# Concurrent and Distributed Programming

Homework 2

Proof-of-Concept for a Smart Home Monitoring Platform implemented using Cloud Computing Services and On-Premise Resources

**Team:**

- Nistor Marian-Sergiu (MSD)
- Samson Theodor (MISS)
- Mihaila Emilian Viorel (MISS)
- Patrauceanu Ionut Andrei (MISS)

**Coordinator:**

- Prof. Dr. Alboaie Lenuta

**1. Project Description** **2**

**2. Implementation** **2**

2.1. Architecture Overview 2

2.2. Workflows 3

2.2.1. Business Logic 3

2.2.2. Business Intelligence 5

2.3. Data Model 6

**3. Chosen Technologies Motivation** **6**

3.1. Compute 6

3.2. Storage 7

3.3. Object Detection and Emailing 7

**4. Appendices** **8**

# 1. Project Description

The project's intent is to provide a proof-of-concept implementation for a smart home monitoring and automated alarming platform, which makes use of cloud computing services. The concept revolves around the idea of a company which previously made use of fully in-house approach, and opted to migrate to cloud solutions. Thus, the platform is comprised of on-premise components (hosted on servers managed by the company in discussion), partially migrated components (e.g. servers hosted on EC2 instances), and fully migrated components (e.g. in-house HTTP API migrated to API Gateway HTTP API, with Lambda backends - fully serverless).

This experiment was conducted in order to analyze the benefits and disadvantages of migrating services to the cloud, taking into consideration several metrics, such as the ease of development, ease of maintenance, cost, latency, availability, scalability.

The cloud provider of choice is AWS (Amazon Web Services), being one of the most popular and reliable platforms.

# 2. Implementation

## 2.1. Architecture Overview

The application is comprised of multiple loosely coupled components, which satisfy the aforementioned specifications for a smart home monitoring platform, and provide means of the business to monitor key performance indicators, which consider traffic and customer adoption metrics. The platform's components are the following:

- **The Frames Publishing Authority**, which periodically receives video data from surveillance cameras, and stores it in a specialized S3 bucket. There are no requirements for the clients which make use of this component, apart from using JPEG encoding for the video frames and providing a valid session ID, as each camera is associated to a session).
- **The Alarming Authority**, which is a job that gets triggered every time the publishing authority uploads a new frame to the specialized S3 bucket. The job compares the given frame with the previously uploaded frame. If the most recent frame contains new entities (detected via a Rekognition client), the customer is alerted.
- **The Monitoring Dashboard**, which provides means for the customer to create a new session, or join an existing one, and see the video stream received from the cameras that were previously registered in the given session.
- **The Video Data Server**, which serves surveillance camera video data to the monitoring platform, via WebSockets.
- **The Session Information HTTP API**, which queries and updates data regarding the existing sessions, the registered cameras and each customer's personal information. The HTTP API exposes the following methods:
    - */create-session*, used for creating a new session, and specifying the customer's contact information

- */get-alert*, which retrieves the alerting status for a given customer (a customer can specify whether he should get alerted when an entity is detected by a camera)
- */get-cameras*, which retrieves the identifiers for the surveillance cameras registered for a specific session
- */get-session*, which checks for the existance of a given session
- */register-camera*, which registers a new surveillance camera for a specific session
- */set-alert*, which sets the alert state for a given session.

- **The Business Intelligence Dashboard**, which provides the mock company's managers and business intelligence engineers means of identifying and monitoring key performance indicators regarding the platform's evolution. The data that is being fed to the dashboard is periodically computed by a scheduled job, which makes use of machine learning-based predictions for anticipating traffic and customer adoption metrics.

## 2.2. Workflows

As previously stated, the aforementioned components serve two purposes: providing the customers means for monitoring and being alarmed when their personal cameras detect unwanted activity (business logic - *Appendix 1*), and providing the business means of identifying and monitoring key performance indicators (business intelligence - *Appendix 2*). The workflows that comprise each sub-system will be described in the following sections.

### 2.2.1. Business Logic

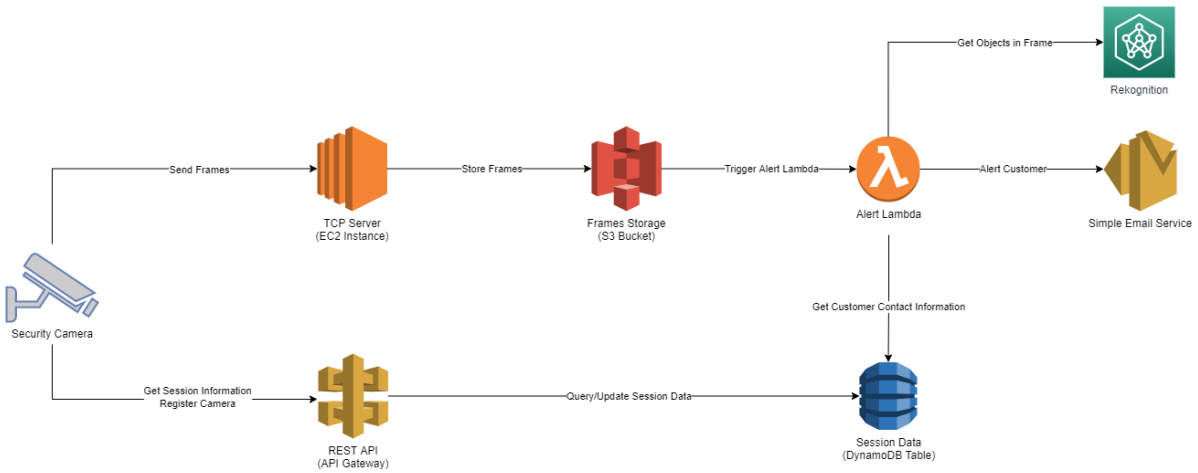



**Figure 1.** Frame Publishing and Customer Alerting Flow

*Figure 1* depicts the flow that covers the stage of storing data from the surveillance cameras, and, if required, alerting the customer, if an unwanted entity was captured by the personal video camera.

First, the Session Information HTTP API (depicting a fully migrated HTTP API) is called, validating and retrieving session data. Optionally, the security camera can be registered, if this is its first usage. Then, the Session Information HTTP API would update the information stored in the Session Data DynamoDB table.

After the session information is retrieved, the security camera starts sending video frames to the Frames Publishing Authority, hosted on an EC2 instance (thus, depicting a partially migrated resource). This authority stores the surveillance camera frames in a specialized S3 bucket.

Upon each video frame upload in the S3 bucket, the Alarming Authority will perform an object recognition task (provided by Rekognition), and decide which are the newly detected entities. If such unwanted entities are detected, customer data is retrieved from the Session Data DynamoDB table, and the customer is emailed, via Simple Email Service (SES).

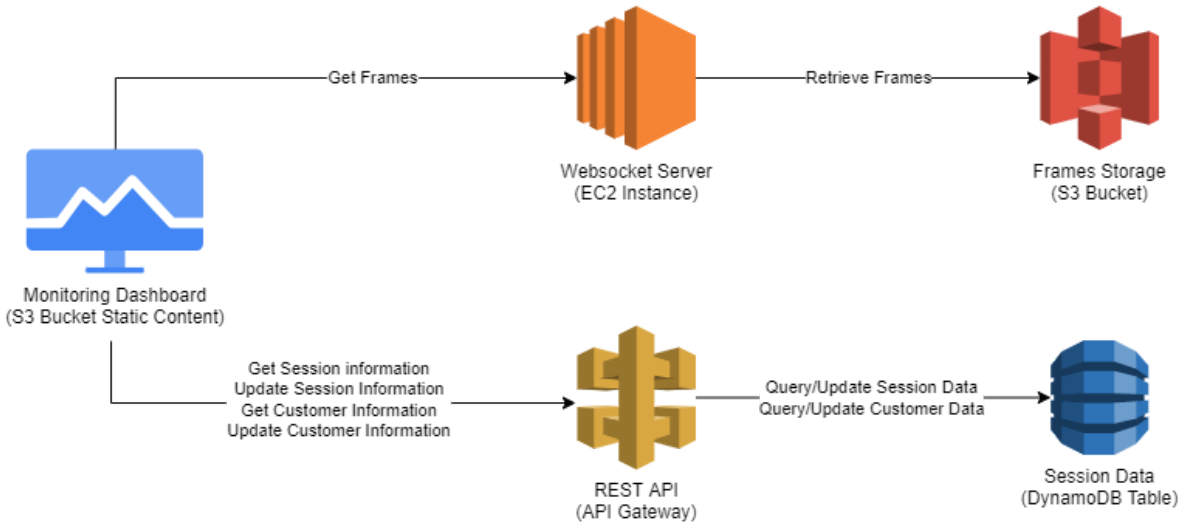



**Figure 2.** Monitoring Dashboard Flow

The surveillance camera video frames are accessible to the customer, within the monitoring dashboard. *Figure 2* describes the workflow that populates the Monitoring Dashboard (statically hosted in an S3 bucket) with video data.

Whenever a customer connects to the dashboard, he is prompted to either join an existing session, or create a new one. The session data is validated by specialized endpoint in the Session Information HTTP API, which queries the Session Data DynamoDB table.

After the session is initialized and validated, the Monitoring Dasboard initiates a connection to the Video Data Server, retrieving video data from each of the customer's personal surveillance cameras, in real time (which are stored in an S3 bucket).

### 2.2.2. Business Intelligence

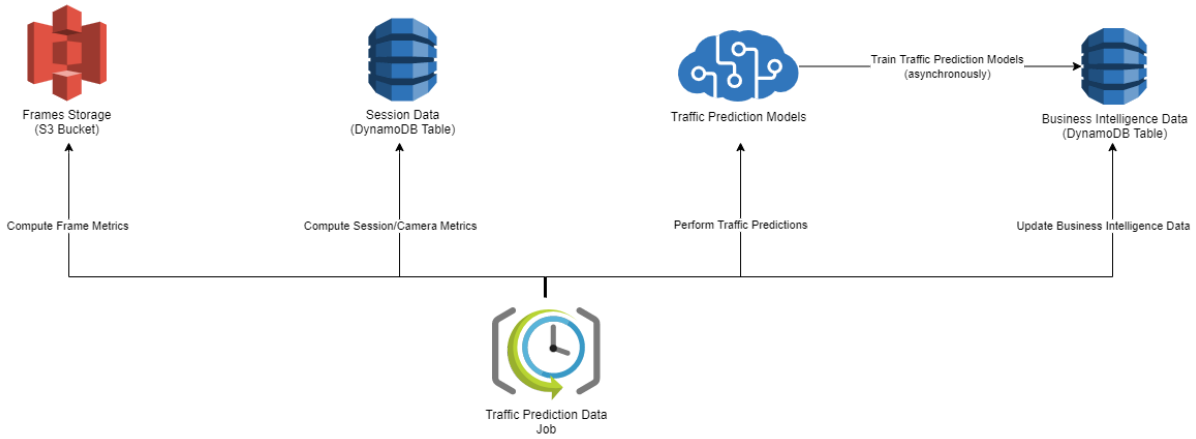



**Figure 3.** Traffic Prediction Flow

*Figure 3* describes the workflow that stores traffic indicators data and forecasts in a specialized Business Intelligence DynamoDB table. The workflow takes the form of a daily scheduled job, which runs on in-house servers (mimicing the state of a component which was not migrated to the cloud yet). The job accesses cloud services directly using an IAM user role assumed using an access key-secret key pair.

The data is queried from the Frames Storage S3 bucket and the Session Data DynamoDB table. Using pre-trained traffic forecasting models, it provides predictions for the traffic that will be encountered during the following days. The traffic prediction models are trained on a separate, non-scheduled job, and act as timeseries forecasting statistical models. As a predictor, the platform trains a Random Forest Regressor on historical data. Additional experiments were performed using the ARIMA model (Autoregressive Integrated Moving Average), which would allow for the elimination of the training jobs, as this model can perform forecasts directly on the evaluation data, given its autoregressive nature.

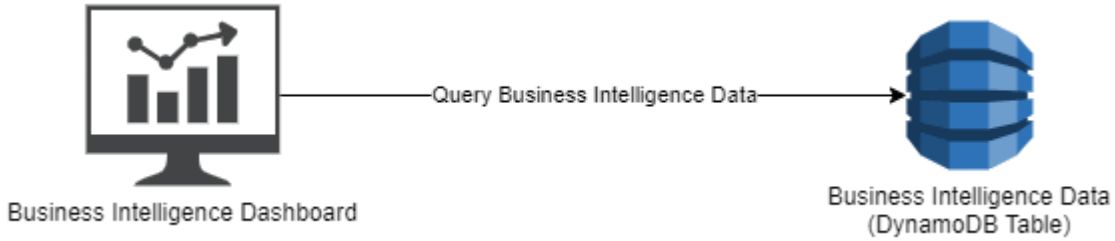



**Figure 4.** Business Intelligence Dashboard Flow

Finally, *Figure 4* depicts the flow that populates the Business Intelligence Dashboard with traffic and customer adoption data and predictions, for key performance business indicators (e.g. sessions count per day, cameras count per day, video frames received per day). The data is retrieved from the Business Intelligence Data DynamoDB table.

## 2.3. Data Model

The data model satisfies the need for storing both structured data (e.g. customer information, session data, camera data) and unstructured data (e.g. surveillance camera video data) in an organized, yet maintainable way.

The structured data is stored in DynamoDB tables, which allow for dynamic schemas, providing an easier way of extending the model. The platform depends on two separate tables, each handling one of the two subsystems:

- The Session Data Storage Table, storing data regarding the session ID, the registered camera ID's, the customer's email, and the alerting state
- The Business Intelligence Data Storage Tables (three tables storing metrics regarding sessions, cameras and video frames), storing the date for a given traffic metric, and the metric's value. The data can either be historical or forecasted.

The unstructured data is stored in the Video Frames Data S3 Bucket, which has a folder structure that organizes data based on session ID, camera ID, and the timestamp for a given video frame.

# 3. Chosen Technologies Motivation

## 3.1. Compute

Both servers that handle video data transfer (The Frames Publishing Authority - TCP server, and the Video Data Server - WebSockets server) have been hosted on EC2 instances, due to the nature of the video data traffic encountered by the platform. The traffic is constant and predictable, given the fact that the surveillance cameras are expected to continuously transmit data to the platform. Frequent horizontal scaling is not required. Opting for a Lamba approach could result in cold-start issues in this specific scenario. At most, eventual pre-scaling would suffice the need for provisioning enough compute capabilities. Opting for a Lamba approach could result in cold-start issues in this specific scenario. A real life example of a platform that makes use of EC2 instances is Netflix for hosting their cloud-based video content production tools.

For unpredictable loads, the platform makes use of an API Gateway HTTP API instance, that provides the required functionality using Lambda backends, while Lambda's are using a pay-per-use model, and the invocations are going to be less frequent and short-lived. Most modern companies opt for API Gateway when implementing microservice-based applications hosted in the cloud.

Additionally, a Lambda instance was used as a trigger when frames are added to the specialized S3 bucket.

.

## 3.2. Storage

As specified in the aforementioned sections, DynamoDB was used for storing structured data (session, camera and customer data), given the need for a easily maintainable storage method that provides a dynamic, extendable No-SQL schema. Additionally, Dynamo ensures millisecond access to data, which is critical to the platform, given the SLA's that could be imposed regarding the customer alert system.

Unstructured data was stored in S3 buckets, ensuring storage space scalability and removing the dependency on a physical machine's disk space.

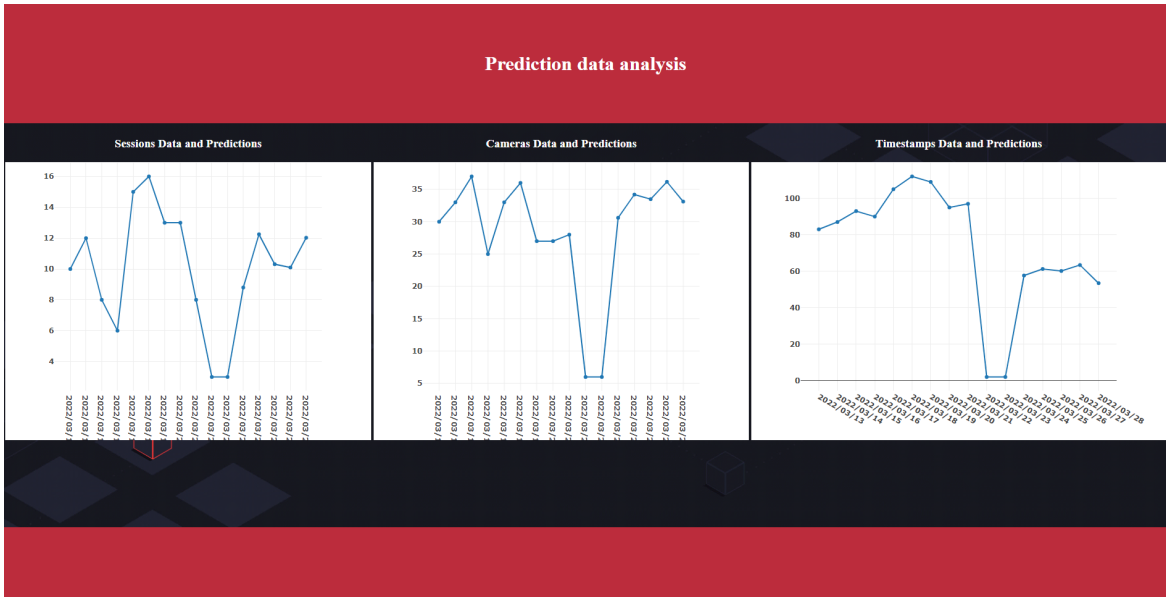
An example of a popular company that is a customer to both DynamoDB and S3, is Dropbox, which uses these storage services for performing replication, backups, and capacity management.

## 3.3. Object Detection and Emailing

The object detection tasks were performed using a Rekognition client. The video frames were downloaded in the Alert Lambda's temporary storage space, from the specialized S3 bucket, so as to perform object detection. If malicious entities are detected, the customers are alerted using SES (Simple Email Service).

# 4. Appendices


**Appendix 1.** Smart Home Monitoring Dashboard




**Appendix 2.** Business Intelligence Dashboard