

Test report

Smoke-it web shop

Maarten Hormes

S3-CB03

2021

Contents

Introduction.....	3
Unit & Integration test	4
System test	6
UATs.....	16
E2E testing.....	17
Version history	19

Introduction

This document will be used to document the results of the tests for the smoke-it application. There will be a section for the unit and integration test. The results are tracked via SonarQube and will be stated in the form of code coverage.

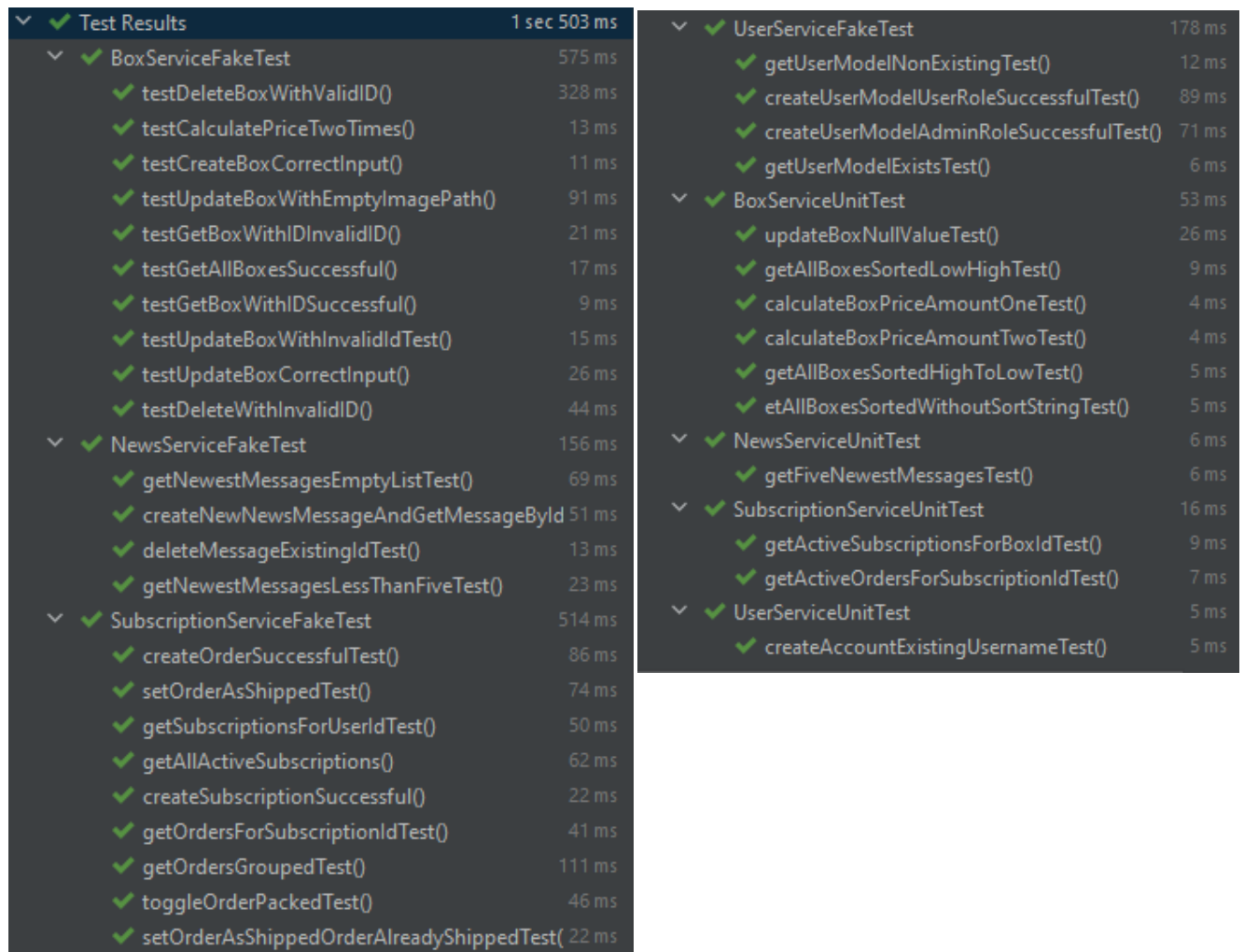
The back-end has another form of testing. These are the system tests performed via Postman. Like mentioned in the test strategy in the test plan file, these tests have to be performed manually. The option to automate them is there, and will hopefully be used in the future. For now, these tests are nothing more than API calls to my back-end, that can be seen as successful by checking the return body of the request. Screenshots of these will be provided.

For testing the whole system there are 2 ways of testing currently set up. The first being the User Acceptance Testing (UAT). These tests have to be performed by a user of the application. Currently, there are one or more test cases for each of the user stories. The tests have been set up in a way to cover all of the user story's acceptance criteria.

The last form of testing documented in this file will be the End to End (E2E) test, performed with the help of Cypress. These E2E tests will test all of the user stories currently present in the application. At this moment these tests have not all been setup. The results of the present tests will be shown in the form of a screenshot, showing the successful result.

Unit & Integration test

The way of testing should be clear by reading the test strategy in the test plan document. Since listing all tests here does not feel like very useful, I included a screenshot with the title of each unit and integration test present in the application. The screenshot should give a good indication of how the tests are set up. The fake tests are the integration test, and are called like that since they use a fake version of the database.



✓ Test Results	1 sec 503 ms
✓ BoxServiceFakeTest	575 ms
✓ testDeleteBoxWithValidID()	328 ms
✓ testCalculatePriceTwoTimes()	13 ms
✓ testCreateBoxCorrectInput()	11 ms
✓ testUpdateBoxWithEmptyImagePath()	91 ms
✓ testGetBoxWithInvalidID()	21 ms
✓ testGetAllBoxesSuccessful()	17 ms
✓ testGetBoxWithIDSuccessful()	9 ms
✓ testUpdateBoxWithInvalidIDTest()	15 ms
✓ testUpdateBoxCorrectInput()	26 ms
✓ testDeleteWithInvalidID()	44 ms
✓ NewsServiceFakeTest	156 ms
✓ getNewestMessagesEmptyListTest()	69 ms
✓ createNewNewsMessageAndGetMessageById()	51 ms
✓ deleteMessageExistingIdTest()	13 ms
✓ getNewestMessagesLessThanFiveTest()	23 ms
✓ SubscriptionServiceFakeTest	514 ms
✓ createOrderSuccessfulTest()	86 ms
✓ setOrderAsShippedTest()	74 ms
✓ getSubscriptionsForUserIdTest()	50 ms
✓ getAllActiveSubscriptions()	62 ms
✓ createSubscriptionSuccessful()	22 ms
✓ getOrdersForSubscriptionIdTest()	41 ms
✓ getOrdersGroupedTest()	111 ms
✓ toggleOrderPackedTest()	46 ms
✓ setOrderAsShippedOrderAlreadyShippedTest()	22 ms
✓ UserServiceFakeTest	178 ms
✓ getUserModelNonExistingTest()	12 ms
✓ createUserModelUserRoleSuccessfulTest()	89 ms
✓ createUserModelAdminRoleSuccessfulTest()	71 ms
✓ getUserModelExistsTest()	6 ms
✓ BoxServiceUnitTest	53 ms
✓ updateBoxNullValueTest()	26 ms
✓ getAllBoxesSortedLowHighTest()	9 ms
✓ calculateBoxPriceAmountOneTest()	4 ms
✓ calculateBoxPriceAmountTwoTest()	4 ms
✓ getAllBoxesSortedHighToLowTest()	5 ms
✓ etAllBoxesSortedWithoutSortStringTest()	5 ms
✓ NewsServiceUnitTest	6 ms
✓ getFiveNewestMessagesTest()	6 ms
✓ SubscriptionServiceUnitTest	16 ms
✓ getActiveSubscriptionsForBoxIdTest()	9 ms
✓ getActiveOrdersForSubscriptionIdTest()	7 ms
✓ UserServiceUnitTest	5 ms
✓ createAccountExistingUsernameTest()	5 ms

Below you can find a screenshot of the test coverage calculated by SonarQube. This test coverage is not including every file in the project. Some files, like the spring security, will not be tested by the unit and integration tests, since the security is needed when accessing the application from outside of the app. Next to that, controllers and model converters have been excluded, since they will be tested by both the system test and the E2E test.

The picture shows the total coverage over all listed files. Files not listed here are excluded like explained above.

school

View as

List



to select files



to navigate

15 files

Coverage 87.8%

New Code: Since October 21, 2021

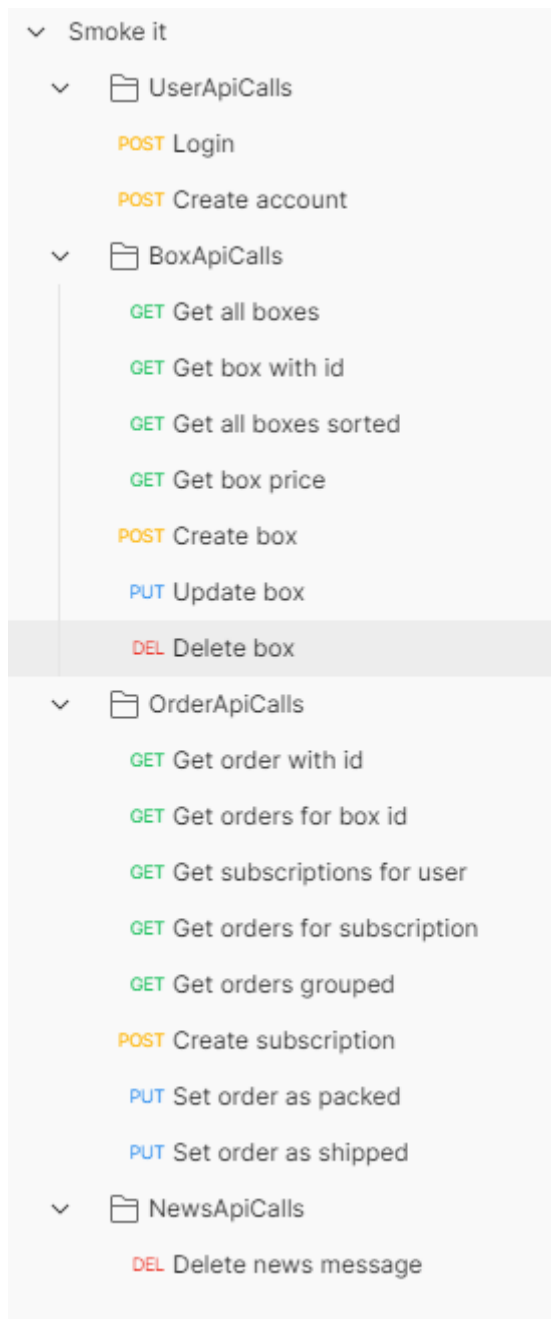
	Coverage	Uncovered Lines	Uncovered Conditions
src/main/java/fontys/sem3/smoke_it/SmokeltApplication.java	33.3%	2	–
src/main/java/fontys/sem3/smoke_it/model/GroupedOrders.java	69.2%	4	–
src/main/java/fontys/sem3/smoke_it/model/NewsMessageModel.java	69.6%	4	3
src/main/java/fontys/sem3/smoke_it/repository/DataSourceNews.java	71.4%	2	–
src/main/java/fontys/sem3/smoke_it/model/SubscriptionModel.java	74.4%	8	2
src/main/java/fontys/sem3/smoke_it/model/UserModel.java	84.0%	1	3
src/main/java/fontys/sem3/smoke_it/service/SubscriptionService.java	88.5%	3	7
src/main/java/fontys/sem3/smoke_it/model/BoxModel.java	89.3%	0	3
src/main/java/fontys/sem3/smoke_it/service/NewsService.java	93.1%	1	1
src/main/java/fontys/sem3/smoke_it/service/BoxService.java	97.0%	1	0
src/main/java/fontys/sem3/smoke_it/repository/DataSourceBoxes.java	100%	0	0
src/main/java/fontys/sem3/smoke_it/repository/DataSourceSubscriptions.java	100%	0	0
src/main/java/fontys/sem3/smoke_it/repository/DataSourceUser.java	100%	0	–
src/main/java/fontys/sem3/smoke_it/model/OrderModel.java	100%	0	–
src/main/java/fontys/sem3/smoke_it/service/UserService.java	100%	0	0

15 of 15 shown

System test

Like mentioned in the introduction, the system tests are performed in postman, manually. Below you can find a screenshot for each of these 'tests'. There is no test setup to check the returned body, so if the body is unclear you can find a short description below the picture to elaborate why the shown body can be seen as successful.

First an overview of all end points:



Create a new user:

The response body shows the newly created user object, with encrypted password and auto generated ID.

The screenshot shows a REST client interface with the following details:

- URL:** `http://localhost:8080/user/register`
- Method:** `POST`
- Body (JSON):**

```
{  "username": "admin",  "password": "admin",  "role": "ADMIN"}
```
- Status:** `200 OK`, `321 ms`, `541 B`
- Response Body (JSON):**

```
{  "id": 1,  "username": "admin",  "password": "$2a$10$24rTNw9gPizqmWBHFAtY0uz67HOMyw1PHi5CEP/AwNZbBNhz56kbW",  "email": null,  "role": "ADMIN"}
```

Login:

The JWT token returned in the body shows the user is logged in successfully.

The screenshot shows a REST client interface with the following details:

- URL:** `http://localhost:8080/login`
- Method:** `POST`
- Body (JSON):**

```
{  "username": "admin",  "password": "admin"}
```
- Status:** `200 OK`, `262 ms`, `648 B`
- Response Body (JSON):**

```
{  "Authorization": "Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJhZG1pb251bnV4bGUyOiJBRRE1JT0iImV4cCI6MTY0MDUxNjU5NCwidXN1cklkIjoxfjQ.pIWz_yKZDouLmm-vw9-GEMhop9UzcoWv391iND0KoEWwk0wKzSEGN0k64ECTr9-fPz6wbdrFacNTtgoTPz7-qA"}
```

Create box:

The body shows the created box object.

The screenshot shows a REST client interface with a POST request to `http://localhost:8080/boxes/create`. The request body is in `form-data` format. The response is a 200 OK status with a response time of 151 ms and a body size of 661 B. The response body is displayed in JSON format, showing the created box object.

Key	Value	Description
<input checked="" type="checkbox"/> basePrice	29.99	
<input checked="" type="checkbox"/> content	Something, else, test	
<input checked="" type="checkbox"/> description	Some text to test	
<input checked="" type="checkbox"/> imagePath		
<input checked="" type="checkbox"/> imageFile	TICT Corona Contact Tracing... X	

```
1 {
2   "id": "a16d8da9-c500-4bd9-be9c-cc492824c1be",
3   "name": "testBox",
4   "basePrice": 29.99,
5   "content": "Something, else, test",
6   "description": "Some text to test",
7   "imagePath": "file:///C:/Users/Gebruiker/Desktop/smoke-it_its/BE/images/testBox.pdf",
8   "imageFile": null
9 }
```

Update box:

Updated the box created above. Body responds with a 200 OK code.

The screenshot shows a REST client interface with a PUT request to `http://localhost:8080/boxes/update`. The request body is in `form-data` format. The response is a 200 OK status with a response time of 87 ms and a body size of 371 B. The response body is displayed in Text format, showing the number 1.

Key	Value	Description
<input checked="" type="checkbox"/> content	Something, else, test	
<input checked="" type="checkbox"/> description	Some text to test	
<input checked="" type="checkbox"/> imagePath		
<input checked="" type="checkbox"/> imageFile	Select Files	
<input checked="" type="checkbox"/> id	a16d8da9-c500-4bd9-be9c-cc4...	

```
1
```


Get all boxes:

The body shows a list of all boxes. Only 1 box I created, so list only shows 1.

The screenshot shows a REST client interface. The method is GET and the URL is `http://localhost:8080/boxes/`. The response status is 200 OK, with a response time of 26 ms and a body size of 660 B. The response body is displayed in JSON format, showing a single object in an array:

```
1 [
2   {
3     "id": "a16d8da9-c500-4bd9-be9c-cc492824c1be",
4     "name": "testBox",
5     "basePrice": 29.99,
6     "content": "Something, else, test",
7     "description": "Some text to test",
8     "imagePath": "file:///C:/Users/Gebruiker/Desktop/smoke-it_its/BE/images/
9       testBox.",
10    "imageFile": null
11  }
12 ]
```

Get box with id:

The body shows the box we created at the beginning. It seems like the get all boxes, but shows 1 object instead of 1 object in a list.

The screenshot shows a REST client interface. The method is GET and the URL is `http://localhost:8080/boxes/a16d8da9-c500-4bd9-be9c-cc492824c1be`. The response status is 200 OK, with a response time of 33 ms and a body size of 658 B. The response body is displayed in JSON format, showing a single object:

```
1 {
2   "id": "a16d8da9-c500-4bd9-be9c-cc492824c1be",
3   "name": "testBox",
4   "basePrice": 29.99,
5   "content": "Something, else, test",
6   "description": "Some text to test",
7   "imagePath": "file:///C:/Users/Gebruiker/Desktop/smoke-it_its/BE/images/testBox.",
8   "imageFile": null
9 }
```

Get box price:

The endpoint shows the price for the amount of boxes selected. In this example the amount is 4.

GET ⌵ <http://localhost:8080/boxes/a16d8da9-c500-4bd9-be9c-cc492824c1be/pr...> Send ⌵

Params ● Auth Headers (7) Body Pre-req. Tests Settings Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	amount	4			
	Key	Value	Description		

Body ⌵ 200 OK 30 ms 430 B Save Response ⌵

Pretty Raw Preview Visualize JSON ⌵ ≡

```
1 27.590799999999998
```

Delete box with id:

The body shows a 200 OK code, combined with a true indicating the box was deleted successful.

DELETE ⌵ <http://localhost:8080/boxes/delete/a16d8da9-c500-4bd9-be9c-cc492824c1be/pr...> Send ⌵

Params Auth Headers (8) Body Pre-req. Tests Settings Cookies

Headers 👁 7 hidden

	KEY	VALUE	DES	...	Bulk Edit	Presets	⌵
<input checked="" type="checkbox"/>	Authorization	Bearer eyJ0eXAiOiJKV1QiLCJhbG...					
	Key	Value	Description				

Body ⌵ 200 OK 47 ms 416 B Save Response ⌵

Pretty Raw Preview Visualize JSON ⌵ ≡

```
1 true
```

Create subscription:

The body shows the created subscriptions. The amount left is already 1 less than the amount bought. The first order for the subscription is already created.

POST

http://localhost:8080/subscriptions/create/

Send

ParamsAuthHeaders (9)BodyPre-req. Tests SettingsCookies

rawJSONBeautify

```
1  {
2    "boxId": "7233d9ba-a678-4668-a49a-91a7f8882b33",
3    "userId": 0,
4    "amountBought": 2,
5    "frequency": 2,
6    "email": "test@mail.com",
7    "name": "Maarten",
8    "address": "some street 4",
9    "postal": "6045EA",
10   "city": "Roermond"
11 }
```

Body200 OK91 ms655 BSave Response

PrettyRawPreviewVisualizeJSON

```
1  {
2    "subscriptionId": 1,
3    "boxId": "7233d9ba-a678-4668-a49a-91a7f8882b33",
4    "userId": 0,
5    "amountBought": 2,
6    "amountLeft": 1,
7    "frequency": 2,
8    "totalCost": 0.0,
9    "email": "test@mail.com",
10   "name": "Maarten",
11   "address": "some street 4",
12   "postal": "6045EA",
13   "city": "Roermond"
14 }
```

Get subscriptions for user id:

Only 1 subscription created for user 0 shows a list with 1 subscription object

GET localhost:8080/subscriptions/0

Params Auth Headers (8) Body Pre-req. Tests Settings Cookies

Headers 7 hidden

KEY	VALUE	DES	ooo	Bulk Edit	Presets
Authorization	Bearer eyJ0eXAiOiJKV1QiLCJhbG...				
Key	Value				Description

Body 200 OK 32 ms 645 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 1,
3   "boxId": "7233d9ba-a678-4668-a49a-91a7f8882b33",
4   "userId": 0,
5   "amountBought": 2,
6   "amountLeft": 1,
7   "frequency": 2,
8   "totalCost": 0.0,
9   "name": "Maarten",
10  "email": "test@mail.com",
11  "address": "some street 4",
12  "postal": "6045EA",
13  "city": "Roermond"
14 }
```

Get orders for subscription:

Above subscription has id 1. Here we get the orders from this subscription

GET http://localhost:8080/subscriptions/ordersFor/1

Params Auth Headers (8) Body Pre-req. Tests Settings Cookies

Headers 7 hidden

KEY	VALUE	DES	ooo	Bulk Edit	Presets
Authorization	Bearer eyJ0eXAiOiJKV1QiLCJhbG...				
Key	Value				Description

Body 200 OK 42 ms 647 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "orderId": 1,
3   "subscriptionId": null,
4   "boxId": null,
5   "userId": null,
6   "amountBought": 0,
7   "amountLeft": 0,
8   "frequency": 0,
9   "email": null,
10  "name": null,
11  "address": null,
12  "postal": null,
13  "city": null,
14  "packed": false,
15  "shipped": false,
16  "deliveryDate": "2022-01-01"
17 }
```

Get order with id:

We get the same order, but by id instead of subscription id

GET http://localhost:8080/subscriptions/orders/1 Send

Params Auth **Headers (8)** Body Pre-req. Tests Settings Cookies

Headers 7 hidden

	KEY	VALUE	DES	Bulk Edit	Presets
<input checked="" type="checkbox"/>	Authorization	Bearer eyJ0eXAiOiJKV1QiLCJhbG...			
	Key	Value			Description

Body 200 OK 28 ms 710 B Save Response

Pretty Raw Preview Visualize **JSON**

```
1 {
2   "orderId": 1,
3   "subscriptionId": 1,
4   "boxId": "7233d9ba-a678-4668-a49a-91a7f8882b33",
5   "userId": 0,
6   "amountBought": 2,
7   "amountLeft": 1,
8   "frequency": 2,
9   "email": "test@mail.com",
10  "name": "Maarten",
11  "address": "some street 4",
12  "postal": "6045EA",
13  "city": "Roermond",
14  "packed": false,
15  "shipped": false,
16  "deliveryDate": "2022-01-01"
17 }
```

Orders for box id:

We once again get the same order, but this time by the box id that is ordered

GET http://localhost:8080/subscriptions/grouped/7233d9ba-a678-4668-a49a-91a7f8882b33 Send

Params Auth **Headers (8)** Body Pre-req. Tests Settings Cookies

Headers 7 hidden

	KEY	VALUE	DES	Bulk Edit	Presets
<input checked="" type="checkbox"/>	Authorization	Bearer eyJ0eXAiOiJKV1QiLCJhbG...			
	Key	Value			Description

Body 200 OK 45 ms 712 B Save Response

Pretty Raw Preview Visualize **JSON**

```
1 {
2   "orderId": 1,
3   "subscriptionId": 1,
4   "boxId": "7233d9ba-a678-4668-a49a-91a7f8882b33",
5   "userId": 0,
6   "amountBought": 2,
7   "amountLeft": 1,
8   "frequency": 2,
9   "email": "test@mail.com",
10  "name": "Maarten",
11  "address": "some street 4",
12  "postal": "6045EA",
13  "city": "Roermond",
14  "packed": false,
15  "shipped": false,
16  "deliveryDate": "2022-01-01"
17 }
```

Get orders grouped:

This endpoint is used in the admin portal, to show how many orders there are for each of the boxes. This also includes the flags indicating the packing/shipping urgency. This example is run on my production database to include more information

GET

http://localhost:8080/subscriptions/grouped

Send

ParamsAuthHeaders (8)BodyPre-req. TestsSettingsCool

Headers7 hidden

	KEY	VALUE	DE	...	Bulk Edit	Presets
<input checked="" type="checkbox"/>	Authorization	Bearer eyJ0eXAiOiJKV1QiLCJhb...				

Body200 OK48 ms847 BSave Respons

PrettyRawPreviewVisualizeJSON

```
1  [
2    {
3      "boxName": "Daddy's box",
4      "boxID": "68d6738d-8984-4685-88a7-884dd5d340e4",
5      "amount": 1,
6      "packFlag": 1,
7      "shipFlag": 0
8    },
9    {
10     "boxName": "Roll-Kit",
11     "boxID": "7233d9ba-a678-4668-a49a-91a7f8882b33",
12     "amount": 2,
13     "packFlag": 0,
14     "shipFlag": 0
15   },
16   {
17     "boxName": "Smoke-Kit",
18     "boxID": "368f2c6a-70fd-4021-992f-4ef521c2922d",
19     "amount": 3,
20     "packFlag": 2,
21     "shipFlag": 1
22   },
23   {
24     "boxName": "Supply-Kit",
25     "boxID": "a3c3714c-03d0-4f41-831e-b2a4c88008d7",
26     "amount": 4,
27     "packFlag": 1,
28     "shipFlag": 0
29   }
30 ]
```

Set order as packed:

Here we see shipping status being false, but packed as true

PUT localhost:8080/subscriptions/orders/pack/52 Send

Params Auth Headers (9) Body Pre-req. Tests Settings

Headers 8 hidden

	KEY	VALUE	DES	ooo	Bulk Edit	Pres
<input checked="" type="checkbox"/>	Authorization	Bearer eyJ0eXAiOiJKV1QiLCJhbG...				
	Key	Value				Description

Body 200 OK 21 ms 722 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "orderId": 52,
3   "subscriptionId": 40,
4   "boxId": "68d6738d-8984-4685-88a7-884dd5d340e4",
5   "userId": 0,
6   "amountBought": 2,
7   "amountLeft": 1,
8   "frequency": 1,
9   "email": "islay@whisky.com",
10  "name": "Me, myself and I",
11  "address": "Port Ellen 4",
12  "postal": "1234AA",
13  "city": "Scotland",
14  "packed": true,
15  "shipped": false,
16  "deliveryDate": "2021-12-01"
17 }
```

Set order as shipped:

Once the order is packed we can ship it. Both packed and shipped are true

PUT http://localhost:8080/subscriptions/orders/send/52 Send

Params Auth Headers (9) Body Pre-req. Tests Settings

Headers 8 hidden

	KEY	VALUE	DES	ooo	Bulk Edit	Pre
<input checked="" type="checkbox"/>	Authorization	Bearer eyJ0eXAiOiJKV1QiLCJhbG...				
	Key	Value				Description

Body 200 OK 43 ms 721 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "orderId": 52,
3   "subscriptionId": 40,
4   "boxId": "68d6738d-8984-4685-88a7-884dd5d340e4",
5   "userId": 0,
6   "amountBought": 2,
7   "amountLeft": 0,
8   "frequency": 1,
9   "email": "islay@whisky.com",
10  "name": "Me, myself and I",
11  "address": "Port Ellen 4",
12  "postal": "1234AA",
13  "city": "Scotland",
14  "packed": true,
15  "shipped": true,
16  "deliveryDate": "2021-12-01"
17 }
```

UATs

Each of the user acceptance tests can be found in the test plan document. To make this file more complete, I added the results of the tests in this document. Here you can find the ID of the test, the name and the result.

successful - TC-AS-01 - Login to the application
successful - TC-AS-02 - Login with incorrect credentials
successful - TC-AS-03 - Login with no connection to the database
successful - TC-AS-04 - See all existing boxes
successful - TC-AS-05 - Add new subscription box
successful - TC-AS-06 - Add new subscription box without all necessary information
successful - TC-AS-07 - Edit existing box
successful - TC-AS-08 - Delete existing box
successful - TC-AS-09 - Edit information on about us page - part of app non existing
successful - TC-AS-10 - See information on about us page
successful - TC-AS-11 - Edit information on contact page - part of app non existing
successful - TC-AS-12 - See information on contact page
successful - TC-AS-13 - See the newsfeed
successful - TC-AS-14 - Add new message to newsfeed
successful - TC-AS-15 - Add new message with invalid content/title
successful - TC-AS-16 - Delete an existing message

successful - TC-CS-01 - See all existing subscription boxes
failed - TC-CS-02 - Sort boxes based on base price
successful - TC-CS-03 - See an individual box's details
successful - TC-CS-04 - See the fluctuating price of any box
successful - TC-CS-05 - Continue to checkout
successful - TC-CS-06 - Continue to checkout without month/amount selection
successful - TC-CS-07 - Give personal information for checkout
successful - TC-CS-08 - Give personal information for checkout fail
successful - TC-CS-09 - See information on about us page
successful - TC-CS-10 - Create an account
successful - TC-CS-11 - Create an account with existing username
successful - TC-CS-12 - Create an account during checkout
successful - TC-CS-13 - Receive confirmation about order
failed - TC-CS-14 - Receive email confirmation about order
successful - TC-CS-15 - Login to the application
successful - TC-CS-16 - Login to the application incorrect details
successful - TC-CS-17 - See order history

successful - TC-S-01 - See amount of ordered boxes
successful - TC-S-02 - See amount of ordered boxes for amount of time
successful - TC-S-03 - See shipping details per order
successful - TC-S-04 - See additional notes added to order
successful - TC-S-05 - See a flag indicating pack/ship urgency

E2E testing

The login tests include 3 tests. These are a admin login, a login as user and a login to a non-existing account that checks for the correct BE response

< Tests	✓ 3	✗ --	○ --	04.39	● ↓ ↺
cypress/integration/E2E/smoke-it/LoginTest.js					
▼ Login test					
✓ successfull login admin					
✓ successfull login user					
✓ failed login					

The registration tests consists of 2 tests. One test the successful creation of an account. The other on tries to create the same account again, resulting in an error. This error is checked for

< Tests	✓ 2	✗ --	○ --	05.29	● ↓ ↺
cypress/integration/E2E/smoke-it/RegistrationTest.js					
▼ Registration test					
✓ successfull registration for user					
✓ failed registration for user (same username)					

The box tests test the basic CUD operations for the boxes.

< Tests	✓ 3	✗ --	○ --	08.94	● ↓ ↺
cypress/integration/E2E/smoke-it/BoxTest.js					
▼ Box CRUD test					
✓ Add new box					
✓ update existing box					
✓ delete existing box					

The order tests include 3 tests. One of these creates a new order. The second one does the same, but uses the login option that the checkout page offers. The last one packs and ships an order from the admin portal.

< Tests	✓ 3	✗ --	○ --	19.09	● ↓ ↺
cypress/integration/E2E/smoke-it/OrderTest.js					
▼ Order test					
✓ create new order					
✓ create new order w/ login redirect					
✓ pack/send order					

Version history

When?	What?
14/12/2021	First draft of document
17/12/2021	Added postman screenshots