UNIVERSIDADE ESTADUAL DO MARANHÃO
CENTRO DE CIÊNCIAS TECNOLÓGICAS
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DA COMPUTAÇÃO
E SISTEMAS - PECS

# IMPLANTAÇÃO DO PROJETO URCA - USO RACIONAL DE CARROS PARA MOBILIDADE URBANA NAS CIDADES INTELIGENTES

LUIZ CARLOS CHAVES LIMA JUNIOR

SÃO LUÍS - MA
2021

UNIVERSIDADE ESTADUAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DA COMPUTAÇÃO
E SISTEMAS - PECS

LUIZ CARLOS CHAVES LIMA JUNIOR

# IMPLANTAÇÃO DO PROJETO URCA - USO RACIONAL DE CARROS PARA MOBILIDADE URBANA NAS CIDADES INTELIGENTES

Trabalho apresentado ao curso de Mestrado Profissional em Engenharia de Computação e Sistemas na Universidade Estadual do Maranhão como pré-requisito para obtenção do título de Mestre sob orientação do Prof. Dr. Carlos Henrique Rodrigues de Oliveira.

SÃO LUÍS - MA
2021

Lima Júnior, Luiz Carlos Chaves.
Implantação do projeto URCA – uso racional de carros para mobilidade urbana nas cidades inteligentes / Luiz Carlos Chaves Lima Júnior. – São Luís, 2021.

68 f

Dissertação (Mestrado) – Curso de Engenharia da Computação e Sistemas, Universidade Estadual do Maranhão, 2021.

Orientador: Prof. Dr. Carlos Henrique Rodrigues de Oliveira.

1.URCA. 2.Banco de dados. 3.Analytics. 4.Rastreamento de objetos. 5.OpenCV. 6.Mobilidade urbana. 7.URCARONA. I.Título.

CDU: 004.8:656.13

**LUIZ CARLOS CHAVES LIMA JUNIOR**

# IMPLANTAÇÃO DO PROJETO URCA - USO RACIONAL DE CARROS PARA MOBILIDADE URBANA NAS CIDADES INTELIGENTES

Dissertação apresentada ao Mestrado Profissional em Engenharia de Computação e Sistemas (PECS) da Universidade Estadual do Maranhão, como registro para obtenção do grau de Mestre em Engenharia de Computação e Sistemas.

Trabalho aprovado. São Luís – MA, 31 de março de 2021.

**Prof. Dr. Carlos Henrique Rodrigues de Oliveira**

Orientador

**Profa. Dra. Patrícia Helena Moraes Rêgo**

Primeiro membro

AUREA CELESTE DA COSTA RIBEIRO:86179993220

Assinado de forma digital por AUREA CELESTE DA COSTA RIBEIRO:86179993220
Dados: 2021.04.05 18:49:14 -03'00'

**Profa. Dra. Aurea Celeste da Costa Ribeiro**

Segundo membro

## DEDICATÓRIA

À minha mãe Conceição, por sempre estar ao meu lado e me dar todo apoio necessário;
À minha segunda família, Valéria, Erick e Rafael pelos incentivos e pensamentos positivos;
À família URCA, Professor Carlos Henrique, Aurelianny e Ana Paula.

## AGRADECIMENTOS

Ao professor Carlos Henrique pelas orientações e pela confiança depositada em mim para execução deste projeto;

Às minhas colegas de equipe Ana Paula e Aurelianny por apoio constante;

A todos que participam do projeto, pelo suporte oferecido.

## RESUMO

Iniciado no segundo semestre de 2015, o projeto URCA é uma solução de baixo custo voltada para melhoria da mobilidade urbana, na qual o intuito é propor às pessoas a utilização de caronas solidárias, supondo-se que uma parte considerável de motoristas que trafegam diariamente nas vias públicas, na maior parte de tempo, circula sozinha nos veículos.

Dessa forma, com a adesão destes motoristas à ideia da solução, menos carros entrariam nas vias públicas e, consequentemente, traria uma melhor fluidez ao trânsito e benefícios à saúde e ao meio ambiente. Além disso, também é sugerida, ao órgão regulador de trânsito, a concessão de benefícios fiscais, como redução de tributos relativos aos veículos para quem aderir à solução, de forma a atrair mais adeptos.

Entretanto, é necessário que haja um mecanismo para coletar informações como: quantas pessoas trafegam sozinhas nos veículos e suas respectivas placas de identificação. Para isso, o projeto URCA propõe a implementação de uma solução composta por um sistema de monitoramento por câmeras, em conjunto com um algoritmo, que fará a coleta destas informações e as enviará a um banco de dados, que por sua vez, será integrado à aplicação URCA Analytics, esta última responsável por permitir a visualização e manipulação das informações coletadas, além de possuir um aplicativo de caronas.

Sendo assim, este trabalho concentrou-se na implantação da solução no Campus Paulo VI da Universidade Estadual do Maranhão, que contou com a criação de um banco de dados, utilizando o MySQL Workbench, assim como o desenvolvimento de aplicações em Python, uma em conjunto com técnicas de visão computacional e detecção e rastreamento de objetos, que permitiu a automatização do envio das capturas ao banco de dados, bem como um algoritmo de geração de capturas aleatórias, a aplicação URCA Analytics e a demonstração de funcionalidades básicas do aplicativo de caronas.

Desse modo, obteve-se resultados positivos na captura de informações dos dois carros usados nos testes, além da concessão do registro do software do aplicativo de caronas intitulado “URCARONA v.1.0”, junto ao INPI, sendo todas estas etapas e resultados relatados neste documento.



Palavras-chave: URCA; Banco de Dados; Analytics; Rastreamento de Objetos; OpenCV; Mobilidade Urbana, URCARONA.

## *ABSTRACT*

*Started in the second half of 2015, the URCA project is a low-cost solution aimed at improving urban mobility, in which the intention is to propose to people the use of free rides, assuming that a considerable number of drivers who travel daily in the city's public roads, for the most part, circulate alone in vehicles.*

*Thus, with the adherence of these drivers to the idea of the solution, fewer cars would enter public roads and, consequently, would bring better fluidity to traffic and benefits to health and the environment. In addition, it is also suggested to the traffic regulator to grant tax benefits, such as reducing vehicle taxes for those who adhere to the solution, in order to attract more followers.*

*However, there is a need for a mechanism to collect information such as: how many people travel alone in vehicles and their respective identification plates. For this, the URCA project proposes the implementation of a solution consisting of a camera monitoring system, together with an algorithm, which will collect this information and send it to a database, which in turn, will be integrated into the URCA Analytics application, the latter responsible for allowing the visualization and manipulation of the collected information, in addition to having a free ride application.*

*Therefore, this work focused on the implementation of the solution at the Paulo VI Campus of the State University of Maranhão, which included the creation of a database, using the MySQL Workbench, as well as the development of applications in Python, one together. with computer vision techniques and object detection and tracking, which allowed the automation of sending captures to the database, as well as an algorithm for generating random captures, the URCA Analytics application and the demonstration of basic features of the hitchhiking application.*

*In this way, positive results were obtained in capturing information from the two cars used in the tests, in addition to granting the registration of the software for the ride application entitled "URCARONA v.1.0", with the INPI, with all these steps and results reported in this document.*



*Keywords: URCA; Database; Analytics; Object Tracking; OpenCV; Urban Mobility, URCARONA.*

## LISTA DE FIGURAS

## LISTA DE TABELAS

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| ANSI | *American National Standards Institute* |
| CPE | *Customer Premises Equipament* |
| EER | *Enhanced entity-relationship* |
| ERB | Estação Rádio Base |
| R-CNN | *Region-based Convolutional Neural Networks* |
| DBA | *DataBase Administrator* |
| DDL | *Data Definition Language* |
| ISO | *International Organization for Standardization* |
| OCR | *Optical Character Recognition* |
| RGB | *Red Green Blue* |
| SGBD | Sistema Gerenciador de Base de dados |
| SQL | *Structured Query Language* |
| SSD | *Single Shot MultiBox Detector* |
| UEMA | Universidade Estadual do Maranhão |
| URCA | Uso Racional de Carros |
| YOLO | *You Only Look Once* |

**SUMÁRIO**

# 1. INTRODUÇÃO

O projeto URCA – Uso Racional de Carros nas Cidades Inteligentes – é uma solução de baixo custo voltada para melhoria da mobilidade urbana, onde a ideia principal é propor às pessoas a utilização de caronas solidárias, pressupondo-se que uma parte considerável de motoristas que trafegam diariamente nas vias públicas, na maior parte de tempo, faz seu trajeto sem passageiros [1, 2].

Desse modo, se estes motoristas passarem a adotar o uso de caronas ou oferecerem caronas a outros motoristas nas mesmas condições, menos carros adentrariam as vias e, consequentemente, haveria impactos positivos como: melhor fluidez no trânsito, redução da emissão de poluentes gerada pela queima de combustíveis dos veículos e redução dos custos com deslocamento, que são relativamente altos e causam grandes prejuízos à economia brasileira [3].

Entretanto, é esperado que haja uma certa resistência por parte dos motoristas em abrir mão da utilização de seus carros particulares para aderir ao uso ou disponibilização de caronas solidárias. Em vista disso, sugere-se a concessão de benefícios fiscais, pelos órgãos reguladores de trânsito, como, por exemplo, a redução de taxas relativas aos veículos, como IPVA, licenciamento ou descontos em multas, de forma tornar a solução mais atrativa para a população [4].

Outro fator importante é que, para que os órgãos reguladores de trânsito possam conceder tais benefícios, faz-se necessário um mecanismo para obter informações essenciais sobre os carros, como a quantidade de ocupantes no interior dos veículos e suas respectivas placas de identificação. Nesse caso, o URCA atuará como um sistema de monitoramento que será capaz de fornecer essas informações [4].

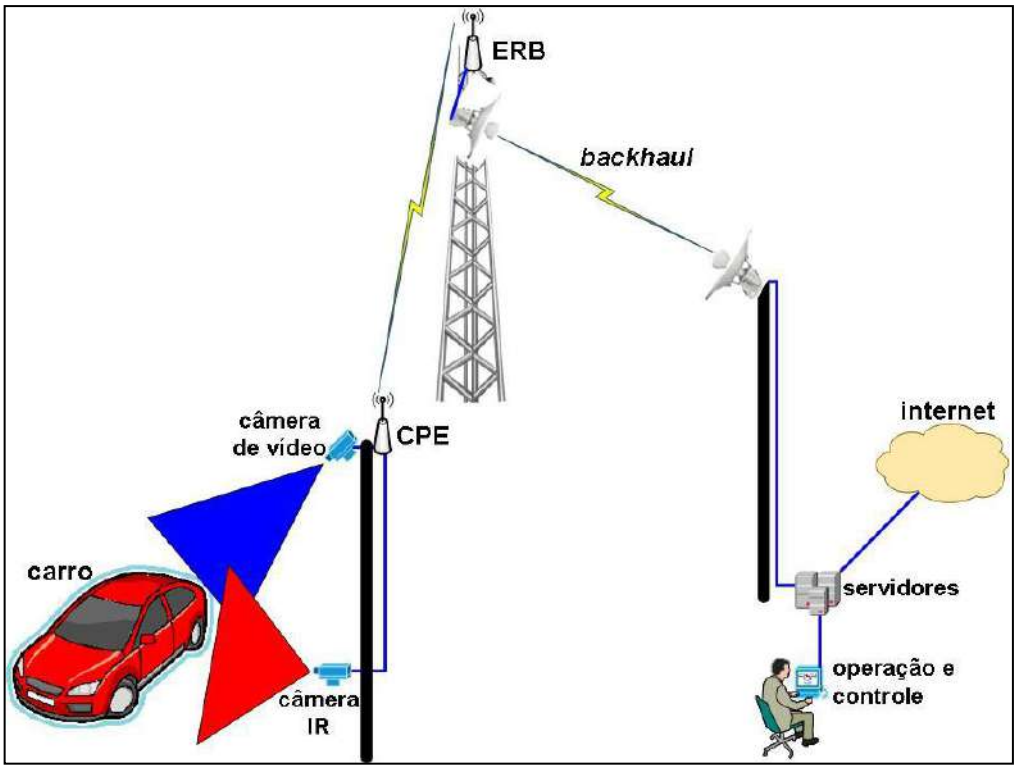
O processo de coleta dessas informações será executado da seguinte forma: ambas as informações serão coletadas, pelo URCA, de maneira similar à captura de placas feita pelos radares de avanço colocados em redutores de velocidade e em semáforos, por meio da atuação de duas câmeras [1].

A primeira delas será uma câmera de infravermelho para atender a necessidade de visualizar a parte interna dos automóveis, que geralmente contém película insulfilm escurecida (fumê) nos vidros e a segunda será uma câmera de vídeo *RGB* para leitura dos caracteres das placas [1].

Em seguida, os quadros capturados pelas câmeras serão analisados por um algoritmo que irá extrair as informações de cada um deles e as enviará a um banco de dados conectado

a uma aplicação, para que seja visualizada e manipulada pelos operadores. A topologia do URCA pode ser vista na Figura 1.

Figura 1. Topologia URCA.




Fonte [3]

Detalhando a topologia da Figura 1, temos os seguintes passos [3]:

a) O carro entra no campo de visão das câmeras, que filmam constantemente as vias, e os quadros são transmitidos inicialmente a um CPE[1], que pode ser um computador ou um minicomputador, o qual possui um código para analisar os frames e extrair informações;

b) Após a obtenção das informações, o CPE será responsável por encaminhá-las pela rede a uma ERB que as transmitirá, por meio de um *backhaul*[2] aos servidores no centro de operações e controle;

c) Por fim, com a informações recebidas dos servidores do centro de operações

[1] Trata-se de um termo genérico que está relacionado à tecnologia que significa "equipamento dentro das instalações do cliente.
[2] É a porção de uma rede hierárquica de telecomunicações que a ligação entre o núcleo da rede, ou *backbone*, e as sub-redes periféricas.

e controle, os operadores poderão visualizá-las, fazer o tratamento dessas informações e enviá-las ao órgão regulador via internet.

Sendo assim, o URCA terá como foco as principais avenidas e ruas das cidades, onde há maior incidência de congestionamentos, e terá um horário de funcionamento predefinido, preferencialmente nos horários de "pico" onde o acúmulo de carros nas vias é mais crítico, visando obter, de forma eficiente, informações que serão usadas pelos órgãos reguladores de trânsito como parâmetros para concessão de benefícios ou tomadas de decisão [1].

Embora o foco, de fato, seja que a aplicação do URCA possa ser implantada nas ruas e avenidas das cidades, neste primeiro momento, os esforços estão concentrados na implantação da solução na área do campus Paulo VI da UEMA, como um protótipo, de forma que seja possível a realização de testes para analisar o comportamento desta solução e aplicar devidas correções em seu funcionamento antes de ser enviadas às ruas.

Para isso, foram definidas algumas etapas que precisavam ser concluídas e integradas, para dar-se início ao processo de implantação da solução, dentre as quais temos:

1.1. Reconhecimento e tratamento de imagens;
1.2. Comunicação de dados;
1.3. Desenvolvimento do aplicativo URCARONA;
1.4. Criação de um banco de dados para guardar as informações;
1.5. Automatização do envio das informações ao banco de dados;
1.6. Implementação da aplicação Urca *Analytics*;
1.7. Implantação da solução no Campus Paulo VI.

Dentre as etapas citadas, as etapas de reconhecimento e tratamento de imagens, comunicação de dados e desenvolvimento do aplicativo URCARONA, já foram abordadas, e estão descritas, de forma detalhada, nos trabalhos Solução Tecnológica de Reconhecimento de Imagens para o Projeto URCA de Uso Racional de Carros nas Cidades Inteligentes [1], Solução Tecnológica de Captura e Armazenamento de Dados para o Projeto URCA [2] e na dissertação de mestrado URCA – Uso Racional de Carros nas Cidades Inteligentes [3], respectivamente.

Portanto, o escopo deste trabalho concentra-se nas conclusões das etapas de:

- **Criação de um banco de dados para guardar as informações:** para que as informações coletadas possam ser armazenadas e posteriormente acessadas;

- **Automatização do envio das informações ao banco de dados:** para agilizar o envio das informações coletadas pelas câmeras durante o monitoramento.
- **Implementação da aplicação Urca *Analytics***: que se trata de uma aplicação que visa fazer o tratamento das informações e prover algumas funcionalidades previamente definidas;
- **Implantação da Solução:** para análise do comportamento da solução em campo, bem como a realização de testes e correções.

Dessa forma, este documento visa demonstrar o andamento de cada uma delas, pontuando os resultados alcançados e relatando os processos de conclusão.

## 1.1. Objetivos

### 1.1.1. Objetivo Geral

Implantar a solução URCA em alguma das vias internas do Campus Paulo VI da UEMA, para que esta possa atuar fazendo o monitoramento de carros, coletando e enviando informações referentes ao número de ocupantes e a placas de identificação ao banco de dados para serem acessadas pela aplicação URCA *Analytics*, e analisar seu comportamento.

### 1.1.2. Objetivos Específicos

- Criar um banco de dados para armazenar as informações coletadas pelas câmeras;
- Criar uma tabela específica no banco de dados para receber placas especiais, para serem comparadas com as placas capturadas e utilizadas como um parâmetro de isenção;
- Desenvolver um algoritmo de geração de informações aleatórias (placas e quantidade de passageiros) para simular o algoritmo de capturas e preencher tabelas do banco de dados para testes da aplicação URCA Analytics;
- Automatizar o envio das informações coletadas pelo algoritmo de capturas ao banco de dados;
- Desenvolver a aplicação URCA *Analytics* para manipulação e visualização dos dados;
- Integrar as etapas da solução;
- Realizar testes para verificação de funcionamento solução e aplicação de correções.

## 1.2. Metodologia

A metodologia deste trabalho, segue de acordo com os objetivos específicos, contando com o sucesso de cada um deles para a conclusão do objetivo geral. Inicialmente, foi criado um banco de dados com duas tabelas, onde a primeira deverá receber automaticamente as informações coletadas pelo algoritmo de capturas conectado às câmeras e uma segunda tabela, que será preenchida manualmente com placas especiais que serão utilizadas para verificar se alguma delas se encontra entre as placas contidas na primeira tabela.

Desse modo, para que ocorra o preenchimento da primeira tabela de forma automática, na etapa seguinte foi feita a automatização do envio de informações pelo algoritmo de capturas e a implementação de um algoritmo geração de informações aleatórias de capturas para complementar o preenchimento da tabela, haja vista que, estima-se uma quantidade de pelo menos cinquenta carros para testes no URCA Analytics e a quantidade de carros utilizados para os testes, foi de apenas dois.

Após esta etapa, foi desenvolvida a aplicação URCA Analytics, a qual possui algumas funcionalidades como: permitir a visualização dos dados, consultar informações específicas, atualizações e exclusões de informações do banco de dados (tabela “placas_especiais”), bem como a geração de listas salvas em arquivos .CSV e plotagem de gráficos.

Dessa forma, a próxima etapa feita foi integrar as aplicações ao banco de dados de modo que, primeiramente, foi conectada a aplicação do CPE, que contém as câmeras e o algoritmo de capturas e em seguida a aplicação URCA Analytics. Por fim, a aplicação foi implantada em uma rua interna do Campus Paulo VI da UEMA, onde foram feitos testes de funcionamento, correções de problemas e verificação dos resultados.

## 1.3. Justificativa

Como as etapas de tratamento de imagens, comunicação de dados e do aplicativo do projeto URCA já tiveram avanços, este trabalho propõe dar andamento as etapas faltantes, que são essenciais para implantação da solução.

Dessa forma, com o sucesso da criação do banco de dados, da plataforma URCA *Analytics* e da integração de todo o conjunto, incluindo as etapas iniciais, pode se dar início ao processo de implantação da solução no campus Paulo VI e, com isso, começar a realização de testes para avaliar a confiabilidade e eficiência da solução e, assim, dar mais um passo em direção da implantação à solução nas ruas e avenidas.

## 1.4. Estrutura do documento

Este relatório está estruturado da forma como segue. No Capítulo 2 é apresentada a fundamentação teórica, onde apresentam-se os conceitos e informações que serviram como base para este trabalho. O Capítulo 3 apresenta o desenvolvimento do trabalho e é mostrado o que foi realizado durante a evolução do projeto são expostos. No Capítulo 4, a implantação da solução e os resultados obtidos durante o processo. O Capítulo 5 traz as conclusões e aprendizados obtidos no desenvolvimento do projeto. Por fim, são apresentados no APÊNDICE A, no APÊNDICE B, os códigos das aplicações desenvolvidas e no APÊNDICE C, o registro do aplicativo URCARONA.

# 2. FUNDAMENTAÇÃO TEÓRICA

Nesta seção serão mostrados os conceitos utilizados, nos quais foram embasados os processos que serão apresentados no Capítulo 3, além de facilitar a compreensão de alguns termos específicos que surgirão ao longo deste documento.

## *2.1.* Geração de dados aleatórios em *Python*

Gerar dados aleatórios pode ser uma maneira eficiente para realização de testes com o intuito de prever o comportamento de uma determinada aplicação, principalmente em situações em que os dados de entrada esperados sejam gerados por uma fonte que ainda está em fase de desenvolvimento [5].

Pode-se citar, por exemplo, geração de números aleatórios feita para estudar um modelo ou comportamento de um programa para diferentes faixas de valores [6]. Desse modo, nesta seção será mostrada a aplicação de alguns módulos de geração de dados aleatórios disponíveis na linguagem *Python* [7].

### *2.1.1.* Módulo *Random*

Este módulo atua na implementação de números pseudoaleatórios, como inteiros, *float* (reais), *strings*, para várias distribuições, tais como: uniforme, normal (gaussiana), lognormal, exponencial negativa, gama e beta, dentre outras [7]. Devido a isso, foi de fundamental importância para geração das informações aleatórias de capturas, principalmente, porque conta com algumas funções, das quais serão destacadas as funções para números inteiros e funções para sequências, que foram amplamente utilizadas no desenvolvimento do código.

Dentre as funções para números inteiros, têm-se [8]:

- ***randrange* (parar)**: retorna um elemento selecionado aleatoriamente, compreendido entre 0 e valor estipulado no parâmetro “parar”;
- ***randrange* (iniciar, parar, passo)**: retorna um elemento selecionado aleatoriamente, que pertença a faixa de valores estipulados nos parâmetros “iniciar”, “parar” e “passo”.
- ***randint (a,b)***: função simplificada da função *randrange*, equivalente a *randrange(a, b+1)*, usada quando se define, tanto os limites inferiores, quanto os superiores de números que se quer gerar, a qual retornará um

elemento $N$ selecionado aleatoriamente, sendo que $a <= N <= b$.

A Figura 2 mostra exemplos de aplicações das funções para números inteiros.

Figura 2. Exemplos de uso das funções randrange e randint na linguagem Python

```
01  import random
02
03  random.randrange(100)
04  # returns 65
05
06  random.randrange(100)
07  # returns 98
08
09  random.randrange(0, 100, 3)
10  # returns 33
11
12  random.randrange(0, 100, 3)
13  # returns 75
14
15  random.randint(1,6)
16  # returns 4
17
18  random.randint(1,6)
19  # returns 6
```

Fonte: EnvatoTuts+[3]

Como observado na Figura 2, nas execuções das funções *randrange* () nas linhas 3 e 6, apenas o parâmetro "parar" é informado e, dessa forma, os resultados nas linhas 4 e 7, pertencem ao intervalo delimitado que é de 0 a 100. Da mesma forma, seguem as execuções das outras linhas, retornando os resultados de acordo com os parâmetros que foram passados.

Quanto às funções para sequências têm-se [8]:

- ***choice(seq)***: tem como resultado um elemento aleatório de uma sequência não-vazia representada pelo parâmetro "*seq*". Caso o parâmetro "*seq*" esteja vazio, gera a exceção *IndexError*[4].
- ***Shuffle(x,[random)]***: embaralha a sequência no lugar do elemento "x". O

[3] ENVATOTUTS+. **Mathematical Modules in Python: Random**. Disponível em: <https://code.tutsplus.com/tutorials/mathematical- modules-in-*Python*-random--cms-27738?ec_unit=translation-info-language>. Acessado em 08 de julho de 2020.
[4] Exceção gerada quando um subscript de sequência está fora do intervalo.

parâmetro “*random*” é opcional e trata-se de uma função de 0 argumento que retorna um *float* aleatório entre [0,0 e 1,0].

A Figura 3 traz exemplos de execução dessas funções em *Python*.

Figura 3. Exemplos de uso das funções *choice* e *shuffle* na linguagem Python

```
>>> choice(['win', 'lose', 'draw'])      # Single random element from a sequence
'draw'

>>> deck = 'ace two three four'.split()
>>> shuffle(deck)                        # Shuffle a list
>>> deck
['four', 'two', 'ace', 'three']
```

Fonte: Adaptado de [8]

Nos resultados exibidos na Figura 3, percebe-se que, na função *choice* (retângulo vermelho), é retornado um valor aleatório dentre os elementos da lista. Já na função *shuffle* (retângulo azul), é retornada a própria lista que foi passada como parâmetro, porém com os elementos embaralhados.

### *2.1.2.* Módulo *String*

Este também é um dos módulos da linguagem *Python*, que foi utilizado na geração dos caracteres das placas de identificação de veículos, então considerou-se importante o conhecimento de algumas de suas funções para ajudar no entendimento de alguns quesitos referente à implementação do algoritmo de geração de informações aleatórias.

De acordo com a documentação disponível em [9], *strings* são um tipo de dado do *Python*, que implementa todas as operações de sequência de caracteres comuns e possuem uma série de funções, métodos e constantes, dentre as quais foram destacadas as seguintes:

- *str*.***capitalize()***: método que retorna uma cópia da *string* com o seu primeiro caractere em *titlecase*, onde caracteres como dígrafos apenas terão sua primeira letra alterada para maiúscula, ao invés de todos os caracteres, e o restante em minúsculo;
- *str*.***format***(**args*, ***kwargs*): trata-se de uma função de formatação personalizada que executa uma operação de formatação de *strings*, onde a *string* na qual este método é chamado, pode conter texto literal ou campos

para substituição delimitados por chaves “{}”. Em seguida, retorna uma cópia da *string* onde cada campo para substituição é substituído com o valor da *string* do argumento correspondente;

- *string.ascii_lowercase*: constante que retorna letras minúsculas e que independe da localidade e não pode ser alterada. Ex.: “abcdefghijklmnopqrstuvwxyz”;
- *string.ascii_uppercase*: constante que retorna letras maiúsculas, e também independe da localidade, além de não poder ser alterada; Ex: “ABCDEFGHIJKLMNOPQRSTUVWXYZ”;
- *string.digits*: constante que retorna como valor os números '0123456789'.

Alguns exemplos de aplicação das funções e constantes do módulo *String* apresentadas, são demonstradas nas Figuras 4 e 5.

Figura 4. Exemplos de aplicações de constantes do módulo *String*

```
import string

letras_minusculas = string.ascii_lowercase

letras_maiusculas = string.ascii_uppercase

numeros = string.digits

print(" Letras Minúsculas: %s \n" % letras_minusculas,
      "Letras Maiúsculas: %s \n" % letras_maiusculas,
      "Números: %s" % numeros)
```

```
Output:
Letras Minúsculas: abcdefghijklmnopqrstuvwxyz
Letras Maiúsculas: ABCDEFGHIJKLMNOPQRSTUVWXYZ
Números: 0123456789

Process finished with exit code 0
```

Fonte: Adaptado de [8]

A Figura 4 mostra a execução de um algoritmo, onde as variáveis

“letras_minusculas”, “letras_maiusculas” e “numeros” fazem chamadas às constantes *string.ascii_lowercase*, *string*.ascii_*uppercase* e *string.digits*, respectivamente e têm seus resultados exibidos na saída. Vale ressaltar que, por meio dessas constantes e da utilização da função *choice()* do módulo *Random* é que foram geradas as placas de identificação de forma aleatória.

Já na Figura 5, foram demonstrados exemplos de utilização do método *capitalize* e da função *format*, sendo esta última, utilizada na exibição de dados no URCA Analytics.

Figura 5. Exemplos de aplicações do método *capitalize* e da função *format*

```
1   import string
2
3   palavra_minuscula = "capitalize"
4   primeira_letra_maiuscula = str.capitalize(palavra_minuscula)
5
6   print("Exemplo do Método Capitalize: %s " % primeira_letra_maiuscula)
7
8   a = "formatação"
9   b = "personalizada"
10  c = "String"
11
12  print("Exemplo de {} {}, utlizando o módulo {}".format(a,b,c))
```

```
Output:
Exemplo do Método Capitalize: Capitalize
Exemplo de formatação personalizada, utlizando o módulo String

Process finished with exit code 0
```

Fonte: Do Autor

Observa-se na Figura 5, que na utilização do método *capitalize*, a variável “palavra_minuscula” recebe como entrada uma *string* com os caracteres “*capitalize*”, todos minúsculos, e na saída tem-se apenas a primeira letra em maiúsculo, “*Capitalize*”. No caso da função *format*, ela permitiu que os valores das variáveis “a”, “b” e “c”, ocupassem os espaços delimitados por chaves “{}”.

## 2.2. Bancos de Dados

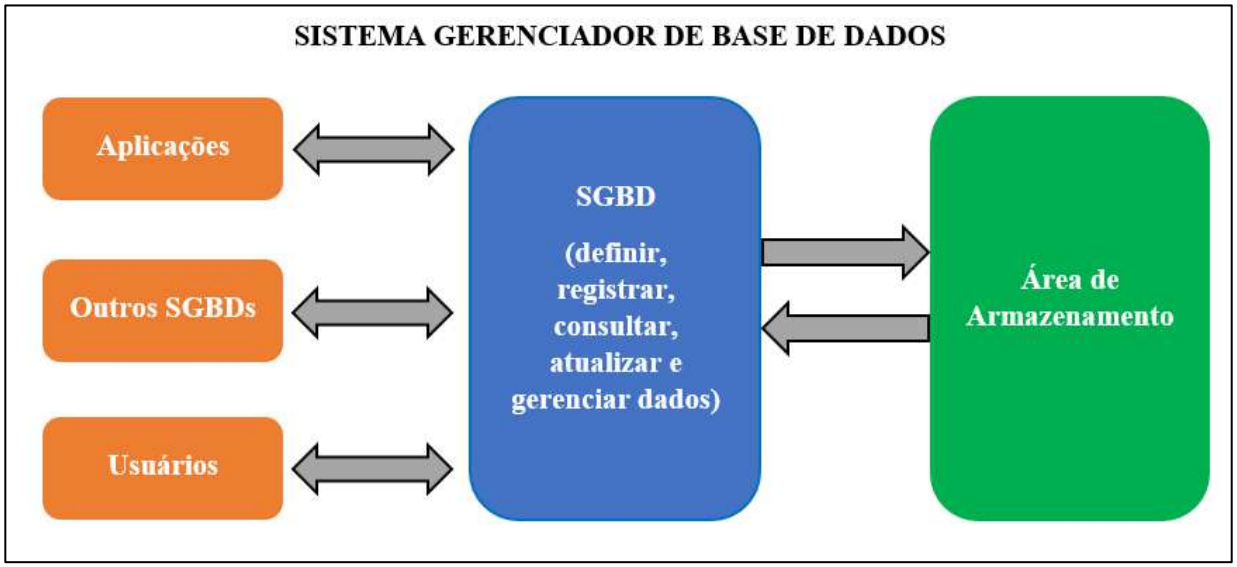
Nesta seção serão tratados conceitos referentes aos bancos de dados, que podem ser ditos como um conjunto de dados integrados que tem a finalidade de servir a um grupo de sistemas [10]. Sendo assim, nos subtópicos seguintes serão apresentadas definições de SGBD, modelo relacional, linguagem SQL e a ferramenta *MySQL Workbench* de modo a nortear o entendimento das ferramentas e práticas utilizadas na criação do banco de dados

do projeto URCA.

### 2.2.1. SGBD

Um Sistema Gerenciador de Base de Dados ou SGBD, que vem do inglês *Data Base Management System*, é um sistema computadorizado projetado para auxiliar na manutenção e utilização de grandes coleções de dados, que oferece uma série de vantagens, como: reforçar a segurança e a integridade dos dados, prover um método de acesso de dados eficiente, facilitar a gestão dos dados, dentre outras, composto por dados, *hardware*, *software* e usuários [11,12]. A Figura 6 mostra a representação de um SGBD.

Figura 6. Representação de um SGBD




Fonte: Adaptada de BMC BLOGS[5]

Dessa forma, os dados podem ser ditos como uma representação simbólica, quantificada ou quantificável ou, em outras palavras, são fatos conhecidos que podem ser registrados e possuem significado implícito [13, 14].

Quanto ao hardware, se refere aos elementos físicos que compõem o SGBD, como, por exemplo, mídias de armazenamento e canais de entrada/saída [15]. Já o software no âmbito do SGBD, pode ser considerado um conjunto de programas que se encontra entre o banco de dados armazenado e o próprio SGBD [12].

Tratando-se dos usuários, esses podem dividir-se em três categorias [12]: programador de aplicação que é quem cria aplicações que acessarão o sistema de banco de

[5] BMC BLOGS. DBMS: An Intro to Database Management Systems. Disponível em: <https://www.bmc.com/blogs/dbms-database-management-systems/>. Acessado em 20 de março de 2021

dados; o usuário final, que consulta e atualiza o banco de dados utilizando as aplicações desenvolvidas pelos programadores; e o administrador de banco de dados – DBA – que é responsável por gerir o SGDB.

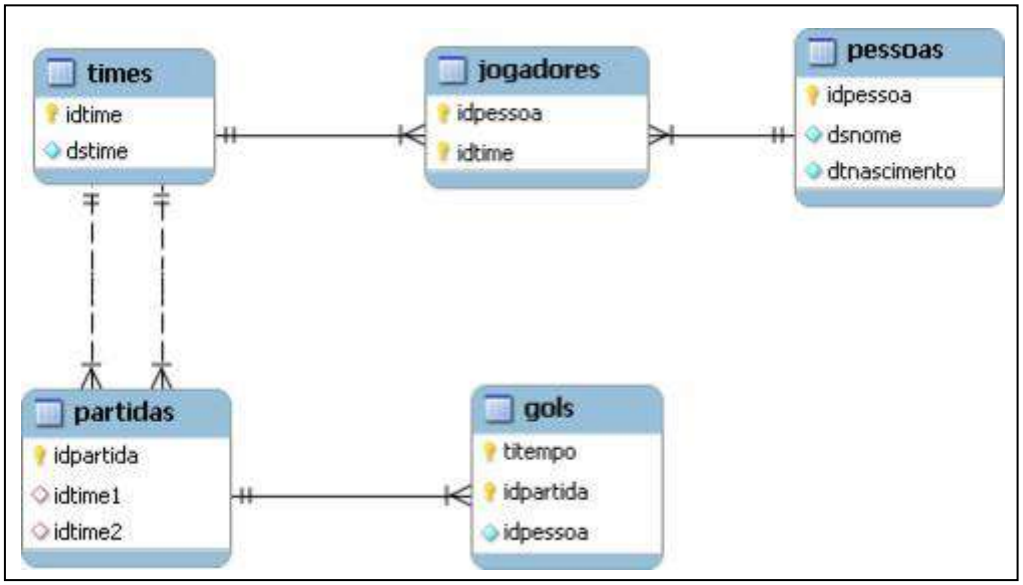
### 2.2.2. Modelo Relacional

É um modelo de dados representativo proposto por Edgar Frank Codd, em 1970 [12], e trata-se de um modelo de banco de dados fundamentado na ideia de que os dados devem ser guardados em tabelas, e sua definição é baseada na teoria dos conjuntos, ramo da matemática que estuda conjuntos, que são uma coleção de elementos [16].

Segundo Helland [17], os bancos de dados relacionais possuem tabelas com linhas e colunas, onde cada coluna em uma linha fornece uma célula que é de um tipo conhecido. Além disso, Helland [17] afirma que a Linguagem de Definição de Dados (DDL) especifica as tabelas, linhas e colunas e pode ser alterada dinamicamente a qualquer momento, transformando a forma dos dados.

Helland [17] ainda ressalta que, o princípio fundamental no modelo relacional é que todo inter-relacionamento é alcançado por meio de comparações de valores, sendo que esses valores identifiquem objetos no mundo real ou indiquem propriedades desses objetos. A Figura 7 traz um exemplo de banco de dados relacional.

Figura 7. Banco de dados relacional




Fonte: Site Compartilhando[6]

[6] COMPARTILHANDO. **Questões – Banco de Dados – Modelagem Relacional**. Disponível em <http://jkolb.com.br/questoes-banco-de- dados-modelagem-relacional/>. Acessado em 4 de julho de 2020.

De acordo com Oliveira [16], atualmente, esse modelo ainda é amplamente utilizado pelo fato de prover acesso facilitado aos dados, possibilitando aos usuários utilizar uma grande variedade de abordagens no tratamento das informações, além da possibilidade de uso dos sistemas gerenciadores de bancos de dados, que executam comandos na linguagem SQL e têm a responsabilidade de gerenciar o acesso, a manipular e organizar os dados, principalmente no que diz respeito à segurança. Devido a isso, este foi o modelo utilizado na criação do banco de dados do URCA, no qual foram utilizados o MySQL e a ferramenta MySQL Workbench que serão demonstrados no tópico a seguir.

## 2.3. Linguagem SQL e MySQL Workbench

O termo *Structured Query Language* (SQL), refere-se a uma linguagem padrão para trabalhar com bancos de dados relacionais[7], que devido a sua popularização, foi padronizada na década de 80, pelas organizações ISO e ANSI [18] e é amplamente utilizada pelos SGBDs em várias operações, como gerenciar usuários, criar objetos, inserir e deletar registro, entre muitas outras[8].

O SQL não é propriamente uma linguagem de programação, entretanto, atua de forma similar na criação de tabelas, consulta e manipulação de informações em bancos de dados relacionais, por meio de comandos e *querys*, que são requisições de informações ao banco de dados que tem como resposta, o envio de uma tabela ou um conjunto de tabelas ou um determinado dado específico, de acordo com o que é solicitado[9].

Para isso, existem plataformas de interação com os bancos de dados, como o MySQL Workbench, que de acordo com sua documentação [19], é uma ferramenta gráfica para trabalhar com servidores e bancos de dados MySQL, a qual suas funcionalidades oferecem cobertura a cinco tópicos principais, que são:

- **Desenvolvimento SQL:** permite criar e gerenciar conexões com servidores de banco de dados e permite que configurações de parâmetros de conexão, além de oferecer a capacidade de executar *querys* SQL nas conexões do banco de dados usando o Editor SQL embutido;
- **Modelagem de dados (design):** permite a criação de modelos do esquema do banco de dados graficamente, a engenharia reversa e direta entre um

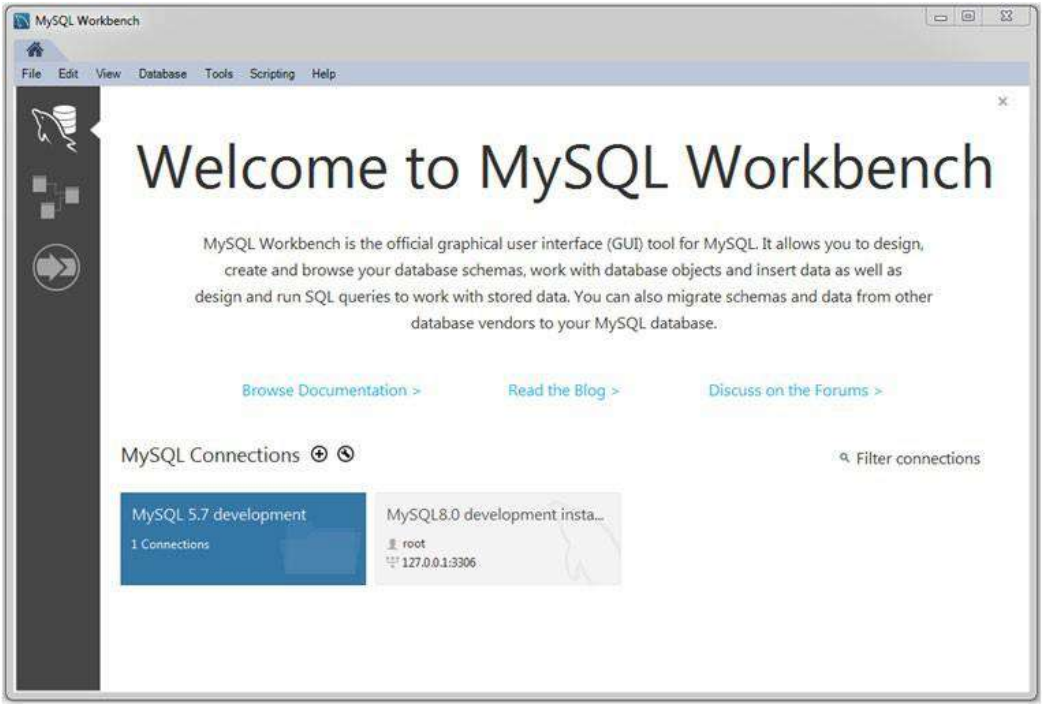
[7] ALURA. **O que é SQL?**. Disponível em: < https://www.alura.com.br/artigos/o-que-e-sql >. Acessado em 01 de julho de 2020.
[8] CLEMENTE, M. **O que é e como usar uma Query**. STAGE - Rock Content. 2019. Disponível em: < https://rockcontent.com/blog/query/ >. Acessado em 01 de julho de 2020.
[9]LINK OFICIAL. **Você sabe a Diferença entre SQL e MySQL?**. Disponível em: < https://www.linkoficial.com.br/diferenca-entre-sql-e-mysql/ >. Acessado em 01 de julho de 2020.

esquema e um banco de dados ativo e a edição de todos os aspectos do banco de dados usando o seu Editor de Tabelas;

- **Administração do servidor:** Permite administrar instâncias do servidor MySQL, administrando usuários, executando backup e recuperação, inspecionando dados de auditoria, visualizando a integridade do banco de dados e monitorando o desempenho do servidor MySQL;
- **Migração de dados:** permite a migração do Microsoft SQL Server, Microsoft Access, Sybase ASE, SQLite, SQL Anywhere, PostreSQL e outras tabelas, objetos e dados RDBMS para o MySQL. A migração também suporta a migração de versões anteriores do MySQL para as versões mais recentes;
- **Suporte ao MySQL Enterprise:** fornece suporte para produtos corporativos, como MySQL Enterprise Backup, MySQL Firewall e MySQL Audit.

A Figura 8 mostra a tela inicial do MySQL Workbench.

Figura 8. Tela de abertura do MySQL Workbench



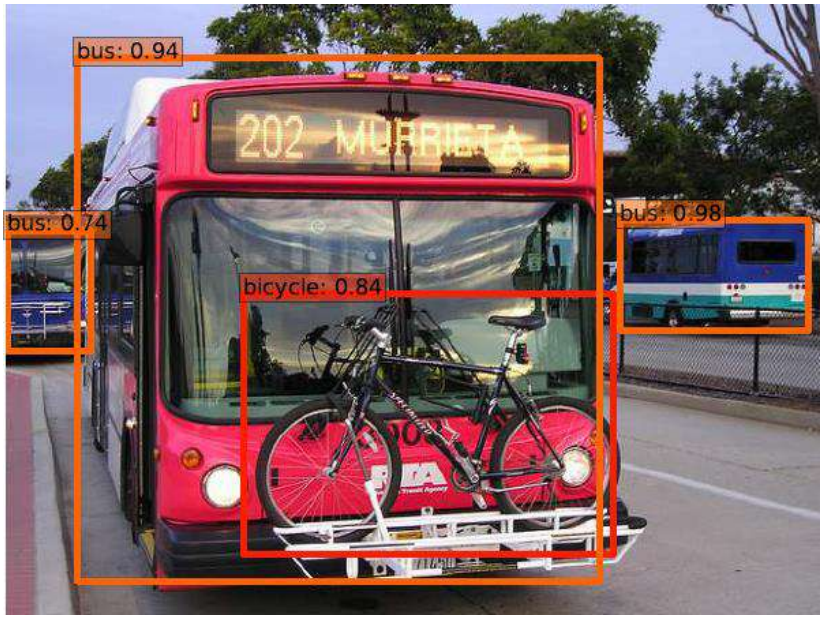

Fonte: [19]

Nesta tela, mostrada na Figura 8, é possível criar conexões com o banco de dados, fazer modelagem e também processos de migração [19]. Agora que já foram vistos os conceitos relativos ao banco de dados, na Seção 2.4 será falado sobre conceitos e técnicas usados para o desenvolvimento do algoritmo de automatização das capturas.

## 2.4. Rastreamento de objetos (*Object Tracking*)

O rastreamento de objetos, ou *object tracking* é um processo que consiste em estimar a movimentação de objetos através de uma sequência de imagens fornecidas por uma câmera e acompanhar o deslocamento desses objetos em tempo real por meio da análise de semelhanças, da combinação de amostras ou ainda pelo fluxo de cada pixel, sendo o primeiro passo para este processo, a detecção dos mesmos nas imagens [20]. Para isso, utiliza-se técnicas como, a *Single Shot MultiBox Detector* (SSD), além de outras, para detecção de objetos, mostrada na Figura 9.

Figura 9. Detecção de objetos (ônibus e bicicleta) utilizando SSD




Fonte: Adaptada de [21]

A técnica SSD trata-se de um método para detectar objetos utilizando uma única rede neural, sendo que tal característica a torna mais rápida em relação a modelos anteriores, como o YOLO [22] e Faster R-CNN [23], além de ser capaz de fazer múltiplas detecções mantendo um alto índice de acurácia como mostrado na Figura 9 [21].

Dessa forma, de acordo com Liu *et al.* [21], o SSD funciona discretizando o espaço de saída das caixas delimitadoras em um conjunto de caixas padrão de diferentes proporções e escalas, baseado na localização do mapa de características. Sendo assim, no momento da previsão, a rede gera pontuações para a presença de cada categoria de objeto em cada caixa padrão e produz ajustes na caixa para melhor corresponder à forma do objeto. Além disso, a rede combina previsões de vários mapas de características com diferentes resoluções para lidar naturalmente com objetos de vários tamanhos.

Sendo assim, devido às vantagens que esta técnica oferece, a mesma foi utilizada no desenvolvimento do algoritmo responsável pelo processo de automatização das capturas do projeto URCA, tornado viável a detecção dos carros.

### *2.5. Biblioteca OpenCV*

A visão computacional é uma ciência, que faz parte do âmbito da programação, e tem como foco fazer um computador processar e entender imagens e vídeos ou, de modo mais claro, fazer o computador enxergar [24]. Entretanto, o avanço da pesquisa estava condicionado à criação de uma biblioteca que disponibilizasse funções de programação com o código otimizado, portátil e desejavelmente gratuito [25].

Então, para atender a essas demandas foi desenvolvida a biblioteca *OpenCV* (*Open Source Computer Vision Library*) que, atualmente, conta com mais de 2500 algoritmos otimizados que incluem um conjunto abrangente de algoritmos de visão computacional e aprendizado de máquina clássicos e de última geração [25].

Dentre o leque de utilidades destes algoritmos, encontram-se sua capacidade de detectar e reconhecer rostos, identificar objetos, classificar ações humanas em vídeos, rastrear movimentos de câmera, rastrear objetos em movimento, dentre outros, sendo uma biblioteca amplamente utilizada em empresas, grupos de pesquisa e por órgãos governamentais[10] com suporte para plataformas como: Android, IOS, ARM, dentre outras[11].

Para exemplificar algumas funções e classes, dentre outras, da biblioteca *OpenCV*, retiradas de sua documentação[12], são mostradas a seguir:

[10] OPENCV. **About**. Disponível em: <https://opencv.org/about/>. Acessado em 17 de março de 2021.
[11] OPENCV. **Plataforms**. Disponível em: <https://opencv.org/platforms/>. Acessado em 17 de março de 2021.
[12] OPENCV. **OpenCV Modules.** Disponível em: < https://docs.opencv.org/4.5.1/>. Acessado em 17 de março de 2021.

- **cv2.dnn.readNetFromCaffe:** é uma classe que tem como função ler um modelo de rede armazenado no formato do framework Caffe. É através dessa classe que as características da rede SSD para detecção de objetos, dentre outras, são carregadas;

- **cv2.rectangle():** função que desenha um retângulo simples, espesso ou preenchido de acordo com as coordenadas obtidas pelo detector de objetos, como, por exemplo, as caixas delimitadoras (*bounding box*);

- **cv2.imshow():** função usada para exibir uma imagem na janela especificada.

Vale ressaltar que essas três funções foram essenciais para a automatização das capturas do URCA, bem como o algoritmo *Centroid Tracker*, mostrado na Seção 2.6.

## *2.6.* Algoritmo *Centroid Tracker*

Outra ferramenta essencial para o desenvolvimento projeto foi o algoritmo *Centroid Tracker*, onde suas características, como por exemplo, a atribuição de um identificador único para cada objeto detectado, também foram de fundamental importância para a automatização das capturas do URCA. Este algoritmo, desenvolvido em Python, é usado na detecção de objetos e tem esse nome pelo fato de que seu funcionamento depende da distância euclidiana entre centroides de objetos existentes e de centroides de novos objetos entre quadros subsequentes em um vídeo [26].

Desse modo, o funcionamento do *Centroid Tracker* segue uma série de passos [26]:

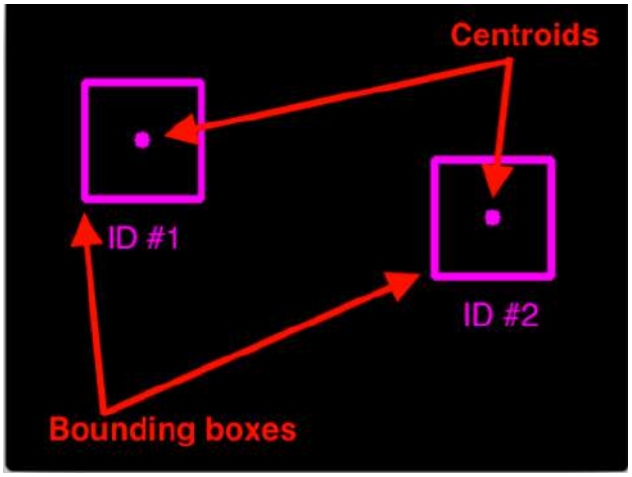
1. Obter as coordenadas da caixa delimitadora e calcular os centroides;
2. Calcular a distância euclidiana entre as novas caixas delimitadoras e os objetos existentes;
3. Atualizar as coordenadas x e y de objetos existentes;
4. Registrar novos objetos;
5. Remover o registro de objetos antigos ou perdidos que saíram do quadro.

No passo 1, o algoritmo assume que está sendo passado um conjunto de coordenadas (x, y), vindas das caixas delimitadoras (*bounding box*) produzidas por um detector de

objetos, neste caso o SSD, para cada objeto detectado nos quadros dos vídeos. Após a obtenção das coordenadas, deve-se calcular o centroide (coordenadas x e y do centro) da caixa delimitadora. A Figura 10 representa a aceitação de um conjunto de coordenadas de caixa delimitadora e o cálculo do centroide no passo 1 [26].

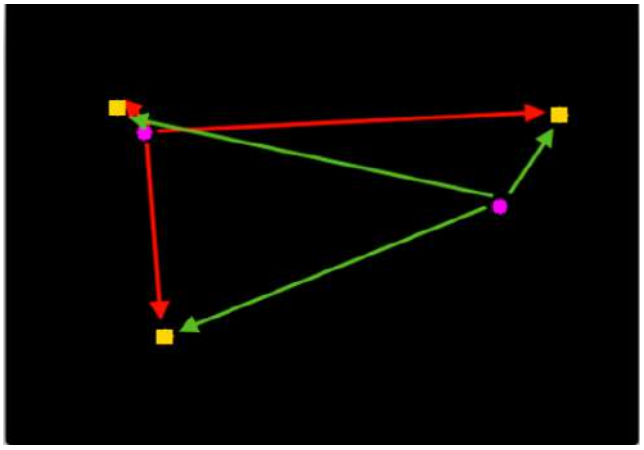
Já o passo 2 segue, representado na Figura 11, o seguinte parâmetro: para cada quadro subsequente do vídeo, aplica-se o passo 1, porém, ao invés de atribuir um novo ID exclusivo para cada objeto detectado, primeiro precisa-se determinar se há possibilidade de associar os centroides dos novos objetos (círculos) com os dos antigos (quadrados). Para isso, calcula-se a distância euclidiana (destacada com setas verdes) entre cada par de centroides de objetos existentes e os centroides dos objetos que entram, onde no algoritmo é calculada pela função "*scipy.spatial.distance.cdist"* [26].

Figura 10. Representação do passo 1




Fonte: [adaptada de 26]

Figura 11 Representação do passo 2

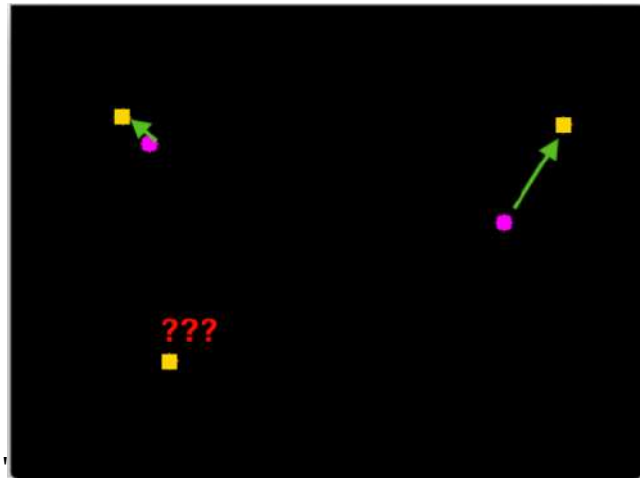


Fonte: [adaptada de 26]

Na Figura 11, também é possível ver que foram detectados três objetos na imagem, onde os dois pares próximos são dois objetos já existentes. Em seguida, calcula-se as distâncias euclidianas entre cada par de centroides originais, destacados em amarelo, e novos centroides, destacados em roxo para serem usadas no passo 3 [26].

Em seguida, no passo 3 a ideia é que o algoritmo suponha que um determinado objeto se moverá potencialmente entre os quadros subsequentes, entretanto a distância entre os centroides para os quadros Q e Q + 1 será menor do que todas as outras distâncias entre objetos. Dessa forma, se forem escolhidos centroides com distâncias mínimas entre os quadros subsequentes para serem associados, o rastreamento de objetos irá funcionar corretamente. Na Figura 12, é possível ver como o algoritmo rastreador escolhe associar centroides que possuem suas respectivas distâncias euclidianas menores [26].

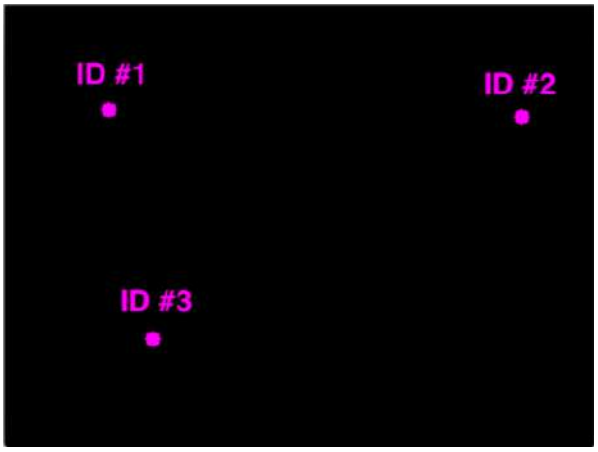
Figura 12. Representação do passo 3




Fonte: [adaptada de 26]

Quanto ao ponto no canto inferior esquerdo da Figura 12, este será registrado como um novo objeto no passo 4, como representado na Figura 13. Neste passo, os novos objetos são adicionados à lista de objetos rastreados e a eles são atribuídos um novo ID exclusivo e, além disso, as coordenadas das caixas delimitadoras para esse objeto também são armazenadas [26].

Figura 13. Representação do passo 4




Fonte: [adaptada de 24]

Por fim, no passo 5 é feito o tratamento para quando objeto se perde ou sai do campo de visão, removendo objetos antigos da lista de objetos registrados quando estes não puderem ser correspondidos a nenhum objeto existente para um total de $N$ quadros subsequentes [26].

Sendo assim, a utilização deste algoritmo foi essencial para o processo de automatização das capturas, pois o centroide encontrado nos quadros tornou possível estimar quais deles serão utilizados para extração de informações e a utilização deste algoritmo em conjunto com o algoritmo de capturas é apresentada no Capítulo 3 deste documento.

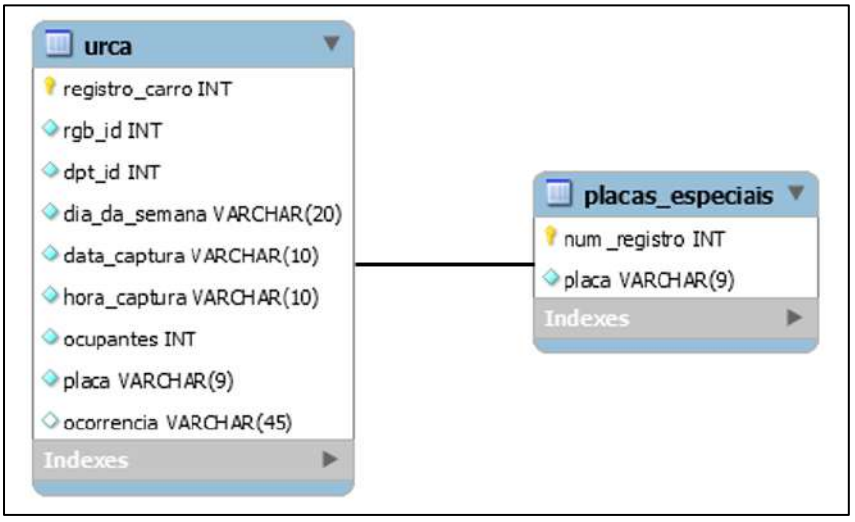
# 3. DESENVOLVIMENTO

Esta seção tem como foco, a apresentação do desenvolvimento dos objetivos propostos, descrevendo de forma detalhada os processos e técnicas utilizadas para conclusão de cada um deles.

## 3.1. Criação do Banco de Dados

Nesta primeira fase, foram feitas a modelagem e a criação do banco de dados, onde foi escolhido o MySQL, visto que este SGBD atende às necessidades atuais do projeto. Sendo assim, utilizando a ferramenta para criação de diagramas *EER Diagram* do *MySQL Workbench* foi feita a modelagem de um banco de dados simples, mostrada na Figura 14, composto inicialmente por duas tabelas, que será utilizado para guardar as informações coletadas pelo algoritmo de capturas, bem como fornecer estas informações à aplicação URCA Analytics.

Figura 14. Modelagem do Banco de Dados do Projeto URCA.




Fonte: Do Autor

Dessa forma, na Figura 14 têm-se a criação da tabela “urca” e tabela “placas_especiais”, sendo o objetivo da primeira, receber diretamente as informações fornecidas pela aplicação de capturas, enquanto a segunda será manipulada diretamente pela aplicação URCA Analytics para o cadastro de placas consideradas “especiais”, como, por exemplo, placas de táxi, ambulâncias, viaturas de polícia, dentre outras, onde estas placas serão consideradas isentas, caso sejam capturadas pela solução.

Detalhando melhor as tabelas criadas na modelagem do banco de dados, pode-se dizer que ambas são compostas por um número definido de colunas, onde as primeiras são chaves primárias, tais como “registro_carro”, na tabela “urca” e “num_registro”, na tabela “placas_especiais”, que servirão para atribuir, automaticamente, um identificador exclusivo do tipo *INT*[13] a cada uma de suas entradas e para a formação de possíveis relacionamentos com outras tabelas.

Além disso, as tabelas contam com outras colunas, que serão povoadas com informações, informações vindas das aplicações, dentre as quais estão:

- Na tabela “urca”:
  - ***rgb_id (INT)*:** receberá os identificadores dos quadros capturados pela câmera de vídeo *RGB*;
  - ***dpt_id (INT)*:** da forma análoga à coluna *rgb_id*, receberá os identificadores dos quadros capturados pela câmera de infravermelho;
  - **dia_da_semana (*VARCHAR*[14])*:*** receberá a informação que conterá o dia da semana em que a captura foi registrada;
  - **data_captura (*VARCHAR*)*:*** receberá a informação referente à data da captura;
  - **hora_captura (*VARCHAR*)*:*** receberá a informação contendo as horas, minutos e segundos em que a captura foi registrada;
  - **ocupantes *(INT)*:** receberá a informação referente a quantidade de ocupantes no interior de um determinado carro, que será extraída, pelo algoritmo de capturas, do quadro capturado pela câmera de infravermelho;
  - **placa (*VARCHAR*)*:*** receberá a informação contendo os caracteres da placa de identificação de um determinado carro, que serão extraídos, pelo algoritmo de capturas, do quadro capturado pela câmera de vídeo RGB;
  - **ocorrência (*VARCHAR*)**: diferentemente das sete primeiras colunas, que são povoadas pela aplicação de capturas, esta coluna receberá, da

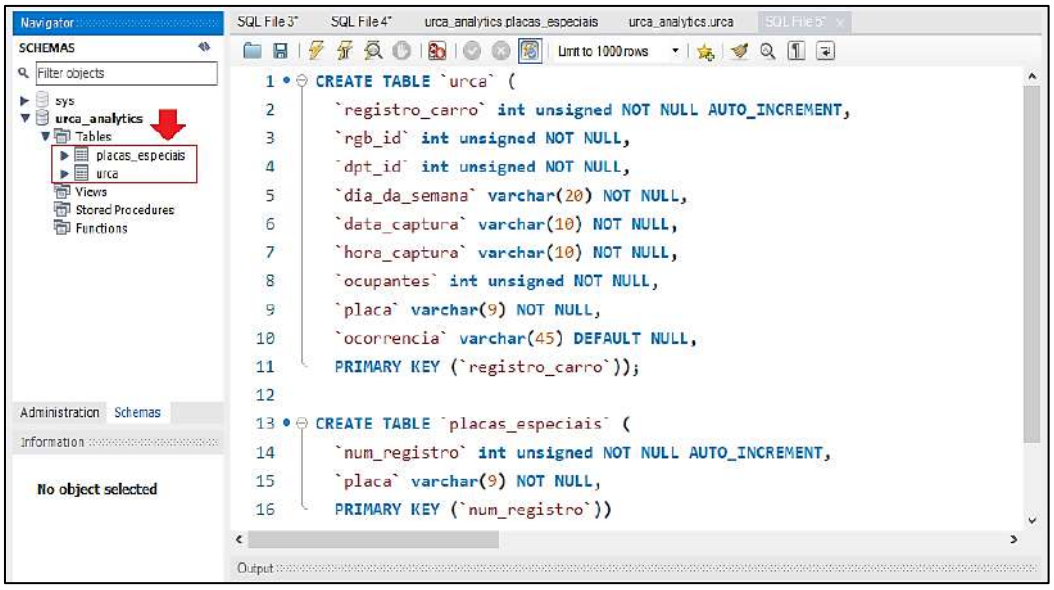
[13] Variável do tipo inteiro.
[14] Variável tipo caracteres de comprimento determinado ou indeterminado.

aplicação URCA Analytics, a ocorrência atribuída a um determinado carro obtida após o tratamento das informações.

- Na tabela “placas_especiais”:
  - **placas (*VARCHAR*):** receberá entradas de placas consideradas “especiais” vindas da aplicação URCA Analytics.

Sendo assim, após a etapa de modelagem, o banco de dados foi implementado no MySQL Workbench, contendo as tabelas e respectivas colunas demonstradas na modelagem, por meio dos comandos DDL, apresentado na Figura 15.

Figura 15. Implementação do Banco de Dados: Criação das Tabelas “urca” e “placas_especiais”




Fonte: Do Autor

Com o banco de dados criado, o próximo passo seria estabelecer a comunicação entre o banco e a aplicação de capturas, desse modo, será possível testar se a aplicação é capaz de povoar as tabelas corretamente com informações que posteriormente serão manipuladas por meio do URCA Analytics. Entretanto, é necessário tornar o processo de extração de informações das capturas automático, e esta etapa é demonstrada na Seção 3.2.

## *3.2.* Automatização do envio das informações ao banco de dados

Para tornar o funcionamento do URCA mais prático e eficiente, bem como preencher as tabelas do banco de dados com informações vindas diretamente da aplicação de capturas,

havia necessidade de se automatizar o processo de obtenção das capturas, uma vez que as entradas (quadros de vídeo) para extração de informação, no início, eram feitas manualmente.

Esse processo se dava por meio da análise de filmagens feitas pelas câmeras RGB e de infravermelho, onde os quadros dos vídeos eram separados individualmente e o operador elegia qual deles seria a melhor entrada para o algoritmo de capturas, baseado em condições definidas como ideais, tais como, menor quantidade de ruídos nas imagens e melhor posicionamento [3]. A Figura 16 traz exemplos de quadros ideais para extração de informações.

Figura 16. Exemplo de quadro ideais de entrada: RGB (à esq.) e infravermelho (à dir.)

Fonte: [3]

Na Figura 16, o quadro da câmera de infravermelho, à direita, nestas condições, permitirá que o algoritmo extraia a informação referente à quantidade de ocupantes no carro, neste caso um único ocupante, marcado pelo retângulo verde, assim como o quadro RGB, à esquerda, permitirá a extração dos caracteres da placa, marcado pelo retângulo azul.

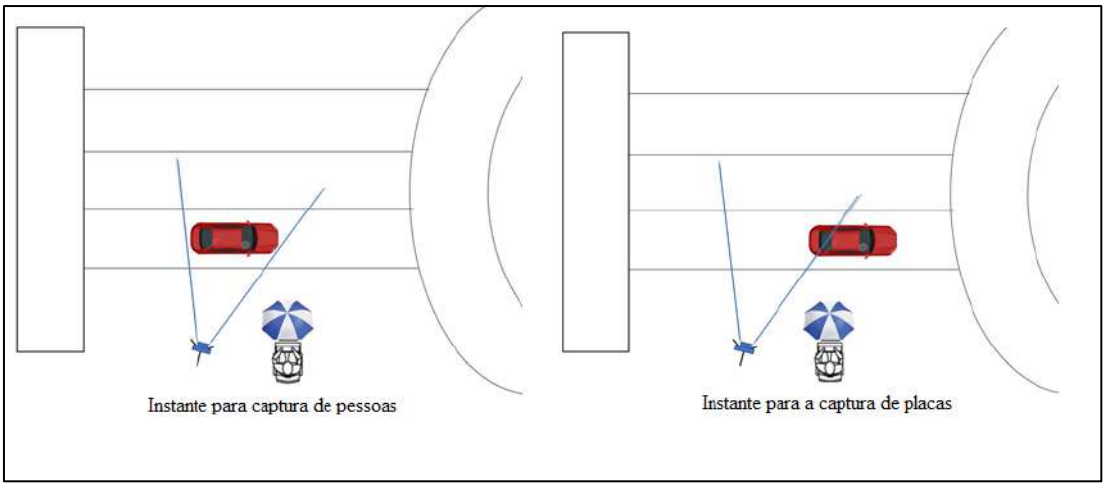
Sendo assim, para a obtenção dessas informações nas condições mostradas na Figura 16, foi utilizado um único dispositivo, o *Intel Realsense D435*[15], que conta com as câmeras RGB e infravermelho no mesmo dispositivo, que foi posicionado a uma distância de aproximadamente 3,5 metros da pista de rolamento, a 1,30 metros em relação ao chão e, de acordo com o que é especificado por Costa [3], em ângulo entre 70º e 90º como mostrado nas Figuras 17 e 18, para possibilitar a obtenção das informações.

[15] INTEL REALSENSE. ***Intel RealSense Depth Camera D435***. Disponível em: <.https://www.intelrealsense.com/depth-camera-d435/>. Acessado em: 22 de março de 2021.

Figura 17. Demonstração do posicionamento da câmera no campus Paulo VI – UEMA. À direita, distância da câmera em relação ao carro e, à esquerda, em relação ao chão.


Fonte: Do Autor

Figura 18. Esquema para as capturas do número de pessoas e de placas




Fonte: Adaptada de [3]

Desse modo, como mostrado na Figura 18, quando um carro se deslocando da esquerda para direita, estivesse no centro do campo de visão da câmera, o algoritmo obteria o número de pessoas e ao fim do campo de visão da câmera, seriam obtidos os caracteres das placas dos carros [3].

Partindo dessas premissas, pensou-se na ideia de um gatilho (*trigger*) para capturar os quadros nos instantes mostrados na Figura 16 e dessa maneira obter, automaticamente, as entradas necessárias para cada tipo de extração de informações. Após um período de estudos, chegou-se à implementação de uma aplicação que combinou o algoritmo de

capturas [3] com técnicas de detecção de objetos e visão computacional, que funciona da seguinte maneira:

1. Utilizando a detecção de objetos e visão computacional, os carros são detectados e marcados com caixas delimitadoras (*bounding box*) à medida que passam no campo de visão das câmeras;
2. Com as coordenadas das caixas delimitadoras, é calculado o ponto central (centroide) de cada caixa marcada nos quadros;
3. São criados pontos para estimar em qual instante os quadros devem ser cortados.

Detalhando o funcionamento da aplicação, na primeira parte foi utilizado o detector de objetos do tipo SSD [21], o qual havia sido previamente treinado para fazer a detecção de certos tipos de classes de objetos, dentre os quais estavam os carros, como mostradas no trecho do código apresentado na Figura 19.

Figura 19. Lista de classes treinadas pelo detector SSD

```
30      # inicializa a lista de rótulos de classe MobileNet SSD foram treinados para detecção
31      classes = ["background", "aeroplane", "bicycle", "bird", "boat",
32                 "bottle", "bus", "car", "cat", "chair", "cow", "diningtable",
33                 "dog", "horse", "motorbike", "person", "pottedplant", "sheep",
34                 "sofa", "train", "tvmonitor"]
35      # carrega o modelo serializado do disco
36      rede = cv2.dnn.readNetFromCaffe(conf_args["prototxt_caminho"],conf_args["model_caminho"])
```

Fonte: Do Autor

Percebe-se na linha 31 da Figura 19, a inicialização de uma variável chamada "classes" contendo de vários tipos de classes de objetos escritas em inglês, devido ao treinamento utilizado ter sido feito nessa linguagem, onde, destacado em laranja, encontra-se a classe "*car*", que significa "carros" em português.

Apesar da rede ter sido treinada para reconhecer objetos além de carros, a aplicação foi codificada para ignorar todas as classes diferentes da classe "*car*" durante o processo de detecção, trazendo assim mais confiabilidade aos resultados retornados por ela.

Além disso, é possível ser visto na linha 36 da Figura 19, a variável "rede", onde são carregados por meio da classe "*cv2.dnn.readNetFromCaffe*" da biblioteca OpenCV, os caminhos para o arquivo "*.prototxt*", que contém um texto descrevendo a arquitetura da rede, e para o arquivo "*.caffeModel*" que contém os arquivos com o aprendizado da rede SSD.

Então, a rede SSD foi utilizada para detectar objetos da classe nos quadros da câmera RGB, uma vez que os quadros da câmera de infravermelho não ofereceram imagens que permitissem a captura de carros nos testes realizados.

Devido a isso, as coordenadas obtidas pela detecção dos carros nos quadros RGB foram "espelhadas" nos quadros de infravermelho onde foram desenhadas caixas delimitadoras com a biblioteca OpenCV por meio da função "*cv2.rectangle()*" e mostradas em uma tela com a função "*cv2.imshow()*", como mostrado na Figura 20.

Figura 20. Caixas delimitadoras desenhadas nos quadros infravermelho (esquerda) e RGB (direita)

Fonte: Do Autor

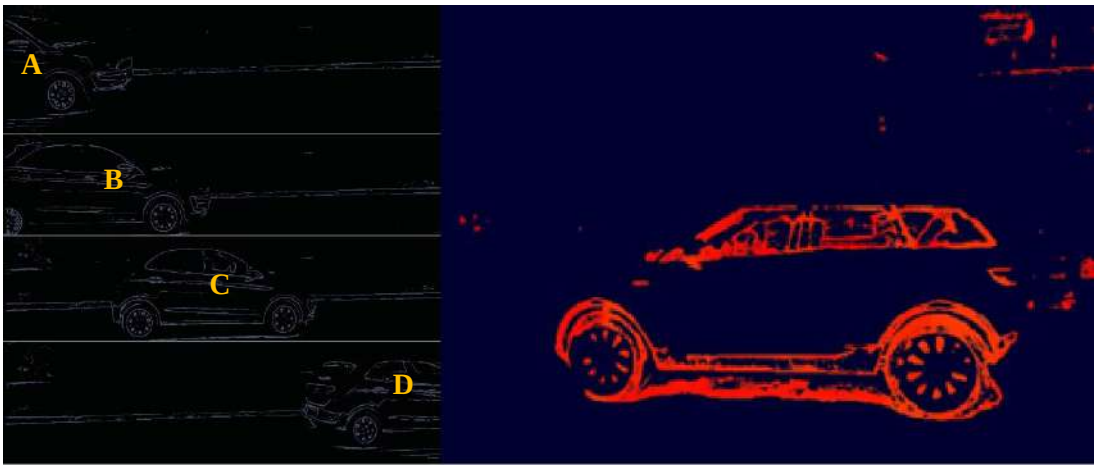
Em sequência, foi feito o cálculo dos centroides de cada uma das coordenadas de caixas delimitadoras obtidas pelo detector SSD, por meio do algoritmo *Centroid Tracker* [26] mostrado, na Seção 2.6. A função deste algoritmo será retornar as coordenadas x e y do ponto central onde será utilizado o valor de x, já que o deslocamento do carro ocorre na horizontal, como parâmetro de estimativa do instante em que será feito o corte e permitirá que cada carro detectado tenha identificadores únicos, reduzindo o número de falsos positivos.

Dessa maneira, chega-se ao passo final para a implementação do gatilho, onde foram criados quatro pontos (A, B, C e D) com alguns valores estimados, chamados de "*trigger points*". Uma vez obtido o valor da coordenada x, este será comparado com os valores estimados nos quatro pontos e caso seja menor que o valor do ponto estimado, o algoritmo da aplicação salvará o quadro atual e também, em variáveis, informações como: hora, minutos e segundos, assim como a data, o dia da semana e o ID do carro que foi detectado.

Esse processo foi feito por tentativa e erro, utilizando dois carros próprios pertencentes a membros do projeto, em um cenário propício, no caso uma rua com pouca

movimentação de veículos do Campus Paulo VI, pois era necessário que os carros fossem manobrados diversas vezes até que o valor estimado dos pontos fosse ideal para retornar imagens com características similares às da Figura 16. As Figuras 21 e 22 trazem as imagens obtidas de um dos carros em cada ponto em comparação às imagens desejadas.

Figura 21. Quadros infravermelhos salvos nos "*trigger points"* (esquerda) e quadro desejado (direita)




Fonte: Adaptada de [3]

Figura 22. Quadros RGB salvos nos "*trigger points"* (esquerda) e quadro desejado (direita)


Fonte: Adaptada de [3]

Sendo assim, após algumas tentativas conseguiu-se ajustar os pontos estimados de modo que alguns deles salvassem imagens que mais se aproximavam das características das entradas desejadas sendo eleitos os quadros C dos infravermelhos (Figura 21) e o quadro D

da câmera RGB (Figura 22) como os que mais se aproximavam das entradas desejadas.

Então, para verificar se as entradas seriam lidas com sucesso pelo algoritmo de capturas [3] e que o mesmo retornaria as informações referentes a quadros, as imagens foram submetidas ao algoritmo, o qual retornou os seguintes resultados mostrados na Figura 23.

Figura 23. Resultados do algoritmo de capturas para os pontos C (infravermelho) e D (RGB)


Fonte: Do Autor

Desse modo, pode ser visto na Figura 23, que as imagens passadas como entrada para o algoritmo de capturas [3], retornaram informações corretas sobre a quantidade de pessoas e os caracteres da placa. Logo, pode-se constatar que a coleta das informações do algoritmo pode ocorrer de maneira automática e estas informações poderão ser enviadas ao banco de dados.

## 3.3. Algoritmo de geração de capturas aleatórias

Devido ao fato de que apenas dois carros tenham sido usados nos testes mostrados na Seção 3.2, fez necessária a implementação de um algoritmo de geração de informação de capturas aleatórias para complementar a inserção de dados na tabela “urca”.

A razão para criação deste algoritmo se baseou no fato de ter sido estimado que, para os testes de funcionamento do URCA Analytics, seriam necessárias pelo menos cinquenta capturas. Sendo assim, foi criado um algoritmo, utilizando a linguagem *Python* que fosse capaz de retornar os seguintes resultados:

- Número aleatório de pessoas variando de 1 a 5;
- Caracteres aleatórios de placas de identificação de veículos no padrão brasileiro;
- Data, hora e dia da semana de cada registro gerado;
- Números identificadores dos quadros capturados por cada câmera;

No primeiro momento, concentrou-se na geração de placas aleatórias, onde foram criadas duas variáveis do tipo *string*, uma chamada "letras", para receber os caracteres compostos por letras, e uma chamada "numeros", para receber os caracteres numéricos, utilizando, respectivamente, os métodos *ascii_uppercase* e *digits* ambos da biblioteca *String*.

Em seguida, foram criadas mais sete variáveis, também do tipo *string*, onde três delas representavam as letras e as outras quatro os números, para representar o antigo padrão brasileiro de placas de identificação, mostrado na Figura 24. Logo após, cada variável recebeu uma entrada aleatória fornecida pelo método *choice* da biblioteca *Random*.

Figura 24. Modelo Antigo de Placas de Identificação de Veículos Brasileiros


Fonte: Site AKY Tudo[16]

Sendo assim, as variáveis que representavam letras, receberam caracteres

[16]AKY TUDO. **Placas Detran.** Disponível em < http://www.bandeiragroup.com.br/projetos/akytudo/placas-detran/>. Acessado em 6 de julho.

alfabéticos, e as que representavam números, receberam caracteres numéricos e estes foram concatenados para serem exibidos em uma única variável. Após a conclusão da geração de placas aleatórias, seguiu-se para geração aleatória da quantidade de pessoas.

Esta etapa foi feita de maneira mais simples, utilizando uma variável do tipo inteiro, denominada “ocupantes”, para representar a quantidade de pessoas, a qual recebeu o resultado do método *randint* da biblioteca *Random*, onde foi definida uma faixa de escolhas entre os números 1 e 5. Os resultados podem ser vistos na Figura 25.

Figura 25. Resultado da Geração Aleatória de Quantidade de pessoas e Placas

```
C:\Urca_Analytics\venv\Scripts\python.exe C:/Urca_Analytics/teste7.py
Digite a quantidade de itens:
5
 Quantidade de Pessoas: 5
 Placa do Veículo: XTN1289
----------------------------------------
 Quantidade de Pessoas: 1
 Placa do Veículo: YLS2597
----------------------------------------
 Quantidade de Pessoas: 3
 Placa do Veículo: PFS4531
----------------------------------------
 Quantidade de Pessoas: 3
 Placa do Veículo: ULG2825
----------------------------------------
 Quantidade de Pessoas: 5
 Placa do Veículo: BXZ3341
----------------------------------------

Process finished with exit code 0
```

Fonte: Do Autor

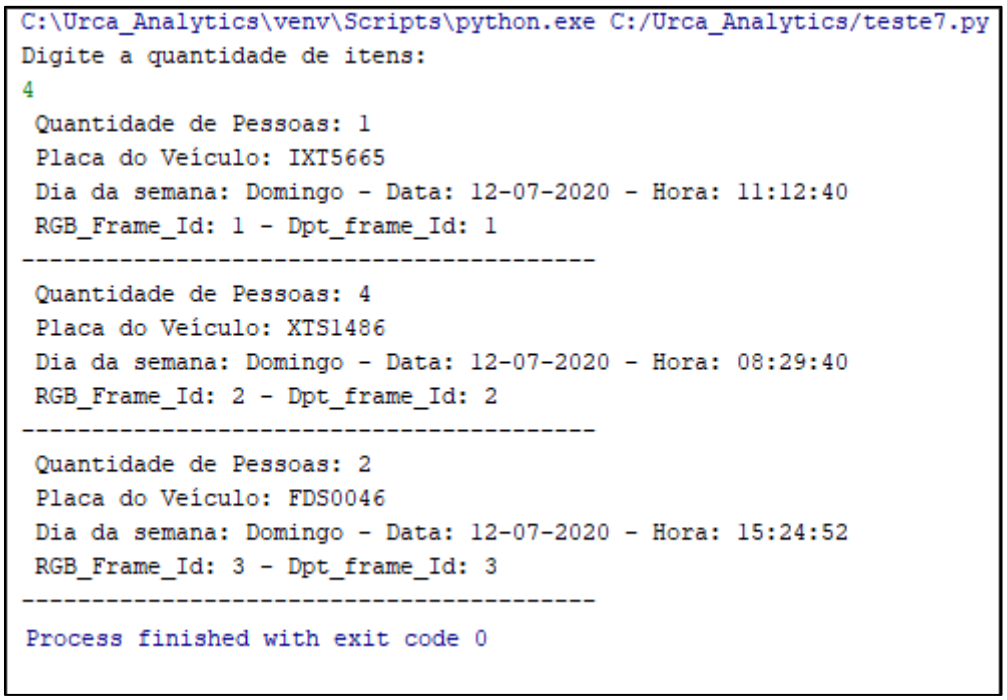
É válido ressaltar que, está sendo utilizado o padrão antigo, pelo fato deste ainda ser mais predominante no cenário atual e de que os carros utilizados nos testes ainda utilizam este padrão. Entretanto, a solução também será capaz de detectar o padrão Mercosul, uma vez que o algoritmo de capturas [3], conta com uma biblioteca especifica para o reconhecimento de caracteres de placas, chamada OpenALPR[17], a qual fornece suporte para tal padrão.

Concluídas essas duas etapas, a próxima é a geração de dados temporais aleatórios e, por fim, identificadores de quadros aleatórios. Para isso, foi utilizado o módulo *datetime*[18], especificamente objetos do tipo *timedelta* e o método *strftime*, para a geração de dados temporais e, para a geração dos identificadores, foram utilizadas variáveis como contadores

[17] *PYPIi*. ***openalpr 1.1.0***. Disponível em <https://pypi.org/project/openalpr/>. Acessado em 4 de julho de 2020.
[18] *PYTHON*. ***datetime – Basic date and time types.*** Disponível em:< https://docs.python.org/3/library/datetime.html#module-datetime>. Acessado em 4 de julho de 2020.

incrementais. Dessa forma os resultados são mostrados na Figura 26.

Figura 26. Adição de Dados Temporais às Capturas




Fonte: Do Autor

Desse modo, com a conclusão do algoritmo de geração de informações aleatórias, é possível avançar para o desenvolvimento da aplicação URCA Analytics, que será abordado na Seção 3.4.

## *3.4.* Aplicação URCA Analytics

O URCA Analytics foi uma aplicação criada com o intuito de manipular as informações geradas e salvas no banco de dados pelo algoritmo de capturas da solução [3], fazendo o tratamento das mesmas e classificando-as de acordo com regras previamente definidas.

Desse modo, a aplicação atuará de acordo com as ocorrências previstas na Tabela 1 e terá a seguinte rotina de funcionamento:

- Inicialmente o URCA Analytics acessará as informações armazenadas na tabela “urca” do banco de dados;
- Em seguida, será feita uma análise em cada uma das linhas e colunas acessadas e definidas em qual tipo de ocorrência elas se encaixam;

- Por fim, essas informações são apresentadas na tela de acompanhamento.

Para isso, a aplicação se integrará ao banco de dados criando na Seção 3.1 e a cada entrada acessada, o algoritmo analisará as informações e atribuirá à coluna “ocorrência” da tabela “urca”, uma das cinco ocorrências definidas na Tabela 1.

Tabela 1. Ocorrências definidas no URCA Analytics

| **OCORRÊNCIAS** | | |
|---|---|---|
| **Tipos** | **Requisitos** | **Cores** |
| Erros de ID | IDs de quadros diferentes | **Cinza** |
| Placas Especiais | Ambulâncias, táxis, viaturas de polícia e etc. | **Azul Ciano** |
| Fora do Horário | Fora do horário de funcionamento estabelecido | **Azul Escuro** |
| Único Ocupante | Carro trafegando com um único ocupante | **Vermelho** |
| Liberados | Carro trafegando com mais de um ocupante | **Verde** |

Fonte: Do Autor

Sendo assim, detalhando os requisitos para cada ocorrência do URCA Analytics, temos primeiramente a ocorrência “Erros de ID”, que foi pensada como um mecanismo para detectar a presença de inconsistências na atribuição de IDs de quadros capturados pelo algoritmo de capturas [3].

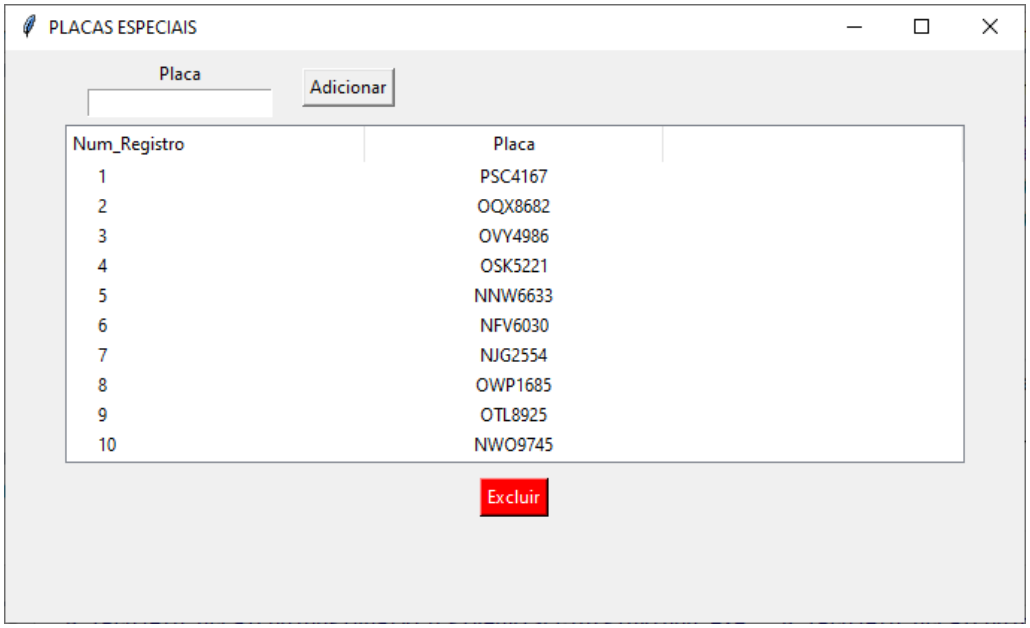
Então, a regra é que tanto o quadro RGB, tanto o quadro infravermelho, tenham IDs idênticos pois isto indica que ambos os quadros pertençam ao mesmo carro (evitando erros de atribuição como, por exemplo, reconhecimento de passageiros de carro A atribuído na placa de carro B) e caso sejam verificados que os IDs são diferentes, na coluna “ocorrência” da tabela “urca” na linha referente ao erro, será atribuída a expressão “Erro de ID”.

Já a ocorrência “Placa Especial” está direcionada a placas consideradas “especiais”, as quais pode-se citar: placas de ambulâncias, de viaturas policiais, de carros de autoescola, de táxis, dentre outros. O motivo da criação desta ocorrência foi para isentar veículos que possuem essas características, das regras estabelecidas pela solução, pois o foco, inicialmente, é coletar informações de carros particulares.

Entretanto, a atribuição desta ocorrência está condicionada ao cadastro prévio dessas placas na tabela “placas_especiais” (Figura 28) do banco de dados. Em vista disso, criou-se

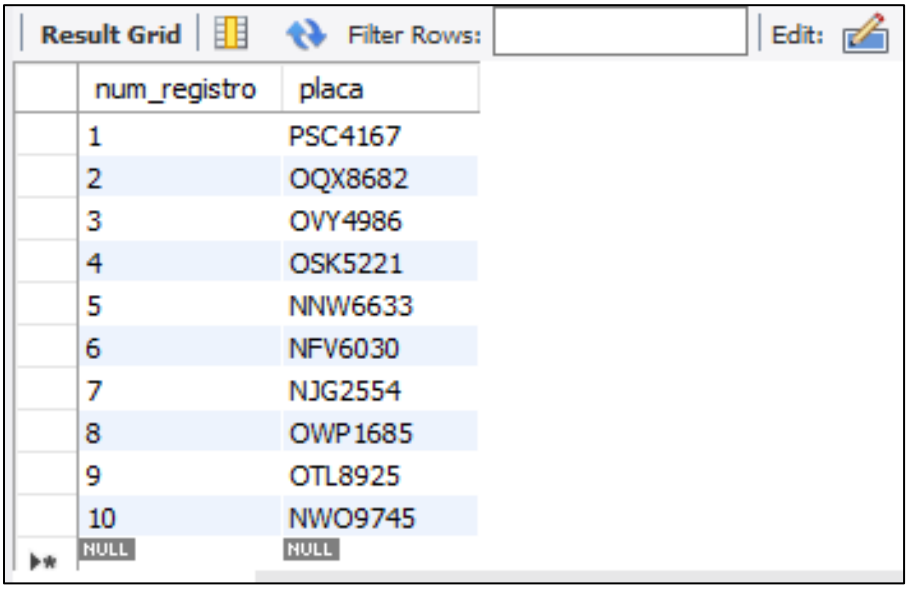
um meio de inserção e exclusão destas placas, na tabela em questão, por meio de uma interface simples mostrada na Figura 27.

Figura 27. Adição e exclusão de placas na tabela “placas_especiais”




Fonte: Do Autor

Figura 28. Tabela “placas_especiais” preenchida no MySQL Workbench pelo URCA Analytics




Fonte: Do Autor

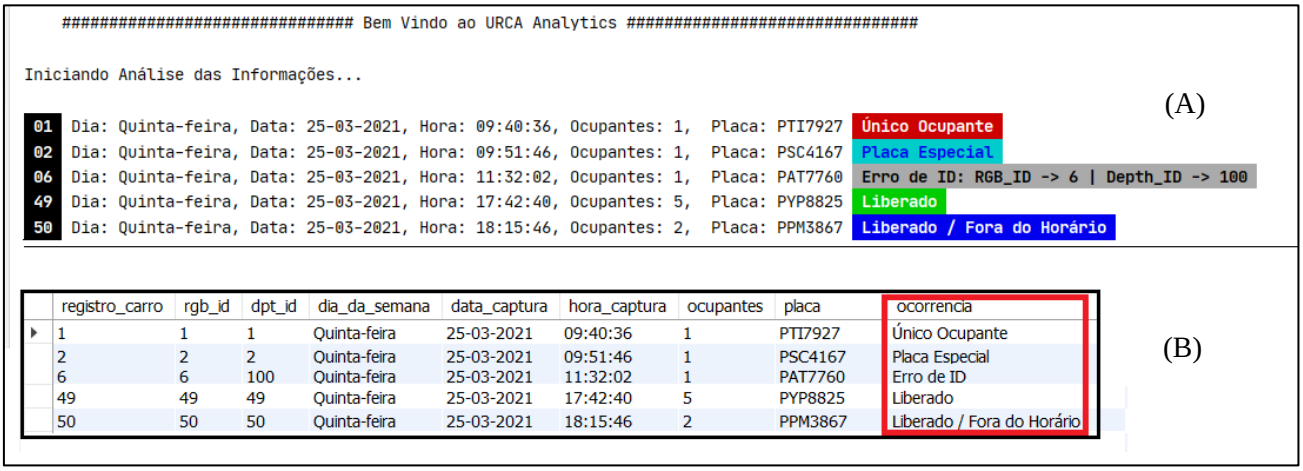
Nesse caso, se uma determinada placa que for capturada estiver presente na tabela “placas_especiais” do banco de dados mostrada na Figura 28, a coluna “ocorrência” da tabela

“urca” será preenchida com “Placa Especial”, como mostra a Figura 29 (B).

A próxima ocorrência, “Fora do Horário” será aplicada quando um carro passar pelo campo de visão das câmeras antes ou após o horário de funcionamento da solução. Para exemplificar, pode-se supor que o horário de funcionamento foi estabelecido das 8:00 às 18:00 horas. Nesse caso, se um carro passar às 7:00 ou às 19:00, suas informações serão registradas na coluna “ocorrências” da tabela “urca” como “Fora do Horário”, pois a solução ainda não se encontra em funcionamento ou já foi encerrada.

Por fim, têm-se as ocorrências “Único Ocupante” e “Liberado”, que serão atribuídas aos carros que passarem no campo de visão da solução durante o horário de funcionamento estabelecido, sendo que a primeira, será vinculada aos carros que em que a quantidade de ocupantes no interior do veículo for de apenas uma pessoa e a segunda quando a quantidade de ocupantes for superior. De forma análoga, ao procedimento das outras ocorrências, estas duas últimas também serão inseridas pelo URCA Analytics na coluna “ocorrência” da tabela “urca” e todas podem ser vistas na Figura 29 (A), que traz a tela de acompanhamento do URCA Analytics

Figura 29. Tela de acompanhamento do URCA Analytics (A) e inserção de ocorrências na coluna “ocorrência” da tabela “urca” vista no MySQL Workbench (B).

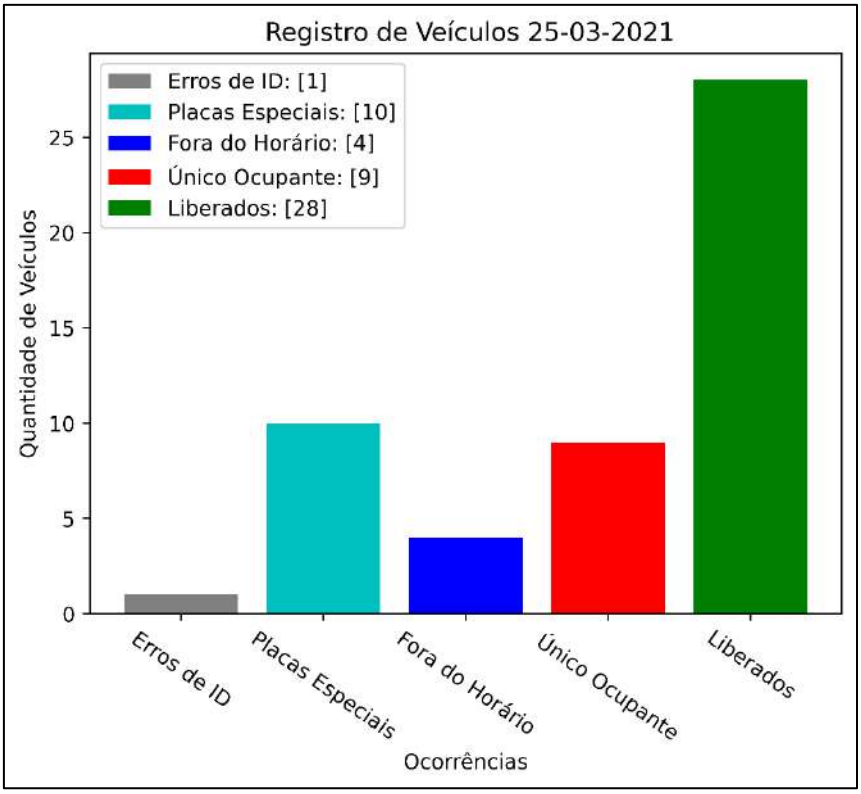



Fonte: Adaptado do Autor

Além da tela de acompanhamento mostrada na Figura 29 (A), o URCA Analytics possui funcionalidades como: a geração de gráficos (Figura 30), permitir a buscas seletivas de dados, onde o operador pode retornar e mostrar na tela apenas os dados que lhe são de interesse (Figura 31), assim como a gravação de listas contendo as informações em arquivos

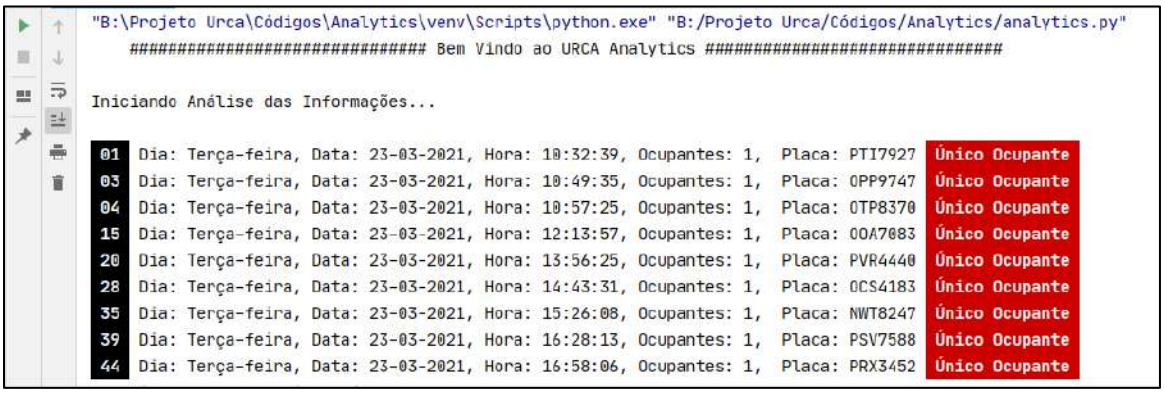
.CSV, como mostrado na Figura 32.

Figura 30. Gráfico de barras de Ocorrências por quantidade de veículos gerado pelo URCA Analytics



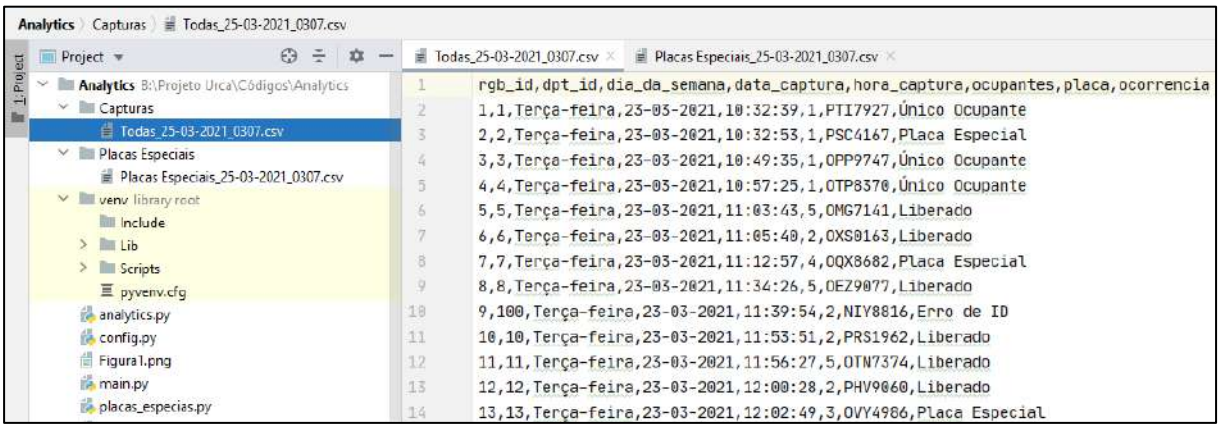

Fonte: Do Autor

Figura 31. Consulta de ocorrências específicas retornando apenas as marcadas como Único Ocupante




Fonte: Do Autor

Figura 32. Geração de lista .CSV de todas as capturas e das placas especiais



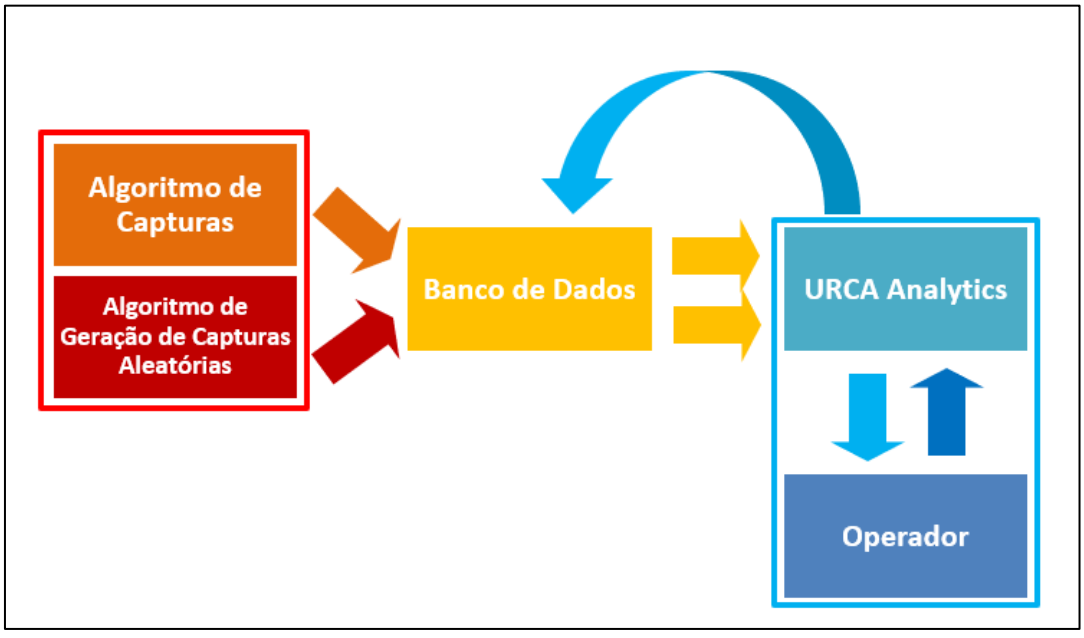

Fonte: Do Autor

Sendo assim, com aplicação URCA Analytics desenvolvida, resta fazer a integração das etapas concluídas e em seguida implantar a solução para ver os resultados.

## 3.5. Integração das aplicações

Após o desenvolvimento das etapas apresentadas nas seções de 3.1 a 3.4, é preciso fazer integração de todo o conjunto para que o sistema possa ser implantado. Sendo assim a solução funcionará de acordo com esquema mostrado na Figura 33:

Figura 33. Esquema de Integração das Aplicações




Fonte: Do Autor

Explicando o esquema da Figura 33, têm-se que o algoritmo de capturas, inicialmente envia informações dos carros capturados pelas câmeras para à tabela “urca” e em seguida, a tabela é preenchida pelo algoritmo de geração de capturas aleatórias, haja vista que, para estes testes foram utilizados apenas dois carros, uma quantidade considerada pequena para os testes.

Em seguida, quando a aplicação URCA Analytics é ativada pelo operador e tem acesso a essas informações, iniciando o processo de tratamento, baseado nas regras definidas, para cada entrada o URCA Analytics irá interagir com o banco de dados adicionando informações na coluna “ocorrências” da tabela “urca”. Além dessa interação para adição das ocorrências a aplicação também interage quando o operador insere placas para serem cadastradas na tabela “placas_especiais”.

Dessa forma, utilizou-se a biblioteca *MySQLdb*[19] para conectar as aplicações desenvolvidas em *Python* ao banco de dados implementado no MySQL WorkBench como é mostrado no trecho do código na Figura 34.

Figura 34. Codificação da conexão da aplicação URCA Analytics ao banco de dados

```
1   import MySQLdb
2   from tkinter import *
3   from config import config
4   import time
5   import matplotlib.pyplot as plt
6   from datetime import date
7
8   cnx = MySQLdb.connect(**config, charset='utf8')
9   cursor = cnx.cursor(MySQLdb.cursors.DictCursor)
10  capturas = None
11  oc = None
12
13  class Analytics:
```

Fonte: Do Autor

A Figura 34, traz apenas a conexão do URCA Analytics com o banco de dados como exemplo, devido ao processo ser exatamente o mesmo, tanto no algoritmo de capturas, quanto no algoritmo de geração de capturas aleatórias. Logo, com a conclusão

[19] *PYPI*. ***mysqlclient 2.0.3.*** Disponível em: < https://pypi.org/project/mysqlclient/>. Acessado em 20 de março de 2021

da etapa de integração das aplicações ao banco de dados partiu-se para instalação da solução e análise dos resultados, os quais serão tratados no Capítulo 4.

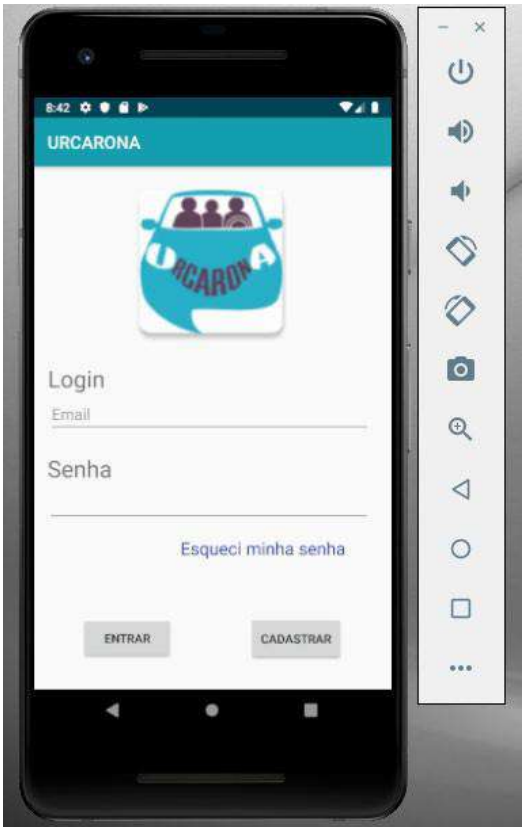
## 3.6. Aplicativo URCARONA

Ainda na fase de desenvolvimento, a solução também conta com o URCARONA, que é um aplicativo que faz parte do conjunto da solução URCA, desenvolvido para ser um intermediário que conectará as pessoas que desejam utilizar caronas e os motoristas, o qual funcionará de acordo com as seguintes premissas e restrições [3]:

- As caronas são solidárias, não havendo troca monetária por elas;
- Os usuários devem passar por um processo de validação de cadastro para verificação de antecedentes criminais;
- Haverá uma pontuação para cada carona, onde usuários com pontuação baixa deverão ser excluídos do sistema.

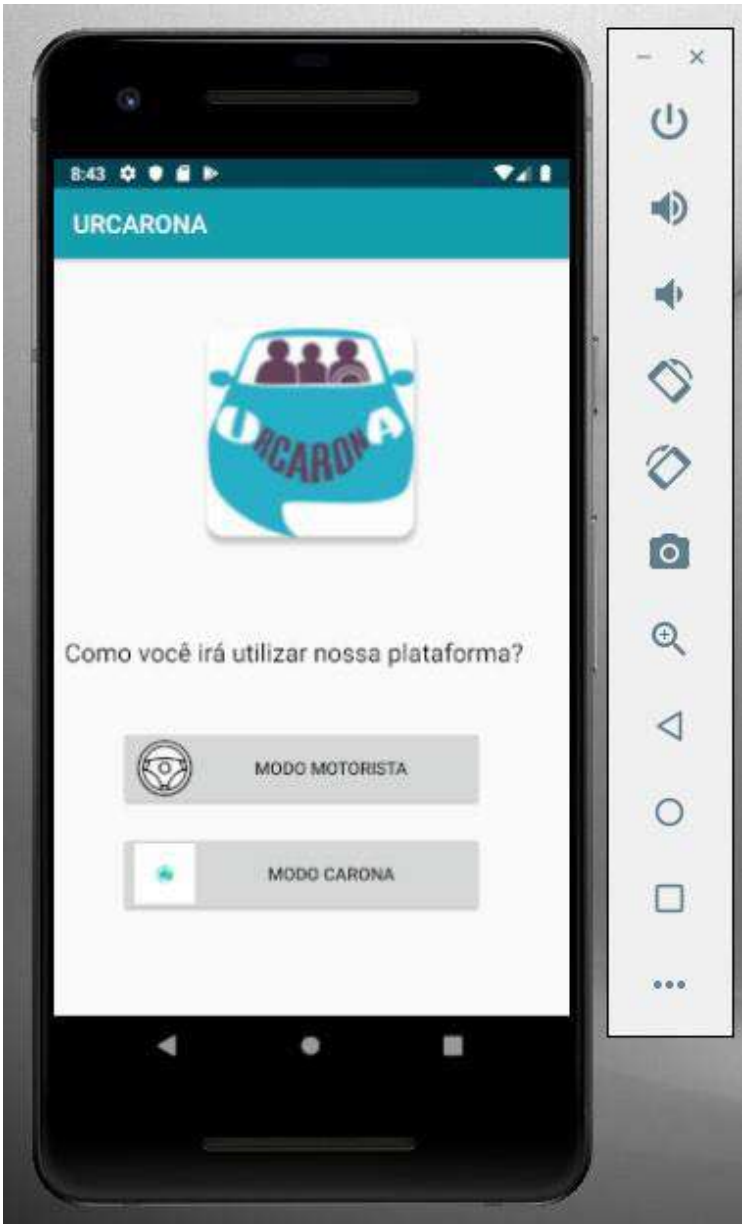
Até o momento, o aplicativo conta com as seguintes telas [3]:

Figura 35. Tela de Login URCARONA



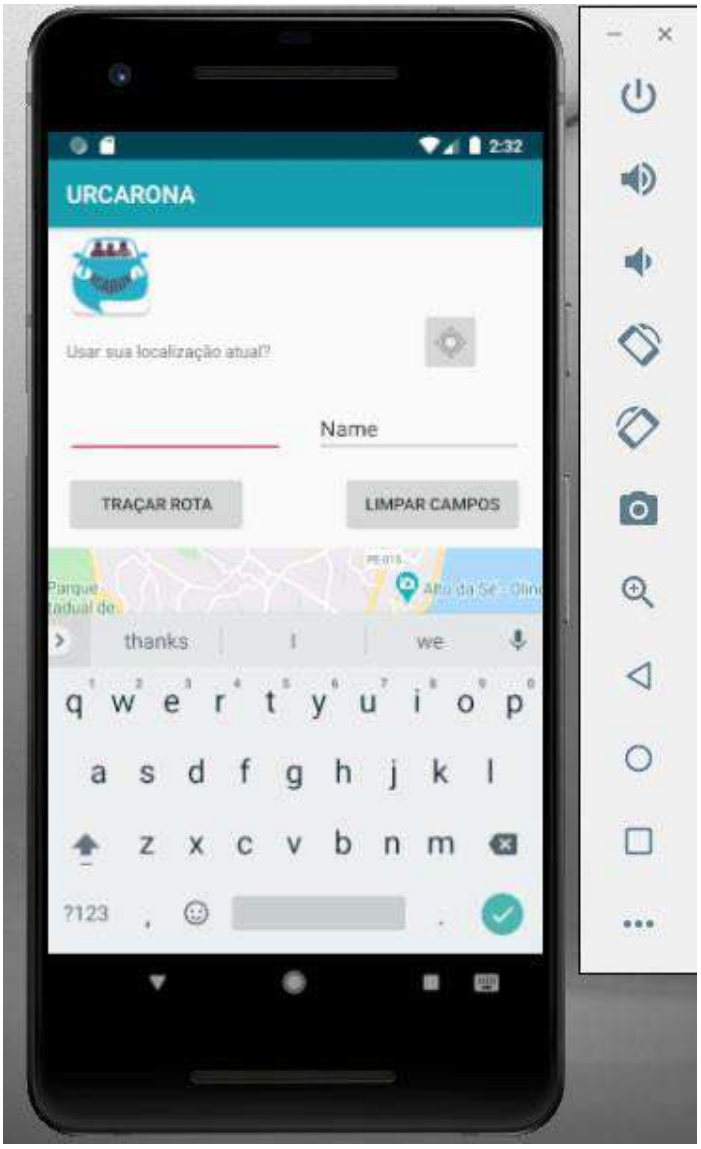

Fonte: [3]

Figura 36. Modo de Atuação URCARONA




Fonte: [3]

Figura 37. Oferecer Carona URCARONA




Fonte: [3]

Até o momento, o aplicativo encontra-se na versão 1.0 e apresenta os requisitos mínimos de funcionamento como cadastro e rotas de carona que estão sendo testados para detecção de erros. Porém, o seu desenvolvimento gerou um certificado de registro de software e o registro de uma marca, mostrados no APENDICE C.

# 4. IMPLANTAÇÃO DA SOLUÇÃO E RESULTADOS

Para realização de testes e análise dos resultados, a solução URCA foi implantada em uma rua com baixa movimentação de carros do Campus Paulo VI – UEMA, para que os testes pudessem ser feitos com mais segurança e melhores condições.

Vale ressaltar que a ideia principal é que esta solução seja implantada no pórtico do Campus, na pista de entrada, entretanto houve muita dificuldade na realização de testes no pórtico, devido a constantes interrupções pelos carros que passavam pela via de entrada, que muitas vezes paravam para pedir informações e também pelo fato de não haver segurança para manobrar os carros utilizados.

Desse modo, foi utilizada uma pista interna do campus, onde os carros foram posicionados de modo a simular o sentido que os carros passam na pista de entrada, no caso da esquerda para direita, para futuramente, quando a solução tiver sido completamente ajustada e testada, possa ser retomada a ideia de sua implantação no pórtico. O desenvolvimento desta etapa é descrito ao longo deste capítulo com os resultados obtidos durante o processo.

## 4.1. Instalação e ajustes dos dispositivos em campo

Nesta etapa, o dispositivo *Intel Realsense D435* foi posicionado com suas câmeras focadas em ângulo estimado na faixa entre 70 a 90 graus [3], e em seguida, foram medidas as distâncias da altura da câmera até ao chão e da câmera ao ponto em que o carro passará na pista de rolamento.

Para isso, um dos carros foi estacionado lateralmente na pista de rolamento para permitir a visualização das janelas e em frente às câmeras, onde, em seguida, o algoritmo de capturas foi ativado de modo a permitir a visualização do campo de visão dos quadros infravermelhos e RGB como mostra a Figura 38.

Logo após, uma trena foi utilizada, de forma que uma das extremidades foi fixada ao carro e a outra foi deslocada no sentindo horizontal junto com as câmeras fixadas a um tripé, no intuito de encontrar a melhor distância para visualizar a parte interna do veículo, sendo observados os quadros infravermelhos.

Desse modo, chegou-se à conclusão que com as distâncias de 3,5 metros (da câmera em relação ao carro) e de 1,3 metros (da câmera em relação ao chão) era possível a visualização do interior do veículo.

Na sequência, o carro foi deslocado aos poucos para esquerda, desta vez sendo observados os quadros RGB, com o objetivo de verificar se também era possível visualizar os caracteres da placa do veículo, sendo constado que naquela posição em um ponto perto do final do campo de visão à direita era possível visualizar os caracteres lateralmente, como visto na Figura 38.

Figura 38. Visualização do carro por meio do algoritmo de capturas.

Fonte: Do Autor

A Figura 38 mostra a visualização por meio do algoritmo de capturas a 3,5 metros do carro e altura de 1,3 metros, nas câmeras de infravermelho e RGB, onde nos quadros acima, o carro está posicionado exatamente em frente às câmeras e percebe-se, no quadro infravermelho, a visualização do interior do veículo. Já nos quadros abaixo, o carro foi deslocado para direita até um ponto onde fosse possível identificar os caracteres da placa pelo quadro RGB. Sendo assim, foram feitos testes com os veículos em movimento para ver os resultados.

## 4.2. Execução dos algoritmos da solução e resultados

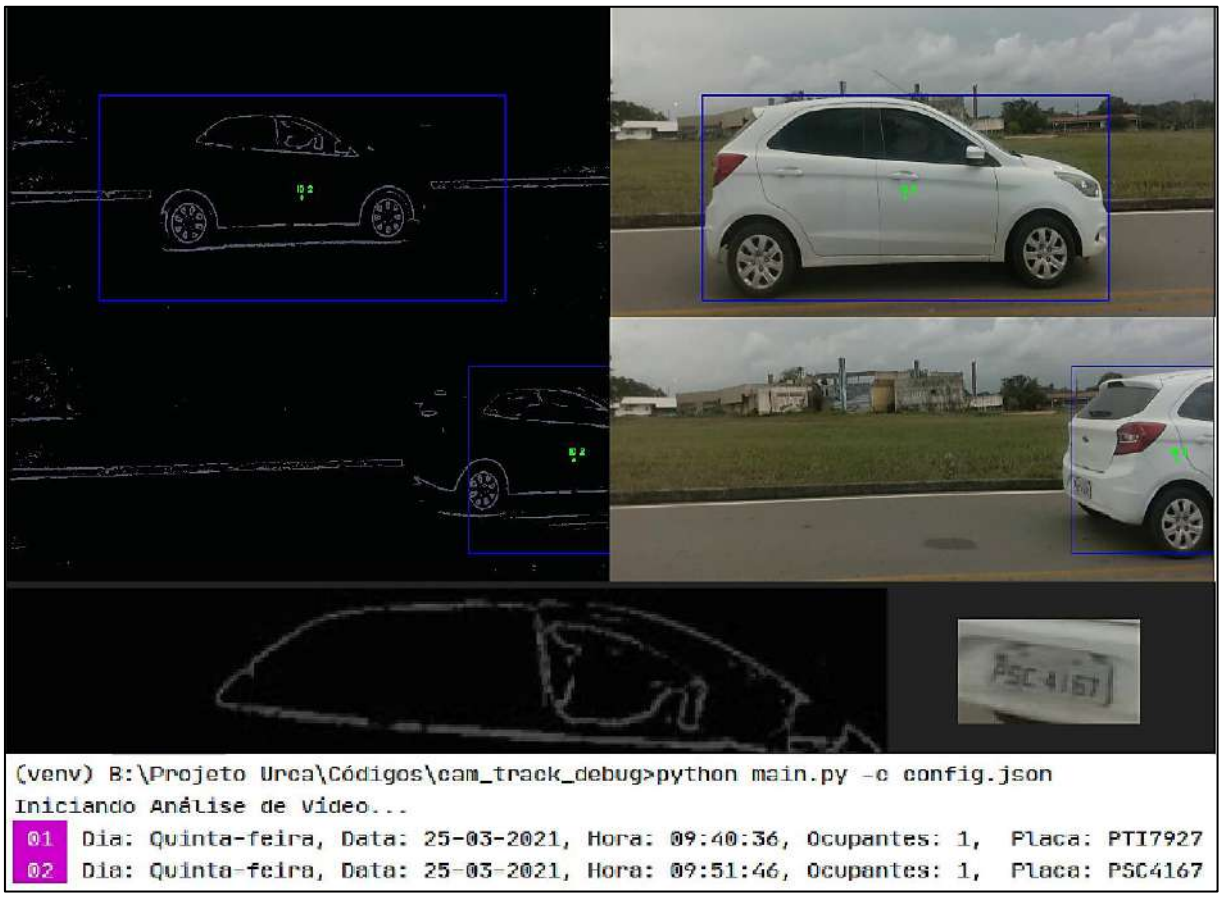
Nesse ponto, foi feita a execução dos algoritmos da solução em conjunto, primeiramente com o algoritmo de capturas e o algoritmo de geração de capturas aleatórias e, em seguida, o URCA Analytics.

Para esta etapa, foi utilizado um notebook contendo a aplicação de capturas, o algoritmo de geração de capturas aleatórias e o URCA Analytics, além de dois carros que passaram pelo campo de visão da solução e após a execução do algoritmo de capturas e do algoritmo de geração de capturas aleatórias, foram gerados os resultados mostrados nas Figuras 39, 40 e 41.

Figura 39. Resultados do primeiro carro passando pelo algoritmo de capturas

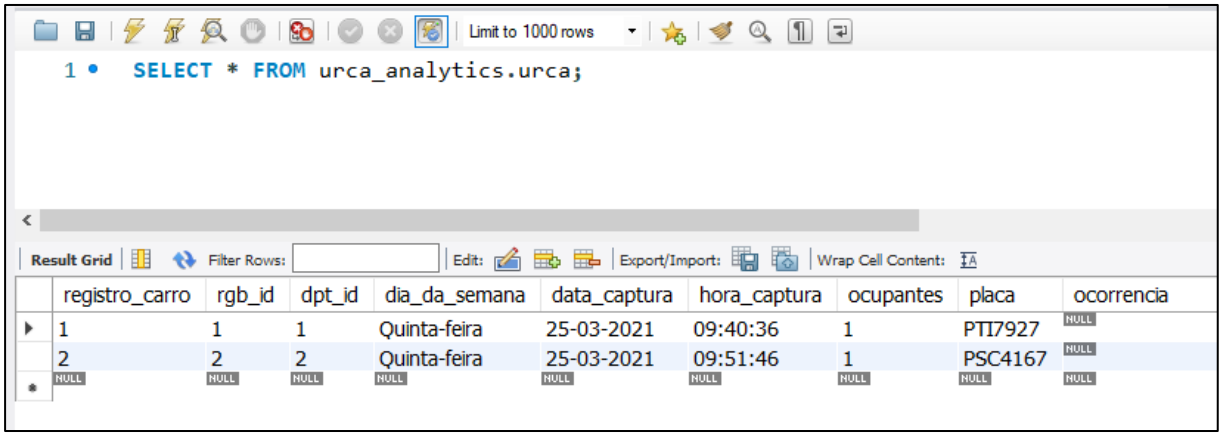

Fonte: Do Autor

Figura 40. Resultados do segundo carro passando pelo algoritmo de capturas




Fonte: Do Autor

Figura 41. Inserção das informações no banco de dados feita pelo algoritmo de capturas




Fonte: Do Autor

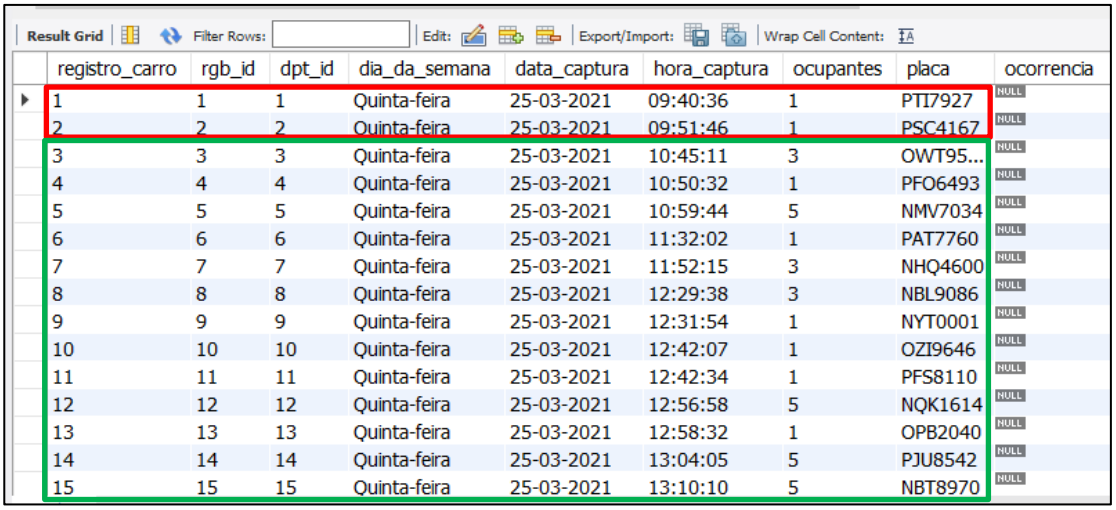
Figura 42. Resultados da execução do algoritmo de geração de capturas aleatórias

```
(venv) B:\Projeto Urca\Códigos\cam_track_debug>python main.py -c config.json

Iniciando Preenchimento Aleatório...
03  Dia: Quinta-feira, Data: 25-03-2021, Hora: 10:45:11, Ocupantes: 3,  Placa: OWT9510
04  Dia: Quinta-feira, Data: 25-03-2021, Hora: 10:50:32, Ocupantes: 1,  Placa: PFO6493
05  Dia: Quinta-feira, Data: 25-03-2021, Hora: 10:59:44, Ocupantes: 5,  Placa: NMV7034
06  Dia: Quinta-feira, Data: 25-03-2021, Hora: 11:32:02, Ocupantes: 1,  Placa: PAT7760
07  Dia: Quinta-feira, Data: 25-03-2021, Hora: 11:52:15, Ocupantes: 3,  Placa: NHQ4600
08  Dia: Quinta-feira, Data: 25-03-2021, Hora: 12:29:38, Ocupantes: 3,  Placa: NBL9086
09  Dia: Quinta-feira, Data: 25-03-2021, Hora: 12:31:54, Ocupantes: 1,  Placa: NYT0001
10  Dia: Quinta-feira, Data: 25-03-2021, Hora: 12:42:07, Ocupantes: 1,  Placa: OZI9646
11  Dia: Quinta-feira, Data: 25-03-2021, Hora: 12:42:34, Ocupantes: 1,  Placa: PFS8110
12  Dia: Quinta-feira, Data: 25-03-2021, Hora: 12:56:58, Ocupantes: 5,  Placa: NQK1614
13  Dia: Quinta-feira, Data: 25-03-2021, Hora: 12:58:32, Ocupantes: 1,  Placa: OPB2040
14  Dia: Quinta-feira, Data: 25-03-2021, Hora: 13:04:05, Ocupantes: 5,  Placa: PJU8542
15  Dia: Quinta-feira, Data: 25-03-2021, Hora: 13:10:10, Ocupantes: 5,  Placa: NBT8970
16  Dia: Quinta-feira, Data: 25-03-2021, Hora: 13:11:28, Ocupantes: 5,  Placa: OLK2343
17  Dia: Quinta-feira, Data: 25-03-2021, Hora: 13:21:47, Ocupantes: 4,  Placa: NJT3645
18  Dia: Quinta-feira, Data: 25-03-2021, Hora: 13:29:28, Ocupantes: 2,  Placa: NRQ9546
19  Dia: Quinta-feira, Data: 25-03-2021, Hora: 13:37:30, Ocupantes: 2,  Placa: PBI2864
20  Dia: Quinta-feira, Data: 25-03-2021, Hora: 14:02:35, Ocupantes: 1,  Placa: NTI2020
21  Dia: Quinta-feira, Data: 25-03-2021, Hora: 14:12:56, Ocupantes: 5,  Placa: OFZ5751
22  Dia: Quinta-feira, Data: 25-03-2021, Hora: 14:15:49, Ocupantes: 5,  Placa: PCH7060
23  Dia: Quinta-feira, Data: 25-03-2021, Hora: 14:22:07, Ocupantes: 4,  Placa: NVD0621
```

Fonte: Do Autor

Figura 43. Inserção dos dados no banco de dados pelo algoritmo de capturas e pelo algoritmo de geração de capturas aleatórias

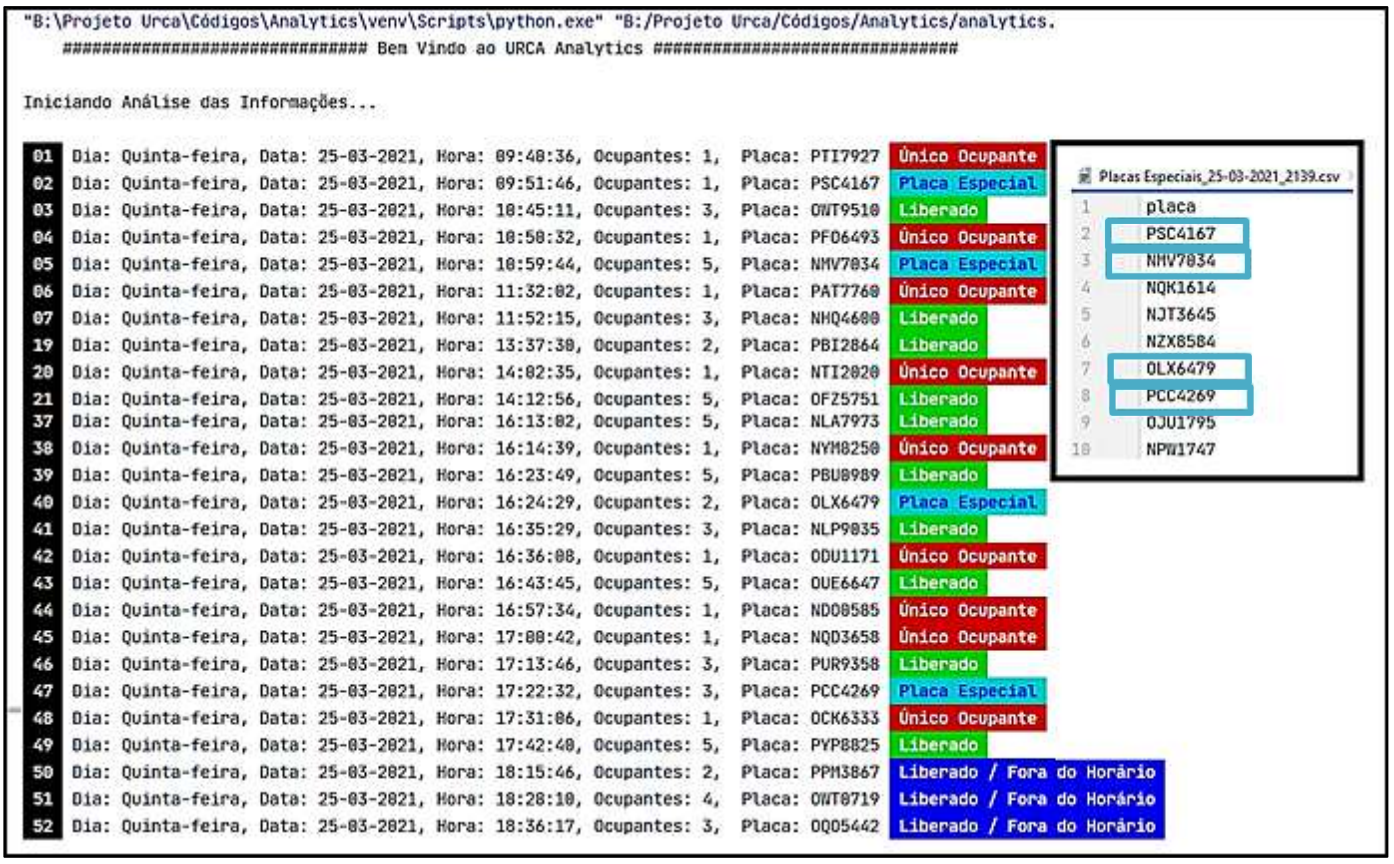



| registro_carro | rgb_id | dpt_id | dia_da_semana | data_captura | hora_captura | ocupantes | placa | ocorrencia |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | Quinta-feira | 25-03-2021 | 09:40:36 | 1 | PTI7927 | NULL |
| 2 | 2 | 2 | Quinta-feira | 25-03-2021 | 09:51:46 | 1 | PSC4167 | NULL |
| 3 | 3 | 3 | Quinta-feira | 25-03-2021 | 10:45:11 | 3 | OWT95... | NULL |
| 4 | 4 | 4 | Quinta-feira | 25-03-2021 | 10:50:32 | 1 | PFO6493 | NULL |
| 5 | 5 | 5 | Quinta-feira | 25-03-2021 | 10:59:44 | 5 | NMV7034 | NULL |
| 6 | 6 | 6 | Quinta-feira | 25-03-2021 | 11:32:02 | 1 | PAT7760 | NULL |
| 7 | 7 | 7 | Quinta-feira | 25-03-2021 | 11:52:15 | 3 | NHQ4600 | NULL |
| 8 | 8 | 8 | Quinta-feira | 25-03-2021 | 12:29:38 | 3 | NBL9086 | NULL |
| 9 | 9 | 9 | Quinta-feira | 25-03-2021 | 12:31:54 | 1 | NYT0001 | NULL |
| 10 | 10 | 10 | Quinta-feira | 25-03-2021 | 12:42:07 | 1 | OZI9646 | NULL |
| 11 | 11 | 11 | Quinta-feira | 25-03-2021 | 12:42:34 | 1 | PFS8110 | NULL |
| 12 | 12 | 12 | Quinta-feira | 25-03-2021 | 12:56:58 | 5 | NQK1614 | NULL |
| 13 | 13 | 13 | Quinta-feira | 25-03-2021 | 12:58:32 | 1 | OPB2040 | NULL |
| 14 | 14 | 14 | Quinta-feira | 25-03-2021 | 13:04:05 | 5 | PJU8542 | NULL |
| 15 | 15 | 15 | Quinta-feira | 25-03-2021 | 13:10:10 | 5 | NBT8970 | NULL |

Fonte: Do Autor

Na Figura 43, destacado em vermelho estão as linhas preenchidas pelo algoritmo de capturas e, destacadas em verde, estão as linhas preenchidas pelo algoritmo de geração de capturas aleatórias. Sendo assim, com o banco de dados povoado, é possível executar o URCA Analytics para que ele faça o tratamento das informações que foram armazenadas. Dessa forma, os resultados são apresentados nas Figuras 44 e 45.

Figura 44. Resultados do URCA Analytics

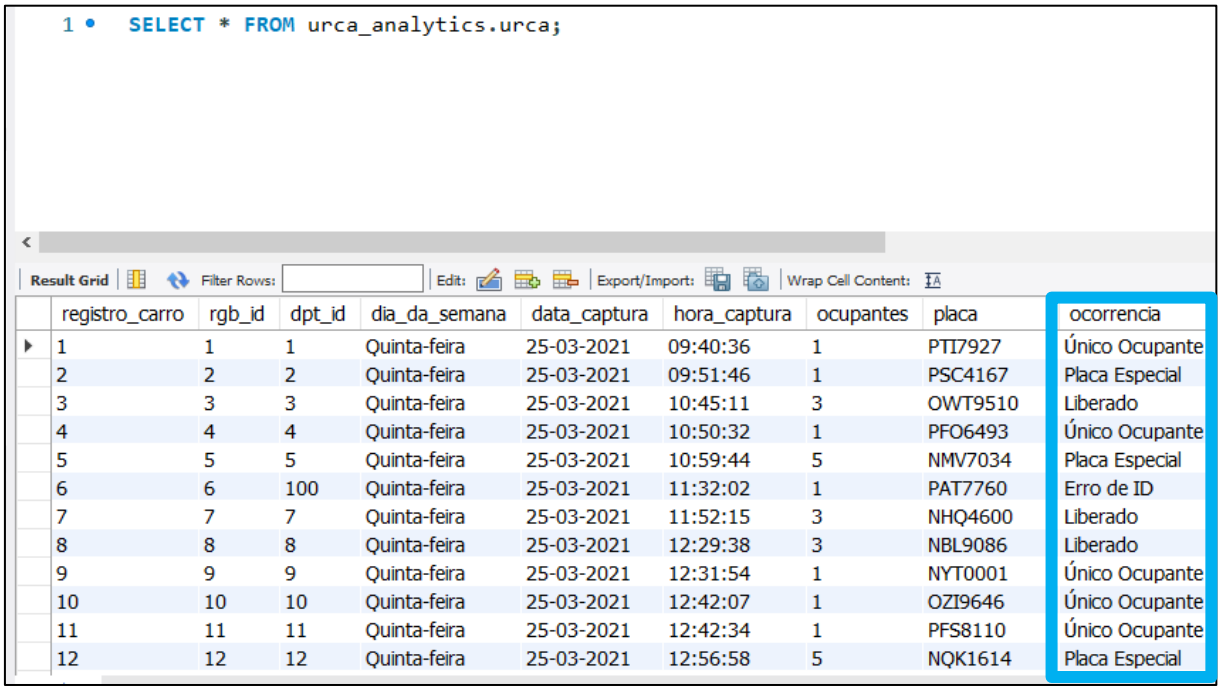



Fonte: Do Autor

A Figura 44 traz as capturas devidamente sinalizadas com cada ocorrência e também se nota que as placas que foram capturadas e que pertenciam à lista de placas especiais, inserida manualmente por meio do URCA Analytics, foram destacadas na cor azul ciano, assim como as outras que também foram marcadas com suas respectivas cores.

Sendo assim, resta mais uma verificação, que é se a aplicação URCA Analytics atualizou no banco de dados as ocorrências para cada linha e o resultado é apresentado na Figura 45.

Figura 45. Atualização da coluna “ocorrências” pelo URCA Analytics



```
1 •  SELECT * FROM urca_analytics.urca;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

| registro_carro | rgb_id | dpt_id | dia_da_semana | data_captura | hora_captura | ocupantes | placa | ocorrencia |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | Quinta-feira | 25-03-2021 | 09:40:36 | 1 | PTI7927 | Único Ocupante |
| 2 | 2 | 2 | Quinta-feira | 25-03-2021 | 09:51:46 | 1 | PSC4167 | Placa Especial |
| 3 | 3 | 3 | Quinta-feira | 25-03-2021 | 10:45:11 | 3 | OWT9510 | Liberado |
| 4 | 4 | 4 | Quinta-feira | 25-03-2021 | 10:50:32 | 1 | PFO6493 | Único Ocupante |
| 5 | 5 | 5 | Quinta-feira | 25-03-2021 | 10:59:44 | 5 | NMV7034 | Placa Especial |
| 6 | 6 | 100 | Quinta-feira | 25-03-2021 | 11:32:02 | 1 | PAT7760 | Erro de ID |
| 7 | 7 | 7 | Quinta-feira | 25-03-2021 | 11:52:15 | 3 | NHQ4600 | Liberado |
| 8 | 8 | 8 | Quinta-feira | 25-03-2021 | 12:29:38 | 3 | NBL9086 | Liberado |
| 9 | 9 | 9 | Quinta-feira | 25-03-2021 | 12:31:54 | 1 | NYT0001 | Único Ocupante |
| 10 | 10 | 10 | Quinta-feira | 25-03-2021 | 12:42:07 | 1 | OZI9646 | Único Ocupante |
| 11 | 11 | 11 | Quinta-feira | 25-03-2021 | 12:42:34 | 1 | PFS8110 | Único Ocupante |
| 12 | 12 | 12 | Quinta-feira | 25-03-2021 | 12:56:58 | 5 | NQK1614 | Placa Especial |

Fonte: Do Autor

Como é observado na Figura 45, as linhas da coluna “ocorrências” foram atualizadas com sucesso e de acordo com o tratamento das informações feito pelo URCA Analytics, mostrando que a integração do conjunto de aplicações obteve sucesso e que a implantação da solução, nessas condições, teve resultados satisfatórios.

## 4.3. Confiabilidade e Acurácia dos testes

Quanto à confiabilidade e acurácia dos resultados apresentados, foram utilizados, como base, os resultados de campo apresentados por Costa [3] nas velocidades de 20, 40 e 60 km/h, onde, naquele momento, os testes mostraram um índice de confiabilidade na detecção das placas de 78% e 93% nas velocidades de 20 e 40 km/h, respectivamente, e uma falha em apenas um caractere na detecção a 60 km/h e quanto à acurácia na detecção de pessoas um índice de aproximadamente 90%.

Sendo assim, os testes demostrados neste trabalho foram realizados com velocidades entre 20 e 30 km/h, apresentando confiabilidade na detecção de placas no carro 1 de 92.30% e no carro 2 de 92.70% e mantendo a acurácia na detecção de pessoas

na faixa de 90%.

Vale ressaltar que as resoluções utilizadas nas câmeras para estes testes foram de 1280 x 720, ganhando uma melhora expressiva na detecção de placas, as quais foram alcançadas devido às recentes atualizações nos firmwares da câmera, o que permitiu a utilização de resoluções superiores, sendo que durante os testes realizados no primeiro momento por Costa [3], só era possível utilizar resoluções inferiores. Sendo assim, faz-se necessária a realização de novos testes, em um ambiente seguro, para verificar se também há melhorias no índice de confiabilidade nas velocidades de 40 e 60 km/h, os quais serão sugeridos como trabalhos futuros.

# 5. TRABALHOS FUTUROS

Como trabalhos futuros, pode-se sugerir o estudo de um meio para viabilizar a automatização das capturas de placas especiais, partindo da premissa de que estas possuem cores diferenciadas das placas de veículos particulares e que talvez seja possível utilizar estas características juntamente com técnicas de visão computacional para alcançar este objetivo.

Também pode-se sugerir a realização de novos testes de campo, em um ambiente seguro, nas velocidades de 40 e 60 km/h com a resolução das câmeras em 1280 x 720, para verificar se há aumento no índice de confiabilidade da detecção de placas e também de pessoas.

Além disso, também será preciso pensar em um meio para proteger o equipamento quando a solução for implantada no ambiente externo, bem como a integração do aplicativo URCARONA a todo o conjunto da solução.

# 6. CONCLUSÕES

Desse modo, com a conclusão das etapas propostas neste trabalho, foi possível observar por meio dos resultados obtidos na implantação, que os objetivos apresentados na Seção 1.1 foram alcançados e a solução URCA tem grande potencial de utilização em um cenário como o apresentado, abrindo espaço para testes da aplicação da solução em outros cenários, onde seria possível o registro de mais carros como, por exemplo, no pórtico do Campus Paulo VI da UEMA.

Também é importante ressaltar que para os avanços dos testes, faz-se necessário, a aquisição ou desenvolvimento de um dispositivo de proteção para os equipamentos no local da instalação, uma vez que a ideia é que a solução atue em um ambiente *outdoor* e, dessa maneira, os equipamentos ficarão protegidos de ações climáticas e de vandalismo.

Sendo assim, no período de desenvolvimento deste trabalho, além dos resultados apresentados, o projeto URCA também foi contemplado com a concessão do Certificado de Registro de Programa de Computador Junto ao INPI (APÊNDICE C), para o aplicativo de caronas, intitulado “URCARONA v.1.0” e já se trabalha para dar entrada no pedido de registro da aplicação URCA Analytics.

Logo, com a conclusão desta etapa, foi possível adquirir conhecimentos muito válidos na área de visão computacional e do rastreamento e detecção de objetos, que certamente podem ser usados como ferramentas para aperfeiçoamento das aplicações, para, em um futuro breve, a solução ganhe as ruas e avenidas das cidades.

## REFERÊNCIAS


[1] THOMAZ, V. F. **Solução tecnológica de reconhecimento de imagens para o projeto URCA de Uso Racional de Carros nas Cidades Inteligentes**. TCC (Graduação em Engenharia da Computação) – Universidade Estadual do Maranhão. São Luís, p.74. 2017.

[2] COSTA, A. P. F. **Solução tecnológica de captura e armazenamento de dados para o projeto URCA**. TCC (Graduação em Engenharia da Computação) – Universidade Estadual do Maranhão. São Luís, p.72. 2017.

[3] COSTA, A. P. F. **URCA – Uso Racional de Carros nas Cidades Inteligentes**. Dissertação (Mestrado em Engenharia da Computação e Sistemas) – PECS, Universidade Estadual do Maranhão. São Luís, p.108. 2020.

[4] COSTA, A. P. F. URCA – **Uso Racional de Carros nas Cidades Inteligentes**. TCC Qualificação (Mestrado em Engenharia da Computação e Sistemas) – PECS, Universidade Estadual do Maranhão. São Luís, p.60. 2018.

[5] SOLOMON, B. ***Generating Random Data in Python(Guide)***. *Real Python*. Disponível em: <https://realPython.com/Python-random/> Acessado em: 15 de junho de 2020.

*[6] TUTORIALS POINT*. ***Generating random number list in Python***. Disponível em: <https://www.tutorialspoint.com/generating-random-number-list-in-Python>. Acessado em: 14 de junho de 2020.

[7] *PYNATIVE*. ***Python Random Module to Generate Random Data***. Disponível em: <https://pynative.com/Python-random-module/>. Acessado em: 30 de junho de 2020.

[8] *Python*. **Generate pseudo-random numbers**. Random. Documentation. Disponível em: < https://docs.Python.org/3/library/random.html />. Acessado em: 7 de julho de 2020.

[9] *Python*. ***Common string operations.*** *String. Documentation*. Disponível em: <https://docs.Python.org/pt-br/3/library/string.html#string.ascii_lowercase>. Acessado em: 7 de julho de 2020.

[10] HEUSER, C, A. **Projeto de Banco de Dados**. 4 ed. Instituto de Informática da UFRGS. 1998.

[11] SINGH, S. ***DATABASE MANAGEMENT SYSTEM***. *Review Article. Journal of Management Research and Analysis*. Faculty of Management Studies, BHU, Varanasi, India. 2015.

[12] DATE, C. J. **Introdução a sistemas de bancos de dados**. 8. ed. Rio de Janeiro: Elsevier, 2003.

[13] SETZER, V. W.; SILVA, F. S. C. **Bancos de dados: aprenda o que são, melhore seu conhecimento, construa os seus**. São Paulo: Edgard Blücher, 2005.

[14] ELMASRI, R.; NAVATHE, S.B. **Sistemas de banco de dados**. 6. ed. São Paulo: Addison Wesley, 2011.

[15] COSTA. E.R. **Bancos de dados relacionais**. Faculdade de Tecnologia de São Paulo. São Paulo, 2011.

[16] OLIVEIRA, S. S. **Bancos de dados Não-Relacionais: um novo paradigma para armazenamento de dados em sistemas de ensino colaborativo**. Revista da Escola de Administração Pública do Amapá, 2014.

[17] HELLAND, P. ***The Singular Success of SQL***. ACM Queue. Volume 14. 2016. Disponível em: < https://dl.acm.org/doi/10.1145/2956641.2983199#sec-comments >. Acessado em 4 de julho de 2020

[18] LIMA, A G. A. **Padrão SQL e sua evolução**. Artigo. Campinas, UNICAMP, 2004.

[19] MYSQL. **MySQL Worckbench**. Documentation. Disponível em: < https://dev.mysql.com/doc/workbench/en/>. Acessado em 4 de julho de 2020.

[20] MONTANARI, R. **Detecção e classificação de objetos em imagens para rastreamento de veículos**. Dissertação. Instituto de Ciências Matemáticas e de Computação. Universidade de São Paulo. São Carlos. 2015. 77p.

[21] LIU, W., ANGUELOV, D., ERHAN, D., SZEGEDY, C., REED, S., FU, C.Y., BERG, A.C. ***SSD: Single Shot Multibox Detector***. *European conference on computer vision*, 21{37. Springer, 2016.

[22] REDMON, J., DIVVALA, S., GIRSHICK, R., FARHADI, A.: ***You only look once: Unified, real-time object detection***. CVPR. 2016.

[23] REN, S., HE, K., GIRSHICK, R., SUN, J.: ***Faster R-CNN: Towards real-time object detection with region proposal networks***. NIPS. 2015.

[24] SZELISKI, R. ***Computer Vision: Algorithms and Applications***. Springer. 2011.

[25] CULJAK, I. ABRAM, D. PRIBANIC. T, DZAPO. H, CIFREK, M. ***A brief introduction to OpenCV***. in MIPRO. 2012 Proceedings of the 35th International Convention, 2012. pp. 1725-1730.

[26] ROSEBROKE, A. HOFFMAN. D, McDUFFEE, D. THANKI, A. PAUL, S. ***Raspberry Pi for Computer Vision***. Hobbyist Bundle - v1.0.1. Pyimagesearch. 2019.

## APÊNDICE A

**REPÚBLICA FEDERATIVA DO BRASIL**
MINISTÉRIO DA ECONOMIA
**INSTITUTO NACIONAL DA PROPRIEDADE INDUSTRIAL**
DIRETORIA DE PATENTES, PROGRAMAS DE COMPUTADOR E TOPOGRAFIAS DE CIRCUITOS INTEGRADOS

# Certificado de Registro de Programa de Computador

Processo Nº: **BR512020001679-0**

O Instituto Nacional da Propriedade Industrial expede o presente certificado de registro de programa de computador, válido por 50 anos a partir de 1° de janeiro subsequente à data de 15/02/2020, em conformidade com o §2°, art. 2° da Lei 9.609, de 19 de Fevereiro de 1998.

**Título:** URCARONA v.1.0

**Data de criação:** 15/02/2020

**Titular(es):** UNIVERSIDADE ESTADUAL DO MARANHÃO - UEMA

**Autor(es):** CARLOS HENRIQUE RODRIGUES DE OLIVEIRA; ANA PAULA FERREIRA COSTA; LUIZ CARLOS CHAVES LIMA JUNIOR; AURELIANNY ALMEIDA DA CUNHA

**Linguagem:** HTML; JAVA

**Campo de aplicação:** SV-01; TP-01

**Tipo de programa:** AP-01; GI-01; SO-04; SO-05; TC-03

**Algoritmo hash:** SHA-512

**Resumo digital hash:**
74b221a2be6da1329da8256b0fdc5e08498026cbbd6a5fe9c0eb3291366d114bc7114d7b547ebdf8d9bb7c61336dd161bf784dde82259d6f854d8c161fcfae33

**Expedido em:** 25/08/2020

**Aprovado por:**
Helmar Alvares
Chefe da DIPTO - Portaria/INPI/DIRPA Nº 09, de 01 de julho de 2019