|  |  |  |
| --- | --- | --- |
| Description: Description: بازنشسته.jpg | ارتش جمهوری اسلامی ایران (فرماندهی جنگال و سایبر راهبردی آجا ) |  |

عنوان طرح:

طراحی و پیادهسازی سامانه هوشمند نظارت بر تصاویر دوربین های پیرامونی تحت شبکه

نام و نام خانوادگی مجری طرح:

حسین غلامی

استاد راهنما

نام کامل استاد راهنما

استادمشاور

نام کامل استاد مشاور

پاییز 1400

به نام خدا

صفحه تعهد نامه

چكيده

امنیت یکی از مفاهیم اولیه ای است که انسان از ابتدا با آن درگیر بوده و همواره در صدد افزایش آن است. یکی از مواردی که در سال های اخیر برای افزایش امنیت رونق گرفته و پیشرفت کرده، ایجاد امنیت توسط سیستم­های نظارتی و دوربین­ها است. دوربین­ها نمی­توانند به صرف ذخیره اطلاعات امنیت را تامین کنند، برای ایجاد امنیت نیازمند آن هستیم که شخصی بر دوربین ها نظارت دائمی داشته باشد که فرآیندی هزینه­بر است و احتمال خطای انسانی وجود دارد. در این پروژه هدف پیاده­سازی سیستمی است که بتواند به کمک شخص ناظر آمده و فرآیند نظارت کردن بر دوربین­ها را با کمک هوش مصنوعی و پردازش در یک سرویس دهنده تسهیل کند. در این پروژه سه برنامه کاربردی ارائه شده، یک برنامه­کاربردی بر روی رایانه سرویس دهنده ارائه شده که اطلاعات دوربین ها را دریافت و فرآیند مدیریت کاربران را انجام دهد، دومین برنامه کاربردی بر روی رایانه سرویس دهنده دیگری اجرا شده که پردازش­های مربوط به هوش­مصنوعی را انجام خواهد داد و در آخر برنامه کاربردی دیگری برای کاربران سیستم ارائه شده که بتوانند فرآیند دسترسی به اطلاعات دوربین ها را برعهده بگیرند. در این پروژه برای دوربین ها سطح دسترسی تعریف شده که کاربران بر اساس سطح دسترسی خود به دوربین های مربوطه دسترسی پیدا کنند. همچنین اطلاعات دوربین ها در رایانه سرویس دهنده ذخیره شده که کاربران بتوانند به اطلاعات ذخیره شده دسترسی پیدا کنند. برای این امر از سه سرویس RabbitMQ ، MinIO ، Redis استفاده شد. از RabbitMQ برای به جریان انداختن اطلاعات به صورت زمان واقعی، از MinIO برای ذخیره سازی اطلاعات دوربین ها و دسترسی به آن در شبکه، از Redis برای همگام سازی سه برنامه کاربردی با یکدیگر و از PyQt برای پیاده­سازی رابط کاربری برنامه های کاربردی استفاده شد.

واژه‌های کلیدی:

VideoStraming,RabbitMQ,MinIO,Docker,PyQt

|  |  |
| --- | --- |
| فهرست مطالب | صفحه |

[1 فصل اول مقدمه........ 6](#_Toc92866752)

[1-1 پیشگفتار و اهمیت موضوع 7](#_Toc92866753)

[2-1 کار های پیشین انجام شده در این زمینه 9](#_Toc92866754)

[1-2-1 نرم افزار مدیریت دوربین Cisco Meraki 9](#_Toc92866755)

[2-2-1 نرم افزار مدیریت دوربین Rhombus 10](#_Toc92866756)

[3-2-1 نرم افزار مدیریت دوربین Blue Iris Software 11](#_Toc92866757)

[4-2-1 نرم افزار مدیریت ویدیویی EyeLine 12](#_Toc92866758)

[5-2-1 نرم افزار مدیریت دوربین ContaCam Video Management Software 12](#_Toc92866759)

[6-2-1 نرم افزار مدیریت دوربین ZoneMinder 13](#_Toc92866760)

[7-2-1 نرم افزار مدیریت دوربین Ivideon Server 13](#_Toc92866761)

[8-2-1 نرم افزار مدیریت دوربین Luxriot Evo 14](#_Toc92866762)

[9-2-1 جمع بندی 15](#_Toc92866763)

[3-1 طرح مسئله و پیش فرض ها 15](#_Toc92866764)

[1-3-1 استاندارد ONVIF.......................... 15](#_Toc92866765)

[2-3-1 راه حل پیشنهادی............... 20](#_Toc92866766)

[4-1 ساختار پایان نامه 21](#_Toc92866767)

[2 فصل دوم سرویس های مورد استفاده 22](#_Toc92866770)

[1-2 مفهوم یکپارچهسازی 23](#_Toc92866771)

[2-2 بستر اجرای سرویس ها (داکر) 26](#_Toc92866772)

[3-2 داکر کامپوز 29](#_Toc92866773)

[1-3-2 دستورالعمل‌های کامپوزفایل 30](#_Toc92866774)

[4-2 سرویس RabbitMQ 34](#_Toc92866775)

[1-4-2 مراکز توزیع پیام (Exchange ها) 36](#_Toc92866776)

[2-4-2 صف ها در RabbitMQ......... 40](#_Toc92866777)

[3-4-2 تولید کننده و مصرف کننده اطلاعات 41](#_Toc92866778)

[4-4-2 نکاتی تکمیلی در مورد برخی ویژگی ها 42](#_Toc92866779)

[5-4-2 راهاندازی RabbitMQ با داکر 44](#_Toc92866780)

[5-2 سرویس Redis 45](#_Toc92866781)

[1-5-2 انواع ذخیره سازی اطلاعات در Redis 46](#_Toc92866782)

[2-5-2 مکانیزم لیست کنترل دسترسی 47](#_Toc92866783)

[3-5-2 راهاندازی Redis با داکر کامپوز 51](#_Toc92866784)

[6-2 سرویس MinIO 51](#_Toc92866785)

[7-2 جمع بندی 55](#_Toc92866786)

[3 فصل سوم طراحی و معماری برنامه های کاربردی 56](#_Toc92866787)

[1-3 چالش های طراحی معماری 57](#_Toc92866788)

[1-1-3 چالش دریافت اطلاعات از دوربین ها 58](#_Toc92866789)

[2-1-3 چالش انتقال تصویر در زمان واقعی و تعیین سطح دسترسی 59](#_Toc92866790)

[3-1-3 چالش پردازش در زمان واقعی 60](#_Toc92866791)

[4-1-3 چالش ذخیره سازی اطلاعات 61](#_Toc92866792)

[5-1-3 چالش هماهنگ سازی برنامه ها 61](#_Toc92866793)

[2-3 طراحی معماری 62](#_Toc92866794)

[1-2-3 جریان اطلاعات...................... 62](#_Toc92866795)

[2-2-3 دسترسی به اطلاعات ذخیره شده 66](#_Toc92866796)

[3-3 پیاده سازی برنامه های کاربردی 67](#_Toc92866797)

[1-3-3 توسعه Qt با پایتون PyQt5 67](#_Toc92866798)

[2-3-3 توسعه برنامه کاربردی سرور 70](#_Toc92866799)

[3-3-3 توسعه برنامه کاربردی سرور پردازشی 76](#_Toc92866800)

[4-3-3 توسعه برنامه کاربردی کاربر 81](#_Toc92866801)

[5-3-3 جمع بندی............................. 85](#_Toc92866802)

[4 فصل چهارم بکارگیری و نحوه کارکرد سیستم برای کاربران و مدیر سیستم.....................86](#_Toc92866803)

[1-4 راه اندازی و پیشنیاز ها 87](#_Toc92866804)

[2-4 راهنمای استفاده کاربران 87](#_Toc92866805)

[1-2-4 راهنمای راه اندازی برنامه کاربردی سرور 88](#_Toc92866806)

[2-2-4 راهنمای راه اندازی برنامه کاربردی سرور پردازشی 88](#_Toc92866807)

[3-2-4 راهنمای راه اندازی برنامه کاربردی کاربر 88](#_Toc92866808)

[3-4 جمع‌بندی فصل چهارم 88](#_Toc92866809)

[5 فصل پنجم نتیجه‌گیری و پیشنهادها 89](#_Toc92866810)

[1-5 نتیجهگیری 90](#_Toc92866812)

[2-5 پیشنهادها 90](#_Toc92866813)

[6 مراجع. 91](#_Toc92866814)

|  |  |
| --- | --- |
| فهرست اشكال | صفحه |

[شکل ‏1–1- تصویری نمونه از نرم افزار مدیریت دوربین ‏Cisco Meraki 11](#_Toc92866815)

[شکل ‏1–2- تصویری نمونه از نرم افزار مدیریت دوربین ‏Rhombus 12](#_Toc92866816)

[شکل ‏1–3 تصویری نمونه از نرم افزار مدیریت دوربین ‏ Blue Iris Software 12](#_Toc92866817)

[شکل ‏1–4 تصویری نمونه از نرم افزار مدیریت دوربین EyeLine 13](#_Toc92866818)

[شکل ‏1–5 تصویری نمونه از نرم افزار مدریت دوربین ‏ ContaCam Video Management Software 14](#_Toc92866819)

[شکل ‏1–6 تصویری نمونه از نرم افزار مدیریت دوربین ‏ ZoneMinder 15](#_Toc92866820)

[شکل ‏1–7 تصویری نمونه از نرم افزار مدیریت دوربین ‏ Luxriot Evo 15](#_Toc92866821)

[شکل ‏1–8 پیکربندی بسته های RTSP در شبکه 20](#_Toc92866822)

[شکل ‏2–1-مثالی از یکپارچه‌سازی مبتنی بر فایل 24](#_Toc92866823)

[شکل ‏2–2-مثالی از یکپارچهسازی مبتنی بر پایگاهداده 25](#_Toc92866824)

[شکل ‏2–3-نمونه از یکپارچهسازی بر اساس ارتباط مستقیم میان برنامه‌ها 25](#_Toc92866825)

[شکل ‏2–4-نمونه‌ای از یکپارچهسازی با کمک یک کارگزار پیام 26](#_Toc92866826)

[شکل ‏2–5-موتور و هسته داکر که روند فرایند مدیریت برنامه را مشخص میکند. [2] 28](#_Toc92866827)

[شکل ‏2–6- نمایی از معماری داکر و فرایند چرخش دستورات و اطلاعات. [2] 29](#_Toc92866828)

[شکل ‏2–7-نمونه‌ای از یک فایل dockerfile 30](#_Toc92866829)

[شکل ‏2–8-معماری کلی نرم‌افزار RabbitMQ [3] 36](#_Toc92866830)

[شکل ‏2–9- داشبورد مدیریتی نرم‌افزار RabbitMQ 37](#_Toc92866831)

[شکل ‏2–10 38](#_Toc92866832)

[شکل ‏2–11 39](#_Toc92866833)

[شکل ‏2–12 42](#_Toc92866834)

[شکل ‏2–13-نحوه استفاده از کلیدمسیریابی برای صف‌ها 44](#_Toc92866835)

[**شکل ‏2–14-اتصال دو مصرف‌کننده‌ی اطلاعات به یک صف** 44](#_Toc92866836)

[شکل ‏2–15-محتوای داکر فایل برای راه‌اندازی RabbitMQ 45](#_Toc92866837)

[شکل ‏2–16- بخش مرتبط با RabbitMQ در داکر کامپوز 46](#_Toc92866838)

[شکل ‏2–17 50](#_Toc92866839)

[شکل ‏2–18- بخش مرتبط با Redis در داکر کامپوز 52](#_Toc92866840)

[شکل ‏3–1 59](#_Toc92866841)

[شکل ‏3–2 61](#_Toc92866842)

[شکل ‏3–3 64](#_Toc92866843)

[شکل ‏3–4 65](#_Toc92866844)

[شکل ‏3–5 66](#_Toc92866845)

[شکل ‏3–6 69](#_Toc92866846)

[شکل ‏3–7 70](#_Toc92866847)

[شکل ‏3–8 72](#_Toc92866848)

[شکل ‏3–9 77](#_Toc92866849)

[شکل ‏3–10 79](#_Toc92866850)

[شکل ‏3–11 83](#_Toc92866851)

# فصل اول مقدمه

## پیشگفتار و اهمیت موضوع

امروزه امنیت یکی از مهمترین آرمان­های هر کشوري تلقی می­شود و در صورت عدم تامین یا وجود نقص و تنگنا در ایجاد آن، حیات سالم شهروندان به چالش کشیده میشود. [1] طبعاً مکانها با تمام خصوصیات خود و نوع برداشت ذهنی عملکرد انسان در آنها، می­تواند شکل­دهنده به فضایی با نوع امنیت مثبت یا منفی باشد. به بیانی واضح­تر، بعضی فضاها به دلیل خصوصیاتی که دارند، می توانند ظرفیت تولید جرائم بالایی را داشته باشند. همزمان و درعین حال که جرائم در مکانها صورت میگیرند، حاکی از این واقعیت است که، کنترل و نظارت دولتها بر برخی مکانها درحداقل سطح ممکن قرار دارد و بدیهی است که امنیت چه در بعد عینی و چه در بعد ذهنی، در آنها کاهش می­یابد. کوچه­ها و خیابان­هایی که حداقل میزان عبور و مرور را توسط ساکنین، گشت نیروي انتظامی و ... دارند، بدلیل وجود حداقل نظارت، شاهد افزایش انواع جرائم (سرقت، قتل، قاچاق مواد مخدر و ...) می­باشند و به طور کلی فضاهایی با حداقل نظارت را پدید می­آورند. بنابراین، باید از روشها و ابزارهایی استفاده کرد که امنیت اجتماعی مکانهاي مختلف را افزایش دهد و در نتیجه انگیزه مجرمان و خلافکاران در ارتکاب جرم کاهش پیدا کند. یکی از این روشها بکارگیري سیستم نظارت هوشمند در فضاهاي جرم­خیز است. معمولاً در شهرهاي بزرگ به دلیل گستردگی قلمرو و وسعت حوزة استحفاظی، برخی اماکن از تیررس مأموران انتظامی دور می­ماند و یا در زمانی که آنها مشغول پاسبانی از نقاط دیگر هستند، در جایی دیگر به دلیل نبود نظارت، جرم صورت می­گیرد. پس باید یا تعداد افراد بیشتر شود که بنابه دلایلی (بودجه و شرایط سخت استخدام نیرو) این کار تقریباً غیرممکن است و یا باید ناظران الکترونیکی مراقب امنیت محل باشند. بنابراین، براي رسیدن به نظم اجتماعی، برنامه­ریزي امور اجتماعی و ساماندهی عملکردها می­تواند در پیشگیري از جرائم غیرقابل کنترل ثمربخش باشد. امروزه با پیشرفت تکنولوژي و کوچک شدن زمان و کاهش فواصل، شرایط کاهش تخلفات اجتماعی با قانونمند کردن شهروندان در سایه حاکمیت قانون بیش از گذشته فراهم شده است، اما باید دقت کرد که شیوه­هاي برنامه ریزي­هاي اجتماعی توسط مسئولین کشور با توجه به زمان و مکانهاي مختلف در این برنامه­ها حائز اهمیت است. شایان ذکر است، دوربین مداربسته با افزایش خطر دستگیري مجرمان موجب پیشگیري از جرم می­شود. با توجه به اینکه مجرمان در هنگام ارتکاب جرم به خطر دستگیري بیشتر از شدت مجازات فکر می­کنند، چون دوربین مدار بسته این خطر را افزایش می­دهد در پیشگیري از جرم بسیار مؤثر است [2]. استفاده از دوربین های نظارتی همانگونه که برای بکارگیری در محیط های شهری و ایجاد امنیت لازم است برای ارگان های نظامی از اهمیت بیشتری برخوردار است چرا که این محیط ها از اهمیت بیشتری برخوردار هستند و اگر امنیت این مکان ها مورد سوال واقع شود، احساس امنیت در جامعه از بین خواهد رفت. در عامه ضرب المثلی گفته می­شود که "هر چه بگندد نمکش می­زنند، وای به روزی که بگندد نمک" دقیقا همین ضرب المثل گویا همه چیز است. گفتیم که استفاده از دوربین های نظارتی لازم است ولی استفاده از این دوربین ها برای تامین امنیت، زمانی موثر خواهد بود که اشخاصی وجود داشته باشند و بر این دوربین ها نظارت دائمی داشته باشند. در صورتی که این اطلاعات زیر نظر گرفته نشود، از وقوع حادثه جلوگیری نخواهد شد و تنها احساس امنیت کاذب به وجود می­آید. پس نیاز داریم که دوربین ها به صورت تمام وقت بررسی شوند که این کار باید توسط اشخاص انجام شود. و هر جایی که انسان وظیفه بررسی را خود به عهده می­گیرد احتمال خطای انسانی وجود دارد. برای کاهش خطای انسانی می­توان اقداماتی همچون کوتاه کردن شیفت کارمندان اتاق مانیتورینگ یا افزایش تعداد آن ها را انجام داد ولی همچنان احتمال وجود خطا وجود دارد. در ادامه راهی ارائه می­شود که بتوان احتمال وجود خطا را کاهش داد.

در سال های اخیر با پیشرفت هوش مصنوعی و یادگیری ماشین، الگوریتم هایی ارائه شده اند که بتوان با کمک آن­ها فرایندها را ساده تر نمود. (الگوریتم هایی از قبیل تشخیص انسان، تشخیص پلاک، تشخیص چهره و....) شرکت ها در دنیا از این الگوریتم ها استفاده کرده و محصولات هوشمندی را روانه بازار کرده اند. که می­توان به الگوریتم های تشخیص چهره برای باز کردن قفل تلفن های همراه اشاره نمود. از این قبیل الگوریتم ها می­توان در بحث امنیت و سیستم های نظارتی نیز استفاده نمود و آن ها را به اتاق مانیتورینگ آورد و احتمال وجود خطای انسانی را کاهش داد. برای مثال میتوان الگوریتم های تشخیص انسان را برای مناطقی که ورود اشخاص به نواحی ممنوع است به اجرا در آورد تا در صورتی که اشخاص غیرمجاز وارد آن مناطق شدند، هشداری به صدا درآید یا در صورتی که پلاکی غیرمجاز وارد محدوده ای که اجازه ورود به را نداشت وارد شد، هشدار به صدا در آید و ... برای یک محیط مشخص می­توان از الگوریتم های متعددی استفاده کرد که نیازمند یک سیستم یکپارچه برای بررسی و کنترل است، پیاده سازی چنین سیستمی که بتواند هر نیازی که مطرح شود را برآورده کند، کار ساده ای نیست و نیاز است که این چنین سیستمی به صورت بومی پیاده­سازی شود تا نیازهای روزمره و گوناگون ارگان ها و شرکت ها را برآورده کند. در ادامه شرکت هایی که چنین سرویس هایی را در دنیا ارائه می­دهند بررسی می­کنیم.

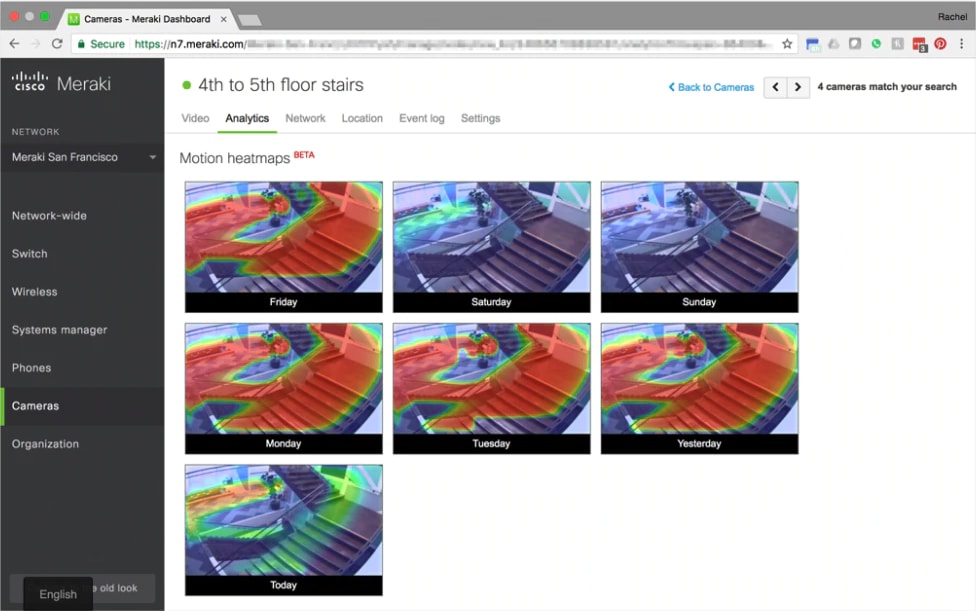
## کار های پیشین انجام شده در این زمینه

انتخاب دوربین های امنیتی تنها گام مهم در اجرای نظارت تصویری نیست، نرم افزارهای مدیریت تصویر در پشت دوربین­ها باید از عملکردهایی که کاربر به دنبال آن است پشتیبانی کند. قبل از اینکه به بررسی نرم­افزارهای مدیریت ویدئویی بپردازیم، درک ارزشِ داشتن یک راه حل یکپارچه برای دریافت اطلاعات همه دوربین­ها بسیار مهم است، زیرا نرم افزاری که قصد دارد اطلاعات تمامی دوربین ها را دریافت کند، می­بایست تحت یک پروتکل استاندارد با دوربین­ها تامل داشته باشد و سرویس هایی از این جنس به کاربران خودش ارائه دهد، جزئیات این مسئله را در بخش های بعدی خواهیم دید ولی اکنون به تعدادی از نرم افزارهای مدیریت تصویر ویدئویی می­پردازیم و هر کدام را به صورت مختصر بررسی می­کنیم.

### نرم افزار مدیریت دوربین Cisco Meraki

Cisco Meraki طیف گسترده ای از راه حل های مرتبط با ویدئو را برای امنیت شما ارائه می دهد اما همه این راه‌حل‌ها یک وجه مشترک دارند، این نرم‌افزار دارای تجزیه و تحلیل ویدئویی قدرتمند با هوش­مصنوعی است و ویژگی های تشخیص حرکت و جستجوی حرکت را پشتیبانی می­کند. به طور مثال این نرم­افزار این امکان را به شما می دهد که به گذشته مناطقی که عبور کرده اید مراجعه کنید و نقطه ی دقیق گم شدن کلید خود بیابید. این­ یکی از ویژگی­ هایی است که قطعاً نمی توانید با یک نرم افزار رایگان مدیریت تصویر دوربین به دست آورد.

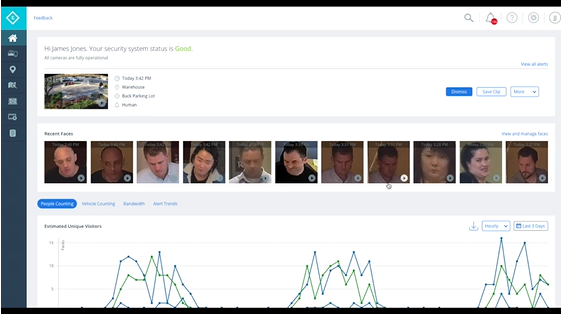
در این نرم افزار همه دوربین های شما به فضای ابری متصل خواهند شد و به هر دوربین بیش از 500 گیگابایت حافظه برای ذخیره سازی اطلاعات تخصیص داده می­شود، این امکان می­تواند باعث صرفه جویی در هزینه ها شود زیرا دیگر نیازی به NVR وجود نخواهد داشت. Meraki همچنین با بسیاری از راه حل ها از جمله برنامه­کاربردی تلفن همراه ادغام شده تا یک راه حل امنیتی جامع برای محل کار شما ایجاد کند که شامل کنترل، دسترسی و نظارت تصویری است. در تصویر شکل ‏1–1 نمونه ای از پنل این نرم افزار قابل مشاهده است که یک نمودار تجمیعی از مکان های مورد تحرک در طول هفته استخراج شده است. همانطور که قابل مشاهده است که در روز های شنبه و یکشنبه میزان تردد از راه پله ها به شدت کاهش یافته است. [3]



شکل ‏1–1- تصویری نمونه از نرم افزار مدیریت دوربین ‏Cisco Meraki

### نرم افزار مدیریت دوربین Rhombus

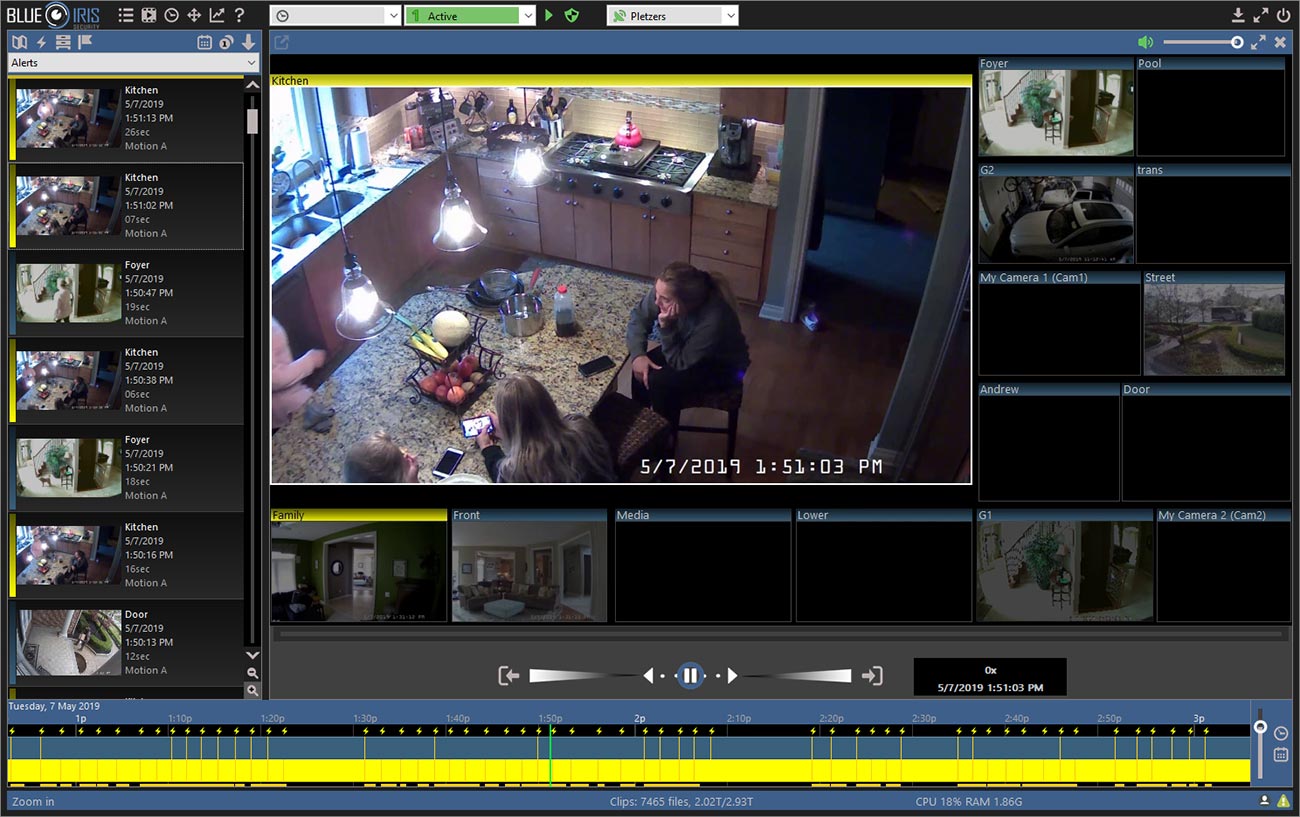
مشابه Meraki، Rhombus یک راه حل امنیتی جامع با دوربین های خود ارائه می دهد. نصب این راه حل ساده تر است و این نرم افزار دوربین­ها را از طریق اینترنت به یکدیگر متصل می کند. Rhombus دارای یک داشبورد بسیار شکیل است که می‌توانید از طریق آن بر آنچه در فضای خود اتفاق می‌افتد نظارت کنید. از لحاظ میزان امکان تجزیه و تحلیل، Rhombus هشدارهای مفیدی را با فیلم های فیلتر شده برای حرکت، چهره ها، رویدادهای حسگر، رفتار غیر معمول و بسیاری موارد دیگر ارائه می دهد. این نرم افزار علاوه بر مزایای ابری (از جمله ذخیره‌سازی ویدیوها)، ویژگی‌های نوآورانه زیادی مانند تشخیص رفتار غیرعادی، تشخیص چهره، و تشخیص پلاک دارد. Rhombus، از مزیت ادغام با چندین راه حل دیگر، از جمله Kisi[[1]](#footnote-1)، برای ایجاد زیرساخت امنیتی کامل تری برخوردار است. در تصویر شکل ‏1–2 چهره هایی که توسط این نرم افزار در طول روز در یک دوربین ذخیره شده اند نمایش داده شده است. [4]



شکل ‏1–2- تصویری نمونه از نرم افزار مدیریت دوربین ‏Rhombus

### نرم افزار مدیریت دوربین Blue Iris Software

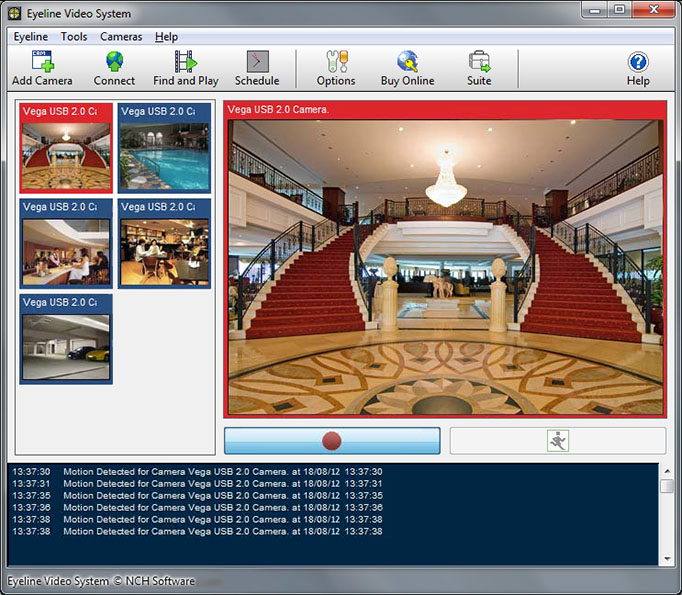
Blue Iris یک سیستم نظارت تصویری مبتنی بر فضای ابری است که اجازه می‌دهد دوربین را از راه دور مشاهده و کنترل کنید. این نرم افزار با مجموعه عظیمی از دوربین های بازار سازگار است، اما به صورت عمومی به این گونه شناخته شده که با دوربین های HD اجرا می­شود. نسخه رایگان Blue Iris نیازی به لایسنس اضافی برای اضافه کردن چندین دوربین را ندارد و به راحتی می توان همه دوربین ها را در یک سیستم واحد دریافت کرد اما برای فعال شدن بخش پردازش و ذخیره سازی اطلاعات در این نرم افزار به لایسنس نیاز است. این نرم افزار حجم زیادی از گزینه های ویرایش و پخش ویدیو را ارائه می­دهد، اما برخی از مشتریان اشاره کرده اند که اغلب شکاف های زمانی و تاخیر وجود دارد. در تصویر شکل ‏1–3 می­توان نمونه رابط کاربری اصلی این نرم افزار را مشاهده نمود. [5]



شکل ‏1–3 تصویری نمونه از نرم افزار مدیریت دوربین ‏ Blue Iris Software

### نرم افزار مدیریت ویدیویی EyeLine

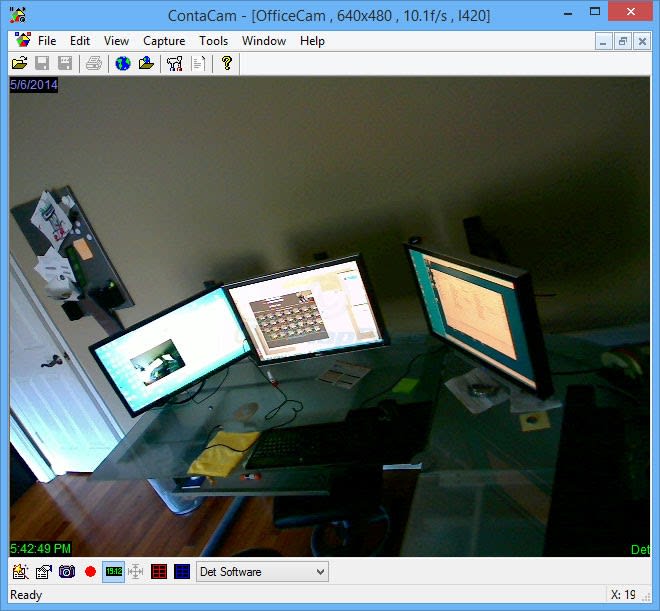
نرم افزار مدیریت ویدئو EyeLine یک گزینه جذاب برای مشتریانی است که به دنبال یک گزینه نرم­افزار مدیریت ویدئو هستند که از دوربین های زیادی پشتیبانی می­کند. EyeLine می تواند بیش از 100 منبع دوربین را به طور همزمان نظارت و ضبط کند. این نرم افزار به شما این امکان را می دهد که فیلم را در زمان واقعی در محل یا آنلاین با ورود از راه دور تماشا کنید. زمانی فیلم ها بر روی فضای ابری ذخیره خواهند شد که مؤلفه تشخیص حرکت در دوربین تحریک شود. در شکل ‏1–4 می­توانیم تصویر نمونه­ی این نرم افزار را مشاهده نماییم. [6]



شکل ‏1–4 تصویری نمونه از نرم افزار مدیریت دوربین EyeLine

### نرم افزار مدیریت دوربین ContaCam Video Management Software

نرم‌افزار مدیریت دوربین ContaCam خود را به‌اندازه کافی سریع، سبک و همه‌کاره می‌داند تا بتواند نیازهای امنیتی کاربران را برآورده کند. در این نرم افزار راه‌اندازی دوربین‌ها بسیار آسان است و از کاربران هنگام تنظیم می‌خواهد که به سادگی بین تشخیص حرکت یا ضبط مداوم فیلم­ها تصمیم بگیرند، نام دوربین را وارد کنند و مدت زمان نگهداری فایل‌ها را پیکربندی کنند. این نرم افزار یک گزینه ایجاد ویدئو خلاصه روزانه را ارائه می دهد که به خوبی با فناوری تشخیص چهره و پلاک ادغام شده است. ContaCam به داشتن عملکرد پایدار شناخته شده است اما گاهی اوقات هنگام جفت شدن با دوربین های IP مشکلاتی دارد. در شکل ‏1–5 می­توانیم این نرم افزار را مشاهده کنیم. [7]



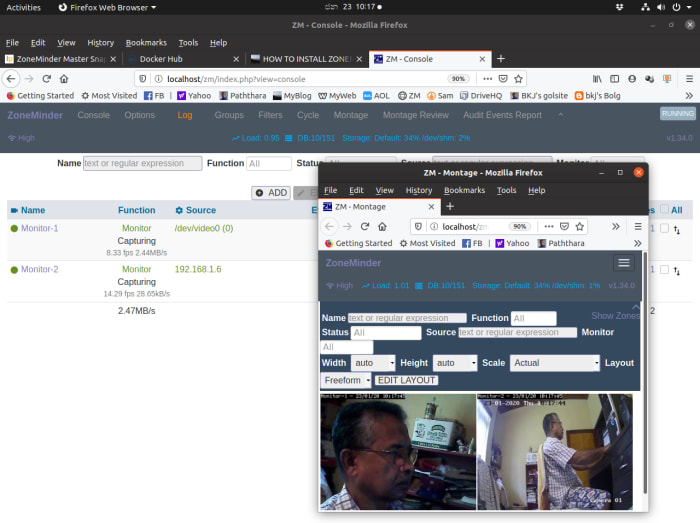
شکل ‏1–5 تصویری نمونه از نرم افزار مدریت دوربین ‏ ContaCam Video Management Software

### نرم افزار مدیریت دوربین ZoneMinder

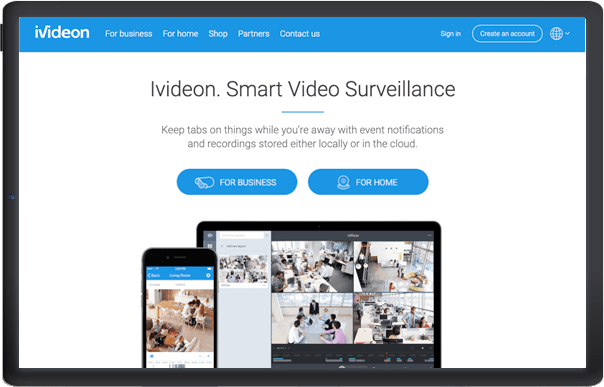
ZoneMinder یک گزینه محبوب برای امنیت خانه و محیط کار است. راه اندازی این نرم افزار بسیار آسان است و با طیف گسترده ای از دوربین های امنیتی کار می­کند. مشتریان پاسخ‌های منفی کمی به این نرم‌افزار دارند، زیرا این نرم‌افزار چیزی بیش از یک نرم‌افزار تجمیع کننده اطلاعات دوربین باشد، یک مجموعه کامل از برنامه‌های پردازش تصویر را در خود دارا است که این برنامه ها درکنار یکدیگر سرویس­دهی می­کنند. ZoneMinder به خوبی طراحی شده است و یافتن سریع گزینه­ای که کاربر به دنبال آن است را برای هر فرد ساده می­کند. در شکل ‏1–6 می­توانیم این نرم افزار را مشاهده کنیم. [8]

### نرم افزار مدیریت دوربین Luxriot Evo

Luxriot Evo یک نرم‌افزار مدیریت ویدئو کاملا رایگان است که بر ارائه نرم‌افزارهای با عملکرد خاص و همچنین خدمات به مشتریان تمرکز دارد. این نرم افزار دارای تعدادی گزینه ویرایش است. این ویرایش­ها شامل ویرایش صدا ویرازش دقت فریم بر ثانیه تصویر ویدیویی و همچنین فیلترهای جلوه های ویژه دیجیتال است. این نرم افزار قابلیت پخش مستقیم ویدیو به YouTube Live را دارا است و کاربران میتوانند اطلاعات خود را مستقیما در youtube به جریان بیاندازد. اگرچه این نرم‌افزار تقریباً با همه ی دوربین­های IP کار می‌کند، اما محدودیتی دارد که نمی‌تواند بیش از 10 کانال را به طور همزمان پشتیبانی کند، بنابراین برای یک دفتر یا تجارت کوچک بهترین است. [9]



شکل ‏1–6 تصویری نمونه از نرم افزار مدیریت دوربین ‏ ZoneMinder



شکل ‏1–7 تصویری نمونه از نرم افزار مدیریت دوربین ‏ Luxriot Evo

### جمع بندی

از میان نرم افزارهای فوق که بررسی شد، می­توان دریافت که نرم­افزارهایی که در این حوزه معرفی شده اکثرا اطلاعات را در سرورهای خود ذخیره و پردازش می­کنند. پر واضح است که نمی­توان اطلاعات امنیتی و نظامی یک کشور را به سرورهای یک شرکت خارجی منتقل نمود. همچنین هیچ یک از نرم­افزار­های فوق امکان پیاده­سازی الگوریتم های هوش مصنوعی شخصی­سازی شده را دربه کاربر ارائه نمی­دهند که بتوان آن را متناسب با نیاز تغییر و استفاده نمود. پس نیازمند ساختن سیستم یکپارچه ای هستیم که توانایی دریافت و تجمیع و ذخیره اطلاعات سرور و اجرای الگوریتم­های هوش مصنوعی دلخواه را بر روی اطلاعات دوربین ها داشته باشد.

## طرح مسئله و پیش فرض ها

در بخش­ پیش نرم افزارهای مدیریت دوربین متفاوتی را بررسی نمودیم و دریافتیم که یکی از چالش­های اصلی در جمع آوری اطلاعات دوربین­ها این است که دوربین­ها توسط شرکت های متفاوتی در سراسر دنیا تولید می­شوند و برای دریافت این اطلاعات، نیاز داریم که از نقطه ای مشخص یا از استانداردی واحد استفاده کنیم تا بتوانیم اطلاعات دوربین­ها را دریافت کنیم. در این بخش ابتدا استاندارد اصلی مشترک که شرکت های سازنده دوربین موظف­اند پیاده سازی کنند را بررسی می­کنیم و سپس به طرح مسئله و راه حل پیشنهادی می­پردازیم. در حقیقت تنها پیش فرض اصلی ما رجوع به نقطه مشترک دوربین­ها یعنی استانداردONVIF است.

### استاندارد ONVIF

هر کسی که با دوربین های امنیتی کار کرده، حتما یک بار اسم استاندارد ONVIF[[2]](#footnote-2) را شنیده است، ONVIF یک انجمن صنعتی است که رابط­های استاندارد، برای همکاری موثر، میان محصولات امنیت فیزیکی مبتنی بر شبکه را فراهم و ارتقا می­بخشد. سنگ بناهای ONVIF عبارتند از:

• استاندارد سازی ارتباط میان محصولات امنیتی فیزیکی مبتنی بر شبکه

• قابلیت تعامل دستگاه های متفاوت بدون توجه به برند تجاری

• در اختیار بودن برای همه شرکت ها و سازمان ها

ONVIF در سال 2008 توسط Axis Communications ، Bosch Security Systems و Sony Corporation تاسیس شد و دارای یک عضو از هر شش قاره است. عضویت در ONVIF برای تولیدکنندگان، توسعه دهندگان نرم افزار، مشاوران، طراحان سیستم، کاربران نهایی و سایر گروه های علاقه­مند که مایل به شرکت در فعالیت های ONVIF هستند ، رایگان است.

امروزه بیش از 18000 محصول سازگار با یکدیگر را بصورت جمعی ارائه شده است. این امر منجر به نصب میلیون ها محصول سازگار ONVIF در بازار شده است. هر محصول باید حداقل یکی از پروفایل­ها راپشتیبانی کند:

• پروفایل A برای پیکربندی کنترل دسترسی­ها

• پروفایل C برای کنترل ورود خروج و مدیریت رویدادها

• پروفایل G برای ذخیره سازی و اطلاعات

• پروفایل Q برای نصب سریع

• پروفایل S برای پخش جریانی اصلی دوربین ها به صورت زنده

• پروفایل T برای پخش جریانی پیشرفته از اطلاعات دوربین ها به صورت زنده

بیشترین حسن این استاندارد برای کاربران نهایی است، چرا که به مصرف کنندگان،با انعطاف بیشتری می­توانند محصول مورد نظر خود را انتخاب کنند. این امر منجر به رقابتی شدن بازار و کاهش قیمت­ها می­شود. این استاندارد به شما امکان انتخاب محصولات متناسب با نیاز خود را می­دهد بدون اینکه به یک برند خاص دوربین محدود شوید. به عنوان مثال، یک نرم افزار مدیریت فیلم سازگار با این استاندارد، به شما این امکان را می­دهد تا دستگاه های سازگار را از سازندگان مختلف سخت افزار تهیه و استفاده کنید.

از آن جایی که در این پروژه قصد داریم اطلاعات دوربین ها را به طور زنده دریافت کنیم نیاز داریم که دوربین ها از حداقل یکی دو پروفایل S یا T استاندارد ONVIF پشتیبانی کند. در ادامه ویژگی های این دو پروفایل را معرفی خواهند شد.

**Profile-S : برای پخش جریان اصلی دوربین ها به صورت زنده**

پروفایل s برای سیستم­های ویدیوئی مبتنی بر ip طراحی شده است. یک دستگاه Profile S (به عنوان مثال دوربین شبکه IP ) دستگاهی است که می­تواند داده­های ویدئویی را از طریق شبکه به یک سرویس گیرنده ارسال کند. سرویس گیرنده (به عنوان مثال ، یک نرم افزار مدیریت ویدیو است) برنامه­ای است که می تواند پخش ویدئو را از طریق یک شبکه از طریق پیکربندی ، درخواست و کنترل کند. این پروفایل از ویژگی هایی همچون کنترل PTZ، کنترل صدا، پخش چندمنظوره و انجام عملیات کنترلی(کنترل رله) برای دستگاه­های سازگارپشتیبانی می­کنند. [10]

**Profile-T : برای پخش جریانی پیشرفته از اطلاعات دوربین ها به صورت زنده**

پروفایلT نیز برای سیستم های ویدیویی مبتنی برIP طراحی شده است. پروفایلT از ویژگی های پخش جریانی ویدئو مانند استفاده از فشرده سازی هایH.264 و H.265 ، تنظیمات تصویربرداری و رویدادهایی همانند تشخیص حرکت و کنترل نور محیط پشتیبانی می­کند. ویژگی­های اجباری برای این دستگاه­ها شامل پیاده­سازی وب سرور داخلی برای کنترل تنظیمات دوربین است. این پروفایل علاوه بر مواردی که در پروفایلS ذکر شد، باید از پروتکل HLS نیز پشتیبانی کند و امکان رمزنگاری کردن اطلاعات دوربین­را فراهم کند. [11]

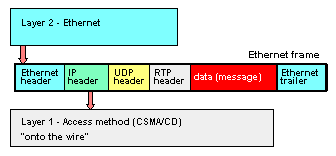
همانطور که مشاهده می­شود دروبین هایی با پروفایل T از امکانات بیشتری نسبت به دوربین هایی با پروفایل S برخوردار هستند و می­توانند اطلاعات خود را رمزنگاری شده و با کیفت بالاتری ارائه دهند وهمچنین امکان ایجاد ارتباط دو سویه در این پروفایل فراهم شده، ولی دلیل اصلی که نام این دو پروفایل را پخش زنده جریان نامیده اند، امکان پیاده­سازی پروتکل RSTP است که با کمک آن می­توان جریان اطلاعات دوربین ها را مدریت کرد. در ادامه بیشتر با این پروتکل آشنا می­شویم.

**پروتکل RTSP**

پروتکل جریان بخشی در زمان واقعی (RTSP[[3]](#footnote-3)) یک سیستم ارتباطی شبکه در لایه برنامه کاربردی است که با برقراری ارتباط مستقیم با سرویس دهنده، جریان داده های چندرسانه ای را به دستگاه دیگری منتقل می کند.این پروتکل جریان چندرسانه­ای بین دستگاه­های سرویس دهنده و سرویس گیرنده را با استفاده از یک کنترل از راه دور شبکه برای همگام سازی جریانی، رسانه های مداوم مانند صدا و تصویر ، ایجاد و کنترل می­کند. این پروتکل خود پیام­های چندرسانه­ای را منتقل نمی­کند بلکه با سرویس دهنده ای که اطلاعات چندرسانه­ای(مانند تصویر دوربین) را پخش می­کند ارتباط برقرار می­کند. برای مثال وقتی کاربر، ویدیویی را که در حال پخش است قطع می­کند، RTSP درخواست کاربر را برای مکث فیلم به سرور پخش جریانی ویدئو منتقل می­کند، و پخش ویدیو از سرویس دهنده متوقف می­شود.

وقتی کاربر یا برنامه ای تلاش می کند ویدئو را از یک منبع از راه دور پخش کند، دستگاه سرویس گیرنده درخواست را سرور ارسال می­کند تا گزینه های موجود، مانند مکث ، پخش و ضبط را تعیین کند. پس سرویس دهنده لیستی از انواع درخواست هایی را که می تواند از طریق بپذیرد، برمی­گرداند. هنگامی که سرویس گیرنده می­داند چگونه درخواستش را ارسال کند، یک درخواست توصیف رسانه را به سرویس­دهنده پخش جریانی منتقل می­کند و سرویس دهنده با توصیف رسانه پاسخ می دهد و هنگامی که سرویس گیرنده درخواست راه اندازی را ارسال می­کند و سرور با اطلاعات مربوط به مکانیزم حمل و نقل بسته ها پاسخ می دهد. پس از اتمام مراحل نصب ، سرویس گیرنده با ارسال مکانیزم انتقال مشخص شده، درخواست راه اندازی، جریان پخش را با گفتن اینکه سرور برای ارسال شروع می­شود آغاز می­کند. این پروتکل به عنوان راهی برای اجازه دادن به کاربران برای پخش صوتی و تصویری مستقیم بر بستر شبکه و نه بارگیری فایل های چندرسانه­ای در دستگاه است. این پروتکل همچنین امکان انعطاف پذیری زیادی را فراهم می­کند تا جایی که سرویس گیرنده­ها می توانند ویژگی هایی را که می­خواهند استفاده کنند را درخواست کنندو دریابند که آیا سرویس دهنده از چنین امکانی پشتیبانی می­کند.

همانطور که پیشتر گفته شد RSTP یک پروتکل لایه کاربردی است که برای انتقال جریان تصویر در شبکه استفاده می شود. این پروتکل نرم افزار سرویس گیرنده کاربر را قادر می سازد تا با کنترل از راه دور سرویس دهنده اطلاعات را با عملکردهایی مانند مکث و پخش کنترل کند. برای کنترل این پروتکل جلسه[[4]](#footnote-4) RTP در لایه برنامه برنامه کاربردی استفاده می شود. در شکل زیر نمونه ای از بسته مربوط به پروتکل حمل و نقل بیدرنگ (RTP) قابل مشاهده است، این پروتکل یک پروتکل بر بستر شبکه است که از انتقال صدا و فیلم در زمان واقعی پشتیبانی می کند. یک بسته RTSP بر روی یک ارتباط UDP ، سوار می شود و شامل اطلاعات زمان سنجی و هماهنگ سازی در هدر آن برای مونتاژ مناسب در انتهای دریافت می باشد. [12]



شکل ‏1–8 پیکربندی بسته های RTSP در شبکه

### راه حل پیشنهادی

با توجه به پیش فرض فوق که دوبین ها از پروتکل RTSP پشتیبانی کنند، می­توان برنامه ای نوشت که اطلاعات را از دوربین­ها با پروتکل فوق دریافت کند، همچنین با دریافت اطلاعات دوربین­ها می­توان آن­ها را ذخیره نمود و آن ها را در شبکه به کاربران ارسال کرد. پس برای پیاده­سازی یک معماری حداقل باید یک برنامه بر روی کامپیوتر میزبان وجود داشته باشد که اطلاعات را از دوربین ها دریافت و پردازش کند و یک برنامه در سمت کاربر اجرا شود که اطلاعات پردازش­شده دوربین ها را از کامپیوتر میزبان دریافت کند. اما راهکار برای پردازش چیست، اولین و کارآمدترین راهکاری که به ذهن می­آید این است که اطلاعات دریافت شده از دوربین ها علاوه بر کاربر، به یک برنامه کاربردی دیگر ارسال شود و فرآیند پردازش را انجام دهد و کاربر تصاویر اصلی دوبین ها و اطلاعات پردازش شده تصاویر را در کنار هم دریافت کند. پس به سه برنامه کاربردی نیاز داریم که به شرح زیر است:

* برنامه کاربردی سرور: برنامه ای که اطلاعات دوربین ها را دریافت و ذخیره کرده و به کاربر و برنامه سرور پردازشی ارسال می­کند.
* برنامه کاربردی سرور پردازشی: برنامه ای که اطلاعات دوربین ها را دریافت، پردازش کرده و به کاربر ارسال میکند.
* برنامه کاربر: برنامه ای که اطلاعات خام و پردازش شده دوربین ها را دریافت به نمایش می­گذارد و همچنین به اطلاعاتی که برنامه سرور ذخیره کرده ، دسترسی پیدا میکند و آن ها را به نمایش می­گذارد.

از آن جایی که این برنامه قرار است در یک محیط امنیتی اجرا شود و کاربران نباید به اطلاعات تصاویر هر نقطه ای دست رسی داشته باشند، این برنامه به یک ویژگی اساسی نیاز دارد و آن کنترل بندی مناطق و تعیین سطح دسترسی برای کاربران است که مدیر سیستم بتواند با کنترل و تعیین مناطق و تعیین سطح دسترسی کاربران امنیت مورد رضایت را تامین نماید. چنین سیستمی نیازمند پیاده­سازی یک معماری منسجم است که بتوان عملکردهای فوق را در شبکه و با تامین امنیت کافی فراهم نماید. هدف اصلی این پروژه ارائه این معماری برای توسعه الگوریتم های هوش­مصنوعی است که از مدل­ها و الگوریتم های هوش مصنوعی که امروزه با سرعت زیادی در حال پیشرفت و توسعه هستند بتوان استفاده نمود و بتوان سیستم هایی هوشمند که قابلیت پیاده­سازی سناریوهای دلخواه را داشته باشند طراحی نمود.

## ساختار پایان نامه

در این رساله، که در پنج فصل تدوین شده، در ابتدای هر فصل ابتدا مروری بر مفاهیم کلی، نرم‌افزارهای به‌کاررفته، انجام شده است و در ادامه جزئیات در بخش­ها و طراحی آن را شرح داده خواهد شد.در فصل دوم سرویس های مورد استفاده در این پروژه، نحوه بکارگیری آن ها و ویژگی هایی که در اختیار ما میگذارند تشریح می­شود.در فصل سوم ابتدا به طراحی معماری انجام می­شود و در ادامه هر یک برنامه کاربردی سرور، سرور پردازشی و برنامه کاربردی کاربر، پرداخته می­شود و در فصل چهارم به نحوه بکارگیری و استفاده از سیستم فوق در شبکه تشریح می­شود. در فصل پنجم نیز به نتیجه­گیری و پیشنهادهایی که در ادامه راه این پروژه می­تواند انجام شود، پرداخته می­­شود.



# فصل دوم سرویس های مورد استفاده

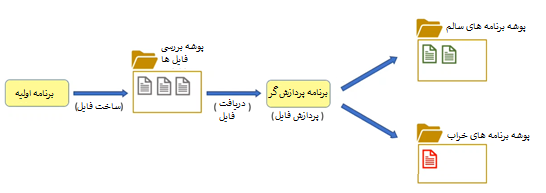
**پیاده­سازی سیستم دریافت و ذخیره­سازی اطلاعات**

در این بخش ابتدا به معرفی بستر اجرای سرویس های مورد استفاده در این پروژه می­پردازیم و سعی می­کنیم با ابزارهای یکپارچه‌سازی بر بستر شبکه بتوانیم، شرایطی برای دریافت و انتقال و ذخیره‌سازی اطلاعات فراهم کنیم.

## مفهوم یکپارچه­سازی

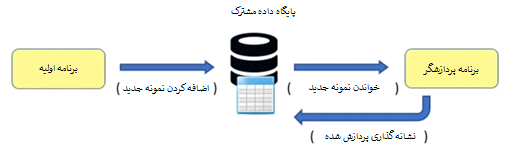
یک سیستم کامل، از چند برنامه تشکیل‌شده که وظایف متفاوتی را انجام می­دهند، در اغلب مواقع این برنامه‌ها نیاز دارند با یکدیگر تعامل داشته باشند و باهم ارتباط برقرار کنند. به ایجاد کردن ارتباط میان برنامه‌های متفاوت یکپارچه‌سازی[[5]](#footnote-5) می­گویند. شیوه‌های متفاوتی از یکپارچه­سازی وجود دارد.

مدل اول یکپارچه­سازی مبتنی بر فایل است. در این مدل، برنامه اولیه فایلی که باید پردازش شود را ایجاد می­کند و در پوشه‌ای مشخص قرار می­دهد، سپس برنامه‌ای دیگر فایل‌هایی که در این پوشه قرارگرفته‌اند را بررسی می­کند. برای مثال برنامه دوم و فایل­های خراب را از سالم جدا کرده و در پوشه‌هایی متفاوت قرار می­دهد. قابل‌مشاهده است که این دو برنامه می­توانند مستقل از هم کار کنند و حتی می­توانند با زبان‌های برنامه‌نویسی متفاوتی ایجادشده باشند. در شکل ‏2–1می­توان مثال فوق را مشاهده کرد.[[6]](#footnote-6)



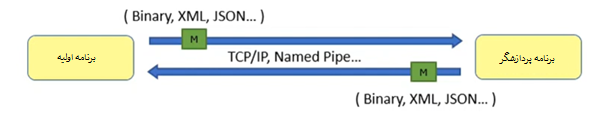
شکل ‏2–1-مثالی از یکپارچه‌سازی مبتنی بر فایل

مدل دوم یکپارچه­سازی­هایی مبتنی بر پایگاه­داده مشترک یا توزیع‌شده است. در این مدل ابتدا یکی از برنامه­ها اطلاعاتی را در پایگاه­داده تغییر می­دهد. سپس برنامه دیگری آن را برداشته و پردازش می­کند و مجدداً به پایگاه­داده برمی­گرداند و به آن برچسب پردازش‌شده می­زند. به‌طور مشابه، این دو برنامه می­توانند مستقل از یکدیگر فعالیت کنند. در شکل شکل ‏2–2 می­توان مدل فوق را بررسی نمود.[[7]](#footnote-7)



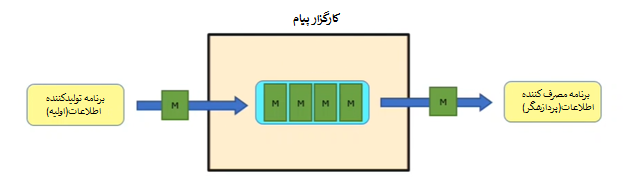
شکل ‏2–2-مثالی از یکپارچه­سازی مبتنی بر پایگاه­داده

مدل سوم یکپارچه­سازی­هایی مبتنی بر ارتباط مستقیم است، به‌نحوی‌که برنامه‌ها به‌صورت مستقیم با یکدیگر ارتباط tcp/ip برقرار کرده و بعد از برقراری ارتباط شروع به ارسال پیام به یکدیگر می­کنند. فرمت پیام‌ها می­تواند به گونه‌های متفاوتی ازجمله باینری یا مبتنی برنوشته مانند, xml, json ... باشد. در شکل ‏2–3 می­توان نمونه‌ای از این ارتباط را مشاهده کرد.[[8]](#footnote-8)



شکل ‏2–3-نمونه از یکپارچه­سازی بر اساس ارتباط مستقیم میان برنامه‌ها

مدل چهارم یکپارچه­سازی بر پایه پیام دادن نامتقارن با کمک یک کارگزار پیام[[9]](#footnote-9) است. برنامه اولیه یا تولیدکننده اطلاعات در یک‌سو با هر فرمتی که بخواهد می­تواند، پیام ارسال کند ولی این بار یک برنامه میانی پیام‌ها را دریافت می­کند (که اغلب به آن message broker یا message bus می­گویند) پیام‌ها را در یک لیست قرار داده که به آن Queue یا صف می­گویند و آن‌ها را به هدف، مقصد یا مصرف‌کننده منتقل می­کند. در شکل ‏2–4 می­توان نمونه‌ای از این شیوه انتقال اطلاعات را مشاهده کرد. این روش، یک روش محبوب است که در سیستم­های متفاوتی می­تواند مورداستفاده قرار بگیرد.



شکل ‏2–4-نمونه‌ای از یکپارچه­سازی با کمک یک کارگزار پیام

این شیوه از یکپارچه‌سازی مزایای بسیاری دارد که به شرح زیر است:

* جدا بودن برنامه تولیدکننده از برنامه مصرف‌کننده:
  + برنامه تولیدکننده اطلاعات و برنامه مصرف‌کننده نیازی ندارد یکدیگر را بشناسند.
  + آدرس و تکنولوژی‌های مورداستفاده یکدیگر را نمی­دانند.
  + تنها چیزی که نیاز است نسبت به هم آگاه باشند، فرمت اطلاعات ارسالی و دریافتی است.
* ایجاد محیطی مطمئن، قابل‌اعتماد برای انجام پردازش‌ها و ارتباط میان برنامه­ها:
  + برنامه‌های تولیدکننده اطلاعات می­تواند به ارسال اطلاعات بپردازد، درحالی‌که هیچ برنامه‌ای در طرف دیگر، اطلاعات را مورداستفاده قرار نمی­دهد، به این معنا که اطلاعات در برنامه میانی به‌صورت موقت ذخیره می­شوند و در ادامه هرگاه که برنامه مصرف‌کننده اطلاعات، فعالیت خود را آغاز کرد، می­تواند از اطلاعات استفاده نماید.
  + درصورتی‌که برنامه مصرف‌کننده در پردازش شکست بخورد، می­تواند درخواست ارسال مجدد داشته باشد.
* فراهم کردن راهی برای پیاده­سازی معماری­هایی افقی در تعداد بالا
  + اگر برنامه مصرف‌کننده اطلاعات به‌تنهایی نتواند همه پیام‌ها را پردازش کند، می­توان چندین مصرف‌کننده اطلاعات را در کنار هم قرارداد تا فرایند سریع­تر انجام شود.

این شیوه بسیار پربازده‌تر از مدل‌های یکپارچه­سازی مبتنی بر پایگاه­داده و عموما برای جریان­داده مورد استفاده قرار میگیرد[[10]](#footnote-10).

## بستر اجرای سرویس ها (داکر)

به‌طورکلی داکر[[11]](#footnote-11) یک سکو برای توسعه، انتقال و اجرای برنامه‌ها است. با کمک داکر می­توانیم نرم­افزار­ها را مستقل از زیرساخت به‌سرعت پیاده‌سازی کنیم. همچنین با کمک این نرم‌افزار می­توان زیرساخت و سخت‌افزار مورداستفاده در پروژه‌ها را مدیریت کرد و فاصله زمانی میان پیاده‌سازی و رسیدن به مرحله محصول نهایی را به حداقل رساند.

داکر توانایی بسته‌بندی و اجرای برنامه­های ما را در محیط ایزوله‌ای به نام کانتینر[[12]](#footnote-12) فراهم می­کند. این ایزوله شدن شرایطی ایجاد می­کند که کانتینرها را بتوان به‌صورت هم‌زمان روی کامپیوتر میزبان اجرا کرد. این کانتینرها در مقابل ماشین‌های مجازی بسیار سبک هستند و می­توانند بدون نظارت بر روی کامپیوتر میزبان اجرا شوند. به‌طورکلی داکر ابزار و سکویی برای مدیریت کردن این کانتینرها است تا توسعه سریع و ساده­تر شود و بتوان پردازش قابل توسعه­ای، روی خوشه‌ای از کامپیوترها اجرا کرد. همچنین هنگامی‌که برنامه کامل شد و به مرحله محصول نهایی رسید، بدون در نظر گرفتن نوع زیرساخت، قابل‌انتقال به محیط‌های دیگر (مرکز داده‌ها، شرکت‌های خدمات دهنده ابری،...) باشد.

به‌طورکلی هسته اصلی داکر از سه بخش تشکیل‌شده که در شکل ‏2–5 قابل‌مشاهده است، بخش اول یک سرویس‌دهنده است که مدیریت اشیا ساخته‌شده توسط داکر را بر عهده دارد، اشیائی همچون فایل ایمیج (فشرده‌شده کانتینرها)، کانتینرها، شبکه‌ها و فضاهایی که تخصیص داده می­شوند. این برنامه فرایند محاسباتی[[13]](#footnote-13) نام‌گذاری می­شود. بخش دوم یک برنامه رابط است (REST API) که دستوراتی که ما می­دهیم را به فرایند محاسباتی منتقل می­کند. بخش سوم یک رابط و خط دستور (CLI)[[14]](#footnote-14) است که فرمان‌های ما در آن وارد می­شود. خط دستور از یک REST API استفاده می­کند تا وظیفه کنترل و ارتباط با فرایند محاسباتی را یا با استفاده از فرمان­هایی از پیش نوشته‌شده یا به‌صورت واردکردن در خط دستور انجام دهد.



شکل ‏2–5-موتور و هسته داکر که روند فرایند مدیریت برنامه را مشخص می­کند. [2]

به سه دلیل اصلی در این پروژه برای پیاده‌سازی سرویس ها از داکر استفاده شد که به شرح زیر است:

پیشرفت و توسعه مداوم برنامه‌ها؛

داکر با کار در محیط‌های استاندارد و با استفاده از کانتینرهای محلی این اجازه را به توسعه‌دهندگان برای پیاده­سازی برنامه‌های پیچیده می­دهد و چرخه عمر توسعه را ساده می­کند. چراکه بخش‌های مختلف برنامه می­توانند مستقل از هم با نسخه‌ها و حتی زبان­های برنامه‌نویسی متفاوت پیاده­سازی و اجرا شوند. به این عمل فرایند یکپارچه‌سازی پیوسته[[15]](#footnote-15)/ تحویل پیوسته[[16]](#footnote-16) می­گویند.

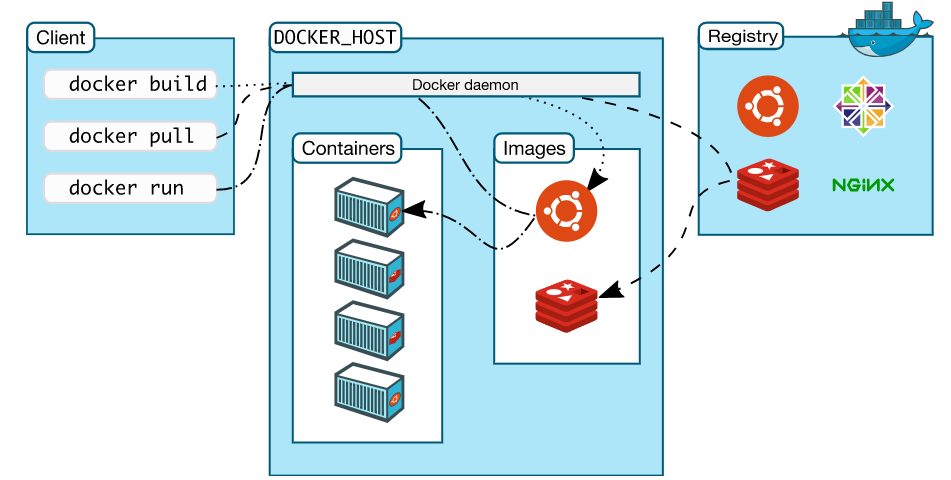
بستر مبتنی بر کانتینرهای داکر، ایجاد برنامه‌هایی قابل‌حمل[[17]](#footnote-17) را امکان‌پذیر می‌کند. کانتینرهای داکر می‌توانند بر روی کامپیوتر محلی توسعه‌دهنده، در ماشین‌های فیزیکی یا مجازی در یک مرکز داده، بر روی ارائه‌دهندگان خدمات ابری یا در مخلوطی از این محیط‌ها اجرا شوند که کمک می­کند برنامه به‌سادگی روی کامپیوتر توسعه‌دهنده آماده و سپس برای پیاده‌سازی عملی به سرورهایی باقدرت و پاسخ‌گویی بیشتر منتقل شود.

کانتینرهای داکر سبک و سریع هستند چراکه تنها برنامه‌هایی که موردنیاز است را در خود جای‌داده‌اند. این‌یک جایگزین مناسب و مقرون‌به‌صرفه به‌جای ماشین‌های مجازی است که نیازمند پیش‌نیازهای بیشتری برای پیاده‌سازی محیط‌های ایزوله هستند؛ بنابراین می­توان از ظرفیت محاسبه بیشتری برای رسیدن به اهداف خود استفاده کرد. داکر برای برنامه‌هایی با چگالی بالا (بخش‌های متفاوت) و عملکردهای کوچک و متوسط که نیاز به انجام کارهای بیشتر با منابع کمتری دارند، بسیار مناسب است.

داکر از یک معماری کاربر-سرویس­دهنده­ای استفاده می­کند. به این شکل که کاربر داکر، با فرایند محاسباتی در ارتباط است که وظیفه ساخت و اجرا و توزیع کانتینرها را بر عهده دارد. کاربر داکر و فرایند اصلی می­توانند در یک کامپیوتر باشند یا می­توانند از طریق REST API یا رابط­های شبکه با یکدیگر در ارتباط باشند. به‌طورکلی برنامه‌ها در محیطی به نام کانتینر اجرا می­شوند که نمونه‌ای قابل‌اجرا از فایلی به نام ایمیج است. برای ساخت فایل ایمیج یا باید خودمان آن فایل را از ابتدا ایجاد کنیم، یا از ایمیج های استانداری که در بخش رجیستری داکر (داکرهاب) قرارگرفته استفاده کنیم. عموماً برای سبک بودن محیط اجرای کار حتی درصورتی‌که بخواهیم فایل ایمیجی را خودمان طراحی کنیم از یک ایمیج مبتنی بر لینوکس ساده شروع کرده و هر چیزی که به آن نیاز داریم را به آن اضافه می­کنیم، ولی در اکثر مواقع برنامه‌های موردنیاز در داکرهاب وجود دارند که می­توان از آن بهره­برد.

برای مثال در شکل شکل ‏2–6 با دستور اول یک فایل ایمیج ساخته می­شود. با دستور دوم یک فایل ایمیج از بخش رجیستری (داکرهاب) به کامپیوتر میزبان منتقل می­شود و با دستور سوم، یکی از فایل­های ایمیج ساخته‌شده به‌صورت کانتینر درآمده و اجرا می­شود.

به این شکل کانتینرها ایجاد می­شوند. حال برای ارتباط کانتینرها با یکدیگر از مکانیزم های دیگری که پیش­تر در شکل ‏2–7 دیدیم استفاده می­شود. مکانیزم هایی همچون ایجاد شبکه یا اجازه دسترسی به حجم‌های فیزیکی که ارتباط میان کانتینرها را فراهم کند.



شکل ‏2–6- نمایی از معماری داکر و فرایند چرخش دستورات و اطلاعات. [2]

COPY requirements.txt ./

RUN pip install --no-cache-dir -r requirements.txt

COPY main.py ./

COPY util.py ./

CMD [ "python", "./main.py" ]

شکل ‏2–7-نمونه‌ای از یک فایل dockerfile

## داکر کامپوز[[18]](#footnote-18)

همان‌طور که در شکل ‏2–6 قابل‌مشاهده شد، برای اجرای هر مرحله می­بایست دستوری را اجرا کنیم تا مراحل پیش­رود و برنامه اجرا شود برای خودکار کردن این کار را می­توان از داکر کامپوز استفاده نمود. داکر کامپوز ابزاری است که می‌توان با استفاده از آن چند تا کانتینر را راه‌اندازی کرد و تمام موارد مربوط به راه‌اندازی آنها را در آن لحاظ کرد. داکر کامپوز با استفاده از یک YAML فایل تمام موارد مربوط به راه‌اندازی سرویس ما را دریافت و بر اساس آن سرویس که می‌تواند شامل چندین کانتینر باشد را با یک دستور راه‌اندازی می‌کند. از داکر کامپوز می‌توان در تمام محیط‌های کاری اعم از production, staging, development, testing و محیط‌های CI/CD استفاده کرد.

روابط بین کانتینرها را مشخص کرد. داکر کامپوز یک ابزار برای تعریف و اجرای برنامه‌های حاوی چند کانتینر است. این برنامه از یک فایل YAML برای پیکربندی این مجموعه استفاده می­کند. سپس با اجرای تنها یک دستور همه‌ی سرویس­های خود را فعال و آماده‌به‌کار می­کند. ساخت برنامه‌ها با داکر کامپوز از سه بخش تشکیل‌شده است.

برخی از مزایای استفاده از داکرکامپوز:

* با استفاده از آن می‌توان بر روی یک هاست چند تا محیط ایزوله با یکدیگر داشت.
* در زمان ایجاد تغییر کانتینر‌ها را مجدد ایجاد نماید.
* با استفاده از متغیرهای که تعریف می‌شود می‌توان سرویس‌ را در محیط‌های مختلف پیاده‌سازی کرد.
* اطلاعات کانتینرها‌ را حسب نیاز بر روی volumeهایی که ایجاد می‌کند قرار دهد.
* داکر کامپوز می‌تواند به خوبی با سرویس‌های CI/CD ادغام شود و تمام موارد مورد نیاز را به صورت خودکار انجام دهد.

معمولا این سناریو در راه‌اندازی سرویس‌ها انجام می‌شود. ابتدا داکرفایل برای ایجاد ایمیج‌ها آماده می‌شود. )که پیشتر در مورد ایجاد داکرفایل صبحت کردیم( سپس با توجه به نیاز موجود برای راه‌اندازی سرویس و ارتباطات و عملکرد آنها با هم یک کامپوز فایل آماده می‌گردد. سپس با استفاده از دستورات داکرکامپوز و با توجه به کامپوز فایل سرویس با تمام شرایط مد نظر را‌ه‌اندازی و در اختیار قرار می‌گیرد. می‌توان برای کل سرویس و یا هر جزئی‌ از آن نیز تست نوشت و تمام موارد را تست کرد. حال در ادامه مروری خیلی کوتاه اما مفید بر نگارش کامپوز فایل خواهیم داشت.

### دستورالعمل‌های کامپوزفایل

در این بخش در مورد دستور‌العمل‌های کامپوز فایل صحبت می‌کنیم. شایان ذکر است که نسخه‌ی ۳ کامپوزفایل بررسی خواهد شد که در زمان نگارش این داکیومنت جدیدترین نسخه‌ی آن می‌باشد. تنها دستورالعمل‌های services, networks و volumes به صورت مسیر پیش‌فرض در کامپوزفایل می‌باشند. همانطور که می‌دانید yaml به مکان دستورالعمل‌ها حساس می‌باشد از این رو در زمان ایجاد yaml file می‌بایست به موقعیت مکانی تمام دستورالعمل‌ها دقت کرد زیرا اگر در جای درستی قرار نگرفته باشند خطا داده و فایل شما اجرا نخواهد شد. در ضمن پسوند فایل می‌تواند yml. یا yaml. باشد و هر دو آن درست می‌باشد.

زمانی که یک کامپوزفایل اجرا می‌شود همانند این است که با دستورات مختلف ایمیج دریافت و یا ساخته شود و یا اینکه کانتینر با استفاده از تنظیماتی که قرار داده شده است ایجاد شود و یا اینکه شبکه با کانفیگ مشخص ایجاد شود. در واقع تمام آپشن‌ها و دستورالعمل‌هایی که در کامپوزفایل مورد استفاده قرار می‌گیرد همان دستورات داکر است که در حال استفاده می‌باشد.

* دستورالعمل ‌build: با استفاده از این دستورالعمل می‌توان تنظیمات مربوط به ساخت ایمیج در زمان اجرای سرویس را داد. می‌توان مسیر و اسم داکرفایل را مشخص کرد و آپشن‌های زمان build همانند arg را در آن قرار داد و یا اینکه ایمیج با چه نامی ساخته شود. دستورالعمل‌های مهم زیر مجموعه‌ی build عبارتند از:

دستورالعمل context: مسیری که داکرفایل در آنجا قرار دارد و یا آدرس ریپوی git آن را مشخص می‌کند.

دستورالعمل dockerfile: داکرفایل اگر با نام Dockerfile داخل مسیری که اشاره شد وجود ندارد و نام دیگری دارد با این آپشن مشخص می‌کند.

دستورالعمل args: دقیقا همان arg داخل داکرفایل است که می‌توان در زمان ساخت ایمیج آن را قرار داد.

دستورالعمل labels: در زمان ساخت ایمیج به آن لیبل داده می‌شود.

دستورالعمل command: دستورالعمل پیش‌فرض کانتینر را جایگذاری می‌کند.

* دستورالعمل ‌container\_name: اسم کانتینر را مشخص می‌کند. اگر اسم برای کانتینر انتخاب نشود ترکیبی از اسم سرویس به همراه دایرکتوری آن قرار می‌دهد.
* دستورالعمل ‌depends\_on: این دستورالعمل بسیار کاربردی می‌باشد و اگر بین سرویس‌های داخل کامپوز فایل اولویت زمانی در راه‌اندازی اهمیت داشته باشد با استفاده از این دستورالعمل می‌توان سرویسی را به سرویس دیگری وابسته کرد. در تصویر زیر برای اینکه سرویس web راه‌اندازی شود نیاز است تا ابتدا سرویس db و سرویس redis راه‌اندازی شود. تازمانی که هر دو سرویس راه‌اندازی نشوند سرویس web راه‌اندازی نخواهد شد. نمونه ای از اجرای این دستور عمل در شکل ‏2–8 قابل مشاهده است.

version: '3'

services:

  web:

    build: .

    depends\_on:

      - db

      - redis

  redis:

    image: redis

  db:

    image: posgres

شکل ‏2–8-نمونه استفاده از دستور عملdepends\_on

* دستورالعمل restart\_policy: تنظیمات مربوط به restart کانتینر می‌باشد. این تنظیمات مربوط به restart را کنترل می‌کند. در مثال زیر زمانی که موضوعی باعث fail کانتینر شود با ۵ ثانیه تاخیر ۳ بار تلاش می‌کند تا کانتینر را اجرا کند. نمونه از اجرای این دستورالعمل را میتوان در شکل ‏2–9مشاهده کرد.

version: '3'

services:

  redis:

    image: redis:alpine

    deploy:

      restart\_policy:

        condition: on-failure

        delay: 5s

        max\_attempt: 3

        window : 120s

شکل ‏2–9- نمونه استفاده از دستور العمل restart\_policy

* دستورالعمل ‌env\_file: اضافه کردن متغیرهای محلی با استفاده از فایل که می‌تواند شامل یک متغیر و یا لیستی از آنها باشد. به مثال‌های زیر توجه کنید. در شکل ‏2–10 میتوانیم نمونه ای از استفاده از این دستورالعمل را مشاهده کنیم.

env\_file: .env

env\_file:

  - ./common.env

  - ./apps/web.env

شکل ‏2–10- نمونه ای از استفاده از دستور العمل env\_file

* دستورالعمل ‌environment: از این دستورالعمل برای زمانی که بخواهیم یک یا چند متغیر را در کانتینر مقداردهی کنیم استفاده می‌کنیم. تفاوت در این است که تمام متغیرها داخل خود کامپوزفایل تعریف خواهد شد و فایل دیگه‌ای لود نخواهد شد. به هر دو صورت زیر از آن استفاده می‌شود. در شکل ‏2–11میوانیم نمونه ای از آن را مشاهده کنیم.

environment:

  - RACK\_ENV=development

  - SHOW=true

  - SESSION\_SECRET

شکل ‏2–11- نمونه ای از استفاده از دستورالعمل environment

* دستورالعمل ‌image: ایمیج مورد استفاده را مشخص می‌کند و کانتینر از روی این ایمیج آماده می‌شود. البته می‌توان با استفاده از دستورالعمل build ایمیج مورد استفاده را از روی داکرفایل نیز ایجاد کرد. در داکر فایل های پیشین این مورد را دیدیم.
* دستورالعمل ports: که با استفاده از آن می‌توان پورت‌ها رو از داخل کانتیر به هاست publish کرد. به صورت اختصار به صورت HOST:CONTAINER استفاده می‌شود. برای مثال پورت ۸۰ کانتیر به پورت ۸۰۸۰ هاست مپ می‌شود. می‌توان تنها پورت کانتیر را قرار داد که آنگاه سرویس داکر آن را بر روی یکی از رندم پورت‌های هاست مپ می‌کند. و می‌توان به صورت اختصاصی مشخص کرد که این پورت بر روی کدام پورت و ip سروس هاست bind شود و یا اینکه در چه پروتکلی قرار داشته باشد. در مثال‌های زیر تمام موارد آورده شده است. در شکل ‏2–12 میتوانیم نمونه ای از استفاده این دستورالعمل را مشاهد کنیم.

ports:

 - "3000"

 - "3000-3005"

 - "8000:8000"

 - "9090-9091:8080-8081"

 - "49100:22"

 - "127.0.0.1:8001:8001"

 - "127.0.0.1:5000-5010:5000-5010"

شکل ‏2–12-نمونه ها متفاوت از استفاده از دستورالعمل ports

* دستورالعمل ‌volumes: به دو روش یکی قرار دادن بر روی مسیری از دایرکتوری هاست و دیگری ایجاد والیوم با استفاده از درایورهای مختلف می‌توان داده‌های کانتینر را بر روی هاست ذخیره کرد. در استفاده از والیوم نیز می‌توان به صورت مختصر یعنی HOST:CONTAINER استفاده کرد یا به صورت طولانی‌تر تمام هر قسمت را توضیح داد. در مثال زیر چند نمونه از استفاده‌ی مختصر آن آورده شده است. در شکل ‏2–13 می­توانیم نمونه هایی متفاوت از استفاده از این دستورالعمل را مشاهده کنیم

volumes:

  # Just specify a path and let the Engine create a volume

    - /var/lib/mysql

  # Specify an absolute path mapping

    - /opt/data:/var/lib/mysql

  # Path on the host, relative to the Compose file

    - ./cache:/tmp/cache

  # User-relative path

    - ~/configs:/etc/configs/:ro

  # Named volume

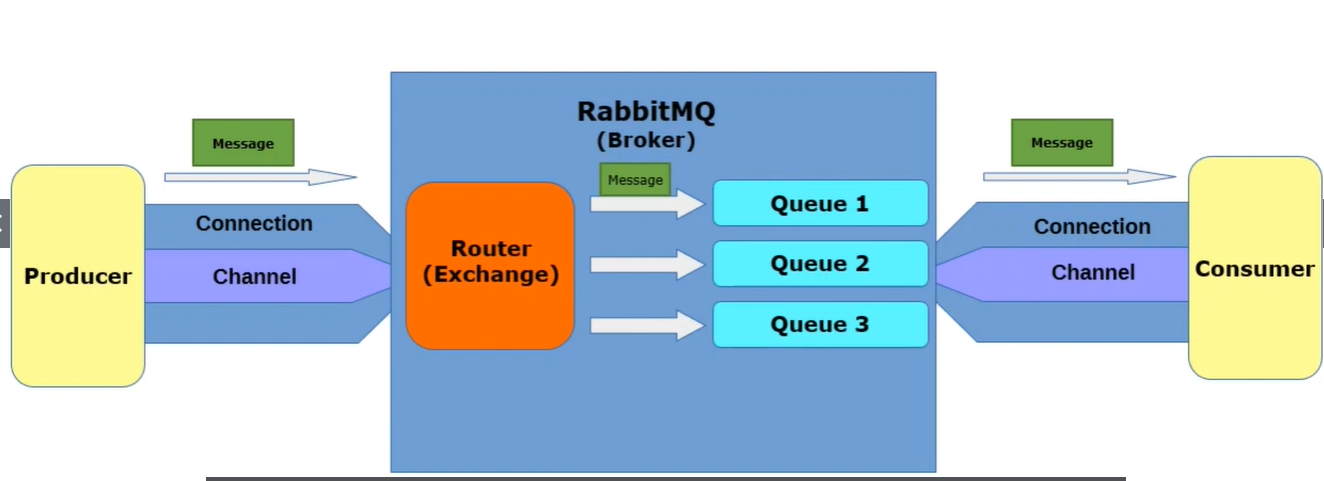
   - datavolume:/var/lib/mysql

شکل ‏2–13- نمونه های متفاوت از بکارگیری دستورالعمل volumes

## سرویس RabbitMQ

RabbitMq یک کارگزار پیام است که می­توان با کمک آن سیستم‌های بزرگی را یکپارچه­سازی کرد. به این شکل که پیام را ابتدا از تولیدکنندگان اطلاعات دریافت می­کند و به مصرف‌کنندگان اطلاعات منتقل می­کند. این نرم‌افزار متن‌باز و با زبان برنامه‌نویسی Erlang نوشته‌شده است و از پروتکل­های بسیاری ازجمله AMQP، STOMP، MQTT، HTTP و WebSocket پشتیبانی می­کند. به همین جهت از این نرم‌افزار می­توان به‌عنوان پلی میان سخت‌افزار و محیط ابری یادکرد. این نرم‌افزار بر روی ویندوز، لینوکس و مک قابل‌استفاده است و همچنین به‌صورت کانتینر داکر در رجیستری داکر (داکرهاب) قرارگرفته است که استفاده از آن را بسیار ساده می­کند.

در شکل ‏2–14معماری کلی این نرم‌افزار قابل‌مشاهده است، تولیدکنندگان اطلاعات[[19]](#footnote-19) و مصرف‌کنندگان اطلاعات[[20]](#footnote-20) در دو سمت این معماری و این نرم‌افزار در مرکز شکل قرارگرفته‌اند. آن‌ها به این نرم‌افزار متصل شده و به تبادل اطلاعات می­پردازند. در این معماری اجزای دیگری ازجمله صف[[21]](#footnote-21)، مرکز توزیع[[22]](#footnote-22) و کانال‌ها[[23]](#footnote-23) قابل‌مشاهده هستند.

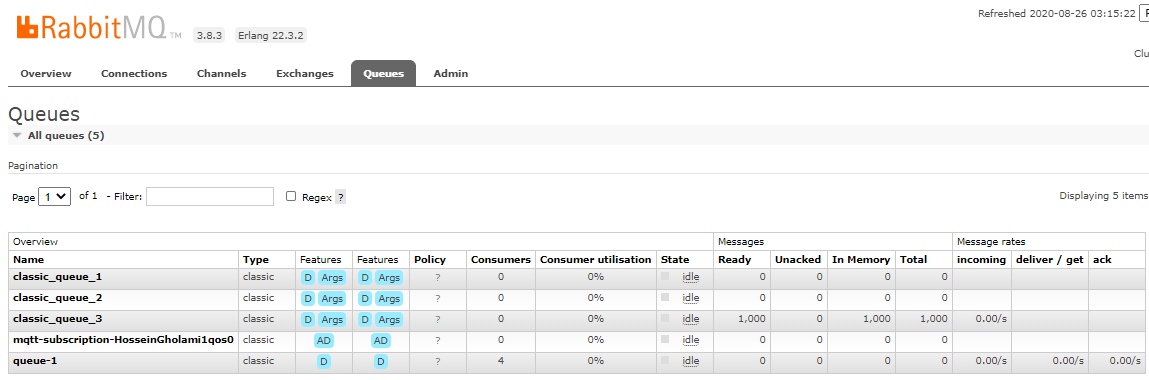


شکل ‏2–14-معماری کلی نرم‌افزار RabbitMQ [3]

که در ادامه با بررسی یک مثال به شرح وظایف هرکدام از این اجزا می­پردازیم. و در ادامه هر یک از بخش ها را به طور کامل شرح می­دهیم.

برنامه هایی را در نظر بگیرید که در هر ثانیه میخواهند یک پیام را به برنامه ای دیگر در شبکه منتقل کند، که ترتیب انتقال و عدم تاخیر و عدم حذف شدن اطلاعات در انتقال اطلاعات برای آن الویت دارد. پس واضح است که پیام ها باید در یک نقطه جمع شده و سپس به صف مربوطه خود منتقل شوند و برنامه دیگر اطلاعات را از انتها صف بردارد. با یک مثال ساده تر عملکرد کلی سیستم را شرح میدهم. کلینیک یک بیمارستان را در نظر بگیرید که تعدادی پزشک از رشته های متفاوت دارد، و اشخاص قصد دارند به این بیمارستان مراجعه کرده و هر کدام پیش پزشک مد نظر خود بروند. همه افراد ابتدا به پذیرش کلینیک مراجعه کرده و پذیرش پزشک مخصوص خود را میگیرند. سپس هر کدام به صف مربوطه پزشک خود میروند و هرگاه پزشک بخواهد می­تواند افرادی که در صف هستند را ویزیت کند. در این مثال تولید کننده اطلاعات، بیمارانی هستند که بیمارستان مراجعه میکنند. کانال ورودی ، بیمارستان است. کانال خروجی اتاق دکتر است. مرکز توزیع همان پذیرش بیمارستان است که افراد را به صف انتظار دکترها هدایت می­کند. و اتاق انتظار پزشک صف ها هستند.

RabbitMQ یک پنل داشبورد تحت وب برای بررسی شرایط صف‌ها، تبادل داده‌ها و کانال‌ها و ارتباطات می­دهد که در آن می­توان اطلاعات مفیدی هنگام به‌کارگیری در عمل به دست آورد. در شکل ‏2–15 صفحه مربوط به‌صف­های را می­توانید مشاهده کنید، برای مثال در این تصویر 4 مصرف‌کننده اطلاعات بر روی صف queue-1 قرارگرفته شده تا در صورت انتقال اطلاعات به این صف، پیام‌ها دریافت شوند. در صف classic\_queue\_3 هزار پیام در حافظه قرارگرفته و هیچ مصرف‌کننده اطلاعاتی به این صف متصل نیست.



شکل ‏2–15- داشبورد مدیریتی نرم‌افزار RabbitMQ

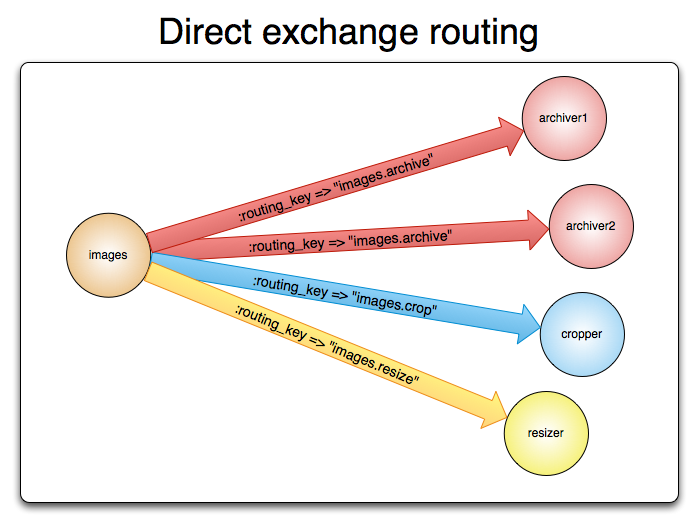
با توجه به توضیحات فوق و بررسی وظایف کلی هر بخش، به بررسی دقیق­تر هر یک از اجزا میرویم.

### مراکز توزیع پیام (Exchange ها)

مراکز توزیع پیام موجودیت هایی هستند که پیام ها به آنها ارسال می­شود. مراکز توزیع پیام یک پیام را دریافت می­کنند و آن را به صف ها یا هیچ هدایت می­کنند. الگوریتم مسیریابی مورد استفاده بسته به نوع مراکز توزیع پیام و قوانینی به نام اتصالات[[24]](#footnote-24) استفاده می­شود. مراکز توزیع پیام از 4 نوع کلی تشکیل شده اند، مرکز توزیع مستقیم[[25]](#footnote-25)، مرکز توزیع خروج به همه[[26]](#footnote-26)، مرکز توزیع موضوعی[[27]](#footnote-27) و مرکز توزیع سری[[28]](#footnote-28) علاوه بر موارد فوق برای تعریف هر مرکز توزیع باید یک نام منحصر به فرد، پایداری[[29]](#footnote-29)(این که بعد از ریست شدن سرویس آیا این مرکز توزیع وجود داشته باشد یا حذف شود.) حذف خودکار(اگر صف های متصل شده به این سرویس اتصالشان قطع شود، مرکز توزیع حذف شود) و سایر پارامترهایی که میتوان برای کنترل بهتر داد(این موارد اختیاری هستند و شامل تعیین پارامترهایی برای کنترل آن در میزبان هایی دیگر تعریف میشوند.) در ادامه به انواع متفاوت این مراکز توزیع پرداخته میشود.

**مرکز توزیع مستقیم**

مرکز توزیع مستقیم پیام ها را بر اساس کلیدمسیریابی که در آن تعیین شده به صف های منتخب متصل به آن ارسال می­کند. این مراکز توزیع برای ارسال های تک به تک[[30]](#footnote-30) مناسب هستند.(اگر چه می­توان از آنها در ارسال های سراسری[[31]](#footnote-31) نیز استفاده نمود.) در شکل ‏2–16 می­توان نمونه ای از استفاده از این مرکز توزیع پیام را مشاهده کرد.



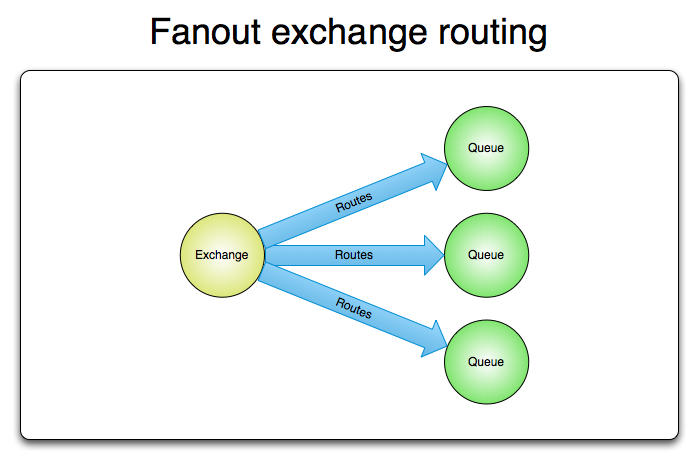
شکل ‏2–16-نمونه مرکز توزیع مستقیم

در این مدل صف های زیادی به مرکز توزیع پیام با کلید مسیریابی منحصر به فرد خودشان متصل شده اند و وقتی یک پیام جدید می­آید بر اساس کلیدمسیریابی که در آن قرار گرفته است به صف های مورد نظر خود ارسال می­شود. برای مثال در تصویر فوق مرکز توزیع image وجود دارد و صف های archive1 و archive2 و ... به این مرکز اتصال با کلید های مسیریابی image.archive1 و image.archive2 و... متصل شده اند. حال اگر یک پیام جدید بیاید، که کلید مسیریابی image.archive1 داشته باشد تنها به صف archive1 منتقل خواهد شد. این در حالی است که اگر بسته ای با کلید مسیریابی salam به این مرکز توزیع ارسال شود به هیچ یک از صف های فوق نخواهد رفت و پیام حذف خواهد شد.

مراکز توزیع مستقیم اغلب برای توزیع وظایف بین چندین برنامه (نمونه هایی از یک برنامه کاربردی) به صورت چرخشی استفاده می شود. هنگام انجام این کار، درک این نکته مهم است که پیام ها بین مصرف کنندگان و نه بین صف ها متعادل می شوند.

**مراکز توزیع خروج به همه**

در مراکز توزیع خروج به همه پیام‌ها را به تمام صف‌هایی که به آن متصل هستند، مستقل از کلید مسیریابی هدایت ‌خواهند شد و اگر N صف به یک مرکز توزیع خروج به همه متصل شده باشد شده باشد، زمانی که یک پیام جدید برای آن مبادله منتشر می شود، یک کپی از آن پیام به تمام N صف تحویل داده می شود. تبادلات Fanout برای مسیریابی پخش پیام ها ایده آل هستند، چرا که پیام را به تمامی صف ها ارسال میکنند. برای مثال; بازی‌های آنلاین چند نفره (MMO) می‌توانند از این ویژگی برای به‌روزرسانی‌های تابلوی امتیازات خود استفاده کنند. یا سایت‌های خبری ورزشی می‌توانند از مراکز توزیع خروج به همه برای کاربران خود در زمان واقعی استفاده کنند. یا سیستم های توزیع شده می توانند برای روز رسانی های وضعیت های مختلف و پیکربندی از این ویژگی استفاده کنند. چت های گروهی می­توانند پیام ها را بین کاربران خود با استفاده از این مرکز توزیع، توزیع کنند. در شکل ‏2–17 نمونه از این این مدل را می­توان مشاهده کرد.



شکل ‏2–17-نمونه مراکز توزیع خروج به همه

**مراکز توزیع موضوعی**

مراکز توزیع موضوعی، یک پیام را به یک یا چند صف بر اساس تطابق بین یک کلید مسیریابی پیام و نوع اتصال صف بین یک یا چند صف مبادله می‌کند. نوع تبادل موضوع اغلب برای اجرای انواع الگوهای انتشار/اشتراک[[32]](#footnote-32) استفاده می‌شود. تبادل موضوع معمولا برای مسیریابی چندپخشی پیام ها استفاده می شود. مبادلات موضوعی دارای مجموعه بسیار گسترده ای از موارد استفاده هستند. هر زمان که مشکلی شامل چندین مصرف کننده/برنامه کاربردی می شود که به طور انتخابی نوع پیام هایی را که می خواهند دریافت کنند انتخاب می کنند، استفاده از مبادلات موضوعی باید در نظر گرفته شود. برخی از کاربردهای آن شامل، توزیع داده های مربوط به موقعیت جغرافیایی خاص، به عنوان مثال، نقاط فروش. پردازش کار پس‌زمینه توسط چندین کارگر انجام ‌شود که هر کدام قادر به انجام مجموعه‌ای از وظایف هستند. به روز رسانی قیمت سهام. به‌روزرسانی‌های خبری که شامل دسته‌بندی یا برچسب‌گذاری است (مثلاً فقط برای یک ورزش یا تیم خاص). ساختارها یا بسته‌بندی‌های نرم‌افزاری خاص معماری/سیستم‌عامل که در آن هر سازنده می‌تواند تنها یک معماری یا سیستم‌عامل را مدیریت کند.

**مراکز توزیع سری**

مراکز توزیع سری برای مسیریابی در چندین ویژگی طراحی شده است که به راحتی به عنوان سر پیام نسبت به یک کلید مسیریابی بیان می شوند. مبادلات سری ویژگی کلید مسیریابی را نادیده می گیرند. در عوض، ویژگی های مورد استفاده برای مسیریابی از ویژگی headers گرفته شده است. اگر مقدار هدر با مقدار مشخص شده در هنگام اتصال برابر باشد، پیامی مطابق در نظر گرفته می شود و به صف مربوطه ارسال خواهد شد.

می­توان یک صف را با استفاده از بیش از یک سر برای تطبیق به یک تبادل سری متصل کرد. در این مورد، کارگزار به یک اطلاعات بیشتر از توسعه‌دهنده برنامه نیاز دارد، یعنی آیا باید پیام‌هایی را با هر یک از هدرها یا همه آنها در نظر بگیرد؟ این همان چیزی است که آرگومان الزام آور "x-match" برای آن است. زمانی که آرگومان "x-match" روی "any" تنظیم شود، فقط یک مقدار هدر منطبق کافی است. از طرف دیگر، تنظیم "x-match" بر روی "all" الزام می کند که همه مقادیر باید مطابقت داشته باشند. این مراکز توزیع همانند مراکز توزیع مستقیم هستند و تنها تفاوت آن در نحوه ارسال بسته ها است که نوع تصمیم گیری به جای کلیدمسیریابی به سر بسته می­سپارد.

### صف ها در RabbitMQ

صف‌ها در RabbitMQ بسیار شبیه به صف‌ها در سایر سیستم‌های صف پیام و وظیفه[[33]](#footnote-33) هستند: آنها پیام‌هایی را ذخیره می‌کنند که توسط برنامه‌ها مصرف کننده اطلاعات، مصرف می‌شوند. صف ها برخی از ویژگی ها را با مراکز توزیع به اشتراک می گذارند، اما برخی ویژگی های اضافی نیز دارند که شامل موارد زیر هستند: نام منحصر به فرد، پایداری(اگر سرویس ریست شد، صف ها حذف نشوند.) انحصاری (فقط توسط یک مصرف کننده اطلاعات مصرف شوند) حذف خودکار (صفی که حداقل یک مصرف کننده اطلاعات داشته است، با لغو اشتراک آخرین مصرف کننده حذف می شود) آرگومان‌ها (اختیاری؛ استفاده شده توسط افزونه‌ها و ویژگی‌های خاص کارگزار مانند زمان نگهداری هر پیام، محدودیت طول صف و ...)

قبل از اینکه بتوان از یک صف استفاده کرد، باید آن را تعریف کرد. تعریف یک صف باعث ایجاد آن می شود اگر قبلا وجود نداشته باشد. اگر صف از قبل وجود داشته باشد و نام آن مانند آنچه در لیست صف های موجود تعریف شده باشد، تعریف اثری نخواهد داشت و با خطای 406 مواجه خواهیم شد.

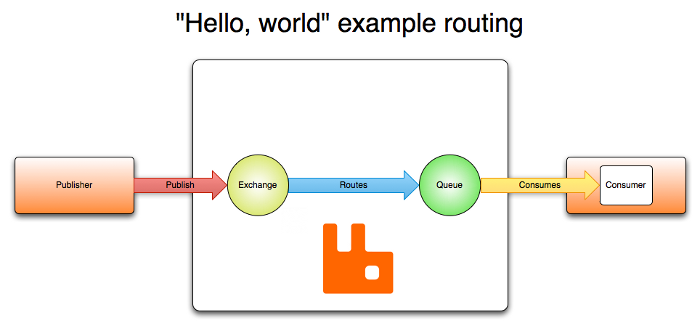
برنامه‌ها ممکن است نام‌های صف را انتخاب کنند یا از کارگزار بخواهند نامی برای آنها ایجاد کند. نام صف ممکن است تا 255 بایت نویسه UTF-8 باشد. ربیت می تواند یک نام صف منحصر به فرد از طرف برنامه ایجاد کند. برای استفاده از این ویژگی، یک رشته خالی را به عنوان آرگومان نام صف ارسال کنید.

صف ها را می توان به عنوان پایدار یا گذرا تعریف کرد. ربیت یک صف پایدار را روی دیسک ذخیره میکند این در حالی که اگر یک صف گذرا تعریف شود این اطلاعات در صورت امکان در حافظه ذخیره خواهند شد. که منجر به افزایش سرعت انتقال می­شود.

همانطور که پیشتر گفته شد، اتصالات، قوانینی هستند که مراکز توزیع بر اساس آن تصمیم میگیرند که یک بسته به کدام صف ارسال شود. برای دستور دادن به مرکز توزیع E برای مسیریابی پیام‌ها به صف Q، Q باید به E متصل شود. اتصالات ممکن است یک ویژگی کلید مسیریابی اختیاری داشته باشد که توسط برخی از انواع تبادل استفاده می‌شود. هدف کلید مسیریابی انتخاب پیام‌های خاصی است که به یک مرکز توزیع منتشر میکند تا به صف محدود شده مورد نظر خود هدایت شود. به عبارت دیگر، کلید مسیریابی مانند یک فیلتر عمل می کند. مثال زیر را در نظر بگیرید؛ فرض کنید که قصد سفر با هواپیما دارید و مقصد شما اصفهان است. مرکز توزیع مانند فرودگاه است. که میتوانید به فرودگاه بروید و اگر بلیت درستی نداشته باشید به مقصد نخواهید رسید ولی اگر بلیت درست(کلیدمسیریابی) صحیح داشته باشید به هواپیما منتثل خواهید شد(به صف منتقل خواهید شد) و با پرواز هواپیما به مقصد خواهید رسید(برنامه ای محتوا را از صف بردارد).

### تولید کننده و مصرف کننده اطلاعات

دیدیم که RabbitMQ کنترل میان تولید کننده اطلاعات و مصرف کننده اطلاعات را فراهم میکند. مصرف کننده اطلاعات به صف متصل شده و در صورتی که پیامی داخل آن ها شود، آن پیام ها را دریافت می­کند. در عین حال تولید کننده اطلاعات پیام های خود را آماده میکند و با کلید مسیریابی مشخص به مراکز توزیع ارسال می­کند. این دو برنامه به طور کامل از یکدیگر مستقل هستند و می­توان هر کدام از آن ها را حتی با یک زبان هایی برنامه نویسی متفاوت نوشت که هر کدام در کامپیوتر هایی که در شبکه قرار گرفته اند به هم متصل شوند. عموما پروتکل ارتباطی با rabbitmq، AMQP است



شکل ‏2–18-نمونه یک ارتباط در RabbitMQ

AMQP (پروتکل صف پیشرفته پیام) یک پروتکل پیام رسانی است که برنامه های کاربردی کلاینت منطبق را قادر می سازد تا با کارگزاران میان افزار پیام رسان منطبق ارتباط برقرار کنند. AMQP یک پروتکل قابل برنامه ریزی است به این معنا که موجودیت های AMQP و طرح های مسیریابی در درجه اول توسط خود برنامه ها تعریف می شوند، نه توسط برنامه های ارسال کننده و دریافت کننده اطلاعات. بر این اساس، برای عملیاتی که صف‌ها و مبادلات را اعلام می‌کند، پیوندهای بین آنها را تعریف می شود، مراکز توزیع ور صف‌ها پیش بینی می‌شود. این به توسعه دهندگان برنامه آزادی زیادی می دهد.

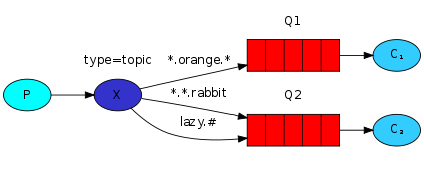
### نکاتی تکمیلی در مورد برخی ویژگی ها

هنگامی‌که هر صف تعریف می­شود، در خود بخشی با عنوان «کلید­ مسیریابی[[34]](#footnote-34)» دارد و باید به یک مرکز توزیع پیام متصل شود. صف‌ها با کمک کلید انتقال بسته‌هایی که وارد، یک مرکز انتقال می­شوند را می­توانند در خود جای دهند. برای مثال stock.usd.ny و quick.orange.rabit می­توانند هر دو نام تاپیک برای بسته‌ها باشند و صف‌هایی با کلید مسیریابی فوق می­توانند، این پیام‌ها را دریافت کنند. دراین‌بین برای RabbitMQ از سه کاراکتر ‘.’ و ‘#’ و’\*’ برای مدیریت کردن بهتر بسته‌ها هنگام تقسیم بسته‌ها میان صف‌ها استفاده می­کند.

کاراکتر ‘.’ به معنی ایجاد ساختار است و هنگامی‌که در کلید مسیریابی استفاده شود باید دقیقا تاپیک پیام و کلیدمسیریابی یکسان باشد. کاراکتر ‘\*’ به این معنی است که در بخش‌هایی که این کاراکتر قرارگرفته می­تواند تاپیک بسته با کلیدمسیریابی متفاوت باشد و کاراکتر ‘#’ به این معناست، درصورتی‌که تا قبل از این کاراکتر یکسان بود، بسته به صف منتقل شود. مثال زیر را در نظر بگیرید:

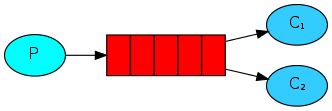
بسته‌ای با تاپیک quick.orange.rabbit به یک مرکز انتقال می­رسد و چهار صف با کلیدهای مسیریابی زیر به این مرکز انتقال متصل هستند.

حالت اول کلیدمسیریابی quick.orange است، این پیام دریافت نمی­شود، چراکه تعداد بخش‌های ایجادشده با ‘.’ یکسان نیست. حالت دوم کلیدمسیریابی \*.orange.\* است، این پیام دریافت می­شود، چراکه تعداد بخش‌های ایجادشده با نقطه یکسان و از دو بخش اول و آخر بسته صرف‌نظر می­شود. حالت سوم، کلیدمسیریابی quick.# است، این پیام دریافت می­شود چراکه تاپیک بسته با هر آنچه که قبل از ‘#’ است، یکسان می­باشد. این مکانیزم می­تواند برای دسته‌بندی پیام‌های ورودی بسیار مؤثر باشد. در شکل ‏2–19 می­توان مثال دیگری را مشاهده نمود.



شکل ‏2–19-نحوه استفاده از کلیدمسیریابی برای صف‌ها

در این ساختار همان‌طور که پیش­تر بررسی شد، می­توانیم از چندین مصرفت کننده اطلاعات به‌طور هم‌زمان استفاده کنیم تا بتوانیم حجم بسته‌های داخل صف را مدیریت کنیم. در این سیستم یک پارامتری تحت عنوان پیش­دستی[[35]](#footnote-35) تعریف می­شود، به این معنا که تعداد بسته‌های درخواستی از صف را کنترل می­کند که بسته‌ها به‌صورت چندتایی ارسال شوند. در این حالت، بسته‌ها به حالت گذرا تغییر وضعیت یافته تا هنگامی‌که سرویس­دهی این بسته‌ها تمام شد، یک تعداد بسته جدید تقاضا شود. این امر موجب می­شود تعداد پیام‌هایی که برای ارسال و دریافت بین سرور صف و سرویس‌دهنده منتقل می­شود، کاهش یابد و درنتیجه بازدهی بالاتر رود. برای مثال به شکل ‏2–20 توجه کنید.



**شکل ‏2–20-اتصال دو مصرف‌کننده‌ی اطلاعات به یک صف**

فرض کنید در این شکل دو مصرف‌کننده‌ی اطلاعات، به این صف متصل شده‌اند و در صف 50 بسته وجود دارد. C1 با پیش­دستی 5 به این صف متصل شده و C2 که کامپیوتری با امکانات سخت‌افزاری قوی­تری است، با پیش­دستی 10 به این صف متصل می­شود. حال در لحظه صفر C1 درخواست 5 بسته و C2 درخواست 10 بسته می­کند. در این حالت همچنان 50 بسته در صف وجود دارد، با این تفاوت که 15 بسته به حالت گذرا درآمده است. بعد از پردازش توسط C1 وC2 اعلان پایان کار داده و تقاضای بسته جدید کرده به این شکل بسته‌ها از صف حذف‌شده و بسته‌های جدید برای آن‌ها ارسال می­شود.

برای استفاده از RabbitMQ در پروژه میتوان از زبان­های برنامه نویسی متعددی استفاده نمود در این پروژه از زبان پایتون و کتابخانه Pika استفاده شد که توسط خود rabbitMq معرفی و پیشنهاد شده است. که در فصل های بعدی این کتابخانه را بهتر بررسی خواهیم نمود.

### راه­اندازی RabbitMQ با داکر

از پیش­تر در بخش مروری بر داکر به یاد داریم که می­توان ایمیج های استاندارد را از داکرهاب، بارگذاری و مورداستفاده قرارداد. RabbitMQ خود یکی از ایمیج های استاندارد است که به‌سادگی از آن استفاده می­شود. برای این کار می­توانیم در داخل داکر فایل از ایمیج استاندارد شروع کرده و پلاگین­هایی که قصد فعال‌سازی آن را داریم، با اجرای دستور مربوطه به‌کارگیریم تا ایمیج جدیدی با ویژگی‌هایی که مدنظر داریم ساخته شود. در شکل ‏2–21 محتوای داکر فایل را می‌توان مشاهده کرد.

FROM rabbitmq:3-management

COPY ./Conf/definitions.json /etc/rabbitmq/

COPY ./Conf/enabled\_plugins /etc/rabbitmq/

COPY ./Conf/rabbitmq.conf /etc/rabbitmq/

RUN rabbitmq-plugins enable rabbitmq\_shovel

RUN rabbitmq-plugins enable rabbitmq\_shovel\_management

RUN rabbitmq-plugins enable rabbitmq\_federation

RUN rabbitmq-plugins enable rabbitmq\_federation\_management

EXPOSE 5672

EXPOSE 15672

شکل ‏2–21-محتوای داکر فایل برای راه‌اندازی RabbitMQ

حال با به­کارگیری داکر کامپوز تنها کافی است دستور دهیم که ایمیج فوق ساخته شود و پورت مخصوص به ارتباط با کانتینر و کامپیوتر میزبان برقرار شود. همچنین می­توان فایل تنظیمات را تغییر داد و آن را در داخل کانتینر قرار داد. در شکل ‏2–22 می­توان دستوراتی که در فایل docker-compose.yaml برای بالا آمدن کانتینر RabbitMQ نوشته‌شده را مشاهده کرد.

services:

  queue:

    build: rbmq/.

    container\_name: rabbitmq

    ports:

      - 5672:5672

      - 15672:15672

    networks:

      - app-tier

شکل ‏2–22- بخش مرتبط با RabbitMQ در داکر کامپوز

## سرویس Redis

Redis یک ذخیره کننده ساختار اطلاعات، داخل حافظه‌ای متن‌باز است که می­تواند به‌عنوان پایگاه­داده، حافظه نهان و ابزاری برای یکپارچه‌سازی عمل کند. Redis از ساختارهای اطلاعات متفاوتی ازجمله؛ رشته­ها، هش­ها، لیست­ها، مجموعه­ها، مجموعه­های مرتب‌شده برای کاربردهای متفاوت و جریان­های اطلاعات (سری زمانی‌ها) پشتیبانی می­کند و توانایی اجرا بر روی خوشه­ای از کامپیوترها را دارا است.

در این نرم‌افزار بعضی از امکانات محاسباتی ازجمله اضافه کردن به رشته‌ها، تغییر در اعداد، تغییر در لیست‌ها، محاسبه اشتراکات دو مجموعه، بررسی المان‌های منحصربه‌فرد و ...را انجام داد. Redis به زبان **ANSI C** نوشته‌شده و بر روی همه‌ی سیستم‌عامل‌ها بدون هیچ‌گونه پیش‌نیاز قابل‌اجرا است. اگرچه این برنامه بر روی همه‌ی سیستم‌عامل‌ها توانایی اجرا دارد ولی خود برنامه، استفاده از سری سیستم‌عامل‌های Linux را پیشنهاد می­دهد. اینکه Redis اطلاعات را در حافظه ذخیره می‌کند، کاملا صحیح است اما به این معنی نیست که پس از خاموش شدن و یا هر اتفاقی که باعث خالی شدن حافظه شود، داده‌های ما پاک می‌شوند. بلکه Redis برای نگه‌داری دائمی داده‌ها آن‌ها را با توجه به تنظیماتی که برای آن مشخص کرده‌ایم به دیسک اصلی سیستم منتقل کرده و بعد از پاک شدن حافظه مجدد می‌تواند آن‌ها را منتقل کند و کار را از سر بگیرد. این ویژگی باعث شده اصطلاحاً به آن **on-disk persistence** بگویند.

زمانی از ذخیره‌سازی موقت[[36]](#footnote-36) استفاده می‌شود که قصد داشته باشیم دسترسی به هارددیسک کمتر انجام شود. به‌عبارت‌دیگر در ذخیره­سازی موقت، اطلاعات در حافظه موقت ذخیره می‌شود که این فرآیند سرعت دسترسی به اطلاعات و بارگذاری آن‌ها را افزایش می‌دهد. از این طریق در کنار صرفه‌جویی در زمان و افزایش سرعت، دسترسی کمتری به منابع موردنیاز انجام می‌شود. این امر نیز به بهینه‌سازی بیشتر کمک می‌کند. به این نکته نیز باید اشاره کرد که در Redis اطلاعات در حافظه ذخیره می‌شوند، این امر باعث می‌شود دسترسی به آن‌ها با سرعت بسیار بیشتری انجام شود؛ اما این سکه روی دیگری نیز دارد و امکان ذخیره‌سازی دائمی اطلاعات را در Redis وجود نخواهد داشت. در این پروژه ازآنجا که پس از تجمیع اطلاعات نیازمند ارتباط میان برنامه ها را داریم از این سرویس استفاده شد.

### انواع ذخیره سازی اطلاعات در Redis

در Redis دو عنصر، کلید و مقدار[[37]](#footnote-37) داریم. عنصر مقدار می­تواند انواع مختلفی داشته باشد. اجازه دهید به برخی این مقادیر و کاربردهای آن‌ها نگاهی بیندازیم.

**رشته‌ها:** اگر مقدار از نوع رشته بود، می­توان عملیات درج، بهنگام سازی، حذف، دریافت را از آن کلید انجام داد. مثلاً با دستور زیر می­توان یک کلید ساخت و یک‌رشته در آن درج کرد:

|  |  |
| --- | --- |
| **یک کلید به اسم user\_1 ساخته‌شده و  book1 به‌عنوان مقدار برای این کلید در نظر گرفته می­شود.** | **SET user\_1 book1** |
| **مقدار مورد (book1) نظر برای کلید user\_1 را برگردانده می­شود.** | **GET user\_1** |

**لیست:** با استفاده از لیست می­توان یک آرایه دلخواه داشت که بتوان در این آرایه، عنصری را اضافه یا کم کرد و عملیات مختلف دیگر را انجام داد.

**دسته‌ها**[[38]](#footnote-38)**:** اگر مقدار موجود از نوع دسته‌ها باشد درواقع یک لیست وجود دارد که هیچ‌کدام از عناصر آن تکراری نیستند.

**دسته‌های منظم شده**[[39]](#footnote-39)**:** همانند مقدارهایی از نوع دسته است با این تفاوت که هر عنصر از مجموعه دارای وزن است و این وزن می­تواند به‌صورت مرتب نگهداری شود. (مثلاً از وزن کم به زیاد)

**هش­ها[[40]](#footnote-40):** اگر با انواع JSON آشنایی داشته باشید درک مقادیر هش‌ها ساده­تر است. این مقادیر می­توانند اشیایی مانند JSON را در خود جای دهند. بدین شکل که ابتدا کلیدی برای ذخیره‌سازی اطلاعات دریافت کرده و سپس به ذخیره‌سازی اطلاعات به‌صورت زوج مرتب‌هایی از کلید و مقدار می­پردازد.

علاوه بر موارد فوق، Redis می­تواند مقادیر دیگری را نیز ذخیره و بازیابی کند. یکی از انواع آن‌ها، HyperLogLog است. فرض کنید می­خواهید تعداد تکرار یک عنصر خاص از یک لیست را به دست آورید. اگر این لیست بسیار بزرگ باشد، این کار به‌راحتی انجام نمی­پذیرد. مقادیر HyperLogLog می‌تواند با دقت بسیار بالا (اما نه ۱۰۰ درصد) تعداد تکرار یک عنصر خاص را حدس بزند. این کار با استفاده از الگوریتم‌های خاصی امکان‌پذیر است. انواع دیگری مانند ذخیره‌سازی عناصری از جنس موقعیت‌های مکانی و یا Bitmap های نیز در Redis وجود دارند که کاربردهای خاص خود را دارند.

### مکانیزم لیست کنترل دسترسی

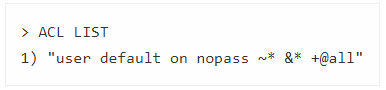
مکانیزم لیست کنترل دسترسی یا ACL[[41]](#footnote-41) قابلیتی است که به اتصالات خاصی اجازه می­دهد از نظر دستورات قابل اجرا و کلیدهای قابل دسترسی محدود شوند. روش کار به این صورت است که پس از اتصال، یک کاربر باید با ارائه یک نام کاربری و یک رمز عبور معتبر احراز هویت کند. اگر مرحله احراز هویت موفقیت آمیز بود، اتصال با یک کاربری معین و محدودیت هایی که کاربر دارد متصل می شود. Redis را می­توان به گونه ای پیکربندی کرد که اتصالات جدید که قبلاً با یک کاربر "پیش فرض" احراز هویت شده اند، به اطلاعات دسترسی پیدا کنند. (این پیکربندی پیش فرض است) بنابراین پیکربندی کاربر پیش فرض، به عنوان یک اثر جانبی، توانایی ارائه تنها زیرمجموعه خاصی از عملکردها را برای اتصالات دارد. قبل از استفاده از ACL، ممکن است بخواهید از خود بپرسید هدفی که می خواهید با اجرای این لایه حفاظتی به دست آورید چیست. به طور معمول دو هدف اصلی وجود دارد که ACL به خوبی انجام می­دهد:

* شما می‌خواهید با محدود کردن دسترسی به دستورات و کلیدها، امنیت را بهبود ببخشید تا کلاینت‌های غیرقابل اعتماد دسترسی نداشته باشند و کلاینت‌های مورد اعتماد فقط حداقل سطح دسترسی را به پایگاه داده برای انجام کار مورد نیاز داشته باشند. به عنوان مثال، برخی از مشتریان خاص ممکن است فقط بتوانند دستورات "فقط خواندنی" را اجرا کنند.
* شما می‌خواهید ایمنی عملیاتی را بهبود ببخشید، به طوری که به دلیل خطاهای نرم‌افزاری یا اشتباهات دستی، دسترسی فرآیندها یا انسان‌ها به Redis مجاز نباشد و به داده‌ها یا پیکربندی برنامه ها آسیب برساند. به عنوان مثال، هیچ دلیلی وجود ندارد که شخصی کاربری عادی دارد بتواند دستور FLUSHALL، که کل دیتابیس را پاک می­کند، را فراخوانی کند.

یکی دیگر از کاربردهای معمولی ACL مربوط به نمونه های Redis مدیریت شده است. Redis اغلب به عنوان یک سرویس مدیریت شده توسط تیم های داخلی شرکت ارائه می شود که زیرساخت­های Redis را برای سایر مشتریان داخلی مدیریت می کنند، یا در یک راه اندازی نرم افزار به عنوان یک سرویس توسط ارائه دهندگان سرویس ابری ارائه می شود. در هر دو چنین تنظیماتی، ما می خواهیم مطمئن باشیم که دستورات پیکربندی برای مشتریان مستثنی شده است.

ACL با استفاده از یک DSL[[42]](#footnote-42) (زبان خاص دامنه) تعریف می شوند که توصیف می کند یک کاربر مشخص می تواند چه کاری انجام دهد یا نه. چنین قوانینی همیشه از اولین تا آخرین، از چپ به راست اجرا می شوند، زیرا گاهی اوقات ترتیب قوانین برای درک اینکه کاربر واقعاً قادر به انجام چه کاری است مهم است. به طور پیش فرض یک کاربر تعریف شده است که به آن پیش فرض می گویند. ما می‌توانیم از دستور ACL LIST برای بررسی ACLهای فعال فعلی استفاده کنیم و تأیید کنیم که پیکربندی یک نمونه Redis تازه شروع شده و با پیکربندی پیش‌فرض متفاوت است.

دستور شکل ‏2–23 لیستی از کاربران را با همان قالبی که در فایل های پیکربندی Redis استفاده می­شود به نمایش گذاشته است و به معنای دریافت ACL های فعلی تنظیم شده برای کاربران را گزارش می دهد. کلمه اول در هر سطر عبارت "user" و سپس نام کاربری است. کلمات بعدی قوانین ACL هستند که چیزهای مختلفی را توصیف می کنند. ما با جزئیات نشان خواهیم داد که قوانین چگونه کار می کنند، اما در حال حاضر کافی است بگوییم که کاربر پیش فرض برای دسترسی بدون نیاز به رمز عبور، برای دسترسی به هر کلید ممکن (~\*) پیکربندی شده است. کانال Pub/Sub (&\*)، و قادر به فراخوانی هر دستور ممکن (+@all) است. همچنین، در این مورد خاص کاربر پیش‌فرض، داشتن قانون nopass به این معنی است که اتصالات جدید به طور خودکار با کاربر پیش‌فرض بدون نیاز به تماس صریح AUTH احراز هویت می‌شوند.



شکل ‏2–23- دریافت کل سطح دسترسی کاربران به اطلاعات مجاز در حالت پیش فرض

در زیر لیستی از قوانین معتبر ACL آمده است. برخی از قوانین فقط کلمات واحدی هستند که به منظور فعال کردن یا حذف، یا انجام یک تغییر معین در کاربر ACL استفاده می‌شوند. قوانین دیگر پیشوندهای char هستند که با نام دستورات یا دسته‌ها، یا الگوهای کلیدی و غیره ترکیب می‌شوند.

**فعال و غیر مجاز کاربران:**

on: کاربر را فعال می­کند: امکان احراز هویت به عنوان این کاربر وجود دارد.

off: کاربر را غیرفعال می­کند: دیگر امکان احراز هویت با این کاربر وجود ندارد، اما اتصالات قبلاً تأیید شده همچنان کار خواهند کرد.

**دستورات مجاز و غیر مجاز:**

+<command>: دستور را به لیست دستوراتی که کاربر می تواند فراخوانی کند اضافه می­کند. قابل استفاده با | برای اجازه دادن به دستورات فرعی (به عنوان مثال "+config|get").

-<command>: دستور را از لیست دستوراتی که کاربر می تواند فراخوانی کند حذف می­کند. قابل استفاده با | برای مسدود کردن دستورات فرعی (به عنوان مثال "-config|set").

+@<category>: همه دستورات را در این دسته اضافه می­کند تا توسط کاربر فراخوانی شود، با دسته بندی های معتبر مانند @admin، @set، @sortedset، ... و غیره، لیست کامل را با فراخوانی ACL CAT مشاهده کنید. فرمان دسته ویژه @all به معنای تمام دستورات است، هم دستوراتی که در حال حاضر در سرور وجود دارد و هم آنهایی که در آینده از طریق ماژول ها بارگذاری شوند.

-@<category>: +@<category> را می پسندد اما دستورات را از لیست دستوراتی که مشتری می تواند فراخوانی کند حذف می کند.

+<command>|first-arg: یک آرگومان اول خاص از یک دستور غیرفعال را مجاز کنید. توجه داشته باشید که این فرم به عنوان منفی مانندSELECT|2- مجاز نیست، بلکه فقط افزودنی است که با "+" شروع می شود.

**دستور فعال سازی برای همه**:

نام مستعار برای +@all. توجه داشته باشید که به معنای توانایی اجرای تمام دستورات آینده بارگذاری شده از طریق سیستم ماژول ها است. nocommands: نام مستعار برای -@all.

کاربران را می توان به دو روش اصلی ایجاد و تغییر داد:

* با استفاده از ارسال دستور در خط فرمان ACL SETUSER.
* تغییر پیکربندی سرور، جایی که کاربران می‌توانند تعریف شوند را هنگام فعال سازی سرویس تغییر داد و سرور را دوباره راه‌اندازی شود.

### راه­اندازی Redis با داکر کامپوز

از پیش‌تر در بخش مروری بر داکر به یاد داریم که می­توان ایمیج های استاندارد را از داکرهاب، بارگذاری و مورداستفاده قرارداد، Redis یکی از ایمیج های استاندارد است که می­توان به‌سادگی از آن استفاده نمود، برای استفاده از این ایمیج با داکر کامپوز، تنها کافی است، پورت مخصوص به ارتباط با کانتینر آن را به کامپیوتر میزبان متصل کرده و فضایی در کامپیوتر میزبان برای ذخیره‌سازی اطلاعات برای آن اختصاص دهیم تا با شروع به کار مجدد کانتینر، اطلاعات ذخیره‌شده در آن از بین نرود، همچنین می­توان فایل تنظیمات مربوط به Redis را تغییر داد و در کانتینر قرارداد و دستور داد تا برنامه Redis بر اساس تنظیماتی که در کانتینر قرارگرفته، اجرا شود. در شکل ‏4–17 می­توان دستوراتی که در فایل docker-compose.yaml برای بالا آمدن کانتینر Redis نوشته‌شده را مشاهده کرد.

  cache:

    image: redis:latest

    container\_name: redis

    ports:

        - 6379:6379

    volumes:

        - ./redis/config/redis.conf:/redis.conf

        - ./redis/Data:/data

    command: [ "redis-server", "/redis.conf" ]

    networks:

      - app-tier

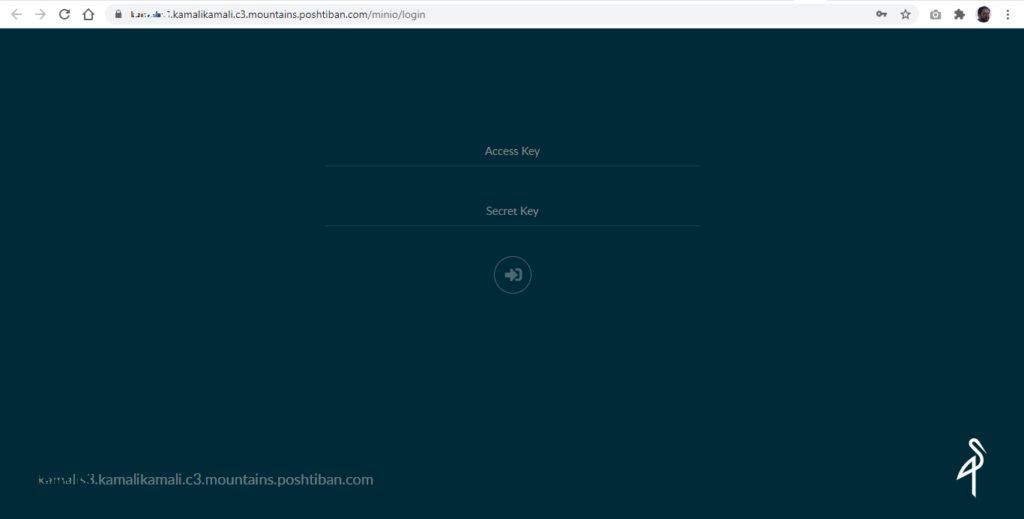
شکل ‏2–24- بخش مرتبط با Redis در داکر کامپوز

## سرویس MinIO

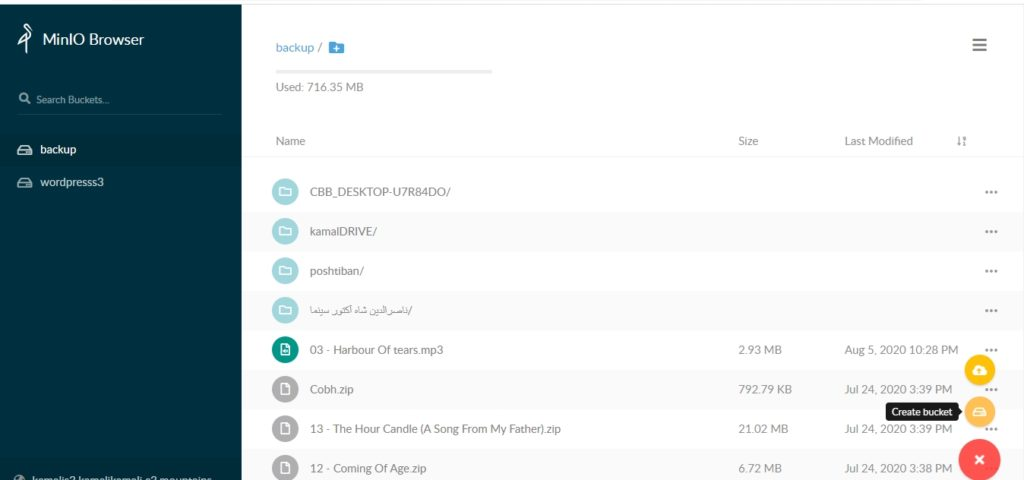
بر اساس پروپوزال نیازمند آن هستیم که بتوانیم اطلاعات دوربین­ها را در سرور ذخیره سازی نماییم و اطلاعات ذخیره شده را به کاربر نمایش دهیم برای انجام این کار نیاز داریم که اطلاعات دوربین ها را در سرور ذخیره کنیم. و با کمک یک سرویس اطلاعات ذخیره شده را به کاربر نمایش دهیم. به طور عمومی ذخیره سازی فیلم ها به صورت Object Storage انجام می­شود و به معنا است که فیلم به صورت یک فایل ذخیره خواهد شد. Object Storage روشی برای ذخیره‌سازی فایل‌ها است که با کمک آن می‌توان انواع فایل­ها اعم از عکس و فیلم و متن را ذخیره شوند. مزیت شیوه نسبت به فایل سیستم محلی، این است که شما ‌می‌توانید فایل‌های خود را در فضای ابری آپلود و دانلود کنید. همچنین این امکان وجود دارد که برای فایل‌های آپلود شده سطح دسترسی تعیین شود و در صورت نیاز فایل‌ها به صورت عمومی به همه کاربران برنامه کاربردی تحویل داده شود. معروف ترین ابزاری در حال حاضر این سرویس را ارائه میکند minIO است.

MinIO (به فارسی: مین آی او) یک نرم افزار متن باز، سمت سرور و سازگار با AWS S3 برای پیاده سازی Object Storage است. این سرویس به صورت اختصاصی برای نگهداری داده در مقیاس بالا و ایجاد یک ابر خصوصی است که با زبان GO طراحی شده است. این ابزار با ساختار S3 سازگار است و رابط کاربری مناسبی در اختیار توسعه دهنده قرار می‌دهد. وقتی با داده‌های بی‌ساختار و استاتیک با حجم بالا مانند اطلاعات دوربین روبه‌رو ­شویم، بهترین گزینه ها نصب MinIO در سمت سرور است. بدین ترتیب تصاویر، ویدیوها، بک‌آپ‌ها، لاگ‌ها و فایل‌های مختلف را با حجم چند کیلو بایت تا 5 ترابایت (تنها برای یک فایل) به راحتی و با سرعت ذخیره سازی شوند.

MinIO قابلیت استقرار بر روی یک دیسک واحد یا کلاسترهای توزیع شده را دارد. همچنین می‌تواند به عنوان رابط کاربری در راه‌کارهای ذخیره سازی ابجکتی مثل Azure Blob storage و آبجکت استوریج Google Cloud Platform یا در محیط داکر برای ارائه آبجکت استوریج به عنوان میکرو سرویس، بکار گرفته شود. در ساختار S3 برای اتصال به Object Storage به سه مقدار کلید دسترسی، کلید خصوصی و آدرس نقطه نهایی نیاز است. همچنین با توجه به این که در روش ذخیره سازی آبجکتی داده‌ها در یک ساختار flat ذخیره می شوند، مفهومی به نام دایرکتوری یا پوشه وجود ندارد؛ بنابراین برای دسته بندی فایل‌ها در آبجکت استوریج Bucket (سطل) ایجاد می­شود. می­توان به تعداد مورد نیاز bucket ساخت تا فایل ها را دسته بندی کرد. در MinIO فایل‌های ذخیره سازی شده در باکت می‌توانند مسیر (path) داشته باشند و تا زمانی که فایلی در مسیر وجود داشته باشد می­توانید آن مسیر را به عنوان یک پوشه در نظر بگیرید ولی باید به یاد داشته باشید که در ذخیره سازی آبجکتی پوشه یا دایرکتوری وجود ندارد. این برنامه کاربردی همانند rabbitMq یک پنل وب دارد که در شکل ‏2–25 و شکل ‏2–26 می­توانید آن را مشاهده کنید.



شکل ‏2–25- صفحه ورود پنل وب سرویس MinIO



شکل ‏2–26-پنل وب سرویس MinIO

### راه اندازی MinIO با داکر کامپوز

برای پیاده سازی این سرویس از پیش‌تر در بخش مروری بر داکر به یاد داریم که می­توان ایمیج­های استاندارد را از داکرهاب، ‏بارگذاری و مورداستفاده قرارداد، ‏MinIO‏ یکی از ایمیج های استاندارد است که می­توان به‌ سادگی از آن ‏استفاده نمود، برای استفاده از این ایمیج با داکر کامپوز، تنها کافی است، پورت مخصوص به ارتباط با ‏کانتینر آن را به کامپیوتر میزبان متصل کرده و فضایی در کامپیوتر میزبان برای ذخیره‌سازی اطلاعات ‏برای آن اختصاص دهیم تا با شروع به کار مجدد کانتینر، اطلاعات ذخیره‌شده در آن از بین نرود و همچنین اطلاعات دوربین ها را بتواند به کاربر نمایش دهد، سپس دستور اجرای سرویس را وارد کرد تا بر اساس جایی که داخل کانتینر به بیرون متصل شده عمل کند. در مورد نحوه اتصال و ارتباط با بیرون در ادامه تشریح خواهد شد. در شکل ‏4–17 می­توان ‏دستوراتی که در فایل ‏docker-compose.yaml‏ برای بالا ‏آمدن کانتینر ‏MinIO‏ نوشته‌شده را مشاهده کرد.

هنگام تنظیم پارامتر های این سرویس باید پارامتر های محیطی برای کنترل کاربر ادمین تعیین نمود که به صورت پیش فرض admin و admin1234 در نظر گرفته شد. همچنین برای اتصال اطلاعات ذخیره دوربین به داخل کانتینر، یک پوشه در داخل پروژه تحت عنوان Data تعریف شد که برنامه ذخیره کننده اطلاعات دوربین، فایل هایی یک دقیقه ای در پوشه خواهد ریخت این پوشه به پوشه داخل لینوکس کانتینر متصل شده و هنگام بالا آمدن ، این دستور وارد شد که سرویس بر روی آن اجرا شود. پس این بدان معناست که اگر برنامه ذخیره کننده اطلاعات، اطلاعات خود را در این پوشه از کامیپوتر سرور منتقل کند این اطلاعات در شبکه قابل دسترس، کاربران خواهد بود.

  minio:

    image: "minio/minio"

    restart: always

    environment:

      MINIO\_ACCESS\_KEY: admin

      MINIO\_SECRET\_KEY: admin1234

    volumes:

      - "./Data:/data"

    ports:

      - "9000:9000"

      - "9001:9001"

    command: server /data --console-address ":9001"

    networks:

      - app-tier

شکل ‏2–27- بخش مرتبط با سرویس MinIO در فایل داکر کامپوز

## جمع بندی

در بخش‌های فوق هر یک از نرم‌افزارهای به‌کاررفته در این پروژه را بررسی کردیم و ویژگی هایی که در اختیار ما قرار میدهند را دریافتیم ، دیدیم که میتوان با RabbitMQ بسته های تصاویر را درون صف هایی قرارداد و جریان داده ایجاد کرد. همچنین دیدیم که با Redis میتوان کاربران کنترل نمود و پارامترهایی را در آن ذخیره کرد و در پایان نیز دیدیم که میتوان اطلاعات ذخیره شده در سرور را با سرویس MinIO در اختیار کاربران قرارداد حال با توجه به توضیحات فوق در فصل بعد این سرویس ها را در کنار یک دیگر قرار داده و معماری پیشنهادی را تشریح می­کنیم و برنامه های کاربردی که با کمک این سرویس ها در اختیار ما قرار میدهند را طراحی خواهیم نمود.

# فصل سوم طراحی و معماری برنامه های کاربردی

**طراحی و معماری برنامه های کاربردی**

در این فصل پیش سرویس ها و نحوه اجرای آن ها را بررسی نمودیم حال وقت آن رسیده که ابتدا معماری که ابزارهای فوق در کنار یکدیگر فعال خواهند بود را معرفی کنیم و سپس به سراغ طراحی هر یک از برنامه های کاربردی که کاربران می­خواهند از آن استفاده کنند بپردازیم.

## چالش های طراحی معماری

بر اساس پروپوزال نیازمندی و هدف این پروژه آن است که بتوانیم اطلاعات دوربین ها دریافت کنیم سپس اطلاعات را پردازش شده را در اختیار کاربرنهایی قراردهیم. کاربر باید بتواند به اطلاعات ذخیره شده دوربین ها دسترسی داشته باشد. و اطلاعات دوربین ها را در زمان واقعی به همراه اطلاعات پردازش شده دریافت کند. برای رسیدن به هدف فوق معماری زیر در سه برنامه کاربردی ارائه شد. که در یک شبکه داخلی ، یک برنامه بر روی کامپیوتر سرور، یک برنامه بر روی سرور پرازشی و یک برنامه بر روی کلاینت اجرا شود.[[43]](#footnote-43) در شکل ‏3–1 می­توانیم معماری کلی سیستم را مشاهده کنیم.

در این معماری داکر را بر روی کامپیوتر سرور به نصب خواهیم کرد و سپس سرویس ها که شامل سرویس RabbitMQ، Redis، MinIO هستند را به اجرا خواهیم آورد. سپس برنامه کاربردی سرور را روی این سیستم اجرا خواهیم کرد تا بتواند دوربین هایی که در شبکه قرار گرفته اند را کنترل کند و اطلاعات آن ها را دریافت کند همچنین وظیفه کنترل سایر برنامه ها را نیز انجام دهد. این بدان معناست که سایر برنامه ها نیازمند این برنامه هستند. در ادامه برنامه سرورپردازشی را اجرا خواهیم نمود تا در آن فرایند پردازش اطلاعات دوربین ها را انجام دهد این برنامه نتیجه پردازش خود را به به برنامه سرور بر می­گرداند تا کاربران بتوانند با اتصال به برنامه سرور به اطلاعات پردازش شده دسترسی پیدا کنند. در پایان نیز برنامه کاربر میتواند بر هر کامپیوتری در شبکه به سرور اصلی متصل شده و از امکانات آن همچون دسترسی به فایل های ذخیره شده ی دوربین ها، دریافت زنده تصاویر دوربین ها ، دریافت زنده اطلاعات پردازش شده دوربین ها دسترسی پیدا کند. برای تشریح بهتر هر کدام از این بخش ها ابتدا چالش هایی که با آن ها مواجه هستیم را بررسی می­کنیم و راهکاری که برای هر کدام ارائه خواهیم داد.



شکل ‏3–1-معماری کلی پروژه

### چالش دریافت اطلاعات از دوربین ها

اولین چالشی که در این پروژه مطرح است دریافت اطلاعات از دوربین هاست. دوربین های متفاوتی در بازار برای تهیه موجود است ولی همانطور که در فصل اول بررسی نمودیم، در میان آن ها استانداردی تحت عنوان onvif وجود دارد که با پروتکل rtsp اطلاعات دوربین ها را منتقل می­کنند. حال چالش اصلی این است، با فرض اینکه دوربین ما استاندارد باشد و بتواند اطلاعات خود را تحت پروتکل rtsp در شبکه منتشر کند. چگونه میخواهیم این اطلاعات را دریافت کنیم. برای رسیدن به این مقصود از کتابخانه opencv استفاده شد، چرا که در این کتابخانه توانایی دریافت اطلاعات به صورت فریم فریم با پروتکل rtsp موجود است و می­تواند جریان اطلاعات تصویر را به صورت فریم دیکد کند. بدین شکل که در یک حلقه while اطلاعات دوربین ها را دریافت می­کند. و در اختیار کاربر قرار می­دهد. برای اینکار به ip دوربین و اطلاعات احراز هویتی آن در شبکه نیاز داریم.[[44]](#footnote-44)

### چالش انتقال تصویر در زمان واقعی و تعیین سطح دسترسی

چالش بعدی که با آن مواجه هستیم نحوه انتقال فریم های تصویر است، در فصل قبل دیدیم که می­توان توسط RabbitMQ مراکز توزیع داده تعریف کرد که اطلاعات را به آن ارسال نمود و صف هایی ایجاد کرد که بتوان به این مراکز متصل شده و اطلاعات را دریافت کنند. در این پروژه قصد داریم که چندین کاربر بر اساس سطح دسترسی خود بتوانند اطلاعات دوربین ها دریافت کنند. برای اینکه بتوان اطلاعات یک دوربین را بین چندین کاربر تقسیم نمود، نوع مرکز انتقال داده را باید از نوع خروج به همه تعیین نمود. بدین معنا که اگر صفی به این مرکز توزیع داده متصل شد، یک کپی از اطلاعاتی که داخل این مرکز توزیع قرار دارد به تمامی صف ها منتقل شود و همه کاربران بتوانند به اطلاعات دوربین ها دسترسی داشته باشند. ولی مسئله اینجاست که کدام کاربران اجازه دسترسی به اطلاعات این دوربین ها را دارند. یک راهکار این است که این مراکز توزیع اطلاعات هر کدام در یک ماشین مجازی که خود RabbitMQ در اختیار ما قرار میدهد قرار بگیرند تا امنیت اطلاعات تامین شود و فرایند احراز هویت برای دسترسی به اطلاعات دوربین ها حتی توسط خود RabbitMQ انجام شود یا راهکار دیگر این است که همه مراکز توزیع داده در یک ماشین قراربگیرند و سطح دسترسی افراد در Redis تعیین شود و همچنین نام مراکز توزیع داده نیز در آن قرارگیرد و کاربران در Redis احراز هویت شوند و براساس احراز هویت خود نام دوربین هایی که میتوانند به آن دسترسی داشته باشند در اختیار آنها قرارگیرد که تنها به توانند به دوربین های مجاز خود متصل شوند. در مدل اول سطح امکانات سخت افزاری که توسط کاربر در سرور تعریف میشود بیشتر است. ولی روش دوم که از نظر سخت افزاری روش دیگری زیرساخت سخت افزاری بسیار مناسب است. در این پروژه از روش دوم برای انجام این کار استفاده شد که برای کاربران در redis سطح دسترسی تعریف می­شود و همچنین برای آنها در ACL نام کاربری و رمز عبور تخصیص داده می­شود که بتوانند بعد از احراز هویت در redis بر اساس سطح دسترسی خود به دوربین های مجاز خود دسترسی داشته باشند.

### چالش پردازش در زمان واقعی

در بخش قبل دیدیم که اطلاعات دوربین ها در مراکز توزیع اطلاعات RabbitMQ با نرخی مشخص وارد می­شوند یعنی با ایجاد یک صف می­توان اطلاعات دوربین ها را دریافت کرد. اجازه دهید مسئله را با یک مثال تشریح کنم، فرض کنید که اطلاعات نرخ 30 فریم بر ثانیه وارد مرکز اطلاعات می­شوند. و ما قصد داریم که همه فریم های این تصویر را پردازش کنیم. این بدان معناست که مدل هوش مصنوعی ما باید هر فریم را در حداکثر 30/1 ثانیه یا ساده تر بگویم، در 33 میلی ثانیه پردازش کند. در صورتی که زمان پردازش از عدد فوق بیشتر شود نمی­تواند از صفی که اطلاعات به آن وارد میشوند نمونه بردارد و صف پر خواهد شد و سیستم دچار اخلال می­شود. پس راهکاری باید برای حل این مسئله ارائه شود. راهکار ساده این است که نیازی نیست همه فریم ها پردازش شوند. و از n فریم که از صف برداشته می­شود تنها کافی است 1 فریم پردازش شود. بگذارید مثال فوق را برای n=15 تکرار کنم. اطلاعات با نرخ 30 فریم بر ثانیه وارد می­شوند و ما قصد داریم از هر 15 فریم تنها یک فریم را پردازش کنیم پس در یک ثانیه تنها 2 فریم پردازش خواهند شد ولی در عوض مدل هوش مصنوعی ما زمانی به اندازه 30/15 خواهد داشت که به بیان ساده تر 500 میلی ثانیه خواهد داشت. پس در صورتی که یک پارامتر با این ویژگی وارد مسئله کنیم. می­توان در عمل مدل های هوش مصنوعی متفاوتی را بکارگرفت. پس برای راهکاری که برای چالش پردازش بکارگرفته شد این است که، پارامتری را در اختیار کاربر قرار خواهیم داد که بتواند متناسب با مدل هوش مصنوعی خود این پارامتر را بهینه کند. در شکل شکل ‏3–2 می­توان نحوه پردازش در زمان واقعی را مشاهد نمود.



شکل ‏3–2- نحوه کنترل پردازش بسته ها در زمان واقعی

### چالش ذخیره سازی اطلاعات

بر اساس پروپوزال می­دانیم که باید اطلاعات دوربین ها به نحوی ذخیره کنیم که بتوانیم آن ها را به کاربر نمایش دهیم. برای کنترل این بخش هنگامی که اطلاعات دوربین ها دریافت میکنیم، پیکسل اول تصویر را یک شماره رخداد[[45]](#footnote-45) میزنیم. تا ترتیب اطلاعات ها را داشته باشیم. سپس با کمک این شماره ترتیب­ها که ساخته شد میزان زمان تصاویر دریافتی را محاسبه می­کنیم. در ادامه با کمک کتابخانه opencv سعی می­کنیم فریم های تصویر را مجددا کد کرده و به فایل های ویدیویی یک دقیقه ای تبدیل کنیم. با این کار اطلاعات دوربین ها ذخیره خواهند شد و برای اشتراک گذاری آن ها با کاربران از سرویس MinIO که در فصل قبل دیدیم بهره میگیریم. به این شکل که محلی که فایل های 1 دقیقه دوربین ها را ذخیره می­کنیم را به سرویس مورد نظر متصل کنیم و از آنجایی که این سرویس بر روی کامیپوتری است که برنامه سرور انجام شده با حداقل فشار (چه بر شبکه، چه سخت افزاری) این کار انجام خواهد شد. دلیل جدا کردن و ایجاد فایل های یک دقیقه ای نیز بسیار ساده است. چرا که وقتی یک فایل را باز کرده و آن را پر می­کنیم اطلاعات بر روی ram نوشته خواهند شد و اگر این فایل را نبدنیم و به صورت قطعه قطعه در نیاوریم. سیستم پایدار نخواهد ماند. هر چند که بهینه سازی دقیقی بر روی عدد 1 دقیقه صورت نگرفت ولی می­توان از این عدد به عنوان عددی که انسان میتوان به سادگی تحیلیل کند استفاده نمود.

### چالش هماهنگ سازی برنامه ها

همانطور که در ابتدای این بخش مرور کردیم سیستم مورد نظر در سه کامپیوتر مجزا اجرا خواهد شد که نیازمند تامل با یکدیگر هستند چرا که وقتی در برنامه کاربردی سرور یک دوربین جدیدی تعریف می­کنیم، نیاز داریم تا در برنامه های کاربردی سرور پردازشی و کاربر این دوربین تعریف شده جدید دیده شود پس باید چند مکانیزم کلی برای بروز رسانی برنامه سرور نسبت به دو برنامه کاربردی سمت کاربر و سرور پردازشی پیاده شود. همچنین میان سرور پردازشی و برنامه سمت کاربر نیز به هماهنگ سازی نیاز داریم چرا که در صورتی که یک الگوریتم جدید بر روی دوربین ها فعال شوند، برنامه کاربردی سمت کاربر باید از آنها مطلع شود.

یکی از هماهنگ سازی ها بین برنامه کاربردی سرور با برنامه کاربردی کاربر و سرور پردازشی از جنس فعال یا غیر فعال شدن دوربین جدید است. یکی دیگر از هماهنگی ها بین برنامه کاربردی سرور با سایرین هنگام دسترسی به اطلاعات ذخیره شده دوربین هاست. بدین شکل که وقتی دوربین ها فعالیت میکنند فایل های خود را هر یک دقیقه ذخیره خواهند کرد. و وقتی فایلی جدید ذخیره شد برنامه کاربردی کاربر باید بتواند به آن فایل جدید دسترسی پیدا کند. برای حل این مسئله از سرویس redis استفاده شد بدین شکل هر یک از موارد فوق که ذکر شد در این دیتابیس ذخیره خواهند شد. و همچنین یک کلید بازیابی[[46]](#footnote-46) در هر یک از برنامه ها باید ایجاد شود تا با کلیک بر این گزینه، برنامه ها اطلاعات خود را نسبت redis به روز کنند.

## طراحی معماری

در معماری پیشنهادی سه برنامه کاربردی ارائه شد. هر برنامه از یک بخش پشته و یک بخش رابط کاربری تشکیل شده است. در بخش رابط کاربری تنظیمات سرویس ها انجام خواهد شد و در بخش پشته برنامه، فرایند ارتباط با سرویس ها اتفاق خواهد افتاد. در این بخش نحوه کنترل سرویس ها که معماری کلی ما را تشکیل میدهند را بررسی خواهیم نمود و معماری را از دو دیدگاه جریان اطلاعات و دسترسی به اطلاعات ذخیره شده بررسی خواهیم نمود.

### جریان اطلاعات

قصد داریم که اطلاعات دوربین ها را به صورت زنده در شبکه بین کاربران منتشر کنیم و برای این کار به یک بروکر پیام نیاز داریم که بتوانیم اطلاعات را به آن منتقل کرده و در سوی دیگر اطلاعات را بین کاربران تقسیم کنیم. از بخش چالش ها متوجه شدیم که در بروکر خود باید نوع مرکز توزیع پیام را از جنس خروج به همه تعیین شود. ولی در مورد تعداد و نوع چینش آنها بحث نکردیم.

برای ایجاد این جریان اطلاعات نیاز داریم که اطلاعات دوربین ها را از یکدیگر متمایز کنیم. برای ایجاد تمایز میان اطلاعات دو روش پیش روی ماست؛ یکی اضافه کردن کلیدمسیریابی به بسته ها و ارسال همه آن ها به یک مرکزتوزیع پیام و راهکار دوم، ایجاد یک مرکز توزیع اطلاعات جدید برای هر دوربین که اطلاعات دوبین ها از یکدیگر متمایز شوند. حسن روش اول سادگی در کنترل بسته هاست که همه بسته ها اطلاعات خود را به یک نقطه ارسال می­کنند و صف ها باید از طریق کلید مسیریابی بسته های متناسب با خود را بردارند. این راهکار برای مواردی که بخواهیم اطلاعاتی که تاخیر زمانی در آن ها چالش نباشد راهکار بهتری است. چرا برنامه های ارسال کننده و دریافت کننده اطلاعات تنها به یک نقطه متصل می­شودند. ولی وقتی چالش انتقال ویدیو است برای جلوگیری از تاخیر، در اینکه کلید مسیریابی بسته ها چک شوند و اگر یک کاربر بخواهد به اطلاعات دو دوربین دسترسی داشته باشد، بسته ها منتظر خواهند شد راهکار اول حذف خواهد شد. پس برای هر یک از دوربین ها یک مرکزتوزیع اطلاعات باید تعریف شود که دوربین ها اطلاعات خود را به آن منتقل کنند. برای مثال در شکل ‏3–3 می­بینیم که برای دو دوربین با نام های Cam1 , Cam2 دو مرکز توزیع اطلاعات C1 وC2 ساخته شده است که به مرکز توزیع اطلاعات c1 دو صف با نام c1-1 و c1-2 متصل شده است و اطلاعات دوربین cam.1 به هر دو این صف ها وارد وارد می­شود. برنامه کاربردی سرور، وظیفه آماده کردن مراکز توزیع اطلاعات و پروسه دریافت اطلاعات از دوربین ها، و ارسال آن ها به بروکر پیام را بر عهده خواهد داشت[[47]](#footnote-47). همچنین این برنامه باید اطلاعاتی که برنامه اطلاعاتی همچون نام مرکزتوزیع اطلاعات، سطح دسترسی که برای دوربین تعیین شده،... را در دیتابیس redis قراردهد تا برنامه کاربردی سرور پردازشی و کاربر بتوانند به این مرکز توزیع اطلاعات متصل شوند.



شکل ‏3–3-فرایند ارسال اطلاعات به سرویسrabbit برای ایجاد جریان داده

برای توضیح معماری فرایند نحوه پردازش اطلاعات، مجدد به شکل ‏3–3 توجه کنید، صف c1-1 میتواند یکی از کاربرها باشد که اطلاعات دوربین را به صورت زنده نمایش می­دهد و صف c1-2 میتواند برنامه پردازشی باشد که اطلاعات دوربین ها را برای پردازش دریافت می­کند. بله درست است، برنامه پردازش اطلاعات دوربین همانند یک کاربر به صف ها متصل شده و اطلاعات دوبین ها را دریافت و پردازش می­کند. همچنین همانند یک دوربین اطلاعات پردازش شده را به یک مرکز توزیع اطلاعات جدید مانند Exchange c2 منتقل میکند و کاربر میتواند با ایجاد یک صف به این اطلاعات دسترسی پیدا کند.

پس به تعداد الگوریتم هایی که برای یک دوربین فعال خواهیم کرد یک صف به مرکزتوزیع اطلاعات دوربین مورد نظر ساخته خواهد شد و همچنین یک مرکز توزیع اطلاعات جدید تا اطلاعات پردازش شده به آن منتقل شوند. پس برنامه کابردی کاربر می­تواند با ایجاد یک صف به مرکز توزیع اطلاعات اصلی دوربین جریان اطلاعات را دریافت کند و با ایجاد یک صف به مرکز تورزیع اطلاعات پردازش شده اطلاعات پردازش شده دوربین را دریافت کند. در شکل ‏3–5 می­توان مراکز توزیع اطلاعاتی که برنامه کاربردی کاربر میتواند به آن متصل شود را مشاهده نمود، کاربر با اتصال به c1-camera می­تواند اطلاعات اصلی دوبین ، با اتصال بهdata-broker-exhange1 به اطلاعات پردازش شده توسط مدل هوش مصنوعی 1 و با اتصال به data-broker-exchange2 به اطلاعات پردازش شده توسط مدل هوش مصنوعی 2 دسترسی پیدا کند. کل اتفاقی که در این بازه خاکستری رخ داده است بر عهده برنامه کاربردی سرور پردازشی است همچنین برنامه سرور پردازشی نیازمند ذخیره اطلاعاتی همچون سطح دسترسی پردازش اطلاعات نام مرکزتوزیع اطلاعات جدید و... را در redis دارد تا برنامه کاربردی کاربر به مراکز توزیع اطلاعات پردازش شده که برای یک دوربین فعال یا غیرفعال شده است دسترسی پیدا کند .



شکل ‏3–4- فرآیند برنامه کاربردی سرور پردازشی

در نهایت برای یک دوربین مراکز توزیع اطلاعات متفاوتی ساخته خواهد شد که کاربر باید بتواند آن ها را تشخیص دهد و متناسب با سطح دسترسی خود بتواند به آن ها متصل شود و به اطلاعات آن ها دست یابد. برای اینکار از سرویس redis که سایر برنامه ها اطلاعات خود را در آن ذخیره کرده اند، استفاده شد. در ردیس مدل های فعال و الگوریتم های فعال و هر آن چیزی که متناسب با سطح دسترسی خود می­تواند به آن ها برای یک دوربین فعال در نظر گرفته شود، وجود خواهد داشت. بدین شکل برنامه کاربردی کاربر برای اتصال به سیستم ابتدا خود را در redis احراز هویت می­کند و به اطلاعات دوربین هایی که برای دسترسی به آن ها مجاز است، متصل می­شود. و سعی می­کند بر اساس آنچه برای او فراهم شده به هر کدام از این مراکز توزیع اطلاعات متصل شود و از اطلاعات آن بهره ببرد. برای این کار کافی است که تنها یک صف به این مراکز توزیع اطلاعات بسازد و به آن متصل شود تا بتواند اطلاعات دوربین­ها را دریافت کند. شکل ‏3–6 معماری از دیدگاه برنامه کاربر است. که برای مثال یک کاربر با ایجاد یک صف به اطلاعات مستقیم دوربین c1 دسترسی پیدا کرده و آن ها دریافت می­کند. برای این دوربین دو نوع الگوریتم پردازشی نیز پیاده سازی شده است که کاربر میتواند با اتصال به این اکسچنج از آن ها استفاده کند ولی نکرده است. همچنین این کاربر یا یک کاربردیگر در نقطه ای دیگر در شبکه به اطلاعات دوربین c2 متصل شده و جریان اطلاعات را دریافت میکند برای این دوربین نیز 3 الگوریتم پردازشی در حال اجراست. که آن کاربر به alg-2 متصل شده و اطلاعات را پردازش شده را نیز درکنار اطلاعات دوربین ها دریافت می­کند.



شکل ‏3–5-معماری از دیدگاه برنامه کاربردی کاربر

### دسترسی به اطلاعات ذخیره شده

در بخش چالش ذخیره سازی اطلاعات دیدیم که با کمک opencv میتوان اطلاعات ها هر یک دقیقه را ذخیره نمود و با کمک MinIO میتوان اطلاعات را در باکت ها قرارداد و برای کاربران برای اطلاعات تنظیم شده لینک دانلود ایجاد نمود. در این بخش نحوه ایجاد این فایل ها و نحوه تمایز اطلاعات دوبین ها پرداخته و نحوه کنترل و دسترسی به یک زمان مشخص را شرح خواهیم داد.

برای کنترل و تمایز اطلاعات دوربین ها، سیستم عامل میزبان، کافی است. فایل­های ذخیره شده دوربین­ها را بر اساس نامی که برای هر دوربین دسته بندی کرده، پوشه بندی کند تا هنگامی که سرویس MinIO را به آن قسمت متصل می­کنیم هر پوشه را به عنوان یک باکت در شبکه در نظر بگیرد و امکان دانلود فایل ها برای برنامه کاربردی کاربر فراهم و به اطلاعات یک دوربین مشخص دسترسی پیدا کند.

حال برای دسترسی به یک زمان مشخص از اطلاعات یک دوربین نیاز داریم که هنگام ذخیره کردن اطلاعات دوربین ها در نام فایلی که قصد داریم آن ها را ذخیره کنیم، یه برچسب زمانی بزنیم تا در سمت دیگر بتوانیم اطلاعات را دیکد کنیم یکی از رایج ترین برچسب های زمانی برچسب زمانی "زمان یونیکس"[[48]](#footnote-48) است. که یک سیستمی است که به زمان اشاره میکند. این تعداد ثانیه های است که از شروع یونیکس گذشته است، تاریخ شروع این سیستم 1 ژانویه 1970 است. کتابخانه time در پایتون از این زمان استفاده می­کند. که در این پروژه برای نام فایل های ذخیره شده دوربین ها از کتابخانه استفاده شد.

برنامه ذخیره کردن اطلاعات در دوربین ها در برنامه کاربردی سرور اجرا میشود. این برنامه یک رشته پردازشی مجزا است که هنگامی که اطلاعات دوربین ها دریافت می­شود. فرایند ذخیره­سازی اطلاعات نیز به صورت موازی رخ خواهد داد.

در مقابل برنامه کاربردی کاربر میتواند از طریق سرویس MinIO فایل هایی که برای یک دوربین در سرور ذخیره شده اند دسترسی پیدا کند و فایل ها را بررسی کند. و هنگامی که بخواهد به اطلاعات ذخیره شده دوربین ها را استفاده کند کافی است که در برنامه کاربردی نام دورین مورد نظر را انتخاب کند و فایل هایی که برای آن دوربین در سرور ذخیره شده را بررسی کرده و فایل مورد نظر خود را بر روی سیستم خود دانلود کرده و آن را برای خود پخش نماید.

## پیاده سازی برنامه های کاربردی

از بخش قبل بخاطر داریم که برای هر یک از برنامه های کاربردی باید دو بخش پشته و رابط کاربری را طراحی کنیم. در این بخش به طراحی پشته و رابط کاربری هر یک از برنامه های کاربردی سرور، سرور پردازشی و کاربر میپردازیم. بخش رابط کاربری در این پروژه بر اساس پروپوزال PyQt5 در نظر گرفته شد با مقدمه­ای بر Qt و بهترین راهکار برای این طراحی برنامه های کاربردی با Qt و نحوه توسعه آن در پایتون کمی آشنا شویم. سپس به سراغ طراحی هر یک از برنامه های کاربردی خواهیم رفت و نحوه چینش پراسس ها و رشته های پردازشی برنامه را مشاهده خواهیم نمود.

### توسعه Qt با پایتون PyQt5

PyQt یک ابزار پایتون و یک سکو برای توسعه GUI[[49]](#footnote-49) است که به عنوان کتابخانه پایتون پیاده­سازی شده است و قابلیت اجرا بر روی تمامی سیستم عامل ها را داراست. PyQt نرم افزاری رایگان است که توسط شرکت انگلیسی Riverbank Computing ساخته شده و این رابط یکی از قدرتمندترین و محبوب ترین کتابخانه است. پایکیوت دارای مجموعه ای از ماژولهای متفاوت می­باشد که هرکدام از آنها وظایف ویژه و خاصی عهده دار هستند که در ادامه به بررسی آنها را به صورت اجمالی خواهیم پرداخت.

* QtCore: ماژول QtCore شامل کلاسهای غیر گرافیکی بسیاری است. این ماژول هسته مرکزی بوده و در بر دارنده قسمتهای پر اهمیتی نظیر حلقه وقایع و مکانیسم سیگنال و اسلات می­باشد.
* QtGui: این ماژول همه کلاسهای گرافیکی را شامل می­شود.
* QtNetwork: این ماژول کلاسهایی را شامل می­شود که در ساخت برنامه­های سمت سرویس دهنده و سرویس گیرنده بر پایه پروتکلهای TCP و یا UDP مورد استفاده قرار میگیرند. کلاسهای کلاینت HTTP، FTP و نیز DNS در این ماژول قرار می­گیرند.
* QtOpenGL: این ماژول با استفاده از اپن جی ال زمینه خلق و ایجاد کار بر روی تصاویر سه بعدی را فراهم می­سازد.
* QtScript:این ماژول قابلیت اجرای اسکریپت جاوا را به برنامه­های نوشته شده می­دهد. به عبارت دیگر این ماژول زمینه توسعه و اسکریپت نویسی را برای کاربران فراهم می­سازد.
* QtWidget: این ماژول یکی از هسته مرکزی رابط کاربری است چرا که می­تواند ورودی­هایی که کاربر وارد می­کند را به هسته اصلی منتقل کند.
* ...

به طور عمومی در این پروژه ما از ماژول های QtCore,QtGui,QtWidget برای کنترل کردن رابط کاربری استفاده نمودیم. برای کنترل و طراحی رابط کاربری دو روش عمومی وجود دارد.

1- روش نوشتن آن از پایه است بدین شکل با کمک برنامه نویسی و اضافه کردن widget هایی که در کلاس Qwidget وجود دارد. المان های ورودی و خروجی را به پنجره اضافه کنیم. به طور مثال در رشته کد زیر که ساده ترین برنامه ممکن است، یک Qlabel به برنامه صفحه اضافه می­کنیم. و محتوای آن را با Hello World! پر میکنیم و به نمایش قرار میدهیم. در شکل ‏3–7 می­توان کد و خروجی این رشته کد را مشاهده کرد.

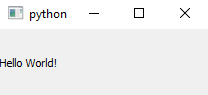
from PyQt5.QtWidgets import \*

app = QApplication([])

label = QLabel('Hello World!')

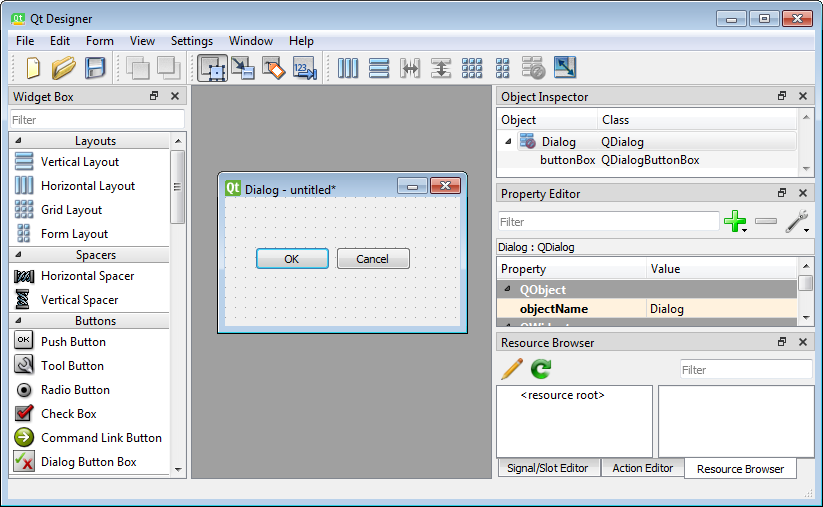
label.show()

app.exec\_()



شکل ‏3–6-کد و خروجی کد ساده ترین برنامه گرافیکی

2-راهکار دوم که سرعت توسعه رو را افزایش داده استفاده از ابزارهایی است که برای توسعه رابط کاربری معرفی شده­اند. یکی از این ابزارها که کمک می­کند برنامه ها را با سرعت بیشتری توسعه دهیم نرم افزار Qt Designer است، در شکل ‏3–8 می­توان چهارچوب کلی این نرم افزار را مشاهده کرد. این ابزار برای ساخت سریع رابط های کاربری گرافیکی با ویجت ها و از کلاس QtGUI است. این ابزار بسیار ساده با تنها کشیدن و رها کردن[[50]](#footnote-50) و شرابط چیدمان اجزایی مانند دکمه‌ها، فیلدهای متنی، جعبه‌های ترکیبی و موارد دیگر را به سادگی به شما می‌دهد.



شکل ‏3–7-نمایی از نرم افزار Qt-designer

این ابزار فایل های .ui تولید می­کند، این فایل­ها یک قالب خاص مبتنی بر XML هستند که ویجت­ها را به صورت درختی ذخیره می­کند. می‌توان این فایل‌ها را در به زبان برنامه‌نویسیC++ یا Python تبدیل کرد. در این پروژه برای طراحی رابط کاربری برنامه­های کاربردی از این نرم افزار استفاده شد. در همه برنامه کاربردی پوشه ای تحت عنوان ui قرار گرفته است که در داخل آن صفحات اپلیکیشن طراحی شده است. در ادامه بخش های متفاوت طراحی شده را بررسی خواهیم نمود.

حال برای تبدیل این فایل های .ui به فایل های پایتون نیاز داریم که از ابزار PyQt.uic.pyuic استفاده کنیم به همین منظور کافی است در خط دستور، فرمان شکل ‏3–8 را برای تبدیل فایل ServerUI.ui به فایل ServerUI.py وارد کنیم. با اجرای این دستور فایل پایتون ServerUI.py ساخته می­شود. و در ادامه می­توانیم از این فایل به عنوان رابط گرافیکی در پروژه استفاده کنیم.



شکل ‏3–8-دستور تبدیل فایل \*.ui به \*.py

در این روش که فایل ui را مستقل از منطق برنامه بنویسیم امر بسیار مناسبی است چرا که بدین شکل میتوانیم بخش ظاهر اپلیکیشن را به یک تیم و بخش توسعه برنامه را به یک تیم دیگر محول کنیم تا سرعت فرآیند توسعه محصول افزایش یابد. فایل پایتون ساخته شده حاوی یک کلاس است که در شکل ‏3–9 می­توانیم مشاهده کنیم. در متد setupUi تمامی ویجت هایی که در ابزار QtDesigner طراحی کردیم تعریف می­شود. حال برای نوشتن منطق برنامه کافی است که یک شی از جنس Qwidget بسازیم و در بخشی که می­خواهیم رابط گرافیکی برنامه را به آن معرفی کنیم کلاس فوق را به آن متصل کنیم. و به سراغ نوشتن منطق برنامه ها، رخداد ها و فرآیند ها برویم. که مثلا با فشردن این کلید چه برنامه ای اجرا شود و چه اتفاقی رخ دهد.

from PyQt5 import QtCore, QtGui, QtWidgets

class Ui\_MainWindow(object):

    def setupUi(self, MainWindow):

......

شکل ‏3–9-محتوی کد پایتون ساخته شده از نرم افزار Qt-designer

### توسعه برنامه کاربردی سرور

اجازه دهید تا از با دیدگاه کلی تر این برنامه بررسی کنیم، ما قصد داریم که به تعداد نامحدودی دوربین به سیستم اضافه کنیم پس باید یک مکانیزم برای اضافه کردن دوربین­ها و حذف دوربین­­ها در نظر بگیریم همچنین قصد داریم که بتوانیم عملکرد دوربین ها را کنترل کنیم، به معنا که بتوانیم دوربین های مجموعه را فعال یا غیرفعال کنیم. ما میخواهیم که برای هر دوربین سطح دسترسی معین تعریف شود. همچنین قصد داریم کاربرانی با سطح دسترسی معین تعریف کنیم که بر اساس سطح دسترسی خود بتوانند به این دوربین ها نظارت داشته باشند.

پر واضح است که برای هر دوربین فعال نیاز داریم که فرایندهایی یکسان، همچون دریافت اطلاعات، ارسال به بروکر و ذخیره سازی اطلاعات را انجام شود. پس نیاز داریم که به ازای هر دوربین یک فرآیند جدید که 3 وظیفه فوق را انجام می­دهد. بر روی کامیپوتر سرور اجرا کنیم. درشکل ‏3–10 می­توانیم نمایی کلی از اتفاقی که در برنامه کاربردی سرور رخ میدهد را مشاده کنیم. پس ابتدا به بررسی فرایندی که دوربین های فعال در سیستم باید انجام دهند میپردازیم. سپس به بررسی نحوه فعال سازی فرآیند فوق و همچنین پیاده سازی رابط کاربری میپردازیم تا طراحی برنامه کاربردی سرور را بررسی کنیم.



شکل ‏3–10-فرآیندی که به ازای هر دوربین باید در برنامه کاربردی سرور اجرا شود.

* **فرآیند دوربین های فعال**

نکته ای که باید در نظر بگیریم این است که آیا سه وظیفه ای که در نظر داریم باید به ترتیب اجرا شود یا باید به صورت موازی اجرا شود. پاسخ این سوال مشخص است، آن ها باید به طور موازی اجرا شوند پس فرآیندی که در نظر داریم از 3 رشته پردازشی تشکیل شده است. رشته پردازشی اول باید اطلاعات را از دوربین ها دریافت کند، رشته پردازشی دوم باید اطلاعات دریافتی را به سرور ارسال کند و رشته پرداشی سوم باید اطلاعات را ذخیره کند. برای اینکه اهمیت موازی سازی این ها بهتر درک کنیم حالت زیر را در نظر بگیرید، در لحظه صفر فریم اول را از دوربین دریافت نموده ایم و باید در 3 میلی ثانیه بعدی فریم بعدی را دریافت کنیم، سوال اینجاست که آیا میتوانیم در مدت 3 میلی ثانیه هم اطلاعات را به بروکر پیام ارسال کنیم و هم فایل را ذخیره کنیم؟ پاسخ به این سوال به میزان فشار روی سیستم بستگی دارد. اگر سیستم[[51]](#footnote-51) زیاد درگیر نباشد ممکن است بتوانیم این کار را انجام دهیم ولی در صورتی که سیستم درگیر باشد و این زمان بیشتر از 3 میلی ثانیه انجام شود. یک فریم از دوربین را از دست خواهیم داد. پس به همین منظور به سه رشته پردازشی نیاز داریم.

تکنیک های متفاوتی برای هماهنگ سازی رشته های پردازشی وجود دارد من جمله Semafore، mutex، Signals، و البته صف. در این پروژه برای همگام­سازی میان رشته های پردازشی از صف استفاده شد. به این معنا که یک رشته پردازشی وظیفه دریافت اطلاعات از دوربین­ها را بر عهده می­گیرد و با دریافت فریم، آن را به رشته پردازشی ارسال فریم ها به بروکر ارسال می­کند. رشته پردازشی ارسال بسته ها به بروکر بعد از ارسال آن را به رشته پردازشی ذخیره­ساز اطلاعات ارسال می­کند.

نکته ای که در قسمت وجود دارد این است که هنگام ارسال اطلاعات به بروکر باید نوع اطلاعات را به بایت تغییر دهیم و در سمت دیگر مجدد اطلاعات را از بایت به ماتریس اطلاعات تبدیل کنیم. برای اینکه بتوانیم این کار را انجام دهیم نیاز داریم که ابعاد ماتریس را پیش از ارسال در اطلاعات ارسالی کد کنیم. می­دانیم که وقتی اطلاعات را به بایت تبدیل میکنیم اعداد 8 بیتی خواهند شد و بزرگترین عددی که در مبنای 10، برای اعداد 8 بیتی می­توانیم نمایش دهیم عدد 255 است. پس ابعاد تصویر را با کمک یک تابع ساده کد می­کنیم تا در بازه 0 تا 255 قرارگیرند و در بایت اولیه فریم ارسالی قرار می­دهیم و در سمت دیگر بر اساس تابع دیکدر ابعاد تصویر را بازسازی می­کنیم. تابع کد کننده ابعاد تصویر را میتوانیم در شکل ‏3–11 مشاهده کرد.

def coding\_size(x):

    return np.uint8((x[0]/8,x[1]/8))

شکل ‏3–11-تابع کد کننده حجم تصویر

به عنوان مثال اگر ابعاد تصویر 1920 \* 1080 باشد آن را به 240\*135 تبدیل می­کند تا در بازه مذکور واقع شوند. به طور مشابه در سمت گیرنده نیز معکوس این عملیات را انجام رخ خواهد داد. حال در سمت گیرنده کافی است برای تبدیل این رشته بایت اطلاعات دریافتی به ماتریس تصویر دو بایت اول این اطلاعات را برداشته و نحوه خواندن رشته بایت دریافتی را بدست آوریم. کل فرایندی که در فوق شرح دادیم در فایل sender.py از برنامه کاربردی سرور قرار دارد. این فایل یک پایتون است که با اجرای ورودی هایی مشخص که شامل نام کاربری و رمز عبور بروکر پیام، نام مرکز توزیع اطلاعات، که باید اطلاعات را به آن بریزد[[52]](#footnote-52)، آدرس دوربینی که باید اطلاعات را از آن بخواند، را هنگام اجرا دریافت میکند به طور مثال می­توان این فایل را با اجرای دستور زیر اجرا کرد.

python sender.py c1 rtsp://192.168.1.10:554 guest guest

با اجرای این برنامه در صورتی که مرکز توزیع اطلاعات c1 ساخته شده باشد و دوربینی با ادرس فوق در شبکه موجود باشد اطلاعات از دوربین ها خوانده شده، به مرکز توزیع اطلاعات ارسال شده و همچنین در محل مورد نظر ذخیره خواهند شد.

* **طراحی رابط کاربری**

این برنامه از آن جا که توانایی کنترل سیستم را دارد برای امنیت فیزیکی از برنامه برای ورود به آن یک صفحه لاگین ساخته شد. که علاوه بر امنیت فیزیکی سرور وظیفه چک کردن فعال بودن سرویس ها را به عهده دارد. به معنا که اگر سرویس ها توسط داکر فعال نشده باشند، خطایی مبنی بر آن خواهد داد.

در ادامه با فرض ورود به سیستم با برنامه ای رو به رو خواهیم شد که 2 وظیفه اصلی دارد، یکی اضافه کردن کاربر جدید به سیستم و دیگری کنترل دوربین های متصل به سیستم، این کار در 3 تب انجام خواهد شد که در فصل بعد در راهنمای استفاده از برنامه به طور کامل شرح داده خواهد شد. ولی عملکرد کلی به نحوی خواهد بود که در یک بخش کاربر دوربین­های جدید را به مجموعه اضافه کند و هنگام اضافه کردن دوربین وضعیت دوبین ها بررسی شوند و هنگامی که دوربین تعریف شد در بروکر پیام یک مرکز توزیع پیام برای آن ساخته شود تا بتوان اطلاعات را به آن منتقل کرد و همچنین در تب بعد بتوان فرآیند دوربین های فعال که در بالا بررسی نمودیم را برای دوربین های مورد نظر اجرا کند. همچین در فرایند بالا دیدیم که دوربین ها اطلاعات را ذخیره می­کنند پس یک مکانیزم نیز باید برای حذف اطلاعات در این پیاده سازی شود. علاوه بر موارد فوق باید یک مکانیزم نیز برای اضافه کردن کاربران جدید به مجموعه تعریف کنیم. با توجه به توضیحات فوق باید گزینه هایی در رابط کاربری تعریف کنیم که بتوانیم اطلاعات را از کاربر دریافت کنیم که در این قسمت آن ها را بررسی خواهیم نمود.

* کلیدی اضافه کردن دوربین جدید
  + آدرس دوربین
  + نام منحصر به فرد برای دوربین(چون باید با همین نام در سرور برای آن مرکز توزیع اطلاعات تعریف کنیم و همچنین اطلاعات را در پوشه ای با این نام ذخیره کنیم)
  + سطح دسترسی دوربین
* لیست انتخابی برای انتخاب کنترل دوربین های تعریف شده در سرور
* کلید فعال کردن دوربین (فعال کردن فرآیند دوربین فعال برای دوربین انتخاب شده از لیست دوربین های تعریف شده در سرور)
* کلید غیرفعال کردن دوربین (غیر فعال کردن فرآیند دوربین فعال برای دوربین انتخاب شده از لیست دوربین های تعریف شده در سرور)
* کلید حذف دوربین (غیرفعال کردن و حذف دوربین برای دوربین انتخاب شده از لیست دوربین های تعریف شده در سرور)
* لیست انتخابی برای کنترل دوربین هایی که از آن ها در سرور اطلاعات وجود دارد.
* کلید حذف اطلاعات ذخیره شده دوربین (حذف اطلاعات دوربین انتخابی از لیست دوربین هایی که از آن ها در سرور اطلاعات وجود دارد)
* کلید تعریف کاربر جدید
  + نام کاربری
  + رمز عبور
  + سطح دسترسی
* لیست انتخابی از کاربران تعریف شده در سیستم
  + کلید حذف کاربر (حذف کاربر انتخابی از لیست کابران تعریف شده در سیستم)

پس برای طراحی رابط کاربری برنامه کاربردی سرور به ویجت های فوق نیاز داریم تا بتوانیم سیستم را مدیریت کنیم. حال باید برای هر کدام از کلید های تعریف شده فوق عملکردی که باید انجام دهند را تعریف کنیم و به رابط کاربری که توسط نرم افزار QtDesigner طراحی نمودیم متصل کنیم.

حال به صورت نمونه، کدی که برای که برای فشردن کلید "تعریف دوربین جدید" اجرا خواهد شد را بررسی خواهیم نمود در شکل ‏3–12 می­توان این رشته کد را مشاهده نمود. با فشردن کلید اضافه کردن دوربین جدید، ابتدا ورودی کاربر که در نام دوربین خود را تعریف کرده است بررسی می­شود که آیا دوربینی با این نام قبلا در سرور تعریف شده است یا خیر در صورتی که به نام تکراری باشد خطای تغییر نام را به کاربر نمایش خواهد داد. سپس فیلد مربوط به آدرس دوربین را بررسی می­کند که آیا چنین دوربینی در شبکه وجود دارد یا خیر، سپس برای دوربین در سرور یک مرکز توزیع اطلاعات آماده می­کند و یک جایگاه اجرای فرآیند برای آن ایجاد می­کند، و اطلاعات آن را در سرویس redis ذخیره می­کند. به عنوان مثال یکی از فیلدهای آن ac است که فعال یا غیر فعال بودن دوربین را برای سایر برنامه های کاربردی مشخص می­کند. همچنین مرکز توزیع اطلاعاتی که برای این دوربین در نظر گرفته شده است را در redis قرار می­دهد تا سایر برنامه های کاربردی بتوانند به اطلاعات این دوربین دسترسی پیدا کنند.

    def add\_camera(self):

        cam\_name=self.ui.SE\_lineEdit.text()

        if not( cam\_name in self.cam\_handel):

            self.cam\_handel.update({cam\_name:''})

            self.log\_signal.txt.emit('''camera name is valid''')

        else:

            self.log\_signal.txt.emit('''please change your camera name, camera name must be unique.''')

            return

        cam\_ip=self.ui.CIP\_lineEdit.text()

        if cam\_ip=='0':

            cam\_ip=0

        cap = cv.VideoCapture(cam\_ip)

        if not cap.isOpened():

            self.log\_signal

            self.log\_signal.txt.emit("the ip of cammera is invalid")

            return

        else:

            self.log\_signal.txt.emit("camera is valid")

        #free the memory

        del cap

        #get level of access

        camera\_access\_level=self.ui.CamNameComboBox\_3.currentText()

        if create\_exchange(RABBIT\_SERVER\_IP,RABBIT\_PORT,self.username,self.password,"ex\_"+cam\_name):

            pid=-1

            #save every things on data

            self.cam\_handel[cam\_name]=[QProcess(),cam\_ip,camera\_access\_level,pid]

            self.ui.CamNameComboBox.addItem(cam\_name)

            self.log\_signal.txt.emit("camera added inside server successfully")

            self.log\_signal.txt.emit("--------------------------------------")

            self.Redis\_client.hset(cam\_name, "ex", "ex\_"+cam\_name)

            self.Redis\_client.hset(cam\_name, "lv", camera\_access\_level)

            self.Redis\_client.hset(cam\_name, "mn", "jangal\_"+cam\_name)

            self.Redis\_client.hset(cam\_name, "ip", cam\_ip)

            self.Redis\_client.hset(cam\_name, "ac", "F")

        else:

            self.log\_signal.txt.emit("camera is not valid")

شکل ‏3–12- رشته کد افزودن دوربین جدید در برنامه کاربردی سرور

### توسعه برنامه کاربردی سرور پردازشی

اگر بخواهیم این برنامه را مشابه فوق با دیدگاهی کلی بررسی کنیم، باید بگوییم که قصد داریم به تعداد دلخواه الگوریتم پردازشی بر روی دوربین های فعال سیستم اضافه کنیم برای همین باید یک مکانیزم برای اضافه کردن یک لایه پردازشی به دوربین مورد نظر پیاده سازی کنیم و همچنین حذف این لایه پردازشی را در نظر بگیریم. به این معنا که بتوانیم این لایه های پردازشی را فعال یا غیر فعال کنیم. همچنین قصد داریم که برای لایه پردازشی تعیین شده سطح دسترسی معین تعریف شود تا هر کاربر نتواند به اطلاعات پردازش شده دوربین ها دسترسی داشته باشد. همچنین بر اساس چالش پردازش در زمان واقعی که پیشتر بررسی نمودیم، می­دانیم از هر n فریم تنها قرار است یک فریم پردازش شود.

پر واضح است که برای هر لایه پردازشی نیاز داریم که فرایندهایی یکسان، همچون دریافت از بروکر پیام مانند یک کاربر، پردازش 1 فریم از n فریم و ارسال مجدد اطلاعات پردازش شده را به بروکر پیام انجام دهیم. پس نیاز داریم که به ازای هر لایه پردازشی یک فرآیند جدید که 3 وظیفه فوق را انجام می­دهد، اجرا کنیم. پس ابتدا به بررسی فرایندی که لایه های پردازشی فعال در سیستم باید انجام دهند میپردازیم در شکل ‏3–13 میتواند فرایندی که برای هر الگوریتم در سرور پزداشی اجرا شود را مشاهده کنیم. سپس به بررسی پیاده سازی رابط کاربری میپردازیم تا طراحی برنامه کاربردی سرور پردازشی را بررسی کنیم.



شکل ‏3–13-فرآیندی که برای هر لایه پردازشی باید در برنامه کاربردی سرور پردازشی اجرا شود.

* **فرآیند لایه های پردازشی فعال**

مشابه قبل نکته ای که باید در نظر بگیریم این است که آیا سه وظیفه ای که در نظر داریم باید به ترتیب اجرا شود یا باید به صورت موازی اجرا شود. برای پاسخ این سوال باید نکته ای را در نظر بگیریم که تا قبل از اینکه پردازش انجام نشود، نمی­توان اطلاعاتی را به بروکر پیام ارسال نمود. پس این دو رخ داد در پشت یکدیگر قرار دارند این درحالی است که فرایند دریافت اطلاعات از بروکر مستقل از این این امر است. پس در این فرآیند به 2 رشته پردازشی نیاز داریم. یکی برای دریافت اطلاعات و جدا کردن بسته ای که باید پردازش شود. و دیگری رشته پردازشی­ای که وظیفه پردازش بسته ها و ارسال اطلاعات پردازش شده به بروکر پیام را دارد.حال اگر این دو رشته به صورت موازی اجرا نشوند، هنگامی که در حال پردازش بسته ها هستیم دیگر اطلاعات دیگری از سرور دریافت نخواهد شد. و این مطلوب ما نیست. همانند برنامه کاربردی سرور برای همگام­سازی رشته­های پردازشی از صف، میان این دو رشته پردازشی استفاده شد. به این معنا که یک رشته پردازشی وظیفه دریافت اطلاعات از دوربین ها ار بروکر پیام را بر عهده میگیرد و از هر n فریم یک فریم آن را به رشته پردازشی دیگر ارسال می­کند در آن ابتدا فریم دریافتی پردازش می­شود و در ادامه به مرکز توزیع پیام مربوطه که با عنوان لایه پردازشی از آن یاد می­کنیم ارسال شود.

مدل های هوش مصنوعی که در این پروژه برای تست قرار داده شده اند، خروجی هایی عددی دارند که مختصات حاشیه تصاویر را تعیین می­کنند. مدل هایی که در این پروژه قرار گرفت شامل تشخیص چشم، تشخیص صورت، تشخیص صورت و چشم هستند. که همگی آن ها در پوشه model قرار گرفته اند. نحوه اضافه شدن آن ها به این فرآیند هنگام اجرای فرآیند تعیین می­شود. بدین شکل که هنگام اجرای فرآیند یکی از ورودی ها الگوریتم انتخابی است، این انتخاب را می­توان در شکل ‏3–14مشاهده کرد.

اضافه کردن یک الگوریتم جدید به پروژه بسیار ساده است. برای این کار کافی است مدل هوش مصنوعی خود را در پوشه model قرار دهید و نامی که قصد دارید داخل رابط کاربری برای آن نمایش داده شود را بر روی پوشه بگذارید. سپس تابعی با ورودی و خروجی مشخص تعریف شود، که فریم ورودی را دریافت می­کند و مختصات های خروجی را برمیگرداند. نام این تابع را می­بایست get\_object\_position تعریف کنید. سپس خط کامنتی که در رشته کد شکل ‏3–14 مشاهده می­کنید را کامل کنید و اسم فولدری که تعیین کرده اید را در بخش مشخص شده وارد کنید. به طور خلاصه یکی تعریف اسم تابع و تعریف اسم فولدر است. و صد البته که خروجی ای که قرار است ای مدل برگرداند باید از جنس مخصات هایی مربعی باشد، که در یک لیست از تاپل هایی که 4 متخصات مسطیلی دارند قرار گرفته اند را باشد. به عنوان مثال خروجی تابع get\_object\_position میتواند اعدادی مشابه زیر باشد.

[(120,130,150,180),(140,120,170,180),…]

if ALGORITHM =='face':

    from model.face.object\_detection import get\_object\_position

elif ALGORITHM =='eyes':

    from model.eyes.object\_detection  import get\_object\_position

elif ALGORITHM =='face\_eyes':

    from model.face\_eyes.object\_detection  import get\_object\_position

#elif ALGORITHM =='YOUR\_FOLDER\_NAME':

    # from model.'YOUR\_FOLDER\_NAME'.object\_detection  import

شکل ‏3–14-اضافه کردن مدل هوش مصنوعی جدید به پروژه

در صورتی که جسمی در تصویر تشخیص داده شود، یک json به سمت مرکز توزیع داده لایه پردازشی ارسال خواهد شد که محتوی متخصات­های شئ تشخیص داده شده است و در صورتی که جسمی داخل تصویر نباشد یک json با عنوان محتوای خالی به این مرکز توزیع پیام ارسال کرد تا از عدم وجود شئ اطلاع رسانی کند تا در صورتی محیطی را انتخاب شده است، غیر فعال شود.

ورودی هایی که این فرآیند دریافت می­کند شامل موارد زیر است. نام الگوریتم انتخابی کابر که توسط رابط کاربری تعیین می­شود. نام مرکز توزیع اطلاعات ورودی که باید اطلاعات دوربین را از آن دریافت کرد. نام مرکز توزیع اطلاعات خروجی که نتیجه پردازش را باید به آن منتقل شود. نام کاربری و رمز عبور و آدرس سرویس rabbitmq ، میزان n که از هر n فریم یک فریم پردازش شود. برای مثال در صورتی که سرویس ربیت فعال بوده و مرکز توزیع داده c1 فعال باشد و به آن اطلاعات و دوربین ارسال شود و مرکز توزیع اطلاعات c1\_pr نیز تعریف شده باشد و با دستور زیر می­توان این فرآیند را به صورت کامل اجرا کرد و خروجی اطلاعات پردازش شده دوربین c1 را با الگوریتم تشخیص چهره به مرکز توزیع داده c1\_pr منتقل شود.

python sender.py face c1 , c1\_pr 127.0.0.1 15672 guest guest 60

* **طراحی رابط کاربری برنامه کاربردی سرور**

این برنامه از آن جا که توانایی کنترل پردازش اطلاعات را دارد برای امنیت فیزیکی برنامه، برای ورود به آن یک صفحه لاگین ساخته شد تا علاوه بر امنیت فیزیکی سرور وظیفه چک کردن فعال بودن سرویس ها را بر عهده بگیرد. به معنا که اگر سرویس ها توسط داکر فعال نشده باشند، خطا خواهد داد.

در ادامه با فرض ورود به سیستم با برنامه ای رو به رو خواهیم شد که وظیفه اصلی آن کنترل اجرای الگوریتم­های هوش مصنوعی خواهد بود و این کار در 2 تب انجام می­دهد که در فصل بعد در راهنمای استفاده از برنامه به طور کامل شرح خواهیم داد. عملکرد کلی این برنامه به این نحوه خواهد بود که در یک بخش کاربران برای دوربین­های فعال یک لایه پردازشی جدید تعریف کنند و جایگاهی برای اجرای فرآیند برنامه لایه پردازشی ایجاد شود و در بخش دیگر بتوان مدل هوش مصنوعی و تعداد پردازش فریم ها را کنترل کرد و فرآیند برنامه لایه پردازشی را فعال کرد. با توجه به توضیحات فوق باید گزینه هایی در رابط کاربری تعریف کنیم که بتوانیم اطلاعات را از کاربر دریافت کنیم که در این قسمت آن ها را بررسی خواهیم نمود.

* لیست انتخابی از دوربین های فعال
* کلید تعریف لایه پردازشی جدید
  + نام لایه پردازشی (نام مرکز توزیع اطلاعات که اطلاعات پردازش شده به آن ارسال خواهند شد)
  + لیست انتخابی از سطح دسترسی های موجود برای دوربین فعال فعلی
* کلید بازیابی (لیست دوربین های فعال را آپدیت خواهد نمود)
* لیست انتخابی لایه های پردازشی
* لیست انتخابی مدل های هوش مصنوعی موجود
* گام شمار تعداد پردازش فریم(انتخاب این که از n فریم یک فریم را برای پردازش انتخاب کند.)
* کلید فعال سازی فرآیند لایه پردازشی (اجرای برنامه لایه پردازشی)
* کلید غیرفعال سازی فرآیند لایه پردازشی (متوقف کردن برنامه لایه پردازشی)

پس برای طراحی رابط کاربری برنامه کاربردی سرور پردازشی به ویجت های فوق نیاز داریم تا بتوانیم سیستم را مدیریت کنیم. حال باید برای هر کدام از کلید های تعریف شده فوق عملکردی که باید انجام دهند را تعریف کنیم و به رابط کاربری که توسط نرم افزار QtDesigner طراحی نمودیم، متصل کنیم.

حال به صورت نمونه، رشته کدی که برای که برای فشردن کلید "فعال سازی فرآیند لایه پردازشی" اجرا خواهد شد را بررسی خواهیم نمود، این رشته کد را میتوان در شکل ‏3–15 مشاهده نمود.

    def active\_processor\_layer(self):

        processor\_name=self.ui.SelectprocessComboBox.currentText()

        if not processor\_name=="":

            pid=self.process\_handel[processor\_name]['proc']['pid']

            if pid == -1:

                model=self.ui.AlgComboBox.currentText()

                self.process\_handel[processor\_name]['model']=model

                frame\_hop=self.ui.process\_spinBox.value()

                runstr = "python"

                args = ["Sender.py",

                        self.process\_handel[processor\_name]['model'],#algorithm

                        self.process\_handel[processor\_name]['ex'],#input exchange

                        processor\_name,

                        self.server\_username,

                        self.server\_password,

                        self.server\_address,

                        self.server\_port,

                        str(frame\_hop),

                        ]

                process=self.process\_handel[processor\_name]['proc']['mp']

                process.setProgram(runstr)

                process.setArguments(args)

                ok, pid = process.startDetached()

                if ok:

                    self.process\_handel[processor\_name]['proc']['pid']=pid

                    #prepare data for redis and other side of connection

                    update\_redis\_alg(self.Redis\_client,

                                     self.process\_handel[processor\_name]["name"],

                                     model,

                                     processor\_name,#exchange\_name,

                                     "T",

                                     self.process\_handel[processor\_name]["lv"](

                    self.send\_log('starting send meta-data to server')

            else:

                self.send\_log('its already activated')

شکل ‏3–15- رشته کد فعال کردن لایه پردازشی جدید در برنامه کاربردی سرور پردازشی

با فشردن کلید فعال سازی فرآیند لایه پردازشی، ابتدا لایه پردازشی انتخاب شده را از لیست لایه های پردازشی ایجاد شده دریافت می­کنیم و در صورتی که آیتمی تعریف شده بود، فرآیندی که در مرحله قبل برای آن ایجاد شده است را بررسی می­کنیم و در صورتی که این فرآیند فعال نشده باشد پارامترهای ورودی آن را تنظیم کرده و فایل sender.py را با ورودی هایی که برای آن در نظر گرفته­ایم اجرا خواهیم کرد. و وضعیت آن را به حالت فعال شده در می­آوریم سپس به سرویس redis اضافه می­کنیم که این مدل هوش مصنوعی برای این دوربین با این سطح دسترسی در کامپیوتر سرور پردازشی تعریف شده است. همچنین مرکز توزیع اطلاعاتی که برنامه کاربردی کاربر باید برای دریافت این اطلاعات پردازش شده این دوربین باید به آن متصل شود را بدست آورد.

### توسعه برنامه کاربردی کاربر

اگر بخواهیم این برنامه را نیز مشابه دو برنامه فوق با دیدگاهی کلی بررسی کنیم، باید بگوییم در این برنامه کاربردی قصد داریم به اطلاعات دوربین ها و الگوریتم های پردازشی فعالی که وجود دارند دسترسی پیدا کنیم و همچنین اطلاعات ذخیره شده را از سرور دریافت و به نمایش بگذاریم. برای اجرای این برنامه نیاز داریم تا از کاربر آدرس سرور و نام کاربری و رمز عبور را دریافت کنیم. بر اساس نام کاربری و رمزعبوری که در برنامه کاربردی سرور از پیش تعیین شده سطح دسترسی آن را تشخیص داده و دوربین های فعال شده ای که برای دسترسی به آن ها مجاز است را به کاربر نمایش دهیم.

برنامه کاربردی کاربر دو وظیفه اصلی دارد؛ یکی بازپخش اطلاعات ذخیره شده و دیگری نمایش زنده اطلاعات دوربین ها است. پر واضح است که برای طراحی بازپخش اطلاعات و پخش زنده اطلاعات نیاز به دو رابط کاربری مجزا داریم. که هر کدام از آنها فرآیندی مجزا خواهند بود که توسط برنامه کاربردی اجرا خواهند شد. چرا که میخواهیم چندین پنجره پخش زنده یا چندیدن پنجره بازپخش اطلاعات را به طور همزمان اجرا کنیم. در شکل ‏3–16 می­توان نحوه چینش فرآیندها را مشاهده نمود. پس ابتدا به بررسی هر کدام از فرآیند های پخش زنده ، باز پخش اطلاعات می­پردازیم و ویجت هایی که برای طراحی رابط کاربری هر کدام از آنها نیاز داریم را معرفی خواهیم کرد.



شکل ‏3–16- فرآیندهایی که برای هر دوربین باید در برنامه کاربردی کاربر اجرا شود.

* **فرآیند بازپخش اطلاعات**

در فرآیند بازپخش اطلاعات، نیاز داریم تا از طریق سرویس MinIO به اطلاعات ذخیره شده دوربین­ها دسترسی پیدا کنیم و و لیستی از اطلاعاتی که برای دوربین مورد نظر در سرور وجود دارد را بدست آوریم. اطلاعات را به کاربر نمایش دهیم و زمانی که قصد بازپخش آن را دارد، زمان مربوطه را انتخاب کرده و آن را از سرور دریافت کند و به نمایش بگذارد. برای پیاده­سازی این بخش سه رشته پرداشی موازی نیاز داریم. یک رشته پردازشی برای نمایش اطلاعات دریافت شده، یک رشته پردازشی برای دریافت اطلاعات از سرور و همچنین یک رشته پردازشی برای کنترل رابط گرافیکی که فرایند ها را کنترل نماید. رشته پردازشی دریافت اطلاعات وظیفه دارد که اطلاعات را از سرور دریافت کند. رشته پردازشی نمایشگر وظیفه دارد اطلاعات پردازش شده را به کاربر نمایش دهد و رشته پردازشی کنترل کننده وظیفه دارد دستورات کاربر را به اجرا در آورد.

لیست ویجت هایی که در رابط کاربری بازپخش اطلاعات به آن ها نیاز داریم به شرح زیر است.

* یک کلید بازیابی که اطلاعات ذخیره شده سرور را بروزرسانی کند.
* یک لیست انتخابی از فایل های ذخیره شده دوربین بر روی سرور.
* یک کلید دریافت که گزینه انتخابی لیست اطلاعات موجود را به سیستم کاربر منتقل کند.
* یک لیست انتخابی از فایل هایی که از سرور بر روی سیستم کاربر منتقل شده.
* یک کلید حذف که فایل انتخابی از لیست اطلاعات موجود بر روی سیستم کاربر را حذف کند.
* یک کلید پخش که فایل انتخابی از لیست اطلاعات موجود بر روی سیستم کاربر را پخش کند.
* یک صفحه نمایش که فایل انتخابی از لیست اطلاعات موجود بر روی سیستم کاربر را نمایش دهد.

اطلاعات ذخیره شده در این برنامه در پوشه­ای داخل برنامه کاربردی ذخیره خواهند شد تا کاربر بتواند با هر پخش­کننده تصویری که علاقه داشت، اطلاعات را دریافت کند. این اطلاعات در پوشه /download قرار گرفته خواهند شد.

* **فرآیند پخش زنده اطلاعات**

در فرآیند پخش زنده اطلاعات می­بایست اطلاعات دوربین ها را به همراه اطلاعات پردازش شده ای که از سمت سرور می­آید به کاربر نمایش دهیم. نکته­ای که باید در نظر بگیریم این است سه وظیفه ای که در نظر داریم باید به صورت موازی اجرا شود. پس به سه رشته پردازشی نیاز داریم. یک رشته پردازشی برای دریافت جریان اطلاعات دوربین ها، یک رشته پردازشی برای دریافت اطلاعات پردازش شده دوبین ها، یک رشته پردازشی برای نمایش اطلاعات دوربین ها نیاز داریم. برای هماهنگ سازی رشته های پردازشی برخلاف قبل که از صف استفاده شد، در اینجا از مکانیزم سیگنال که Pyqt در اختیار ما قرار می­دهد، استفاده می­کنیم. چرا که نسبت به مکانیزم صفی که پایتون در اختیار ما قرار میدهد از سرعت بالاتری بر خوردار است. ولی ممکن است به علت سرعت بالای سیستم برخی از بسته ها که با فاصله زمانی کمتری دریافت میشوند نمایش داده نشوند. که مطلوب ما فرآیند ها داخل سرور است که هیچ چیزی از بین نرود و عدم نمایش یک فریم به علت سرعت بالا در سمت کاربر مسئله خاصی ندارد و مطلوب ماست.

رشته پردازشی دریافت اطلاعات دوربین به بروکر پیام متصل شده و اطلاعاتی که توسط برنامه کاربردی سرور ارسال می­شود را دریافت می­کند و سپس توسط سیگنال به رشته پردازشی رابط کاربری ارسال میکند و رابط کاربری آن را به نمایش می­گذارد، ولی پیش از به نمایش گذاشتن اطلاعات بررسی می­کند که آیا اطلاعاتی برای نمایش توسط مستطیل های تخشیص اشیا وجود دارد یا خیر در صورتی که فعال شده باشد به دور لیست مختصات هایی که برای آن ارسال شده مستطیل هایی اضافه خواهد کرد. برای تشریح بهتر این مسئله باید عملکرد رشته پردازشی بعدی را نیز شرح داده شود.

علاوه بر موارد فوق در نظر بگیرید که یک رشته پردازشی نیز به بروکر پیام متصل شده و اطلاعات پردازش شده ای که توسط برنامه کاربردی سرور پردازشی به بروکر ارسال می­شود را دریافت می­کند. از بخش­های پیش دانستیم که اطلاعات دریافتی یک لیست از مختصات هایی است که اشیائی که در شکل تشخیص داده شده اند، تشکیل شده است، پس با دریافت این اطلاعات پارامترهایی که در رشته پردازشی رابط کاربری هستند به روزرسانی خواهند شد و عملکرد سیستم کامل می­شود.

مسئله ای که هنوز بحث نکردیم نحوه کنترل نمایش این الگوریتم هاست و این که اگر الگوریتم جدیدی برای این دوربین بر روی سرور فعال شود برنامه کاربر به چه نحو مطلع خواهد شد. برای این کار بهتر است ابتدا ویجت هایی که نیاز داریم برای فرآیند پخش زنده اطلاعات استفاده کنیم بررسی کنیم. برای طراحی فرآیند پخش زنده به ویجت های زیر احتیاج داریم:

* یک پنجره نمایش اطلاعات دوربین
* یک لیست انتخابی از مدل های هوش­مصنوعی فعال بر روی سرور پردازشی.
* یک کلید بروزرسانی که لیست انتخابی مدل های هوش مصنوعی فعال را بروزرسانی کند.
* یک کلید فعال سازی که رشته پردازشی دریافت اطلاعات پردازش شده دوربین ها را بر اساس لیست انتخابی فعال کند.

توجه کنید که برای غیر فعالسازی و عدم نمایش اطلاعات پردازش شده در لیست انتخابی یک گزینه تحت عنوان None قرار گرفت تا در صورتی که قصد حذف نمایش اطلاعات پردازش شده را داشتید. با فعال نمودن این گزینه غیرفعال شود و برای به روز رسانی الگوریتم های هوش­مصنوعی فعال شده برای دوربین مشخص توسط کلید بازیابی انجام می­شود.

* **طراحی رابط کاربری برنامه کاربردی کاربر**

این برنامه از آن جا که دسترسی به اطلاعات دوربین ها را برای کاربر ممکن می­کند، و کاربران باید بر اساس سطح دسترسی خود به اطلاعات دوربین ها دست پیدا کنند. این برنامه کاربردی به یک صفحه لاگین نیاز دارد تا دوربین هایی که در سطح دسترسی مجاز کاربر هستند را به وی نمایش دهد.

در ادامه با فرض ورود به سیستم با برنامه ای رو به رو خواهیم شد که وظیفه کنترل دو فرآیند پخش زنده و بازپخش را بر عهده دارد، این کار را در دو تب مجزا انجام می­دهد. در فصل بعد در راهنمای استفاده از برنامه به طور کامل شرح داده خواهد شد. در بخش مربوط به پخش زنده کاربر باید بتواند تا به دوربین های فعال دسترسی داشته باشد و بتواند فرآیند پخش زنده را برای دوربین فعال اجرا کند. همچنین در بخش بازپخش اطلاعات دوربین­ها کاربر باید توانایی دسترسی به اطلاعات دوربین­هایی که از آنها در سرور اطلاعات ذخیره شده است، را بدست آورد. علاوه بر موارد فوق در صورتی که یک دوربین جدید به مجموعه اضافه یا حذف شود. یا اطلاعات جدید در سرور ذخیره شود یا از سرور حذف شود، برنامه کاربردی کاربر باید بتواند بدون باز و بسته کردن برنامه از تغییرات آن مطلع شود. برای نیاز­های فوق به یک رابط کاربری با ویجت های زیر نیازداریم.

* کلید بازیابی (لیست دوربین ها را به روز خواهد شد)
* لیست انتخابی دوربین های فعال
* کلید فعال سازی فرآیند پخش زنده اطلاعات
* لیست انتخابی دوربین هایی که از آنها اطلاعات روی سرور ذخیره شده است.
* کلید فعال سازی فرآیند بازپخش اطلاعات

پس برای طراحی رابط کاربری برنامه کاربردی سرور پردازشی به ویجت های فوق نیاز داریم تا بتوانیم سیستم را مدیریت کنیم. حال باید برای هر کدام از کلید های تعریف شده فوق عملکردی که باید انجام دهند را تعریف کنیم و به رابط کاربری که توسط نرم افزار QtDesigner طراحی نمودیم متصل کنیم.

### جمع بندی

در این فصل ابتدا به طراحی معماری پرداختیم و سعی کردیم وظیفه ای که هر سرویس در معماری ایجاد می­کنند و این که با چه چالش­هایی در طراحی سیستم مواجه هستیم، بپردازیم. در ادامه اجزای ساخت معماری که برنامه های کاربردی سرور، برنامه کاربردی سرور پردازشی و برنامه کاربردی کاربر هستند تشریح شد و فرآیند ها رشته های پردازشی و شرح وظایفی که در هر کدام از آن ها اجرا می­شوند، را بررسی نمودیم و سعی کردیم بخش های مختلف را تشریح کنیم. در فصل بعد به نحوه راه اندازی سیستم و نحوه کارکرد با این 3 برنامه خواهیم پرداخت و راهنمایی برای بکارگیری سیستم در واقعیت معرفی خواهیم نمود.

# فصل چهارم بکارگیری و نحوه کارکرد سیستم برای کاربران و مدیر سیستم

**بکارگیری و نحوه کارکرد سیستم برای کاربران و مدیر سیستم**

برای اینکه سیستم را بتوانیم در عمل بکار بگیریم، باید پیش­نیازی هایی را آماده کنیم و سیستم را برای بکارگیری در عمل تنظیم کنیم در این فصل ابتدا آماده­سازی زیرساخت های پروژه و نصب پیش­نیازها را توضیح خواهیم داد و در ادامه به آموزش تنظیمات بخش های مختلف نرم افزار های پروژه که شامل برنامه کاربردی سرور ، برنامه کاربردی سرور پردازشی و برنامه کاربردی کاربر هستند میپردازیم.

## راه اندازی و نصب پیشنیاز ها

برای اجرای این پروژه به چند نرم افزار پایه نیاز داریم. که در ادامه به نصب آن ها میپردازیم و سرویس ها را فعال خواهیم نمود. برای اجرای این پروژه به اولین چیزی که نیاز داریم نرم افزار داکر است. که می­بایست بر روی کامپیوتر سرور نصب شود، نحوه این نرم افزار برای سرورهای ویندوزی و لینوکسی متفاوت است که هر دوی آن ها را بررسی خواهیم نمود. همچنین برای راه اندازی این پروژه به پایتون نیز نیاز داریم که در سرورهای لینوکسی به صورت پیشفرض وجود دارد ولی برای سرورهای ویندوزی به نصب آن نیاز داریم.

### نصب داکر بر روی ویندوز

برای این کار به تعدادی پیشنیاز احتیاج داریم که باید انجام شود. به طور کلی 2 شیوه برای راه­اندازی داکر در ویندوز وجود دارد که نحوه اجرای کانتینرها بر روی سخت افزار موجود را مشخص می­کند، روش اول، hyper-v backend است، که کانتینر ها توسط خود ویندوز کنترل خواهند شد. روش دوم استفاده از WSL است که کوتاه شده windows subsystem for linux است، بدین شکل که در داخل ویندوز به ماشین مجازی ای از جنس لینوکس اجرا شده و کانتینر ها در آن محیط کنترل خواهند شد. در روش دوم باید بر اساس [15] فعال سازی شود. در هر دو روش در بایوس دستگاه باید امکان شبیه­سازی[[53]](#footnote-53)فعال شده باشد و حافظه رم حداقل 4 گیگابایت و یک پردازنده 64 بیتی موجود باشد. کانتینرها و ایمیج های ایجاد شده با Docker Desktop بین تمام حساب های کاربری در ماشین­هایی که در آن نصب شده است به اشتراک گذاشته می شود. این به این دلیل است که همه حساب‌های ویندوز از یک ماشین مجازی برای ساخت و اجرای کانتینرها استفاده می‌کنند. توجه داشته باشید که هنگام استفاده از WSL امکان اشتراک گذاری کانتینرها و ایمیج ها بین حساب های کاربری وجود نداشته باشد. در ادامه باید نرم افزار Docker desktop دانلود و نصب شود. [16]

### نصب داکر بر روی لینوکس

داکر را می­توان بر روی اکثر توزیع های لینوکس نصب نمود ولی برای نمونه، نصب داکر بر روی توزیع ubuntu شرح داده می­شود. برای اینکار تنها باید به یک پردازنده 64 بیتی و یک سیستم عامل به روز نیاز دارید. بهتر است پیش از ادامه نسخه قبلی داکر را توسط دستور زیر حذف نمایید.

$ sudo apt-get remove docker docker-engine docker.io containerd runc

برای نصب داکر در ابونتو، دو روش وجود دارد، روش اول، نصب از طریق ریپازیتوری­ها[[54]](#footnote-54) است، روش دوم گرفتن پکیج دبین آن و نصب به صورت دستی است که عموما برای کامپیوترهایی که به اینترنت دسترسی ندارند استفاده می­شود. در این بخش تنها به نصب از طریق روش اول بسنده می­کنیم. برای اینکار ابتدا پیکج apt را به روز رسانی میکنیم و پیشنیاز هایی که برای دریافت فایل نیاز دارد را ازطریق دستور زیر حل خواهیم کرد.

$ sudo apt-get update

$ sudo apt-get install \

ca-certificates \

curl \

gnupg \

lsb-release

در ادامه کلید PGP داکر را دریافت و از طریق دستور زیر تنظیم می­کنیم.

$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

و با دستور زیر یک محیط برای نصب docker engin فراهم می­کنیم. در هنگام تنظیم می­توانیم به جای کلید واژه stable از کلید واژه های nightly یا test استفاده کنیم، که برای تنظیم و انتخاب نسخه مورد دانلود است، نسخه night نسخه­ای است که هنوز منتشر نشده و روی آن کار می­کنند است.

$echo \

"deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu \

$(lsb\_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

حال با دستور زیر آخرین نسخه از docker-ce را دریافت می­کنیم.

$ sudo apt-get install docker-ce docker-ce-cli containerd.io

در ادامه با دستور زیر لیست نسخه هایی که میتوانیم نصب کنیم را بدست می­آوریم.

$ apt-cache madison docker-ce

و از نسخه بین نسخه های موجود، آن نسخه ای که برای سیستم عامل و توزیع لینوکس کامپیوتر میزبان مناسب است را انتخاب میکنیم و در پایان نصب docker-engine را نصب می­کنیم.

$ sudo apt-get install docker-ce=<VERSION\_STRING> docker-ce-cli=<VERSION\_STRING> containerd.io

### نصب پایتون

برنامه های کاربردی در این پروژه تماما با پایتون 3.8 توسعه داده شده است، از آن جایی که در ویندوز این برنامه به صورت پیش­فرض وجود ندارد، می­بایست آن را نصب نماییم. برای اینکار به [17] رفته و فایل مربوطه دانلود و نصب می­کنیم.

### نصب کتابخانه ها و ایمیج های استاندارد

1-برای این کار به دایرکتوری اصلی پروژه که شامل محتوای زیر است بروید.

‎├── Client application

├── Data

├── Server

├── Server\_process layer

├── docker-compose.yml

├── env

├── rbmq

├── redis

└── requirements.txt

2-دستور زیر را در خط فرمان وارد کنید تا کتابخانه های مورد نیاز پایتون نصب شوند.

pip install requirements.txt

3- دستور زیر را در خط فرمان وارد کنید تا ایمیج های مورد نیاز داکر دریافت شوند.

docker-compose build

## راهنمای استفاده کاربران

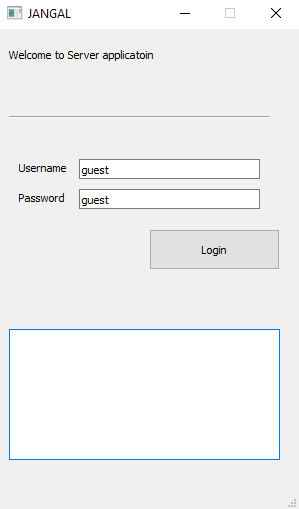
در بخش قبل پیش نیاز های پروژه را دریافت و نصب نمودیم حال وقت آن رسیده که سرویس ها را فعال کرده و یک از برنامه ها را اجرا کنیم. در این بخش راهنمای استفاده از هر یک از برنامه کاربردی سرور، برنامه کاربردی سرور پردازشی و برنامه کاربردی کاربر را بررسی خواهیم نمود.

### راهنمای راه اندازی برنامه کاربردی سرور

پیش از آن که برنامه کاربردی سرور را اجرا کنیم نیاز داریم که سرویس­های مورد استفاده پروژه را که در بخش قبل دریافت نمودیم فعال کنیم. برای اینکار کافی است در مسیر اصلی پروژه دستور زیر را وارد نماییم.

docker-compose up

با بالا آمدن سرویس­های پروژه تنها کافی است برنامه کاربردی سرور را اجرا کنیم، برای اینکار وارد پوشه سرور شده و برنامه server.py را با دستور زیر به اجرا در بیاورید.

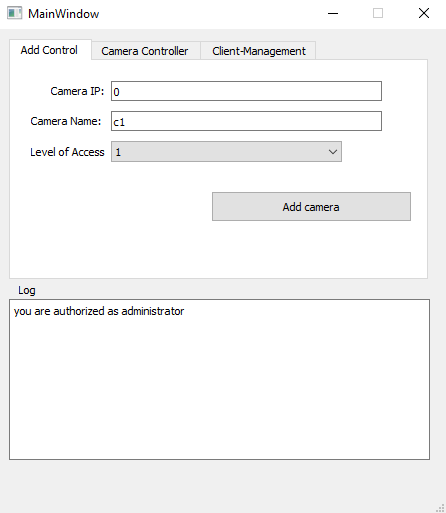
python server.py

با اجرای این دستور فوق برنامه گرافیکی که در شکل ‏4–1 مشاهده می­کنید، به اجرا در خواهد آمد. در این صفحه از شما رمز عبور و نام کاربری سرور را می­خواهد. این نام کاربری و رمز عبور در حقیقت، شناسه کاربری کاربر مدیرسیستم در سرویس rabbitmq است. که با وارد شدن در پنل وب این سرویس و می­توان آن را تغییر داد. (جزئیات بیشتر در پیوست آمده است.)

این نام کاربری و رمز عبور به صورت پیش­فرض guest و guest است. در ادامه با فشردن کلید login میتوان وارد این نرم افزار شد. خطا هایی که این نرم افزار می­تواند بدهد در پایین صفحه نوشته خواهد شد که به شرح زیر است: خطای ‏authorizatoin fail -->not\_authorized‎‏ به معنای آن است که شناسه کاربری یا رمز عبور اشتباه است.‏

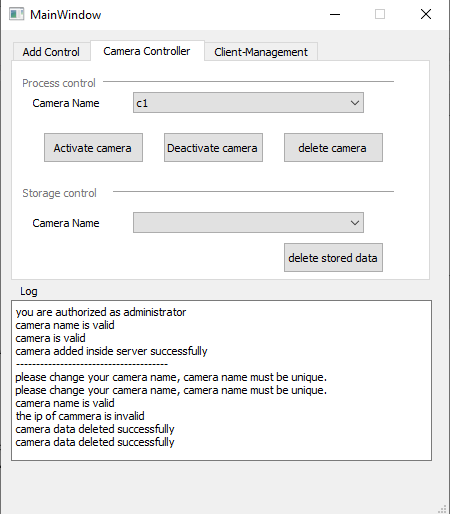
شکل ‏4–1-صفحه ورود برنامه کاربردی سرور

خطای ‏authorizatoin fail -->offline‎‏ به معنای آن است که نرم افزار داکر اجرا نشده است.

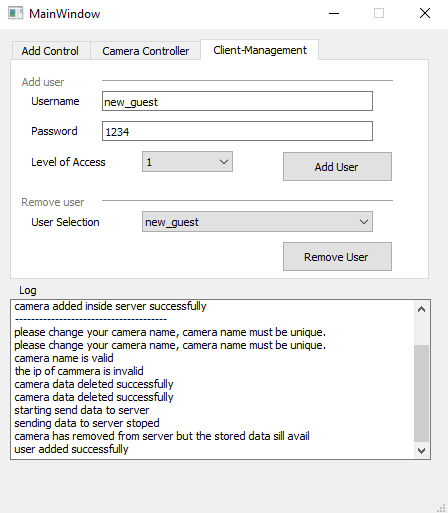
با وارد شدن به برنامه کاربردی سرور، صفحه لاگین بسته شده و صفحه ای که در شکل ‏4–2 مشاهده می­کنید رو به رو خواهید شد.‏ این نرم افزار از چند تب‏ تشکیل شده است. که شامل تب اضافه کردن دوربین جدید به سیستم، تب کنترل کردن دوربین های سیستم، تب کنترل کاربران سیستم است.

با تب اضافه کردن دوربین جدید به سیستم شروع می­کنیم. در خط اول لینک ‏rtsp‏ دوربین مربوطه را وارد شود. در خط دوم نامی منحصر به فرد برای دوربین تعیین شود. سطح دسترسی به دوربین مربوطه را مشخص شود. کلید اضافه کردن دوربین جدید فشرده شود. با فشردن این کلید، یک مرکز توزیع داده بر اساس نامی که برای دوربین انتخاب کرده اید در بروکر تعریف می­شود و بستری برای اجرای فرآیندی که در فصل پیش صحبت کردیم فراهم می­شود. حال در این میان خطاهایی می­توانند ایجاد شوند. خطای ‏please change your camera name, camera name must be unique‏ به ‏معنای آن است که نام مورد نظر استفاده شده و باید نام دیگری برای دوربین انتخاب کنید.‏ خطای ‏the ip of cammera is invalid‏ به معنی این است که لینک دسترسی به دوربین اشتباه ‏وارد شده است.‏ در صورت صحیح بودن همه فیلد ها ‏camera added inside server successfully‏ چاپ می­شود.

شکل ‏4–2- تب اول از برنامه کاربردی سرور-اضافه کردن دوربین جدید

حال به سراغ تب دوم خواهیم رفت، برای این کار کافی است روی آن کلیک کنیم در شکل ‏4–3 می­توانیم محتوای تب دوم یعنی کنترل دوربین ها را مشاهده کنیم. این تب از دو بخش اصلی تشکیل شده است، بخش اول کنترل فرآیند دوربین­ها است و بخش دوم، کنترل اطلاعات ذخیره شده است. در بخش اول یک لیست انتخابی داریم که در آن دوربین هایی که در تب اول تعریف شده اند قرار میگیرند، با انتخاب دوربین مورد نظر می­توانیم هر یک از سه کلید زیر آن را بفشاریم. کلید اول از سمت چپ برای فعال کردن فرآیند دریافت اطلاعات از دوربین ، ارسال به سرور و ذخیره اطلاعات است. کلید دوم برای غیر فعال کردن فرآیند فعال شده است. کلید سوم برای حذف دوربین است. در صورتی که دوربین انتخابی را حذف کنید باید مجدد در تب اول آن را تعریف کنید. با فشردن این کلید اطلاعات ذخیره شده دوربین حذف نخواهند شد. و تنها جایگاه فرآیند آن حذف می­شود. این بدان معناست که میتوان مجددا دوربینی با این نام در صفحه قبل ایجاد کرد و هنگام ساخت دوربین جدید با خطای please change your camera name, camera name must be unique مواجه نخواهید شد. با فشردن کلید اول در صورتی که موفقیت آمیز باشد نوشتار starting send data to server چاپ خواهد شد. با فشردن کلید دوم نوشتار sending data to server stoped‏ چاپ خواهد شد. در بخش دوم این تب یعنی کنترل اطلاعات ذخیره شده بر روی سرور، می­توان اطلاعاتی که از دوربین ها روی سرور ذخیره شده اند را حذف نمود. بدین شکل در لیست انتخابی دوربین هایی که از آن ها در سرور اطلاعات وجود دارد قرار خواهند گرفت، با انتخاب دوربین مورد نظر و فشردن کلید حذف اطلاعات می­توان این اطلاعات را حذف نمود. اطلاعات ذخیره شده در سرور در پوشه Data در مسیر اصلی پروژه قرار دارند، که می­توان به صورت مستقیم به آن اطلاعات دسترسی پیدا کرد. این پوشه در حقیقت به پوشه data که داخل کانتینر MinIO قرار گرفته است، متصل شده و با تغییر(حذف و اضافه کردن) محتوای این پوشه محتوای داخل کانتینر تغییر می­کند. پس کلید حذف اطلاعات در حقیقت محتوای دوربین انتخابی از این پوشه را پاک می­کند.

شکل ‏4–3- تب دوم از برنامه کاربردی سرور- کنترل دوربین­ها

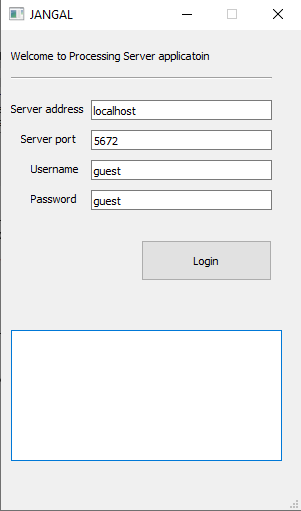
حال به سراغ تب سوم یعنی، تب کنترل کاربران می­رویم، این تب را می­توان در فشردن نام آن باز کرد و در شکل ‏4–4 می­توانیم محتوای آن را مشاهده کنیم. در این تب می­توان کاربران جدیدی در سیستم ثبت نمود. که به اطلاعات دوربین ها ‏دسترسی پیدا کنند. این بخش از دو بخش اضافه کردن کاربر جدید و حذف کاربر موجود تشکیل شده است.‏ در بخش اضافه کردن کاربر جدید می­توان شناسه کاربری و رمز عبور را برای کاربر تعیین نمود و سطح ‏دسترسی آن کاربر را از 1 تا 4 مشخص نمود و کلید اضافه کردن کاربر جدید را فشرد با این کار در redis در بخش acl یک کاربر جدید تعریف می­شود که، برنامه کاربردی کاربر می­تواند با این نام کاربری و رمز عبور بر اساس سطح دسترسی­ای که برایش تعریف شد وارد برنامه شود. در بخش حذف کاربر نیز می­توان کاربران اضافه شده به سیستم را از طریق ‏لیست انتخابی از کاربران فعال‏ انتخاب کردو و با ‏فشردن کلید حذف کاربر آن ها را‏ از سیستم حذف نمود.‏ در صورتی که کاربر با موفقیت به سیستم اضافه شود، user added successfully چاپ خواهد شد و در صورت حذف کاربر از سیستم user removed successfully چاپ خواهد شد.

شکل ‏4–4-تب سوم از برنامه کاربردی سرور-کنترل کاربران

### راهنمای راه اندازی برنامه کاربردی سرور پردازشی

این برنامه را می­توان بر روی کامپیوتری مستقل از کامپیوتر سرور اصلی اجرا کرد. این سیستم تنها نیاز به پایتون دارد. و برای اجرای آن روی یک سیستم دیگر کافی است که کتابخانه های پایتونی که در فایل ‏requaetment.txt‏ قرار گرفته اند را نصب کرد. برای اجرای این برنامه کافی است که پوشه ‏serverprovcessor ‎‏ را به سیستم مد نظر منتقل نمود و وارد آن شد و با اجرای دستور زیر این برنامه را اجرا کرد.

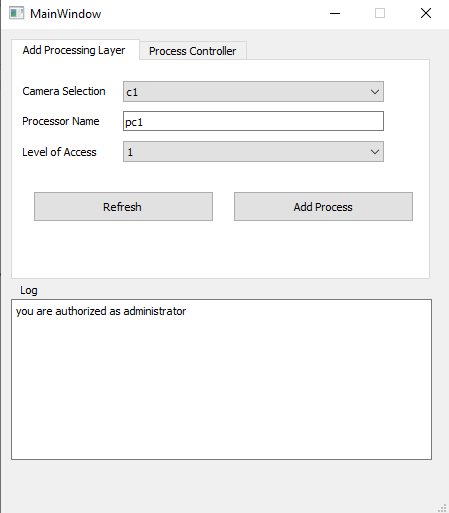
python server.py

در شکل ‏4–5 می­توان صفحه لاگین برنامه کاربردی سرور پردازشی را مشاهده کرد. که پس اجرای این برنامه نمایش داده خواهد شد. در خط اول این برنامه باید آدرس ip کامیپوتری که برنامه کاربردی سرور در آن اجرا شده است را وارد کنید. در صورتی که این برنامه را بر روی همان سیستم اجرا میکنید کافی است که ip، 127.0.0.1 یا عبارت localhost را وارد کنید

در خط بعدی پورت مربوط به سرویس rabbit را وارد کنید، که به صورت دیفالت 5672است. مگر اینکه در داخل داکر کامپوز تغییر یابد یا بر اساس تنظیمات شبکه پورت مذکور بر روی پورت دیگری منتقل شود. نام کاربری و رمز عبور نیز، مشابه برنامه کاربردی سرور guest , guest ، یعنی کاربر پیش فرض بروکر است. با تنظیم مقادیر فوق می­توان کلید ورود را فشرد و وارد نرم افزار سرور پردازشی شد. در صورتی که آدرس سرور به درستی وارد نشود با خطای rabbit\_not\_found مواجه خواهید شد. یا در صورت اشتباه بودن نام کاربری و رمز عبور با خطای authorizatoin fail -->not\_authorized رو به رو خواهید شد.

شکل ‏4–5-صفحه لاگین برنامه کاربردی سرور پردازشی

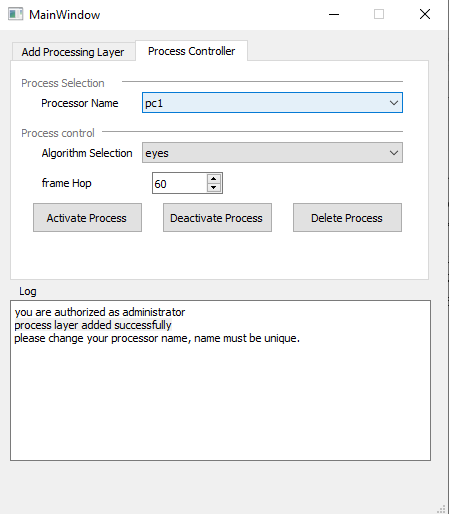
در صورتی که بدون خطا وارد سیستم شوید عبارت you are authorized as administrator چاپ خواهد شد و این صفحه بسته شده و صفحه شکل ‏4–6 نمایان خواهد شد. این برنامه از دو تب تشکیل شده که فرآیند پردازش بسته ها را به عهده دارند. تب اول برای اضافه کردن لایه پردازشی جدید و تب دوم برای کنترل لایه پردازشی اضافه شده است. ابتدا به سراغ تب اول یعنی اضافه کردن لایه پردازشی جدید می­رویم.

در این تب در لیست انتخابی اول می­توان به دوبین­های فعال شده بر روی برنامه کاربردی سرور دسترسی پیدا کرد، در پایین این صفحه یک کلید بازیابی وجود دارد تا این برنامه از دوربین های فعال شده جدید روی سیستم مطلع شوند. ‏پیشنهاد می­شود قبل از اضافه کردن لایه پردازشی جدید ابتدا کلید ‏refresh‏ را فشار دهید تا از آخرین وضعیت سرور ‏اطلاع کسب کنید.

در ادامه نیاز است که نامی منحصر به فرد برای لایه پرداشی مد نظر اعمال کنید و در آخرین گزینه از لیست انتخابی موجود ‏سطح دست رسی برای پردازش را انتخاب شود و سپس بر روی کلید اضافه کردن لایه پردازشی فشار دهید. لازم به ذکر است که در صورتی که سطح دست رسی یک دوربین بالاتر باشد نمی­توان لایه پردازشی را برای ‏سطوح پایین تر انتخاب نمود. به عنوان مثال اگر سطح دسترسی یک دوربین 3 باشد نمی توان سطح دسترسی 1 و 2 را ‏برای آن اعمال نمود[[55]](#footnote-55). با فشردن کلید اضافه کردن لایه پردازشی جدید خطای زیر ممکن است رخ دهند. خطای please change your processor name, name must be unique.‎‏ خطای یکسان بودن نام لایه پردازشی است و کاربر باید نام لایه پردازشی را تغییر دهد. و در صورتی که عملیات اضافه کردن لایه موفقیت انجام شود process layer added successfully‏ چاپ خواهد شد. با اینکار یک مرکز توزیع داده با نامی که در نظر میگیرید بر روی بروکر پیام آماده خواهد شد تا اطلاعات پردازش شده به آن منتقل شود.

شکل ‏4–6-تب اول برنامه کاربردی سرور پردازشی – اضافه کردن لایه پردازشی جدید

حال به سراغ تب بعدی، یعنی تب کنترل لایه پردازشی می­رویم نمای کلی این تب را می­توان در شکل زیر مشاهده نمود.‏‎ ‎ برای وارد شدن به تب کافی است بر روی نام کلیک کنید.

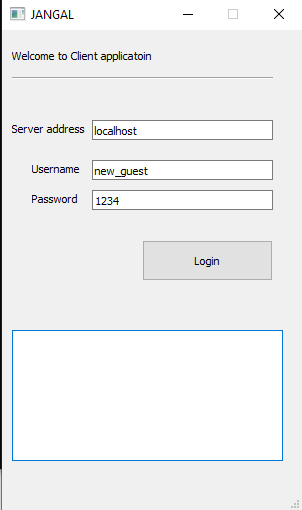
این تب از دو بخش تشکیل شده است. بخش اول انتخاب لایه پردازشی مورد نظر است که در تب اول تعریف شد و ‏بخش دوم امر کنترل آن است که می­توان مدل هوش مصنوعی مورد نظر را انتخاب نمود.‏ گزینه بعدی یعنی پرش فریم،‏ کنترل تعداد فریم هایی است که قرار است پردازش شوند و انتخاب این گزینه به این معنا است که از هرn فریم، تنها یک فریم ‏پردازش می­شود. که بسته به مدل هوش مصنوعی و زمان پردازش هر فریم آن باید کنترل شود. چرا که زمان اجرای هر ‏الگوریتم هوش مصنوعی بر روی یک فریم از تصویر با الگوریتم دیگری متفاوت است و بستگی به سیستم در حال ‏محاسبه و تعداد پردازش هایی که بر روی آن سیستم دارد.‏ به طور کلی دوربین ها با 30 فریم بر ثانیه تصویر برداری میکنند. (در تنظیمات داخل دوربین قابل تعیین است) ‏پس می­توان گفت وقتی این پارامتر را بر روی 60 تعیین می­کنیم از هر 60 فریم یک فریم پردازش می­شود، 60 فریم در این مثال به معنای دو ثانیه است، که یعنی هر دو ثانیه یک فریم پردازش شود. برای شروع فرآیند پردازش نیز با فشردن کلید ‏فعال­سازی فرآیند‏ می­توان این لایه را فعال نمود. با فعال نمودن این لایه کاربر قادر خواهد بود به ‏اطلاعات پردازش شده دوربین دسترسی پیدا کند و با فشردن کلید ‏غیرفعال­سازی‏ فرآیند، فرآیند پردازش متوقف ‏خواهد شد. همچنین با فشردن کلید حذف‏ لایه پردازشی، این لایه حذف شده و نام آن آزاد می­شود و مرکز توزیع داده حذف می­شود و جایگاه اجرای فرآیند حذف خواهد شد.

شکل ‏4–7-تب دوم از برنامه کاربردی سرور پردازشی – کنترل لایه پردازشی

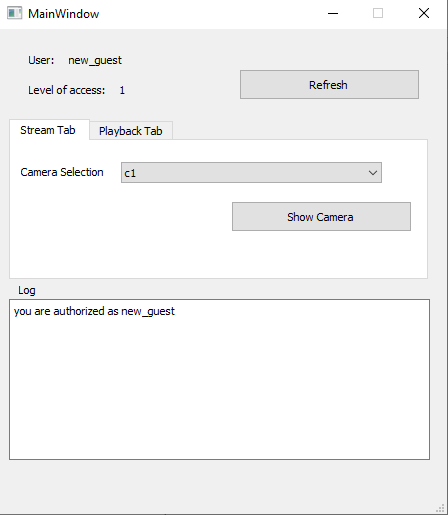
### راهنمای راه اندازی برنامه کاربردی کاربر

این برنامه را می­توان بر روی کامپیوتری در شبکه داخلی کامپیوتر سرور اجرا کرد. این سیستم تنها نیاز به پایتون دارد. و برای اجرای آن روی یک سیستم دیگر کافی است که کتابخانه های پایتونی که در فایل ‏requaetment.txt‏ قرار گرفته اند را نصب کرد. برای اجرای این برنامه کافی است که پوشه ‏client ‎‏ را به سیستم مد نظر منتقل نمود و وارد آن شد و با اجرای دستور زیر این برنامه را اجرا کرد.

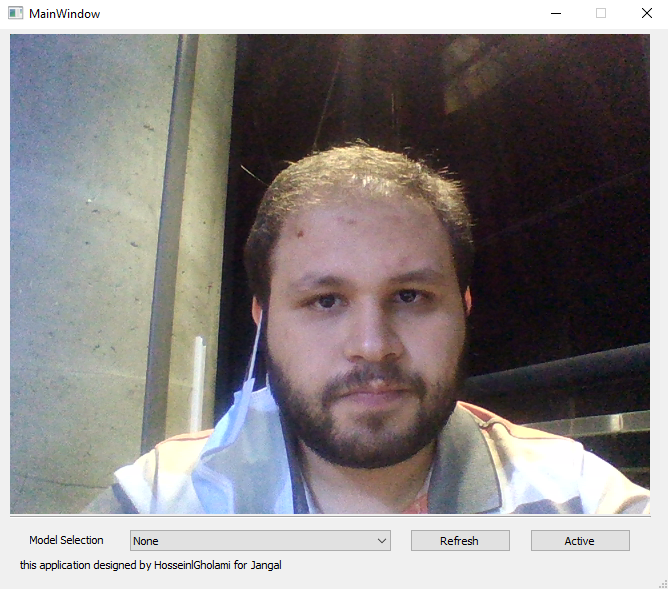
python client.py

برنامه کاربردی کاربر برای دسترسی به اطلاعات دوربین ها، اطلاعات پردازش شده دوربین ها و بازپخش و دانلود اطلاعات ذخیره شده دوربین ها طراحی شده است. و براساس نام کاربری و رمز عبوری که کاربر وارد می­کند، توسط سیستم شناسایی خواهد شد و اطلاعات دوبین های مرتبط با خودش را به وی نمایش خواهد داد. با اجرای دستور فوق برنامه گرافیکی شکل ‏4–8 به کاربر نمایش داده خواهد شد که در خط اول ip کامپیوتری که برنامه کاربردی سرور در آن اجرا شده است را وارد کنید.

شکل ‏4–8-صفحه لاگین برنامه کاربردی کاربر

در این برنامه ابتدا ip کامپیوتری که برنامه سرور بر روی آن اجرا شده است را وارد کنید و در خط بعدی با نام کاربری و گذرواژه ای که در اپلیکیشن سرور تعیین شد میتوان وارد اپلیکیشن شد. در صورتی که نام کاربری و رمز عبور اشتباه باشد با خطای authorizatoin fail -->not\_authorized رو به روی خواهید شد. و در صورتی که نام کاربری و رمز عبور درست باشد، عبارت you are authorized as USERNAME چاپ خواهد شد. سپس این صفحه بسته شده و صفحه رابط کاربری کاربر نمایش داده خواهد شد. این صفحه را می­توانید در شکل ‏4–9 مشاهده کنید. در این صفحه در بخش بالا نام کاربری و سطح دسترسی کاربر نمایش داده می­شود. در کنار آن یک کلید بازیابی وجود دارد تا اطلاعاتی که در تب ها هستند را به روزرسانی کند. این برنامه دو وظیفه اصلی پخش زنده اطلاعات و باز پخش اطلاعات را بر عهده دارد، که این امکان در دو تب تعبیه شده است. ابتدا به بررسی تب اول می­پردازیم در این تب یک لیست انتخابی وجود دارد که در آن لیست دوربین های فعال در برنامه کاربردی سرور نمایش داده می­شود. با انتخاب از این لیست به سادگی میتوان فرآیند مربوط به پخش زنده اطلاعات را فعال کرد و به اطلاعات زنده دوربین دسترسی پیدا کرد. در صورتی که فرآیند پخش زنده به درستی اجرا شود، show camera :CAMERA\_NAMEبه نمایش خواهد آمد و در صورتی که این پنجره برای یک دوربین از قبل در حال اجرا باشد، برای مدیریت منابع اجازه اجرا داده نخواهد شد و خطای this cam window is already open, close that first بر روی لاگ ثبت خواهد شد. با فعال شدن این فرآیند در شکل ‏4–10 می­توان این پنجره را مشاهده نمود که در یک پنجره که به صورت مجزا باز می­شود، اطلاعات زنده دوربین انتخابی به نمایش گذاشته می­شود. همانطور که قابل مشاهده است در پایین این پنجره دو کلید و یک لیست انتخابی وجود دارد که وظیفه دریافت اطلاعات پردازش شده را بر عهده دارند. یک کلید بازیابی وجود دارد تا از الگوریتم های هوش­مصنوعی که پس از باز شدن پنجره برای این دوربین فعال شده اند با خبر شد و لیست انتخابی به روز شود. همانطور که در شکل ‏4–12 مشاهده می­کنید با فشردن بر روی لیست انتخابی، الگوریتم های هوش مصنوعی که برای این دوربین فعال هستند به نمایش در می­آید.

شکل ‏4–9-تب اول برنامه کاربردی کاربر- پخش زنده اطلاعات

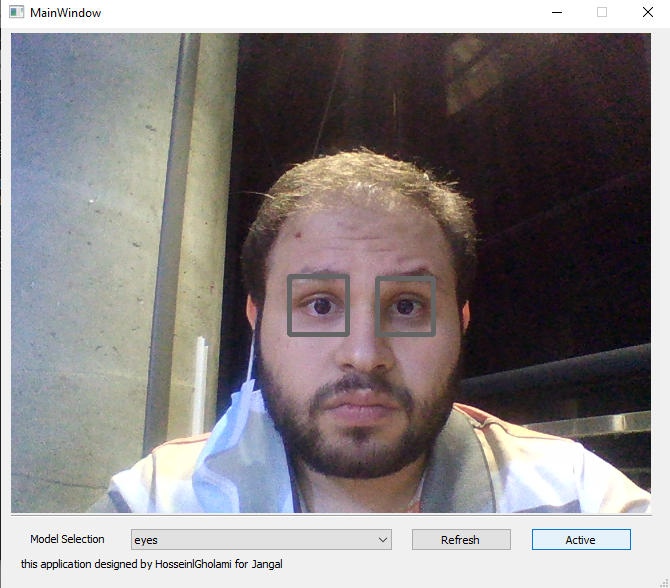


شکل ‏4–10پنجره پخش زنده اطلاعات دوربین

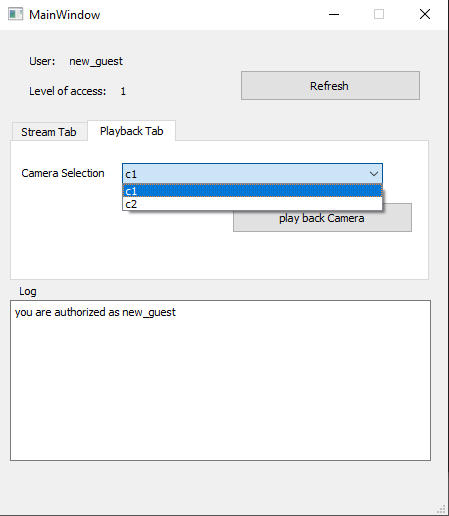


شکل ‏4–11-لیست انتخابی از الگوریتم های هوش مصنوعی فعال شده برای دوربین باز شده.

حال می­توان مدل هوش­مصنوعی مدنظر را انتخاب نمود و فشردن کلید فعال­سازی به اطلاعات این مدل دسترسی پیدا کرد. برای مثال با انتخاب مدل eyes و فشردن کلید فعال سازی این مدل بر روی ویدیو به نمایش خواهد آمد. در تصویر زیر میتوان نمونه آن را مشاهده نمود. از آن جایی که این الگوریتم را ما برای 60 فریم تنظیم کردیم و اطلاعات دوربین با نرخ 30 فریم بر ثانیه می­آید، هر دو ثانیه یک فریم پردازش خواهد شد، و جای چشم ها را تشخیص می­دهد. همچنین برای غیر فعال کردن نمایش اطلاعات پردازش شده کافی است در لیست انتخابی بر روی گزینه None کلیک کرده و سپس بر کلید فعال­سازی را انتخاب کنیم.

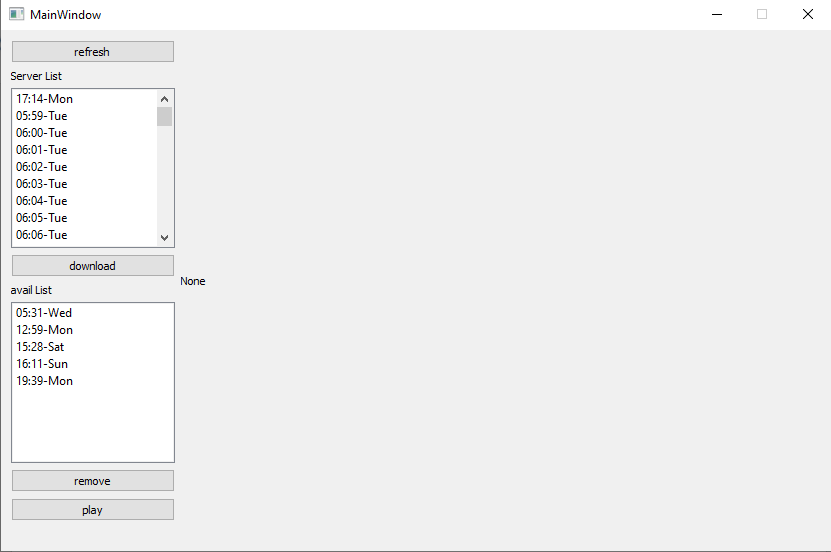


شکل ‏4–12-نتیجه فعال سازی الگوریتم تشخیص چشم بر روی دوربین

حال به سراغ تب دوم برنامه کاربردی کاربر یعنی باز پخش اطلاعات ذخیره شده میرویم در شکل ‏4–13 می­توانیم محتوایات این تب را مشاهده کنیم. در این تب یک لیست انتخابی وجود دارد که دوربین­هایی که از آن ها در سرور اطلاعات داریم را به نمایش میگذارد. همچین یک کلید وجود دارد که با انتخاب دوربین مدنظر فرآیند بازپخش اطلاعات، برای دوربین انتخابی فعال می­شود.

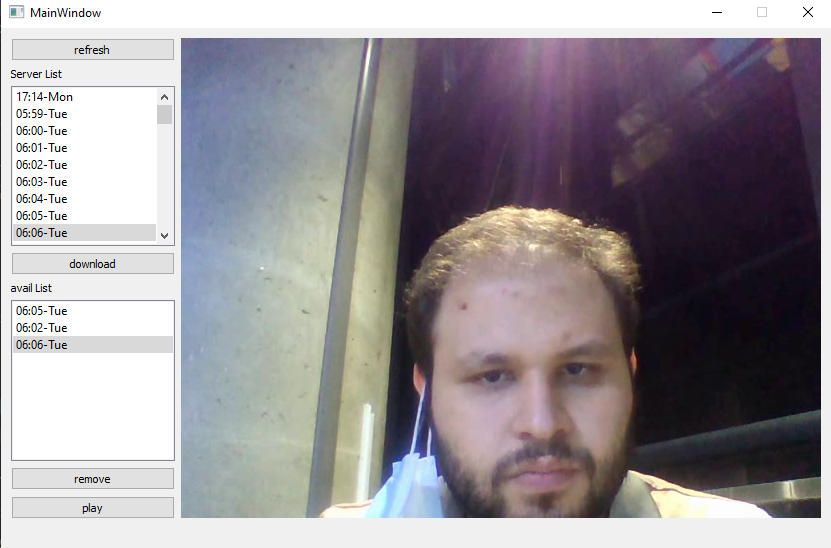
شکل ‏4–13-تب دوم برنامه کاربردی کارب-بازپخش اطلاعات ذخیره­شده

با انتخاب این گزینه رابط کاربری بازپخش اطلاعات ذخیره شده به عنوان یک پنجره جدید باز می­شود که در شکل ‏4–14 قابل مشاهده است. این پنجره از دو بخش اصلی تشکیل شده، یه بخش برای دسترسی به اطلاعات روی سرور، که یک کلید بازیابی برای دسترسی به آخرین اطلاعات ذخیره شده روی سرور وجود دارد، یک لیست انتخابی برای نمایش اطلاعاتی که روی سرور وجود دارد که بر اساس روز هفته و زمان اطلاعات روی سرور را نمایش می­دهد، و یک کلید دریافت اطلاعات که فایل انتخابی که بر روی سرور است را به کامپیوتر کابر منتقل می­کند. بخش دوم این برنامه از یک لیست که



شکل ‏4–14-پنجره بازپخش اطلاعات ذخیره شده دوربین

شامل اطلاعاتی است که از سرور بر روی کامپیوتر کاربر دریافت شده تشکیل می­دهد که با دو کلید که در زیر آن قرار گرفته مرتبط است، کلید اول کلید حذف است، بدین معنا که گزینه انتخابی از لیست اطلاعات موجود در کامپیوتر کاربر را حذف می­نماید و کلید بعدی کلید پخش اطلاعات است که در سمت راست تصویر فایل یک دقیقه دریافت شده را به نمایش می­گذارد. در شکل ‏4–15می­توان نحوه نمایش اطلاعات را مشاهده نمود. همچین لازم به ذکر است که اطلاعات دریافت شده در پوشه ای به نام download که در مسیری برنامه کاربر اجرا شده است. قرار خواهند گرفت و کابر با هر برنامه پخش ویدویی که مد نظر داشته باشد میتواند به اطلاعات دوربین ها دسترسی پیدا کند.



شکل ‏4–15- پخش اطلاعات ذخیره شده در پنجره بازپخش اطلاعات ذخیره شده

## جمع‌بندی فصل چهارم

در این فصل ابتدا نحوه نصب سیستم فوق را در یک مجموعه بررسی نمودیم و سعی کردیم جزئیات و پیش­نیاز هایی که هنگام نصب باید در نظر گرفته شود را بررسی نماییم. سپس به سراغ راه اندازی هر یک از برنامه های کاربردی رفتیم و راه­اندازی کامل سیستم را بررسی نمودیم. در فصل بعد، معماری کلی در رابطه سیستم را مرور خواهیم داشت و پیشنهادهایی برای بهتر کردن عملکرد بخش های مختلف یا و نوآوری هایی که در راستای توسعه این محصول می­توان انجام داد را بررسی خواهیم نمود.

# فصل پنجم نتیجه‌گیری و پیشنهادها



## نتیجه­گیری

در این پژوهش

## پیشنهادها

پیشنهاد اول بجای اینکه از پراسس یا فرایند ها استفاده کنیم آن ها را داخل یک کانتینر داکر ببریم تا ببریم و با کوبرنتیس و این داستانا بتوانیم لود بالانسینگ رو بین کامپیوتر ها انجام بدیم. الان همه فشار روی ram کامپیوتر سرور هست.

بجای pyqt از Django و سرویس وب استفاده شه که رو هر دیوایسی بشه وصل شد به سیستم مثل گوشی و...

چیدن اکسچنج های ربیت به حالت federated که هنگام کلاستر کردن ربیت ساده تر بشه قصه رو هندل کردن و هر کاربر رو برد توی vhost جدا گانه

# مراجع

|  |  |
| --- | --- |
| [1] | https://www.eghtesadnews.com/fa/tiny/news-273274, تعداد خودرو ها در سطح شهر, تهران: اقتصاد نیوز, 1397. |
| [2] | Docker overview, https://docs.docker.com/, 2021. |
| [3] | https://www.rabbitmq.com/documentation.html, RabbitMQ documentation 3.8.4, 2021. |

**Abstract**

Dangerous driving is one of the most important causes of car accidents. If drivers know that their behavior is being recorded while driving, they will be driving safer. Also, if it is possible to implement a system that encourages people to drive safely by being aware of the type of driving behaviors of individuals, this will lead to a reduction in traffic accidents. This project aims to implement a method for detecting driving events (including dangerous right and left lanes, dangerous lane changes left and right, dangerous braking, and acceleration) with the help of sensor information so that driver information Receive and store in a supercomputer and use a solution to calculate how safe people are driving. More precisely, using a new windowing method on time series, a decision tree on window similarities (using the Fast-DTW algorithm) was trained to detect driving events, then to receive and Data storage RabbitMQ was used as a message exchanger, and Redis was used as a temporary database. Finally, a statistical method was used to score and compare people's driving. The dangerousness of driving introduces them as a benchmark for comparison. With the help of the results presented in this study, an insurance plan can be implemented so that drivers who show safer driving pay fewer insurance premiums so that this plan can be used as an incentive for Improved driving culture was considered.

**Key Words:** Internet of Things, Machine learning, Driver Profiling, Driving culture

1. یک شرکتی است دستگاه هایی در زمینه کنترل ورود و خروج و قفل و باز کردن درب ها در اختیار استفاده کنندگان از این سرویس قرار میدهد. [↑](#footnote-ref-1)
2. Open Network Video Interface Forum [↑](#footnote-ref-2)
3. Real-Time Streaming Protocol [↑](#footnote-ref-3)
4. Session [↑](#footnote-ref-4)
5. integration [↑](#footnote-ref-5)
6. در این پروژه سرویس MinIO به این شکل عمل می­کند. [↑](#footnote-ref-6)
7. در این پروژه سرویس redis بهاین شکل عمل می­کند. [↑](#footnote-ref-7)
8. در این پروژه دریافت اطلاعات از دوربین ها به نحوه پیاده سازی شده است. [↑](#footnote-ref-8)
9. Message Broker [↑](#footnote-ref-9)
10. در این پروژه سرویس RabbitMq به این شکل عمل می­کند. [↑](#footnote-ref-10)
11. Docker [↑](#footnote-ref-11)
12. container [↑](#footnote-ref-12)
13. Daemon process [↑](#footnote-ref-13)
14. Command line [↑](#footnote-ref-14)
15. CI(continuous integration) [↑](#footnote-ref-15)
16. CD(continuous delivery) [↑](#footnote-ref-16)
17. Portable [↑](#footnote-ref-17)
18. docker-compose [↑](#footnote-ref-18)
19. Producer [↑](#footnote-ref-19)
20. Consumer [↑](#footnote-ref-20)
21. Queue [↑](#footnote-ref-21)
22. Router(Exchange) [↑](#footnote-ref-22)
23. Channels [↑](#footnote-ref-23)
24. Binding [↑](#footnote-ref-24)
25. Direct Exchange [↑](#footnote-ref-25)
26. Fanout Exchange [↑](#footnote-ref-26)
27. Topic Exchange [↑](#footnote-ref-27)
28. Header Exchange [↑](#footnote-ref-28)
29. Durability [↑](#footnote-ref-29)
30. Unicast [↑](#footnote-ref-30)
31. Broadcast [↑](#footnote-ref-31)
32. publish/subscribe [↑](#footnote-ref-32)
33. Task [↑](#footnote-ref-33)
34. Routing key [↑](#footnote-ref-34)
35. Prefetch [↑](#footnote-ref-35)
36. Caching [↑](#footnote-ref-36)
37. Value [↑](#footnote-ref-37)
38. Sets [↑](#footnote-ref-38)
39. Sorted Sets [↑](#footnote-ref-39)
40. Hashes [↑](#footnote-ref-40)
41. Access Control List [↑](#footnote-ref-41)
42. domain specific language [↑](#footnote-ref-42)
43. میتواند همه برنامه ها بر روی یک کامپیوتر اجرا شوند ولی بهتر است که کامپیوتر سرور از ram بیشتری برخوردار باشد و کامپیوتر سرور پردازشی از کارت گرافیک قوی برخوردار باشد. و کامپیوتر کلاینت میتواند یک کامپیوتر ساده باشد. [↑](#footnote-ref-43)
44. در ادامه در بخش نرم افزار سرور با جزئیات بیشتر به نحوه پیاده سازی پرداخته خواهد شد. [↑](#footnote-ref-44)
45. sequence number [↑](#footnote-ref-45)
46. Refresh [↑](#footnote-ref-46)
47. برنامه کاربردی سرور وظایف دیگری نیز دارد که در ادامه بررسی خواهیم نمود. [↑](#footnote-ref-47)
48. Unix time (also known as Epoch time, Posix time,seconds since the Epoch,or UNIX Epoch time( [↑](#footnote-ref-48)
49. Graphical User Interface رابط کاربری گرافیکی [↑](#footnote-ref-49)
50. Drag and drop [↑](#footnote-ref-50)
51. منظور از درگیری سیستم درگیری حاقظه و پردازنده و کارت شبکه و ... است. [↑](#footnote-ref-51)
52. با نام دوربین تعریف شده برابر است. [↑](#footnote-ref-52)
53. virtualization [↑](#footnote-ref-53)
54. Install using the repository [↑](#footnote-ref-54)
55. نمایش داده نخواهد شد. [↑](#footnote-ref-55)