

G5 Securitas



Course: 41492 - Software Engineering

Team: Diogo Vicente
Eduardo Santos
Eva Bartolomeu
João Ferreira
Pedro Sobral
Pedro Almeida





Index

1. Agile Principles + SCRUM Practices
2. Documentation
3. CI/CD
4. Architecture & Workflow
5. Networking in AWS
6. Scalability in AWS
7. Security in our System
8. API Gateway Paradigm
9. Solutions' Demo
10. Contributions of Each Team Member

Agile Principles + SCRUM Practices

<https://g5-securitas.atlassian.net/jira/software/projects/GS/boards/1>

List all events triggered in the client's building

 Anexar  Adicionar um problema secundário  Associar problema 

Descrição

As an client, I want list all events triggered in the client's building, so that it stays up to date.

1st cenário:

Given the client's buildings has the following events:

Building ID	Camera ID	Date	Time
1	123456789	01/12/2022	15:12:12
1	987654321	30/11/2022	16:12:12

When client asks for the events of their buildings



Then 2 events should have been found

And an event should have the camera ID: '123456789'; Date: 01/12/2022; Time: 15:12:12

And an event should have the camera ID: '987654321'; Date: 30/11/2022; Time: 16:12:12


Problemas associados

is blocked by

☒ GS-85 Endpoints to retrieve all events triggered   CONCLUÍDO ✓

Em Progresso ▾



Details ^


Responsável  Não atribuído
[Assign to me](#)

Etiquetas Nenhum

Sprint GS Sprint 5


Story point estimate 5

Development  Create branch
 Create commit ▾

Criador  Eva Bartolomeu

Created 18 de outubro de 2022 às 20:41

Updated 2 de dezembro de 2022 às 00:59

 Configurar

Documentation

<https://eduardosantoshf.github.io/es-project/>

The screenshot displays the documentation website for GS Securitas. The top navigation bar includes the GS Securitas logo, links to 'Docs' and 'Blog', and a GitHub icon. A sidebar on the left lists the site's structure, with 'Introduction to the Project' currently selected. The main content area features the title 'Introduction to the Project' and three paragraphs of introductory text. A green 'Edit this page' link is positioned below the text. At the bottom right, a 'Next' button links to the 'Architecture' page. The footer contains the copyright notice 'Copyright © 2022 GS Securitas'.

GS Securitas Docs Blog

GitHub

Introduction to the Project

Architecture

Architecture Design

Architecture Specification

AWS cloud architecture

Project Members

Technologies Used

Documentation

C/CD

Client Web UI

Management Web UI

Sites Managment API

Intrusion Managment API

Definitions about User Stories

User Story Readiness Criteria

User Story Acceptance Criteria

User Story priorities

User Story points

Introduction to the Project

Introduction to the Project

This project will tackle the development of a software solution for a security company named SecCom. SecCom is a company that ensures critical buildings are not broken into, through the installation and operation of CCTV cameras on-premises.

Although, SecCom is still not foresting the technological advancements in the security monitoring field, having several people monitoring the cameras of the most critical buildings.

The goal of your team is to help SecCom with their transition to the digital world, creating an automatic system that can identify intruders without human-intervention and act accordingly.

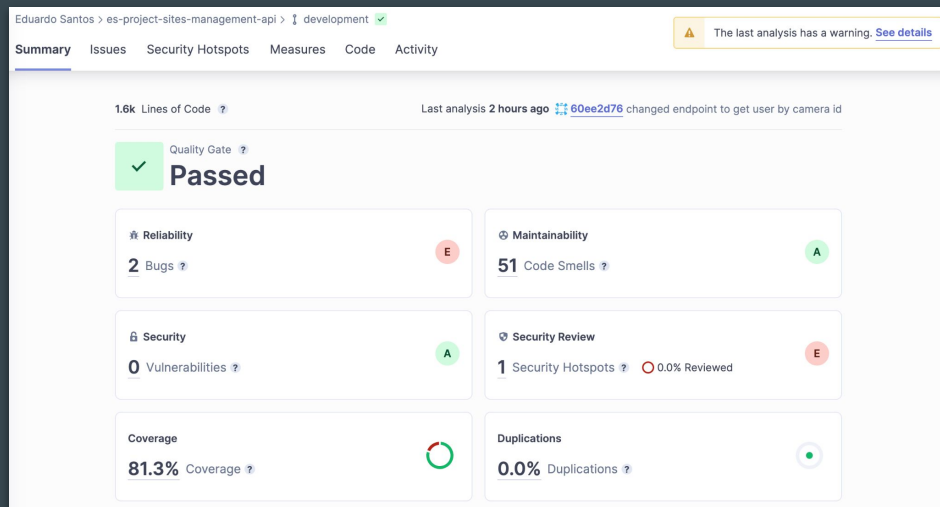
With their digital transition, SecCom will also install several light and sound alarms on the spaces they are protecting and wishes for them to be automatically activated every time an intruder is detected. Besides this, SecCom also expects that your team develops a top-to-bottom solution they can use manage the cameras and alarms installed on-premises and to manage all their clients and buildings monitored.

[Edit this page](#)

Next
[Architecture »](#)

Copyright © 2022 GS Securitas

CI/CD - Testing



```
name: Sites Management API Coverage and Testing

on:
  push:
    paths:
      - '.github/workflows/sites-management-api-coverage.yaml'
      - sitesManagementAPI/**
      - '**.md'
      - '**.pdf'
      - '**.docx'
      - '**.gitignore'
      - '**.txt'
  pull_request:
    paths:
      - sitesManagementAPI/**
      - '**.md'
      - '**.pdf'
      - '**.docx'
      - '**.gitignore'
      - '**.txt'
```

```
steps:
  - name: Pulling git repo
    uses: actions/checkout@v3
    with:
      fetch-depth: 0

  - name: Install tox and any other packages
    run: |
      pip install tox

  - name: Run tox
    run: tox -e py

  - name: SonarCloud Scan
    uses: SonarSource/sonarcloud-github-action@master
    env:
      GITHUB_TOKEN: ${ secrets.GITHUB_TOKEN } # Needed to get PR information, if any
      SONAR_TOKEN: ${ secrets.SONAR_TOKEN_SITES_MANAGEMENT_API }
    with:
      projectBaseDir: sitesManagementAPI/
```

CI/CD - Elastic Container Service

Clusters > intrusion-management-api-cluster

Cluster : intrusion-management-api-cluster

[Update Cluster](#)[Delete Cluster](#)

Get a detailed view of the resources on your cluster.

Cluster ARN `arn:aws:ecs:eu-west-3:389574951348:cluster/intrusion-management-api-cluster`

Status **ACTIVE**

Registered container instances 1

Pending tasks count 0 Fargate, 0 EC2, 0 External

Running tasks count 0 Fargate, 1 EC2, 0 External

Active service count 0 Fargate, 1 EC2, 0 External

Draining service count 0 Fargate, 0 EC2, 0 External

[Services](#)[Tasks](#)[ECS Instances](#)[Metrics](#)[Scheduled Tasks](#)[Tags](#)[Capacity Providers](#)[Create](#)[Update](#)[Delete](#)[Actions](#)

Last updated on December 18, 2022 10:00:38 PM (0m ago)



Filter in this page

Launch type

ALL

Service type

ALL

< 1:1 >

<input type="checkbox"/>	Service Name	Status	Service type	Task Definition	Desired tasks	Running tasks	Launch type	Platform version
<input type="checkbox"/>	intrusion-man-api-service	ACTIVE	REPLICA	intrusion-man-api-task:31	1	0	EC2	--

CI/CD - Deployment

```
name: Deploy HDM to AWS

on:
  workflow_dispatch:
  inputs:
    logLevel:
      description: 'Log level'
      required: true
      default: 'warning'
      type: choice
      options:
        - info
        - warning
        - debug

pull_request:
  branches: [ main ]

paths:
  - HumanDetection/**
  - .github/human-detection-deployment.yaml
  - '**.md'
  - '**.pdf'
  - '**.docx'
  - '**.gitignore'
  - '**.txt'

push:
  branches: [ main ]

paths:
  - HumanDetection/**
  - .github/human-detection-deployment.yaml
  - '**.md'
  - '**.pdf'
  - '**.docx'
  - '**.gitignore'
  - '**.txt'

env:
  ECR_REGISTRY_ALIAS: p3i3q7f7
  ECR_REPOSITORY: human-detection-repository # set this to your Amazon ECR repository name
  ECS_SERVICE: human-detection-service # set this to your Amazon ECS service name
  ECS_CLUSTER: human-detection-cluster # set this to the path to your Amazon ECS task definition
  ECS_TASK_DEFINITION: .aws/human-detection-task-definition.json # file, e.g. .aws/task-definition.json
  HDM_CONTAINER_NAME: human-detection-container # set this to the name of the API container in the
  # containerDefinitions section of your task definition
  RABBIT MQ_EXCHANGE_NAME: human-detection-exchange
  RABBIT MQ_QUEUE_NAME: human-detection-queue
```

```
jobs:
  deploy:
    name: Deploy
    runs-on: ubuntu-latest

    steps:
      - name: Checkout
        uses: actions/checkout@v3

      - name: Configure AWS credentials (us-east-1)
        uses: aws-actions/configure-aws-credentials@v1
        with:
          aws-access-key-id: ${ secrets.AWS_ACCESS_KEY_ID }
          aws-secret-access-key: ${ secrets.AWS_SECRET_ACCESS_KEY }
          aws-region: us-east-1

      - name: Login to Amazon ECR
        id: login-ecr-public
        uses: aws-actions/amazon-ecr-login@v1
        with:
          registry-type: public

      - name: Build, tag, and push docker hdm image to Amazon ECR Public
        id: build-hdm-image
        env:
          ECR_REGISTRY: ${ steps.login-ecr-public.outputs.registry }
          IMAGE_TAG: ${ github.sha }
          RABBIT MQ_URL: ${ secrets.RABBIT MQ_URL }
          RABBIT MQ_USERNAME: ${ secrets.RABBIT MQ_USERNAME }
          RABBIT MQ_PASSWORD: ${ secrets.RABBIT MQ_PASSWORD }
          REDIS_URL: ${ secrets.REDIS_URL }
        run: |
          cd HumanDetection/human-detection-module
          docker build -t $ECR_REGISTRY/$ECR_REGISTRY_ALIAS/$ECR_REPOSITORY:$IMAGE_TAG .
          docker push $ECR_REGISTRY/$ECR_REGISTRY_ALIAS/$ECR_REPOSITORY:$IMAGE_TAG
          echo "Images $ECR_REGISTRY/$ECR_REGISTRY_ALIAS/$ECR_REPOSITORY:$IMAGE_TAG" >> $GITHUB_OUTPUT

      - name: Configure AWS credentials (change to eu-west-3 region)
        uses: aws-actions/configure-aws-credentials@v1
        with:
          aws-access-key-id: ${ secrets.AWS_ACCESS_KEY_ID }
          aws-secret-access-key: ${ secrets.AWS_SECRET_ACCESS_KEY }
          aws-region: eu-west-3

      - name: Update HDM container in ECS task definition
        id: build-task-def
        uses: aws-actions/amazon-ecs-render-task-definition@v1
        with:
          task-definition: ${ env.ECS_TASK_DEFINITION }
          container-name: ${ env.HDM_CONTAINER_NAME }
          image: ${ steps.build-hdm-image.outputs.image }
          environment-variables: |
            RABBIT MQ_URL=${ secrets.RABBIT MQ_URL }
            RABBIT MQ_USERNAME=${ secrets.RABBIT MQ_USERNAME }
            RABBIT MQ_PASSWORD=${ secrets.RABBIT MQ_PASSWORD }
            RABBIT MQ_EXCHANGE_NAME=${ env.RABBIT MQ_EXCHANGE_NAME }
            RABBIT MQ_QUEUE_NAME=${ secrets.RABBIT MQ_QUEUE_NAME }
            REDIS_URL=${ secrets.REDIS_URL }
            INTRUSION_MANAGEMENT_API_URL=${ env.INTRUSION_MANAGEMENT_API_URL }

      - name: Deploy Amazon ECS task definition
        uses: aws-actions/amazon-ecs-deploy-task-definition@v1.4.10
        with:
          task-definition: ${ steps.build-task-def.outputs.task-definition }
          service: ${ env.ECS_SERVICE }
          cluster: ${ env.ECS_CLUSTER }
          wait-for-service-stability: false
```

AWS Architecture

<https://eduardosantoshf.github.io/es-project/aws-cloud-architecture/>

Security

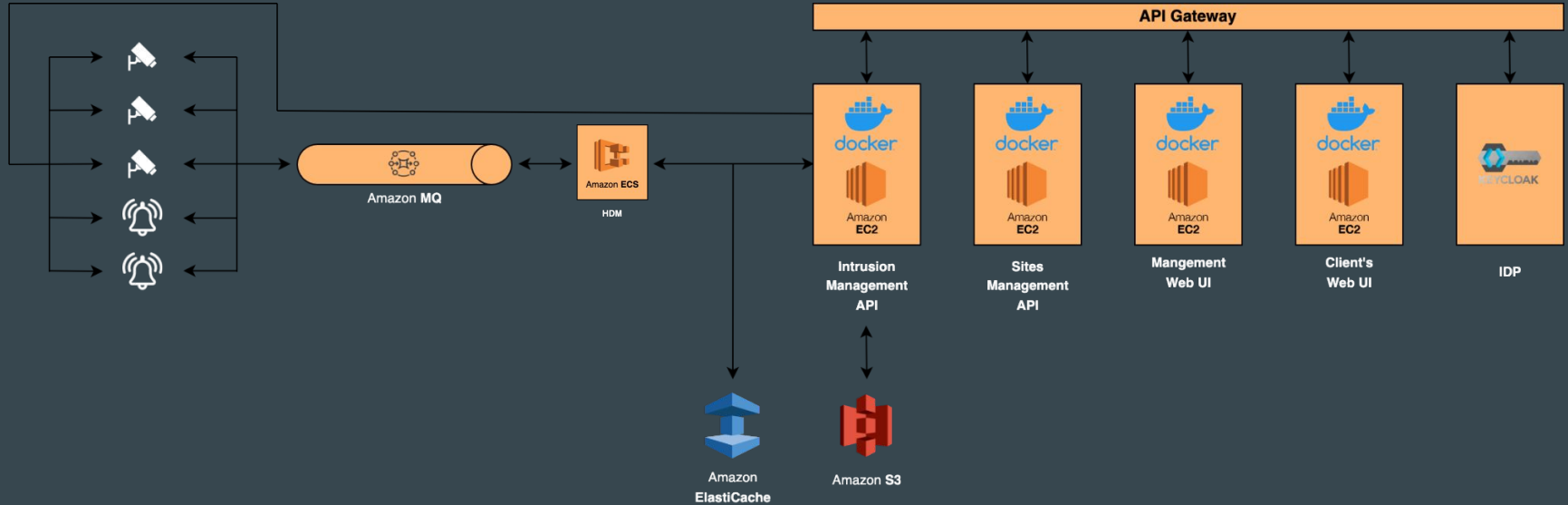
AA in our system was implemented using an IdP

- Keycloak
- OpenID Connect specification
- Access token



GET	/sites-man-api/users/{user_id}/cameras	Read User Cameras	✓	🔒
GET	/sites-man-api/users/{user_id}/alarms	Read User Alarms	✓	🔒
GET	/sites-man-api/users/{user_id}/properties	Read User Properties	✓	🔒

Architecture & Workflow



Solutions' Demo

Client Web UI: <http://securitas-lb-1725284772.eu-west-3.elb.amazonaws.com/>

Management Web UI: <http://ec2-13-37-224-243.eu-west-3.compute.amazonaws.com/admin>

Client demo: <https://youtu.be/TfNBq4YmH1s>

Admin demo: <https://youtu.be/hIMHacRem7o>

Contributions of Each Team Member

- Diogo Vicente - Sites Management API, IDP, CI/CD
- Eduardo Santos - HDM, CI/CD, Amazon ECS/EC2/MQ
- Eva Bartolomeu - Client UI, Management UI, IDP, Intrusion Management API
- João Ferreira - AWS, CI/CD, Sites Management API
- Pedro Sobral - Cameras, Intrusion Management API, CI, Management UI
- Pedro Bastos - Cameras & HDM, Elastic Cache, AWS Gateway, CI