using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using MongoDB.Bson;

using MongoDB.Driver;

namespace ShareHubServer.Controllers.Api {

[Route("api/BoxGateway")]

public class BoxGatewayController : Controller {

#region check nfc

[HttpPost("CheckNfc")]

public ActionResult CheckNfcResponder(string nfcId, string boxKey) {

if (!Utilities.Sanitized(nfcId, boxKey)) {

return StatusCode(405, "one or more arguments were null");

}

return Json(CheckNfc(nfcId, boxKey));

}

internal ResultBase CheckNfc(string nfcId, string boxKey) {

IMongoCollection<DBEntry.User> usersCollection = Program.Database.GetCollection<DBEntry.User>("users");

IMongoCollection<DBEntry.Community> communitiesCollection = Program.Database.GetCollection<DBEntry.Community>("communities");

List<DBEntry.User> dbUsers = usersCollection.Find(x => x.NFCid == nfcId).ToList();

if (dbUsers.Count != 1) {

return (new Result() {

authorized = false,

message = "no one has that nfc id",

success = false,

type = "unauthorized",

});

}

DBEntry.User dbUser = dbUsers[0];

List<DBEntry.IBoxCarrier> boxCarriers = usersCollection.Find(

x => x.Boxes.Any(y => y.Key == boxKey))

.ToList().ToList<DBEntry.IBoxCarrier>().Concat(

communitiesCollection.Find(

x => x.Boxes.Any(y => y.Key == boxKey))

.ToList().ToList<DBEntry.IBoxCarrier>()).ToList();

if (boxCarriers.Count != 1) {

return (new Result() {

authorized = true,

message = "key not found",

success = false,

type = "unsuccesful"

});

}

DBEntry.IBoxCarrier boxCarrier = boxCarriers[0];

if (boxCarrier is DBEntry.User user) {

if (user.Username == dbUser.Username) {

return (new Result() {

authorized = true,

message = "all good",

success = true,

type = "succesful"

});

}

else {

return (new Result() {

authorized = true,

message = "the user is not the owner of this box",

success = false,

type = "unsuccesful"

});

}

}

else if (boxCarrier is DBEntry.Community community) {

if (community.Users.Contains(dbUser.Username)) {

return (new Result() {

authorized = true,

message = "all good",

success = true,

type = "succesful"

});

}

else {

return (new Result() {

authorized = true,

message = "the user is not part of this box's community",

success = false,

type = "unsuccesful"

});

}

}

else {

return (new Result() {

authorized = true,

message = "error in IBoxCarrier type handling",

success = false,

type = "unsuccesful"

});

}

}

#endregion

#region move box

[HttpPost("MoveBox")]

public ActionResult MoveBoxResponder(string userKey, string boxKey, string to, string toType) {

if (!Utilities.Sanitized(userKey, boxKey, to, toType)) {

return StatusCode(405, "one or more arguments were null");

}

return Json(MoveBox(userKey, boxKey, to, toType));

}

internal ResultBase MoveBox(string userKey, string boxKey, string to, string toType) {

if (!(toType == "community" || toType == "user")) {

return (new Result() { authorized = false, success = false, type = "unsuccessful", message = "fromType or toType is not valid" });

}

(UserGatewayController.CheckUserResult userCheck, DBEntry.User dbUser) = UserGatewayController.CheckUser(userKey);

if (!userCheck.authorized) {

return (userCheck);

}

IMongoCollection<DBEntry.User> usersCollection = Program.Database.GetCollection<DBEntry.User>("users");

IMongoCollection<DBEntry.Community> communitiesCollection = Program.Database.GetCollection<DBEntry.Community>("communities");

List<DBEntry.User> boxesUser = usersCollection.Find(x => x.Boxes.Any(y => y.Key == boxKey)).ToList();

List<DBEntry.Community> boxesCommunity = communitiesCollection.Find(x => x.Boxes.Any(y => y.Key == boxKey)).ToList();

if (boxesCommunity.Count + boxesUser.Count != 1) {

return (new Result() {

authorized = true,

success = true,

type = "unsuccessful",

message = "box does not exist"

});

}

DBEntry.Box box = boxesCommunity.Select(x => x.Boxes.First(y => y.Key == boxKey)).Concat(boxesUser.Select(x => x.Boxes.First(y => y.Key == boxKey))).ElementAt(0);

if (boxesCommunity.Count == 1) {

DBEntry.Community community = boxesCommunity[0];

List<DBEntry.Box> boxes = community.Boxes.ToList();

boxes.Remove(box);

community.Boxes = boxes.ToArray();

communitiesCollection.ReplaceOne(x => x.UniqueName == community.UniqueName, community);

}

else {

DBEntry.User user = boxesUser[0];

List<DBEntry.Box> boxes = user.Boxes.ToList();

boxes.Remove(box);

user.Boxes = boxes.ToArray();

usersCollection.ReplaceOne(x => x.Username == user.Username, user);

}

if (toType == "community") {

List<DBEntry.Community> communities = communitiesCollection.Find(x => x.UniqueName == to).ToList();

if (communities.Count != 1) {

return (new Result() {

authorized = true,

success = false,

type = "unsuccessful",

message = "to does not exist"

});

}

DBEntry.Community community = communities[0];

List<DBEntry.Box> boxes = community.Boxes.ToList();

boxes.Add(box);

community.Boxes = boxes.ToArray();

communitiesCollection.ReplaceOne(x => x.UniqueName == community.UniqueName, community);

}

else {

List<DBEntry.User> users = usersCollection.Find(x => x.Username == to).ToList();

if (users.Count != 1) {

return (new Result() {

authorized = true,

success = false,

type = "unsuccessful",

message = "to does not exist"

});

}

DBEntry.User user = users[0];

List<DBEntry.Box> boxes = user.Boxes.ToList();

boxes.Add(box);

user.Boxes = boxes.ToArray();

usersCollection.ReplaceOne(x => x.Username == user.Username, user);

}

return (new Result() {

authorized = true,

success = true,

type = "successful",

message = "all good",

});

}

#endregion

#region rename box

[HttpPost("RenameBox")]

public ActionResult RenameBoxResponder(string userKey, string boxKey, string newName) {

if (!Utilities.Sanitized(userKey, boxKey, newName)) {

return StatusCode(405, "one or more arguments were null");

}

return Json(RenameBox(userKey, boxKey, newName));

}

internal ResultBase RenameBox(string userKey, string boxKey, string newName) {

(UserGatewayController.CheckUserResult userResult, DBEntry.User user) = UserGatewayController.CheckUser(userKey);

if (!userResult.success) {

return (userResult);

}

IMongoCollection<DBEntry.User> usersCollection = Program.Database.GetCollection<DBEntry.User>("users");

IMongoCollection<DBEntry.Community> communitiesCollection = Program.Database.GetCollection<DBEntry.Community>("communities");

List<DBEntry.User> boxesUser = usersCollection.Find(x => x.Boxes.Any(y => y.Key == boxKey)).ToList();

List<DBEntry.Community> boxesCommunity = communitiesCollection.Find(x => x.Boxes.Any(y => y.Key == boxKey)).ToList();

if (boxesCommunity.Count + boxesUser.Count != 1) {

return (new Result() {

authorized = true,

success = true,

type = "unsuccessful",

message = "nfc id does not exist"

});

}

DBEntry.Box box = boxesCommunity.Select(x => x.Boxes.First(y => y.Key == boxKey)).Concat(boxesUser.Select(x => x.Boxes.First(y => y.Key == boxKey))).ElementAt(0);

box.Name = newName;

if (boxesUser.Select(x => x.Boxes.First(y => y.Key == box.Key)).Count() == 1) {

DBEntry.User replaceUser = boxesUser.ElementAt(0);

for (int i = 0; i < replaceUser.Boxes.Length; i++) {

if (replaceUser.Boxes[i].Key == box.Key) {

replaceUser.Boxes[i] = box;

}

}

usersCollection.ReplaceOne(x => x.Username == replaceUser.Username, replaceUser);

}

else {

DBEntry.Community replaceCommunity = boxesCommunity.ElementAt(0);

for (int i = 0; i < replaceCommunity.Boxes.Length; i++) {

if (replaceCommunity.Boxes[i].Key == box.Key) {

replaceCommunity.Boxes[i] = box;

}

}

communitiesCollection.ReplaceOne(x => x.UniqueName == replaceCommunity.UniqueName, replaceCommunity);

}

return (new Result() {

authorized = true,

message = "all good",

success = true,

type = "successful"

});

}

#endregion

#region init new box

internal class NewBoxResult : ResultBase {

public string key;

public string owner; //who registered you

}

[HttpPost("NewBox")]

public ActionResult NewBoxResponder(string nfcId, string location) {

if (!Utilities.Sanitized(nfcId,location)) {

return StatusCode(405, "one or more arguments were null");

}

return Json(NewBox(nfcId, location));

}

internal NewBoxResult NewBox(string nfcId, string location) {

IMongoCollection<DBEntry.User> usersCollection = Program.Database.GetCollection<DBEntry.User>("users");

List<DBEntry.User> users = usersCollection.Find(x => x.NFCid == nfcId).ToList();

if (users.Count != 1) {

return (new NewBoxResult() {

authorized = false,

type = "unauthorized",

message = "nfc id did not exist",

success = false,

key = null,

owner = null

});

}

ObjectId objectid = ObjectId.GenerateNewId();

DBEntry.User user = users[0];

List<DBEntry.Box> boxes = user.Boxes.ToList();

boxes.Add(new DBEntry.Box() { Key = objectid.ToString(), Location = location, Name = "unnamed", \_id = objectid });

user.Boxes = boxes.ToArray();

usersCollection.ReplaceOne(x => x.Username == user.Username, user);

return (new NewBoxResult {

authorized = true,

key = objectid.ToString(),

message = "all good",

owner = user.Username,

success = true,

type = "succesful"

});

}

#endregion

}

}