پروژه طراحی و پیاده‌سازی پلتفرم اینترنت اشیاء

عنوان:

مستند ساختار کد پلتفرم

ارائه دهنده:

کارگروه پلتفرم، گروه پژوهشی اینترنت اشیاء، دانشگاه صنعتی امیرکبیر

کد سند:

ISRC-AUT-970522.0

تاریخ انتشار:

۱۳۹۷/۰۵/۲۲

اطلاعات سند

نام پروژه:	پروژه طراحی و پیاده‌سازی پلتفرم اینترنت اشیاء
عنوان سند:	مستندات ساختار کد پلتفرم
نام گروه:	گروه تضمین کیفیت و کنترل پروژه
کد سند:	ISRC-AUT-970522.0
نگارش:	۱/۱
نام تهیه کنندگان:	
تاریخ تهیه:	۱۳۹۷/۰۷/۰۲
نام بازبینی کننده:	
تاریخ آخرین بازبینی:	
نام تائیدکننده:	
تاریخ تائید:	
وضعیت:	
تاریخ انتشار:	
نوع طبقه‌بندی سند:	

فهرست مطالب

فهرست شکل‌ها

فهرست جداول

۱- مقدمه

در این سند ساختار کلی کد پلتفرم شرح داده شده است. سند از سه فصل اصلی تشکیل شده است. فصل اول مباحث مربوط به React framework که به عنوان Front-End استفاده شده است را مورد بررسی قرار می‌دهد. در فصل دوم Laravel که بعنوان framework زبان PHP جهت توسعه Back-End سایت بکار رفته تشریح می‌گردد. در نهایت زبان Go که به عنوان زبان توسعه اجزای داخلی پلتفرم استفاده شده شرح داده شده است.

۲- واسط کاربری تحت وب پلتفرم

رابط کاربری پروژه برروی وب و با استفاده از فریم‌ورک React و زبان جاوا اسکریپت انجام شده است. فریم‌ورک React در سال‌های اخیر جزو پرطرفدارترین فریم‌ورک‌های برنامه نویسی رابط کاربری شناخته شده است و در حال حاضر تعداد زیادی از اپلیکیشن‌های تحت وب، موبایل و حتی دسکتاپ با استفاده از این فریم‌ورک گسترش می‌یابند. همچنین در این پروژه از کتابخانه‌ی Redux نیز استفاده شده است که در ادامه معرفی می‌گردد.

در این فصل ابتدا نحوه راه‌اندازی پروژه و مفاهیم اولیه شرح داده شده و سپس نکات مربوط به کد نویسی پروژه توضیح داده شده است. در این فصل فرض بر این است که خواننده با زبان جاوا اسکریپت، فریم‌ورک React و کتابخانه Redux آشنایی دارد.

۲-۱- آشنایی با فریم‌ورک React و نحوه اجرای یک پروژه

React (که به صورت React.js یا ReactJS نیز خوانده می‌شود)، یک فریم‌ورک متن‌باز جاوا اسکریپت برای ساخت رابط‌های کاربری و اجزای(Component) صفحات وب است. این فریم‌ورک توسط فیس‌بوک و اینستاگرام و جامعه‌ای از توسعه‌دهندگان و شرکت‌ها به صورت انفرادی توسعه و نگهداری می‌شوند. براساس آنالیزهای جاوا اسکریپت سرویس Libescore، React در حال حاضر در سایت‌های نت‌فلیکس، Imgur، بلیچر رپورت، فیدلی، ایر بی‌ان‌بی و ... مورد استفاده قرار می‌گیرد.

React و React Native از جمله پروژه‌های متن‌باز شرکت فیس‌بوک هستند که در صدر محبوب‌ترین پروژه‌های وب‌گاه گیت‌هاب قرار دارند.

برای راه‌اندازی یک پروژه برپایه فریم‌ورک React لازم است nodejs و npm برروی سیستم مورد نظر نصب باشد. برای نصب و راه‌اندازی موارد گفته شده می‌توان از منابع زیر استفاده کرد:

- Nodejs: https://nodejs.org/en/download
- npm: https://www.npmjs.com/get-npm

پس از نصب موارد گفته شده باید پروژه React ایجاد گردد. اگر پروژه از قبل ایجاد شده است نیازی به این مرحله نیست. برای ایجاد یک پروژه در React از زیر استفاده می‌شود:

```
npm install -g create-react-app
create-react-app my-app
cd my-app
npm start
```

اگر پروژه از قبل ساخته شده باشد کافیست فقط وارد پوشه اصلی پروژه شده و دستورات زیر را وارد کنید:

```
cd iotrc-platform
npm install
npm start
```

Redux –۲–۲

کتابخانه Redux یک کتابخانه JavaScript متن باز برای مدیریت وضعیت برنامه است که معمولا همراه با کتابخانه‌های React یا Angular برای ایجاد رابط‌کاربری استفاده می‌شود .Redux یک کتابخانه کوچک همراه با یک API ساده و محدود است که برای یک طرف قابل پیش بینی برای حالت برنامه، طراحی شده است. این کتابخانه تحت المان زبان برنامه نویسی کاربردی قرار دارد.

۲–۳– ساختار پروژه

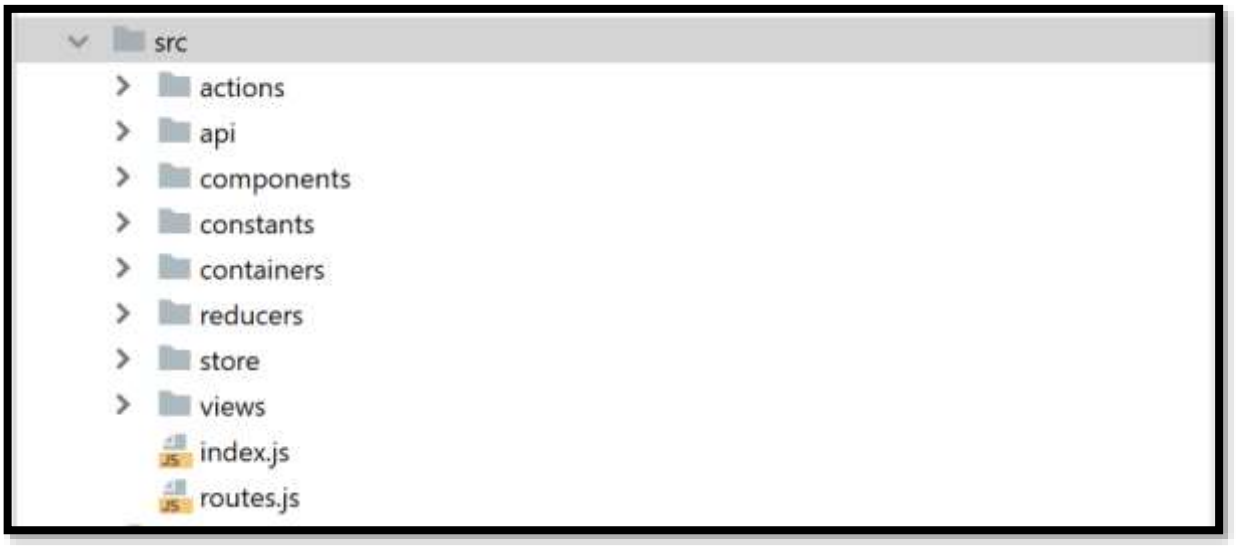
ساختار کلی پروژه در شکل زیر نمایش داده شده است:

شکل ۱- ساختار پروژه (React)

- node_module: در این پوشه کتابخانه‌ها و نیازمندی‌های برنامه که توسط npm نصب شده‌اند قرار دارد. این پوشه توسط برنامه نویس نباید تغییر داده شود.
- Public: در این پوشه فایل هایی که به صورت ایستا توسط وب سرور برای مرورگر ارسال می شود قرار می‌گیرد. این فایل‌ها شامل عکس‌ها و آیکون‌های استفاده شده، فونت‌ها و ... است.
- SCSS: در این پوشه استایل‌های مورد نیاز قرار داده شده‌اند.
- src: در این پوشه سورس کد پروژه قراره گرفته و به زبان جاوا اسکریپت در آن فایل‌های اصلی برنامه قرار داده شده‌اند.

۲-۳-۱- ساختار سورس کد برنامه

پوشه src شامل کدهای اصلی برنامه است که ساختار آن در شکل زیر نشان داده شده است.



شکل ۲- ساختار سورس کد برنامه

کدها براساس نوع عملکرد در پوشه‌های مختلف دسته بندی شده اند که شرح هر کدام به صورت زیر می‌باشد:

- actions: در این پوشه Actionهای مربوط به Redux قرار داده شده است
- api: درخواست‌ها و پاسخ‌ها و ارتباطات با سرور در این پوشه قرار دارد.
- Components: کامپوننت‌های مورد استفاده در رابط‌کاربری مانند کارت‌ها، لیست‌ها و ... در این پوشه تعریف شده‌اند.
- Containers: این پوشه قالب کلی رابط‌کاربری را در بر می‌گیرد.
- Reducers: Reducerهای مربوط به Redux در این پوشه قرار دارند.
- Store: ذخیره سازهای Redux در این پوشه قرار دارند.
- Views: صفحات وب طراحی شده در این پوشه قرار دارند.

در ادامه فایل‌های برنامه در جدول زیر توضیح داده شده‌اند:

جدول ۱- سایر فایل های برنامه

ردیف	نام فایل	محل قرارگیری	توضیحات
۱	AppActions.js	src>actions	در این فایل Actionهای مربوط به Redux نوشته شده است. تمام درخواست‌های ارسال شده به سرور

ردیف	نام فایل	محل قرارگیری	توضیحات
			باید از طریق Actionهای نوشته شده در این فایل انجام پذیرد.
۲	Config.js	src>api	در این فایل تنظیمات مربوط به درخواست های ارسالی به سرور قرار می‌گیرد. این تنظیمات شامل سرایند درخواست ها، احراز هویت درخواست‌ها و ... است.
۳	index.js	src>api	تمام درخواست‌های ارسال شده به سرور ار طریق این فایل انجام می‌شوند. برای هر endpoint در سمت سرور یک متد در این فایل نوشته شده است که ورودی و خروجی درخواست ارسال شده را کنترل می‌کند.
۴	فایل‌های درون پوشه components	src>components	این فایل‌ها بخش‌های مختلف رابط‌کاربری مانند لودینگ، لیست‌ها و جدول‌ها و ... را شامل می‌شود. برای اطلاع از هر کدام از کامپوننت‌ها به آدرس زیر مراجعه کنید: https://reactstrap.github.io/components
۵	Full.js	src> containers>Full	Routeهای برنامه در این فایل نوشته شده اند. هر Route به یک فایل از پوشه views نگاشت می‌شود.
۶	فایل‌های درون پوشهReducers	src>reducers	Reducerهای redux در این فایل‌ها نوشته شده‌اند. هر کدام مسئولیت تغییر در یکی از موجودیت‌های ذخیره شده در store را بر عهده دارند.
۷	Index.js	src>store	ذخیره سازی داده‌ها در این فایل و توسط Redux انجام می‌شود.

۲-۳-۲- فراخوانی سرویس‌ها

برای فراخوانی سرویس‌ها از Redux استفاده شده است ابتدا باید یک متد برای فراخوانی endpoint مربوطه در فایل Api.js ایجاد کنید در این متد، آدرس سرویس، نوع فراخوانی و ورودی‌های query و بدنه فراخوانی را قرار می‌دهید.

برای مثال در کد زیر نمونه فراخوانی از نوع GET قرار داده شده است. همانطور که می‌مشاهده می‌شود آدرس مربوطه و تنظیمات مربوط به فراخوانی GET در این متد قرار داده شده است.

```
module.exports.getThingCodec = function (thingId, dispatch) {
  return fetchData(`/things/${thingId}/codec`, getConfig(), dispatch)
};
```

در کد دیگر، نمونه فراخوانی از نوع POST مشاهده می‌شود. در این نوع فراخوانی علاوه بر آدرس و تنظیمات لازم است بدنه درخواست ارسالی نیز به متد داده شود. خط سوم چگونگی قرار دادن داده در بنده درخواست را نشان می‌دهد.

```
module.exports.testCodecAPI = function (thingId, data, decode, dispatch) {
  const config = postConfig();
  Object.assign(config, {body: getFormData(data)});
  return fetchData(`/things/${thingId}/test?decode=${decode}`, config,
dispatch)
};
```

پس از ایجاد متد مربوطه به فراخوانی سرویس‌ها لازم است یک Redux Action برای اتصال آن به رابط‌کاربری ایجاد شود. در این Action به صورت ناهمگام سرویس‌ها فراخوانی شده و در ذخیره ساز Redux ذخیره شده و یا به صورت مستقیم از طریق callback به رابط‌کاربری بازگردانده می‌شود.

در ادامه نمونه‌ای از Actionهای استفاده شده برای فراخوانی سرویس‌های نمایش داده شده است. در این مثال در خط سوم پس از فراخوانی متد مربوطه، نتیجه باز گردانده شده بررسی می‌گردد. در صورت موفقیت آمیز بودن فراخوانی از طریق callback نتیجه به رابط‌کاربری بازگردانده می‌شود. همچنین مقدار موجودیت User نیز در ذخیره ساز Redux تغییر داده می‌شود (خط ۷).

```
export function editProfile(data, cb) {
  return (dispatch) => {
    const promise = editProfileAPI(data, dispatch)
    promise.then((response) => {
      if (response.status === 'OK') {
        cb && cb(true, 'یافت ویرایش باموفقیت')
        dispatch(updateUser(response.result))
      } else {
        cb && cb(false, response.result)
      }
    }).catch((err) => {
      console.log(err)
    })
  }
}
```

۳- سرویس رابط برنامه‌نویسی کاربردی (API Server)

رابط برنامه‌نویسی کاربردی در این پروژه با فریم‌ورک Laravel که بر روی زبان PHP است، نوشته شده است. Laravel یکی از فریم‌ورک‌های زبان PHP است که برای توسعه کاربردهای وب در نظر گرفته شده است و بر پایه مدل MVC[1] کار می کند. فریم ورک Laravel، برنامه نویسی برنامه های کاربردی تحت وب با زبان PHP را ساده‌تر می‌نماید و کمک بسزایی برای انجام پروژه های PHP و توسعه آسان آن‌ها می کند. فریم‌ورک Laravel بر روی اجزای مختلف فریم‌ورک سیمفونی ساخته شده است و به برنامه شما پایه‌ای بزرگ از کد های قابل اعتماد و تست شده می دهد. Laravel مجموعه ای از بهترین راه حل ها با نحو یا syntax پر معنا و خلاقانه را ارائه می‌دهد که به درستی انجام می پذیرند. Laravel توسط آسان‌سازی کارهای معمول مانند احراز هویت، روتینگ، جلسه‌ها ، کار با بانکهای اطلاعاتی و ... که تقریبا در تمامی پروژه های تحت وب استفاده می‌شوند، مسائل و مشکلات ناشی از توسعه را هم برای توسعه دهنده و هم برای کارفرما کاهش می‌دهد.

لازم به ذکر است Laravel، Symfony، CakePHP و CodeIgniter از محبوب ترین فریم‌ورک‌های زبان PHP هستند که بررسی میزان محبوبیت آنها در گوگل ترند حاکی از رشد روز افزون فریم‌ورک Laravel است. در این پروژه برای نگهداری داده‌های کاربران، بسته‌ها، اشیا، گذرگاه‌ها، پروژه‌ها و سایر اطلاعت پلتفرم از پایگاه‌داده Mongodb استفاده شده است. فریم‌ورک Laravel به صورت پیش‌فرض از Mysql استفاده می‌کند و برای استفاده از Mongodb از کتابخانه jenssegers/mongodb استفاده شده است.

در این فصل ابتدا اجزای پروژه و سپس نحوه پیاده‌سازی رابط‌های کاربری برنامه‌نویسی شرح داده می‌شود.

۳-۱- نصب و راه‌اندازی

فریم‌ورک Laravel همانند بقیه پروژه‌های PHP با استفاده از سیستم مدیریت وابستگی‌های Composer نصب و مدیریت می‌شود. برای نصب Laravel ابتدا کامپوزر نصب کرده سپس از دستور زیر استفاده می‌گردد:

```
composer create-project --prefer-dist laravel/laravel iot-back
```

[1] Model View Controller

سپس برای نصب رابط Mongodb دستور زیر را باید وارد کرد:

```
composer require jenssegers/mongodb
```

۳-۲- ساختار پروژه

Laravel یک فریم‌ورک بسیار سطح بالا است و ساختار پروژه مشخص است. در ادامه به قسمت‌هایی از پروژه که توسط تیم توسعه تغییر یافته پرداخته می‌شود.

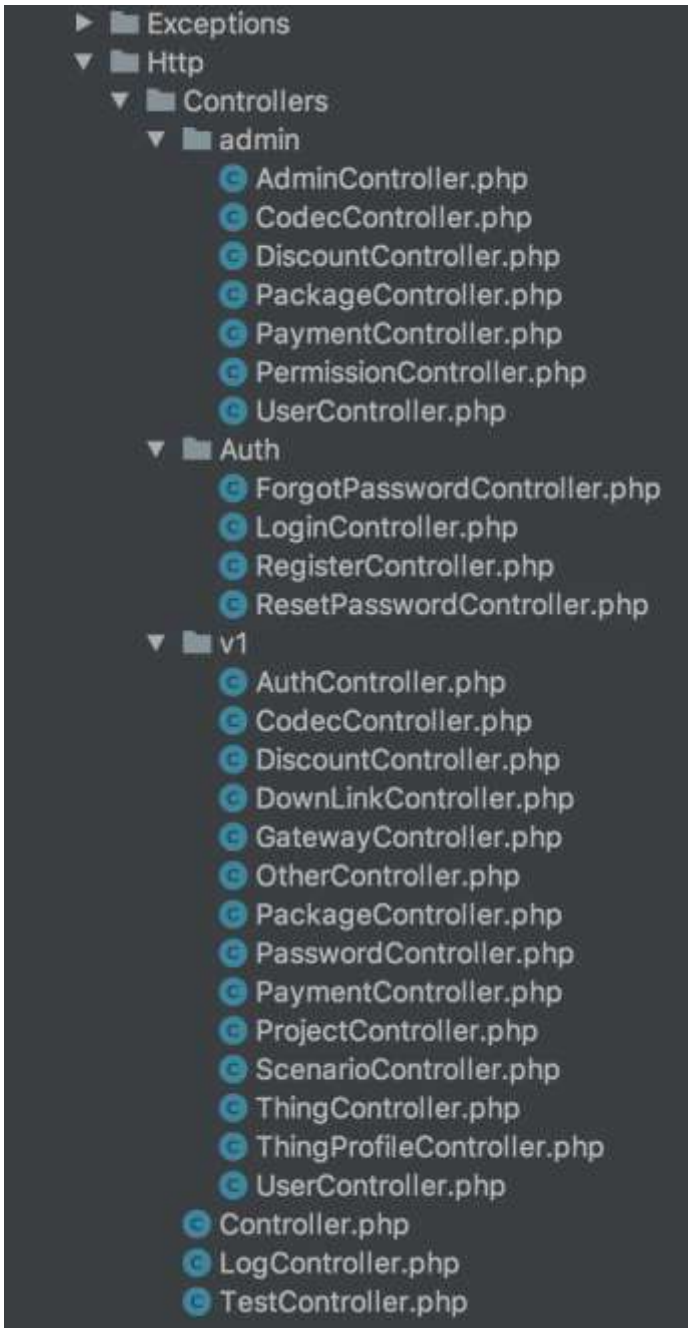
قسمت های اصلی کد به شرح زیر می‌باشد که در ادامه هر کدام تشریح می‌گردد.

- کنترلرها (controller)
- میان‌افزارها (middleware)
- سرویس‌ها (services)
- مسیرها (routes)
- مدل‌ها (models)
- فایل تنظیمات (environment)

۳-۲-۱- کنترلرها

کنترلرها به عنوان یکی از اساسی‌ترین و مهم‌ترین بخش‌های یک وب سایت و یا کاربرد در نظر گرفته می‌شوند.. در الگو و معماری MVC، حرف سوم (c) معادل عبارت Controller است. کنترلرها به عنوان یک واسط بین کاربر، View و Model عمل می‌کنند و به عبارتی قلب تپنده‌ی یک کاربرد می‌باشند. در این پروژه با توجه به عدم وجود View، کنترلرها اصلی‌ترین قسمت کد هستند. در ادامه توضیح داده خواهد شد که چگونه هر URL یا Route به یک تابع در یک کنترلر نگاشت می‌شود.

کنترلرها را با توجه به توابع مربوط به هم که معمولا این توابع مربوط به هم برای یک مدل هستند جدا و طبقه‌بندی می‌کنند. در یک سطح بالاتر این کنترلر ها بنا بر ورژن کاربرد و کاربر عادی بودن یا ادمین بودن جدا سازی میشود که در شکل زیر مشاهده می‌گردد.



شکل ۳- نحوه قرار گیری کنترلرها

در جدولی که در ادامه آمده است لیست کنترلرهای این پروژه و توضیح هر یک مشاهده میشوند که در مسیر app/Http/Controllers قرار دارند.

شایان ذکر است کنترلرهایی که در جدول نیامده فاقد تابع مورد استفاده می‌باشند و صرفا برای آینده هستند. پوشه Auth هم شامل کنترلرهای پیش‌فرض Laravel می‌باشد که استفاده نشده‌اند.

جدول ۲- کنترلرهای ادمین

نام کنترلر	توضیحات
Admin/AdminController	توابع عام مربوط به پنل مدیریتی
Admin/CodecController	توابع مدیریت کدک‌های گلوبال
Admin/DiscountController	توابع مربوط به کدهای تخفیف و مدیریت آن‌ها
Admin/PackageController	توابع مربوط به مدیریت پکیج‌ها و مدیریت آن‌ها
Admin/PaymentController	توابع مربوط به پرتال‌های پرداخت و تراکنش‌های برای مدیر پلتفرم
Admin/PermissionController	توابع مربوط به ساخت پرمیشن‌ها و نقش‌ها و مدیریت آن‌ها
Admin/UserController	توابع مربوط به مدیریت کاربران سامانه.

جدول ۳- کنترلرهای نسخه اول

نام کنترلر	توضیحات
v1/AuthController	توابع عام مربوط به احراز هویت و گرفتن توکن و ثبت‌نام
v1/CodecController	توابع مربوط به ساخت ویرایش حذف و ارسال کدک برای اشیا
v1/DownlinkController	توابع مربوط به ارسال داده به شی
v1/GatewayController	توابع مربوط به مدیریت گذرگاه ها
v1/OtherController	توابع عام پلتفرم
v1/PasswordController	توابع مربوط به بازیابی رمز عبور
v1/ProjectController	توابع مربوط به ساخت ویرایش حذف و مدیریت پروژه‌ها
v1/ScenarioController	توابع مربوط به ساخت ویرایش حذف و ارسال سناریوها
v1/ThingController	توابع مربوط به ساخت ویرایش حذف و مدیریت اشیا
v1/ThingProfileController	توابع مربوط به مدیریت پروفایل اشیا لورا.
v1/PaymentController	توابع مربوط به پرداخت و تراکنش‌های هر کابر
v1/UserController	توابع مربوط به پروفایل کاربر

نام کنترلر	توضیحات
v1/PackageController	توابع مربوط به بسته‌ها و استفاده کاربر از آنها

۳-۲-۲- میان افزارها

میان‌افزار مکانیزم مناسبی ارائه می‌دهد که به وسیله آن می‌توان درخواست‌های HTTP را قبل از ورود به برنامه فیلتر کرد. برای مثال، Laravel شامل یک میان‌افزار است که بوسیله آن می‌توان مشخص کرد که کاربر برنامه به درستی احراز هویت شده است یا خیر. اگر کاربر تایید نشده باشد، میان‌افزار درخواست را با کد ۴۰۳ پاسخ داده و به کنترلر نخواهد رسید. در ادامه بیشتر به میان‌افزار در Laravel پرداخته خواهد شد.

البته، میان‌افزار های دیگری نیز برای انجام کارهای مختلف، علاوه بر احراز هویت نوشته شده است. برای مثال، یک CORS middleware مسئول اضافه کردن سرصفحه‌های مناسب به تمام پاسخ‌هایی است که از برنامه خارج می‌شوند. یک میان‌افزار ثبت وقایع، تمام درخواست‌های ورودی به برنامه را ثبت می‌کند.

در فریم‌ورک Laravel چند میان‌افزار به صورت پیش فرض وجود دارند که از جمله آن‌ها می‌توان به میان‌افزار احراز هویت و حفاظت CSRF اشاره کرد. همه این میان‌افزارها در دایرکتوری app/Http/Middleware قرار دارند.

در این پروژه از میان افزارهای پیش‌فرض Laravel استفاده نشده است. میان افزارهای پروژه در جدول زیر آمده است:

جدول ۴- میان افزارهای پروژه

نام کنترلر	توضیحات
AdminMiddleware	میان افزار ادمین که وظیفه جلوگیری از دسترسی کاربران عادی به قسمت‌های مدیریتی را دارد.
AuthJwtMiddleware	میان‌افزار اختصاصی پلتفرم که با توکن و مکانیزم JWT کار می‌کند و وظیفه احراز هویت کاربران را دارد.
CrossMiddleware	میان افزار Cross Origin برای اضافه کردن Header برای صدا زدن واسطهای کاربری از هر دامنه دیگر.

۳-۲-۳- سرویس‌ها

سرویس‌ها به زبان ساده یک سطح از انتزاع (abstraction) برای کنترلرها هستند. یعنی در توابع کنترلرها توابع سرویس‌ها فراخوانی میشوند به طوری که جزییات پیاده‌سازی در سرویس‌ها و مسیر کلی انجام فعالیت هر رابط کاربری برنامه نویسی در کنترلر است.

برای هر کنترلر یک سرویس وجود دارد و علاوه بر این‌ها یک سری سرویس برای ارتباط با قسمت های مختلف پلتفرم وجود دارد از جمله Core Backend و Lora App Server که وظیفه هماهنگ نگه داشتن پلتفرم با سایر قسمت ها را برعهده دارند. توابع این دو سرویس در کنترلرهای مختلف مورد استفاده قرار گرفته است. این سرویس‌ها در فولدر app/Repository/services قرار دارند.

شکل ۴- نحوه قرار گیری سرویس‌ها

۳-۲-۴- مسیرها

به کمک مسیریابی در Laravel می‌توان مشخص کرد که در صورت فراخوانی یک URL چه تابعی از چه کنترلری فراخوانی شده و چه پاسخی به کاربر ارسال شود. به نوعی وظیفه نگاشت URL به توابع کنترلرها بر عهده دارند.

تمام مسیرهای Laravel در فایل‌های route که در دایرکتوری routes قرار می‌گیرند، تعریف شده است. این فایل‌ها به صورت خودکار توسط فریم‌ورک بارگذاری می‌شوند. فایل routes/web.php مسیرهایی را برای رابط وب تعریف می‌کند. این مسیرها به گروه web middleware اختصاص داده می‌شوند که قابلیت‌هایی مانند حالت جلسه و حفاظت CSRF را ارائه می‌دهند. مسیرهای موجود در routes/api.php در گروه middleware api قرار می‌گیرند.

دراکثر برنامه‌های کاربردی Laravel، بایستی کار خود را با تعریف مسیرها در فایل routes/web.php شروع کنید. می‌توان به مسیرهای تعریف شده در فایل routes/web.php با وارد کردن URL مسیر تعریف

شده در مرورگر، دسترسی داشت. به عنوان مثال، اگر در مرورگر خود آدرس http://your-app.dev/user را وارد کنید، می‌توانید به مسیر زیر دسترسی داشته باشید.

```
1 | Route::get('/user', 'UserController@index');
```

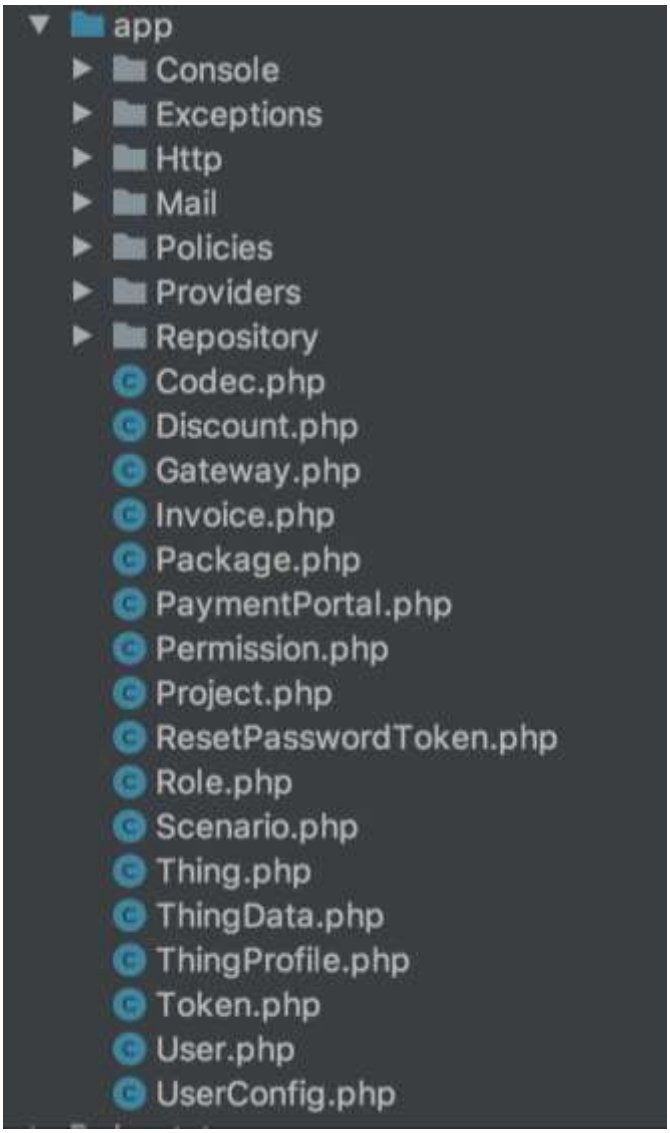
برای مثال لیست تعدادی از این مسیرها را در شکل زیر مشاهده می‌کنید. لیست کامل مسیرها در Postman پروژه وجود دارد.

Method	URI	Action
GET|HEAD	/	Closure
POST	api/admin/codec	App\Http\Controllers\admin\CodecController@create
GET|HEAD	api/admin/codec	App\Http\Controllers\admin\CodecController@list
GET|HEAD	api/admin/codec/{codec}	App\Http\Controllers\admin\CodecController@get
DELETE	api/admin/codec/{codec}	App\Http\Controllers\admin\CodecController@delete
PATCH	api/admin/codec/{codec}	App\Http\Controllers\admin\CodecController@update
POST	api/admin/discount	App\Http\Controllers\admin\DiscountController@create
GET|HEAD	api/admin/discount	App\Http\Controllers\admin\DiscountController@all
DELETE	api/admin/discount/{discount}	App\Http\Controllers\admin\DiscountController@delete
GET|HEAD	api/admin/packages	App\Http\Controllers\admin\PackageController@all
POST	api/admin/packages	App\Http\Controllers\admin\PackageController@create
PATCH	api/admin/packages/{package}	App\Http\Controllers\admin\PackageController@update
DELETE	api/admin/packages/{package}	App\Http\Controllers\admin\PackageController@delete
GET|HEAD	api/admin/packages/{package}/activate	App\Http\Controllers\admin\PackageController@activate
GET|HEAD	api/admin/payment	App\Http\Controllers\admin\PaymentController@list
GET|HEAD	api/admin/payment/overview	App\Http\Controllers\admin\PaymentController@overview
GET|HEAD	api/admin/payment/portals	App\Http\Controllers\admin\PaymentController@portals

شکل ۵- نمونه ای از مسیرهای پروژه

۳-۲-۵- مدل‌ها

مدل‌ها در واقع تعریف موجودیت‌های قابل ذخیره سازی در پایگاه داده هستند که یک سری روابط یک به یک، یک به چند و یا چند به چند دارند. هر مدل دارای یک سری ویژگی‌ها است که در فایل آن مشخص شده است. مدل ها در دایرکتوری app هستند. در شکل زیر لیستی از مدل های پروژه آورده شده است:



شکل ۶- لیستی از مدل های پروژه

۳-۲-۶- فایل تنظیمات

در فولدر پروژه یک فایل env. وجود دارد که اطلاعات و تنظیمات محیط اجرا شدن اپلیکیشن در آن وجود دارد. لیست تنظیمات کلیدی در جدول زیر آورده شده است:

جدول ۵- لیست تنظیمات

نام متغیر	توضیحات
APP_KEY	یک رشته ۴۰ کارکتری مورد استفاده در رمزنگاری‌ها
APP_DEBUG	در صورت فعال بودن خطاها با پشته به صورت کامل نشان داده می‌شود
APP_URL	آدرس اپلیکیشن
DB_HOST	آی پی یا دامنه پایگاه داده
DB_PORT	پورت پایگاه داده
DB_DATABASE	نام کلکسیون پایگاه داده مونگو
DB_USERNAME	نام کاربری پایگاه داده
DB_PASSWORD	رمز عبور پایگاه داده
CLIENT_ORIGIN	در صورت استفاده از دامنه متفاوت برای CROS
MAIL_HOST	آی پی یا دامنه ایمیل سرور
MAIL_PORT	پورت ایمیل سرور
MAIL_PASSWORD	رمز عبور ایمیل سرور
NOCAPTCHA_SECRET	کلید پنهان ری‌کپچا
NOCAPTCHA_SITEKEY	کلید سایت ری‌کپچا
KAVENEGAR_API_KEY	کلید واسط کاربری کاوه نگار برای اس ام اس
LORA_BASE_URL	آدرس LoRa app server
CORE_BASE_URL	آدرس Core Backend
CORE_SECRET	رمز عبور فراخوانی های Core Backend
LORA_ORG_ID	شناسه Organization در لورا سرور
LORA_NET_SERVER_ID	شناسه Network server در لورا سرور
LORA_SERVICE_PROFILE_ID	شناسه Service Profile در لورا سرور
LAN_BASE_URL	آی پی یا دامنه LAN Server
ZARINPAL_SANDBOX	محیط ایزوله یه عملیاتی زرین‌پال
ZARINPAL_MERCHANTID	کلید درگاه زرین پال
FRONT_URL	آدرس کلاینت اپلیکیشن
PROMETHEUS_URL	آدرس پنل پرومتئوس

PORTAINER_URL	آدرس پنل پورتینر

۳-۲-۷- لیست رابط‌های توسعه برنامه‌های کاربردی

لیست APIهایی که توسط این سرور در اختیار واسط کاربری قرار می‌گیرد در آدرس ذیل قرار داد.

https://documenter.getpostman.com/view/74948/RVftkXdi#a1c32890-1865-d97d-7bf0-6f00362f7efe

در پیوست ۱ این مستند، فایل JSON معادل request/response این APIها آمده است.

۴- هسته پلتفرم

هسته پلتفرم که از آن به backend پلتفرم نیز نامه برده می‌شود، به زبان Go توسعه داده شده است. زبان Go است توسط شرکت Google در سال ۲۰۰۹ ارایه شد و یکی از طراحان اصلی آن Ken Thampson یکی از دو مبدع زبان C است. این زبان با تجربه زبان‌های قبلی و با این دید که بتواند مشکلات Google را برای توسعه سامانه‌های مبتنی بر شبکه با نرخ پردازش زیاد و مقیاس‌پذیری بالا طراحی شده است. شایان ذکر است وظیفه اصلی بخش backend پلتفرم اینترنت اشیاء، ارتباط با اشیاء به منظور جمع‌آوری داده از اشیاء و ارسال دستورات به اشیاء است. با توجه به تعداد بسیار زیاد اشیاء، مقیاس‌پذیری و پشتیبانی مناسب از پردازش‌های همزمان و موازی یکی از نیازمندی‌های اصلی در پلتفرم اینترنت اشیاء است که با نیازمندی‌های طراحی زبان GO هم‌پوشانی خوبی دارد.

راهنمای نصب و استفاده از زبان Go در وبسایت https://golang.org/doc/install وجود دارد. برای نصب آن می‌توان آخرین نسخه آن را برای نسخه Ubuntu از سایت دانلود کرده و با استفاده از دستور tar از حالت فشرده خارج کنید. در ضمن باید متغیرهای محیطی آن را نیز تنظیم کنید. همچنین می‌توانید با دستور apt-get install golang[version] نسخه دلخواه زبان Go را نصب کرد.

در این زبان آنچه توسط پکیج‌ها در اختیار سایرین قرار می‌گیرد با حرف بزرگ آغاز می‌شود. مثلا تابع New که با N بزرگ می‌باشد در خارج از پکیج نیز قابل دسترس می‌باشد و این در حالی است که تابع private که با حرف بزرگ آغاز نشده است در خارج از پکیج خود قابل دسترس نمی باشد. در کد توسعه داده شده تمامی آنچه توسط پکیج‌ها در اختیار دیگران قرار می‌گیرد و ماژول‌ها دارای کامنت می‌باشند. برای درک بهتر کد، آشنایی کلی با زبان Go به خصوص مفهوم موازی‌سازی و کانال‌ها ضروری است.

۴-۱- ساختار پروژه

در ادامه مولفه‌های پروژه که هم نام با عناصر به کار رفته در معماری پلتفرم می‌باشد تشریح شده است. تمامی مولفه‌ها دارای مجموعه‌ای از تنظیمات می‌باشند. به صورت کلی نیازی به تغییر این تنظیمات وجود ندارد اما در صورت نیاز این تنظیمات از طریق environment variableها قابل تغییر می‌باشند و لیست آن‌ها نیز در کنار هر مولفه در فایل env.example. آمده است.

۴-۱-۱- مولفه‌ی PM

این مولفه وظیفه‌ی ساخت و مدیریت پروژه‌ها و اشیا را بر عهده دارد. پروژه‌ها در قالب دو داکر مجزا ساخته می‌شود که یکی حاوی مولفه‌ی runner و دیگری redis می‌باشد. این دو مولفه در کنار یکدیگر پروژه‌ی کاربر را شکل می‌دهند که بستر اجرای سناریوها و کدک‌های او می‌باشد. مولفه‌ی pm از یک پایگاه داده‌ی mongo برای ذخیره‌سازی اطلاعات اشیا و پروژه‌های کاربر استفاده می‌کند. این اطلاعات با استفاده از یک API که مبتنی بر REST عمل می‌کتد در اختیار سایر مولفه‌های قرار می‌گیرد و به آن‌ها این امکان را می‌دهد که پروژه‌ها و اشیا را ساخته یا پاک کنند.

برای اجرای این مولفه می‌توان از داکر آن استفاده کرد یا آن را از source اجر نمود. در هر حالت توجه شود که نیاز است دسترسی ماژول به داکر و ایمیج‌ها فراهم شده باشد. این مولفه از پکیج‌های زیر ساخته شده است.

جدول ۶- لیست پکیج‌های PM

پکیج	توضیحات
Actions	پیاده‌سازی APIها در این پکیج صورت می‌گیرد.
Runner	پیاده‌سازی رابط داکر در جهت ساخت پروژه در این پکیج می‌باشد.

تنظیمات این مولفه DB_URL می‌باشد که آدرس پایگاه داده‌ای mongo این مولفه می‌باشد.

۴-۱-۲- مولفه‌ی GoRunner

GoRunner مولفه‌ای است که در داکر اجرا می‌شود و سرویس‌های codec و scenario را برای کاربر فراهم می‌آورد. این مولفه از دو زبان Go و Python تشکیل شده است. زبان Python برای پیاده‌سازی کدهایی می‌باشد که کتابخانه‌های کاربر در codec و scenario را فراهم می‌آورند. در مستند کتابخانه‌های برنامه‌نویسی به تفصیل توابع قابل فراخوانی تشریح شده‌اند. این مولفه از پکیج‌های زیر تشکیل شده است:

جدول ۷- لیست پکیج‌های مولفه GoRunner

پکیج	توضیحات
actions	پیاده‌سازی APIها در این پکیج صورت می‌گیرد.
codec	پیاده‌سازی عملیات‌های encode و decode در این پکیج می‌باشد.
runner	پیاده‌سازی runner در این پکیج می‌باشد.
scenario	پیاده‌سازی سناریو و سرور RPC آن در این پکیج می‌باشد.
runtime.py	این پکیج در واقع یک پروژه به زبان python می‌باشد. این پروژه رابط پایگاه داده‌ای، ارسال پیامک و ایمیل و ... را برای سناریوهای کاربر فراهم می‌آورد.

تمامی تنظیمات این مولفه در زمان ساخت پروژه و به صورت اتوماتیک صورت می‌گیرد.

۴-۱-۳- مولفه‌ی uplink و downlink

این مولفه‌ها وظیفه ارسال و دریافت اطلاعات از لایه‌ی پایین را برعهده دارند. ارتباط با لایه‌ی پایین از طریق پروتکل mqtt و با لایه‌ی بالا از طریق http صورت می‌گیرد. این مولفه داده‌ها را از uplink دریافت می‌کند، آن‌ها را decode می‌کند و در نهایت در پایگاه داده ذخیره می‌کند. در جهت downlink داده‌ها را از طریق http دریافت کرده و آن‌ها را encode می‌کند. در جهت افزایش سرعت در uplink یک pipeline پیاده‌سازی شده است. Pipeline در Go مجموعه‌ای از توابع هستند که می‌توانند به صورت موازی اجرا شده و از طریق کانال‌ها ارتباط برقرار می‌کنند. این pipeline به ترتیب در زمان دریافت داده آغاز شده، داده را در ساختار داده‌ای پلتفرم قرار می‌دهد و پس از یافتن پروژه مرتبط با آن شی داده‌ی آن را رمزگشایی کرده و در نهایت در پایگاه داده‌ی mongo ذخیره می‌کند. این مولفه‌ها پکیج‌های زیر را دارا می‌باشد:

جدول ۸- لیست پکیج‌های مولفه‌های uplink و downlink

پکیج	توضیحات
actions	پیاده‌سازی APIها در این پکیج صورت می‌گیرد.
decoder	پیاده‌سازی رابط با pm در جهت رمزگشایی داده‌ها از طریق پروژه کاربر
encoder	پیاده‌سازی رابط با pm در جهت رمزگذاری داده‌ها از طریق پروژه کاربر

این مولفه‌ها تنظیمات اصلی زیر را دارا می‌باشند:

جدول ۹- تنظیمات مولفه های uplink و downlink

نام تنظیم	توضیحات
DB_URL	آدرس مولفه‌ی pm که عمل decode و encode را انجام می‌دهد.
BROKER_URL	آدرس mqtt broker که دریافت و ارسال داده‌ها با لایه‌ی پایین از طریق آن صورت می‌گیرد.

۴-۱-۴- مولفه‌ی DM

این مولفه وظیفه‌ی مدیریت داده‌ها را برعهده دارد. همانطور که پیشتر اشاره شد داده‌ها در مولفه‌ی uplink در پایگاه داده‌ای ذخیره می‌شوند و می‌توان به آن‌ها دسترسی داشت. وظیفه‌ی اصلی این مولفه در واقع فراهم آوردن بخشی از پرس‌وجوهایی است که نیاز به آن‌ها در سامانه بیشتر خواهد بود. این پروژه بدون پکیج بوده و دارای یک main می‌باشد.

این پروژه راه ارتباطی با loraserver نیز دارا بوده که در کنار وظیفه‌ی اصلی آن امکان جمع‌آوری لاگ‌های گیت‌وی را به آن می‌دهد. این ارتباط از طریق کلاینت loraserver صورت می‌گیرد.

تنظیمات این مولفه DB_URL می‌باشد که آدرس پایگاه داده‌ای mongo این مولفه می‌باشد.

۴-۱-۵- مولفه‌ی GM

این مولفه وظیفه‌ی مدیریت gateway را برعهده دارد. کارکرد اصلی این مولفه در پلتفرم رمزگشایی داده‌ها می‌باشد. این امر به کاربر امکان می‌دهد که در صورت نیاز بتواند داده‌های خام دریافت شده از gateway را به صورت دستی رمزگشایی نماید و به این ترتیب از صحت کلیدها و داده اطمینان حاصل کند.

۴-۱-۶- پایگاه داده

ساختار پایگاه داده در Core به شکل زیر می‌باشد:

Database: isrc

جدول ۱۰- ساختار پایگاه داده Core

نام collection	توضیحات
pm	اطلاعات پروژه‌ها در این قسمت ذخیره می‌گردد.
data	داده‌های دریافتی از اشیا در این قسمت ذخیره می‌گردد.
errors	خطاها و رویدادهای پروژه‌های کاربر در این قسمت ذخیره می‌گردد.

۴-۲- اجرای مولفه‌ها

در جهت سهولت در اجرای مولفه‌ها و نیازمندی‌های نرم‌افزاری پلتفرم اسکریپت start.sh در پروژه فراهم شده است. با اجرای دستور زیر می‌توانید راهنمای این اسکریپت را مشاهده نمایید:

```
./start.sh
```

در ادامه آنچه از این اسکریپت کاربردی می‌باشد تشریح می‌گردد.

- اجرای مولفه‌ی pm:

```
./start.sh pm up -d
```

- اجرای پروژه‌های کاربر (پروژه‌های در حال اجرا بدون تاثیر می‌مانند.)

```
./start.sh uprojects
```

- اجرای مولفه‌ی dm:

```
./start.sh dm up -d
```

- اجرای مولفه‌ی uplink:

```
./start.sh uplink up -d
```

- اجرای مولفه‌ی downlink:

```
./start.sh downlink up -d
```

- اجرای مولفه‌ی gm:

```
./start.sh gm up -d
```

تمامی دستورات up –d برای اجرا مولفه می‌باشد، در صورتی که بخواهید مولفه را باز اجرا نمایید کافی است آن را با restart جایگزین نمایید.

شایان ذکر است به هنگام راه‌اندازی دوباره پلتفرم لازم است داکرهای پروژه‌های کاربر به صورت دستی راه‌اندازی گردد.

۴-۳- لیست رابط‌های توسعه برنامه‌های کاربردی

لیست APIهایی که توسط هسته در اختیار سایر بخش‌ها قرار می‌گیرد در آدرس ذیل قرار داد.

https://documenter.getpostman.com/view/3119716/RWTspaW5

در پیوست ۲ این مستند، فایل JSON معادل request/response این APIها آمده است.

لیست APIهایی که توسط بخش LAN Server در اختیار هسته قرار می‌گیرد در آدرس ذیل قرار داد.

https://documenter.getpostman.com/view/3119716/RWTspaW6

در پیوست ۳ این مستند، فایل JSON معادل request/response این APIها آمده است.

.

۵- نصب و راه‌اندازی پلتفرم با استفاده از iso

یکی از راه‌های نصب که در بعضی از برنامه‌ها استفاده می‌گردد، نصب از طریق iso می‌باشد، در این روش سازندگان برنامه یک سیستم عامل خاص را هدف قرار داده و iso نصب آن را تغییر می‌دهند. در این تغییرات نیازمندی‌های برنامه را به iso اضافه می‌کنند و برخی از تنظیمات پیش‌فرض را نیز انجام می‌دهند. در نهایت کاربران می‌توانند با استفاده از iso سیستم عامل و برنامه را به صورت همزمان و بدون هیچ مشکلی نصب نمایند.

پکیج نصب ارائه شده، شامل نرم‌افزارهای لازم جهت کار با پلتفرم اینترنت اشیا می‌باشد. این ابزارها شامل داکر، کلاینت پایگاه‌داده‌ای و .. می‌باشد. پس از نصب نیاز است در اولین گام کارهای زیر را انجام دهید:

```
sudo usermod -aG docker $USER
docker network create isrc
/opt/init.sh
```

این دستور داکرهای پروژه را در سیستم نصب می‌کند. این داکرها private بوده و به صورت آنلاین قابل دسترس نمی‌باشند.

شایان ذکر است با توجه به نیازمندی‌های گسترده‌ی پلتفرم در هنگام استفاده از این image نیاز است که ارتباط با آن با اینترنت برقرار باشد.

برای راه‌اندازی کامل سیستم ابتدا به فولدر Server رفته و دستورات زیر را اجرا کنید:

```
./start.sh mongo up -d
./start.sh loraserver up -d
./start.sh pm up -d
./start.sh lanserver up -d
./start.sh uplink up -d
./start.sh downlink up -d
./start.sh dm up -d
./start.sh gm up -d
```

در ادامه به فولدر web رفته و دستور زیر را اجرا کنید:

```
docker-compose up -d
```

در ادامه در فولدر web به فولدر hosts/dev.front رفته و در فایل src/api/index.js تنظیمات زیر را مطابق با آدرس IP سیستم تغییر دهید.

```
const BASE_URL = 'http://185.116.162.237:7070/api/v1'
const BASE_FILES_URL = 'http://185.116.162.237:7070'
const BASE_ADMIN_URL = 'http://185.116.162.237:7070/api/admin'
```

بعد از انجام این تغییر دستورات زیر را اجرا کنید:

```
docker exec -ti web /bin/bash
cd home/nginx/dev.front
npm run build
exit
```

در نهایت پلتفرم در آدرس IP سیستم شما و پورت 7080 قابل دسترس می‌باشد. APIهای کاربردی پلتفرم در همان IP و پورت 7070 در دسترس قرار دارند. در صورت نیاز به **مشاهده** وضعیت اشیا LoRa در همان IP، پورت 8080 یک نسخه از پلتفرم loraserver.io قابل دسترس می‌باشد.

در این نصب دو پایگاه داده‌ای جداگانه در پورت‌های 27017 و 27018 ساخته می‌شوند، که به ترتیب برای core و backend می‌باشند.