

# My-Drive

CLOUD solutions  
Files storage and sharing





## Content:

---

-	Intro	3
1.	Presentation	4
1.1.	Authorization	5
1.2.	Files	6
1.3.	Sharing	7
2.	Security	7
3.	Scalability	7
4.	Installation & Implementation	8
5.	Usage costs	9
6	Terms and conditions	10
7	Technical support	11
8.	WEB Services ( API ), applications development	11
8.1.	Web services, authorization service (API)	12
8.2.	Web Services, the file server (API)	17
8.3.	Programmable web services support	22
8.4.	Web Server for web apps and presentation web pages.	22

---

March 2021

**webdo.com**

Copyright Q-Bis Consult S.R.L.

[office@qbis.ro](mailto:office@qbis.ro)

[www.qbis.ro](http://www.qbis.ro)



## Intro

My-Drive offer file storage and file access solutions from different locations using heterogeneous resources. Web based cross platform and cross devices client application. VPN or guaranteed Internet connection are NOT required.

The file storage uses CLOUD solutions from Amazon AWS offered for different geographic regions for more speed. For Europe there are data centers in Frankfurt, Paris, Dublin and Milan. For Europe and GDPR concerns, the solution covers all required security requirements and more.

Files are stored using the AWS S3 object storage technology and a private CLOUD zone. The storage and the file transfer is protected using high encryption.

The Amazon AWS offers a virtual unlimited storage space; pricing is based on monthly usage. There is a price per GB per month of storage and a price for files download, upload is free of charges. All files are stored encrypted using the AES 256 library and the data transfer is protected with SSL certificates by the HTTPS protocol.

The solution can scale horizontally with the number of API servers. It is a hybrid solution based on CLOUD resources and VPC (servers) for access requests. One VPC server offer access to minimum one hundred users based on VPC capabilities and average loading. Low cost EC2 VPC servers are used; scalability is based on servers' number rather on one server power.

My-Drive offer users and users group management. The Web Application offers support for files sharing between users or users groups.

Web software development support is offered thru the system API access points:

- Authorization
- File access
- Web server

Used services implements open standards, can be integrated with third party software applications and offer support for new development.



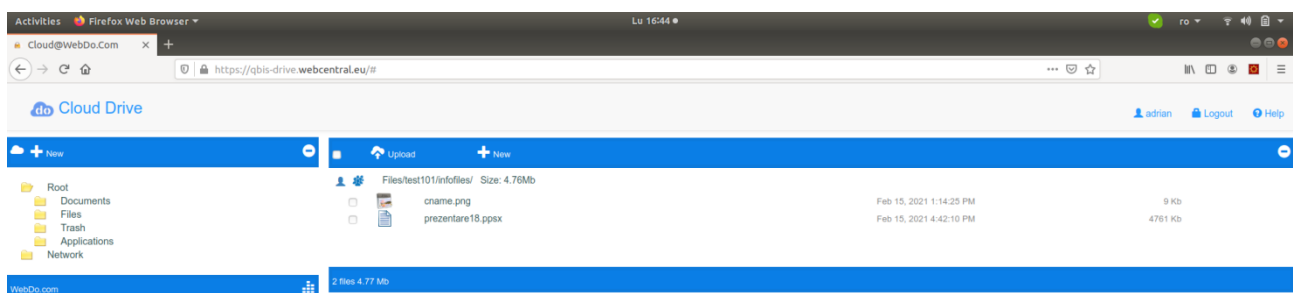
## 1. Presentation

The solution uses CLOUD and infrastructure services from Amazon AWS to implement easy to be use services for remote working; it is focused on security and scalability offering simple web interfaces and integration with third-party solutions based on open standards.

Private CLOUD from Amazon AWS, the S3 object storage, it is used for files storage. Files are encrypted all the time, during transfer and at rest. Encryption / decryption process is created on data transfer stream, data transfer from the AWS infrastructure uses encrypted transfer protocols (HTTPS).

The S3 solution it is launched by Amazon AWS from 2006. The services we use for My-Drive are developed from 2015 being used for highly scalable web applications. The actual release it is optimized to be used by organizations and companies of any size.

The web based application uses responsive pages that mimic “File Explorer” applications. There are options to manage folders and files. Fixed folders (like Documents) and sub folders of them are used.



The “Network” folder offers access to folders that are shared to the user directly or to a user group from which the current user belong. The right part of the UI manages the folders and its tree structure, the right part manages the files list per folder.

Information related to the system usage is presented. The user can change its password from the UI. The UI is created to be easily used by offering access to necessary functionality.

More files can be uploaded simultaneous, however a large number can overload the local Internet connection. The web-app tries several times to upload the files in case of errors.

Each folder can be shared to another user or to a user group. Shared folders access rights are allocated, there can be “read” or “write” access rights (write include read!).

Related details are presented into the help files attached to the web UI.  
(Ex: [HelpFile](#))



## 1.1. Authorization

The Authorization is built using modern open standards in order to offer compatibility with currently developed systems and software applications.

In order to be highly scalable and in respect of RESTful web services standards, the JWT (Json Web Token) authorization model is used.

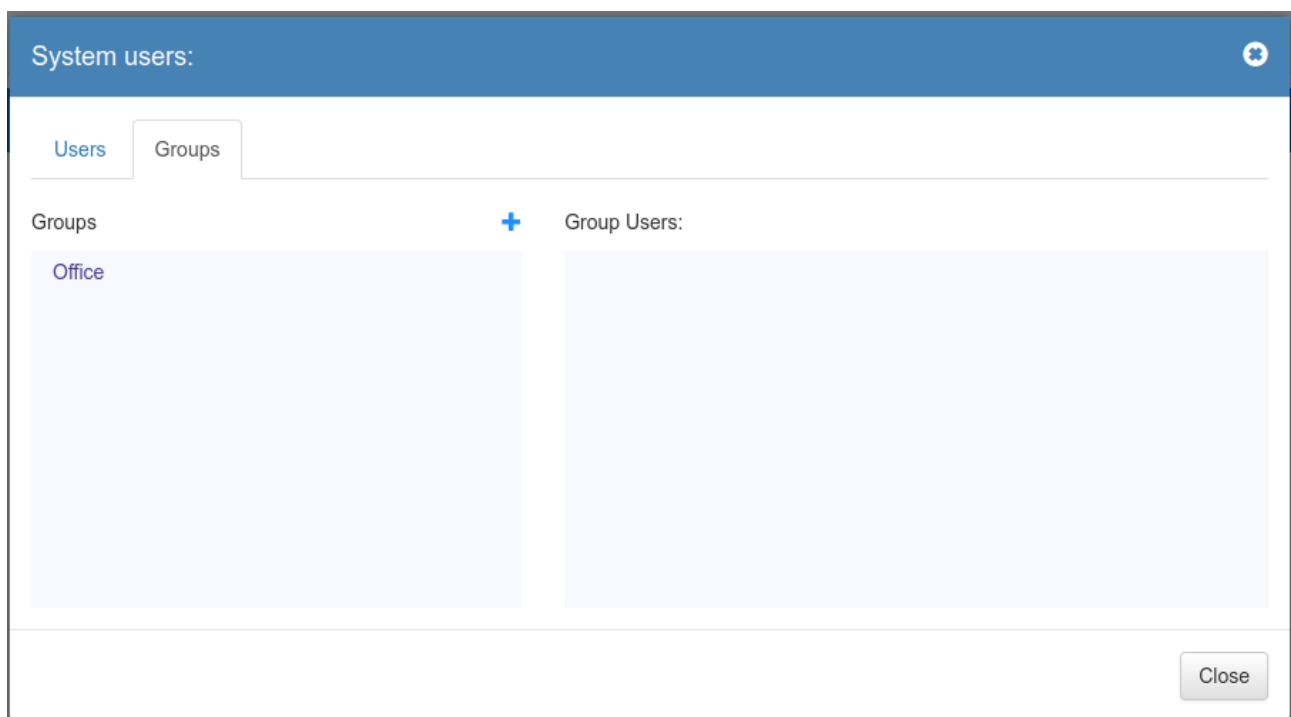
NO working sessions are used; the JWT authorization model it is used for services that do not require a stable connection between a client and a specific server, each request is solved by the first available server thru a balanced service model.

The Authorization service offers all required interfaces (API) for the system and it is prepared to offer services to integrate different web apps that require authorization.

All required information uses high encryption at rest. Passwords are never stored, access rights information are stored in a separate location per user using CLOUD resources. Server's number can scale up and down in order to offer best services at one moment in time. The system can handle billions authorization requests having same response time with proper installation

The authorization services is protected against brute force type of attack and can be protected against DOS attacks using a balancing solution from AWS.

Users can be part of users groups, other applications can use the users groups structures to offer different services accordingly.



Administration of the users and users groups is done from the My-Drive web application using the “administrator” account. Support for users and users groups management is



offered. One user can be suspended, the interface is easy to be used, company organization scheme is required to allocate users rights properly. Users list, groups list and users links into groups are managed here.

One user can change its password from the UI, the system administrator can suspend one user access and can change passwords. One account can be opened with its password only, there is no supper user or assimilated. However one account can be accessed by the administrator by changing the account password, in wich case the end user will notice the password change.

The authorization service is independent to the rest required services. The UI to manage the users and groups is implemented into My-Drive web application.

## **1.2. Files**

Files are the object of this system. Being uploaded for storage or for remote locations access, My-Drive offers the easy secured access solution for your files.

Services offers files management and support to edit text based files. Syntax highlight editors are used for text based files like JS, HTML, CSS and other types. The road map includes file editors for Office document types XLSX and DOCX.

File uploading speed is related to the Internet connection for each user. The used resources spent per user server side are very low. In case the number of users is high and the system is overloaded, a scalable implementation should be considered. It can scale balancing the services requests to more infrastructure EC2 VPC servers. One low-end EC2 VPC can offer services to hundred users, it depends on usage also.

Files are encrypted at rest using AES 256 algorithm, for that reason the in file search cannot be implemented without a compromise in security (usage of unencrypted indexes). However a search can be made on file names.

The storage is a private CLOUD AWS S3 bucket, the storage uses object storage technology. It offers scalability for services in terms of availability, speed and storage limits, however it comes also with some limitations.

Encrypted files cannot be used outside the system even if somehow there is access to the physical storage disks of the S3 infrastructure.

Files are organized into folders and subfolders. There are implicit folders like "Documents" and subfolders of them.



The AWS S3 accept uploads of files up to 7Tb, however because such operation it is very time and resources consuming it is not feasible at this time, the uploading size is software limited to 5Gb per file. Generally Office files are small, presentations PPTX being larger, however having less than 50Mb.



### 1.3. Sharing

The user can share the access to one of its own folders and subfolders of it to another user or to a user group from which it belongs.

Folders to which an user can have access (shred from) are available into the “Network” folder. Each access starts with the user name who shared the folder in first place.

  These buttons are used to share folders to other users or to users groups. Only folders can be shared (files into folders). One user can share folders to a user groups from which it belong or to any user individually.

## 2. Security

The system uses different layers of security being safe to be used in any circumstances.

- Data streams are used from the client thru the S3 storage, files are not stored temporarily for any reason
- Data transfer is encrypted all the time, the HTTPS is used from the client to the API service, AES 256 is used on data stream lately to the S3 storage.
- Authorization service resist to different attacks
- The VPC servers access is restricted by AWS security services
- The balancing solution from AWS offers DOS attack type protection
- Files are stored encrypted (AES 256).

AES 256 is a high encryption algorithm that is accepted by governmental institutions as encryption standard.

By design the system protects your files against ransom ware attacks.

The web services and used web servers offer protection to known attack forms. Files cannot be altered at rest without proper authorization. The web application files (web interfaces) are stored encrypted as any other file into the private S3 CLOUD used by the system.

Because VPN's services are not required, risks that comes with VPN access to company network from remote location are eliminated.

Administrator account is used for the web app files storage (website).

The Administrator “Applications/webroot/website” folder is public. Files can be read with a web browser using the WebApp web address.

Es: <https://<web address>/index.html>

## 3. Scalability

The scalability of the system uses the services requests balancing to a number of VPC servers, scaled horizontally.



More small servers can offer better services than a big one in this case. The loading is at the network services rather to processing and memory capabilities of one server instance.

The requests balancing offers a security plus to DOS type of attacks.

The system is ultra-scalable; AWS offers the dynamic scale of the servers' numbers depending on the moment resources loading, offering the services availability at any time.

## **4. Installation & Implementation**

For details, please read the installation guide. Also information can be found into the web presentation page, a list of recommended integrators is available.

The installation requires several steps to have all elements in place. Complex resources are required in order to use the Internet and AWS infrastructure properly ( domain name, DNS settings, SSL certificate).

The application servers are preinstalled and available into the Amazon AWS Marketplace. Configuration requires few steps, however in order to be available, the services should be allocated to AWS security groups and roles. Next is a list of prerequisites and steps to implement the solution.

### **4.1. Amazon AWS account**

In order to implement the solution an AWS account is required, should be created if not exist. The account can be used immediately; however some actions cannot be performed without an added payment method.

The AWS account if not exist should be opened in the name of the client (end user), use a company email address for the account. The account can be transferred later in case is required.

AWS will check the payment method (credit card) when it is created.

4.2. Using the AWS website and S3 services, a CLOUD private bucket should be created. For low latency, chose your AWS region that is closest to your location.

### **4.3. Access restrictions (AWS IAM)**

An IAM role should be created in order to give access to the applications server's rights to use the S3 bucket.

When you create the role, it can use the policy to use all S3 resources, in case it will not be used for something else into the future. Create a new policy first, specifically to use the S3 bucket created previously, that will limit access to one specific bucket which offers more protection if you are using more S3 buckets for different requirements.

4.4. Using the EC2 interface, one VPC should be created based on My-Drive AMI from the AWS Marketplace. Check the installation guide for recommended VPC instance types.



Allocate a fixed IP to the instance using the Elastic IP in EC2.  
Allocate previously created IAM role to the new instance.

Check the security services into the EC2 security group in order to give access to the offered services. (Check installation guide for more).

4.5. In order to offer services to a Web Application, a domain name is required. The easiest way is to use the Route 53 service from AWS to buy a domain name; in this case you will have all required services in one place. Route 53 offers DNS services for your domain name. The domain name invoice will be on the same bill.  
The DNS records should be created or actualized in order to route to the services. Please check the installation guide for more related information and cases.

4.6. Once the DNS services are in place, the server configuration web application can be opened. Setup the server to access the resources and services offer.

4.7. The server will start with a self-signed certificate; you will see a “not certified” security message into the Web Browser; it is ok to use it first for configuration. Later a certified SSL certificate is required. SSL certificates are sold by different providers, free SSLs can be created from “Let’s Encrypt”, use a SSL certificate according with the implementation and your needs, ssls.com sells wildcard certificates that can be used on domain and sub-domains which can offer flexibility later.  
The SSL certificate offers trust between a web browser and a web server/service.  
Use the My-Drive configuration to set the SSL certificate if it is the first server, each second added server will read the stored certificate from S3 CLOUD private bucket.

4.8 At first server installation, install the Web Application Interface using the configuration app. Please see the installation guide for details.

4.9. The web application for file access (My-Drive) should be up and running. One VPC server is enough for around hundred users and average usage. Later one instance can be upgraded to more resources or servers can be used in parallel ( require AWS and DNS configurations for balancing).

4.10. Scalability requires installation of more VPC servers and AWS settings.  
- install and configure more VPC instances, each will have access to the system configuration after initial setting.  
- create and configure a balancing service using the EC2 page  
- add your instances to the balancer  
- modify the DNS records according with the new configuration (balancer)  
For ultra-scalable services, an auto-scaling can be used using the AWS EC2 services.

The installation guide present details on each required step.

## **5. Usage cost**

There are infrastructure and CLOUD costs:  
- CLOUD S3 storage is around 0.03 USD per GB per month  
- download request ( from EC2) is around 0.09 USD per GB.



The infrastructure cost depends on the required resources and the implementation type. For scalable solutions a number of servers and balancing services should be considered.

Recommended VPC's prices vary from 25USD to 75USD per month. Balancing services cost is around 5USD per address (load balancer address). For reserved instances prices are lower.

The software cost is per VPC and instance type, it is priced with each instance type and available into the AWS Marketplace information.

For details the AWS price list for S3, EC2 VPC and ELB should be used.

Yearly a SSL certificate is required; price starts at 5USD for one domain and is at least 70USD for a wildcard certificate. Prices vary depending on facilities offered by the provider. If only the data transfer security is the concern a cheap SSL certificate is as good.

The domain name is around 13USD per year for .com or .NET.

Services cannot be compared with a local file server keeping in mind the security concerns and access model. A local file server is vulnerable to different attacks including ransom ware. On the other hand a local file server requires VPN's and expensive Internet connections to offer services for remote users.

## **6. Terms and conditions**

The copyright owner of the system applications is Q-Bis Consult SRL (the provider).

Open standards and open source where used to build the applications and services, however the solution copyright owner is Q-Bis Consult SRL with exceptions provided to external resources.

The web applications source code can be modified and can be installed at choice; however the server applications (services) can be installed only as specified by the provider using Amazon AWS preinstalled servers from the AWS Marketplace according with the recommended types (see installation guide).

A 15 days trial period is offered for software services, AWS infrastructure cost is not included into the trial period.

The software services fees are included into the price of the VPC's rented from the AWS. No additional fee need to be paid. The AWS list price for the EC2 EMI details the price for AWS infrastructure and the price for software. Prices are per hour.

The client has access to the last releases of the servers and applications. There are tools to be used for upgrades.



The technical used solutions are tested and used from years (starting with 2014). Due to the low cost of the solution and the virtual big value of the stored data, the software for the server's services and web application are delivered "as it is" without any warranty of any kind; Q-bis Consult SRL cannot be liable for any data loss that may occur using the software from any cause.

In the event that the local law may establish that compensation is required, the compensation cannot exceed the value of the paid service for the software for one month, but not more than 50USD. These compensation conditions are standard in IT industry, amounts may vary but the concept.

The above conditions can be modified into the future, the last model is always available on the product presentation web page.

## **7. Technical support**

Basic technical support is offered by email to a system administrator in case there is no integrator that offers technical support or maintenance.

Depending on requests complexity there can be additional fees per intervention when there is no maintenance contract. The fees level relates to complexity and confidentiality agreements required for the operation.

Being a software application solution that requires installation and implementation to the CLOUD provider with different scenarios, under a final client account, the technical support may be the subject of the implementation contract with an integrator.

Q-bis Consult present a list of software integrators on the product presentation web-pages.

## **8. WEB Services ( API ), applications development**

Due to the system construction based on open standards, the offered services can be consumed by modern software application and integration with third party software.

*Presented information in this chapter requires some programming skills*

Scenario: Develop a CRM software solution

My-Drive services can be used by another software solution for authorization and files access.

The WEB server (application server) can offer hosting for the new web application. Here we talk about WEB apps based on JavaScript frameworks like AngularJS, Angular, VueJs, React and other alike.

The CRM software will require additional database services that should be provided from another source (ask if it is required). By using My-Drive web services, the new CRM app solves two major tasks from start:

- A secured authorization service
- Easy to be used CLOUD services to store files ( per client, contracts, invoices, images )



The solution is secure and scalable by default, if the database access respects same standards, the success is guaranteed

Amazon AWS Aurora (RDS) offers support for very large databases (managed clusters) of PostgreSQL and MySQL. API servers for PostgreSQL can be implemented with Q-Bis Consult solutions for Postgresql CRUD API.

### **8.1. Web services, authorization service (API)**

RESTful web services, HTTPS protocol, the response is a JSON object.

#### **User API:**

##### **authorization (login)**

```
method: GET
url: "https://<authserver>/users/login"
headers: {
  'Content-Type': "application/json; charset=utf-8",
  'Authorization': "BASIC <authstring>"
}
```

authstring = Base64( "<utilizator>:<parola>" )

Response:  
{login:"OK", token:<JWT token>}

##### **update ( password, last name, first name )**

```
method:POST
url: "https://<authserver>/admin/updateme"
headers:{
  "Content-Type": "application/json; charset=utf-8",
  "Authorization": "Bearer <token>"
}
data:{
  f:<first name>
  l: <last name>
  pwc: <new password>
}
```

Response:  
{token:<JWT token>}

#### **Administrator:**

##### **Authorization for administrator (token)**



Used only by the “administrator” account, returns an additional token requested for users management requests:

```
method:POST
url: "https://<authserver>/admin/admlogin"
headers:{
    "Content-Type": "application/json; charset=utf-8",
    "Authorization": "Bearer <JWT token>"
}
data: {}
```

Response:  
{token:<token>}

### Create user

```
method:POST
url: "https://<authserver>/admin/createuser"
headers:{
    "Content-Type": "application/json; charset=utf-8",
    "Authorization": "Bearer <adm credit>"
}
data:{
    user:<user>
    firstname:<first name>
    lastname: <last name>
    pwc: <password>
}
```

Response: { \_action: "create", \_uid: "system", \_xt: "<DateTime>", f: "<first name>", l: "<last name>", id: "<id-user>", u: "<user>" }

### User Update (by administrator)

```
method:POST
url: "https://<authserver>/admin/updateuser"
headers:{
    "Content-Type": "application/json; charset=utf-8",
    "Authorization": "Bearer <adm credit>"
}
data:{
    uid: "<id-utilizator>",
    user: "<utilizator>",
    update:{
        f: "<first name>",
        l: "<lastname>",
        newpwt: "<new password>"
    }
}
```



Update – only new values should be here.

Response:

```
{
  msg:"write data",
  response:{
    alloc:true,
    data:[
      id:"<id-user>",
      f:"<first name>",
      l:"<last name>",
      u:"<user>",
      g:[
        {g:"<group>",id:"<id>"}
      ],
      _action:"update",
      _uid:"system",
      _xt: "<DateTime>"
    ]
  }
}
```

Response data is response.data[0]

### Users List:

Returns users list. Parts of the list can be requested.

```
method:POST
url: "https://<authserver>/admin/getusers"
headers:{
  "Content-Type": "application/json; charset=utf-8",
  "Authorization": "Bearer <adm credit>"
}
data:{
  nro:<records number>,
  lastfrom:<requests from number>
}
```

data.lastfrom it is not mandatory, can be used to request users starting with one number.

Response: (type array)

```
[
  {
    id:"<id-user>",
    f:"<first name>",
    l:"<last name>",
    u:"<user>",
    g:[
```



```

                {g:"<group>",id:"<id>"}
            ],
            _action:"update",
            _uid:"system",
            _xt: "<DateTime>"
        }
    },
    ...
]

```

### User search:

```

method:POST
url: "https://<authserver>/admin/finduser"
headers:{
    "Content-Type": "application/json; charset=utf-8",
    "Authorization": "Bearer <adm credit>"
}
data:{
    user:"<user>"
}

```

response: same as previous (users list)

### Users list information:

```

method:POST
url: "https://<authserver>/admin/usersinfo"
headers:{
    "Content-Type": "application/json; charset=utf-8",
    "Authorization": "Bearer <adm credit>"
}, data:{}

```

Response: number of the users into the system

### Create users group

```

method:POST
url: "https://<authserver>/admin/creategroup"
headers:{
    "Content-Type": "application/json; charset=utf-8",
    "Authorization": "Bearer <adm credit>"
},
data:{
    g:"<group>"
}

```

Răspuns:



```

{
  msg:"write data",
  response:{
    alloc:true,
    data:[
      {
        g:"<group>",
        id:"<id grup>"
      }
    ]
  }
}

```

## Update group information

```

method:POST
url: "https://<authserver>/admin/savegroup"
headers:{
  "Content-Type": "application/json; charset=utf-8",
  "Authorization":"Bearer <adm credit>"
},
data:{
  id:"<id grup>",
  u:{
    id:"<id user>",
    u:"<user>"
  }
}

```

```

Response:
{
  msg:"write data",
  response:{
    alloc:true,
    data:[
      {
        id:"<id user>",
        f:"<first name>",
        l:"<last name>",
        u:"<user>",
        g:[
          {g:"<group>",id:"<id grup>"},...
        ]
      }
    ]
  }
}

```



## Groups list

```
method:POST
url: "https://<authserver>/admin/getgroups"
headers:{
    "Content-Type": "application/json; charset=utf-8",
    "Authorization": "Bearer <adm credit>"
}
data:{nro:1000}
```

```
Response:
{
    records:[
        {g:"<grup>", id:"<id grup>"}
        ,...
    ]
}
```

## 8.2. Web Services, the file server (API)

### Default folders

Returns default folders information ( root ).

If the user is new it creates the initial folders structure. Same response is returned in both cases.

```
method:POST
url:"https://<appserver>/sfd/defaults"
headers:{
    "Content-Type": "application/json; charset=utf-8",
    "Authorization": "Bearer <token>"
}
data: {}
```

```
Response:
{
    apps:true,
    fixed:true,
    fldname:"Root",
    folders:[
        {fldname:"Documents",id:"Documents"},
        {fldname:"Files",id:"Files"},
        {fldname:"Trash",id:"Trash"},
        {fldname:"Applications",id:"Applications"}
    ],
    id:"Root",
    parentid:"Root",
    ownerid:"<cale>"
}
```



```
}
```

### Create folder

```
method:POST
url:"https://<appserver>/sfd/createfolder"
headers:{
  "Content-Type": "application/json; charset=utf-8",
  "Authorization": "Bearer <token>"
}
data:{
  "folder": "<foldername>",
  "parentid": "<folder path>"
}
```

Response:

```
{apps:false,
fixed:false,
fldname:"<foldername>",
folders:[],
id:"<foldername>",
parentid:"<calea catre folder>",
ownerid:"<cale initiala>"}
```

### Delete folder

```
method:POST
url:"https://<appserver>/sfd/deletefolder"
headers:{
  "Content-Type": "application/json; charset=utf-8",
  "Authorization": "Bearer <token>"
}
data:{
  "folderid": "<folder path>"
}
```

```
Ex: {
  "folderid": "Documents/n1/n2/n3"
}
```

Response:

```
{folderid: "<calea catre folder>"}
Ex: {folderid: "Documents/n1/n2/n3"}
```

### Folder information: (include subfolders names)

```
method:POST
url:"https://<appserver>/sfd/folderdata"
```



```
headers:{
  "Content-Type": "application/json; charset=utf-8",
  "Authorization": "Bearer <token>"
}
data:{
  id: "<folder path>"
}
```

Response:

```
Ex: {
  "data": {
    "id": "n1",
    "fldname": "n1",
    "createdAt": "2021-02-14T06:54:34.938Z",
    "rights": {},
    "folders": [
      {
        "fldname": "n2"
      }
    ],
    "fixed": false,
    "apps": false,
    "ownerid": "GLJTSFUTTXJLWGYH",
    "parentid": "Documents"
  }
}
```

### Folder files:

```
method:POST
url:"https://<appserver>/upload/s3enc/sfd/folderfiles"
headers:{
  "Content-Type": "application/json; charset=utf-8",
  "Authorization": "Bearer <token>"
}
data:{
  id: "<cale folder>"
}
```

Response:

```
{
  "files": [
    {
      "name": "info.txt",
      "size": 2,
      "createdAt": "2021-02-08T15:06:33.000Z"
    }
  ],
  "parentid": "Documents"
}
```



```
}
```

### Copy files:

```
method:POST
url:"https://<appserver>/sfd/copyfiles"
headers:{
  "Content-Type": "application/json; charset=utf-8",
  "Authorization": "Bearer <token>"
}
data:{
  files:[
    {
      id:"<file.name>",
      ext:"<extension>",
      parentid:"<path to file>",
      preext:"<just filename>"
    }
  ],
  folderid:"<new path>"
}
```

### Sample:

```
data: {
  files:[
    {id:"info.txt",ext:"txt",parentid:"Documents/test",preext:"info"}
  ],
  folderid:"Files/test2"
}
```

### Response:

```
{
  "files": [
    {
      "label": "info.txt",
      "id": "info.txt",
      "parentid": "Documents",
      "sdate": "2021-02-08T15:06:33.000Z",
      "size": 1,
      "preext": "info",
      "ext": "txt",
      "nid": "info.txt"
    }
  ],
  "errors": []
}
```



## Delete files

```
method:POST
url:"https://<appserver>/sfd/deletefile"
headers:{
    "Content-Type": "application/json; charset=utf-8",
    "Authorization": "Bearer <token>"
}
data:{
    opt:false,
    files[
        {id:<filename>}
    ],folderid:<id folder>
}
```

Ex:

```
data = {files:[{id:"file1.xlsx"}],folderid:"Files/test_folder",opt:false}
```

Response:

```
data:{
    errors:[],
    files:[{id:<filename>}]
}
```

## File upload:

```
method:POST
url:"https://<appserver>/upload/s3enc/" + folderid + "?token=" + key;
data:{s3enc:<fileobject>}
```

“multipart form upload”

Folderid – folder path, replace “/” with “\*” and use encodeURI(folderid)

Sample response:

```
{
    ename: "s3enc",
    cnt: "Content-Disposition: form-data; name=\"s3enc\"; filename=\"q-plan.docx\"",
    contenttype: "application/vnd.openxmlformats-officedocument.wordprocessingml.document",
    filename: "q-plan.docx",
    name: "q-plan.docx",
    start: "-----187855873322272168611499597250",
    end: "-----187855873322272168611499597250--",
    poz: 220,
    tpe: "Content-Type: application/vnd.openxmlformats-officedocument.wordprocessingml.document"
}
```

## File download:



```
Method: GET
url:"https://<appserver>/simple/getfile/<fileinfo>
query:{
    token:<token>
}
fileinfo = (filepath + "/" + filename).replace(replace(/\\/g, "" )
Ex: "Applications*folder*new*info.txt"
```

The Response has "headers" set in order to open "SaveAs" in browser.

### **8.3. Programmable web services support**

The application servers offers support for software application development offering its services as standard API's.

New services can be added to existing services for different purposes.

The web services interfaces (API) use open standards and respect RESTful principles.

### **8.4. Web Server for web apps and presentation web pages**

Two standard web ports are opened:

- port 80, opened for a redirect page that send the request to the 443 secured port
- port 443, secured port opened by the application server

The application server serves specific web services for drive application and can server also static web pages stored into the private S3 CLOUD.

The web server service benefits the scalability of the entire system; the web pages are stored in CLOUD and are secured.

Text editors are available for the website files, the access is granted using the "administrator" account and the web application of My-Drive; the location is "Applications/webroot/website".

The Administrator "Applications/webroot/website" folder is public. Files can be read with a web browser using the WebApp web address.

Es: <https://<web address>/index.html>