API Documentation

User

go top

update user

```
{
  url: 'udUser',
  method: 'post',
  param: {
    uid, // int, required
    name,
    username,

  },
  return_ok: {
    code: 201,
    msg: 'ok',
  },
  return_fail: {
    code: errCode.REGION_ERROR,
    msg: 'Duplicate regions!',
  }
}
```

Config

go top

add regions (with auth)

```
{
  url: 'region',
  method: 'post',
  param: {
    name, // string [5, 20] no duplicates
  },
  return_ok: {
    code: 201,
    msg: 'ok',
  },
  return_fail: {
    code: errCode.REGION_ERROR,
    msg: 'Duplicate regions!',
  }
}
```

get regions

```
{
  url: 'regions',
  method: 'get',
  return_ok: {
    code: 200,
    data: [
      {
        name, // string
        id, // int
      }
    ]
  },
  return_fail: {
    code: errCode.REGION_ERROR,
    msg: 'Duplicate regions!',
  }
}
```

get states

```
// if region_id is not null, search by region_id
// else return all states
{
  url: 'states',
  method: 'get',
  param: {
    region_id, // int allowNull
  }
  return_ok: {
    code: 200,
    data: [
      {
        name, //string
        id, // int
        region_id, // int
      }
    ]
  },
  return_fail: {
    code: errCode.STATE_ERROR,
    msg: 'can\'t get states',
  }
}
```

get stores

get business types

```
// if region_id is not null, search by region_id
// else return all states
{
  url: 'businessTypes',
  method: 'get',
  return_ok: {
    code: 200,
    data: [
      {
        name, // string
        type, // int
      }
    ]
  },
  return_fail: {
    code: errCode.STATE_ERROR,
    msg: 'can\'t get states',
  }
}
```

get categories

```
{
  url: 'categories',
  method: 'get',
  return_ok: {
    code: 200,
    data: [
      {
        id, // int
        name, // string
        code, // int
      }
    ]
  },
  return_fail: {
    code: err.code,
    msg: err.msg,
  }
}
```

get types

```
{
  url: 'types',
  method: 'get',
  param: {
    cate_code?, // int
  },
  return_ok: {
    code: 200,
    data: [
      {
        id, // int
        name, // string
        code, // int
        cate_code, // int
        Category: {
          name, // string
        }
      }
    ]
  },
  return_fail: {
    code: err.code,
    msg: err.msg,
  }
}
```

Staff

Add Staff

```
{
  url: '/staff'
  method: 'post',
  param: {
    name, // string [3, 15]
    job_title, // int 1 - salesperson 2 - store manager 3 - region manager, required, can only h
    store_assigned, // int
    region_assigned, // int at least one required, either store_assigned or region_assigned
    salary, // bigint 100 = 1 rmb
  },
  return_ok: {
    code: 201,
    msg: 'User Created!'
  },
  return_fail: {
    code: errCode.REGISTRATION_ERROR,
    msg: 'User ERROR'
  }
}
```

Get Staff

// dynamic criteria { id || region_assigned || store_assigned || job_title }
// results ordered by
// order: [
// ['job_title', 'DESC'],
// ['salary', 'DESC'],
// ['region_assigned'],
// ['store_assigned'],
//],

```
{
  url: '/staff'
  method: 'get',
  param: {
    id?, // int allowNull
    region_assigned?, // int allowNull
    store_assigned?, // int allowNull
    job_title?, // int allowNull
    salary?, // int allowNull find staff who has a salary less than some value
    page?, // int page number, default 1
    size?, // size, default 20
  },
  return_ok: {
    code: 200,
    data: {
      count, // int, number of results
      rows: [
        id, // int
        name, // string
        job_title, // int
        store_assigned, // int
        region_assigned, // int
        salary, // bigint
        Store: null || {
          name, // string
        },
        Region: null || {
          name, // string
        }
      }
      ]
    }
  },
  return_fail: {
    code: errCode.STAFF_ERROR,
    msg: 'User ERROR'
  }
}
```

Auth

go top

register

```
{
  url: '/register'
  method: 'post',
  param: {
    username, // string [3, 15]
    password, // string [6, 20]
    role_id, // int 1-home 2-business 99-admin
    ... // extra params required, see below
  },
  return_ok: {
    code: 201,
    msg: 'User Created!'
  },
  return_fail: {
    code: errCode.REGISTRATION_ERROR,
    msg: 'User ERROR'
  }
}
// if role_id is 1, extra params needed below
{
  marriage_status, // int 0-single 1-married 2-divorced
  gender, // int 0-male 1-female 2-NA
  birth, // int 10 digit timestamp
  annual_income, // bigint, 1 rmb = 100, last two digits are decimals
  street, // string [1, 30]
  city, // string [1, 20]
  state_id, // int provided by server end, see api get '/states'
  zip_code, // int length = 5
  uid, // int
}
// if role_id is 2, extra params needed below
{
  name, // string [3,15]
  annual_income, // bigint, 1 rmb = 100, last two digits are decimals
  street, // string [1, 30]
  city, // string [1, 20]
  state_id, // int provided by server end, see api get '/states'
  zip_code, // int length = 5
  cate, // string provided by server end, see api get 'businessCate'
  uid, // int
}
// if role_id is 99, extra params needed below
{
  name, // string [1, 10]
  uid, // int
}
```

Login

```
{
  url: '/login',
  method: 'post',
  param: {
    username, // string [3, 15]
    password, // string [6, 20]
  },
  return_ok: {
    code: 200,
    token, // string store login status
    data: {
      uid, // int user id
      name, // string
      role, // int
      auth: [1, 2], // int[]
      ... // extra attributes, see below
    }
  },
  return_fail: {
    code: errCode.LOGIN_ERROR,
    msg: 'login ERROR'
  },

  // if home user
  {
    marriage_status, // int 0-single 1-married 2-divorced
    gender, // int 0-male 1-female 2-NA
    age, // int [0, 200]
    annual_income, // bigint, 1 rmb = 100, last two digits are decimals
    street, // string [1, 30]
    city, // string [1, 20]
    state_id, // int provided by server end, see api get '/states'
    zip_code, // int length = 5
  }
  // if business user
  {
    name, // string [3,15]
    annual_income, // bigint, 1 rmb = 100, last two digits are decimals
    street, // string [1, 30]
    city, // string [1, 20]
    state_id, // int provided by server end, see api get '/states'
    zip_code, // int length = 5
    cate, // string provided by server end, see api get 'businessCate'
  }
}
```

Login By Token

```
{
  url: 'token',
  method: 'post',
  param: {

  },
  return_ok: {
    code: 200,
    data: {
      uid, // int user id
      name, // string
      role, // int
      auth: [1, 2], // int[]
      ... // extra attributes, see above, same as login
    }
  },
  return_fail: {
    code: errCode.Token_ERROR,
    msg: err.msg,
  },
}
```

Store

go top

Add Store

```
{
  url: 'store',
  method: 'post',
  param: {
    manager_id // int staff_id, allow null
    region_id, // int
    name, // string [3, 20] name cannot be duplicated within one region
    street, // string [1, 30]
    city, // string [1, 20]
    state_id, // string provided by server end, see api get '/states'
    zip_code, // int length = 5
  },
  return_ok: {
    code: 201,
    msg: 'success',
  },
  return_fail: {
    code: errCode.STORE_ERROR,
    msg: 'error when creating store',
  },
}
```

Get Store

// dynamic criteria, all optional

// order by: region_id, state_id, name

```
{
  url: 'store',
  method: 'get',
  param: {
    id, // int
    manager_id // int // staff_id, allow null
    region_id, // int
    state_id, // int provided by server end, see api get '/states'
    page?, // int
    size?, // int
  },
  return_ok: {
    code: 200,
    data: {
      count, // int, number of results
      rows: [
        {
          id, // int
          manager_id, // int
          region_id, //int
          name, // string [3, 20] name cannot be duplicated within one region
          street, // string [1, 30]
          city, // string [1, 20]
          state_id, // string provided by server end, see api get '/states'
          zip_code, // int length = 5
          sales, // bigint total amount of profits
          transactions, // int total amount of transactions
        }
      ],
    }
  },
  return_fail: {
    code: errCode.STORE_ERROR,
    msg: '',
  },
}
```

Delete Store

```
{
  url: 'deleteStore',
  method: 'get',
  param: {
    id, // int
  },
  return_ok: {
    code: 200,
    msg: 'deleted',
  },
  return_fail: {
    code: errCode.STORE_ERROR,
    msg: '',
  },
}
```

Set Store Manager

// if the store already has a manager, the original manager's job will be salesman and assigned to this store

// if the manager already has job

```
{
  url: 'storeManager',
  method: 'post',
  param: {
    manager_id, // int
    store_id, // int
  },
  return_ok: {
    code: 200,
    msg: 'success',
  },
  return_fail: {
    code: errCode.STORE_ERROR,
    msg: '',
  },
}
```

Product

Add Product

```
{
  url: 'addProduct',
  method: 'post',
  param: {
    name, // string
    cate, // int
    type, // int
    price, // bigint, eg: 10000 (meaning 100.00$)
    amount, // int number of inventory
    unit, // string, eg: '100g' '1 packet'
    // createTS, // product create date unix timestamp
    // listTS, // product list date unix timestamp, if null, it is not listed, null by default
    imgSrc, // string, product main img, allow null
    imgList, // array of imgSrc, length <= 3, eg: ['xadsxasdx', 'axds', 'xadxdas']
  },
  return_ok: {
    code: 201,
    msg: 'created',
  },
  return_fail: {
    code: errCode.Product_ERROR,
    msg: '',
  },
}
```

List Product

// a product can be available in many stores
// when a product is created, it is not listed
// a product needs to be listed to be available for customers

```
{
  url: 'list',
  method: 'post',
  param: {
    pid, // int product id, the product to be listed
    sid, // int[], store ids which are to list the product
  },
  return_ok: {
    code: 200,
    msg: 'Product Listed',
  },
  return_fail: {
    code: errCode.Product_ERROR,
    msg: '',
  },
}
```

Unlist Product

// product will not be available in all stores

```
{
  url: 'unlist',
  method: 'post',
  param: {
    pid, // int, product id
  },
  return_ok: {
    code: 200,
    msg: 'success',
  },
  return_fail: {
    code: errCode.Product_ERROR,
    msg: '',
  },
}
```

Get Product

// dynamic criteria searching

// order by cate, type, listed, available, price

// for a customer search, cate, type required and listed is true by default

```
{
  url: 'product',
  method: 'get',
  param: {
    cate, // int,
    type, // int,
    pid, // int, product id, allowNull
    sid, // int, store id, allowNull
    price, // bigint, find products whose price is lower than given price
    listed, // boolean, allowNull,
    available, // boolean, allowNull, whether the product has an inventory
  },
  return_ok: {
    code: 200,
    msg: 'success',
  },
  return_fail: {
    code: errCode.Product_ERROR,
    msg: '',
  },
}
```

update product

```
{
  url: 'udProduct',
  method: 'post',
  param: {
    id, // int, required, product id
    cate, // int,
    type, // int,
    price, // bigint, find products whose price is lower than given price
    listed, // boolean, allowNull, if null, no modification applies
    amount, // int, amount of product inventory
    description, // string
  },
  return_ok: {
    code: 200,
    msg: 'success',
  },
  return_fail: {
    code: errCode.PRODUCT_ERROR, // for example, the id does not exist
    msg: '',
  },
}
```