پروژه طراحی و پیاده‌سازی پلتفرم اینترنت اشیاء

عنوان:

مستندات طراحی

ارائه دهنده:

کارگروه پلتفرم، گروه پژوهشی اینترنت اشیاء، دانشگاه صنعتی امیرکبیر

کد سند:

ISRC-AUT-970519.0

تاریخ انتشار:

۱۳۹۷/۰۵/۱۹

اطلاعات سند

نام پروژه:	پروژه طراحی و پیاده‌سازی پلتفرم اینترنت اشیاء
عنوان سند:	مستندات طراحی
کد سند:	ISRC-AUT-970519.0
نگارش:	۱/۰
نام تهیه کنندگان:	تیم فنی
تاریخ تهیه:	۱۳۹۷/۰۵/۱۹
نام بازبینی کننده:	مدیر پروژه
تاریخ آخرین بازبینی:	
نام تائیدکننده:	
تاریخ تائید:	
وضعیت:	نسخه نهایی
نوع طبقه‌بندی سند:	محرمانه

چکیده:

سند مربوطه نسخه نهایی طراحی سامانه اینترنت اشیا است که نسخه اولیه آن در فازهای قبلی ارائه گردیده بود. با توجه به تغییراتی که در پیاده‌سازی و یا تغییرات مفاهیم در حین توسعه و تست رخ داده است ویرایشات مربوطه انجام شده است. در نتیجه اکثر موارد ذکر شده قبلی با آخرین نسخه ویرایش آن در این سند گردآوری شده‌اند. در این سند سعی شده شمای کلی معماری پلتفرم به همراه ابزارهایی که استفاده گردیده در این سند توصیف گردند. در طراحی این پلتفرم از معماری میکروسرویس استفاده شده است که طبق بررسی‌های انجام شده، یکی از معماری‌های مطلوب جهت توسعه پلتفرم‌های اینترنت اشیا می‌باشد..

فهرست مطالب

فهرست اشکال

فهرست جداول:

۱- مقدمه

فاز طراحی یکی از فازهای اصلی در توسعه نرم‌افزار است. اسناد طراحی در حقیقت سنگ‌بنای مراحل توسعه، تست و ارزیابی سیستم می‌باشند. در فصل دوم این مستند معماری کلان پلتفرم به همراه کلیت مولفه‌های آن مورد بررسی قرار می‌گیرد. فصل سوم اجزای داخلی پلتفرم به همراه فرآیندها یا سناریوهای آن را ارائه می‌دهد. در ادامه در فصل چهارم پروتکل‌های ارتباطی با اشیا و نحوه ارتباط با Gatewayها تشریح می‌گردد. فصل پنجم نحوه ارتباط برنامه‌های کاربردی با پلتفرم را شرح می‌دهد. سپس در فصل ششم پروتکل‌ها و APIهای طراحی شده جهت ارتباط با ابزارهای تحلیل داده مورد بررسی قرار می‌گیرد. فصل هفتم جزییات بیشتر از نحوه حسابرسی و شارژینگ را مطرح می‌کند. در فصل هشتم، طراحی پایگاه داده پلتفرم ارائه می‌گردد. فصل نهم به مدیریت اشیا اشاره‌ای دارد. در فصل دهم ابزارهای مدیریتی 3rd party مورد استفاده در سامانه را تشریح می‌کند. علاوه بر فصول ذکر شده، در پیوست نیز کد برخی از سناریوها و پیوست‌ها آمده است.

۲- معماری کلان پلتفرم

۲-۱- مقدمه

یکی از فاکتورهایی که نقش کلیدی در موفقیت یک محصول نرم‌افزاری دارد، معماری آن است. طبیعتا طراحی پلتفرم هم از این قضیه مستثنی نیست و تامین همه نیازمندی‌ها بخصوص نیازمندی‌های غیرکارکردی شامل دسترس‌پذیری و مقیاس‌پذیری در گرو استفاده از یک معماری مناسب برای سامانه است. معماری سامانه، اجزای داخلی سامانه و نحوه تعاملات آن‌ها را در راستای برآورد انتظارات کارفرما مشخص می‌کند. همان‌طور که در پروپوزال پروژه نیز شده است معماری ارائه شده بر اساس میکروسرویس می‌باشد. در ابتدا معماری میکروسرویس و مزایای آن تشریح شده است. سپس معماری پلتفرم پیشنهادی با جزئیات کامل ارائه و مورد بررسی قرار می‌گیرد.

۲-۲- معماری میکروسرویس

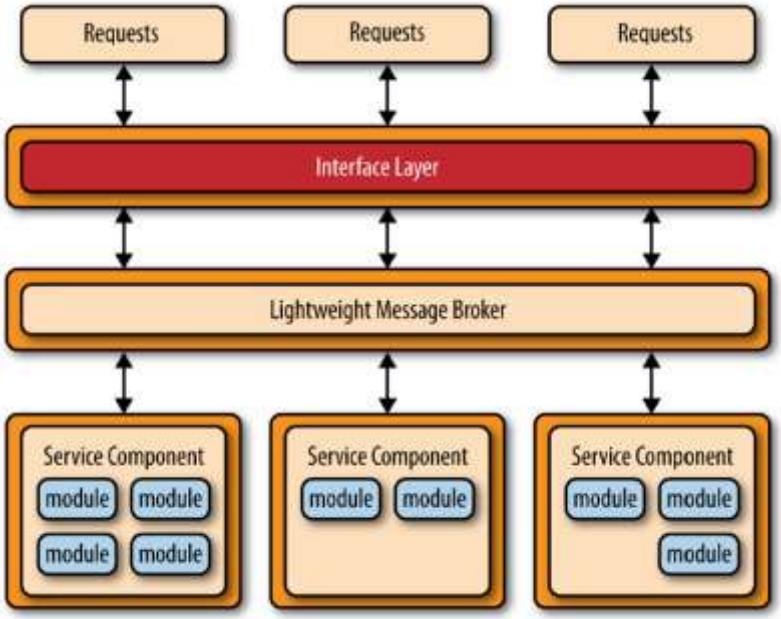
در حوزه معماری سیستم‌های نرم‌افزاری، الگوهای معماری متعدد و دسته‌بندی‌های متفاوتی وجود دارد. مهمترین معماری‌هایی که در حال حاضر استفاده می‌شوند شامل معماری‌های لایه‌ای[۱] (مانند MVC[۲])، مبتنی بر رویداد[۳]، میکرو کرنل، میکروسرویس و Space-Based می‌باشد.

معماری میکروسرویس برخلاف روشهای سنتی سامانه‌های یکپارچه، یک سیستم توزیع شده متشکل از چندین سرویس است که هر سرویس به صورت مجزا پیاده‌سازی شده و سرویس مربوطه را از طریق API در اختیار سایر سرویس‌ها و مشتریان قرار می‌دهد. این معماری در حقیقت سرعت و کارایی بالایی را در توسعه و تغییر سیستم فراهم می‌کند و معماری کلاسیک مبتنی بر سرویس را بهبود می‌دهد. شمای کلی این معماری در شکل ۲-۱ نشان داده شده است.

[۱] Layered (n-tier)

[۲] Model View Controller

[۳] Event Driven

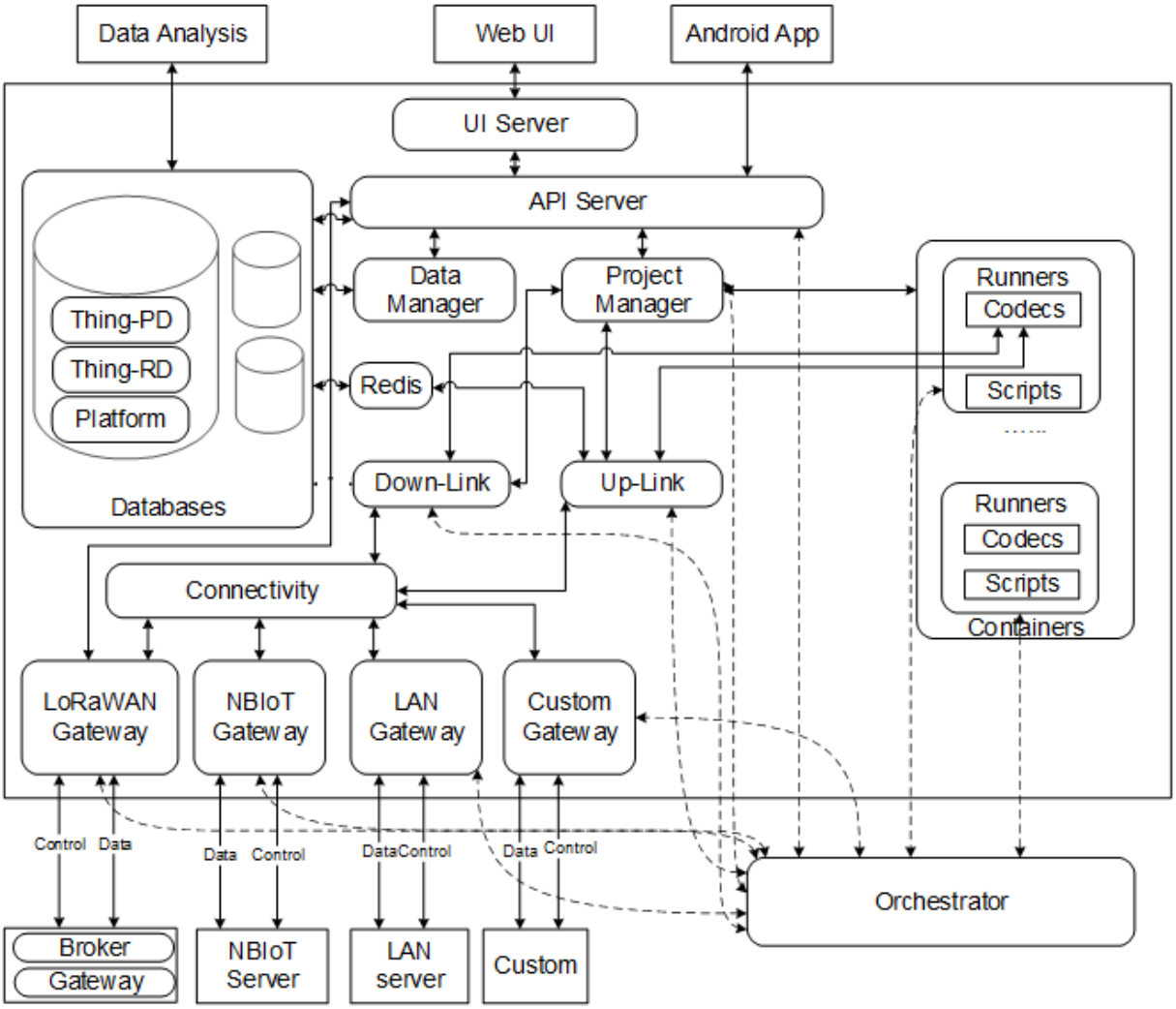


شکل ۲-۱ معماری میکروسرویس

در معماری میکروسرویس، کارکردهای سامانه در قالب سرویس‌هایی پیاده‌سازی می‌شوند که هر سرویس یک واحد مستقل است و همه موارد مورد نیاز جهت اجرای خود را دارد. علاوه بر آن می‌تواند در هر کجای شبکه استقرار پیدا کند. برای توزیع درخواست‌های داده شده به سامانه و همچنین ارتباطات بین سرویس‌ها عموما از یک Message Broker در این معماری استفاده می‌شود. استفاده از این معماری برای پیاده‌سازی پلتفرم بومی عام منظوره پیشنهادی دارای چندین مزیت است. اول اینکه اکثر نیازمندی‌های کارکردی سامانه می‌توانند به صورت سرویس‌های مجزا در قالب این معماری پیاده‌سازی شوند که فرایند نگه‌داری و قابلیت اطمینان آن را راحت‌تر می‌کند چرا که به راحتی می‌توان تشخیص داد کدام واحد دچار خطا شده است. این معماری به دلیل توزیع‌شدگی به صورت ذاتی مقیاس‌پذیر بوده و امکان اضافه کردن قابلیت‌های غیرکارکردی به این معماری وجود دارد. همچنین در طراحی‌هایی که کامپوننت‌های مختلف وجود دارند و یا توسعه آن‌ها برون‌سپاری می‌گردد امکان توسعه آن‌ها به زبان‌های مختلف و در تیم‌های جداگانه به راحتی از طریق APIهای استاندارد فراهم می‌گردد.

۲-۳- معماری پلتفرم مبتنی بر میکروسرویس

شمای کلی معماری ارائه شده برای پلتفرم اینترنت اشیا در شکل ۲-۲ نشان داده شده است. در این شکل اجزای مختلف پلتفرم، ارتباطات داخلی و ارتباطات خارجی آن مشخص شده است.



شکل ۲-۲. معماری سیستم

در جدول ۲-۱ واحدهای متفاوت این معماری به شکل خلاصه معرفی شده‌اند و مشخص شده است که هر واحد از طریق کدام زبان برنامه‌نویسی پیاده‌سازی شده و یا برای راه‌اندازی آن از چه تکنولوژی و ابزاری استفاده شده است.

جدول ۲-۱. واحدها به همراه شرح وظایف

نام واحد	وظیفه	زبان / تکنولوژی
Data Analysis	این واحد برای اتصال به ابزارهای تحلیل داده مورد استفاده قرار می‌گیرد.	Java/Apache Spark
Web UI	این واحد وب سایت پلتفرم را پیاده می‌کند. ارتباط با API سرور جهت فراخوانی واسط‌های سمت سرور نیز در این قسمت صورت می‌پذیرد.	HTML/CSS/ReactJS
UI server	این واحد وظیفه‌ی ارسال صفحات وب سایت پلتفرم را برعهده دارد.	NodeJS
API-Server	واسط‌ها و توابع back-end سمت سرور توسط این واحد پیاده‌سازی می‌گردد. واسط‌های فراهم شده هم توسط وب و هم توسط موبایل قابل استفاده و فراخوانی است.	PHP (Laravel Framework)
Project Manager	وظیفه مدیریت پروژه‌های ایجاد شده توسط کاربر را به عهده دارد و به هر پروژه یک container اختصاص می‌دهد.	Go
Data Manager	این واحد وظیفه پیاده‌سازی توابع مربوط به پرس‌وجوهای بر روی داده‌ها، را بر عهده دارد.	Go
Containers	مجموعه‌ای از Containerها را شامل می‌گردد که هر Container وظیفه مدیریت سناریوها و کدک‌های آن پروژه را بر عهده دارد.	Python3/Go
Down-Link	این واحد وظیفه انتقال اطلاعات از پلتفرم به اشیاء را بر عهده دارد.	Go
Up-Link	این واحد وظیفه انتقال اطلاعات از اشیاء به پلتفرم را به عهده دارد.	Go
Connectivity	این واحد وظیفه مدیریت اتصال به gatewayهای پروتکلهای ارتباطی با اشیا را به عهده دارد.	Go
LoRaWAN Gateway	Gateway ارتباط با اشیا را از طریق LoRa Server فراهم می‌کند.	Go
LAN Gateway	Gateway ارتباط با اشیا را از طریق LAN Server فراهم می‌کند.	Go
Orchestrator	این واحد وظیفه مدیریت تمامی‌واحدهای را بر عهده دارد.	Portainer
Database	این واحد وظیفه مدیریت داده‌های پلتفرم را به عهده دارد.	MongoDB
Redis	به عنوان یک محل ذخیره سازی داده به صورت in memory، نقش یک واسط را بین پایگاه داده و واحد Up-Link ایفا می‌کند. با وجود این بخش نرخ داده‌ای که می‌توانیم در پایگاه داده‌ای ذخیره کنیم افزایش پیدا می‌کند.	Redis
Android App	یک برنامه کاربردی موبایل است که امکان اتصال مستقیم و استفاده از توابع ارائه شده توسط API Server را دارد.	--

در ادامه این مستند عملکرد واحدها به تفصیل مورد بحث و بررسی قرار خواهد گرفت. شایان ذکر است که پیاده‌سازی ابزار تحلیل داده و کاربردهای اندرویدی در حوزه توسعه پلتفرم نبوده است.

۳- مولفه‌ها و فرآیندهای پلتفرم اشیا

۳-۱- مقدمه

در این بخش ابتدا مولفه‌های درون پلتفرم اینترنت اشیا تشریح می‌گردد. پس از معرفی سرویس‌ها، فرآیندهای پایه‌ای که در راستای مدیریت اشیا و پلتفرم تعریف شده است مورد بررسی قرار می‌گیرد.

۳-۲- مولفه API-Server

این مولفه به نوعی در قلب پلتفرم قرار دارد و وظیفه پیاده‌سازی توابع Backend وب را برعهده دارد. تمام توابع مرتبط به سمت کاربر (دریافت داده، نمایش داده و ..) و مدیریت پلتفرم (مدیریت کاربران، مدیریت درگاه‌های پرداخت و ...) در این واحد پیاده‌سازی شده‌اند. این مولفه از یک سمت با واحدهای Project Manager، Data Manager، Gateway Manager و در برخی موارد با Gatewayهای ارتباطی با اشیا مانند LoRaWAN (در پلتفرم از پیاده‌سازی loraserver.io به عنوان یک درگاه LoRaWAN استفاده می‌شود.) در ارتباط است. در سمت دیگر نیز با واسط‌های گرافیکی کاربران نهایی و مدیر پلتفرم (Front-End) و برنامه‌های کاربردی (مانند کاربرد سمت اندروید) در ارتباط است. این مولفه در این ارتباط، سرویس‌های لازم را فراهم می‌کند و در صورت لزوم برخی از سرویس‌ها را دریافت می‌کند. توابع ارائه شده توسط این بخش به دو دسته اصلی زیر تقسیم شده‌اند:

۱- توابع کاربران نهایی که برخی از مهمترین آنها عبارتند از

- احراز هویت شامل ورود به سیستم، ثبت نام و ویرایش کاربران
- مدیریت پروژه‌ها شامل ساخت پروژه، و به روزرسانی اطلاعات پروژه
- مدیریت اشیاء شامل ساخت شی، حذف شی، دریافت و ارسال داده، نوع درگاه ارتباطی (LAN و LoRaWAN)
- مدیریت پروفایل اشیاء شامل ساخت پروفایل، بازیابی پروفایل و حذف پروفایل‌های اشیا
- مدیریت کدک شامل ساخت کدک و بازیابی کدک، صحت‌سنجی و آزمایش کدک و طبقه‌بندی کدک‌ها به دو دسته‌ی عمومی و شخصی
- مدیریت سناریوها شامل ساخت سناریو، به روزرسانی سناریو، بازیابی سناریو و صحت سنجی سناریو

- مدیریت Gatewayها
- حسابرسی و پرداخت کاربران
- داشبورد کاربر
- ...

۲- توابع مدیریت پلتفرم که برخی از مهمترین آنها عبارتند از

- مدیریت کاربران و اطلاعات آن‌ها
- مدیریت اجازه دسترسی‌ها و نقش‌ها (نقش‌ها مجموعه‌ای از اجازه‌های دسترسی هستند که به کاربران تخصیص می‌یابند.)
- مدیریت درگاه‌های پرداخت و کدهای تخفیف
- Impersonate کردن در نقش کاربران
- مدیریت کانتینرهای (Containers) پروژه‌ها از طریق داشبوردهای portainer
- گزارش‌گیری از تراکنش‌های سیستم
- ارائه نمودارهای مختلف معیارهای پلتفرم از طریق ابزار Prometheus
- مدیریت‌های قالب‌های عمومی کاربران
- و...

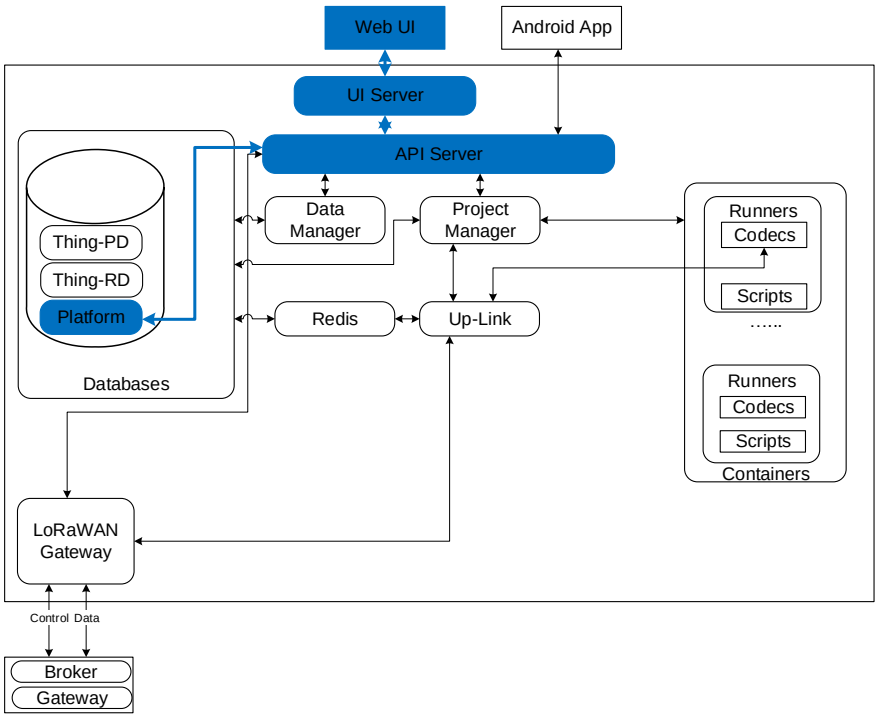
همه توابع ذکر شده، قابلیت ارائه سرویس به واسطهای گرافیکی تحت وب (Web UI , UI Server) و کاربردهای سمت اندروید (android App) را دارند. تمامی APIها به صورت Postman در اختیار کارفرما به صورت یک فایل الکترونیکی در تحویل فاز آخر قرار داده خواهد شد.

۳-۳- مولفه UI Server و Web UI

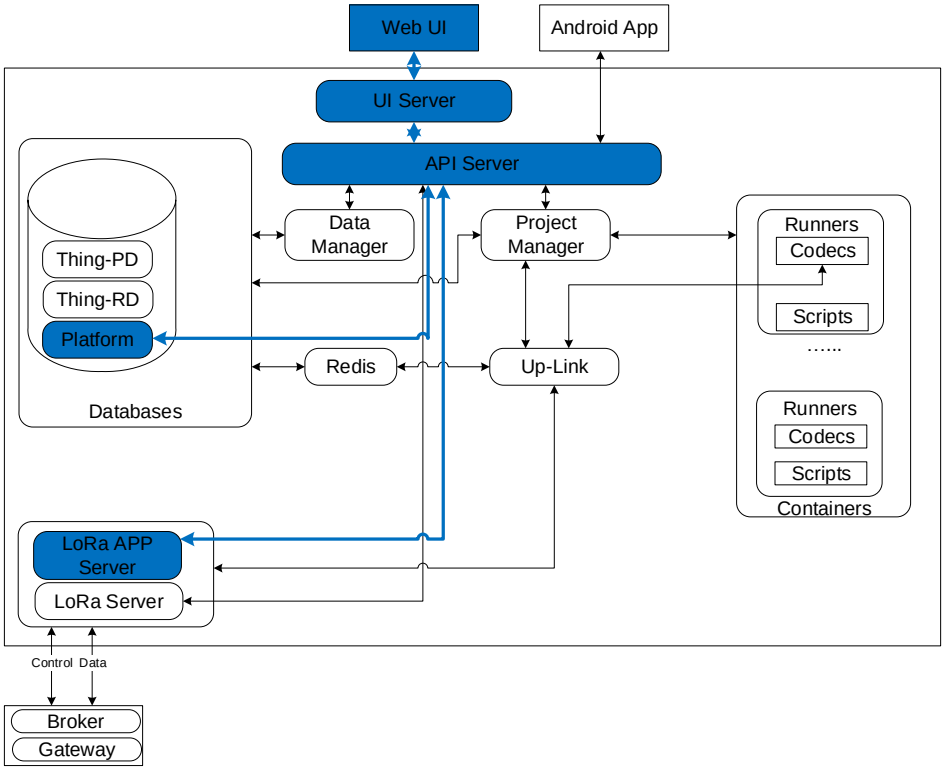
این مولفه‌ها همان‌طور که قبلا گفته شد بخش Front-end پلتفرم را تشکیل می‌دهند. که با توجه به جدا بودن Back-end، در نتیجه APIهای فراهم شده برای این بخش توسط سایر برنامه‌های کاربردی نیز قابل استفاده است. واحد Web UI ظاهر گرافیکی سایت و فراخوانی APIهای مربوط به Back-end را برعهده دارد و UI Server نیز فایل‌های UI را در اختیار کاربران قرار می‌دهد.

برای نمونه دو فرآیند اصلی ثبت نام کاربر و تعریف شی در شکل ۳-۱ و شکل ۳-۲ نمایش داده شده است. در شکل‌ها واحدهای درگیر در فرآیند نیز با رنگ آبی نشان داده شده‌اند. در سناریوی ثبت نام کاربر

(شکل ۳-۱) ابتدا کاربر از واسط کاربری خود گزینه ثبت نام را انتخاب می‌کند. پس از انتخاب کاربر توابع مورد نیاز برای ثبت اطلاعات از API-Server فراخوانی می‌شود و اطلاعات کاربر در پایگاه داده مربوط به کاربران ثبت می‌گردد. در سناریوی تعریف شی (شکل ۳-۲)، کاربر از طریق واسط کاربری گزینه مرتبط به تعریف شی را انتخاب می‌کند. پس از انتخاب کاربر، توابع مرتبط از بخش API-Server فراخوانی می‌شود و اطلاعات شی مورد نظر پس از طی روال تایید آن در سمت LoRaWAN Gateway، در پایگاه داده ثبت می‌گردد. نیازمندی-ها و پروسه فعال‌سازی اشیا به دو روش OTAA و ABP قبلا در سند تحلیل نیازمندی‌های کارکردی اینترنت اشیا –سرویس‌های پایه (IoT-RA-BS-v1.16) شرح داده شده است.



شکل ۳-۱. فرآیند ثبت نام کاربر



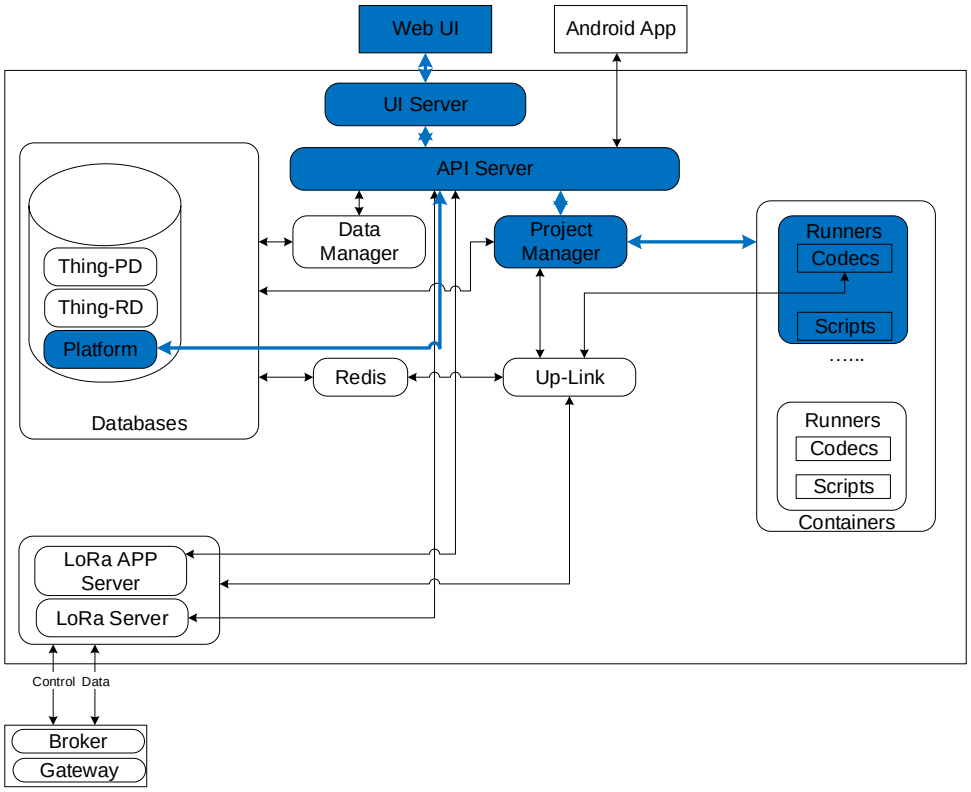
شکل ۳-۲. فرآیند تعریف شی

۳-۴- مولفه Project Manager

این واحد وظیفه مدیریت پروژه‌های تعریف شده کاربران را دارد که شامل ایجاد پروژه، فعال یا غیر فعال‌سازی پروژه می‌باشد. در

شکل ۳-۳ سناریوی تعریف پروژه نشان داده شده است و واحدهای مرتبط با رنگ آبی مشخص شده‌اند. در این سناریو کاربر از طریق منوی تعریف پروژه که در واسط کاربری فراهم شده است یک پروژه ایجاد می‌کند. پس از دریافت درخواست کاربر از طریق واسط کاربری، توابع مرتبط در سمت سرور فراخوانی می‌شود که این امر باعث فعال شدن واحدهای Project Manager، Runner و Database می‌شود و در نهایت پروژه تعریف می‌گردد. واحد Runners مجموعه‌ای از کانتینرها می‌باشد و در حقیقت سناریو و کدک‌های کاربران را مدیریت می‌کند و به ازای هر پروژه یک کانتینر تعریف می‌گردد که کدهای کاربر در آن اجرا می‌گردد. در این صورت خرابی کدک یا سناریو یک کاربر تاثیری در کدک‌ها و سناریوهای سایر کاربران ندارد. با توجه به اینکه هر پروژه یک Container برای خود دارد در نتیجه حجم مورد استفاده آن به ازای هر کاربر، بر اساس تعداد کاربران و اندازه دیتاسنتر در مراحل پایانی پروژه بررسی شده و ارائه می‌گردد. البته وضعیت این Runnerها

نیز در پلتفرم مدیریتی در اختیار مدیر پلتفرم قرار خواهد گرفت. اطلاعات ایجاد شده نیز در پایگاه داده Platform، جهت استفاده‌های بعدی ذخیره می‌گردد.

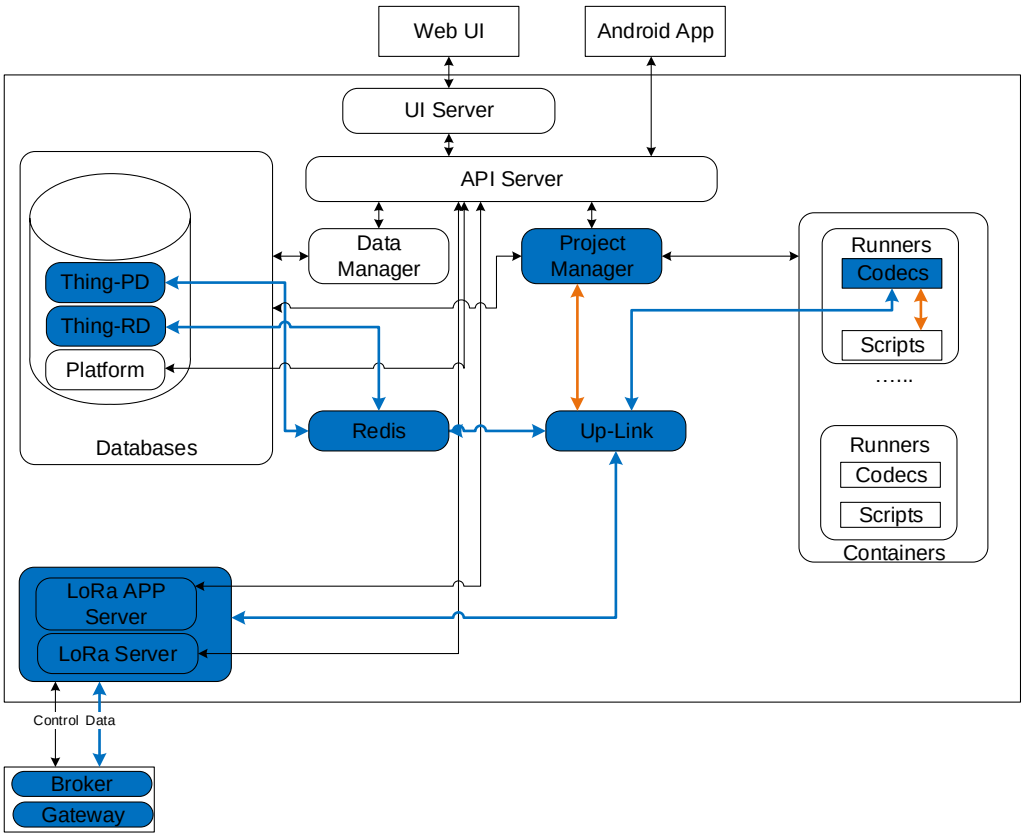


شکل ۳-۳. سناریوی تعریف پروژه

۳-۵- مولفه Up-link

این مولفه وظیفه انتقال اطلاعات از شی به پلتفرم را به عهده دارد. در این بخش داده‌ها به دو صورت پارس شده و خام در پایگاه داده قرار می‌گیرند. در شکل ۳-۴ سناریوی دریافت داده نشان داده شده است و واحدهای مرتبط با رنگ آبی نشان داده شده‌اند. اطلاعات خام دریافتی از سمت LoRaGateway از طریق uplink به سمت پایگاه داده ارسال می‌گردد. برای افزایش سرعت نیز از Redis به عنوان یک منبع ذخیره سازی داده in-memory استفاده شده است که توانایی پاسخ گویی به تعدادی زیادی از داده‌های دریافت شده در کسری از میلی ثانیه را داراست در نتیجه امکان Lost داده‌های دریافتی به دلیل نرخ زیاد داده‌ها از بین می‌رود. برای پارس کردن داده‌ها نیز، از کدکی که برای اشیا از سمت کاربران تعریف شده است استفاده

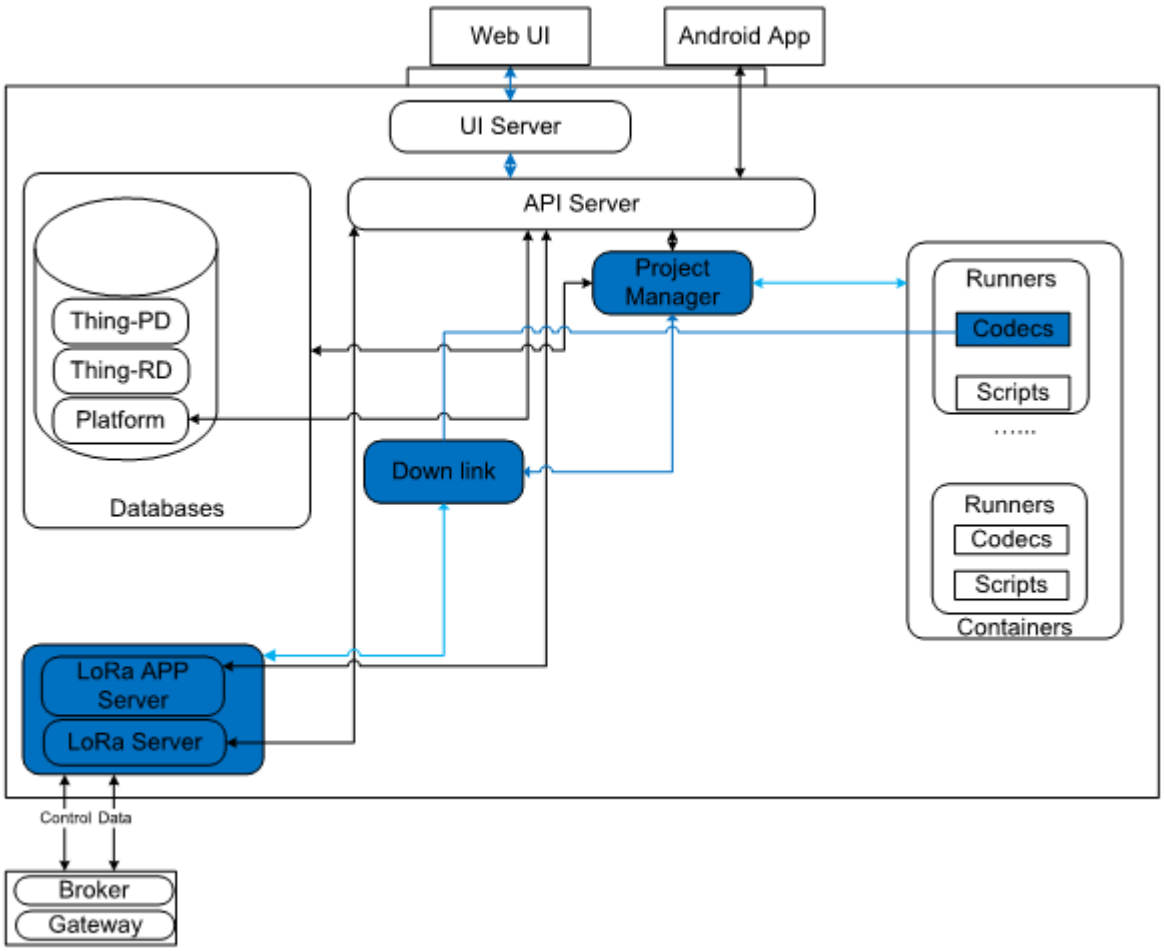
می‌گردد و اطلاعات اشیا در ابتدا پردازش و سپس جهت ذخیره‌سازی به پایگاه داده ارسال می‌گردد. در این حالت، واحد Up-Link اطلاعات مربوط به کدک کاربران را از Runner یا کانتینر مربوطه دریافت می‌کند. اینکه کدام Runner، کدک مربوطه را داراست از طریق Project Manager مشخص می‌شود. سپس اطلاعات پارس شده شی را استخراج می‌کند و در پایگاه داده ذخیره می‌کند.



شکل ۳-۴. سناریوی دریافت داده

۳-۶- مولفه Down-link

این بخش وظیفه انتقال داده از پلتفرم به اشیاء را به عهده دارد. روال آن در شکل ۳-۵ آمده است. واحد Down-link پس از دریافت اطلاعات از سمت کاربر، از طریق Project Manager، کدک‌های مرتبط شامل encoding مربوطه به آن شی را پیدا کرده، داده مد نظر را با استفاده از encoder مربوطه کد کرد و تبدیل به یک bit stream می‌کند که می‌تواند به شی مربوطه ارسال گردد. با توجه به شناسه شی، داده شده، اطلاعات از طریق Downlink به سمت LoRa Gateway ارسال می‌گردد.



شکل ۳-۵. سناریوی ارسال داده

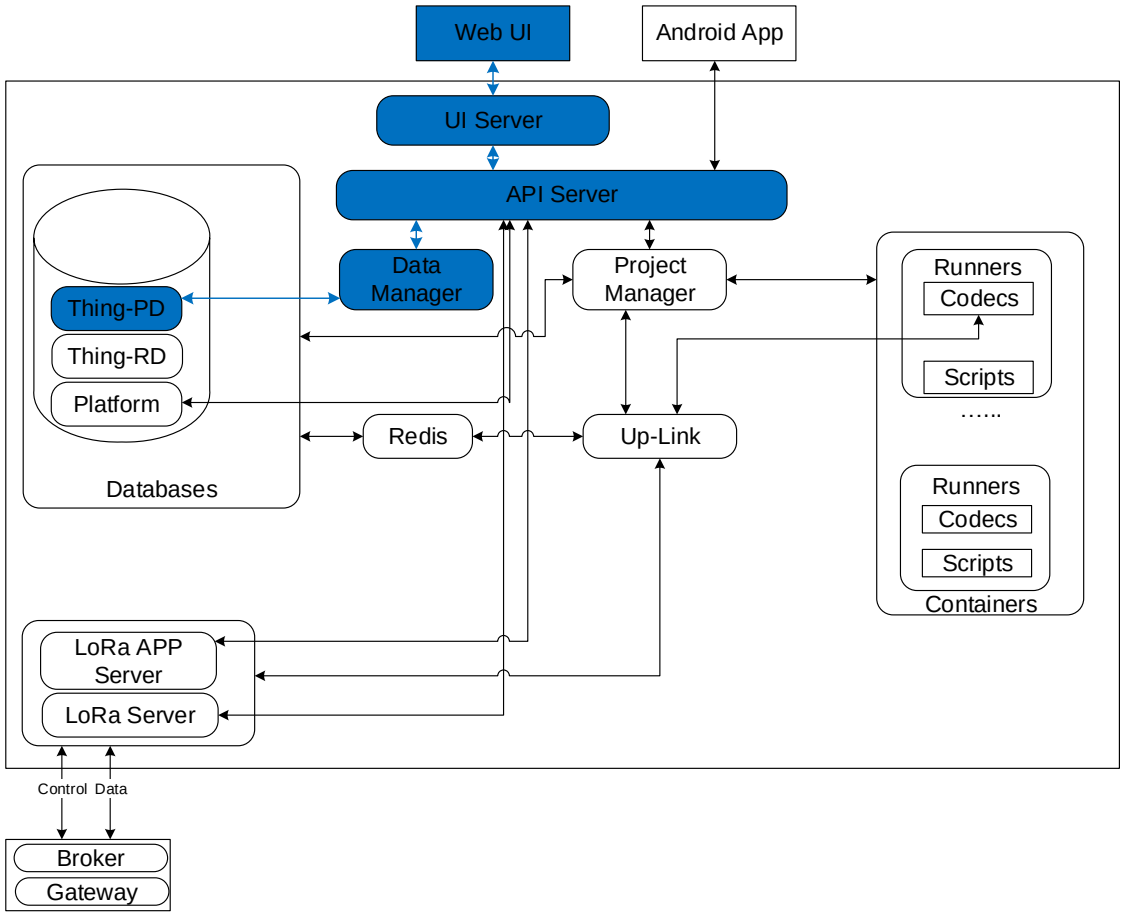
۳-۷- مولفه Data Manager

این واحد وظیفه پیاده‌سازی توابع مرتبط به عملیات پرس و جوی[۱] کاربران بر روی داده‌های پارس شده‌ی در پایگاه داده را بر عهده دارد. با توجه به اینکه درخواست‌های متعددی از واسط‌های گرافیکی کاربر برای نمایش داده به صورت نمودار و جدولی و جزئیات نمایش آن‌ها شامل Aliasing و Windowing ارسال می‌گردد مدیریت و نحوه پاسخگویی به این درخواست‌ها در این مولفه قرار داده شده است.

در شکل ۳-۶ فرآیند نمایش داده ترسیم شده است که واحدهای مرتبط با رنگ آبی نشان داده شده‌اند. در این سناریو ابتدا کاربر از طریق واسط کاربری گزینه نمایش داده‌های شی را انتخاب می‌کند. انتخاب کاربر

[۱] Query

در سمت سرور باعث فراخوانی توابع مرتبط می‌شود. در ادامه توابع بازیابی از واحد Data Manager صدا زده می‌شود و در نهایت اطلاعات مورد نظر بازیابی می‌گردد و به کاربر نمایش داده می‌شود.



شکل ۳-۶. سناریوی نمایش داده

۳-۸- مولفه Orchestrator

همان‌طور که در شکل ۲-۲ نیز نشان داده شد، این واحد با تمام واحدهای دیگر در ارتباط است و به نوعی مدیریت این واحدها را برعهده دارد. در این مولفه از پروژه Portainer پروژه‌ای متن باز جهت فراهم آوردن یک بستر مدیریتی برای داکر با ظاهری کاربر پسند استفاده شده است. نودهای فیزیکی که دارای docker engine می‌باشند در این بستر ثبت شده و سپس مدیریت می‌گردند. از طریق این ابزار می‌توان وضعیت حافظه، پردازش و شبکه نودهای فیزیکی، کانتینرها و .. را در قالب نمودارهایی مشاهده کرد. همچنین با استفاده از این سامانه

می‌توان کانتینرهای هر یک از سیستم‌ها را مشاهده و در صورت لزوم آن‌ها را روشن، خاموش و بازنشانی نمود. با استفاده از این سامانه می‌توان از کانتینرها در زمان اجرا نسخه‌ی پشتیبان تهیه کرد و نسخه‌ی imageای که این کانتینرها از آن ساخته شده‌اند را مشاهده نمود.

این سامانه مدیر پلتفرم را از از محیط CLI و دستور docker بی‌نیاز می‌کند و با قابلیت پشتیبانی از چندین نود این اجازه را به مدیر سامانه می‌دهد که تمام دیتاسنتر خود را به صورت یکپارچه و از یک درگاه مدیریت کند. نحوه کاربری آن در مستندات کاربری آمده است.

۴- پروتکل‌های ارتباطی با اشیاء

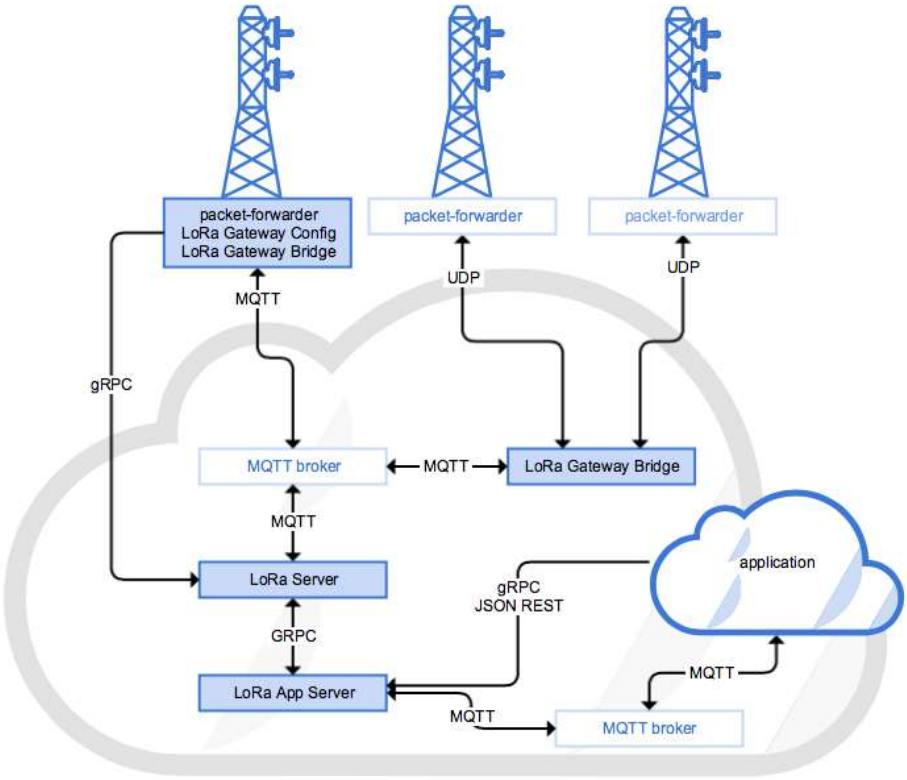
۴-۱- مقدمه

ارتباط با اشیاء یکی از نیازمندی‌های اصلی پلتفرم پیشنهادی است. پروتکل‌های متفاوتی برای این امر وجود دارد. در معماری ارائه شده دو پروتکل زیر قرار دارند :

۱. پروتکل LoRaWAN
۲. پروتکل ارتباط IP از طریق LAN

۴-۲- پروتکل LoRaWAN

در این بخش برای ارتباط با اشیا LoRa از پروژه متن باز LoraServer.io استفاده شد. این پروژه مجموعه ای از برنامه‌های کاربردی متن باز است که امکان ارتباط بین درگاه‌های دریافت/ارسال کننده پیام از/به اشیا LoRa و ابزارها یا پلتفرم‌های مدیریتی را در اینترنت اشیاء فراهم می‌کند. LoRa یک پروتکل در سطح لایه Mac است که ارتباطات بیسیم را فراهم می‌کند و استاندارد LoRaWAN ارتباط میان این اشیا و برنامه‌های کاربردی را استانداردسازی می‌کند. این پروتکل ارتباط بین اجزا را در توان پایین ولی با برد بالا فراهم می‌کند. در شکل ۴-۱ معماری پروژه متن باز LoRaServer.io نمایش داده شده است.



شکل ۴-۱. معماری مدل ارتباطی مبتنی بر LoRaServer

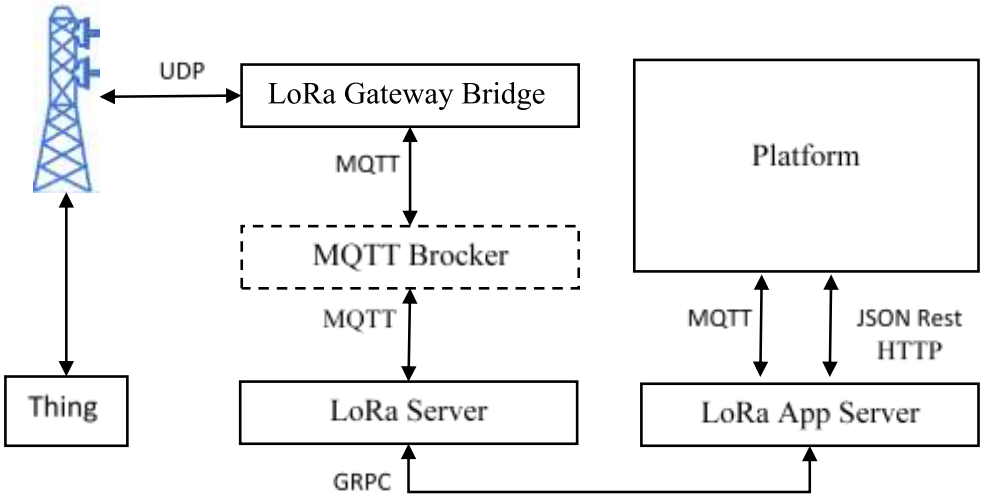
معماری ارائه شده توسط LoRaServer.io از مولفه‌های زیر تشکیل شده است.

- LoRa nodes: اشیایی که داده‌ها را از طریق درگاه ارتباطی به شبکه ارسال و دریافت می‌کنند. این دستگاه‌ها می‌توانند برای مثال حسگرهای اندازه گیری کیفیت هوا، دما و رطوبت یا می‌توانند عملگرهایی مانند شیر برقی و ... باشند.
- LoRa Gateway: این مولفه وظیفه ارسال و دریافت داده از اشیا را بر عهده دارد. درگاه‌ها معمولا از نرم‌افزارهای forwarder بسته‌ها استفاده می‌کنند که می‌تواند داده‌ها را به چند سرور ارسال نماید. در برخی موارد همان-طور که در شکل نیز نشان داده شده است امکان دارد این مولفه LoRa Gateway Bridge را نیز در خود داشته باشد، وظیفه‌ی این مولفه تبدیل داده‌های Gateway به ساختاری جهت ارسال به MQTT می باشد. (در ادامه به این مولفه بیشتر پرداخته می‌شود.)
- LoRa Gateway Bridge: این مولفه مسئول برقراری ارتباط با درگاه ارتباطی است. این مولفه پکت‌های UDP دریافتی از Gateway را به فرمت JSON بر روی MQTT می‌برد. این مولفه مزایای مانند تامین امنیت،

آسان سازی اشکال یابی و ... را داراست. البته استفاده از این مولفه اجباری نبوده و Gatewayها می‌توانند به طور مستقیم داده‌ها را از MQTT ارسال کنند.

- LoRa Server: مسئول هماهنگی تمام مولفه‌ها در شبکه است. این واحد از اشیا فعال در شبکه مطلع است. هنگامی‌که یک شی جدید به شبکه متصل می‌شود این واحد از App server در مورد وضعیت شی اطلاعات می‌گیرد که در صورت لزوم آن دستگاه را در سیستم ثبت کند. موارد دیگر از جمله مدیریت عمل de-duplicat (حذف داده‌های تکراری) داده‌های دریافتی از سمت چند Gateway، اعتبار سنجی این داده‌ها و ارسال آن به سمت Application-Server و در صورت لزوم دریافت پیام برگشتی از آن را برعهده دارد.
- LoRa App Server: این واحد مسئول پیاده‌سازی کاربردهاست به شکلی که با LoRa Server هماهنگ باشند. قابلیت‌های مدیریت اشیا به ازای هر کاربرد و سازمان و مدیریت گذرگاه به ازای هر سازمان را در اختیار قرار میدهد. سایر قابلیت‌های مربوط به مدیریت کاربران و ارتباط آن با کاربردها را نیز فراهم می‌کند. ارتباط با کاربردها از طریق JSON بر روی MQTT فراهم خواهد شد. API مناسبی مبتنی بر JSON Rest جهت توسعه کاربردها در پروژه LoRa تدارک دیده شده است.
- Application: جایگاه کاربردها را نمایش می‌دهد. هر کاربرد از طریق یک عنوان و با استفاده از MQTT داده‌ها را از دستگاه‌ها جمع آوری می‌کند. برنامه کاربردی حتی قادر به ارسال داده از طریق MQTT نیز هست. در صورت لزوم نیز توانایی ارتباط API از gRPC یا JSON Rest را دارد.

بخش‌های ذکر شده از پروژه LoRa شامل Lora Gateway Bridge، MQTT Brocker، LoRa Server و LoRa APP Server در پلتفرم اینترنت اشیا نیز مورد استفاده قرار گرفته است. در حقیقت در حال حاضر بخشی از پلتفرم اینترنت اشیا می‌باشند. در ادامه جایگاه پلتفرم ارائه شده در تعامل با مولفه‌های ذکر شده مورد بررسی قرار گرفته است.



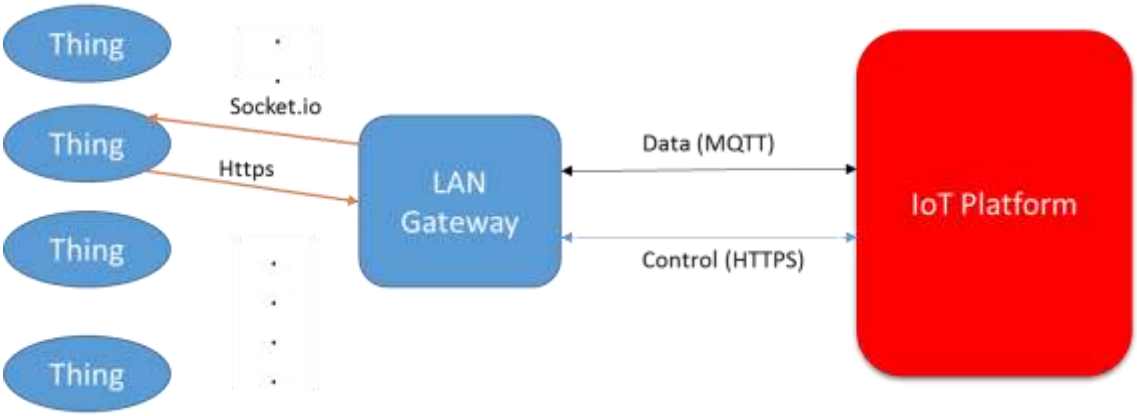
شکل ۴-۲. ارتباط اشیا با پلتفرم از طریق Loraserver.io

همان‌طور که در شکل ۴-۲ نشان داده شده است بخش LoRa App Server مسئول تعامل با پلتفرم است. در حال حاضر در معماری ارائه شده MQTT برای انتقال داده و ارتباط مبتنی بر HTTP به عنوان کنترلی استفاده می‌گردد.

اطلاعات مربوط به تنظیمات کلی پروتکل، تنظیمات شبکه، پروتکلی کنترلی و داده مابین پلتفرم و سرور LoRa و سایر موارد در مستند (IoT-RA-BS-v1.16) ذکر شده است.

۴-۳- پروتکل LAN

دیگر پروتکل ارتباطی با اشیا پروتکل LAN (خانواده IEEE 802 و مشخصا IEEE 802.3، IEEE 802.11 و ...) می‌باشد. شمای کلی ارتباطات شکل ۴-۳ نمایش داده شده است.



شکل ۴-۳. معماری گذرگاه LAN جهت ارتباط با اشیا

لازم به ذکر است که این پروتکل علاوه بر پلتفرم، بر روی اشیاء نیز باید پیاده‌سازی گردد، بنابراین استفاده از پروتکل‌های استاندارد موجود برای این منظور (در عمل) الزامی است. ساختار طراحی در سه بخش زیر آمده است:

۱- اشیاء برای اتصال به LanServer با استفاده از token وارد می‌شوند. برای ارسال داده از اشیاء به پلتفرم، یک درخواست POST با استفاده از HTTPS به LanServer ارسال می‌شود که شامل داده‌ها و کلید شی می‌باشد. سپس این داده‌ها برای ارسال به پلتفرم روی کانال MQTT گذاشته می‌شوند. درخواست POST ارسال شده از شیء به LanServer به آدرس '/push' و با فرمت JSON است که شامل token و data است. این مرحله قابلیت اتصال اشیا به صورت امن را فراهم می‌کند.

۲- برای ارسال داده از پلتفرم به اشیاء، داده‌ها از پلتفرم با استفاده از کانال MQTT به LanServer ارسال می‌شوند. LanServer که در حال گوش دادن به این کانال است با دریافت داده با استفاده از Socket.io داده‌ها را برای شیء ارسال می‌کند. LanServer می‌تواندکلید و داده‌ها را به صورت JSON دریافت می‌کند و به اشیاء ارسال می‌کند. این قابلیت نیز برآورده کردن امکان ارسال داده از اشیا به پلتفرم را بر اساس نیازمندی‌ها تامین کند. البته همانطور که قابلیت encoding برای قسمت LoRa در پلتفرم قرار دارد، می‌تواند از این قابلیت در این قسمت نیز استفاده کرد.

۳- بر روی LanServer یک API با استفاده از HTTP وجود دارد تا دستورات کنترلی به آن ارسال گردد. یکی از دستورات کنترلی، دستور اضافه کردن اطلاعات یک شیء به پایگاه داده‌ی بر روی LanServer است. این API دارای آدرس '/device' و نوع درخواست POST است که حاوی type (نوع درخواست کنترلی) و data (داده‌های مربوط به درخواست) است. با ارسال این درخواست توکن مربوط به شی جدید با استفاده از JWT تولید و برگردانده می‌شود. این قابلیت نیز امکان ارسال داده کنترلی به اشیا از طریق پلتفرم را فراهم می‌کند.

۴-۴- واحدهای مرتبط به انتخاب پروتکل ارتباطی با اشیاء

همانگونه که در شکل ۲-۲ نشان داده شده است برای ارتباط با درگاه‌ها واحد connectivity در پلتفرم طراحی شده است که وظیفه مدیریت اتصال به درگاه‌ها را دارد. در پیاده‌سازی این واحد به عنوان یک بخش انتزاعی دیده شده است و در حقیقت در دل خود Gatewayها پیاده‌سازی شده است.

۵- پروتکل ارتباطی با برنامه‌های کاربردی

۵-۱- مقدمه

یکی از قابلیت‌هایی که پلتفرم در اختیار قرار می‌دهد، امکان توسعه برنامه‌های کاربردی است که می‌توانند از امکانات پلتفرم برای دسترسی به اشیاء و کار با پلتفرم استفاده کنند. همان‌طور که قبلا هم توضیح داده شد، این APIهای از طریق واحد API Server فراهم می‌گردد. هر کاربر با توجه به نیاز خود می‌تواند برنامه خاص خود را توسعه دهد. پلتفرم پیشنهادی واسط‌های برنامه‌نویسی که در قالب یک سری پروتکل پیاده‌سازی می‌شوند را در اختیار این برنامه‌های کاربردی قرار می‌دهد. به عنوان مثال امکان توسعه یک برنامه کاربردی خارج از پلتفرم با قابلیت اتصال به پلتفرم برای مدیریت وجود خواهد داشت.

۵-۲- کلیات پروتکل

کلیات پروتکل پیشنهادی مبتنی بر معماری REST است که در واسط کاربری نیز استفاده شده است. REST مخفف عبارت Representational State Transfer است و متکی بر یک پروتکل ارتباطی بدون حالت[۱]، کلاینت/سرور[۲] و با قابلیت cache کردن می‌باشد. در این معماری از HTTP برای تعامل بین واحدها استفاده می‌شود. ایده اصلی معماری REST این است که به جای استفاده از مکانیزم‌های پیچیده ای مانند CORBA، RPC یا SOAP برای اتصال ماشین‌ها ازHTTP ساده برای برقراری ارتباط بین ماشین‌ها استفاده شود.

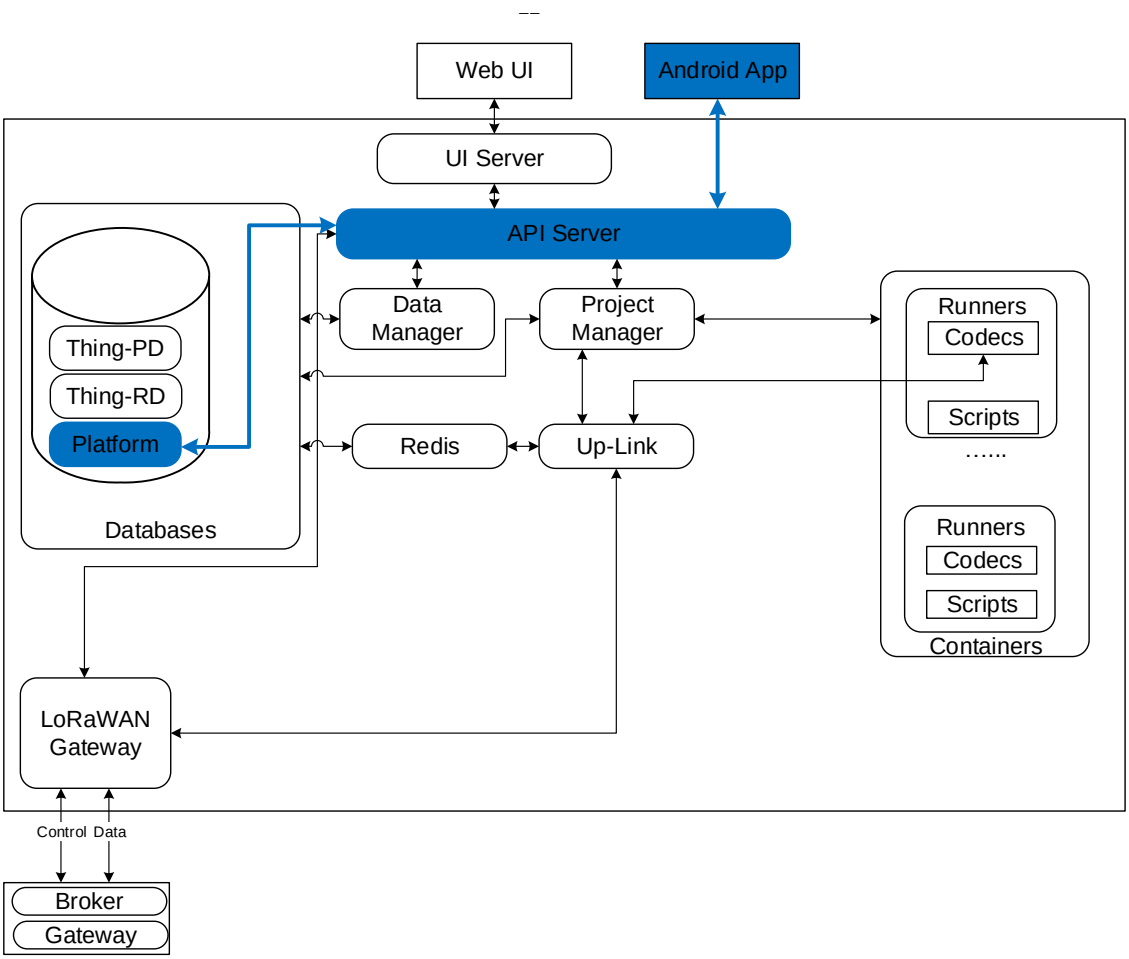
از لحاظ رویکرد برنامه نویسی REST جایگزینی ساده برای سرویس‌های وب است. توسعه‌پذیری در تعاملات میان اجزا، عمومیت واسط ها، توسعه مستقل اجزا و استفاده از واسطه‌ها از کلیدی ترین اهداف معماری REST می‌باشد. لازم به ذکر است که استفاده از معماری REST در برنامه‌نویسی کارایی، سادگی، انعطاف‌پذیری و قابلیت اطمینان را افزایش می‌دهد.

[۱] Stateless

[۲] Client/Server

۵-۳- جزییات API مربوط به برنامه‌های کاربردی

یکی از ویژگی‌های API ارائه شده این است که وابستگی به انواع ابزار هایی که از پلتفرم استفاده میکنند ندارد. توابع API به زبان PHP نوشته شده‌اند و امکان دسترسی به تمامی توابع ارائه شده توسط API-Server که در بخش۳-۲ ارائه شده است توسط برنامه‌های کاربردی به صورت مستقیم وجود دارد. به عنوان مثال سناریوی ثبت نام کاربر از طریق اپلیکیشن اندروید در شکل ۵-۱ نشان داده شده است. در این سناریو کاربر از طریق اپلیکیشن موجود در موبایل خود گزینه ثبت نام را انتخاب می‌کند. در ادامه توابع مربوط به ثبت نام از بخش API-Server فعال می‌گردند و اطلاعات کاربر در سیستم ثبت می‌گردد.



شکل ۵-۱. سناریوی ثبت نام کاربر از طریق اپلیکیشن اندروید

همان‌طور که قبلا ذکر گردید تمام APIهای فراهم شده به صورت Postman در یک فایل الکترونیکی در اختیار کافرما قرار خواهد گرفت تا Exportهای لازم از آن‌ها انجام دهد و بسته به نوع کاربرد مورد استفاده قرار گیرد.

۶- پروتکل ارتباطی با ابزارهای تحلیل داده

۶-۱- مقدمه

با توجه به بررسی‌های انجام شده در سند تحلیل نیازمندی‌ها (IoT-RA-DP-v1.0)، سامانه باید قابلیت اتصال به ابزارهای تحلیل داده را فراهم کند. در حقیقت در پلتفرم‌های اینترنت اشیا با توجه به حجم داده زیادی که از اشیا دریافت می‌کنند امکان اتصال به ابزار تحلیل داده بسیار حائز اهمیت می‌باشد. این سرویس در واقع محل اتصال پلتفرم اینترنت اشیاء به ابزارهای تحلیل داده است که داده‌های جمع‌آوری شده را در اختیار ابزار تحلیل داده قرار می‌دهد.

۶-۲- اتصال به ابزارهای داده

در معماری ارائه شده برای اینترنت اشیاء که در شکل ۲-۲ نشان داده شده است واحد Data Analyses مسئول پیاده‌سازی سرویس اتصال به ابزار تحلیل داده است. با توجه به ارزیابی صورت گرفته در تحلیل نیازمندی‌ها، ابزار Spark برای تحلیل داده در نظر گرفته شد که از زبان Java استفاده می‌کند. ابزار انتخاب شده برای پردازش داده‌های با مقیاس بزرگ مناسب می‌باشد. این واحد از سه API به نام‌های SparkTextRead، SparSQLDataRead و SparkMongoDataRead تشکیل شده است.

SparkTextRead جهت تعامل با منابع داده‌ای متنی در نظر گرفته شده است. در این بخش، برنامه‌های طراحی شده با استفاده از کتابخانه‌های مختلفی مانند map و reduce که توسط ابزار spark فراهم می‌گردد، می‌تواند پردازش‌ها و عملیات مختلفی را بر روی داده‌های منابع متنی انجام دهند. بر حسب نوع داده مورد نیاز، APIهای مختلفی را می‌توان در این قسمت جهت ارتباط با پلتفرم قرار داد.

SparSQLDataRead نیز جهت تعامل با پایگاه داده‌های ساخت یافته پیاده‌سازی شده است. در صورتی که پلتفرم در آینده نیاز به پایگاه داده‌های از نوع SQL داشته باشد از APIهای توسعه یافته در این بخش می‌توان استفاده کرد. جهت اتصال به پایگاه داده SQL نیز از توابع load/save یا توابع jdbc استفاده می‌شود و سپس می‌توان از توابع مختلف ابزار spark استفاده کرد. در حال حاضر نیز اتصالات مربوط به پایگاه داده PostgreSQL فراهم شده است.

SparkMongoDataRead نیز برای اتصال به پایگاه‌داده‌ی Mongo از رابط پیشنهادی آن برای زبان جاوا استفاده می‌کند. بدین ترتیب می‌توان داده‌های جمع‌آوری شده در پایگاه‌داده‌ی پلتفرم را بازیابی کرد. برای اتصال به پایگاه‌داده از کلاس MongoClient استفاده می‌گردد. پس از بازیابی داده‌ها، مانند خواندن از پایگاه‌داده‌های ساخت‌یافته عمل می‌کنیم و پس از آن می‌توان از توابع مختلف ابزار spark استفاده کرد.

۷- حسابرسی

۷-۱- مقدمه

پلتفرم ارائه شده قابلیت ارائه سرویس جهت انجام امور تراکنش‌ها و مدیریت پرداخت کاربران را دارد. برای استفاده رایگان نیز، اکانت یک یا دو ماهه برای یک یا دو سنسور ایجاد گردیده است. با توجه به اینکه پکیج انتخابی به صورت تعداد پروژه و سنسور در ماه می‌باشد نوع پرداخت به شکل prepaid می‌باشد. در ادامه نوع سرویس‌های فراهم شده در این بخش مورد بررسی قرار خواهد گرفت

۷-۲- سرویس‌های ارائه شده در این بخش

سرویس‌های ارائه شده در این بخش دو کاربر نهایی و مدیر پلتفرم را پوشش می‌دهد.

- سمت کاربر
  - خرید بسته توسط کاربر با امکان انتخاب یکی از درگاه‌های بانکی
  - امکان مشاهده و استفاده از کد تخفیف تولید شده توسط ادمین
  - مشاهده وضعیت بسته‌های خریداری شده
  - مشاهده تراکنش‌های انجام شده

- مدیریت بسته‌ها در سمت مدیر پلتفرم(ادمین)
  - نمایش کل تراکنش‌های موفق و ناموفق به ازای کل پلتفرم و به ازای هر کاربر
  - تعریف بسته‌های جدید
  - تعریف کد تخفیف برای کاربران
  - فعال و غیر فعال کردن درگاه‌های پرداخت تعریف شده
  - مشاهده بسته‌های تعریف شده در سیستم

۷-۳- فرایند خرید و پرداخت

نحوه ارتباط با سیستم‌های بانکی و پرداخت از طریق درگاه بانکی است. با توجه به زمانبر بودن دریافت نماد الکترونیکی، در حال حاضر عمل پرداخت به صورت ساختگی از طریق درگاه زرین پال فراهم شده و تست گردید.

۸- پایگاه‌داده

۸-۱- مقدمه

پایگاه داده MongoDB در پلتفرم استفاده شده است. پایگاه داده MongoDB را می‌توان یکی از پرمخاطب‌ترین پایگاه‌های داده موجود در جمع اعضای خانواده NoSQL دانست. این پایگاه داده یک مدل منعطف، پویا و سندگرا را ارائه می‌کند که ساختاری با خروجی بسیار بالا و قابلیت مقیاس‌پذیری آسان را دارا است. این نوع پایگاه داده در پروژه‌ها و سیستم‌هایی که با حجم بزرگی از داده‌ها مواجه هستند مورد نیاز است.

بدلیل متغیر بودن ساختار داده‌ها و حجم زیاد داده‌ها در پلتفرم اینترنت اشیا، یک پایگاه داده‌ی NoSQL که در آن از جدول‌بندی داده‌ها استفاده نمی‌شود، باید استفاده گردد. به همین دلیل یکی از مناسب‌ترین گزینه‌ها استفاده از MongoDB می‌باشد. در این بخش مفاهیم مطرح، طراحی پایگاه داده و همچنین پایگاه داده‌های استفاده شده در سیستم تعریف می‌گردند.

۸-۲- خوشه بندی در پایگاه داده Mongo

در پایگاه داده MongoDB می‌توان داده‌ها را خوشه‌بندی کرد، به این معنا که همزمان چندین پایگاه داده را راه‌اندازی کرده و سپس با سیستم Clustering آن‌ها را مدیریت کرد. این قابلیت مزایای زیر را دارد:

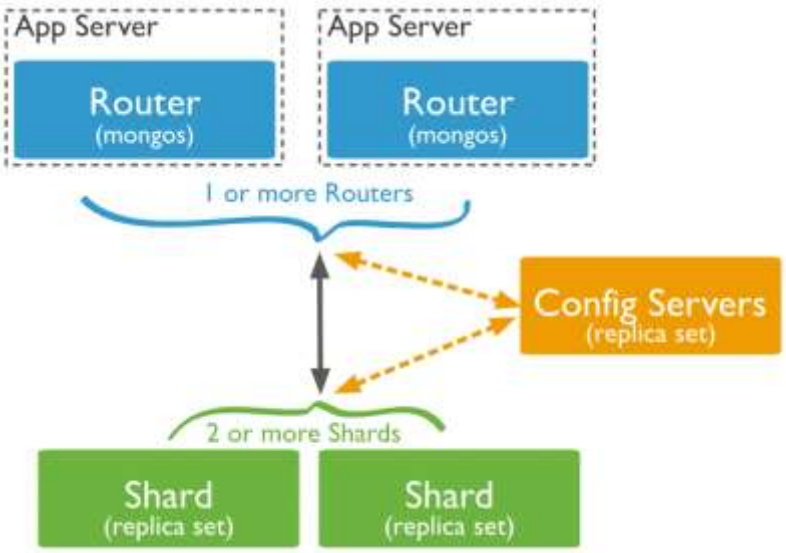
- سرعت خواندن و نوشتن بیشتر
- استفاده بهینه‌تر از فضا
- خواندن و نوشتن به صورت Partial

در این پایگاه داده دو نوع خوشه‌بندی وجود دارد:

- Hashed Sharding: در این نوع خوشه بندی،MongoDB داده‌ها را با روش Hashکردن، میان چندین پایگاه داده تقسیم می‌کند. در این حالت اگر یکی از پایگاه داده‌ها از سرویس خارج گردید و داده‌های آن از بین رفت، بقیه داده‌ها قابل دسترس هستند.
- Ranged Sharding: در خوشه‌بندی مرتب‌سازی‌شده، یک داده به قسمت‌های مختلفی تقسیم شده و در چندین پایگاه داده ذخیره می‌گردد.

با توجه به طراحی فعلی و نوع استفاده از داده‌های پلتفرم در طراحی از نوع Hashed Sharding استفاده شده است. چرا که در این طراحی با توجه به تنوع داده‌ها توزیع بار برابرتری صورت می‌پذیرد.

معماری ارائه شده در شکل ۸-۱ ساختار خوشه بندی ذکر شده را نمایش می‌دهد.



شکل ۸-۱. معمای خوشه‌ای

- config: پایگاه داده مبدا را که بقیه MongoDBها به آن متصل می‌شوند را Config یا همان Configuration می‌نامند که تنظیمات مربوط به Clustering را در خود ذخیره می‌کند.
- Shard: پایگاه داده‌هایی را که به Config متصل می‌شوند و داده‌ها را به صورت خوشه‌بندی شده در خود ذخیره می‌کنند.
- Mongos: این قسمت مثل یک روتر عمل می‌کند و رابطی بین برنامه ی کاربر و شاردهای کلاستر است.

۸-۳- تکثیر پایگاه‌داده

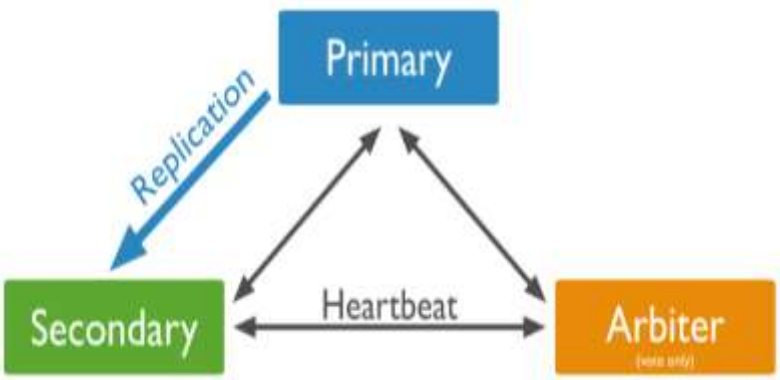
امکان دارد تحت شرایط خاصی، ماشینی که پایگاه داده روی آن اجرا می‌شود از سرویس خارج گردد و داده‌های آن از بین برود. پایگاه داده Mongo برای این موضوع راه‌حل تکثیر (Replication) را پیشنهاد می‌کند، به صورتی که چندین پایگاه داده Mongo راه‌اندازی کرده و یک کپی از هر داده در آن‌ها نگهداری می‌شود.. از مزایای تکثیر می‌توان به موارد زیر اشاره کرد:

- دسترسی پایدار به داده‌ها

- از بین نرفتن داده‌ها
- کاربر می‌تواند درخواست خواندن داده را، همزمان به چندین سرور ارسال کند
- استفاده از نسخه‌های اضافی برای اهداف اختصاصی مانند: بازیابی فوری، تهیه نسخه پشتیبان(Backup) یا گزارش‌دهی

فرآیندها یا پایگاه‌داده‌های Mongo که یک مجموعه داده تکراری را نگه داری می‌کنند با عنوان Replica set معرفی می‌گردد. در هر Replica Set تنها یک پایگاه داده اصلی (primary) وجود دارد. عملیات خواندن در این نوع معماری را می‌توان طوری تنظیم کرد که هم توسط پایگاه داده اصلی و هم توسط پایگاه داده ثانویه (secondary) قابل انجام باشد. در نتیجه می‌توان از طریق افزودن پایگاه داده secondary به Replica set، کارایی در عملیات خواندن از پایگاه داده را افزایش داد. اما در عملیات نوشتن در پایگاه داده، تنها در پایگاه داده اصلی نوشته می‌شود و سپس به سایر پایگاه داده‌های موجود در Replica-set پخش می‌شود. در نتیجه افزودن پایگاه داده‌های ثانویه به مجموعه تاثیر در سرعت عملیات نوشتن ندارد.

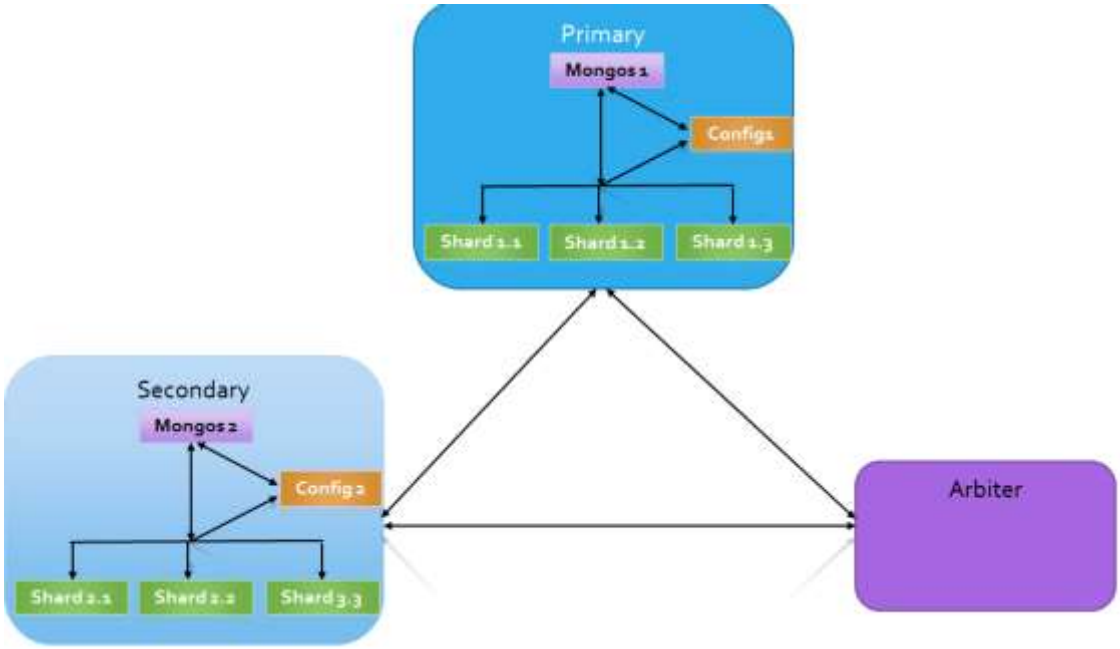
در این نوع معماری در حالتی که چندین پایگاه Secondary وجود داشته باشد، در صورت قطع ارتباط و از دست رفتن پایگاه داده اصلی، سایر اعضا یا Secondaryها در رابطه با انتخاب پایگاه داده اصلی تصمیم گیری می‌کنند. اما در صورتی که در Replica Set تنها پایگاه داده اصلی و ثانویه وجود داشته باشد، از مولفه‌ای به نام داور (Arbiter) استفاده می‌گردد. معماری آن درشکل ۸-۲ نشان داده شده است. Arbiter مانند بقیه اجزا داده‌ای را نگه‌داری نمی‌کند. در این صورت در حالتی که پایگاه داده اصلی از دسترس خارج گردد، Arbiter در انتخاب پایگاه داده ثانویه به عنوان پایگاه داده اصلی نقشی اساسی بازی می‌کند. . باید در نظر داشت که نباید arbiter در سیستم‌هایی که میزبان Replicationهای اصلی و ثانویه است راه‌اندازی گردد.



شکل ۸-۲. معماری تکثیر پایگاه داده

۸-۴- معماری پیشنهادی

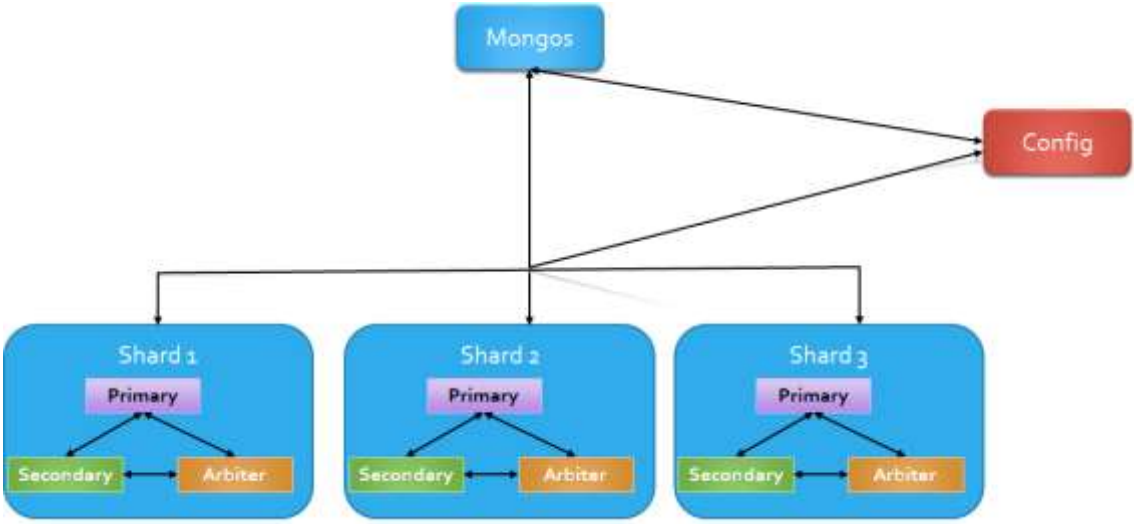
معماری‌های ارائه شده عموما در دو مدل ترکیب می‌گردند. در شکل ۸-۳، پایه و اساس بر اساس معماری تکثیر است ولی در داخل primary و secondary از تکنولوژی sharding استفاده شده است. در این حالت ۶ پایگاه داده جهت خوشه‌بندی و دو Mongo برای Config و پایگاه داده‌ای برای داور بودن راه‌اندازی می‌گردد. و در آخر نیز دو mongos راه اندازی کرده که replSet آن‌ها یکسان قرار می‌گیرد. در این مدل هم سرعت خواندن و هم سرعت نوشتن به صورت جداگانه در primary و Secondary افزایش پیدا کرده است.



شکل ۸-۳. معماری تکثیر با استفاده از خوشه بندی

نوع دیگری از ترکیب معماری در شکل ۸-۴ است. این نوع معماری متداول تر بوده است. در این حالت mongos درخواست‌های دریافتی ما بین shardهای مختلف تقسیم می‌کند. که در داخل هر shard نیز یک Replica-set تعریف شده است که پشتیبانی داده‌های موجود در آن shard را فراهم می‌کند. در این حالت با تقسیم داده‌ها به صورت متناسب ما بین shardها و تقسیم درخواست بر اساس بار هر shard می‌توان سرعت خواندن و نوشتن را بسیار بهبود بخشید. در نهایت availability داده‌ها نیز افزایش پیدا می‌کند. در پلتفرم پیشنهادی این ترکیب از Replication و Sharding پیشنهاد و تست شده است که باعث می‌گرد قابلیت تحمل

و ترمیم نسبت به خرابی بهبود یابد و هم افزایش سرعت و کاهش بار روی ماشین‌های مرتبط به پایگاه داده ایجاد گردد.



شکل ۸-۴. معماری خوشه بندی با استفاده از تکثیر

۸-۵- انواع پایگاه داده‌های پلتفرم

۸-۵-۱- پایگاه داده‌های خام

در این بخش از پایگاه داده اطلاعات سنسورها بدون پردازش خاص ذخیره سازی می‌گردد. در معماری پلتفرم اینترنت اشیاء که در شکل ۲-۲ نشان داده شده است این بخش با نام Thing-RD مشخص شده است.

۸-۵-۲- پایگاه داده‌های پارس شده

در این بخش از پایگاه داده اطلاعات پردازش شده از سنسورها ذخیره سازی می‌گردد. در معماری پلتفرم اینترنت اشیاء که در شکل ۲-۲ نشان داده شده است این بخش با نام Thing-PD مشخص شده است.

۸-۵-۳- پایگاه داده‌های پلتفرم

در این بخش از پایگاه داده اطلاعات مربوط به دیگر بخش‌های پلتفرم اینترنت اشیاء مانند کاربران، پرداخت و پروژه‌ها ذخیره سازی می‌گردد. در معماری پلتفرم اینترنت اشیاء که در شکل ۲-۲ نشان داده شده است این بخش با نام Platform به عنوان یکی از اجزای Databases مشخص شده است.

۸-۶- سایر موارد مرتبط به پایگاه داده‌ها

همانگونه که در شکل ۲-۲ نشان داده شده است برای ارتباط با پایگاه داده در Up-Link از Redis نیز استفاده شده است. این واحد نقش یک واسط (حافظه میانجی) را بین پایگاه داده و واحد Up-Link ایفا می‌کند. این واحد اطلاعات مربوط به اشیاء را پیش از ثبت شدن در پایگاه داده نگهداری می‌کند. به هنگامی‌که حجم داده‌های جمع آوری شده از اشیاء افزایش میابد این واحد در کاهش بار کاری پایگاه داده موثر خواهد بود. سناریوی استفاده از این مولفه در بخش مولفه Up-link شرح داده شده است.

۹- طراحی فرآیند مدیریت اشیا

با توجه به نیازهای کارفرما و طراحی انجام شده در رابطه با مدیریت اشیا که تعریف کامل آن‌ها در فصل ۳ با عنوان مولفه‌ها و فرآیندهای پلتفرم اشیا آورده شده است. اهم موارد مربوط به مدیریت اشیا به شرح زیر می‌باشد:

- مراحل شناسایی و دریافت داده یک شی دارای پیش نیازهایی است
  - ابتدا باید یک پروژه برای آن شی ساخته شود.
  - گذرگاه مربوطه که به شی متصل است تعریف گردد.
  - پروفایل شی ساخته گردد.
- پس از انجام پیش‌نیازهای مربوطه مراحل زیر جهت فعال سازی شی انجام می‌گردد.
  - در مرحله تعریف شی، اطلاعات آن به همراه پروفایل آن شی معرفی می‌گردد.
  - در مرحله آخر فعال سازی شی با استفاده از وارد کردن اطلاعات خاص در لورا (بر اساس ABP و OTAA بودن اشیا) و LAN انجام می‌پذیرد.
- پس از انجام مراحل بالا شی آماده ارسال کدک به آن و نمایش داده می‌باشد.
- در نهایت می‌توان برای یک پروژه سناریو نیز ساخت.

علاوه بر موارد ذکر شده امکانات زیر نیز در رابطه با اشیا در اختیار کاربر قرار داده شده است:

- افزودن، حذف و ویرایش تکی اطلاعات هر شی
- افزودن، حذف و ویرایش اطلاعات اشیا به صورت دستی
- امکان بررسی لاگ گذرگاه جهت عیب‌یابی
- امکان بررسی لاگ کاربران
- و ...

۱۰-ابزارهای آماده گزارشگیری

علاوه بر ابزار آماده Portainer که به عنوان واحد مدیریت داکرها استفاده می‌شود. در سامانه از ابزار فراهم شده توسط Prometheus نیز استفاده شده است. Prometheus یک پروژه متن باز جهت جمع آوری metricها از سیستم‌های مختلف و نمایش آن‌ها در قالب نمودار و ... می‌باشد. این metricها می‌توانند شامل هر کلید و مقداری باشند و می‌توانند از راه‌ها مختلف تولید شوند. یکی از برنامه‌هایی که توسط همین گروه توسعه یافته است exporter نام دارد که وضعیت سیستم فیزیکی شما را در قالب metric در آورده و به Prometheus ارسال می‌کند، که امکان رسم نمودار و .. از سیستم شما را می‌دهد. Prometheus کتابخانه‌هایی به زبان‌ها محتلف دارد که اجازه می‌دهد با استفاده از آن‌ها شما metricهای خاصی از برنامه‌ها خودتان را به ۴۱rometheus ارسال کنید. سامانه حاضر نیز از این قاعده مستثنی نبوده و پارامترهای خاص هر یک از اجزا را در ۴۱rometheus گزارش می‌دهد. نحوه کاربری آن در مستندات کاربری آمده است.

۱۱-پیوست شماره یک

- سناریو ارسال ایمیل

```
from scenario import Scenario


class S1(Scenario):
    def run(self, data=None):
        sender = 'ceitiotlabtest@gmail.com'
        receivers = ['parham.alvani@gmail.com']

        message = 'From: From AIoTRC <ceitiotlabtest@gmail.com>\n' \
                  'To: To Parham Alvani <parham.alvani@gmail.com>\n' \
                  'Subject: Rule Engine Notification [Thing: self.id]\n\n' \
                  'Data:' + str(data) + '\n' \
                                        'Sent by Rule Engine. Scenario:1.'
        self.send_email(host='smtp.gmail.com', port=587,
                        username="ceitiotlabtest",
                        password="ceit is the best",
                        sender=sender,
                        receivers=receivers, message=message)
```

- سناریو ارسال اس ام اس

```
from scenario import Scenario
from kavenegar import KavenegarAPI


class S2(Scenario):
    def run(self, data=None):
        v = self.redis.get("Value")
        if v is None:
            self.redis.set("Value", 0)
        else:
            self.redis.set("Value", int(v) + 1)

        api = KavenegarAPI('7045456F755733434A55456C667144316D6A32734B4D793350764669736F626E')
        params = {
            # optional
            'sender': '',
            # multiple mobile number, split by comma
            'receptor': '09390909540',
            'message': str(v),
        }
        api.sms_send(params)
```

- کدک شماره یک: با اطلاعات Location نود

```
from codec import codec
import cbor


class ISRC(Codec):
    thing_location = 'loc'

    def decode(self, data):
        print("Hello")
        d = cbor.loads(data)

        if 'lat' in d and 'lng' in d:
            d['loc'] = self.create_location(d['lat'], d['lng'])
            del d['lat']
            del d['lng']

        return d

    def encode(self, data):
        return cbor.dumps(data)
```

- کدک شماره دو: بدون اطلاعات Location نودها

```
from codec import Codec
import cbor


class ISRC(Codec):

    def decode(self, data):
        d = cbor.loads(data)
        return d

    def encode(self, data):
        return cbor.dumps(data)
```