**Deskripsi Service**

**SOAP**

Return value pada service merupakan string berformat xml. Tag terluar (root) selalu <retval>. Misal:  
<retval>  
 <status>error</status>  
 <desc>username sudah ada</desc>  
 <other\_attribute>some value</other\_attribute>   
</retval>

1. createUser(string username, string password, string email, string nama\_lengkap, string alamat, string provinsi, string kota, string kodepos, string telepon)

menambahkan user baru

return string berformat xml (atas):

* <status> berisi `ok` jika sukses, `error` jika gagal
* <desc> berisi error message, hanya ada jika status = `error`
* <token> berisi string token login, akan disimpan di cookie / local storage. Hanya ada jika status ok

1. createBarang(string token, string nama\_barang, int harga, int stok, string kategori, string deskripsi, string imgdata)

menambahkan user baru. Jika imgdata == “”, akan digunakan gambar default

return string berformat xml (atas):

* <status> berisi `ok` jika sukses, `error` jika gagal
* <desc> berisi error message, hanya ada jika status = `error`
* <id> berisi id barang yang baru ditambahkan, hanya ada jika status = `ok`

**REST**

Barang

Model data barang dalam bentuk json:

* id : integer id barang
* nama : string nama barang
* harga : integer harga
* stok : integer stok barang
* kategori : string kategori barang
* deskripsi : string deskripsi barang
* jumlah\_beli : integer jumlah pembelian

1. GET barang/<id\_barang>

mendapatkan barang dengan id = <id\_barang>. error jika id tidak ada

return string berformat json:

* status : `ok` jika sukses, `error` jika gagal
* desc : error message, hanya ada jika status = `error`
* barang : json representasi barang, hanya ada jika status = `ok`

1. GET kategori/<nama\_kategori>

mendapatkan barang-barang dengan kategori = <nama\_kategori>. error jika kategori tidak ada

GET parameter (url encoded):

* (opsional) page=<no\_page> untuk nomor page. default=0
* (opsional) sort=name atau sort=harga. default=name
* (opsional) order=asc atau order=desc. default=asc

return string berformat json:

* status : `ok` jika sukses, `error` jika gagal
* desc : error message, hanya ada jika status = `error`
* hasil : array of json representasi barang, hanya ada jika status = `ok`

1. GET search/<entry\_search>

mencari barang dengan query = <entry\_search>

GET parameter (url encoded):

* (opsional) page=<no\_page> untuk nomor page.
* (opsional) sort=name atau sort=harga.
* (opsional) order=asc atau order=desc.

return string berformat json:

* status : `ok` jika sukses, `error` jika gagal
* desc : error message, hanya ada jika status = `error`
* hasil : array of json representasi barang, hanya ada jika status = `ok`

1. GET populer

mencari 10 barang dengan jumlah pembelian terbanyak

return string berformat json:

* status : `ok` jika sukses, `error` jika gagal
* desc : error message, hanya ada jika status = `error`
* hasil : array of json representasi barang, hanya ada jika status = `ok`

1. PUT barang/<id\_barang>

mengupdate data barang. Error jika token invalid / bukan token admin.

PUT data berformat json:

* barang : json representasi barang. Seluruh isi data, kecuali id, akan berubah
* token : string token login user.
* (opsional) imgdata : string base64 untuk data image barang

return string berformat json:

* status : `ok` jika sukses, `error` jika gagal
* desc : error message, hanya ada jika status = `error`

1. DELETE barang

menghapus data barang. Error jika token invalid / bukan token admin.

DELETE data berformat JSON:

* ids : array of integer, id barang yang akan dihapus.
* token : string token login user.

return string berformat json:

* status : `ok` jika sukses, `error` jika gagal
* desc : error message, hanya ada jika status = `error`

1. GET cart

Mendapatkan detail barang pada cart

GET data berformat JSON:

* ids : array of integer, id barang pada cart

return string berformat json:

* status : `ok` jika sukses, `error` jika gagal
* desc : error message, hanya ada jika status = `error`
* hasil : array of json representasi barang, hanya ada jika status = `ok`

1. POST cart/<id\_barang>

mengecek apakah barang cukup jika dimasukkan ke cart. Error jika barang tidak cukup atau token invalid.

POST data berformat JSON:

* jumlah: integer, jumlah barang yang ingin dimasukkan
* token : string token login user.

return string berformat json:

* status : `ok` jika sukses, `error` jika gagal
* desc : error message, hanya ada jika status = `error`

1. POST buy

membeli barang. Error jika barang tidak mencukupi atau token invalid.

POST data berformat JSON:

* cart : json object dengan format {id\_barang1 : jumlah1, id\_barang2 : jumlah2, dst...}
* token : string token login user.

return string berformat json:

* status : `ok` jika sukses, `error` jika gagal
* desc : error message, hanya ada jika status = `error`

User

Model data credit card dalam bentuk json:

* card\_name : string nama kartu kredit
* card\_number : string nomor kartu kredit
* card\_date : string tanggal kadaluarsa kartu kredit

Model data user dalam bentuk json:

* username : string username
* password : string password user
* email : string email user
* nama\_lengkap : string nama lengkap user
* alamat : string alamat user
* provinsi : string provinsi user
* kota : string kota user
* kodepos : string kode pos user
* telepon : string telepon user
* jumlah\_transaksi : integer jumlah transaksi user
* has\_card : boolean apakah user sudah mengisi info credit card atau tidak
* role : "admin" atau "user"

1. GET user/<username>

mendapatkan user dengan username = <username>. error jika user tidak ada atau token invalid.

GET parameter (url encoded):

* token = string token login user.

return string berformat json:

* status : `ok` jika sukses, `error` jika gagal
* desc : error message, hanya ada jika status = `error`
* user : json representasi user, kecuali password, hanya ada jika status = `ok`

1. GET user/<username>/card

mendapatkan apakah user sudah mendaftarkan card atau belum. error jika user tidak ada atau token invalid.

GET parameter (url encoded):

* token = string token login user.

return string berformat json:

* status : `ok` jika sukses, `error` jika gagal
* desc : error message, hanya ada jika status = `error`
* has\_card : true/false, hanya ada jika status = `ok`

1. POST login

mengecek login. error jika username atau password salah.

POST data berformat JSON:

* username : string username
* password : string password

return string berformat json:

* status : `ok` jika sukses, `error` jika gagal
* desc : error message, hanya ada jika status = `error`
* token: string token login, akan disimpan di cookie / local storage. Hanya ada jika status ok

1. POST logout

Logout.

POST data berformat JSON:

* username: string username
* token: string token login

return string berformat json:

* status : `ok` jika sukses, `error` jika gagal
* desc : error message, hanya ada jika status = `error`

1. PUT user/<username>

mengupdate data user dengan username = <username>. error jika username tidak ada atau token invalid.

PUT data berformat JSON:

* user: json representasi user.
* token: string token login user

return string berformat json:

* status : `ok` jika sukses, `error` jika gagal
* desc : error message, hanya ada jika status = `error`

1. PUT user/<username>/card

mengupdate informasi kartu kredit user. error jika username tidak ada atau token invalid.

PUT data berformat JSON:

* card: json representasi credit card.
* token: string token login user

return string berformat json:

* status : `ok` jika sukses, `error` jika gagal
* desc : error message, hanya ada jika status = `error`