

SFC WEB API

System Requirements

1. Design Principles

- 1.1. Design around **resources**
- 1.2. Focuses on **business entities**
- 1.3. Separate URIs for individual resource
- 1.4. **Stateless** request model
- 1.5. Use **HATEOAS** approach (navigation links, related resources)
- 1.6. URIs **should** be based on nouns not verbs
- 1.7. Design the API as an **abstraction of the data items (DB)**
- 1.8. A resource **does not have** to be based on a single data item (table)
- 1.9. Provide through the API the relationship between data items
- 1.10. Create a consistent naming convention
- 1.11. Follow the URIs pattern :
 - ./resources (get all items for this resource)
 - ./resources/item (get one single item)
- 1.12. Keep the URIs **as simple as possible** :
 - ./resources/item/nested_resoucers (recommended nested resources)
- 1.13. The more requests, the bigger the server load
- 1.14. Combine related information into a bigger resource (Extraneous Fetching Anti-pattern)

2. Anti-Design Principles

- 2.1. Do not create API mirroring the database
- 2.2. Do not create complex nested resources URIs :
 - ./resources/item/resources/item/resources/...
- 2.3. Do not create APIs that exposes a large number of small resources (**Chatty I/O Anti-pattern**)
- 2.4. Avoid client to send multiple requests to get all the desired data
- 2.5. Do not return to the client data that was not required
- 2.6. Do not return to the client very-large objects (**Extraneous Fetching Anti-pattern**)
- 2.7. Be careful with pseudo-resources and query-string parameters

3. Scenarios (use-cases)

3.1.Tracking Station

3.1.1. Login into station

- Validate login and password
- Validate the rights to login into the station
- Retrieve the last logged record (station_location)

3.1.2. Change the station

- Validate the rights to login into the station
- Validate login and password
- Update the login record (station_location)

3.1.3. Scan the unit

- Validate the unit route
- Validate the unit state (lock, fail, etc)
- Update the unit state
- Update the unit route
- Update the unit log
- Update the work order

3.2.Fail Station

3.2.1. Login into station

- Validate login and password
 - Validate the rights to login into the station
 - Retrieve the last logged record (station_location)
-

3.2.2. Change the station

- Validate the rights to login into the station
 - Validate login and password
 - Update the login record (station_location)
-

3.2.3. Scan the unit

- Validate the unit route
 - Validate the unit state (lock, fail, etc)
 - Update the unit state
 - Update the unit route
 - Update the unit log
 - Update the work order
-

3.2.4. Fail the unit

- Validate the unit route
- Validate the unit state (lock, fail, etc)
- Update the unit state (fail status)
- Update the unit route
- Update the unit log
- Update the work order

3.3.Assembly Station

3.3.1. Login into station

- Validate login and password
 - Validate the rights to login into the station
 - Retrieve the last logged record (station_location)
 - Retrieve the scan list (wo_parts)
-

3.3.2. Change the station

- Validate the rights to login into the station
 - Validate login and password
 - Update the login record (station_location)
 - Retrieve the scan list (wo_parts)
-

3.3.3. Scan the unit

- Validate the unit route
 - Validate the unit state (lock, fail, etc)
 - Update the unit state (fail status)
 - Update the unit route
 - Update the unit log
 - Update the work order
 - Retrieve the scan list (wo_parts)
-

3.3.4. Scan the parts

- Validate work order parts rules
- Assembly parts into unit
- Update parts status

4. API Resources

4.1. Users

4.2. Stations

4.3. WorkOrders

4.4. Wips

4.5. Parts

5. API Resources Actions

5.1. Users

- Login
- Logout
- Change station
- Check rights
- Change password

5.2. WorkOrders

- Get scan list (by product, process, etc)
- Validate parts (part number, eee-code, etc...)

5.3. Wips

- Scan serial number
- Check state (active, lock, failed, etc...)
- Assembly parts (collection)

5.4. Parts

- Scan serial number
- Assembly part (single part into a unit)
- Get assembled parts

6. API Reference

GET	/users(?offset&size)
Description	Get all users
Query String	(optional) [offset:int] Initial position to get records (Default : 1) (optional) [size:int] Number of records to get (Default : 100)
Request Body	none
Response Body	[{ id:int, login:string, photo:string }, { id:int, login:string, photo:string }, { id:int, login:string, photo:string },]
GET	/users/{userId}
Description	Get single user by ID
Query String	none
Request Body	none
Response Body	{ id:int, login:string, photo:string }
POST	/users
Description	Create new user
Query String	none
Request Body	{ login:string, password:string }
Response Body	{ id:int, login:string, photo:string }
POST	/users/login
Description	Log in to the API
Query String	none
Request Body	{ login:string, password:string, stationId:int }
Response Body	{ token:string }
POST	/users/{userId}/logout
Description	Log out from the API
Query String	[userId:int] User id
Request Body	none
Response Body	none

PUT	/users/{userId}
Description	Update user by ID
Query String	none
Request Body	{ login:string, password:string }
Response Body	{ id:int, login:string, photo:string }
DELETE	/users/{userId}
Description	Delete user by ID
Query String	none
Request Body	none
Response Body	none
GET	/stations(?offset&size)
Description	Get all stations
Query String	(optional) [offset:int] Initial position to get records (Default : 1)
	(optional) [size:int] Number of records to get (Default : 100)
Request Body	none
Response Body	[{ id:int, code:string, name:string, type:string }, { id:int, code:string, name:string, type:string }, { id:int, code:string, name:string, type:string },]
GET	/workorders(?offset&size)
Description	Get all work orders
Query String	(optional) [offset:int] Initial position to get records (Default : 1)
	(optional) [size:int] Number of records to get (Default : 100)
Request Body	{ product:string, mpn:string, status:string }
Response Body	[{ id:int, no:string, product:string, mpn:string, status:int }]
GET	/workorders/{workorderId}
Description	Get single work order by ID

Query String	none
Request Body	none
Response Body	<pre>{ id:int, no:string, product:string, mpn:string, status:int }</pre>
GET	/workorders/{workorderId}/wips
Description	Get all the wips from the work order
Query String	none
Request Body	none
Response Body	<pre>[{ id:int, no:string, product:string, mpn:string, status:int, workorderId:int, routeId:int }]</pre>
GET	/workorders/{workorderId}/wips/{wipId}
Description	Get a single wip from the work order
Query String	none
Request Body	none
Response Body	<pre>{ id:int, no:string, product:string, mpn:string, status:int, workorderId:int, routeId:int }</pre>
GET	/workorders/{workorderId}/parts(?station)
Description	Get all the parts (BOM) from the work order
Query String	(optional) [station:string] Station code to get parts from
Request Body	none

Response Body	<pre>[{ id:int, serial:string, partName:string, partNumber:string, eeeCode:string }]</pre>
GET	/workorders/{workorderId}/parts/{partId}
Description	Get a single the part from the work order
Query String	none
Request Body	none
Response Body	<pre>{ id:int, serial:string, partName:string, partNumber:string, eeeCode:string }</pre>
GET	/workorders/{workorderId}/parts/{partName}
Description	Get all the parts for the given part name
Query String	none
Request Body	none
Response Body	<pre>[{ id:int, serial:string, partName:string, partNumber:string, eeeCode:string }]</pre>
POST	/workorders
Description	Create new work order
Query String	none
Request Body	<pre>{ no:string, product:string, mpn:string, target:int }</pre>

Response Body	<pre>{ id:int, no:string, product:string, mpn:string, status:int }</pre>
PUT	/workorders/{workorderId}
Description	Update work order by ID
Query String	none
Request Body	<pre>{ input:int, output:int, status:string }</pre>
Response Body	<pre>{ id:int, no:string, product:string, mpn:string, status:int }</pre>
DELETE	/workorders/{workorderId}
Description	Delete work order by ID
Query String	none
Request Body	none
Response Body	none
GET	/wips(?offset&size)
Description	Get all the wips
Query String	(optional) [offset:int] Initial position to get records (Default : 1)
	(optional) [size:int] Number of records to get (Default : 100)
Request Body	<pre>{ product:string, mpn:string, currentStation:string, nextStation:string }</pre>

Response Body	<pre>[{ id:int, no:string, product:string, mpn:string, status:int, workorderId:int, routeId:int }]</pre>
GET	/wips/{wipId}
Description	Get a single wip by Id
Query String	none
Request Body	none
Response Body	<pre>{ id:int, no:string, product:string, mpn:string, status:int, workorderId:int, routeId:int }</pre>
GET	/wips/{wipId}/parts
Description	Get all assembled parts with the wip
Query String	none
Request Body	none
Response Body	<pre>[{ id:int, serial:string, partName:string, partNumber:string, eeeCode:string }]</pre>
GET	/wips/{wipId}/parts/{partId}
Description	Get a single assembled part with the wip
Query String	none
Request Body	none

Response Body	<pre>{ id:int, serial:string, partName:string, partNumber:string, eeeCode:string }</pre>
POST	/wips/{wipId}/check?station
Description	Only checks the wip status and whether it is ok to pass the station
Query String	[station:string] Station code to check the wip
Request Body	none
Response Body	none
POST	/wips/{wipId}/scan?station
Description	Scan the wip to pass or fail into the station
Query String	[station:string] Station code to pass/fail the wip
Request Body	{ testItem:string, symptom:string}
Response Body	none
POST	/wips/{serialNo}/check?station
Description	Only checks the wip status and whether it is ok to pass the station
Query String	[station:string] Station code to check the wip
Request Body	none
Response Body	none
POST	/wips/{serialNo}/scan?station
Description	Scan the wip to pass or fail into the station
Query String	[station:string] Station code to pass/fail the wip
Request Body	{ testItem:string, symptom:string}
Response Body	none

RESOURCE	GET	POST	PUT	DELETE
/users	Get all users	Create a new user	Bulk update users	Delete all users
/users?filter	Get users with filter	n/a	n/a	Delete users with filter
/users/{id}	Get the user detail	n/a	Update the user detail	Delete the user

RESOURCE	GET	POST	PUT	DELETE
/users/{id}? action=login	n/a	Login the user	n/a	n/a
/users/{id}? action=logout	n/a	Logout the user	n/a	n/a
/users/{id}? action=rights	Get user rights	Check user rights	n/a	n/a
/stations	Get all stations	Create new station	Bulk update stations	Delete all stations
/stations?filter	Get stations with filter	n/a	n/a	Delete stations with the filter
/stations/{id}	Get the station detail	n/a	Update the station detail	Delete the station
/workorders	Get all WOs	Create new WO	Bulk update WOs	Delete all WOs
/workorders? filter	Get WOs with filter	n/a	n/a	Delete WOs with filter
/workorders/{id}	Get WO detail	n/a	Update WO detail	Delete WO
/workorders/{id}/ wips	Get all WIPs for the WO	n/a	Bulk update WIPs for the WO	Delete all WIPs for the WO
/workorders/{id}/ wips?filter	Get WIPs for the WO with filter	n/a	n/a	Delete WIPs for the WO with filter
/workorders/{id}/ parts	Get all PARTs for the WO	Add part for the WO	Bulk update PARTs for the WO	Delete all PARTs for the WO
/workorders/{id}/ parts?filter	Get PARTs for the WO with filter	n/a	n/a	Delete PARTs for the WO with filter
/workorders/{id}/ routes	Get ROUTE for the WO	n/a	Update ROUTE for the WO	Remove ROUTE for the WO
/workorders/{id}/ routes?filter	Get ROUTE for the WO with filter	n/a	Update ROUTE for the WO with filter	Remove ROUTE for the WO with filter
/wips	Get all WIPs	n/a	Bulk update WIPs	Delete all WIPs
/wips?filter	Get WIPs with filter	n/a	Update WIPs with filter	Delete WIPs with filter
/wips/{id}	Get WIP detail	n/a	Update WIP detail	Delete WIP
/wips/{id}? action=check	n/a	Check WIP status	n/a	n/a
/wips/{id}? action=scan	n/a	Scan WIP for PASS or FAIL	n/a	n/a
/wips/{id}? action=history	Get WIP history (wip_log)	n/a	n/a	n/a

RESOURCE	GET	POST	PUT	DELETE
/wips/{id}/workorders	Get WO for the WIP	n/a	Update WO for the WIP	n/a
/wips/{id}/parts	Get all PARTs for the WIP	Add PARTs for the WIP	Update PARTs for the WIP	Delete PARTs for the WIP
/wips/{id}/parts?filter	Get PARTs for the WIP using filter			Delete PARTs for the WIP using filter
/wips/{id}/routes	Get ROUTE for the WIP	n/a	Update ROUTE for the WIP	n/a
/wips/{id}/routes?filter	Get ROUTE for the WIP with filter	n/a	Update ROUTE for the WIP with filter	n/a
/parts	Get all PARTs	Create new PART	Bulk update PARTs	Delete all PARTs
/parts?filter	Get PARTs with filter	n/a	Update PARTs with filter	Delete PARTs with filter
/parts/{id}	Get PART detail	n/a	Update PART detail	Delete PART
/parts/{id}?action=scan	n/a	Scan the PART	n/a	n/a
/parts/{id}/wips	Get WIP for the PART	Add WIP for the PART	Update WIP for the PART	Delete WIP for the PART
/parts/{id}/wips?filter	Get WIP for the PART with filter	Add WIP for the PART with filter	Update WIP for the PART with filter	Delete WIP for the PART with filter

7. API Sequence Diagram

8. Business Logic Sequence Diagram

Thursday, 10 January 2019