Cloud para gerir o conteúdo educativo.
C3 Socket Server	Contém uma *socket* para cada C3 conectado e permite comunicação bidirecional de dados.
Syncthing	Permite a sincronização do conteúdo entre os dispositivos C3 de forma distribuída.
MongoDB (CloudDB)	Permite o armazenamento de dados que são mostrados na aplicação. Contém informações dos utilizadores, dispositivos, conteúdo, entre outros modelos essenciais da lógica do C3 Cloud.

4.4 Ferramentas e Tecnologias Utilizadas

Nesta seção são apresentadas as tecnologias adotadas no desenvolvimento do projeto.

4.4.1 Frontend

Para o desenvolvimento da interface de utilizador (componente Cloud UI na Figura 4-2 e páginas apresentadas no Capítulo 6), foi utilizado o Angular 6 [22] uma vez que é uma *framework* bastante madura que permite desenvolver interfaces web com facilidade. Esta inclui um fluxo de dados contínuo que circula na interface, controladores e serviços. Além disto, o Angular é conhecido e utilizado por todos os elementos da equipa de desenvolvimento da entidade de acolhimento, sendo mais um motivo para a escolha desta *framework*.

Para a interface web foi utilizado um *template* da Metronic [23] que serviu de projeto base. Este inclui os estilos das páginas já com CSS (*Cascading Style Sheet*) feito e pronto a utilizar. O

código está organizado em módulos, componentes, e várias ferramentas que vêm por omissão, tais como *Bootstrap* [24] e angular-cli [25], entre outras.

Na Tabela 4-3 encontram-se os principais módulos utilizados no *frontend*.

Tabela 4-3 - Módulos mais relevantes no desenvolvimento do frontend

Módulo	**Descrição**
Angular Material [26]	Componentes do *Material Design* [27] para o angular.
Angular Bootstrap [28]	Componentes do *Bootstrap* [24] para o angular.
Chart.js [29]	Permite mostrar informação em gráficos com dados dinâmicos e em tempo real. Inclui oito tipos de gráficos apresentados no HTML5.
Codemirror [30]	Consiste num editor de texto implementado em JavaScript que é executado no browser. Especializa-se na edição de código e contém um conjunto de temas CSS que personaliza o editor de texto de acordo com a linguagem pretendida.
Google Maps [31]	Permite mostrar num mapa onde se encontram os C3.
Font Awesome [32]	Pacote de ícones para apresentar na interface.
Material Icons [33]	Pacote de ícones para apresentar na interface.
Socket.IO-client [34]	Ferramenta que permite a conexão ao servidor de sockets com Socket.IO.
Ngx-uploader [35]	Este módulo permite enviar ficheiros para o servidor utilizando o browser; incluindo através de operações *drag and drop*, mostrar o progresso dos uploads e fazer a gestão de uma fila de uploads.

4.4.2 Backend

A REST API no *backend* foi escrita em Node.js [36], um interpretador de JavaScript. Foi escolhida esta ferramenta pelos seguintes aspetos [37]:

- **JavaScript Fullstack** – a utilização da mesma linguagem de programação no *backend* e no *frontend* tornou o desenvolvimento web mais ágil e vantajoso;
- **Baixa curva de aprendizagem** – sendo utilizado o JavaScript no *frontend* e no *backend*, o programador consegue alternar facilmente entre os projetos, proporcionando maior produtividade e capacidade de partilhar o código. Por último, mas não menos importante, JavaScript é uma linguagem de aprendizagem fácil com grande apoio da comunidade;
- **Ecossistema Node.js** – o potencial do Node.js aumenta com a comunidade de utilizadores. O Node.js possui uma ferramenta que permite a gestão de código reutilizável e aberto pela comunidade para utilizar em projetos. Esta chama-se NPM (Node Package Manager) [38] e possibilita que o Node.js se adapte a várias situações com pacotes de código diferentes.
- **Dados consistentes –** a transmissão de dados entre os componentes do projeto é feita utilizando objetos JSON, tanto na base de dados como no *backend* e no *frontend*.

Na Tabela 4-4 encontram-se brevemente explicados os módulos mais relevantes do Node.js utilizados no desenvolvimento da solução C3 Cloud.

Tabela 4-4 - Módulos mais relevantes no desenvolvimento do backend

Módulo	**Descrição**
Express [39]	Encaminhamento de *endpoints* de forma eficaz e simplificada.
Passport [40]	*Middleware* relativo à autenticação e permissão de pedidos HTTP.
Express-session [41]	*Middleware* para manter sessões de utilizadores no *browser*.
Mongoose [42]	Permite a conexão a uma base de dados MongoDB, com validações, *hooks*, construção de *queries* e, o mais relevante, modelos de objetos a escrever na base de dados (relativos à logica de negócio).
Socket.IO server [43]	Com este módulo, o C3 Cloud pode receber e estabelecer conexões que vêm a partir dos C3. Estes são válidos através da sua licença e o seu ID do Syncthing.
Socket.IO client [44]	O socket.io disponibiliza *software* de cliente que permite a conexão ao servidor Socket.io presente no C3 Cloud.

Node-Syncthing [45]	Este módulo possibilita a comunicação e controlo da ferramenta Syncthing através da REST API que esta expõe.
Swagger [46]	Permite construir a documentação da API utilizando um documento com formato YAML (YAML Ain't Markup Language). Este documento é utilizado para criar um website com uma documentação da API de forma interativa, isto é, permite executar os pedidos HTTP em tempo real à API.

4.4.3 Controlo de Versões

Na Tabela 4-5 encontram-se as ferramentas e serviços utilizados para a gestão e versionamento do código desenvolvido.

Tabela 4-5 – Serviços e ferramentas utilizadas para controlo de versões de código

Ferramenta	Descrição
Git [47]	É um software de controlo de versões para código fonte. Permite a fácil consulta de modificações entre as várias versões, criar ramos de código.
Jira [48]	É um software comercial desenvolvido pela Atlassian [49]. Permite a monitorização de tarefas e acompanhamento de projetos, garantindo a gestão das suas atividades num único lugar.
Bitbucket [50]	É um serviço para armazenar projetos de software na Cloud. Tem integrações com o Jira e a gestão do código é feito em conjunto com o Git.
Vs Code [51]	É um editor de código desenvolvido pela Microsoft para Windows, Linux e macOS. Este inclui o IntelliSense, uma funcionalidade que permite uma melhor experiência no desenvolvimento (realce de sintaxe, completação automática de código, informações de funções e parâmetros, entre outras características). Tem o Git incorporado sendo possível gerir o código a partir do editor.

4.5 Integração do C3 Cloud com o C3

Para ativar um dispositivo C3, o administrador deste começa por inserir a licença adquirida. Esta licença é única por cada C3 e deve ser comprada ao contactar a equipa Critical Links. Ao inserir a licença, os serviços relativos à Cloud C3 iniciam e, com estes, o fluxo principal de comunicação entre o C3 e o C3 Cloud (Figura 4-3).

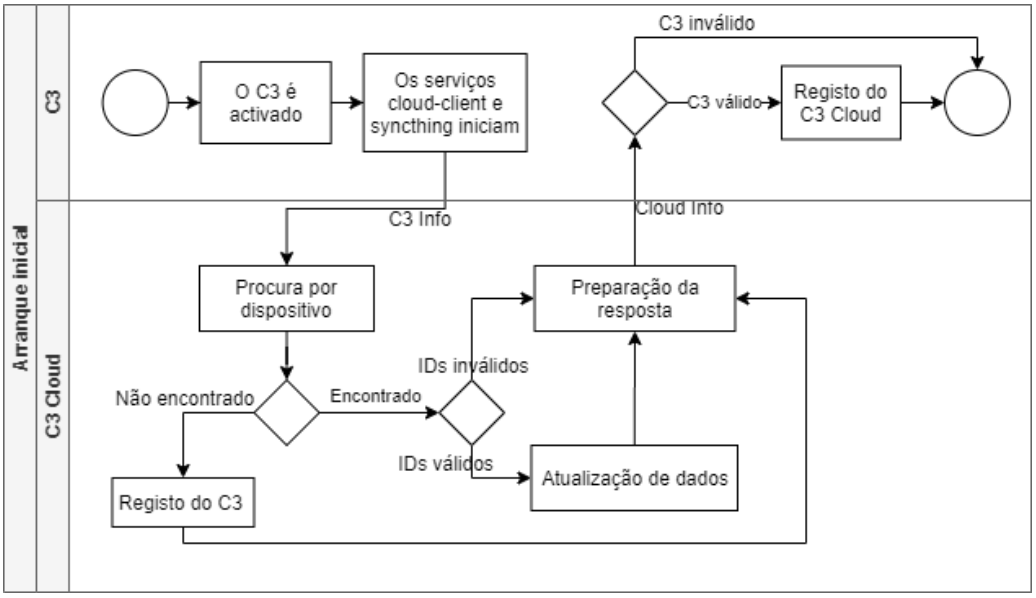


Figura 4-3 Diagrama de fluxo do arranque inicial de um C3

Através da ligação bidirecional de dados com o C3 SocketServer (Figura 4-2), o C3 envia dois identificadores: a sua licença e um identificador gerado pelo Syncthing durante o arranque [52]. O C3 Cloud recebe estes dados e trata-os. Caso nenhum C3 seja encontrado com a licença fornecida, o C3 é registado no C3 Cloud, tanto na base de dados como no Syncthing. Caso já esteja registado, são validadas as informações do C3 e atualizadas. Caso esteja tudo correto, o C3 Cloud devolve as informações necessárias para que haja comunicação através do Syncthing. Deste modo, é garantido que não haja tentativas bem-sucedidas de um dispositivo juntar-se à rede sem ter uma licença válida.

4.6 Problemas encontrados no C3 Cloud v2.0

Esta seção contém os principais problemas encontrados após o lançamento do C3 Cloud 2.0. O maior problema apareceu com o aumento do número de C3 conectados à rede Syncthing. Num cenário real controlado pela equipa de desenvolvimento, foi testada a migração de um C3 de cada vez da versão 1.0 para a versão 2.0 do C3 Cloud Control. O problema apareceu quando 300 dispositivos estavam *online* conectados ao mesmo C3 Cloud Control. A ferramenta de sincronização Syncthing não foi desenvolvida para soluções de grande escala e, consequentemente, cada dispositivo C3 ligado tinha um custo elevado de utilização de CPU e de RAM. O uso de uma topologia *full-mesh* não estava otimizado para ter vários dispositivos

conectados ao mesmo tempo entre si, o que levou à necessidade de realizar otimizações na topologia.

Para a correção do problema da topologia, foi implementado um mecanismo dinâmico de seleção de topologia, em que as variantes implementadas foram as seguintes:

- *Full mesh* – todos os C3 ligados entre si;
- *Meshes* parciais – os C3 estão ligados entre si, mas com limite máximo de conexões;
- *Hub & Spoke* – os C3 têm apenas uma ligação com o servidor central, o C3 Cloud Control.

A *full mesh* foi a primeira topologia implementada e, como referido nesta seção, não estava a funcionar bem com 300 dispositivos conectados ao C3 Cloud. A razão deste problema deve-se ao facto de a rede ser gerida através da característica Introdutora do Syncthing, explicado na seção 3.4. No C3 Cloud, o Syncthing faz a gestão de toda a rede automaticamente, o que acaba por aumentar a complexidade com o aumento de clientes C3. Se houver 300 dispositivos a querer o mesmo conteudo, o syncthing "apresenta" todos os dispositivos entre si e estes passam a comunicar entre si. O problema consiste

As *meshes* parciais foram implementadas para corrigir o problema da *full mesh*, dando origem ao C3 Cloud v3. Estas consistem na mesma abordagem, mas com um limite máximo de conexões a estabelecidas. Desta forma, o Syncthing consegue uma gestão da rede automática de forma melhorada, com menos carga e menos utilização de recursos do que com a abordagem *full mesh*. Esta topologia no C3 Cloud 3.0 contém um novo componente, o Syncthing Manager (Figura 4-4), que permite gerir as instâncias Syncthing que existem no C3 Cloud Control. Este lançar novas instâncias, atribuir dispositivos C3 a cada instância e equilibrar automaticamente o número de dispositivos que cada uma tem. O Syncthing Manager acede ao *backend* de forma a obter os dispositivos e conteúdos que estão registados.

Na topologia *Hub & Spoke*, os C3 têm apenas uma conexão com o servidor central. Desta forma, vai reduzir a carga nos C3 e aumentar no C3 Cloud. A sua implementação consiste em desligar a característica Introdutora do Syncthing que, deste modo, não tira proveito da rede distribuída dos C3.

Houve mais um problema relacionado com a ferramenta Syncthing e com o uso excessivo de CPU e RAM. Este foi detetado pela equipa de desenvolvimento do Syncthing e permitiu resolver problemas em cenários de larga escala (muitos dispositivos e conteúdos), como ocorre com o C3 Cloud Control. As correções foram lançadas no dia 17 de março de 2020, com o Syncthing 1.4.0 [53], e resultaram numa diminuição do tamanho da base de dados do Syncthing, com alterações nos modelos da base de dados. Foi corrigido um problema de uso de RAM quando muitos dispositivos estão conectados a pedir os mesmos ficheiros ao servidor central. Além disto, a interface do utilizador do Syncthing torna-se "pesada" na solução final, uma vez que não tem qualquer tipo de paginação e, em consequência, não consegue carregar toda a

informação em tempo real. No entanto, este problema não é preocupante porque a interface não precisa de ser utilizada dado que o Syncthing expõe uma REST API que permite controlá-lo através de uma aplicação externa.

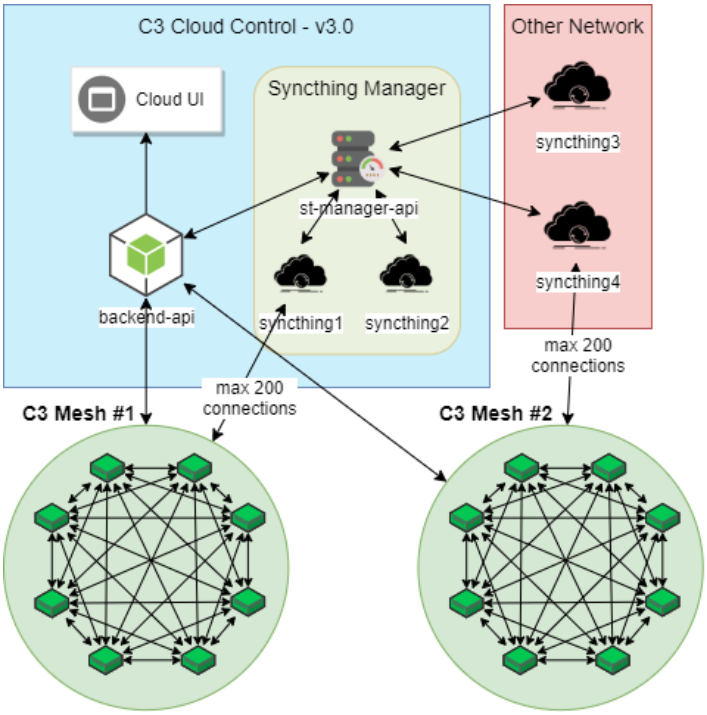


Figura 4-4 - Arquitetura do C3 Cloud versão 3.0 (simplificada)

5 Estudo Comparativo da Sincronização entre as Versões 1.0 e 2.0 do C3 Cloud Control

Este capítulo descreve o processo de avaliação efetuado à rede de sincronização do C3 Cloud 2.0, em termos de desempenho, em comparação com a versão inicial.

5.1 Cenários de Sincronização

Para cada uma das versões do C3 Cloud Control, versões 1.0 (abordagem cliente-servidor) e 2.0 (abordagem P2P), foi implementado um cenário de teste específico que pretende recriar exemplos de mundo real.

O primeiro cenário foi construído com a primeira solução do C3 Cloud e segue uma abordagem cliente-servidor, em que o servidor central envia conteúdos para outros nós usando o Rsync [12], uma ferramenta de sincronização de ficheiros. De forma a poupar largura de banda, a ferramenta tem uma opção para comprimir o conteúdo e também consegue enviar apenas o que ficou por sincronizar no caso do processo de sincronização ter parado repentinamente.

O segundo cenário foi implementado com a nova solução do C3 Cloud, que tem como base a abordagem P2P entres os nós C3, utilizando a ferramenta de sincronização Syncthing. Este consiste na solução desenvolvida pelo estagiário e mais detalhes encontram-se no Capítulo 3. Neste cenário, foi utilizado um servidor central, a versão 2.0 do C3 Cloud Control, para gerir os dispositivos e conteúdos da rede.

Em ambos os cenários, foram utilizados os mesmos conteúdos para testar a sincronização. Estes consistem em ficheiros compressíveis (imagens, vídeos, músicas e ficheiros PDF, entre outros) e conteúdos incompressíveis (ZIP, ISO e TAR). Foram definidas, para o efeito, as seguintes três variações de conteúdos:

- **Conteúdo 1**:
  - Contém 5 ficheiros incompressíveis que ocupam 300 MB;
  - Simula os conteúdos mais comuns;
- **Conteúdo 2**:
  - Composto por 10 000 ficheiros compressíveis que ocupam 300 MB;
  - Permite obter resultados sobre a eficácia de compressão da ferramenta de sincronização;
- **Conteúdo 3:**
  - Contém 2 ficheiros incompressíveis que ocupam 2,7 GB;
  - Simula ficheiros grandes.

A Figura 5-1 mostra os servidores de teste para ambos os cenários. Estes estão conectados a uma rede de velocidade de 1 Gbit/s, exceto o nó 3, que tem apenas 100 Mbit/s. Os nós 1 e 4

estão conectados à mesma rede, para testar se o Syncthing utiliza a melhor conexão para sincronizar conteúdos mais rapidamente.

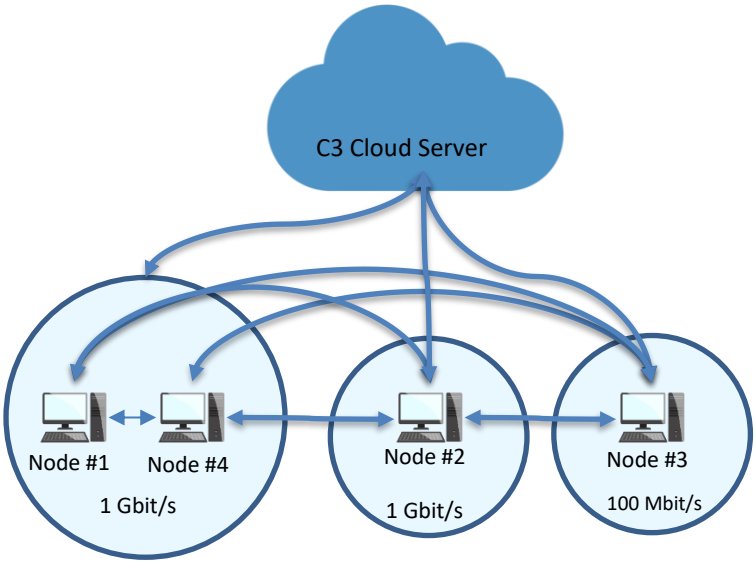


Figura 5-1 - Servidores de teste para ambos os cenários

Para ambos os cenários, as sincronizações foram feitas de duas maneiras distintas:

- **Ordem sequencial**: os nós e o conteúdo são sincronizados um após o outro. O segundo nó inicia apenas quando o primeiro terminou de sincronizar e assim sucessivamente. Para cada par dispositivo-conteúdo, foi registada a duração de cada sincronização e a quantidade de dados que o servidor central transferiu;
- **Tudo de uma vez**: os conteúdos 1 e 3 são enviados para todos os nós ao mesmo tempo. Foi registado a quantidade total de dados enviada pelo servidor central. O conteúdo 2 não entrou neste teste pelo motivo explicado na secção seguinte.

5.2 Resultados

Para o cenário 1, é esperado que o servidor central envie a totalidade do conteúdo uma vez que não tem servidores adicionais para ajudar. Na sincronização do tipo enviar “tudo de uma vez”, é esperado que o servidor central envie 3GB (300MB + 2700MB). Este valor, sendo destinado apenas a um nó, é depois multiplicado pelo número total de nós. Por este motivo é esperado o envio total de 12 GB.

Para o cenário 2, espera-se que o servidor central envie a totalidade do conteúdo, que totaliza 3 GB (300 + 2700), mas apenas para o primeiro nó. Para os seguintes, o servidor central (C3 Cloud Control) terá assistência de outros nós para concluir a sincronização. Além disso, espera-

se que os servidores que estão na mesma rede se conectem e troquem os conteúdos entre si mais rapidamente do que com outros em outras redes.

Para medir o tráfego gerado na rede, foi utilizada uma ferramenta chamada ntopng [54]. Esta disponibiliza uma interface gráfica e a Figura 5-2 mostra um *screenshot* de exemplo com o tráfego enviado/recebido pelos serviços que utilizam internet em tempo real. A ferramenta foi utilizada para analisar e registar a quantidade de dados que o servidor central envia e, também, a velocidade e a duração da transferência.

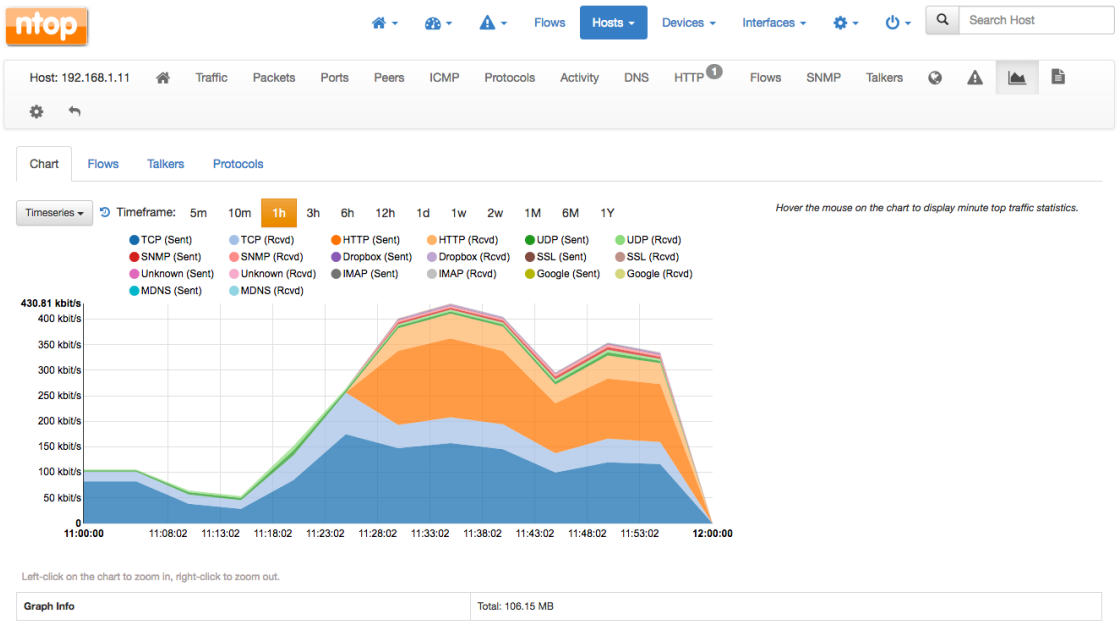


Figura 5-2 - Interface gráfica da ferramenta ntopng, com o tráfego enviado/recebido por protocolo e aplicação

A Tabela 5-1 mostra os resultados obtidos ao enviar todos os conteúdos de uma vez, conforme explicado na secção anterior. No cenário 1, o servidor central enviou um total de 12 GB, enquanto que no cenário 2 enviou 10,1 GB. Existe uma diferença de 1,9 GB de tráfego que se deve à ferramenta de sincronização utilizada, o Syncthing.

Tabela 5-1 - Resultados da sincronização de "tudo ao mesmo tempo"

Cenário	Total enviado pelo servidor central (GB)
1	12
2	10,1

A Tabela 5-2 mostra os resultados ao enviar os conteúdos com ordem sequencial, conforme explicado na secção anterior.

Tabela 5-2 - Resultados da sincronização com ordem sequencial no cenário 1 (C3 Cloud v1.0)

CONTEÚDO	SERVIDOR	DURAÇÃO (SEGUNDOS)	DÉBITO MÉDIO (MBIT/S)	DADOS ENVIADOS PELO SERVIDOR CENTRAL (MB)
#1	1	54	42	300
	2	56	42	300
	3	295	10	300
	4	54	41	300
#2	1	27	42	133
	2	29	43	133
	3	95	14	133
	4	27	42	133
#3	1	545	39	2700
	2	596	40	2700
	3	2100	8	2700
	4	587	38	2700

Neste cenário do tipo cliente-servidor, o conteúdo era sincronizado em todos os nós com uma velocidade média de 42 Mbit/s, exceto no nó 3 que estava conectado a um débito menor à Internet, conforme descrito anteriormente. Este só conseguia obter aproximadamente 10 Mbit/s de débito e demorou mais tempo para sincronizar, cerca de 75% a mais que os outros nós. Por este motivo, o nó 3 não foi considerado na seguinte discussão.

O conteúdo 1 foi sincronizado em 55 segundos (em média) e o servidor central enviou 300 MB, o que corresponde ao tamanho do conteúdo 1. Apesar dos conteúdos 1 e 2 terem o mesmo tamanho, o conteúdo 2 demorou metade do tempo para sincronizar, com duração média de 28 segundos. A razão é que o servidor central enviou apenas 133 MB uma vez que comprimiu os ficheiros. O conteúdo 3 possui 2700 MB e levou em média de 576 segundos (9 minutos e 36 segundos) para sincronizar completamente e o servidor central enviou o tamanho completo do arquivo para cada servidor.

A Tabela 5-3 contém os resultados do segundo cenário usando a ordem sequencial. Os resultados estão por ordem de sincronização, em que o servidor 1 foi o primeiro a receber na totalidade os conteúdos enviados pelo servidor central.

Tabela 5-3 - Resultados da sincronização com ordem sequencial no cenário 2

CONTEÚDO	SERVIDOR	DURAÇÃO (SEGUNDOS)	DÉBITO MÉDIO (MBIT/S)	DADOS ENVIADOS PELO SERVIDOR CENTRAL (MB)
#1	1	56	42	300
	2	52	42	280
	3	270	10	135
	4	14	29	21
#2	1	-	-	-
	2	-	-	-
	3	-	-	-
	4	-	-	-
#3	1	544	39	2700
	2	486	43	2200
	3	1774	5	1130
	4	73	30	290

Neste novo cenário do tipo P2P, o conteúdo 1 foi sincronizado em todos os nós com uma velocidade média de 42Mbit/s, exceto nos servidores 3 e 4. O nó 3 foi conectado a uma rede de débito inferior, conforme descrito anteriormente, e só atingiu em média 7 Mbit/s. O nó 4 está em torno de 30 Mbit/s, o que é menor do que no cenário anterior.

Examinando a coluna “dados enviados pelo servidor central”, podemos observar que os valores diminuem comparativamente ao nó anterior, o que significa que os nós já sincronizados são capazes de ajudar o servidor central a distribuir os conteúdos. O servidor central enviou o tamanho total dos conteúdos para servidor 1, sendo este o primeiro a receber os conteúdos. Para os seguintes servidores, o servidor central utilizou menos dados uma vez que os servidores da rede participaram na sincronização.

Observando as linhas do conteúdo 1, o servidor 3 levou 270 segundos para sincronizar, sendo mais rápido 25 segundos do que no cenário cliente-servidor. Em relação ao conteúdo 3, no cenário cliente-servidor, demorou 2100 segundos (cerca de 35 minutos), enquanto que no cenário do syncthing demorou 1774 segundos (cerca de 19 minutos). No cenário do syncthing, o nó 4 levou 14 segundos para sincronizar o conteúdo 1 e 73 segundos para sincronizar o conteúdo 3. Este nó está na mesma rede do nó #1, e os conteúdos foram obtidos deste em vez de recorrer ao nó central.

Os testes foram interrompidos para o conteúdo 2 uma vez que a sincronização estava a demorar mais de 5 minutos para sincronizar 300 MB de conteúdo. Não satisfeito com esses resultados, foram realizados mais testes usando discos HDD e SSD, o que permitiu obter respostas

relativamente à sincronização demorada. Os testes consistiram em sincronizar o conteúdo 2 em servidores com um discos diferentes (HDD e SDD). Os resultados foram os seguintes:

- HDD: 7 minutos (416 segundos);
- SSD: 50 segundos.

Em ambos os testes, a ferramenta Syncthing no servidor central enviou 165 MB em vez de 300 MB, o que significa que também conseguiu comprimir o conteúdo.

A Figura 5-3 mostra a duração e o tráfego gerado no servidor central ao sincronizar o conteúdo 1 paras os quatro nós nos cenários 1 e 2, por ordem sequencial. Embora os resultados sejam os mesmo das tabelas, é possível compará-los visualmente e tirar conclusões mais facilmente.

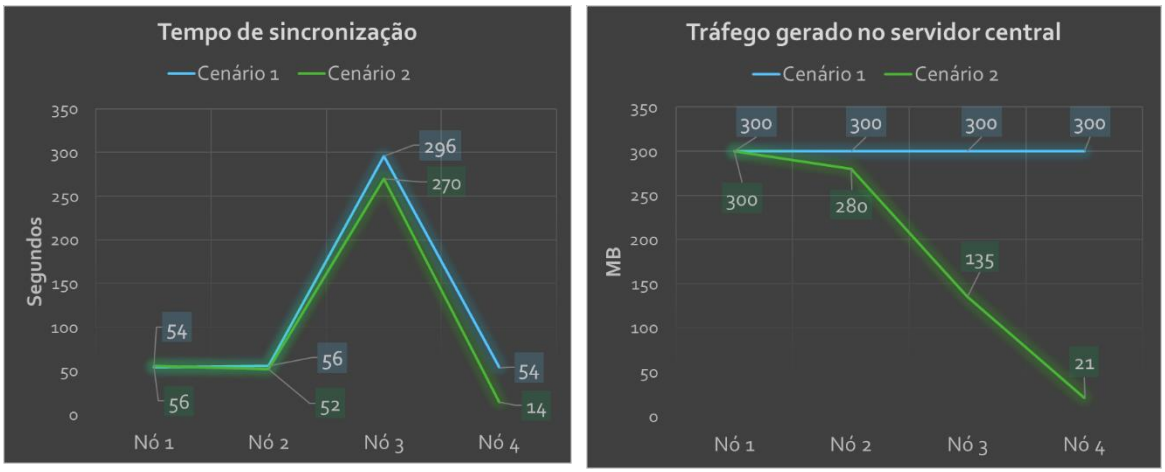


Figura 5-3 – Comparação da sincronização do conteúdo 1 por ordem sequencial nos cenários 1 e 2

Observamos que os servidores (nós) 1 e 2 têm a duração e o tráfego muito parecidos em ambos os cenários. O servidor 3 tem valores inferiores no cenário 2 e o servidor 4 obteve diferenças significativas na duração e no tráfego gerado no servidor central nos dois cenários. Este resultado no servidor 4 reflete o proveito da melhor conexão P2P com o servidor 1 uma vez que estão na mesma rede.

A Figura 5-4 apresenta os resultados da sincronização do conteúdo 3 por ordem sequencial nos cenários 1 e 2. Os resultados são parecidos com a figura anterior, mas o conteúdo 3 tem 2700 MB e, por este motivo, os valores obtidos são maiores. No entanto, as conclusões são as mesmas.

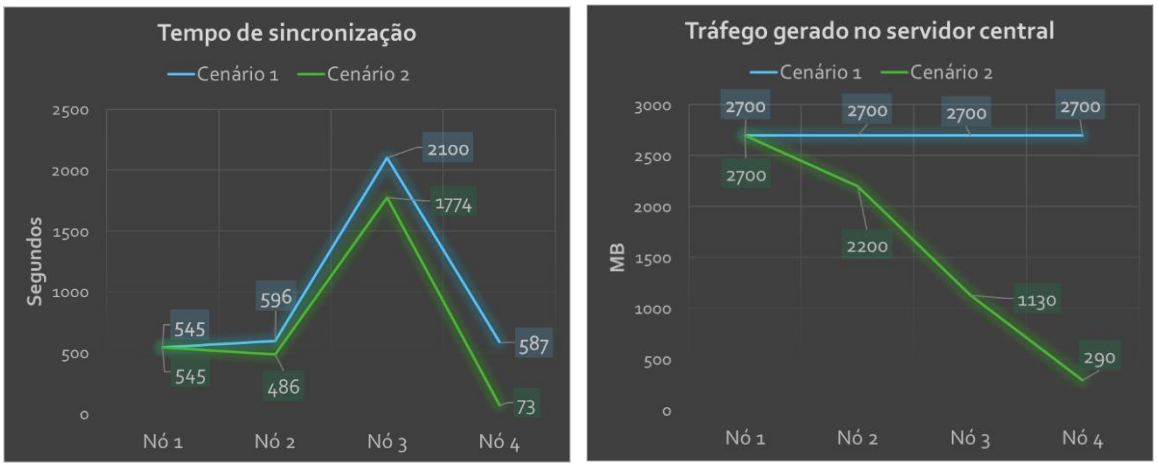


Figura 5-4 - Comparação da sincronização do conteúdo 2 por ordem sequencial nos cenários 1 e 2

5.3 Discussão dos resultados

Como esperado, no cenário 1 (cliente-servidor), que corresponde ao C3 Cloud v1, o servidor central enviou a quantidade total de dados, replicando os mesmos ficheiros para os quatro servidores. Ou seja, quanto mais dispositivos e conteúdos entram nesta rede, maior é a carga central e uso de largura de banda.

Por exemplo, com cenários mais próximos da realidade (mais de 1000 dispositivos e 100 GB de conteúdo para sincronizar), o servidor central ficará sobrecarregado. Certamente, o problema de escalabilidade poderia ser resolvido crescendo horizontalmente, isto é, utilizando mais servidores centrais para oferecer suporte a mais dispositivos, mas o custo financeiro seria maior.

Comparando a quantidade de dados que os servidores centrais enviaram nos cenários 1 e 2, podemos concluir que o cenário 1 enviou a totalidade dos dados, que é de 12 GB. Quanto ao cenário 2, o servidor central enviou 7,05 GB no total, o que resultou em menos 41% de tráfego. Como esses testes foram executados sequencialmente, a rede P2P tirou proveito das conexões permitindo que os servidores com o conteúdo começassem a sincronizar com os outros servidores igualmente interessados em obter o conteúdo.

O resultado mais importante pode ser encontrado no cenário ponto-a-ponto, com os respetivos resultados apresentados na Tabela 5-3, quando o conteúdo 3 é sincronizado no servidor 4. A coluna "dados enviados pelo servidor central " tem um valor de 290 MB, o que significa que o servidor central enviou apenas 290 MB de 2,7 GB.

A explicação para este resultado pode ser encontrada na Figura 5-5, onde é apresentado um *screenshot* da interface Syncthing no servidor 4. Como os nós 1 e 4 estão na mesma rede e o Syncthing privilegia conexões diretas, as sincronizações são muito mais rápidas e geram menos tráfego no servidor central. Podemos comprovar nesta figura que o nó 1 enviou 2,41 GB para o nó 4, o que corresponde a grande parte do tamanho do conteúdo 3 (2,41 GB + 0,29GB = 2,7GB).

Além disso, podemos igualmente observar que o nó 1 foi admitido pelo servidor central, o que é uma característica de segurança importante discutida na Secção 3.3.

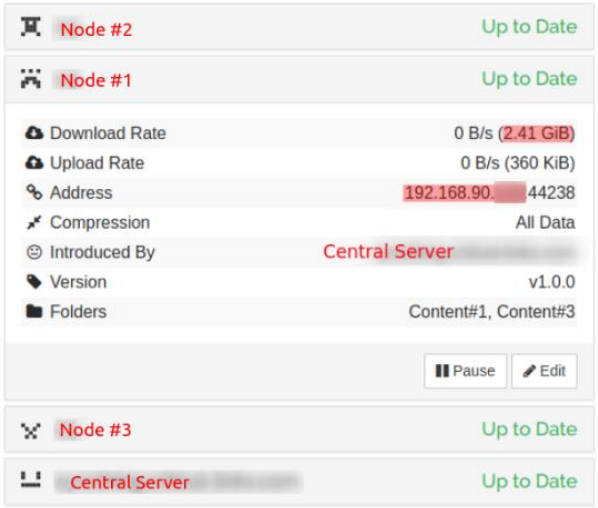


Figura 5-5 - Screenshot da interface do Syncthing, no Servidor 4

Comparando a duração e o tráfego para sincronizar, o conteúdo 3 nesses servidores (1 e 4), em vez de levar 540 segundos para sincronizar, demorou 73 segundos e, em vez do servidor central enviar o conteúdo completo, enviou 290 MB de um total de 2,7 GB. Portanto, a sincronização foi 86% mais rápida e economizou 89% da largura de banda.

Não são apresentados resultados para o conteúdo 2 do cenário 2 porque o Syncthing mostrou ter alguns problemas ao sincronizar um grande número de arquivos em discos HDD. Este é um problema conhecido dos programadores do Syncthing [55] e que pode ter acontecido durante os testes. Sabendo que o conteúdo 2 possui 10.000 ficheiros e que o disco HDD demorou 350 segundos e o disco SSD 50 segundos, então, os resultados sugerem que a causa da sincronização lenta é causada pela baixa velocidade de escrita nos discos HDD.

A maior limitação nos cenários reais é a rede de baixo débito. Por este motivo, a característica de compressão de dados é importante uma vez que queremos enviar o menor número de dados possível apesar de aumentar a utilização da CPU nas operações de compressão e descompressão. Tanto o Rsync como o Syncthing compactam os dados em trânsito [56] [57], o que significa que os dados são compactados no nível do pacote quando são enviados. Para comparar a quantidade de dados compactados durante a sincronização, o conteúdo 2 é o único com ficheiros compressíveis (imagens, vídeos e música, entre outros). Como possui 300 MB, podemos observar a quantidade de dados que o servidor central realmente enviou e tirar conclusões. O Rsync conseguiu uma taxa de compressão de 55%, passando de 300 MB para 133 MB, enquanto que o Syncthing compactou 45%, passando de 300 MB para 165 MB.

O resultado obtido ao sincronizar servidores na mesma rede foi 86% mais rápido e economizou 89% da largura de banda, em comparação com servidores fora da rede. Esta é a principal característica a ter em conta para economizar largura de banda. A Figura 5-6 mostra uma forma de tirar proveito desta característica, com um nó online, carregado de conteúdos, e quatro *offline*, que podem ou não ter conexão à internet, mas que estão conectados entre si na mesma rede. Embora estejam *offline*, estes servidores obtêm o conteúdo rapidamente porque o Syncthing tira proveito da conexão de rede local entre os servidores.

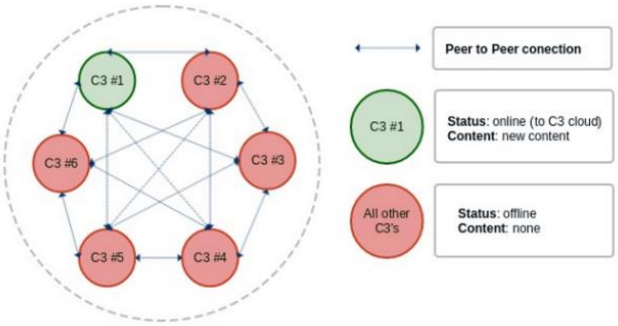


Figura 5-6 - Rede local de servidores C3 a sincronizar um conteúdo

As escolas equipadas com a solução C3, geralmente, têm mais de um servidor e frequentemente podem ter 100 nós. Se alguns desses nós forem pré-carregados com os conteúdos mais usuais para a língua pretendida, estes podem sincronizar os conteúdos com outros servidores na rede P2P com maior eficiência. A Figura 5-7 mostra mais um exemplo de um possível cenário real de duas escolas com três nós cada. Os servidores *online* podem comunicar com outros nós *online*, mas servidores *offline* apenas comunicam com os locais (servidores *offline* não têm conexão à internet, mas estão ligados à rede local).

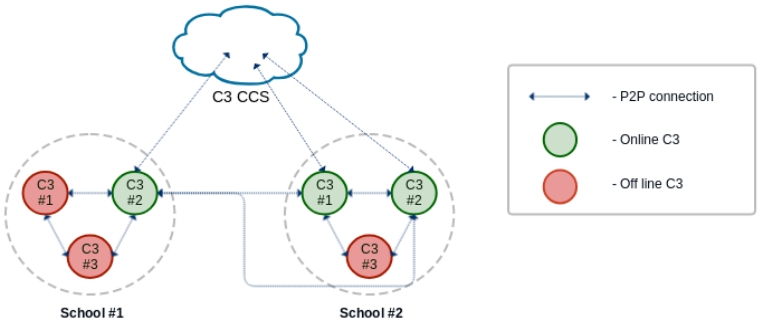


Figura 5-7 - Exemplo de duas redes locais com servidores C3 online e offline a sincronizar entre si

Os resultados dos testes comparativos mostraram que o C3 Cloud Control v2 atingiu os objetivos planeados. Ao comunicar com um nó na mesma rede (em LAN), a sincronização de conteúdos *peer-to-peer* aumentou a velocidade de transferência e poupou largura de banda.

6 RESULTADOS DA IMPLEMENTAÇÃO

Este capítulo apresenta o resultado do trabalho desenvolvimento ao longo do estágio descrito neste documento, o C3 Cloud Control 2.0, na perspetiva do utilizador do *frontend*. A interface web apresentada conforme o resultado da fase 2 do objetivo do estágio descrito neste relatório (seção 1.4).

6.1 Autenticação

A primeira página na interface do utilizador web é a de autenticação, como mostra a Figura 6-1. Nesta página, o utilizador deve inserir as suas credenciais (*username* e *password*) para aceder ao sistema. Geralmente, os utilizadores são professores ou donos de escolas.

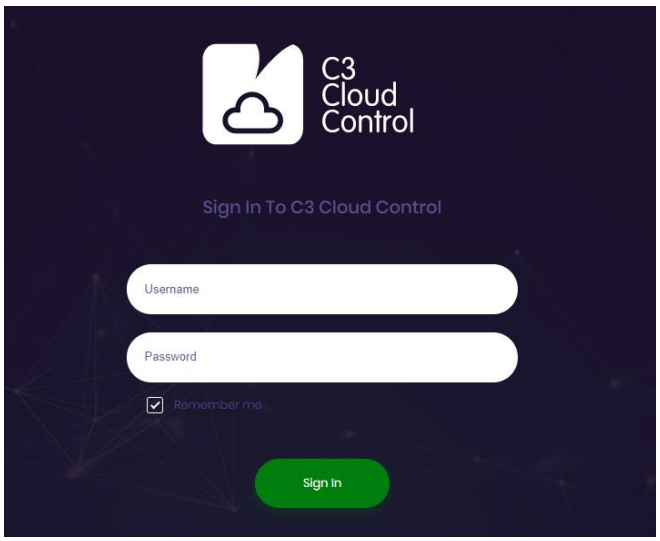


Figura 6-1 – Página de login do C3 Cloud

Apenas o administrador pode criar utilizadores e as respetivas credenciais de autenticação, o que significa que não é possível fazer o seu próprio registo.

6.2 Dashboard

Ao fazer o *login*, o utilizador é encaminhado para a página de entrada, também chamada de *dashboard*, ilustrada na Figura 6-2. Nesta, é possível ver dados gerais sobre dispositivos associados ao utilizador que entrou: quantos estão registados, configurados, *online* e a sincronizar. Para um acesso facilitado, mostra ainda um mapa com as localizações dos mesmos.

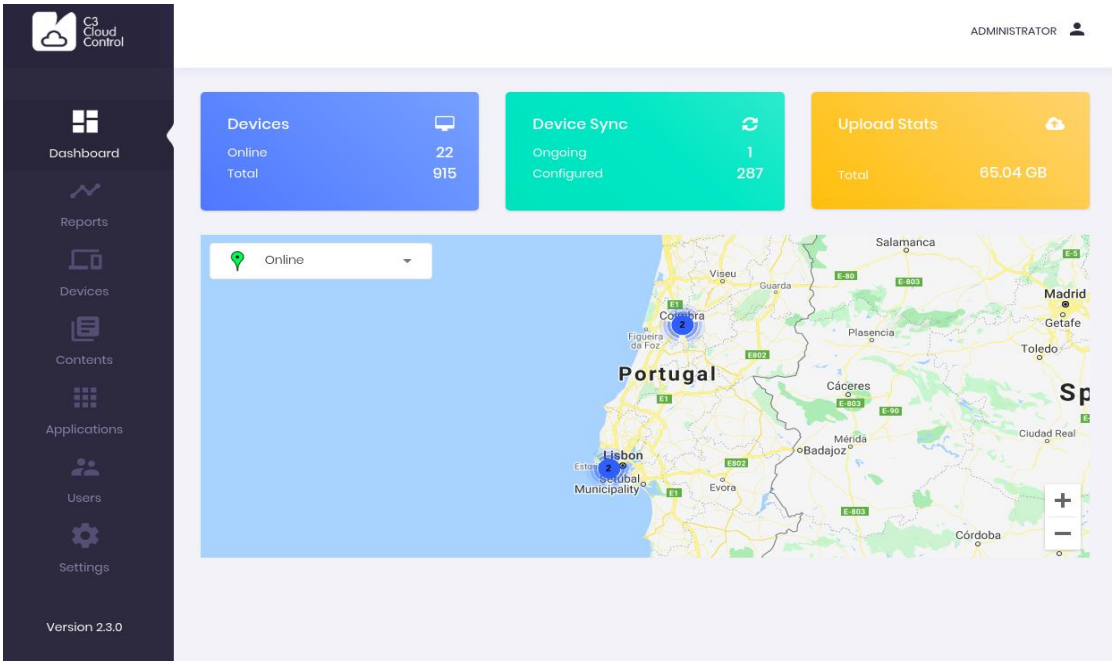


Figura 6-2 - Página de entrada do C3 Cloud

6.3 Conteúdos

Um dos pilares do C3 Cloud Control são os conteúdos. Neste contexto, um conteúdo consiste num conjunto de ficheiros e pastas. Para criar um conteúdo é obrigatório indicar um nome e um tipo, sendo as outras características opcionais. O conteúdo pode ter uma descrição, língua, *thumbnail* e o nível de visibilidade, como mostra a Figura 6-3.

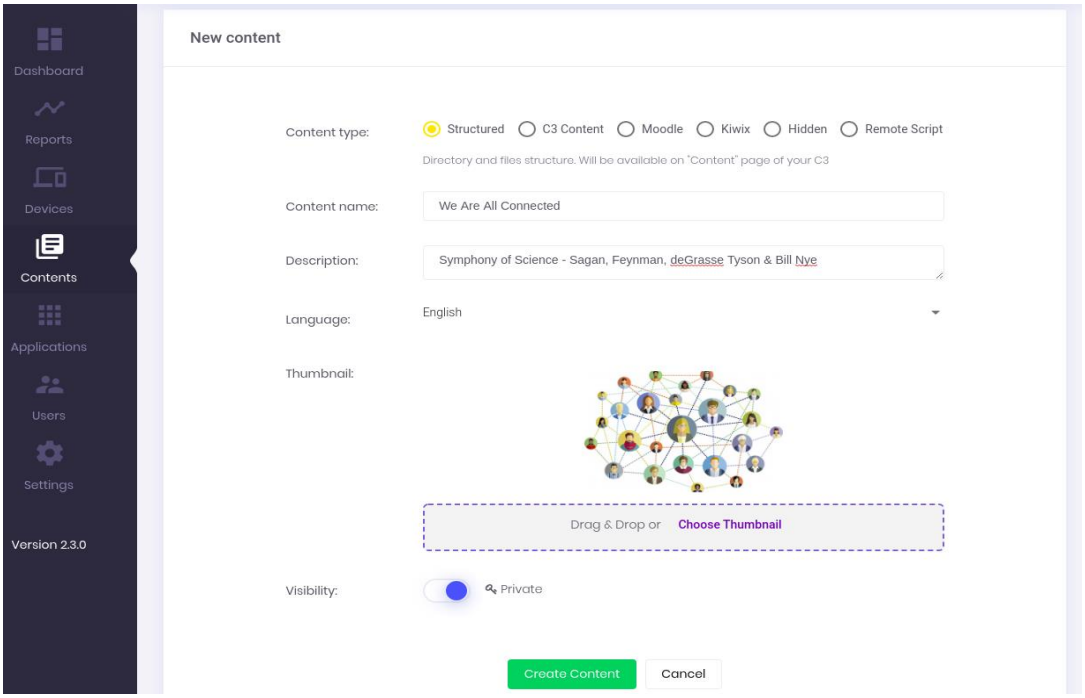


Figura 6-3 - Criação de um novo conteúdo

A Figura 6-4 mostra a listagem de conteúdos com paginação e uma procura pelo nome de conteúdo. É possível ordenar pelo nome, pelo tamanho, pelo número de subscrições e, ainda, pelas datas de criação e atualização.

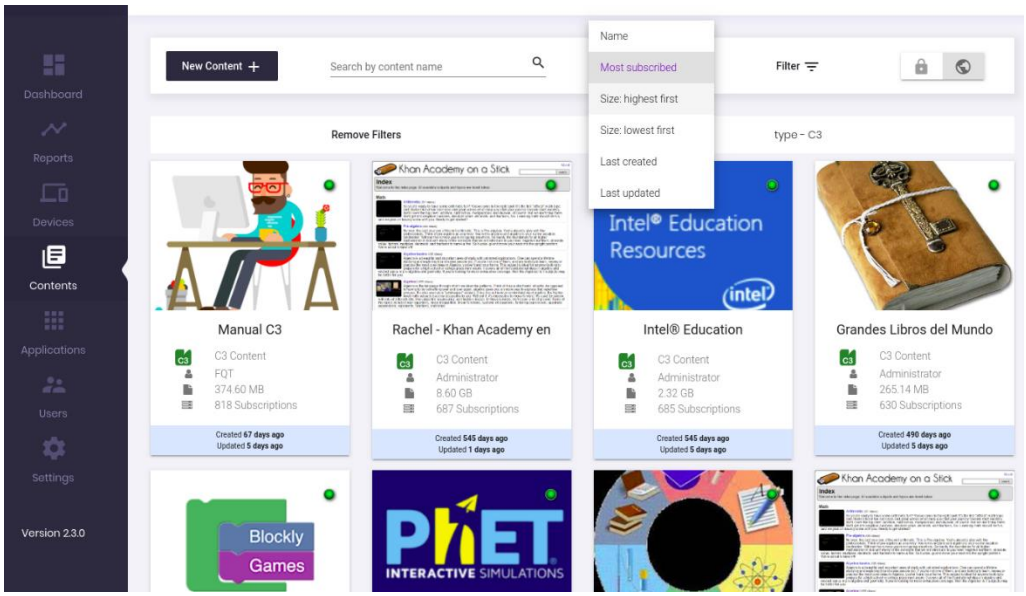


Figura 6-4 – Listagem de conteúdos

Por omissão, aparecem todos os conteúdos públicos. No entanto, é possível mostrar apenas os conteúdos criados pelo utilizador. Cada "cartão" representa um conteúdo com alguma informação geral sobre o mesmo. De forma ter uma procura mais apurada, é possível aplicar filtros de pesquisa, como mostra a Figura 6-5.

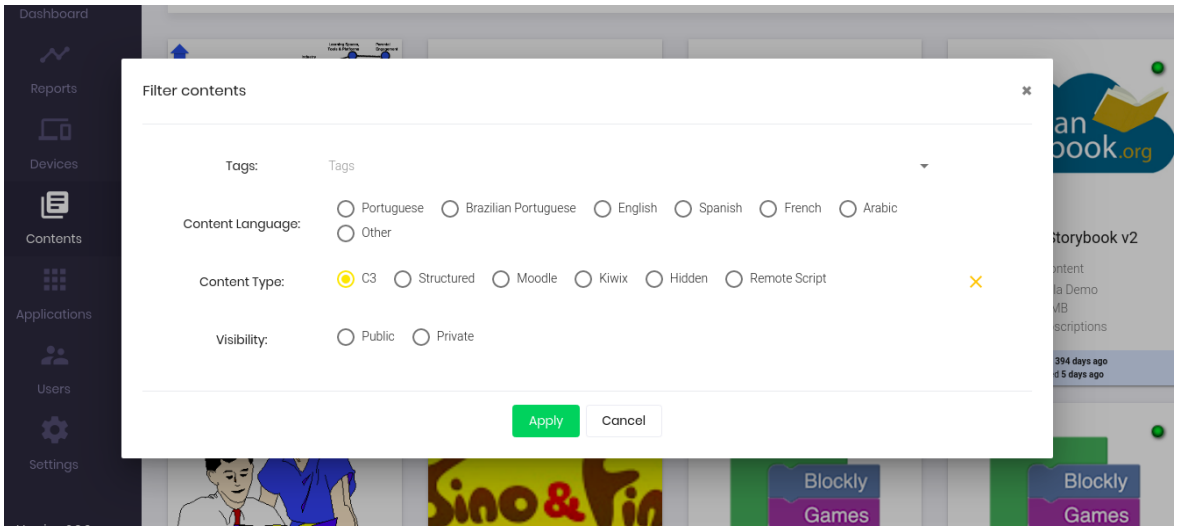


Figura 6-5 - Filtros de conteúdos

A Figura 6-6 mostra a página de detalhes de um conteúdo. Contém informação sobre o criador, a data de criação e atualização, o seu tamanho e a língua, entre outras informações. Nesta página pode editar-se os detalhes e a imagem de apresentação de conteúdo.

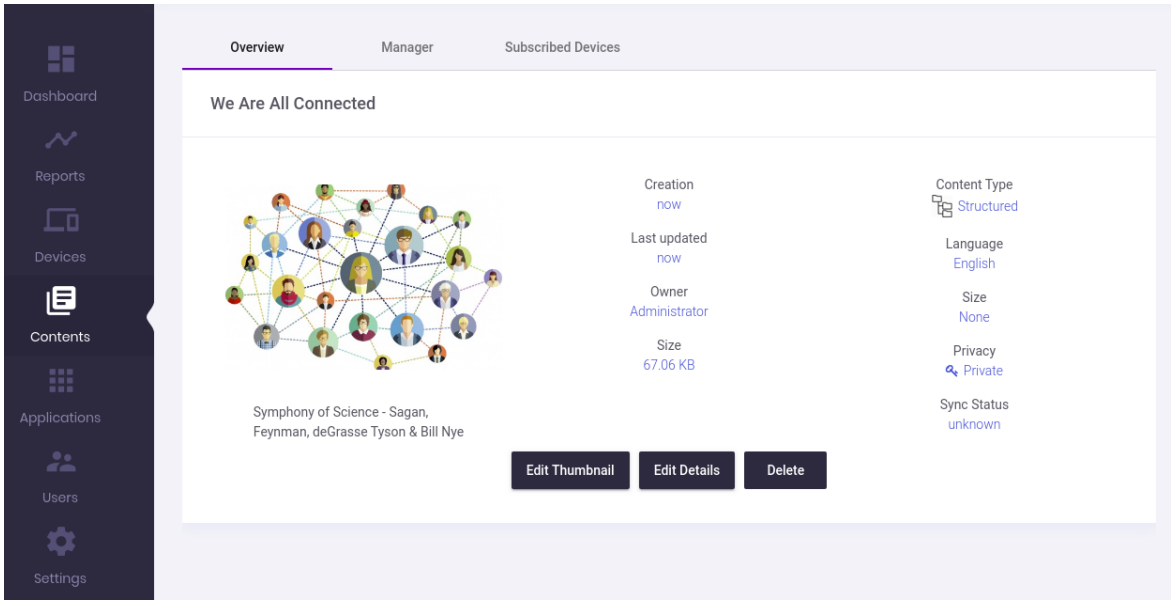


Figura 6-6 - Detalhes de um conteúdo

Ainda nos detalhes do conteúdo, pode gerir-se os ficheiros e os dispositivos que subscreveram a este conteúdo. A Figura 6-7 mostra como é feita a gestão de ficheiros de um conteúdo. Existe um botão que coloca o conteúdo "em manutenção". Neste modo, este está em pausa e não é sincronizado até sair da manutenção. Para adicionar/editar/apagar algum ficheiro, é preciso entrar em modo de manutenção. A figura mostra que este exemplo de conteúdo tem uma pasta "Biology" e ficheiro de vídeo descarregado do Youtube.

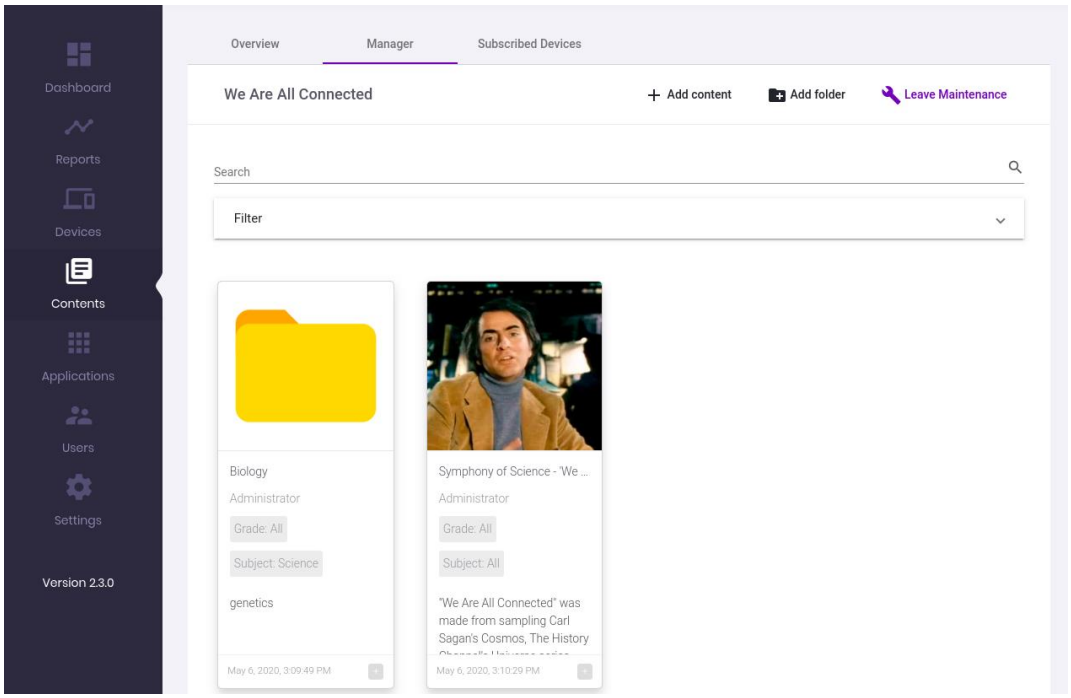


Figura 6-7 - Gestor de ficheiros

Na Figura 6-8, encontram-se as formas existentes de fazer *upload* de ficheiros para o conteúdo. Pode ser feito *upload* de ficheiros individualmente ou toda uma pasta. É possível descarregar diretamente na interface vídeos do Youtube e ainda criar um *link* para uma página online.

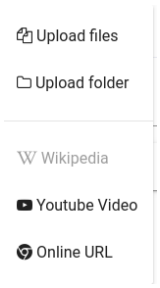


Figura 6-8 – Formas de fazer uploads para um conteúdo

Ainda nos detalhes de um conteúdo, como mostra a Figura 6-9, pode ver-se quais são os dispositivos que subscreveram este conteúdo, com o seu estado de sincronização. Nesta página é possível subscrever múltiplos dispositivos ao conteúdo, como mostra a Figura 6-10.

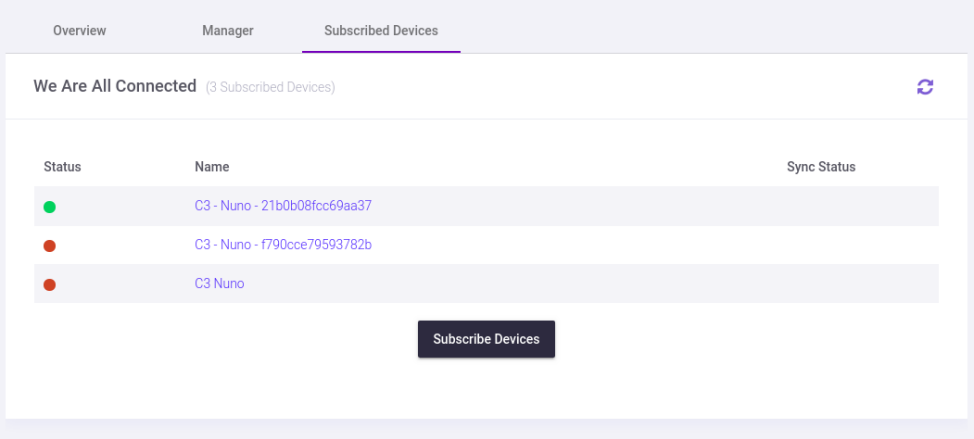


Figura 6-9 - Dispositivos subscritos a um conteúdo

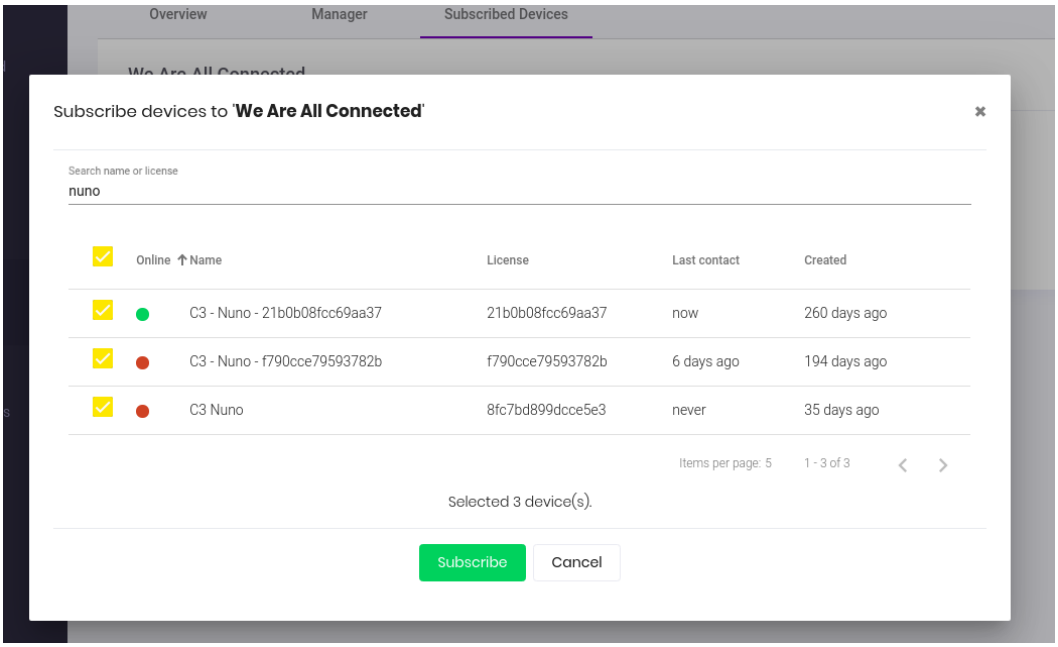


Figura 6-10 - Subscrever múltiplos dispositivos a um conteúdo

6.4 Dispositivos

Os dispositivos são o segundo pilar do C3 Cloud Control. Para registar um dispositivo é necessário ter a sua chave de licença, que é único por dispositivo e deve ser adquirida ao contactar a equipa da Critical Links. Ao inseri-la, o C3 Cloud Control verifica se é valida, como mostra a Figura 6-11 e, a partir daí, este está pronto a ser utilizado.

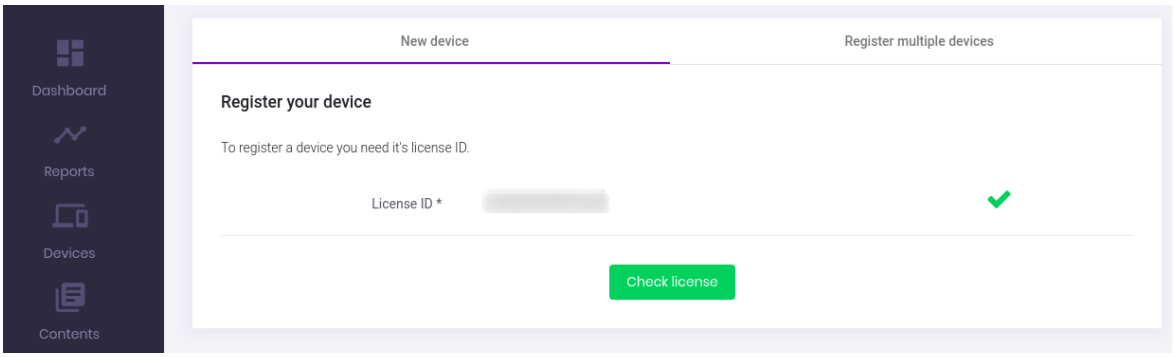


Figura 6-11 - Registar um dispositivo

Caso o utilizador queira registar vários dispositivos, este pode construir um ficheiro CSV com duas colunas: o nome e a licença do dispositivo. Opcionalmente, pode ser atribuído uma localização para mostrar no mapa, conteúdos para sincronizar e utilizadores para gerir os dispositivos a registar. A Figura 6-12 mostra um exemplo de registo de sete licenças validas.

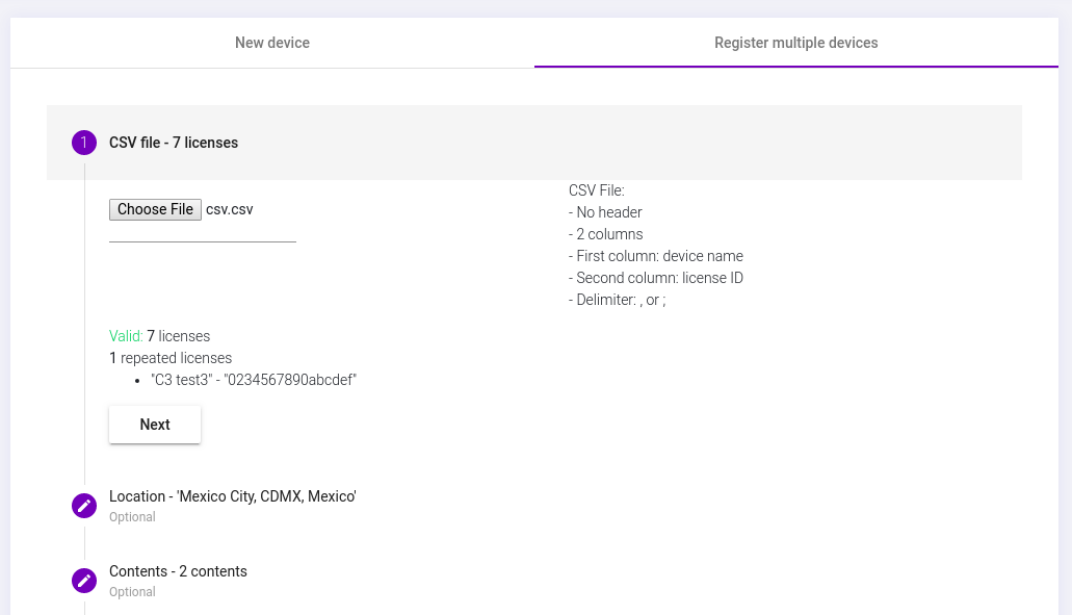


Figura 6-12 – Registo de vários dispositivos com um ficheiro CSV

Após o registo de dispositivos, estes surgem na página dos dispositivos, como mostra a Figura 6-13. Esta página também é a dos grupos, mas, por omissão, o grupo selecionado é o "*All*", em que aparecem os dispositivos a que o utilizador tem acesso. Os três "cartões" na parte superior apresentam alguma informação sobre os dispositivos, quantos dispositivos tem no total e quantos estão online, por exemplo. Os dispositivos aparecem numa listagem tabelada com o seu estado, licença, versão, última comunicação e estado de sincronização.

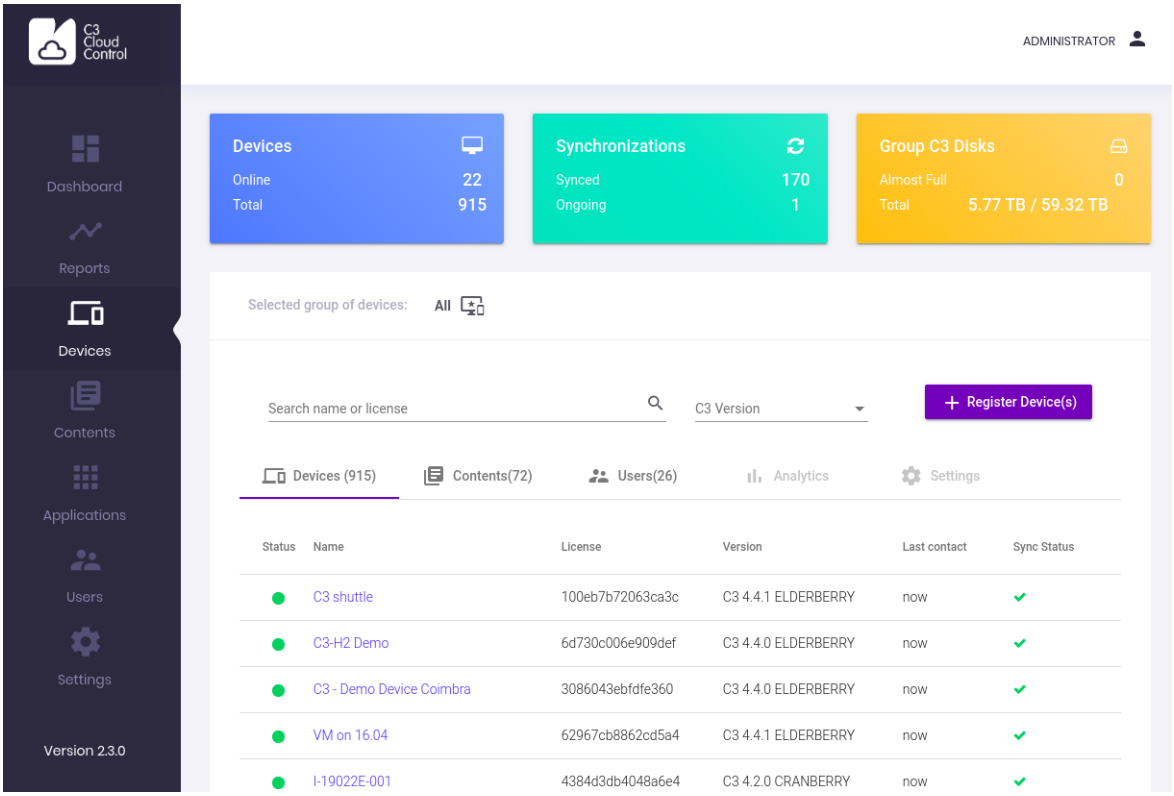


Figura 6-13 - Lista de todos os dispositivos

Para facilitar a procura de dispositivos numa determinada versão do C3, existe uma lista de seleção de versões no grupo selecionado. A Figura 6-14 mostra um exemplo da filtragem de dispositivos por versão. Também se pode pesquisar facilmente pelo nome ou licença do dispositivo.

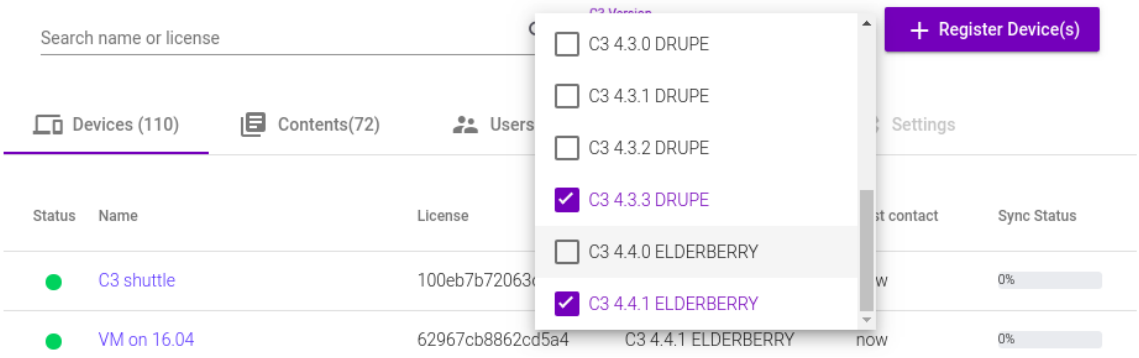


Figura 6-14 - Filtragem de dispositivos por versão

Ao clicar num dispositivo, é aberta uma página de detalhes do mesmo (Figura 6-15). Nesta, são apresentados mais detalhes dos dispositivos, com opções para editar alguns deles e

configurações de sincronização. Pode configurar-se um limite para a taxa de *upload* e de *download* das sincronizações e, ainda, horas do dia para que as sincronizações ocorram.

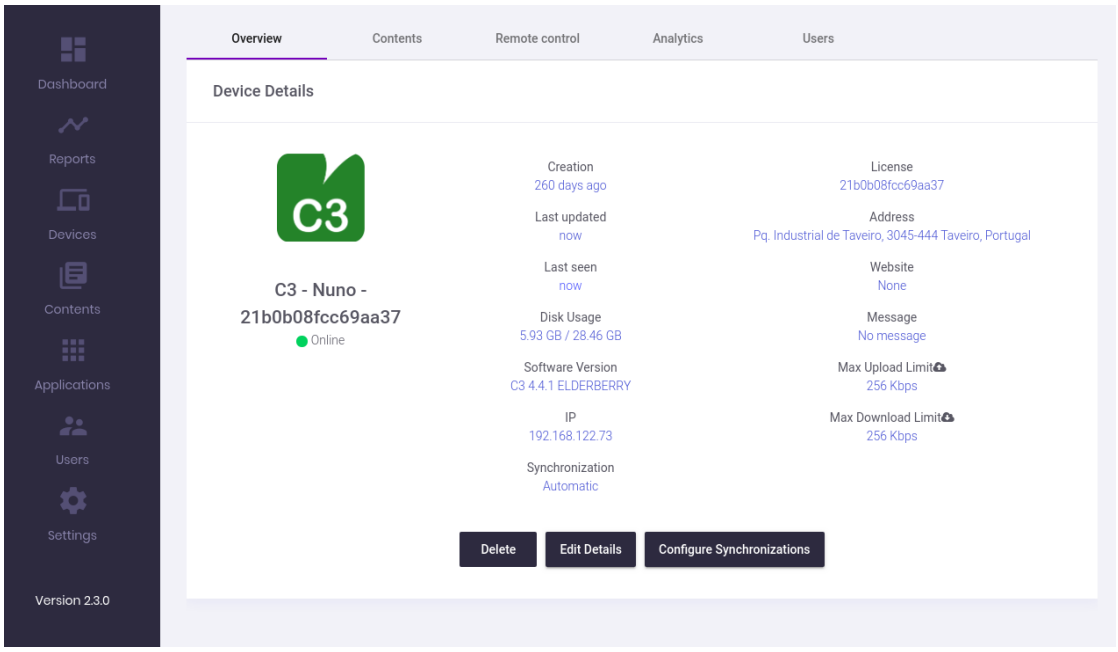


Figura 6-15 - Detalhes de um dispositivo

Ainda nos detalhes do dispositivo, pode verificar-se o progresso de *download* dos conteúdos subscritos, como mostra a Figura 6-16. Podem ser removidos ou adicionados conteúdos nesta página.

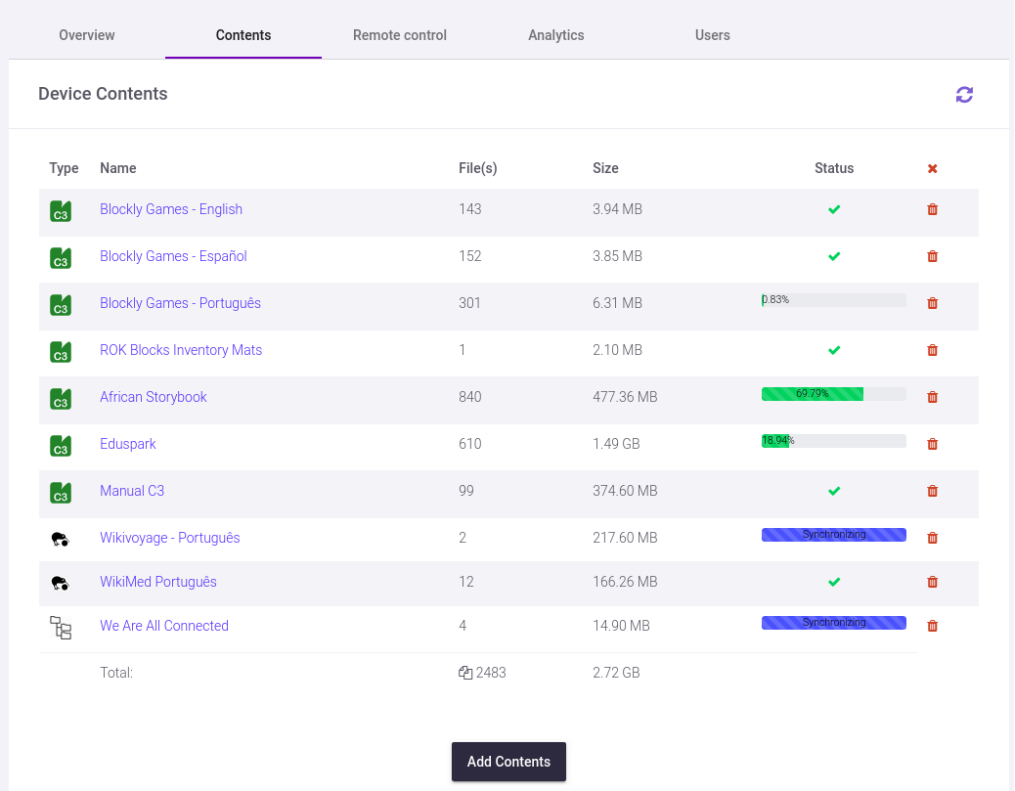


Figura 6-16 - Conteúdos e estado de sincronização de um dispositivo

O dispositivo envia dados sobre a abertura de conteúdos, indicando os que são mais utilizados pelos utilizadores. Na Figura 6-17 pode ver-se a análise gerada no C3 Cloud sobre a utilização dos conteúdos. Podemos saber qual o *browser* e sistema operativo mais utilizado e qual o utilizador e *role* de utilizador que abriu mais conteúdos. Podemos aplicar filtros de conteúdos e de datas. Os dados aparecem apenas quando o cursor passa por cima do gráfico.

De forma a facultar assistência remota, apenas o administrador pode abrir e fechar um túnel SSH no dispositivo. Este fica acessível através dum servidor controlado pela empresa. A Figura 6-18 mostra a abertura de um túnel SSH para acesso remoto. Além de abrir um túnel SSH, podem ser enviados pedidos ao dispositivo através de controlo remoto avançado, como mostra a Figura 6-19. O dispositivo tem uma lista de ações que pode executar e esta lista encontra-se nesta seção de controlo remoto avançado. Ao selecionar uma ação, aparece o seu nome, descrição, e alguns parâmetros (se necessário). Ao enviar a ação, o dispositivo executa-a e responde com a resposta assim que terminar.

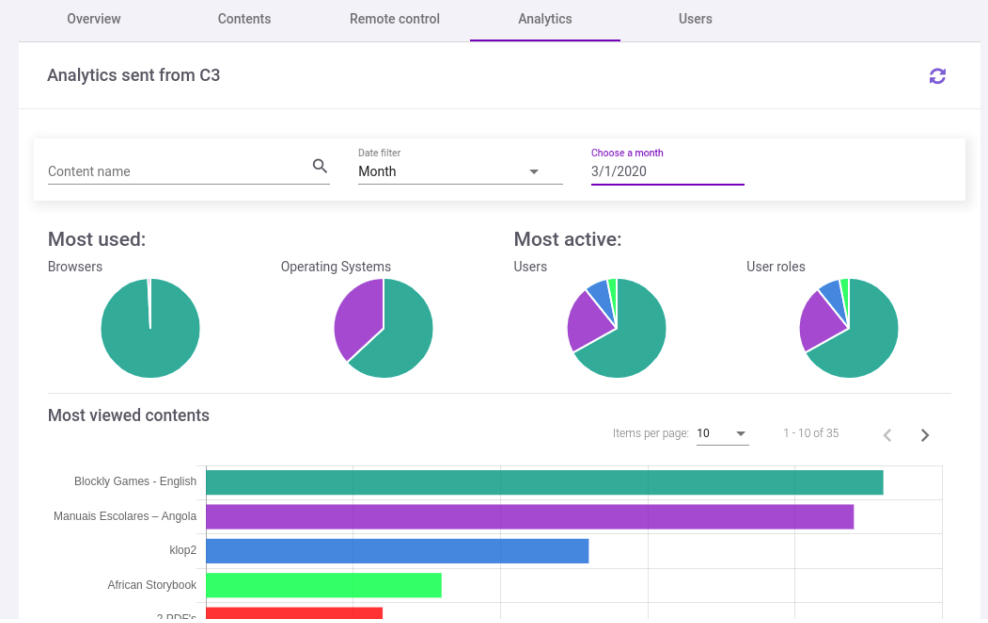


Figura 6-17 - Análise de dados sobre o uso de conteúdos

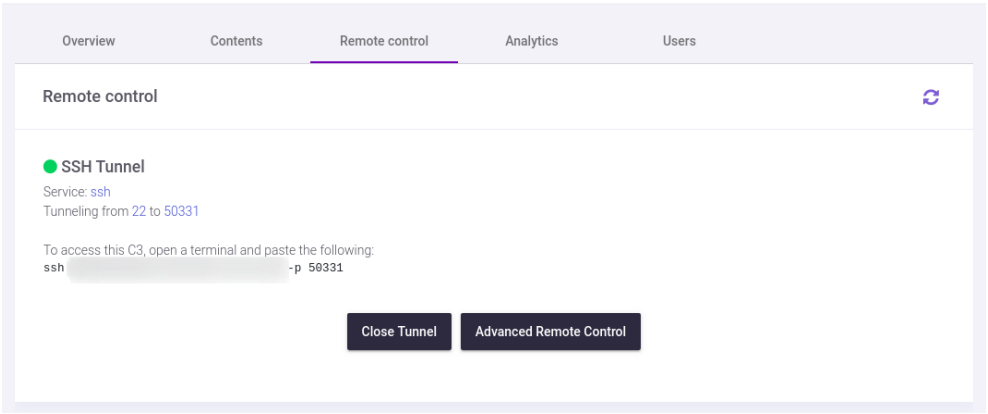


Figura 6-18 – Abertura de um túnel SSH para controlo remoto de um dispositivo

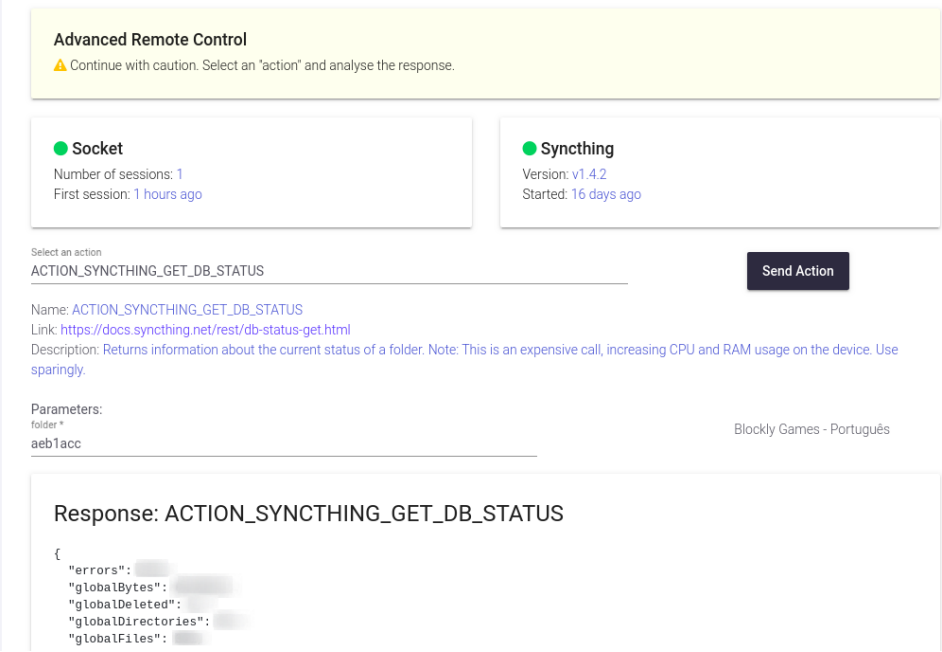


Figura 6-19 - Controlo remoto avançado de um dispositivo

6.5 Grupos de dispositivos

Os grupos servem para organizar e controlar vários dispositivos ao mesmo tempo. Cada utilizador tem os seus grupos e pode criar, editar e apagar grupos de dispositivos. A Figura 6-20 apresenta um exemplo. Estes estão estruturados em forma de árvore para manter a funcionalidade que existia na versão 1.0.

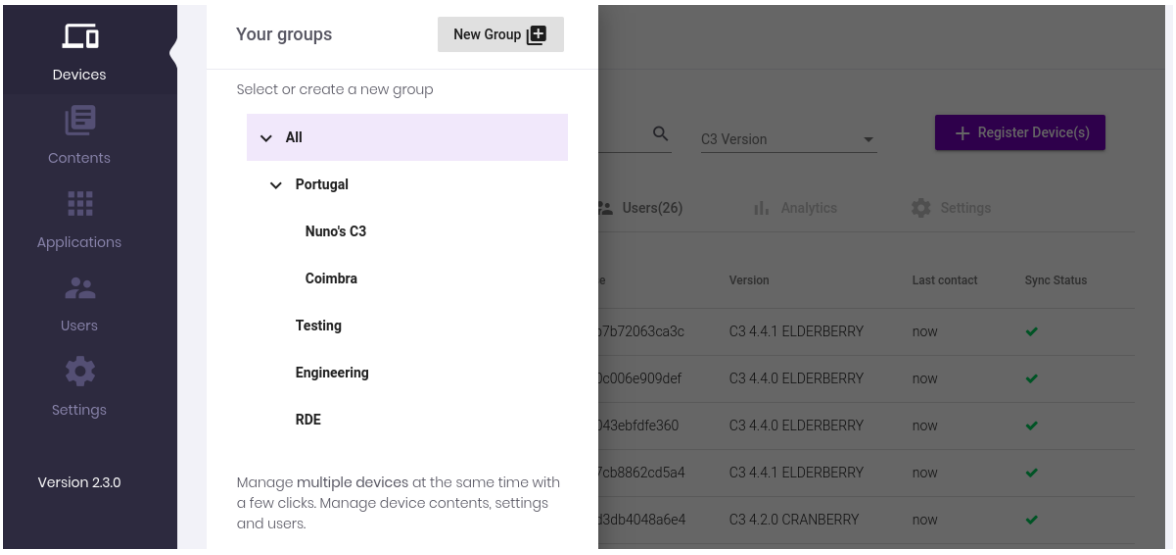


Figura 6-20 – Organização de dispositivos por grupos

Ao clicar em "*New Group*", aparece um formulário a pedir o nome do grupo, uma descrição e, uma vez que tem estrutura de árvore, qual o grupo onde deve ser inserido, como mostra a Figura 6-21.

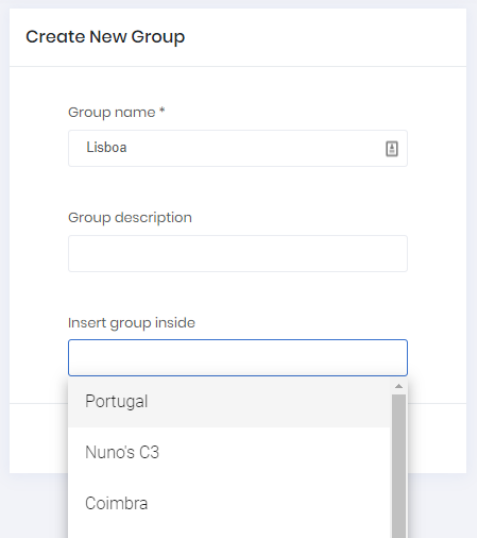


Figura 6-21 - Criação de um novo grupo

No exemplo escolhido, ao selecionar o grupo "Coimbra", são mostrados os cinco dispositivos que este tem, como se pode verificar na Figura 6-22. Os dispositivos deste grupo têm 30 conteúdos no total, como mostra a Figura 6-23, mas nem todos têm todos os conteúdos, como se pode verificar no número ao lado da lupa na figura. Um conteúdo pode ser removido dos dispositivos do grupo ao clicar na barra vermelha do lado direito da figura. Ao clicar na lupa, abre-se uma janela de diálogo com detalhes dos dispositivos que têm o conteúdo subscrito, como mostra a Figura 6-24. Nesta janela, pode cancelar-se a subscrição de dispositivos ao clicar no botão com um caixote do lixo vermelho.

Para adicionar conteúdos aos dispositivos do grupo pode clicar-se em "*Add contents to* Coimbra". Uma janela de diálogo como o da Figura 6-25 aparece e podem ser selecionados os conteúdos a adicionar ao grupo. Para um procura facilitada do conteúdo a adicionar, tem uma caixa de texto de procura e filtros pela linguagem ou tipo de conteúdo.

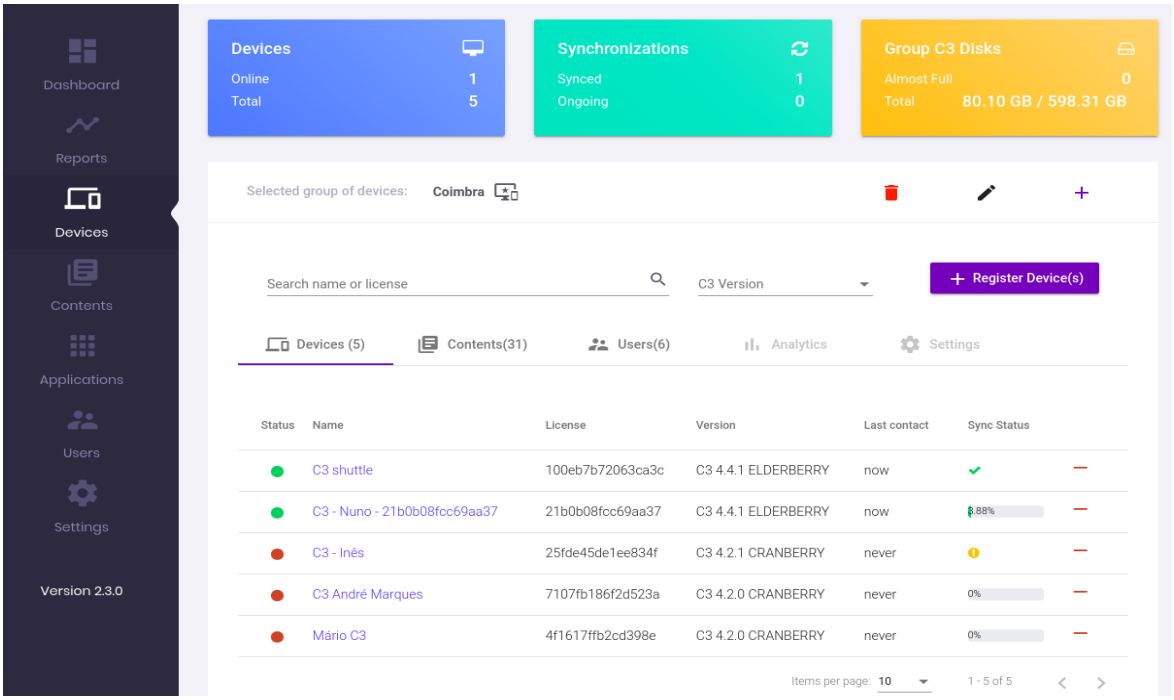


Figura 6-22 - Grupo de dispositivos

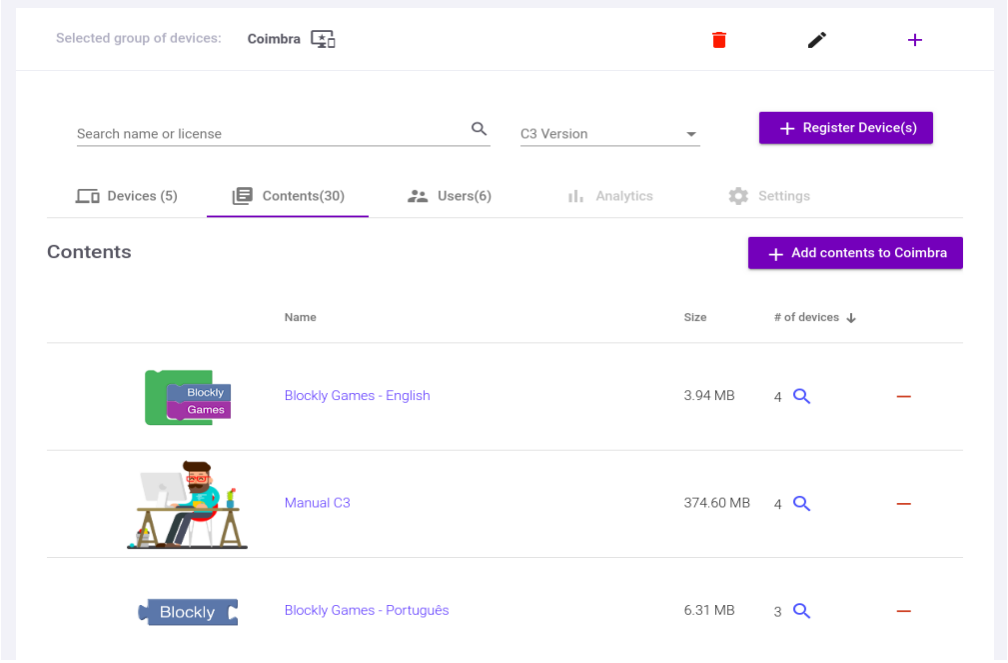


Figura 6-23 - Conteúdos de um grupo de dispositivos

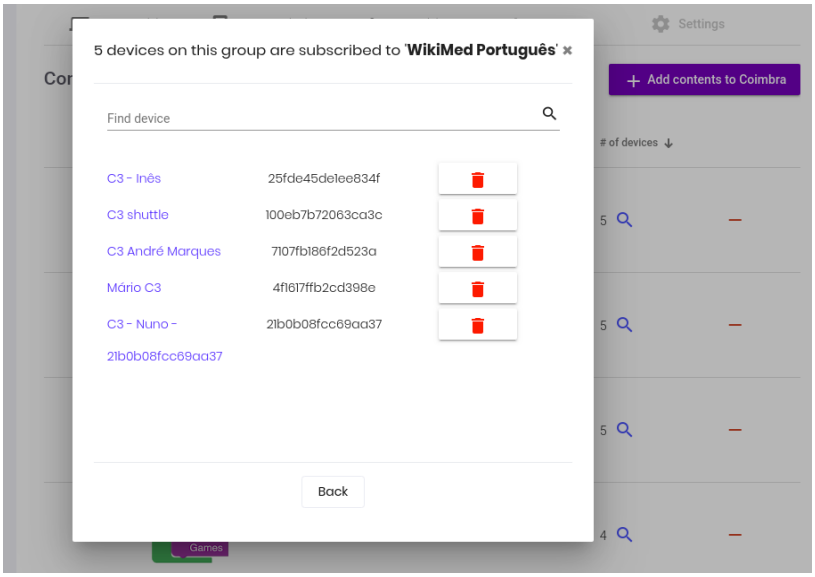


Figura 6-24 - Cancelar subscrição de conteúdos num grupo de dispositivos

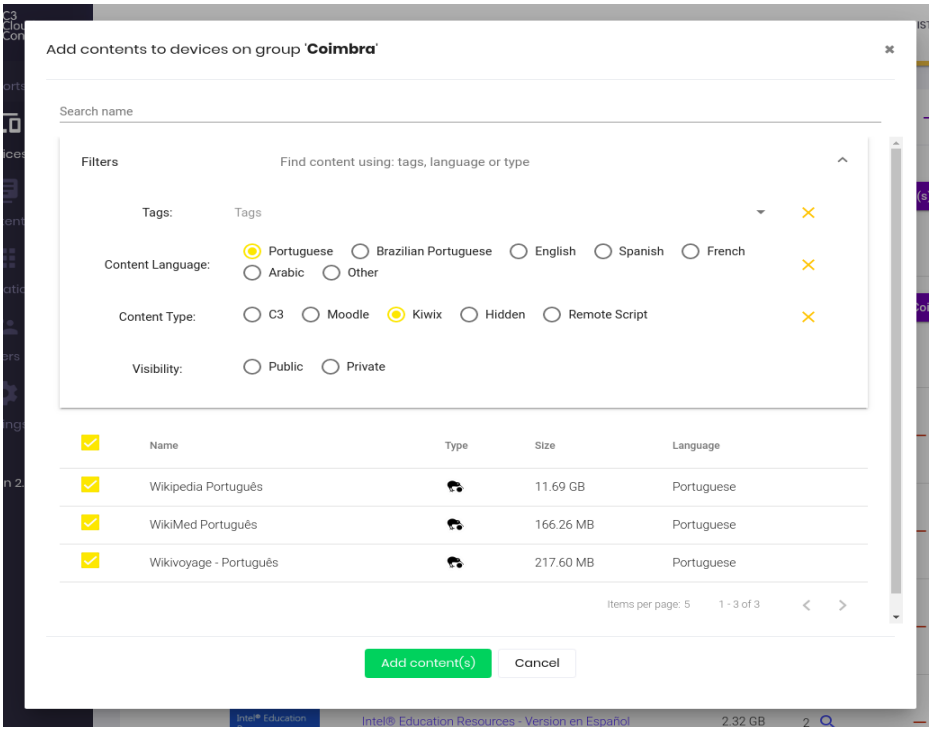


Figura 6-25 - Subscrever conteúdos para os dispositivos de um grupo

6.6 Relatórios

Foi criada uma seção de relatórios para ter uma vista geral das sincronizações que estão a ocorrer (Figura 6-26).

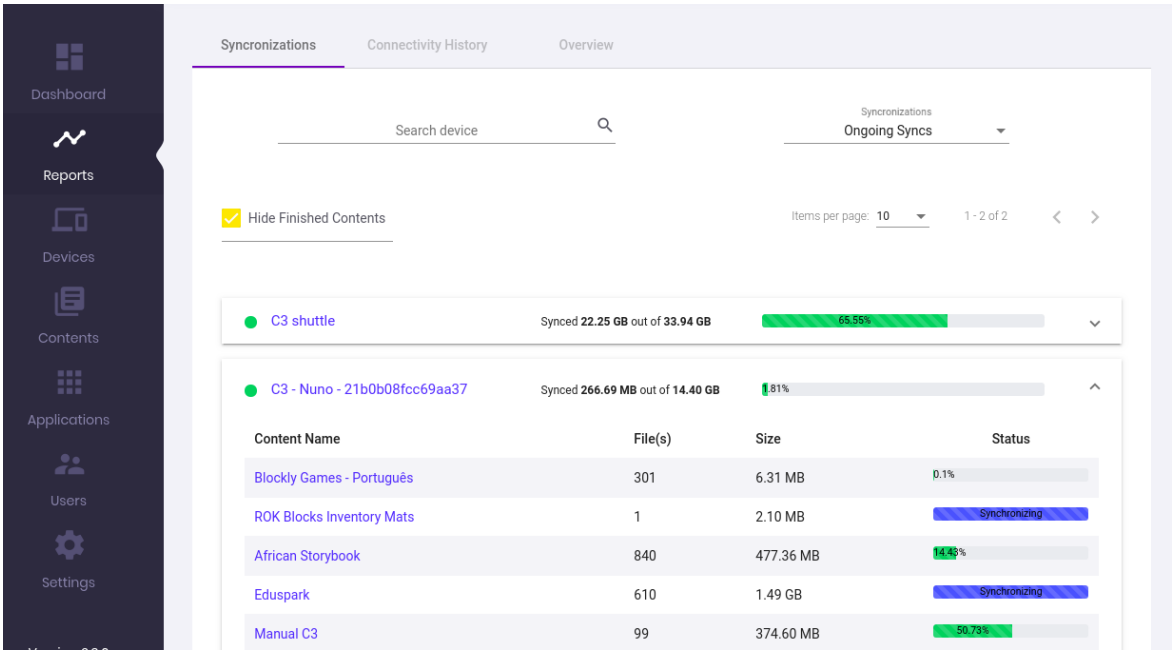


Figura 6-26 - Sincronização dos dispositivos

6.7 Definições

Foi criado uma página para ter controlo e uma forma de visualizar as instâncias Syncthing, apenas existente no C3 Cloud v3.0. A Figura 6-27 mostra essa página com apenas algumas opções de ficar em pausa e resumir as sincronizações dos dispositivos. Por cada instância, pode saber-se quantos dispositivos tem, quantos estão conectados e de quantos *bytes* fez *upload*.

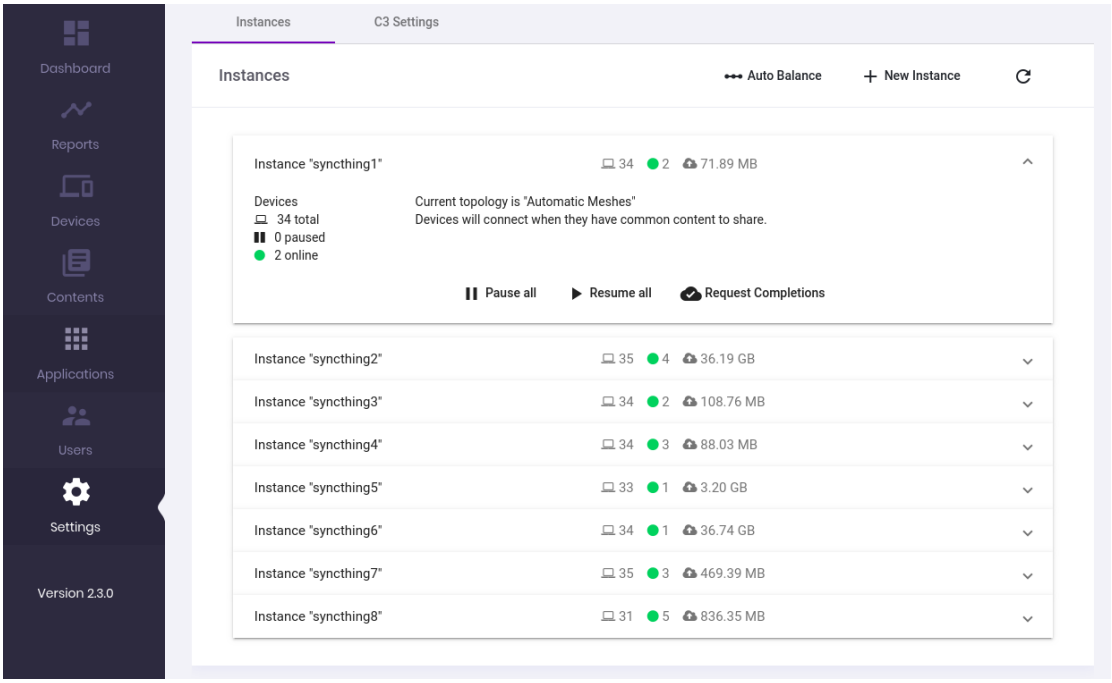


Figura 6-27 - Definições e visualização das instâncias Syncthing

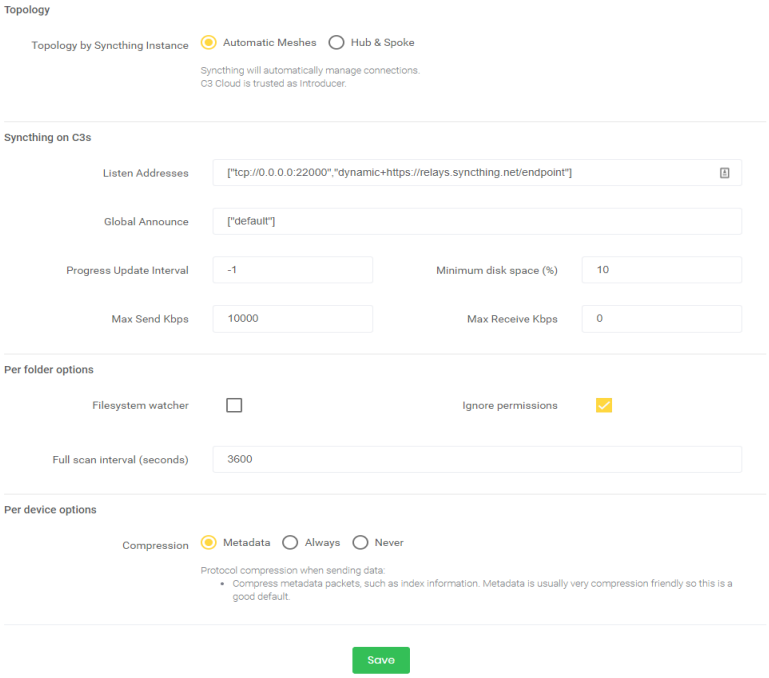


Figura 6-28 - Configurações do Syncthing para todos os dispositivos conectados ao C3 Cloud

7 CONCLUSÕES E TRABALHO FUTURO

Neste capítulo final faz-se uma apreciação sobre o trabalho desenvolvido compõem-se algumas considerações sobre possíveis melhorias no serviço C3 Cloud Control que, eventualmente, possam vir a ser realizadas após o final do estágio. Por último, é feita uma apreciação pessoal do estágio realizado.

7.1 Conclusões

O projeto atribuído ao estagiário consistiu em substituir o serviço denominado C3 Cloud Control da empresa Critical Links, de uma forma suave e faseada, por uma nova versão. Proporcionou ao estagiário um contacto direto com os clientes finais. As decisões que tomou e o resultado do seu trabalho tiveram, um impacto direto nas suas experiências de utilização. O estágio também proporcionou um primeiro contacto com as dificuldades e os desafios do ponto de vista de engenharia, inerentes à construção de um sistema moderno e escalável.

Os objetivos e os planos de tarefas propostos foram seguidos até ao fim com sucesso e encontram-se, atualmente, em utilização. O serviço C3 Cloud foi completamente substituído e ultrapassadas as limitações da primeira versão. O resultado do desenvolvimento permite a gestão de vários dispositivos com facilidade, o que não era possível com a versão inicial. O desenvolvimento de uma interface do utilizador web com componentes modernos também facilitou a utilização e melhorou a experiência de utilizador.

No estudo das topologias de rede concluiu-se que a topologia a adotar para a nova versão do C3 Cloud Control deveria tirar partido dos benefícios das topologias centralizada e distribuída: o controlo e gestão centralizados de dispositivos C3 e conteúdos e, ainda, a sincronização distribuída de conteúdos. O Syncthing provou ser uma ferramenta poderosa, segura e fiável para sincronizar ficheiros. Esta permite construir a rede distribuída e encaixa-se facilmente nos dispositivos C3 e nos C3 Cloud Control.

No estudo comparativo do C3 Cloud v1.0 e v2.0, concluímos que ambos usam compressão de dados quando possível, em cerca de 50% do conteúdo. Concluímos ainda que a nova versão enviou menos 41% tráfego, uma vez que a rede P2P tirou proveito das conexões entre os C3. O resultado mais significativo ocorreu no teste com dispositivos C3 na mesma rede, em que um destes já tem os conteúdos pretendidos. A sincronização foi 86% mais rápida e economizou 89% da largura de banda, em comparação com o C3 que comunicou com o C3 Cloud Control. Este resultado demonstra que a comunicação P2P funciona e que a transferência é mais rápida em LAN.

Um dos principais objetivos para a Cloud C3 era escalar sem problemas até 10 000 dispositivos, ou seja, o número de dispositivos conectados ao C3 Cloud Control deveria poder crescer e os

C3 deveriam conseguir sincronizar-se entre si sem problemas. No entanto, o estagiário concluiu que a ferramenta Syncthing não consegue atingir o objetivo pretendido utilizando a topologia *full mesh*. Se tivesse de recomeçar o trabalho do princípio, teria organizado mais cenários de grande escala, para garantir que a topologia *full mesh* não iria trazer problemas com a ferramenta escolhida. Entretanto este problema foi ultrapassado ao dividir a carga em *meshes* parciais.

7.2 Trabalho Futuro

O melhoramento do serviço é sempre uma mais valia. Com a REST API desenvolvida e bem documentada, esta pode implementar novos pedidos e o C3 Cloud pode ter mais funcionalidades.

A interface de utilizador pode ser melhorada de forma geral. Por exemplo, a página de entrada (*dashboard*) pode apresentar informação mais relevante ao utilizador. Também deveria ser possível abrir o conteúdo como um *preview* de como vai aparecer no C3. Os professores (um dos utilizadores da solução C3), poderiam criar mais tipos de conteúdo como trabalhos, testes, *quizzes*, perguntas e enunciados.

Há melhoramentos que podem ser feitos ao nível do *upload* de ficheiros. Atualmente, este tipo de operação é feito na interface do utilizador web, com indicação do progresso e notificação de término. No entanto, deveria haver mais estratégias de carregamento para ficheiros que demorem mais tempo a carregar, seja pela velocidade de ligação ou pelo tamanho do conteúdo.

Considera-se, ainda, que seja necessário melhorar o controlo sobre instâncias do Syncthing, facilitar o levantamento de novas instâncias e ter melhor perceção da carga (CPU e RAM) de cada uma. O escalonamento horizontal dos servidores centrais (C3 Cloud Crontrol) também pode ser uma mais valia para melhorar o desempenho das sincronizações.

7.3 Considerações finais

Para atingir os principais objetivos propostos, o estagiário teve de partir das bases teóricas e práticas adquiridas ao longo do percurso académico e ter uma boa aproximação a um contexto real de trabalho. Este teve o cuidado de seguir as melhores práticas possíveis de programação para melhor legibilidade do código produzido por todos os membros da equipa e, também, para que o projeto não estagnasse por falta de documentação. Através do estágio, a teoria aprendida pôde ser aplicada para solucionar problemas reais, enfrentando desafios inerentes à área de desenvolvimento de software.

O desenvolvimento deste projeto proporcionou, sem dúvida, um grande aumento de conhecimentos e experiência ao nível não só da parte técnica como também do trabalho em equipa dentro de uma empresa. A constante necessidade de comunicação entre os vários membros da equipa que integram o projeto se revelou uma mais-valia no crescimento do

estagiário enquanto profissional, dando-lhe mais ferramentas que o vão certamente ajudar a continuar num mercado de trabalho cada vez mais competitivo e exigente. Esta experiência contribuiu bastante para a maturidade do estagiário a nível pessoal e profissional.

REFERÊNCIAS

[1] "Critical Links," [Online]. Available: https://www.critical-links.com/. [Acedido em Novembro 2019].

[2] "Critical Links como grupo da Critical Software," [Online]. Available: https://www.critical-links.com/about-us/. [Acedido em 1 1 2020].

[3] J. V. Dijk, "Digital divide: Impact of access," em *The International Encyclopedia of Media Effects,* 2017.

[4] "ISEC," [Online]. Available: https://www.isec.pt/pt/Instituto/. [Acedido em 23 02 2018].

[5] "Lançamento do C3 Cloud Server 1.0," 10 Janeiro 2012. [Online]. Available: https://www.critical-links.com/2012/01/10/pressrelease-13/.

[6] "Lançamento do C3 Cloud Control 2.0," Janeiro 2019. [Online]. Available: https://www.critical-links.com/2019/01/03/press-release/.

[7] "Principios de segurança do Syncthing," 20 Julho 2020. [Online]. Available: https://docs.syncthing.net/users/security.html.

[8] S. Goyal, "Centralized vs Decentralized vs Distributed," Julho 2015. [Online]. Available: https://medium.com/delta-exchange/centralized-vs-decentralized-vs-distributed-41d92d463868.

[9] "Página de entrada do Dropbox," [Online]. Available: https://www.dropbox.com/. [Acedido em 20 7 2020].

[10] "Página de entrada do Onedrive," [Online]. Available: https://www.microsoft.com/pt-pt/microsoft-365/onedrive/online-cloud-storage. [Acedido em 20 7 2020].

[11] "Página de entrada do Google drive," [Online]. Available: https://drive.google.com/. [Acedido em 20 7 2020].

[12] "rsync official manual page," [Online]. Available: https://linux.die.net/man/1/rsync. [Acedido em 1 1 2020].

[13] "Sea File Webpage," [Online]. Available: https://www.seafile.com/en/home/.

[14] "OwnCloud official webpage," [Online]. Available: https://owncloud.org/. [Acedido em 15 1 2020].

[15] “Resilio Sync official wepage,” [Online]. Available: https://www.resilio.com/individuals/. [Acedido em 15 1 2020].

[16] “Syncthing,” 2020. [Online]. Available: https://syncthing.net/.

[17] “Página oficial do Syncthing,” [Online]. Available: https://syncthing.net/. [Acedido em 1 1 2020].

[18] “Syncthing usage data,” 10 2019. [Online]. Available: https://data.syncthing.net/.

[19] “Syncthing Github page,” [Online]. Available: https://github.com/syncthing/syncthing.

[20] “Github page of Syncthing Relay Server,” [Online]. Available: https://github.com/syncthing/relaysrv. [Acedido em 1 1 2020].

[21] “Página de entrada do MongoDB,” [Online]. Available: https://www.mongodb.com/. [Acedido em 20 7 2020].

[22] “Angular 6,” [Online]. Available: https://v6.angular.io/docs. [Acedido em 1 1 2020].

[23] “Metronic templates,” [Online]. Available: https://keenthemes.com/metronic/. [Acedido em 1 1 2020].

[24] “Bootstrap,” [Online]. Available: https://getbootstrap.com/. [Acedido em 1 1 2020].

[25] “Angular Cli,” [Online]. Available: https://cli.angular.io/. [Acedido em 1 1 2020].

[26] “Angular material,” [Online]. Available: https://material.angular.io/. [Acedido em 1 1 2020].

[27] “Materia Design,” [Online]. Available: https://material.io/design/. [Acedido em 1 1 2020].

[28] “Angular Bootstrap,” [Online]. Available: https://ng-bootstrap.github.io/#/home. [Acedido em 1 1 2020].

[29] “ChartJS official website,” [Online]. Available: https://www.chartjs.org/. [Acedido em 1 1 2020].

[30] “CodeMirror official website,” [Online]. Available: https://codemirror.net/. [Acedido em 1 1 2020].

[31] “Google Maps Javascript,” [Online]. Available: https://developers.google.com/maps/documentation/javascript/tutorial. [Acedido em 1 1 2020].

[32] “Font awesome,” [Online]. Available: https://fontawesome.com/. [Acedido em 1 1 2020].

[33] “Material Icons,” [Online]. Available: https://material.io/resources/icons. [Acedido em 1 1 2020].

[34] “Socket.io Client,” [Online]. Available: https://socket.io/docs/client-api/.

[35] “ngx-uploader,” [Online]. Available: https://github.com/bleenco/ngx-uploader.

[36] “NodeJS official,” [Online]. Available: https://nodejs.org/en/about/. [Acedido em 1 1 2020].

[37] “NodeJS for web development,” [Online]. Available: https://medium.com/swlh/11-reasons-to-use-node-js-for-web-application-development-5b2ed5877ff0. [Acedido em 1 1 2020].

[38] “Node package manager,” [Online]. Available: https://www.npmjs.com/. [Acedido em 1 1 2020].

[39] “Pagina oficial do Express.js,” [Online]. Available: https://expressjs.com/.

[40] “Página oficial do Passport.js,” [Online]. Available: http://www.passportjs.org/.

[41] “Página do módulo npm "express session",” [Online]. Available: https://www.npmjs.com/package/express-session. [Acedido em 1 1 2020].

[42] “Página oficial do Mongoose,” [Online]. Available: https://mongoosejs.com/.

[43] “Página oficial do Socket.io servidor,” [Online]. Available: https://socket.io/docs/server-api/.

[44] “Página oficial do Socket.io cliente,” [Online]. Available: https://socket.io/docs/client-api/.

[45] “Módulo npm do "node-syncthing",” [Online]. Available: https://www.npmjs.com/package/node-syncthing. [Acedido em 1 1 2020].

[46] “Página oficial do Swagger,” [Online]. Available: https://swagger.io/.

[47] “Git,” [Online]. Available: https://git-scm.com/. [Acedido em 1 1 2020].

[48] “Jira,” [Online]. Available: https://www.atlassian.com/software/jira. [Acedido em 1 1 2020].

[49] “Atlassian,” [Online]. Available: https://www.atlassian.com/. [Acedido em 1 1 2020].

[50] “Bitbucket,” [Online]. Available: https://bitbucket.org/. [Acedido em 1 1 2020].

[51] V. S. Code. [Online]. Available: https://code.visualstudio.com/.

[52] “Compreender o ID de dispositivo Syncthing,” [Online]. Available: https://docs.syncthing.net/dev/device-ids.html. [Acedido em 1 1 2020].

[53] “Syncthing release 1.4.0,” [Online]. Available: https://github.com/syncthing/syncthing/releases/tag/v1.4.0.

[54] “High-Speed Web-based Traffic Analysis and Flow Collection,” [Online]. Available: https://www.ntop.org/products/traffic-analysis/ntop/.

[55] J. Borg, “Tiny file synchronization is super slow,” [Online]. Available: https://github.com/syncthing/syncthing/issues/5568.

[56] “Manual do Rsync: Compressão,” [Online]. Available: https://download.samba.org/pub/rsync/rsync.html.. [Acedido em 15 1 2020].

[57] “Syncthing compression,” [Online]. Available: https://docs.syncthing.net/users/config.html. [Acedido em 1 1 2020].

[58] 2015. [Online]. Available: https://www.fastmetrics.com/internet-connection-speed-by-country.php.

[59] “Syncthing usage data,” August 2018. [Online]. Available: https://data.syncthing.net/.

[60] “Syncthing Docs - send-only-folder,” [Online]. Available: https://docs.syncthing.net/users/foldertypes.html#send-only-folder.

[61] “Syncthing Docs - receive-only-folder,” [Online]. Available: https://docs.syncthing.net/users/foldertypes.html#receive-only-folder.

[62] “ntopng,” [Online]. Available: https://www.ntop.org/products/traffic-analysis/ntop/.

[63] Syncthing, “Relaying,” [Online]. Available: https://docs.syncthing.net/users/relaying.html.

[64] J. Borg, “Large amount of files and big file size sync problems,” December 2015. [Online]. Available: https://forum.syncthing.net/t/large-amount-of-files-and-big-file-size-sync-problems/6044/41.

[65] R. compression, “Rsync manual page,” [Online]. Available: https://download.samba.org/pub/rsync/rsync.html.

[66] Syncthing, “Syncthing Config,” [Online]. Available: https://docs.syncthing.net/users/config.html.

[67] C. Links, “C3 Around the World, Costa Rica Case Study,” [Online]. Available: https://www.critical-links.com/2019/06/30/c3-around-the-world-costa-rica-case-study/.

[68] S. Goyal, “Centralized vs Decentralized vs Distributed,” [Online]. Available: https://medium.com/delta-exchange/centralized-vs-decentralized-vs-distributed-41d92d463868.

[69] “Topologia centralizada e distribuida fornecida pelo Syncthing,” [Online]. Available: https://forum.syncthing.net/t/topology-of-several-node-sync-network-star-or-grid/7269.

[70] “Syncthing Docs - introducer,” [Online]. Available: https://docs.syncthing.net/users/introducer.html. [Acedido em 1 1 2020].

ANEXO A - PROPOSTA DE ESTÁGIO

DEPARTAMENTO DE ENGENHARIA
INFORMÁTICA E DE SISTEMAS

PROPOSTA DE ESTÁGIO

Ano Lectivo de 2017/2018

em Mestrado em Informática e Sistemas no ramo de Desenvolvimento de Software

TEMA

C3 Content Cloud Server

SUMÁRIO

Este projeto de estágio pretende fazer a atualização de um componente já existente, conhecido como C3 Content Cloud Server, e será dividido em três fases:

- Fase 1: Desenvolvimento de uma API REST que comunicará com o serviço atual;
- Fase 2: Desenvolvimento de uma interface Web que comunicará com o C3 Content Cloud Server através da API REST desenvolvida na fase anterior;
- Fase 3: Substituição do backend do C3 Content Cloud Server, por uma versão moderna e escalável.

No final do projeto, o componente C3 Cloud Server terá sido totalmente substituído sem nunca ter ocorrido disrupção do serviço atual.

1. ÂMBITO

Este projeto insere-se num contexto real de trabalho, e, no final, será colocado em ambiente de produção. Atualmente, o servidor C3 Content Cloud Server serve mais de 4000 clientes em vários países. No entanto, na sua génese, este servidor não foi construído para escalar para este número de clientes, apresentando sérias limitações na sua capacidade para receber novos clientes. Além disso, não expõe APIs standard e bem documentadas para interagir com sistemas de terceiros. Neste sentido, este projeto de estágio irá permitir a substituição de um componente atualmente em produção, de uma forma faseada e suave. O objetivo aqui será minimizar o impacto no produto e em simultaneamente permitir a sua completa substituição com impacto mínimo para os clientes atuais.

Este projeto permitirá ao estagiário um contacto direto com ambiente de cliente e de produção, e o impacto que as decisões que tomar terão na experiência de utilização do cliente final. Insere-se igualmente num processo que a Critical Links está a levar a cabo para modernizar e estabilizar o seu portfólio de produtos.

2. OBJECTIVOS

O presente projecto pretende atingir os seguintes objectivos genéricos:

- Criação de uma API REST standard para comunicação com um servidor já existente;

- Criação de uma interface Web que utilizará a API anteriormente descrita;
- Substituição do backend, sem alterações na API REST criada.

3. PROGRAMA DE TRABALHOS

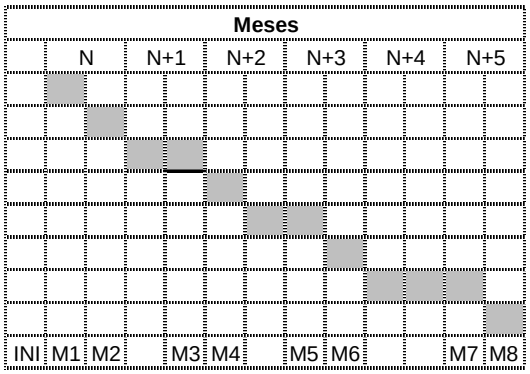
O estágio consistirá nas seguintes actividades e respectivas tarefas:

- ***T1*** – *Documentação da API existente* – Documentar exaustivamente a API existente de comunicação com o componente C3 Content Cloud Server. Esta API será, doravante, identificada como API legada.
- ***T2*** – *Desenho de uma API Rest* – Desenhar uma API REST, seguindo todas as boas práticas de desenvolvimento para o efeito, de forma a permitir expor uma estratégia de acesso à API legada através de pedidos REST.
- ***T3*** – *Broker API REST <-> API legada*. Desenho de arquitetura e desenvolvimento de um software broker que permita fazer a tradução entre a nova API REST que foi anteriormente desenhada, e a API legada.
- ***T4*** – *Planeamento de Interface WEB* – Estudo e seleção de uma ou várias frameworks para implementação de uma interface Web que faça uso da API REST desenvolvida. Desenho da arquitetura da solução.
- ***T5*** – *Interface Web* – Desenvolvimento de uma interface Web que utiliza a API REST desenvolvida
- ***T6*** – *Planeamento de Backend* – Estudo e seleção das ferramentas e linguagens de programação para implementar um novo backend que substituirá o atual C3 Content Cloud Server. Desenho da nova arquitetura.
- ***T7*** – *Implementação do novo backend* – De acordo com a planificação efetuada na T6, desenvolvimento do novo backend que irá substituir o atual.
- ***T8*** – *Ambiente de Produção* - Implementação, em ambiente de produção, do novo sistema desenvolvido até agora. Estratégias para substituição da solução legada.

4. CALENDARIZAÇÃO DAS TAREFAS

As Tarefas acima descritas, incluindo os testes de validação de cada módulo, serão executadas de acordo com a seguinte calendarização:

O plano de escalonamento dos trabalhos é apresentado em seguida:



INI		Início dos trabalhos
M1	(INI + 2 Semanas)	Tarefa T1 terminada
M2	(INI + 4 Semanas)	Tarefa T2 terminada
M3	(INI + 8 Semanas)	Tarefa T3 terminada
M4	(INI + 10 Semanas)	Tarefa T4 terminada
M5	(INI + 16 Semanas)	Tarefa T5 terminada
M6	(INI + 18 Semanas)	Tarefa T6 Terminada
M7	(INI + 24 Semanas)	Tarefa T7 Terminada
M8	(INI + 26 Semanas)	Tarefa T8 Terminada

5. RESULTADOS

Os resultados do estágio serão consubstanciados num conjunto de documentos a elaborar pelo estagiário de acordo com o seguinte plano:

M1:

R1.1: Documentação exaustiva da API legada.

M2:

R2.1: Proposta formal de uma API REST e respetivas regras de implementação.

R2.2: Tabela de equivalências da API legada para a API REST.

R2.3: Tabela com novos métodos não relacionados com a API legada.

M3:

R3.1: Arquitetura do Broker.

R3.2: Metodologia de testes unitários.

R3.3: Metodologia de testes de sistema.

R3.4: Instruções de instalação e configuração.

M4:

R4.1: Lista de frameworks web consideradas (estado da arte).

R4.2: Justificação para a escolha da framework ou frameworks.

M5:

R5.1: Instruções de instalação e configuração.

R5.2: Documentação de uso para cliente.

M6:

R6.1: Lista detalhada das ferramentas abordadas e suas valências (estado da arte)

R6.2: Justificação da escolha final das ferramentas.

M7:

R7.1: Documento de arquitetura do backend.

R7.2: Instruções de instalação e configuração.

R7.3: Manual de operação do componente

M8:

R8.1: Instruções de instalação, configuração e manutenção da solução completa.

R8.2: Manual final do produto C3 Content Cloud Server 2.0

6. LOCAL DE TRABALHO

Instalações da Critical Links SA no Parque Industrial de Taveiro, Coimbra.

7. METODOLOGIA

O presente trabalho será feito num contexto profissional. O produto final do projeto será utilizado em cenário de produção.

A Critical Links possui um processo de desenvolvimento de software, que será seguido pelo estagiário. Todos os requisitos que serão alvo de implementação estarão devidamente registados e atribuídos num sistema de gestão de tarefas. Adicionalmente, decisões que sejam tomadas durante a implementação de requisitos, poderão ser documentadas diretamente nos mesmos.

Toda a informação formal produzida será registada na Wiki interna da empresa. Essa informação poderá, consoante haja necessidade, ser exportada para formato Word para entrega a pessoas externas de interesse.

Haverá reuniões semanais de acompanhamento do projeto, assim como pontos de situação cada vez que um requisito seja terminado

No final do projeto, toda a documentação relevante estará registada na Wiki da empresa.

8. ORIENTAÇÃO

ISEC:

Nome (nome@isec.pt)
Categoria

Entidade de Acolhimento:
Helder Pereira (hmpereira@critical-links.com)
Diretor de Engenharia. Mestre em Engenharia Informática

9. CARACTERIZAÇÃO E REMUNERAÇÃO

- Data de início
- Data de fim
- 8 horas diárias em regime flexivel
- Formação on-job. Formação esporádica dada pelas empresas do grupo Critical
- Remuneração base para estágio IEFP.

ANEXO B - ANÁLISE DO C3 CLOUD V1.0

API documentation

- Document Goals
- C3 Cloud Profiles
- C3 Cloud Permissions
  - Administrator
  - Device Manager
  - Content Manager
- C3 Cloud Operations
  - 4.1 Tasks & stats operations
  - 4.2 User managing
  - 4.3 Content managing
  - 4.4 Device managing
    - 4.4.1 Device groups
- 5. API Endpoints
- 6. Models
  - 6.1 Default Response
  - 6.2 User
  - 6.3 Group
  - 6.4 School
  - 6.5 School Sync State
  - 6.6 Package
  - 6.7 Package Node
  - 6.8 Package Sync State
  - 6.9 Package Content
  - 6.10 Stat
  - 6.11 Job
- 7. Enums - status, types, modes
  - 7.1 ConnnectionState
  - 7.2 DistributionMode
  - 7.3 JobState
  - 7.4 PackageStatus
  - 7.5 PackageType
  - 7.6 PackageContentType
  - 7.7 SSynchronizeState
  - 7.8 UserType

Document Goals

The main goal of this document is to provide the current versions (v1.0 & v1.5) and the planned (v2.0) API endpoints created, specifying every input and output of every request. Each endpoint contains the partial URL, the used headers, the type of request (GET, PUT, POST, DELETE), a description and the input/output data if exists.

It is also documented the C3 cloud existing profiles and the operations each profile is allowed to perform.

C3 Cloud Profiles

Currently, C3 Cloud Server has 3 types of profiles:

- Administrator (by default, only the user "admin" is an administrator. This is not changeable and it's not possible to create additional administrators) - Administers the whole C3 Cloud Server. This is the only profile that can create new users;
- Device Manager - Can create new devices, and assign content to his own devices. Can create new content
- Content Manager - Can only create new content and administer his own created content.

C3 Cloud Permissions

Administrator

The C3 Cloud administrator shall be capable of performing the following operations:

- Manage content on the cloud;
- Manage C3 devices;
- Check tasks & stats;
- Manage users;

Device Manager

A C3 Cloud Device Manager shall be capable of performing the following operations:

- Manage content on the cloud;
- Manage C3 devices;
- Check tasks & stats;

Content Manager

A C3 Cloud Device Manager shall be capable of performing the following operations:

- Manage content on the cloud;

C3 Cloud Operations

This chapter contains all the possible operations a user can do.

4.1 Tasks & stats operations

- Clear finished tasks when the task finishes it is possible to remove them from the list, leaving only tasks with other status;

4.2 User managing

- Add new user add a new CCS user providing the username, password, role and name;
- Edit user edit a previously added C3 device's information;
- Remove user remove a previously added CCS user;

4.3 Content managing

- Add new content add new content by giving it a name, description, keywords and selecting the type of content (C3 content, moodle, kiwix, contentHub, Hidden);
- Edit content edits the information given when creating the content ;
- Remove content removes the content;

4.4 Device managing

- Add new device add a new C3 device and assign them a License ID (it is not be possible to add a device without its license Id);
- Edit device information edit a previously added C3 device's information;
- Remove device remove a previously added C3 device;
- Move device organize devices by moving them to other groups;
- Distribute updates/synchronizes all the information and content associated to the selected C3 device. When changes are made to a C3 device, it is not updated until the user distributes the changes.
- Edit device distribution mode it is possible to distribute/synchronize the C3 device: manually, as soon as possible, in the period between X and Y hours.
- Edit device content it is possible to add and remove content which exists on the cloud;

4.4.1 Device groups

- Add group create a group to separate C3 devices and therefore keeping devices organized by groups;
- Remove group if a group is no longer required it is possible to remove it;
- Move group organize groups by moving them to other groups;

5. API Endpoints

API version 1.0.

API version 1.5.

6. Models

This section contains object models of the existing API. For each object it is shown the variable name, value type and a value description.

6.1 Default Response

Key	Value Type	Value Description
data	Object	Data can be any object, varies according to the request made
success	Boolean	Response success is true or false
message	String	Response message
errorTrace	String	Response errorTrace, in case of error

6.2 User

Key	Value Type	Value Description
id	Integer	User ID
type	Integer	Type of user (profile/role) • Guest = 0; • Normal = 1; • SchoolAdministrator = 2; • Administrator = 3;
creation	Long	User creation GMT timestamp
last	Long	User last access GMT timestamp
name	String	User name
login	String	User login credentials
password	String	User password holder

6.3 Group

Key	Value Type	Value Description
id	Integer	School id
parentId	String	Parent school id (-1 if it is already the parent)
name	String	School Name
description	String	Shool description
subGroups	Collection<CDGroup> (CDGroup = this)	Collection of sub groups existing on the current group

6.4 School

Key	Value Type	Value Description
actionPeriod	String	Action period
address	String	School address

bandwidth	Long	School bandwidth
connectionState	Integer	State of the connection
creationDate	Long	School creation GMT timestamp
diskFree	Long	School disk free space
diskTotal	Long	School total disk space
distributionMode	Integer	School mode to distribute the content: manually, as soon as possible, in the period between X and Y hours.
groupId	Integer	School group id
id	Integer	School id
ip	String	School IP
lastContact	Long	GMT timestamp of the last contact (not sure)
lastDate	Long	GMT timestamp of the last time the school was distributed
licenseId	String	School license id, with minimun 16 chars containing only [0123456789abcdef]
location	String	Latitude and longitude separated by a comma. Example: "40.1979, -8.5135"
message	String	School's message
messageTimestamp	Long	School's message GTM timestamp of its creation/alteration
name	String	School name
os_release	String	OS release
pollingInterval	Long	Polling interval in seconds to fetch information about the school
port	Integer	Schools port
synchronizeState	Integer	State of school's synchronization
url	String	School website url
userId	Integer	Creator id of this school
userName	String	Creator name of this school

6.5 School Sync State

int schoolId;
int packageId;
CDSJobState jobState;
std::string message;
SyncInfo syncInfo;

Key	Value Type	Value Description
schoolId	Integer	School id
packageId	Integer	Package id
message	String	Message
jobState	Integer	Job state, see JobState enum
totalItems	Double	Total items
totalBytes	Double	Total bytes
totalFiles	Double	Total files
syncItems	Double	?
syncBytes	Double	?

tranferredBytes	Double	Transfered bytes
percentDone	Integer	Percent done
rate	Double	Rate
xfer	Double	Transfer
toCheck	Double	?
total	Double	?
totalPercentageDone	Double	Total percetage done

6.6 Package

Key	Value Type	Value Description
creationDate	Long	Package GTM timestamp of the creation date
contentType	Integer	Type of content: • DEFAULT = 0; • MOODLE = 1; • KIWIX = 2;
description	String	Package description
id	Integer	Package id
itemsCount	Integer	Number of items inside the package
itemsSize	Integer	Size in Bytes of all content inside the package
keywords	String	Keywords to associate to the package
lastUpdate	Long	Last update GMT timestamp
name	String	Package name
ownerId	Integer	Owner id
ownerName	String	Owner name
sortPriority	Integer	Default value is -1. It is only filled in SContent.listSchoolPackagesSubscribed
status	Integer	Status of the content: • READY = 0; • MAINTENANCE = 1; • UPDATE = 2;
type	Integer	Type of the content: • Public = 0; • Private = 1;

6.7 Package Node

Key	Value Type	Value Description
directory	Boolean	Boolean indicating if current node is a directory or not
path	String	Path of the node

6.8 Package Sync State

Key	Value Type	Value Description
packageId	Integer	Package id
subscribedSchools	Integer	Number of schools with this package
synchronizedSchools	Integer	Number of schools synchronized with this package

6.9 Package Content

Key	Value Type	Value Description
path	String	Path to content
size	Integer	Size of content
lastUpdate	Long	Last update GMT timestamp

6.10 Stat

Key	Value Type	Value Description
endTime	Long	End time GMT Timestamp
packageName	String	Size of content
schoolName	String	Last update GMT timestamp
startTime	Long	Start time GMT Timestamp
totalSize	Long	Total size in bytes
transferedRate	Float	Transfered rate
transferedSize	Long	Transfered size in bytes

6.11 Job

Key	Value Type	Value Description
eta	Integer	Expected time to finish the job
jobState	Integer	Integer indicating the state of the job • WAITING = 0 • RUNNING = 1 • CANCEL = 2 • ERROR = 3 • FINISHED = 4 • EXPIRED = 5 • SHEDULED = 6
message	String	Message of the job (task)
packageId	Integer	Package id
packageName	String	Package name
rate	Float	Transfer rate
schoolId	Integer	School id
schoolName	String	School name
totalSize	Long	Total size in bytes
transferSize	Long	Transfer size

7. Enums - status, types, modes

7.1 ConnnectionState

- OK = 0
- INVALIDPASSWORD = 1
- INVALIDHOST = 2
- TIMEOUT = 3
- UNKNOWN

7.2 DistributionMode

- UNKOWNM
- MANUAL
- ASAP
- SHEDULED

7.3 JobState

- WAITING = 0
- RUNNING = 1
- CANCEL = 2
- ERROR = 3
- FINISHED = 4
- EXPIRED = 5
- SCHEDULED = 6

7.4 PackageStatus

- READY = 0
- MAINTENANCE = 1
- UPDATE = 2

7.5 PackageType

- PUBLIC = 0
- PRIVATE = 1

7.6 PackageContentType

- DEFAULT = 0
- MOODLE = 1
- KIWIX = 2

7.7 SSynchronizeState

- SYNCHRONIZED = 0
- SYNCHRONIZING = 1
- BUSY = 2
- BLOCK = 3
- SCHEDULED = 4
- UNAVAILABLE = 5
- WAITING = 6
- ERROR = 7
- NOTSYNC = 8
- UNKNOWN = 9

7.8 UserType

- GUEST
- NORMAL (content manager)
- SCHOOL_ADMIN (device manager)

- ADMIN

ANEXO C - DOCUMENTAÇÃO DO C3 CLOUD V2.0

API v2.0

Index

1 Architecture

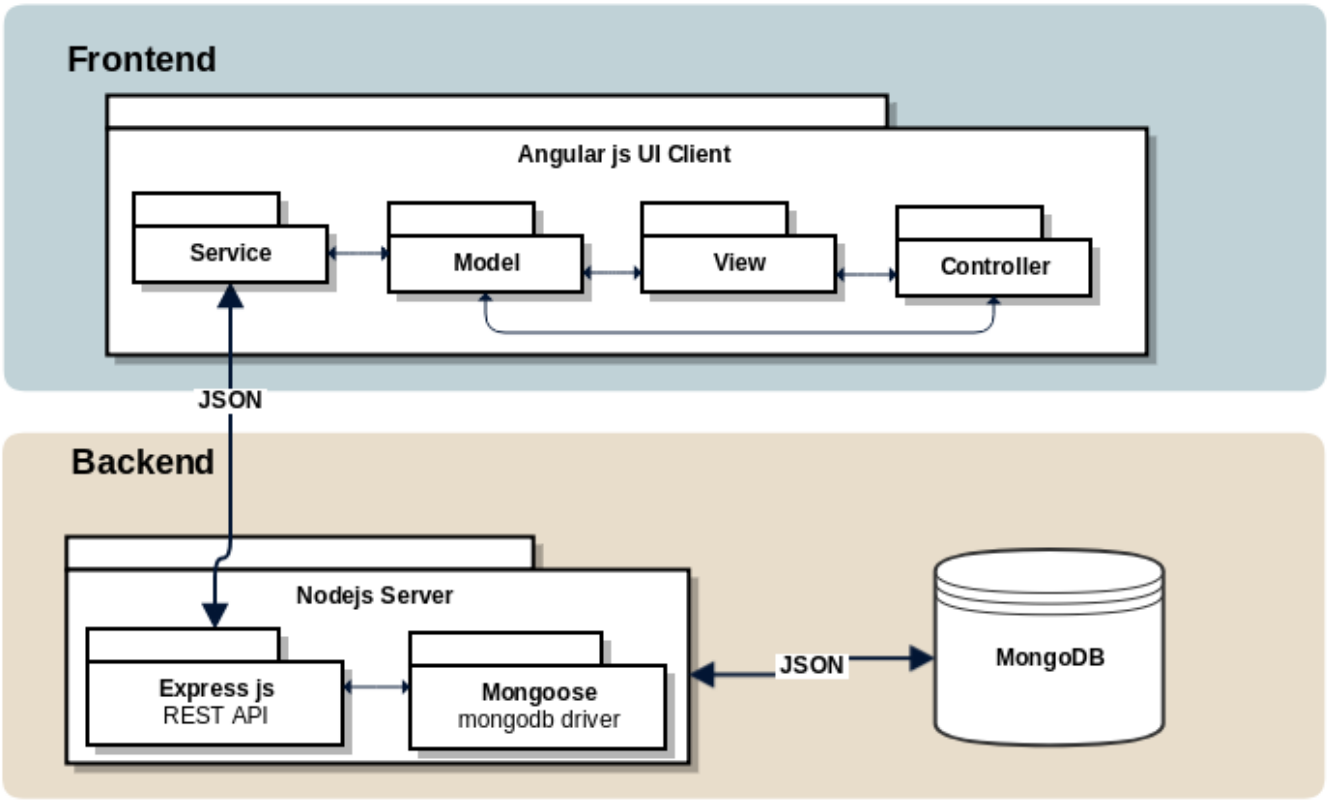
2 Models
- 2.1 User
- 2.2 Device
- 2.3 Device Connection
- 2.4 Content
- 2.5 Device Group

3 Enums
- 3.1 User
- 3.2 Content

4 Responses
- 4.1 Success
- 4.2 Error

5 Endpoints
- 5.1 System
- 5.2 Auth
- 5.3 User
- 5.4 Device
- 5.5 Group
- 5.6 Contents
- 5.7 Syncthing

Architecture



Models

Some variables are not sent to frontend but exists on the backend (example: user password).

User

User Model

```
{
        id                                  String
        username                String                  // required,
unique
        avatar                          String                  //
reconsider if is necessary
        name                            String                  // required
    creation                Date                  // updated on creation
    lastLogin                Date                  // updated when
logged in
    myDevices                [Device]          // user devices
    myContents                [Content]          // user created contents
    type                        Number                  // min: 1, max: 3
}
```

User Model Example

Device

Device Model

```
{
        name                                    String
        creation                              Date
        address                             String
        location                            String
        message                                    String
        website                                    String
        owner                                   User
        deviceUsers                           [User]
        deviceContents                 [Content]
        config: {
                lan                                        String
                bandwidth                 Number
                ip                                  String
                licenseId                 String
                syncthingId                 String
        }
}
```

Device Model Example

Device Connection

Device Model

```
{
        device                          Device
        connected                    Boolean
        inBytesTotal              Number
        paused                          Boolean
        at                                  Date
        clientVersion             String
        address                          String
        type                           String
        outBytesTotal             Number
}
```

Device Model Example

Content

Content Model

```
{
        name                        String
        folderId                String
        name                        String                // required,
unique
        description                String
        thumbnailPath        String                // default: '
/thumbnails/no-image'
        creationDate        Date
        lastUpdate                Date
        totalBytes                Number
        totalFiles                Number
        contentType                Number                // min: 0,
max: 4, default: 0
        visibility                Number                // min: 0, max:
1, default: 0
        owner                        User
        keywords                [String]
        subscribedDevices        [Device]
}
```

Content Model Example

Device Group

Device Group Model

```
// device groups could just be an array of devices, instead of a tree .
helder?
{
        name                                String
        description                        String
        deviceList                        [Devices]
        subscribedContents        [Content]
}
```

Enums

User	type	• 1 ContentManager • 2 DeviceManager • 3 Admin
Content	status	• paused • unknown • unshared • stopped • scanning • iddle • syncing • outofsync
	type	• 0 C3 Content • 1 Moodle • 2 Kiwix • 3 Intel Content Hub • 4 Hidden
	visibility	• 0 Public • 1 Private
	distributionMode	• 1 Automatic • 2 Scheduled

Responses

This API responses follow a standard documented on this chapter.

To get a better understanding of the responses given by the REST API, HTTP status codes and JSON objects are used to maintain a consistent reliable response.The following code block represents the main body of a general response.

Success

On success, the HTTP status code in the response status is 200 and it returns *what is asked* in JSON format. The response data may contain an object or an array of objects acording to the request

Error

The main goal with error responses is to create a source of information to not only inform the user of a problem, but of the solution to that problem as well. It is used HTTP codes in order to fully separate the errors:

- 400 (Bad Request) - Any case where a required parameter is missing, or a parameter is invalid.
- 401 (Unauthorized) - The auth token was provided but invalid.
- 403 (Forbidden) - Not allowed to access.
- 404 (Not Found) - Endpoint does not exist.
- 500 (Internal Server Error) -The request is probably valid but something went wrong on the server.

The point of a unique code foreach error message is to be translated by any client. The status and the message are merely informative.

Error example

```
{ code: 1, status: 401, message: "Not logged in" }
```

Endpoints

In token-based authentication, the client exchanges hard credentials (which are username and password) for a piece of data called token. For each request, instead of sending the hard credentials, the client will send the token to the server to perform authentication and then authorization.

> **Access Token**
> The login is the only request which return the user's **access token**.
> All the other requests require the **access token** in the header request in order to identify who is the user or else 401 Unauthorized is replied.

	Description	Method	Endpoint	Params	Response data
System	Get current system status and resource usage	GET	*/api/v2/system*	-	...
	Get version	GET	*/api/v2/system/version*	-	...

	Description	Method	Endpoint	Params	Response data
Auth	get current user	GET	*/api/v2/auth*	-	User
	login	POST	*/api/v2/auth*	{username: string, password: string}	User
	logout	DELETE	*/api/v2/auth*	-	-
	Change current user password	PUT	*/api/v2/auth/password*	currentPassword - mandatory newPassword - mandatory newPasswordConfirmation - mandatory	User

	Description	Method	Endpoint	Params	Response data
User	Get all	GET	*/api/v2/users*	-	User List
	Create	POST	*/api/v2/users*	{}	User
	Get user	GET	*/api/v2/users/:user*	-	User
	Edit user	PUT	*/api/v2/users/:user*	{}	User
	Delete user	DELETE	*/api/v2/users/:user*	-	-
	Get users with given type	GET	*/api/v2/users/type/:type*	-	User

	Description	Method	Endpoint	Params	Response data
Device	Get all devices	GET	*/api/v2/devices*	-	Device List
	Create one device	POST	*/api/v2/devices*	• name - *mandatory* • licenceId - *mandatory* - minimum 16 chars containing only [0123456789abcdef] • groupId - *optional* • message - *optional* • address - *optional* • website - *optional* • location - *optional* - array with lat and long [latitude, longitude] • ip - *optional* - device IP • distributionMode - *optional* - see distribution mode • actionPeriod - *optional* - object with 'init' and 'end' timestamps • maxRecvKbps - *optional* - limit receive bandwidth • minSendKbps - *optional* - limit send bandwidth • globalAnnounceServers - *optional* - list of servers for C3 to announce its existence	Device

	Create multiple devices with a .csv file Obs: CSV Header is optional, but if exists, the excpected columns are "name" and "licenseId". The CSV file must contain 2 columns: *column1* contains device names, *column2* contains license Ids.	POST	**/api/v2/devices/mass**	• file - *mandatory* - CSV file • groupId - *optional* • users - *optional* - • contents - *optional* - d •	# of added devices & Errors
	Get device	GET	**/api/v2/devices/*:device***	-	Device
	Edit device	PUT	**/api/v2/devices/*:device***	{}	Device
	Delete device	DELETE	**/api/v2/devices/*:device***	-	-
	Get device users	GET	**/api/v2/devices/*:device*/deviceUsers**	-	User List
	Add user to device	PUT	**/api/v2/devices/*:device*/deviceUsers/*:userId***	-	-
	Remove user from device	DELETE	**/api/v2/devices/*:device*/deviceUsers/*:userId***	-	-
	Get device contents	GET	**/api/v2/devices/*:device*/deviceContents**	-	Content List
	Subscribe content to device	PUT	**/api/v2/devices/*:device*/deviceContents/*:contentId***	-	-
	Unsubscribe content from device	DELETE	**/api/v2/devices/*:device*/deviceContents/*:contentId***	-	-
	check if contents on device are the same of the contents on device	GET	**/api/v2/devices/**		

	Description	Method	Endpoint	Params	Response data
Group	get all groups	GET	**/api/v2/groups**	(Optional - get tree format) query param: tree=1	Group List (or Group Tree)
	create group	POST	**/api/v2/groups**	• name - *mandatory* • description - *optional* • parent - *optional* - default is group 'All'	Group
	get group	GET	**/api/v2/groups/*:group***	all *optional* • name • description • parent	Group
	edit group	PUT	**/api/v2/groups/*:group***		-
	delete group	DELETE	**/api/v2/groups/*:group***		-
	get devices inside group	GET	**/api/v2/groups/*:group*/devices**		Device List
	add device to group	PUT	**/api/v2/groups/*:group*/devices/*:deviceId***	-	-
	remove device from group	DELETE	**/api/v2/groups/*:group*/devices/*:deviceId***	-	-
	get devices inside group	GET	**/api/v2/groups/*:group*/contents**		Content List
	add content to group	PUT	**/api/v2/groups/*:group*/contents/*:contentId***		-
	remove content from group	DELETE	**/api/v2/groups/*:group*/contents/*:contentId***		-

	check if "group contents" are the same of each device inside this group	GET	/api/v2/groups/:group/sameContents		True or False
	foreach device in given group, change their "subscribed contents" to match the group content	PUT	/api/v2/groups/:group/sameContents		-

	Description	Method	Endpoint	Params	Response data
Contents	get all contents	GET	/api/v2/contents	-	
	create content	POST	/api/v2/contents	{}	
	get content	GET	/api/v2/contents/:content	-	
	edit content	PUT	/api/v2/contents/:content	-	-
	delete content	DELETE	/api/v2/contents/:content	-	
	get subscribed devices	GET	/api/v2/contents/:content/subscribedDevices	-	
	get content status	GET	/api/v2/contents/:content/status	-	-
	get content nodes (files and folders)	GET	/api/v2/contents/:content/nodes	-	
	change content nodes	POST	/api/v2/contents/:content/nodes	-	
	add thumbnail or replace current	PUT	/api/v2/contents/:content/thumbnail	-	-
	remove thumbnail	DELETE	/api/v2/contents/:content/thumbnail	-	-
	upload new content	POST	/api/v2/contents/:content/upload	-	True or False
	extract content	POST	/api/v2/contents/:content/extract		
	Enter maintenance (pause)	PUT	/api/v2/contents/:content/maintenance		
	Leave maintenance (unpause)	DELETE	/api/v2/contents/:content/maintenance		
	Get all distinct used content keywords	GET	/api/v2/contents/distinctkeywords		

	Description	Method	Endpoint	Params	Response data
Syncthing	get syncthing information (version)	GET	/api/v2/sync		
	get a device completion, connection, status list	GET	/api/v2/sync/devices/:device		Device Connection List
	get device completion list	GET	/api/v2/sync/devices/:device/completions		Device Connection
	get device connection list	GET	/api/v2/sync/devices/:device/connections		
	get device stats	GET	/api/v2/sync/devices/:device/stats		
	check if is paused	GET	/api/v2/sync/devices/:device/pause	-	True or False
	pause	PUT	/api/v2/sync/devices/:device/pause	-	-
	unpause	DELETE	/api/v2/sync/devices/:device/pause	-	-
		GET	/api/v2/sync/contents/:content		Device Connection
		GET		-	-
		GET		-	-
				-	-

				-	-
				-	-
	get addresses found by discovery server	***GET***		-	