eloy-aws-api-service/src/handlers/getDeviceById/getDeviceById.go (91.5%)   not tracked   not covered   covered

```go
package main

import (
    "eloy-aws-api-service/src/handlers/types"
    "fmt"
    "os"
    "encoding/json"

    "github.com/aws/aws-sdk-go/aws"
    "github.com/aws/aws-lambda-go/lambda"
    "github.com/aws/aws-lambda-go/events"
    "github.com/aws/aws-sdk-go/aws/session"
    "github.com/aws/aws-sdk-go/service/dynamodb"
    "github.com/aws/aws-sdk-go/service/dynamodb/dynamodbattribute"
    "github.com/aws/aws-sdk-go/service/dynamodb/dynamodbiface"
)

var ERROR_MISSING_ID_FIELD = 1
var ERROR_INTERNAL_SERVERS_DATABAE = 2
var ERROR_NO_ITEM_FOUNDED = 3


type SuccessResponse struct {
   Device   types.Device    `json:"data"`
}


type dynamoDBAPI struct {
    DynamoDB dynamodbiface.DynamoDBAPI
}

var databseStruct *types.DatabseStruct
var dynamodbapi *dynamoDBAPI

func init(){
    databseStruct = new(types.DatabseStruct)
    region := os.Getenv("AWS_REGION")
    dynamodbapi = new(dynamoDBAPI) // crate a setter that  can be used for inserting
    sess, err := session.NewSession(&aws.Config{Region: &region},)
    databseStruct.SessionError = err
    svc := dynamodb.New(sess)
    dynamodbapi.DynamoDB = dynamodbiface.DynamoDBAPI(svc)

    if err != nil {
        fmt.Println("There is an error while creating database session: " + err.Error())
    }

    // Get table name from OS's environment
    fetchedTableName :=os.Getenv("DEVICES_TABLE_NAME")
    if len(fetchedTableName)==0 {
        databseStruct.TableName =  nil;
        fmt.Println("It is not possible to fetch device tabel name")
    }else{
        databseStruct.TableName = aws.String(fetchedTableName)
    }
}


// get a device from DynamoDB database with provided id
func (ig *dynamoDBAPI) getFromDatabase(id string) ( *dynamodb.GetItemOutput, error) {

    var input = &dynamodb.GetItemInput{
        TableName: databseStruct.TableName,

        Key: map[string]*dynamodb.AttributeValue{
            "id": {
                S: aws.String(id),
            },
        },
    }

     result, err := ig.DynamoDB.GetItem(input)

     return result, err
}


// main AWS lambda function starting point.
// It gets an id from client, parse it and tries to get corresponding device fromdynamodb.
func GetDeviceById(request events.APIGatewayProxyRequest) (events.APIGatewayProxyResponse, error) {


    // there is some internal server error
    if databseStruct.SessionError != nil || databseStruct.TableName == nil {
        return events.APIGatewayProxyResponse{
            Body:       createErrorResponseJson(ERROR_INTERNAL_SERVERS_DATABAE),
            StatusCode: 404,
        }, nil
    }

    // get requested id from APIGatewayProxyRequest
    id := request.PathParameters["id"]

    // If no id provided, return HTTP error 404
    if id == "" {
        return events.APIGatewayProxyResponse{
            Body:       createErrorResponseJson(ERROR_MISSING_ID_FIELD),
            StatusCode: 404,
        }, nil
    }

    result, err := dynamodbapi.getFromDatabase(id)

    validationResult := validateDatabaseResult(result, err)

    return validationResult , nil
}


func validateDatabaseResult(result *dynamodb.GetItemOutput, err error)( events.APIGatewayProxyResponse) {

    // If an internal error occured in the database, return HTTP error 500
    // todo: log here
    if err != nil {
        return events.APIGatewayProxyResponse{
            Body:       createErrorResponseJson(ERROR_INTERNAL_SERVERS_DATABAE),
            StatusCode: 500,
        }
    }


    // If no item founded, return error 404
    if len(result.Item) == 0 {
        return events.APIGatewayProxyResponse{
            Body:       createErrorResponseJson(ERROR_NO_ITEM_FOUNDED),
            StatusCode: 404,
        }
    }


    // returned founded item as json file with 200 HTTP status code.
    // foundedDeviceAsJson, _ := json.Marshal(item)
    return events.APIGatewayProxyResponse{
        Body: createSuccessResponseJson(result),
        StatusCode: 200,
    }
}


func createErrorResponseJson(errorState int) (jsonString string) {


    if errorState == ERROR_MISSING_ID_FIELD {
            errorResponse := types.ErrorResponse { ErrorMessage: types.ErrorMessage { Code: 404,Message: "No ID Field Provided",},}
            errorResponseJson, _ := json.MarshalIndent(&errorResponse, "", "\t")
            return string(errorResponseJson)

    } else if errorState == ERROR_INTERNAL_SERVERS_DATABAE {
            errorResponse := types.ErrorResponse { ErrorMessage: types.ErrorMessage { Code: 500,Message: "Internal Server's Error occured",},}
            errorResponseJson, _ := json.MarshalIndent(&errorResponse, "", "\t")
            return string(errorResponseJson)

    } else {
            errorResponse := types.ErrorResponse { ErrorMessage: types.ErrorMessage { Code: 404,Message: "Desired device with provided id was not founded",},}
            errorResponseJson, _ := json.MarshalIndent(&errorResponse, "", "\t")
            return string(errorResponseJson)
    }
}


func createSuccessResponseJson(result *dynamodb.GetItemOutput) (jsonString string) {

    // create json file of database's returned Item
    item := types.Device{}
    dynamodbattribute.UnmarshalMap(result.Item, &item)

    successResponse := SuccessResponse {
        item,
    }
    successResponseJson, _ := json.MarshalIndent(&successResponse, "", "\t")

    return string(successResponseJson)
}

func main() {

    lambda.Start(GetDeviceById)
}
```