

Introduction:

The camera on your shiny new phone can sense a user's mood based on their facial features, where mood can be characterized as either happy, sad or neutral to start with. We like you to design and implement a back-end application that leverages the phone's mood-sensing capability to collect mood data and provide insights:

- Upload a mood capture for a given user and location.

Mood Sense API

- Return the mood frequency distribution for a given user.

Returned by using Quicksight









- Return the proximity to locations (home, office, shopping center, ...) where a given user is happy. Specifically, create skeleton code to implement REST service cloud backend APIs for the application.

Preferably in Python or Java, but you are free to use any framework that you are comfortable with.

Feel free to make any reasonable assumptions about the scope of your implementation (e.g. a 3rd party library for obtaining location characteristics, given a set of GPS coordinates) It is not necessary for the code to be runnable, but you should have at least identified any 3rd party libraries and components that you intend to use and we would like to see at least some code skeleton and pseudo code.

Implementation Aspects:

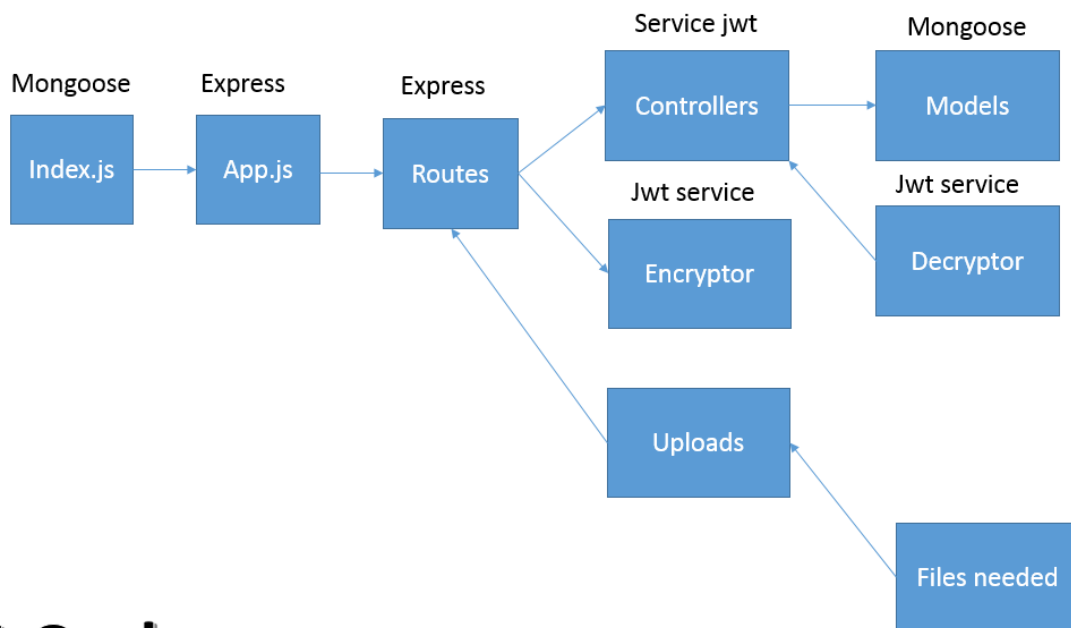
- Define API: **Mood Sense API**

 Controllers	5/21/2018 12:54 PM	File folder
 Middlewares	5/21/2018 12:51 PM	File folder
 Models	5/21/2018 12:53 PM	File folder
 Routes	5/21/2018 12:56 PM	File folder
 Services	5/21/2018 12:56 PM	File folder
 Uploads	5/21/2018 12:51 PM	File folder
 app.js	5/21/2018 12:58 PM	JScript Script File
 index.js	2/20/2018 7:37 AM	JScript Script File

- Create dev project: **API Structure**

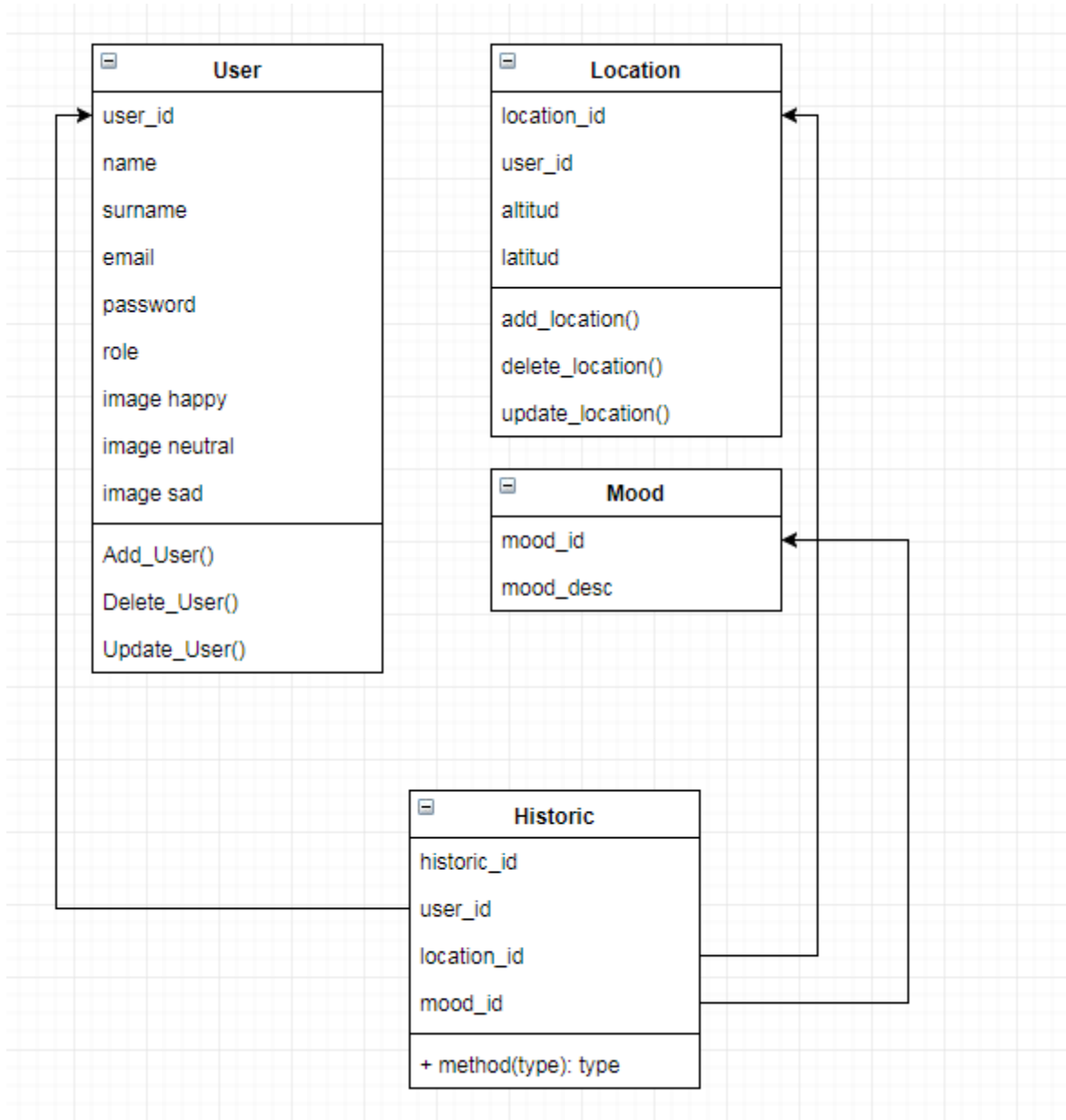
<https://github.com/ecadena212/TiempoDevelopment>

- Layout code structure



API Code structure

- Design data model and key data structures



- Define data persistence using any data store of your choice.

Amazon Elastic cache

- Define Implementation of operations

Step 1: User login to application by using app credentials or social network credentials

Step 2: User edits his profile updating 3 pictures (sad, neutral, happy) to train mood algorithm.

Step 3: User preload his preferred locations, so each time he is there and he unlock his phone a location and mood will be captured on historic table.

Step 4: User can activate an optional option that if it is activated phone will be storing a register each time user unlocks the phone and if location is not stored it will be stored automatically on user's profile.

- Input validation

Angular form validators: <https://angular.io/guide/form-validation>

- Authentication

jwt-simple library

- Authorization

App Login or social network credentials login (Cognito Aws service)

- Unit test

Unit testing will be done, using Codestar aws service:

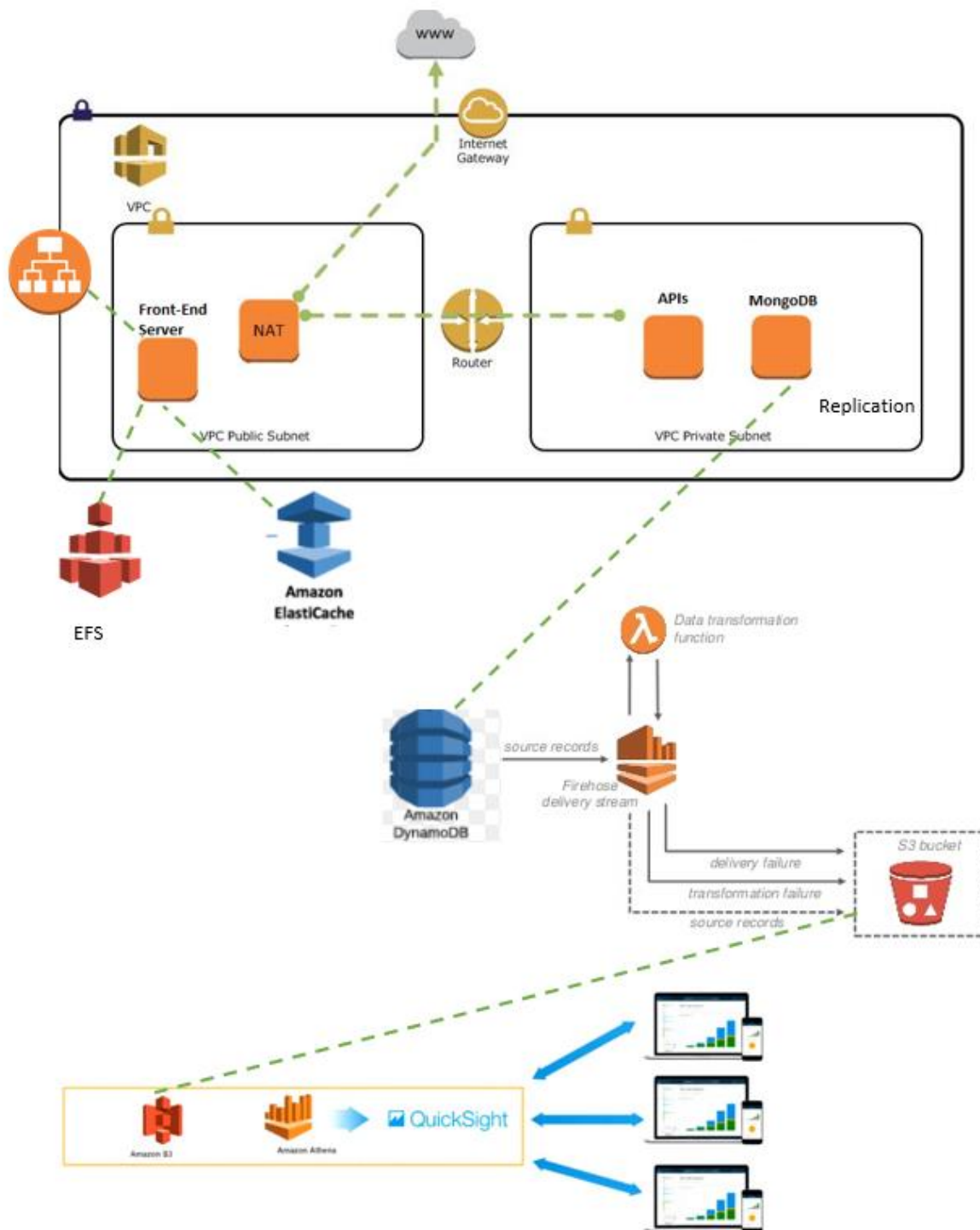
<https://aws.amazon.com/blogs/devops/performing-unit-testing-in-an-aws-codestar-project/>

- README for your design, implementation and assumptions.

Technologies Used:

APIs	MongoDB, Express, Mongoose, jwt-simple
AWS	VPC, EC2, NAT Gateway, Route 53, ELB, EBS, Lambda, EFS, DynamoDB, Kinesis, Firehose, S3 Quicksight, codestar


Infrastructure:





EC2- MongoDB

https://aws.amazon.com/marketplace/pp/B00NO1HJ56?ref=cns_srchow



EC2- APIs

https://aws.amazon.com/marketplace/pp/B00NO1HJ56?ref=cns_srchow



EC2- Angular

https://aws.amazon.com/marketplace/pp/B00NO1HJ56?ref=cns_srchow