



XUNTA DE GALICIA
CONSELLERÍA DE EDUCACIÓN
E ORDENACIÓN UNIVERSITARIA

I.E.S. ARMANDO
COTARELO VALLEDOR



Curso 2020-2021

MyCloudApp

Aplicación web de nube dixital para
almacenar ficheiros

Data presentación: 11/12/2020

**DAVID MIGUEL
FAJARDO OUBIÑA**

Proxecto final do Ciclo Formativo de Técnico Superior en Administración de Sistemas
Informáticos en Rede. Curso 2019/2020. IES ARMANDO COTARELO VALLEDOR

Índice

1.	Obxectivos e descrición abreviada do proxecto	3
2.	Estudo do mercado e plan económico	4
3.	Requirimentos técnicos e humanos	8
4.	Planificación	11
5.	Desenvolvemento técnico.....	13
6.	Prototipo	23
7.	Dificultades atopadas.....	26
8.	Conclusións	27
9.	Bibliografía, referencias e recursos utilizados.....	29
10.	Anexos	31

1. Obxectivos e descrición abreviada do proxecto

MyCloudApp consiste nunha aplicación web de nube dixital para almacenar arquivos.

O obxectivo é lograr un paquete de software multiplataforma que corra un servidor de ficheiros ao que se lle envíen arquivos e os almacene nun directorio dado, facendo que actúe como un NAS dentro da rede local da casa ou da oficina, a modo de almacén persoal, sendo accesible dende calquera dispositivo cun navegador web con acceso á rede a través dunha web-app.

A idea xorde de que en vista do rápido que os usuarios adquiren novos produtos tecnolóxicos, non é estraño que se teña algún PC antigo sen uso na casa. MyCloudApp xorde de tentar aproveitar eses equipos, dándolles unha segunda vida como un servidor de ficheiros.

Dende sempre fascináronme as tecnoloxías de nube, e como usuario recorrente de servizos como Google Drive, OneDrive, iCloud ou Dropbox (por citar os máis comúns) producíame curiosidade poder chegar a entender como algo que considero tan útil podía ser feito e funcionar internamente.

Neste proxecto, a unha escala moito menor, tento acadar a funcionalidade principal coa que xorden eses programas: poder gardar nun espazo común, accesible dende todos os dispositivos dun usuario, toda a información que este queira manter compartida.

Estrutura: A app está baseada en dous bloques principais:

- Servidor (Backend): É o encargado de xestionar a lóxica trala aplicación. Recibe os ficheiros e os administra nun directorio dado.
- Cliente (Frontend): Será onde os usuarios se conecten a través do navegador (do PC, móbil ou outro dispositivo con navegador web). Alí poderán facer todas as accións permitidas, como visualizar os directorios e arquivos, subir novos ficheiros ou eliminalos.

2. Estudo do mercado e plan económico

A finalidade de este proxecto é poder levar unha solución de servizo de ficheiros na nube rápida, moderna e a baixo custo ás pequenas e medianas empresas.

A miña intención é que mediante a reutilización de equipos ou o aproveitamento dos existentes xa en funcionamento, se poida gañar unha funcionalidade tan relevante do modo máis sinxelo posible, facilitando a vida dos nosos clientes á vez que reducimos a cantidade de refugалlos electrónicos xerados.

Vendo os prezos que se manexan actualmente, acadaríase a rentabilidade do servizo mediante 20 instalacións básicas ó mes. O obxectivo sería lograr unhas 300 instalacións no primeiro ano da empresa.

Sector:

Debido a que se trata dun concepto relativamente recente, os servizos de ficheiros na nube continúan sendo un mercado en expansión, con aínda moitos novos clientes potenciais. Un gran número de empresas, especialmente de pequeno tamaño por ser as que dispoñen de menor orzamento e persoal, continúan sen informatizarse completamente, debido á falta de organización dixital. É por isto que o noso enfoque de mercado céntrase nestas pequenas e medianas empresas, que por esas carencias, poidan estar pospoñendo esa informatización á espera dun sistema cunha maior rentabilidade respecto ó custo.

Produto:

Solucións como MyCloudApp promoven facilitar esa transición á organización dixital proveendo un sistema de ficheiros onde centralizar os datos compartidos da oficina de forma rápida e accesible dende calquera dispositivo. Debido ó enfoque de reutilización de equipos, os custos redúcense amplamente, acadando así un prezo atractivo para este tipo de empresas.

Análise PESTEL:

O sector da informática avanza a pasos axigantados cada ano. As tecnoloxías baseadas na nube atópanse experimentando un dos maiores crecementos dos últimos tempos.

As páxinas web están en constante evolución e cambio. Actualmente existen modernas tecnoloxías para a construción de aplicacións web SPA (Single Page Application) con modernas interfaces, que fan non só máis cómoda e agradable a experiencia de uso fronte ás aplicacións de escritorio ou móbiles tradicionais, se non que facilitan enormemente o despregue e implementación das mesmas no ritmo laboral a través da rede, sen necesidade de realizar instalacións nos equipos clientes e funcionando dende o navegador.

Isto promove que cada vez vexamos máis ferramentas e apps baseadas en tecnoloxías web, fronte ás clásicas apps de escritorio ou móbil.

Factores	Oportunidades	Ameazas
Políticos	A nacionalidade española da empresa supón una vantaxe de cara a exportar dentro e fóra da Unión Europea.	Altos impostos de cara ás empresas de carácter tecnolóxico.
Económicos	Os salarios baixos españois facilitan a contratación de novos traballadores.	O mal estado da economía promove que as empresas sexan máis conservadoras á hora de investir en tecnoloxía.
Sociais	Os clientes apreciarán a estratexia de prezos baixos do produto.	A falta de experiencia con tecnoloxías na nube pode desanimar dalgúns clientes de dar o paso.
Tecnolóxicos	O uso de tecnoloxías modernas xenera una mellor experiencia de usuario e sensación de fluidez ó traballar.	Unha competidora máis grande, con maior presuposto e persoal, que decidise facer un produto similar con esas novas tecnoloxías, seríalle sinxelo igualarnos.
Ecolóxicos	A nosa reutilización de equipos evita que se xeneren novos residuos tecnolóxicos, contribuído o medio ambiente.	A nosa venta de servidores pode chocar moralmente coa publicidade ecolóxica da reutilización de equipos da que presumimos.
Legais	Este tipo de negocio resulta sinxelo a nivel legal.	Potenciais problemas coas leis de propiedade intelectual en caso de que alguén utilizase o noso software sen consentimento, debido á falta de regulación no plaxio de código.

Análise estratéxico - vantaxe diferencial:

A gran maioría das solucións existentes no mercado baseadas na web, empregan tecnoloxías "estáticas" (HTML+CSS+JS). A velocidade do desenvolvemento destas tecnoloxías nos últimos anos provoca que exista moita diferenza entre esas webs, e as feitas mediante compoñentes con frameworks como Vue, Angular ou React.

A fluidez na experiencia de usuario, a escalabilidade dos deseños estruturados por compoñentes, e a facilidade de construción respecto ás solucións máis clásicas con HTML e JS plano, fan a nosa solución máis moderna, atractiva e funcional.

Análise DAFO:

	Orixe interno	Orixe externo
Puntos débiles	Debilidades: O produto está menos maduro ca competencia, carecemos de base de clientes sólida ao empezar desde cero.	Ameazas: A competencia ten máis anos de experiencia e un produto máis pulido despois de tanto tempo no mercado
Puntos fortes	Fortalezas: O produto foi deseñado coas tecnoloxías web máis novas a día de hoxe.	Oportunidades: A cantidade de clientes potenciais que aínda non dispoñen de servizo na nube

Financiación:

Instalación - Existen dous tipos de instalacións á dispoñibilidade dos clientes:

- **Básica:** Instalación e configuración de MyCloudApp.
 - Non inclúe equipo, só o software. A empresa deberá adquirir un equipo pola súa conta ou empregar un xa existente.
 - Inclúe 3 meses de *soporte básico* gratuíto.
- **Completa:** Instalación e configuración de MyCloudApp + equipo hardware.
 - Inclúe un equipo valorado en 200€.
 - Inclúe 6 meses de *soporte premium* gratuíto.

Táboa comparativa	Instalación básica	Instalación completa
Software MyCloudApp:	Si	Si
Hardware incluído:	Non	Si
Tipo de soporte:	Básico	Premium
Tempo de proba do soporte:	3 meses	6 meses
Prezo (pago único):	100€	400€

Soporte: O servizo de soporte disporá de dúas opcións:

- **Soporte básico:** Control do estado do servidor mediante software de monitorización Zabbix. Notificacións ante calquera problema co equipo servidor (CPU, rede, discos saturados, problemas de RAM) por SMS, correo electrónico ou teléfono.
 - Tempo de contestación ante unha incidencia grave de 8h.
 - Tempo de resolución dunha incidencia grave nun máximo de 24h.
- **Soporte premium:** Inclúe todas as características do *soporte básico*. Engade informes semanais con gráficas sobre o uso do equipo servidor e acceso a un panel de monitorización onde visualizar a información en tempo real.
 - Tempo de contestación ante unha incidencia grave de 1h.
 - Tempo de resolución dunha incidencia grave nun máximo de 2h.

Táboa comparativa	Soporte básico	Soporte premium
Monitorización do servidor:	Si	Si
Notificacións de incidencias:	Si	Si
Informes semanais:	No	Si
Acceso ó panel web:	No	Si
Tempo de contestación de incidencia grave leve:	8h 16h	1h 2h
Tempo de resolución de incidencia grave leve:	24h 48h	2h 4h
Prezo (subscrición mensual):	25€ / mes	75€ / mes

3. Requirimentos técnicos e humanos

Capital humano necesario

Contrataranse inicialmente a cinco traballadores:

- Un/unha encargado/a da instalación e configuración do software MyCloudApp nos equipos propios para preparalos para a entrega ou proporcionados pola empresa in situ, e o despregue e configuración do servizo na empresa cliente.
- Tres encargados/as da xestión dos erros detectados mediante Zabbix e soporte dos clientes, repartindo entre os tres o horario de soporte de 07h a 23h de luns a sábado, así como o a asistencia de forma presencial ante incidencias cando non sexa posible solventalas de forma remota.
- Un/Unha encargado/a da publicidade e marketing da empresa. Con coñecementos en SEO que promova unha páxina web empresarial onde anunciarnos e campañas de anuncios a través de Google Adds ou Facebook Adds.

Capital monetario necesario

Estimacións para os primeiros 6 meses de vida da empresa:

- 1200€ de salario bruto mensual por traballador: $1200 \times 5 \times 6 = 36.000\text{€}$
- Presuposto para a campaña publicitaria: 10.000€
- Adquisición dunha furgoneta de empresa: 10.000€
- Arrendamento e preparación dunha oficina básica nun baixo en Vilagarcía de Arousa: $(800 \times 6) + 3000 = 7.800\text{€}$

Total: 65.800€

- Baseándonos nestas estimacións, consideramos que a inversión inicial necesaria para comezar o negocio de xeito cómodo rondaría os 75.000€.

Requirimentos técnicos: tecnoloxías empregadas

Nun proxecto tan amplo con tan alta variedade de campos, utilízanse unha gran cantidade de tecnoloxías. Para simplificalo, só vanse listar as principais, as que máis peso teñen no desenvolvemento do proxecto, ou xeneralizando nas que están formadas por varias subferramentas pequenas.

- **Visual Studio Code**

Editor multiplataforma de código desenvolvido por Microsoft. Conta con gran cantidade de plugins que permiten estender as súas funcionalidades, adaptándoo á cada tipo de proxecto según que tecnoloxías se estean a empregar.



- **NPM**

Node Package Manager. É o xestor de paquetes para NodeJS por defecto. Conta cunha gran variedade de módulos instalables dende consola que permiten estender as funcionalidades de Node de diversas formas.



- **Node JS**

Entorno de execución multiplataforma para a capa de servidor baseado na linguaxe JavaScript. Asíncrono, con entrada e saída de datos nunha arquitectura orientada a eventos e baseado no motor V8 de Google. Posibilita a utilización de código JavaScript dende o lado do servidor.



- **Express JS**

Marco de aplicación web para NodeJS. Diseñado para facilitar a creación de aplicacións web e APIs. É o framework estándar para NodeJS.



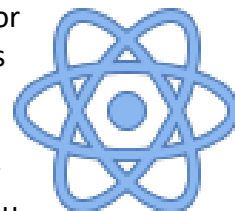
- **Postman**

Ferramenta de probas de APIs. Permite crear peticións sobre APIs dunha forma moi sinxela e así testear o seu correcto funcionamento.



- **React JS**

Librería de JavaScript (as veces mal chamada framework) mantida por Facebook e deseñada para a creación de interfaces de usuario e sitios SPA (Single Web Application). Mantén unha estrutura por compoñentes e un funcionamento mediante estados que o fai moi eficiente no consumo de RAM e datos de rede, evitando recargar compoñentes nalgúns situacións onde outros frameworks ou librerías vense obrigados.



- **Bootstrap 4**

Biblioteca HTML, CSS e JS para o deseño de interfaces frontend, orientada a dispositivos móbiles. É a librería de este tipo máis empregada, sendo o 2º proxecto máis destacado de todo GitHub.



- **W3.CSS**

Librería CSS orientada a dispositivos móbiles. Creada por w3schools, foi deseñada como unha alternativa máis simple, compacta e fácil de empregar á Bootstrap. Foi deseñada tamén para ser independente de Bootstrap, jQuery ou calquera outra librería coñecida, permitindo o uso simultáneo con outras sen crear conflito.

- **Mongo DB**

Sistema de base de datos NoSQL, orientado a documentos. En vez de almacenar os datos en táboas como nas bases de datos relacionais, MongoDB os garda en estruturas de datos BSON (similar a JSON) cun esquema dinámico, facendo que a integración cos datos en certas apps (sobre todo pequenas) sexa máis sinxela e rápida que con sistemas SQL.



- **Zabbix**

Software de monitoreo que recolle e centraliza información de dispositivos de rede, servidores, aplicacións ou servizos. Componse de dous compoñentes:



1. zabbix-server: o programa como tal. Instalado nun servidor, recibe os datos e os organiza, permitindo múltiples opcións e organizacións, así como establecer sensores en certos parámetros para recibir alertas en caso de problemas (moita RAM usada, caída de rede, etc...).
2. zabbix-agent: instálase nos clientes e recolle información sobre as características de hardware, versións de software, medidas de uso e moito máis. Envía esa información cifrada ó zabbix-server.

4. Planificación

Fases do proxecto:

1. Servidor Node JS (backend) que reciba ficheiros a través da rede.
2. Servidor que almacene e xestione os ficheiros recibidos nun directorio de traballo.
3. Cliente React (frontend) que se conecte ó servidor.
4. Rexistro e inicio de sesión de usuarios contra unha base de datos MongoDB.
5. Autenticación: Precisar de conta para acceder ao workdir do servidor.
6. Monitorización do estado do servidor con Zabbix.

A continuación detállanse cronoloxicamente os avances de cada etapa.

1. 7/10/2020 – 16/10/2020

Inicio dun proxecto de Node. Agregáronse as dependencias e o marco de expressJS. Existe un módulo chamado [express-fileupload](#) que permite subir arquivos e procesalos a través de express. Con el, pódese enviar ficheiros ó servidor Node, pero aínda que non se fai nada con eles (por agora só se imprimen en consola).

2. 16/10/2020 – 28/10/2020

Tras implantarse, construíuse unha API que procese as rutas do navegador e do sistema de ficheiros do equipo, ademais dos métodos para subir, descargar e eliminar ficheiros; máis crear e eliminar directorios. Mediante *Postman*, testáronse os diferentes métodos da API, comprobando que todo funciona.

1º prototipo de backend terminado!

3. 29/10/2020 – 17/11/2020

Para o frontend, utilizarase a librería de JS React. Mediante create-react-app iniciase o proxecto de cliente e xéranse os diferentes compoñentes da aplicación web.

Unha das librerías CSS máis coñecidas e usadas é Bootstrap. Existe un módulo chamado [react-bootstrap](#) que integra as clases CSS de Bootstrap en forma de compoñentes de React, permitindo maquetar a aplicación directamente dende JavaScript sen empregar as clases CSS con `<div>`'s como se fai xeralmente con Bootstrap.

Este compoñente ademais se complementa con [Bootswatch](#), un módulo que provee varios temas para Bootstrap, permitindo modificar a estética da app con só modificar a importación do tema. A futuro isto podería facerse cun botón ou selector na propia app, que alternase os temas dispoñibles, pero por agora só se pode cambiar dende o código.

4. 18/11/2020 – 20/11/2020

Empregouse unha base de datos MongoDB nun hosting de [mLab](#), o hosting oficial de MongoDB. Isto evidentemente non é o ideal por motivos de protección de datos (almacénanse as credenciais dos usuarios nun hosting externo e non nunha base de datos local) que aínda que mLab cumpre coa regulación Europea actual, tamén esixe ter conexión a Internet para funcionar, impedindo a implementación nunha rede interna sen saída a Internet.

Se o proxecto fose adiante e se levase a cabo a nivel empresarial, sería máis oportuno descargar MongoDB e manter unha base de datos en local nos servidores da empresa, solventando eses posibles inconvenientes. Pero como non é un requirimento, para facilitar o desenvolvemento do prototipo, optouse pola opción dun hosting por agora.

Un dos motivos de empregar MongoDB é que o nivel de complexidade precisado para almacenar os usuarios é mínimo. Só se require dun modelo de usuario que garde o nome, o correo, a contrasinal e a data de creación. [Mongoose](#) é unha ferramenta de modelado de obxectos para MongoDB. Con el xérase un modelo no servidor que será empregado no rexistro dos usuarios como esquema para almacenar a información en MongoDB.

5. 21/11/2020 – 28/11/2020

A autenticación foi a parte máis complexa de todas. Tras varios intentos con MySQL e MongoDB, óptase por Mongo pola súa simplicidade (MySQL resultaba innecesario para unha base de datos tan simple, que só almacena credenciais de usuarios).

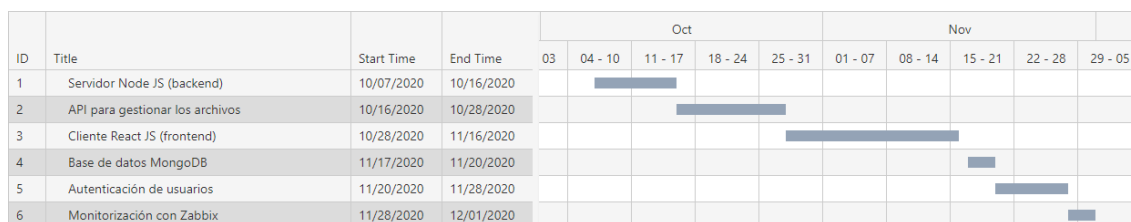
Tras buscar os módulos correspondentes de rexistro e inicio de sesión, procurando cumprir cos estándares de seguridade informática actuais no manexo das credenciais, deseñouse unha estrutura de compoñentes onde o inicio de sesión che redirixe polos distintos compoñentes da App comprobando en todo momento que estees autenticado, e impedindo o acceso ó compoñente que lista os contido do directorio do servidor se non está iniciada a sesión correctamente, redirixíndote á páxina principal se non.

Debido a que os compoñentes de autenticación fan de ‘porta de entrada’ ó contido da app, encapsulouse todo o feito ata agora nun directorio **/app/** dentro do módulo de autenticación (no apartado de desenvolvemento técnico explícase), de forma que os directorios se reestruturaron completamente, quedando os compoñentes de autenticación como directorios ‘pais’ dos feitos ata agora na app que traballa sobre a API, e non ó mesmo nivel como se plantexou inicialmente.

6. 29/11/2020 – 1/12/2020

Como complemento ao prototipo, para amosar un dos principais complementos do soporte, farase unha pequena proba con Zabbix onde se amosa como recolle a información do equipo monitorizado co axente, que no caso real da empresa sería o servidor no que estivese correndo MyCloudApp.

Diagrama de Gantt



5. Desenvolvemento técnico

Simplificando as fases do proxecto, detalladas no apartado anterior, quedaría:

1. Servidor Node (backend)
2. Cliente React (frontend)
3. Autenticación: Rexistro e inicio de sesión de usuarios
4. Monitorización do servidor con Zabbix

➤ **NOTA!:** A gran cantidade de código dos compoñentes empregados nesta aplicación fai inviable amosalo na súa totalidade neste traballo, polo que só se destacarán algunhas partes de exemplo. Todo o código, así como o seu historial de cambios (e polo tanto de avance ó longo das semás), está dispoñible para calquera [neste repositorio](#) do meu GitHub.

1. Servidor Node (backend)

O backend consta dunha API (Interface de programación de aplicacións) en NodeJS. Baséase en expressJS, un marco para NodeJS que facilita a creación destas APIs.

Unha API, moi resumidamente, é un conxunto de 'métodos' preestablecidos mediante os que as apps se comunican co servidor, facilitando a realización das accións contempladas na API respecto á se se implantasen de xeito manual sen ela.

O código do servidor agrúpase nun directorio de traballo **/src** (source).

Este, divídese en 3 subdirectorios:

```
lib
middlewares
routes
```

- **/lib:** Este directorio agrupa dúas ferramentas:
 - a. moveFile.js Almacenar arquivos recibidos nun directorio dado.
 - b. path.js Procesa a URL, devolvendo o directorio apropiado para cada sistema operativo (Windows ou Linux/MacOS).

- **/middlewares:** Funcións intermediarias para o control de erros.
Exemplo: DirExists.js. Erra se se intenta crear un directorio xa existente.
- **/routes:** Métodos da API, establecen que pode facer a aplicación:
 - a. content.js Carga contido (directorio ou ficheiro) no servidor Node
 - b. delete.js Elimina un ficheiro dun directorio
 - c. dir.js Crea un directorio
 - d. download.js Descarga un ficheiro
 - e. upload.js Almacena un ficheiro

➤ Principais limitacións actuais da API:

- Non permite descargar un directorio completo. Requiriría un novo método que comprima a carpeta nun .zip e o descargue.
- Non permite eliminar directorios. Require un novo método que comprobe se hai ficheiros dentro del, o notifique, o usuario acepte ou non dependendo do contido e un segundo método que o elimine recursivamente.

Fóra desas carpetas, no directorio **/src** atópanse 2 ficheiros máis:

1. storage.js Almacena nunha variable a ruta do equipo onde traballa o servidor.
2. index.js Arquivo de inicio do servidor.

Mostra do contido de **index.js**:

```
src > index.js > ...
1  const express = require('express')
2  const cors = require('cors')
3
4  const contentRouter = require('./routes/content')
5  const uploadRouter = require('./routes/upload')
6  const downloadRouter = require('./routes/download')
7  const deleteRouter = require('./routes/delete')
8  const dirRouter = require('./routes/dir')
9
10 const missed = require('./middlewares/missed')
11 const err = require('./middlewares/err')
12 const direxists = require('./middlewares/direxists')
13
14 const port = process.env.PORT || 5000
15 const app = express()
16
17 // Middlewares
18 app.use( express.json() )
19 app.use( cors() )
20
21 // Routes
22 app.get( '/', (req, res) => res.send( 'MyCloudApp API' ) )
23 app.use( '/content', contentRouter )
24 app.use( '/upload', uploadRouter )
25 app.use( '/download', downloadRouter )
26 app.use( '/delete', deleteRouter )
27 app.use( '/dir', dirRouter )
28
29 // Errors
30 app.use( missed )
31 app.use( direxists )
32 app.use( err )
33
34 // Server
35 app.listen( port, () => console.log( 'Server running on port', port ) )
```

Como se aprecia na captura, o funcionamento é moi esquemático:

1. Primeiro impórtanse os compoñentes precisados de *express*, os métodos da API e os middlewares para os erros.
2. Finalmente, asígnanse os métodos da API á ruta http correspondente, cárganse os middlewares e iníciase o servidor no porto 5000.

O último que faltaría para que funcione, é configurar un arquivo environment (**.env**) coa citada variable que se recolle en *storage.js*: a ruta de traballo do servidor, o directorio onde almacenará os arquivos.

```
.env
1 # - MYCLOUDAPP_STORAGE -
2 #
3 # This variable stores the root directory where the server gonna store the files
4 #
5 #
6 # Linux example: MYCLOUDAPP_STORAGE="/home/userexample/root-folder"
7 # Windows example: MYCLOUDAPP_STORAGE="C:\\Users\\userexample\\root-folder"
8 #
9 #
10 # !!!IMPORTANT!!!
11 # This is just an example file.
12 # After set the value, rename this file (or copy the content to a new one) as: .env
13
14
15
16 MYCLOUDAPP_STORAGE="C:\\Users\\Dawichii\\Desktop\\MyTestDir"
```

2. Cliente React (frontend)

React traballa por compoñentes, fragmentando a app en partes mínimas con funcións específicas, de forma que cando se recarga información nun deles, en vez de recargase toda a páxina, só se recarga o compoñente que variou, o que o fai moi eficiente en consumo de RAM e datos.

Para a app, prepáranse primeiro os compoñentes “complementarios” con funcións moi concretas que poden empregarse en calquera momento da app.

Alert.js	Renderiza unha alerta de React-Bootstrap.
Loading.js	Renderiza un spinner que fai a animación de carga.
FormModal.js	Bootstrap ten un compoñente chamado Modal que amosa unha pequena ventá emerxente interactiva (parecido ós <i>alert()</i> de JavaScript) no que podes amosar información. É comunmente empregado para formularios.

Eses 3 compoñentes se completan con 4 máis: 2 formularios e os 2 principais.

Os 2 formularios:

- MkDirForm.js** Un formulario que recolle un texto, que se empregará como nome do directorio creado polo método **/dir** da API
- FilesForm.js** Un formulario no que insertar ficheiros que se subirán ó servidor.

E por último, os 2 principais:

Dir.js

É o contedor da *app*. Onde se listan os directorios e ficheiros.

Dende el cárganse os **Dirents** (o último compoñente), que son unhas tarxetas rectangulares onde se visualiza información (un directorio, un ficheiro, a flecha de volver ó directorio anterior se estamos dentro de un subdirectorio, etc).

Tamén carga dous formularios na súa cabeceira facendo uso de FormModal: un para subir ficheiros (que chama ó *FilesForm*) e outro para crear novos directorios (que chama a *MkDirForm*). Ambas accións reciben como parámetro o directorio actual no Dir, polo que os novos directorios crearanse onde nos atopemos ó cargar o formulario; e o mesmo en canto á subida de ficheiros, almacenándose na nosa carpeta actual.

Dirent.js

Renderiza cada unha das tarxetas dos directorios e ficheiros, distinguindo entre eles e asignando iconos segundo corresponda. Tamén asigna os iconos de descarga e borrado, executando as funcións correspondentes ao facer clic.

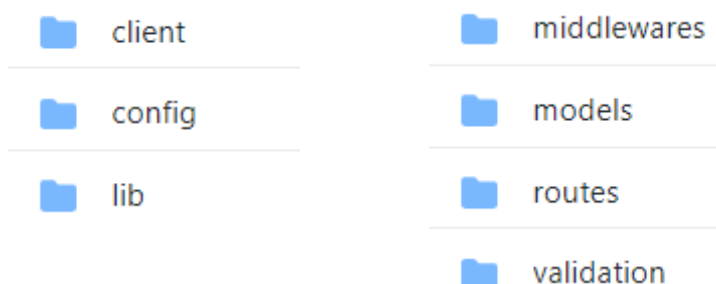
- **NOTA:** A estrutura orixinal na que se organizaban todos estes compoñentes modificouse coa implantación da autenticación no frontend. Para evitar confusións, explicando a organización que había neste momento e a nova no seguinte apartado; comentarei a organización destes compoñentes máis adiante, canto estean integrados co rexistro e inicio de sesión.

3. Autenticación: Rexistro e inicio de sesión de usuarios

Esta foi sen duda a parte máis complexa do traballo. Precisaba tantas modificacións tanto a nivel de backend como de frontend, que finalmente houbo que reestruturar todo o proxecto dende cero para seguir correctamente os estándares actuais e non xerar un entramado de importacións entre carpetas imposible de manter.

Tras moitos tutoriais, cursos e documentacións; optouse por unha estrutura diferente á plantexada inicialmente, de forma que o frontend está integrado dentro do servidor nun directorio **/client/** e non separado ó mesmo nivel como se pensou inicialmente.

Reestruturación do backend. Estructura actual no directorio raíz do servidor:



- Directorios que non se modificaron: **lib/** e **middlewares/**

Ambos foron xa explicados e non precisan máis presentación.

- Directorio con novo contido: **routes/**

Unha pequena mención para o directorio de rutas, onde se cargou un subdirectorio **routes/api/** cun só arquivo: **users.js**. Este arquivo valida a creación de usuarios e o inicio de sesión, en base aos estándares de seguridade actuais cos compoñentes de validación máis empregados.

- Novos directorios: **client/**, **config/**, **models/** e **validation/**

models/

Contén **User.js**, o arquivo co esquema de mongoose que fai de plantilla no rexistro de usuarios en MongoDB.

Esquema empregado para rexistrar os usuarios -->

```
1  const mongoose = require("mongoose")
2  const Schema = mongoose.Schema
3
4  // Create Schema - User model
5  const UserSchema = new Schema({
6    name: {
7      type: String,
8      required: true
9    },
10   email: {
11     type: String,
12     required: true
13   },
14   password: {
15     type: String,
16     required: true
17   },
18   date: {
19     type: Date,
20     default: Date.now
21   }
22 })
```

config/	Contén 3 arquivos con configuracións para o servidor:
<i>keys.js</i>	Contén a configuración da conexión á BD de MongoDB.
<i>storage.js</i>	Foi explicado anteriormente. Atopábase na raíz no primeiro backend, pero agora alóxase neste directorio.
<i>passport.js</i>	Emprega un módulo chamado passport . É un middleware para autenticacións con Node que facilita e securiza a implementación de autenticación con servidores.
validation/	Contén 2 ficheiros login.js e register.js que validan a nivel de servidor os campos dos formularios de rexistro e inicio de sesión, impedindo ataques que modifiquen a validación a nivel de cliente que fan os formularios de React.
client/	Almacena todo o frontend da aplicación. Contén unha app React completa polo que merece unha sección a parte.

Frontend React - client/

Debido ó funcionamento dos módulos de autenticación para React creouse un proxecto novo dende 0, no que máis adiante nun directorio concreto, importaranse todos os compoñentes creados previamente na sección de Frontend.

➤ Explicación rápida: **Redux**.

En React, existían dous tipos de compoñentes: *functional* components e *class* components. Os segundos están baseados na estrutura típica de programación orientada a obxectos por clases, mentres que os primeiros baseábanse en funcións estándar de JavaScript. Para solventar as limitacións dos primeiros, creáronse unhas ferramentas chamadas **Hooks**, que proveen de funcionalidades extras a estes compoñentes baseados en funcións.

Enriba destes Hooks, a comunidade volcou a desenvolver novas tecnoloxías e funcionalidades. Unha de elas é Redux. Esta tecnoloxía controla o estado dos Hooks con dous métodos: As **actions** e os **reducers**.

Actions: describen unha modificación nun compoñente.

Reducers: describen a modificación do estado de ese compoñente.

Dende hai un par de anos, isto comeza a ser o estándar en React, e na propia documentación oficial anima a empregar estes novos métodos en vez da programación orientada a obxectos clásica, xa que se integra mellor co JavaScript nativo.

É por isto que me esforcei en aprender esta tecnoloxía, para poder empregalas no meu cliente e gañar un coñecemento tan recente e relevante. Aínda non comprendo os conceptos máis técnicos dos reducers, pero atópome cómodo co uso de Hooks.

Presentada a tecnoloxía na que se basea o novo cliente, procedo a amosalo:

Ao ser unha react-app, o directorio de traballo é de novo **src/**.

- Fóra del, só precisa mencionar o ficheiro de environment (.env), que aloxa as distintas configuracións para a URL empregada por React e un pequeno arreglo.

Contido de **/client/src/** :

actions	utils
api	App.js
components	index.js
reducers	serviceWorker.js
test	store.js

Como calquera aplicación de React, a base é o **index.js**. Nel impórtanse os estilos de Bootstrap e cárgase o compoñente **App.js**.

Nese nivel hai 2 arquivos máis: **serviceWorker.js** e **store.js**

serviceWorker.js É un arquivo predefinido da propia React que mellora lixeiramente o rendemento da app. Aínda que a miña intención foi usalo, existen incompatibilidades con algunhas dependencias que non me permiten empregalo de momento, polo que o seu uso no index.js permanece comentado. En futuras actualizacións das dependencias, se se resolvesen os conflitos, podería habilitarse.

store.js Almacena o *rootReducer* Redux (a base para os demais reducers)

Subdirectorios de **client/** :

client/reducers/ Almacena os dous reducers da app, **authReducer** e **errorReducer**. Un controla se o usuario está autenticado e o outro os erros. Un terceiro arquivo, index.js, os combina ambos e os exporta xuntos nun obxecto chamado *combineReducers({})*

client/actions/ Contén as funcións que modifican o estado dos reducers.
Exemplo: *loginUser()* e *logoutUser()*.

client/api/ Contén un único ficheiro **api.js** que mediante o módulo **Axios** fai as peticións http ó servidor. Frameworks máis complexos como Vue ou Angular dispoñen dun cliente http integrado, pero React precisa deste módulo para facelo.

client/utills/ Contén un único ficheiro *setAuthToken.js* que aplica o token a Axios se inicia a sesión correctamente ou o elimina se non.

client/components/ Aquí atópanse os componentes que renderizan contido na app.

Compoñentes da app:

client/components/layout/

Navbar.js Barra de navegación da app.

Landing.js Compoñente base ao chegar á app. Enlaza a *Register.js* e *Login.js*

client/components/auth/

Register.js Formulario de rexistro

Login.js Formulario de inicio de sesión.

client/components/private-route/

PrivateRoute.js Comproba se estás autenticado, evitando que accedas ó contido da app e redirixíndote a */login* se non o estás

client/components/dashboard/

Dashboard.js Base da parte só accesible se estás autenticado. É un compoñente de paso, onde compróbase a autenticación e en caso de ser válida, se redirixeche ó compoñente **Dir** (o compoñente base do frontend do subapartado anterior). De forma que aínda que ti vexas o compoñente **Dir**, éste queda encapsulado polo **Dashboard**, que en todo momento comprobará a autenticación, o que fai que toda a App quede segura ante ataques escribindo rutas na URL.

- Este último foi o motivo de toda a reestructuración. Como o *Dashboard debía conter o Dir*, os imports quedaban completamente enrevesados se o **Dashboard** fose un subdirectorio do **Dir**, non tería sentido.

client/components/app/

Neste directorio app atópase todos os compoñentes que xa se viron no primeiro frontend. Os dous formularios *FilesForm* e *MkdirForm* agrupáronse baixo o directorio de *forms/*, o resto queda igual.

Con isto, finalmente, remátase o prototipo! 🍌

Ante calquera duda coa organización ou o código, recomendo de novo visitar o repositorio de [GitHub do proxecto](#), onde atoparedes todo o amosado aquí.

4. Compilación dunha app Node en instaladores nativos executables

Co produto rematado, o interesante á hora de desplegar o software sería contar cos instaladores clásicos para os distintos sistemas operativos, e non estar distribuindo o código base, que require instalación extras de Node e as súas dependencias.

Para isto existe un módulo chamado [pkg](#) que compila o código xunto con Node e todas as dependencias requeridas polo proxecto nos 3 instaladores de Windows, Linux e MacOS.





```
PS C:\Users\Dawichii\Documents\GitHub\cloud-app> npm install -g pkg
npm WARN deprecated request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/3142
npm WARN deprecated har-validator@5.1.5: this library is no longer supported
C:\Users\Dawichii\AppData\Roaming\npm\pkg -> C:\Users\Dawichii\AppData\Roaming\npm\node_modules\pkg\lib-es5\bin.js
+ pkg@4.4.9
added 124 packages from 155 contributors in 7.728s
PS C:\Users\Dawichii\Documents\GitHub\cloud-app> pkg server.js
> pkg@4.4.9
> Targets not specified. Assuming:
node12-linux-x64, node12-macos-x64, node12-win-x64
> Fetching base Node.js binaries to PKG_CACHE_PATH
  fetched-v12.18.1-linux-x64 [=====] 100%
  fetched-v12.18.1-win-x64   [=====] 100%
  fetched-v12.18.1-macos-x64 [=====] 100%
PS C:\Users\Dawichii\Documents\GitHub\cloud-app> 
```

Como se ve na imaxe, de xeito sinxelo e tras instalar globalmente o paquete pkg, chámase ó comando pasándolle o arquivo base do servidor base (server.js neste caso).

Pódese pasar un parámetro para especificar cal dos 3 instaladores queremos, pero se non se lle pasa crea os 3 tipos por defecto.

É una gran vantaxe en comodidade á hora de realizar a instalación, pero incrementa moi enormemente o tamaño do servidor, que pasa duns poucos KB a 60MB de tamaño. Isto débese a que o instalador leva dentro tanto a instalación de NodeJS como todos os paquetes que ten a app como dependencias, que ocupan moito espazo.

De xeito práctico, consiste en empacar todo o software requerido para que a app funcione xunto coa app, distribuíndoo nun so arquivo en formato executable para cada S.O.

 server.js	29/11/2020 14:05	Archivo JavaScript	2 KB
 server-linux	02/12/2020 16:23	Archivo	56.539 KB
 server-macos	02/12/2020 16:23	Archivo	56.651 KB
 server-win.exe	02/12/2020 16:23	Aplicación	52.474 KB

Na imaxe anterior pode apreciarse os 3 arquivos creados pola saída do comando pkg e o seu tamaño.

5. Monitorización con Zabbix

Farase unha pequena implementación para amosar o resultado final dun monitoreo real.

1. Instalaríase o software [Zabbix](#) nun servidor noso, dende onde controlaríamos todos os nosos clientes.
2. Unha vez instalado e configurado, instalaríase no servidor do cliente (onde corre MyCloudApp) o [zabbix-agent](#). A configuración deste axente é realmente sinxela, só se precisa da IP do noso Zabbix-Server e asignar o PSK co que se cifrarán as comunicacións.

Configurado o axente, accédese ó panel web do Zabbix e créase un novo host, precisando o hostname, IP e PSK do equipo que ten o axente.

Agora, a parte importante, é decidir como sensorizalo. Zabbix funciona por 'items' (sensores) que detectan diferentes valores do sistema, e uns 'triggers' (disparadores) que devolven unha alerta cando o valor supera ou baixa dun umbral establecido no trigger. Estes sensores agrúpanse nunhas plantillas categorizadas segundo o tipo de información que recollen.

Exemplo:

Module CPU Recolle valores sobre a CPU (carga, latencias, nº núcleos...)

Module ICMP Realiza un ping cada 30s e devolve unha alerta se falla 3 veces

Con todo configurado, e esas plantillas aplicadas, se por exemplo configuramos un windows para que non conteste os pings (e así simular unha caída dun servidor) zabbix alertaríanos con este erro:

Estado	Información	Equipo	Problema • Gravedad	Duración
PROBLEMA		Test Host	Unavailable by ICMP ping	27s

E unha vez solventado (neste test, contestando de novo os pings; nunha situación real, arranxando o problema e levantando de novo o servidor) amosaríase unha notificación de resolto:

Estado	Información	Equipo	Problema • Gravedad	Duración
21:17:30 RESUELTO		Test Host	Unavailable by ICMP ping	1m

6. Prototipo

Pantalla de bienvenida (Landing)



Here you will be able to store all your files organized as you want!

Do you already have an account?

Log In

If not, create one. It's free!

Register

Pantalla de rexistro



[← Back to home](#)

Register below


Already have an account? [Log in](#)


 Name

Your name

@ Email

name@example.com

 Password

 Confirm Password

Sign up

Pantalla de inicio de sesión



[← Back to home](#)

Login below

Don't have an account? [Register](#)

@ Email

🔑 Password

Login

Sesión Iniciada! Neste caso cunha conta de probas chamada "John".

Como aínda non está implementado os directorios pessoais para cada conta, todos os usuarios accede a ese mesmo directorio con ese contido, pero a proba de que a app distingue entre usuarios é ese username 'John' visualizado no compoñente, que se extrae do state que almacena as credenciais.



Hello, John! 🗝

Upload Files 📁

Create Directory 📁

📁 hola test

📁 test

📁 test2

📄 abstract.png 📄 🗑

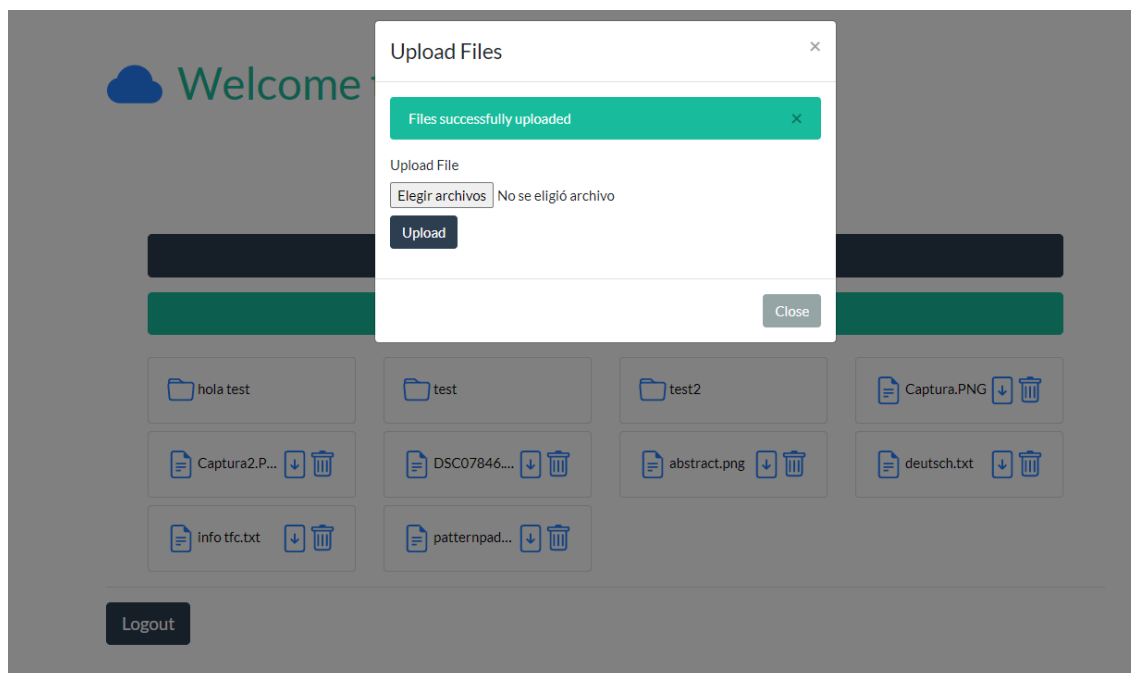
📄 deutsch.txt 📄 🗑

📄 info tfc.txt 📄 🗑

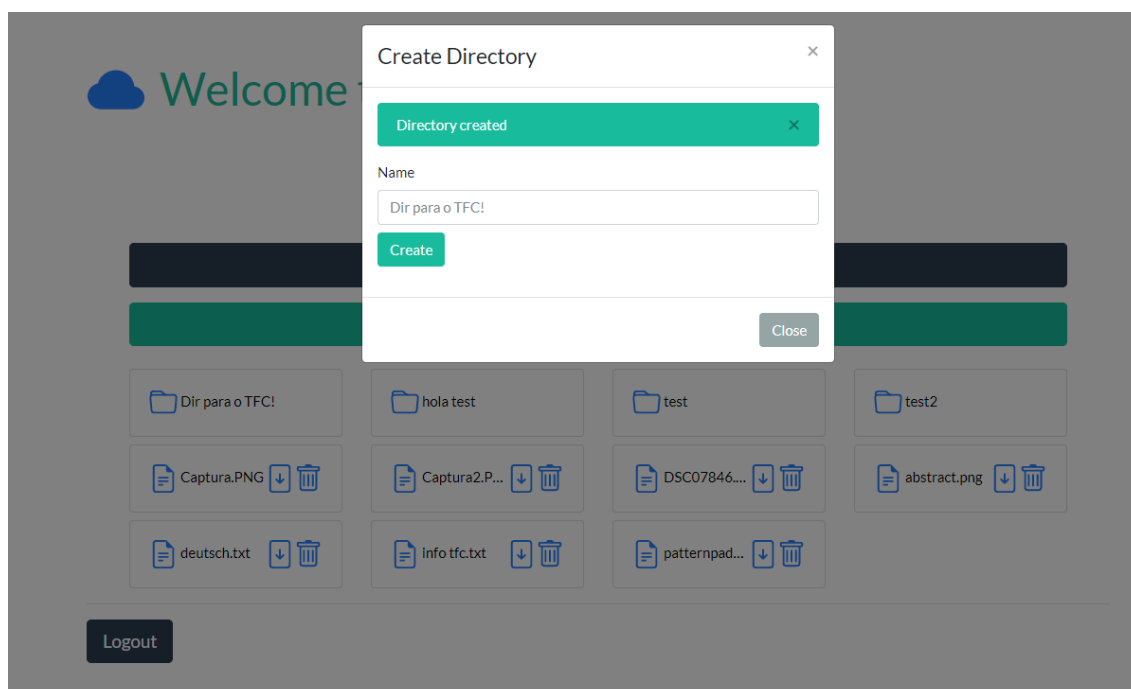
📄 patternpad... 📄 🗑

Logout

Carga de novos ficheiros (agregáronse 3 ao fondo, é instantáneo)




Creación de directorios




Ao ser deseñado con Bootstrap 4, non require máis esforzo extra para facer unha versión móbil, esta xa funciona!



Hello, John ! 

Upload Files 


Create Directory 

 Dir para o TFC!

 hola test

 test

 test2

 Captura.PNG



7. Dificultades atopadas

Inicialmente costou comezar, xa que son unhas tecnoloxías que non dominaba moito, pero propúxenme o reto de facer o TFC con elas e tras moito esforzo, conseguiuase!

As grandes pedras no camiño do desenvolvemento desta app foron sobre todo a autenticación. Esta característica precisa dunha inxente cantidade de módulos e dependencias para evitar ocos de seguridade, non o imaxinaba tan denso. Costou moito atopar unha integración que se levase ben co resto da app.

Fixen probas con moitos módulos de rexisto e login distintos, e case todos nalgún punto daban erros coa app xa creada. De feito, o que finalmente empreguei, foi descartado nun inicio pola cantidade de erros que xeraba, pero era o máis limpo a nivel de funcionamento do código e por tanto sinxelo de replicar (comparado aos baseados en MySQL que incrementan a complexidade), polo que tras moito insistir rematei reestruturando a app completa no resultado final, e tras arranxar un par de erros coas dependencias reducindo as versións a outras máis estables manualmente, funcionou!

Aínda quedan algúns bugs e limitacións que foron mencionándose no documento e explicarei máis detalladamente no seguinte apartado, pero o resultado actual acadado non podía nin imaxinalo fai unhas semanas, cando me pelexaba precisamente co citado tema da autenticación.

Outra das grandes dificultades que merece a pena mencionar foi o uso de *functional components* e *redux*. É un concepto que descubrín fai uns meses e fascinoume, polo que estaba decido a que a miña app empregase esa estrutura de código. Foi complexo de entender a nivel funcional, pero unha vez comprendido é unha ferramenta incríbel para programar, da que espero sacar moito proveito nun futuro.

8. Conclusións

O proxecto foi unha montaña rusa. Dende o principio atopábame moi convencido coa idea, pero nas primeiras fases do desenvolvemento, comecei a desesperarme ao verme superado pola envergadura do mesmo e o descoñecemento sobre algunhas tecnoloxías precisadas por el.

Pouco a pouco, fun sacando adiante diferentes características, sempre cunha montaña de erros acumulados por resolver coas dependencias e o código. Eventualmente conseguín acadar unha app funcional, e pasou a ser un tema de parchear erros e resolver conflitos das dependencias. Tras estas semás pulíndoo, atópome moi orgulloso do resultado, aínda que lamentando que non se poidera acadar algunhas características que desexaba implantar como os directorios persoais para cada usuario, pero que por razóns de tempo e esforzo, foi inabarcable. Con sorte, co proxecto en GitHub, seguirei desenvolvendoo cando dispoña de tempo para traballar en él con quen se preste a colaborar.

A experiencia

A pesar do estrés, a experiencia foi realmente positiva xa que vexo agora unhas capacidades de resolución de erros na implementación de código que non tiña antes. É incríbel os múltiples coñecementos sobre o stack MERN (*Mongo + Express + React + Node*) que me levo, especialmente de JavaScript con React. Teño moitas ganas de explorar máis ese software e probar a súa versión para dispositivos móbiles React Native.

Pero por riba de todo, o que máis cambiou foi o tema de git e o control de versións. Levo 2 meses facendo commits diarios do proxecto cos pequenos cambios, empregando as branch para as diferentes versións do proxecto cando se facían probas coa autenticación, e familiarizándome con npm e as opcións de software libre para programadores, e ese coñecemento non ten prezo.

Como curiosidade, ata animeime a facer unha 'tarxeta' cos meus datos publicada como paquete npm [aquí](#).

Citar respecto ó Zabbix, que a idea de empregar ese software débese a que nas prácticas o empreguei en produción e pareceume unha ferramenta moi útil, e que calquera empresa informática que de soporte pode implementala dun xeito moi sinxelo.

Posibles melloras

Son múltiples as cousas que quedaron polo camiño no descenso a realidade respecto á idea orixinal. Pero as teño ben localizadas e analizadas, e con un pouco máis de tempo poderíase haber acadado algún dos seguintes puntos. Lamentablemente ó ser só unha persoa e coas prácticas polo medio, o tempo escaseou e por iso estes puntos permaneceron no tinteiro.

Diferenciarei entre 3 tipos de cousas que quedaron por facer:

Bugs - Erros que non se chegaron a solventar

1. Eliminación de arquivos

Os ficheiros, tras o clic no icono de papeleira, elimínanse correctamente e de xeito instantáneo no servidor (nos logs de Node compróbase), pero o cliente React bloquea a sesión actual do navegador (nalgún foro lin que pode ser problema de Axios) polo que a app se bloquea. Se abres unha pestana nova coa mesma URL podes continuar empregando a app perfectamente, coa túa sesión aínda aberta, e verás o ficheiro borrado e todo saíu ben. Pero a sesión na pestana onde te atopabas, queda completamente bloqueada.

2. Subida dun único arquivo

Mentres que subir varios ficheiros é instantáneo e resólvese sen erros, se seleccionas un único arquivo na subida, a app bloquéase. Teño localizado o erro na dependencia *express-fileupload*, que xenera conflito con outra pero ata agora non conseguín solventalo.

Funcionalidades básicas da app que non se implementaron

1. Eliminación de directorios

A miña idea era empregar un método na API similar ó que elimina ficheiros, pero debido á posibilidade dos directorios de conter ficheiros ou non, esixe maior complexidade. Debido as complicacións que sufrín con ese método, rematou quedando pendente por falta de tempo.

Posibles melloras nunha futura versión empresarial do produto

1. Directorios persoais diferentes para cada usuario

Actualmente a app traballa sobre un directorio común */content/*. Habería que crear subdirectorios nel co nome de usuario de forma que se xerasen */content/david/* e */content/pablo/* ao iniciar sesión, establecendo co compoñente *Private-Route.js* que só se tivese acceso ao directorio igual ó teu nome de usuario.

2. Descargar carpetas

Existen módulos que comprimen un directorio xenerando un .zip. Tras iso, empregaríase o método *download* xa creado da API para descargalo como calquera outro ficheiro e despois eliminaríase do servidor, quedando todo como estaba pero cunha carpeta comprimida descargada.

3. Mover carpetas

JavaScript ten unha función de arrastre co rato *DragDrop()*. Con ela poderíase establecer que cando alguén arrastrase o icono dun ficheiro ou carpeta a un directorio, fixérase algo similar ao *moveFile* como un "*moveDir*" a ese directorio, permitindo reorganizar os contidos dende a app co rato.

9. Bibliografía, referencias e recursos utilizados

Quería poder resumir esta sección, xa que a inxente cantidade de documentación precisada para este proxecto ocupa case dúas páxinas, pero durante estes dous fun apuntando nun arquivo de texto todas as documentacións, webs, tutoriais e repositorios que consultaba. As primeiras organizadas por categoría podo situar máis ou menos o momento de acceso xa que foron en orden de necesidade segundo avanzaba a app, pero moitos repositorios e artigos tornáronse unha constante no meu escritorio do PC, sendo consultados moitas veces o longo de varias semás, polo que categorizalos por fecha é case imposible.

Visual Studio – 30/09/2020

https://es.wikipedia.org/wiki/Visual_Studio_Code
<https://code.visualstudio.com>

NPM - 2/10/2020

<https://es.wikipedia.org/wiki/Npm>
<https://www.npmjs.com/>

Node JS -2/10/2020

<https://es.wikipedia.org/wiki/Node.js>
<https://nodejs.org/es/>

Postman - 4/10/2020

https://es.wikipedia.org/wiki/The_Postman
<https://www.postman.com/company/about-postman/>

React – 29/10/2020

<https://es.wikipedia.org/wiki/React>
<https://es.reactjs.org/>

Bootstrap – 29/10/2020

[https://es.wikipedia.org/wiki/Bootstrap_\(framework\)](https://es.wikipedia.org/wiki/Bootstrap_(framework))
<https://getbootstrap.com/>
<https://www.w3schools.com/bootstrap4/>
<https://react-bootstrap.github.io/>
<https://icons.getbootstrap.com/>
<https://bootswatch.com/>

W3.CSS – 25/10/2020

<https://www.w3schools.com/w3css/default.asp>

Mongo DB – 18/11/2020

<https://es.wikipedia.org/wiki/MongoDB>
<https://www.mongodb.com/es>
<https://www.mongodb.com/products/compass>
<https://mlab.com/>
<https://mongoosejs.com/>
<https://robomongo.org/>
<https://www.genbeta.com/desarrollo/robomongo-y-mongodb>

Zabbix - 1/12/2020

<https://es.wikipedia.org/wiki/Zabbix>
<https://www.zabbix.com/>

Github Usuarios – Entre o 7/10/2020 e o 28/11/2020

<https://github.com/anmol098>
<https://github.com/guilyx>
<https://github.com/ananyaneogi>
<https://github.com/anuraghazra>
<https://github.com/antoniosarosi>
<https://github.com/illuspas/Node-Media-Server>
<https://github.com/carbuero>
<https://github.com/kamranahmedse>

Especial mención polos módulos de autenticación:

<https://github.com/bnb>
<https://github.com/rishipr>
<https://github.com/PlankCipher>

Especial mención polos coñecementos de React/Redux:

<https://github.com/Klerith>

Principais repositorios máis consultados – Entre o 29/10/2020 e o 28/11/2020

<https://github.com/reactjs/es.reactjs.org>
https://github.com/PlankCipher/iC0dE_Magazine_Code_Samples
<https://github.com/devicons/devicon>
<https://github.com/rishipr/mern-auth>

Profesores destacados dos cursos online feitos

<https://www.udemy.com/user/550c38655ec11/>
<https://www.udemy.com/user/nicolas-schurmann/>
<https://www.udemy.com/user/victor-robles-2/>

Tutoriais seguidos de distintas webs, neste orde aproximado

<https://www.digitalocean.com/community/tutorials/how-to-create-a-web-server-in-node-js-with-the-http>
<https://www.agiliacenter.com/servidor-http-basico-con-node-js/>
<https://medium.com/javascript-espa%C3%B1ol/configuraci%C3%B3n-de-nodemon-en-un-servidor-node-js>
https://developer.mozilla.org/es/docs/Learn/Server-side/Express_Nodejs/Introduction
<https://medium.com/edureka/expressjs-tutorial-795ad6e65ab3>
<https://medium.com/@haybams/build-a-restful-api-with-node-js-and-express-js-d7e59c7a3dfb>
<https://medium.com/@asfo/desarrollando-una-sencilla-api-rest-con-nodejs-y-express-cab0813f7e4b>
<https://es.reactjs.org/tutorial/tutorial.html>
<https://www.w3schools.com/react/>
<https://medium.com/edge-coders/all-the-fundamental-react-js-concepts-jammed-into-this-single-medium-article-c83f9b53eac2>
<https://medium.com/better-programming/getting-started-with-react-js-part-1-59598ef17780>
<https://medium.com/mobile-web-dev/learn-react-by-building-a-to-do-app-react-functionalities-explained-74f466e9396>
<https://medium.com/@beaucarnes/learn-the-mern-stack-by-building-an-exercise-tracker-mern-tutorial-59c13c1237a1>
<https://medium.com/codingthesmartway-com-blog/the-mern-stack-tutorial-building-a-react-crud-application-from-start-to-finish-part-2-637f337e5d61>
<https://blog.bitsrc.io/build-a-login-auth-app-with-mern-stack-part-1-c405048e3669>
<https://dev.to/andrewbaisden/creating-mern-stack-applications-2020-4a44>
<https://www.digitalocean.com/community/tutorials/getting-started-with-the-mern-stack>
<https://medium.com/technoetics/create-basic-login-forms-using-react-js-hooks-and-bootstrap-2ae36c15e551>
<https://dev.to/dipakkr/implementing-authentication-in-nodejs-with-express-and-jwt-codelab-1-j5i>
<https://coderwhodreams.com/blog/creating-private-routes-and-handling-session-in-react-js/>

<https://codingthesmartway.com/the-mern-stack-tutorial-building-a-react-crud-application-from-start-to-finish-part-3/>
<https://openwebinars.net/blog/creando-un-crud-con-javascript-construyendo-el-backend-basado-en-una-api/>
<https://serverless-stack.com/chapters/what-does-this-guide-cover.html>
<https://medium.com/technoetics/create-basic-login-forms-using-react-js-hooks-and-bootstrap-2ae36c15e551>
<https://dev.to/dipakkr/implementing-authentication-in-nodejs-with-express-and-jwt-codelab-1-j5i>
<https://coderwhodreams.com/blog/creating-private-routes-and-handling-session-in-react-js/>
<https://coderrocketfuel.com/article/how-to-delete-a-file-from-a-directory-with-node-js>
<https://icodemag.com/how-to-build-a-login-register-app-with-the-mern-stack-part-4-setting-up-frontend-routes/>

10. Anexos

Iconos

Os iconos empregados na sección 3 proceden de [Devicon](#), un repositorio de GitHub de iconos sobre programación; e de [iconos8](#), unha web de iconos gratuítos.

Diagrama de Gantt

Xerado con <https://online.visual-paradigm.com/es/diagrams/features/gantt-chart-tool/>