Project #1

# Mvc API .net core: stamps system

Description: Arduino hardware read rfid badge and call an api, api check the owner of the badge id, register the stamp and return status, datetime recorded and employee name.

## Api:

Method: POST

Route: /stamp/new

Body:

{

badgeId: 123456

stampId: “A2:A3:A4:A5”

}

Result:

{

Status: true or false

Employeename: “example name”

Datetime: datetime

}

Action Steps:

1. on received post, api select from “vw\_assignedbadge” the owner of the badge, take first record if exist, if not exist return false.
2. If exist select from table “stamper” the name of the stamper based on stampid else return false
3. Select on table “stamp” if exist any record for employee based on employeeid return on step 1, if exist check if last one based on timestamp is with status “IN”
4. Insert into table “stamp” a new record with the data obtained from previous steps, if step 3 found that last record was “IN” then insert the new record as “OUT” else as “IN”

# Mvc API .net core: open door

Description: using the same table of previous api, if badge id exist call another api

## Api

Method: POST

Route: /security/opendoor

Body:

{

badgeId: 123456

stampId: “A2:A3:A4:A5”

}

Result:

{

Status: true or false

Employeename: “example name”

Datetime: datetime

}

Action steps:

1. on received post, api select from “vw\_assignedbadge” the owner of the badge, take first record if exist, if not exist return false.
2. If exist select from table “stamper” the name of the stamper based on stampid else return false
3. If previous steps ok, call api:
   1. <http://10.0.155.249/relay/0?turn=on> with authorization berear "Authorization: Basic YWRtaW46UGFzc3dvcmQwMQ=="
4. Return value true

# Database

Database is postgresql, connection string as to be in a config value

Tables:

**vw\_assignedbadge**

A screenshot of a computer

Description automatically generated

**Stamper**

A screenshot of a computer

Description automatically generated

-- This script only contains the table creation statements and does not fully represent the table in the database. It's still missing: indices, triggers. Do not use it as a backup.

-- Sequence and defined type

CREATE SEQUENCE IF NOT EXISTS stamper\_id\_seq;

-- Table Definition

CREATE TABLE "public"."stamper" (

"id" int4 NOT NULL DEFAULT nextval('stamper\_id\_seq'::regclass),

"stamperId" text,

"stamperName" text,

PRIMARY KEY ("id")

);

**Stamp**

**A screenshot of a computer

Description automatically generated**

-- This script only contains the table creation statements and does not fully represent the table in the database. It's still missing: indices, triggers. Do not use it as a backup.

-- Sequence and defined type

CREATE SEQUENCE IF NOT EXISTS stamps\_id\_seq;

-- Table Definition

CREATE TABLE "public"."stamps" (

"id" int4 NOT NULL DEFAULT nextval('stamps\_id\_seq'::regclass),

"badgeId" float4,

"TimeStamp" timestamp DEFAULT now(),

"Stamper" text,

"GivenName" text,

"Surname" text,

"employeeId" float4,

"EMPLOYEEEMAIL" text,

"status" text,

PRIMARY KEY ("id")

);

# Project:

Project has to be developed in mvc .net core, with controller folder, model folder

Project has to run in Kubernetes environment

We need a full project source code, with dockerfile, docker-compose and all needed file and component to run it.

Database already exist

Next project will be the web interface for view the record inserted, edit them and export them.