Integrate applications with the application provider

CS3 2022 – Cloud Storage Synchronization and Sharing

Overview

1 Introduction to the cs3.app.provider

2 Set up a demo application

3 Use the demo application

4 Q&A

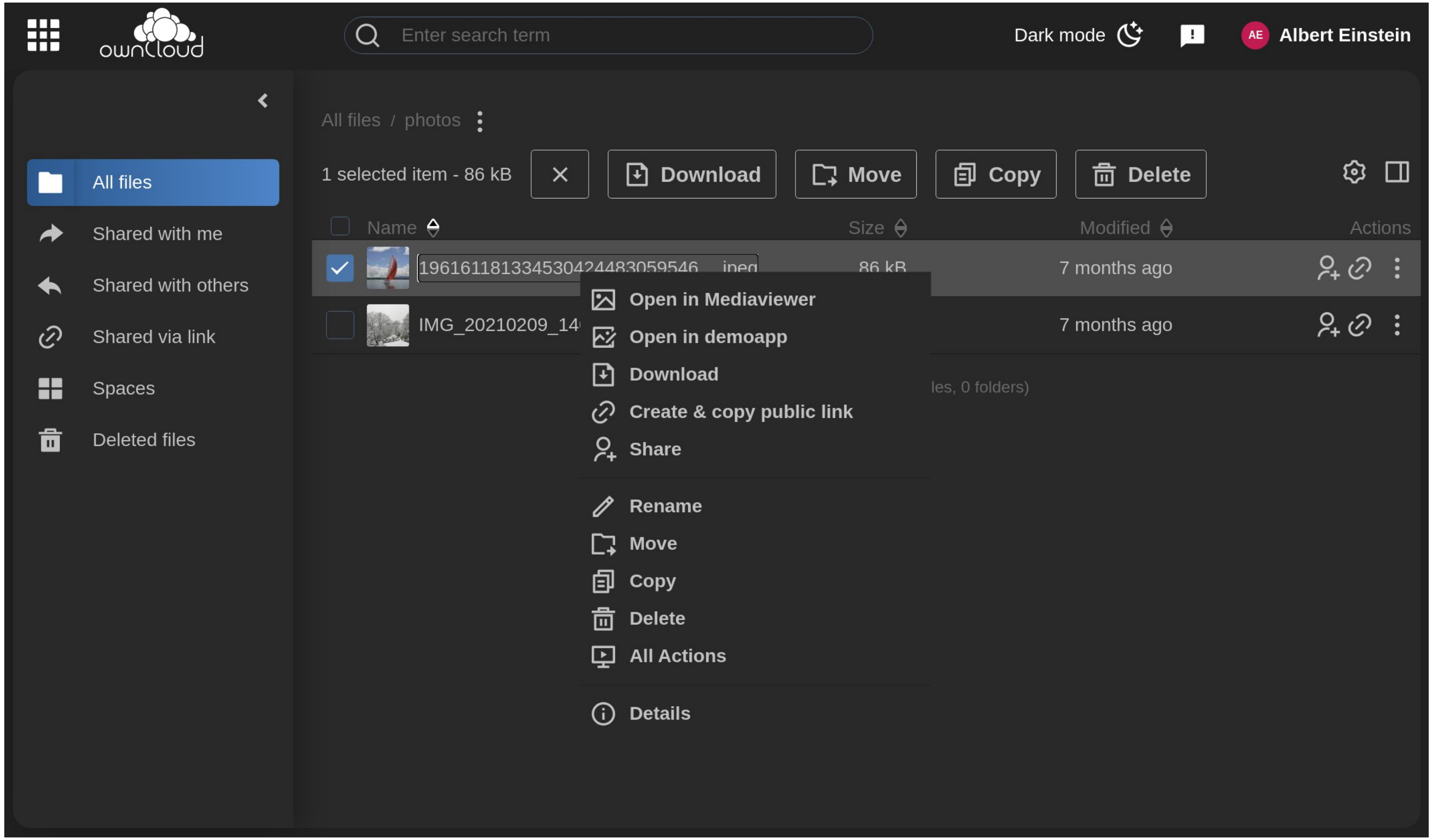
Willy Kloucek

Introduction to the cs3.app.provider

How to easily integrate apps

Check out https://owncloud.dev/

Why do we have the cs3.app.provider?



Check out https://owncloud.dev/

How to build an cs3.app.provider?

- 🔗 cs3.app.provider docs
- An app provider needs to provide the OpenInApp grpc call endpoint

- 🔗 cs3.app.registry docs
- An app provider needs to register itself at a registry via the AddAppProvider grcp call

Check out https://owncloud.dev/

How to use apps as a client?

- 🔗 [cs3.app.provider docs]
- 🔗 [cs3.app.registry docs]
- Do `ListAppProviders` and `OpenInApp` grpc calls if you speak grpc

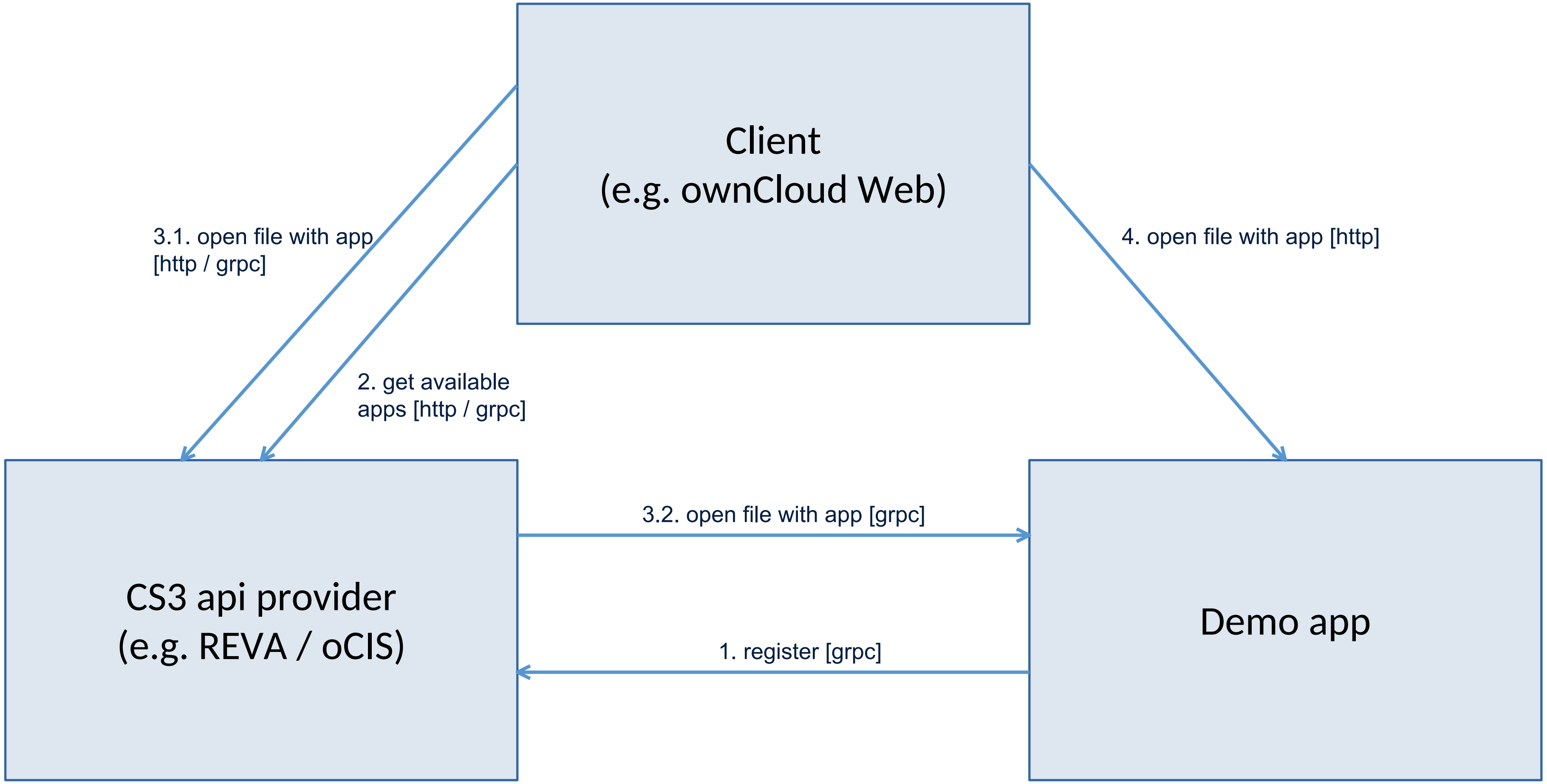
- 🔗 owncloud.dev apps docs
- Do `/app/list` and `/app/open` http calls if you speak http

Check out https://owncloud.dev/

Set up a demo application

Check out https://owncloud.dev/

Demo application architecture


Client
(e.g. ownCloud Web)

3.1. open file with app
[http / grpc]

2. get available
apps [http / grpc]

4. open file with app [http]

CS3 api provider
(e.g. REVA / oCIS)

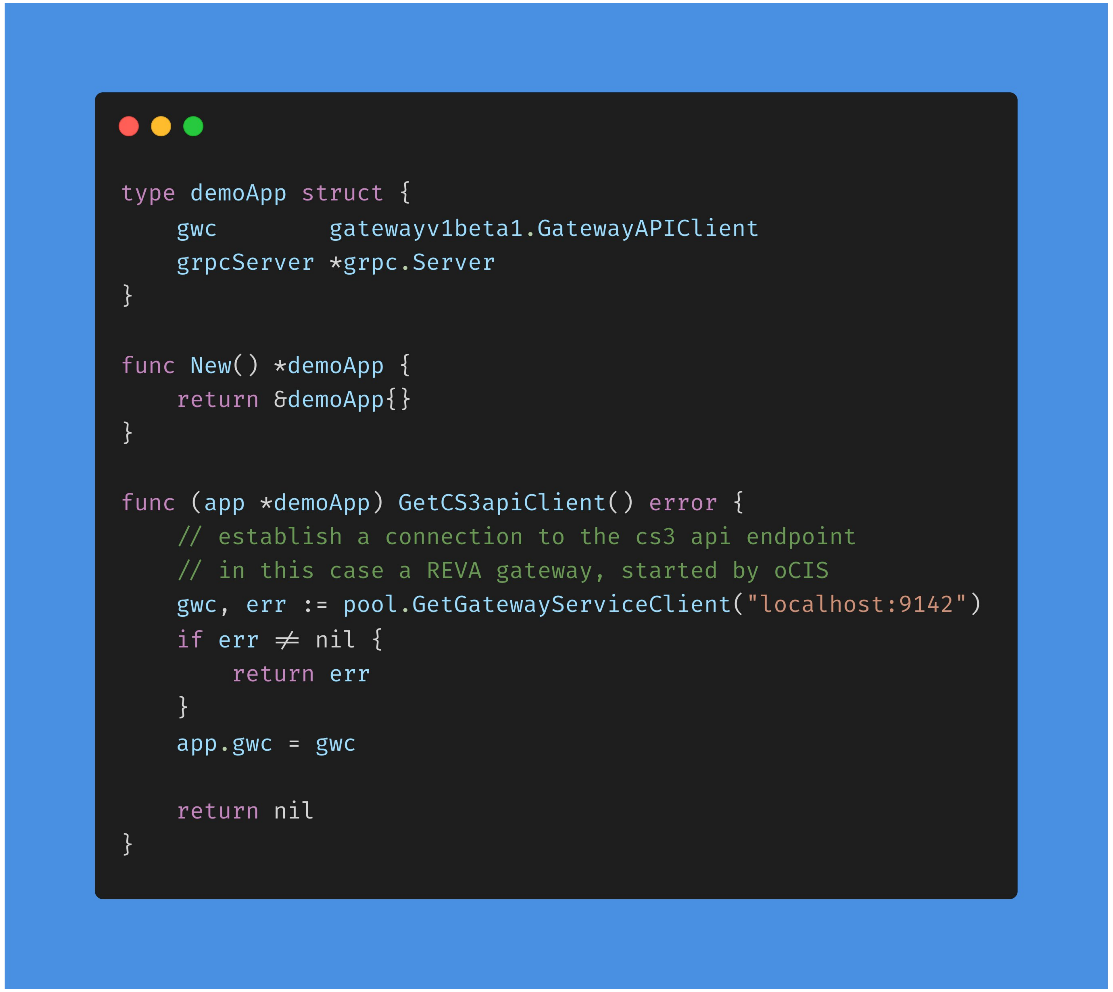
3.2. open file with app [grpc]

1. register [grpc]

Demo app

Check out https://owncloud.dev/

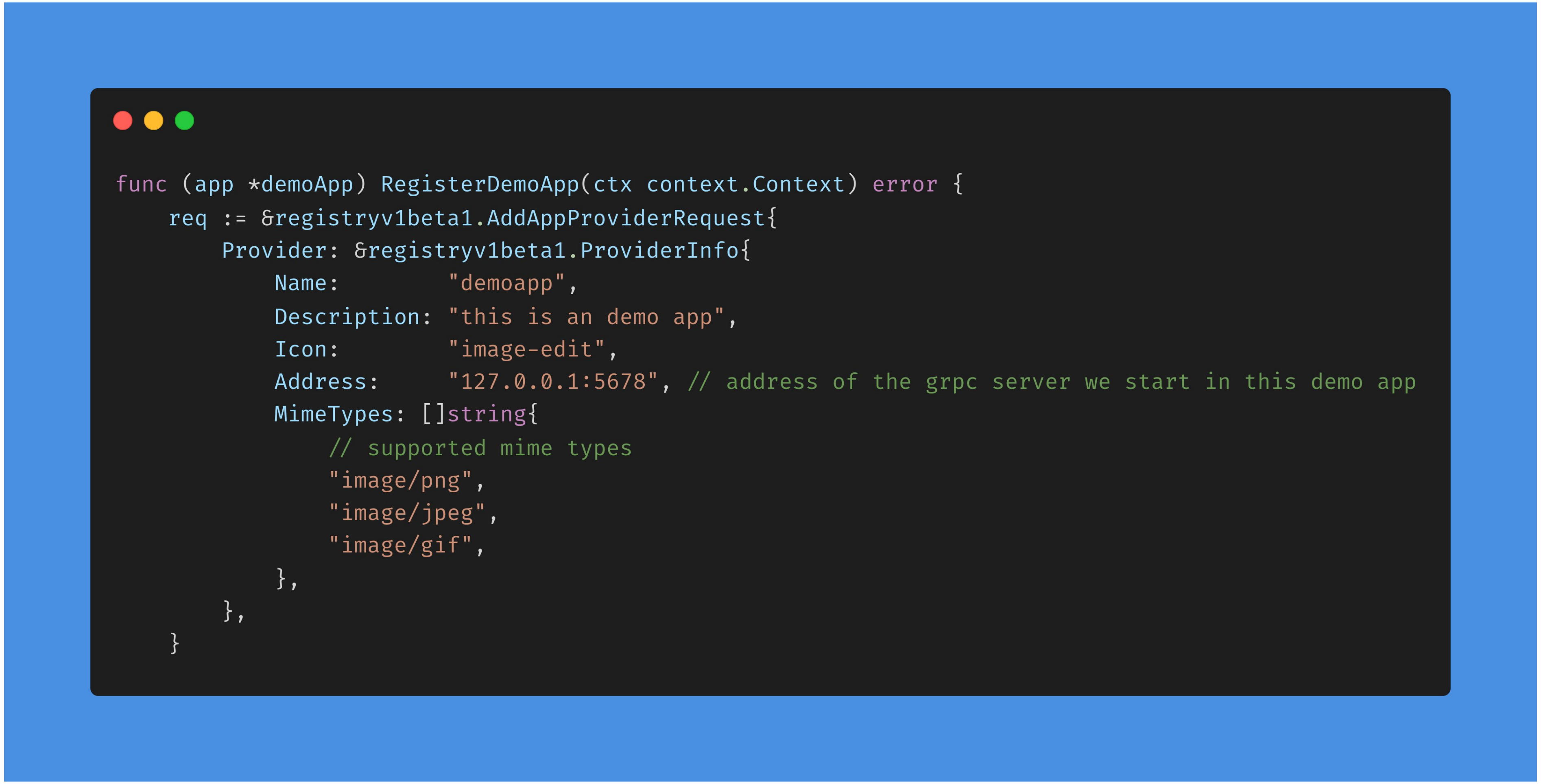
Connection to the CS3api



```go
type demoApp struct {
    gwc        gatewayv1beta1.GatewayAPIClient
    grpcServer *grpc.Server
}

func New() *demoApp {
    return &demoApp{}
}

func (app *demoApp) GetCS3apiClient() error {
    // establish a connection to the cs3 api endpoint
    // in this case a REVA gateway, started by oCIS
    gwc, err := pool.GetGatewayServiceClient("localhost:9142")
    if err != nil {
        return err
    }
    app.gwc = gwc

    return nil
}
```

Check out https://owncloud.dev/

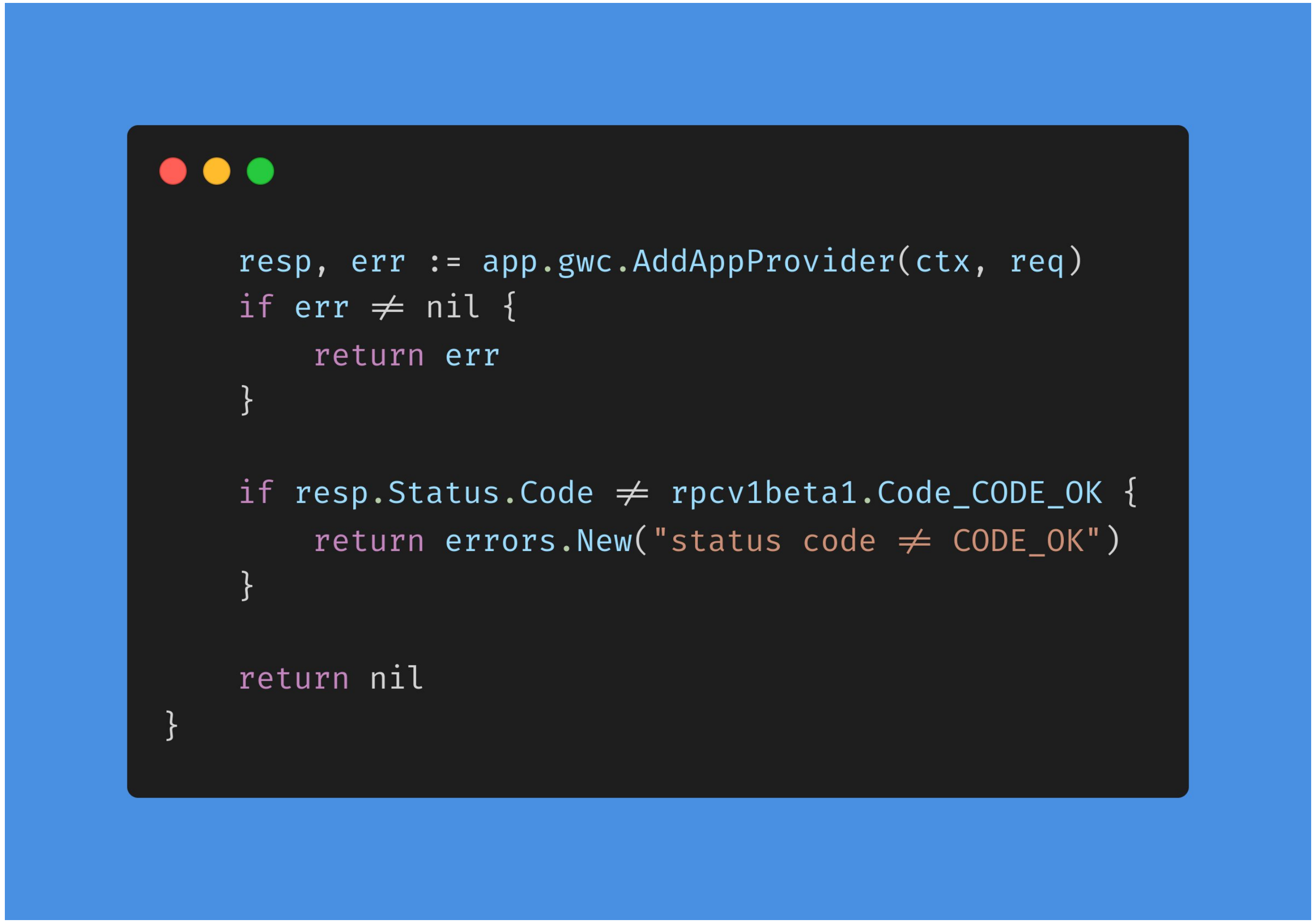
Register the app



```go
func (app *demoApp) RegisterDemoApp(ctx context.Context) error {
    req := &registryv1beta1.AddAppProviderRequest{
        Provider: &registryv1beta1.ProviderInfo{
            Name:        "demoapp",
            Description: "this is an demo app",
            Icon:        "image-edit",
            Address:     "127.0.0.1:5678", // address of the grpc server we start in this demo app
            MimeTypes: []string{
                // supported mime types
                "image/png",
                "image/jpeg",
                "image/gif",
            },
        },
    }
```

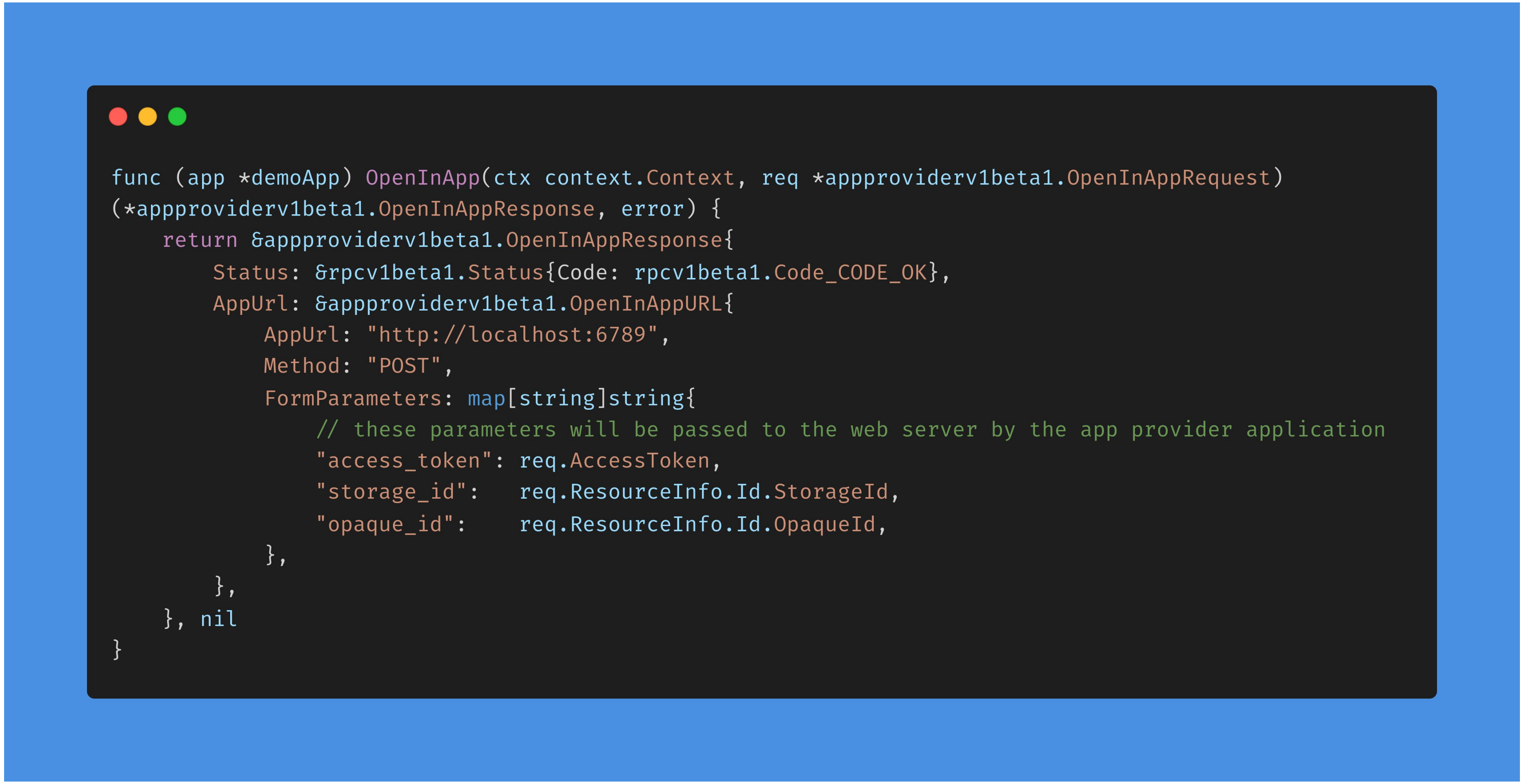
Check out https://owncloud.dev/

Register the app



```
    resp, err := app.gwc.AddAppProvider(ctx, req)
    if err ≠ nil {
        return err
    }

    if resp.Status.Code ≠ rpcv1beta1.Code_CODE_OK {
        return errors.New("status code ≠ CODE_OK")
    }

    return nil
}
```

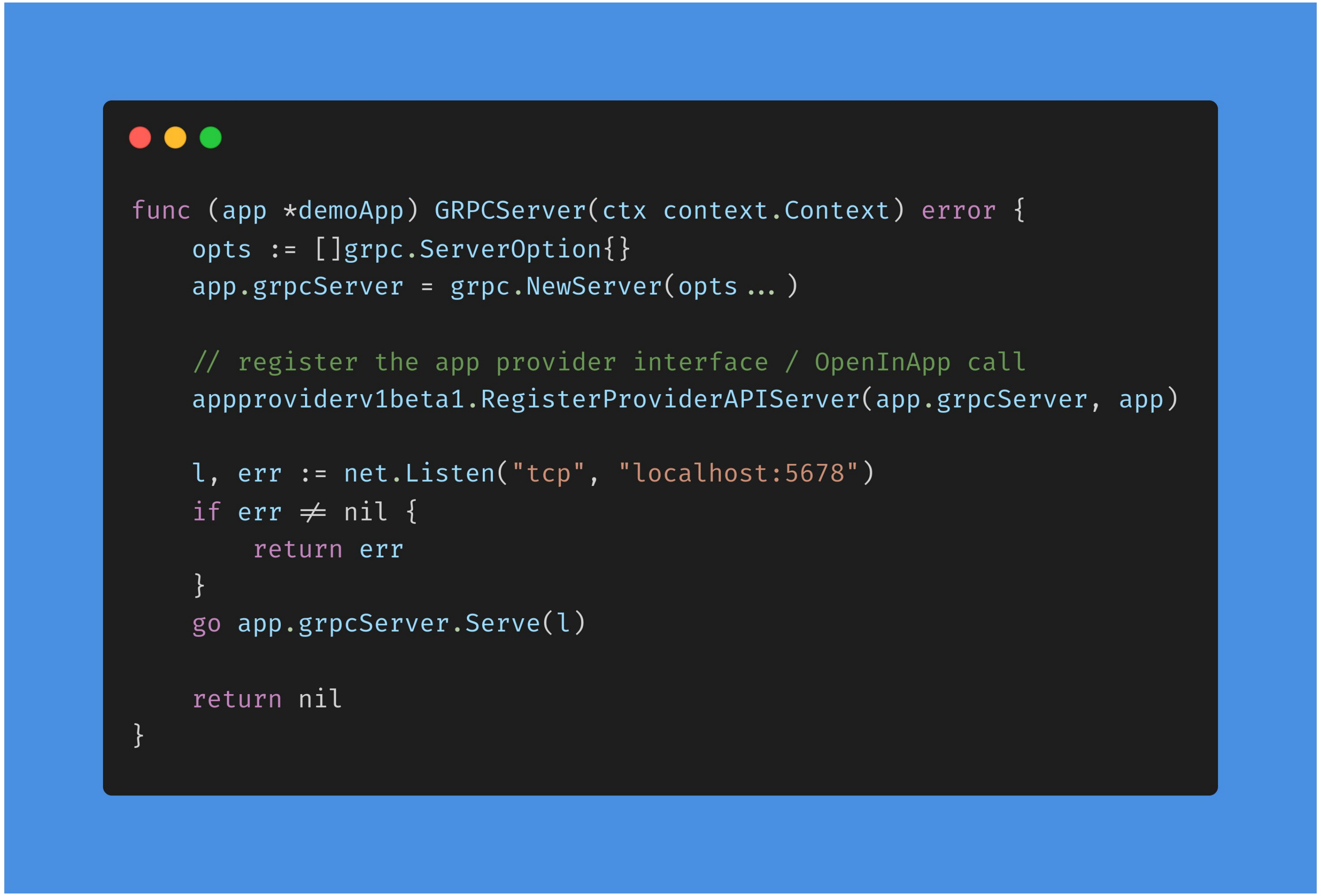
Implement the OpenInApp call



```
func (app *demoApp) OpenInApp(ctx context.Context, req *appproviderv1beta1.OpenInAppRequest)
(*appproviderv1beta1.OpenInAppResponse, error) {
    return &appproviderv1beta1.OpenInAppResponse{
        Status: &rpcv1beta1.Status{Code: rpcv1beta1.Code_CODE_OK},
        AppUrl: &appproviderv1beta1.OpenInAppURL{
            AppUrl: "http://localhost:6789",
            Method: "POST",
            FormParameters: map[string]string{
                // these parameters will be passed to the web server by the app provider application
                "access_token": req.AccessToken,
                "storage_id":   req.ResourceInfo.Id.StorageId,
                "opaque_id":    req.ResourceInfo.Id.OpaqueId,
            },
        },
    }, nil
}
```

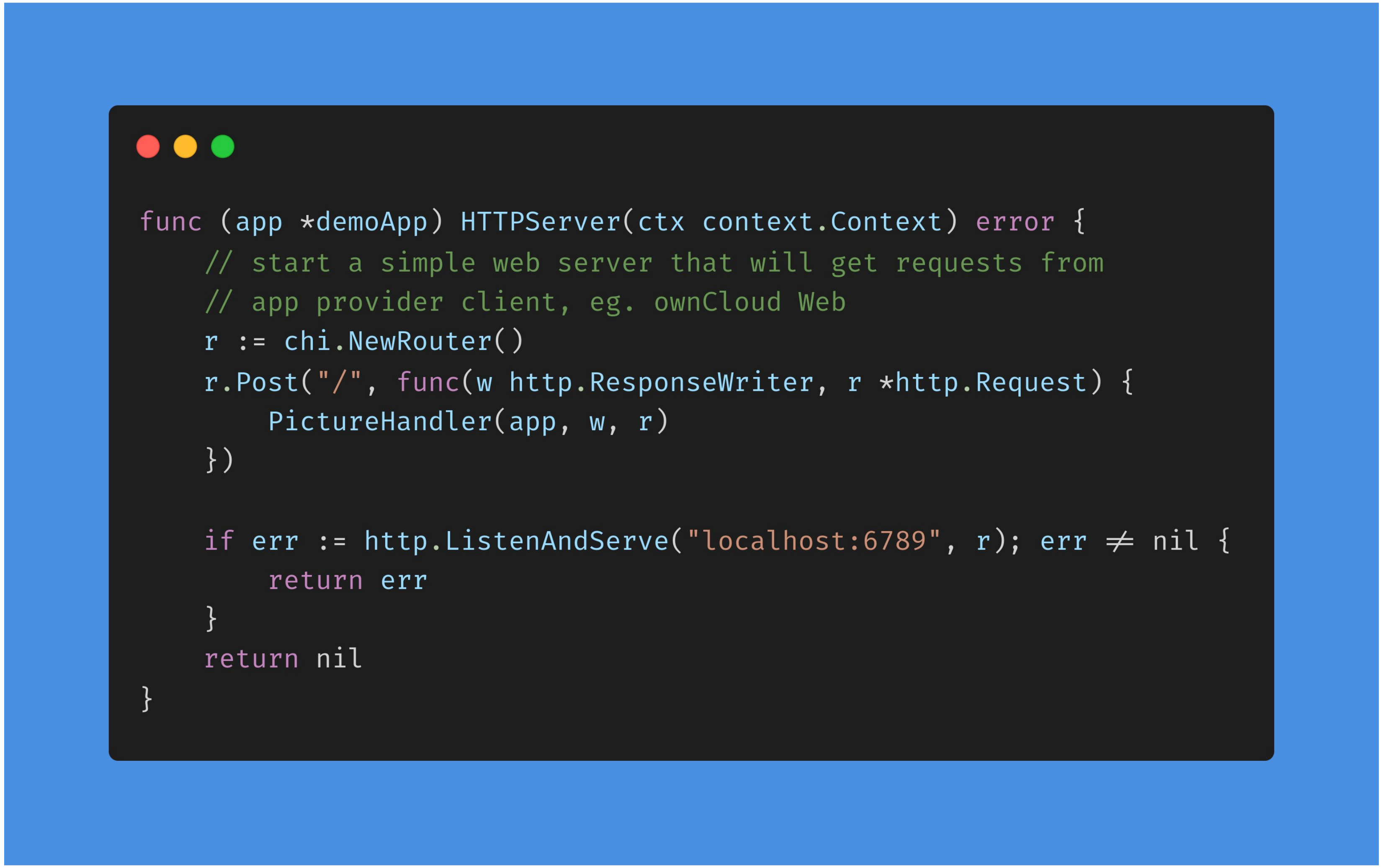
Check out https://owncloud.dev/

Start the GRPC server



```
func (app *demoApp) GRPCServer(ctx context.Context) error {
    opts := []grpc.ServerOption{}
    app.grpcServer = grpc.NewServer(opts…)

    // register the app provider interface / OpenInApp call
    appproviderv1beta1.RegisterProviderAPIServer(app.grpcServer, app)

    l, err := net.Listen("tcp", "localhost:5678")
    if err ≠ nil {
        return err
    }
    go app.grpcServer.Serve(l)

    return nil
}
```

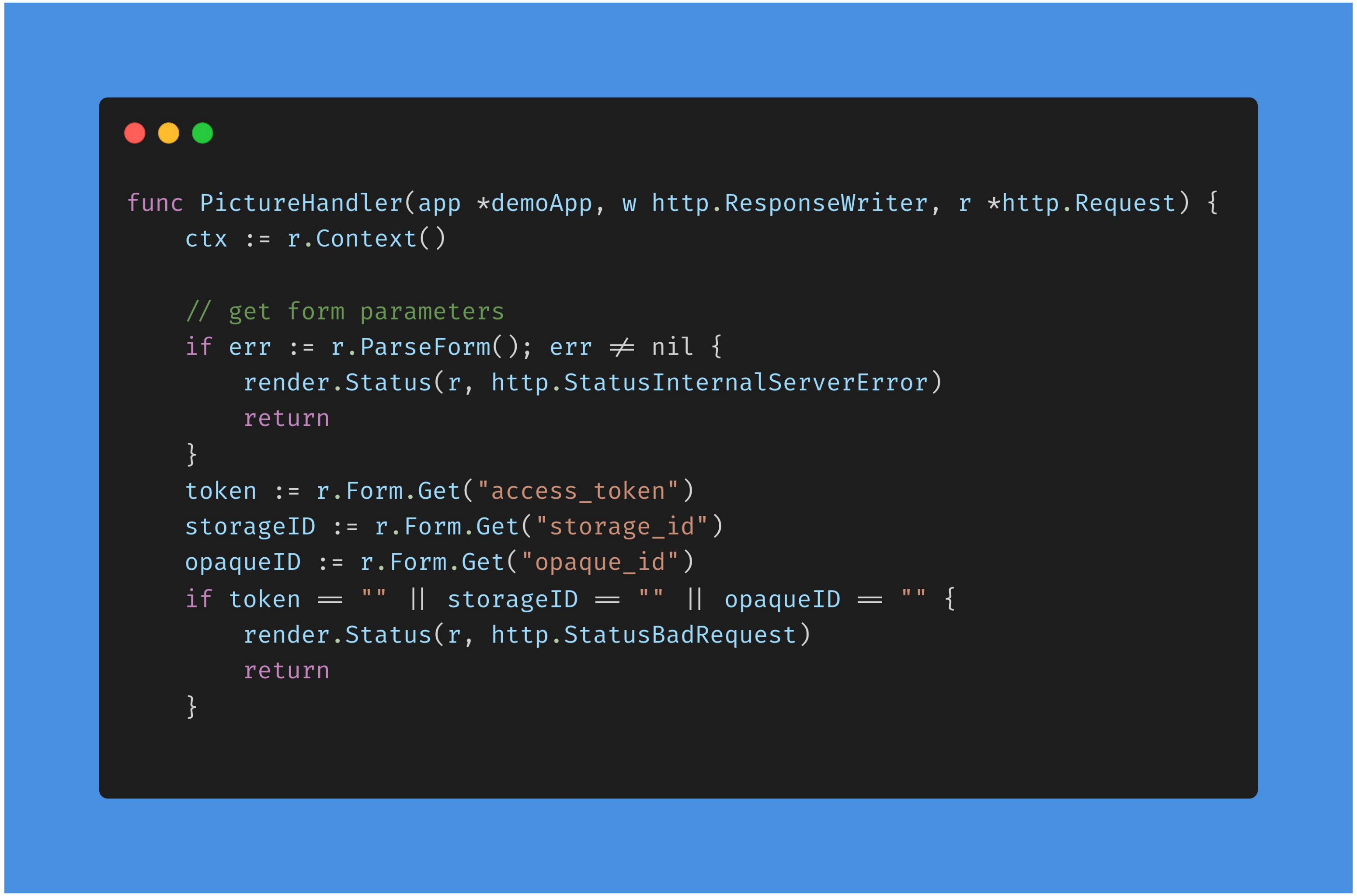
Start the HTTP server



```go
func (app *demoApp) HTTPServer(ctx context.Context) error {
    // start a simple web server that will get requests from
    // app provider client, eg. ownCloud Web
    r := chi.NewRouter()
    r.Post("/", func(w http.ResponseWriter, r *http.Request) {
        PictureHandler(app, w, r)
    })

    if err := http.ListenAndServe("localhost:6789", r); err != nil {
        return err
    }
    return nil
}
```

Check out https://owncloud.dev/

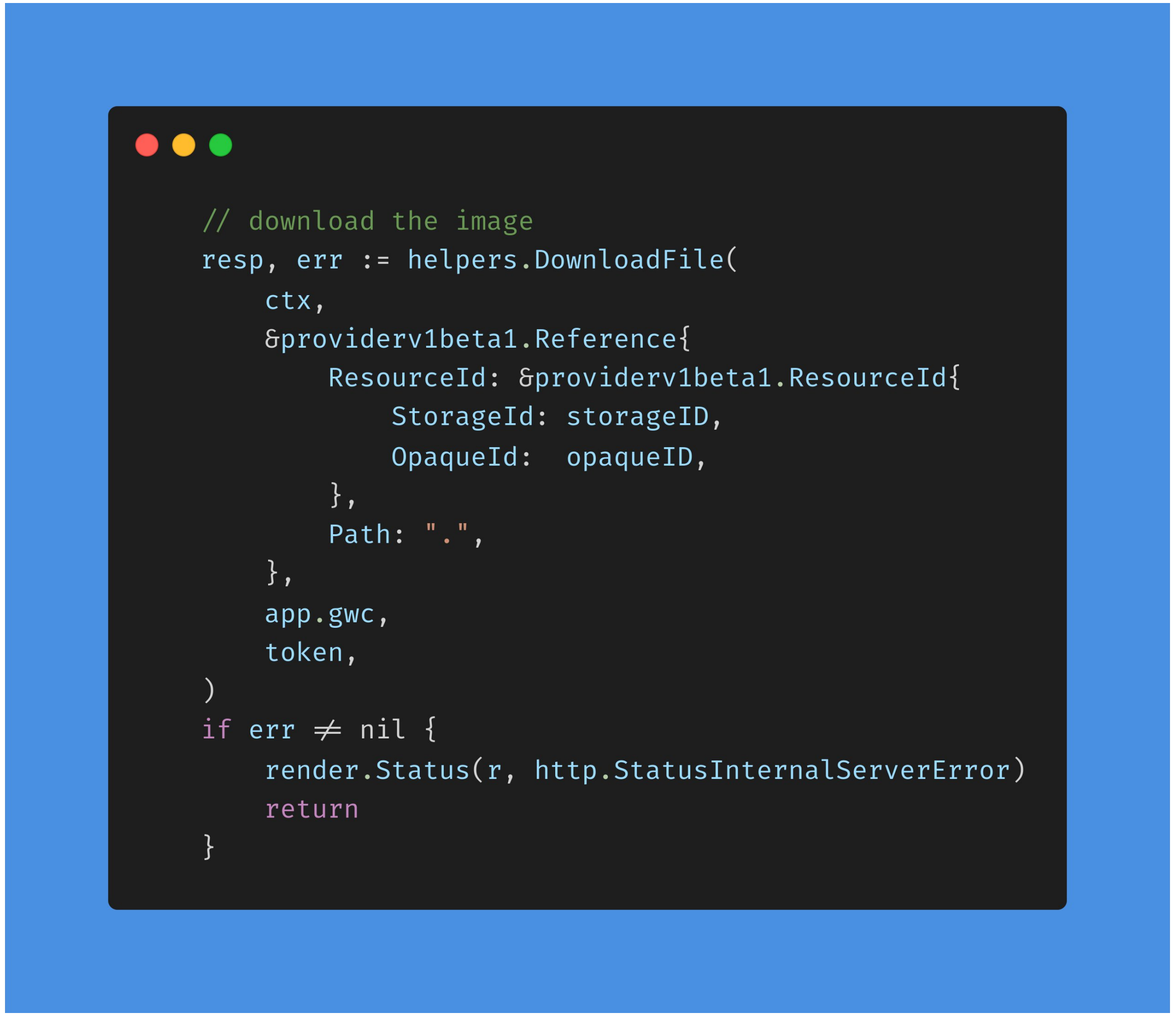
Implement the HTTP handler / app logic



```go
func PictureHandler(app *demoApp, w http.ResponseWriter, r *http.Request) {
    ctx := r.Context()

    // get form parameters
    if err := r.ParseForm(); err != nil {
        render.Status(r, http.StatusInternalServerError)
        return
    }
    token := r.Form.Get("access_token")
    storageID := r.Form.Get("storage_id")
    opaqueID := r.Form.Get("opaque_id")
    if token == "" || storageID == "" || opaqueID == "" {
        render.Status(r, http.StatusBadRequest)
        return
    }
```

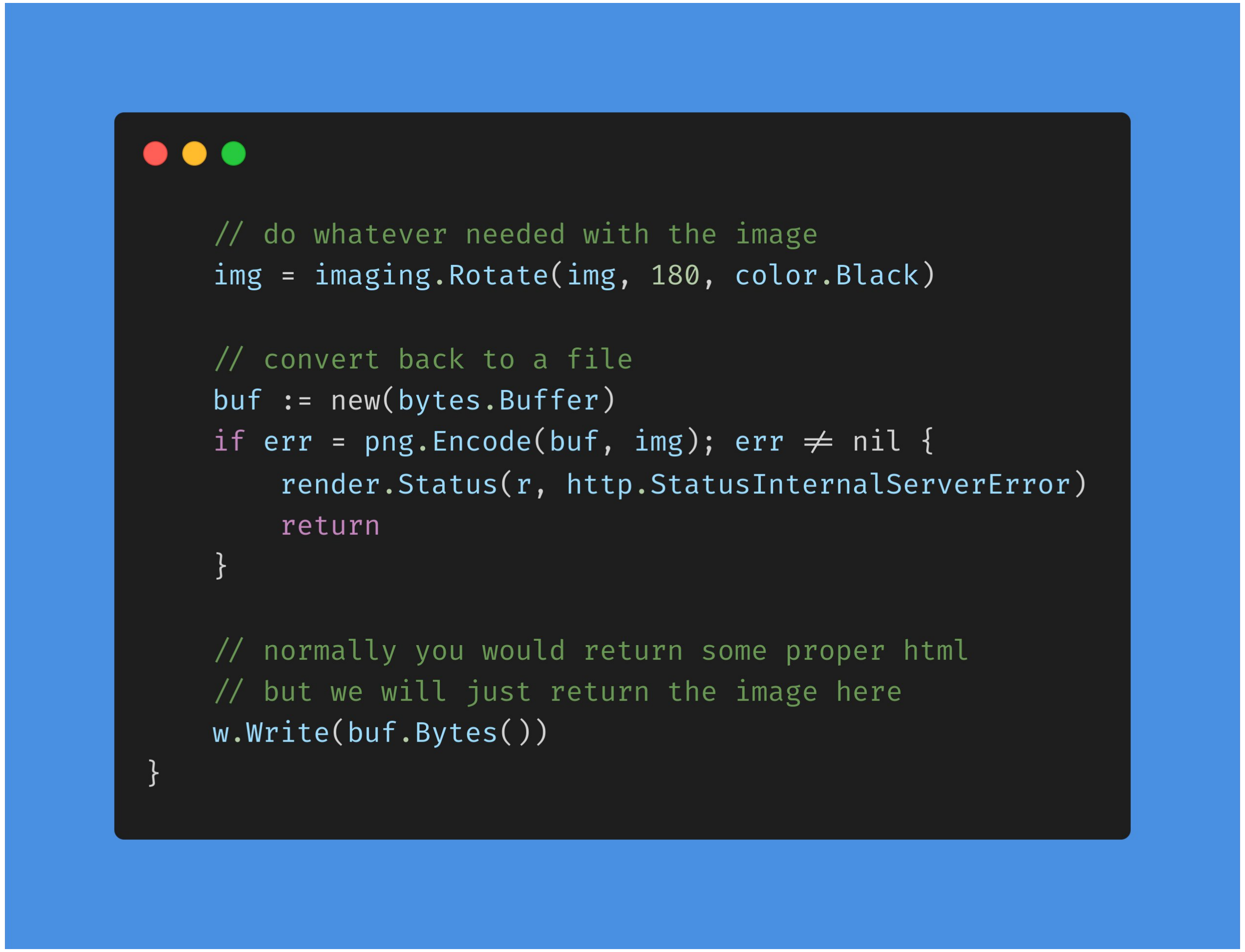
Check out https://owncloud.dev/

Implement the app logic



```
// download the image
resp, err := helpers.DownloadFile(
    ctx,
    &providerv1beta1.Reference{
        ResourceId: &providerv1beta1.ResourceId{
            StorageId: storageID,
            OpaqueId:  opaqueID,
        },
        Path: ".",
    },
    app.gwc,
    token,
)
if err ≠ nil {
    render.Status(r, http.StatusInternalServerError)
    return
}
```

Implement the app logic

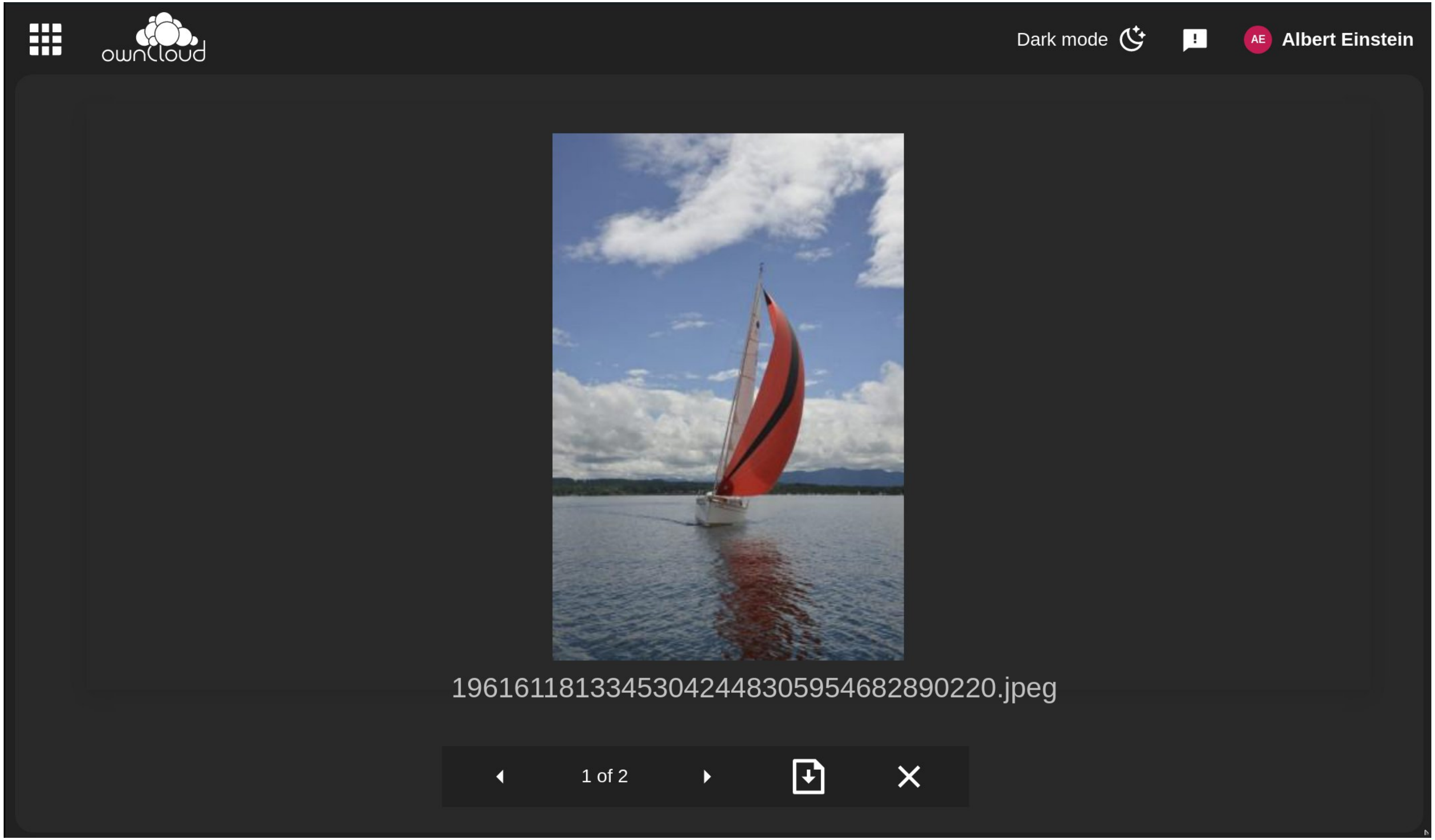


```
    // do whatever needed with the image
    img = imaging.Rotate(img, 180, color.Black)

    // convert back to a file
    buf := new(bytes.Buffer)
    if err = png.Encode(buf, img); err != nil {
        render.Status(r, http.StatusInternalServerError)
        return
    }

    // normally you would return some proper html
    // but we will just return the image here
    w.Write(buf.Bytes())
}
```

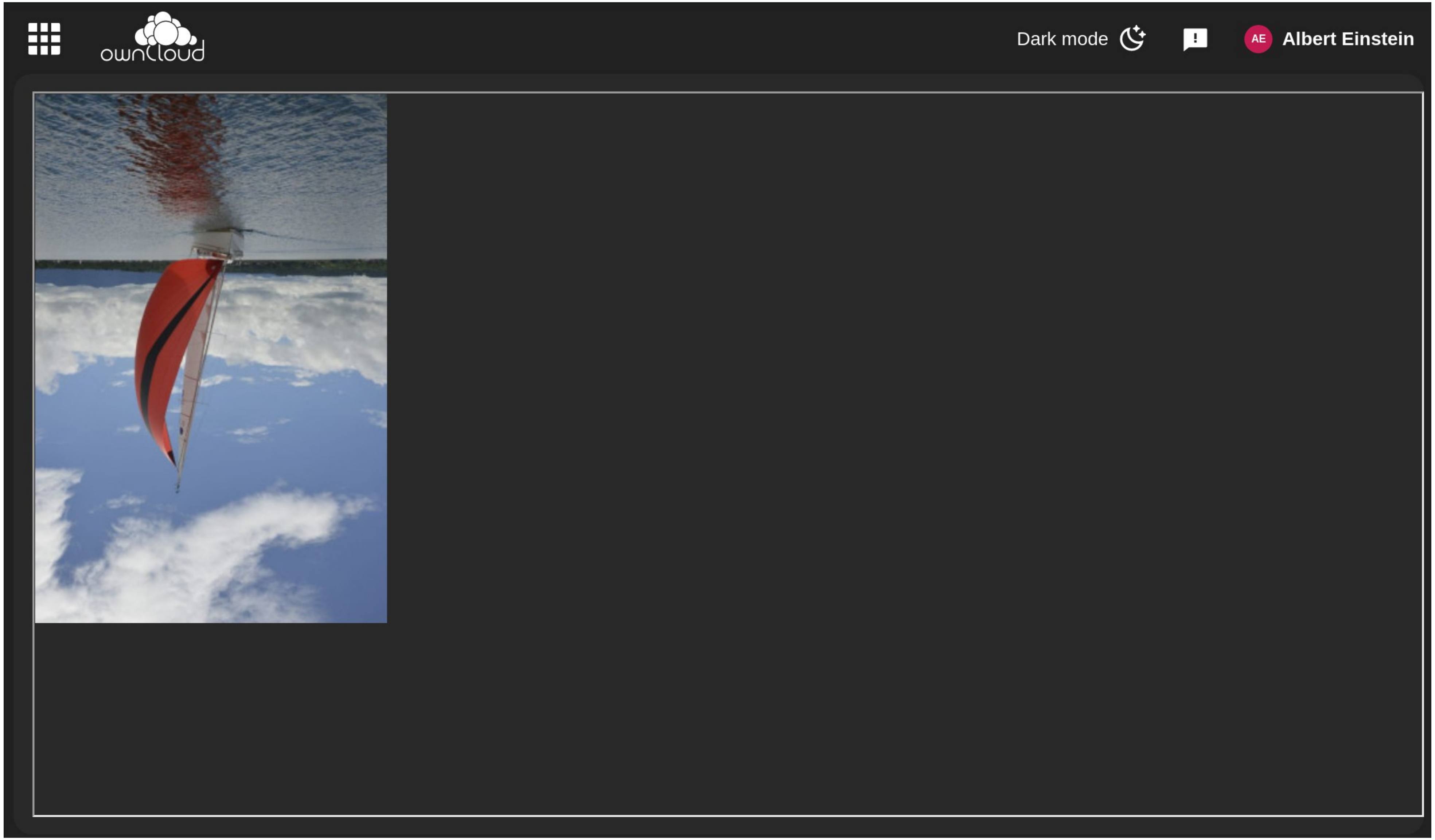
Use the demo application

Open an image with our demo application

Check out https://owncloud.dev/



Check out https://owncloud.dev/



Check out https://owncloud.dev/

Q&A

Find this demo app on github.com/wkloucek/cs3-demo-app

Check out https://owncloud.dev/

ownCloud