到项目所在地址

D:\bysj 在地址栏中输入 cmd

跳出命令行模式

安装 运行项目

# install dependencies

npm install

# serve with hot reload at localhost:8080

npm run dev

# build for production with minification

npm run build

因为npm比较慢 所以去网上找了一个cnpm的镜像

1. 说明：因为npm安装插件是从国外服务器下载，受网络影响大，可能出现异常，如果npm的服务器在中国就好了，所以我们乐于分享的淘宝团队干了这事。来自官网：“这是一个完整 npmjs.org 镜像，你可以用此代替官方版本(只读)，同步频率目前为 10分钟 一次以保证尽量与官方服务同步。”
2. 官方网址：[http://npm.taobao.org](http://npm.taobao.org/)
3. 安装：命令提示符执行npm install cnpm -g --registry=[https://registry.npm.taobao.org](https://registry.npm.taobao.org/)
4. 注意：安装完后最好查看其版本号cnpm -v或关闭命令提示符重新打开，安装完直接使用有可能会出现错误

注：cnpm跟npm用法完全一致，只是在执行命令时将npm改为cnpm。

1. cnpm install
2. cnpm run dev
3. cnpm run build

（第三步可不做）

在sublime中vue代码高亮

<http://blog.csdn.net/huang_xiao_wei/article/details/70237496>

运行前端界面

到项目所在地址

D:\bysj 在地址栏中输入 cmd

在cmd命令行中输入 cnpm run dev

关于数据库

先设计数据库表 还有关系 联系 每个表中的主键 属性

注意：用powerdesigner画ER图的时候不需要在表里面写外键

先画ER图，再转物理模型，再生成sql语句。

在workbench中新建数据库，运行sql语句。

在饿了么官网上下载了一个框架，好像不全，连build、router都没有，看来不能偷懒。

[Vue warn]: Unknown custom element: <router-view> - did you register the component correctly? For recursive components, make sure to provide the "name" option.  
  
found in  
  
---> <App> at src\App.vue  
       <Root>

少了一个router包，研究了很久

在router里面的index.js 配置里有个默认首页 是“/”

然后router文件里的index.js里写了 “/”路径指的是有一个.vue文件

如果要写新的页面 在这个router里的 index.js加个路由就好

全局进入直接用这个就好了，前面那么多会被绕晕

公共组件 就是写一个比如叫做common的东西 然后在所有的页面都先import一下





Component template should contain exactly one root element.

template中根目录下只能包含一个,应该全部写入根目录(<div>)里

后端技术探索

因为前端是vue，都是JavaScript，用nodejs写后端，数据库用mongoDB。

KeystoneJS有express、vue的框架，有mongoose。

Keystonejs官网：

KeystoneJS使得在node.js中搭建数据库驱动的网站、应用程序和API变得容易了。

KeystoneJS在底层使用了[express.js](http://expressjs.com/) web服务器框架，并通过对象建模框架[mongoose](http://keystonejs.com/zh/docs/getting-started/mongoosejs.com)使用[MongoDB](http://mongodb.com/)数据库。

前端vue项目会运行，就是找到目录，输入cmd 然后cnpm run dev

后端keystone 找到目录，输入cmd，输入node keystone

如果在js文件里写了数据库，keystone就会自动关联该数据库。

如果没有写，会自动建立一个数据库。

表建在 keystone》models里面的js文件

表里的数据类型都用string，这样建表比较快^\_^

每次改了数据库 要重新运行node keystone 刷新 <http://127.0.0.1:3000/keystone>

网站账号密码 账号为默认邮箱[user@keystonejs.com](mailto:user@keystonejs.com) 密码为admin

刚刚删了建好的数据库表中的ID属性 添加数据的时候报错

MongoError: E11000 duplicate key error collection: bysj.cpus index: key\_1 dup key: { : null }

打开Mongoclient，点击右上角connect

选择数据库，点击右下角 connect now 按钮

接下来 清空数据库 左上角 drop database

建表的时候 可以不用ID，因为会自动生成哈希值当做ID，然后也有语句自动创建这条记录的生成时间。外键先写进来，类型是string，一会改成关系

keystone.js里面 表分类

keystone.set('nav', {

cpus: 'cpus',

users: 'users',

customers: 'customers',

});

Update.js>0.0.1-admins.js 可以设置登录邮箱以及密码

[31401395@stu.zucc.edu.cn](mailto:31401395@stu.zucc.edu.cn) asdfghjkl;’

接口的url：127.0.0.1:3000/api/login

传一个form

里面包含 username password

先在里面建立一个api

再在routes》index.js里面注册该路径

var routes = {

api: importRoutes('./api'),

};

在routes》index.js里面注册该文件

exports = module.exports = function (app) {

app.all('/api/test',routes.api.test)

}

然后就是写index.js代码了 包含数据库的读写等操作

前端界面改了一点，比如不要客户经理这一层，只有客户登录选购产品，然后再调登录的api，每个页面都要检查cookies，用户是否登录，如果没有登录要跳转到首页，如果登录了可以有登出的操作。

Keystone》module》login.js里面的

customer是别名

通过telephone和password两个条件定位出一个客户

通过查出来的customer可以获取他的id和name

id是默认的哈希值

登录后的每个页面都需要get username（具体步骤下面再写），还有退出登录

Product页面 点击加入购物车按钮 跳转到购物车 然后加载购物车（读取数据库里的内容），set一个默认值的订单

购物车页面，修改之后 要保存订单数据

查看资产 做后续的操作 按钮的位置设计

1、{{username}} 放在需要显示username的地方

2、return 里面 放 username:window.sessionStorage.getItem('username'),

3、搞定

突然发现了一个bug

如果登录状态下，点击首页，会跳转到最初没登录的界面，需要做个逻辑判断。

Vue里面有<div v-if> </div> <div v-else> </div> 如果有这个可以控制 就少一个界面。

原来的 登录之后点击确定按钮跳转到log\_in 改成登录之后刷新界面

今天把登录 退出 权限控制写好

登录之后有下拉框 可以退出，退出就是直接跳转到首页。

然后在methods之前加上了一段如下的代码

created(){

var userID = window.sessionStorage.getItem('userID');

if(!userID){

this.$router.push({path: '/'});

}

},

如果想在网址里输入shoppingcart 进入购物车 会强制跳转到首页 未登录状态

一期（过年前）想把购物车的读取什么的做好，目前就差这个界面的展开加载等操作了。

通过订单状态（1-13）判断订单是否在购物车、资产里，可以少一张表。

二期（过年后）写其他功能的界面和api。

购物车的展开之前已经做好了，接下来是各项的组件 计算价格 以及点击加入购物车按钮 跳转到购物车时，新增一条数据，明天做下。

今天上午逛街，下午修了几个购物车小细节，最下面距离还要改。api开了头，数据库改动了一下，mongoclient可能需要删一遍数据库。接下来一周可能因为过年回老家没网，没时间写代码，尝试论文开个头吧。

写了一个api 忘记在index.js里面加路径啦 导致404 not found

但是还是没有找到原因。。。

找到原因了 路径少了个斜杠/

删除只要调用删除的接口就可以删掉，然后购物车加载的时候这行会自动消失，但是取到id还要去饿了么查一下row什么的。修改是在提交订单的时候修改，价格还要再用公式算下。

今天查了一下删除的方法，数组放在一个v-for里，点击删除按钮传入index，然后再删除。提交按钮还要再做一下，修改提交一起，修改状态的数字，状态要数字和实际代表的内容映射。

最下面一行内容的显示被遮住了，要解决这个问题。√ 搞定

还有价格问题。

显示里的系统要不改成订购时间。

还有就是资产界面的问题。

退订是直接在这边的操作，变更和续订这边这是个中转站，也是一个提示框，有取消和确定，确定就直接到购物车里面做续订或者变更了

row.itemId可以获取订单编号，然后调用接口可以删除该订单

调试接口的时候 觉得数据库表设计的一项 name是多余的 要删除，

用mongoClient 删除表 再重启一下keystone 就可以了

测试删除接口时，改一下订单Id，就要重启keystone，然后再运行。

目前可以点击按钮 然后api里面的订单id是具体的哈希值 比如

5a8fcf2a0bd70ae794a65752 然后这个订单就删除了，刷新购物车 就没了。

删除按钮 返回id 。接下来要改成动态的接收你的话应该是 data:{id:xxxxx}。

axios.post('http://127.0.0.1:3000/api/deleteOrder', {

firstName: 'Fred',

lastName: 'Flintstone'

})

.then(function (response) {

console.log(response);

})

.catch(function (error) {

console.log(error);

});

data 相当于一个表单，动态data:{id:xxxxx}，把后面的xxx改成那个变量就行

Source tree 的github要登录，登录不成功，重置密码了还不行，写点东西提交一下。

删除 搞定 √ 退订修改了一下 有弹框， 产品界面优化了一下。

显示里的系统要不改成订购时间。 √ 搞定

还有就是资产界面的问题。

退订是直接在这边的操作，变更和续订这边这是个中转站，也是一个提示框，有取消和确定，确定就直接到购物车里面做续订或者变更了

提交按钮还要再做一下，修改提交一起，修改状态的数字，状态要数字和实际代表的内容映射。

还有价格问题。

button里的 type ，可以控制按钮类型，背景颜色等。

type="primary"

type="danger"

.demo-table-expand .el-form-item 中的width属性，控制展开行列数

如果是 100% 就只有一列，如果是50% 就是两列。

因为之前写的购物车界面可以复用，所以资产页面写的比较轻松

资产页面还缺少：单价，订单生失效时间。 续订、变更、退订的接口。

购物车界面还缺少：单价，提交接口。

其他的：修改状态，不同的查询页面根据状态的不同显示不同的查询结果（前端v-if解决）。

所以还需要写：1、价格公式；2、提交接口（修改）=续订（改时间）=变更（改配置）。

这里制定一下业务规则：续订 变更 会生成新的订单

比如我订购3个月 ，但是一个月的时候觉得不够用 我要升级配置 ，会下一个新的订单，那原来的订单可以删掉。就用这个新的。

如果我订购了三个月，没到期，续订了，原来的订单怎么办还要再考虑一下。

加了个价格公式，数字之间可以计算，addOrder里面初始化值的单位去掉，然后显示再加上，改了api，要重启node keystone，才能看效果-，-

还差个修改的api，（订购时间也影响价格。

还差个修改的api，订单状态变化查询结果不同，注册

昨天增加了截止时间，但是创建时间去掉了

刚刚试了输出修改配置后的数据，有些改了有些没改

修改update的接口还要再想想

值不变是因为没有加props.row

但是有的参数加了props.row 之后会报类型错误，明天看

现在参数改了可以保存下来了，然后就是更新了。

数据盘大小 这个不能更新的原因找到了

因为数据库里设置的表 变量名是dataHardDiskSize 但是前端输入的变量名是 inputDisk，统一改成了dataHardDiskSize 然后就好了，接下来再改剩下的带宽的问题，然后就可以保存了。

数据类型问题，把接口传回来的数据遍历一遍，改成number类型，然后bandwidth就可以修改，输出也是对了。