**Architecture Description**

**NodeJS Server:**

Simple Https Server:

- Express for Routing
- Pug for Html Page Templates
- Several NPM Modules (Session, FileUpload, Websocket

Simple Secure Websocket Server for Real Time Data display to the Client

The Main Routes are:

- Index: Welcome Page with the options to login yourself. Furthermore, there is a Testing button to path index/test for testing purposes
- Main:

**Lambda Functions:**

The used Lambda functions are mainly used to interact with the DynamoDb and the S3 Buckets. Each Lambda Functions uses the Same Role “WildRydesLambda” (We never changed the name) with the Full Access to DynamoDb and S3

Each Lambda Function can be invoked with the corresponding Lamda Libs. For Javascript it is the AWS.Lambda Lib. Each Invokation needs params (Javascript)

```
var params = {
  FunctionName: "authUser",
  InvocationType: "RequestResponse",
  LogType: "Tail",
  Payload: '{ "param1Name" : "' + param1 +'", "param2Name" : "' + param2 + '"}'
};
```

Method: authUser

Params: username, password

Response: on Success: 200

Response on Failure: 500, 403 (Wrong Credentials), 400 (username or password not set

Response Body: None

Description: Takes the username and querys DynamoDb. Checks if the password in the Db is the same as the parameter password if user is found. Returns 200 if it succeeded.

Method: addFile

Params: username, file(file.txt), wordCount (optional = 0), content(file content, optional = undefined)

Response on Success: 200

Response on Failure: 500 (Error while query), 400 (Params not set)

Response Body: Error Code or All Clear

Description: Takes the filename and filecontent from a user. If the file doesn't exist from the user it saves a new map entry for the user and sets the wordcount to it. Furthermore the content is saved into a s3 bucket with the name user-text-files. Filename in Buckets is always username_filename. If the file exists it gets overwritten.

Method: getFiles
Params: username
Response on Success: 200
Response on Failure: 500 (Error while query), 400 (Params not set)
Response Body: DynamoDb Data Items as a nested JSON Object with the calculated file features. They key of each object is the name of the file
Description: Querys the DynamoDb for all Userfile

Method: getFileByFileName
Params: username, file
Response: on Success: 200
Response on Failure: 500 (Query Failure), 400 (Wrong parameters)
Response Body: Json with Data Buffer of the files content
Description: Returns a Data Buffer from requested File.

Method: getFileCalcData
Params: username, file
Response: on Success: 200
Response on Failure: 500 (Query Failure), 400 (Wrong parameters)
Response Body: Json with Payload.Body with all the calculated fields, for now its

- WordCount
- finished

Description: Retrieves up to date Data from the DynamoDB

**Testing the NodeJS and Lambda Functions:**